CHAPTER 7

RECOGNIZING HAND SIGNS
7.1 MOTIVATION AND APPLICATIONS

Since the introduction of computers, not much has changed as far as input to computer system is concerned. This is probably due to the fact that the existing input devices are adequate. Due to advancement in hardware technology new applications and improved hardware are regularly made available. Input to computers is by means of keyboards, mice, light pen, trackball, keypads etc. Users are familiar with these but these devices inherently limit the speed and naturalness of interaction with the computer.

Vision based interfaces are available for computers. At the present moment the computer is able to “see”. Hence users are allowed far richer and user-friendly man-machine interaction.

Computer recognition of hand signs may provide a more natural-computer interface, allowing people to point, or rotate a CAD model by rotating their hands or even operate machines with this.

The motivation for this work is provided by the ongoing surge in interest in recognizing human hand signs. Hand sign recognition has various applications like computer games, machinery control (e.g. crane), and mouse replacement.

One common goal of human-computer interaction is direct manipulation, that is, the user can interact with the objects of the system directly instead of through an intermediary interface. Direct manipulation reduces cognitive load and makes building a mental model of the system user friendly. A major problem that remains is the ability to interact with the system in a useful manner. Keyboard control is far too abstract for most interactive applications, mouse control is inherently 2-dimensional and voice control is still fairly limited to narrow subjects, if reliable function is required. Some applications require many degrees of freedom, but to control all of them using
joysticks, dials or mice is daunting task for most people and require specific training.

This is especially problematic for immersive applications or where a high degree of intuitiveness is required. In an immersive application the user cannot see the keyboard or other interaction tools unless they are given a virtual representation, which is often an unsatisfactory solution. Another problem is that applications like virtual or enhanced reality are intended to be as natural and free-form for previous training or a high level of symbolism. The ideal interaction method would be intuitive, explorative and direct.

Sign interaction appears to be a promising paradigm for such applications. Hand sign is a hand held in a simple configuration and pose. Signs are an important part of daily human communication, and easily understood at least by other humans. While their exact execution varies, their meaning is more or less fixed, at least within the culture of the originator. The goal for virtual reality is to provide an interface to a virtual environment similar to reality in some aspects, and this makes gesture interaction attractive. It is hoped that by using signs, the user can transfer skills into the virtual environment and the learning effort required to master the interface could be lessened.

An attempt has been made to recognize signs made by hand using artificial neural networks. MATLAB has been used for this work.

7.2 METHODOLOGY

The aim of this attempt is to develop a method to recognize hand signs, based on a pattern recognition technique named Orientation Histogram; employing histograms of local orientation. The recognition process uses two-stage classification approach, which is a hybrid of local orientation technique and artificial neural networks. The orientation histogram is used as a feature vector for sign classification and interpolation. Feature vectors are then passed to artificial neural network for training and recognition of signs.

7.3 STEPS FOR RECOGNITION

1. Image Database Planning
2. Image Preprocessing
3. Feature Vector Extraction
4. Sign Recognition

A block diagram of the general structure of the method is shown in figure 7.1.

![Figure 7.1: Block Diagram of Technique Used](image)

7.3.1 Image Database Planning

A database is created consisting of images which are used for training and testing the system. Database can comprise of either hand drawn images, digitized or scanned photographs etc. Digital photographs have been used for creating the database, as they are the most realistic approach. Image database was created using a digital camera with resolution of 640x480 pixels.

The database compilation depends on the type of application it is put to use. If an application is a crane controller for example, operated by the same person for long periods, the algorithm doesn’t have to be robust on different person’s images. The applications can be of many forms and since the attempt is being made for no specific application, different images have been experimented with. Finally the database has been created by:

- **Training Set**: It consists of thirty six sets of images, each set consisting of twenty images. Out of these thirty six sets, ten sets of images form the training set for the numerical character sign recognition network and the remaining twenty six sets of images comprise the training set for alphabet sign recognition network.

