Indian Institute of Technology Madras

CS6910 Deep Learning

Assignment 1

Vimal Suresh Mollyn ED17B055
Dyava Naveen Reddy ME17B140
Ramakrishnan NA18B030

March 4, 2022

Vimal Suresh Mollyn, ED17B055
Dyava Naveen Reddy, ME17B140
Ramakrishnan, NA18B030

Contents

1 Task 1

1.1 Function approximation task

- This section describes the implementation of the neural network for regression of 2-D data. Firstly, the data was split into train and validation and test in the ratio 70:10:20.
- The wights are initialised using Xavier initialization technique, where the weights are distributed uniformly in the range of (-y,y) where y = 1/sqrt(n) (n is the number of inputs to a given neuron).
- The stop parameter for the training was set to the difference between the previous epoch error and present epoch error to be less than 1e-5 and the validation error stops decreasing after the order 1e-2.
- The values of the hyperparameters which yielded the best performance are as given below:
  - Learning rate : 1e-3
  - Momentum : 0.9
  - Number of nodes in hidden layer 1 : 8
  - Number of nodes in hidden layer 2 : 16

1.2 Observations

- We need to obtain the values of hidden layer nodes such that the model converges in the lesser number of epochs and doesnt overfit the training data
- We have experimented with the number of nodes in layer 1, number of nodes in layer 2, and learning rate parameter for hyperparameter tuning and chose the those hyperparameters which gave smooth convergence of training error in lesser number of epochs
- We have plotted the rate of convergence of training error and validation error vs. epochs for different combination of nodes in hidden layer 1 and hidden layer 2

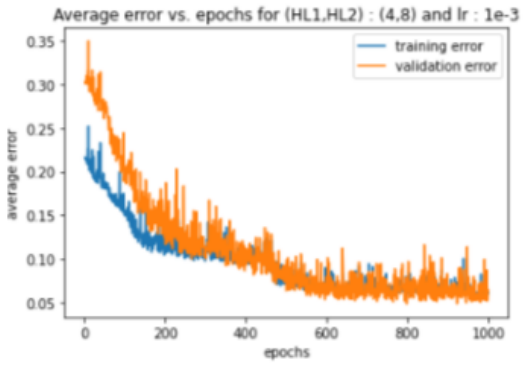

Figure 1: Average error vs. epoch for 4 nodes in hidden layer 1 and 8 nodes in hidden layer 2 and learning rate 1e-3

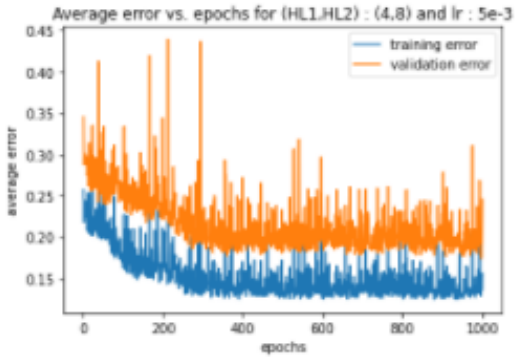

Figure 2: Average error vs. epoch for 4 nodes in hidden layer 1 and 8 nodes in hidden layer 2 and learning rate 5e-3

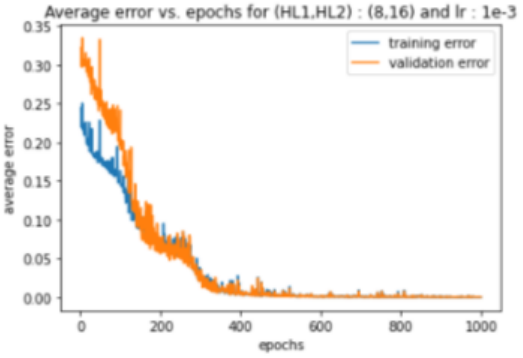

Figure 3: Average error vs. epoch for 8 nodes in hidden layer 1 and 16 nodes in hidden layer 2 and learning rate 1e-3

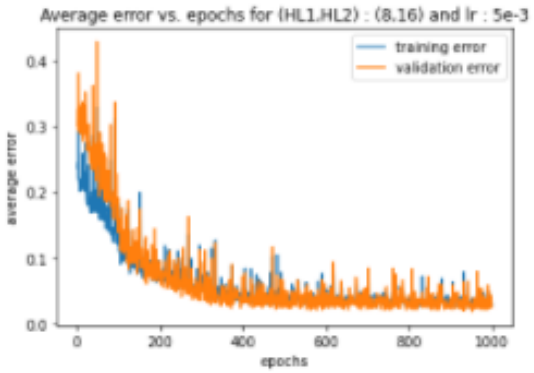

Figure 4: Average error vs. epoch for 8 nodes in hidden layer 1 and 16 nodes in hidden layer 2 and learning rate 5e-3

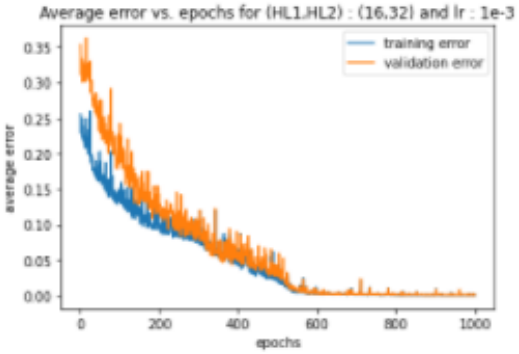

Figure 5: Average error vs. epoch for 16 nodes in hidden layer 1 and 32 nodes in hidden layer 2 and learning rate 1e-3

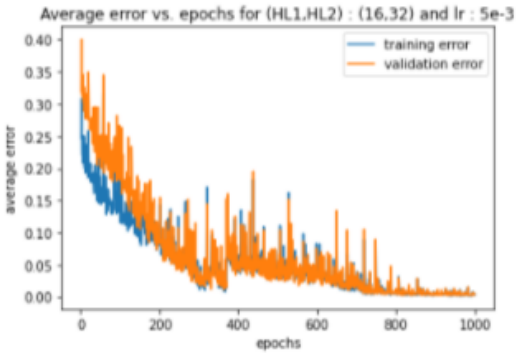

Figure 6: Average error vs. epoch for 16 nodes in hidden layer 1 and 32 nodes in hidden layer 2 and learning rate 5e-3

- It is observed as the number of nodes to be learnt increases the training time increases. By empirical results the number of training examples needed is

10 times the number of model parameters, In our case we have 350 training samples.

- Thus by observation the model with 8 nodes in hidden layer 1 and 16 nodes in hidden layer 2 gives the optimal performance.
- We have also tried the range of learning rate for SGD optimizer from [5e-3, 1e-3], it is observed that the training error oscillates by larger values for learning rate = 5e-3, thus the learning rate value of 1e-3 is found to be optimal

HL1	HL2	lr	Validation error	Epochs
4	8	1e-3	0.05	1000
8	16	1e-3	0.01	500
16	32	1e-3	0.01	600
4	8	5e-3	0.15	1000
8	16	5e-3	0.05	500
16	32	5e-3	0.01	800

1.3 Implementation details:

- The weights are initialised using Xavier initialization
- The learning rate parameter for generalised delta rule is 1e-3
- The momentum for generalised delta rule is 0.90
- The number of nodes in hidden layer 1 is 8
- The number of nodes in hidden layer 2 is 16

1.4 Plot for Model output vs Desired output:

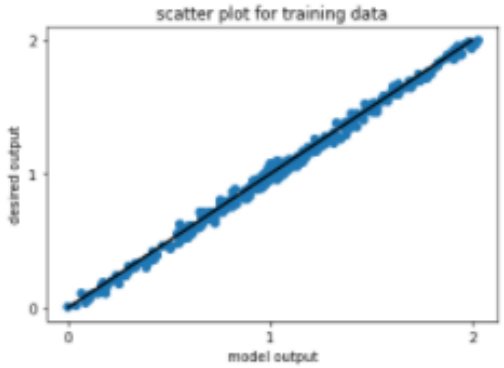

Figure 7: Scatter plot of Model output and Desired output for Training data

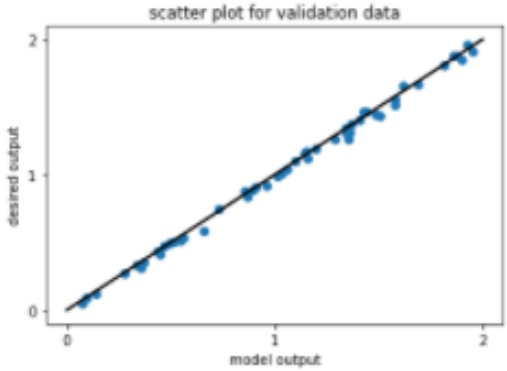

Figure 8: Scatter plot of Model output and Desired output for Validation data

1.5 Plots of desired function and approximated function:

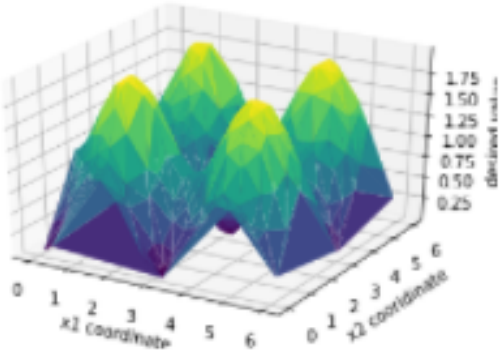

Figure 9: Surface plot of Desired function

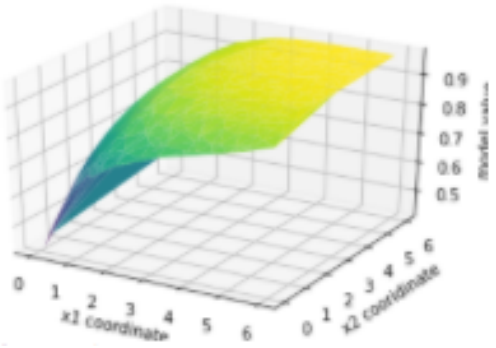

Figure 10: Surface plot of Approximated function after 1 epoch

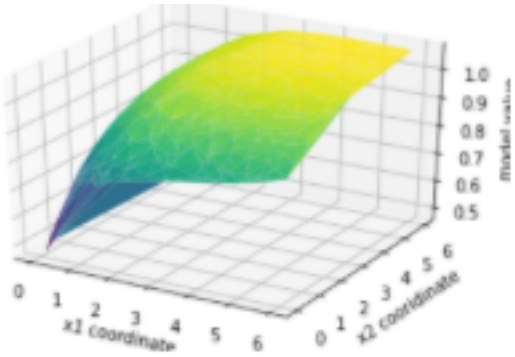

Figure 11: Surface plot of Approximated function after 2 epoch

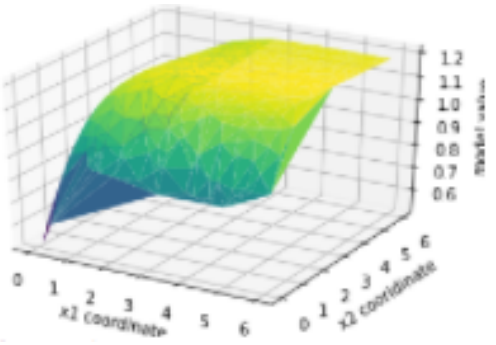

Figure 12: Surface plot of Approximated function after 10 epoch

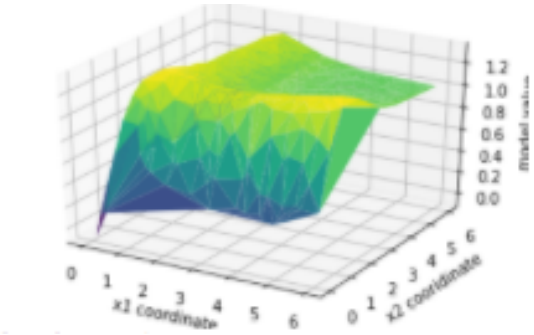

Figure 13: Surface plot of Approximated function after 50 epoch

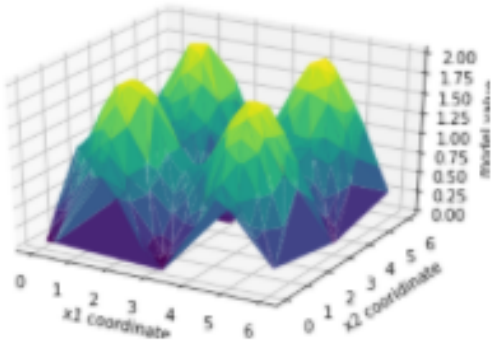

Figure 14: Surface plot of Approximated function after training

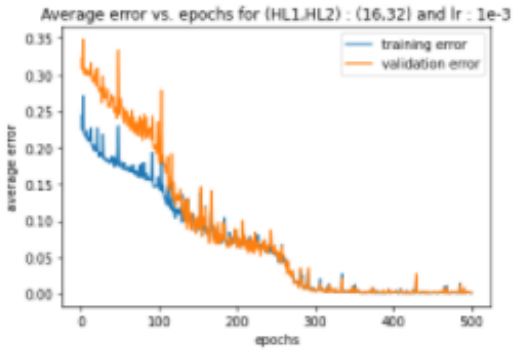

Figure 15: Average error vs. epoch for the desired model

2 Task 2a

2.1 Classification Task for 2-D data

- In this task we have to perform binary classification of a 2-D data whose labels are given as 0 or 1 in the data set using a 2 layered Neural Network.

- The data set contains 3 categories of data - training data, validation data and the test data. The Neural Network model is trained using training data and validated using the validation data given in the data set. Further predictions are made for test data. All the three data sets given are plotted below:

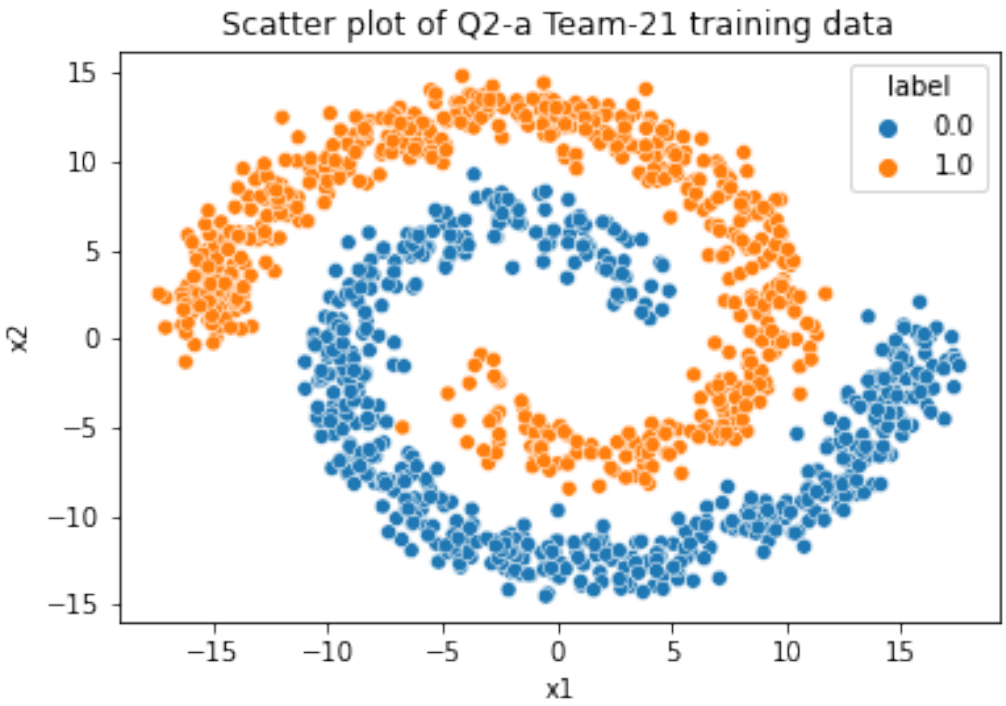

Figure 16

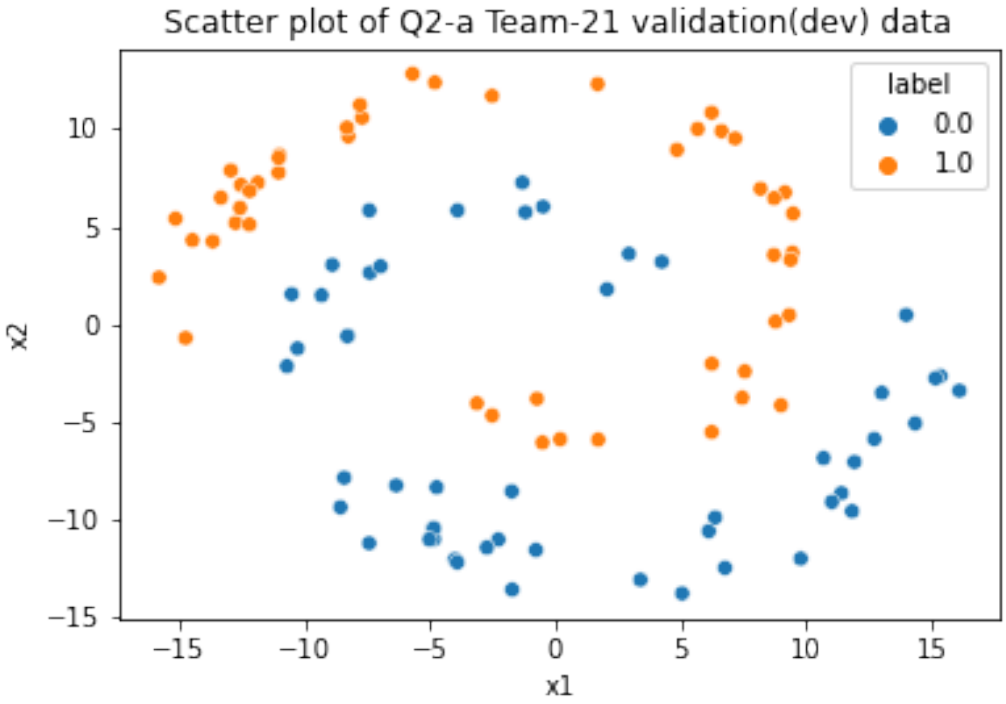

Figure 17

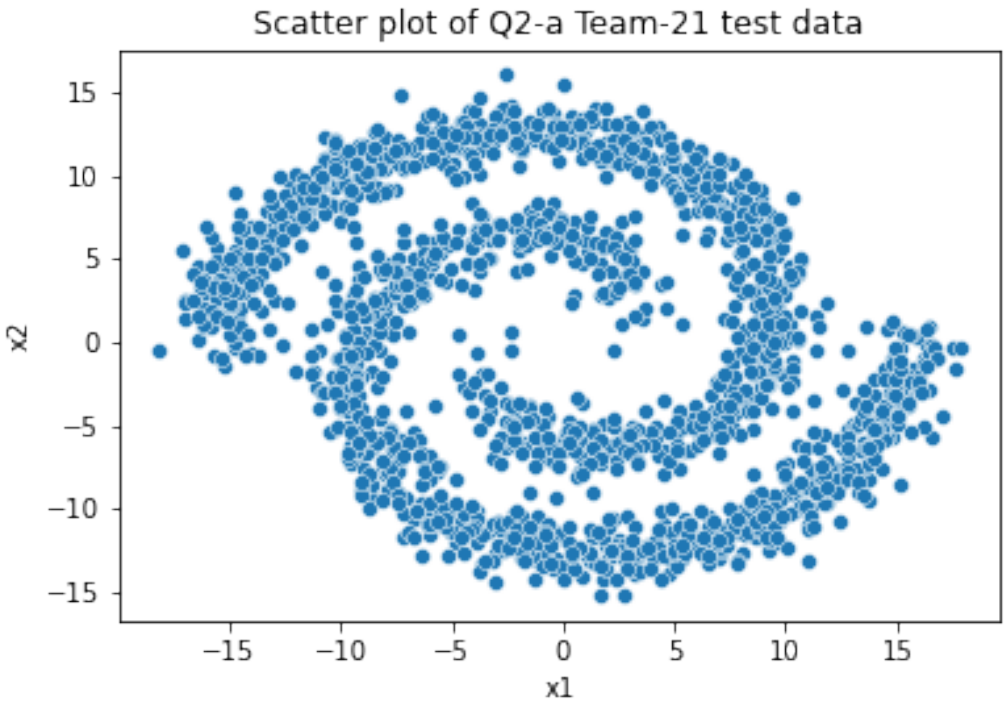

Figure 18

- The given data is not normalized because the given data is spread equally along both the directions for both the class. The model performed very well and converged fast without normalized fearures.
- The model architecture is as follows:
  - Input Layer - 2 dimensional input
  - Hidden Layer - 1 - Tanh activation function
  - Hidden Layer - 2 - Tanh activation function
  - Output Layer - 2 logits & softmax activation function

