

RT
394

A Neuro-Heuristic Approach for Segmenting Handwritten Arabic Text

By

Alaa A. Hamid

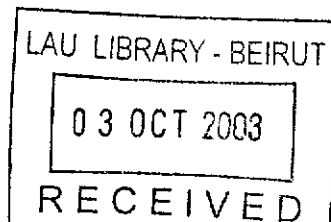
**Submitted in partial fulfillment of the requirements
For the degree of Master in Science in Computer Science**

Thesis Advisor: Dr. Ramzi Haraty

**Natural Sciences Division
LEBANESE AMERICAN UNIVERSITY**

Beirut

February 2001



A Neuro-Heuristic Approach for Segmenting Handwritten Arabic Text

By

Alaa Hamid

Signatures Redacted

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

Signatures Redacted

Dr. Nashaat Mansour
Associate Professor of Computer Science
Lebanese American University

Signatures Redacted

Dr. May Abboud
Associate Professor of Computer Science
Lebanese American University

A NEURO-HEURISTIC APPROACH FOR SEGMENTING HANDWRITTEN ARABIC TEXT

ABSTRACT

By

Alaa A. Hamid

The segmentation and recognition of Arabic handwritten text has been an area of great interest in the past few years. However, a small number of research papers and reports have been published in this area. There are several major problems with Arabic handwritten text processing: Arabic is written cursively and many external objects are used such as dots, 'Hamza', 'Madda', and diacritic objects. In addition, Arabic characters have more than one shape according to their position inside a word. More than one character can also share the same horizontal space, creating vertically overlapping connected or disconnected blocks of characters. This makes the problem of segmentation of Arabic text into characters, and their classification even more difficult.

In this work a technique is presented that segments difficult handwritten Arabic text. A conventional algorithm is used for the initial segmentation of the text into connected blocks of characters. The algorithm then generates pre-segmentation points for these blocks. A neural network is subsequently used to verify the accuracy of these segmentation points. Another conventional algorithm uses the verified segmentation points and segments the connected blocks of characters. These characters can then be used as input to another neural network for classification.

Two major problems were encountered in the above scenario. First, the segmentation phase proved to be successful in vertical segmentation of connected blocks of characters. However, it couldn't segment characters that were overlapping horizontally, and this affects any neural network classifier.

Second, there are a lot of handwritten characters that can be segmented and classified into two or more different classes depending on whether you look at them separately, or in a word, or even in a sentence. In other words, character segmentation and classification, especially handwritten Arabic characters, depends largely on contextual information, and not only on topographic features extracted from these characters.

Contents

CHAPTER 1 INTRODUCTION	9
1.1. Statement of the problem	9
1.2. Purpose and importance of the study	10
1.3. Layout of the thesis.....	11
CHAPTER 2 ARTIFICIAL NEURAL NETWORKS.....	12
2.1. What are ANNs?.....	12
2.1.1. Analogy to the brain.....	12
2.1.2. Artificial neurons and how they work	14
2.1.3. Comparison of traditional computing and ANNs.....	15
2.1.4. Comparison of expert systems and ANNs	15
2.2. History of ANNs.....	17
2.3. Major components of an ANN and how they work.....	18
2.4. ANN's learning methods	21
2.5. Multi-layer perceptrons and generalized feedforward networks	21
CHAPTER 3 LITERATURE REVIEW.....	23
3.1. History of optical character segmentation and recognition	23
3.2. Comparison of segmentation and recognition techniques.....	24
3.2.1. Segmentation techniques.....	24
3.2.2. Preprocessing and feature extraction techniques.....	27
CHAPTER 4 NEURO-CONVENTIONAL SEGMENTATION METHOD	28
4.1. Character Segmentation and Recognition Obstacles	28
4.1.1. General difficulties	28
4.1.2. Handwritten text specific difficulties.....	29
4.1.3. Arabic text specific difficulties.....	30
4.2. Proposed Technique	33
4.2.1. The data set.....	33
4.2.2. Binarization	34
4.2.3. Character block extraction	34
4.2.4. Skeletonization	36
4.2.4.1. Black pixels properties	36
4.2.4.2. Skeletonization algorithm.....	38
4.2.5. Feature extraction	40
4.2.5.1. Feature set design.....	40

4.2.5.2.	Extraction method	41
4.2.6.	Pre-segmentation point generation	46
4.2.7.	Pre-segmentation point validation by ANN	47
4.2.7.1.	ANN architecture	47
4.2.7.1.1.	ANN Size	48
4.2.7.1.2.	Transfer functions	49
4.2.7.1.3.	ANN Implementation	50
4.2.7.1.4.	ANN input design	54
4.2.7.2.	ANN Training and Testing	55
4.2.7.2.1.	Back-propagation learning method	56
4.2.7.2.2.	Error Criteria	57
4.2.7.2.3.	Back-Propagation Tanh Axon	58
4.2.7.2.4.	Momentum learning rule	58
4.2.7.2.5.	Termination of training process	62
CHAPTER 5	EXPERIMENTAL RESULTS	64
5.1.	BC extraction results	64
5.2.	Segmentation ANN results	64
5.2.1.	Un-segmentable Objects	65
5.2.2.	Experimental Results of different ANN architectures	66
5.2.3.	Applying threshold to ANN results	69
5.3.	Comparison of segmentation results in the literature	71
CHAPTER 6	CONCLUSION AND FURTHER WORK	73
APPENDIX	A PRELIMINARY DESIGN FOR AN OCR NEURAL NETWORK	75
A-1.	The Data Set	75
A-2.	Feature Set Design	75
A-3.	Classification ANN	77
A-3.1.	ANN Size	77
A-3.2.	ANN Input Design	80
A-3.3.	Results	81
A-4.	Conclusion and Future Work	81
REFERENCES	83

List of Figures

Figure 1. A typical biological neuron [Camargo1990].	13
Figure 2. A basic artificial neuron.	14
Figure 3. A MLP.	22
Figure 4. Two lines occupying a shared vertical space.	30
Figure 5. A Hamza external object.	30
Figure 6. The shapes the character ح takes according to its position inside a word.	31
Figure 7. Three characters written in completely different ways.	31
Figure 8. The letter س may be missed when appearing in the middle of a word.	31
Figure 9. Characters with similar contours.	32
Figure 10. Two characters occupying a shared horizontal space.	32
Figure 11. Arabic ligatures and their constituent characters.	33
Figure 12. A sample handwritten address.	34
Figure 13. Two examples where pl is not an 8-simple point.	36
Figure 14. An example where pl is an eight-isolated point.	37
Figure 15. Two examples where pl is an eight-endpoint.	37
Figure 16. Four examples where pl is a border point.	38
Figure 17. An example of a skeletonized BC.	39
Figure 18. An example of a hole in a normal image (right) and a skeletonized image (left).	42
Figure 19. Four examples of corner points.	42
Figure 20. Three examples where pl is a fork point.	42
Figure 21. An example of extracted features.	46
Figure 22. Sigmoids corresponding to: (a) Equation 1, (b) Equation 2 and (c) Equation 3.	50
Figure 23. The mapping for the PE of the axon family.	51
Figure 24. Segmentation ANN used to validate pre-segmentation points. (Source: NeuroSolutions Software)	52
Figure 25. Sample ANN input file.	54
Figure 26. A simple performance surface.	55
Figure 27. Segmentation ANN used to validate pre-segmentation points with the back-propagation layer added. (Source: NeuroSolutions Software)	60
Figure 28. Behavior of MSE on training and cross-validation sets.	62
Figure 29. Range of a valid segmentation point.	65
Figure 30. Three words containing miss-located external objects.	66
Figure 31. Distribution of ANN responses	70
Figure 32. Architecture of classification ANN.	79
Figure 33. Sample ANN input file.	80

List of Tables

Table 1. Comparison of traditional computing and ANNs.....	15
Table 2. Comparison of ES and ANNs.	16
Table 3. Major features extracted for each column of BC.	44
Table 4. Architecture of Segmentation ANN.....	49
Table 5. The full synapses used in the segmentation ANN.	51
Table 6. Manually evaluated input sets points.	54
Table 7. Optimum learning parameter values reached in the segmentation ANN. ...	59
Table 8. Summary of un-segmentable objects.....	65
Table 9. Segmentation ANN results using 48,000 training exemplars, tested on 10,000 exemplars.....	68
Table 10. ANN results after rejecting patterns with responses in the (0,0.5) range ..	71
Table 11. Comparison of segmentation results in the literature.....	72
Table 12. Major features extracted for each image to be classified.	76
Table 13. 52 classes of Arabic images.	77
Table 14. Architecture of Classification ANN.....	78
Table 15. Manually evaluated input sets.	80

Chapter 1 Introduction

Despite the widespread availability of computers, or perhaps because of it, an ever-increasing amount of data is obtained in handwritten form on paper. The handwriting recognition problem is the problem of transforming this data into machine-readable form, accurately and quickly.

In the following chapter, the problem of handwritten Arabic text segmentation and recognition is described. The motivation behind this research and its industrial and scientific importance is also discussed. Later, the thesis organization is presented.

1.1. Statement of the problem

This research describes a hybrid method to segment Arabic handwritten text. The method contains two main components. The first is a heuristic algorithm, which is responsible for scanning handwritten text, extracting blocks of connected characters (BCs), and then extracting features to be used in the second component. It is also responsible for generating pre-segmentation points, which are validated by the second component, the ANN. The ANN verifies whether all the segmentation points found are correct or incorrect.

The main problem with handwritten text recognition is the presence of lines, non-character objects, and noise or ‘salt-and-pepper’ in the scanned image. Characters can also

1.3. Layout of the thesis

The remainder of this thesis is divided into 5 chapters. Chapter 2 is an introduction to the field of artificial neural networks and the techniques used in this research. If the reader feels he (or she) has adequate knowledge of some of the theoretical topics of neural networks, this chapter can be skipped. However, in the rest of the thesis, references will be made to terms introduced in this chapter whenever necessary.

Chapter 3 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 4 describes in detail the proposed approach used to segment Arabic handwritten text. The design of the heuristic and neural network components is also presented.

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

Chapter 2 Artificial Neural Networks

2.1. What are ANNs?

Artificial neural networks, or ANNs, are collections of mathematical models that simulate some of the observed properties of biological nervous systems. The key element of ANNs is the structure of its information processing system. It is composed of a large number of highly interconnected processing elements (PEs) that are analogous to biological neurons and are linked together with weighted connections that are analogous to biological synapses.

2.1.1. Analogy to the brain

The exact workings of the human brain are still a mystery. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell known as a neuron. It is assumed that these cells are what provide us with our abilities to remember, think, and apply previous experiences to our every action. The average brain contains around 100 billion neurons connected to each other. Each neuron can connect with up to 200,000 other neurons. The power of the human mind comes from the sheer numbers of these neurons and the multiple connections between them [Camargo1990].

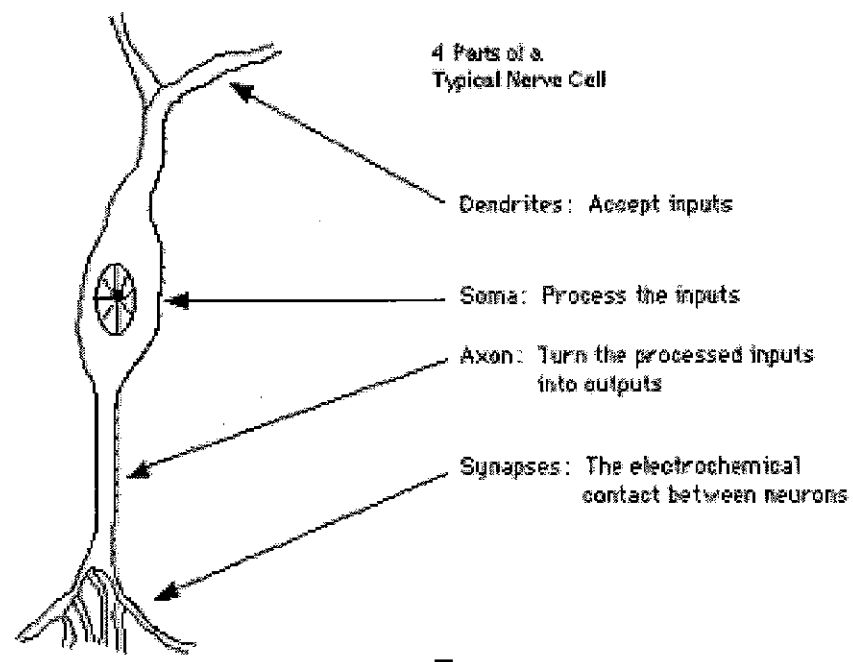


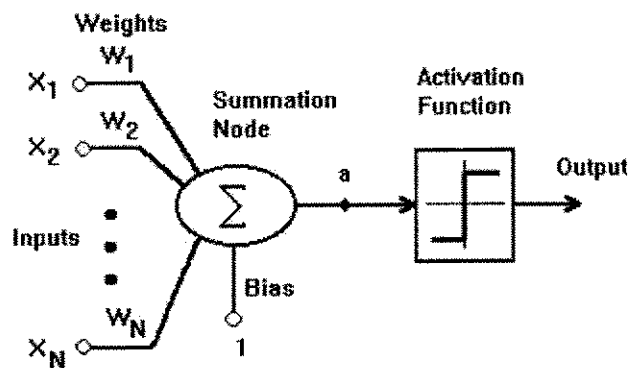
Figure 1. A typical biological neuron [Camargo1990].

A typical neuron is divided into four parts: the *dendrites*, the *soma* (cell body), a long fiber called the *axon*, and the *synapses* as shown in Figure 1. The dendrites act as input surface for signals from other neurons. These inputs are received through the synapses of other neurons. The soma then processes these incoming signals and sends the processed values to other neurons through the axon and synapses.

2.1.2. Artificial neurons and how they work

A more technical investigation of a single neuron shows that it can have an input vector X of N dimensions. These inputs go through a vector W of weights of N dimensions, as shown in Figure 2.

Processed by the summation node, " a " is generated where " a " is the "dot product" of vectors X and W plus a bias. " a " is then processed through an activation or transfer function which compares the value of " a " to a predefined threshold. If " a " is below the threshold, the neuron will not fire. If it is above the threshold, the neuron will fire one pulse whose amplitude is predefined.



$$a = W_1 X_1 + W_2 X_2 + \dots + W_N X_N + \text{Bias}$$

$$\text{output} = \text{Threshold}[a]$$

$$\text{where } \text{Threshold}[a] = \begin{cases} -1, & \text{for all } a \leq 0 \\ 1, & \text{for all } a > 0 \end{cases}$$

Figure 2. A basic artificial neuron.

2.1.3. Comparison of traditional computing and ANNs

Neural networks offer a different way to analyze data, and to recognize patterns within that data, than traditional computing methods. However, they are not a solution for all computing problems. Traditional computing methods work well for problems that can be well characterized, like balancing checkbooks and keeping ledgers. Table 1 identifies the basic differences between the two computing approaches.

Characteristics	Traditional Computing (Including Expert Systems)	Artificial Neural Networks
Processing style	Sequential	Parallel
Functions	Via rules; Concepts; Calculations	Via images; Pictures; Controls
Learning Method	By rules	By example
Applications	Accounting; Word processing; Math inventory; Digital communications	Sensor processing; Speech recognition; Pattern recognition; Text recognition

Table 1. Comparison of traditional computing and ANNs.

2.1.4. Comparison of expert systems and ANNs

Typically, an expert system consists of two parts, an inference engine and a knowledge base. The inference engine is generic. It handles the user interface, external files, program access, and scheduling. The knowledge base contains the information that is specific to a particular problem. This knowledge base allows an expert to define the rules, which govern a process.

Efforts to make expert systems general have run into a number of problems. As the complexity of the system increases, the system simply demands too much computing

resources and becomes too slow. Expert systems have been found to be feasible only when narrowly confined.

ANNs offer a completely different approach to problem solving. They try to provide a tool that both programs itself and learns on its own. Neural networks are structured to provide the capability to solve problems without the benefits of an expert and without the need for programming. They can seek patterns in data that no one knows are there.

A comparison of artificial intelligence's expert systems and neural networks is presented in Table 2.

Characteristics	Von Neumann Architecture Used for Expert Systems	Artificial Neural Networks
Processors	VLSI (traditional processors)	Artificial Neural Networks; variety of technologies; hardware development is on going
Processing approach	Processes problem rule at a one time; sequential	Multiple, simultaneously
Connections	Externally programmable	Dynamically self programming
Self learning	Only algorithmic parameters modified	Continuously adaptable
Fault tolerance	None without special processors	Significant in the very nature of the interconnected neurons
Neurobiology in design	None	Moderate
Programming	Through a rule based complicated	Self-programming; but network must be set up properly
Ability to be fast	Requires big processors	Requires multiple custom-built chips

Table 2. Comparison of ES and ANNs.

2.2. History of ANNs

The first step towards ANNs came in 1943 when Warren McCulloch, a neuro-physiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. They modeled a simple neural network with electrical circuits. [Anderson1997]

As computers started to get advanced in the 1950s, it became possible to begin to model the theories concerning human thought. Nathaniel Rochester from the IBM research laboratories led the first effort to simulate a neural network. That first attempt failed, but later attempts were successful. [Anderson1997]

In 1956 the Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks. One of the outcomes of this process was to stimulate research in both the intelligent side and in the much lower level neural processing part of the brain. [Anderson1997]

In the years following the Dartmouth Project, John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes. Also, Frank Rosenblatt, a neuro-biologist of Cornell, began work on the Perceptron. The Perceptron, which resulted from this research, was built in hardware and is the oldest neural network still in use today. A single-layer perceptron was found to be useful in classifying a continuous-valued set of inputs into one of two classes. The perceptron computes a weighted sum of the inputs, subtracts a threshold, and passes one of two possible values out as the result. [Anderson1997]

In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models they called ADALINE and MADALINE. These models were named for their use of Multiple

ADaptive LINEar Elements. MADALINE was the first neural network to be applied to a real world problem. It is an adaptive filter, which eliminates echoes on phone lines. This neural network is still in commercial use. [Anderson1997]

In 1982 several events caused a renewed interest. John Hopfield of Caltech presented a paper to the national Academy of Sciences. Hopfield's approach was not to simply model brains but to create useful devices. With clarity and mathematical analysis, he showed how such networks could work and what they could do. [Anderson1997]

By 1985 the American Institute of Physics began what has become an annual meeting - Neural Networks for Computing. By 1987, the Institute of Electrical and Electronic Engineer's (IEEE) First International Conference on Neural Networks drew more than 1,800 attendees. [Anderson1997]

Today, neural networks discussions are occurring everywhere. However, its future lies in hardware development. Currently most neural network development is simply proving that the principle works. Complex neural networks, due to processing limitations, take weeks to learn. As a result, companies are now focusing on developing specialized chips to implement these networks.

2.3. Major components of an ANN and how they work

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

Component 1. Weighting Factors: A neuron usually receives many simultaneous inputs. Each input has its own relative weight, which gives the input the impact that it needs on the PE's summation function. Some inputs are made more important than others so that they have a greater effect on the PE as they combine to produce a neural response.

Weights are adaptive coefficients within the network that determine the intensity of the input signal as registered by the artificial neuron. They are a measure of an input's connection strength. These strengths can be modified in response to various training sets and according to a network's specific topology or through its learning rules.

Component 2. Summation Function: The first step in a PE's operation is to compute the weighted sum of all of the inputs. This simple summation function is found by multiplying each component of the input vector by the corresponding component of the weight vector and then adding up all the products, as shown in figure 2. The result is a single number, not a multi-element vector.

The summation function can be more complex than just the simple input and weight sum of products. The input and weighting coefficients can be combined in many different ways before passing on to the transfer function. In addition to a simple product summing, the summation function can select the minimum, maximum, majority, product, or several normalizing algorithms. The specific algorithm for combining neural inputs is determined by the chosen network architecture.

Component 3. Transfer Function: The result of the summation function is transformed to a working output through an algorithmic process known as the transfer function. In the transfer function the summation total can be compared with some threshold to determine the neural output. If the sum is greater than the threshold value, the PE

generates a signal. If the sum of the input and weight products is less than the threshold, no signal (or some inhibitory signal) is generated. Both types of response are significant.

The threshold, or transfer function, is generally non-linear. Linear (straight-line) functions are limited because the output is simply proportional to the input. Linear functions are not very useful. The transfer functions that are most commonly used are of the sigmoidal type. The most important characteristic of sigmoid functions is symmetry relative to the origin. Functions like the hyperbolic tangent and the arctangent are symmetric relative to the origin, while the logistic function, for example, is symmetric relative to a point of coordinates (0, 0.5). Symmetry relative to the origin gives sigmoids a bipolar character that normally tends to yield conditioned error surfaces. Sigmoids like the logistic tend to originate narrow valleys in the error function, which slows the speed of the training process.

Component 4. Output Function (Competition): Each PE is allowed one output signal, which it may output to hundreds of other neurons. Normally, the output is directly equivalent to the transfer function's result. Some network topologies, however, modify the transfer result to incorporate competition among neighboring PEs. Neurons are allowed to compete with each other, inhibiting PEs with weak strength.

Component 5. Error Function: In most learning networks the difference between the current output and the desired output is calculated. This raw error is then transformed by the error function to match a particular network architecture. The most basic architectures use this error directly, but some square the error while retaining its sign, some cube the error, other paradigms modify the raw error to fit their specific purposes. The artificial neuron's error is then typically propagated into the learning function of another PE.

Component 6. Learning Function: The purpose of the learning function is to modify the variable connection weights on the inputs of each PE according to some neural based algorithm, in order to minimize the error value.

2.4. ANN's learning methods

There are two types of learning: supervised and unsupervised. In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights that control the network. This process occurs over and over as the weights are continually refined. During the training of a network the same set of data is processed many times as the connection weights are ever refined.

In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaptation. Currently, unsupervised training is not well understood and is still the subject of research.

2.5. Multi-layer perceptrons and generalized feedforward networks

Multi-layer perceptrons (MLPs) are the best known and most widely used kind of neural network [Almeida1997]. They are formed by PEs of the type shown in figure 2. Most often, PE's are interconnected in a *feedforward manner*, i.e., with interconnections

that do not form any loops. In feedforward networks, PEs are usually arranged in layers. The typical network has an input layer, an output layer, and at least one hidden layer, as shown in Figure 3. There is no theoretical limit on the number of hidden layers. Each layer should be fully connected to the succeeding layer. The number of layers and the number of PEs per layer are important parameters to be determined. The main advantage of MLPs is that they are easy to use, and that they can approximate any input/output map. The key disadvantages are that they train slowly, and require lots of training data (typically three times more training samples than network weights).

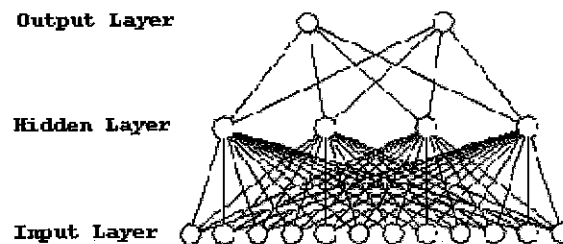


Figure 3. A MLP.

Generalized feedforward networks are a generalization of the MLP such that connections can jump over one or more layers. In theory, a MLP can solve any problem that a generalized feedforward network can solve. In practice, however, generalized feedforward networks often solve the problem much more efficiently.

Chapter 3 Literature Review

3.1. History of optical character segmentation and recognition

The history of handwriting recognition systems is not complete without mentioning the optical character recognition (OCR) systems that preceded them. OCR is a problem recognized as being as old as the computer itself. There have been many papers and technical reports published reviewing the history of OCR technologies. Modern OCR was said to have begun in 1951 due to an invention by M. Sheppard called GISMO, a robot reader-writer. In 1954, a prototype machine developed by J. Rainbow was used to read uppercase typewritten letters at very slow speeds. By 1967, companies such as IBM finally marketed OCR systems. However in the late 60's, these systems were still very expensive, and therefore could only be used by large companies and government agencies. Today, OCR systems are less expensive and can recognize more fonts than ever before [Kulkarni].

A number of systems have been developed to recognize Arabic text. One of these is TextPert 3.7 Arabic, produced by CTA Inc., which runs on the Macintosh Arabic system. Another is Al-Qari'al-Ali, a version of the program known as MULTREC, produced by Alamiah Software Co. Both of these programs were able to recognize certain computer printed texts of good quality with a reasonable degree of accuracy considering the difficulties of the Arabic text [Zemanec1994].

Another system designed by Fehri and Ben Ahmed used a hybrid of Radial Basis Function Networks and Hidden Markov Models to recognize printed Arabic text after identifying the used font. The results showed an increase in the recognition rate when the font is known prior to segmentation process [BenAhmad].

Finally, the best-known Arabic text recognition system ever developed is Sakhr OCR developed by Sakhr. The system uses an artificial neural network with a segmentation accuracy of 98% and a recognition accuracy of 99.8% for printed text.

Research now focuses on hand-printed numeral, character and cursive handwriting recognition. Unfortunately, the success of OCR could not carry on to handwriting recognition, due to the variability in people's handwriting.

3.2. Comparison of segmentation and recognition techniques

A lot of techniques exist in the literature that deals with developing accurate segmentation, preprocessing and feature extraction techniques.

3.2.1. Segmentation techniques

For the first problem, the diverse styles and sizes of handwriting both play a large factor in the failure of current techniques. In some cases even a human being would not be able to segment handwriting containing characters which are tightly packed together and illegible. Segmentation systems also have to deal with the variability of handwriting from

one person to another. Researchers have used different approaches to handle the segmentation problem.

Approach 1. The first is a global approach, which involves the recognition of the whole word by the use of identifying features. This means keeping a catalog of the features of all the words that the system can recognize, which is inflexible when recognizing handwritten text.

Approach 2. The second approach requires that a word be first segmented into letters. The letters are then recognized individually and can be used to match up against individual words. Segmentation methods follow two major approaches:

- 1) *Independent segmentation:* separate segmentation and recognition stages, with segmentation breaking up a connected block of text into parts and then recognizing the parts. In this approach, connected text can be segmented at different locations: character boundaries and potential segmentation points.
 - a. *Character boundaries.* Here, if the segmentation step fails to accurately isolate a character, then most certainly that character will be miss-recognized. This makes the segmentation step the most critical step in the recognition process. To work effectively, segmenting at character boundaries relies heavily on heuristic information about the character set, the dimensions of characters, and the writing process. This approach is very difficult for cursive text where the segmentation points between characters are ambiguous and the shapes of characters vary.
 - b. *Potential segmentation points.* This means segmenting a word into parts possibly smaller than a character. The usual recognition scheme here is to recognize the parts and then combine them into characters. The advantage

of segmenting into primitives and not characters is that it is easier to identify a set of potential segmentation points, which would include all the actual segmentation points, than to directly identify the actual points.

2) *Interleaved segmentation-recognition*: involves interleaved stages of segmentation and recognition. The segmentation stage suggests a set of possible segmentation points at which to segment, and then the recognition stage modifies the confidence in a connection point. This technique is particularly effective in dealing with the ambiguity of cursive text and can handle mistakes in segmentation by re-segmenting when recognition fails. Some of the different ways this technique can be implemented include:

- a. *Elastic matching*. Segment a word at all potential segmentation points. The resulting sequence of symbol parts is then matched to a database of stored symbols, which then produces the symbol (or set of symbols) that best matches the sequence of parts. This resembles approximate string matching that attempts to find the best match between two potentially differing strings of symbols, and is very similar to the techniques of speech recognition.
- b. *Recursive segmentation-recognition*. Segment a word into symbols, and then attempt to recognize the symbols. If the symbols cannot be recognized, then the procedure is recursively repeated by segmenting again.

The mentioned techniques have been implemented using heuristic methods, artificial neural network methods, mathematical morphology methods, and hybrid methods [Breuel1997] and [Al-Badr].

3.2.2. Preprocessing and feature extraction techniques

Research on preprocessing techniques deal with the choice of whether to convert raw handwriting into a more efficient form, i.e., whether to binarize the handwriting or keep it in gray-scale form. Researchers are disputing whether the handwriting should be skeletonized or should remain the way it is to preserve features.

Feature extraction poses the problem of choosing the right features to extract and the right technique to perform the task. For example, researchers may choose between extracting features such as the entire contours of characters or by extracting many features such as end-points, corner points, holes, and so on.

An important distinction is between topographic and non-topographic features. For topographic features, the components of the feature vectors correspond in some simple way to the two-dimensional locations in the input image. Non-topographic features are moment-based representations of the input image [Breuel1997].

Some systems attempt to learn feature extraction methods, starting only with a raw image, while others rely on sophisticated heuristic feature extraction methods. In the next chapter, a system is presented that relies on sophisticated heuristic methods to extract the required features for segmentation and classification.

Chapter 4 Neuro-Conventional Segmentation Method

4.1. Character Segmentation and Recognition Obstacles

There are several major problems with Arabic segmentation and recognition. They can be classified into 3 main categories: general difficulties, handwritten text specific difficulties, and Arabic text specific difficulties.

4.1.1. General difficulties

The following difficulties are common among character segmentation and recognition methods in general:

Problem 1. Presence of lines and other non-character objects.

Problem 2. Presence of noise or ‘salt-and-pepper’ in the scanned image.

Problem 3. Linguistic problems, i.e., if a dictionary is used as a spelling checker to improve the accuracy of the recognition process, then proper names, acronyms, or other words that are not likely to be in the lexicon will decrease recognition accuracy.

4.1.2. Handwritten text specific difficulties

The following difficulties are specific to handwritten text segmentation and recognition methods:

Problem 1. Variety in character size, i.e., characters may be written in many different sizes without changing their meaning.

Problem 2. Variety in writing instrument, i.e., characters may vary in line thickness, color, and/or stroke quality. Thick stroke causes the following problems:

- a. Touching characters.
- b. Holes in letters, e.g., ق or ف letters get filled up partially or completely.
- c. Thin stroke or low contrast may result in broken characters. Gaps in the stroke may also cause a lot of errors.

Problem 3. Different writers and the same writer under different conditions will slant their letters differently; that is, their handwriting undergoes a shear along the horizontal axis.

Problem 4. Translation problems, i.e., characters are not always written in the same position relative to the enclosing borders of the scanned image.

Problem 5. Presence of similar symbols, like 1 and .¹

Problem 6. Introduced and dropped spaces between characters make the task of word separation more difficult.

Problem 7. Handwritten text consists of elongated strokes making the separation of lines a difficult task when they overlap, as shown in Figure 4.

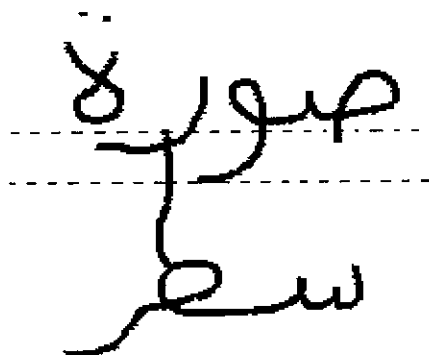


Figure 4. Two lines occupying a shared vertical space.

4.1.3. Arabic text specific difficulties

The following difficulties are specific to Arabic text segmentation and recognition:

Problem 1. Arabic is written cursively, i.e., more than one character can be written connected to each other, forming a block of characters (BC).

Problem 2. Arabic uses many types of external objects, such as dots, 'Hamza', 'Madda', and diacritic objects. A 'Hamza' is shown in Figure 5. These make the task of line separation and text segmentation into BCs and characters more difficult.



Figure 5. A Hamza external object.

Problem 3. Arabic characters can have more than one shape according to their position inside the BC: beginning, middle, final, or standalone, as shown in Figure 6.



Figure 6. The shapes the character ح takes according to its position inside a word.

Problem 4. Different writers and the same writer under different conditions will write some Arabic characters in completely different ways, as shown in Figure 7.

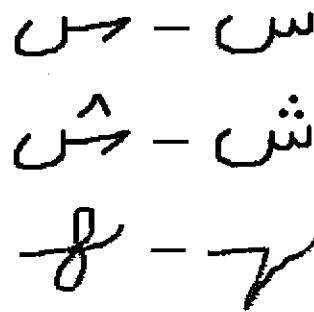


Figure 7. Three characters written in completely different ways.

Problem 5. The letter س also complicates segmentation and recognition when it occurs in the middle of a word as shown in Figure 8 .

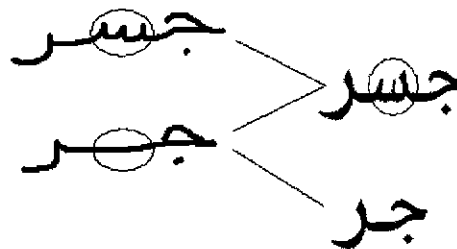


Figure 8. The letter س may be missed when appearing in the middle of a word.

Problem 6. Other characters have very similar contours and are difficult to segment and recognize especially when non-character and external objects are present in the scanned image. Figure 9 shows a list of such characters.

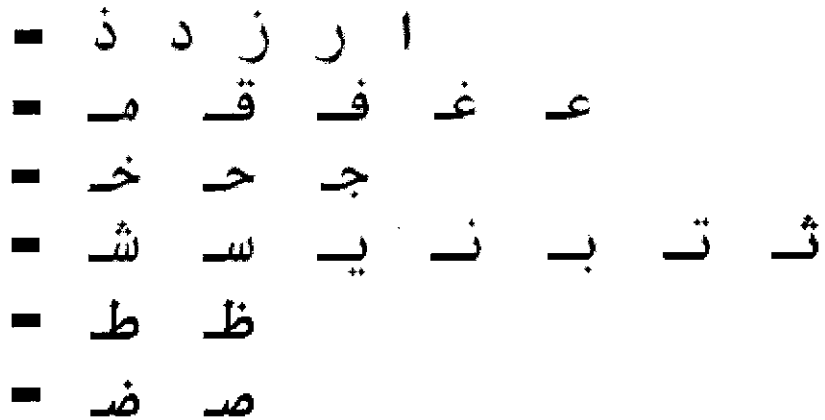


Figure 9. Characters with similar contours.

Problem 7. Characters that do not touch each other but occupy a shared horizontal space increase the difficulty of BC segmentation, as illustrated in Figure 10.

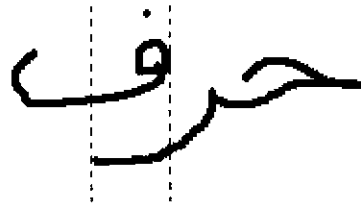


Figure 10. Two characters occupying a shared horizontal space.

Problem 8. Arabic uses many ligatures especially in handwritten text. Ligatures are connected characters that occupy a shared area, as shown in Figure

11. Segmenting ligatures into their constituent characters is very difficult because in most cases no vertical or horizontal line exists to segment a ligature.

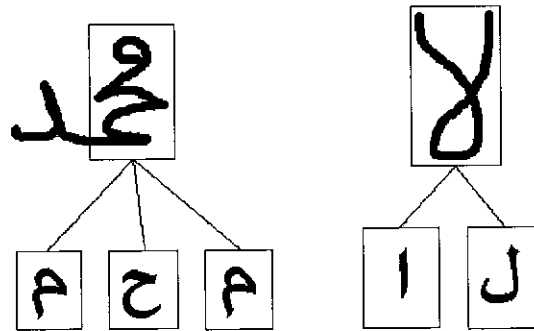


Figure 11. Arabic ligatures and their constituent characters.

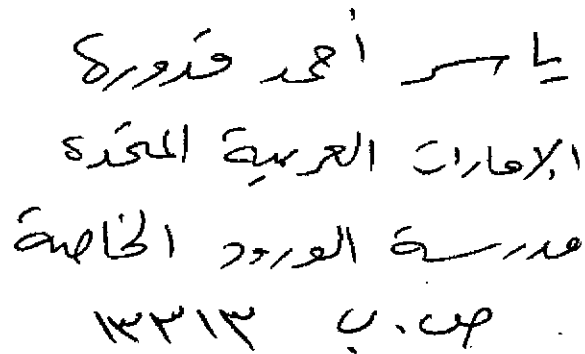
4.2. Proposed Technique

There are a number of steps that need to be taken before handwritten text can be recognized by a computer. These include sample data collection and analysis, scanning, binarization, and segmentation.

4.2.1. The data set

Since there is no standard benchmark database for Arabic handwritten text, samples were acquired randomly from various students and faculty members around the university. They were asked to write down their own mailing address on A4 sized paper. These addresses were then scanned using an Agfa SnapScan 1212p scanner at 150 pixels per inch and saved in monochrome Windows Bitmap (BMP) format. The images had different sizes ranging from 260 x 140 pixels to 1200 x 400 pixels. 360 addresses have been

collected consisting of about 4000 words or 9000 BCs. A sample address is shown in Figure 12.



طاهر أحمد قنديل
الإمارات العربية المتحدة
مدينة الورود الخاصة
ص.ب. ١٣٣١٢

Figure 12. A sample handwritten address.

4.2.2. Binarization

After the images were acquired, they were converted into monochrome bitmap form. Before any segmentation or processing could take place, it was then necessary to convert the images into binary representations. A heuristic algorithm generated a matrix of ones (1's) and zeros (0's) for each image. Each black pixel was represented with a 1 and each white pixel with a 0. In this form, segmentation and preprocessing could take place more easily.

4.2.3. Character block extraction

BC extraction is the first step of the segmentation phase. In order to solve Problem 7 mentioned in section 4.1.2 and Problem 7 mentioned in section 4.1.3, it was necessary to recursively extract connected BCs. Furthermore, dots, diacritics, and other external objects

are used heavily in Arabic. This makes the task of linking these external objects to main character objects, to create BCs, a very difficult process. A heuristic algorithm was implemented with a 94% accuracy, scanned the whole binary matrix of the image and performed the following steps:

Step 1. Before any processing could occur, invalid isolated black pixels, or ‘*pepper*’ were removed. A black pixel was identified as pepper and discarded if it had a maximum of one black neighbor pixel.

Step 2. Recursively, identify each group of connected black pixels as an object.

Step 3. 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.

Step 4. For each child object, determine the distance to all parents.

Step 5. 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. This formula was found by experimentation and yielded the best results.

Step 6. Merge all children to all parents who are at a maximum distance of R . This will lead to child objects being duplicated and merged to more than one parent. This is done to solve the problem of children that are located near objects other than their parent ones.

Higher accuracy couldn’t be achieved in BC extraction because of:

- Duplicating child objects in the last step;
- Child objects whose parent was located at a distance more than R .

4.2.4. 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]. An important feature of this algorithm is that it preserves the connectivity feature of BCs. Before describing the algorithm, it is necessary to define four properties of black pixels.

4.2.4.1. Black pixels properties

A black pixel is said to be *eight-simple* if changing it from black to white (one to zero) does not alter the connectivity property of the remaining black pixels of the BC, as shown in Figure 13. In other words, it does not disconnect the BC if it was discarded. The most difficult task of the thinning algorithm is the decision of whether a point has this property or not.

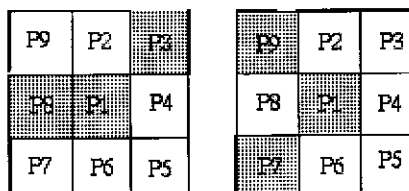


Figure 13. Two examples where $p1$ is not an 8-simple point.

A black pixel is said to be *eight-isolated* if it has no neighboring black pixels, as shown in Figure 14.

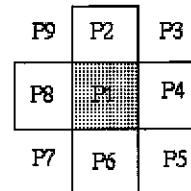


Figure 14. An example where $p1$ is an eight-isolated point.

An *eight-endpoint* is a black pixel that has exactly one black neighbor pixel, as shown in Figure 15. Eight-endpoints should not be discarded because if we went on deleting endpoints the whole BC would shrink to only one pixel.

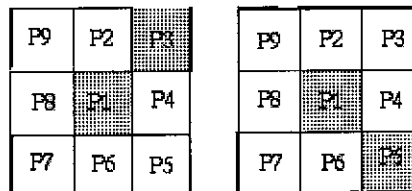


Figure 15. Two examples where $p1$ is an eight-endpoint.

A black pixel is said to be an *east border point* if its east neighbor is a white pixel. Similarly, west, north, and south border points can be defined. Four examples of border points are shown in Figure 16.

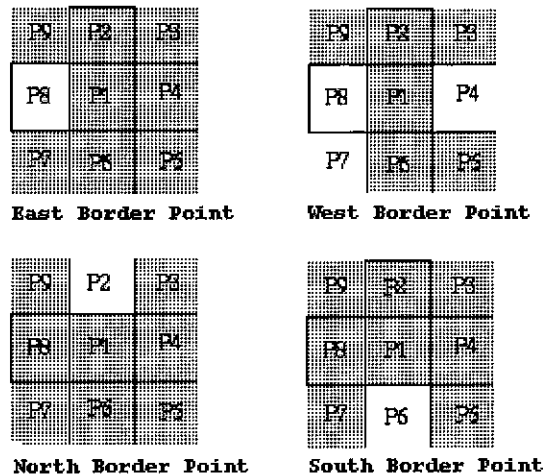


Figure 16. Four examples where $p1$ is a border point.

4.2.4.2. Skeletonization algorithm

A thinning iteration is divided into four sub-iterations. Thinning iterations should be repeated until they discard no more black pixels. Each step in these iterations is a thinning sub-iteration in one direction. Every sub-iteration deletes only points of one direction type; the others remain, irrespective of their simplicity. This way we can make sure that pixels are removed symmetrically from the borders. Every thinning iteration removes a layer until finally only the skeleton remains. The four sub-iterations are as follows:

Sub-iteration 1. Discard all east border points that are eight-simple but are neither eight-isolated nor eight-endpoint.

Sub-iteration 2. Discard all west border points that are eight-simple but are neither eight-isolated nor eight-endpoint.

Sub-iteration 3. Discard all north border points that are eight-simple but are neither eight-isolated nor eight-endpoint.

Sub-iteration 4. Discard all south border points that are eight-simple but are neither eight-isolated nor eight-endpoint.

Note that the order of the sub-iterations is important: east, west, north, and then south. This order produced the best results. The above algorithm is a sequential one. Points are examined one by one and discarded if possible. A parallel approach could greatly enhance the performance of the skeletonization phase. A BC skeletonized by this algorithm is illustrated in Figure 17.

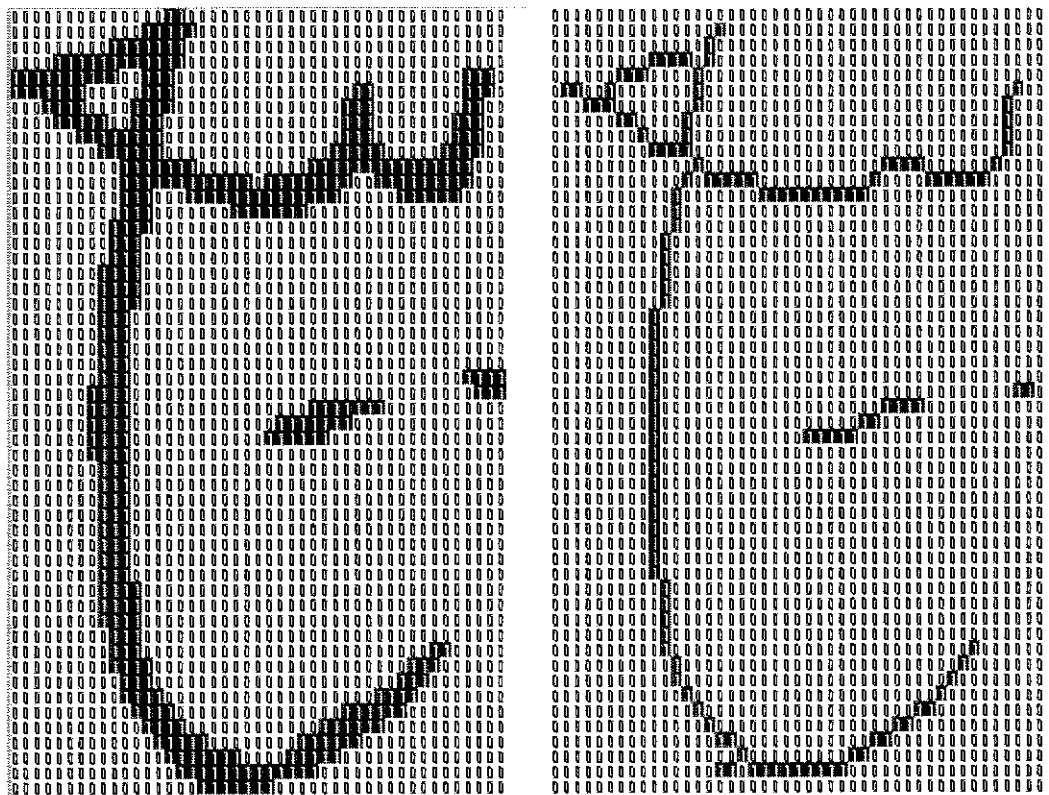


Figure 17. An example of a skeletonized BC.

4.2.5. Feature extraction

Scanned images often contain too much non-essential information, so a process called feature extraction is used to get information suitable for use in segmentation and recognition. Feature extraction lowers the number of features in order to lower the dimensionality of the input and thus the computational complexity of the system.

4.2.5.1. Feature set design

We have to look for features which are invariant and which capture the characteristics of handwritten text by filtering out all the attributes that make the same character assume different appearances. There are two main choices:

- Topographic or geometric features.
- Non-Topographic or moment-based features.

Topographic features are a common and proven choice for handwritten character recognition. They can also be divided into *point* features and *area* features. Point features occur at a well-defined point in a two-dimensional space, e.g., the endpoint of a line. Area features occur over a finite two-dimensional area, e.g., a hole caused by the circular movement of a pen.

Each feature is represented by a number of attributes. These attributes quantify the nature of the feature, by, for example, specifying its position or size. An attribute may be represented as a continuous value, a discrete value, or a binary value.

The features that are extracted from the raw data image are one aspect of its representation. A second aspect is the relationships that exist between features. In spatial terms we can think of simple binary relationships such as ‘above’ or ‘close-to’ that describe the relative position of two features. This information is also important in segmentation and recognition systems.

4.2.5.2.Extraction method

Some systems attempt to learn feature extraction methods, starting only with a raw image, while others rely on sophisticated hand-coded feature extraction methods. Two important examples of learning feature extraction methods applied to handwritten character segmentation and recognition are the Karhunen-Loeve transformation [Grother1992], and the topographic feature maps obtained through weight sharing in the system described in [Burges1993] and [LeCun1989].

However, for MLP-based systems, hand-coded heuristic methods for extracting features such as endpoints, holes and corner points are a common and proven choice.

Holes are islands of white pixels completely surrounded by black pixels. A hole is shown in Figure 18. Holes are the most difficult feature to extract. A recursive method was used which scanned all white pixels of an image and for each pixel searched for borders in all directions. If the pixel was completely surrounded by black pixels and/or image edges then all traversed white pixels were marked as hole pixels.

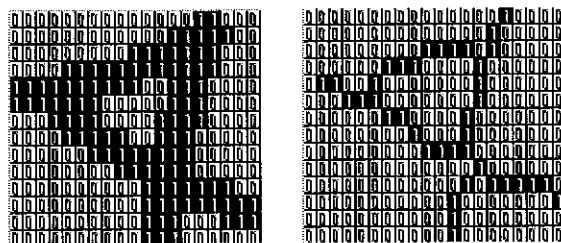


Figure 18. An example of a hole in a normal image (right) and a skeletonized image (left).

A corner point is a black or white skeleton point that has at least two connected branches on right angles to each other, as shown in Figure 19.

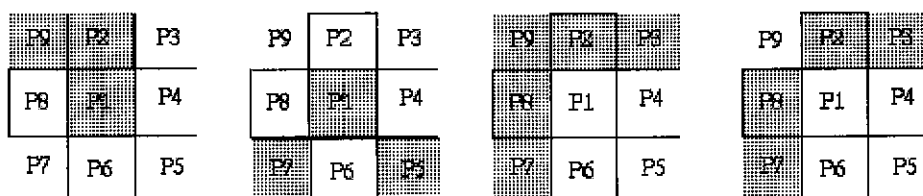


Figure 19. Four examples of corner points.

