



# Machine Printed Character Recognition System Using Feature Point Extraction and Neural Network Classifier

Shanir Camacho Jara\*, Li Hong  
shanir2007@hotmail.com, lihongcsu@mail.edu.pe

## Abstract

Optical Character Recognition has become the aim of many research studies in the last decades, and that is just because its great influence in many industries such as banking, shipping, commerce, communications, marketing, license plate recognition, etc. Due to the great importance and promising future of this field, the purpose of this work is to introduce a system that, using feature character extraction and neural network classifier trained with the Back-propagation algorithm, is able to recognize machine printed English characters.

---

## 1. Introduction

The optical character recognition has its beginning around 1929 with the work done by Gustav Tauschek. It is usually abbreviated to OCR, is the mechanical or electronic translation of images, into machine-editable text. OCR is a field of research in pattern recognition [TVETER, 1998], artificial intelligence and machine vision [KURZWEIL, 1990]. Though, academic research in the field continues, the focus on OCR has shifted to implementation of proven techniques.

In this paper we include the whole framework to develop an OCR system using two methods for feature point extraction; the neural network development and the adjustments to get optimum results; and finally we perform further neural network analysis on sample data and compare the obtained results from the two proposed methods. In the last section we discuss the conclusions and future work.

### 1.1 Applications of OCRs

The main business and industrial applications of character and document recognition in the last forty years have been in the automatic classification of letters using handwritten postal, ZIP codes, automatic reading of administrative formularies of fixed structure, car plate automatic detection, recognition of text signs in traffic applications or automatic reading of amounts in bank. By supporting these applications, recognition capability has expanded in multiple dimensions: mode of writing, scripts, types of documents, and so on [CHELLAPILLA, 2004].

CONGRESO

INTERNACIONAL DE

COMPUTACIÓN Y

TELECOMUNICACIONES

COMTEL 2009

School of Information  
Science & Engineering,  
Central South University,  
Hunan 410083, China



40  
UIGV

## 1.2 Approaches to character recognition

Developing an OCR system is a complicated task and requires a lot of effort. Since the field of OCR started, investigators have proposed many different approaches like Artificial Neural Networks, Learning Vector Quantizing, Bayesian Techniques, Hidden Markov Models, and a variety of filtering techniques (such as Wavelets) just to cite a few.

However, the approaches that have had more acceptance for their effectiveness and efficiency is the use of artificial neural networks, because this technique can dramatically simplify the code and improve quality of recognition while achieving good performance [ANDERSON, 1995] [DAVALO, 1991]. Another benefit of using neural network in OCR is extensibility of the system – ability to recognize more character sets than initially defined. Therefore, the Artificial Neural Network (ANN) is a wonderful tool that can help to resolve such kind of problems [CYBENKO, 1989] [FUNAHASHI, 1989]. Therefore, after much research in this field we have decided to use ANN to implement the system, choosing the back-propagation algorithm to accomplish this task.

## 2. Our Approach

We have used the feature point recognition approach combined with a neural network classifier. Therefore, before starting to explain the system in detail, we need to introduce the main framework used to build the present system.

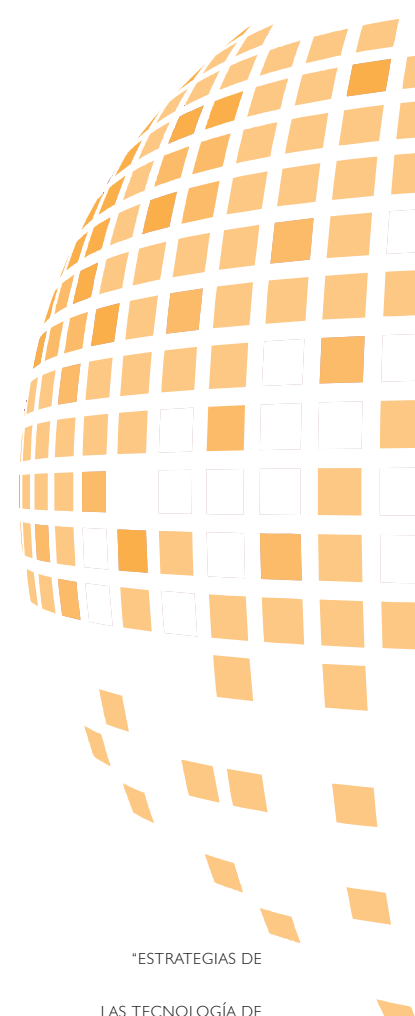
### 2.1 Feature Point Extraction

Feature point extraction, is a very important process in pattern recognition because the recognition performance of the system is highly related to the quality of the feature extraction. In literature, many definitions for a feature point have been proposed [SCHMIDT, 1997]; we can simply define features like a set of selected measurements extracted from the input pattern. The features are supposed to be invariant or less sensitive with respect to the commonly encountered variations and distortions, and also containing fewer redundancies. In many cases, the decision of what to measure or what features need to be extracted is based on human designers, and also dependent on the practical situations such as the availability of measurements and the cost of measurements. This system recognizes patterns using two methods to extract the features:

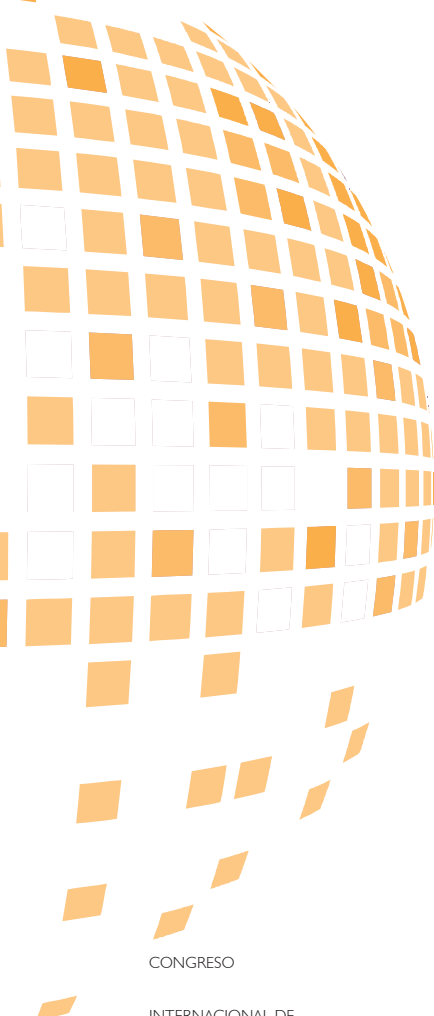
- Discrete Features
- Zoning

#### 2.1.1. Discrete Features

For each character, the following features may be extracted [RAMESH, 1989] [KUNDU, 1989]. These kind of systems scan each pattern-image to find: the number of vertical and horizontal lines; number of T-joints; the



"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"



CONGRESO

INTERNACIONAL DE

COMPUTACIÓN Y

TELECOMUNICACIONES

COMTEL 2009

number of X-joints; the number of bend points; presence of an isolated dot; total number of end points, etc. People tend to be sensitive to these features, for example the fact that the lines in a “Z” are connected in a certain way is more important than the individual lengths of those lines. These relationships are what should be used for discrete feature extraction.

### 2.1.2. Zoning

With this method the character image is divided into  $N \times M$  zones. From each zone, features are extracted to form the feature vector [TAKAHASHI, 1991]. In this case we computed the average gray level for each of the zones. The goal of zoning is to obtain the local characteristics instead of global characteristics.

