Prediction and Classification of Healthiness of Fruits Based on the Size

TEAM MEMBERS:

PRAGADEESWARAN K – RA2211047010135

PRASHETH P -RA2211047010138

CHAKKARA SAI KOWSHIK REDDY – RA2211047010140

SAPARE SRI DATTA KAMAL – RA2211047010143

LOGAPRIYAN R – RA2211047010144

ABSTRACT:

The increasing demand for high-quality fruits in both local and global markets necessitates effective methods for assessing fruit quality. Traditional methods of evaluating fruit healthiness are often labor-intensive, subjective, and can lead to inconsistencies in quality control. To address these challenges, this project focuses on the prediction and classification of the healthiness of various Indian fruits based on their size and visual characteristics, utilizing advanced deep learning techniques.

Leveraging the **FruitNet dataset**, which comprises over 14,700 high-quality images of six popular Indian fruits—apple, banana, guava, lime, orange, and pomegranate—this study employs a convolutional neural network (CNN) architecture, specifically the **InceptionResNetV2** model. The dataset is organized into three categories: good quality, bad quality, and mixed quality, providing a comprehensive framework for the training and evaluation of the model.

The project follows a systematic approach to data preparation, model selection, training, and evaluation. Initially, the dataset is divided into training, validation, and test sets, with preprocessing steps including resizing, normalization, and data augmentation to enhance model performance and mitigate overfitting. The transfer learning technique allows the model to leverage pre-trained weights from the InceptionResNetV2 architecture, which has demonstrated remarkable capabilities in image classification tasks.

The model's performance is evaluated using key metrics, including accuracy, confusion matrix, and classification reports. Results indicate that the model achieves high accuracy in classifying fruit healthiness based on size, with the ability to generalize well to unseen data. The findings of this research not only demonstrate the potential of machine learning in agricultural applications but also provide valuable insights into the relationship between fruit size and quality.

The implications of this study extend to stakeholders in the agricultural sector, including farmers, suppliers, and retailers, who can benefit from automated quality assessment tools to improve marketability and consumer satisfaction. By harnessing the power of deep learning and computer vision, this project contributes to the ongoing evolution of precision agriculture, offering innovative solutions to traditional challenges in fruit quality evaluation.

OBJECTIVE:

The primary aim of this project is to develop an effective model for predicting and classifying the healthiness of various Indian fruits based on their size and visual characteristics. The specific objectives are outlined as follows:

1. **To Develop a Predictive Model**:
    - Utilize advanced deep learning techniques, particularly convolutional neural networks (CNNs), to create a predictive model capable of assessing the quality of fruits. By training on high-quality image data, the model will learn to differentiate between various healthiness levels, such as good, bad, and mixed quality.
2. **To Analyze Fruit Size and Quality Relationship**:
    - Investigate the correlation between fruit size and healthiness, aiming to determine how size impacts quality classification. This objective involves examining size distribution across different categories of fruit healthiness to provide insights into agricultural practices and standards.
3. **To Enhance Data Preparation Techniques**:
    - Implement comprehensive data preprocessing and augmentation strategies to improve the robustness of the model. This includes normalizing image data, resizing images to a consistent format, and using data augmentation methods such as random rotations and flips to create a more diverse training dataset.
4. **To Evaluate Model Performance**:
    - Assess the model's performance using various metrics, including accuracy, precision, recall, and F1-score. This evaluation will help quantify the effectiveness of the model in predicting fruit healthiness and provide a clearer understanding of its strengths and weaknesses.
5. **To Utilize Transfer Learning**:

- Leverage transfer learning techniques, specifically using the InceptionResNetV2 model pre-trained on large image datasets. This will allow the project to capitalize on existing knowledge, reducing training time and improving model accuracy.

6. **To Contribute to Precision Agriculture**:
   - Provide a technological solution for farmers and suppliers to automate fruit quality assessment. By offering a reliable and efficient method for evaluating fruit healthiness, this project aims to enhance quality control processes in the agricultural sector, thereby improving marketability and reducing waste.
7. **To Facilitate Stakeholder Decision-Making**:
   - Generate actionable insights and recommendations for stakeholders, including farmers, distributors, and retailers, to make informed decisions about fruit cultivation, harvesting, and marketing strategies based on quality predictions.
8. **To Support Future Research**:
   - Establish a foundation for further studies in fruit classification and health assessment, encouraging the exploration of additional features (beyond size) that may influence fruit quality. This could include aspects like color, texture, and environmental factors.

LITERATURE SURVEY:

Study Title	Authors	Methodology	Fruits Assessed	Accuracy (%)
A General Machine Learning Model for Assessing Fruit Quality Using Deep Image Features	Rania Barakat et al.	Vision Transformers (ViT)	Apples, Cucumbers, Grapes, Oranges, etc.	99.50 (Apples), 100 (Grapes), 97 (Guavas)
Fruit Quality Assessment with Densely Connected Convolutional Neural Network	Mohsen Fallahzadeh et al.	DenseNet201 with Data Augmentation	Various Fruits	99.67 (fine-grain quality assessment)
Deep Learning for Fruit Quality Assessment	S. A. Al-Mahdy et al.	CNN and Transfer Learning	Apples, Bananas, etc.	98.62 (MobileNetV2), 99.26 (DenseNet201)

Fruit Classification Using CNN and Vision Transformers	Y. Liu et al.	CNN, Vision Transformers, Data Augmentation	Multiple Fruits	99.0 - 99.5 (specific fruits vary)

Problem Statement:

The assessment of fruit quality is a critical aspect of the agricultural and food supply chain, impacting both consumer health and market dynamics. Traditional methods for evaluating fruit quality often rely on human judgment, which can be subjective, inconsistent, and time-consuming. This reliance on manual inspection can lead to misclassification and quality degradation, resulting in economic losses and reduced consumer trust.

With the global increase in food demand, exacerbated by population growth and urbanization, there is a pressing need for efficient and accurate methods for fruit quality assessment. The existing challenges include variability in quality due to factors like handling, storage conditions, and transportation. Moreover, manual inspection can become impractical when dealing with large quantities of produce.

