Chapter-4

Modified Automatic Script Identification System

4.1 Introduction

In the previous chapter a modular neural network based automatic script identification system is presented. The system has several disadvantages. To increase the number of script classes, individually trained neural classifiers should be added to the system. This increases the time required to build the system. Even if the database of training patterns increases, the time to build the system multiplies. To overcome these drawbacks, other possibilities of constructing more efficient systems are explored in this chapter.

The objectives of the improvements for basic script identification system can be presented as follows.

(i) **Reduction in the time required to build the system.** This can be achieved by improvement in the classifiers. The feedforward neural network based classifiers with backpropagation training algorithm used in the previous system are the source of increased design time. Better neural network classifiers can be utilized in the construction of the system.

(ii) **Increase in the number of document images for each class.** This increases the training and testing database for the system. The increased database assures the critical test for the developed system.

(iii) **Increase in the number of classes.** The previous system was designed only to classify three scripts. Increase in the number of classes increases the complexity of the system. The system should overcome this and perform better in such cases also. Another effect of increasing the number of classes is increase in the database.

(iv) **Improvement in feature extraction.** The previous system utilized only five features. The number of features can be increased for better accuracy.

---

Some parts of the material in this chapter appear in the following paper:
i. *Neural network based script identification system for Indian documents: A New Approach*, 90\textsuperscript{th} Indian Science Congress -2003, January 3-7, Bangalore, India.
The above objectives are accomplished in the modified script identification system. The organization of this chapter is as follows. In section 4.2, datasets and database development techniques are discussed. The method to increase the number of features which are used for classification is dealt with in detail in section 4.3. The detailed study, design procedure and algorithms of radial basis neural network classifiers and probabilistic neural network classifiers are explained in section 4.4. Systems developed based on these classifiers are discussed in section 4.5. The details of experiments conducted on these systems, results obtained are described in section 4.6. The discussions about the results are made in section 4.7. The conclusions are presented in section 4.8.

4.2 Database development

A total of 270 typed documents of size 64 x 64 pixels, 30 each in English, Hindi, Kannada, Tamil, Gujarati, Malayalam, Oriya, Telugu, and Punjabi scripts, are prepared in the corresponding Shreelipi (Indian language word processor) fonts of size 11, other than English for which Times New Roman font is used. The reasons for choosing document of size 64 x 64 pixels were already mentioned in section 5 of third chapter.

Figure 4.1: Sample document images in nine language scripts.
Documents are written in MS-Word and then ported into MS-Paint program. In MS-Paint, the documents are manually divided into 64 x 64 pixels. All the document images are converted such that they have only two tones or levels either '0' or '1'. Sample documents are shown in figure 4.1.

These 270 documents form the basic dataset. Using this set, another three sets are formed by introducing random noise of increasing order. The second set is formed by introduction of 20 noisy pixels in each document image of the first set. Similarly the third and fourth sets are formed by introducing 40 and 60 noisy pixels in each document of the basic set correspondingly. The noisy images of a single document image are shown in figure 4.2. Thus there are 270 x 4 = 1080 document images in the database. The detailed algorithm is described below.

Algorithm 4.1: Noise introduction in document images

1. Read the input document image $K$ from the database. Let the number of rows be $m$ and number of columns be $n$.
2. Choose the noise level (e.g., as 20 pixels or 40 pixels or 60 pixels). Fix it to variable named $\text{noise\_level}$.
3. Make $i = 1$.
4. Choose a first random number ($x$) between 1 and $m$.
5. Choose a second random number ($y$) between 1 and $n$.
6. Check the pixel $K(x, y)$. If it is equal to '0' convert it to '1' and if it is equal to '1' convert it to '0'.

Figure 4.2: Effect of introducing noise in document image (a) original image, (b) 20 pixels noisy image, (c) 40 pixels noisy image, (d) 60 pixels noisy image.
7. Make \( i = i + 1 \).
8. If \( i \) is less than or equal to the noise level (the pre-defined number of noisy pixels) go to step 4.
9. The final \( K \) is a noisy image. Repeat the above steps for all the document patterns in the basic dataset.

