Chapter 4. Classification of symbols of Gujarati script using Wavelets and Multilayer Perceptron

4.1 Introduction

This chapter deals with the classification of the printed isolated Gujarati alphabets extracted from the laser printed documents. As discussed in chapter-2, classification is made up of two major tasks: viz. 1. Extraction of the important features from the glyph. 2. Recognition of each glyph using these important features.

Very little work is found in the literature regarding the recognition of Gujarati script. In our experiments two fairly representative sets of printed Gujarati characters and modifiers are chosen and subjected to classification. In our experiments we have used Daubechies D4 wavelets as a feature extractor and Multilayer Perceptron, a Artificial Neural Networks architecture as a classifier and achieved significant recognition accuracy.

Traditional MLP architecture consists of nonlinear transfer functions in all the hidden layers and the output layer which makes the convergence process slow. So instead of using nonlinear transfer function at the output layer, we have used linear activation function [4] and got significant accuracy as described below.

The sample and test images for the Gujarati alphabets are obtained from the scanned images of printed Gujarati text and their features are extracted in terms of wavelet coefficients. Two Multi-Layer Perceptron (MLP) networks, one for the classification of alphabets which fall in the middle zone and the other one for classifying the modifiers which fall in the lower zone are designed. These networks achieves 94.46 % and 96.32 % of accuracy for alphabets and modifiers respectively on a test set which is higher than those quoted by any previous documented effort found in literature for the Gujarati script [38].

The chapter is divided in to six sections. After giving an introduction in the first section, the second section describes the generation of database with some examples of various zone glyphs. We highlight two dimensional wavelet transform and feature extraction technique in the third section. In the fourth section, we discuss the classification of isolated symbols using MLP architecture of ANN with wavelet features as an input using linear activation function in the neurons of output layer. The revised Universal Approximation theorem of ANN architecture using MLP technique is stated in the fifth section. At the end, the sixth section summarizes the work presented here.

4.2. Character modeling

The dataset used in our experiments consisting of various types of fonts with the size ranging from 10 to 15. The details of the datasets used for the development of OCR for Gujarati symbols followed by the scanning and binarization techniques are discussed in the following two subsections.

4.2.1 Gujarati alphabets:

The Gujarati script consists of 10 numerals, 34 consonants and 5 vowels as middle zone characters, 4 to 5 lower modifiers and more than 10 upper modifiers. In this chapter we have studied the characters from images of laser printed documents which used four fonts of Guj_Regular_SULEKH, Guj_Diamond_SULEKH, Guj_Hastalikhit_SULEKH, Guj_Komal_SULEKH, Gopika, GUJ_Saral_Normal of Microsoft Word with font sizes ranging from 10 to 15.

4.2.2 Scanning and binarization

The character images were selected from binarized images of documents scanned at 300 pixels per inch resolution using the HP-Scan Jet II scanner. The character images were normalized to (32 x 32) array of binary pixels. The Daubechies D4 wavelet transform was applied to the character images and 256 low-low coefficients were used to construct the feature vector. Examples of normalized characters used as input to the neural networks architectures are given in Table 1, 2 ,3 and 4.

Table 1. Gujarati characters

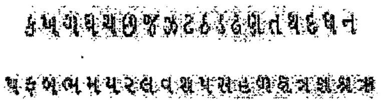
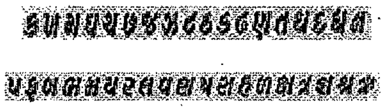
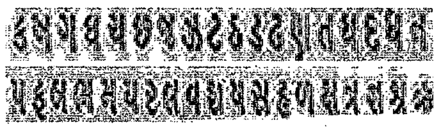
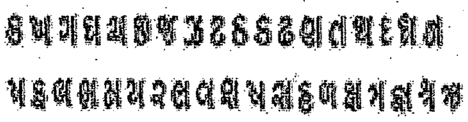
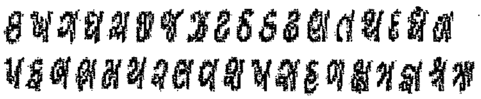
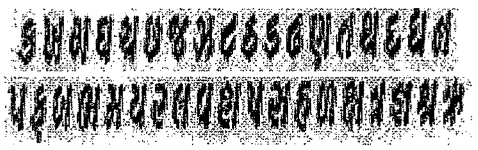
Index	Examples of Middle zone consonants and conjuncts
1	 Guj_Regular_SULEKH with font size 10
2	 Bold and Italic Guj_Diamond_SULEKH with font size 13
3	 Guj_Hastalikhit_SULEKH with font size 15
4	 Bold Guj_Komal_SULEKH with font size 14
5	 Bold and Italic Guj_Komal_SULEKH with font size 15
6	 Bold and Italic Guj_Diamond_SULEKH with font size 15

