Project Report

Name: Glancy Cajitan Rodrigues
N-Number: N15963551
NYU ID: GCR253

Code:

```
import java.io.*;
import java.util.Scanner;
import java.math.*;
class Project {

	public static void main(String args[])throws IOException
	{
		int j,k,l=10;
		int s[][]=new int[10][35];

		//Reading pattern file
		try {
				BufferedReader br = new BufferedReader(new FileReader("test.txt"));
				String line;

				j=0;
				while ((line = br.readLine()) != null)
				{
					for(k=0;k<35;k++)
					{
						if(line.charAt(k)=='#')
							s[j][k]=1;
						else
							s[j][k]=-1;
					}
					j=j+1;
				}
		}
		catch (Exception e)
		{
			System.out.println(e);
		}

		// calculating all the combination and no of pattern recognized in each case by
decreasing no of elements in combination
		int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
		int n = arr.length;
```

```
		for(l=10;l>4;l--)
		{
			Combination(arr, n, l,s);
		}
	}

	static void Combination(int arr[], int n, int r, int s[][])
	{
		int data[]=new int[r];
		combinationUtil(arr, n, r, 0, data, 0,s);
	}

	static void combinationUtil(int arr[], int n, int r, int index, int data[], int i,int s[][])
	{
			if (index == r)
			{
				int a[][] = calc(data,s);//to consider patterns in one particular
combination only
				int w[][] = calculatewt(a, r);
				int x[]=new int[35];
				int y[]=new int[35];
				int count
=0,c_noise=0,c_rev=0,c_bbbn=0,c_bbbr=0,c_spurious=0,c_spr=0;


				for (int j=0; j<r; j++)
				{
					for(int l=0;l<35;l++)
					{
						x[l]=a[j][l];
					}
					y=OutputCalc(x,w);
					if(match(x,y))
					{
						count++;
						//to check display input and output pattern if correctly
identified
						/*
						System.out.println("Input");
						display(x);
						System.out.println("Output");
						display(y);
						*/


					}
```

```
                                        //System.out.println(data[j]+" "); //to display all patterns
checked

                                }
                                //System.out.println("\ncount of correctly identified patterns=
"+count+"\n");

                                /*to check and display where all the patterns are correctly classified
                                  and to check their noise handling capacity*/
                                if(r==count)
                                {
                                        /* to display only the correctly classified patterns*/

                                        for(int k=0;k<r;k++)
                                        System.out.print(data[k]+" ");
                                        System.out.println("\n"+count);


                                        System.out.println("Checking Noise capacity by changing 1 digit
bit by bit\n");

                                        for (int j=0; j<r; j++) //if only one digit has been changed..tested
by changing one digit at a time for 35 times
                                        {
                                                for(int l=0;l<35;l++)
                                                {
                                                        x[l]=a[j][l];
                                                }
                                                c_bbbn=noisecheck_bit_by_bit_noise(w,x);
                                                c_bbbr=noisecheck_bit_by_bit_reverse(w,x);

                                                System.out.println("for "+data[j]+" When digit was
reversed "+c_bbbn+" times pattern got correctly classifed out of 35 times");
                                                System.out.println("for "+data[j]+" When noise got
added "+c_bbbn+" times pattern got correctly classifed out of 35 times");
                                                System.out.println();
                                        }
                                        //checking noise handling capacity of network by increasing no
of changing digits from 1 to 15
                                        System.out.println("\nChecking Noise capacity by changing no
of digits randomly\n");
                                        for(int m=1;m<16;m++)
                                        {
                                                c_noise=0;
                                                c_rev=0;
                                                c_spr=0;
                                                for (int j=0; j<r; j++)
                                                {
```

```
                                                for(int l=0;l<35;l++)
                                                {
                                                        x[l]=a[j][l];
                                                }

                                                if(noisecheck_reverse(w,x,m))
                                                        c_rev=c_rev+1;
                                                if(noisecheck_noise(w,x,m))
                                                        c_noise=c_noise+1;
                                                if(spurious_check_random(w,x,m))
                                                        c_spr=c_spr+1;

                                        }
                                        System.out.println("if "+m+" digits are reversed
"+c_rev+" patterns got classified correctly from ttotal "+r+" patterns");
