Project Development Phase

Sprint 2

Date	7 November 2022
Team ID	PNT2022TMID20581
Project Name	Project - AI-Powered Nutrition Analyzer For Fitness Enthusiasts

Building A CNN model

Steps to Build a Deep Learning Model

1. Defining the model architecture
2. Configure the learning process
3. Train The Model
4. Save the Model
5. Predictions

In [1]:

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

In [2]:

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

In [3]:

```
#performing data agumentation to train data
x_train = train_datagen.flow_from_directory(r'E:\IBM_Project\TRAIN_SET',target_si
#performing data agumentation to test data
x_test = test_datagen.flow_from_directory(r'E:\IBM_Project\TEST_SET',target_size=
```

```
Found 2626 images belonging to 5 classes.
Found 1055 images belonging to 5 classes.
```

In [4]:

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense,Flatten,Conv2D,MaxPooling2D,Dropout
import scipy
```

In [5]:

```
model=Sequential()
```

In [6]:

```
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())
```

In [7]:

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

In [8]:
```
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
_________________________________________________________________
```

In [9]:
```
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metr
```

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

```
C:\Users\HAPPY\AppData\Local\Temp\ipykernel_5084\44351418.py:1: UserWarning: `M
odel.fit_generator` is deprecated and will be removed in a future version. Plea
se use `Model.fit`, which supports generators.
  classifier.fit_generator(

Epoch 1/20
526/526 [==============================] - 61s 114ms/step - loss: 0.1382 - accu
racy: 0.9524 - val_loss: 0.0255 - val_accuracy: 0.9867
Epoch 2/20
526/526 [==============================] - 19s 35ms/step - loss: 0.0216 - accur
acy: 0.9935 - val_loss: 0.0121 - val_accuracy: 0.9943
Epoch 3/20
526/526 [==============================] - 19s 36ms/step - loss: 0.0184 - accur
acy: 0.9943 - val_loss: 0.0411 - val_accuracy: 0.9820
Epoch 4/20
526/526 [==============================] - 20s 38ms/step - loss: 6.9527e-05 - a
ccuracy: 1.0000 - val_loss: 0.0472 - val_accuracy: 0.9754
Epoch 5/20
526/526 [==============================] - 20s 37ms/step - loss: 3.6160e-05 - a
ccuracy: 1.0000 - val_loss: 0.0350 - val_accuracy: 0.9848
Epoch 6/20
526/526 [==============================] - 20s 37ms/step - loss: 8.3112e-05 - a
ccuracy: 1.0000 - val_loss: 0.0514 - val_accuracy: 0.9801
Epoch 7/20
526/526 [==============================] - 20s 38ms/step - loss: 1.3292e-05 - a
ccuracy: 1.0000 - val_loss: 0.0401 - val_accuracy: 0.9848
Epoch 8/20
526/526 [==============================] - 20s 39ms/step - loss: 7.7658e-06 - a
ccuracy: 1.0000 - val_loss: 0.0274 - val_accuracy: 0.9896
Epoch 9/20
526/526 [==============================] - 20s 38ms/step - loss: 6.0170e-06 - a
ccuracy: 1.0000 - val_loss: 0.0297 - val_accuracy: 0.9820
Epoch 10/20
526/526 [==============================] - 21s 40ms/step - loss: 4.4193e-06 - a
ccuracy: 1.0000 - val_loss: 0.0412 - val_accuracy: 0.9791
Epoch 11/20
526/526 [==============================] - 22s 42ms/step - loss: 3.5176e-06 - a
ccuracy: 1.0000 - val_loss: 0.0342 - val_accuracy: 0.9801
Epoch 12/20
526/526 [==============================] - 22s 41ms/step - loss: 2.7885e-06 - a
ccuracy: 1.0000 - val_loss: 0.0397 - val_accuracy: 0.9820
Epoch 13/20
526/526 [==============================] - 22s 42ms/step - loss: 2.1186e-06 - a
ccuracy: 1.0000 - val_loss: 0.0264 - val_accuracy: 0.9915
Epoch 14/20
526/526 [==============================] - 19s 36ms/step - loss: 1.3492e-06 - a
ccuracy: 1.0000 - val_loss: 0.0445 - val_accuracy: 0.9810
Epoch 15/20
526/526 [==============================] - 20s 38ms/step - loss: 1.7433e-06 - a
ccuracy: 1.0000 - val_loss: 0.0324 - val_accuracy: 0.9848
Epoch 16/20
526/526 [==============================] - 21s 40ms/step - loss: 1.2478e-06 - a
ccuracy: 1.0000 - val_loss: 0.0336 - val_accuracy: 0.9829
```

```
Epoch 17/20
526/526 [==============================] - 20s 38ms/step - loss: 8.8117e-07 - a
ccuracy: 1.0000 - val_loss: 0.0345 - val_accuracy: 0.9877
Epoch 18/20
526/526 [==============================] - 20s 38ms/step - loss: 4.0506e-07 - a
ccuracy: 1.0000 - val_loss: 0.0398 - val_accuracy: 0.9801
Epoch 19/20
526/526 [==============================] - 19s 37ms/step - loss: 4.2712e-07 - a
ccuracy: 1.0000 - val_loss: 0.0400 - val_accuracy: 0.9801
Epoch 20/20
526/526 [==============================] - 20s 37ms/step - loss: 2.8308e-07 - a
ccuracy: 1.0000 - val_loss: 0.0295 - val_accuracy: 0.9801
```

Out[10]:
```
<keras.callbacks.History at 0x24b6018c220>
```

In [11]:
```
classifier.save('nutrition.h5')
```

In [12]:
```
from keras.models import load_model
from keras.preprocessing import image
from tensorflow.keras.utils import load_img, img_to_array
model=load_model("nutrition.h5")
```

In [13]:
```
img=load_img(r"E:\IBM_Project\Sample\orange.jpg",grayscale=False,target_size=(64,
x=img_to_array(img)
x=np.expand_dims(x,axis=0)
predict_x=model.predict(x)
pred = np.argmax(predict_x,axis=1)
pred
```

```
1/1 [==============================] - 0s 174ms/step
```

Out[13]:
```
array([2], dtype=int64)
```

In [14]:
```
index=['APPELES','BANANA','ORANGE','PINEAPPLE','WATERMELON']
result=str(index[pred[0]])
result
```

Out[14]:
```
'ORANGE'
```

In [ ]: