HANDWRITING ARABIC CHARACTER RECOGNITION USING LENET NEURAL NETWORK

Rashad A.Al-Jawfi,
Faculty of science , Department of Mathematics and Computer science.
algofi@yemen.net.ye

Abstract

Character recognition has served as one of the principal proving grounds for neural network methods and has emerged as one of the most successful applications of this technology. In this paper, a new network is designed to recognize a set of handwritten Arabic characters. This new network consists of tow stage. The first is to recognize the main shape of the character. And the second stage is for dots recognition. Also, the characteristics, structure, and the training algorithm for the network are presented.

Introduction

There are two ways to computerized the old printed Arabic books by means of computers for the facilitate discussion and research:

1- Enter those printed books manually and this is very expensive and requires great efforts that the Arabs abilities fails to do because there are millions of books that have scientific and historical value.
2- Entering the printed books using scanners and then using a software for recognizing symbols. This way seems faster and appears more economic.

Here appears the importance of developing that can perform the task of recognition precisely.

Most of the methods for pattern recognition, especially template matching techniques, are over sensitive to shifts in position and distortions in shape of the stimulus patterns, and it is necessary to normalize the position and the shape of the stimulus pattern beforehand. Therefore, the finding of an algorithm for pattern recognition which can cope with shifts in position and distortions in shape of the pattern has long been desired.

Fukushima (Fukushima,1980), proposed a new algorithm that gives an important solution to this problem. This algorithm can be realized with a multilayered network consisting of neuron-like cells. The network is called 'neocognitron'. The ability of the neocognitron has already been demonstrated in various experiments. The system was trained to recognize English handwritten numerals.

Naoum (Naoum et al,2000), use neocognitron to recognize the isolated handwritten Arabic characters.

LeCun (LeCun et al,1990) design a net called LeNet as an isolated non Arabic character recognizer.

In this paper we try to use LeNet to recognize the handwritten Arabic characters invariant to shift is position, change in size, and any distortion.

The Character Recognition Process

Figure 1. Shows the "typical" character recognition process, which starts with an optical image and ultimately produces a symbolic interpretation.

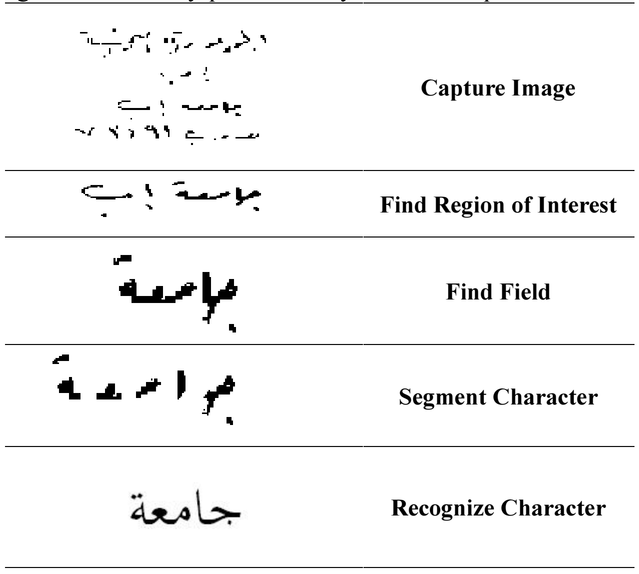


Fig 1.

The process is divided into a series of tasks that are usually executed independently. It begins with image capture in which an optical image is converted to a bit-map. Next the region of interest is located. Then the desired field is found. This field is then usually size-normalized and sometimes de-slanted. Finally, the characters are segmented and recognized. Note that the recognition phase is only one step in a long process. In our system we modify this model, using feedback from down-stream stages to influence up (Yuhas & Ansari, 1994).

LeNet

LeNet recognizes one character at time. The LeNet architecture is shown in Figure 2. In this figure Each small box represents a neural-net "unit" or "neuron" [Yuhas 1994]. The maps are generated by convolutions with feature extraction kernels. Input images are sized to fit in a 20x20 pixel field, but enough blank pixels are added around the border of this field to avoid edge effects in the convolution calculations, and returns a rank-ordered list of possible single-character interpretations of the input image, along with confidence scores for each interpretation.