Recent advancements in deep learning and computer vision have opened new avenues for automating the quality assessment process. However, the implementation of these technologies in fruit quality evaluation remains limited due to various obstacles. These include the need for high-quality labelled datasets, the complexity of developing robust models that can generalize well across different fruit types and environmental conditions, and the integration of these systems into existing supply chain operations.

The objective of this project is to develop a comprehensive deep learning model capable of accurately predicting and classifying the healthiness of fruits based on their size and visual features. By leveraging image processing techniques and deep neural networks, this study aims to create a system that enhances the accuracy and efficiency of fruit quality assessments, ultimately supporting better decision-making in the agricultural sector.

This project seeks to address the following questions:

- How can deep learning algorithms be effectively trained to differentiate between varying quality levels of fruits based on size and appearance?
- What are the limitations of current methodologies, and how can these be overcome to achieve high accuracy in diverse conditions?
- How can the findings of this research be applied in real-world agricultural and supply chain contexts to improve fruit quality management?

Existing Issues in Fruit Quality Assessment

The assessment of fruit quality faces several existing issues that hinder effective evaluation and management in agricultural practices. These issues can be broadly categorized into the following areas:

1. **Subjectivity in Manual Inspection**: Traditional fruit quality assessment relies heavily on human judgment, which can lead to inconsistencies and subjectivity in evaluations. Different inspectors may have varying standards for quality, resulting in misclassification and potential economic losses for producers and suppliers .

2. **Time-Consuming Processes**: Manual inspection of fruits is labor-intensive and time-consuming, especially when large volumes of produce are involved. This inefficiency can lead to delays in the supply chain, affecting the freshness and marketability of fruits .

3. **Lack of Standardized Protocols**: There is no universally accepted standard for fruit quality assessment, making it difficult to compare results across different studies or practices. This lack of standardization complicates the integration of automated systems into existing quality control protocols .

4. **Variability Due to External Factors**: Fruit quality is influenced by numerous factors such as climate, soil conditions, handling practices, and storage environments. This variability poses a challenge for developing robust machine learning models that can generalize well across different conditions .

5. **Data Limitations**: High-quality labeled datasets are essential for training effective deep learning models. However, the availability of such datasets in the agricultural domain is often limited. Many existing datasets lack diversity in terms of fruit types, quality levels, and environmental conditions.

6. **Integration Challenges**: Implementing automated fruit quality assessment systems in existing agricultural practices can be challenging. Farmers and suppliers may face barriers such as high costs, lack of technical expertise, and resistance to change from traditional methods(

7. **Overfitting and Generalization Issues**: Deep learning models, while powerful, can be prone to overfitting when trained on limited datasets. This overfitting can result in models that perform well on training data but fail to generalize to unseen fruit samples, leading to poor accuracy in real-world applications

8. **Consumer Perception and Acceptance**: Finally, the successful implementation of automated systems also depends on consumer acceptance. There may be

skepticism regarding the reliability of machine-generated quality assessments compared to human evaluations.

Module Explanation:

The project is structured into several key modules that work together to facilitate the prediction and classification of fruit healthiness based on size and visual features. Below is a detailed explanation of each module:

1. Data Acquisition and Preprocessing

- **Data Collection**: The project utilizes the FruitNet dataset, which contains over 14,700 high-quality images of six Indian fruits, categorized into three quality classes: good, bad, and mixed. This dataset provides a robust foundation for training machine learning models.
- **Image Processing**: Images are resized to a uniform dimension (e.g., 160x160 pixels) for consistency. This step is crucial as deep learning models require inputs of the same size. Data augmentation techniques, such as random rotation and flipping, are applied to increase the diversity of training images and reduce overfitting

2. Feature Extraction

- **Transfer Learning**: The project employs the InceptionResNetV2 model, a pre-trained convolutional neural network (CNN) that excels in image classification tasks. The model extracts features from fruit images, capturing complex patterns that distinguish different quality levels.
- **Feature Selection**: By utilizing a pre-trained model, the project leverages existing knowledge from a broad range of images, enhancing the model's ability to generalize from limited training data.

3. Model Development

- **Model Architecture**: A sequential model is created, incorporating layers such as convolutional layers, pooling layers, and dropout layers. The final output layer uses a softmax activation function to predict the probabilities of each quality class .
- **Hyperparameter Tuning**: The model's learning rate, batch size, and number of epochs are fine-tuned to optimize performance. This process is vital for achieving high accuracy and minimizing loss during training .

4. Training and Validation

- **Training the Model**: The model is trained using the training dataset, with validation occurring on a separate set to monitor performance. Early stopping is implemented to prevent overfitting, ensuring the model maintains good generalization capabilities .
- **Evaluation Metrics**: Performance metrics such as accuracy, precision, recall, and F1-score are computed to assess model performance on the test set. Confusion matrices visualize the model's classification effectiveness, providing insights into which classes are often confused .

5. Deployment

- **Model Saving**: After training, the model is saved for future use, enabling it to be deployed in practical applications. This deployment can support automated fruit quality assessment in agricultural settings .
- **User Interface (Optional)**: While not explicitly covered in the initial project, a user interface can be developed to allow users to upload fruit images and receive quality assessments in real time. This enhancement would significantly improve usability and accessibility .

6. Future Enhancements

- **Integration with IoT Devices**: Future iterations could explore integrating the model with Internet of Things (IoT) devices for real-time monitoring and assessment of fruit quality in supply chains .
- **Expanding the Dataset**: To improve the model’s accuracy, additional data could be collected from diverse environments and fruit types, addressing the current limitations in dataset diversity .

