Republic of Iraq

Ministry of Higher Education

and Scientific Research

Al-Nahrain University

College of Science

Printed Arabic Character Recognition Using Neural Network

A Thesis

Submitted to the College of Science, Al-Nahrain University as a Partial Fulfillment of the Degree of Master of Science in Computer Science

By

Huda Mahdi Abass

(B. Sc. 2006)

Supervisors

Dr. Ban N. Dhannoon. **Dr. Haithem Al-Ani**

2011 **1432**

بسم الله الرحمن الرحيم

و يسئلونك عن الروح قل الروح من

أمر ربي وما أوتيتم من العلم إلا

قليلا

صدق الله العظيم

سورة الاسراء/ الاية 85

Supervisor Certification

We certify that this thesis was prepared under our supervision at the department of Computer Science / College of Science /Al- Nahrain University, by **Huda Mahdi Abass** as partial fulfillment of the requirements for the degree of master of Master of Science in Computer Science.

Signature:

Name: Dr. Ban N. Dhannoon.

Title: Assistant Professor

Date: / /2011

Signature:

Name: Dr. Haithem Al-Ani.

Title: lecturer

Date: / /2011

In view of the available recommendations, we forward this thesis for debate by the examination committee.

Signature:

Name: Dr. Haithem Al-Ani

Title: Head of the Department of Computer Science,

Al-Nahrain University.

Date: / / 2011

Certification of the Examination Committee

We certify that we have read this thesis and as an examining committee, examined the student in content and what is related to it and that in our opinion it meets the standard of a thesis the degree of Master of Science in Computer Science.

Signature :
Name: Dr. Abdul-Moneem S. Rahma
Title: **Professor**
Date: / / 2011
(**chairman**)

Signature :
Name: Dr. Jane Jaleel Stephane Thamer
Title: **Assist Professor**
Date: / / 2011
(**member**)

Signature :
Name: Dr. Sawsan Kamal
Title: **Lecturer**
Date: / / 2011
(**member**)

Signature :
Name: Dr. Ban N. Dhannoon
Title: **Assistant Professor**
Date: / / 2011
(**supervisor**)

Signature :
Name: Dr. Haithem Al-Ani.
Title: **lecturer**
Date: / / 2011
(**supervisor**)

Approved by the dean of the college of science, Al-Nahrain University.

Signature:
Name: **Dr. Laith Abdul Aziz AL-Ani**
Title: **Assistant Professor**
Date: / / 2011
(Dean of College of Science)

الإهداء

إلى من أفنى عمره في العطاء الدائم، إلى الذي أرشدني و وقاني من زلاتي حتى أصبحت ما أنا عليه . . . . والدي الغالي

إلى من كان دعاؤها سر نجاحي وشمعة طريقي،عنوان التضحية ونبع العطاء . . . .والدتي الغالية

إلى نسيم الحياة الجاري، إلى من قاسموني الحياة، الورود العطرة التي أمدتني بعبق الحياة الجميل . . . . إخواني الأحباء

إلى بهجة النفس والروح، إلى رفاق الدرب الطويل . . . .أصدقائي الأوفياء

إلى حملة العلم والقيم . . . . أساتذتي الأعزاء

Acknowledgement

Before anything

Praise be to Allah for all things

I would like to express my sincere gratitude and appreciation to my supervisors Dr. Ban N. Dhannoon and Dr. Haithem Al-Ani for their guidance, supervision and efforts during the course of work.

My deep gratitude to the employees and stuff of Computer Science Department.

Thanks to my parents, sisters and my brothers for their help and patience during this work.

Finally thanks to all my friends for supporting and giving me advices.

Huda

Abstract

Character recognition has been an active area of pattern recognition, not only because it improves man machine communication, but also because it provides a solution for processing large volumes of data automatically. Many systems of characters [e.g. Latin, Chinese, Korean, etc.] have been investigated for automatic recognition using both off-line (i.e. digitized images) and on-line data acquisition techniques.

The purpose of this work is to recognize different forms of printed arabic characters written in three different fonts(Times new roman, Arial and Tahoma) using back-propagation neural network.

The character recognition process must be proceeded by many necessary process to improve and prepare the character image to be ready for feature extraction. The processes include removing the noise if any , apply image binarization by using global threshold and then specifying the boundaries of the character in the image which is followed by another binarization process using local threshold. The character is then thinned using a two- step thinning process, the preprocessed character is input to feature extraction stage which determine the features of the character out of 100 different features which where defined for the whole set of arabic characters and the character features are used in three layer back-propagation neural network for character classification.

This work was tested on a sample of printed characters and the correct average recognition rate was 97%.

Table of Contents

Table of contents

List of Abbreviations

ACRS	**Arabic Character Recognition System**
ASCII	**American Standard Code for Information Interchange**
NN	**Neural Network**
OCR	**Optical Character Recognition**

Chapter One

General Introduction

Chapter one

General introduction

1.1 Introduction

Character Recognition systems offer potential advantages by providing an interface that facilitates interaction between man and machine. Some of the application areas where Optical Character Recognition (OCR) plays a vital role include archiving documents, automatic verification of checks and data entry**.**

In the past two decades, valuable work has been notice in the area of character recognition, and a large number of technical papers and reports were devoted to this topic. Several recognition techniques have been used over the past few decades by many researchers**.** These techniques were applied for the automatic recognition of both printed and hand written characters. Immense research has been expanding on the recognition of Latin, Chinese and English characters. Against this background, only few papers have been addressed to the problem of Arabic character recognition**.** One of the main reasons for this is that, characteristics of the Arabic language do not allow direct implementation of many algorithms used for other languages having English or Chinese like characters [Naw03]

In general, the recognition of handwritten digitized cursive script is known to be the most challenging problem due to the variation in written styles and lack of dynamic information related to the shape of the pattern [Zer03].

Arabic writing is similar to English in that it uses letters (which consists of 28 basic letters), numerals, punctuation marks, as well as spaces and special symbols. It differs from English, however, in its representation of vowels since Arabic utilizes various diacritical markings.

Arabic letter might have up to four different shapes, depending on its relative position in the text**,** for instance, the letter (غ) has four different shapes: at the beginning of the word (preceded by a space), in the middle of the word (no space around it), at the end of the word (followed by a space), and in isolation (preceded by an unconnected letter and followed by a space). Arabic writing is cursive and is such that words are separated by spaces. However, a word can be divided into smaller units called sub words (A portion of a word including one or more connected characters).

Some Arabic characters are not connectable with the succeeding character. Therefore, if one of these characters exists in a word, it divides that word into two sub words[Adn97].

These characters appear only at the tail of a sub word, and the succeeding character forms the head of the next sub word. Figure **1** shows five Arabic words with one, two, and more sub words. The first word consists of one sub word that has three letters; the second has two sub words with two and one letter, respectively; the third word contains three sub words each consisting of only one letter; the fourth word contains

four sub words each consisting of only one letter and the fifth word contains five sub words each consisting of only one letter.

داران روان دار باب قفص

(a) (b) (c) (d) (e)

Figure l:Arabic words with constituent sub words.

Among the many applications that have been proposed for Neural Networks, character recognition has been one of the most successful. Many neural network architectures have been used in OCR implementation [Naw03]. Neural networks have been applied to various pattern classification and recognition problems for their learning ability, discrimination power, and generalization ability[Nei02].

1.2 Literature survey

Many researchers have considered OCR as headline in their work; some of their published works are the following:

- **[Meh96]** *(***Recognition of Multifont Farsi / Arabic Characters Using a Fuzzy Neural Network***)* an algorithm was developed for recognition of printed Farsi characters with various fonts, irrespective of size, rotation and stork. The system uses Pseudo-Zernike Moments as input features and the classifier consists of a complex of neural networks NN) and fuzzy neural network (FNN). The advantage of using FNN i**s** it's ability to classify similar patterns. The performance of the system is evaluated on a database consisting of more than 3700 character samples. The achieved accuracy is 99.85%

♦ **[Nal97] (Application of Artificial Neural Network Model for Optical, Character Recognition)** An application of artificial neural network approach for optical character recognition (OCR) was used. They examine a simple pattern recognition system using artificial neural network to simulate character recognition. A simple feed-forward neural network model has been trained with different set of noisy data. The back-propagation method is used for learning in neural network. The experiment result shows 70% recognition rate for noisy data up to 99% .

♦ **[Adn99] Recognition of Printed Arabic Words with Fuzzy ARTMAP Neural Network** This paper presents a new method for the recognition of Arabic text using global futures and Fuzzy ARTMAP neural network. The method is divided into three major steps**.** The first step is digitization and pre-processing to create connected component.

The second step is concerned with feature extraction, where global features of the input word are used to extract features such as number of sub words, number of peaks within the sub word, number and position of the complementary character, etc., to avoid the difficulty of segmentation stage. The third step is the classification and is composed of a single Fuzzy ARTMAP. The method was evaluated with 3255 images of 217 Arabic words with different fonts (each word has 15 samples), and the mean correct classification rate was 95.25%.

♣ **[Sam99] (Neural Network Recognition of Hand-Printed Characters)** The technique adopted in this paper for recognizing hand-printed Latin characters using neural network combined with a structural approach. The method is based on structural primitives such as carves, straight lines and loops, in a manner similar to that in which human

beings describe characters geometrically. The system was tested on a sample of handwritten characters from several individuals whose writing ranged from acceptable to poor in quality and correct average recognition rate obtained using cross-validation was 86%.

♣ **[Naw03] (An Approach to Off-line Arabic Character Recognition using Neural Networks)** This paper presents a technique for the automatic recognition of Arabic Characters. The technique is based on Neural Pattern Recognition Approach. The main features of the system are preprocessing of the text, segmentation of the text to individual characters, Feature extraction using centralized moments technique and recognition using RBF Network.

♦ **[Fen07]** (**Character Recognition Using Parallel BP Neural Network**) A novel character recognition method of license plate number based on parallel BP neural networks were presented. This will enhance the accuracy of the recognition system that aims to read automatically the Chinese license plate. In the proposed methodology, the character is binarized and the noise is eliminated in the preprocessing stage, then the character feature is extracted by using skeleton, and it normalized to size 8*16 pixels. Finally, the character feature is put into the parallel neural networks and the character is recognized. The proposed method in character recognition is effective, and promising results have been obtained in experiments on Chinese license plates.

♦ **[Yua08]** (**Augment Document Image Binarization by Learning**): In this research, a binarization task has two stages: learning stage and performing binarization stage. The document property

knowledge, learned in first stage, is used to evaluate binarization. They are fed back to binarization to optimize it by adjusting the binarization parameters.

The binarization results are improved and the diversity among binary results shrinks comparing with the conventional methods. As a whole, the character recognition performance will be improved.

♣ **[Rag08] (Arabic Character Recognition Based on Moments Method)** : This research perform the moment method to recognize the characters. Moments and functions of moments have been extensively employed as invariant global features of images in characters recognition. Recognition system has been performed on the printed Arabic characters and the percentage accuracy for recognize 65 printed characters was 96.92307% while in hand written characters the recognition percentage is decreased, due to irregularities appears in the handwritten characters.

1.3 Aim of Thesis

An adaptive character recognition system is implemented to recognize different fonts and forms of different printed arabic characters using back-propagation neural network.

1.4 Thesis Layout :

The work outline is divided into the following chapters :-

- ***Chapter one :***

  In this chapter a general overview has been produced for Arabic Character Recognition. Some of the related work and objectives of the work have been listed.

- ***Chapter two :***

  This chapter describe in some details the basic concepts of recognition and discuss the types of this approach.

- ***Chapter three :***

  The details of algorithms which are used in this work will be presented.

- ***Chapter four :***

  It includes the results that obtained from the used methods .

- ***Chapter five:***

  In this chapter, the conclusions and suggestions for future work will be presented.

Chapter Two

Theoretical Overview

Chapter Two

Theoretical Overview

2.1 Introduction

Optical character recognition is a process of converting a printed document or scanned page into ASCII characters that a computer can recognize.

Computer systems equipped with such an OCR system improve the speed of input operation, decrease some possible human errors and enable compact storage, fast retrieval and other file manipulations. The range of applications include postal code recognition, automatic data entry into large administrative systems, banking, automatic cartography and reading devices for blind. Accuracy, flexibility and speed are the main features that characterize a good OCR system [Hab94].

Document usually contains words each word consist of characters. In order to recognize a word it first should be isolated from the document then split into its characters, the image of the characters required a set of processes in order to recognized it.

The current work steps for recognizing the character is considered. Other activities are not considered in the current work.

2.2 The characteristics of Arabic characters

Some Arabic characters characteristics are [Maj96]:

1) In Arabic alphabet, there are 28 isolated characters, each of which has two to four representations depending on the position (beginning, middle, end, or isolated) in the word, which increases the number of classes from 28 to 100.

2) Generally speaking, different Arabic characters have different sizes.

3) Characters may have a dot, two dots, or three dots, which are called secondary part of the character, associated with its main part, and can be above, below, or even inside the character.

4) There are pairs and groups of characters, which differ only by the secondary parts.

2.3 Image Enhancement

The principle objective of the enhancement techniques is to process a given image so that the result is more suitable than the original image for a specific application [Pra78].

The range of applications include using enhancement techniques as preprocessing step to ease the next processing step or as post processing step to improve visual perception of a processed image, or image enhancement may be an end in itself [Umb98].

The enhancement needed in the present work was the smoothing of the images. By smoothing an image, this process mean removing or reducing the noise in the image. Many smoothing techniques are available, like smoothing by averaging, smoothing using median value, etc.

The smoothing technique adopted in the present work is the median technique because of its high ability in removing noise while preserving sharp signals like edges [Gon94].

