Internship Research Project

In Machine learning

& Deep learning

Under supervise of

Pr R Sujatha

10/14/2022 – 12/9/2022

Indian fruits condition Recognition: machine learning model with high accuracy applied on some Indian fruits dataset .

Omar Elsayed

Abstract

A machine learning & deep learning research project used some good quality Indian fruit images dataset the got separated and processed separately and together with machine learning & deep learning model created by (inceptionv3,Xception,VGG19,VGG16) transfer learning application and also applying some other algorithm we will walk throw the code in details, then testing the performance of a model by some plots and the confusion matrix with showing some of those visualization, and an Xcel table for brief of the numeric result ,all to get the best accuracy so we can use it as a useful model to apply it in the farmers, agriculture industries, wholesalers, hawkers, and customers, and fruit export companies

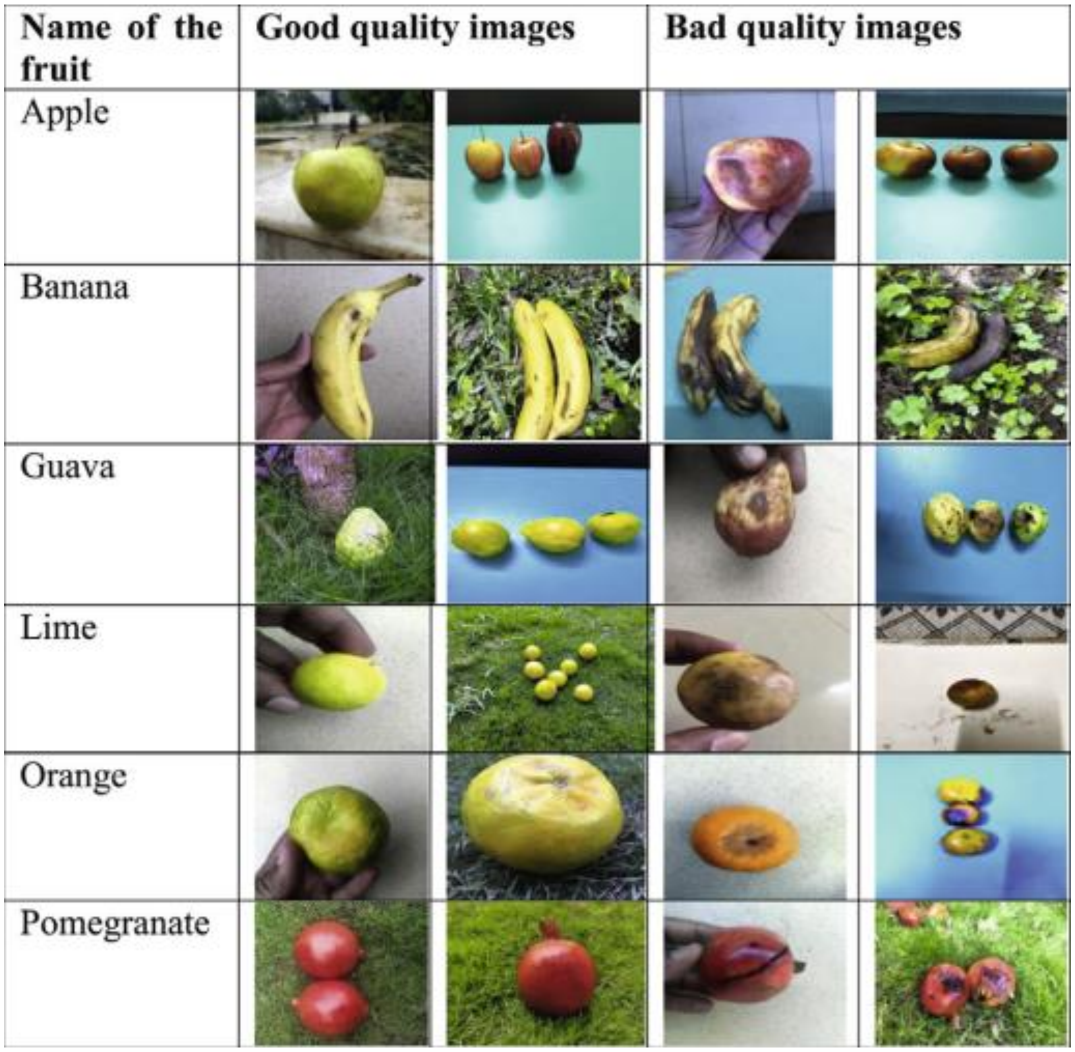
Introduction

A machine learning & deep learning research project is made to generate models with good accuracy by applying different transfer learning techniques (Inceptionv3-Xception-VGG19-VGG16) to classify the most famous Indian fruits and there condition if the fruit is in a good codition and Suitable for eating or it is in a bad condition, and for this project we chose **FruitNet: Indian fruits image dataset with quality for machine learning applications** [1] to work on, the profit percentage share of fruit market is substantial with respect to the total of the agriculture output. In the agro-industry fast and accurate fruit classification is very needed. The fruits can be classified into different classes as per their external features like shape, and size and color using the computer vision and deep learning techniques, so this research aims to serve those fields by trying different methods to get the most accurate model that can be used in many different areas of agriculture fields and the fields related to it, and we will show all the processes and their results in our research in an easy way

Literature Review

1. The dataset is comprehensive which consist of 12000+ high-quality images of six different classes, The dataset contains good quality, and bad quality, fruit images The dataset is ready to build fruit classification with very good quality applications which are beneficial for farmers, agriculture industries, wholesalers, hawkers, and customers, and fruit export companies.

It consists of six classes of Indian fruits named as apple, banana, guava, lime, orange, and pomegranate. They further categorized under good quality, and bad quality. the fruit images were taken in different background, and in different light, The fruit images captured in the natural and artificial lighting conditions with different angles and background in different months (from July to October). Images pre-processing is done using python. at the pre-processing they changed the dimensions of the image to 256 × 256 because it is the standard resolution required to build object classification or object detection model, but hear we going to re-size them again according to each deep learning application we use.



Name of the fruit	Good quality images		Bad quality images	
Apple				
Banana				
Guava				
Lime				
Orange				
Pomegranate				

A quick walk throws about what we did we split the data to training and validation with python then we coped each fruit separately in a new folder then we applied our code with the four-transfer learning application we chose to work by on each folder and the mean folder also then we got the result and calculated the recall and precision and visual it and finally show the confusion matrix

2. [2] that article presents the dataset that contain 759 images of healthy and unhealthy citrus fruits and leaves, of the famous species among the fruit plants is a citrus plant, which is full of vitamin C and broadly used in the region of the Middle East and Africa, and the most common disease identified by the domain experts and researchers are the Greening, the Melanose, the Downy, the Black the spot, the Canker, the Scab, and the Anthracnose.

3. [3] that article presents the dataset of the guava (disease affected, and disease-free) it contains a total of 681 images that are collected from the guava garden of Bangladesh Agriculture University, Mymensingh, Dhaka.

4.[4] A comprehensive review on detection of some plant disease using machine learning and deep learning approaches using different techniques of transfer learning by a dataset of plant leaves.

5. [5] A lightweight network for the mummy berry disease recognition, during the growth of the blueberries, they are often infected by the mummy berry disease (Monilinia vaccinii-corymbosi (Reade)), which leads to the decline of the blueberry most of the deep convolutional neural networks, such as (GoogLeNet, AlexNet, VGGNet and ResNet 50), have a large parameters due to their deep and complex network structure, production and huge economic losses.

6. [6] A Leaf disease recognition using the image processing and deep learning techniques, this approach limits the resulting models' ability to estimate leaf disease severity or identify multiple anomalies occurring on the same leaf.

7.[7] A deep learning system for classify the strawberry disease based on the cloud service classifying crop diseases and insect pests disasters in China.

Considering incidence of different strawberry diseases and their impact on the strawberry yield and quality, study of fifteen common strawberry diseases. Using the Scrapy web crawler to crawl nearly 3000 strawberry disease images

8.[8] A sunflower image dataset that containing the total of four hundred and sixty-seven original images and sixteen hundred and sixty-eight augmented images prepared from the original images of healthy and disease-affected sunflower on leaves and on blooms.

9.[9] An environment-friendly agricultural deep learning system for ensuring that the long-term security of the food for the people of Bangladesh with the accuracy of 89.59% using Random Forest classifier for jackfruit disease recognition

10.[10] evaluating the agronomic potential and genetic dissimilarity of the saladette type dwarf tomato plant populations through the use of artificial neural networks (ANNs).to analyze The mean fruit weight, the transverse and the longitudinal fruit diameter, the fruit shape, the pulp thickness, locule number, internode length, soluble solids content, and β-carotene, lycopene, and the leaf zingiberene contents.