Implementation:

Importing Libraries: The necessary libraries for data manipulation, visualization, and deep learning are imported

Data Preparation: The code processes the dataset by reading images from specified directories and labeling them according to their quality:

Data Splitting: The dataset is split into training and testing sets:

Image Augmentation: Data augmentation techniques are applied to enhance the training dataset:

Creating Image Generators: Generators are created to flow images into the model during training:

Model Buildingedhe model architecture is defined using transfer learning from InceptionResNetV2:

Compiling and Training the Model: The model is compiled and trained using the training data:

Model Evaluation: The model is evaluated on the test set, and metrics are computed:

Fine-tuning the Model: Fine-tuning is performed to improve the model performance:

```python
import numpy as np
import pandas as pd
import os
import time
import matplotlib.pyplot as plt
import cv2
import seaborn as sns

sns.set_style("darkgrid")
import shutil
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.models import Model

# pprevent annoying tensorflow warning
import logging

logging.getLogger("tensorflow").setLevel(logging.ERROR)
import warnings

warnings.simplefilter("ignore")

import pathlib
sdir = r"../input/fruitnet-indian-fruits-dataset-with-quality/Processed
Images_Fruits"
bad_path = r"../input/fruitnet-indian-fruits-dataset-with-
quality/Processed Images_Fruits/Bad Quality_Fruits"
good_path = r"../input/fruitnet-indian-fruits-dataset-with-
quality/Processed Images_Fruits/Good Quality_Fruits"
mixed_path = r"../input/fruitnet-indian-fruits-dataset-with-
quality/Processed Images_Fruits/Mixed Qualit_Fruits"

BATCH_SIZE = 32
IMG_SIZE = (160, 160)
filepaths = []
labels = []
ht = 0
wt = 0
samples = 0
sample_count = 20
for quality in [bad_path, good_path, mixed_path]:
    fruit_list = os.listdir(quality)
    for fruit in fruit_list:
        fruit_path = os.path.join(quality, fruit)
        img_list = os.listdir(fruit_path)
        for i, img in enumerate(img_list):
            img_path = os.path.join(fruit_path, img)
            if i < sample_count:
                img = plt.imread(img_path)
                ht += img.shape[0]
                wt += img.shape[1]
                samples += 1
            filepaths.append(img_path)
            if quality == mixed_path:
                labels.append(fruit + "_mixed")
            else:
                labels.append(fruit)
Fseries = pd.Series(filepaths, name="filepaths")
Lseries = pd.Series(labels, name="labels")
df = pd.concat([Fseries, Lseries], axis=1)
df
```

```
                                                filepaths        labels
0      ../input/fruitnet-indian-fruits-dataset-with-q...     Guava_Bad
1      ../input/fruitnet-indian-fruits-dataset-with-q...     Guava_Bad
2      ../input/fruitnet-indian-fruits-dataset-with-q...     Guava_Bad
```

```
3      ../input/fruitnet-indian-fruits-dataset-with-q...     Guava_Bad
4      ../input/fruitnet-indian-fruits-dataset-with-q...     Guava_Bad
...                                                  ...           ...
19521  ../input/fruitnet-indian-fruits-dataset-with-q...  Banana_mixed
19522  ../input/fruitnet-indian-fruits-dataset-with-q...  Banana_mixed
19523  ../input/fruitnet-indian-fruits-dataset-with-q...  Banana_mixed
19524  ../input/fruitnet-indian-fruits-dataset-with-q...  Banana_mixed
19525  ../input/fruitnet-indian-fruits-dataset-with-q...  Banana_mixed

[19526 rows x 2 columns]
```

```
df['labels']=df['labels'].astype('category')
```

```
df['labels'].value_counts()
```

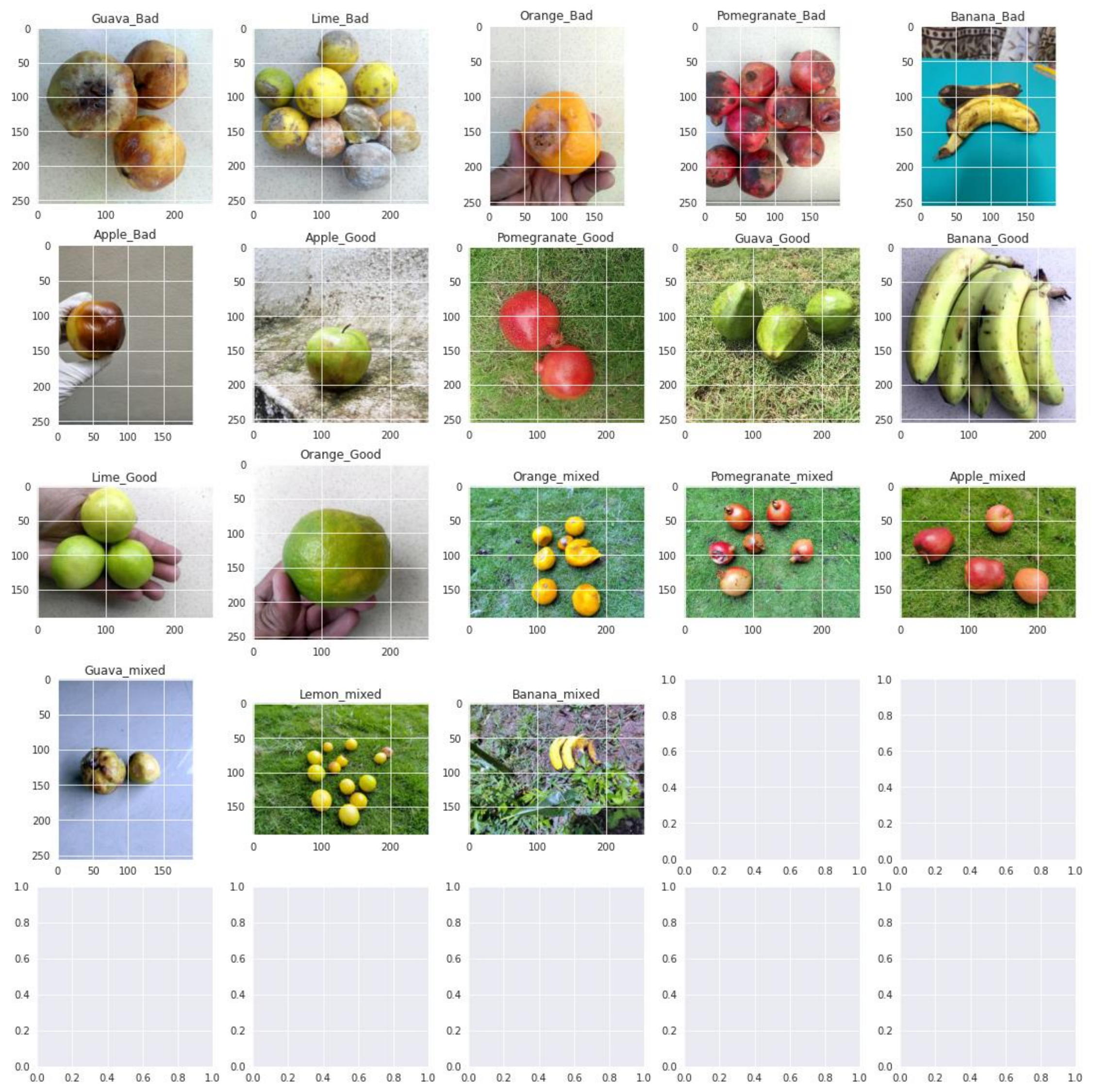
```
Pomegranate_Good     5940
Orange_Good          1216
Pomegranate_Bad      1187
Orange_Bad           1159
Guava_Good           1152
Apple_Good           1149
Apple_Bad            1141
Guava_Bad            1129
Banana_Good          1113
Lime_Good            1094
Banana_Bad           1087
Lime_Bad             1085
Banana_mixed          285
Lemon_mixed           278
Guava_mixed           148
Orange_mixed          125
Pomegranate_mixed     125
Apple_mixed           113
Name: labels, dtype: int64
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df['labels'].unique()
```

```
['Guava_Bad', 'Lime_Bad', 'Orange_Bad', 'Pomegranate_Bad', 'Banana_Bad',
..., 'Pomegranate_mixed', 'Apple_mixed', 'Guava_mixed', 'Lemon_mixed',
'Banana_mixed']
Length: 18
Categories (18, object): ['Apple_Bad', 'Apple_Good', 'Apple_mixed',
'Banana_Bad', ..., 'Orange_mixed', 'Pomegranate_Bad', 'Pomegranate_Good',
'Pomegranate_mixed']
```

```
fig, ax = plt.subplots(nrows=5, ncols=5, figsize=(15,15),
constrained_layout=True)
ax=ax.flatten()
j=0
for i in df['labels'].unique():

    ax[j].imshow(plt.imread(df[df['labels']==i].iloc[0,0]))
    ax[j].set_title(i)
    j=j+1
```

