ENPM808A – Introduction to Machine Learning

Final Project

Author – Syed Mohammad Ibrahim

UMD ID: iamibi

UID: 118428369

Email: iamibi@umd.edu

Model Selection Process

Problem Statement

From the given problem statement, we can imply that the data we should be predicting – Cmd_vel_v, Cmd_vel_w – is continuous in nature. Therefore, I chose the following two models:

- Linear Regression
- Neural Network

Pre-Processing

- From the provided data of 1094 columns, first 1080 columns consist of LIDAR data which must be reduced in number for ease of work. So, I reduced the 1080 columns of LIDAR data into 10 feature columns, where each of these new columns represented 108 columns from the original data. That is, each column contains mean of the values of every 108 columns from the original LIDAR data.
- Similarly, it was important to process rest of the 12 columns (1094 (total number of columns) - 1080 (LIDAR data columns)-2 (output columns)). Where I summed up the following the following columns together.
    - Final_goal_qk, Final_goal_qr
    - Local_goal_qk, Local_goal_qr
    - Robot_pos_qk, Robot_pos_qr

  To generate 3 columns. So, output of my preprocessing step was as follows:
    - 10 columns (generated from 1080 LIDAR data columns)
    - 3 columns (generated from merging 6 columns of the column mentioned above)
    - 2 output columns (Cmd_vel_v, Cmd_vel_w)
    - 6 columns (remaining 6 columns (Final_goal_x, Final_goal_y, Local_goal_x, Local_goal_y, Robot_pos_x, Robot_pos_y))
- The training data(N) was separated into final training dataset(N-K) and the validation dataset(K) in the ratio of 8:2
- To generate the models, a final dataset was used

- These models were evaluated on the validation data and then the validation and learning curves were generated
- Tune the hyperparameters based on the values of E_{in} and E_{val}
- The model with least E_{val} is selected
- The selected model is then trained on the original processed data (N)
- Then the model is tested on the processed data
- Finally, $E_{out}(E_{test})$ is computed

Rationale

The model was chosen based on the E_{in} values, which for respective models are:

Linear Regression:

E_{in}: 0.0764

E_{val}: 0.0786

E_{out}: 0.08907

Neural Network:

E_{in}: 0.04027

E_{val}: 0.0483

E_{out}: 0.08084

Validations Usage

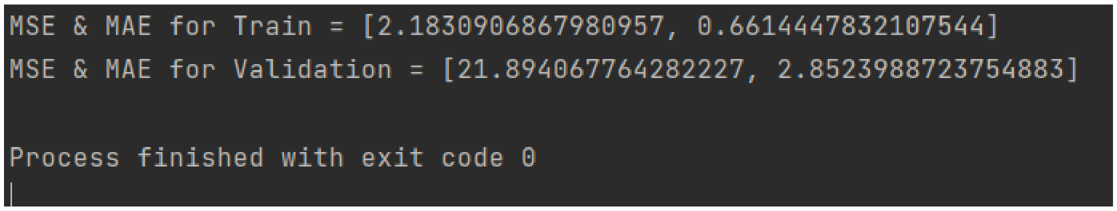
Validation was used for selecting best combination of hyperparameters like number of layers, number of nodes, activation function, optimization function, regularization parameters for a given model in such a way that E_{val} is minimum. Furthermore, validation was also used to select the best model that produced least E_{val} value.

Hyperparameters and Final Model Selection

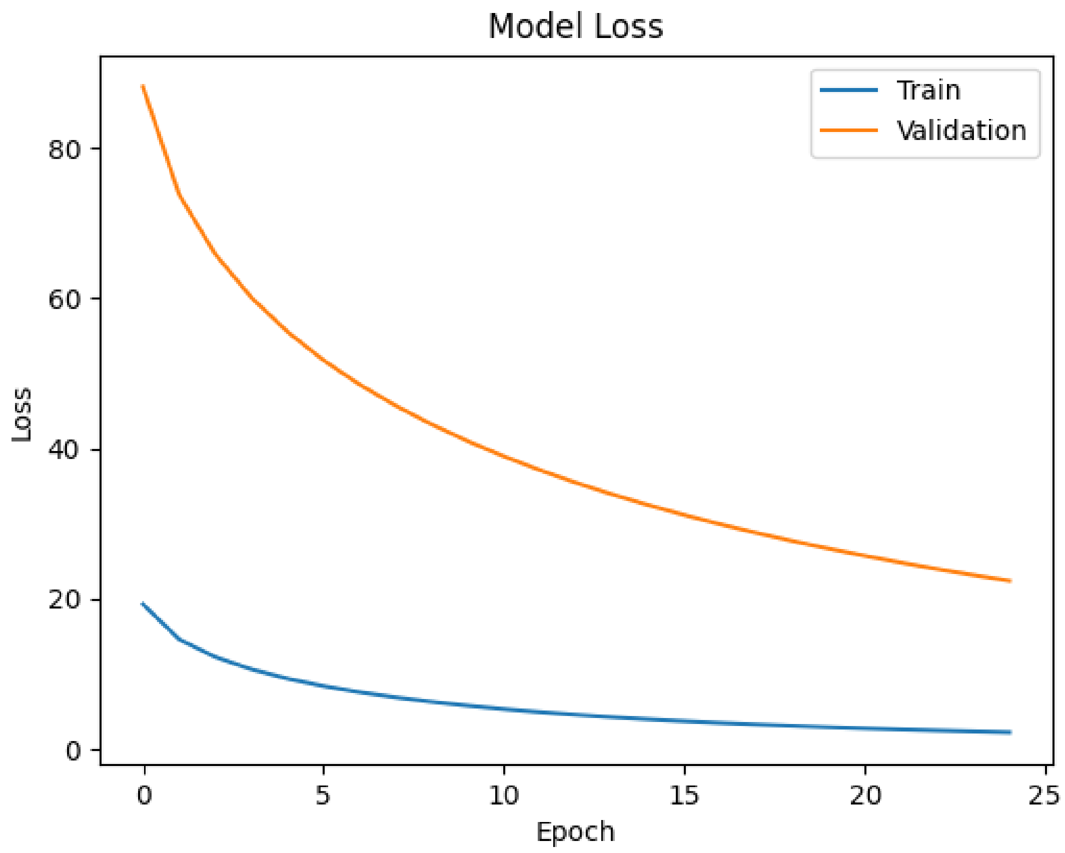
I carried out this process of hyperparameter selection by varying one hyperparameter at a time and keeping rest of the hyperparameter fixed. Further, I calculated E_{val} and E_{in} values after each update and plotted validation and learning curves for each the same.

