CS6910 : Assignment 1

Sarthak Vora - EE19B140 & Akshat Joshi - EE19B136

March 17, 2022

1 Introduction

In this assignment, we implement 3 tasks based on Deep Neural Networks. The 3 tasks are summarized below-

- Task 1 - Function Approximation
- Task 2 - Classification

  Task 2a - Binary Classification

  Task 2b - Multi-Class Image Classification

We are given 3 different datasets for each of the tasks. We primarily used ***PyTorch*** framework to develop the model along with secondary libraries i.e `numpy`, `matplotlib`, `scikit-learn`. The details about the models & training in different tasks is explained in the subsequent sections. Code inferencing details are given below.

2 Task 1 : Function Approximation

A **neural network** is a universal function approximator. In this task, we are given a 2-d input **(x1, x2)** and 1-d output **y**. We need to derive a model such that we get an approximate estimate of output **y** through a function -

$$y \approx f(x1, x2) \tag{1}$$

2.1 Implementation

Pattern mode of learning is followed while training i.e **batch size = 1**. We use `torch.optim.SGD()` optimizer in **PyTorch** as Generalized delta rule optimizer. The values of hyperparameters used for this task are as follows:

Learning rate	$2e-6$
Weight update	Generalized delta rule
Loss function	Mean Squared Error
Momentum	0.9
Nodes in hidden layer 1 (l1)	8
Nodes in hidden layer 2 (l2)	4
Activation function for hidden layers	tanh
Activation function for output layer	Linear

2.2 Dataset

Function approximation dataset `func_app1.csv` has **701** examples. The data was split into **train-val-test** with split ratio **7:2:1**. The dataset splitting code is present in `dataset_split.py`

2.3 Metrics

Training Set Average Loss: 55.16
Test Set Average Loss: 48.64
Validation Set Average Loss: 6.17

2.4 Plots

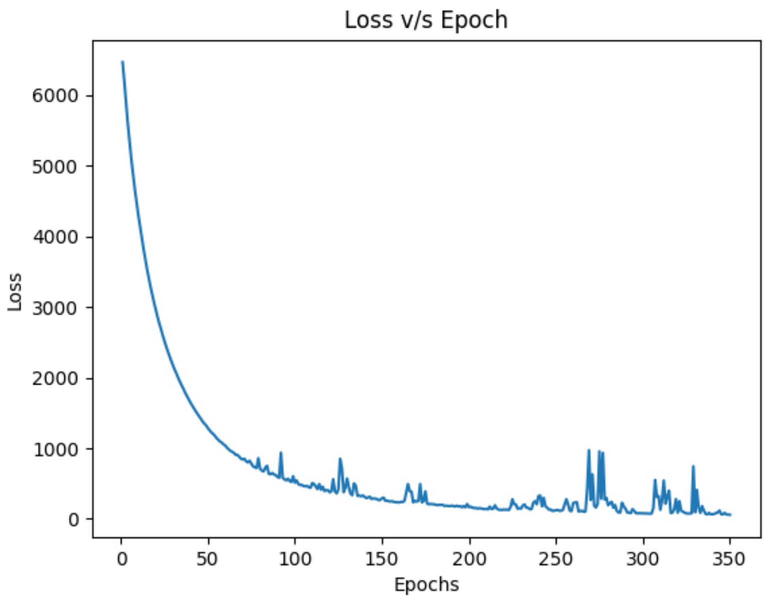


Figure 1: Loss v/s Epoch

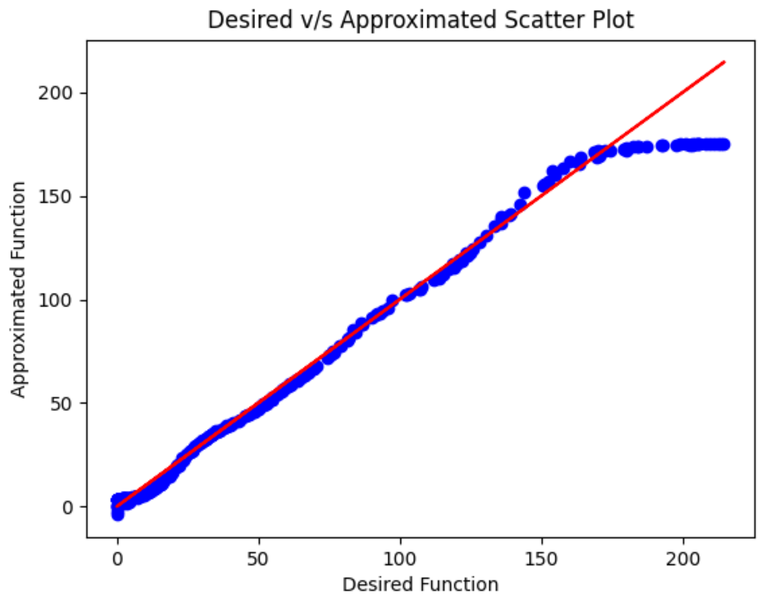
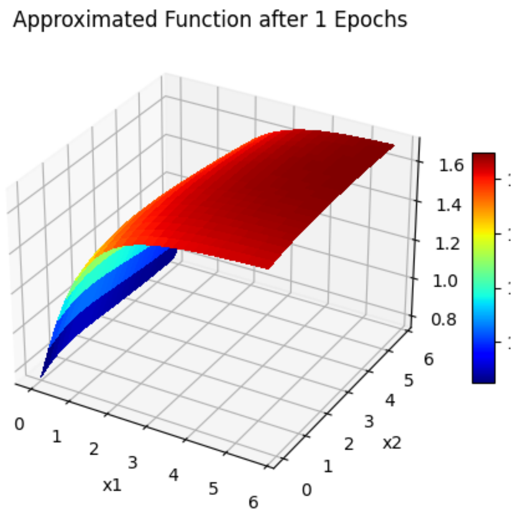
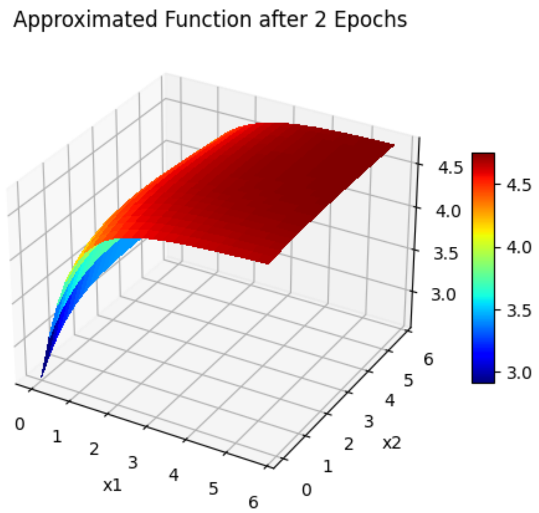


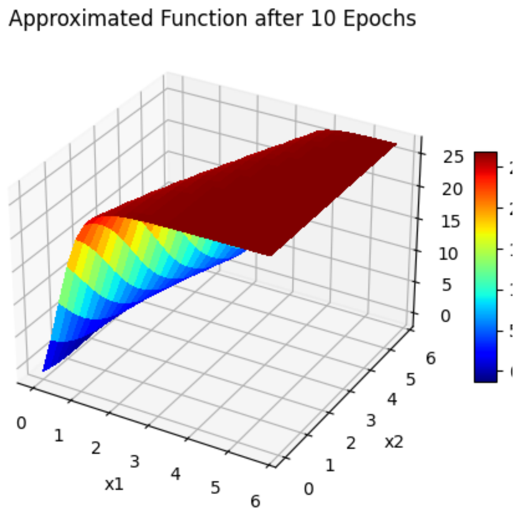
Figure 2: Scatter plot of Desired v/s Appoximated output



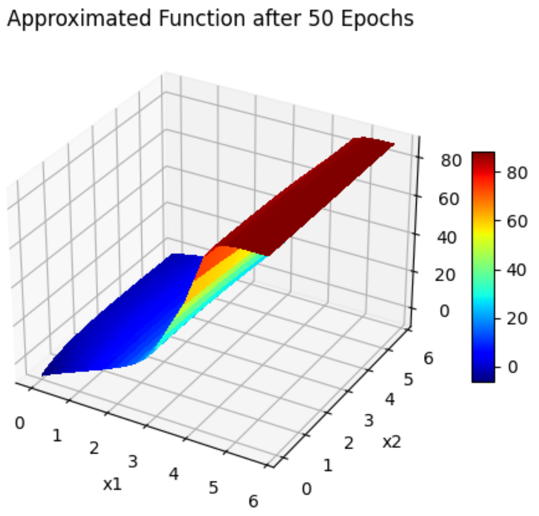
(a) Approximated function at the end of Epoch 1



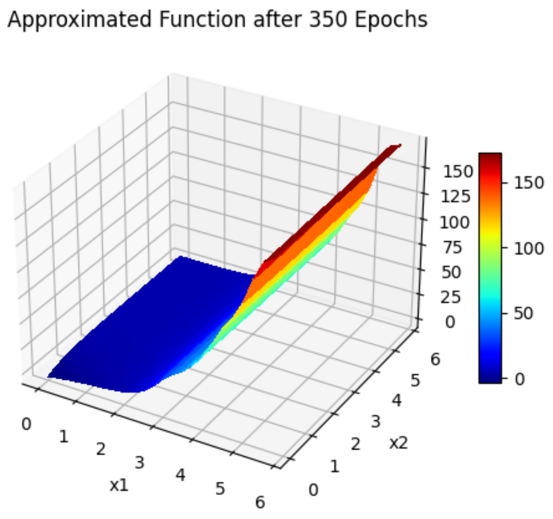
(b) Approximated function at the end of Epoch 2



(c) Approximated function at the end of Epoch 10



(d) Approximated function at the end of Epoch 50



(e) Approximated function at the end of Epoch 350

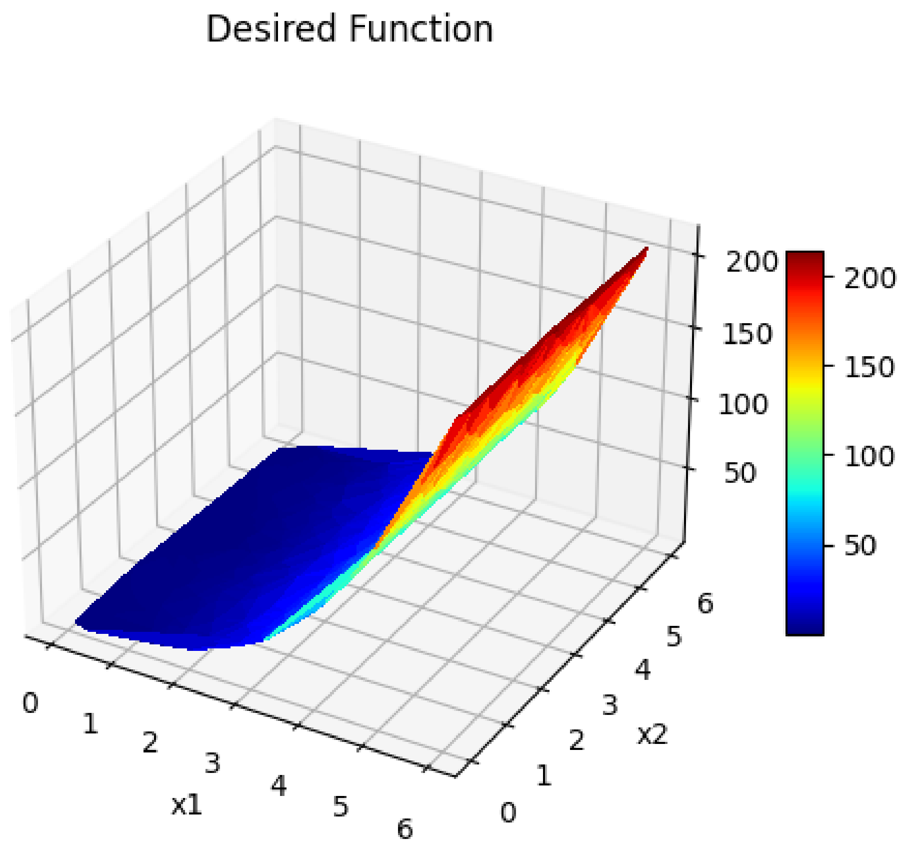


Figure 4: Desired Function

2.5 Observation

- As epochs increase, the similarity between the approximated and desired output increases.
- If learning rate is high, oscillations are observed in the average loss v/s epoch plot.
- If learning rate is low, the training becomes slower but the average loss v/s epoch curve sees less oscillations, and requires more number of epochs to train.
- For a given learning rate, as epochs increase the oscillations in the average error plot goes up as seen in Figure 1.
- Most part of the scatter plot lies along the y=x line indicating good fit.

3 Task 2a - Binary Classification

3.1 Implementation

A fully connected network with 2 hidden layers has been implemented for this task.

For training the model, pattern mode of learning (batch size = 1) is used along with the generalized delta rule. The following parameters were used for training the model:

Learning rate	0.0001
Weight update	Generalized delta rule
Loss function	Cross-Entropy Error
Momentum	0.7
Nodes in hidden layer 1 (l1)	4
Nodes in hidden layer 2 (l2)	2
Activation function for hidden layers	tanh
Activation function for output layer	Sigmoid

This has been implemented using the SGD optimizer built into PyTorch.

3.2 Dataset

The model was trained on the **train_t25.csv** file which consists of **1200** training examples. It was then validated on the **dev_t25.csv** file which has **100** examples.

Also, the model was tested on **test_t25.csv** file. The output of the test is stored in **test_output.csv** file.

3.3 Test Set Output

The test set output is attached in this link

3.4 Metrics

On the training set the following metrics were obtained: (99.08% percent accuracy)

Average Loss: 0.04273941702228816
True Positives: 621
True Negatives: 568
False Positives: 6
False Negatives: 5
Accuracy: 0.9908333333333333
Precision: 0.9904306220095693
Recall: 0.9920127795527156
F1 score: 0.9912210694333599

On the validation set the following metrics were obtained: (98% accuracy)
Average Loss: 0.07842664239229634
True Positives : 49
True Negatives: 49
False Positives: 1
False Negatives: 1
Accuracy: 0.98
Precision: 0.98
Recall: 0.98
F1 score: 0.98

3.5 Plots

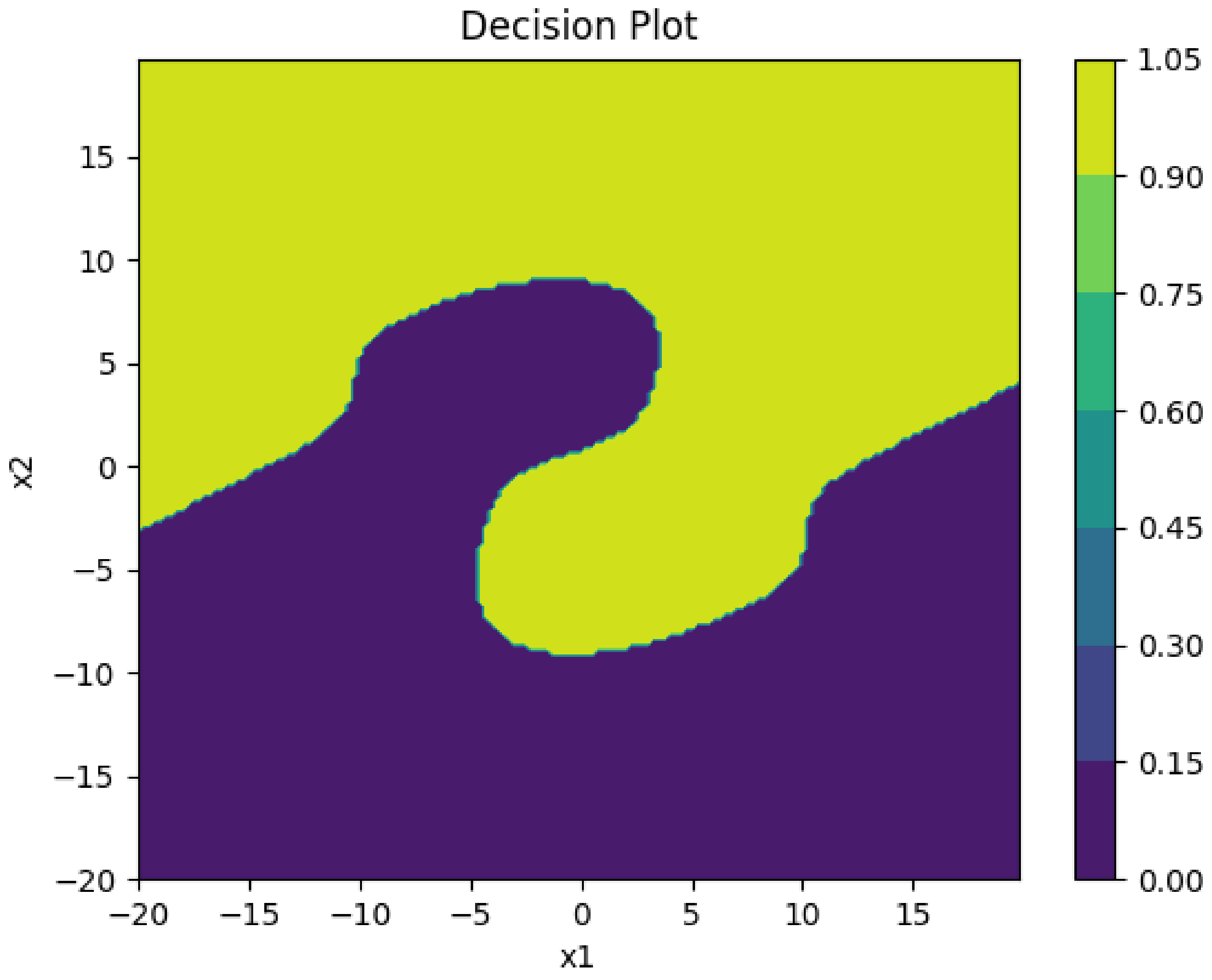
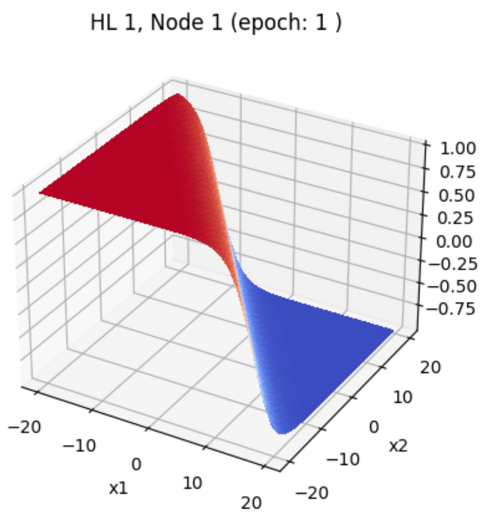
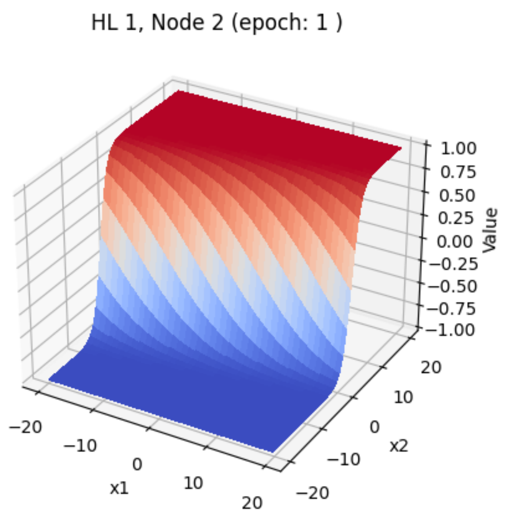


