Assignment 1

Group No 20

Tanay Dixit (EE19B123) , Atharva Chougule (CS19B016) , Kaustubh Miglani (CS19B060)

27th Feb , 2022

CS6910

1 Function Approximation

In this task we are asked to identify the approximate function using a neural network with given specifications. We use Multi Layer Feed Forward network with 2 hidden layers and $tanh()$ as activation. For training we make use of pattern mode training ie: training the network one sample at a time. In other words the mini batch size = 1.

We make use of the *pytorch* library for implementing the generalised delta rule optimizer. The algorithm corresponds to the *SGD* class with non zero momentum. We randomly initialize the weights . The tuneable hyperparameters are located in the third block of the $config.yaml$ file. We make use of random search to find the best set of parameters based upon the performance of the validation data.

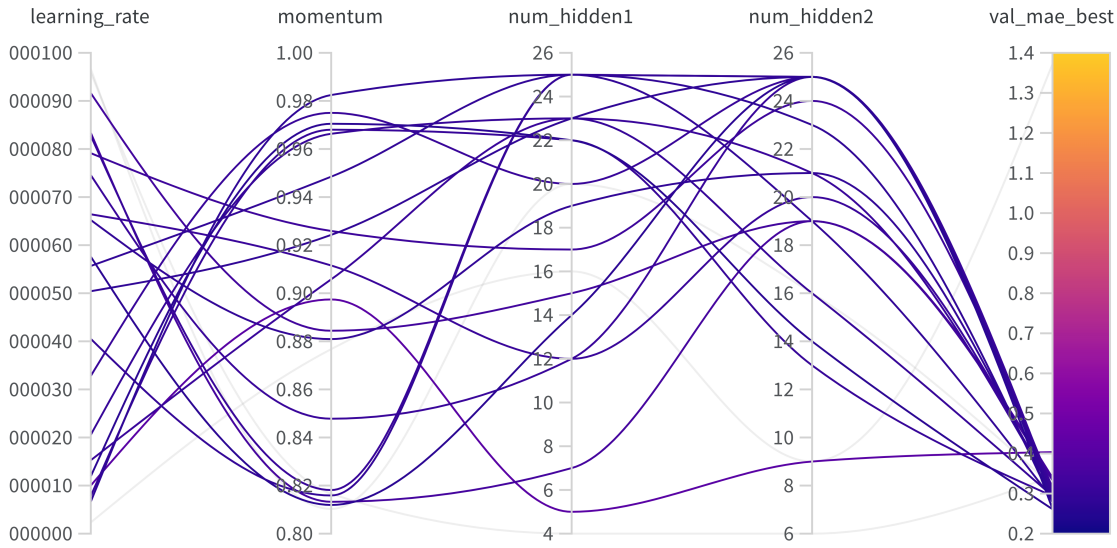
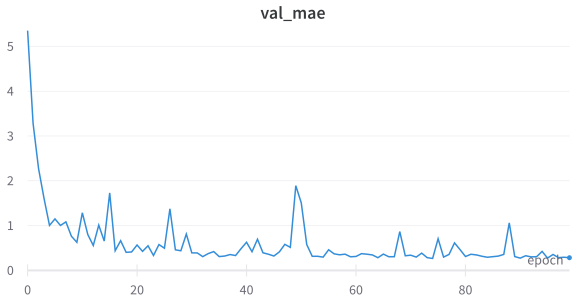
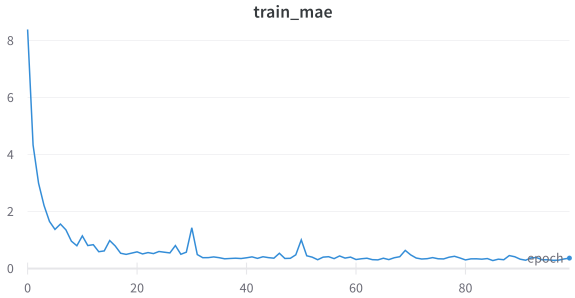


Figure 1: Hyperparameter Search

We make use of the optimal parameters in figure 1 and train the model. The train mean absolute error plot is as follows. We train our model using the Mean Squared Error loss for 100 epochs and validate it every epoch. We report the mean absolute error (MAE) for both train and validation data. The plots are as follows



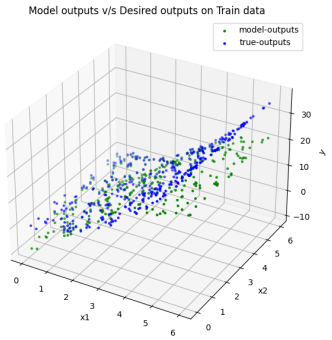
(a) Validation Error



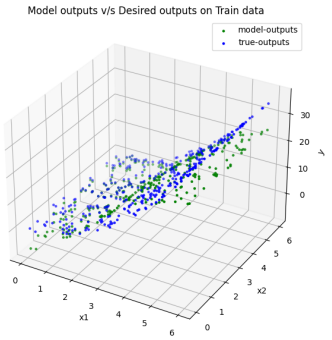
(b) Training Error

For epochs $1, 2, 10$ and 50 we plot the model outputs v/s the desired outputs.

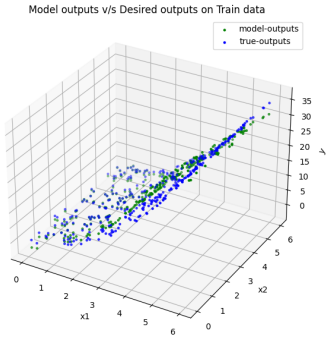
Observations: As the number of epochs increases, the training error follows a decreasing trend, the validation error also follows a general decreasing trend except at some places and seems to converge as the number of epochs keep on increasing. This converging error can be in some part attributed to the noise in the data.



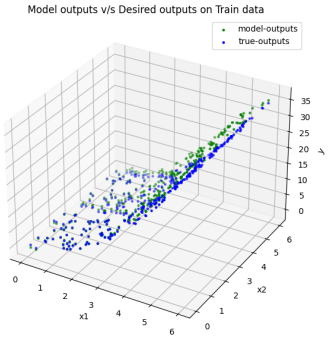
(a) Epoch 1



(b) Epoch 2



(c) Epoch 10



(d) Epoch 50

Figure 3: Desired function v/s Approximated function for various epochs

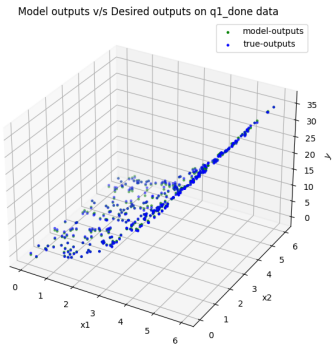


Figure 4: Desired function v/s Approximated function after training completed

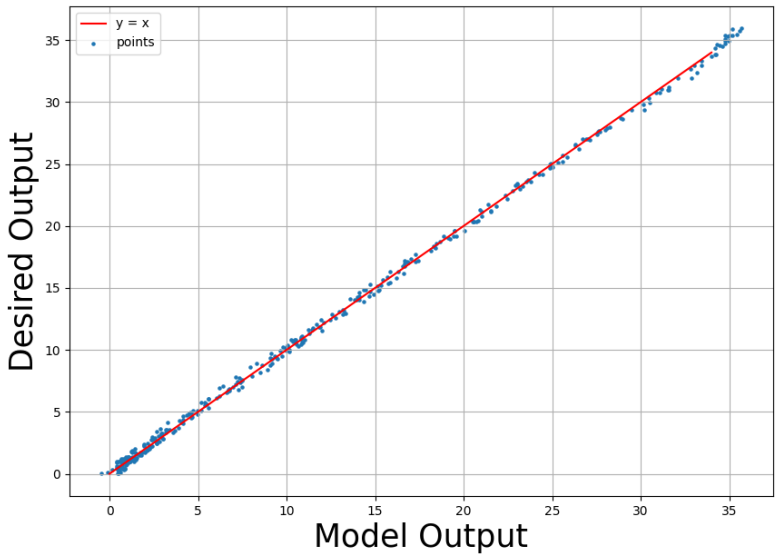


Figure 5: Caption

We also plot the scatter plot of the outputs after the model is trained. This can be found in figure 5.

