Innovative Systems Design and Engineering
ISSN 2222-1727 (Paper) ISSN 2222-2871 (Online)
Vol.9, No.1, 2018

Design of Neural Network System to Communicate a Blind Person with a Computer Using a Braille

Tekle Haylekiros Assefa
School of Electrical and Computer Engineering, Addis Ababa University, Ethiopia

Tesfay Hailekiros Assefa
Department of Water Resources and Irrigation Engineering, Weldia University, Ethiopia

Abstract
Artificial intelligence systems are widely used nowadays in solving different problems facing mankind. Artificial intelligence systems were developed based on how human brains can think and function; this distinguished character gives these systems the wide range of applications as compared to conventional systems. Among the artificial intelligent systems involves artificial neural intelligent, fuzzy systems and neural fuzzy systems.
Blind person interface to computer is among the problem which the world is facing. In this paper the Artificial Neural Network (ANN) is used in designing the Braille system which is capable of enabling the interface between a blind person and the computer. The Multilayer perceptron (MLP) with four layers and nineteen neural is used for the implementation of pattern recognition of the Braille. The pattern of the Braille is used as the inputs to the MLP whereby through MATLAB developed program the patterns training are achieved. The developed system is capable of enabling the blind person to typing letters, numbers and to enter different commands to the computer.

1. Introduction

1.1. Background

Braille is a system of writing that uses patterns of raised dots to inscribe characters on paper. It therefore allows visually-impaired people to read and write using touch instead of vision. It is a way for blind people to participate in a literate culture. First developed in the nineteenth century, Braille has become the pre-eminent tactile alphabet. Its characters are six-dot cells, two wide by three tall. Any of the dots may be raised, giving 26 or 64 possible characters. Although Braille cells are used world-wide, the meaning of each of the 64 cells depends on the language that they are being used to depict (King, 2001).

Different languages have their own Braille codes, mapping the alphabets, numbers and punctuation symbols to Braille cells according to need. Braille characters can also be used to represent whole words or groups of letters. These contractions allow fewer Braille cells to encode more text, saving the expensive printing costs of Braille text and making Braille faster to use for some experienced users (King, 2001).

Artificial Neural Networks (ANNs) are a computer technique designed to simulate the way of human brain do different tasks. Artificial neural networks can do that by a lot of parallel distributed processing unit (node). These unites are mathematical models called (neurons). The neurons have the ability to process and store the information same as the biological neurons do. ANN consists of an interconnected group of artificial neurons and processes information (Zurada, 1996). The most commonly used ANN structure contained many layers, the first layer is the input layer, then one or more hidden layer, and finally the output layer. Each layer consists of at least one or more neurons. These neurons are connected by a connection line, which indicate the flow of information from one node to the next and from the input layer to the output layer through the network (Zurada, 1996).

1.2. Problem statement

Visually impaired people are an integral part of all societies and they can play an effective role in the development as everyone else. They face difficulty in communication and pursuing their goals. In this era of technology, the knowledge resources are at the finger tips, but in the world of blind people, they have only two sources of knowledge ,audio and Braille script. Therefore it is necessary to have a way they can communicate and interact with the computer which is used in many services nowadays.

Neural network-based applications are useful wherever there is a need to bridge the gap in communication between humans and machines. Human-Machine Interface is an area of scientific research where the objective is to make machines respond to human voice, gestures, and thinking. One area that such an application is significant, and can improve the communication ability, is to provide an interface that can translate computer-generated commands to Braille and vice versa. This would be a significant aid to the blind. This paper is concerned with the translation of text to and from Braille code by a number using neural networks. Neural networks can be used to perform pattern classification with good results. Braille character recognition also falls into the same field of pattern recognition. This paper presents a pattern recognition approach based on a neural

network that can recognize Braille characters. The proposed neural network model uses backpropagation training algorithm.

1.3. Objectives

1.3.1. General Objective

The general objective of this paper is to design a neural network to provide an interface that can translate computer-generated commands to Braille and vice versa, using a set of training patterns that adequately represent Commonly used words and commands in Word processing.

1.3.2. Specific objectives

i. To design a neural network that can do Braille characters' pattern recognition
ii. To use Braille characters' pattern for typing alphabets and numbers in computers
iii. To simulate the designed neural network using MATLAB

2. Literature Review

In this literature review, the artificial neural networks in general, its history, description are discussed. Some applications of the Neural Networks which facilities the human being, especially in Braille area are also reviewed.

2.1. Artificial Neural Network

Artificial Neural Network is an information processing system which is inspired by the models of biological neural network. It is an adaptive system that changes its structure or internal information that flows through the network during the training time. In terms of definition Artificial Neural Network is Computer simulation of a "brain like" system of interconnected processing units (Foram S., 2014).

2.1.1. Architecture

Artificial neural network (ANN) is a machine learning approach that models human brain and consists of a number of artificial neurons. Neuron in ANNs tend to have fewer connections than biological neurons. Each neuron in ANN receives a number of inputs. An activation function is applied to these inputs which results output value of the neuron. Knowledge about the learning task is given in the form of examples called training examples. Architecture of the Artificial Neural Network consists Input layer, Hidden layer, Output layer.