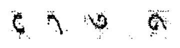
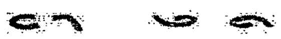
Table 2. Gujarati Numerals

Font	Style size	Character Images
Gopika	(normal)11	૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯
Gopika	(Bold)15	૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯
Guj_Regular_SULEKH	(normal)11	૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯
Guj_Regular_SULEKH	(normal)15	૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯
GUJ_Saral_Normal	(normal)15	૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯
GUJ_Saral_Normal	(Bold)11	૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯

Table 3. Gujarati Vowels

Font	Style size	Character Image
Guj_Regular_SULEKH	(Normal)10	અ ઇ ઈ ઉ ઊ
Guj_Regular_SULEKH	(Normal)15	અ ઇ ઈ ઉ ઊ
Guj_Hastalikhit_SULEKH	(Bold) 11	અ ઇ ઈ ઉ ઊ
Guj_Hastalikhit_SULEKH	(Bold) 15	અ ઇ ઈ ઉ ઊ
Guj_Diamond_SULEKH	(Bold and Italic) 15	અ ઇ ઈ ઉ ઊ

Table 4. Gujarati Lower Modifiers

Font	Style and size	Character Image
Guj_Regular_SULEKH	(Normal)10	
Guj_Regular_SULEKH	(Bolt and Italic)15	
Guj_Hastalikhit_SULEKH	(Bolt and Italic)15	

4.3. Wavelet transforms in Two Dimensions

One-dimensional wavelet transforms introduced in the 1st chapter can easily be extended to two-dimensional functions like images. In order to develop the discrete wavelet transform of images one two-dimensional scaling function $\varphi(x,y)$ and three two-dimensional wavelet functions, $\psi^H(x,y)$, $\psi^V(x,y)$ and $\psi^D(x,y)$, are required. Each is the product of a one-dimensional scaling function φ (section 1.3.2 of chapter-1) and corresponding wavelet function ψ (section 1.3.2 of chapter-1). Excluding products that produce one-dimensional results, like $\varphi(x)\psi(x)$, the four remaining products produce the separable scaling function [13].

$$\varphi(x,y) = \varphi(x)\,\varphi(y) \tag{4.1}$$

And separable, "directionally sensitive" wavelets

$$\psi^H(x,y) = \psi(x)\,\varphi(y) \tag{4.2}$$

$$\psi^V(x,y) = \varphi(x)\,\psi(y) \tag{4.3}$$

$$\psi^D(x,y) = \psi(x)\,\psi(y) \tag{4.4}$$

These wavelets measure functional variations-intensity or gray-level variations for images-along different directions: ψ^H measures variations along columns (for example, horizontal edges), ψ^V responds to variations along rows (like vertical edges), and ψ^D corresponds to variations along diagonals.

Given separable two-dimensional scaling and wavelet functions, extension of the one-dimensional discrete wavelet transform to two dimensions is straightforward. We can define the scaled and translated basis functions:

$$\varphi_{j,m,n}(x,y) = 2^{j/2}\varphi(2^j x - m,\ 2^j y - n) \qquad (4.5)$$

$$\psi^i_{j,m,n}(x,y) = 2^{j/2}\psi^i(2^j x - m,\ 2^j y - n), \qquad i = \{\mathrm{H,V,D}\} \qquad (4.6)$$

where index i identifies the directional wavelets in equations (4.2) to (4.4). Rather than an exponent, i is a superscript that assumes the values H, V and D. The discrete wavelet transform of function $f(x,y)$ of size M x N is then

$$W_\varphi(j_0,m,n) = \frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\varphi_{j_0,m,n}(x,y) \qquad (4.7)$$

$$W^i_\psi(j,m,n) = \frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\psi^i_{j,m,n}(x,y) \qquad i = \{\mathrm{H,V,D}\} \qquad (4.8)$$

