ECE371 Neural Networks and Deep Learning Assignment 1

Zhang Jinbin, 22308242
School of Electronics and Communication Engineering
Sun Yat-sen University, Shenzhen Campus
zhangjb73@mail2.sysu.edu.cn

Abstract: This report details the implementation of a flower image classification task using the PyTorch framework. By adopting a pre-trained ResNet18 model and incorporating data augmentation techniques, we successfully classified a dataset containing five types of flowers: daisy, dandelion, rose, sunflower, and tulip. The experimental results show that the fine-tuned model achieved a classification accuracy of 91.40% on the validation set, demonstrating the effectiveness of deep convolutional neural networks in image classification tasks. It also confirms the importance of data augmentation and transfer learning in enhancing model performance.

Keywords: Convolutional Neural Network, Transfer Learning, Image Classification, Data Augmentation, ResNet

1 Introduction

Image classification is a fundamental problem in the field of computer vision. With the advancement of deep learning, Convolutional Neural Networks (CNNs) have shown remarkable advantages in image classification tasks. This assignment aims to apply deep learning techniques, specifically a pre-trained ResNet18 model, to solve a flower classification problem. Using a dataset of five flower categories (daisy, dandelion, rose, sunflower, and tulip), we implemented a high-accuracy classification model through transfer learning and data augmentation.

The key challenge of the task lies in effectively training a model to distinguish between different types of flowers, which may have similarities in color, shape, and texture, thus increasing classification difficulty. Through this experiment, we will gain a deeper understanding of the working principles of CNNs, the application of transfer learning, and the impact of data augmentation on model performance.

2 Related Work

The application of deep learning in image classification has become a research hotspot in recent years. Since AlexNet [1] made a breakthrough in the ImageNet challenge, deep CNNs have rapidly evolved in the computer vision field.

VGGNet [2] improved model performance by using deeper network structures and small-sized convolution kernels (3×3). GoogLeNet [3] introduced the Inception module, which extracts multi-scale features by using different-sized convolution kernels in parallel. ResNet [4] proposed residual connections, effectively solving the vanishing gradient problem in deep network training, making much deeper networks possible and significantly boosting classification performance.

Transfer learning [5] is another important research direction. By utilizing models pre-trained on large-scale datasets (like ImageNet), the learned features can be transferred to new tasks, which can significantly reduce training time and improve performance on smaller datasets. The study by

Yosinski et al. [6] showed that the lower-level features of a pre-trained model are generic, while the higher-level features are more task-specific.

Data augmentation is an effective method for improving a model's generalization ability. Shorten and Khoshgoftaar [7] reviewed various data augmentation techniques, including geometric transformations (e.g., flipping, rotation, cropping) and photometric transformations (e.g., changing brightness, contrast, saturation). These techniques can increase the diversity of training samples, reduce overfitting, and enhance model robustness.

This assignment uses ResNet18 as the base model, combined with transfer learning and multiple data augmentation techniques, to achieve a high-performance flower classification system.

3 Method

3.1 Dataset and Preprocessing

The flower dataset used in this experiment contains five categories: daisy (588 images), dandelion (556 images), rose (583 images), sunflower (536 images), and tulip (585 images). We split the dataset into training and validation sets with an 8:2 ratio. To improve the model's generalization ability, I applied several data augmentation techniques:

1. **RandomResizedCrop:** Randomly crops the image and resizes it to 224×224 pixels, increasing the model's robustness to variations in object location and scale.
2. **RandomHorizontalFlip:** Horizontally flips the image with a 50% probability, making the model independent of the object's orientation.
3. **RandomRotation:** Randomly rotates the image within a ±30 degree range, enhancing the model's invariance to rotational transformations.
4. **ColorJitter:** Randomly adjusts the brightness, contrast, saturation, and hue, enabling the model to better adapt to different lighting conditions.
5. **RandomAffine:** Applies shear and scale transformations, further increasing the diversity of the data.

Furthermore, the images were normalized using the mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] of the ImageNet dataset. This helps to accelerate model convergence and improve performance.

3.2 Network Model

This experiment employed a pre-trained ResNet18 model, which has learned rich feature representations from the ImageNet dataset. The core innovation of ResNet (Residual Network) is the introduction of the Residual Block, whose mathematical expression is:

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{1}$$

where $\mathcal{F}(x, \{W_i\})$ represents the residual mapping to be learned, and x is the identity mapping. This structure allows gradients to be directly passed to shallower layers through the identity mapping, effectively mitigating the vanishing gradient problem in deep networks.

ResNet18 consists of 18 layers, composed of multiple residual blocks. To adapt it for the flower classification task, the model's fully connected layer was modified, changing the output dimension from the original 1000 (ImageNet classes) to 5 (flower classes): ...

$$\text{model.fc} = \text{nn.Linear(num\_ftrs, num\_classes)} \tag{2}$$

where `num_ftrs` is the feature dimension (512), and `num_classes` is the number of classes (5).

