R
348
C. 1

أبجـــد ABJAD

An Off-line Arabic Handwritten
Recognition System

Hicham El Zabadani

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science
at the
Lebanese American University

LEBANESE AMERICAN UNIVERSITY 1924

Beirut, Lebanon
May 2002

Signatures Redacted

Dr. Ramzi Haraty (Advisor)
Assistant Professor of Computer Science
Lebanese American University

Signatures Redacted

Dr. Nash'at Mansour
Associate Professor of Computer Science
Lebanese American University

Signatures Redacted

Dr. May Abboud
Associate Professor of Mathematics
Lebanese American University

To my parents, Mahmoud and Hajar...
To my fiancée, Layal...

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to Professor Ramzi Haraty, for supervising me and for providing the original inspiration for this work. I am also indebted to Mr. Alaa Hamid who has helped me admirably with enthusiastic guidance and support.

Special thanks go to Dr. Nash'at Mansour and Dr. May Abboud, committee members of my MS thesis defense, to Yousef Kaddoura for his friendship and support, and to my colleagues at Gezairi Transport for their help.

ABJAD

Abstract

In this work we present a system for the recognition of handwritten Arabic text using neural networks. This work builds upon previous work done by [Hamid 2001]. That part dealt with the vertical segmentation of the written text. However, faced with some problems like overlapping characters that share the same vertical space, we tried to fix that problem by performing horizontal segmentation. In this research we will use two basic neural networks to perform the task; the first one to identify blocks that need to be horizontally segmented, and the second one to perform the horizontal segmentation. Both networks use a set of features that are extracted using a heuristic program. The system was tested with over 1500 characters (each character has on average about 50 rows) and the rate of recognition obtained was over 90%. This strongly supports the usefulness of proposed measures for handwritten Arabic text.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Whether we like it or not, the world is undergoing an information technology revolution. People are forced into contact with computers and our dependence upon them continues to increase. That is why computers should be easier to use. Nowadays, as most of the world's information processing is done electronically, it becomes more efficient if we make the transfer of information between people and machines more simple and reliable.

1.1 Background

As computer power has increased over the years, and their applicability has also increased, one of the most important researches made is to make computers easier to communicate with [Hamid 2001]. One of the major obstacles to perform this task is the fact that most of our business data is still being processed on papers.

Many researchers are still trying to replicate human functions, and this has been the subject of many researches in the artificial intelligence field. Optical Character Recognition is one area that is being addressed and this is due to many reasons; there are too many applications that use recognition such as postal-code recognition, check recognition and document reading. The process of recognition can be divided into two categories: the recognition of machine printed data and the recognition of hand printed data. Concerning machine printed data, the problem is not so hard to be solved since they are uniform in size and position. However, hand printed data are non uniform; they can be written in many sizes and styles. Therefore, reading machine printed characters is much easier than reading hand printed characters. In fact, there are several products nowadays that are in the market and they are producing good results. In spite of many years of research, optical readers that are able to read hand printed characters are rare and even if they exist, they can only recognize digits that are well printed.

1.2 History of Optical Character Recognition

The history of optical character recognition (OCR) is old in the field of pattern recognition. At the beginning, characters were easy to deal with and researchers expected that the problem would be easily solved. In 1929, the first attempt to perform optical character recognition was by a German, Tauscheck [Anshu 1999]. Tauscheck used the principle of template matching and employed the optical and mechanical technologies available at that time. Light passes through mechanical masks and then captured by a photodetector. When an exact match occurs, no light will pass through the mask and reach the photodetector. This kind of technologies is still being used today, almost 70 years later.

The history of OCR is divided into three main categories: template matching, structural analysis and neural networks.

1.2.1 Template Matching

The basic idea in template matching is to superimpose a test template and a known template, and then to take the total sum of the differences between each sampled value and the corresponding template value; if the difference is within acceptable predetermined amount, then the templates match. In 1962, RCA made a very sophisticated OCR using electron tube technology [Anshu 1999]. The system used 91 channels, and the development team concluded that the system could recognize all of the English and Russian characters. However, no commercial product was produced and development in this field did not progress at RCA.

Template matching seems like a reliable method of character recognition, but it has some weaknesses. Firstly, it is time consuming, since it involves comparing all possible templates. Secondly, shifting and rotation of the characters would produce unreliable results.

1.2.2 Structural Analysis

Structural analysis is a technique used mainly for hand printed characters. As we mentioned above, template matching is reliable only for machine printed characters, that is why researchers started their researches on structural analysis. Unlike template matching, there are no mathematical principles behind structural analysis. Instead, heuristics are used to perform the recognition. Most of these heuristics are based on the idea that any

structure could be broken into parts; these parts can be described by features and relationships between the different parts.

In 1961, Weeks used the idea of the slit/stroke for his recognition module [Nagy 1982]. Johnson and Dimond used the same idea except they used sonde instead of a slit [Mori et al. 1992]. Basically, all structural analysis methods used the overall structure of the character's bitmap to determine features. There are many different. methods such as contour analysis. However, it has been shown that it is quite difficult to develop a guaranteed algorithm for recognition, that's why heuristics were used instead. While these heuristics are somewhat reliable, structural feature extraction has proven to be difficult in handwritten character recognition because of the variations in handwritten data.

1.2.3 Neural Networks

Most of the methods discussed above were studied at an early stage in the history of OCR. In 1986, the development of the back-propagation algorithm for multi-layer feed-forward neural networks was reported by Rumelhart, Hilton and Williams [Haykin 1994]. The advantage of using a neural network for pattern classification is that it can construct nonlinear decision boundaries between the different classes in a non-parametric fashion, and thereby offer a practical method for solving highly complex pattern classification problems. One of the early models was designed at AT&T to recognize postal zip-codes. They used a neocognition model in conjunction with traditional back propagation. They achieved a recognition rate of 90% with 1% false positive [Fukushima et al. 1983].

Lee and Kim experimented with different neural net structures for recognition; they proposed a recurrent neural network based on the standard three layer feed forward multi-layer perceptron (MLP). With a training set of 4000 digits and a testing set of 2000 digits, they achieved a 97.3% correct recognition rate and no rejection class was taken [Lee and Kim 1995].

In general, neural network techniques have been successfully used for handwritten data recognition. They perform better then traditional image processing techniques alone.

1.3 Scope of the work

The recognition of Arabic characters represents a significant challenge due to the large set of features in the Arabic script. Each of the 28 letters has on average four shapes depending on its position within a word (start, middle, end or isolated; see Table 1). Also, because Arabic words are written cursively from right to left and contain several connected letters, character segmentation is necessary before starting the recognition phase. Some words contain broken parts because some characters cannot be connected to the others. Vowel diacritics and writing style add another degree of complexity to the recognition task.

Table 1 Different positions of a character.

Isolated	Start	Middle	End
م	مـ	ـمـ	ـم

In this thesis, we are working on the second part of a system that is divided into three parts. The first part deals with the vertical segmentation of Arabic handwritten text. It was done and tested by Hamid [Hamid 2001]. The second part which is the main topic of this thesis, deals with the horizontal segmentation of the words that could not be segmented vertically. The third part is still being worked on, deals with the classification of all characters produced by the first two parts.

ABJAD uses two multi-layer perceptron neural networks to perform the horizontal segmentation. The first one determines which block should be horizontally segmented, and the second one determines which rows are valid segmentation points; and thereby, will finish the horizontal segmentation.

The input of the second part of ABJAD is the output of the first part (see Figure 1a), which is formed of both single characters and blocks of overlapping characters. The second part will identify each block as separate character or a block that needs to be segmented horizontally (see Figure 1b).

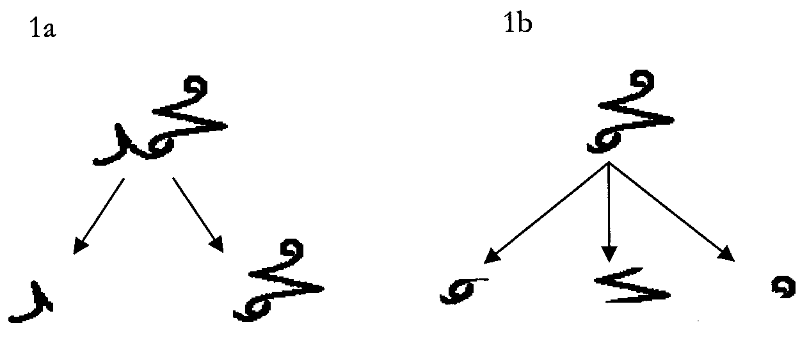


Figure 1: (1a) Input of the first part and its output. (1b) Input of the second part and its output.

1.4 Organization of the thesis.

The remainder of this thesis is divided into 5 chapters. Chapter 2 describes a historical review of the optical character recognition field. It also presents a comparison of the techniques found in the literature for the segmentation and recognition of handwritten text.

Chapter 3 describes in detail the proposed approach used to identify ligature blocks. The design of the heuristic and neural network components is also presented.

Chapter 4 describes in detail the proposed approach used to segment horizontally Arabic handwritten text.

Chapter 5 presents the experimental results of this approach and finally a conclusion is drawn in Chapter 6.

Chapter 2

LITERATURE REVIEW

Before describing a new handwriting recognition system in the later chapters, it is worth presenting here the field of automatic handwriting recognition in its entirety. After describing a classification of the field, applications dealing with handwriting recognition systems are discussed and work by other authors is presented to demonstrate the approaches taken.

2.1 Classification of handwriting recognition problems

Having discussed the need for automatic handwriting recognition in general, it is useful to examine the field more closely and to identify several areas with different applications. The literature is divided into groups of researchers, each one concentrating on a special area of handwriting recognition.

2.1.1 On-line versus off-line

Handwriting recognition systems are generally polarized between those receiving their data directly from some sort of pen device attached to the computer (on-line), and those which recognize handwriting already present on a piece of paper (off-line). In the literature *dynamic* is sometimes used to mean on-line and *static* off-line.

So far, the majority of systems have tackled the easier problem, on-line, where the time ordering of strokes is available as well as pen information (up/down). Overlapping strokes can easily be distinguished and stroke positions are accurately known. On the other hand, off-line systems have to cope with different pen types, wide strokes which overlap and a lack of ordering information.

