CS 441 - Final Project

You only need to complete and submit this once for the group. Be sure to add the other group members to the submission in Gradescope. List them here also.

Group Member Names	**NetIDs**
Wei-Chuan, Tsai	wctsai3
Chu-Haoh, Hsiao	chuhaoh2

Complete the sections below. You do not need to fill out the checklist.

Total Points Available **[ ] / 150**

1. Problem description [ ] / 20
2. Model comparison
    a. Which models [ ] / 5
    b. Hyperparameter experiments [ ] / 35
    c. Best model/parameters result [ ] / 10
3. Analysis: Training Size or Features [ ] / 30
4. Stretch Goal: Innovation
    a. Approach description, experiments, innovation [ ] / 25
    b. Publishable with justification [ ] / 25
5. Attribution / Group Contributions [ ] -5 if incomplete (or page not selected)

1. Problem Description

Give an overview of the problem you are trying to solve.

House price regression:
https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques
Use given data to predict the house price.

Explain what you are trying to predict, and what inputs (features) are available for prediction.

The train.csv have 1460 rows and 81 columns.
The first column Id that should be ignored when training.
There are 19 columns contain Nan data that need to be handled.
There are 34 columns that type is numerical
There are 43 columns(include some Nan columns) that type is not numerical that should use encoding to let it able to train.

Explain the experimental setup. How many examples for train, val, test? What are the metrics?

First of all, this kaggle competition doesn't have the ground truth of test data, so need to split the validation data from training data. The percentage of training and validation are 80% and 20 %. The data will be shuffled before splitting. Then should convert dataframe to pytorch tensor for training. Before training, the data will be normalized and will be converted back when evaluating. After the evaluation through validation data, use full train data set without split 80/20 to predict the test dataset.

What are some of the challenges in solving this problem?

The data preprocessing is tricky as there are lots of Nan data in raw data and it is difficult to choose which method to fill that missing data.

2. Model Comparison

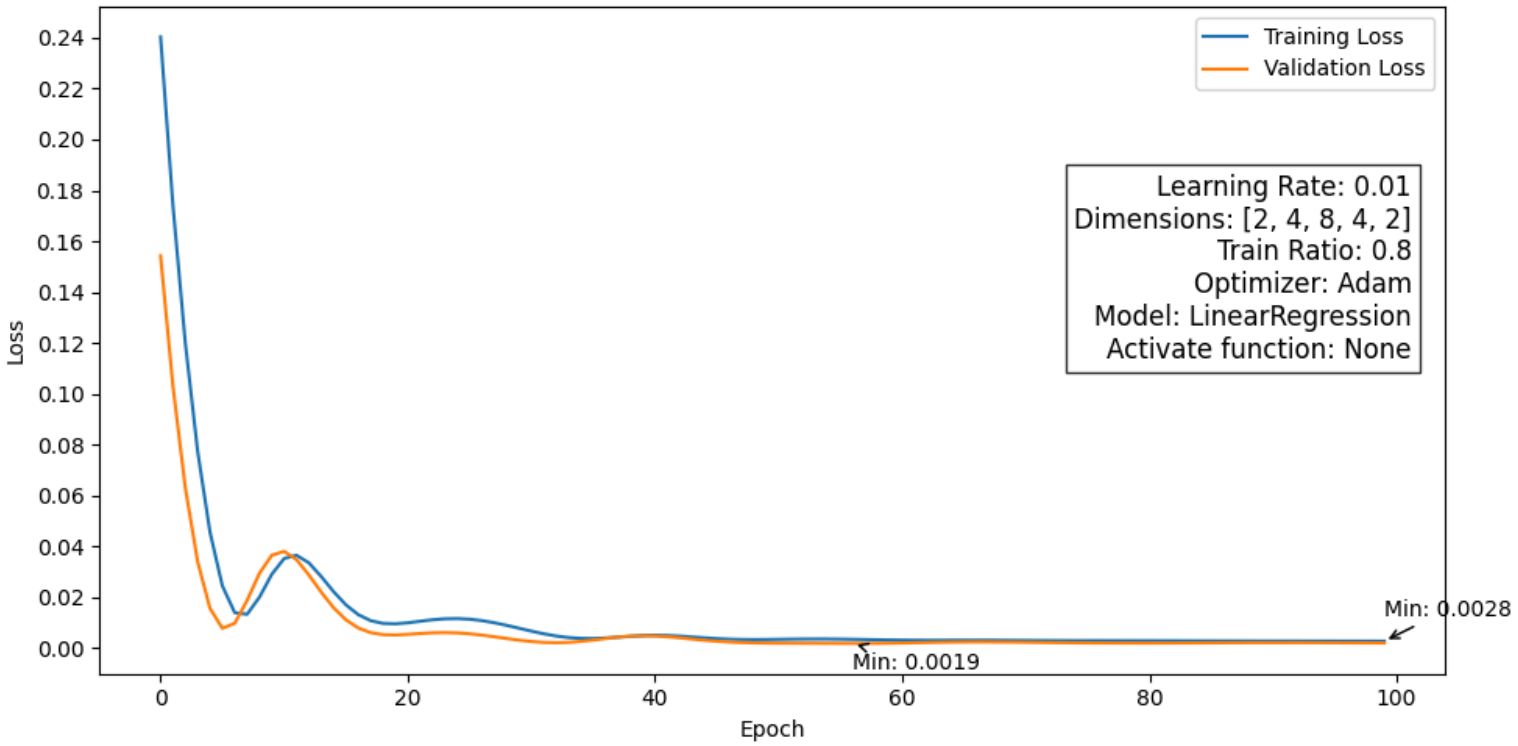
Which three models will you compare? Which hyperparameter(s) will you test?

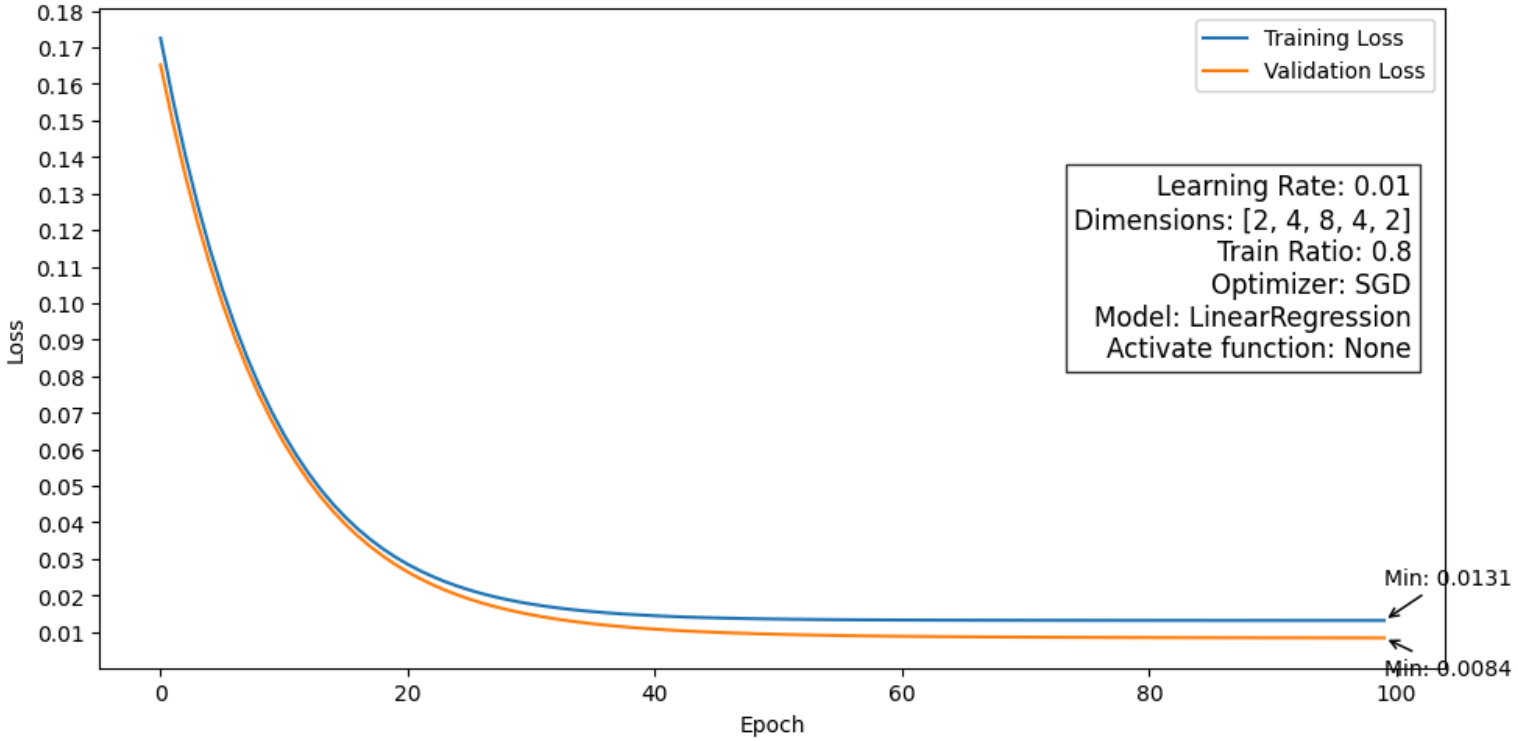
Pure linear regression model without activation function. Depth Dimensions Activation function Learning rate Train split Epochs Optimizer
CNN multi-layers Conv1d with activation function Depth Dimensions Activations function Learning rate Train split Epochs Optimizer
Graph neural network for linear regression Nodes Edges Learning rate Train split Epochs Optimizer

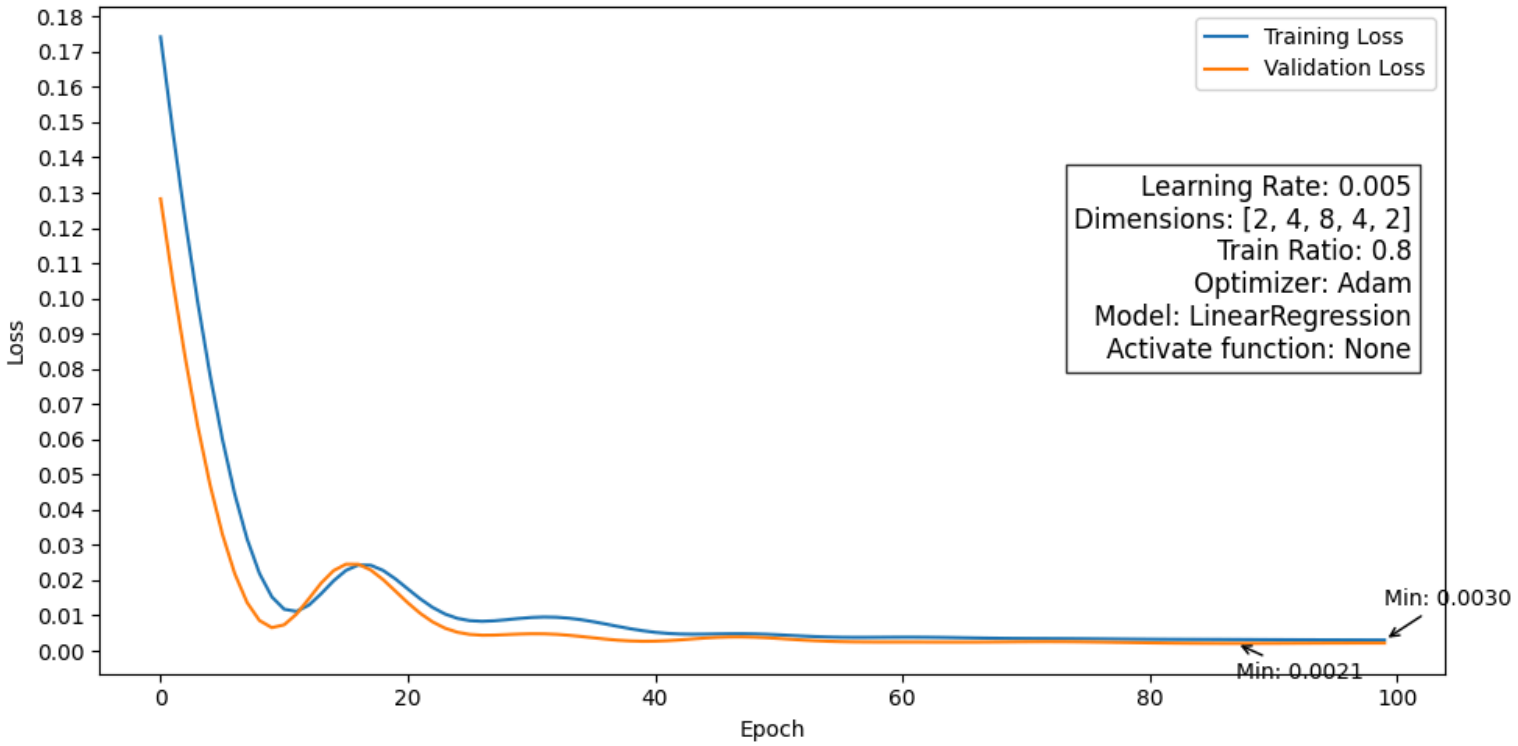
Use tables or plots to show the evaluated hyperparameter values for each model, and indicate which is best. These experiments should use only the training and validation sets. Feel free to delete the boxes, as long as it's clear.

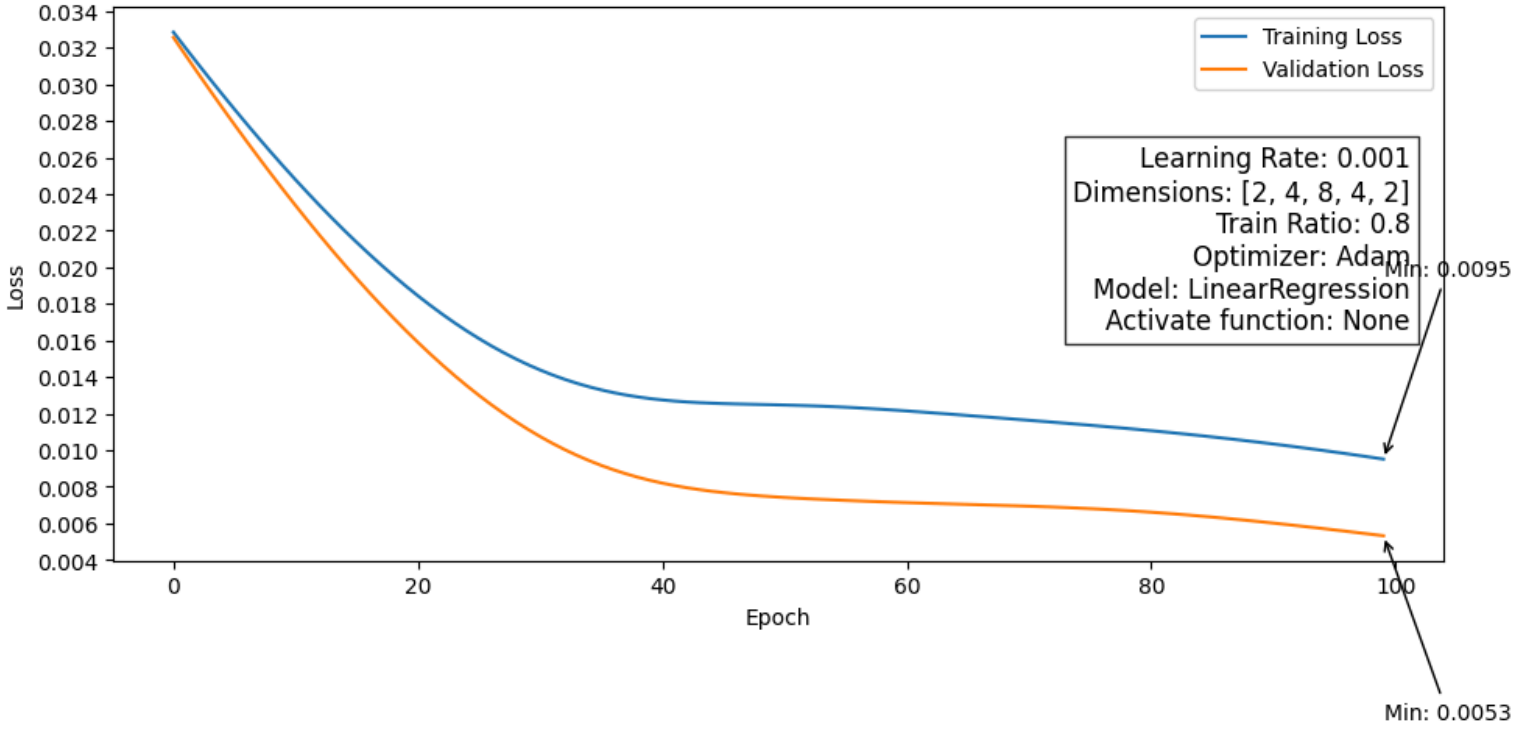
Here will only use 100 epochs for easily identify difference