2.3.1 Median Filtering

Median filter is one of the best edges preserving smoothing filters. It consists of a sliding window encompassing an odd number of pixels. The center pixel in the window is replaced by the median of the pixels within the window[Pra78].

The median m of a set of values is located such that half of the values in the set are less than m and half are greater than m . In order to perform median filtering in a neighborhood of a pixel; first sort the values of the central pixel and its neighbors, determine the median, and assign this value to the central pixel.

The principal function of the median filter is to force points with very distinct intensities to be more like their neighbors, thus actually eliminating intensity spikes that appear isolated in the area of the filter mask [Gon87].

2.4 Binarization

Binary images are the simplest type of images that take one of two values, typically black and white, or '0' and '1'. A binary image is referred to as a 1 bit/pixel image because it takes only 1 binary digit to represent each pixel.

This type of image is most frequently used in computer vision application where only information required for the task is general shape, or outline, information [Umb97].

Image binarization is a classical and foundational problem in image process. Especially in OCR task, it affects the character recognition performance directly. Designing binarization method according to task or goal-directed binarization is a good means to improve binarization. In many practical applications, it needs to process a batch of same-type documents, which have similar properties, such as font, character size, layout, style.

A gray level image may be converted to a binary image by thresholding process. The process of converting the image into binary mode is called binarization. There are two famous categories of binarization method : Global methods and Local methods.[Yua08]

Global methods treat each pixel independently, converting each to black or white based on a single threshold value. If a pixel's color intensity is higher than the global threshold it is assigned one value, otherwise it is assigned the opposite value. In contrast local methods, make use of the color information in nearby pixels to determine an appropriate threshold for a particular pixel. The goal of the binarization is to separate the character from the background in the gray image and reduce the image color into black and white[Sna04].

The binarization arithmetic is shown in Eq.(2.1), where f(x ,y) is the original character image, g(x, y) is the binarized image and T is the threshold , x, y are the location of the pixels been considered.

if $f(x, y) \geq T$ *then* $g(x, y) = 1$ otherwise g(x ,y)=0 (2.1)

From the equation one can see that the key problem in binarization is how to choose the threshold . [Yua08]

2.4.1 Problems with Thresholding

The major problem with thresholding is that we consider only the intensity, not any relationships between the pixels.

There is no guarantee that the pixels identified by the thresholding process are contiguous. One can easily include extraneous pixels that aren't part of the desired region, and he can just as easily miss isolated pixels within the region (especially near the boundaries of the region).

These effects get worse as the noise gets worse, simply because it's more likely that a pixels intensity doesn't represent the normal intensity in the region. When we use thresholding, we typically have to play with it, sometimes losing too much of the region and sometimes getting too many extraneous background pixels[Gon01].

2.4.2 Local Thresholding

Another problem with global thresholding is that changes in illumination across the scene may cause some parts to be brighter (in the light) and some parts darker (in shadow) in ways that have nothing to do with the objects in the image. Anyone can deal, at least in part, with such uneven illumination by determining thresholds locally.

That is, instead of having a single global threshold, the threshold itself allow to smoothly vary across the image [Sna04].

2.4.3 Automated Methods for Finding Thresholds

To set a global threshold or to adapt a local threshold to an area, we usually look at the histogram to see if we can find two or more distinct

modes one for the foreground and one for the background. Recall that a histogram is a probability distribution:

$$p(g) = ng/n \qquad (2.2)$$

That is, the number of pixels *ng* having grayscale intensity *g* as a fraction of the total number of pixels *n* [Gon01].

2.5 Thinning

It is well known that one of the key issues in pattern recognition area is how to extract distinctive features from the input patterns. A pattern itself consists of lots of data bits, many of which can be deleted from the image without distorting the information. In other words, through the process called "thinning", the number of data of an input pattern can be significantly reduced without losing any stroke features [Kim92].

One of the most crucial phases in the process of text recognition is skeletonization (or thinning). It is a procedure which transforms a pattern to a skeleton of a unit width. It has been proven effective in a broad range of image processing applications, such as character recognition, inspection of printed circuit boards, chromosome shape analysis, reduction of the required memory space for storing the essential structural information of the patterns, and reduction of the required computational time. Although the design of thinning algorithms has been a very active research area, few researchers addressed the thinning of Arabic characters.

Due to the characteristics of Arabic characters, they do not allow direct application of the techniques developed for other languages such as Latin and Chinese; [Maj95].

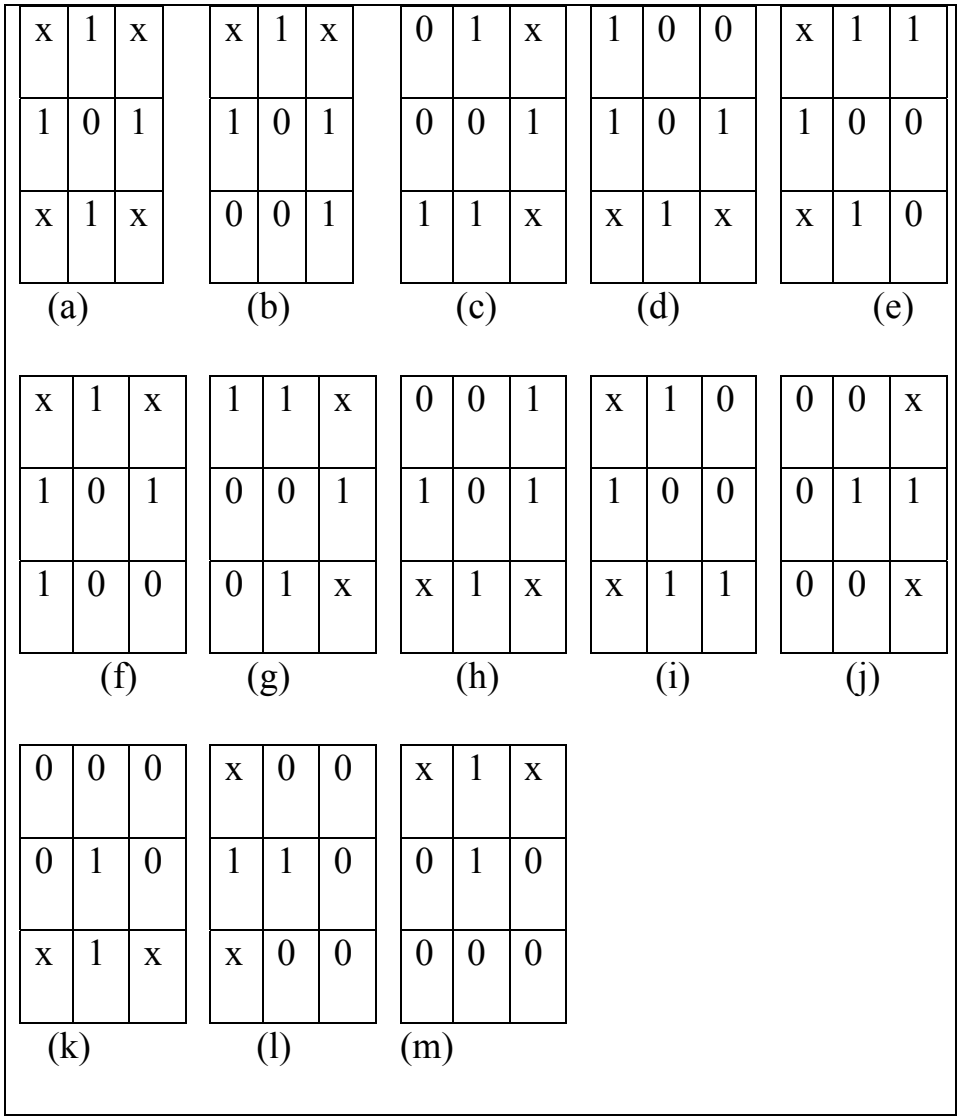
Thinning was introduce to describe the global properties of objects, and to reduce the original image into a more compact representation.

The skeleton expresses the structural connectivity's of the main components of an object and it has width of one pixel in the discrete case. These kinds of techniques have a wide range of applications, so for example skeleton has been applied successfully in solving character recognition problems [Pra78].

Building the object skeleton can be achieved in three different steps, initial smoothing, skeleton, and finial smoothing.

2.5.1 Initial Smoothing [ken79]

Smoothing of an input image is to fill up small holes and to remove a small spurs. A common approach is to use template matching where the center pixel is processed when the template find to match in the image. The templates are as shown in figure(2-1), the symmetric patterns have also been considered and deletion patterns are merged to one pattern.



Figure(2-1)The pattern used in initial smoothing. The patterns (a-i) are used for filling, while patterns (j-m) are used for deleting .

In these templates the '0' represent the background(white) pixel, '1' represent a symbol (black) pixel, and the 'x' stands for 'don't care' pixel. The center pixels of the Figures (a-i) are filled, while the center pixels in Figures (j-m) are deleted.

2.5.2 Skeletonization

A skeleton may be defined as a connected set of medial lines along the limbs of a figure. Generally, for a skeletonization algorithm to be effective, it should ideally compress data and retain the significant features of the pattern. Although there is no formal definition of a skeleton, these appears to be a general agreement that a good skeletonization algorithm must meet the following requirements:[Hon04]

1. Preserve the connectivity of skeletons

2. Converge to skeletons of unit width.

3. Approximate the medial axis.

4. Achieve a high data reduction efficiency

Ideally a skeleton must topologically equivalent to the object. If the object is connected, then the skeleton must also be connected, it must run along the medial axis of the object and must be one pixel thick. The basic idea of the skeleton is to eliminate redundant information but retain only the topological information concerning the shape and structure of the object that can help with recognition[Hon04].

Thinning can be achieved either by skeletonization method or by medial axes approach .

- Thinning is perhaps the simplest approach to skeletonization. It may be defined as the process of systematically stripping away the outmost layers of a figure until only a connected unit-width skeleton remains.
- Medial axes approach is the set of all points having more than one closest point on the object's boundary. Originally referred to as the topological skeleton. In mathematics the closure of the medial axis is known as the cut locus[Kim92].

The polygonization is performed on the skeleton points to remove redundant points[Jiq00]. Which is the finial smoothing.

2.5.3 Final Smoothing

The resulting skeleton by the thinning algorithm is 8-connected, where non-intersection point may have a 4-neighbor pixel or an 8-neighbor pixel which are adjacent. The smoothing templates shown in figure (2-2) are used to delete the 4-neighbor pixel (the center pixels in these patterns are deleted). After the conversion a non-intersection point has only two black pixels in its 8-neighbors. This conversion will increase the speed of recognition algorithm and improve the recognition performance.[Jia97]

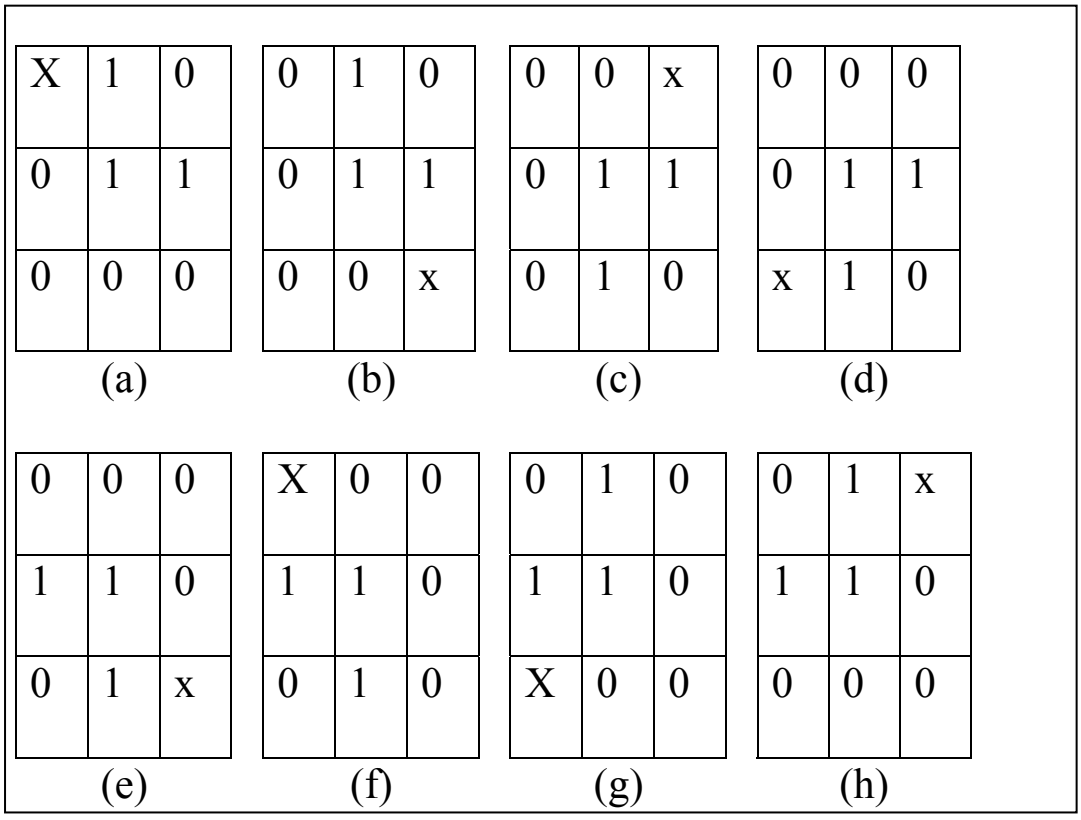


Figure (2.2) Smoothing templates used in boundary pixel check

2.6 Character Extraction

Isolated characters are extracted from original image by detecting the character boundary. First, a line of characters detection has been performed by scanning the input image horizontally. Two lines can be separated easily because of existing blank rows between them. Then for each line of characters a vertical scanning is performed in similar fashion to separate the characters [Abd07].

