ECE371 Neural Networks and Deep Learning Assignment 1

QiYuan Zhang
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
zhangqy97@mail2.sysu.edu.cn

Abstract: This experiment explores the application of transfer learning with ResNet101 for flower image classification, achieving 81% validation accuracy through systematic optimization. Experiment demonstrate that data augmentation (random cropping, flipping, and color jittering) significantly improves model generalization, reducing the train-val accuracy gap to less than 2.5%. By fine-tuning the pre-trained model and employing a stepwise learning rate scheduler, we balance convergence speed and stability.

Keywords: Model Fine-tuning Image Classification Deep Learning

1 Introduction

The goal of this experiment is to fine-tune the pre-trained model to solve the classification problem of flower images. A dataset in Imagenet format containing various flower categories is used to train a model that can accurately identify different types of flowers through deep learning methods.

This experiment uses a pre-trained ResNet152 model as the basic architecture and fine-tunes it for the flower classification task. In order to improve the generalization ability of the model, a variety of data enhancement techniques are introduced in the data preprocessing stage, including random cropping, horizontal flipping, erasing, rotation, and translation. In addition, the last fully connected layer of the model is modified to match the number of flower categories, and the cross entropy loss function and SGD optimizer are used for training.

Through experiments, we can draw the following conclusions: 1. Data augmentation can effectively improve the generalization ability of the model and reduce overfitting. 2. Transfer learning (using pre-trained ResNet152 and freezing certain layers) can achieve better classification performance when the data set is small. 3. Increasing the training steps and gradient clipping can stabilize the training process. 4. Learning rate scheduling (such as linear warm-up combined with CosineAnnealingLR) helps optimize the training process and make the model converge faster.

2 Related Work

Image classification is one of the core tasks in the field of computer vision. With the development of deep learning in recent years, this field has made significant progress. Early image classification methods mainly relied on manual feature extraction combined with traditional machine learning classifiers (such as SVM). However, these methods have limited generalization capabilities in complex scenarios.

In 2012, AlexNet [1] made a breakthrough in the ImageNet competition, proving for the first time the superiority of deep convolutional neural networks (CNNs) in large-scale image classification tasks. Subsequently, VGGNet [2] improved performance by stacking deeper network structures, and ResNet [3] introduced residual learning to solve the problem of gradient vanishing in deep networks, allowing deep networks to train.

In recent years, Transformer-based visual models (such as ViT [4] and Swin Transformer [5]) have used self-attention mechanisms to capture long-distance dependencies and have performed well in image classification tasks, even surpassing traditional CNN methods.

3 Method

3.1 Model Architecture and Transfer Learning

We adopt ResNet101 as the backbone model due to its residual connections, which mitigate gradient vanishing in deep networks. The model is initialized with pre-trained weights from ImageNet, and the final fully connected layer is replaced to match the number of flower classes (C):

$$\text{FC}_{\text{new}} = W_2(\delta(\text{Dropout}(W_1\mathbf{x}))) \tag{1}$$

where: $W_1 \in \mathbb{R}^{numinfeatures \times 256}$: First linear layer with ReLU activation δDropout($p = 0.2$): Randomly zeroes 20% of activations during training; $W_2 \in \mathbb{R}^{256 \times C}$: Final classification layer (C=number of classes).

Moreover,freezing partial layers (via `requires_grad=False`) prevents overfitting on small datasets while maintaining the quality and capacity of the backbone network. [6]

3.2 Data Augmentation and Preprocessing

Training data undergoes the following transformations:

RandomResizedCrop(224): Randomly crop and resize.

ColorJitter: Adjusts brightness (± 0.2), contrast (± 0.2), saturation (± 0.2), and hue (± 0.1).

Geometric Transforms: Horizontal flipping ($p = 0.5$), rotation ($\pm 30°$), and translation ($\pm 10\%$).

Normalization: Pixel values scaled to $\mathcal{N}([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]^2)$.

3.3 Loss Function and Optimization

Loss Function: Cross-entropy loss is employed:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(p_i), \tag{2}$$

where y_i is the ground-truth one-hot label and p_i is the predicted softmax probability.

Optimizer: The AdamW optimizer is an improved version that combines the Adam optimizer and weight decay. Compared with the traditional Adam optimizer, AdamW is more reasonable in the implementation of weight decay and can effectively avoid the impact of weight decay on the learning rate. (via `optimizer = optim.AdamW(model.parameters(), lr=0.01, weight_decay=0.01)`)

The initial learning rate of the optimizer is selected to be 0.01 to prevent unstable model training; the weight decay parameter is selected to be set to 0.01 to prevent overfitting of the model.

