In [3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

In [4]:
```
ls
```

```
drive/  sample_data/
```

In [5]:
```
cd//content/drive/MyDrive/Colab Notebooks/Dataset
```

```
/content/drive/.shortcut-targets-by-id/1LL5lvl6AsdVwW9LWVu_GXEUCoV7jYm-c/Dataset
```

In [6]:
```
ls
```

```
IBM_review.pptx  photo-1589820296156-2454bb8a6ad1.jpg  TRAIN_SET/
nutrition.h5     TEST_SET/
```

Importing Neccessary Libraries

In [7]:
```
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 computation function
#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
```

Image Data Agumentation

In [8]:
```
#setting parameter for Image Data agumentation to the training data
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
#Image Data agumentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```

Loading our data and performing data agumentation

In [9]:
```
#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')
```

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

In [10]:
```
print(x_train.class_indices)#checking the number of classes
```

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

In [11]:
```
print(x_test.class_indices)#checking the number of classes
```

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

In [12]:
```
from collections import Counter as c
c(x_train .labels)
```

Out[12]:
```
Counter({0: 995, 1: 1374, 2: 1019, 3: 275, 4: 475})
```

Creating the model

In [13]:
```
# 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
```

In [14]:
```
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 model

In [15]:
```
# Compiling the CNN
# categorical_crossentropy for more than 2
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Fitting the model

In [16]:
```
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 test set
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versi
on. 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 [==============================] - 1500s 2s/step - loss: 0.6022 - accuracy: 0.7608 - val_loss: 0.6050 - val_accuracy: 0.7621
Epoch 2/10
828/828 [==============================] - 51s 62ms/step - loss: 0.4223 - accuracy: 0.8415 - val_loss: 0.4744 - val_accuracy: 0.8149
Epoch 3/10
828/828 [==============================] - 50s 70ms/step - loss: 0.3822 - accuracy: 0.8579 - val_loss: 0.4508 - val_accuracy: 0.8127
Epoch 4/10
828/828 [==============================] - 50s 61ms/step - loss: 0.3606 - accuracy: 0.8594 - val_loss: 0.4128 - val_accuracy: 0.8471
Epoch 5/10
828/828 [==============================] - 51s 61ms/step - loss: 0.3412 - accuracy: 0.8743 - val_loss: 0.4203 - val_accuracy: 0.8321
Epoch 6/10
828/828 [==============================] - 52s 62ms/step - loss: 0.3289 - accuracy: 0.8729 - val_loss: 0.4781 - val_accuracy: 0.8084
828/828 [==============================] - 51s 62ms/step - loss: 0.3006 - accuracy: 0.8859 - val_loss: 0.4085 - val_accuracy: 0.8461
Epoch 8/10
828/828 [==============================] - 52s 63ms/step - loss: 0.2810 - accuracy: 0.8862 - val_loss: 0.4500 - val_accuracy: 0.8073
Epoch 9/10
828/828 [==============================] - 50s 60ms/step - loss: 0.2838 - accuracy: 0.8925 - val_loss: 0.4216 - val_accuracy: 0.8332
Epoch 10/10
828/828 [==============================] - 52s 63ms/step - loss: 0.2580 - accuracy: 0.9016 - val_loss: 0.3874 - val_accuracy: 0.8459
```

Out[16]:

Saving our model

In [17]:
```
# Save the model
classifier.save('Nutrition.h5')
```

Nutrition Image Analysis using CNN

Predicting our results

In [18]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

In [19]:
```
img = image.load_img("/content/drive/MyDrive/Colab Notebooks/Dataset/TRAIN_SET/APPLES/r07740461_10067.jpg",target_size= (64,64))#loading of the image
img
```

Out[19]:

In [20]:
```
x=image.img_to_array(img)#conversion image into array
```

In [21]:
```
x
```

Out[21]:
```
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)
```

In [22]:
```
x.ndim
```

Out[22]: 3

In [23]:
```
x=np.expand_dims(x,axis=0) #expand the dimension
```

In [24]:
```
x.ndim
```

Out[24]: 4

In [25]:
```
pred = classifier.predict(x)
```

```
1/1 [==============================] - 0s 125ms/step
```

In [26]:
```
pred
```

Out[26]:
```
array([[1., 0., 0., 0., 0.]], dtype=float32)
```

In [27]:
```
labels=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
labels[np.argmax(pred)]
```