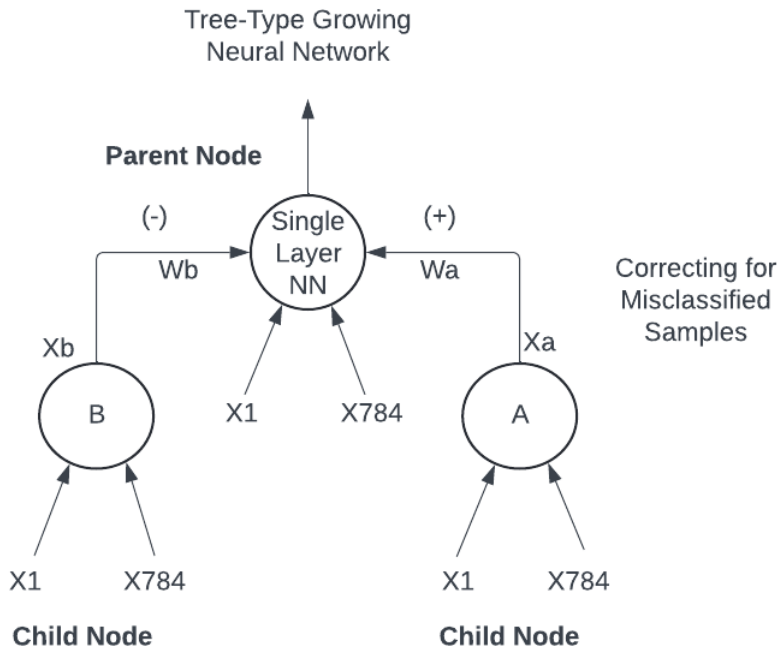
ELL784 Machine Learning Assignment 2

Kashish Srivastava (2024AIB2289)

26th August 2024

1. Making a Tree-Type Growing Neural Network

a. Tree-Type Growing Neural Network Definition:



Growing Tree-Type Neural Network Architecture

The structure of network grows incrementally, with new child node being added to address misclassified sample by parent node. The network as a whole is used to classify samples as 0 or not 0 (i.e. 0 vs all classifier)

Dataset used for this assignment is MNIST dataset. Loss Function used with the model = **Mean Squared Error (MSE).**

b. Correction of Misclassified Classes:

Parent node of the NN is basically just a binary classified with 784 inputs. Since there are some misclassified samples from dataset, we can correct them by classifying observations into 4 class. Using these classes, we can add child nodes which will correct the parent classifier.

Classes	Actual	Desired
C1	0	0
C2	1	1
C3	0	1
C4	1	0

We will add 2 child node A and B which will give out Xa and Xb (respectively), which will be fed into parent node.

Classes	Xa	Xb
C1	0	X
C2	X	0
C3	1	0
C4	0	1

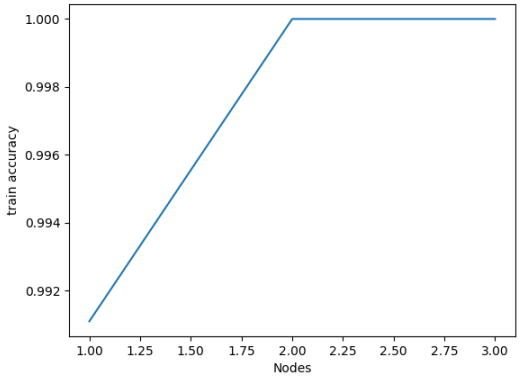
c. Training the model:

Each Layer of Tree NN was trained for 30 epochs with learning rate = 0.08. Stopping conditions for growth of NN is maximum depth (layer). Here, we have used maximum depth=3.