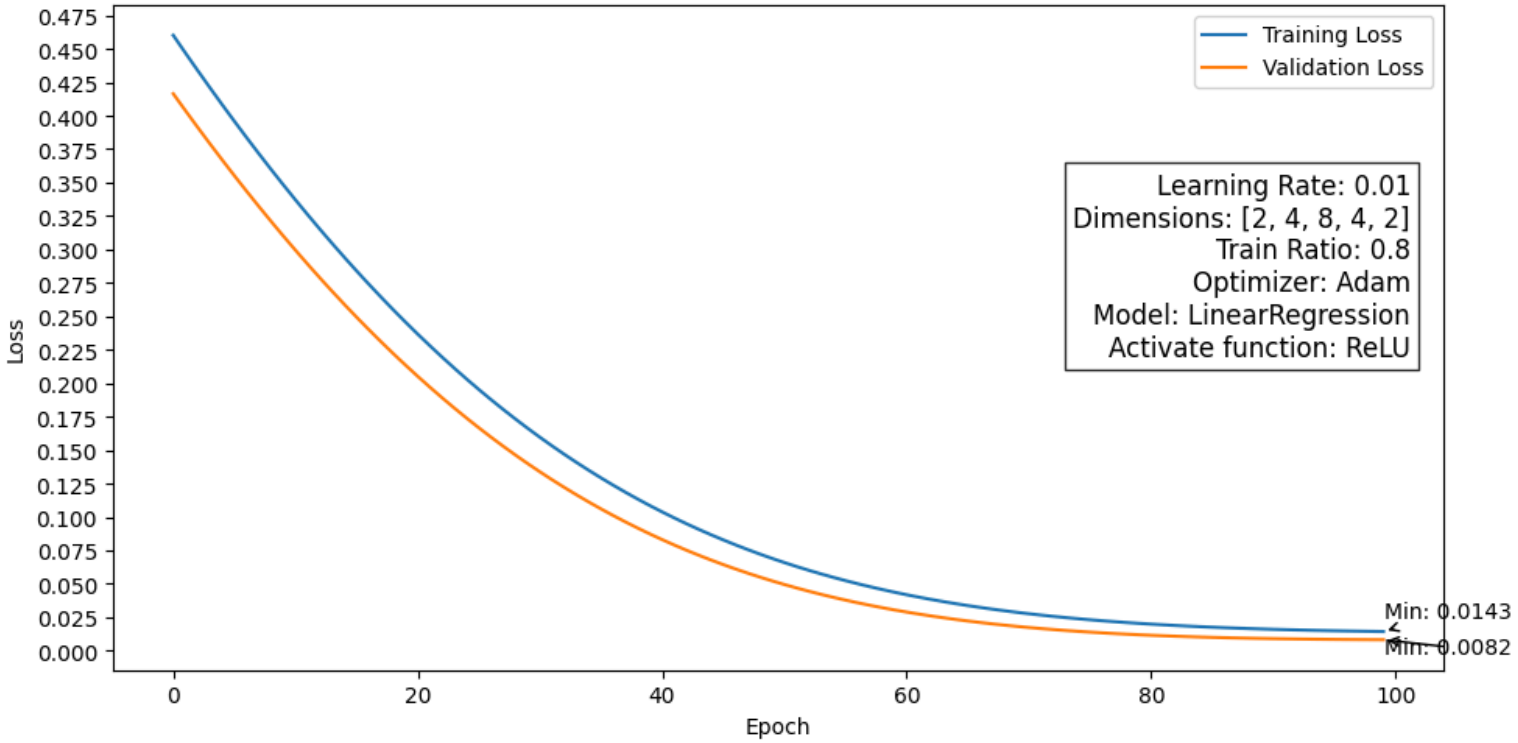
Linear Regression

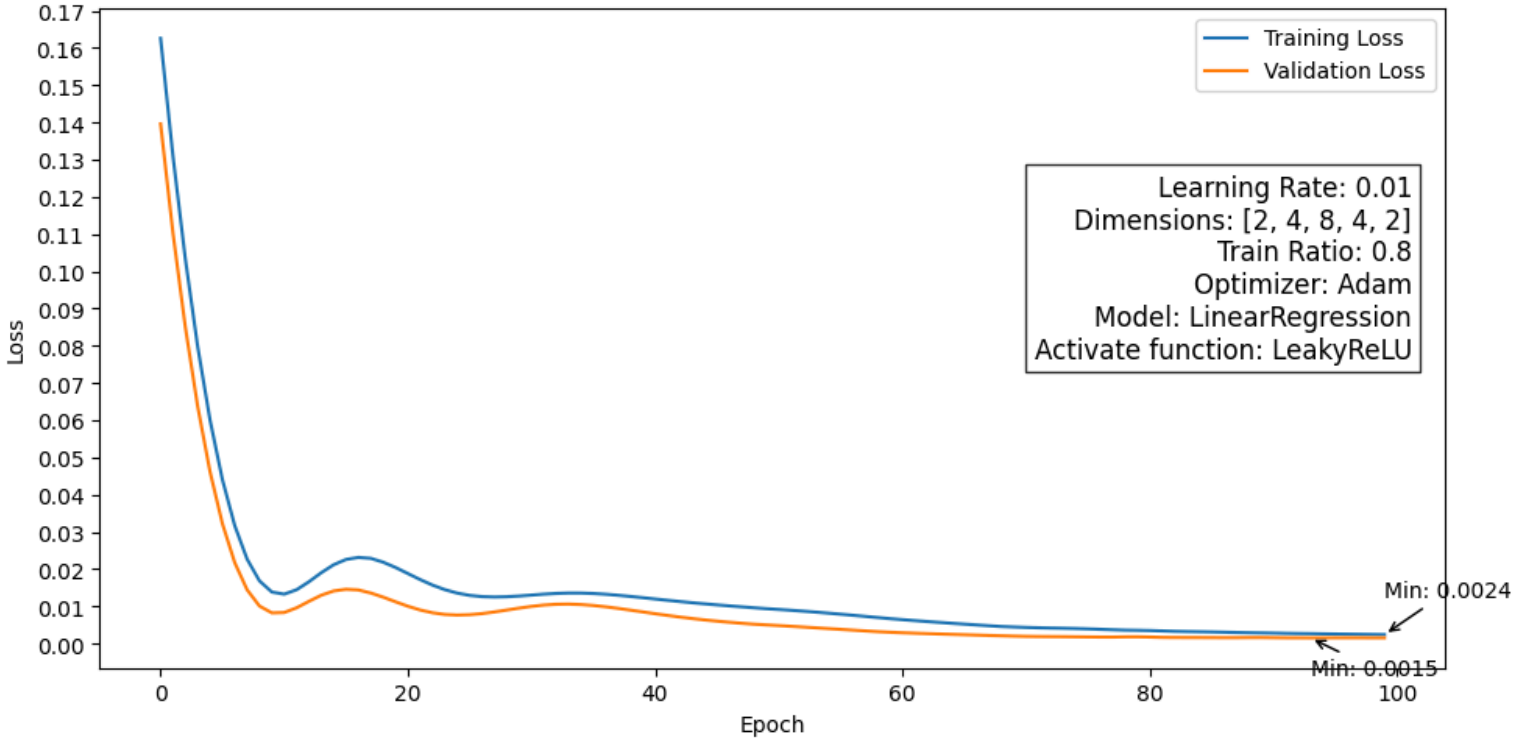

Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: None
Min: 0.0028
Min: 0.0019
Loss
Epoch

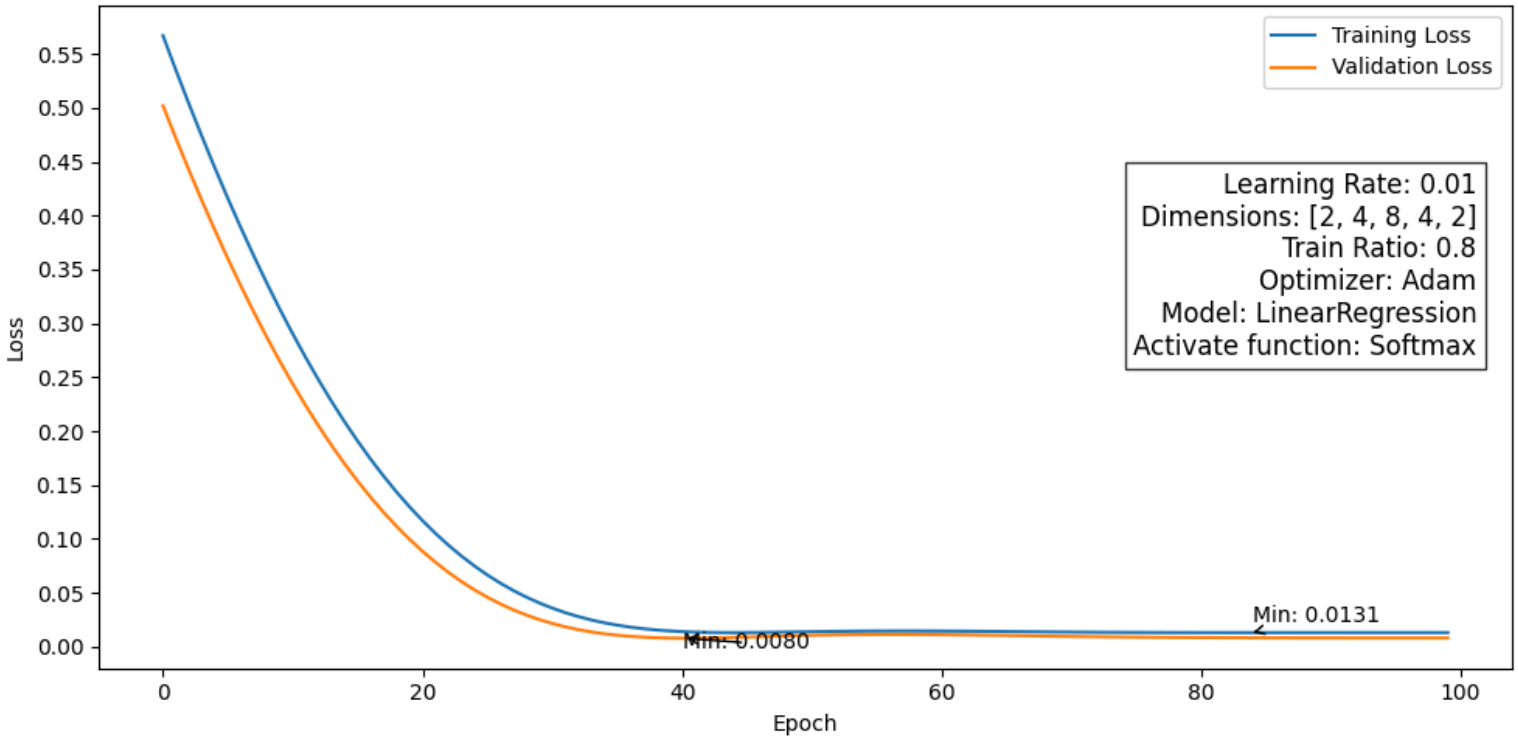

Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: SGD
Model: LinearRegression
Activate function: None
Min: 0.0131
Min: 0.0084
Loss
Epoch

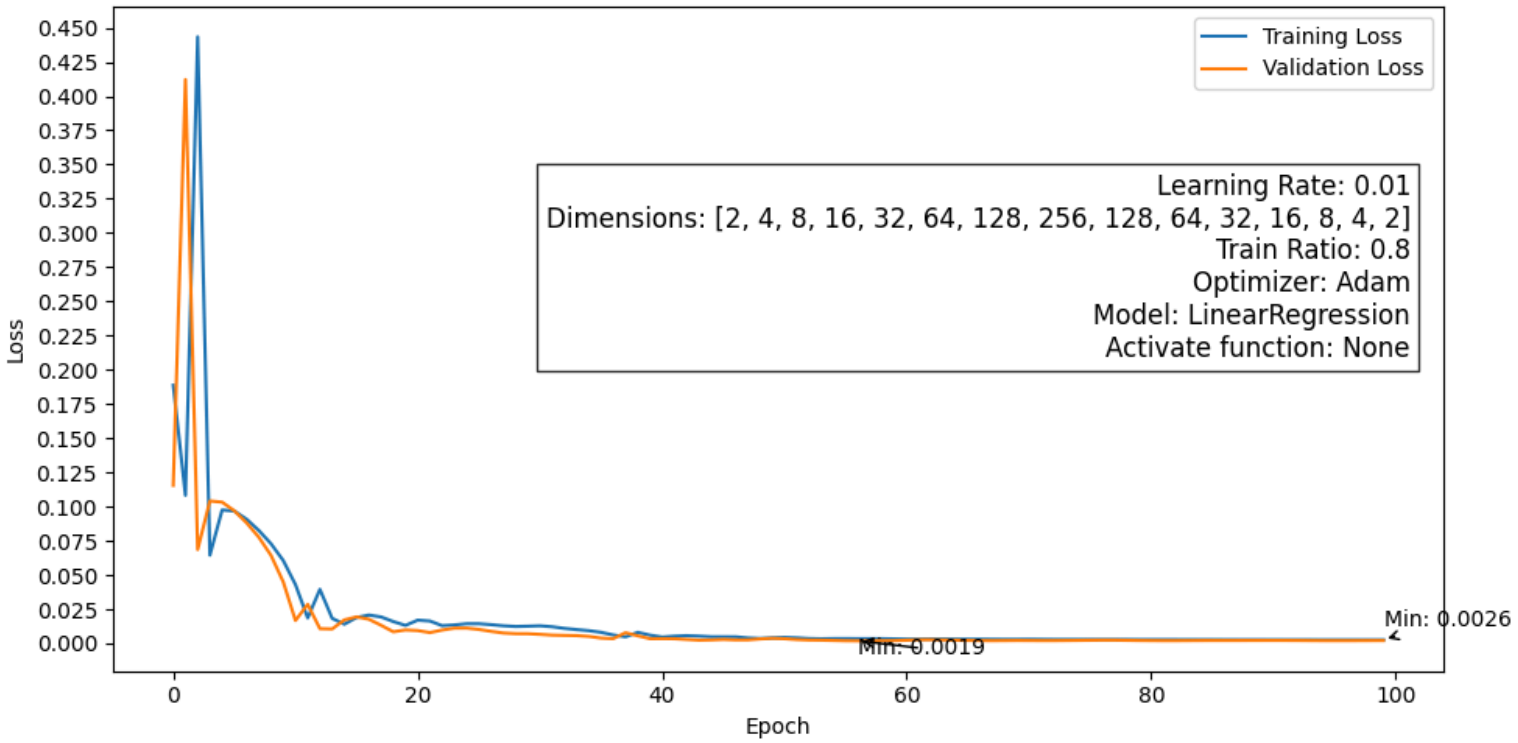

Training Loss
Validation Loss
Learning Rate: 0.005
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: None
Min: 0.0030
Min: 0.0021
Loss
Epoch

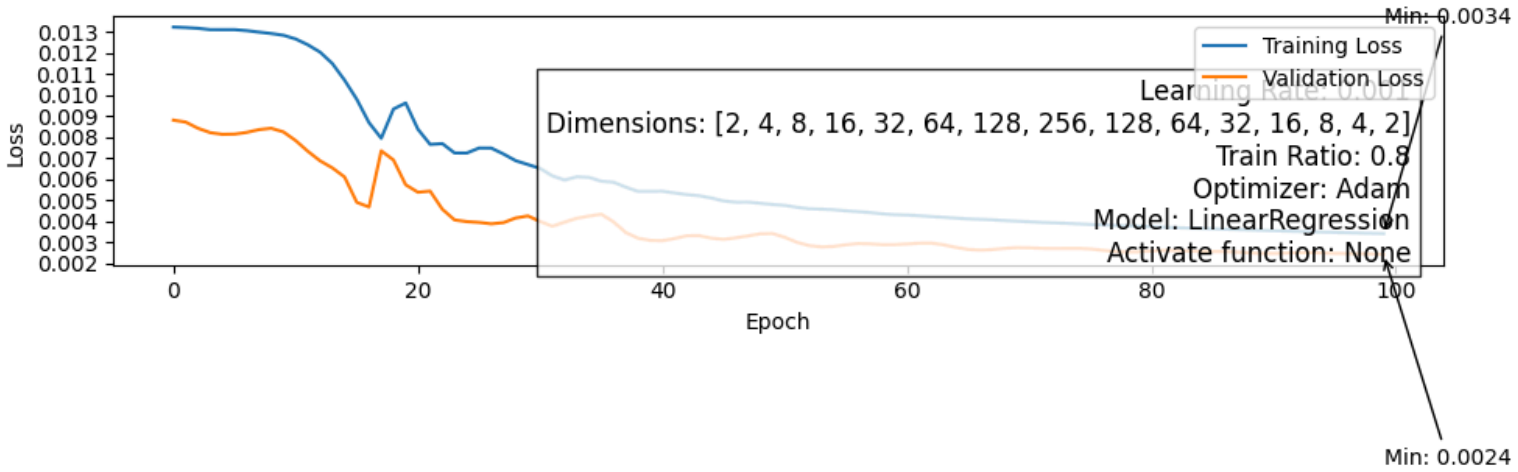

Training Loss
Validation Loss
Learning Rate: 0.001
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: None
Min: 0.0095
Min: 0.0053
Loss
Epoch


Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: ReLU
Min: 0.0143
Min: 0.0082
Loss
Epoch


Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: LeakyReLU
Min: 0.0024
Min: 0.0015
Loss
Epoch

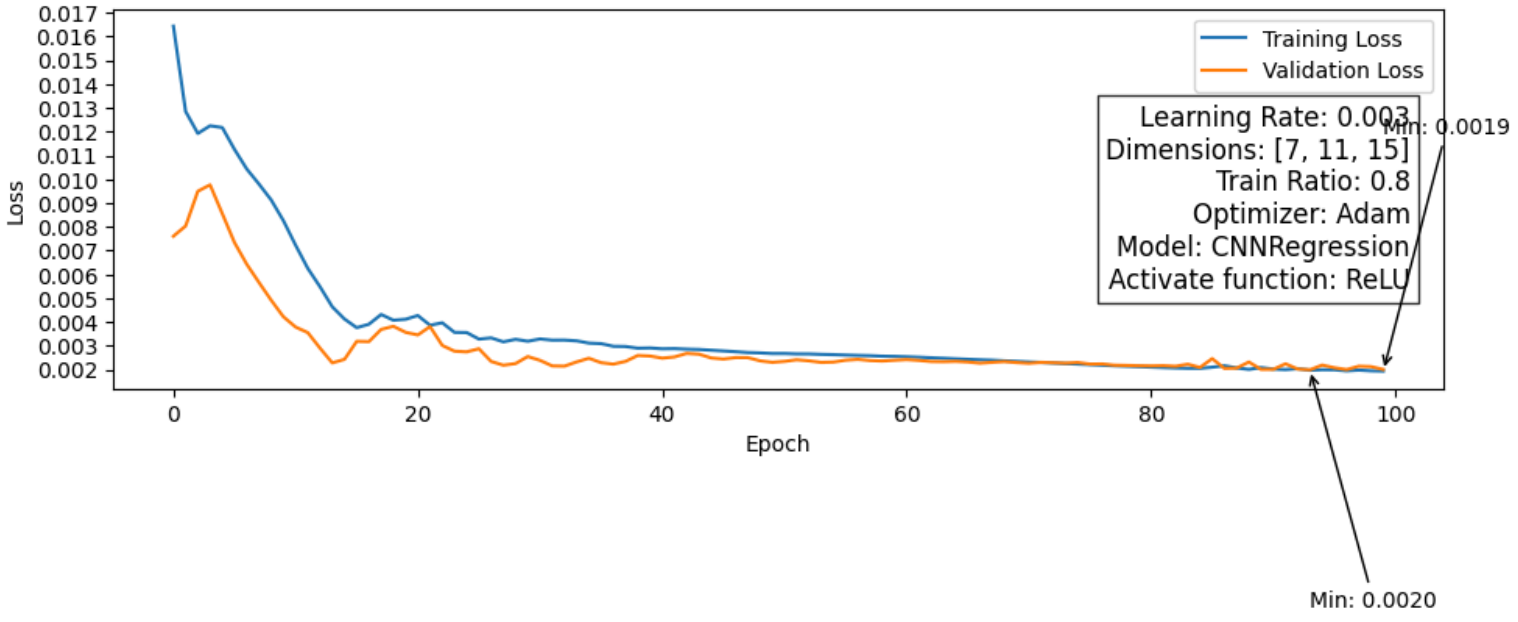

Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: Softmax
Min: 0.0131
Min: 0.0080
Loss
Epoch

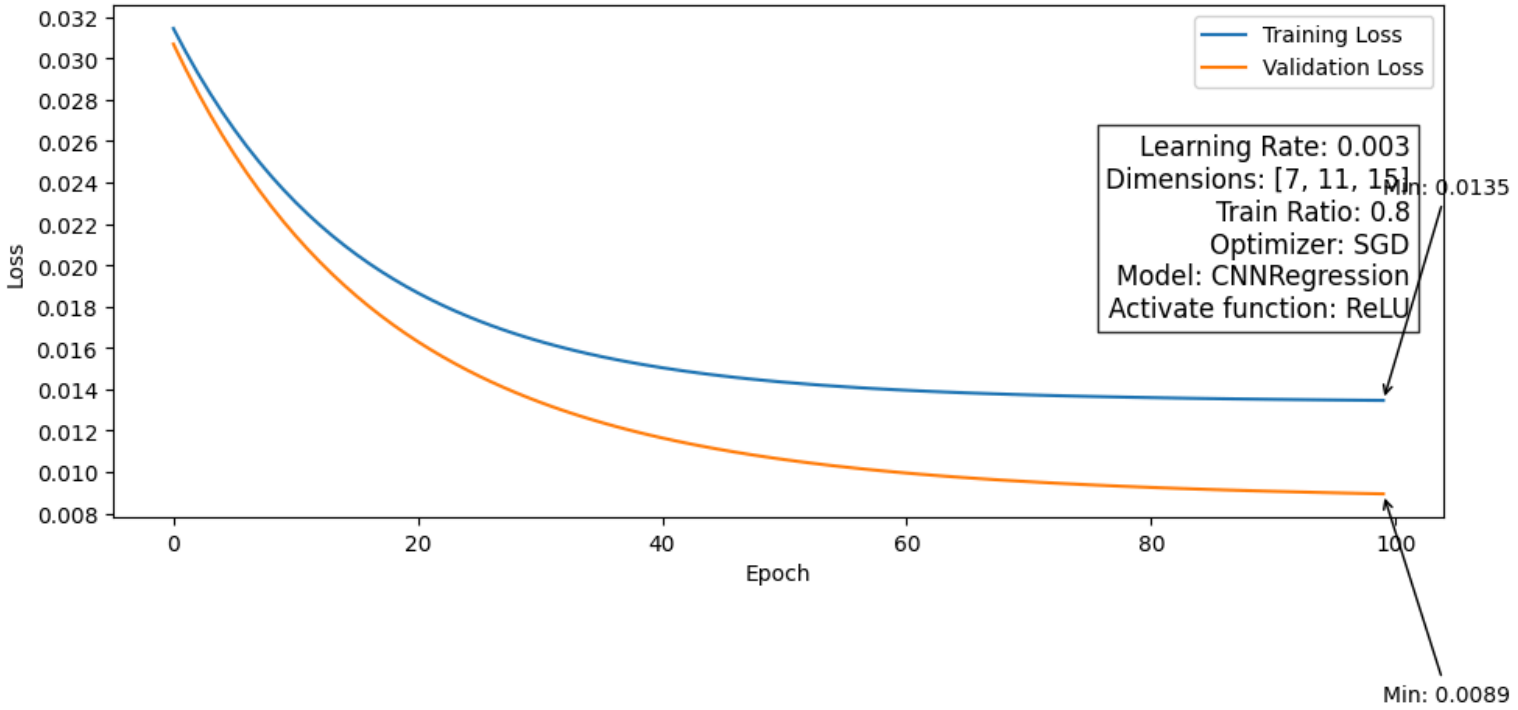

Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 16, 32, 64, 128, 256, 128, 64, 32, 16, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: None
Min: 0.0026
Min: 0.0019
Loss
Epoch

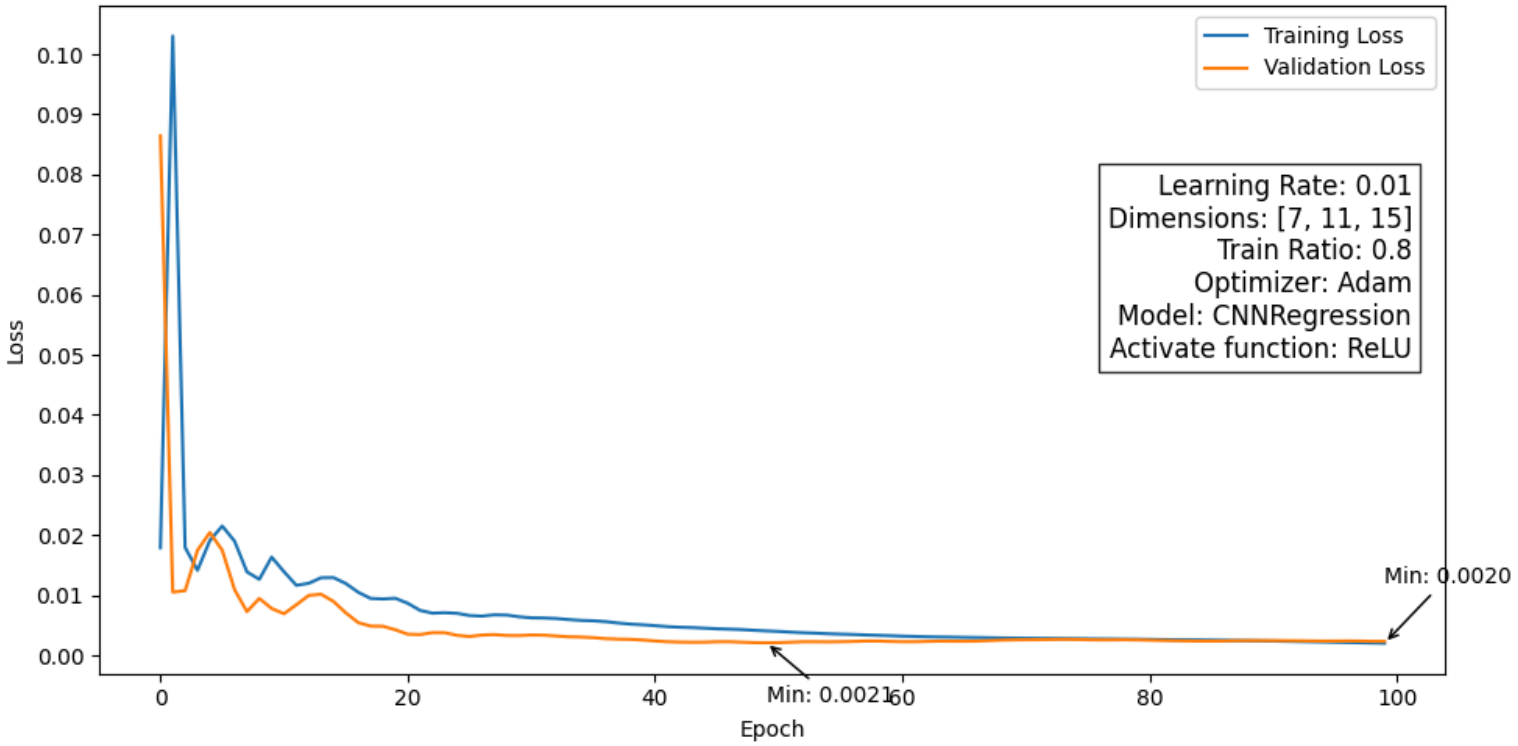


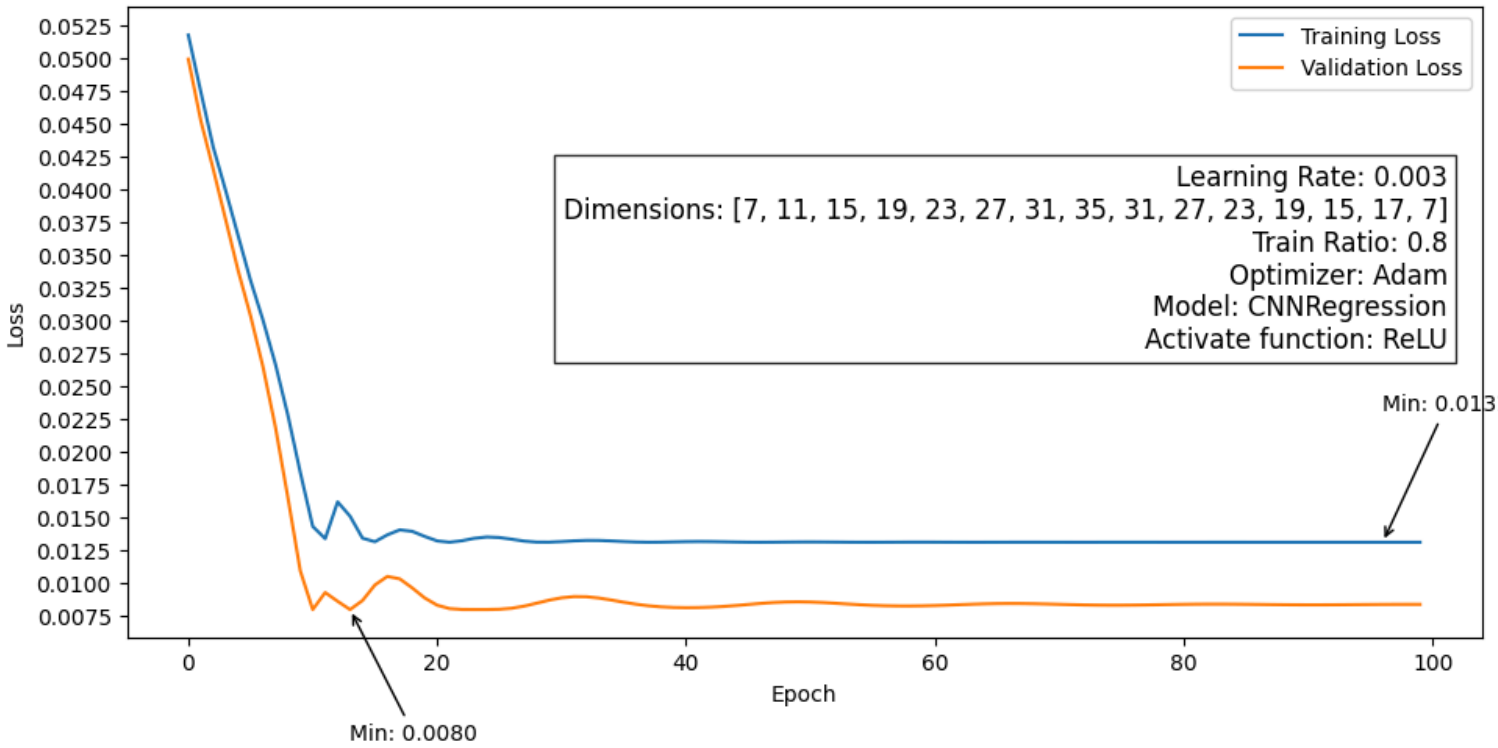
Depth/Dimensions : This task is not too complex, so if you increase the depth and dimensions as deeper as possible, it will not actually increase the accuracy of results but will let the model difficult to train at the start. However, if the depth and dimensions are not enough, the results will be too bad and can't fit the training data. So the only way that we can know the suitable dimensions is through experiments.
Activation function : As I mentioned above, in this case it is not necessary to let the model be complex, so the activation function is optional and the experiments show adding the activation function didn't improve the results. In contrast, the results of experiments of ReLU, LeakyReLU and Softmax reduce the performance of the model.
Learning rate : Learning rate is based on the experiment's results also and seems it is not necessary to add learning rate decay here.
Train split : Based on the history of machine learning.
Epochs : Use the validation data to know where need to stop early, if the validation error increases and training loss decreases, you may need to stop at that cross point.
Optimizer : Based on the results of experiments, Adam do better in this case.

