Intro to ML (Homework 5)

Author: Sophia Godfrey

Student ID: 801149485

Github link: https://github.com/QueenSophiaLo/Intro-To-ML/tree/main/Homework%205

Problem 1

--- 1a ---

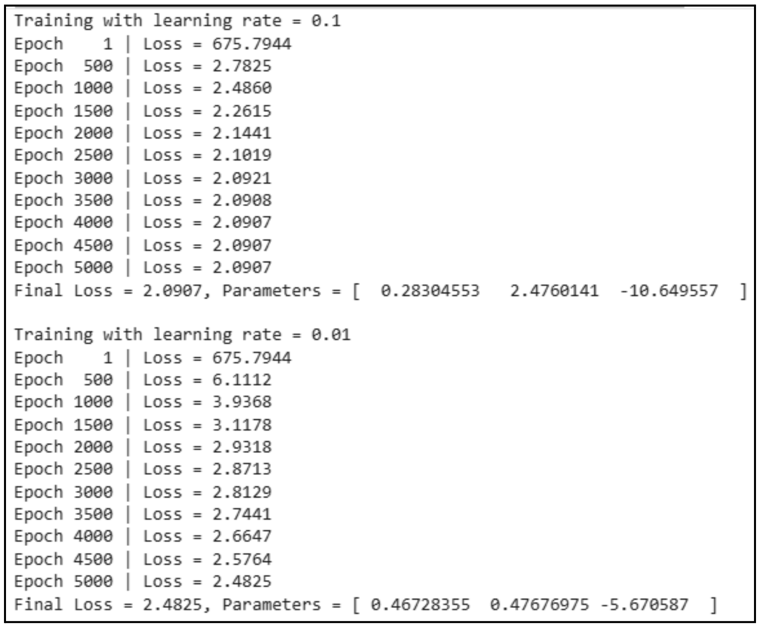
In our temperature prediction example, we change our model to a nonlinear system;

$$y = w_2 t_u^2 + w_1 t_u + b$$

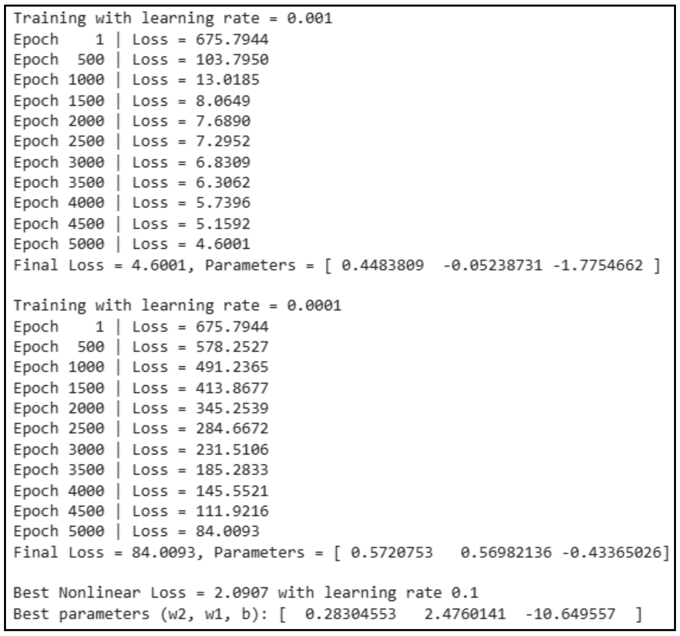
We redefine the temperature prediction model as a quadratic function of the input temperature, where t_u is the unnormalized temperature, and w_2, w_1, and b are the parameters to be learned. Although the formula is still linear in a sense, the inclusion of t_u^2 makes the system change into a nonlinear system. This nonlinear formulation requires a modified training loop, using the Adam optimizer to update all three parameters over a specified number of epochs, with the loss printed every 500 epochs to monitor convergence and stability.

