International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 04, April-2015

Handwritten Malayalam Word Recognition System using Neural Networks

Manoj Kumar P.
Assistant Professor in Computer Science,
CUCEK, CUSAT,
Pulincunnoo, Kerala, India.

Sandeep Chandran,
Assistant Professor in Information Technology,
LBS ITWE,
Trivandrum, Kerala, India.

Abstract: **The work describe an intelligent system for free hand entry of characters and words using light pen model. The system developed will recognize the character and words. The various approaches for handwritten character recognition are studied in the literature review phase. The different approaches are string matching schemes, structural approach, Template matching, using neural networks etc. The central objective of this project is demonstrating the capabilities of Artificial Neural Network implementations with back propagation algorithm in recognizing Malayalam characters. An emerging technique in the character recognition application area is the use of Artificial Neural Network implementation with networks employing specific guides (learning rules) to update the links (weights)between their nodes .Such network can be fed the data from the graphic analysis of the input picture and trained to output characters on one or another form . One such network with supervised learning rule is the Multi – Layer Perception (MLP) model. It uses the generalized Delta Learning Rule for adjusting its weight and can be trained for a set of input /desire output values in a number of iterations. The very nature of this particular model is that it will force the output to one of nearby values if a variation of input is fed to the network that it is not the technical approach is followed is processing input characters detecting line segments, obtaining the direction feature vector and training the network for a set of desired characters corresponding to the input characters. Finally, the word is recognized by checking the database trained for, thus solving the proximity issue.**

I.INTRODUCTION

Handwriting recognition is classically separated in two distinct domains: online and offline recognition. These two domains are differentiated by the nature of the input signal. For offline recognition, a static representation resulting from the digitalization of a document is available. handwriting recognition refers to the recognition of handwritten paper documents which are optically scanned.

The difficulty of recognition varies with a number of factors:

- Restrictions on the number of writers.
- Constraints on the writer: entering characters in boxes or in combs, lifting the pen between characters, observing a certain stroke order, entering strokes with a specific shape.
- Constraints on the language: limiting the number of symbols to be recognized, limiting the size of the vocabulary, limiting the syntax and/or the semantics.
- Many different applications currently exist, such as, check, form, mail or technical document processing. Whereas, online recognition systems are based on dynamic information acquired during the production of the handwriting.

- Malayalam Script

Malayalam is the principal language of the South Indian State of Kerala. It belongs to the southern group of Dravidian Languages. Malayalam is spoken by over 50 million people. The Malayalam character set compromises of 95 characters consisting of the following character types:

- Vowels
- Consonants
- Anuswaram, Visargam and Chandrakkala
- Chillu
- Consonant signs
- Vowel signs

There are 13 vowels, 36 consonants, 5 chillu, 4 consonant signs, 12 vowel signs, numbers and rest contributing to anuswaram etc.

Due to the peculiarities of the Malayalam language, developing a recognition system to recognize the variety of characters is a cumbersome process.

A variety of techniques of Pattern Recognition such as Template Matching, Neural Networks, Syntactical Analysis, Wavelet Theory, Hidden Markov Models, Bayesian Theory etc. have been explored to develop recognizers for different languages such as Latin, Chinese, Arabic etc.

The proposed method uses direction feature extraction techniques and Neural Networks to distinguish characters and accomplish recognition tasks.

Objectives

The main objectives of this paper are to develop a handwritten Malayalam word recognition system.

The two phases identified are:

i) To recognize Handwritten Malayalam character
ii) To develop Malayalam word recognition system

Neural Networks with back propagation algorithm is suggested for the recognition process. The input can be given either by using light pen model.

II.SYSTEM STUDY

The word is divided into different segments. The characters are written in separate panels. The features are extracted and given as input to a neural network. The characters are identified. The identified characters are obtained and are checked for word. A database of different words is stored. The written word is checked in the database and the appropriate Unicode of the characters are retrieved.

A. Modules identified

The entire system is divides into different modules. The various modules identified in character recognition are:

i) Preprocessing
ii) Feature extraction
iii) Zoning
iv) Training using Neural Networks
v) Character identification

B. Preprocessing

The preprocessing provide the acquired data I a suitable form for further processing. In this phase the input image is generally cleaned from noise and error caused by the acquisition process. A great number of well-defined algorithms for signal processing are currently used during the preprocessing phase. However, in handwriting recognition, the preprocessing deals with more specific problems than in other fields of pattern recognition. For example, the binarization (thresholding) of the image. Another problem arises in several applications in several applications of handwriting recognition is thinning. Here in preprocessing noise detection and normalization is done.

C. Noise detection

Incomplete Images are not considered and are not accepted for recognition. They are categorized to non recognizable.

D. Normalization

The size of the panel adopted is of 15*12 matrix. This is adopted writing area. The characters written in that area are accepted for recognition. The characters are shifted to that particular writing area.