### 2.2 Back-propagation neural network

The algorithm behind back-propagation networks was first described by Paul Werbos in 1974 [WERBOS, 1974], but it wasn't until 1986 through the work of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams [RUMELHART, 1986], that it gained recognition, and it led to a “renaissance” in the field of artificial neural network research.

A back-propagation network usually consists of three (or sometimes fewer) layers of neurons: the input layer, the hidden layer(s), and the output layer. We can see graphically the architecture in the figure 1:

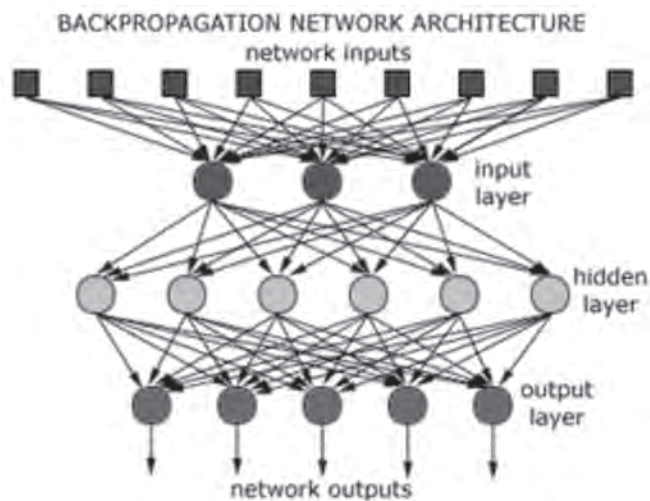


Figure. 1 Back-propagation network architecture

This algorithm works by what is known as supervised training. The principle of back propagation is actually quite easy to understand [CARLING, 1992]. The basic steps are:

1. Initialize the network with small random weights.

2. Present an input pattern to the input layer of the network.
3. Feed the input pattern forward through the network to calculate its activation value.
4. Take the difference between desired output and the activation value to calculate the network's activation error.
5. Adjust the weights feeding the output neuron to reduce its activation error for this input pattern.
6. Propagate an error value back to each hidden neuron that is proportional to their contribution of the network's activation error.
7. Adjust the weights feeding each hidden neuron to reduce their contribution of error for this input pattern.
8. Repeat steps 2 to 7 for each input pattern in the input collection.
9. Repeat step 8 until the network is suitably trained.

The challenge is to find the best way to update the weights and thresholds in each iteration (step 7) to minimize the error [MOHAMMED, 2005].

### 3. System Design

In this section we drill down into the main components and architecture of the system and the neural network classifier.

#### 3.1 Neural Network Architecture

The neural network architecture is based on the multilayer perceptron using a sigmoidal output function. However, before we train the network, we need to prepare the input patterns. In the figure 2 we show some processes carried out by the system to get the input data. After we have the input data ready, the system starts the training process.

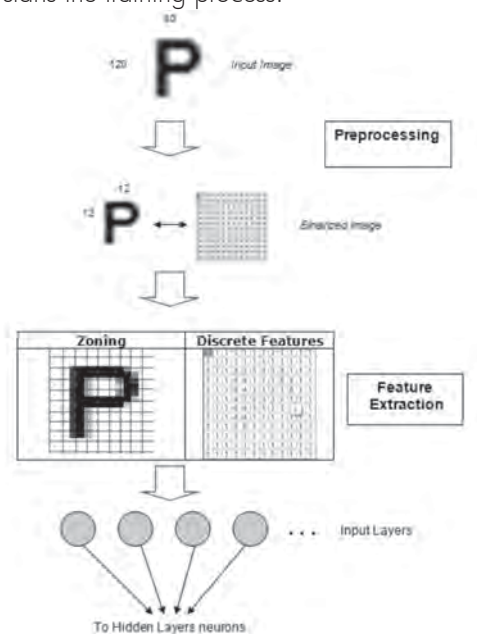
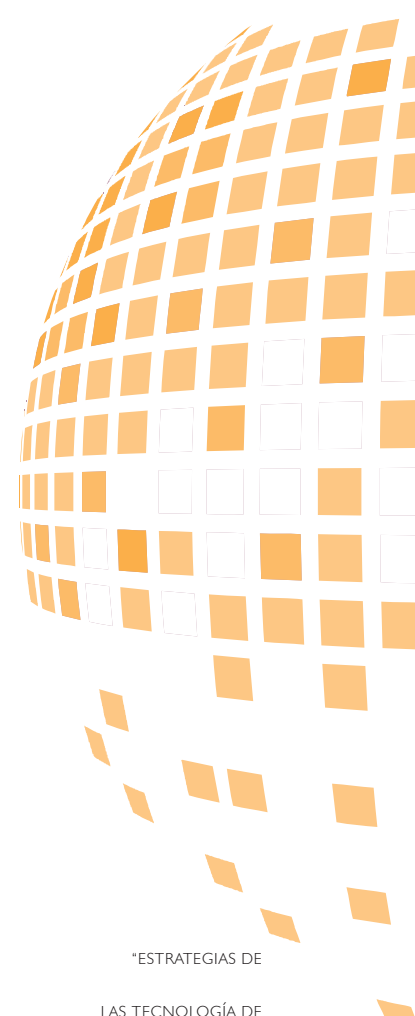
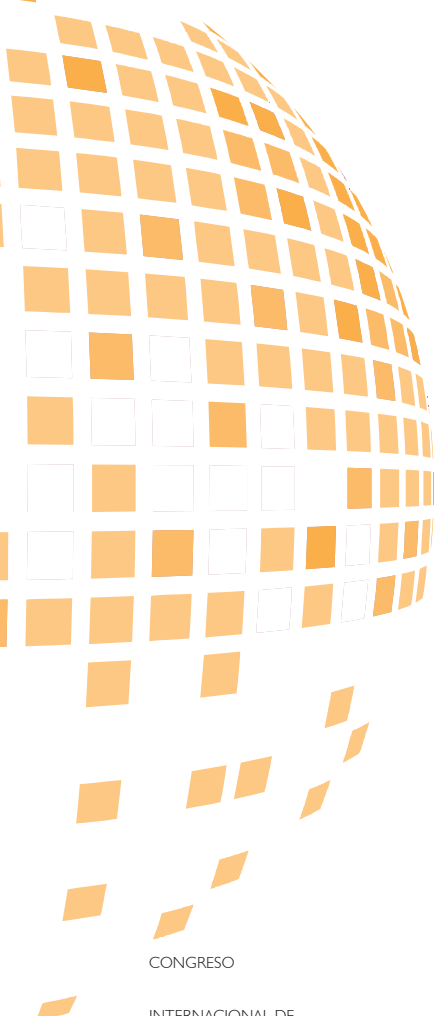


Figure. 2 Input patterns



"ESTRATEGIAS DE  
LAS TECNOLOGÍAS DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"



About the how to calculate the error:

Here, we have the variables:

- $output\_o$  = Activation value of the output neuron.
- $error\_o$  = Error at the output neuron
- $error\_h$  = Error at a hidden neuron
- $weight\_ho$  = A weight connecting a hidden neuron to the output neuron

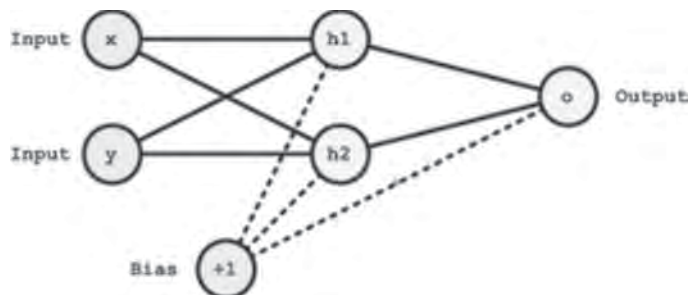


Figure. 3 Calculation of the error value

The error feed back to a hidden neuron is calculated by:

$$error\_h = error\_o * Derivative(output\_o) * weight\_ho$$

The calculation of the Derivative is explained in the later sections.