2.7 Feature Extraction

In order to increase the efficacy of the recognition process, the feature of characters should be extracted and used instead of the hole character matrix.

The main task of image analysis system is to extract information useful for solving application based problems.
Feature extractions refer to the process of finding a mapping that reduces the dimensionality of the pattern by extracting some numerical measurements from raw input pattern.

There are many different feature-extraction definitions, each attempted by some computer vision researchers; all of them carry the same idea. The following are some of these definitions:

- ♣ It is the process of reducing the representation of an image to a small number of components carrying enough discriminating information [Gon87].
- ♣ It is the process of finding a mapping that reduces the dimensionality of the patterns by extracting some numerical measurements from raw input pattern [Set97].
- ♣ It is the process of forming a new (often smaller) set of features (refined representation) from the original feature set [Mao94].

Feature extraction can avoid the curse of dimensionality, improve the generalization ability of classifiers, and reduce the computational requirements of the classifier [Mao94].

There are two methods used in feature extraction processes, analysis method (which is used in the present work) and mathematical method. Where analyses method depend on the relations between the pixels in the character and mathematical method (like moment method) were have been used in finding the location and orientation of an object [Kep90]

2.8 Neural Network

A neural network is a massively parallel distributed process made up of simple processing units called neuron, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

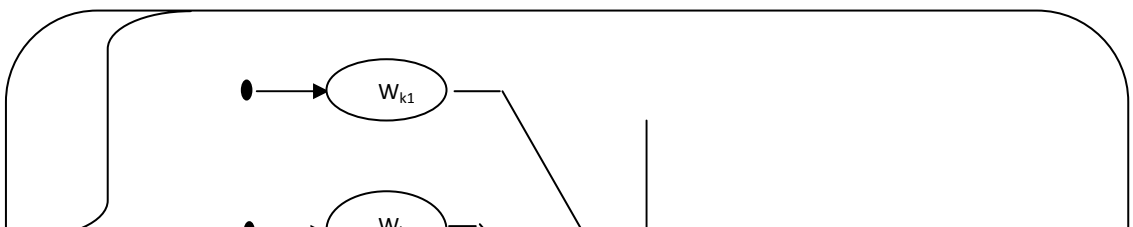
1. Knowledge is acquired by the network from its environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The procedure used to perform the learning process is called the learning algorithm, the function of which is to modify the synaptic weights of the network in an orderly fashion to attain a desired objective [Gon01].

2.8.1 Models of a neuron

A neuron is an information-processing unit that is fundamental to the operation of neural network. Figure (2.3) shows the model of a neuron, which forms the basis for designing (artificial) neural networks. Three basic elements of the neuronal model illustrated below [Ben96]:

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. Specifically, a signal x_i at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. It is important to make a note of the manner in which the subscripts of the synaptic weight w_{kj} are written. The first subscript refers to the neuron in question and the second subscript refers to brain.
2. artificial neuron may lie in a range that includes negative as well as positive values.
3. An adder for summing the input signals, weighted by the respective synapses of the neuron; the operations described here constitute a linear combiner.
4. An activation function for limiting the amplitude of the output of a neuron. The activation function is also referred to as a squashing function in that it squashes (limits) the permissible amplitude range of the output signal to some finite value.



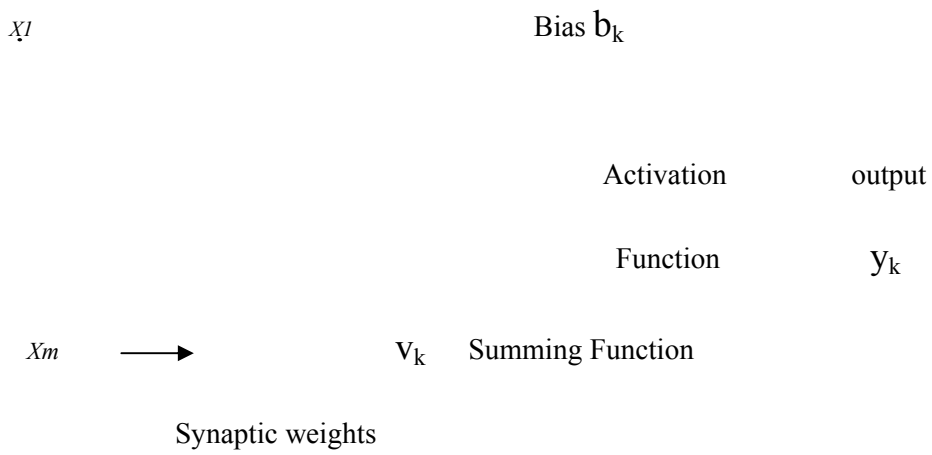


Figure (2.3) Model of neuron

2.8.2 Activation and Output Rules[Ben96]

Neural network also need a rule which gives the effect of the total input on the activation of the unit. A function F_k which needs is take the total input $s_k(t)$ and the current activation $y_k(t)$ and produces a new value of the activation of the unit k:

$$Y_k(t+1)=F_k(y_k(t),s_k(t)). \qquad (2.3)$$

Often, the activation function is a non decreasing function of the total input of the unit:

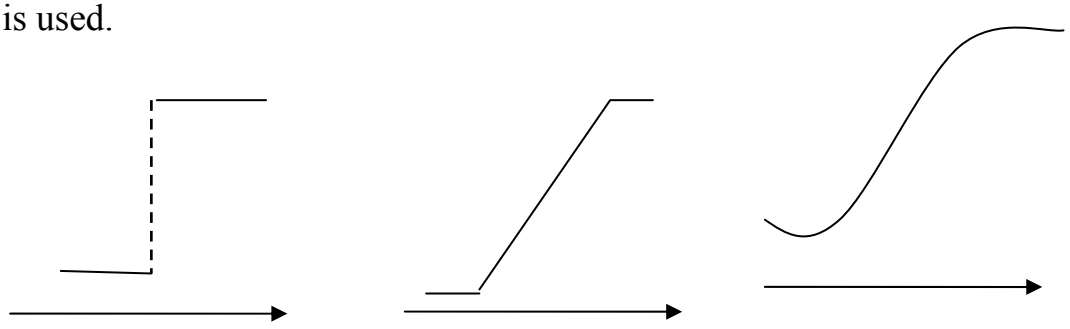
$$y_k(t+1) = f_k\big(s_k(t)\big) = f_k\left(\sum j\, w_{jk}(t)y_j(t) + y_j(t) + \theta_k(t)\right) \qquad (2.4)$$

Generally, some sort of threshold function is used these functions are : a hard limiting threshold function (a *sin* function), or a linear or semi-linear function, or a smoothly limiting threshold (see figure 2.4).

For this smoothly limiting function often a sigmoid (S-shaped) function like

$$y_k = F(s_k) = \frac{1}{1+e^{-s_k}} \qquad (2.5)$$

is used.



sin semi-linear sigmoid

Figure (2.4) Various activation functions for a unit.

In some cases, the output of a unit can be a stochastic function of the total input of the unit. In that case the activation is not deterministically determined by the neuron input, but the neuron

input determines the probability p that a neuron get a high activation value:

$$P(y_k \leftarrow 1) = \frac{1}{1+e^{-sk}} \qquad (2.6)$$

2.8.3 Network Architecture

Generally, neural networks can be categorized into 2 main types, namely supervised networks and unsupervised networks. The way the network architecture was designed has taken the ability of its training algorithm into account. In most newly proposed network topologies, the design of their corresponding training algorithms are deemed essential.

Apparently, a successful network architecture must be supported by an effective and simple enough training algorithm[Jam91].

- Supervised neural networks[Siu07]

Supervised neural networks are the mainstream of neural network development. The differentiable characteristic of supervised neural networks lies in the inclusion of a teacher in their learning process. The basic block diagram of supervised learning for all neural network models can be described through figure 2.5. For the learning process, the network needs training data examples consisting of a number of input-output pairs.

The desired output vector in the training data set serve as a teacher for network's learning.

In the training process, the error signals are constructed from the difference between the desired output and system output.

Through iterative training procedure the network's weights are adjusted by the error signal in a way that the network output tries to follow the desired output as close as possible.

The supervised network's architectures can vary depending on the complexity and nature of data to be handled.

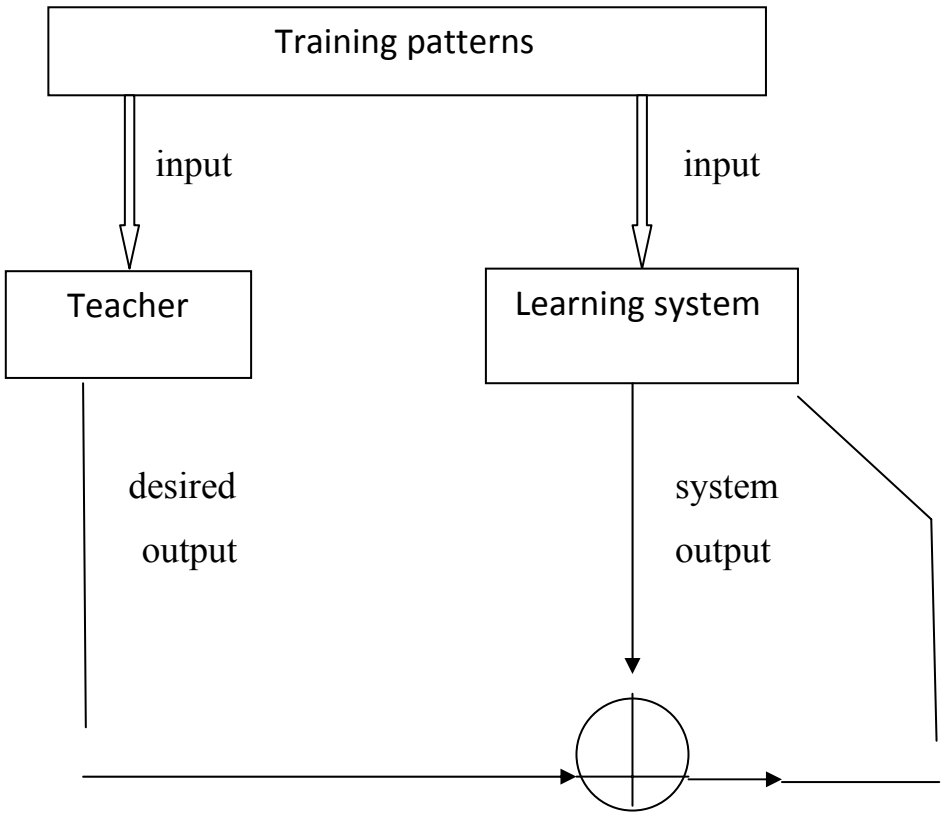


Figure 2.5 Overview of supervised learning

- Unsupervised neural networks

Unlike the supervised networks, unsupervised networks do not have a teacher in the training data set. The learning process of unsupervised neural networks is carried out from a self-organizing behavior. In the course of training, no external factor is used to affect the weights adjustment of the network. The correct outputs are not available during the course of training. For instance, a typical unsupervised network consists of an input layer and a competitive layer. Neuron on the competitive layer compete with each other via a simple competitive learning rule to best represent a given input pattern[Gan04]. Through

competitive learning, the network output automatically reflects some statistical characteristics
of input data such as data cluster, topological ordering. An unsupervised system is represented in figure (2.6).

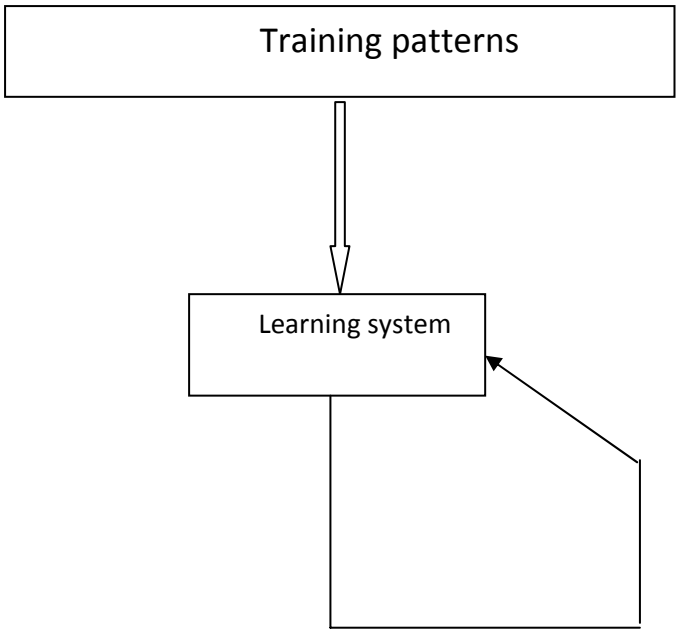


Figure (2.6) Unsupervised learning process

2.8.4 Back Propagation

An neural networks was found to be useful in addressing problems requiring recognition of complex patterns (like characters shape in our work) and performing nontrivial mapping functions is the back-propagation network (BPN)[Jam91].

The serious constraint of the back-propagation algorithm is that the function approximated should be differentiable. If the inputs and

desired outputs of a function are known then back-propagation can be used to determine weights of the neural network by minimizing the error over a number of iterations. The weight update equations of all the layers (input, hidden, output) in the multilayer perceptron neural network are almost similar, except that they differ in the way the local error for each neuron is computed [Gan04].

The back propagation network is a multilayer feed forward network with a different transfer function in the artificial neuron and a more powerful learning rule. The learning rule is known as back propagation, which is a kind of gradient descent technique with backward error (gradient) propagation, as depicts in fig 2.7. The training instance set for the network must be presented many times in order for the interconnection weights between the neurons to settle into a state for correct classification of input patterns. The back-propagation network in essence learns a mapping from a set of input patterns (e.g., extracted feature) to a set of output patterns (e.g., class information) [Lim94].

