Off-line Nepali Handwriting Recognition Using MLP and RBF Neural Networks

A Dissertation Proposal

Submitted By

Ashok Kumar Pant

Roll No. 12

Date: 10 Apr 2012

Submitted To

Central Department of Computer Science and Information Technology
Kirtipur, Kathmandu, Nepal

Under the Supervision of

Prof. Dr. Shashidhar Ram Joshi

and

Co-Supervision of

Dr. Sanjeeb Prasad Panday

1 Introduction

Off-line handwriting recognition is the task of determining what characters or words are present in a digital image of handwritten text. The problem of handwriting recognition can be viewed as a classification problem where we need to identify the most suitable character the given figure matches to. It is a sub-field of Optical Character Recognition (OCR), whose domain can be machine-print or handwriting but is more commonly machine-print. The recognition of Nepali Handwritten text present unique challenges and benefits, as it belongs to devnagari script then the recognition of text in other script like Roman, Arabic, chinese, etc.

A recognition system can either be on-line or off-line. In on-line recognition system the real time temporal information about the character being written is available such as the stroke order, pen up and pen down time, sequence of points traced, etc. Generally on-line recognition is preformed in fixed writing bench with some digital pen or plotter. Operating in off-line mode gives as input the complete picture of character that we need to recognize. We don't have any temporal information about the text written. The input to off-line recognition system is the image captured with digital camera or scanned with some digital scanner. The on-line problem is somewhat easier than off-line since there is more information available.

Artificial neural network is non-linear,parallel, distributed, highly connected network having capability of adaptivity, self-organization, fault tolerance, evidential response and VLSI (Very Large Scale Integration) implementation which closely resembles with physical nervous system. Physical nervous system is highly parallel, distributed information processing system having high degree of connectivity with capability of self learning. Virtual intelligence can be mimicked with the help of Artificial Neural Network (ANN). Artificial neural network algorithms can be used to simulate the real, complex biological nervous system to solve the real world problems with good performance (e.g. as measured by good predictive ability, low generalization error), or performance mimicking human error patterns, as the information processing is done in the brain. The main idea behind using ANN as recognition system is its close resemblance with physiological nervous system. Application areas of ANNs include system identification and control (vehicle control, process control), game-playing and decision making (chess, racing), pattern recognition (radar systems, face identification, object recognition), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases), visualization and e-mail spam filtering and so on.

2 Problem Definition

The high-level task of off-line handwriting recognition is to classify the ordered sequence of images of off-line characters. In this research work problem of Nepali handwritten character recognition is addressed. The recognition task is carried out with Artificial Neural Network. Many geometric and statistical features are extracted from images so that the performance and accuracy of recognition system is achieved in the range of human ability of recognition. The system performs character recognition by quantification of the character into a mathematical vector entity using the geometrical and statistical properties of the character image. The sub problems in the domain of off-line handwriting recognition such as, noise removal, image binarization, object skeletonization, size normalization, etc. have great impact on recognition procedure. These sub-problems are also addressed with the most suitable solutions in the literature for this type research work.

3 Objectives

The objective of this research work is to investigate various feature extraction techniques and to compare Neural Network based pattern recognition techniques namely Multilayer Feed-forward Network and Radial Basis Function Network to improve accuracy of off-line Nepali Handwriting Recognition. Comparative Performance matrices are analysed. The sub-problem field of off-line handwriting recognition is also addressed. Main objectives are given below.

- To compare performance and efficiency of Multilayer Perceptron and Radial Basis function Neural Network on Off-line Nepali Handwriting Recognition Problem.
- To investigate Geometric and Statistical feature extraction techniques for off-line Nepali handwritten text.
- To investigate preprocessing techniques (segmentation, skeletonization, normalization, etc.) for handwritten documents.

4 Research Methodology

The methods used in the main module of this research work are described in this chapter. Sections are organized as follows. State of the Art for the handwriting recognition is presented in section 4.1. Preprocessing steps for the off-line handwritten characters is described in section 4.2. Section 4.3 contains the feature extraction techniques form the character images. Training and Recognition procedure is described in section 4.4.

4.1 Literature Review

The recognition of handwriting by machines has been a research topic for over 40 years. Before the age of digital computers there was no much researches in the field of handwriting recognition. The early researches after the digital age were concentrated either upon machine-printed text or upon a small set of well-separated handwritten text or symbols. Machine-printed Character Recognition(CR) generally used template matching and for handwritten text, low-level image processing techniques were used on the binary image to extract feature vectors,which were then fed to statistical classifiers [1].

CR research is somewhat limited until 1980 due to the lack of powerful computer hardware and data perception devices. The period from 1980-1990 witnessed a growth in CR system development [2] due to rapid growth in information technology [3]. However, the CR research was focused on basically the shape recognition techniques without using any semantic information. This led to an upper limit in the recognition rate, which was not sufficient in many practical applications. Research progress on the off-line and on-line recognition during 1980-1990 can be found in [4] and [5] respectively.