    The output class is decided by the logit which has the maximum value
- The hyper-parameter tuning is done using Weights & Biases tool for visualisation and it was executed using grid search. The following are the hyper-parameters which are used in grid search
  - Learning rate : [0.1, 0.001]
  - Momentum : [0.8, 0.85, 0.95]
  - hidden layer 1 : [1, 2, 4, 8, 15, 16, 20, 25, 30, 32]
  - hidden layer 2 : [1, 2, 3, 4, 8, 10, 16, 32]

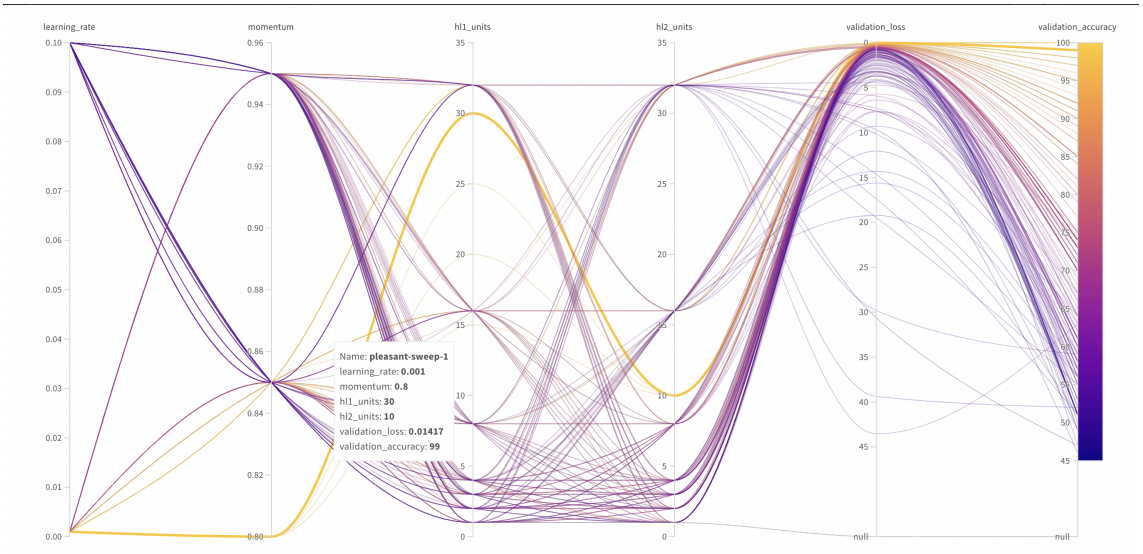

Figure 19: Visualisation of the hyper-parameter performances in the Weights Biases visualization tool

- After grid search hyper-parameter tuning the best hyper-parameters found are listed below:
  - Learning rate : 0.001
  - Momentum : 0.8
  - hidden layer 1 : 30
  - hidden layer 2 : 10

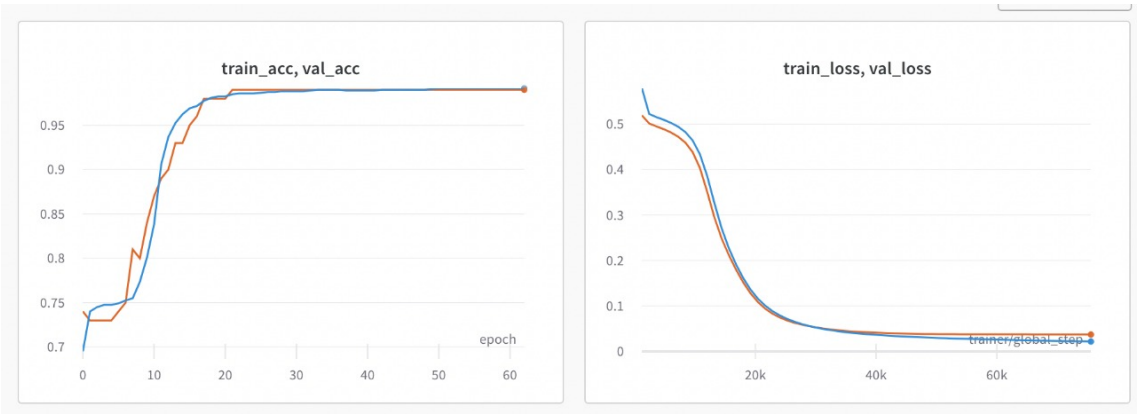

Figure 20: Train Validation accuracy, losses vs epochs for the best model

- All the hyper-parameter combinations are trained for 100 epochs and selected accordingly.
  The figure below shows the scatter plot of predictions on the test data