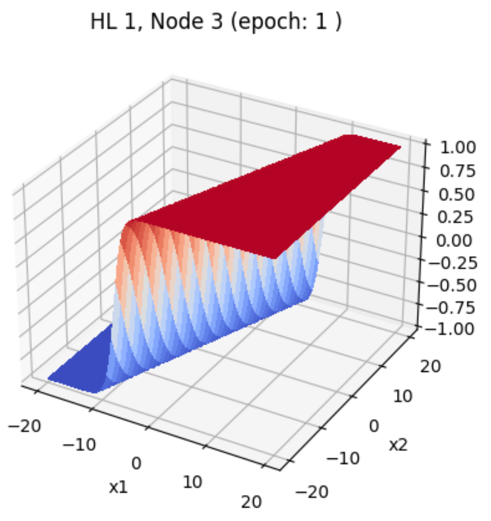
Figure 5: Decision Plot showing the decision boundary



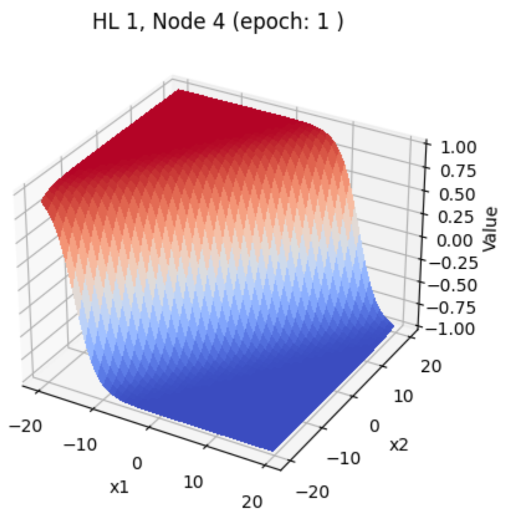
(a) Plot for Hidden Layer 1, Node 1 (epoch: 1)



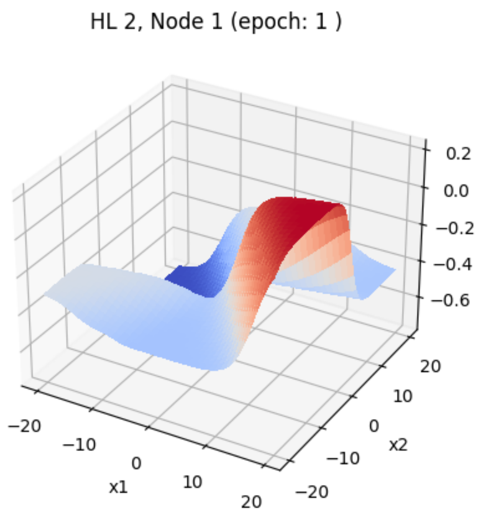
(b) Plot for Hidden Layer 1, Node 2 (epoch: 1)

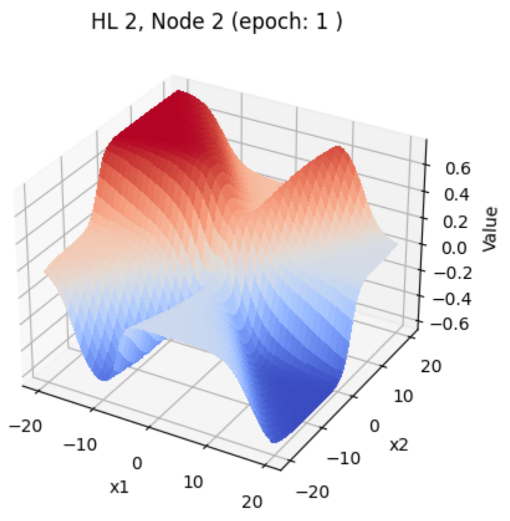


(c) Plot for Hidden Layer 1, Node 3 (epoch: 1)



(d) Plot for Hidden Layer 1, Node 4 (epoch: 1)

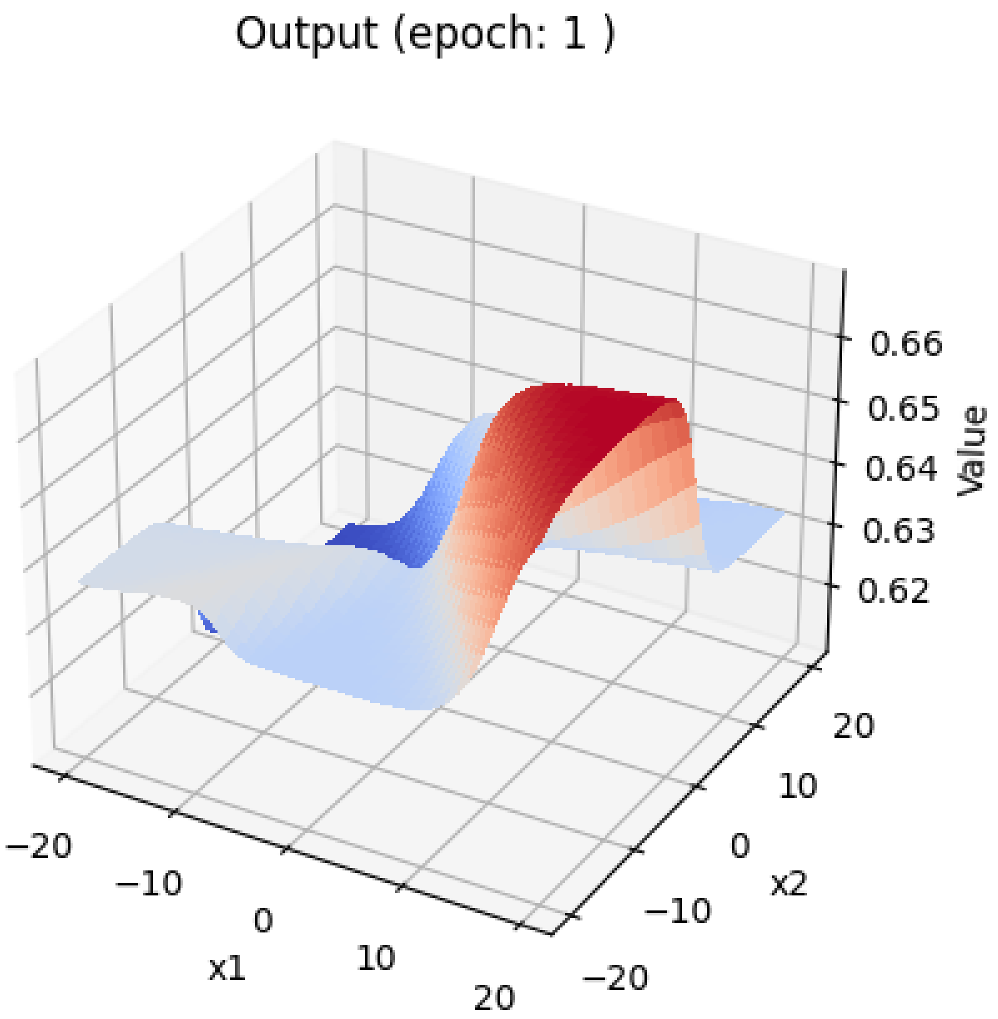


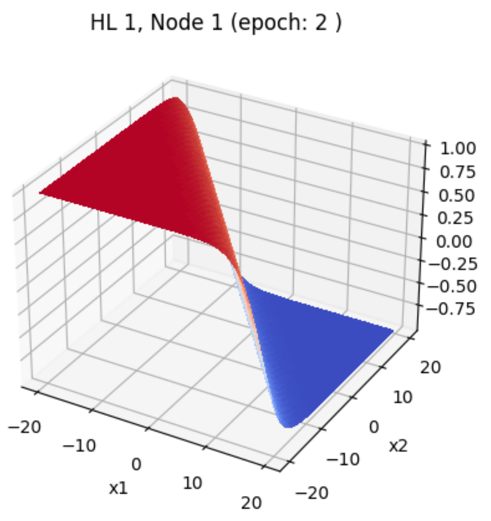


(e) Plot for Hidden Layer 2, Node 1 (epoch: 1)

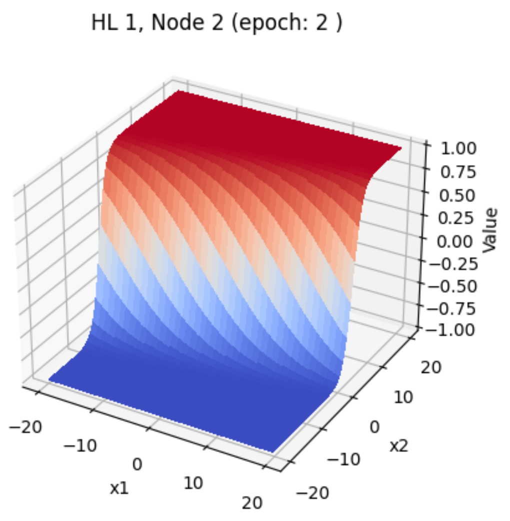
(f) Plot for Hidden Layer 2, Node 2 (epoch: 1)

Figure 7: Plot for Output Layer (epoch: 1)

