National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Pratik Anil Dhaygude
Student ID: x22108670

School of Computing
National College of Ireland

Supervisor: Barry Haycock

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Pratik Anil Dhaygude
Student ID:	x22108670
Programme:	MSc Data Analytics **Year:** 2024
Module:	MSc Research Project
Supervisor:	Barry Haycock
Submission Due Date:	12th August 2024
Project Title:	Configuration Manual
Word Count:	..... **Page Count** .....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: *Pratik dhaygude*

Date: 12/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Introduction

This configuration manual helps to run the code developed for this study. Step-by-step implementation of these instructions will ensure smooth code execution without errors. The guide also covers the hardware requirements for code execution, including the recommended minimum specs. Adhering to these guidelines will help for the complete replication of project results, allowing for further analysis and making it easier to continue the research.

System Specification

Hardware Configuration

Below are the required system specifications to execute the code:

- **Processor:** Intel Core i5
- **System Memory:** 1TB SSD Hard Disk
- **RAM:** 8GB

Software Configuration

The software requirements are discussed below:

- **Windows Edition:** Windows 11, Mac
- **Integrated Development Environment:** Jupyter Notebook
- **Scripting Language:** Python 3 +
- **Storage:** Local System Storage

Python Libraries

To perform exploratory data analysis and machine learning model implementation some Python libraries are required to be installed which can done by using pip command. In Jupyter Notebook, the libraries can be installed by using the “pip install <library name>” command.

Following are the libraries needed to be installed in Jupyter Notebook:

- **requests:** The requests library is an easy interface to send HTTP requests and interact with web resources. It makes it easy to fetch data from APIs and handle responses smoothly.
- **pandas:** pandas provide data analysis tools to work with structured data. It’s used for analyzing and manipulating tabular data.
- **Numpy:** The numpy is an essential tool for numeric calculations in Python. It processes large multi-dimensional arrays and provides mathematical functions for calculations
- **os:** Os library provides functions for file operations, managing directories, and handling environment variables.

- **seaborn:** seaborn is used to visualize beautiful and informative statistical graphics.
- matplotlib: It's a comprehensive library for creating all sorts of visualizations and animations.
- **Sklearn:** The scikit-learn provides a toolkit to implement machine learning algorithms like classification, regression, clustering, and model evaluation
- **Tensorflow:** Tensorflow is a robust framework for developing machine learning and deep learning models
- **Tkinter:** It provides a way to create windows, dialogs, buttons, labels, and other UI elements in a Python application.

```
[12]: import numpy as np
      import pandas as pd
      import matplotlib.pyplot as plt
      import seaborn as sns
      import os
      import tensorflow as tf
      from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
      from tensorflow.keras.models import Model
      from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
      from tensorflow.keras.applications import ResNet50, VGG16, InceptionV3
      from tensorflow.keras.optimizers import Adam
      from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
      from sklearn.model_selection import train_test_split
      from sklearn.ensemble import StackingClassifier, BaggingClassifier, AdaBoostClassifier
      from sklearn.linear_model import LogisticRegression
      import tkinter as tk
      from tkinter import filedialog
```

Project Development

After installing the required Python libraries, the code is ready for execution.

```
[7]: # Data Augmentation and Preparation
     datagen = ImageDataGenerator(
         rescale=1./255,
         validation_split=0.2,
         rotation_range=20,
         width_shift_range=0.2,
         height_shift_range=0.2,
         horizontal_flip=True,
         vertical_flip=True
     )

     train_generator = datagen.flow_from_directory(
         data_dir,
         target_size=(224, 224),
         batch_size=128,
         class_mode='categorical',
         subset='training'
     )

     validation_generator = datagen.flow_from_directory(
         data_dir,
         target_size=(224, 224),
         batch_size=128,
         class_mode='categorical',
         subset='validation'
     )

     Found 3457 images belonging to 5 classes.
     Found 860 images belonging to 5 classes.
```

Model: "sequential"

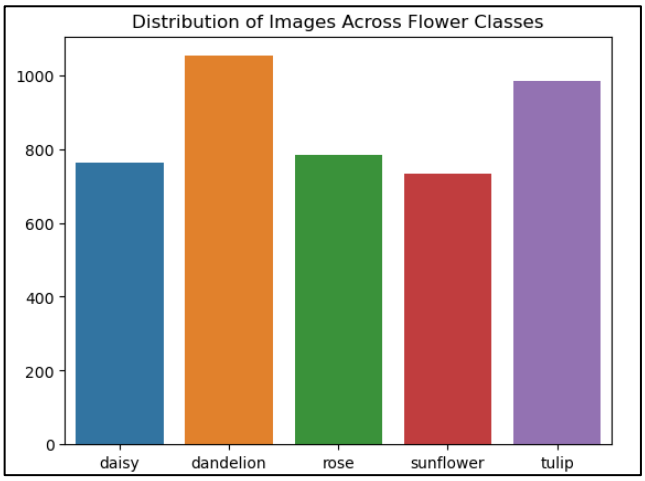
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 512)	12,845,568
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 5)	2,565