- **Testing Set**: It contains thirty-six sets of ten images each.
7.3.2 Image Preprocessing

The image data requires preprocessing for adjusting the variations due to light and making the image size compatible to the system. Preprocessing includes resizing, converting to grayscale and rotating the image etc.

1. **Image Resizing**: The images included in the database are resized to a fixed size to have uniform dataset. The images are resized to a fixed resolution of 160x120 pixels. This resizing is necessary because otherwise the image database data created by the program code in MATLAB would become too heavy and will slow down the processing. Also, the resized image is adequate enough for recognition.

2. **Conversion to Gray Scale**: The images resized in previous step are converted to grayscale to have uniform backgrounds.

3. **Rotating the Images**: Some images were rotated to overcome the change in the angle of capturing such images.

7.3.2.1 Vector Extraction

Recognition should not be affected by the position of the hand in the image for which a histogram is calculated as to how often each gradient orientation element occurs in image. Thus each orientation element is treated the same, i.e., independent of its location.

![Figure 7.2: Pattern Recognition System](image)

The pattern recognition method used in the system is shown in figure 7.2. In the system, some transformation T, converts an image into a feature...
vector. In the implemented method, transformation $T$ is the orientation histogram technique. The input image is transformed into corresponding image gradients using this technique. Then, by taking the inverse of tangent, corresponding orientation of each gradient element is calculated. These gradient orientations constitute the required feature vector, which is then compared with feature vectors of a training set of signs.

The local orientation is calculated using image gradients. Two $2 \times$ tap $x$ and $y$ derivative filters have been used for calculating image gradients [10]. The tapping of filter defines the number of pixels being captured by the filter at one instance. The outputs of the $x$ and $y$ derivative operators are $dx$ and $dy$. The gradient direction (orientation) is given by $\arctan(dx, dy)$. Gradient direction or edge orientation is the feature vector which is fed to the neural network. Another feature that could have been extracted from the image would be the gradient magnitude using the formula

$$\text{Magnitude} = \sqrt{dx^2 + dy^2}$$

But using gradient magnitude would allow testing the algorithm only with similar images. Once the image has been processed, the output will be a single vector containing a number of elements equal to the number of bins of the orientation histogram.

### 7.3.2.2 Operation

The entire operation is divided into steps and is described here:

**Step 1 Presenting Training Data**

Firstly, the training image database is input to the MATLAB memory. The user can choose the sign for which he/she wants to train the system. All the training images pertaining to the selected sign class are presented to the memory.

**Step 2 Gradient Orientation Calculations**

The images read in step 1 are resized to $160 \times 120$ pixels. From the resized images, image gradients $dx$ and $dy$ are calculated. For calculating image gradients, image matrix is convolved with 2-tap derivative filters. For
the x direction $x = [-1 1]$ is used. For the y direction $y = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$, which is the same as $x$ but transposed and multiplied by $-1$. The resulting matrices of $dx$ and $dy$ are divided element by element and then inverse tangent ($\tan^{-1}$) of $dx/dy$ is calculated. This gives the gradient orientation.

**Step 3 Histogram Preparation**

Gradient orientation matrix obtained in step 2 is scanned for angular values ranging from $-90^\circ$ to $90^\circ$ to create orientation histogram. This is because for real elements of $x$, $\tan(x)$ is in the range of $-90^\circ$ to $+90^\circ$.

Determining the number of the histogram bins is an important issue as this defines the differentiation among the images. The histogram bins constitute the input vector to the neural network. Smaller the vector the faster is the processing.

### 7.3.2.3 Sign Recognition

The orientation histogram vector extracted in the previous step is the input to the neural network implemented in this step. This dissertation is intended to examine the possibility of a system based around the visual recognition of hand signs. The hand sign images vary in quality, position and rotation. Therefore, the ability to recognize different instances of the same sign is important. The Cognitron’s ability to classify different images into the same classes as a human, suggests that this network provides a very good basis for recognition of hand signs. The cognitron network developed here comprises of three stages. First stage has a pre-processing layer which consists of neurons equal to the number of elements in the feature vector. Second stage has a layer of inhibitory and excitatory neurons feeding to the output layer. This output layer is the classifying layer which classifies the input vector.

