Faculty of Engineering

Credit Hours Engineering Programs

Mechatronics Engineering and Automation

Academic Year 2019/2020 – Spring 2019

CSE 489

Machine Vision

Project No. (2)

Neural Networks

Date due 27-Apr-19
Date handed in 27-Apr-19

Submitted by:

------------------- **-----------**

-------------------- **-----------**

Contents

1 Problem Defination and importance **1**
2 Methods and algorithms **2**
2.1 Feed Forward net 2
2.2 BackProbagation 3
3 Data set description **3**
4 experimental results and discussion **4**
4.1 the rain and test a single layer neural network 5
4.2 Repeat 4.1 using A hidden layer of 300 neurons 7
4.3 Repeat 4.1 using two hidden layers of 300 neurons 9
4.4 Repeat 4.1 using three hidden layers of 300 neurons 12
4.5 Backpropagation 13
4.6 Conculsion 14
4.7 Summary 15
5 Appendix **17**
5.1 visualize data 17
5.1.1 Visualize Training Dataset 17
5.1.2 Visualize Test Dataset 17
5.2 question Two/three 18
5.2.1 Training 18
5.2.2 Test 19
5.3 Backprobagation 20

List of figures

Figure 1 Feed-Forward Net 3
Figure 2 N Layered-Feed-Forward Neural Network with Back-Propagation Training Algorithm 4
Figure 3 Sample from the Training Dataset 5
Figure 4 Sample from the Test Dataset 5
Figure 5 Back-Propagation Error Plot 16
Figure 6 Output Probability 17

1 PROBLEM DEFINATION AND IMPORTANCE

We need to classify handwritten digits from 0 to 9 to ten different classes. The dataset used is Mnist dataset which is a popular dataset to be used to train the machines for pattern recognition on real-world data. This dataset includes binary images of handwritten digits. The images are pre-processed and ready to be fed to the neural network. The craft ship here is to design the architecture to minimize the error as least as possible.

2 METHODS AND ALGORITHMS

2.1 FEED FORWARD NET

The feedforward neural network was the first and simplest type of artificial neural network devised.In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network

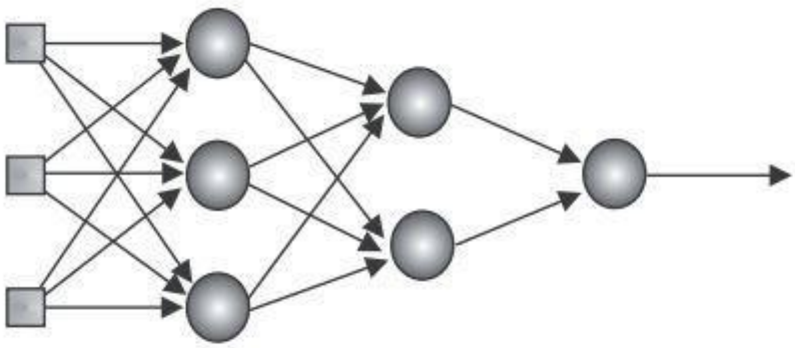


Figure 1 Feed-Forward Net

The output signal y is generated by first computing a weighted sum of the inputswhere w_j are connection weights, x_j are the input signals nd b is the bias term. In order to simplify the notation, the bias term is often written w0x0, where w0 = b and x0 is always 1. Thus, s can then be Expressed as

$$s = \sum_{j=0}^{n} w_j x_j .$$

The neuron output is obtained as

$$y = \sigma\left(\sum_{j=0}^{n} w_j x_j\right)$$

Where sigma is the activation function.

2.2 BACKPROBAGATION

Is a training algorithm consisting of 4 steps:

1. Initialization of weights

2. Feed forward

3. Back propagation of errors

4. Updating the weights and the biases.

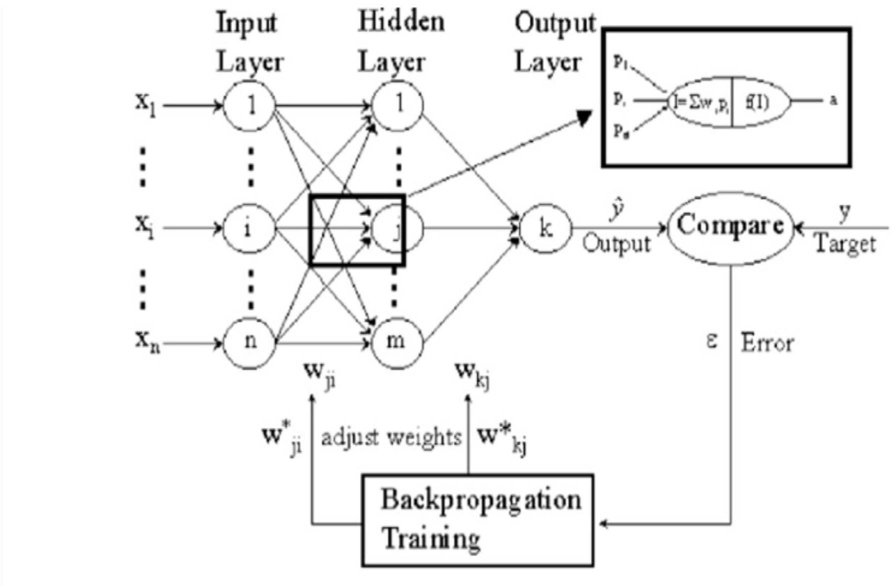


Figure 2 N Layered-Feed-Forward Neural Network with Back-Propagation Training Algorithm

3 DATA SET DESCRIPTION.

The original black and white (bi-level) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

The dataset has a training set of 60,000 examples, and a test set of 10,000 examples

