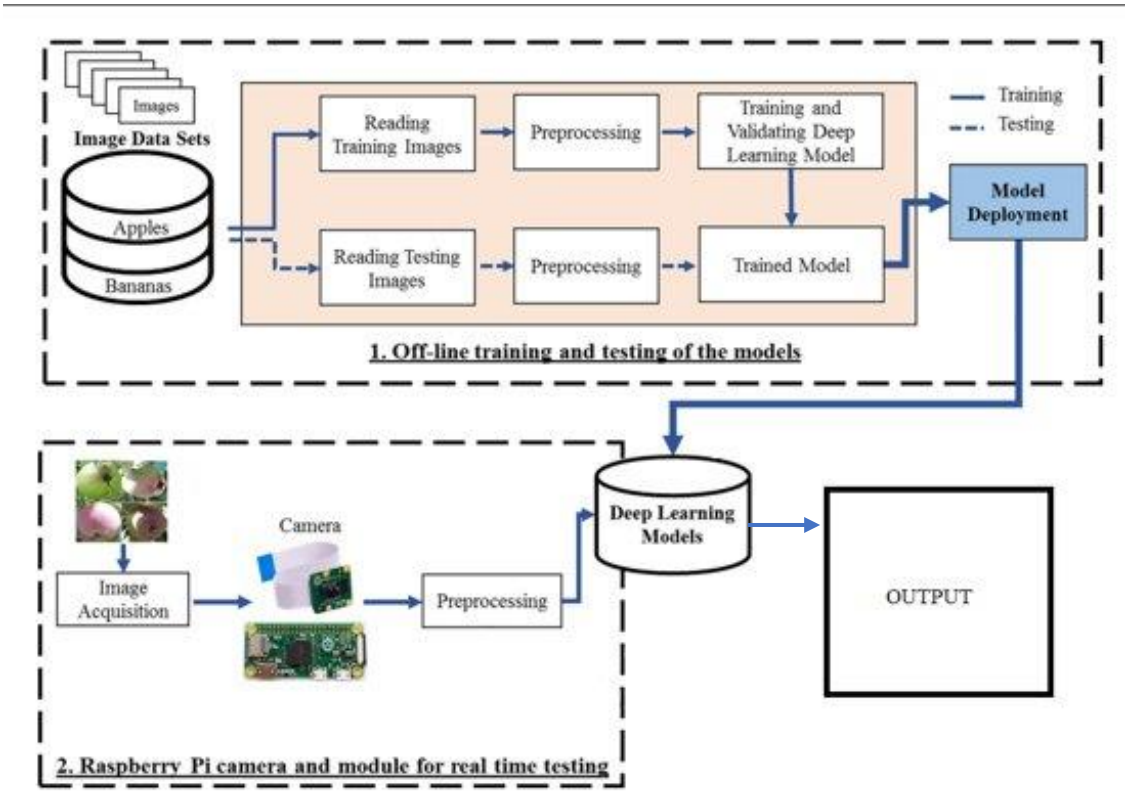
# INTRODUCTION

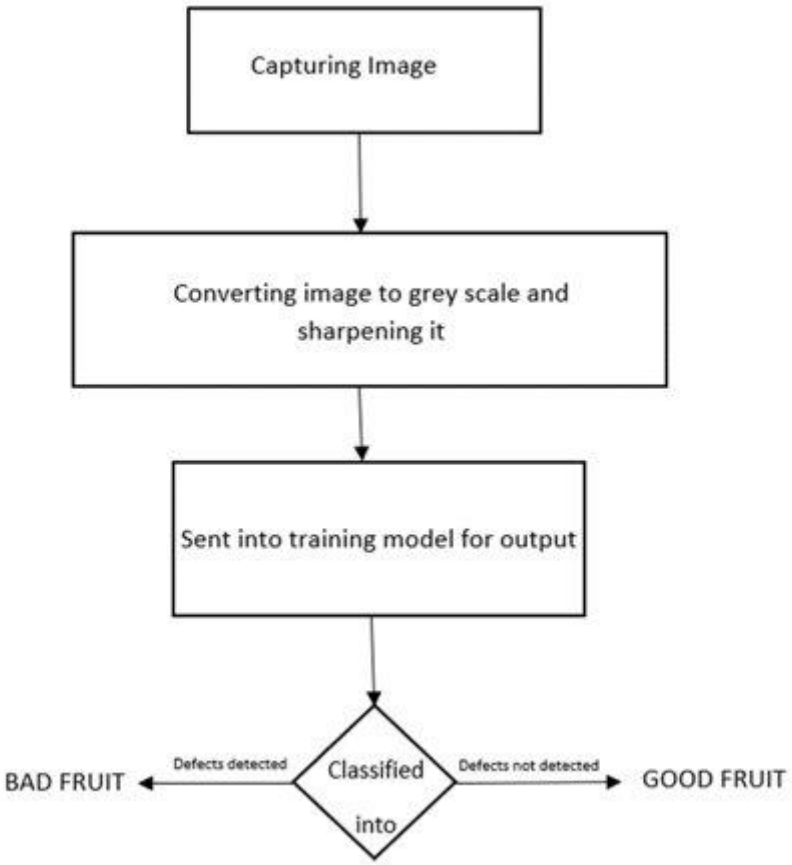
India is an agriculture country. Different types of fruits and vegetables are produced where all the pre-harvest and post-harvest processes are done manually with the help of labor. Separating fruits of lower quality from fresh fruits manually can become a tedious task when done on a larger scale. It takes a lot of man-power to segregate the fruits into batches based on quality  before they can be packaged and shipped. One of the most common ways in which segregation is done is by identifying spots on the fruit. Most of the fruits have a tendency to develop brown (or) black spots on the surface as a result of rotting from the inside. Our project aims at developing a model that can classify a fruit and label it as defective, on the identification of any spots on its surface using a machine learning approach. This reduces the need for human interference in the process.

# DESIGN

## PROPOSED MODEL



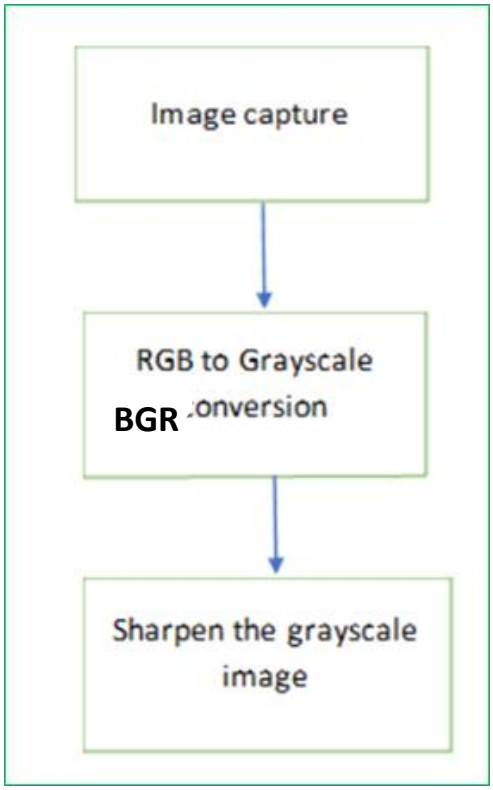
The project consists of 3 stages. In the first stage, a deep learning model is designed and is trained and tested on the static dataset consisting of 8120 images of both good and bad quality of fruits of the type Apple, Banana and Guava. In the second stage, an image of the fruit is captured by the Raspberry Pi camera connected to the Raspberry Pi Zero 2 W and is uploaded to the google drive. In the final stage, the uploaded image is taken as input to the deep learning model and the output class is predicted.