A fork point is a black skeleton point that has at least three connected branches as shown in Figure 20.

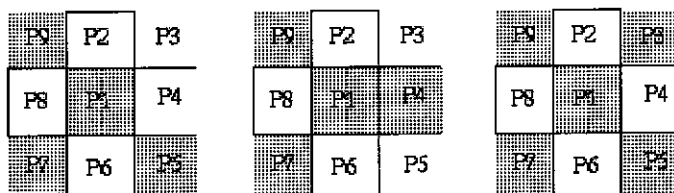


Figure 20. Three examples where $p1$ is a fork point.

In order to decide whether a column is a valid segmentation point or not, the features listed in Table 3 were extracted for each column of an image.

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

Table 3. Major features extracted for each column of BC.

Figure 21 shows an example of attribute values of features generated for a sample BC image and its skeletonized representation.

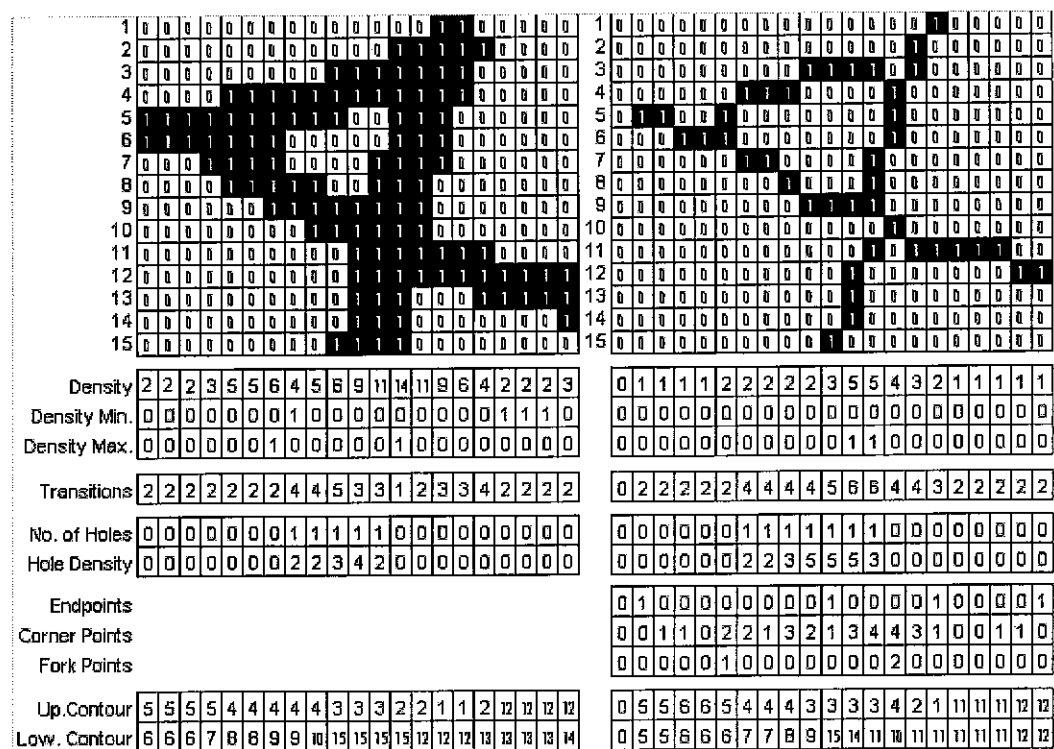


Figure 21. An example of extracted features.

4.2.6. Pre-segmentation point generation

The objective of this module is to over-segment all the BCs based on the features extracted in the feature extraction module. A simple heuristic segmentation algorithm was implemented which first scans the BC looking for minimas and maximas in black pixel densities, holes, endpoints, corner points, fork points, and upper and lower contours common in handwritten cursive text. Based on these features, the algorithm decides on pre-segmentation points.

Finally, the algorithm performs a final check to see if one pre-segmentation point was not too close to another by comparing the distance between these points with the average character width. The average character width was determined from the average BC height. Assuming that the width of a character is, in most cases, less than its height, an approximation of character width was estimated as a percentage of the average BC height.

The final result is a set of pre-segmentation points that are going to be validated by the ANN.

4.2.7. Pre-segmentation point validation by ANN

The objective of the segmentation ANN is to validate the accuracy of the pre-segmentation points generated by the heuristic algorithm presented in the previous section. To train the ANN with both accurate and erroneous segmentation points, the output from the heuristic segmentation algorithm was used. It was necessary to manually separate the pre-segmentation points into valid and invalid points and save them to a file together with the extracted set of features and desired output for each point.

4.2.7.1. ANN architecture

A generalized feedforward neural network was used to validate the accuracy of the proposed segmentation points. This neural network is a generalization of the multi-layer perceptron (MLP) such that each layer feeds forward to all subsequent layers. In theory, a MLP can solve any problem that a generalized feedforward network can solve. In practice, however, generalized feedforward networks often solve problems much more efficiently [Almeida1997], [Verma].

4.2.7.1.1. ANN Size

An important criterion in ANN design is the network size. The number of PEs in a hidden layer is associated with the mapping ability of the network. The larger the number, the more powerful the network is. However, if one continues to increase the network size, there is a point where the generalization gets worse. This is due to the fact that we may be over-fitting the training set, so when the network works with patterns that it has never seen before the response is unpredictable.

There is no rule of thumb to determine good network architecture just from the number of inputs and outputs. It depends critically on the number of training cases, the amount of noise, and the complexity of the function or classification the network is supposed to learn. There are cases with one input and one output that require thousands of hidden units, and cases with a thousand inputs and a thousand outputs that require only one hidden unit, or none at all. To solve this issue many different networks with different number of hidden units were tried at first, starting with the smallest possible number of PEs. The generalization error for each one was estimated, and the network with the minimum estimated generalization error that learned best to identify correct segmentation points was chosen.

The best ANN architecture reached consisted of 52 inputs, 1 output, and 4 hidden layers. The 52 inputs were feature attributes of a pre-segmentation point and the output was the validity of the point. The ANN architecture is summarized in Table 4.

	PEs	Transfer function
Input Layer	52	Linear
Layer 1	41	Tanh
Layer 2	27	Tanh
Layer 3	20	Tanh
Layer 4	16	Tanh
Output Layer	1	Tanh

Table 4. Architecture of Segmentation ANN.

4.2.7.1.2. Transfer functions

Transfer functions are generally non-linear. The kind of non-linearity that is most commonly used has the general form shown in Figure 22. It has two horizontal asymptotes, and is uniformly increasing, with a single point where the curvature changes sign. Curves with this general shape are called *sigmoids*. Some of the most common expressions of sigmoids are:

$$S(s) = \frac{1}{1 + e^{-s}} = \frac{1 + \tanh(s/2)}{2}$$

Equation 1. Logistic function.

$$S(s) = \tanh(s)$$

Equation 2. Hyperbolic tangent function.

$$S(s) = \arctan(s)$$

Equation 3. Arctangent function.

The most important characteristic of sigmoid functions is symmetry relative to the origin. The hyperbolic tangent and the arctangent are symmetric relative to the origin, while the logistic function is symmetric relative to a point of coordinates (0, 0.5). Symmetry relative to the origin gives sigmoids a bipolar character that normally tends to yield conditioned error surfaces. Sigmoids like the logistic tend to originate narrow valleys in the error function, which slows the speed of the training process. In the segmentation ANN, the hyperbolic tangent function is used on all hidden PEs.

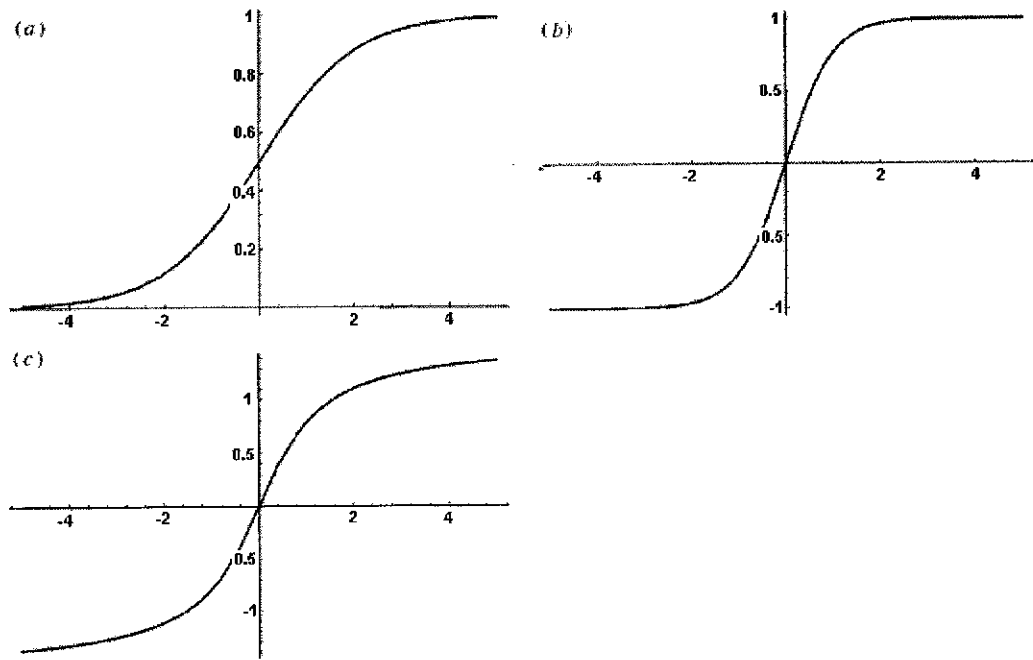


Figure 22. Sigmoids corresponding to: (a) Equation 1, (b) Equation 2 and (c) Equation 3.

4.2.7.1.3. ANN Implementation

The design of the segmentation ANN described was implemented using NeuroSolutions, version 3.022 by NeuroDimensions, Inc. The implemented ANN is shown in Figure 24.

Each of the axons shown represents a layer, or vector, of PEs. All axons are equipped with a summing junction at their input and a splitting node at their output. This allows axons to accumulate input from, and provide output to, an arbitrary number of components.

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. This functionality is illustrated in Figure 23.

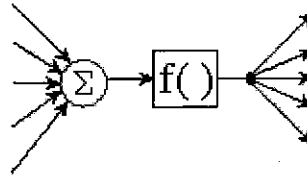


Figure 23. The mapping for the PE of the axon family.

Axons can receive input from, and provide output to both axons and synapses within the network. The full synapse shown in Figure 24 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. 15 full synapses have been used in the segmentation ANN and their architecture is shown in Table 5.

Input Axon	Output Axon	Inputs	Outputs	Weights
Input axon	Hidden axon 1	52	41	2132
	Hidden axon 2	52	27	1404
	Hidden axon 3	52	20	1040
	Hidden axon 4	52	16	832
	Output axon	52	1	52
Hidden axon 1	Hidden axon 2	41	27	1107
	Hidden axon 3	41	20	820
	Hidden axon 4	41	16	656
	Output axon	41	1	41
Hidden axon 2	Hidden axon 3	27	20	540
	Hidden axon 4	27	16	432
	Output axon	27	1	27
Hidden axon 3	Hidden axon 4	20	16	320
	Output axon	20	1	20
Hidden axon 4	Output axon	16	1	16

Table 5. The full synapses used in the segmentation ANN.

