Swaraj Wankhade

CS795:

Assignment 4

Analysis:

Steps involved for neural network training

1. Process input data

   (Files included)

2. Process targets

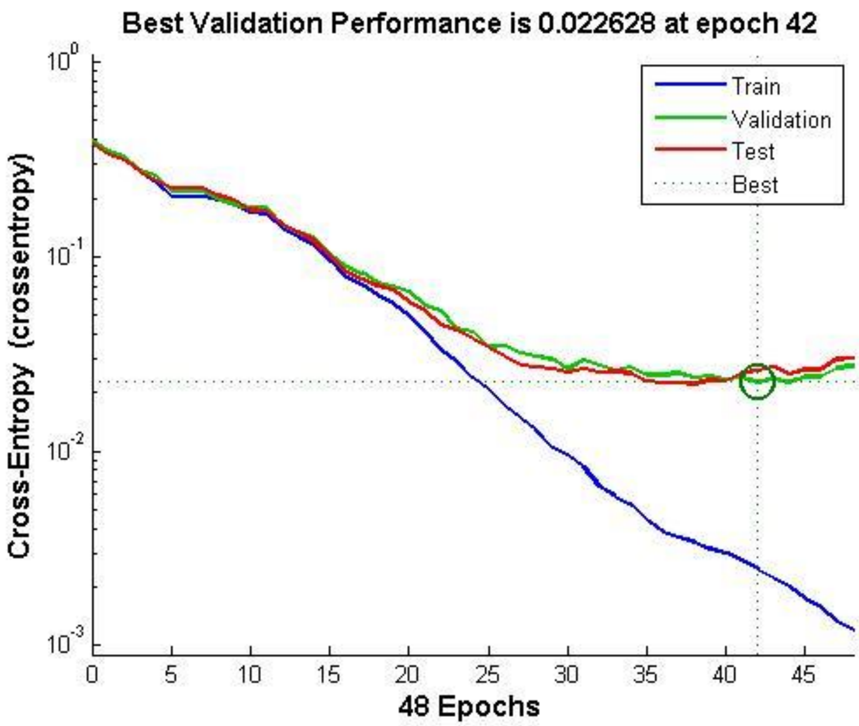
   (Files included)

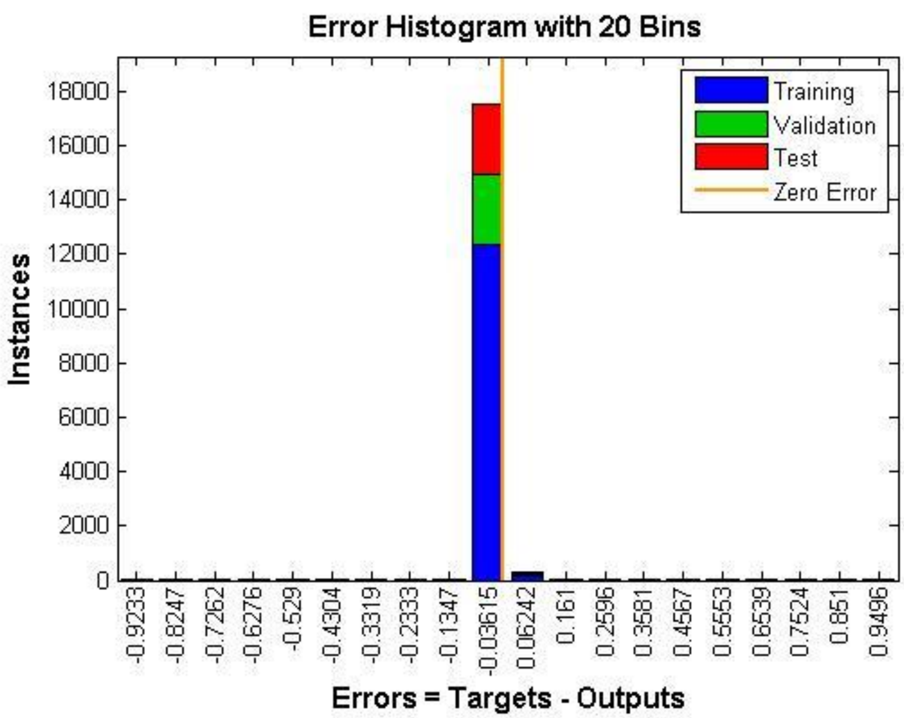
3. Using input and target files , train neural network to get maximum performance

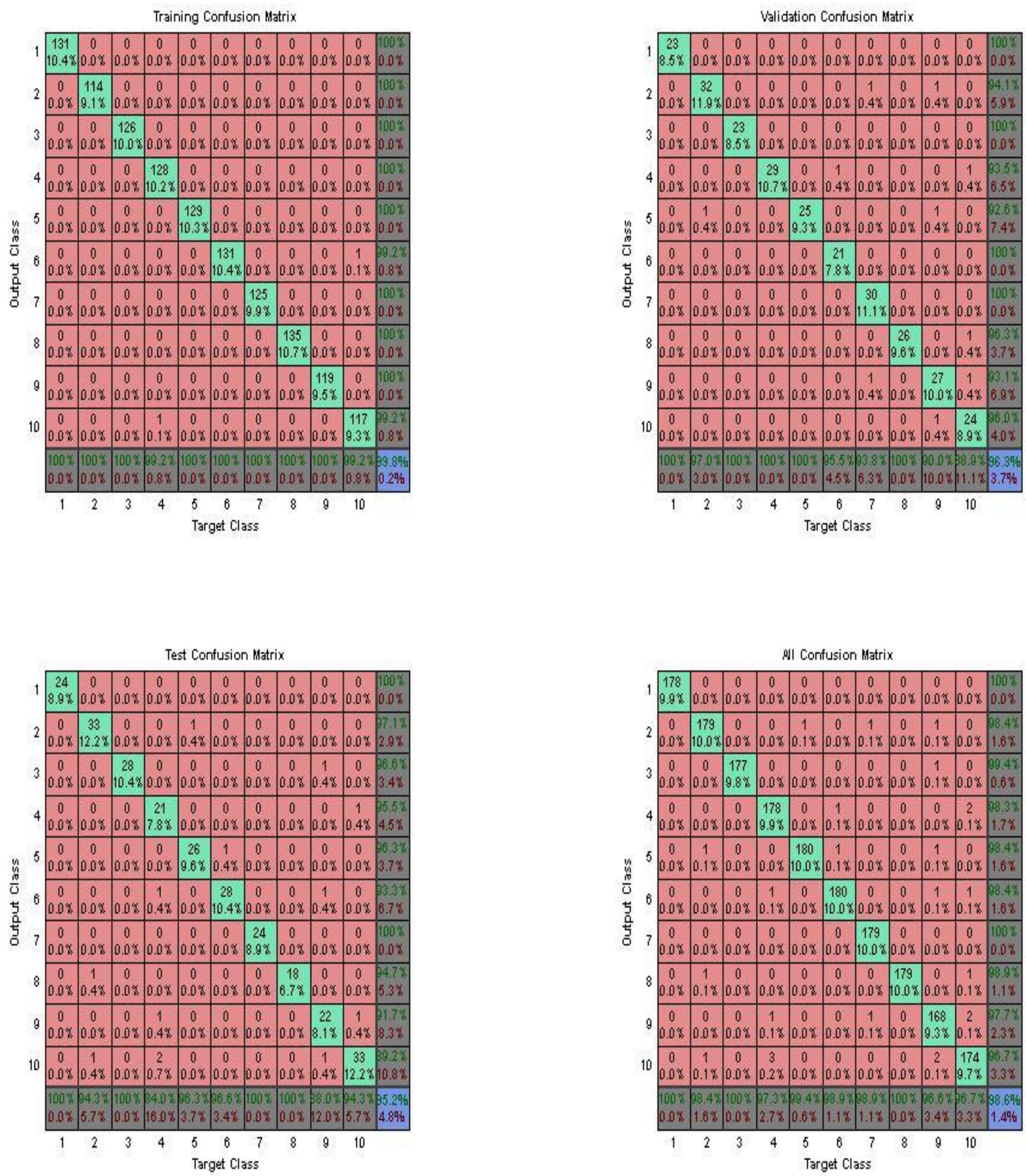
   Analysis:
   10 nodes, 100 nodes and 500 nodes were used for training. As the number of nodes increases the overall accuracy also increases from 10 to 100, but after 100, for 500 it seems to fall down. For training with 500 nodes maximum gradient is reached. The accuracy from the confusion matrix is more than 95% in all the three cases.