                                        System.out.println("if "+m+" digits noise added
"+c_noise+" patterns got classified correctly from ttotal "+r+" patterns");
                                        System.out.println("if "+m+" digits noise reversed
"+c_spr+"    spurious patterns got classified correctly");
                                        System.out.println();
                                }


                                System.out.println("Checking digits not present in
combination");
                                if(data.length<10)
                                {
                                        int data1[]=calc1(data);//to check non-considered
pattern to be used to check spurious pattern
                                        int b[][]=calc(data1,s);
                                        //check spurious patterns identified correctly
                                        if(r<10)
                                        {
                                                c_spurious=check_spurious(b,w,data1);
                                                System.out.println(c_spurious+" Spurious
patterns are available for this pattern among digits not available in sample");
                                        }

                                }

                                System.out.println("\n\n\n");


                        }

                        return;
                }
                if (i >= n)
```

```
                return;
            data[index] = arr[i];
            combinationUtil(arr, n, r, index+1, data, i+1,s);
            combinationUtil(arr, n, r, index, data, i+1,s);
}

static int[][] calc(int subscripts[], int s[][])
{
        int a[][]=new int[subscripts.length][35];
        for(int k=0;k < subscripts.length; k++)
        {
                for(int j=0;j<35;j++)
                {
                        a[k][j]=s[subscripts[k]][j];
                }
        }
        return a;
}

static int[] calc1(int data[])
{
        int data1[]=new int[10-data.length];
        int temp[]=new int[10];
        int k=0;
        for(int j=0;j<data.length;j++)
        {
                temp[data[j]]=1;
        }
        for(int j=0;j<10;j++)
        {
                if(temp[j]!=1)
                {
                        data1[k]=j;
                        k++;
                }
        }

        return data1;
}
static int[][] calculatewt(int s[][],int n)
{

        int w[][]=new int[35][35];
        int l,j,k;
        for(l=0;l<35;l++)
                for(j=0;j<35;j++)
                        w[l][j]=0;
        for(l=0;l<n;l++)
```

```
                for(j=0;j<35;j++)
                        for(k=0;k<35;k++)
                                w[j][k]=w[j][k]+s[l][j]*s[l][k];
        //diagonal elements zero
        /*for(l=0;l<35;l++)
                for(k=0;k<35;k++)
                        if(l==k)
                                w[l][k]=0;
    */
        return w;
}

static int[] OutputCalc(int[] x, int[][] w)
{
        int yin[]=new int[35];
        int y[]=new int[35];
        int i,j;
        for(i=0;i<35;i++)
                for(j=0;j<35;j++)
                        yin[i]=yin[i]+w[j][i]*x[j];
        for(i=0;i<35;i++)
        {
                        if(yin[i]>0)
                                y[i]=1;
                        else
                                y[i]=-1;
        }
        return y;
}

static void display(int[] x)
{
        int i;
        for(i=0;i<35;i++)
        {
                if(x[i]==1)
                        System.out.print("#");
                if(x[i]==(-1))
                        System.out.print(" ");
                if(i%5==4)
                        System.out.println();
        }
}

static boolean match(int[] x,int[] y)
{
        int i;
        for(i=0;i<35;i++)
```

```
                if(x[i]!=y[i])
                        return false;
        return true;

}
static int noisecheck_bit_by_bit_noise(int[][] w, int[] x)
{
        int x2[]=new int[35];
        int y[]=new int[35];
        int c2=0;
        for(int k=0;k<35;k++)
                x2[k]=x[k];
        for(int j=0;j<35;j++)
        {
                if(j!=0)
                        x2[j-1]=x[j-1];
                x2[j]=0;
                y=OutputCalc(x2,w);
                if(match(x,y))
                {
                        c2++;
                        /*System.out.println("no change if "+j+"th digit has noisy value" );
                        System.out.println("noisy input");
                        display(x1);
                        System.out.println("output");
                        display(y);
                        */
                }
                /*
                else
                {
                        System.out.println("noisy output if "+j+"th digit has noisy value" );
                        System.out.println("noisy input");
                        display(x1);
                        System.out.println(" noisy output");
                        display(y);
                }
                */

        }
        return c2;
}
static int noisecheck_bit_by_bit_reverse(int[][] w, int[] x)
{
        int x1[]=new int[35];

        int y[]=new int[35];
        int c1=0,c2=0;
```

```
		for(int k=0;k<35;k++)
			x1[k]=x[k];

			for(int j=0;j<35;j++)
			{
				if(j!=0)
					x1[j-1]=x[j-1];
				if(x1[j]==1)
					x1[j]=-1;
				else
					x1[j]=1;
				y=OutputCalc(x1,w);
				if(match(x,y))
				{
