Assignment -3

Build CNN Model for Classification Of Flowers

Assignment Date	11 October 2022
Student Name	PL. VetriSelvan
Student Roll Number	912419104037
Maximum Marks	2 Marks

Task 1:

1. Download the Dataset : Dataset

Solution:

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

Build CNN model for Classification of Flowers

1. Download the Dataset dataset

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

     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# unzip the file
!unzip "/content/drive/MyDrive/Colab Notebooks/VetriSelvan/Flowers-Dataset.zip"
```

```
[3]  # unzip the file
     !unzip "/content/drive/MyDrive/Colab Notebooks/VetriSelvan/Flowers-Dataset.zip"
```

```
  inflating: flowers/tulip/8712270243_8512cf4fbd.jpg
  inflating: flowers/tulip/8712270665_57b5bda0a2_n.jpg
  inflating: flowers/tulip/8712282563_3819afb7bc.jpg
  inflating: flowers/tulip/8713357842_9964a93473_n.jpg
  inflating: flowers/tulip/8713387500_6a9138b41b_n.jpg
  inflating: flowers/tulip/8713388322_e5ae26263b_n.jpg
  inflating: flowers/tulip/8713389178_66bceb71a8_n.jpg
  inflating: flowers/tulip/8713390684_041148dd3e_n.jpg
  inflating: flowers/tulip/8713391394_4b679ea1e3_n.jpg
  inflating: flowers/tulip/8713392604_90631fb809_n.jpg
  inflating: flowers/tulip/8713394070_b24561b0a9.jpg
  inflating: flowers/tulip/8713396140_5af8136136.jpg
  inflating: flowers/tulip/8713397358_0505cc0176_n.jpg
  inflating: flowers/tulip/8713397694_bcbcbba2c2_n.jpg
  inflating: flowers/tulip/8713398114_bc96f1b624_n.jpg
  inflating: flowers/tulip/8713398614_88202e452e_n.jpg
  inflating: flowers/tulip/8713398906_28e59a225a_n.jpg
  inflating: flowers/tulip/8713407768_f880df361f.jpg
  inflating: flowers/tulip/8717900362_2aa508e9e5.jpg
  inflating: flowers/tulip/8722514702_7ecc68691c.jpg
  inflating: flowers/tulip/8723767533_9145dec4bd_n.jpg
  inflating: flowers/tulip/8729501081_b993185542_m.jpg
  inflating: flowers/tulip/8733586143_3139db6e9e_n.jpg
  inflating: flowers/tulip/8748266132_5298a91dcf_n.jpg
  inflating: flowers/tulip/8750288831_5e49a9f29b.jpg
  inflating: flowers/tulip/8757486380_90952c5377.jpg
  inflating: flowers/tulip/8758464923_75a5ffe320_n.jpg
  inflating: flowers/tulip/8758519201_16e8d2d781_n.jpg
  inflating: flowers/tulip/8759594528_2534c0ec65_n.jpg
  inflating: flowers/tulip/8759597778_7fca5d434b_n.jpg
```

Task 2:

2. Image Augmentation

Solution:

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

train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True, vertical_flip = True, zoom_range = 0.2)

x_train = train_datagen.flow_from_directory(r"/content/flowers", target_size = (64,64) , class_mode = "categorical", batch_size = 100)
"
```

2. Image Augmentation

```
[22] from tensorflow.keras.preprocessing.image import ImageDataGenerator

     train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True, vertical_flip = True, zoom_range = 0.2)

     x_train = train_datagen.flow_from_directory(r"/content/flowers", target_size = (64,64) , class_mode = "categorical", batch_size = 100)
```

```
Found 4317 images belonging to 5 classes.
```

Task 3:

3. Create Model

Solution:

```
from tensorflow.keras.models import Sequential
```

from tensorflow.keras.layers import
Convolution2D,MaxPooling2D,Flatten,Dense

model = Sequential()

3. Create Model

```
[23] from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense

     model = Sequential()
```

Task 4:

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

Solution:

model.add(Convolution2D(32, (3,3), activation = "relu", input_shape = (64,64,3)))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(300, activation = "relu"))
model.add(Dense(150, activation = "relu")) #multiple dense layers
model.add(Dense(5, activation = "softmax")) #output layer

