ECE371 Neural Networks and Deep Learning Assignment 1

Hanhan Zeng 22308006
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
`zenghh29@mail2.sysu.edu.cn`

Abstract: This report presents a comprehensive study on flower image classification using deep learning techniques based on PyTorch. We implemented both ResNet18 and ResNet34-based classifiers and explored various strategies to improve model performance, including advanced data augmentation, optimizer selection, and learning rate scheduling. The training process and results are compared with the official MMPretrain (MMClassification) ResNet18 model to analyze the gap between a teaching-level implementation and an industrial-level framework. The report also discusses the challenges encountered, the effectiveness of different techniques, and provides suggestions for further improvement. Training and validation loss and accuracy curves are visualized to illustrate the model's learning dynamics, and the impact of increasing model depth is analyzed.

Keywords: Image Classification, Deep Learning, ResNet18, PyTorch, Data Augmentation

1 Introduction

Image classification is a fundamental task in computer vision, serving as the basis for many advanced applications such as object detection, scene understanding, and medical image analysis. In this assignment, we focus on the classification of flower images into five categories using deep convolutional neural networks. The primary objective is to gain hands-on experience with the end-to-end deep learning workflow, including data preprocessing, model fine-tuning, and performance evaluation.

Additionally, we aim to understand the differences between a custom PyTorch implementation and a state-of-the-art industrial framework like MMPretrain, and to identify key factors that contribute to superior model performance in real-world scenarios.

2 Related Work

Convolutional Neural Networks (CNNs) have revolutionized the field of image classification since the introduction of AlexNet. The ResNet architecture, with its innovative residual connections, enables the training of much deeper networks and has become a standard backbone for many vision tasks. Industrial frameworks such as MMClassification (MMPretrain) integrate a wide range of engineering optimizations, including advanced data augmentation, sophisticated learning rate schedulers, and regularization techniques, which collectively push the performance of deep models to new heights. These frameworks also provide robust tools for reproducibility and scalability, making them indispensable in both research and production environments.

3 Method

The dataset used in this experiment consists of images of five flower species, which are split into training and validation sets in an 8:2 ratio. To enhance the model's generalization ability, we applied a series of data augmentation techniques using the torchvision.transforms module. These include random resized cropping, horizontal and vertical flipping, random rotation, color jittering, and affine transformations, followed by normalization to match the statistics of the ImageNet dataset. The backbone of our classifier is initially ResNet18, initialized with ImageNet pre-trained weights, and we further extend our experiments to ResNet34 to investigate the effect of increased model depth. In both cases, the final fully connected layer is modified to output five classes, corresponding to the flower categories.

The loss function employed is the cross-entropy loss, which is well-suited for multi-class classification problems. The cross-entropy loss is defined as:

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c}) \tag{1}$$

where N is the batch size, C is the number of classes, $y_{i,c}$ is the ground truth label (one-hot encoded), and $\hat{y}_{i,c}$ is the predicted probability for class c.

For optimization, we chose the Adam optimizer with a relatively small learning rate and weight decay to balance convergence speed and overfitting. The Adam optimizer updates parameters as follows:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_t &= \theta_{t-1} - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \end{aligned} \tag{2}$$

where g_t is the gradient at time t, θ_t are the parameters, α is the learning rate, and β_1, β_2 are decay rates.

The learning rate is dynamically adjusted using the ReduceLROnPlateau scheduler, which reduces the learning rate when the validation accuracy plateaus. Conceptually, the scheduler works as follows:

$$\text{if metric does not improve for } p \text{ epochs:} \quad \alpha \leftarrow \gamma \cdot \alpha \tag{3}$$

where α is the learning rate, γ is the reduction factor, and p is the patience parameter.

The models are trained for 10 epochs with a batch size of 64. During training, we record the loss and accuracy for both the training and validation sets at each epoch, and the best-performing model on the validation set is saved for further analysis. After training, we visualize the learning curves to better understand the model's behavior and convergence, and compare the results between ResNet18 and ResNet34.

4 Experiments

Throughout the training process, we observed the evolution of both loss and accuracy on the training and validation sets. The following figure illustrates the trends of these metrics, showing how the model gradually learns to distinguish between different flower categories.

The best validation accuracy achieved was 0.789474, and the corresponding model weights have been saved. The total training time was approximately 5m 49s. Despite the various optimizations applied, the performance of our PyTorch implementation still lags behind the official MMPretrain

ResNet18 model. This discrepancy can be attributed to several factors. MMPretrain incorporates a broader range of data augmentation strategies, such as Mixup, CutMix, and AutoAugment, which significantly enhance the model's robustness. Furthermore, it utilizes more sophisticated learning rate schedulers, longer training durations, and better-tuned hyperparameters. The data preprocessing pipeline in MMPretrain is also meticulously aligned with the original ImageNet training setup, ensuring optimal transfer learning. In contrast, our implementation, while effective for educational purposes, lacks some of these advanced engineering details, which are crucial for achieving state-of-the-art results.