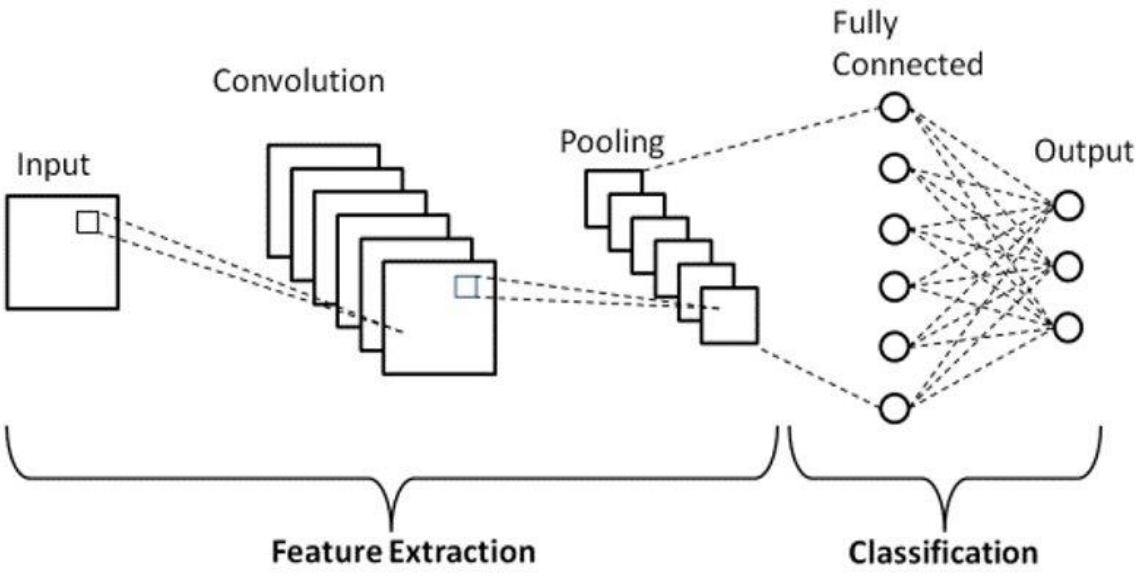

The dataset used is the FruitNet dataset consisting of 6 types of fruits (apple, Banana, Guava, Lime, Pomegranate and Orange) and 3 quality types per fruit (good, mixed, bad).

The pre-processing of the data is carried out in two stages.

1) ***Data Augmentation***: The dataset was highly imbalanced and had to be balanced to avoid overfitting or underfitting of the model.
2) ***Image-processing***: The images are first converted from BGR to Gray scale and then sharpened using a kernel ( [ [0,-1,0] , [0,4,0] , [0,-1,0] ] ) before the images are fed into the neural network. This is done to identify the black/brown spots in the bad quality fruits.