11.[11] A grape leaf diseases identification by a united convolutional neural networks (CNNs) architecture based on an integrated method is proposed. achieving an average validation accuracy of 99.17% and a test accuracy of 98.57%.

12.[12] A Citrus greening disease recognition algorithm based on classification using TRL-GAN, with the classification network is ResNeXt101 gives 97.45% accuracy.

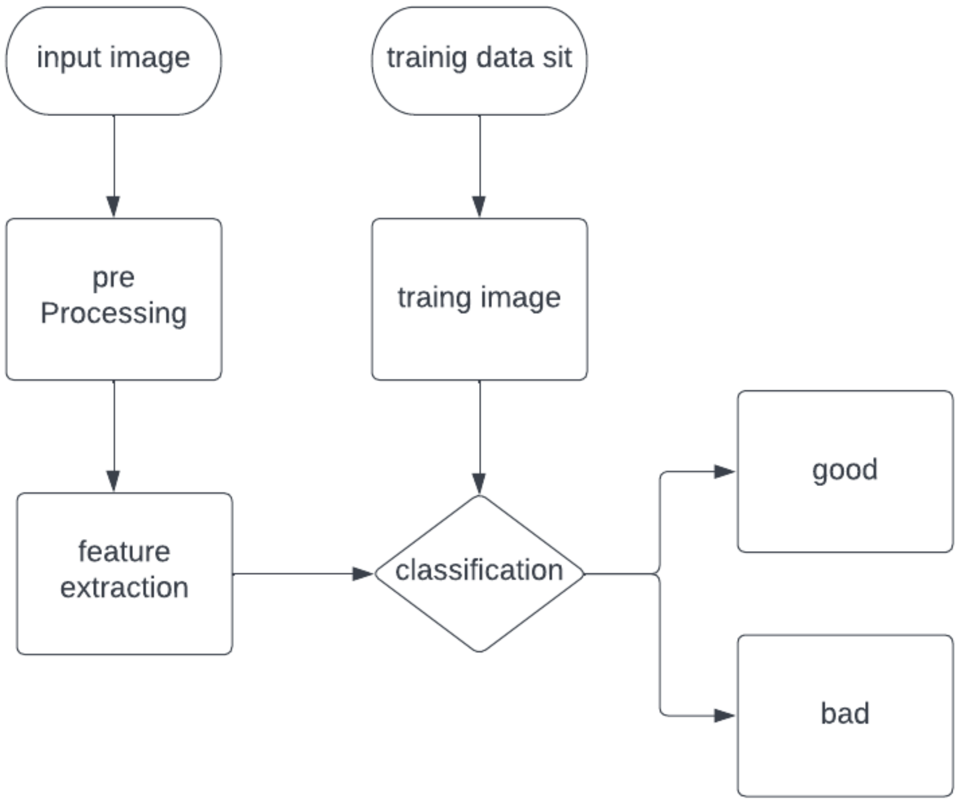
13.[13] automated pest classification in Mango farms, pests that attack the fruit, stem, root or mango leaf. Addressing the need for an early stage automated or semi-automated pest identification system, using VGG-16 deep-learning model to supplement the last layer with a fully connected network training of consisting of 2-layers.

14[14] plant classification and crop disease classification deep learning by based on plant and crop disease images that were acquired using a visible light camera, which contained 4,720 various images of flowers and leaves and an open database of crop diseases was also used, Gives a 98.55% accuracy for the thermal plant image dataset and a 90.04% accuracy for the Paddy crop dataset

15.[15] Grapes and mango leaf disease detection by transfer learning in India employing a dataset of 8,438 images of diseased and healthy leaves collected from the Plant Village dataset and acquired locally, using AlexNet to get accuracy rate of the detection of 99% and 89% for Grape leaves and Mango leaves respectively.

16.[16] papaya disease recognition system in order to help distant farmers, so it does two main things disease detection and disease detection and, giving More than 90% classification accuracy

GeneralSteps



Work Steps

1.uplode the data to the drive

2.split the data to train and validation

3. re-size all the images

4.prepare and create the model

5.generate more image

6.Prepare the training set and the validation set

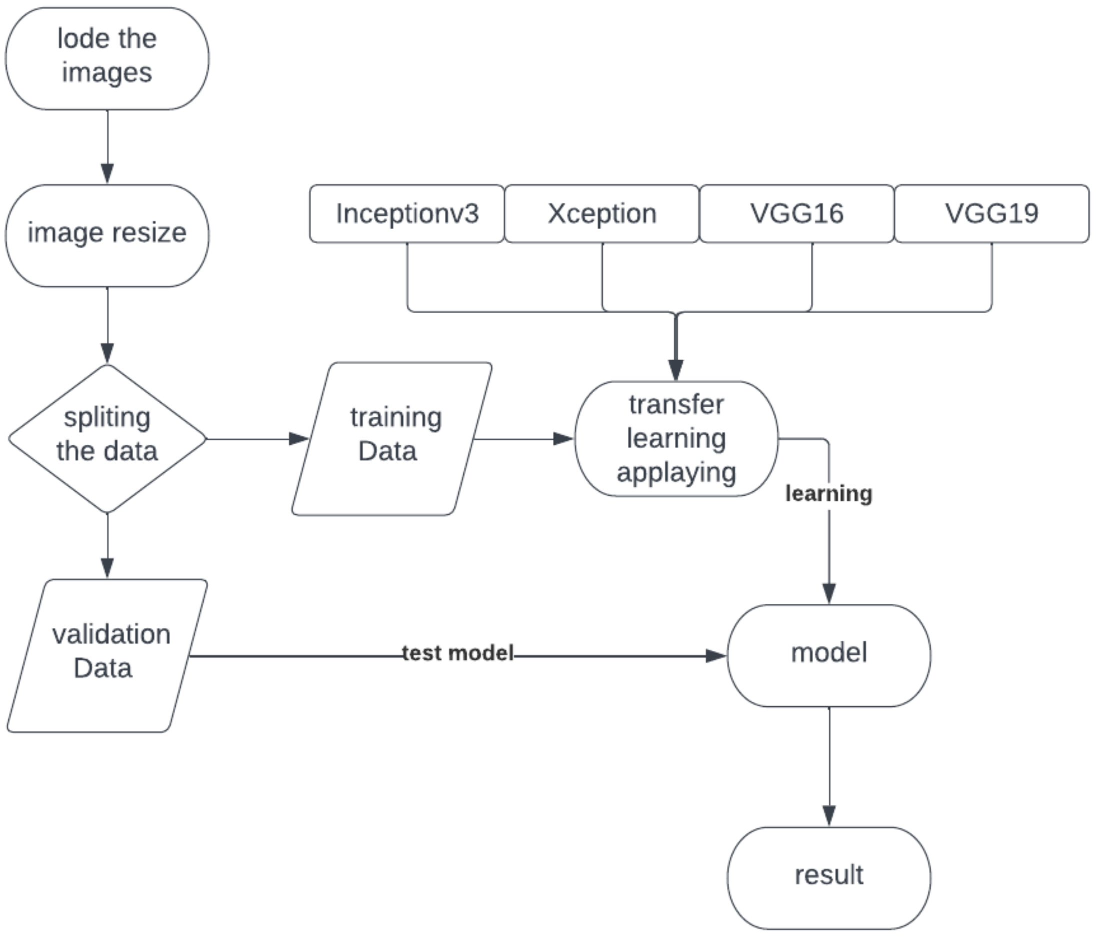
7.fit the model

8.visualis the recall and precision

9. draw the confusion matrix

10.save the model

System architecture



Code description and wake throw

In this project we used Google Colab to write our code so first we going to import the libraries we need in the project .

