**Neural Networks And Fuzzy control**

**ECE-407**

**Braille Pattern Recognition**

**By**

**V. V. Ravi Theja Reddy (13BEC0745)**

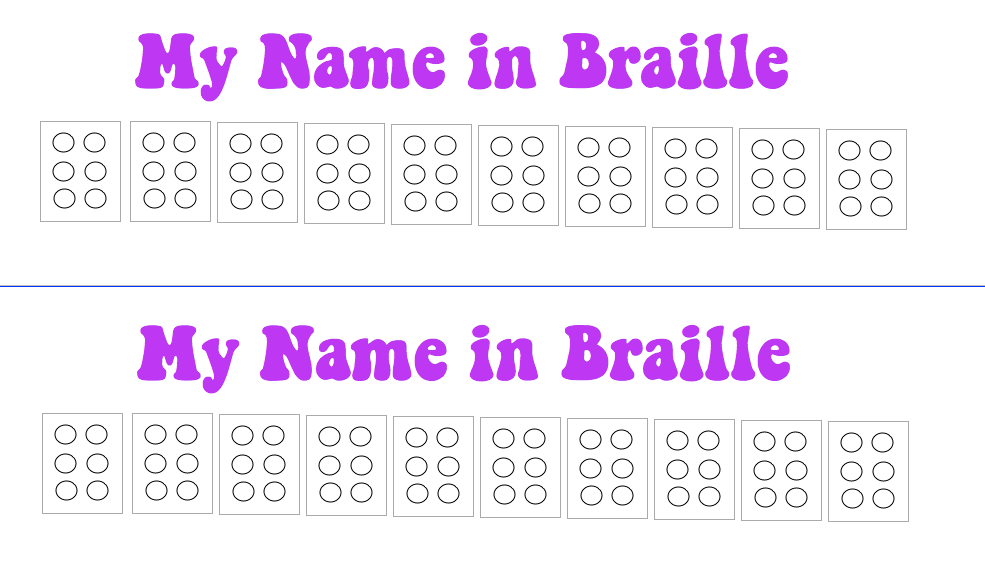
**Abhijeet Viswam (13BEC0731)**

**Slot: E1**

**Submitted to:**

**Prof.Sankar Gansesh S**

**Braille Pattern Recognition**

Braille Alphabet Pattern

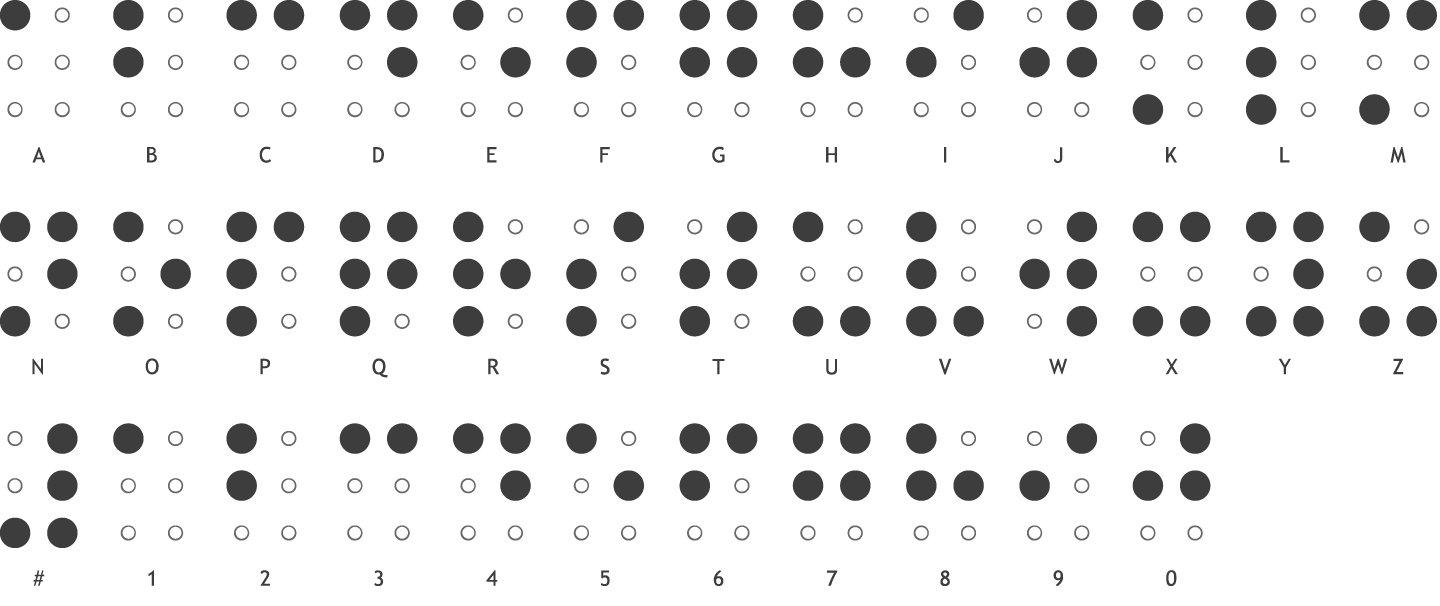
1 4

2 5

3 6

Total 26 = 64 possible input patterns

Braille Alphabets:

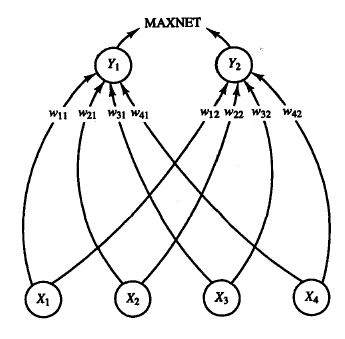


28 possible output patterns – 26 alphabets, #, Blank space

Binary representation – A(100000), B(110000), C(100100), D(100110)…….

**Hamming net** can be used to classify the 64 input patterns and determine which of the 28 exemplar output vectors is most similar to input.

**Architecture: (Hamming Net)**

 6 input neurons – X1 to X6

28 output neurons – Y1 to Y28

Exemplar vectors

e(1)= {1,-1,-1,-1,-1,-1},

e(2)= {1,1,-1,-1,-1,-1},….

e(28)= {-1,-1,-1,-1,-1,-1}

Weight Matrix, W – 28x6

**Algorithm(Hamming Net):**

Assuming that the image recognition software scans the printed braille pattern and saves it as a text file “braille.txt”

1. Initialise necessary variables
2. Initialise 28 exemplar vectors of integer type
3. Calculate 28x6 floating point weight matrix, using
4. Read file “braille.txt”
5. Copy input string e.g. 100101001011000101100101…. from “braille.txt”
6. Start main loop, iterate till end of input string
   1. Copy next 6 characters of input string to temporary string
   2. Convert each character of temporary string to binary integer array

‘1’ =1, ‘0’= -1 E.g. “100101” 🡪 inp\_array= {1, -1, -1, 1, -1, 1}

* 1. Perform Hamming Net Algorithm, followed by MAXNET to find the exemplar most similar to inp\_array.
  2. Map output binary to corresponding character A, B, C… Z, #, blank space and write to the next position in output string.
  3. If character is #, set flag ‘number’ as 1 and delete last written character in output string
  4. If character is blank space, reset flag ‘number’ as 0
  5. If ‘number’ flag equals to 1, and character is A to J, replace with corresponding number in output string

1. Write output string to file “deciphered.txt”.

**Implementation:**

We have implemented the above hamming net algorithm using a C++ code.

