Pattern Recognition

Mini-project #6

Neural Networks

Instructor: Dr. Yazdi

Student : Hamze Ghaedi

Stdn: 9831419

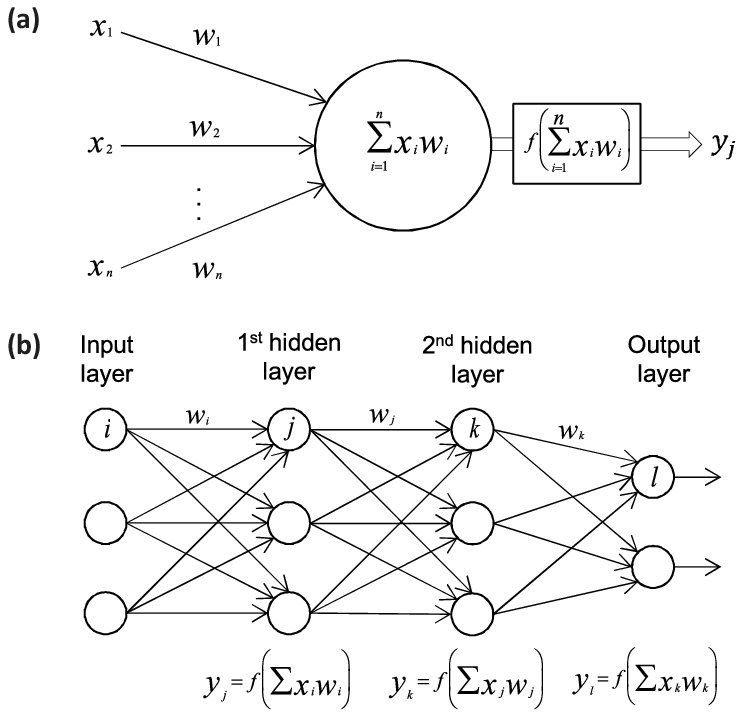
Abstract

Neural networks are one of the most promising ideas in the field of pattern recognition and machine learning. Their capabilities in capturing complex relations among inputs and outputs (hence supervised problems!), makes them a good function approximator or a good classifier schema for broad variety of datasets. although, requiring thousands of samples (much more than classical machine learning techniques) usually restricts their applicability.

In this project we will study the neural networks and their properties.

Neural Networks

An artificial neural network, similar to human brain, is an ensemble of interconnected units called neuron (or neurode). The figure below illustrates a typical unit (or neuron) of a neural network along with a neural network:



Each neuron, takes a vector of inputs and after some processing, outputs a signal indicates the response of the neuron to the given inputs. Neurons are connected together using links having unknown weights. In our neuron depicted above, the processing step consists of summing weighed inputs up and applying a non-linear function (called activation function) to the result in order to produce the corresponding output signal.

Activation Functions

Generally, there is no way to find the best activation function for a typical network, some of the most used activation functions are listed below.

1) Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

2) Tanh

$$f(x) = \tanh(x)$$

3) ReLu

$$f(x) = \max(0, x)$$

Training a network

Training a neural network is accomplished by finding appropriate values for the link weights with respect to some criterions. Generally, the criterion, measures similarity (or difference) between network output per given sample and its corresponding true target. Some popular criterion functions (or loss function) are listed below:

1) Mean Square Error (MSE)

$$J(x;W) = \Sigma_x \left|\left|f(x;W) - y\right|\right|^2$$

2) Mean Absolute Error (MAE)

$$J(x;W) = \Sigma_x \left|\left|f(x;W) - y\right|\right|$$

Cross entropy loss function

Two loss functions introduced above, are natural for regression problems. Cross entropy is common loss function best suited for classification problems. assuming outputs are one-hot encoded, Cross entropy loss function is as follows:

$$L = -\frac{1}{N}\Sigma_{i=1}^{N} y_i \log\left(\hat{y}_i\right)$$

Where the y_i is ith element of true output vector (which is 0 or 1) and $\hat{y}_i$ is the ith element of the predicted vector after applying SoftMax transform.

Gradient Descent and Backpropagation

So, we must find unknown weights with respect to minimizing loss function hence, we have an optimization problem. Gradient descent algorithms are the most popular techniques currently have been used to solve a general optimization problem. When the number of layers of a neural network increased the sophisticated special version of GD algorithm called Backpropagation is implemented in order to train a multilayer net.

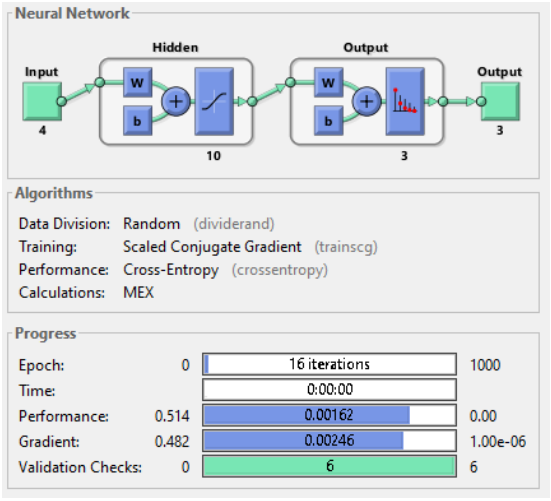
Problem #1

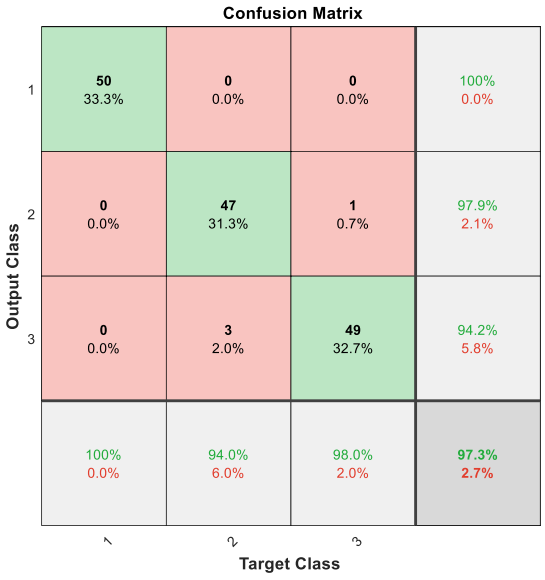
Iris Flower Dataset

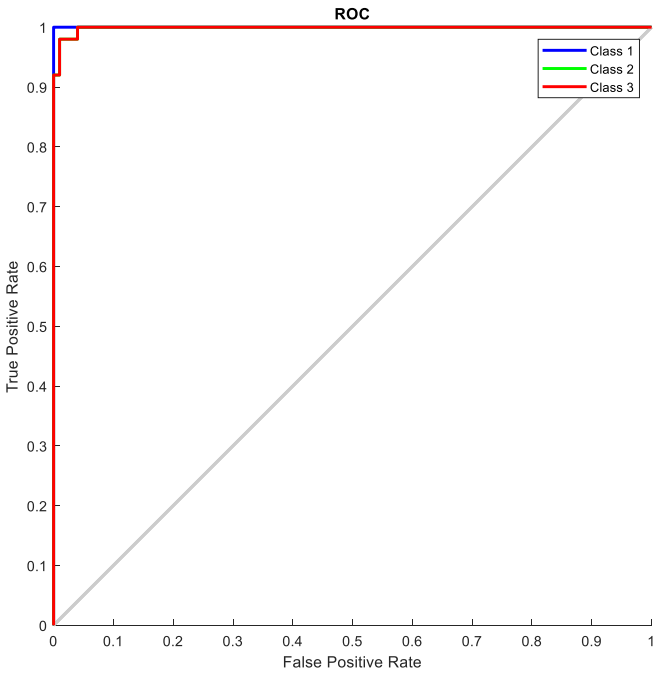
Iris Flowers dataset consists of 150 samples each with 4 features extracted from images of flowers. There are three categories of flowers hence, the targets are 3 elements vectors encoded in one-hot format.