Creating a simple neural network

Model creation started with a basic neural network with input layer, 2 hidden layer and the output layer. Further, I also used hyperparameters like activation functions (i.e., “relu” and “linear”), optimizer (i.e., “Adagrad”), number of nodes in each layer, loss function etc.



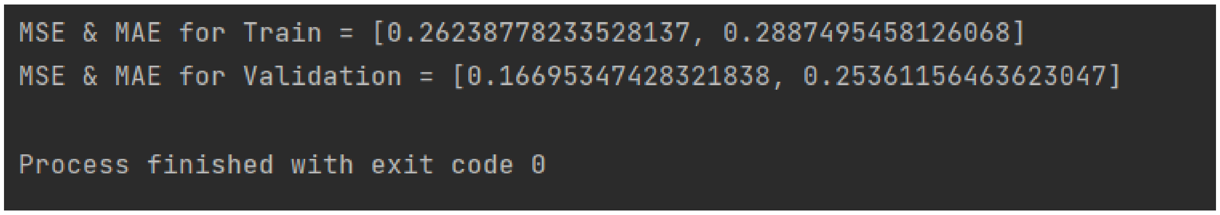
```
MSE & MAE for Train = [2.1830906867980957, 0.6614447832107544]
MSE & MAE for Validation = [21.894067764282227, 2.8523988723754883]

Process finished with exit code 0
```



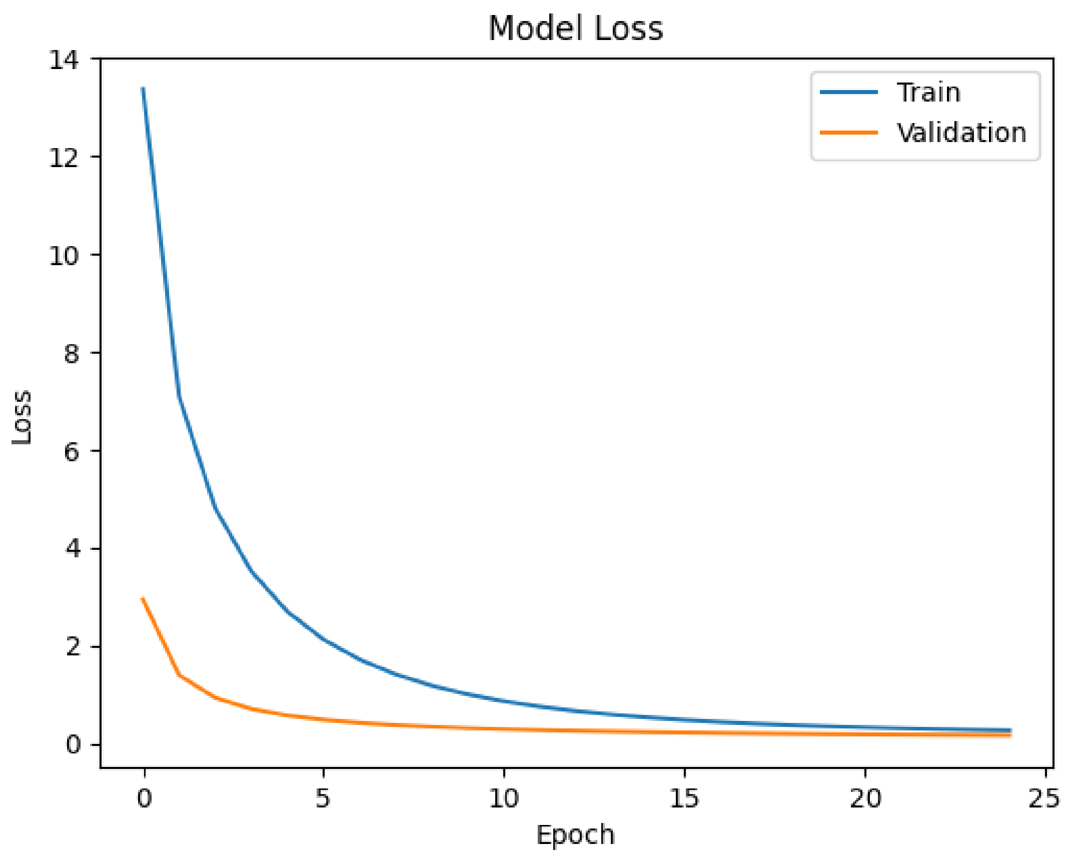
Thus, the model was trained on the training data and was further evaluated on the training data and validation data to produce the Mean Squared Error (MSE) loss of 2.183 and 21.894 respectively.

