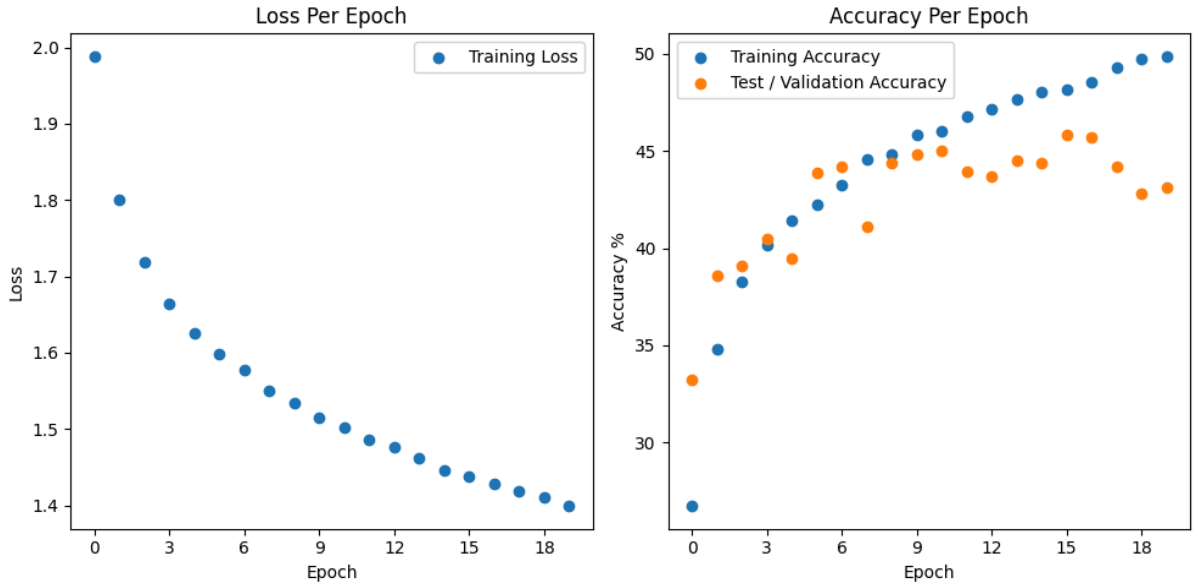
Problem 1

Develop a multi-layer perceptron with three hidden layers (you pick the dimensions of the hidden layers) for the CIFAR-10 dataset.

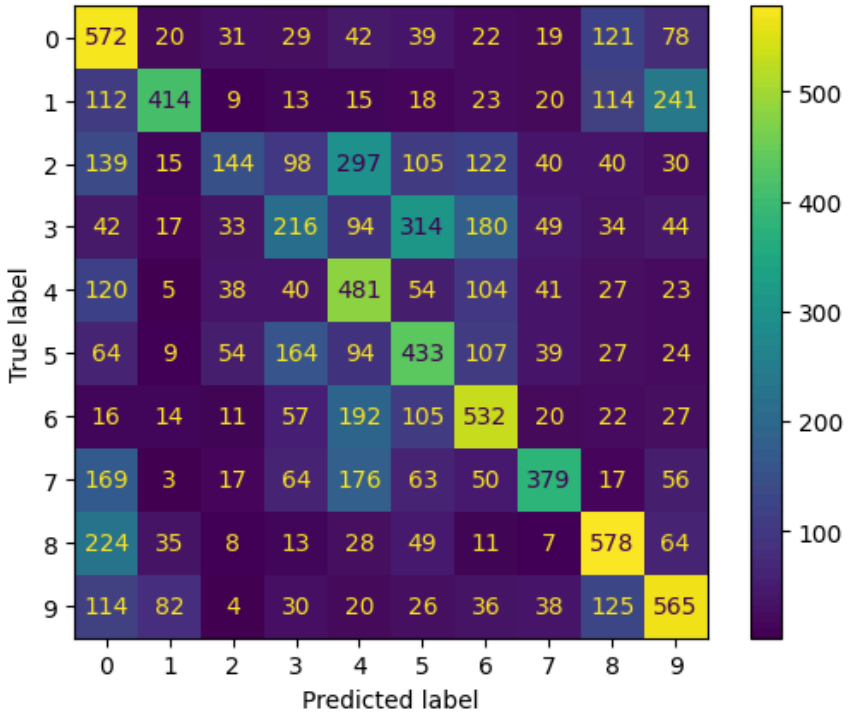
1.a. Train the model from scratch (with randomized parameters) and plot the results (training loss and accuracy, validation accuracy) after 20 epochs. Does your network need more epochs for full training? Do you observe overfitting? Make sure to save the trained parameters and model. Report and plot your training and validation results. Report precision, recall, F1 score, and confusion matrix. (25pt)

Model Width: 100 Neurons
Model Depth: 3 Hidden + 1 Output

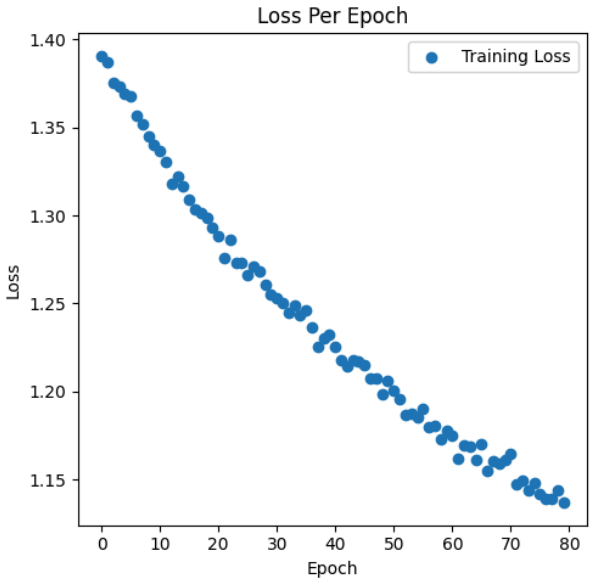
20 Epochs:

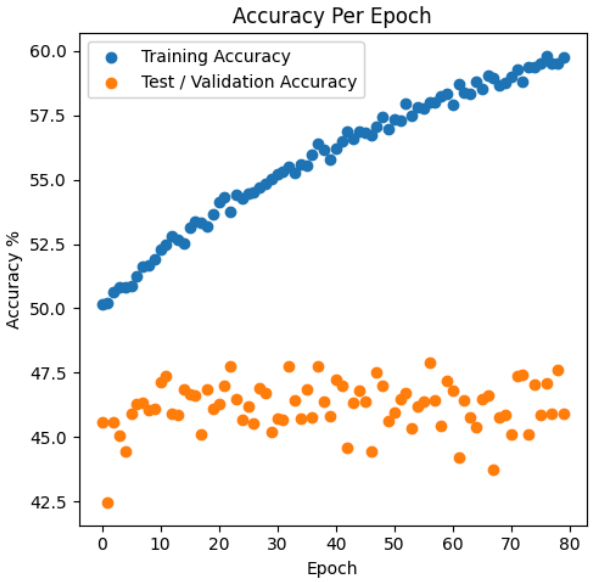


Property	Final Value Against Test Dataset
Accuracy	43.14%
Precision	44.85%
Recall	43.31%
F1 Score	42.29%

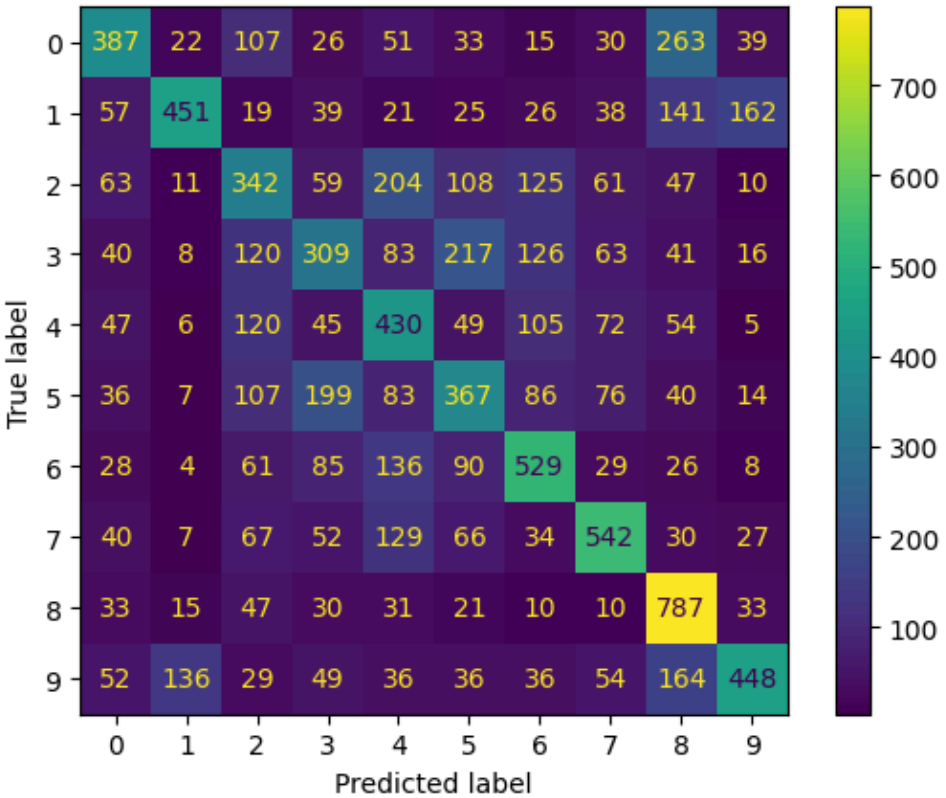


80 Additional Epochs (100 Total):





Property	Final Value Against Test Dataset
Accuracy	45.92%
Precision	49.94%
Recall	45.96%
F1 Score	45.67%

