National College of Ireland

Configuration Manual

MSc Research Project

MSc in Data Analytics

Janush Prajosh

Student ID: 22132732

School of Computing

National College of Ireland

Supervisor: Arjun Chikkankod

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Janush Prajosh		
Student ID:	22132732		
Programme:	MSc in Data Analytics	**Year:**	2023 - 2024
Module:	MSc in Research Project		
Lecturer:	Arjun Chikkankod		
Submission Due Date:	14-12-2023		
Project Title:	Identification of Rotten Fruits using Deep Learning Techniques		
Word Count:	1524	**Page Count:**	20

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: Janush Prajosh

Date: 15-12-2023

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):	

Configuration Manual

Janush Prajosh

Student ID: 22132732

1 Introduction

This document provides a guide on how to reproduce the findings mentioned in the project report. It outlines the tools for conducting experiments. This manual provides a detailed explanation of the steps involved in executing the code.

2 Environment Setup

2.1 Google Collab

The research involves the modelling of four specific neural networks to examine their model performance based on unbalanced and balanced dataset for identification of rotten and fresh fruits. The Python programming language has been utilized to achieve the model construction. To run the code, we utilized the Google Collab integrated development environment (IDE). We have uploaded the finalized code in the form of. ipynb notebooks. To import these Python notebooks into Google Collab you can follow these steps.

1. Go to the URL of the Google Collab and sign in with the Google account of your choice.
2. Go to File-> Open notebook and click on Upload as shown in Figure 1 and upload the. ipynb file of choice.

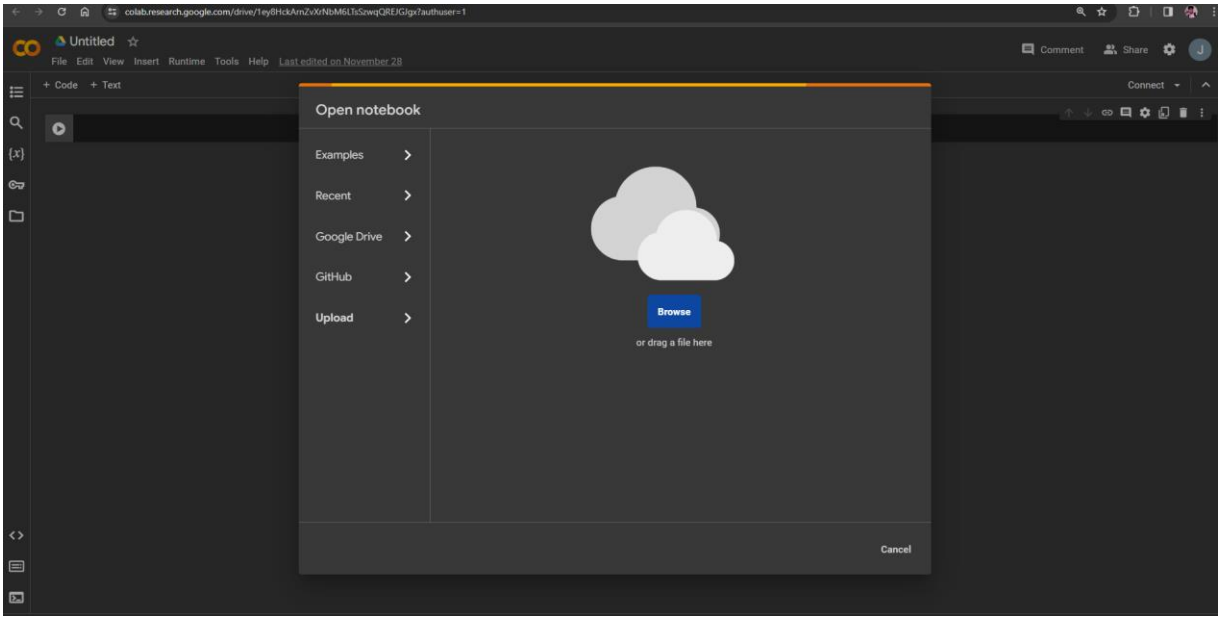

Figure 1: Google Collab window uploading .pynb files from local system

3. The. ipynb file of selection will be opened in Google Collab. Next click on the Connect button in the top right corner of the notebook. This will connect to a runtime in the cloud.
4. Please note that Runtime-shape will possess the value High-RAM and GPU mode only after upgrading to Collab Pro.
5. So, we have upgraded our Google Colab account to Colab Pro and used Hardware accelerator as V100 GPU as shown in the below figure 2.

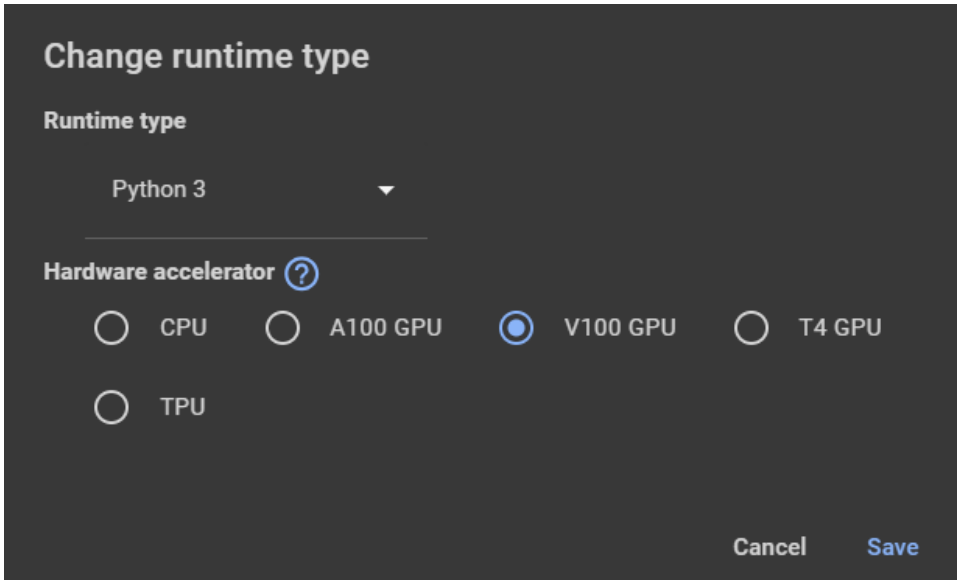

Figure 2: Runtime Selection in Jupyter Notebook using Colab Pro

6. Code Execution of each cell in the Notebook will be executed using V100 GPU by clicking on the Runcell button just to the left of each cell.

2.2 Google Drive

To utilize Google Collab, a cloud-based platform, for executing programs it is necessary to store the dataset in a cloud storage environment that is our Google Drive. We need to follow these steps to save the dataset in our google drive account.

1. Go to the URL[2] of the Google Drive Homepage.
2. Sign in with the same Google account through which Google Collab was accessed.
3. Once logged in, click on MyDrive and select Colab notebooks.
4. Upload the Dataset folders. Our original archived dataset needs to be uploaded. To access these folders that are now part of your drive, the code block has been shown in Figure 3 is to be executed in Google Collaboratory. This process is called the drive

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

mount, and it is mandatory for data access from the drive.

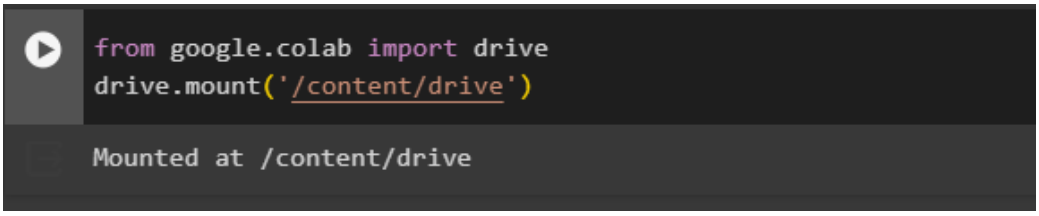

Figure 3: Mounting the google drive for accessing the Dataset.

3 Data Preparation

The dataset to be used for the modelling process is to be downloaded from the URL[3] through the Download ZIP option in Kaggle. The dataset will contain two folders namely Train and Test. The data is moved into google drive and for oversampled images we create a copy of the original data folder and rename it Oversampled.