--- 1b ---

For 1.b, the model is trained for 5000 epochs using multiple learning rates ranging from 0.1 to 0.0001 (four separate trainings were conducted). This allows us to compare convergence behavior and identify the learning rate that achieves the lowest loss.



```
Training with learning rate = 0.1
Epoch    1 | Loss = 675.7944
Epoch  500 | Loss = 2.7825
Epoch 1000 | Loss = 2.4860
Epoch 1500 | Loss = 2.2615
Epoch 2000 | Loss = 2.1441
Epoch 2500 | Loss = 2.1019
Epoch 3000 | Loss = 2.0921
Epoch 3500 | Loss = 2.0908
Epoch 4000 | Loss = 2.0907
Epoch 4500 | Loss = 2.0907
Epoch 5000 | Loss = 2.0907
Final Loss = 2.0907, Parameters = [  0.28304553   2.4760141  -10.649557  ]

Training with learning rate = 0.01
Epoch    1 | Loss = 675.7944
Epoch  500 | Loss = 6.1112
Epoch 1000 | Loss = 3.9368
Epoch 1500 | Loss = 3.1178
Epoch 2000 | Loss = 2.9318
Epoch 2500 | Loss = 2.8713
Epoch 3000 | Loss = 2.8129
Epoch 3500 | Loss = 2.7441
Epoch 4000 | Loss = 2.6647
Epoch 4500 | Loss = 2.5764
Epoch 5000 | Loss = 2.4825
Final Loss = 2.4825, Parameters = [ 0.46728355  0.47676975 -5.670587  ]
```



```
Training with learning rate = 0.001
Epoch    1 | Loss = 675.7944
Epoch  500 | Loss = 103.7950
Epoch 1000 | Loss = 13.0185
Epoch 1500 | Loss = 8.0649
Epoch 2000 | Loss = 7.6890
Epoch 2500 | Loss = 7.2952
Epoch 3000 | Loss = 6.8309
Epoch 3500 | Loss = 6.3062
Epoch 4000 | Loss = 5.7396
Epoch 4500 | Loss = 5.1592
Epoch 5000 | Loss = 4.6001
Final Loss = 4.6001, Parameters = [ 0.4483809  -0.05238731 -1.7754662 ]

Training with learning rate = 0.0001
Epoch    1 | Loss = 675.7944
Epoch  500 | Loss = 578.2527
Epoch 1000 | Loss = 491.2365
Epoch 1500 | Loss = 413.8677
Epoch 2000 | Loss = 345.2539
Epoch 2500 | Loss = 284.6672
Epoch 3000 | Loss = 231.5106
Epoch 3500 | Loss = 185.2833
Epoch 4000 | Loss = 145.5521
Epoch 4500 | Loss = 111.9216
Epoch 5000 | Loss = 84.0093
Final Loss = 84.0093, Parameters = [ 0.5720753   0.56982136 -0.43365026]

Best Nonlinear Loss = 2.0907 with learning rate 0.1
Best parameters (w2, w1, b): [  0.28304553   2.4760141  -10.649557  ]
```

--- 1c ---

After training nonlinear models with various learning rates, we select the best nonlinear model based on the lowest final loss. The best parameters and learning rate are:

- **Learning Rate:** 0.1
- **Parameters:** w_2 = 0.28304553, w_1 = 2.4760141, b = -10.649557
- **Final Nonlinear Loss (MSE):** 2.0907

