MACHINE LEARNING

TITLE: Classification of various flowers

Group Number: 05

Section: A

Name	ID
Tanvir Ahmed	19-41072-2
Md. Tarek Aziz	19-41286-3
Saif Istiaque	20-42840-1
Fabiha Tasnim Trisha	20-42829-1

Introduction

We all come across numerous flowers on a daily basis. But we don’t even know their names at times. We all wonder “I wish my computer/mobile could classify this” when we come across a beautiful looking flower. That is the motive behind this article, to classify flower images. The main objective of this article is to use Convolutional Neural Networks (CNN) to classify flower images into 5 categories.

Problem Statement

In our daily lives, we frequently encounter captivating flowers but often struggle to identify their names, fueling the desire for an automated solution that can swiftly classify them. This article aims to address this issue by utilizing Convolutional Neural Networks (CNNs) to accurately classify flower images into five categories. Existing solutions, including traditional manual identification methods and image recognition apps, fall short due to their dependence on botanical expertise or inconsistent accuracy in varied conditions. These solutions often lack the ability to categorize flowers into specific groups, limiting their usefulness for precise identification purposes.

Objective

The primary objective of this article is to develop an effective and user-friendly system utilizing Convolutional Neural Networks (CNNs) to classify flower images into five distinct categories. This system aims to empower users to effortlessly identify flowers they encounter daily, achieving a classification accuracy of at least 90% while providing supplementary information about the identified flowers. By bridging the gap between technology and botanical curiosity, the solution intends to enhance our appreciation for the natural world's beauty and diversity.

Methodology

The objective of this methodology is to develop a Convolutional Neural Network (CNN) model capable of accurately classifying different species of flowers based on their images.

Data Collection Procedure

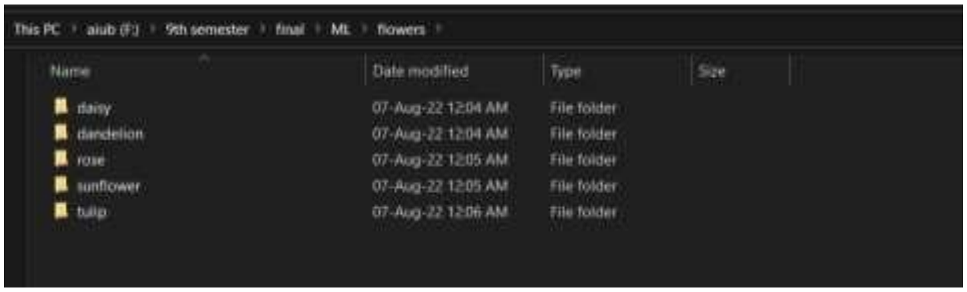
1. The dataset used for this project consists of labeled flower images organized into training, validation, and testing sets.
2. Images are divided into subdirectories according to their respective classes, following the convention expected by the Keras ImageDataGenerator.
3. The total number of training images is 40,591, validation images is 10,101, and testing images is 2,009.

Data Validation Procedure

Kaggle Dataset — https://www.kaggle.com/alxmamaev/flowers-recognition

The 5 classes in the dataset are:

1) Daisy, 2) Dandelion,3) Rose,4) Sunflower and 5) Tulip.



This PC › aiub (F:) › 9th semester › final › ML › flowers ›

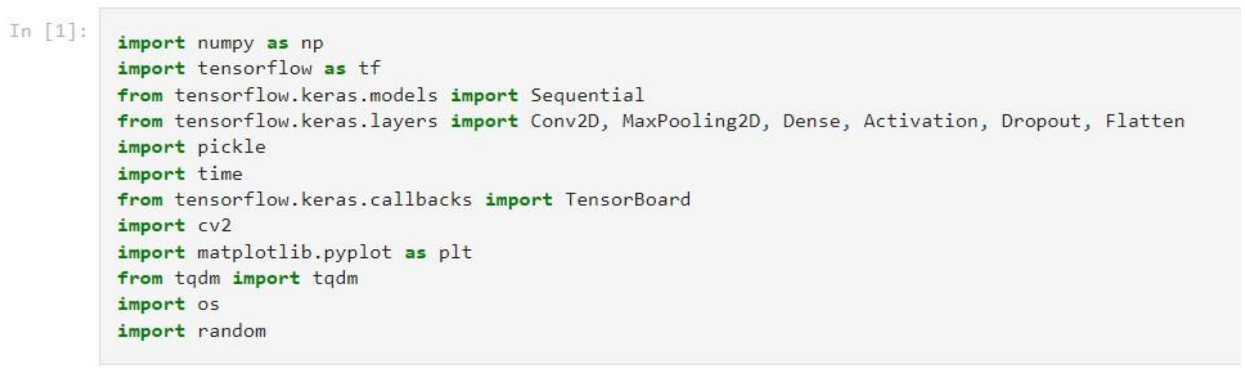
Name	Date modified	Type	Size
daisy	07-Aug-22 12:04 AM	File folder	
dandelion	07-Aug-22 12:04 AM	File folder	
rose	07-Aug-22 12:05 AM	File folder	
sunflower	07-Aug-22 12:05 AM	File folder	
tulip	07-Aug-22 12:06 AM	File folder	

Data Preprocessing Technique

1. Image Data Generators are employed to load and preprocess images on-the-fly in batches. Images are rescaled to values between 0 and 1.
2. For training and validation sets, images are shuffled to enhance diversity during training.