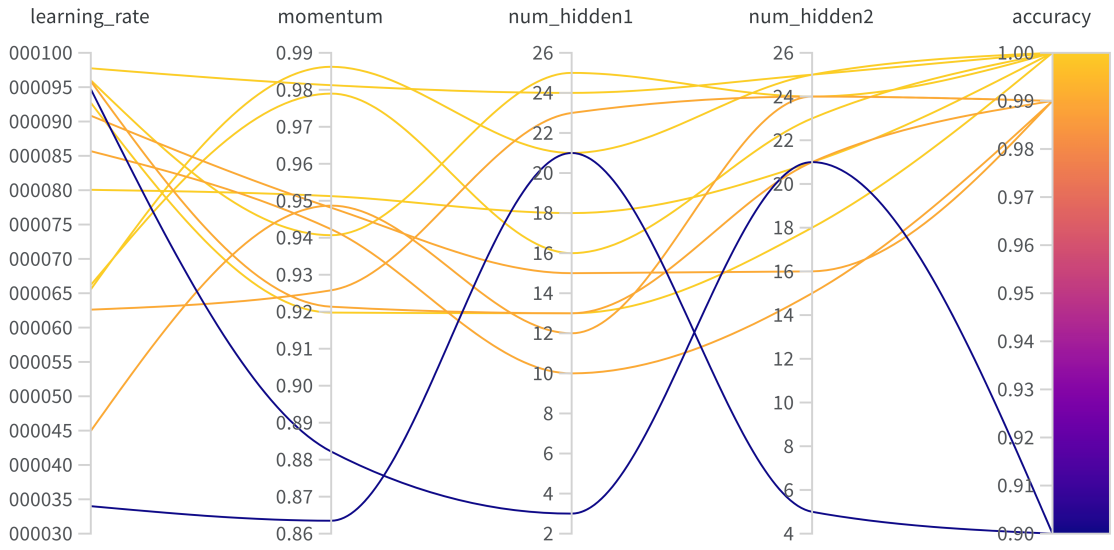


Figure 6: Hyperparameter Search

2 Classification Task for 2d data

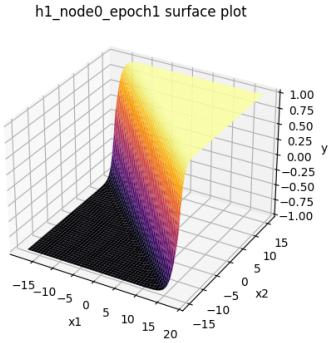
For this task we make use of the same model as before but change the loss to Binary cross entropy loss as the dataset had 2 target classes. We again make use of the validation data to find the best hyperparameters for the given dataset. The hyperparameter search can be found 6. We use the best parameters obtained and again train it using the generalised delta rule optimizer for 100 epochs.

2.1 Surface of nodes

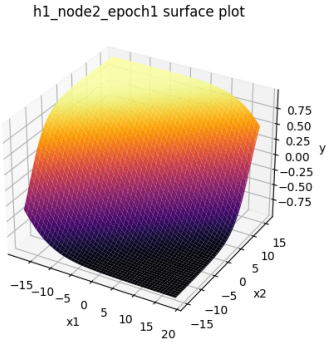
To analyse what each node learns and how it contributes to the final output we plot the surface output. We analyse how the surface changes epoch wise.
We have chosen 8 random nodes out of 13 for hidden layer 1 and also 8 random nodes out of 18 for hidden layer 2. We have also plotted the surface plot of output node from which we can see how the model learns as the number of epochs increase.

2.1.1 Epoch 1

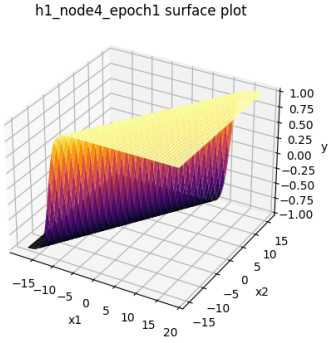
Hidden Layer 1: We randomly choose 8 nodes from hidden layer 1 and plot the following in Figure 7



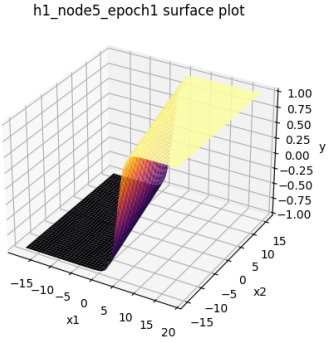
(a) Node 0



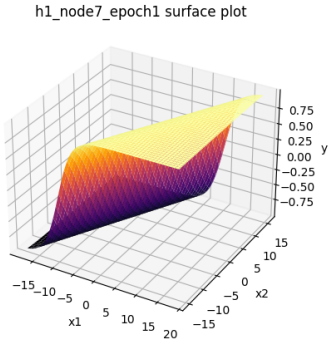
(b) Node 2



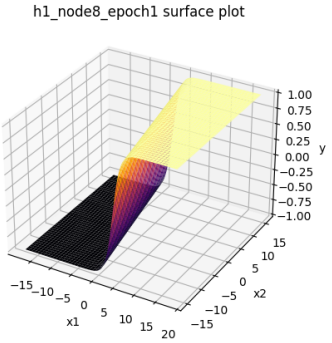
(c) Node 4



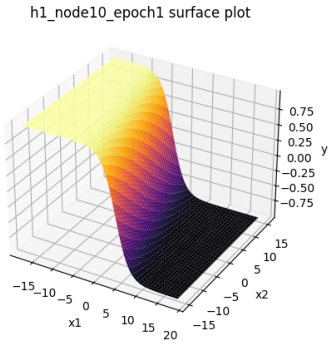
(d) Node 5



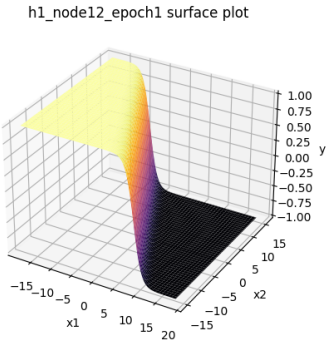
(e) Node 7



(f) Node 8



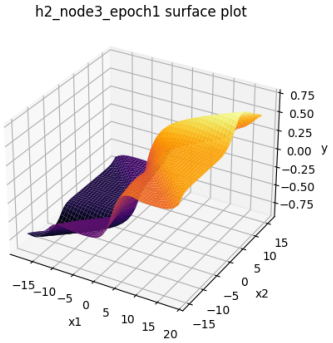
(g) Node 10



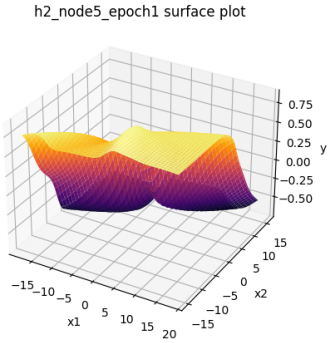
(h) Node 12

Figure 7: Surface plots for Hidden Layer 1

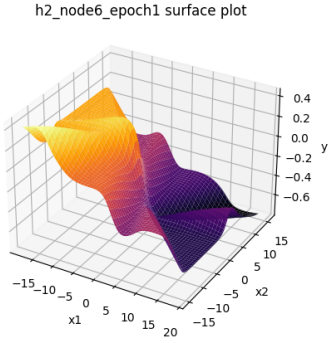
Hidden Layer 2: We randomly choose 8 nodes from hidden layer 2 and plot the following in Figure 8