Hidden Layers Count

Further, I increased the number of hidden layers while keeping all other hyperparameters fixed. This setting resulted in the decrease in the training and validation MSE loss to 0.262 and 0.166.

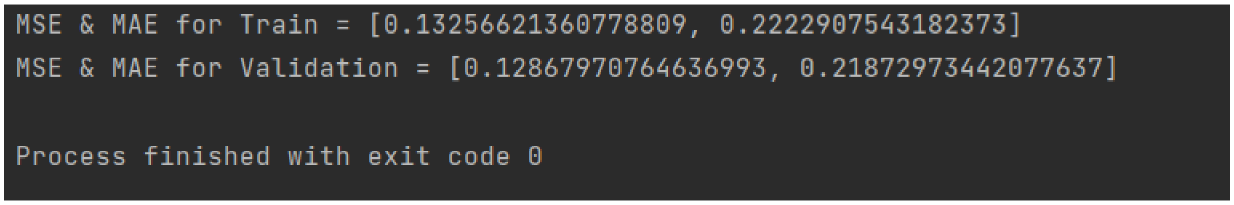


```
MSE & MAE for Train = [0.26238778233528137, 0.2887495458126068]
MSE & MAE for Validation = [0.16695347428321838, 0.25361156463623047]

Process finished with exit code 0
```

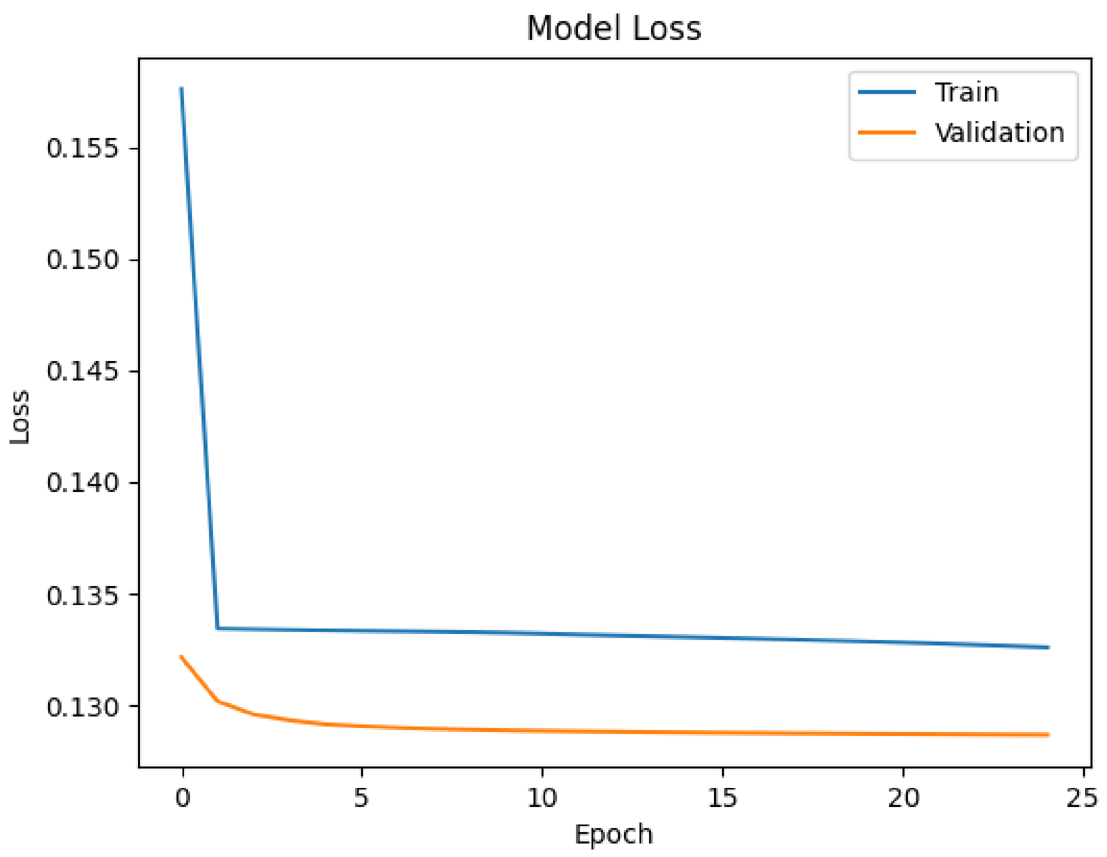


Number of Nodes

In this step, I changed the hyperparameter - **number of nodes** for each hidden layer from **10** to **100** while keeping all other hyperparameter same as before. This new setting further decreased the MSE loss for training and validation to 0.1325 and 0.1286 respectively.