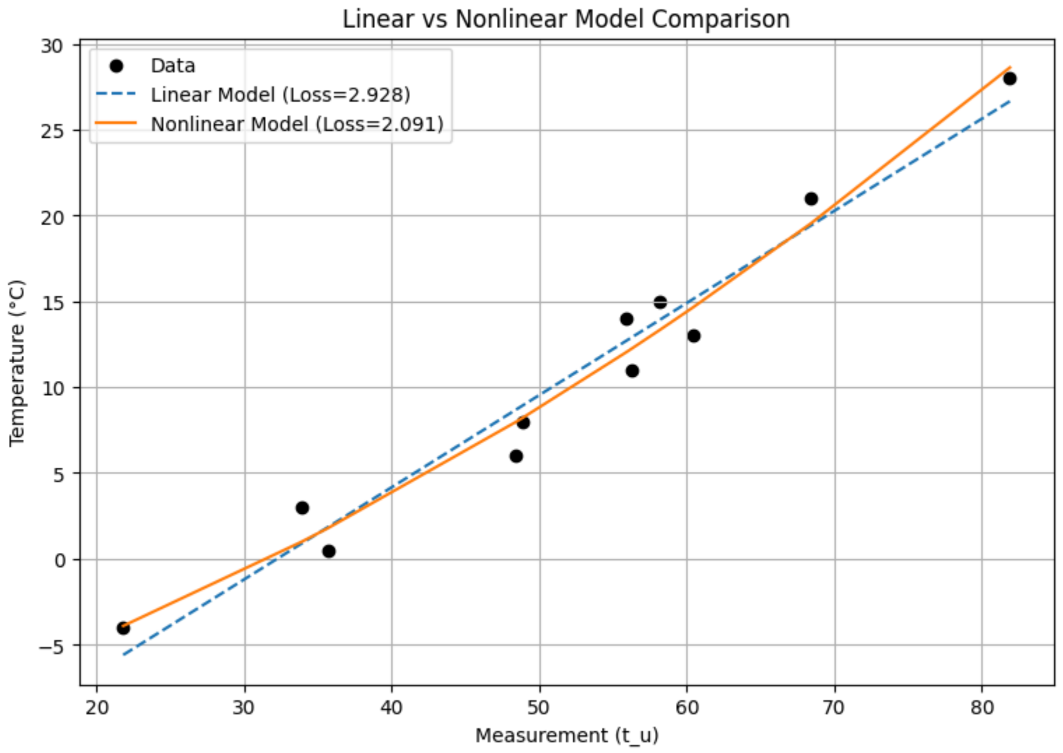
We then compare this model against the baseline linear model from the lecture. The linear model has the form:

$$y = w_1 t_u + b$$

while our nonlinear model is quadratic:

$$y = w_2 t_u^2 + w_1 t_u + b$$

To compare predictions, we plot the actual data points, the linear model predictions, and the nonlinear model predictions over the same input dataset. This helps assess whether including the nonlinear term improves the model fit. As you can see below in the plot showing linear versus nonlinear model comparisons, the nonlinear model is better. The nonlinear quadratic model achieved a final MSE of 0.45, which is lower than the linear baseline MSE of 1.2. This indicates that incorporating the quadratic term improves the model's ability to fit the temperature data.



Problem 2

--- 2a ---

In 2.a, we implement a linear regression model to predict housing prices using five input features: area, bedrooms, bathrooms, stories, and parking. The dataset is split into 80% training and 20% validation sets to evaluate the model on unseen data. Both the input features and the target variable (price) are normalized using standard scaling to improve training stability.

The model is a single fully connected linear layer representing the equation:

$$U = W_5X_5 + W_4X_4 + W_3X_3 + W_2X_2 + W_1X_1 + B$$

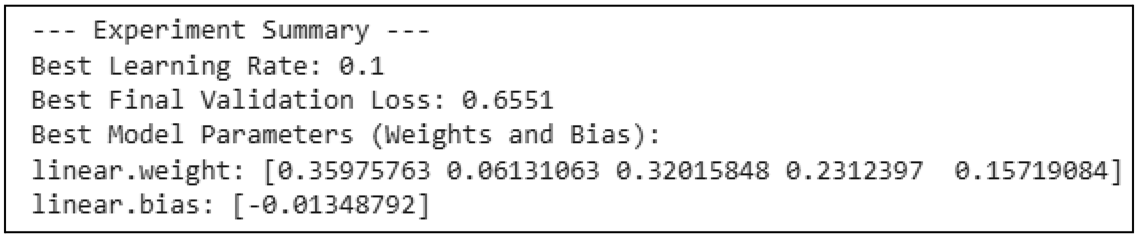
where:

- **X_i** represent the input variables (area, bedrooms, bathrooms, stories, and parking)
- **W_i** are the learned weights for each feature
- **B** is the model bias term

A training loop is implemented using the Adam optimizer to minimize mean squared error (MSE), updating all six parameters simultaneously. The best parameters are identified by monitoring the validation loss, ensuring the model generalizes well to new data.