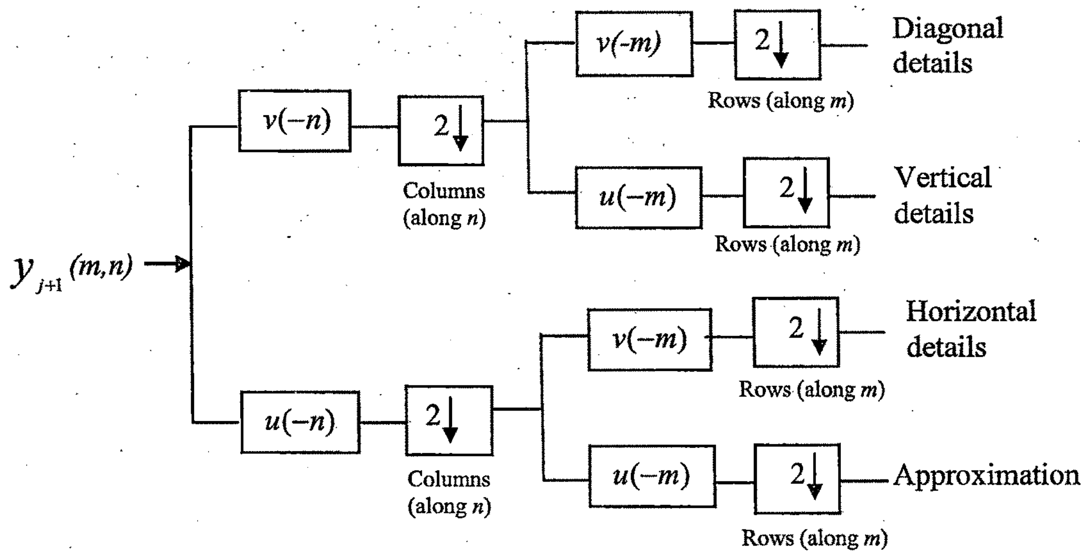
As in the one dimensional case, j_0 is an arbitrary starting scale and the $W_\varphi(j_0,m,n)$ coefficients define an approximation of $f(x,y)$ at scale j_0. The $W^i_\psi(j,m,n)$

coefficients add horizontal, vertical, and diagonal details for scales $j \geq j_0$. We normally let $j_0 = 0$ and select $N = M = 2^J$, J is a positive constant, so that $j = 0, 1, 2 \ldots, J\text{-}1$ and m, $n = 0, 1, 2 \ldots, 2^J\text{-}1$. Given the W_φ and W^i_ψ of equations (4.7) and (4.8), $f(x, y)$ is obtained via the inverse discrete wavelet transform

$$f(x,y) = \frac{1}{\sqrt{MN}}\sum_m\sum_n W_\varphi(j_0,m,n)\varphi_{j_0,m,n}(x,y) + \frac{1}{\sqrt{MN}}\sum_{i=H,V,D}\sum_{j=j_0}^{\infty}\sum_m\sum_n W^i_\psi(j,m,n)\psi^i_{j,m,n}(x,y$$

4.3.1. Wavelet transform as a feature extractor

Like the one-dimensional discrete wavelet transform, the two-dimensional discrete wavelet transform can be implemented using digital filters and downsamplers. As in the one-dimensional case, image *f(x, y)* is used as the y_{j+1} (m, n) input. Convolving its rows with *v(-n)* and *u(-n)* and downsampling its columns, we get two subimages whose horizontal resolutions are reduced by a factor of 2. The highpass or detail component characterizes the image's high-frequency information with vertical orientation. Both subimages are then filtered columnwise and downsampled to yield four quarter-size output subimages viz. diagonal (High-High), vertical (High-Low), horizontal (Low-High) detail components and one approximation components (Low-Low). This process is known as first level decomposition. These images are the inner products of *f(x, y)* and the two-dimensional scaling and wavelet functions in equations (4.1) to (4.4), followed by downsampling by two in each dimension.