### 7.4 NEURAL NETWORK

Neural network developed for hand sign recognition is based upon Cognitron network. Cognitron has been developed by Fukushima [86].
7.4.1 Structure

The cognitron is constructed of layers of neurons connected by synapses. As shown in figure 7.3 a presynaptic neuron in one layer feeds a postsynaptic neuron in the next layer. There are two types of neurons: Excitatory cells, which tend to cause the postsynaptic cell to fire, and inhibitory cells, which reduce this tendency. The firing of a neuron depends upon the weighted sums of its excitatory and inhibitory inputs; however, the actual mechanism is more complex than simple summation.

Figure 7.3: Presynaptic and Postsynaptic Neurons

Figure 7.4 shows that each neuron connects only to neurons in the nearby area, called its connection region. This limited region is consistent with the structure of the visual cortex, where connections are seldom made between neurons farther than one millimeter apart.
The output of the excitatory neuron is determined by the ratio of its excitatory inputs to inhibitory inputs.

The total excitatory input to a neuron $E$ is simply the weighted sum of the excitatory inputs from the neurons in the previous layer. Similarly, the total inhibitory input $I$ is the weighted sum of the inhibiting inputs from the same neurons. Mathematically these are expressed as,

$$E = \sum a_i u_i \quad (7.1)$$
$$I = \sum b_j v_j \quad (7.2)$$

Where,
- $a_i$ = the weight of the $i$th excitatory synapse
- $u_i$ = the $i$th excitatory input
- $b_j$ = the weight of the $j$th inhibitory synapse
- $v_j$ = the $j$th inhibitory input
Figure 7.5: Connection and Competition regions

The output of a neuron is then calculated as follows:

\[
NET = \frac{(1+E)}{(1+I)} - 1 \quad (7.3)
\]
\[
OUT = NET \text{ for } NET \geq 0 \quad (7.4)
\]
\[
OUT = 0 \text{ for } NET < 0 \quad (7.5)
\]
\[
OUT = \frac{(E - I)}{(1+I)}, \text{ for } +ve \text{ NET} \quad (7.6)
\]

When the inhibitory input is small (I<<1), OUT can be approximated as 
\[
OUT = E - I, \text{ which agrees with the formula for a conventional linear threshold element (with a threshold of zero).}
\]

Hence large excitatory and inhibitory inputs result in:
\[
OUT = \frac{(E/I)}{1} - 1 \text{ if } E\gg1 \text{ and } I\gg1 \quad (7.7)
\]

In this case, OUT is determined by the ratio of the excitatory inputs to inhibitory inputs, rather than by their difference. The OUT is limited, provided that both types of input increase or decrease at the same rate X.

**The Inhibitory Neuron**

In the cognitron, a layer consists of both excitatory and inhibitory cells. As shown in figure 7.4, a layer 2 neuron has a connection region over which it has synaptic connections to a set of layer 1 neuron outputs. Similarly, in layer 1,
there is an inhibitory neuron with the same connection region. Synaptic weights coming into inhibitory cells are not modified during training; their weights are preselected so that the sum of the weights into any inhibitory neuron is equal to one. With this restriction, the output of the inhibitory cell INHIB is simply the weighted sum of its inputs, which in this case is the arithmetic mean of the excitatory outputs to which it connects. Hence,

\[ \text{INHIB} = \sum c \cdot \text{OUT}_i \]  

(7.8)

Where,

\[ \sum c = 1 \]

\[ \text{OUT}_i = \text{inhibitory input } i \]
Lateral Inhibition

In figure 7.6, every layer 2 neuron is shown to receive lateral inhibition from neurons in its competition region. An inhibitory neuron, shown as black, sums inputs from all neurons in the competition region and outputs a signal that tends to inhibit the target neuron. This method is functional, but computationally slow. It comprises a large feedback system involving every neuron in a layer; many computational iterations are required for it to stabilize.