E. Feature Extraction

Feature extraction is defined as the problem of extracting from the raw data the information, which is most relevant for classification purpose, in this sense of minimizing within the class pattern variably while enhancing the between the class pattern variability. It should be clear that different feature extraction methods fulfill these requirements to a varying degree, depending on the specific recognition problem and the available data. A feature extraction method that proves to be successful in one application domain may turn out to be not very useful in another domain.

Selection of feature extraction methods is probably a single most important factor in achieving high recognition performance. In addition the performance also depends on the type of classifier used. Different feature types may need different type classifiers. Also the choice of feature extraction methods limits or dictates the nature and output of preprocessing steps. Some feature extraction method work on grey level sub images of single characters, while other work on solid four or eight connected symbols segmented from the binary raster image, thinned symbols, skeletons or symbol contours. The following subsection explains the feature extraction technique adopted for the present work.

F. Direction feature extraction

The feature extraction method used in the proposed work is direction feature extraction. The line segments that would be determined in each character image were categorized in to four types: 1) Vertical lines 2) Horizontal lines 3) Right diagonal and 4) Left diagonal. Aside from these four line representations, the technique also located intersection points between each type of line. To facilitate the extraction of direction features, the following steps were required to prepare the character pattern:

1. Starting point and intersection point location
2. Distinguish individual line segments
3. Labeling line segment information

Starting point and intersection point location:

To locate the starting point of the character, the first black pixel in the lower left hand side of the image is found. The choice of this starting point is based on the fact that in cursive English hand writing, many characters begin in the lower left hand side. Subsequently, intersection points between line segments are marked. Intersection points are determined as being those foreground pixels that have more than two foreground pixel neighbors.

Distinguish individual line segments: As mentioned earlier, four types of line segments were to be distinguished as compromising each character pattern. The neighboring pixels along the thinned pattern/ character boundary were followed from the starting point to known intersection points. Upon arrival at each subsequent intersection, the algorithm conducted a search in a clockwise direction to determine the beginning and end of individual line segments. Hence, the commencement of a new line segment was located IF:

1. The previous direction was up-right or down-left AND the next direction is down-right or up-left OR
2. The previous direction is down-right or up-left AND the next direction is up-right or down-left OR
3. The direction of a line segment has been changed in more than three types of direction OR
4. The length of the previous direction type is greater than three pixels.

Labeling line segment information:

Once an individual line segment is located, the black pixels along the length of this segment are coded with a direction number as follows:

Vertical Segment –2,
Right diagonal line-3,
Horizontal line segment-4 and
Left diagonal line-5

The figure illustrates the process of making individual line segments.

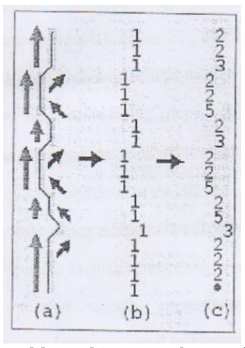


Fig1 (a) Original line, (b) Line in binary file, (c) After distinguishing directions

For example, Malayalam character 'ᓗ' can be drawn in the 15*12 panel as:

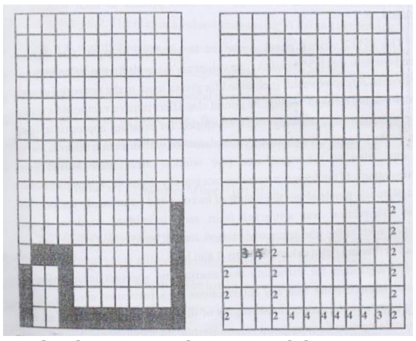


Fig 2 Sample Character & Character with line segment values

G. Zoning

In order to provide an input vector to the neural network the character representation was broken down into a number of windows of equal size(zoning) whereby the number, length and types of lines present in each window was determined.

The 15*12 writing panel is divided to windows of equal size. Here the proposed window size is 5*4 matrix. The values are assigned for the different types of line segments. A feature vector is obtained for giving input to the network Formation of feature vectors through zoning: As neural classifiers require vectors of a uniform size for training, a methodology was developed for creating appropriate feature vectors. In the first step, the character pattern marked with direction information was zoned into windows of equal size. If the image matrix was not equally divisible, it was padded with extra backgrounds pixels along the length of its row s and columns. In the next step, direction information was extracted from each individual window. Specific information such as the line segment direction, length, intersection points etc. were expressed as floating point values between -1 and 1.

The algorithm for extracting and storing line segment information first locates the starting point and any intersections in a particular window. It then proceeds to extract the number and lengths of line segments resulting in an input vector containing nine floating-point values. Each of the values compromising the input vector was defined as follows:

1. The presence of horizontal lines, 2. The total length of horizontal lines, 3. The presence of right diagonal lines, 4. The total length of right diagonal lines, 5. The presence of vertical lines, 6. The total length of vertical lines, 7. The presence of left diagonal lines, The total length of left diagonal lines and 9. The presence of intersection points.

