NEAR EAST UNIVERSITY

GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

ARABIC CHARACTER RECOGNITION USING NEURAL NETWORKS

Qais ALBENI

Master Thesis

Department of Computer Engineering

Nicosia – 2005

Qais Albeni : **Arabic Character Recognition Using Neural Networks**

Approval of the Graduate School of Applied and Social Sciences

Prof. Dr. Fakhraddin Mamedov
Director

We certify this thesis is satisfactory for the award of the Degree of Master of Science in Computer Engineering

Examining Committee in charge:

Assoc. Prof. Dr. Adnan Khashman, Chairman, Chairman of Electrical and Electronic Engineering Department, NEU

Assist Prof. Dr Firudin Muradov, Member, Computer Engineering Department, NEU

Assist. Prof. Dr. Adil Amirjanov, Member, Computer Engineering Department, NEU

Assoc. Prof. Dr. Rahib Abiyev, Supervisor, Computer Engineering Department, NEU

ACKNOWLEDGMENTS

First I thank Allah for guiding me and taking care of me all the time, my life is so blessed because of him.

I also would like to take this opportunity my thesis supervisor, Assoc. Prof. Dr. Rahib Abiyev for giving me the opportunity to work with him and guided and helping me throughout my thesis and other courses.

Thank you M. Ruzieh and G. Alhafiz for being my best friends and for and for the comfort and support, also thanks B. Sekeroglu, K. Dimililer, and C. Kavalcioglu for comfort and support, and I would like to thank everyone I met at N.E.U,

Finally, I would like to dedicate this thesis to my parents and the rest of my family, they always supported me and to them I owe everything.

ABSTRACT

Character recognition plays an important role in the modern world. It can solve more complex problems and makes human's job easier. Character recognition has many applications. It can widely used to recognize zip code or postal code for mail sorting.

There are different techniques that can be used to recognize characters. In this thesis the used methods for recognition of characters are discussed. It is given that one of the efficient methods for character recognition is artificial neural network (ANN). Neural networks have such characteristics as: vitality, parallelism of computations, learning and generalization abilities, analytic description of linear and non-linear problems etc. Due to these characteristics neural network becomes great of importance for application in such areas such as artificial behavior, artificial intelligence, pattern recognition, theory of control and decision making, identification, optimal control, robotics etc.

In this thesis using neural network the recognition of Arabic characters is considered. Some of Arabic characters have four different ways of writing. In this thesis the steps of Arabic character recognition have been explained. The feed-forward neural network is applied for character recognition. The simulation of the system is carried out using MATLAB package. Neural network is used to train and identify written digits. The training is carried out using back-propagation algorithm. After training and testing, recognition rate reached 98%. The developed character recognition system is tested for two types of noise that separately added to the characters and non-noisy characters.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Pattern recognition is becoming more and more important in the modern world. It helps human ease their jobs and solve more complex problems. As an example of character recognition application is handwritten character recognition. Handwritten character recognition is the widely used in our world. Such system is developed for zipcode or postal code that can be employed in mail sorting. This can help humans to sort mails with postal code that are difficult to identify.

For more than thirty years, researchers have been working on character recognition. Over the past few years, the number of companies involved in research on character recognition has continually increased. The advance of character processing results from a combination of various elements, for example: improvements in the recognition rate, the used of complex system to integrate various kinds of information, and new technologies such as high quality, high speed scanner and a cheaper and more powerful CPUs.

Some character recognition systems allow us to input the characters into the system. This can be done either by controlling a mouse or using a third-parity drawing tablet or input scanned images.

Character recognition is not a new technology; the ultimate goal of designing a character recognition system with an accuracy rate of 100% is quite illusionary, because even human beings are not able to recognize every written text without any doubt. For example, most people can not even read their own note.

The work of recognizing Arabic characters is still limited. The importance of recognizing Arabic characters is also to be considered by Arabic non-speaking people such as Farsi, Kurds, Persians, and Urdu-speaking who use the Arabic characters in writing although the pronunciation is different.

The aim of this thesis is the development of a neural system for recognition of Arabic characters. Thesis includes four chapters, conclusion, and appendices.

In first chapter pattern recognition, pattern classification and a group of important application, how pattern recognition is done by human, and how machines recognize pattern, the stages of the pattern recognition problem are described. Character recognition problem is described.

Chapter two describe the artificial neural network and how the neural structure works, the major and necessary component of the neural nets, the activation function of the neural network, about the models and architecture of the neural system, and how it is possible to train and teach the neural system.

In the third chapter written languages and the nature of Arabic language and the Arabic language characteristics, the steps for recognition of characters using MATLAB and stages of Arabic character recognition are described. Artificial neural networks, using back-propagation method, will be used to train and identify Arabic character.

The recognition of noisy character has been applied. Two types of noise were added separately to the character that are used for testing, salt & pepper and spackle noise were both are added in order to test the systems efficient in recognition.

Conclusion contains results obtained from the thesis, appendices includes list of Arabic characters, listing of programs written for Arabic character recognition.

CHAPTER ONE
REVIEW ON PATTERN RECOGNITION

1.1 Overview

This chapter presents an introduction about the pattern recognition. Stages of Pattern recognition, pattern recognition application, in particularly character recognition and Neural network Application is explained.

1.2 Pattern Recognition

By the time they are five years old, most children can recognize digits and letters. Small characters, large characters, handwritten, machine printed or rotated–all are easily recognized by the young. The characters may be written on a cluttered background, on crumpled paper or may even be partially occluded. We take this ability for granted until we face the task of teaching a machine how to do the same. Pattern recognition is the study of how machines can observe the environment, learn to distinguish patterns of interest from their background, and make sound and reasonable decisions about the categories of the patterns.

The best pattern recognizers in most instances are humans, yet we do not understand how humans recognize patterns, where artificial intelligence came, since computers can surely be taught to recognize patterns. Indeed, successful computer programs that help banks score credit applicants, help doctors diagnose disease and help pilots land airplanes depend in some way on pattern recognition.

Automatic (machine) recognition, description, classification, and grouping of patterns are important problems in a variety of engineering and scientific disciplines such as biology, psychology, medicine, marketing, computer vision, artificial intelligence, and remote sensing. There are different kinds of patterns. A pattern could be a fingerprint image, a handwritten cursive word, a human face, or a speech signal. Given a pattern, its recognition/classification may consist of one of the following two tasks:

1. supervised classification (e.g., discriminate analysis) in which the input pattern is identified as a member of a predefined class,
2. Unsupervised classification (e.g., clustering) in which the pattern is assigned to a hitherto unknown class [1].

1.3 Pattern Recognition by Human

Human vision involves the analysis of image sequences with the purpose of recognizing objects, calculating the shape and physical properties of surfaces, determining position and motion relationships of rigid and deformable bodies in a scene. The following four stages in pattern recognition by human beings:

- Early vision: This is the first stage of visual processing. Early visual processes compute elementary properties of images such as brightness, texture, color, motion flow, and stereo disparity. The gradient of these quantities is then used to segment the image into its component regions.
- Autonomous navigation: From sequences of images the human brain autonomously calculates motion parameters and scene structure.
- Shape reconstruction: The brain also calculates the 3D shape of objects from 2D images using a number of visual cues: stereoscopy, texture, motion parallax, shading, boundaries.
- Visual pattern recognition: The human visual system can detect and recognize complex visual patterns with a minimal amount of training. It is believed that trained information is stored in the synapse of the brain's neuron.

1.4 Pattern Recognition in Machine

Machine recognizes pattern quite differently form human beings. To a machine, a pattern in the form of an image is just an unrelated collection of pixels (picture elements). The machine has no understanding of the image. It can store the image as the collection of pixels, or the image can be subjected to a transformation function whose output would be stored instead. Recognition would be based on comparison and retrieval from stored data in a knowledge database.

1.4.1 Modeling of Human Recognition by Machine

One of the main differences between the way human beings perceive the world and how the machine does it, is that human beings are less 'exact' than a machine. For example when shown "A" letter, a human being will deduce the type of the object as "A". To the machine, the letter will be encoded with a fixed number that represent the letter "A". If presented with a letter that is somewhat "a", a human being can still associate the letter as "A" without much effort. But to a machine, the new letter will be encoded with a number quite different from that of "A".

To enable the machine to simulate human cognitive behavior, soft computing is introduced. Currently the main areas in soft computing are:

- Fuzzy logic is motivated by the way human beings describe the world. The attributes that human beings assign to an entity are normally discrete while a more continuous treatment would have been more accurate. For example, words like cold, warm and hot are used to describe temperature. The measure of temperature should have been continuous with a range from a fixed minimum to a fixed maximum. However, human beings are more comfortable with working with finite discrete classes than with a continuous infinite quantum. Fuzzy logic introduces discrete fuzziness into computer by making it work and reason with fuzzy components. A temperate of 35.7° C would be considered as 25% cold, 75% warm and 10 % hot.
- Neural network is motivated by the findings of the working of neurons in the human brain. It is believed that knowledge is stored in the brain's neurons as complex and intricate network of connections (synapses) between neurons. As more knowledge is acquired, the arrangement of the synapses changes. A single piece of information is stored by not a single neuron but by probably thousands of them, each with a small contribution. This makes it possible for a brain tumor patient to continue to function even after a brain surgery that removes some part of his brain.

  Implementation of neural computing in computer is by the use of data structure that simulates the network of neurons. The synapses are simulated as small weights in 2D arrays. The effect of training is to alter the content of such arrays

of weights so that it will eventually reflect the characteristics of the trained input patterns. This approach is in contrast with traditional computing where information is normally stored in a knowledge database file. Information retrieval in a database approach is based on record searching. The system can only retrieve data that has been previously stored. In the neural network approach, the system can approximate a result even when the input is not one of those trained. This is a typical human characteristic and is very useful for pattern recognition where it is not possible to train all variations of the input.

- Genetic algorithm takes the machine one step further. Inspired by the Darwinian evolution process, a population of structures is maintained. As the system interacts with the outside world, not only learning takes place, but there will also be a change in the data structure to best facilitate the learning process [2].

1.5 Stages in a Pattern Recognition Problem

A pattern recognition investigation may consist of several typical stages, enumerated below. Not all stages may be present; some may be merged together; also, there may be some application-specific data processing that may not be regarded as one of the stages listed.

- Formulation of the problem: gaining a clear understanding of the aims of the investigation and planning the remaining stages.
- Data collection: making measurements on appropriate variables and recording details of the data collection procedure (ground truth).
- Initial examination of the data: checking the data, calculating summary statistics and producing plots in order to get a feel for the structure.
- Feature selection or feature extraction: selecting variables from the measured set that are appropriate for the task. These new variables may be obtained by a linear or nonlinear transformation of the original set (feature extraction). To some extent, the division of feature extraction and classification is artificial.
- Pattern classification or clustering. This may be viewed as exploratory data analysis and it may provide a successful conclusion to a study. On the other hand, it may be a means of preprocessing the data for a supervised or unsupervised classification procedure.

- Apply discrimination or regression procedures as appropriate. The classifier is designed using a training set of exemplar patterns.
- Assessment of results. This may involve applying the trained classifier to an independent *test set* of labeled patterns [3].

1.6 Pattern Recognition Application

The primary applications of Pattern Recognition in widespread use today are face, voice, fingerprint, signature, ear, DNA, character, and iris recognition.

- Face Recognition: Facial recognition works by taking many images or pictures of the face, and extracting the unique facial features, such as the ears, nose, eyes, mouth, etc. There are a number of shortcomings with facial recognition. For example, suppose an image is taken of your face. Then, you put on dark sunglasses and a hat. There is a high probability that the facial recognition system will not recognize you in the second set of circumstances. Also, changes in the environment can also have a negative impact (such as changes in lighting, differences in the background setting, etc.) Finally, the task of identifying people by face recognition can be easily divided into two sub-tasks. Firstly, there is the task of recognizing a face in a scene and extracting it from the surrounding "noise." Secondly, a computer system must then be able to extract unique features from that image and compare them with images already stored (see Figure 1.1).

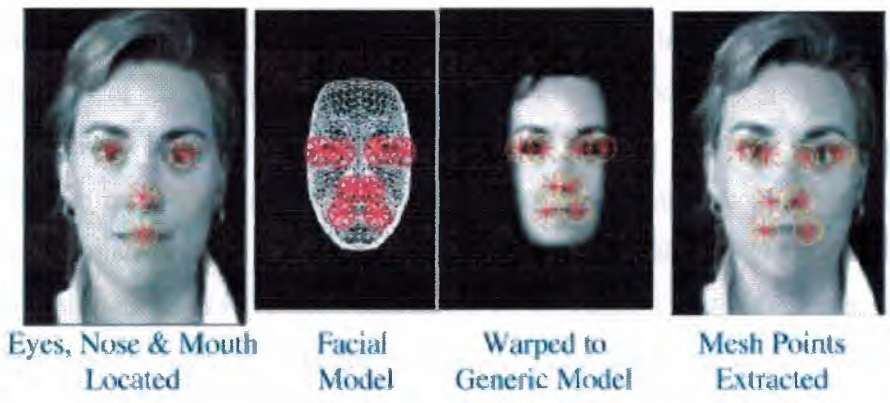


Figure 1.1 Face Recognition

- Voice Recognition: Voice recognition works by first having the subject recite a text phrase numerous times, and then extracting the unique voice inflections. The most popular environment for voice recognition systems are for telephony based verification applications (for example, when you call your credit card company, you normally get an automated response. Rather than dialing in your Social Security number to access your account, voice recognition technology will automatically verify you). Voice recognition also has a number of disadvantages, such as extraneous, outside noises when a user is in the process of verification, or the user themselves has physical ailment (such as a cold or flu) which could affect the quality of the voice sample.
- Fingerprint recognition: Fingerprint recognition is probably the most popular and widely used of the recognition technologies. The minutiae features (such as the valleys and ridges) of the fingerprint are extracted by the fingerprint scanner, as well as other unique characteristics. In almost all scenarios, it is the index finger that is scanned for enrollment and verification. However, most fingerprint systems of today allow for multiple fingers to be enrolled and verified. Fingerprint recognition has disadvantages, particularly if the fingerprint from which the sample is to be derived is damaged in a particular way, or is contaminated by dirt.
- Signature Recognition: Signature Recognition examines the unique way in which we sign our name. it is not the way the signature looks that is captured, but rather the various behavioral characteristics (timing, speed, and pressure) when signing our name that are used to create the enrollment and verification templates. An advantage of Signature Recognition is that is it has a high level of resistance to impostors. A disadvantage is that this technology can have high levels of error rates. Signature Recognition has been used to a certain extent by the financial sector.
- Ear Recognition: The goal of Ear Recognition is to examine the unique features of the geometry of the earlobe, in a manner similar to that of Hand Geometry Recognition. Studies have been conducted which suggest that the earlobe can be unique among groups of people. However, Ear Recognition is still in the extreme, beginning phases of research and development. Many of the concepts are still in theory, with scientific studies, still in the preliminary works. However,

there have been movements to examine the feasibility of Ear Recognition. One such movement is known as the Forensic Ear Identification research group (FEARID).

- DNA recognition: The use of DNA has gained much interest as a means to verify and identify individuals. DNA stands for deoxyribonucleic acid, and is found in all living cells, and is unique among individuals. DNA has been used primarily in forensics to identify criminals, as well as exclude criminal suspects. It still has a long way to go before it is used in commercial applications like the other recognition technologies, in terms of rapid, automated verification and identification. It can take up to weeks right now to conduct DNA analysis. At the present time, the use of DNA differs from the other recognition in three areas:
  - Actual samples of DNA are used, thus there are no templates created;
  - While other biometrics are analyzed and compared in real time, DNA is not;
  - No images of DNA are taken because actual samples are used.
- Character Recognition: character recognition is probably one of the most popular and widely used of the recognition technologies. The features of the character are extracted by the scanner. In almost all scenarios, the characters are the result of the words or text segmentation. However, most character recognition systems of today allow multiple languages to be verified. Character recognition has disadvantages, particularly if the character pattern is damaged in a particular way that will affect the accuracy of the system (Figure 1.2).

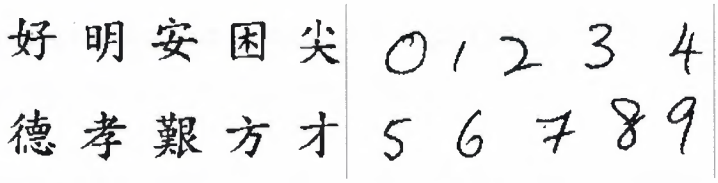


Figure 1.2 Character Recognition

- Iris and Retinal Recognition: Iris recognition involves examining the unique features in the texture and patterns of the iris, which is located in the front of the eye. Retinal recognition involves examining the pattern of blood vessels in the retina, which is located in the back of the eye, at the juncture of the optic nerve.

