Principle of Artificial Intelligence HW2:
0811562 何祁恩 0810761 羅雯瀠 (No information for the third person … 3 email without reply)

Part 1&2: A simple Neuron Network Implementation &Convolutional Neural Network (CNN) Implementation

1. Model Architecture

Part1

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 30, 30, 32)	896
conv2d_9 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 64)	0
flatten_4 (Flatten)	(None, 12544)	0
dense_8 (Dense)	(None, 128)	1605760
dense_9 (Dense)	(None, 10)	1290

Total params: 1626442 (6.20 MB)
Trainable params: 1626442 (6.20 MB)
Non-trainable params: 0 (0.00 Byte)

Part2

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 32)	18464
conv2d_3 (Conv2D)	(None, 10, 10, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204928
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290

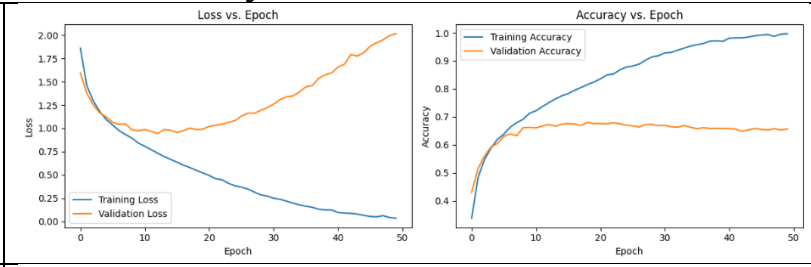
Total params: 262570 (1.00 MB)
Trainable params: 262570 (1.00 MB)
Non-trainable params: 0 (0.00 Byte)

In the model of Part 1, we implemented a very simple structure, utilizing only Conv2D and Dense layers. In the model of Part 2, we increased the depth of the model and incorporated dropout more frequently to prevent overfitting. We anticipate that this model will be more precise than the Part 1 model.

2. Training Loss & Accuracy

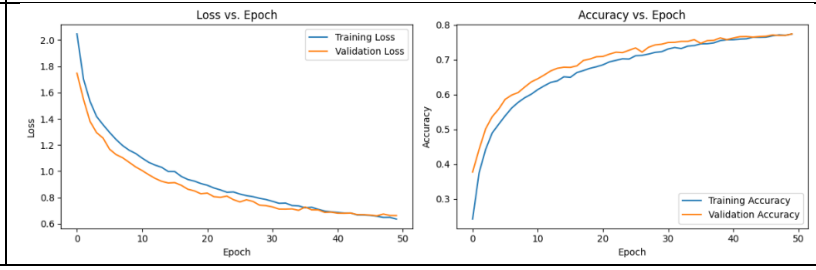
Part1

Test Accuracy: 65.68%



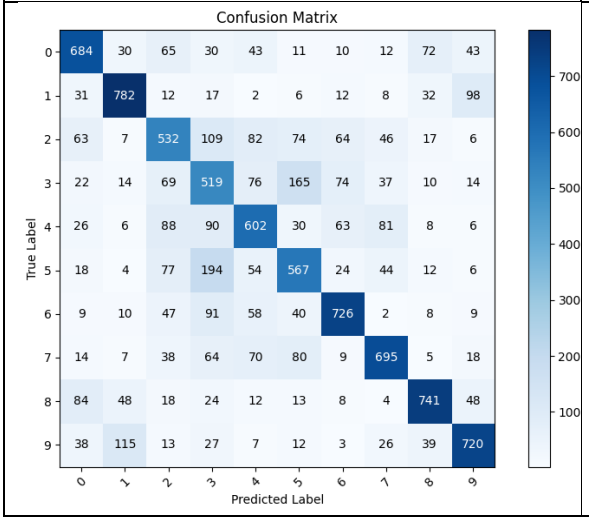
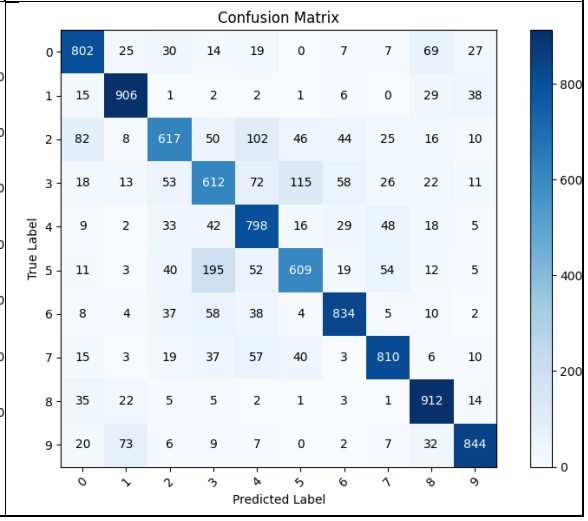
Part2

Test Accuracy: 77.44%



From the epoch-based plots of loss and accuracy above, we can observe that both training loss and training accuracy exhibit improvement with each epoch for both models. However, in the Part 1 model, there is a big gap between validation accuracy and training accuracy, accompanied by an increase in validation loss. This suggests that the model may be too simplistic, leading to overfitting under these training parameters. On the other hand, in the Part 2 model, validation accuracy closely tracks training accuracy, indicating a higher potential for generalization. Additionally, when evaluating both models with previously unseen test images, Part 1 achieves an accuracy of approximately 65%, whereas Part 2 demonstrates a notable improvement at 77%.