Here is an example of the six images with their corresponding labels

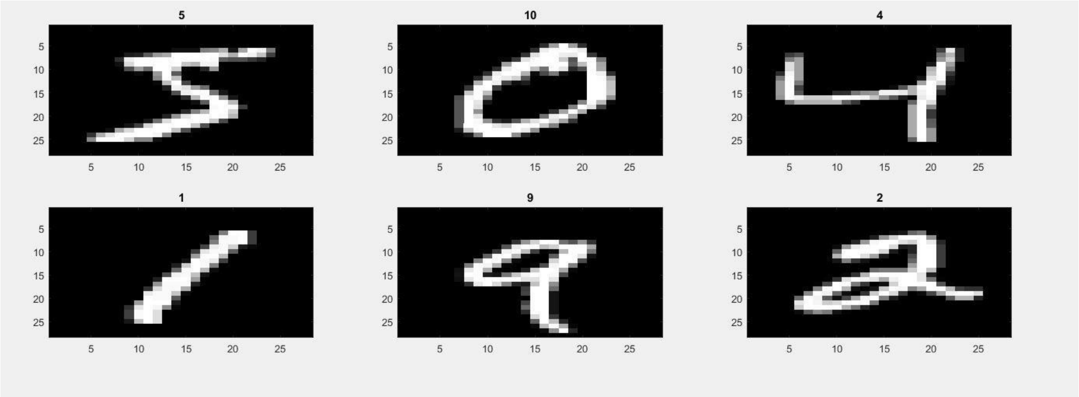


Figure 3 Sample from the Training Dataset

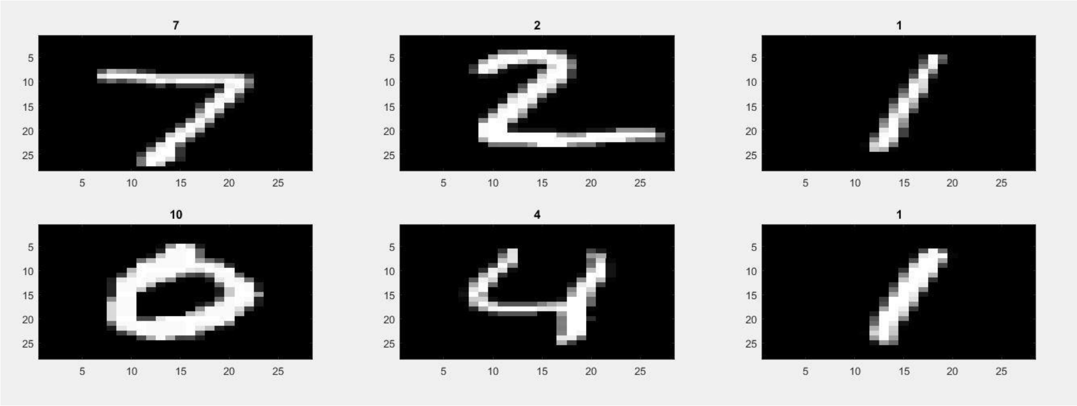
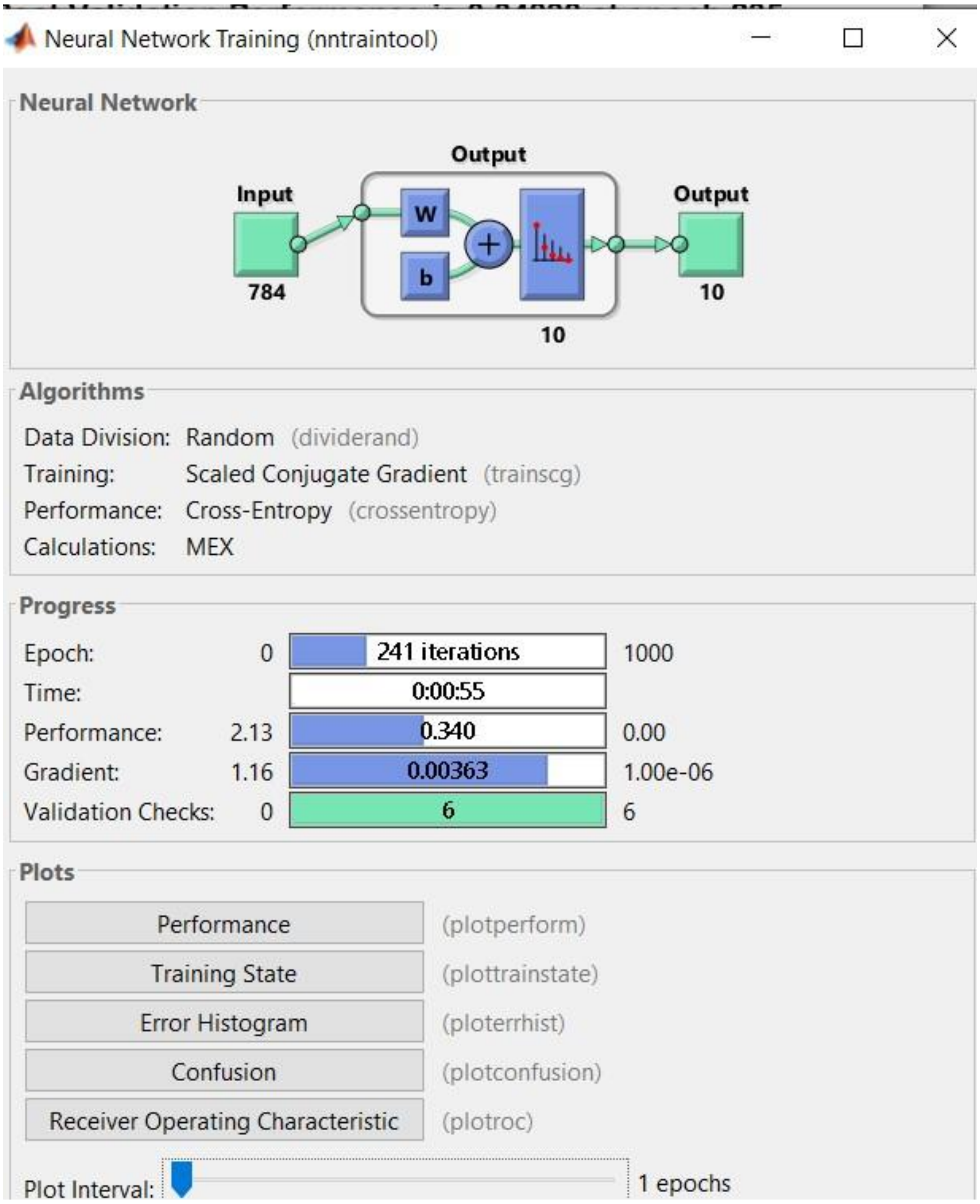