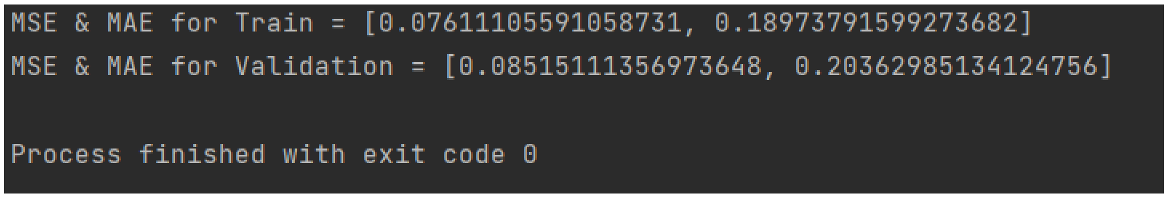


```
MSE & MAE for Train = [0.13256621360778809, 0.2222907543182373]
MSE & MAE for Validation = [0.12867970764636993, 0.21872973442077637]

Process finished with exit code 0
```

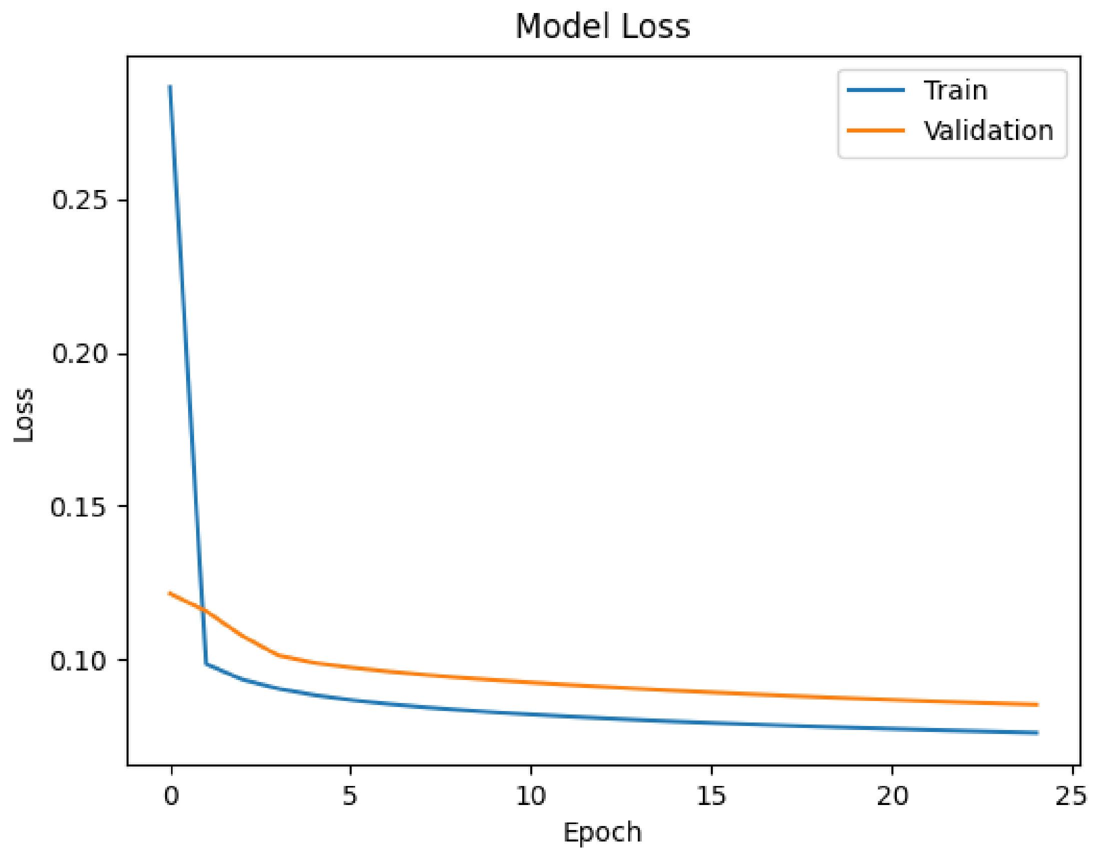


Hyperparameter Change – Activation – relu

In this step, I updated hyperparameter- “activation” (to “relu”) for each layer while keeping all other hyperparameter fixed. As a result, there was an increase in the MSE loss for training and validation (0.0761 and 0.0851).

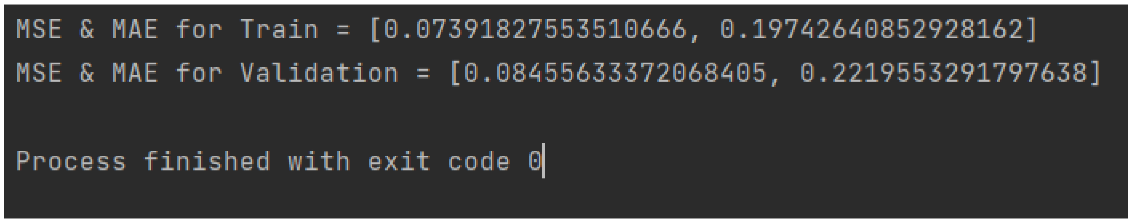


```
MSE & MAE for Train = [0.07611105591058731, 0.18973791599273682]
MSE & MAE for Validation = [0.08515111356973648, 0.20362985134124756]

Process finished with exit code 0
```