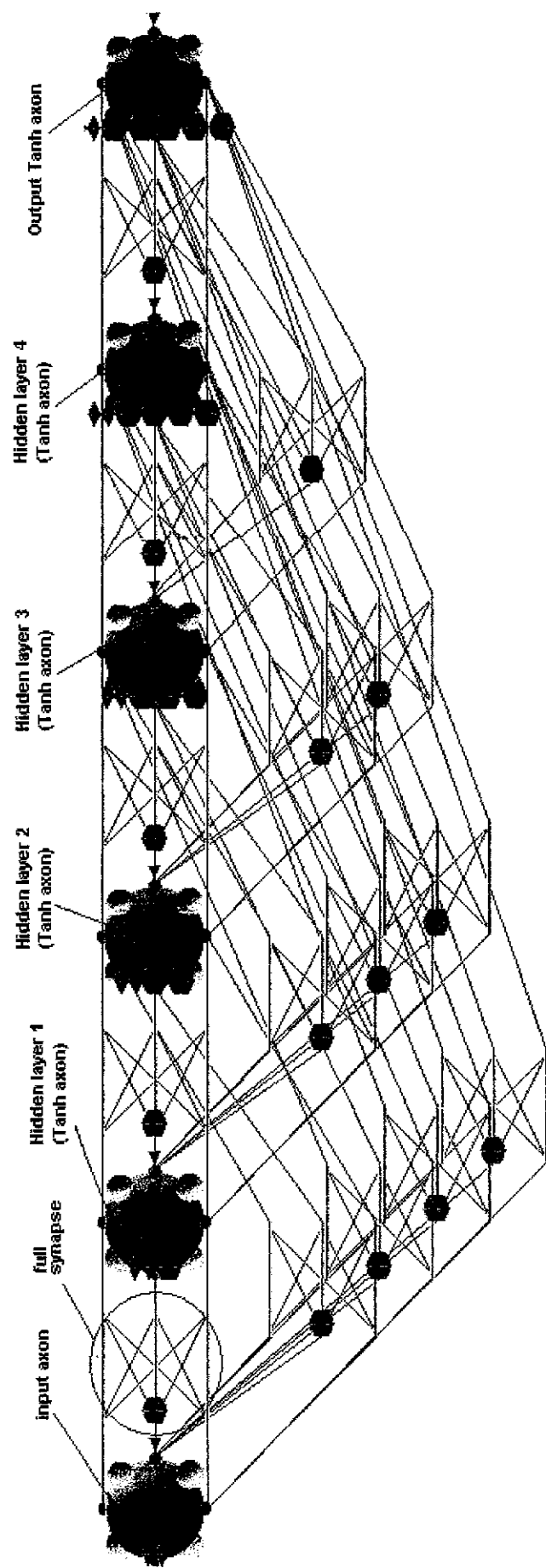


Figure 24 Segmentation ANN used to validate pre-segmentation points. (Source: NeuroSolutions Software)

A strict separation between training and test sets was carefully maintained. Even the repeated use of a test set only for evaluating different versions or revisions of a single network was avoided. This is because it can easily lead to an overestimation of the performance of a particular design.

4.2.7.2. ANN Training and Testing

Training is the process by which the free parameters of the network (i.e. the weights) get optimal values. It is in fact a search in the so-called *weight-space*, or performance surface. This is the space spanned by all weights in the network. The goal of the search is finding a point in this weight-space which minimizes a certain error criterion.

A simple performance surface is illustrated in Figure 26. 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. In general, the performance surface is contained within a $N+1$ dimensional space, where N is the number of weights in the network.

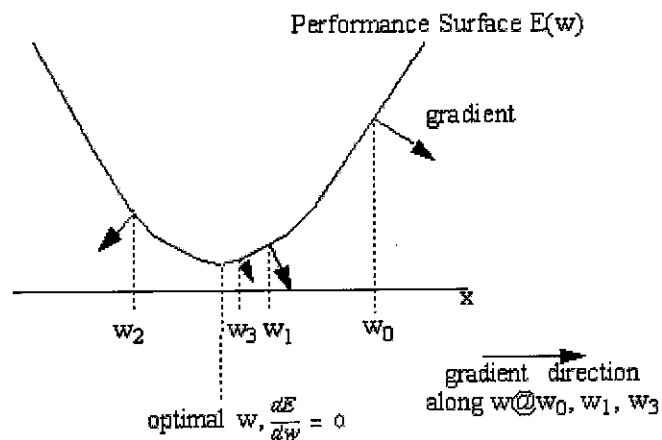


Figure 26. A simple performance surface.

4.2.7.2.1. Back-propagation learning method

The method used to train the segmentation ANN is the *back-propagation* method. Essentially, it is a three-step process:

- Step 1.** The input data is propagated forward through the network to compute the system output.
- Step 2.** The error between the desired and actual output is computed.
- Step 3.** This error is then propagated backward through the network, modifying weights on each layer until the first layer is reached.

Backpropagation modifies 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.

The advantage of using back-propagation is that it is simple and can be implemented easily. The disadvantages 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. Back-propagation 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.

NeuroSolutions solves a lot of these problems. It implements back-propagation of the error in a secondary "plane" that sits on top of the axons and synapses. This is called the back-propagation plane and is shown in Figure 27.

4.2.7.2.2. *Error Criteria*

Supervised learning requires a metric of how the network is doing. This metric is determined by calculating the sensitivity that a cost function has with respect to the network's output. This cost function, J , is normally positive, but should decrease towards zero as the network approaches the desired response. The literature has presented several cost functions, but the quadratic cost function, shown in Equation 4, is by far the most widely applied.

$$J(t) = \frac{1}{2} \sum_i f(d_i(t) - y_i(t))^2$$

Equation 4. Quadratic cost function.

The *L2Criterion* component is a square error criterion and it implements the quadratic cost function. The error reported is simply the squared Euclidean distance between the network's output and the desired response as shown in Equation 5, where $d(t)$ and $y(t)$ are the desired response and network's output, respectively.

$$e_i(t) = -(d_i(t) - y_i(t))$$

Equation 5. Error function.

The *L2Criterion* passes the computed error to the *Back Criteria* control. This control is designed to stack on top of the *L2Criterion*, and communicate the received error values from the *L2Criterion* with the back-propagation components to perform back-propagation.

The *Delta Threshold Transmitter* controls the communication of the Back Criteria component based on the amount of error change from one iteration to the next. The Back Criteria is allowed to transmit the error value when the change between successive iterations crosses a specified threshold, which is chosen to be 0.0001. This threshold value can also be specified to change (i.e., incremented, decremented, or scaled by a constant) each time it is crossed.

4.2.7.2.3. Back-Propagation Tanh Axon

The *Back Tanh Axon* is a layer of PE's with a transfer function that is derivative of the Tanh Axon. It attaches to a forward Tanh Axon for use in the back-propagation network. For each Tanh Axon a Back Tanh Axon is attached. Back-propagation requires all Back Axons perform two operations. First, given an error at their output, Back Axons must calculate gradient information for all adaptive weights within their dual components. Second, they must derive the relative error at their input to be back propagated to any components that precedes them.

4.2.7.2.4. Momentum learning rule

Momentum components update the weights of full synapses attached to them during the training process. They try to find a global minimum of a performance surface by taking steps in the direction estimated by the attached back-propagation component. 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. In addition, a step size that works well for one location in weight space may be unstable in another.

The Momentum component avoids these difficulties by providing a gradient descent with some inertia, so that it tends to move along a direction that is the average estimate for down. The amount of inertia is dictated by a momentum parameter. The higher the momentum, the more it smoothes the gradient estimate and the less effect a single change in the gradient has on the weight change. The major benefit of using momentum components is that they can avoid being trapped in local minima.

A large step size was selected at first while training the segmentation ANN. When the simulation diverged, the network was reset and training was started all over with a smaller step size. Starting with a large step size and decreasing it until the network became stable, found a value that solved the problem in few iterations. Small step sizes were used to fine-tune the convergence in the final stages of training.

The optimum values reached by experimentation for the step sizes and momentum parameters are shown in Table 7.

	Step Size	Momentum
Layer 1	1.0	0.7
Layer 2	0.1	0.7
Layer 3	0.01	0.7
Layer 4	0.001	0.7
Output Layer	0.0001	0.7

Table 7. Optimum learning parameter values reached in the segmentation ANN.

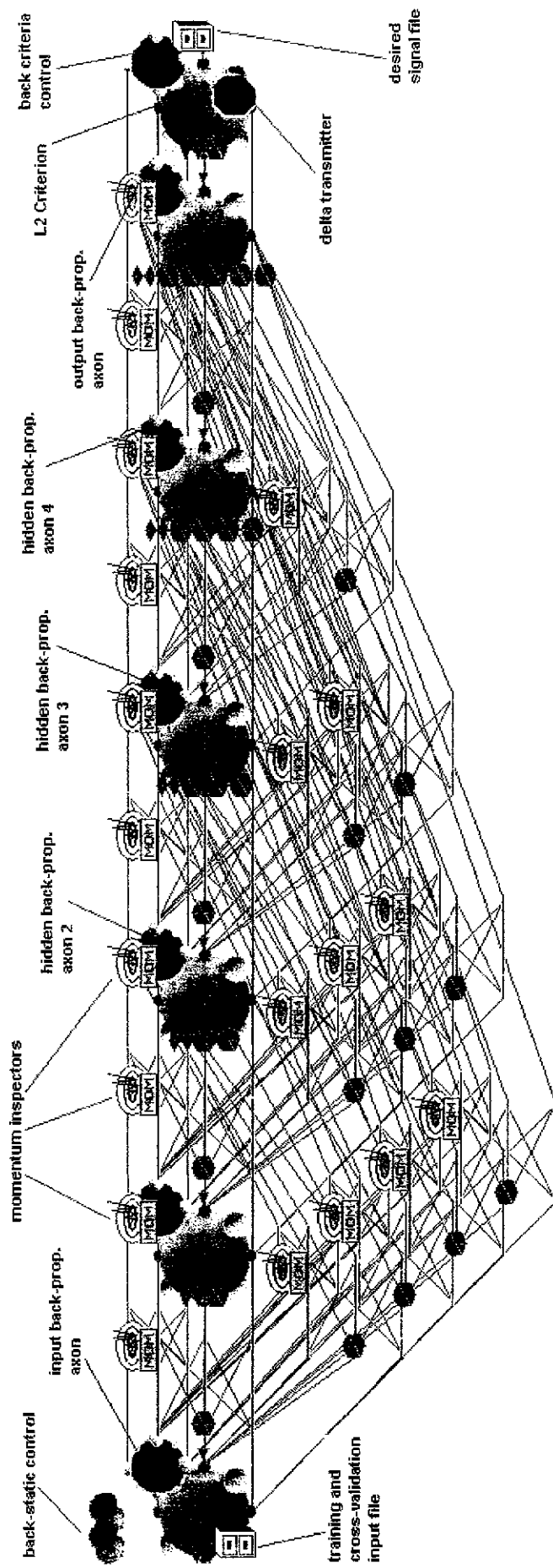


Figure 27 Segmentation ANN used to validate pre-segmentation points with the back-propagation layer added. (Source: NeuroSolutions Software)

4.2.7.2.5. Termination of training process

An important criterion is the termination of the training process. Cross-validation was used which is a highly recommended method for stopping network training. It monitored the mean square error (MSE) on an independent set of data and stopped training when this error began to increase. This is considered to be the point of best generalization, after which the network starts to memorize the input set. Figure 28 illustrates a sample learning curve on training and cross-validation sets. Learning was stopped when the MSE of the cross-validation started to increase.

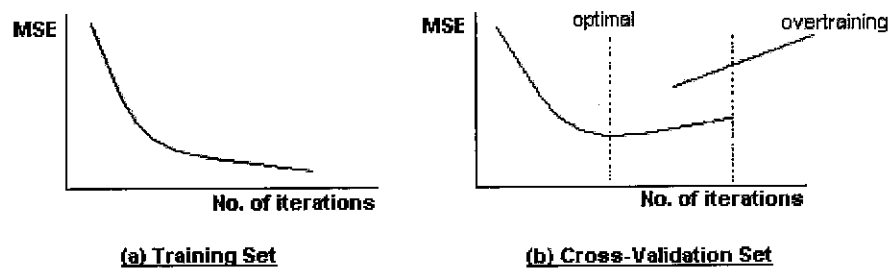


Figure 28. Behavior of MSE on training and cross-validation sets.

Other methods of terminating the training process include:

1. Limit the total number of iterations (hence the training time), stopping the training regardless of the networks performance.
2. Stop 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.

3. 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.

Chapter 5 Experimental Results

5.1. BC extraction results

The heuristic algorithm that is implemented for extracting BCs, described in section 4.2.3, achieved 94% accuracy. Higher accuracy could not be achieved due to the following main reasons: First, some BCs had external child objects that were located at a far distance from the parent BC. Therefore, the algorithm assigned them to BCs that they didn't belong to. Other external objects were located at an approximately the same distance to more than one parent BC. Therefore, they were duplicated and a copy was assigned to each one of those parent BCs.

5.2. Segmentation ANN results

The output range of the ANN 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 should be ignored.