Both of these recognition technologies are deemed to be among the most reliable and stable.

This is so because the structure of the iris and the blood vessel pattern in the retina hardly change over the lifetime of the user, unless they have a disease of the eye. Iris scanning has been used widely at airports around the world. Retinal recognition is primarily used for extremely high security applications (See Figure 1.3), such as for physical access entry into government and military installations, etc [4].

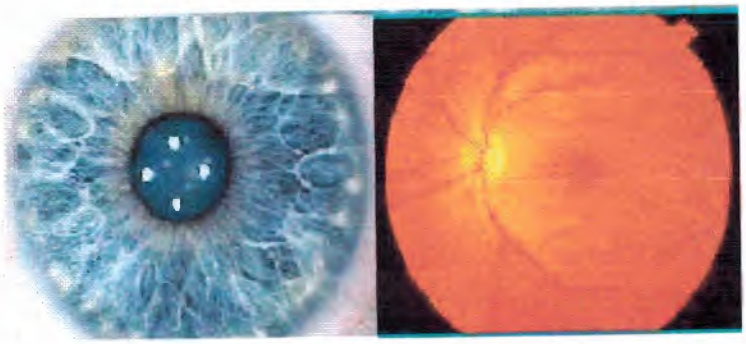


Figure 1.3 Iris and Retina.

1.7 State of Art Understanding of Character Recognition Problem

Much work has been done in character recognition. The word recognition rates for unconstrained handwriting recognition systems can range from about 50%, to more than 90%. Due to differing data sets and varying sizes [5],

A system for recognizing unconstrained Turkish handwritten text is described in [5]. Turkish has agglutinative morphology and theoretically an infinite number of words that can be generated by adding more suffixes to the word. This makes lexicon-based recognition approaches, where the most likely word is selected among all the alternatives in a lexicon, unsuitable for Turkish.

We describe our approach to the problem using a Turkish prefix recognizer. First results of the system demonstrates the promise of this approach, with top-10 word recognition

rate of about 40% for a small test data of mixed handprint and cursive writing. The lexicon-based approach with a 17,000 word-lexicon (with test words added) achieves 56% top-10 word recognition rate.

In [6] the technique is proposed for recognizing Arabic characters [6]. This technique involves of three parts: body classifier, complementary classifier, and aggregate classifier. The body classifier is designed to recognize the main body of the unknown character. It uses a Hopfield network to enhance the unknown character and to get rid of noise and associated complementary. Furthermore, it uses a back-propagation network to recognize the main body of the enhanced unknown character. The complementary classifier is responsible of recognizing the number of dots or zigzag that are associated with the body of character and their position. The aggregate classifier combines the results of the previous two classifiers and classifies the whole unknown character. The proposed technique had been implemented and had shown a reasonable recognition rate [6].

Neural network classifiers with single-layer training can be applied efficiently to complex real-world classification problems such as the recognition of handwritten digits [7]. The STEPNET procedure is introduced, which decomposes the problem into simpler sub problems which can be solved by linear separators. Provided appropriate data representations and learning rules are used, performances which are comparable to those obtained by more complex networks can be achieved. We present results from two different data bases: a European data base comprising 8,700 isolated digits and a zip code data base from the U.S. Postal Service comprising 9,000 segmented digits. A hardware implementation of the classifier is briefly described.

The research that use geometrical features and neural networks to automatically recognize (read) off-line handwritten Arabic words is given in [8]. The nature of handwritten Arabic characters and hence the problems that could be faced when automatically (optically) recognizing them are discussed. This research concentrates on the feature extraction process, i.e. extraction of the main geometrical features of each of the extracted handwritten Arabic characters. A complete system able to recognize Arabic-handwritten characters of only a single writer is proposed and discussed. A

review of some of the previous trials in the field of off-line handwritten Arabic character recognition is included.

The system first attempts to remove some of the variations found in the images that do not affect the identity of the handwritten word (slant correction, slope correction, and baseline estimation). Next, the system codes the skeleton of the word so that feature information about the lines in the skeleton is extracted (segmentation and feature extraction). The features include locating endpoints, junctions, turning points, loops, generating frames (segmentation step), and detecting strokes. These features are then passed on to the recognition system for recognition. The character classification is achieved in this research using a feed forward error back propagation neural network. A 69.7% recognition rate has been achieved for the character frames of data.

The review on the state of the art in off-line Roman cursive handwriting recognition is given [9]. The input provided to an off-line handwriting recognition system is an image of a digit, a word, or - more generally – some text and the system produce, as output, an ASCII transcription of the input. This task involves a number of processing steps, some of which are quite difficult. Typically, preprocessing, normalization, feature extraction, classification, and post processing operations are required. We'll survey the state of the art, analyze recent trends, and try to identify challenges for future research in this field.

In [10] a new method on off-line recognition of handwritten Arabic script is presented. The method does not require segmentation into characters, and is applied to cursive Arabic script, where ligatures, overlaps and style variation pose challenges to the recognition system. The method trains a single hidden Markov model (HMM) with the structural features extracted from the manuscript words. The HMM is composed of multiple character models where each model represents one letter from the alphabet. The performance of the proposed method is assessed using samples extracted from a historical handwritten manuscript.

Binary weights are favored in electronic and optical hardware implementations of neural networks as they lead to improved system speeds. Optical neural networks based on fast ferroelectric liquid crystal binary level devices can benefit from the many orders of magnitudes improved liquid crystal response times. An optimized learning algorithm

for all-positive perceptrons is simulated on a limited data set of hand-written digits and the resultant network implemented optically. First, gray-scale and then binary inputs and weights are used in recall mode. On comparing the results for the example data set, the binarized inputs and weights network shows almost no loss in performance [11].

In [12] an effort towards the development of Arabic cheque databases for research in the recognition of hand-written Arabic cheques is described. Databases of real-life Arabic legal amounts, Arabic sub-words, courtesy amounts, Indian digits, and Arabic cheques are described. This paper highlights some characteristics of the Arabic language and presents the various steps that have been completed to build these databases including segmentation, binarization and data tagging. It also describes a solid validation procedure including grammars and algorithms used to verify the correctness of the tagging process. Detailed descriptions of the database organization and class distribution are included. These databases aim to facilitate experimental comparisons between various recognition methods, and will be provided to all interested researchers upon request to CENPARMI.

One of efficient method for character recognition is neural network. The applications are expanding because neural networks are good at solving problems, not just in engineering, science and mathematics, but in medicine, business, finance and literature as well. Their application to a wide variety of problems in many fields makes them very attractive. Also, faster computers and faster algorithms have made it possible to use neural networks to solve complex industrial problems that formerly required too much computation.

A list of some applications mentioned in the literature follows:

- Aerospace: High performance aircraft autopilots, flight path simulations, aircraft control systems, autopilot enhancements, aircraft component simulations, aircraft component fault detectors
- Automotive: Automobile automatic guidance systems, warranty activity analyzers
- Banking: Check and other document readers, credit application evaluators

- Defense: Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/image identification
- Electronics: Code sequence prediction, integrated circuit chip layout, process control, chip failure analysis, machine vision, voice synthesis, nonlinear modeling
- Financial: Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit line use analysis, portfolio trading program, corporate financial analysis, currency price prediction
- Insurance: Policy application evaluation, product optimization
- Manufacturing: Manufacturing process control, product design and analysis, process and machine diagnosis, real-time particle identification, visual quality inspection systems, beer testing, welding quality analysis, paper quality prediction, computer chip quality analysis, analysis of grinding operations,
- chemical product: design analysis, machine maintenance analysis, project bidding, planning and management, dynamic modeling of chemical process systems
- Medical: Breast cancer cell analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement, emergency room test advisement
- Robotics: Trajectory control, forklift robot, manipulator controllers, vision systems
- Speech: Speech recognition, speech compression, vowel classification, text to speech synthesis
- Securities: Market analysis, automatic bond rating, stock trading advisory systems
- Telecommunications: Image and data compression, automated information services, real-time translation of spoken language, customer payment processing systems [13].

A hand-written Arabic character recognition system using storke-based spartial and temporal features with two-stage classification techniques is proposed in [14]. This

classification Techniques is comprised of K-mean algorithm followed by a rule-base classification, and the system is intended to be used for online user-dependent application in portable devices with limited computational and memory resources such as PDAs. We have integrated this recognition system into a PDA and obtained an average user-dependent recognition rate of 95%.

1.9 Summary

In this chapter, the pattern recognition background and necessary information were explained. In the next chapter, the parts of neural networks (structure, Mathematical, etc) that are used in the application part of this thesis will be studied

CHAPTER TWO
NEURAL SYSTEM FOR CHARACTER RECOGNITION

2.1 Overview

This chapter presents an introduction about the neural networks, biological neural networks, Structure and mathematical modeling (neuron models, major components of artificial neurons, and Network Architectures), and learning of Neural System (learning laws and methods -supervised and unsupervised training-) will be explained.

2.2 Artificial Neural Networks

Neural networks have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced [13].

The neurons are transporting incoming information on their outgoing connections to other neurons. In neural net terms these connections are called weights. The information is simulated with specific values stored in those weights. By simply changing these weight values the changing of the connection structure can also be simulated (See Figure 2.1).

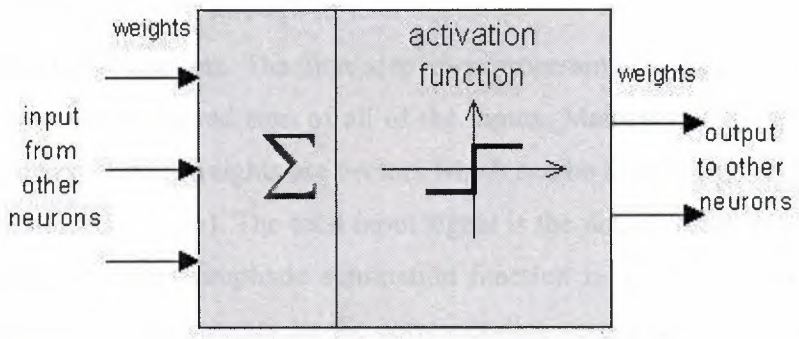


Figure 2.1 Structure of a Neuron in a Neural Net

Information (called the input) is sent to the neuron on its incoming weights. This input is processed by a propagation function that adds up the values of all incoming weights. The resulting value is compared with a certain threshold value by the neuron's activation function. If the input exceeds the threshold value, the neuron will be activated, otherwise it will be inhibited. If activated, the neuron sends an output on its outgoing weights to all connected neurons and so on [15].

2.2.1 Major Component of Artificial Neural Networks

These are the seven major components which make up an artificial neuron. These components are valid whether the neuron is used for input, output, or is in one of the hidden layers.

- Weighting Factors: A neuron usually receives many simultaneous inputs. Each input has its own relative weight which gives the input the impact that it needs on the processing element's summation function. These weights perform the same type of function as do the varying synaptic strengths of biological neurons. In both cases, some inputs are made more important than others so that they have a greater effect on the processing element as they combine to produce a neural response. Weights are adaptive coefficients within the network that determine the intensity of the input signal as registered by the artificial neuron. They are a measure of an input's connection strength. These strengths can be modified in response to various training sets and according to a network's specific topology or through its learning rules.
- Summation Function: The first step in a processing element's operation is to compute the weighted sum of all of the inputs. Mathematically, the inputs and the corresponding weights are vectors which can be represented as (i1, i2 . . . i*n*) and (w1, w2 . . . w*n*). The total input signal is the dot, or inner, product of these two vectors. This simplistic summation function is found by multiplying each component of the i vector by the corresponding component of the w vector and then adding up all the products. Input1 = i1 * w1, input2 = i2 * w2, etc., are added as input1 + input2 + . . . + input*n*. Some summation functions have an additional process applied to the result before it is passed on to the transfer function. This process is sometimes called the activation function [16].

- Transfer Function: The transfer function may be a linear or a nonlinear function of *n*. A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve. Two of the most commonly used functions are discussed below. The *hard limit transfer function*, shown on the left side of Figure 2.2, sets the output of the neuron to 0 if the function argument is less than 0, or 1 if its argument is greater than or equal to 0. The use this function to create neurons that classify inputs into two distinct categories.

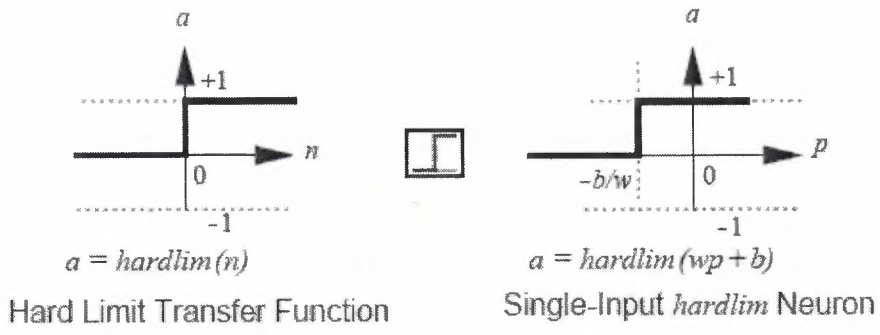


Figure 2.2 Hard Limit Transfer Function

The graph on the right side of Figure 2.2 illustrates the input/output characteristic of a single-input neuron that uses a hard limit transfer function. Here we can see the effect of the weight and the bias.

The *log-sigmoid transfer function* is shown in Figure 2.3.

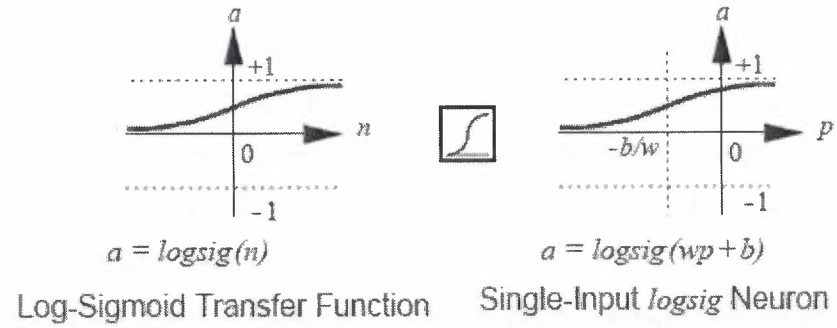


Figure 2.3 Log-Sigmoid Transfer Function

This transfer function takes the input (which may have any value between plus and minus infinity) and squashes the output into the range 0 to 1, according to the expression:

$$a = \frac{1}{1 + e^{-n}}$$

The log-sigmoid transfer function is commonly used in multilayer networks that are trained using the back propagation algorithm, in part because this function is differentiable [13].

Most of the used transfer function is summarized in Table 2.1.

Table 2.1 Transfer Functions

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	a = 0, n < 0 a = 1, n ≥ 0		hardlim
Symmetrical Hard Limit	a = -1, n < 0 a = +1, n ≥ 0		hardlims
Linear	a = n		purelin
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$		logsig

- Scaling and Limiting: After the processing element's transfer function, the result can pass through additional processes which scale and limit. This scaling simply multiplies a scale factor times the transfer value, and then adds an offset. Limiting is the mechanism which insures that the scaled result does not exceed an upper or lower bound. This limiting is in addition to the hard limits that the original transfer function may have performed.
- Output Function (Competition): Each processing element is allowed one output signal which it may output to hundreds of other neurons. Normally, the output is directly equivalent to the transfer function's result. Neurons are allowed to

compete with each other, inhibiting processing elements unless they have great strength. Competition can occur at one or both of two levels. First, competition determines which artificial neuron will be active, or provides an output. Second, competitive inputs help determine which processing element will participate in the learning or adaptation process.

- Error Function and Back-Propagated Value: In most learning networks the difference between the current output and the desired output is calculated. This raw error is then transformed by the error function to match particular network architecture. The most basic architectures use this error directly, but some square the error. The artificial neuron's error is then typically propagated into the learning function of another processing element. This error term is sometimes called the current error. The current error is typically propagated backwards to a previous layer. Normally, this back-propagated value, after being scaled by the learning function, is multiplied against each of the incoming connection weights to modify them before the next learning cycle.
- Learning Function: The purpose of the learning function is to modify the variable connection weights on the inputs of each processing element according to some neural based algorithm. This process of changing the weights of the input connections to achieve some desired result can also be called the adaption function, as well as the learning mode. There are two types of learning: supervised and unsupervised which i will explain later on [16].