					c1++;
					/*
					System.out.println("no change if "+j+"th digit is reversed" );
					System.out.println("noisy input");
					display(x1);
					System.out.println("output");
					display(y);
					*/
				}
				/*else
				{
					System.out.println("noisy output if "+j+"th digit is reversed" );
					System.out.println("noisy input");
					display(x1);
					System.out.println(" noisy output");
					display(y);
				}*/

			}

			return c2;
		}

static boolean noisecheck_noise(int[][] w,int[] x,int p)
{
		int digit;
		int min=0;
		int max=34;
		int x2[]=new int[35];
		int y[]=new int[35];
		for(int k=0;k<35;k++)
			x2[k]=x[k];
		for(int j=0;j<p;j++)
		{
```

```
                digit=    (int)(Math.random() * (max - min) + min);
                x2[digit]=0;
        }
        if(match(x,x2))
        {
                return false;
        }
        else
        {
                y=OutputCalc(x2,w);
                if(match(x,y))
                {
                        /*to display*/
                        /*
                        System.out.println("correctly identified");
                        System.out.println("noisy input");
                        display(x2);
                        System.out.println("output");
                        display(y);*/
                        return true;
                }
                else
                {
                        /*to display*/
                        /*
                        System.out.println("incorrectly identified");
                        System.out.println("noisy input");
                        display(x2);
                        System.out.println(" noisy output");
                        display(y);*/
                        return false;
                }
        }
}
static boolean noisecheck_reverse(int[][] w,int[] x,int p)
{
        int digit;
        int min=0;
        int max=34;
        int x1[]=new int[35];
        int y[]=new int[35];
        for(int k=0;k<35;k++)
                x1[k]=x[k];
        for(int j=0;j<p;j++)
        {
                digit=    (int)(Math.random() * (max - min) + min);
                if(x1[digit]==1)
                        x1[digit]=-1;
```

```
                else
                        x1[digit]=1;
        }
        if(match(x,x1))
        {
                return false;
        }
        else
        {
                y=OutputCalc(x1,w);
                if(match(x,y))
                {
                        /*to display*/
                        /*
                        System.out.println("correctly identified");
                        System.out.println("noisy input");
                        display(x1);
                        System.out.println("output");
                        display(y);
                          */
                        return true;
                }
                else
                {
                        /*
                        System.out.println("incorrectly identified");
                        System.out.println("noisy input");
                        display(x1);
                        System.out.println(" noisy output");
                        display(y);
                          */
                        return false;
                }
        }

}

static int check_spurious(int b[][],int w[][], int[] data1)
{
        int x[]=new int[35];
        int y[]=new int[35];
        int count=0;
        for (int j=0; j<data1.length; j++)
        {
                for(int l=0;l<35;l++)
                {
                        x[l]=b[j][l];
                }
```

```
                y=OutputCalc(x,w);
                if(match(x,y))
                {
                        count++;
                        /* display*/
                        /*System.out.println(data1[j]+" is a spurious pattern");
                        System.out.println("correctly identified");
                        System.out.println("input");
                        display(x);
                        System.out.println(" output");
                        display(y);
                */
                }
        }
        return count;
}
static boolean spurious_check_random(int[][] w,int[] x,int p)
{
        int digit;
        int min=0;
        int max=34;
        int x1[]=new int[35];
        int y[]=new int[35];
        for(int k=0;k<35;k++)
                x1[k]=x[k];
        for(int j=0;j<p;j++)
        {
                digit=  (int)(Math.random() * (max - min) + min);
                if(x1[digit]==1)
                        x1[digit]=-1;
                else
                        x1[digit]=1;
        }
        if(match(x,x1))
        {
                return false;
        }
        else
        {
                y=OutputCalc(x1,w);
                if(match(x1,y))
                {
                        /*to display*/

                        System.out.println("correctly identified spurious pattern");
                        System.out.println("sample pattern already present");
                        display(x);
```

```
                        System.out.println("Spurious input pattern");
                        display(x1);
                        System.out.println("output");
                        display(y);

                        return true;
                }
                else
                {
                        return false;
                }
        }
    }
}
```

