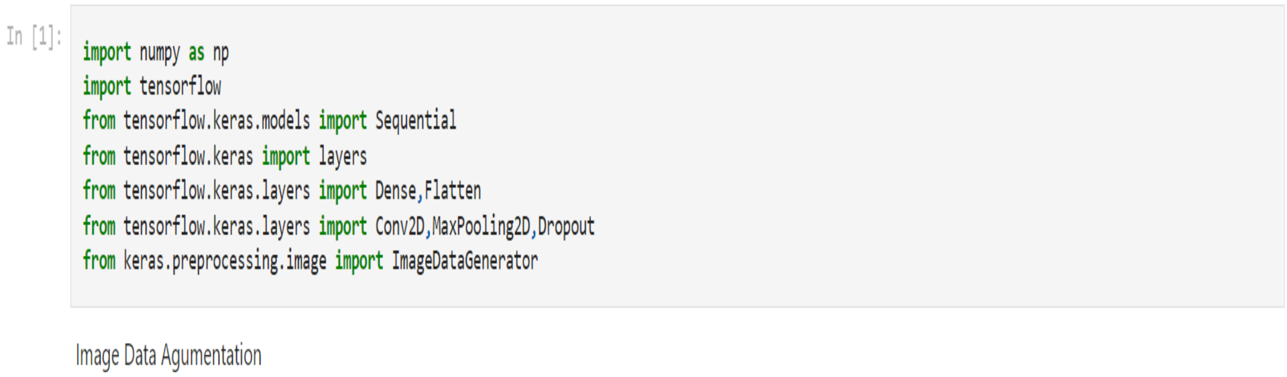
Model Building

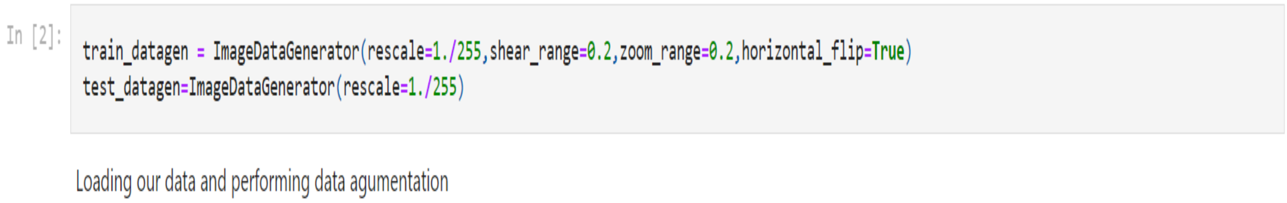
Importing the Model Building Libraries



In [1]:

```
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
```

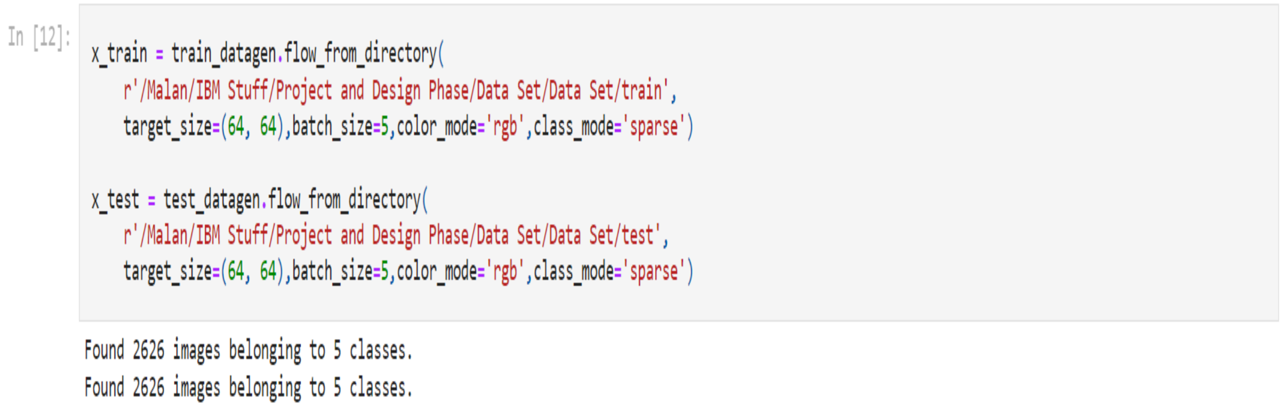
Image Data Agumentation



In [2]:

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

Loading our data and performing data agumentation

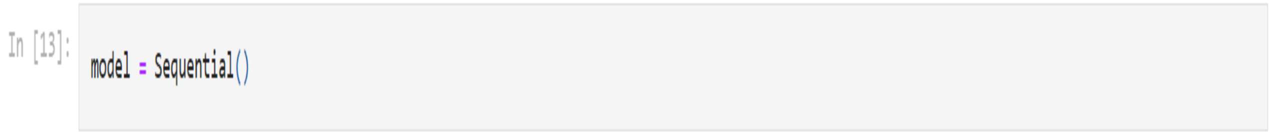


In [12]:

```
x_train = train_datagen.flow_from_directory(
    r'/Malan/IBM Stuff/Project and Design Phase/Data Set/Data Set/train',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')

x_test = test_datagen.flow_from_directory(
    r'/Malan/IBM Stuff/Project and Design Phase/Data Set/Data Set/test',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
```

```
Found 2626 images belonging to 5 classes.
Found 2626 images belonging to 5 classes.
```

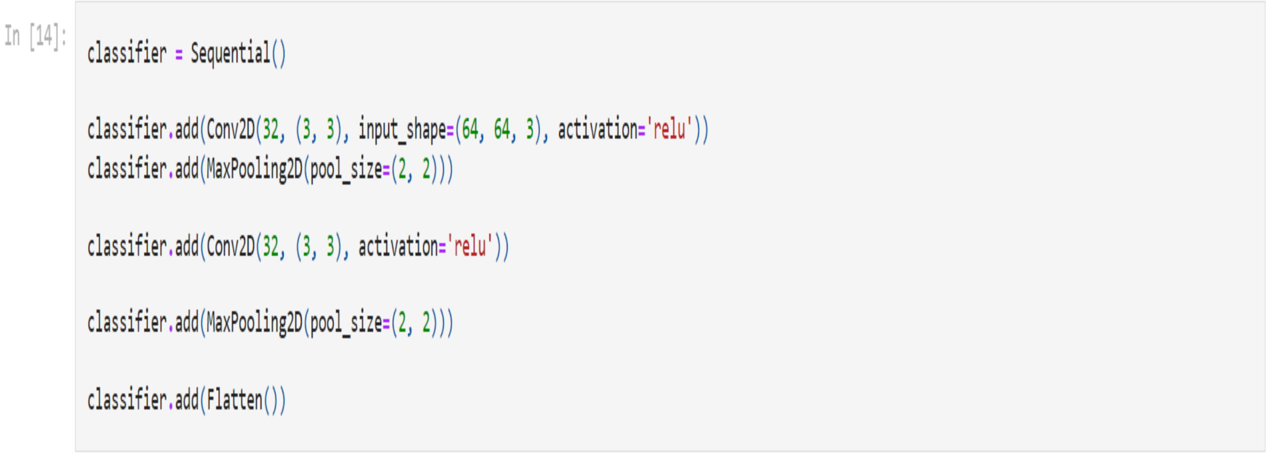
Initializing the Model



In [13]:

```
model = Sequential()
```

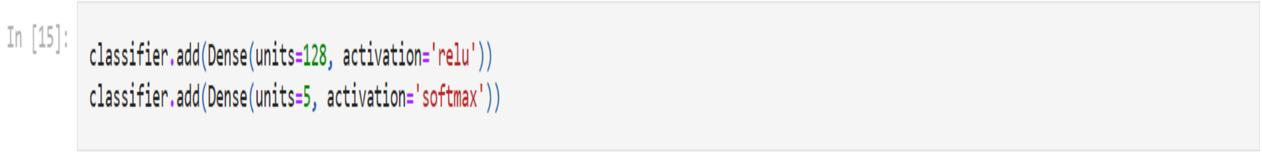
Adding CNN Layers



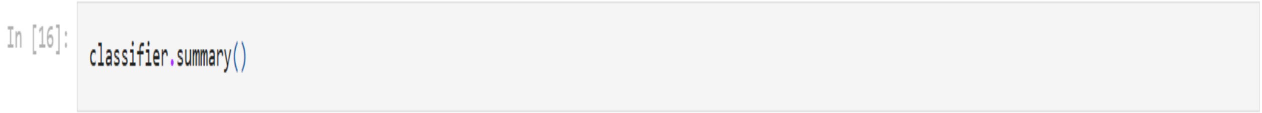
In [14]:

```
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())
```

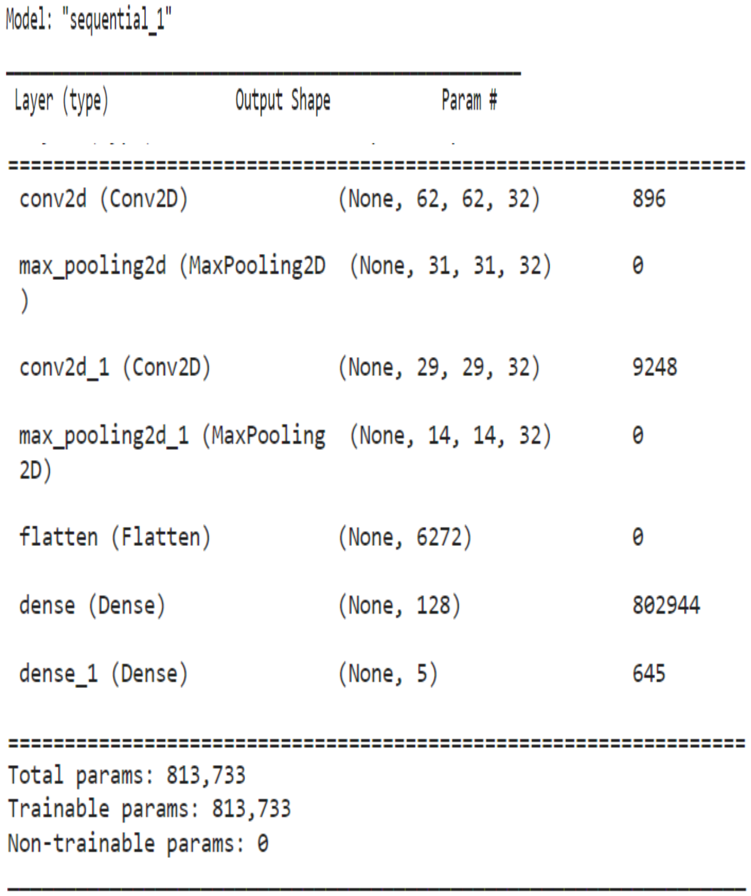
Adding Dense Layers



```
In [15]: classifier.add(Dense(units=128, activation='relu'))
         classifier.add(Dense(units=5, activation='softmax'))
```