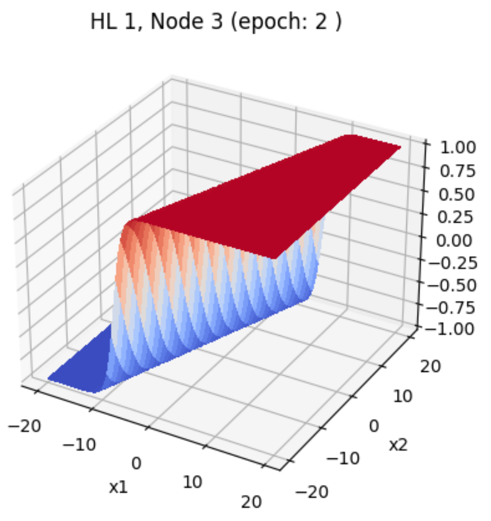




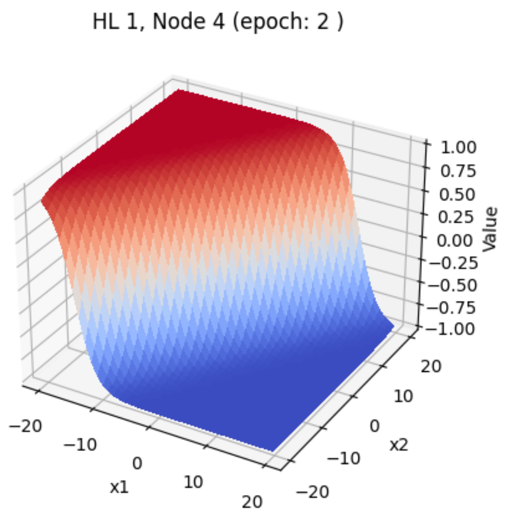
(a) Plot for Hidden Layer 1, Node 1 (epoch: 2)



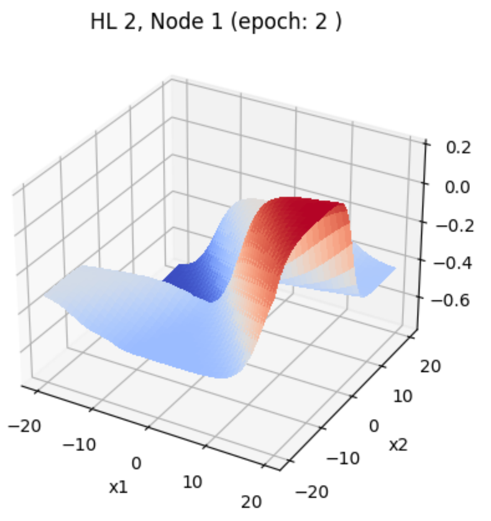
(b) Plot for Hidden Layer 1, Node 2 (epoch: 2)

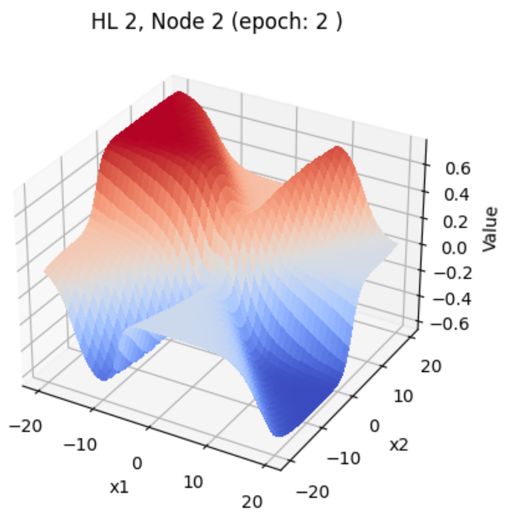


(c) Plot for Hidden Layer 1, Node 3 (epoch: 2)



(d) Plot for Hidden Layer 1, Node 4 (epoch: 2)

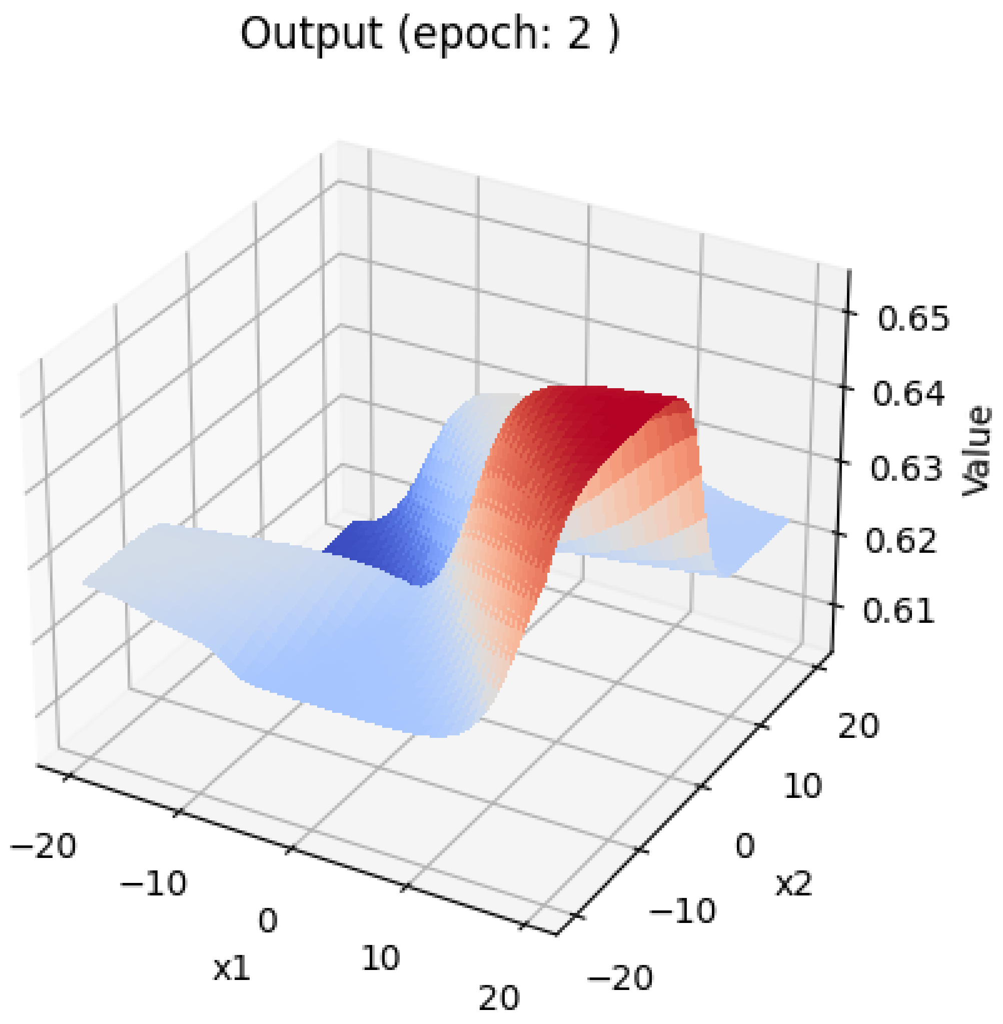


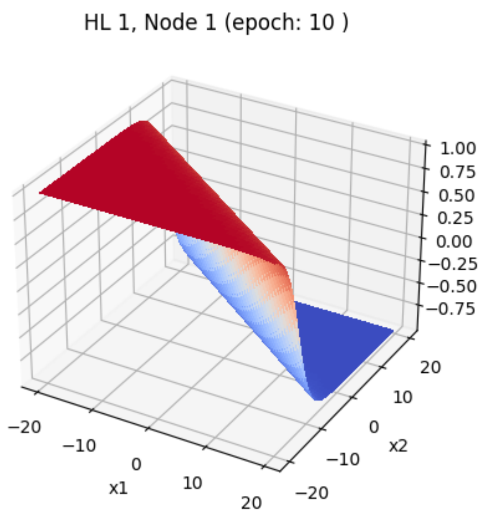


(e) Plot for Hidden Layer 2, Node 1 (epoch: 2)

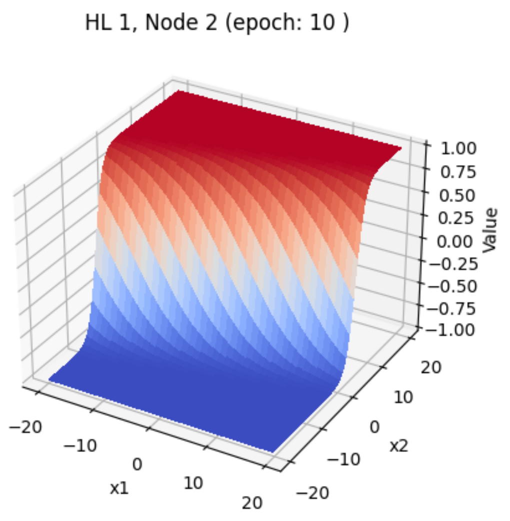
(f) Plot for Hidden Layer 2, Node 2 (epoch: 2)

Figure 9: Plot for Output Layer (epoch: 2)

