Student Number: 236807

Executive Summary

This paper builds several convolutional neural networks (CNN) to solve the classification problem of the CIFAR-100 dataset [1], and explore the hyperparameter tuning performances of the CNN model based on personal attempts and professional attempts (VGGNet [2]).

1. APPROACH

1.1. Base Model

The baseline model is constructed firstly, and there are three hidden layer groups defined in the base model.

The first group of layers is a combination of:

1. A convolutional layer;
2. A ReLU layer;
3. and a max pooling layer.

   To be specific, the convolutional layer's hyperparameters are out_channels = 8, kernel_size = 4, stride = 2 and padding =1, which can transfer the shape of input layer from 32 * 32 * 3 to 16 * 16 * 8; The pooling layer's hyperparameters are kernel_size = 2 and stride = 2, which will not change the shape of data.

The second group of layers is a combination of:

1. A convolutional layer;
2. A ReLU layer;
3. and an average pooling layer.

   The convolutional layer's hyperparameters are out_channels = 16, kernel_size = 4, stride = 2 and padding =1, which can transfer the shape of input layer from 16 * 16 * 8 to 8 * 8 * 16; The pooling layer's hyperparameters are kernel_size = 2 and stride = 2, which as the same with above mentioned max pooling layer, will not change the shape of data.

The third group of hidden layers is consisted of:

1. A fully connected layer;
2. and a ReLU layer.

   The latte's hyperparameters are 64 and 10, which will help convert the data to 10 output values for output layer.

1.2. Fine Tune

After the base mode is trained, in order to make the model have better performance, the hyperparameters will be tuned. There are two groups of hyperparameters might need to be tuned, the first group include learning rate and batch size, and the second group contains the hyperparameters of the layers, namely channels, kernel size, stride, padding, and so on.

2. METHODOLOGY

2.1. Download Dataset

The data is download by python code provided by the dataset [1]. There are two datasets (one is trainset, and one is testset). The normalization is implemented with the download process.

2.2. Define Hyperparameters

Except the hyperparameters in the model, the rest of hyperparameters is defined here, they are batch size, learning rate, and epochs.

For the base model, the batch size = 4, learning rate = 0.001, which is set personally without any references.

2.3. Initialize

The downloaded two data sets are load in to two data loaders, the loss is set to be CrossEntropyLoss(), and optimizer is set to be SGD.

2.4. Training and Testing

During training, in order to check the on-going performance, the loss will be print every 2000 iterations. After training, the accuracy of the model on the trainset and testset will be calculated.

For the base model, the training accuracy is 15.32% and the testing accuracy is 15.12%.

3. RESULT AND DISCUSSION

3.1. Hyperparameter Tuning

Firstly, the few hyperparameters need to be tunned are the learning rate and batch size. Since the base model is trained for quite a long time and the loss hardly decreased, the batch size is set to be larger, and the learning rate is set to be smaller.

The tuning results between the two hyperparameters are given in the below table.

Training Index	Learning Rate	Batch Size	Training Accuracy	Testing Accuracy
Base	0.001	4	15.32%	15.12%
1	0.0001	4	32.29%	32.14%
2	0.0001	10	38.25%	38.16%
3	0.0001	20	26.16%	26.72%
4	0.01	10	63.00%	60.49%
5	0.1	10	58.33%	56.78%

Table 1: first group of hyperparameter tuning results

Because the observed loss fluctuates greatly, the first guess of the reason is that the learning rate is too high, so that the gradient cannot be reduced to the lowest point. Therefore, the initial idea is to reduce the learning rate (Training Index 1). After obtaining a certain improvement in the accuracy rate, the batch size was increased to reduce the training time of the model (Training Index 2 and 3). After trying two batch sizes, batch size = 10 may be a better choice. On this basis, a larger learning rate has been tried (Training Index 4). When training the Training Index 4, the loss was obviously decreasing, but the speed was slower, and the accuracy rate was improved very high. So tried a larger learning rate (Training Index 5). In general, a more appropriate learning rate is 0.01 and the batch size is 10.

Secondly, the dropout is applied. The result of different dropout rates is shown below.