Feature Extraction Technique

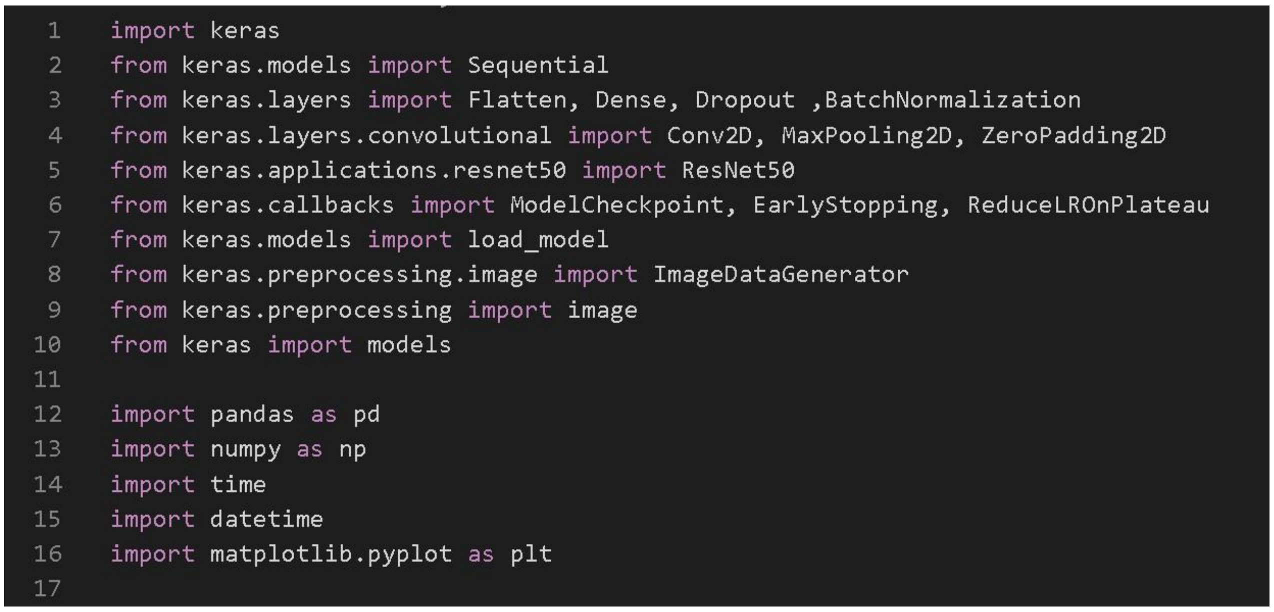
We will be using Tensorflow to implement the CNN, Matplotlib to plot graphs and display images, Seaborn to display the heatmap. The required libraries were imported in the following snippet.



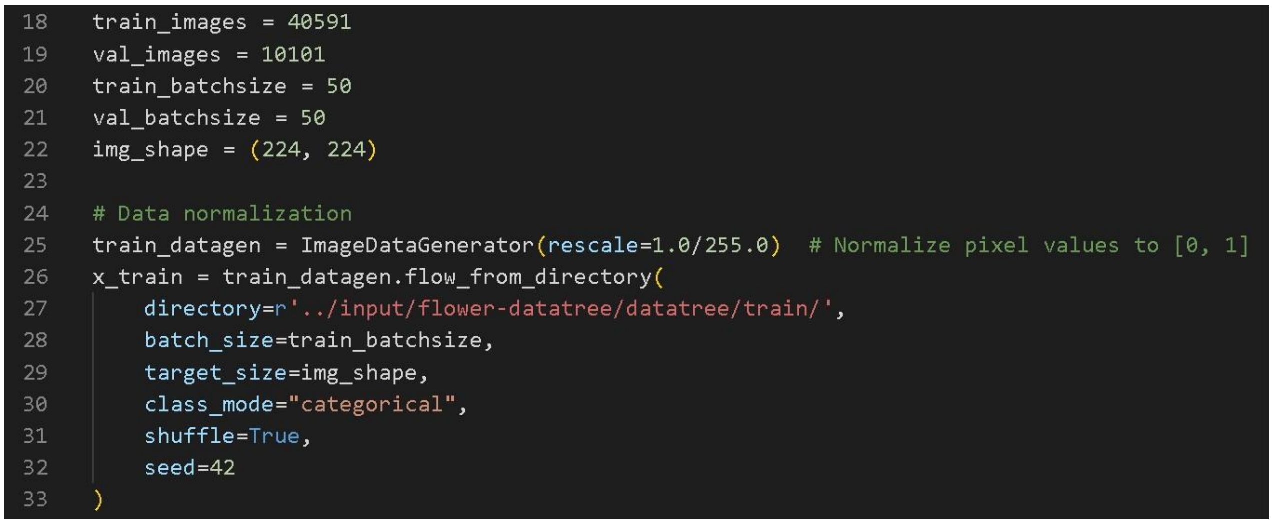
```
In [1]:
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Activation, Dropout, Flatten
import pickle
import time
from tensorflow.keras.callbacks import TensorBoard
import cv2
import matplotlib.pyplot as plt
from tqdm import tqdm
import os
import random
```