Since the on-line data are a one-dimensional stream of information, techniques from speech recognition have been successfully applied to this problem, including Hidden Markov Models [Bellegarda et al. 1994] and time-delay neural networks [Shenckel et al. 1994]. The data from the tablet are usually (x,y) coordinates sampled at a constant frequency in time and presented in terms of arc-length, curvature, and angle, with information about whether the pen is touching the tablet. A particular problem of on-line recognition is how to handle delayed strokes – strokes which are written after the rest of the word, as in doting 'تـ's and crossing 'ط's. Some authors choose to manage without this extra data; [Schenkel et al.] record its existence as a 'hat' feature associated with the strokes over which the delayed strokes occur, and [Bengio et al. 1994] represent the surrounding visual context of all strokes so that the dot is seen above the cusp of the 'تـ'. Although applications and techniques vary considerably, the general classification of both off-line and on-line handwriting analysis is similar (see Figure 2)

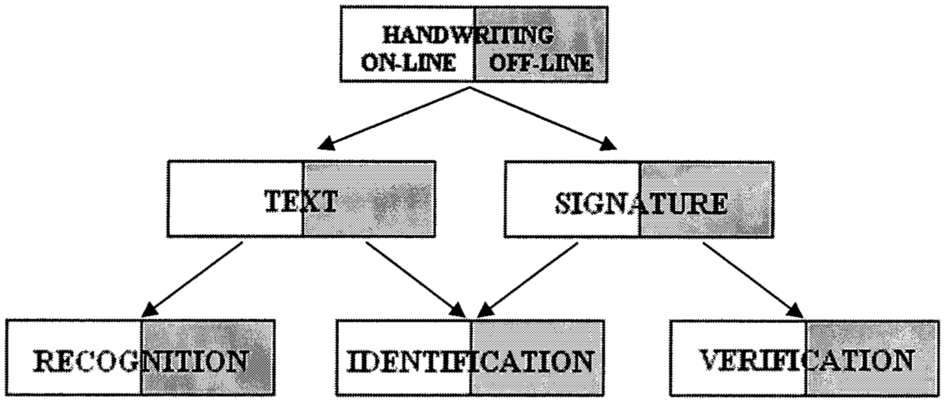


Figure 2: Subdivision of machine handwriting recognition.

2.1.2 Writer independence

Handwriting styles are extremely diverse, depending both on the pattern used to teach handwriting to an individual and on the individual's idioscript. Because of this, it is more difficult to devise a system to recognize many peoples' handwriting than one which need only to recognize the writing of a single author. Instead of creating a system which can recognize anybody's handwriting, the problem of multiple writers could be tackled by a system which is able to adapt to the current writer. Adaptation to the writer's style could be used when recognizing a lot of material by the same author, but it would not be useful when identifying the city names on envelopes for example.

2.1.3 Vocabulary size

The task of recognizing words from a small lexicon is much easier than from a large lexicon (where words are similar to each other). Thus, an important criterion in assessing system performance is the size of the lexicon used. The lexicon will depend on the application of the recognition system. For a general text transcription system, a lexicon of 60,000 words, would cover about 98% of concurrences, and for specific domains, such as reading cheque values in words, or postal towns from envelopes, the vocabulary can be much smaller [Waibel and Lee 1990]. Alternatively, it may be necessary for the system to recognize non-words if the user is likely to write words that does not exist in the lexicon, such as abbreviations, foreign words or names.

2.1.4 Isolated and overlapping characters

In cursive script, it is hard to distinguish the boundaries between letters – the difference between 'نتـ' and 'تنـ' is very slight. Moreover, tackling overlapping characters (which exists in the Arabic script) is a real challenge – the case of 'محـ' or 'لم'. The task can be simplified by forcing the writer to write each character on separate vertical space, so that no overlapping characters will be produced. A number of authors have investigated the problem of recognizing isolated characters (section 2.3.1), particularly the problem of reading postal codes.

Other author has described methods of segmenting pages into words and distinguishing between gaps in words and gaps between words [Srihari et al. 1993].

2.1.5 Optical character recognition

Off-line handwriting recognition has much in common with optical character recognition (OCR) – the reading of print by the computer. This application has received much attention during the 1980s and successful solutions have been found, with commercial packages available for microcomputers which can read type in a variety of fonts and in a certain amount of noise. The history and current status of OCR are reviewed by [Mori et al. 1992] and [Palvidis 1993]. In more difficult situations, these commercial packages are still not in a satisfactory level. Authors describe problems working with unusual character sets and fonts, poor quality documents in special formats [Bos and van der Moer 1993].

The reason why the success of OCR has not carried over into handwriting recognition is the variability in handwriting. For type in a fixed font, all letters 'و' are produced from a single archetype, and thus are very similar on the page, except when it is corrupted by a relatively small amount of noise. The process of handwriting is much more variable in all of these processes and suffers from variations due to other effects such as co-articulation – the influence of one letter on the other. Also, with type, the symbols are usually distinct (except certain ligatures, as 'لا', which can be learnt as separate symbol) so the problem of segmentation is not present.

As a consequence of this the relatively simple techniques used in OCR, such as template matching, are inadequate when presented with the greater variability in handwriting so relatively little research in the OCR literature carries over to handwriting recognition.

2.2 Applications

This section reviews some of the more important applications that may be developed for off-line handwriting recognition.

2.2.1 Cheques

One important commercial application for off-line cursive script is in the machine reading of bank cheques. While the amount in figures is easier to read, it should be checked that the amount in words is the same, and this can be used for confirmation where the numerical amount is unclear. Such a system would only need to have a small vocabulary. Given a system that achieved high accuracy without a lexicon, one could check that the payee corresponded to the account to be credited. Such a system might also include signature verification, bringing about an increase in security. Given the number of cheques passing through the banking system each day, a cheque reading system, even if only able to confidently verify half of the cheques, would save much time on an unpleasant job. Cheques which could not be confidently verified by machine would still be processed manually, so accuracy would still be maintained. The project supported by the French post office has the goal of achieving a 1 in 100,000 error rate from the combined recognition of literal and numerical amounts, but permitting 50% of cheques to be rejected for manual sorting [Leroux et al. 1991].

2.2.2 From postcodes to address

Off-line systems capable of recognizing isolated handwritten digits have already been created and installed in many post offices around the world, as part of automatic mail-sorting machines. Given a system to locate the postcode on an envelope [Wang and Srihari 1988; Martins and Allinson 1991; Palumbo et al. 1992] this can be read and used to direct mail automatically. Clearly certain countries such as the USA are at an advantage in having digit-only zip-codes and many researchers have already tackled this problem with reasonable success (section 2.3.1).

To process more mail automatically, systems must begin to use the information contained in the rest of the address. This allows the uncertainty in the postal code classification to be removed by comparing candidate zip codes with candidate addresses in a database of all address / zip code combinations, giving more high confidence classifications. Furthermore, for countries with limited resolution in the postcode, the address can be used to increase the resolution of sorting.

2.2.3 Form processing

Another major application which is now receiving attention is the automatic processing of forms. Forms are widely used to collect data from the general public. Most of the information must be stored in databases and can be processed automatically once entered into the computer. Data entry is currently the bottle-neck in the process. Several authors have written programs to segment the handwritten data from the pre-printed form and then to transcribe the handwritten data. In some applications, this may be isolated capital letters written in boxes, but work is moving on to hand print [Breuel 1994; Garris et al. 1994]. Although forms must usually be hand printed to keep the writing as legible as possible, for human as well as machine processing, cursive recognition would still be useful for processing those forms that have been mistakenly filled out in cursive script.

2.2.4 Other applications

A variety of other office document processing systems using off-line handwriting recognition can easily be considered. Already many companies use electronic document processing systems which manipulate the scanned images of documents rather than the documents themselves. One way to reduce the data storage is to extract the information and store text in ASCII. Documents would then be easily searchable and index construction would be possible to make. Further possibilities exist in reading handwritten documents for the blind or in automatic reading of faxes.

Of course, the advantages of handwriting recognition are not restricted to English or to the Roman alphabet, though these have probably attracted most research. In the literature there is a wide range of papers describing handwriting recognition in a more than one language. The basic problems of handwriting recognition are common to all languages, but because of the diversity of scripts, some languages may be harder to recognize. For example, Japanese Kanji [Mori and Yokosawa 1988] and Chinese [Lu et al. 1991] characters are stroke-based, and characters are easy to segment from one another, but characters are very complex and there are many classes to distinguish. Arabic and roman alphabets can be recursive, and Arabic requires accurate recognition of diacritic marks.

2.3 Existing off-line handwriting recognition systems

This section reviews some of the off-line handwriting systems. To do this, it is convenient to classify them into isolated character and cursive script systems.

2.3.1 Isolated characters or digits

[Suen et al 1980] provide a good review of handwriting recognition upt to year 1980, concentrating on isolated character recognition – which had been the focus of research until then. They describe a variety of feature based approaches and divide these into global features (templates or transformations such as Fourier, Walsh or Hadamard); point distributions (zoning, moments, n-tuples, characteristic loci and crossings and distances) and geometrical or topological features. The later were and remained the most popular techniques and involve separate detectors for each of several types of features such as loops, curves, straight sections, endpoints, angles and intersections. For instance, some use cross-points, end-points and bend-points as their features [Impedovo et al. 1994]. Other use features like end-points, junction, curve and loops, each of which is associated with a numerical quantity, such as curvature or length, before being decoded in a neural network (a feed-forward neural network or an adaptive feedback classifier) [Elliman and Banks 1991].

[Nellis and Stonham 1991] use sets of global morphological features created by separately examining the left, right, top and bottom edges of each character. The profile of the character from each edge is coded as a separate feature for classification by a neural network. Other authors have tackled the problem of recognizing isolated digits or characters in the last few years [Hepp 1991; Idan and Chevalier 1991], particularly since the increasing availability of data has made this a standard test problem for testing pattern recognition methods. Isolated digit classifiers have now become so good that researchers are concentrating on reading whole zip codes where digits are often touching, and finding optimal combinations of multiple classifiers now seems to be a more promising way to reduce the error rate than finding better classifiers.

2.3.2 Off-line cursive script

The problem of off-line cursive script has received a little attention in the past years partly because of the difficulty of the problem, but also because of the lack of the data. [Simon 1992] makes the distinction between the segmentation approach and the global approach, according to whether words are identified by recognizing individual letters or by recognizing words as a whole.

All the authors described below incorporate some form of preprocessing to normalize and clean the data. In each case, a recognition strategy then hypothesizes character or word identities, and because exact recognition is very difficult, all the approaches use a lexicon to constrain the responses to a known vocabulary.

[Kimura et al 1993] created a system for reading city or state names in addresses. These authors take a dual approach, with a first, quick classification to reduce the lexicon size, followed by a more accurate second classification using different techniques. The first stage finds a rough explicit segmentation and each segment is classified as a letter. The second stage finds a different explicit segmentation by splitting the word into disjoint boxes and joining the boxes together using dynamic programming to form complete characters. These are then passed to a character classifier. These authors report results of 91.5% recognition with a lexicon of 1000 words on the CEDAR database of words segmented from addresses in the U.S. mail [Hull 1993].

[Cheriet and Suen's 1993] approach is also letter-based. However, their approach is to extract a number of key letters from each cursive word – partly the initial letter and those clearly identifiable by ascenders, descenders or loops. For a small vocabulary task (reading cheques0 as described in their paper, identifying these key letters might be sufficient to identify most words, but the authors propose their techniques as a way of filtering, to reduce the number of words in the lexicon of possible matches.

