Carl Glahn

SID: 998092557

ECS 171

Report for HW 2

Notes for code:

I put this in the code as well but credit for all of the m files in the Deep Learning ToolBox folder as well as the HW2_nntester go to Rasmus Berg Palm on github, the code can be found at https://github.com/rasmusbergpalm/DeepLearnToolbox. Although I modified his code for graphing purposes as well as a few other modifications, ultimately most of it was him

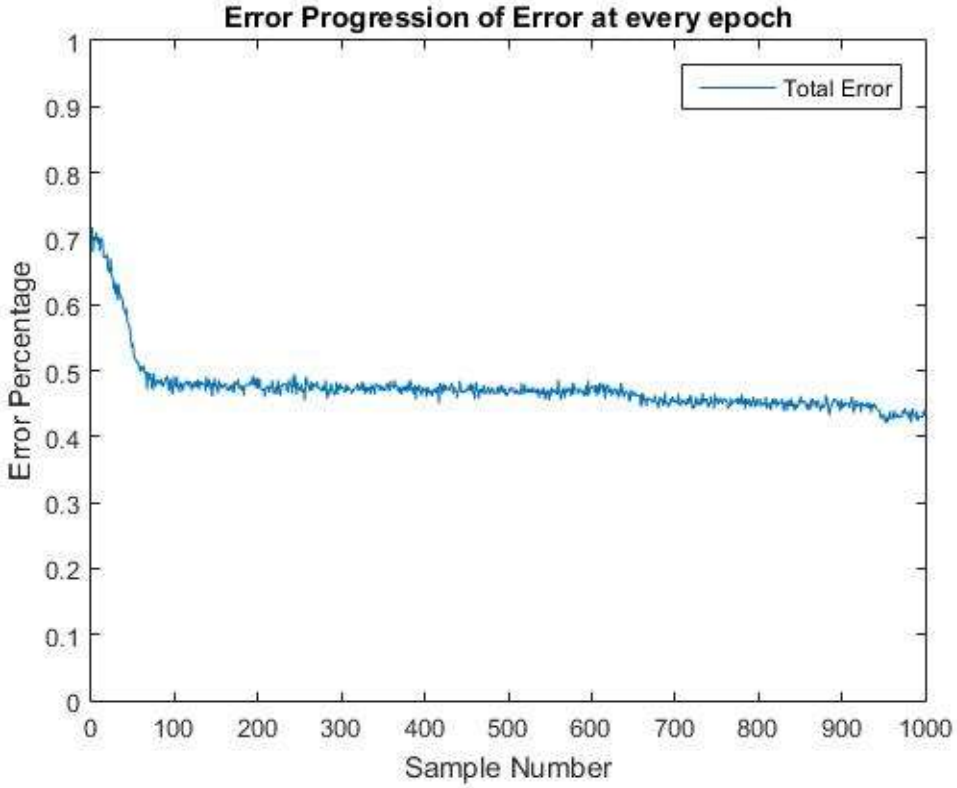
I would also like to thank the TA Ameen Eetemadi for pointing me to the toolbox

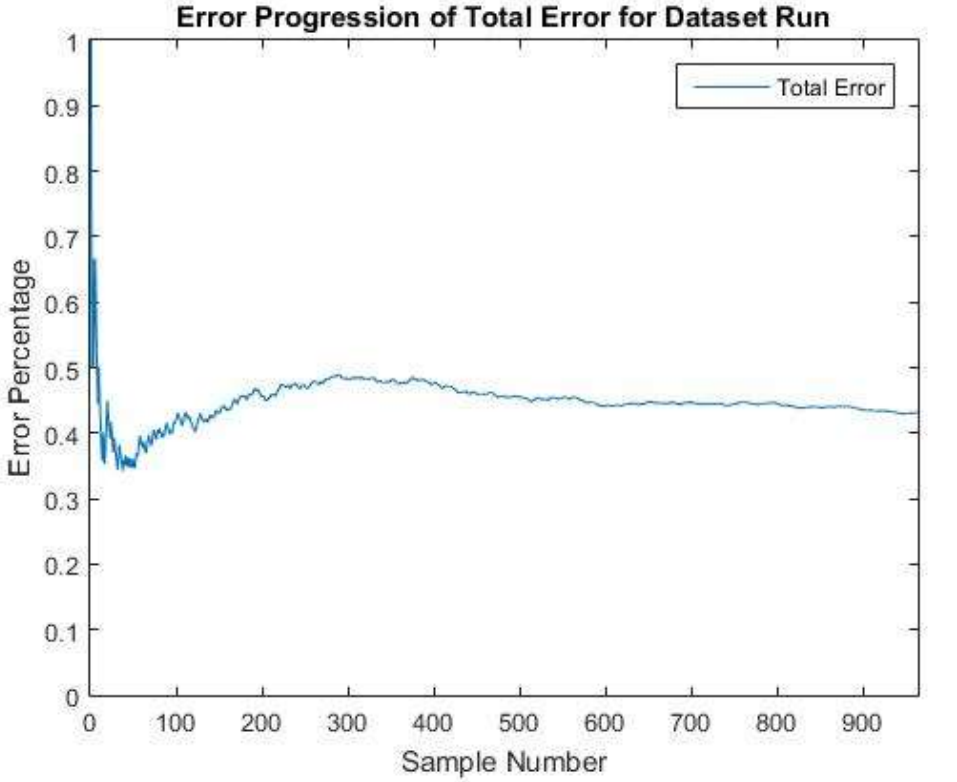
Also a requested by Rasmus Berg Palm, I am to cite his thesis as well:

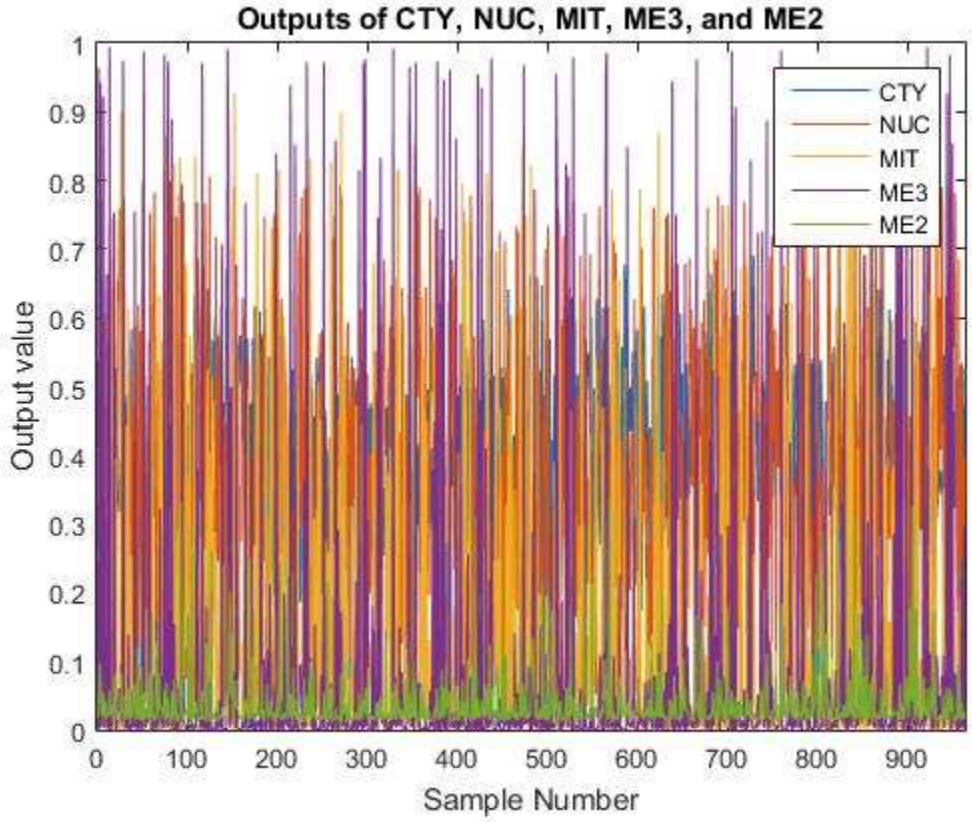
> Palm, Rasmus Berg. Prediction as a Candidate for Learning Deep Hierarchical Models of Data. Thesis. Technical University of Denmark, 2012. Lyngby: Technical U of Denmark, DTU Informatics, 2012. Print.