### 4.3 Modified feature extraction

Feature extraction is an integral part of any recognition system. The aim of feature extraction is to describe the pattern by means of the minimum number of features or attributes that are effective in discriminating pattern classes. In the presently considered problem of classification of English, Hindi, Kannada, Gujarati, Tamil, Malayalam, Oriya, Punjabi, and Telugu document image patterns, black pixel distribution in each script can effectively be used as a potential feature. Pixel distribution is a characteristic of a script and if it is considered along some specific directions it can discriminate different pattern classes.

A feature extraction method is used to reduce the bitmap image of a sample pattern corresponding to pixel distribution into a vector of real numbers required for classification. To produce the feature vector for an input document pattern we adopted a two-stage procedure as follows. In the first stage, each of the documents is morphologically dilated in the horizontal, vertical, right diagonal and left diagonal directions, using 3 x 3 masks [43] (as mentioned in the second chapter - Feature extraction). The results of using this method on nine document samples, one each in nine scripts, are shown in figure 4.3 (a).

In the next stage, these four modified versions of the image and the original are used for feature extraction. Each of these images is subjected to the bar mask encoder shown in figure 4.3 (b). The number of pixels in each of these resulting regions is counted. A single feature value is given by the number of marked bits (pixels) divided by the total number of pixels in the corresponding region of each image. The total number of pixels in the image is a constant value (i.e., 64 x 64 = 4096). Thus a set of fifty (ten features for each version of the image) feature values is obtained for every document image sample. The summary of this procedure is detailed in algorithm 4.2.
Figure 4.3(a): The original document images and their modified version.

Figure 4.3(b): Bar mask encoder. H1, H2, H3, H4, H5 are horizontal capture regions and V1, V2, V3, V4 and V5 are vertical capture regions.
**Algorithm 4.2: Modified feature extraction**

1. Read the input document image of 64 x 64 pixels into the matrix $E$ and threshold it to the two levels.

2. Consider the structuring elements of 3 x 3 pixels in the horizontal, vertical, left diagonal and right diagonal directions as follows.

   
   $SE = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

   $SE_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

   
   $SE_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

   $SE_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

3. Prepare the new matrices $E_1$, $E_2$, $E_3$, and $E_4$ which are of the same sizes as that of $E$, by performing the operation dilation (algorithm 2.3) on $E$, by $SE_1$, $SE_2$, $SE_3$, and $SE_4$ respectively.

4. Consider the $E$, $E_1$, $E_2$, $E_3$ and $E_4$ individually. Apply the bar mask encoding method for each image of these modified images. That is dividing each image into five horizontal regions and five vertical regions (total of 10 regions, as shown in figure 4.3b).

5. Consider at first the bar regions of $E$. Count the number of '1's in each of the regions and divide it by the total number of pixels in that region. This process results in feature values for f1 to f10. Similarly obtain the values of features f1 to f10 for each of $E_1$, $E_2$, $E_3$, and $E_4$. Thus this process yields 50 features for a single document pattern.

6. Repeat the steps 1 to 5 for all the patterns in the document database.

7. Normalize the resultant features in the range -1 and +1.

**4.4 Classifiers**

There are various methods for pattern classification problems and each of them has its own pros and cons. Table 4.1 presents a summary of the evaluation of the representative methods [46] based on training time, hardware requirement and classification time. Here recognition rate is excluded from the listed criteria because it is difficult to evaluate.
Training time means the time required to develop the classifier by training or statistical calculation after it is designed. Hardware requirement generally means the size of the computer memory.

Selecting the classification method depends on the given problem and other requirements of the problem domain. According to table 4.1 the radial basis function network (RBF) ranks in the middle for all criteria. Therefore the RBF network could be a reasonable choice for those classification problems which do not have any particular requirements.