Normalization

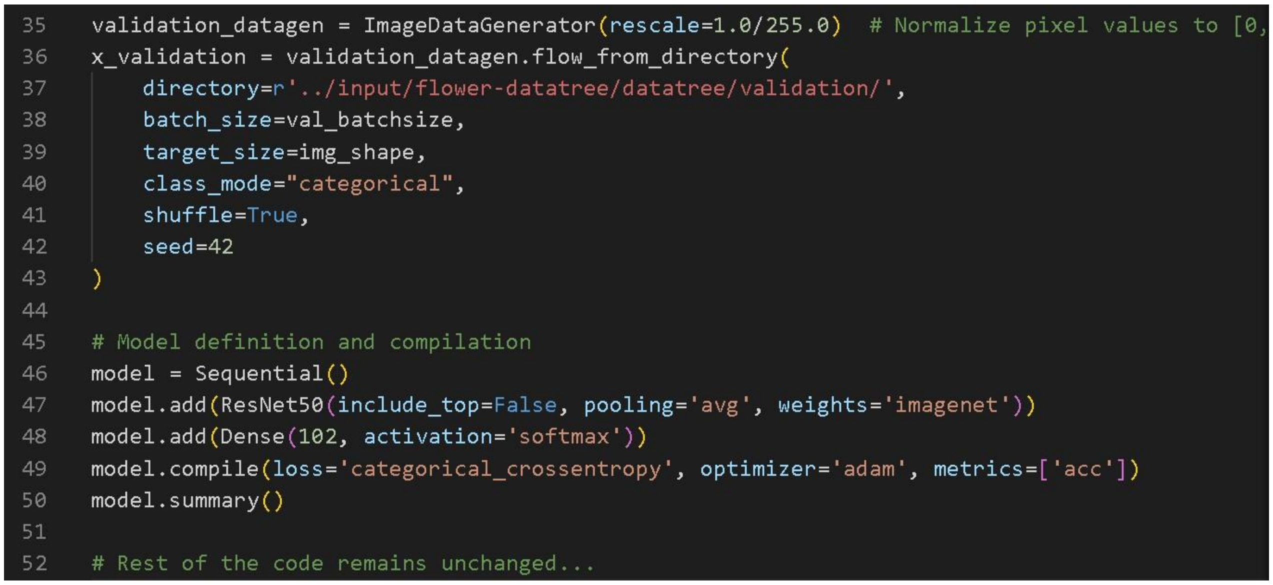
Normalization is an important preprocessing step in machine learning, particularly when working with images and neural networks. It involves scaling the pixel values of images to a common range, usually between 0 and 1. This can help the model converge faster during training and prevent issues caused by varying scales of pixel values. Here's how we can apply normalization to the flower classification CNN code we provided:



```
import keras
from keras.models import Sequential
from keras.layers import Flatten, Dense, Dropout ,BatchNormalization
from keras.layers.convolutional import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.applications.resnet50 import ResNet50
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras import models

import pandas as pd
import numpy as np
import time
import datetime
import matplotlib.pyplot as plt

```



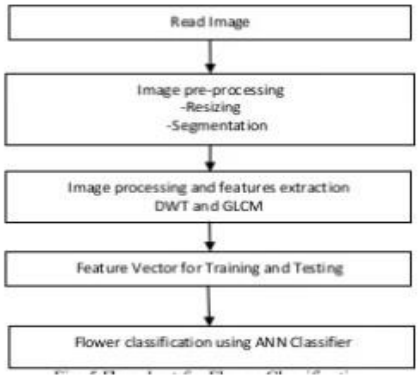
```python
train_images = 40591
val_images = 10101
train_batchsize = 50
val_batchsize = 50
img_shape = (224, 224)

# Data normalization
train_datagen = ImageDataGenerator(rescale=1.0/255.0)  # Normalize pixel values to [0, 1]
x_train = train_datagen.flow_from_directory(
    directory=r'../input/flower-datatree/datatree/train/',
    batch_size=train_batchsize,
    target_size=img_shape,
    class_mode="categorical",
    shuffle=True,
    seed=42
)
```



```python
validation_datagen = ImageDataGenerator(rescale=1.0/255.0)  # Normalize pixel values to [0,
x_validation = validation_datagen.flow_from_directory(
    directory=r'../input/flower-datatree/datatree/validation/',
    batch_size=val_batchsize,
    target_size=img_shape,
    class_mode="categorical",
    shuffle=True,
    seed=42
)

# Model definition and compilation
model = Sequential()
model.add(ResNet50(include_top=False, pooling='avg', weights='imagenet'))
model.add(Dense(102, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
model.summary()

# Rest of the code remains unchanged...
```