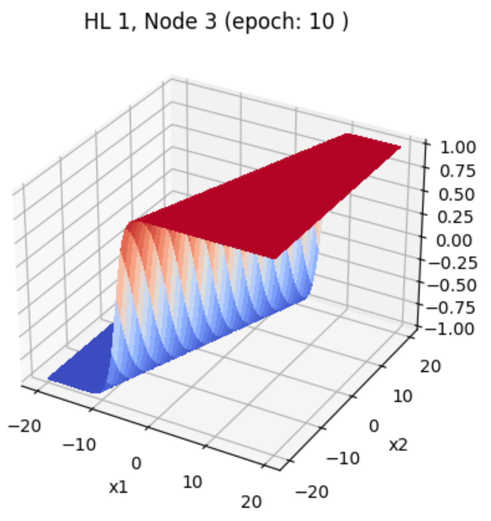




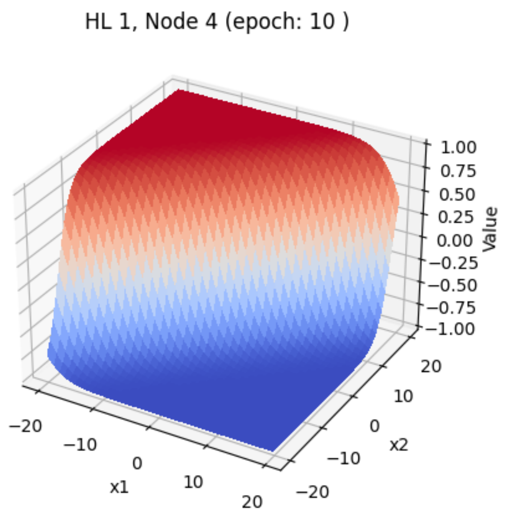
(a) Plot for Hidden Layer 1, Node 1 (epoch: 10)



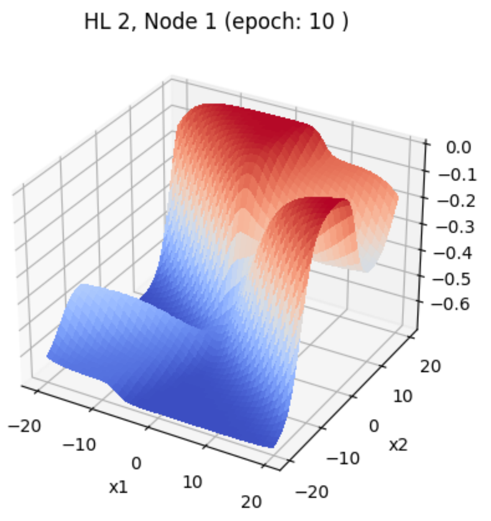
(b) Plot for Hidden Layer 1, Node 2 (epoch: 10)



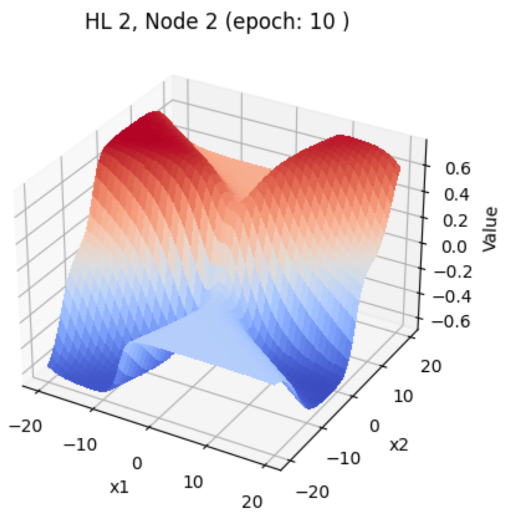
(c) Plot for Hidden Layer 1, Node 3 (epoch: 10)



(d) Plot for Hidden Layer 1, Node 4 (epoch: 10)

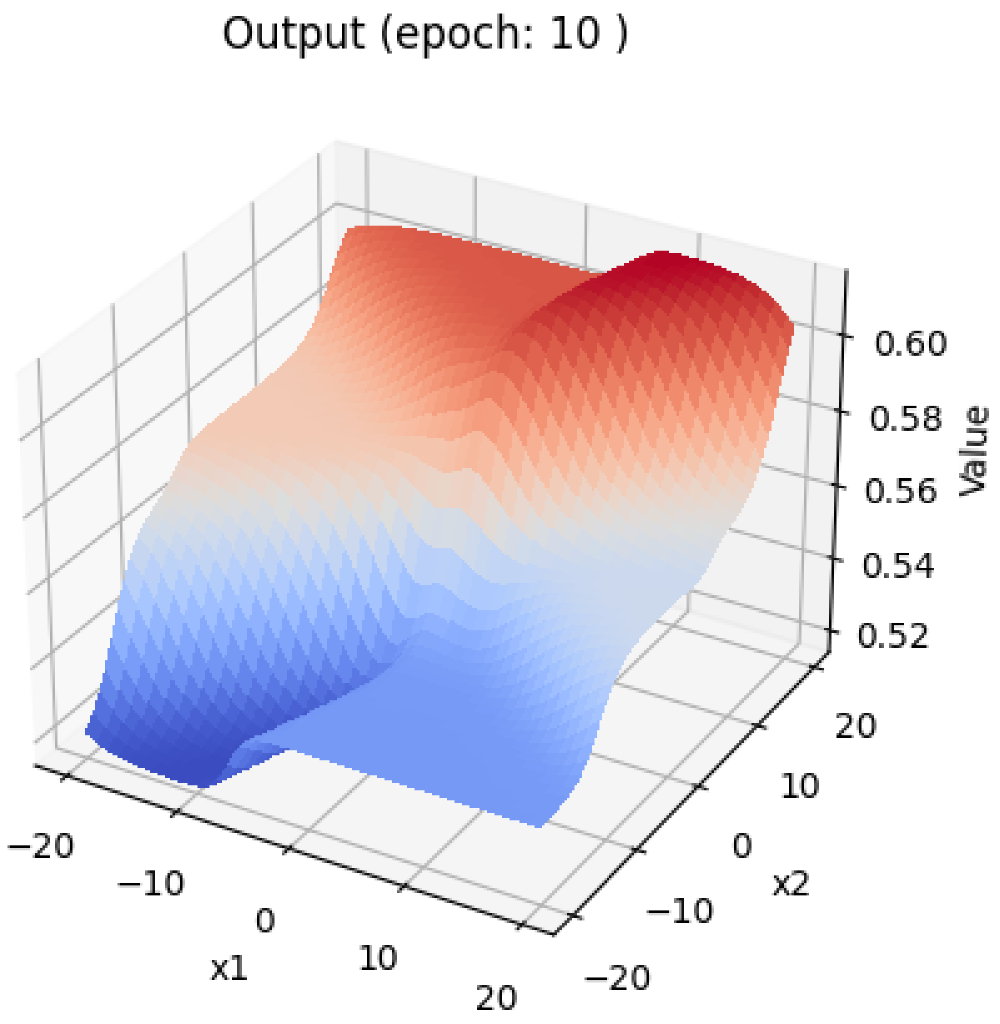


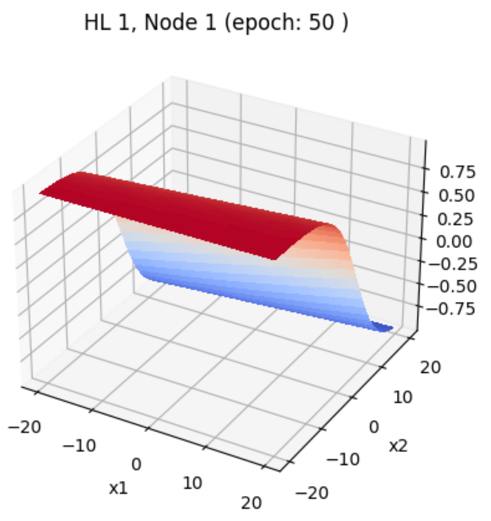
(e) Plot for Hidden Layer 2, Node 1 (epoch: 10)



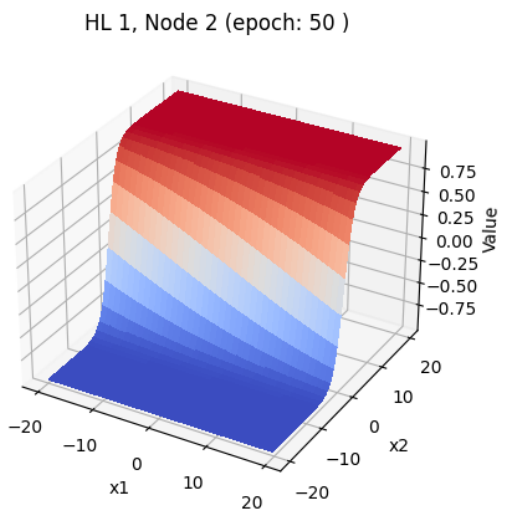
(f) Plot for Hidden Layer 2, Node 2 (epoch: 10)

Figure 11: Plot for Output Layer (epoch: 10)

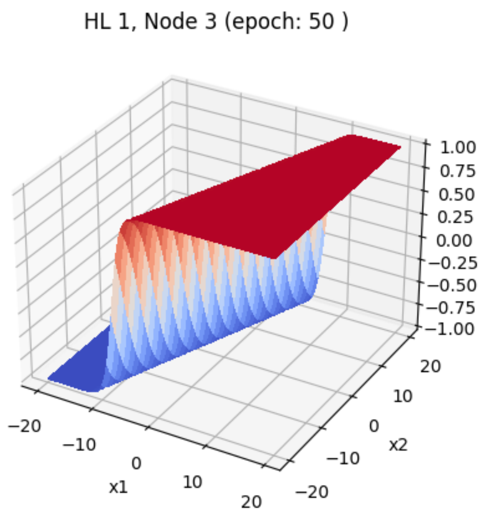




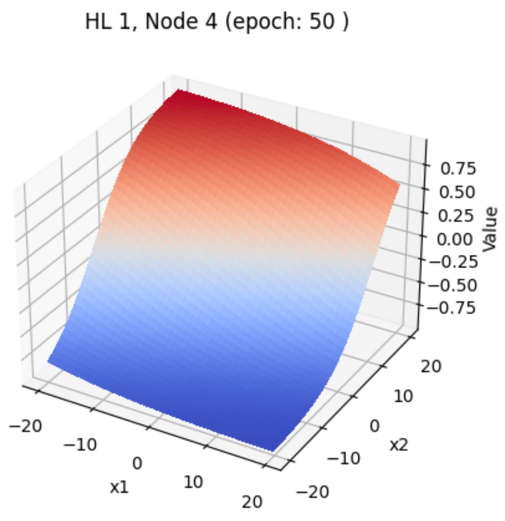
(a) Plot for Hidden Layer 1, Node 1 (epoch: 50)