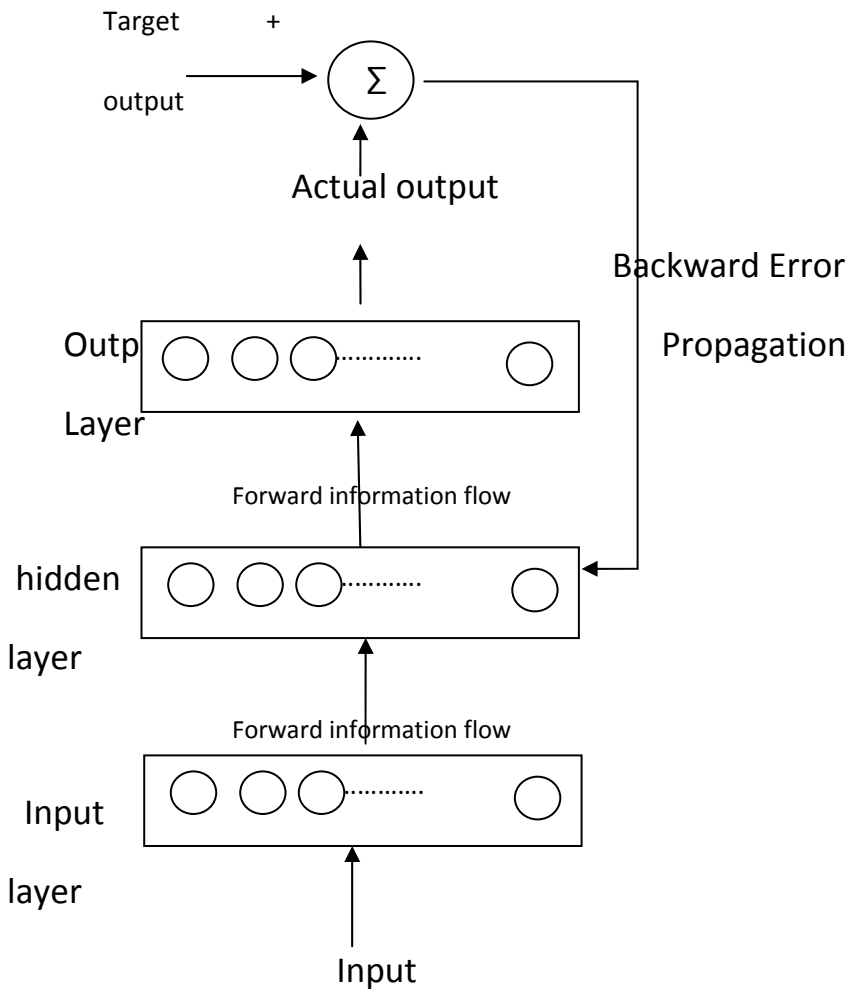


Figure (2.7) The back-propagation network

Chapter Three

Character Recognition System

Chapter Three

Character Recognition System

3.1 Introduction

Surprising number of everyday problems is difficult to solve by traditional algorithms. A problem may qualify as difficult for a number of different reasons; for example, the data may be too long, too noisy, irregular; the problem may be difficult to model, or it may simply take too long to solve.

It is easy to find examples: finding the shortest path connecting a set of cities, dividing a set of different tasks among a group of people to meet a deadline, or fitting a set of various sized boxes into fewest trunks. In the past, programming might have carefully hand crafted a special-purpose program for each problem; now they can reduce their time significantly by using neural network [Gra95].

In this work, the algorithms that are implemented by building a software system named "Arabic Character Recognition System (ACRS)". In "ACRS", a Neural Network (NN) is used to recognize letters.

3.2 System Design and Structure of " ACRS"

The structure of ACRS is shown in figure (3.1), which attempts to recognize separated Arabic Characters using NN.

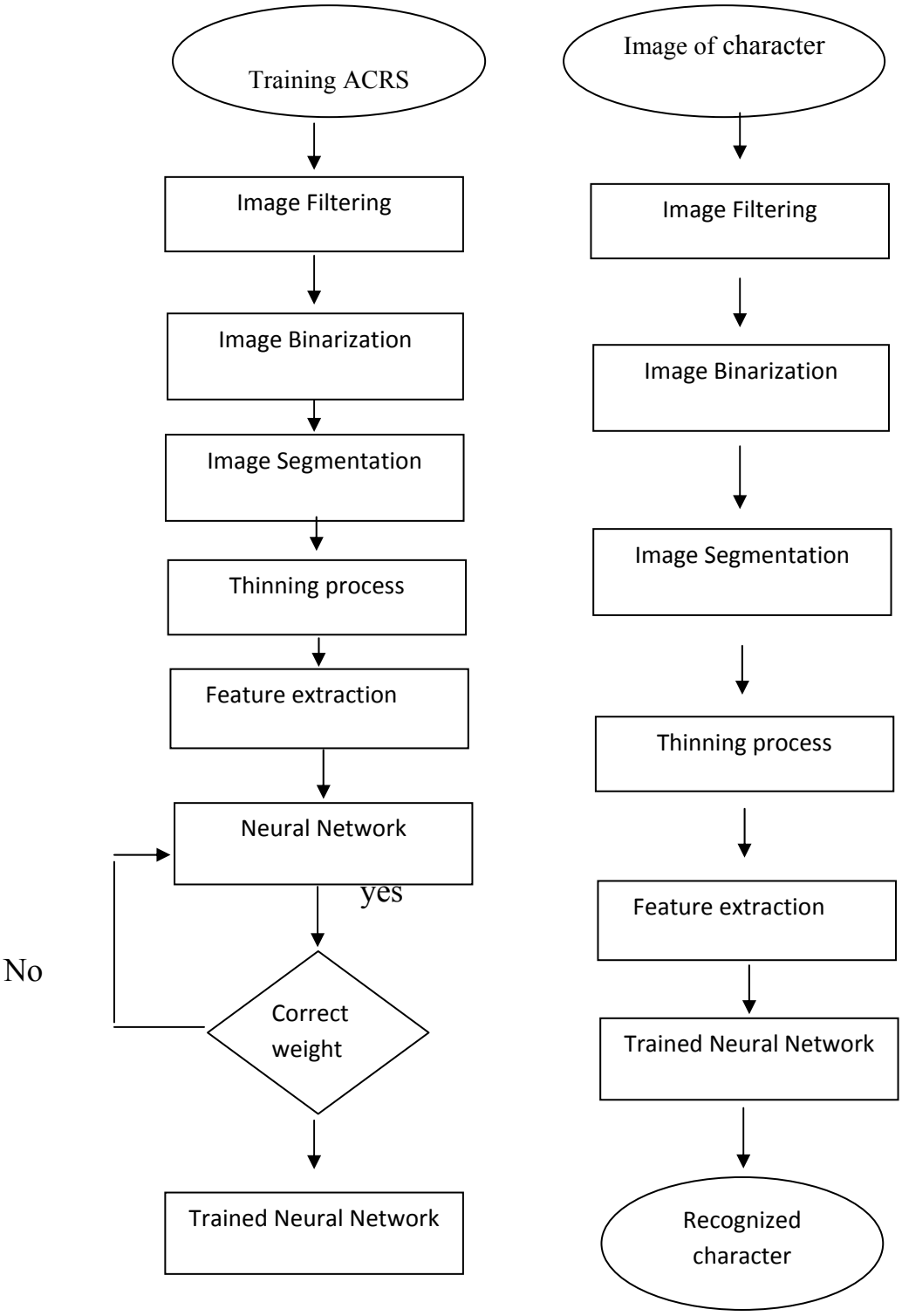


Figure (3.1) Structure of arabic character recognition system

3.3 Image Filtering

In this work the *median filter* has been used, which, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median). Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of impulse noise, also called salt-and-pepper noise because of its appearance as white and black dots superimposed on an image. In order to perform median filtering at a pixel in an image, the following steps will be used:

1- First take 3*3 window
2- Sort the values of the pixel in window and its neighbors
3- Determine their median, and assign this value to the central pixel.

Note that the median, j, of a set of values is such that half the values in the set are less than or equal to j, and half are greater than or equal to j.

3.4 Image Binarization Algorithm

The binarization stage plays an important role in (ACRS), which can affect the accuracy of character recognition.

There are two types of binarization depending on the way that threshold calculated, they are:

- local threshold
- global threshold

In this work, two methods have been used. The first is global threshold which was used before segmentation process. This method have benefits and worse one of its benefits is that this method work on a whole picture (if the picture contain one character or more it will not effect on the work of this method). But because of the problems shown in 2.4.1. The second method (local threshold) was used on the filtered image before thinning process .

Local threshold depends on the histogram output value. The histogram of an image is a plot of the gray level values versus the number of pixels at that value. The shape of a histogram provide information about the nature of the image, or sub-images if it was considering object within the image. The histogram provides us with statistical features. These statistical features provide us with information about the characteristic of the gray level distribution for the image or sub-image.

Algorithm (3.1) illustrates the Global threshold steps. Figure (3.2) explain an example of performing binarization algorithm using global threshold on filtered image and resulted of binarized image.

Algorithm (3.1) global threshold

Input: Filtered image
Height of the image
width of the image

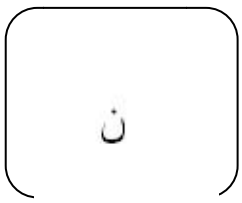
Output : Threshold

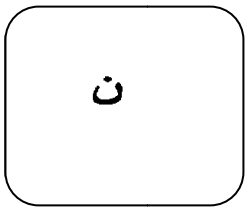
Steps :

Step1**:** calculate threshold using the following equation

Threshold (average)=(Max color value + min color value) div 2

Step2: loop from 0 to height
Step3**:** loop from 0 to width
Step4: check if pixels in image is > average
then increase counter1 and add the pixel value to add1
Else increase counter2 and add the pixel value to add2
Step5: calculate the average of max value
Step6: calculate the average of min value
Step7: calculate threshold by using the following equation
average=(0.5* (average of max + average of min))





(a) (b)

Figure (3.2) (a) the filtered image

(b) the binarized image

3.5 Segmentation algorithms

In this work the letter have been segmented (determining the boundaries of letter) by using two algorithms. So it must perform the following :

1- Extract image data from the binarized image.
2- Determine the start and end row for each character in the pre-processed image by algorithm (3.2).
3- Determine the start and end column for each character in every line obtained from above step by algorithm (3.3).
4- Concatenation the output from algorithm(3.2) and algorithm (3.3).

The output of algorithm (3.2) is shown in figure (3.3) while figure (3.4) explains an example of algorithm (3.3) performance.

Algorithm (3.2) row extract

Input:

The binarized image
The height of the image
The width of the image

Output:

row-record which contain first pixels of both first and last line of the character in the image

Steps :

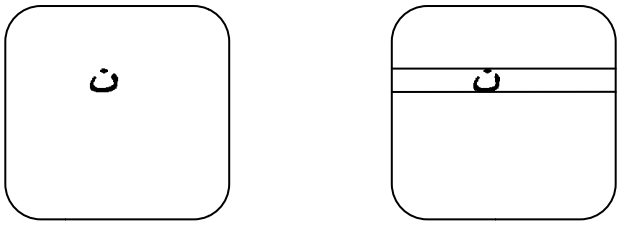
Remark : *This procedure use two sub-programs, the first search for the first black point in the picture while the second search for the first white point . they will be illustrate in this paragraph*

First sub-program (search for black point)

Step 1: loop from 0 to height
Step2: check pixel if it is black then store pixel row-location in row-record
Step3: call second sub-program and send the location of next pixel

second sub-program (search for white point)

step1: start from next black pixel in the image
step2: check if pixel is white then store the row-location of pixel



(a) (b)

Figure (3.3)

(a) the binarized image

(b) the row extracted image

Algorithm (3.3) Column extract

Input :

Binarized image and the row extracted image
Width of image

Output :

Column-record which contain began and end column of character in the image

Steps :

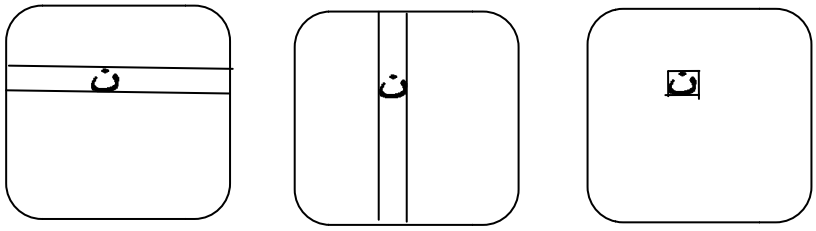
First sub-program (search for black point)

Step1: start from extracted row from algorithm (3.2)
Step2:search for black pixel and store the column location in record
Step3: call second sub-program

second sub-program (search for whit point)

step1: start search from next black pixel
step2**:** search for white pixel and store the column-location of white pixel

as a result of the two algorithms the axis of the upper left and lower right are specified.



(a) (b) (c)

Figure (3.4) (a) Row extracted image

(b) Column extracted image

(c) Finale segmented image

3.6 Thinning Algorithm

Thinning process convert binary shapes obtained from edge/boundary detection or thresholding to 1-pixel wide lines. For example, the threshold version of hand written or printed alphanumeric can be thinned for better representation and further processing. It is iteratively delete pixels inside the shape to shrink it without shortening it or breaking it apart. The method use in the present work is two-step method.

The actions of step 1 and step2 are explained in the following process:

- At first it checks the number of black pixels around the black pixel under consideration.

– If this number is less than 2 then it is not eligible to be removed. But for more than 2 numbers, it considers the number of white-black color combinations around the considered pixel.