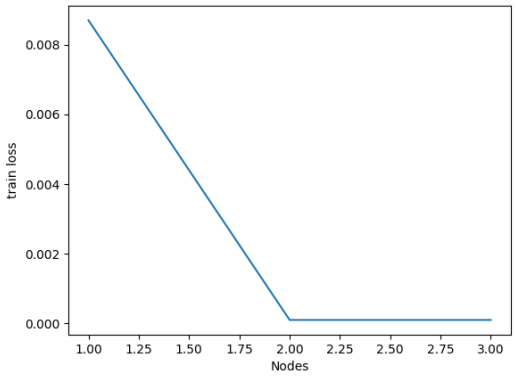
All misclassified samples from parent nodes were corrected by child nodes and parent node was retrained with additional Inputs Xa and Xb. Following table values refer to Layers i.e. the max depth for each NN model. Layer 1 refers to just a binary classifier without adding child node.

Layer 2,3 refers to Parent node + child nodes which correct the misclassification by Parent node. By running treeNN.py script, observed values from training and testing the model are as follows:

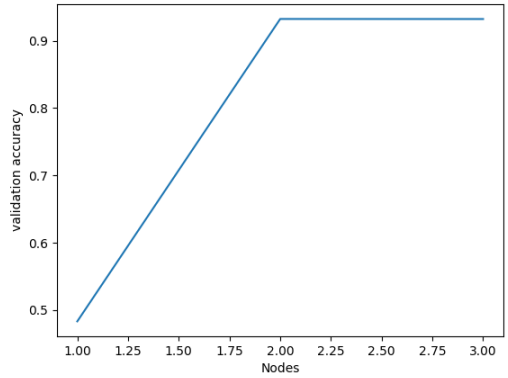
Layers	Training Loss	Training Accuracy	Test Loss	Test Accuracy	Validation Loss	Validation Accuracy
1 (Root)	0.0087	0.9911	0.00992	0.98	0.5109	0.4832
2	0.0001	1.0	0.00873	0.9888	0.0507	0.9324
3	0.0001	1.0	0.0085	0.9899	0.0507	0.9323