Results:

Part 1 Pattern Recognition

1)What is the maximum number of patterns that can stored and recalled successfully----

- For this I started with combination of 10 digits as input and checked whether all the patterns are correctly classified or not. I got only digit 1 classified correctly. Thus I decreased no of digits by 1 . So now I have 9 digits so checked all the combination of 9 digits no. I went on decreasing 1 by 1 digit at each time and checked all the combinations of that. When I took no of digits as 5 , I got 19 combinations for which all the patterns were classified successfully.

  Output of all the patterns and no of correctly classified patterns

```
0 1 2 3 4 5 6 7 8 9
count of correctly identified patterns= 1

0 1 2 3 4 5 6 7 8
count of correctly identified patterns= 2

0 1 2 3 4 5 6 7 9
count of correctly identified patterns= 1

0 1 2 3 4 5 6 8 9
count of correctly identified patterns= 2

0 1 2 3 4 5 7 8 9
count of correctly identified patterns= 1

0 1 2 3 4 6 7 8 9
count of correctly identified patterns= 2

0 1 2 3 5 6 7 8 9
count of correctly identified patterns= 2

0 1 2 4 5 6 7 8 9
count of correctly identified patterns= 1

0 1 3 4 5 6 7 8 9
count of correctly identified patterns= 3

0 2 3 4 5 6 7 8 9
count of correctly identified patterns= 0

1 2 3 4 5 6 7 8 9
count of correctly identified patterns= 1

0 1 2 3 4 5 6 7
count of correctly identified patterns= 1

0 1 2 3 4 5 6 8
count of correctly identified patterns= 1
```

```
0 1 2 3 4 5 6 9
count of correctly identified patterns= 1

0 1 2 3 4 5 7 8
count of correctly identified patterns= 1

0 1 2 3 4 5 7 9
count of correctly identified patterns= 1

0 1 2 3 4 5 8 9
count of correctly identified patterns= 1

0 1 2 3 4 6 7 8
count of correctly identified patterns= 3
```

Output where all the patterns are correctly classified:

```
0 1 2 3 4
count of correctly identified patterns= 5

0 1 2 4 6
count of correctly identified patterns= 5

0 1 2 4 7
count of correctly identified patterns= 5

0 1 3 4 7
count of correctly identified patterns= 5

0 1 4 5 7
count of correctly identified patterns= 5

0 1 4 6 7
count of correctly identified patterns= 5

0 1 4 7 8
count of correctly identified patterns= 5

0 1 4 7 9
count of correctly identified patterns= 5

1 2 3 4 7
count of correctly identified patterns= 5

1 2 4 5 7
count of correctly identified patterns= 5

1 2 4 6 7
count of correctly identified patterns= 5

1 2 4 7 8
count of correctly identified patterns= 5
```

```
1 2 4 7 9
count of correctly identified patterns= 5

1 3 4 7 8
count of correctly identified patterns= 5

1 3 4 7 9
count of correctly identified patterns= 5

1 4 5 7 8
count of correctly identified patterns= 5

1 4 5 7 9
count of correctly identified patterns= 5

1 4 6 7 8
count of correctly identified patterns= 5

1 4 6 7 9
count of correctly identified patterns= 5

0 1 2 4
count of correctly identified patterns= 4

0 1 2 5
count of correctly identified patterns= 4
```

- The same thing I tried by taking weight matrix as diagonal element as zero.
  But the no of combinations that I got previously got reduced to 6. Thus it gave good result without making diagonal element zero.