After 1990, image processing techniques and pattern recognition were combined using artificial intelligence. Along with powerful computers and more accurate electronic equipments such as scanners, cameras and electronic tablets, there came in efficient, modern use of methodologies such as artificial neural networks (ANNs), hidden Markov models (HMMs), fuzzy set reasoning, and natural language processing.

Character segmentation from cursive handwritten documents is a difficult task. So in literature most of the researches were conducted on separated characters. Isolated off-line Devnagari word recognition work is given in [6] and isolated character recognition in [7]. Research work [8] describes the template based Nepali alphanumeric handwriting recognition. Recent work on off-line devanagari character recognition carried out by Sharma et.al. (2006) uses quadratic classifiers for recognition and achieved 98.86% recognition accuracy for devanagari numerals and 80.36% recognition accuracy devanagari characters [9]. Research work [10] done in year 2010 describes multiple classifier combination for Off-line Handwritten Devnagari Character Recognition and achieves 92.16% recognition accuracy for devanagari characters. Paper [11] compares Support Vector Machine (SVM) and ANN for off-line devanagari character recognition problem. Handwriting Recongnition system based on cloud computing in given in paper [12].

On Devnagari,a few techniques have been tested but no comparison of various recognition techniques available in literature is made. Also there is lack of benchmark databases for Handwritten Devnagari Script to test the recognition systems. Only small lab experiments have been found in the literature.

Although research on recognizing isolated handwritten characters has been quite successful, recognizing off-line cursive handwriting has been found to be a challenging problem. There is a large corpus of research on the application of character recognition in different domains, but no system to date has achieved the goal of system acceptability.

4.2 Preprocessing

The image preprocessing technique [13] for the off-line handwritten character images is described in this section. The main steps are given in the algorithm 4.1.

Algorithm 4.1 Image Preprocessing

```
1: Read image.
2: Convert RGB images to gray scale image.
3: Remove noise using median filter.
4: Convert the gray scale images to binary image.
5: Image Inversion (making background black & foreground white).
6: Determine the universe of discourse of image.
7: Normalize the image to a predefined size of 36x36 pixels.
8: Convert the normalized image to single pixel thick skeleton image.
```

4.3 Feature Extraction

After Pre-processing of the character images the next stage of character recognition is feature extraction. Feature extraction step plays one of the most important roles in the recognition. It is the heart of recognition system. A good feature set should represent characteristic of the class that helps distinguish it from other classes, while remaining invariant to characteristic differences within the class. Hundreds of features are available in the literature [14, 15].

In this research work combination of statistical features (Moment Invariants, Centroid and Area) and topological features (Character Geometry, Euler Number and Eccentricity) are used to describe the character property. These features are extracted from individual preprocessed handwritten character images.

4.3.1 Geometric Features

Geometric features are extracted from skeletonized character image based on the basic line types that form the skeleton of the character. These features may represent global and local properties of characters and have high tolerances to distortions and style variations. They also tolerate a certain degree of translation and rotation. These topological features may encode some knowledge about the contour of the object or may require some knowledge as to what sort of components make up that object. This technique implements the idea given in [16]. Each pixel in the image is traversed. Individual line segments, there directions and intersection points are identified from segmented character image.

For this, given pre-processed character image is divided into 3x3 windows of equal size (zoning). Then number, length, and type of lines and intersection points are identified in each zone. The line segment that would be determined in each zone is categorized into four types; vertical lines, horizontal lines, right diagonal and left diagonal. To extract direction features, the following steps are required

1. Starting points and intersection point identification
2. Individual line segment identification
3. Labelling line segment
4. Line type normalization

From each zone following properties are extracted.

1. The number of horizontal lines.
2. The number of vertical lines.
3. Number of Right diagonal lines.
4. Number of Left diagonal lines.
5. Normalized Length of all horizontal lines.
6. Normalized Length of all vertical lines.
7. Normalized Length of all right diagonal lines.
8. Normalized Length of all left diagonal lines.
9. Number of intersection points

It results feature vector of dimensional 9 for each zone. A total of 81 (9x9) features are obtained.

4.3.2 Moment Invariant Features

Moment invariants are important tools in object recognition problem. These techniques grab the property of image intensity function. The invariant moments used in this thesis work have been proposed in [17]. These seven moments are well-known to be invariant to position, size and orientation of the character. They are pure statistical measures of the pixel distribution around the center of gravity of the character and allow capturing the global character shape information.