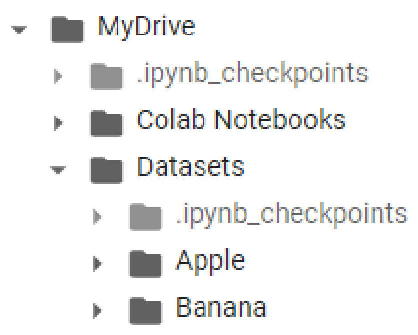
We will show the wake throw on the Apple model using Inceptionv3

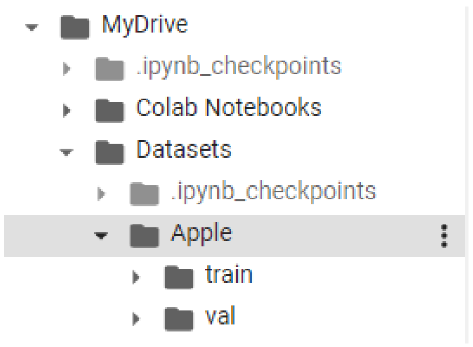
```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.models import load_model
from sklearn.metrics import classification_report, confusion_matrix
import itertools
```

In this project we going to use the transfer learning applications and we chose inception v3 so we need to prepare the images to be able to be used in it ,the image size in this dataset is 256 x 256 and the inception v3 works with image size 299 x 299

And as we mentioned earlier, we split the data to training and validation with python then we coped each fruit separately in a new folder all on google drive

connected to our colab so the data folders looks like that





So we will work on the Apple model as an example, so we will resize the image size to fit the inception v3 and upload the training and validation data paths to our colab

```
IMAGE_SIZE = [299, 299]

train_path = '/content/drive/MyDrive/Datasets/Apple/train'
valid_path = '/content/drive/MyDrive/Datasets/Apple/val'
```

Then we prepare our model normally we add preprocessing layer to the front of inception v3 and don't train existing weights and we apply the image size we add

```
inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)
for layer in inception.layers:
  layer.trainable = False
```

Then we get the getting number of classes by glob, and we flatten the output layer and add the dense

```
folders = glob('/content/drive/MyDrive/Datasets/Apple/train/*')
x = Flatten()(inception.output)
prediction = Dense(len(folders), activation='softmax')(x)
```

then we create a model object and prepare the matrixes

```
model = Model(inputs=inception.input, outputs=prediction)
accuracy = tf.keras.metrics.CategoricalAccuracy()
auc = tf.keras.metrics.AUC()
precision =  tf.keras.metrics.Precision()
recall = tf.keras.metrics.Recall()
```

then tell the model what cost and optimization method to use

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

then using image data generator to generate more image to get more accuracy

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

then getting the training set and test set ready

```
training_set = train_datagen.flow_from_directory(train_path,
                                                 target_size = (299, 299),
                                                 batch_size = 32,
                                                 class_mode = 'categorical
',shuffle= True)
test_set = test_datagen.flow_from_directory(valid_path,
                                            target_size = (299, 299),
                                            batch_size = 32,
                                            class_mode = 'categorical',shu
ffle= False)
```

then we going to fit our data into the model and we going to use the early stopping criteria to to get rid of unnecessary epochs and to training reduce time

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_accuracy', mode='max',  patien
ce=4)

r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=25,
  steps_per_epoch=len(training_set),
  #validation_steps=len(test_set)
  callbacks=[early_stopping])
```

then we going to use matplotlib to visualize the matrixes :

the loss and the val_loss

the accuracy and the val_accuracy

the auc

the precision

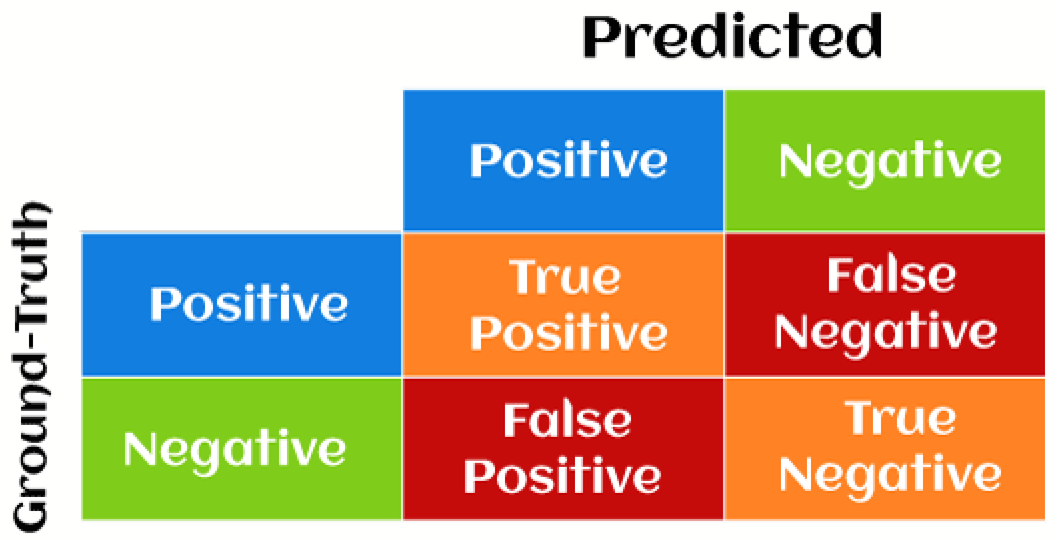
the recall

and finally the precision and recall

```
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
#plot the auc
plt.plot(r.history['auc'], label='AUC')
plt.xlabel('Epoch')
plt.ylabel('AUC')
plt.legend()
plt.show()
plt.savefig('auc')
# plot the precision
plt.plot(r.history['precision'], label='Precision')
plt.xlabel('Epoch')
plt.ylabel('Precision')
plt.legend()
plt.show()
plt.savefig('prec')
# plot the recall
plt.plot(r.history['recall'], label='Recall')
plt.xlabel('Epoch')
plt.ylabel('Recall')
plt.legend()
plt.show()
plt.savefig('reca')
# plot the precision/recall
plt.plot(r.history['precision'], label='Precision')
plt.plot(r.history['recall'], label='Recall')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend()
plt.show()
plt.savefig('recaprec')
```

and finally we prepare the confusion matrix and draw it

Confusion Matrix helps us to display the performance of a model or how a model has made its prediction in Machine Learning. [17]



```
def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion ma
trix', cmap=plt.cm.Blues):

    """
    This function prints and plots the confusion matrix.

    Normalization can be applied by setting `normalize=True`.

    """
    plt.figure(figsize=(10,10))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))

    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        cm = np.around(cm, decimals=2)
        cm[np.isnan(cm)] = 0.0
        print("Normalized confusion matrix")

    else:
        print('Confusion matrix, without normalization')
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


target_names = []

for key in training_set.class_indices:

    target_names.append(key)

Y_pred = model.predict_generator(test_set)

y_pred = np.argmax(Y_pred, axis=1)

print('Confusion Matrix')

cm = confusion_matrix(test_set.classes, y_pred)

plot_confusion_matrix(cm, target_names, title='Confusion Matrix')
```

last thing saving the model