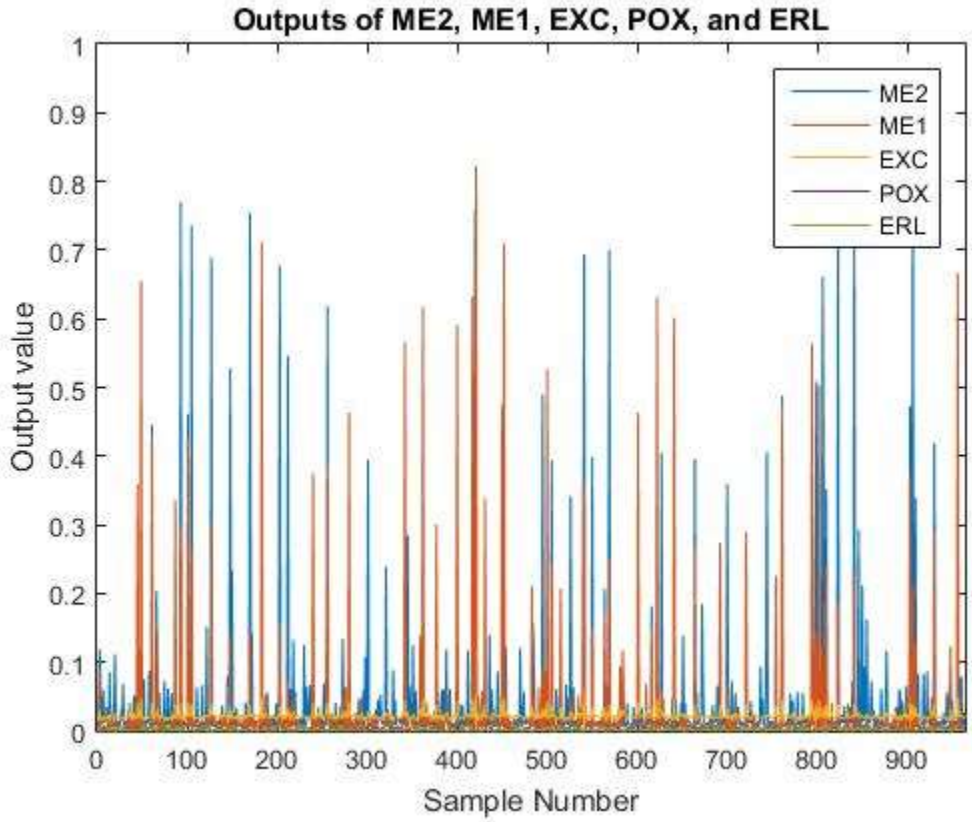
1. As requested I created a neural net that met your specifications. I plotted the weights for each node separately and I also decided to plot the sigmoid values for the activation functions as the information that that gives is far more interesting. I defined the end perceptron function as the max value of my array would be a 1 (the guess of my neural net) and everything else would be a zero. Naturally if my neural net guessed the wrong classification, then I would define that as an error. Something to note is that all of these graphs (except the first one) are for the last epoch of the data (I did 1000 epochs), so don't be too surprised of if they don't move a lot. Also I didn't bother graphing the weights for the test run because they don't get updated.

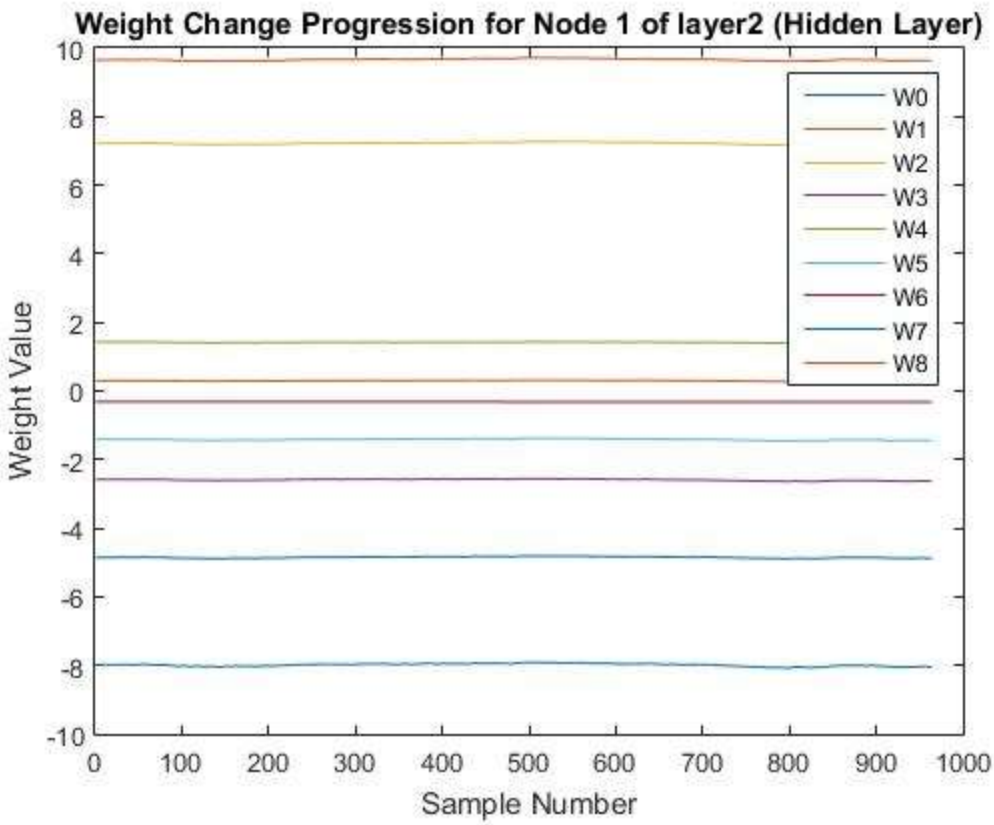
First I will present all of the graphs for the training run:

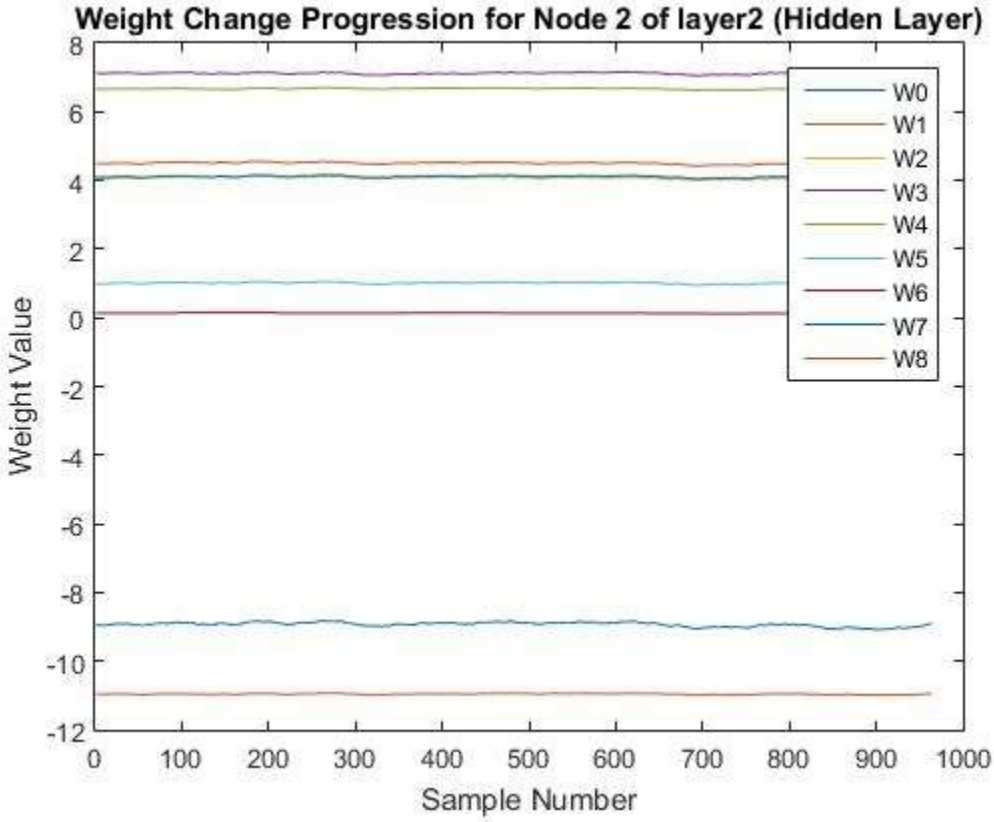


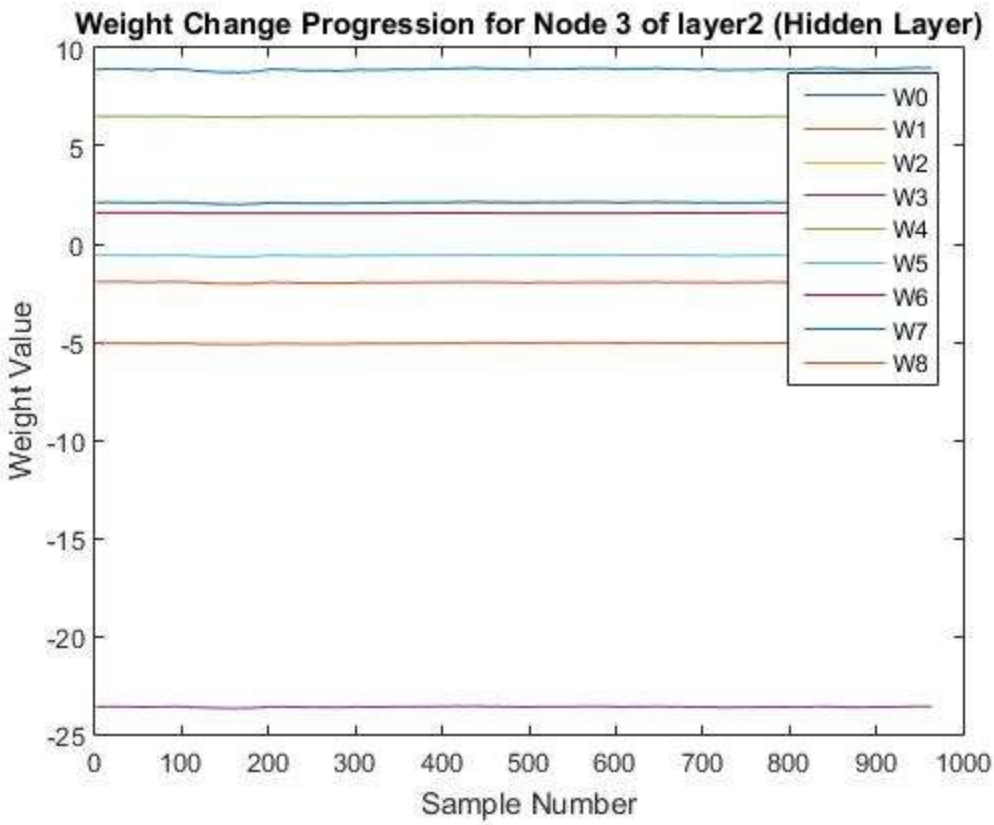


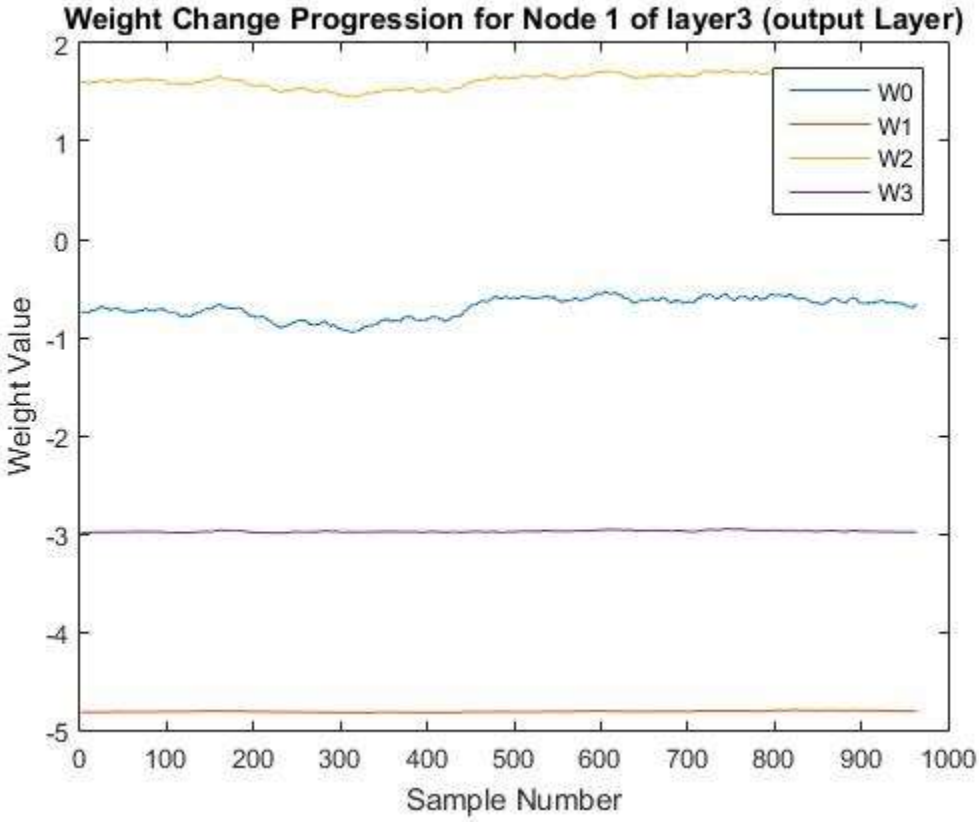

Outputs of CTY, NUC, MIT, ME3, and ME2
CTY
NUC
MIT
ME3
ME2
Output value
Sample Number
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
100
200
300
400
500
600
700
800
900

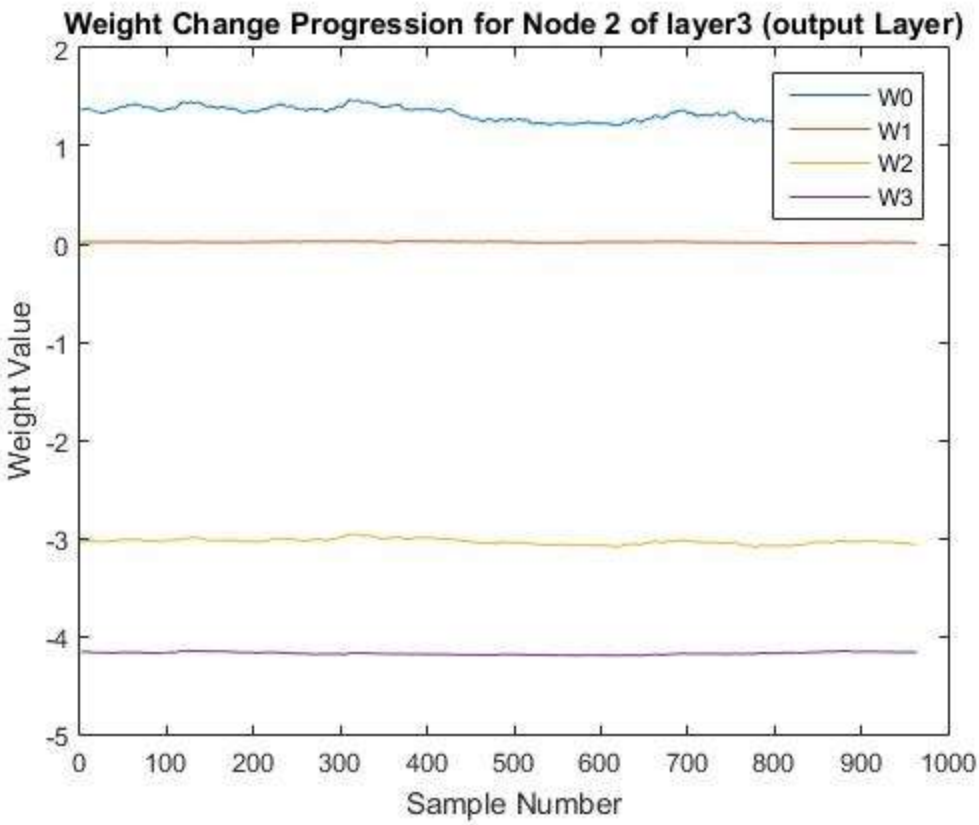

Outputs of ME2, ME1, EXC, POX, and ERL
ME2
ME1
EXC
POX
ERL
Output value
Sample Number
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
100
200
300
400
500
600
700
800
900