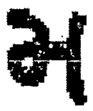
[Fig. 3.1 Image Decomposition using DWT.]

Figure (3.1) shows the process of computing Discrete Wavelet Transform of a function *f*(x,y) of size M x N. Let $y_{j+1}(m,n)$ be the $(j+1)^{st}$ level of resolution of f where $0 \quad m < M$ and $0 \quad n < N$. The process starts by convolving y_{j+1} with high pass or detail components v (column wise, along n) and then it is down sampled. Same process is repeated for low pass or approximation components, u. The resultant outputs are further convolved with v and u respectively (row wise, along m) and down sampled. These computations decomposes the given two dimensional array into four parts, viz. diagonal (High-High), vertical (High-Low), horizontal (Low-High) detail components and one approximation components (Low-Low). That is known as first level decomposition.

In our experiments of character recognition of Gujarati script, the procedure of feature extraction of a normalized image of the size 32 x 32 can be summarized as below:

1. Input : The image is binarized in to 32 x 32 matrix. This binarized matrix is given as an input to the algorithm.
2. 2. Convolution: Convolve this binarized image with father wavelet v and mother wavelet u followed by down- sampling by a factor of 2 twice. The image is now divided in to four subbands as discussed in the next step.
3. The four subbands (2^2) parts viz. low-low, low-high, high-low, high-high coefficients as shown in figure 3. Each part is of the size (16 x 16).
4. Consider only low-low (approximation) coefficients which captures the core information of the image.
5. These coefficients are considered as an input to the recognizer (Neural Network architecture).

As an example, consider the figure 3.2 which shows an image of Gujarati Numeral /m and Fig. 3.3(a) gives binary representation of the image after scaling it to the size 32 x 32. Figure 3.3(b) shows binary image obtained by thresholding Low-Low coefficients of Daubechies D6 wavelet transform after first level of decomposition.



[Fig. 3.2 Image of Gujarati /m]

```
11000000001111111111111111111111   1111111111111100
11000000001111111111111111111111   1111111111111111
00000000001111111111111111100111   1000011111111111
00000000000001111111111111100000   0000001111111100
00000000000000011111111111000000   1111100011111000
00000000000000001111111111000000   1111110001111001
11100000000000000111111111000001   1111110001111001
11111111111000000011111111000001   1111000001000001
11111111111100000011111111000001   1000000000000001
11111111111100000011111111000011   0000110000000001
11111111111000000011111100000011   0001110011110001
11111100000000000000000000000011   1001110011110001
11100000000000000100001000000011   1100000011110001
11000000000000000000000000000011   1110000111110001
00000000000000000000000000000011   1111111111110000
00000000000000000000000000000011   1111111111111000
00000001110000000000000000000011
00000001110000000000000000000011
00000011111000000001111110000011
00000011111000000111111110000011
00000000110000000111111110000011
11000001000000001111111100000011
11000000000000001111111100000011
11100000000000011111111100000011
11110000000000011111111100000001
11111100000001111111111100000001
11111111111111111111111100000001
11111111111111111111111100000001
11111111111111111111111111000001
11111111111111111111111111000000
11111111111111111111111111100000
11111111111111111111111111110001
```

[Fig.3.3. (a) 32x32 Original binarized image (b) 16x16 wavelet-compressed image]

4.4. Classification of Gujarati symbols using MLP networks

In this section, we discuss the use of Neural Network classifiers for the recognition of printed Gujarati alphabets where extracted features using wavelets are supplied as an input vector. This section is divided in to three parts viz. first part discusses the change in weight updation rule in back propagation algorithm of MLP due to the consideration of linear transfer function in output layer of MLP instead of nonlinear one, second part demonstrates the applicability of this approach using XOR problem. In the third subsection, we present the experimental results obtained after applying this approach of MLP classifier to the middle and lower zone characters of the Gujarati script.

4.4.1. Multilayer Perceptron network with linearity in the output layer:

Multi Layer Perceptron (MLP) is the most commonly used of the many ANN architectures used for Pattern Recognition tasks. And in our experiment we have used linear activation function at the output layer for the sake of faster convergence. Following properties of MLP make it suitable as a classifier for the development of Optical Character Recognition (OCR) system for the Gujarati script.