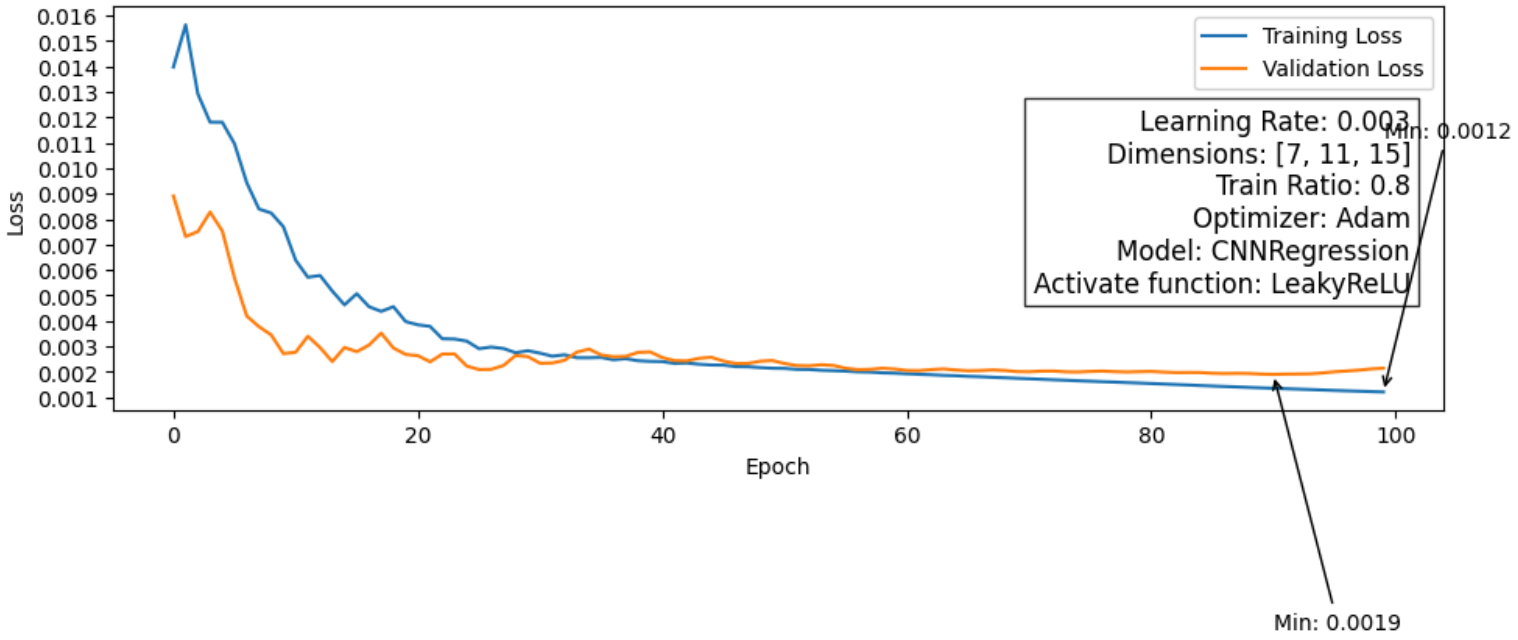
CNN Conv1d Regression

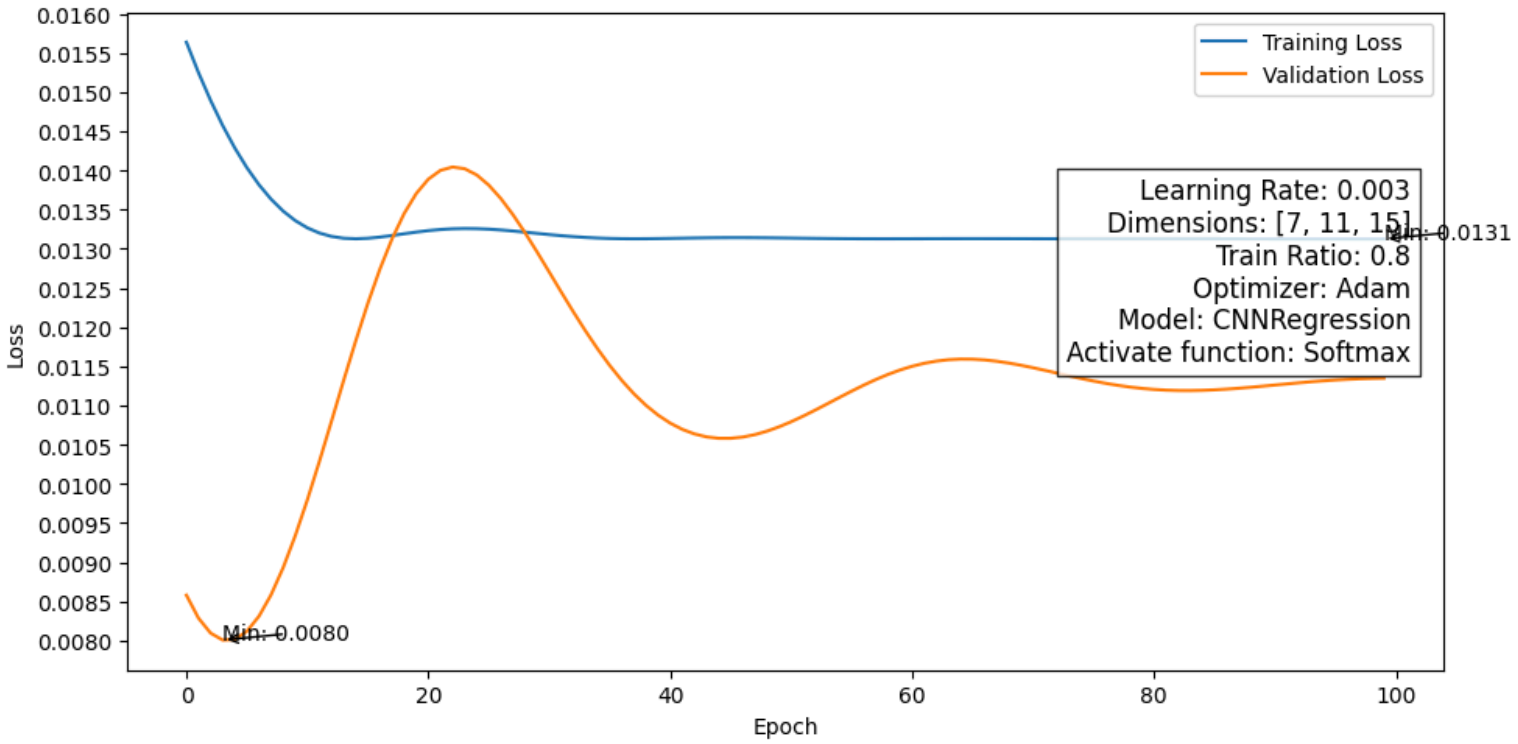



Training Loss
Validation Loss
Learning Rate: 0.003
Dimensions: [7, 11, 15]
Train Ratio: 0.8
Optimizer: SGD
Model: CNNRegression
Activate function: ReLU
Min: 0.0135
Min: 0.0089
Loss
Epoch


Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [7, 11, 15]
Train Ratio: 0.8
Optimizer: Adam
Model: CNNRegression
Activate function: ReLU
Min: 0.0020
Min: 0.0021
Loss
Epoch

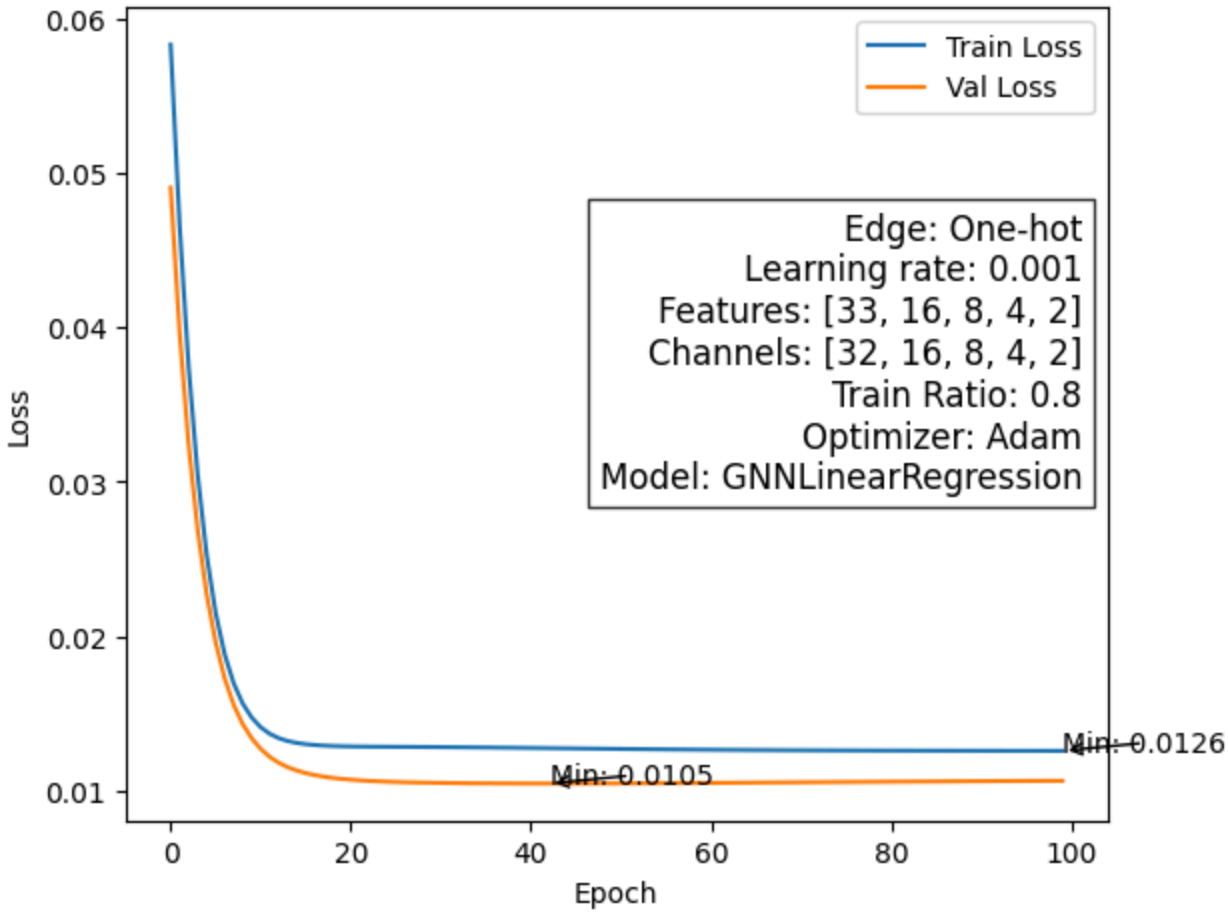

Training Loss
Validation Loss
Learning Rate: 0.003
Dimensions: [7, 11, 15, 19, 23, 27, 31, 35, 31, 27, 23, 19, 15, 17, 7]
Train Ratio: 0.8
Optimizer: Adam
Model: CNNRegression
Activate function: ReLU
Min: 0.0131
Min: 0.0080
Loss
Epoch

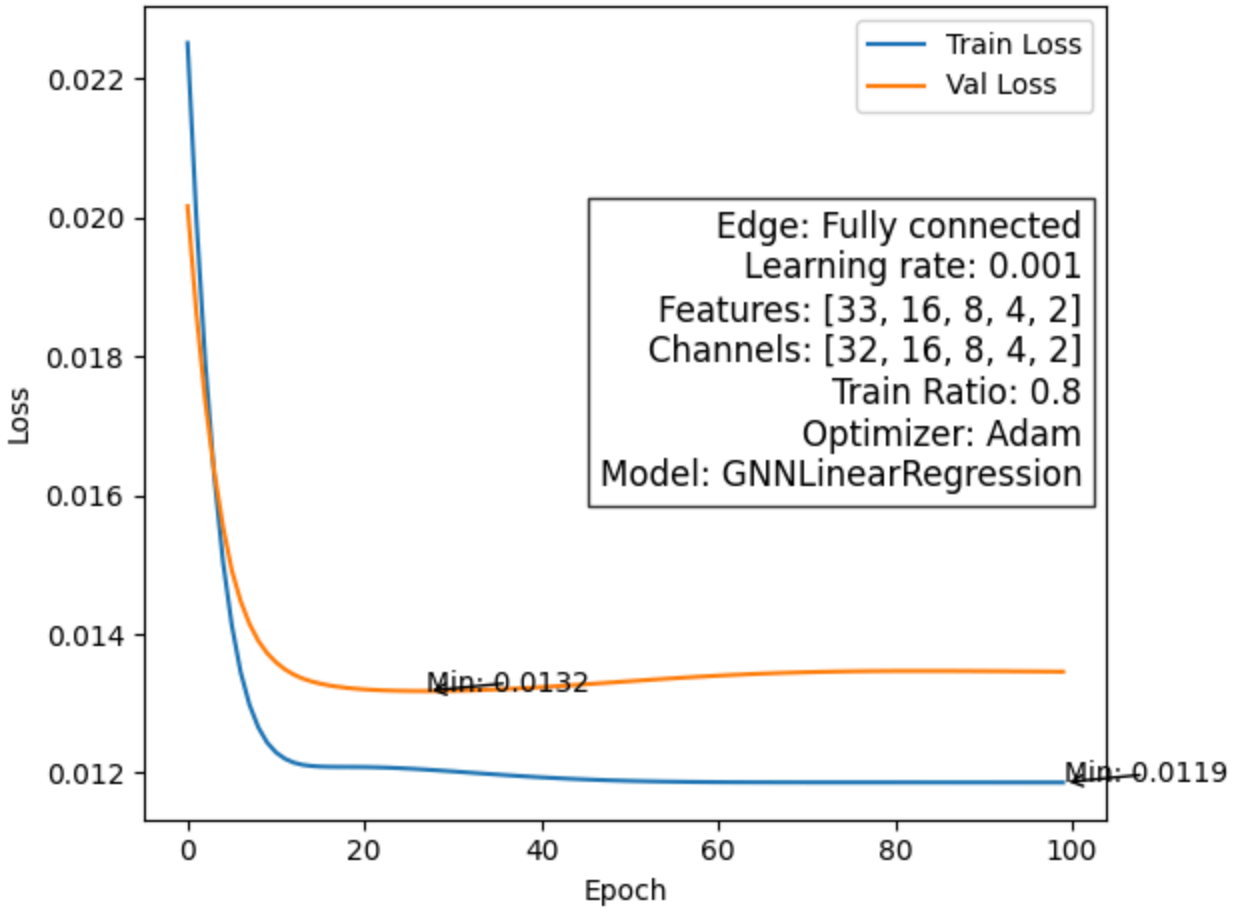

Training Loss
Validation Loss
Learning Rate: 0.003
Dimensions: [7, 11, 15]
Train Ratio: 0.8
Optimizer: Adam
Model: CNNRegression
Activate function: LeakyReLU
Min: 0.0012
Min: 0.0019
Loss
Epoch