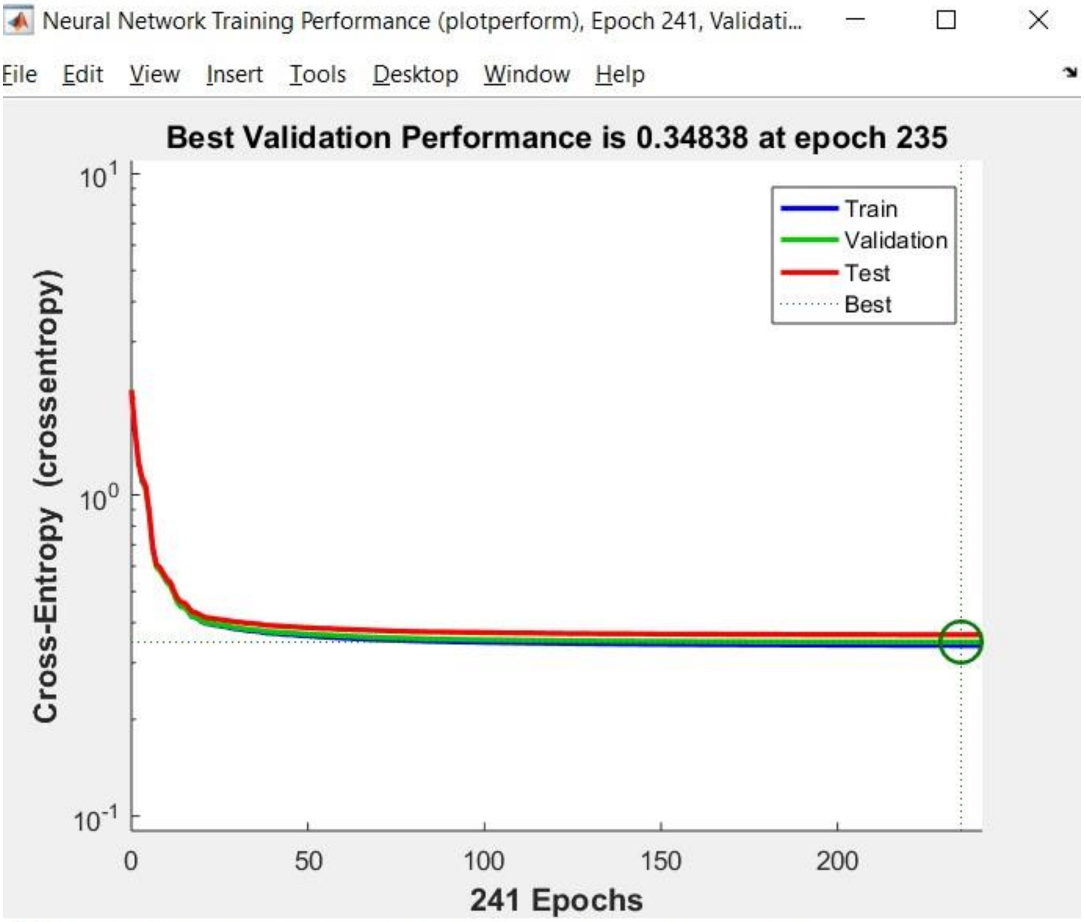
Figure 4 Sample from the Test Dataset

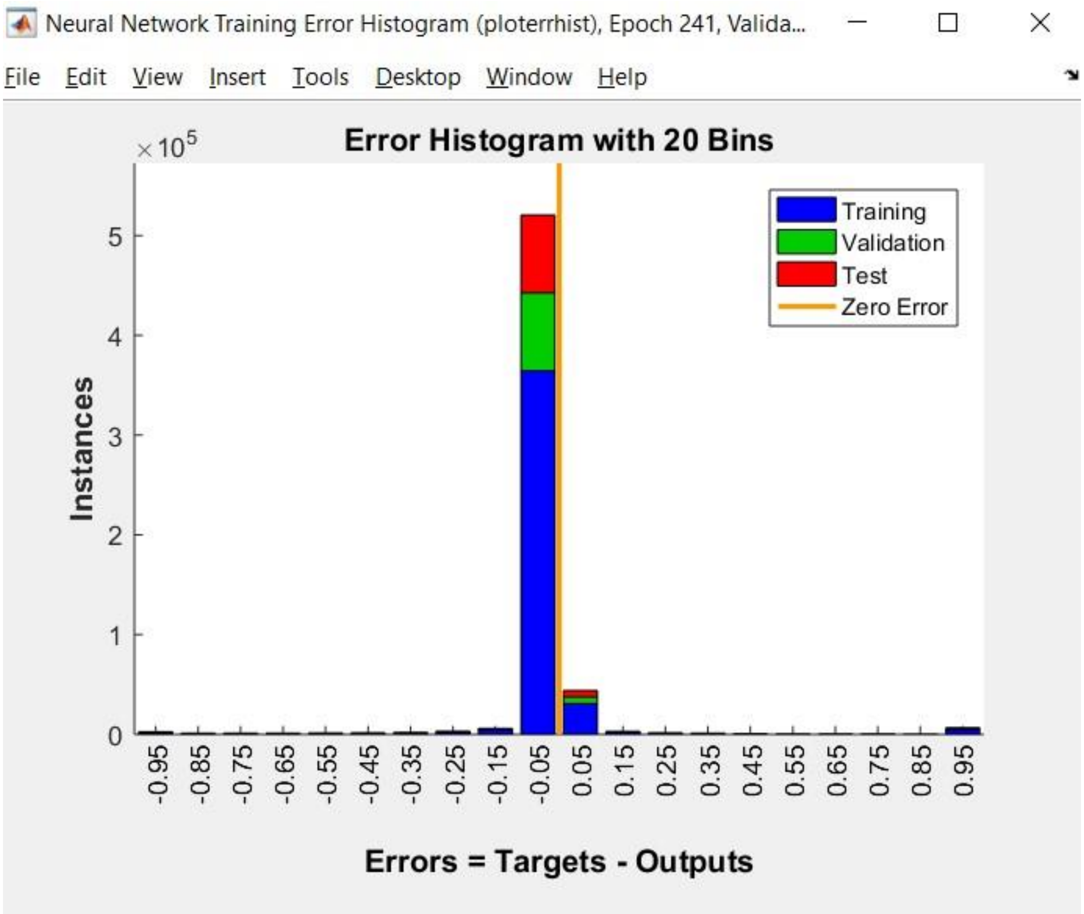
4 EXPERIMENTAL RESULTS AND DISCUSSION

```
Training function: Scaled Conjugate Gradient.
Epoch: The number of input-output pairs that are
presented during the accumulation
```

4.1 THE RAIN AND TEST A SINGLE LAYER NEURAL NETWORK

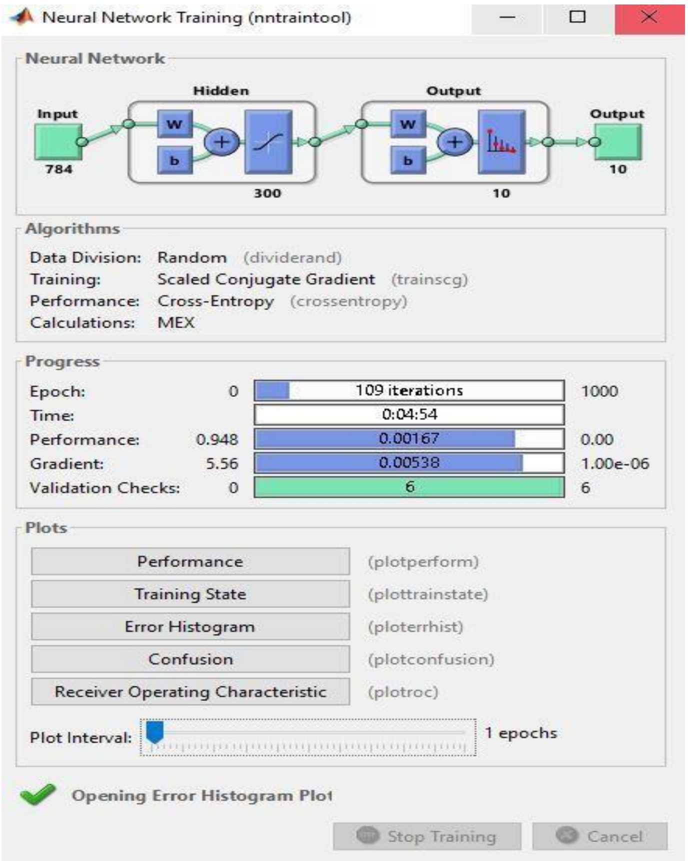


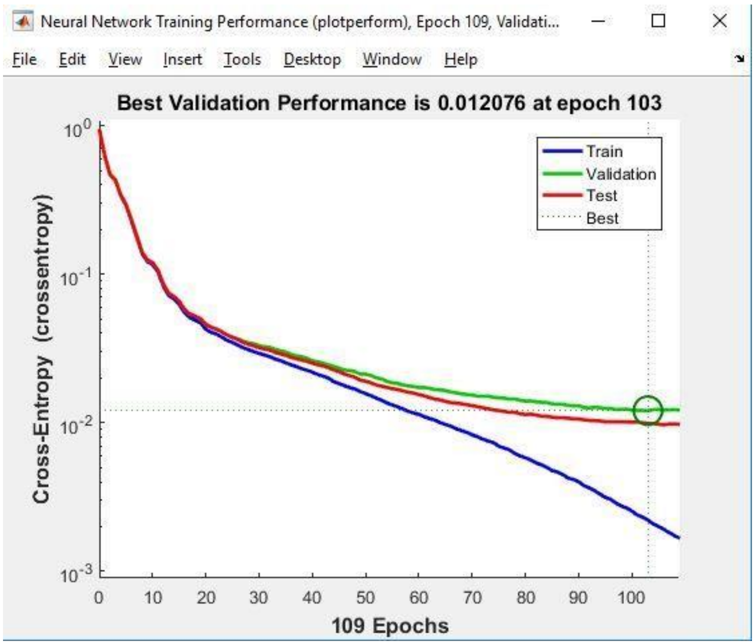

Neural Network Training Performance (plotperform), Epoch 241, Validati...
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 0.34838 at epoch 235
Cross-Entropy (crossentropy)
10^1
10^0
10^-1
0
50
100
150
200
241 Epochs
Train
Validation
Test
Best