As an example, the first floating point value represents the number of horizontal lines in a particular window. During processing, the number starts from 1.0 to represent "no line" in the window. If the window contains a horizontal line, the input decreases by 0.2. The reason a value commencing at 1.0 and decreasing by 0.2 was chosen was mainly because in preliminary experiments, it was found that the average number of line following a single direction in a particular window was 5. However in some cases, there were a small number of windows that contained more than five lines and hence in these cases the input vector contained some negative values. Hence values that tallied the number of line type in particular window were calculated as follows:

Value=1-(number of lines/10)(2)...................................(1)*

For each value that tallied the number of lines present in a particular window, a corresponding input value tallying the total length of the lines was also stored. To illustrate, the horizontal line length can be used as an example. The number starts at 0 to represent "no horizontal lines " in a particular window. If a window has a horizontal line, the input will increase by the length of the line divided by the maximum window length or window height, multiplied by two. The reason this formula is used, is because it is assumed that the maximum length of one single line type is two times the largest window size. As an example, if the line length is 7 pixels and the window size is 10 pixels by 13 pixels, then the line length will be 7/(13*2)=0.269.

$$\text{Length}=\frac{\text{number of pixels in a particular direction}}{(\text{Window height or width})*2}$$

The operations discussed above for the encoding of horizontal line information must be performed for the remainder of direction. The last input vector value represents the number of intersection points in the character.

It is calculated in same manner as for the number of lines present. The windows are of 5*4 matrix. Nine equal 5*4 windows are obtained from the 15*12 panel. The line segments are distinguished.

		2	
		2	
4	3	2	

Fig 3 Sample 5*4 zone

From each zone the 10 feature vector values are found. The feature vector for the above zone is as follows:

The number of horizontal line segment -1
The number of right diagonal line segment -1
The number of vertical line segment -3
The number of left diagonal line segment- Nil
The number of intersections – Nil

0.8	0.1	0.8	0.1	0.8	0.3	1	0.0	1	0.2

Fig 4 Feature Vector

Each of the 10 values of the 9 zones are obtained. So a total of 95 values are found. This will constitute the input vector to the neural network.

III. MULTILAYER PERCEPTRON

The most common neural network model is the multilayer Perceptron (MLP). This type of neural network is known as a supervised network because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to produce the output when the desired output is unknown. This is perhaps the most popular network architecture in use today and discussed at length in most neural network text books. The units each perform a biased weighted some of their inputs and pass this activation level through a transfer function to produce their output, and the units are arranged in a layered feed forward topology. The network thus has a simple interpretation as a form of input output model, with the weights and thresholds the free parameters of the model. Such networks can model functions of all most arbitrary complexity, with the number of layers and the number of units in each layer, determining the function complexity. Important issues in multi layer Perceptrons design include specification of the number of hidden layers and the number of units in these layers. The number of input and output units is defined by the problem.

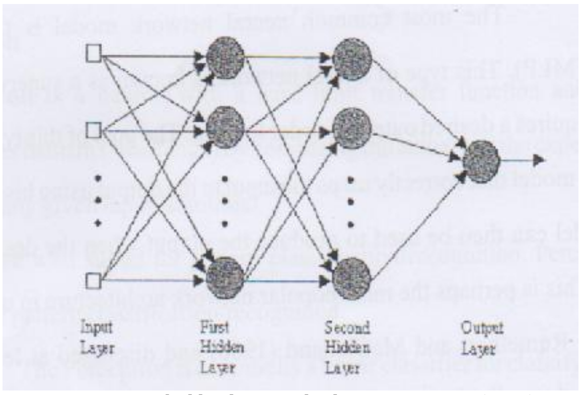


Figure 5 Two hidden layer multiplayer Perceptron (MLP)

The inputs are fed in to the input layer and get multiplied by interconnection weights as they are passed from the input layer to the first hidden layer. Within the first hidden layer, they get summed, and then processed by a nonlinear function (usually the hyperbolic tangent). As the processed data leaves the first hidden layer, again gets multiplied by interconnection weights, the summed and processed by the second hidden layer. Finally the data is multiplied by interconnection weights then processed one last time with in the output layer to produce the neural network.

The MLP and many other neural network learn using an algorithm called back propagation. With back propagation, the input data is repeatedly presented to the neural network. With each presentation the output of the neural network is compared to the desired output and an error is computed. This error is then fed back(back propagated) to the neural network and used to adjust the weights such that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output. This process is known as “training”.

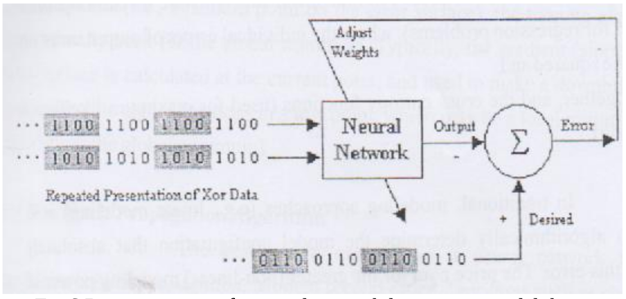


Fig 6 Demonstration of a neural network learning to model the exclusive-or (Xor) data

network. With each presentation, the error between the network output and the desired output is computed and fed back to the neural network. The neural network uses this error to adjust its weights such that the error will be decreased. This sequence of events is usually repeated until an acceptable error has been reached or until the network no longer appears to be learning.