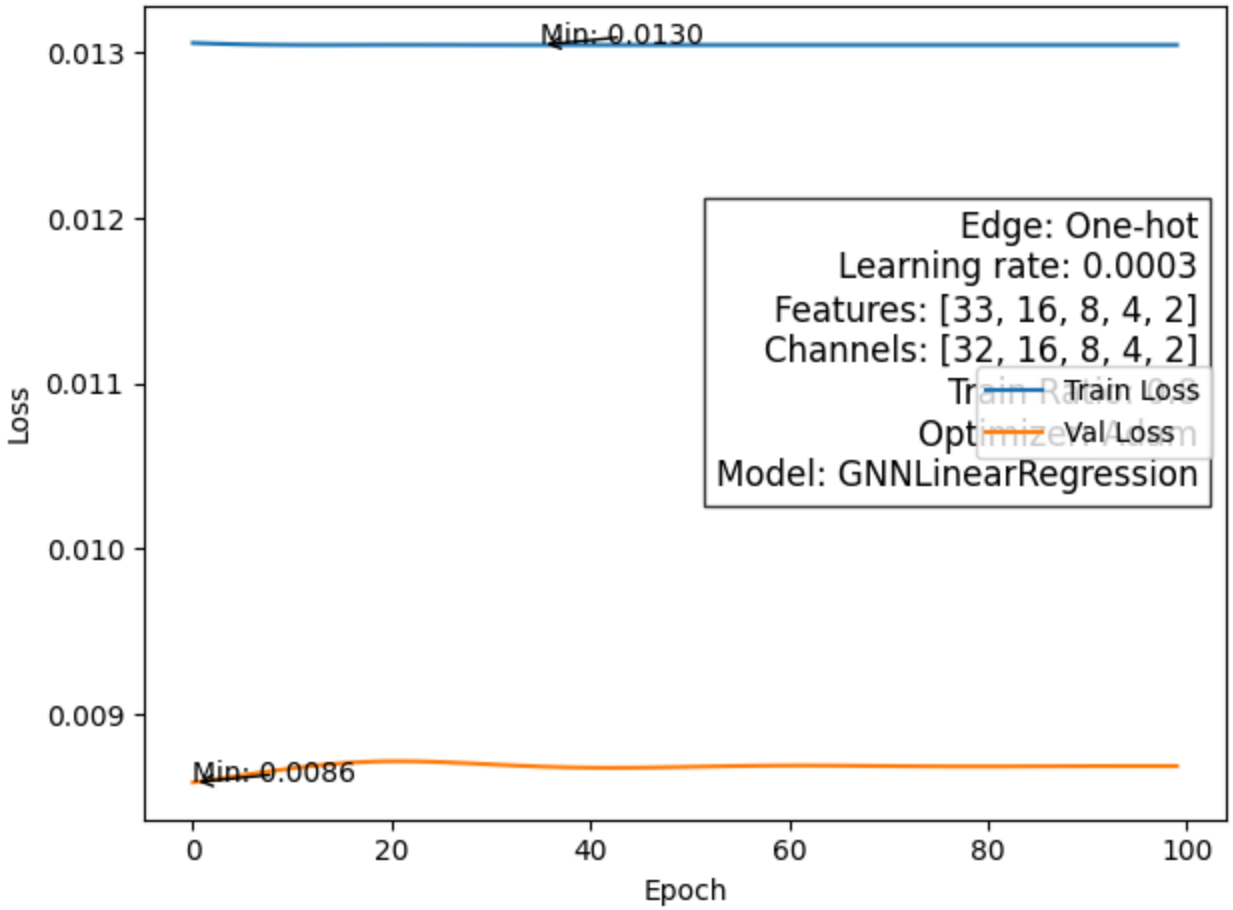


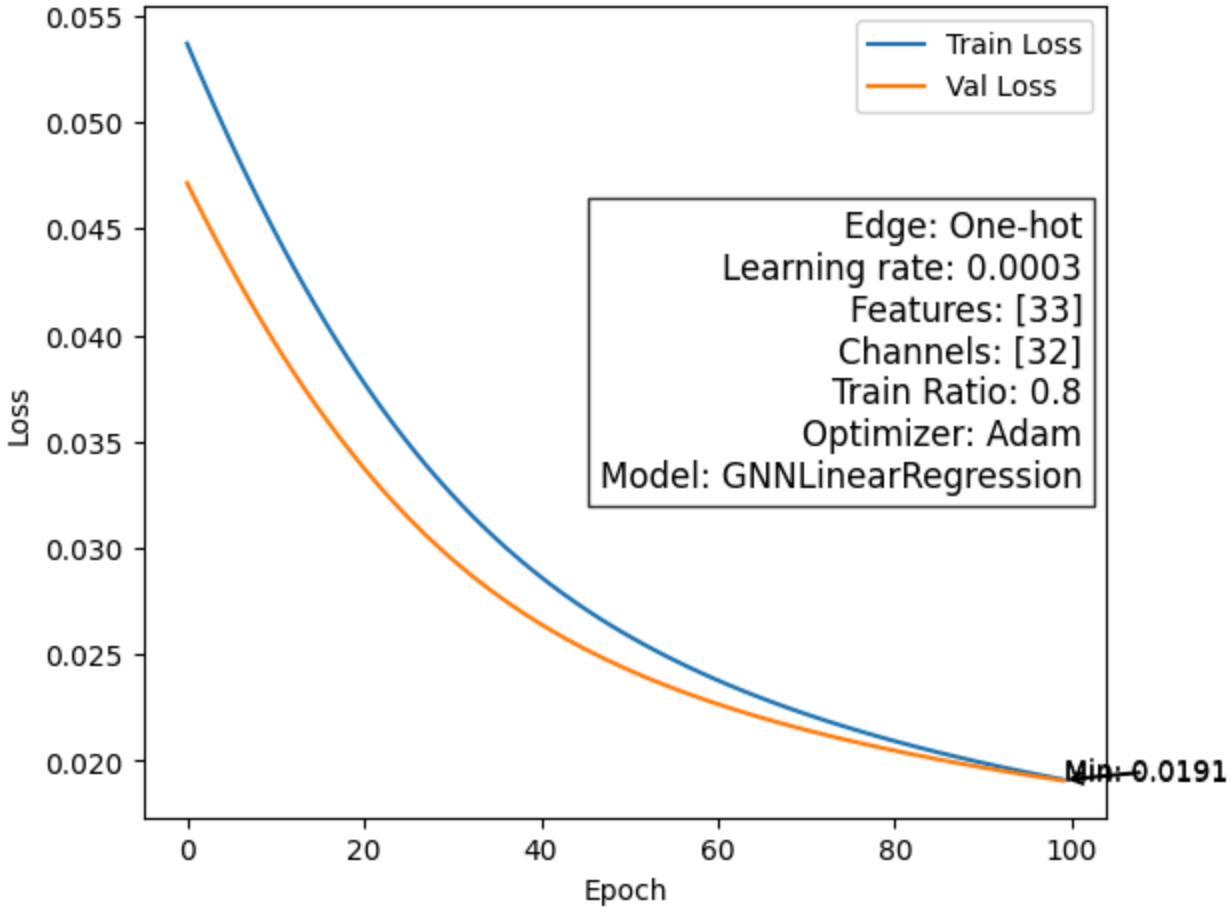
Depth/Dimensions : Keep adding the depth and dimension didn't significantly increase the performance of the model, however it didn't reduce the performance in epochs 50 also. This change affects the slope of the loss plot, the more complex the model be, the more difficult to converge. But if the epochs increase, the complex model will show the sign of overfit, so it is better to keep the model simpler.
Activation function : Use softmax didn't make sense here and the model can not fit well. After comparing Relu and LeakyRelu, it is obvious that Relu is better in this situation as the cross point of training loss and validation loss earlier than the minimum loss, so it should stop early to let it make sense. And if you choose to remove the activation function in cnn, it will reduce the performance.
Train split : Based on the history of machine learning.
Epochs : Use the validation data to know where you need to stop early, if the validation error increases and training loss decreases, you may need to stop at that cross point.
Optimizer : Adam is better than SGG in this case.

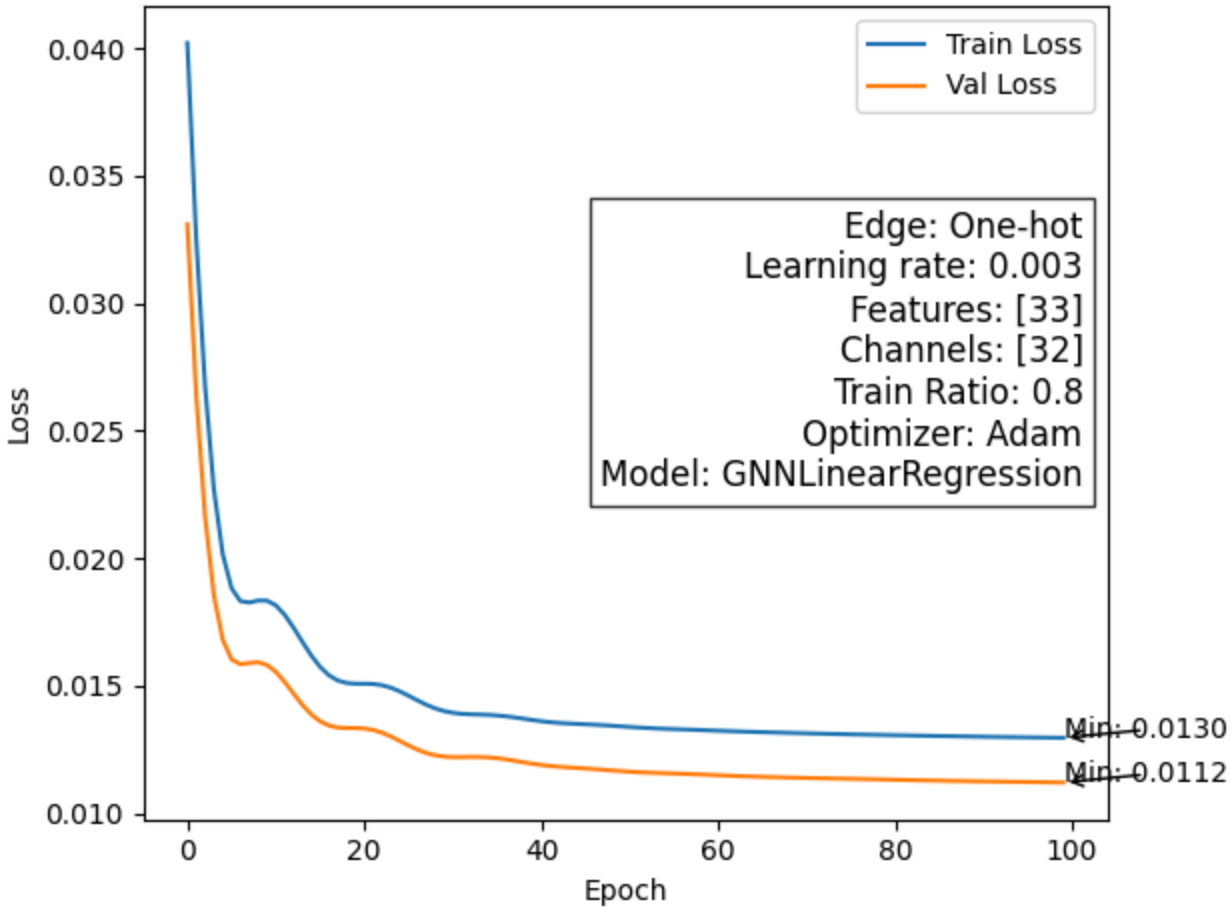
GCN Graph Neural Network Linear regression

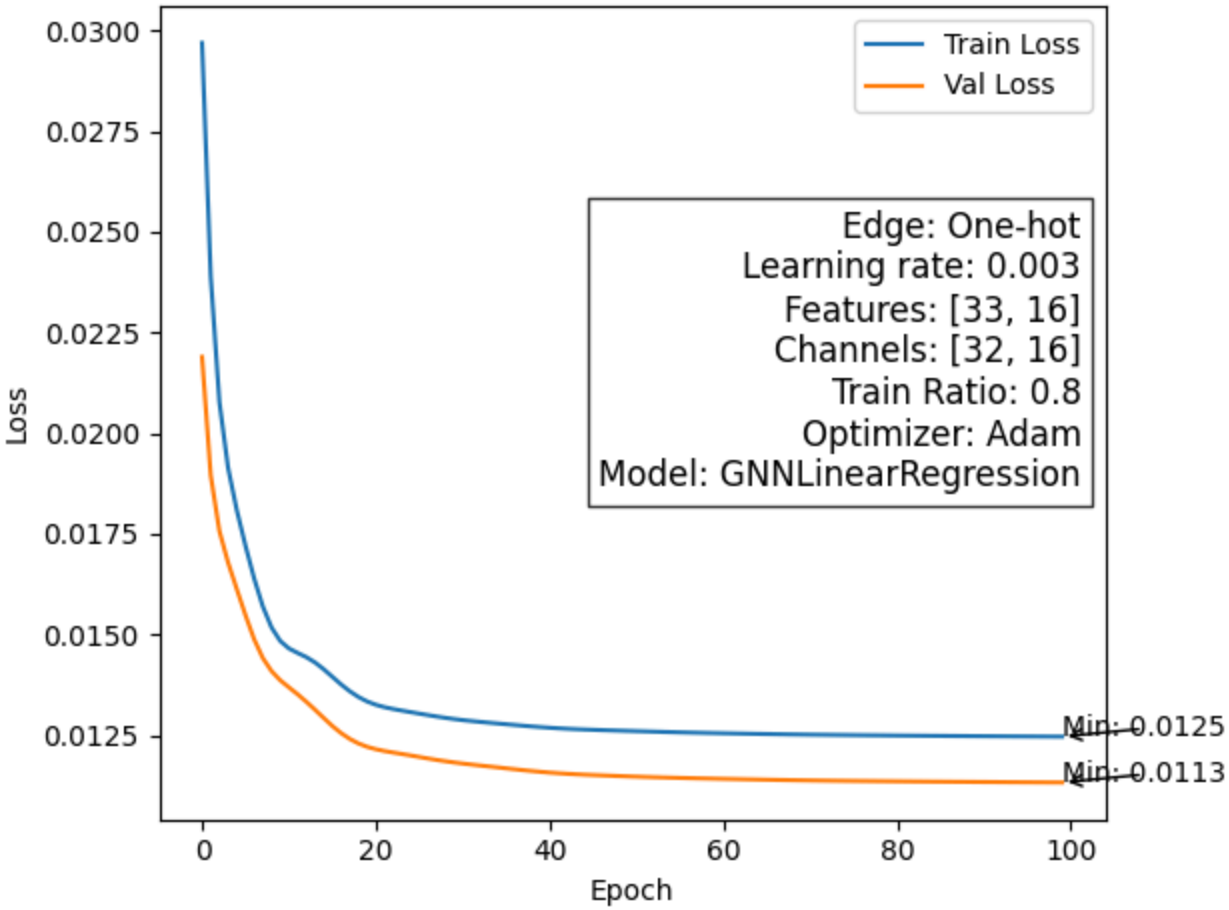

Train Loss
Val Loss
Edge: One-hot
Learning rate: 0.001
Features: [33, 16, 8, 4, 2]
Channels: [32, 16, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0126
Min: 0.0105
Loss
Epoch
0.06
0.05
0.04
0.03
0.02
0.01
0
20
40
60
80
100

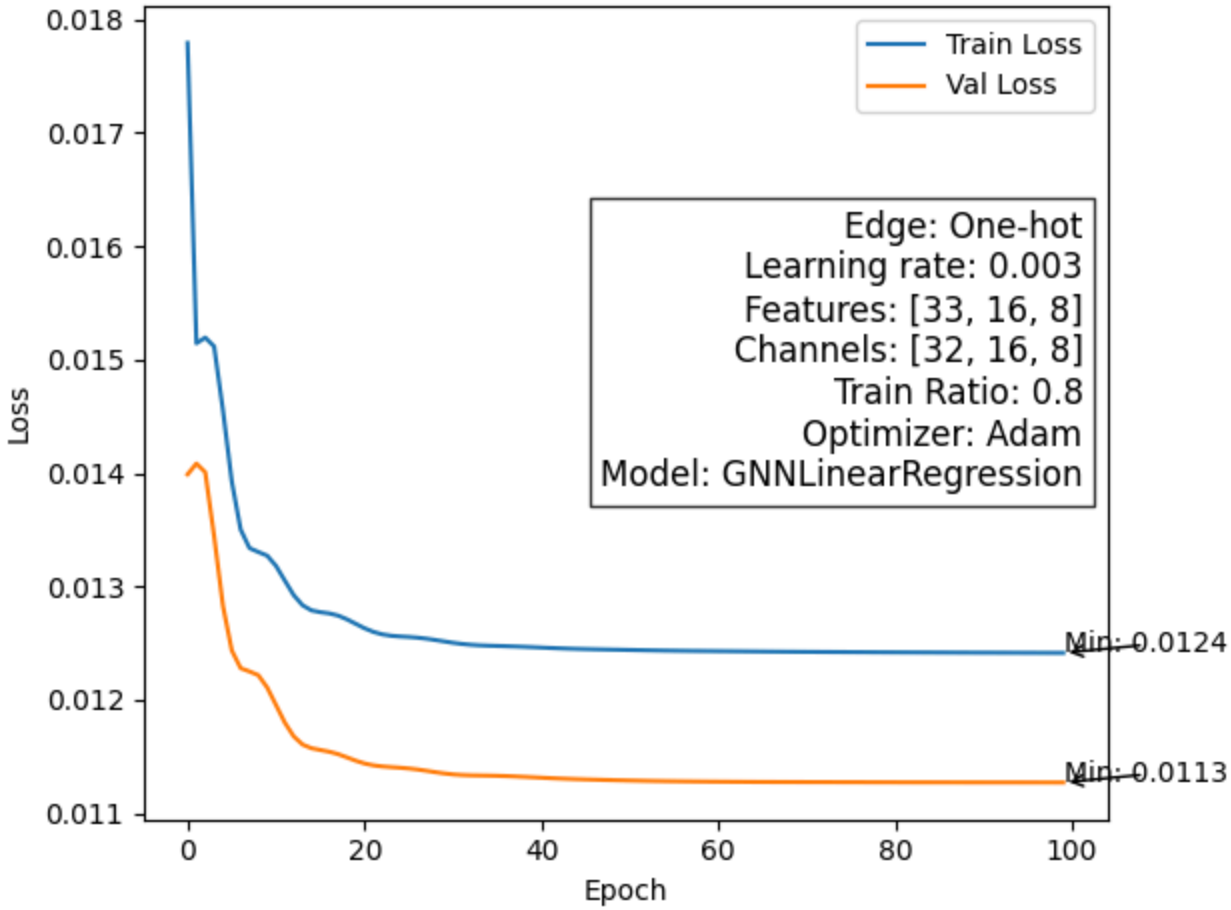

Train Loss
Val Loss
Edge: Fully connected
Learning rate: 0.001
Features: [33, 16, 8, 4, 2]
Channels: [32, 16, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0132
Min: 0.0119
Loss
Epoch
0.022
0.020
0.018
0.016
0.014
0.012
0
20
40
60
80
100

