Report of ECGR 5106 Homework 1

Name: Liyuan Zhang
Student ID: 801432783
GitHub Link: https://github.com/LeeYuuuan/ECGR-5106-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.

1a. Training a model from scratch.

I designed a simple MLP structure with the following structure:

Input layer: 3072 dimensions, corresponding to the flattened input data (32×32×3 for CIFAR-10 images).

Hidden layer: 64 units.

Output layer: 10 units, representing the 10 classes.

After 20 epochs training, the training loss and test loss continued to decrease, which means we need more training.

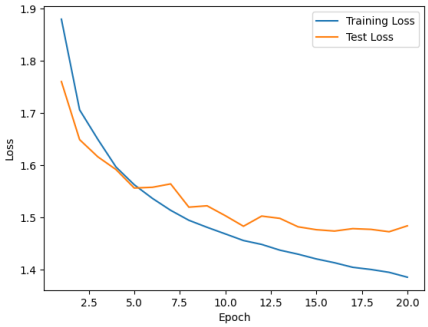


Figure 1 loss curve after 20 epochs training

Thus, I trained this model for additional 30 epochs, as shown below.

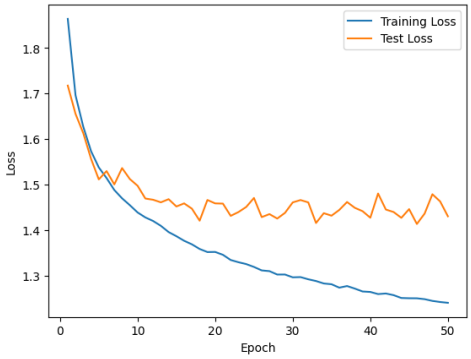


Figure 2 loss curve after training 50 epochs

After 50 epochs training, the test loss no longer decreased, which suggests that the model is well trained. However, the training loss still decreasing, which means a little over fitting.

Furthermore, the accuracy curve and other evaluation metrics are shown below.

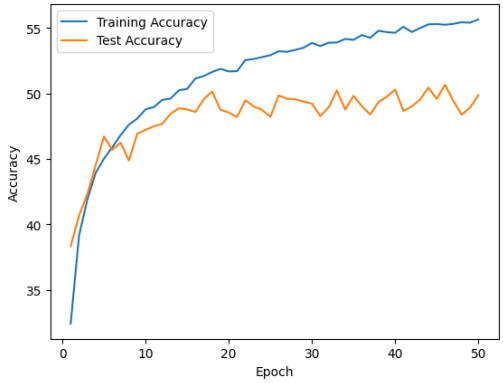
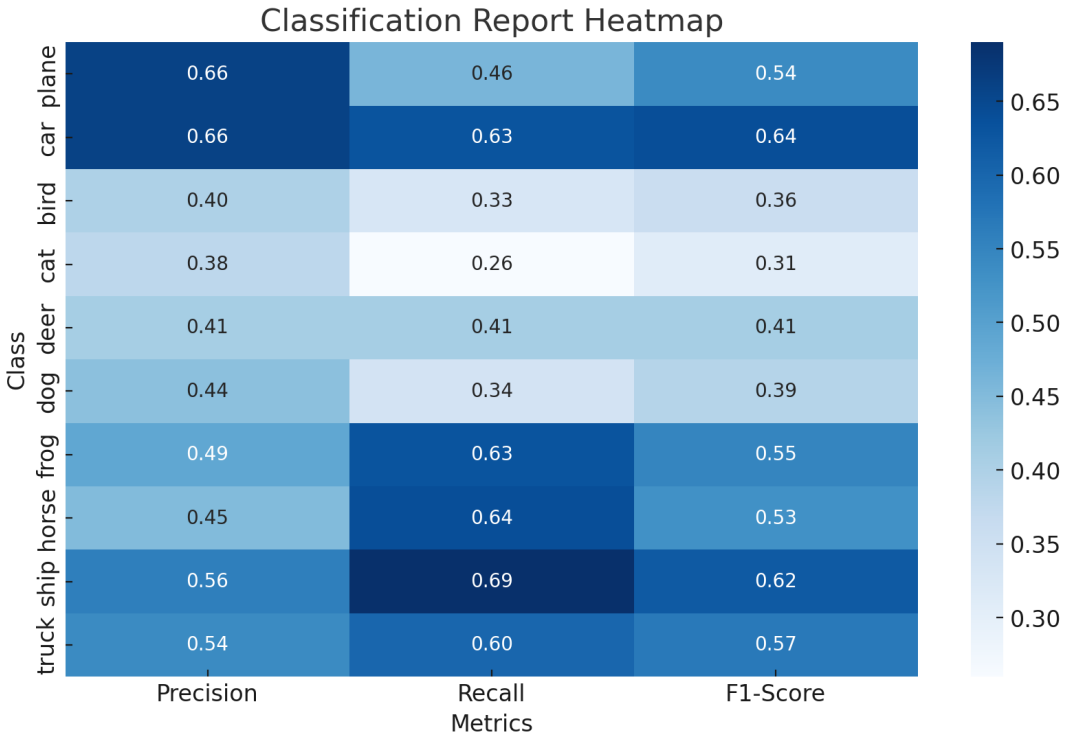