4 Code Execution Steps

The codebase consists of three distinct folders for actual data, the under sampled data and for the over sampled data. pertaining to rotten and fresh fruits. Eachof these folders will contain image files used to build the 4 neural network algorithms for each type. All the implementation code is on one file and that contains the following steps, Exploratory Data Analysis, Image Augmentation, Class Balancing, Image Augmentation of balanced images, details about models that were created and the results.

4.1 Importing Dependent Libraries

The neural networks that have been modelled in the research have used in-built libraries such as Keras to achieve functionalities like base model importing. The code to import dependent libraries is shown in Figure 4.

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

```
[ ]  import glob, random, re
     import os, sys
     import pandas as pd
     import shutil
     import cv2
     from sklearn.preprocessing import MinMaxScaler
     import matplotlib.pyplot as plt
     import seaborn as sb
     import numpy as np
     #Sharpening of images
     from skimage.io import imshow, imread
     from scipy.signal import convolve2d
     import warnings
     warnings.filterwarnings("ignore")
     from tensorflow.keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
     from tensorflow.keras.models import Sequential, Model
     from tensorflow.keras.layers import Conv2D, GlobalAveragePooling2D, Dense, Flatten, Layer, Input, Dropout
     from keras.applications.inception_v3 import InceptionV3
     from keras.applications.resnet_v2 import ResNet50V2
     from keras.applications.efficientnet_v2 import EfficientNetV2B0
```

Figure 4: Importing of necessary libraries

Figure 5 represents the block of code to set the path for data folder and read the count of images in each category.

```
[ ]  fresh_apples = '/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshapples/'
     rotten_apples = '/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenapples/'
     fresh_banana = '/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshbanana/'
     rotten_banana = '/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenbanana/'
     fresh_oranges = '/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshoranges/'
     rotten_oranges = '/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenoranges/'

[ ]  categories = os.listdir('/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/')
     print(categories)

     ['freshbanana', 'rottenbanana', 'freshapples', 'freshoranges', 'rottenapples', 'rottenoranges']

[ ]  count_fresh_apples = len(os.listdir(fresh_apples))
     count_rotten_apples = len(os.listdir(rotten_apples))
     count_fresh_banana = len(os.listdir(fresh_banana))
     count_rotten_banana = len(os.listdir(rotten_banana))
     count_fresh_oranges = len(os.listdir(fresh_oranges))
     count_rotten_oranges = len(os.listdir(rotten_oranges))
     count_fresh_apples, count_rotten_apples, count_fresh_banana, count_rotten_banana, count_fresh_oranges, count_rotten_oranges

     (1693, 2342, 1581, 2224, 1466, 1595)
```

Figure 5: Data Path and Count

As seen in Figure 6, illustrate the code to generate bar plot of the total number of images in each class.

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

```
#plotting graph for leaf count
fig = plt.figure(figsize = (10, 5))
values=[count_fresh_apples, count_rotten_apples, count_fresh_banana, count_rotten_banana, count_fresh_oranges, count_rotten_oranges]
#creating the bar plot
plt.bar(categories, values, color ='blue', width = 0.4)

plt.xlabel("Classes")
plt.ylabel("Number of Images")
plt.show()
```

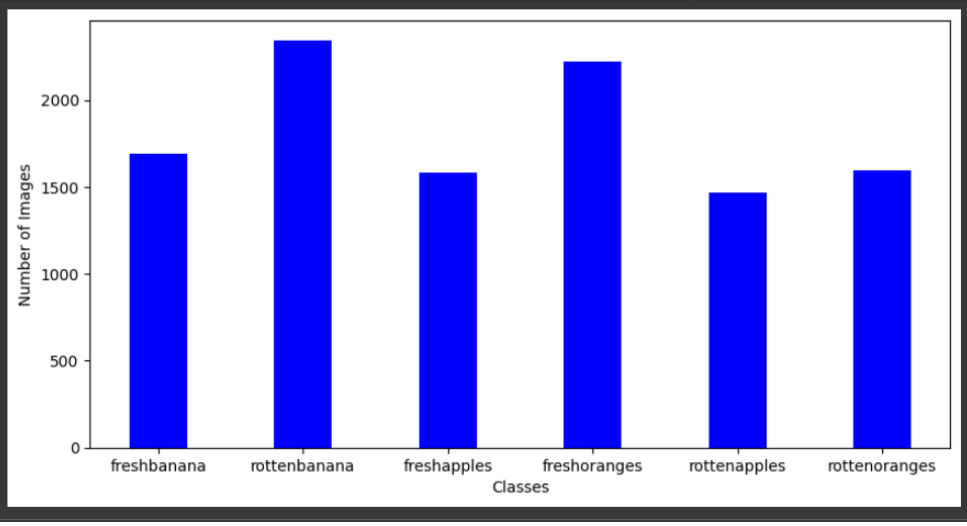

Figure 6: Data Classes and count

In figure 7, the code to generate list of images with file extension as .png file.

```
freshapples = glob.glob(fresh_apples + "*.png")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d  images.\nFirst 5 filenames:" % len(freshapples))
print ('\n'.join(freshapples[:5]))
```

```
Total of 1693  images.
First 5 filenames:
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshapples/rotated_by_45_Screen Shot 2018-06-08 at 5.15.39 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshapples/rotated_by_45_Screen Shot 2018-06-08 at 5.18.16 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshapples/rotated_by_45_Screen Shot 2018-06-08 at 5.15.14 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshapples/rotated_by_45_Screen Shot 2018-06-08 at 5.19.15 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshapples/rotated_by_45_Screen Shot 2018-06-08 at 5.18.58 PM.png
```

```
rottenapples = glob.glob(rotten_apples + "*.png")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d  images.\nFirst 5 filenames:" % len(rottenapples))
print ('\n'.join(rottenapples[:5]))
```

```
Total of 2342  images.
First 5 filenames:
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenapples/rotated_by_75_Screen Shot 2018-06-07 at 2.19.15 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenapples/rotated_by_75_Screen Shot 2018-06-07 at 2.20.04 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenapples/rotated_by_75_Screen Shot 2018-06-07 at 2.44.36 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenapples/rotated_by_75_Screen Shot 2018-06-07 at 2.50.43 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenapples/rotated_by_75_Screen Shot 2018-06-07 at 2.56.57 PM.png
```

```
freshbanana = glob.glob(fresh_banana+"*.png")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d  images.\nFirst 5 filenames:" % len(freshbanana))
print ('\n'.join(freshbanana[:5]))
```

```
Total of 1581  images.
First 5 filenames:
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshbanana/rotated_by_45_Screen Shot 2018-06-12 at 9.38.51 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshbanana/rotated_by_45_Screen Shot 2018-06-12 at 9.40.56 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshbanana/rotated_by_45_Screen Shot 2018-06-12 at 9.39.33 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshbanana/rotated_by_45_Screen Shot 2018-06-12 at 9.43.27 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshbanana/rotated_by_45_Screen Shot 2018-06-12 at 9.40.10 PM.png
```

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

```
rottenbanana = glob.glob(rotten_banana + "*.png")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d  images.\nFirst 5 filenames:" % len(rottenbanana))
print ('\n'.join(rottenbanana[:5]))
```

```
Total of 2224  images.
First 5 filenames:
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenbanana/rotated_by_60_Screen Shot 2018-06-12 at 9.21.20 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenbanana/rotated_by_60_Screen Shot 2018-06-12 at 9.27.15 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenbanana/rotated_by_60_Screen Shot 2018-06-12 at 9.25.18 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenbanana/rotated_by_60_Screen Shot 2018-06-12 at 9.21.25 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenbanana/rotated_by_60_Screen Shot 2018-06-12 at 9.25.46 PM.png
```

```
freshoranges = glob.glob(fresh_oranges+"*.png")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d  images.\nFirst 5 filenames:" % len(freshoranges))
print ('\n'.join(freshoranges[:5]))
```

```
Total of 1466  images.
First 5 filenames:
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshoranges/rotated_by_30_Screen Shot 2018-06-13 at 12.15.39 AM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshoranges/rotated_by_30_Screen Shot 2018-06-13 at 12.16.08 AM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshoranges/rotated_by_30_Screen Shot 2018-06-13 at 12.11.57 AM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshoranges/rotated_by_30_Screen Shot 2018-06-13 at 12.18.23 AM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/freshoranges/rotated_by_30_Screen Shot 2018-06-13 at 12.17.31 AM.png
```

```
rottenoranges = glob.glob(rotten_oranges + "*.png")
# Print out the first 5 file names to verify we're in the right folder.
print ("Total of %d  images.\nFirst 5 filenames:" % len(rottenoranges))
print ('\n'.join(rottenoranges[:5]))
```

```
Total of 1595  images.
First 5 filenames:
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenoranges/rotated_by_45_Screen Shot 2018-06-12 at 11.22.47 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenoranges/rotated_by_45_Screen Shot 2018-06-12 at 11.20.52 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenoranges/rotated_by_45_Screen Shot 2018-06-12 at 11.25.07 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenoranges/rotated_by_45_Screen Shot 2018-06-12 at 11.23.09 PM.png
/content/drive/MyDrive/Colab Notebooks/archive/dataset/train/rottenoranges/rotated_by_45_Screen Shot 2018-06-12 at 11.29.44 PM.png
```

Figure 7: Filenames with .png format

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

5 Image Augmentation

Figure 8 illustrates the read and convert the image and show image shape and plot image in RGB format.