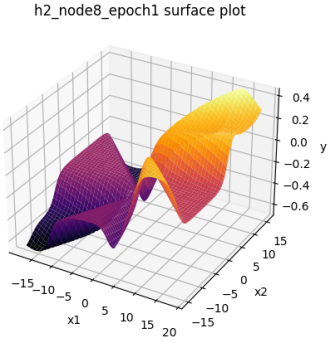
(a) Node 3



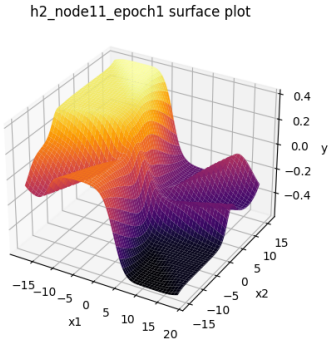
(b) Node 5



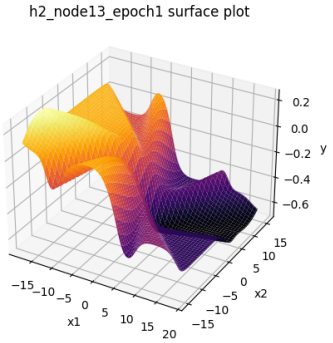
(c) Node 6



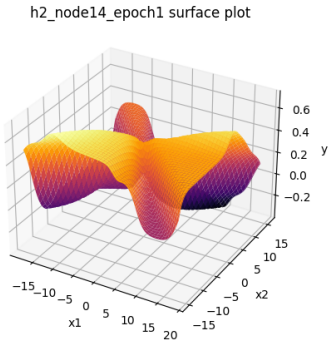
(d) Node 8



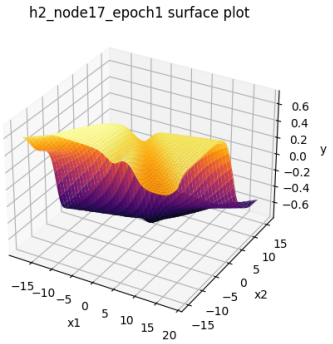
(e) Node 11



(f) Node 13



(g) Node 14



(h) Node 17

Figure 8: Surface plots for Hidden Layer 2

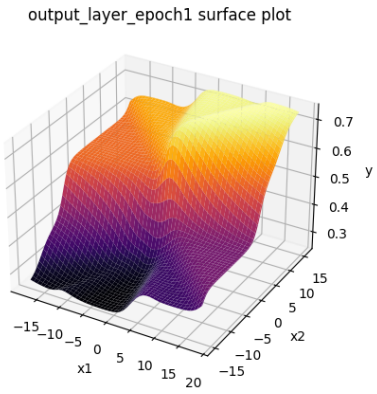
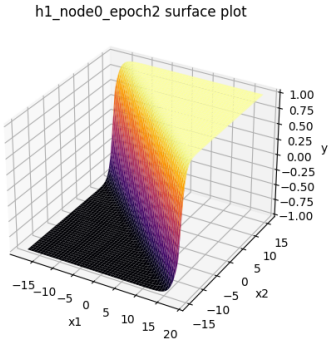


Figure 9: Output Node

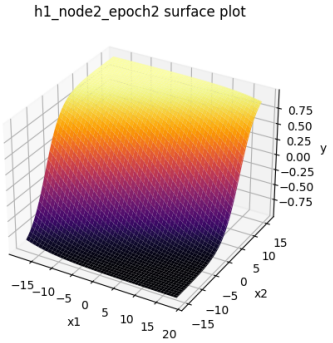
Output node: The surface plots for the output node can be found in Figure 9

2.1.2 Epoch 2

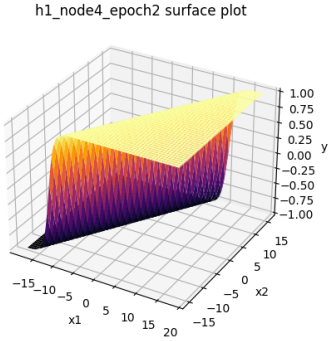
Hidden Layer 1: We randomly choose 8 nodes from hidden layer 1 and plot the following in Figure 10



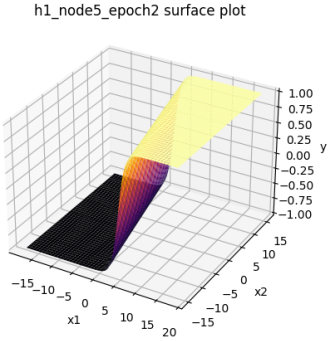
(a) Node 0



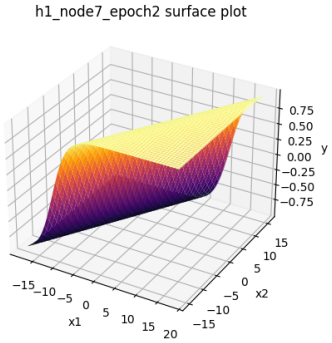
(b) Node 2



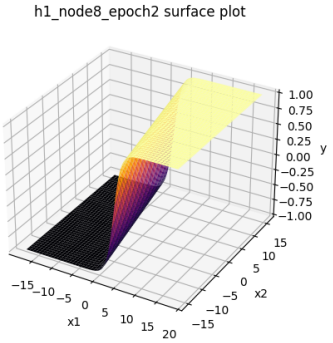
(c) Node 4



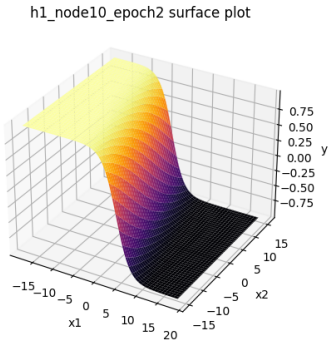
(d) Node 5



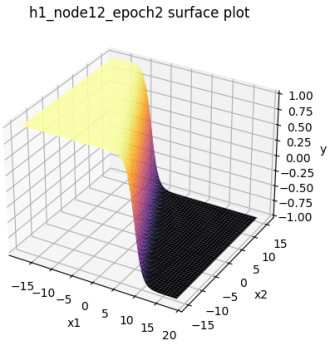
(e) Node 7



(f) Node 8



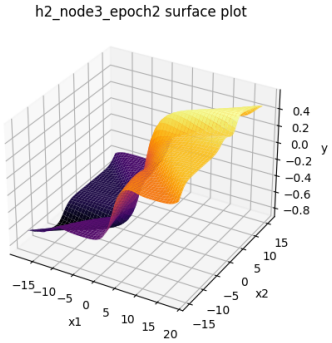
(g) Node 10



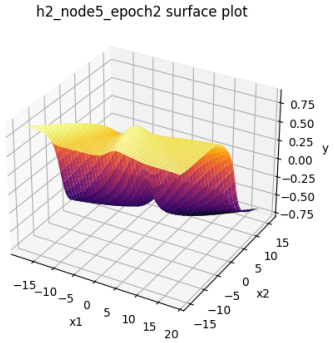
(h) Node 12

Figure 10: Surface plots for Hidden Layer 1

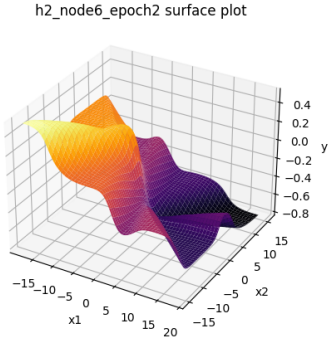
Hidden Layer 2: We randomly choose 8 nodes from hidden layer 2 and plot the following in Figure 11