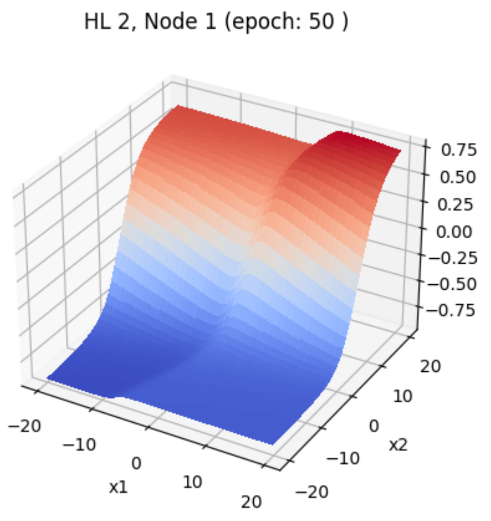
(b) Plot for Hidden Layer 1, Node 2 (epoch: 50)

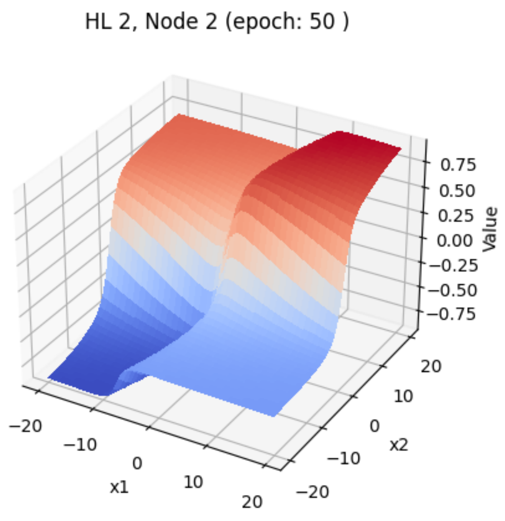


(c) Plot for Hidden Layer 1, Node 3 (epoch: 50)



(d) Plot for Hidden Layer 1, Node 4 (epoch: 50)

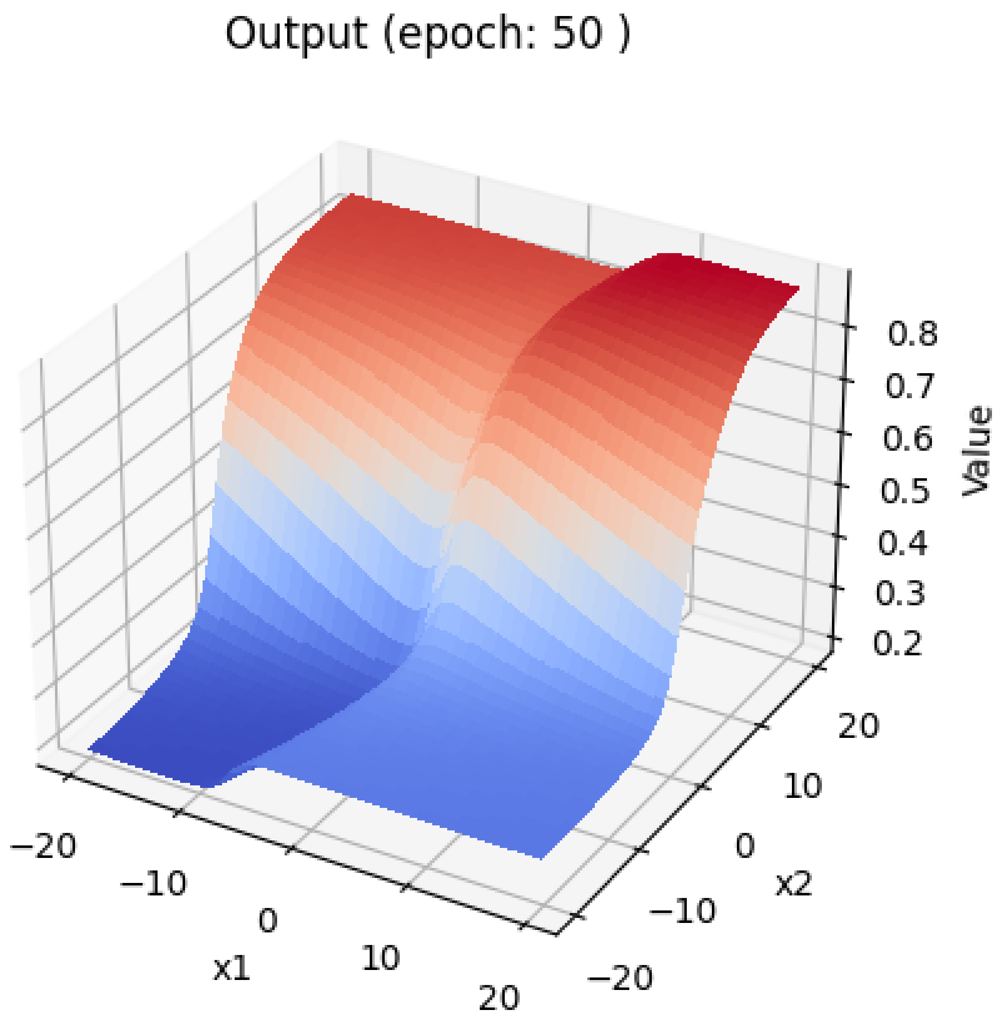


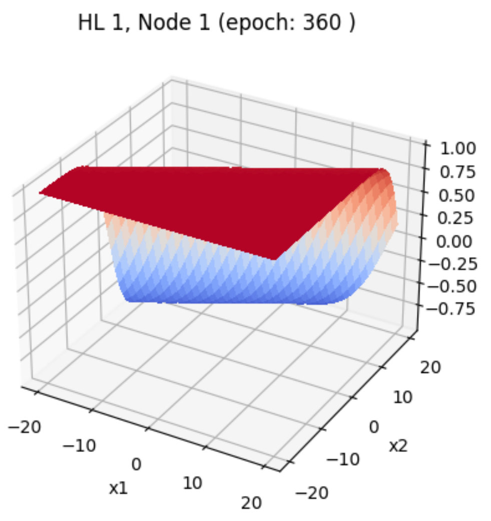


(e) Plot for Hidden Layer 2, Node 1 (epoch: 50)

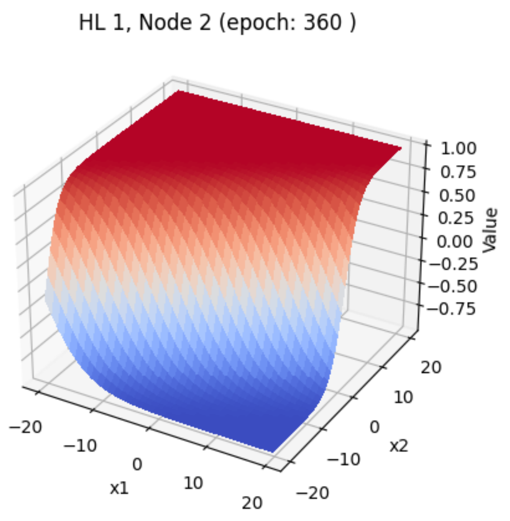
(f) Plot for Hidden Layer 2, Node 2 (epoch: 50)

Figure 13: Plot for Output Layer (epoch: 50)

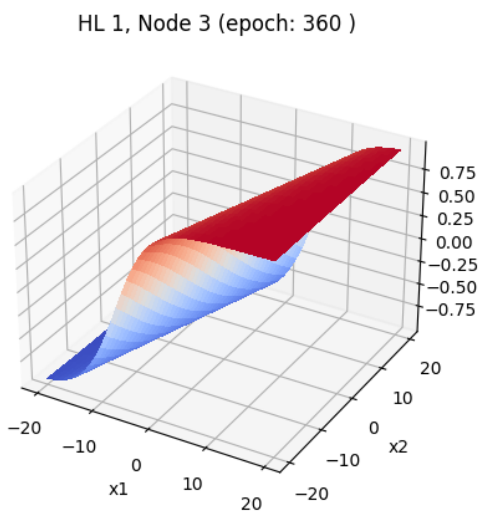




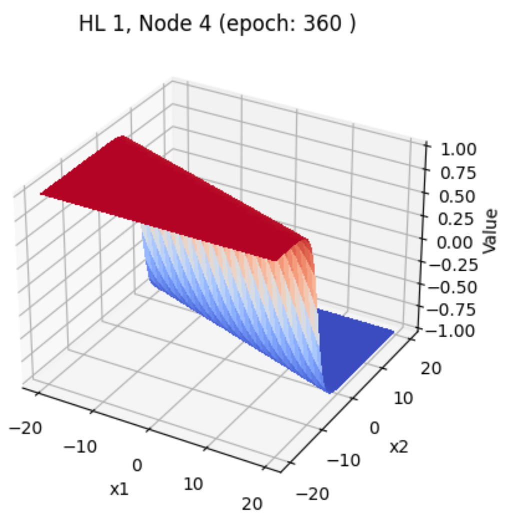
(a) Plot for Hidden Layer 1, Node 1 (epoch: 360)