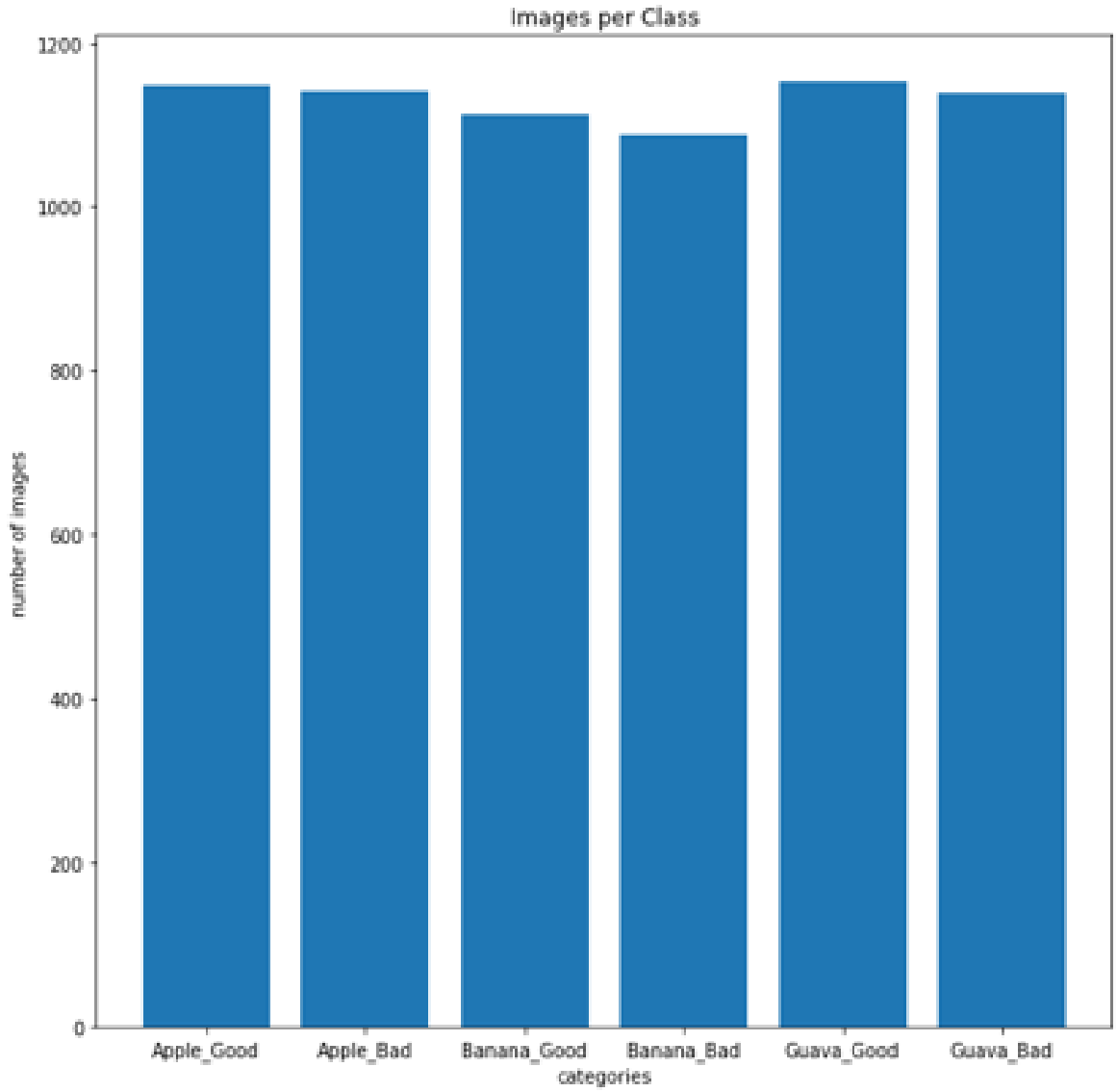

The deep learning model used for the problem is a Convolution Neural Network with 3 convolution layers, 2 pooling layers and 1 dropout layer. The dataset has been split as 80% training data and 20% testing data.




The model achieved an accuracy of 94% after training for 10 epochs.

# RESULTS

VISUALIZATION OF THE BALANCED DATASET:

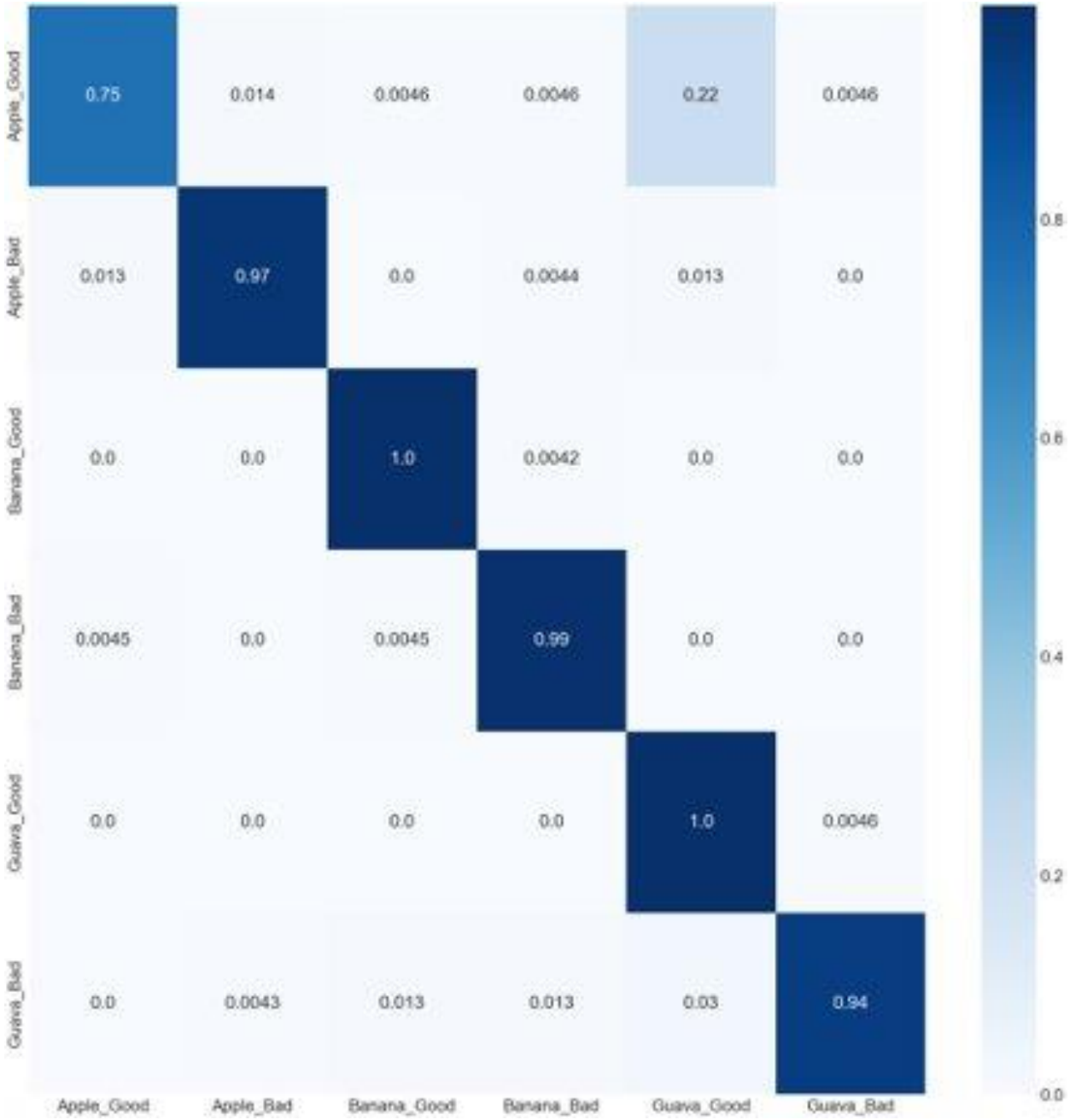



CLASSIFICATION REPORT:

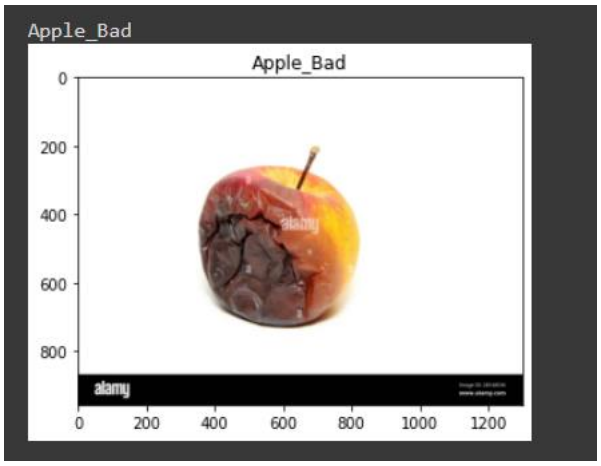
**** Classification Report ****

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Apple_Good | 0.98 | 0.75 | 0.85 | 219 |
| Apple_Bad | 0.98 | 0.97 | 0.98 | 228 |
| Banana_Good | 0.98 | 1.00 | 0.99 | 237 |
| Banana_Bad | 0.97 | 0.99 | 0.98 | 222 |
| Guava_Good | 0.79 | 1.00 | 0.88 | 217 |
| Guava_Bad | 0.99 | 0.94 | 0.96 | 232 |
| accuracy | | | 0.94 | 1355 |
| macro avg | 0.95 | 0.94 | 0.94 | 1355 |
| weighted avg | 0.95 | 0.94 | 0.94 | 1355 |

