Sprint-2

Model Building (Training, Saving, Testing the model)

Date	11 November 2022
Team ID	PNT2022TMID21496
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.
  - Orange
  - Pineapple
  - Watermelon
  - Banana
  - Apple

Drive link :
https://drive.google.com/drive/folders/1NxPZRRJtUgdPZd_81N61gTjNEaqdwcR7?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 :11 November 2022

Team ID :PNT2022TMID21496

Project Name : AI-powered Nutrition Analyzer for Fitness Enthusiasts

Data Collection

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
cd/content/drive/MyDrive/Colab Notebooks
```

```
/content/drive/MyDrive/Colab Notebooks
```

```
!unzip 'Dataset.zip'
```

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 TrainsetAnd Testset

```
#Applying Image DataGenerator Functionality To Trainset And Testset
x_train = train_datagen.flow_from_directory(r'/content/drive/MyDrive/Colab Notebooks/Datas
```

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

```
#Applying Image DataGenerator Functionality To Testset
x_test = test_datagen.flow_from_directory(r'/content/drive/MyDrive/Colab Notebooks/Dataset
```

```
Found 929 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

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
```

Initializing The Model

```
model=Sequential()
```

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())
```

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
_________________________________________________________________
```

Configure The Learning Process

```
# Compiling the CNN
# categorical_crossentropy for more than 2
classifier.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Train The Model

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.f
  This is separate from the ipykernel package so we can avoid doing imports until
Epoch 1/20
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserW
  return dispatch_target(*args, **kwargs)
824/824 [==============================] - 1088s 1s/step - loss: 0.5946 - accuracy: 0
Epoch 2/20
824/824 [==============================] - 71s 86ms/step - loss: 0.4141 - accuracy: 0
Epoch 3/20
824/824 [==============================] - 67s 81ms/step - loss: 0.3753 - accuracy: 0
Epoch 4/20
824/824 [==============================] - 69s 83ms/step - loss: 0.3600 - accuracy: 0
Epoch 5/20
824/824 [==============================] - 68s 83ms/step - loss: 0.3251 - accuracy: 0
Epoch 6/20
824/824 [==============================] - 67s 82ms/step - loss: 0.3078 - accuracy: 0
Epoch 7/20
824/824 [==============================] - 70s 85ms/step - loss: 0.2914 - accuracy: 0
Epoch 8/20
824/824 [==============================] - 68s 82ms/step - loss: 0.2832 - accuracy: 0
Epoch 9/20
824/824 [==============================] - 68s 82ms/step - loss: 0.2655 - accuracy: 0
Epoch 10/20
824/824 [==============================] - 71s 86ms/step - loss: 0.2530 - accuracy: 0
Epoch 11/20
824/824 [==============================] - 69s 84ms/step - loss: 0.2376 - accuracy: 0
Epoch 12/20
824/824 [==============================] - 69s 84ms/step - loss: 0.2253 - accuracy: 0
Epoch 13/20
824/824 [==============================] - 67s 82ms/step - loss: 0.2157 - accuracy: 0
Epoch 14/20
824/824 [==============================] - 68s 82ms/step - loss: 0.1983 - accuracy: 0
Epoch 15/20
824/824 [==============================] - 85s 103ms/step - loss: 0.2154 - accuracy:
Epoch 16/20
824/824 [==============================] - 70s 85ms/step - loss: 0.1912 - accuracy: 0
Epoch 17/20
824/824 [==============================] - 68s 83ms/step - loss: 0.1658 - accuracy: 0
Epoch 18/20
824/824 [==============================] - 68s 83ms/step - loss: 0.1512 - accuracy: 0
Epoch 19/20
824/824 [==============================] - 66s 81ms/step - loss: 0.1516 - accuracy: 0
Epoch 20/20
824/824 [==============================] - 68s 83ms/step - loss: 0.1422 - accuracy: 0
<keras.callbacks.History at 0x7fb17d56db10>
```

Saving The Model

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

Testing the Model

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

```
from keras.utils.image_utils import load_img
from tensorflow.keras.utils import img_to_array
#loading of the image
img =load_img(r'/content/drive/MyDrive/Colab Notebooks/Sample Images/Test3.jpg',grayscale=
```

```
#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 21ms/step
array([2])
```

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

```
'ORANGE'
```

Colab paid products - Cancel contracts here

✓ 0s completed at 2:37 PM