Name:Prathap S

RollNo:611219106058

Date:08\10\2022

Assignment 3

1.Download the data set /content/Flowers-Dataset.zip

importing 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,Reshape
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
```

DataSet Augmentation

1.dataset consists of 5 different classes

2.daisy

3.dandelion

4.rose

5.sunflower

6.tulip

Unzip the dataset

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

```
Mounted at /content/gdrive
```

```
!unzip gdrive/MyDrive/Flowers-Dataset.zip
```

```
pictures = 'flowers/'
```

```
train_data = ImageDataGenerator(rescale = 1./255,
                                shear_range = 0.2,
                                zoom_range = 0.2,
                                horizontal_flip = True,
                                validation_split = 0.30)
test_data = ImageDataGenerator(rescale = 1./255,validation_split = 0.30)
```

```
training_set = train_data.flow_from_directory(pictures,
                                              target_size=(64,64),
                                              batch_size=100,
                                              class_mode='categorical',
                                              shuffle=True,
                                              color_mode='rgb',
                                              subset = 'training')

testing_set = test_data.flow_from_directory(pictures,
                                              target_size=(64,64),
                                              batch_size=100,
                                              class_mode='categorical',
                                              shuffle=True,
                                              color_mode='rgb',
                                              subset = 'validation')
```

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

Model Build Using CNN

1.Create the model

```
Model = Sequential()
```

```
#convolution layer 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(0.4))
```

```
#convolution layer and Pooling layer 2
Model.add(Conv2D(filters=32,kernel_size=3,activation='relu'))
Model.add(MaxPool2D(pool_size=2,strides=2))
Model.add(Dropout(0.4))

#Flattening the images
Model.add(Flatten())

#Fully Connected layers
Model.add(Dense(64,activation='relu'))
Model.add(Dropout(0.4))
Model.add(Dense(5,activation='softmax'))
```

```
Model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 62, 62, 48)        1344

 max_pooling2d_8 (MaxPooling  (None, 31, 31, 48)       0
 2D)

 dropout_12 (Dropout)        (None, 31, 31, 48)        0

 conv2d_9 (Conv2D)           (None, 29, 29, 32)        13856

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

 dropout_13 (Dropout)        (None, 14, 14, 32)        0

 flatten_4 (Flatten)         (None, 6272)              0

 dense_8 (Dense)             (None, 64)                401472

 dropout_14 (Dropout)        (None, 64)                0

 dense_9 (Dense)             (None, 5)                 325

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

2.Compile the model

```
Model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

3.

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

lr = ReduceLROnPlateau(monitor='val_accuracy',
                       factor=0.2,patience=5,
                       min_lr=0.00001)

callback = [early_stop,lr]
```