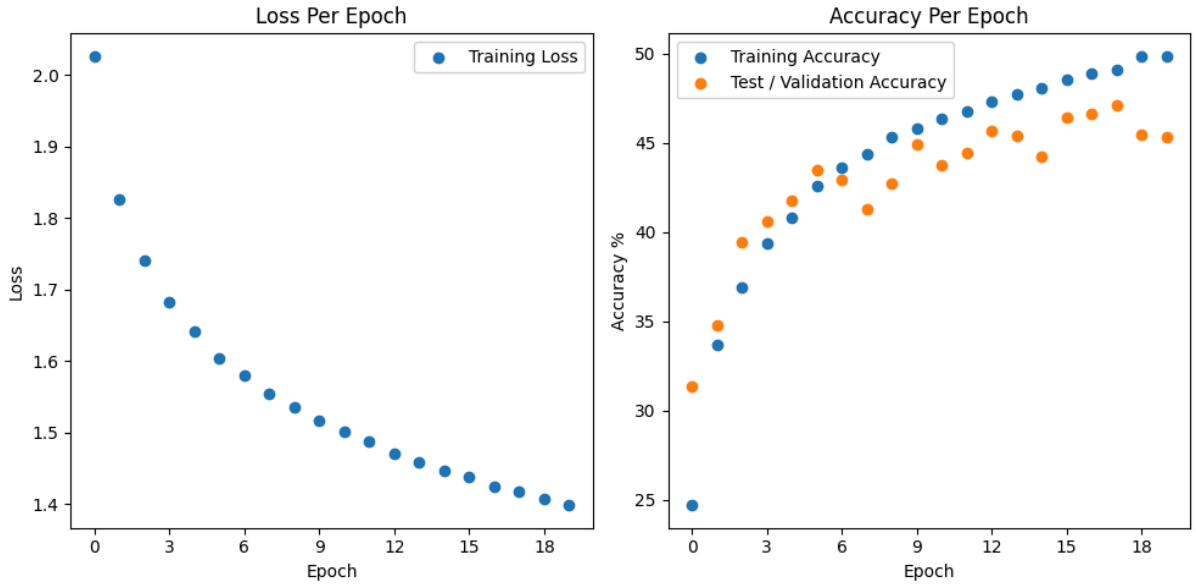


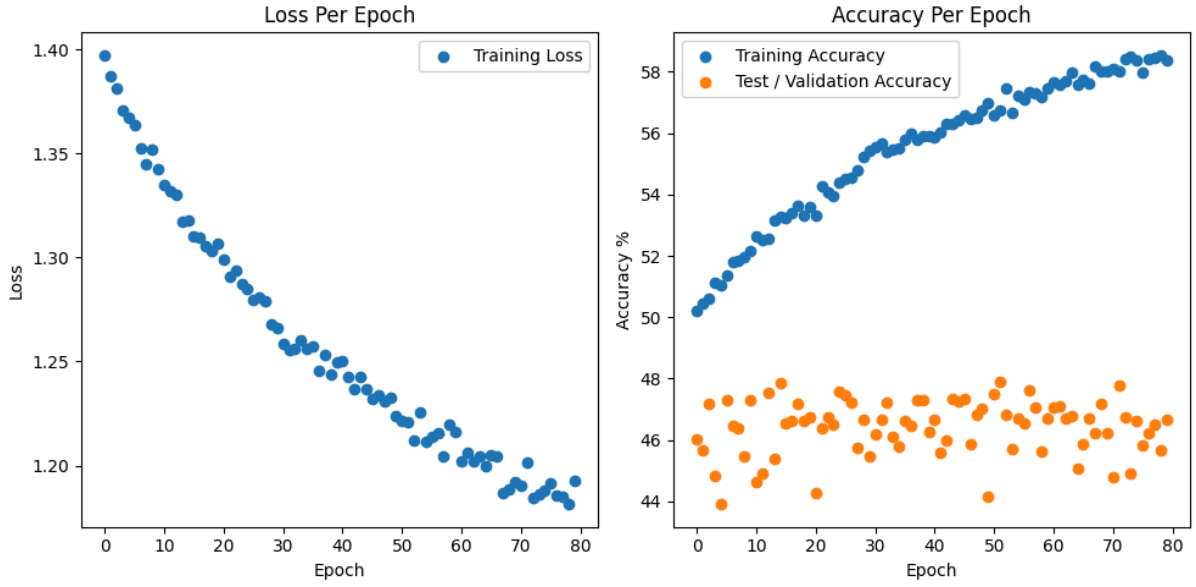
Does your network need more epochs for full training? Do you observe overfitting?
Potentially around 5 more epochs of training (after 20 epochs) would be good for me to consider the model ‘fully trained’.

There is significant overfitting as seen by the large gap between training and test accuracy in the plots.

