School of Computing

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

Fruit Classification Report

<Full Name of Author>

Submitted in accordance with the requirements for the degree of
<Name of Degree> (*e.g.* BSc Computer Science)

<Session> (*e.g.* 2021/22)

<Module code and name> (*e.g. COMP3931 Individual Project*)

Table of Contents

Chapter 1: Introduction and Background Research

1.1 Introduction

In term of fruit classification Artificial Intelligence (AI) plays the significant role, specially have a huge impact on the agricultural industry. The project is used to develop an AI-based solution for the fruit sorting which will replace the current manual methods which are commonly used in the industry.

1.2 The Necessity for Innovation

1.2.1 Current Limitations in Fruit Sorting

Manual methods for fruit sorting which are using now a days having the certain drawbacks,

- Labor-intensive
- Time-taking
- Likely have error
- Impacting productivity and profitability.

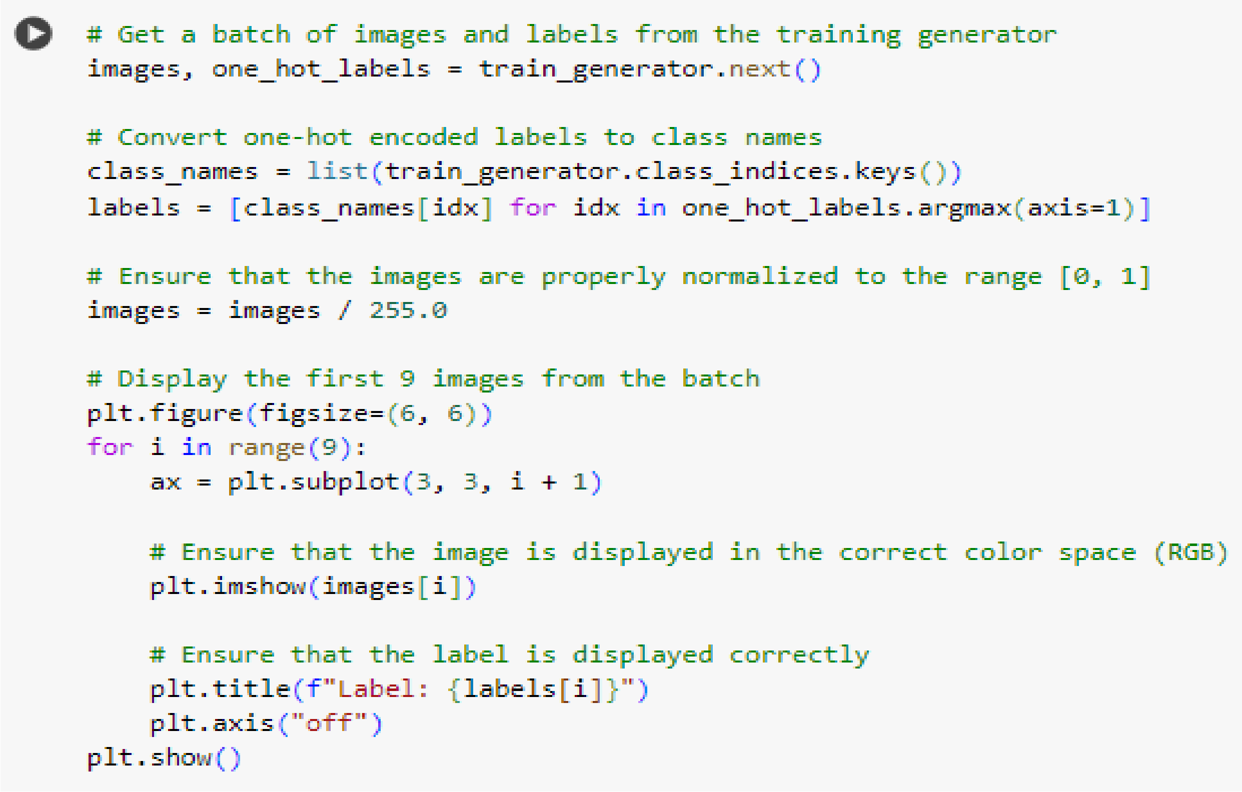
These above discussed limitations causes efficiency and accuracy in term of sorting process, which prominence the need of the alternative modern solution of this problem.

1.2.2 The Potential of AI

The use of AI techniques helps in solving the limitations of the manual methods in the following ways,

- Provide automation in sorting process
- Augmenting the efficiency, accuracy, and scalability of the problem

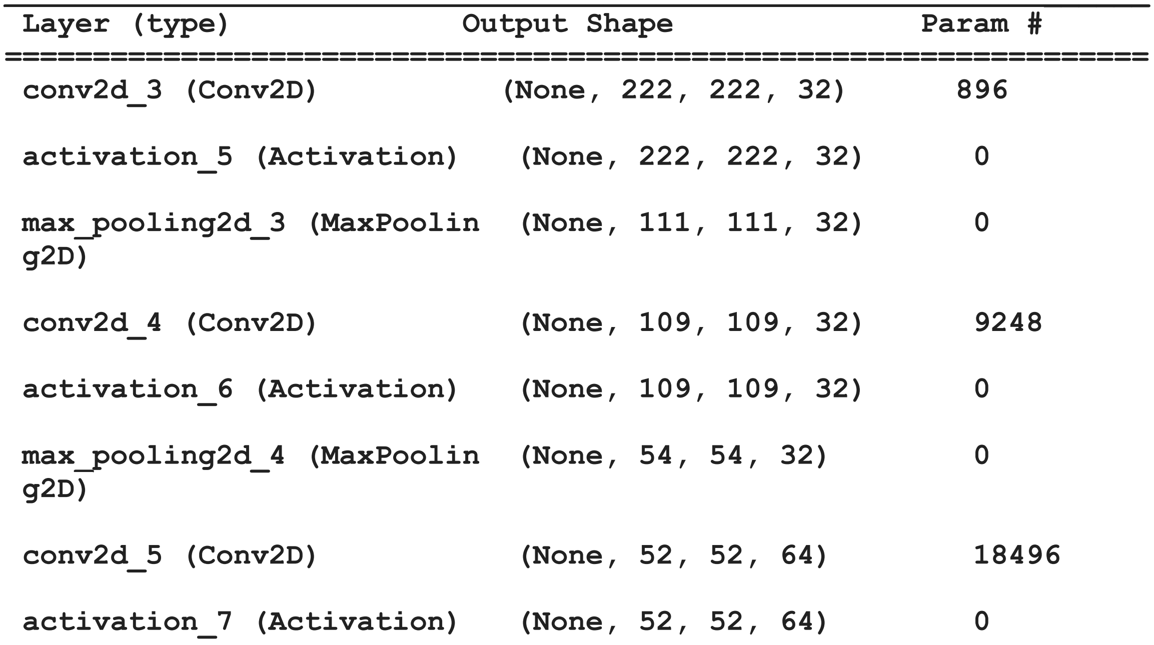
By using the above mentioned progress which provide transformative advancement in fruit classification in agricultural industry, which will lead the improved efficiency, scalability and the economics benefits.



```
# Get a batch of images and labels from the training generator
images, one_hot_labels = train_generator.next()

# Convert one-hot encoded labels to class names
class_names = list(train_generator.class_indices.keys())
labels = [class_names[idx] for idx in one_hot_labels.argmax(axis=1)]

# Ensure that the images are properly normalized to the range [0, 1]
images = images / 255.0

# Display the first 9 images from the batch
plt.figure(figsize=(6, 6))
for i in range(9):
    ax = plt.subplot(3, 3, i + 1)

    # Ensure that the image is displayed in the correct color space (RGB)
    plt.imshow(images[i])

    # Ensure that the label is displayed correctly
    plt.title(f"Label: {labels[i]}")
    plt.axis("off")
plt.show()
```

1.2.3 Advancements in AI Techniques

CNNs (Convolutional Neural Networks) a state of the art Ai technique has shown the outstanding performance in the task of the image recognition tasks, fruit classification having the direct application on the project's objective.



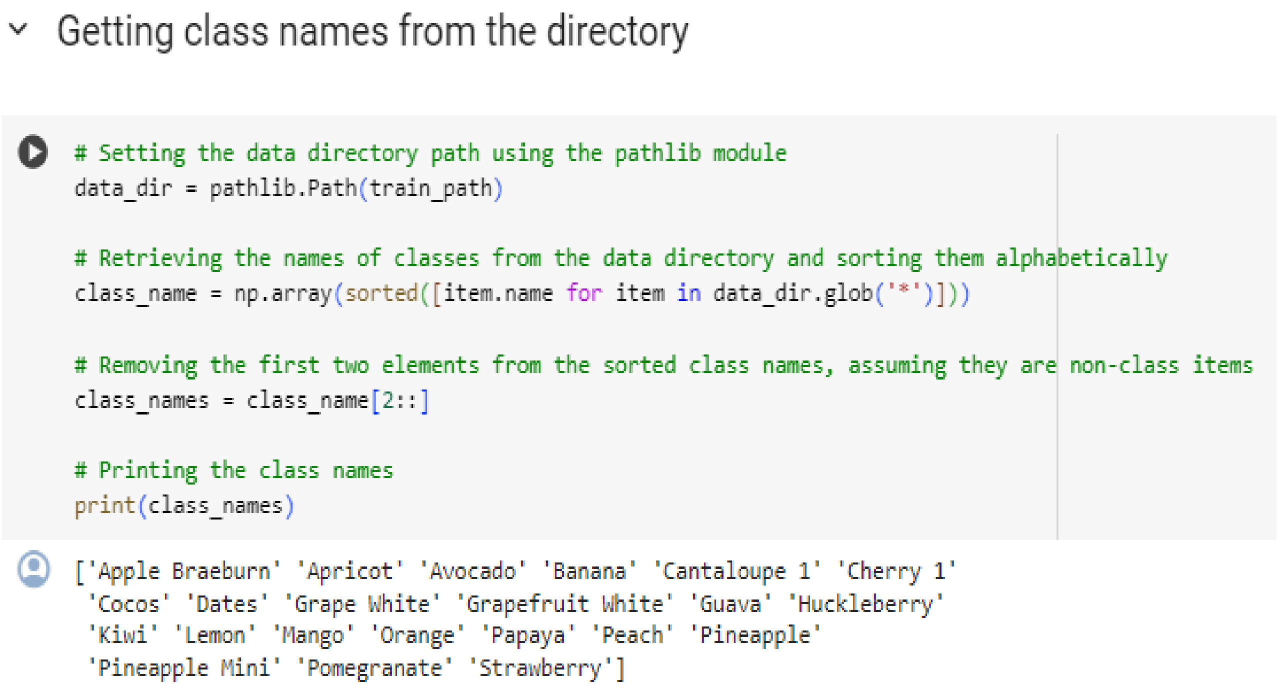
```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)          (None, 222, 222, 32)      896

 activation_5 (Activation)   (None, 222, 222, 32)      0

 max_pooling2d_3 (MaxPoolin  (None, 111, 111, 32)      0
 g2D)

 conv2d_4 (Conv2D)           (None, 109, 109, 32)      9248

 activation_6 (Activation)   (None, 109, 109, 32)      0

 max_pooling2d_4 (MaxPoolin  (None, 54, 54, 32)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 52, 52, 64)        18496

 activation_7 (Activation)   (None, 52, 52, 64)        0
 max_pooling2d_5 (MaxPoolin  (None, 26, 26, 64)        0
 g2D)

 flatten_1 (Flatten)         (None, 43264)             0

 dense_2 (Dense)             (None, 1024)              44303360

 activation_8 (Activation)   (None, 1024)              0

 dropout_1 (Dropout)         (None, 1024)              0

 dense_3 (Dense)             (None, 23)                23575

 activation_9 (Activation)   (None, 23)                0

=================================================================
Total params: 44355575 (169.20 MB)
Trainable params: 44355575 (169.20 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

1.2.4 Integration with IoT for Real-Time Monitoring

Joining AI with IoT (Internet of Things) helps in improving the technology and allow for real-time monitoring of the fruit;

- Quality
- Ripeness
- Inventory Management

This real-time monitoring helps in process to gain valuable insights, quality control and enabling proactive decision-making, so there should must require getting the class name from the directory, here below is the code for performing that particular operations;



Getting class names from the directory

```
# Setting the data directory path using the pathlib module
data_dir = pathlib.Path(train_path)

# Retrieving the names of classes from the data directory and sorting them alphabetically
class_name = np.array(sorted([item.name for item in data_dir.glob('*')]))

# Removing the first two elements from the sorted class names, assuming they are non-class items
class_names = class_name[2::]