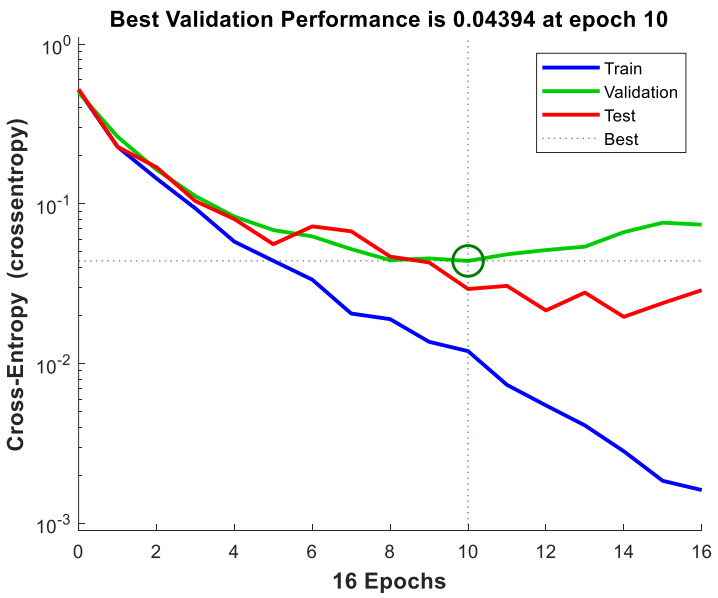
PART (a)

Results of classifying Iris flowers using MATLAB pattern network with default adjustment is as follows:

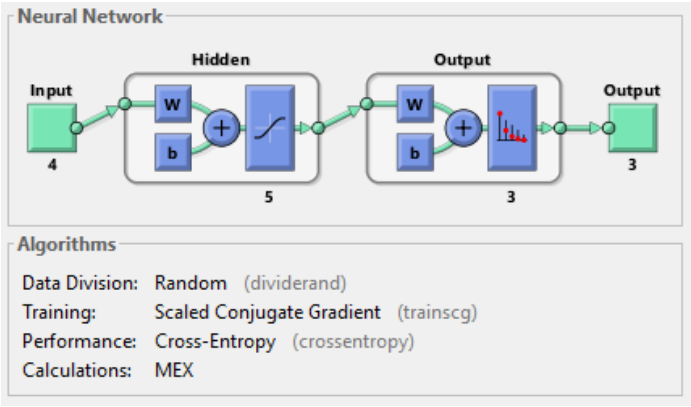


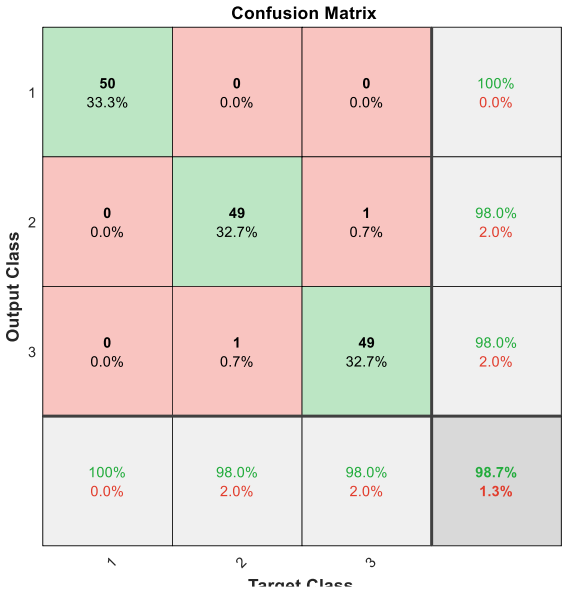


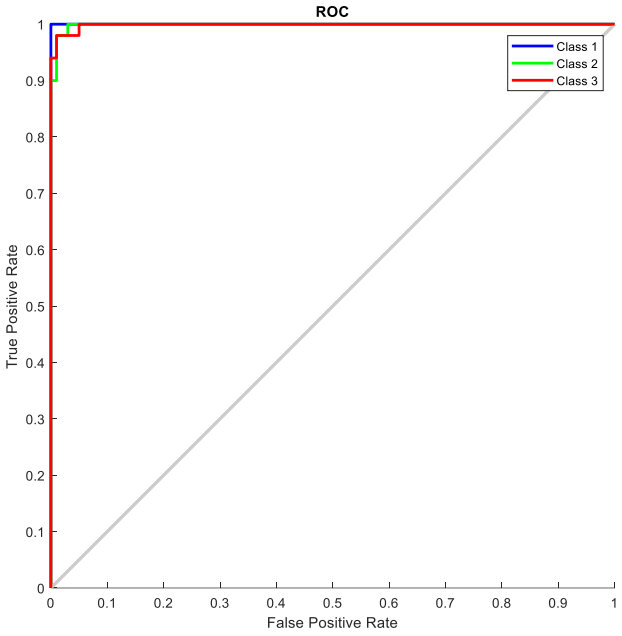


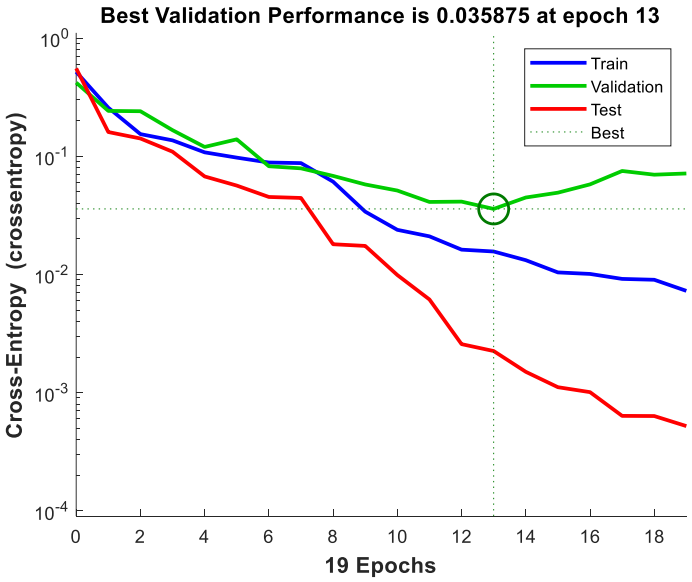


PART (b) – Hyperparameter adjustment (#neurons changed)

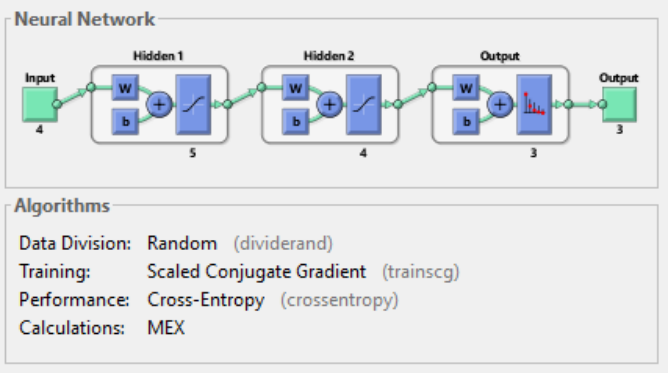


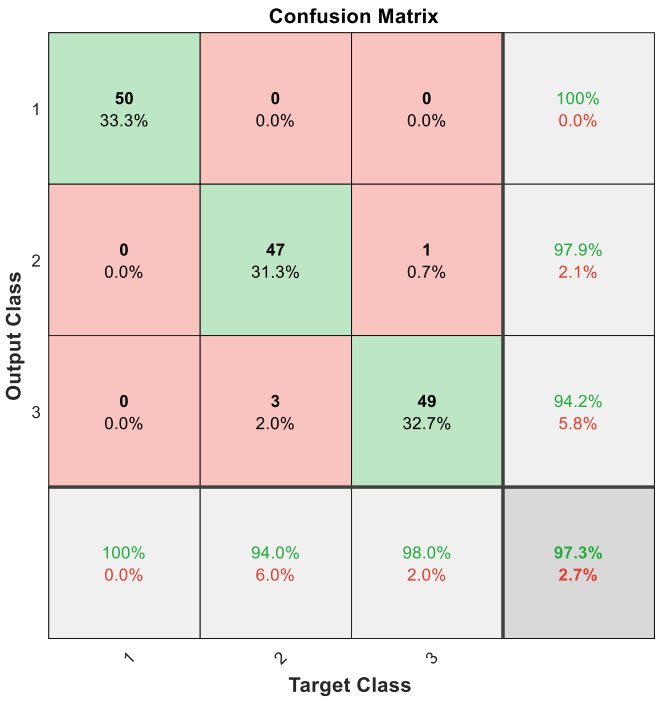


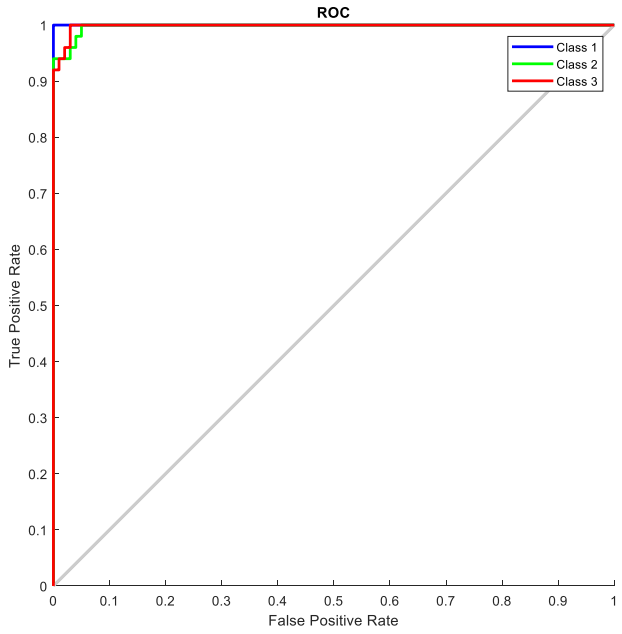


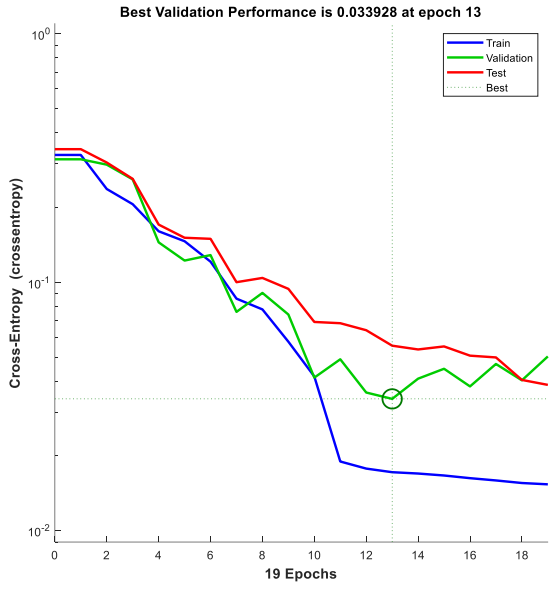


PART (b) – Hyperparameter adjustment (#neurons & #layers changed)