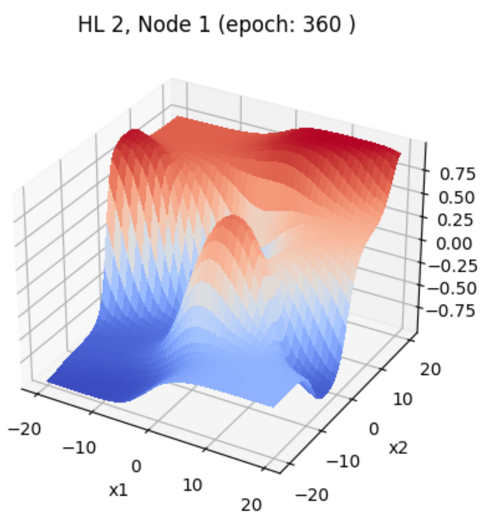
(b) Plot for Hidden Layer 1, Node 2 (epoch: 360)

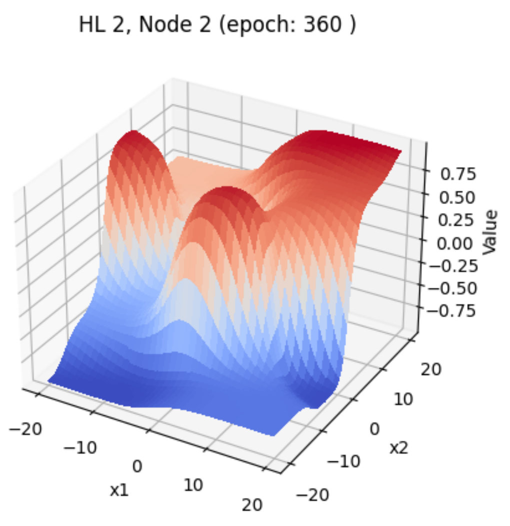


(c) Plot for Hidden Layer 1, Node 3 (epoch: 360)



(d) Plot for Hidden Layer 1, Node 4 (epoch: 360)





(e) Plot for Hidden Layer 2, Node 1 (epoch: 360)

(f) Plot for Hidden Layer 2, Node 2 (epoch: 360)

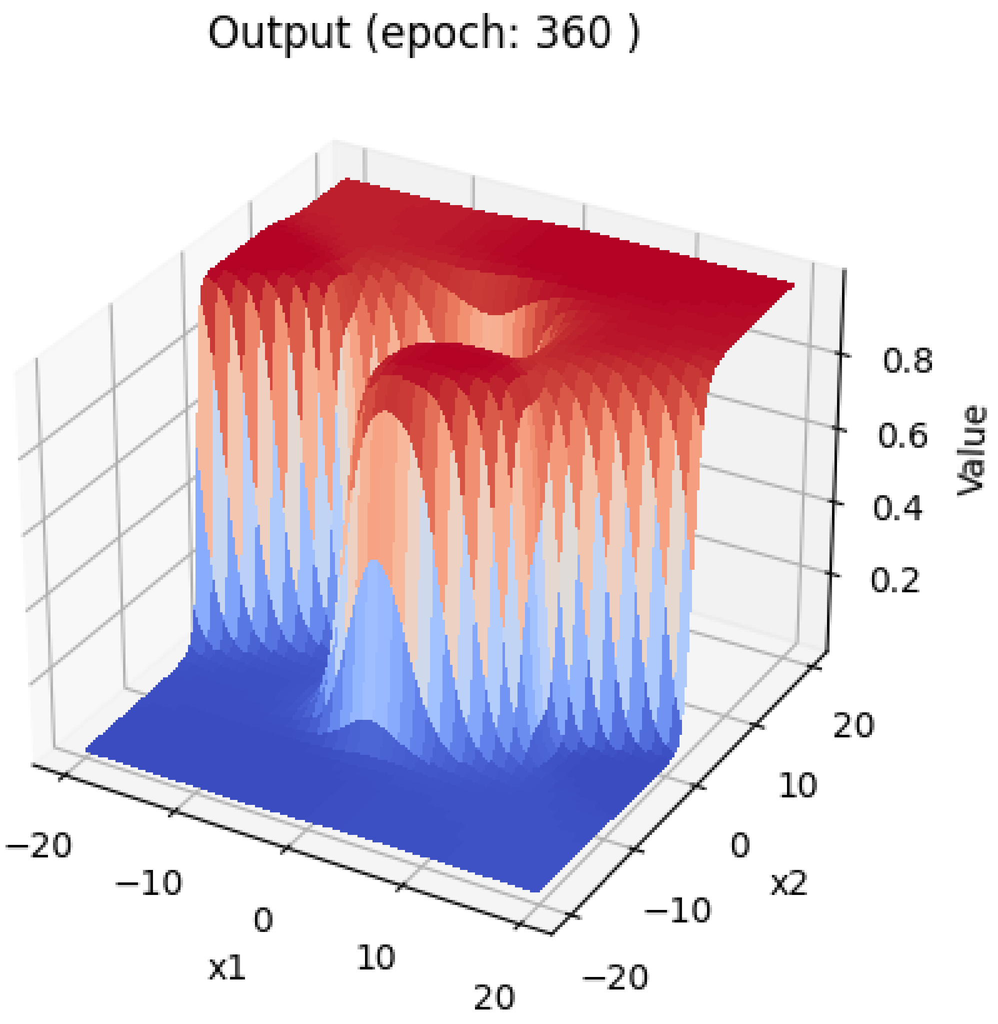


Figure 15: Plot for Output Layer (epoch: 360)

3.6 Observation

- As epochs increase the output layer becomes closer and closer to the actual decision boundary.
- As epochs increase, the hidden layer surfaces evolves to fit the correct decision boundary
- The decision boundary is symmetric. That is rotating the plot by 180 degrees does not change the decision boundary.
- As the number of nodes in the hidden layers increased the model had a tendency to overfit on the training data.

4 Task 2b : Multi-class Image Classification

In this task, we perform single-label multi-class classification of image data. The 5 classes are -

{ Street:0, Coast:1, Forest:2, Tallbuilding:3, Opencountry:4 }

4.1 Implementation

A fully-connected Neural Network with 60 input features and 5 output features was implemented. We use 3 different Weight update rules - i) **Generalized Delta rule** ii) **Delta rule** iii) **Adam Optimizer**
The weights are initialized to the same value for all the 3 weight update rules using `torch.manual_seed()` in **PyTorch**
The common hyperparameters used for all the 3 submethods are as follows -

Learning rate	$2e-5$
Loss function	Cross-Entropy
Momentum	0.9
Nodes in hidden layer 1 (l1)	256
Nodes in hidden layer 2 (l2)	8
Activation function for hidden layers	Tanh
Activation function for output layer	Softmax

The ideal number of **epochs**, the **learning rate** and number of **hidden nodes** for all the 3 methods was tuned via **RayTuning optimization** in `ray` library. A random combination of hyperparameters was taken by `ray.tune()` and validation accuracy was found across all the possible methods. The combination with **best validation accuracy** was considered and the above hyperparameters were chosen based on that. However, due to random noise, the above values may not be ideal. The metrics from above hyperparameters are shown below. Intuitively, epoch$_{Delta}$ > epoch$_{GeneralizedDelta}$ > epoch$_{Adam}$

4.2 Dataset

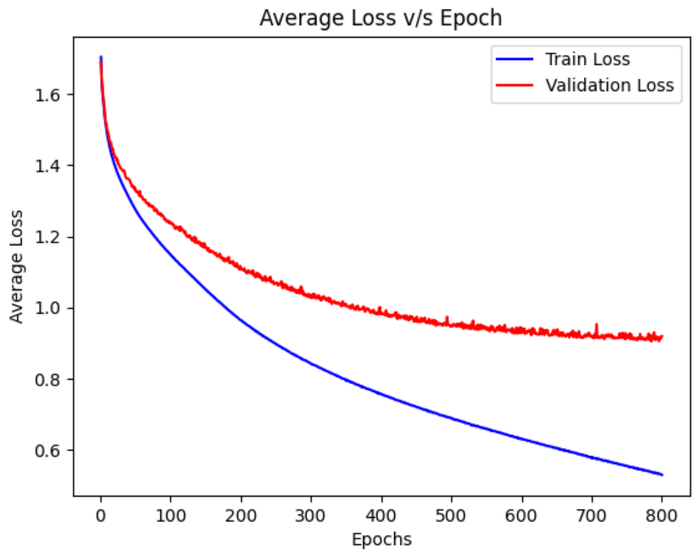
Image data in `image_data_dim60.txt` has **2688** datapoints with each row of dimension $60x1$. The dataset has been split into **train-val** with split ratio **80:20**. Corresponding to every image data point, there exists a label in `image_data_labels.txt`. For our group, the labels in [0, 1, 5, 6, 7] are to be considered out of the given 8 label classes.