It is important to note that each pattern is presented in turn, and the network slightly adjust it before moving on to the next pattern. If we simply let the network perfectly correct the errors before moving onto the next pattern, it would never learn a generalized solution for the entire input collection.

### 3.2 Components of the system

The main components and their relationships are showed in the next figure:

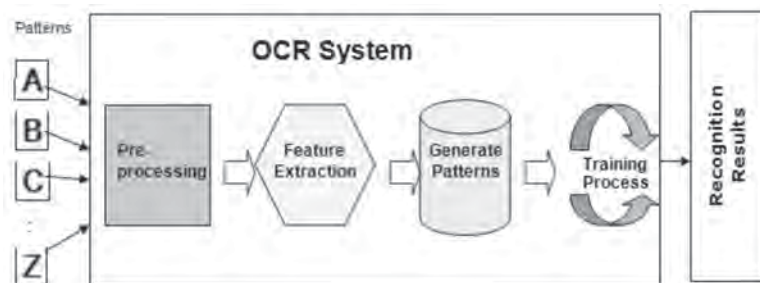
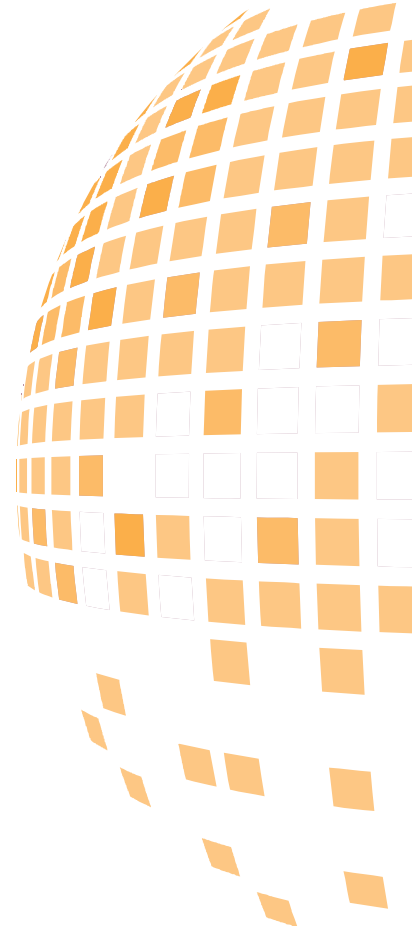


Figure. 4 System components



As we can see, the patterns-images are the input to the system that will help to train the neural network. Each one of the components will be explained in detail in the following sections.

### 3.3 Data structure model

We have found that many applications do not extract their entities, like neurons, layers of neurons, or a network of layers, instead they implement the entire neuron network architecture in a single class. In some cases, it is arguable what is better, but in most cases, it is favorable to split all these entities into distinct classes, what leads not only to easier understanding, but also allows reusing of all these components and building new neural networks architectures from smaller generic pieces. And, the present system follows this method, in this way it fulfills the Object Oriented Model (OOP) [ROGERS, 1996].

Therefore, we have created several classes, among the most important ones we can mention:

- Neuron: base class for all neurons, which encapsulates such common entities like a neuron's weight, output value, and input value.
- Layer: is a collection of neurons. This is a base class, which encapsulates common functionality for all neuron's layers.
- Network: represents a neural network, what is a collection of neuron's layers. It implements specific neural network architecture.
- Activation Function: are used in activation neurons (the type of neuron, where the weighted sum of its inputs is calculated and then the value is passed as input to the activation function, and the output value becomes the output value of the neuron).
- Back Propagation Learning: This class contains all the implementation of the back-propagation algorithm explained in the previous section.

Additionally, we can look at the relationship between classes in the next figure:

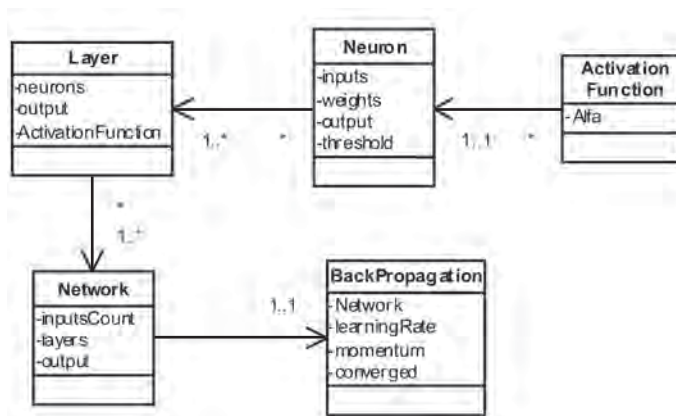
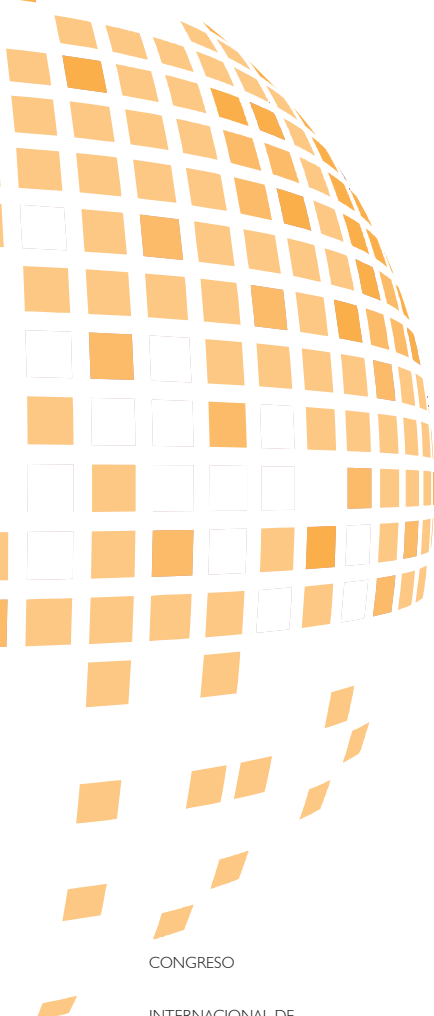


Figure. 5 Class diagram of the system

"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"



CONGRESO

INTERNACIONAL DE

COMPUTACIÓN Y

TELECOMUNICACIONES

COMTEL 2009

## 4. Preprocessing

The preprocessing is the first step and is quite straightforward; once we input the image into the system we perform binarization.

### 4.1 Binarization

Document image binarization (thresholding) refers to the conversion of a gray-scale image into a binary image. There are two categories of thresholding:

- Global, picks one threshold value for the entire document image which is often based on an estimation of the background level from the intensity histogram of the image.
- Adaptive (local), uses different values for each pixel according to the local area information.

In this case, we have used Global thresholding (value = 128), which means that the image function  $Z(x, y)$  now takes two values:  $Z(x,y)=0$  for printed pixels, and  $Z(x,y)=1$  for background pixels, instead of the 256 gray level values. In the figure 6 we present a screenshot of the system when we input a grayscale image (size 17x20) and in the figure 7 what we get after the preprocessing (12x12 pixel).