# Printing the class names
print(class_names)
```

```
['Apple Braeburn' 'Apricot' 'Avocado' 'Banana' 'Cantaloupe 1' 'Cherry 1'
 'Cocos' 'Dates' 'Grape White' 'Grapefruit White' 'Guava' 'Huckleberry'
 'Kiwi' 'Lemon' 'Mango' 'Orange' 'Papaya' 'Peach' 'Pineapple'
 'Pineapple Mini' 'Pomegranate' 'Strawberry']
```

1.3 Evolution of Classification Methods

1.3.1 From Manual to Automated

As the many of the technologies evolving, like manual entering the data is now shifted towards the automated version as in same case with the fruit classification technique, manual sorting methods to automated techniques driven in advancement in technology, this increased the efficiency in handling growing volume produce.

1.3.2 Breakthroughs in Machine Learning

The introduce of the CNNs in the ML (Machine Learning), by the use of CNNs surpassed the old methods by effectively taking patterns and feature in the images, which helps in the fruit classification task.

1.4 Advancements in AI and Their Application

1.4.1 AI's Role in Agriculture Today

These are many of the AI solutions are practically applied in various of the agriculture field, such as the;

- Monitoring of crop
- Prediction of yield
- Detection of diseases

These sort of the things are used in the fruit classifications for similar advancement.

1.4.2 The Advent and Impact of Machine Learning

A subset of the ML (Machine Learning), DL (Deep learning) has played the vital role in contribution of agricultural challenges, by powerful computational resources and investigating large datasets, which making the accurate predictions.

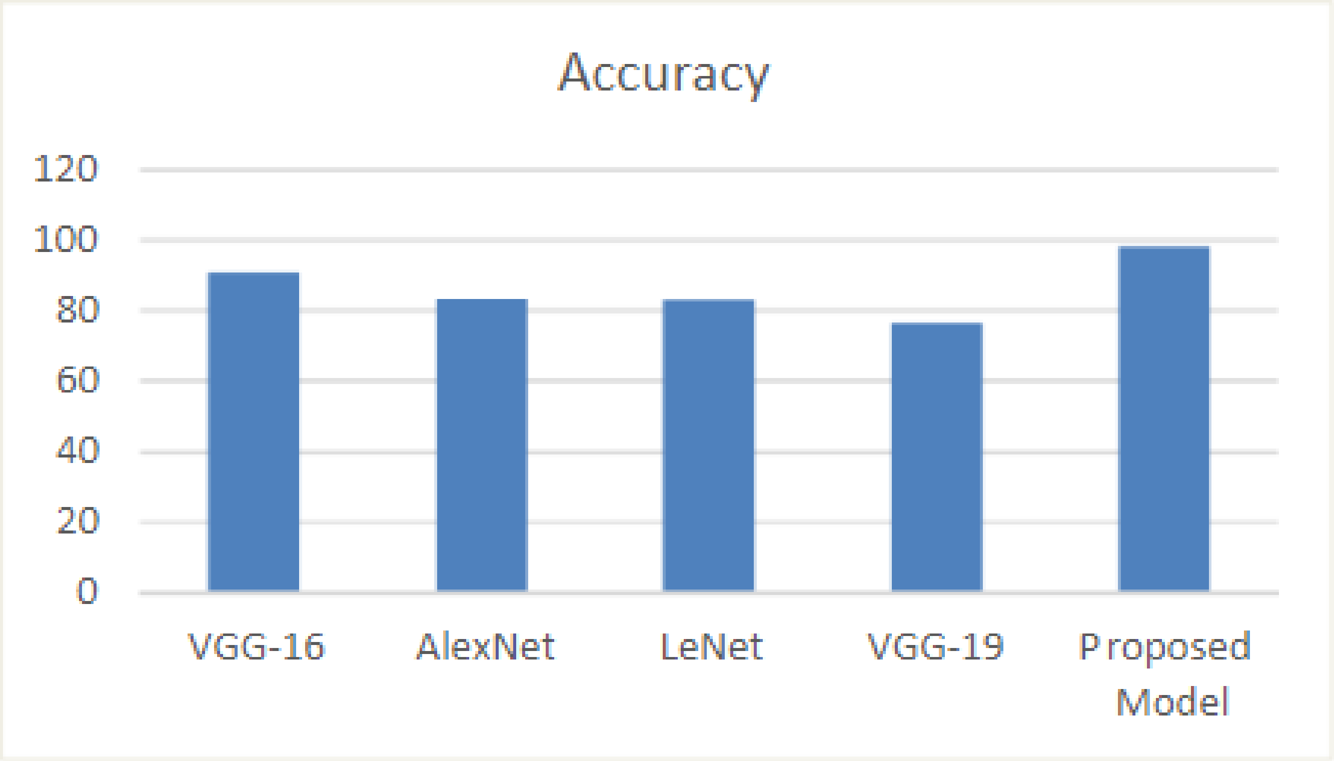
1.4.3 Current AI Advancements and Their Implications

Improved model architecture, algorithm's optimization, and techniques used in the transferred learning as the advancement of the in AI (Artificial Intelligence), these increased the accuracy of the several pre-trained models, also helpful in the fruit classification model.

1.5 Review of Existing AI Solutions

1.5.1 Comparison of Different AI Models

There would be analysis of the various AI models, having SVMs (Support Vector Machines), Random Forests, aware of the strengths and weakness of each of the models.



Here is the comparison of Proposed model with pre-trained models.

1.5.2 Critical Evaluation of Models

The evaluation of each of the AI models will be based upon the following features;

- Implemented Methodologies
- Computational complexity

- Strength in real-world environments.

This above evaluation will identify gaps and areas for improvement, which will be addressed in the proposed solution.

1.5.3 Identifying the Gaps

Identification of the gaps in any of the existing AI system are given below;

- Scalability issues.
- Misbalancing in tha data.
- Extraction of the domain specific issues.

These all of the gaps will serve as the principle guiding for developments of the advanced classification problems.

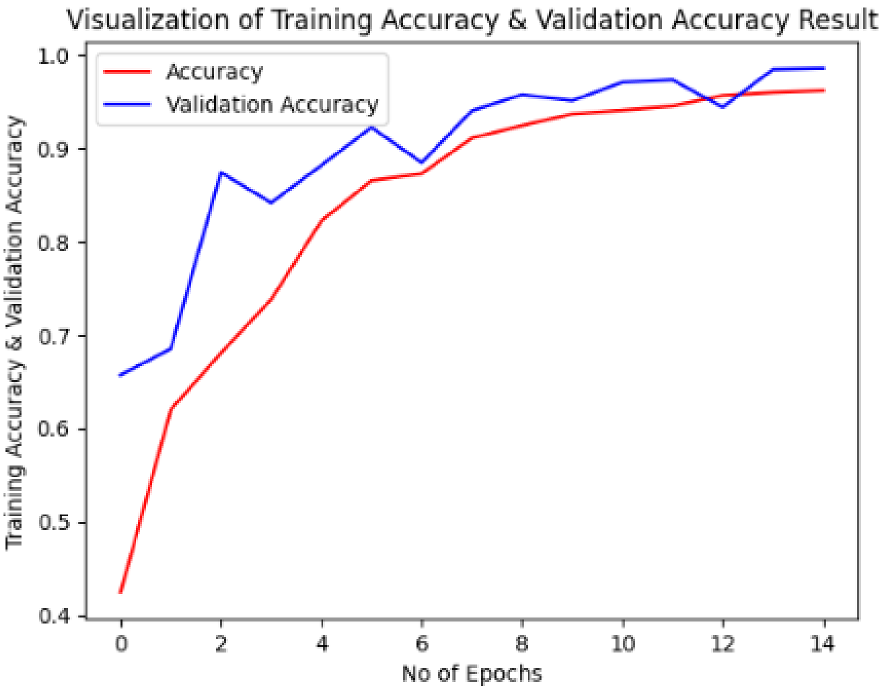
1.6 Proposed Solution and Justification

1.6.1 Justification for Using CNNs

As discussed earlier identifying the gaps form the previous architecture make the guideline for the other modern architectures, that's why use of the CNNs is the best choice from the old models, this having the ability of getting the complex image features and pattern, make suitable for use in classification purposes.

1.6.2 Expected Outcomes and Benefits

Improvements in the accuracy, scalability, and efficiency are the perks of the using CNN's technique, which helps a lot in the classification methods, the improvement helps in the sorting operations, waste reduction which increased in the productivity of agricultural industry.



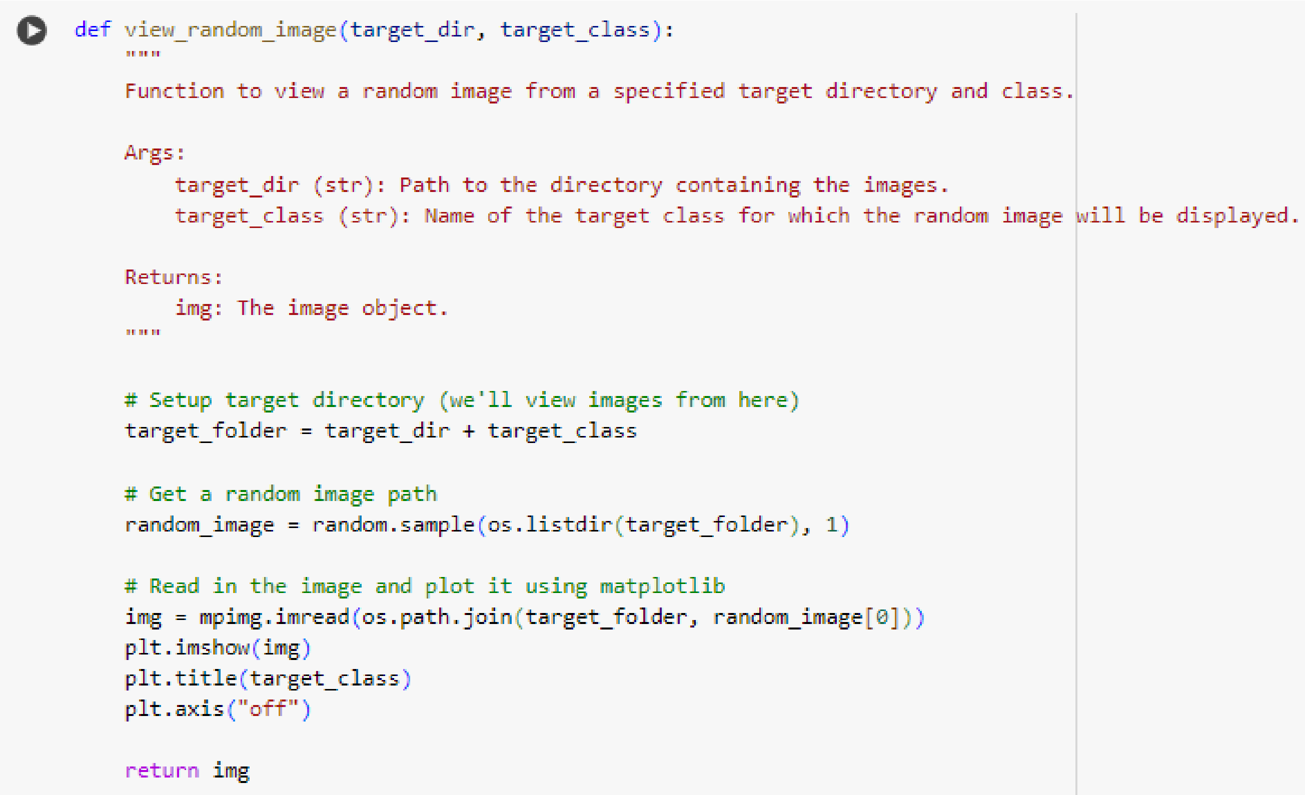
1.7 Overview of CNN Architecture

1.7.1 CNN Basics

By fetching the features from the input image, CNNs is particularly used, this consists of convolutional, pooling, and most important fully connected layer which flatten the image and also get spatial hierarchies from input image.

1.7.2 Application of CNNs to Fruit Classification

The CNNs is applied on the training dataset having labeled fruit images, based on the visual features learning is happened and distinguished between the different fruit types, by using this overall goal is to improving accuracy of the system, which is also be need for the fruit classification task.



```
def view_random_image(target_dir, target_class):
    """
    Function to view a random image from a specified target directory and class.

    Args:
        target_dir (str): Path to the directory containing the images.
        target_class (str): Name of the target class for which the random image will be displayed.

    Returns:
        img: The image object.
    """

    # Setup target directory (we'll view images from here)
    target_folder = target_dir + target_class

    # Get a random image path
    random_image = random.sample(os.listdir(target_folder), 1)

    # Read in the image and plot it using matplotlib
    img = mpimg.imread(os.path.join(target_folder, random_image[0]))
    plt.imshow(img)
    plt.title(target_class)
    plt.axis("off")

    return img
```

Chapter 2: Methods

2.1 Introduction to Methodology

This chapter is used to show the methods which are utilize in the fruit classification task and majorly highlights the importance of the research, the chosen of this method is used for considering the project scope and them researches, this chapter is also used for an overview of the organized stricture of the methods that we are using now a day in the improvement purposes.

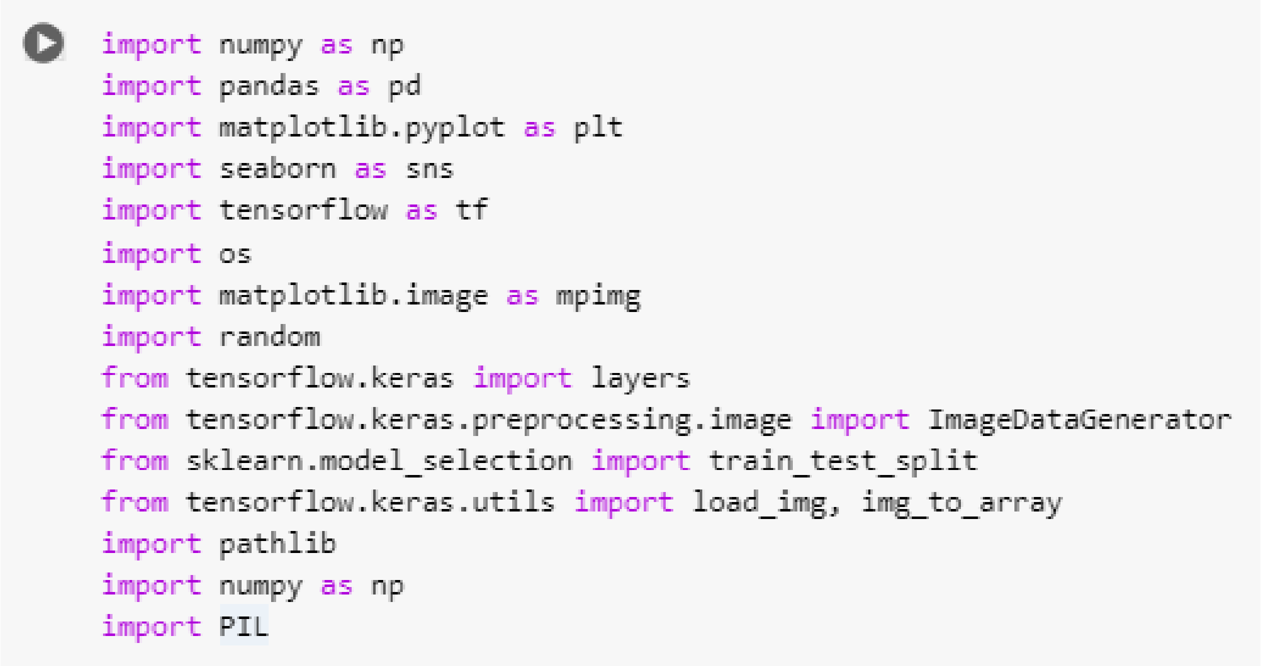
2.2 Development Tools and Software Environment

The selection of the tools that are used in the development purposes and the reproducibility of the set up environment is very much important, so this section discussed it completely about the set up tools we used in the fruit classification task.

2.2.1 Software and Libraries

As we discussed in the previous chapter the classification task we have to use the machine learning and deep learning techniques, so the software and libraries are using for the fruit classification task are given below:

- For programming language uses Python 3.7
- For developing of the machine learning models **TenserFlow** and **Keras** are used.
- For handling and manipulation of the data **NumPy** and Pandas are used.
- In order to visualize the data **Matplotlib** and **Seaborn** is used.
- For modeling tools **Scikit**-**learn** library is utilizing.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import os
import matplotlib.image as mpimg
import random
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import load_img, img_to_array
import pathlib
import numpy as np
import PIL
```