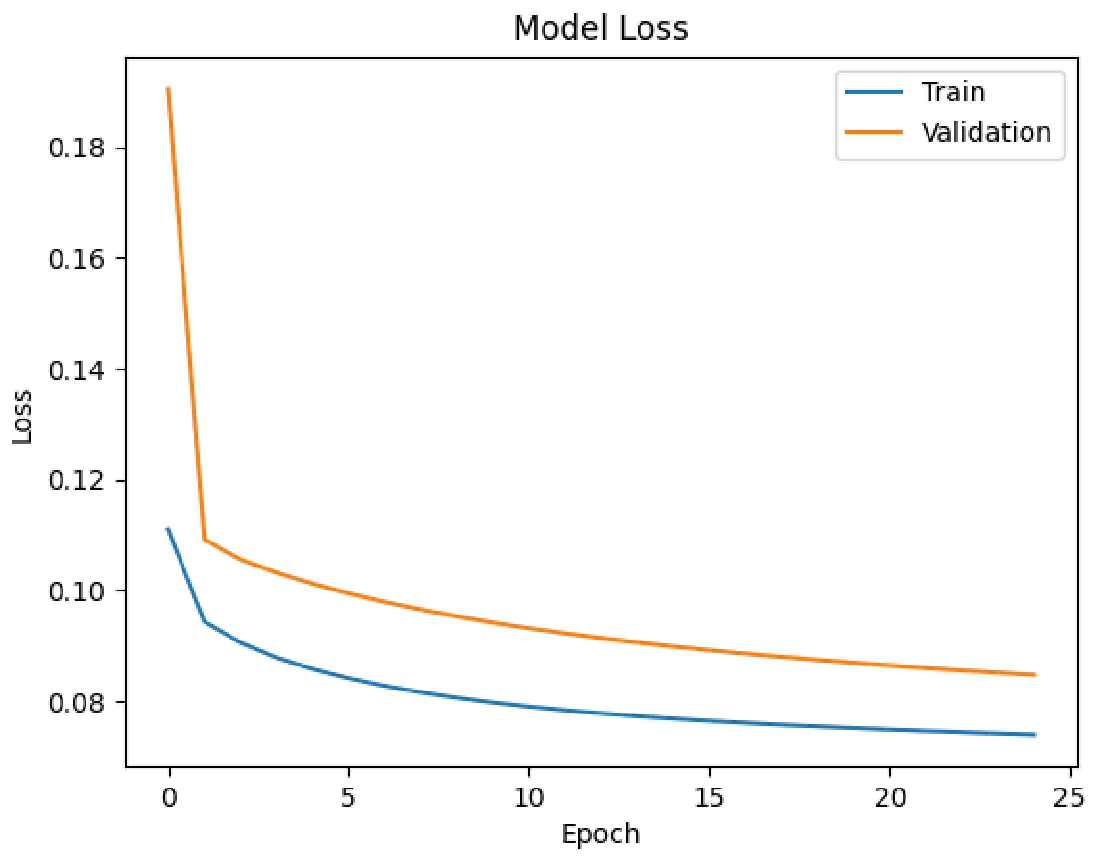


Activation parameter in random order

Further, I increased the number of hidden layers and changed hyperparameter – "activation" (to relu and linear) for different layers as shown in the below screenshot. This new setting reduced the training and validation loss to 0.0739 and 0.0845 respectively. (Note: no other hyperparameters were updated other than the two mentioned in this step)

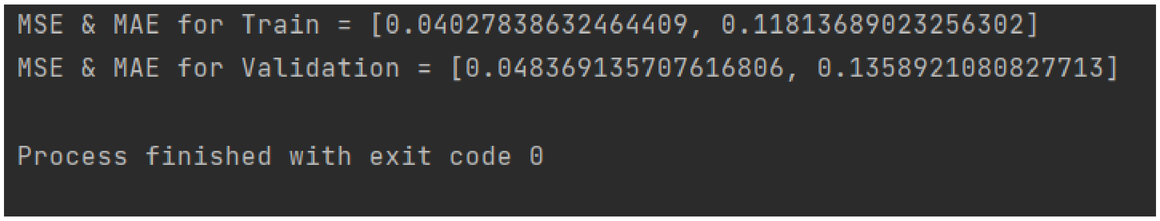


```
MSE & MAE for Train = [0.07391827553510666, 0.19742640852928162]
MSE & MAE for Validation = [0.08455633372068405, 0.2219553291797638]

Process finished with exit code 0
```