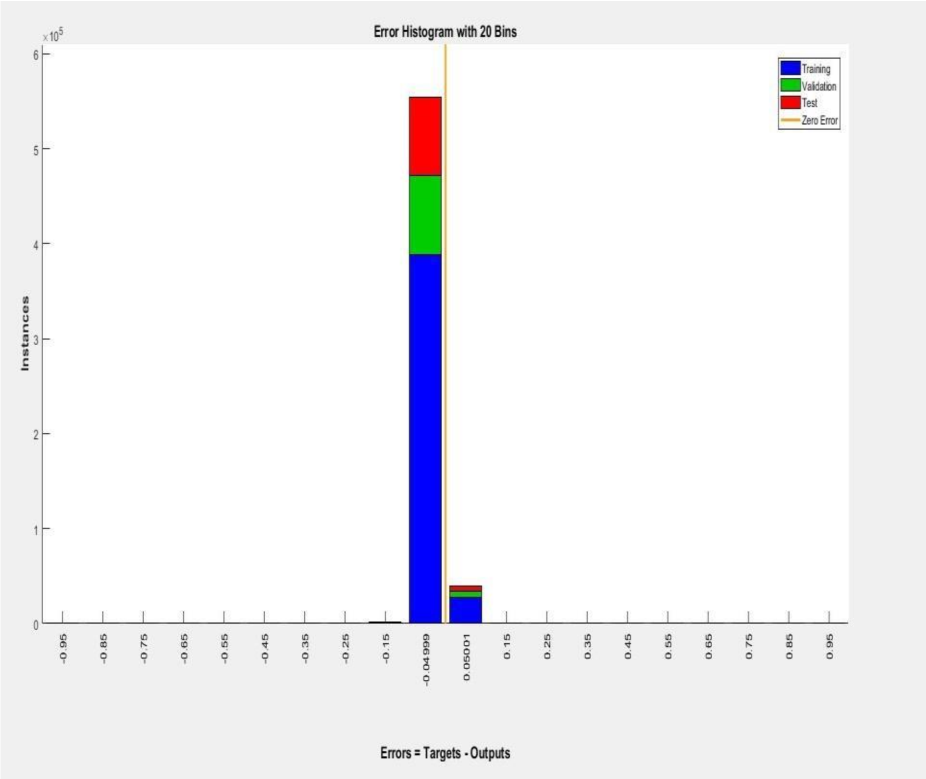

Neural Network Training Error Histogram (ploterrhist), Epoch 241, Valida...
File Edit View Insert Tools Desktop Window Help
Error Histogram with 20 Bins
× 10^5
Instances
0
1
2
3
4
5
-0.95 -0.85 -0.75 -0.65 -0.55 -0.45 -0.35 -0.25 -0.15 -0.05 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
Errors = Targets - Outputs
Training
Validation
Test
Zero Error

```
>> AhmedAbdullah
>> AhmedAbdullahTestData
The accuracy is 85.040000 >>
```