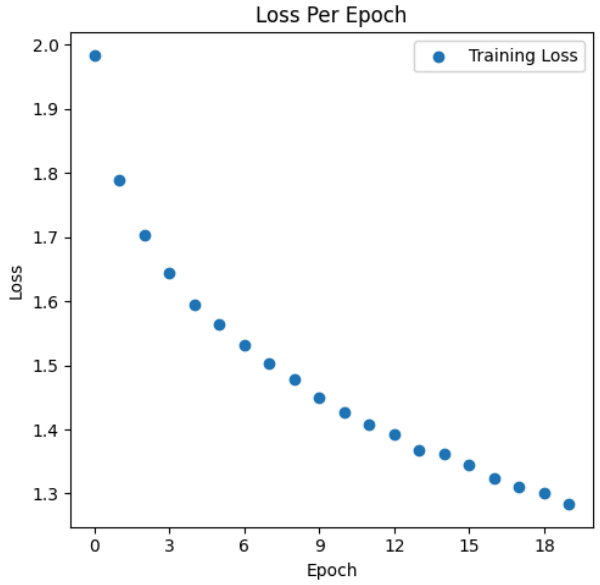
1.b. Explore the complexity of the network by increasing its width and depth. How do the training and validation results change? Compare them against the baseline. Do you see any overfitting? (25pt)

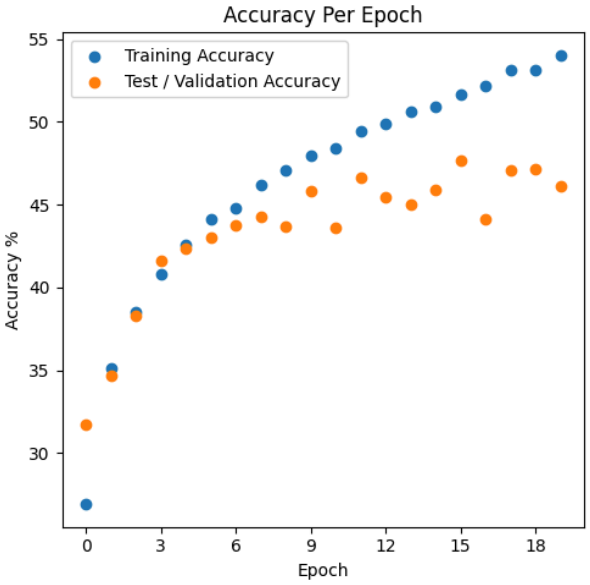
Model Width: 100 Neurons
Model Depth: 4 Hidden + 1 Output

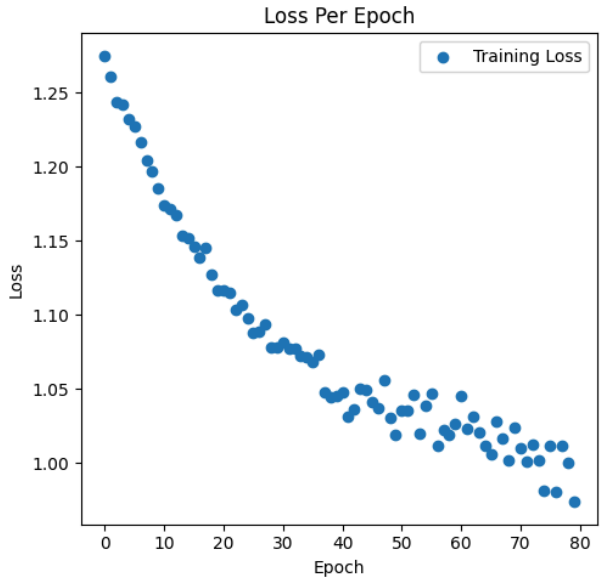


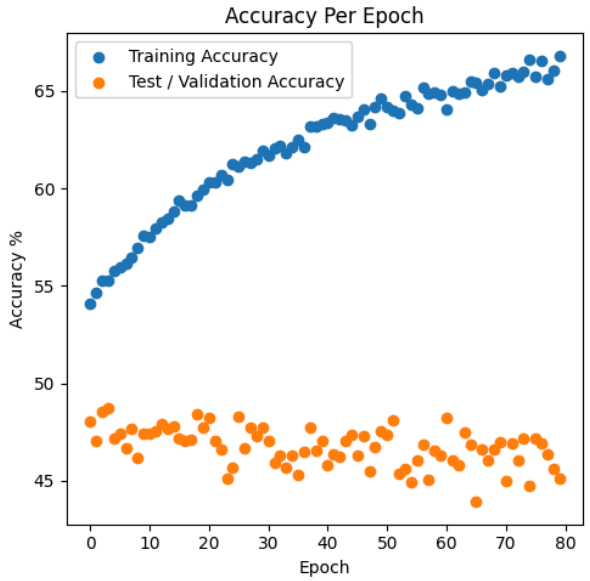


Model Width: 200 Neurons
Model Depth: 3 Hidden + 1 Output









How do the training and validation results change? Compare them against the baseline. Do you see any overfitting?

Looking at the plots from a macro point of view, increasing the complexity has a tendency to increase overfitting. This is shown in that as the complexity increases, the gap between training and test (validation) accuracy increases.

However if one focuses on the end result, it will notice that despite the increased overfitting, the test results do see an improvement across the board.