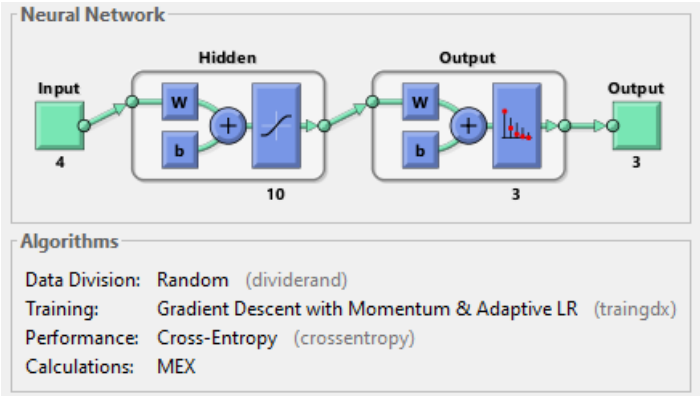


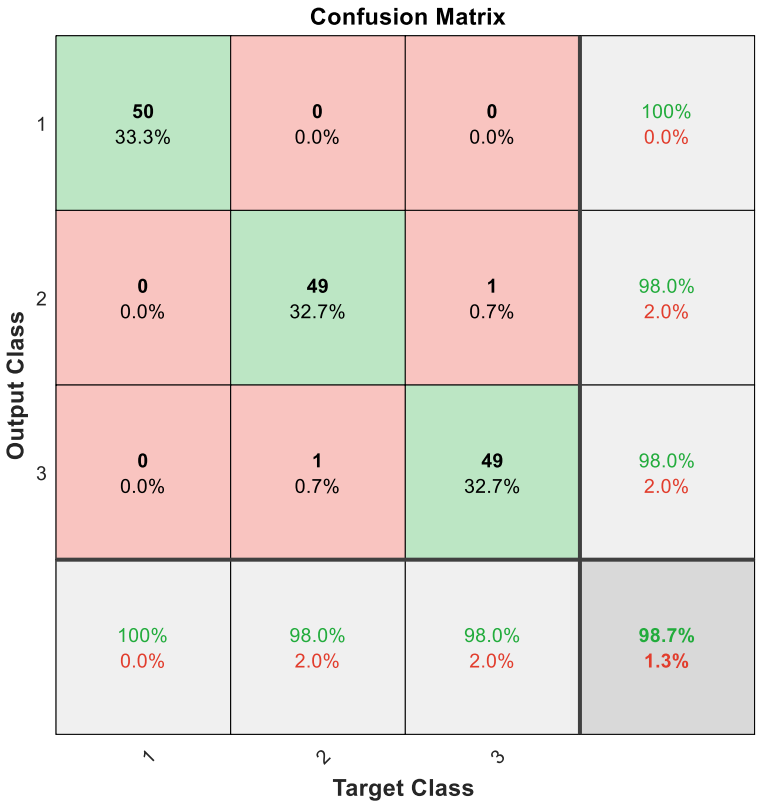


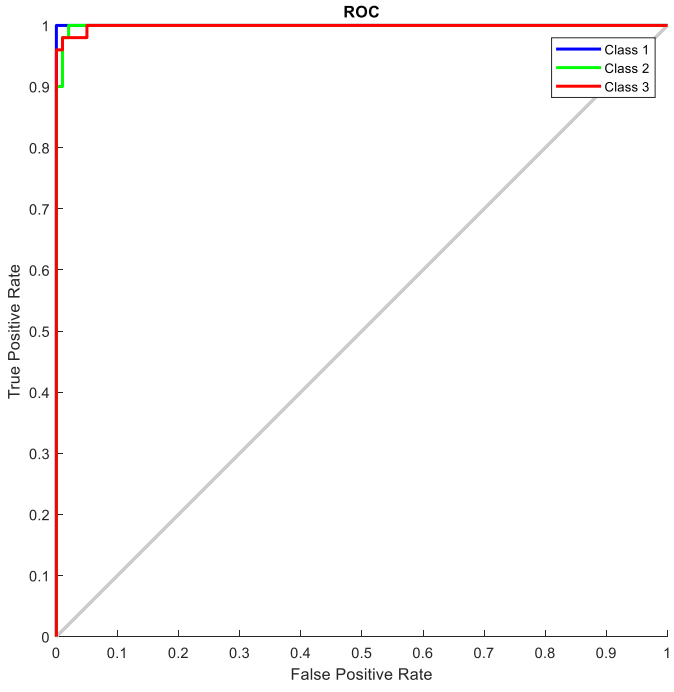


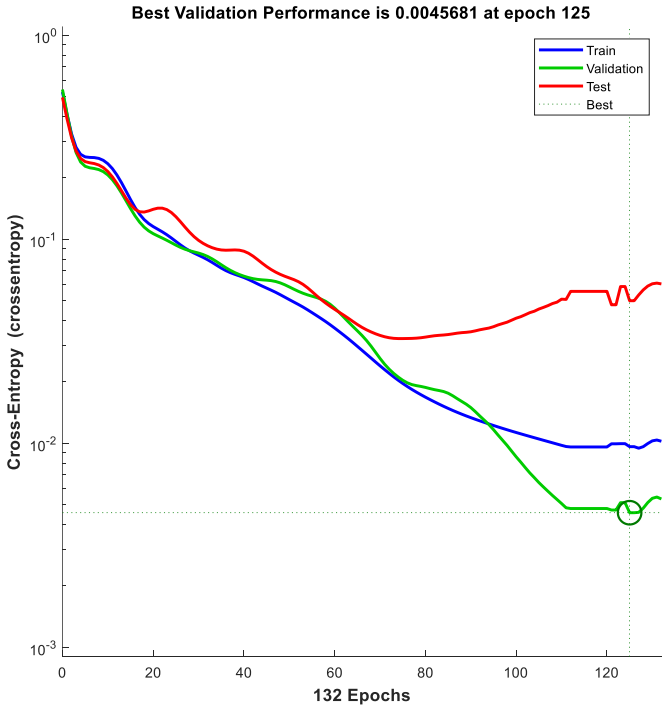


PART (b) – Hyperparameter adjustment (trainFcn changed & LR = 0.5)