4.3 Metrics

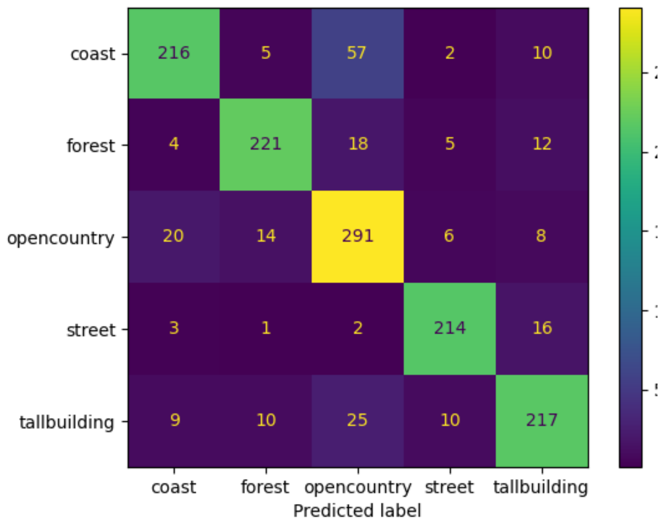
Weight Update	**Train Accuracy**	**Validation Accuracy**
Delta	83.02%	63.71%
Generalized Delta	94.56%	64.86%
Adam	97.56%	65.71%

4.4 Plots

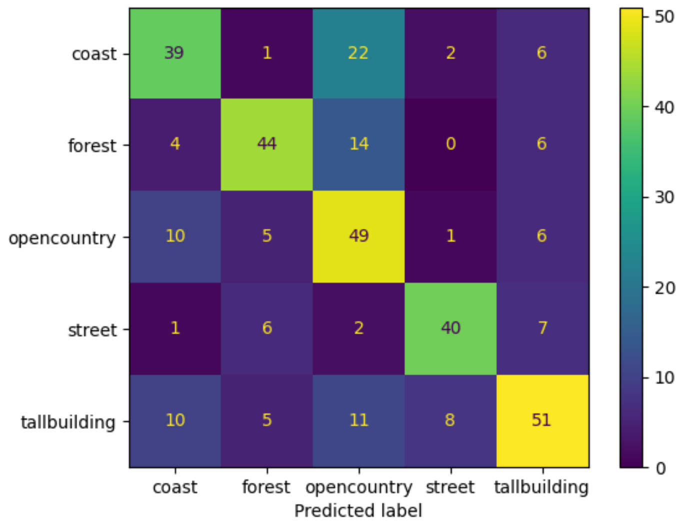
Delta Rule



(a) Loss v/s Epoch curve

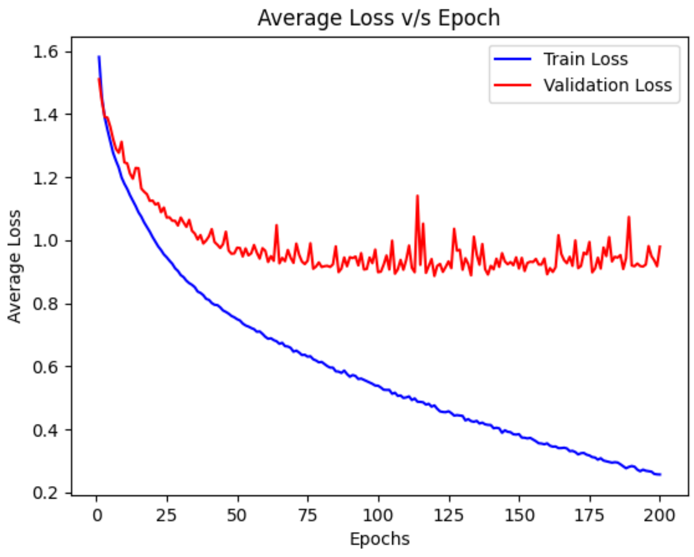


(b) Confusion matrix for training data

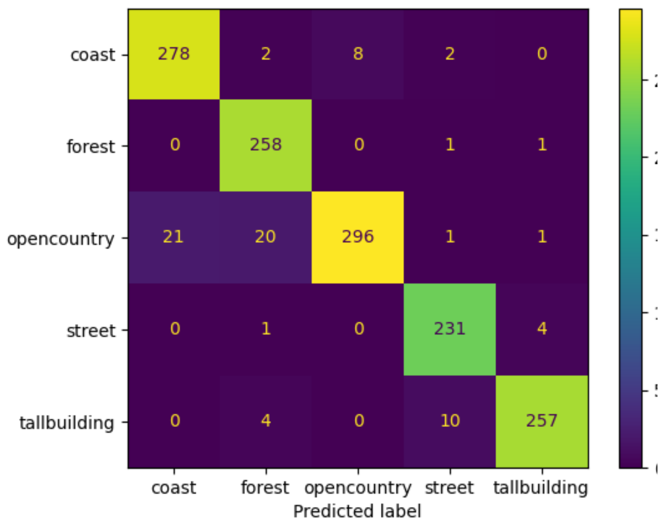


(c) Confusion matrix for validation data

Generalized Delta Rule

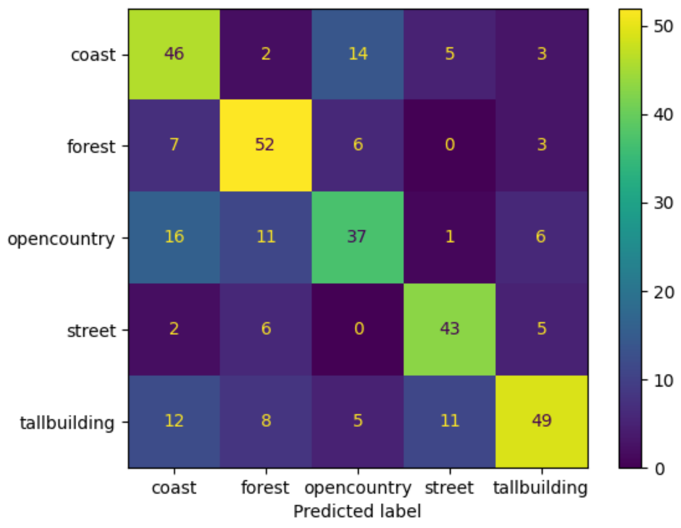


(a) Loss v/s Epoch curve



	coast	forest	opencountry	street	tallbuilding
coast	278	2	8	2	0
forest	0	258	0	1	1
opencountry	21	20	296	1	1
street	0	1	0	231	4
tallbuilding	0	4	0	10	257

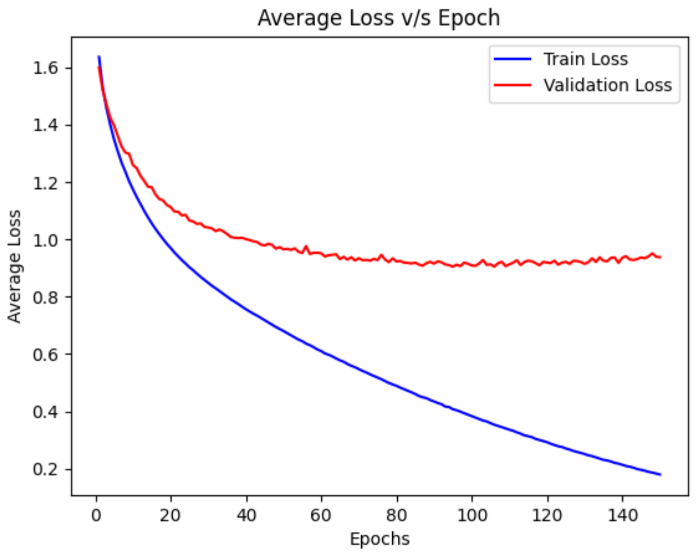
(b) Confusion matrix for training data



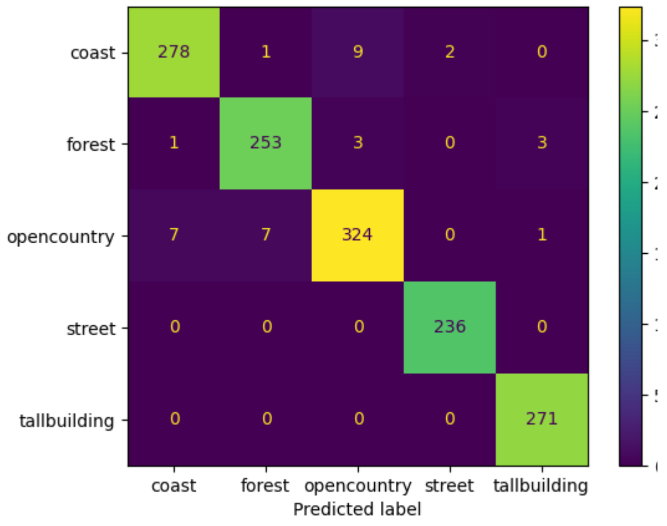
	coast	forest	opencountry	street	tallbuilding
coast	46	2	14	5	3
forest	7	52	6	0	3
opencountry	16	11	37	1	6
street	2	6	0	43	5
tallbuilding	12	8	5	11	49

(c) Confusion matrix for validation data

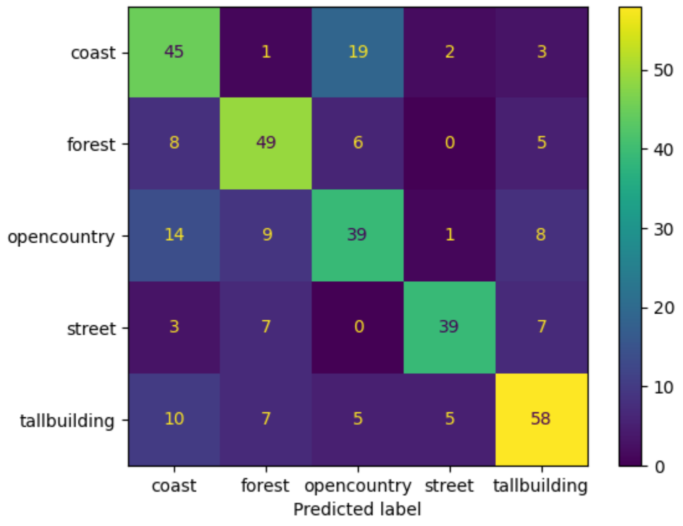
Adam Rule



(a) Loss v/s Epoch curve



(b) Confusion matrix for training data



(c) Confusion matrix for validation data

4.5 Observation

- The number of epochs for different rules are -

Weight Update	**Epochs**
Delta	800
Generalized Delta	200
Adam	150

- We observe that the number of epochs **decreases** as the **level of the optimizer** increases from Delta, Generalized Delta to Adam rule.
- From the plots above, **Adam rule** has the smoothest descent across all the weight update rules. However, the oscillations are slightly visible in Delta and Generalized Delta rule.
- The **irregularities** in the `loss v/s epoch` curve occur due to the oscillations of the optimizer around the point of minima of the loss function.
- We see that **left diagonal** of any confusion matrix above has more number of datapoints than any other grid element. This shows a prominent match for the output labels

---