2.2.2 Version Control and Coding Standards

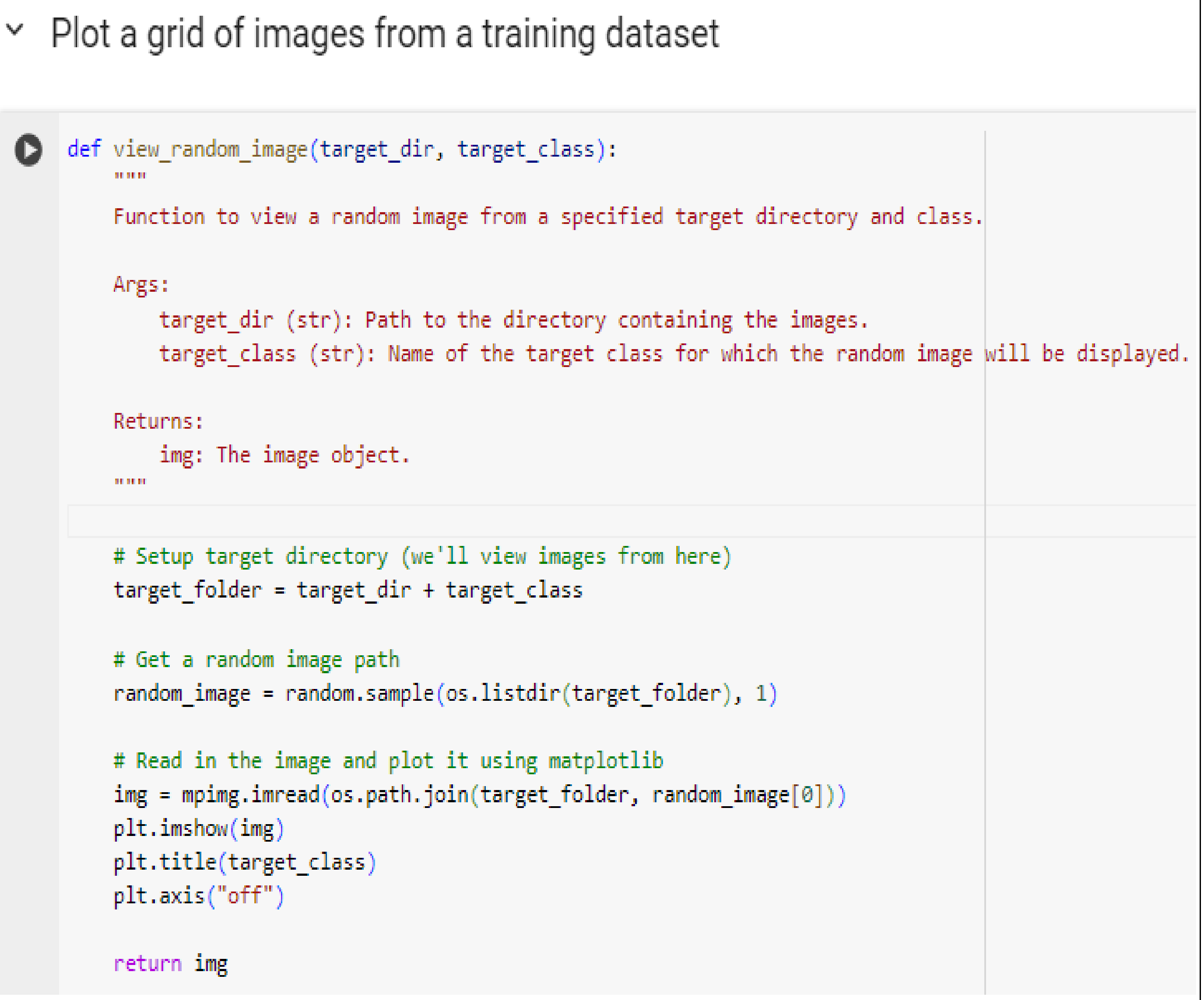
- In order to track the changes and manage collaboration Git is very common to use that’s why I am using, which is also quite helpful.
- Therefore, different repositories are used in order to organize the data in such a way having data in separate directories, code, documentations, and also code in the separate directories.
- PEP 8 confirms reliability and also be the consistency of the code base, which adherence the coding standards.

2.3 Data Collection and Dataset Overview

2.3.1 Dataset Source and Composition

- The dataset which is used is publicly available, which is gathered by the agricultural research institutes and also be the repositories having the collected data.
- Selecting the images for the fruit classification, must include the images having more diversity in fruit types, lighting and background’s variation required for sufficient representation of each of the class.
- To ensure an equal distribution of the different samples and fruit classifiers required data balancing techniques.

Now, there must be the Plot a grid of images from a training dataset is required, here below the code for performing that functionality;



Plot a grid of images from a training dataset

```
def view_random_image(target_dir, target_class):
    """
    Function to view a random image from a specified target directory and class.

    Args:
        target_dir (str): Path to the directory containing the images.
        target_class (str): Name of the target class for which the random image will be displayed.

    Returns:
        img: The image object.
    """

    # Setup target directory (we'll view images from here)
    target_folder = target_dir + target_class

    # Get a random image path
    random_image = random.sample(os.listdir(target_folder), 1)

    # Read in the image and plot it using matplotlib
    img = mpimg.imread(os.path.join(target_folder, random_image[0]))
    plt.imshow(img)
    plt.title(target_class)
    plt.axis("off")

    return img
```

2.3.2 Data Quality Assurance

- For achieving the consistency in image quality, resolution of the image, proper lighting conditions, and also background uniformity are steps needed in whole process.
- In order to remove and identify the low-quality images from the whole dataset, required manual as well as automated quality checks needed.

2.4 Preprocessing Techniques

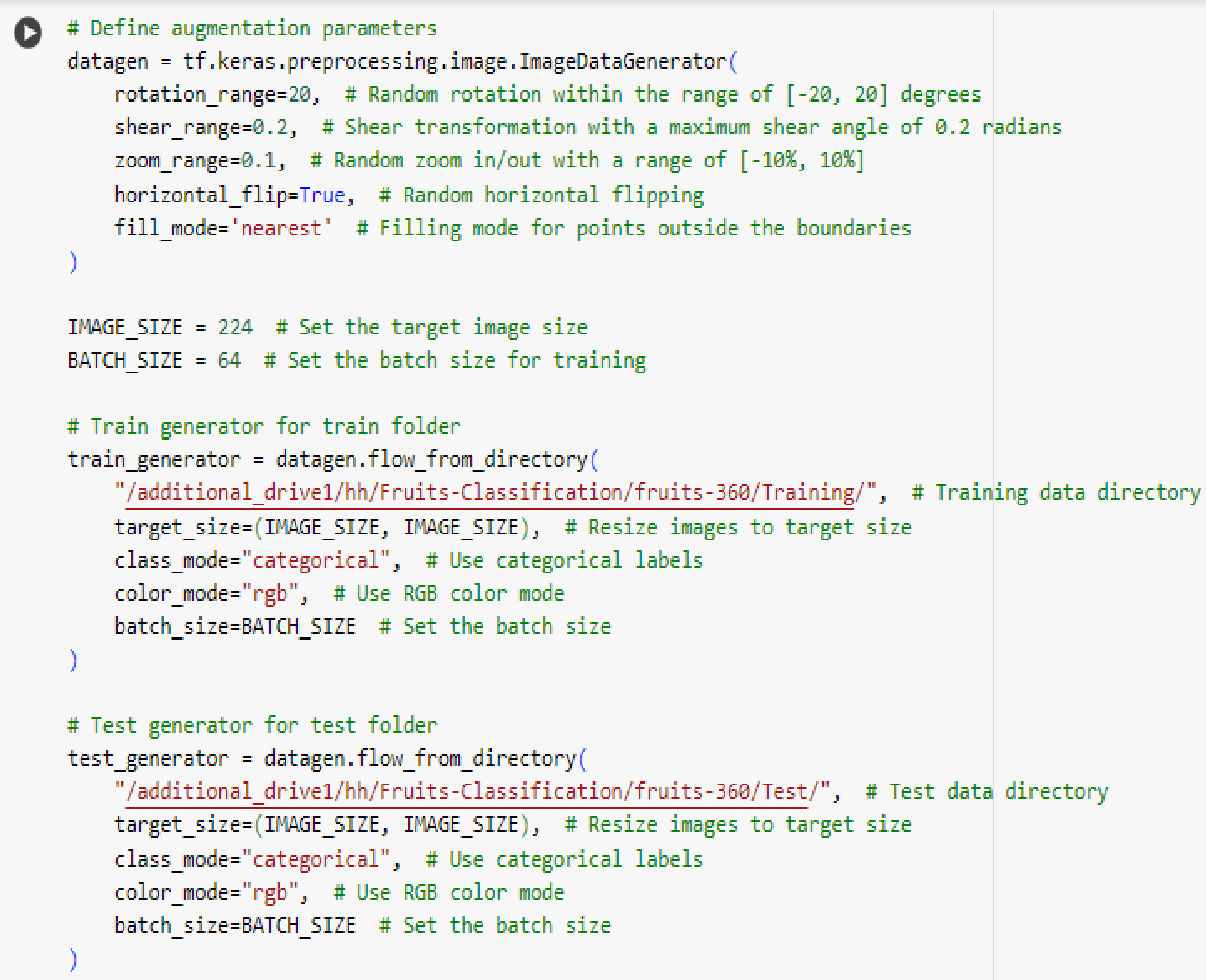
Preprocessing steps which are performed on the dataset, and their necessity, this section explain the whole process:

2.4.1 Resizing and Standardization

Standardizing resolution of the image and choosing a size of the specific also be explained earlier, by ensuring the uniformity in input data standardizing and also be the resizing were employed techniques. Data Preprocess and Augmentation: With these added parameters in the ImageDataGenerator, the data augmentation will include:

- Random rotation of images up to 20 degrees.
- Random zooming inside pictures up to 10%.
- Random horizontal flipping of images.

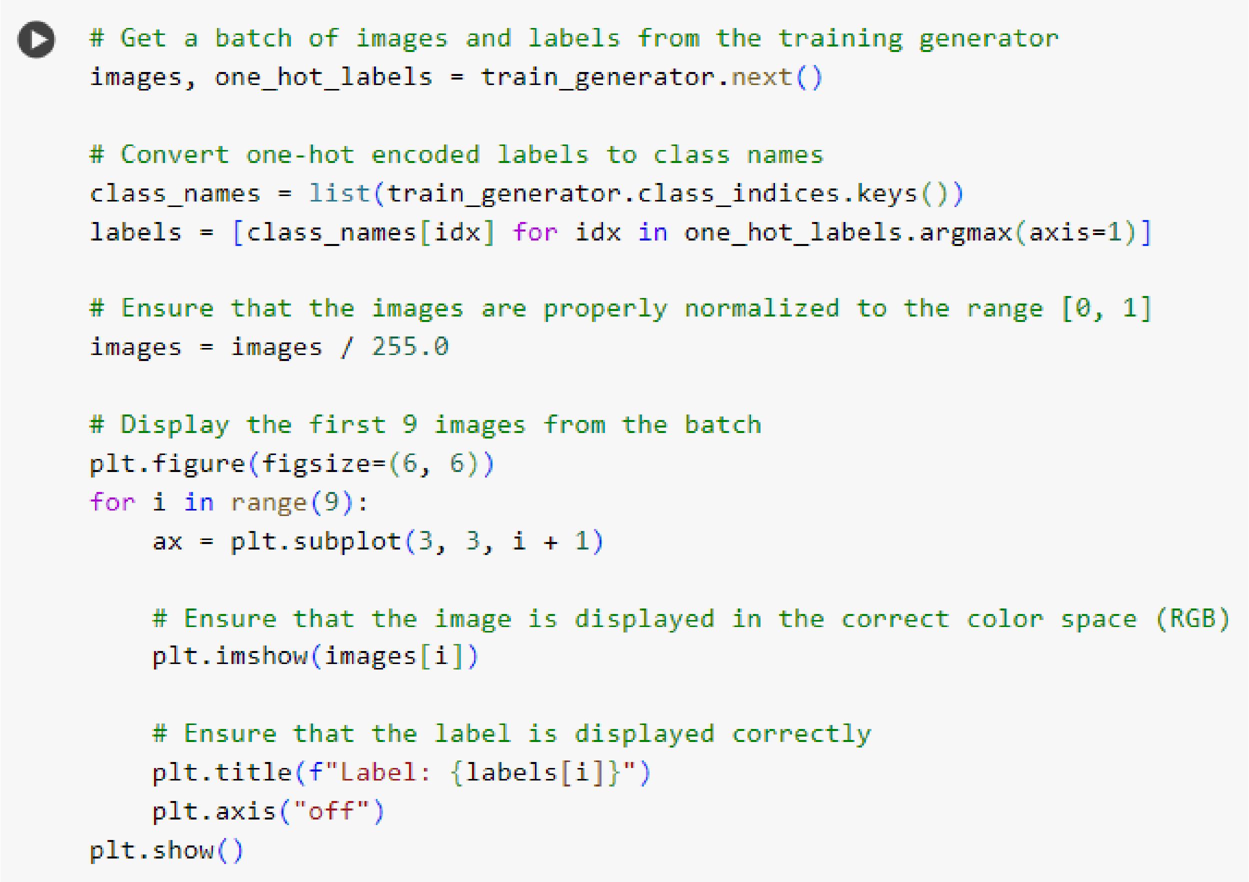
The fill mode parameter specifies how newly created pixels are filled.