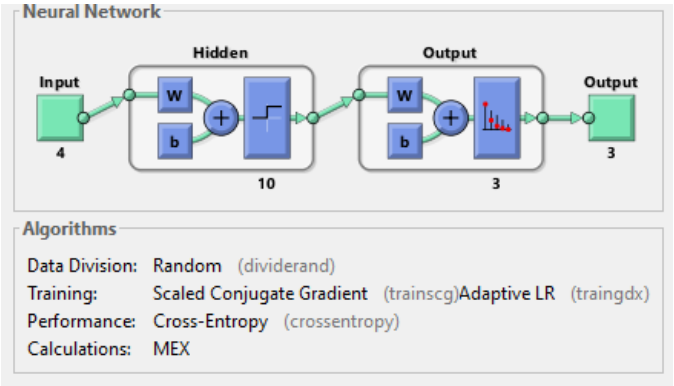


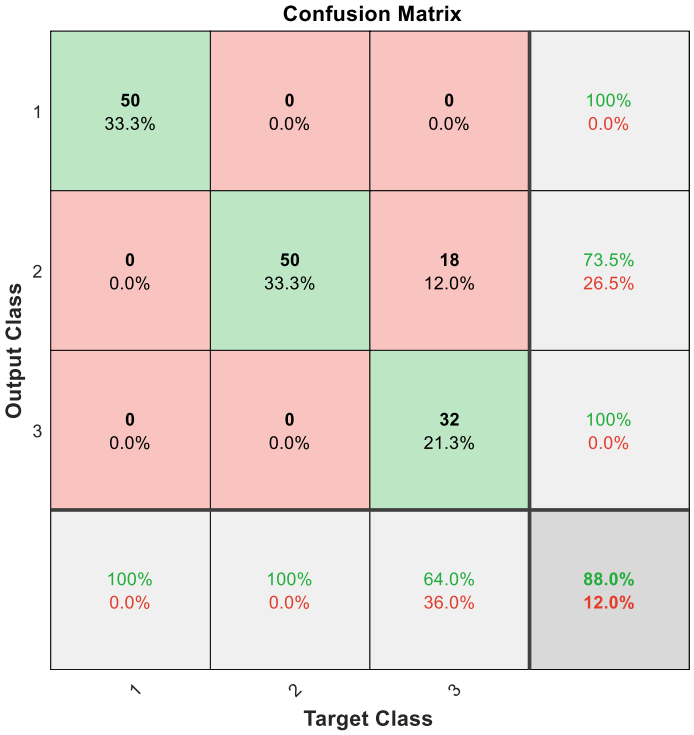


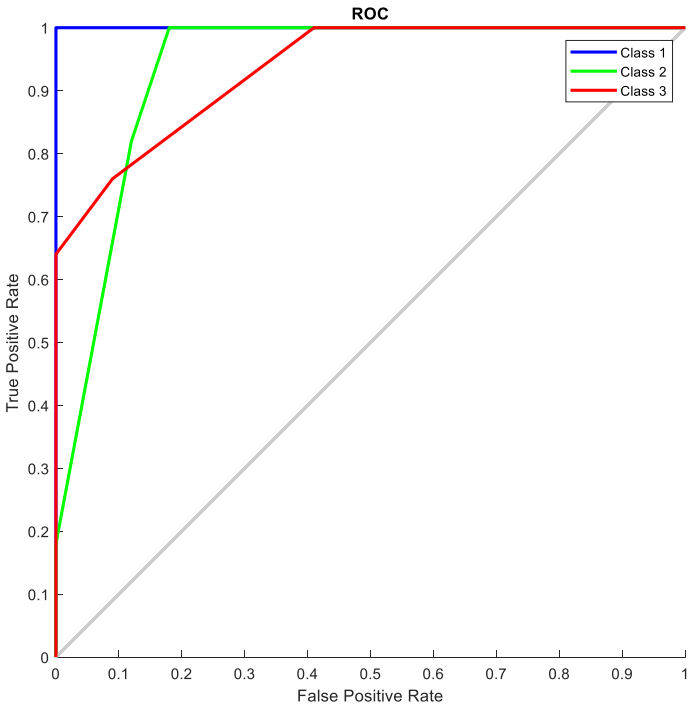


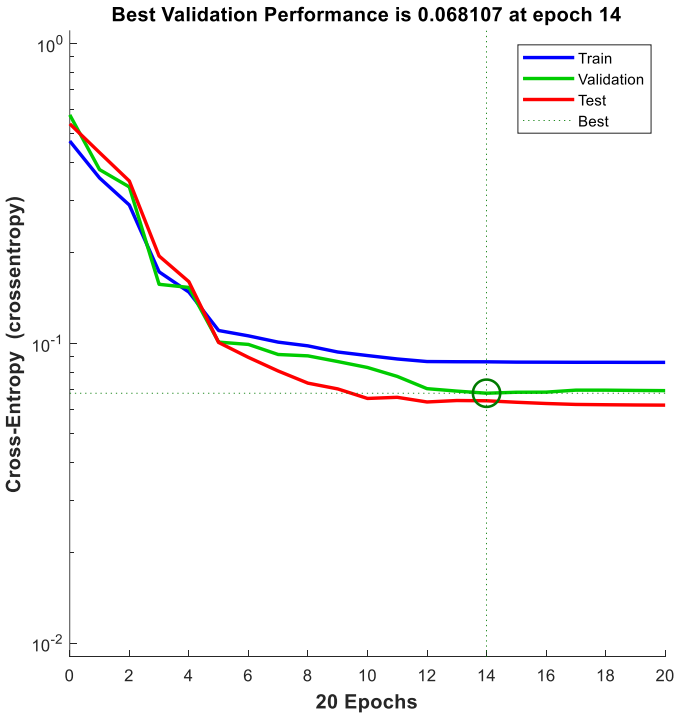
PART (b) – Hyperparameter adjustment (Activation function changed)

```
net.layers{1}.transferFcn = 'hardlim';
```

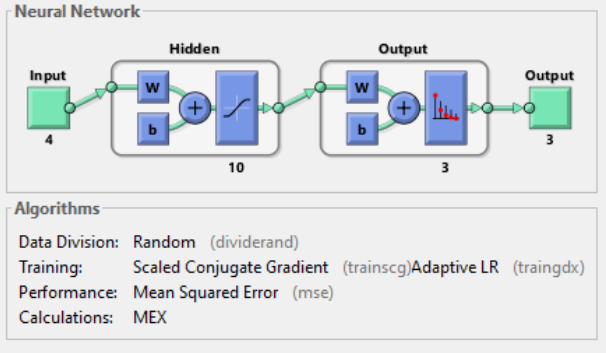


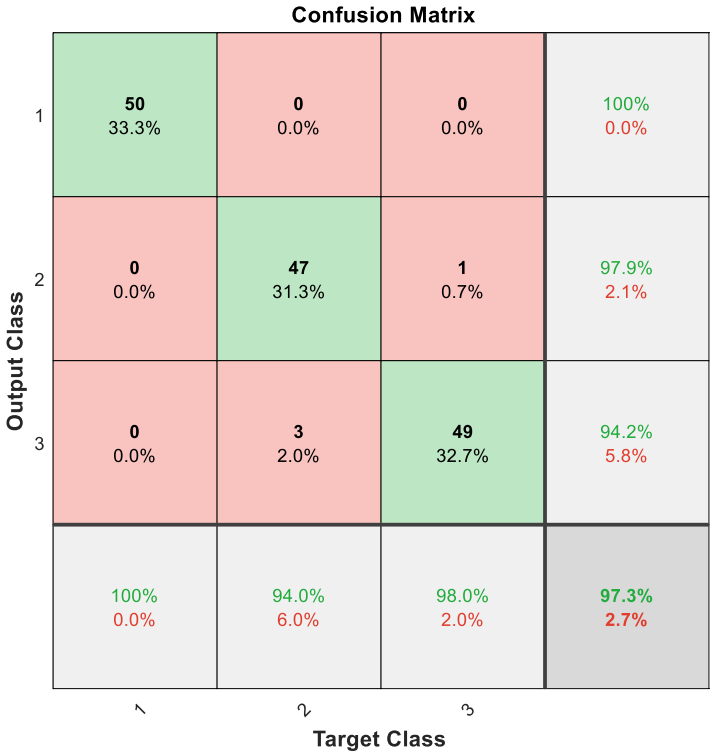


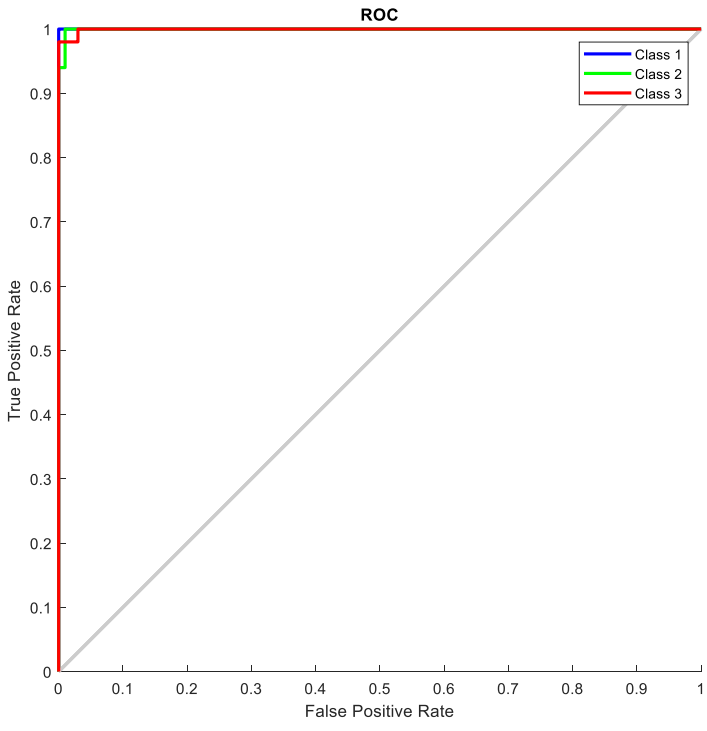


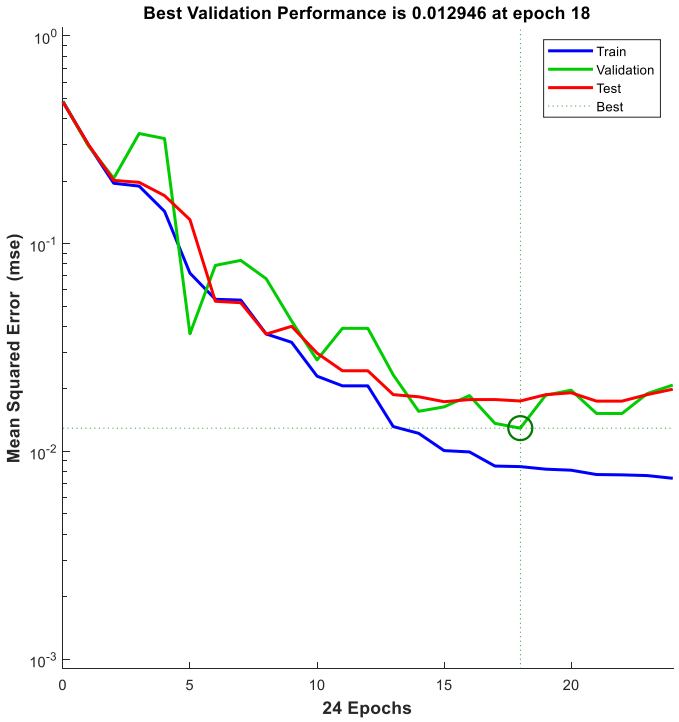


PART (b) – Hyperparameter adjustment (Loss function changed)