Complexity	Metric	Value	Baseline Δ
100 x 4	Accuracy	45.31%	2.17%
100 x 4	Precision	46.02%	1.17%
100 x 4	Recall	45.51%	2.20%
100 x 4	F1 Score	43.72%	1.43%
1000 x 3	Accuracy	46.14%	3.00%
1000 x 3	Precision	47.46%	2.61%
1000 x 3	Recall	46.12%	2.81%
1000 x 3	F1 Score	45.73%	3.44%

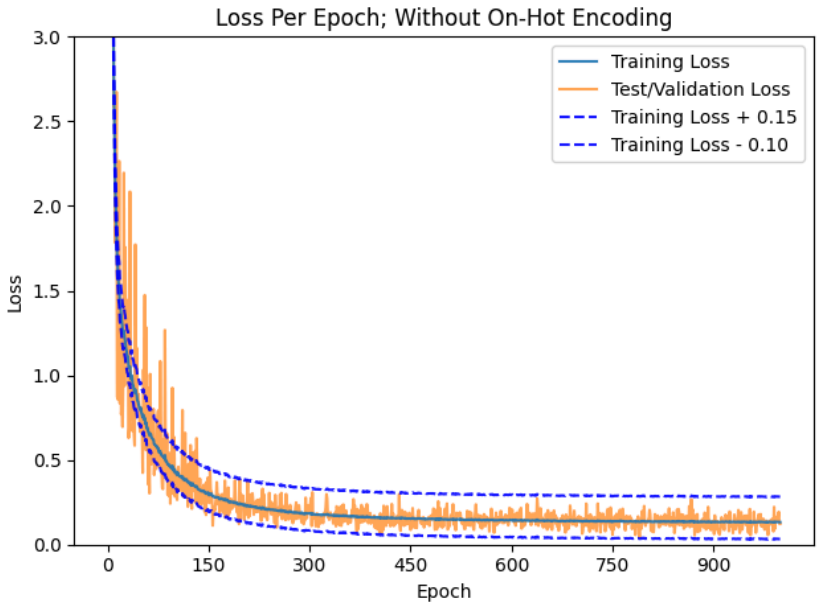
Note: Table tabulated at the 20th Epoch mark

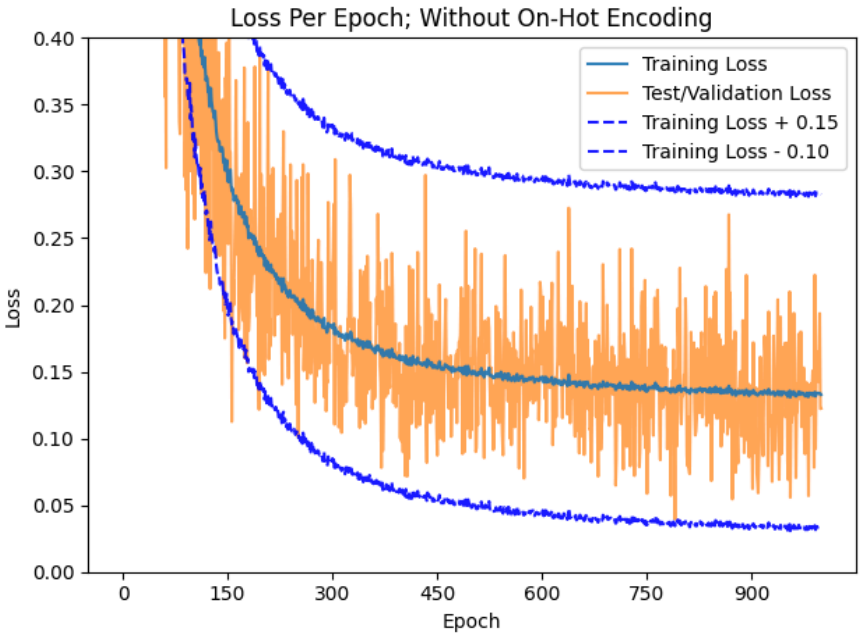
Problem 2

Please implement the following steps for the housing dataset we overviewed during the lectures.

2.a. Build a multi-perceptron network that regresses the housing price (based on 20%, 80% split). Use the same number of features we did in the lecture without on-hot encoding. Please plot the training and validation results and report final accuracy and model complexity(20pt)

Model Width: 100 Neurons
Model Depth: 3 Hidden + 1 Output
Encoding: Binary Encoding ONLY. Dropped *furnishingstatus* field.
Training Time: 1000 Epochs





Report final accuracy and model complexity:
The test/validation loss per epoch optimally fit each other. There doesn't appear to be any over or underfitting at the macro scale.

Looking at it from a micro (per epoch) scale and towards the end of the training, the test loss can deviate from the training loss by:

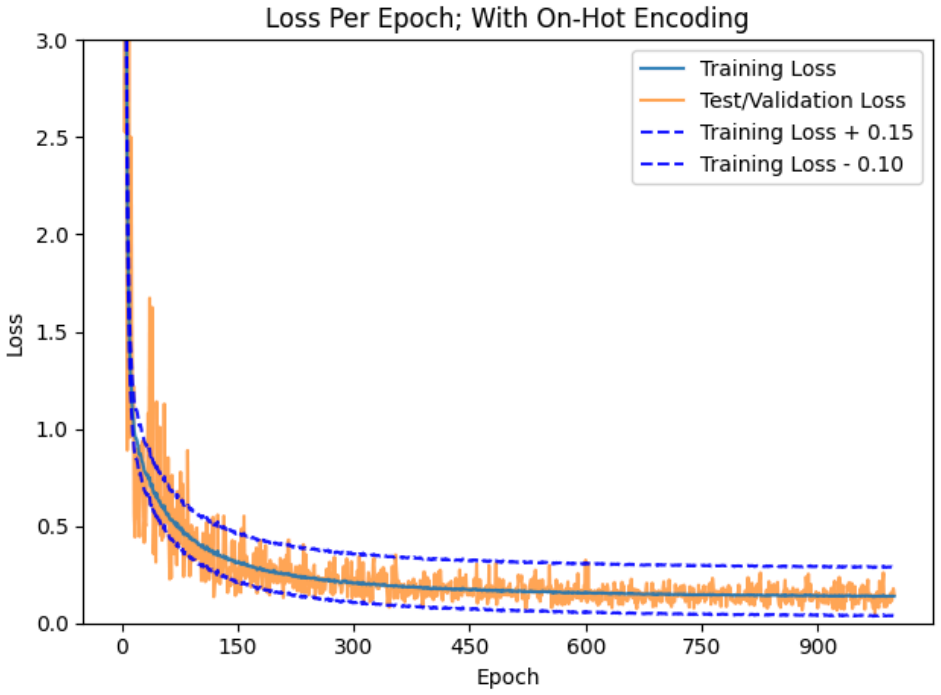
$$\text{Test Loss} \approx \text{Training Loss} - 0.10 \leq \text{Training Loss} \leq \text{Training Loss} + 0.15$$

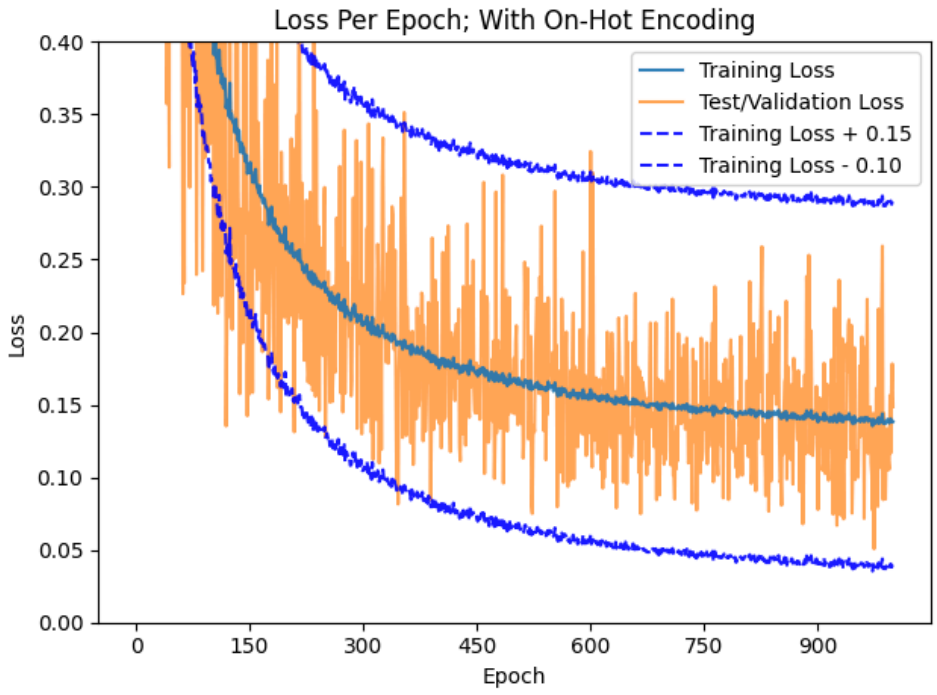
I would say the model is optimally complex for the current scenario (at least up to 1000 epochs).

The continued downward trend for the graph indicates the model can be trained further for increased accuracy.

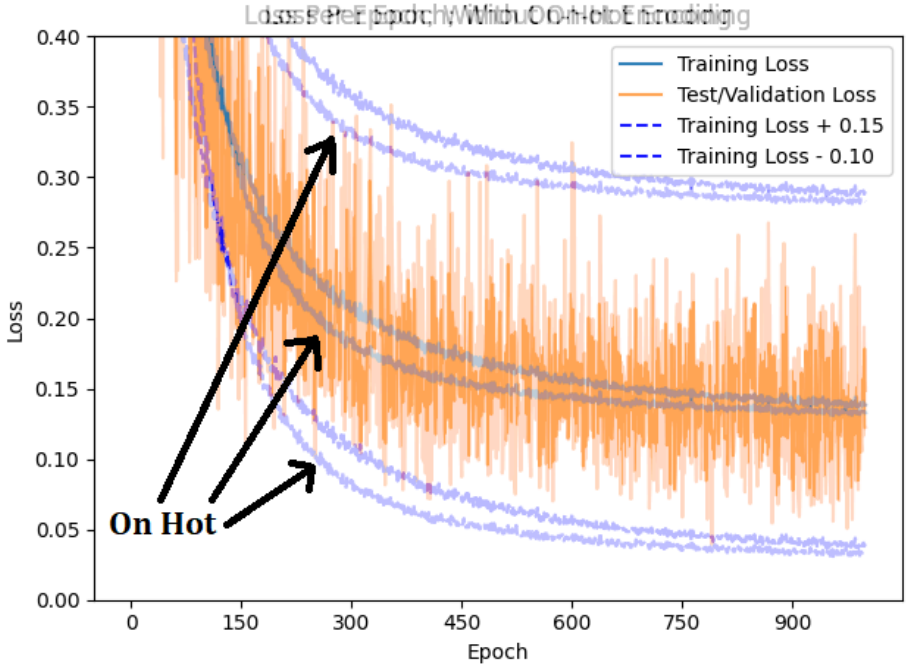
2.b. Build a multi-perceptron network that regresses the housing price (based on 20%, 80% split). Use the same number of features we did in the lecture, but this time also add on-hot encoding. Please plot the training and validation results and report the final accuracy and model complexity. Do you see the meaningful changes against 2.b. (20pt)