A. Training Multilayer Perceptrons

Once the number of layers, and number of units in each layer, has been selected, the network's weights and thresholds must be set to minimize the prediction error made by the network. This is the role of the training algorithms. The historical cases that you have gathered are used to automatically adjust the weights thresholds in order to minimize this error. This process is equivalent to fitting the model represented by the network to the training data available. The error of a particular configuration of the network can be determined by running all the training cases through the network, comparing the actual output generated with the desired target outputs. The differences are combined together by an error function to give the network error. The most common error functions are the sum squared error (used for regression problems), where the individual errors of output units on each case are squared and summed together, and the cross entropy functions (used for maximum likelihood classification).

In traditional modeling approaches (e. g linear modeling) it is possible to algorithmically determine the model configuration that absolutely minimizes this error. The price paid for the grater (non – linear) modeling power of neural networks is that although we can adjust a network to lower its error, we can never sure that the error could not be lower still.

A helpful concept here is the error surface. Each of the weights and thresholds of the networks (i. e, the free parameters of the model) is taken to be a dimension in space. The N+*l* th dimension is the network error. For any possible configuration of weights the error can be plotted in the N+*l* th dimension forming an error surfacing. The objective of network training is to find the lowest point in this many dimensional surface.

In a linear model with some squared error function, this error surface is a parabola (a quadratic), which means that is smooth bowl –shape with single minimum it is therefore "easy" to locate the minimum.

Neural network error surfaces are much more complex , and characterized by a number of unhelpful features, such as local minimum (which lower than the surrounding terrain, but above the global minimum), flat-sports plateaus saddle –points, and long narrow ravines.

It is not possible to analytically determine where the global minimum of the error surface is, and so neural network training is essentially exploration of the error surface. From an initially random configuration of weights and thresholds (i.e, a random point on the error surface), the training algorithms incrementally seek for the global minimum. Typically, the gradient (slope) of the error surface is calculated at the current point, and used to make a downhill move. Eventually, the algorithm stops in a low point, which may be a local minimum (but hopefully is the global minimum)

B. The Back Propagation Algorithm

The best known example of a neural network training algorithm is back propagation. Modem second-order algorithms such as conjugate gradient descent and Levenberg - Marquardt are substantially faster (e.g, an order of magnitude faster) for many problems , but back propagation still has advantages in some circumstances, and is the easiest algorithm to understand. We will introduce this now, and discuss the more advanced algorithms later. There, also heuristic modifications of back propagation which work well for some problem domains, such as quick propagation and Delta- Bar delta

In back propagation, the gradient vector of the error surface is calculated. This vector points along the line of steepest descent from the current point, so we know that if we move along it a "shop" distances, we will decrease the error. A sequence of such moves (slowing as we near the bottom) will eventually find a minimum of some sort. The difficult part is to decide to how large the steps should be.

Large steps may converge more quickly but may also overstep the solution or(if the error surface is very eccentric) go off in the wrong direction. A classic example of this in neural network training is where the algorithm progress very slowly along a steep, narrow valley, bouncing from one side across to the other. In contrast very small steps may go in the correct direction but they also require a large number of iterations. In the practice, the step size is proportional to this slope (so that the algorithms settles down in a minimum) and to a special constant the learning rate. The correct setting for the learning rate is application- dependent, and is typically chosen by experiment:; it may also be time varying, getting smaller as the algorithm progresses.

The algorithm is also usually modified by inclusion of a momentum term: this encourage movement in a fixed direction, so that id several steps are taken in the same direction , the algorithm "picks up speed", which gives it the ability to(sometimes) escape local minimum, and also to move rapidly over flat spots and plateaus

The algorithm therefore progress iteratively, through a number of epochs. On each epoch, the training cases are each submitted in turn to the network, and target and actual outputs compared and the error calculated. This error together with the error surface gradient is used to adjust the weights, and then the process repeats. . The initial work configuration is random, and training stop when a given number of epochs elapses, or when the error reaches an acceptable level, or when the error stop improving (you can select which of these stopping conditions to use).

The back propagation network was probably the main reason behind the re popularization of neural networks after the publication of "learning internal Representations by Error Propagation" in 1986. The original network utilized multiple layers of weight-sum units of the type ***f=g(w'x+b***), where g was a sigmoid function or logistic function such as used in logistic regression. The employment of the chain rule of differentiation in deriving the appropriate parameter updates results in an algorithm that seems to "back propagate error', hence the nomenclature. However it is essentially a form of gradient descent. Determining the optimal parameters in a model of this type is not trivial, and steepest gradient descent methods cannot be replied upon to give the solution without a good. Starting point. In recent times networks with the same architecture as the back propagation network are referred to as Multi- Layer perceptions. This name does not impose any limitations on the type of algorithm used for learning.