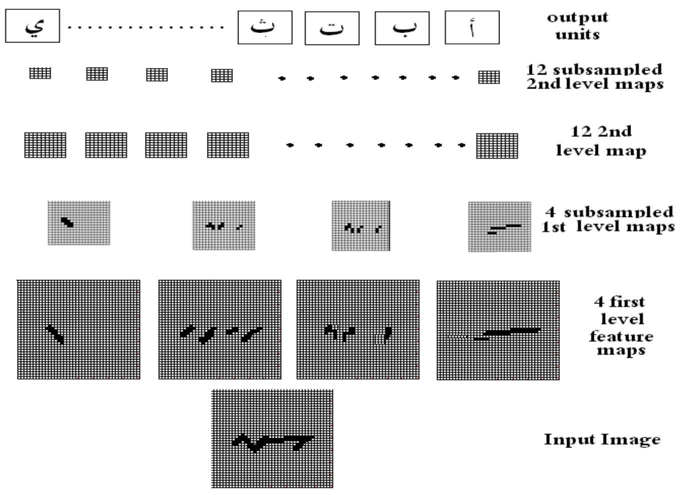

ي
ث
ت
ب
أ
output units
12 subsampled 2nd level maps
12 2nd level map
4 subsampled 1st level maps
4 first level feature maps
Input Image

Fig. 2

The activation of the input In this figure, and the first two layers of the network are indicted: darker shading indicates greater activity. The weighted connections between units are highly structured.
The maps are generated by convolutions with feature extraction kernels.

Segmentation

If our objective is to recognize a string of characters, the string has to be cut up into individual characters, a process called segmentation. The component do this operation is the segmenter. Each partial image ("segment") is then in turn classified by the recognizer.

The difficulty of the segmentation task strongly depends on the quality and type of the string. For machine printing of poor quality or for handwriting, where different characters might touch and single characters might be broken into several pieces, the task is very difficult. Each segments is called "Connected Components". Each of the three connected components in Figure 3(a) is individual digit. This is not always the case; in Figure 3(b) there are three digits and three connected components as well. Not one of the connected components is an individual digit. The "ط" and "ف" are combined to one connected component and the "س" is disconnected into two connected components. There is need for a much more sophisticated segmenter. The first connected component must be bisected into two segments, and remaining two connected components joined to one segment.

Whatever segmentation method is used, there are limitations in using the pipeline scheme shown in Figure 3.

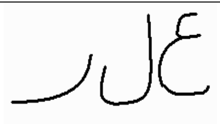
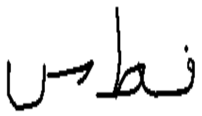
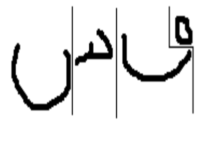
		
(a)	**(b)**	**(c)**

Fig 3

For the characters shown in Figure 3(c), we consider the "tentative cuts" where a connected components analysis locates gaps between blobs of ink. Such cuts are within the "س" and the "ف" and between them. The next step in our segmentation process is to pass segments and possible segment combinations to LeNet for scoring as possible characters. Figure 4, shows these segments and segment combinations along with their top scoring classification and confidence level. The number near the top of each box is the most likely classification of the "ink" in the box. The number near the bottom is LeNet's relative confidence in the classification. In order to choose a consistent segmentation of the image we must ensure that all the black area is accounted for, and that it is only used once (LeCun, et al,1990).

Recognition-based Segmentation

The segmentation that gives the best combined score is chosen. This "recognition driven" segmentation is usually used in conjunction with dynamic programming to efficiently find the optimal solution. In this approach the recognition engine and the segmenter are tightly coupled. This is a method we believe to be applicable to multi-character recognition.

The number of possible segmentations of a specific image is extremely large. Even if we could rely only on the recognition field to determine the correct segmentation, it would not be feasible to try scoring all possible segments, even with an extremely fast classifier; we could not reach a solution in reasonable computing time. It is necessary to limit the number of candidate segmentations as much as possible. We would like to limit the candidates we consider, to the smallest possible set while retaining high confidence that the correct solution is a member of it.