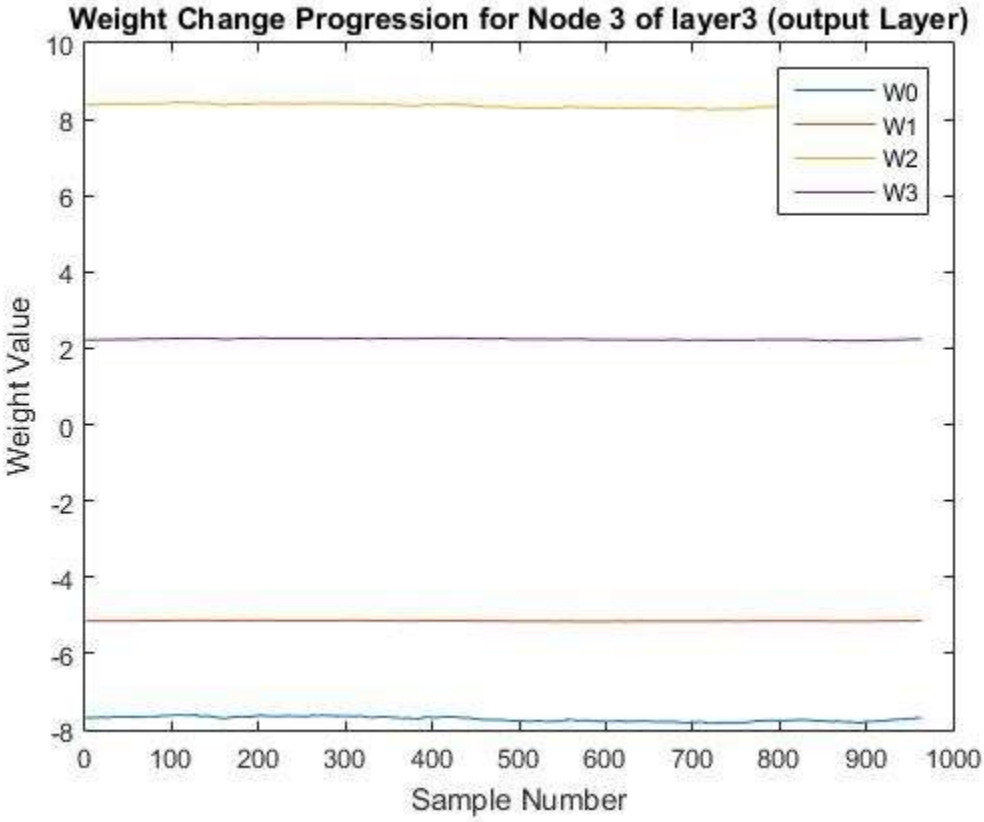

Weight Change Progression for Node 1 of layer2 (Hidden Layer)
Weight Value
Sample Number
W0
W1
W2
W3
W4
W5
W6
W7
W8
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
100
200
300
400
500
600
700
800
900
1000

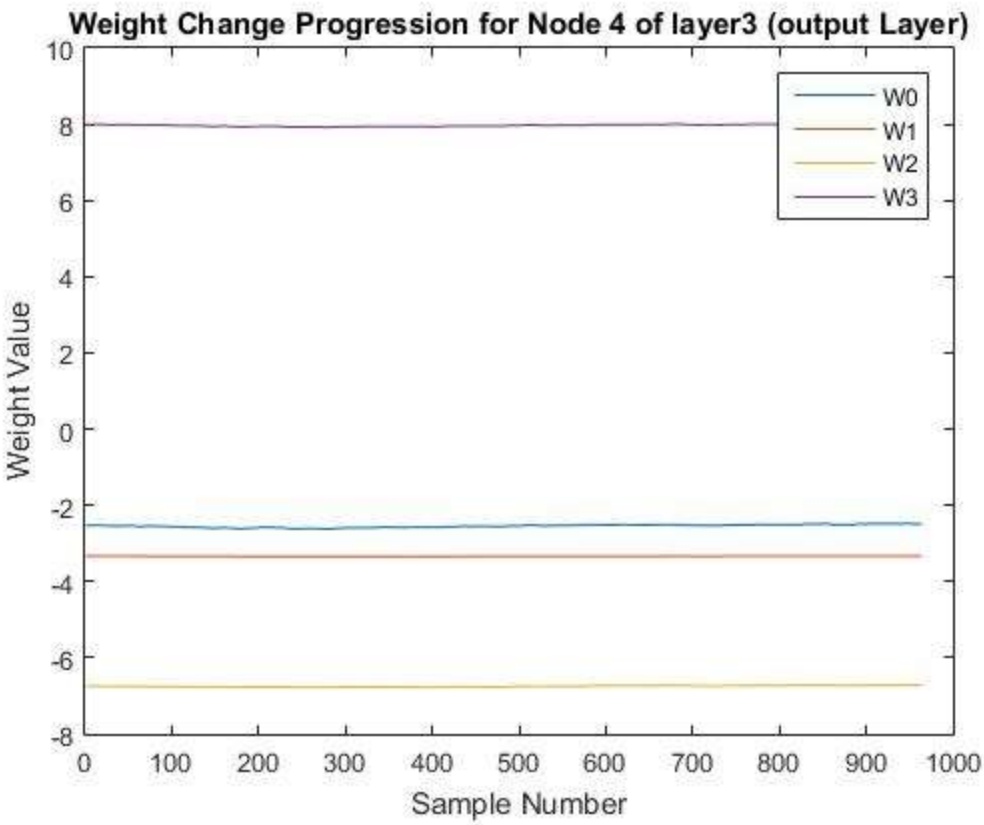

Weight Change Progression for Node 2 of layer2 (Hidden Layer)
Weight Value
Sample Number
W0
W1
W2
W3
W4
W5
W6
W7
W8
8
6
4
2
0
-2
-4
-6
-8
-10
-12
0
100
200
300
400
500
600
700
800
900
1000

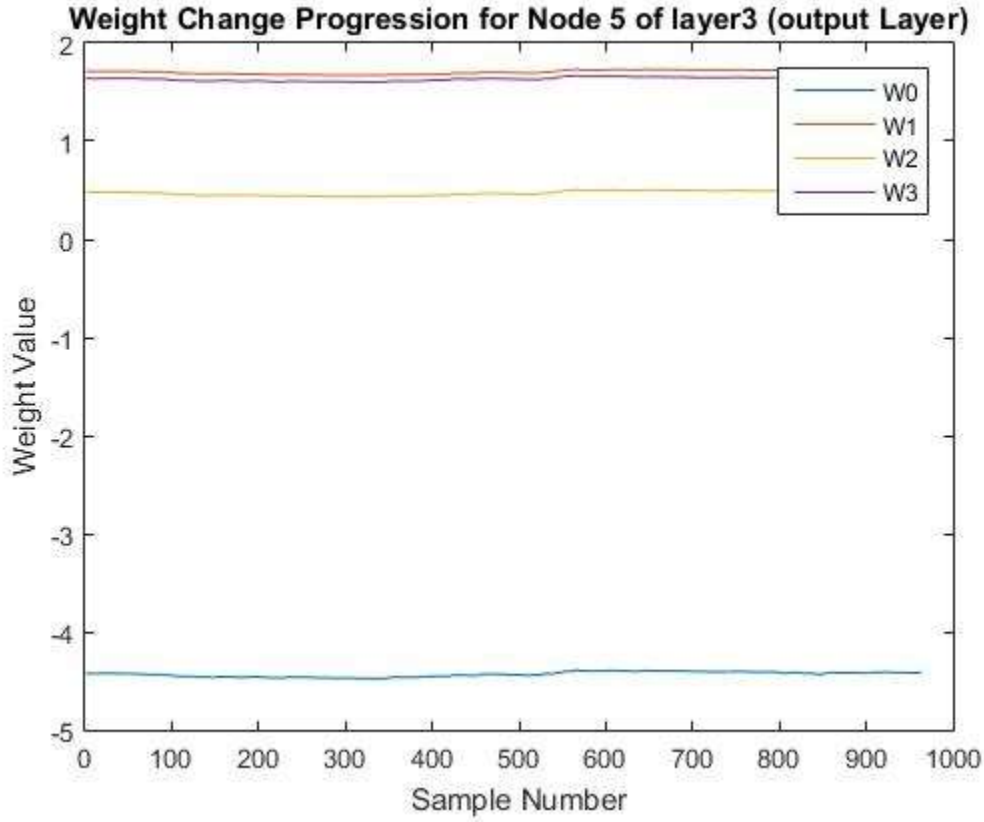

Weight Change Progression for Node 3 of layer2 (Hidden Layer)
Weight Value
Sample Number
W0
W1
W2
W3
W4
W5
W6
W7
W8
10
5
0
-5
-10
-15
-20
-25
0
100
200
300
400
500
600
700
800
900
1000