Papers by [Srihari and Bozinovic 1987] take an explicit segmentation approach, but here each segment need not to correspond to a character. They find presegmentation points which include all the boundaries between characters, they also split some characters into two or more pieces. They then find features (16 in all, including dots, curves, strokes, loops and crusps) within the segments by a series of event detectors and use the features to construct letter hypotheses according to statistics of feature occurrences gathered during training. Words are hypothesized via a stack method, where the most likely prefixes are stored and expanded until the word end is reached. After the first iteration of this procedure, the stack contains all the hypothesis for the first letter in order of likelihood. The top hypothesis is then expanded by looking at what letters could follow. The resultant two letter sequences are put onto the stack, to be expanded when they are the most likely sequences. At the end of the word, the lexically correct word that is the highest on the stack is chosen as the best match. They conducted a number of experiments, using different writers and different lexica (780 and 7800 words). Testing on a single-author database of horizontal, non-slanting writing, a 77% recognition rate was obtained on the small lexicon, 48% on the large. A second single-author database yielded a 71% recognition rate on the smaller lexicon.

[Yanikoglu and Sandon 1993] take a similar approach. They find possible character segmentation points and attempt to classify segments or groups of up to three segments with a neural network classifier trained on isolated letters. Incorrect segmentations tend to get lower classification scores than when a letter is correctly segmented, and when the scores are combined in a hidden Markov model, the best hypothesis for the groupings of segments and their identities is found. Results of 70% for single-author cursive word recognition are quoted for a lexicon of 30,000 words.

[Edelman et al.1990] have developed a handwriting reader which relies on the alignment of letter prototypes. Here, anchor points (e.g. endpoints, turning points at the top, bottom, left or right of a character) are found in the test word and these points are used to match the word against a set of prototype curves, coded as splines, which can be composed into lower-case characters. The system is hand-designed and is not trained automatically. Using 30,000 word lexicon, these authors obtained an 81% recognition rate on the training set and around 50% on the test sets.

The problem of reading the amount on cheques has been tackled by a number of authors in the problem posed by the French post office. The task here is to recognize amounts written (in words) on postal cheques and to use these to verify the amounts written in figures. [Moreau et al.1991] identify a few characteristics of the cursive words and match these to a set of reference words with dynamic programming. The identified words are used together with a grammar to verify the amount in figures. With a 60% rejection rate, the error rate achieved is 0.2%.

Chapter 3

LIGATURE IDENTIFICATION

As we discussed in the previous chapters, [Hamid 2001] has completed the first part of the system, which consists of segmenting words vertically. His system was unable to deal with ligatures (blocks of characters that are situated one above the other) since there is no vertical line that can separate those characters (Figure 3). Here comes our first aim in this system which is to identify all kind of ligatures that can be found in the lexicon.

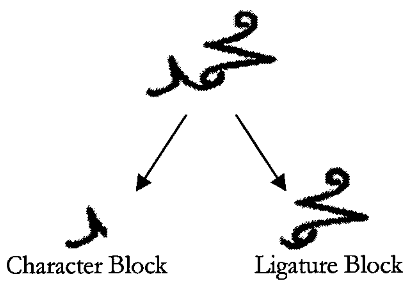


Figure 3 Horizontal Segmentation.

3.1 Ligature Identification Obstacles

Before attempting to segment horizontally, the system should be able to recognize every single block as either ligature or character block (Figure 3). However, there are several major problems related to ligature recognition:

- **Variety in Size:** The same character may be written in different sizes without changing the meaning of the character (Figure 4).

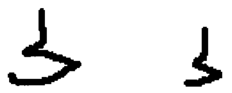


Figure 4 Same ligatures written in different sizes.

- **Variety in Shape:** Characters may vary in shape, i.e., in line thickness, color, or stroke direction. This would cause some problems like touching characters, filled holes, or broken characters (Figure 5).

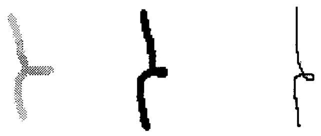


Figure 5 Same ligatures written in different shapes.

- **Variety in Style:** Every writer has his own writing style, i.e., different writers or the same writer in different conditions could write the same character in a different style (Figure 6).

Figure 6 Same ligatures written in different styles.

- **Similarity in Block Shapes:** Sometimes two different blocks, i.e., ligature and character block, could have the same shape as in figure 7. This may lead to some errors in the recognition process.

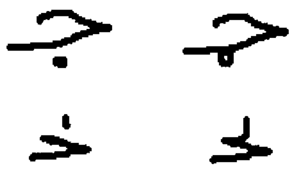


Figure 7 Two blocks having similar shapes.

3.2 Proposed Technique

Before attempting to start ligature recognition, there are some steps that need to be done like data collection, data analysis, scanning, binarization, and finally block classification.

3.2.1 Data Collection

Samples were randomly collected from various students at the Lebanese American University and Gezairi Transport Company. People were asked to write character and ligature blocks. These samples were then scanned at 100 pixels per inch, and saved in monochrome Windows Bitmap format.

3.2.2 Feature Extraction

In areas of pattern recognition, features can be characterized as a way to distinguish one class of objects from another in a more concise and meaningful manner than is offered by the raw representations. Therefore, it is so important to define meaningful features when we plan to develop a good recognizer, although it has been known that a general solution has not been found yet. In many cases, features are generally defined by hand based on the experience and intuition of the designer.

Depending on the problems given, there are a number and variety of features that can be defined in terms of extracting methods and ways of representation. In many practical applications, it is not unusual to encounter problems involving hundreds of features. The designer usually believes that every feature is meaningful for at least some of the discriminations. However, it has been observed in practice that, beyond certain point, the inclusion of additional features leads to worse rather than better performance. Furthermore, including more features means simply increasing processing time.

If we see that some features do not help class discrimination, we can get the idea that somehow some of these features are redundant. So the feature selection is defined as one of redundancy reduction by retaining useful features and getting rid of the useless ones.

Ligature classification, which is a typical example of pattern recognition, contains the same problem. Due to diversity in writing styles, size and tools used, variations in shape can be easily found, even in ones written by a single person.

3.2.2.1 Extraction Method

This stage transforms the block to be recognized into a sequence of features that will be used as the input of the neural network in a later stage. However, let us first understand some of the major basic features.

- **Loops:** Loops can be found from the skeleton or by performing a connected-component analysis on the original image, to find areas of background color not connected to the region surrounding the word (Figure 8). A loop is coded by a number representing its area. Authors like [Srihari and Bozinovic 1987] use the topology of a word as a feature. However this is not always a good choice of invariant since extra loops can easily be formed, or loops that could be expected might not be fully closed.

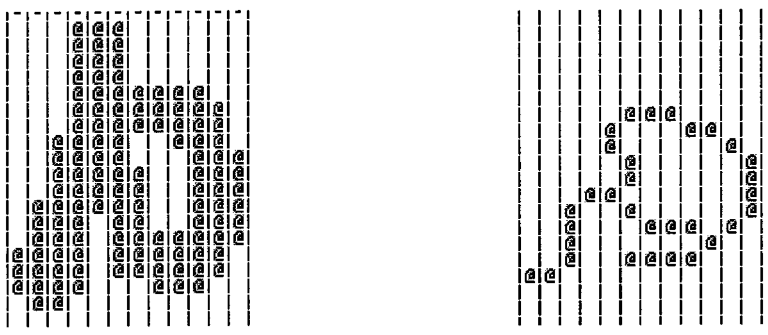


Figure 8 Example of a hole in a normal image and a skeletonized one.

- **Junctions:** Junctions are easily found in the skeleton of the block, as points with more than two neighbours. Junctions indicate points where two strokes meet or cross.
- **End Points:** End points are points in the skeleton with only one neighbour and mark the ends of strokes, though they can be produced by artifacts of the skeletonization algorithm (Figure 9).

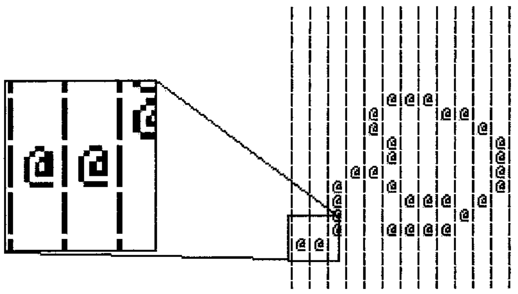


Figure 9 An end point colored with red.

- **Turning Points:** Points when the direction of a skeleton segment changes from upward to downward are recorded as top turning points. Similarly left, right and bottom turning points can be found (Figure 10).

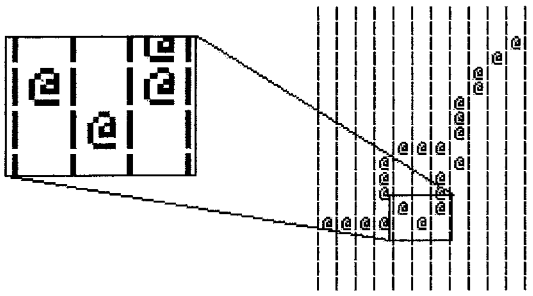


Figure 10 A bottom turning point shown in red.

- **Center of Gravity:** The center of gravity used in ligature identification helped a lot reduce the error rate. Using a simple algorithm, the center of gravity was obtained and then it was used relatively to the width and height of the block.

In order to decide whether a block is a ligature or character one, the list of features listed in Table 2 were used.

Table 2 Major features extracted for each block.

Feature	Attribute	Description	Value
Width and Height		Block width and height in pixel.	(0,∞)
Center of Gravity		X and Y coordinate of the center of gravity.	(0,∞)
Black Pixel Density	Total Col. Density Minima	How many columns contain density minima?	(0,width)
	Total Row Density Minima	How many rows contain density minima?	(0,height)
	Total Col Density Maxima	How many columns contain density maxima?	(0,width)
	Total Row Density Maxima	How many rows contain density maxima?	(0,height)
Transitions	Row Max. Transitions	What is the maximum number of transitions that can be found in a column?	(0,Height)
	Column Max. Transitions	What is the maximum number of transitions that can be found in a column?	(0,width)
Junctions	Total Junctions	Sum of the Junctions	(0,∞)
Loops	Max. Loop Density	What is the maximum number of pixels that separate the borders of a loop?	(0,∞)
	Max. Loop Transition	What is the maximum number of transitions between loops?	(0,∞)
End Points	Total End Points	Sum of the end points.	(0,∞)
Turning Points	Total Turning Points	Sum of the turning points.	(0,∞)
Contours	Total Upper Contour	Sum of the indices of the pixels that form the upper contour.	(0,∞)
	Total Lower Contour	Sum of the indices of the pixels that form the lower contour.	(0,∞)

3.2.2.2 Feature File Preparation

Now that we have created a feature file that will be used as the input of the neural network, we should first perform a manual separation of all points as valid and invalid points. We then save them in one file that will contain the extracted set of features and the desired output for each point.