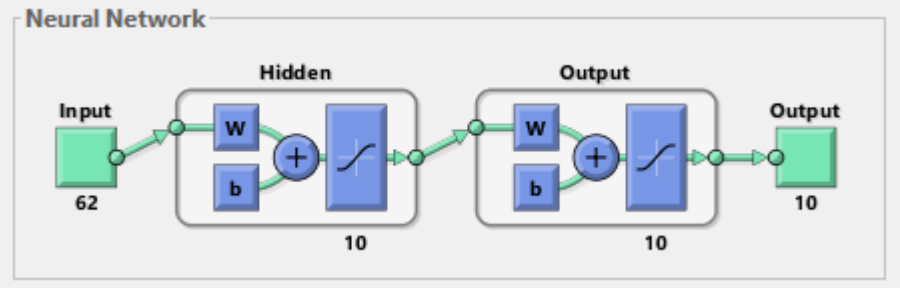


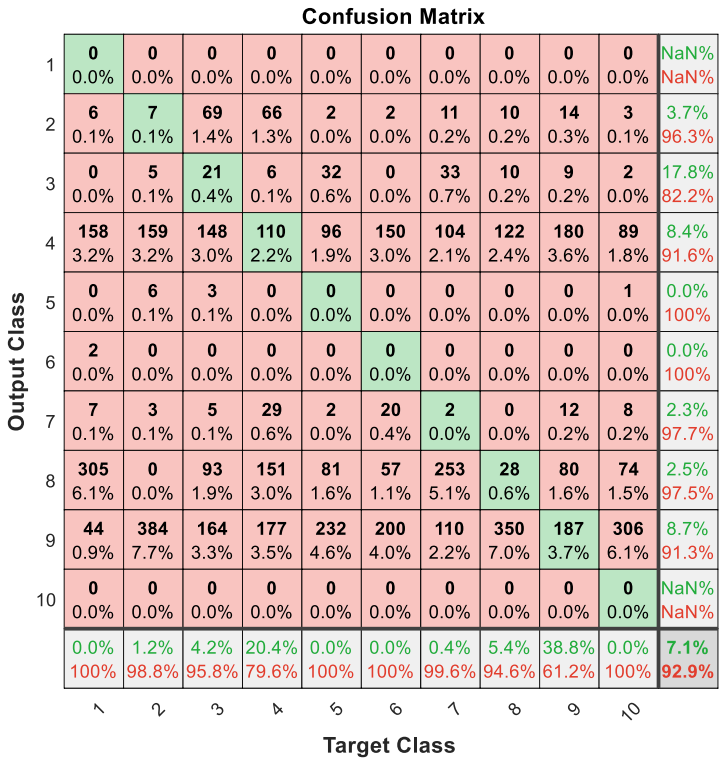


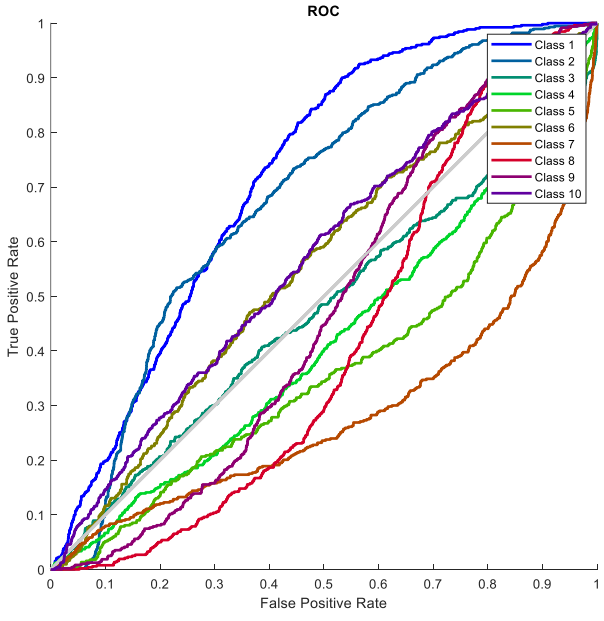


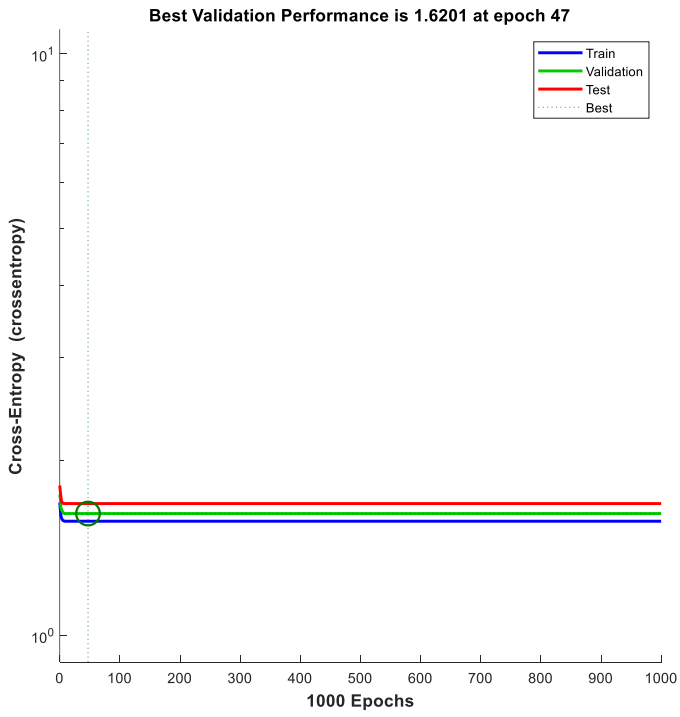
Problem #2

Fitting a neural network to MNIST dataset







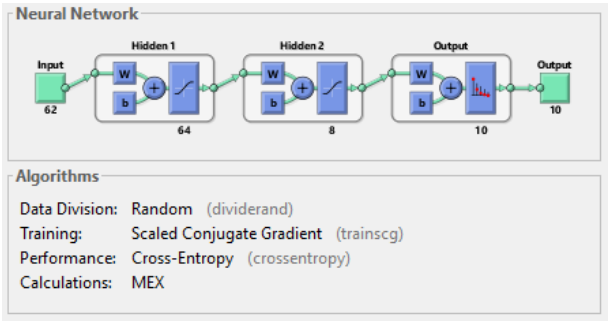


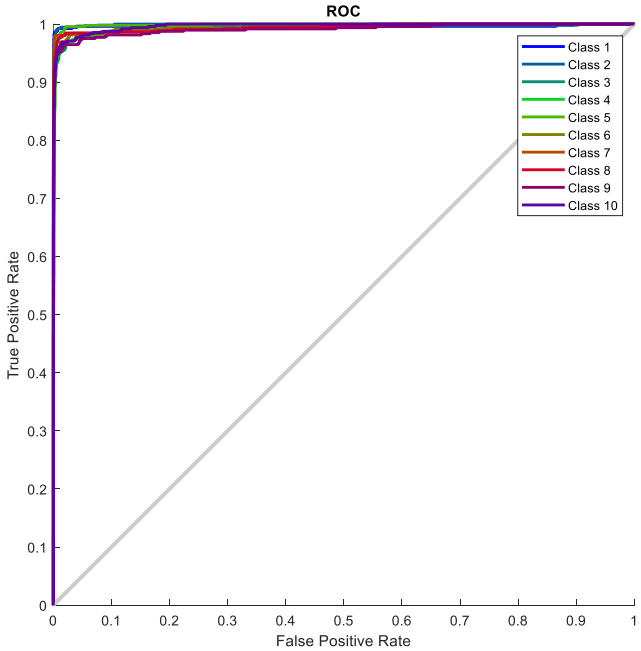
Dataset preparation

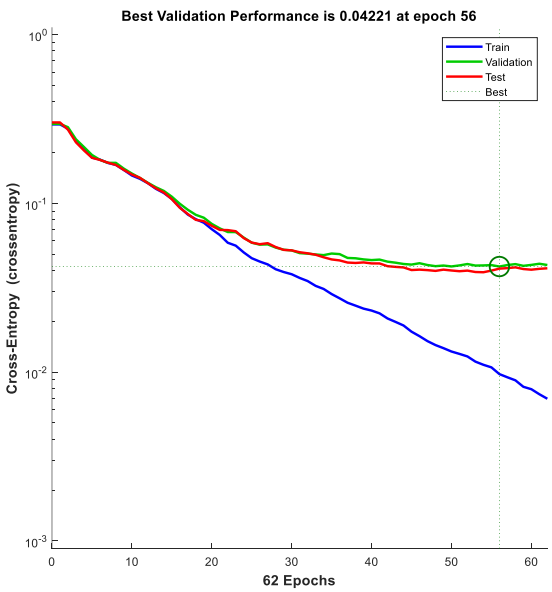
Script below, coverts the targets from sparse categorical encoding into one-hot format:

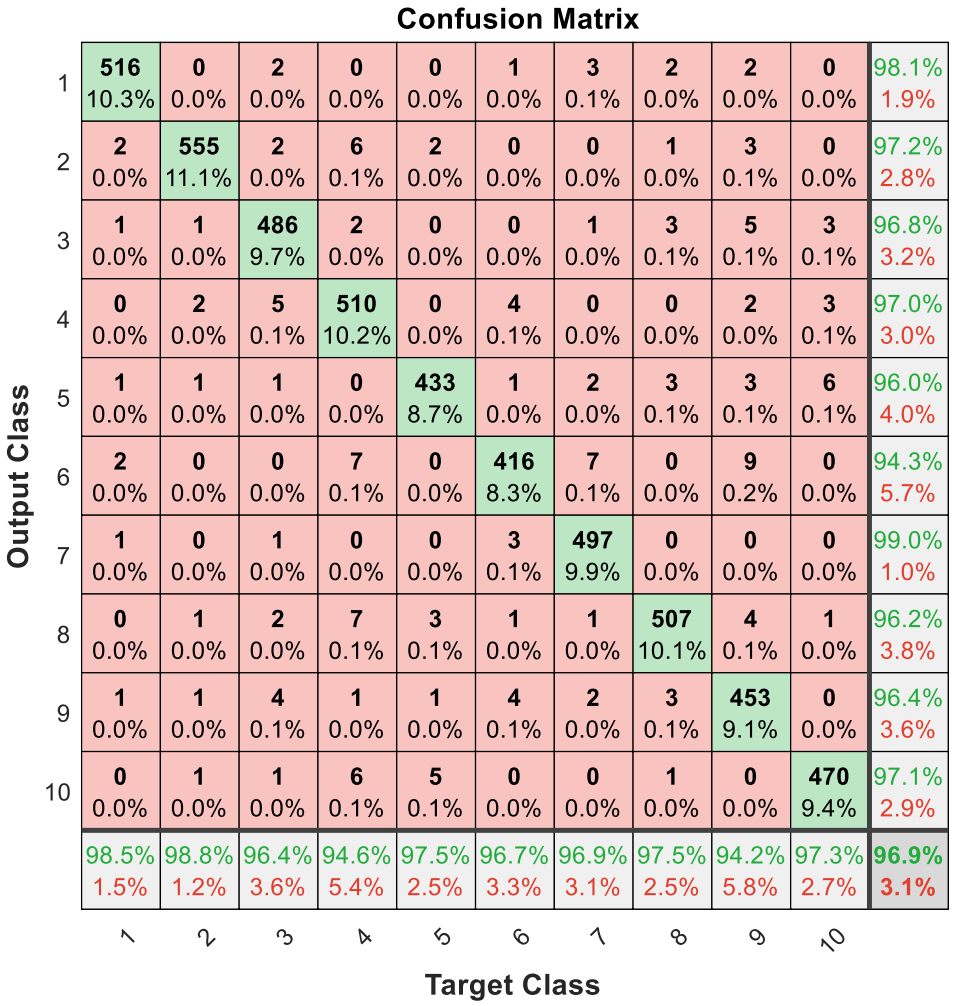
```
x_train = load('mnist\Train_Data.csv');

y_train = load('mnist\Train_labels.csv');

y_train_onehot = zeros(5000,10);

for i = 1:5000

y_train_onehot(i,y_train(i) + 1) = 1;

end
```

Fitting a neural network to MNIST dataset (#neurons & #layers changed)



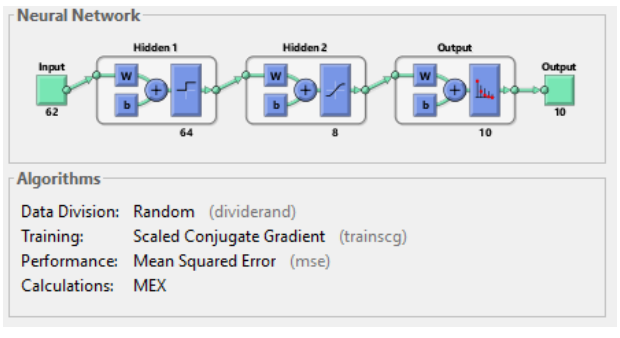


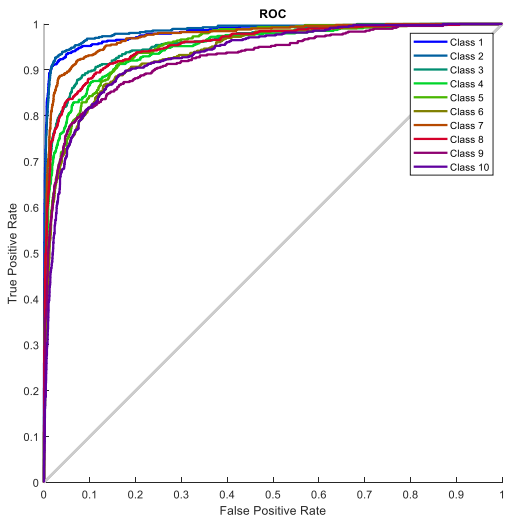


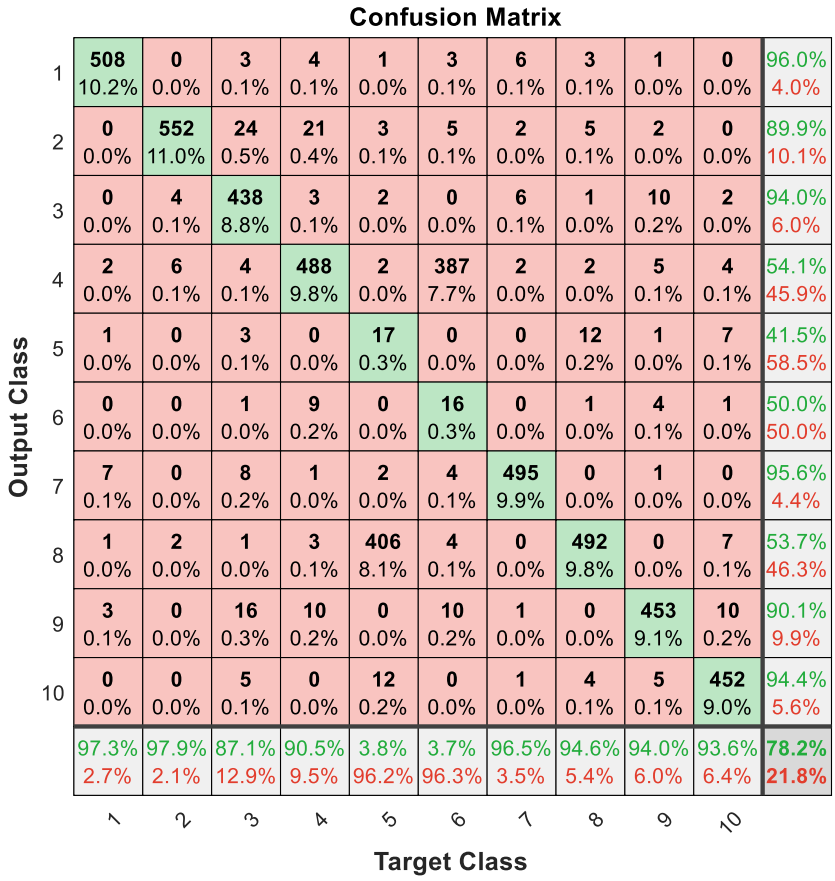


Changing activation function & loss function

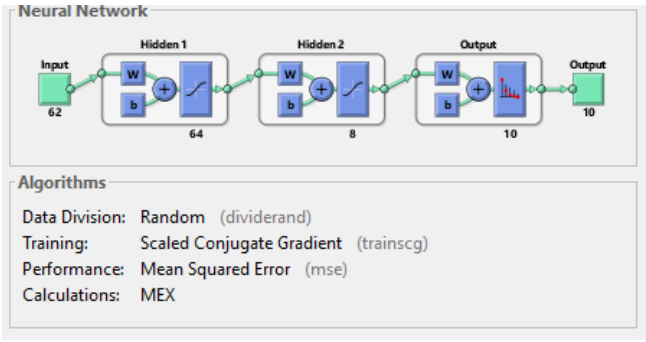
```
net.layers{1}.transferFcn = 'hardlim';
net.layers{2}.transferFcn = 'tansig';
```

