Date : 07.10.2022

Name : Y.Anjaneyulu

Roll No :MECR19EC162

Assignment 3 - Build CNN Model For Classification Of Flowers

- Unzip dataset

```
!unzip '/content/Flowers-Dataset.zip'
```

```
Archive: /content/Flowers-Dataset.zip
  inflating:
  flowers/daisy/100080576_f52e8ee070_n.jp
  g inflating:
  flowers/daisy/10140303196_b88d3d6cec.jp
  g inflating:
  flowers/daisy/10172379554_b296050f82_n.jp
  g inflating:
  flowers/daisy/10172567486_2748826a8b.jpg
  inflating:
  flowers/daisy/10172636503_21bededa75_n.jp
  g inflating:
  flowers/daisy/102841525_bd6628ae3c.jpg inflating:
  flowers/daisy/10300722094_28fa978807_n.jpg inflating:
  flowers/daisy/1031799732_e7f4008c03.jpg inflating:
  flowers/daisy/10391248763_1d16681106_n.jp
  g inflating:
  flowers/daisy/10437754174_22ec990b77_m.jp
  g inflating:
  flowers/daisy/10437770546_8bb6f7bdd3_m.jp
  g inflating:
  flowers/daisy/10437929963_bc13eebe0c.jpg
  inflating:
  flowers/daisy/10466290366_cc72e33532.jp
  g inflating:
  flowers/daisy/10466558316_a7198b87e2.jp
  g inflating:
```

```
        flowers/daisy/10555749515_13a12a026e.jp
        g  inflating:
        flowers/daisy/10555815624_dc211569b0.jp
        g inflating:
        flowers/daisy/10555826524_423eb8bf71_n.jp
        g  inflating:
        flowers/daisy/10559679065_50d2b16f6d.jpg
        inflating:
        flowers/daisy/105806915_a9c13e2106_n.jp
        g  inflating:
        flowers/daisy/10712722853_5632165b04.jp
        g  inflating:
        flowers/daisy/107592979_aaa9cdfe78_m.jp
        g inflating:
        flowers/daisy/10770585085_4742b9dac3_n.jp
        g  inflating:
        flowers/daisy/10841136265_af473efc60.jpg inflating:
        flowers/daisy/10993710036_2033222c91.jpg inflating:
        flowers/daisy/10993818044_4c19b86c82.jp
        g  inflating:
        flowers/daisy/10994032453_ac7f8d9e2e.jp
        g  inflating:
        flowers/daisy/11023214096_b5b39fab08.jp
        g inflating:
        flowers/daisy/11023272144_fce94401f2_m.jp
        g  inflating:
        flowers/daisy/11023277956_8980d53169_m.jp
        g  inflating:
        flowers/daisy/11124324295_503f3a0804.jpg
        inflating:
        flowers/daisy/1140299375_3aa7024466.jpg
        inflating:
        flowers/daisy/11439894966_dca877f0cd.jp
        g inflating:
        flowers/daisy/1150395827_6f94a5c6e4_n.jp
        g inflating:
        flowers/daisy/11642632_1e7627a2cc.jpg
        inflating: flowers/daisy/11834945233_a53b7a92ac_m.jpg inflating:
        flowers/daisy/11870378973_2ec1919f12.jpg
inflating:
        flowers/daisy/11891885265_ccefec7284_n.jp
        g  inflating:
        flowers/daisy/12193032636_b50ae7db35_n.jp
        g  inflating:
        flowers/daisy/12348343085_d4c396e5b5_m.jp
        g  inflating:
        flowers/daisy/12585131704_0f64b17059_m.jp
        g  inflating:
```

```
flowers/daisy/12601254324_3cb62c254a_m.jp
g  inflating:
flowers/daisy/1265350143_6e2b276ec9.jpg
inflating:
flowers/daisy/12701063955_4840594ea6_n.jp
g inflating:
flowers/daisy/1285423653_18926dc2c8_n.jpg
inflating:
flowers/daisy/1286274236_1d7ac84efb_n.jpg
inflating:
flowers/daisy/12891819633_e4c82b51e8.jpg
inflating:
flowers/daisy/1299501272_59d9da5510_n.jp
g  inflating:
flowers/daisy/1306119996_ab8ae14d72_n.jp
g  inflating:
flowers/daisy/1314069875_da8dc023c6_m.jp
g  inflating:
flowers/daisy/1342002397_9503c97b49.jpg
inflating:
flowers/daisy/134409839_71069a95d1_m.jpg
inflating:
flowers/daisy/1344985627_c3115e2d71_n.jp
g inflating:
flowers/daisy/13491959645_2cd9df44d6_n.jp
g inflating:
flowers/daisy/1354396826_2868631432_m.jpg
inflating:
flowers/daisy/1355787476_32e9f2a30b.jpg
inflating:
flowers/daisy/13583238844_573df2de8e_m.jp
g inflating: flowers/daisy/1374193928
a52320eafa.jpg
```