2.1.2. Input Layer

The Input Layer is a layer which communicates with the external environment. Input layer presents a pattern to neural network. Once a pattern is presented to the input layer, the output layer will produce another pattern. It also represents the condition for which purpose this paper is training the neural network (Karsoliya, 2012).

2.1.3. Output Layer

The Output layer of the neural network is what actually presents a pattern to the external environment. The number of output neurons should be directly related to the type of the work that the neural network is to perform (Karsoliya, 2012).

2.1.4. Hidden Layer

The Hidden layer of the neural network is the intermediate layer between Input and Output layer. Activation function applies on hidden layer if it is available. Hidden layer consists hidden nodes. Hidden nodes or hidden neurons are the neurons that are neither in the input layer nor the output layer (Foram S., 2014).

2.2. Types of neural network

There are several types of neural networks but in this report two types are reviewed: Multi-Layer Perceptron (MLP), Radial Basis Function (RBF)

2.2.1. Multi Layer Perceptron

2.2.1.1. General Architecture

Multi-layer perceptron represent a generalization of the single-layer perceptron as described in the previous section. A single layer perceptron forms a half–plane decision region. On the other hand multi-layer perceptron can form arbitrarily complex decision regions and can separate various input patterns. The capability of multi-layer perceptron stems from the non-linearity used within the nodes. If the nodes were linear elements, then a single-layer network with appropriate weight could be used instead of two- or three-layer perceptron. Figure 2 shows a typical multi-layer perceptron neural network structure. As observed it consists of the following layers:

(i) Input Layer: A layer of neurons that receives information from external sources, and passes this information to the network for processing. These may be either sensory inputs or signals from other systems outside the one being modeled.
(ii) Hidden Layer: A layer of neurons that receives information from the input layer and processes them in a hidden way. It has no direct connections to the outside world (inputs or outputs). All connections from the hidden layer are to other layers within the system.

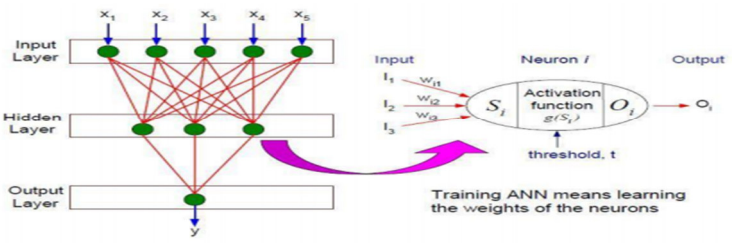

Figure 1: Architecture of ANN

(iii) Output Layer: A layer of neurons that receives processed information and sends output signals out of the system.

(iv) Bias: Acts on a neuron like an offset. The function of the bias is to provide a threshold for the activation of neurons. The bias input is connected to each of the hidden and output neurons in a network. The number of input neurons corresponds to the number of input variables in the neural network, and the number of output neurons is the same as the number of desired output variables. The number of neurons in the hidden layer(s) depends upon the particular NN application.

2.2.2. Radial basis function network

Radial basis function (RBF) networks are feed-forward networks trained using a supervised training algorithm. They are typically configured with a single hidden layer of units whose activation function is selected from a class of functions called basis functions. While similar to back propagation in many respects, radial basis function networks have several advantages. They usually train much faster than back propagation networks (Foram S., 2014).

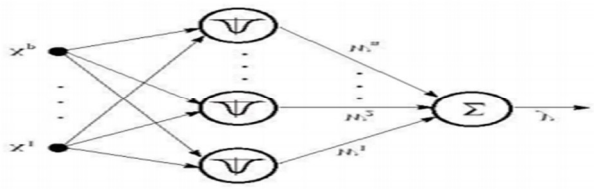

Figure 2: Radial basis function neural network

2.3. Learning Methods of ANN

The learning methods in neural networks are classified into two basic types (Waleed, 2017): these are Supervised Learning and Unsupervised Learning.

In Supervised learning a teacher is present during learning process. In this process expected output is already presented to the network. Also every point is used to train the network.

In Unsupervised learning a teacher is absent during the learning process. The desired or expected output is not presented to the network. The system learns of its own by discovering and adapting to the structural features in the input pattern. Among all these methods Supervised and Unsupervised methods are most popular methods for training the network.

2.4. Activation Function

Activation function, f is performing a mathematical operation on the signal output. Activation functions are chosen depending upon the type of problem to be solved by the network. There are some common activation functions such as linear activation function, Piecewise Linear activation function, Tangent hyperbolic function, sigmoidal function, threshold function.

Linear activation function will produce positive number over the range of all real number. Threshold function is either binary type or bi-polar type. If the threshold function is binary type its range is in between 0 to 1 and if the threshold function is bi -polar type its range lies between -1 to 1.

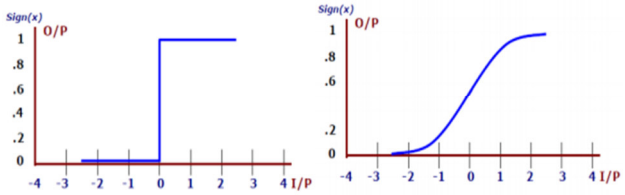

Figure 3: Threshold function

Piecewise linear function is also named as saturating linear function. This function has binary and bipolar range for saturating limits of the output. Its range lies between -1 to 1.