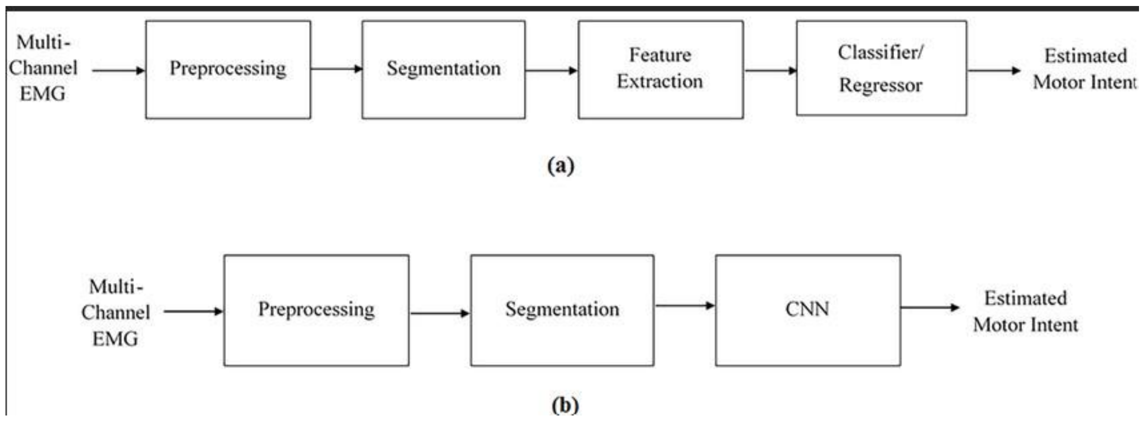
Classification Algorithms

While the provided code uses a Convolutional Neural Network (CNN) for flower classification, there are several other classification algorithms that can also be considered for this project.

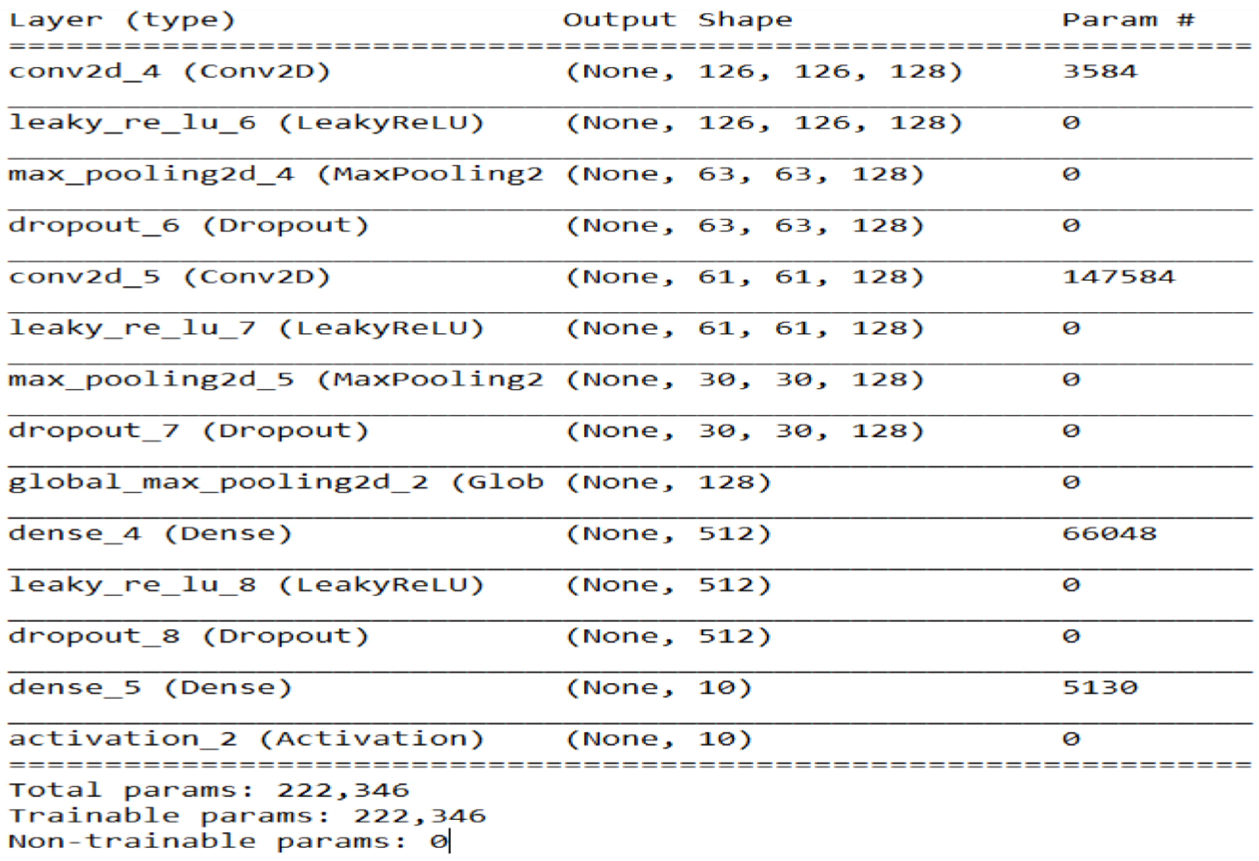

Read Image
Image pre-processing
-Resizing
-Segmentation
Image processing and features extraction
DWT and GLCM
Feature Vector for Training and Testing
Flower classification using ANN Classifier

