Deep Learning Assignment2

RE6111024 葉嘉浤

***Abstract*—This assignment is to implement LeNet5 with tensorflow and pytorch, then compare to our handcraft framework. The code can be found at https://github.com/chyeh1126/DeepLearning-2023/tree/main/HW3.**

***Index Terms*—Deep Learning, Neural Network, Image Classification**

I. Introduction

In the field of computer vision, common tasks include image classification, object detection, and so on. The goal of image classification is to use the features presented in an image to let the model output probability values of the presence of certain objects in the image, thereby determining whether certain objects exist in the image.

II. Method

In this section, we will introduce the model used in this assignment and the related hyperparameter settings.

A. LeNet5

LeNet5 is a convolutional neural network model for image classification proposed by Yann LeCun and others in the year 1998. The following figure and table show the architecture of LeNet5 and the related hyperparameter setting.

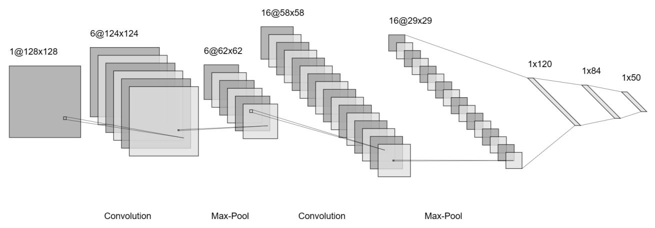

Fig. 1. The structure of LeNet5

TABLE I
Hyperparameter setting of LeNet5

Layer	Setting
Convolution 1	kernel size = 5×5, in channels = 1, out channels = 6
Pooling 1	kernel size = 2×2
Convolution 2	kernel size = 5×5, in channels = 6, out channels = 16
Pooling 2	kernel size = 2×2
Linear 1	in dim = $5 \times 5 \times 16$, out dim = 120
Linear 2	in dim = 120, out dim = 84
Linear 3	in dim = 84, out dim = 50

III. Experiment

The dataset for this experiment consists of 63,325 images for training, 450 for validation, and 450 for testing, all of which belong to 50 different categories.The images are read as gray-scaled images, then resized to 32×32. For convenience, we use ”LeNet5-HC”, ”LeNet5-TF”, ”LeNet5-PT” to represent handcraft LeNet5, tensorflow version LeNet5, and pytorch version LeNet5. To compare the different version LeNet5, we use accuracy, inference time, space complexity, and flops to evaluate model performance and efficiency.

Besides the overall performance comparison, we further conduct additional two experiments. The first is to compare the training time and the inference time of LeNet5-TF by using static graph and dynamic graph. We use ”LeNet5-TF-S” and ”LeNet5-TF-D” to represent the model implemented with the two different graph. The second is to compare the overall performance of the original LeNet5-PT and the pruning LeNet5-PT. We use ”LeNet5-PT-P” to represent the pruning LeNet5-PT.

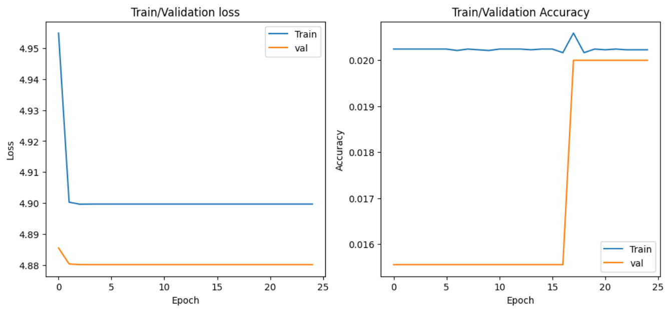

Fig. 2. Training and Validation performance of LeNet5-HC

The above figure is the loss and accuracy curve of LeNet5-HC at training set and validation set. According to the left figure, the training and validation loss converge at early stage. The right figure shows the training and validation accuracy. we can see the training accuracy changes slightly and almost remain 0.02, while the validation accuracy increases to 0.02 after 16th epoch.

The above figure is the loss and accuracy curve of LeNet5-TF at training set and validation set. The left figure suggests that the training and validation loss decreases as the epoch increases and will converge to 3.3 or 3.4. According to the right figure, both two types of accuracy increases as the epoch increases and the values are approximately 0.14.

