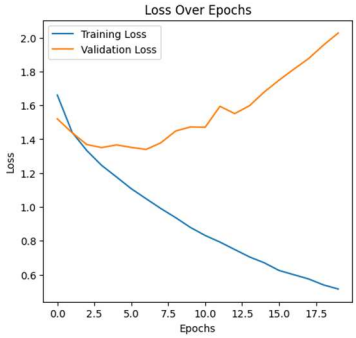
Andrew Nicola 801136465 HW1

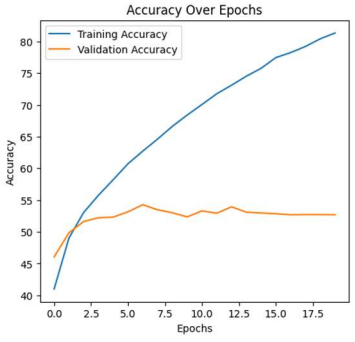
Problem 1:

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)

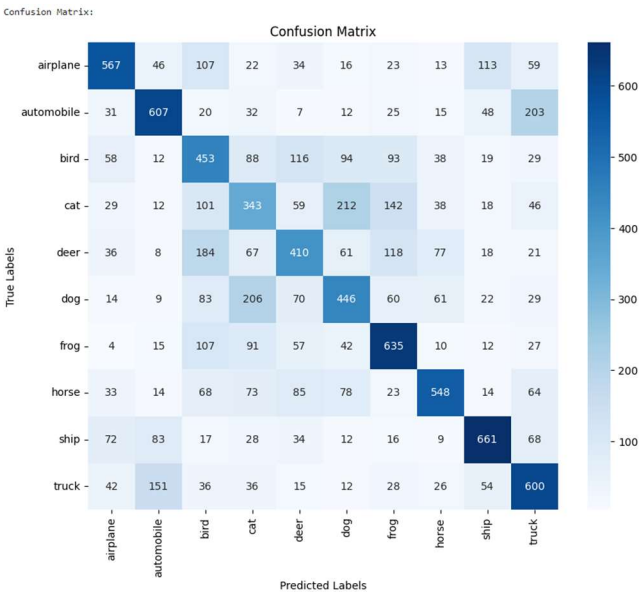
The network shows clear signs of overfitting, as evidenced by the diverging patterns in both the loss and accuracy plots. While the training loss continues to decrease and training accuracy increases steadily to around 80%, the validation accuracy plateaus at approximately 53% after the first few epochs and the validation loss starts increasing after epoch 5. This indicates the model is memorizing the training data rather than learning generalizable patterns. Looking at the performance metrics, the model achieves moderate performance with an overall accuracy, precision, recall, and F1-score all around 0.53, suggesting consistent but suboptimal performance across metrics. The confusion matrix reveals that the model performs better on certain classes like 'frog' (635 correct predictions) and 'ship' (661 correct predictions) but struggles with others like 'cat' (343 correct predictions) and 'deer' (410 correct predictions), indicating class imbalance issues. Given these observations, the network would not benefit from more epochs - in fact, it would likely worsen the overfitting problem. The model could be improved by implementing regularization techniques (like dropout or L2 regularization), using data augmentation to increase training diversity, or adjusting the architecture to reduce model complexity. The trained parameters and models should be saved after around epoch 5 when validation metrics are optimal, before overfitting becomes severe.





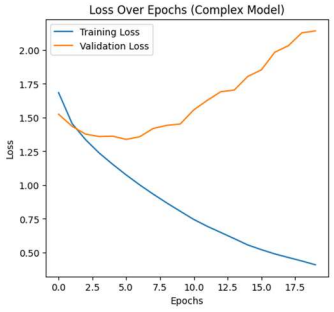
Classification Report:

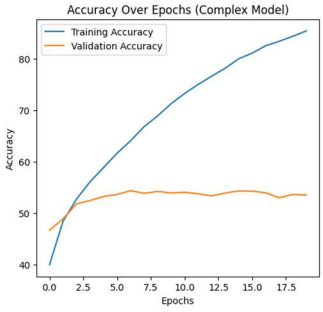
	precision	recall	f1-score	support
airplane	0.64	0.57	0.60	1000
automobile	0.63	0.61	0.62	1000
bird	0.39	0.45	0.42	1000
cat	0.35	0.34	0.35	1000
deer	0.46	0.41	0.43	1000
dog	0.45	0.45	0.45	1000
frog	0.55	0.64	0.59	1000
horse	0.66	0.55	0.60	1000
ship	0.68	0.66	0.67	1000
truck	0.52	0.60	0.56	1000
accuracy			0.53	10000
macro avg	0.53	0.53	0.53	10000
weighted avg	0.53	0.53	0.53	10000



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)