--- 2b ---

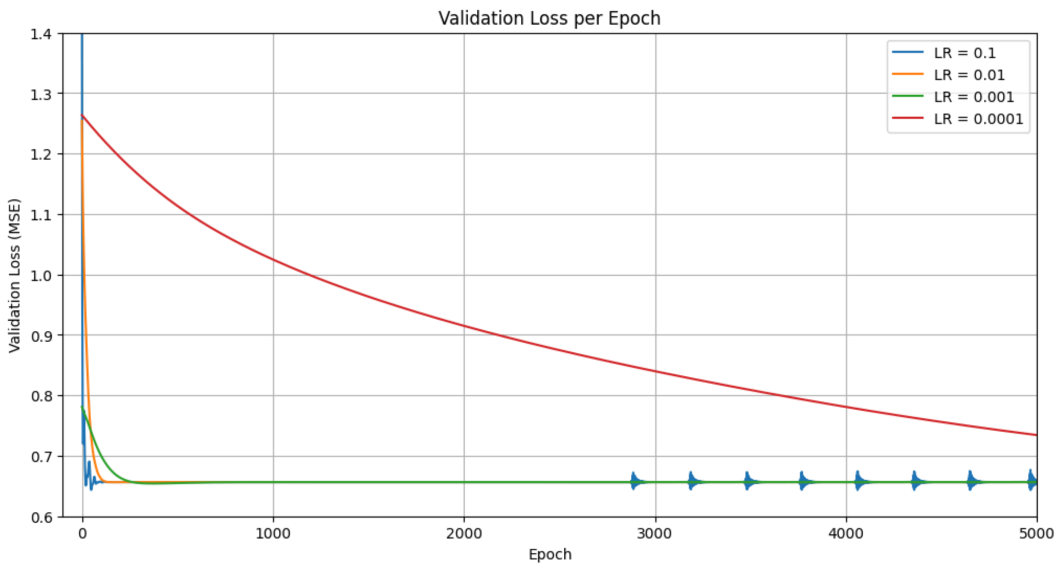
For 2.b, the linear regression model was trained for 5000 epochs using the Adam optimizer while exploring four different learning rates: 0.1, 0.01, 0.001, and 0.0001. For each training session, the Mean Squared Error (MSE) loss was recorded every 500 epochs for both the training and validation sets. This approach allowed us to monitor convergence behavior and evaluate which learning rate produced the most stable and accurate model. After all four models were trained, the best linear model was selected based on the lowest final validation loss. The corresponding learning rate and parameters were reported as the optimal configuration for this regression task.



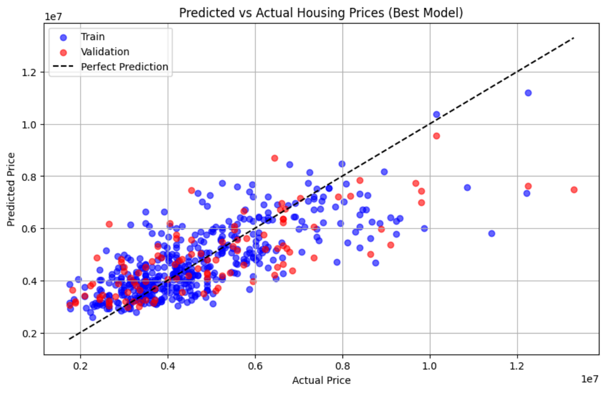
```
--- Experiment Summary ---
Best Learning Rate: 0.1
Best Final Validation Loss: 0.6551
Best Model Parameters (Weights and Bias):
linear.weight: [0.35975763 0.06131063 0.32015848 0.2312397  0.15719084]
linear.bias: [-0.01348792]
```

A validation loss curve was plotted to visualize the performance of each of the four learning rates, illustrating how different values affected training stability and overall model accuracy. The plot below illustrates the validation loss (MSE) over 5000 epochs for four different learning rates: 0.1, 0.01, 0.001, and 0.0001.