2.3 Neuron Models

The scalar input *p* is multiplied by the scalar *weight w* to form *wp*, one of the terms that are sent to the summer. The other input, is multiplied by a (offset) *bias b* and then passed to the summer. The summer output *n*, often referred to as the *net input,* goes into a (activation function) *transfer function f,* which produces the scalar neuron output, If we relate this simple model back to the biological neuron the weight corresponds to the strength of a synapse, the cell body is represented by the summation and the transfer function, and the neuron output *a* represents the signal on the axon (see Figure 2.4).

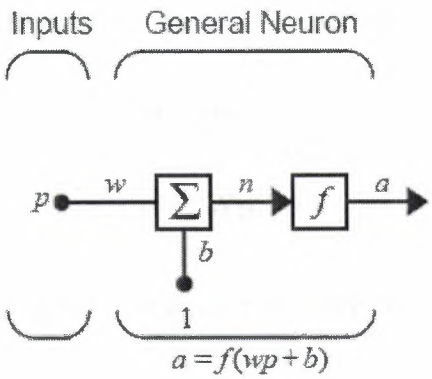


Figure 2.4 Single-Input Neuron

The neuron output is calculated as

$$a = f(wp + b)$$

The actual output depends on the particular transfer function that is chosen. We will discuss transfer functions in the next section. The bias is much like a weight, except that it has a constant input of 1.

Typically, a neuron has more than one input. A neuron with R inputs is shown in Figure 2.5. The individual inputs are $p_1, p_2, \ldots, p_R$ each weighted by corresponding $w_{1,1}$, $w_{1,2}$, ... ,$w_{1,R}$.elements of the *weight matrix W.*

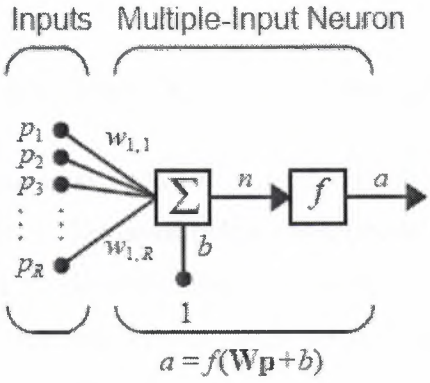


Figure 2.5 Multiple-Input Neuron

The neuron has a bias, which is summed with the weighted inputs to form the net input:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_R + b.$$

This expression can be written in matrix form:

$$n = \mathbf{W}\mathbf{p} + b$$

Where the matrix W for the single neuron case has only one row, now the neuron output can be written as

$$a = f(\mathbf{W}\mathbf{p} + b)$$

We would like to draw networks with several neurons, each having several inputs. Further, we would like to have more than one layer of neurons. A multiple-input neuron using this notation is shown in Figure 2.6.

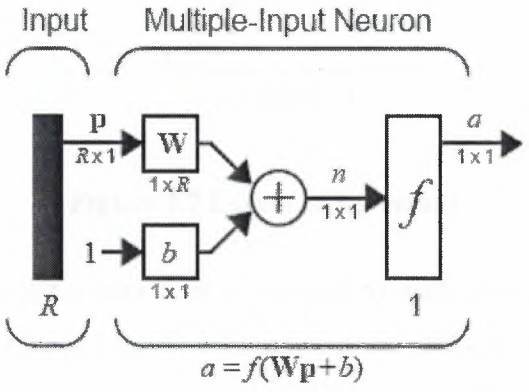


Figure 2.6 Neuron with *R* Inputs, Abbreviated Notation

As shown in Figure 2.6, the input vector *P* is represented by the solid vertical bar at the left, the dimensions of *P* are displayed below the variable as *R x 1*, indicating that the input is a single vector of *R* elements. These inputs go to the weight matrix W, which has *R* columns but only one row in this single neuron case. A constant 1 enters the neuron as an input and is multiplied by a scalar bias *b*. The net input to the transfer function *f* is *n*, which is the sum of the bias *b* and the product *Wp*. The neuron's output is a scalar in this case. If we had more than one neuron, the network output *a* would be a vector [13].

2.4 Network Architectures

A single-*layer* network of S neurons is shown in Figure 2.7. Note that each of the *R* inputs is connected to each of the neurons and that the weight matrix now has *R* rows.

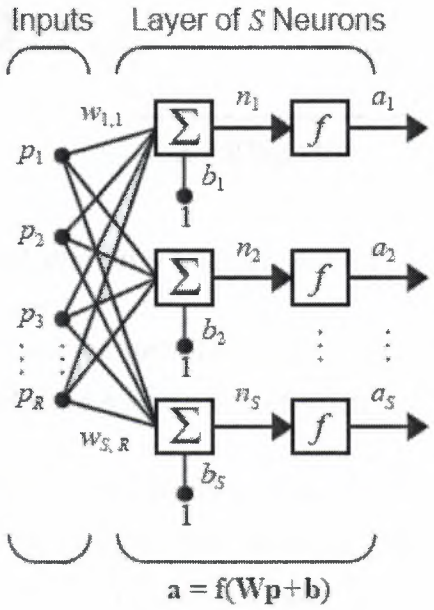


Figure 2.7 Layers of *S* Neurons

Each element of the input vector *p* is connected to each neuron through the weight matrix *W*. Each neuron has a bias b_i, a summer, a transfer function *f* and an output a_i. Taken together, the outputs form the output vector *a, t*he input vector elements enter the network through the weight matrix *W*:

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

The *S*-neuron, *R*-input, one-layer network also can be drawn in abbreviated notation, as shown in Figure 2.8.

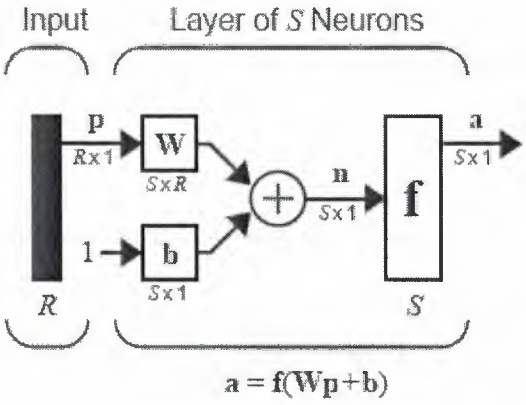


Figure 2.8 Layers of S Neurons, Abbreviated Notation.

Here again, the symbols below the variables tell you that for this layer, *P* is a vector of length *R*, *W* is a matrix, and a and *b* are vectors of length *S*.

Now consider a network with several layers. Each layer has its own weight matrix *W*, its own bias vector *b*, a net input vector *n* and an output vector *a*, the weight of the first layer is written as W^1, and the weight matrix of then second layer is written as W^2, this notation is used in the three-layer network shown in Figure 2.9.

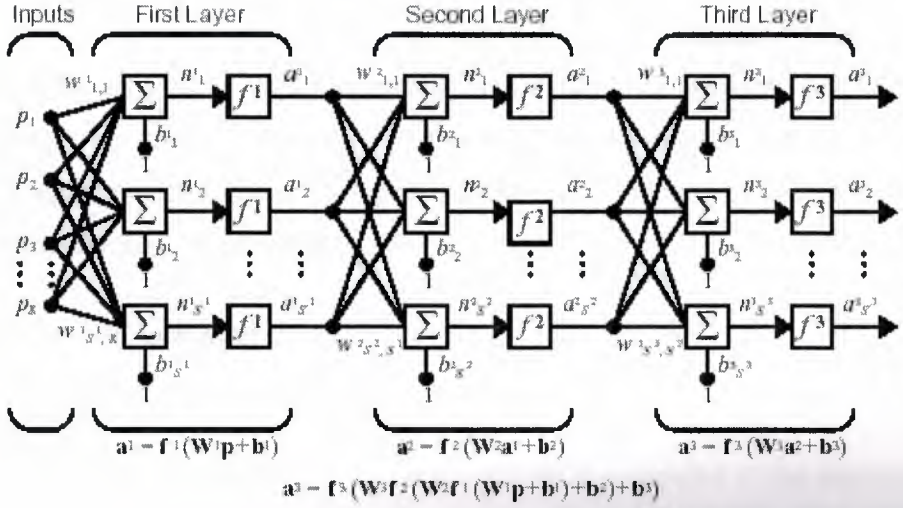


Figure 2.9 Three-Layer Network

As shown, there are *R* inputs, S1 Neurons in the first layer, S2 neurons in the second layer, etc. Thus different layers can have different numbers of neurons. Note that the

multilayer networks are more powerful than single layer network. The same three-layer network discussed previously also can be drawn using our abbreviation notations shown in Figure 2.10 [13].

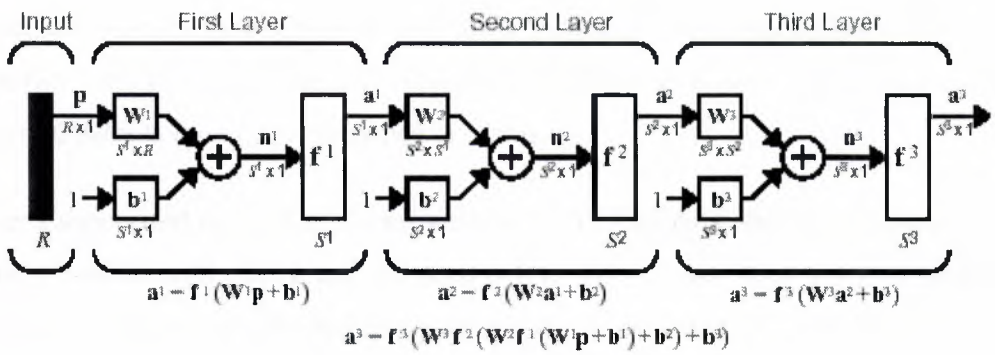


Figure 2.10 Three-Layer Network, Abbreviation Notation.

2.5 Training an Artificial Neural Network

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins. There are two approaches to training - supervised and unsupervised.

Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help.

2.5.1 Supervised Learning

The majority of artificial neural network solutions have been trained with supervision. In this mode, the actual output of a neural network is compared to the desired output. Weights, which are usually randomly set to begin with, are then adjusted by the network so that the next iteration, or cycle, will produce a closer match between the desired and the actual output. The learning method tries to minimize the current errors of all processing elements. This global error reduction is created over time by continuously modifying the input weights until acceptable network accuracy is reached. With

supervised learning, the artificial neural network must be trained before it becomes useful. Training consists of presenting input and output data to the network. This data is often referred to as the training set. That is, for each input set provided to the system, the corresponding desired output set is provided as well. This training is considered complete when the neural network reaches a user defined performance level. This level signifies that the network has achieved the desired statistical accuracy as it produces the required outputs for a given sequence of inputs.

After a supervised network performs well on the training data, then it is important to see what it can do with data it has not seen before. If a system does not give reasonable outputs for this test set, the training period is not over. Indeed, this testing is critical to insure that the network has not simply memorized a given set of data but has learned the general patterns involved within an application.

2.5.2 Unsupervised Learning

Unsupervised learning is the great promise of the future. It shouts that computers could someday learn on their own in a true robotic sense. Currently, this learning method is limited to networks known as self organizing maps. These kinds of networks are not in widespread use. They are basically an academic novelty. Yet, they have shown they can provide a solution in a few instances, proving that their promise is not groundless. These networks use no external influences to adjust their weights. Instead, they internally monitor their performance. These networks look for regularities or trends in the input signals, and makes adaptations according to the function of the network. Even without being told whether it's right or wrong, the network still must have some information about how to organize itself. This information is built into the network topology and learning rules.

2.5.3 Learning Rates.

The rate at which ANNs learn depends upon several controllable factors. In selecting the approach there are many trade-offs to consider. Obviously, a slower rate means a lot more time is spent in accomplishing the learning to produce a convenient trained system. Generally, several factors besides time have to be considered when discussing the training task, which is often described as Network complexity, size, paradigm

selection, architecture, type of learning rule or rules employed, and desired accuracy must all be considered. These factors play a significant role in determining how long it will take to train a network. Changing any one of these factors may either extend the training time to an unreasonable length or even result in an unacceptable accuracy.

2.5.4 Learning Laws

Many learning laws are in common use. Most of these laws are some sort of variation of the best known and oldest learning law. Man's understanding of how neural processing actually works is very limited. Learning is certainly more complex than the simplifications represented by the learning laws currently developed. A few of the major laws are Hebb's rule, Hopfield Law, Delta Rule and Kohonen's learning rule [13].

2.6 Back-propagation Training Algorithm

The back-propagation algorithm is perhaps the most widely used training algorithm for multi-layered feed-forward networks. However, many people find it quite difficult to construct multilayer feed-forward networks and training algorithms, whether it is because of the difficulty of the math or the difficulty involved with the actual coding of the network and training algorithm.

Multilayer feed-forward networks normally consist of three or four layers, there is always one input layer and one output layer and usually one hidden layer, although in some classification problems two hidden layers may be necessary, this case is rare however. The term input layer neurons are a misnomer, no sigmoid unit is applied to the value of each of these neurons. Their raw values are fed into the input layer. Once the neurons for the hidden layer are computed, their activations are then fed to the next layer, until all the activations finally reach the output layer, in which each output layer neuron is associated with a specific classification category. In a fully connected multilayer feed-forward network (See Figure 2.11), each neuron in one layer is connected by a weight to every neuron in the previous layer. A bias is also associated with each of these weighted sums. Thus in computing the value of each neuron in the hidden and output layers one must first take the sum of the weighted sums and the bias and then apply f(sum) (the sigmoid function) to calculate the neuron's activation [4].

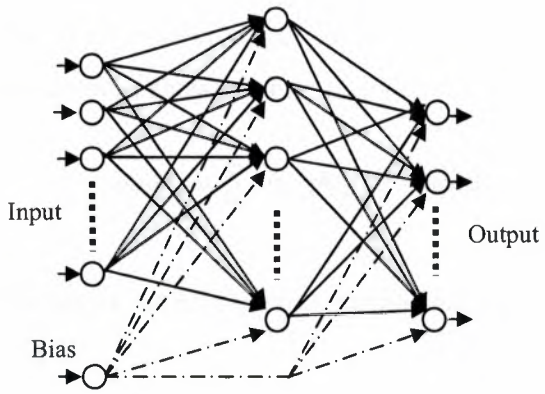


Figure 2.11 Multilayer Feed-Forward Network

2.6.1 Feed Forward Phase

The feed-forward phase can be described through three steps: input (I), hidden (H), and output layer (O).

- Input layer (I): The output of the input layer is equal to the input of the input layer as shown in Figure 2.11.

$$Output_I = Input_I$$

- Hidden Layer (H): The input of the hidden layer is equal to the sum of multiplying all the input layer output by the corresponding weight that connect between the two layers as shown in Figure 2.11.

$$Input_H = \sum_i weight_{IHi} * output_I$$

The output of the hidden layer is the result of the sigmoid transfer function of the hidden layer input.

$$Output_H = \frac{1}{1 + e^{-Input_H}}$$

- Output layer (O): The input of the Output layer is equal to the sum of multiplying all the hidden layer output by the corresponding weight that connect between the two layers as shown in Figure 2.11.

$$Input_O = \sum_j weight_{HOj} * output_H$$

The output of the output layer is the result of the sigmoid transfer function of the output layer input.

$$Output_O = \frac{1}{1 + e^{-Input_O}}$$

2.6.2 Backpropagation Phase

After the feed-forward phase the output of the output layer ($Output_O$), is compared with the target value of the neural net (Target), the result is the network error ($error_O$).

$$error_O = T\arg et - Ouput_O$$

The purpose of the back-propagation training is to minimize the error of all training pattern by adjusting the weights values, the new value of the hidden-output layer weight is updated according to the following equation.

$$Nweight_{HO} = Oweight_{HO} + \eta * (error_O * Output_O * (1 - Output_O)) * Ouput_H$$

Where $Nwieght_{HO}$ denotes for the new hidden-output layer weights, $Owwieght_{HO}$ denotes for the old hidden-output layer weight, η the learning rate. The new weight of the input-hidden layer weight also updated according to the following equation.

$$Nweight_{IH} = Oweight_{IH} + \eta * (error_H * Output_H * (1 - Output_H)) * Ouput_I$$

Where $Nwieght_{IH}$ denotes for the new input-hidden layer weights, $Owwieght_{IH}$ denotes for the old input-hidden layer weight. The following algorithm summarizes the training back-propagation.

1. Perform the forward-propagation phase for an input pattern and calculate the output error.
2. Change all weigh values of each weight matrix using the formula.

Weight (old) + learning rate * output error * output (neuron i) * (1-output (neuron i)) * output (neuron i-1).