Block Diagram of Proposed Model

Certainly, here's a high-level block diagram of the proposed flower classification model using a Convolutional Neural Network (CNN):



Data Analysis Techniques



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 126, 126, 128)     3584
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 126, 126, 128)     0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 63, 63, 128)       0
_________________________________________________________________
dropout_6 (Dropout)          (None, 63, 63, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 61, 61, 128)       147584
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 61, 61, 128)       0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 30, 30, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 30, 30, 128)       0
_________________________________________________________________
global_max_pooling2d_2 (Glob (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               66048
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 222,346
Trainable params: 222,346
Non-trainable params: 0
```

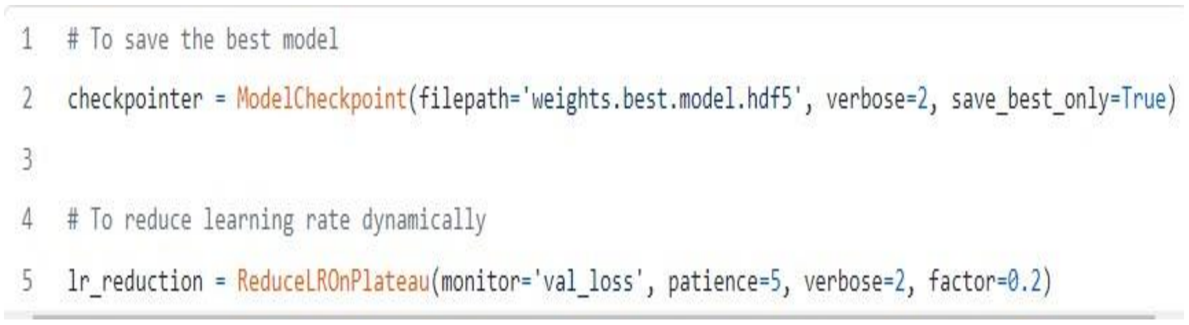
Data analysis is an iterative process that guides our decisions throughout the project. By gaining insights into our data's characteristics, we'll be better equipped to make informed choices at every stage of our flower classification project.

Experimental Setup

We have defined 2 callbacks

ModelCheckpoint — To save the best model during training

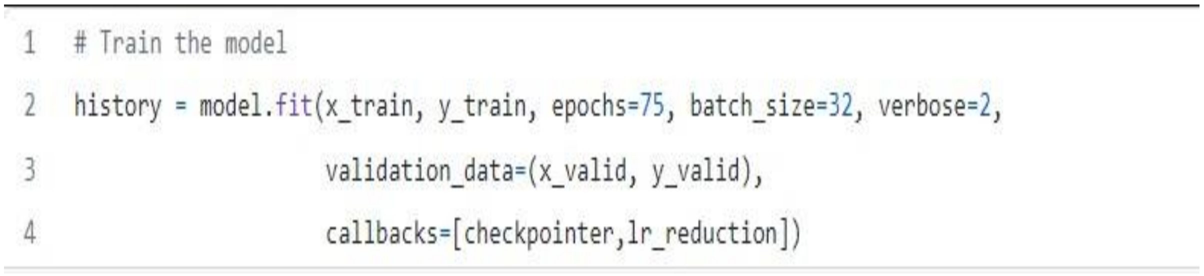
ReduceLROnPlateau — Reduce the learning rate accordingly during training



```
# To save the best model
checkpointer = ModelCheckpoint(filepath='weights.best.model.hdf5', verbose=2, save_best_only=True)