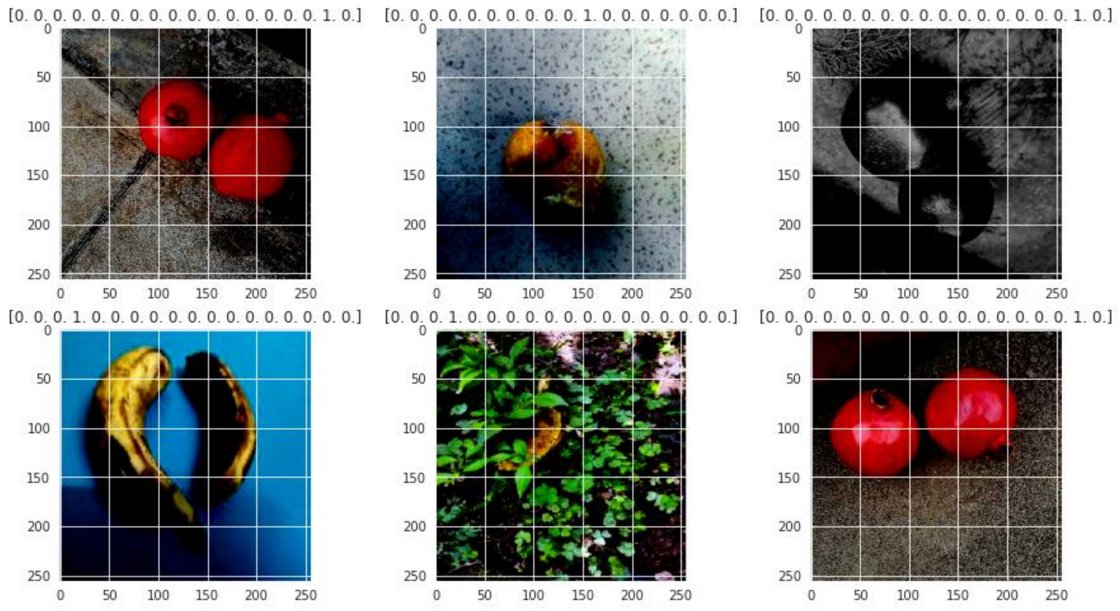


```
from sklearn.model_selection import train_test_split
X_train, X_test=train_test_split(df, test_size=0.2, random_state=123)

print(X_train.shape)
print(X_test.shape)

(15620, 2)
(3906, 2)

from tensorflow.keras.applications.inception_resnet_v2 import
InceptionResNetV2
from tensorflow.keras.applications.inception_resnet_v2 import
preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator

trainGen = ImageDataGenerator(preprocessing_function=preprocess_input,
                              validation_split=0.2)
testGen =ImageDataGenerator(preprocessing_function= preprocess_input)
```

```
X_train_img = trainGen.flow_from_dataframe(dataframe=X_train,
                                           x_col='filepaths',
                                           y_col='labels',
                                           class_mode='categorical',
                                           subset='training',
                                           batch_size=32)
X_val_img = trainGen.flow_from_dataframe(dataframe=X_train,
                                         x_col='filepaths',
                                         y_col='labels',
                                         class_mode='categorical',
                                         subset='validation',
                                         batch_size=32)
X_test_img =testGen.flow_from_dataframe(dataframe=X_test,
                                        x_col='filepaths',
                                        y_col='labels',
                                        class_mode='categorical',
                                        batch_size=32,
                                        shuffle=False)
```

```
Found 12496 validated image filenames belonging to 18 classes.
Found 3124 validated image filenames belonging to 18 classes.
Found 3906 validated image filenames belonging to 18 classes.
```

```
fit, ax= plt.subplots(nrows=2, ncols=3, figsize=(15,8))
ax=ax.flatten()
j=0
for _ in range(6):
    img, label = X_test_img.next()
    #print(img.shape)   #  (1,256,256,3)
    ax[j].imshow(img[0],)
    ax[j].set_title(label[0])
    #plt.show()
    j=j+1
```



```
X_test_img[0][0].shape
```

```
(32, 256, 256, 3)
```

```
image_shape=(256,256,3)
```

```
X_train_img.class_indices
```

```
{'Apple_Bad': 0,
 'Apple_Good': 1,
 'Apple_mixed': 2,
 'Banana_Bad': 3,
 'Banana_Good': 4,
 'Banana_mixed': 5,
 'Guava_Bad': 6,
 'Guava_Good': 7,
 'Guava_mixed': 8,
 'Lemon_mixed': 9,
 'Lime_Bad': 10,
 'Lime_Good': 11,
 'Orange_Bad': 12,
 'Orange_Good': 13,
 'Orange_mixed': 14,
 'Pomegranate_Bad': 15,
 'Pomegranate_Good': 16,
 'Pomegranate_mixed': 17}
```

```
X_val_img.class_indices
```

```
{'Apple_Bad': 0,
 'Apple_Good': 1,
 'Apple_mixed': 2,
 'Banana_Bad': 3,
 'Banana_Good': 4,
 'Banana_mixed': 5,
 'Guava_Bad': 6,
 'Guava_Good': 7,
 'Guava_mixed': 8,
 'Lemon_mixed': 9,
 'Lime_Bad': 10,
 'Lime_Good': 11,
 'Orange_Bad': 12,
 'Orange_Good': 13,
 'Orange_mixed': 14,
 'Pomegranate_Bad': 15,
 'Pomegranate_Good': 16,
 'Pomegranate_mixed': 17}
```

```
X_test_img.class_indices
```

```
{'Apple_Bad': 0,
 'Apple_Good': 1,
 'Apple_mixed': 2,
 'Banana_Bad': 3,
 'Banana_Good': 4,
 'Banana_mixed': 5,
 'Guava_Bad': 6,
 'Guava_Good': 7,
 'Guava_mixed': 8,
 'Lemon_mixed': 9,
 'Lime_Bad': 10,
 'Lime_Good': 11,
 'Orange_Bad': 12,
 'Orange_Good': 13,
 'Orange_mixed': 14,
 'Pomegranate_Bad': 15,
 'Pomegranate_Good': 16,
 'Pomegranate_mixed': 17}
```