Figure. 6 Gray scale image

Binary Image:



Figure. 7 Processed image

## 5. Feature extraction

The feature points selection problem implies the selection from the whole set of available features of the subset, allowing the most discriminative power. The choice of a good feature point subset is crucial in the classification process and if the considered feature point set does not include all the information needed to discriminate samples belonging to different classes, the achievable performance may be unsatisfactory, regardless of the learning algorithm effectiveness [GUYON, 2003]. In this work we have proposed two feature point selection methods that will be in charge of extracting the relevant features from the samples.

### 5.1 Discrete Features

This process first analyzes the input image searching for discrete feature points (Table 1), we found more convenient to just loop through the entire

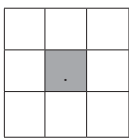
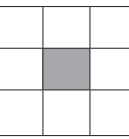
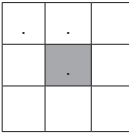
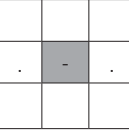
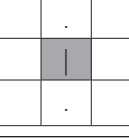
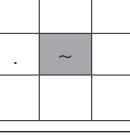
image of the character and examine each pixel in turn. Then, if the pixel is black (0), we check its eight neighbors (Figure 8).

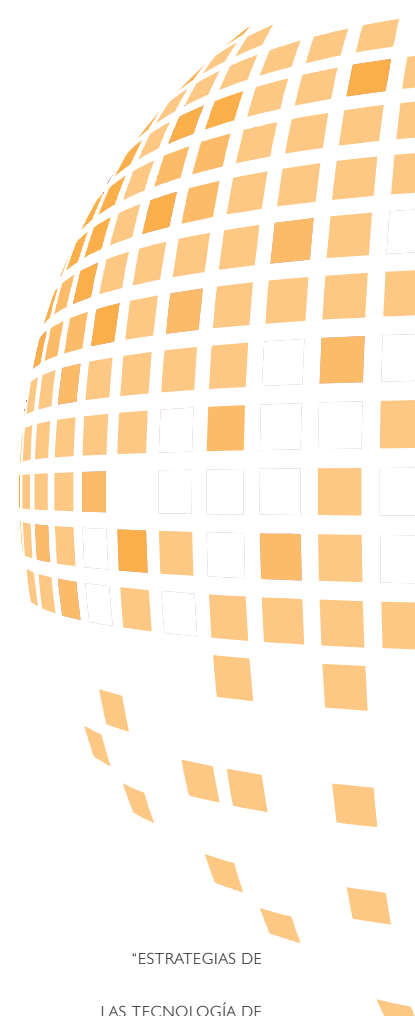
2	3	4
1	X	5
8	7	6

Figure. 8 X's Neighbor pixels

From the table we can conclude that, since each neighbor can also only be on (0) or off (255), there are 256 possible combinations of neighborhoods. From these 256, we have found that, for our system, just 10 represent significant feature points. In the next table we can look at them:

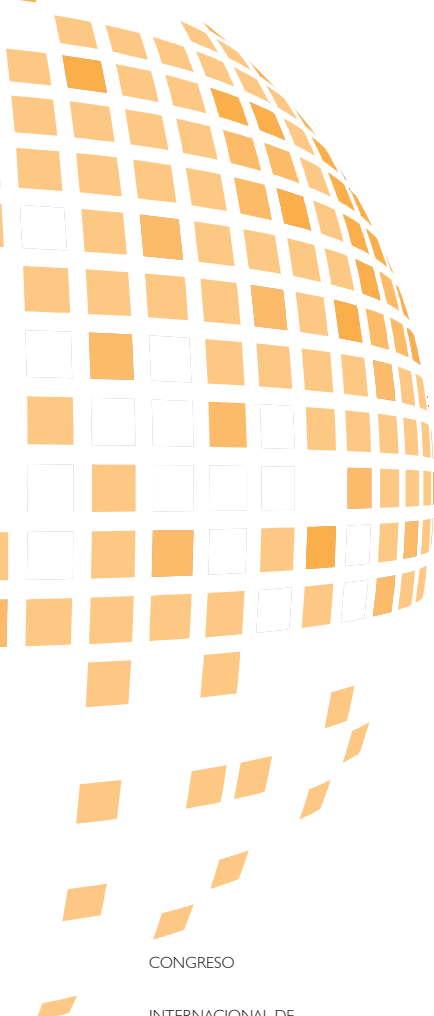
Table 1. Primitive Features Points

Nº	Feature Point	Examples
0	Isolated point (all neighbors are off, which represents noise, then we obviate it)	
1	Blank space (neighbors can be on or off)	
2	Black point (one or more neighbors points are on)	
3	Horizontal line (two neighbors aligned horizontally)	
4	Vertical line (two neighbors aligned vertically)	
5	End point (it has a single neighbor)	



"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"





6	Diagonal (two neighbors aligned diagonally)	
7	T-joint (three neighbors)	
8	Corner or bend point	
9	Center (all the neighbors are on)	
10	Cross	

We can explain better this process with an example: In the figure 9 we have the image of the letter "H" after the pre-process (W=White and B=Black). After we loop through this matrix labeling the discrete features mentioned in Table 1, we get the new matrix of the figure 10:



Figure. 9 Binarized image of "H"

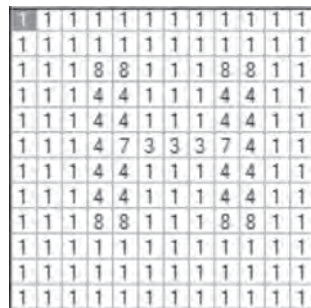


Figure. 10 Discrete feature points labeled

It's clear that we can work with more discrete features, but according to our experiment results these feature points make the system not to overload with too many uninteresting points, besides it makes the feature point extractor faster and reliable.

## 5.2 Zoning

The system takes the features based on the brightness map of the image. To create this map we split the image into squares like is showed in the figure, and then calculate the average value of each square.

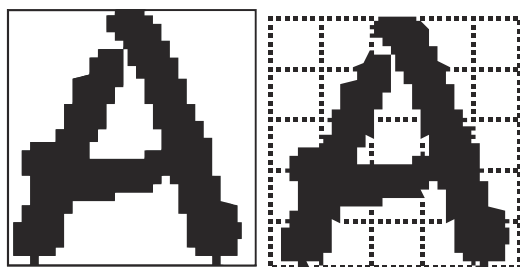


Figure. 11 Zoning method

## 6. Generating Patterns

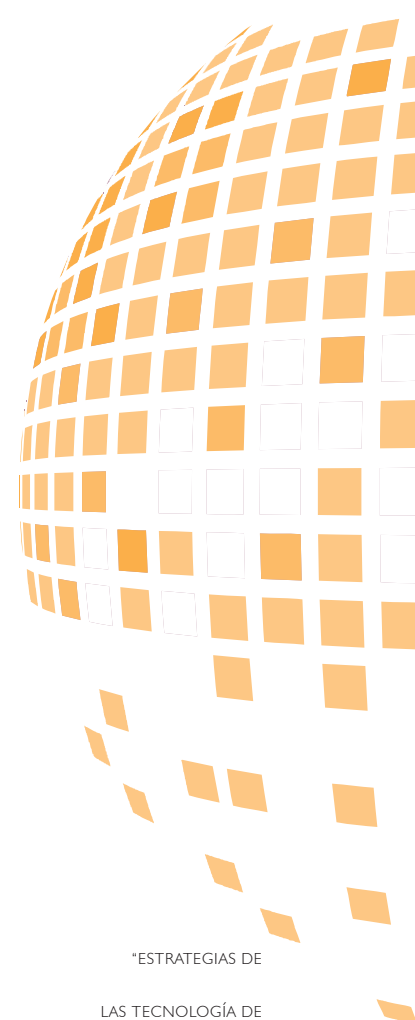
Basically, each training pattern consists of two single-dimensional arrays of float numbers: Inputs and Outputs (target) arrays.