3. Test Image Predict Result

Part1 Confusion Matrix	Part2 Confusion Matrix
	

Analyzing the confusion matrices generated by the two models for predicting test images, it is observed that Part 2's confusion matrix exhibits a higher concentration of values along the diagonal. This suggests that Part 2 has more predictions aligning with the actual results. Additionally, across the 10 categories in CIFAR-10, Part 2's model demonstrates higher predictive accuracy compared to Part 1.

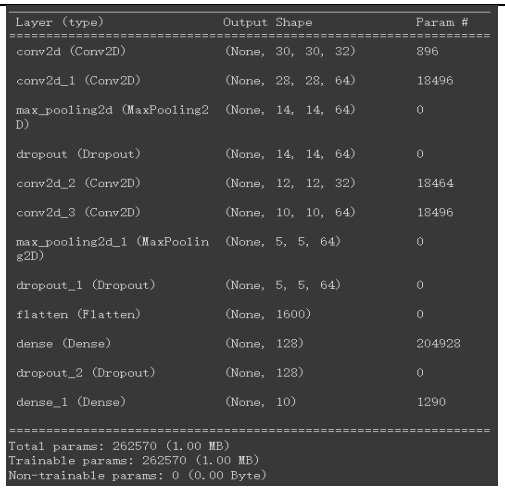
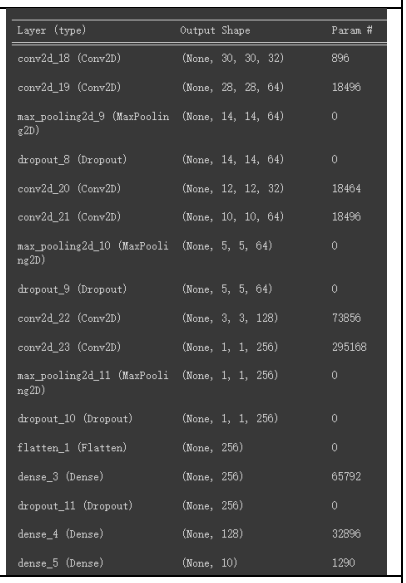
4. Model Comparison Summary Table

Aspect	Part1 Model	Part2 Model
Depth	Shallow	Deeper
Dropout	No	Yes
Overfitting	Probably (Potential)	Regulation applied
Robustness	Moderate	More Robust
Complexity	Lower	Higher
Training Time	Faster	Longer
Generalization	Limited	Better
Test Accuracy	65.68%	77.44%

Part 2's model is deeper and incorporates dropout layers, contributing to its ability to prevent overfitting. Additionally, it features more convolutional layers, offering the potential to learn more complex hierarchical features. Moreover, the Part 2 model employs a higher dropout rate in the final dense layer, potentially enhancing its generalization capabilities. In conclusion, the Part 2 model is likely to be more robust against overfitting.

Part 3: Hyperparameter selection

(a.) How number of layers affect model?

Aspect	Model1	Model2
Architecture		
Layers	12	17
Test Accuracy	77.44%	78.75

Only the model architecture was modified by adding more layers to make it deeper; all other parameters remained the same in part2. The experiment results indicate a slightly higher accuracy for the deeper model. However, upon researching benchmarks for the CIFAR-10 dataset, the top 10 models are notably more complex and deeper than mine. Furthermore, they employ techniques to address the vanishing gradient problem, contributing to their higher performance.

(b.) How the choice of loss function affect model?

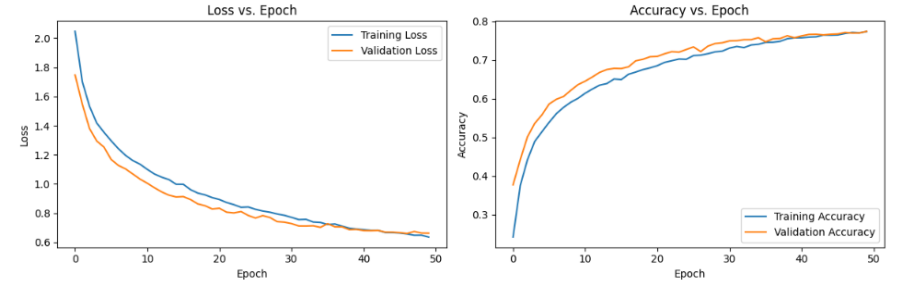
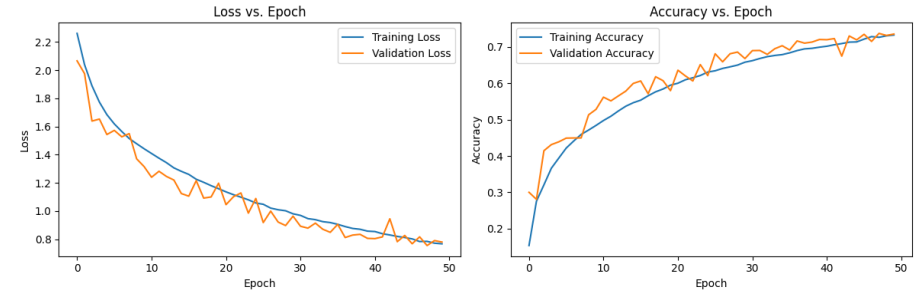
Loss function	Test Accuracy
categorical_crossentropy	77.44%
mean_squared_error	77.57%
mean_absolute_error	66.36%

