Import the ImageDataGenerator library

```
from keras.preprocessing.image import ImageDataGenerator
```

Configure ImageDataGenerator Class

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

Apply Image DataGenerator Functionality To Trainset And Testset

```
x_train = train_datagen.flow_from_directory(
    r'/content/drive/MyDrive/Project/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=32,color_mode='rgb',class_mode='categorical')
x_test = test_datagen.flow_from_directory(
    r'/content/drive/MyDrive/Project/Dataset/TEST_SET',
    target_size=(64, 64),batch_size=32,color_mode='rgb',class_mode='categorical')
```

```
 Found 3838 images belonging to 5 classes.
 Found 280 images belonging to 5 classes.
```

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

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

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

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

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

```
 Counter({0: 913, 1: 1306, 2: 964, 3: 240, 4: 415})
```

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

```
 Counter({0: 82, 1: 48, 2: 55, 3: 35, 4: 60})
```

Importing The Model Building Libraries

```
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D,Dropout
from keras.preprocessing.image import ImageDataGenerator
```

Initializing The Model

```
model=Sequential()
```

Adding CNN Layers

```
classifier = Sequential()
classifier.add(Conv2D(32,(3, 3), input_shape=(64, 64, 3),activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Conv2D(32, (3, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Flatten())
```

Adding Dense Layers

```
classifier.add(Dense (units=128, activation='relu'))
classifier.add(Dense (units=5, activation='softmax'))
```

```
classifier.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 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
_________________________________________________________________
```

Configure The Learning Process

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

Train The Model

```
classifier.fit_generator(
    generator=x_train,steps_per_epoch = len(x_train),
    epochs=20,validation_data=x_test,validation_steps = len(x_test))
```

```
 /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_
   This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/20
120/120 [==============================] - 1251s 10s/step - loss: 0.8456 - accuracy: 0.6
Epoch 2/20
120/120 [==============================] - 30s 249ms/step - loss: 0.5028 - accuracy: 0.8
Epoch 3/20
120/120 [==============================] - 32s 262ms/step - loss: 0.4244 - accuracy: 0.8
Epoch 4/20
120/120 [==============================] - 32s 262ms/step - loss: 0.3783 - accuracy: 0.8
Epoch 5/20
120/120 [==============================] - 30s 246ms/step - loss: 0.3574 - accuracy: 0.8
Epoch 6/20
120/120 [==============================] - 30s 248ms/step - loss: 0.3395 - accuracy: 0.8
Epoch 7/20
120/120 [==============================] - 31s 259ms/step - loss: 0.3226 - accuracy: 0.8
Epoch 8/20
120/120 [==============================] - 31s 261ms/step - loss: 0.2843 - accuracy: 0.8
Epoch 9/20
120/120 [==============================] - 32s 265ms/step - loss: 0.2841 - accuracy: 0.8
Epoch 10/20
120/120 [==============================] - 29s 243ms/step - loss: 0.2765 - accuracy: 0.8
Epoch 11/20
120/120 [==============================] - 30s 246ms/step - loss: 0.2471 - accuracy: 0.9
Epoch 12/20
120/120 [==============================] - 32s 262ms/step - loss: 0.2460 - accuracy: 0.9
Epoch 13/20
120/120 [==============================] - 31s 259ms/step - loss: 0.2415 - accuracy: 0.9
Epoch 14/20
120/120 [==============================] - 29s 244ms/step - loss: 0.2496 - accuracy: 0.9
Epoch 15/20
120/120 [==============================] - 32s 264ms/step - loss: 0.2184 - accuracy: 0.9
```

```
Epoch 16/20
120/120 [==============================] - 32s 264ms/step - loss: 0.2170 - accuracy: 0.9
Epoch 17/20
120/120 [==============================] - 30s 245ms/step - loss: 0.2262 - accuracy: 0.9
Epoch 18/20
120/120 [==============================] - 32s 266ms/step - loss: 0.1966 - accuracy: 0.9
Epoch 19/20
120/120 [==============================] - 32s 265ms/step - loss: 0.1762 - accuracy: 0.9
Epoch 20/20
120/120 [==============================] - 30s 246ms/step - loss: 0.1792 - accuracy: 0.9
<keras.callbacks.History at 0x7f6e00cf07d0>
```