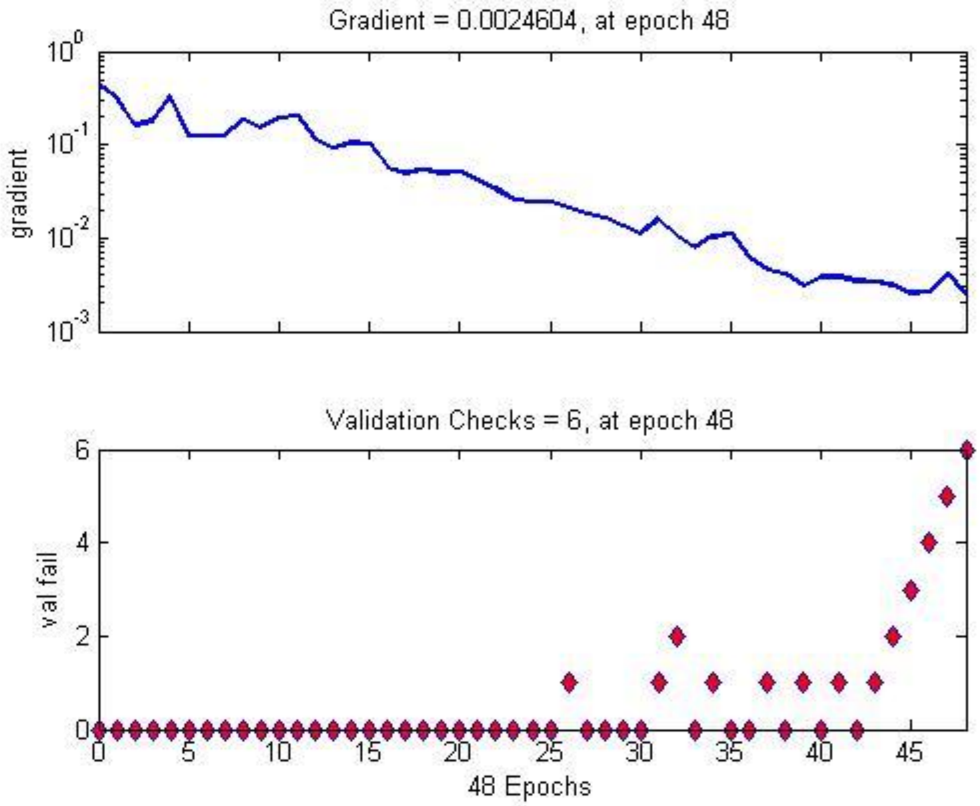
   From confusion Matrix:

   We have
   For 10 Nodes: 95.2
   For 50 Nodes: 96.7
   For 100 Nodes: 98.5
   For 500 Nodes: 97.0

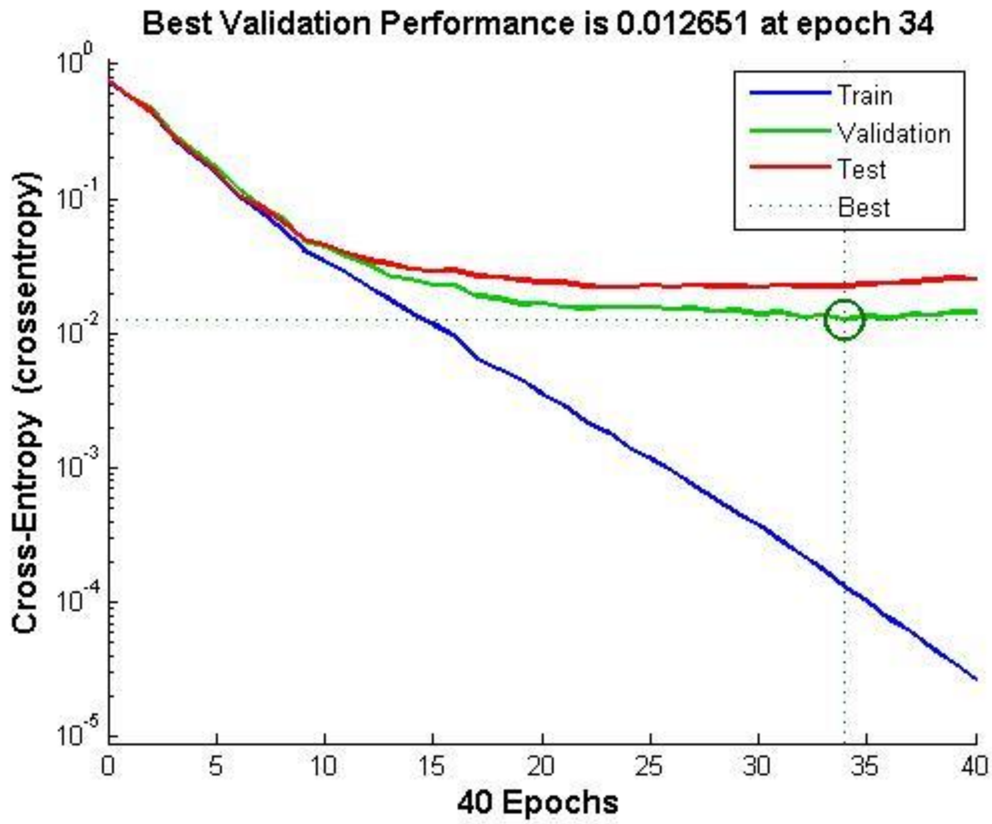

Best Validation Performance is 0.022628 at epoch 42
Cross-Entropy (crossentropy)
10^0
10^-1
10^-2
10^-3
0
5
10
15
20
25
30
35
40
45
48 Epochs
Train
Validation
Test
Best