Figure 7.7: Improved Lateral Inhibition

To accelerate calculations, Fukushima proposed a method of forward lateral inhibition. As shown in figure 7.7 above, an additional lateral inhibition cell processes the output of each excitatory cell to simulate the desired lateral inhibition. First, it defines a signal equal to the total inhibitory influence in a competition region as follows:

$$\text{LAT\_INHIB} = \sum g_i \text{OUT}_i \quad (7.9)$$

Where,

- \(\text{OUT}_i\) = the output of the \(i\)th neuron in a competition region
- \(g_i\) = the weight from that neuron to the lateral-inhibition neuron
- \(g_i\) is so chosen that \(\sum g_i = 1\).
The output of the inhibitory neuron is then calculated as follows:

\[
OUT' = \left[\frac{1 + OUT_i}{1 + LAT_{INHIB}}\right] - 1 \quad (7.10)
\]

By utilizing this relationship, the cognitron neuron closely emulates the responses of the biological neuron.

Using lateral inhibition layer requires use of a large number of inhibitory neurons as given by the formula:

No. of inhib neurons = no. of lat_inhib neurons x no. of neurons in the input layer.

So, using lateral inhibition mechanism is feasible only in large sized and dedicated systems where features are directly extracted from the image pixel information.

### 7.4.2 Network Architecture

The neural network architecture used for hand sign recognition is shown in figure 7.8 which consists of three layers.

**Figure 7.8: Network Architecture**

The neural network architecture used for hand sign recognition is shown in figure 7.8 which consists of three layers.
The input layer consists of 36 input nodes corresponding to 36 feature elements in the input vector. The number of output nodes is 10 for the network developed for numeral sign recognition network and the number of output nodes is 26 for alphabet sign recognition network.

The neural network model for implementing cognitron in MATLAB has been developed in two parts. Firstly, the input vector is divided into connection regions whose width is defined by the number of hidden neurons. For example, with 27 hidden neurons, the input vector is divided into 27 connection regions with each having a width of 10 elements. Now, to calculate the INHIBITORY neuron output, each connection region is averaged with the formula given below:

$$I = \sum c_i \text{OUT}_i$$  \hspace{1cm} (7.11)

Where,

- \text{OUT}_i = i_n, inhibitory input
- c_i = the weight connecting i_n input to the inhibitory neuron
- \sum c_i = 1

In the second part, a neural network is developed with an input layer, a hidden layer comprising of two sets of neurons and an output layer. The input layer consists of 36 neurons corresponding to 36 elements in the feature vector. The hidden layer consists of excitatory neurons arranged in two groups. To the first group of excitatory neurons, the input vector is fed in the form of connection regions as described above. These neurons output the excitatory input for the next set of excitatory neurons as given by:

$$E = \sum a_i u_i$$  \hspace{1cm} (7.12)

Where,

- a_i = the weight of the /th excitatory synapse
- u_i = the output from the /th input neuron

To the second set of excitatory neurons present in the hidden layer, INHIB output vector is also fed as an input. At this second set of neurons, the output of neuron given by following formula is calculated.

$$\text{OUT} = (E - I)/ (1+I)$$  \hspace{1cm} (7.13)
This output is then propagated to the output layer consisting of 10 neurons (numerical sign recognition network) and 26 neurons (alphabet sign recognition network) for classification.

7.5 TRAINING THE ARTIFICIAL NEURAL NETWORK

The neural networks are trained in MATLAB by varying the learning parameters and network size (number of hidden nodes). The different learning parameters which are varied to obtain the required accuracy are discussed below. The different networks are configured and tested by varying the following parameters.