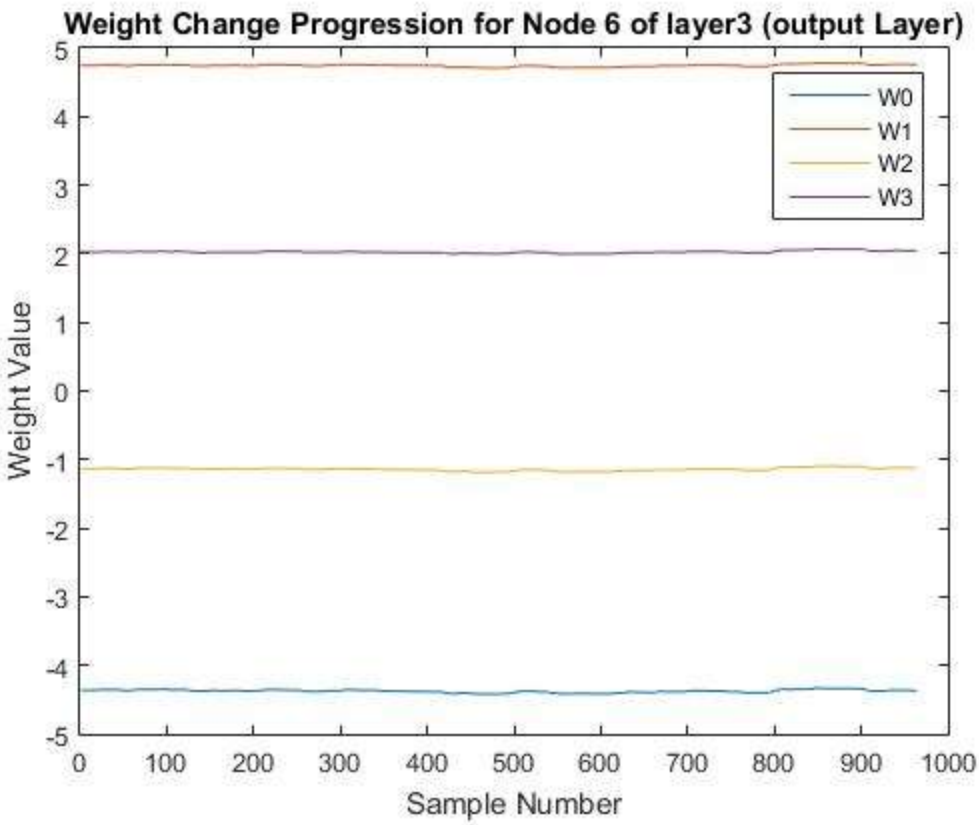

Weight Change Progression for Node 1 of layer3 (output Layer)
Weight Value
Sample Number
W0
W1
W2
W3
2
1
0
-1
-2
-3
-4
-5
0
100
200
300
400
500
600
700
800
900
1000

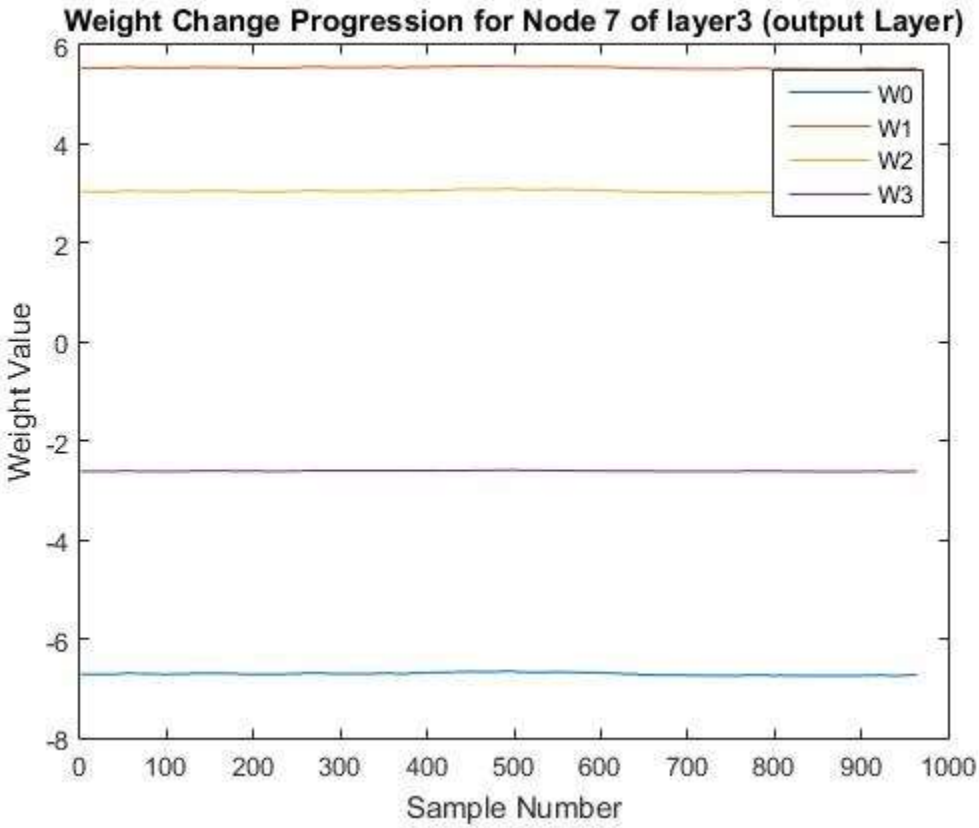

Weight Change Progression for Node 2 of layer3 (output Layer)
W0
W1
W2
W3
Weight Value
Sample Number
2
1
0
-1
-2
-3
-4
-5
0
100
200
300
400
500
600
700
800
900
1000

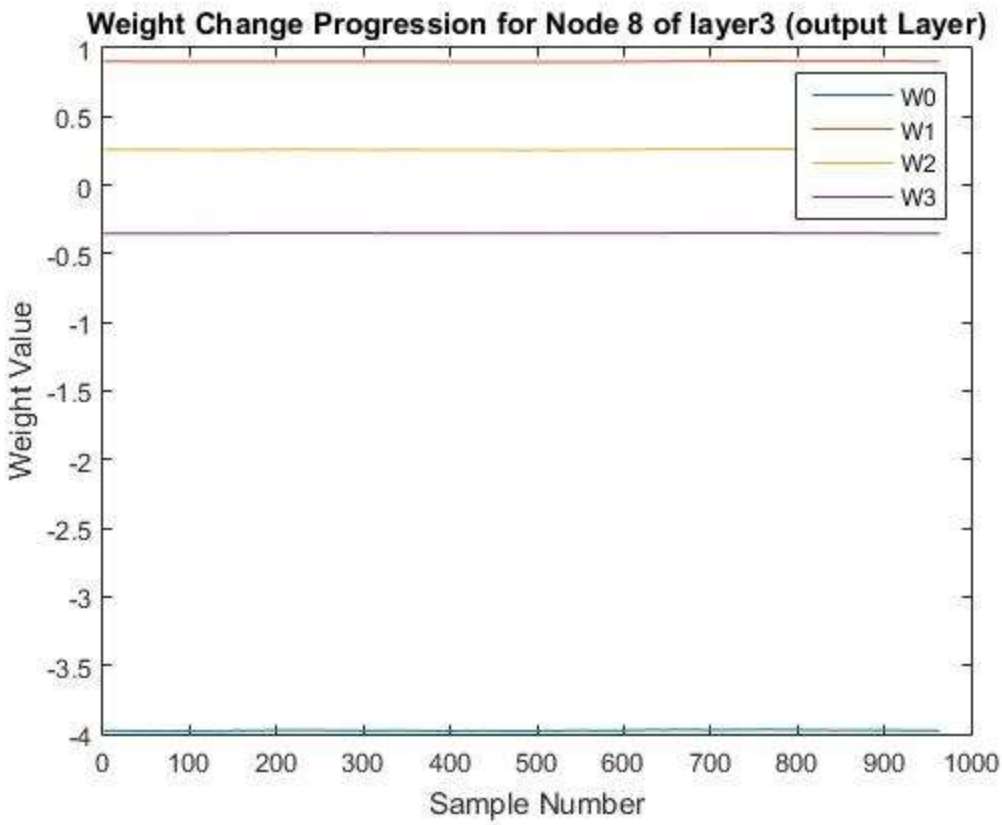

Weight Change Progression for Node 3 of layer3 (output Layer)
W0
W1
W2
W3
Weight Value
Sample Number
10
8
6
4
2
0
-2
-4
-6
-8
0
100
200
300
400
500
600
700
800
900
1000