```
model.save('/content/drive/MyDrive/saved models/Apple2_model.h5')
```

More details

There are four metrics combinations in confusion matrix, which are: [17]

- **True Positive:** This combination tells how many times a model correctly classifies a positive sample as Positive?
- **False Negative:** This combination tells how many times a model incorrectly classifies a positive sample as Negative?
- **False Positive:** This combination tells how many times a model incorrectly classifies a negative sample as Positive?
- **True Negative:** This combination tells how many times a model correctly classifies a negative sample as Negative?

Precision is defined as the *ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples* (either correctly or incorrectly).

1. Precision = True Positive/True Positive + False Positive
2. Precision = TP/TP+FP

- TP- True Positive
- FP- False Positive

- The precision of a machine learning model will be low when the value of;

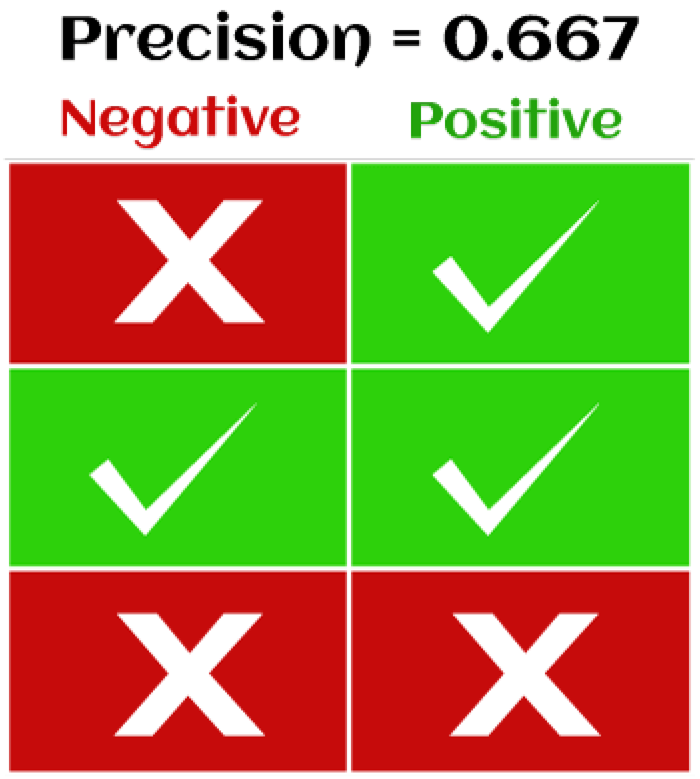
1. TP+FP (denominator) > TP (Numerator)

- The precision of the machine learning model will be high when Value of;

1. TP (Numerator) > TP+FP (denominator)

Precision = TP/TP+FP

Precision = 2/2+1 = 2/3 = 0.667



What is Recall?

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The ***recall measures the model's ability to detect positive samples***. The higher the recall, the more positive samples detected.

1. Recall = True Positive/True Positive + False Negative
2. Recall = TP/TP+FN

- TP- True Positive
- FN- False Negative

- Recall of a machine learning model will be low when the value of; TP+FN (denominator) > TP (Numerator)
- Recall of machine learning model will be high when Value of; TP (Numerator) > TP+FN (denominator)

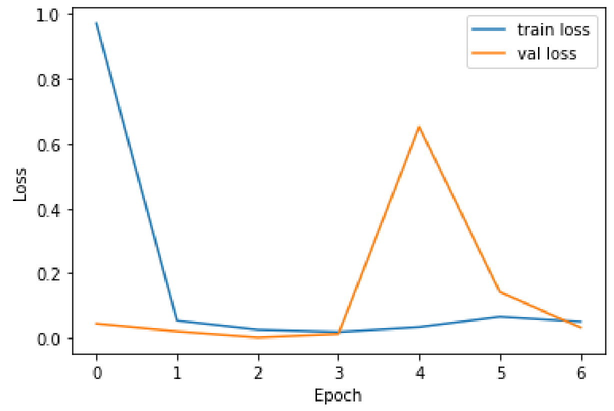
Unlike Precision, Recall is independent of the number of negative sample classifications. Further, if the model classifies all positive samples as positive, then Recall will be 1.

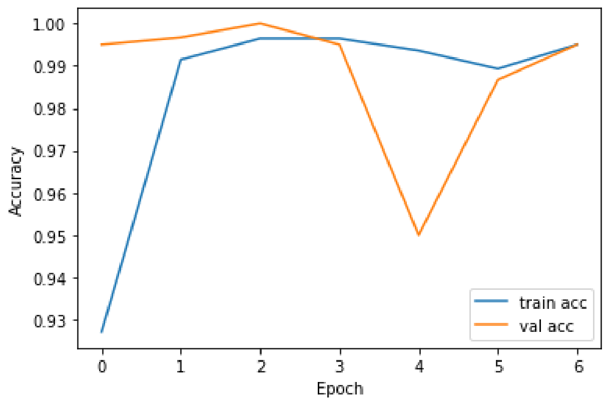
Result

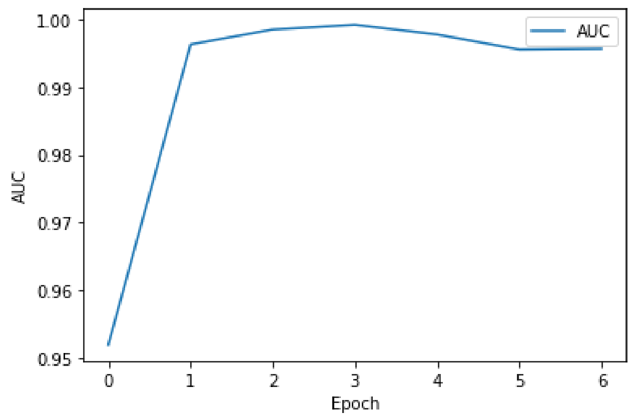
The visualizations for the Apple model we used matplotlib to draw for the loss and the val_loss, accuracy and the val_accuracy, the auc, the precision, the recall, and the precision and recall we got for that result

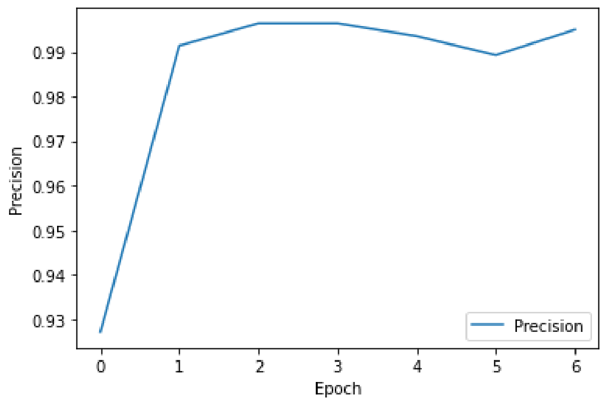
```
Epoch 1/25
44/44 [==============================] - 687s 16s/step - loss: 0.9706 -
accuracy: 0.9271 - auc: 0.9519 - precision: 0.9271 - recall: 0.9271 -
val_loss: 0.0437 - val_accuracy: 0.9950 - val_auc: 0.9950 - val_precision:
0.9950 - val_recall: 0.9950
Epoch 2/25
44/44 [==============================] - 31s 711ms/step - loss: 0.0533 -
accuracy: 0.9914 - auc: 0.9963 - precision: 0.9914 - recall: 0.9914 -
val_loss: 0.0197 - val_accuracy: 0.9967 - val_auc: 0.9983 - val_precision:
0.9967 - val_recall: 0.9967
Epoch 3/25
44/44 [==============================] - 32s 735ms/step - loss: 0.0249 -
accuracy: 0.9964 - auc: 0.9985 - precision: 0.9964 - recall: 0.9964 -
val_loss: 0.0011 - val_accuracy: 1.0000 - val_auc: 1.0000 - val_precision:
1.0000 - val_recall: 1.0000
Epoch 4/25
44/44 [==============================] - 31s 715ms/step - loss: 0.0178 -
accuracy: 0.9964 - auc: 0.9992 - precision: 0.9964 - recall: 0.9964 -
val_loss: 0.0118 - val_accuracy: 0.9950 - val_auc: 1.0000 - val_precision:
0.9950 - val_recall: 0.9950
Epoch 5/25
44/44 [==============================] - 32s 724ms/step - loss: 0.0331 -
accuracy: 0.9936 - auc: 0.9978 - precision: 0.9936 - recall: 0.9936 -