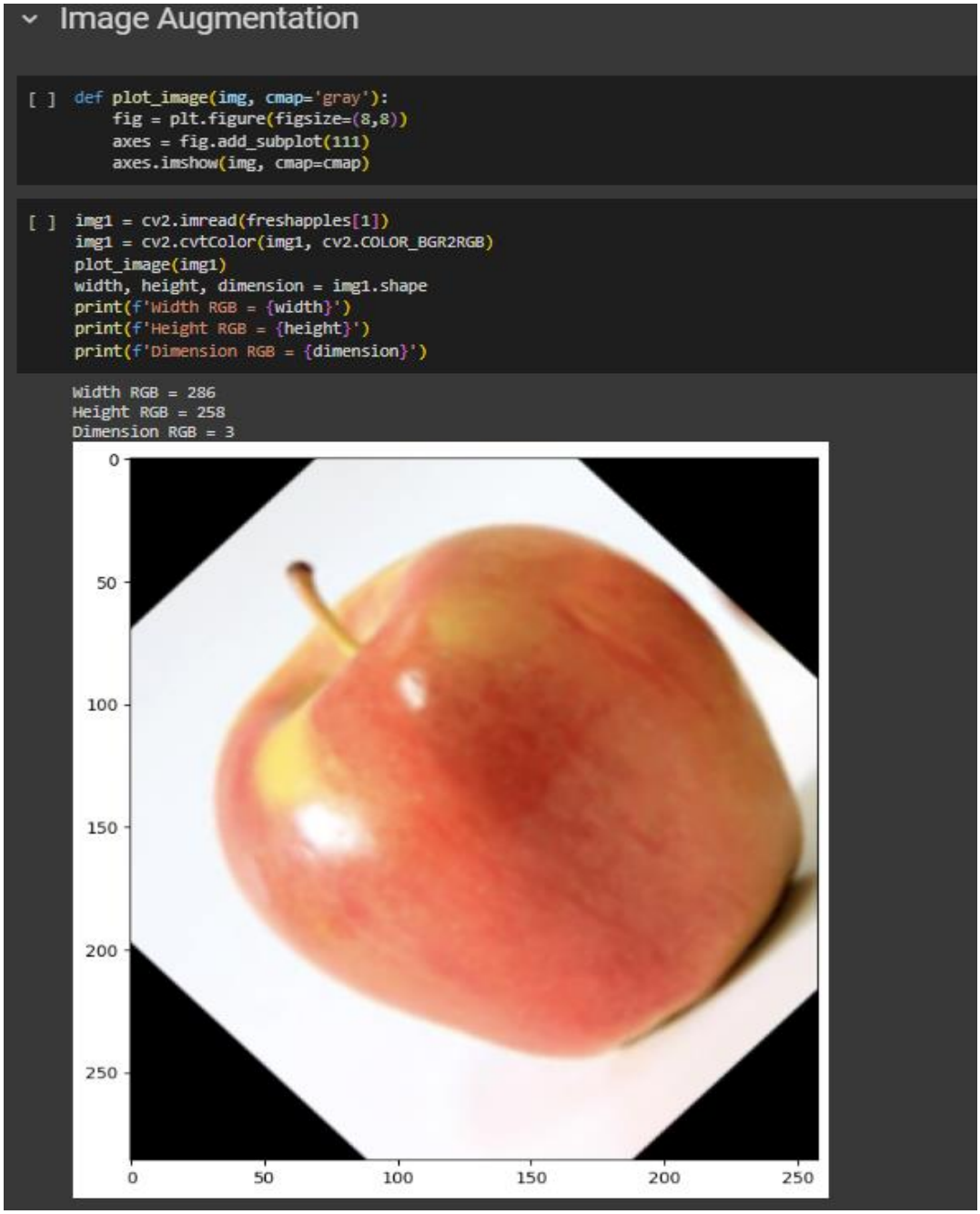

Figure 8: Converted Image in RGB format

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

Figure 9 illustrates the read the image in grayscale.

```
img1_gray = cv2.cvtColor(img1, cv2.COLOR_RGB2GRAY)
plot_image(img1_gray)
width, height = img1_gray.shape
print(f'Width Grayscale = {width}')
print(f'Height Grayscale = {height}')
print(f'Image Shape Grayscale {img1_gray.shape}')
```

```
Width Grayscale = 286
Height Grayscale = 258
Image Shape Grayscale (286, 258)
```

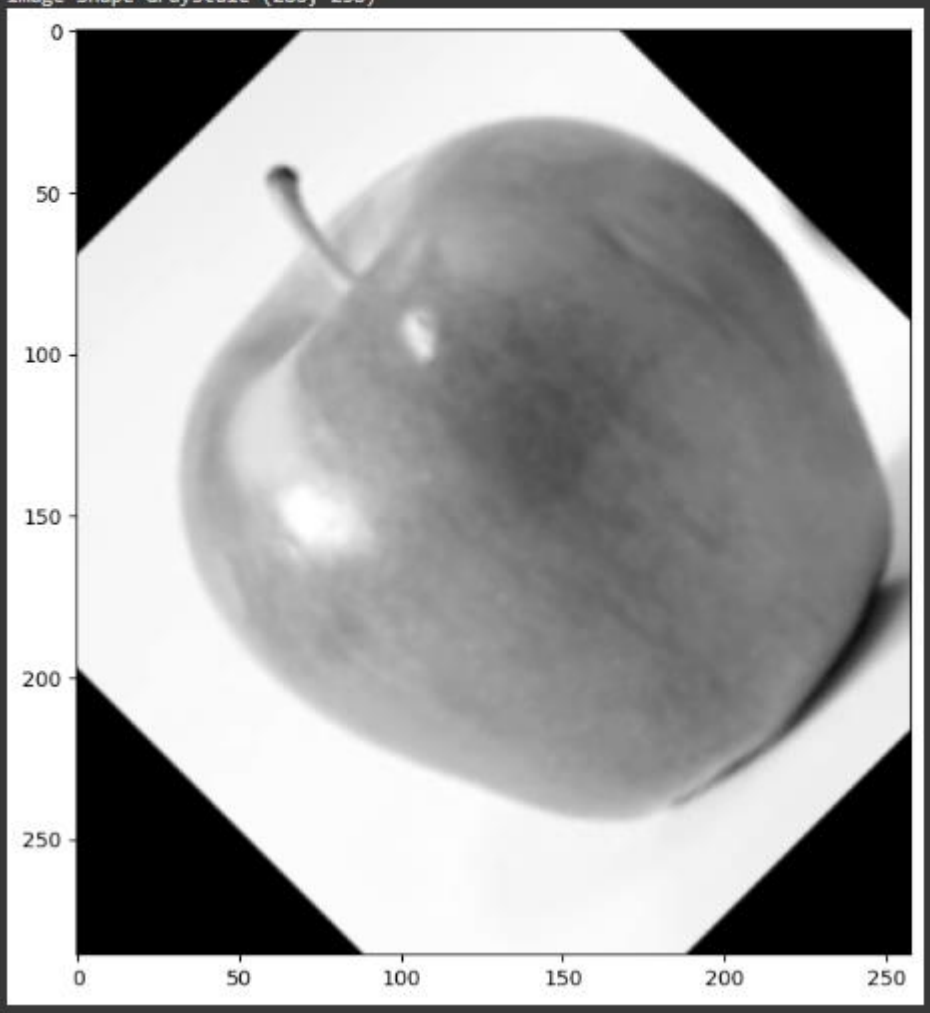

Figure 9: Read GrayScale Image

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

Figure 10 and 11, illustrate the code to generate adaptive thresholds of the images and display the contours.