<table>
<thead>
<tr>
<th>Serial Number</th>
<th>Classifiers</th>
<th>Training time</th>
<th>Hardware Requirement</th>
<th>Classification time</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Radial basis function</td>
<td>Middle</td>
<td>Middle</td>
<td>Middle</td>
</tr>
<tr>
<td>02</td>
<td>Back propagation</td>
<td>Long</td>
<td>Small</td>
<td>Short</td>
</tr>
<tr>
<td>03</td>
<td>K-nearest neighbour</td>
<td>None</td>
<td>Large</td>
<td>Long</td>
</tr>
<tr>
<td>04</td>
<td>Learning vector quantization</td>
<td>Long</td>
<td>Small</td>
<td>Short</td>
</tr>
<tr>
<td>05</td>
<td>Minimum distance classifier</td>
<td>Short</td>
<td>Small</td>
<td>Short</td>
</tr>
</tbody>
</table>

The architecture and the training methods of an RBF network are well known [47], [48], [49], [50], [51], [10], [52]. However most of the training algorithms are for function approximation. The latest literature on the training algorithm of RBF networks in problems of pattern classification are given in [53] and [46]. The design procedure of RBF networks as given by Young-Sup Hwang [46] is explained in the following section.

4.4.1 *Radial basis function neural network*

In the nervous systems of biological organisms there is evidence of neurons whose response characteristics are "local" or "tuned" to some region of input space. An example is the orientation-sensitive cells of the visual cortex, whose response is sensitive to local regions in the retina [54].
In the context of neural networks, hidden units provide a set of "functions" that constitute an arbitrary "basis" for the input patterns (vectors) when they are expanded into the hidden space; these functions are called Radial-basis functions (RBF) [12]. The radial-basis function network, in its most basic form, involves three layers with entirely different roles. The input layer is made up of source nodes (sensory units) that connect the network to its environment. The second layer, the only hidden layer in the network, applies a nonlinear transformation from the input space to the hidden space; in most applications the hidden space is of high dimensionality. The output layer is linear, supplying the response of the network to the activation pattern applied to the input layer.

The design procedure of the radial basis function network can be explained in two stages. In the first stage a single RBF neuron model is described. In the next stage design procedure is explained.

4.4.1a Single RBF neuron model

The diagrammatic representation of the single radial basis function neuron with r inputs is shown in the figure 4.4.

The radial basis neuron is different from that of neurons of other feedforward networks used in the previous chapter. The main difference is that the input to the radial basis transfer function is the vector distance between its weight vector \( w \) and the input vector \( p \) (\( f_1, f_2, \ldots, f_t \)), multiplied by the bias \( b \). The \( \| \text{dist} \| \) box in the figure 4.4 accepts the input vector \( p \) and the single row input weight matrix, and produces the dot product of the two.
The transfer function for a radial basis neuron is:

\[ \text{Radbas}(n) = \exp(-n^2) \]  

(4.1)

The plot of the Radbas transfer function is shown in figure 4.5. The radial basis function has a maximum of 1 when its input is 0. As the distance between \( w \) and \( p \) decreases, the output increases. Thus a radial basis neuron acts as a detector which produces 1 whenever the input \( p \) is identical to its weight vector. The bias \( b \) allows the sensitivity of the radial basis neuron to be adjusted. For example, if a neuron had a bias of 0.1 it would output 0.5 for any input vector \( p \) at vector distance of 8.326 (0.8326 / \( b \)) from its weight vector \( w \).

4.4.1b Design procedure of RBF network

The design procedure of a radial basis network is shown in figure 4.6. The adopted design is very simple and is as follows.

(i) Data: Let \( p = [p_1, p_2, p_3, ..., p_q] \) where \( p_1, p_2, p_3, ..., p_q \) are individual patterns. Each pattern \( p \) is a column feature vector of size \( r \) (i.e. \( p_i = [f_1, f_2, f_3, ..., f_r]^T \)). Thus the size of \( p \) is \( [r \times q] \). If \( q \) number of patterns are available for design, then there must be \( q \) number of outputs. If \( s \) is the number of classes (the number of classes among which training patterns to be classified). Then the output for each pattern should have \( s \) nodes, among which one at a time will be selected. Therefore for \( q \) number of patterns, size of output vector is \( [s \times q] \).
(ii) Construction of input and output layer nodes: By the above data and since size of input / training vector $\mathbf{p}$ is $[r \times q]$, the input layer should have $r$ input nodes (number of features). Since the target vector $\mathbf{t}$ is $[s \times q]$, the output layer should have $s$ nodes.

