ECE371 Neural Networks and Deep Learning Assignment 1

Xin Wen 22308181
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
wenx76@mail2.sysu.edu.cn

Abstract: This report presents the completion of two exercises in the context of flower image classification tasks. In the first exercise, we fine-tune classification models using the flower dataset, leveraging the BaseDT library for data preparation and experimenting with various pre-trained models and hyperparameters to achieve optimal performance. The second exercise involves completing a training script. Experimental results demonstrate the effectiveness of different configurations, with DenseNet achieving the highest accuracy of 97.7113% in fine-tuning tasks.

Keywords: Deep Learning, Image Classification, ResNet

1 Introduction

In this assignment, we try to finish the two exercises that the assignment requires.

The first exercise is to complete the fine-tune training of classification models. In this exercise, we first write a Python script to prepare the flower datasets, split the datasets into training and validation sets, and organize the datasets into ImageNet format. We use the BaseDT library to accomplish it. Then, we build profiles for fine-tuning, and use the tools/train.py to fine-tune the model. To achieve higher accuracy, we try to use diffenert parameters and diffenert pre-trained models.

The second exercise asks us to complete the training script and accomplish the training process.

2 Related Work

The evolution of deep neural network architecture has driven breakthroughs in image classification tasks. He et al. [1] used residual networks (ResNet) to address the vanishing gradient problem in deep networks, enabling the training of very deep networks with hundreds of layers. Xie et al. [2] introduced the idea of aggregating multiple paths in a single network to advance ResNet, naming it ResNeXt. Huang et al. [3] introduced dense cross-layer connectivity to maximize feature reuse and improve gradient flow, resulting in DenseNet. Szegedy et al. [4] proposed Inception-v2 and Inception-v3, which uses factorized convolution to reduce the number of parameters and improve computational efficiency. Howard et al. [5] proposed MobileNetV3, which uses network architecture search (NAS) and NetAdapt algorithm to optimize the model.

3 Method

In this section, we will describe the methods used in the assignment.

3.1 Exercise 1

3.1.1 Data Preparation

To prepare the datasets, we use the BaseDT library to split the datasets into training and validation sets in an 8:2 ratio and organize the datasets into ImageNet format. The BaseDT library provides a convenient way to handle the data preparation process, and it supports various dataset formats, including IMAGENET, VOC, and COCO. So we utilize BaseDT for data preprocessing tasks, aimed at avoiding reinventing the wheel. The data preprocess script can be found at `data_preprocess.py`.

3.1.2 Fine-tune Training

To fine-tune the model, we use the TODO inheritance mechanism to create a profiles for ResNet50. And we modify the model's head, the dataset configuration, the evaluation method and learning rate to adapt to the flower dataset. Then we download the pre-trained model from the model zoo of MMpretrain. Finally, we use the tools/train.py to fine-tune the model. The training script can be found at `train.ipynb`. The training result can be found at section 4.1.

3.1.3 Comparision experiments

To achieve higher accuracy, we try to use different parameters and different pre-trained models. The detail of our Comparision experiments can be found at section 4.1.

3.2 Exercise 2

In this exercise, we are required to complete the training script and accomplish the training process. First, we need to add five data augmentation methods and normalization methods, we choose random horizontal flip, random vertical flip, random rotation, random crop and color jitter as our data augmentation methods. Then, we need to modify the last fully connected layer, add the loss function and optimizer. We choose cross entropy as our loss function and SGD as our optimizer, with a learning rate of 0.001 and momentum of 0.9. Finally, we are asked to save the best model. After completing the training script, we run the training script and save the best model. The training script can be found at `main.py`.

To analysis the training process, we use pyplot to visualize the training process. The training curves can be found at section 4.2.

4 Experiments

In this section, we will present our experimental results and analysis.

4.1 Exercise 1

4.1.1 Baseline

We first use the ResNet50 as our baseline model. We use the SGD optimizer with a learning rate of 0.01 and momentum of 0.9. We also We set the batch size to 32 and the number of epochs to 30.

The best accuracy of the baseline model is 95.9507% on the validation set. The training process is shown in Figure 1.

4.1.2 Compare Frozen Blocks

We try to freeze diffenert blocks of the model to see the effect of frozen layers on the accuracy. The ResNet50 model has 4 blocks, and we try to freeze the first block, the first two blocks, the first three blocks and all the blocks. The results are shown in Table 1.