Except for the loss function, all the remaining parameters are the same as the model in Part 2. Since the CIFAR-10 dataset involves assigning an input image to one of several predefined classes, I believe cross-entropy is the best choice for the loss function. Cross entropy is specifically designed for multi-class classification problems and works well when the output is a probability distribution over multiple classes. On the other hand, (MSE) and (MAE) are more suitable for regression tasks, where the goal is to predict continuous values. In our experiment results, the MAE did not perform as well as expected. However, the MSE in our experiment had almost the same accuracy as the cross-entropy.

(c.) How the choice of optimizer affect model?

Optimizer	Test Accuracy
Adam	77.44%
SGD	38.72%
rmsprop	73.58%

Result:

Optimizer	Training Process
Adam	
SGD	
RMSprop	

Except for the optimizer, all the remaining parameters are the same as the model in Part 2. The training curve indicates that the model using the Adam optimizer has already converged in Part 2. However, when using Stochastic Gradient Descent (SGD) as the optimizer with the same training parameters, the model has not yet converged. Therefore, it can be anticipated that training the model using SGD as the optimizer will require more time. Additionally, when using the Root Mean Square Propagation (RMSprop) optimizer, the training time is faster than with Adam. However, the accuracy is slightly lower than that achieved with Adam. No wonder that the Adam is the most popular optimizer!

(d.) How the epoch affect model?

We also used the same model with the different epoch to train. Here is the result, if the model has not converged yet, the smaller epoch will result in the worse accuracy. However, in some case, we need to use the validation loss to determine the early break condition to avoid the overfitting issue.

Part 4: Further Studies for CNN

<a>.

Identify some reasons of the accuracy deterioration while increasing the layers.

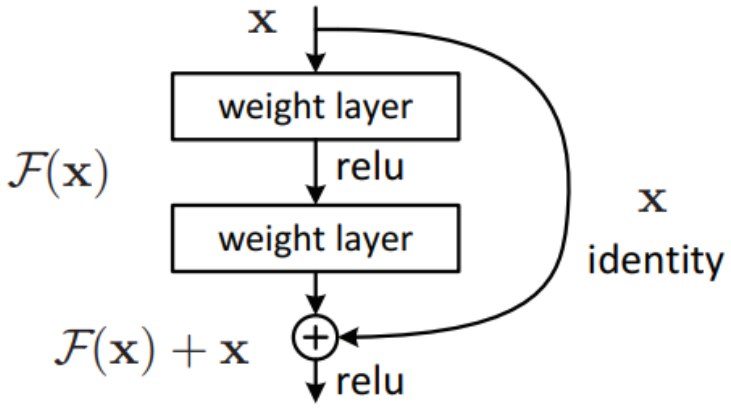
1. Vanishing/Exploding Gradients
2. Overfitting
3. Lack of feature reuses

While increasing the number of layers in a neural network can potentially enhance its representational capacity and ability to capture complex patterns, the three issues mentioned above can pose obstacles to effective learning. The vanishing gradient issue may cause backpropagation to result in slow learning or even prevent learning altogether. Similarly, exploding gradients may lead to numerical instability during weight updates, preventing the model from converging. Additionally, a model with a large number of parameters may have the capacity to memorize the training data, leading to overfitting. Lastly, the addition of more layers may also result in a failure of the network to effectively reuse lower-level features, potentially causing the model to underperform.

<b>

Then, find at least one network architecture that solves the problem. Write a paragraph for the architecture to explain how exactly it has been done to overcome the problem

ResNet is a good network architecture to solve the above problems because of the skip connection are introduced in the network architecture.



ResNet Skip Connection Figure (ref: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/)

The vanishing gradient problem can be addressed through the use of skip connections. This allows gradients to flow more easily during backpropagation. Furthermore, by providing a direct path through the skip connections, ResNet can be trained more effectively, even with a deeper network. Additionally, these skip connections enable the reuse of features from earlier layers, allowing the model to learn both fine and coarse-grained features. This facilitates better representation learning and can lead to improved generalization without encountering overfitting issues.