Sprint-1

Model Building

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/1jzDjV7jYcIzIlieagaJdubMJ3YeLsry1/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

Data Collection

Download the dataset here

```
# Unzipping the dataset
```

```
!unzip '/content/Dataset.zip'   inflating: Dataset/TRAIN_SET/WATERMELON/r_288_100.jpg
    inflating: Dataset/TRAIN_SET/WATERMELON/r_289_100.jpg     inflating:
    Dataset/TRAIN_SET/WATERMELON/r_28_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_290_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_291_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_292_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_293_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_294_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_295_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_296_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_297_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_298_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_299_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_29_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_2_100.jpg                  inflating:
    Dataset/TRAIN_SET/WATERMELON/r_300_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_301_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_302_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_303_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_304_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_305_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_306_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_307_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_308_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_309_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_30_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_310_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_311_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_312_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_313_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_314_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_315_100.jpg                inflating:
    Dataset/TRAIN_SET/WATERMELON/r_31_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_32_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_33_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_34_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_35_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_36_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_37_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_38_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_39_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_3_100.jpg                  inflating:
    Dataset/TRAIN_SET/WATERMELON/r_40_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_41_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_42_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_43_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_44_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_45_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_46_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_4_100.jpg                  inflating:
    Dataset/TRAIN_SET/WATERMELON/r_50_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_57_100.jpg                 inflating:
    Dataset/TRAIN_SET/WATERMELON/r_5_100.jpg                  inflating:
    Dataset/TRAIN_SET/WATERMELON/r_6_100.jpg                  inflating:
    Dataset/TRAIN_SET/WATERMELON/r_7_100.jpg                  inflating:
    Dataset/TRAIN_SET/WATERMELON/r_81_100.jpg                 inflating:
       Dataset/TRAIN_SET/WATERMELON/r_8_100.jpg                                          inflating:
       Dataset/TRAIN_SET/WATERMELON/r_9_100.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
     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

1. 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,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
                                                            0
      max_pooling2d (MaxPooling2D (None, 31, 31, 32) )

      conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248
                                                            0
      max_pooling2d_1 (MaxPooling (None, 14, 14, 32) 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
```