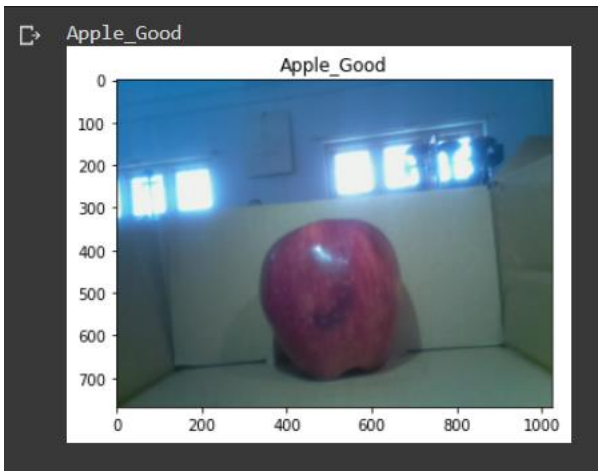
CONFUSION MATRIX:



Testing the model with image:




Testing the model with image taken by Pi Camera:




# FUTURE SCOPE

- A setup that can revolve the camera to capture the images of the fruit from all direction to provide better results.
- The use of a bounding box that can detect individual fruits in an image of multiple fruits and classify them individually.
- Automatic segregation of fruits with the help of actuators and motors.

# REFERENCES


- FruitNet: Indian fruits image dataset with quality for machine learning applications (By Vishal Meshram , Kailas Patil Vishwakarma University, India)
- Automatic Fruit Quality Inspection System (By Manali R. Satpute, Sumati M. Jagadle)

- A Real-Time Smart Fruit Quality Grading System Classifying by External Appearance and Internal Flavor Factors (By Han Suk Choi, Je Bong Cho, Sang Gyun Kim, Hong Seok Choi)
- Real-time visual inspection system for grading fruits using computer vision and deep learning techniques, Information Processing in Agriculture (By N. Ismail and O. A. Malik)

# APPENDIX

Components:

1. Raspberry Pi Zero 2 W
2. Pi Camera 5MP
3. Pi Cable
4. SD Card 8GB (for loading the raspberry pi OS)

Software:

1. Google Colab
2. PuTTY
3. VNC Viewer

CODE:

```
[ ]  import os
     os.environ["TF_GPU_ALLOCATOR"]="cuda_malloc_async"

     import tensorflow as tf
     gpus = tf.config.experimental.list_physical_devices('GPU')
     tf.config.experimental.set_memory_growth(gpus[0], True)
```

```
[ ]  from tensorflow.keras.layers import Dense, Activation, Dropout, Flatten
     from tensorflow.keras.layers import Convolution2D, MaxPooling2D
     from tensorflow.keras.optimizers import SGD, RMSprop, Adam
     from keras.utils import np_utils
     from sklearn import metrics
     from sklearn.utils import shuffle
     from sklearn.model_selection import train_test_split
     import matplotlib.image as mpimg
     import matplotlib.pyplot as plt
     import numpy as np
     import os
     import cv2
     import random
     from numpy import *
     from PIL import Image
```

```
[ ] data_path = "./FruitNetDataset/Dataset/"
```

```
[ ] CATEGORIES = ["Apple_Good", "Apple_Bad", "Banana_Good", "Banana_Bad","Guava_Good","Guava_Bad"]
```

```
training = []

def createTrainingData():
    for category in CATEGORIES:
        path = data_path+category
        print(path)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            img_array = cv2.imread(os.path.join(path,img))
            new_array=cv2.resize(img_array,(200,200))
            training.append([new_array, class_num])

            for a in training:
               gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
               sharpen = np.array([[0,-1,0],[0,4,0],[0,-1,0]])
               img = cv2.filter2D(src=gray, kernel=sharpen, ddepth=-1)
```

```
[ ] createTrainingData()
```

```
./FruitNetDataset/Dataset/Apple_Good
./FruitNetDataset/Dataset/Apple_Bad
./FruitNetDataset/Dataset/Banana_Good
./FruitNetDataset/Dataset/Banana_Bad
./FruitNetDataset/Dataset/Guava_Good
./FruitNetDataset/Dataset/Guava_Bad
```

```
[ ] len(training)
```

```
6771
```

```
[ ] random.shuffle(training)
```

```
[ ] X =[]
    y =[]
    for features, label in training:
        X.append(features)
        y.append(label)
    X = np.array(X).reshape(-1, 200,200 , 3)
```

```
[ ] max(y)
```

```
5
```

```
import matplotlib.pyplot as plt

x=["Apple_Good", "Apple_Bad", "Banana_Good", "Banana_Bad","Guava_Good","Guava_Bad"]

# giving the values against
# each value at x axis
y=[Apple_Good, Apple_Bad, Banana_Good, Banana_Bad,Guava_Good,Guava_Bad ]

plt.figure(figsize=(10,10))
plt.bar(x, y)

# setting x-label as pen sold
plt.xlabel("categories")

# setting y_label as price
plt.ylabel("number of images")
plt.title("Images per Class")
plt.show()
```

```
[ ] X = X.astype('float32')
    X /= 255
    from keras.utils import np_utils
    Y = np_utils.to_categorical(y,6)
    print(Y[100])
    print(shape(Y))
```

```
[0. 0. 0. 0. 1. 0.]
(6771, 6)
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 4)
```

```
[ ] y_train=np.array(y_train)
    y_test=np.array(y_test)
```

```
[ ] print(len(X_test),len(y_test))
```

```
1355 1355
```

```
[ ] batch_size = 1
    nb_classes =6
    nb_epochs = 15
    img_rows, img_columns = 200, 200
    img_channel = 3
    nb_filters = 32
    nb_pool = 2
    nb_conv = 3
```

```
[ ] import tensorflow as tf
    tf.config.list_physical_devices(
        device_type=None
    )
```

```
[PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'),
 PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
from keras import backend as K
K.clear_session()
model = tf.keras.Sequential([
        tf.keras.layers.Conv2D(32, (3,3), padding='same', activation=tf.nn.relu,
                               input_shape=(200, 200, 3)),
        tf.keras.layers.MaxPooling2D((2, 2), strides=2),