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2)
])
```

```
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense,
Conv2D, MaxPooling2D, GlobalAveragePooling2D
pre_trained= InceptionResNetV2(include_top=False, input_shape=image_shape)

#for layers in pre_trained.layers:
#    layers.trainable=False
pre_trained.trainable=False

inputs = pre_trained.input
x = data_augmentation(inputs)
x = pre_trained(x, training=False)
x = GlobalAveragePooling2D()(x)
x = Dropout(0.4)(x)
output=Dense(18, activation='softmax')(x)
model = Model(inputs=inputs, outputs=output)



base_learning_rate = 0.0001
model.compile(loss='categorical_crossentropy',

optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
              metrics=['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/inception_resnet_v2/inception_resnet_v2_weights_tf_dim_orderi
ng_tf_kernels_notop.h5
219062272/219055592 [==============================] - 7s 0us/step
219070464/219055592 [==============================] - 7s 0us/step
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
sequential (Sequential)      (None, 256, 256, 3)       0
_________________________________________________________________
inception_resnet_v2 (Functio (None, 6, 6, 1536)        54336736
_________________________________________________________________
global_average_pooling2d (Gl (None, 1536)              0
_________________________________________________________________
dropout (Dropout)            (None, 1536)              0
_________________________________________________________________
dense (Dense)                (None, 18)                27666
=================================================================
Total params: 54,364,402
Trainable params: 27,666
Non-trainable params: 54,336,736
_________________________________________________________________
```

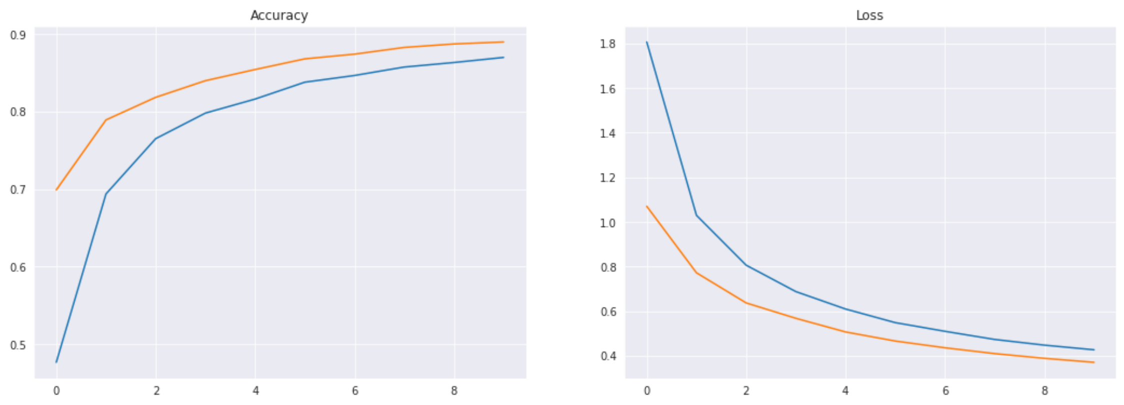
```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss',patience=1)

initial_epoch = 10
results = model.fit(X_train_img,epochs= initial_epoch,
                              validation_data=X_val_img,
                                callbacks=[early_stop])
```