(iii) Fixing of RBF units (construction of hidden layer): There are many methods to create hidden layer neurons. One of the simplest and most general ways to choose the number of hidden layer neurons is to create a neuron for each training pattern. In the present design, this method is adopted. Hence the number of hidden nodes can be fixed, in this case as $q$ (because $q$ no of patterns exists).
(iv) **Fixing of weights from input layer to hidden layer and calculation of outputs of radial basis neurons:** The output of hidden layer, radial basis neuron is given by

\[ a_1 = \exp(-n \cdot n) \]  

(4.2)

where \( n \) is a product of distance \( dist \) and bias \( b_1 \) i.e. \( n = dist \cdot b_1 \), and \( dist = \|w1 - p\| \). Here \( \| . \| \) denotes Euclidean distance.

The radial basis function has a maximum '1' when its input is '0'. As the distance between \( w \) and \( p \) decreases the output increases.

If \( a_1 \) is an output from neuron \( i \), then distance \( dist \) gives the column vector of distances from the input pattern \( p_i \) to the patterns stored in each radial basis neuron.

The weights for the input layer are set to the transpose of the matrix formed from total number of training pairs, \( p \) (i.e. \( w1 = \) transpose of \( p \)). The patterns stored in all the radial basis neurons are available in \( w1 \). Therefore to find the output of the radial basis neuron we should first compute, \( dist = \|w1 - p\| \) which is a Euclidean distance. The term \( dist \) should be multiplied element by element with the bias \( b_1 \) of radial basis layer.

(v) **Fixing the biases of radial basis neurons:** In the method of design adopted, there will be one bias for each radial basis neuron. These biases can be set to \( b_1 = \text{ones}(q, 1) \cdot \sqrt{-\log(0.5)} / \text{spread} \), where \( \text{spread} \) can be specified (or varied) as a design parameter. The term \( \sqrt{-\log(0.5)} \) amounts to value 0.8326. The significance of fixing this term is, in the function \( a_1 = \exp(-n^2) \), if bias is set to 0.8326 / \( \text{spread} \), it gives radial basis functions that cross 0.5 at weighted outputs of +/- \( \text{spread} \). Thus it determines the width of area in the input space for which each neuron responds.

(vi) **Fixing the weights from hidden layer to output layer and calculation of outputs:** The number of nodes in the hidden layer is known (\( q \)) as well as those in the output layer (\( s \) classes). Each hidden layer neuron is connected to each output node. The layer is linear. Therefore the weights and biases are found by simulating the first layer outputs \( a1 \) and then solving the linear matrix expression, \([w2, b2] [a1; \text{ones}] = t\). In this expression, \( a1 \) (outputs of first layer) and target \( t \) are known. Therefore the weights and biases of the second layer can be found by \( w, b = t / [p; \text{ones}(1, q)] \). When the weights and biases of the second layer are known, the outputs of the linear layer can be given by the equation \( a2 = w2 \cdot a1 + (b2 \cdot \text{ones}(1, q)) \).
Algorithm 4.3: Radial basis neural network algorithm.

1. Let \([p, t]\) be the training vectors and targets, \(p\) is of size \(r \times q\) (it consists of \(r\) features of all \(q\) training patterns) and \(t\) is of size \(s \times q\) (it consists of targets for all patterns), \(r\) is the size of the feature vector (in the experiment it is 50), \(s\) is the number of classes (In experiments \(s\) is 9).

2. Since the feature vector size is \(r\), construct the input layer with \(r\) input nodes. The number of hidden layer nodes is equal to the number of training patterns. Hence fix the hidden layer nodes as \(q\). The output layer nodes should be equal to the number of classes. Thus the output layer should have \(s\) nodes.

3. Fix the weights \((w_1)\) of the first layer as transpose of input pattern vectors \((p')\).

4. Fix the first layer biases \((b_1)\) to 0.8326 / spread. The spread is a parameter of the design of the network and is given.