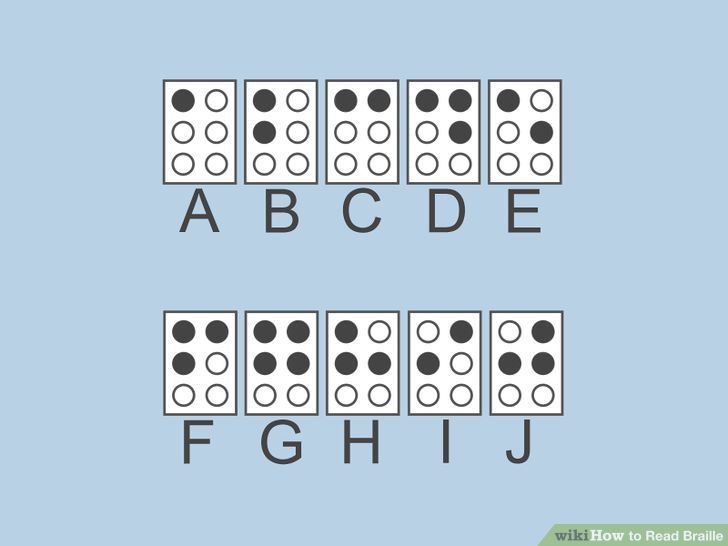
Basically here we have two phases:

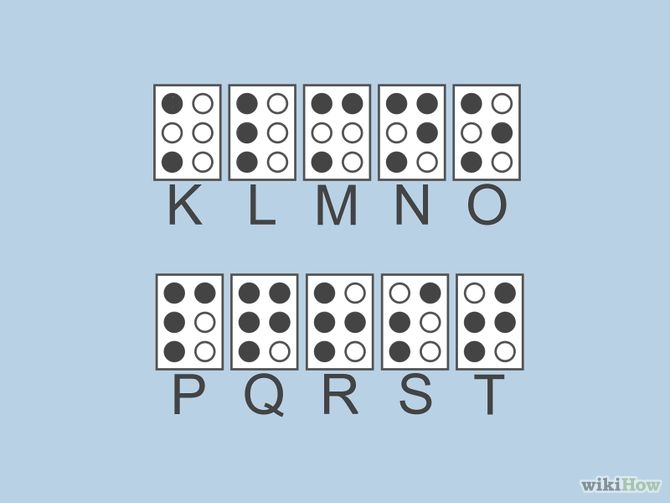
1. **Training Phase:**
   1. Here we take the binary form(digit) value of each braille letter which we already know. And convert them to bipolar forms.
   2. Now we have created a Weight matrix (6x28) using the above algorithm which can be used in testing phase.
2. **Testing Phase:**
   1. Here we have taken a set 50 random braille letters where we don’t know what letter it corresponds to in English. Now we have to find the corresponding English letter using above algorithm.
   2. We convert these letters into binary form and then into bipolar form to perform the algorithm.
   3. Now we have used weight matrix to find the hamming distance of the input for each English letter present.
   4. We also added a bias of value 3 to hamming distance to remove the negative values.
   5. Now the letter which gives maximum hamming distance is the desired letter.
   6. We have to find the desired output for all the 50 inputs using loops.

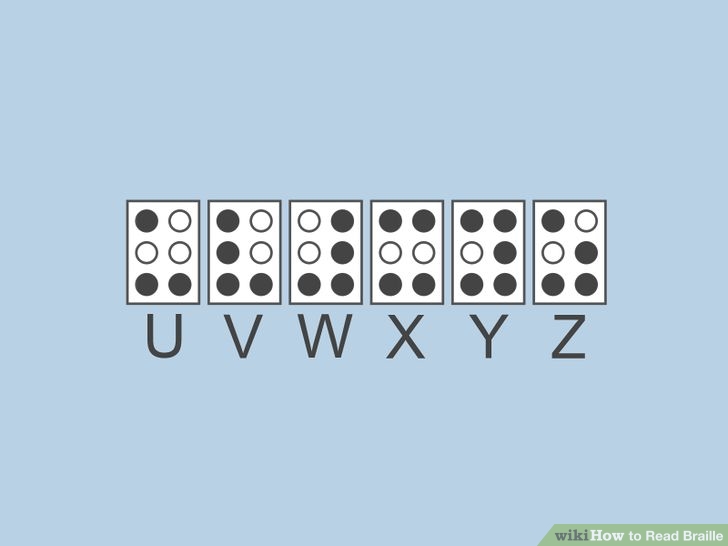
We have attached the input photos from which we have taken input.