3. Go to step 1.
4. The algorithm ends, if all output patterns match their target pattern.

Back-propagation is a supervised learning algorithm and is mainly used by multilayer perceptrons, because it didn't require large space in memory, easy to implement the algorithm, the error level usually accepted and calculated quickly, and finally the results compared with others is much better.

2.7 Neural Networks for Arabic Character Recognition

Many researches have been done on the Arabic characters, but a few of them have used neural networks. Here is an example of one of the research on arabic character recognition [6], This work uses two neural networks to recognize the Arabic characters: Hopfield and backpropagation. Where the Hopfield network is used for eliminating noise and enhancing the body of the unknown character and the backpropagation network (BPN) is a multilayer neural network trained by backpropagation of errors algorithm. It is simply a gradient descent method to minimize the total squared error of the output computed by the network. This work proposes a recognition technique that is divided into three parts: body classifier, complementary classifier, and aggregate classifier. Such organization reduces the large number of patterns into to 33 patterns as well as minimizing the similarity among the patterns of different characters see Figure 2.12.

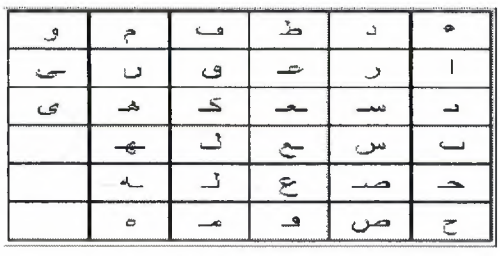


Figure 2.12 Main Body of Arabic Characters

- Body Classifier: This part is designed to classify the main body of the unknown character regardless of the dots or zigzag. In addition, it is responsible of removing the noise and enhancing the main body of the character See Figure 2.13.

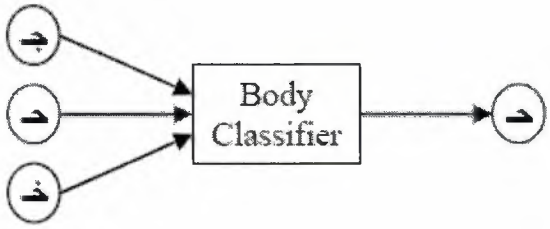


Figure 2.13 Body Classifier

- Complementary Classifier: This part is responsible of recognizing the complementary of the unknown character. It has to determine the number of dots, position, and whether they are dots or zigzag.

- Complementary Classifier: This part combines the results of the two previous parts and produces the class that the unknown character belongs to. Figure 2.14 shows the function of this part [6].

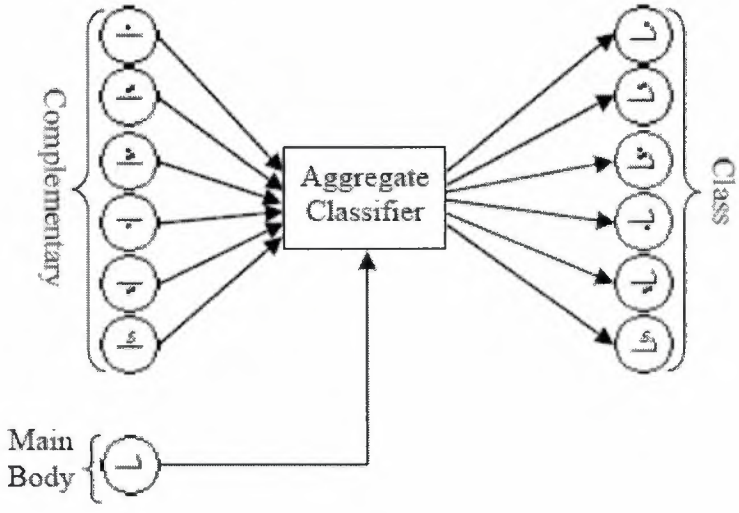


Figure 2.13 Aggregate classifier

2.8 Summary

The background and necessary information about artificial neural networks, The neurons model, the neural architecture, the learning methods of neural networks (Supervised and Unsupervised Learning methods), and the advantage and learning of Back-propagation were explained.

CHAPTER THREE
DEVELOPMENT OF NEURAL SYSTEM FOR CHARACTER RECOGNITION

3.1 Overview

In this chapter an introduction about the written language, the nature of Arabic characters, character recognition, MATLAB, and Program Stages (Database of the characters used, gray scale of the character image, scale down of the gray image, and the used neural networks architecture) will be explained.

3.2 Written Language

The written form of language is contained in printed documents, such as newspapers, magazines and books, and in handwritten matter, such as found in notebooks and personal letters, fundamental characteristics of writing are:

1. it consists of artificial graphical marks on a surface;
2. its purpose is to communicate something;
3. this purpose is achieved by virtue of the mark's conventional relation to language

Different writing systems, or scripts, represent linguistic units, words, syllables and phonemes, at different structural levels. In alphabetic writing systems, principal examples of which are the Latin, Greek and Russian scripts, alphabets are the primitive elements, or characters, which are used to represent words. Several languages such as English, Dutch, French, etc, share the Latin script. The Arabic script which represents a set pf alphabets with dots, zigzag, and diacritical marks, the Devanagari script, which represents syllables as well as alphabets, is used by several Indian languages including Hindi. The Chinese script, which consists of ideograms, is an alternative to alphabets. Each script has its own set of icons, which are known as characters or letters that have certain basic shapes. Each script has its rules for combining the letters to represent the shapes of higher level linguistic units [18].

3.2.1 The Nature of Arabic Characters

The work of recognizing Arabic characters is still limited. The importance of recognizing Arabic characters is also to be considered by Arabic non-speaking people such as Farsi, Curds, Persians, and Urdu-speaking who use the Arabic characters in writing although the pronunciation is different. It is necessary to make a quick revision of the nature of Arabic characters.

The main characteristics of Arabic Writing can be summarized as follows:

- Arabic text (printed or handwritten) is in cursive and in general, from right to left. Arabic letters are normally connected to each other on the baseline.
- Arabic writing uses letters (which consist of 28 basic letters), ten Hindi numerals, punctuation marks, as well as spaces and special symbols.
- Some of the Arabic characters are located under the baseline (for example ر, and ز).
- Some other Arabic characters consist of two parts (for exampleط and ظ).
- In the representation of vowels, Arabic uses diacritical markings. the presence or the absence vowel diacritics indicate different meaning in what would otherwise be the same word, if word is isolated diacritical are essential to distinguish between the two possible meaning , (i.e. عَلم and عُلِمَ) if it occurs in a sentence, contextual information inherent in the sentence can be used to infer the appropriate meaning.

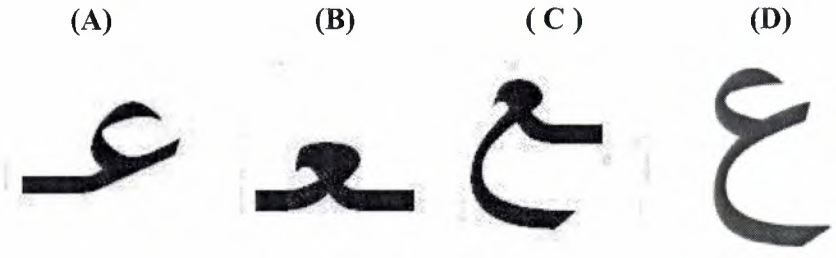


Figure 3.1 Different Shapes of the Arabic Lletter (ع A'in) in: (A) Beginning, (B) Middle, (C) End, and (D) Isolated Position

- Arabic letter might have up to four different shapes (Table 3.1), depending on its relative position in the world that increases the number of classes from 28 to 100

(Table 3.1). For example the letter (ع A'in) have four different shapes: at the beginning of the word, in the middle of the word, at the end of the word and in isolation (stand alone). Furthermore, there are two supplementary characters that operate on vowels to create a kind of stress (Hamza ء) and elongation (Madda ~); the latter operates only on the Alif (Table 3.2). The character Lam-Alif (لا) is created as combination of two characters Lam (ل) and Alif (ا), when the character Alif is written immediately after the character Lam. This new character together with the combination of Hamza (ء) and Madda (~), increases the number of classes to 120(Table 3.1,3.2).

- Different Arabic characters may have exactly the same shape, and they are to be distinguished from one another only by the addition of a complementary character (position and number of dots associated with it) [8].

Table 3.1 All Arabic Forms

Name	Isolated	Start	Middle	End
Alif	ا			ـا
Ba	ب	بـ	ـبـ	ـب
Ta	ت	تـ	ـتـ	ـت
Tha	ث	ثـ	ـثـ	ـث
Jeem	ج	جـ	ـجـ	ـج
Hha	ح	حـ	ـحـ	ـح
Kha	خ	خـ	ـخـ	ـخ
Dal	د			ـد
Thal	ذ			ـذ
Ra	ر			ـر
Zay	ز			ـز
Seen	س	سـ	ـسـ	ـس
Sheen	ش	شـ	ـشـ	ـش
Sad	ص	صـ	ـصـ	ـص
Dhad	ض	ضـ	ـضـ	ـض
Tta	ط	طـ	ـطـ	ـط
Za	ظ	ظـ	ـظـ	ـظ

Ain	ع	عـ	ـعـ	ـع
Ghain	غ	غـ	ـغـ	ـغ
Fa	ف	فـ	ـفـ	ـف
Qaf	ق	قـ	ـقـ	ـق
Kaf	ك	كـ	ـكـ	ـك
Lam	ل	لـ	ـلـ	ـل
Meem	م	مـ	ـمـ	ـم
Noon	ن	نـ	ـنـ	ـن
Ha	ه	هـ	ـهـ	ـه
Waow	و			ـو
Ya	ي	يـ	ـيـ	ـي

Table 3.2 Supplementary Characters (Hamza and Madda) and Their Position With Respect to the Alif, Waow and Ya.

Name	**Isolated**	**Start**	**Middle**	**End**
Alif	أ			ـأ
Alif	أ			ـأ
Alif	إ			ـإ
LamAlif	لا			ـلا
LamAlif	لآ			ـلآ
LamAlif	لإ			ـلإ
Waow	ؤ			ـؤ
Ya	ئ	ـئـ	ـئـ	ـئ

2.3 Character Recognition

Nowadays, there is much motivation to provide computerized document analysis systems. Character recognition contributes to this progress by providing techniques to convert large volumes of data automatically. There are so many papers and patents advertising high recognition rates; this gives the impression that automation problems seem to have been solved. However, the failure of some real applications show that

performance problems subsist on composite and degraded documents (i.e., noisy characters, tilt, mixing of fonts, etc.) and that there is still room for progress. Various methods have been proposed to increase the accuracy of character recognizers. In fact, at various research laboratories, the challenge is to develop robust methods that remove as much as possible the typographical and noise restrictions while maintaining rates similar to those provided by limited-font commercial machines.

3.2.1 General Character Recognition Steps

- Feature Extraction: This step is crucial in the context of document analysis where several variations may be caused by a number of different sources: geometric transformation because of low data quality, slant and stroke width variation because of font changing, etc. It seems reasonable to look for features which are invariant and which capture the characteristics of the character by filtering out all attributes which make the same character assume different appearances. The classifier could store a single prototype per character applies normalizing transformations to reduce certain well-defined variations as far as possible. The inevitably remaining variations are left for learning by statistical adaptation of the classifier.
- Character Learning: The keys of printed character learning are essentially training set and classification adaptation to new characters and new fonts. The training set can be given either by user or extracted directly from document samples. In the first case, the user selects the fonts and the samples to represent each character in each font and then guides the system to create models. The user must use sufficient number of samples in each font according to the difficulty of its recognition. However, it is difficult in a font context to collect a training set of characters having the expected distribution of noise and pitch size. Suggested parameterized models for imaging defects, based on a variety of theoretical arguments and empirical evidence. In the second case, the idea is to generate the training set directly from document images chosen from a wide variety of fonts and image quality and to reflect the variability expected by the system.
- Contextual Processing: Contextual processing attempts to overcome the shortcoming of decisions made on the basis of local properties and to extend the

perception on relationships between characters into word. Most of the techniques try to combine geometric information, as well as linguistic information. Characters are merged into groups which are matched against words in a dictionary using pattern matching method. In the situation where no acceptable words are found, the list of confused characters is passed through a specific net and the output is taken as the most likely word [19].

3.3 Programming Language

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice or high-productivity research, development, and analysis.

The reason that I have decided to use MATLAB for the development of this project is its toolboxes. Toolboxes allow you to *learn* and *apply* specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. It includes among others image processing and neural networks toolboxes.

3.4 Program Stages

Arabic character recognition passes through several steps as shown in Figure 3.2.

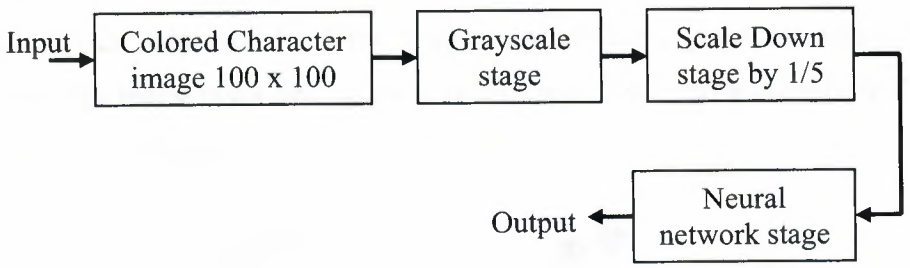


Figure 3.2 Arabic Character Recognition Stages

Character image are present as colored image (i.e. three dimentional matrix red, green, and blue), then it scale to gray color (two dimention), after that scaling down the image by factor 5 will reduce the matrix size from 100x100 to 20x20, to be input to neural network.

3.4.1 Image Database(Preprocessing Stage)

The starting point of the project was the creation of a database with all the images that would be used for training and testing (Appendix B). The image database can have different formats, but the images in this thesis are .jpg format by Adobe Photoshop program. This meant that they have same sizes and same resolutions, and then the value of the characters images pixel taken after the gray scaling and scale down stages, and combined in a .dat file acting as a database for the program.

3.4.2 Grayscale Factor

A grayscale (or gray level) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a `gray' color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image. Often, the grayscale

intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white.

Grayscale images are very common, in part because much of today's display and image capture hardware can only support 8-bit images. In addition, grayscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images.

(A) **(B)**

Figure 3.3 Grayscale of Character Qaf (ق)

Figure 3.3 (A) shows a character Qaf (ق) colored image, where after applying the grayscale Stage the result is shown in Figure 3.3(B).

3.4.3 ScaleDown Factor

Scaling is defined as the increase or reduction of image size by a fixed ratio. Scaling down is like a reversal of the process of enlargement. We first smooth the image by convolution with a spatially resolution. However, for a scale down by a specific factor in the respective directions, the used factor hear is 1/5 of the original image, so the image width to height ratio of the reduced result remains equal to that of the original image width to height ratio.

(A) (B)

Figure 3.4 Scale Down of Character Qaf (ق)

Figure 3.4 (A) Shows the output of the previous step (Grayscale step) and the Figure 3.4 (B) shows the result of the Scale down operation of the gray scale character Qaf.

3.4.4 Neural Network Structure

The underlying premise behind neural networks is that a large number of relatively dumb objects can exhibit intelligent behavior when they are linked together in the right way. Neural nets are composed of neurons which are linked to other neurons. Each neuron is a simple device that totals up all the inputs it receives from other neurons, and if the total is above a fixed threshold, the neuron will fire and send inputs to all the neurons that are linked to it. A neural network's flexibility comes from the configuration of the links, which can amplify, nullify or even negate the input signals.

Arabic recognition uses a large (but simple) Three-layer neural network to learn and recognize patterns (input, Hidden, and Output Layer). The Character image is digitized onto a grid of input neurons (as shown in Figure 3.5). The output of the input neurons are input of the hidden layer, each possible answer is represented by a single output neuron. As in most neural networks, the data (or programming) is encoded in the links between neurons.