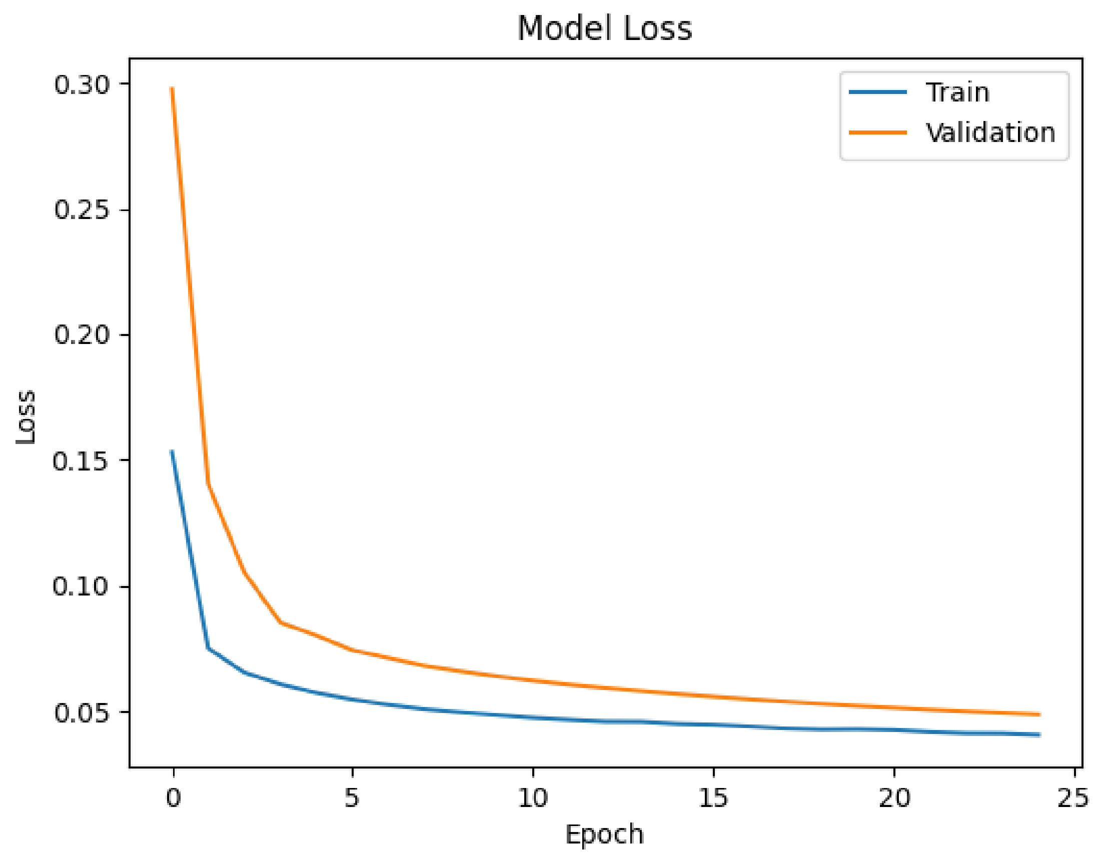


Hyperparameter Change – optimizer and learning rate

In this step, I updated optimizer to Adam and assigned the hyperparameter – "learning factor" (lr) to 0.001 (Note: no other hyperparameters were changed in this step). This new setting in the model reduced the loss values for training and validation to 0.04027 and 0.04836 respectively.

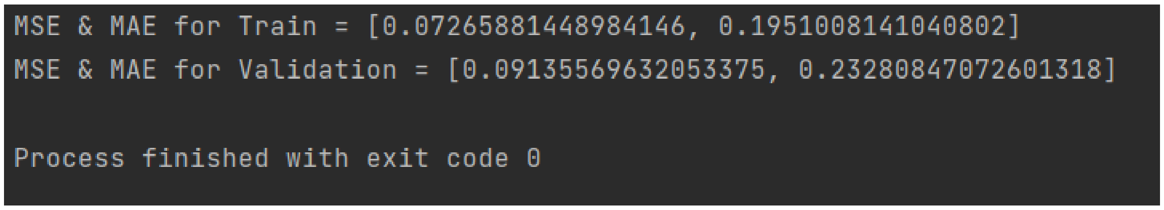


```
MSE & MAE for Train = [0.04027838632464409, 0.11813689023256302]
MSE & MAE for Validation = [0.048369135707616806, 0.1358921080827713]

Process finished with exit code 0
```