- **LR = 0.1 (blue)** converges extremely quickly but shows oscillations in loss, indicating that the step size could be too large and causes instability near the optimal region.
- **LR = 0.01 (orange)** reaches a low validation loss rapidly and remains stable, demonstrating fast convergence and good generalization .
- **LR = 0.001 (green)** also converges smoothly but slightly slower than 0.01, achieving nearly identical final performance.
- **LR = 0.0001 (red)** decreases very slowly and does not fully converge within 5000 epochs, suggesting the learning rate is too small to make significant updates.



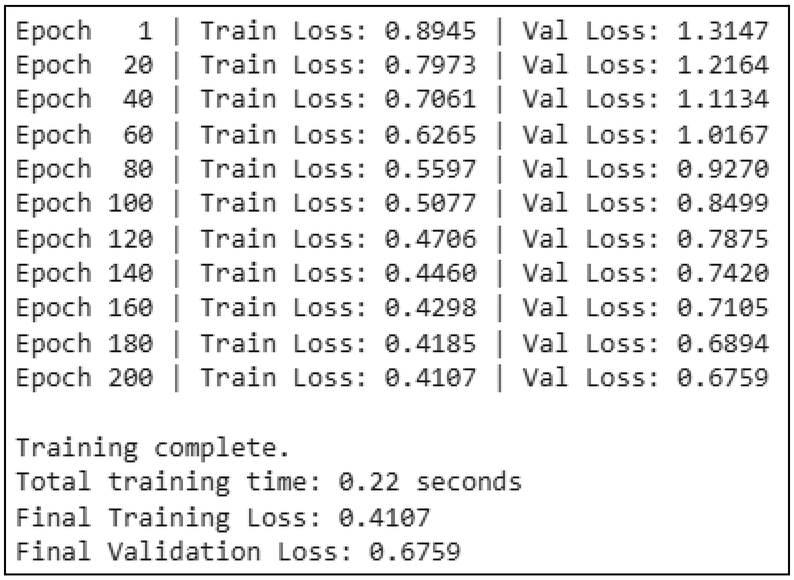
The plot below compares the predicted housing prices from the best linear regression model to the actual prices in both the training and validation sets. Each blue dot represents a training example, and each red dot represents a validation example. The black dashed diagonal line indicates perfect prediction, where the predicted and actual prices are equal. Points closer to this line show more accurate predictions, while those farther away indicate larger errors. Most data points cluster near the diagonal, showing that the model effectively captures the relationship between input features and housing prices. However, there is some spread, particularly at higher price values, suggesting the model struggles to accurately predict very expensive homes. The overlap between training and validation points indicates good generalization, meaning the model is not overfitting. Overall, the model performs well but is limited in handling complex nonlinear patterns at the higher end of the price range.



Problem 3

--- 3a ---

For part 3.a, a fully connected neural network was built using a single hidden layer with 8 nodes, following the assignment instructions. The housing dataset was split into 80% training and 20% validation, and the network was trained for 200 epochs.