The standard moments m_{pq} of order $(p+q)$ of an image intensity function $f(x,y)$ is given by,

$$m_{pq} = \int\limits_{-\infty}^{\infty} \int\limits_{-\infty}^{\infty} x^p y^q f(x,y) dx dy \;\; p,q = 0,1,2,... \tag{4.1}$$

A uniqueness theorem states that if $f(x,y)$ is piecewise continues and has non-zero values only in a finite part of the x_{vis}, y_{vis} plane, moments of all order exist and the moment sequence (m_{pq}) is uniquely determined by $f(x,y)$. Conversely, (m_{pq}) is uniquely determines $f(x,y)$.

For discrete domain,The 2-D moment of order $(p+q)$ for a digital image $f(x,y)$ of size MxN is given by

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x,y) \;\; p,q = 0,1,2,... \tag{4.2}$$

The corresponding central moment of order $(p+q)$ is defined as

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x-\bar{x})^p (y-\bar{y})^q f(x,y) \;\; p,q = 0,1,2,... \tag{4.3}$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}} \; and \; \bar{y} = \frac{m_{01}}{m_{00}} \tag{4.4}$$

The normalized central moments, denoted by η_{pq}, are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{4.5}$$

where

$$\gamma = \frac{p+q}{2} + 1 \;\; for \;\; p+q = 2,3,... \tag{4.6}$$

A set of seven invariant moments can be derived from the second and third moments [17] which are invariant to translation, scale change, mirroring, and rotation, are given as follows.

$$\phi_1 = \eta_{20} + \eta_{02} \tag{4.7a}$$
$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{4.7b}$$
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{12} - \eta_{03})^2 \tag{4.7c}$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{4.7d}$$
$$\begin{aligned}\phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} - 3\eta_{12})^2 - 3(\eta_{12} + \eta_{03})^2] \\ & + (3\eta 21 - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned} \tag{4.7e}$$
$$\begin{aligned}\phi_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\end{aligned} \tag{4.7f}$$
$$\begin{aligned}\phi_7 = {} & 3(\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + 3(\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned} \tag{4.7g}$$

4.3.3 Euler Number

Euler number is the difference of number of objects and the number of holes in the image. This property is affine transformation invariant. Euler number is computed by considering patterns of convexity and concavity in local 2-by-2 neighbourhoods [18]. Euler number can be calculated by using local information and does not require connectivity information. Calculating the Euler number of a binary image can be done by counting the occurrences of three types of 2x2 binary patterns.

$$P1 = \left(\begin{matrix} 1 & 0 \\ 0 & 0 \end{matrix}, \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix}, \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix}, \begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix} \right)$$

$$P2 = \left(\begin{matrix} 0 & 1 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 0 \\ 1 & 1 \end{matrix}, \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}, \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \right)$$

$$P3 = \left(\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}, \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \right)$$

Let $C1$, $C2$ and $C3$ be the number of occurrences of patterns $P1$, $P2$ and $P3$ respectively, then the Euler Numbers for the image with 4- and 8-connectivity is simple given as [18],

$$E_4 = \frac{1}{4}(C1 - C2 + 2C3) \tag{4.8}$$

$$E_8 = \frac{1}{4}(C1 - C2 - 2C3) \tag{4.9}$$

4.3.4 Normalized Area of Character Skeleton

The area of the object within the binary image is simply the count of the number of pixels in the object for which $f(x, y) = 1$. Normalized regional area is the ratio of the number of the pixels in the skeleton of binary image to the total number of pixel in the image.

4.3.5 Centroid of Image

Centroid specifies the center of mass of the object region in given image. Horizontal centroid coordinate is calculated by dividing the sum of all horizontal positions of the object which are non zero with area of the object. Similarly vertical centroid coordinate is calculated by dividing the sum of all vertical positions of the object which are non zero with area of the object.

4.3.6 Eccentricity

Eccentricity is the ratio of length of semi-major axis to the length of semi-minor axis of the smallest ellipse that fits the skeleton of the image.

4.4 Recognition

After the feature extraction phase process of training and testing begins. In the training phase recognition system learns patterns of different classes from input feature vectors. The learning is done in supervised manner. After the training phase system is ready to test in unknown environment. Recognition system is then tested against testing feature vectors and accuracy and efficiency of the system is calculated. Here in this research recognition is carried out using two neural network algorithms, multilayer feedforward neural network with Levenberg-Marquardt learning and radial basis function networks with orthogonal least square learning. Section 4.4.1 describes the MLP algorithm and section 4.4.2 describes the RBF algorithm.

4.4.1 Multilayer Feedforward Backpropagation Network

A multilayer feedforward neural network consists of a layer of input units, one or more layers of hidden units, and one layer of output units. A neural network that has no hidden units is called a Perceptron. However, a perceptron can only represent linear functions, so it isn't powerful enough for the kinds of applications we want to solve. On the other hand, a multilayer feedforward neural network can represent a very broad set of non-linear functions. So, it is very useful in practice. This multilayer architecture of Network determines how the input is processed. The network is called feedforward because the output from one layer of neurons feeds forward into the next layer of neurons. There are never any backward connections, and connections never skip a layer. Typically, the layers are fully connected, meaning that all units at one layer are connected with all units at the next layer. So, this means that all input units are connected to all the units in the layer of hidden units, and all the units in the hidden layer are connected to all the output units.

