SPRINT-2

Model Building

Training,Saving,Testing the Model

Date	02 November 2022
Team ID	PNT2022TMID15882
Project Name	AI-powered Nutrition Analyzer for Fitness Enthusiasts

Dataset:

In our dataset we have collected images of the five variety of fruits.

- Apple
- Orange
- Pineapple
- Watermelon
- Banana

Image Pre-processing:

- Import The ImageDataGenerator Library
- Configure ImageDataGenerator Class
- Apply Image DataGenerator Functionality To Training dataset And Testing dataset

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

- # Data Collection:

  Download the test and train data using the drive link

  Drive link : https://drive.google.com/file/d/1jzDjV7jYcIzlIieagaJdubMJ3YeLsry1/view?usp=share_link

- # Image Preprocessing

  - ## Image Data Augmentation

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

  - ## Applying Image DataGenerator Functionality To Train dataset And Test dataset

```
x_train = train_datagen.flow_from_directory(r'TRAIN_SET',
                                            target_size=(64, 64),
                                            batch_size=5,
                                            color_mode='rgb',
                                            class_mode='sparse')

x_test = test_datagen.flow_from_directory(r'TEST_SET',
                                          target_size=(64, 64),
                                          batch_size=5,
                                          color_mode='rgb',
                                          class_mode='sparse')
```

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

- # Model Building

  - ## Importing The Model Building Libraries

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

  - ## Initializing The Model

```
classifier=Sequential()
```

  - ## Adding CNN Layers

```
# 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 Dense Layers

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

```
[54]: classifier.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 62, 62, 32)        896

 max_pooling2d_8 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 conv2d_9 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_9 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 6272)              0

 dense_8 (Dense)             (None, 128)               802944

 dense_9 (Dense)             (None, 5)                 645

=================================================================
Total params: 813,733
Trainable params: 813,733
Non-trainable params: 0
_________________________________________________________________
```

- Configure The Learning Process

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

- Train The Model

```
classifier.fit_generator(x_train,
                    steps_per_epoch = len(x_train) ,
                    epochs = 20,
                    validation_data = x_test,
                    validation_steps = len(x_test) )
```

```
Epoch 1/20
/tmp/ipykernel_114941/4012943898.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  classifier.fit_generator(x_train,
526/526 [==============================] - 5s 9ms/step - loss: 0.1562 - accuracy: 0.9455 - val_loss: 0.0384 - val_accuracy: 0.9820
Epoch 2/20
526/526 [==============================] - 4s 7ms/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0528 - val_accuracy: 0.9791
Epoch 3/20
526/526 [==============================] - 6s 11ms/step - loss: 1.7018e-04 - accuracy: 1.0000 - val_loss: 0.0316 - val_accuracy: 0.9801
Epoch 4/20
526/526 [==============================] - 5s 9ms/step - loss: 1.1498e-04 - accuracy: 1.0000 - val_loss: 0.0199 - val_accuracy: 0.9953
Epoch 5/20
526/526 [==============================] - 8s 14ms/step - loss: 6.1711e-05 - accuracy: 1.0000 - val_loss: 0.0234 - val_accuracy: 0.9829
Epoch 6/20
526/526 [==============================] - 8s 15ms/step - loss: 3.1265e-05 - accuracy: 1.0000 - val_loss: 0.0367 - val_accuracy: 0.9801
Epoch 7/20
526/526 [==============================] - 8s 14ms/step - loss: 1.8764e-05 - accuracy: 1.0000 - val_loss: 0.0320 - val_accuracy: 0.9810
Epoch 8/20
526/526 [==============================] - 7s 14ms/step - loss: 1.5496e-05 - accuracy: 1.0000 - val_loss: 0.0298 - val_accuracy: 0.9810
Epoch 9/20
526/526 [==============================] - 8s 14ms/step - loss: 8.6502e-06 - accuracy: 1.0000 - val_loss: 0.0253 - val_accuracy: 0.9829
Epoch 10/20
526/526 [==============================] - 7s 14ms/step - loss: 8.4784e-06 - accuracy: 1.0000 - val_loss: 0.0205 - val_accuracy: 0.9848
Epoch 11/20
526/526 [==============================] - 8s 14ms/step - loss: 0.0745 - accuracy: 0.9817 - val_loss: 0.4974 - val_accuracy: 0.7630
Epoch 12/20
526/526 [==============================] - 8s 14ms/step - loss: 0.0431 - accuracy: 0.9871 - val_loss: 0.0241 - val_accuracy: 0.9924
Epoch 13/20
526/526 [==============================] - 8s 14ms/step - loss: 1.6750e-04 - accuracy: 1.0000 - val_loss: 0.0305 - val_accuracy: 0.9791
Epoch 14/20
526/526 [==============================] - 8s 15ms/step - loss: 7.6800e-05 - accuracy: 1.0000 - val_loss: 0.0094 - val_accuracy: 1.0000
Epoch 15/20
526/526 [==============================] - 8s 15ms/step - loss: 3.6856e-05 - accuracy: 1.0000 - val_loss: 0.0151 - val_accuracy: 0.9924
Epoch 16/20
526/526 [==============================] - 8s 15ms/step - loss: 2.1933e-05 - accuracy: 1.0000 - val_loss: 0.0159 - val_accuracy: 0.9896
Epoch 17/20
526/526 [==============================] - 8s 15ms/step - loss: 1.3638e-05 - accuracy: 1.0000 - val_loss: 0.0200 - val_accuracy: 0.9867
Epoch 18/20
526/526 [==============================] - 8s 15ms/step - loss: 9.5200e-06 - accuracy: 1.0000 - val_loss: 0.0211 - val_accuracy: 0.9848
Epoch 19/20
526/526 [==============================] - 7s 14ms/step - loss: 5.6362e-06 - accuracy: 1.0000 - val_loss: 0.0219 - val_accuracy: 0.9829
Epoch 20/20
526/526 [==============================] - 8s 14ms/step - loss: 4.6769e-06 - accuracy: 1.0000 - val_loss: 0.0148 - val_accuracy: 0.9896
<keras.callbacks.History at 0x7fcd027870a0>
```

- Saving The Model