7.5.1 Learning Rate

Learning rate can limit as well as expand the extent of weight adjustments in a training cycle. A high learning rate reacts quickly to input changes, and can make networks unstable. If the rate is too high, the changes can be too extreme and cripple the network’s prediction ability. However, if the learning rate is too low, the network training time is substantially increased. A high learning rate is useful to accelerate learning until the weight adjustments begin to stagnate. However, the higher learning rate increases the risk that the weight search jumps over a minimum error condition, which could jeopardize the integrity of the network and cause back propagation learning to fail.

The learning rate for the current work has been varied over the range from 0 to 0.9.

7.5.2 Momentum

Momentum describes the proportion of the weight change that is added to each subsequent weight change. Low momentum causes weight oscillation and instability, preventing the network from learning. High momentum cripples network adaptability. For stable back propagation, the momentum factor should be kept less than one (unity). Momentum factors close to unity are needed to smooth error oscillations when they occur.

In this work, momentum has been varied over the range of 0 to 0.6.
7.5.3 Number of Hidden Nodes

Good accuracy is attributed to the choice of number of hidden nodes. With less number of hidden nodes, the network takes more number of cycles to converge. More number of cycles means slow but efficient learning. The convergence, which is finally achieved, is not a result of any local minima but the final stable convergence. With increasing number of hidden nodes, network learns faster. Learning is faster because more weights means more elements contributing towards output, which consequently means easy convergence. At this point convergence may result due to local minima.

There is no hard and fast rule for determining the appropriate number of neurons in the hidden layer. As a general rule, the average of the total number of neurons present in the input layer and the output layer has been taken as the median for the range over which the number of hidden nodes would be varied.

The average of the number of input and output neurons for Numeral Sign Recognition Network is \((36+10)/2\) = 23. So, the number of hidden neurons has been varied from 18 to 27 for the hidden layer of Numeral Sign Recognition Network. Similarly, the average of the number of input and output neurons for Alphabet Sign Recognition Network is \((36+26)/2\) = 31. So, the number of hidden neurons has been varied from 26 to 34.

7.6 DETERMINATION OF ACCURACY

The accuracy of neural network is found by presenting the network with test data. The test data consists of 360 different patterns from 36 different signs. Number of correct classifications divided by total number of test patterns gives the accuracy. The network is trained by varying different network parameters and then tested. The network with highest degree of accuracy is chosen for the recognition of hand signs. The parameters which have been taken into consideration are learning rate, momentum, accuracy and the number of neurons in hidden layer. MATLAB has been used extensively.
7.7 RESULTS

The objective of the work was to recognize Hand Signs in images with highest possible degree of accuracy. To achieve this, feature vectors are extracted from training and testing patterns sign images using MATLAB. The feature vectors thus extracted are gathered and compiled as training and testing data. Using the training data as input to neural network, recognition results have been obtained. Results are in the form of '.MAT' files saved from MATLAB. These files contain absolute values of outputs of network, number and values of weights, input training patterns and cycle by cycle learning process. A complete compilation of results is in the CD-ROM attached with this dissertation. Graphical representations of these results are presented here with explanations.

Sign image set includes signs for both numerical characters and alphabets. The recognition program was developed on an Intel Pentium 4 based computer system. This system houses 80 GB hard disk memory and 512 MB SDRAM. Due to hardware constraints, two separate neural networks were developed, one for Numerical Characters and the other for Alphabets. The results have been compiled accordingly.

For recognition of hand signs from images, Cognitron network with Back-propagation algorithm has been used. The back-propagation algorithm has two important parameters: learning rate and momentum. These parameters affect the convergence of the neural network. The optimum value of these two parameters for the neural network has been decided by testing different configuration of learning rate and momentum and analyzing the accuracy. Along with these two parameters, an important parameter is the network size. The network size means the number of hidden neurons.

7.7.1 Number of Hidden Nodes

The input nodes in the input layer and the output nodes in the output layer in a neural network are decided prior to the training of neural network, as the information about them is available. But the number of hidden nodes in hidden layer has to be decided by trial and observation method. Large number
of hidden nodes may result in network complexity and small number of nodes may result in slow convergence of network. To find an optimum solution the network was trained with different number of nodes in hidden layer. There is an effect of these on accuracy.