4.2 REPEAT 4.1 USING A HIDDEN LAYER OF 300 NEURONS.



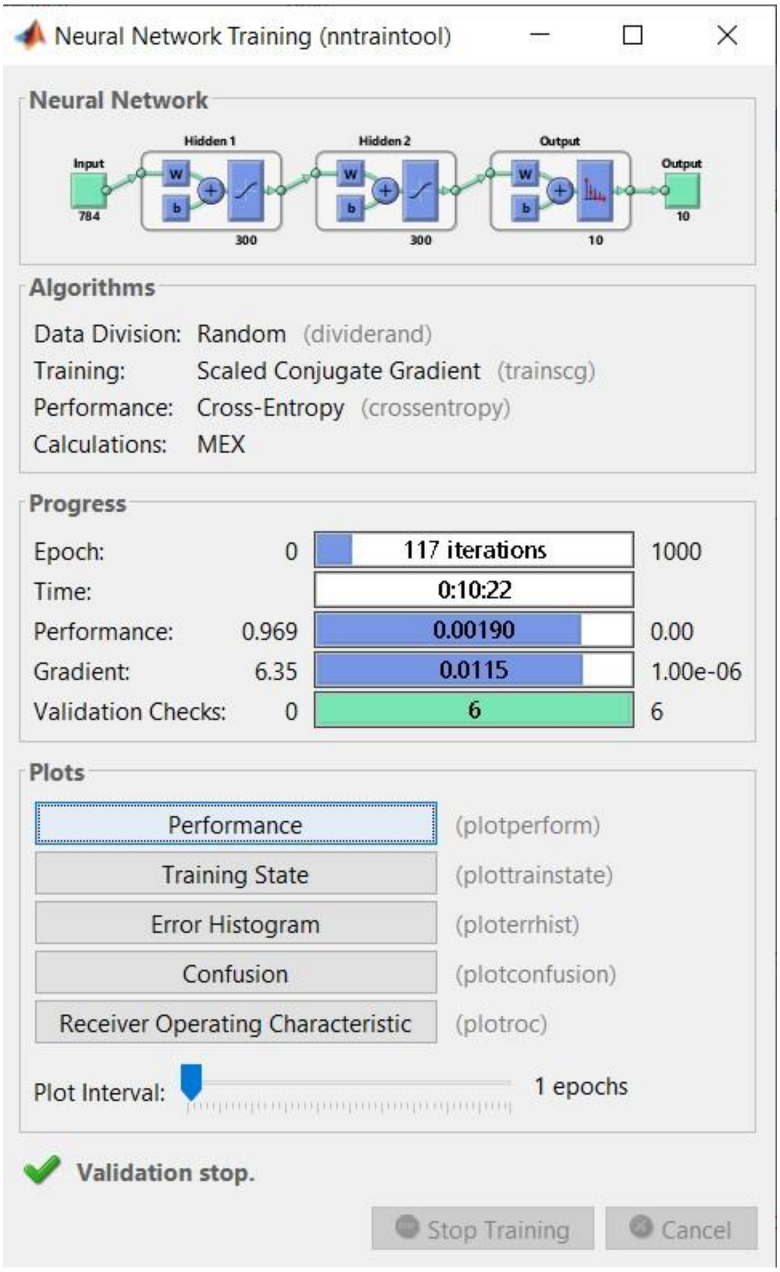


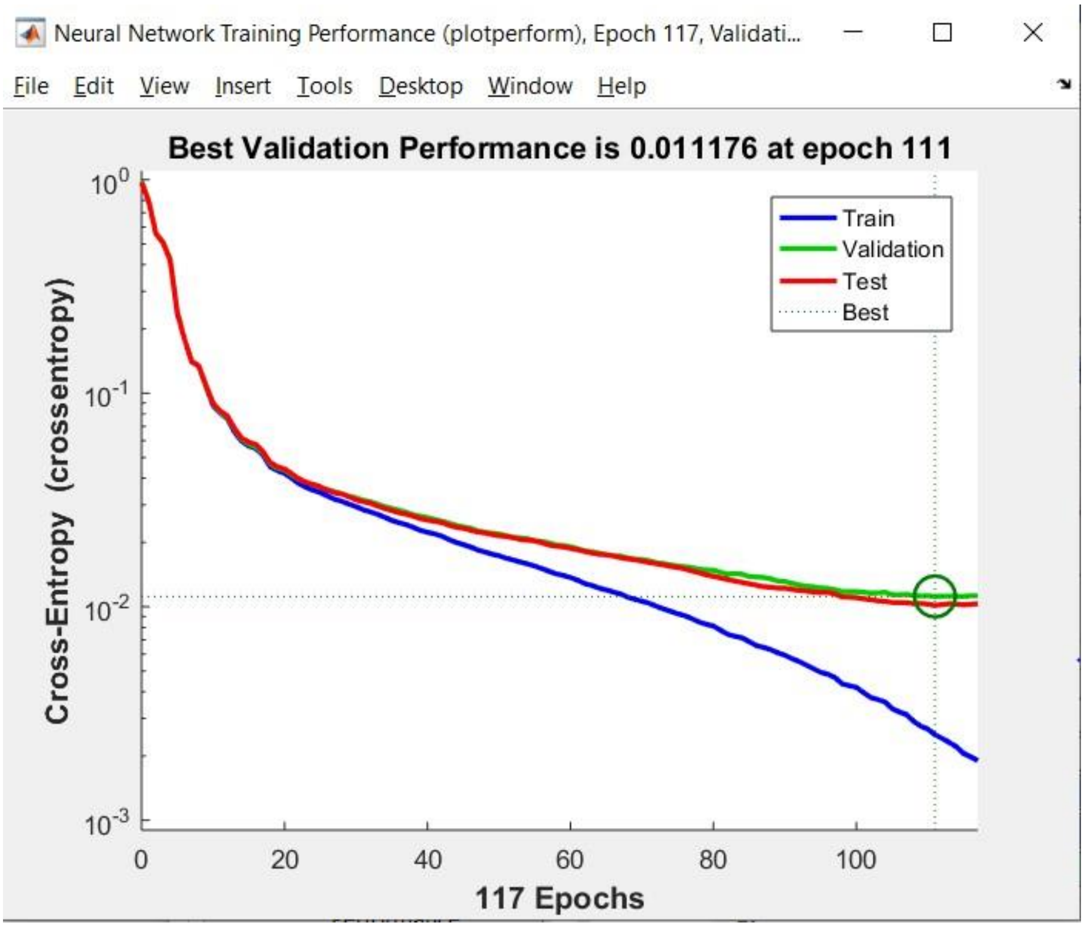


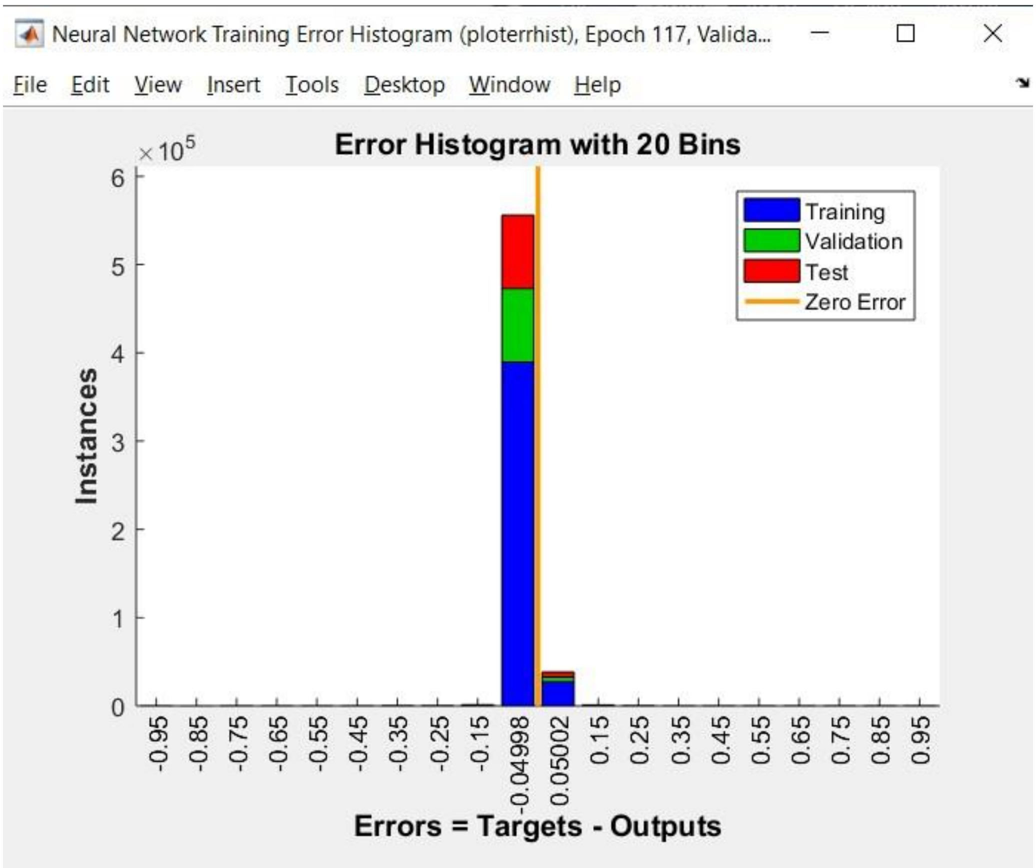
The accuracy is :

```
>> AhmedAbdullah
>> AhmedAbdullahTestData
The accuracy is 97.010000 >>
```