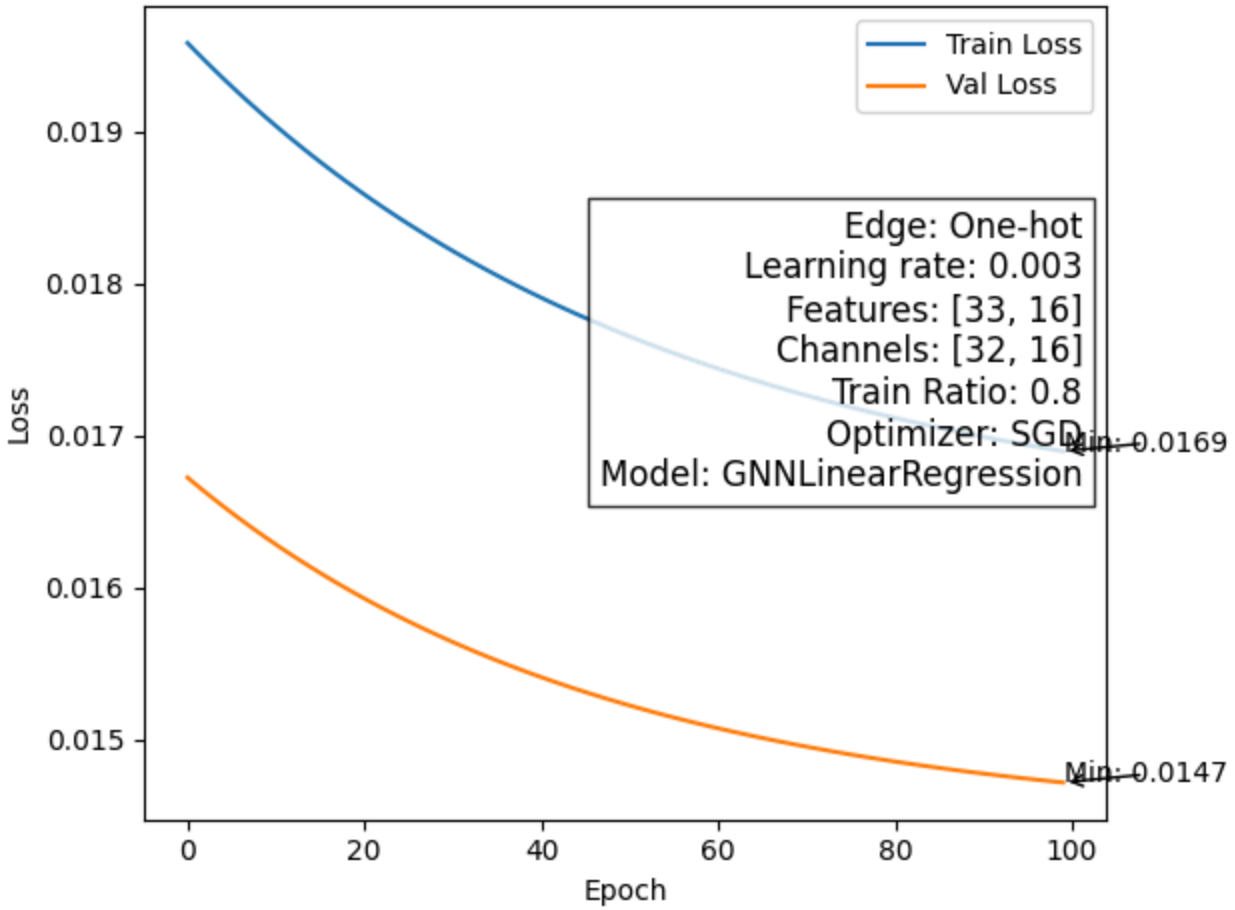

Min: 0.0130
Min: 0.0086
Edge: One-hot
Learning rate: 0.0003
Features: [33, 16, 8, 4, 2]
Channels: [32, 16, 8, 4, 2]
Model: GNNLinearRegression
Train Loss
Val Loss
Loss
Epoch
0.013
0.012
0.011
0.010
0.009
0
20
40
60
80
100


Train Loss
Val Loss
Edge: One-hot
Learning rate: 0.0003
Features: [33]
Channels: [32]
Train Ratio: 0.8
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0191
Loss
Epoch
0.055
0.050
0.045
0.040
0.035
0.030
0.025
0.020
0
20
40
60
80
100


Train Loss
Val Loss
Edge: One-hot
Learning rate: 0.003
Features: [33]
Channels: [32]
Train Ratio: 0.8
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0130
Min: 0.0112
Loss
Epoch
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0
20
40
60
80
100


Train Loss
Val Loss
Edge: One-hot
Learning rate: 0.003
Features: [33, 16]
Channels: [32, 16]
Train Ratio: 0.8
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0125
Min: 0.0113
Loss
Epoch
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0.0150
0.0125
0
20
40
60
80
100


Train Loss
Val Loss
Edge: One-hot
Learning rate: 0.003
Features: [33, 16, 8]
Channels: [32, 16, 8]
Train Ratio: 0.8
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0124
Min: 0.0113
Loss
Epoch
0.018
0.017
0.016
0.015
0.014
0.013
0.012
0.011
0
20
40
60
80
100

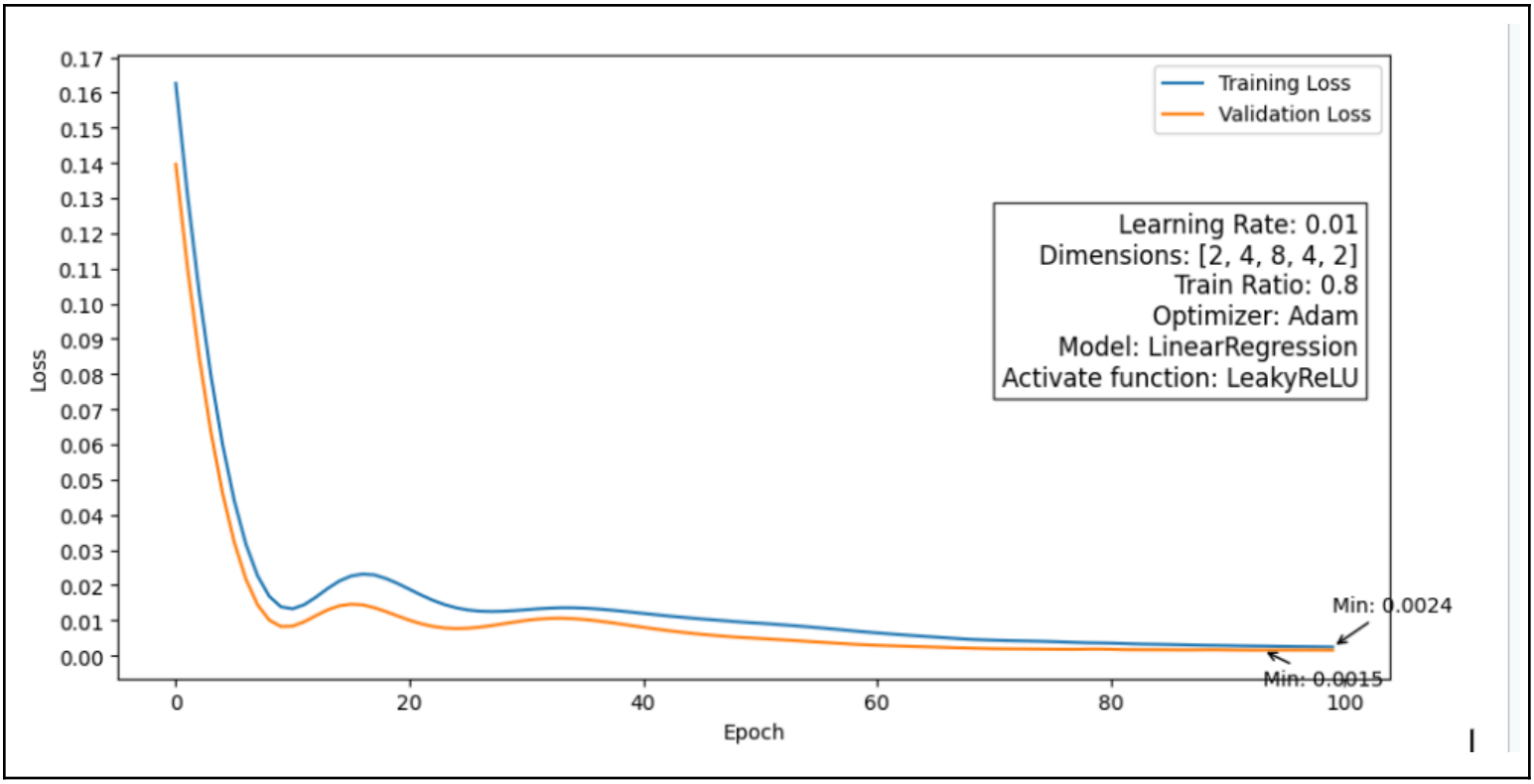


Which model and hyperparameters are best overall? Train it using the combination of the train and validation sets and report test performance.

0.14017

submission.csv
Complete · 10m ago · Leaky
0.14017

LinearRegression model
Learning rate : 0.01
Dimension : [2, 4, 8, 4, 2]
Optimizer : Adam
Activation function : LeakyReLU


Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.8
Optimizer: Adam
Model: LinearRegression
Activate function: LeakyReLU
Min: 0.0024
Min: 0.0015
Loss
Epoch
0.17
0.16
0.15
0.14
0.13
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100

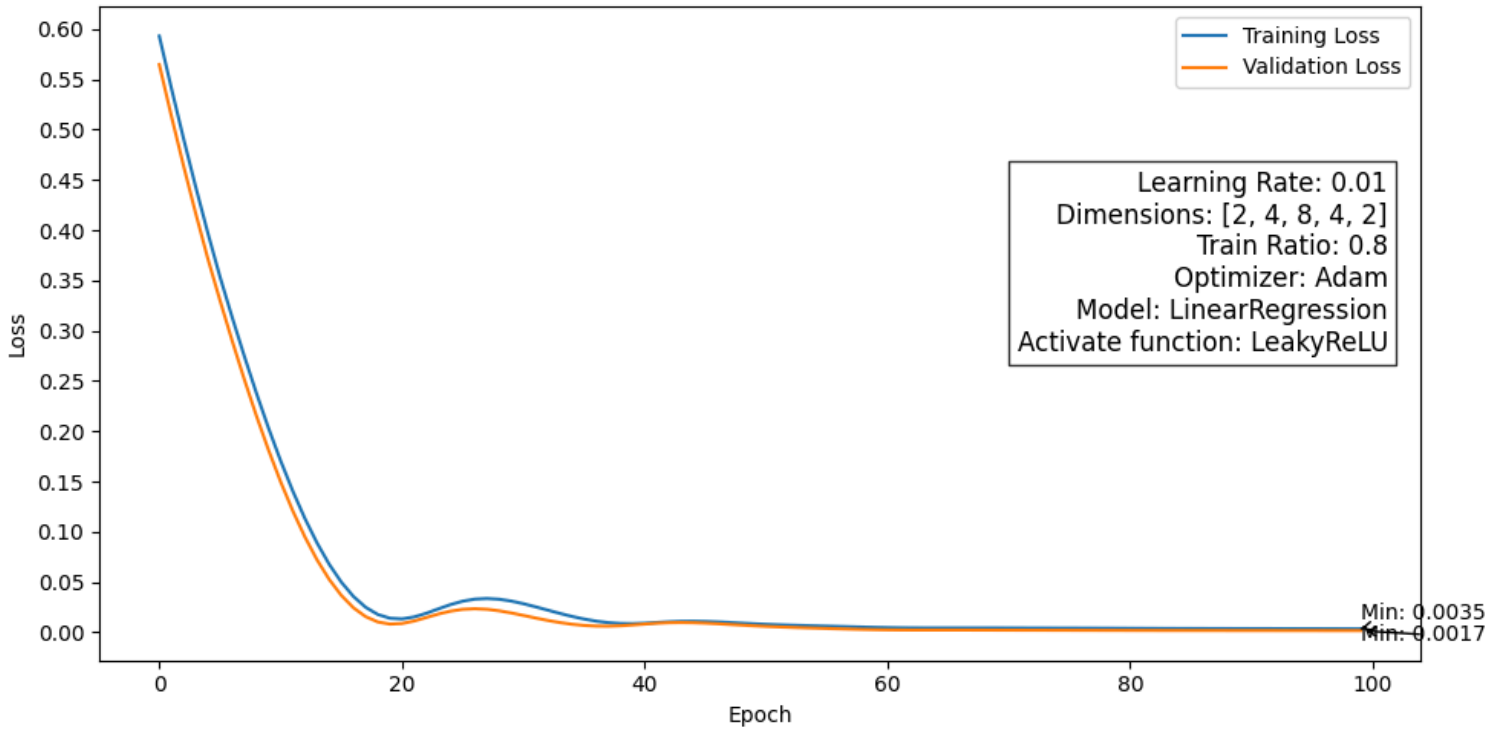
3. Additional Analysis: Either Effect of Training Data Size or Feature Analysis

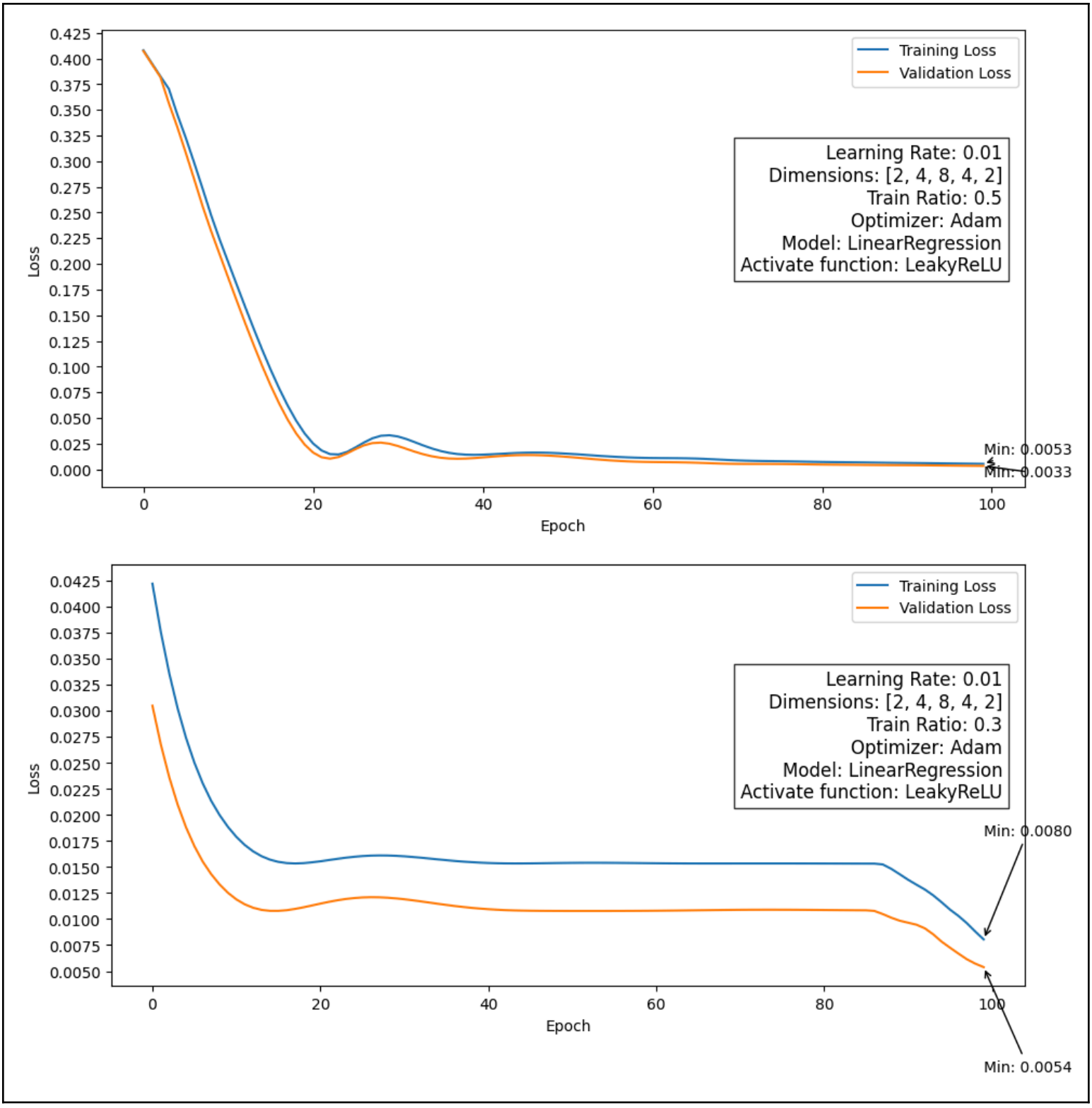
Which type of analysis are you doing?

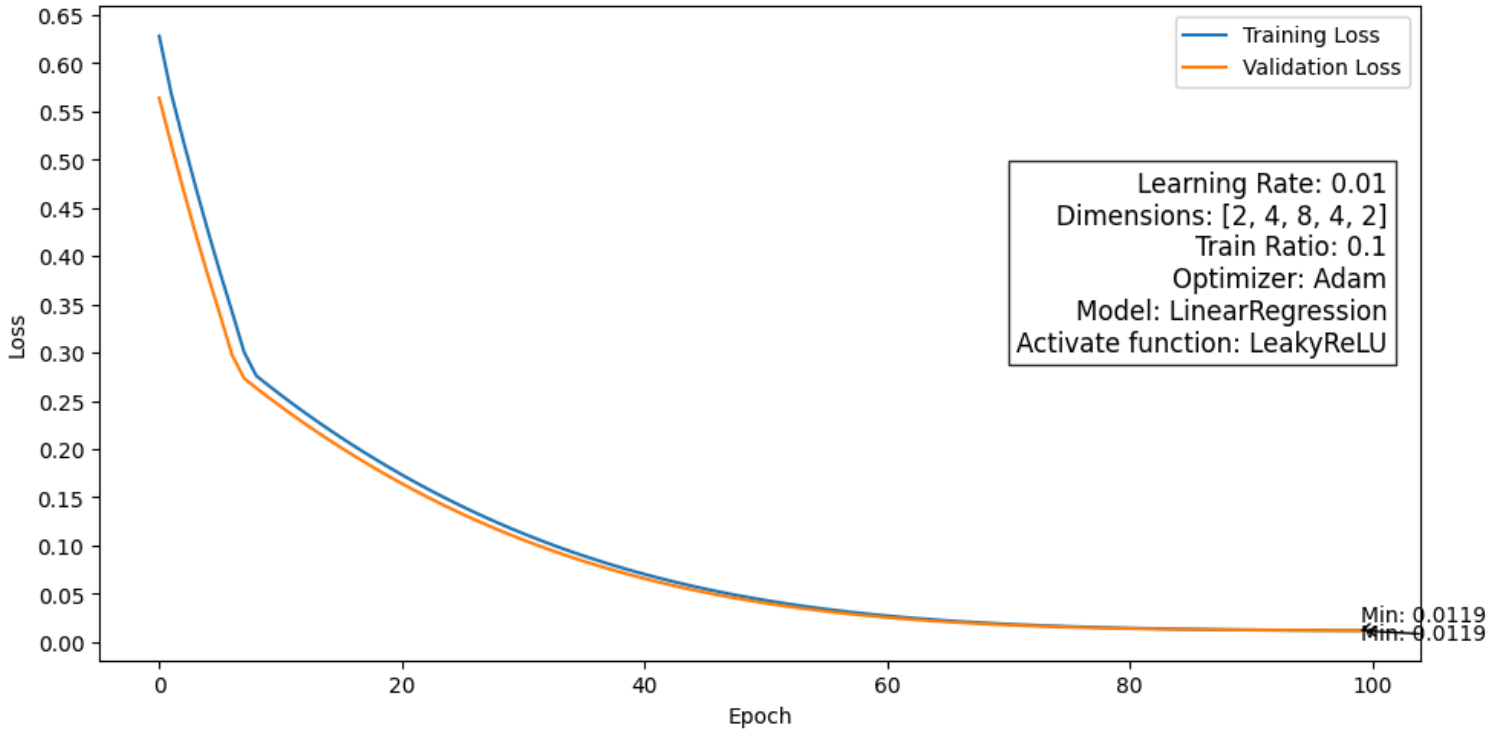
Training size

Present your experiments and conclusions. If evaluating features, include performance of your best model with different subsets of features and importance analysis using L1 regularization, mutual information, or similar. If evaluating training size, compare all three models using the selected hyperparameters.

