ECE371 Neural Networks and Deep Learning Assignment 1

Peng Kairan 22308143
School of Electronics and Communication Engineering
Sun Yat - sen University, Shenzhen Campus
pengkr5@mail2.sysu.edu.cn

Abstract: This report presents the implementation of a flower classification model using transfer learning with a pre-trained ResNet18. The model is adapted to classify 5 flower categories by modifying the final fully-connected layer. Data augmentation techniques and hyperparameter tuning are employed to enhance generalization. After 25 training epochs, the model achieves a best validation accuracy of 91.75%.

Keywords: Transfer Learning; Residual Network; Data Augmentation

1 Introduction

The objective of this assignment is to develop a robust flower classification model using a pre - trained neural network. We utilize ResNet18 as the backbone network and adapt it to the target task through transfer learning. Key steps include data preprocessing, data augmentation strategies, and hyperparameter optimization.

2 Related Work

In the domain of image classification, transfer learning has emerged as a pivotal technique, leveraging features learned by pre-trained models on large-scale datasets like ImageNet. This approach significantly reduces training time and resource requirements. The ResNet (Residual Network) architecture, notably ResNet18, addresses the vanishing gradient problem in deep networks through its residual blocks, enabling training of deeper networks. ResNet18's balance of complexity and performance makes it widely applicable in various visual tasks. For flower classification, prior studies have combined data augmentation methods (such as rotation, flipping) to enhance model generalization and adjusted hyperparameters to optimize performance. These works validate the effectiveness of transfer learning, data augmentation, and hyperparameter tuning in fine-grained image classification, providing a methodological foundation for this assignment—employing the pre-trained ResNet18 model, applying data augmentation to the flower dataset, and optimizing hyperparameters to achieve accurate classification.

3 Methodology

3.1 3.1 Dataset and Preprocessing

The flower dataset consists of 3,670 images across 5 classes, split into 2,936 training images and 734 validation images. To mitigate overfitting, the following data augmentation techniques are applied: random resized cropping (224×224), random horizontal/vertical flipping, color jittering (brightness/contrast/saturation ±20%), and random rotation (±20°). Images are normalized using ImageNet statistics (mean = [0.485, 0.456, 0.406], standard deviation = [0.229, 0.224, 0.225]) and converted to tensors.

, .

3.2 3.2 Model Architecture

Using ResNet18 as the backbone, we retain all convolutional layers and replace only the final fully - connected layer. The backbone outputs 512 - dimensional features via adaptive average pooling, which are fed into a new linear layer ‘fc = nn.Linear(512, 5)‘ to map to the 5 flower classes.

3.3 3.3 Training Configuration

The cross - entropy loss is used as the loss function. The optimizer is Stochastic Gradient Descent (SGD) with parameters: learning rate 0.001, momentum 0.9, and weight decay 1e - 4. A StepLR learning rate scheduler (decaying lr by 10% every 7 epochs) is employed. The model is trained for 25 epochs with a batch size of 32 on an NVIDIA GPU.

4 Experiments

4.1 4.1 Evaluation Metrics

Model performance is evaluated using accuracy on both training and validation sets.

4.2 4.2 Key Results

During training, the training loss decreases from 0.91 in epoch 0 to 0.20 in epoch 24, with validation loss stabilizing around 0.25. The training accuracy reaches 92.76% in epoch 24, while the validation accuracy peaks at 91.75% in epochs 3 and 16. At epoch 5, training accuracy is 90.12% and validation accuracy is 88.95%; at epoch 15, training accuracy is 92.19% and validation accuracy is 89.30%. Data augmentation significantly improves model generalization and reduces overfitting. The learning rate scheduler adjusts the learning rate every 7 epochs, contributing to stable convergence and performance improvement.