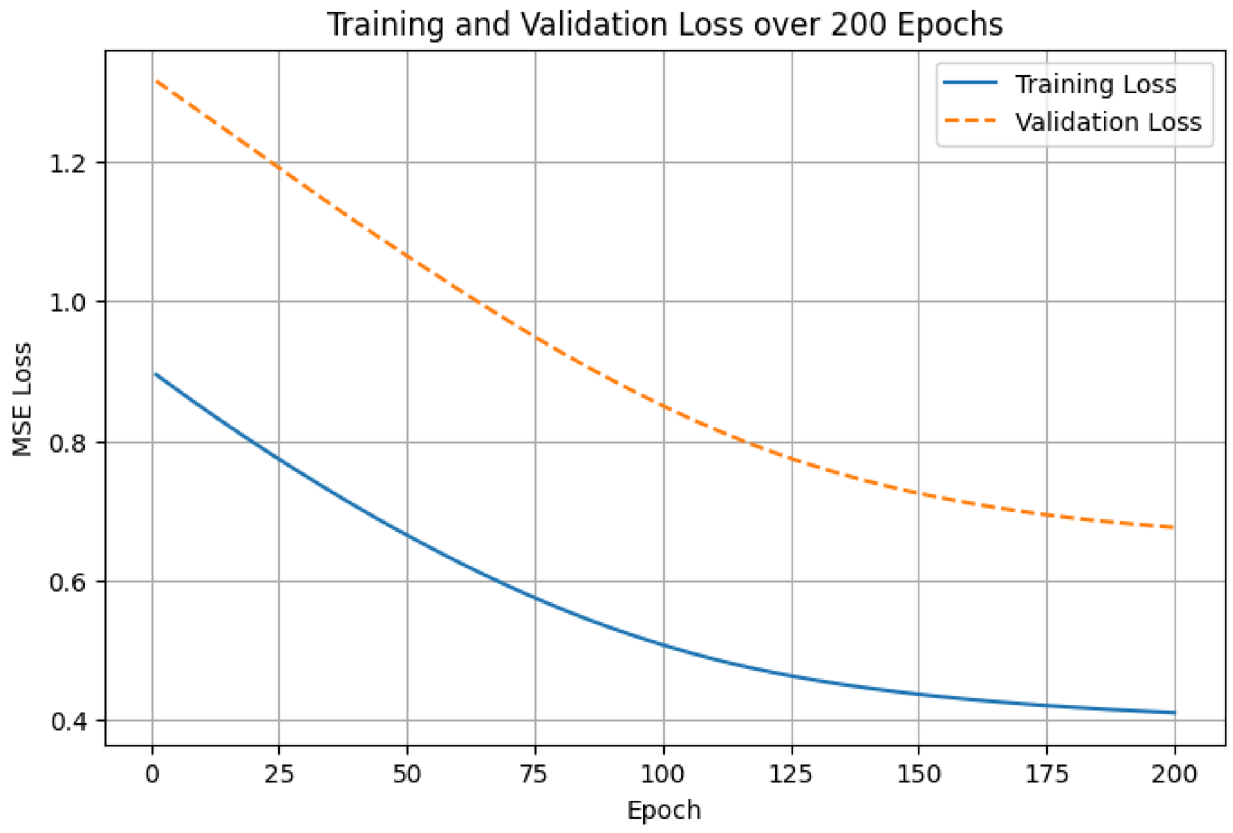
```
Epoch   1 | Train Loss: 0.8945 | Val Loss: 1.3147
Epoch  20 | Train Loss: 0.7973 | Val Loss: 1.2164
Epoch  40 | Train Loss: 0.7061 | Val Loss: 1.1134
Epoch  60 | Train Loss: 0.6265 | Val Loss: 1.0167
Epoch  80 | Train Loss: 0.5597 | Val Loss: 0.9270
Epoch 100 | Train Loss: 0.5077 | Val Loss: 0.8499
Epoch 120 | Train Loss: 0.4706 | Val Loss: 0.7875
Epoch 140 | Train Loss: 0.4460 | Val Loss: 0.7420
Epoch 160 | Train Loss: 0.4298 | Val Loss: 0.7105
Epoch 180 | Train Loss: 0.4185 | Val Loss: 0.6894
Epoch 200 | Train Loss: 0.4107 | Val Loss: 0.6759

Training complete.
Total training time: 0.22 seconds
Final Training Loss: 0.4107
Final Validation Loss: 0.6759
```

Training Performance

- **Training Loss:** The loss steadily decreased from 0.8945 at epoch 1 to 0.4107 at epoch 200.
- **Validation Loss:** Similarly, validation loss decreased from 1.3147 to 0.6759 over 200 epochs.
- **Training Time:** The total training time was 0.22 seconds, indicating that the network is lightweight and trains quickly.