Model Width: 100 Neurons
Model Depth: 3 Hidden + 1 Output
Encoding: Binary Encoding AND On-Hot Encoding
Training Time: 1000 Epochs





Report the final accuracy and model complexity:
In an effort to better understand the difference between the plot in 2.a) and 2.b) I overlaid the 2 plots on top of each other and changed to opacity for the upper plot. I added 3 arrows pointing to which of the training loss lines point to the loss recorded in 2.b), that is which line was achieved using on hot encoding.

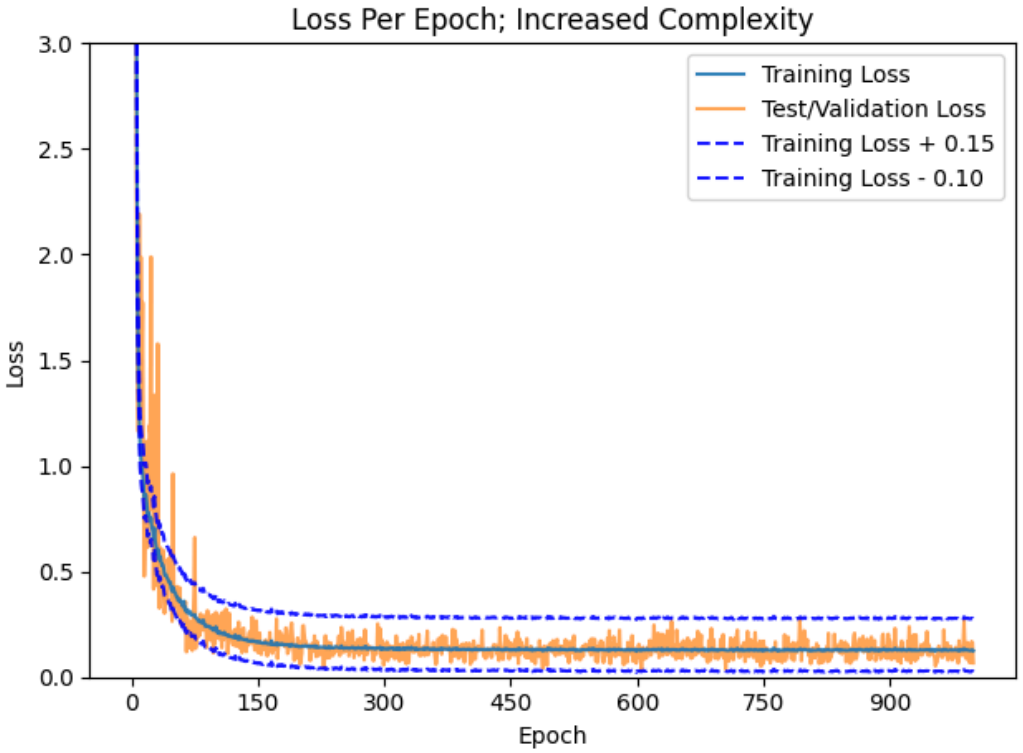


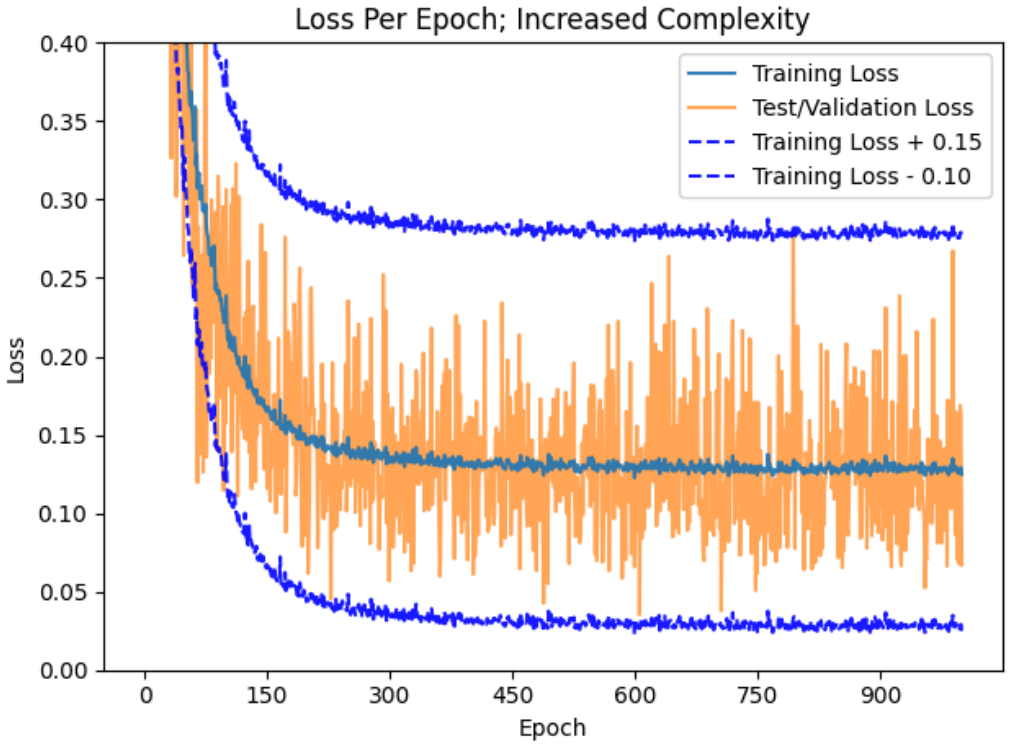
From the above image, I can conclude that everything I had to say regarding final accuracy and model complexity for this section is the same as was stated in the equivalent question for 2.a) with a single caveat.