3.2.3 ANN Architecture

It has been recognized that multilayer feedforward networks are capable of forming arbitrarily complex decision boundaries and can represent any Boolean function. The development of the *back-propagation* learning algorithm for determining weights in a multi-layer feedforward network has made these networks the most popular of all the networks.

Figure 11 shows a typical 3-layer perceptron. In general, a standard L-layer feedforward network consists of one input stage, L - 1 hidden layers, and one output layer of units which are successively connected in a feedforward fashion with no connections between units in the same layer and no feedback connections between layers. We denote $w_{ij}^{(l)}$ as the weight on connection between the i^{th} unit in layer (l-1) to j^{th} unit in layer l.

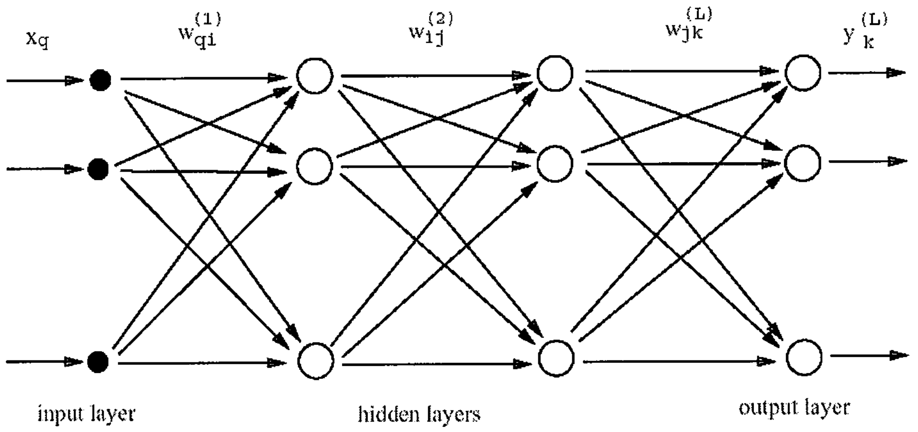


Figure 11 A typical 3-layer feedforward network architecture.

The task of a learning algorithm is to automatically determine the weights in the network such that a certain cost function is minimized.

Let $\{(x^{(1)}, d^{(1)}),(x^{(2)},d^{(2)}),\ldots,(x^{(p)},d^{(p)})\}$ be a set of *p* training patterns (input-output pairs), where $x^{(i)} \in R^n$ is the input vector in the n-dimensional pattern space, and $d^{(i)} \in [0,1]^m$ is the desired output vector in the m-dimensional hyper-cube. For classification purposes, m is set to the number of classes. The squared-error cost function, which is most frequently used in the ANN literature, can be defined as

$$E = \frac{1}{2}\sum_{i=1}^{p} \|\mathbf{y}^{(i)} - \mathbf{d}^{(i)}\|^2. \quad \mathbf{(1)}$$

The back-propagation algorithm is a gradient-descent method to minimize the above squared-error cost function in Equation (1). It can be described as follows:

1. Set the weight and thresholds of the neuron to random values.
2. Present an input.
3. Calculate the output of the neuron.
4. Alter the weights to reinforce correct decisions and discourage wrong decisions, hence reducing the error. So for the network to learn we shall increase the weights on the active inputs when we want the output to be active, and to decrease them when we want the output to be inactive.
5. Now present the next input and repeat steps 3 to 4

There are many issues in designing feedforward networks. These issues include: (i) how many layers are needed for a given task?; (ii) how many units per layer?; (iii) what can we expect a network to generalize on data not included in the training set?; and (iv) how large should the training set be for "good" generalization? The following sections will answer most of these questions.

3.2.4 ANN Size

The size of the network plays an important and basic role in achieving the optimal result. The first step in defining the size of the network is to know the number of hidden layers that the network will consist of. In the case of ligature identification, one hidden layer will be enough. How would we know that one layer is enough? This question was answered by trying a number of networks; each consists of different number of hidden layers. As a result, the best performance was obtained by the network that consisted of one hidden layer.

The next step was to identify the number of processing elements (PEs) for each layer. This implies that each layer contains a vector of PEs and that the parameters selected apply to the entire vector. The parameters are dependent on the neural model, but all require a nonlinearity function to specify the behavior of the PEs. In addition, each layer has an associated learning rule and learning parameters.

With 35 inputs, 1 output and 1 hidden layer that contains 10 PEs, the network was built. The 35 inputs were the features that we collected in previous sections and the output was the decision of the network about whether the block is a ligature or character block (Table 3).

Table 3 Architecture of the ANN.

	PEs	Transfer Function
Input Layer	35	Linear
Hidden Layer 1	10	Tanh
Output Layer	1	Tanh

3.2.5 ANN Implementation

The design of the network described above was implemented using NeuroSolutions version 4.17, by NeuroDimensions, Inc. as it is shown in Figure 12.

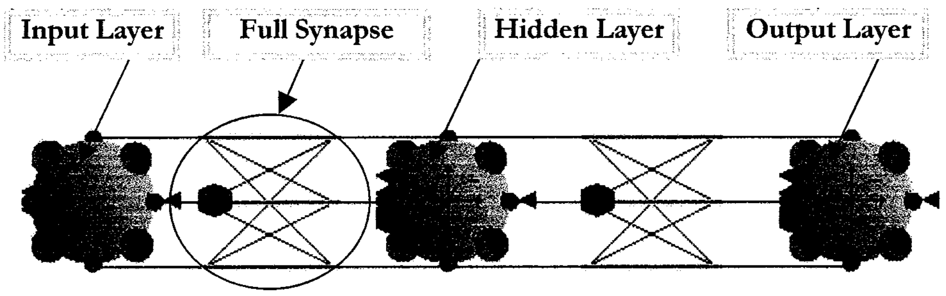


Figure 12 Ligature identification ANN
(NeuroSolution Software).

Each Layer in Figure 12 represents an axon, which is a vector of PEs. All axons are equipped with a summing junction at their input and a splitting node at their output. This allows multiple components to feed an axon, which then processes their accumulated activity.

Axons have two main functions. First they sum all of their inputs and then apply a function to that sum. The applied function may be either linear or nonlinear. The input axon, for example, simply applies an identity (linear) map between its input and output activity. All hidden and output axons apply the hyperbolic tangent map between its input and output. The Tanh axon used in these layers also applies a bias to each neuron in the layer. This will squash the range of each neuron in the layer to between -1 and 1.

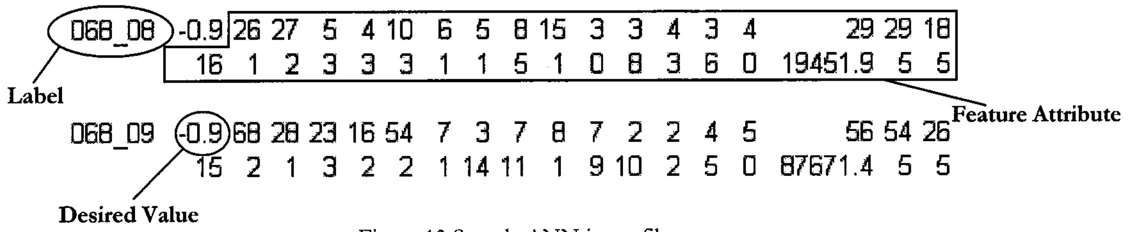
Axons can receive input from, and provide output to both axons and synapses within the network. The full synapse shown in Figure 3.10 provides a fully connected linear map between its input and output axons. Since each axon represents a vector of PEs, the full synapse simply performs a matrix multiplication. For each PE in its output axon, the full synapse accumulates a weighted sum of activations from all neurons in its input axons.

3.2.6 ANN Input Design

As we described in previous sections, features generated using a heuristic algorithm were validated manually in order to produce the input file for the ANN. This file, which consists of about 15000 records, was then split into three sets: training, cross validation and testing.

- Training: This set, which was the biggest one, about 60% of the main file, was used to train the network.
- Cross Validation: Which is about 25% of the main set. While the network is being trained, after each epoch, the network is tested using the cross validation set in order to adjust the network weights. We will discuss this issue in later sections.
- Testing: When training is complete, testing was done using the testing set, which is about 15% of the main set.

All of the three sets have the same format. Each record in the three files consists of a label, to identify the block being identified, the desired output, and the 35 values that represents the features of the block as shown in figure 13.



```
068_08 -0.9 26 27  5  4 10  6  5  8 15  3  3  4  3  4          29 29 18
             16  1  2  3  3  3  1  1  5  1  0  8  3  6  0  19451.9  5  5

068_09 -0.9 68 28 23 16 54  7  3  7  8  7  2  2  4  5          56 54 26
             15  2  1  3  2  2  1 14 11  1  9 10  2  5  0  87671.4  5  5
```

Label
Feature Attribute
Desired Value

Figure 13 Sample ANN input file.

3.2.7 ANN Training and Testing

Training is the process by which the free parameters of the network (i.e., the weights) get optimal values. The weights are updated using either supervised or unsupervised learning. In this system supervised learning is used because with supervised learning, the network is able to learn from the input and the error (the difference between the output and the desired response). The ingredients for supervised learning are therefore the input, the desired response, the definition of error, and a learning law. Error is typically defined through a cost function. Good network performance should result in a small value for the cost. A learning law is a systematic way of changing the weights such that the cost is minimized. In supervised learning the most popular learning law is backpropagation.

The network is trained in an attempt to find the optimal point on the performance surface, as defined by the cost definition. A simple performance surface is illustrated in figure 14. This network has only one weight. The performance surface of this system can be completely represented using a 2D graph. The x-axis represents the value of the weight, while the y-axis is the resulting cost. This performance surface is easy to visualize because it is contained within a two-dimensional space.

Backpropagation changes each weight of the network based on its localized portion of the input signal and its localized portion of the error. The change has to be proportional (a scaled version) of the product of these two quantities. The mathematics may be complicated, but the idea is very simple. When this algorithm is used for weight change, the state of the system is doing gradient descent; moving in the direction opposite to the largest local slope on the performance surface. In other words, the weights are being updated in the direction of down.