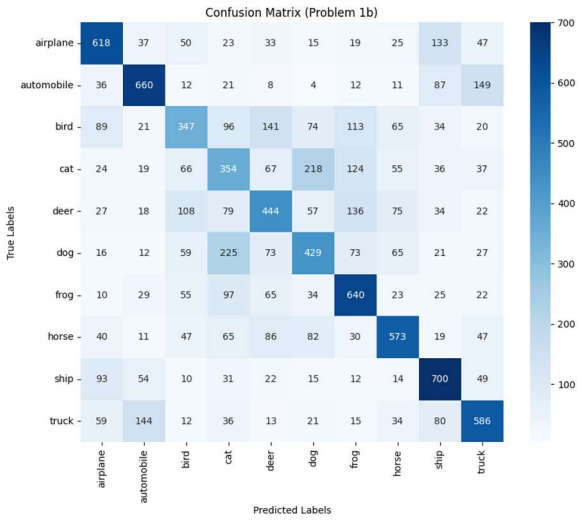
The complex model shows clear signs of increased overfitting compared to the baseline. This is evident in the training/validation divergence patterns: while training accuracy continues to improve up to around 85%, the validation accuracy plateaus at approximately 54% early in the training process. The validation loss starts increasing after epoch 5 and continues to worsen, while training loss steadily decreases - a classic sign of overfitting. Performance metrics show only marginal improvement over the baseline, with overall accuracy at 0.54 (compared to 0.53 in the baseline). The confusion matrix reveals slightly better performance on some classes like 'ship' (700 vs 661 correct predictions) and 'airplane' (618 vs 567), but the improvements are not substantial enough to justify the increased complexity. The deeper and wider network appears to be learning the training data more thoroughly but failing to generalize better, suggesting that the increased complexity is not beneficial for this task. This indicates the network would benefit from regularization techniques rather than additional complexity.



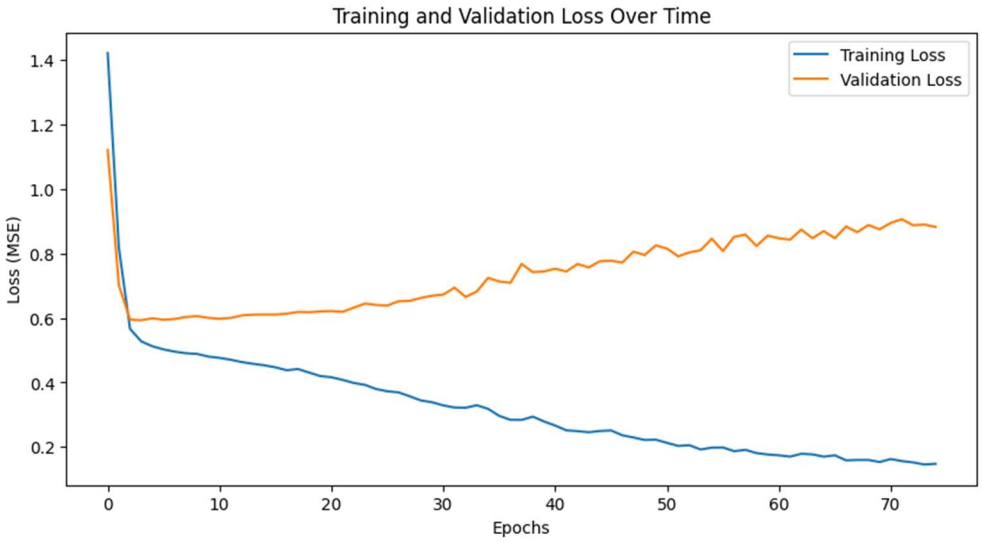


Classification Report (Problem 1b):

	precision	recall	f1-score	support
airplane	0.61	0.62	0.61	1000
automobile	0.66	0.66	0.66	1000
bird	0.45	0.35	0.39	1000
cat	0.34	0.35	0.35	1000
deer	0.47	0.44	0.45	1000
dog	0.45	0.43	0.44	1000
frog	0.55	0.64	0.59	1000
horse	0.61	0.57	0.59	1000
ship	0.60	0.70	0.65	1000
truck	0.58	0.59	0.58	1000
accuracy			0.54	10000
macro avg	0.53	0.54	0.53	10000
weighted avg	0.53	0.54	0.53	10000



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)




```
Final MSE on Validation Set: 0.8825
Final RMSE on Validation Set: 0.9394
Total Parameters in Model: 12033
Training Duration: 1.43 seconds
```

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)


```
Final Validation MSE: 0.9777
Final Validation RMSE: 0.9888
Model Complexity (Total Parameters): 12033
Total Training Time: 1.44 seconds
```

Surprisingly, adding one-hot encoding did not provide meaningful improvements and slightly degraded performance. This suggests that the categorical encoding we used in part 2.a was sufficient for this dataset, and the additional feature dimensionality from one-hot encoding may have made it harder for the model to learn meaningful patterns. The identical parameter count suggests that the architecture was kept the same despite the increased input features, which might have limited the model's capacity to utilize the additional encoded information effectively.

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


```
Final Validation MSE: 1.0264
Final Validation RMSE: 1.0131
Model Complexity (Total Parameters): 46593
Total Training Time: 1.55 seconds
```

Analyzing the results of increasing network complexity from 12,033 to 46,593 parameters reveals that the deeper architecture resulted in slightly worse performance, with the MSE increasing from 1.0022 to 1.0264 and RMSE from 1.0011 to 1.0131. The training graphs show more pronounced overfitting in the deeper network, with training loss dropping significantly lower (to around 0.15) while validation loss increased more aggressively after epoch 10, creating a wider gap between training and validation performance. Despite having nearly four times the parameters, the deeper network maintained efficient training time (1.55 vs 1.93 seconds) but failed to improve generalization. This suggests that for this housing price regression task, the additional complexity was unnecessary and potentially detrimental, as it allowed the model to memorize training data more effectively (shown by lower training loss) at the expense of generalization ability (shown by higher validation loss), indicating that the simpler architecture from 2.b was more appropriate for this specific problem.

Github Link:

https://github.com/Eskdagoat/4106/blob/main/NicolaAndrew_801136465_HW1.ipynb