Usually, determining the number of input units and output units is clear from the application. However, determining the number of hidden units is a bit of an art form, and requires experimentation to determine the best number of hidden units. Too few hidden units will prevent the network from being able to learn the required function, because it will have too few degrees of freedom. Too many hidden units may cause the network to tend to over-fit the training data, thus reducing generalization accuracy.

Consider a multilayer feed-forward network as shown in Figure 1. The net input to unit I in layer $k + 1$ is given by

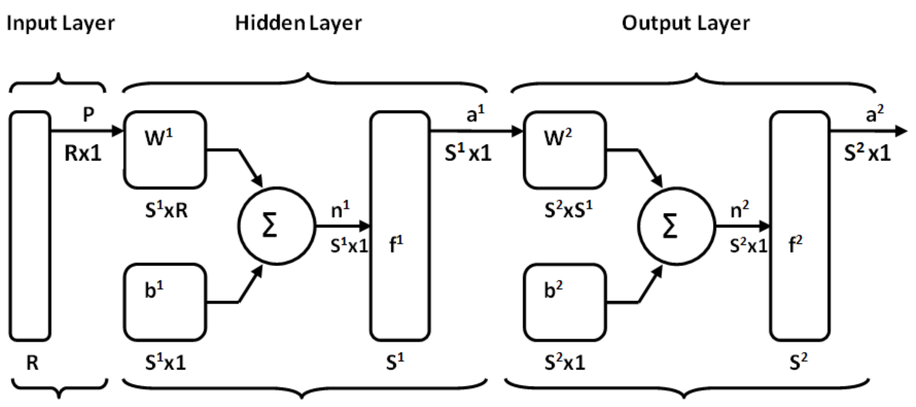


Figure 1: Feedforward Multilayer Perceptron

$$n_i^{k+1} = \sum_{i=1}^{Sk} w_{ij}^{k+1} a_j^k + b_i^{k+1} \tag{4.10}$$

where Sk is the number of neurons in the k^{th} layer, w_{ij}^{k+1} is the weight from j^{th} neuron of layer k to neuron i of the layer $k+1$, a_j^k is the output of neuron j from the layer k and b_i^{k+1} is the bias connected to i^{th} neuron in the layer $k+1$.

The output of unit I will be,

$$a_i^{k+1} = f^{k+1}(n_i^{k+1}) \tag{4.11}$$

where $f^{k+1}(\cdot)$ is the activation function used in layer $k+1$

For a M layer network the system equations in the matrix form are given by

$$a^0 = p \tag{4.12}$$

$$a^{k+1} = f^{k+1}(W^{k+1}a^k + b^{k+1}), k = 0, 1, 2, ..., M-1 \tag{4.13}$$

Where p is the input layer of the network

The network learns association between a given set of input-output pairs $(p_i, t_i), i = 1, 2, ..., Q$ for Q number of training samples.

Now, next step of calculating feed forward quantities for all layers in the network is the back propagation training. In this phase we decide how neural network learn weights and biases with minimizing the generalization error. The quantity of weight and bias to adjust in the network parameters is send back to network from output layer to first hidden layer. In this research work Levenberg-Marquardt Backpropagation Learning rule is applied. Levenberg-Marquardt Backpropagation Learning algorithm is given in section 4.4.1.1.

4.4.1.1 Levenberg-Marquardt (LM) Learning Algorithm

Many algorithms focus on standard numerical optimization, that is, using alternative methods for computing the weights associated with network connections. The most popular algorithms for this optimization are the conjugate gradient and Newton's methods. Newton's method is considered to be more efficient in the speed of convergence, but its storage and computational requirements go up as the square of the size of the network. The LM algorithm is efficient in terms of high speed of convergence and reduced memory requirements compared to the two previous methods. In general, with networks that contain up to several hundred weights, the LM algorithm has the fastest convergence [19].

For LM , the performance index to be minimized is defined as,

$$F(w) = \sum_{p=1}^{Q} \sum_{k=1}^{K} (t_{kp} - a_{kp})^2 \tag{4.14}$$

where $w = [w_1 \; w_2 \; w_3 \; ... \; w_N]^T$ consists of all the weights of the network (including bias), t_{kp} is the target value of the k^{th} output and p^{th} input pattern, a_{kp} is the actual value of the k^{th} output and p^{th} input pattern, N is the number of weights, and K is the number of the network outputs.