Good accuracy for less number of hidden nodes is attributed to the fact that the network takes more number of cycles to learn. More number of cycles means slow but efficient learning. The convergence, which is finally achieved, is not a result of any local minima but the final stable convergence. With increasing number of hidden nodes, the number of weights also increases. More number of weights means faster learning. Learning is faster because more weights means more elements contributing towards output, which consequently means easy convergence. But sometimes faster learning converges to local minima, resulting in a decrease in accuracy. The larger the number of the hidden nodes, higher is the hardware requirement.

The neural network was trained with different number of nodes in the hidden layer with momentum kept constant at 0.1 and learning rate also maintained constant at 0.1 for both Numerical Sign and Alphabet Sign recognition neural networks. To study the effect of hidden nodes, the two parameters are kept constant and the number of hidden nodes is varied from 18 to 27 for numerical characters and 26 to 34 for alphabets. Figure 7.9 and figure 7.10 show the effect of the number of nodes in hidden layer on the accuracy of recognition under test conditions.

From figure 7.9 it is observed that with increasing number of hidden nodes the network converges faster. In each case the recognition accuracy has been calculated as the number of characters correctly recognized over total number of characters in the testing set. Though for all the values of hidden nodes fairly good accuracy is obtained, the network with lesser number of hidden nodes provide better recognition accuracy.
The best accuracy of 94% is obtained for the numerical character recognition network with 21 hidden nodes with the limits of target error of 0.005. The figure also shows the number of epochs required for achieving desired target error.

The response of the Alphabet Recognition network with varying number of hidden nodes is observed and shown in figure 7.10.

It is observed from the figure that with the increasing number of hidden nodes, the networks converge faster. The best accuracy of 91.54% is obtained for the network with 28 hidden nodes with the limits of target error of 0.005, while learning rate and momentum are kept constant at 0.1 and 0.1 respectively. The variation in the number of epochs required is also depicted.

Figure 7.9: Variation in Accuracy (%age) and Number of Epochs with Change in Number of Hidden Nodes for Constant Learning Rate of 0.1 and Momentum of 0.1 and Target Error of 0.005 for Numerical Sign Recognition Network
7.7.2 Momentum

To study the effect of momentum, the neural networks were trained with different values of momentum keeping learning rate and no. of neurons constant. For the target error of 0.005, it is discussed in section 7.7.1 that the maximum accuracy is obtained with no. of hidden neurons set at 21 for Numerical Sign Recognition (figure 7.9) and with no. of hidden neurons set at 28 for Alphabet recognition. The effect of momentum is observed for constant learning rate of 0.1 and constant no. of hidden neurons 21 (for Numerical Character recognition) and the variation of percentage of accuracy and number of epochs with change in momentum is plotted in figure 7.11.
As can be seen from the bar chart of figure 7.11, the number of training cycles decreases for a range of momentum from 0.05 to 0.6. It can be observed that the maximum value of accuracy of 96% is obtained at momentum of 0.5, while the number of cycles to train the network is considerably low. Hence it can be concluded that for achieving target error of 0.005, the no. of hidden neurons at 21 and the momentum of 0.5 provide maximum accuracy of 96% for numerical characters.

In section 7.7.1, it is observed that for a target error of 0.005, the best accuracy is obtained with 28 hidden neurons, for alphabets (figure 7.10) while momentum has been kept constant at 0.1. In figure 7.12 the variation of accuracy and number of epochs is plotted with change in momentum for a constant learning rate of 0.1 for alphabet recognition network.

From the figure 7.12, it is observed that the network converges with decrease in number of cycles. Though for all the values of momentum fairly
good accuracies are obtained, the maximum value of accuracy is achieved at the momentum of 0.4. Hence it can be concluded that for achieving target error of 0.005, the learning rate of 0.1 and the momentum of 0.4 provide maximum accuracy of 95.77% for alphabet recognition network.