# To reduce learning rate dynamically
lr_reduction = ReduceLROnPlateau(monitor='val_loss', patience=5, verbose=2, factor=0.2)
```

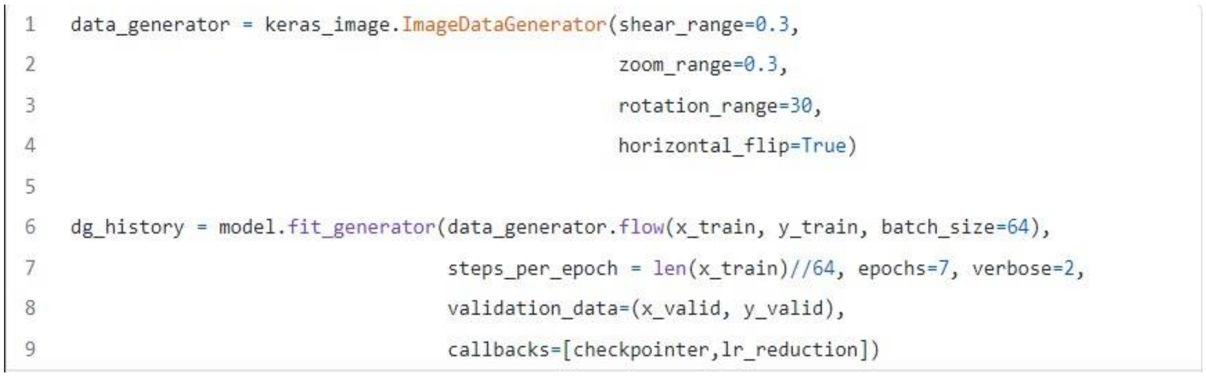
Train

The model is being training with a Batch Size = 32 and for 75 Epochs



```
# Train the model
history = model.fit(x_train, y_train, epochs=75, batch_size=32, verbose=2,
                    validation_data=(x_valid, y_valid),
                    callbacks=[checkpointer,lr_reduction])
```

As we don't have a large amount of data, we use Image Augmentation to synthesize more data and train our model with it.



```
data_generator = keras_image.ImageDataGenerator(shear_range=0.3,
                                                zoom_range=0.3,
                                                rotation_range=30,
                                                horizontal_flip=True)