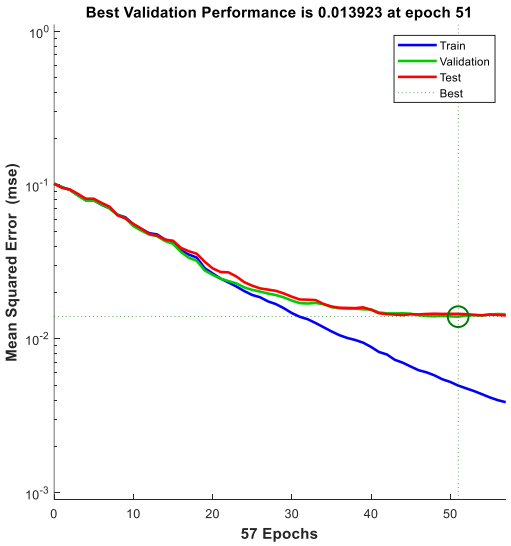


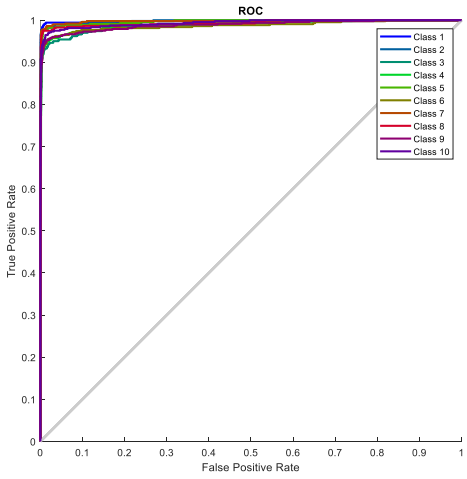




Changing loss function

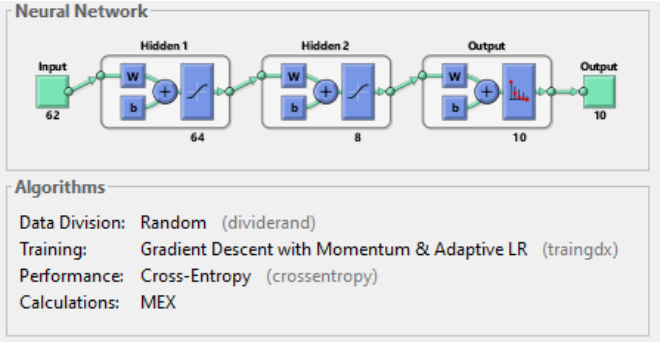


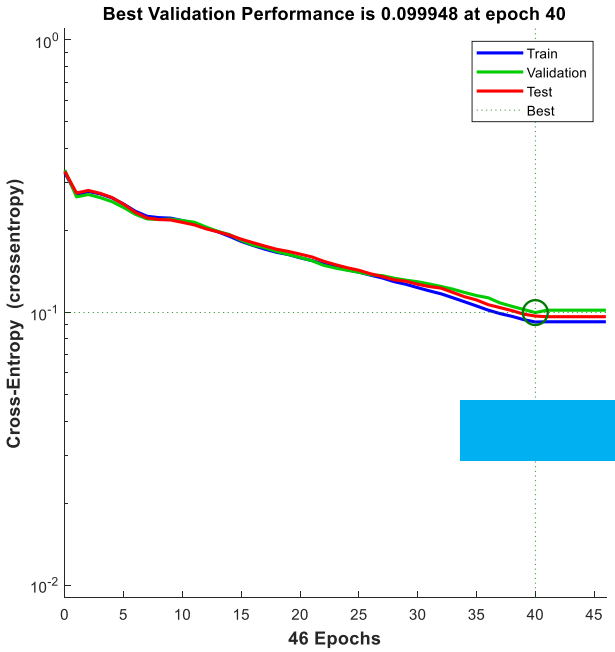


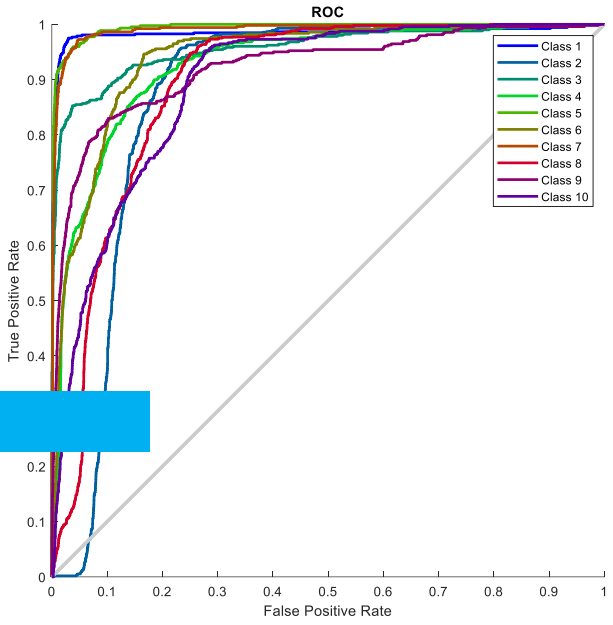



Confusion Matrix
Output Class
Target Class
1 511 10.2% 0 0.0% 3 0.1% 2 0.0% 0 0.0% 4 0.1% 3 0.1% 1 0.0% 1 0.0% 3 0.1% 96.8% 3.2%
2 0 0.0% 551 11.0% 3 0.1% 0 0.0% 3 0.1% 2 0.0% 0 0.0% 2 0.0% 3 0.1% 1 0.0% 97.5% 2.5%
3 3 0.1% 4 0.1% 463 9.3% 3 0.1% 4 0.1% 3 0.1% 2 0.0% 3 0.1% 6 0.1% 2 0.0% 93.9% 6.1%
4 2 0.0% 1 0.0% 14 0.3% 504 10.1% 0 0.0% 5 0.1% 1 0.0% 4 0.1% 7 0.1% 8 0.2% 92.3% 7.7%
5 1 0.0% 0 0.0% 4 0.1% 1 0.0% 422 8.4% 3 0.1% 2 0.0% 2 0.0% 1 0.0% 6 0.1% 95.5% 4.5%
6 2 0.0% 3 0.1% 3 0.1% 7 0.1% 2 0.0% 400 8.0% 3 0.1% 3 0.1% 8 0.2% 1 0.0% 92.6% 7.4%
7 1 0.0% 0 0.0% 5 0.1% 2 0.0% 3 0.1% 3 0.1% 501 10.0% 0 0.0% 4 0.1% 0 0.0% 96.5% 3.5%
8 0 0.0% 1 0.0% 2 0.0% 4 0.1% 0 0.0% 2 0.0% 0 0.0% 501 10.0% 2 0.0% 3 0.1% 97.3% 2.7%
9 1 0.0% 2 0.0% 5 0.1% 12 0.2% 0 0.0% 5 0.1% 1 0.0% 0 0.0% 445 8.9% 2 0.0% 94.1% 5.9%
10 1 0.0% 2 0.0% 1 0.0% 4 0.1% 11 0.2% 2 0.0% 0 0.0% 4 0.1% 5 0.1% 457 9.1% 93.8% 6.2%
97.9% 2.1% 97.7% 2.3% 92.0% 8.0% 93.5% 6.5% 94.8% 5.2% 93.2% 6.8% 97.7% 2.3% 96.3% 3.7% 92.3% 7.7% 94.6% 5.4% 95.1% 4.9%
1 2 3 4 5 6 7 8 9 10

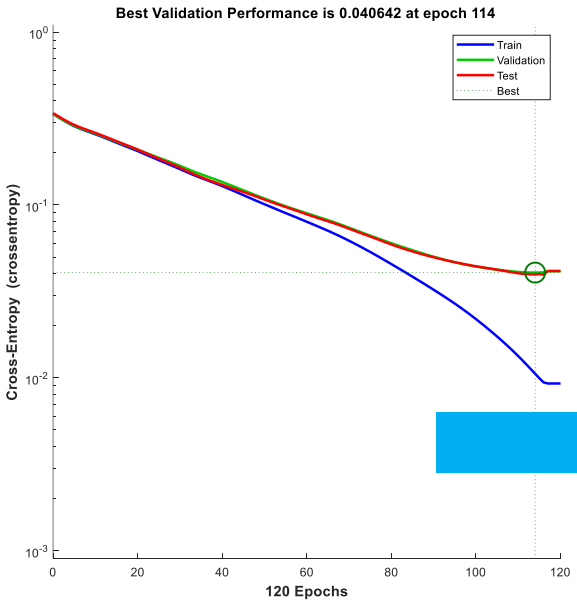
Changing trainFcn and learning rate(LR = 10 vs LR = 0.5)