Equation (4.14) can be written as

$$F(w) = E^T E \tag{4.15}$$

where $E = [e_{11} \; ... \; e_{K1} \; e_{12} \; ... \; e_{K2} \; ... \; e_{1P} \; ... \; e_{KQ}]^T$

$e_{kp} = t_{kp} - a_{kp}$, $k = 1, 2, ..., K$, $p = 1, 2, ..., Q$ where E is the cumulative error vector (for all input patterns). From equation (4.15) the Jacobian matrix is defined as

$$J = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \cdots & \frac{\partial e_{11}}{\partial w_N} \\ \frac{\partial e_{21}}{\partial w_1} & \frac{\partial e_{21}}{\partial w_2} & \cdots & \frac{\partial e_{21}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{K1}}{\partial w_1} & \frac{\partial e_{K1}}{\partial w_2} & \cdots & \frac{\partial e_{K1}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{1Q}}{\partial w_1} & \frac{\partial e_{1Q}}{\partial w_2} & \cdots & \frac{\partial e_{1Q}}{\partial w_N} \\ \frac{\partial e_{2Q}}{\partial w_1} & \frac{\partial e_{2Q}}{\partial w_2} & \cdots & \frac{\partial e_{2Q}}{\partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_{KQ}}{\partial w_1} & \frac{\partial e_{KQ}}{\partial w_2} & \cdots & \frac{\partial e_{KQ}}{\partial w_N} \end{bmatrix} \tag{4.16}$$

The increment of weights Δw at iteration t can be obtained as follows:

$$\Delta w = -(J^T J + \mu I)^{-1} J^T E \tag{4.17}$$

where I is identity matrix, μ is a learning parameter and J is Jacobian of K output errors with respect to N weights of the neural network.

Now the weights are calculated using the following equation

$$w_{t+1} = w_t + \Delta w \tag{4.18}$$

For $\mu = 0$ it becomes the Gauss-Newton method. For very large value of μ the LM algorithm becomes the steepest decent algorithm. The learning parameter μ is automatically adjusted at each iteration to meet the convergence criteria. The LM algorithm requires computation of the Jacobian matrix J at each iteration and inverse of $J^T J$ matrix of dimension NxN, This large dimensionality of LM algorithm is sometimes unpleasant to large size neural network.

The learning parameter μ is updated in each iteration as follows. Whenever the $F(w)$ value is decreases, μ is multiplied by decay rate β_{inc} whenever $F(w)$ is increases, μ is divided by decay rate β_{dec} in new step.

The LM Back-propagation training is illustrated in the Algorithm 4.2.

Algorithm 4.2 Levenberg-Marquardt Back-propagation Algorithm

```
 1: Initialize the weights using Nguyen-Widrow weight initialization algorithm and parameter
    $\mu$.
 2: Compute the sum of the squared errors over all inputs $F(w)$.
 3: Solve Eq.(4.17) to obtain the increment of weights.
 4: Recompute the sum of squared errors $F(w)$ using $w + \Delta w$ as trial w,and adjust
 5: if trial $F(w) \leq F(w)$ in Step 2 then
 6:     $w = w + \Delta w$
 7:     $\mu = \mu.\beta_{inc}$
 8:     Goto Step 2.
 9: else
10:     $\mu = \frac{\mu}{\beta_{dec}}$
11:     Goto Step 4.
12: end if
```

4.4.2 Radial Basis Function Network

Radial-basis function network is used to perform a complex pattern classification task, the problem basically solved by transforming it into a high dimensional space in a non-linear manner. Non linear separability of complex patterns is justified by Cover's theorem,which stated as, "A complex pattern-classification problem cast in a high dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space".

Radial basis functions are embedded into a two layer feed-forward network. Such a network is characterized by a set of inputs and set of outputs. In between the inputs and outputs there is a layer of processing units called hidden units. Each of them implements a radial basis function. Generally there is a single hidden layer. In pattern classification problem inputs represent feature values, while each output corresponds to a class. A general RBF Network architecture is given in Figure 2.

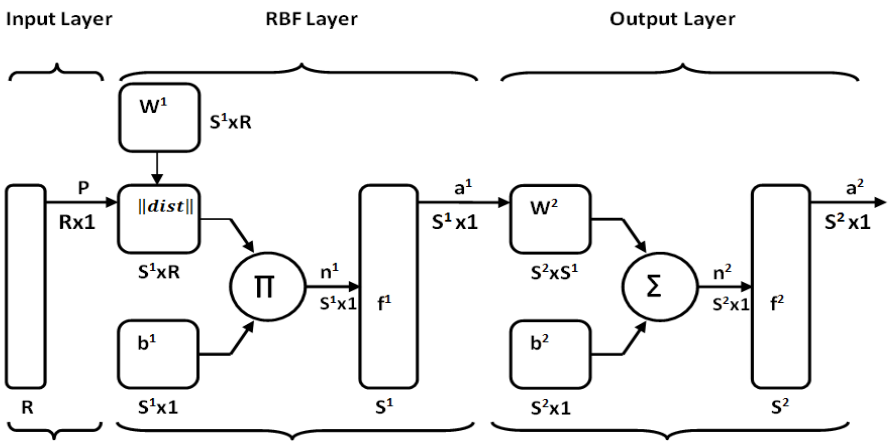


Figure 2: RBF Neural Network

Let the input vector $x = [x_1, x_2, ..., x_R]^T$ is feed to RBF Network.The network output can be obtained by