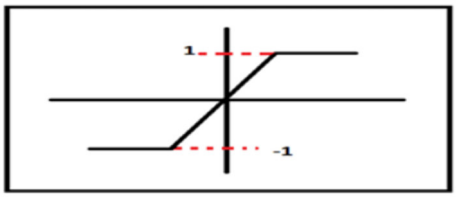

Figure 4: Piecewise linear function

Sigmoidal function is non-linear curved S-shaped function. This function is the most common type of activation function used to construct the neural network. Sigmoidal function is strictly increasing function by nature.

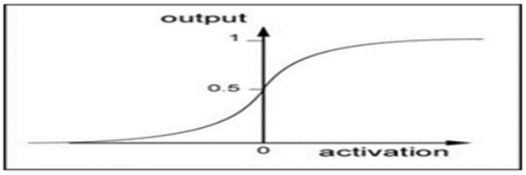

Figure 5: Sigmoidal function

2.5. Application of Artificial Neural Network

There are various types of applications of Artificial Neural Network:

- System identification and control;
- Game-playing and decision making (backgammon, chess, poker);
- Pattern recognition (radar systems, face identification, object recognition and more)
- Sequence recognition (gesture, speech, handwritten text recognition)
- Medical diagnosis, Financial applications (e.g. automated trading systems)
- Data mining (or knowledge discovery in databases, "KDD"), Visualization and e-mail spam filtering.

2.6. Braille

Reading is one way to get information, but for those who are blind it will be difficult if to read the regular letters. Therefore, letters for the blind people was specially designed, named braille letters. Braille letters is consist of six points, which is three lines with two points. Six points can be arranged in such a way to create a variety of combinations. Usually, braille letters is read by touching the dot on the braille paper using fingers (Joko S, 2015).

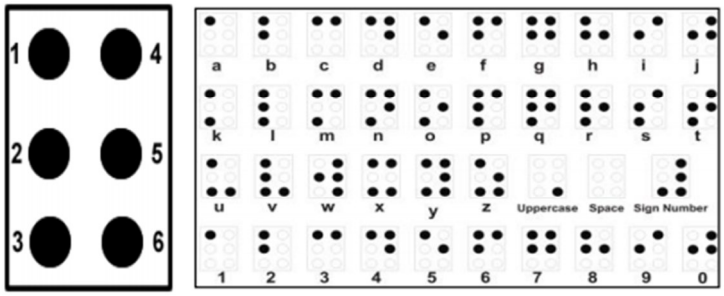

Figure 6: Dot braille character structure

The Braille system is a widely used method by the blind to read and write. Information technology revolution is changing the way Braille reading and writing, making it easier to use. All kinds of materials can be put into Braille representation, such as bank statements, bus ticket, maps, and music note.

2.7. Application of artificial neural Network in Braille

Joko Subur1, Tri Arief Sardjono1, and Ronny Mardiyanto1 (2015), came up with a Braille letter that is characters designed for the blind consisting of six embossed points, arranged in a standard braille character. Braille letters is touched and read using fingers, therefore the sensitivity of the fingers is important. Those characters need to be memorized, so it is very difficult to be learned. The aim of their research was to create a braille characters recognition system and translate it to alphanumeric text. Webcam camera was used to capture braille image from braille characters on the paper sheet. Cropping, grayscale, thresholding, erotion, and dilation techniques are used for image preprocessing. Then, artificial neural network method was used to recognize the braille characters. Their system recognized braille characters with 99% accuracy even when the braille image was tilted up to 1 degrees but above 1 degree the accuracy begun to reduce (Joko S, 2015). Mohammed Waleed came up with an artificial neural networks designed to identify the number's image from (0-9) in Braille representation system. Networks were trained and tested to be used for identify the scanned English number in

Braille representation system. Some of the numbers were noised with some type of noise to simulate somehow the real world environment. According to their result of the identification of number written in Braille representation using Artificial Neural Networks, the training accuracy was 97.1% and testing accuracy was 85%. He proposed that as a future work, it might be to expand the database to contain a letter and even to work with whole word and sentences that written in Braille representation (Waleed, 2017).

3. Methodology

In this paper MATLAB is used to design the artificial neural network to translate file management operations into braille so that a blind person can use the word processor effectively. The ANN used in the paper is the Multi- layer perceptron with one input layer, two hidden layers and one output layer as shown in figure 7.