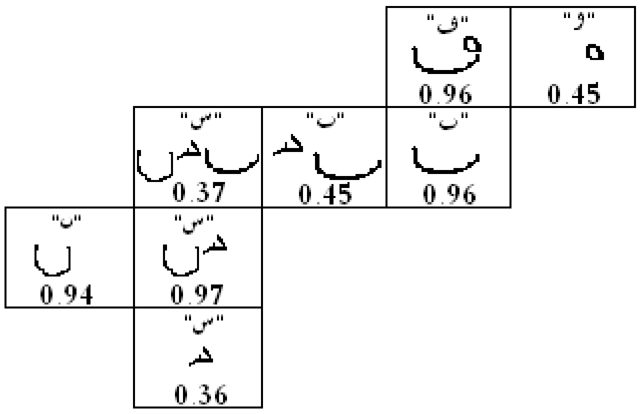


Fig. 4

Backpropagation Learning Algorithm

The backpropagation algorithm has been widely used as a learning algorithm in feed forward multilayer neural networks. The backpropagation is applied to feed forward artificial neural networks with one or more hidden layers.

Based on this algorithm, the network learns a distributed associative map between the input and output layers.

In general, the difficulty with multilayer Perceptrons is calculating the weights of the hidden layers in an efficient way that result in the last output error. To update the weights, one must calculate an error. At the output layer this error is easily measured; this is the difference between the actual and target outputs. At the hidden layers, however, there is no direct observation of the error; hence, some other technique must be used to calculate an error at the hidden layers that will cause minimization of the output error, as this is the ultimate goal.

We want to train a multi-layer feed forward network by gradient descent to approximate an unknown function, based on some training data consisting of pairs (x,t). The vector x represents a pattern of input to the network, and the vector t the corresponding target. The overall gradient with respect to the entire training set is just the sum of the gradients for each pattern; in what follows we will therefore describe how to compute the gradient for just a single training pattern.

During the training session of the network, a pair of patterns is presented (x_k, t_k), where x_k is the input pattern and t_k is the target or desired pattern. The x_k pattern causes output responses at each neuron in each layer and, hence, an actual output O_k at the output layer. At the output layer, the difference between the actual and target outputs yields an error signal. This error signal depends on the values of the weights of the neurons in each layer. This error is minimized, and during this process new values for the weights are obtained. The speed and accuracy of the learning process of updating the weights also depends on a learning rate.

Handwriting Arabic Character Recognition

the main shapes of Arabic alphabet

Arabic writing and Arabic characters have many features that make the Arabic character recognition system differs than the recognition systems for other languages such as Latin and Chinese. Arabic words are written from right to left in a cursive script in both handwritten and typewritten (Amin, 2003). Also, Arabic characters have many characteristics that complicate the recognition of such characters. Some of these characteristics are listed below:

1. Arabic alphabet consists of 29 characters. The shape of most of these characters is a function of their positions within the word, where each character can have up to four different forms that increase the number of patterns from 29 to about 60 patterns (Amin, 1997).
2. Most of Arabic characters (17 of 29) have a character complementary that is associated with the body of the character. This complementary may be dot, two dots, three dots, or zigzag (in Arabic it is called hamza). It can be above the character (ـف), below (ـب) , or inside the character (ج) (Saleh, 1998).
3. There are many groups of characters that have the same body, but they are distinguished by the number of dots (ـنـ،ـتـ،ـثـ), the position of dots (خ،ج), or whether it is dot or zigzag (ـنـ،ـئـ) (Saleh, 1998).
4. An Arabic character may connect to one side (either right or left), to both sides, or to neither side. For example, the 'Alif' connects to the right but never to the left. This information offers a clear-cut method of (usually) validating or increasing the confidence level in the classification of some characters. For example, if a 'Waw' was mixed with a 'Meem', the conflict could easily be resolved to the 'Meem's' advantage if the suspected character was connected to the left, for a 'Waw' never connects to the left (Kharma & Ward, 2001).

A decision tree was used for classifying Arabic characters. The tree is quite shallow- 4 levels deep, which is better for speed. Also, the tests used for classification are themselves simple. This combination, together with multiple testing features is designed to ensure speed of execution, as well as certainty of classification. (Kharma & Ward, 2001).

Since the recognition is not affected by adding noise, if the dots are ignored, the number of different shapes to be recognized will be reduced from 103 to 6٨ as shown in table 1.