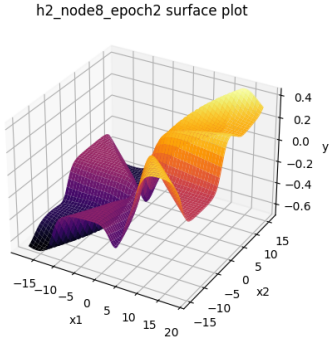
(a) Node 3



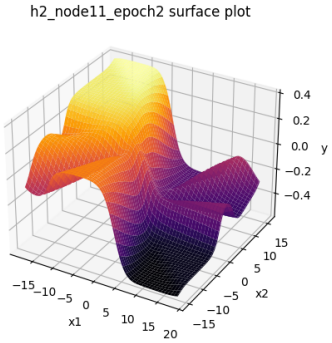
(b) Node 5



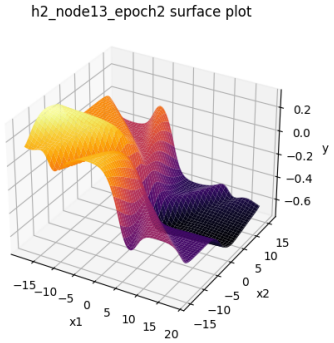
(c) Node 6



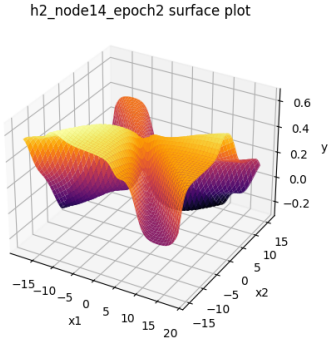
(d) Node 8



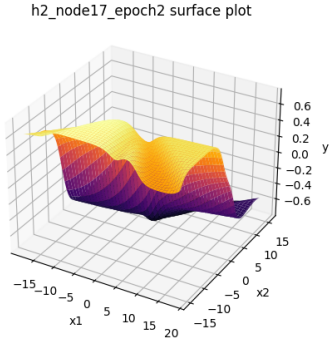
(e) Node 11



(f) Node 13



(g) Node 14



(h) Node 17

Figure 11: Surface plots for Hidden Layer 2

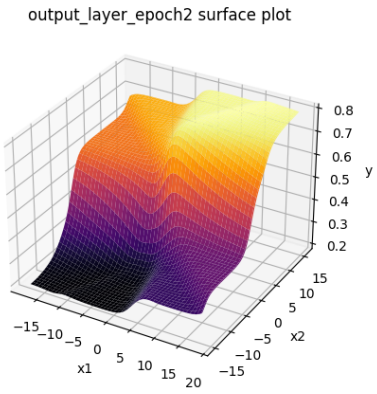
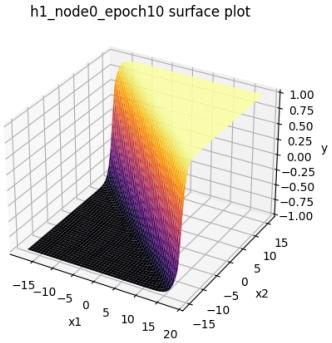


Figure 12: Output Node

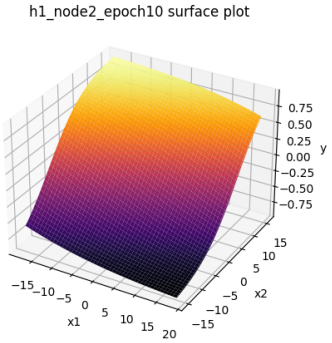
Output node: The surface plots for the output node can be found in Figure 12

2.1.3 Epoch 10

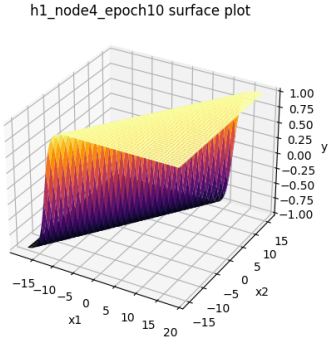
Hidden Layer 1: We randomly choose 8 nodes from hidden layer 1 and plot the following in Figure 13



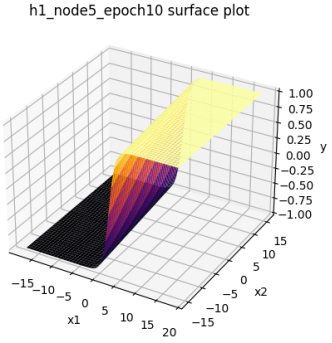
(a) Node 0



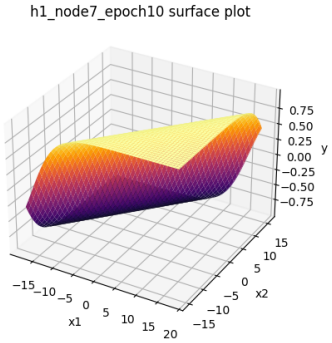
(b) Node 2



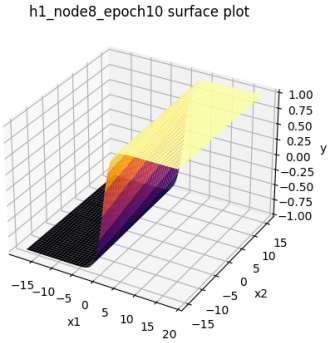
(c) Node 4



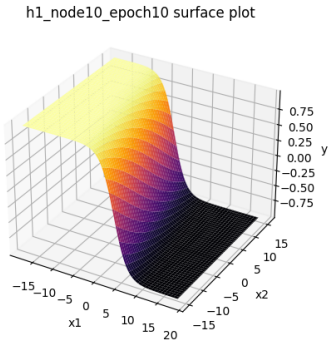
(d) Node 5



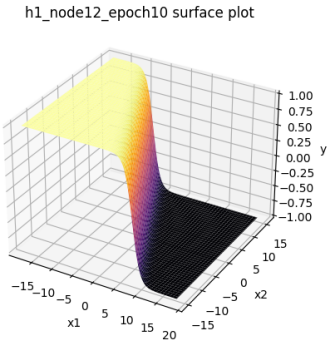
(e) Node 7



(f) Node 8



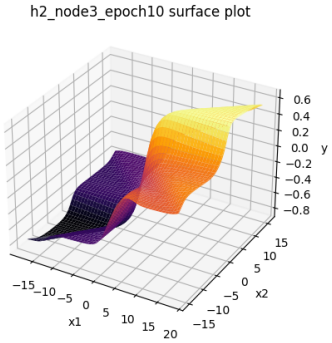
(g) Node 10



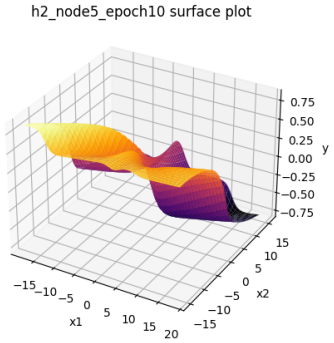
(h) Node 12

Figure 13: Surface plots for Hidden Layer 1

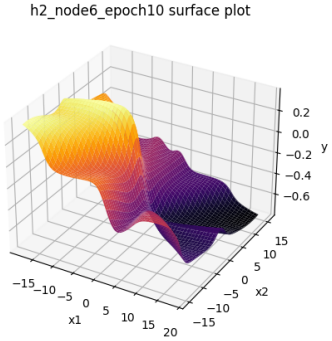
Hidden Layer 2: We randomly choose 8 nodes from hidden layer 2 and plot the following in Figure 14