Output of no of combination where all patterns got correctly classified when diagonal value of weight matrix is made zero.

```
0 1 2 4 7
count of correctly identified patterns= 5

0 1 3 4 7
count of correctly identified patterns= 5

1 2 4 6 7
count of correctly identified patterns= 5

1 2 4 7 8
count of correctly identified patterns= 5

1 4 5 7 8
count of correctly identified patterns= 5

1 4 6 7 9
count of correctly identified patterns= 5

0 1 2 4
```

```
count of correctly identified patterns= 4

0 1 2 5
count of correctly identified patterns= 4
```

2)Does it matter which ones you try to stored

Yes It does matter which digits I store for calculating weight matrix.
I got all patterns classified when I tried combinations of 5.
Since we have 10 digits there are total 252 combinations possible but in results I got only 19 combinations where all patterns got classified successfully.

3) Specify the choice of those patterns? Is there another choice that works nearly as good?

From Output I could say that

```
0 1 2 3 4
0 1 2 4 6
0 1 2 4 7
```

These type of patterns works well.
The reason behind this is in above describe pattern all the patterns have very less similarity.
If I take almost similar patterns then it becomes difficult to train neural network.

For eg 0 ,3, 6, 8, 9 this combination very good similarities and they differ at only 3-4 bits. Thus it becomes difficult to train such type of neural network.

If in above 3rd eg if I change 0 with 3 or 6 or 8 or 9 still it will give me all the patterns correctly classified.

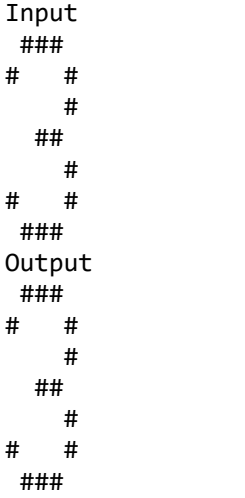
Output for this is as below:

```
0 1 2 4 7
count of correctly identified patterns= 5

1 2 3 4 7
count of correctly identified patterns= 5

1 2 4 6 7
count of correctly identified patterns= 5


1 2 4 7 8
count of correctly identified patterns= 5

1 2 4 7 9
count of correctly identified patterns= 5
```

Display Some correctly identified patterns

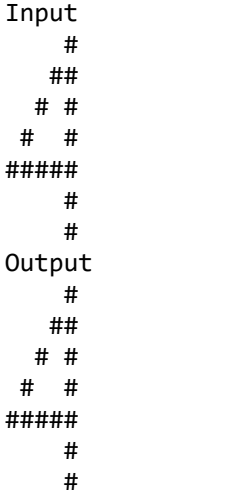
```
Input
 ###
#   #
#   #
#   #
#   #
#   #
 ###
Output
 ###
#   #
#   #
#   #
#   #
#   #
 ###

Input
  #
 ##
# #
  #
  #
  #
#####
Output
  #
 ##
# #
  #
  #
  #
#####
Input
 ###
#   #
   #
  #
  #
 #
#####
Output
 ###
#   #
   #
  #
  #
 #
#####
```

Input



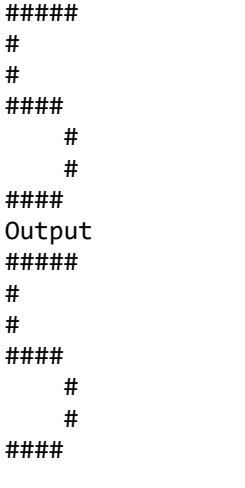
Output

Input



Output

Input



Output

```
Input
 ####
#
#
####
#   #
#   #
 ###
Output
 ####
#
#
####
#   #
#   #
 ###

Input
#####
    #
   #
   #
  #
  #
 #
Output
#####
    #
   #
   #
  #
  #
 #

Input
 ###
#   #
#   #
 ###
#   #
#   #
 ###
Output
 ###
#   #
#   #
 ###
#   #
#   #
 ###
```

```
Input
 ###
#   #
#   #
 ####
    #
    #
####
Output
 ###
#   #
#   #
 ####
    #
    #
####
```

Part 2 Noise

2. How much noise can your net handle? Consider separately missing data values and mistakes in the input patterns.

- For combinations where all the patterns were correctly classified I introduced noise for such patterns by two methods either by making the value zero or by reversing the bit.

- First I checked one pattern and tried changing one by one bit from 0^{th} place to 34^{th} place.