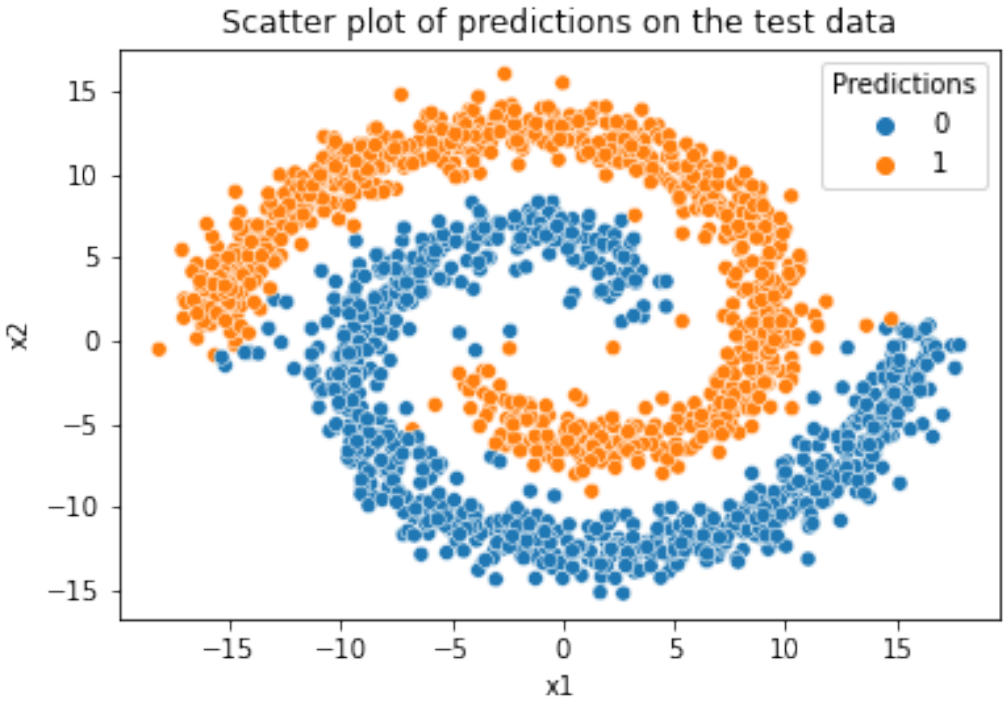

Figure 21: Scatter plot of predictions on the test data

2.2 Decision Region Plots

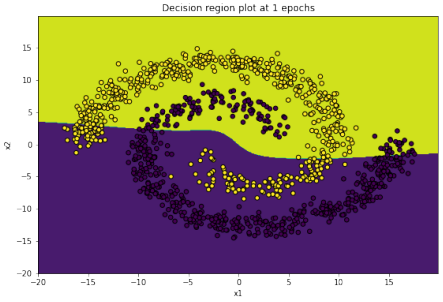

Figure 22

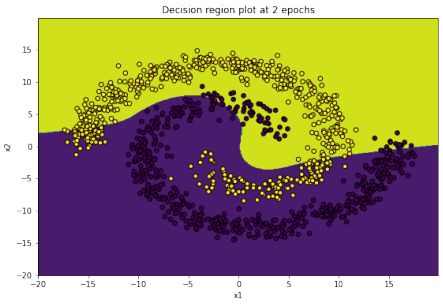

Figure 23

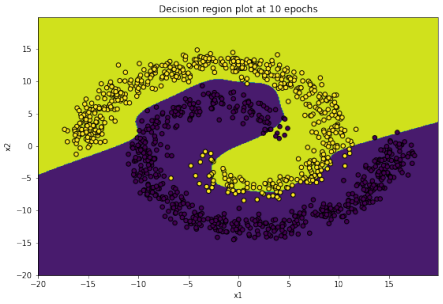

Figure 24

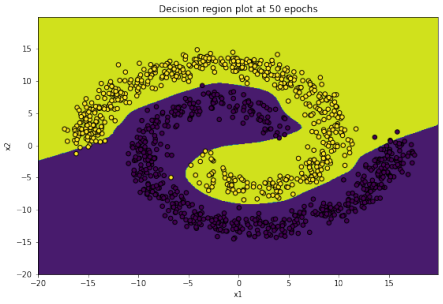

Figure 25

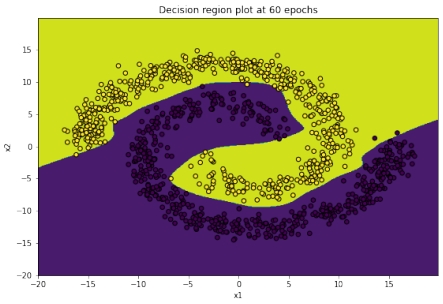

Figure 26

2.3 Surface Plots of first 10 units of layer-1 at 1, 2, 10, 50 and 60 epochs

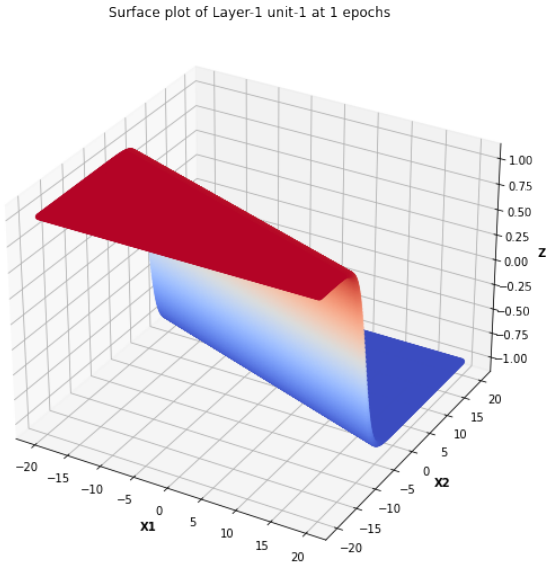

Figure 27

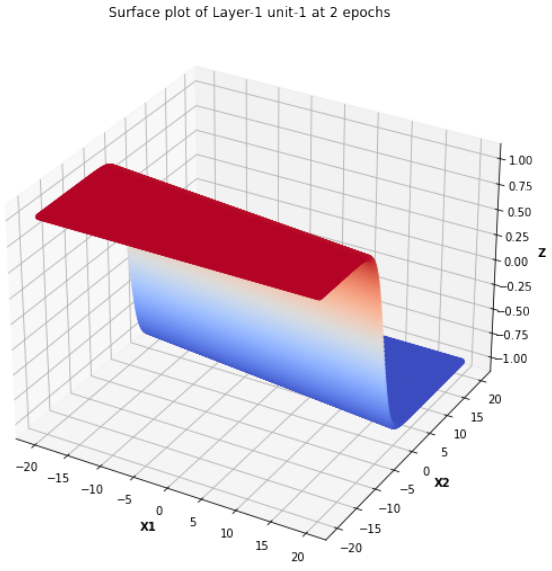

Figure 28

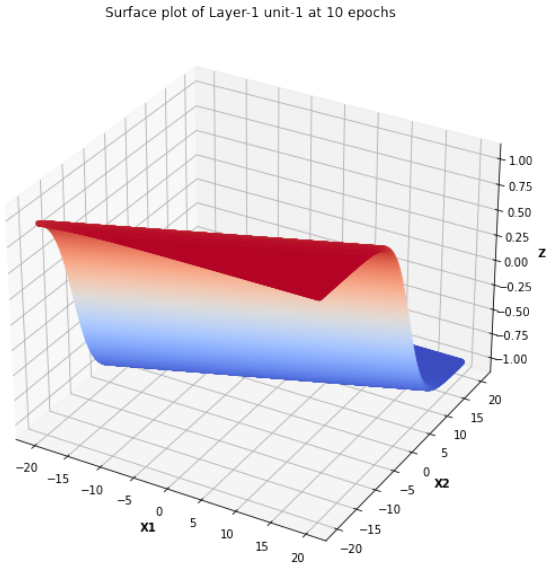

Figure 29

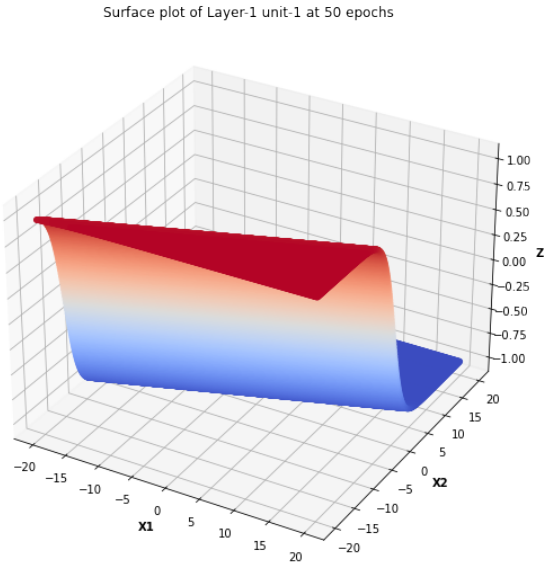

Figure 30

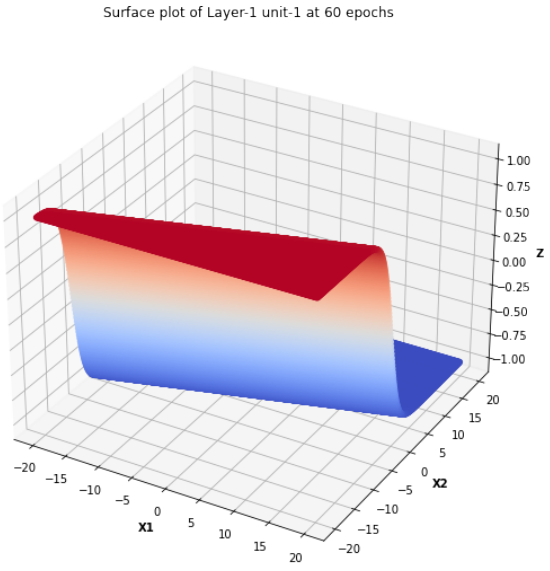

Figure 31

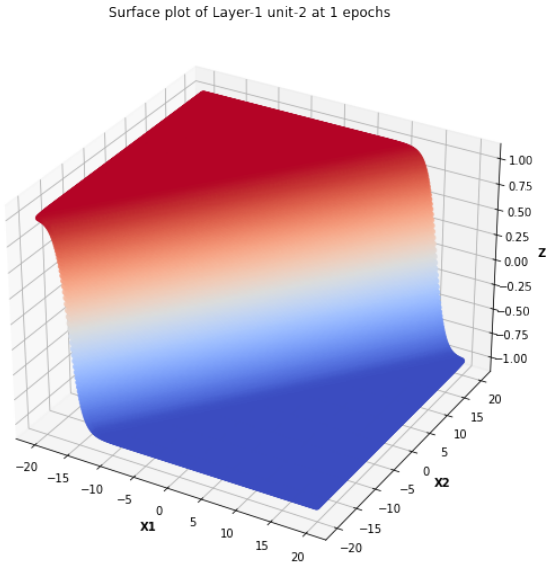

Figure 32

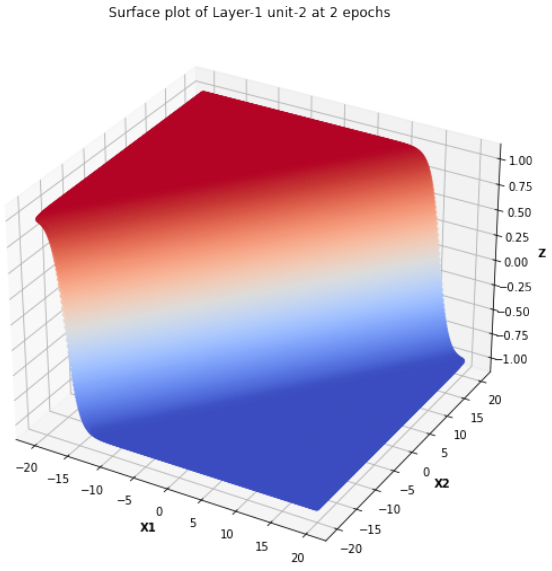

Figure 33

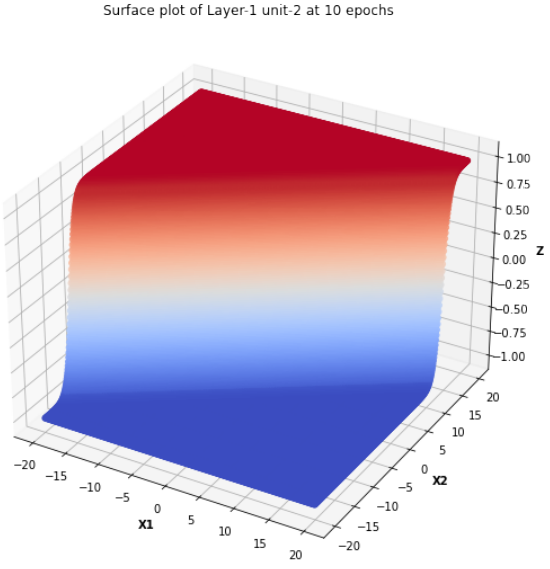

Figure 34

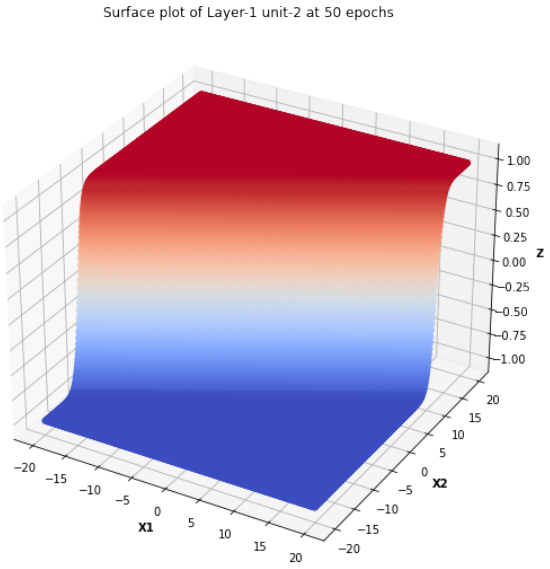

Figure 35

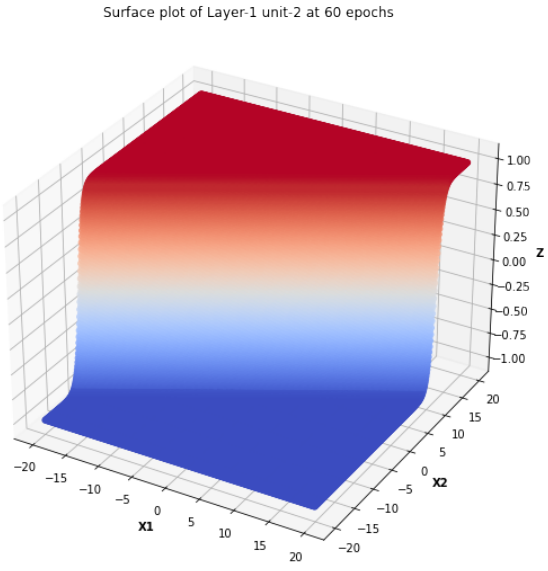

Figure 36

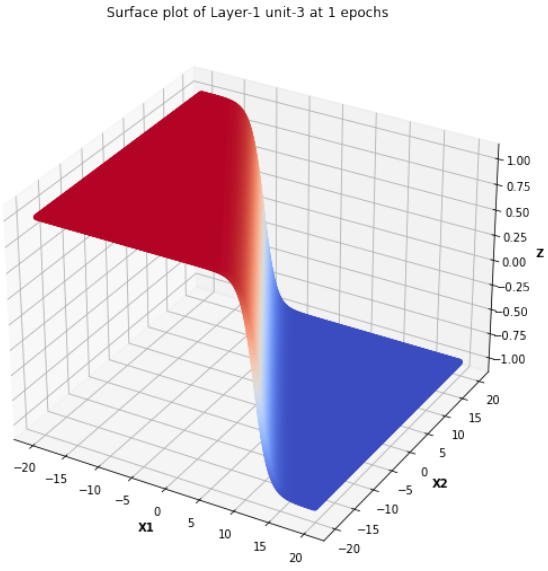

Figure 37

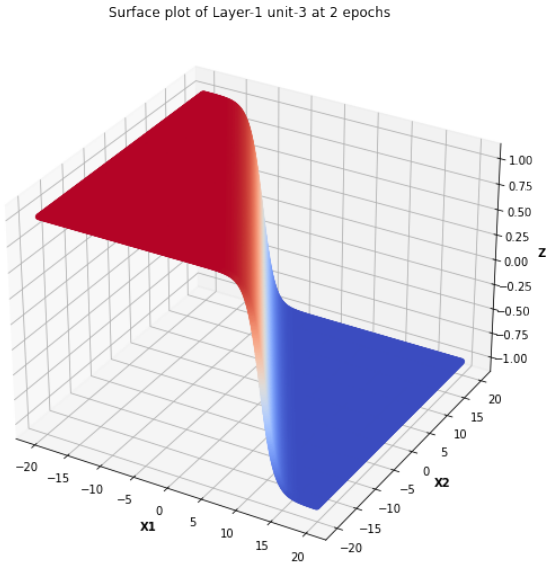