The back propagation network generated much enthusiasm at the times and there was much controversy about whether such learning could be implemented in the brain or not partly because a mechanism for reverse signaling was not obvious at the time, but most importantly because there was no plausible source for the 'teaching' or 'target' signal.

C. Limitations

Multilayered networks are capable of performing just about any linear or nonlinear computation, and can approximate any reasonable function arbitrarily well. Such networks overcome. However, while the network being trained might theoretically be capable of performing correctly, back propagation and its various might not always find a solution.

Picking the learning rate for a nonlinear network is a challenge. As with linear networks, a learning rate that is too large leads to unstable learning. Conversely, a learning rate that is too small results in incredibly long training times. Unlike linear networks, there is no easy way of picking a good learning rate for nonlinear multilayer networks. With the faster training algorithms in the default parameter values normally perform adequately.

The error surface of a nonlinear is more complex than the error surface of a linear network. The problem is that nonlinear transfer functions in multilayer networks introduce many local minima in the error surface. As gradient descent is performed on the error surface it is possible for the network solution to become trapped in one of these local minima. This can happen, depending on the initial starting conditions. Settling in local minima in the error surface. As gradient descent is performed on the error surface it is possible for the network solution to become trapped in one of these local minima. This can happen, depending on the initial starting conditions. Settling in a local minimum can be good or bad depending on how close the local minimum is to the global minimum and how Iowan error is required. In any case, be cautioned that although a multilayer back propagation network with enough neuron can implement just about any function, back propagation does not always find the correct weights for the optimum solution. We might want to reinitialize the network and retrain several times to guarantee that you have the best solution.

Networks are also sensitive to the number of neurons in their hidden layers. Too few neurons can lead contribute to overfitting, in which all training points are well fitted, but the fitting curve oscillates wildly between these points.

IV. SYSTEM DESIGN

The entire system is divided to different sub sections. The word is written in the prescribed panels of size 15*12. Each character is pre processed. The line segments values are found are assigned at appropriate locations. The four line segments – horizontal, vertical, left diagonal and right diagonal are checked using appropriate rules.

Once the general direction of line segments was determined, a methodology was developed for creating appropriate feature vectors. In the first step, the character pattern marked with direction information was zoned into windows of equal size. In the next step, direction information was extracted from each individual window. Specific information such as the line segment direction, length, intersection points etc were expressed as floating point values between -0 and 1[10].

The characters are divided to different zone. Each zone is of size 5*4. From each zone the direction feature vector is found. The direction feature vector is the input to the neural network. Neural network is trained for character recognition. The words are stored in database. Unicode of each character is stored in the appropriate table. The Unicode of the hand written word is retrieved from the database

The goal of the DF was to simplify each character's boundary through identification of individual stroke or lines segments in the image

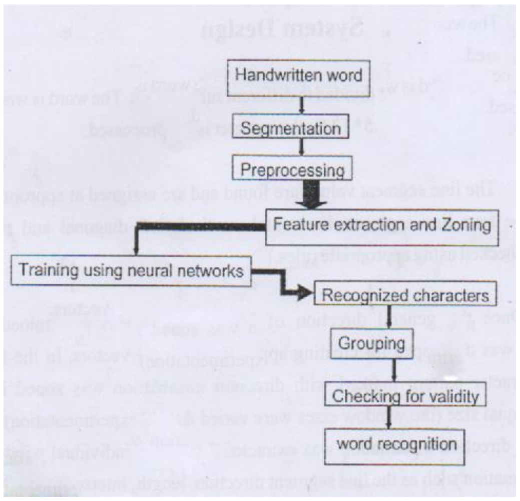


Fig 7 Block Diagram of the system

The neural classifier chosen for the task of character recognition were back- propagation (BP) networks. For experimentation purposes, the architectures were modified varying the number of inputs, outputs, hidden units, hidden layers and the various learning terms. The number of input each network was associated with the size of the feature vector for each image. Most successful vector configurations were of 81 for the direction feature.

Here a multilayer perception with back propagation algorithm is proposed for training or recognition of characters. There will one input layer, one hidden layer and one output layer. A three layer network is suggested

The number of nodes in the input in the input layer is 90, since 90 values are obtained from the 15*12 panel. These are floating point values between +1 and -1.

The hidden layer consists of 100 nodes and the output layer of 10 nodes.95 characters is identified. The back propagation network is trained with a large sample set of each character.

The invention of back propagation algorithm has played a large part in the resurgence of interest in the field of artificial neural networks. Back propagation is a systematic method of training multilayer artificial neural network. It has a mathematical foundation that is strong if not practical.