5. Find the outputs \(a_1\) of the first layer using the expression \(a_1 = \exp(-n \cdot n)\) where \(n = \|w_1 - p\| \cdot b_1\). That is \(\|\cdot\|\) indicates Euclidean distance between \(w_1\) and \(p\).

6. Calculate the second layer weights \(w_2\) and biases \(b_2\) by using the following expression, 
\[ [w_2, b_2] \cdot [a_1; \text{ones}] = t. \]

7. Calculate the second layer outputs by the following expression. Since the layer is linear it is given by the equation \(a_2 = w_2 \cdot a_1 + (b_2 \cdot \text{ones} (1, q))\).

Validation of the constructed network:

8. Let \(p_1\) be the testing vectors. Calculate the first layer outputs by calculating 
\(a_1 = \exp(-n \cdot n)\) where \(n = \|w_1 - p\| \cdot b_1\) which is the product of the Euclidean distance \(\|\|\) and bias \(b_1\).

9. Using obtained output of the first layer \(a_1\), and weights and biases of the second layer calculate the outputs of the second layer.

4.4.2 Probabilistic neural network

A probabilistic neural network is a two-layered structure. The first layer is a radial basis layer and the second is a competitive layer. The first layer computes the distances from the input vector to the training input vectors and produces a vector whose elements indicate how close the input is to a training input. The second layer sums these contributions for each class of inputs to produce as its net output, a vector of probabilities.
The maximum of these probabilities is considered and the class for which it belongs is selected.

4.4.2a Design procedure
The inputs to the radial basis layer are the outputs obtained from feature extractor module. (In the reported experiments it is a vector of size 50.) This layer consists of radial basis neurons equal to the number of training patterns (in our experiments it is 225). The weights for this layer are set to the transpose of the matrix formed from the total number of training pairs. The net input to the radial basis neurons is the vector distance between its weight vector $w$ and the input vector $p$, multiplied by bias $b$.

The output $Y$ of a radial basis neuron is given by the function,

$$ Y = \exp \left( -n^2 \right) \quad (4.3) $$

where $n = ||w - p||$, $b$ and $||.||$ denotes Euclidean distance. The radial basis function has a maximum of '1' when its input is 0. As the distance between $w$ and $p$ decreases the output increases.

Each bias in the first layer is set to the $\sqrt{(-\log(0.5))/\text{spread}}$ or $0.8326/\text{spread}$. This gives the radial basis functions that cross 0.5 at weighted input of $+/\text{spread}$. This determines the width of an area in the input space to which each neuron responds. The bias $b$ allows the sensitivity of the radial basis neuron to be adjusted. For example, if a neuron has a bias 0.1 it outputs 0.5 for any input vector $p$ at vector distance of 8.326 ($0.8326/b$) from its weight vector $w$. A larger spread leads to a large area around the input vector where the radial basis neurons respond with significant outputs. Therefore if spread is small the radial basis function is very steep so that the neuron with the weight vector closest to the input has a much larger output than other neurons. The network tends to respond with the target vector associated with the nearest design vector.

Competitive layer receives net input (column) vector obtained from the radial basis neurons. The number of neurons in this layer is equal to the number of classes $k$. The weights are set to the matrix $T$ of target vectors. Each vector has a '1' only in the row associated with that particular class of input and zeroes elsewhere. The multiplication of the matrix $T$ and $Y$ (column vector), sums the elements of $Y$ due to each of the $k$ input
classes. From this result, the second layer competitively produces '1' corresponding to the largest element and zeroes elsewhere. Thus the network has classified the input vector into a specific one of \(k\) classes because that class had the maximum probability being correct. The summary is given in method 4.4.

**Algorithm 4.4: Probabilistic neural network algorithm**

1. Let \([p, t]\) be the training vectors and targets, \(p\) is of the size \(r \times q\) (It consists of \(r\) features of all \(q\) training patterns) and \(t\) is of the size \(s \times q\) (it consists of targets for all patterns), \(r\) is the size of feature vector (in experimented case it is 50), \(s\) is number of classes (here it is 9).