4.3 REPEAT 4.1 USING TWO HIDDEN LAYERS OF 300 NEURONS.

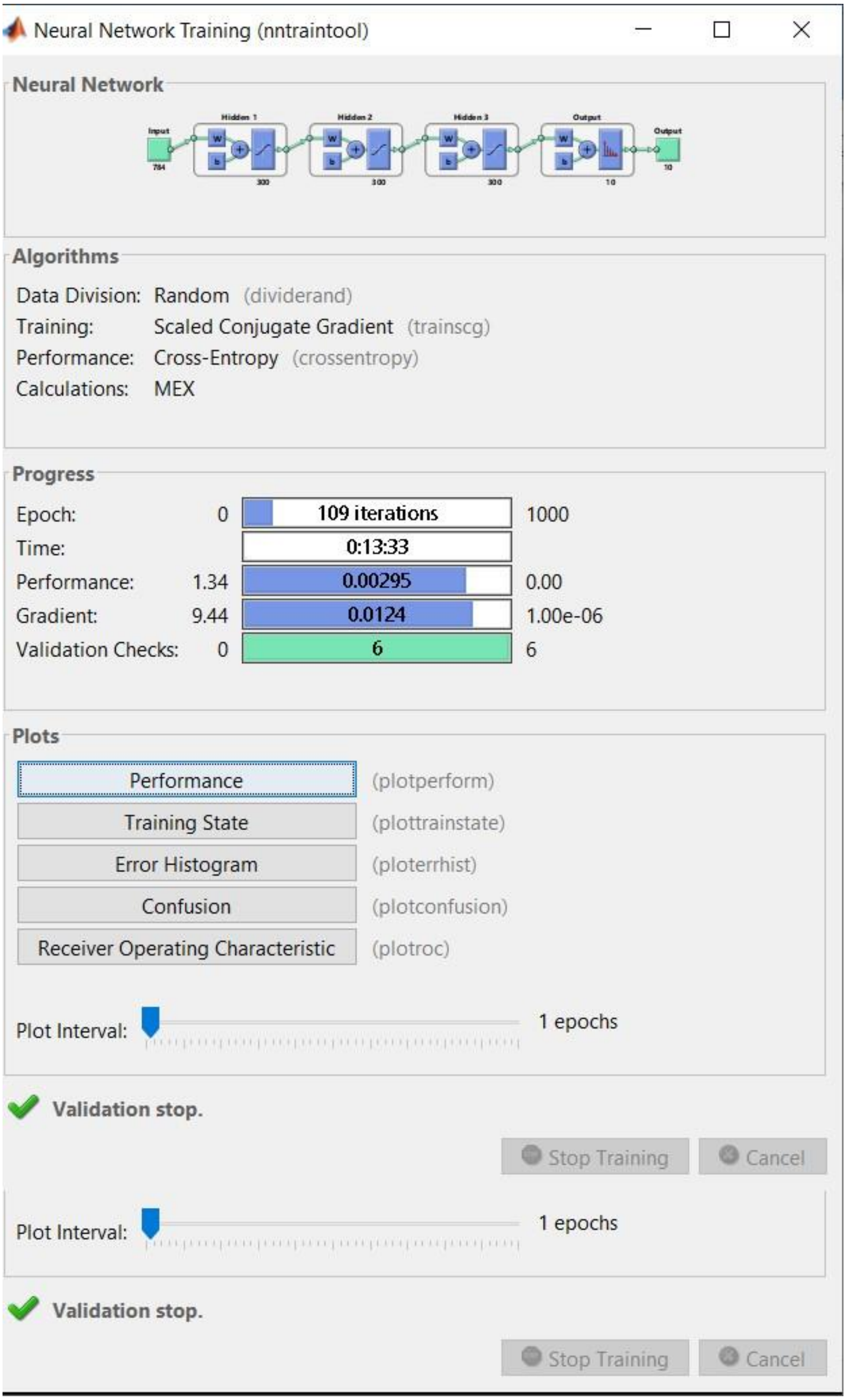


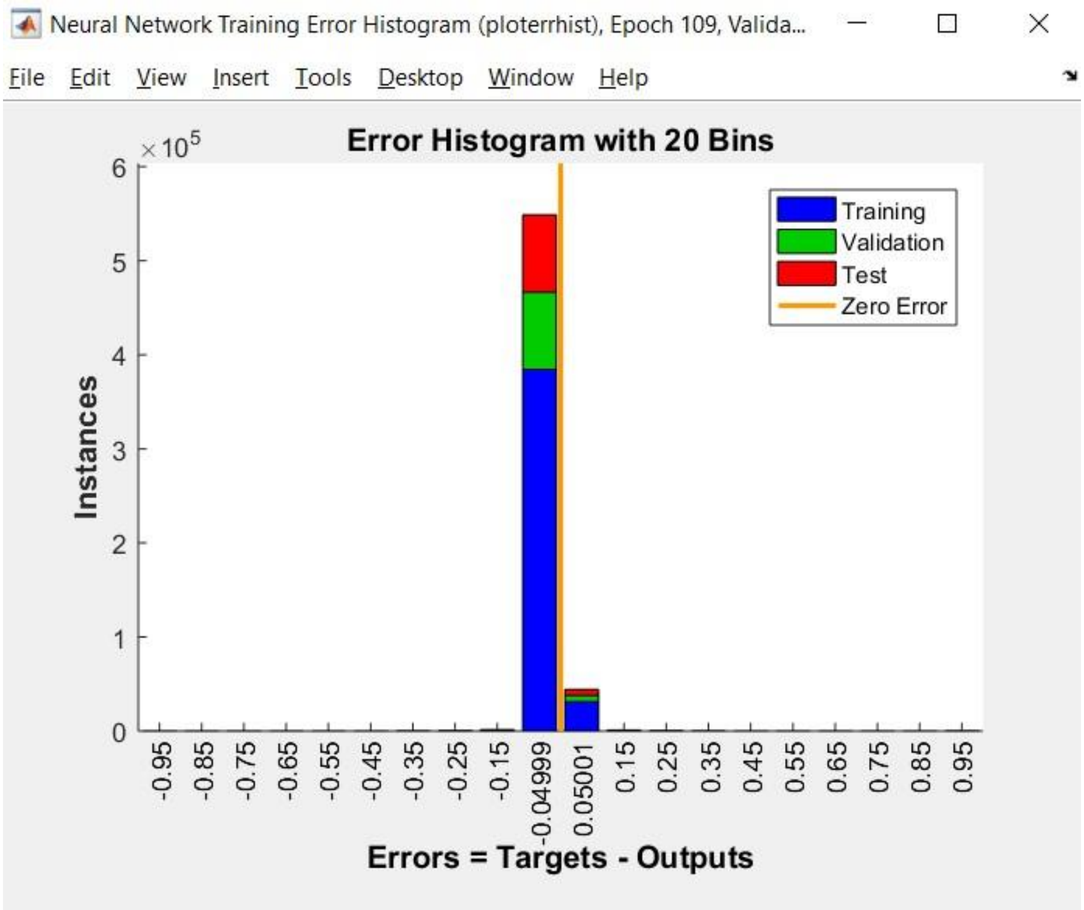

Neural Network Training Performance (plotperform), Epoch 117, Validati...
File Edit View Insert Tools Desktop Window Help
Best Validation Performance is 0.011176 at epoch 111
Train
Validation
Test
Best
Cross-Entropy (crossentropy)
10^{0}
10^{-1}
10^{-2}
10^{-3}
0 20 40 60 80 100
117 Epochs


Neural Network Training Error Histogram (ploterrhist), Epoch 117, Valida...
File Edit View Insert Tools Desktop Window Help
Error Histogram with 20 Bins
$\times 10^{5}$
Training
Validation
Test
Zero Error
Instances
0 1 2 3 4 5 6
-0.95 -0.85 -0.75 -0.65 -0.55 -0.45 -0.35 -0.25 -0.15 -0.04998 0.05002 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95
Errors = Targets - Outputs

```
>> AhmedAbdullah
>> AhmedAbdullahTestData
The accuracy is 96.980000 >>
```

4.4 REPEAT 4.1 USING THREE HIDDEN LAYERS OF 300 NEURONS.






```
>> AhmedAbdullah
>> AhmedAbdullahTestData
The accuracy is 96.910000 >>
```

4.5 BACKPROPAGATION

.