- The model of each hidden neuron in the network includes a nonlinear activation function. The important point to emphasize here is that the non-linearity is smooth (i.e., differentiable everywhere).
- The network contains one or more layers of hidden neurons that are not part of the input or output of the network. These hidden neurons enable the network to learn complex tasks by extracting progressively more meaningful features from the input patterns (vectors).
- The network exhibits a high degree of connectivity, determined by the synapses of the network. A change in the connectivity of the network requires a change in the population of synaptic connections or their weights.

The MLP used in this study has 1 input layer followed by 1 or more hidden layers and 1 output layer. Each layer is made up of units called neurons. A neuron in the input layer simply stands for a gateway to provide input to the network and does no computation. A neuron in the hidden layer (e.g. j^{th} neuron) is a computing unit which sums (s_j) its weighted (w_{ji}) inputs (x_i) from the previous layer and then applies the sigmoidal/logistic activation function, say *F,* on the weighted sum and a neuron in the output layer is again a computing unit which sums its weights with the output of the previous layer and then applies the linear activation function as shown below :

$$s_j = \sum_{i=1}^{p} w_{ij} x_i \tag{4.9}$$

$$F(s_j) = 1/\left(1 + e^{-\lambda s_j}\right) \tag{4.10}$$

where, λ is the parameter relating to the shape of the sigmoidal function and *p* is the number of neurons in the previous layer.

The error signal at the output layer can be described at j^{th} neuron in the output layer for n^{th} training example by

$$e_j(n) = d_j(n) - y_j(n) \tag{4.11}$$

where *dj(n)* corresponds to desired response and *yj(n)* corresponds to the computed response of j^{th} neuron of the output layer

i.e. $y_j(n) = \phi(s_j(n))$

ϕ is an activation function at output layer which is considered to be a linear function instead of the traditional nonlinear function, i.e. $\phi(x) = x$ and let $\xi(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n)$ be a Sum of Squared Error (SSE) where set *C* includes all the neurons in the output layer and $e_j(n)$ as described in equation (4.11).

Now, considering $\delta_j(n)$ to be the gradient of the error function $\xi(n)$ quantity at the output neuron *j*. Then it can be described by the standard gradient descent approach

$$\delta_j(n) = -\frac{\partial\xi(n)}{\partial(s_j(n))}$$

$$= -\frac{\partial\xi(n)}{\partial(e_j(n))}\frac{\partial e_j(n)}{\partial(y_j(n))}\frac{\partial y_j(n)}{\partial(s_j(n))}$$

$$= e_j(n)$$

$$\therefore \quad \delta_j(n) = e_j(n) \tag{4.12}$$

Therefore the gradient quantity for the j^{th} neuron at hidden layer will certainly take only the error signal as a part of a gradient of output layer.

The MLP is fully connected in the sense that every neuron in the current layer is connected to every neuron in the next layer by a weighted link through which the state of the neuron is transmitted. The weights on the links constitute the information stored by the network.

The training of the network for reflecting a set of input-output patterns is a process for adjusting the weights on the network connections such that the resulting network captures the desired input-output behaviour.

The Backpropagation algorithm is used to train the network, i.e, to adjust the weights on the links for given training patterns by propagating the effect of error signal observed at the output nodes to the links in the output layer and the hidden layer with the help of linearity at output layer as defined by equation (4.12) and non linearity at hidden layer. The updation of weights is done after exposing each training pattern to the network. The training process is continued repeatedly on the training patterns until the sum of squares of error (the absolute difference between the desired and computed values) of the output nodes falls below a predetermined small value (tolerance).

4.4.2. Numerical experiments using linearity in output layer in MLP

First of all this approach of linearity in the output layer is applied to the standard XOR problem with 2 neurons in the input layer and 1 neuron in the output layer. Several experiments are carried out by changing the number of neurons in the hidden layer. Table 4.2 depicts the results of the experiments.

Table-5 classification of XOR

Neurons in the Hidden layer	Learning Rate	Iterations	Tolerance
1	0.1	Does not converge	0.0001
2	0.1	13000	0.0001
3	0.1	5000	0.0001
4	0.1	3310	0.0001

Using linear transfer function in the neurons of the output layer also we have achieved the desired accuracy. The results obtained in table 5 have encouraged us to use this approach in the development of OCR system for the Gujarati script.

