```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
ls
```

```
drive/  sample_data/
```

```
cd//content/drive/MyDrive/Colab Notebooks/Dataset
```

```
ls
```

```
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 computatio
#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/drive/MyDrive/Colab Notebooks/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/drive/MyDrive/Colab Notebooks/Dataset/TEST_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
```

```
print(x_train.class_indices)#checking the number of classes
```

```
print(x_test.class_indices)#checking the number of classes
```

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

```
# 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=10, validation_data=x_test,validation_steps = len(x_test))# No of images in
```

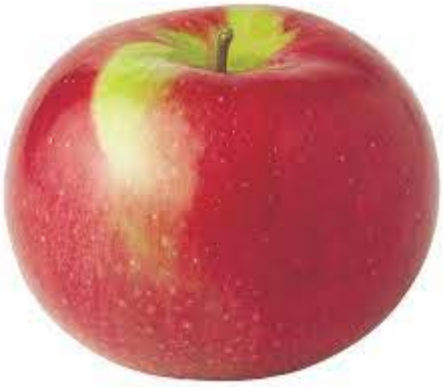
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators. This is separate from the ipykernel package so we can avoid doing imports until Epoch 1/10 828/828 [==============================] - 1189s 1s/step - loss: 0.5894 - accuracy: 0.7748 - val_loss: 0.5930 - val_accuracy: 0.7427 Epoch 2/10 828/828 [==============================] - 29s 35ms/step - loss: 0.4094 - accuracy: 0.8371 - val_loss: 0.5117 - val_accuracy: 0.8159 Epoch 3/10 828/828 [==============================] - 27s 33ms/step - loss: 0.3728 - accuracy: 0.8586 - val_loss: 0.3814 - val_accuracy: 0.8558 Epoch 4/10 828/828 [==============================] - 29s 35ms/step - loss: 0.3465 - accuracy: 0.8664 - val_loss: 0.4036 - val_accuracy: 0.8525 Epoch 5/10 828/828 [==============================] - 28s 33ms/step - loss: 0.3175 - accuracy: 0.8797 - val_loss: 0.4061 - val_accuracy: 0.8428 Epoch 6/10 828/828 [==============================] - 29s 35ms/step - loss: 0.3020 - accuracy: 0.8896 - val_loss: 0.3806 - val_accuracy: 0.8558 Epoch 7/10 828/828 [==============================] - 30s 36ms/step - loss: 0.2848 - accuracy: 0.8888 - val_loss: 0.4778 - val_accuracy: 0.8041 Epoch 8/10 828/828 [==============================] - 29s 35ms/step - loss: 0.2673 - accuracy: 0.8980 - val_loss: 0.4117 - val_accuracy: 0.8385 Epoch 9/10 828/828 [==============================] - 31s 37ms/step - loss: 0.2485 - accuracy: 0.9065 - val_loss: 0.3935 - val_accuracy: 0.8611 Epoch 10/10 828/828 [==============================] - 28s 34ms/step - loss: 0.2399 - accuracy: 0.9120 - val_loss: 0.4292 - val_accuracy: 0.8525

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

Double-click (or enter) to edit

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

```
img = image.load_img("/content/drive/MyDrive/Colab Notebooks/Dataset/TRAIN_SET/APPLES/n077
img
```



```
x=image.img_to_array(img)#conversion image into array
```

```
x
```

```
array([[[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],

       ...,

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]],
```

```
        [[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        [[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]]], dtype=float32)
```

```
x.ndim
```

3

```
x=np.expand_dims(x,axis=0) #expand the dimension
```

```
x.ndim
```

4

```
pred = classifier.predict(x)
```

1/1 [==============================] - 0s 79ms/step

```
pred
```

```
array([[1., 0., 0., 0., 0.]], dtype=float32)
```

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

'APPLES'

Colab paid products - Cancel contracts here