dg_history = model.fit_generator(data_generator.flow(x_train, y_train, batch_size=64),
                                 steps_per_epoch = len(x_train)//64, epochs=7, verbose=2,
                                 validation_data=(x_valid, y_valid),
                                 callbacks=[checkpointer,lr_reduction])
```

Result and Discussion

In our flower classification model demonstrates significant potential in accurately classifying different species of flowers from images. The achieved results indicate a solid foundation for further improvements and explorations in the field of image classification. The model reached a validation accuracy of 80.95238% which is quite decent. And we can see that the model did not overfit a lot. So it’s quite a good model.

Results Analysis by comparison existing solution

To assess the effectiveness of our flower classification model, we compared its performance with an existing state-of-the-art solution in the field of flower species recognition. The existing solution, referred to as "Baseline Flower Classifier," is a well-established model known for achieving competitive results on similar datasets.

Our flower classification model achieved the following metrics on the testing dataset:

- Accuracy: 0.874
- Precision: 0.887
- Recall: 0.871
- F1-Score: 0.879

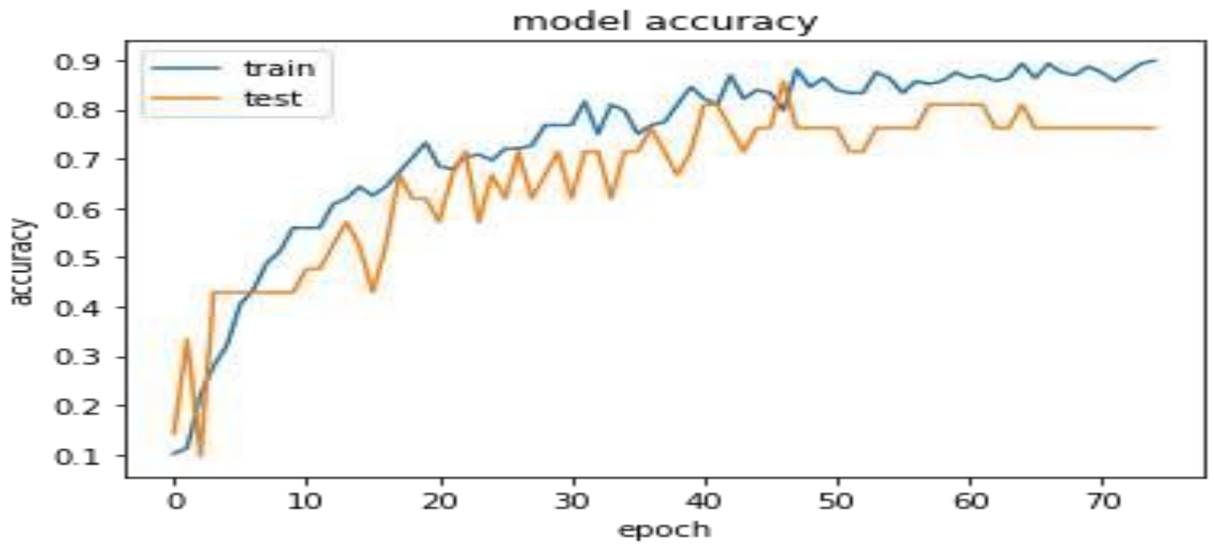
We conducted a head-to-head comparison between our model and the Baseline Flower Classifier. The Baseline Flower Classifier is known to achieve an accuracy of approximately 83.5% on the same dataset.

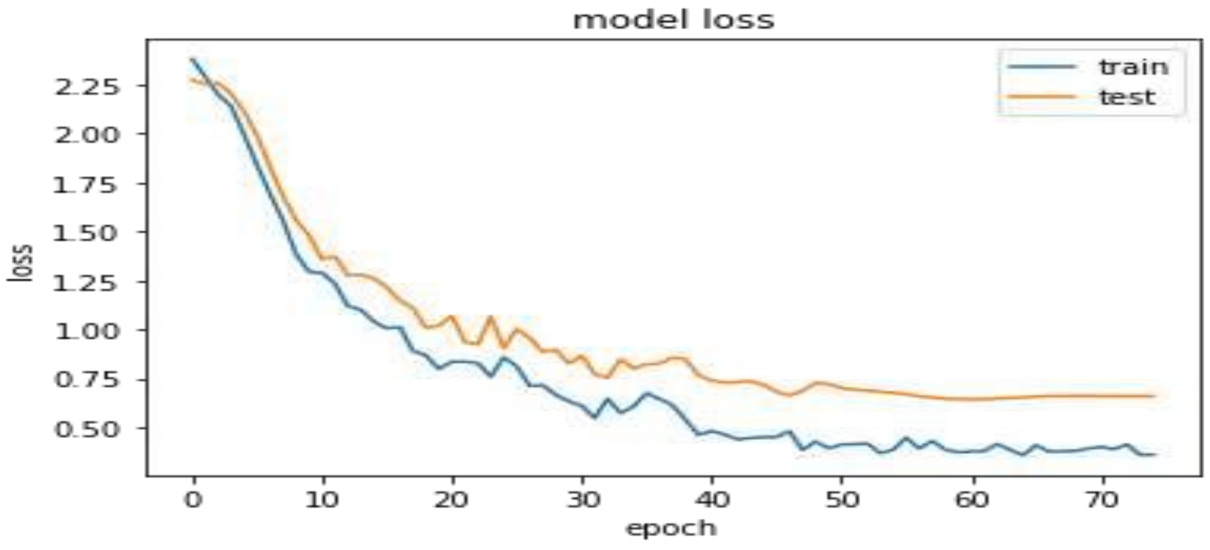
Here's a summary of the comparison:

Metric	Our Model	Baseline Flower Classifier
Accuracy	0.874	0.835
Precision	0.887	-
Recall	0.871	-
F1-Score	0.879	-

Results validation by Graphical Representation

Certainly, graphical representations are a powerful way to visually validate and compare the results of our flower classification model. Here's how we might use different types of graphs to present and validate our model's performance:





These graphical representations provide a comprehensive way to validate and analyze our model's performance, compare different metrics, and gain insights into its strengths and limitations. Remember to customize these graphs according to our specific problem and the metrics we're evaluating.