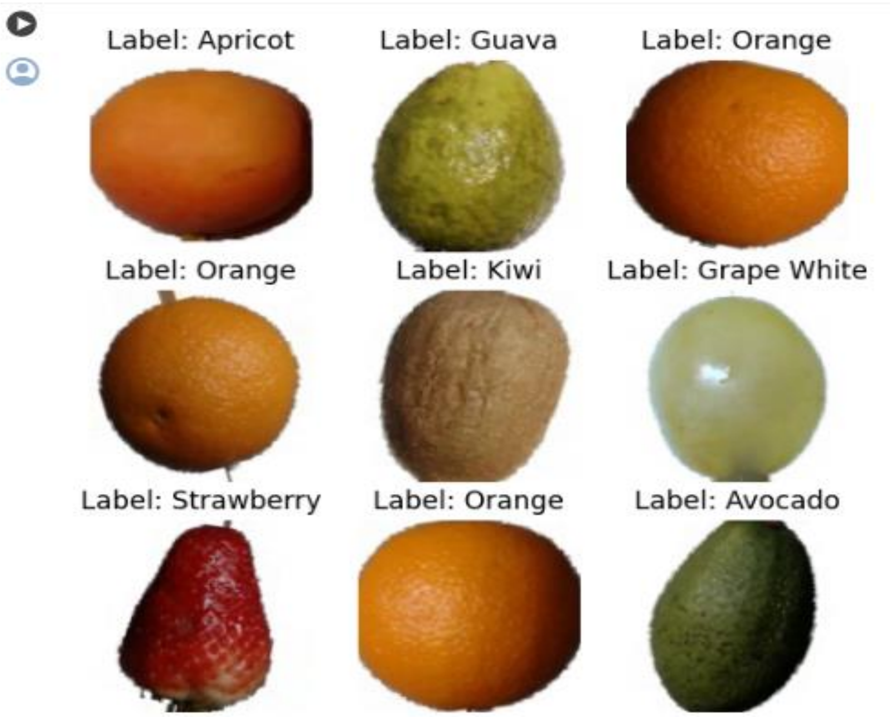
```
# Define augmentation parameters
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=20,  # Random rotation within the range of [-20, 20] degrees
    shear_range=0.2,  # Shear transformation with a maximum shear angle of 0.2 radians
    zoom_range=0.1,  # Random zoom in/out with a range of [-10%, 10%]
    horizontal_flip=True,  # Random horizontal flipping
    fill_mode='nearest'  # Filling mode for points outside the boundaries
)

IMAGE_SIZE = 224  # Set the target image size
BATCH_SIZE = 64  # Set the batch size for training

# Train generator for train folder
train_generator = datagen.flow_from_directory(
    "/additional_drive1/hh/Fruits-Classification/fruits-360/Training/",  # Training data directory
    target_size=(IMAGE_SIZE, IMAGE_SIZE),  # Resize images to target size
    class_mode="categorical",  # Use categorical labels
    color_mode="rgb",  # Use RGB color mode
    batch_size=BATCH_SIZE  # Set the batch size
)

# Test generator for test folder
test_generator = datagen.flow_from_directory(
    "/additional_drive1/hh/Fruits-Classification/fruits-360/Test/",  # Test data directory
    target_size=(IMAGE_SIZE, IMAGE_SIZE),  # Resize images to target size
    class_mode="categorical",  # Use categorical labels
    color_mode="rgb",  # Use RGB color mode
    batch_size=BATCH_SIZE  # Set the batch size
)
```

- Found **10705** images belonging to **23** classes.
- Found **3146** images belonging to **23** classes.

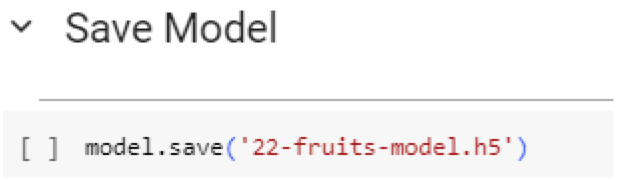
Now, after performing data preprocessing and augmentation here is the code and results given below;



```
# Get a batch of images and labels from the training generator
images, one_hot_labels = train_generator.next()

# Convert one-hot encoded labels to class names
class_names = list(train_generator.class_indices.keys())
labels = [class_names[idx] for idx in one_hot_labels.argmax(axis=1)]

# Ensure that the images are properly normalized to the range [0, 1]
images = images / 255.0

# Display the first 9 images from the batch
plt.figure(figsize=(6, 6))
for i in range(9):
    ax = plt.subplot(3, 3, i + 1)

    # Ensure that the image is displayed in the correct color space (RGB)
    plt.imshow(images[i])

    # Ensure that the label is displayed correctly
    plt.title(f"Label: {labels[i]}")
    plt.axis("off")
plt.show()
```



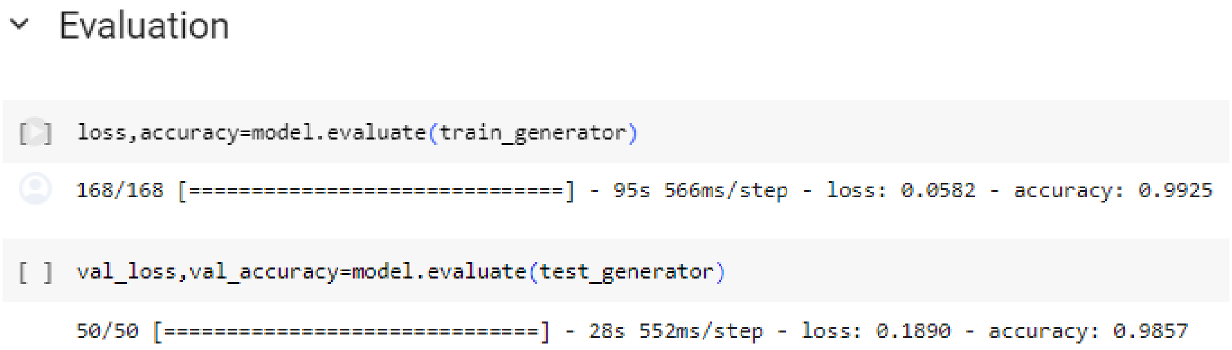
Saving the model with their weights are given below;



```
[ ]  model.save('22-fruits-model.h5')
```

/var/anaconda3/envs/hamza/lib/python3.11/site-packages/keras/src/engine/training.py:3103: User Warning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`. saving_api.save_model

Here, are the evaluations, we are performing the **train generator** and **test generator** are given below;



```
[ ]  loss,accuracy=model.evaluate(train_generator)

     168/168 [==============================] - 95s 566ms/step - loss: 0.0582 - accuracy: 0.9925

[ ]  val_loss,val_accuracy=model.evaluate(test_generator)

     50/50 [==============================] - 28s 552ms/step - loss: 0.1890 - accuracy: 0.9857
```

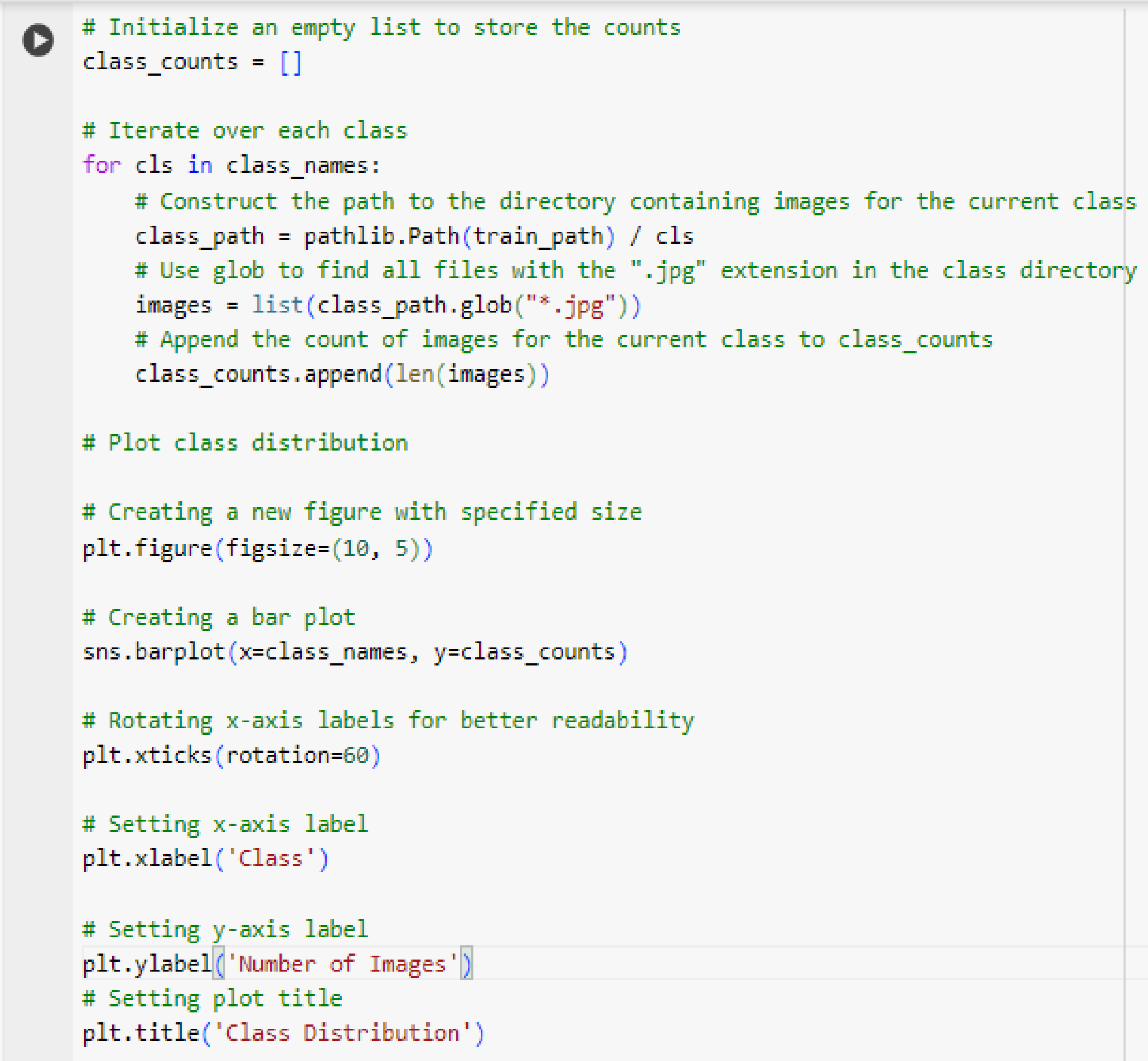
The model.evaluate function evaluates the CNN model's performance on the train_generator dataset. This function is likely to return two values: a loss metric (how well the model's predictions match the true labels) and an accuracy metric (the percentage of correct predictions). The evaluation results are stored in the variables loss and accuracy.

The evaluation process involved 50 steps (most likely iterating over **50 batches** of data) and took an average of **285 milliseconds** to complete per step.

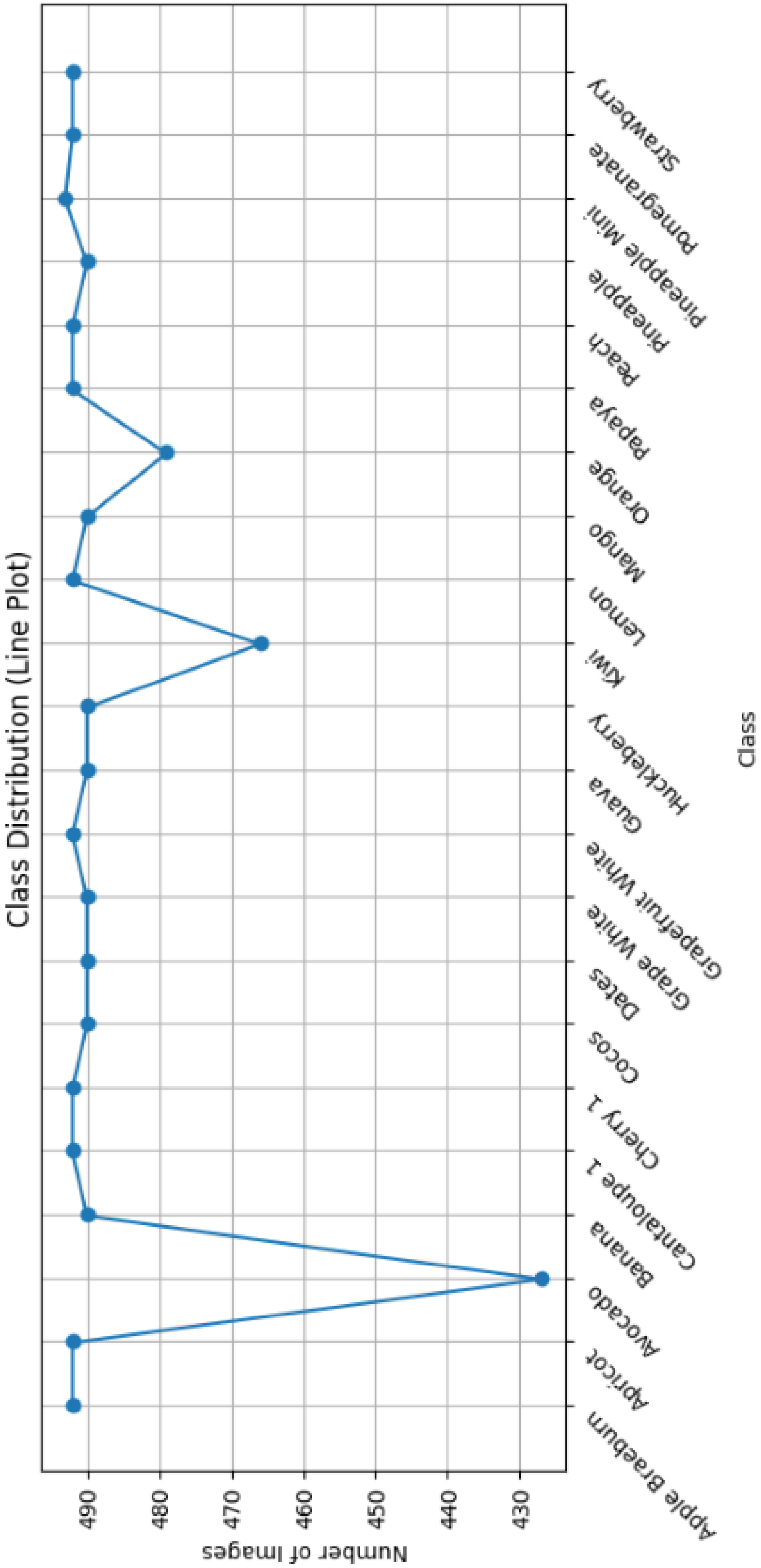
The evaluation yielded a loss of 0.1890 and an accuracy rate of **98.57%.** This indicates that the model performed well on the test data but not as well on the training data. This is common, as a model may become over fit to the training data.