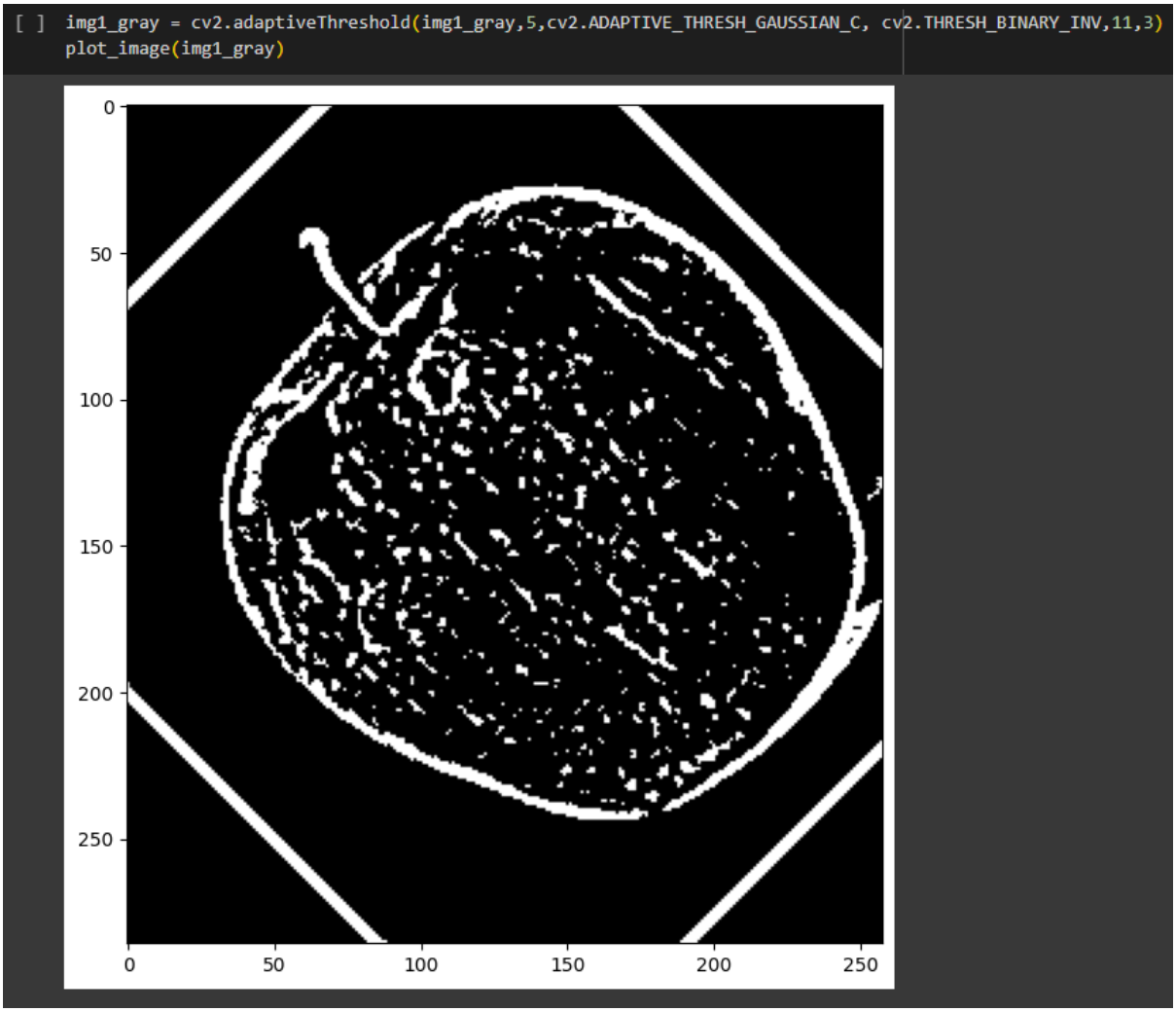

Figure 10: Adaptive threshold

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

```
contours = cv2.findContours(img1_gray, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours = contours[0] if len(contours) == 2 else contours[1]
contours = sorted(contours, key=cv2.contourArea, reverse=True)
for c in contours:
    x,y,w,h = cv2.boundingRect(c)
    img1_ROI = img1[y:y+h, x:x+w]
    break
plot_image(img1_ROI)
width, height, dimension = img1_ROI.shape
print(f'Width = {width}')
print(f'Height = {height}')
print(f'Dimension = {dimension}')
```

```
Width = 214
Height = 154
Dimension = 3
```

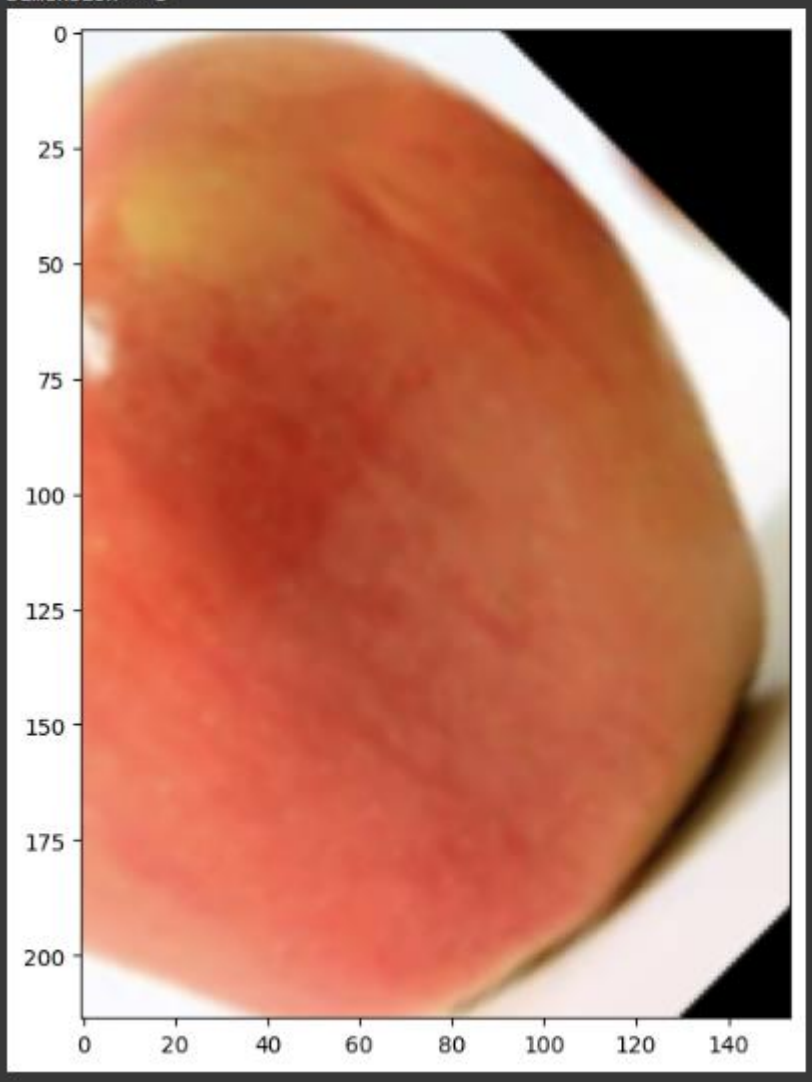

Figure 11: Adaptive threshold and contouring

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

6 Class Balancing

6.1 UnderSampling

Figure 12 illustrates the code to create train and test folder for under sampled images.