Figure 38

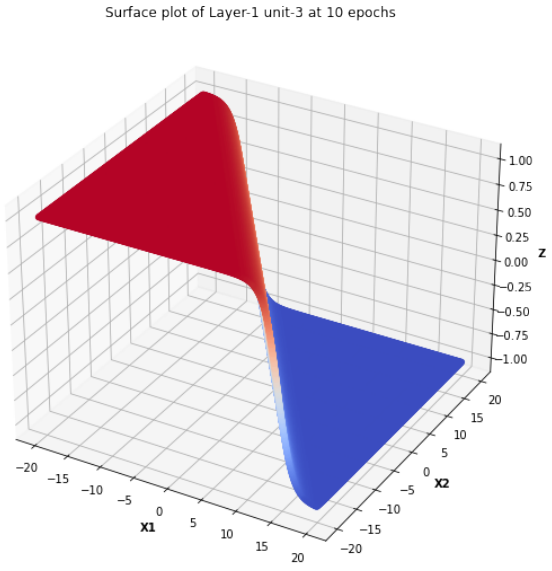

Figure 39

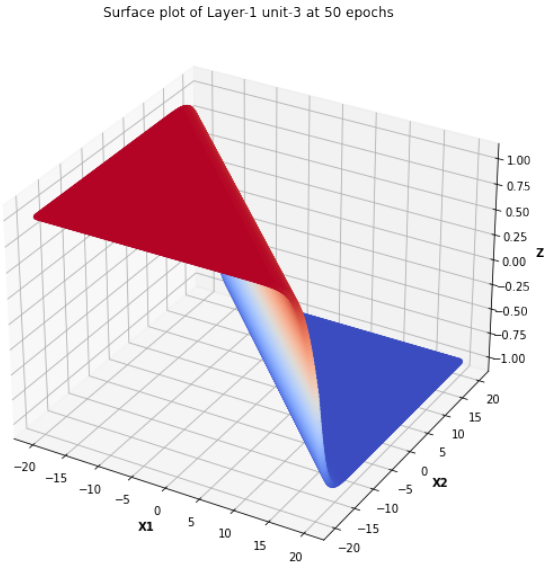

Figure 40

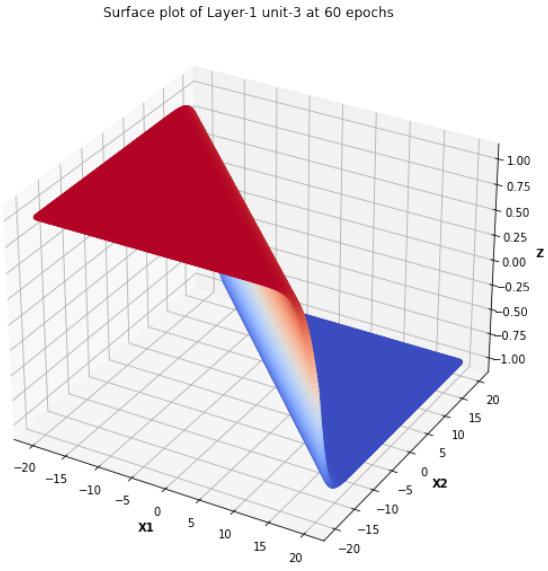

Figure 41

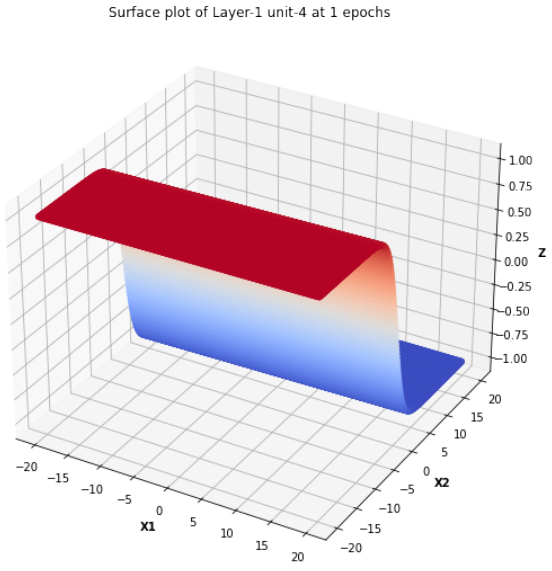

Figure 42

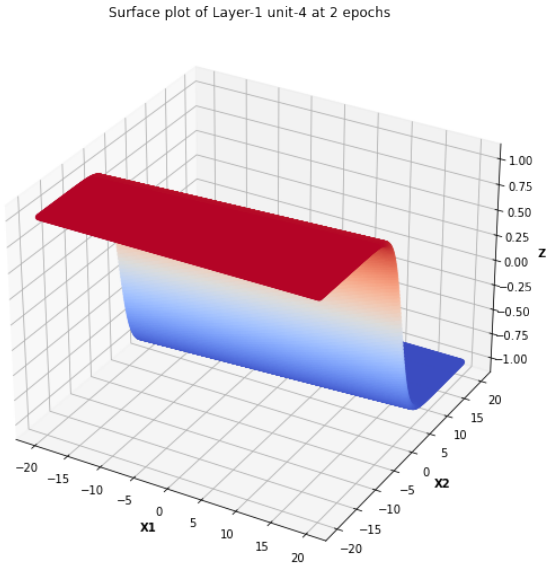

Figure 43

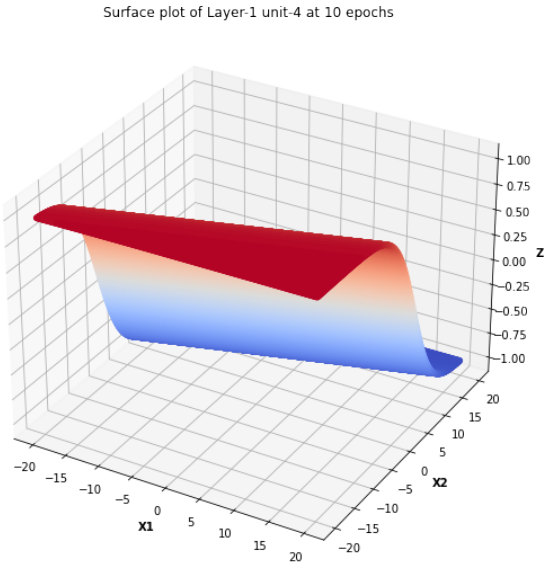

Figure 44

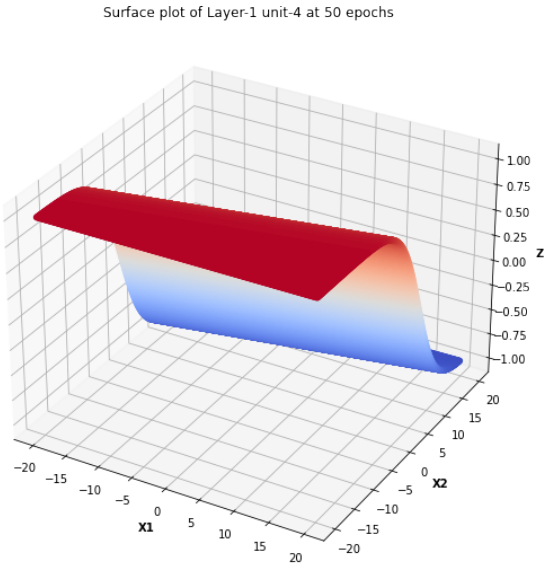

Figure 45

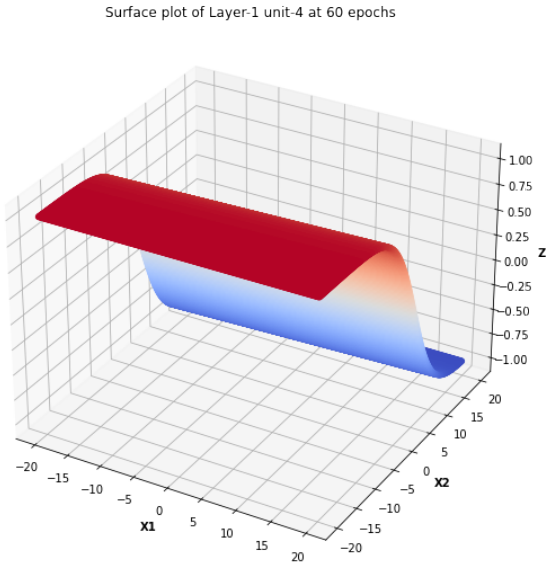

Figure 46

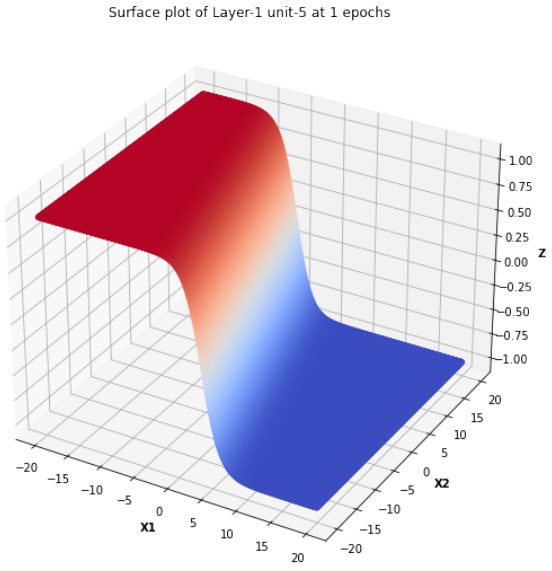

Figure 47

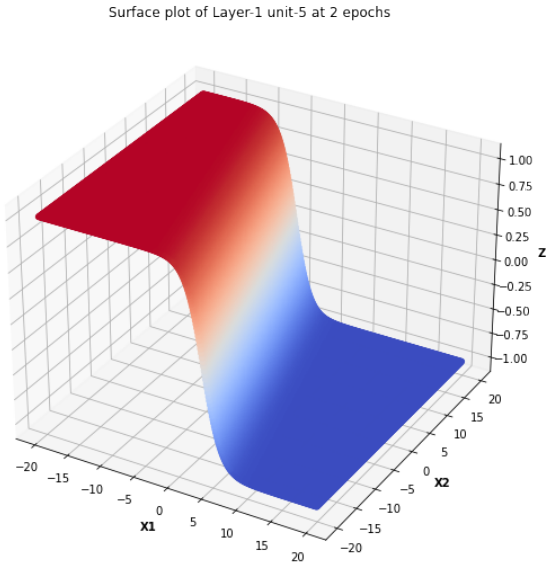

Figure 48

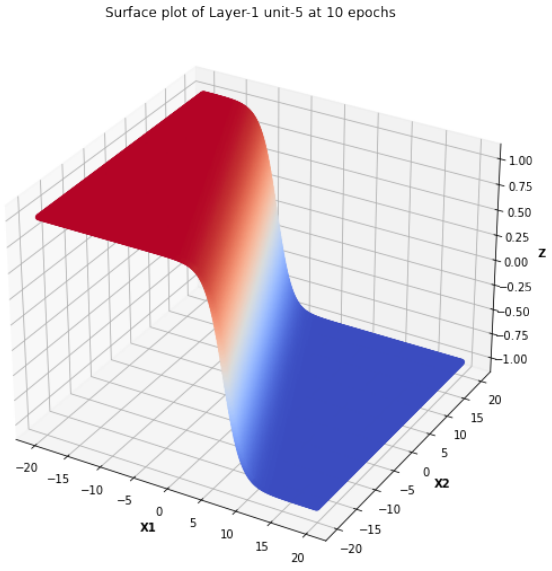

Figure 49

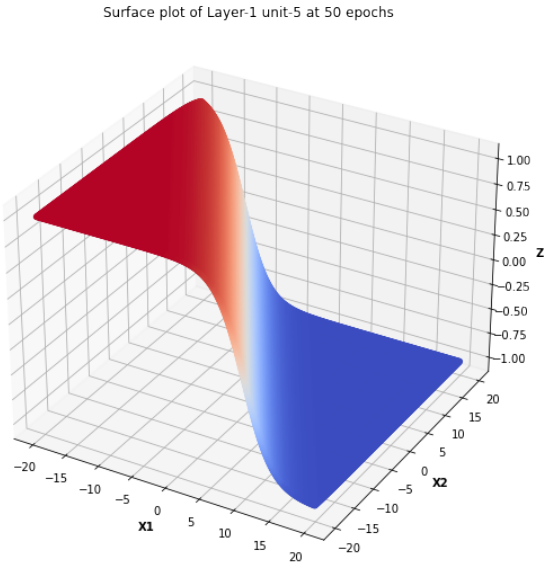

Figure 50

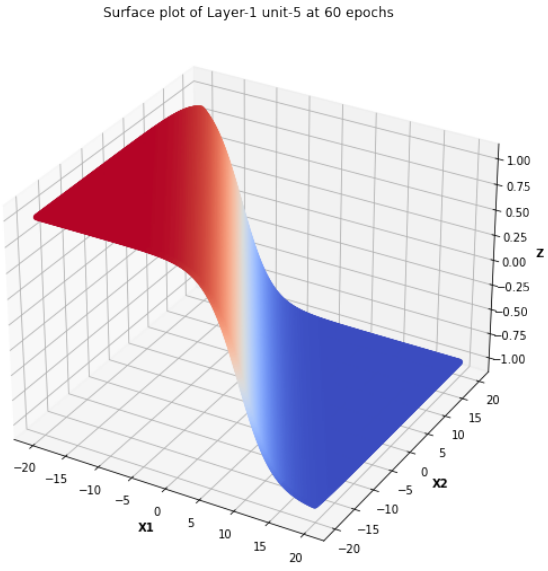

Figure 51

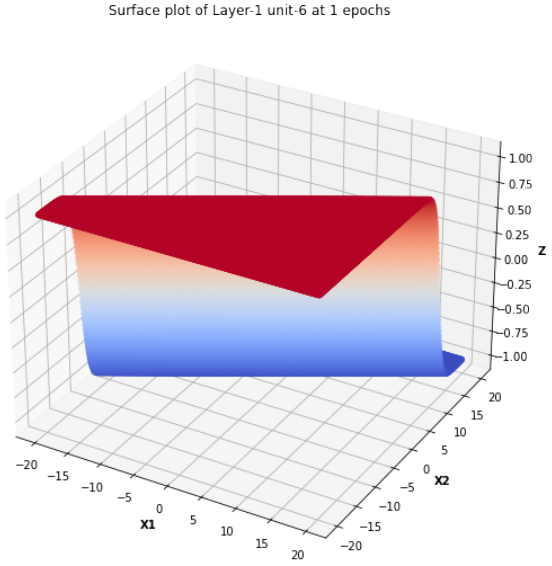

Figure 52

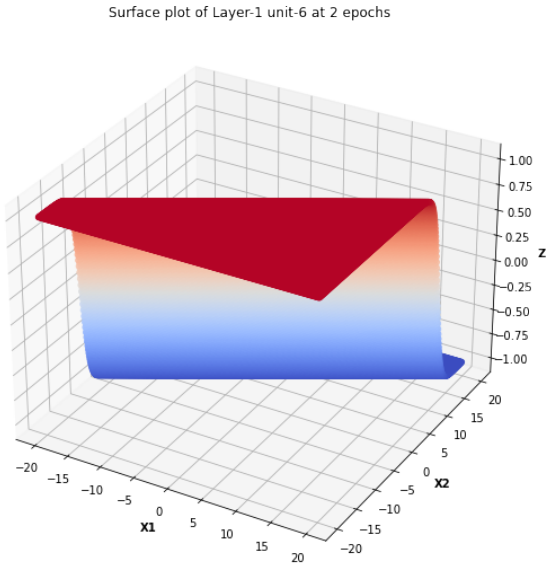

Figure 53

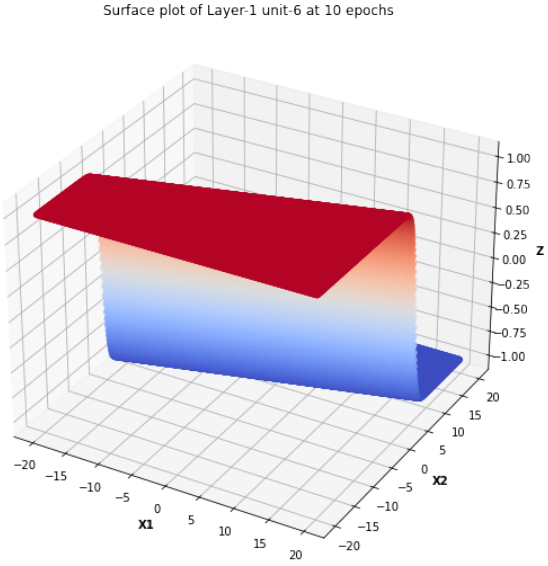

Figure 54

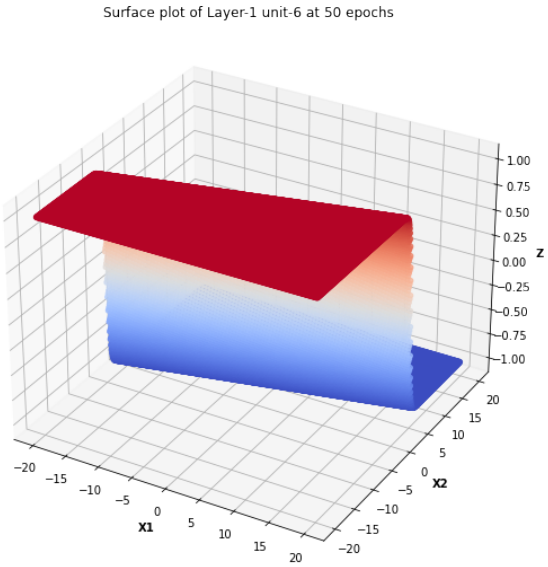

Figure 55

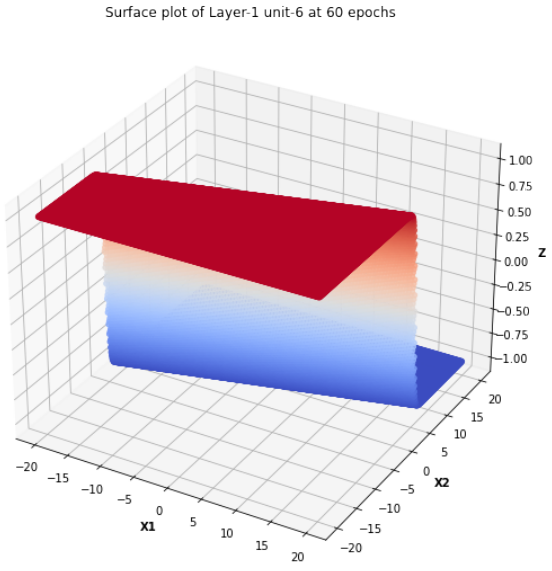

Figure 56

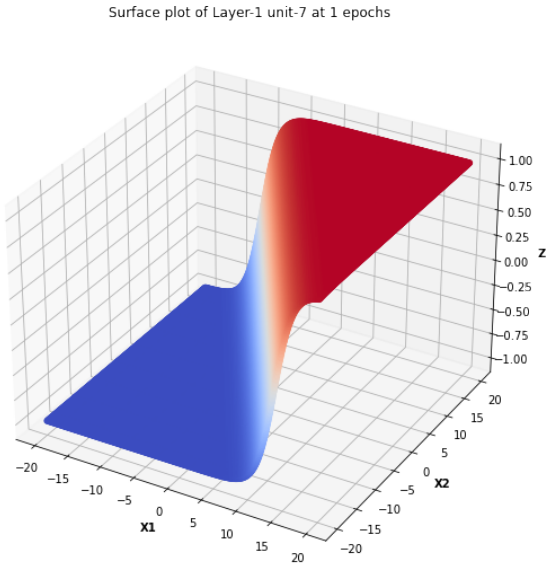

Figure 57

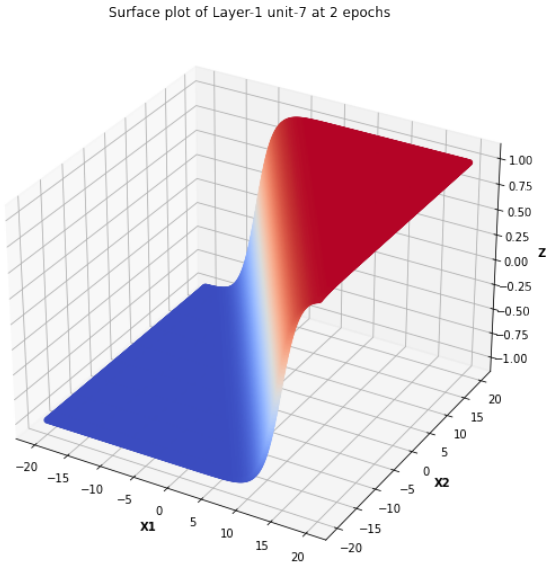

Figure 58