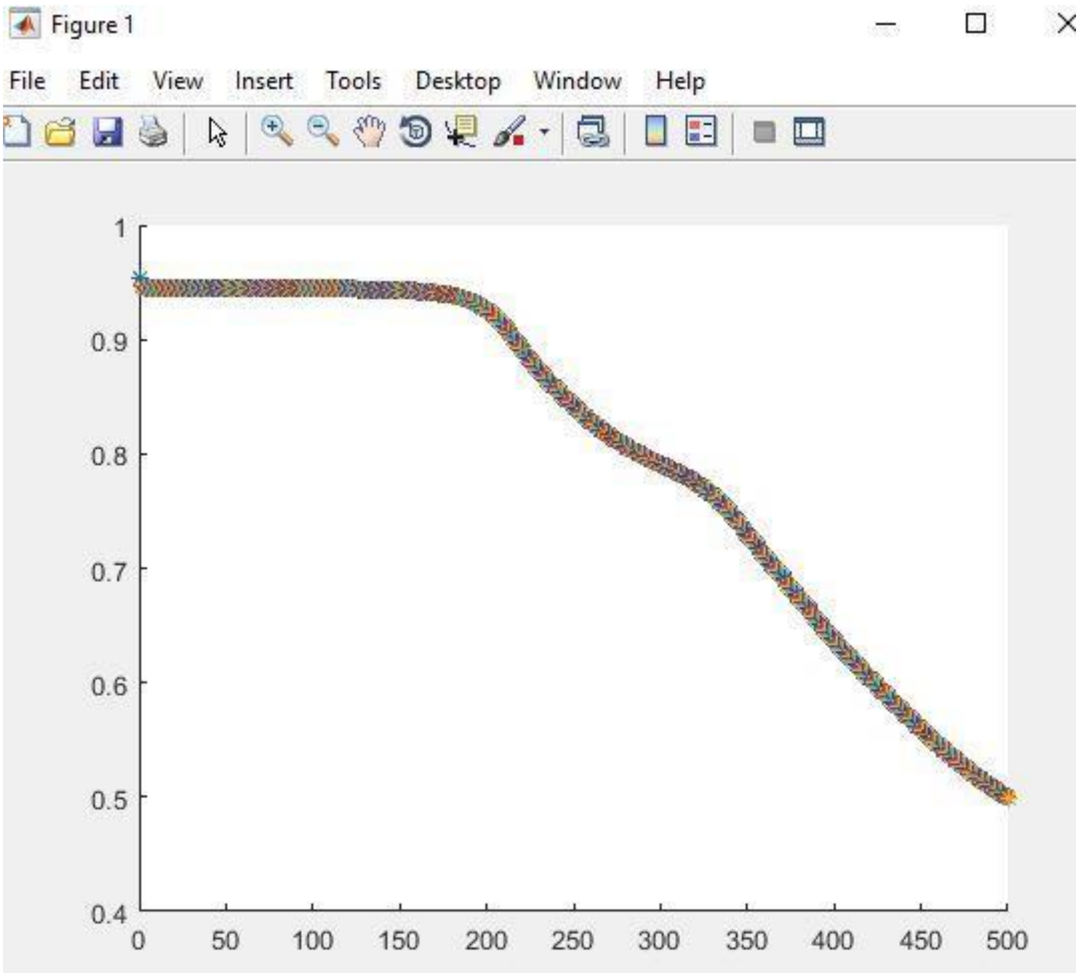
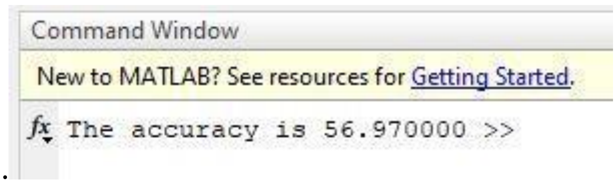


Figure 5 Back-Propagation Error Plot



.

4.6 CONCULSION

The choice of the sigmoid function because we wanted the output results range from 0 to one. The index with the highest probability gives is believed that it gives the correct prediction.

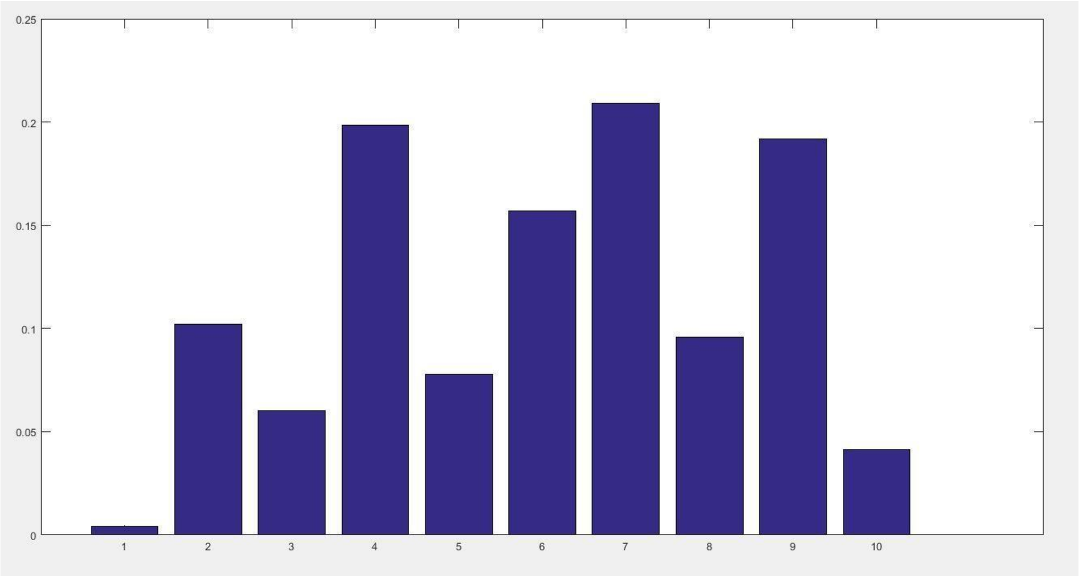


Figure 6 Output Probability

In the above image the correct output is “4”.However, the network classified it “7” because the probability of “7” is slightly higher than “4”

The patternet Feed-Forward Matlab function gives a more optimized output than Feed-Forward Back-Propagation algorithm implemented here.

The increase of number of hidden layers does not grantees an increase in the network accuracy. It also may decrease the network accuracy and increases the computation time needed for training dramatically.

The network with zero hidden layers show the least accuracy

4.7 SUMMARY

Architecture	Epoch	Time	Accuracy

No Hidden Layer	241	00:00:55	85.04
1 Hidden Layer	109	0:04:54	97.01%
2 Hidden Layer	117	0:10:22	96.98%
3 Hidden Layer	109	0:13:13	96.91%
Feed Forward Back Propagation	500	-	56.97%

5 APPENDIX

5.1 VISUALIZE DATA

5.1.1 Visualize Training Dataset

```
images = loadMNISTImages('train-
images.idx3ubyte'); % initialize figure
labels = loadMNISTLabels('train-
labels.idx1ubyte'); % initialize figure
labels = labels';
% transpose
labels(labels==0)=10;
% dummyvar function doesn´t take zeroes
dumVar=dummyvar(labels);
%
figure                                            %
initialize          figure          colormap(gray)
%    set    to    grayscale    for    i    =    1:6
% preview first 36 samples        subplot(3,3,i)
% plot them in 6 x 6 grid
    digit = reshape(images(:, i), [28,28]);      %
row  =  28  x  28  image            imagesc(digit)
% show the image
    title(num2str(labels(i)))                     %
show the label end
```