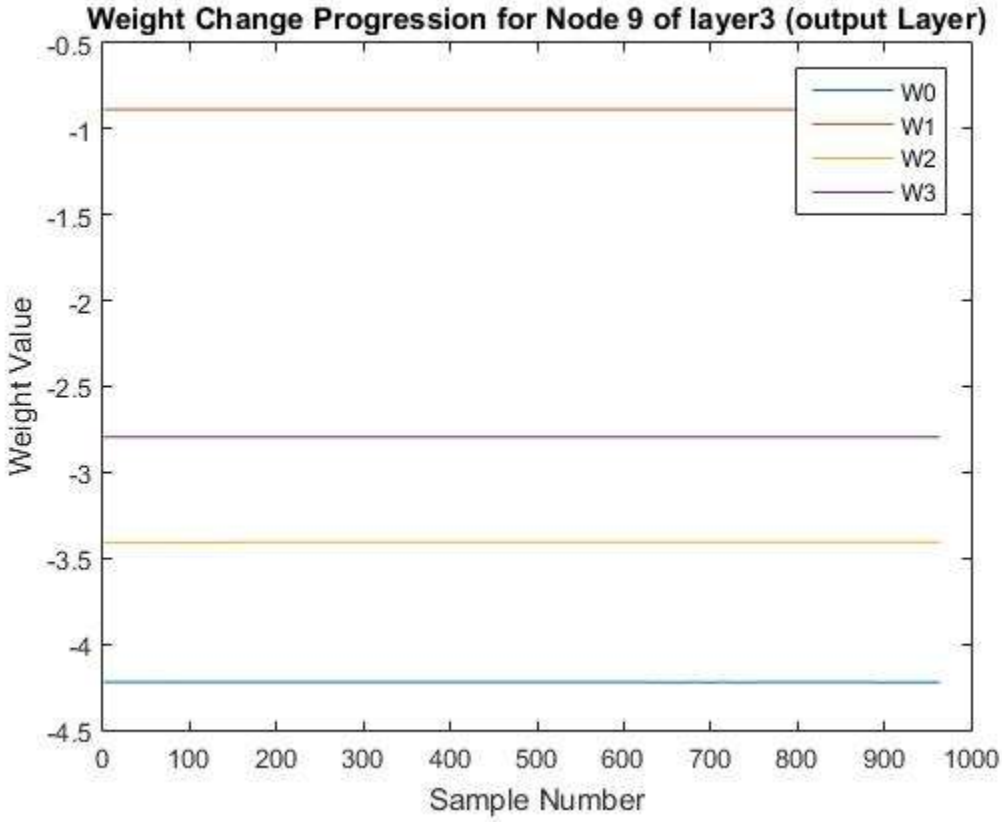

Weight Change Progression for Node 4 of layer3 (output Layer)
W0
W1
W2
W3
Weight Value
Sample Number
10
8
6
4
2
0
-2
-4
-6
-8
0
100
200
300
400
500
600
700
800
900
1000

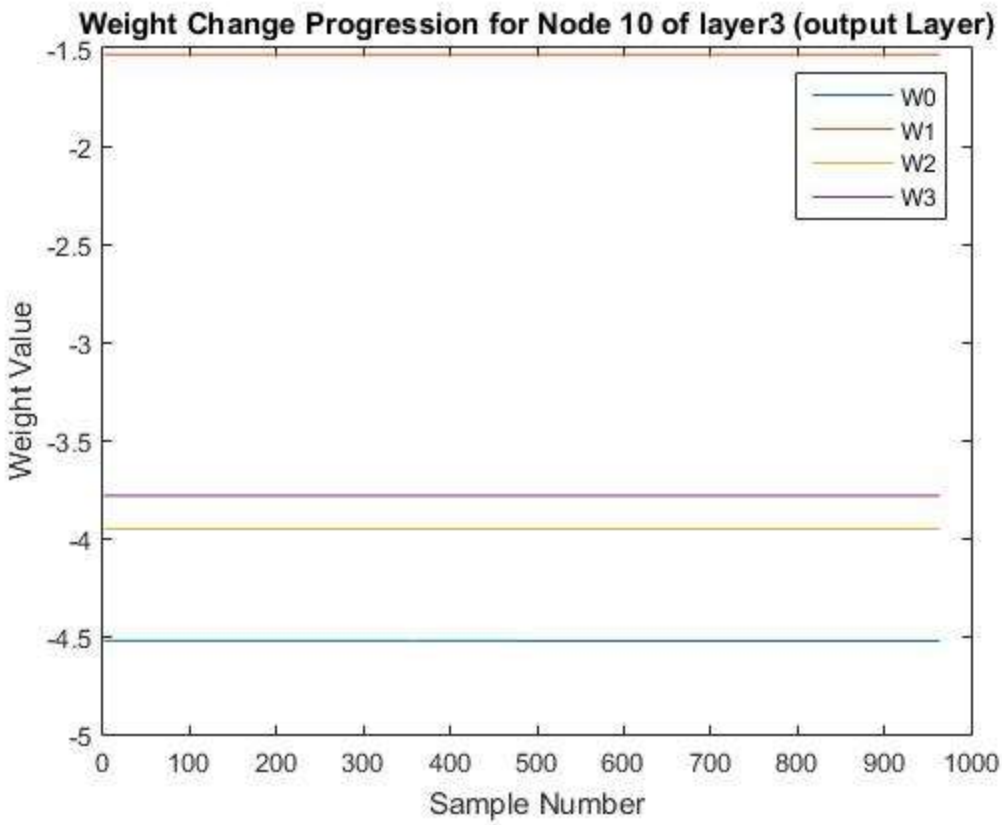

Weight Change Progression for Node 5 of layer3 (output Layer)
W0
W1
W2
W3
Weight Value
Sample Number
2
1
0
-1
-2
-3
-4
-5
0
100
200
300
400
500
600
700
800
900
1000


Weight Change Progression for Node 6 of layer3 (output Layer)
W0
W1
W2
W3
Weight Value
Sample Number
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
100
200
300
400
500
600
700
800
900
1000

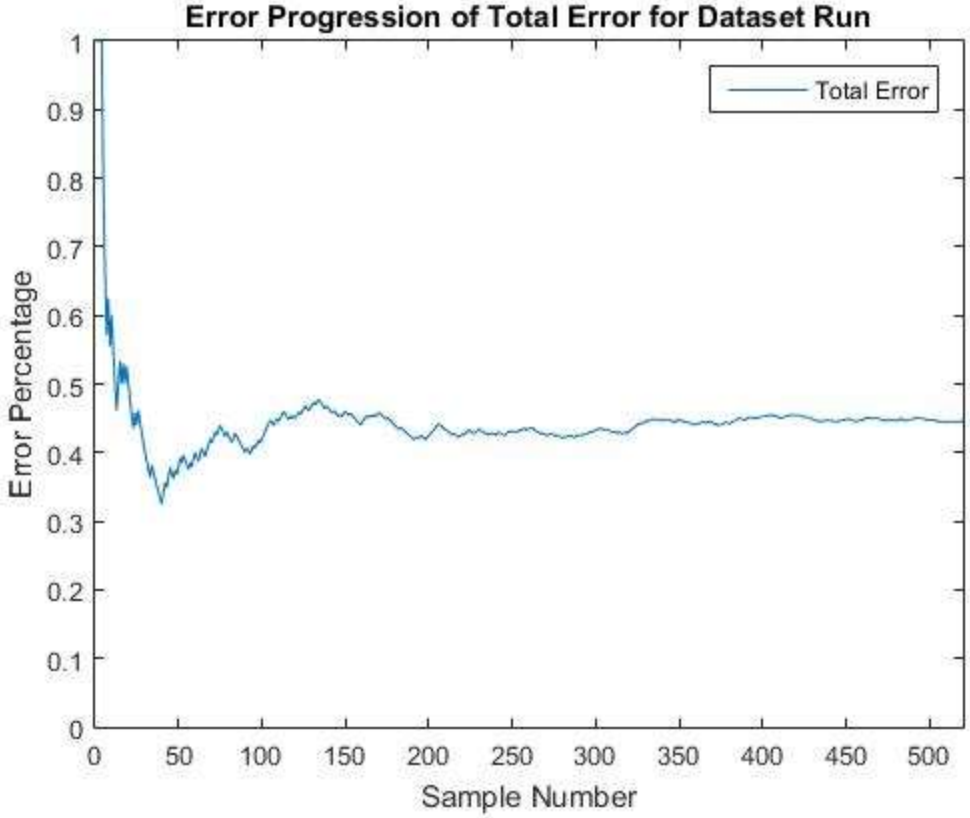

Weight Change Progression for Node 7 of layer3 (output Layer)
W0
W1
W2
W3
Weight Value
Sample Number
6
4
2
0
-2
-4
-6
-8
0
100
200
300
400
500
600
700
800
900
1000