– If this number is not equal to 1 then it does nothing. But if that number is 1, it performs one of two types of checking named "first type checking" and "second type checking".

- If (i,j) is considered black pixel figure (3.5) then according to the "first type checking", it would be white if (i,j+1) or (i+1,j) or both (i-1,j) and (i,j-1) are white and according to the "second type checking", it would be white if (i-1,j) or (i,j-1) or both (i,j+1) and (i+1,j) are white.

i-1,j-1	**i-1,j**	**i-1,j+1**
i,j-1	**i,j**	**i,j+1**
I+1,j-1	**i+1,j**	**i+1,j+1**

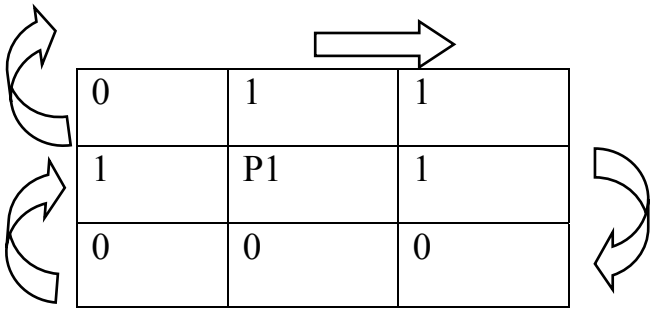
Figure (3.5) 3 x3 window frame of considered pixel and its surroundings 8 pixels

Some definitions: To decide whether a pixel P1 should be deleted, examine its 8 neighbors in the 3 by 3 neighborhood, p_2 , p_3 , p_4 , p_5 , p_6 , p_7 , p_8 , and p_9 .

- $N(P_1)$: number of non-zero neighbors:
  $N(P_1)= P_2+P_3+….+P_9$

P9	P2	P3
P8	P1	P4
P7	P6	P5

- $S(P_1)$: number of 0 to 1 (or 1 to 0) transitions in the sequence $(P_2,P_3,..,P_9)$.



Algorithm (3.4) illustrate the two step thinning algorithm steps and figure (3.6) explain an example of performing two step thinning algorithm on an image and resulted of thinned image.

Algorithm (3.4): Two Step Thinning Algorithm

Input :
The segmented image
The boundaries of character // *it consist four counters*
rs: refer to row start
re: refer to row end
cs: refer to column start
ce: refer to column end

Output :
new thinned image

Steps:
Step1**:** loop until (not change)
Step2: loop from rs to re
Step3: loop from cs to ce
Step4: Check if the pixel is black then for each pixel in the window 3x3 calculate np (no. of nonblack pixel) and sp (no. of transition)
Step5: mark all pixel satisfy all the following $(2 \leq N(P_1) \leq 6)$ AND $(S(P_1)=1)$ AND $(P_2 \cdot P_4 \cdot P_6=0)$ AND $(P_4 \cdot P_6 \cdot P_8=0)$ AND $(P_7 \neq 0)$ Then delete all marked pixels
Step6: mark all pixels that satisfy all the following $(2 \leq N(P_1) \leq 6)$ AND $(S(P_1) = 1)$ AND $(P_2 \cdot P_4 \cdot P_8=0)$ AND $(P_2 \cdot P_6 \cdot P_8=0)$ AND $(P_3 \neq 0)$ Then delete all marked pixels
End loop in step3
End loop in step2
End loop in step1

(a) (b)

Figure (3.6) (a) Segmented image (b) Thinned image

3.7 Feature extraction process

This process is dived into two phases :

- building the image pixels description table
- feature extraction.

Where building the image pixels description table is responsible to decide from which pixel feature extraction begin while feature extraction is responsible to extract the features from thinned image. The paragraph below explain in details the two phases.

3.7.1 Building the image pixels description table

The output of this algorithm will be stored in a record contains the following fields:

1- Pixel type.
2- Pixel location according to x (height) and y (width).
3- Used or not used (it is special field that advert if it's used or not in feature extraction procedure) it take one of the two values 0 not used or 1 used.
4- Pixels number refer to number of pixels in each letter.

A 3*3 window is used to trace along the path of the skeleton, recording the structural information of the trace path. The pixels types are explained in figure (3.7).

In this algorithm the pixels will be treated as follows :

1- Terminal if the pixel has one neighborhood
2- Connection if the pixel has two neighborhoods
3- Across if the pixel has three neighborhoods
4- Spilt point if the pixel has zero neighborhood

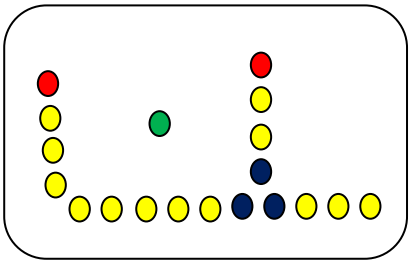
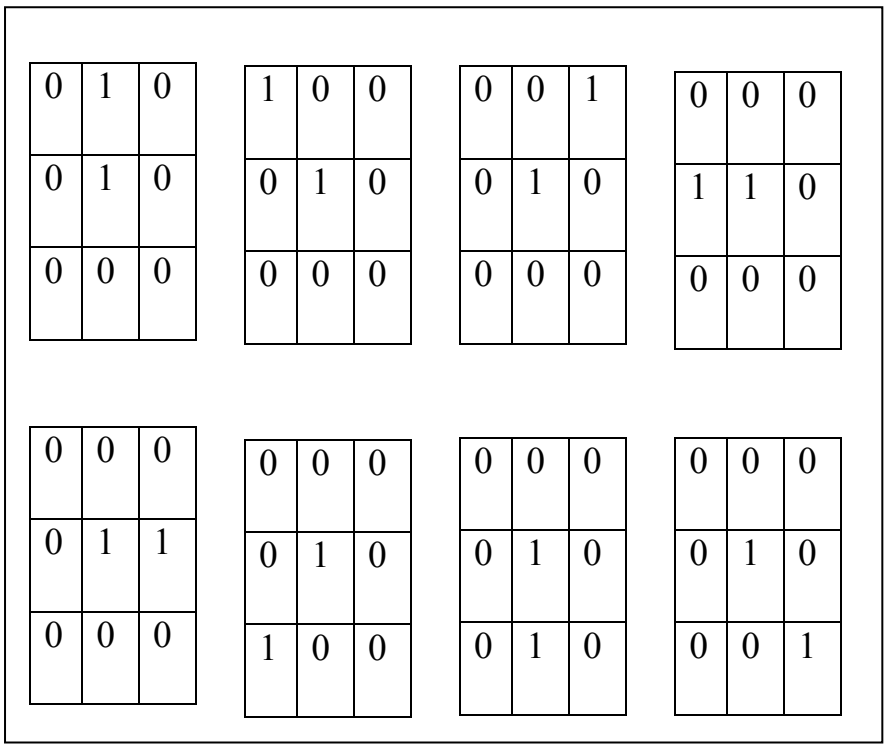
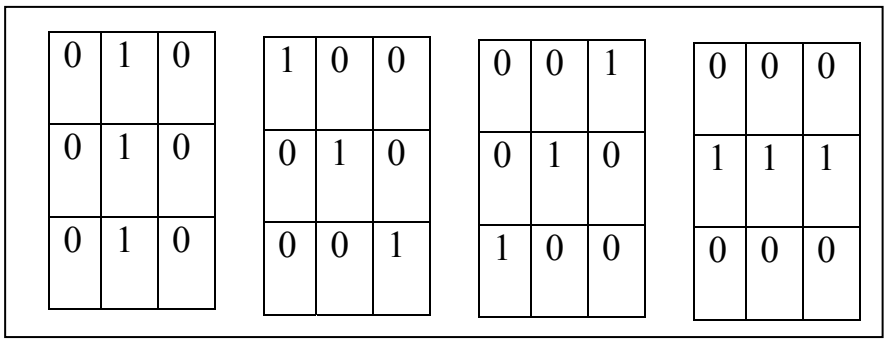


Figure (3.7) Represents the types of pixels

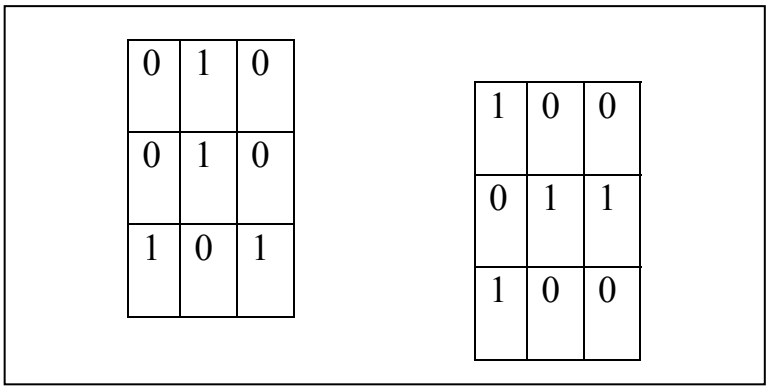
In the figure above, the pixels are represented as follows:



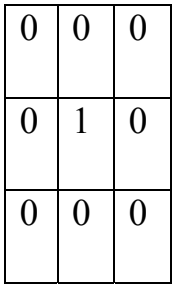
(a)



(b)



(c)



(d)

figure(3.8) a- Represent terminal pixels

b- Represent connected pixels

c- Represent across pixels

d- Represent split pixels

3.7.2 Feature Extraction Algorithm

The thinned letter will be divided into four quarters and use the image pixels description table with the segmented letter to find its features .

The features that have been extracted are 12 features from each character as described below. The features does not depend only on the existence or absence of the features but also it depends on the location of each feature (beginning arroba and ending arroba), so the features will be increased into 100 features. The basic feature can be classified into the following types:

Table (3.1) The basic features

1	Columnar long line
2	Columnar short line
3	Horizontal long line
4	Horizontal short line
5	Aslope short line to the left
6	Aslope short line to the right
7	Bitty arch to the right
8	Bitty arch to the left
9	Aslope long line to the right
10	Aslope long line to the left

11	Split point
12	Bitty arch down

Some definitions to explain the algorithm illustrated down :

Horizontal : refer to horizontal counter

Vertical : refer to vertical counter

Aslope right: refer to aslope right counter

Aslope left : refer to aslope left counter

The output of image pixels description table will be the starting point in feature extraction algorithm. The first terminal point will be used in the beginning and then every five pixels will be taken to explain their track and the counter will be increased according to the track of the pixels. When the feature found is stored, the counters will be equal to zero and this application will be repeated until no more pixels that are not used exists .

Algorithm (3.5)illustrate the feature extraction algorithm steps.

Algorithm (3.5) feature extraction algorithm

Input : The thinned image
The boundaries of character // *it consist four counters*
rs: refer to row start
re: refer to row end
cs: refer to column start
ce: refer to column end

Output :
CFA :character features array contain records this record include which contain feature type, feature location and features number

Steps:

Step1**:** start from terminal pixel
Step2: while (pixels counter < pixels number) do
Step3: while (pixels < 5) do
Step4: take window of 3*3 that make the terminal pixel in the center of this window. And store the pixel as began pixel .
Step5: Check the direction of its neighborhoods
Case1: it increase or decrease in a horizontal drift then
Increase horizontal
Case2: it increase or decrease in a vertical drift then
Increase vertical
Case3: it increase or decrease in aslope right drift then
Increase aslope right

Case4: it increase or decrease in aslope left drift then
increase aslope left
Step6: change field used in image description table to 1
Step7: increase pixels by 1
Step8:increase pixels counter by 1
Step9: store the end pixel used in this loop
End while in step3
Step10: after exit from while in step3 do the following

- find beginning arroba to the beginning pixel
- find end arroba to the end pixel
- store the feature type according to the counters above with its begin and end arroba , begin and end pixel
- then make the pixels equal to zero.

Step11: repeat step 3
End while in step2

Some cases there are no pixels found in the feature search and the pixels counter not equal to pixels number and there will be infinite loop so there is some solution to exit from infinite loop

First solution : find another terminal pixel but it must be not used and use it to enter to the second loop.

Second solution : find split point to add it to the features

Third solution: find an across pixel and check if it's neighborhood not used then take it to be used in the second loop.

After finishing this procedure, some features duplications in the letter appears, for example :

The letter (ا) it must be represent as one feature it was long vertical line but by using the above procedure the output will be

short vertical line

short vertical line

short vertical line

to solve this problem follow the following steps:

check if there is two features that have the same type and the same begin and end arroba and the difference between their x pointer is equal to zero or between their y pointer is equal to zero so collect the two features in one feature and convert their type from short to long.

And if there is short aslope to left and short aslope to right together with the same arroba it will be converted to bitty arch with the comely drift.

3.8 Neural network algorithm

In this work neural network with three layers where used , one input layer, one output layer, and one hidden layer.

- ***Input layer*** : there is no computations in this layer but it's only input to the next layer and it consist of 100 input which is the characteristic of Arabic character .
- ***Output layer*** :it consist of 100 node that represent four set of Arabic characters (the beginning letters(22) , middle letters(22) , isolated letters(28) , and end letters (28))
- ***Hidden layer*** : it consist of number of node it is different from network to another because there is no specified method to determine the number of hidden node . but in this project it consist of 25 node.

Algorithm (3.8) illustrate the creation step of neural network and it is return one value (0,1) if it is successful it's return (1) else return (0) .