Table 1. Main Arabic characters shapes

Isolated	Beginning	Median	End	Isolated	Beginning	Median	End	Isolated	Beginning	Median	End
أ	-	-	ـا	ب	بـ	ـبـ	ـب	ح	حـ	ـحـ	ـح
د	-	-	ـد	ر	-	-	ـر	س	سـ	ـسـ	ـس
ط	طـ	ـطـ	ـط	ص	صـ	ـصـ	ـص	ع	عـ	ـعـ	ـع
ف	فـ	ـفـ	ـف	ق	قـ	ـقـ	ـق	ك	كـ	ـكـ	ـك
ل	لـ	ـلـ	ـل	م	مـ	ـمـ	ـم	ن	نـ	ـنـ	ـن
ه	هـ	ـهـ	ـه	و	-	-	ـو	ي	يـ	ـيـ	ـي
ء	-	ـئـ	-	لا	-	-	ـلا				

In order to recognize Arabic character, the problem will be divided into three parts: body classifier, where the main body of the unknown character regardless will be classified of the dots or zigzag. In addition, the noise and character complementary that is associated with the body of the unknown character will be removed using Hopfield network. This part classifies the main body through two stages. The first will be used to enable this network of removing the noise and enhancing the body of the character, the body of Arabic characters will be stored within the network. Thus, this network will consider the character complementary as a noise and the output of the network will be the stored body that is similar to the body of the unknown character. The output will be send to another stage in this part as an input and outputs the class that the unknown pattern belongs to, using a backpropagation algorithm. The second part is the complementary classifier, where the complementary of the unknown character recognized. It has to determine the number of dots, position, and whether they are dots or zigzag. Thus, the dots and their number can be recognized by their size. And the third part is the aggregate classifier. Where the result of the previous parts combined and produces the class that the unknown character belongs to.

Using LeNet for Arabic character recognition

Our system of Arabic character recognition can be divided into two stages as following:

We will start with main body recognition using LeNet, and after that, dots recognition using search, as following:

Firstly, noise will be isolated from the input character. Then, the search area will be determining dependently on the type of shape, some areas will be above the main shape and others below it. And lastly, searching the determined search area in the isolated noise array for single, double, or triple dots depending on the shape of character.

Network design

input and output unit.

The input of the network is a 16 by 16 normalized image. The output is composed of 68 units (one per class) and uses place coding.

feature Maps and Weight Sharing.

The detection of a particular feature at any location on the input can be easily done using the "weight sharing". Distinctive features of an object can appear at various locations on the input image. Therefore it seems judicious to have a set of feature detectors that can detect a particular instance of a feature anywhere on the input plane. It can be interpreted as imposing equality constraints among the connection strengths. This technique can be implemented with very little computational overhead.

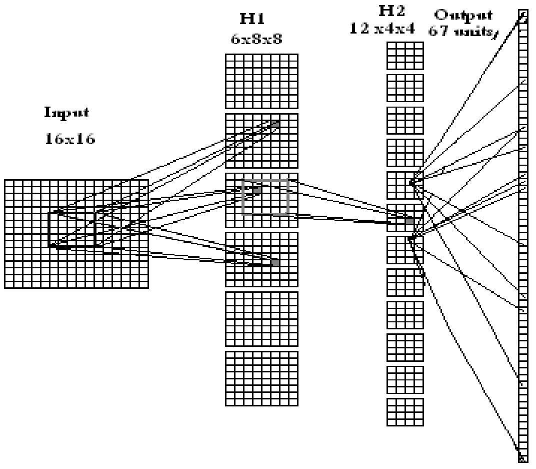
Weight sharing not only greatly reduces the number of free parameters in the network but also can express information about the geometry and topology of the task. In our case, the first hidden layer is composed of several planes that we call feature maps. All units in a plane share the same set of weights, thereby detecting the same feature at different locations. Since the exact position of the feature is not important, the feature maps need not have as many units as the input.

Network Architecture.

The network is represented in Figure 5. Its architecture is a direct extension of the one proposed in (LeCun, 1989). The network has two hidden layers named $H1$ and $H2$.

$H1$ is composed of 6 groups of 64 units arranged as 6 independent 8 by 8 feature maps. Each unit in a feature maps takes input on a 5 by 5 neighborhood on the input plane. The motivation is high resolution may be needed to detect the presence of a feature, while its exact position need not be determined with equally high precision.

It is known that the kinds of features that are important at one place in the image are likely to be important in other places (LeCun et al, 1998). Therefore corresponding connections on each unit in a given feature map are constrained to have the same weights. On other words, each of the unit in $H1_1$ uses the same set of 25 weights. The function performed by a feature map can thus be interpreted as a nonlinear sub sampled convolution with a 5 by 5 kernel.



5

Of course, units in another map share another set of 25 weights. Units do not share their biases. Each unit thus has 25 input plane take their input from a virtual background plane whose state is equal to constant, predetermined background level.

layer $H1$ comprises 384 units (8 by 8 times 6), 9984 connections (384 times 25) plus 384 biases, but only 534 free parameters (384 biases plus 25 times 6 feature kernels) since many connections share the same weight.