Error Histogram with 20 Bins
Instances
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
-0.9233
-0.8247
-0.7262
-0.6276
-0.529
-0.4304
-0.3319
-0.2333
-0.1347
-0.03615
0.06242
0.161
0.2596
0.3581
0.4567
0.5553
0.6539
0.7524
0.851
0.9496
Errors = Targets - Outputs
Training
Validation
Test
Zero Error

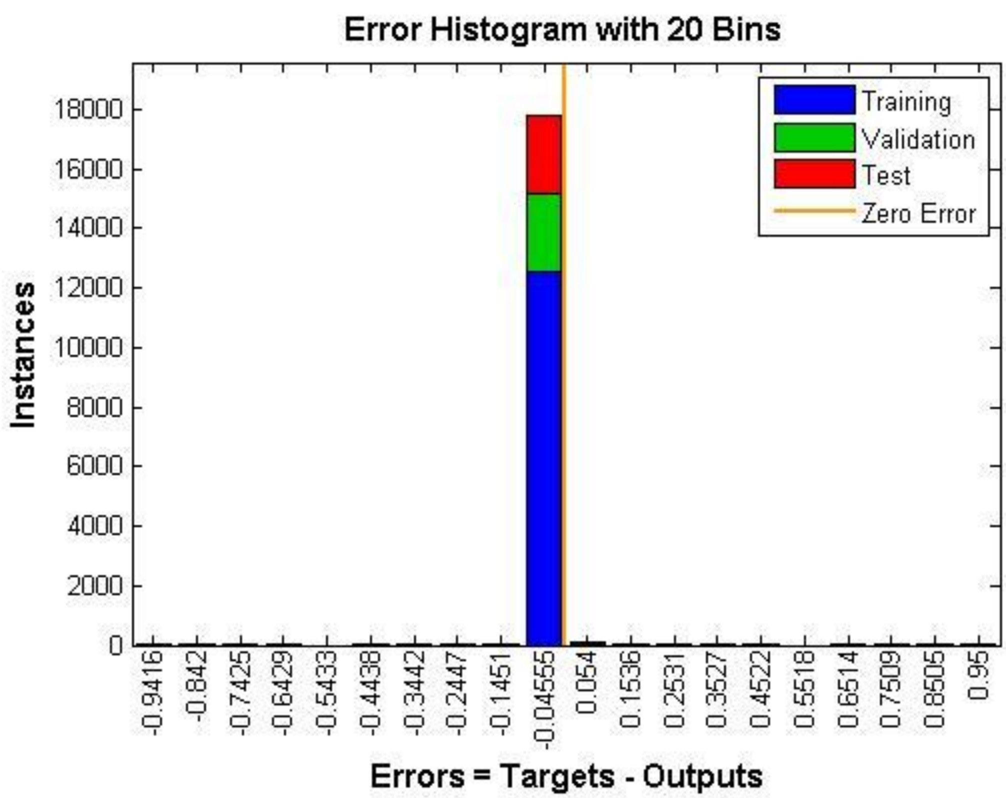

Training Confusion Matrix
Validation Confusion Matrix
Test Confusion Matrix
All Confusion Matrix
Output Class
Target Class


Gradient = 0.0024604, at epoch 48
gradient
10^0
10^{-1}
10^{-2}
10^{-3}
Validation Checks = 6, at epoch 48
val fail
6
4
2
0
0
5
10
15
20
25
30
35
40
45
48 Epochs

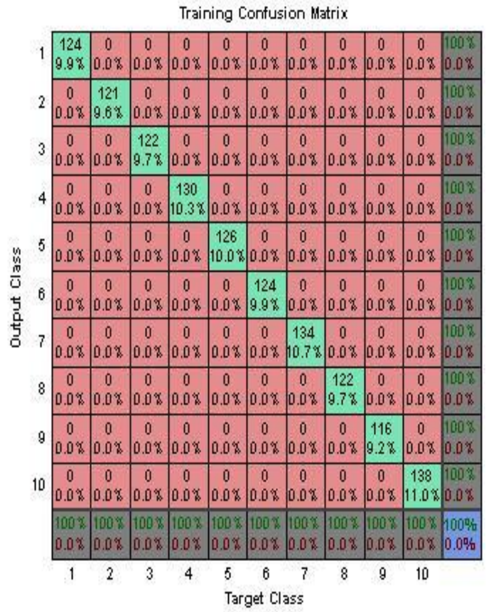
For 50 nodes:

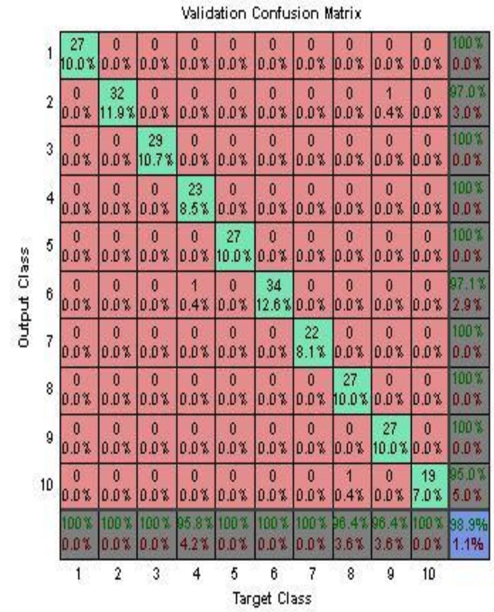


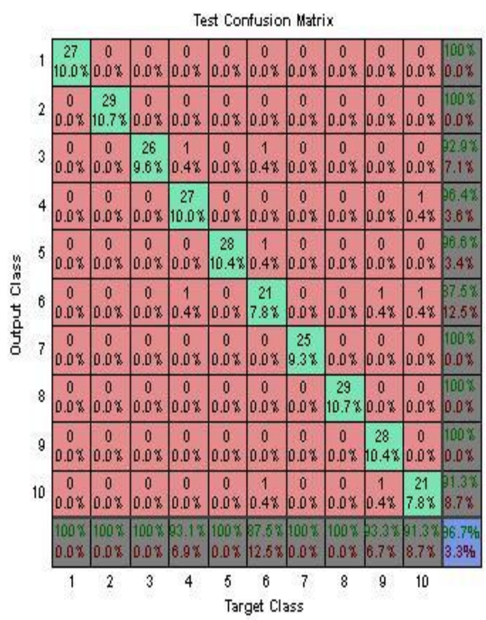
Best Validation Performance is 0.012651 at epoch 34
Cross-Entropy (crossentropy)
40 Epochs
Train
Validation
Test
Best