- Input: is an array which contains the input data. Namely, the feature vector representations of each letter.
- Output: is the array of the pattern which represents an expected result. There are as many elements in this array as many characters the system is able to recognize. So, to recognize English letters from "A" to "Z" we will need 26 elements in the Outputs array.

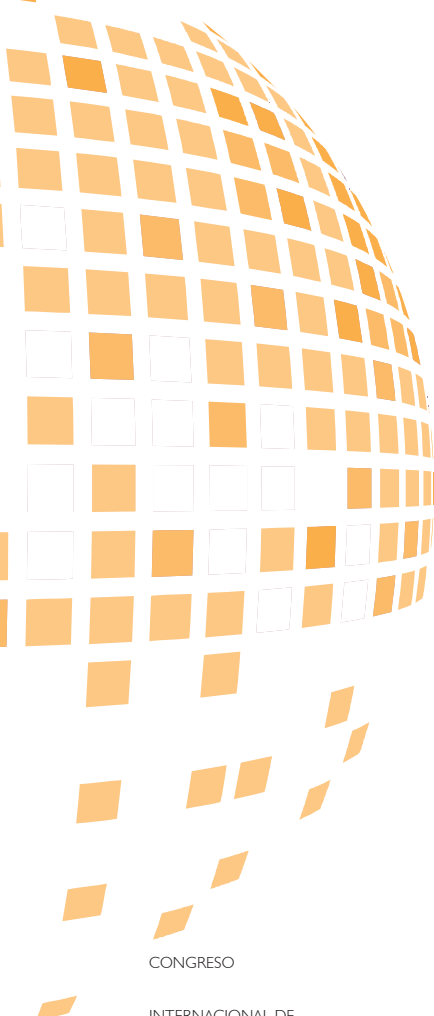
In other words, being  $K$  the size of the extracted feature vector, for each input vector  $P = \{x_1, x_2 \dots x_K\}$ , its target output is represented as the vector  $\{t_1, t_2 \dots t_{26}\}$ . Here  $x_i$  ( $1 < i < K$ ) represents the  $i$ th window measurement and  $t_j$  ( $1 < j < 26$ ) is the target output for supervised learning, i.e. if  $t_1 = 1$  and all other  $t_j$  are 0, then the input pattern is "A". Similarly, if  $t_2 = 1$  and all other  $t_j$  are 0, then the input pattern represents a "B". In this order, if the target output  $t_j = 1$  in position  $j$ , then it represents a printed character of the order  $j$  in the sequence A...Z.

## 7. Adjusting Network for performance

In previous items we have explained how Back-propagation algorithm works, but plain back-propagation is terribly slow and we want it to go faster. There are some adjustments that can be set to speed up the learning phase:



"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"



CONGRESO

INTERNACIONAL DE

COMPUTACIÓN Y

TELECOMUNICACIONES

COMTEL 2009

## 7.1 Scaling data to improve performance

There is a widespread belief that the input to the network must be scaled down to values between 0 and 1 if the activation function used is the standard sigmoid and -1 to 1 for tanh. But in fact, the inputs can be any real value; however the network may choke completely or learn very slowly if the magnitudes of the inputs are too large.

Consequently, to make sure our network won't "choke", we have implemented a process to scale down the values to the range [0...1]: In this way we look for the maximum element value of the matrix and then divide all the elements of the matrix by it.

Outputs are another story, for functions like the standard sigmoid with a range of 0 to 1 or tanh which runs from -1 to 1, we can only get outputs within this range, so we don't need scaling here.

## 7.2 Sigmoid Function:

The sigmoid function can be defined by the expression:

$$f(x) = 2 / (1 + e^{-2x}) - 1$$

And its derivative:  $f'(x) = 1 - f(x)^2$ .

We can see it graphically:

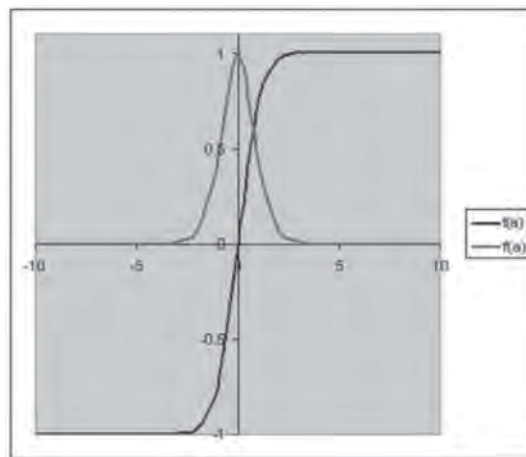


Figure. 12 Graph of the sigmoid function

But because we like the neurons to be working in the range of 0 to 1 we have adjust the function slightly to:

$$y = \frac{1}{1 + e^{-x}}$$

And its derivative:  $f'(x) = f(x)(1 - f(x))$ . So we get,

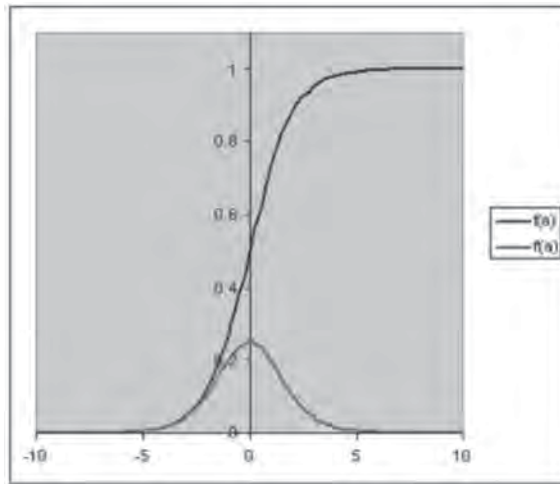


Figure. 1.3 Modified sigmoid function

This adjustment will bring us the benefit of being computationally less demanding, which will be important for larger networks.

### 7.3 Amount of hidden layers

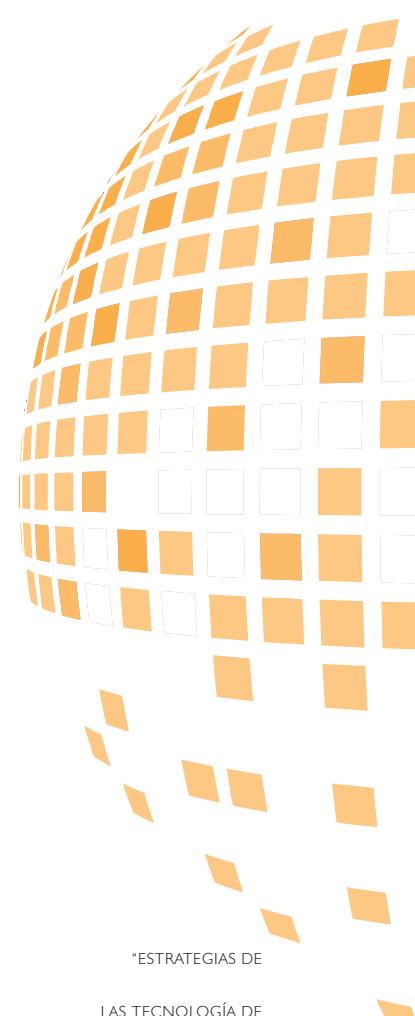
The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, but the size of the network plays a very important role. The right size for the network should be large enough to be able to learn the differences between the different classes and small enough to be unable to distinguish the differences between the feature vectors of the same class.