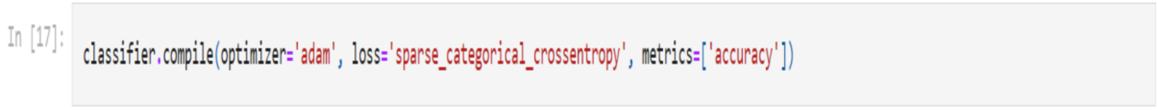


```
In [16]: classifier.summary()
```



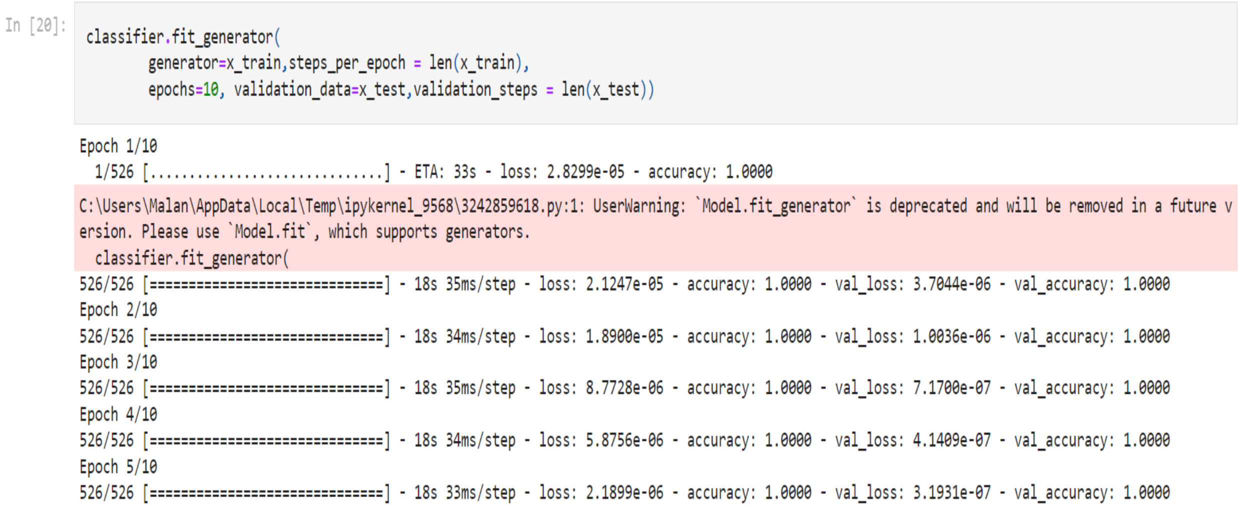
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 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



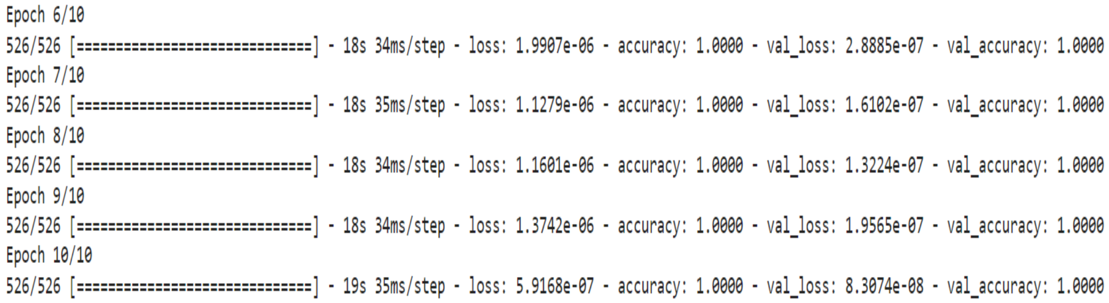
```
In [17]: classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Train The Model



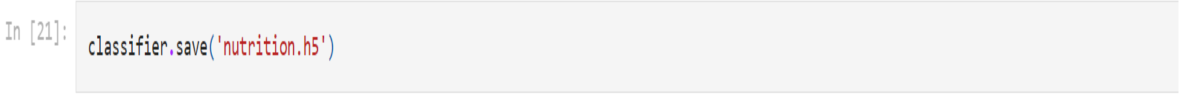
```
In [20]: classifier.fit_generator(
                generator=x_train,steps_per_epoch = len(x_train),
                epochs=10, validation_data=x_test,validation_steps = len(x_test))
```