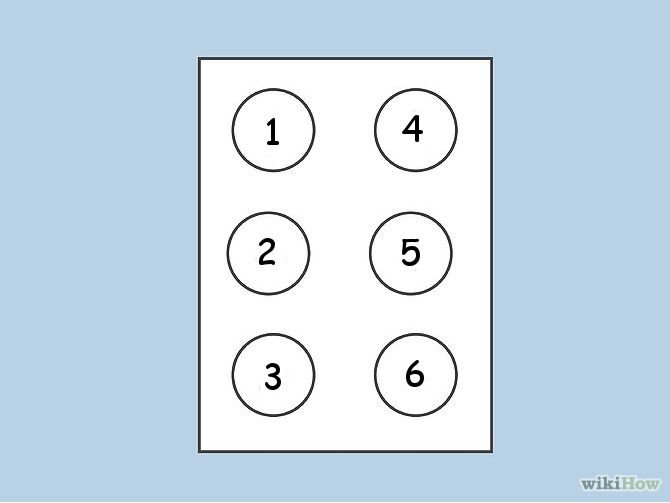
The code which we have used to perform this project.

The output in the form of console window.

The references of input:







**CODE:**

#include <iostream>

#include <string.h>

#include <fstream>

#include<vector>

#include<iomanip>

using namespace std;

float testbinary[50][6];

char res[27]={'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',' '};

void binary(int num,int index,int i)

{

int rem;

if (i == -1)

return;

rem = num % 2;

binary(num / 2,index,i-1);

testbinary[index][i]=rem;

}

int main()

{

float w[6][27];

int i=0,j=0,result;

ifstream fin;

fin.open("TRAINING.txt", ios::in);

char my\_character ;

cout<<"Training data The alphabets in 6 digit binary Form"<<endl;

while (!fin.eof() )

{

fin.get(my\_character);

cout << my\_character;

if(my\_character == '0')

w[i][j] = -0.5;

else

w[i][j] = 0.5;

i++;

if(i==6)

{

j++;

i=0;

cout<<endl;

}

}

cout<<"\r "<<endl;

cout <<"Weight Matrix 6x28:\n";

for (int i=0; i < 6; i++)

{

for(j=0; j < 26; j++)

{

std::cout << std::setprecision(1);

std::cout << std::setw(6);

cout << w[i][j];

}

cout<<endl;

}

cout<<endl;

cout<<"Testing data to find the alphabet\n";

std::fstream myfile("TESTING.txt", std::ios\_base::in);

int test1[50];

int a;

i=0;

while (myfile >> a)

{

cout<<a<<"\t";

test1[i]=a;

i++;

}

cout<<endl;

cout<<"In binary that is in Braille"<<endl;

for(int k=0;k<50;k++)

{

binary(test1[k],k,5);

}

for(int k=0;k<50;k++)

{

for(j=0;j<6;j++)

{

cout<<testbinary[k][j];

}

cout<<"\t";

}

for(int k=0;k<50;k++)

{

for(i=0;i<6;i++)

{

if(testbinary[k][i]==0)

testbinary[k][i]=-1;

}

}

float max1;

cout<<"\n\nInput\t"<<"Hamming distance\t"<<"Output\t"<<endl;

for(int k=0;k<50;k++)

{

float mid[26]={0};

for(i=0;i<26;i++)

{

mid[i]=mid[i]+(float)3; //bias adding

for(j=0;j<6;j++)

{

mid[i]=mid[i]+(float)((float)w[j][i]\*(float)testbinary[k][j]);