4.Training the models

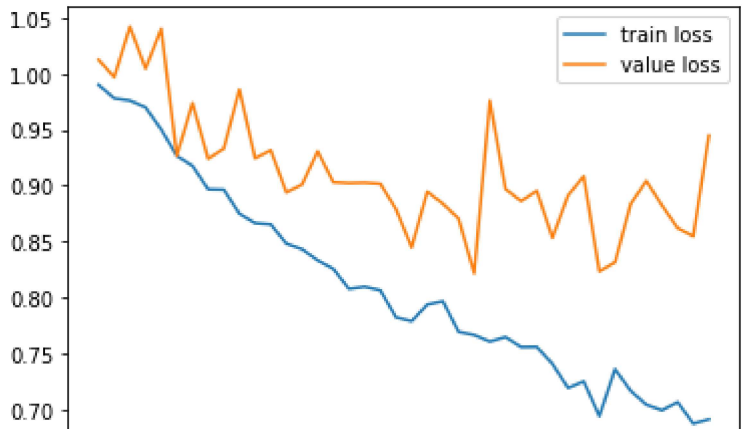
```
Result = Model.fit(x=training_set, validation_data=testing_set, epochs=40)
```

```
Epoch 1/40
31/31 [==============================] - 33s 1s/step - loss: 0.9899 - accuracy: 0.608
Epoch 2/40
31/31 [==============================] - 32s 1s/step - loss: 0.9779 - accuracy: 0.619
Epoch 3/40
31/31 [==============================] - 40s 1s/step - loss: 0.9759 - accuracy: 0.620
Epoch 4/40
31/31 [==============================] - 40s 1s/step - loss: 0.9698 - accuracy: 0.609
Epoch 5/40
31/31 [==============================] - 37s 1s/step - loss: 0.9501 - accuracy: 0.625
Epoch 6/40
31/31 [==============================] - 33s 1s/step - loss: 0.9265 - accuracy: 0.634
Epoch 7/40
31/31 [==============================] - 36s 1s/step - loss: 0.9176 - accuracy: 0.636
Epoch 8/40
31/31 [==============================] - 32s 1s/step - loss: 0.8966 - accuracy: 0.650
Epoch 9/40
31/31 [==============================] - 36s 1s/step - loss: 0.8963 - accuracy: 0.645
Epoch 10/40
31/31 [==============================] - 37s 1s/step - loss: 0.8747 - accuracy: 0.657
Epoch 11/40
31/31 [==============================] - 35s 1s/step - loss: 0.8663 - accuracy: 0.674
Epoch 12/40
31/31 [==============================] - 33s 1s/step - loss: 0.8652 - accuracy: 0.664
Epoch 13/40
31/31 [==============================] - 33s 1s/step - loss: 0.8480 - accuracy: 0.675
Epoch 14/40
31/31 [==============================] - 33s 1s/step - loss: 0.8431 - accuracy: 0.672
Epoch 15/40
31/31 [==============================] - 33s 1s/step - loss: 0.8331 - accuracy: 0.672
Epoch 16/40
31/31 [==============================] - 33s 1s/step - loss: 0.8254 - accuracy: 0.670
Epoch 17/40
31/31 [==============================] - 33s 1s/step - loss: 0.8075 - accuracy: 0.682
Epoch 18/40
31/31 [==============================] - 33s 1s/step - loss: 0.8095 - accuracy: 0.681
```

```
Epoch 19/40
31/31 [==============================] - 32s 1s/step - loss: 0.8062 - accuracy: 0.682
Epoch 20/40
31/31 [==============================] - 33s 1s/step - loss: 0.7821 - accuracy: 0.693
Epoch 21/40
31/31 [==============================] - 33s 1s/step - loss: 0.7786 - accuracy: 0.698
Epoch 22/40
31/31 [==============================] - 33s 1s/step - loss: 0.7934 - accuracy: 0.696
Epoch 23/40
31/31 [==============================] - 33s 1s/step - loss: 0.7964 - accuracy: 0.700
Epoch 24/40
31/31 [==============================] - 33s 1s/step - loss: 0.7691 - accuracy: 0.703
Epoch 25/40
31/31 [==============================] - 33s 1s/step - loss: 0.7663 - accuracy: 0.710
Epoch 26/40
31/31 [==============================] - 32s 1s/step - loss: 0.7603 - accuracy: 0.703
Epoch 27/40
31/31 [==============================] - 33s 1s/step - loss: 0.7644 - accuracy: 0.702
Epoch 28/40
31/31 [==============================] - 33s 1s/step - loss: 0.7554 - accuracy: 0.706
Epoch 29/40
```

5.Plot loss and accuracy

```
#Loss
plt.plot(Result.history['loss'], label='train loss')
plt.plot(Result.history['val_loss'], label='value loss')
plt.legend()
plt.show()

#Accuracy
plt.plot(Result.history['accuracy'], label='train accuracy')
plt.plot(Result.history['val_accuracy'], label='value accuracy')
plt.legend()
plt.show()
```