3.1. Inputs from the Braille

The MLP has six inputs and one output which are shown the table below:

Table 1: Input and out patterns with their target value for Litters

X_1	X_2	X_3	X_4	X_5	X_6	OUTPUT	
						Letter Out put	Target Value
1	0	0	0	0	0	A	2
1	1	0	0	0	0	B	3
1	0	0	1	0	0	C	4
1	0	0	1	1	0	D	5
1	0	0	0	1	0	E	6
1	1	0	1	0	0	F	7
1	1	0	1	1	0	G	8
1	1	0	1	0	0	H	9
0	1	0	1	0	0	I	10
0	1	0	1	1	0	J	11
1	0	1	0	0	0	K	12
1	1	1	0	0	0	L	13
1	0	1	1	0	0	M	14
1	0	1	1	1	0	N	15
1	0	1	0	1	0	O	16
1	1	1	1	0	0	P	17
1	1	1	1	1	0	Q	18
1	1	1	0	1	0	R	19
0	1	1	1	0	0	S	20
0	1	1	1	1	0	T	21
1	0	1	0	0	1	U	22
1	1	1	0	0	1	V	23
0	1	0	1	1	1	W	24
1	0	1	1	0	1	X	25
1	0	1	1	1	1	Y	26
1	0	1	0	1	1	Z	27

Table 2: Input and out patterns with their target value for Numbers and some special symbols

X1	X2	X3	X4	X5	X6	OUTPUT	
						Number/Symbol Output	Target Value
1	0	0	0	0	0	1	2
1	1	0	0	0	0	2	3
1	0	0	1	0	0	3	4
1	0	0	1	1	0	4	5
1	0	0	0	1	0	5	6
1	1	0	1	0	0	6	7
1	1	0	1	1	0	7	8
1	1	0	1	0	0	8	9
0	1	0	1	0	0	9	10
0	1	0	1	1	0	0	11
0	0	1	1	1	1	#	39
0	0	0	0	0	0	SPACE	40
0	0	0	0	0	1	UPPERCASE	41
1	0	0	0	0	1	YES	42
0	0	0	1	0	1	NO	43
1	0	0	1	0	1	FILE	44
1	1	0	0	1	1	SAVE	45
1	1	1	0	1	1	EDIT	46

3.2. Neural Network Training

To train the MLP, I used the back propagation algorithm. The training phase of back propagation algorithm can be summarized in the following steps:

i. The weights of the network were initialized by setting all the weights to a value of 0.2.
ii. Scale the input/output data.
iii. The structure of the network was selected – the network has one input layer made up of six neurons. Two hidden layers made up of six neurons each. One output layer made up of one neuron(see figure 7 below)
iv. The activation function for the neurons was chosen to be the sigmoid function.
v. The training pair from the training set was selected and the input vector was applied to the network input.
vi. The output of the network was calculated based on the initial weights and input set.
vii. The error between network output and the desired output was calculated. (the target vector from the training pair)
viii. Error was propagated backward and weights adjusted in such a way that minimizes the error. Starting from the output layer and going backward to input layer.
ix. Steps 5−8 were repeated for each vector in the training set until the error for the set is lower than the required minimum error.

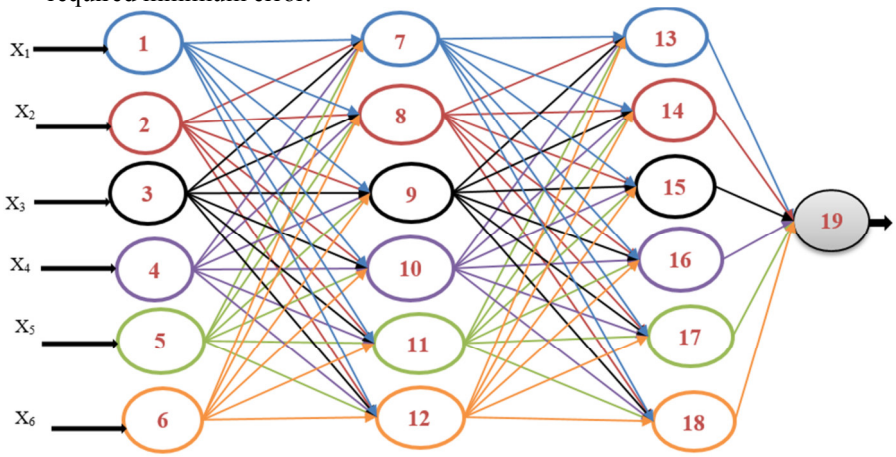

Figure 7: The neural network used for the Braille training with six inputs, two hidden layers and single output

4. Results and Discussion

4.1. Introduction

The designing of the system was done through implementation of multilayer perceptron with four layers which

are input, out and two hidden layers. The patterns recognition program for multilayer perceptron for Braille application was developed by using MATLAB software, where training of MLP by using the programmed code is illustrated under this chapter.