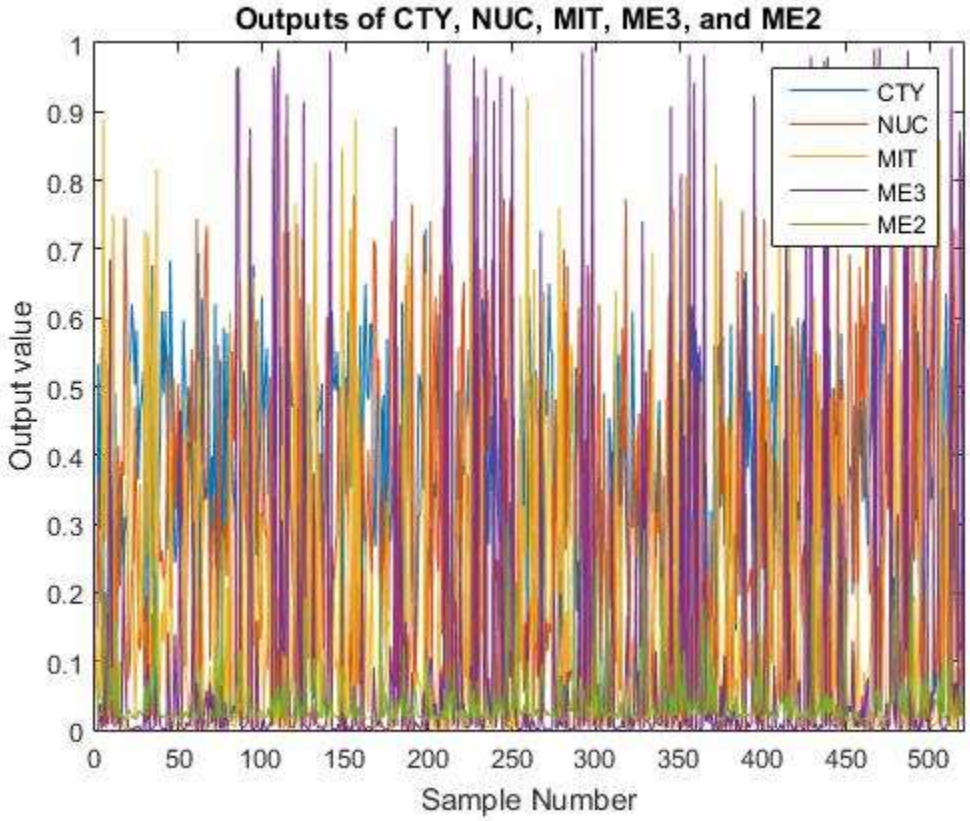

Weight Change Progression for Node 8 of layer3 (output Layer)
Weight Value
Sample Number
W0
W1
W2
W3
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
0
100
200
300
400
500
600
700
800
900
1000

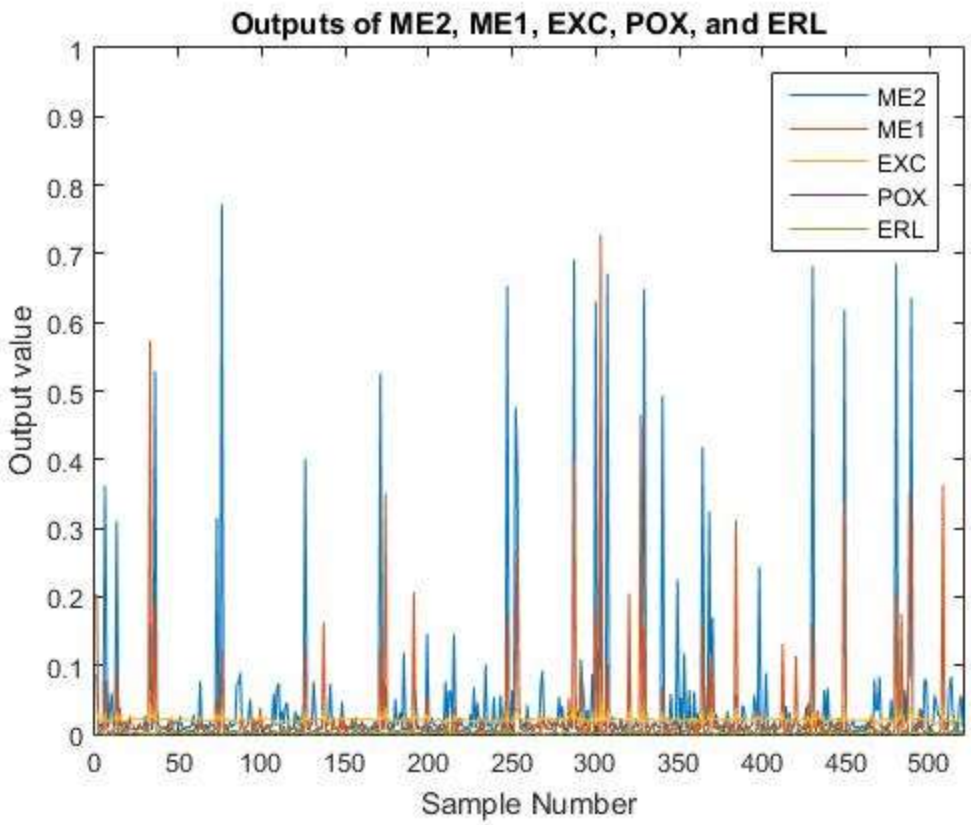

Weight Change Progression for Node 9 of layer3 (output Layer)
Weight Value
Sample Number
W0
W1
W2
W3
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
-4.5
0
100
200
300
400
500
600
700
800
900
1000



And now for the test set:


Error Progression of Total Error for Dataset Run
Total Error
Error Percentage
Sample Number
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400
450
500


Outputs of CTY, NUC, MIT, ME3, and ME2
CTY
NUC
MIT
ME3
ME2
Output value
Sample Number
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
50
100
150
200
250
300
350
400
450
500



2. For this problem I trained on the entire dataset and got an error of .40 which is about 3% down from the error on the previous problem. The activation functions are outlined below, the denotation is a(node number)(layer number). Note: I rounded the weights to the 5th decimal place for clarity purposes.

   a12 = sigm(X0*-8.4841 + X1*8.5331 + X2*7.3010 +X3*2.3006 + X4*10.8892 + X5*-5.7692 + X6*3.7306 + X7*1.8993 + X8*-11.1406)

   a22 = sigm(X0*0.8344 + X1*-5.2220 + X2*-1.192 +X3*-0.9350 + X4*5.5081 + X5*-5.2890 + X6*0.3161 + X7*-1.1181 + X8*19.1777)

   a32 = sigm(X0*-10.2659 + X1*1.3687 + X2*4.4129 +X3*-30.6628 + X4*-1.1989 + X5*1.4537 + X6*2.3686 + X7*-0.7075 + X8*-1.1617)

   a13 = sigm(X0*1.6438 + a12* -1.5232 + a22* -2.4020+a32* -3.5647)

   a23 = sigm(X0* -1.3133+ a12* -2.5845+ a22* 2.3450+a32* -3.1626)

   a33 = sigm(X0* -5.1800+ a12* 4.5440+ a22* 2.9965+a32* -2.7060)

   a43 = sigm(X0* -4.1732+ a12* -6.2190+ a22* 1.3218+a32* 7.6088)

a53 = sigm(X0* -3.3235+ a12* 1.0785+ a22*-2.7146+a32*2.0794)

a63 = sigm(X0*-10.8512+ a12*7.4590+ a22*-0.0548+a32*4.2566)

a73 = sigm(X0*-7.1431+ a12*7.7224+ a22*-4.5811+a32*-1.3534)

a83 = sigm(X0*-2.8288+ a12*-1.9701+ a22*-2.3621+a32*1.6966)

a93 = sigm(X0*-4.4701+ a12*1.3594+ a22*-0.5763+a32*-0.1432)

a103 = sigm(X0*-2.4853+ a12*0.5170 + a22*-7.5827+a32*0.1416)

3. See attached sheets for handwritten calculations. As a side note I manually started the weights at .5 for simplicity's sake (makes it easier to calculate and grade). The final weights are displayed down below.