2.4.2 Additional Preprocessing Steps

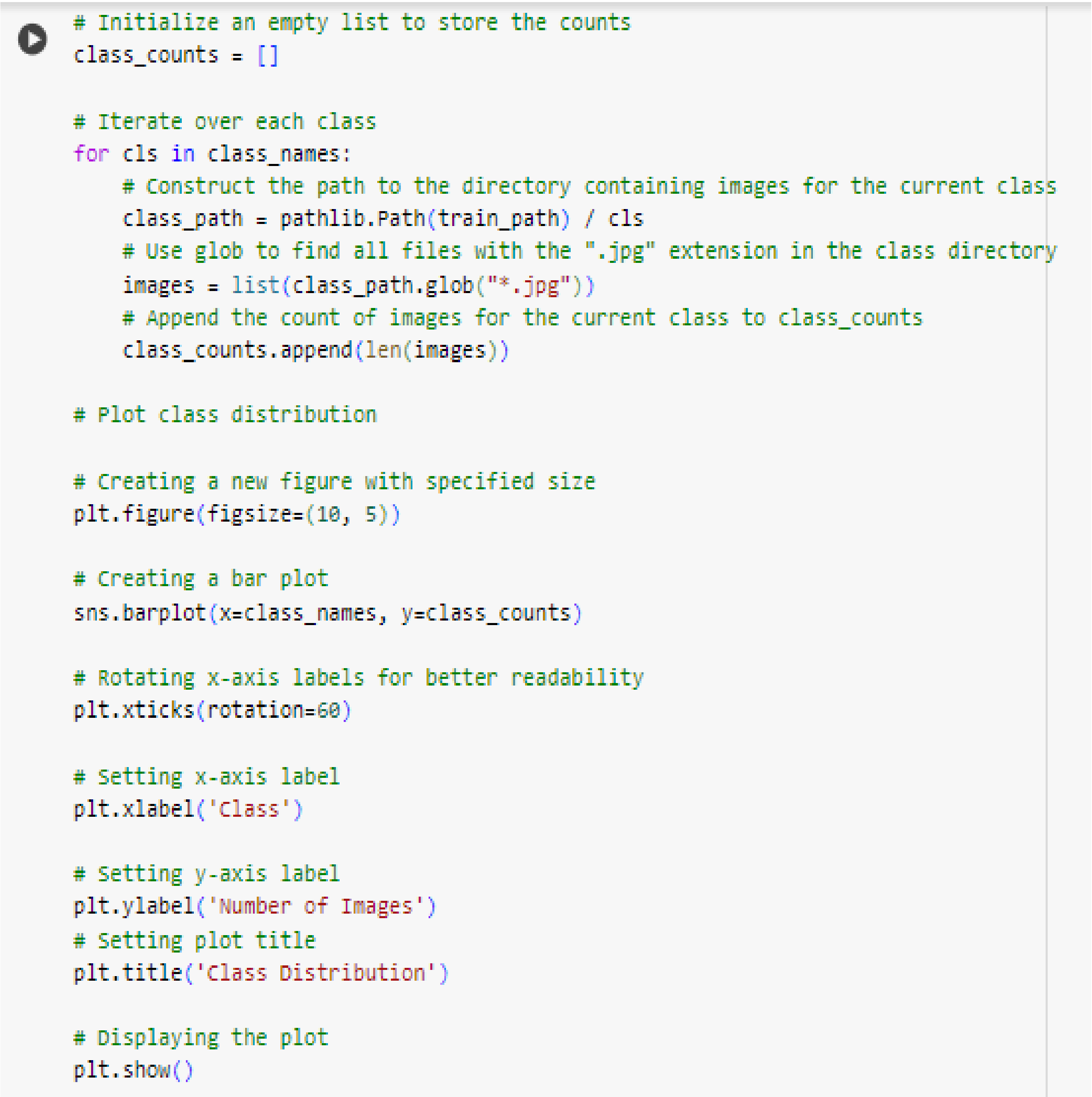
After standardizing and resizing of the image, there would also need to be additional preprocessing is required, such as normalization of color and reduction of the noise is also needed.



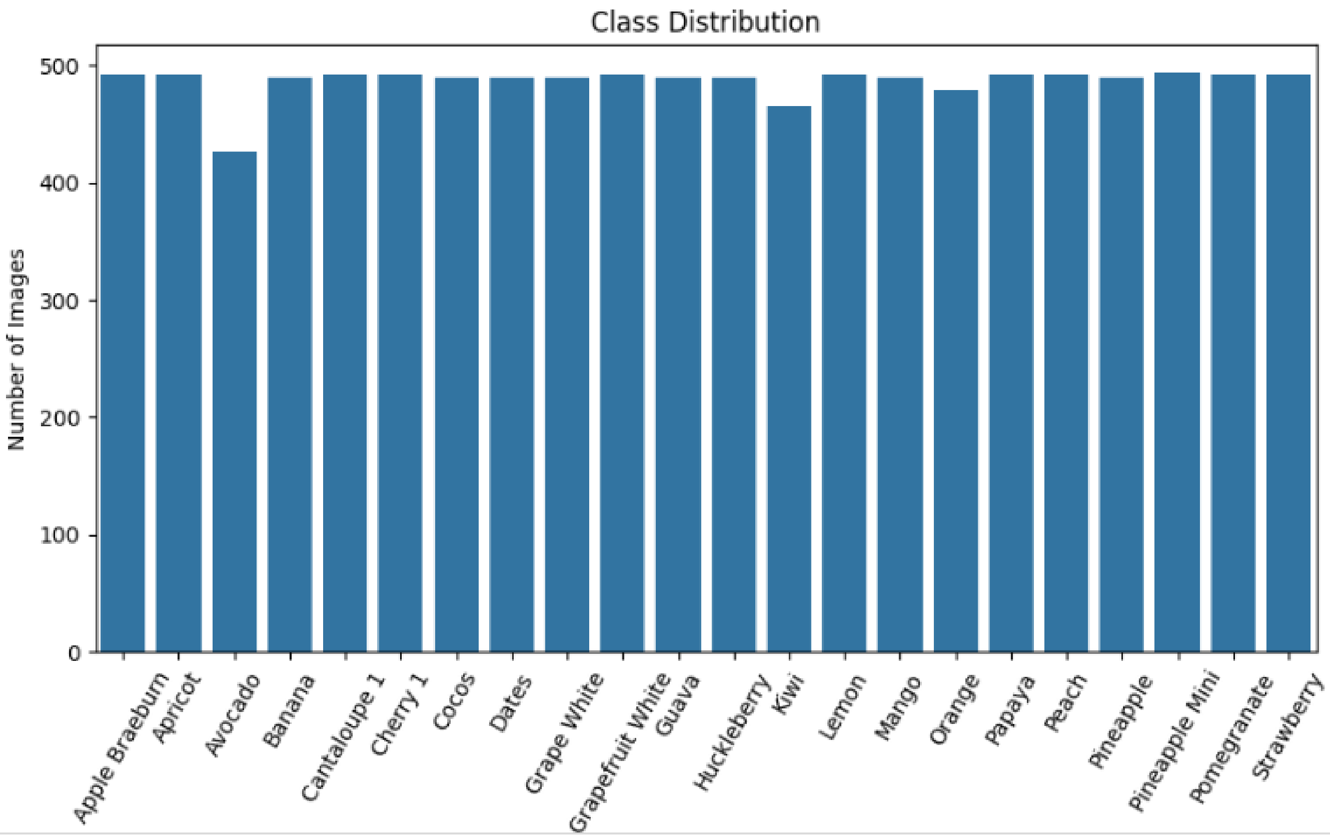
```
# Initialize an empty list to store the counts
class_counts = []

# Iterate over each class
for cls in class_names:
    # Construct the path to the directory containing images for the current class
    class_path = pathlib.Path(train_path) / cls
    # Use glob to find all files with the ".jpg" extension in the class directory
    images = list(class_path.glob("*.jpg"))
    # Append the count of images for the current class to class_counts
    class_counts.append(len(images))

# Plot class distribution

# Creating a new figure with specified size
plt.figure(figsize=(10, 5))

# Creating a bar plot
sns.barplot(x=class_names, y=class_counts)

# Rotating x-axis labels for better readability
plt.xticks(rotation=60)

# Setting x-axis label
plt.xlabel('Class')

# Setting y-axis label
plt.ylabel('Number of Images')
# Setting plot title
plt.title('Class Distribution')
```


Class Distribution (Line Plot)
Number of Images
Class
490
480
470
460
450
440
430
Apple Braeburn
Apricot
Avocado
Banana
Cantaloupe 1
Cherry 1
Cocos
Dates
Grape White
Grapefruit White
Guava
Huckleberry
Kiwi
Lemon
Mango
Orange
Papaya
Peach
Pineapple
Pineapple Mini
Pomegranate
Strawberry

Another, distribution of the class among as many of the fruits, here is the code below for the generation of the python code and also be the visualization available;



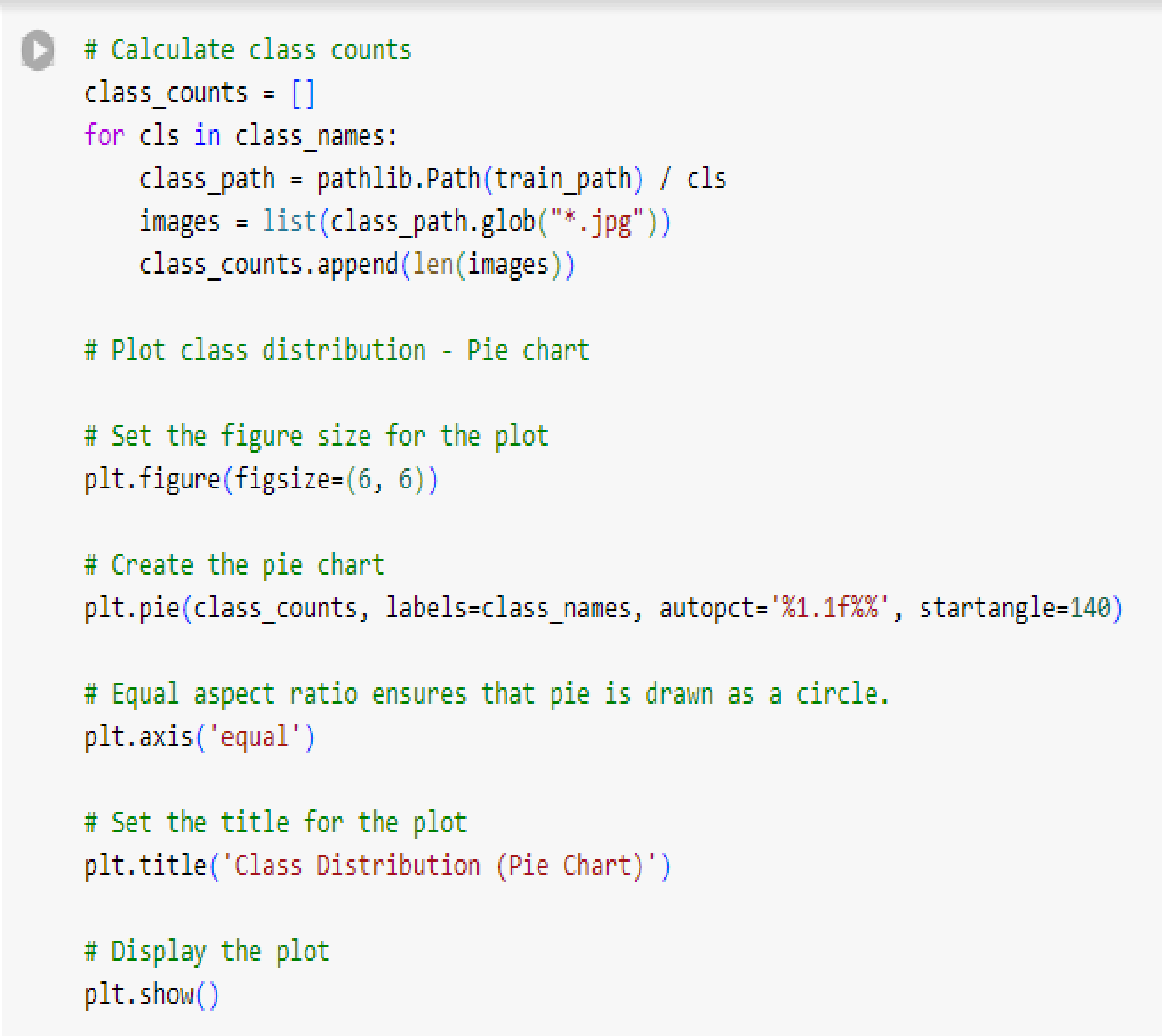
```python
# Initialize an empty list to store the counts
class_counts = []

# Iterate over each class
for cls in class_names:
    # Construct the path to the directory containing images for the current class
    class_path = pathlib.Path(train_path) / cls
    # Use glob to find all files with the ".jpg" extension in the class directory
    images = list(class_path.glob("*.jpg"))
    # Append the count of images for the current class to class_counts
    class_counts.append(len(images))

# Plot class distribution

# Creating a new figure with specified size
plt.figure(figsize=(10, 5))

# Creating a bar plot
sns.barplot(x=class_names, y=class_counts)

# Rotating x-axis labels for better readability
plt.xticks(rotation=60)

# Setting x-axis label
plt.xlabel('Class')

# Setting y-axis label
plt.ylabel('Number of Images')
# Setting plot title
plt.title('Class Distribution')

# Displaying the plot
plt.show()
```