4.2. Program Code

The program for Braille implementation was developed which is able to train any Braille pattern as they were mentioned in chapter 3. Consider the developed program below;

```
 clc; clear all;
 disp('*DESIGN OF NEURAL NETWORK TO COMMUNICATE A BRAIL WITH A COMPUTER*')
disp('*Here are the Details of the Neural Network Training for Brail in MATLAB*')
 t=1;
 while (t ~=0)
 input1= input('Enter the signal inputs to the Brail, row vector with [6] Elements OR Press [CTRL+C] to terminate:');
%input1=[1 0 0 0 0 0 ];
 i=input1;
 ShiftNum=2;
 if     (i(1)== 0 && i(2)== 0 && i(3)== 1 && i(4)== 1 && i(5)== 1 && i(6)== 1)
 disp('*This is for "Numeric Key(#)" and hence insert the other input again for Training*')
 input1= input('Enter the signal inputs to the Brail, row_vector with [6] Elemets OR Press [CTRL+C] to terminate:');
 ShiftNum=3; end
 if     (i(1)== 0 && i(2)== 1 && i(3)== 0 && i(4)== 1 && i(5)== 0 && i(6)== 1)
 disp('*This is for LETTERS Key and hence insert the other input again for Training*')
 input1= input('Enter the signal inputs to the Brail, row_vector with [6] Elemets OR Press [CTRL+C] to terminate:'); end
 i=input1;
if     (i(1)== 1 && i(2)== 0 && i(3)== 0 && i(4)== 0 && i(5)== 0 && i(6)== 0) Target_Value= 2;
else if (i(1)== 1 && i(2)== 1 && i(3)== 0 && i(4)== 0 && i(5)== 0 && i(6)== 0) Target_Value= 3;
else if (i(1)== 1 && i(2)== 0 && i(3)== 0 && i(4)== 1 && i(5)== 0 && i(6)== 0) Target_Value= 4;
else if (i(1)== 1 && i(2)== 0 && i(3)== 0 && i(4)== 1 && i(5)== 1 && i(6)== 0) Target_Value= 5;
else if (i(1)== 1 && i(2)== 0 && i(3)== 0 && i(4)== 0 && i(5)== 1 && i(6)== 0) Target_Value= 6;
else if (i(1)== 1 && i(2)== 1 && i(3)== 0 && i(4)== 1 && i(5)== 0 && i(6)== 0) Target_Value= 7;
else if (i(1)== 1 && i(2)== 1 && i(3)== 0 && i(4)== 1 && i(5)== 1 && i(6)== 0) Target_Value= 8;
else if (i(1)== 1 && i(2)== 1 && i(3)== 0 && i(4)== 0 && i(5)== 1 && i(6)== 0) Target_Value= 9;
else if (i(1)== 0 && i(2)== 1 && i(3)== 0 && i(4)== 1 && i(5)== 0 && i(6)== 0) Target_Value= 10;
else if (i(1)== 0 && i(2)== 1 && i(3)== 0 && i(4)== 1 && i(5)== 1 && i(6)== 0) Target_Value= 11;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 0 && i(5)== 0 && i(6)== 0) Target_Value= 12;
else if (i(1)== 1 && i(2)== 1 && i(3)== 1 && i(4)== 0 && i(5)== 0 && i(6)== 0) Target_Value= 13;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 1 && i(5)== 0 && i(6)== 0) Target_Value= 14;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 1 && i(5)== 1 && i(6)== 0) Target_Value= 15;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 0 && i(5)== 1 && i(6)== 0) Target_Value= 16;
else if (i(1)== 1 && i(2)== 1 && i(3)== 1 && i(4)== 1 && i(5)== 0 && i(6)== 0) Target_Value= 17;
else if (i(1)== 1 && i(2)== 1 && i(3)== 1 && i(4)== 1 && i(5)== 1 && i(6)== 0) Target_Value= 18;
else if (i(1)== 1 && i(2)== 1 && i(3)== 1 && i(4)== 0 && i(5)== 1 && i(6)== 0) Target_Value= 19;
else if (i(1)== 0 && i(2)== 1 && i(3)== 1 && i(4)== 1 && i(5)== 0 && i(6)== 0) Target_Value= 20;
else if (i(1)== 0 && i(2)== 1 && i(3)== 1 && i(4)== 1 && i(5)== 1 && i(6)== 0) Target_Value= 21;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 0 && i(5)== 0 && i(6)== 1) Target_Value= 22;
else if (i(1)== 1 && i(2)== 1 && i(3)== 1 && i(4)== 0 && i(5)== 0 && i(6)== 1) Target_Value= 23;
else if (i(1)== 0 && i(2)== 1 && i(3)== 0 && i(4)== 1 && i(5)== 1 && i(6)== 1) Target_Value= 24;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 1 && i(5)== 0 && i(6)== 1) Target_Value= 25;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 1 && i(5)== 1 && i(6)== 1) Target_Value= 26;
else if (i(1)== 1 && i(2)== 0 && i(3)== 1 && i(4)== 0 && i(5)== 1 && i(6)== 1) Target_Value= 27;
            % FOR SYMBOLS1
else if (i(1)== 0 && i(2)== 0 && i(3)== 0 && i(4)== 0 && i(5)== 0 && i(6)== 0) Target_Value= 39;
else if (i(1)== 0 && i(2)== 0 && i(3)== 0 && i(4)== 0 && i(5)== 0 && i(6)== 1) Target_Value= 40;
else if (i(1)== 1 && i(2)== 0 && i(3)== 0 && i(4)== 0 && i(5)== 0 && i(6)== 1) Target_Value= 41;
else if (i(1)== 0 && i(2)== 0 && i(3)== 0 && i(4)== 1 && i(5)== 0 && i(6)== 1) Target_Value= 42;
else if (i(1)== 1 && i(2)== 0 && i(3)== 0 && i(4)== 1 && i(5)== 0 && i(6)== 1) Target_Value= 43;
```



```
else if (i(1)== 1 && i(2)== 1 && i(3)== 0 && i(4)== 0 && i(5)== 1 && i(6)== 1) Target_Value= 44;
else if (i(1)== 1 && i(2)== 1 && i(3)== 1 && i(4)== 0 && i(5)== 1 && i(6)== 1) Target_Value= 45;