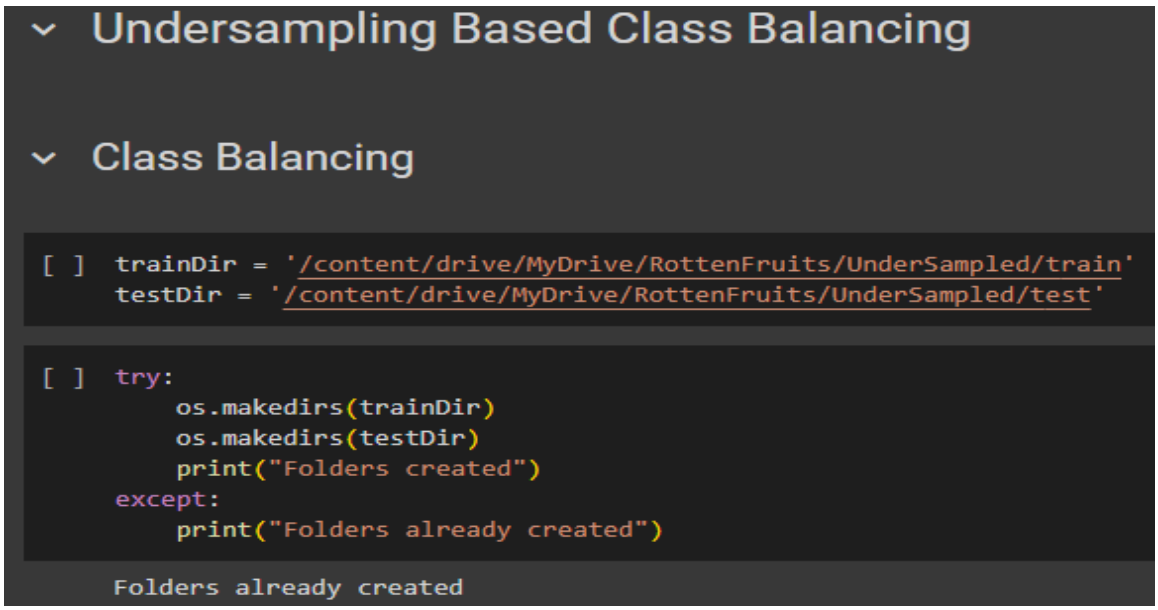

Figure 12: code to create train and test folder for under sampled images.

Figure 13, illustrate the code to get the minimum number of class count and generate data function for under sampled data.

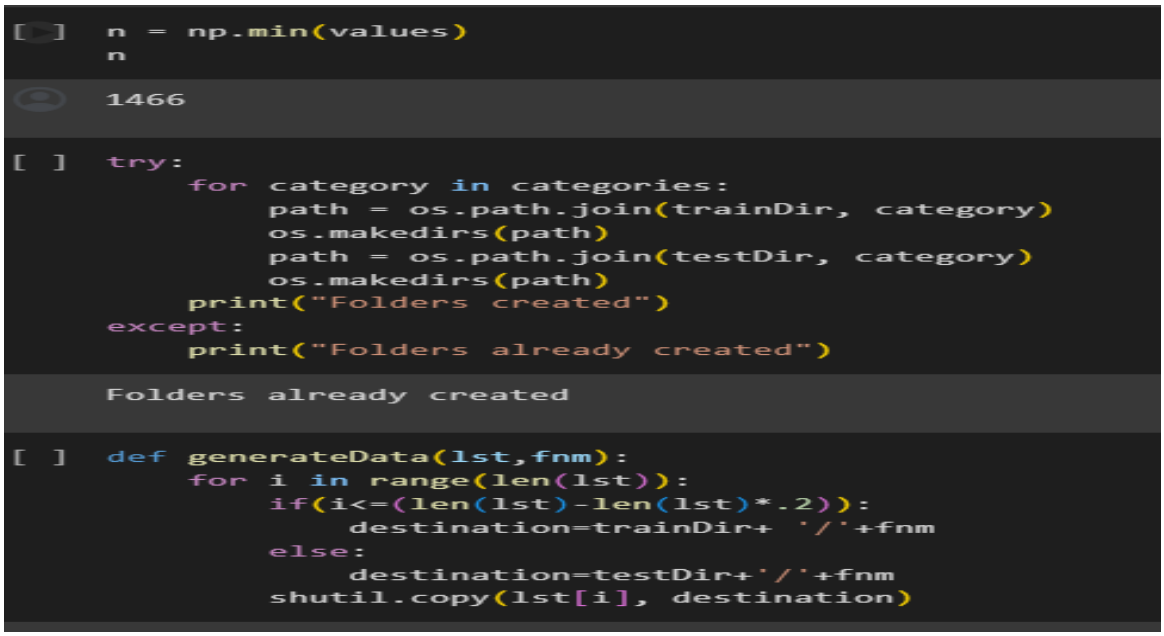

Figure 13: Count for min to undersampled

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

The Figure 14 and 15, illustrate the code to copy images into their respective category folder.

```
[ ]  try:
         generateData(freshapples[0:n],"freshapples")
         print("Images set in training and testing folders")
     except:
         print("Images already set in training and testing folders")

     Images set in training and testing folders

[ ]  try:
         generateData(rottenapples[0:n],"rottenapples")
         print("Images set in training and testing folders")
     except:
         print("Images already set in training and testing folders")

     Images set in training and testing folders

[ ]  try:
         generateData(freshbanana[0:n],"freshbanana")
         print("Images set in training and testing folders")
     except:
         print("Images already set in training and testing folders")

     Images set in training and testing folders

[ ]  try:
         generateData(rottenbanana[0:n],"rottenbanana")
         print("Images set in training and testing folders")
     except:
         print("Images already set in training and testing folders")

     Images set in training and testing folders
```

Figure 14: undersampling images for classes

```
     try:
         generateData(freshoranges[0:n],"freshoranges")
         print("Images set in training and testing folders")
     except:
         print("Images already set in training and testing folders")

     Images set in training and testing folders

[ ]  try:
         generateData(rottenoranges[0:n],"rottenoranges")
         print("Images set in training and testing folders")
     except:
         print("Images already set in training and testing folders")

     Images set in training and testing folders
```

Figure 15: Images in each class

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

6.2 Oversampling

Figure 16 illustrates the code set train and test folder for Oversampled data. Also, the code to extract the maximum number of class count.

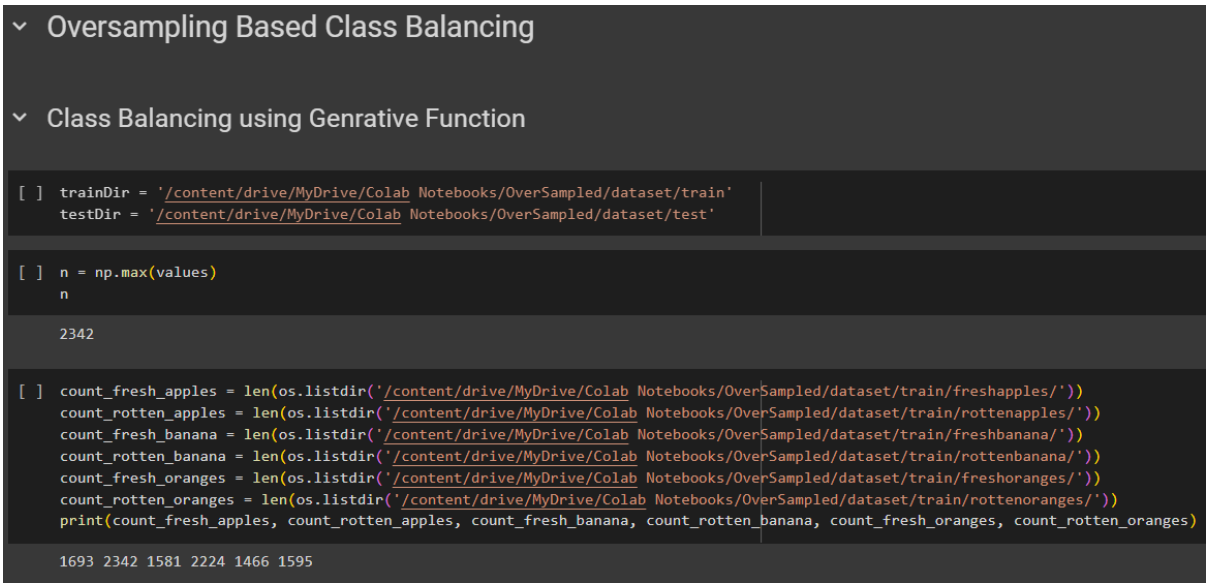

Figure 16: Count for maximum to oversample

Figure 17 illustrates the code to generate ImageDataGenerator to create images using a same class image till the count reaches the maximum number.