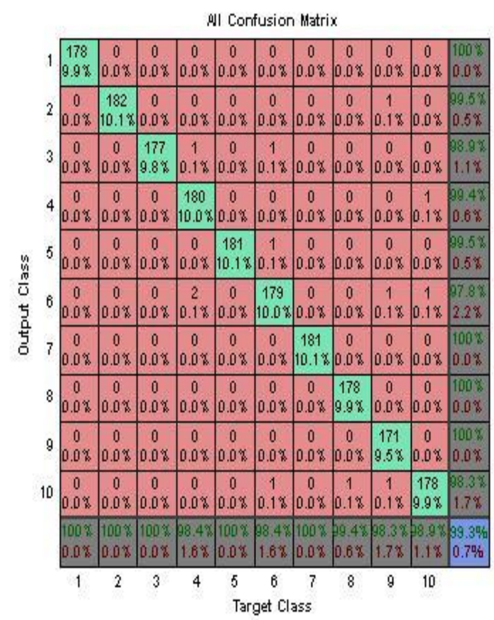


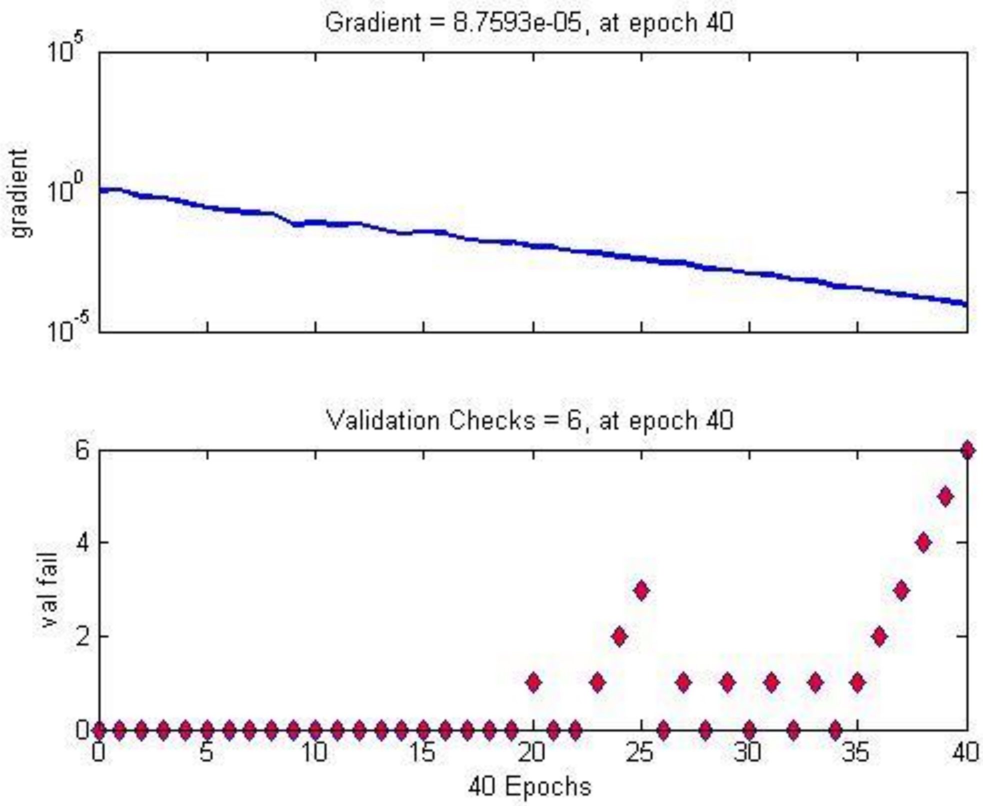
Error Histogram with 20 Bins
Instances
Errors = Targets - Outputs
Training
Validation
Test
Zero Error