Layer $H2$ is composed of 12 features maps. Each feature map contains 16 units arranged in a 4 by 4 plane. The connection scheme between $H1$and $H2$ is quite similar to the one between the input and $H1$. Each unit in $H2$ combines local information coming from 4 of the 6 different feature maps in $H1$. Its receptive field is composed of six 5 by 5 neighborhoods centered around units that are at identical positions within each of the 6 maps. Connections falling off the boundaries are treated like as in $H1$. To summarize, layer $H2$ contains 192 units (12 times 4 by 4) and there is a total of 19392 connections between layers $H1$ and $H2$ (192 units times 101 input lines). All these connections are controlled by only 792 free parameters (6 feature maps times 100 weights plus 192 biases). Connections entering $H1$ and $H2$ are local and are heavily constrained. Units , connections with previous layer, and free parameters of all layers are in table 2.

The output layer has 68 units and is also fully connected to $H2$, it consists of 13124 weights. In summary, the network has 900 units, 42500 connections, and 14450 independent parameters.

Network Training

The nonlinear function used at each node was a scaled hyperbolic tangent Symmetric functions of that kind are believed to yield faster convergence, although the learning can be extremely slow if some weights are too small (LeCun et al, 1998). The target values for the output units were chosen within the quasi linear range of the sigmoid. This prevents the weights from growing indefinitely and prevents the output units from operating in the flat spot of the sigmoid. The output cost function was the mean squared error.

Before training, the weights were initialized with random values.

The process starts with applying the first pattern, then the patterns were repeatedly presented in a constant order. The weights were updated according to the so-called stochastic gradient (updating after each presentation of a single pattern) as opposed to the "true" gradient procedure (averaging over the whole training set before updating the weights) From empirical study (supported by theoretical arguments), the stochastic gradient was found to converge much faster than the true gradient, especially on large, redundant data bases.

Table 2. units, connections and free parameters of all layers

Layer	Units	Connections with previous layer	Free parameters
Input	256	---	---
First hidden layer	384	9984	534
Second hidden layer	192	19392	792
Output	68	13124	13124
Summation	900	42500	14450

feature work

The needing for a lot of interest to recognize handwriting Arabic character is stile required, to minimize a lot of errors depending with the design of handwriting language itself. Also the recognition of the pictures that not related with written text and select it as a picture stile open question.

References

1. Amin, A. (1997) Off line Arabic Character Recognition A Survey. ICDAR'97,vol.2 pp. 596-599.
2. Amin, A. (2003) Recognition of Hand-printed Characters Based on Structural Description and Inductive Logic Programming, PRL vol. 24, No. 16, pp. 3187-3196.
3. Burges, C. J. C. (1998) A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2 (2), pp.121–167.
4. El-Wakil, S., Shoukry, A. (1989) On-line Recognition of Handwritten Isolated Arabic Characters, Pattern Recognition, Vol 22, No 2, pp 97-105.
5. Fukushima, k. (1980) neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by a shift in position, biol. Cybernetics 36, pp.193-202.
6. Hashimoto, W.(2003) Quadratic forms in natural images, Network: Computation in Neural Systems 14 (4), pp.765–788.
7. Kharma, N., Ward, R. (2001) A Novel Invariant Mapping Applied to Hand-written Arabic Character Recognition, Pattern Recognition, vol. 34, no 11, pp.2115-2120.
8. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L. (1990) Handwritten Diget Recognition with a Back-propagation Network, in Advances in Neural Information Processing 2, (NIPS 89) D. L. Touretsky ed., pp.396-404.
9. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998) Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11), pp.2278–2324.
10. Meuller, K.-R., Mika, S., R¨atsch, G., Tsuda, K., Sch¨olkopf, B. (2001) An Introduction to Kernel–Based Learning Algorithms, IEEE Transactions on Neural Networks 12 (2), pp.181–202.
11. Naoum, r., Al-rawi, H. And Jabir, A. (2000) supervised neocognitron for handwritten Arabic characters recognition, Basrah J. Science, A, vol. 18, no.2, pp.109-126.
12. Saleh, S. (1998) An On-Line Automatic Arabic Document Reader, MSc. Thesis, University of Basrah, Iraq.
13. Yuhas, B., Ansari, N. (1994) Neural Networks in Telecommunications, Kluwer Academic Publishers.