$$t_k = \sum_{j=1}^{M} w_{kj}\phi_j(x) + b_k \tag{4.19}$$

where $\phi_j(\cdot)$ denotes the radial basis function of the j^{th} hidden neuron, w_{kj} denotes the hidden-to-output weight from j^{th} hidden neuron to k^{th} output neuron, b_k denotes the bias connected to k^{th} output neuron, and M denotes the total hidden neurons.

A radial basis function is a multidimensional function that describes the distance between a given input vector and a pre-defined centre vector. There are different types of radial basis function. A normalized Gaussian function usually used as a radial basis function, which is given as

$$\phi_j(x) = \exp(-\frac{\|x - \mu_j\|^2}{2\sigma_j^2}) \tag{4.20}$$

where μ_j and σ_j denote the centre and spread width of the j^{th} neuron,respectively.

Generally, the RBFNN training can be divided into two stages:

1. Determine the parameters of radial basis functions, i.e., Gaussian centre and spread width.
2. Determine the output weight w with supervised learning method. Usually Least-Mean-Square (LMS)

The first stage is very crucial, since the number location of centres in the hidden layer will influence the performance of the RBFNN directly. In the next section, the principle and procedure of self-structure RBF algorithm will be described.

4.4.2.1 Orthogonal Least Square Training Algorithm

Orthogonal Least Squares (OLS) is the most widely used supervised algorithm for radial basis neural network training. OLS is a forward stepwise regression procedure. Starting from a large pool of candidate centres ,i.e., from training examples, OLS sequentially selects the centre that results in the largest reduction of sum-square-error at the output.

For M basis vectors $\Phi = (\phi_1, \phi_2, ..., \phi_M)$, orthogonal set $Q = (q_1, q_2, ..., q_M)$ can be derived with the help of Grahm-Schmidt orthogonalization procedure as follows:

$$q_1 = \phi_1 \tag{4.21}$$

$$\alpha_{ik} = \frac{q_i^T \phi_k}{q_i^T q_i} \; 1 \le i \le k \tag{4.22}$$

$$q_k = \phi_k - \sum_{i=1}^{k-1} \alpha_{ik} q_i \tag{4.23}$$

$$for\ k = 2, 3, ..., M$$

The orthogonal set of basis vectors Q is linearly related to the original set Φ by the relationship [20] $\Phi = QA$, where A is upper triangular matrix given by

$$A = \begin{bmatrix} 1 & \alpha_{12} & ... & \alpha_{1M} \\ 0 & 1 & & ... \\ ... & & & \alpha_{M-1,M} \\ 0 & ... & 0 & 1 \end{bmatrix} \tag{4.24}$$

Using this orthogonal representation, the RBF solution is expressed as

$$T = \Phi W = QG \tag{4.25}$$

and the Least Square (LS) solution for the weight vector G in the orthogonal space is

$$G = (Q^T Q)^{-1} Q^T T \tag{4.26}$$

Since Q is orthogonal, $Q^T Q$ is then diagonal and each component of G can be extracted independently without ever having to compute a pseudo-inverse matrix,as

$$g_i = \frac{q_i^T T}{q_i^T q_i} \tag{4.27}$$

The sum of squares of the target vector T is given as

$$T^T T = \sum_{i=1}^{M} g_i^2 q_i^T q_i + E^T E \tag{4.28}$$

OLS select a subset of regressors in stepwise forward manner by selecting the regressor q_i,which contributes to reduction of the error (relative to the total $T^T T$) by

$$[err]_i = \frac{g_i^2 q_i^T q_i}{T^T T} \tag{4.29}$$

The summary of Orthogonal Least Square RBF training [20] is given in Algorithm 4.3.