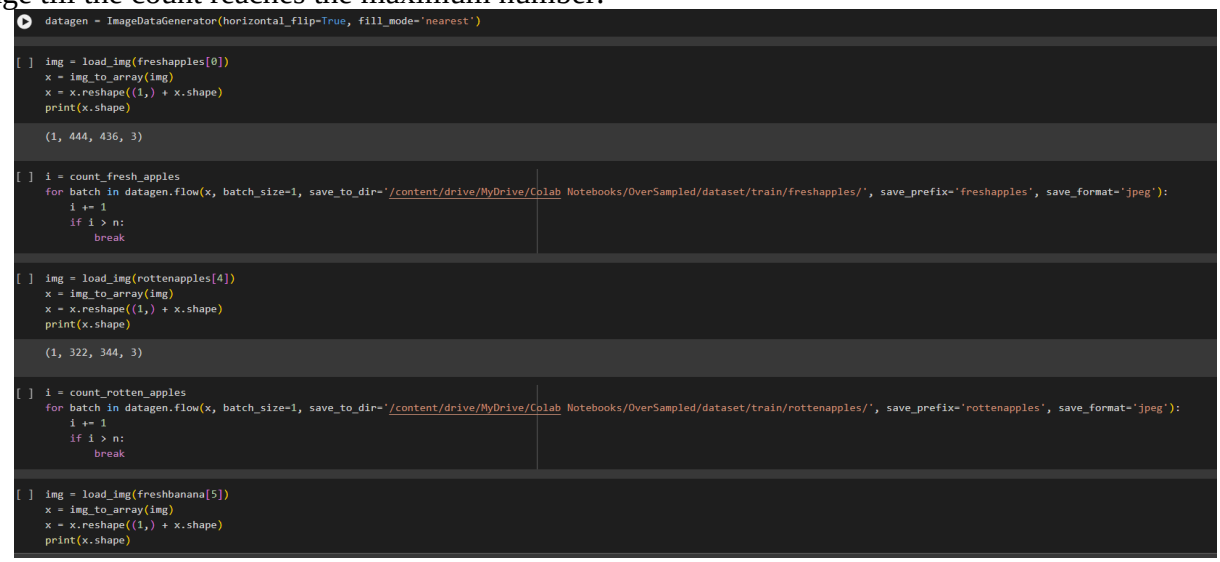

Figure 17: Image Data generator for fresh and rotten images

The Figure 17 and 18, illustrate the code to get the maximum number of classes count and set path for new oversampled folder for all the classes.

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

```
i = count_rotten_banana
for batch in datagen.flow(x, batch_size=1, save_to_dir='/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/rottenbanana/', save_prefix='rottenbanana', save_format='jpeg'):
    i += 1
    if i > n:
        break
```

```
[ ] img = load_img(freshoranges[5])
    x = img_to_array(img)
    x = x.reshape((1,) + x.shape)
    print(x.shape)

    (1, 290, 312, 3)
```

```
[ ] i = count_fresh_oranges
    for batch in datagen.flow(x, batch_size=1, save_to_dir='/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/freshoranges/', save_prefix='freshoranges', save_format='jpeg'):
        i += 1
        if i > n:
            break
```

```
[ ] img = load_img(rottenoranges[4])
    x = img_to_array(img)
    x = x.reshape((1,) + x.shape)
    print(x.shape)

    (1, 230, 264, 3)
```

```
[ ] i = count_rotten_oranges
    for batch in datagen.flow(x, batch_size=1, save_to_dir='/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/rottenoranges/', save_prefix='rottenoranges', save_format='jpeg'):
        i += 1
        if i > n:
            break
```

Figure 18: Image Data generator for fresh and rotten images

Figure 19 illustrates the code to get the class count and generate graph for it.

```
count_fresh_apples = len(os.listdir('/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/freshapples/'))
count_rotten_apples = len(os.listdir('/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/rottenapples/'))
count_fresh_banana = len(os.listdir('/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/freshbanana/'))
count_rotten_banana = len(os.listdir('/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/rottenbanana/'))
count_fresh_oranges = len(os.listdir('/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/freshoranges/'))
count_rotten_oranges = len(os.listdir('/content/drive/MyDrive/Colab Notebooks/OverSampled/dataset/train/rottenoranges/'))
print(count_fresh_apples, count_rotten_apples, count_fresh_banana, count_rotten_banana, count_fresh_oranges, count_rotten_oranges)
#plotting graph for leaf count
fig = plt.figure(figsize = (10, 5))
values=[count_fresh_apples, count_rotten_apples, count_fresh_banana, count_rotten_banana, count_fresh_oranges, count_rotten_oranges]
#creating the bar plot
plt.bar(categories, values, color ='blue', width = 0.4)

plt.xlabel("Classes")
plt.ylabel("Number of Images")
plt.show()
#Visualizing after oversampling
```

```
2331 2343 2315 2342 2306 2320
```

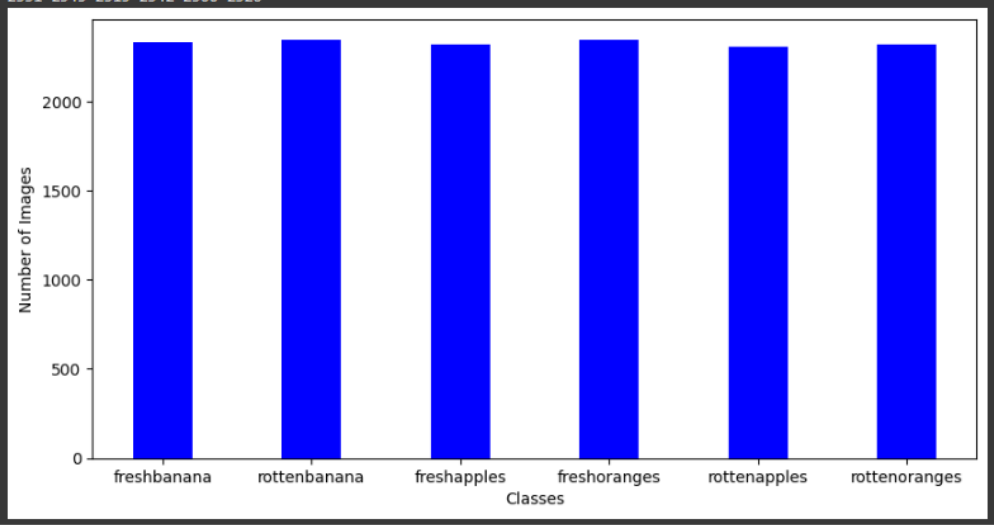

Figure 19: Oversampled images

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

7 Image Augmentation

Figure 20 illustrates the code to use ImageDataGenerator to generate augmented images for the deep learning models for actual data.

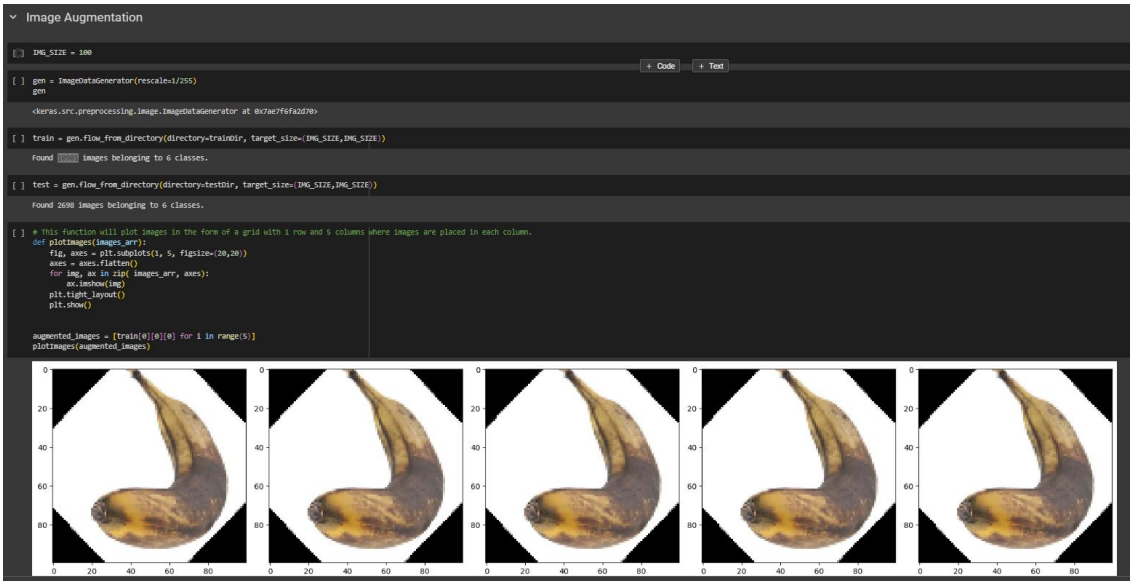

Figure 20: Image Data Generator Actual data

Figures 20 show the code to use ImageDataGenerator to generate augmented images for the deep learning models for under sampled data.