Training Confusion Matrix
Output Class
Target Class

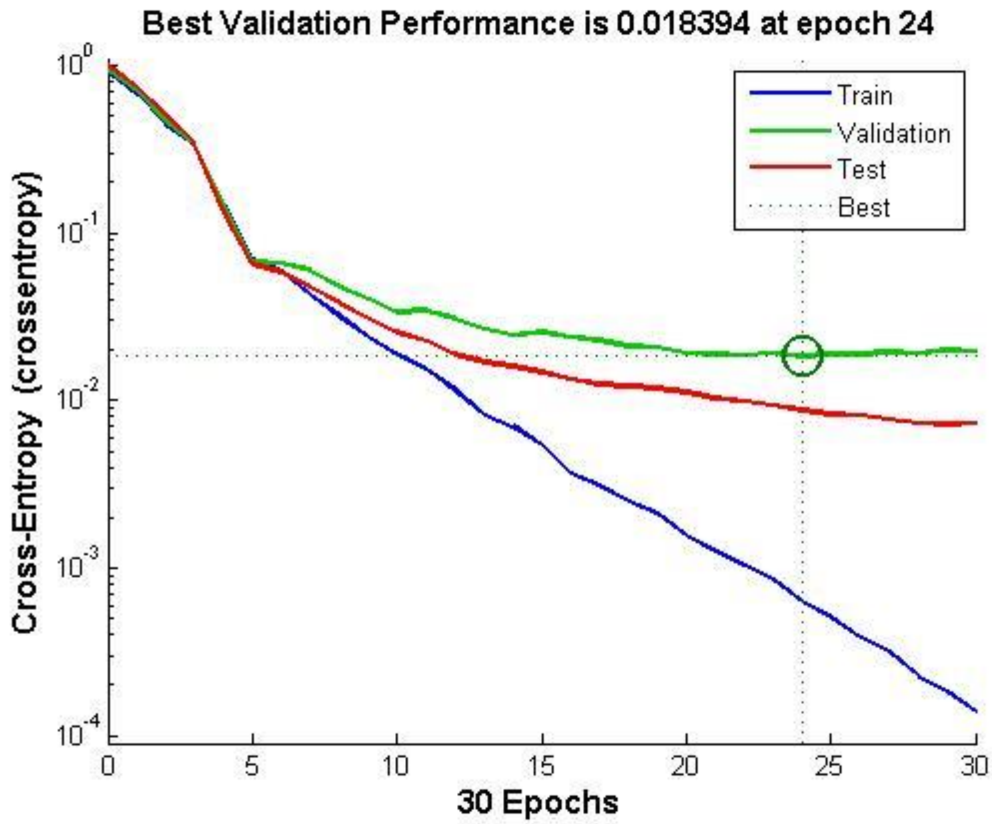

Validation Confusion Matrix
Output Class
Target Class

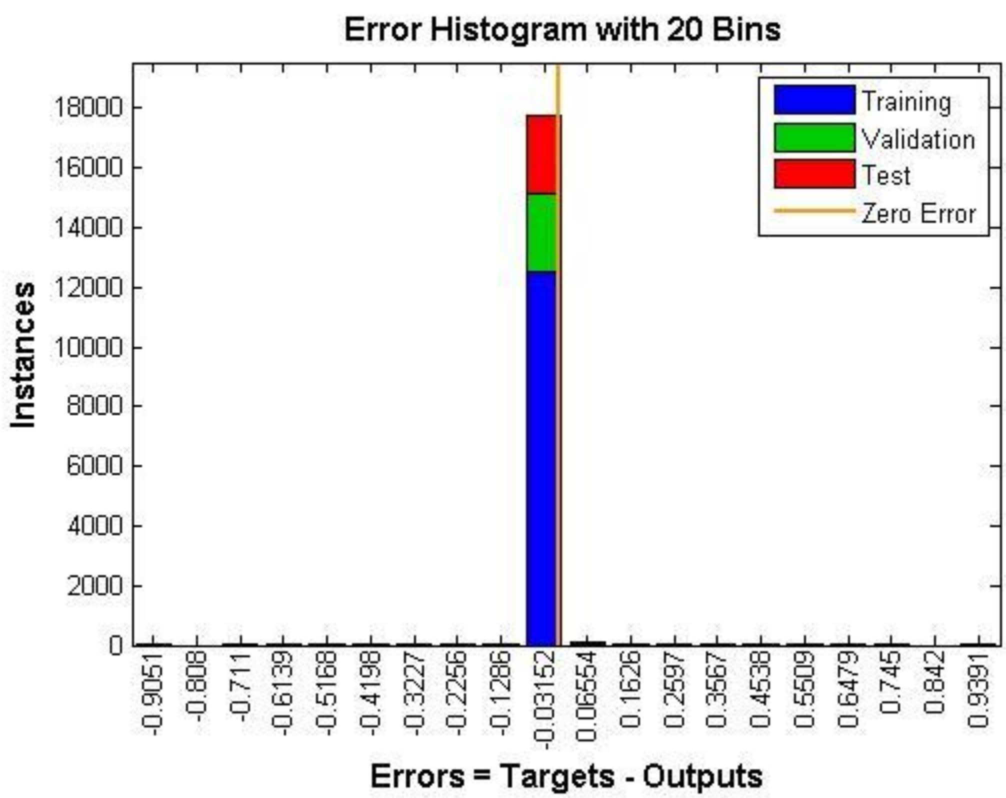

Test Confusion Matrix
Output Class
Target Class

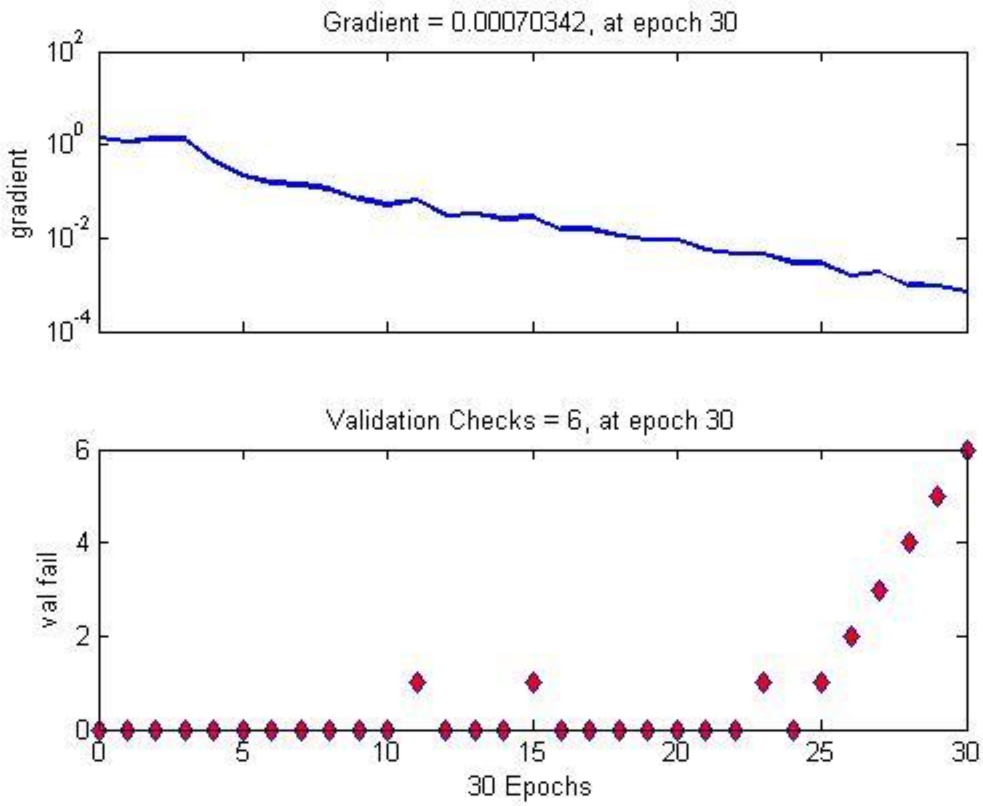

All Confusion Matrix
Output Class
Target Class


Gradient = 8.7593e-05, at epoch 40
gradient
10^{5}
10^{0}
10^{-5}
Validation Checks = 6, at epoch 40
val fail
6
4
2
0
0
5
10
15
20
25
30
35
40
40 Epochs

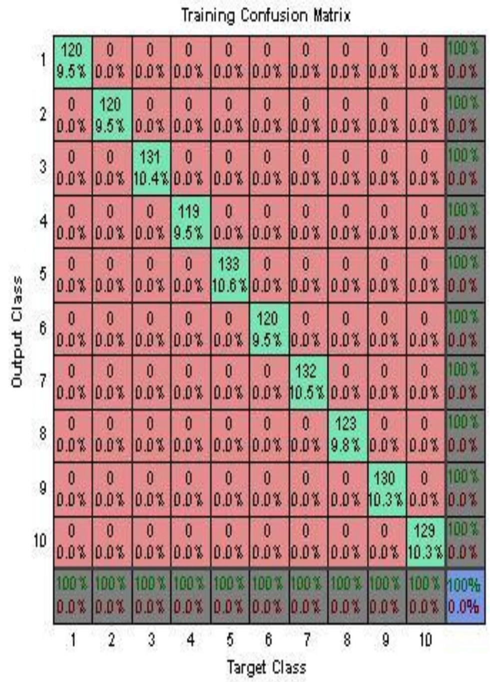
For 100 nodes:

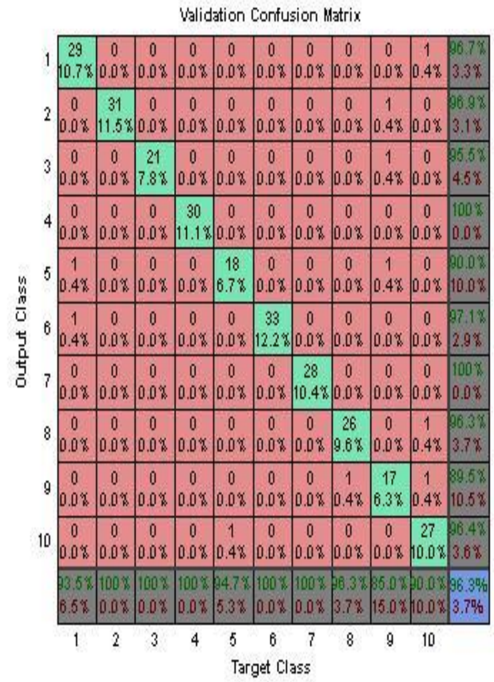


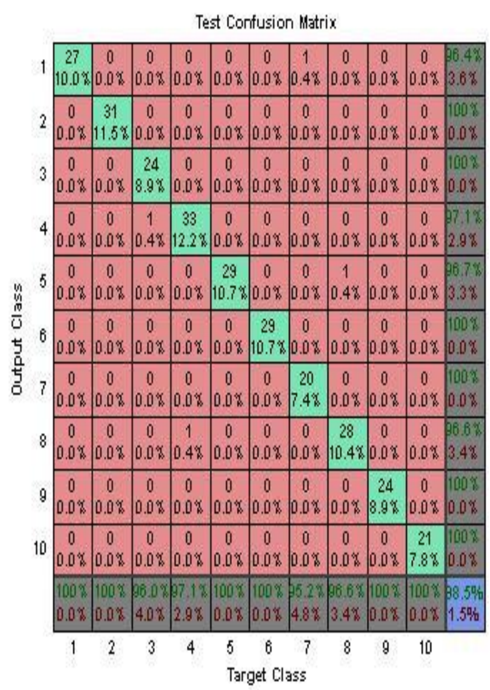


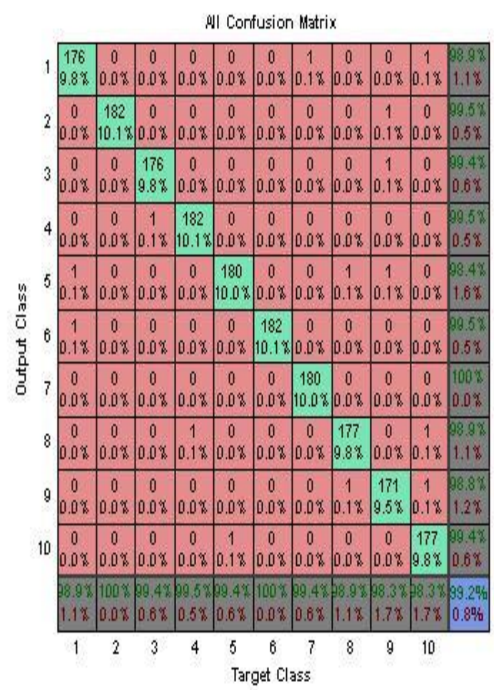


Gradient = 0.00070342, at epoch 30
gradient
Validation Checks = 6, at epoch 30
val fail
30 Epochs