2. Since the feature vector size is \(r\) construct the input layer with \(r\) input nodes. The hidden layer nodes are equal to the number of training patterns. Hence, fix the hidden layer nodes as \(q\). The output layer nodes should be equal to the number of classes. Thus the output layer should have \(s\) nodes.

3. Fix the weights \((w_1)\) of the first layer as transpose of input pattern vectors \((p')\).

4. Fix the first layer biases \((b_1)\) to \(\sqrt{\log (0.5) / \text{spread}}\) or \(0.8326 / \text{spread}\). Spread is a parameter of the design of the network and allows flexibility to the radial basis neuron to accommodate the nearby patterns.

5. Find the outputs \(a_1\) of the first layer using the expression \(a_1 = \exp (-n \cdot n)\) where \(n = \|w_1 - p\| \cdot b_1\). That is \(\|\cdot\|\) indicates Euclidean distance between \(p\) and \(w_1\).

6. Second layer is competitive. The number of neurons in this layer is equal to the number of classes \(s\). The weights are set to the matrix \(t\) of target vectors. Each vector should have '1' only in the row associated with that particular class for which input is highest and zeroes elsewhere. Hence the weights assigned are \(w_2 = t\). This layer receives the net input (column) vector \(a_1\) that is obtained from the radial basis layer. In the column vector \(a_1\), maximum value and corresponding radial basis neuron is noted. The class node associated with this radial basis neuron generates output '1' and '0' s elsewhere.
Validation of the constructed network:

7. Let \( p \) be the testing vectors. Calculate the first layer outputs by using the expression \( a_1 = \exp(-n \cdot n) \) where \( n = \|w_1 - p\| \cdot b_1 \) which is the product of Euclidean distance \( \| \cdot \| \) and bias \( b_1 \).

8. Using the obtained output of the first layer \( a_1 \), and the weights and biases of the second layer calculate the outputs of the second layer. The second layer output contains one for which the class is selected.

4.5 Developed systems

The language of a document is reflected in the image of that document in at least two ways: the script or the character set and writing convention of the language [6]. Presently nine languages, English, Hindi, Kannada, Tamil, Gujarati, Malayalam, Oriya, Punjabi and Telugu, are considered and hence script identification is sufficient to classify the document by language.

Two systems are developed. The first one is based on the radial basis neural network classifier and second one is based on the probabilistic neural network classifier. The architectures of the two systems are shown in figures 4.7 and 4.8 respectively.

![Figure 4.7: Radial basis neural network classifier based document image script identification system. E, H, K, T, G, M, O, P and Te corresponds to English, Hindi, Kannada, Tamil, Gujarati, Malayalam, Oriya, Punjabi and Telugu respectively.](image-url)
In these systems unknown pattern is fed to a pre-processor. The function of the pre-processor is to convert the document image to the required size and conversion to the two tones. The resulting image which consists of only '1's and '0's (Black and White) is the input to the modified feature extractor discussed in section 4.3. It converts the image into 50 feature values. These 50 features values are the inputs to the already designed neural network classifier. The neural network classifier utilizes the knowledge it has acquired during the training and classifies the unknown input document image into one of the nine language script classes, English, Hindi, Kannada, Tamil, Gujarati, Malayalam, Oriya, Punjabi and Telugu.

![Diagram of the probabilistic neural network classifier based system.](image)

Figure 4.8: Probabilistic neural network classifier based system. E, H, K, T, G, M, O, P and Te corresponds to English, Hindi, Kannada, Tamil, Gujarati, Malayalam, Oriya, Punjabi and Telugu respectively.
4.6 Experiments and results

The database of document images of 64 x 64 pixels, developed as explained in section 4.2 is divided into the training set and the test set. Training set includes 180 documents, 20 documents in each of nine classes. The number of documents in training set affects the number of hidden neuron in the radial basis layer of both classifiers. As the number of training patterns increases, the number of hidden neurons increases, thus increasing the computational cost and complexity of the neural network. Hence the training set has been limited to 180 nodes in the following experiments. Test set includes the remaining documents of the database.