To find the right size, we consider more suitable to increase gradually the number of hidden layers since 0 until 3 hidden layers. In general, the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance [HABRA, 2005].

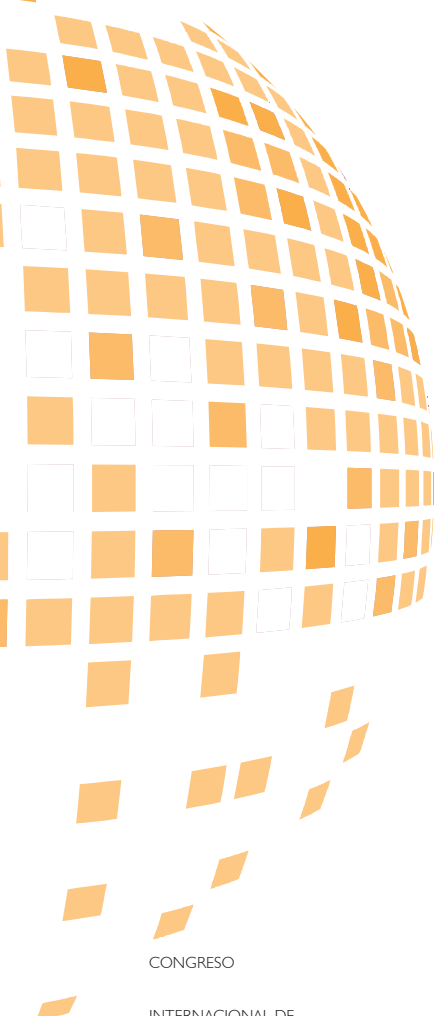
Table 2. Tests to find the correct amount of hidden layers

# Hidden Layers	Train time	# Iterations	Error	Misclassified
0	0:00:23	5441	0.999	0
1	0:00:14	2303	0.998	0
2	0:01:27	10000	25.410	26
3	0:01:46	10000	25.890	26

As we can see in Table 2, without any hidden layer the network is already able to recognize (since misclassified = 0), but when we add one hidden layer, the error and training time decreased favorably.



"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"



CONGRESO

INTERNACIONAL DE

COMPUTACIÓN Y

TELECOMUNICACIONES

COMTEL 2009

### Amount of neurons in the hidden layer

After deciding the number of hidden layers, now we face the problem of how many neurons should be used in the hidden layer to increase performance. To solve it, first we define a function  $G(x)$ ,

$$G(x) = Z * \# \text{ Characters in the output.}$$

And to find out the best value for  $Z$ , we run some tests.

Table 3. Tests to find the correct amount of neurons in the hidden layer

Z (%)	training time	# iterations	error
100%	0:00:14	2303	0.9980
75%	0:00:09	1667	0.9995
50%	0:00:06	1926	0.9983
30%	0:00:05	2161	0.9988
5%	0:00:05	2597	0.9985

It's obvious that we can obtain good performance results if we set the hidden layer neurons amount to the same value as the half of the amount in the output (50% of amount of Characters).

After all these adjustments, we have the network set to get misclassified value of "0/26", which means that the trained network can successfully recognize all patterns from the training set, with the best performance.

## 8. Experiments

Since the development and testing are done, we are ready to run some experiments. Therefore, first we labeled 26 images representing each one of the pattern to recognize, and run the learning process using the two mentioned methods. After that, we selected a dataset of 140 non-regular images (slanted and washed images) from all the 26 letters the English vocabulary has. The results are:

Table 4. Results of the first experiment

	Zoning (12 X 12)	Discrete Features
Training time	1' 26"	3' 30"
Iterations	11275	28267
Training error	0.0099996	0.0099998
Correct guess	119	110
Incorrect	21	30

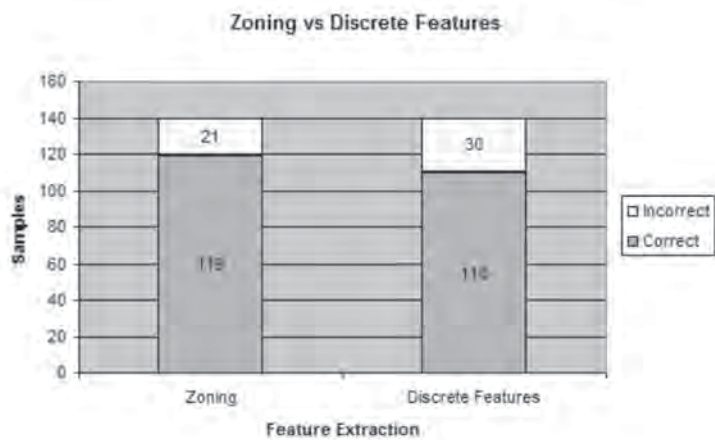


Figure. 14 Recognition rate comparison

In both cases the average training error is about 0.009. And, we can notice a difference in the training time: using the discrete features approach the network takes longer to converge than using zoning. Another important difference is the recognition rate, as is showed in the figure 14. With zoning we get a recognition rate of 85.00%, however, with the discrete features we get 78.57%, slightly lower.

A second experiment was carried out, this time to observe the behavior of the network when we add blur and noise to the samples (separately):

First, we work with blurriness. This is done by using the Gaussian method to contaminate our character recognition data. After that, we input the blur samples to the system and run the recognition process. We can see the results in the Figure 15 and 16 for the Zoning and Discrete Features, respectively. In each run, we change the blur amount, increasing the variable sigma from 0.5 to 3.

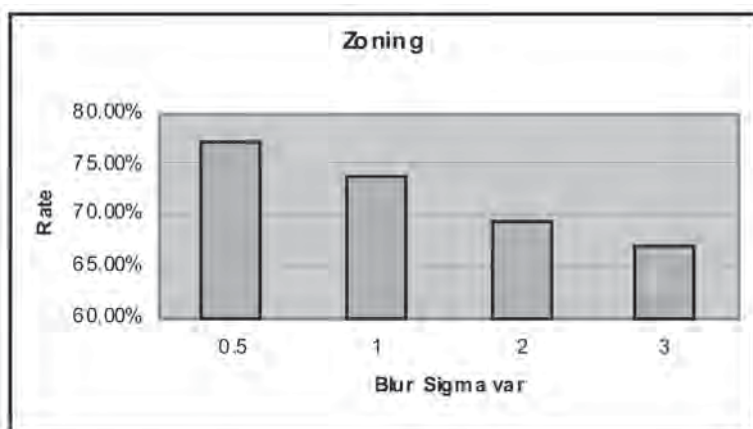
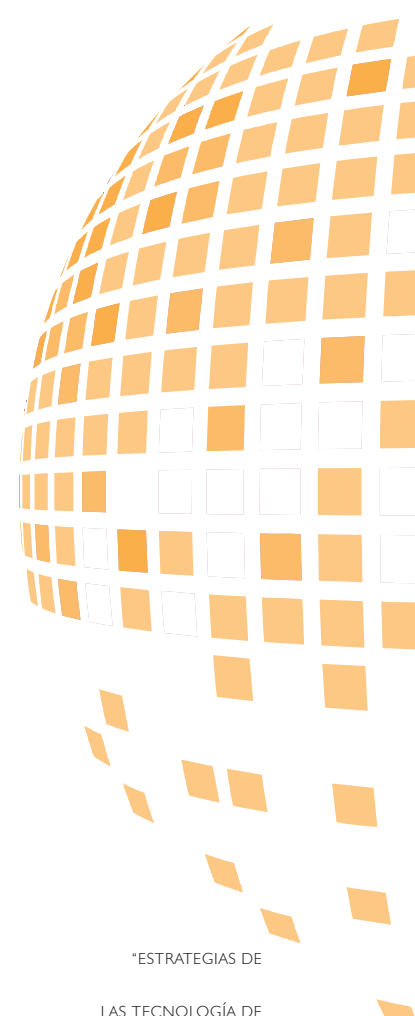
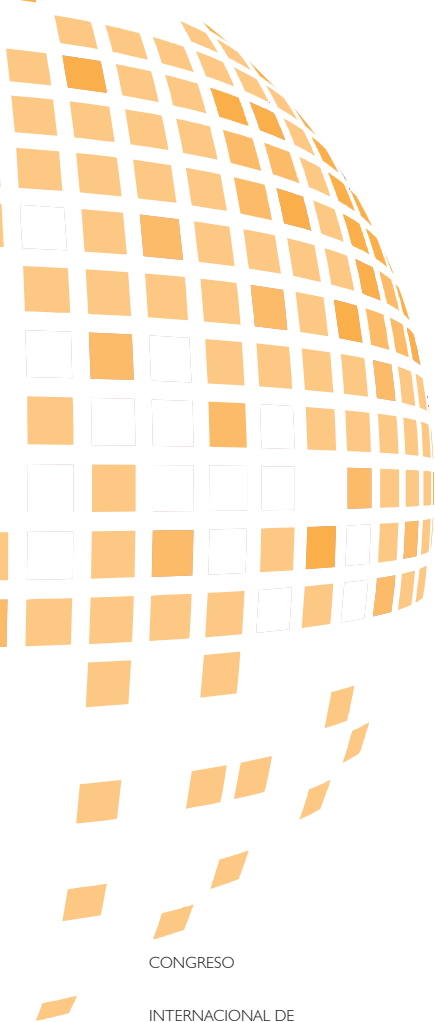