Importing Necessary Libraries

```
import warnings
warnings.filterwarnings("ignore")

import numpy as np import matplotlib.pyplot as
plt import pandas as pd from
tensorflow.keras.models import Sequential
from tensorflow.keras.layers import
Dense,Activation,Dropout,Conv2D,Flatten,MaxPool2D,Resh
from     tensorflow.keras.applications.resnet50     import
```

```
ResNet50 from tensorflow.keras.applications.resnet50
import preprocess_input from
tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import
ImageDataGenerator,load_img,img_to_array from tensorflow.keras.callbacks
import EarlyStopping, ReduceLROnPlateau
```

- Image Augumentation

Dataset consist of 5 classes.

Daisy - European Species of Aster family.

Sunflower - Identified as the genus of Helianthus.

Tulip - It belongs to the species of spring blooming geophytes.

Rose - It belongs to the family of rosaceae.

Dandelion - Indentifies as the genus of Asterceae.

```
path = 'flowers/'
train_data_gen = ImageDataGenerator(rescale = 1./255,
                            shear_range =
                            0.2, zoom_range =
                            0.2, horizontal_f
                            lip = True,
                            validation_s plit
                            = 0.30)
test_data_gen = ImageDataGenerator(rescale = 1./255,validation_split = 0.30)
training_set = train_data_gen.flow_from_directory(path, target_s ize=(64, 64),
batch_si ze=100, class_mode='c ategorical', shuffle=True, color_mode='rgb',
subset = 'training')

testing_set = test_data_gen.flow_from_directory(path, target_s ize=(64,
                                64), batch_si ze=100, class_mode='c
                                        ategorical', shuffle=True,
                            color_mode='rgb', subset = 'validation')
```

```
Found 3024 images belonging
to 5 classes. Found 1293
images belonging to 5
classes.
```

- Create the model

```
model = Sequential()
```

- Add Layers (Convolution,MaxPooling,Flatten,Dense- Hidden Layers,Output)

```
#convolution and Pooling layer 1
model.add(Conv2D(filters=48,kernel_size=3,activation='relu',input_shape
=(64,64,3)))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Dropout(0.2))
#convolution and Pooling layer 2
model.add(Conv2D(filters=32,kernel_size=3,activat
ion='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
model.add(Dropout(0.2))
#Flattenin
g the
images
model.add(
Flatten())
#Fully Connected layers
model.add(Dense(64,activat
ion='relu'))
model.add(Dropout(0.2))
model.add(Dense(5,activation

='softmax')) model.summary()
```

```
Model: "sequential"

 Layer (type)                Output Shape              Param #
=================================================================
```

conv2d (Conv2D)	(None, 62, 62, 48)	1344
max_pooling2d (MaxPooling2D	(None, 31, 31, 48)	0

)				
dropout (Dropout)	(None, 31, 31, 48)			0
conv2d_1 (Conv2D)	(None, 29, 29, 32)			13856
max_pooling2d_1 (MaxPooling 2D)	(None, 14, 14, 32)			0
dropout_1 (Dropout)	(None,	14, 14,	32)	0
flatten (Flatten)	(None,	6272)		0
dense (Dense)	(None,	64)		401472
dropout_2 (Dropout)	(None,	64)		0
dense_1 (Dense)	(None,	5)		325

```
=================================================================
Total params: 416,997
Trainable params: 416,997
Non-trainable params: 0
```

- Compiling the Model

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accura
cy'])
```

- Fitting the Model

