ELL784 Machine Learning Assignment 2

Kashish Srivastava (2024AIB2289)

26th August 2024

1. Train ANN over MNIST Dataset and tuning Hyperparameters.

a. Data Preparation

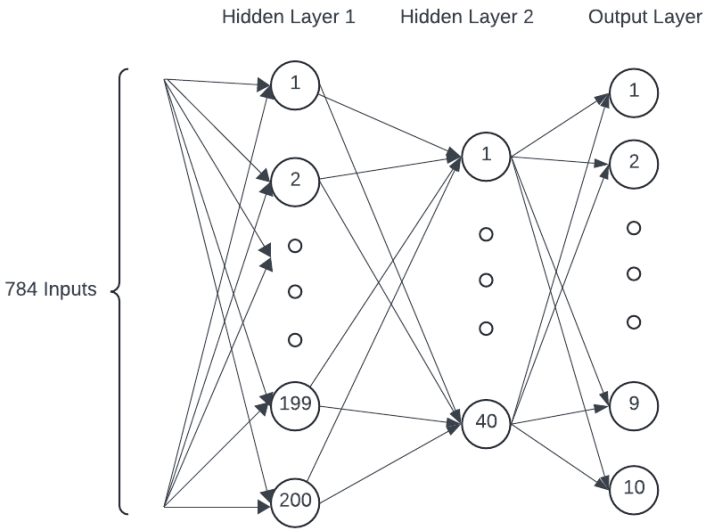
The dataset used in this assignment is a subset of MNIST database of handwritten digits has 48,000 data points. It was further divided into training (46000) and testing set (2000). Each data point in this dataset corresponds to 784-pixel values as columns and a Label for each row i.e. value in 0-9 which tells the digit which is represented by the row.

Labels for the dataset had to be One Hot Encoded to be used with the Model

b. Defining Model and Loss Functions

Model used for this specific task consists of:

784 Input Layer -> Hidden Layer 1 with 200 neurons (Sigmoid) -> Hidden Layer 2 with 40 neuron (Sigmoid) -> Output Layer with 10 neurons (SoftMax)



Architecture of Artificial Neural Network

Loss Function used for the Model = Mean Squared Error (MSE)

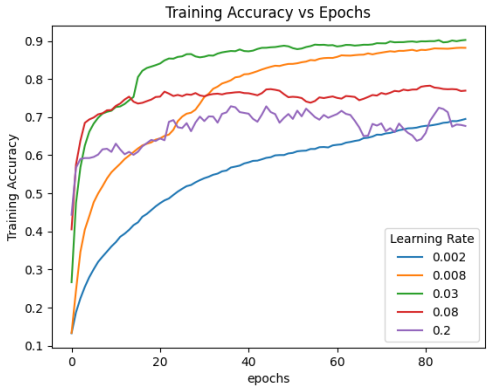
c. K-Fold Cross Validation

K- Fold Cross Validation was used for model evaluation and hyperparameter tuning. In this, the Training Dataset (46k points) was split into K subset. The model is trained on

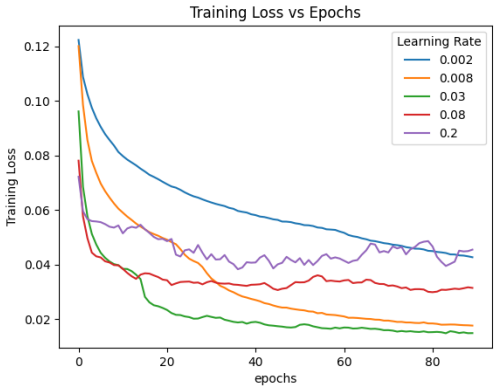
K-1 folds and validated on the remaining fold. This process was repeated K times, with each fold being used as the validation set once. In this report K was set to 10.
This process ensure model is not overly dependent on a particular train – test split, providing best estimate for generalizability.

d. GridSearch in one variable (Learning Rate)

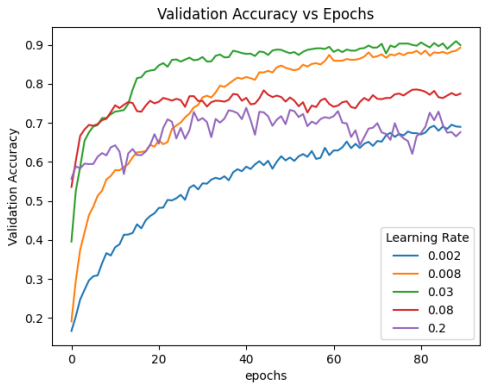
GridSearch was used to systematically search for the optimal hyperparameter configuration by evaluating the model's performance across a grid (in this case 1-D array) of hyperparameter (Learning Rate).
Model was tuned over a range of values for Learning Rate:
Learning Rate = [0.002,0.008,0.03,0.08,0.2]