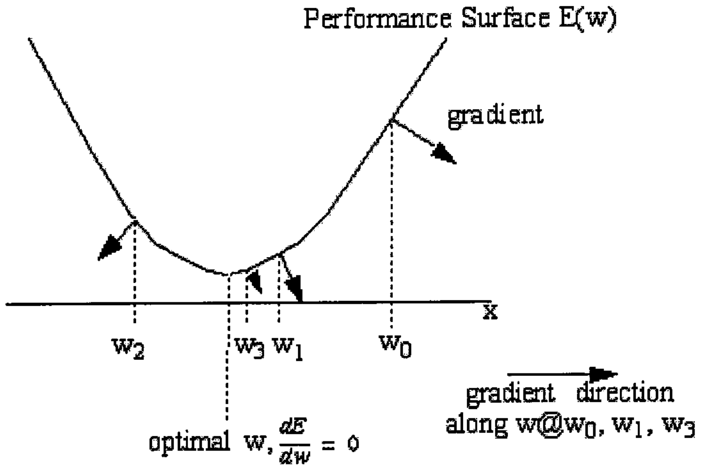


Figure 14 Simple performance surface.

3.2.7.1 Backpropagation

The beauty of backpropagation is that it is simple and can be implemented efficiently in computers. The drawbacks are just as important: The search for the optimal weight values can get caught in local minima, i.e.

the algorithm thinks it has arrived at the best possible set of weights even though there are other solutions that are better. Backpropagation is also slow to converge. In making the process simple, the search direction is noisy and sometimes the weights do not move in the direction of the minimum. Finally, the learning rates must be set heuristically.

An important issue is the selection of a step size. The idea is that the larger the step size the faster the minimum will be reached. However, if the step size is too large, then the algorithm will diverge and the error will increase instead of decrease. If the step size is too small then it will take too long to reach the minimum, which also increases the probability of getting caught in local minima.

Another issue is how to choose the initial weights. The search must start someplace on the performance surface. That place is given by the initial condition of the weights. In the absence of any a priori knowledge and to avoid symmetry conditions that can trap the search algorithm, the weights should be started at random values. However, the network's PEs have saturating nonlinearities, so if the weight values are very large, the PE can saturate. If the PE saturates, the error that goes through becomes zero, and previous layers may not adapt. Small random weight values will put every PE in the linear region of the sigmoid at the beginning of learning.

3.2.7.2 Termination of Training

The stop criteria for learning are very important. The stop criterion based on the error of the cross validation set was used in this network. Other methods limit the total number of iterations (hence the training time), stopping the training regardless of the networks performance. Another method stops training when the error reaches a given value. Since the error is a relative quantity, and the length of time needed for the simulation to get there is unknown, this may not be the best stop criterion. Another alternative is to stop on incremental error. This method stops the training at the point of diminishing returns, when an iteration is only able to decrease the error by a negligible amount. However, the training can be prematurely stopped with this criterion because performance surfaces may have plateaus where the error changes very little from iteration to iteration.

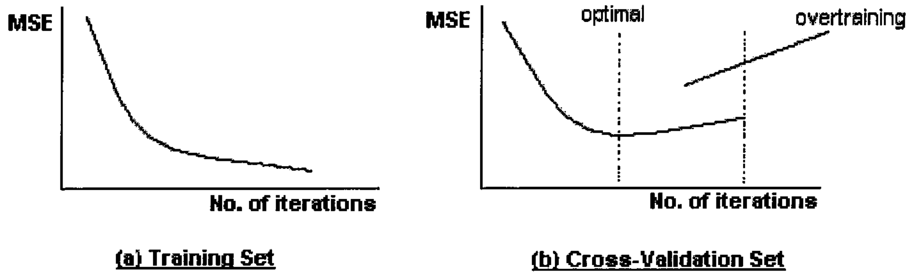


Figure 15 Behavior of MSE on training and cross-validation sets.

Cross validation computes the error in a test set at the same time that the network is being trained with the training set. It is known that the Mean Square Error (MSE) will keep decreasing in the training set, but may start to increase in the test set. This happens when the network starts "memorizing" the training patterns (Figure 15).

Chapter 4

HORIZONTAL SEGMENTATION

At this stage, we have a system that can perform vertical segmentation for the scanned image [Hamid 2001] and ligature block identification. We are left with the part that deals with horizontal segmentation, which will be our main concern in this chapter.

As it is shown in figure 16, Scanning, Binarization, Skeletonization, and Feature Extraction were done by [Hamid 2001] and were used to perform both vertical segmentation and horizontal segmentation. Ligature identification was discussed in chapter 3. When a block is identified as a ligature block, it will be prepared for horizontal segmentation. However, if a block was identified as a character block, it will be ready for the final step in this system which is the classification step.

In this chapter we will discuss briefly the first four steps which will be used for preparing the input of the network that will segment horizontally the ligature block. For further readings please refer to [Hamid 2001].

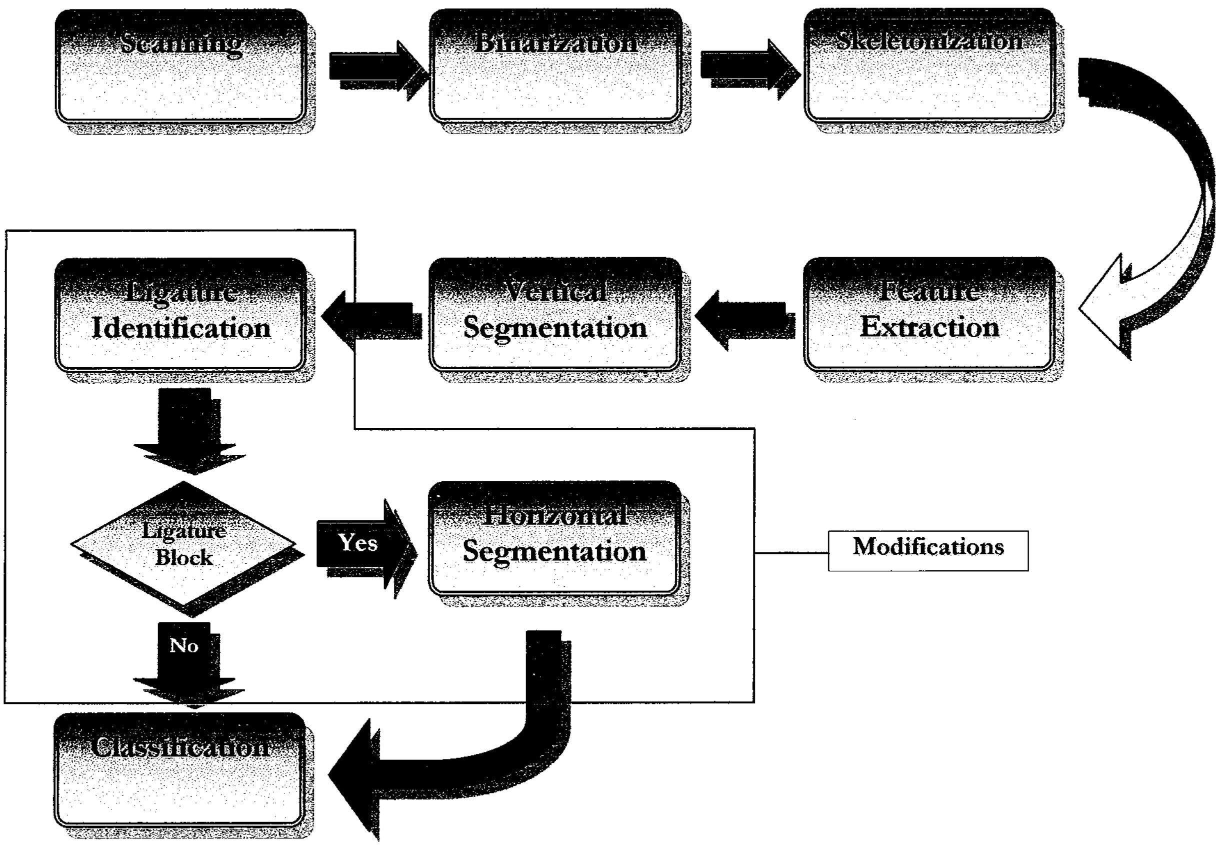


Figure 16 A map showing the modifications done to the system.

4.1 Horizontal vs. Vertical Segmentation

There are no major differences between horizontal and vertical segmentation, both perform the same tasks. However, instead of extracting features from the original Character Block (BC), features are extracted from a Transposed Character Block (TBC).

4.2 Transposed Character Block

In order to segment horizontally, each ligature block needs to be transposed so that it will be possible to extract features for each column (which is ordinarily a row in the ligature block) as it is shown in figure 17.

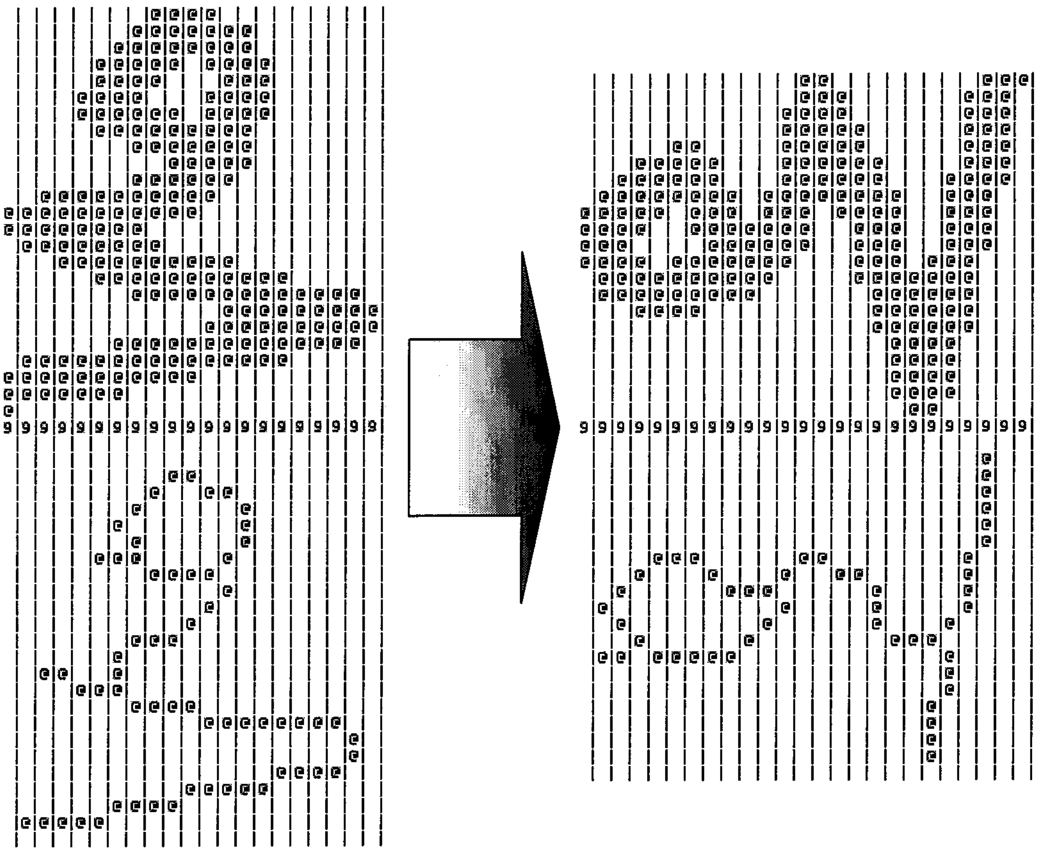


Figure 17 Ligature block and its corresponding transposed block.