   else Target_Value= 47;
end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;
end;end;end;end;end;end;end;end;end;end;end;
%THE MAIN PROGRAMMING CODE
%learningRate = input('Enter the learningRate,[L] a constant between 0 and 1->');
Learning_Rate = 0.27;
g = inline('logsig(x)');% Activation Function,(Sigmoid function)
l = inline('purelin(x)');% activation function in the last layer(linear function)
[row, col] = size(input1);
NumberOftheInputNeurons = col;
NumberOfFirstHiddenNeurons = col;
NumberOfSecondHiddenNeurons = 6;
NumberOfTheOutputNeurons = 1;
%Initial Weights for training
Weight_input_Hidden_Layer1 =[ 0.2  0.2  0.2  0.2  0.2  0.2; 0.2  0.2  0.2  0.2  0.2  0.2
                              0.2  0.2  0.2  0.2  0.2  0.2; 0.2  0.2  0.2  0.2  0.2  0.2
                              0.2  0.2  0.2  0.2  0.2  0.2; 0.2  0.2  0.2  0.2  0.2  0.2];
Weight_input_Hidden_Layer2 =[ 0.2  0.2  0.2  0.2  0.2  0.2; 0.2  0.2  0.2  0.2  0.2  0.2
                              0.2  0.2  0.2  0.2  0.2  0.2; 0.2  0.2  0.2  0.2  0.2  0.2
                              0.2  0.2  0.2  0.2  0.2  0.2; 0.2  0.2  0.2  0.2  0.2  0.2];
Weight_output_Hidden_Layer = [0.2  0.2  0.2  0.2  0.2  0.2]';
epochs = 0;
errorMatrix = [];
errorThreshold=0.1;
while(true)
totalError = 0;
epochs = epochs + 1;
%Forward Propagation
for i = 1:row
hidden_output1 = g(input1(1, 1:end)*Weight_input_Hidden_Layer1); %g is the activation function in the
network
hidden_output2 = g(hidden_output1(1, 1:end)*Weight_input_Hidden_Layer2);
Final_output = l(hidden_output2*Weight_output_Hidden_Layer);% l is activation function in out put layer only
error = abs(Target_Value  - Final_output);
error = sum(error);
totalError = totalError + error;
if(error ~= 0)
 %look on "delta_hidden_output2"
%delta_final_output = final_output.*(1-final_output).*(target - final_output);
delta_final_output3 = (Target_Value - Final_output);% Linear Activation Function
delta_hidden_output2                                                    =(hidden_output2).*(1-
hidden_output2).*(delta_final_output3*Weight_output_Hidden_Layer');
delta_hidden_output1                                                    =(hidden_output1).*(1-
hidden_output1).*(delta_hidden_output2*Weight_input_Hidden_Layer2');
%For the 3rd Layer, Backward Propagation
for m = 1:NumberOfSecondHiddenNeurons
for n = 1:NumberOfTheOutputNeurons
   % finding the changes to updating the weights
current_changes3 = Learning_Rate*delta_final_output3(1, n) * hidden_output2(1, m);
Weight_output_Hidden_Layer(m, n) = Weight_output_Hidden_Layer(m, n) + current_changes3; end;end
%For the 2nd Layer, %look on "weight_input_hidden1"
for m = 1:NumberOfFirstHiddenNeurons
for n = 1:NumberOfSecondHiddenNeurons
current_changes2 = Learning_Rate*delta_hidden_output2(1, n) * Weight_input_Hidden_Layer1(1, m);
Weight_input_Hidden_Layer2(m, n) = Weight_input_Hidden_Layer2(m, n) + current_changes2;
end;end
%For the 1st Layer
```



```
for m = 1:NumberOftheInputNeurons;
for n = 1:NumberOfFirstHiddenNeurons
current_changes1 = Learning_Rate*delta_hidden_output1(1, n) * input1(1, m);
Weight_input_Hidden_Layer1(m, n) = Weight_input_Hidden_Layer1(m, n) + current_changes1;
end;end;end;end
totalError = totalError / (row);
errorMatrix(end + 1) = totalError;
if(totalError<=0.001)
break
end ;end
% Values for displaying
input_from_the_Brail=input1
Target_Value
Learning_Rate
epochs
Final_output
%Final Weight of each layer
Weight_input_Hidden_Layer1
Weight_input_Hidden_Layer2
Weight_output_Hidden_Layer
% How good is the result?
Final_output
totalError
Percentage_Error=abs((totalError/Final_output)*100)
% THE OUTPUT TO THE COMPUTER BASED ON THE BRAIL'S INPUT
                %FOR LETTERS and NUMBERS
if     (totalError>=0.001)   disp('*NB: The input is NOT WELL TRAIND')
else if (ShiftNum==2 && Target_Value == 2)   disp('*NB: The Training output represents for the Letter: A')
else if (ShiftNum==2 && Target_Value == 3)   disp('*NB: The Training output represents for the Letter: B')
else if (ShiftNum==2 && Target_Value == 4)   disp('*NB: The Training output represents for the Letter: C')
else if (ShiftNum==2 && Target_Value == 5)   disp('*NB: The Training output represents for the Letter: D')
else if (ShiftNum==2 && Target_Value == 6)   disp('*NB: The Training output represents for the Letter: E')
else if (ShiftNum==2 && Target_Value == 7)   disp('*NB: The Training output represents for the Letter: F')