Figure. 15 Zoning with Blur samples



"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"



CONGRESO

INTERNACIONAL DE

COMPUTACIÓN Y

TELECOMUNICACIONES

COMTEL 2009

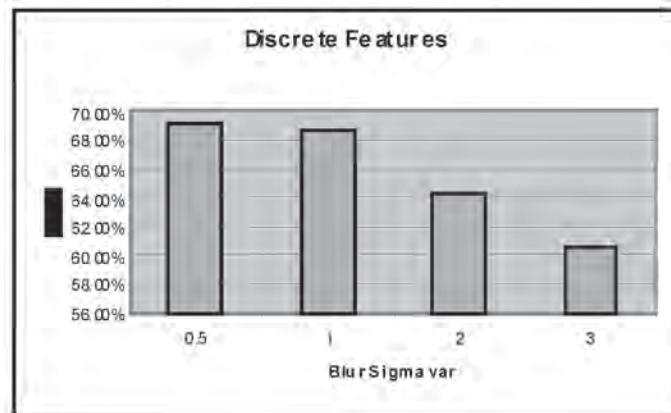


Figure. 16 Discrete Features with Blur samples

As was expected, increasing the blur amount decreases the recognition rate in both, the zoning and discrete features method. Although, with the Zoning method we still get a higher recognition rate (Figure 17).

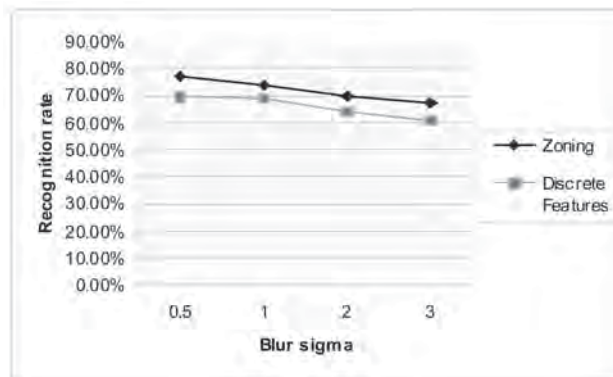


Figure. 17 Zoning Vs Discrete Features

In the second part of this experiment, we want to know what result we get if we add noise to the samples, so we applied the Salt & Pepper noise. In each one of the four runs we increase the noise amount (figures 18 and 19), which obviously lower the recognition rates.

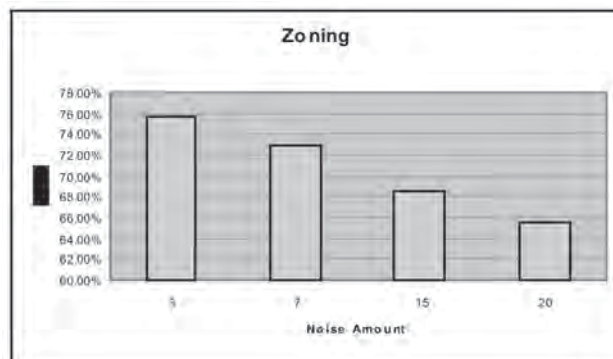


Figure. 18 Zoning with noisy samples

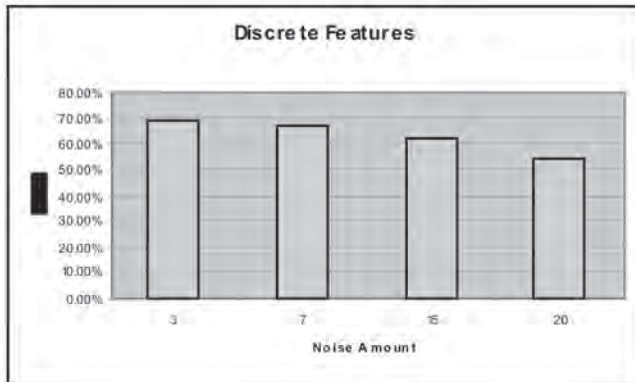


Figure. 19 Discrete Features with noisy samples

As we can notice in the figure 20, with noise images also we get better results using the zoning method.

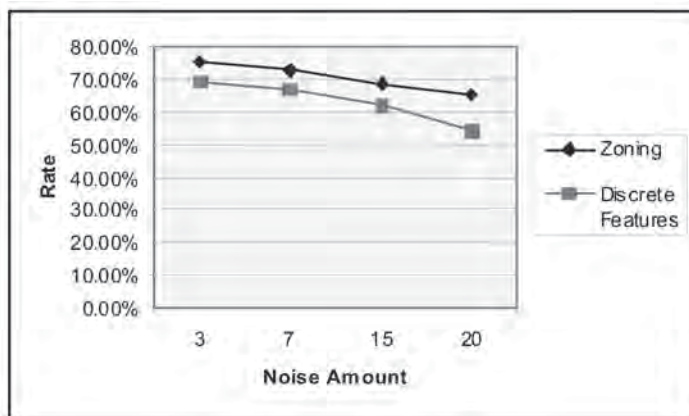
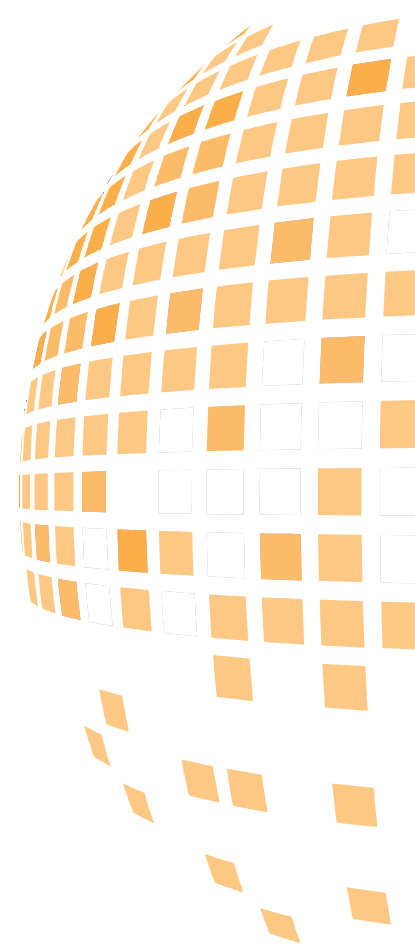