Algorithm 4.3 OLS Algorithm

```
1: At the first step,for $1 \le i \le M$, compute
2:             Orthogonalize vector: $q_1^{(i)} = \phi_i$
3:             Compute LS Solution: $g_1^{(i)} = \frac{{q_1^{(i)}}^T T}{{q_1^{(i)}}^T q_1^{(i)}}$
4:             Compute error reduction: $[err]_i^{(1)} = \frac{{g_1^{(i)}}^2 {q_1^{(i)}}^T q_1^{(i)}}{T^T T}$
5:    select regressor that yields heights reduction in error $q_1 = \arg_{q_1^{(i)}} \max [err]_1^{(i)} = \phi_{i_1}$
6: At the $k^{th}$ step,for $1 \le i \le M$, and $i$ not already selected
7:             Orthogonalize vector: $\alpha_{jk}^{(i)} = \frac{q_j^T \phi_i}{q_j^T q_j}$ , $q_k^{(i)} = \phi_i - \sum_{i=1}^{k-1} \alpha_{jk}^{(i)} q_j$ , $for 1 \le j \le k$
8:             Compute LS Solution: $g_k^{(i)} = \frac{{q_k^{(i)}}^T T}{{q_k^{(i)}}^T q_k^{(i)}}$
9:             Compute error reduction: $[err]_k^{(1)} = \frac{{g_k^{(i)}}^2 {q_k^{(i)}}^T q_k^{(i)}}{T^T T}$
10:   select regressor $q_k = \arg_{q_k^{(i)}} \max [err]_k^{(i)} = \phi_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} q_j$
11: Stop at iteration M if rasidual error falls below pre-specified threshold $\epsilon$
12:                 $1 - \sum_{j=1}^{M} [err]_j \le \epsilon$
13: The regressors $\{\phi_{i_1}, \phi_{i_2}, \cdots, \phi_{i_M}\}$ define the final subset of RBF centers.
```

5 Database Creation

To do experimentation with this research work, benchmark database for Off-line Nepali Handwritten Characters will be created. There will be three separate databases for Nepali handwritten consonants, vowels and numerals. More detail about Nepali consonant, vowel, and numeral database is given in the section 5.1, section 5.2 and section 5.3 respectively. Training and testing samples for recognition system will be then selected from each database in certain ratio.

5.1 Nepali Handwritten Consonant Database

Nepali language contains 36 Devanagari alphabets for Consonants. Figure 3 shows one sample for Nepali handwritten consonants. To create training and testing database for recognition system, handwritten consonant images are scanned with digital scanner and cropped for individual characters. There will be about 1000 samples for each 36 classes of consonants. Samples will be taken from different writers.

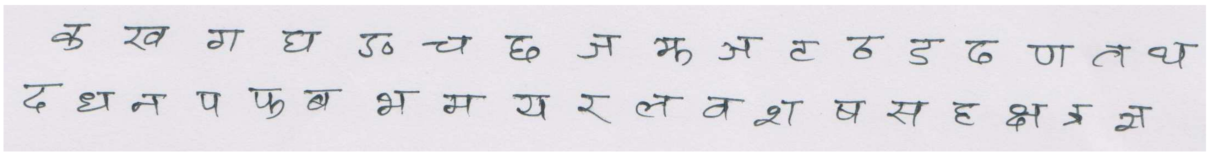


Figure 3: Nepali Handwritten Consonants

5.2 Nepali Handwritten Vowel Database

Nepali language contains 12 Devanagari alphabets for vowels. Figure 4 shows one sample for Nepali handwritten vowels. To create training and testing database for recognition system, handwritten vowel samples are scanned with digital scanner and cropped for individual characters. There will be about 1000 samples for each 12 classes of vowels. Samples will be taken from different writers.

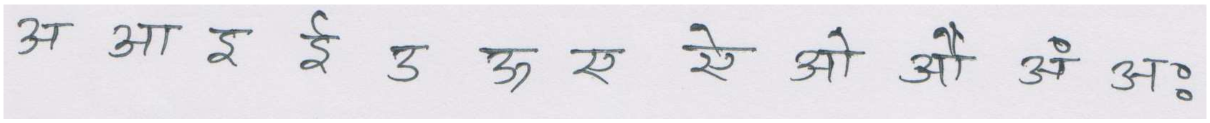


Figure 4: Nepali Handwritten Vowels

5.3 Nepali Handwritten Numeral Database

Nepali language contains 10 Devanagari numerals. Figure 5 shows one sample for Nepali handwritten numerals. To create training and testing database for recognition system, handwritten numeral images are scanned with digital scanner and cropped for individual characters. There will be about 1000 samples for each 10 classes of numerals. Samples will be taken from different writers.