val_loss: 0.6514 - val_accuracy: 0.9500 - val_auc: 0.9580 - val_precision:
0.9500 - val_recall: 0.9500
Epoch 6/25
44/44 [==============================] - 31s 709ms/step - loss: 0.0654 -
accuracy: 0.9893 - auc: 0.9956 - precision: 0.9893 - recall: 0.9893 -
val_loss: 0.1420 - val_accuracy: 0.9867 - val_auc: 0.9883 - val_precision:
0.9867 - val_recall: 0.9867
Epoch 7/25
44/44 [==============================] - 32s 736ms/step - loss: 0.0501 -
accuracy: 0.9950 - auc: 0.9957 - precision: 0.9950 - recall: 0.9950 -
val_loss: 0.0317 - val_accuracy: 0.9950 - val_auc: 0.9966 - val_precision:
0.9950 - val_recall: 0.9950
```

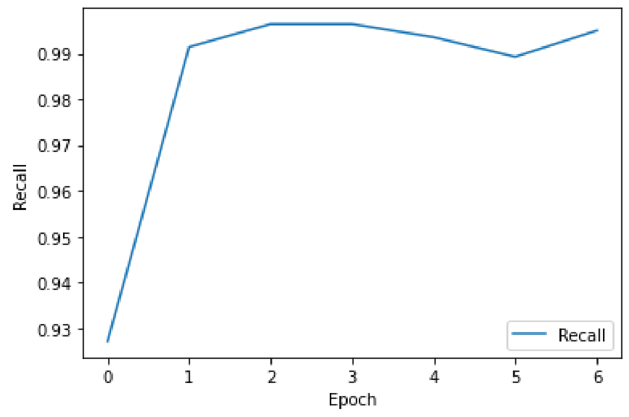
Will be

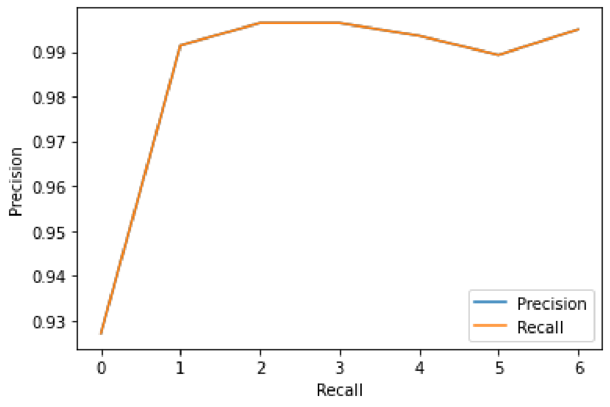

train loss
val loss
Loss
Epoch
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6


Accuracy
Epoch
train acc
val acc
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
1
2
3
4
5
6

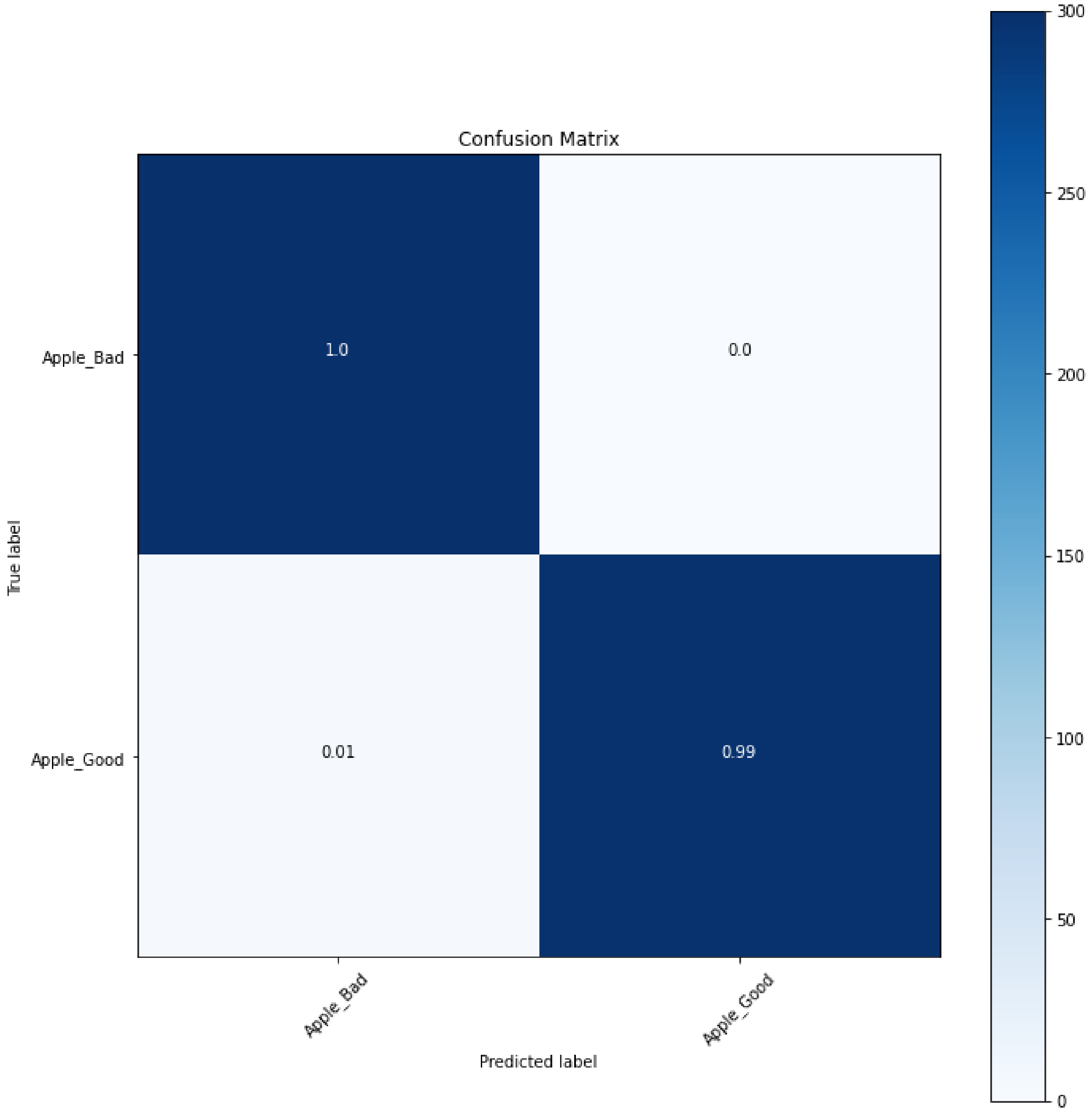

AUC
AUC
Epoch
1.00
0.99
0.98
0.97
0.96
0.95
0
1
2
3
4
5
6


Precision
Precision
Epoch
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
1
2
3
4
5
6


Recall
Recall
Epoch
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
1
2
3
4
5
6


Precision
Recall
Precision
Recall
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
1
2
3
4
5
6

And the confusion matrix will be shown as



This table is showing a brief for all the models result for the inceptionv3

	Epoch	loss	accuracy	precision	recall	val_accuracy	val_precision	val_recall
	1	0.9706	0.9271	0.9271	0.9271	0.995	0.995	0.995
	2	0.0533	0.9914	0.9914	0.9914	0.9967	0.9967	0.9967
Apple	4	0.0178	0.9964	0.9964	0.9964	0.995	0.995	0.995
	5	0.0331	0.9936	0.9936	0.9936	0.95	0.95	0.95
	last 7	0.0501	0.995	0.995	0.995	0.995	0.995	0.995
	1	0.4018	0.9764	0.9764	0.9764	0.99	0.99	0.99
	2	0.0383	0.9964	0.9964	0.9964	1	1	1
Banana	4	0.0471	0.9986	0.9986	0.9986	1	1	1
	5	2.7328e	1	1	1	0.9967	0.9967	0.9967
	last 6	1.0212e	1	1	1	0.9967	0.9967	0.9967
	1	0.6966	0.9543	0.9543	0.9543	0.9983	0.9983	0.9983
	2	0.6966	0.9971	0.9971	0.9971	0.9983	0.9983	0.9983
Guava	4	0.0366	0.9986	0.9986	0.9986	0.9967	0.9967	0.9967
	5	4.4948e	1	1	1	1	1	1
	last 7	3.4721e	1	1	1	0.9983	0.9983	0.9983
	1	0.3423	0.9529	0.9529	0.9529	1	1	1
	2	0.0261	0.9971	0.9971	0.9971	1	1	1
Lime	3	0.0195	0.9971	0.9971	0.9971	1	1	1
	4	0.0114	0.9979	0.9979	0.9979	1	1	1
	last 5	0.008	0.9986	0.9986	0.9986	0.9983	0.9983	0.9983
	1	1.0512	0.9143	0.9143	0.9143	0.9567	0.9567	0.9567
	3	0.1409	0.9829	0.9829	0.9829	0.9783	0.9783	0.9783
Orange	5	0.1657	0.9836	0.9836	0.9836	0.97	0.97	0.97
	7	0.0892	0.9936	0.9936	0.9936	0.9783	0.9783	0.9783
	last 8	0.1258	0.9893	0.9893	0.9893	0.9817	0.9817	0.9817
	1	0.9817	0.9621	0.9621	0.9621	1	1	1
	2	0.0000e	1	1	1	1	1	1
Pomegranate	3	0.0000e	1	1	1	1	1	1
	4	0.0000e	1	1	1	1	1	1
	last 5	3.4060e	1	1	1	1	1	1
	1	1.6609	0.8962	0.9002	0.8956	0.9794	0.9794	0.9794
	3	0.4233	0.9752	0.9755	0.9752	0.9425	0.9425	0.9425
Indian fruit all	5	0.3059	0.9846	0.9846	0.9846	0.9656	0.9656	0.9656
	7	0.2392	0.9893	0.9893	0.9893	0.9831	0.9831	0.9831
	last 8	0.2441	0.9888	0.9888	0.9888	0.9831	0.9872	0.9872

This table is showing a brief for all the models result for the Xception

	Epoch	loss	accuracy	precision	recall	val_accuracy	val_precision	val_recall
	1	0.0232	0.9979	0.9979	0.9979	0.9983	0.9983	0.9983
	2	1.9220e	1	1	1	0.9983	0.9983	0.9983
Apple	3	0.0114	0.9971	0.9971	0.9971	0.9983	0.9983	0.9983
	4	6.2746e	0.9993	0.9993	0.9993	0.9933	0.9933	0.9933
	last 5	0.0025	0.9993	0.9993	0.9993	0.9967	0.9967	0.9967
	1	0.0622	0.9864	0.9864	0.9864	0.9867	0.9867	0.9867