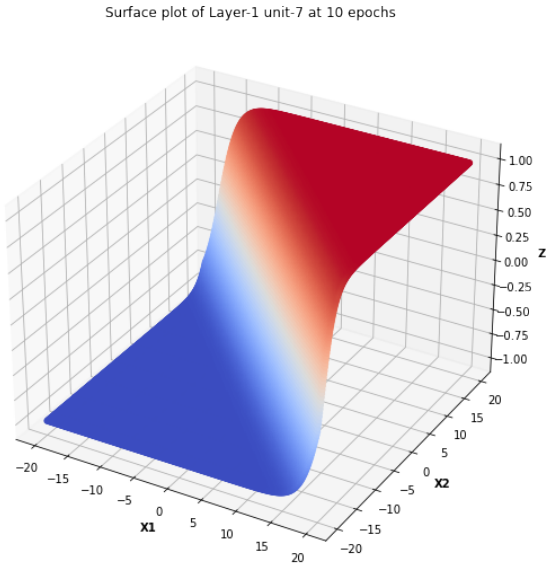

Figure 59

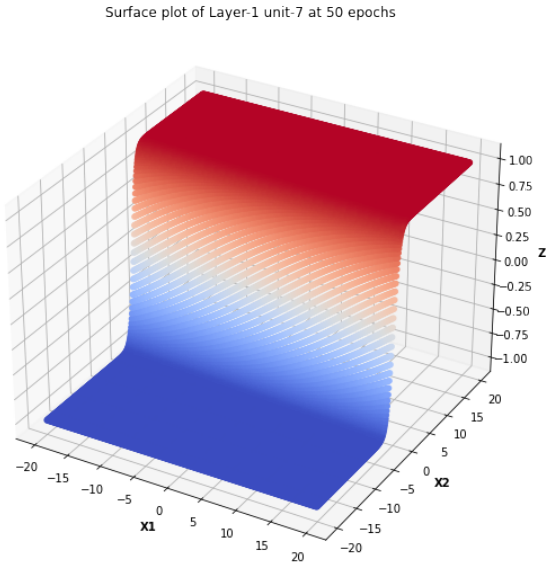

Figure 60

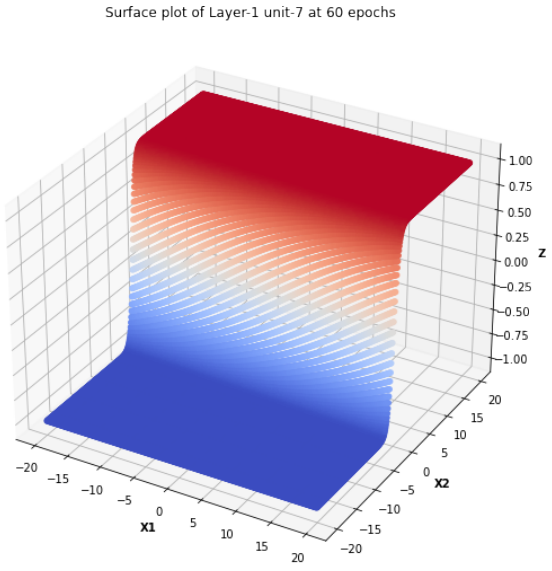

Figure 61

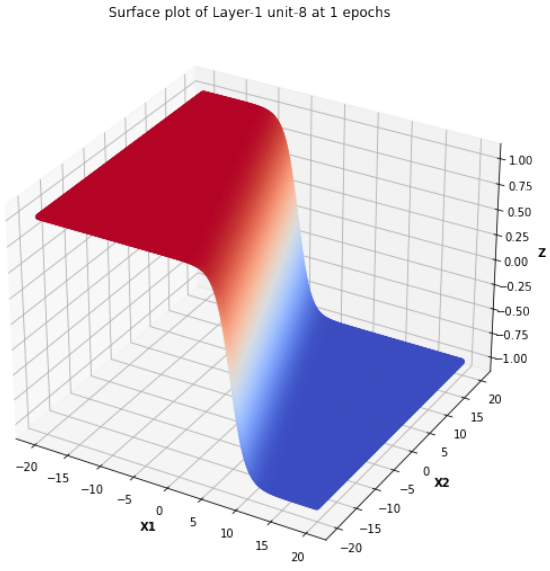

Figure 62

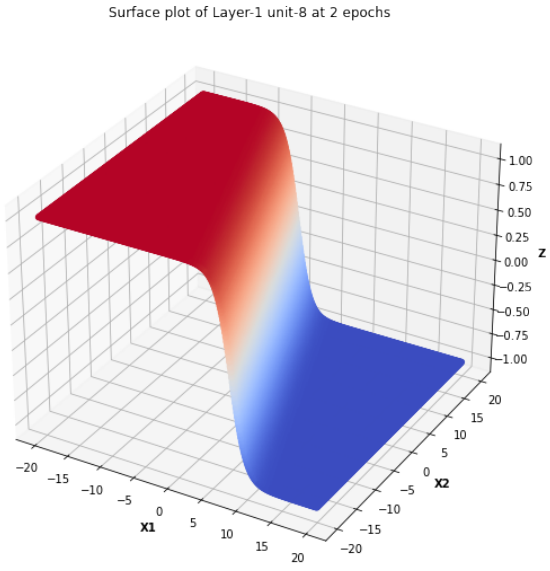

Figure 63

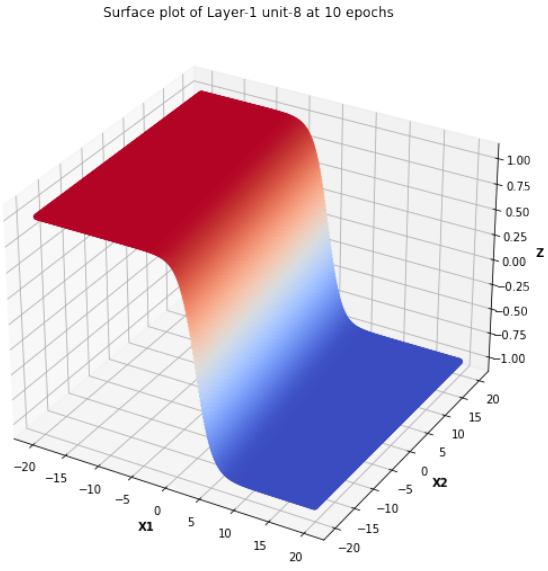

Figure 64

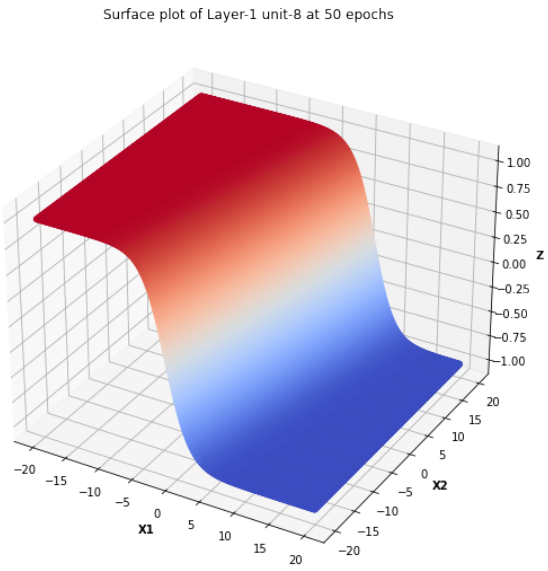

Figure 65

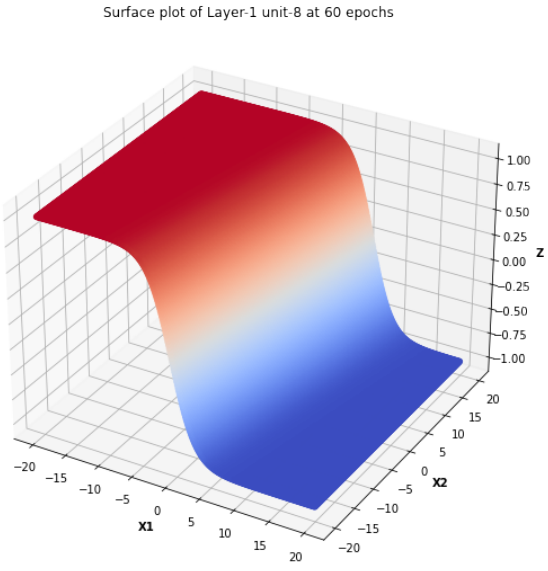

Figure 66

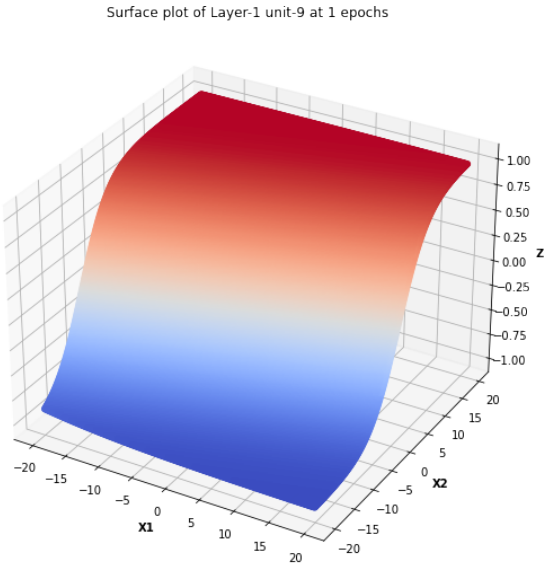

Figure 67

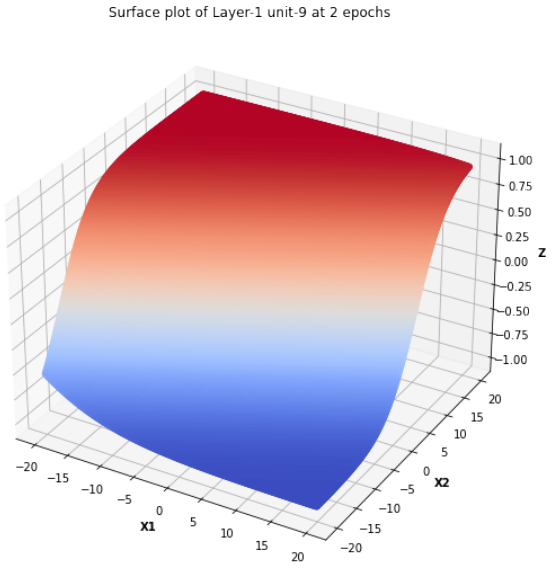

Figure 68

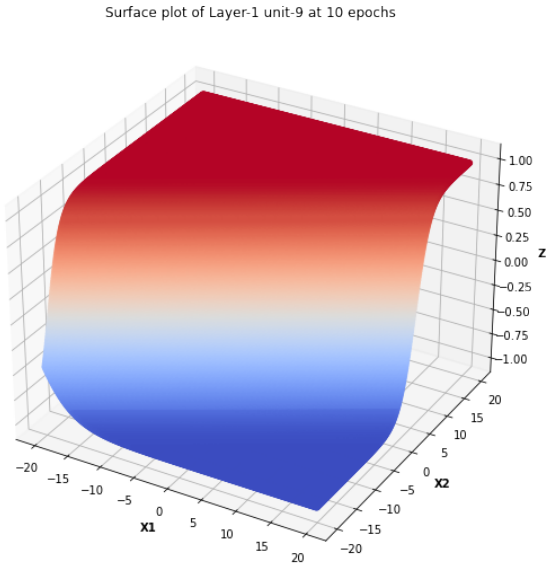

Figure 69

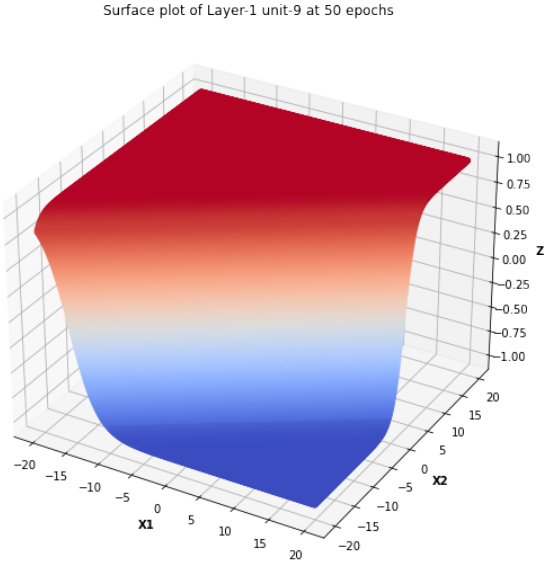

Figure 70

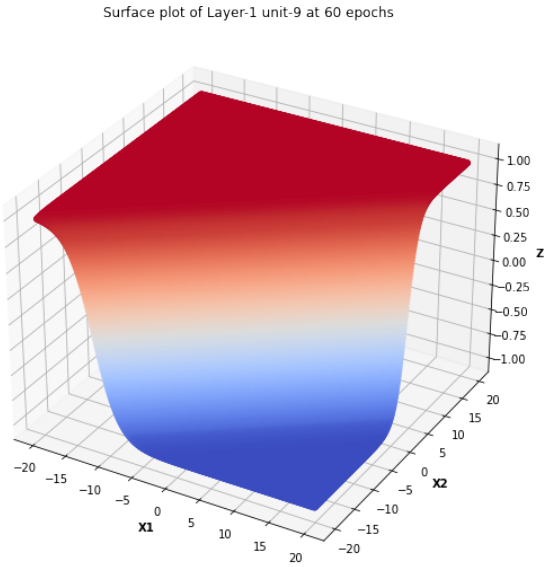

Figure 71

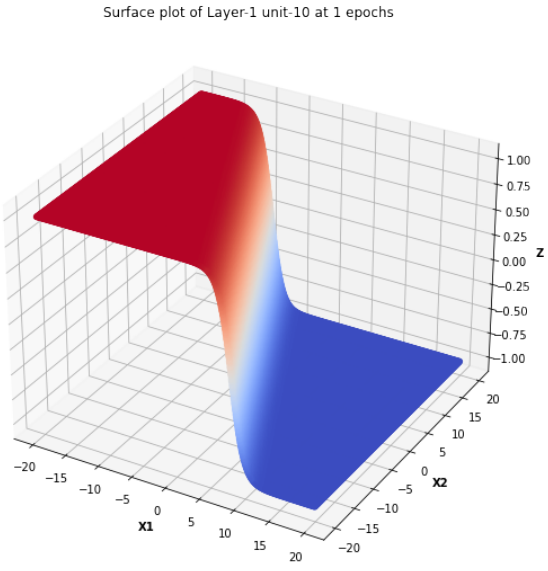

Figure 72

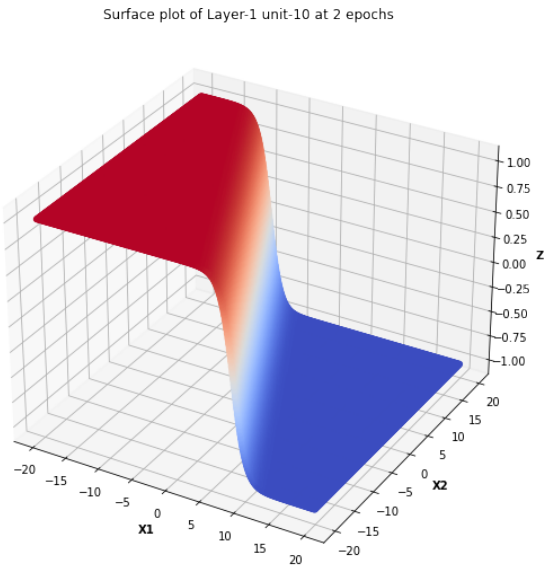

Figure 73

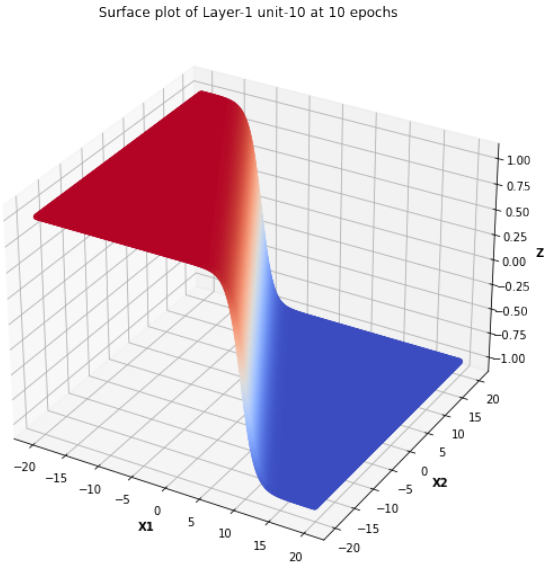

Figure 74

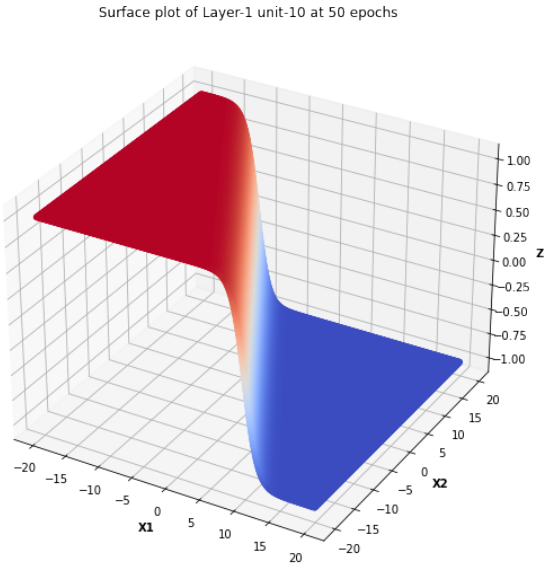

Figure 75

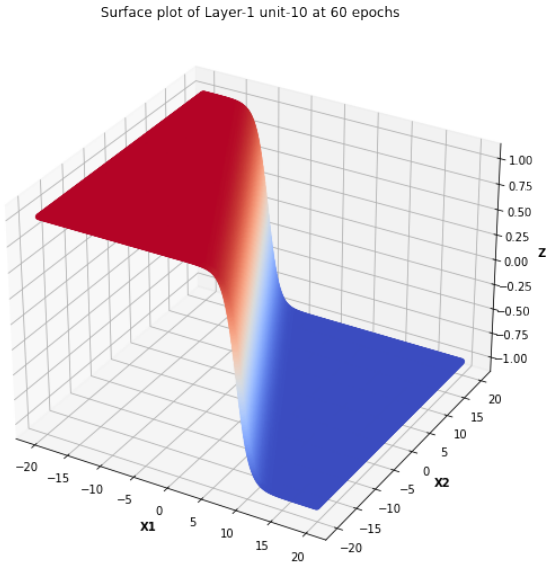
Surface plot of Layer-1 unit-10 at 60 epochs
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Z
−20
−15
−10
−5
0
5
10
15
20
X1
20
15
10
5
0
−5
−10
−15
−20
X2