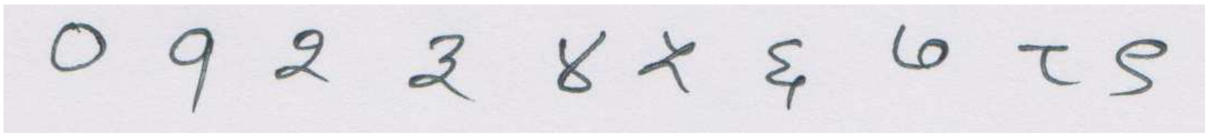


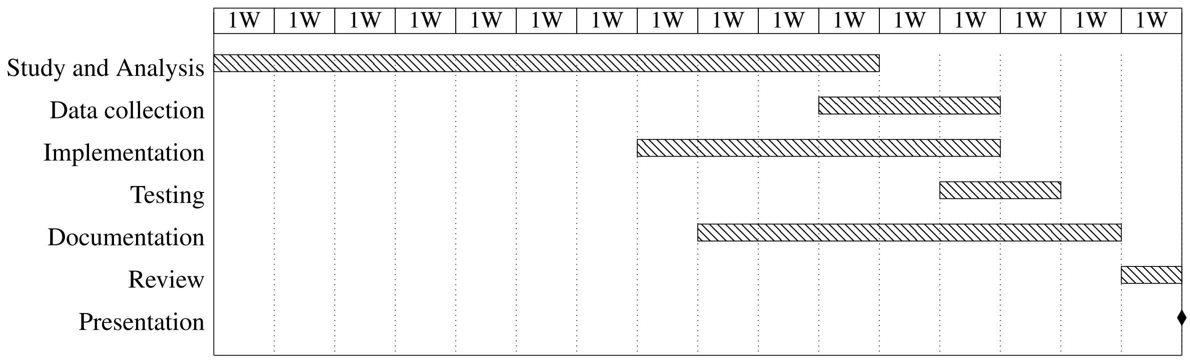
Figure 5: Nepali Handwritten Numerals

6 Expected Result

The attainable accuracy and efficiency defined by the Recognition models could be achieved as a final result.

7 Working Schedule

Working Schedule for the dissertation work is given below. Horizontal axis contain the time slots for the work. Each slot represent one week of time. Vertical axis contain tasks to be done.



Bibliography

[1] C. Y. Suen, M. Berthod, and S. Mori, "Automatic recognition of handprinted characters: The state of the art," *Proceedings of IEEE*, vol. 68, pp. 469–487, Apr. 1980.

[2] V. K. Govindan and A. P. Shivaprasad, "Character recognition: A review," *Pattern Recognition*, vol. 23, no. 7, pp. 671–683, 1990.

[3] R. M. Bozinovic and S. N. Srihari, "Off-line cursive script word recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, pp. 68–83, Jan. 1989.

[4] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of OCR research and development," *Proceedings of IEEE*, vol. 80, pp. 1029–1058, July 1992.

[5] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in on-line handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-12, pp. 787–808, Aug. 1990.

[6] B. Shaw, S. K. Parui, and M. Shridhar, "Offline handwritten devanagari word recognition: A segmentation based approach," in *ICPR*, pp. 1–4, 2008.

[7] P. S. Deshpande, L. Malik, and S. Arora, "Fine classification & recognition of hand written devnagari characters with regular expressions & minimum edit distance method," *JCP*, vol. 3, no. 5, pp. 11–17, 2008.

[8] K. C. Santosh and C. Nattee, "Template-based nepali handwritten alphanumeric character recognition," *Thammasat International Journal of Science and Technology (TIJSAT), Thammasat University, Thailand*, vol. 12, no. 1, 2007.

[9] N. Sharma, U. Pal, F. Kimura, and S. Pal, "Recognition of off-line handwritten devnagari characters using quadratic classifier," in *Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 805–816, 2006.

[10] S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu, and M. Kundu, "Multiple classifier combination for off-line handwritten devnagari character recognition," June 30 2010.

[11] S. Arora, D. Bhattacharjee, M. Nasipuri, L. Malik, M. Kundu, and D. K. Basu, "Performance comparison of SVM and ANN for handwritten devnagari character recognition," June 30 2010.

[12] Y. Gao, L. Jin, C. He, and G. Zhou, "Handwriting character recognition as a service: A new handwriting recognition system based on cloud computing," in *ICDAR*, pp. 885–889, IEEE, 2011.

[13] N. VenkateswaraRao, A. Shrikhna, B. RaveendraBabu, and G. R. M. Babu, "An efficient feature extraction and classification of handwritten digits using neural networks," *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, vol. 1, Oct. 2011.

[14] O. D. Trier, A. K. Jain, and T. Taxt, "Feature-extraction methods for character-recognition: A survey," *Pattern Recognition*, vol. 29, pp. 641–662, Apr. 1996.

[15] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4–37, Jan. 2000.

[16] M. Blumenstein, B. Verma, and H. Basli, "A novel feature extraction technique for the recognition of segmented handwritten characters," in *ICDAR*, pp. 137–141, 2003.

[17] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. IT-8, pp. 179–187, 1962.

[18] W. K. Pratt, *Digital Image Processing (2nd Edition)*. New York, New York: John Wiley & Sons, Inc., 1991.

[19] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, pp. 989–993, Nov. 1994.

[20] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. on Neural Networks*, vol. 2, pp. 302–309, 1991.