Figure. 20 Zoning Vs Discrete Features

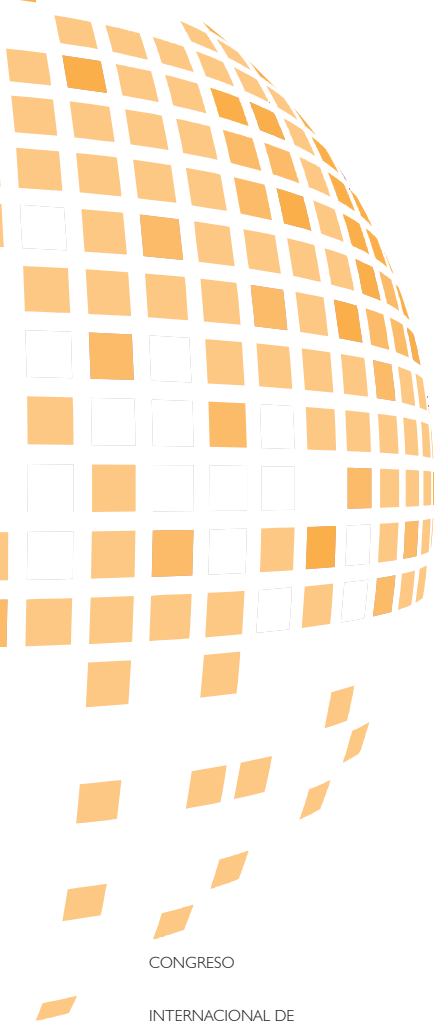
## 9. Conclusions

1. We have showed the differences between the two feature point extraction techniques adopted in this paper for recognizing machine-printed English characters combined with a neural network classifier, and also a comparison of results obtained using both of them.
2. Experimental results showed us that the performance of our OCR system described above is considerable high taking into account the fact that we have tested on several non regular images (raw, washed, blurry and noisy images) and that we have not used any noise filtering techniques.
3. In our study, we obtained recognition rates as high as 85.00% with the zoning and 78.57% using the discrete feature technique. The recognition rates are high for most trials with both techniques.



"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"





CONGRESO

INTERNACIONAL DE

COMPUTACIÓN Y

TELECOMUNICACIONES

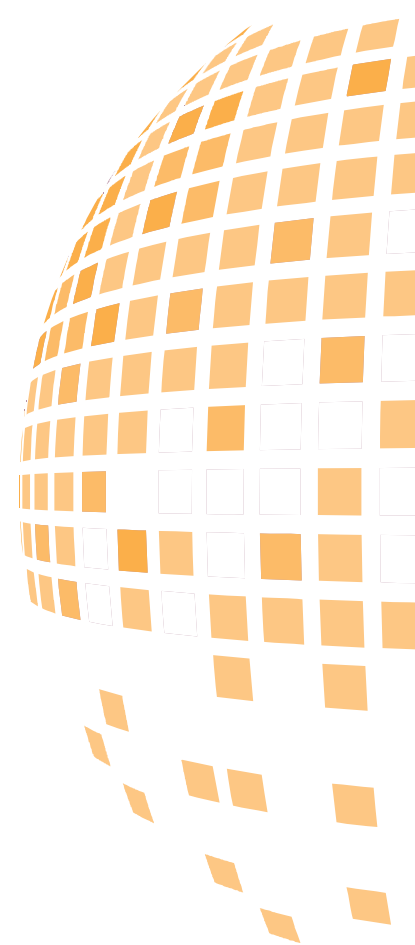
COMTEL 2009

4. Also further analysis has been performed when the character data is contaminated with noise and blur of varying amplitude. The results showed that the degradation in performance is graceful and predictable.
5. We have showed that even in neural networks OCR development we found the Object Oriented Model well suitable.

## 10. References

1. [ANDERSON, 1995] ANDERSON J. (1995) An Introduction to Neural Networks [M]. MIT Press. March 1995.
2. [KURZWEIL, 1990] KURZWEIL R. (1990) The age of intelligent machines [M]. MIT Press. Cambridge. 1990
3. [DAVALO, 1991] DAVALO E. and NAIM P. (1991) Neural Networks [M]. August 1991. 145p.
4. [TVETER, 1998] TVETER, (1998) R. The Pattern Recognition Basis of Artificial Intelligence [M]. Wiley-IEEE Computer Society Press. March 1998.
5. [ROGERS, 1996] ROGERS J. (1996) Object-Orientated Neural Networks in C++ [M]. 1st edition. October 1996.
6. [HABRA, 2005] HABRA Abdul. Neural Networks [EB/OL] [2005] <http://www.tek271.com/articles/neuralNet/IntoToNeuralNets.html>.
7. [CYBENKO, 1989] CYBENKO, G. (1989) "Approximations by superposition of a sigmoidal function" [J]. Math. Control Signals Systems. vol. 2. 1989. pp. 303-314
8. [FUNAHASHI, 1989] FUNAHASHI, K. (1989). The approximate realisation of continuous mappings by neural networks [J]. 1989. p.183-192
9. [WERBOS, 1974] WERBOS, P.J., (1974) "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences" [D]. Harvard University. 1974.
10. [RUMELHART, 1986] RUMELHART D.E., and MCCLELLAND J.L., (1986) Parallel distributed processing: explorations in the microstructure of cognition, I & II [M], MIT Press, Cambridge, MA. 1986.
11. [CHELLAPILLA, 2004] CHELLAPILLA K. and SIMARD P. (2004) "Using Machine Learning to Break Visual Human Interaction Proofs (HIPs) [J]" Advances in Neural Information Processing Systems 17, Neural Information Processing Systems (NIPS'2004), MIT Press. 2004.

12. [CARLING, 1992] CARLING A. (1992) Back propagation. *Introducing Neural Networks* [M], 1992. p. 133-154.
13. [MOHAMMED, 2005] MOHAMMED A. and WALID A. (2005) Speeding Up Back-Propagation Neural Networks [C]. 2005 Informing Science and IT Education Joint Conference. 2005.
14. [GUYON, 2003] I. Guyon and A. Elisseeff. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.
15. [SCHMIDT, 1997] C. SCHMIDT and R. MOHR, (1997) "Local gray value invariants for image retrieval" *IEEE transactions on pattern analysis and machine intelligence*, vol 19, May 1997.
16. [TAKAHASHI, 1991] H. TAKAHASHI, (1991) "A neural net OCR using geometrical and zonal pattern features" in *Proceedings of the First International Conference on Document Analysis and Recognition*, (Saint-Malo, France), p.821-828, 1991
17. [RAMESH, 1989] S.R.RAMESH (1989), "A generalized character recognition algorithm: A graphical approach," *Pattern Recognition*, vol.22, no.4, pp.347-350, 1989.
18. [KUNDU, 1989] A. KUNDU, Y. He, and P. BAHL (1989), "Recognition of handwritten word: First and second order hidden Markov model based approach," *Pattern Recognition*, vol.22, no. 3, pp. 285-297, 1989.



"ESTRATEGIAS DE  
LAS TECNOLOGÍA DE  
LA INFORMACIÓN Y  
COMUNICACIÓN EN  
EL CONTEXTO DE LA  
CRISIS MUNDIAL"