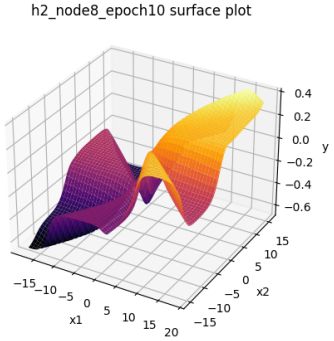
(a) Node 3



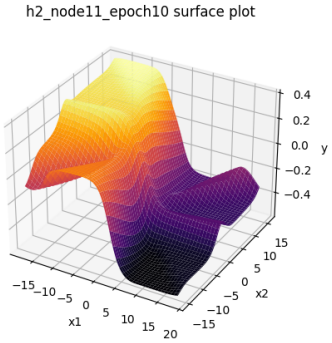
(b) Node 5



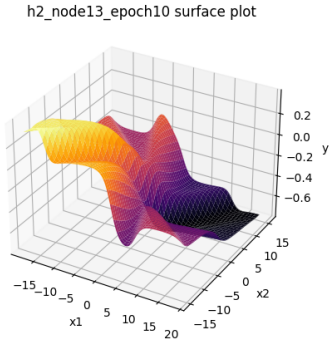
(c) Node 6



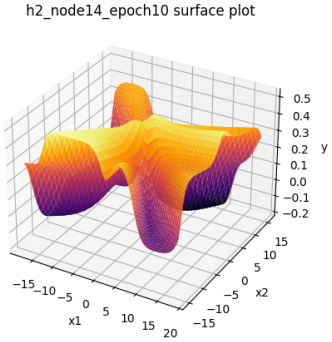
(d) Node 8



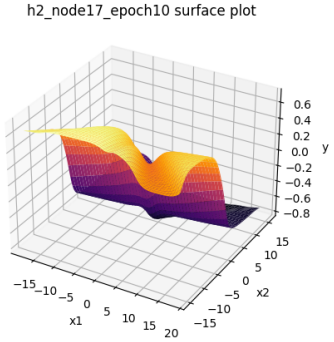
(e) Node 11



(f) Node 13



(g) Node 14



(h) Node 17

Figure 14: Surface plots for Hidden Layer 2

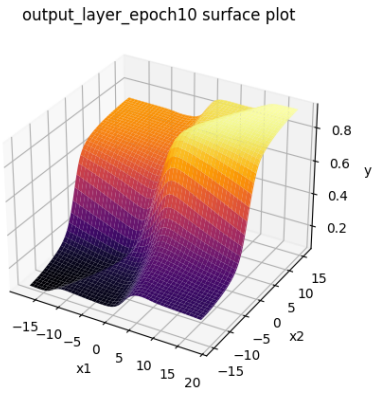
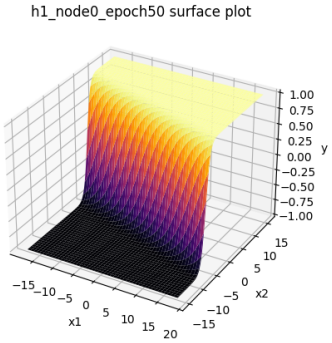


Figure 15: Output Node

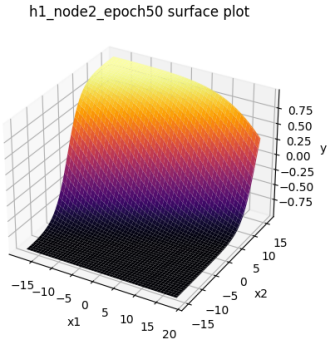
Output node: The surface plots for the output node can be found in Figure 15

2.1.4 Epoch 50

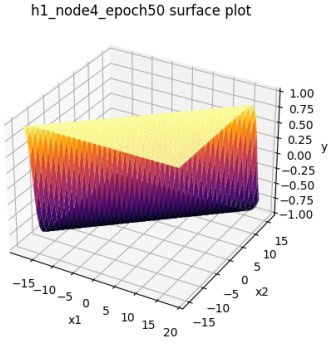
Hidden Layer 1: We randomly choose 8 nodes from hidden layer 1 and plot the following in Figure 16



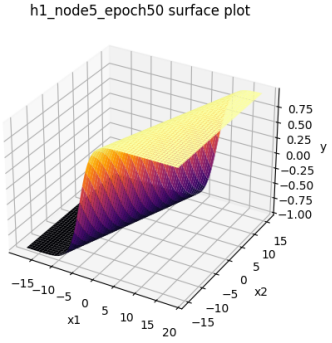
(a) Node 0



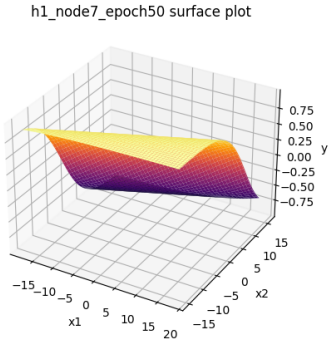
(b) Node 2



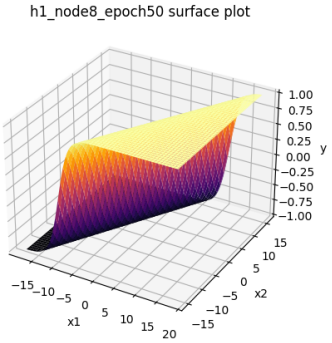
(c) Node 4



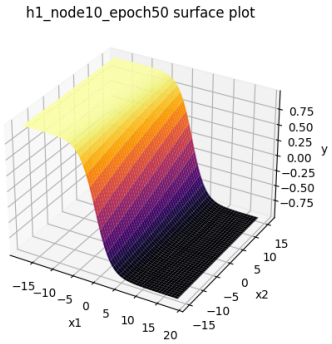
(d) Node 5



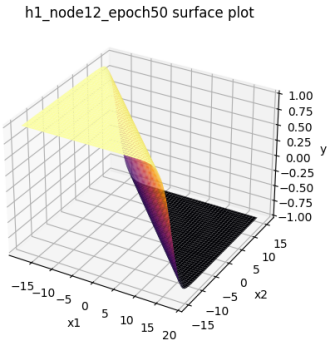
(e) Node 7



(f) Node 8



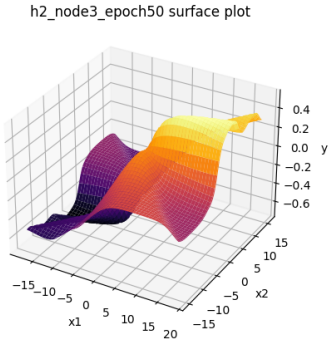
(g) Node 10



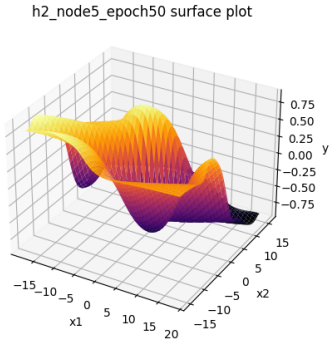
(h) Node 12

Figure 16: Surface plots for Hidden Layer 1

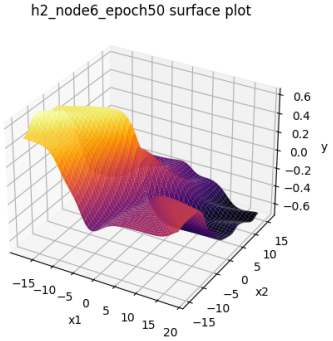
Hidden Layer 2: We randomly choose 8 nodes from hidden layer 2 and plot the following in Figure 17