else if (ShiftNum==2 && Target_Value == 8)   disp('*NB: The Training output represents for the Letter: G')
else if (ShiftNum==2 && Target_Value == 9)   disp('*NB: The Training output represents for the Letter: H')
else if (ShiftNum==2 && Target_Value == 10)  disp('*NB: The Training output represents for the Letter: I')
else if (ShiftNum==2 && Target_Value == 11)  disp('*NB: The Training output represents for the Letter: J')
else if (ShiftNum==2 && Target_Value == 12)  disp('*NB: The Training output represents for the Letter: K')
else if (ShiftNum==2 && Target_Value == 13)  disp('*NB: The Training output represents for the Letter: L')
else if (ShiftNum==2 && Target_Value == 14)  disp('*NB: The Training output represents for the Letter: M')
else if (ShiftNum==2 && Target_Value == 15)  disp('*NB: The Training output represents for the Letter: N')
else if (ShiftNum==2 && Target_Value == 16)  disp('*NB: The Training output represents for the Letter: O')
else if (ShiftNum==2 && Target_Value == 17)  disp('*NB: The Training output represents for the Letter: P')
else if (ShiftNum==2 && Target_Value == 18)  disp('*NB: The Training output represents for the Letter: Q')
else if (ShiftNum==2 && Target_Value == 19)  disp('*NB: The Training output represents for the Letter: R')
else if (ShiftNum==2 && Target_Value == 20)  disp('*NB: The Training output represents for the Letter: S')
else if (ShiftNum==2 && Target_Value == 21)  disp('*NB: The Training output represents for the Letter: T')
else if (ShiftNum==2 && Target_Value == 22)  disp('*NB: The Training output represents for the Letter: U')
else if (ShiftNum==2 && Target_Value == 23)  disp('*NB: The Training output represents for the Letter: V')
else if (ShiftNum==2 && Target_Value == 24)  disp('*NB: The Training output represents for the Letter: W')
else if (ShiftNum==2 && Target_Value == 25)  disp('*NB: The Training output represents for the Letter: X')
else if (ShiftNum==2 && Target_Value == 26)  disp('*NB: The Training output represents for the Letter: Y')
else if (ShiftNum==2 && Target_Value == 27)  disp('*NB: The Training output represents for the Letter: Z')
        %FOR NUMBERS
else if (ShiftNum==3 && Target_Value == 2)  disp('*NB: the training output represents for the NUMBER: 1')
else if (ShiftNum==3 && Target_Value == 3)  disp('*NB: the training output represents for the NUMBER: 2')
else if (ShiftNum==3 && Target_Value == 4)  disp('*NB: the training output represents for the NUMBER: 3')
else if (ShiftNum==3 && Target_Value == 5)  disp('*NB: the training output represents for the NUMBER: 4')
else if (ShiftNum==3 && Target_Value == 6)  disp('*NB: the training output represents for the NUMBER: 5')
```



```
else if (ShiftNum==3 && Target_Value == 7)  disp('*NB: the training output represents for the NUMBER: 6')
else if (ShiftNum==3 && Target_Value == 8)  disp('*NB: the training output represents for the NUMBER: 7')
else if (ShiftNum==3 && Target_Value == 9)  disp('*NB: the training output represents for the NUMBER: 8')
else if (ShiftNum==3 && Target_Value == 10) disp('*NB: the training output represents for the NUMBER: 9')
else if (ShiftNum==3 && Target_Value == 11) disp('*NB: the training output represents for the NUMBER: 0')
        %FOR SPECIAL SYMBOLS
else if (ShiftNum==2 && Target_Value == 39) disp('*NB: the training output represents for the special Button called: SPACE')
else if (ShiftNum==2 && Target_Value == 40) disp('*NB: the training output represents for the special Button called: UPPER')
else if (ShiftNum==2 && Target_Value == 41) disp('*NB: the training output represents for the special Button called: YES')
else if (ShiftNum==2 && Target_Value == 42) disp('*NB: the training output represents for the special Button called: NO')
else if (ShiftNum==2 && Target_Value == 44) disp('*NB: the training output represents for the special Button called: FILE')
else if (ShiftNum==2 && Target_Value == 45) disp('*NB: the training output represents for the special Button called: SAVE')
else if (ShiftNum==2 && Target_Value == 46) disp('*NB: the training output represents for the special Button called: EDIT')
else   disp('*NB: There is no such INPUT KEY in the Brail,..Retry Please!!')
end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;
end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end;end; end; end
    % This is the FULL PROGRAM CODE which EXCUTES correctly!
```