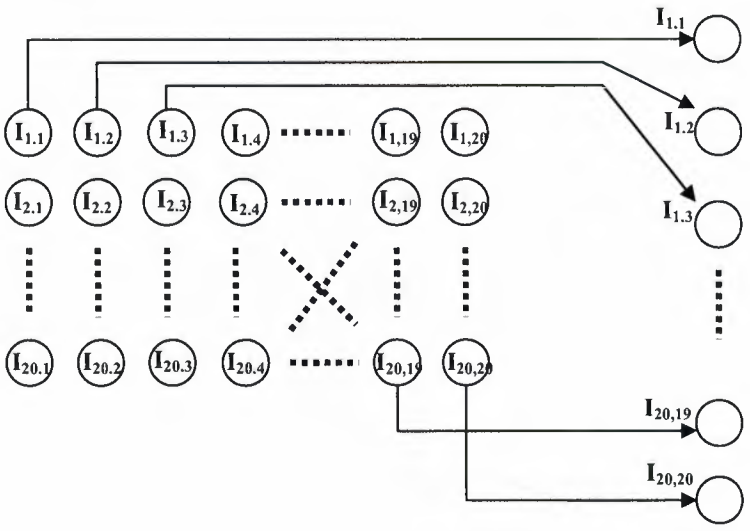


Figure 3.5 Digitize Image to a Grid of Inputs.

After the neuron in the first layer received its input, it will apply the linear combiner and the activation function to the inputs and produce the hidden input. This hidden input, as you can see from Figure 3.6, will become the input for the neurons in the next layer (hidden layer).

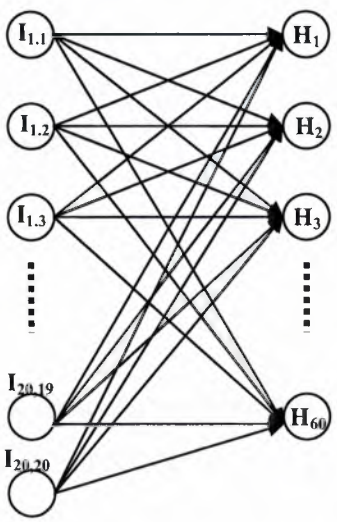


Figure 3.6 Input-Hidden Layer Feed-Forward Connection

After the neuron in the hidden layer received its input, it will apply the linear combiner and the activation function to the inputs and produce the hidden output. This hidden

output, as you can see from Figure 3.7, will become the input for the neurons in the next layer (output layer), Table 3.3 will summarize the networks specifications.

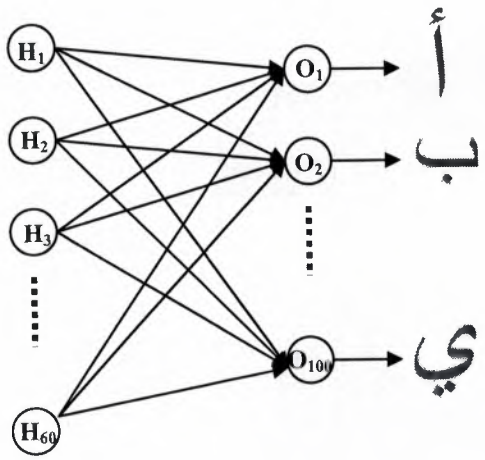


Figure 3.7 Hidden-Output Layer Feed-Forward Connection

Table 3.3 The Neural Network Specification

Input layer neurons	400
Hidden layer neurons	60
Output layer neurons	100
Momentum rate	0.30
Learning rate	0.05

3.5 Modeling

Let us consider the design at a computer program that must translate 20x20 matrixes in binary file which represent all the alphabets. { أ, ب, ت , ث, ج, ح, خ, د, ذ, ر, ز, س, ش, ص, ض, ط, ظ, ع, غ, ف, ق, ك, م, ن, ه, و, ي}

The input it's the file of .DAT Alif (ا) to Ya (ي) matrix 20x20 which is an array of gray pixels (i.e. each pixel may have a value between 0 and 255) is used to represent individual characters as shown in Table 3.4, then the division operation will convert the pixels value to a value between 0-1, to be ready as input of the input layer.

Table 3.4 Character Matrix of Qaf (ق)

98	158	158	158	158	158	158	158	158	158	158	153	39	110	156	158	158	158	158	98
158	255	255	255	255	255	255	255	255	175	211	189	35	40	215	255	255	255	255	158
158	255	255	255	255	255	255	255	238	50	38	147	115	92	252	255	255	255	255	158
158	255	255	255	255	255	255	255	238	108	50	230	254	247	255	255	255	255	255	158
158	255	255	255	255	255	255	255	255	253	231	255	255	255	255	255	255	255	255	158
158	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	158
158	255	255	255	255	255	255	255	255	255	255	219	115	136	242	255	255	255	255	158
158	255	255	255	255	255	255	255	255	255	236	56	33	33	98	252	255	255	255	158
158	255	255	255	255	255	255	255	255	255	142	69	132	46	36	207	255	255	255	158
158	255	255	187	197	255	255	255	255	253	76	159	250	143	33	130	255	255	255	158
158	255	250	88	238	255	255	255	255	252	67	39	61	41	33	82	254	255	255	158
158	255	210	124	255	255	255	255	255	255	145	35	33	33	35	66	251	255	255	158
158	255	141	202	255	255	255	255	255	255	252	195	147	162	181	65	251	255	255	158
158	254	96	238	255	255	255	255	255	255	255	255	255	255	222	67	251	255	255	158
158	252	73	238	255	255	255	255	255	255	255	255	255	255	166	65	251	255	255	158
158	252	63	164	254	255	255	255	255	255	255	255	252	184	44	89	254	255	255	158
158	254	103	39	129	215	241	246	246	241	222	169	81	35	36	188	255	255	255	158
158	255	215	46	33	36	45	51	51	46	37	33	33	42	155	253	255	255	255	158
158	255	255	216	98	46	36	34	34	36	41	63	127	222	254	255	255	255	255	158
98	158	158	158	157	145	125	118	118	123	139	153	158	158	158	158	158	158	158	98

Out put of the neurons will be match with the different types of patterns of the network and it will identify whether it is Alif (ا) or Ba (ب) or Ta (ت) and so on up to Ya (ي).

The Figure 3.8 shows the training performance of the isolate characters verses the epoch's number, the training performance that met the goal after 1991 epoch in 85.40 seconds.

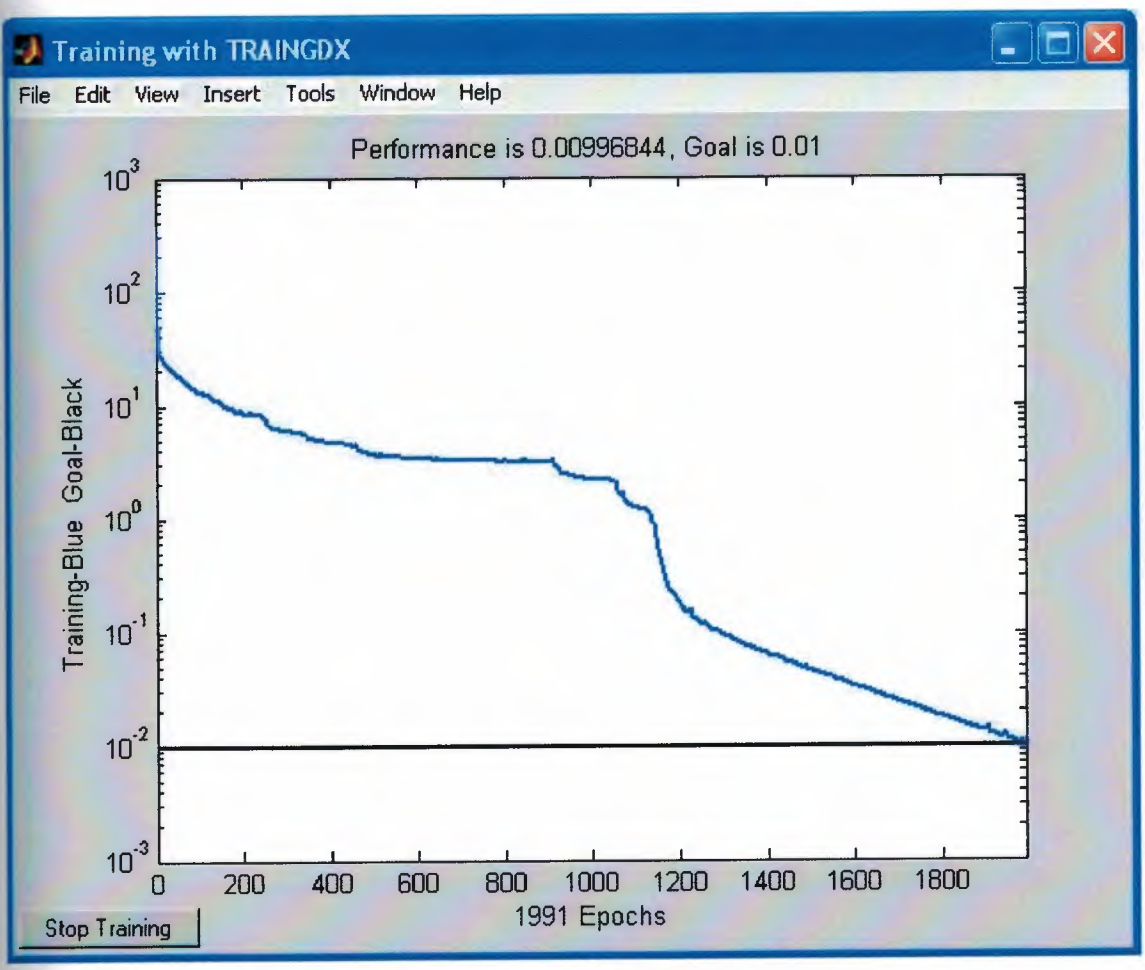


Figure 3.8 The Performance of Isolated Character After Training

Figure 3.9 shows the training performance of the beginning word characters, the training performance that met the goal after 1419 epoch in 48.39 seconds.

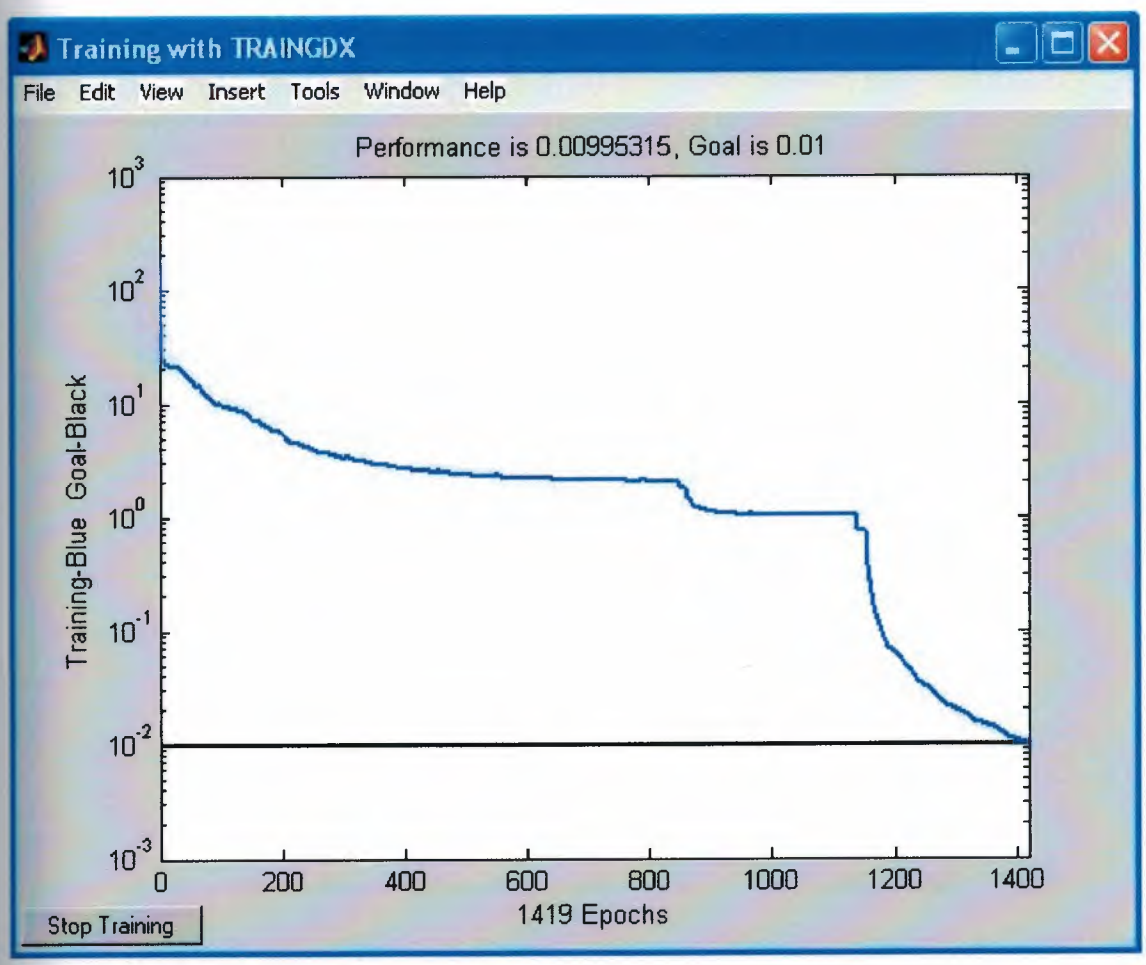


Figure 3.9 The Performance of Beginning Characters After Training

Figure 3.10 shows the training performance of the middle word characters, the training performance that met the goal after 1009 epoch in 37.07 seconds.

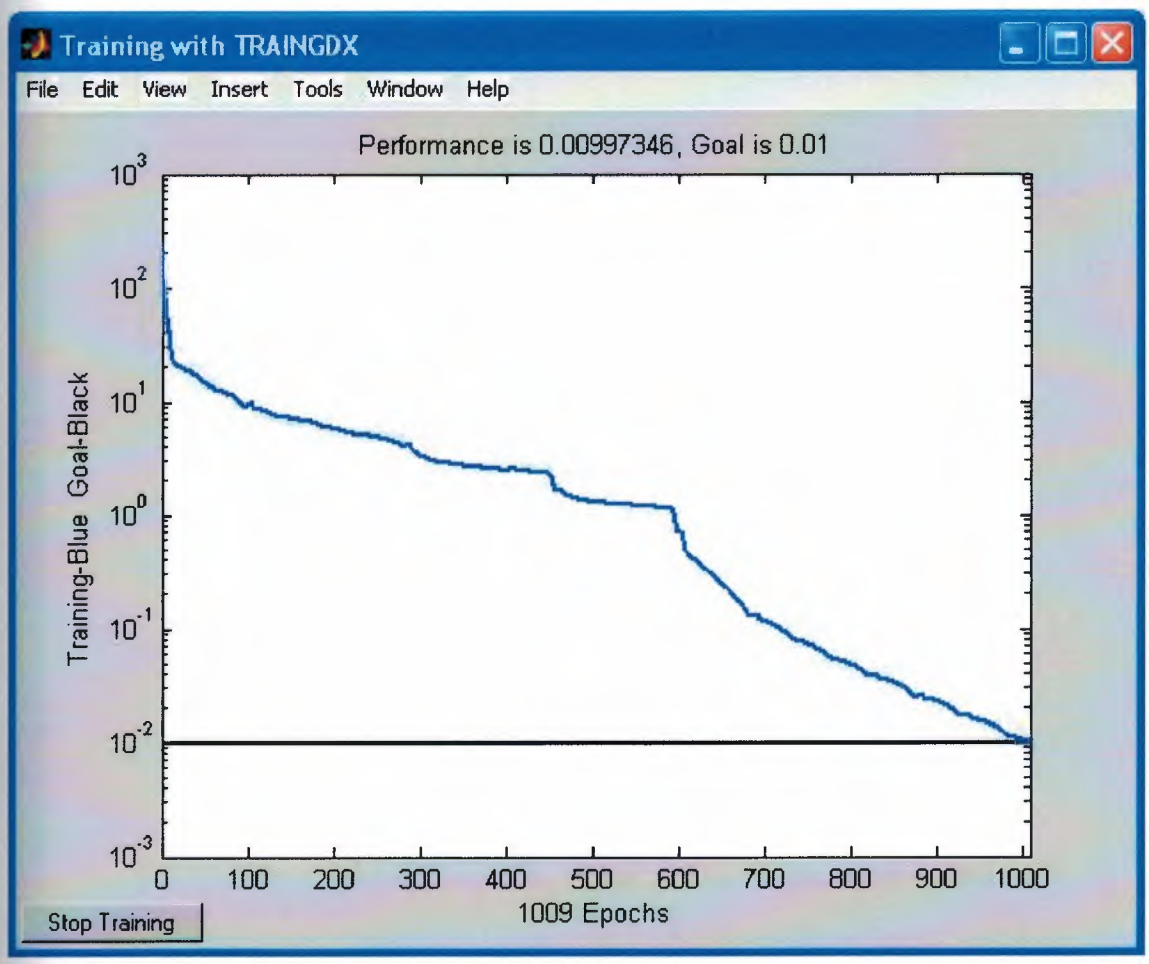


Figure 3.10 The Performance of Middle Characters After Ttraining

Figure 3.11 shows the training performance of the end word characters, the training performance that met the goal after 1961 epoch in 78.03 seconds with a percentages reaches to 98.76%.