Weights Going to Layer 2	X0	X1	X2	X3	X4	X5	X6	X7	X8
To node 1	0.497596	0.498606	0.498534	0.49887	0.499687	0.498798	0.5	0.498846	0.499471
To node 2	0.497596	0.498606	0.498534	0.49887	0.499687	0.498798	0.5	0.498846	0.499471
To node 3	0.497596	0.498606	0.498534	0.49887	0.499687	0.498798	0.5	0.498846	0.499471

Weights Going to Layer 3	a0	a1	a2	a3
To node 1	0.494838	0.495456	0.495456	0.495456
To node 2	0.494838	0.495456	0.495456	0.495456
To node 3	0.500836	0.500736	0.500736	0.500736
To node 4	0.494838	0.495456	0.495456	0.495456
To node 5	0.494838	0.495456	0.495456	0.495456
To node 6	0.494838	0.495456	0.495456	0.495456
To node 7	0.494838	0.495456	0.495456	0.495456
To node 8	0.494838	0.495456	0.495456	0.495456
To node 9	0.494838	0.495456	0.495456	0.495456
To node 10	0.494838	0.495456	0.495456	0.495456

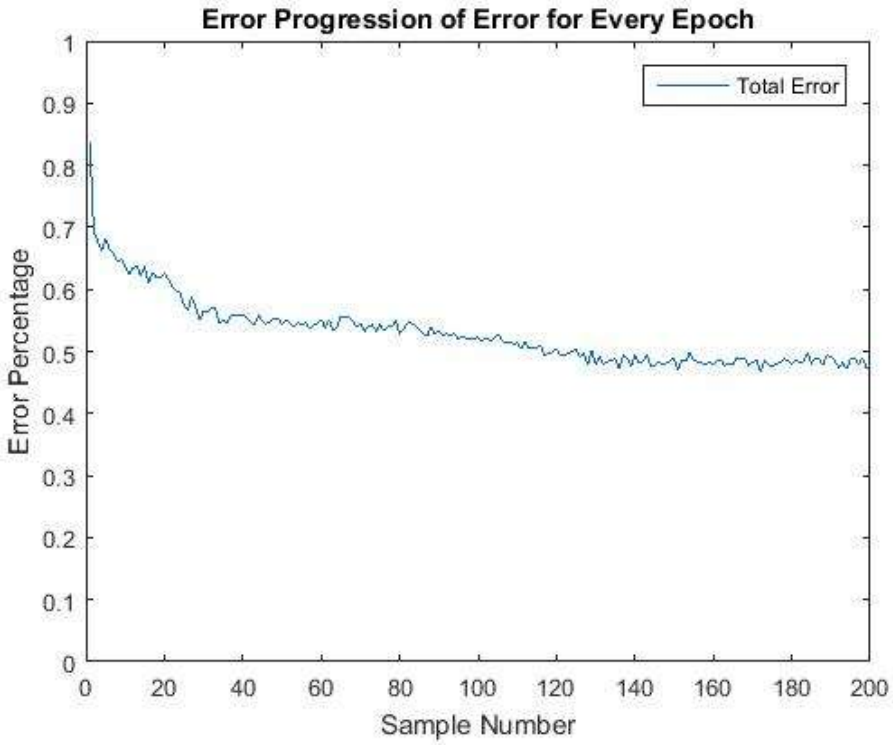
4. To address this problem I first made all the neural nets, trained them with 500 epochs and tested them. To ensure that the randomly generated weights didn't produce a particularly low/high error I took the mean of 5 runs on every neural net. What I found is that all of the machines get to about the same level of accuracy after 500 epochs, with the accuracy improving by a few percent as the number of nodes increased. Interestingly enough it seemed that the 3 hidden layer neural net preformed worse than the 1 and 2 hidden layer versions. The results for these tests are below.

Layers/nodes	3	6	9	12
1	0.446421	0.436538462	0.401923077	0.419230769
2	0.456538462	0.419615385	0.4175	0.416153846
3	0.457692308	0.434615385	0.438461538	0.421153846

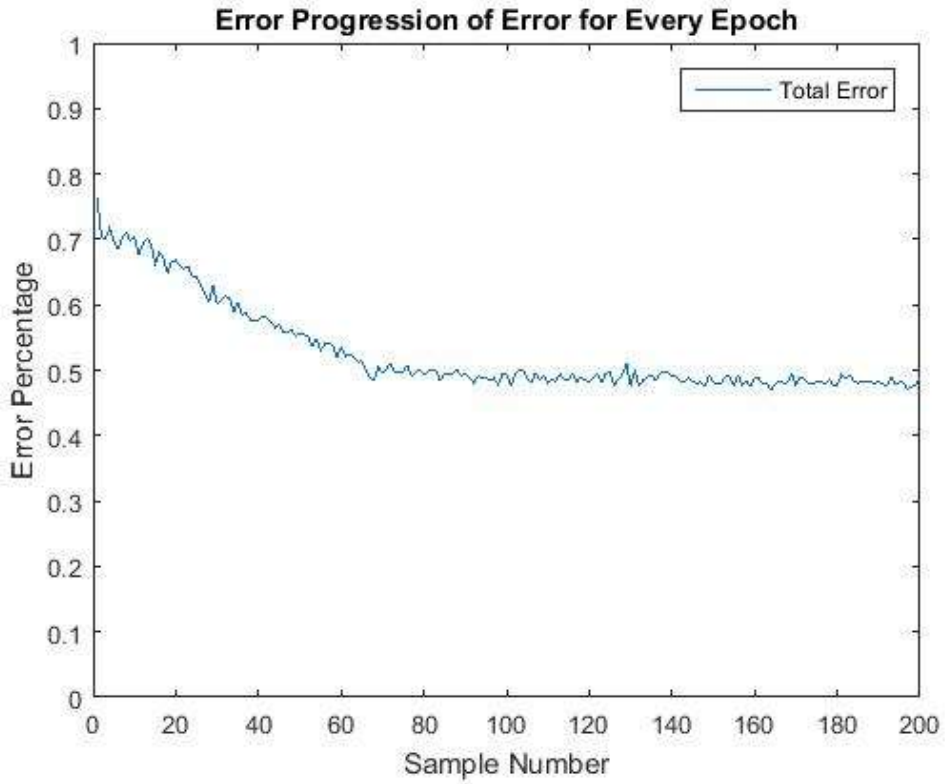
As you can see from my results, it would seem that having 12 nodes with 2 hidden layers is the best option, however only by a marginal amount.

To further investigate the differences between these neural nets I reran the program with only 200 epochs to better examine the learning curve for each neural net. Naturally all the final errors were higher but I did notice that the nets with more nodes seems to learn faster and reach their max accuracy sooner. I won't post all of the graph as that would be excessive, however I did think that these 2 in particular were representative.

1 Layer 3 Nodes: reaches its max at about epoch 120



1 Layer 6 Nodes: reaches its max at about epoch 80



The table below is the results from that test run, and they seem to reinforce my statement about which neural net is the optimal configuration.

Layers/nodes	3	6	9	12
1	0.471538462	0.430769231	0.42	0.420769231
2	0.523076923	0.418461538	0.415	0.411538462
3	0.581538462	0.459230769	0.419230769	0.426538462

5. For this problem I decided to train with my whole sample to have the most accuracy possible. My output was highest for the node that I encoded as a representation of "NUC" which makes it the prediction for my neural net.
6. As discussed in lecture, a good way to capture the accuracy of one's neural net it to use both a precision measure and a sensitivity measure for each classification. This way we not only know how many positives did the neural net get right, but also how of the positives did it capture overall. For example, there are only three or so ERL yeast samples in our entire dataset, so naturally my neural network guesses very low for ERL consistently. Given this, we both our precision and sensitivity would be 0 as we would never guess positively for that classification, and therefore our uncertainty would be high for that classification. NUC on the other hand is quite popular in our data set, and so my neural net has many opportunities to train on these samples and gets many of them correct. Naturally my precision and sensitivity measures for this

classification would be much higher and therefore our uncertainty would be much lower. As to the last unknown sample, in order to give an exact number I would have to write up some code, however barring that I would say my neural network's guess of NUC has a fairly low amount of uncertainty to it.