4.3 The Data Set

In the case of horizontal segmentation, there is no need for collecting complete phrases, since the main objective is to segment ligature blocks. Samples were acquired from various students and faculty members at the Lebanese American University and from employees at Gezairi Transport

Company. These samples, which were about 150, included only ligatures. A sample is shown in figure 18.

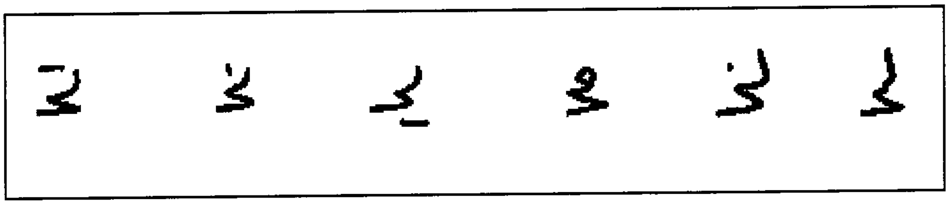


Figure 18 Sample ligature blocks collected

4.4 Common Tasks

In this section we will briefly discuss the main common tasks that are done while segmenting horizontally and vertically.

1. Binarization: The main task in binarization is to convert the bitmap image into binary representations. First, bitmaps were converted into monochrome bitmap form. A heuristic algorithm [Hamid 2001] was used to generate a matrix of ones (1) and zeroes (0) for each image. This algorithm scans the monochrome bitmap and converts each black pixel into the character "0" and each white one with the character "1".
2. Character Block Extraction: A heuristic algorithm was implemented with a 94% accuracy, scanned the whole binary matrix of the image and performed the following steps:
   - A black pixel was identified as pepper and discarded if it had a maximum of one black neighbor pixel.

- Recursively, identify each group of connected black pixels as an object.
- Classify objects as child or parent ones. If the weight, or black pixel density, of the object is less than half of average weight of all objects, then it is marked as a child. Otherwise, it is marked as a parent.
- For each child object, determine the distance to all parents.
- For each child object, determine the distance range, *R*, for all possible parents, which is equal to 150% of the distance to the nearest parent.
- Merge all children to all parents who are at a maximum distance of *R*.

3. Skeletonization: Skeletonization, or thinning, is an image-processing step that reduces BCs to their skeletons, i.e., transforming characters into arc segments one pixel thick. A skeletonization algorithm must not alter the shape of the BC. This includes the preservation of connected components and the number of cavities and holes. The skeletonization process is required in order to extract certain features like corner points, end points, and fork points. An adaptation of Rosenfeld's skeletonization algorithm produced acceptable results with few enhancements and modifications [Rosenfeld].

As it is shown in figure 19, the system starts the process of segmentation (horizontal and vertical) by scanning the document in monochrome bitmap format. Suppose in this case "الحمرا" is the word that should be segmented. At the second level, binarization takes place, followed by the process of character block extraction. The output of the second level will be three blocks of characters. These are the blocks that are not connected to each others.

Here comes the process of vertical segmentation, the system will identify valid segmentation points in the BCs. Since only the 2nd BC contains more than one character, then it will be segmented into three other BCs.

Now we have five BCs, and one of them is a ligature block. At this stage we need to identify which one is a ligature block. The five BCs will be tested, and the ligature block will be identified using the ANN that was explained in chapter three. The final step is to segment the identified ligature block horizontally. This process is done by the ANN that will be discussed in this chapter.

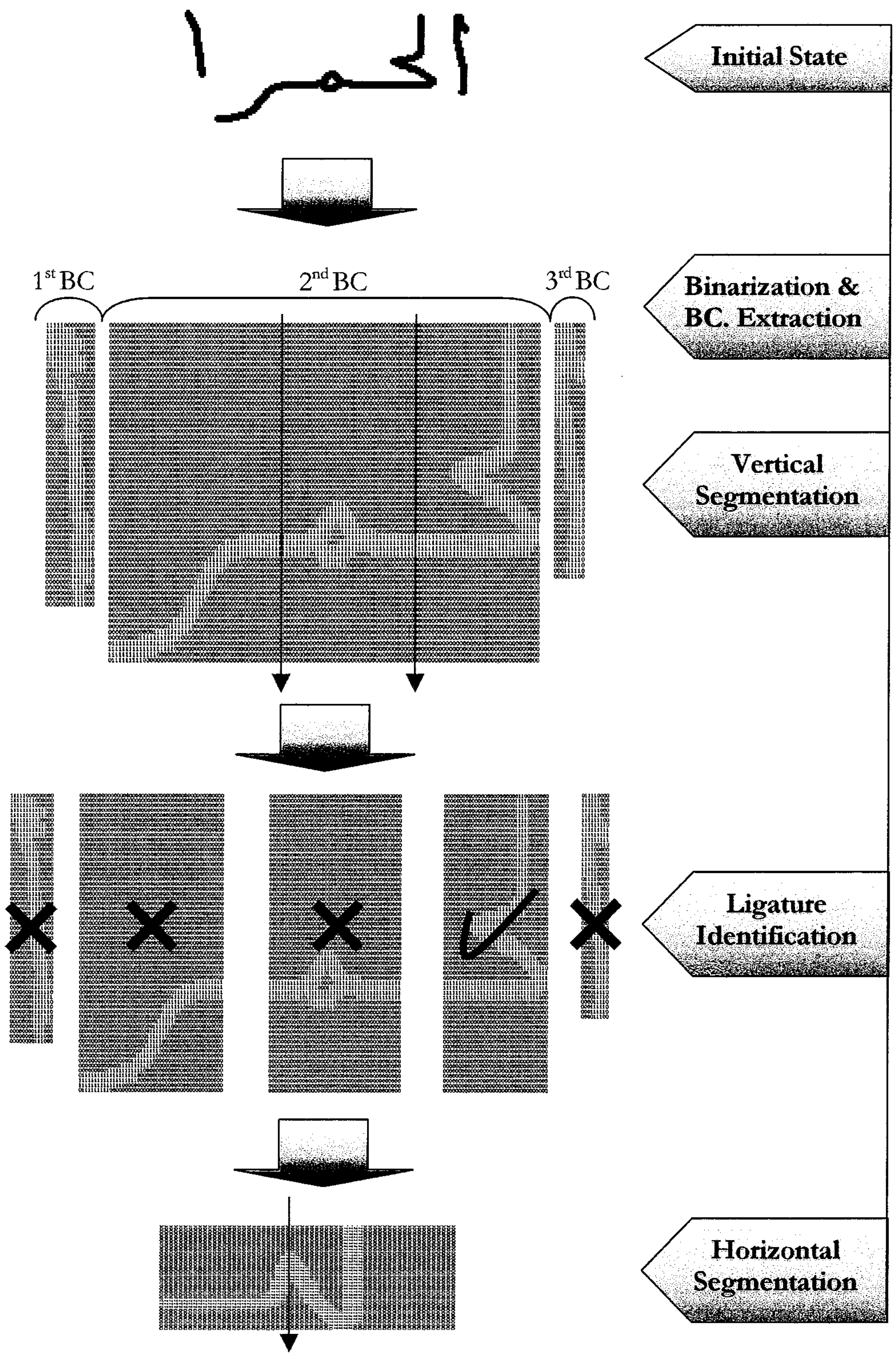


Figure 19 Detailed description of the system.

4.5 Feature Extraction

Feature extraction is a process used to get information suitable for use in segmentation and recognition. In the previous chapter we discussed the method used to extract useful features. This method will be used the same to extract features for horizontal segmentation, however, the features will be different.

Table 4 shows the major features used in horizontal segmentation. Holes, corner points, fork points, and end points were part of the features.

Table 4 Major features extracted for each column of BC.

Feature	**Attributes**	**Description**	**Attribute Value**
Image width and height		Image width and height in pixels.	(0,∞)
Black pixel density	Black pixel density / width	Number of black pixels in the row divided by the image width.	[0,1]
	Density minima	Does the row cross a density minimum?	0 or 1
	Density maxima	Does the row cross a density maximum?	0 or 1
Transitions	Number of transitions crossed	Scan the image horizontally and count the number of foreground-background and background-foreground transitions crossed by row.	[0,height)
Holes	Number of holes crossed	Count the number of holes (or islands of white pixels completely surrounded by black pixels) crossed by row.	[0,height)
	Total hole densities / width	Total number of hole pixels crossed by row divided by the image width.	[0,1]
Endpoints	Number of endpoints crossed	Number of endpoints crossed by row.	[0,height]
Corner points	Number of corners crossed	Number of corner points crossed by row.	[0,height]
Fork points	Number of fork points crossed	Number of fork points crossed by row.	[0,height]
Relative index of row in image		Index of row divided by image width.	[0,1]
Upper and lower contours	Upper and lower contour index / width	Index of upper most and lower most black pixel crossed by row divided by image width.	[0,1]
	Upper and lower contour minima or maxima	Does the row cross an upper or lower contour minima or maxima?	0 or 1
Feature relationships	Index of nearest left and right feature / 100	Index of nearest left and right feature in a 100-pixel width range.	[0,1]

4.6 ANN Architecture

The ANN architecture used was the same as the one used for ligature identification with some few modifications. Since horizontal segmentation needs more feature than ligature identification, the ANN was more complex. In the next few sections we will discuss only the modifications.

4.6.1 ANN Size

The best ANN architecture reached consisted of 52 inputs, 1 output, and 2 hidden layers. The 52 inputs were feature attribute of a pre-segmentation point and the output was the validity of the point. The ANN architecture is summarized in Table 5.

Table 5 Architecture of the ANN.

	PEs	Transfer Function
Input Layer	52	Linear
Hidden Layer 1	30	Tanh
Hidden Layer 2	15	Tanh
Output Layer	1	Tanh

4.6.2 ANN Implementation

The design of the segmentation ANN described was implemented using NeuroSolutions, Version 4.0 by NeuroDimensions, Inc. the implemented ANN is shown in figure 20.

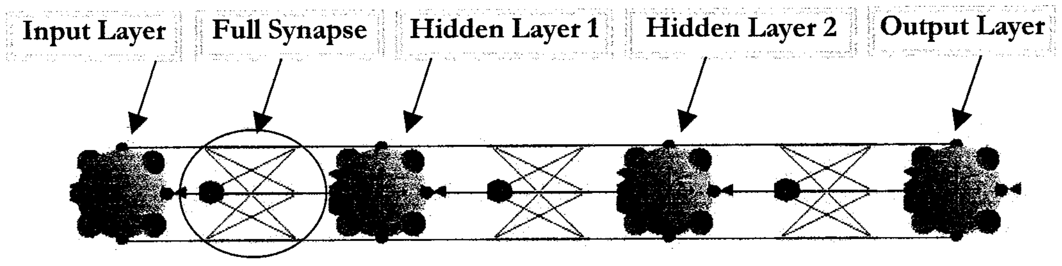


Figure 20 Segmentation ANN used to validate pre-segmentation points.

