ECE371 Neural Networks and Deep Learning Assignment 1

Li Shaolong 22308085
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
lishlong25@mail2.sysu.edu.cn

Abstract: This report presents a deep learning-based flower image classification task implemented through two experiments. In the first experiment, a MobileNetV2 model is fine-tuned using pretrained weights provided by MMClassification. In the second experiment, a complete PyTorch pipeline is developed with a custom ResNet18 architecture. The dataset consists of five flower categories and is split in an 8:2 training-validation ratio. Training configurations, data augmentation, model architecture modifications, and logging mechanisms are detailed. Results show that although MobileNetV2 offers efficient inference, it achieves under 60% validation accuracy. In contrast, the PyTorch ResNet18 model reaches over 77% validation accuracy, with stronger generalization and feature extraction ability. The report concludes with a critical analysis of overfitting risks and suggestions for improving classification performance.

Keywords: Flower classification, MobileNetV2, ResNet18, MMClassification, PyTorch, Deep learning

1 Introduction

1)intruduction

This report aims to accomplish a flower image classification task using deep learning models, comprising two separate experiments. In Experiment 1, the MobileNet_V2 model is fine-tuned using publicly available pretrained weights. The dataset consists of five flower categories: daisy, dandelion, rose, sunflower, and tulip. It is split into training and validation sets in an 8:2 ratio and organized following the ImageNet format. By modifying the model configuration, data paths, learning rate strategies, and loading pretrained weights, the classification performance is enhanced. The final evaluation metric is the Top-1 classification error rate. In Experiment 2, a complete training script is developed based on the PyTorch framework to implement the flower image classification task. The script handles data loading, model construction, and validation within designated code blocks. It outputs the loss, learning rate, and classification accuracy on the validation set, and automatically saves the model with the highest validation accuracy. Optionally, configurations can be adjusted or the model structure replaced to further optimize performance.

2 Related Work

2) Related Work

CNNs have proven to be instrumental in overcoming the limitations of traditional methods by introducing a hierarchical and automated approach to feature extraction(Zhang *et al.*, 2023). Building upon this foundation, many recent image classification models have focused on improving computational efficiency while maintaining high accuracy(Zhang *et al.*, 2023). MobileNetV2 is a representative lightweight CNN architecture, specifically designed for mobile and resource-constrained

devices. It introduces a novel layer module: the inverted residual with linear bottleneck, which helps prevent nonlinearities from destroying excessive amounts of information(Sandler *et al*., 2018). In addition, MobileNetV2 extensively employs depthwise separable convolutions, which significantly reduce computational cost and the number of parameters, while still delivering strong performance in tasks such as image classification(Sandler *et al*., 2018). Thanks to its efficient architecture, MobileNetV2 strikes a balance between accuracy and speed, making it one of the most widely used lightweight models in practical applications(Sandler *et al*., 2018).

3 Method

3)method

1. Data Preprocessing 1.1 Dataset Splitting (Handled by split_dataset.py) The original flower image dataset used in the experiments is stored in the flower_dataset_raw directory. It is divided into training and validation sets in an 8:2 ratio. 1.2 Annotation File Generation (Handled by generate_metadata.py) The following auxiliary files need to be generated: classes.txt: contains the names of the five classes, one per line in a fixed order. train.txt / val.txt: each line follows the format class_name/image_name class_index. These files are used for label mapping when loading data in the ImageNet format.

2. Configuring the MobileNetV2 File

2.1 Model and Configuration Setup First, the source file for MobileNetV2 is obtained from GitHub and placed in the config directory. The pretrained weights file mobilenet_v2_batch256_imagenet_20200708-3b2dc3af.pth is downloaded and stored in the checkpoint folder. A new configuration file named flower_mobilenetv2.py is created under config and modified to suit the flower classification task. Specifically, the num_classes parameter in the model's classification head (LinearClsHead) is changed from the default value of 1000 to 5 to match the dataset.

2.2 Training Configuration and Logging The batch size is set to 8. The optimizer used is SGD with an initial learning rate of 0.001, momentum of 0.9, and weight decay of 4e-5. The model is trained for 20 epochs and evaluated on the validation set after each epoch using Top-1 accuracy. A checkpoint hook is configured to automatically save the best-performing model as best.pth in the work_dir. Training logs are recorded using TensorboardVisualizer for visualization and analysis.