Output when 3% error introduced that is single bit is changed at a time

```
0 1 2 3 4
count of correctly identified patterns= 5

Checking Noise capacity by changing 1 digit bit by bit

for 0 When digit was reversed 33 times pattern got correctly classifed out of 35 times
for 0 When noise got added 33 times pattern got correctly classifed out of 35 times

for 1 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 1 When noise got added 35 times pattern got correctly classifed out of 35 times

for 2 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 2 When noise got added 35 times pattern got correctly classifed out of 35 times

for 3 When digit was reversed 19 times pattern got correctly classifed out of 35 times
for 3 When noise got added 19 times pattern got correctly classifed out of 35 times

for 4 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 4 When noise got added 35 times pattern got correctly classifed out of 35 times

0 1 2 4 6
count of correctly identified patterns= 5
```

```
Checking Noise capacity by changing 1 digit bit by bit

for 0 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 0 When noise got added 35 times pattern got correctly classifed out of 35 times

for 1 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 1 When noise got added 35 times pattern got correctly classifed out of 35 times

for 2 When digit was reversed 27 times pattern got correctly classifed out of 35 times
for 2 When noise got added 27 times pattern got correctly classifed out of 35 times

for 4 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 4 When noise got added 35 times pattern got correctly classifed out of 35 times

for 6 When digit was reversed 28 times pattern got correctly classifed out of 35 times
for 6 When noise got added 28 times pattern got correctly classifed out of 35 times

0 1 2 4 7
count of correctly identified patterns= 5

Checking Noise capacity by changing 1 digit bit by bit

for 0 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 0 When noise got added 35 times pattern got correctly classifed out of 35 times

for 1 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 1 When noise got added 35 times pattern got correctly classifed out of 35 times

for 2 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 2 When noise got added 35 times pattern got correctly classifed out of 35 times

for 4 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 4 When noise got added 35 times pattern got correctly classifed out of 35 times

for 7 When digit was reversed 35 times pattern got correctly classifed out of 35 times
for 7 When noise got added 35 times pattern got correctly classifed out of 35 times
```

Displaying Pattern of erroneous input and its output:

```
no change if 2th digit is reversed
noisy input
 # #
#   #
#   #
 ####
    #
    #
####
output
 ###
#   #
#   #
 ####
    #
    #
####

no change if 20th digit is reversed
noisy input
 ###
#   #
#   #
 ####
#   #
    #
####
output
 ###
#   #
#   #
 ####
    #
    #
####

noisy output if 20th digit is reversed
noisy input
 ###
#   #
#   #
 ####
#   #
    #
####
 noisy output
 ###
#   #
#   #
 ####
    #
    #
####
```

Displaying Pattern of erroneous input and its output when noise is added:

```
no change if 3th digit has noisy value
noisy input
 ##
#   #
#   #
 ####
    #
    #
####
output
 ###
#   #
#   #
 ####
    #
    #
####
no change if 4th digit has noisy value
noisy input
 ###
#   #
#   #
 ####
    #
    #
####
output
 ###
#   #
#   #
 ####
    #
    #
####
```

- I gradually increased the no of bits where error was introduced and got following output. Random digit to be selected each time to introduce error.

Output of no of pattern correctly classified after adding noise

Statistics of times when it correctly identified pattern:

```
0 1 2 3 4
count of correctly identified patterns= 5


Checking Noise capacity by changing no of digits randomly

if 1 digits are reversed 4 patterns got classified correctly from ttotal 5 patterns
if 1 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 2 digits are reversed 4 patterns got classified correctly from ttotal 5 patterns
if 2 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 3 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 3 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 4 digits are reversed 1 patterns got classified correctly from ttotal 5 patterns
if 4 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 5 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 5 digits noise added 3 patterns got classified correctly from ttotal 5 patterns

if 6 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 6 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 7 digits are reversed 1 patterns got classified correctly from ttotal 5 patterns
if 7 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 8 digits are reversed 4 patterns got classified correctly from ttotal 5 patterns
if 8 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 9 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 9 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 10 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 10 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 11 digits are reversed 4 patterns got classified correctly from ttotal 5 patterns
if 11 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 12 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 12 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 13 digits are reversed 1 patterns got classified correctly from ttotal 5 patterns
if 13 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 14 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
```

if 14 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 15 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 15 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