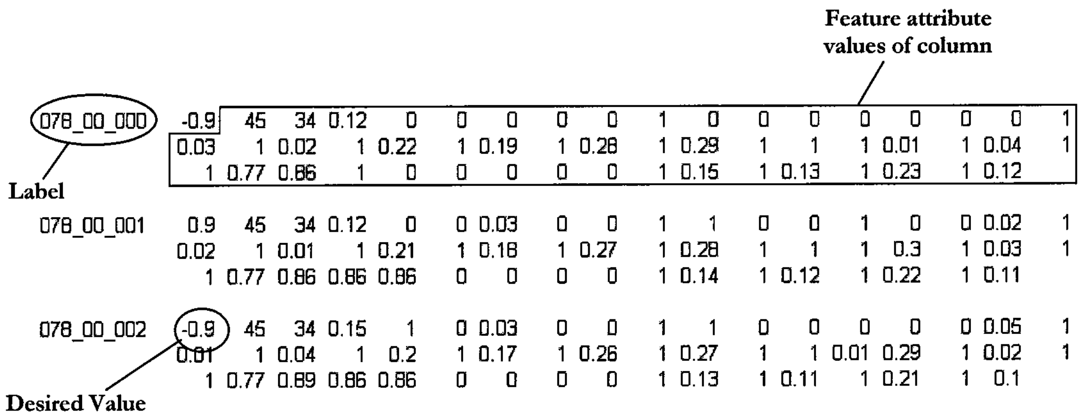
4.6.3 ANN Input Design

The pre-segmentation points generated by the heuristic module were validated manually in order to produce input files for the training, cross validation, and testing phases of the ANN. 64,000 pre-segmentation points were evaluated manually and divided into three parts as shown in Table 6.

Table 6 Manually evaluated input sets points.

Input Set	**Number of exemplars**
Training set	25,600
Cross-validation set	25,600
Testing set	12,800

Each pre-segmentation point is represented by a row in the input files. Each row starts with the ID of the column, its desired value, and then the feature attributes are listed. A sample input file is shown in figure 21.


Feature attribute values of column
Label
Desired Value

```
078_00_000  -0.9   45   34 0.12    0    0    0    0    0    1    0    0    0    0    0    0    0    1
            0.03    1 0.02    1 0.22    1 0.19    1 0.28    1 0.29    1    1    1 0.01    1 0.04    1
               1 0.77 0.86    1    0    0    0    0    0    1 0.15    1 0.13    1 0.23    1 0.12

078_00_001   0.9   45   34 0.12    0    0 0.03    0    0    1    1    0    0    1    0    0 0.02    1
            0.02    1 0.01    1 0.21    1 0.18    1 0.27    1 0.28    1    1    1  0.3    1 0.03    1
               1 0.77 0.86 0.86 0.86    0    0    0    0    1 0.14    1 0.12    1 0.22    1 0.11

078_00_002  -0.9   45   34 0.15    1    0 0.03    0    0    1    1    0    0    0    0    0 0.05    1
            0.01    1 0.04    1  0.2    1 0.17    1 0.26    1 0.27    1    1 0.01 0.29    1 0.02    1
               1 0.77 0.89 0.86 0.86    0    0    0    0    1 0.13    1 0.11    1 0.21    1  0.1
```

Figure 21 Sample ANN input file.

As shown in figure 21, the first and the third row are invalid segmentation points, since the desired value is equal to -0.9. However, the second one is a valid segmentation point, since the desired value is equal to 0.9. The labels are obtained from the image file name, i.e., the three rows shown in figure 4.6 are obtained from the image "078_00" and each row is labeled with a unique ID starting from "000".

Chapter 5

EXPERIMENTAL RESULTS

Implementation and experimentation of the system were performed on a Pentium III 800 MHz computer with 256 RAM . After implementation, the heuristic feature extraction algorithm in conjunction with the ANN was trained and tested on the scanned words mentioned in previous chapters. We will discuss experimental results on both ligature identification and horizontal segmentation in the next sections.

5.1 Performance Evaluation

No Standardized test sets exist for character recognition, and as the performance of an OCR system is highly dependent on the quality of the input, this makes it difficult to evaluate the system. Still, recognition rates are often given, and usually presented as the percentage of points correctly identified. However, this does not say anything about the error committed. Therefore in evaluation of the system, two different performance rates should be investigated:

- Recognition rate: The proportion of correctly identified points.
- Error rate: The proportion of characters incorrectly identified

Because of the time required to detect and correct OCR errors, the error rate is the most important when evaluating whether an OCR system is cost-effective or not.

5.2 Ligature Identification Segmentation Results

The testing set for ligature identification was about 2250 exemplars. About 42% of the testing set consisted of ligature blocks, and the rest were character blocks.

Table 7 describes the total number of blocks that should be identified, the total number of blocks that are ligatures, and the total number of blocks that are character blocks.

Table 7 Distribution of blocks within the testing set.

	Total	**Percentage**
Ligature Blocks	945	42%
Character Blocks	1305	58%
Total Number of Blocks	2250	100%

The output range of the ANN was between -0.9 and 0.9. A positive value indicated that a block is a valid ligature block; a negative value indicated that a block is a character block.

An algorithm checked the results and defined the identification of blocks as ligature or characters blocks. As shown in figure 22, 79% of all the blocks were identified correctly (57% consisting of character blocks identified correctly and 22% consisting of ligature blocks identified correctly).

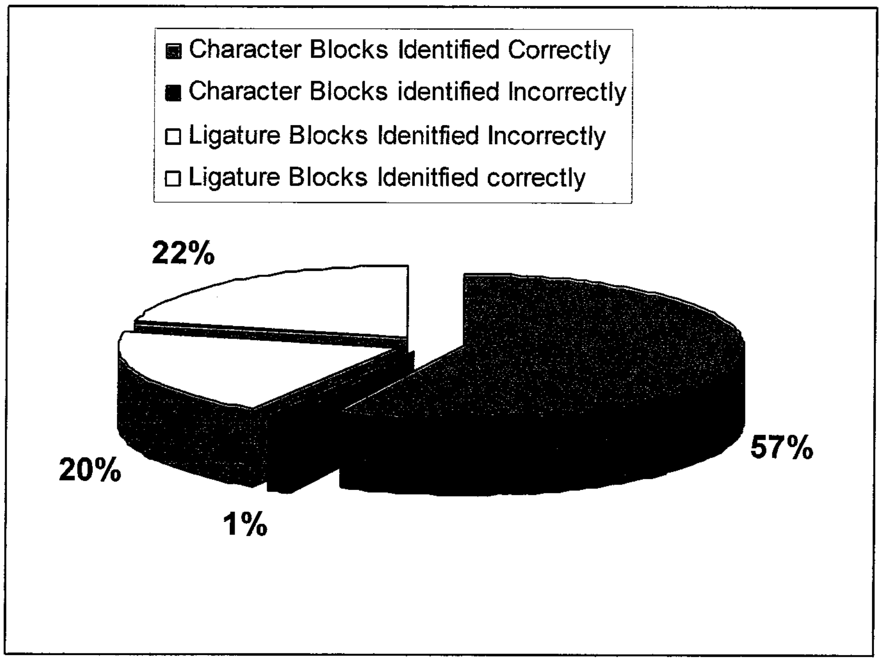


Figure 22 Percentages of the ligature identification ANN results.

The recognition rate in ligature identification would be 79% according to figure 22, while the error rate would be 22%. Table 8 describes the number of ligature blocks identified correctly and incorrectly and the number of character blocks that were identified correctly and incorrectly.

Table 8 Table 5.2 Number of correctly and incorrectly identified blocks.

	Correctly Identified	Incorrectly Identified	Total
Ligature Blocks	495	450	945
Character Blocks	1283	22	1305
Total	1778	472	2250

5.3 Horizontal Segmentation Experimental Results

The testing set for ligature identification was about 13,000 exemplars. About 10% of the testing set consisted of valid segmentation points, and the rest were invalid segmentation points.

Table 9 describes the total number of points that should be identified, the total number of points that are valid segmentation points, and the total number of points that are invalid segmentation points.

Table 9 Distribution of segmentation points within the testing set.

	Total	Percentage
Valid segmentation points	1300	10%
Invalid segmentation points	11700	90%
Total Number of points	13000	100%

As we mentioned above, the output range was between -0.9 and 0.9. A positive value indicated that a point is a valid segmentation point; a negative value indicated that a point is an invalid segmentation point.

As shown in figure 23, 91% of all points were identified correctly as valid or invalid points (88% consisting of invalid points identified correctly and 3% consisting of valid points identified correctly).

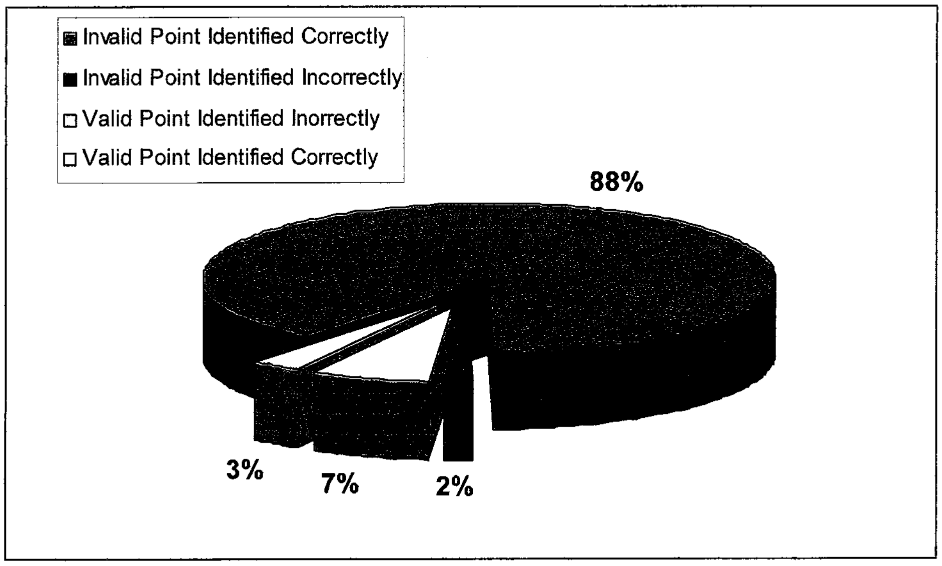


Figure 23 Percentages of the horizontal segmentation ANN results.