The first experiment is conducted on the radial basis neural network based system. The radial basis classifier is designed using the algorithm explained in section 4.4.1. To adjust the best spread, another experiment is conducted to determine the behaviour of classification accuracy with respect to the spread. Classification accuracy is the percentage of samples (including the ones that were also used for training) which are classified correctly. The procedure is explained in the algorithm 4.5. The result of this experiment is shown in figure 4.9. In this graph the spread is varied from 0.00001 to 0.15 in steps of 0.001. After the value 0.15, it remains constant. Hence, for this system a spread of 0.006 yields the best possible overall accuracy of 81.85%. Overall accuracy is defined as percentage of correctly classified samples in the whole database. Class wise accuracy is defined as the percentage of the correctly classified samples out of all samples of that class. The results obtained at this value of spread for 1080 document images are tabulated in table 4.2.

The second experiment is conducted on the probabilistic neural network based system on similar lines. The behaviour of the spread is shown in figure 4.10 and results at the best spread of 0.03 are tabulated in table 4.3. The overall classification accuracy of the system is 97.4%.

Algorithm 4.5: Obtaining the best spread value

1. Let the spread that has to be checked for suitable value varies in the range 0.00001 to 0.15 in steps of 0.001. Start with spread value 0.00001.
2. Design the neural classifier (e.g., using algorithms 4.3 or 4.4).
3. Input all the available patterns to the classifier and compare the obtained results with the true targets of the patterns. Note the total errors of the classifier.

4. Find the classification accuracy.

5. Store the spread value and its corresponding classification accuracy in the respective arrays.

6. Repeat the above steps from step 1 for the next spread value (e.g. spread = 0.02).

7. Find the maximum value in the accuracy array and its corresponding spread. This is the best value of the spread for a selected classifier to a given problem.

![Behaviour of Spread for Radial basis neural network](image)

Figure 4.9: The variation of classification accuracy for various spreads for radial basis neural network based system.
Table 4.2: Matrix showing detection accuracy between languages for radial basis neural network based system.

<table>
<thead>
<tr>
<th>Language</th>
<th>E</th>
<th>H</th>
<th>K</th>
<th>Ta</th>
<th>G</th>
<th>M</th>
<th>O</th>
<th>P</th>
<th>Te</th>
<th>Error/Rej</th>
<th>Class-wise Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>118</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>02</td>
<td>02</td>
<td>98.33</td>
</tr>
<tr>
<td>Hindi</td>
<td>-</td>
<td>71</td>
<td>-</td>
<td>-</td>
<td>01</td>
<td>02</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>59</td>
<td>95.17</td>
</tr>
<tr>
<td>Kannada</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>59</td>
<td>-</td>
<td>-</td>
<td>61</td>
<td>120</td>
<td>00</td>
</tr>
<tr>
<td>Tamil</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>116</td>
<td>-</td>
<td>-</td>
<td>04</td>
<td>-</td>
<td>-</td>
<td>04</td>
<td>96.67</td>
</tr>
<tr>
<td>Gujarati</td>
<td>01</td>
<td>-</td>
<td>-</td>
<td>119</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>01</td>
<td>99.17</td>
</tr>
<tr>
<td>Malayalam</td>
<td>02</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>113</td>
<td>03</td>
<td>02</td>
<td>07</td>
<td>01</td>
<td>94.17</td>
</tr>
<tr>
<td>Oriya</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>02</td>
<td>118</td>
<td>-</td>
<td>-</td>
<td>02</td>
<td>02</td>
<td>98.33</td>
</tr>
<tr>
<td>Punjabi</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>03</td>
<td>-</td>
<td>117</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>97.5</td>
</tr>
<tr>
<td>Telugu</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>08</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>112</td>
<td>01</td>
<td>93.33</td>
</tr>
<tr>
<td>Total Number of Documents</td>
<td>1080</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>81.85</td>
</tr>
</tbody>
</table>

Overall Accuracy

Figure 4.10: The variation of classification accuracy for various spreads for Probabilistic neural network based system.
Table 4.3: Matrix showing detection accuracy between languages for probabilistic neural network based system