Figure 3 Accuracy Curve after 50 epochs training

Precision, recall, f1-score for every classes.



And Confusion Matrix.

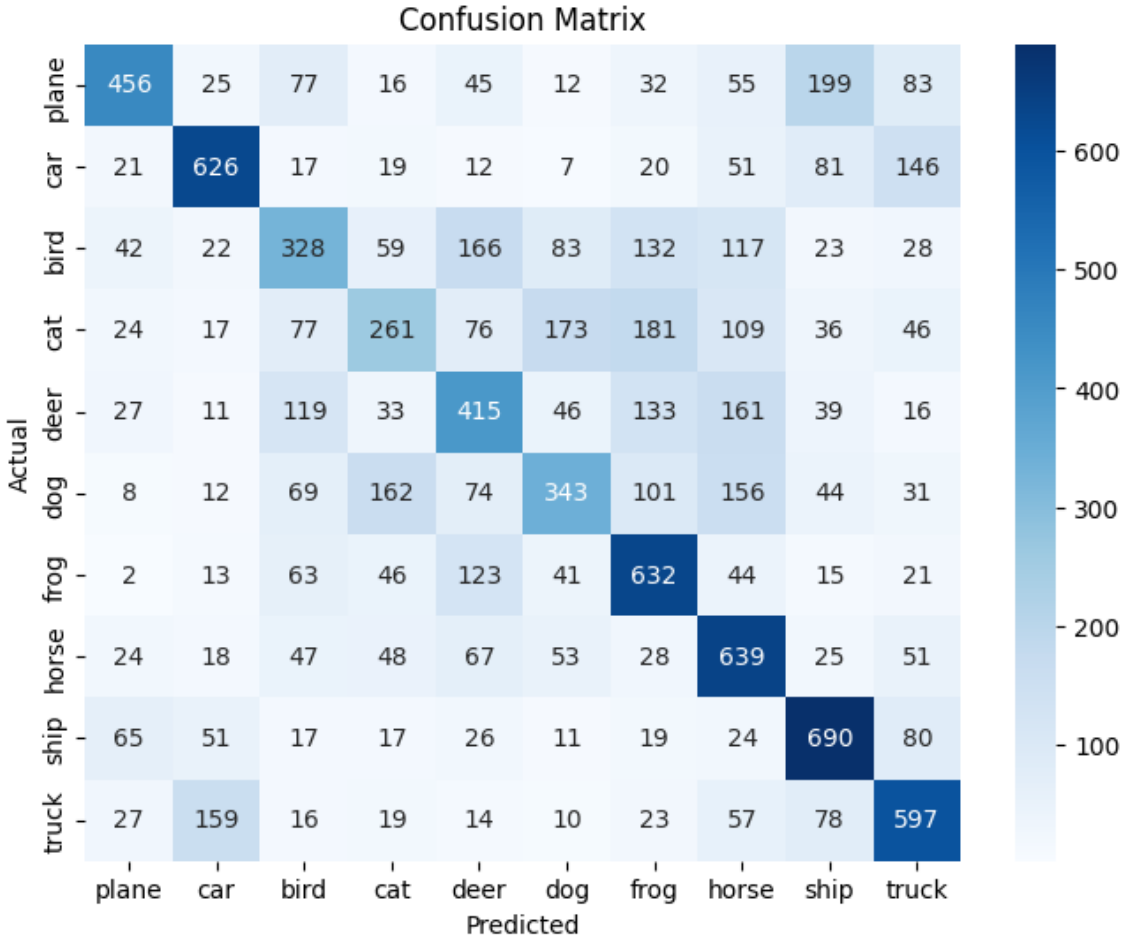


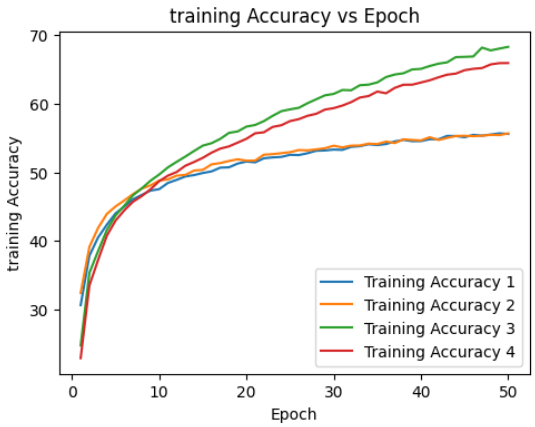
Figure 4 Confusion Matrix

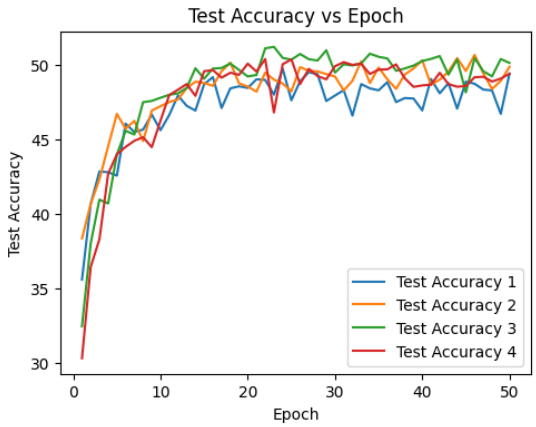
1b. Explore the complexity of the network by increasing its width and depth.

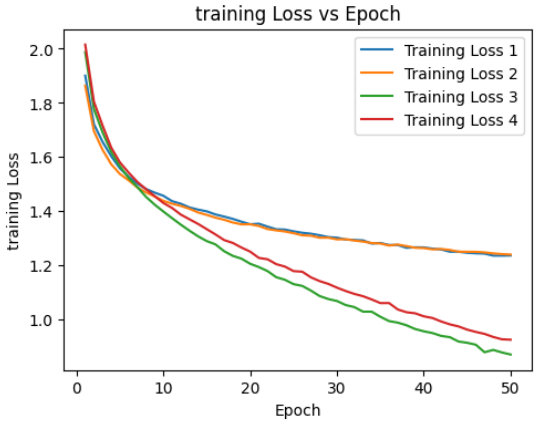
For this problem, I tried 4 different models with the different number of hidden layer and their neurons, as shown below.

Model	Hidden Layer Structure
Model 1	[64, 64]
Model 2	[64, 64, 64]
Model 3	[256, 256, 128, 128, 64]
Model 4	[256, 256, 128, 128, 128, 64]

And then got the Loss and Accureacy results.







Conclusion

From the loss and accuracy curve we can see that as the model complexity increases, the training loss decreases further. However, the test loss starts to increase, indicating that the model is starting overfitting.

Therefore, selecting a model with an appropriate complexity for different problems is crucial. Apparently, for this problem, we could use the simple one to fit the data.

Additionally, saving the model during training is essential to identify the optimal epoch after training.

Problem 2: Housing price regression.

2a. Build a multi-perceptron network that regresses the housing price.

2b. Using One-hot encoding to build this model

Feature Selection

For this problem, we first need to determine the shape and type of the input data and preprocess the input data.

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes	furnished
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no	furnished
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes	semi-furnished
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes	furnished
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no	furnished
...	...	...	...	...	...	...	...	...	...	...	...	...	...
540	1820000	3000	2	1	1	yes	no	yes	no	no	2	no	unfurnished
541	1767150	2400	3	1	1	no	no	no	no	no	0	no	semi-furnished
542	1750000	3620	2	1	1	yes	no	no	no	no	0	no	unfurnished
543	1750000	2910	3	1	1	no	no	no	no	no	0	no	furnished
544	1750000	3850	3	1	2	yes	no	no	no	no	0	no	unfurnished

545 rows × 13 columns

This dataset includes several features, which are:

(Float) number: *'area '*

(Int) number: *'bedrooms', 'bathrooms', 'stories', 'parking'*

(Bool, or categories): *'mainroad', 'guestroom', 'basement', 'hotwaterheating',*

'airconditioning', 'prefarea', 'furnishingstatus'

I use all of them as input feature. And the output feature will be the price of the house.

