```
!unzip '/content/Dataset.zip'
```

```
  inflating: Dataset/TRAIN_SET/WATERMELON/r_269_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_26_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_270_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_271_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_272_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_273_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_274_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_275_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_276_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_277_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_278_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_279_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_27_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_280_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_281_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_282_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_283_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_284_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_285_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_286_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_287_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_288_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_289_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_28_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_290_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_291_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_292_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_293_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_294_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_295_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_296_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_297_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_298_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_299_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_29_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_2_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_300_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_301_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_302_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_303_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_304_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_305_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_306_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_307_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_308_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_309_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_30_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_310_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_311_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_312_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_313_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_314_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_315_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_31_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_32_100.jpg
  inflating: Dataset/TRAIN_SET/WATERMELON/r_33_100.jpg
```

```
          inflating: Dataset/TRAIN_SET/WATERMELON/r_34_100.jpg
          inflating: Dataset/TRAIN_SET/WATERMELON/r_35_100.jpg
```

```
from keras.preprocessing.image import ImageDataGenerator


train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizont
test_datagen = ImageDataGenerator (rescale=1./255)


x_train = train_datagen.flow_from_directory(
r'/content/Dataset/TRAIN_SET',
target_size=(64, 64), batch_size=5, color_mode='rgb', class_mode='sparse')
x_test = test_datagen.flow_from_directory(
r'/content/Dataset/TEST_SET',
target_size=(64, 64), batch_size=5,color_mode='rgb', class_mode='sparse' )
```

```
     Found 4138 images belonging to 5 classes.
     Found 929 images belonging to 3 classes.
```

```
import numpy as np#used for numerical analysis
import tensorflow #open source used for both ML and DL for computation
from tensorflow.keras.models import Sequential #it is a plain stack of layers
from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computatic
#Dense layer is the regular deeply connected neural network layer
from tensorflow.keras.layers import Dense,Flatten
#Faltten-used fot flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dropout #Convolutional layer
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator


#setting parameter for Image Data agumentation to the training data
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizonta
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)


#performing data agumentation to train data
x_train = train_datagen.flow_from_directory(
    r'/content/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
#performing data agumentation to test data
x_test = test_datagen.flow_from_directory(
    r'/content/Dataset/TEST_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
```

```
     Found 4138 images belonging to 5 classes.
     Found 929 images belonging to 3 classes.
```

```
print(x_train.class_indices)
```

```
     {'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}
```

```
print(x_test.class_indices)
```

```
{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2}
```

```
from collections import Counter as c
c(x_train .labels)
```

```
Counter({0: 995, 1: 1374, 2: 1019, 3: 275, 4: 475})
```

```
# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Second convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), activation='relu'))

# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())


# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=5, activation='softmax')) # softmax for more than 2
```

```
classifier.summary()#summary of our model
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 5)                 645

=================================================================
Total params: 813,733
Trainable params: 813,733
```

```
        Non-trainable params: 0

    # Compiling the CNN
    # categorical_crossentropy for more than 2
    classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc


    classifier.fit_generator(
            generator=x_train,steps_per_epoch = len(x_train),
            epochs=20, validation_data=x_test,validation_steps = len(x_test))# No of images ir

         Epoch 1/20
         /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.
           This is separate from the ipykernel package so we can avoid doing imports until
         828/828 [==============================] - 22s 16ms/step - loss: 0.5852 - accuracy:
         Epoch 2/20
         828/828 [==============================] - 13s 16ms/step - loss: 0.4218 - accuracy:
         Epoch 3/20
         828/828 [==============================] - 13s 15ms/step - loss: 0.3603 - accuracy:
         Epoch 4/20
         828/828 [==============================] - 13s 15ms/step - loss: 0.3483 - accuracy:
         Epoch 5/20
         828/828 [==============================] - 13s 15ms/step - loss: 0.3270 - accuracy:
         Epoch 6/20
         828/828 [==============================] - 13s 15ms/step - loss: 0.3124 - accuracy:
         Epoch 7/20
         828/828 [==============================] - 14s 17ms/step - loss: 0.2995 - accuracy:
         Epoch 8/20
         828/828 [==============================] - 13s 15ms/step - loss: 0.2897 - accuracy:
         Epoch 9/20
         828/828 [==============================] - 14s 16ms/step - loss: 0.2716 - accuracy:
         Epoch 10/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.2407 - accuracy:
         Epoch 11/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.2548 - accuracy:
         Epoch 12/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.2269 - accuracy:
         Epoch 13/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.1969 - accuracy:
         Epoch 14/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.2018 - accuracy:
         Epoch 15/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.1783 - accuracy:
         Epoch 16/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.1800 - accuracy:
         Epoch 17/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.1529 - accuracy:
         Epoch 18/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.1516 - accuracy:
         Epoch 19/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.1363 - accuracy:
         Epoch 20/20
         828/828 [==============================] - 12s 15ms/step - loss: 0.1310 - accuracy:
         <keras.callbacks.History at 0x7fa62951fd50>

    # Save the model
    classifier.save('nutrition.h5')
```

```
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("nutrition.h5") #loading the model for testing
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
from tensorflow.keras.utils import load_img, img_to_array
img = load_img(r"/content/Dataset/TEST_SET/BANANA/0SF1HXERAMSH.jpg",
grayscale=False, target_size= (64,64)) #Loading of the image image.img_to_array(img)#image
x = img_to_array(img)
```

```
x
```

```
array([[[207., 209., 169.],
        [209., 211., 171.],
        [212., 214., 175.],
        ...,
        [178., 176., 101.],
        [174., 171.,  92.],
        [168., 165.,  86.]],

       [[212., 214., 174.],
        [214., 216., 176.],
        [216., 218., 179.],
        ...,
        [183., 178., 110.],
        [178., 174., 103.],
        [173., 169.,  96.]],

       [[216., 218., 178.],
        [218., 220., 180.],
        [220., 222., 183.],
        ...,
        [189., 183., 123.],
        [187., 179., 117.],
        [182., 174., 111.]],

       ...,

       [[ 33.,  23.,  13.],
        [ 46.,  37.,  22.],
        [ 60.,  49.,  27.],
        ...,
        [ 28.,  20.,   7.],
        [ 23.,  14.,   9.],
        [ 18.,   6.,  10.]],

       [[ 35.,  25.,  13.],
        [ 51.,  42.,  27.],
        [ 58.,  47.,  29.],
        ...,
        [ 30.,  24.,  12.],
        [ 19.,  15.,   6.],
```

```
                [ 10.,   6.,   3.]],

               [[ 45.,  36.,  19.],
                [ 48.,  39.,  22.],
                [ 40.,  31.,  16.],
                ...,
                [ 20.,  10.,  11.],
                [  9.,   6.,   1.],
                [  8.,  10.,   0.]]], dtype=float32)
```

```
x.ndim
x = np.expand_dims(img,axis = 0)
x.ndim
pred = classifier.predict(x)
analyses =np.argmax(pred,axis=1)
analyses
```

```
1/1 [==============================] - 0s 131ms/step
array([1])
```

```
labels=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
labels[np.argmax(pred)]
```

```
'BANANA'
```