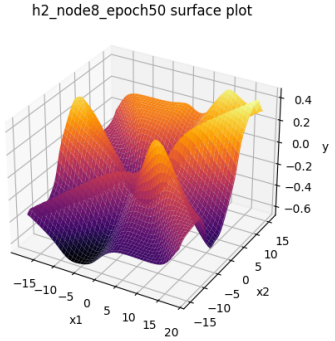
(a) Node 3



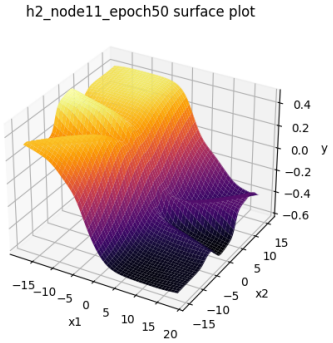
(b) Node 5



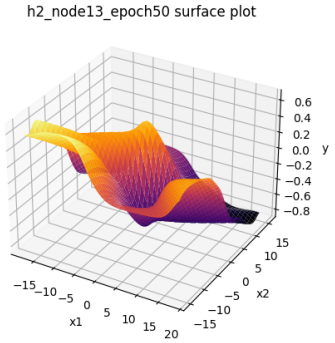
(c) Node 6



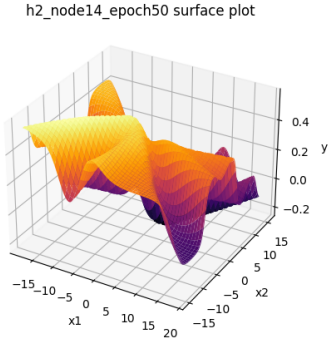
(d) Node 8



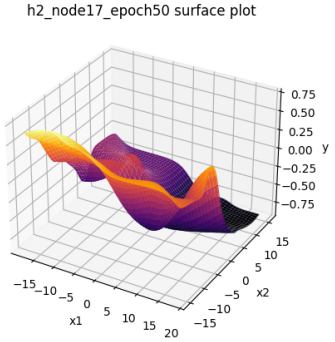
(e) Node 11



(f) Node 13



(g) Node 14



(h) Node 17

Figure 17: Surface plots for Hidden Layer 2

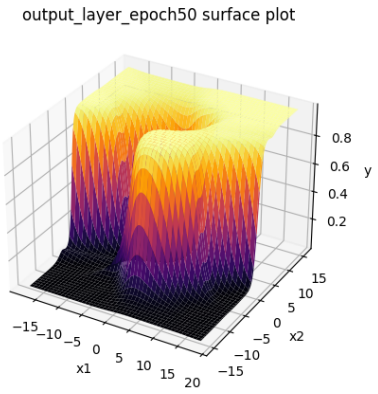
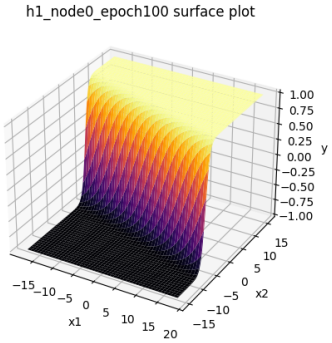


Figure 18: Output Node

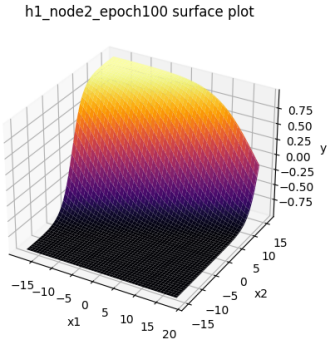
Output node: The surface plots for the output node can be found in Figure 18

2.1.5 Epoch 100

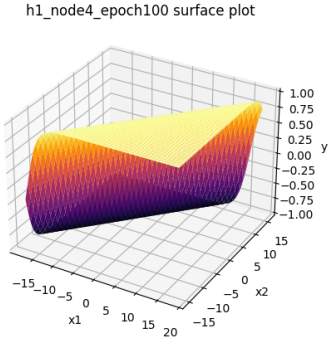
Hidden Layer 1: We randomly choose 8 nodes from hidden layer 1 and plot the following in Figure 19



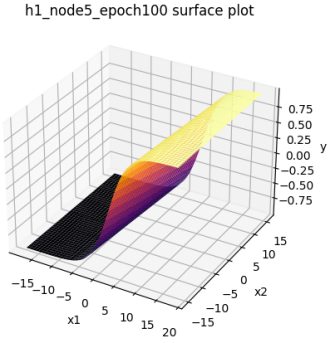
(a) Node 0



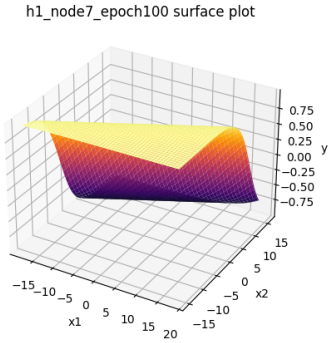
(b) Node 2



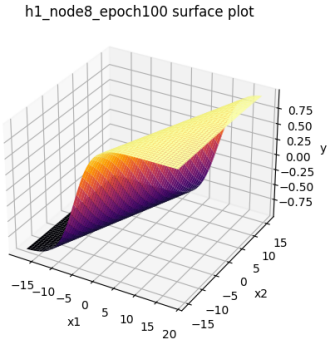
(c) Node 4



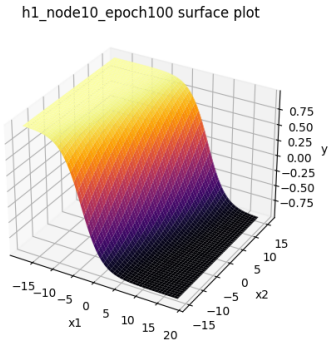
(d) Node 5



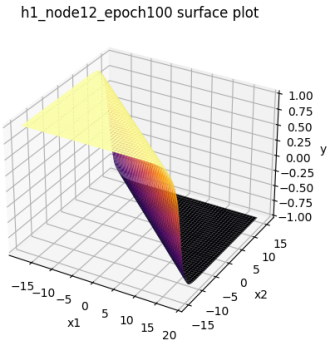
(e) Node 7



(f) Node 8



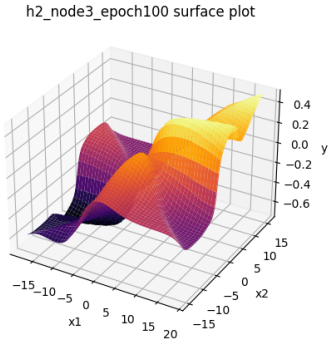
(g) Node 10



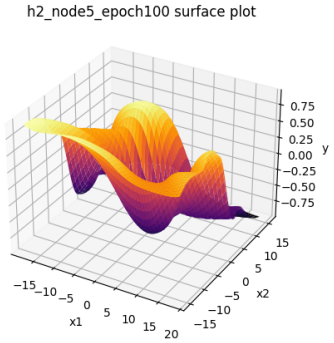
(h) Node 12

Figure 19: Surface plots for Hidden Layer 1

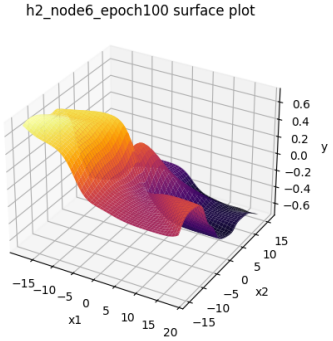
Hidden Layer 2: We randomly choose 8 nodes from hidden layer 2 and plot the following in Figure 20



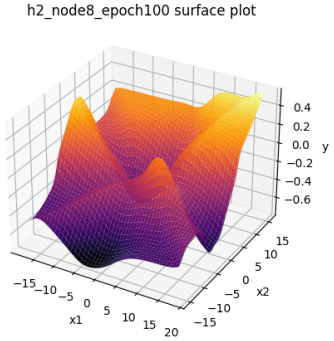
(a) Node 3



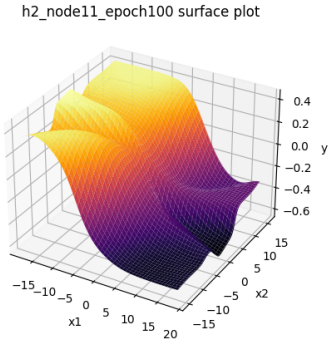
(b) Node 5



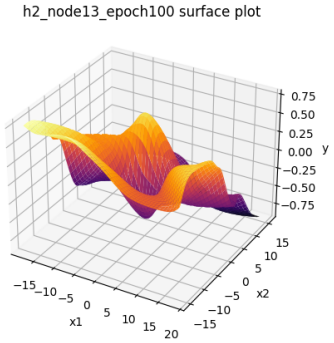
(c) Node 6



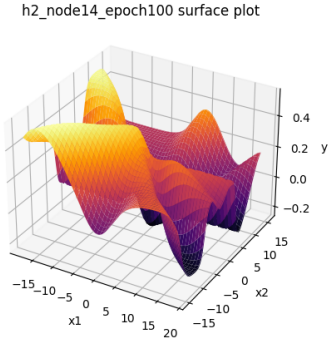
(d) Node 8



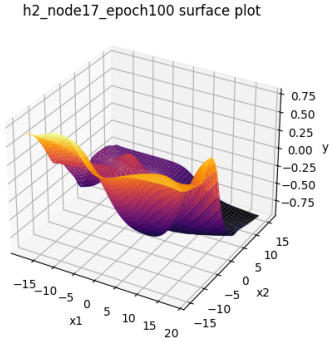
(e) Node 11



(f) Node 13



(g) Node 14



(h) Node 17

Figure 20: Surface plots for Hidden Layer 2

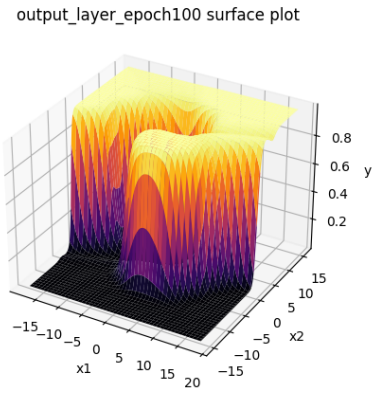


Figure 21: Output Node

Output node: The surface plots for the output node can be found in Figure 21
Observations:

1. The first observation that we see is that as the number of epochs increased the model becomes more and more complex and starts learning the classification data.