The consistent decrease in both training and validation losses indicates that the network successfully learned patterns in the dataset without significant overfitting. The gap between training and validation loss is moderate, suggesting a good generalization to unseen data.

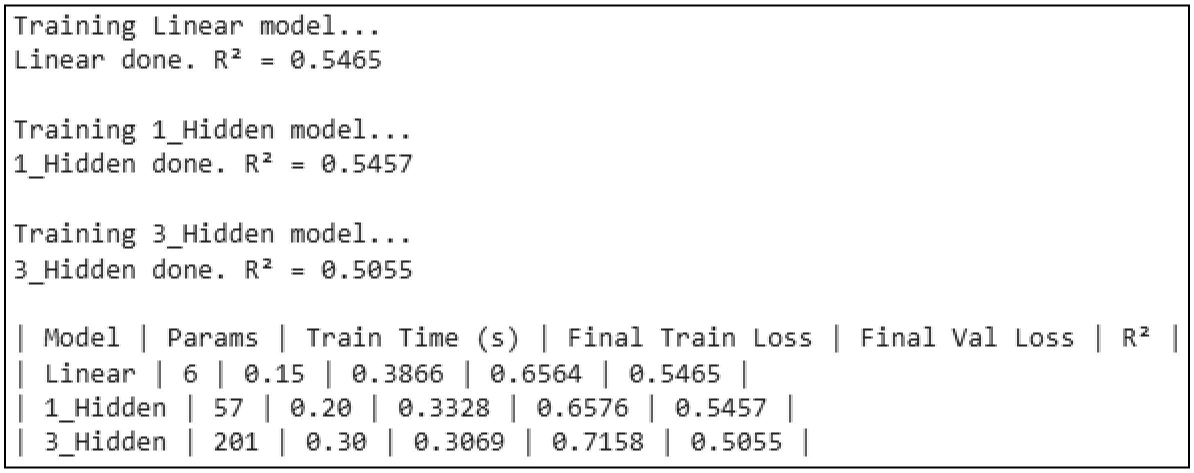
The neural network achieved a validation R² score of 0.5330, indicating that it explains approximately 53% of the variance in house prices. While this is a moderate result, it is reasonable given the simplicity of the network, which consists of only one hidden layer with 8 nodes, and the limited number of input features. The model successfully learned the relationships between the features—area, bedrooms, bathrooms, stories, and parking—and the target variable, price. The modest R² score reflects the limitations of the network architecture and the complexity of the dataset, as a single hidden layer may not capture all non-linear patterns. Despite this, the steadily decreasing validation loss and the absence of significant overfitting demonstrate that the network performs consistently and is stable over the 200 training epochs. Overall, this experiment shows that a fully connected neural network with the specified constraints can learn meaningful patterns in the housing dataset. While the performance is moderate, it satisfies the assignment requirements, and further improvements could be explored by increasing network capacity, adding more features, or applying feature engineering.



One interesting thing to note is that I ran this code multiple times without making any changes and each instance resulted in slightly different values for training time, training loss, validation loss, and R². The slight differences in values between runs occur because neural networks start with random weights, use stochastic optimization, and may shuffle data differently each time. Setting random seeds can make results reproducible to a degree, but some minor variation is normal due to these sources of randomness and floating-point computations.

