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=5,color_mode='rgb',class_mode='sparse')
x_test = test_datagen.flow_from_directory(
    r'/content/drive/MyDrive/Project/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
```

```
 Found 4118 images belonging to 5 classes.
 Found 4118 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)
```

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

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

```
 Counter({0: 995, 1: 1354, 2: 1019, 3: 275, 4: 475})
```

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='sparse_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
824/824 [==============================] - 871s 1s/step - loss: 0.6003 - accuracy: 0.766
Epoch 2/20
824/824 [==============================] - 63s 76ms/step - loss: 0.4349 - accuracy: 0.83
Epoch 3/20
824/824 [==============================] - 63s 76ms/step - loss: 0.3810 - accuracy: 0.86
Epoch 4/20
824/824 [==============================] - 61s 74ms/step - loss: 0.3443 - accuracy: 0.87
Epoch 5/20
824/824 [==============================] - 61s 74ms/step - loss: 0.3228 - accuracy: 0.88
Epoch 6/20
824/824 [==============================] - 64s 77ms/step - loss: 0.2968 - accuracy: 0.88
Epoch 7/20
824/824 [==============================] - 62s 75ms/step - loss: 0.3033 - accuracy: 0.88
Epoch 8/20
824/824 [==============================] - 61s 74ms/step - loss: 0.2744 - accuracy: 0.89
Epoch 9/20
824/824 [==============================] - 61s 74ms/step - loss: 0.2444 - accuracy: 0.90
Epoch 10/20
824/824 [==============================] - 61s 74ms/step - loss: 0.2341 - accuracy: 0.90
Epoch 11/20
824/824 [==============================] - 62s 76ms/step - loss: 0.2359 - accuracy: 0.91
Epoch 12/20
824/824 [==============================] - 61s 74ms/step - loss: 0.2081 - accuracy: 0.92
Epoch 13/20
824/824 [==============================] - 61s 74ms/step - loss: 0.1939 - accuracy: 0.92
Epoch 14/20
824/824 [==============================] - 63s 77ms/step - loss: 0.1787 - accuracy: 0.92
Epoch 15/20
824/824 [==============================] - 61s 74ms/step - loss: 0.1706 - accuracy: 0.93
Epoch 16/20
824/824 [==============================] - 61s 74ms/step - loss: 0.1616 - accuracy: 0.94
Epoch 17/20
824/824 [==============================] - 66s 80ms/step - loss: 0.1617 - accuracy: 0.93
```

```
Epoch 18/20
824/824 [==============================] - 62s 76ms/step - loss: 0.1433 - accuracy: 0.94
Epoch 19/20
824/824 [==============================] - 63s 76ms/step - loss: 0.1135 - accuracy: 0.95
Epoch 20/20
824/824 [==============================] - 63s 76ms/step - loss: 0.1315 - accuracy: 0.95
<keras.callbacks.History at 0x7f7230b8e650>
```

Save The Model

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

Test The Model

```
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("nutrition.h5")
from tensorflow.keras.preprocessing import image


img =image.load_img(
    r"/content/drive/MyDrive/Project/Dataset/TRAIN_SET/WATERMELON/105_100.jpg",
    grayscale=False, target_size= (64,64))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
pred =model.predict(x)
pred
```

```
1/1 [==============================] - 0s 22ms/step
array([[0., 0., 0., 0., 1.]], dtype=float32)
```

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

```
'WATERMELON'
```