Figure 76

2.4 Surface Plots of first 10 units of layer-2 at 1, 2, 10, 50 and 60 epochs

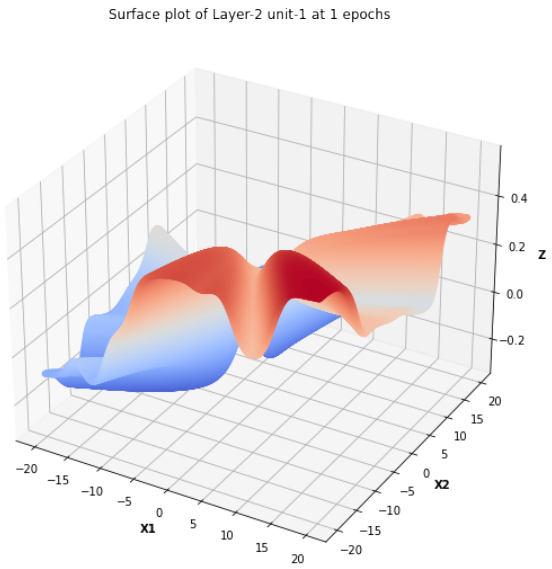

Figure 77

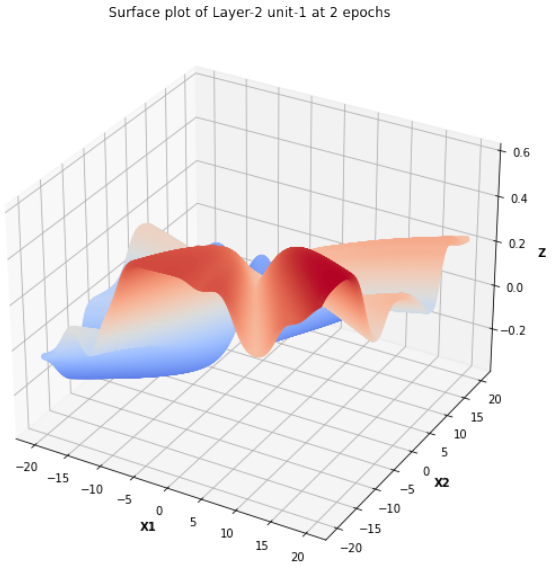

Figure 78

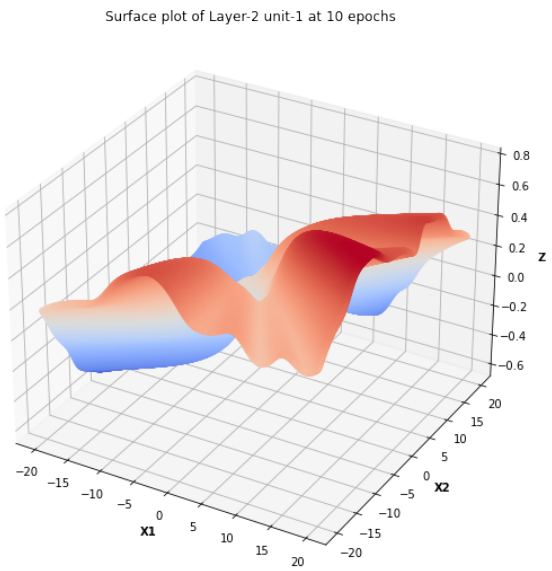

Figure 79

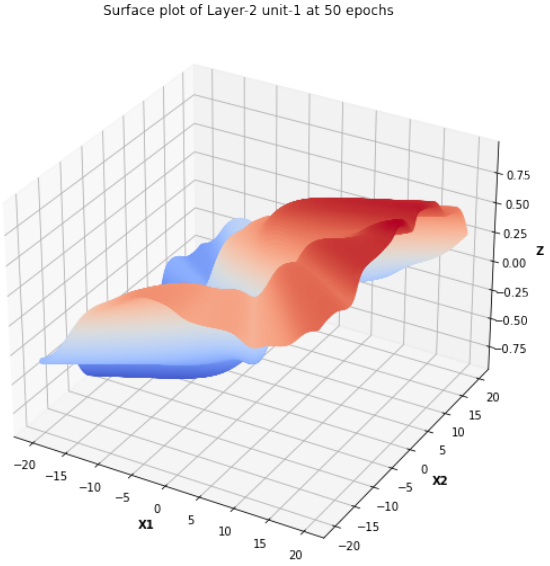

Figure 80

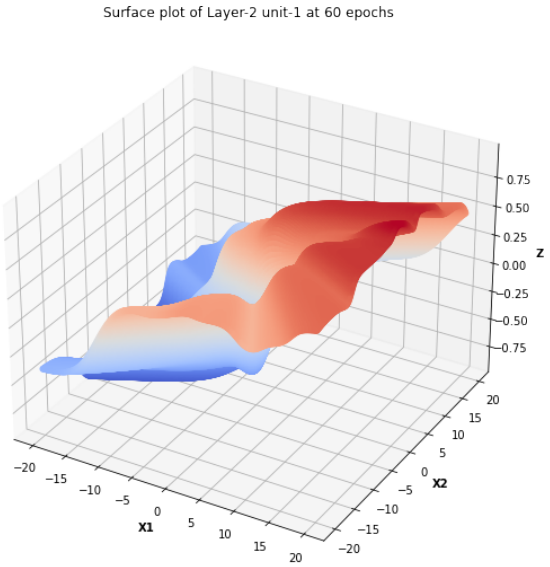

Figure 81

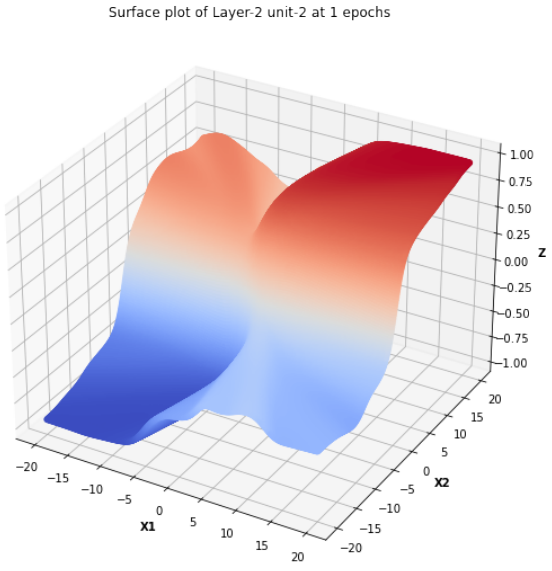

Figure 82

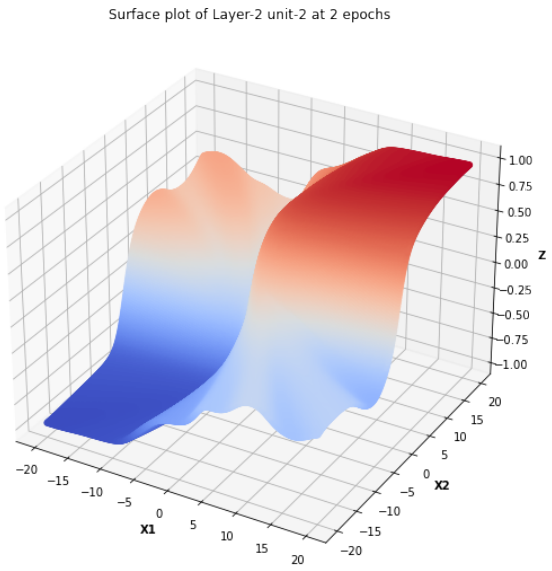

Figure 83

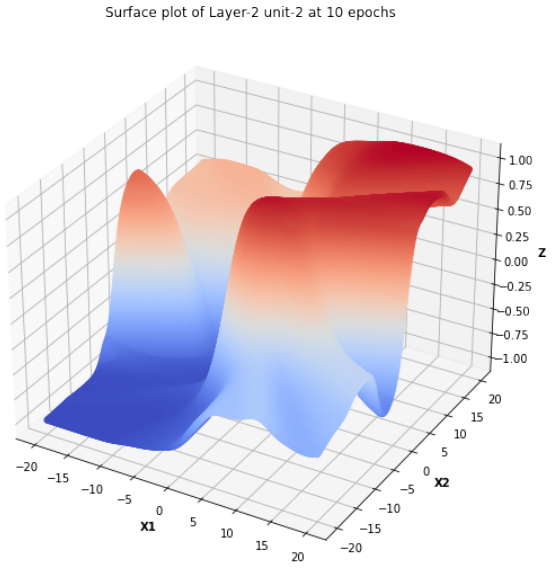

Figure 84

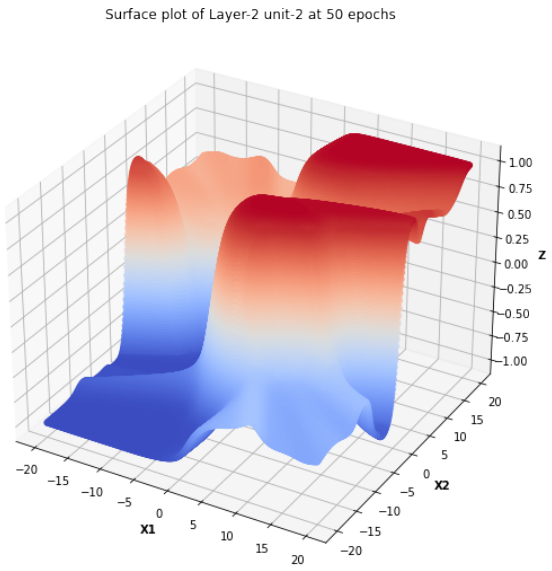

Figure 85

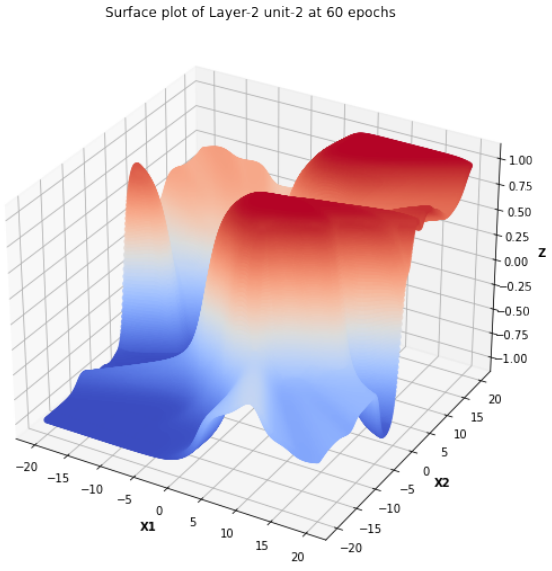

Figure 86

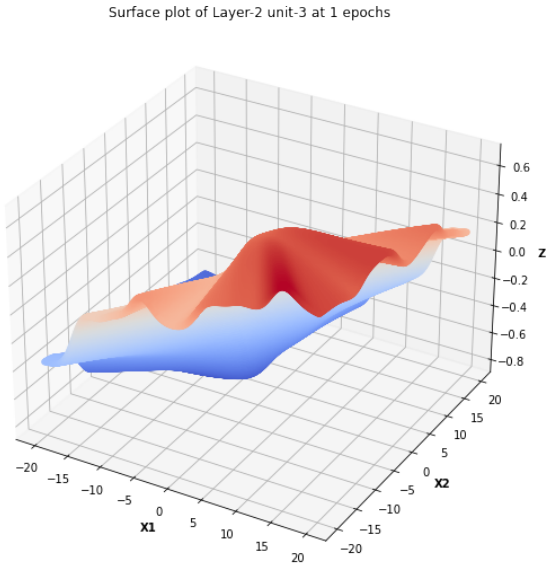

Figure 87

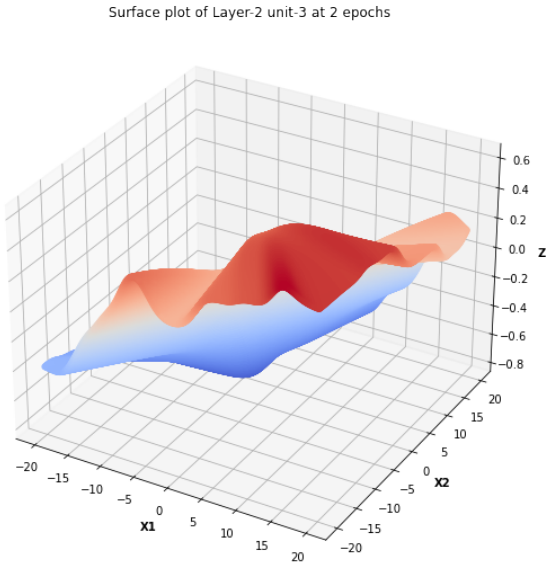

Figure 88

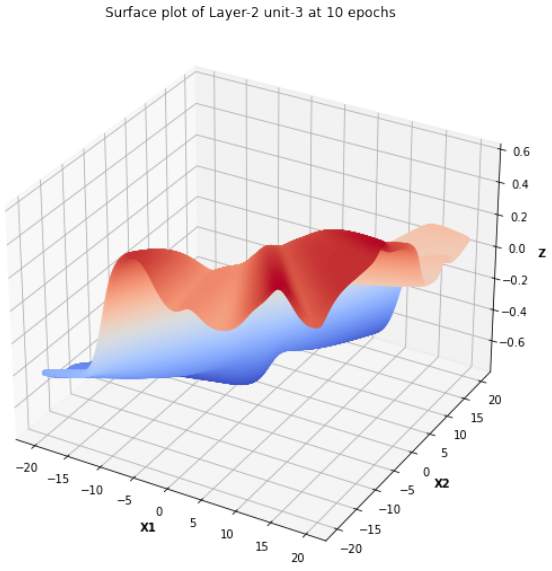

Figure 89

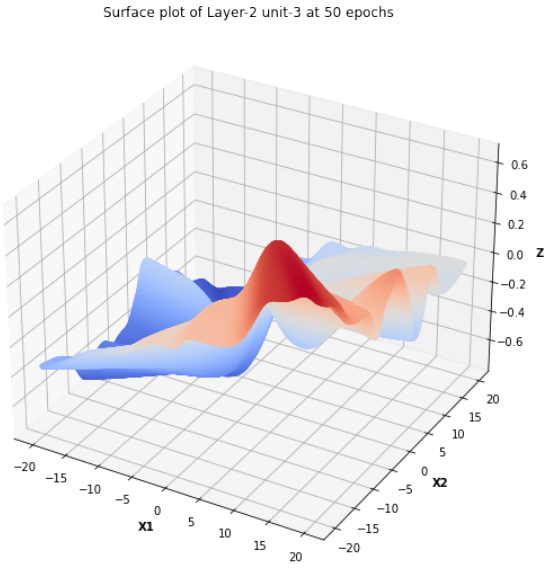

Figure 90

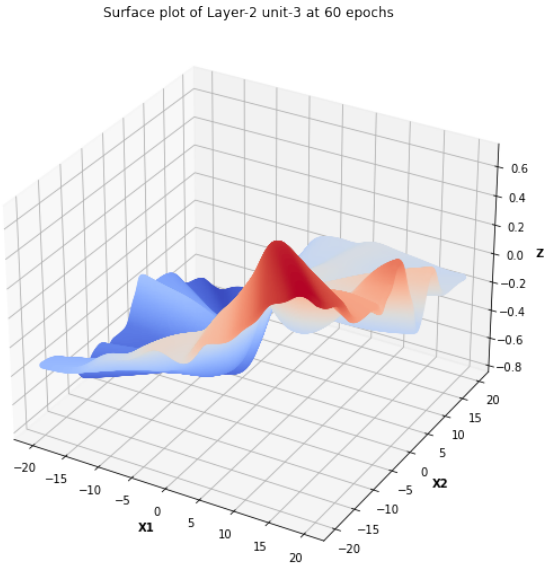

Figure 91

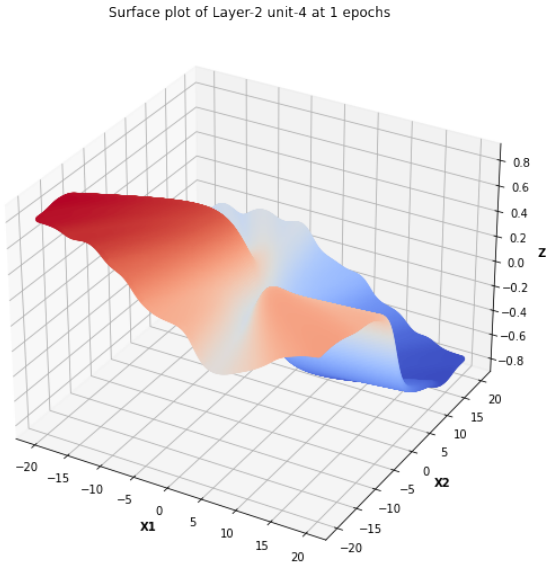

Figure 92

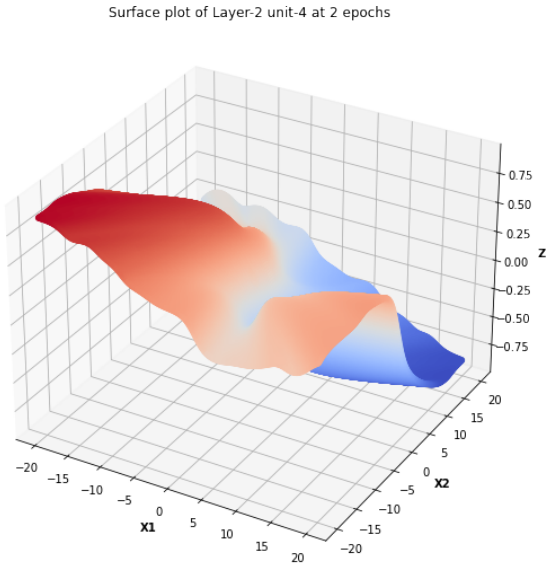

Figure 93

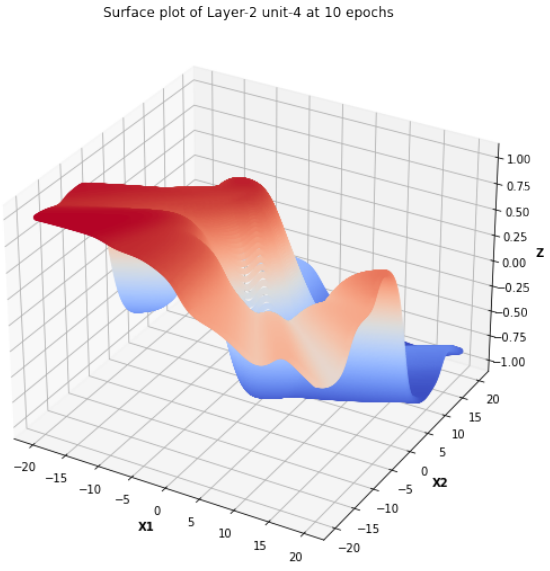

Figure 94

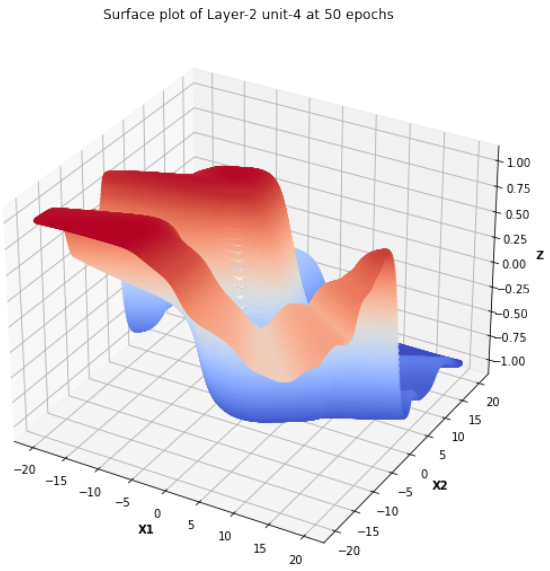

Figure 95

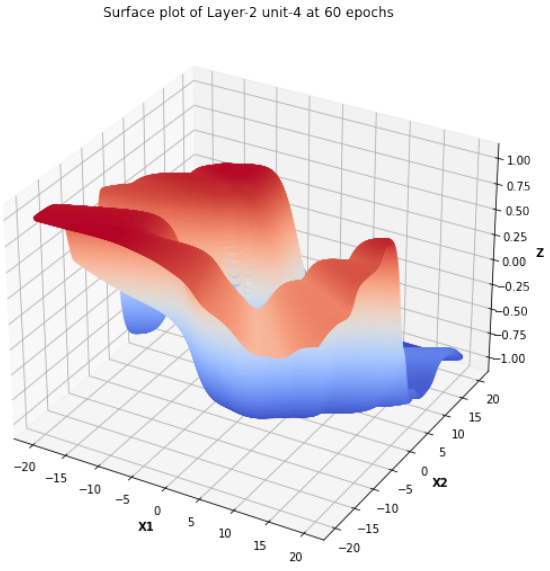

Figure 96

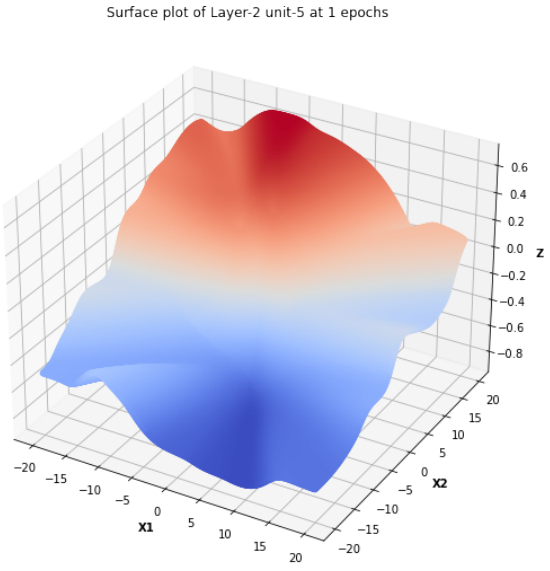

Figure 97

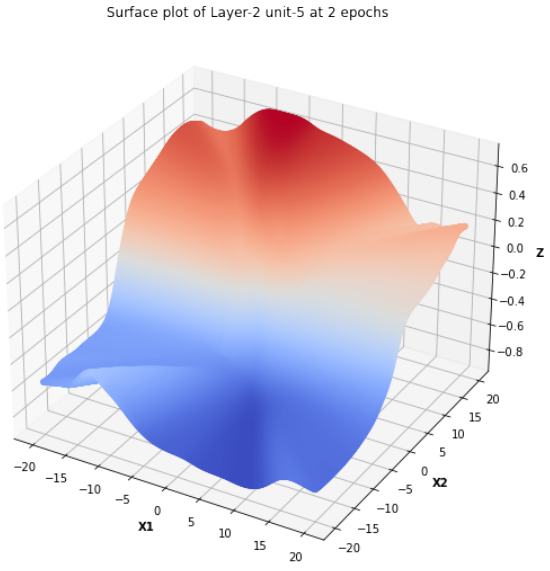

Figure 98

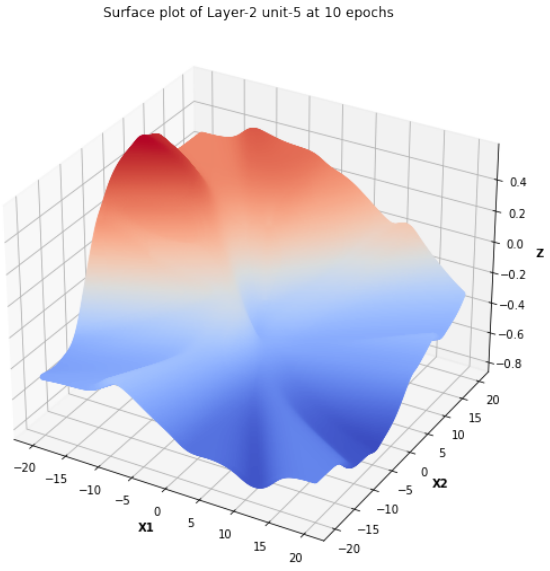

Figure 99

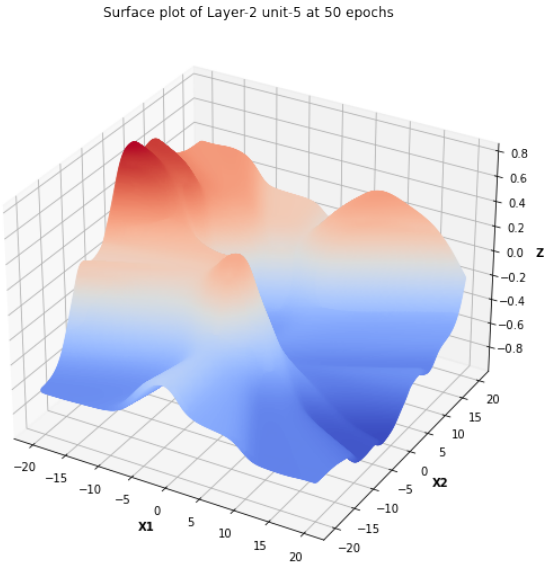

Figure 100

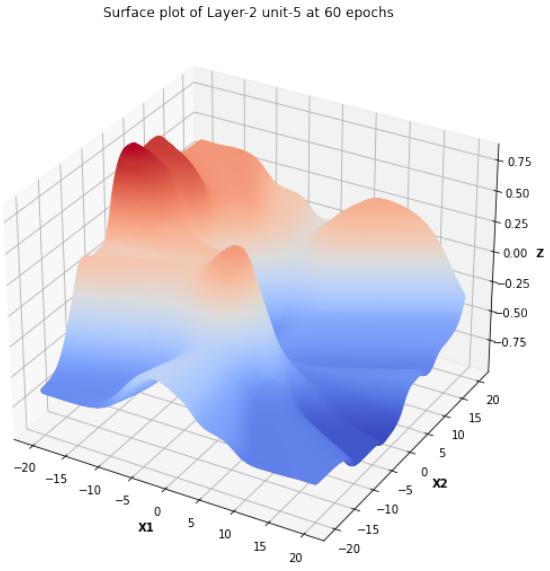

Figure 101

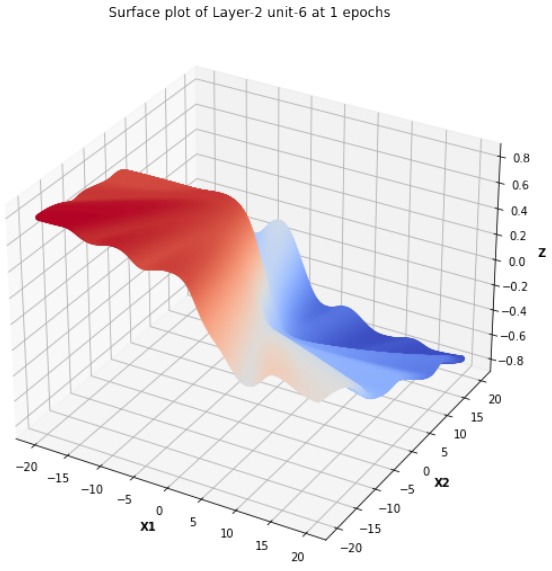