Algorithm (3.6) create neural network

Input : Learning rate

Array of layers

Output : 0: unsuccessful

1: successful

Steps :

Step1: make network.layercount equal to number of array of layers
Step2: Check if network.layercount < 2 then return (0) or go to
Next step //*unsuccessful creation process*
Step3: make network.LearningRate equal to learning rate
Step4: loop i from 1 to number of array of layers
Step5: put the values from array of layers in network
Step6: loop from1 to array of layers[i]
Step7: check if (i≠1) then put random number in baises of network
Step8: loop k= 1 to aol[i-1]
Step9: initialize the weights of nodes (nodes of hidden layer and
output layer) by putting random number
Step9: return (1) // *successful*

The second step in neural network process is training loop. At each loop the input vectors are passed through the input layer, and the output of this layer passed through the hidden layer toward output layer, then the sum of differences between the desired output and the actual output, at the output layer nodes, is used to re-adjust the weights of the network nodes. Algorithm (3.8) implements algorithm (3.7) and in the end of each loop the stopping conditions are checked to decide whether there is need to continue training with a new loop or to stop train process and the learning rate will be adjusted according to the difference between errors_ computed in each loop . At the end of each training phase recognition of 100 characters (including the different shapes for each alphabet character) will be converted to the 28 characters using table (3.2).

Algorithm (3.7) illustrate the neural network training algorithm steps.

algorithm (3.7) training neural network

Input : character features array
Learning Rate
Output : weights of characters
Steps:
Step1: check if correct amount of input is given and correct amount of output is given
Step2: network. learnningreate =Learning rate
Step3: loop i=0 to network.layercount
Step4**:** loop j=0 to network. Layers[i].neuron count
Step5: check if (i=0) then initialize the input layer from character feature array
Step6:check if (i<>0) then set the values to zero
Step7: loop k=0 to network. Layers[i-1].neuron count
calculating the value of neurons by using two step as the following
first step :
network. Layers[i].neurons[j].values=
network. Layers[i] .neurons[j].values +

network. Layers[i-1].neurons[k].values *
network. Layers[i].neurons[j].dendrites [k].weights

second step :

network. Layers[i].neurons[j].values =
Active (network. Layers[i].neurons[j].values +
network. Layers[i].neurons[j].bais)

step8: calculate the difference between the values of neurons and desired output by using step9 and step10:

step9: calculate deltas of output layer by using this equation:

network.layers[network.layercount].neurons[i].delta =
network.layers[network.layercount].neurons[i].value *
(1 - network.layers[network.layercount].neurons[i].value) *
(target[i]-network.layers[network.layercount].neurons[i].value)

Step10: Calculate deltas of hidden layer by using the following equation:

Loop j=network. Layercount-1 to 0
Loop k= 0 to network. Layers [j].neuron count
Network. Layers [j].neurons[k].delta = network.layers[j].
neurons [k]. value * (1 - network. Layers [j] . neurons
[k].value) * network. Layers [j+1]. neurons[i] .dendreates
[k].weight* network. Layer [j+1].neurons [i].delta

Step11: Now the biases and weights of neurons will be adjust by the following steps

Loop i=network. Layer count to 0
Loop j=0 to network. Layers[i].neuroncount
network. layers[i]. neurons[j]. bais = network. layers[i].
neurons[j]. bais + (network.learnningreate * 1 * network.
layers[i].neurons[j].delta);

to adjust the weights

loop k=0 to network.layers[i].neurons[j].dendritecount
network.layers[i].neurons[j].dendrites[k].weight=
network.layers[i].neurons[j].dendrites[k].weight+
(network.learnningreate*network.Layers [i - 1]. neurons[k].
value * network.layers[i].neurons[j].delta)

algorithm (3.8) adjust learning rate

Input : character features array
Learning Rate, which is initialize to 0.8
Output: weights of characters
Steps:
Step1: error_computed=100
Step2:error_computed1=200
Step3: while (error_computed >=0.0001 AND
(error_computed<error_computed1) AND
(error_computed1 -error_computed >=0.0001))
Step4: error_computed1=error_computed
Step5: initialize error_computed to zero
Step6: call training neural network(set of data, lr)
Step7: loop i from 0 to 100
Step8: error_computed=error-computed+ network.layers[2]
.neurons[i].value – desired_output[i]
Step9: compute the average of error_computed by dividing it on 100
Step10: if (error_computed1 – error_computed > 10) then lr=lr * 0.5
Step11:if (error_computed1 – error_computed > 0.001) then lr=lr*0.02
else lr=lr
end loop

Table (3.2) represent the conversion output from number to alphabet

Entered character number	Recognized character
0 , 1	'ا'
2,3,4,5	'ب'
6,7,8,9	'ت'
10,11,12,13	'ث'
14,15,16,17	'ج'
18,19,20,21	'ح'
22,23,24,25	'خ'
26,27	'د'
28,29	'ذ'
30,31	'ر'
32,33	'ز'
34,35,36,37	'س'
38,39,40,41	'ش'
42,43,44,45	'ص'
46,47,48,49	'ض'
50,51,52,53	'ط'
54,55,56,57	'ظ'
58,59,60,61	'ع'
62,63,64,65	'غ'
66,67,68,69	'ف'
70,71,72,73	'ق'
74,75,76,77	'ك'
78,79,80,81	'ل'

82,83,84,85	'م'
86,87,88,89	'ن'
90,91,92,93	'ه'
94,95	'و'
96,97,98,99	'ي'

Chapter Four

Experimental Result

Chapter Four

Experimental Result

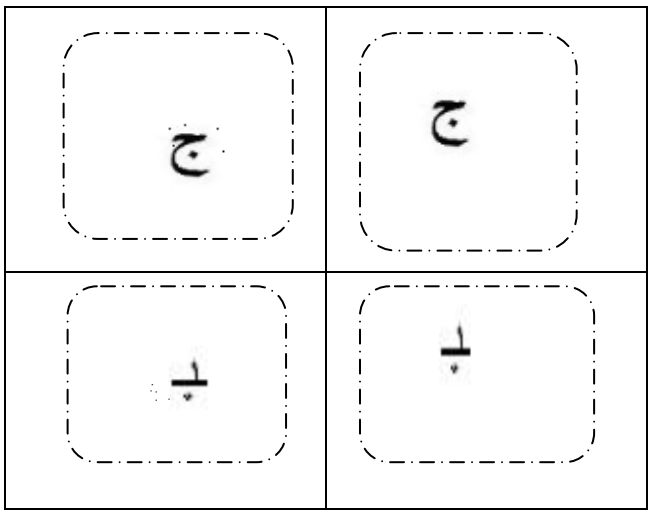
4.1 Introduction

This chapter is dedicated to demonstrate the result of the conducted test to assess the performance of neural network. The tests have been conducted to investigate the output of the three stages of the arabic character recognition, which are: preprocessing stage, feature extraction stage and recognition stage.

4.2 Preprocessing Result

The preprocessing stage contain the following operations: image smoothing, image binarization, image segmentation and image sharpening. In this paragraph the result of all these operation will be explain in details form.

- The result of image smoothing (as shown in figure 4.1)

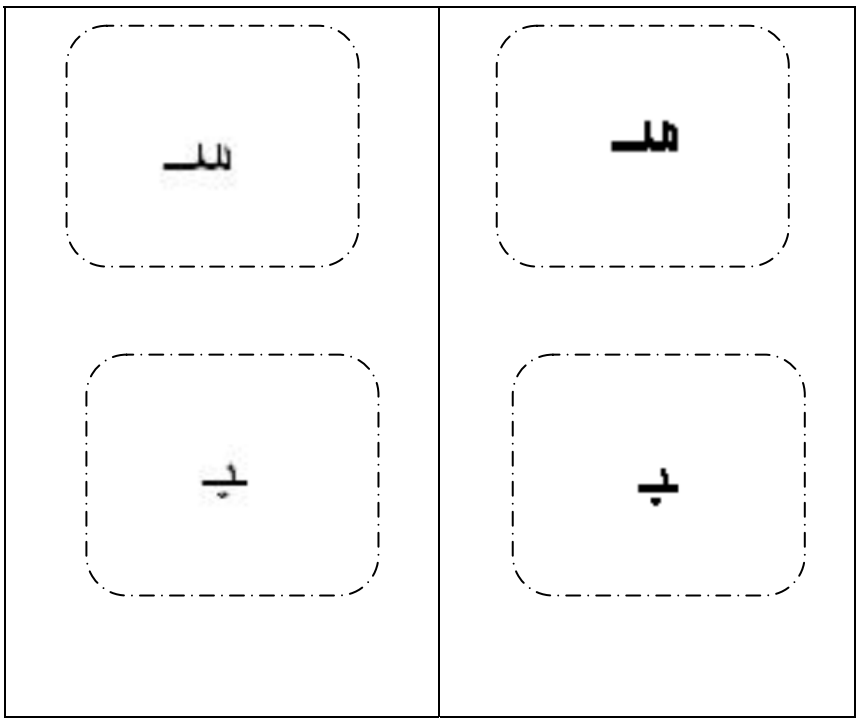


(a) (b)

Figure (4.1) column (a) original image

column (b)filtered image

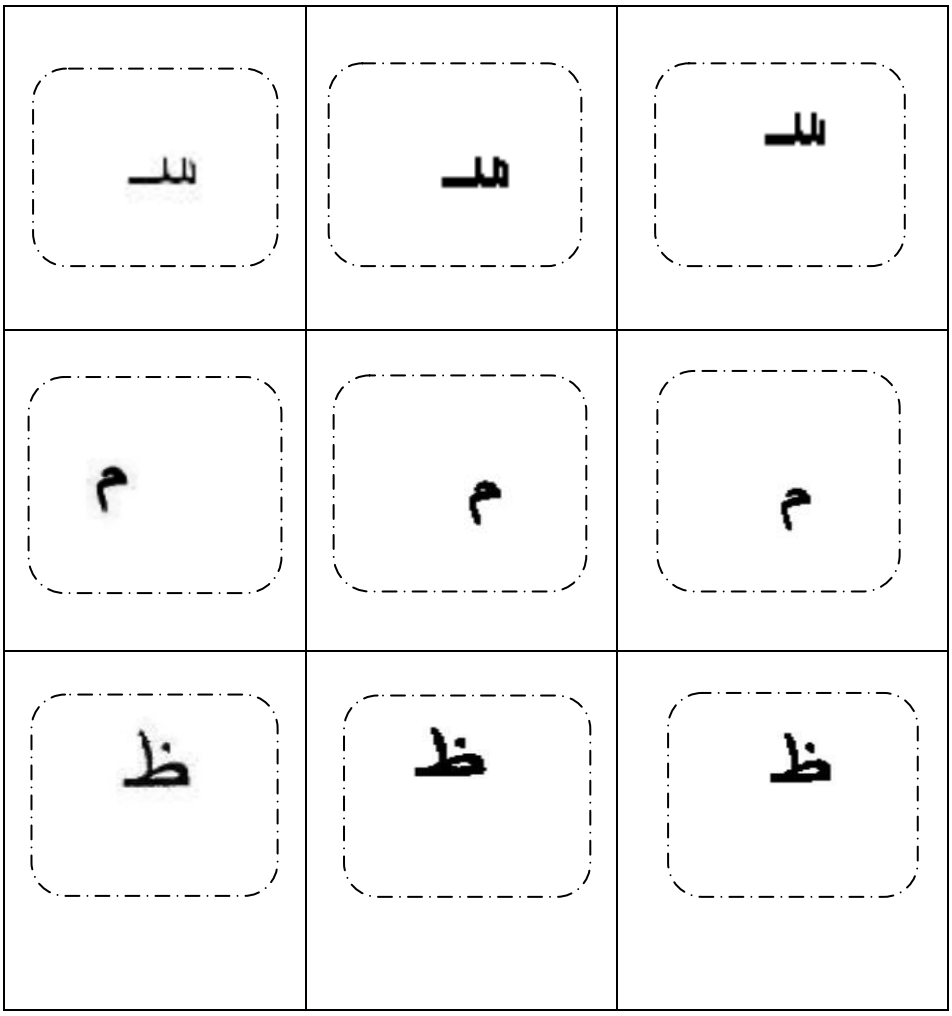
❖ The result of image binarization (by using Global and Local threshold) is shown in figure (4.2)



(a) (b)

Figure (4.2) column (a) the filtered image
column (b) the binarized image

In this process, some problems appears when the global threshold used as shown in letter (س), therefore the local threshold have been used but before thinning operation because it need the exact boundaries of the letter so the binarization process using local threshold could not be do unless the segmentation process have been done. Another problem accrue in letter (ظ) and (م) when global threshold have been used. The problems that have been appears in these letters are the cycle that found in letter (م،ظ) is enclosed or appeared more smaller than in local threshold. The figure (4.3) shows the result of images before and after local threshold.



(a) (b) (c)

Figure (4.3) column (a) represent filtered image

column (b) represent global binarization before segmentation process

column (c) represent local binarization on filtered image before thinning process .