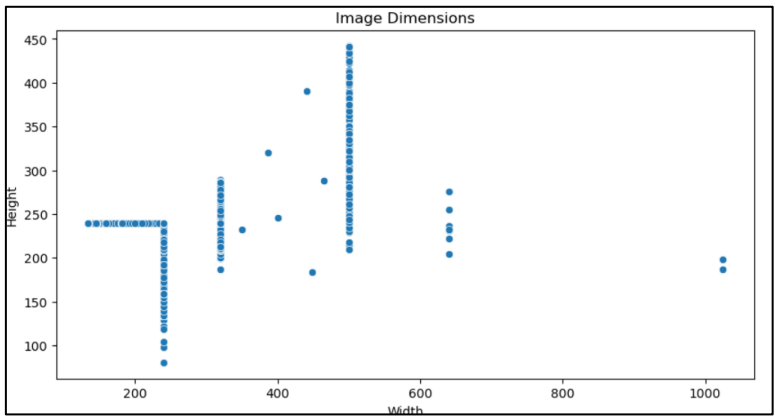
Total params: 12,941,381 (49.37 MB)
Trainable params: 12,941,381 (49.37 MB)
Non-trainable params: 0 (0.00 B)

```
[9]: # Model Architecture
def build_model(base_model):
    model = base_model(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    x = GlobalAveragePooling2D()(model.output)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(len(categories), activation='softmax')(x)
    return Model(inputs=model.input, outputs=predictions)

resnet_model = build_model(ResNet50)
vgg_model = build_model(VGG16)
inception_model = build_model(InceptionV3)

models = [resnet_model, vgg_model, inception_model]

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 8s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 ━━━━━━━━━━━━━━━━━━━━ 7s 0us/step
```

Data Configuration:

- **Image Size**: Images are resized to 224x224 for exploratory data analysis and 128x128 for model training.
- **Categories**: Ensure that the `categories` list matches the subdirectories in the `data_dir`. If you add or remove categories, update the `categories` list accordingly.





DataFrame head is shown below:

```
print(classification_report(y_test,np.argmax(predict,axis=1)))
```

	precision	recall	f1-score	support
0	0.73	0.60	0.66	197
1	0.65	0.83	0.73	253
2	0.64	0.50	0.56	197
3	0.77	0.74	0.76	200
4	0.62	0.67	0.64	233
accuracy			0.68	1080
macro avg	0.68	0.67	0.67	1080
weighted avg	0.68	0.68	0.67	1080

Classification Report

```
[15]: # Training the Base Models
def train_model(model, train_generator, validation_generator):
    model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
    history = model.fit(train_generator, validation_data=validation_generator, epochs=2)
    return history

histories = []
for model in models:
    histories.append(train_model(model, train_generator, validation_generator))
```

```
Epoch 1/2
28/28 ━━━━━━━━━━━━━━━━━━━━ 1390s 44s/step - accuracy: 0.8799 - loss: 0.3400 - val_accuracy: 0.2442 - val_loss: 2.2348
Epoch 2/2
28/28 ━━━━━━━━━━━━━━━━━━━━ 1042s 36s/step - accuracy: 0.9404 - loss: 0.1796 - val_accuracy: 0.2442 - val_loss: 2.6061
Epoch 1/2
28/28 ━━━━━━━━━━━━━━━━━━━━ 2295s 81s/step - accuracy: 0.4593 - loss: 1.3118 - val_accuracy: 0.7070 - val_loss: 0.7392
Epoch 2/2
28/28 ━━━━━━━━━━━━━━━━━━━━ 2121s 75s/step - accuracy: 0.7449 - loss: 0.6568 - val_accuracy: 0.7977 - val_loss: 0.5849
Epoch 1/2
28/28 ━━━━━━━━━━━━━━━━━━━━ 590s 18s/step - accuracy: 0.5862 - loss: 1.0630 - val_accuracy: 0.7535 - val_loss: 1.1857
Epoch 2/2
28/28 ━━━━━━━━━━━━━━━━━━━━ 798s 29s/step - accuracy: 0.8662 - loss: 0.3602 - val_accuracy: 0.8279 - val_loss: 0.7321
```

Model Configuration

- **CNN Model**: The script uses a Convolutional Neural Network (CNN) model with three convolutional layers, followed by max-pooling layers, a dense layer, and a dropout layer. The model is compiled with the Adam optimizer and categorical cross-entropy loss.
- **Input Shape**: The input shape for the model is `(128, 128, 3)`, corresponding to images of size 128x128 with 3 color channels (RGB).

Execution

- **Exploratory Data Analysis (EDA)**: The code includes EDA steps to visualize the number of images per category, sample images, and image dimensions. These plots will be displayed using `matplotlib` and `seaborn`.
- **Model Summary**: The script will print a summary of the CNN model architecture using `cnn_model.summary()`.

Hardware Requirements

- **GPU (Optional)**: If training the model on a large dataset, it is recommended to use a GPU to speed up the training process. TensorFlow will automatically utilize the GPU if available.
- **Memory**: Ensure sufficient memory (RAM) is available, especially if working with a large number of images or large image sizes.

. Customization

- **Model Layers**: You can customize the number of layers, filter sizes, and units in the dense layer by modifying the `create_cnn_model()` function.
- **Image Augmentation**: Consider using `ImageDataGenerator` for data augmentation to improve model generalization. This is not included in the current script but can be easily added.