Neuron is used as the fundamental building block for back propagation networks. A set of inputs is applied, either from outside or from previous layer. Each of these multiplied by a weight, and the products are summed. This summation of products is termed NET and ust be calculated for each neuron in the network. After NET is calculated, the activation Function F is applied to modified it, there by producing the signal OUT

OUT= $1/(1+e^{-NET})$

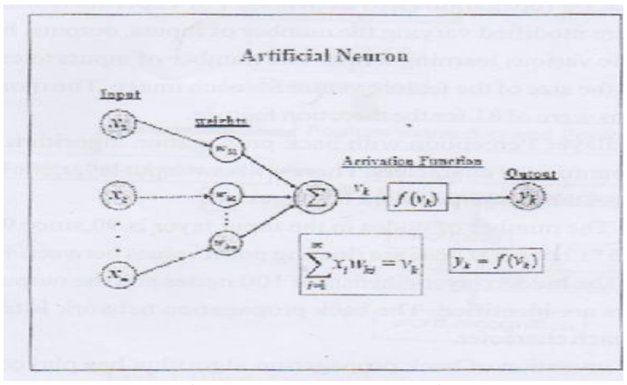


Fig 8 Artificial neuron with activation function

The input x, of the neuron consist of the variables "X., Xn and a bias term, known as the momentum constant, which is equal to one. Each of the input values is multiplied by a weight w_i, after which the results are added. On the result, a simple mathematical function.

F(x) , is performed. the function is also known as the activation function. The calculations the neuron performs are thus given by:

$Y= f(w_0+X_1*w_1+...+x_n* w_n)$

Numerous choices for the functions exist. Frequently used in implementations are sigmoid functions:

$F(u)= 1/|(1+e^{-u})$

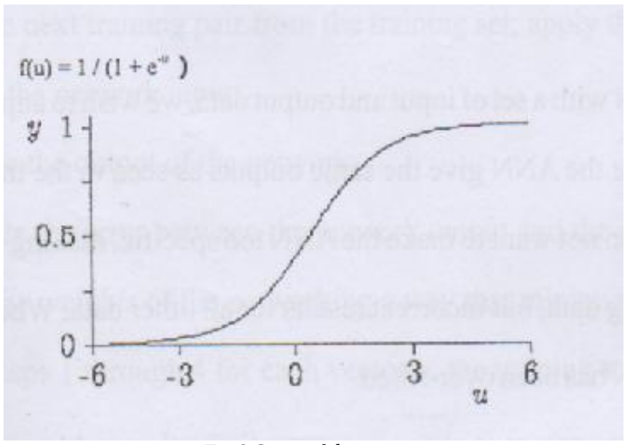


Fig 9 Sigmoid function

The objective of training the network is to adjust the weights so that the application of a set of input produces the desire set of outputs or reasons of brevity, this input- output set can be referred to as vectors. Training assumes that each input vectors is paired with a target vector representing the desire output together these are called training pair. Usually a network is trained over a number of training pairs. For example, the input part of a training pair might consists of a pattern of ones and zeros representing a binary image of a letter of the alphabet.

When training an ANN with a set of input and output data, we wish to adjust the weights in the ANN, to make the ANN give the same outputs as seen in the training data. On the other hand, we do not want to make the ANN too specific, making it give precise results for the training data, but incorrect results for all other data. When this happens, we say that the ANN has been over-fitted.

The training process can be seen as an optimization problem, where we wish to minimize the mean square error of the entire set of training data. This problem can be solved in many different, ranging from standard optimization heuristics like simulated annealing, through more special optimization techniques like genetic algorithms to specialized gradient descent algorithms like back propagation. The most used algorithm is the back propagation algorithm, but this algorithm has some limitations concerning, the extent of adjustment to the weights in each iteration. This problem has been solved in more advanced algorithms like PROP and quick prop before starting the training process, all of the weights must be initialized to small random numbers. This ensures that the network is not saturated by large values of weights, and prevents certain other training pathologies. For example if the weights all start at equal values and the desired performance requires unequal values, the network will not learn.

Training the back propagation network requires the steps that follow:

i) Select the next training pair from the training set; apply the input vector to the network input.

ii) Calculate the output of the network.

iii) Calculate the error between the network output and the desired output.

iv) Adjust the weights of the networking a way that minimizes the error.
v) Repeat steps 1 through 4 for each vector in the training set until the error for the entire set is acceptably low.

The back propagation algorithm works in much the same way as the name suggests: After propagating an input through the network, the error is calculated and the error is propagated back through the network while the weights are adjusted in order to make the error smaller.

Although we want to minimize the mean square error for all the training data, the most efficient way of doing this with the back propagation algorithm, is to train on a data sequentially one input at a time, instead of training on the combined data. However, this means that the order the data is given in is of importance, but it also provides a very efficient way of avoiding getting stuck in a local minima.