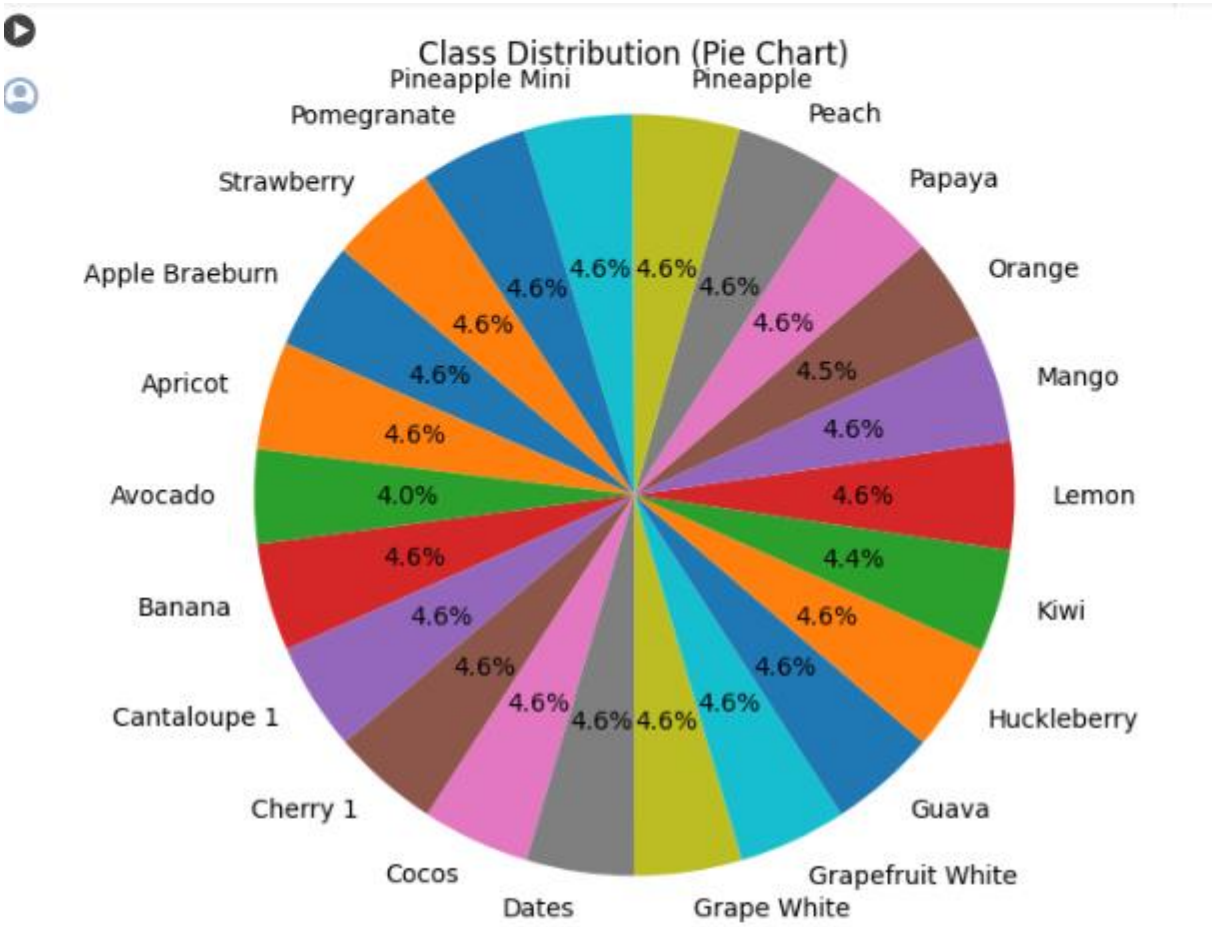
Now, findings which are fetching from the pie plot of the distribution are given below;

- The main finding is that the number of courts varies significantly by region, with far more courts in some regions compared to others.
- The Americas region has the most courts, with over 153,000 courts, which is much higher than any other region (44,164% more than Oceania, which has the least with only 347 courts).
- The total number of courts across all regions ranges from a low of 347 to a high of 153,597.

Now, there is another visualization for the class distribution by the usage of the python code, and the pie charts are given below;



```
# Calculate class counts
class_counts = []
for cls in class_names:
    class_path = pathlib.Path(train_path) / cls
    images = list(class_path.glob("*.jpg"))
    class_counts.append(len(images))

# Plot class distribution - Pie chart

# Set the figure size for the plot
plt.figure(figsize=(6, 6))

# Create the pie chart
plt.pie(class_counts, labels=class_names, autopct='%1.1f%%', startangle=140)

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Set the title for the plot
plt.title('Class Distribution (Pie Chart)')

# Display the plot
plt.show()
```



2.5 Model Architecture

For elucidate the design of the architecture of the model which is required for the fruit classification

2.5.1 Baseline Model

Baseline model for the classification of the fruit required initial model layers, their parameters, rationale which acting behind the section were also presented over there.

2.5.2 Proposed Model Improvements

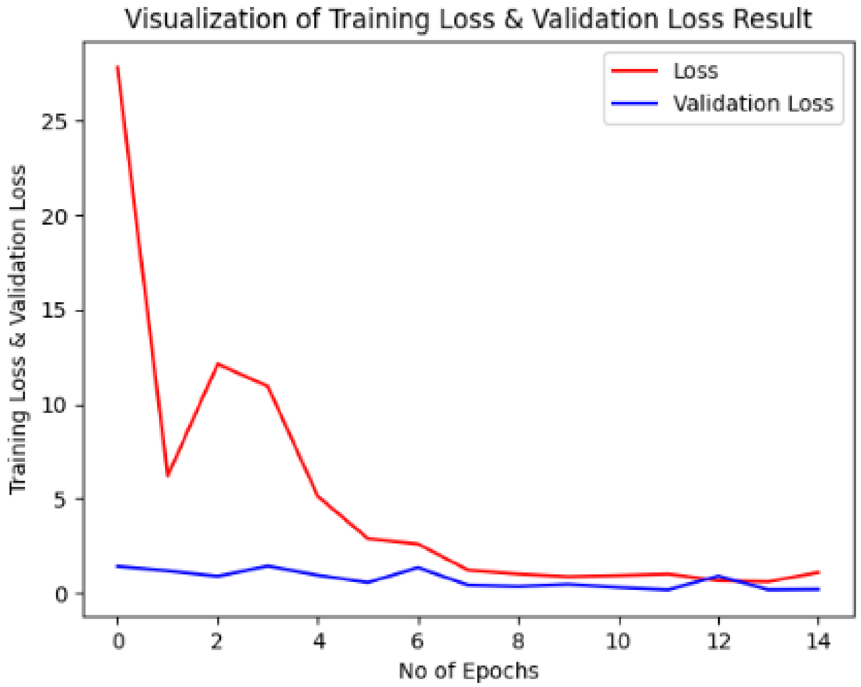
Reasoning behind the changes which are performed in the initial baseline model are added layers or dropout from the layer, also discussed about the effect on the performance of the model.

2.6 Model Training and Validation

The training and validation for the fruit classification model are discussed in this section.

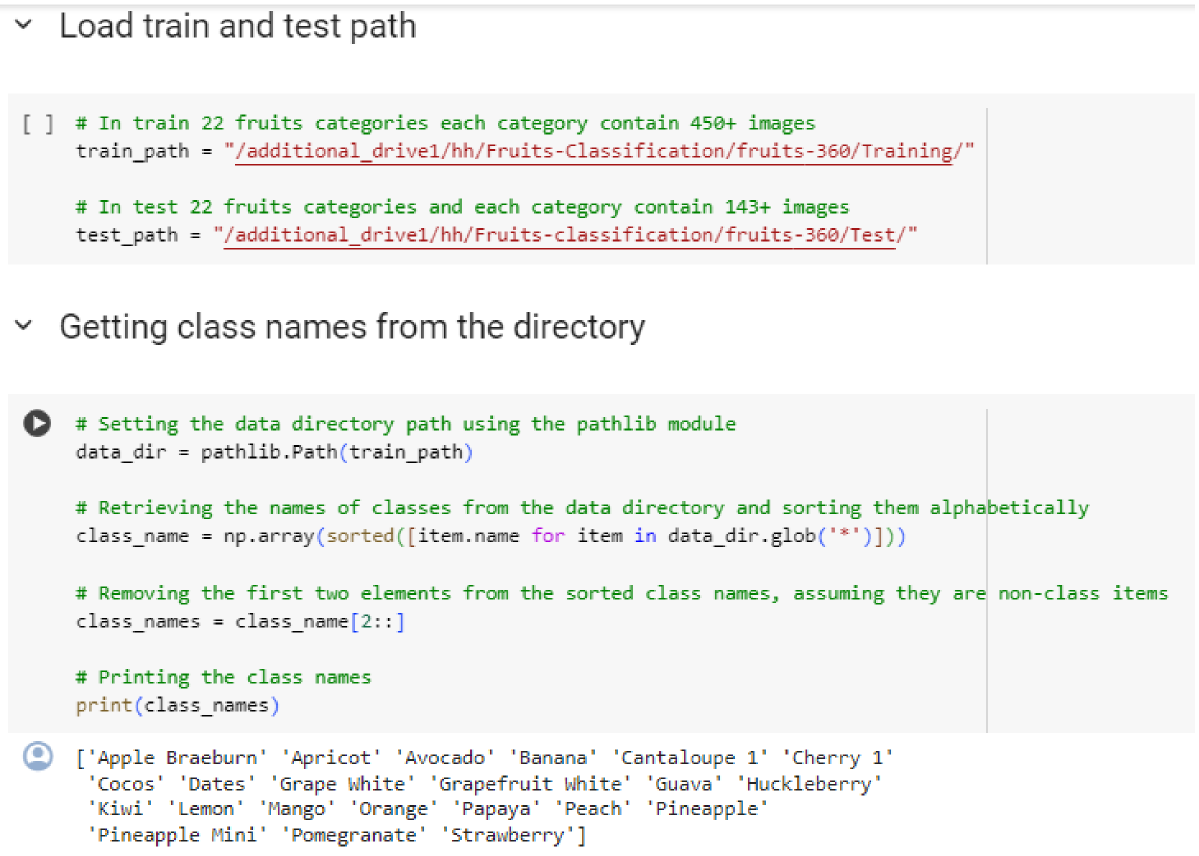
2.6.1 Training Process

Adam optimizer, loss function, and regularization techniques are utilizing in the training process are also be discussed.



2.6.2 Validation Methodology

To prevent the overfitting there must be the methodology of model's validation, usage of the accuracy of model as a metric, and strategies are also employed.



Load train and test path

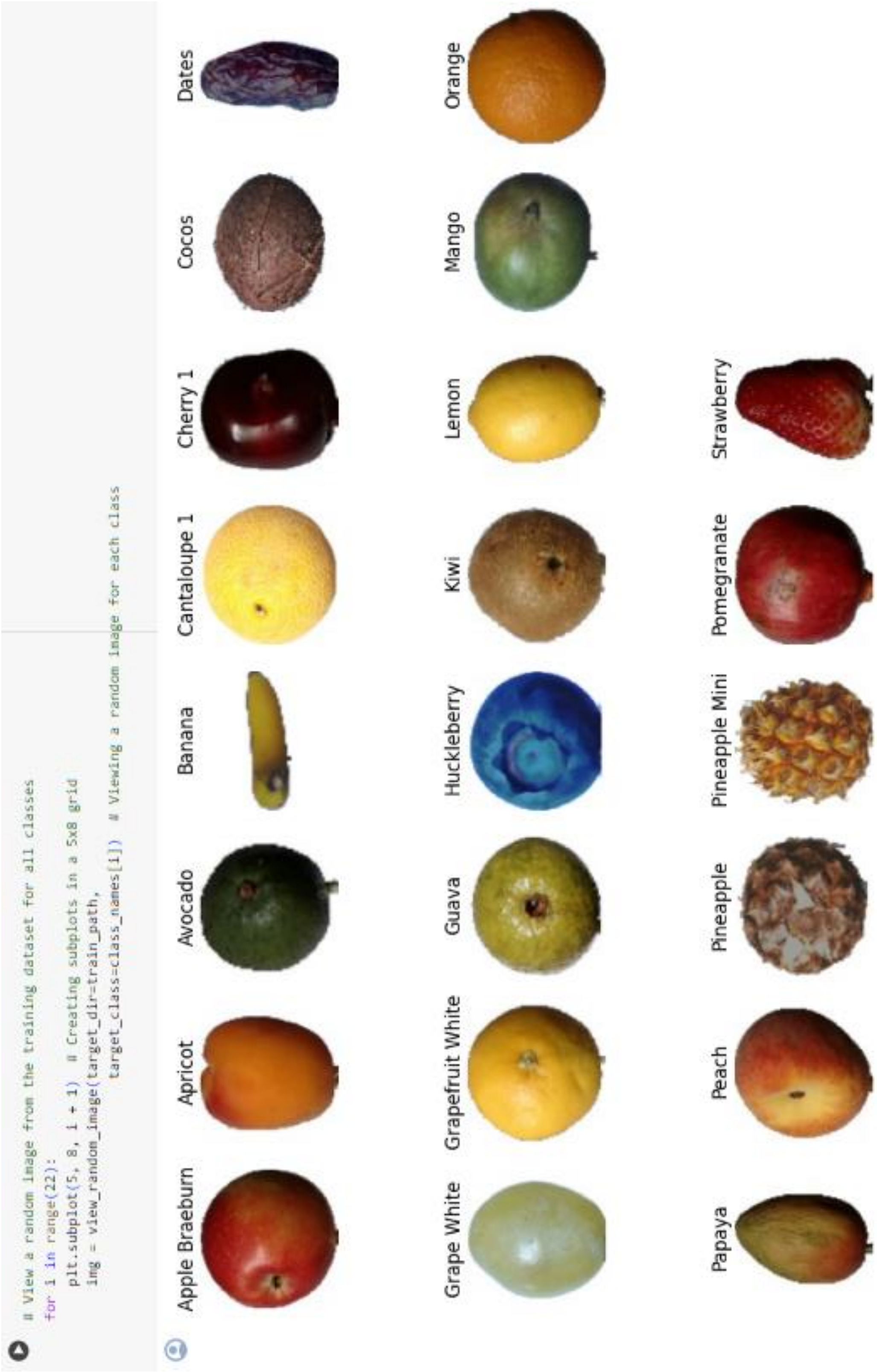
```
[ ] # In train 22 fruits categories each category contain 450+ images
    train_path = "/additional_drive1/hh/Fruits-Classification/fruits-360/Training/"

    # In test 22 fruits categories and each category contain 143+ images
    test_path = "/additional_drive1/hh/Fruits-classification/fruits-360/Test/"
```

Getting class names from the directory

```
# Setting the data directory path using the pathlib module
data_dir = pathlib.Path(train_path)

# Retrieving the names of classes from the data directory and sorting them alphabetically
class_name = np.array(sorted([item.name for item in data_dir.glob('*')]))

# Removing the first two elements from the sorted class names, assuming they are non-class items
class_names = class_name[2::]

# Printing the class names
print(class_names)
```