2. We find that the surface outputs at the nodes of hidden layer 1 are not as complex, while the surface outputs at hidden layer 2 are more complex than at hidden layer 1. This is what was expected because as we go deeper in the neural network the representation of nodes becomes more complex as it is a combination of previous layers and hence the outputs become more and more non linear.

3. The surface plots of hidden layer 1 resemble the tanh function is some way. This is because nodes in hidden layer 1 just represent linear combination of data followed by a single activation function(tan h) and hence the surface plots have to resemble tanh function. And hidden layer 2 combines these various linear transformations of tanh function to give more complex nodes which is reflected in the surface plots.

4. Also the final surface plot of output node after model is finished training resembles the decision boundary plot that is attached below. We can also see how the output layer starts from a very similar surface and then evolved into a complex surface.

2.2 Decision Boundary

To better understand the model predictions we plot the decision boundary.

Observation: The above decision boundary plot tells that the trained model is working

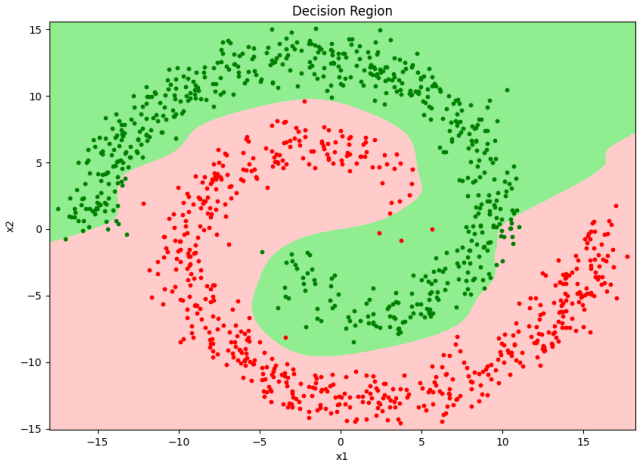


Figure 22: Decision Boundary

correctly and is classifying the data correctly. Just a few data points are classifed incorrectly which is evident from above figure and hence the model which gives an accuracy of 98 % is correctly classifying the 2D dataset.

2.3 Image Classification

In this task we make use of the same model but use the Cross Entropy loss as we have 5 target classes. We split the training data into a 70: 30 split and train on the 70% split. For the same configurations we make use 3 different optimizers namely

- Delta rule
- Generatized Delta rule
- Adam

For each of these 3 optimizers we train the model for 250 epochs and analyse the training accuracy and validation accuracy. The accuracy plots are as follows in Figure 23 and 24.
Observations:

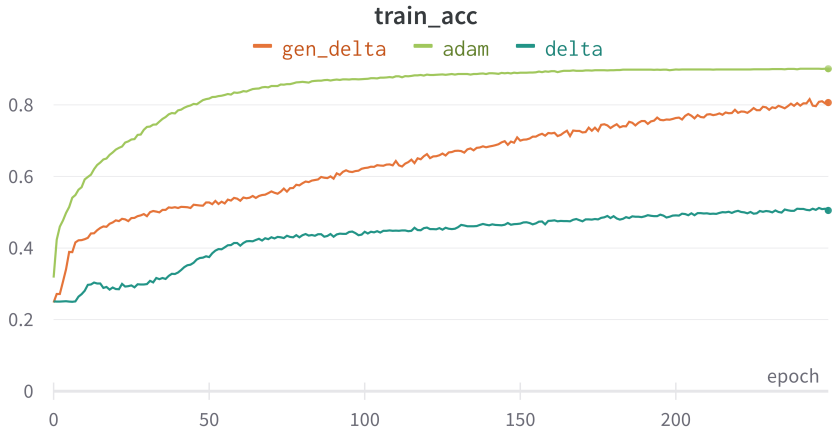


Figure 23: Training Accuracy

1. From the above 2 graphs we can compare the 3 optimization methods i.e Adam, Generalized Delta and Delta. We see that Adam's method converges and flattens out fastest followed by Generalized Delta method , while the Delta method is slowest to converge.
2. It takes around 50 epochs for training and validation curves to flatten out when using Adam's method while it takes around 120 epochs for Generalized Delta to converge

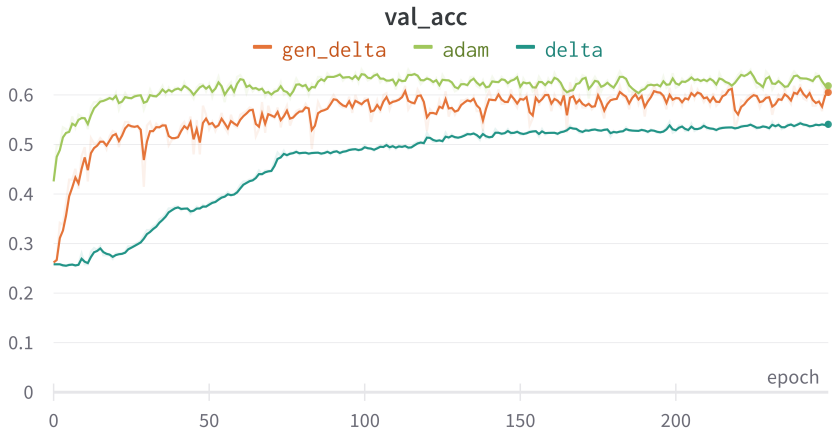


Figure 24: Validation Accuracy

while Delta method is extremely slow in converging and takes more than 200 epochs and it still doesn't give as much accuracy as its other 2 counterparts.

3. From an accuracy point of view too Adam's method and Generalized Delta method are much more accurate than normal Delta method as they give an accuracy of greater than 0.6 while the Delta method doesn't perform as great with an accuracy of around 0.5.

4. Even though Generalized Delta method matches the accuracy given by Adam's Method after considerable number of Epochs(¿250). Adam's method is still superior than Generalized Delta method because it learns much faster and gives a much better accuracy in initial small number of epochs itself.