```
early_stop = EarlyStopping(monitor='val_accuracy',
                           patience=5,verbose=1,
                           mode='auto')

lr = ReduceLROnPlateau(monitor='val_accuracy',
                       factor=0.2
                       ,patience=
                       5,
                       min_lr=0.0
                       0001)
```

```
callback = [early_stop,lr]
```

Training the Model

```
result = model.fit(x=training_set, validation_data=testing_set, epochs=10)
```

Epoch 31/31	1/10 [==============================]	- 30s	966ms/step	- loss:	0.7625	- accuracy: 0
Epoch 31/31	2/10 [==============================]	- 30s	969ms/step	• loss :	0.7454	• accuracy : 0
Epoch 31/31	3/10 [==============================]	- 31s	985ms/step	• loss :	0.7348	• accuracy : 0
1/1 Epoch 31/31	4/10 [==============================]	- 30s	968ms/step	- loss:	0.7144	- accuracy: 0
Epoch 31/31	5/10 [==============================]	- 31s	992ms/step	- loss:	0.7233	- accuracy: 0
1/1 Epoch 31/31	6/10 [==============================]	- 32s	1s/step - loss: 0.7017 - accuracy: 0.73			
Epoch 31/31	7/10 [==============================]	- 30s	963ms/step	- loss:	0.6715	- accuracy: 0
Epoch 31/31	8/10 [==============================]	- 31s	978ms/step	- loss:	0.6512	- accuracy: 0

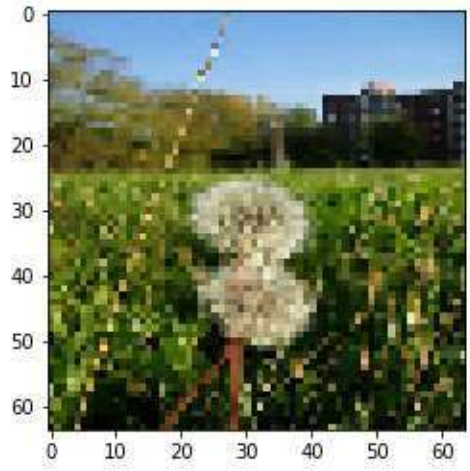
Epoch 9/10							
31/31 [==============================] 10/10	[==============================]	- 31s	982ms/step	• loss :	0.6711	• accuracy :	0
Epoch 31/31	[==============================]	- 30s	974ms/step	• loss :	0.6481	• accuracy :	0

Loss and Accuracy check using plot

```
 #plot the loss plt.plot(result.history['loss'],
label='train loss')
plt.plot(result.history['val_loss'], label='val
loss') plt.legend() plt.show()
 # plot the accuracy
 plt.plot(result.history['accuracy'],
 label='train acc')
 plt.plot(result.history['val_accuracy']
 , label='val acc') plt.legend()
 plt.show()

training_set.class_indices
 classes =
 ['Daisy','Dandelion','Rose','Sunflower','Tulip
 '] def testing(img):
     img =
     image.load_img(img,target_size=(6
     4,64)) x =
     image.img_to_array(img) x =
     np.expand_dims(x,axis=0) pred =
     np.argmax(model.predict(x))
     return print("Predicted class as:",classes[pred])

 def img_show(img):
     img1 =
     image.load_img(img,target_size=(64
     ,64)) plt.imshow(img1)

#test1
 img_show('/content/flowers/sunflower/12471443383_b71e7a
 7480_m.jpg')
 testing('/content/flowers/sunflower/12471443383_b71e7a7
 480_m.jpg')
      Predicted class as: Sunflower
```

```
#test3
img_show('/content/flowers/dandelion/2116997627_30fed8
4e53_m.jpg')
testing('/content/flowers/dandelion/2116997627_30fed84
e53_m.jpg')
```

```
Predicted class as: Daisy
```




```
#test4
img_show('/content/flowers/daisy/1314069875_da8dc
0
23c6_m.jpg')
testing('/content/flowers/daisy/1314069875_da8dc02
3c6_m.jpg')
```

```
Predicted class as: Daisy
```

Conclusion:

The dataset has about 4317 images from 5 different classes.

- Each classes have more than 500 images for training the data.
- 30% of the data taken for validation.
- The accuracy of the model is around 80%.
- The validation accuracy is around 70%.
- The model is built with 2 layered convolutional network considering 1344 trainable parameters.

- Testing the model with unknown images gives 95% accuracy.