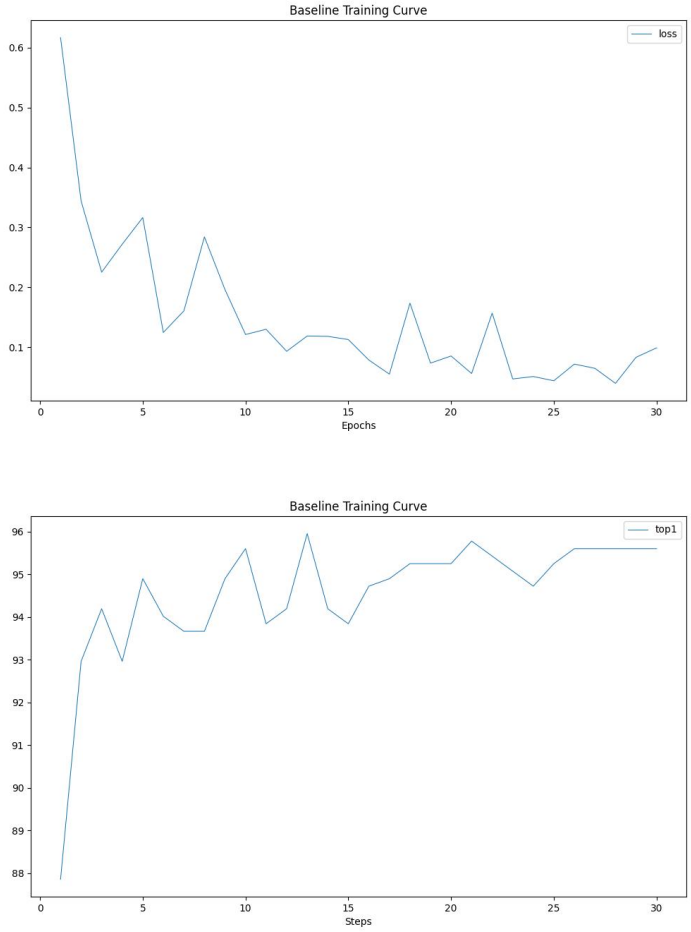

Figure 1: The training process of the baseline model.

From the table, we can see that the accuracy of the model is higher when we freeze the first three blocks. This indicates that freezing the initial layers can preserve the feature extraction capability of the pre-trained model. The accuracy is 97.0070% when we freeze the first three blocks. However, when we freeze all the blocks, the accuracy drops to 94.7183%. This shows that freezing all the blocks can impair the model's capacity.

Frozen Blocks	Accuracy
0	95.9507%
1	95.9507%
2	96.4789%
3	**97.0070**%
all	94.7183%

Table 1: The accuracy of the model with different frozen Blocks.

4.1.3 Compare Learning Rate

We try to use different learning rates to train the model. We set the learning rate to 0.0001, 0.001, 0.01. The results are shown in Table 2.

From the table, we can see that when the learning rate is 0.001, the accuracy is the highest, which is 96.1268%. When the learning rate is too small or too large, the accuracy drops. This indicates that the learning rate is a very important hyperparameter in the training process.

Learning Rate	Accuracy
0.01	95.5986%
0.001	**96.1268**%
0.0001	95.0704%

Table 2: The accuracy of the model with different learning rates.

4.1.4 Compare Pre-trained Model

We try to use different pre-trained models to train the model. We choose ResNet50, ResNeXt50, DenseNet, MobileNet-v3, Inception-v3 as our pre-trained models with the same training parameters. The results are shown in Table 3.

From the table, we can see that the DenseNet model has the highest accuracy, which is 97.7113%. To our surprise, the MobileNet-v3 and Inception-v3 models have lower accuracy than the ResNet50 model. This may because the training parameters are not suitable for these models.

Model	Accuracy
ResNet50	96.1268%
ResNeXt50	97.1831%
DenseNet	**97.7113**%
MobileNet-v3	95.4225%
Inception-v3	95.5986%

Table 3: The accuracy of the model with different pre-trained models.

4.2 Exercise 2

In exercise 2, we use SGD as our optimizer and cross entropy as our loss function. We set the learning rate to 0.001 with a StepLR scheduler and the momentum to 0.9. We also set the batch size to 32 and the number of epochs to 25.

The best accuracy of the model is 94.5614% on the validation set. The training process is shown in Figure 2.

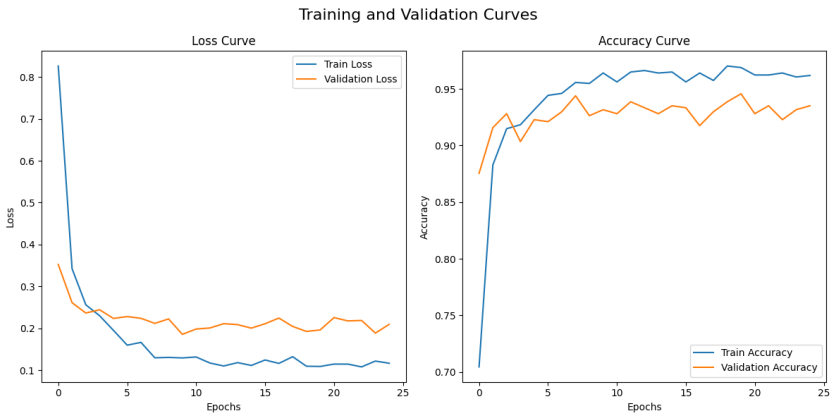

Figure 2: The training process of exercise 2.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[3] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2016. URL https://arxiv.org/abs/1608.06993.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.