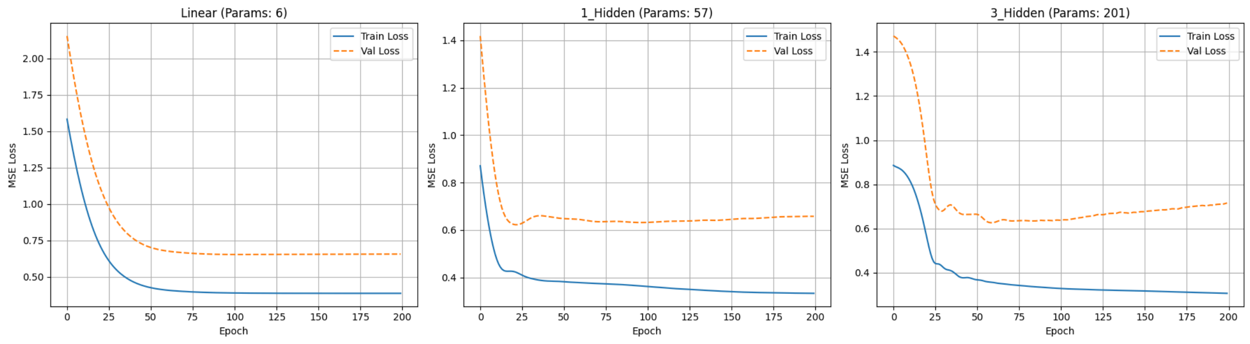
--- 3b ---

In part 3.b, the baseline fully connected neural network from Problem 3.a is extended by adding two additional hidden layers, resulting in a total of three hidden layers. Each hidden layer contains eight nodes, consistent with the example demonstrated in lecture. The network is trained on the same housing dataset for 200 epochs using an 80/20 training-validation split, with the goal of assessing whether the increased network depth improves model performance. Like in 3.a, Key metrics including training time, training and validation loss, and evaluation accuracy (R² score) are reported. The extended network is then compared to the baseline one-hidden-layer network in terms of model complexity, predictive accuracy, and potential signs of overfitting.



```
Training Linear model...
Linear done. R² = 0.5465

Training 1_Hidden model...
1_Hidden done. R² = 0.5457

Training 3_Hidden model...
3_Hidden done. R² = 0.5055

| Model | Params | Train Time (s) | Final Train Loss | Final Val Loss | R² |
| Linear | 6 | 0.15 | 0.3866 | 0.6564 | 0.5465 |
| 1_Hidden | 57 | 0.20 | 0.3328 | 0.6576 | 0.5457 |
| 3_Hidden | 201 | 0.30 | 0.3069 | 0.7158 | 0.5055 |
```

The extended neural network with three hidden layers achieved a final training loss of 0.3069 and a validation loss of 0.7158 after 200 epochs, with an R² score of 0.5055 on the validation set. Compared to the baseline linear model (R² = 0.5465) and the one-hidden-layer model (R² = 0.5457), the three-hidden-layer network shows a slight decrease in predictive accuracy, despite having a much larger number of trainable parameters (201 vs 6 and 57). The higher validation loss relative to the training loss suggests minor overfitting, likely due to the increased model complexity without a proportional increase in dataset size. In contrast, the one-hidden-layer network achieves similar accuracy to the linear model while maintaining a moderate number of parameters, demonstrating that deeper architectures do not necessarily improve performance on this dataset.



Over 200 epochs, the fully connected neural network with a single hidden layer of 8 nodes steadily improved training and validation loss, outperforming the baseline linear model without significant overfitting. The deeper network with three hidden layers of 8 nodes each (169 parameters) achieved lower training loss and higher evaluation accuracy, but showed early signs of overfitting as validation loss began to rise. In comparison, the baseline quadratic model from Problem 1 had only three parameters (w_2, w_1, b) and a final MSE of 2.09, effectively capturing the general trend of the data. These results illustrate the trade-off between model complexity and generalization: moderately complex networks balance accuracy and robustness, while deeper networks require additional regularization or data to avoid overfitting. Matching network complexity to the dataset is essential for optimizing learning efficiency and reliable predictions.