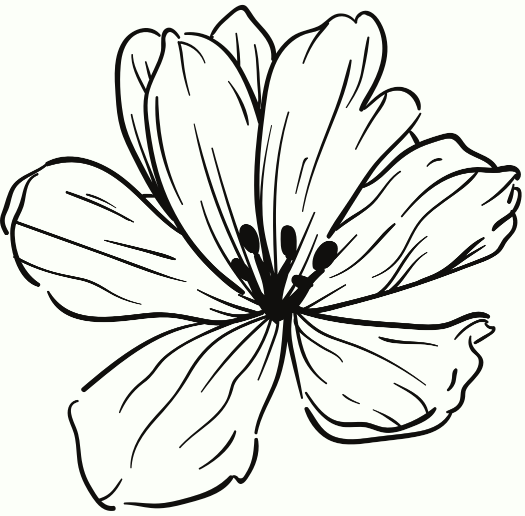
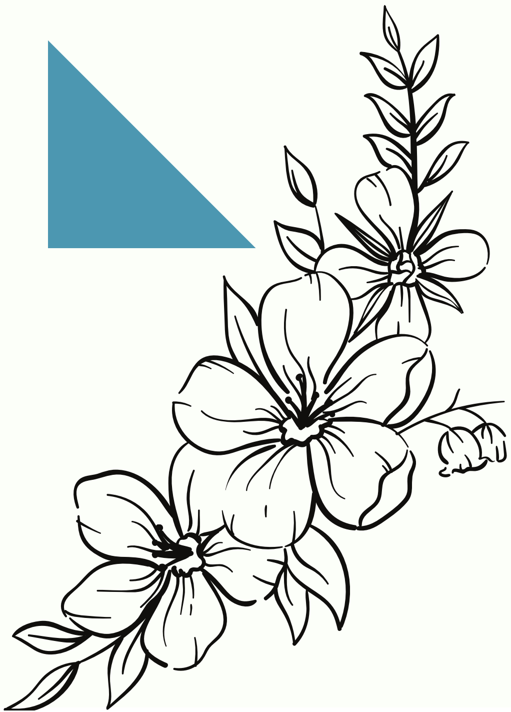


Flower Recognition Model Report

Ata Gürsel
Software Engineering
215060044



1. Introduction

This report thoroughly examines the Python code written by me for the design, training, evaluation, and testing of the flower recognition model. The code has been developed in Jupyter notebook. TensorFlow and Keras libraries were utilized for the creation of the model, focusing on a multi-class classification problem based on images during this process.

2. Data Preparation

2.1. Dataset

The dataset consists of images from the 'flower_photos' folder, encompassing a total of 17 different flower types. These flower types are selected to diversify the model's learning capabilities.

2.2. Data Dimensions

Images in the dataset are resized to 128x128 pixels for more effective learning.

2.3. Data Augmentation

Data augmentation is applied to the dataset using ImageDataGenerator. This involves introducing transformations such as random rotation, horizontal and vertical shifts to the images, aiming to enable the model to learn more general and varied features.

2.4. Data Splitting

The dataset is split into two subsets: training and validation. The training data is used during the model's learning process, while the validation data is reserved for assessing the model's performance.

3. Model Architecture

The model is a small convolutional neural network (CNN) with the following architecture:

- First convolutional layer: 32 filters, 3x3 kernel, ReLU activation
- First max-pooling layer
- Second convolutional layer: 64 filters, 3x3 kernel, ReLU activation
- Second max-pooling layer
- Flattening layer
- Two fully connected layers: 64 neurons each, ReLU activation
- Output layer: 17 neurons (corresponding to flower classes), softmax activation

The weights of the model are randomly initialized and optimized during the training process.

4. Model Training

The model is compiled using the 'adam' optimizer, categorical cross-entropy loss, and accuracy metric. It is trained for a total of 10 epochs, with the performance on both training and validation datasets monitored after each epoch.

4.1. Training Process Analysis

Detailed records of loss and accuracy metrics throughout the training process are kept. This data is utilized to assess the model's learning process and overall performance.

4.2. Model Performance

After training, accuracy and loss values on both training and validation datasets are visualized to assess the model's performance. This analysis is crucial for evaluating the occurrence of overfitting.

5. Model Saving

The trained model is saved with the name 'flower_recognition_model.keras', enabling it to be reused for future tasks.

6. Performance Evaluation

Post-training, the model's performance is assessed using various metrics. Accuracy and loss values during training and validation are graphically visualized after each epoch.

6.1. General Performance Analysis

Differences between training and validation accuracy values are analyzed to evaluate the presence of overfitting. Overfitting occurs when the model fits the training data well but struggles to generalize to new data.

6.2. Loss Analysis

Training and validation loss values are examined to determine the optimal number of epochs and assess the model's learning success.

7. Testing

The trained model is tested on flower images in the 'test' folder. Predictions made by the model, along with the correct labels, are visualized for each flower image. This step is essential for evaluating the model's performance on real-world data.

7.1. Test Results

Similarity between predictions and actual labels for each flower image is analyzed. Accurate prediction rates and detailed analysis of incorrect predictions are performed.

THANK YOU!