❖ The result of segmentation process is shown in figure (4.4)

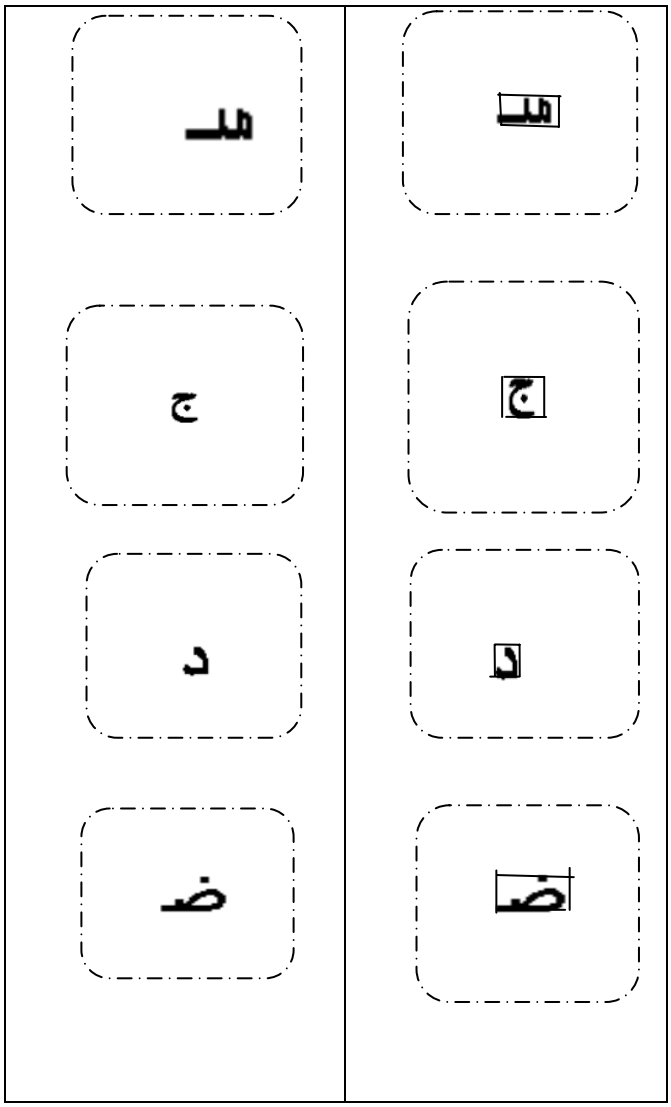
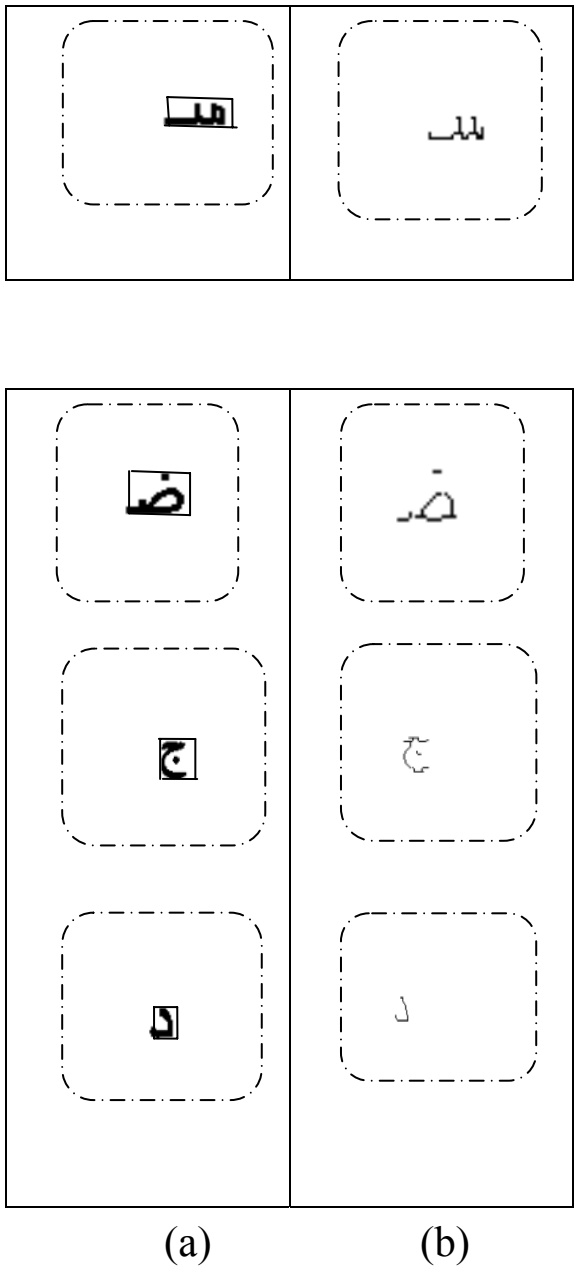


Figure (4.4) column (a) represent binarized image by implementing global threshold

column (b) represent segmented image

❖ The result of thinning process will be illustrated in figure (4.5)



(a) (b)

Figure (4.5) column (a) represent segmented image
column (b) represent thinned image

4.3 Feature Extraction Result

As shown in chapter three, this stage divided into two processes : the first building the image pixels description table for each character as

explain in appendix A, and the second perform feature extraction algorithm .

This section illustrate the output of the two processes in the feature extraction stage of some characters (د ج ـلـ ـسـ)

At the first, the segmented boundaries (the arrobas of the letter) of the letter will be shown in figure (4.6)

1 arroba	4 arroba
2 arroba	3 arroba

Figure (4.6) represent the arrobas of the letter.

- The output of feature extraction stage for (ـسـ) letter is shown in table (4.1) and table (4.2)

  Table (4.1) explain the output of image discretion table.

Pixel type	Number of pixels type
1	4
2	22
3	2
4	0

- The character (ـسـ) will be represented in figure (4.7) to illustrate the features and image discretion table.

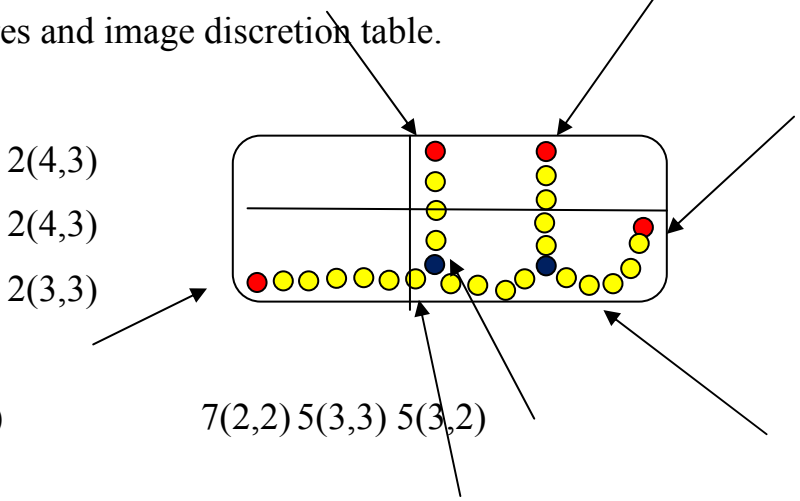


Figure (4.7) represent features and image description table for letter(ســ)

Table (4.2) explain the output of feature extraction algorithm.

Attribute	Began arroba	End arroba	Feature name
2	4	3	Columnar short line
5	3	2	Aslope short line to the left
4	2	2	Horizontal short line
7	4	3	Bitty arch to the right
2	4	3	Columnar short line
5	3	3	Aslope short line to the left
2	3	3	Columnar short line

- The output of feature extraction stage for (ـلـ) letter is shown in table (4.3) and table (4.4)

Pixel type	Number of pixels type
1	3
2	20
3	1
4	0

Table (4.3) explain the output of image description table.

- The character (ﻟ) will be represented in figure (4.8) to illustrate the features and image discretion table.

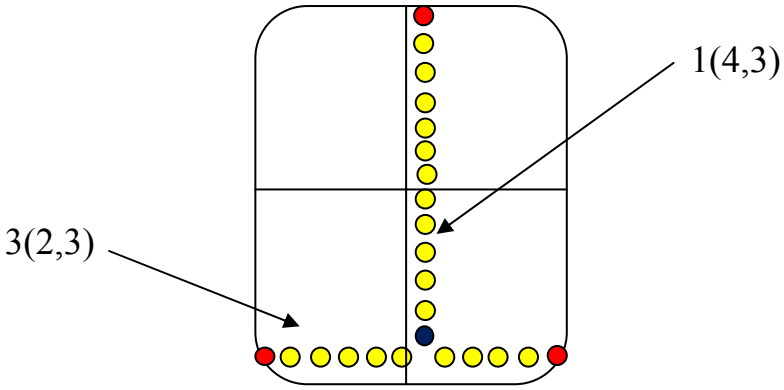


Figure (4.8) represent features and image description table for letter(ﻟ)

Table (4.4) explain the output of feature extraction algorithm.

Attribute	Began arroba	End arroba	Feature name
1	4	3	Columnar long line
3	2	3	Horizontal long line

- The output of feature extraction stage for (ج) letter is shown in table (4.5) and table (4.6)

Table (4.5) explain the output of image description table.

Pixel type	Number of pixels type
1	3
2	48
3	1
0	1

- The character (ج)will be represented in figure (4.9) to illustrate the features and image discretion table.

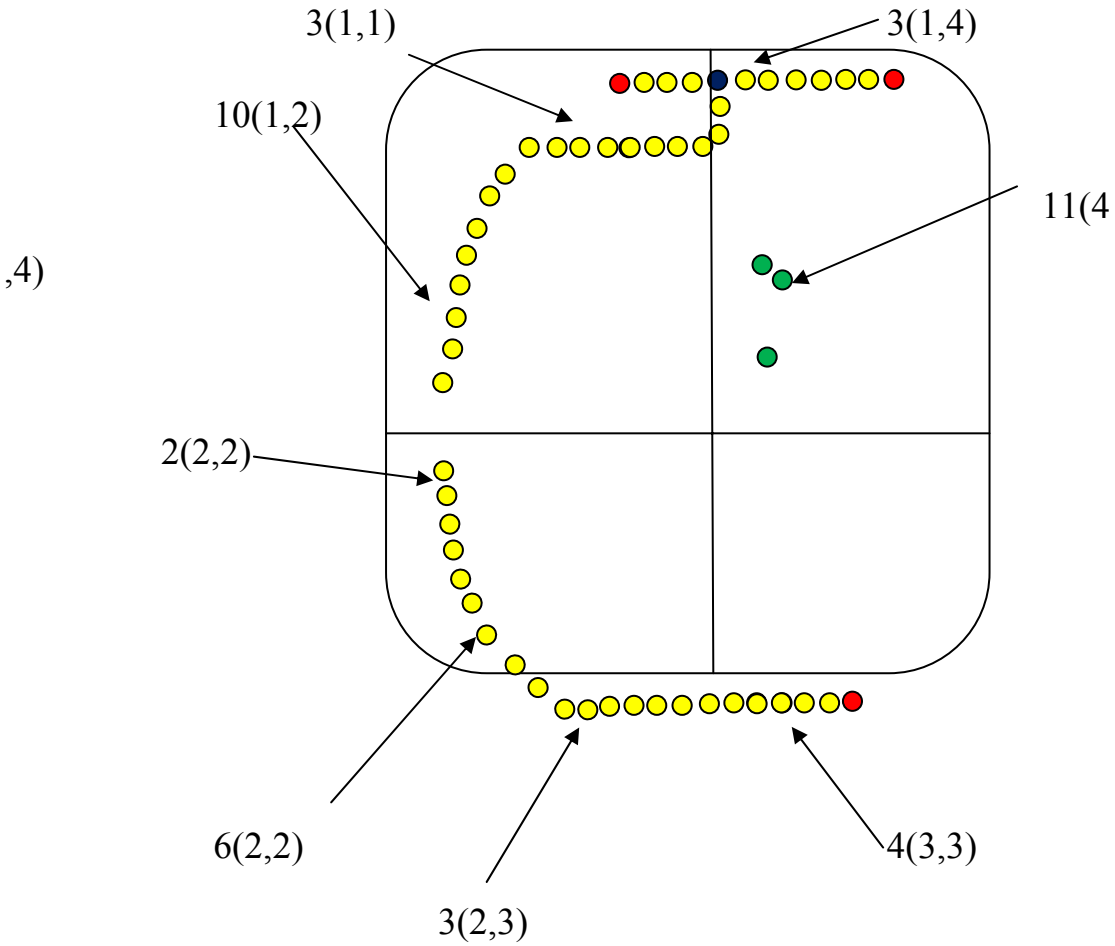


Figure (4.9) represent features and image description table of letter (ج)

Table (4.6) explain the output of feature extraction algorithm.

Attribute	Began arroba	End arroba	Feature name
3	1	4	Horizontal long line
3	1	1	Horizontal long line
10	1	2	Aslope long line to the left
2	2	2	Columnar short line

6	2	2	Aslope short line to the right
3	2	3	Horizontal long line
4	3	3	Horizontal short line
11	4	4	Split point

- The output of feature extraction stage for (ـد) letter is shown in table (4.7) and table (4.8)

Table (4.7) explain the output of image description table.

Pixel type	Number of pixels type
1	3
2	13
3	1
0	0

- The character(ـد)will be represented in figure (4.9) to illustrate the features and image discretion table.

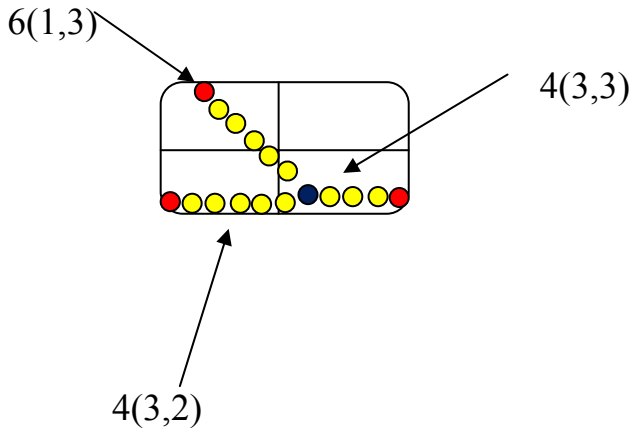


Figure (4.10) represent features and image description table of letter (ـد)

Table (4.8) explain the output of feature extraction algorithm.

Attribute	Began arroba	End arroba	Feature name
6	1	3	Aslope short line to the right
4	3	2	Horizontal short line
4	3	3	Horizontal short line

4.4 Recognition Result

In this section the output of arabic character recognition without neural network and recognition each set of character alone in table (4.9) while table (4.10) show the table of arabic character recognition system using neural network.