}

}

max1=-1;

for(i=0;i<26;i++)

{

if(mid[i]>max1)

{

max1=mid[i];

result=i;

}

}

cout<<test1[k]<<"\t";

cout<<max1<<" \t";

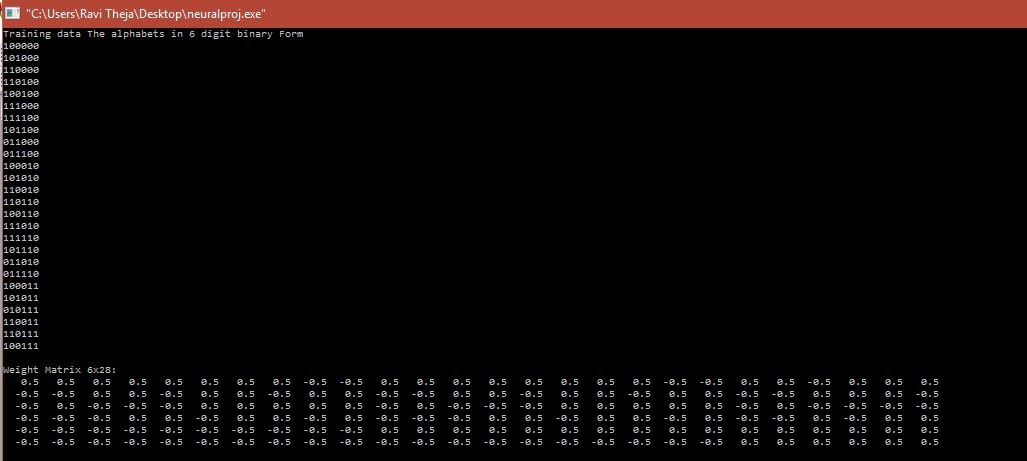
cout<<res[result]<<endl;

}

return 0;