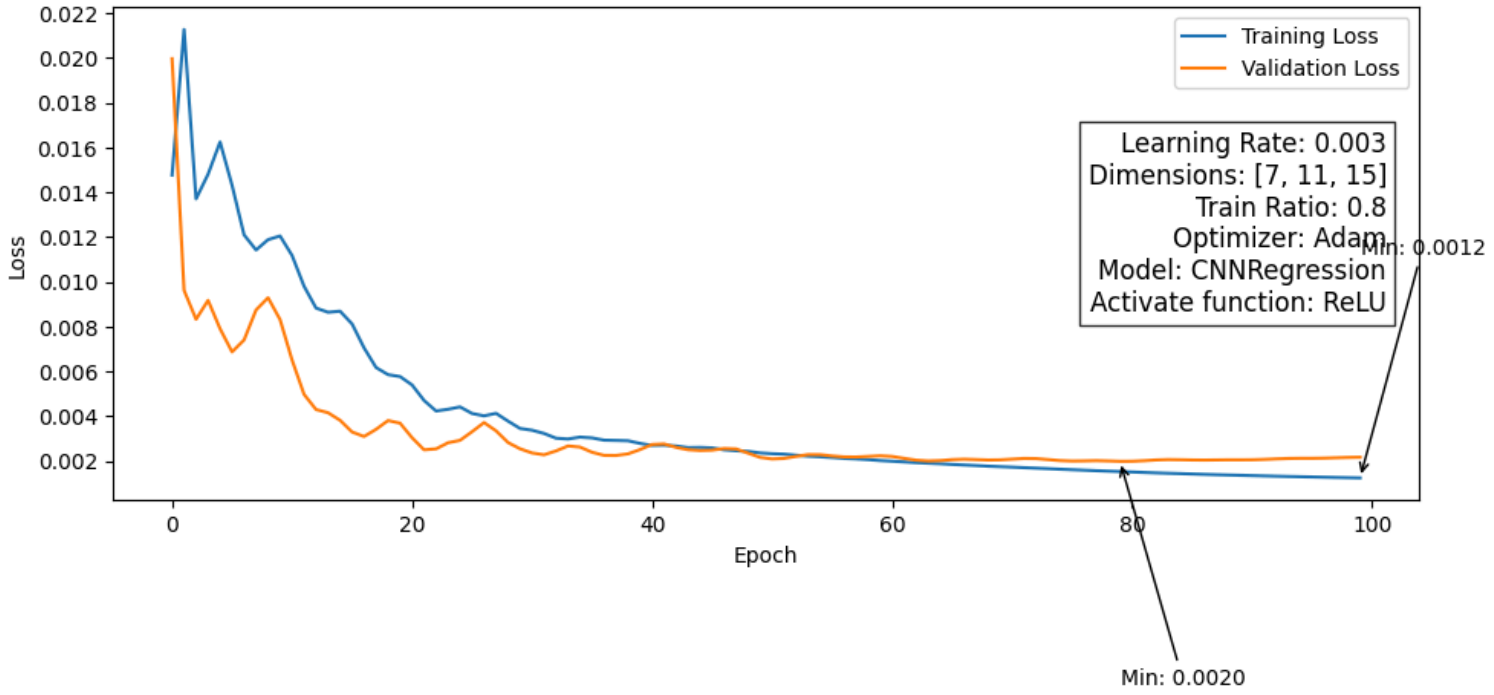
LinearRegression

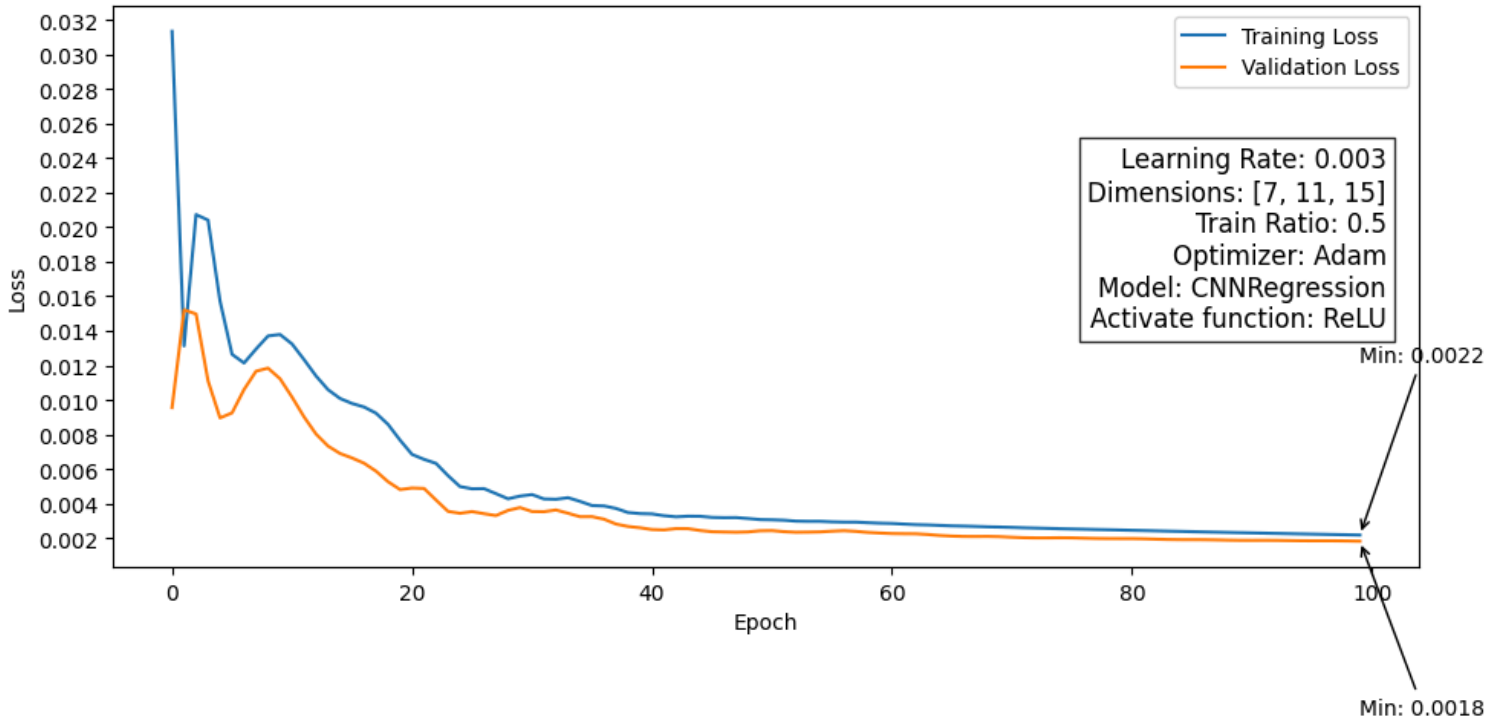


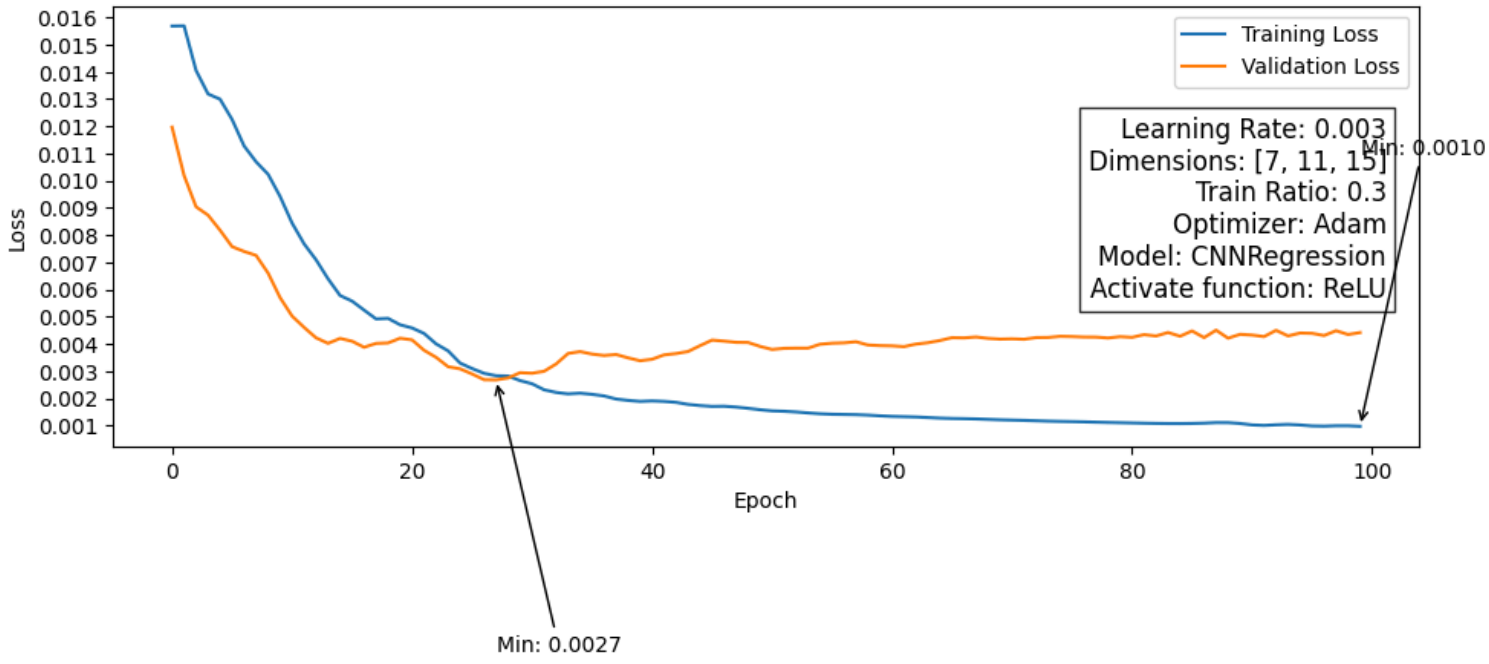

Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.5
Optimizer: Adam
Model: LinearRegression
Activate function: LeakyReLU
Min: 0.0053
Min: 0.0033
Loss
Epoch
Training Loss
Validation Loss
Learning Rate: 0.01
Dimensions: [2, 4, 8, 4, 2]
Train Ratio: 0.3
Optimizer: Adam
Model: LinearRegression
Activate function: LeakyReLU
Min: 0.0080
Min: 0.0054
Loss
Epoch