	2	0.0658	0.995	0.995	0.995	1	1	1
Banana	4	0.0612	0.9957	0.9957	0.9957	0.9983	0.9983	0.9983
	5	0.0102	0.9993	0.9993	0.9993	0.995	0.995	0.995
	last 6	0.0215	0.9971	0.9971	0.9971	0.9967	0.9967	0.9967
	1	0.3698	0.9514	0.9514	0.9514	1	1	1
	2	1.2713e	1	1	1	1	1	1
Guava	3	5.2172e	1	1	1	1	1	1
	4	5.3898e	1	1	1	1	1	1
	last 5	5.0493e	1	1	1	1	1	1
	1	0.311	0.9664	0.9664	0.9664	0.9917	0.9917	0.9917
	2	0.0499	0.9943	0.9943	0.9943	0.9967	0.9967	0.9967
Lime	4	0.0149	0.9986	0.9986	0.9986	0.9967	0.9967	0.9967
	5	0.0178	0.9986	0.9986	0.9986	0.9967	0.9967	0.9967
	last 6	0.0051	0.9993	0.9993	0.9993	0.995	0.995	0.995
	1	0.6007	0.9371	0.9371	0.9371	0.9683	0.9683	0.9683
	4	0.0737	0.9371	0.9371	0.9893	0.9683	0.9683	0.9683
Orange	7	0.0669	0.9371	0.9371	0.9929	0.9817	0.9817	0.9817
	10	0.0345	0.9371	0.9371	0.9936	0.99	0.99	0.99
	last 14	0.0236	0.9371	0.9371	0.9979	0.9683	0.9683	0.9683
	1	0.1481	0.9743	0.9743	0.9743	0.9983	0.9983	0.9983
	3	2.6927e	1	1	1	1	1	1
omegranat	5	1.1477e	1	1	1	1	1	1
	6	5.3465e	1	1	1	1	1	1
	last 7	8.5149e	1	1	1	1	1	1
	1	0.9632	0.9236	0.9269	0.9229	0.9625	0.9625	0.9625
	4	0.2163	0.9857	0.9857	0.9857	0.9861	0.9861	0.9861
dian fruit a	8	0.1909	0.992	0.992	0.992	0.9867	0.9867	0.9867
	11	0.1217	0.995	0.995	0.995	0.9906	0.9906	0.9906
	last 14	0.1581	0.9944	0.9944	0.9944	0.9708	0.9708	0.9708

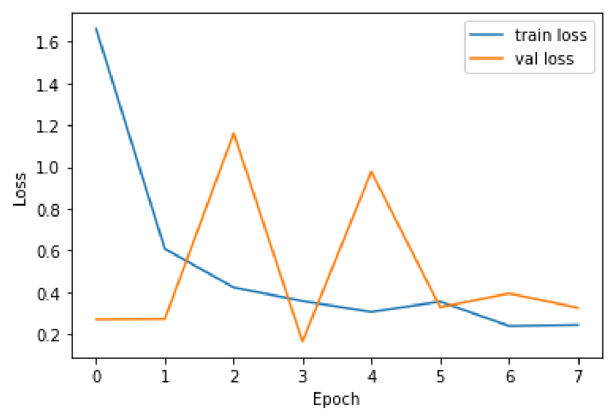
This table is showing a brief for all the models result for the VGG16

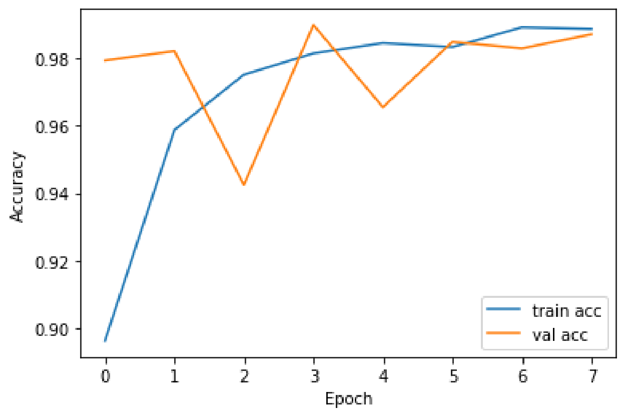
	Epoch	loss	accuracy	precision	recall	al_accuracy	al_precisio	val_recall
	1	0.3344	0.8643	0.8643	0.8643	0.985	0.985	0.985
	3	0.0198	0.9979	0.9979	0.9979	0.9967	0.9967	0.9967
Apple	6	0.0104	0.9986	0.9986	0.9986	0.9967	0.9967	0.9967
	9	0.0053	1	1	1	0.9983	0.9983	0.9983
	last 11	0.0056	0.9993	0.9993	0.9993	0.9983	0.9983	0.9983
	1	0.2478	0.8971	0.8971	0.8971	0.995	0.995	0.995
	3	0.0066	1	1	1	0.9967	0.9967	0.9967
Banana	6	0.0026	1	1	1	1	1	1
	7	0.0021	1	1	1	1	1	1
	last 8	0.002	1	1	1	1	1	1
	1	0.1243	0.9479	0.9479	0.9479	0.9983	0.9983	0.9983
	3	0.0046	0.9986	0.9986	0.9986	1	1	1
Guava	5	0.0022	1	1	1	1	1	1
	6	0.0016	1	1	1	1	1	1
	last 7	0.0014	0.9993	0.9993	0.9993	1	1	1
	1	0.3771	0.8586	0.8586	0.8586	0.9967	0.9967	0.9967
	3	0.0143	1	1	1	0.9983	0.9983	0.9983
Lime	5	0.0079	1	1	1	0.9983	0.9983	0.9983
	7	0.0055	1	1	1	1	1	1
	last 8	0.0051	1	1	1	1	1	1
	1	0.3398	0.8521	0.8521	0.8521	0.9533	0.9533	0.9533
	3	0.0635	0.9843	0.9843	0.9843	0.985	0.985	0.985
Orange	5	0.0451	0.9886	0.9886	0.9886	0.9833	0.9833	0.9833
	6	0.0387	0.9893	0.9893	0.9893	0.9833	0.9833	0.9833
	last 7	0.0582	0.98	0.98	0.98	0.9783	0.9783	0.9783
	1	0.1682	0.9421	0.9421	0.9421	1	1	1
	2	0.0035	1	1	1	1	1	1
Pomegranate	3	0.0025	1	1	1	1	1	1
	4	0.0015	1	1	1	1	1	1
	last 5	0.0013	1	1	1	1	1	1
	1	0.3838	0.8881	0.946	0.8464	0.9964	0.9613	0.9386
	3	0.0504	0.9885	0.9905	0.9856	0.9974	0.9686	0.9592
Indian fruit all	5	0.0248	0.9936	0.995	0.9921	0.9987	0.9883	0.985
	7	0.0189	0.9954	0.9961	0.9945	0.9981	0.9852	0.9831
	last 8	0.0284	0.991	0.992	0.9908	0.9989	0.99	0.9881

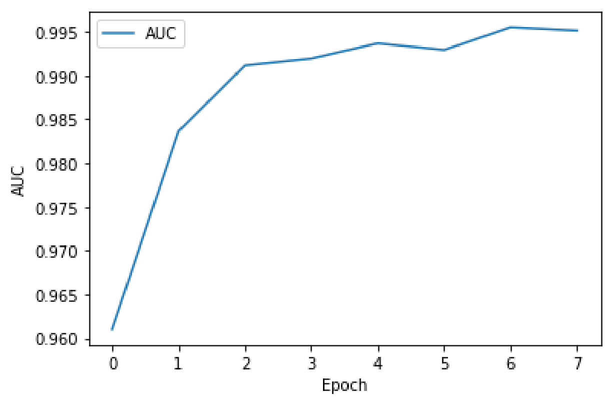
This table is showing a brief for all the models result for the VGG19

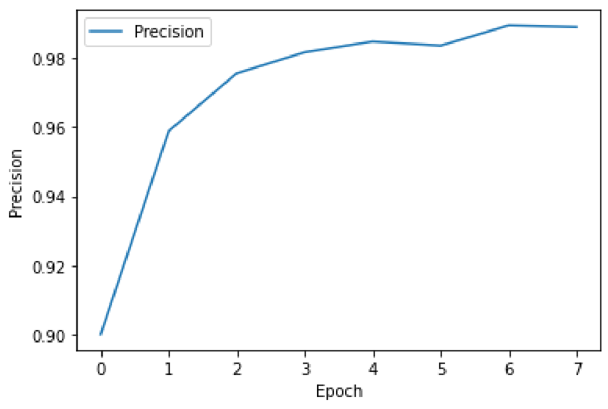
	Epoch	loss	accuracy	precision	recall	al_accurac	al_precisi	val_recall
	1	0.2206	0.9129	0.9129	0.9129	0.9817	0.9817	0.9817
	2	0.0469	0.9914	0.9914	0.9914	0.995	0.995	0.995
Apple	4	0.0247	0.9964	0.9964	0.9964	0.9867	0.9867	0.9867
	5	0.0168	0.9993	0.9993	0.9993	0.9933	0.9933	0.9933
	last 7	0.0088	0.9993	0.9993	0.9993	0.995	0.995	0.995
	1	0.2755	0.8914	0.8914	0.8914	0.9933	0.9933	0.9933