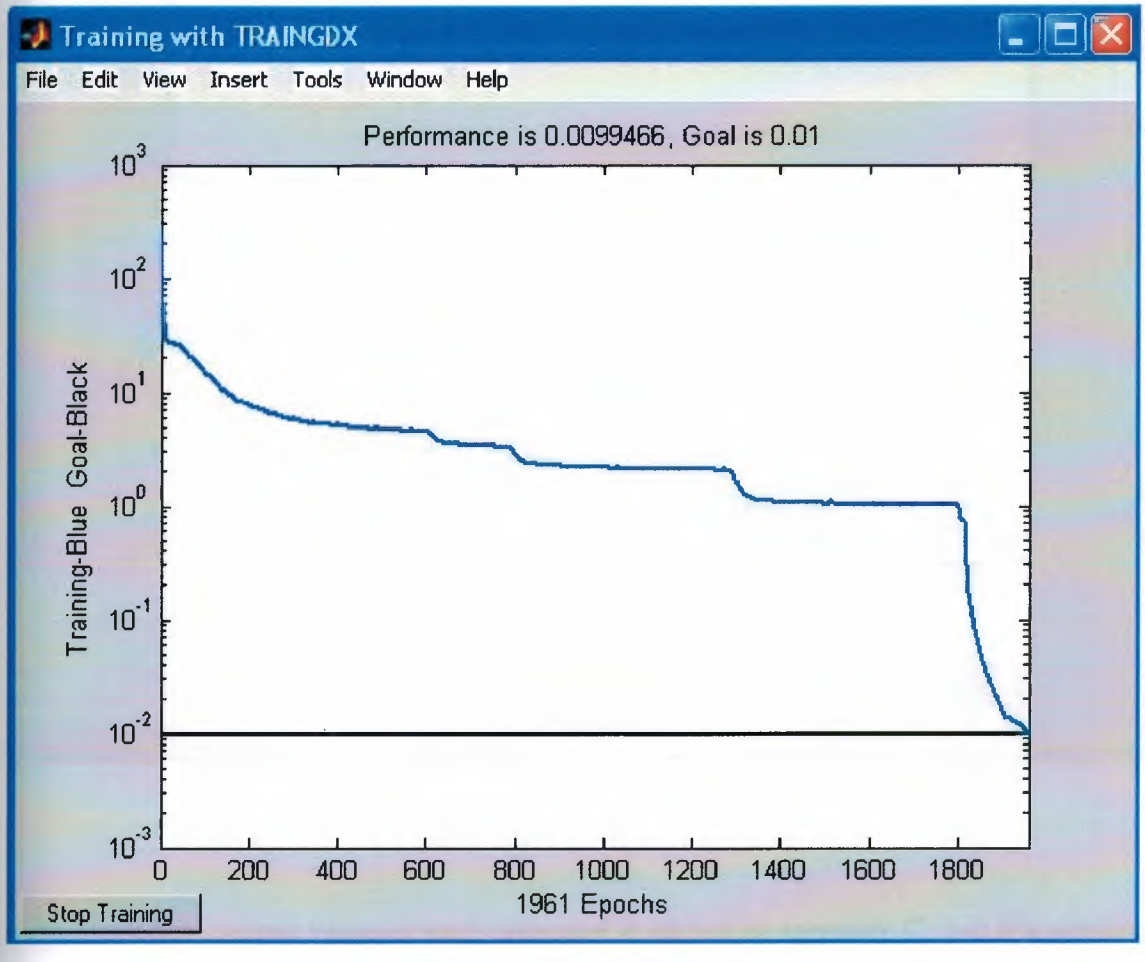


Figure 3.11 The Performance of End Characters After Training

Figure 3.12 shows the training performance of all Arabic characters, the training performance that met the goal after 2702 epoch in 183.35 seconds with a percentages reaches to 98%.

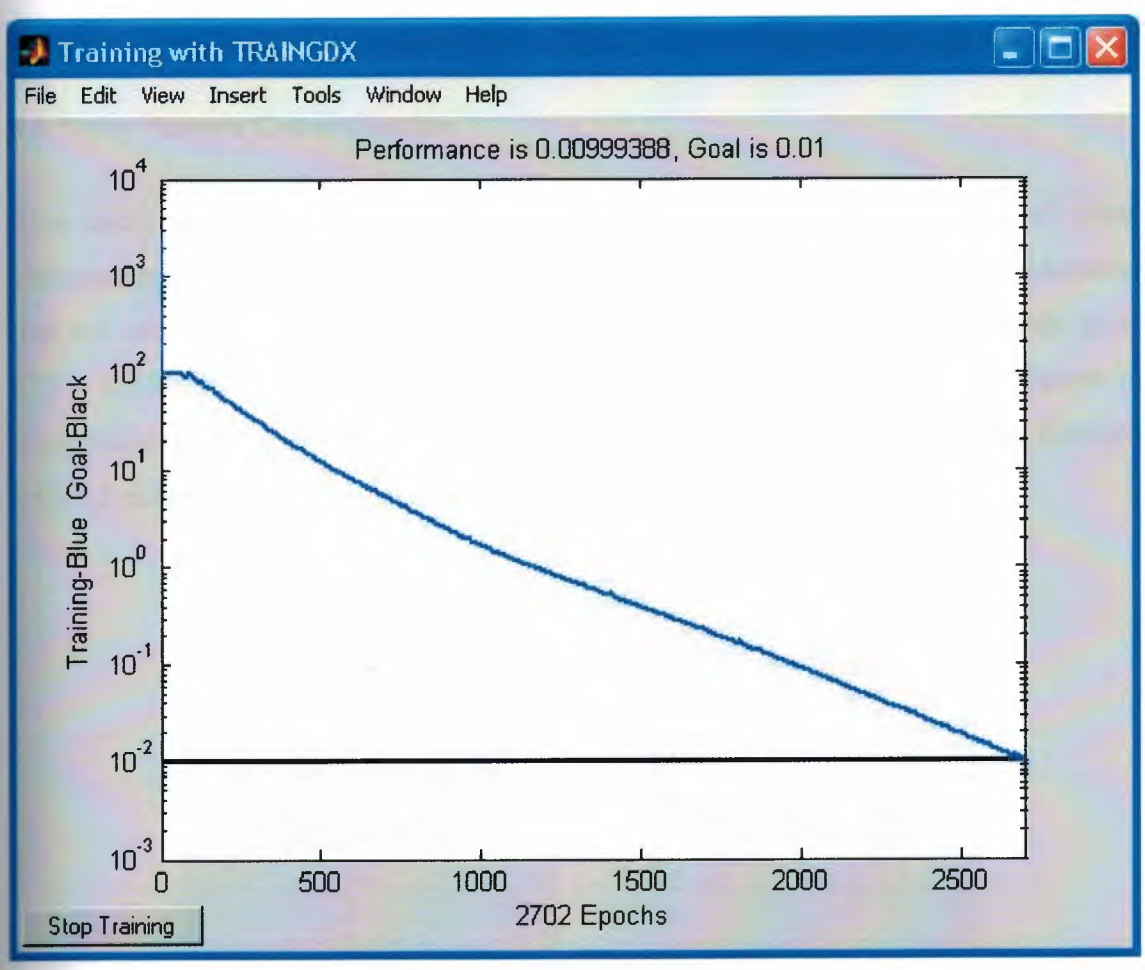


Figure 3.12 The Performance of End Characters After Training

The characters output value of each character is shown in appendix C, and it's obvious that the result is identical to the target state, one active state for each output. For example, the target for Alif (ا) and the corresponding output are given in Figure 3.13

