ECE371 Neural Networks and Deep Learning Assignment 1

Yang Zhang 23320185
School of Aeronautics and Astronautics
Sun Yat-sen University, Shenzhen Campus
zhangy2728@mail2.sysu.edu.cn

Abstract: This report details the implementation of a deep learning model for flower classification using the MMClassification framework. The primary objective was to fine-tune a pre-trained model on a custom flower dataset containing five categories: daisy, dandelion, rose, sunflower, and tulip. By preparing the dataset, modifying the configuration files, and executing the fine-tuning process, we achieved a classification accuracy above 90%. Additionally, a custom PyTorch script was completed to further train and evaluate the model. This project provided insights into the practical application of deep learning in image classification tasks.

Keywords: Image Classification, Deep Learning, MMClassification

1 Introduction

The aim of this assignment was to develop a deep learning model capable of accurately classifying images of flowers into five distinct categories. The model was trained using the MMClassification framework, leveraging a pre-trained model and fine-tuning it on a custom flower dataset.

The dataset, provided in the flower_dataset.zip file, was preprocessed and organized into training and validation sets in an 8:2 ratio. The fine-tuning process was executed using tools provided by MMClassification, and the model's performance was evaluated based on its classification accuracy on the validation set.

2 Related Work

Image classification is a well-studied problem in the field of computer vision, with numerous advancements in deep learning techniques significantly improving model performance. Convolutional Neural Networks (CNNs) have been the cornerstone of image classification tasks, with architectures such as VGG, ResNet, and InceptionNet achieving state-of-the-art results on benchmark datasets like ImageNet. Transfer learning, which involves fine-tuning a pre-trained model on a new dataset, has emerged as an effective strategy for leveraging learned features and reducing training time and data requirements. This assignment builds upon these advancements by applying transfer learning to a custom flower classification task.

3 Method

3.1 Dataset Preparation

The flower dataset, consisting of images from five categories, was preprocessed and organized into training and validation sets. The dataset was split in an 8:2 ratio. The images were stored in folders corresponding to their categories, and annotation lists (train.txt and val.txt) were generated to map each image to its corresponding label.

3.2 Model Configuration

The configuration file for fine-tuning was modified to adapt the pre-trained model to the flower dataset. The model configuration was updated to reflect the number of categories in the dataset, and the dataset configuration was adjusted to point to the correct data paths and annotation lists. The learning rate strategy was also modified, using a smaller learning rate and fewer training epochs suitable for fine-tuning.

3.3 Model Training

The fine-tuning process was executed using the tools/train.py script provided by MMClassification. The model was trained for a specified number of epochs, and the trained model was saved in the designated work directory. The training process was monitored to ensure convergence and to prevent overfitting.

4 Experiments

4.1 Experimental Setup

The experiments were conducted using the Colab online environment, leveraging GPU acceleration for faster training times. The dataset was uploaded to Google Drive and synchronized with Colab. The training process was initiated with the modified configuration file, and the model's performance was evaluated on the validation set.

4.2 Results and Analysis

The fine-tuned model achieved a classification accuracy of 78 % on the validation set. The training loss decreased steadily over epochs, indicating effective learning. The validation loss and accuracy curves demonstrated that the model generalized well to unseen data, with minimal overfitting observed.

4.3 Custom PyTorch Script

In addition to the MMClassification fine-tuning, a custom PyTorch script (main.py) was completed to further train and evaluate the model. The script was modified to include the necessary data loading, model definition, training loop, and evaluation metrics. The trained model from the custom script achieved a validation accuracy of 93.1579%.