2.3.1 Confusion Matrix Plots

We plot the confusion matrix on both training set and test set. The plots are as follows.
Adam Optimizer: The confusion matrices in Figure 25 and Figure 26

Generalized Delta Rule Optimizer: The confusion matrices in Figure 27 and Figure 28
Delta Rule Optimizer: The confusion matrices in Figure 29 and Figure 30

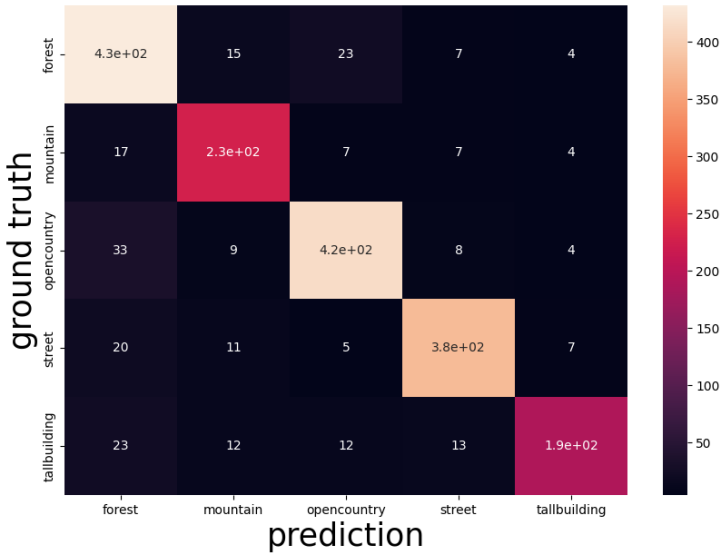


Figure 25: Training data

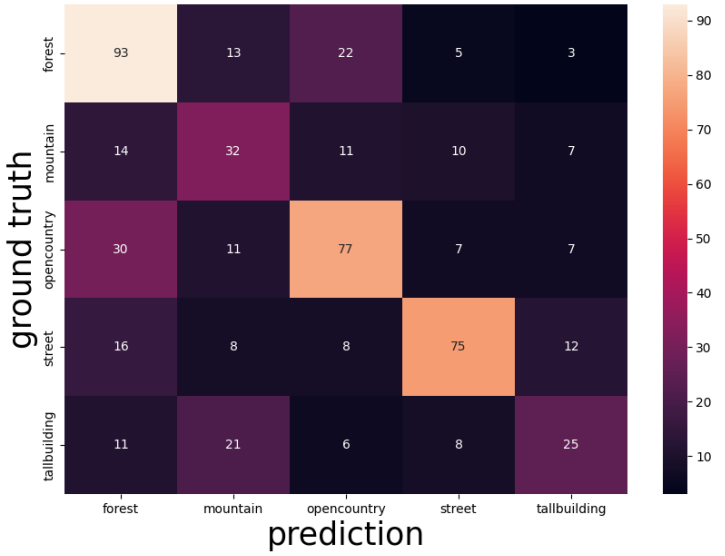


Figure 26: Test data

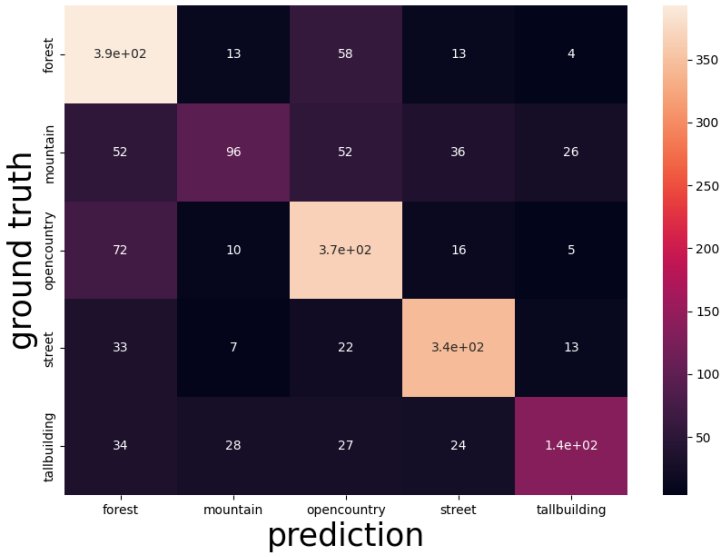


Figure 27: Training data

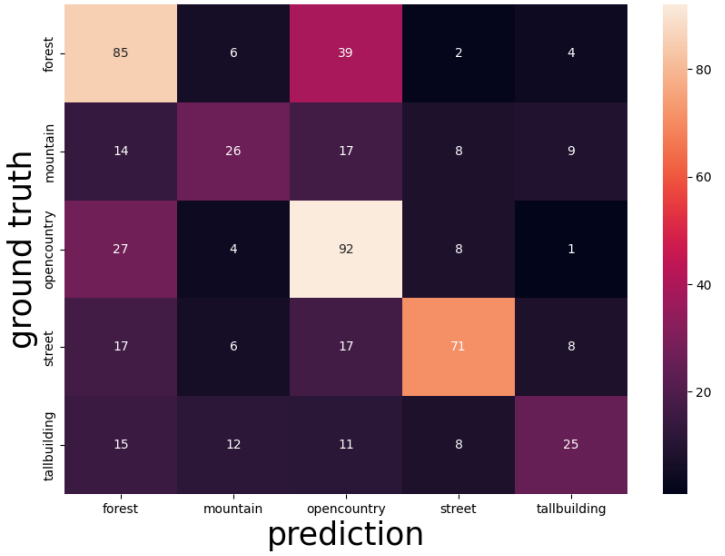


Figure 28: Test data

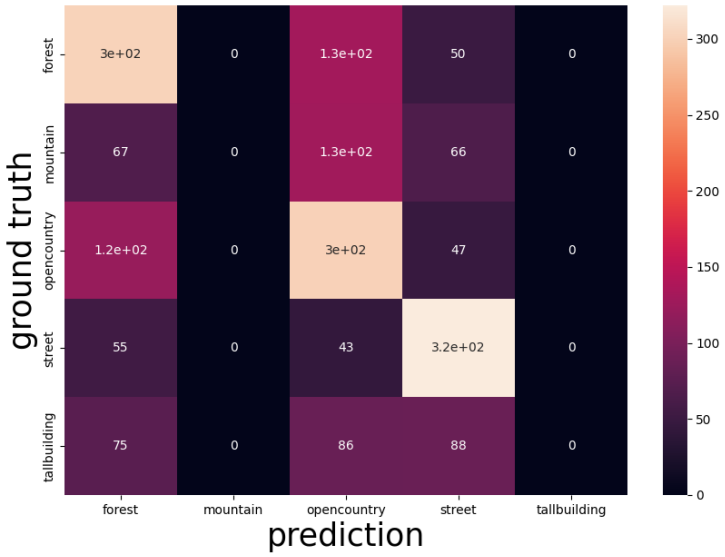


Figure 29: Training data

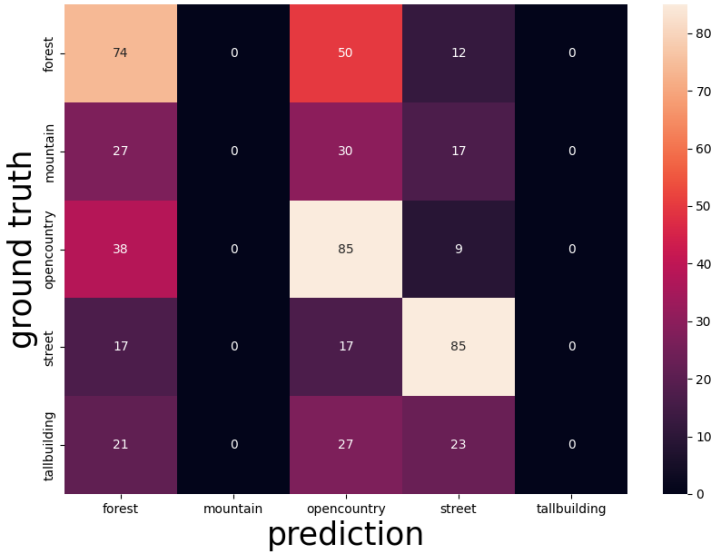


Figure 30: Test data

Observations:

1. We notice that the classes *forest* and *open country* are challenging for the model to learn and it often confuses the two.

2. Since we train the model for 250 epochs we notice that since Adam converges fastest, it shows the best diagonal relationship in the matrix plots, followed by generalized delta rule and then delta rule.

3. Even after training for 250 epochs the model trained using the delta rule does not learn to identify the *mountain* and *tall building* classes.

4. Representing every image by a 60 dimensional vector and considering each feature independent isn't the optimal strategy. This can be seen in the plots as we only achieve a accuracy of 65% at max on a held out validation set even after fine-tuning parameters.

5. We also notice that the model has overfitted on the training data as its accuracy is way higher than the validation accuracy. Since the model configuration was given to us we could not make any changes but one could explore techniques like Dropouts.