Data preprocessing

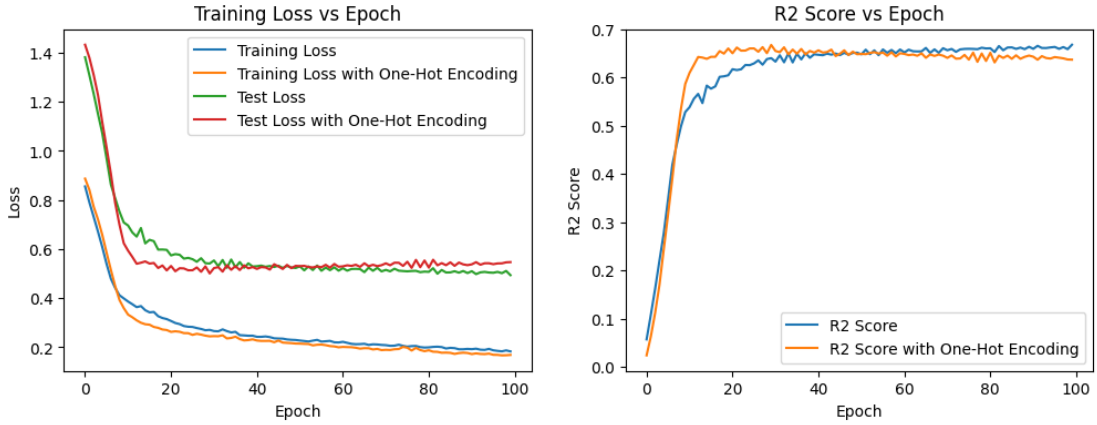
1. While the **categories** data is not a number, we need to first convert that to be a number.
2. As the **price** and **area** are so much bigger Since the numerical values of features such as price and area are significantly large (or, more precisely, these features have high variance compared to other features), it is necessary to scale them so as to ensure that all features are on a similar scale.
3. **For Problem 2b.** We need one more step to convert these **categories** features into one-hot encoding.

Model Structure

I designed a MLP with the dimension of input layer is 12 and 20, matching the dimensionality of the input features both with and without one-hot encoding respectively. And the hidden layer consists of 32 neurons, while the output layer has 1 neuron since this is a regression problem.

Training and Evaluation

After 100 epochs training, according to the loss curve, as shown below, we can we can conclude that the model has been well trained since the test loss would never decrease more.



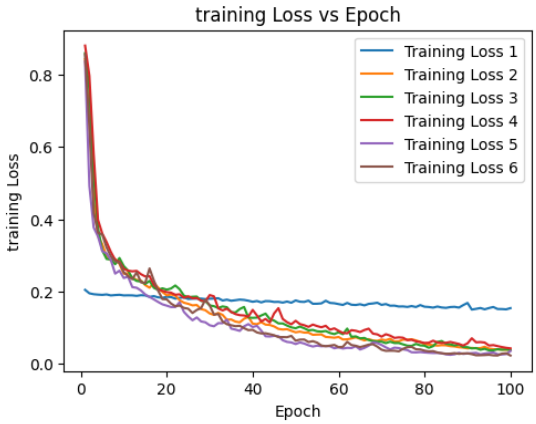
From the results we could see, for this problem, apart from the fact that adding one-hot encoding leads to a faster initial decrease in training loss, it does not seem to have a significant impact on the final results. This may be because both encoding methods contain nearly the same amount of information, as well as the task itself is relatively simple.

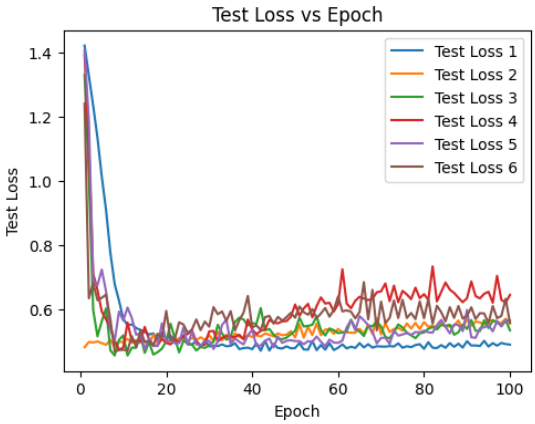
2a. increase the complexity of the network.

I selected five more complex models with increasing depth and width to analyze the impact of model complexity on performance. Their structures are as follows:

Model 1: [128, 128, 128, 64]
Model 2: [128, 128, 128, 128, 64]
Model 3: [128, 128, 128, 128, 128, 64]
Model 4: [256, 256, 256, 256, 128, 64]
Model 5: [256, 256, 256, 256, 256, 128, 64]
Each model consists of multiple hidden layers, where the numbers represent the number of neurons in each layer.

Results & Conclusions





For training loss

All models show a steady decrease in training loss as the number of epochs increases. Models with higher complexity (e.g., more hidden layers or neurons) tend to achieve a lower final training loss, indicating their ability to better fit the training data.

For test loss

Initially, all models experience a rapid decline in test loss. However, Models with higher complexity shows more fluctuations during later epochs, possibly suggesting overfitting, while simpler models display better stability and generalization.