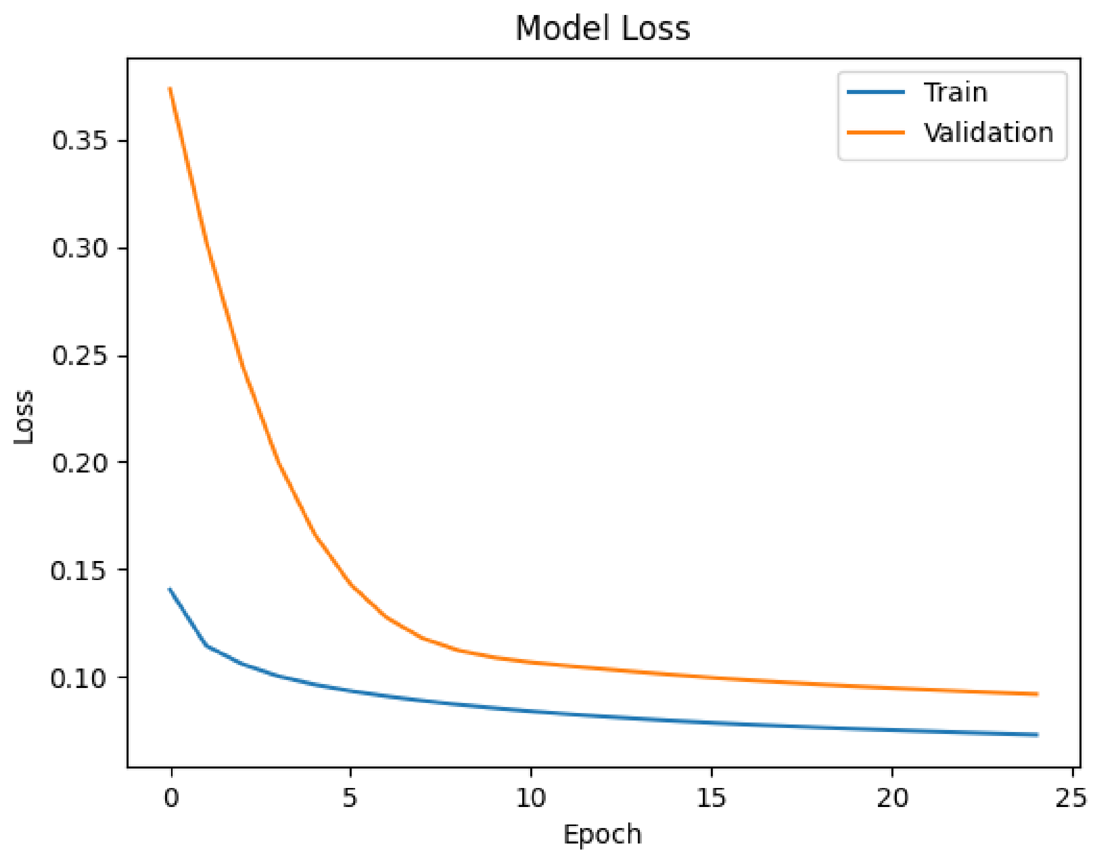


Hyperparameter Change – Linear Rate – 0.00001

In this step, I further reduced the value of learning rate from 0.001 to 0.00001 (Note: no other hyperparameters were changed in this step). This setting resulted in an increase in the loss values for training and validation (to 0.080 and 0.0756 respectively).



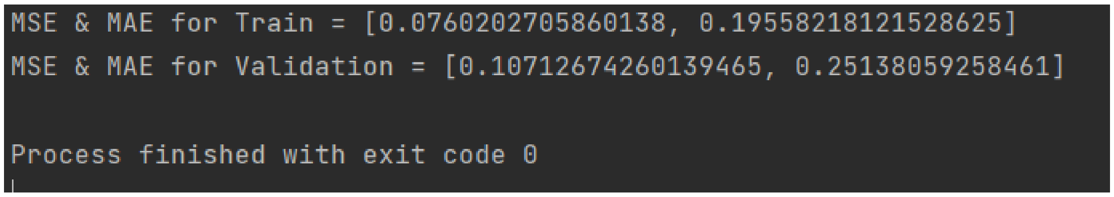
```
MSE & MAE for Train = [0.07265881448984146, 0.1951008141040802]
MSE & MAE for Validation = [0.09135569632053375, 0.23280847072601318]

Process finished with exit code 0
```



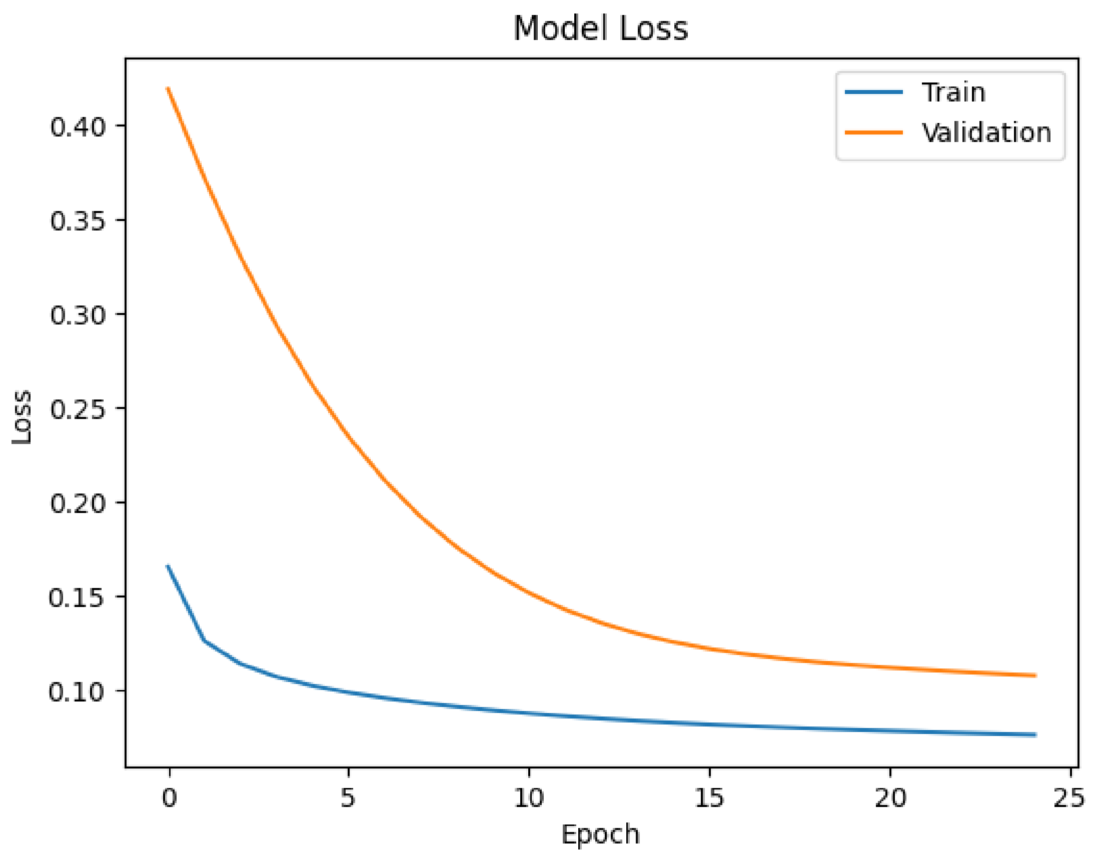
To avoid the increase in the loss values, I rollbacked the learning rate values to previous step.