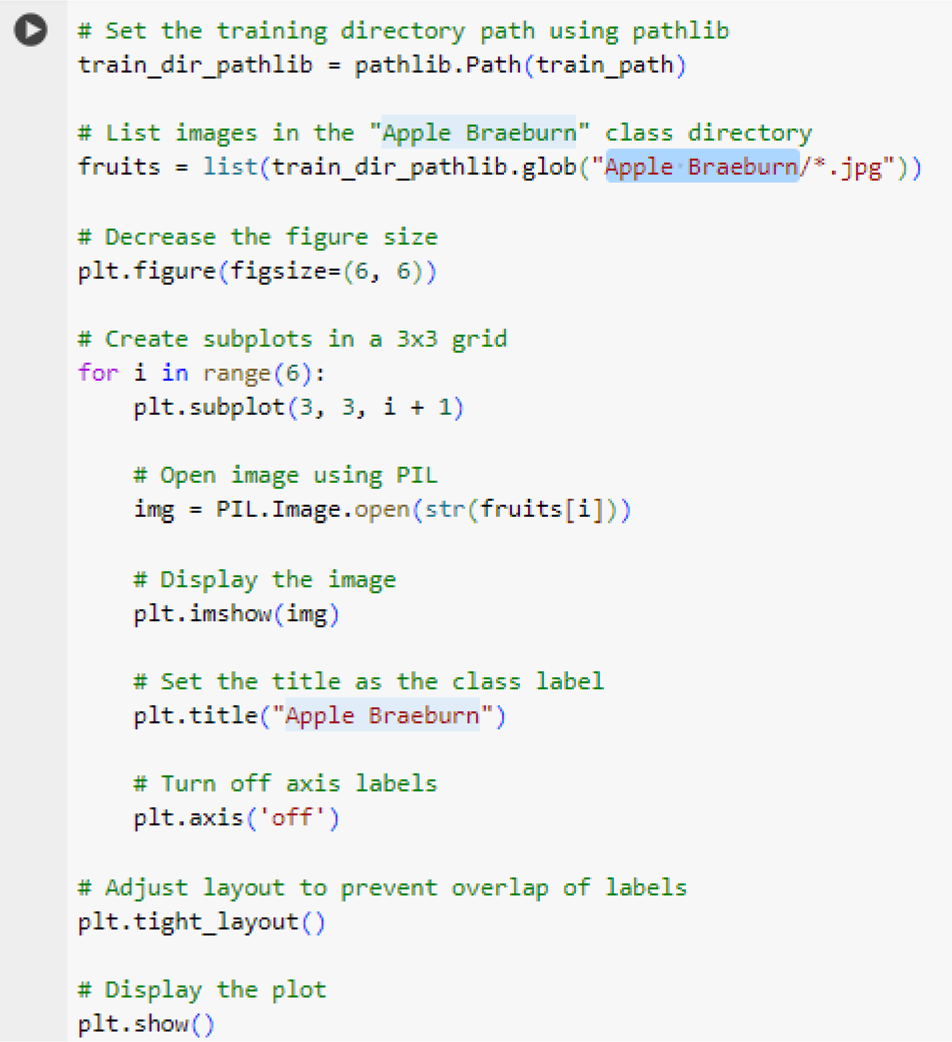
```
['Apple Braeburn' 'Apricot' 'Avocado' 'Banana' 'Cantaloupe 1' 'Cherry 1'
 'Cocos' 'Dates' 'Grape White' 'Grapefruit White' 'Guava' 'Huckleberry'
 'Kiwi' 'Lemon' 'Mango' 'Orange' 'Papaya' 'Peach' 'Pineapple'
 'Pineapple Mini' 'Pomegranate' 'Strawberry']
```

Now, after loading the complete dataset, provide the complete figures for the each of the fruit, here are code and their results are given below;

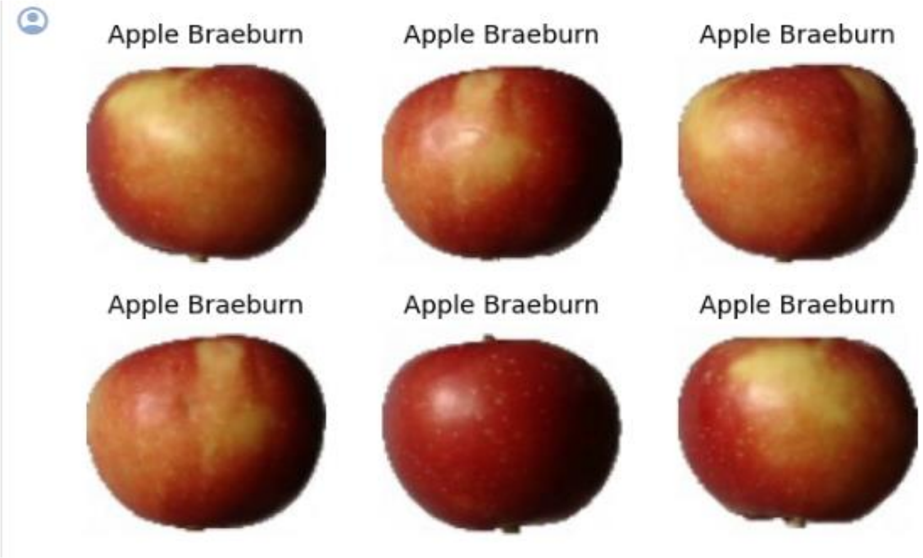


```
# View a random image from the training dataset for all classes
for i in range(22):
    plt.subplot(5, 8, i + 1)  # Creating subplots in a 5x8 grid
    img = view_random_image(target_dir=train_path,
                            target_class=class_names[i])  # Viewing a random image for each class
```

Now, taking these train path, further move on **Apple Braeburn,** and show the results according to the dataset, also code and their results are given below;



```
# Set the training directory path using pathlib
train_dir_pathlib = pathlib.Path(train_path)

# List images in the "Apple Braeburn" class directory
fruits = list(train_dir_pathlib.glob("Apple Braeburn/*.jpg"))

# Decrease the figure size
plt.figure(figsize=(6, 6))

# Create subplots in a 3x3 grid
for i in range(6):
    plt.subplot(3, 3, i + 1)

    # Open image using PIL
    img = PIL.Image.open(str(fruits[i]))

    # Display the image
    plt.imshow(img)

    # Set the title as the class label
    plt.title("Apple Braeburn")

    # Turn off axis labels
    plt.axis('off')

# Adjust layout to prevent overlap of labels
plt.tight_layout()

# Display the plot
plt.show()
```



2.7 Implementation Strategy

The most importantly know the strategy, from which real-world application, how the fruit classification model is dealing in the real-life scenario, this will be done by creating the integrating model into a production environment.

2.8 Challenges Encountered

The challenges facing during the fruit classification, are offered insights into the research process, discussing the challenges such as misbalancing of the data, computational limitations, are addressed, and to overcome these challenges is very important.

2.9 Summary of Methodological Approach

Baseline model for the classification of the fruit required initial model layers, their parameters, also making the improvements in this baseline model is needed, and throughout the most important thing is that training of the model on the dataset is required and then fine tuning the model on the refined data.

To prevent the overfitting there must be the methodology of model’s validation, is also be performed by usage of accuracy as the separate metric.

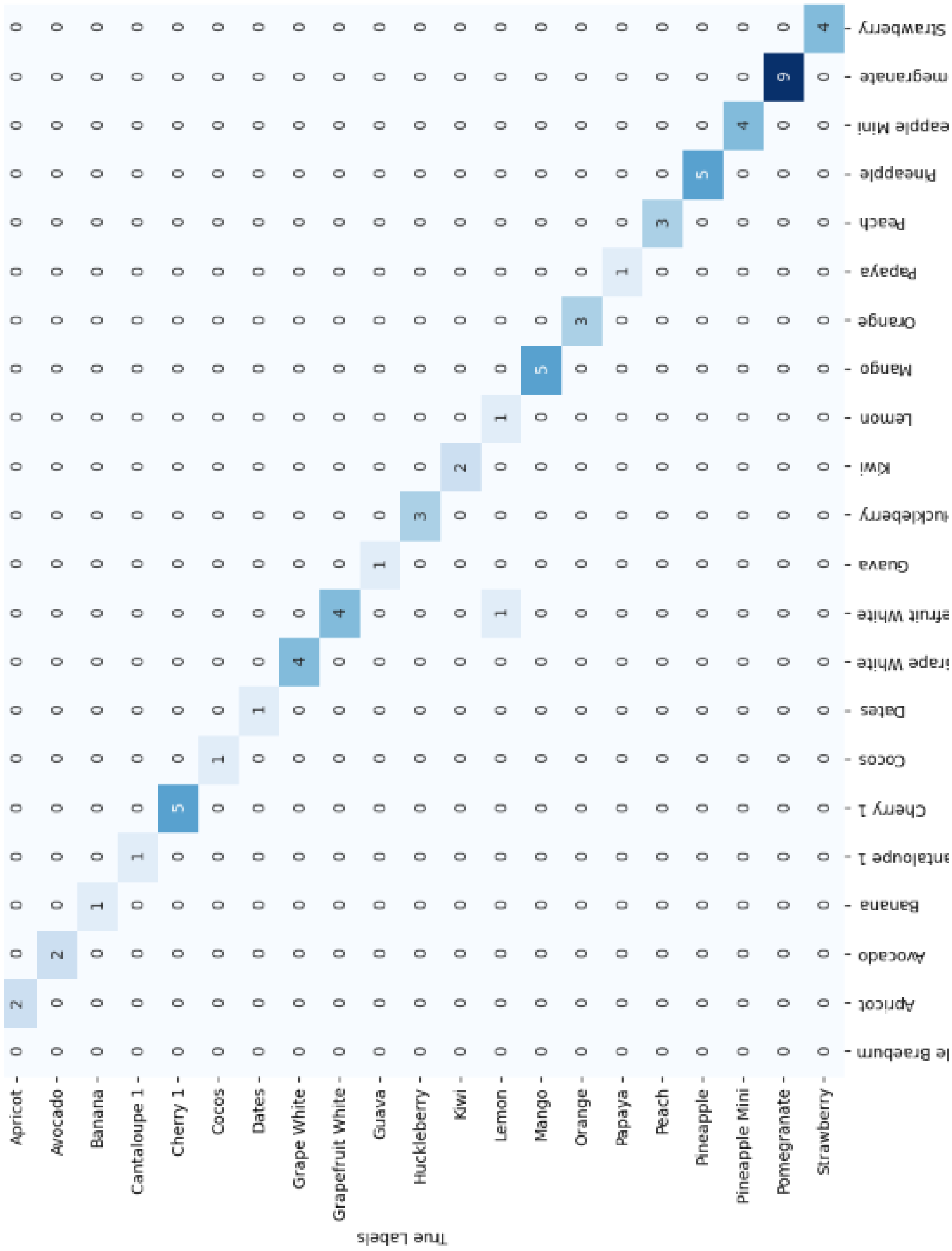
Chapter 3: Results

3.1. Qualitative Evaluation

- Accuracy, precision, and also be the F1-score are utilizing as the Quantitative metrics.
- Visualization of training and validation, making the result for the confusion matrix, and analysis of the accuracy metrics, are also provided in the form of the evaluation of the qualitative metrics.

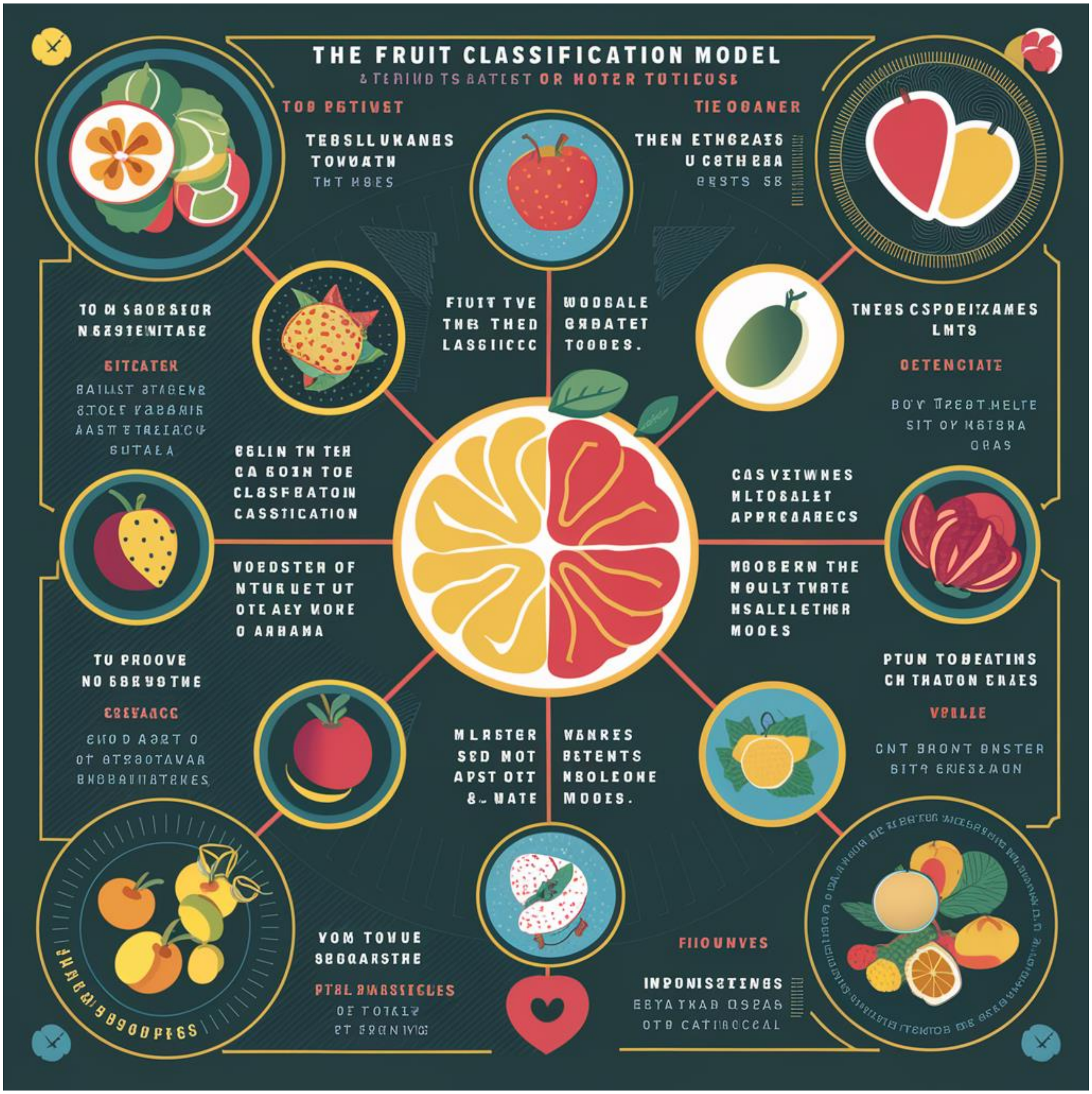
```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         2
           3       1.00      1.00      1.00         2
           4       1.00      1.00      1.00         1
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         5
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         1
           9       1.00      1.00      1.00         4
          10       0.80      1.00      0.89         4
          11       1.00      1.00      1.00         1
          12       1.00      1.00      1.00         3
          13       1.00      1.00      1.00         2
          14       1.00      0.50      0.67         2
          15       1.00      1.00      1.00         5
          16       1.00      1.00      1.00         3
          17       1.00      1.00      1.00         1
          18       1.00      1.00      1.00         3
          19       1.00      1.00      1.00         5
          20       1.00      1.00      1.00         4
          21       1.00      1.00      1.00         9
          22       1.00      1.00      1.00         4

    accuracy                           0.98        64
   macro avg       0.99      0.98      0.98        64
weighted avg       0.99      0.98      0.98        64
```