A heuristic algorithm checked the results of the segmentation ANN on a test set of 10,000 exemplars. The algorithm defined the segmentation of a BC as correct when each known segmentation point was covered by an approved segmentation point by the ANN. Adequate coverage of a segmentation point is achieved when the distance from the known segmentation point to the closest approved point is less than 15% of the average character size. This is illustrated in Figure 29.

any point that lies between these 2 lines is considered to be a valid segmentation point (30% of average character width)

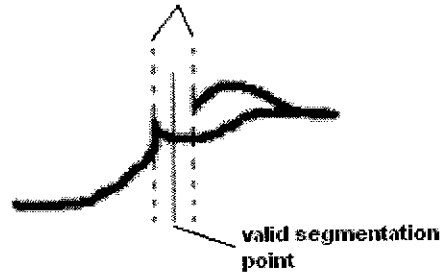


Figure 29. Range of a valid segmentation point.

5.2.1. Un-segmentable Objects

There were some objects that were impossible to segment in handwritten Arabic text. As illustrated in Table 8, every 100 BCs of the collected data, contain 10.16 un-segmentable ligatures and 13.02 characters with miss-located external objects. Figure 30 shows some examples of un-segmentable BCs because of points that are miss-located. In addition, 9.24 ش and س characters occur in every 100 BCs, which are almost always un-segmentable, as illustrated in Figure 8. Other miscellaneous un-segmentable BCs include characters like the letter ض, shown in Figure 30, which is always segmented into the letters ن. and م, or ع.

Un-segmentable objects	Occurrence in 100 BCs
Ligatures	10.16
Characters with miss-located points	13.02
Characters like ش and س	9.24

Table 8. Summary of un-segmentable objects.

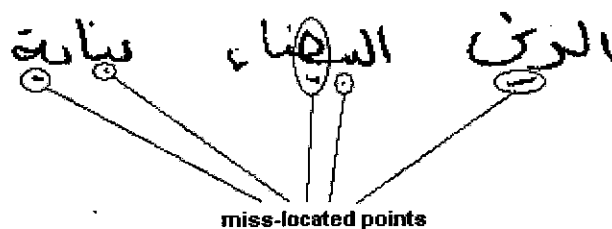


Figure 30. Three words containing miss-located external objects.

5.2.2. Experimental Results of different ANN architectures

Table 9 shows the results of the segmentation ANNs trained on the 48,000 training exemplars and tested on 10,000 exemplars. Many experiments were performed varying settings such as the network type, the number of hidden layers and the number of processing elements in each layer. For each experiment the number of inputs remained the same: 52 input features for each column. The eight ANN architectures that performed best are shown in Table 9.

ANN Architecture			MSE	Correct Points		Incorrect Points	
Network Type	No	Hidden Layers PEs		Invalid points	Valid points	Invalid points marked as valid	Valid points marked as invalid
Feedforward MLP	2	41-27	0.81	1354 (13.54%)		8646 (86.46%)	
Feedforward MLP	3	41-27-20	0.72	2684 (26.84%)		7316 (73.16%)	
Feedforward MLP	4	41-27-20-16	0.41	5311 (53.11%)		4689 (46.89%)	
				3767 (37.67%)	1544 (15.44%)	3326 (33.26%)	1363 (13.63%)
Feedforward MLP	5	41-27-20-16-13	0.56	4225 (42.25%)		5775 (57.75%)	
Feedforward MLP	6	41-27-20-16-13-11	0.50	4633 (46.33%)		5367 (53.67%)	
MLP	5	55-31-19-15-11	0.82	1332 (13.32%)		8668 (86.68%)	
MLP	7	55-31-19-15-13-11-9	0.59	3854 (38.54%)		6146 (61.46%)	
MLP	9	55-31-19-15-13-11-9-7-6	0.73	2336 (23.36%)		7664 (76.64%)	

Table 9. Segmentation ANN results using 48,000 training exemplars, tested on 10,000 exemplars

The highlighted ANN architecture in Table 9 performed best in identifying correct segmentation points and discarding incorrect ones. The minimum MSE achieved was 0.41. The ANN was able to identify the accuracy of 5,311 points out of the 10,000-point testing set. Of the correctly identified points, 3,767 were invalid segmentation points and 1,544 were valid segmentation points.

The ANN incorrectly identified 4,689 points. 3,326 of these points were invalid segmentation points marked as valid, and 1,363 were valid points marked as invalid. It should be noted that the majority of incorrectly identified points were invalid segmentation points marked as valid, which implies that the ANN over-segmented the handwritten BCs.

5.2.3. Applying threshold to ANN results

As shown in the previous section, the segmentation ANN performed badly in identifying invalid segmentation points. After studying the distribution of the ANN results over the range $[-0.9, +0.9]$, it was noted that the majority of these 3,326 points were found in the $[0, +0.5]$ range, as illustrated in Figure 31.

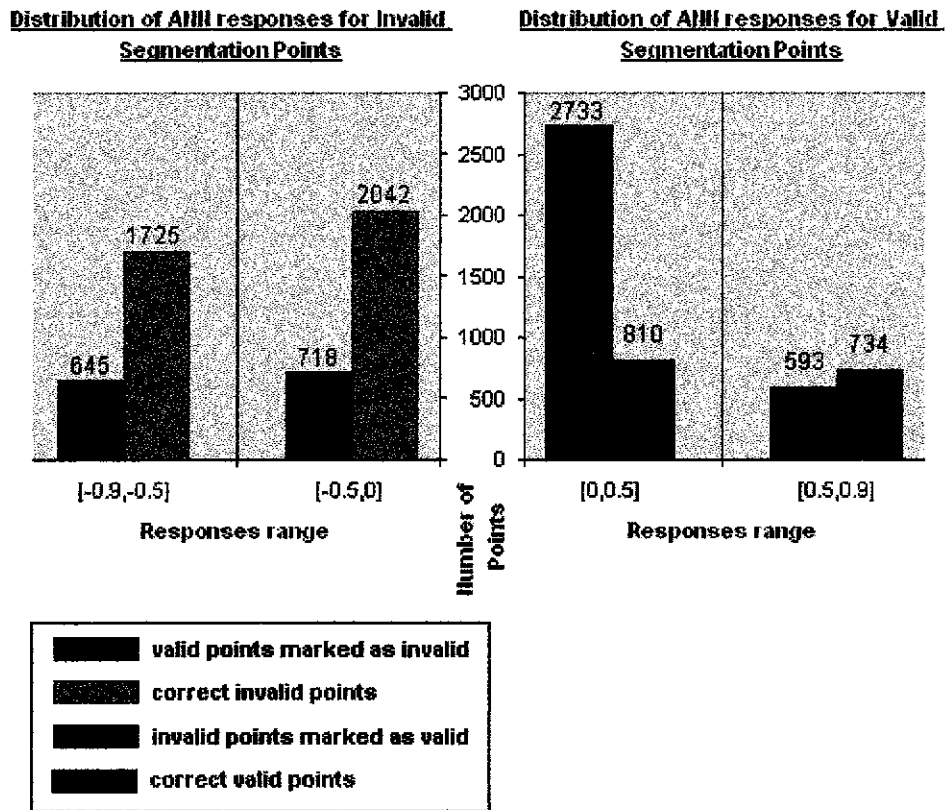


Figure 3) Distribution of ANN responses

To decrease the number of incorrectly identified segmentation points, a threshold value was applied to the ANN output. A module was implemented which checked the ANN responses against a 0.5 threshold. Responses between 0 and +0.5 were therefore rejected. It should be noted that these rejected patterns also include 810 correctly identified valid segmentation points. However, the number of incorrectly identified points, 2,733, in the (0,0.5) range is much greater than the correctly identified points.

Table 10 shows the results after rejecting these patterns.

Correct Points		Incorrect Points		Rejected Points	
Invalid points	Valid points	Invalid points marked as valid	Valid points marked as invalid	Invalid points marked as valid	Valid points
4501 (45.01%)		1956 (19.56%)		3543 (35.43%)	
3767 (37.67%)	734 (7.34%)	593 (5.93%)	1363 (13.63%)	2733 (27.33%)	810 (8.10%)

Table 10. ANN results after rejecting patterns with responses in the (0,0.5) range

As shown in Figure 31, rejecting patterns with responses in other ranges is not efficient because the number of correctly identified points is greater than the number of incorrectly identified points.

5.3. Comparison of segmentation results in the literature

Many researchers have used various techniques for the segmentation of characters in handwritten words. Segmentation accuracy rates of above 90% were achieved by Lee et al. [Lee1996], however the authors were only dealing with printed Latin alphanumeric characters. Srihari et al. [Srihari1993] obtained segmentation accuracies of 83% for handwritten zip codes (no alphanumerics). Han and Sethi [Han1995] 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 et al. [Eastwood1997], segmenting cursive handwriting produced a 75.9% accuracy rate using an ANN-based method. On average our segmentation accuracy using the neuro-conventional technique was just over 53%. Table 11 summarizes the results obtained by various researchers.

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 alphanumerics)	ANN-based method

Table 11. Comparison of segmentation results in the literature.

Chapter 6 Conclusion and Further Work

A heuristic segmentation technique used in conjunction with a generalized feedforward multi-layer neural network has been presented in this paper. It was used to segment difficult handwritten Arabic text, producing promising results. With some modifications more testing shall be conducted to allow the technique to be used as part of a larger system.

The segmentation program over-segmented the BCs it was presented with. This allowed the segmentation ANN to discard improper segmentation points and leave accurate ones. Overall the whole process was very successful, however some limitations still exist.

The segmentation phase proved to be successful in vertical segmentation of connected blocks of characters. However, in Arabic handwritten text, a lot of characters share the same horizontal space. A major limitation of the presented technique is that it couldn't segment horizontally these overlapping characters. Unfortunately, this will certainly affect any classifier algorithm.

Another problem is that there are a lot of handwritten characters that can be segmented and classified into two or more different classes depending on whether you look at them separately, or in a word, or even in a sentence. In other words, character segmentation and classification, especially handwritten Arabic characters, depends largely on contextual information, and not only on the topographic features extracted from these characters. Arabic handwriting recognition is a difficult problem and it is

not yet realistic to expect systems to achieve an acceptable accuracy in large vocabularies or where contextual information is of little use.

In future work, the segmentation technique will be improved in a number of ways. Firstly, the heuristic component of the segmentation system will need to be enhanced further. Originally, one of the main aims of the heuristic algorithm was to keep the number of incorrect segmentation points to a minimum, so that errors and processing time could be reduced. As a result, under-segmentation was noticeable in some BCs. Looking for more features or possibly enhancing the current feature extraction methods can solve this problem.

In the neural network component, more test patterns shall also be used in training and testing, and finally the technique shall be integrated into a complete handwritten recognition system.

Appendix A Preliminary Design for an OCR Neural Network

A-1. The Data Set

The data set used for recognition consisted of images segmented by the segmentation ANN described in this work. The images consisted of approximately 4,000 words or 18,000 segmented characters.

A-2. Feature Set Design

The topographic features that were used to classify character are presented in Table 12. A heuristic algorithm was implemented to extract the mentioned features for each image to be classified.

Feature	Attributes	Description	Attribute Type	Attribute Value
Width-height ratio		Image width divided by height	Continuous	$(0, \infty)$
Black pixel density	Black pixel density / (height * width)	Number of black pixels in the image divided by the total pixels in image.	Continuous	$[0, 1]$
	Average column density / height	Average column black pixel density divided by image height.	Continuous	$[0, 1]$
	Average row density / width	Average row black pixel density divided by width.	Continuous	$[0, 1]$
	Number of holes	Count the number of holes (or islands of white pixels completely surrounded by black pixels) in image	Discrete	$[0, 3]$
Holes	Total hole densities / (height * width)	Total number of hole pixels in image divided by the total pixels in image.	Continuous	$[0, 1]$
	Position of upper most, right most hole.	x-coordinate divided by width; y-coordinate divided by height.	Continuous	$[0, 1]$
	Number of endpoints in image	Number of endpoints in image.	Discrete	$[0, \text{height} * \text{width})$
	Number of corners in image	Number of corner points in image.	Discrete	$[0, \text{height} * \text{width})$
Fork points	Number of fork points in image	Number of fork points in image.	Discrete	$[0, \text{height} * \text{width})$
Upper and lower contours	Index of highest pixel in upper contour / height	Index of highest pixel in upper contour divided by height.	Continuous	$[0, 1]$
	Index of lowest pixel in lower contour / height	Index of lower pixel in lower contour divided by height.	Continuous	$[0, 1]$
Number of disconnected BCs		Number of disconnected BC objects in image.	Discrete	$[1, \infty)$

Table 12. Major features extracted for each image to be classified.