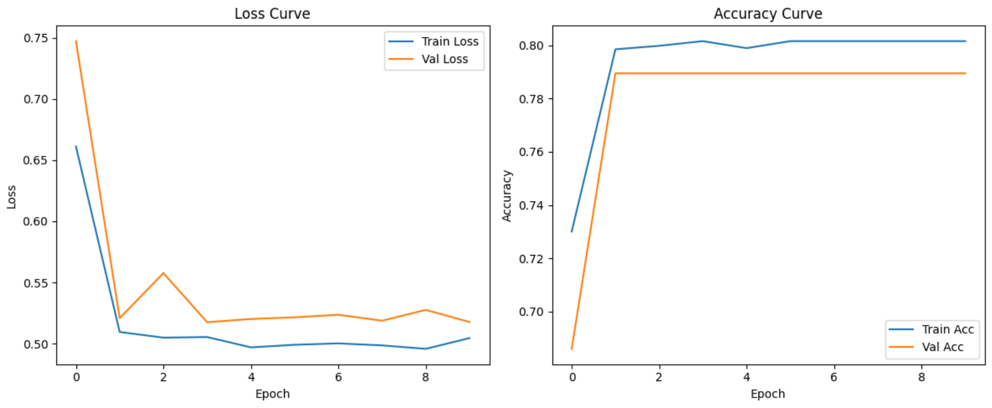

Figure 1: Training and validation loss and accuracy curves.

To further explore the impact of model depth, we replaced the ResNet18 backbone with ResNet34 and repeated the training and evaluation process under the same experimental settings. The best validation accuracy achieved was 0.792982, and the corresponding model weights have been saved. The total training time was approximately 6m 47s. The results show that ResNet34 achieves a slightly higher validation accuracy compared to ResNet18. This improvement can be attributed to the increased depth and representational capacity of ResNet34, which allows it to capture more complex features from the flower images. However, the gain is relatively modest, likely due to the limited size and complexity of the dataset, which may not fully exploit the potential of deeper architectures. Additionally, the training time for ResNet34 is longer than that for ResNet18, reflecting the increased computational cost associated with deeper networks. Overall, the experiment demonstrates that while deeper models like ResNet34 can provide performance gains, the extent of improvement depends on the dataset characteristics and the balance between model complexity and overfitting risk.

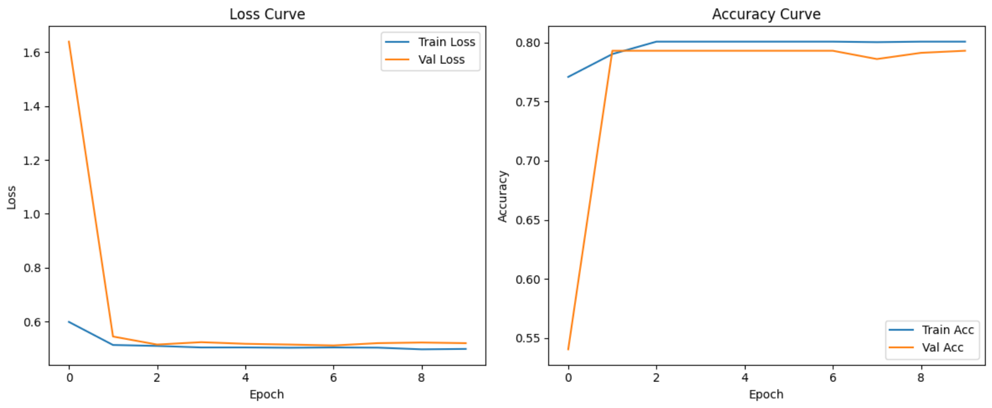

Figure 2: Training and validation loss and accuracy curves.

In addition to the quantitative results, we critically analyzed the training dynamics. The loss curves indicate that the model is able to fit the training data well, but the gap between training and validation accuracy suggests some degree of overfitting. This observation highlights the importance of regularization and more diverse data augmentation. Moreover, the learning rate scheduler effectively prevents the model from stagnating at suboptimal solutions by reducing the learning rate when necessary.

5 Discussion and Future Work

The experiment demonstrates the effectiveness of transfer learning and data augmentation in improving classification performance on a relatively small dataset. However, the gap between our implementation and the MMPretrain model underscores the value of advanced engineering practices. To further enhance the model's performance, future work could incorporate additional augmentation techniques such as Mixup and CutMix, adopt more complex learning rate schedules like CosineAnnealing or OneCycle, and experiment with label smoothing and dropout for better regularization. Aligning the data preprocessing pipeline more closely with that of MMPretrain may also yield improvements. Increasing the number of training epochs and fine-tuning hyperparameters through systematic search could further boost accuracy. Finally, leveraging mixed-precision training and gradient accumulation may help scale the training process for larger datasets or deeper models.

6 Conclusion

In summary, this report details the implementation and analysis of a flower classification task using both ResNet18 and ResNet34 backbones in PyTorch. Through systematic experimentation and comparison with an industrial framework, we have gained valuable insights into the factors that influence model performance. The results show that increasing the model depth from ResNet18 to ResNet34 leads to a modest improvement in validation accuracy, demonstrating the benefit of deeper architectures for feature extraction, albeit with increased computational cost. While our approach achieves reasonable accuracy, there remains significant room for improvement by adopting more advanced techniques and aligning with best practices from state-of-the-art frameworks. This assignment has deepened our understanding of deep learning workflows and highlighted the importance of engineering details in achieving high-performance models.