4. Add the layers (Convolution, MaxPooling, Flatten, Dense-(HiddenLayers), Output)

```
[24] model.add(Convolution2D(32, (3,3), activation = "relu", input_shape = (64,64,3) ))
     model.add(MaxPooling2D(pool_size = (2,2)))
     model.add(Flatten())
     model.add(Dense(300, activation = "relu"))
     model.add(Dense(150, activation = "relu")) #multiple dense layers
     model.add(Dense(5, activation = "softmax")) #output layer
```

Task 5:

5. Compile The Model

Solution:

model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = "adam")
len(x_train)

5. Compile The Model

```
[25] model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = "adam")
     len(x_train)

     44
```

Task 6:

6. Fit The Model

Solution:

model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))

6. Fit The Model

```
[26] model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))

Epoch 1/15
44/44 [==============================] - 14s 299ms/step - loss: 1.4965 - accuracy: 0.3864
Epoch 2/15
44/44 [==============================] - 13s 295ms/step - loss: 1.1444 - accuracy: 0.5281
Epoch 3/15
44/44 [==============================] - 13s 294ms/step - loss: 1.0495 - accuracy: 0.5879
Epoch 4/15
44/44 [==============================] - 13s 293ms/step - loss: 0.9727 - accuracy: 0.6196
Epoch 5/15
44/44 [==============================] - 13s 292ms/step - loss: 0.9343 - accuracy: 0.6345
Epoch 6/15
44/44 [==============================] - 13s 290ms/step - loss: 0.8998 - accuracy: 0.6423
Epoch 7/15
44/44 [==============================] - 13s 291ms/step - loss: 0.8661 - accuracy: 0.6627
Epoch 8/15
44/44 [==============================] - 13s 299ms/step - loss: 0.8560 - accuracy: 0.6648
Epoch 9/15
44/44 [==============================] - 13s 292ms/step - loss: 0.8039 - accuracy: 0.6875
Epoch 10/15
44/44 [==============================] - 13s 300ms/step - loss: 0.7862 - accuracy: 0.6945
Epoch 11/15
44/44 [==============================] - 13s 292ms/step - loss: 0.7879 - accuracy: 0.6931
Epoch 12/15
44/44 [==============================] - 13s 290ms/step - loss: 0.7752 - accuracy: 0.7088
Epoch 13/15
44/44 [==============================] - 13s 292ms/step - loss: 0.7512 - accuracy: 0.7169
Epoch 14/15
44/44 [==============================] - 13s 288ms/step - loss: 0.7153 - accuracy: 0.7267
Epoch 15/15
44/44 [==============================] - 13s 290ms/step - loss: 0.7279 - accuracy: 0.7181
<keras.callbacks.History at 0x7f72c927a6d0>
```

model.summary()

```
[27] model.summary()

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 62, 62, 32)        896

 max_pooling2d_2 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 30752)             0

 dense_6 (Dense)             (None, 300)               9225900

 dense_7 (Dense)             (None, 150)               45150

 dense_8 (Dense)             (None, 5)                 755

=================================================================
Total params: 9,272,701
Trainable params: 9,272,701
Non-trainable params: 0
_________________________________________________________________
```

Task 7:

7. Save The Model

Solution:

```
model.save("flowers.h5")
```

7. Save The Model

```
[28] model.save("flowers.h5")
```

Task 8:

9. Test The Model

Solution:

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

model = load_model("/content/flowers.h5")
img = image.load_img("/content/Tulip-image.jpg", target_size = (64,64))
x = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
pred = model.predict(x)

labels = ['daisy','dandelion','roses','sunflowers','tulips']
print("Input image is")
img

print("Classification of Flower is:",labels[np.argmax(pred)])
```

8. Test The Model

```
[18] from tensorflow.keras.models import load_model
     from tensorflow.keras.preprocessing import image
     import numpy as np

     model = load_model("/content/flowers.h5")
     img = image.load_img("/content/Rose-image.jpg", target_size = (64,64))
     x = image.img_to_array(img)
     x = np.expand_dims(x,axis = 0)
     pred = model.predict(x)

     labels = ['daisy','dandelion','roses','sunflowers','tulips']
     print("Input Image is\n")
     img
```

```
Input Image is
```

```
print("Classification of Flower is:",labels[np.argmax(pred)])
```

```
Classification of Flower is: roses
```