0 1 2 4 6
count of correctly identified patterns= 5

Checking Noise capacity by changing no of digits randomly

if 1 digits are reversed 5 patterns got classified correctly from ttotal 5 patterns
if 1 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 2 digits are reversed 5 patterns got classified correctly from ttotal 5 patterns
if 2 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 3 digits are reversed 3 patterns got classified correctly from ttotal 5 patterns
if 3 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 4 digits are reversed 5 patterns got classified correctly from ttotal 5 patterns
if 4 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 5 digits are reversed 4 patterns got classified correctly from ttotal 5 patterns
if 5 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 6 digits are reversed 4 patterns got classified correctly from ttotal 5 patterns
if 6 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 7 digits are reversed 3 patterns got classified correctly from ttotal 5 patterns
if 7 digits noise added 4 patterns got classified correctly from ttotal 5 patterns

if 8 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 8 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 9 digits are reversed 3 patterns got classified correctly from ttotal 5 patterns
if 9 digits noise added 3 patterns got classified correctly from ttotal 5 patterns

if 10 digits are reversed 4 patterns got classified correctly from ttotal 5 patterns
if 10 digits noise added 3 patterns got classified correctly from ttotal 5 patterns

if 11 digits are reversed 1 patterns got classified correctly from ttotal 5 patterns
if 11 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 12 digits are reversed 1 patterns got classified correctly from ttotal 5 patterns
if 12 digits noise added 5 patterns got classified correctly from ttotal 5 patterns

if 13 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 13 digits noise added 3 patterns got classified correctly from ttotal 5 patterns

if 14 digits are reversed 2 patterns got classified correctly from ttotal 5 patterns
if 14 digits noise added 1 patterns got classified correctly from ttotal 5 patterns

if 15 digits are reversed 1 patterns got classified correctly from ttotal 5 patterns
if 15 digits noise added 3 patterns got classified correctly from ttotal 5 patterns

Displaying Pattern of erroneous input and its output when noise was introduced:

```
correctly identified
noisy input
 ####
#
#
###
#  #
#  #
 ###
output
 ####
#
#
####
#   #
#   #
 ###
correctly identified
noisy input
####
   #
   #
   #
  #
 #

output
#####
    #
   #
   #
  #
  #
 #
correctly identified
noisy input
 ###
#  #
   #
 ####
   #
  #
####
output
 ###
#   #
#   #
 ####
    #
    #
####
```

```
correctly identified
noisy input
 #
##
 #
  #


####
output
  #
 ##
# #
  #
  #
  #
#####
incorrectly identified
noisy input
 ###
#  #
#
 ####
    #

####
 noisy output
 ###
#   #
#
 ###
    #
    #
####
```

Displaying Pattern of erroneous input and its output when bits were reversed:

```
incorrectly identified
noisy input
# ###
 ## #
#  #
  #
 ##
  #

 noisy output
#####
 ## #
   #
   #
  #
  #
 #
correctly identified
noisy input
 #
#   #
#   #
  # #
# #
#  ##
###
output
 ###
#   #
#   #
 ####
    #
    #
####

correctly identified
noisy input
# #
  # #
 ##

  #
  # #
#####
output
  #
 ##
# #
  #
  #
  #
#####
```

Part 3 Spurious Patterns

Find a few of the spurious patterns. These are patterns that the neural network recalls but they are not in the training set. Report exactly how you find them so that your results can be verified.

- For the combinations for which all the combinations are correctly classified , I found the combination of digit that has not been used in that particular combination. And gave as a input to NN and checked whether the pattern is still successfully classified or not but could not get a single spurious pattern with this method.

Output of this is as below:

```
0 1 2 3 4
count of correctly identified patterns= 5

Checking digits not present in combination
0 Spurious patterns are available for this pattern among digits not available in sample



0 1 2 4 6
count of correctly identified patterns= 5

Checking digits not present in combination
0 Spurious patterns are available for this pattern among digits not available in sample



0 1 2 4 7
count of correctly identified patterns= 5

Checking digits not present in combination
0 Spurious patterns are available for this pattern among digits not available in sample



0 1 3 4 7
count of correctly identified patterns= 5

Checking digits not present in combination
0 Spurious patterns are available for this pattern among digits not available in sample
```

- To overcome this I randomly changed input vector by randomly reversing some bits. And new pattern I multiplied with weight matrix . It gives very few spurious patterns.