```
Epoch 1/10
391/391 [==============================] - 334s 806ms/step - loss: 1.8070
- accuracy: 0.4765 - val_loss: 1.0701 - val_accuracy: 0.6988
Epoch 2/10
391/391 [==============================] - 218s 557ms/step - loss: 1.0296
- accuracy: 0.6936 - val_loss: 0.7719 - val_accuracy: 0.7891
Epoch 3/10
391/391 [==============================] - 215s 550ms/step - loss: 0.8059
- accuracy: 0.7649 - val_loss: 0.6367 - val_accuracy: 0.8182
Epoch 4/10
391/391 [==============================] - 218s 558ms/step - loss: 0.6874
- accuracy: 0.7979 - val_loss: 0.5676 - val_accuracy: 0.8396
Epoch 5/10
391/391 [==============================] - 219s 561ms/step - loss: 0.6093
- accuracy: 0.8159 - val_loss: 0.5065 - val_accuracy: 0.8540
Epoch 6/10
391/391 [==============================] - 222s 568ms/step - loss: 0.5483
- accuracy: 0.8377 - val_loss: 0.4657 - val_accuracy: 0.8678
Epoch 7/10
391/391 [==============================] - 222s 566ms/step - loss: 0.5094
- accuracy: 0.8464 - val_loss: 0.4355 - val_accuracy: 0.8739
Epoch 8/10
391/391 [==============================] - 219s 560ms/step - loss: 0.4728
- accuracy: 0.8572 - val_loss: 0.4093 - val_accuracy: 0.8825
Epoch 9/10
391/391 [==============================] - 219s 558ms/step - loss: 0.4473
- accuracy: 0.8631 - val_loss: 0.3882 - val_accuracy: 0.8870
Epoch 10/10
```

```
391/391 [==============================] - 218s 557ms/step - loss: 0.4265
- accuracy: 0.8696 - val_loss: 0.3702 - val_accuracy: 0.8896
```

```
result_df = pd.DataFrame(results.history)
fig, ax=plt.subplots(nrows=1, ncols=2,figsize=(18,6))
ax=ax.flatten()
ax[0].plot(result_df[['accuracy','val_accuracy']])
ax[0].set_title("Accuracy")
ax[1].plot(result_df[['loss','val_loss']])
ax[1].set_title("Loss")
```

```
Text(0.5, 1.0, 'Loss')
```



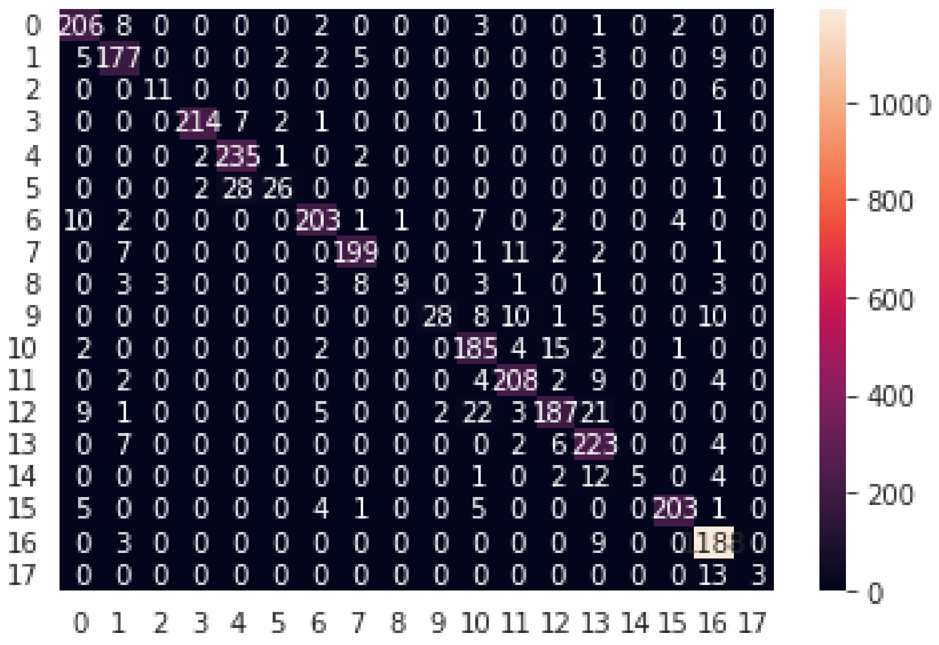
```
pred = model.predict(X_test_img)
pred=np.argmax(pred,axis=1)
```

```
pred_df=X_test.copy()
labels={}
for l,v in X_test_img.class_indices.items():
    labels.update({v:l})
pred_df['pred']=pred
pred_df['pred']=pred_df['pred'].apply(lambda x: labels[x])
```

```
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
print(f"Accuracy Score:
{accuracy_score(pred_df['labels'],pred_df['pred'])}")
sns.heatmap(confusion_matrix(pred_df['labels'],pred_df['pred']),
annot=True, fmt='2d')
```