Regularization

In this step, I imported a class l1 from keras.regularizer library to implement regularization on the model. The regularization was implemented on the 2nd and 3rd hidden layers of the network. This setting resulted in the increase in the loss values for training and validation (to 0.07602 and 0.1071 respectively).



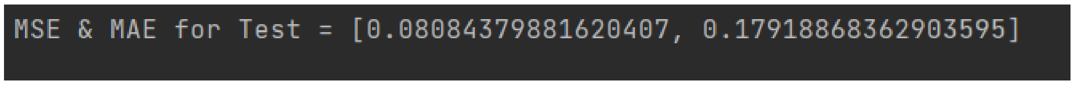
```
MSE & MAE for Train = [0.0760202705860138, 0.19558218121528625]
MSE & MAE for Validation = [0.10712674260139465, 0.25138059258461]

Process finished with exit code 0
```



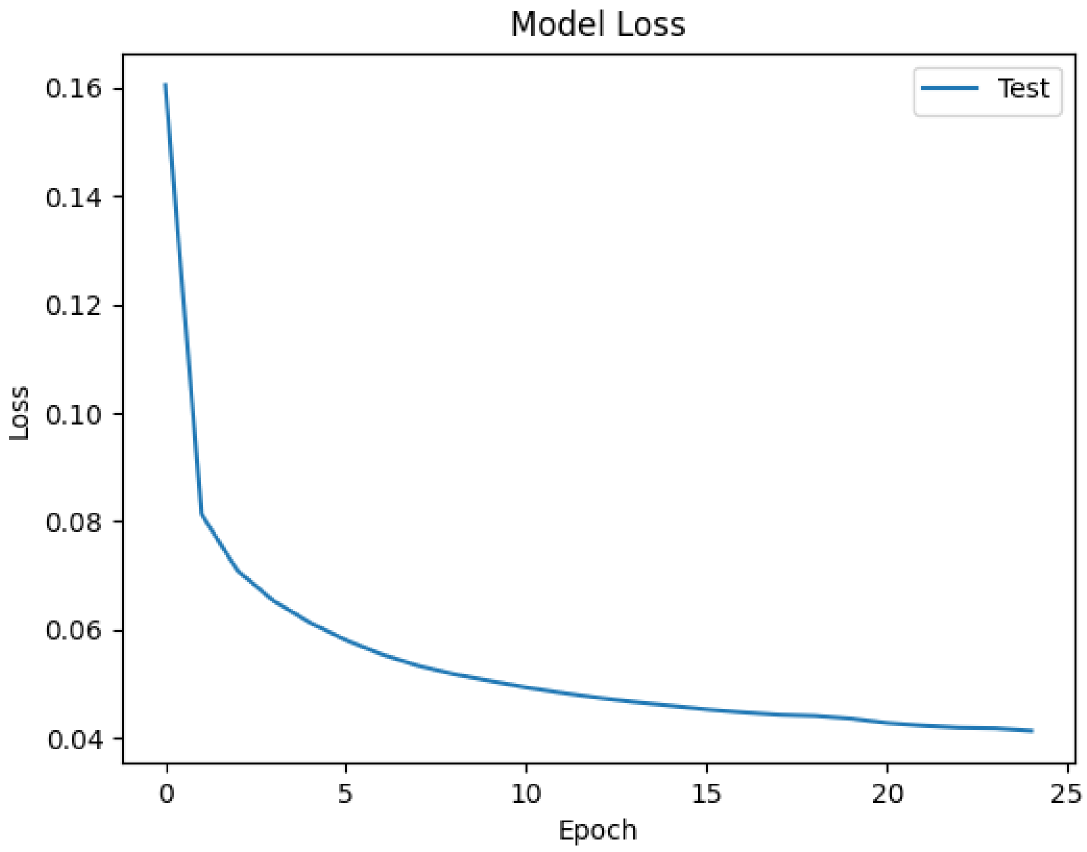
I reverted the regularization changes to the previous values to avoid the increase.

Final Model Selection

Since the hyperparameter and regularization setting in the model in the step "Hyperparameter Change – optimizer and learning rate" has the least MSE loss value for training and validation. Hence, I selected this as my final model.



```
MSE & MAE for Test = [0.08084379881620407, 0.17918868362903595]
```


Model Loss
Test
Loss
Epoch
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0
5
10
15
20
25