First the input is propagated through the ANN to the output. After this error^{ek} on a single output neuron ***k*** *can be calculated as:*

$$\boldsymbol{E_k=d_k-y_k}$$

Where $\boldsymbol{y_k}$ is the calculated output and $\boldsymbol{d_k}$ is the desired output of neuron ***k.*** This error value is used to calculate a $\boldsymbol{\delta_k}$ value, which is again used for adjusting the weights. The $\boldsymbol{\delta_k}$ value is calculated by:

$$\boldsymbol{\delta_k=e_k g'(y_k)}$$

Where ***g'*** is the derived activation function. When the $\boldsymbol{\delta_k}$ value is calculated, we can calculate the $\boldsymbol{\delta_j}$ values for preceding layers. The $\boldsymbol{\delta_j}$ values of the previous layer are calculated from the $\boldsymbol{\delta_k}$ values of this layer. By the following equation:

$$\boldsymbol{\delta j} = \eta g'(yj) \sum_{k-0}^{K} \boldsymbol{\delta hujh}$$

Where ***K*** is the number of neurons in this layer and $\boldsymbol{\eta}$ is the learning rate parameter, which determines how much the weight should be adjusted. The more advanced gradient descent algorithms does not use a learning rate, but a set of more advanced parameters that makes a more qualified that makes a more qualified guess to how much the weight should be adjusted.

Using these $\boldsymbol{\delta}$ values, the $\Delta\boldsymbol{W}$ values that the weights should be adjusted by, can be calculated by:

$$\Delta\, \boldsymbol{W_{jk} = \delta_j y_k}$$

The $\Delta /_{jk}$ value is used to adjust the weight $/_{jk}$ by $W_{jk}=W_{jk}+\Delta /_{jk}$ and the back propagation algorithm moves on to the next input and adjusts the weights according to the output. This process goes on until a certain stop criteria is reached. The stop criteria is typically determined by measuring the mean square error of the training data while training with the data, when this mean square error reaches a certain limit, the training is stopped. More advanced stopping criteria involving both training and testing data are also used.

The primary reason why neural networks are studied is because they have a remarkable ability to process imprecise data and deliver meaningful results. They can detect patterns or trends that are otherwise unnoticeable by other computer techniques. A neural network that has been trained to process a particular type of data may well be considered an expert in analyzing that data type. Further, it can enable us to speculate how the system would perform in different situations.

B. Proposed Network

The MLP network implemented for the purpose of this project is composed of 3 layers, one input, one hidden and one output as shown in the figure. The input layer constitutes of 90 neurons which receive pixel binary data from a 15*12 symbol pixel matrix. The size of this matrix was decided taking into consideration the average height and width of character image that can be mapped without introducing any significant pixel noise.

The hidden layer constitutes of neurons whose number is decided on the basis of optimal results on a trial and error basis.

The output layer is composed of neurons corresponding to each Malayalam characters. The target values are assigned for each character. To initialize the weights a random function was used to assign an initial random number which lies between two present integers named. The weight bias is selected from trial and error observation to correspond to average weights for quick convergence.

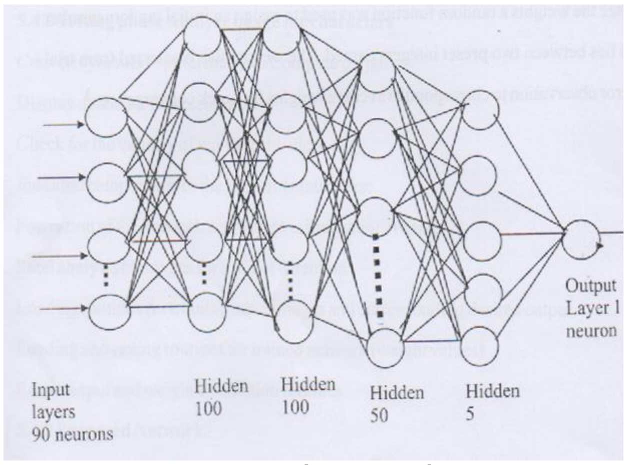


Fig 10 Proposed MLP Network

C. Procedure Steps

S1: Create a panel of size with 15 rows and 12 columns
S2: If any mouse event is found, set the pixel value to 1
S3: Check the connectivity of pixels
S4: Grab the pixels and set the corresponding array values to ones and zeros
S5: Divide the panel to nine equally zoned matrices of size 5 rows and 4 columns
S6: Feature extraction

For each matrix of size 5*4 do the following

Find the horizontal, vertical, right diagonal, left diagonal lines and intersections by checking immediate pixels
See the following values for the line segments
Vertical-2
Right diagonal-3
Horizontal-4
Left diagonal-5

S7: Find the input vector to the network from the nine zones
S8: Design a Neural Network with 90 input nodes, 100 hidden nodes and 10 output nodes
S9: Initially generate the weights for the links, randomly
S10: Set the target values for output nodes
S11: Train each character with samples and make the network to learn

D. Procedure steps for training routine

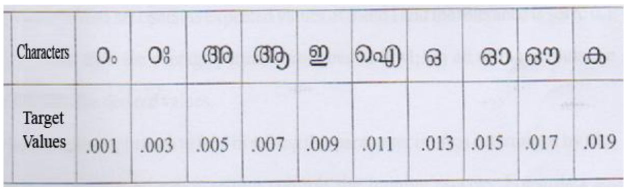
S1: Analyze the input character and find the zones
S2: Read the desired output sample from the database
S3: Add the pattern
S4: Do the training and learning process with the sample
S5: For each character do the following
Calculate the output of the back propagation network
Compare the obtained output with the target value of that value of that character
Compute the error
Back propagate the error across each link and adjust the weight value between the nodes
Repeat for all steps
S6: Check whether the error value is minimum. If yes, then exit. Otherwise continue the process

