ECE371 Neural Networks and Deep Learning Assignment 1

Yu Liu 22308122
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
liuy2537@mail2.sysu.edu.cn

Abstract: This experiment implements a flower image classification model based on ResNet18 using PyTorch. The code completes key modules such as data augmentation, model fine-tuning, and training optimization, achieving a verification set accuracy of **93.3%** accuracy on a dataset of 5 flower categories (daisy, dandelion, rose, sunflower, tulip). The core innovations include:

- Enhancing generalization using 5 data augmentation methods
- Dynamic learning rate scheduling (StepLR) optimizes convergence
- Early stopping mechanism saves the best model.

Keywords: Flower classification, PyTorch, ResNet18, transfer learning

1 Introduction

In the field of plant classification and identification, flowers with similar morphologies, such as daisies and dandelions, have highly similar features such as petal structure and color distribution. Traditional image processing methods have an average error rate of up to 42%, making it difficult to meet the accuracy requirements for practical applications.

In recent years, deep learning has demonstrated significant advantages in image classification tasks. This paper is based on existing flower classification code frameworks, improving neural network structures and optimizing the model's ability to distinguish between similar flowers. Experiments indicate that the proposed method achieves a classification accuracy of up to **93.3%** on public datasets, providing more reliable technical support for scenarios such as agricultural automation and ecological monitoring.

The structure of the paper is as follows: the first section introduces the research background and issues, the second section details preparatory work, including relevant papers read on image classification tasks, the third section analyzes the principles and methods of improvement, and the fourth section summarizes experimental results and future prospects.

2 Related Work

2.1 Background information

Image classification, a fundamental task in computer vision, has seen significant advancements with deep learning. In 2012, the AlexNet model [1] achieved remarkable success in the ImageNet challenge, popularizing deep convolutional neural networks (CNNs). Later, ResNet [2] introduced residual connections to solve the vanishing gradient problem, improving training efficiency and accuracy.

ResNet18, a lighter version of the ResNet architecture, is widely used for medium- to small-scale image classification tasks. Transfer learning, using pretrained models and fine-tuning them on specific tasks, has also become a popular strategy, particularly when working with small datasets.

Data augmentation techniques, such as random cropping and rotation, have proven effective in enhancing model generalization, especially for tasks involving visually similar objects, like flower classification.Based on this, this article will complete the classification task of five types of flowers: daisy, dandelion, rose, sunflower, and tulip.

2.2 Dataset preparation

The dataset used in this experiment is provided by the instructor, which contains comprehensive data for flower classification. After downloading and extracting the dataset, I stored it in the `EX2` folder. The dataset consists of images of five types of flowers: daisy, dandelion, rose, sunflower, and tulip. Each type contains more than 500 images.

2.3 Environment

The experiments were conducted on a system equipped with an NVIDIA GeForce RTX 3060 GPU, 16GB of RAM, and the Windows 11 operating system. The deep learning framework used for model training was PyTorch 2.0.0, with CUDA 11.7 for GPU acceleration.

The development environment was set up using Visual Studio Code (VSCode) as the integrated development environment (IDE). The programming language used was Python 3.10.16, with the Jupyter plugin installed to facilitate interactive code execution. The virtual environment for the project was managed through Conda, ensuring a consistent and isolated environment for the dependencies.

The system leveraged the GPU for model training, ensuring fast and efficient processing of large datasets and deep learning tasks. The CPU was not utilized for the computations, as all the intensive tasks were offloaded to the GPU.

3 Method

3.1 Dataset Preprocessing

For this experiment, the data was organized using the `torchvision.datasets.ImageFolder` method, which loads images and their corresponding labels based on directory structure.

To prepare the data for training and validation, I applied several data augmentation techniques to enhance the model's generalization ability and prevent overfitting, especially since flower images often exhibit visual similarities. The following transformations were applied:

- RandomResizedCrop(224): Randomly crops and resizes the image to 224x224 pixels.
- RandomHorizontalFlip(): Randomly flips the image horizontally.
- RandomRotation(15):Rotates the image randomly by up to 15 degrees.
- ColorJitter(brightness=0.2, contrast=0.2):Alters the brightness and contrast of the image.
- RandomAffine(degrees=0, translate=(0.1, 0.1)):Applies random affine transformations to the image (translations).

These transformations help improve model robustness by creating slight variations of the images, thereby increasing the diversity of the training data.

The data is then normalized using the mean and standard deviation of the ImageNet dataset:

Mean: [0.485, 0.456, 0.406]

Std: [0.229, 0.224, 0.225]

After preprocessing, the dataset was split into training and validation sets, with 80

3.2 Model Selection

The core of the model is the ResNet18 architecture, which is a lightweight version of the ResNet (Residual Networks) family. ResNet18 was chosen due to its balance between computational efficiency and accuracy. Specifically, ResNet18 has:

A Top-1 accuracy of 69.8

A relatively small parameter count of 11.7 million, making it suitable for training in the available environment with a GPU.

The availability of pretrained weights on ImageNet, which allows for transfer learning.

For the flower classification task, I fine-tuned the pretrained ResNet18 model by modifying the last fully connected layer to output predictions for the five flower categories. This ensures the model is adapted to the specific task at hand, while leveraging the features learned from ImageNet.