<table>
<thead>
<tr>
<th>Language</th>
<th>E</th>
<th>H</th>
<th>K</th>
<th>Ta</th>
<th>G</th>
<th>M</th>
<th>O</th>
<th>P</th>
<th>Te</th>
<th>Error/ Rej</th>
<th>Class-wise Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>119</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>01</td>
<td>-</td>
<td>-</td>
<td>01</td>
<td>99.17</td>
</tr>
<tr>
<td>Hindi</td>
<td>-</td>
<td>117</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>03</td>
<td>-</td>
<td>-</td>
<td>03</td>
<td>97.5</td>
</tr>
<tr>
<td>Kannada</td>
<td>-</td>
<td>-</td>
<td>113</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>07</td>
<td>-</td>
<td>-</td>
<td>07</td>
<td>94.17</td>
</tr>
<tr>
<td>Tamil</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>100</td>
</tr>
<tr>
<td>Gujarati</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>100</td>
</tr>
<tr>
<td>Malayalam</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>112</td>
<td>08</td>
<td>-</td>
<td>-</td>
<td>08</td>
<td>93.3</td>
</tr>
<tr>
<td>Oriya</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>100</td>
</tr>
<tr>
<td>Punjabi</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>-</td>
<td>-</td>
<td>00</td>
<td>100</td>
</tr>
<tr>
<td>Telugu</td>
<td>-</td>
<td>-</td>
<td>09</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>111</td>
<td>-</td>
<td>09</td>
<td>92.5</td>
</tr>
<tr>
<td><strong>Total Number of Documents</strong></td>
<td><strong>1080</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall Accuracy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>97.4</strong></td>
</tr>
</tbody>
</table>

4.7 Discussions

The script identification systems presented above are certain notable advantages over those of previous chapter. These have capability to classify nine script classes in place of three script classes of previous systems. Here the training times are greatly reduced.

Comparison of tables 4.2 and 4.3 shows clearly that probabilistic neural network based system outperforms radial basis neural network based system. The main reason for such an improvement is due to the presence of a competitive layer along with radial basis layer in PNN. The other reasons may be even due to choice of simple design of Radial basis neural network classifier in the first system.

The close observation of table 4.2 reveals radial basis system failed in recognizing Kannada documents and also wrongly recognized many Hindi documents as Punjabi documents. Thus overall accuracy remained at 81.85% for this system. The results of PNN based system shown in table 4.3 presents comparatively better results. Tamil, Gujarati, Oriya and Punjabi documents are recognized 100%. Nine Telugu documents are recognized as Kannada documents and similarly seven Kannada documents are recognized as Telugu documents. This clearly shows that system very slightly gets
confused between these two scripts. This fact proves that system classifies on the same lines as that of human brain, since many people (who are not aware of both these languages) usually get the confusion between Kannada and Telugu documents. The documents of these two scripts look alike and even many characters in both these languages are common. Thus results of the PNN based system are closer to the truth and produces higher overall accuracy i.e. 97.4%.

4.8 Conclusions
Young-sup Hwang, Sung-Yang Bang stated as follows in connection with design procedure of radial basis neural network. "While many character recognition methods using neural network have been proposed, only few use an RBF network which actually has powerful pattern recognition ability. This fact seems to be due to the situation that an easy but effective procedure to design a good RBF network to solve a given problem is not known" [36]. The above statement confirms that radial basis network classifiers have powerful recognition abilities but have not been explored to a great extent.

In this chapter an easy to use design procedure for constructing the radial basis neural network is explained in detail. A step ahead, a modified version of the radial basis neural network, that is a probabilistic neural network design procedure is explained and actually applied to build script identification system of nine language documents.

On the other hand, this chapter also presents an modified automatic script identification system which classifies nine language scripts. Since using the neural network for such application is new it can only be compared with the script identification system presented in the previous chapter. The systems presented in this chapter do not depend on training that was the major drawback of the system presented in chapter 3. Training times can be greatly reduced without compromising the recognition abilities. It is also possible to extend the number of languages in such systems. Only requirement is the availability of document images in such languages. No manual analysis of additional script class is required, since the system itself is trainable.