	2	0.0161	0.9964	0.9964	0.9964	0.9967	0.9967	0.9967
Banana	4	0.0067	1	1	1	0.9983	0.9983	0.9983
	5	0.0053	1	1	1	0.9983	0.9983	0.9983
	last 7	0.0039	1	1	1	0.9983	0.9983	0.9983
	1	0.2385	0.9157	0.9157	0.9157	0.995	0.995	0.995
	4	0.0058	0.9986	0.9986	0.9986	0.9983	0.9983	0.9983
Guava	7	0.0021	1	1	1	1	1	1
	9	0.0015	1	1	1	0.9983	0.9983	0.9983
	last 11	0.001	1	1	1	0.9983	0.9983	0.9983
	1	0.2268	0.92	0.92	0.92	0.995	0.995	0.995
	2	0.0289	0.9964	0.9964	0.9964	0.9867	0.9867	0.9867
Lime	4	0.0105	0.9993	0.9993	0.9993	1	1	1
	5	0.0072	1	1	1	1	1	1
	last 7	0.004	1	1	1	1	1	1
	1	0.5429	0.8007	0.8007	0.8007	0.9467	0.9467	0.9467
	4	0.0868	0.9764	0.9764	0.9764	0.9733	0.9733	0.9733
Orange	7	0.0425	0.9921	0.9921	0.9921	0.9833	0.9833	0.9833
	9	0.0272	0.9943	0.9943	0.9943	0.9817	0.9817	0.9817
	last 11	0.0243	0.9943	0.9943	0.9943	1	1	1
	1	0.2127	0.9129	0.9129	0.9129	1	1	1
	2	0.0047	1	1	1	1	1	1
omegranat	3	0.0039	0.9993	0.9993	0.9993	1	1	1
	4	0.0024	1	1	1	1	1	1
	last 5	0.0029	1	1	1	1	1	1
	1	0.5274	0.834	0.9131	0.781	0.9325	0.9469	0.9169
	4	0.0557	0.9848	0.9868	0.9823	0.9839	0.9863	0.9825
dian fruit a	7	0.0451	0.9869	0.9885	0.9857	0.9872	0.9875	0.9869
	9	0.033	0.99	0.9909	0.9894	0.9808	0.9822	0.9806
	last 11	0.0367	0.9871	0.9884	0.9864	0.9853	0.9861	0.9853

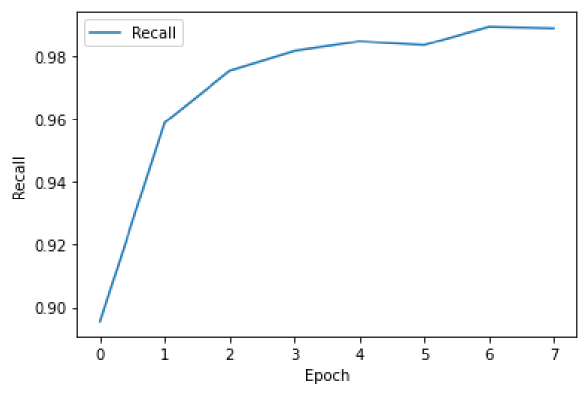
We will show the visualization for the all fruits together for Inceptionv3

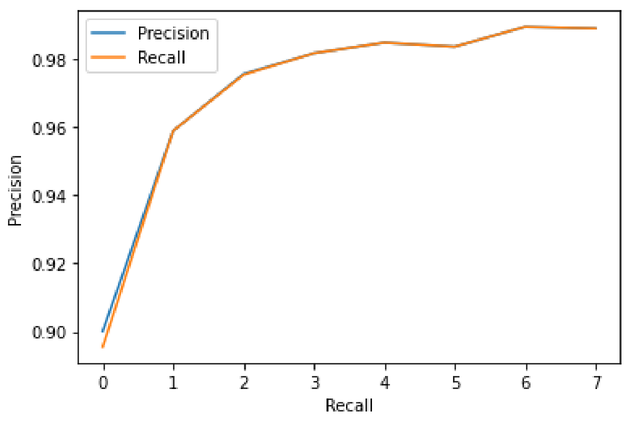




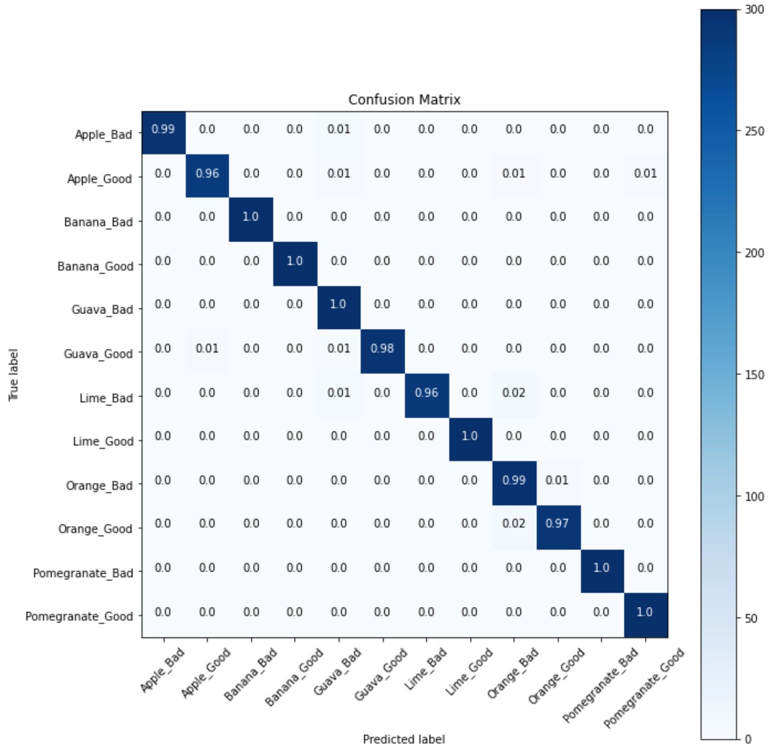






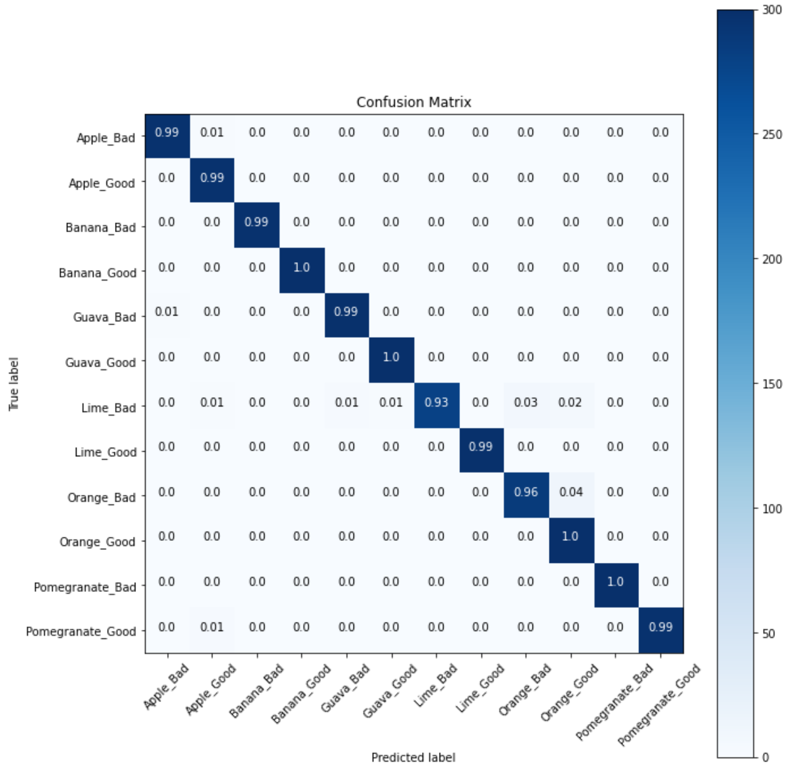


The confusion matrix for all fruits using Inceptionv3



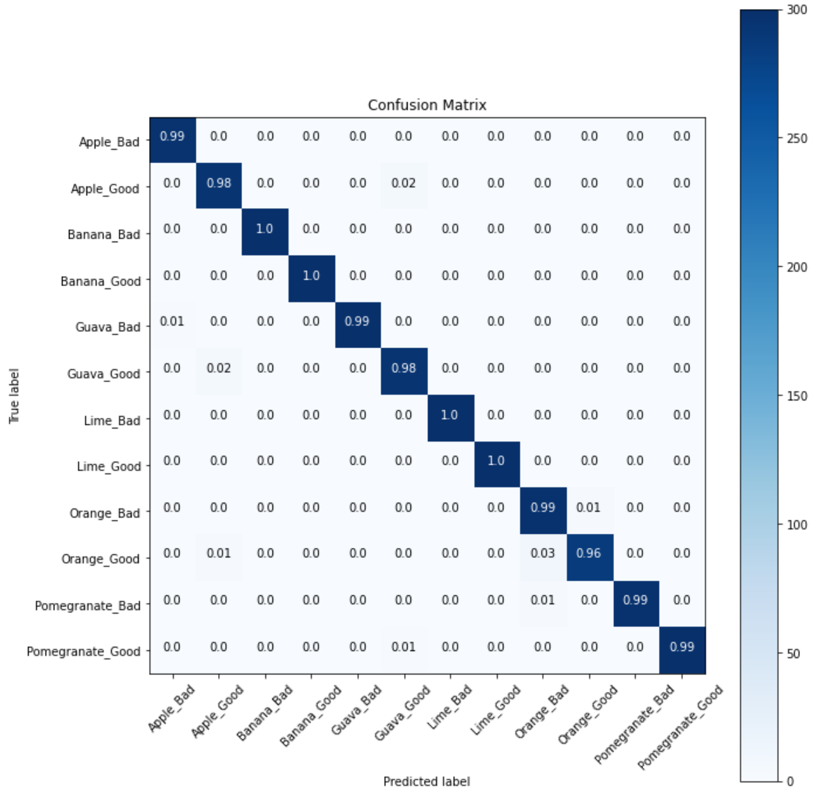
	Apple_Bad	Apple_Good	Banana_Bad	Banana_Good	Guava_Bad	Guava_Good	Lime_Bad	Lime_Good	Orange_Bad	Orange_Good	Pomegranate_Bad	Pomegranate_Good
Apple_Bad	0.99	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Apple_Good	0.0	0.96	0.0	0.0	0.01	0.0	0.0	0.0	0.01	0.0	0.0	0.01
Banana_Bad	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Banana_Good	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Guava_Bad	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Guava_Good	0.0	0.01	0.0	0.0	0.01	0.98	0.0	0.0	0.0	0.0	0.0	0.0
Lime_Bad	0.0	0.0	0.0	0.0	0.01	0.0	0.96	0.0	0.02	0.0	0.0	0.0
Lime_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Orange_Bad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.01	0.0	0.0
Orange_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.97	0.0	0.0
Pomegranate_Bad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Pomegranate_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

The confusion matrix for all fruits using Xception



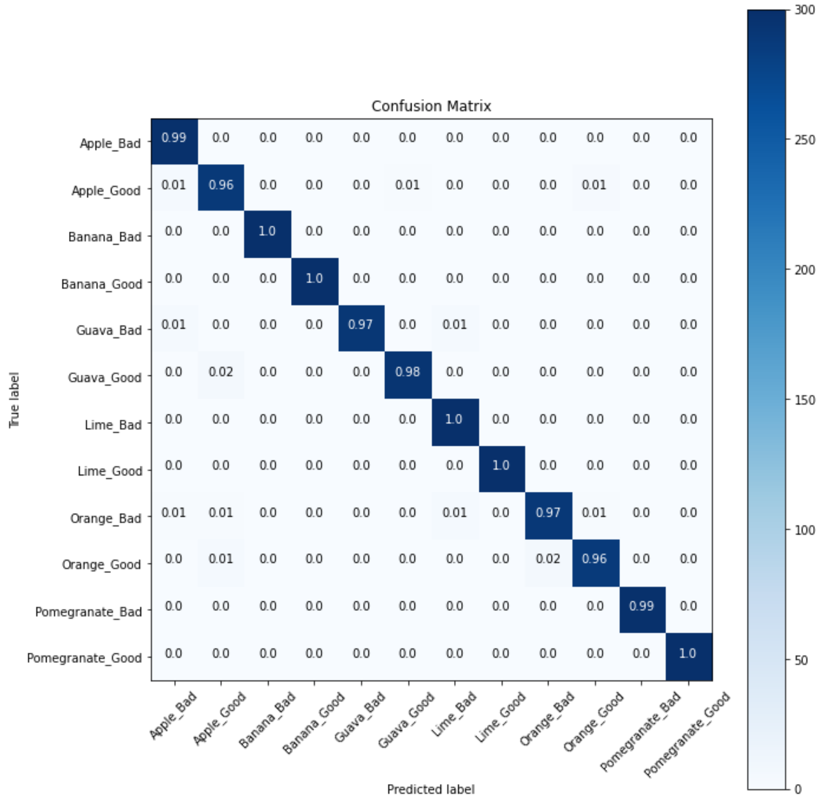
	Apple_Bad	Apple_Good	Banana_Bad	Banana_Good	Guava_Bad	Guava_Good	Lime_Bad	Lime_Good	Orange_Bad	Orange_Good	Pomegranate_Bad	Pomegranate_Good
Apple_Bad	0.99	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Apple_Good	0.0	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Banana_Bad	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Banana_Good	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Guava_Bad	0.01	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Guava_Good	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Lime_Bad	0.0	0.01	0.0	0.0	0.01	0.01	0.93	0.0	0.03	0.02	0.0	0.0
Lime_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0
Orange_Bad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.96	0.04	0.0	0.0
Orange_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
Pomegranate_Bad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Pomegranate_Good	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99