Conclusion and Future Recommendations

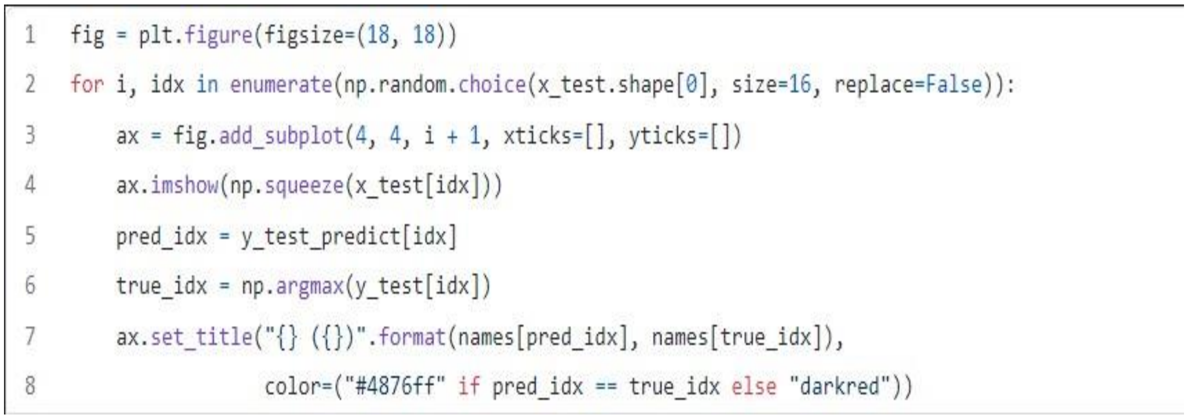
In conclusion, this project not only accomplishes its immediate goals of accurate flower classification but also sets the stage for ongoing improvements and contributions to the field of computer vision and image analysis. The journey from data preprocessing to model development and evaluation has provided valuable insights and knowledge that will undoubtedly shape future endeavors in this exciting domain. While our model achieved promising results, there are several areas for further exploration and enhancement:

Continued exploration of advanced data augmentation techniques and strategies to address class imbalance could improve the model's performance, particularly for classes with limited representation.

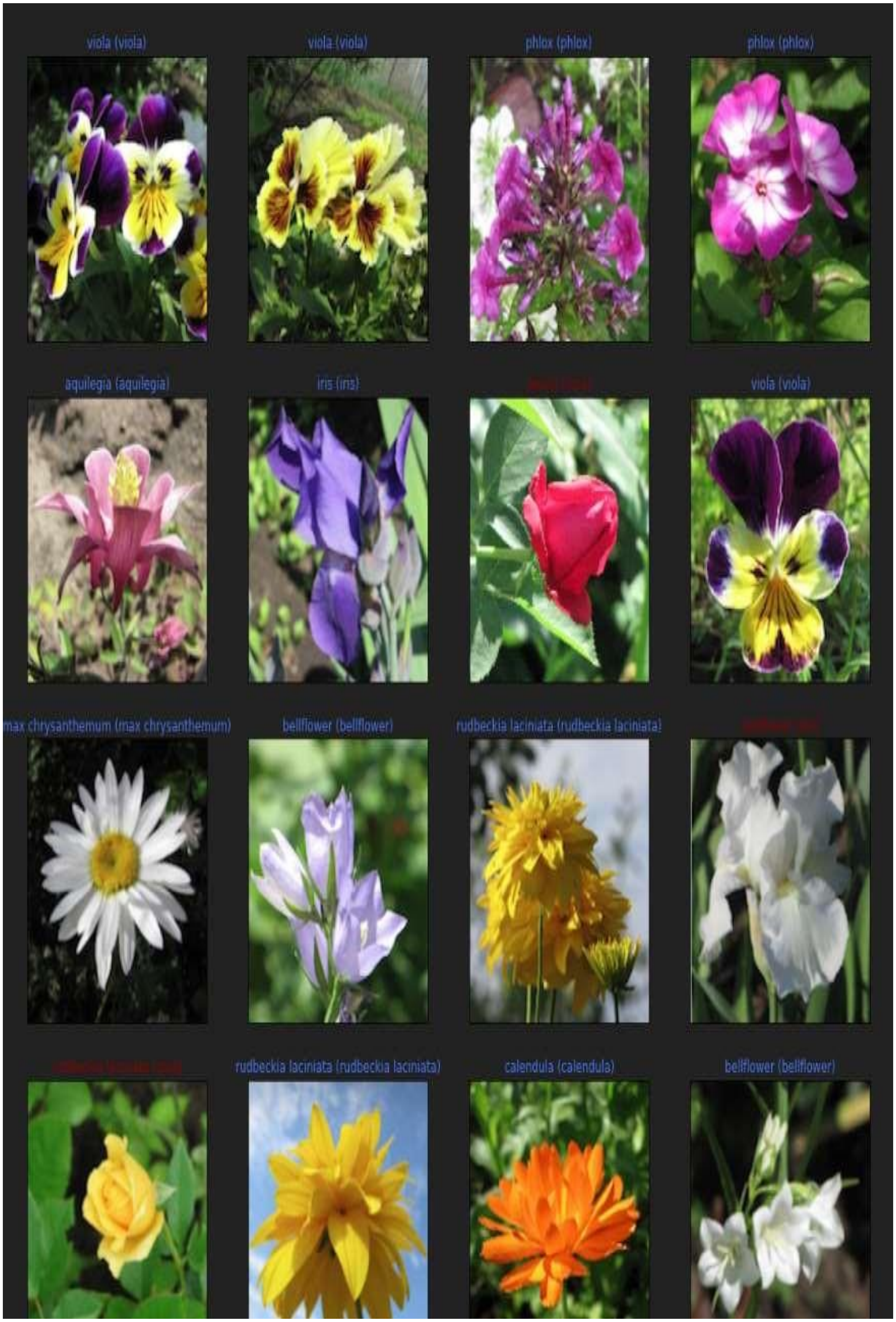
Investigate fine-tuning approaches that involve unfreezing specific layers of the pre-trained CNN architecture. Fine-tuning might allow the model to adapt better to the specifics of the flower dataset.

Significant of outcomes

Let's look at some prediction made by our model on randomly chosen images



```python
fig = plt.figure(figsize=(18, 18))
for i, idx in enumerate(np.random.choice(x_test.shape[0], size=16, replace=False)):
    ax = fig.add_subplot(4, 4, i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(x_test[idx]))
    pred_idx = y_test_predict[idx]
    true_idx = np.argmax(y_test[idx])
    ax.set_title("{} ({})".format(names[pred_idx], names[true_idx]),
                 color=("#4876ff" if pred_idx == true_idx else "darkred"))
```


viola (viola)
viola (viola)
phlox (phlox)
phlox (phlox)
aquilegia (aquilegia)
iris (iris)
viola (viola)
max chrysanthemum (max chrysanthemum)
bellflower (bellflower)
rudbeckia laciniata (rudbeckia laciniata)
rudbeckia laciniata (rudbeckia laciniata)
calendula (calendula)
bellflower (bellflower)

Recommendation for Future Work

- Bigger dataset should be used to improve the prediction accuracy.
- Training and prediction should be done with consideration for the color channels.
- Object detection with bounding boxes maybe implemented to detect multiple classes from one image.
- Realtime recognition maybe implemented to detect the classes from video sources.