Figure 102

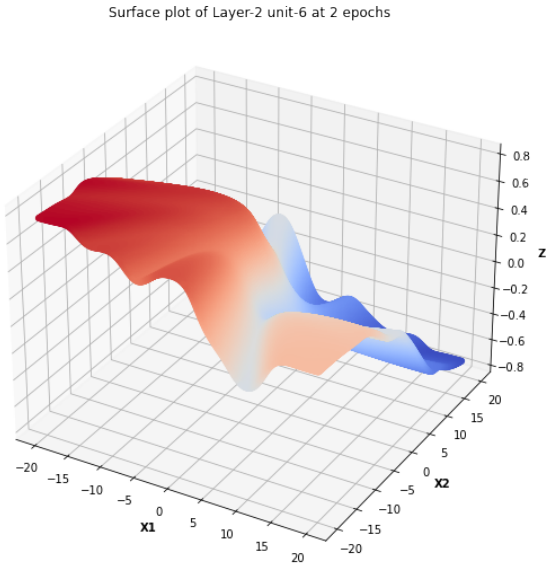

Figure 103

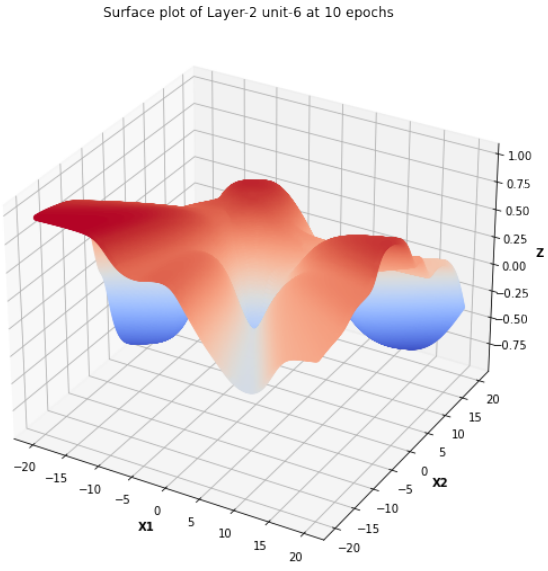

Figure 104

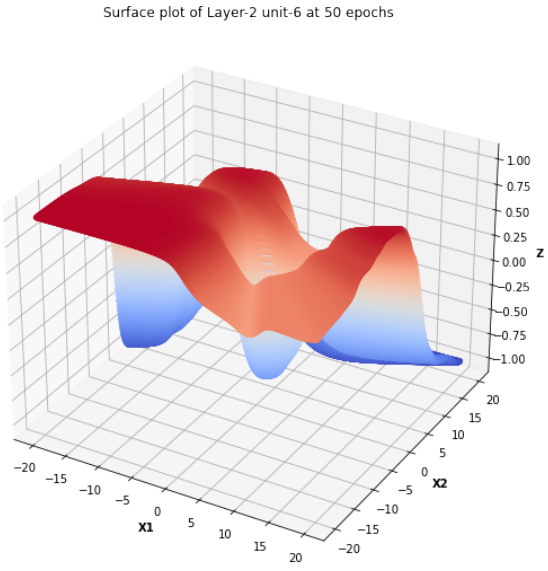

Figure 105

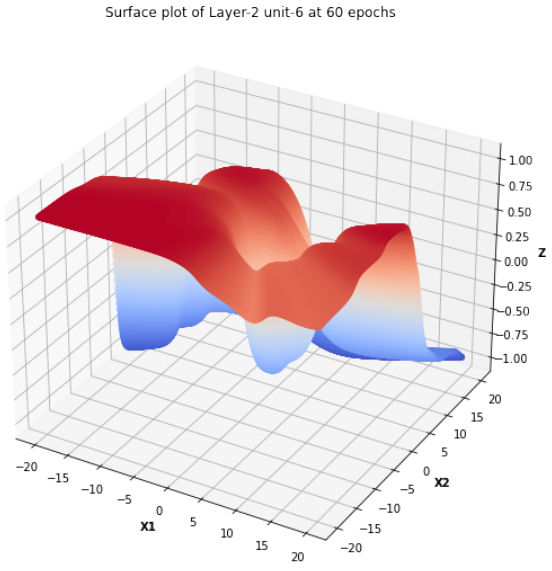

Figure 106

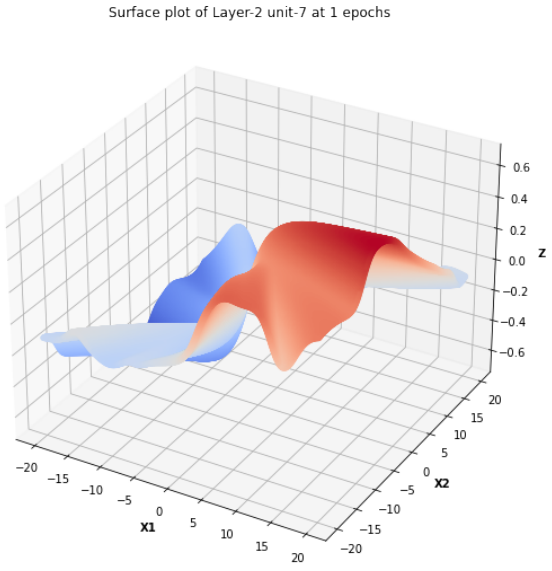

Figure 107

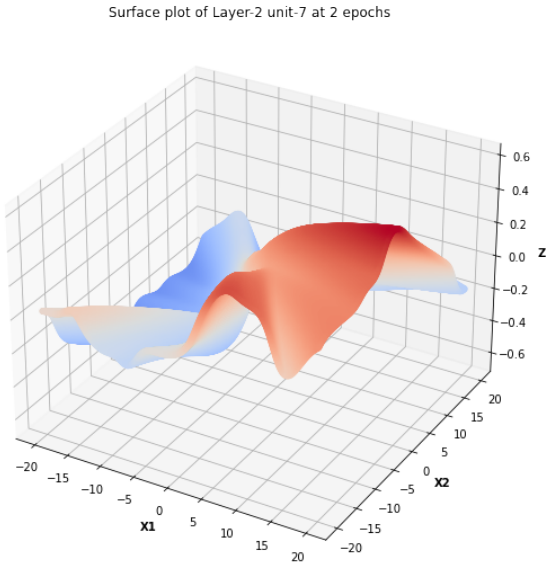

Figure 108

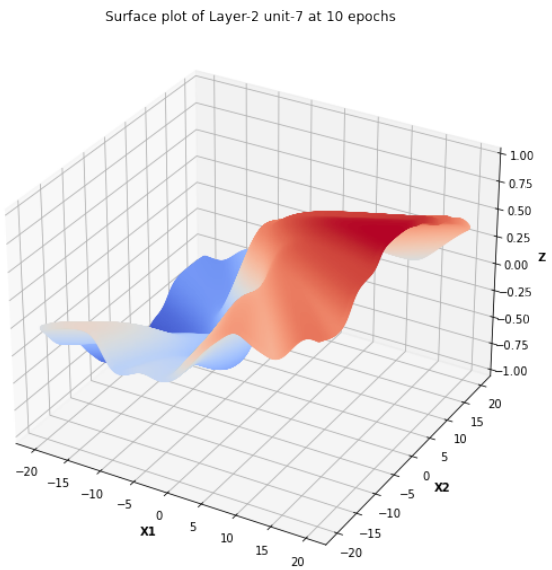

Figure 109

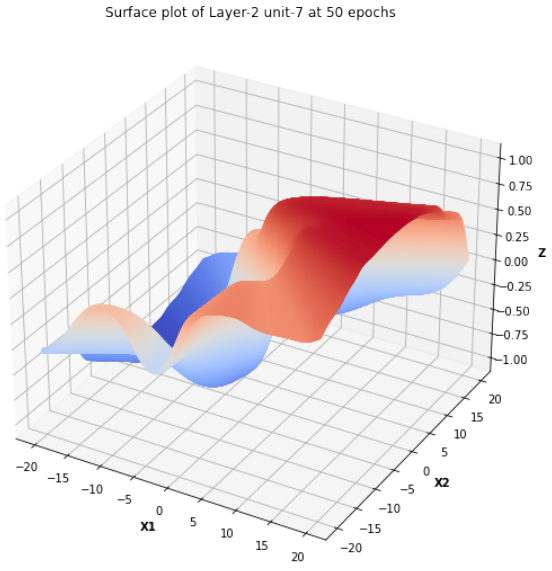

Figure 110

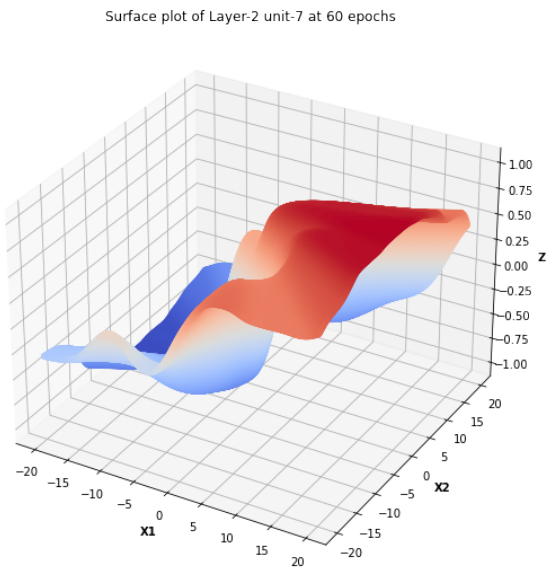

Figure 111

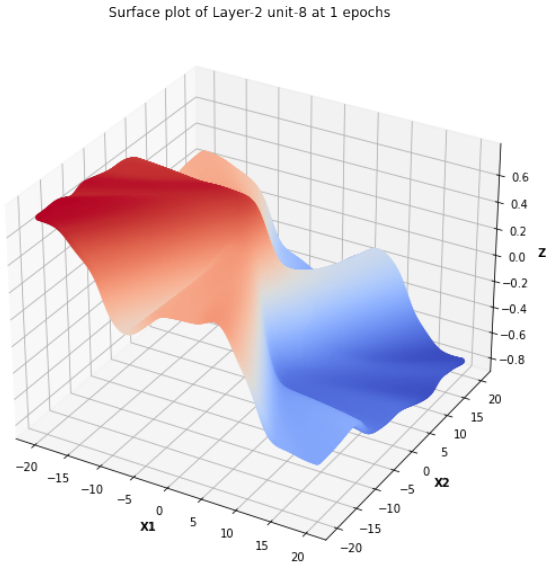

Figure 112

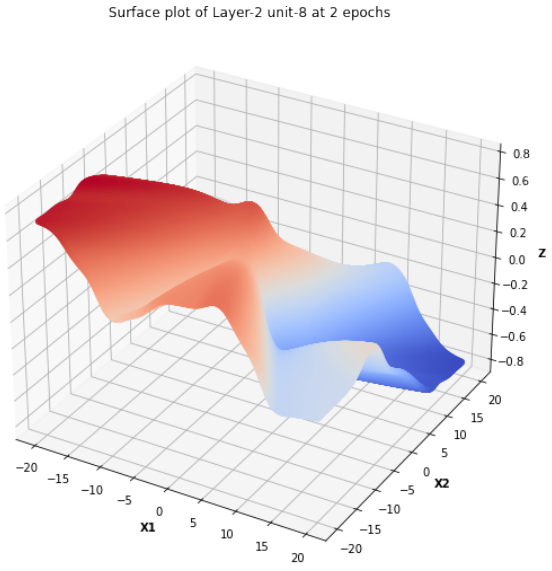

Figure 113

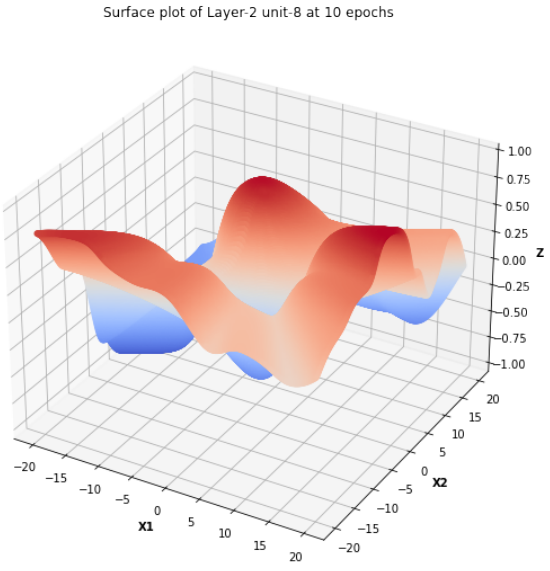

Figure 114

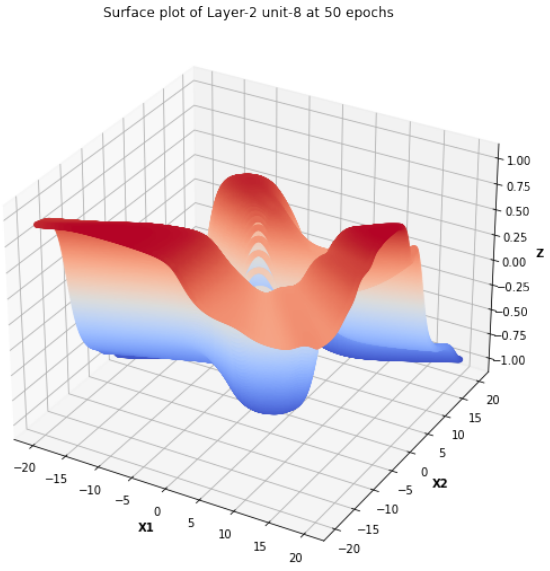

Figure 115

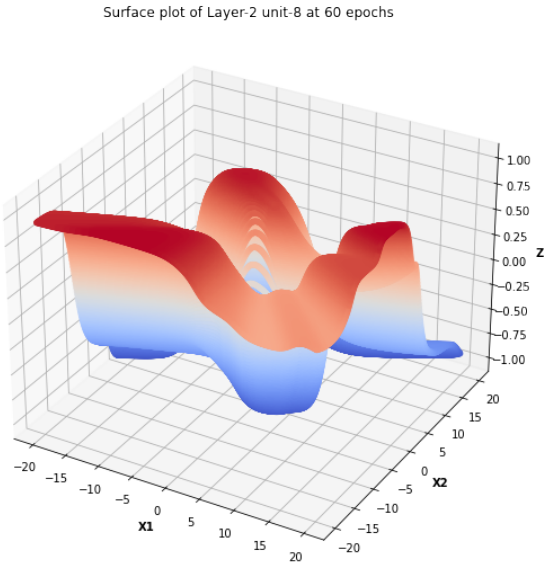

Figure 116

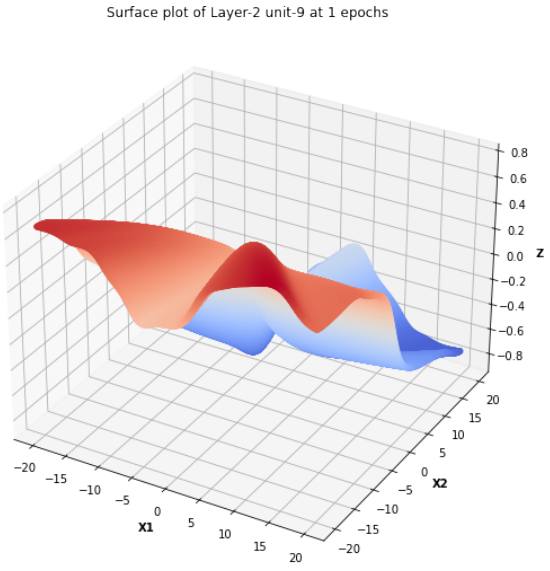

Figure 117

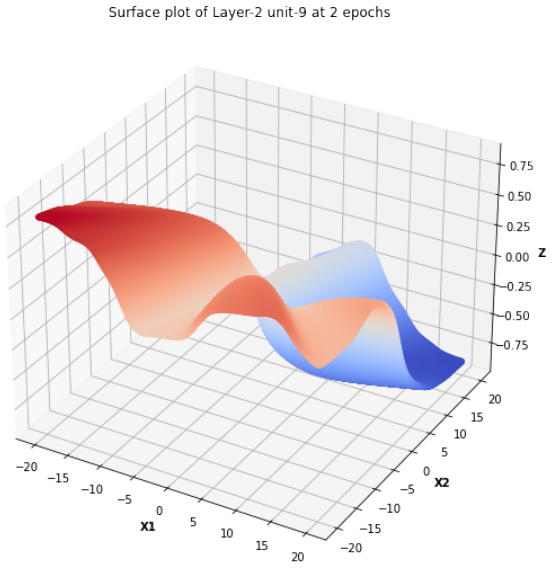

Figure 118

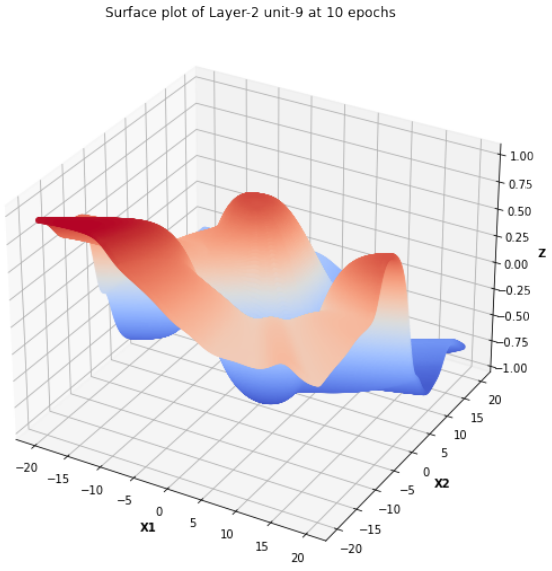

Figure 119

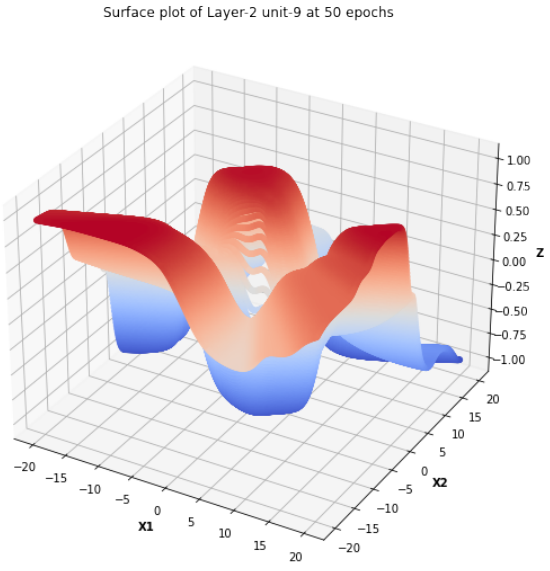

Figure 120

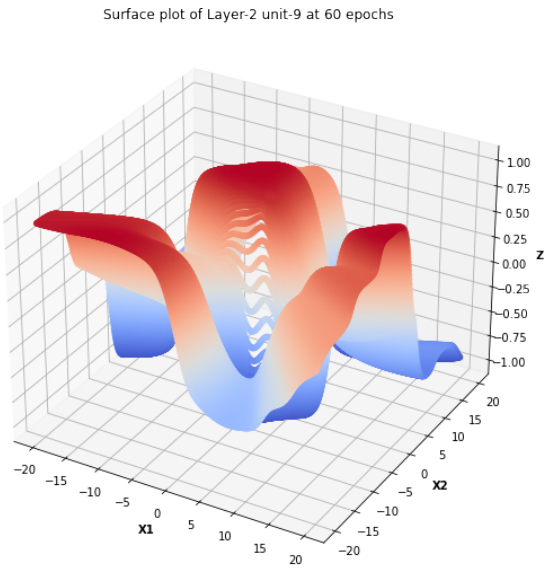

Figure 121

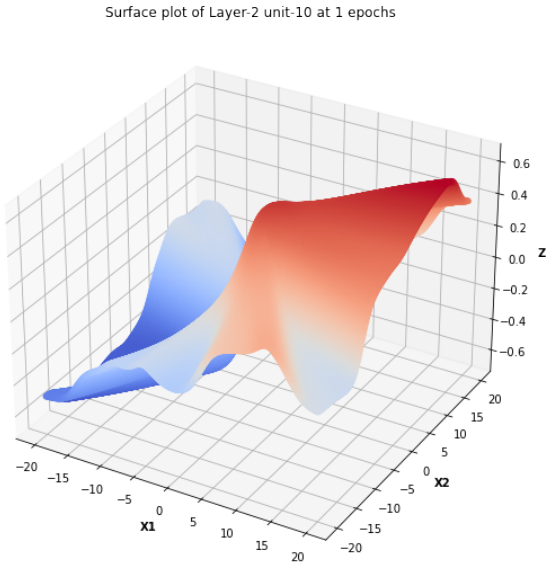

Figure 122

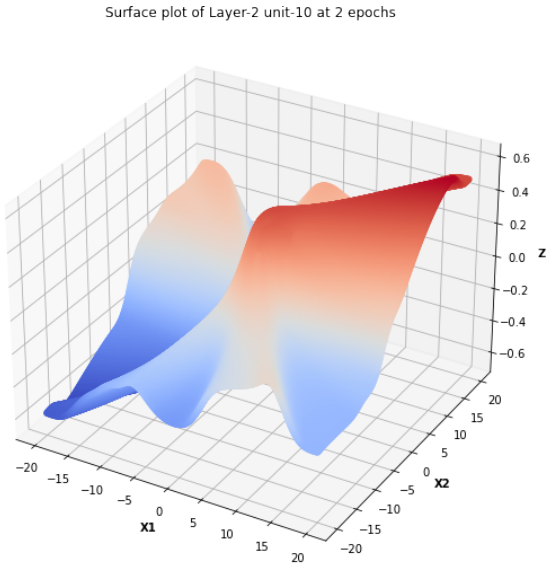

Figure 123

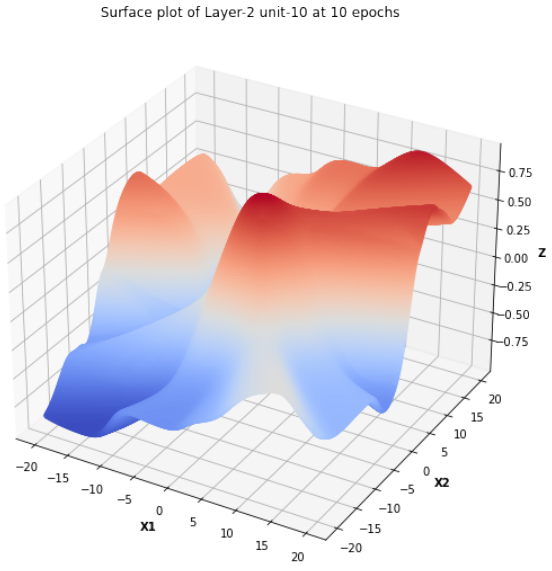

Figure 124

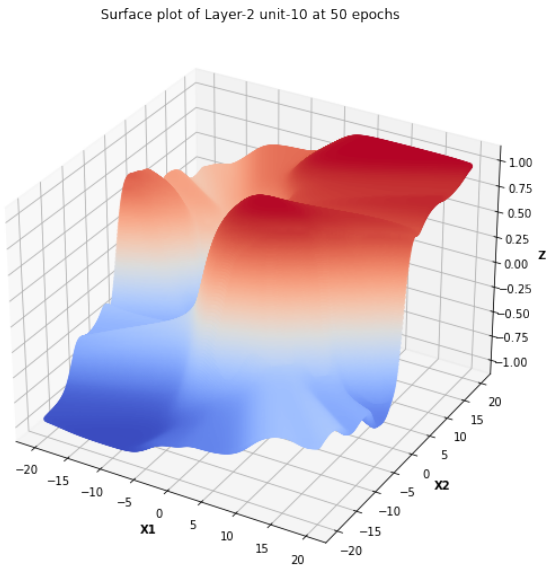

Figure 125

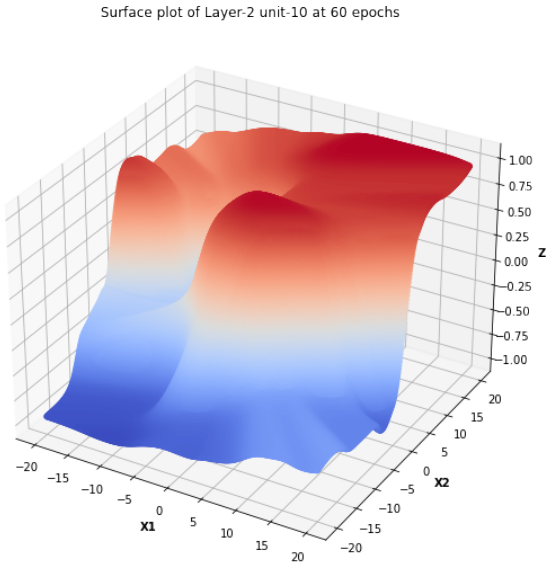
Surface plot of Layer-2 unit-10 at 60 epochs
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Z
−20
−15
−10
−5
0
5
10
15
20
X1
20
15
10
5
0
−5
−10
−15
−20
X2