E. Procedure for word recognition

S1: Obtain the recognized character
S2: Group the character to form a word
S3: Create a database for a specific number of words
S4: Compare the written word with the word stored in the table
S5: If the word is fund display the appropriate Unicode of the characters

F. Setting the target values

The target values for the 10 output neurons vary within the range .001 to 0.19 separated by a difference of .002 between successive ones as shown in Fig 11.



Characters	○ം	○ഃ	അ	ആ	ഇ	ഐ	ഒ	ഓ	ഔ	ക
Target Values	.001	.003	.005	.007	.009	.011	.013	.015	.017	.019

Fig 11

The error of ith output neuron
$\Delta_i = Out_i * (1 - Out_i) * (Target_i - OUT_i)$ ------------------------ *(1)*
The The error of jth hidden layer neuron is
$\Delta H_j = OutH_j * (Target_i - OUT_i) * \sum^{10}_{i=1} (\Delta_i * Wt_{ji})$ ------------ *(2)*

New weight between jth hidden neuron and ith output neuron is
$Wt_{ji} = Wt_{ji} + \eta * \Delta_i * OutH_j$-- *(3)*
The new weight between kth input neuron and jth hidden neuron is
$Wt_{kj} = Wt_{kj} + (\eta * \Delta H_j * I_k)$--*(4)*

Out_i is the output value of ith neuron
$Target_i$ is the target value of the ith neuron
$OutH_i$ is the output value of jth hidden layer neuron is the learning rate.

The learning rate used in the project is .01 this is finalized by trial and error method. The error tolerance is .000001.

The error in a particular iteration is back propagated only if it is greater than the error Tolerance. Typically error tolerance is a small value in the range 0 to 1.

Sets the value for the acceptable difference between the desire output value and the actual output value. This must be real value between 0.0 and 1.0. for example, if young training data set contains expected values of 0 and I and the tolerance is set to 0.1 (the default), then the average pattern error goes to 0 wl; 1 en all of the output are within 0.1 of the desire values

The best approach to take in setting this parameter is often determined by trial and error. The error tolerance setting controls the training process. For a data set containing binary targets (0,1), the tolerance parameter is usually set to 0.1.This means that the output is considered "good" when it is within 0.1 of the desire output that is, 0.9 for a 1, 0.1 for a 0) . When every output is within the tolerance range of the desire output value, the network status is changed to LOCKED and weights updates.

V. CONCLUSION

In this paper a new feature extraction technique (direction feature) for the recognition of Malayalam character is used. Input can be given through light pen model. The system successfully recognizes the characters. The obtained word is compared with the database stored for validity.

Neural Network with backpropogation algorithm is explored in depth. Various other approaches for character recognition is also studies in detail. Malayalam characters are found to be recognized successfully.

It was discovered that a better approach could be developed employing neural network techniques in recognizing characters, one of the major advantages of using neural networks is their inherent ability to respond to variations. This ability is important in particular where handwriting is concerned. This approach can be applied to printed letter also.

REFERENCES

1. Programming Windows by Charley Petzold
2. Microsoft Visual C++.Net step by step
3. Dr. Suneetha Agarwal, Vikas Kumar, Online Character Recognition: IEEE2005
4. IBM R. esea.rch, T. J Watson Research center, Yorktown Heights, New York centre for artificial Intelligence, Ka.ist University, Soul, South Korea, ON_LINE HANDWRITTEN CHARACTER RECOGNITION USING PARALLEL NEURAL NETWORKS: 1994 IEEE
5. Jack. M. Zurada, Introduction to Artificial Neural Systems
6. M. Blumenstein,B. Verma amd H. Basli,A Novel Feature Extraction Technique for the Recognition of statement of segmented Handwritten Characters: Proceeding of CDAR2003
7. Philip Wasserman: Neural Computing
8. Samuel Talleux, Vedat and Emir Tufan, Handwritten character recognition using steerable filters and Neural Networks: 1998IEEE
9. Seethalakshmi , R, SreeRajani, Balachandran, Optical Character Recognition for printed Tamil text using Unicode: JZUS2005
10. Sutat Sea-Tang and Ithipan Methaste, Thai Online Handwritten; t: 1 Character Recognition Using Windowing Back Propagation Neural Networks: ISCIT2004
11. The State- of- the Art: Cheng- Lin Liu, Stefan Jaeger and Maski Makagawa, Online recognition of Chinese Characters: IEEE2004
12. VS. Roshni, Shanifa BEEVI and Revathy, Machine Recognition of Printed Malayalam Characters: Proceeding of the International Conference of Cognition and Recognition, December2005