Adding on hot encoding appears to have increased the learning rate (not the optimizer learning rate) for the dense layer.

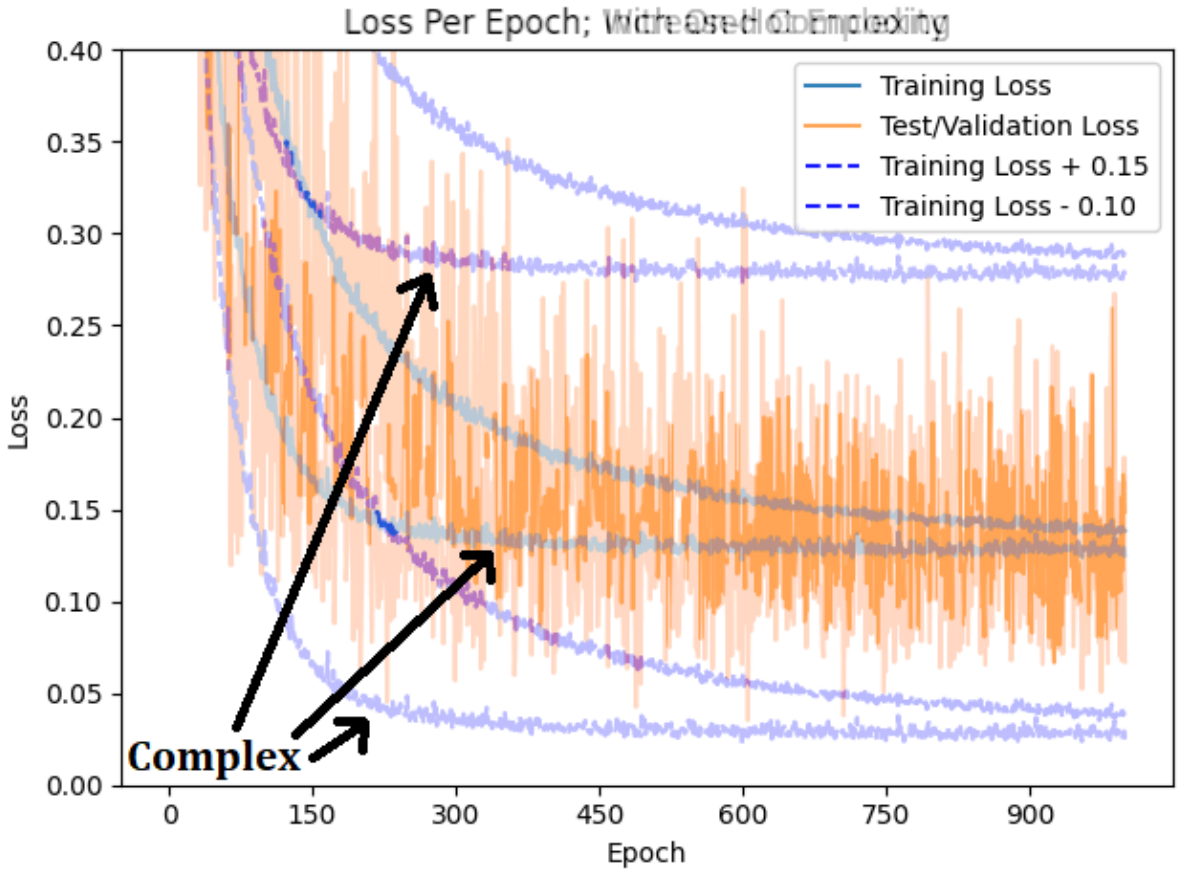
2.c increase the complexity of the network for problem 2. b and compare your results against 2.b. (10pt)

Model Width: 1000 Neurons
Model Depth: 6 Hidden + 1 Output
Encoding: Binary Encoding AND On-Hot Encoding
Training Time: 1000 Epochs





Compare your results against 2b:
Below I have overlaid my loss plots for this section over the prior section (2.b) and added arrows pointing to the training loss for this section.



Making the model more complex appears to have increased the learning rate (not the optimizer learning rate) for the dense layer without any noticeable side effects (no additional overfitting/underfitting).