Table (4.9) represent recognition ratio for every set and all sets without neural network.

Type of character set	No. of character in the set	No. of recognized character before NN.	No. of recognized character after NN.
Isolated	28	27	28
Beginning	22	21	22
Ending	28	25	27
Middle	22	19	20
All sets	100	89	97

Table (4.10) explain the table the output of Recognition stage.

The input character	The recognized character
ـو	و
و	و
ـسـ	س
س	س
ـسـ	س
ـس	س
ـق	ق
ق	ق
ـقـ	ق
ـق	ق
ع	ع
ـعـ	ع
ـع	ع
ع	ع
ـب	ب
ب	ب
ـبـ	ب
ـب	ب
د	د
ـد	د
ـن	ن
ـنـ	ن
ن	ن
ـن	ن

Table (4.11) illustrates the ratio of character recognition during training Neural Network phase.

Number of input character	Learning Rate	Ratio of recognition
10	0.7	7
25	0.4	18
40	0.25	30
40	0.08	35
65	0.5	52
75	0.3	68
90	0.6	72
90	0.4	79
90	0.03	85
100	0.07	91
100	0.04	95
100	0.02	97

Chapter Five

Conclusion and Future Work

Chapter Five

Conclusion and Future Work

5.1 Conclusion

From this work, many conclusions have been drive during the analysis of the test result among these conclusions are the following:

- ♣ In this work, the hidden layer consist of 25 node. This number was taken after applying manual pruning to the hidden layer nodes to reduce computations and obtain satisfied result.
- ♣ The method is based on structural primitives such as curve, straight line , split point and etc. in a manner similar to that in which human beings describe characters geometrically.
- ♣ The features that extracted from the character classify the characters into classes so that it is easier in recognition process using neural network. When the min distance method is used to recognize the input character, the results of recognition was 89%. While using NN increased the result to 97%.

5.2 Future work

Some important proposed aspects for future investigations are given:

- ♣ Use pruning method to reduce the number of node in hidden and input layer to reduce the required time in computation process.
- ♣ Improved the system by trained it with noising data.
- ♣ Use rotate algorithm on this system.
- ♣ Use Scaling algorithm to make the system more adaptive so it can be work with any character size without losing any data in recognition process.
- ♣ Develop the features in this work to be able to recognize more samples like handwritten character.

References

References

- **[And97]** Adnan Amin and Watbiq Mansoor, **"Recognition of Printed Arabic Text using Neural Networks**", IEEE, vol. 2, No.6, PP. 612-615, 1997.
- **[Abd07]** Abdur Rahim Md. Forkan, Shuvabrata Saha, Md. Mahfuzur Rahman, Md. Abdus Sattar, "**Recognition of Conjunctive Bangla Characters by Artificial Neural Network**", International Conference on Information and Communication Technology ICICT 2007, 7-9 March 2007, Dhaka, Bangladesh**,** 2007.
- **[Ben96]** Ben Krose, Patrick van der Smagt, "**An Introduction To Neural Networks**", 1996.
- **[Fen08]** Feng Yanga , Fan Yangb, " **Character Recognition Using Parallel BP Neural Network**", IEEE Trans, 2008.
- **[Gan04]** Ganesh K. Venayagamoorthy, " **Teaching Neural Networks Concepts and Their Learning Techniques",** IEEE American Society for Engineering Education Midwest Section Conference, 2004.
- **[Gon87]** R.C. Gonzales and P. Wintz, "**Digital Image Processing** ", Addison-Wesley publishing company, 1987.
- **[Gra95]** K. Grant, "**An Introduction to Genetic Algorithm**", C/C++ Users Journal, March 1995.
- **[Gon01]** Rafael C. Gonzalez, Publisher: Tom Robbins**, "Digital Image Processing",** 2001.

- **[Hab94]** Habib Mir Mohamad Hosseini and Abdesselam Bouzerdoum, "**A System for Arabic Character Recognition** ", IEEE, No. 9, PP. 120-124, 1994.
- **[HON04]** Hongxu Ni, Sundaram Guansekaran (2004)**, "Image Processing Algorithm For Cheese Shred Evaluation** ", Journal of Food Engineering 61 (2004) 37-45.
- **[Jam91]** James A. Freeman and David M. Skapura**,** Addison-Wesley Publishing Company, "**Neural Network Algorithms**", 1991.
- **[Jiq00]** Jiqiang Song, Min Cai. Michael R. Lyu and ShijieCa, (2000) "**Graphics Recognition from binary Image: one step or two steps**" sjcai@netra.nju.edu.cn department of computer science and engineering, the Chinese university of hongkong china.
- **[Jia97]** Jianming H U and Hong Yan, (1997), "**Structural Primitive extraction and coding for handwritten numeral recognition** ", Pattern Recognition, Vol. 31 No 5 PP493-509, 1998.
- **[Ken79]** Kenneth R. Castelman (1979) "**Digital Image Processing**", Prentice-Hall, INC.
- **[Kep90]** C.Keping, 1990, "**Efficient Parallel Algorithm for The Computation of Two Dimensional Image Moments**", Pattern Recognition, Vol. 21, No.2, pp. 109-119.
- **[Kim 92]** Y. **S.** Kim, W. **S.** Choi, and **S.** W. Kim, "**High-Speed Thinning Processor For Character**

Recognition System", IEEE Transactions on Consumer Electronics, Vol. 38, No. 4, NOVEMBER 1992,

- **[Lim94]** Limin Fu, Published by Mcgraw-Hill, "**Neural Networks in Computer Intelligence**", 1994.
- **[Maj96]** Majid M. Altuwaijri and Magdy A. Bayoumi, "**Arabic Text Recognition Using Neural Networks**", IEEE, Vol. 6, PP. 415-418, 1996.
- **[Maj95]** Majid Altuwaijri and Magdy Bayoumi, "**A New Thinning Algorithm For Arabic Characters Using Self-Organizing Neural Network**", IEEE, Vol. 3, No. 5, PP.1824-1827, 1995.
- **[Mao94]** J. Mao & A. K. Jain, "**Neural Networks and Pattern Recognition**", Comp. Intell.: Imitating Life, J. M. Zurada, R. J. Marks II, C. J. Robinson (Eds.), IEEE Inc., pp. 194-212,1994.
- **[Meh96]** Mehdi Namazi, Ms. Student and Karim Faez, Associate Professor, " **Recognition of Multifont Farsi / Arabic Characters Using a Fuzzy Neural Network** ", IEEE, Vol. 2, PP. 918-922, 1996.
- **[Nei02]** Neila Mezghan, and Amar Mitiche, "**On-line recognition of handwritten Arabic characters using A Kohonen neural network**", IEEE, 2002.
- **[Nal97]** Nallasamy Mani' and Bala Srinivasan, " **Application of Artificial Neural Network Model for Optical, Character Recognition**", IEEE Trans, International conference. 1997.

- **[Naw.03]** S. N. Nawaz, M. Sarfraz, A. Zidouri, and W. G. AI-Khatib, **"An Approach to Offline Arabic Character Recognition Using Neural Networks**", IEEE, Vol. 3, No.1, PP. 1328-1331, 2003.
- **[Pra78]** W.K. Pratt, "**Digital Image Processing**", A Wiley-Interscince Publication, 1978.
- **[Rag08]** Raghad Khrebit Rashid Al-Khalidy "**Arabic Character Recognition Based on Moment Method**", M. Sc. Thesis, Al-Nahrain University, 2008.
- **[Sam99]** Sameer Singh, "**Neural Network Recognition of Hand-Printed Characters**", IEEE Trans, 1999.
- **[Set97]** R.Setiono & H. Liu, "**Neural Network Feature Selectior**", IEEE Tran. On Neural Network, vol. 8, no. 3, May, pp. 654-662, 1997.
- **[Umb97]** S.E. Umbaugh, "**Computer Vision and Image Processing**", Prentice-Hell, 1997.
- **[Yua08]** Yuanping Zhu, "**Augment Document Image Binarization by Learning**", IEEE Trans 19th international conference, PP.1-4, No.23, 2008.
- **[Zer03]** Zermi Narima, Ramdani Messaoud and Bedda Mouldi, **"Neuro-Markovian Hybrid System For Handwritten Arabic Word Recognition**", IEEE, Vol. 2, PP. 878-881, No. 1, 2003.

Appendix

Appendix A

In image pixels description table the last field (used) was converted to (1) when feature extraction algorithm finished its work , this field was the alone filed converted in this work. Because of the number of characters that used in this system this appendix will contain several tables of image pixels description table to some letters without the last field (used) because it was has constant value.

Table 1 the image pixels description table of character (ا)

Pixel type	Location in height	Location in width
1	27	32
2	28	33
2	29	33
2	30	33
2	31	33
2	32	33
2	33	33
2	34	33
2	35	33
2	36	33
2	37	33
2	38	33
2	39	33
2	40	34
2	41	34
2	42	34
2	43	35
2	44	35
2	45	35
2	46	35
2	47	35
1	48	35

Table 2 the image pixels description table of character (÷)

Pixel type	Location in height	Location in width
1	16	50
2	17	51
2	18	51
2	19	52
2	20	52
2	21	52
3	22	52
1	23	43
2	23	44
2	23	45
2	23	46
2	23	47
2	23	48
2	23	49
2	23	50
2	23	51
2	23	52
2	23	53
2	23	54
1	23	55
1	28	48
1	28	49

Table 3 the image pixels description table of character (ت)

Pixel type	Location in height	Location in width
1	21	33
2	21	34
2	21	35
2	21	36
1	21	37
1	24	28
2	25	28
2	26	29
1	26	42
2	27	29
2	27	42
2	28	29
2	28	43
2	29	30
2	29	43
2	30	31
2	30	43
2	31	32
3	31	43
2	32	33
2	32	34
2	32	35

2	32	36
2	32	37
2	32	38
2	32	39
2	32	40
2	32	41
2	32	42
2	32	43
2	32	44
2	32	45
2	32	46
2	32	47
2	32	48
2	32	49
1	32	50

Table 4 the image pixels description table of character (ـْـ)

Pixel type	Location in height	Location in width
1	10	38
1	12	44
1	17	44
1	21	49
2	22	49
2	23	50
2	24	50
2	25	50
2	26	50
2	27	50
1	28	38
2	28	39
2	28	40
2	28	41
2	28	42
2	28	43
2	28	44
2	28	45
2	28	46
2	28	47
2	28	48

Table 5 the image pixels description table of character(÷)

Pixel type	Location in height	Location in width
1	19	42
2	19	43
2	20	44
2	21	45
2	22	46
2	23	47
2	24	47
1	25	36
2	25	37
2	25	38
2	25	39
2	25	40
2	25	41
2	25	42
2	25	43
2	25	44
2	25	45
2	25	46
1	30	45

Table 5 the image pixels description table of character(ج)

Pixel type	Location in height	Location in width
1	25	39
2	25	40
2	25	41
2	25	42
2	25	43
2	25	44
2	25	45
2	25	46
2	25	47
3	25	48
2	25	49
2	25	50
2	25	51
2	25	52
2	26	44
2	26	53
1	27	44
2	28	44
2	29	44
2	30	44
2	31	44
2	32	43

2	33	42
2	34	41
2	35	41
2	36	40
2	37	40
2	38	41
2	39	41
2	40	42
2	41	43
2	42	44
2	43	45
2	44	46
2	44	54
2	44	55
2	45	47
1	45	52
2	46	53
2	46	47
2	46	51
2	46	48
2	46	49
2	46	50
2	46	51
2	46	52

الخلاصة

إن تمييز الحروف هو جزء فعال من تمييز الأنماط ، ليس فقط لأنه يفيد في تحسين الاتصال بين الإنسان والآلة لكن أيضاً لأنه يُزوّدُنا بحَلّ لمعالجة الإحجام الكبيرةِ مِنْ البياناتِ آلياً. العديد مِنْ أنظمةِ الحروف [ومثال على ذلك: - لغة لاتينية، كورية، صينية، الخ.] تم دراستها من وجهة نظر التمييز الآلي في حالتي التمييز غير المباشر (أي الصور الرقمية) والتمييز المباشر لإغراض استخلاص المعلومات.

الغرض من العمل الحالي هو لتمييز اشكال مختلفة من الحروف العربية المطبوعة باستخدام ثلاث خطوط مختلفة باستخدام الشبكات العصبية في تمييز الأحرف العربية.

إن عملية تمييز الأحرف يسبقها سلسلة من الأعمال لغرض إعداد صورة الحرف بما يمكن من استخدامها في استخلاص خواصه . وتشمل هذه العمليات : إزالة الضوضاء الموجودة في صورة الحرف وتحويل الصورة إلى الصيغة الثنائية باعتماد العتبة الموضعية. يلي ذلك عملية تنحيف الحرف باعتماد طريقة التنحيف بخطوتين.

يتم استخلاص الخواص من الأحرف التي تم معالجة صورها بالعمليات السابقة حيث تحدد خواص للحرف من بين مئة خاصية مختلفة تم تعريفها لمجموعة الأحرف وتكون هذه الخواص هي المدخلات إلى التغذية الخلفية للخلايا العصبية ذات طبقات ثلاث تستخدم في تصنيف الأحرف.

وقد فحص العمل الحالي على مجموعة من الأحرف العربية المطبوعة وبإحجام مختلفة وأعطى نتائج 97%.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

تميز الحروف العربية المطبوعة باستخدام الشبكات العصبية

رسالة مقدمة إلى كلية العلوم في جامعة النهرين كجزء من متطلبات نيل درجة الماجستير في علوم الحاسبات

من قبل

هدى مهدي عباس

(بكالوريوس جامعة النهرين 2006)

إشراف

د.بان نديم د. هيثم عبداللطيف

1432 2011