```
classifier.save('Nutrition-Analysis.h5')
```

- Testing The Model

```
from tensorflow.keras.models import load_model
# from keras.preprocessing import image
import keras
import numpy as np
import glob
import matplotlib.pyplot as plt
```

```
model=load_model("Nutrition-Analysis.h5")
```

```
img=keras.utils.load_img(r"test_16.jpg",grayscale=False,target_size=(64,64))

x=keras.utils.img_to_array(img)
x=np.expand_dims(x,axis=0)

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

```
1/1 [==============================] - 0s 51ms/step
array([[0., 0., 0., 0., 1.]], dtype=float32)
```

```
np.argmax(pred)
```

```
4
```

```
index=["APPLES","BANANAS","ORANGE","PINEAPPLE","WATERMELON"]
result=str(index[np.argmax(pred)])
result
```

```
'WATERMELON'
```

```
fig=plt.figure(figsize=(5,5))
count=0
testing_imgs=glob.glob("*.jpg")
for i in testing_imgs:
    count+=1
    img=keras.utils.load_img(i,grayscale=False,target_size=(64,64))


    x=keras.utils.img_to_array(img)
    x=np.expand_dims(x,axis=0)

    pred=model.predict(x)
    print(pred)
    # plt.imshow(img)
    result=str(index[np.argmax(pred)])
    print(result)
    fig=plt.figure(figsize=(5,5))
    plt.title(result)
    plt.axis("off")
    plt.imshow(img)
```

```
1/1 [==============================] - 0s 15ms/step
[[1. 0. 0. 0. 0.]]
APPLES
1/1 [==============================] - 0s 15ms/step
[[1. 0. 0. 0. 0.]]
APPLES
1/1 [==============================] - 0s 15ms/step
[[1. 0. 0. 0. 0.]]
APPLES
1/1 [==============================] - 0s 13ms/step
[[0. 1. 0. 0. 0.]]
BANANAS
1/1 [==============================] - 0s 13ms/step
[[0. 0. 1. 0. 0.]]
ORANGE
1/1 [==============================] - 0s 14ms/step
[[0. 0. 0. 0. 1.]]
WATERMELON
1/1 [==============================] - 0s 15ms/step
[[0. 0. 0. 0. 1.]]
WATERMELON
1/1 [==============================] - 0s 14ms/step
[[0. 1. 0. 0. 0.]]
BANANAS
1/1 [==============================] - 0s 13ms/step
[[0. 1. 0. 0. 0.]]
BANANAS
1/1 [==============================] - 0s 26ms/step
[[0. 1. 0. 0. 0.]]
BANANAS
1/1 [==============================] - 0s 13ms/step
[[0. 0. 0. 0. 1.]]
WATERMELON
1/1 [==============================] - 0s 13ms/step
[[0. 0. 0. 0. 1.]]
WATERMELON
1/1 [==============================] - 0s 14ms/step
[[0.0000000e+00 1.0000000e+00 0.0000000e+00 0.0000000e+00 3.2440544e-20]]
BANANAS
1/1 [==============================] - 0s 13ms/step
[[0. 0. 0. 0. 1.]]
WATERMELON
1/1 [==============================] - 0s 14ms/step
[[0. 0. 0. 1. 0.]]
PINEAPPLE
<Figure size 360x360 with 0 Axes>
```

APPLES

APPLES

APPLES