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 0.98635 0.0055829 0.001472 0.0021997 0.0029103 0.0040847 0.00077465 0.00015901
 0.0056891  0.0050062  0.002928  8.4109e-005  0.0056099  0.00040074  0.0013516
 0.0035199  0.00014667  0.00010608  0.0055687  0.005187  0.00056629  0.0027053
 0.00026116 0.0010996 1.2206e-005 0.00012556 2.1421e-005 1.694e-005
```

Figure 3.13 The Output and its Corresponding Target

3.6 Noisy Pattern Generalization

Also using developed neural character recognition system, the recognition of noisy character has been applied. Two types of noise were added separately to the character that are used for testing, salt & pepper noise (0.2 – 0.7 noise range) and spackle noise (0.2 – 0.5 noise range) were both are added in order to test the systems efficient in recognition. Figure 3.14 shows one of the characters using salt & pepper noise function with 0.5 noise level and its grayscale transformation.

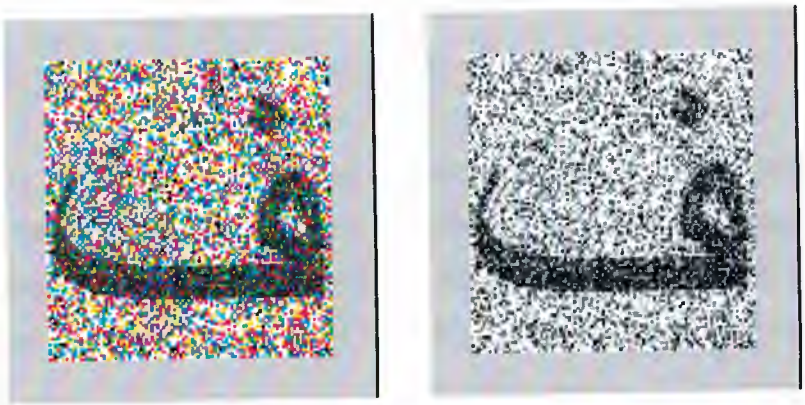


Figure 3.14 The Noisy Character and its Grayscale Transformation Using Salt & Pepper Noise

Figure 3.15 shows one of the characters using speckle noise function with 0.5 noise level and its grayscale transformation.

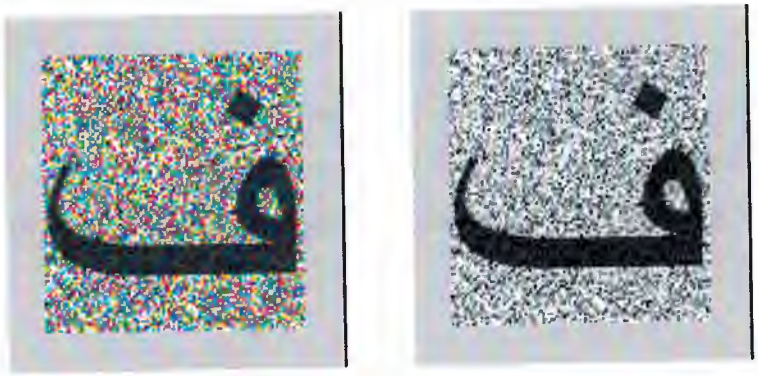


Figure 3.15 The Noisy Character and its Grayscale Transformation Using Speckle Noise

In result of recognition of noisy character the accuracy was 89.75 %, and this rate are good compared with previous work.

The Arabic character recognition for all Arabic characteristics (isolated, beginning, middle, and end) reaches high accuracy for noisy and non-noisy characters as shown in Table 3.5.

Table 3.5 Arabic Character Recognition Accuracy

	Isolated characters	Beginning characters	Middle Characters	End characters	All Characters
Noisy	89.75	.....	....	....	.....
Non-noisy	98.74	98.78	98.55	98.92	98.76

Result of simulated system is compared with the result of Arabic character recognition system that use K-mean algorithm described in [14]. In table comparative results of these research works is given. As shown from table the developed system in this thesis has more accuracy. The results of comparison demonstrate the efficiency of proposed work.

Table 3.6 Arabic Character Recognition Accuracy

	K-mean algorithm	Neural Network
Recognition rate	95 %	98.5%

3.7 Summary

Arabic language is one of the written languages that have special characteristics as described, the character recognition stages, MATLAB program used to solve the problem of Arabic character recognition, and the stages of the program shown in this chapter.

CONCLUSION

Using neural network system, back-propagation learning, to recognize Arabic character recognition system was very successful, there are many advantages of this technique as mentioned. Preprocessing techniques have been described including the way of preparing the database of the characters image. Gray scaling of the image is discussed and the scale down is described.

A back-propagation neural network is used to adjust and learn the characters represented. This analysis exploits the position of a character in the word, and differs from previous ones in the field of Arabic character recognition by dealing with recognition as character position.

The process contains several steps that cannot be separated from each other (i.e. preprocessing, gray scale, scale down and learning). After train a combination of 100 sample represent the 28 Arabic character in all positions, a 98% recognition rate has been achieved for Arabic characters in non-noisy space, and 89% recognition rate has been achieved for noisy Arabic characters using two types of noise that added separately to the character that are used for testing, salt & pepper and spackle noise.

Comparison of results with other researches' is difficult to make, but some researchers achieved a recognition rate as follows 56% in [5], 85% in [6], 89% in [7], 69.7% in [8] and 95% in [14].

Finally, since character recognition is such a large subject, there is plenty of scope for future directions. If more time had been available it would have been interesting to look at some other problems, such as handwritten, and word recognition.

The research for Arabic characters is implemented using MATLAB in Pentium IV.

REFERENCES

[1] Anil K. Jain and Jianchang Mao, "Statistical Pattern Recognition: A Review", IEEE VOL. 22, NO. 1, JANUARY 2000

[2] Tralvex Yeap and Yang Kok Wah, "principles of handwritten character recognition", Singapore.

[3] Andrew R. Webb, "Statistical Pattern Recognition", second edition, Southern Gate, Chichester, West Sussex PO19 8SQ, England,2002.

[4] Hinduja Technologies Group USA, Inc., DBA HTG Advance Systems, http://www.htgadvancesystems.com/advance/index.html

[5] Berrin Yanikoglu and Alisher Kholmatov, "Turkish handwritten text recognition", Sabanci University, Orhanli, Istanbul, Turkey 34956

[6] Saleh F. SALEH, "Arabic Character Recognition Using Neural Networks", Computer Science Department, Zarka Private University, Jordan, salah@zpu.edu.jo.

[7] S. KNERR, L. PERSONNAZ, and G. DREYFUS, "Handwritten Digit Recognition by Neural Networks with Single-Layer Training", 75005 PARIS, FRANCE.

[8] M. Fahmy and S. AlAli, "Automatic Recognition of Handwritten Arabic Characters Using Their Geometrical Features", 2001, http://www.ici.ro/ici/revista/sic2001_2/art1.htm

[9] H. Bunke, "Recognition of Cursive Roman Handwriting Past, Present and Future", the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), University of Bern, Switzerland.

[10] M.S. Korsheed, "Recognising handwritten Arabic manuscripts using a single hidden Markov model", Cambridge, UK, Pattern recognition letters, Letters 24 (2003) 2235–2242.

[11] I. Saxena P. Moerland E. Fiesler A. Pourzand, "Handwritten Digit Recognition with Binary Optical Perceptron" , International Conference on Arti_cial Neural Networks (ICANN'97), Lausanne, Switzerland, October 1997, 1253-1258.

[12] Y. Al-Ohalia, M. Cheriet, C. Suen, "Databases for recognition of handwritten Arabic cheques", Pattern Recognition Society, 8 march 2002.

[13] Martin T. Hagan, Howard B. Demuth, Mark H. Beale, "Neural Network Design", ISBN: 0-9717321-0-8.

[14] M. GHAZAL, K. ASSALEH, and M. ALROUSAN, "Online recognition Of Handwritten Arabic Character", American University of Sharjah, Sharjah, U.A.E.

[15] D. Anderson and G. McNeill, "Artificial Neural Networks Technology", Kaman Science Corporation, New York, 1992.

[16] Jochen Frohlich, "Neural nets", Computer Science Klenzestr Regensburg, 1999.

[17] Samuel Hsiung, "Multilayer Feedforward Network and the Backpropagation Algorithm", 1999, http://www.generation5.org/content/1999/nn_bp.asp

[18] R. Cole, J. Mariani, H. Uszkoretit, and A. Zaenen, "servey of the state of the art in humman language technology", National Science foundation, 1996

[19] R. Cole, J. Mariani, H. Uszkoretit, and A. Zaenen, " survey of the state of the art in human language technology", National Science foundation, 1996.

[20] Rahib Abiyev. Controllers based on Soft computing elements// Electrical, Electronics and Computer Engineering Symposium NEU-CEE2001 & Exhibition. Nicosia, TRNC, Turkey, May 23-25, 2001, pp.182-188.

APPENDIX A

The following Arabic characters are the isolated characters, those images used as input of the program.

Alif Ba Ta Tha

Jeem Hha Kha Dal

Thal Ra Zay Seen

Sheen Sad Dad Tta

Za Ain Ghain Fa

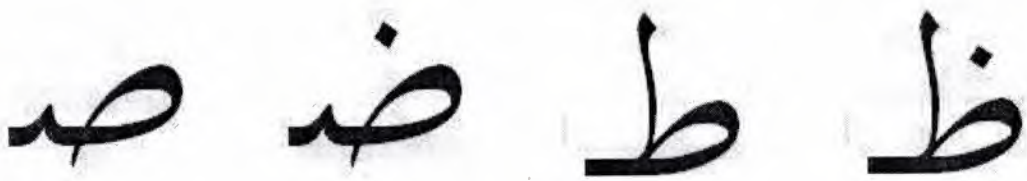
Qaf Kaf Lam Meem

Noon Ha Waow Ya

The Beginning character of the Arabic words

Ba_b Ta_b Tha_b Jeem_b

Hha_b Kha_b Seen_b Sheen_b



Sad_b Dad_b Tta_b Za_b

Ain_b Ghain_b Fa_b Qaf_b

Sad_m

Dad_m

Tta_m

Za_m

Ain_m

Ghain_m

Fa_m

Qaf_m

Kaf_m

Lam_m

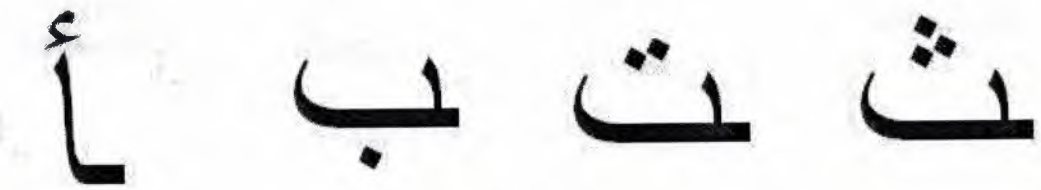
Meem_m

Noon_n

Ha_m

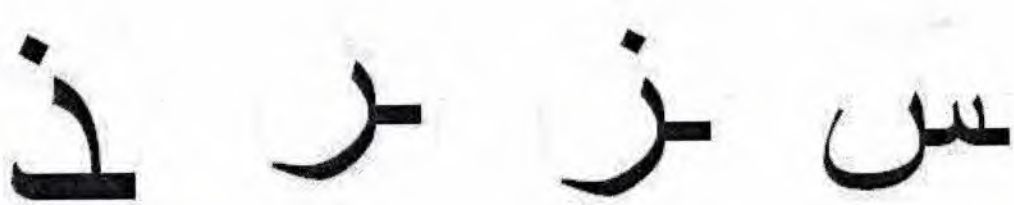
Ya_m

The Middle character of the Arabic words

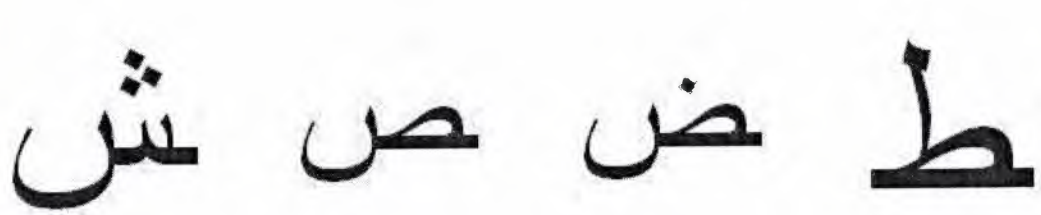


Alif_e Ba_e Ta_e Tha_e

Jeem_e Hha_e Kha_e Dal_e

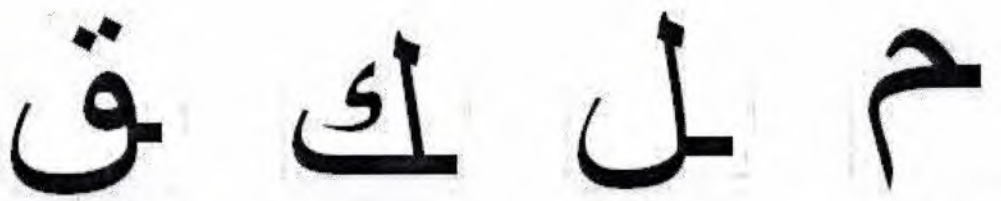


Thal_e Ra_e Zay_e Seen_e



Sheen_e Sad_e Dad_e Tta_e

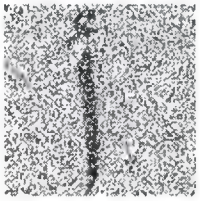
Za_e Ain_e Ghain_e Fa_e



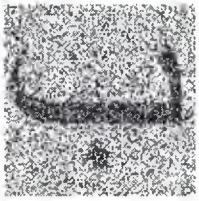
Qaf_e Kaf_e Lam_e Meem_e

Noon_e Ha_e Waow_e Ya_e

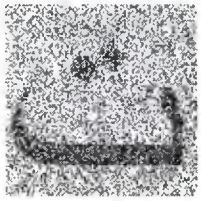
The noisy isolated character of the Arabic words using salt & pepper with level noise of 0.5.



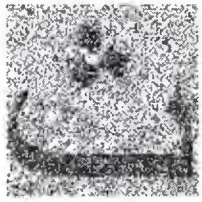
Alif



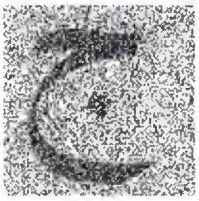
Ba



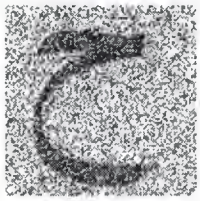
Ta



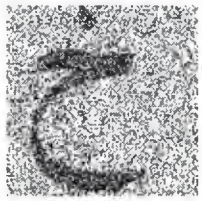
Tha



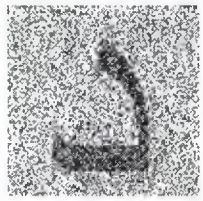
Jeem



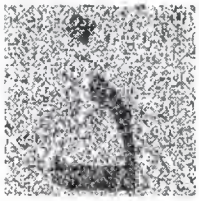
Hha



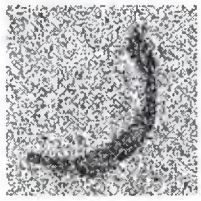
Kha



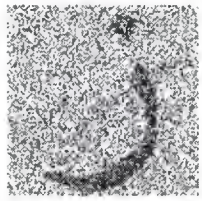
Dal



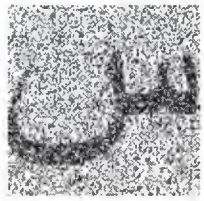
Thal



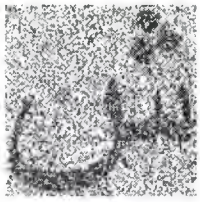
Ra



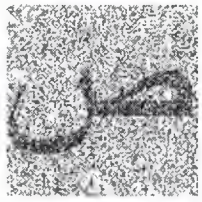
Zay



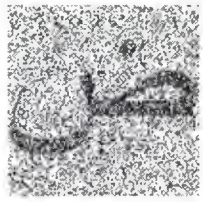
Seen



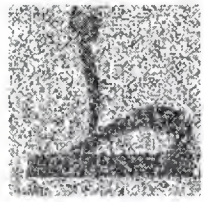
Sheen



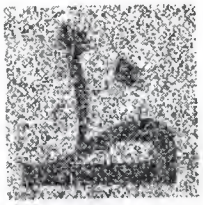
Sad



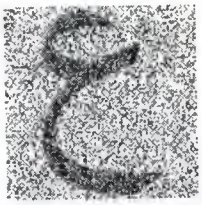
Dad



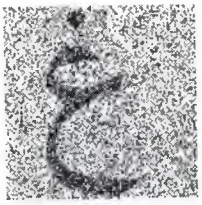
Tta



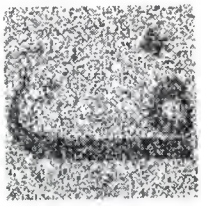
Za



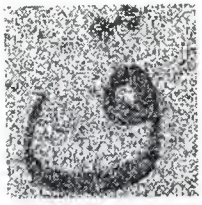
Ain



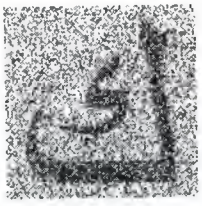
Ghain



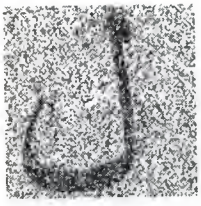
Fa



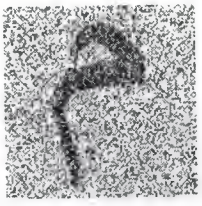
Qaf



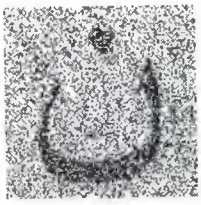
Kaf



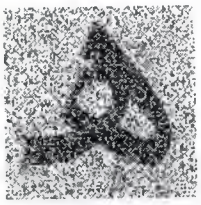
Lam



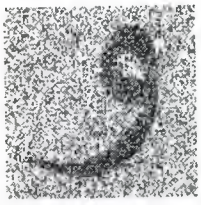
Meem



Noon



Ha

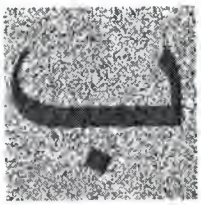


Waow

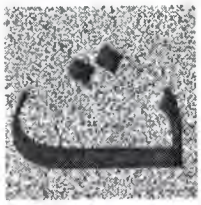
Ya

The noisy isolated character of the Arabic words using speckle with level noise of 0.5.

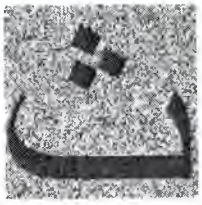
Alif



Ba



Ta

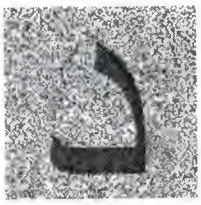


Tha

Jeem

Hha

Kha



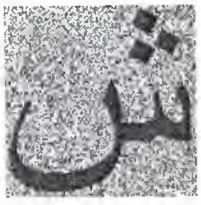
Dal

Thal

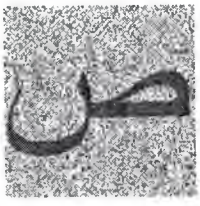
Ra

Zay

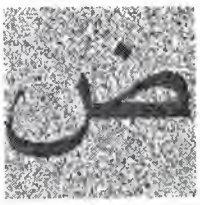
Seen



Sheen



Sad



Dad

Tta

APPENDIX B

Source codes of the program, Gray scale stage, scale down stage, train characters stage, and testing stage.

- Gray Scaling:

```
buffer=pwd;
file ='Alif.jpg'
a=imread(file) %color image
cd(buffer);
i=rgb2gray(a);
imshow(i)
size(i)
i
```

- Scaling down

```
% Changing the spatial resolution to 20 x 20:
% ----------------------------------------
k = imresize(i, (1/5), 'bicubic');
imwrite(k, 'downby2.jpg');
figure;
imagesc(k);
colormap(gray);
title('Downsampled by Factor of 5');
axis image;
k
```

- Train Isolated characters

```
load all.dat
km1=all(:,:)
for i=1:400
   for j=1:28
```

```
    km1(i,j)=km1(i,j)/256;
end
end
start_time=0;
load target.dat;
I1 = 20;   % matrix size
T=target(:,:);
pause
km1=double(km1);
[R,Q] = size(km1)
[S2,Q] = size(T)
S1=60;
P=km1;
P
T
net = newff(minmax(P),[S1 S2],{'logsig' 'logsig'},'traingdx');
Y=sim(net,P)
net.performFcn = 'sse';
net.trainParam.goal = 0.01;
net.trainParam.lr = 0.005;
net.trainParam.show = 20;
net.trainParam.epochs = 100000;
net.trainParam.mc = 0.30;
 [net,tr,Y,E] = train(net,P,T);
Y=sim(net,P)
% get stop time
stop_time = cputime;
% obtain the execution time of the program
execution_time = stop_time - start_time;
% display the execution time of the program
disp('Execution Time: ');
disp('--------------');
execution_time
```

- Train Beginning characters

```
load all.dat
km1=all(:,:)
for i=1:400
   for j=1:28
     km1(i,j)=km1(i,j)/256;
end
end
start_time=0;
load target.dat;
I1 = 20;   % matrix size
T=target(:,:);
pause
km1=double(km1);
[R,Q] = size(km1)
[S2,Q] = size(T)
S1=60;
P=km1;
P
T
net = newff(minmax(P),[S1 S2],{'logsig' 'logsig'},'traingdx');
Y=sim(net,P)
net.performFcn = 'sse';
net.trainParam.goal = 0.01;
net.trainParam.lr = 0.005;
net.trainParam.show = 20;
net.trainParam.epochs = 100000;
net.trainParam.mc = 0.30;
[net,tr,Y,E] = train(net,P,T);
Y=sim(net,P)
% get stop time
stop_time = cputime;
```

```
% obtain the execution time of the program
execution_time = stop_time - start_time;
% display the execution time of the program
disp('Execution Time: ');
disp('--------------');
execution_time
```

- Train Middle character

```
load allmiddle.dat
km1=allmiddle(:,:)
for i=1:400
   for j=1:22
     km1(i,j)=km1(i,j)/256;
end
end
start_time=0;
load target22.dat;
I1 = 20;   % matrix size
T=target22(:,:);
pause
km1=double(km1);
[R,Q] = size(km1)
[S2,Q] = size(T)
S1=60;
P=km1;
P
T
net = newff(minmax(P),[S1 S2],{'logsig' 'logsig'},'traingdx');
Y=sim(net,P)
net.performFcn = 'sse';
net.trainParam.goal = 0.01;
net.trainParam.lr = 0.005;
net.trainParam.show = 20;
```

```
net.trainParam.epochs = 100000;
net.trainParam.mc = 0.30;
[net,tr,Y,E] = train(net,P,T);
Y=sim(net,P)
% get stop time
stop_time = cputime;
% obtain the execution time of the program
execution_time = stop_time - start_time;
% display the execution time of the program
disp('Execution Time: ');
disp('--------------');
execution_time
```

- Train end characters

```
load allend.dat
km1=allend(:,:)
for i=1:400
   for j=1:28
     km1(i,j)=km1(i,j)/256;
end
end
start_time=0;
load target.dat;
I1 = 20;   % matrix size
T=target(:,:);
pause
km1=double(km1);
[R,Q] = size(km1)
[S2,Q] = size(T)
S1=60;
P=km1;
P
T
```

```
net = newff(minmax(P),[S1 S2],{'logsig' 'logsig'},'traingdx');
Y=sim(net,P)
net.performFcn = 'sse';
net.trainParam.goal = 0.01;
net.trainParam.lr = 0.005;
net.trainParam.show = 20;
net.trainParam.epochs = 100000;
net.trainParam.mc = 0.30;
 [net,tr,Y,E] = train(net,P,T);
Y=sim(net,P)
% get stop time
stop_time = cputime;
% obtain the execution time of the program
execution_time = stop_time - start_time;
% display the execution time of the program
disp('Execution Time: ');
disp('--------------');
execution_time
```

- Train all characters

```
load allletters.dat
km1=allletters(:,:)
start_time=0;
%load target.dat;
I1 = 20;   % matrix size
T=eye(100);%target(:,:);
 km1=double(km1);
[R,Q] = size(km1)
[S2,Q] = size(T)
S1=70;
P=km1;
P
T
```

```
net = newff(minmax(P),[S1 S2],{'logsig' 'logsig'},'traingdx');
%Y=sim(net,P)
net.LW{2,1} = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;
net.performFcn = 'sse';
net.trainParam.goal = 0.01;
net.trainParam.lr = 0.005;
net.trainParam.show = 20;
net.trainParam.epochs = 100000;
net.trainParam.mc = 0.30;

[net,tr,Y,E] = train(net,P,T);

Y=sim(net,P)

% get stop time
stop_time = cputime;
% obtain the execution time of the program
execution_time = stop_time - start_time;

% display the execution time of the program
disp('Execution Time: ');
disp('--------------');
execution_time
```

- Test non-noisy characters

```
buffer=pwd;
      I = imread('alif.jpg');
cd(buffer);
i=rgb2gray(I);
imshow(i)
size(i)
i
```