        tf.keras.layers.Conv2D(32, (3,3), padding='same', activation=tf.nn.relu),
        tf.keras.layers.MaxPooling2D((2, 2), strides=2),
        tf.keras.layers.Conv2D(16, (3,3), padding='same', activation=tf.nn.relu),
        tf.keras.layers.MaxPooling2D((2, 2), strides=2),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation=tf.nn.relu),
        tf.keras.layers.Dense(6,  activation=tf.nn.softmax)
    ])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 200, 200, 32)      896

 max_pooling2d (MaxPooling2D  (None, 100, 100, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 100, 100, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 50, 50, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 50, 50, 16)        4624

 max_pooling2d_2 (MaxPooling  (None, 25, 25, 16)       0
 2D)

 dropout (Dropout)           (None, 25, 25, 16)        0

 flatten (Flatten)           (None, 10000)             0

 dense (Dense)               (None, 128)               1280128

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 1,295,670
Trainable params: 1,295,670
Non-trainable params: 0
_________________________________________________________________
```

```
my_callbacks = [
    tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1,patience=3),
    tf.keras.callbacks.ModelCheckpoint(filepath='model.{epoch:02d}-{val_loss:.2f}.h5'),
    tf.keras.callbacks.TensorBoard(log_dir='./logs'),
]
```

```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(X_train, y_train, batch_size = batch_size, epochs = nb_epochs, callbacks=my_callbacks,verbose = 1, validation_data = (X_test, y_test))
```

```
Epoch 1/15
5416/5416 [==============================] - 40s 7ms/step - loss: 1.2624 - accuracy: 0.5085 - val_loss: 0.4959 - val_accuracy: 0.8244
Epoch 2/15
5416/5416 [==============================] - 33s 6ms/step - loss: 0.4414 - accuracy: 0.8517 - val_loss: 0.3174 - val_accuracy: 0.8834
Epoch 3/15
5416/5416 [==============================] - 33s 6ms/step - loss: 0.2639 - accuracy: 0.9106 - val_loss: 0.2044 - val_accuracy: 0.9225
Epoch 4/15
5416/5416 [==============================] - 34s 6ms/step - loss: 0.1812 - accuracy: 0.9431 - val_loss: 0.1819 - val_accuracy: 0.9469
Epoch 5/15
5416/5416 [==============================] - 34s 6ms/step - loss: 0.1441 - accuracy: 0.9553 - val_loss: 0.2407 - val_accuracy: 0.9314
Epoch 6/15
5416/5416 [==============================] - 36s 7ms/step - loss: 0.1056 - accuracy: 0.9653 - val_loss: 0.2588 - val_accuracy: 0.9299
Epoch 7/15
5416/5416 [==============================] - 39s 7ms/step - loss: 0.0959 - accuracy: 0.9723 - val_loss: 0.1555 - val_accuracy: 0.9491
Epoch 8/15
5416/5416 [==============================] - 80s 15ms/step - loss: 0.0841 - accuracy: 0.9734 - val_loss: 0.2934 - val_accuracy: 0.9321
Epoch 9/15
5416/5416 [==============================] - 74s 14ms/step - loss: 0.0984 - accuracy: 0.9716 - val_loss: 0.2114 - val_accuracy: 0.9432
Epoch 10/15
5416/5416 [==============================] - 35s 6ms/step - loss: 0.0773 - accuracy: 0.9771 - val_loss: 0.2906 - val_accuracy: 0.9446
Epoch 10: early stopping
<keras.callbacks.History at 0x18b2831ca90>
```

```
def predict(path):

    test_img=cv2.imread(path)
    #print(test_image.shape)
    test_image=cv2.resize(test_img,(200,200))
    #print(test_image.shape)
    test_image = np.array(test_image).reshape(-1, 200,200 , 3)
    test_image = test_image.astype('float32')
    test_image /= 255
    predicted_label=model.predict(test_image)
    pred=list(predicted_label[0])
    max_conf=max(pred)
    pred_label=CATEGORIES[pred.index(max_conf)]
    plt.title(pred_label)
    plt.imshow(test_img)
```

```
predict("./apple_test.jpg")
```

