ECE371 Neural Networks and Deep Learning Assignment 1

YuHao Li 22308091
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
`liyh578@mail2.sysu.edu.cn`

Abstract: This work focuses on fine-tunning a flower classification model using PyTorch and a pretrained ResNet-18 architecture. The key strategies of this work includes dataset data augmentation, modifying the classification head and importing optimizer with momentum SGD. With the above works, the approach achieved a peak validation accuracy of 93.51% at epoch 23. The results demonstrated that fine-tuning pretrained models enables considerable performance in image classification without extensive dataset or resources.

Keywords: Image Classification, ResNet, Model Fine-Tuning

1 Introduction

Image classification, as one of the core tasks of computer vision, has important applications in many practical scenarios, such as plant recognition and medical image analysis. Deep neural networks provide excellent support for solving image classification problems.

Under the constraints of small-scale datasets and limited computing resources, how to develop high-precision image classification models efficiently still faces challenges. To address this problem, This work fine-tuned a pre-trained model using limited training data and achieved a considerable accuracy rate. The result shows that we can train an image classification model with high accuracy without requiring extensive computational resources and abundant datas via pretrained models.

2 Related Work

The development history of image classification technology is closely related to many fields such as computer vision, machine learning, and neural networks. Recent advances [1] in deep learning have made transfer learning a standard approach for image classification.

In 2015, He Kaiming al. proposed the ResNet(Residual Network) architectures[2], which have shown exceptional performance across image classification due to its residual designs. It groundbreaking use skip connection between blocks, effectively mitigating vanishing gradients in deep networks and enables stable training of networks with over 100 layers while ensuring feature propagation efficiency. In addition, its architecture design also makes it particularly effective for transfer learning.

Beyond that, another model called IGPT(image GPT)[3], proposed by OpenAI in 2020, also performs well in image recognition and classification, which adopt two-stage training and achieve an unsupervised training through pixel sequence autoregression.

ResNet's residual design ensures "uninterrupted gradient flow through shortcut connections, making extremely deep networks trainable and achieving superior transfer learning performance" [2]. It proved that this architecture preserves low-level feature stability during fine-tuning while adapting high-level task-specific patterns, offering greater computational efficiency and feature reusability compared to IGPT's pixel-reconstruction approach, which demands full-scale retraining[4].

3 Method

3.1 Training environment configuration

The training environment is based on python3.9 and equipped with PyTorch architecture version "2.7.0". Therefore, the python script can smoothly run in the virtual environment.

3.2 Data Augmentation and normalization

The given dataset contains 2848 images across 5 classes of flowers(daisy, dandelion, rose, sunflower, tulip), which split into 80% for training and 20% for validation sets. Images are organized in ImageNet format so that it can be used by pretrained ResNet-18 model used in this work. Some augmentation operations are applied as follows: random rotation (±30°), random resized crop (224*224), horizontal/Vertical flipping, Color jitter (brightness/contrat/saturation=0.2). All images are normalized using ImageNet statistics: $\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$. After this works, the quality of the dataset has been improved and it can be adapted to this ResNet-18 model.

3.3 Model Fine-Tuning Strategy

3.3.1 Classification Head Modification

The ResNet-18 backbone is initialized with ImageNet-pretrained weights. The final fully-connected layer is replaced with a new linear layer (512→5 dimensions) to match our classification task.

3.3.2 Loss Function Definition

The loss function is defined using cross-entropy loss function, which is suitable for multi-classification tasks and can automatically processes logits and calculates losses.

3.3.3 Optimizer Definition and Learning Rate Schedule

Fine-tuning is carried out based on the pre-trained ResNet-18 model of ImageNet. The model is optimized using SGD with momentum (lr=0.001, momentum=0.9), which has considerable stability in fine-tuning pretrained models, preventing overfitting on small datasets by avoiding aggressive parameter. The step-decay learning rate is set to decreases tenfold every seven rounds, which prevents early optimization instability and balances feature preservation and adaptation, while momentum accelerates convergence through gradient smoothing.

3.4 Windows multi-process and other parameters adaptation

In the actual experiment, due to the multi-process tasks involved in the model fine-tuning, compatibility issues with the windows system emerged. Therefore, the main program entry was introduced in the code and the multi-process support of the windows system was obtained to ensure the normal operation of the script. At the same time, the model parameters were reset to avoid the possible adaptation problems it may cause.

Otherwise, due to the limitation of video memory, after conducting a training with Batch_size=32 once, the Batch_size parameter was adjusted to 16 and the training was conducted again. The final submitted optimal model is obtained by the function script with Batch_size=16 and epoch=25.

4 Experiments

4.1 Training Process

The results under condition "Batch_size=16,Epochs=25" are presented in Table 1. The model achieves peak validation accuracy of 93.51% at epoch 23. It's probably because the model achieves the best generalization ability in the middle epochs and in subsequent epochs, although the performance on the training set continues to improve, overfitting may causes a decline in generalization ability for the validation set, which is a reasonable result of balancing fitting and generalization.

Meanwhile, since the experiment adopts a learning rate with a fixed step size attenuation, although it can stabilize the early training, when the learning rate drops to a relatively low order of magnitude in the later rounds, the update amplitude of the model is extremely small. On the one hand, this may reduce the risk of over-fitting, but on the other hand, it may also lead to convergence cessation or regression, thereby resulting in a decrease in model performance. In fact, this fixed attenuation setting will lead to the lack of flexibility in model training, making it more difficult to balance the fine-tuning requirements in the later stage. Adopting a more dynamic scheduling method might be able to improve this problem.

The value of training loss eventually stabilized at around 0.22 while the validation loss around 0.2. This is because the mathematical property of cross-entropy loss calculation determines that the loss value will not approach zero but fall within a stable range. Therefore, this is also a reasonable result.

Table 1: Result when Batch_size=16,Epochs=25

Epoch	Learning Rate	Train Loss	Val Loss	Accuracy Rate
1	0.001000	0.7551	0.4084	0.8333
8	0.000100	0.2416	0.2580	0.9140
16	0.000010	0.2223	0.1904	0.9246
23	0.000001	0.2271	0.1981	0.9351
25	0.000001	0.2138	0.2031	0.9263

4.2 Best Accuracy Rate Comparison

Table 2: Best Accuracy Rate Under different Settings

Training Parameters Settings	Best Accuracy Rate
Batch_size=32,epochs=25	0.9316
Batch_size=16,epochs=20	0.9350
Batch_size=16,epochs=25	0.9351

It can be seen from Table 2 that for different batch and epochs Settings, the final optimal accuracy rate does not vary much. In fine-tuning, pre-trained models already have good feature representations, requiring only adaptation of the last few layers. When the batch size is between 16 and 32, the gradient estimation balances stability and efficiency, and the model is insensitive to batch sizes within this range. With 20-25 epochs, the model converges sufficiently close to the optimal solution, and further training does not significantly improve accuracy, leading to similar final accuracies. This reflects the strong initialization of pre-trained models and efficient task adaptation.

References

[1] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *MIT Press*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *IEEE*, 2016.

[3] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining from pixels. In *37th International Conference on Machine Learning: ICML 2020, Online, 13-18 July 2020, Part 3 of 15*, 2021.

[4] K. He, R. Girshick, and P. Dollar. Rethinking imagenet pre-training. , 000(8972782), 2019.