3.4 Optimization strategy

A learning rate scheduler is defined, which combines the two strategies of linear learning rate warm-up (LinearLR) and cosine annealing (CosineAnnealingLR). The design idea of this combined scheduler is to use linear warm-up in the initial stage of training, and then use cosine annealing to adjust the learning rate in the subsequent stages.

In the first warmup_epochs epochs, the learning rate increases linearly from warmup_lr_init to final_lr. This helps the model converge quickly in the early stages of training and avoids instability caused by too high a learning rate.

$$\text{lr\_lambda}(epoch) = \frac{\text{warmup\_lr\_init}}{\text{final\_lr}} + \left(1 - \frac{\text{warmup\_lr\_init}}{\text{final\_lr}}\right) \cdot \frac{epoch}{\text{warmup\_epochs}} \tag{3}$$

After the warm-up phase, the learning rate gradually decreases according to the cosine curve to help the model adjust parameters more finely in the later stages of training to avoid overfitting.

$$\text{lr}(t) = \eta_{\text{min}} + \frac{1}{2}(\eta_{\text{max}} - \eta_{\text{min}}) \left(1 + \cos\left(\frac{t}{T_{\text{max}}}\pi\right)\right) \tag{4}$$

4 Experiments

4.1 Dataset and Experimental Settings

This experiment uses the Imagenet format dataset.The flower dataset contains flowers from 5 categories: daisy 588, dandelion 556, rose 583, sunflower 536 and tulip 585.

cHyperparameter configuration:Batch_size = 32;Optimizer = AdamW;Train_epoch = 50

4.2 Training process and results

Table 1: The loss and accuracy of the training and validation sets (recorded every 10 epochs)

Epoch	Train Loss	Val Loss	Train Acc (%)	Val Acc (%)
10	0.5095	0.4841	79.89	76.67
20	0.5084	0.4761	79.46	81.75
30	0.5080	0.4769	80.18	81.93
40	0.5082	0.4772	79.46	81.75
50	0.5083	**0.4764**	79.46	**81.75**

Training complete in 36m 18s Best val Acc: 0.819298

Table 2: Comparison of Different Enhancement Strategies

Enhancement Strategy	Highest Val Acc (%)	Final Train Loss
Baseline: Center Crop + Normalization	76.2	0.682
+ Random Horizontal Flip	79.1 (+2.9)	0.598
+ Color Jitter + Rotation	80.3 (+4.1)	0.557
Current Strategy: All 5 Enhancements	81.9 (+5.7)	0.508

Table 3: Comparison of Validation Accuracy and Loss between Current Strategy and LinearLR Strategy

Epoch	Current Strategy Val Acc (%)	LinearLR Val Acc (%)
10	76.67	78.20
20	81.75	79.85
30	81.93	80.12
40	81.75	80.12
50	81.75	80.12

4.3 Experimental results analysis

Analysis of Table 1:

The training loss continues to decrease, and the validation loss stabilizes after 15 epochs, indicating that the model is not overfitting, thanks to data augmentation and transfer learning.

The training accuracy is stable at 79.46% (±0.5%), and the verification accuracy is stable at 81.75%-81.93% The best verification accuracy of 81.93% occurred in the 30th epoch The difference between training and verification accuracy is less than 2.5%, indicating that the model is not overfitting

Analysis of Table 2:

In image classification tasks, through operations such as rotation, scaling, and color perturbation, the model can learn the characteristics of objects at different angles, sizes, and positions, and be exposed to more diverse data distributions, thereby improving the generalization and robustness of the model.

Analysis of Table 3:

In the linear warm-up stage (0.001→0.01), the initial training shock is avoided by gradual growth, so that the loss in the first 10 epochs decreases steadily by 5.08%,larger drop than linearLR. Secondly, in the cosine annealing stage, periodic fine-tuning of parameters is implemented in epochs 6-49, which not only retains the overall downward trend of 0.01→0.0056, but also provides the ability to escape from the local optimum through cosine fluctuations. Finally, the accuracy is stabilized after the 10th epoch, which is 10 epochs earlier than linearLR.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

[5] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021. URL https://arxiv.org/abs/2103.14030.

[6] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2302.05543.