4.4.3. Recognition of Gujarati characters using MLP

In this subsection, we employ the most widely used Artificial Neural Networks i.e. Multilayer Perceptron architecture. It is divided in two parts. The first part shows the experiments of identifying symbols of Gujarati numerals using Matlab 6.1 software [3]. The results are found to be encouraging for the numerals. Therefore, we have implemented our own MLP architecture using Java classes. The experiments are carried out further for the large set of middle zone and lower zone symbols of Gujarati script using these Java classes in the second part. Two separate networks are constructed for both the zones. Traditional MLP architecture contains nonlinear transfer function for hidden and output layers both while we have taken nonlinear transfer function for the hidden layer and linear transfer function for the output layer to reduce the complexities. The details are given in the second part of this subsection.

(a) Classification of Gujarati numerals [3]:

The MLP network used in our study is a two-layer feedforward network with nonlinear sigmoidal activation functions. Several experiments with various number of hidden units in the network were carried out and the network with 12 hidden units had been found to yield good recognition in reasonable time. Here 256 wavelets coefficients (out of a total of 1024) are provided as input to the network and the output layer contained 10 neurons, each signifying a single Gujarati digit. The network was trained using the EBP algorithm as described in Section 4. We have carried out the experiments in two ways. Firstly we used Matlab v.6.1 (newff() function) and the network was trained until sum square error(SSE) between the network output and desired output falls below 10^{-8}. The results are summarized in the table 6 below :

Table 6. Result of numerals using MLP network

Input of the RBFN	hidden units	Software	Performance on testing set (%)
256 Daubechies coefficients	12	Matlab v.6.1	93.75

(b) Classification of the middle zone and lower zone symbols of Gujarati script [4]:

As shown in the tables of section 4.2, there are large varieties of middle zone symbols, which consist of various numerals, vowels and consonants in the Gujarati script. Moreover, the script contains large varieties of conjuncts, formed by the various combinations of consonants and vowels, as shown in the chapter 2.

The data base for recognition of glyphs made up of 2986 images of the 52 types (10 numerals, 34 consonants, 3 frequently used conjuncts and 5 vowels) of middle zone symbols and 210 images of 4 types of lower zone modifiers. The images are first normalized to a standard size of 32 x 32 pixels. Then the Daubechies D4 discrete wavelet transform is applied on each image. The 16 x 16 low-low coefficients that result from this transform are then extracted as the features of the images capturing their important characteristics. These features are then exposed to the two MLP networks for training and testing the networks. The details of these computations are provided below:

In the experiments presented here, we have constructed two networks: the first network is for middle zone symbols and the second network is for the lower zone symbols of Gujarati script (section 2.4). Each of these networks has 256 input neurons and as many output neurons as the number of classes (types) of the respective zone.

An implementation of the MLP architecture that can be used to realize these two networks has been developed as a java classes. The main class MLP has three attributes which are object of three classes 1. InputLayer, 2. HiddenLayer and 3. OutputLayer. Each of the three Layer classes an arry of objects of an appropriate Neuron class 1. InputNeuron, 2. HiddenNeuron, 3. OutputNeuron respectively. All these neuron classes are sub-classes of a general class Neuron class. The links among all these layers are generated with the help of Synapse class. The synaptic weighs of the links are updated using this class. The InputLayer class has attributes for the array of input neurons and input patterns etc. and methods like setInputPatterns, setSynapse etc. The corresponding InputNeuron class has attributes for input patterns, testing patterns etc and methods for setting input and testing patterns and for computing the outputs. While the HiddenLayer class has attributes for hidden neurons, input synapse weight, output synapse weight etc and methods for computing forward pass, backward pass, randomizing the weights etc. The corresponding hidden neuron class has attributes input synapse, output synapse etc and methods for computing gradients. In the same way the OutputLayer class has attributes computed output, desired output etc and the methods for getting computed output for each neuron, setting gradients for each neuron etc. The corresponding output neuron class has attributes computed output, desired output etc and methods for computing computed output, gradients etc.