Output of spurious patterns when randomly flipped the bits

```
4 6 7 8
count of correctly identified patterns= 4

correctly identified spurious pattern
sample pattern already present
 ####
#
#
####
#   #
#   #
 ###
Spurious input pattern
 ###
#
#
####
#   #
#   #
 ###
output
 ###
#
#
####
#   #
#   #
 ###
if 1 digits noise reversed 1  spurious patterns got classified correctly
if 2 digits noise reversed 0  spurious patterns got classified correctly
if 3 digits noise reversed 0  spurious patterns got classified correctly
if 4 digits noise reversed 0  spurious patterns got classified correctly
if 5 digits noise reversed 0  spurious patterns got classified correctly
if 6 digits noise reversed 0  spurious patterns got classified correctly
if 7 digits noise reversed 0  spurious patterns got classified correctly
if 8 digits noise reversed 0  spurious patterns got classified correctly
if 9 digits noise reversed 0  spurious patterns got classified correctly
if 10 digits noise reversed 0  spurious patterns got classified correctly
if 11 digits noise reversed 0  spurious patterns got classified correctly
if 12 digits noise reversed 0  spurious patterns got classified correctly
if 13 digits noise reversed 0  spurious patterns got classified correctly
if 14 digits noise reversed 0  spurious patterns got classified correctly
if 15 digits noise reversed 0  spurious patterns got classified correctly

4 6 7 9
count of correctly identified patterns= 4

if 1 digits noise reversed 0  spurious patterns got classified correctly
```

```
if 2 digits noise reversed 0  spurious patterns got classified correctly
if 3 digits noise reversed 0  spurious patterns got classified correctly
if 4 digits noise reversed 0  spurious patterns got classified correctly
if 5 digits noise reversed 0  spurious patterns got classified correctly
if 6 digits noise reversed 0  spurious patterns got classified correctly
if 7 digits noise reversed 0  spurious patterns got classified correctly
if 8 digits noise reversed 0  spurious patterns got classified correctly
if 9 digits noise reversed 0  spurious patterns got classified correctly
if 10 digits noise reversed 0  spurious patterns got classified correctly
if 11 digits noise reversed 0  spurious patterns got classified correctly
if 12 digits noise reversed 0  spurious patterns got classified correctly
if 13 digits noise reversed 0  spurious patterns got classified correctly
if 14 digits noise reversed 0  spurious patterns got classified correctly
if 15 digits noise reversed 0  spurious patterns got classified correctly


1 2 4 7
count of correctly identified patterns= 4

correctly identified spurious pattern
sample pattern already present
 ###
#   #
   #
  #
  #
 #
#####
Spurious input pattern
 ###
#   #
   #
  #
  #
 ##
#####
output
 ###
#   #
   #
  #
  #
 ##
#####
if 1 digits noise reversed 1  spurious patterns got classified correctly
if 2 digits noise reversed 0  spurious patterns got classified correctly
if 3 digits noise reversed 0  spurious patterns got classified correctly
if 4 digits noise reversed 0  spurious patterns got classified correctly
if 5 digits noise reversed 0  spurious patterns got classified correctly
if 6 digits noise reversed 0  spurious patterns got classified correctly
if 7 digits noise reversed 0  spurious patterns got classified correctly
if 8 digits noise reversed 0  spurious patterns got classified correctly
if 9 digits noise reversed 0  spurious patterns got classified correctly
if 10 digits noise reversed 0  spurious patterns got classified correctly
if 11 digits noise reversed 0  spurious patterns got classified correctly
```

```
if 12 digits noise reversed 0  spurious patterns got classified correctly
if 13 digits noise reversed 0  spurious patterns got classified correctly
if 14 digits noise reversed 0  spurious patterns got classified correctly
if 15 digits noise reversed 0  spurious patterns got classified correctly
1 2 4 8
count of correctly identified patterns= 4
```