Save The Model

```
classifier.save('nutrition.h5')
```

Test The Model

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
model = load_model("nutrition.h5")
```

```
img = image.load_img('/content/drive/MyDrive/Project/Dataset/TEST_SET/ORANGE/n07749192_122.jp
                     target_size=(64,64))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
pred = np.argmax(model.predict(x))
op = ['APPLE', 'BANANA', 'ORANGE', 'PINAPPLE', 'WATERMELON']
op[pred]
```

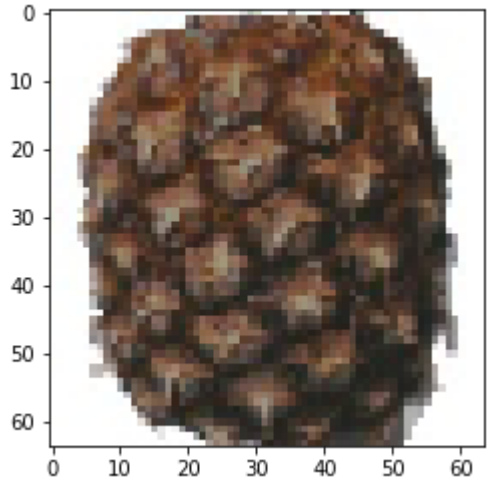
```
1/1 [==============================] - 0s 152ms/step
'ORANGE'
```

```
img
```

```
classes=['APPLE', 'BANANA', 'ORANGE', 'PINAPPLE', 'WATERMELON']
def testing(img):
  img=image.load_img(img,target_size=(64,64))
  x=image.img_to_array(img)
  x=np.expand_dims(x,axis=0)
```

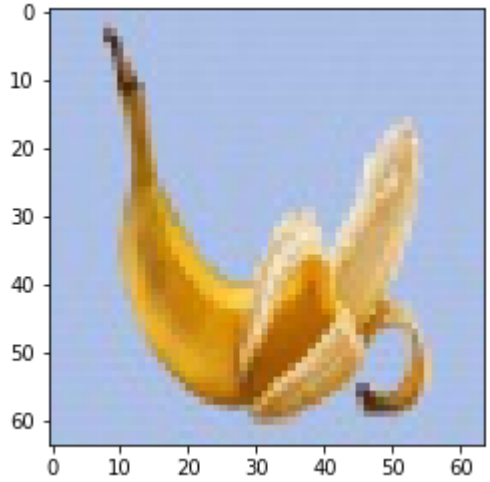
```
  pred=np.argmax(model.predict(x))
  return print("Predicted class as:",classes[pred])

def img_show(img):
  img1=image.load_img(img,target_size=(64,64))
  plt.imshow(img1)


#test1
img_show('/content/drive/MyDrive/Project/Dataset/TEST_SET/PINEAPPLE/2_100.jpg')
testing('/content/drive/MyDrive/Project/Dataset/TEST_SET/PINEAPPLE/2_100.jpg')
```

```
1/1 [==============================] - 0s 22ms/step
Predicted class as: PINAPPLE
```




```
#test2
img_show('/content/drive/MyDrive/Project/Dataset/TEST_SET/BANANA/0SYXUU89Y8VZ.jpg')
testing('/content/drive/MyDrive/Project/Dataset/TEST_SET/BANANA/0SYXUU89Y8VZ.jpg')
```

```
1/1 [==============================] - 0s 21ms/step
Predicted class as: BANANA
```



Colab paid products - Cancel contracts here

✓ 1s completed at 16:35