}

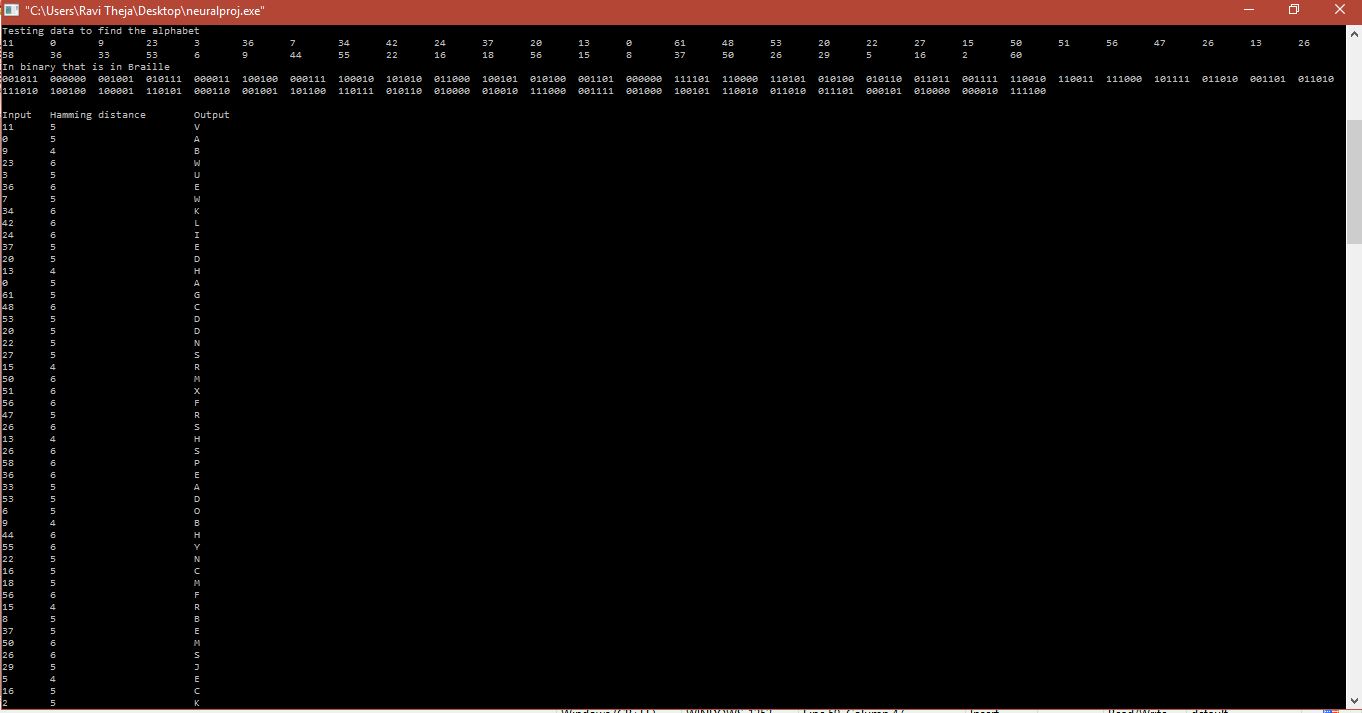
**OUTPUT:**

****

**Training Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Value | Letter | Value | Letter | Value | Letter |
| 100000 | A | 011100 | J | 011010 | S |
| 101000 | B | 100010 | K | 011110 | T |
| 101000 | C | 101010 | L | 100011 | U |
| 110100 | D | 110010 | M | 101011 | V |
| 100100 | E | 110110 | N | 010111 | W |
| 111000 | F | 100110 | O | 110011 | X |
| 111100 | G | 111010 | P | 110111 | Y |
| 101100 | H | 111110 | Q | 100111 | Z |
| 011000 | I | 101110 | R |  |  |

**Testing Output:**

****

**Testing Table:**

|  |  |  |
| --- | --- | --- |
| Input Number | Hamming Distance | Output |
| 11 | **5** | **V** |
| 0 | **5** | **A** |
| 9 | **4** | **B** |
| 23 | **6** | **W** |
| 3 | **5** | **U** |
| 36 | **6** | **E** |
| 7 | **5** | **W** |
| 34 | **6** | **K** |
| 42 | **6** | **L** |
| 24 | **6** | **I** |
| 37 | **5** | **E** |
| 20 | **5** | **D** |
| 13 | **4** | **H** |
| 0 | **5** | **A** |
| 61 | **5** | **G** |
| 48 | **6** | **C** |
| 53 | **5** | **D** |
| 20 | **5** | **D** |
| 22 | **5** | **N** |
| 27 | **5** | **S** |
| 15 | **4** | **R** |
| 50 | **6** | **M** |
| 51 | **6** | **X** |
| 56 | **6** | **F** |
| 47 | **5** | **R** |
| 26 | **6** | **S** |
| 13 | **4** | **H** |
| 26 | **6** | **S** |
| 58 | **6** | **P** |
| 36 | **6** | **E** |
| 33 | **5** | **A** |
| 53 | **5** | **D** |
| 6 | **5** | **O** |
| 9 | **4** | **B** |
| 44 | **6** | **H** |
| 55 | **6** | **Y** |
| 22 | **5** | **N** |
| 16 | **5** | **C** |
| 18 | **5** | **M** |
| 56 | **6** | **F** |
| 15 | **4** | **R** |
| 8 | **5** | **B** |
| 37 | **5** | **E** |
| 50 | **6** | **M** |
| 26 | **6** | **S** |
| 29 | **5** | **J** |
| 5 | **4** | **E** |
| 16 | **5** | **C** |
| 2 | **5** | **K** |
| 60 | **6** | **G** |

**Conclusion:**

This algorithm can be used to find exactly the corresponding letter in English of the Braille Script even though some errors have occurred while Image Processing or some other processes.