## Load a saved weight file and infer

```
import keras
from keras.models import load_model
model_infer=load_model('/content/drive/MyDrive/model.04-0.55.h5')
def saved_model_predict(path):

    test_img=cv2.imread(path)
    #print(test_image.shape)
    test_image=cv2.resize(test_img,(200,200))
    #print(test_image.shape)
    test_image = np.array(test_image).reshape(-1, 200,200 , 3)
    test_image = test_image.astype('float32')
    test_image /= 255
    predicted_label=model_infer.predict(test_image)
    pred=list(predicted_label[0])
    max_conf=max(pred)
    pred_label=CATEGORIES[pred.index(max_conf)]
    print(pred_label)
    plt.title(pred_label)
    plt.imshow(cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB))
```

```
[ ] saved_model_predict('/content/drive/MyDrive/test iages/badddd appi.jpg')
```

```
[ ] def load_images_from_folder(folder):
        images = []
        i=0
        for filename in os.listdir(folder):
            pth = os.path.join(folder,filename)
            if pth is not None:
                saved_model_predict(pth)
                os.remove(pth)

    load_images_from_folder('/content/drive/MyDrive/input_rpi')
```

```
y_pred=model_infer.predict(X_test)
y_pred_labels = np.argmax(y_pred, axis=1)
from sklearn.metrics import multilabel_confusion_matrix
mcm = multilabel_confusion_matrix(y_test, y_pred_labels, labels = [0,1,2,3,4,5])
print(mcm)
from sklearn.metrics import confusion_matrix, classification_report
from matplotlib import pyplot as plt
import seaborn as sns

def plot_confusion_matrix(y_test,y_scores, classNames):
    classes = len(classNames)
    cm = confusion_matrix(y_test, y_scores)
    print("**** Confusion Matrix ****")
    print(cm)
    print("**** Classification Report ****")
    print(classification_report(y_test, y_scores, target_names=classNames))
    con = np.zeros((classes,classes))
    for x in range(classes):
        for y in range(classes):
            con[x,y] = cm[x,y]/np.sum(cm[x,:])

    plt.figure(figsize=(40,40))
    sns.set(font_scale=3.0) # for label size
    df = sns.heatmap(con, annot=True,fmt='.2', cmap='Blues',xticklabels= classNames , yticklabels= classNames)
    df.figure.savefig("image2.png")


plot_confusion_matrix(y_test,y_pred_labels, CATEGORIES)
```