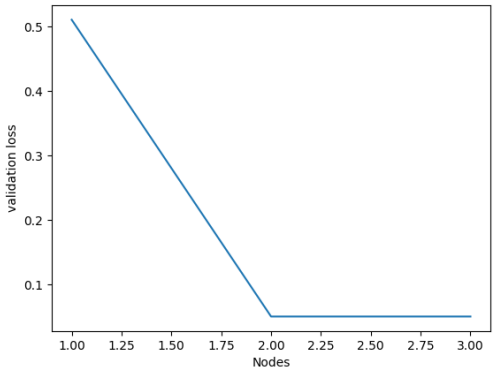
Training Accuracy



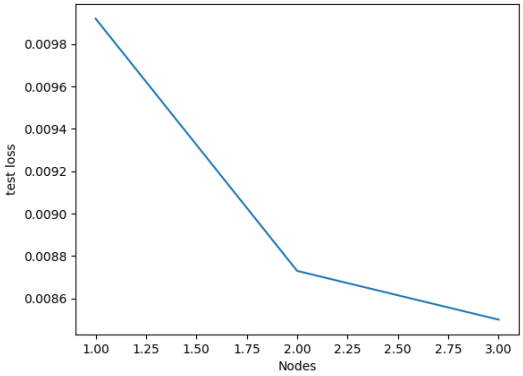
Training Loss



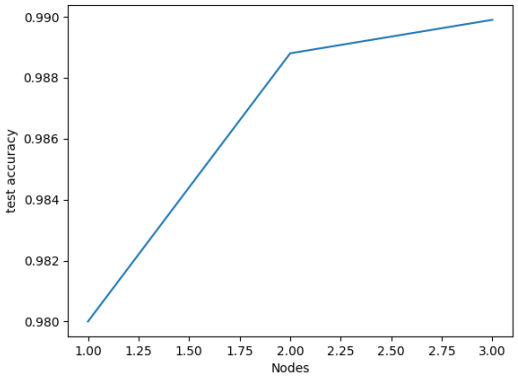
Validation Accuracy



Validation Loss



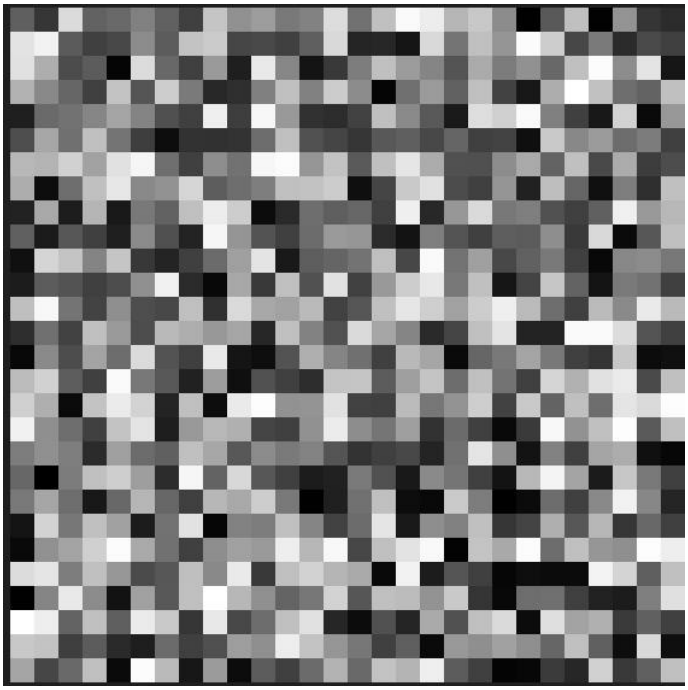
Test Accuracy



Test Loss

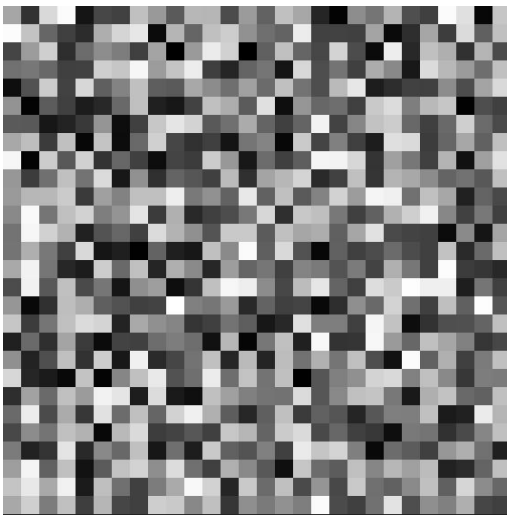
2. **Stopping Training to improve generalization:**
   a. Early stopping can be implemented to stop training, by checking for improvement. If improvements on increasing nodes is negligible, training can be stopped early to improve generalization.
   b. Maximum Depth can be initialized so we can stop growth of NN.

3. **Handling class imbalances at later nodes:**
   a. Oversampling less common classes in later nodes we can handle class imbalances. One example of this can be to duplicate minority class till desired size is achieved.
   b. Under sampling more common class can be utilized to handle class imbalances but this can result in significant reduction of dataset size.
4. **Visualizing what each node has learnt:**
   a. Parent Node:
      Weights of Parent Node were de-normalized (after removing the Wa and Wb weights) and saved as an image file 28*28 pixels.
      Below are the results:

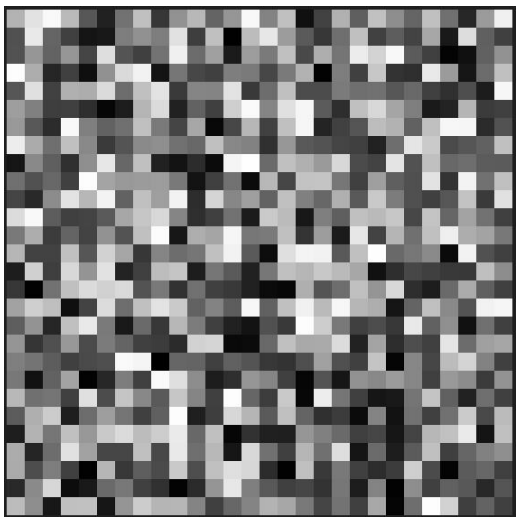


Parent Node De-Normalized Weight Matrix

   b. Child Nodes



Node 1 (+)



Node2(-)