Sprint-2

Model Building

Date	01 November 2022
Team ID	PNT2022TMID37658
Project Name	AI-powered Nutrition Analyzer for Fitness Enthusiasts
Maximum Marks	

Dataset:

- In our dataset we have collected images of the five variety of fruits.
  - Apple
  - Orange
  - Pineapple
  - Watermelon
  - Banana

Drive link :

https://drive.google.com/file/d/1hgEWyKicgrntbY5LSkuW_v6G4C93AQfN/view?usp=share_link

Image Pre-processing:

- Import The ImageDataGenerator Library
- Configure ImageDataGenerator Class
- Apply Image DataGenerator Functionality To Trainset And Testset

Model Building:

- Importing The Model Building Libraries
- Initializing The Model
- Adding CNN Layers
- Adding Dense Layers
- Configure The Learning Process
- Train the model
- Save the model
- Test the model

Date :01 NOVEMBER 2022

Team ID :PNT2022TMID30252

Project Name : AI-powered Nutrition Analyzer for Fitness Enthusiasts

Data Collection

Download the dataset here

```
# Unzipping the dataset
!unzip '/content/Dataset.zip'
```

```
  inflating: Dataset/TEST_SET/PINEAPPLE/5.jpeg
  inflating: Dataset/TEST_SET/PINEAPPLE/5.jpg
  inflating: Dataset/TEST_SET/PINEAPPLE/6.jpeg
  inflating: Dataset/TEST_SET/PINEAPPLE/7.jpeg
  inflating: Dataset/TEST_SET/PINEAPPLE/8.jpeg
  inflating: Dataset/TEST_SET/PINEAPPLE/9.jpeg
  inflating: Dataset/TEST_SET/PINEAPPLE/PINEAPPLE 1.jpeg
   creating: Dataset/TEST_SET/WATERMELON/
  inflating: Dataset/TEST_SET/WATERMELON/1.jpg
  inflating: Dataset/TEST_SET/WATERMELON/10.jpg
  inflating: Dataset/TEST_SET/WATERMELON/11.jpg
  inflating: Dataset/TEST_SET/WATERMELON/12.jpg
  inflating: Dataset/TEST_SET/WATERMELON/13.jpg
  inflating: Dataset/TEST_SET/WATERMELON/14.jpg
  inflating: Dataset/TEST_SET/WATERMELON/15.jpg
  inflating: Dataset/TEST_SET/WATERMELON/16.jpg
  inflating: Dataset/TEST_SET/WATERMELON/17.jpg
  inflating: Dataset/TEST_SET/WATERMELON/18.jpg
  inflating: Dataset/TEST_SET/WATERMELON/19.jpg
  inflating: Dataset/TEST_SET/WATERMELON/2.jpg
  inflating: Dataset/TEST_SET/WATERMELON/20.jpg
  inflating: Dataset/TEST_SET/WATERMELON/3.jpg
  inflating: Dataset/TEST_SET/WATERMELON/4.jpg
  inflating: Dataset/TEST_SET/WATERMELON/5.jpg
  inflating: Dataset/TEST_SET/WATERMELON/6.jpg
  inflating: Dataset/TEST_SET/WATERMELON/7.jpg
  inflating: Dataset/TEST_SET/WATERMELON/8.jpg
  inflating: Dataset/TEST_SET/WATERMELON/9.jpg
 creating: Dataset/TRAIN_SET/
  creating: Dataset/TRAIN_SET/APPLES/
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10012.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10019.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10037.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10065.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10067.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10074.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10104.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10128.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10129.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10166.jpg
```

```
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10183.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10218.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10219.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10239.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10242.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10257.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10266.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10273.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10284.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_1033.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10335.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10336.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10357.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10363.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10369.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10374.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10403.jpg
  inflating: Dataset/TRAIN_SET/APPLES/n07740461_10409.jpg
```

Image Preprocessing

```
#Importing The ImageDataGenerator Library
from keras.preprocessing.image import ImageDataGenerator
```

Image Data Augmentation

```
#Configure ImageDataGenerator Class
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizonta
test_datagen=ImageDataGenerator(rescale=1./255)
```