A-3. Classification ANN

The objective of the classification ANN is to classify segmented images into known characters. Classification involves assigning input patterns to one of a set of discrete classes. To train the ANN, the output from the segmentation module was used. It was necessary to manually classify images and save them to a file together with the extracted set of features and desired output for each point.

In mathematical terms, the classification ANN is a set of functions which map inputs x_i to outputs y_i , where the outputs specify which of the classes the input pattern belongs to. There are 52 classes presented in Table 13.

Classes	No. of elements
Characters	28
Arabic Digits	10
Latin Digits	10
Ligatures (٧)	1
Separators (",", ":", ";", "-", "'")	3
Total	52

Table 13. 52 classes of Arabic images.

A-3.1. ANN Size

A generalized feedforward neural network was used to classify images. This neural network is a generalization of the multi-layer perceptron (MLP) such that each layer feeds forward to all subsequent layers. The best ANN architecture reached consisted of 13 inputs, 52 outputs, and 4 hidden layers. The 13 inputs were feature attributes of an image, and the outputs were the validity of the point. The ANN architecture is summarized in Table 14.

	PEs	Transfer function
Input Layer	13	Linear
Layer 1	16	Tanh
Layer 2	20	Tanh
Layer 3	32	Tanh
Layer 4	41	Tanh
Output Layer	52	Tanh

Table 14. Architecture of Classification ANN.

The design of the classification ANN described was implemented using NeuroSolutions, version 3.022 by NeuroDimensions, Inc. The implemented ANN is shown in Figure 32.

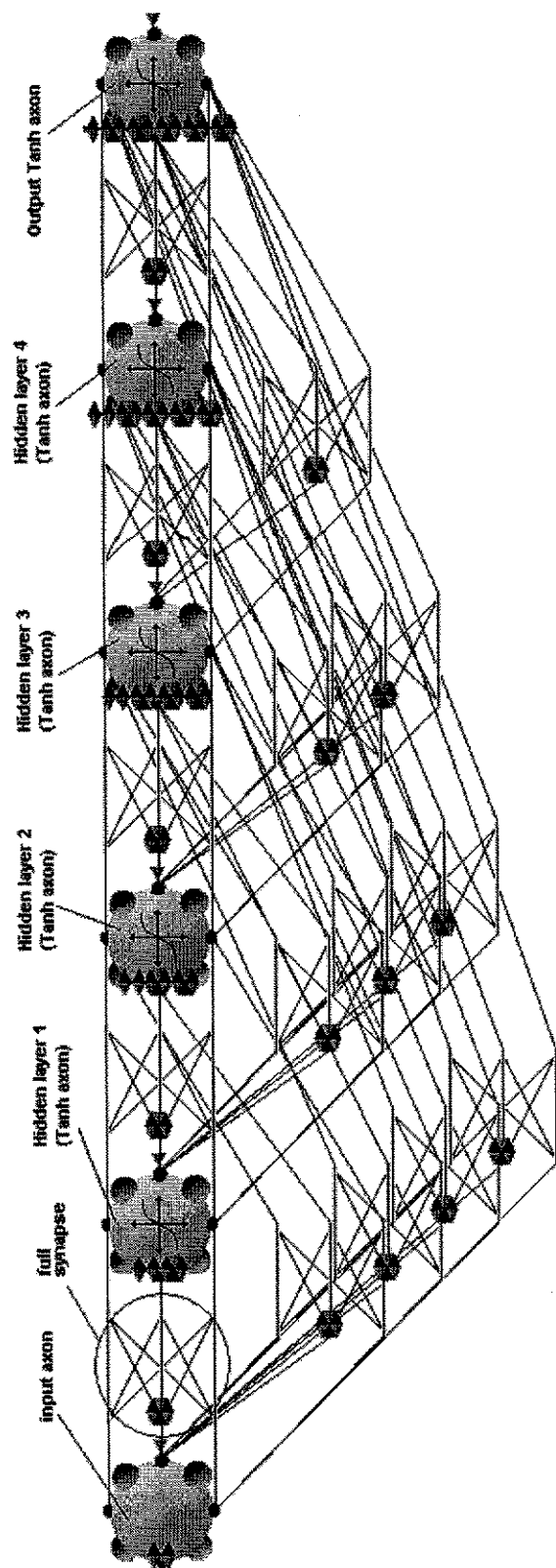


Figure 32. Architecture of classification ANN.

A-3.2. ANN Input Design

The segmented objects generated by the segmentation module were classified manually in order to produce input files for the training, cross-validation, and testing phases of the ANN. 18,000 segmented objects were evaluated manually and divided into three parts as shown in Table 15.

Input Set	Number of exemplars
Training set	10,000
Cross-validation set	4,000
Testing set	4,000

Table 15. Manually evaluated input sets.

Each object image is represented by a row in the input files. Each row starts with the id of the image, its desired class, and then the feature attributes are listed. A sample input file is shown in Figure 33.

```

002_00_000 0.90 8 39 0.23 0 0 0.00 0 0 2 0 0.00 0 0
002_00_001 -0.90 8 39 0.38 0 0 0.21 1 0 2 2 0.00 0 1
002_00_003 -0.90 (8 39 0.38 0 0 0.21 1 0 2 2 0.00 0 1)

```

Diagram labels for the third row:

- image id**: points to the circled text `002_00_003`.
- desired value**: points to the circled text `-0.90`.
- feature attribute values of column**: points to the circled list of values `(8 39 0.38 0 0 0.21 1 0 2 2 0.00 0 1)`.

Figure 33. Sample ANN input file.

A-3.3. Results

The ANN output consisted of 52 values, one for each class. The values range from -0.9 to $+0.9$. A positive value for a class indicates that the input pattern belonged to this class, and a negative value indicates that the input pattern does not belong to this class. The higher the positive value, the more confident the network is about the classification result. Therefore, the class with the highest positive value was selected as the correct class.

Unfortunately, for each input pattern, classes had very close positive values. Close analysis of results on the testing set showed that an average of 20 classes (40%) had positive values for each input pattern. Analysis also showed that the values of the positive results were in the $[0, +0.5]$ range, which made the determination of the correct class a very difficult problem. Selecting the highest positive value to indicate the correct class yielded a 12.4% accuracy for classification.

A-4. Conclusion and Future Work

In this appendix, a preliminary design and implementation of a classification ANN was presented. A feed-forward ANN was used for the classification task, with an accuracy of 12.4%.

It was evident that most of the input patterns can be classified into two or more different classes depending on whether you look at them separately, or in a word, or even in a sentence. In other words, character classification, especially handwritten Arabic

characters, depends largely on contextual information, and not only on the topographic features extracted from these characters.

A lot of enhancements can be made to the classification phase. A larger training set can definitely improve the results. In addition, the extracted topographic features are limited. More momentum-based features should be added if acceptable results are to be reached. Different ANN architectures are to be tested. Furthermore, since the classification process depends largely on contextual information, a lexicon could improve the results. However, the use of a lexicon can be a drawback when proper names, acronyms, or other words that are not likely to be in the lexicon are part of the text to be recognized.

References

- [Al-Badr] Badr Al-Badr and Robert Haralick. Symbol Recognition without Prior Segmentation. The Intelligent Systems Laboratory, University of Washington, Seattle.
- [Almeida1997] L. Almeida 1997. Multilayer Perceptrons. Handbook of Neural Computation (IOP Publishing Ltd and Oxford University Press), pp C1.2: 1-C1.2: 30.
- [Anderson1997] James A. Anderson 1997. Directions for future research in neural networks. Handbook of Neural Computation (IOP Publishing Ltd and Oxford University Press), pp H1.4: 1-H1.4:4.
- [BenAhmad] Mohamed Fehri and Mohamed Ben Ahmad. An Hybrid RBF/HMM Approach for Recognizing Multifont Arabic. Laboratoire Riadi-ENSI, Tunisia.
- [Blumenstein1997] M. Blumenstein and B. Verma. A Segmentation Algorithm used in Conjunction with Artificial Neural Networks for the Recognition of Real-World Postal Addresses. International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'97), Gold Coast, Australia, 1997, pp. 155-160.
- [Blumenstein] M. Blumenstein and B. Verma. Conventional Vs. Neuro-Conventional Segmentation Techniques for Handwriting Recognition: A Comparison. School of Information Technology, Griffith University, Australia.
- [Breuel1997] T. Breuel 1997. Handwritten Character Recognition Using Neural Networks. Handbook of Neural Computation (IOP Publishing Ltd and Oxford University Press), pp. C1.2:1-C1.2:30.

[Burges1993] Burges C J C, Ben J I, Denker J S, Lecun Y and Nohl C R. Off line recognition of handwritten postal words using neural networks. International J. Pattern Recognition and Artificial Intelligence, 1993, pp. 689-704

[Camargo1990] Francisco A. Camargo. Learning Algorithms in Neural Networks. The DCC Laboratory, Computer Science Department, Columbia University, New York, 1990.

[Eastwood1997] B. Eastwood, A. Jennings, and A. Harvey. A Feature Based Neural Network Segmenter for Handwritten Words, Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '97), Gold Coast, Australia, 1997, pp. 286-290.

[Grother1992] P. J. Grother 1992. Karhunen Loeve feature extraction for neural handwritten character recognition. Proc. SPIE 155-166

[Han1995] K. Han, I. K. Sethi. Off-line Cursive Handwriting Segmentation. ICDAR '95, Montreal, Canada, 1995, pp. 894-897.

[Jennings1998] Brett Eastwood and Andrew Jennings. Dynamic Multi-Level Resolution for Handwritten Word Recognition. Department of Computer Systems Engineering, RMIT University, Australia.

[Krishnan1984] Jonathan J. Hull and Ganapathy Krishnan. Optical Character Recognition Techniques in Mail Sorting: A Review of Algorithms. Department of Computer Science, State University of New York, June 1984.

[Kuebert1997] Edward J. Kuebert. Integration of Hand-Written Address Interpretation Technology into the United States Postal Service Remote Computer Reader System. 4th

International Conference on Document Analysis and Recognition (ICDAR'97), August 1997, pp. 892-896.

[Kulkarni] B. Verma, M. Blumenstein, and S. Kulkarni. Recent Achievements in Off-line Handwriting Recognition Systems. School of Information Technology, Griffith University, Australia.

[LeCun1989] LeCun Y, Boser B, Denker J S and Jackel L D. Back-propagation applied to handwritten zip code recognition. *Neural Computation*, 1989, pp. 541-551.

[Lee1996] S-W. Lee, D-J. Lee, H-S. Park. A New Methodology for Gray-Scale Character Segmentation and Recognition, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18, 1996, pp. 1045-1051.

[Mohiuddin] K. M. Mohiuddin and Andras Kornai. Recognition of Cursive Writing on Personal Checks. IBM Almaden Research Center.

[Rosenfeld] Rosenfeld. A Simple Parallel Algorithm for Skeletonization. http://www.cs.mcgill.ca/~laleh/rosen_alg.html.

[Seni] Giovanni Seni and Nasser Nasrabadi. An On-Line Cursive Word Recognition System. Center of Excellence for Document Analysis and Recognition, Department of Electrical and Computer Engineering, State University of New York.

[Srihari1993] S. N. Srihari. Recognition of Handwritten and Machine-printed Text for Postal Address Interpretation. *Pattern Recognition Letters*, 14, 1993, pp. 291-302.

[Srihari] Rohini K. Srihari. Use of Lexical and Syntactic Techniques in Recognizing Handwritten Text. Center for Document Analysis and Recognition (CEDAR), New York.

[Verma] B. Verma. Fast Training of Multilayer Perceptrons. School of Information Technology, Griffith University, Australia.

[Zemanec1994] Joseph Bell and Petr Zemanek. Test of two Arabic OCR Programs. Charles University, Prague, December 1994.