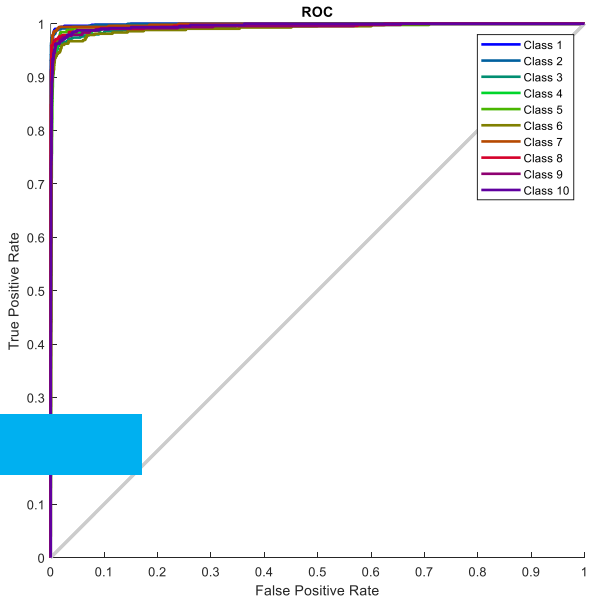






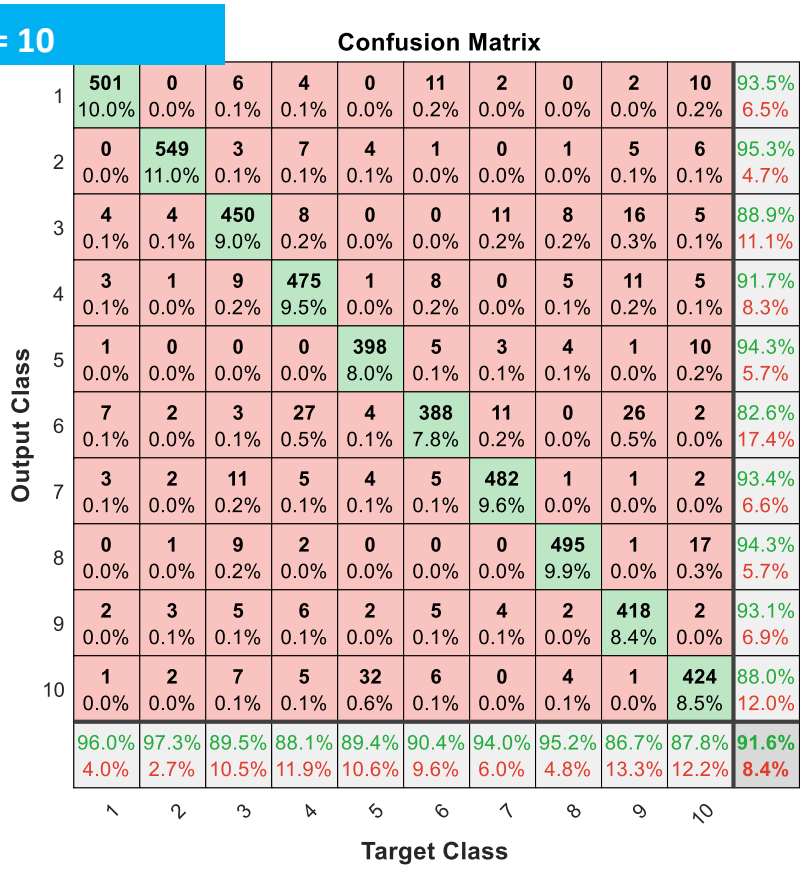
LR = 10



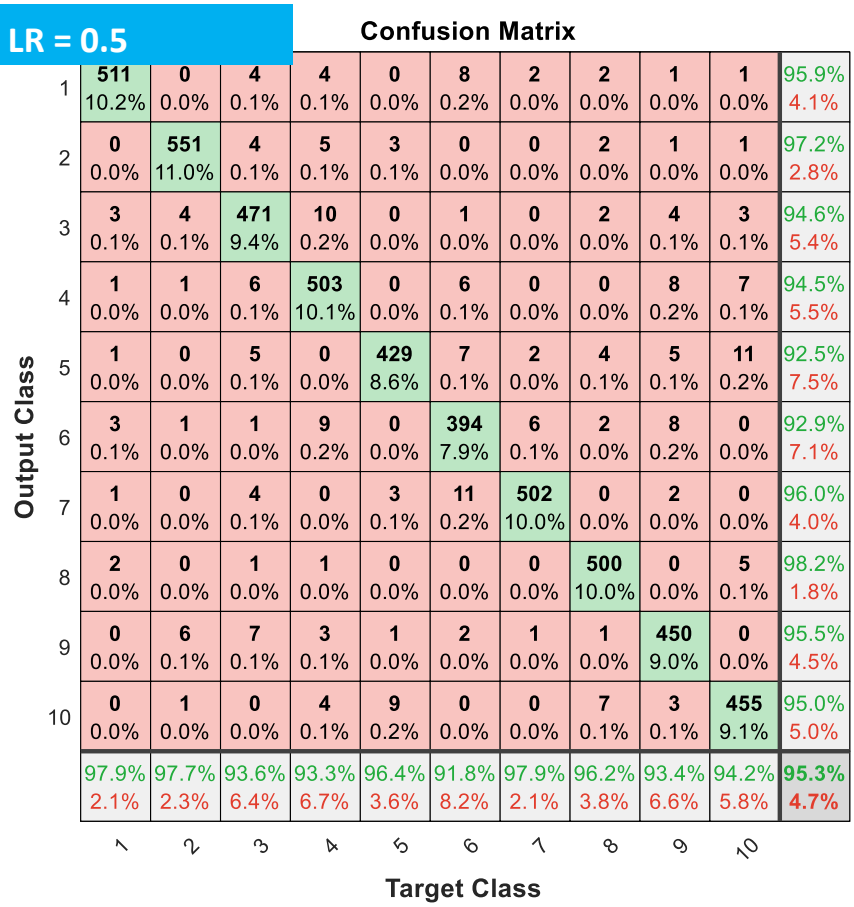


LR = 0.5

LR = 10


Confusion Matrix
Output Class
Target Class
1 501 10.0% 0 0.0% 6 0.1% 4 0.1% 0 0.0% 11 0.2% 2 0.0% 0 0.0% 2 0.0% 10 0.2% 93.5% 6.5%
2 0 0.0% 549 11.0% 3 0.1% 7 0.1% 4 0.1% 1 0.0% 0 0.0% 1 0.0% 5 0.1% 6 0.1% 95.3% 4.7%
3 4 0.1% 4 0.1% 450 9.0% 8 0.2% 0 0.0% 0 0.0% 11 0.2% 8 0.2% 16 0.3% 5 0.1% 88.9% 11.1%
4 3 0.1% 1 0.0% 9 0.2% 475 9.5% 1 0.0% 8 0.2% 0 0.0% 5 0.1% 11 0.2% 5 0.1% 91.7% 8.3%
5 1 0.0% 0 0.0% 0 0.0% 0 0.0% 398 8.0% 5 0.1% 3 0.1% 4 0.1% 1 0.0% 10 0.2% 94.3% 5.7%
6 7 0.1% 2 0.0% 3 0.1% 27 0.5% 4 0.1% 388 7.8% 11 0.2% 0 0.0% 26 0.5% 2 0.0% 82.6% 17.4%
7 3 0.1% 2 0.0% 11 0.2% 5 0.1% 4 0.1% 5 0.1% 482 9.6% 1 0.0% 1 0.0% 2 0.0% 93.4% 6.6%
8 0 0.0% 1 0.0% 9 0.2% 2 0.0% 0 0.0% 0 0.0% 0 0.0% 495 9.9% 1 0.0% 17 0.3% 94.3% 5.7%
9 2 0.0% 3 0.1% 5 0.1% 6 0.1% 2 0.0% 5 0.1% 4 0.1% 2 0.0% 418 8.4% 2 0.0% 93.1% 6.9%
10 1 0.0% 2 0.0% 7 0.1% 5 0.1% 32 0.6% 6 0.1% 0 0.0% 4 0.1% 1 0.0% 424 8.5% 88.0% 12.0%
96.0% 4.0% 97.3% 2.7% 89.5% 10.5% 88.1% 11.9% 89.4% 10.6% 90.4% 9.6% 94.0% 6.0% 95.2% 4.8% 86.7% 13.3% 87.8% 12.2% 91.6% 8.4%
1 2 3 4 5 6 7 8 9 10

LR = 0.5


Confusion Matrix
Output Class
Target Class
1 511 10.2% 0 0.0% 4 0.1% 4 0.1% 0 0.0% 8 0.2% 2 0.0% 2 0.0% 1 0.0% 1 0.0% 95.9% 4.1%
2 0 0.0% 551 11.0% 4 0.1% 5 0.1% 3 0.1% 0 0.0% 0 0.0% 2 0.0% 1 0.0% 1 0.0% 97.2% 2.8%
3 3 0.1% 4 0.1% 471 9.4% 10 0.2% 0 0.0% 1 0.0% 0 0.0% 2 0.0% 4 0.1% 3 0.1% 94.6% 5.4%
4 1 0.0% 1 0.0% 6 0.1% 503 10.1% 0 0.0% 6 0.1% 0 0.0% 0 0.0% 8 0.2% 7 0.1% 94.5% 5.5%
5 1 0.0% 0 0.0% 5 0.1% 0 0.0% 429 8.6% 7 0.1% 2 0.0% 4 0.1% 5 0.1% 11 0.2% 92.5% 7.5%
6 3 0.1% 1 0.0% 1 0.0% 9 0.2% 0 0.0% 394 7.9% 6 0.1% 2 0.0% 8 0.2% 0 0.0% 92.9% 7.1%
7 1 0.0% 0 0.0% 4 0.1% 0 0.0% 3 0.1% 11 0.2% 502 10.0% 0 0.0% 2 0.0% 0 0.0% 96.0% 4.0%
8 2 0.0% 0 0.0% 1 0.0% 1 0.0% 0 0.0% 0 0.0% 0 0.0% 500 10.0% 0 0.0% 5 0.1% 98.2% 1.8%
9 0 0.0% 6 0.1% 7 0.1% 3 0.1% 1 0.0% 2 0.0% 1 0.0% 1 0.0% 450 9.0% 0 0.0% 95.5% 4.5%
10 0 0.0% 1 0.0% 0 0.0% 4 0.1% 9 0.2% 0 0.0% 0 0.0% 7 0.1% 3 0.1% 455 9.1% 95.0% 5.0%
97.9% 2.1% 97.7% 2.3% 93.6% 6.4% 93.3% 6.7% 96.4% 3.6% 91.8% 8.2% 97.9% 2.1% 96.2% 3.8% 93.4% 6.6% 94.2% 5.8% 95.3% 4.7%
1 2 3 4 5 6 7 8 9 10