```
Epoch 1/10
  1/526 [..............................] - ETA: 33s - loss: 2.8299e-05 - accuracy: 1.0000
C:\Users\Malan\AppData\Local\Temp\ipykernel_9568\3242859618.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future v
ersion. Please use `Model.fit`, which supports generators.
  classifier.fit_generator(
526/526 [==============================] - 18s 35ms/step - loss: 2.1247e-05 - accuracy: 1.0000 - val_loss: 3.7044e-06 - val_accuracy: 1.0000
Epoch 2/10
526/526 [==============================] - 18s 34ms/step - loss: 1.8900e-05 - accuracy: 1.0000 - val_loss: 1.0036e-06 - val_accuracy: 1.0000
Epoch 3/10
526/526 [==============================] - 18s 35ms/step - loss: 8.7728e-06 - accuracy: 1.0000 - val_loss: 7.1700e-07 - val_accuracy: 1.0000
Epoch 4/10
526/526 [==============================] - 18s 34ms/step - loss: 5.8756e-06 - accuracy: 1.0000 - val_loss: 4.1409e-07 - val_accuracy: 1.0000
Epoch 5/10
526/526 [==============================] - 18s 33ms/step - loss: 2.1899e-06 - accuracy: 1.0000 - val_loss: 3.1931e-07 - val_accuracy: 1.0000
```



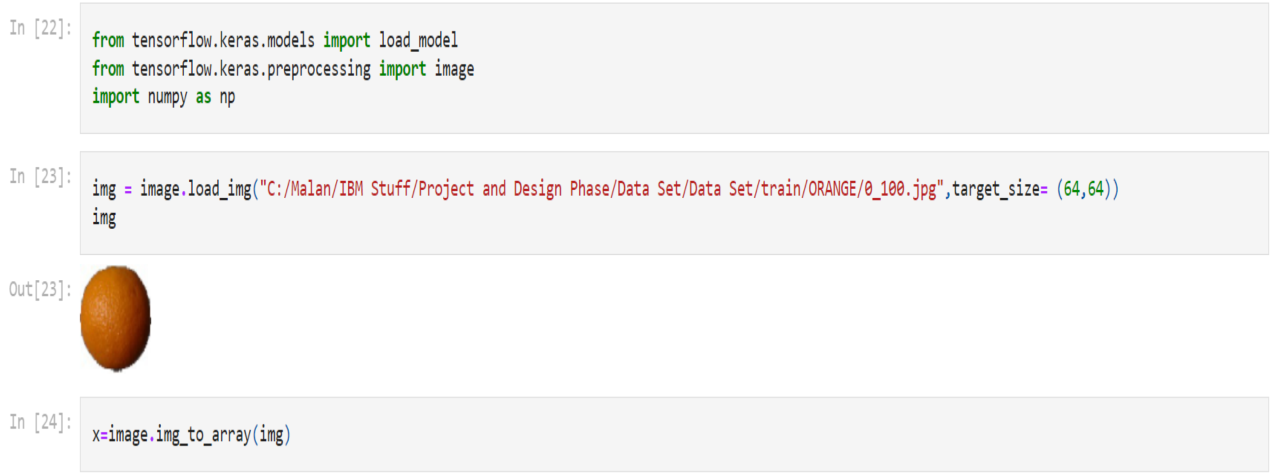
```
Epoch 6/10
526/526 [==============================] - 18s 34ms/step - loss: 1.9907e-06 - accuracy: 1.0000 - val_loss: 2.8885e-07 - val_accuracy: 1.0000
Epoch 7/10
526/526 [==============================] - 18s 35ms/step - loss: 1.1279e-06 - accuracy: 1.0000 - val_loss: 1.6102e-07 - val_accuracy: 1.0000
Epoch 8/10
526/526 [==============================] - 18s 34ms/step - loss: 1.1601e-06 - accuracy: 1.0000 - val_loss: 1.3224e-07 - val_accuracy: 1.0000
Epoch 9/10
526/526 [==============================] - 18s 34ms/step - loss: 1.3742e-06 - accuracy: 1.0000 - val_loss: 1.9565e-07 - val_accuracy: 1.0000
Epoch 10/10
526/526 [==============================] - 19s 35ms/step - loss: 5.9168e-07 - accuracy: 1.0000 - val_loss: 8.3074e-08 - val_accuracy: 1.0000
```

Save The Model