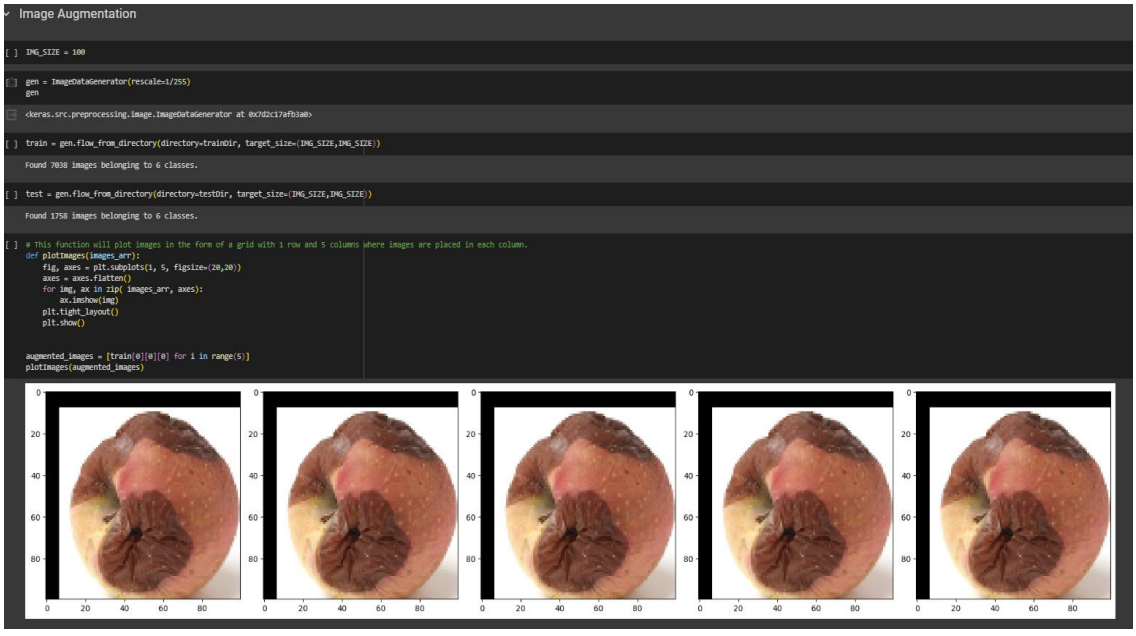

Figure 21: Image Data Generator undersampled data

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

Figures 22 show the code to use ImageDataGenerator to generate augmented images for the deep learning models for oversampled data.

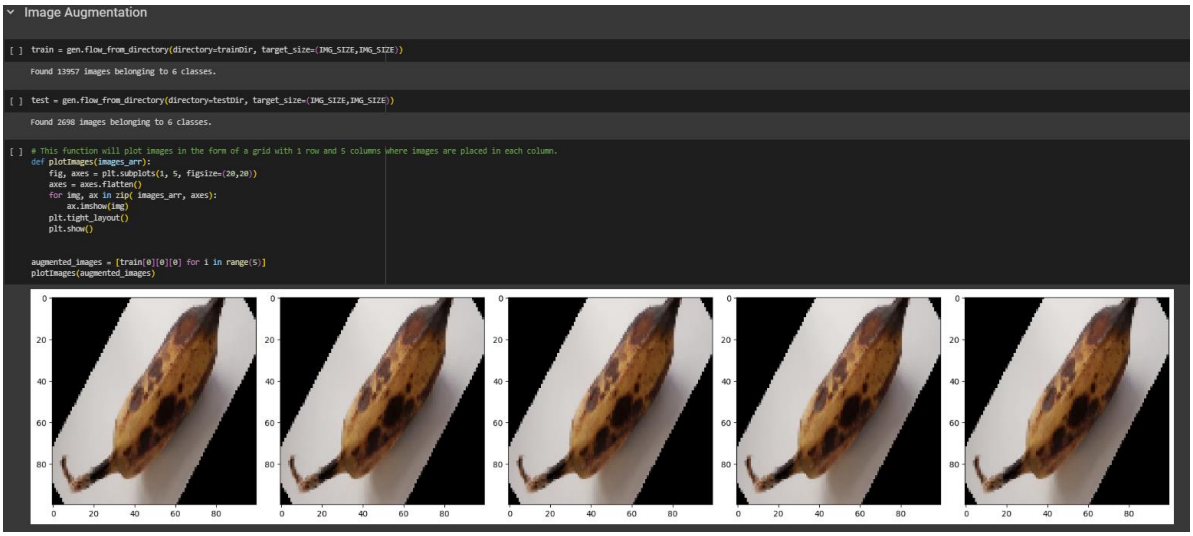

Figure 21: Image Data Generator oversampled data

8 Importing Model and Model Specification

Used keras[3] libraries to import neural network models. For instance, in Figure 22 we can see an example of how to import the base ResNet model. We then make modifications to the base model by adding layers, which allows us to classify fruits as depicted in Figure 23.

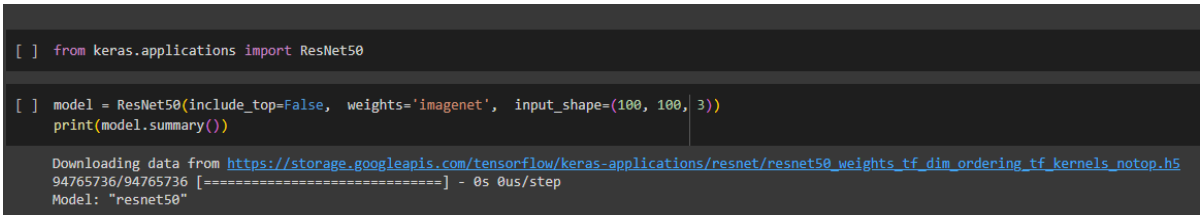

Figure 22: Importing Model from Keras API

```
resnet = model.output
resnet = Flatten()(resnet)
resnet = Dense(64, activation='relu')(resnet)
resnet = Dense(32, activation='relu')(resnet)
resnet = Dropout(0.2)(resnet)
output_layer = Dense(6, activation='sigmoid')(resnet)
model = Model(inputs=model.input, outputs=output_layer)
model.compile(optimizer = 'adam', loss= 'categorical_crossentropy',   metrics = ['accuracy'])
```

Figure 23: Adding Extra layer to our Base Model

8.1 Model compilation and results

To compile the model, use the code provided in Figure 24. The model compilation while the fit function generates the desired results.

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

```
model.compile(optimizer = 'adam', loss= 'categorical_crossentropy',   metrics = ['accuracy'])

history = model.fit(train, validation_data=test, epochs=10)

Epoch 1/10
437/437 [==============================] - 94s 156ms/step - loss: 1.1832 - accuracy: 0.5460 - val_loss: 2.8269 - val_accuracy: 0.1505
Epoch 2/10
437/437 [==============================] - 67s 154ms/step - loss: 0.5291 - accuracy: 0.8293 - val_loss: 0.9393 - val_accuracy: 0.6575
Epoch 3/10
437/437 [==============================] - 68s 155ms/step - loss: 0.3702 - accuracy: 0.8941 - val_loss: 3.0847 - val_accuracy: 0.4770
Epoch 4/10
437/437 [==============================] - 67s 154ms/step - loss: 0.2891 - accuracy: 0.9157 - val_loss: 0.6364 - val_accuracy: 0.8147
Epoch 5/10
437/437 [==============================] - 67s 154ms/step - loss: 0.2425 - accuracy: 0.9276 - val_loss: 0.2331 - val_accuracy: 0.9240
Epoch 6/10
437/437 [==============================] - 67s 154ms/step - loss: 0.2184 - accuracy: 0.9367 - val_loss: 0.6121 - val_accuracy: 0.8299
Epoch 7/10
437/437 [==============================] - 67s 154ms/step - loss: 0.2660 - accuracy: 0.9266 - val_loss: 2.4477 - val_accuracy: 0.1501
Epoch 8/10
437/437 [==============================] - 68s 156ms/step - loss: 0.2779 - accuracy: 0.9208 - val_loss: 0.2693 - val_accuracy: 0.9148
Epoch 9/10
437/437 [==============================] - 67s 154ms/step - loss: 0.1715 - accuracy: 0.9498 - val_loss: 0.2553 - val_accuracy: 0.9274
Epoch 10/10
437/437 [==============================] - 67s 154ms/step - loss: 0.1623 - accuracy: 0.9546 - val_loss: 0.1879 - val_accuracy: 0.9366
```

Figure 24: Model Compilation and results

9 Model Evaluation

We can plot the accuracy and Val accuracy in figure 25 and loss, and Val in figure 26 to effectively evaluate the classification performance of each learning network.