```
% Changing the spatial resolution to 20 x 20:
% ----------------------------------------
k = imresize(i, (1/5), 'bicubic');
imwrite(k, 'downby2.jpg');
figure;
imagesc(k);
colormap(gray);
title('Downsampled by Factor of 5');
axis image;
k
I1 = 20;
for i=1:I1
   for j=1:I1
      km2((i-1)*I1+j)=k(i,j);
   end
end
km=double(km2');
km
pause
for i=1:400
   km(i,1)=km(i,1)/256;
end
Y=sim(net,km)
max=Y(1);
for i=1:28
   if (Y(i)>=max)
      max=Y(i);
      j=i;
   end;
end
max
if (j==1)
   imshow('Alif.jpg');
```

```
    title('The character recognized');
end
if (j==2)
    imshow('Ba.jpg');
    title('The character recognized');
end
if (j==3)
    imshow('Ta.jpg');
    title('The character recognized');
end
if (j==4)
    imshow('Tha.jpg');
    title('The character recognized');
end
if (j==5)
    imshow('Jeem.jpg');
    title('The character recognized');
end
if (j==6)
    imshow('Hha.jpg');
    title('The character recognized');
end
if (j==7)
    imshow('Kha.jpg');
    title('The character recognized');
end
if (j==8)
    imshow('Dal.jpg');
    title('The character recognized');
end
if (j==9)
    imshow('Thal.jpg');
    title('The character recognized');
```

```
end
if (j==10)
   imshow('Ra.jpg');
   title('The character recognized');
end
if (j==11)
   imshow('Zay.jpg');
   title('The character recognized');
end
if (j==12)
   imshow('Seen.jpg');
   title('The character recognized');
end
if (j==13)
   imshow('Sheen.jpg');
   title('The character recognized');
end
if (j==14)
   imshow('Sad.jpg');
   title('The character recognized');
end
if (j==15)
   imshow('Dad.jpg');
   title('The character recognized');
end
if (j==16)
   imshow('Tta.jpg');
   title('The character recognized');
end
if (j==17)
   imshow('Za.jpg');
   title('The character recognized');
end
```

```
if (j==18)
   imshow('Ain.jpg');
   title('The character recognized');
end
if (j==19)
   imshow('Ghain.jpg');
   title('The character recognized');
end
if (j==20)
   imshow('Fa.jpg');
   title('The character recognized');
end
if (j==21)
   imshow('Qaf.jpg');
   title('The character recognized');
end
if (j==22)
   imshow('Kaf.jpg');
   title('The character recognized');
end
if (j==23)
   imshow('Lam.jpg');
   title('The character recognized');
end
if (j==24)
   imshow('Meem.jpg');
   title('The character recognized');
end
if (j==25)
   imshow('Noon.jpg');
   title('The character recognized');
end
if (j==26)
```

```
   imshow('Ha.jpg');
   title('The character recognized');
end
if (j==27)
   imshow('Waow.jpg');
   title('The character recognized');
end
if (j==28)
   imshow('Ya.jpg');
   title('The character recognized');
end
if max<0.70 & max>=0.50
   h=msgbox('most propably recognized                    !!','high level of noise');
elseif max<0.50
   h=msgbox('may not recognized                    !!','low recognition rate');
end
% get stop time
stop_time = cputime;
% obtain the execution time of the program
execution_time = stop_time - start_time;
% display the execution time of the program
disp('Execution Time: ');
disp('--------------');
execution_time
```

- Test Noisy characters

```
buffer=pwd;
      I = imread('alif.jpg');
      J = imnoise(I,'salt & pepper', 0.02);
      imshow(I), figure, imshow(J)
 cd(buffer);
i=rgb2gray(J);
imshow(i)
```

```
size(i)
i
% Changing the spatial resolution to 20 x 20:
% ----------------------------------------
k = imresize(i, (1/5), 'bicubic');
imwrite(k, 'downby2.jpg');
figure;
imagesc(k);
colormap(gray);
title('Downsampled by Factor of 5');
axis image;
k
I1 = 20;
for i=1:I1
   for j=1:I1
      km2((i-1)*I1+j)=k(i,j);
   end
end
km=double(km2');
km
pause
for i=1:400
   km(i,1)=km(i,1)/256;
end
Y=sim(net,km)
max=Y(1);
for i=1:28
   if (Y(i)>=max)
      max=Y(i);
      j=i;
   end;
end
max
```

```
if (j==1)
   imshow('Alif.jpg');
   title('The character recognized');
end
if (j==2)
   imshow('Ba.jpg');
   title('The character recognized');
end
if (j==3)
   imshow('Ta.jpg');
   title('The character recognized');
end
if (j==4)
   imshow('Tha.jpg');
   title('The character recognized');
end
if (j==5)
   imshow('Jeem.jpg');
   title('The character recognized');
end
if (j==6)
   imshow('Hha.jpg');
   title('The character recognized');
end
if (j==7)
   imshow('Kha.jpg');
   title('The character recognized');
end
if (j==8)
   imshow('Dal.jpg');
   title('The character recognized');
end
if (j==9)
```

```
   imshow('Thal.jpg');
   title('The character recognized');
end
if (j==10)
   imshow('Ra.jpg');
   title('The character recognized');
end
if (j==11)
   imshow('Zay.jpg');
   title('The character recognized');
end
if (j==12)
   imshow('Seen.jpg');
   title('The character recognized');
end
if (j==13)
   imshow('Sheen.jpg');
   title('The character recognized');
end
if (j==14)
   imshow('Sad.jpg');
   title('The character recognized');
end
if (j==15)
   imshow('Dad.jpg');
   title('The character recognized');
end
if (j==16)
   imshow('Tta.jpg');
   title('The character recognized');
end
if (j==17)
   imshow('Za.jpg');
```

```
   title('The character recognized');
end
if (j==18)
   imshow('Ain.jpg');
   title('The character recognized');
end
if (j==19)
   imshow('Ghain.jpg');
   title('The character recognized');
end
if (j==20)
   imshow('Fa.jpg');
   title('The character recognized');
end
if (j==21)
   imshow('Qaf.jpg');
   title('The character recognized');
end
if (j==22)
   imshow('Kaf.jpg');
   title('The character recognized');
end
if (j==23)
   imshow('Lam.jpg');
   title('The character recognized');
end
if (j==24)
   imshow('Meem.jpg');
   title('The character recognized');
end
if (j==25)
   imshow('Noon.jpg');
   title('The character recognized');
```

```
end
if (j==26)
   imshow('Ha.jpg');
   title('The character recognized');
end
if (j==27)
   imshow('Waow.jpg');
   title('The character recognized');
end
if (j==28)
   imshow('Ya.jpg');
   title('The character recognized');
end
if max<0.70 & max>=0.50
   h=msgbox('most propably recognized                    !!','high level of noise');
elseif max<0.50
   h=msgbox('may not recognized                          !!','low recognition rate');
end
% get stop time
stop_time = cputime;
% obtain the execution time of the program
execution_time = stop_time - start_time;
% display the execution time of the program
disp('Execution Time: ');
disp('--------------');
execution_time
```

- Test all characters

```
buffer=pwd;
      I = imread('fa.jpg');
      J = imnoise(I,'salt & pepper', 0.2);
      imshow(I), figure, imshow(J)
```

```
cd(buffer);
i=rgb2gray(J);
s=size(i)
for I=1:s(1)
   for j=1:s(2)
      if i(I,j)<150
         i(I,j)=0;
      else
         i(I,j)=1;
      end
   end
end
colormap(gray);
imagesc(i)

% Changing the spatial resolution to 20 x 20:
% -------------------------------------------
k = imresize(i, (1/5), 'bicubic');
imwrite(k, 'downby2.jpg');
figure;
imagesc(k);
colormap(gray);
title('Downsampled by Factor of 5');
axis image;
k

I1 = 20;
for i=1:I1
   for j=1:I1
      km2((i-1)*I1+j)=k(i,j);
   end
end
km=double(km2');
```

```
km

pause
Y=sim(net,km)

max=Y(1);
for i=1:100
   if (Y(i)>=max)
      max=Y(i);
      j=i;
   end;
end
max
if (j==1)|(j==2)
   imshow('Alif.jpg');
   title('The character recognized');
end
if (j==3)|(j==4)|(j==5)|(j==6)
   imshow('Ba.jpg');
   title('The character recognized');
end
if (j==7)|(j==8)|(j==9)|(j==10)
   imshow('Ta.jpg');
   title('The character recognized');
end
if (j==11)|(j==12)|(j==13)|(j==14)
   imshow('Tha.jpg');
   title('The character recognized');
end
if (j==15)|(j==16)|(j==17)|(j==18)
   imshow('Jeem.jpg');
   title('The character recognized');
end
```

```
if (j==19)|(j==20)|(j==21)|(j==22)
   imshow('Hha.jpg');
   title('The character recognized');
end
if (j==23)|(j==24)|(j==25)|(j==26)
   imshow('Kha.jpg');
   title('The character recognized');
end
if (j==27)|(j==28)
   imshow('Dal.jpg');
   title('The character recognized');
end
if (j==29)|(j==30)
   imshow('Thal.jpg');
   title('The character recognized');
end
if (j==31)|(j==32)
   imshow('Ra.jpg');
   title('The character recognized');
end
if (j==33)|(j==34)
   imshow('Zay.jpg');
   title('The character recognized');
end
if (j==35)|(j==36)|(j==37)|(j==38)
   imshow('Seen.jpg');
   title('The character recognized');
end
if (j==39)|(j==40)|(j==41)|(j==42)
   imshow('Sheen.jpg');
   title('The character recognized');
end
if (j==43)|(j==44)|(j==45)|(j==46)
```

```
    imshow('Sad.jpg');
    title('The character recognized');
end
if (j==47)|(j==48)|(j==49)|(j==50)
    imshow('Dad.jpg');
    title('The character recognized');
end
if (j==51)|(j==52)|(j==53)|(j==54)
    imshow('Tta.jpg');
    title('The character recognized');
end
if (j==55)|(j==56)|(j==57)|(j==58)
    imshow('Za.jpg');
    title('The character recognized');
end
if (j==59)|(j==60)|(j==61)|(j==62)
    imshow('Ain.jpg');
    title('The character recognized');
end
if (j==63)|(j==64)|(j==65)|(j==66)
    imshow('Ghain.jpg');
    title('The character recognized');
end
if (j==67)|(j==68)|(j==69)|(j==70)
    imshow('Fa.jpg');
    title('The character recognized');
end
if (j==71)|(j==72)|(j==73)|(j==74)
    imshow('Qaf.jpg');
    title('The character recognized');
end
if (j==75)|(j==76)|(j==77)|(j==78)
    imshow('Kaf.jpg');
```

```
   title('The character recognized');
end
if (j==79)|(j==80)|(j==81)|(j==82)
   imshow('Lam.jpg');
   title('The character recognized');
end
if (j==83)|(j==84)|(j==85)|(j==86)
   imshow('Meem.jpg');
   title('The character recognized');
end
if (j==87)|(j==88)|(j==89)|(j==90)
   imshow('Noon.jpg');
   title('The character recognized');
end
if (j==91)|(j==92)|(j==93)|(j==94)
   imshow('Ha.jpg');
   title('The character recognized');
end
if (j==95)|(j==96)
   imshow('Waow.jpg');
   title('The character recognized');
end
if (j==97)|(j==98)|(j==99)|(j==100)
   imshow('Ya.jpg');
   title('The character recognized');
end

if max<0.70 & max>0.50
   h=msgbox('most propably recognized                    !!','high level of noise');
elseif max<=0.50
   h=msgbox('may not recognized                          !!','low recognition rate');
end
```

APPENDIX C

The output values for the isolated Arabic characters are shown below.

0.98635	0.0068905	0.00040149	0.0042347	0.00053046	0.0034122	
	0.0054492	7.414e-005	0.0050998	0.0015895	0.0028136	
	8.3744e-005	0.0065452	0.00024348	0.00075735	0.0010613	
	0.00027291	0.0010842	0.00055783	0.003162	0.00015785	
	0.0013008	0.00016412	0.0017888	4.1329e-005	0.0010375	
	0.0014482	3.0581e-005				
0.0055829	0.9881	0.0055621	8.1853e-006	0.00080755	8.2518e-005	1.461e-
005	7.4404e-005	0.00026784	8.278e-005	0.0017519	0.0013186	
	6.3393e-005	0.0081623	0.0057246	0.0020957	5.8776e-005	
	6.1824e-005	0.0010041	8.0728e-005	0.00010257	0.00017886	
	0.00015267	0.00017447	0.00020746	4.7972e-005	6.9402e-006	
	0.001744					
0.001472	0.0052961	0.98787	0.0069382	0.00017173	0.0011032	
	0.0027725	0.0016275	2.4873e-005	0.0003821	0.00039234	
	0.00029317	2.4143e-005	0.00049426	0.00078692	0.00048662	
	0.00023002	0.0036683	0.0006297	0.0063726	0.00021189	
	1.1306e-005	5.9014e-005	0.0003722	0.0001948	0.0051917	
	3.6085e-005	0.00020406				
0.0021997	2.3196e-005	0.003486	0.98867	0.0054072	0.0064607	
	0.00079372	0.0030292	0.00019444	0.0017674	0.00022379	
	0.00034574	8.8731e-005	0.00095889	0.00050897	0.0042228	
	0.0016739	0.0010294	0.00066353	0.0001168	0.00016982	
	0.0025047	8.9423e-005	0.00025497	0.0027069	0.0013672	
	0.0036025	1.0879e-005				
0.0029103	0.00034486	0.00025126	0.00098333	0.98564	0.010704	
	0.0013065	4.3644e-005	0.0029978	0.0003705	0.00031705	
	0.00011003	7.0439e-006	0.00020095	0.0035339	0.0028326	
	0.003135	0.00074222	0.0025403	1.4957e-005	0.0031157	
	0.00085642	9.8773e-005	0.0056992	0.0030298	0.00092336	
	2.8762e-005	0.00073856				
0.0040847	0.0009696	0.0018529	0.0038048	0.010339	0.98454	
	0.0048001	0.00020315	0.0011203	0.004412	0.00018922	
	4.9872e-005	0.00078541	2.8708e-005	0.00014237	1.7308e-005	
	0.0047571	0.0058055	0.00015239	4.9722e-005	0.00033086	
	0.0004268	0.0029166	0.0032226	6.0635e-005	0.00034482	
	0.0016629	0.00050899				
0.00077465	0.00018	0.001092	0.00015434	0.0006154	0.0017357	
	0.98854	0.00054527	0.00066908	0.0011704	0.0032712	0.00012516
	0.00089879	0.00043536	0.00051702	0.00021198	0.00010389	
	0.0070069	0.0046304	0.0039049	0.00096038	7.4062e-005	
	0.0069263	0.00010144	0.00020292	0.00020513	4.4808e-005	
	0.0015562					
0.00015901	0.00025825	0.0058682	0.0023954	0.00469	0.002244	
	0.0027318	0.98874	0.0023134	0.00012261	0.00086816	0.0035481
	0.00054799	0.0005314	0.00011347	0.00014205	0.00017622	
	0.00040145	2.2731e-005	0.0046167	0.0025448	0.0027478	
	0.00022283	0.0012281	0.00080894	0.0046214	0.0027158	
	0.0007882					
0.0056891	0.0020881	1.0971e-005	0.00028836	0.0025238	0.0014612	
	0.00075233	0.002883	0.98829	7.865e-005	0.0065049	0.00012266
	0.00088732	0.00010759	4.1838e-005	0.0047045	0.002708	
	0.001237	0.0044344	0.00027729	0.0013612	0.0028036	
	0.0007908	0.00012157	0.001138	1.2654e-005	3.3649e-006	

Target file for beginning and middle characters (22 x 22)

```
1000000000000000000000
0100000000000000000000
0010000000000000000000
0001000000000000000000
0000100000000000000000
0000010000000000000000
0000001000000000000000
0000000100000000000000
0000000010000000000000
0000000001000000000000
0000000000100000000000
0000000000010000000000
0000000000001000000000
0000000000000100000000
0000000000000010000000
0000000000000001000000
0000000000000000100000
0000000000000000010000
0000000000000000001000
0000000000000000000100
0000000000000000000010
0000000000000000000001
```

- A comparison between the threshold and the recognition rate on non-noisy and noisy characters Respectively

Non-noisy characters

Threshold	Recognition Rate
0.85	100 %
0.90	98 %
0.95	97%
0.98	95 %

Noisy Characters

Threshold	Recognition Rate
0.70	91 %
0.75	90 %
0.80	88 %
0.85	84 %