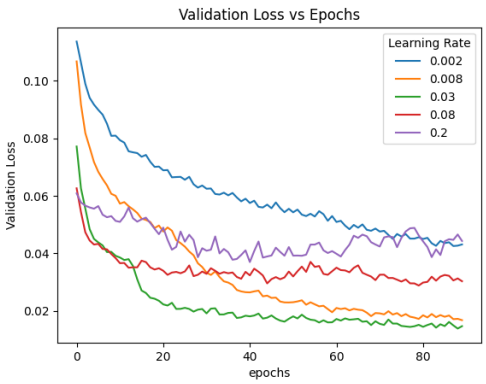
Training Accuracy vs Epochs



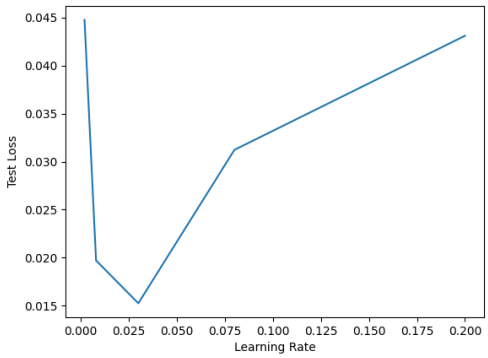
Training Loss vs Epochs



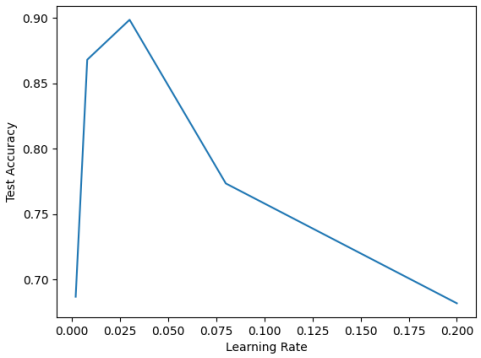
Validation Accuracy vs Epochs



Validation Loss vs Epochs



Test Loss vs Learning Rate



Test Accuracy vs Learning Rate

Epochs = 90

Learning Rate	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Test Loss	Test Accuracy
0.002	0.0427	0.695	0.0425	0.695	0.044	0.687
0.008	0.0163	0.8821	0.0168	0.892	0.0197	0.868
0.03	0.0148	0.90	0.0139	0.90925	0.0152	0.898
0.08	0.02988	0.782	0.0288	0.7855	0.031	0.773
0.2	0.0382	0.728	0.0370	0.7385	0.043	0.682

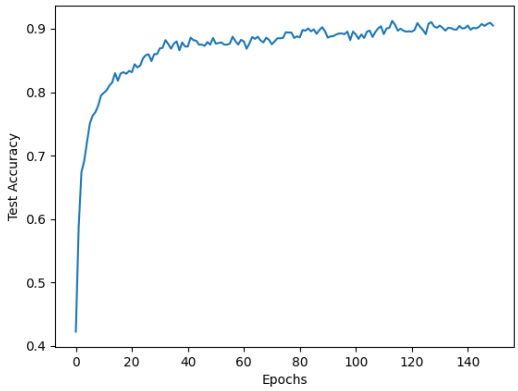
These results were observed by running mnist.py script

e. Training the ANN on dataset

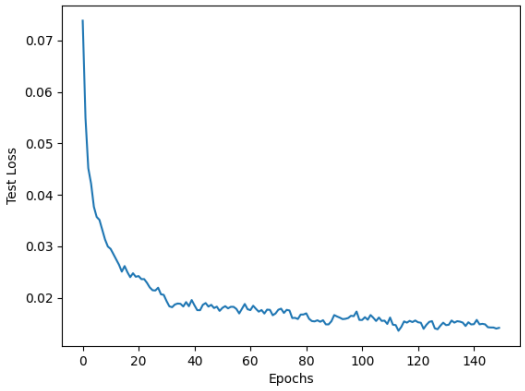
The ANN was trained on the dataset for **150 epoch** with **learning rate = 0.03**.