Applying Image DataGenerator Functionality To Trainset And Testset

```
#Applying Image DataGenerator Functionality To Trainset And Testset
x_train = train_datagen.flow_from_directory(
    r'/content/Dataset/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
#Applying Image DataGenerator Functionality To Testset
x_test = test_datagen.flow_from_directory(
    r'/content/Dataset/TEST_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')

 Found 4118 images belonging to 5 classes.
 Found 974 images belonging to 5 classes.


#checking the number of classes
```

```
print(x_train.class_indices)
```

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

```
#checking the number of classes
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})
```

Model Building

1. Importing The Model Building Libraries

```
import numpy as np
import tensorflow as tf
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
```

2. Initializing The Model

```
model = Sequential()
```

3. Adding CNN Layers

```
# 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())
```

4. Adding Dense Layers

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

```
#summary of our model
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
_________________________________________________________________
```

5. Configure The Learning Process

```
# Compiling the CNN
# categorical_crossentropy for more than 2
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc
```

6. Train The Model

```
#Fitting the model
classifier.fit_generator(generator=x_train,steps_per_epoch = len(x_train),epochs=20, valid
```

```
Epoch 1/20
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.

824/824 [==============================] - 19s 14ms/step - loss: 0.6293 - accuracy:
```

```
Epoch 2/20
824/824 [==============================] - 11s 14ms/step - loss: 0.4228 - accuracy:
Epoch 3/20
824/824 [==============================] - 12s 15ms/step - loss: 0.3789 - accuracy:
Epoch 4/20
824/824 [==============================] - 12s 14ms/step - loss: 0.3555 - accuracy:
Epoch 5/20
824/824 [==============================] - 12s 14ms/step - loss: 0.3281 - accuracy:
Epoch 6/20
824/824 [==============================] - 11s 14ms/step - loss: 0.3357 - accuracy:
Epoch 7/20
824/824 [==============================] - 11s 14ms/step - loss: 0.2962 - accuracy:
Epoch 8/20
824/824 [==============================] - 11s 14ms/step - loss: 0.2780 - accuracy:
Epoch 9/20
824/824 [==============================] - 12s 15ms/step - loss: 0.2577 - accuracy:
Epoch 10/20
824/824 [==============================] - 11s 14ms/step - loss: 0.2429 - accuracy:
Epoch 11/20
824/824 [==============================] - 11s 14ms/step - loss: 0.2243 - accuracy:
Epoch 12/20
824/824 [==============================] - 12s 14ms/step - loss: 0.2073 - accuracy:
Epoch 13/20
824/824 [==============================] - 11s 14ms/step - loss: 0.1976 - accuracy:
Epoch 14/20
824/824 [==============================] - 11s 14ms/step - loss: 0.1726 - accuracy:
Epoch 15/20
824/824 [==============================] - 13s 15ms/step - loss: 0.1652 - accuracy:
Epoch 16/20
824/824 [==============================] - 12s 15ms/step - loss: 0.1594 - accuracy:
Epoch 17/20
824/824 [==============================] - 11s 14ms/step - loss: 0.1771 - accuracy:
Epoch 18/20
824/824 [==============================] - 11s 14ms/step - loss: 0.1467 - accuracy:
Epoch 19/20
824/824 [==============================] - 11s 14ms/step - loss: 0.1316 - accuracy:
Epoch 20/20
824/824 [==============================] - 11s 14ms/step - loss: 0.1398 - accuracy:
<keras.callbacks.History at 0x7f444074a8d0>
```

7. Saving The Model

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

8. Testing The Model

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


from tensorflow.keras.utils import img_to_array
```

```
#loading of the image
img = load_img(r'/content/Sample_Images/Test_Image1.jpg',grayscale=False,target_size= (64,
#image to array
x = img_to_array(img)
#changing the shape
x = np.expand_dims(x,axis = 0)
predict_x=model.predict(x)
classes_x=np.argmax(predict_x,axis=-1)
classes_x
```

```
1/1 [==============================] - 0s 14ms/step
array([0])
```

```
index=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
result=str(index[classes_x[0]])
result
```

```
'APPLES'
```