3.3 Loss Function and Optimizer

We chose Cross-Entropy Loss as the loss function, which is suitable for multi-class classification problems. It is defined as:

$$L(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{3}$$

where y is the one-hot encoded true label, $\hat{y}$ is the model's predicted probability distribution, and C is the number of classes.

The optimization algorithm used is Stochastic Gradient Descent (SGD) with momentum. Momentum can accelerate convergence and help escape local optima:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \tag{4}$$

$$\theta = \theta - v_t \tag{5}$$

where $\gamma = 0.9$ is the momentum coefficient, $\eta = 0.001$ is the learning rate, and $\nabla_\theta J(\theta)$ is the gradient of the loss function with respect to the parameters θ.

A learning rate scheduler (StepLR) was also used, which multiplies the learning rate by 0.1 every 7 epochs. This helps to refine the parameter updates in the later stages of training:

$$\eta_t = \eta_0 \times \gamma^{\lfloor t/s \rfloor} \tag{6}$$

where η_0 is the initial learning rate, $\gamma = 0.1$ is the decay factor, $s = 7$ is the step size, and t is the current epoch.

3.4 Training Process

The training process implemented the following key functions:

1. Moves the model to the appropriate device (GPU or CPU).
2. Loops through training and validation phases for each epoch.
3. In the training phase: performs forward propagation, loss calculation, backpropagation, and parameter updates.
4. In the validation phase: evaluates model performance.
5. Saves the model with the highest validation accuracy.
6. Monitors changes in learning rate, loss, and accuracy.

Backpropagation is the core algorithm of deep learning, used to compute the gradient of the loss function with respect to the model parameters. Based on the chain rule, gradients are propagated backward from the output layer to the input layer. After computing the gradients, the optimizer updates the model parameters based on the calculated gradients, progressively minimizing the loss function.

4 Experiments

4.1 Experimental Setup

The flower classification model based on ResNet18 was implemented using the PyTorch framework. The main hyperparameters for the experiment are set as follows:

- **Batch size:** 32
- **Initial learning rate:** 0.001
- **Momentum coefficient:** 0.9
- **Learning rate scheduler:** Decays by a factor of 0.1 every 7 epochs. ...

- **Epochs:** 25

The model was trained on a GPU (if available), otherwise on a CPU. Data parallelism was used by setting `num_workers=4` to speed up data loading.

4.2 Results and Analysis

Figure 1 shows the model's performance during the training process.

```
Epoch 22/24
----------
Learning Rate: 0.000001
train Loss: 0.2655 Acc: 0.9074
val Loss: 0.2605 Acc: 0.8982

Epoch 23/24
----------
Learning Rate: 0.000001
train Loss: 0.2457 Acc: 0.9192
val Loss: 0.2760 Acc: 0.9035

Epoch 24/24
----------
Learning Rate: 0.000001
train Loss: 0.2290 Acc: 0.9144
val Loss: 0.2600 Acc: 0.9053

Training complete in 20m 18s
Best val Acc: 0.914035
```

Figure 1: Learning rate, loss, and accuracy changes during model training (last 3 epochs).

From the experimental results, it is clear that the model achieved good performance on both the training and validation sets:

- **At epoch 24 (final epoch):**
  - Training set loss: 0.2290, Accuracy: 91.44%
  - Validation set loss: 0.2600, Accuracy: 90.53%
- **Best validation accuracy:** 91.40%
- **Total training time:** 20 minutes 38 seconds

The small gap between the training and validation accuracy indicates that the model did not suffer from severe overfitting, and the data augmentation techniques effectively improved the model's

generalization ability. The learning rate scheduling strategy also effectively helped the model find a better parameter configuration in the later stages of training.

By observing the training process, we can find:

1. In the first few epochs, the model learns rapidly, with the loss decreasing quickly and accuracy improving significantly.
2. As the learning rate decays, the speed of loss reduction and accuracy improvement slows down, but the model's performance continues to improve steadily.
3. In the later stages of training, the validation accuracy fluctuates slightly but generally remains at a high level, indicating that the model has good stability.

These results demonstrate that the methodology—combining a pre-trained model, transfer learning, data augmentation, and appropriate optimization strategies—is effective for the flower classification task.

5 Conclusion

This experiment successfully implemented a flower image classification system based on ResNet18. By leveraging transfer learning and data augmentation, an accuracy of 91.40% was achieved on the five-class flower dataset. The experimental results show that:

1. Pre-trained models can be effectively applied to specific domain image classification tasks through transfer learning.
2. A combination of multiple data augmentation techniques can significantly enhance a model's generalization ability and reduce overfitting.
3. Appropriate optimization strategies and learning rate scheduling are crucial for model convergence and performance.

Future work could consider trying deeper network architectures (such as ResNet50, ResNet101) or other advanced architectures (like EfficientNet, Vision Transformer) to further improve classification performance. Additionally, exploring more data augmentation techniques and regularization methods could further enhance the model's robustness and generalization capabilities.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

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

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[4] 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.

[5] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, volume 27, 2014.

[7] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.