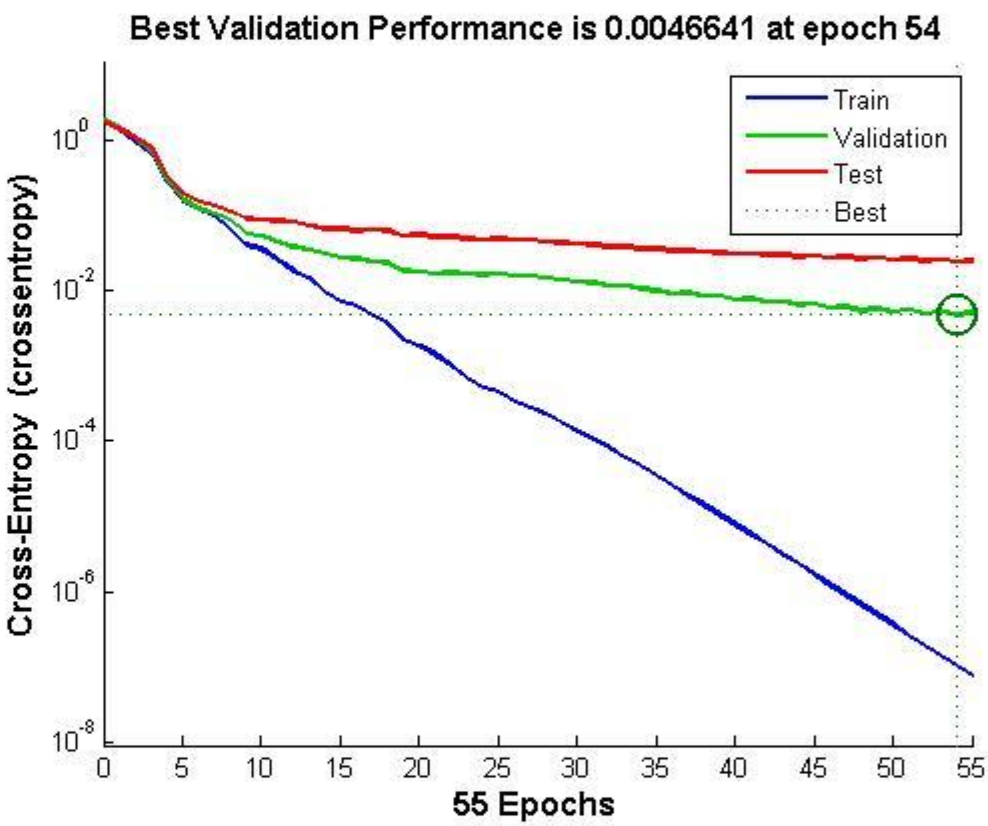

Training Confusion Matrix
Output Class
Target Class

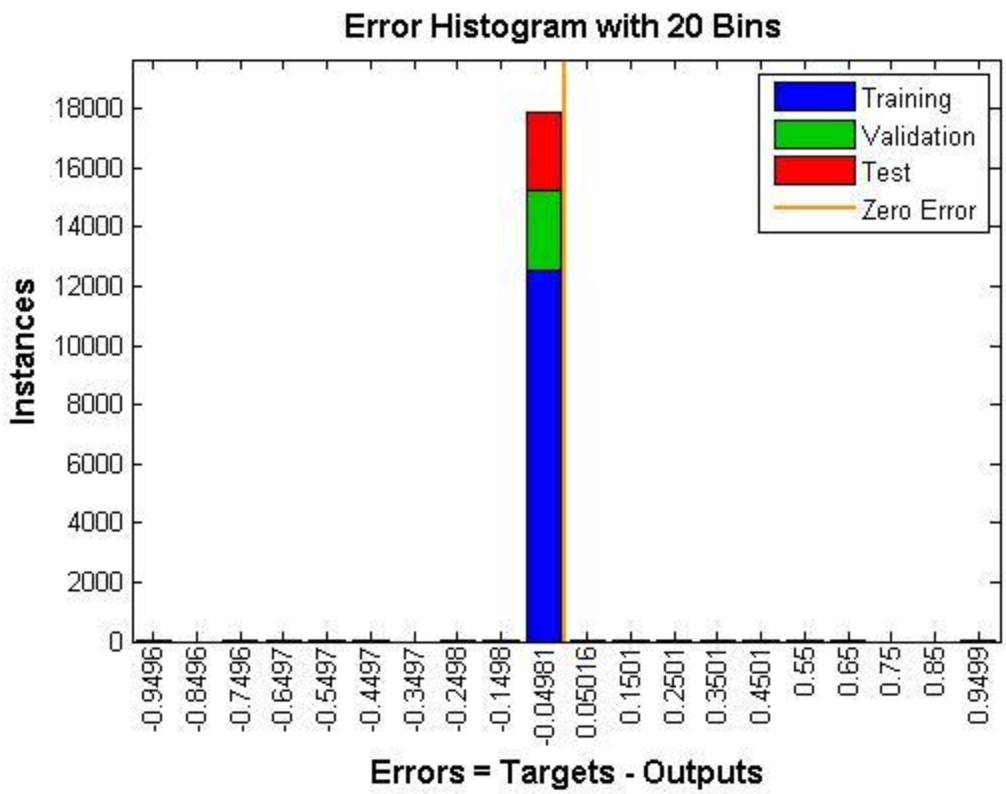

Validation Confusion Matrix
Output Class
Target Class


Test Confusion Matrix
Output Class
Target Class

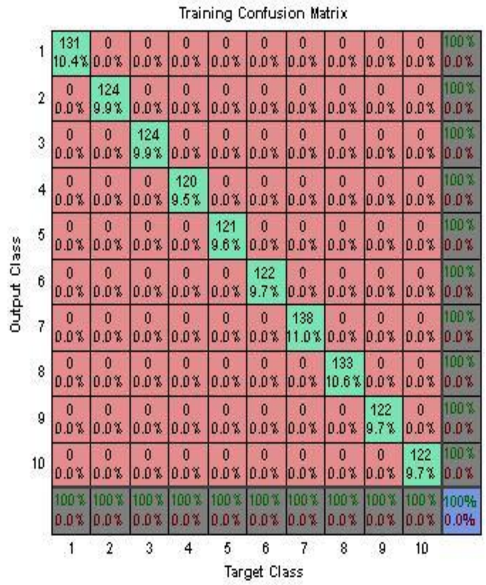

All Confusion Matrix
Output Class
Target Class

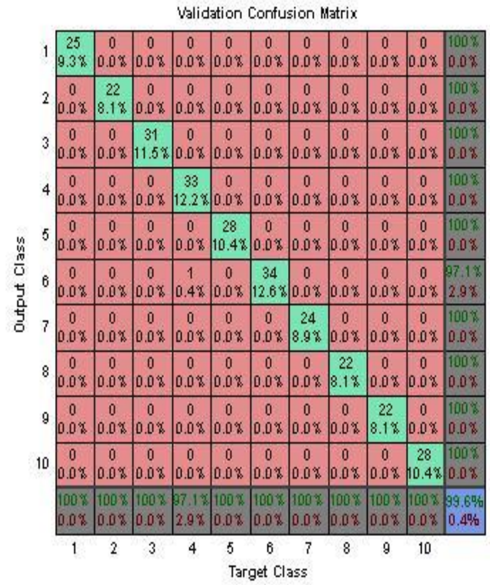
For 500 nodes:

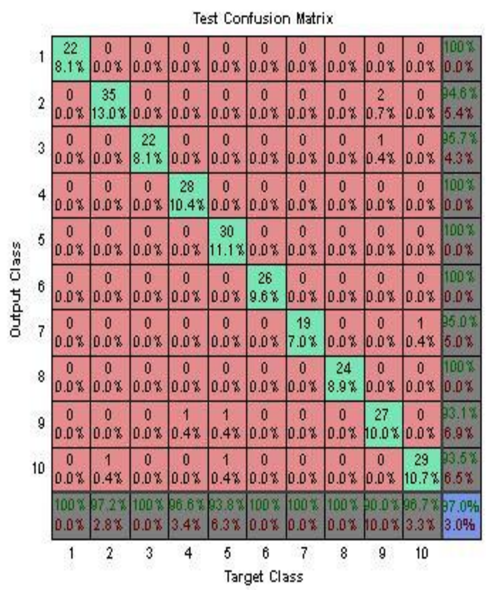


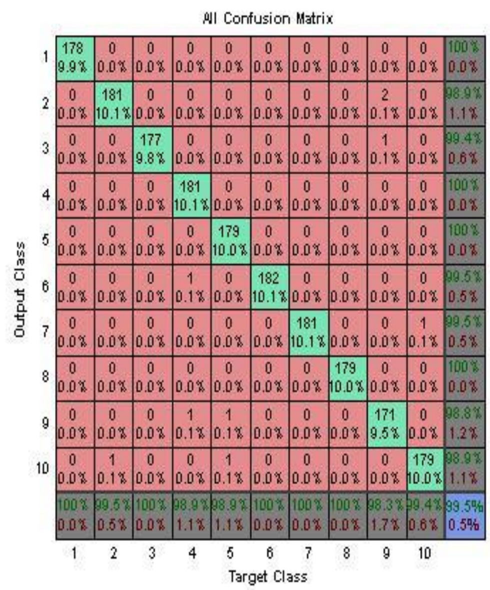


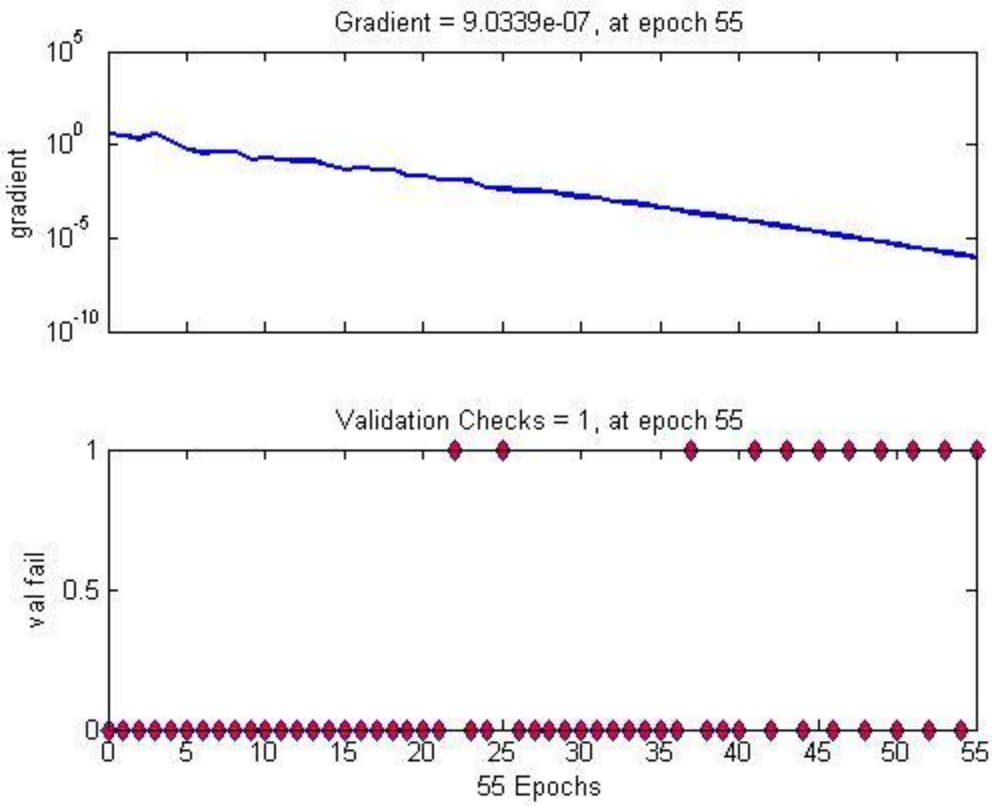
Error Histogram with 20 Bins
Training
Validation
Test
Zero Error
Instances
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
-0.9496
-0.8496
-0.7496
-0.6497
-0.5497
-0.4497
-0.3497
-0.2498
-0.1498
-0.04981
0.05016
0.1501
0.2501
0.3501
0.4501
0.55
0.65
0.75
0.85
0.9499
Errors = Targets - Outputs


Training Confusion Matrix
Output Class
Target Class


Validation Confusion Matrix
Output Class
Target Class


Test Confusion Matrix
Output Class
Target Class

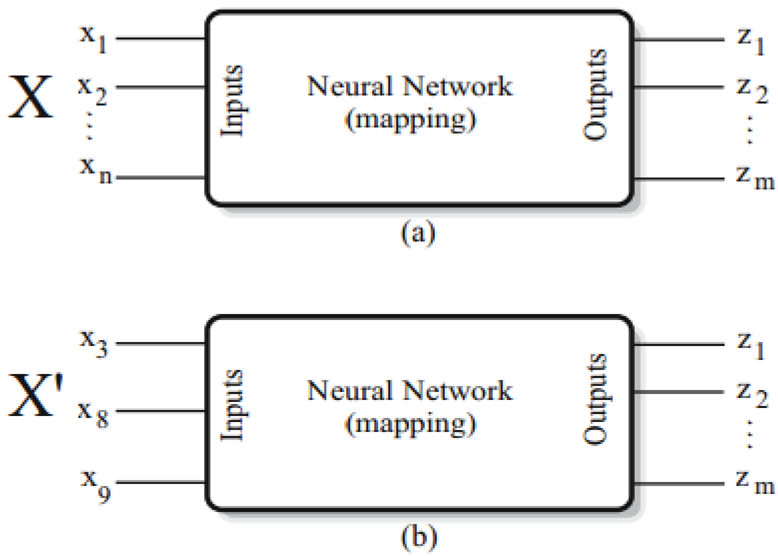

All Confusion Matrix
Output Class
Target Class


Gradient = 9.0339e-07, at epoch 55
gradient
Validation Checks = 1, at epoch 55
val fail
55 Epochs

Task 3:

Using 1024 features for neural networks is cumbersome. So in order to reduce the number of features while keeping the output same, we can make use of the genetic algorithm and the mean-squared error obtained during the training of a neural network to perform feature selection.

We will focus on two typical feature selection problems: mapping and classification. In a mapping problem, a set of input values must produce a set of desired output values. The feature selection problem consists on reducing the number of variables in the input set while producing the same output. Those values that can be removed from the input set may not contain useful information to produce the desired output or may contain redundant information that is already contained in other input variable (feature) or other set of inputs.



On a classification problem, an input set (feature set) X = {x1, x2, ···, xn}, allows identifying the class that these input values belong. Figure 2.a shows an artificial neural network used as a classifier; the input set X = {x1, x2, ···, xn} let the neural network identify the class of each element of the input set. In this case, a feature selection problem consists on computing a new input set that is a subset of X, so that only those features that presumably contain useful information to identify the class are considered.

Consider the input set X = {x1, x2, ··· , xn} with n features, and suppose that there is a subset X with k features taken from the set X, i.e., X = {x3, x8, x9}. Thus, X, a GA individual, can be coded as 0010000110 ···, where the first zero indicates that x1 is not included in X , the third 1 indicates that x3 is included in X , and so on. That is, those features that included in the subset X are represented by one, and those features that left out are represented by zero. This simple coding scheme may be used as long as the number of elements of X be equal to the number of feature to select (k). Finally, it is important to note that each subset of X, a subset of features, may represent an individual with a specific fitness level.

Initial Pool:

To create the initial pool, it is necessary to create a set of individuals; this is a two-step process. First, an individual is created by randomly setting k of its bits to one. Second, we check if this individual is already in the pool; if it is, then a new individual is created; if it is not, then this individual is added to the pool. This process continues until the initial pool is full with different individuals.

Once the initial pool is filled, each individual must be evaluated to assess its fitness. Those most fitted individuals will be the parents of the next generation.

Reproduction and Mutation:

From one generation to the next, the success of GA depends highly on how the individuals are reproduced and mutated. For some implementations of GA, reproduction and mutation are two separated steps; first the individuals are combined using the probability of crossover, then the new individuals are mutated using the probability of mutation. On the other hand, when using GA for feature selection, these two operations must be performed together as the number of features from generation to generation must remain constant.