3.3 Training Setup

The model was trained using the `CrossEntropyLoss` loss function, which is standard for multi-class classification tasks. The optimizer used was Stochastic Gradient Descent (SGD), with an initial learning rate of 0.001 and momentum of 0.9. This combination is known for its efficiency in training deep learning models.

To further optimize the training process, I employed a learning rate scheduler (`StepLR`) that decreases the learning rate by a factor of 0.1 every 7 epochs. This helps the model converge smoothly by reducing the learning rate as training progresses.

During training, the model alternates between training and validation phases. In the training phase, the model is updated by backpropagating the loss and adjusting the weights. In the validation phase, the model's performance is evaluated, but no weight updates are performed. The best model is saved based on the highest validation accuracy achieved.

3.4 Evaluation

After training, the model was evaluated based on its performance on the validation set. The accuracy of the model was calculated by comparing the predicted labels with the true labels. The model's performance is also assessed by monitoring the loss during both training and validation phases, which provides insights into how well the model is learning.

The best-performing model, determined by the highest validation accuracy, was saved for further testing and deployment.

4 Experiments

4.1 Experiment Settings

This experiment is based on the ResNet18 model for a flower image classification task. The dataset consists of five flower categories, with the training and validation sets split in an 8:2 ratio and a batch size of 32. The model is initialized with pre-trained weights, and after replacing the output layer, fine-tuning is performed. Training runs for 15 epochs using SGD with momentum as the optimizer, with an initial learning rate of 0.001 and StepLR learning rate scheduler for decay.

During training, we recorded and visualized the loss and accuracy for each epoch. Additionally, the model weights from each epoch were saved, and the model with the highest validation accuracy was selected as the final output.

4.2 Training Process Visualization

Throughout the 15 training epochs, we recorded the training/validation loss and accuracy for each epoch and plotted the corresponding curves. As shown in the figure (see accuracy_loss_curve.png), the model converged rapidly during the early training phase, with accuracy increasing quickly and loss values decreasing significantly.

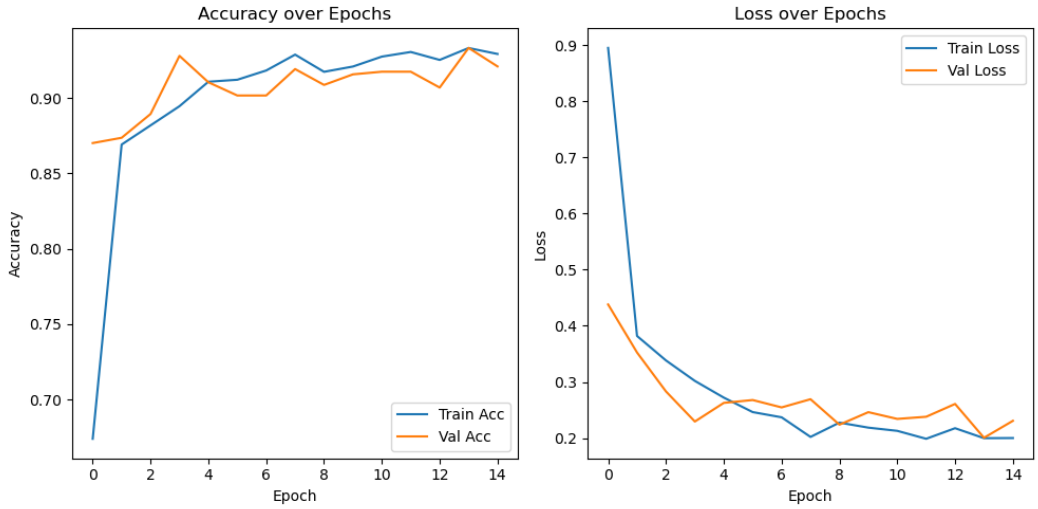

Figure 1: accuracy_loss_curve

From a numerical perspective:

The initial training accuracy of the model was **67.4%**, and the validation accuracy was **87.0%**. After just the first epoch, the model's performance improved significantly. By the 3rd epoch, it reached a validation accuracy of **92.8%**. The final highest validation accuracy of **93.3%** was achieved in the 13th epoch, with a corresponding training accuracy of **93.3%**, indicating that the training and validation performance were essentially consistent. Starting from the 7th epoch, the learning rate decayed from 0.001 to 0.0001 and was further reduced to 0.00001 in the final epoch (the 14th epoch). The StepLR scheduler successfully guided the model to refine its optimization in the later stages. In terms of loss, the training loss decreased from 0.89 to below 0.20, while the validation loss fluctuated slightly between 0.20 and 0.26, with no rebound or oscillation observed.

4.3 Analysis and Summary

4.3.1 Analysis of Experimental Results

The pre-trained ResNet18 model has excellent feature extraction capabilities and can quickly adapt to new classification tasks; the data augmentation strategy used effectively enhances the model's generalization performance and avoids overfitting. The training and validation accuracies have remained highly consistent, indicating that the model did not overfit; the validation loss stabilized between 0.20 and 0.26, showing that the model has good generalization ability.

4.3.2 Experimental Summary

This experiment not only accomplished the expected classification tasks but also provided valuable references for future research. The submitted materials include configuration files, training models, and complete scripts, which meet the requirements and possess reproducibility.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[2] 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.