```
In [21]: classifier.save('nutrition.h5')
```

Test The Model

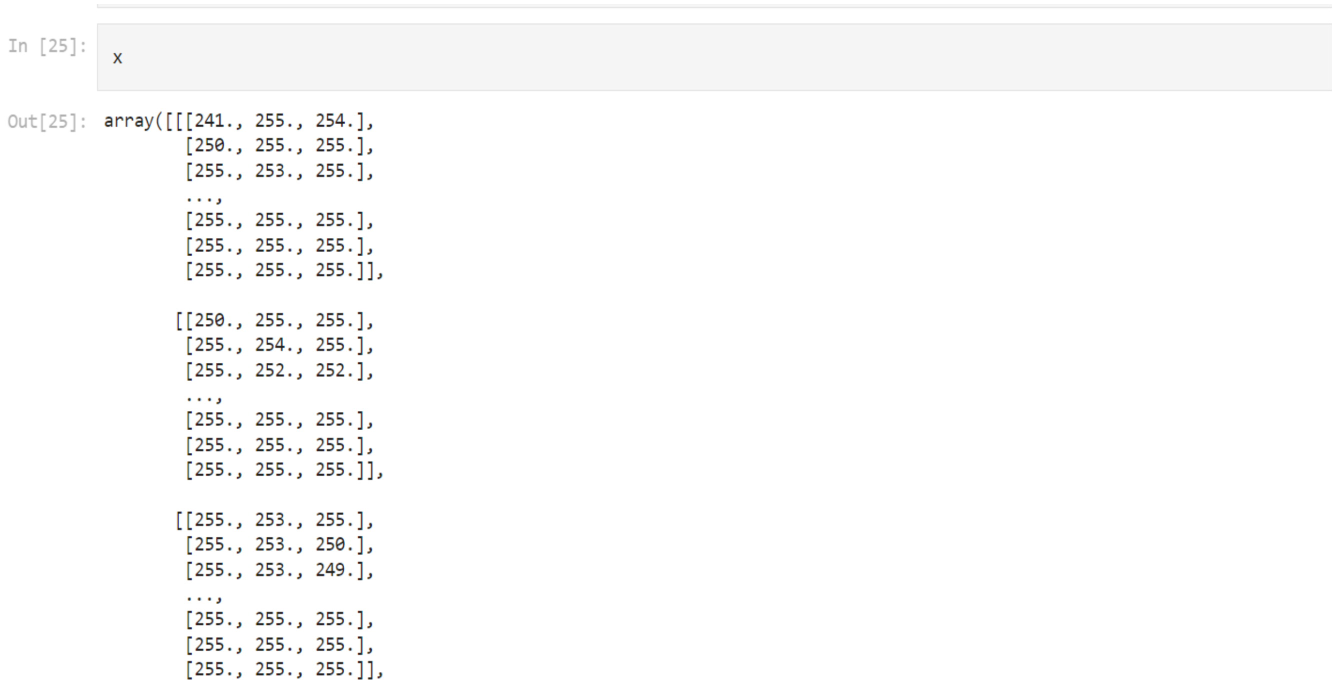


```
In [22]: from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image
         import numpy as np
```

```
In [23]: img = image.load_img("C:/Malan/IBM Stuff/Project and Design Phase/Data Set/Data Set/train/ORANGE/0_100.jpg",target_size= (64,64))
         img
```

Out[23]:

```
In [24]: x=image.img_to_array(img)
```



```
In [25]: x
```

```
Out[25]: array([[[[241., 255., 254.],
                  [250., 255., 255.],
                  [255., 253., 255.],
                  ...,
                  [255., 255., 255.],
                  [255., 255., 255.],
                  [255., 255., 255.]],

                 [[250., 255., 255.],
                  [255., 254., 255.],
                  [255., 252., 252.],
                  ...,
                  [255., 255., 255.],
                  [255., 255., 255.],
                  [255., 255., 255.]],

                 [[255., 253., 255.],
                  [255., 253., 250.],
                  [255., 253., 249.],
                  ...,
                  [255., 255., 255.],
                  [255., 255., 255.],
                  [255., 255., 255.]],
 [[255., 253., 255.],
  [255., 253., 250.],
  [255., 253., 249.],
  ...,
  [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 [26]: x.ndim

Out[26]: 3

In [27]: x=np.expand_dims(x,axis=0)

In [28]: x.ndim

Out[28]: 4

In [29]: pred = classifier.predict(x)

         1/1 [==============================] - 0s 213ms/step

In [30]: pred

Out[30]: array([[0., 0., 1., 0., 0.]], dtype=float32)


In [31]: labels=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
         labels[np.argmax(pred)]

Out[31]: 'ORANGE'

In [ ]:
```