5.1.2 Visualize Test Dataset

```
test_images = loadMNISTImages('t10k-
images.idx3ubyte'); test_labels =
loadMNISTLabels('t10k-labels.idx1ubyte')';
test_labels = test_labels';
% transpose
test_labels(test_labels==0)=10;
% dummyvar function doesn´t take zeroes
dumVar=dummyvar(test_labels);
%
figure                                            %
initialize          figure          colormap(gray)
%    set    to    grayscale      for    i    =    1:6
% preview first 36 samples        subplot(3,3,i)
% plot them in 6 x 6 grid
     digit = reshape(test_images(:, i), [28,28]);
row = 28 x 28 image      imagesc(digit)
% show the image
```

```
     title(num2str(test_labels(i)))
show the label end
```

5.2 QUESTION TWO/THREE

5.2.1 Training

```
images = loadMNISTImages('train-
images.idx3ubyte'); % initialize figure
labels = loadMNISTLabels('train-
labels.idx1ubyte'); % initialize figure
labels = labels';
% transpose
labels(labels==0)=10;
% dummyvar function doesn´t take zeroes
dumVar=dummyvar(labels);
%
% figure                                                    %
initialize figure
% colormap(gray)                                            % set
to grayscale
% for i = 1:6                                              %
preview first 36 samples
%     subplot(3,3,i)                                        %
plot them in 6 x 6 grid
%     digit = reshape(images(:, i), [28,28]);      % row
= 28 x 28 image
%     imagesc(digit)                                        %
show the image
%     title(num2str(labels(i)))                             %
show the label
% end x = images; t = dumVar'; trainFcn =
'trainscg';                                     % use scaled
conjugate gradient for training

hiddenLayerSize = [300 300];/[300]Q3a%[300 300
300]Q3b []Q2                            net =
patternnet(hiddenLayerSize);                    %
create Pattern Recognition Network =patternnet
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
net.performFcn = 'crossentropy';
```

```
[net,tr] = train(net,x,t);
```

5.2.2 Test

```
%load('23AprilTrialNumber01')
test_images = loadMNISTImages('t10k-
images.idx3ubyte'); test_labels =
loadMNISTLabels('t10k-labels.idx1ubyte')';
test_labels = test_labels';
% transpose
test_labels(test_labels==0)=10;
% dummyvar function doesn´t take zeroes
dumVar=dummyvar(test_labels);
%
figure                                             %
initialize figure
colormap(gray)                                     %
set to grayscale
% for i = 1:6                                       %
preview first 36 samples
%     subplot(3,3,i)                                 %
plot them in 6 x 6 grid
%     digit = reshape(test_images(:, i), [28,28]);
% row = 28 x 28 image
%     imagesc(digit)                                 %
show the image
%     title(num2str(test_labels(i)))
% show the label
% end figure
bar(net(test_images(:,2)))
matching=uint8(zeros(10000,1));
Index=0; j_max=1;
j_max_array=uint8(zeros(10000,1));
Max_array=double(zeros(10000,1));
counter=0; for i =1:10000
    Output_matrix=net(test_images(:,i));
Max=min(Output_matrix);      for j=1:10
if(Output_matrix(j)>Max)
Max=Output_matrix(j);
j_max=j;                              end
j_max_array(i)=j_max;
Max_array(i)=Max;      end
    %    [M,Index]=max(Output_matrix);
if (j_max == test_labels(i))
matching(i)=99;
```

```
counter=counter+1;      else
matching(i)=88;      end end
fprintf('The accuracy is %f %\n',
(counter/10000)*100);
```

5.3 BACKPROBAGATION

```
function [ Output ] = dSigmoid( x )
Output = sigmoid(x).*(1 - sigmoid(x)); end
function [ Output ] = sigmoid( x )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
    Output = 1./(1 + exp(-x));

end
fun
cti
on
[we
igh
ts_
v,
wei
ght
s_w
] =
Tra
ini
ng(
act
iva
tio
nFu
nct
ion
,
dAc
tiv
ati
onF
unc
tio
n,
num
ber
ofN
eur
ons
,
ima
ges
,
Tar
get
,
Max
_Ep
och
s,
```

```
bat
chS
ize
,
LEA
RNI
NGR
ATE
)

n=zeros(batchSize);
trainingSetSize = size(images, 2); %60k
input = size(images, 1);%784 numberofClasses
= size(Target, 1);%10 classes

weights_v=rand(numberofNeurons,input);%%vij(784x300)
weights_w=rand(numberofClasses,numberofNeurons);%wjk(10,300)

weights_v=weights_v./size(weights_v,2);
weights_w=weights_w./size(weights_w,2);
figure; hold on; for t=1:Max_Epochs
for k=1:batchSize n(k)=k;
        X=images(:,n(k));
        A=weights_v*X;
        Act_Y=activationFunction(A);%activation A result
        Z=weights_w*Act_Y;
        Act_Output=activationFunction(Z);%activation Z result
targetVector=Target(:,n(k));

        dOutput=dActivationFunction(Z).*(Act_Output-targetVector);
weights_w=weights_w - LEARNINGRATE.*dOutput*Act_Y';

        dHidden=dActivationFunction(A).*(weights_w'*dOutput);
weights_v=weights_v - LEARNINGRATE.*dHidden*X';     end end;

end
```

main

```
close all, clear all, clc; images = loadMNISTImages('train-
images.idx3-ubyte'); % initialize figure
labels = loadMNISTLabels('train-labels.idx1-ubyte'); % initialize
figure labels=labels';
labels(labels==0)=10;                                % dummyvar
function doesn´t take zeroes dumVar=dummyvar(labels);
% 1. Initialization of weights
% Should be something small in order to not overshoot the goal.
LEARNINGRATE = .01;
Max_Epochs=500; batchSize
= 100;
  rng(1);
numberofN
eurons=30
0;
numberofC
lasses=10
;
```

```
input=784
;
Target=double(zeros(numberofClasses,length(labels)));
% Initialization stage: for i= 1:length(labels)
for j=1:numberofClasses          if( j == labels(i))
Target(j,i)= 1;          end      end end
activationFunction = @sigmoid; dActivationFunction
= @dSigmoid;
[weights_v, weights_w,~] = Training(activationFunction,
dActivationFunction, numberofNeurons, images, Target, Max_Epochs,
batchSize, LEARNINGRATE);

test_images = loadMNISTImages('t10k-images.idx3-ubyte');
test_labels = loadMNISTLabels('t10k-labels.idx1-ubyte')';
test_labels = test_labels';                                  %
transpose
test_labels(test_labels==0)=10;                              %
dummyvar function doesn´t take zeroes
dumVar=dummyvar(test_labels);

matching=uint8(zeros(10000,1));
Index=0; j_max=1;
j_max_array=uint8(zeros(10000,1));
Max_array=double(zeros(10000,1));
counter=0; for i =1:10000
    Input_Vector=test_images(:,i);

Output_Vector=sigmoid(weights_w*sigmoid(weights_v*Input_Vector));
    Max=min(Output_Vector);
for j=1:10
        if(Output_Vector(j)>Max)
Max=Output_Vector(j);             j_max=j;

end
        j_max_array(i)=j_max;
Max_array(i)=Max;      end
    %    [M,Index]=max(Output_matrix);      if (j_max ==
test_labels(i))         matching(i)=99;
counter=counter+1;      else          matching(i)=88;
end end fprintf('The accuracy is %f %\n',
(counter/10000)*100);
```