Observations from the training the model are as follows:

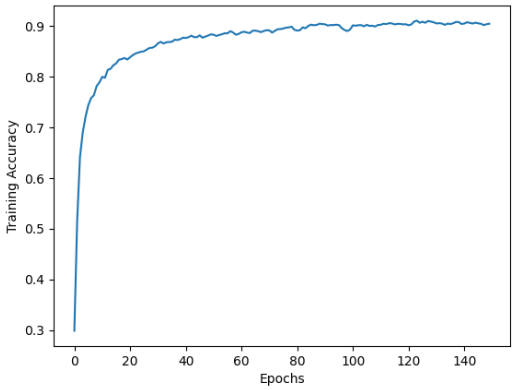
Metric	Value
Training Accuracy	0.911
Training Loss	0.0135
Validation Accuracy	0.9125
Validation Loss	0.014



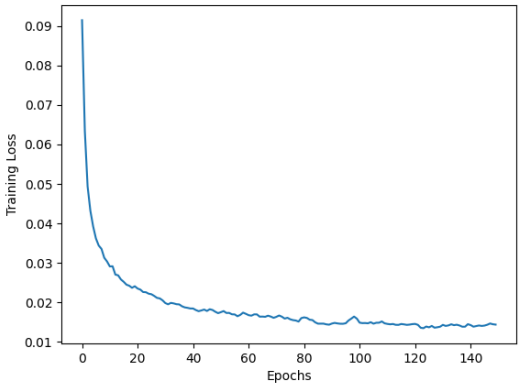
Validation Accuracy vs Epochs



Validation Loss vs Epochs



Training Accuracy vs Epochs

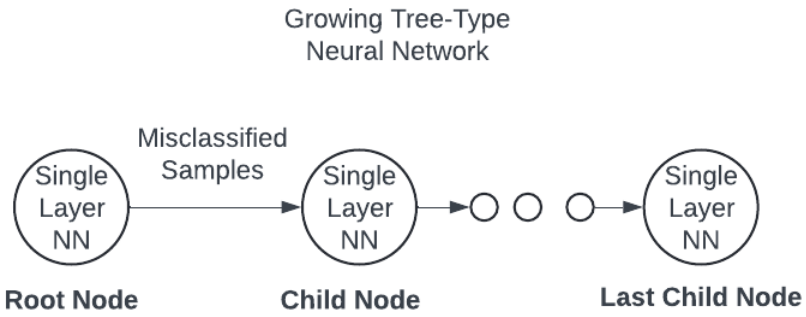


Training Loss vs Epochs

2. Making a Growing Tree-Type Neural Network

a. Growing Tree-Type Neural Network Definition:

It is a hierarchical ANN where each node represents a ANN. The structure of network grows incrementally, with new child node being added to address misclassified sample by parent node.



Growing Tree-Type Neural Network Architecture

Dataset used for this assignment is MNIST dataset. Labels for the dataset had to be One Hot Encoded to be used with the Model. Loss Function used with the model = **Mean Squared Error (MSE)**

b. Training the model:

Each Layer of Tree NN was trained for 100 epochs with learning rate = 0.03, max depth = 10.

All misclassified samples from parent nodes were pushed down to child nodes and were corrected. **Training/Testing Accuracy** values are **cumulative**, whereas **Training/Testing Loss** values are at **individual depth levels**. By running treeNN.py script, observed values from training and testing the model are as follows:

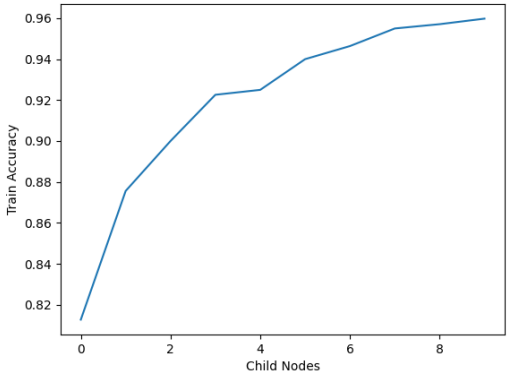
Node	Training Loss	Training Accuracy (Top-Down)	Test Loss	Test Accuracy (Bottom- up)
1 (Root)	0.0707	0.8127	0.075	0.952
2	0.0807	0.8756	0.0669	0.882
3	0.0839	0.900	0.0587	0.767
4	0.08391	0.9226	0.054	0.7055
5	0.0864	0.925	0.0441	0.574
6	0.0842	0.94	0.0380	0.496
7	0.088	0.9464	0.0293	0.378
8	0.0865	0.955	0.02396	0.305
9	0.0926	0.9571	0.0172	0.213
10	0.0887	0.9598	0.010	0.1245

```
Training Tree Node at depth:  1
e:\GITHUB\MERE_WALE\ELL784\Assignment2\NeuralNetwork\activations.py:6:
  sig = 1 + np.exp(-z)
Train_error=0.0707369847875612, Train_accuracy=0.782825
Incorrect Samples 8613
Training Tree Node at depth:  2
Train_error=0.0807087374880833, Train_accuracy=0.3402995471960989
Incorrect Samples 5718
Training Tree Node at depth:  3
Train_error=0.08394869013821885, Train_accuracy=0.19639734172787687
Incorrect Samples 4583
Training Tree Node at depth:  4
Train_error=0.08391368267304895, Train_accuracy=0.22037966397556186
Incorrect Samples 3557
Training Tree Node at depth:  5
Train_error=0.0864128239135356, Train_accuracy=0.03008152937868991
Incorrect Samples 3450
Training Tree Node at depth:  6
Train_error=0.08423568137742575, Train_accuracy=0.21130434782608695
Incorrect Samples 2723
Training Tree Node at depth:  7
Train_error=0.08809578703227028, Train_accuracy=0.09474843922144693
Incorrect Samples 2465
Training Tree Node at depth:  8
Train_error=0.0865241309156371, Train_accuracy=0.16511156186612577
Incorrect Samples 2056
Training Tree Node at depth:  9
Train_error=0.09263882685543261, Train_accuracy=0.04231517509727627
Incorrect Samples 1969
Training Tree Node at depth:  10
Train_error=0.08875370190782374, Train_accuracy=0.062468257998984256
Incorrect Samples 1846
```