Training Index	Dropout Rate	Training Accuracy	Testing Accuracy
6	0.01	54.42%	55.57%
7	0.02	49.75%	55.72%

Table 2: second group of hyperparameter tuning results

Dropout is considered to be a normalization method, and have some benefits to overfitting, though the current performance is not overfitting. The more commonly used dropout rate is 0.2, for comparison, both 0.1 and 0.2 are used for trials. However, the dropout performance cannot be seen from training accuracy.

Except dropout, another method is to add a convolutional layer before the defined layers of base model. To be specific, the add layers hyperparameters are:

1. in_channels = 3;
2. out_channels = 64;
3. kernel_size = 2;
4. and stride = 2.

   The idea is got from professional attempts, namely VGGNet [2].

And other changes to the model is that the width of the two convolutional layers in the original first two groups of layers, they are changed from 8 and 16 to 128 and 256 respectively, which is also based on Figure 1 [2].

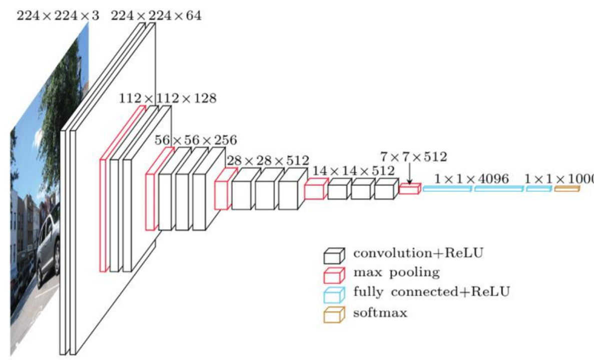

Figure 1: VGGNet [2]

This deeper and wider network performs very well, especially during on-going observations. The downward trend of loss is not particularly volatile, and the decline is fast. The accuracy on train set is 99.72%, and on test set is 68.12%. Hence, the final tuning is on reducing epochs (Table 3) to avoid over fitting. The result of epoch = 25 is illustrated below (Figure 1), the loss decreasing figures are almost the same, no matter the number of epochs.

Training Index	Epochs	Training Accuracy	Testing Accuracy
8	30	78.42%	66.72%
9	20	95.51%	67.20%
10	25	97.10%	65.13%

Table 3: final tuning

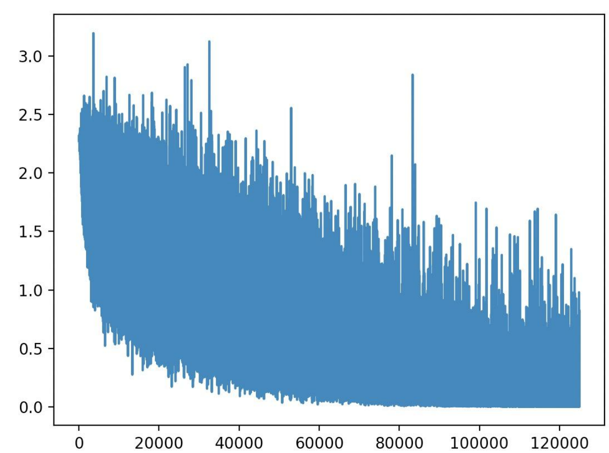

Figure 2: loss decreasing

3.2. The Best Model

As a result, the best model has four hidden layers, and it is referred partially from VGGNet [2]. The result of the model on the train set is 78.68%, and on test set is 66.72%.

4. CONCLUSION

This report explores hyperparameters tuning on a famous image dataset CIFAR-10 [1]. The explored hyperparameters are learning rate, batch size, drop out, depth and width of the CNN.

The learning rate and batch size only can help the model to perform from a very poor level to a poor level, while changing the architecture of the network progress a lot. Hence, the conclusion is that a deeper and wider CNN tend to perform more higher accuracy.

References

[1] A. Krizhevsky, “CIFAR-10 and CIFAR-100 datasets”, Cs.toronto.edu. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html.

[2] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv.org, 2022. [Online]. Available: https://arxiv.org/abs/1409.1556.