GRAY TO BINARY CODE CONVERTER

Submitted by:
ANURAG DUBEY (15MI436)
ROHIT NAGAR (15MI438)

Objective:

To create a model for prediction of output in a gray to binary code converter.

Since we know that for a specific gray code we have a specific binary code which represents a decimal number. We have trained our model with different values of gray code in 5 bits and after the training, our model is able to predict the binary code for the given gray code provided the condition that it must lie within the 5 bits.

Methodology:

The Output values Generation as:

If the GRAY CODE is 00010 then output BINARY CODE will be 00011

If the GRAY CODE is 11011 then BINARY CODE will be 10010

If the GRAY CODE is 01011 then BINARY CODE will be 01101

If the GRAY CODE is 111111 then BINARY CODE will not possible.

Data Generation:-

Decimal Number	Gray Code	Binary Code
0	00000	00000
1	00001	00001
2	00011	00011
3	00010	00011
4	00110	00100
5	00111	00101
6	00101	00110
7	00100	00111
8	01100	01000
9	01101	01001
10	01111	01010
11	01110	01011

12	01010	01100
13	01011	01101
14	01001	01110
15	01000	01111
16	11000	10000
17	11001	10001
18	11011	10010
19	11010	10011
20	11110	10100
21	11111	10101
22	11101	10110
23	11100	10111
24	10100	11000
25	10101	11001
26	10111	11010
27	10110	11011
28	10010	11100
29	10011	11101
30	10001	11110
31	10000	11111

Data Generation:

Data has been generated using conditions of Gray to Binary code conversion for 5 bits.

Data is generated using the different conditions of all the possible cases in 5 bits which is shown above.

Network Layers:

The layers used are
1. Dense layer with 9 neurons with 'relu' as activation function with input dimension 2.
2. Dense layer with 7 neurons with 'relu' as activation function.
3. Dense layer with 5 neurons and activation function as 'relu'.
4. Dense layer with 3 neurons and activation function as 'relu'
5. Dense layer with 1 neuron and activation function as 'relu' is the output layer.

Dense Layer:

A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix W, a bias vector b, and the activations of previous layer a.

Output:

```
[142] result=model.predict(X_test)

[143] result

array([[ 1114.218   ],
       [11124.296   ],
       [ 1115.1089  ],
       [   11.627534],
       [11022.218   ],
       [ 1004.7365  ],
       [10032.3     ]], dtype=float32)

[144] X_test

array([[   11,  1110],
       [   22, 11101],
       [   10,  1111],
       [    2,    11],
       [   16, 11000],
       [   14,  1001],
       [   28, 10010]])
```

Fig.: Comparison between the test data and predicted date

```
[145] plt.scatter(range(7),result,c='r')
      plt.scatter(range(7),y_test,c='g')
      plt.show()
```

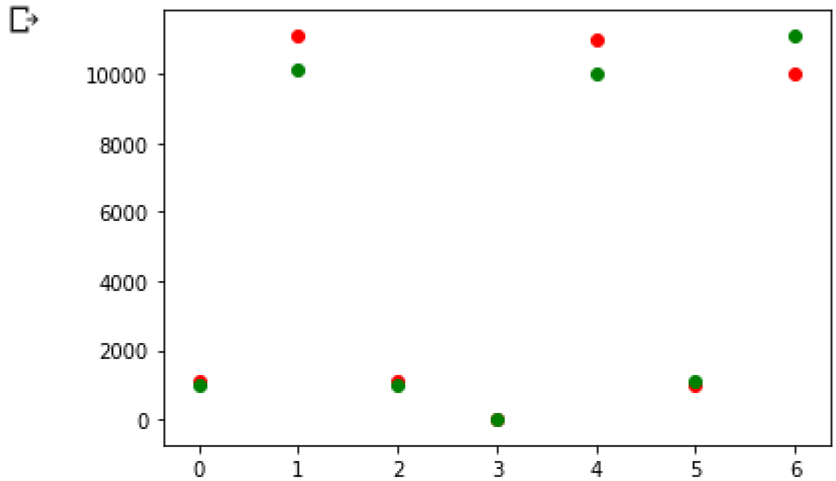

Fig.: Graph between the predicted values and test values