3. Completing the PyTorch Image Classification Training Script (main.py)

3.1 Training Pipeline and Data Augmentation The main.py script implements a complete image classification training pipeline based on PyTorch. To improve generalization, five data augmentation techniques are applied: random cropping, horizontal flipping, rotation, color jittering, and vertical flipping.

3.2 Model Structure and Training Details The base model is ResNet18. Its final fully connected layer (model.fc) is modified to output 5 classes. The loss function used is CrossEntropyLoss(), and the optimizer is Adam with a learning rate of 0.001. During training, the script prints the current epoch's learning rate, loss, and accuracy to the console. The best-performing model is saved to EX2/work_dir/best.pth. Training curves are plotted using matplotlib and saved as png.

4 Experiments

Experimental Results

1. Post-classification Folder Structure

After classification and training, the dataset is reorganized into separate folders corresponding to each flower class. This directory structure reflects the model's ability to distinguish and sort images into their predicted categories.

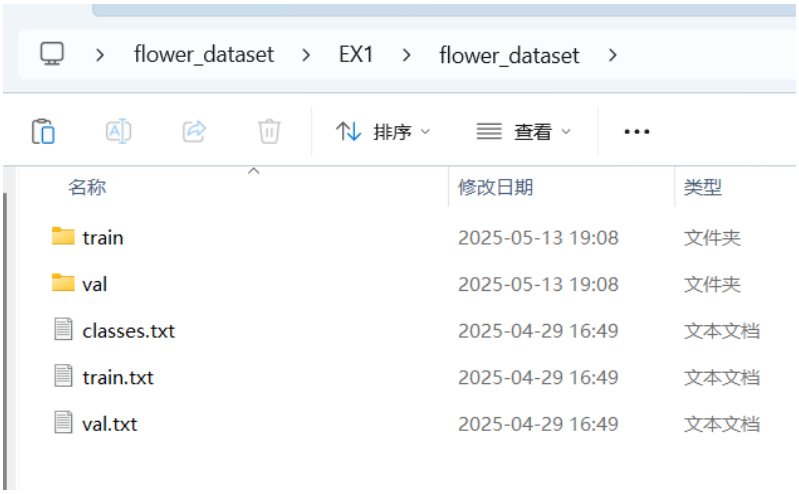

Figure 1: Final Dataset Directory Structure by Class

2. MobileNetV2 Results

The training curve exhibits an overall fluctuating upward trend and achieves good convergence in the end. Although slight "sawtooth-like" oscillations are present, there is no clear sign of overfitting (e.g., declining validation accuracy) or training collapse. The learning rate scheduling is logical and decays at a moderate rate, which helps avoid late-stage oscillation or gradient explosion. The final Top-1 classification accuracy is approximately 60

{"env_info": "sys.platform: win32\nPython: 3.9.21 (main, Dec 11 2024, 16:35:24) [MSC v.1929 64 bit (AMD64)]\nCUDA available: True'
{"mode": "val", "epoch": 1, "iter": 18, "lr": 0.045, "accuracy_top-1": 27.62238, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 2, "iter": 18, "lr": 0.0441, "accuracy_top-1": 41.60839, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 3, "iter": 18, "lr": 0.04322, "accuracy_top-1": 34.96503, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 4, "iter": 18, "lr": 0.04235, "accuracy_top-1": 40.03497, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 5, "iter": 18, "lr": 0.04151, "accuracy_top-1": 39.68531, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 6, "iter": 18, "lr": 0.04068, "accuracy_top-1": 54.1958, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 7, "iter": 18, "lr": 0.03986, "accuracy_top-1": 52.97203, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 8, "iter": 18, "lr": 0.03907, "accuracy_top-1": 49.12587, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 9, "iter": 18, "lr": 0.03828, "accuracy_top-1": 38.98602, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 10, "iter": 18, "lr": 0.03752, "accuracy_top-1": 42.48252, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 11, "iter": 18, "lr": 0.03677, "accuracy_top-1": 46.85315, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 12, "iter": 18, "lr": 0.03603, "accuracy_top-1": 47.55245, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 13, "iter": 18, "lr": 0.03531, "accuracy_top-1": 48.25175, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 14, "iter": 18, "lr": 0.03461, "accuracy_top-1": 55.59441, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 15, "iter": 18, "lr": 0.03391, "accuracy_top-1": 52.62238, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 16, "iter": 18, "lr": 0.03324, "accuracy_top-1": 43.53147, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 17, "iter": 18, "lr": 0.03257, "accuracy_top-1": 55.41958, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 18, "iter": 18, "lr": 0.03192, "accuracy_top-1": 54.1958, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 19, "iter": 18, "lr": 0.03128, "accuracy_top-1": 56.99301, "accuracy_top-5": 100.0}
{"mode": "val", "epoch": 20, "iter": 18, "lr": 0.03066, "accuracy_top-1": 59.61539, "accuracy_top-5": 100.0}