Several experiments with various numbers of hidden units were carried out to determine the optimal number of neurons in the hidden layer of both the networks. First network exhibited good recognition with 50 hidden units. The network has 256 neurons in its input layer and the output layer was made up of 52 neurons (each neuron corresponding to one of the 52 characters to be recognized). The network is trained using the Back-propagation algorithm as described in the section 4 of the chapter 1. The training of the first network was carried out until the SSE became less than 0.089. Training has taken 5 hours to achieve this accuracy.

In the case of lower modifiers, 4 frequently used modifiers are accommodated in the database of the second network and a total of 210 patterns are generated with the various fonts and sizes as mentioned in the second section. Again each of these are normalized in to 32 x 32 binary values and 256 low-low coefficients are considered as a feature vector. 74 of the images out of the total 210 are used as the training set and the remaining 136 are used as a testing set. This network is trained with 20 hidden neurons and the training was stopped when Sum Squared Error (SSE) reached just below 0.01.

The recognition accuracies of both the networks (middle zone and lower zone) are shown in the table 7. Overall (average) recognition accuracy of 94.69% was achieved (1050 characters from test set were identified correctly out of 1111 characters in total) with both the networks.

Table 7. Results of middle and lower zones using MLP networks

Network	**Input of the network**	**Performance in percentage**
Network 1 (Middle zone symbols)	256 Wavelet coefficients	94.46
Network 2 (Lower zone symbols)	256 Wavelet coefficients	96.32

4.5. Verification of Universal approximation theorem for MLP

The Universal approximation theorem, discussed in the chapter 1, says that only one hidden layer with nonlinear transfer function followed by an output layer with linear transfer function is sufficient for the approximation of any continuous function. The approach of using nonlinear transfer functions in one or more hidden layers and the output layer is very common in the literature. There are two consequences of this approach as below:

- Those architectures are more complex than the one considered in the Universal Approximation Theorem
- They are computationally costlier than the architecture with the linear transfer functions in output layer

The experiments shown in the subsection 4.4.3(b) use an MLP architecture exactly similar to the one considered in the Universal Approximation Theorem. The architectures used in the above experiments consisting of only one hidden layer with nonlinear transfer functions followed by an output layer with the linear transfer functions. Thus the architectures follow the statement of universal approximation theorem and become less computationally costly for the large datasets. The high classification accuracies achieved in the experiments can be considered as a numerical justification for the simpler architecture.

4.6. Conclusion and Summary

The major aspect of this chapter is to use linear activation function in the neurons of output layer of multilayer perceptron. Section 4.4 demonstrates the encouraging results in both the zones (middle and lower zones) of Gujarati OCR using this approach.

As shown in the sections 4.4 and 4.5, this approach provides good recognition accuracy with less complex architecture of MLP and adhere the statement of Universal approximation theorem. The results of the classification of Gujarati symbols using this approach is discussed below:

This chapter discusses the usage of wavelets and ANN for recognizing printed Gujarati characters as a part of OCR. Here wavelets are used as a feature extractor and Artificial Neural Networks are used as a classifier. This is an extension to the first recorded attempt [12], of using wavelet feature vectors and Artificial Neural Network classifiers for recognition of printed Gujarati characters. The only other published account of Gujarati OCR [7] used statistical techniques for recognition of a small subset of 10 Gujarati characters and reported a maximum recognition accuracy of 67%. The current work indicates the possibility of achieving recognition accuracy required to produce a commercial OCR. These results are encouraging and similar results may be expected for the complete character set of Gujarati script.

theorem. The results of the classification of Gujarati symbols using this approach is discussed below:

This chapter discusses the usage of wavelets and ANN for recognizing printed Gujarati characters as a part of OCR. Here wavelets are used as a feature extractor and Artificial Neural Networks are used as a classifier. This is an extension to the first recorded attempt [12], of using wavelet feature vectors and Artificial Neural Network classifiers for recognition of printed Gujarati characters. The only other published account of Gujarati OCR [7] used statistical techniques for recognition of a small subset of 10 Gujarati characters and reported a maximum recognition accuracy of 67%. The current work indicates the possibility of achieving recognition accuracy required to produce a commercial OCR. These results are encouraging and similar results may be expected for the complete character set of Gujarati script.