4.3. Program Execution

By using the developed program the blind person can be able to type any letter, number and commands, three illustrations are demonstrate under given scenarios.

4.3.1. Letter Typing

Through typing the correct input code the program is capable of executing the output which is a letter in Braille. Program execution in Matlab command window;

Here are the Details of the Neural Network Training for Brail in MATLAB

Enter the signal inputs to the Brail, row vector with [6] Elements OR Press [CTRL+C] to terminate :->*[1 0 0 0 0 0]*

input_from_the_Brail = 1 0 0 0 0 0

Target_Value = 2

Learning_Rate = 0.2700

epochs = 7

Final output = 1.9995

Weight_input_Hidden_Layer1 =

0.2082	*0.2082*	*0.2082*	*0.2082*	*0.2082*	*0.2082*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*
0.2000	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*
0.2000	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*	*0.2000*

Weight_input_Hidden_Layer2 =

0.2055	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*
0.2055	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*
0.2055	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*	*0.2055*

Weight_output_Hidden_Layer = 0.5021 0.5021 0.5021 0.5021 0.5021 0.5021

Total Error = 5.2779e-04

Percentage_Error = 0.0264

**NB: The Training output represents for the Letter: A*

Through input [1 0 0 0 0 0] which means letter A in Braille, the neural network is trained with the initial weight of 0.2 and target output of 2. The output was programmed in such a way that after the target output is reached the respective letter is displayed, for this case letter “A” is displayed for the given input pattern. Through entering the respective pattern as in the Braille, typing of letters from “A” to “Z” can be achieved in the same manner.

4.3.2. Number Typing

Number typing was achieved through the same codes of letters with the help of shifting letter to number command denote by “#” using “*if then”* command and ten numbers are executed by using codes of letter “A” to “J” . Below is the execution of number 1 by using the same code [1 0 0 0 0 0] which is the code of letter “A”.

Enter the signal inputs to the Brail, row vector with [6] Elements OR Press [CTRL+C] to terminate ->*[0 0 1 1 1 1] .shifting command "#"*
* This is for "Numeric Key(#)" and hence insert the other input again for Training *
Enter the signal inputs to the Brail, row vector with [6] Elements OR Press [CTRL+C] to terminate -> *[1 0 0 0 0 0]*
input_from_the_Brail = 1 0 0 0 0 0
Target_Value = 2
Learning_Rate = 0.2700
epochs = 7
Final_output = 1.9995
Weight_input_Hidden_Layer1 =

0.2082	0.2082	0.2082	0.2082	0.2082	0.2082	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000
0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000
0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000

Weight_input_Hidden_Layer2 =

0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055
0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055
0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055	0.2055

Weight_output_Hidden_Layer = 0.5021 0.5021 0.5021 0.5021 0.5021 0.5021
Total Error = 5.2779e-04; Percentage Error = 0.0264
**NB: the training output represents for the NUMBER: 1*
It is seen that through training the neural network with the same target of 2 and initial weight of 0.2 as for letter "A" , with the help of shifting command "#" number 1 is typed and not A. Through entering different patterns as in Braille, number typing can be achieved from "0" to "9".

4.3.3. Command Typing

Patterns for commands such as space, upper case, yes, no, file, save and edit were developed and these pattern were different from developed patterns of letters and number. These commands were developed so as to enable the blind person to enter the commands to the computer while he/she is working.
Consider the execution of "Yes" command below
Enter the signal inputs to the Brail, row vector with [6] Elements OR Press [CTRL+C] to terminate ->*[1 0 0 0 0 1]*
input_from_the_Brail = 1 0 0 0 0 1
Target_Value = 41, Learning_Rate = 0.2700, epochs = 23
Final_output = 41.0007
Weight_input_Hidden_Layer1 =

1.2806	1.2806	1.2806	1.2806	1.2806	1.2806	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000
0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000
0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	1.2806	1.2806	1.2806	1.2806	1.2806	1.2806

Weight_input_Hidden_Layer2 =

1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643
1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643
1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643	1.0643

Weight_output_Hidden_Layer = 6.8515 6.8515 6.8515 6.8515 6.8515 6.8515
Total Error = 7.3361e-04, Percentage_Error = 0.0018,
**NB: the training output represents for the special called: YES*
Through training the neural network with the target of 41 and initial weight of 0.2 the command "Yes" is implemented. It is seen that the developed neural training program is capable to achieve typing of letters, numbers and commands.

References

Foram S., M. P. (2014). *Review on methods of selecting number of hidden nodes in artificial neural network.* IJCSMC.
Joko S, A. S. (2015). Braille character recognition using ANN. *the first international seminar on Science and Technology.*
Karsoliya, S. (2012). Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture. *International Journal of Engineering Trends and Technology, 3* (6).
King, A. (2001). *Text and Braille Computer Translation.*
Waleed, M. (2017). Braille Identification System Using ANN . *Tikrit Journal of Pure Science,.*
Zurada, M. (1996). *Introduction to Artificial Neural Networks.* Newdelhi.