Experiment:

Below I have included and discussed every graph that was outputted from the code that I have developed for this experiment. Along with this write-up, a demo video and the source code were attached to the submission package for reference.

To follow how each graph is plotted please refer to the code provided for instructive comments for every major line of code. The code is easy to follow as it is written in modules for repeated calculations and concise variable names for each statement as needed.

Results and Discussion:

To start off, the data is first imported from the local machine then split into 70 % training data, 15% testing data and 15% validation data.

```
if __name__ == '__main__':
    #Importing the data from local machine using numpy library
    banknote_db = np.loadtxt(fname="data_banknote_authentication.txt", delimiter=',')
    #randomize using student ID
    np.random.seed(7623)
    np.random.shuffle(banknote_db)
    #Split the data into y and X sets
    x_data = banknote_db[:, :-1]
    y_data = banknote_db[:, -1]
    #Splitting the data into 60% train, 20% test, 20% valid
    X_train_set, temp, t_train_set, t_Temp = train_test_split(x_data, y_data, test_size=0.3, train_size=0.7, random_state=7623)
    X_valid_set, X_test_set, t_valid_set, t_test_set = train_test_split(temp, t_Temp, test_size = 0.5, train_size = 0.5, random_state=7623)
    #Scaling
    sc = StandardScaler()
    X_train_set = sc.fit_transform(X_train_set)
    X_valid_set = sc.transform(X_valid_set)
    X_test_set = sc.transform(X_test_set)
    t_train_set = np.expand_dims(t_train_set, axis=1)
    t_test_set = np.expand_dims(t_test_set, axis=1)
    t_valid_set = np.expand_dims(t_valid_set, axis=1)
```

The code was attached for reference with instructive comments to explain the functionality of each major block of code.

The code has a hard coded value of 1000 epochs for which the data was analyzied fully 1000 times to improve the choice of the vector of parameters as well as to lower the misclassification rate.

For a model with 2 features, we calculate the lowest misclassification rate:

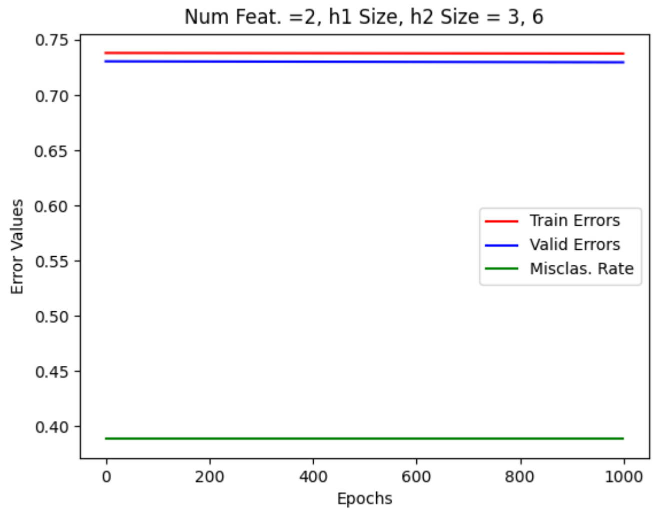


Figure 0: Error values vs Epochs for 2 features

Below is the output on the console:

Num. of features = 2

h1 size: 2 , h2 size: 2 , CE ERR: 0.7018 , MR: 0.4469

h1 size: 2 , h2 size: 3 , CE ERR: 0.8578 , MR: 0.7479

h1 size: 2 , h2 size: 4 , CE ERR: 0.7083 , MR: 0.4625

h1 size: 2 , h2 size: 5 , CE ERR: 0.7708 , MR: 0.226

h1 size: 2 , h2 size: 6 , CE ERR: 0.9264 , MR: 0.6104

h1 size: 3 , h2 size: 2 , CE ERR: 0.7069 , MR: 0.4969

h1 size: 3 , h2 size: 3 , CE ERR: 0.7291 , MR: 0.6708

h1 size: 3 , h2 size: 4 , CE ERR: 0.9624 , MR: 0.4469

h1 size: 3 , h2 size: 5 , CE ERR: 1.1497 , MR: 0.6771

h1 size: 3 , h2 size: 6 , CE ERR: 0.6508 , MR: 0.4375

h1 size: 4 , h2 size: 2 , CE ERR: 0.6928 , MR: 0.5156

h1 size: 4 , h2 size: 3 , CE ERR: 0.7458 , MR: 0.4719

h1 size: 4 , h2 size: 4 , CE ERR: 0.806 , MR: 0.6417

h1 size: 4 , h2 size: 5 , CE ERR: 0.6614 , MR: 0.4469

h1 size: 4 , h2 size: 6 , CE ERR: 0.8331 , MR: 0.4469

h1 size: 5 , h2 size: 2 , CE ERR: 0.7094 , MR: 0.4469

h1 size: 5 , h2 size: 3 , CE ERR: 0.9708 , MR: 0.599

h1 size: 5 , h2 size: 4 , CE ERR: 0.7619 , MR: 0.4469

h1 size: 5 , h2 size: 5 , CE ERR: 0.9811 , MR: 0.4833

h1 size: 5 , h2 size: 6 , CE ERR: 1.0649 , MR: 0.9115

h1 size: 6 , h2 size: 2 , CE ERR: 0.7953 , MR: 0.4469

h1 size: 6 , h2 size: 3 , CE ERR: 1.0341 , MR: 0.4469

h1 size: 6 , h2 size: 4 , CE ERR: 1.0394 , MR: 0.9135

h1 size: 6 , h2 size: 5 , CE ERR: 0.7176 , MR: 0.4292

h1 size: 6 , h2 size: 6 , CE ERR: 0.7916 , MR: 0.5531

The best choice if the data was only 2 features would be to have a first hidden layer size of 3 and the second hidden layer size of 6.

For a model with 3 features, we calculate the lowest misclassification rate:

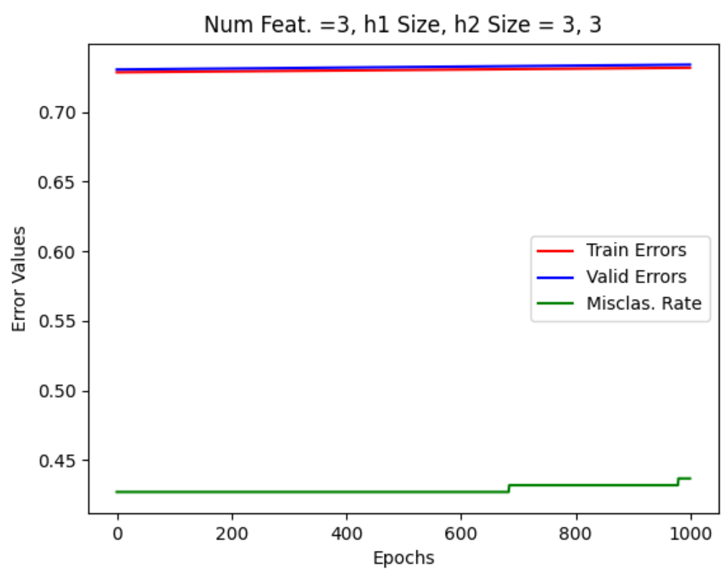


Figure 1: Error values vs Epochs for 3 features

Below is the output on the console:

Num. of features = 3

h1 size: 2 , h2 size: 2 , CE ERR: 0.7693 , MR: 0.4469

h1 size: 2 , h2 size: 3 , CE ERR: 0.7032 , MR: 0.4469

h1 size: 2 , h2 size: 4 , CE ERR: 0.691 , MR: 0.4469

h1 size: 2 , h2 size: 5 , CE ERR: 1.0586 , MR: 0.4469

h1 size: 2 , h2 size: 6 , CE ERR: 0.8194 , MR: 0.5656

h1 size: 3 , h2 size: 2 , CE ERR: 0.7711 , MR: 0.4469

h1 size: 3 , h2 size: 3 , CE ERR: 0.6856 , MR: 0.474

h1 size: 3 , h2 size: 4 , CE ERR: 0.8171 , MR: 0.4052

h1 size: 3 , h2 size: 5 , CE ERR: 0.813 , MR: 0.6552

h1 size: 3 , h2 size: 6 , CE ERR: 1.0233 , MR: 0.4792

h1 size: 4 , h2 size: 2 , CE ERR: 0.7385 , MR: 0.4958

h1 size: 4 , h2 size: 3 , CE ERR: 0.723 , MR: 0.4469

h1 size: 4 , h2 size: 4 , CE ERR: 0.7012 , MR: 0.7094

h1 size: 4 , h2 size: 5 , CE ERR: 0.9429 , MR: 0.8823

h1 size: 4 , h2 size: 6 , CE ERR: 0.8407 , MR: 0.4323

h1 size: 5 , h2 size: 2 , CE ERR: 0.7011 , MR: 0.4469

h1 size: 5 , h2 size: 3 , CE ERR: 1.3393 , MR: 0.4792

h1 size: 5 , h2 size: 4 , CE ERR: 0.7767 , MR: 0.4781

h1 size: 5 , h2 size: 5 , CE ERR: 1.3698 , MR: 0.7677

h1 size: 5 , h2 size: 6 , CE ERR: 0.9861 , MR: 0.4052

h1 size: 6 , h2 size: 2 , CE ERR: 0.7166 , MR: 0.7094

h1 size: 6 , h2 size: 3 , CE ERR: 1.2421 , MR: 0.7229

h1 size: 6 , h2 size: 4 , CE ERR: 1.3061 , MR: 0.4542

h1 size: 6 , h2 size: 5 , CE ERR: 0.8901 , MR: 0.5208

h1 size: 6 , h2 size: 6 , CE ERR: 1.0647 , MR: 0.7719

The best choice if the data was only 3 features would be to have a first hidden layer size of 3 and the second hidden layer size of 3.

For a model with 4 features, we calculate the lowest misclassification rate:

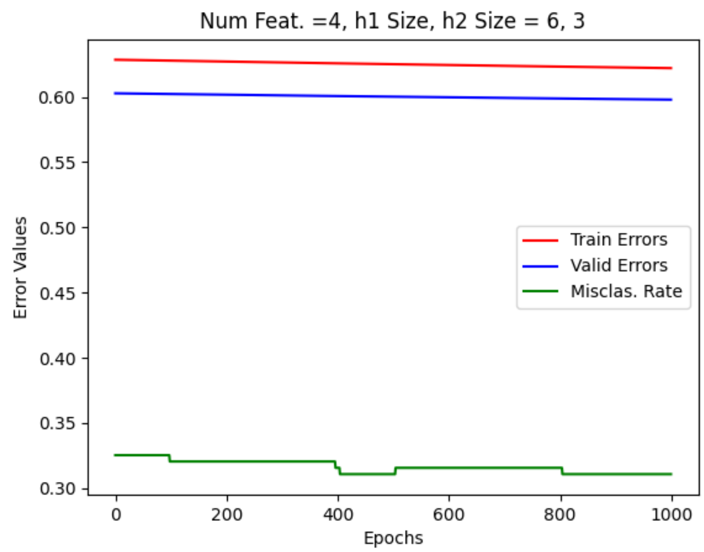


Figure 2: Error values vs Epochs for 4 features

Below is the output on the console:

Num. of features = 4

h1 size: 2 , h2 size: 2 , CE ERR: 0.7066 , MR: 0.4469

h1 size: 2 , h2 size: 3 , CE ERR: 0.6904 , MR: 0.4469

h1 size: 2 , h2 size: 4 , CE ERR: 0.6831 , MR: 0.4469

h1 size: 2 , h2 size: 5 , CE ERR: 0.8508 , MR: 0.8042

h1 size: 2 , h2 size: 6 , CE ERR: 0.9064 , MR: 0.4469

h1 size: 3 , h2 size: 2 , CE ERR: 0.7874 , MR: 0.525

h1 size: 3 , h2 size: 3 , CE ERR: 0.7963 , MR: 0.524

h1 size: 3 , h2 size: 4 , CE ERR: 1.4582 , MR: 0.9812

h1 size: 3 , h2 size: 5 , CE ERR: 1.3401 , MR: 0.4521

h1 size: 3 , h2 size: 6 , CE ERR: 0.9692 , MR: 0.4198

h1 size: 4 , h2 size: 2 , CE ERR: 0.9965 , MR: 0.8333

h1 size: 4 , h2 size: 3 , CE ERR: 0.7888 , MR: 0.4469

h1 size: 4 , h2 size: 4 , CE ERR: 0.7069 , MR: 0.5969

h1 size: 4 , h2 size: 5 , CE ERR: 1.5873 , MR: 0.7312

h1 size: 4 , h2 size: 6 , CE ERR: 1.2782 , MR: 0.7823

h1 size:	5 , h2 size:	2 , CE ERR:	0.6615 , MR:	0.424
h1 size:	5 , h2 size:	3 , CE ERR:	1.1275 , MR:	0.4469
h1 size:	5 , h2 size:	4 , CE ERR:	0.7271 , MR:	0.3573
h1 size:	5 , h2 size:	5 , CE ERR:	0.9718 , MR:	0.5719
h1 size:	5 , h2 size:	6 , CE ERR:	0.6924 , MR:	0.5083
h1 size:	6 , h2 size:	2 , CE ERR:	0.6976 , MR:	0.599
h1 size:	6 , h2 size:	3 , CE ERR:	0.6473 , MR:	0.3823
h1 size:	6 , h2 size:	4 , CE ERR:	1.5776 , MR:	0.9531
h1 size:	6 , h2 size:	5 , CE ERR:	1.5039 , MR:	0.4635
h1 size:	6 , h2 size:	6 , CE ERR:	2.0629 , MR:	0.8271

The best choice if the data was only 4 features would be to have a first hidden layer size of 6 and the second hidden layer size of 3.

From the data above, we can see that for 4 features, we get the lowest misclassification rate occurs when we have two hidden layers, the first of which is of size 6 and second is of size 3. We get a misclassification rate of approximately 0.3823 and a cross validation error of 0.6473.

Below is printed in the console upon running the code:

Results as discovered by running the code:

No. of features = 4

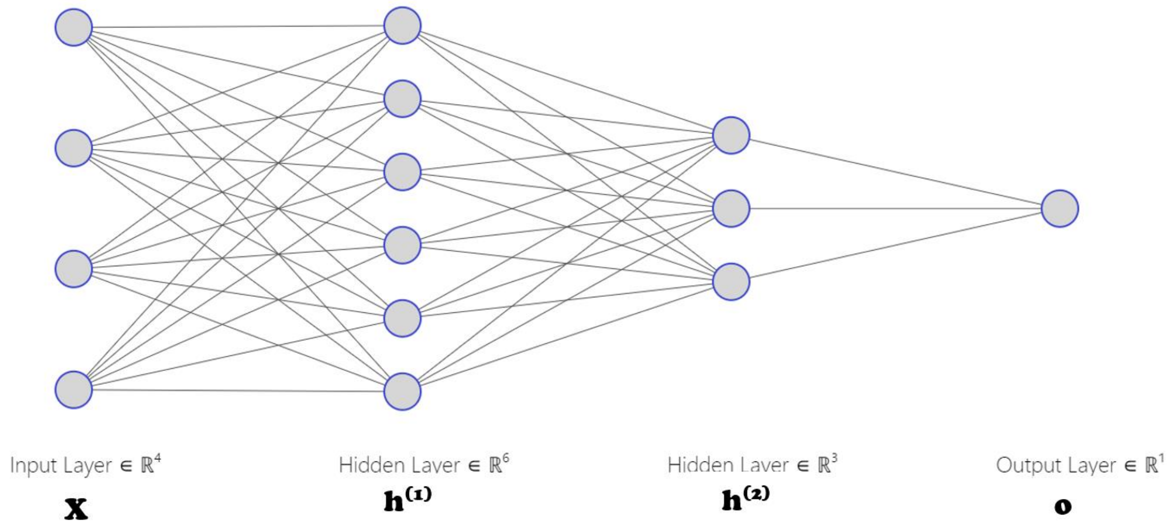
Size of Hidden Layer h1 = 6

Size of Hidden Layer h2 = 3

The minimized Test Error = 0.6188245016919025

The minimized Valid Error = 0.647297648959275

Diagram of the Neural Network:



Above is a visualization of how the neural network arcticture looks like.

After experimentation, we found that lower hidden layer sizes result in lower misclassification rates as compared to higher dimensional sizes for the hidden layers. Therefore, I decided to run the code for sizes 2 to 6 for each hidden layer to minimize the complexity time and performance of the developed code.

Also, for a learning rate choice of 0.005, we notice the cost plots to have a decreasing behaviour as the number of the epoch increases. This is expected for cross entropy losses which was the chosen function to compute the loss for this experiment. With other choices of the loss function, we have less accurate results with an improvement in performance. For which, we can conclude, that the choice of the loss function is also important as it would affect the accuracy of the predictions and the performance so the trade-off should be considered when deciding the archeticture of the neural network.

As mentioned above, for this experiment a first hidden layer n_1 of dimension 6 and a second layer n_2 of dimension 3 yields the lowest cross validation and misclassification rate, thus, it is the optimal choice for this dataset.