Confusion Matrix



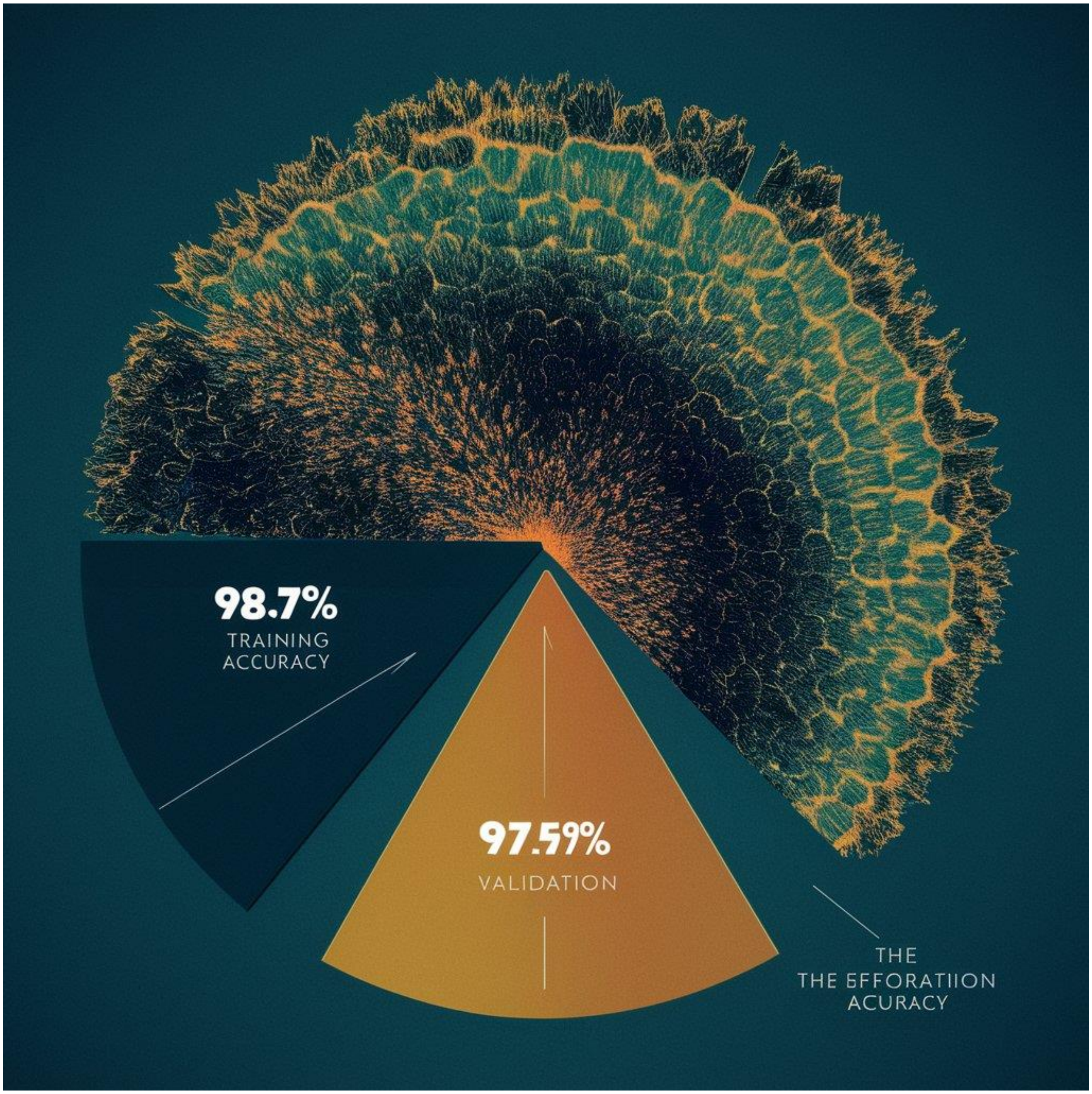
3.2. Discussion

- Performance of the Model in the fruit classification task, depends on the clarification of the qualitative evaluation.
- In order to highlights the effectiveness of the CNN model, there must be the comparison required with other existing models and also be other approaches that are attempted without the modern approaches.
- There is always be the room for the improvement, so suggesting area for the improvements, and also indicates the direction of the future researches.



3.3. Results

- As a result, the effectiveness of the CNN model, is main findings, CNN with the accuracy of the 98.7%, and validation for the accuracy of the 97.6%. Here are the results of above findings;



- The better result is also being achieved, if we provide the good quality of the dataset, and with their diversity involved, plays the huge role on performance.

Chapter 4: Discussion

4.1. Conclusion

In this study, we established the effectiveness of the model, which is classifying by maintaining the high accuracy by using the diverse range of fruit types. In order to include the model's performance in the classification task, required the quantitative evaluation results, includes both of the training and validation accuracy, precision, and also includes F1-score.

The suggestion for this study, includes the followings:

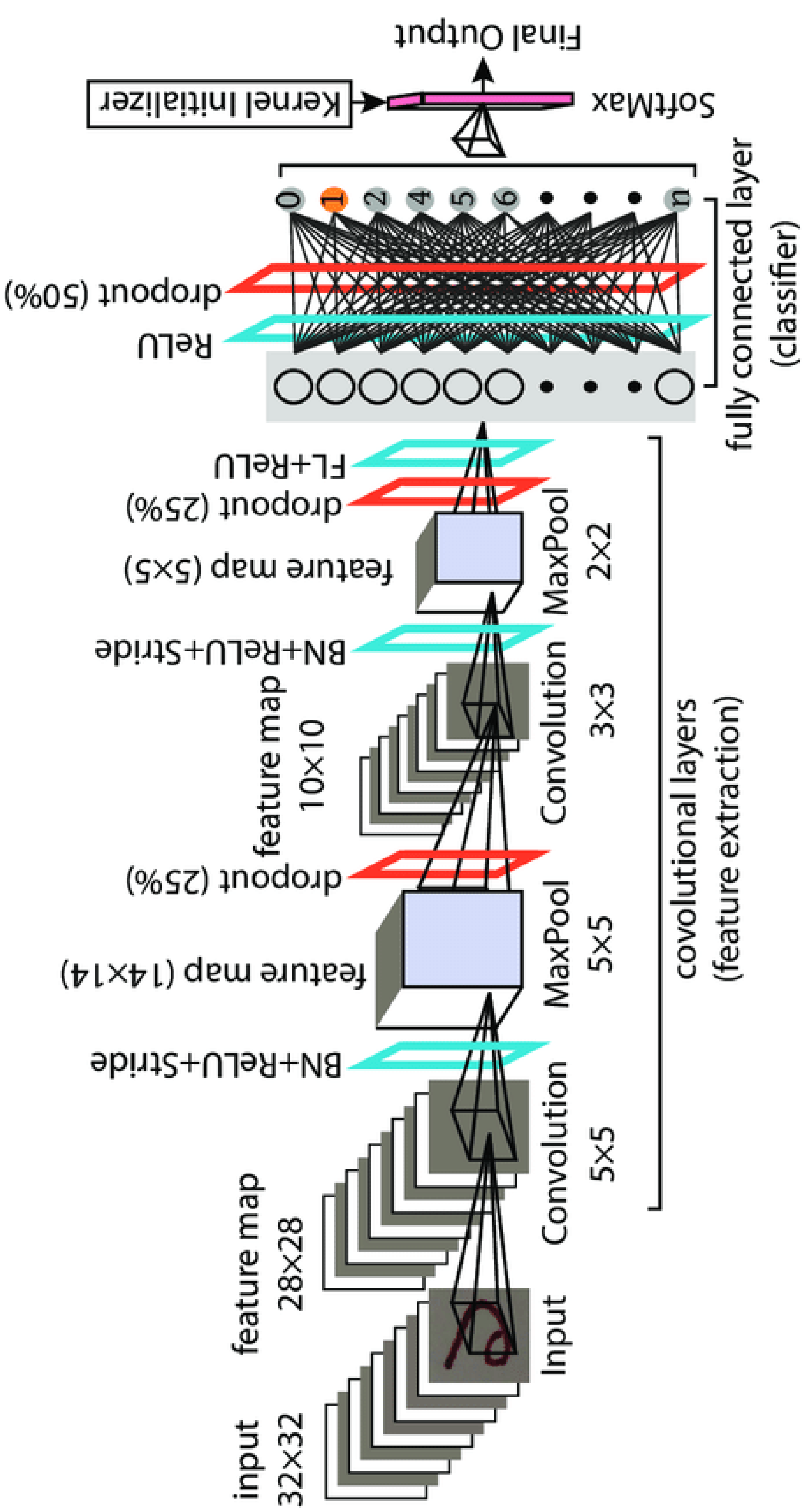
- Quality and diversity in the dataset.
- Automate the recognition of the fruit.
- Quality control in order to perform classification.

4.2. Idea for Future Work

- Advancing the data augmentation technique, includes **Generative Adversarial Networks** (GANs) or **Style Transfer,** which is supposed to increase the diversity of the data during training time.
- Exploration of the fine-tuning and transfer learning, by the use of pre-trained model such as **ImageNet-trained models,** then specify it towards the fine tuning process but the pre-trained models used as the starting point.
- Deployment of the techniques of the ensembles learning, such as the bagging or boosting, which stores the results or predictions of the multiples **CNN** models, which achieve the better performance.
- Refinement of the model according to the specific domain, which is quite helpful, because we have to predict the ripeness or the diseases of the fruit.

Here below the image of the CNN, which we are supposed to be performed in the future analysis, in which each of the layer is presenting the;

- **Input layer**
- **Convolutional layer 5 X 5**
- **Maxpooling 5 X 5**
- **Convolutional layer 3 X 3**
- **Maxpooling 2 X 2**
- **Fully connected layer**


input
32×32
feature map
28×28
Input
Convolution
5×5
BN+ReLU+Stride
feature map (14×14)
MaxPool
5×5
dropout (25%)
feature map
10×10
Convolution
3×3
BN+ReLU+Stride
feature map (5×5)
MaxPool
2×2
dropout (25%)
FL+ReLU
covolutional layers
(feature extraction)
ReLU
dropout (50%)
0
1
2
4
5
6
n
fully connected layer
(classifier)
SoftMax
Kernel Initializer
Final Output

4.3. References

TensorFlow: A System for Large-Scale Machine Learning

- Authors: Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M.
- Publication: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265-283.
- Summary: This paper introduces TensorFlow, a comprehensive open-source platform for machine learning. It emphasizes TensorFlow's scalability, flexibility, and ease of use, making it suitable for both research and production environments. The system is designed to support a wide range of machine learning and deep learning models, facilitating the development of complex models with ease.

Keras: Deep Learning Library for Theano and TensorFlow

- Author: Chollet, F.
- Publication: GitHub. Retrieved from https://github.com/fchollet/keras.
- Summary: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Keras allows for easy and fast prototyping, supports both convolutional networks and recurrent networks, as well as combinations of the two, and runs seamlessly on both CPU and GPU.

Support-Vector Networks

- Authors: Cortes, C., & Vapnik, V.
- Publication: Machine learning, 20(3), 273-297.
- Summary: This paper introduces support-vector networks (SVMs), a set of supervised learning methods used for classification, regression, and outliers detection. SVMs are effective in high-dimensional spaces and are versatile as different Kernel functions can be specified for the decision function.

Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors

- Authors: Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.
- Publication: arXiv preprint arXiv:1207.0580.
- Summary: This paper presents a method to improve the performance of neural networks by preventing the co-adaptation of feature detectors. The authors introduce a technique called dropout, which randomly sets a fraction of input units to 0 at each update during training time, helping to prevent overfitting and improve generalization.

Scikit-learn: Machine Learning in Python

- Authors: Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J.
- Publication: Journal of machine learning research, 12(Oct), 2825-2830.
- Summary: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression, and clustering algorithms, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Visualizing and Understanding Convolutional Networks

- Authors: Zeiler, M. D., & Fergus, R.
- Publication: European conference on computer vision (pp. 818-833). Springer, Cham.
- Summary: This paper discusses the challenges and methods for visualizing and understanding the inner workings of convolutional neural networks (CNNs). It presents techniques for visualizing the features learned by CNNs and for understanding the decision-making process of these networks.

Convolutional Neural Network Based Rotten Fruit Detection Using ResNet50

- Authors: Foong, C.C.; Meng, G.K.; Tze, L.L.
- Publication: Proceedings of the 2021 IEEE 12th Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia, 7 August 2021.
- Summary: This paper presents a method for detecting rotten fruits using a convolutional neural network (CNN) based on the ResNet50 architecture. The approach leverages transfer learning to improve the accuracy of fruit freshness classification, demonstrating the effectiveness of deep learning techniques in practical applications.

Classification of Skin Lesions Using Transfer Learning and Augmentation with Alex-net

- Authors: Hosny, K.M.; Kassem, M.A.; Foaud, M.M.
- Publication: PLoS ONE, 14, e0217293.
- Summary: This study applies transfer learning and data augmentation techniques to classify skin lesions using the AlexNet architecture. The methodology demonstrates the potential of deep learning in medical image analysis, showcasing the effectiveness of transfer learning in leveraging pre-trained models for new tasks.

A Comparative Analysis on Fruit Freshness Classification

- Authors: Karakaya, D.; Ulucan, O.; Turkan, M.
- Publication: Proceedings of the 2019 Innovations in Intelligent Systems and Applications Conference (ASYU), Izmir, Turkey, 31 October–2 November 2019.
- Summary: This paper conducts a comparative analysis of various methods for fruit freshness classification. It evaluates the performance of different machine learning algorithms and deep learning models in accurately determining the freshness of fruits, highlighting the importance of choosing the right model for specific classification tasks.