The confusion matrix for all fruits using VGG16



	Apple_Bad	Apple_Good	Banana_Bad	Banana_Good	Guava_Bad	Guava_Good	Lime_Bad	Lime_Good	Orange_Bad	Orange_Good	Pomegranate_Bad	Pomegranate_Good
Apple_Bad	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Apple_Good	0.0	0.98	0.0	0.0	0.0	0.02	0.0	0.0	0.0	0.0	0.0	0.0
Banana_Bad	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Banana_Good	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Guava_Bad	0.01	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Guava_Good	0.0	0.02	0.0	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.0	0.0
Lime_Bad	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Lime_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Orange_Bad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.01	0.0	0.0
Orange_Good	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.96	0.0	0.0
Pomegranate_Bad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.99	0.0
Pomegranate_Good	0.0	0.0	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.99

The confusion matrix for all fruits usingVGG19



	Apple_Bad	Apple_Good	Banana_Bad	Banana_Good	Guava_Bad	Guava_Good	Lime_Bad	Lime_Good	Orange_Bad	Orange_Good	Pomegranate_Bad	Pomegranate_Good
Apple_Bad	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Apple_Good	0.01	0.96	0.0	0.0	0.0	0.01	0.0	0.0	0.0	0.01	0.0	0.0
Banana_Bad	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Banana_Good	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Guava_Bad	0.01	0.0	0.0	0.0	0.97	0.0	0.01	0.0	0.0	0.0	0.0	0.0
Guava_Good	0.0	0.02	0.0	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.0	0.0
Lime_Bad	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Lime_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Orange_Bad	0.01	0.01	0.0	0.0	0.0	0.0	0.01	0.0	0.97	0.01	0.0	0.0
Orange_Good	0.0	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.02	0.96	0.0	0.0
Pomegranate_Bad	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0
Pomegranate_Good	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Conclusion and future work

We give an introduction and we described the dataset we worked with,

Then we went to Code description and wake throw in details with example to describe the recall, precision, and confusion matrix.

Showing the visualization results for some models (Apple , all fruits)

And some numerical results for all the models

future work:

modify the dataset used here to contain more fruits to be able to use the model in more areas of business

modify the code and use it on a medical dataset

References

[1]https://www.sciencedirect.com/science/article/pii/S2352340921009616

[2] https://doi.org/10.1016/j.dib.2019.104340

[3] https://doi.org/10.1016/j.dib.2022.108174

[4] https://doi.org/10.1016/j.measen.2022.100441

[5] https://doi.org/10.1016/j.atech.2022.100044

[6] https://doi.org/10.1016/j.inpa.2021.10.004

[7] https://doi.org/10.1016/S2095-3119(21)63604-3

[8] https://doi.org/10.1016/j.dib.2022.108043

[9] https://doi.org/10.1016/j.jksuci.2020.04.018

[10] https://doi.org/10.1016/j.fochms.2021.100056

[11] https://doi.org/10.1016/j.inpa.2019.10.003

[12] https://doi.org/10.1016/j.compag.2022.107206

[13] https://doi.org/10.1016/j.compag.2020.105842

[14] https://doi.org/10.1016/j.jksuci.2022.11.003

[15] https://doi.org/10.1016/j.gltp.2021.08.002

[16] https://doi.org/10.1016/j.jksuci.2018.06.006

[17] https://www.javatpoint.com/precision-and-recall-in-machine-learning