Figure 126

2.5 Surface plots of output layer after 1, 2, 10, 50 and 60 epochs

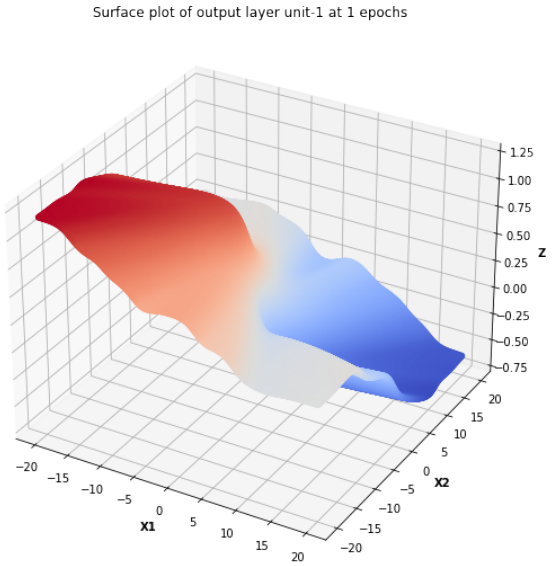

Figure 127

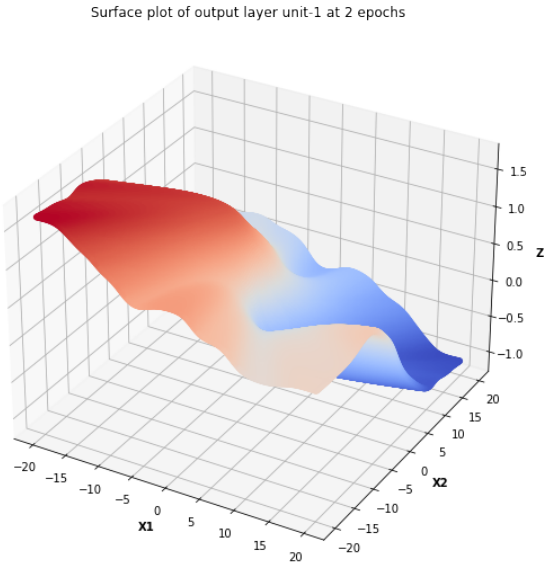

Figure 128

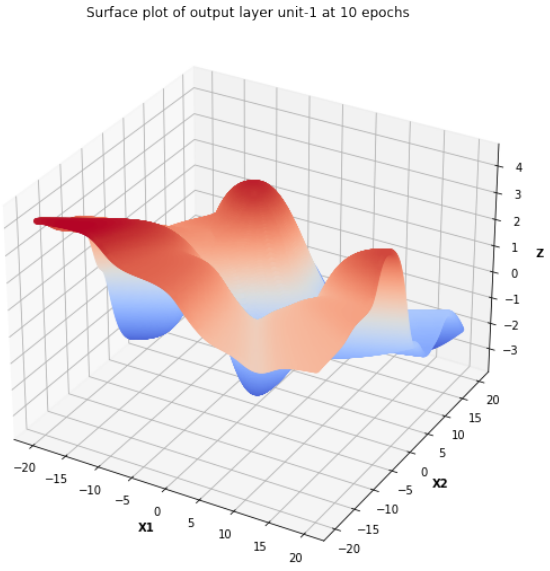

Figure 129

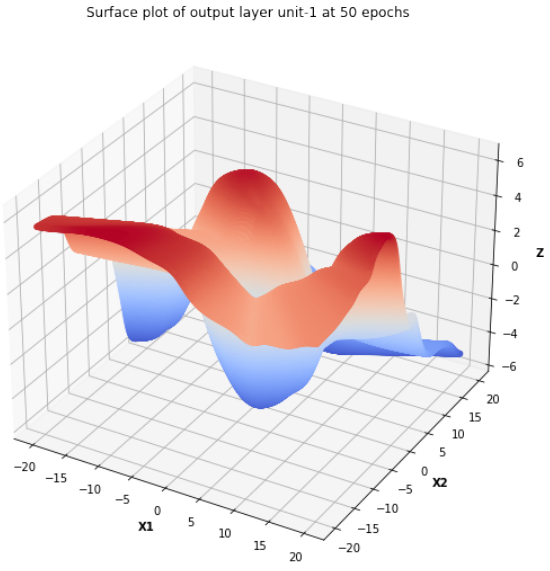

Figure 130

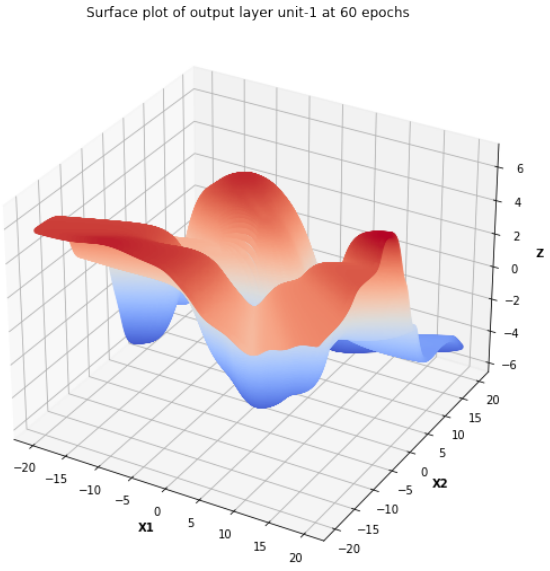

Figure 131

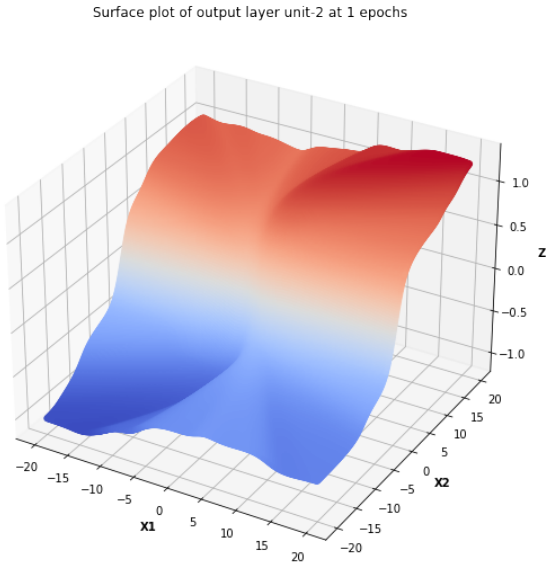

Figure 132

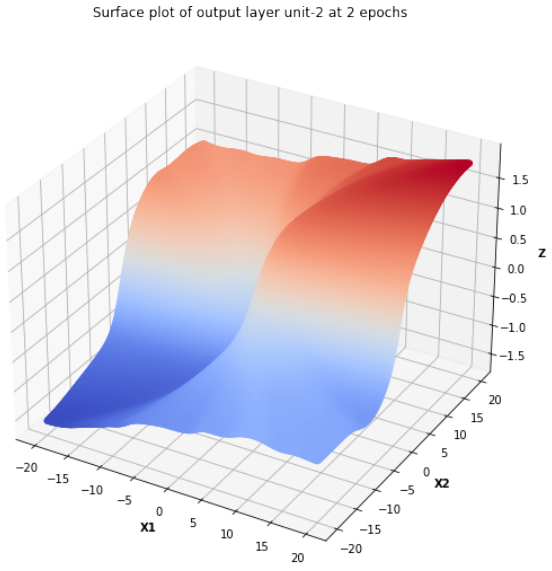

Figure 133

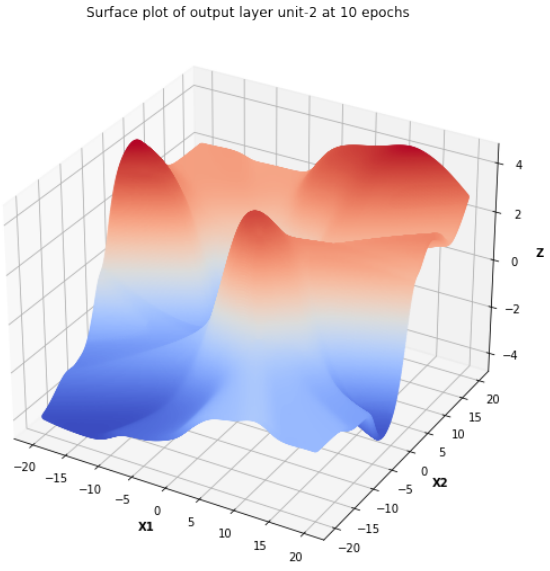

Figure 134

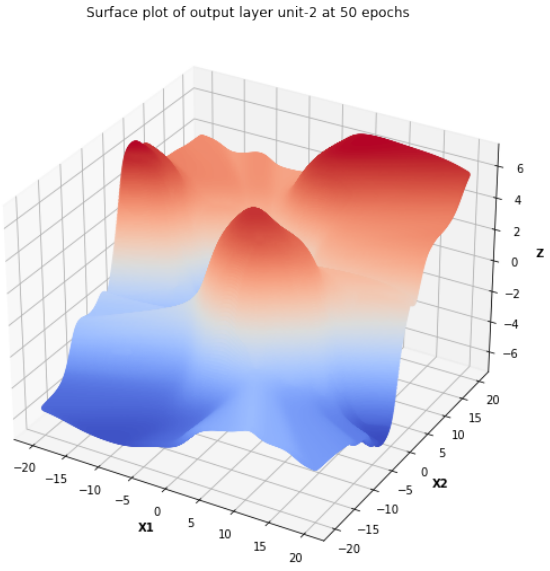

Figure 135

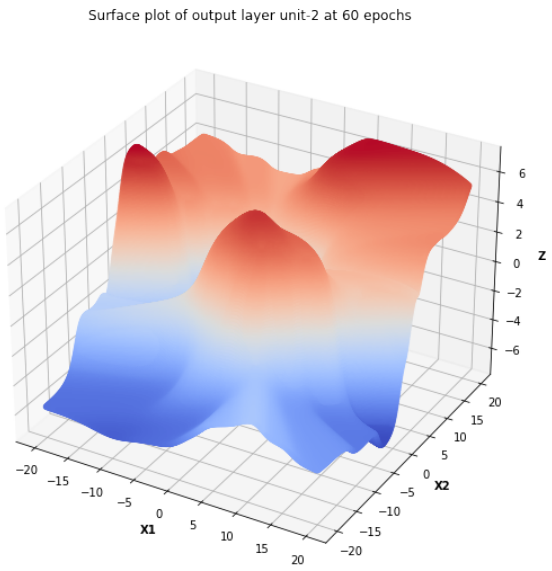

Figure 136

3 Task 2b

In this task we are asked to classify image data. The image data is provided in the form of a 60 dimensional embedding, along with corresponding image labels. The task is a 5 class multi-class classification problem.

3.1 Preprocessing

As the first step in our solution, we first preprocess the data to ensure that the inputs are of a suitable range for neural networks (-1,1). For this, we leverage the *StandardScaler* class from sklearn in order to mean center and normalize by the standard deviation. We initially create a train, val, test split (0.8, 0.1, 0.1) and then fit the StandardScaler to the train dataset. One important point to note is that the train, val and test splits are created in a stratified manner, in order to preserver the class proportions.

3.2 Model Architecture

Next, in our pipeline we explored various configurations of the 2 layer, multi-layer feed forward neural network. The neural network consists of $tanh()$ activated neurons for 2 hidden layers, followed by a softmax activated output layer. The model was trained in pytorch with the *CrossEntropyLoss*. It's important to note that the pytorch CrossEntropyLoss internally activates the raw outputs from the final layer of the neural network with a softmax activation. Hence it is prudent to provide raw logits to this loss function in order to get expected behaviour.

3.3 Hyperparameter Tuning

We tuned the model's hyperparameters with the help of the *weight and biases* package (wandb in our code). In particular, for our implementation, we chose to look at the following ranges of hyperparameters for our model.

- number of hidden units in layer 1: [32, 256, 1024]
- number of hidden units in layer 2: [8, 32, 64]
- optimizer: [Adam, GeneralizedDeltaRule, DeltaRule]

For all of these rules, the weights were randomly initialised with the same random state and the learning rate was set to 0.001 (based on empirical analysis for speed of iteration).

Figure 137 highlights the model hyperparameters that led to the best performance in terms of validation accuracy.
As we used *weights and biases*, a detailed report of our hyperparameter tuning can be found in the following link - `https://wandb.ai/vimaled17b055/CS6910-Assignment1-Task2b/sweeps/k1j42k58?workspace=user-vimaled17b055`

3.4 Effect of Optimizer

To isolate the effect of the optimizer, we chose a set of hyperparameters based on the observations we had from the hyperparameter tuning stage. It was clear to choose the following hyperparemeters:

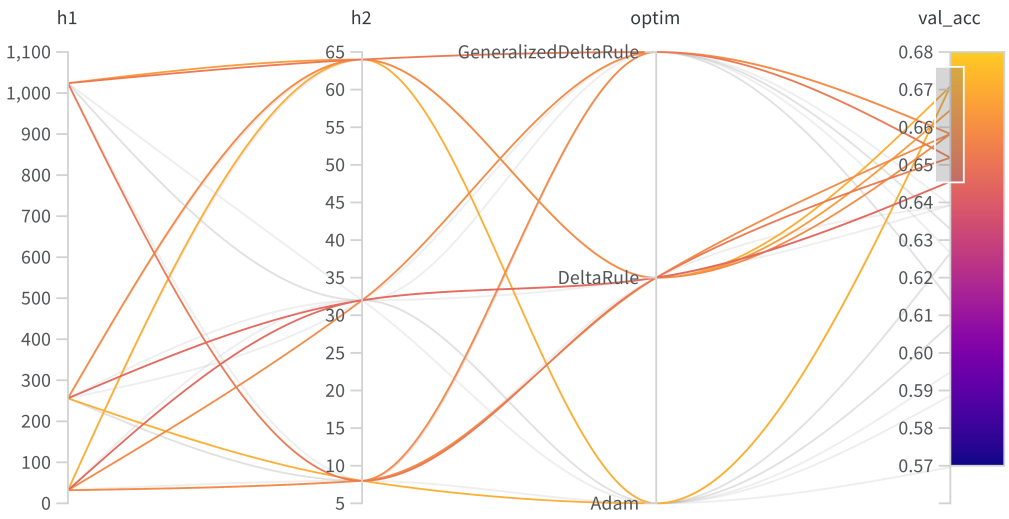

Figure 137: Model Hyperparameters vs Validation accuracy.

- h1: 1024 nodes
- h2: 64 nodes

After having set the number of hidden units in the 2 hidden layers, we then proceeded to train the model 3 different times with each of the 3 optimizers - GeneralizedDeltaRule, DeltaRule and the Adam optimizer. Same as before, we randomly initialised the weights of the network with the same random state as before. The models were trained for 30 epochs each, which we found was enough for convergence. As before, we trained the models with the pattern mode of training (1 sample per batch) and metrics were evaluated at the end of an epoch.

In Figure 138 we have visualized the train and test confusion matrices as well as the training error/loss curves for each of the 3 optimizers. Metrics are shown in table **??**. We can observer the following:

- The delta Rule took the longest to train the model (in fact, from this graph, it is evident that the model could have been trained further) - however, it achieved the highest test accuracy (66.46%), indicating that it did not overfit to the training data.
- The Generalized Delta rule also performed reasonably well at 65.19%. However, signs of overfitting can be observed in from the learning curve graph

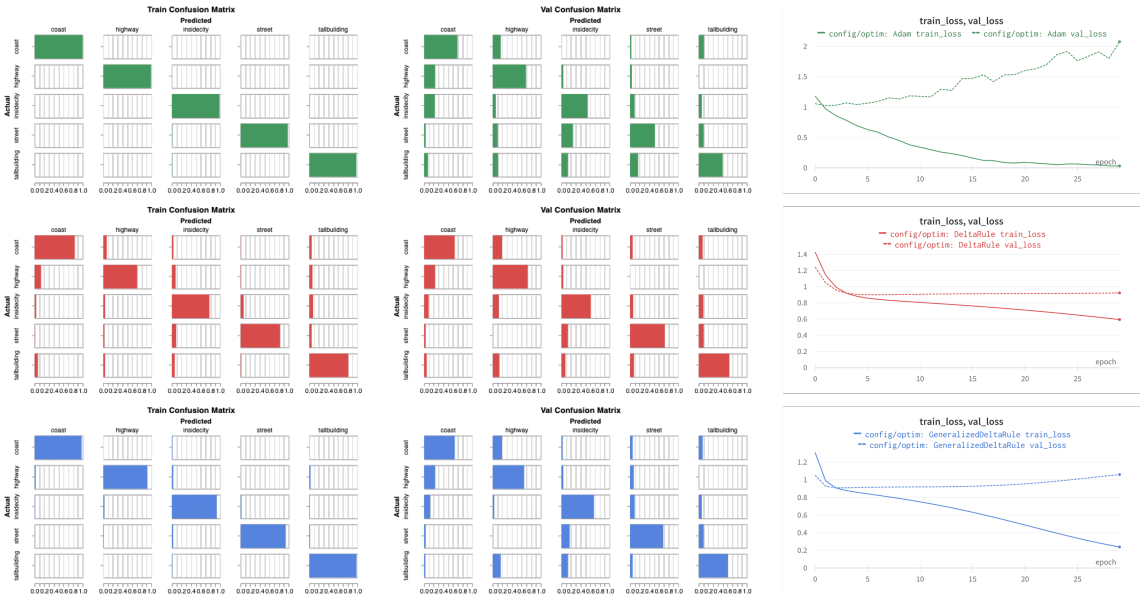

Figure 138: Comparison of Various Optimizers, (Green-Adam, Red-DeltaRule, Blue-GeneralizedDeltaRule)

- The Adam optimizer had the highest training accuracy - indicating that it was able to optimize the fastest amongs the three optimizers. However, this resulted in severe overfitting, leading to poor test accuracy performance of 58.66%.

Optimizer	**Train Accuracy**	**Test Accuracy**
GeneralizedDeltaRule	95.48	65.19
DeltaRule	79.68	66.46
Adam	99.05	58.66

Table 1: Accuracies for various optimizers

3.4.1 Observations from the confusion matrics

The confusion matrices provide useful insights to class imbalance and confusion. In this case, it is evident that the *highway* and *coast* classes get confused a decent amount. We would probably need more examples of each of these classes in order to make the model more robust to these classes.

3.4.2 Epochs for convergence

The training loss curves for each optimizer are shown in figure 139. As seen before, we can observe that the Adam optimizer is able to converge the fastest, followed by the GeneralizedDeltaRule (SGD with momentum) and then finally by the Delta Rule. This makes sense as the Adam and GeneralizedDeltaRule optimizers are adaptive ones that both have momentum terms which help to speed up convergence.

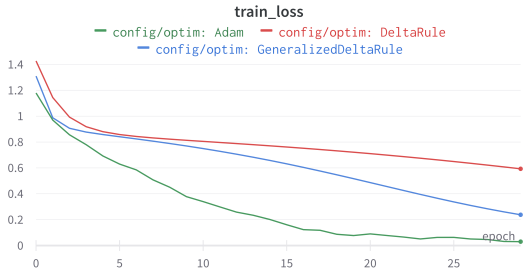

Figure 139: Comparing convergence on training data of various optimizers