```
Accuracy Score: 0.8986175115207373
```

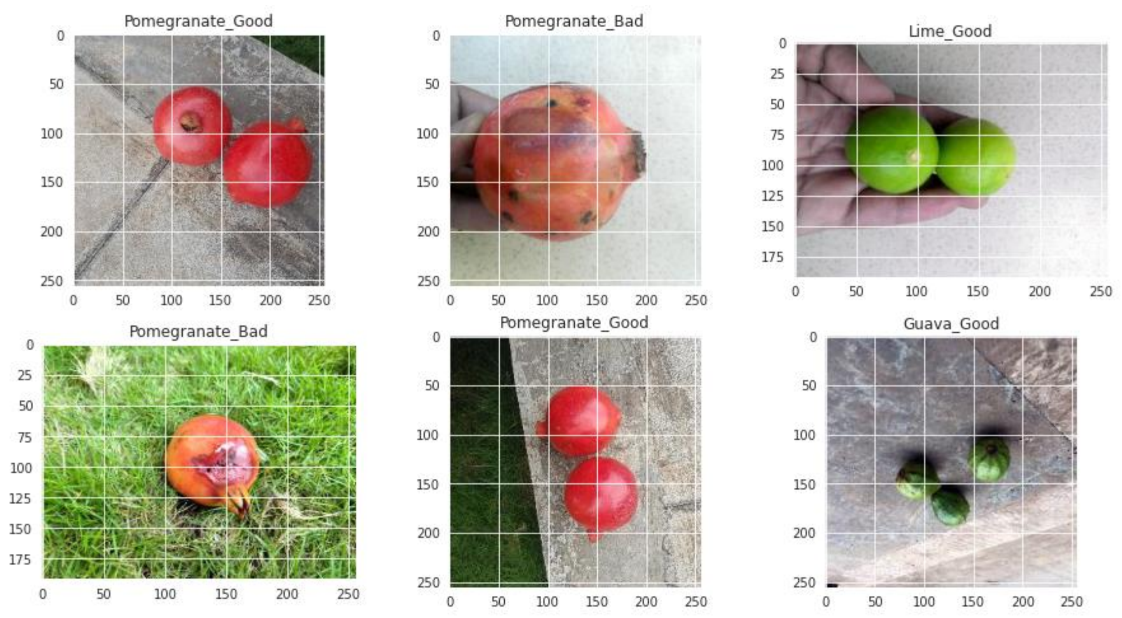
```
<AxesSubplot:>
```



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	206	8	0	0	0	0	2	0	0	0	3	0	0	1	0	2	0	0
1	5	177	0	0	0	2	2	5	0	0	0	0	0	3	0	0	9	0
2	0	0	11	0	0	0	0	0	0	0	0	0	0	1	0	0	6	0
3	0	0	0	214	7	2	1	0	0	0	1	0	0	0	0	0	1	0
4	0	0	0	2	235	1	0	2	0	0	0	0	0	0	0	0	0	0
5	0	0	0	2	28	26	0	0	0	0	0	0	0	0	0	0	1	0
6	10	2	0	0	0	0	203	1	1	0	7	0	2	0	0	4	0	0
7	0	7	0	0	0	0	0	199	0	0	1	11	2	2	0	0	1	0
8	0	3	3	0	0	0	3	8	9	0	3	1	0	1	0	0	3	0
9	0	0	0	0	0	0	0	0	0	28	8	10	1	5	0	0	10	0
10	2	0	0	0	0	0	2	0	0	0	185	4	15	2	0	1	0	0
11	0	2	0	0	0	0	0	0	0	0	4	208	2	9	0	0	4	0
12	9	1	0	0	0	0	5	0	0	2	22	3	187	21	0	0	0	0
13	0	7	0	0	0	0	0	0	0	0	0	2	6	223	0	0	4	0
14	0	0	0	0	0	0	0	0	0	0	1	0	2	12	5	0	4	0
15	5	0	0	0	0	0	4	1	0	0	5	0	0	0	0	203	1	0
16	0	3	0	0	0	0	0	0	0	0	0	0	0	9	0	0	1188	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	3

```python
print(pred_df[pred_df['labels']==pred_df['pred']].head(6))
fig, ax=plt.subplots(nrows=2, ncols=3, figsize=(15,8))
ax=ax.flatten()
imlist=pred_df[pred_df['labels']==pred_df['pred']].head(6).reset_index()
for i in range(0,6):
    ax[i].imshow(plt.imread(imlist['filepaths'][i]))
    ax[i].set_title(imlist['labels'][i])
```

```
                                               filepaths             labels  \
8323   ../input/fruitnet-indian-fruits-dataset-with-q...  Pomegranate_Good   
4204   ../input/fruitnet-indian-fruits-dataset-with-q...   Pomegranate_Bad   
16867  ../input/fruitnet-indian-fruits-dataset-with-q...         Lime_Good   
4413   ../input/fruitnet-indian-fruits-dataset-with-q...   Pomegranate_Bad   
12590  ../input/fruitnet-indian-fruits-dataset-with-q...  Pomegranate_Good   
14868  ../input/fruitnet-indian-fruits-dataset-with-q...        Guava_Good   

                   pred  
8323   Pomegranate_Good  
4204    Pomegranate_Bad  
16867         Lime_Good  
4413    Pomegranate_Bad  
12590  Pomegranate_Good  
14868        Guava_Good  
```



Fine Tuning

```
pre_trained.trainable = True

# Let's take a look to see how many layers are in the base model
print("Number of layers in the base model: ", len(pre_trained.layers))

# Fine-tune from this layer onwards
fine_tune_at = 100

# Freeze all the layers before the `fine_tune_at` layer
for layer in pre_trained.layers[:fine_tune_at]:
  layer.trainable = False
```

```
Number of layers in the base model:  780
```

```
model.compile(loss='categorical_crossentropy',
              optimizer = 
tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate/10),
              metrics=['accuracy'])

model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
sequential (Sequential)      (None, 256, 256, 3)       0
_________________________________________________________________
inception_resnet_v2 (Functio (None, 6, 6, 1536)        54336736
_________________________________________________________________
global_average_pooling2d (Gl (None, 1536)              0
_________________________________________________________________
dropout (Dropout)            (None, 1536)              0
```

```
________________________________________________________________
dense (Dense)                (None, 18)               27666
================================================================
Total params: 54,364,402
Trainable params: 53,536,290
Non-trainable params: 828,112
________________________________________________________________
```