Figure 7.12: Variation in Accuracy (%age) and Number of Epochs with Change in Momentum for Constant Learning Rate of 0.1 and No. of Hidden Nodes constant at 28 and Target Error of 0.005 for Alphabet Sign Recognition Network

7.7.3 Learning Rate

The high value of learning rate depicts that network learns faster, whereas low value of learning rate indicates that learning will be slow, i.e. a large number of cycles are required for learning. However, small value of this parameter is preferred to avoid local minima and saturation problem. Similarly if the momentum parameter is high the learning boosts up further. It is used for avoiding weight oscillations. So, the values for these parameters must be decided only after fair amount of trials and observations.

The neural networks were trained for different learning rates at constant momentum of 0.5 (for numerical character recognition network) and 0.4 (for alphabet recognition network) and no. of hidden neurons set at 21 (for
numerical characters recognition network) and 28 (for alphabet recognition network) and target error of 0.005.

The bar chart in figure 7.13 shows the variation in accuracy and number of cycles with change in learning rate at constant momentum of 0.1 and target error of 0.005.

From the above bar chart it can be observed that with increase in learning rate the required number of cycles decrease rapidly resulting fast convergence of network to solution. But, simultaneously the accuracy decreases for the larger values of learning rate. It is observed that the best accuracy of 98% is obtained for learning rate 0.4 while momentum and target error have been kept constant at 0.5 and 0.005 respectively.

The accuracy obtained for alphabet recognition with target error of 0.005 and constant momentum of 0.4 has also been observed and plotted in figure 7.14 for different values of learning rates.
From the bar chart of figure 7.14, it can be observed that keeping momentum fixed with value 0.4, if we increase learning rate number of cycles decreases rapidly to converge to a target error of 0.005. But, in the same time the accuracy of recognition decreases up to a certain point. In this case the optimum value of accuracy of 97.31% is obtained when the value of learning rate is set to 0.5 and the network converges after 458 cycles.

![Variation in Accuracy (%age) and No. of Epochs with Change in Learning Rate](image)

**Figure 7.14: Variation in Accuracy (%age) and Number of Cycles with Change in Learning Rate for Constant Momentum of 0.4 and No. of Hidden Nodes Constant at 28 and Target Error of 0.005 for Alphabet Sign Recognition Network**

From figure 7.13 and figure 7.14, it is observed that the maximum accuracy of 98% is obtained for numerical character recognition with learning rate at 0.4 and a maximum recognition accuracy of 97.31% for alphabets with lower target error of 0.005.

**7.7.4 Recognition Accuracy**

To find the best neural network model for a solving a particular problem is always very difficult task. There is no theoretical or empirical formula to
model a network as an optimum one. This task must be performed by trial and observation technique.

7.8 CONCLUSIONS

In this dissertation work, two neural networks with 36 input nodes in the input layer corresponding to the input features have been developed for recognizing 10 numerical signs and 26 alphabetical signs. The numerical signs network has been trained by a training set which consists of the values after extraction of features from 200 sign images. Similarly, the alphabetical sign recognition network has been trained with a training set obtained from 520 sign images. After training the networks have been tested by varying three different parameters, number of hidden nodes, momentum and learning rate. The test data included 360 different sign images.

The best result for numerical sign recognition has been obtained with the network having learning rate of 0.4, momentum of 0.5 and 21 hidden nodes with target error limit of 0.005. The recognition accuracy of this system has been observed to be 98%. And the network for alphabet sign recognition comprises of 28 hidden neurons with momentum of 0.4 and learning rate of 0.5. The recognition accuracy obtained using this system is 97.31%. These recognition efficiencies are comparable with the published results.

The artificial neural networks created using MATLAB have been able to abstract from the image presented the relevant data for achieving the objective, i.e., they identify hand signs they represent. The system is an off-line system, which requires a hand sign’s image data to be first captured and then recognized. These hand signs can be further associated with commands or control actions to be implemented in a system.