6.Save the model


```
Model.save('flower.h5')
```

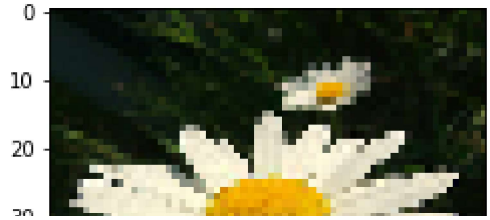
Test The Model

```
training_set.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

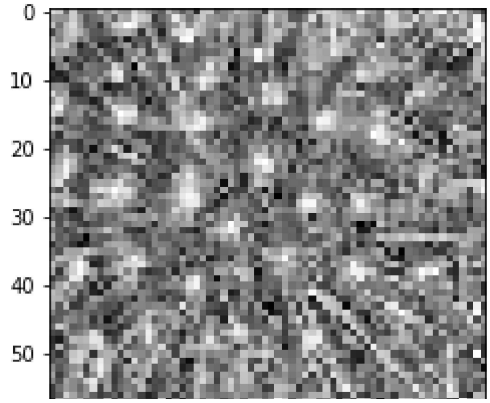
```
classes = ['Daisy','Dandelion','Rose','Sunflower','Tulip']
def testing(img):
    img = image.load_img(img,target_size=(64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x,axis=0)
    pred = np.argmax(model.predict(x))
    return print("Predicted class as:",classes[pred])

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


#Image1
img_show('/content/flowers/daisy/162362897_1d21b70621_m.jpg')
testing('/content/flowers/daisy/162362897_1d21b70621_m.jpg')
```

```
Predicted class as: Daisy
```



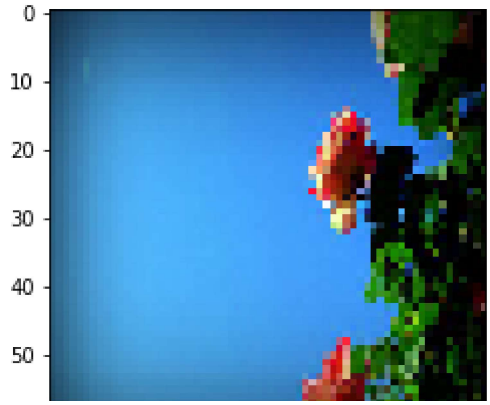
```
#Image2
img_show('/content/flowers/dandelion/17570530696_6a497298ee_n.jpg')
testing('/content/flowers/dandelion/17570530696_6a497298ee_n.jpg')
```

```
Predicted class as: Daisy
```



```
#Image3
img_show('/content/flowers/rose/18490508225_0fc630e963_n.jpg')
testing('/content/flowers/rose/18490508225_0fc630e963_n.jpg')
```

```
Predicted class as: Daisy
```

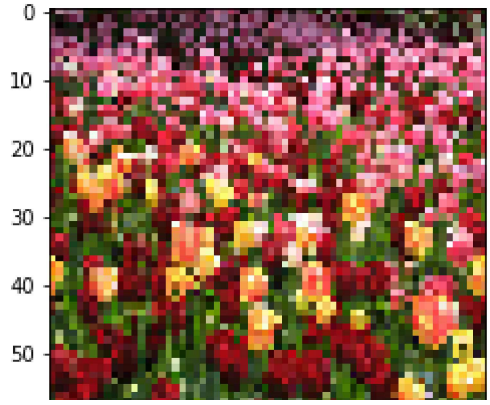


```
#Image4
img_show('/content/flowers/sunflower/14925398301_55a180f919_n.jpg')
testing('/content/flowers/sunflower/14925398301_55a180f919_n.jpg')
```

```
Predicted class as: Sunflower
```


```
#Sample5
img_show('/content/flowers/tulip/3502085373_edc2c36992_n.jpg')
testing('/content/flowers/tulip/3502085373_edc2c36992_n.jpg')
```

```
Predicted class as: Tulip
```



Colab paid products - Cancel contracts here