```
len(model.trainable_variables)
```

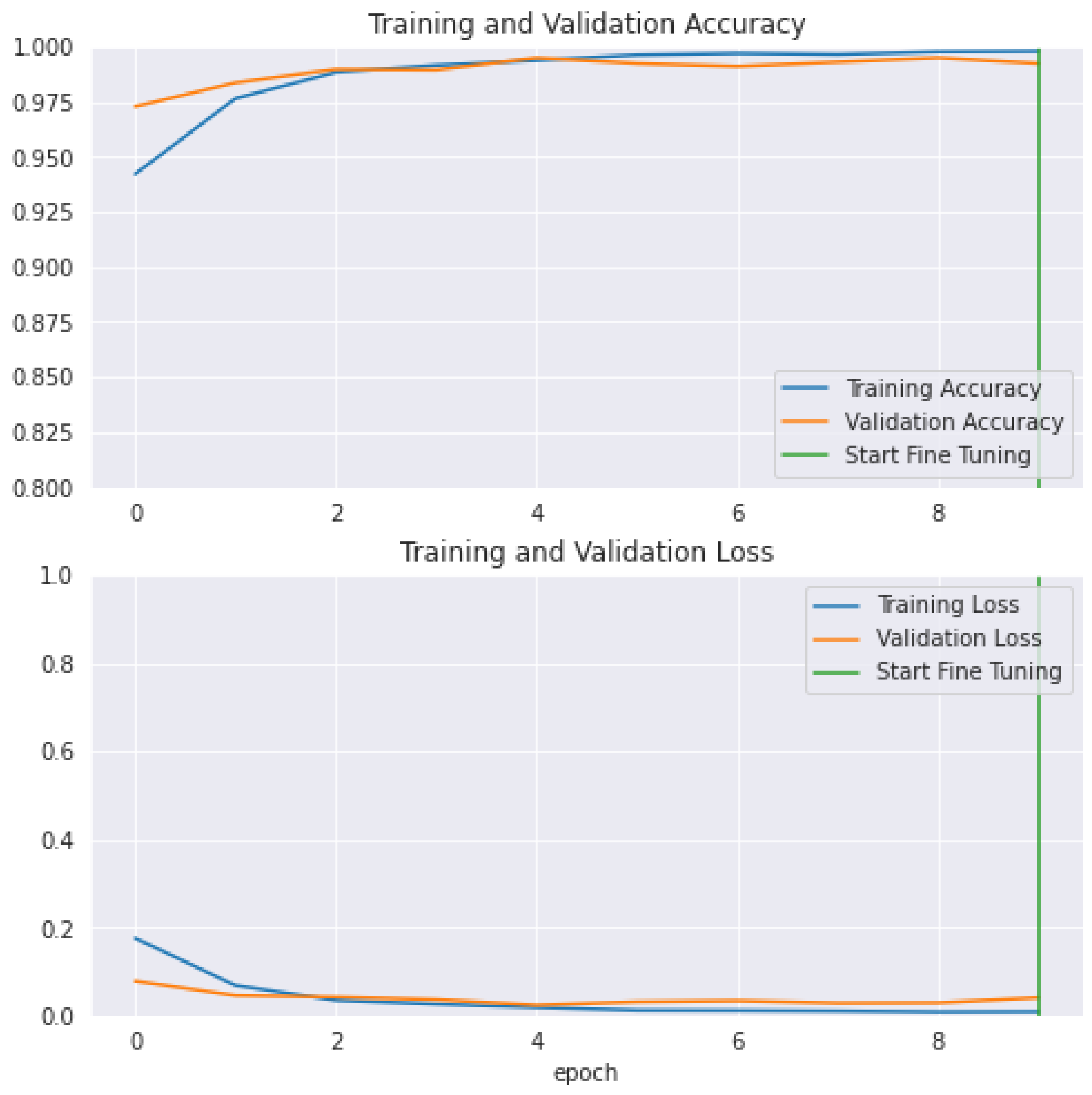
```
426
```

```
fine_tune_epochs = 10
total_epochs =  initial_epoch + fine_tune_epochs

history_fine = model.fit(X_train_img,
                         epochs=total_epochs,
                         initial_epoch=10,
                         validation_data=X_val_img)
```

```
Epoch 11/20
391/391 [==============================] - 370s 875ms/step - loss: 0.1761
- accuracy: 0.9421 - val_loss: 0.0790 - val_accuracy: 0.9728
Epoch 12/20
391/391 [==============================] - 342s 874ms/step - loss: 0.0691
- accuracy: 0.9765 - val_loss: 0.0462 - val_accuracy: 0.9837
Epoch 13/20
391/391 [==============================] - 338s 865ms/step - loss: 0.0351
- accuracy: 0.9884 - val_loss: 0.0427 - val_accuracy: 0.9898
Epoch 14/20
391/391 [==============================] - 342s 874ms/step - loss: 0.0266
- accuracy: 0.9915 - val_loss: 0.0369 - val_accuracy: 0.9894
Epoch 15/20
391/391 [==============================] - 342s 874ms/step - loss: 0.0191
- accuracy: 0.9938 - val_loss: 0.0242 - val_accuracy: 0.9949
Epoch 16/20
391/391 [==============================] - 339s 864ms/step - loss: 0.0125
- accuracy: 0.9962 - val_loss: 0.0318 - val_accuracy: 0.9923
Epoch 17/20
391/391 [==============================] - 339s 866ms/step - loss: 0.0120
- accuracy: 0.9969 - val_loss: 0.0340 - val_accuracy: 0.9910
Epoch 18/20
391/391 [==============================] - 340s 867ms/step - loss: 0.0109
- accuracy: 0.9964 - val_loss: 0.0290 - val_accuracy: 0.9930
Epoch 19/20
391/391 [==============================] - 340s 866ms/step - loss: 0.0089
- accuracy: 0.9976 - val_loss: 0.0296 - val_accuracy: 0.9949
Epoch 20/20
391/391 [==============================] - 339s 866ms/step - loss: 0.0093
- accuracy: 0.9978 - val_loss: 0.0413 - val_accuracy: 0.9923
```

```
acc = []
val_acc = []
loss = []
val_loss = []
```

```
acc += history_fine.history['accuracy']
val_acc += history_fine.history['val_accuracy']

loss += history_fine.history['loss']
val_loss += history_fine.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.ylim([0.8, 1])
plt.plot([initial_epoch-1,initial_epoch-1],
          plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.ylim([0, 1.0])
plt.plot([initial_epoch-1,initial_epoch-1],
         plt.ylim(), label='Start Fine Tuning')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```



```
model.save("fruitNet.h5")

def predict_fruit_quality(model, img_path):
    # Load the image
    img = image.load_img(img_path, target_size=(256, 256))  # Update
target size to (256, 256)
    img_array = image.img_to_array(img)  # Convert image to array
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    img_array = preprocess_input(img_array)  # Preprocess the input for
the model

    # Make prediction
    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction, axis=1)[0]

    # Map the predicted class back to the fruit label
    labels = {v: k for k, v in X_train_img.class_indices.items()}  # Get
the mapping of classes
    predicted_label = labels[predicted_class]
```

```python
    return predicted_label

# Example usage
image_path = '/kaggle/input/fruitnet-indian-fruits-dataset-with-
quality/Processed Images_Fruits/Bad
Quality_Fruits/Guava_Bad/IMG20200728184716.jpg'  # Replace with the path
to your image
predicted_quality = predict_fruit_quality(model, image_path)
print(f'Predicted Fruit Quality: {predicted_quality}')
```

```
Predicted Fruit Quality: Guava_Bad
```