The recognition rate in horizontal segmentation would be 91% according to figure 23, while the error rate would be 9%. Table 10 describes the number of valid segmentation points identified correctly and incorrectly and the number of invalid segmentation points that were identified correctly and incorrectly.

Table 10 Number of correctly and incorrectly identified points.

	Correctly Identified	Incorrectly Identified	Total
Valid segmentation points	390	910	1300
Invalid segmentation points	11440	260	11700
Total	11830	1170	13000

Out of 1300 segmentation points that should be identified as valid, the ANN identified 390, which is about 1/3 of the total number of valid segmentation points.

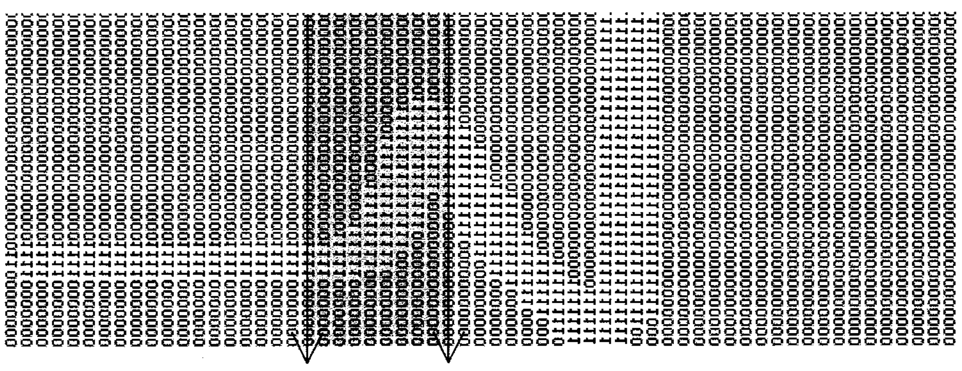


Figure 24 Range of valid segmentation points.

Figure 24 shows the range of segmentation points that should be identified as valid. It is enough that at least one point out of all segmentation points included in this range would be identified by the ANN as valid segmentation point. This makes the result of the ANN a good result, since 1/3 of the total valid segmentation points were identified by the ANN. This will guarantee that in each ligature block, at least one valid segmentation point will be identified by the ANN, which is enough to cut the ligature into two separate characters.

5.4 Comparison of Segmentation Results.

Many researchers have used various techniques for the segmentation of characters in handwritten words. Segmentation accuracy rates of above 90% were achieved by [Lee 1996]; however, the authors were only dealing with printed Latin alphanumeric characters. [Srihari 1993] obtained segmentation accuracies of 83% for handwritten zip codes (no alphanumeric). [Han 1995] achieved an 85.7% accuracy using a heuristic algorithm for the segmentation of words on 50 envelopes from real mail pieces. Finally, experiments conducted by [Eastwood 1997], segmenting cursive handwriting produced a 75.9% accuracy rate using an ANN-based method.

Table 11 Comparison of segmentation results in the literature.

Author	**Segmentation accuracy [%]**	**Data set used**	**Method used**
Blumenstein and Verma [Blumenstein1997]	81.21	Griffith University Latin handwriting database	Neuro-conventional method
Eastwood et al. [Eastwood1997]	75.9	Cursive Latin handwriting from CEDAR database.	ANN-based method
Han and Sethi [Han1995]	85.7	Latin handwritten words on 50 real mail envelopes	Heuristic algorithm
Lee et al. [Lee1996]	90	Printed Latin alphanumeric characters.	ANN-based method
Srihari et al. [Srihari1993]	83	Handwritten zip codes (no alphanumeric)	ANN-based method

On average our segmentation accuracy using the neuro-conventional technique was about 91% for horizontal segmentation and 80% for ligature identification. Table 11 summarizes the results obtained by various researchers.

Chapter 6

CONCLUSION AND FUTURE WORK

The work presented in this paper is the continuation of what [Hamid 2001] has started. In his paper there was one network that was responsible of segmenting blocks vertically. He was unable to segment what is called ligatures, since ligatures are set of characters situated one above the other and share the same vertical space. That's why there is no vertical line that can cut two or more characters within a ligature block.

In this paper, we have presented two main Artificial Neural Networks. The first has the job of identifying ligature blocks. However, the second is responsible of horizontally segmenting all ligature blocks identified by the first one.

The segmentation phase proved to be successful in both networks. The first one, dealing with ligature identification, had a recognition rate of 80%. However, the other one, which is responsible of the horizontal segmentation process, had a recognition rate of over 90%.

In future work, ligature identification and horizontal segmentation could be somehow improved. For example, extracting more features could improve both ligature identification and horizontal segmentation. Moreover, ANN's could be trained more so that it will give better and accurate results.

Finally, a classification system should be integrated into the overall system to obtain a complete Arabic handwritten recognition system.

REFERENCES

Bellegarda, J.B., Nahamoo, D., Nathan, K.S. and Bellegarda, E.J. (1994) Supervised hidden Markov modeling for on-line handwriting recognition. In International Conference on Acoustics, Speech and Signal Processing, volume 5, pp. 149-152

Bengio, Y., Simard, P. and Frasconi, P. (1994) Learning long-term dependencies with gradient descent difficult. IEEE Transactions on Neural Networks 5 (2): 157-166

Bos, B. and van der Moer, A. (1993) The Bakunin project and optical character recognition. In (OCRHD 1993), pp. 11-15

Breuel, T.M. (1994) A System for the off-line recognition of handwritten text. Technical Report 94-02, IDIAP, CP 609, 1920 Martigny, Switzerland.

Cheriet, M. and Suen, C.Y. (1993) Extraction of key letters for cursive script recognition. Pattern Recognition Letters 14: 1009-1017.

Eastwood, B., Jennings, A. and Harvey, A.. (1997) A feature based neural network segmenter for handwritten words. Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 1997), Gold Coast, Australia, pp. 286-290

Edelman, S., Ullman, S. and Flash, T. (1990) Reading cursive script by alignment of letter prototypes. International Journal of Computer Vision 5 (3): 303-331.

Elliman, D.G. and Banks, R.N. (1991) A comparison of two neural networks for hand-printed character recognition. In IEE 2nd Neural Networks, number 349 in IEE, pp. 224-228.

Fukushima, K. et. al. (1983) Neocognitron: A neural network model for a mechanism of visual pattern recognition. IEEE Trans. on Systems, Man and Cybernetics. pp. 826-834.

Garris, M.D., Blue, J.L., Candela, G.T., Dimmick, D.L., Geist, J., Grother, P.J., Janet, S.A. and Wilson, C.L., (1994) NIST form-based handprinted recognition system. Document Understanding Mailing List.

Hamid, A. (2001) A neuro-heuristic approach for segmenting handwritten Arabic text. Lebanese American University. Masters thesis.

Han, K. and Sethi, I.K.. (1995) Off-line cursive handwriting segmentation. In (ICDAR 1995). Montreal, Canada, pp. 894-897

Haykin, S. (1994). A comprehensive foundation. Neural Networks. Prentice Hall.

Hepp, D.J. (1991) An application of backpropagation to the recognition of handwritten digits using morphologically derived features. Proceedings of the SPIE 1451: 228-233.

Hull, J.J. (1993) A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Idan, Y. and Chevalier, R.C. (1991) Handwritten digits recognition by a supervised Kohonen-like learning alogorithm. IJCNN 91 3: 2576-2581.

Impedovo, S. ed. (1994) Fundamentals in Handwriting Recognition, volume 124 of NATO ASI Series F: Computer and Systems Sciences. Springer Verlag.

Kimura, F., Shridhar, M. and Chen, Z. (1993) Improvements of a lexicon directed algorithm for recognition of unconstrained handwritten words. In (IWFHR 1993), pp. 122-131.

Lee, S. and Kim, Y. (1995) A new type of recurrent neural network for handwritten character recognition. IEEE.

Lee, S.W., Lee, D.J. and Park, H.S. (1996) A new methodology for gray-scale character segmentation and recognition. IEEE Transaction on Pattern Analysis and Machine Intelligence, 18, pp. 1045-1051

Leroux, M., Salome, J.C. and Bardard, J. (1991) Recognition of cursive script words in a small lexicon. In (ICDAR 1991), pp. 774-782.

Lu, S.W., Ren, Y. and Suen, C.Y. (1991) Hierarchical attributed graph representation and recognition of handwritten Chinese characters. Pattern Recognition 24 (7): 617-632.

Martins, W. and Allison, N.M. (1991) Visual search of postal codes by neural networks using human examples. In IEE 2nd Conference on Neural Networks.

Moreau, J.V., Plessis, B., Bougeois, O. and Plagnaud, J.L. (1991) A postal cheque reading system. In (ICDAR 1991), pp. 758-766.

Mori, S., Suen, C.Y. and Yamamoto, K. (1992) Historical review of OCR: research and development. Document Image Analysis. pp. 244-269.

Mori, Y. and Yokosawa, K. (1988) Neural networks that learn to discriminate similar Kanji characters. Neural Information Processing Systems.

Nagy, G. (1982) Optical character recognition: Theory and Practice. Handbook of Statistics. Krishnaiah and Kanal, eds. volume 2. pp. 621-649.

Nellis, J., and Stonham, T.J. (1991) A fully integrated hand-printed character recognition system using artificial neural networks. In IEE 2nd Conference on Neural Networks, number 349 in IEE, pp. 219-223.

Palumbo, P.W., Srihari, S.N., Soh, J., Sridhar, R. and Demajenko, V. (1992) Postal address block location in real time. IEEE Computer 25 (7): 34-42.

Pavlidis, T. (1993) Recognition of printed text under realistic conditions. Pattern Recognition Letters 14: 317-326.

Schenkel, M., Guyon, I. and Henderson, D. (1994) On-line cursive script recognition using time delay neural networks and hidden Markov models. In International Conference on Acoustics, Speech and Signal Processing, volume 2, pp. 637-640.

Simon, J.C. (1992) Off-line cursive word recognition. Proceedings of the IEEE 80 (7): 1150-1161.

Srihari, S.N. and Bozinovic, R.M. (1987) A multi-level perception approach to reading cursive script. Artificial Intelligence 33: 217-255.

Srihari, S.N., Govindaraju, V. and Shekhawat, A. (1993) Interpretation of handwritten addresses in US mailstream. In (ICDAR 1993), pp. 291-294.

Suen, C.Y., Berthod, M. and Mori, S. (1980) Automatic recognition of handprinted characters – the state of the art. Proc. IEEE 68 (4): 469-487.

Waibel, A. and Lee, K.F. eds. (1990) Readings in Speech Recognition. Morgan Kaufmann.

Wang, C.H. and Srihari, S.N. (1988) A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces. International Journal of Computer Vision 2: 125-151.

Yanikoglu, B.A. and Sandon, P.A. (1993) Off-line cursive handwriting recognition using style parameters. Technical Report PCS-TR93-192, Dartmouth College, NH.