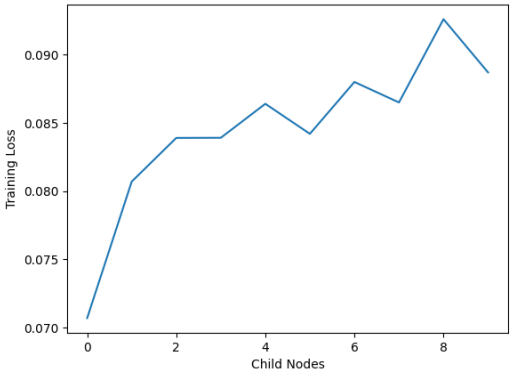
Training Error/Accuracy at individual level (Node 1 -> Node 10)

```
Test_error=0.010111425759223582, Test_accuracy=0.1245
Test_error=0.017253859106317247, Test_accuracy=0.213
Test_error=0.023967955781006255, Test_accuracy=0.305
Test_error=0.029342082565057267, Test_accuracy=0.378
Test_error=0.0380696976547617, Test_accuracy=0.496
Test_error=0.04412376374659193, Test_accuracy=0.574
Test_error=0.054031308986198234, Test_accuracy=0.7055
Test_error=0.05873250226737075, Test_accuracy=0.767
Test_error=0.06690971493757444, Test_accuracy=0.882
Test_error=0.07556930459175375, Test_accuracy=0.952
```

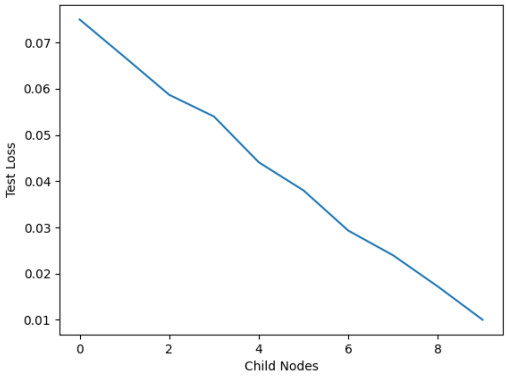
Test Error at each level, Cumulative Test Accuracy (Node 10 ->Node 1)



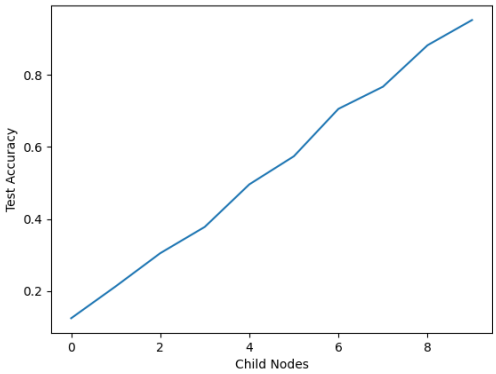
Training Accuracy



Training Loss (Individual Level)



Test Loss



Test Accuracy

3. **Stopping Training to improve generalization:**
   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

4. **Handling class imbalances at later nodes:**
   By oversampling less common classes in later nodes we can handle class imbalances. One example of this can be to duplicate minority class and skew/rotate/flip the image.