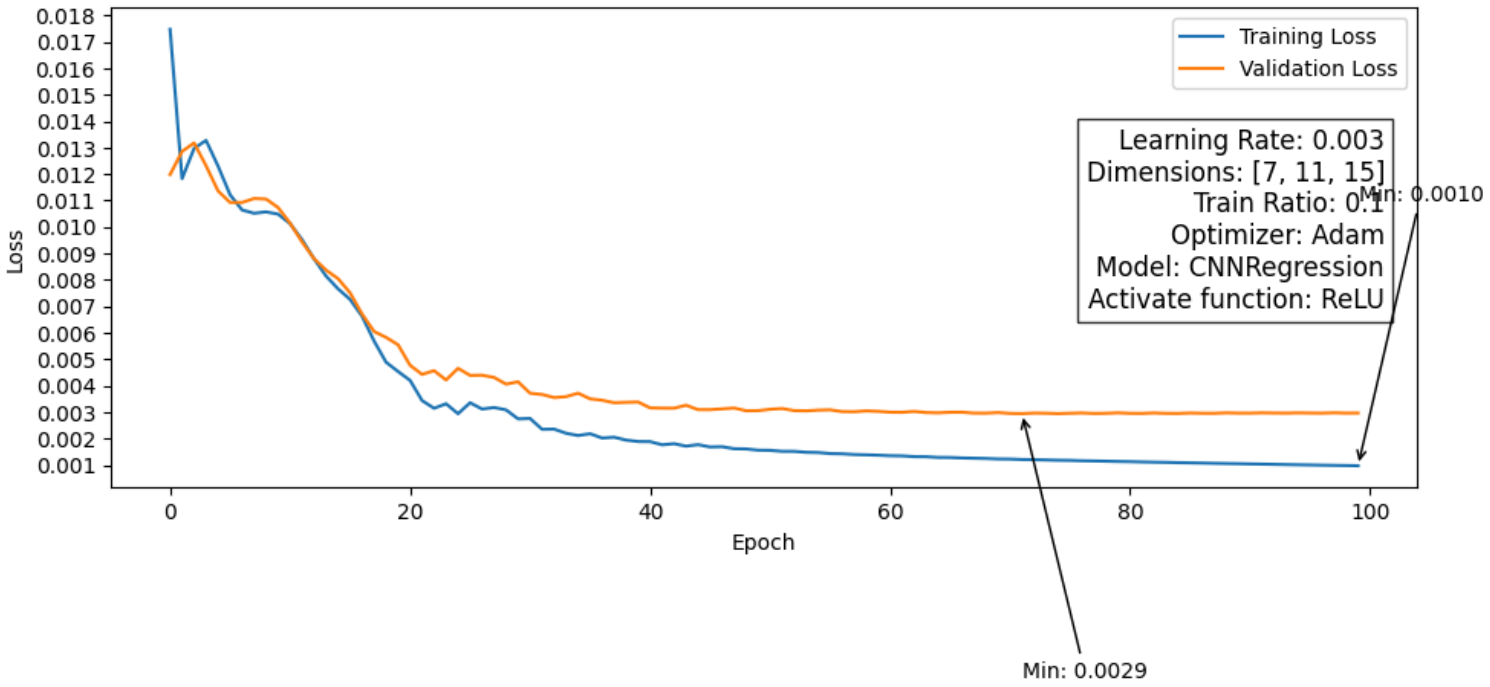


CNNRegression

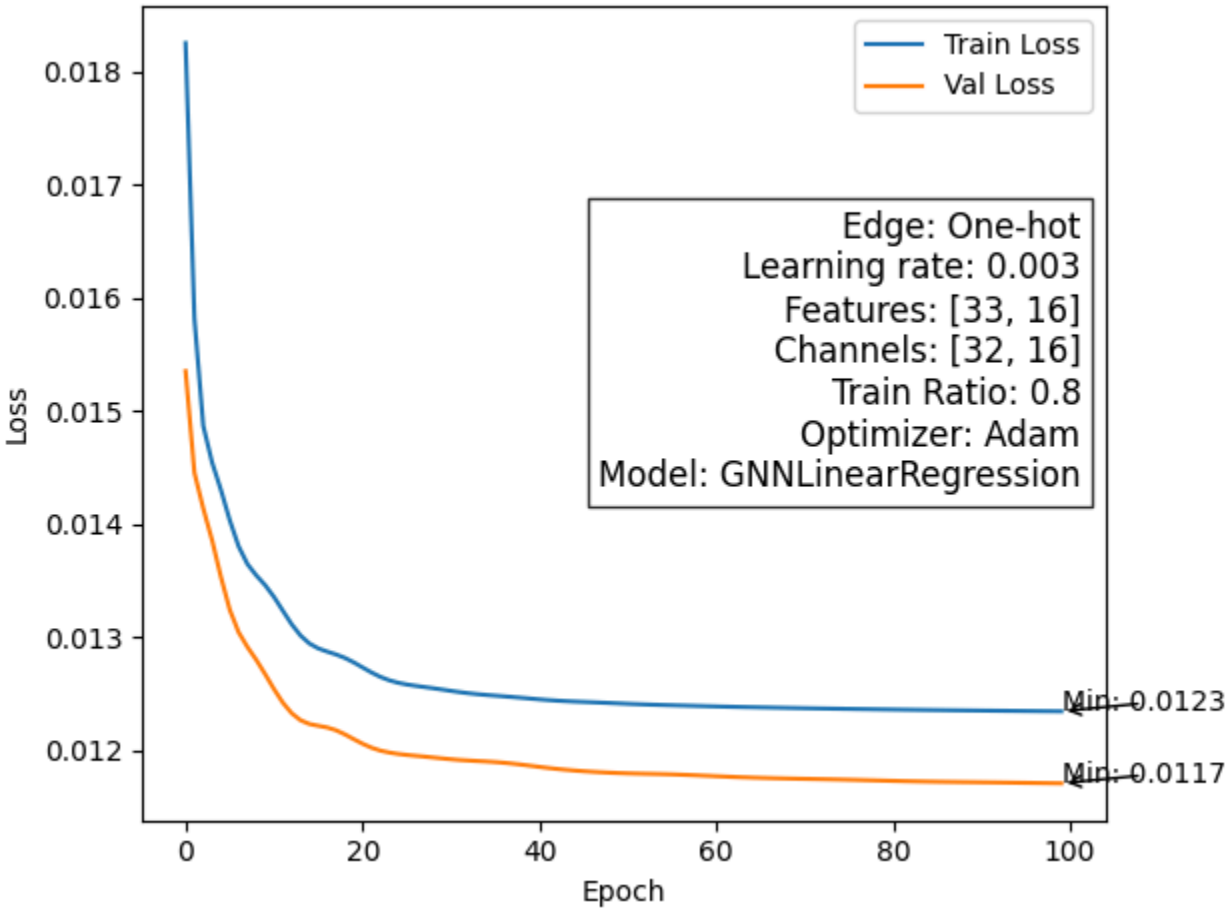


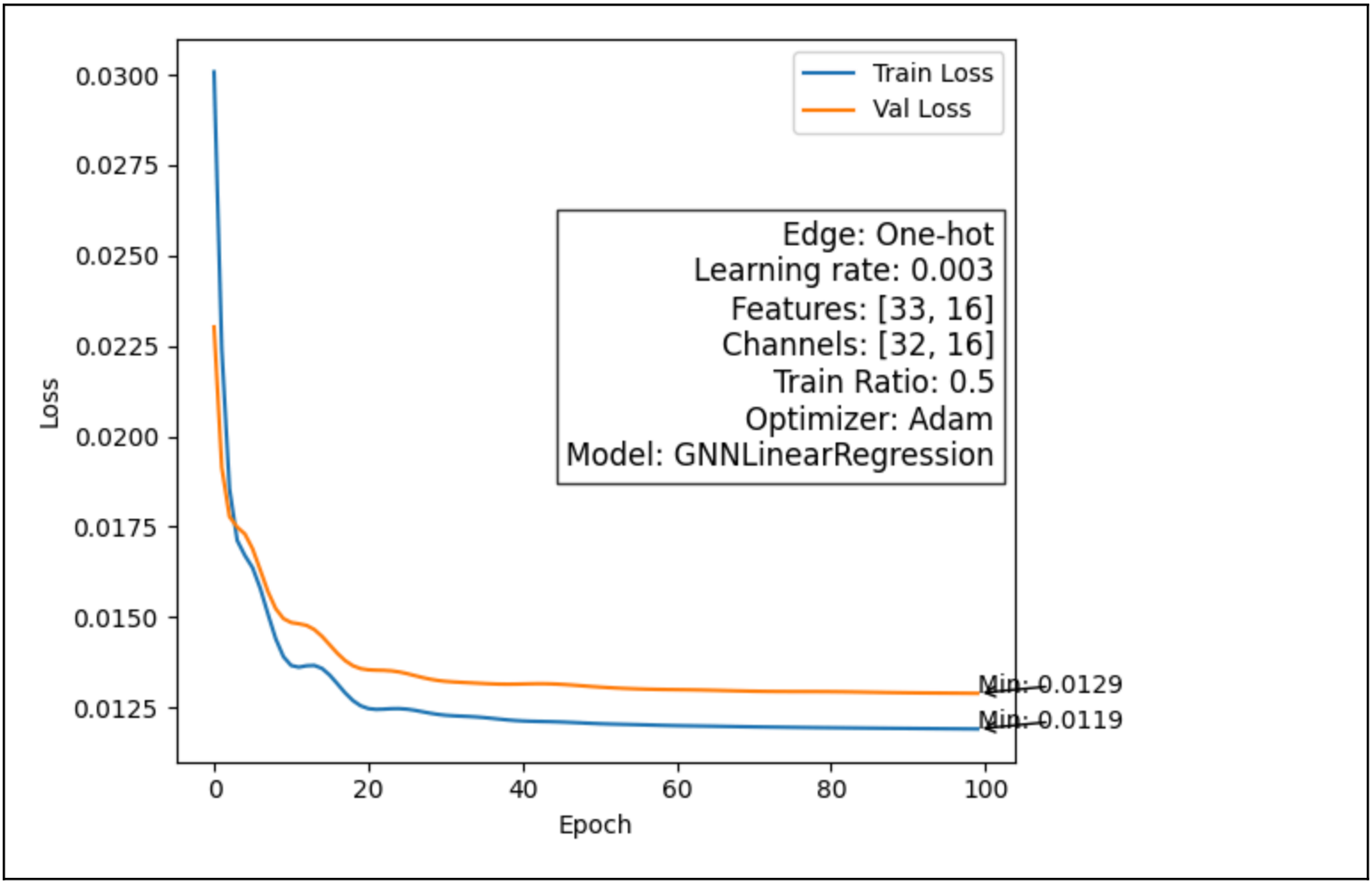

Training Loss
Validation Loss
Learning Rate: 0.003
Dimensions: [7, 11, 15]
Train Ratio: 0.5
Optimizer: Adam
Model: CNNRegression
Activate function: ReLU
Min: 0.0022
Min: 0.0018
Loss
Epoch

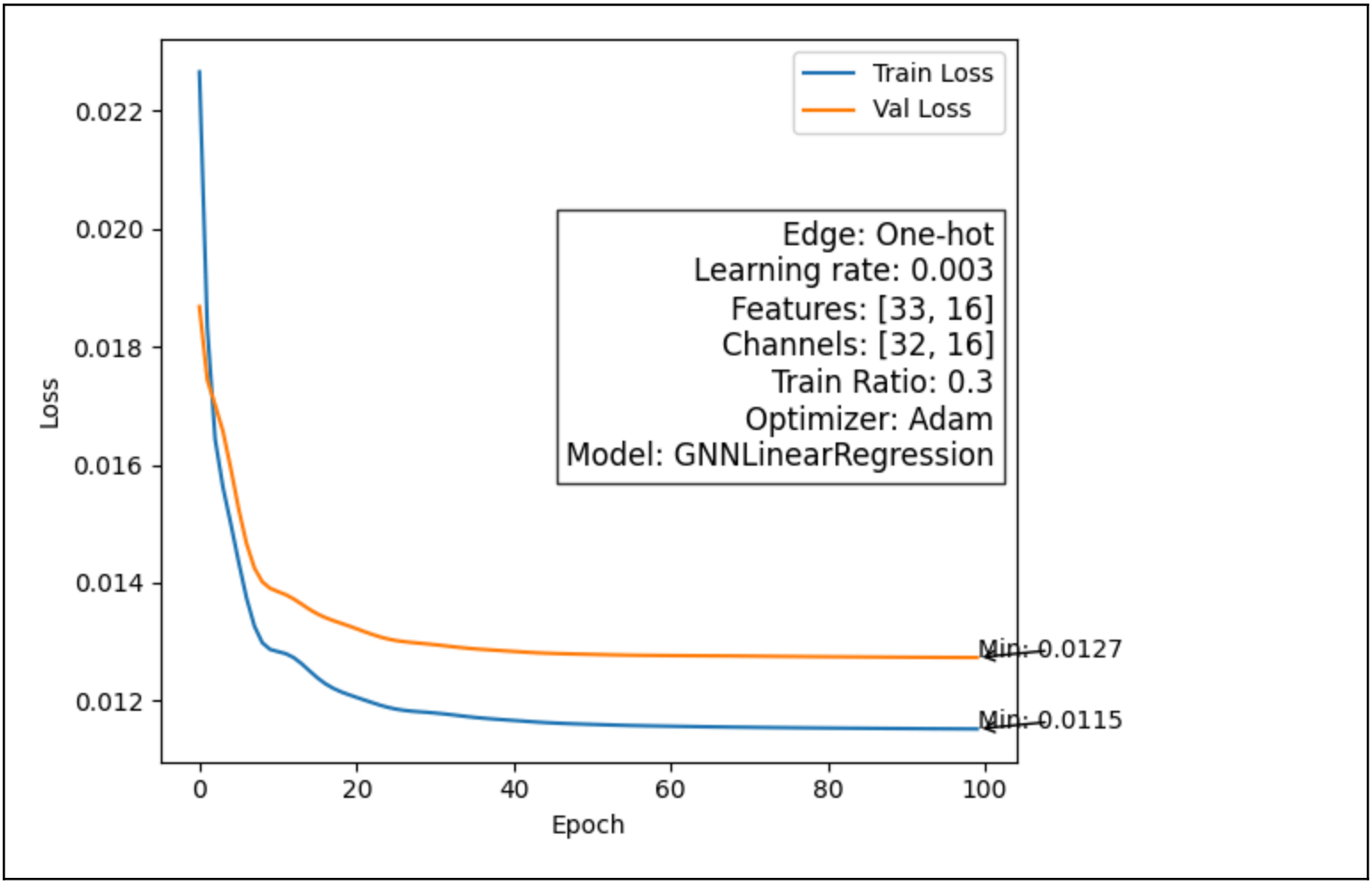

Training Loss
Validation Loss
Learning Rate: 0.003
Dimensions: [7, 11, 15]
Train Ratio: 0.3
Optimizer: Adam
Model: CNNRegression
Activate function: ReLU
Min: 0.0010
Min: 0.0027
Loss
Epoch

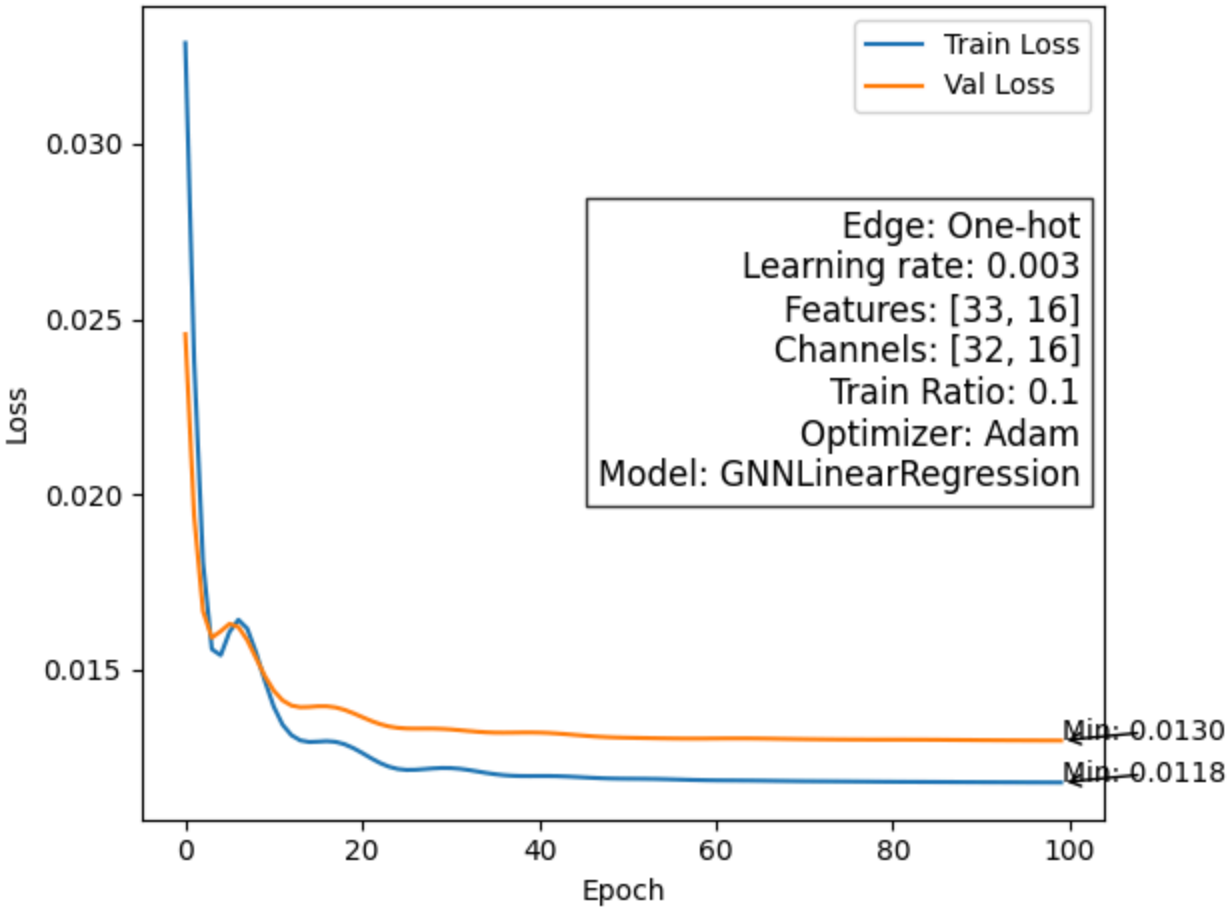


GNNRegression




Train Loss
Val Loss
Edge: One-hot
Learning rate: 0.003
Features: [33, 16]
Channels: [32, 16]
Train Ratio: 0.5
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0129
Min: 0.0119
Loss
Epoch
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0.0150
0.0125
0
20
40
60
80
100


Train Loss
Val Loss
Edge: One-hot
Learning rate: 0.003
Features: [33, 16]
Channels: [32, 16]
Train Ratio: 0.3
Optimizer: Adam
Model: GNNLinearRegression
Min: 0.0127
Min: 0.0115
Loss
Epoch
0.022
0.020
0.018
0.016
0.014
0.012
0
20
40
60
80
100



We can see that both LinearRegression and CNNRegression do better in larger train ratio, the larger train size, and the performance slightly decrease as the train ratio decreases. This because that LinearRegression and CNNRegression need more data, the more data it receive that we believe that them will have better performance. As we believe that our model is doing well. However, it might get some good results at the lower training size if and only if just randomly select the most representative data for training, otherwise the performance will be worse. For GNNRegression, there are no any significantly different as we change the training size. We believe that is because of the message passing of the graph neural network, it enables information exchange and aggregation among nodes, so sometimes it will be overfit, but in this case with a smaller training size, the graph neural network can learn well with only small data.

4. Stretch Goal: Innovation

Describe your proposed approach

Traditionally, we will use simple linear regression or CNN to solve the linear regression problem like this house price prediction case. But now we could try an innovative way to solve the problem, using Graph Neural Network, a different type of neural network.

Provide and explain the experimental analysis of your proposed approach

Edge : In this case if we want to use the graph neural network to train the model, we need to identify the relations between data to build the edges. The easiest way is using fully-connected edges, which means that all data have relations with all other data, however we can know from the experiments that this is not good enough as it doesn't make sense that the data should be fully-connected. So we would use the data that we didn't use before, that is all object(string) data that needs to be further encoded for training. We use all that data to build the edge index of the graph neural network, we believe that this makes sense to represent the relations of the data.
Features / Channels : The dimensions of the graph neural network are difficult to choose as the graph neural network is more difficult to train than traditional CNN. In this case, we believe that the model should not be too complex as the message passing of neural network will forward the information to all others nodes, so it might be overfit if we add too many dimensions / channels. And the results prove the hypothesis we made that the deeper the network be will not improve the performance of the results.

What is innovative about your approach?

Graph neural network use topology to represent data. There are two important features, nodes and edges.
Unlike traditional convolution neural network use matrix to represent data, graph neural network can accept various input size and different feature size as it use topology. In this case, we use homogeneous graph in graph neural network as there is no extra data can use heterogeneous graph. We use nodes to represent each row and encode the feature in the nodes of numerical data. Then we use one-hot encoding to know the relation of different data and use them to build the edge index. Then we use GCN to train the network.

Is your approach potentially publishable? If not (the usual case), skip this part. If so, explain the research contributions and how it fits in with related work. What venue would you consider for publication?

I don’t think so. The concept could be borrowed, I think that topology is more detailed way to represent some kinds of data, however there are still some barriers that need to be solved to further improve graph neural network.

5. Acknowledgments / Attribution

Link to your code (required!!)

https://github.com/wctsai3/CS441-Project

Link to your data (if not pre-selected)

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data

External citations or resources

https://www.kaggle.com/code/fazzie/pytorch-regression

Reference for data preprocessing

Group member contributions

wctsai3 : build model and do experiments
chuhaoh2 : build model and do experiments

Did group members contribute roughly equally or unequally? If unequally, explain and specify whether one member went above and beyond, or someone contributed less than agreed or expected.

Equally

50/50