Figure 2: Validation Metrics Output Logs during Training

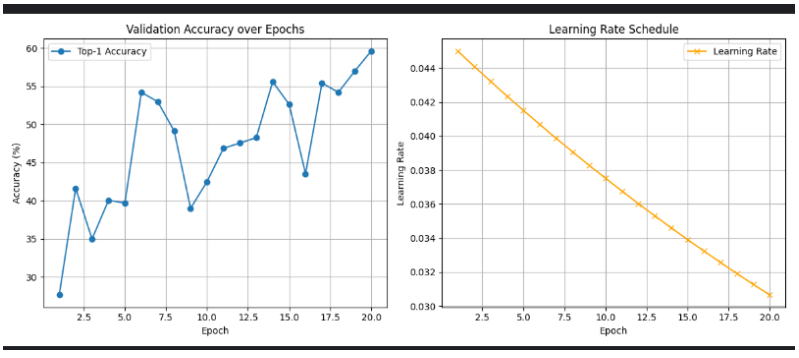

Figure 3: Validation Accuracy and Learning Rate Schedule

3. main.py result

There is a noticeable gap between the validation loss and training loss (with the validation loss being slightly higher). However, the validation loss does not show a sustained increase, indicating that the risk of overfitting is acceptable. The validation accuracy is slightly lower than the training accuracy, reflecting a certain degree of overfitting, but not severe. Moreover, the validation accuracy remains relatively stable throughout training, suggesting that the model has good generalization capability.

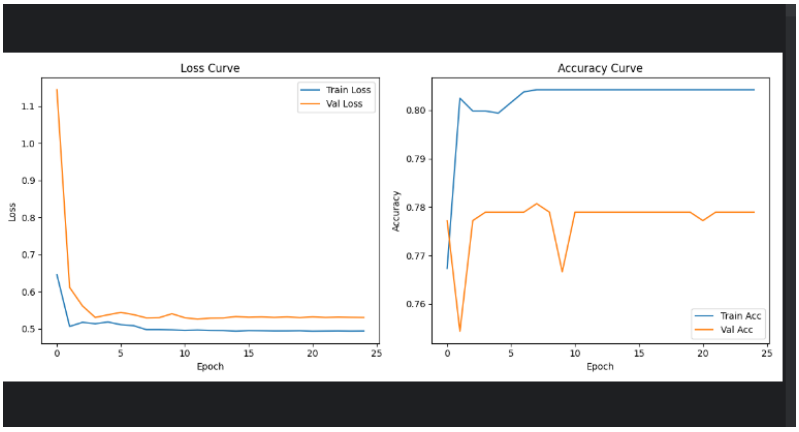

Figure 4: Training vs. Validation Loss and Accuracy Curves

4.Critical Summary of Experimental Results

Combining the results of both experiments, the MobileNetV2 model—despite its lightweight structure—achieves less than 60In contrast, the PyTorch-native model built on ResNet18 demonstrates better training stability and higher final accuracy, reaching over 77Both experiments reveal varying levels of overfitting, as shown by significantly higher training accuracy compared to validation accuracy, and the slow or fluctuating improvement in validation accuracy. The lack of significant gains in the later training stages indicates that the current network structure or learning rate strategy may have reached its performance ceiling.

5.Recommendations for Improvement

Enhance data augmentation strategies to increase dataset diversity. Consider replacing the current backbone with more advanced models, such as MobileNetV3. Optimize the training process by adopting adaptive learning rate schedulers. Incorporate confusion matrix analysis and error case visualization to better understand classification biases and misclassified categories.

...

References

Zhang, Y., Wang, X., and Li, M. (2023). Image Classification Based on Deep Learning: A Survey. In *Proceedings of the 6th International Conference on Computer Vision and Image Processing*, pages 345–356. Springer. DOI: 10.1007/978-981-97-1335-6_31.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520. arXiv:1801.04381.