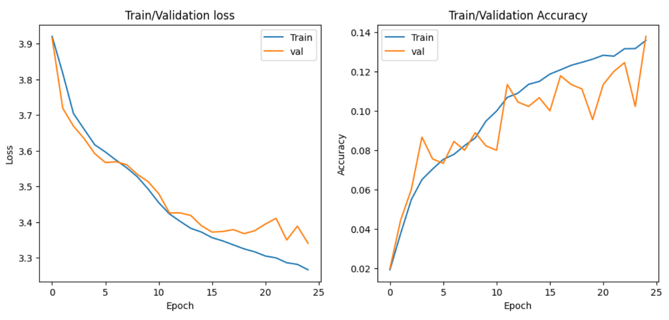

Fig. 3. Training and Validation performance of LeNet5-TF

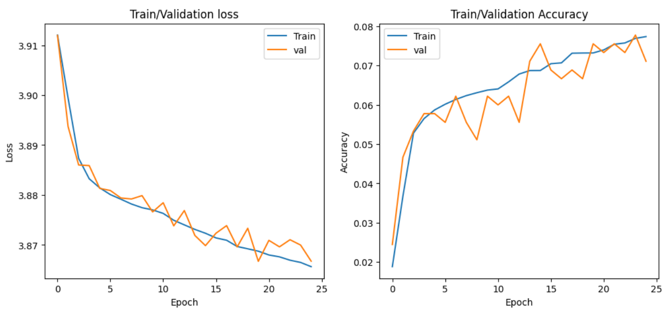

Fig. 4. Training and Validation performance of LeNet5-PT

The above figure is the loss and accuracy curve of LeNet5-PT at training set and validation set. The left figure suggests that the training and validation loss decreases as the epoch increases and will converge to 3.87. According to the right figure, both two types of accuracy increases as the epoch increases and the values are approximately 0.07.

TABLE II
Overall performance on testing set

Model	**Metric**			
	Accuracy	***Inference time***	***Space complexity***	***Flops***
LeNet5-HC	0.02	0.16	-	-
LeNet5-TF	0.18	0.09	0.25	0.027
LeNet5-PT	0.09	0.01	0.37	0.053

The above table records the performance of three versions of LeNet5. We compare the accuracy and the inference time of three versions of LeNet5. As for the space complexity and the flops, we only compare LeNet5-TF and LeNet5-PT. LeNet5-TF gets the highest accuracy on testing set, which is 0.18, while the accuracy of LeNet5-HC is only 0.02. LeNet5-PT achieves the fastest inference time. It only takes 0.01 second to inference. LeNet5-HC needs 0.16 second to inference, which is the most time-comsuming. According to the space complexity, LeNet5-PT requires memory capacity of 0.37 MB to storage, slightly larger than LeNet5-TF, which needs 0.25 MB. For the flops, the value of LeNet5-PT is also larger than LeNet5-TF, which implies LeNet5-PT needs to run more operations than LeNet5-TF.

TABLE III
Performance of Static graph and Dynamic graph

Model	**Metric**	
	Training time	***Inference time***
LeNet5-TF-S	17.09	0.208
LeNet5-TF-D	22.22	0.193

The above table shows the performance of the static graph method and that of the dynamic graph method. According to the table, LeNet5-TF-S is faster than LeNe5-TF-D in training stage, whlie LeNet5-TF-D is faster than LeNet5-TF-S in inference stage.

TABLE IV
Performance of original model and pruning model on testing set

Model	**Metric**			
	Accuracy	***Inference time***	***Space complexity***	***Flops***
LeNet5-PT	0.09	0.01	0.37	0.054
LeNet5-PT-P	0.09	0.01	0.37	0.027

The above table shows the performance of the original pytorch LeNet5 and that of the pruning pytorch LeNet5. According to the table, the accuracy, inference time, space complexity are same at two models.But the flops of LeNet5-PT is larger than LeNet5-PT-P, implies that the amount of operations decreases by pruning the original LeNet5-PT.

IV. Conclusion

In this assignment, we implement LeNet5 with tensorflow and pytorch, and compare to our handcraft LeNet5 with an image classification task. We compare them with accuracy, inference time, space complexity, and flops. We further compare the inference time and training time with static graph method to those with dynamic graph method in tensorflow, and compare the performance and speed of original model and those of pruning model in pytorch.

According to the experiments, the tensorflow version has the best performance and needs less memory space and calculations, while the torch version is the most efficiency. We compare the tensorflow version with static graph and dynamic graph, and find that use static graph method will have less training time; use dynamic graph method will inference slight fast than static graph method. We also prune the pytorch version to compare the original one with the 4 metrics. The performances of two versions are almost same, but the pruning model requires less computational power tha the original one.

References

- LeNet5 : https://www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/
- Tensorflow space complexity : https://pypi.org/project/model-profiler/
- Tensorflow measure FLOPs : https://pypi.org/project/keras-flops/

- Pytorch space complexity : https://blog.csdn.net/weixin_41519463/article/details/102468868
- Pytorch measure FLOPs : https://blog.csdn.net/yiran103/article/details/97756720
- Pytorch summary model : https://clay-atlas.com/blog/2020/04/26/pytorch-cn-note-use-torchsummary-visualization-model/
- Pytorch pruning : https://pytorch.org/tutorials/intermediate/pruning_tutorial.html