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

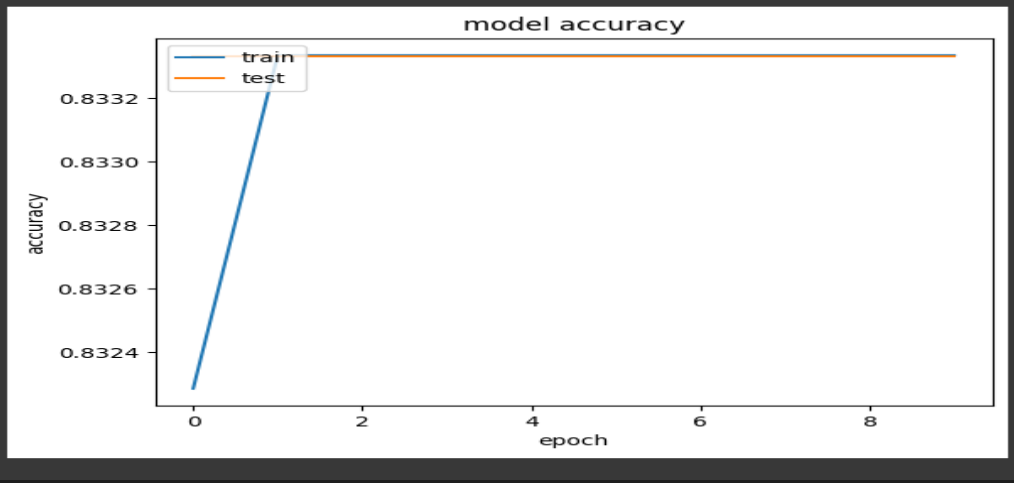

Figure 25: Accuracy and Val_Accuracy

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/

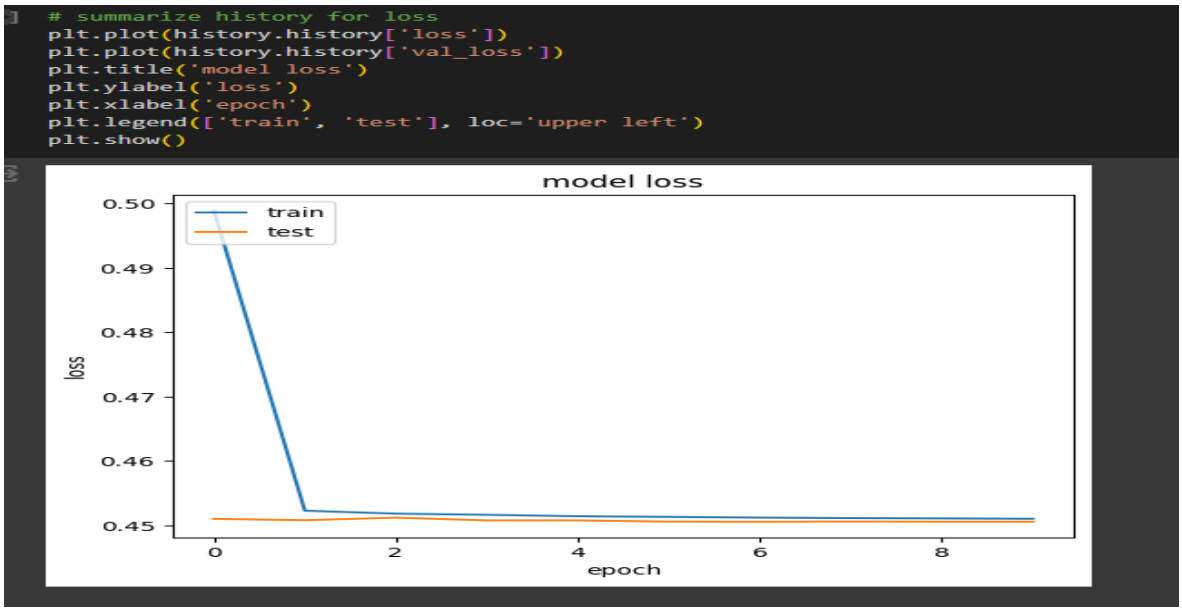

Figure 25: Loss and Val_Loss

10 Model Results

As shown in figure 26. It is the overall Model performance for the oversampled dataset.

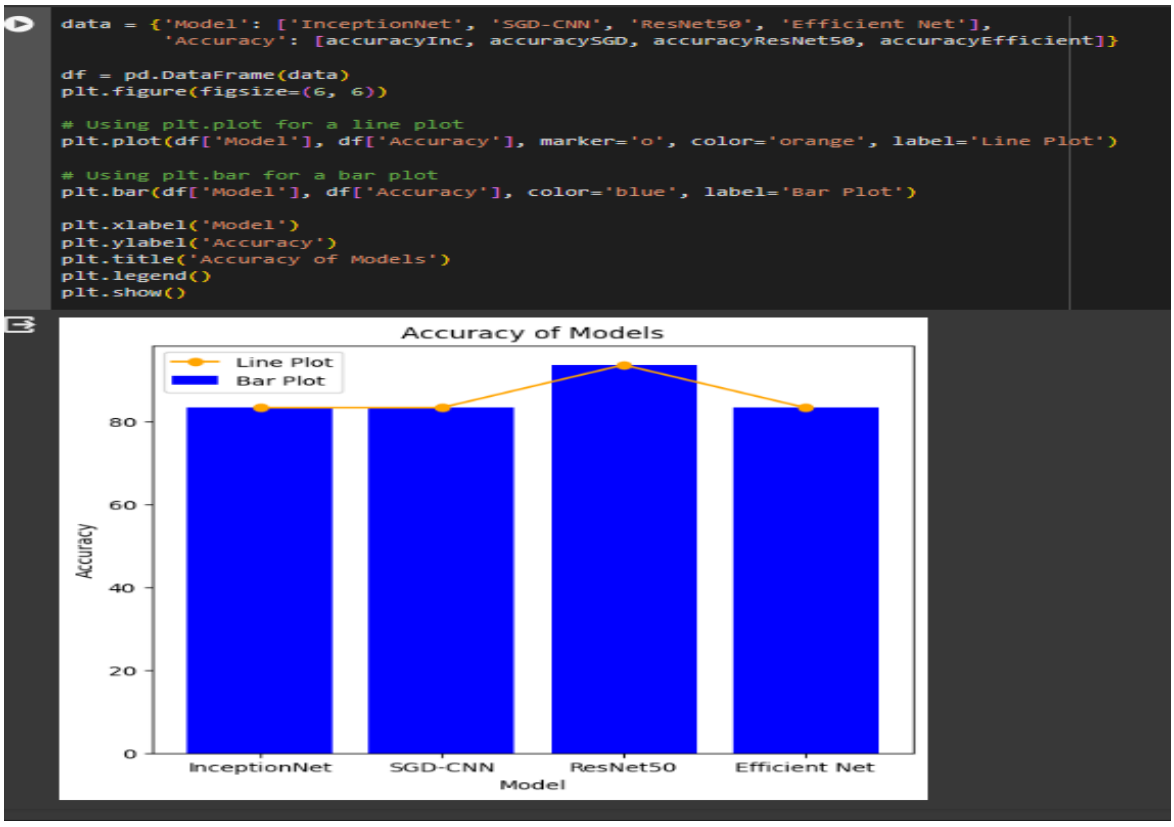

Figure 26: Overall Model performance bar plot

[2] https://www.google.com/drive/

[3] https://www.kaggle.com/datasets/sriramr/fruits-fresh-and-rotten-for-classification

[4] https://keras.io/api/applications/