Assignment-3

VirtualEye - Life Guard for Swimming Pools to Detect Active Drowning

Assignment Date	November 14, 2022
Student Name	SATHEESH RATHNAVEL
Student Roll Number	2127190801073
Maximum Marks	2 Marks

Question-1:

Download the dataset

Solution:

Download the given dataset in the given attached link.

Question-2:

Image Augmentation

Solution:

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

```
In [36]: #importing keras libraries

         #!pip install tensorflow
         from tensorflow.keras import losses
         from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
In [46]: #Splitting the folder into training, testing and validation

         !pip install split-folders
         import splitfolders
         splitfolders.ratio('C:/Users/santhosh/Desktop/assignment/Flowers-Dataset', output="C:/Users/santhosh/Desktop/assignment/output",

         Requirement already satisfied: split-folders in c:\users\santhosh\anaconda3\lib\site-packages (0.5.1)
         Copying files: 4317 files [00:08, 535.10 files/s]
```

```
In [47]: #Generate train data and test data using ImageDataGenerator function

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

```
In [48]: # Fetching xtrain and xtest for image classification

         xtrain = train_datagen.flow_from_directory('C:/Users/santhosh/Desktop/assignment/output/train',target_size=(220,220),class_mode=
         xtest = test_datagen.flow_from_directory('C:/Users/santhosh/Desktop/assignment/output/test',target_size=(220,220),class_mode= 'ca

         Found 3452 images belonging to 6 classes.
         Found 435 images belonging to 6 classes.
```

Question 3:

Create Model

Solution:

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

```
In [49]: #importing Sequential model and required libraries for classification using Convlutional Neural Networks
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

Question 4:

- **Add Layers (LSTM, Dense-(Hidden Layers), Output)**
- **Compile the Model**
- **Fit the Model**
- **Save The Model**
- **Test The Model**

Solution:

```
model = Sequential()
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(220,220,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

#This softmax activation is used for the classification of images into 5 classes
model.add(Dense(5,activation='softmax'))
model.compile(optimizer='adam',loss=losses.categorical_crossentropy,metrics=['accuracy'
])
#fit the model xtrain and validate using xtest
model.fit(xtrain, steps_per_epoch=len(xtrain), epochs=30,validation_data=xtest,
validation_steps=len(xtest))
#Save the model as flower-predict.h5
model.save('C:/Users/santhosh/Desktop/assignment/flower_predict.h5')
#Image classification part
import numpy as np
from tensorflow.keras.preprocessing import image
# Loading a sample image for testing the classifier
img =
image.load_img('C:/Users/santhosh/Desktop/assignment/output/val/daisy/10994032453_ac
7f8d9e2e.jpg',target_size=(220,220))
img #Output image
x= image.img_to_array(img) # Covert image date into array of rgb values
x
x=np.expand_dims(x,axis=0) #expand the dimension of the array
model.predict(x) # Predict the output of classification
op=['daisy','dandelion','rose','sunflower','tulip']
```

predicted output
pred = np.argmax(model.predict(x))
op[pred]

In [50]:
```
#Creating an instance of sequential model and add convolution layers into it

model = Sequential()
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(220,220,3)))
```

In [51]:
```
#Add a max pooling layer with a size of 2*2
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

#This softmax activation is used for the classification of images into 5 classes
model.add(Dense(5,activation='softmax'))
```

In [52]:
```
#Compile the model
model.compile(optimizer='adam',loss=losses.categorical_crossentropy,metrics=['accuracy'])
```

In [53]:
```
#fit the model xtrain and validate using xtest
model.fit(xtrain, steps_per_epoch=len(xtrain), epochs=30,validation_data=xtest, validation_steps=len(xtest))
```

In [55]:
```
#Save the model as flower-predict.h5
model.save('C:/Users/santhosh/Desktop/assignment/flower_predict.h5')
```

In [57]:
```
#Image classification part
import numpy as np
from tensorflow.keras.preprocessing import image
```

In [70]:
```
# Loading a sample image for testing the classifier
img = image.load_img('C:/Users/santhosh/Desktop/assignment/output/val/daisy/10994032453_ac7f8d9e2e.jpg',target_size=(220,220)
```

In [71]:
```
img #Output image
```

Out[71]:

In [72]:
```
x= image.img_to_array(img) # Covert image date into array of rgb values
```

In [73]:
```
x
```

Out[73]:
```
array([[[ 85.,  55.,  63.],
        [101.,  71.,  79.],
        [110.,  80.,  92.],
        ...,
        [124.,  87.,  95.],
        [110.,  78.,  93.],
        [ 89.,  65.,  79.]],

       [[ 92.,  68.,  68.],
        [ 91.,  71.,  72.],
        [106.,  81.,  84.],
        ...,
        [115.,  78.,  85.],
        [103.,  77.,  80.],
        [ 87.,  67.,  76.]],

       [[ 90.,  63.,  68.],
        [ 86.,  65.,  70.],
        [101.,  76.,  80.],
        ...,
        [108.,  73.,  79.],
        [ 98.,  70.,  84.],
        [ 84.,  63.,  68.]],

       ...,

       [[ 95., 127.,  80.],
        [106., 133.,  90.],
        [124., 159., 129.],
        ...,
        [ 55.,  60.,  30.],
        [ 77.,  63.,  60.],
        [102.,  90., 136.]],

           [[ 96., 128.,  79.],
            [105., 134.,  90.],
            [125., 160., 127.],
            ...,
            [ 58.,  67.,  48.],
            [ 84.,  79., 133.],
            [ 32.,  41.,  24.]],

           [[ 90., 122.,  75.],
            [ 93., 124.,  82.],
            [120., 156., 118.],
            ...,
            [ 33.,  33.,  23.],
            [ 46.,  43.,  26.],
            [100.,  88., 110.]]], dtype=float32)
```

In [74]:
```
x=np.expand_dims(x,axis=0) #expand the dimension of the array
```

In [75]:
```
model.predict(x) # Predict the output of classification
```

Out[75]:
```
array([[2.88144489e-16, 1.00000000e+00, 2.84125234e-26, 7.68427070e-14,
        1.24847975e-17]], dtype=float32)
```

In [76]:
```
op=['daisy','dandelion','rose','sunflower','tulip']
```

In [77]:
```
# Finding the maximum argument value of prediction and print the corresponding predicted output
pred = np.argmax(model.predict(x))
```

]:
```
op[pred]
```

:[27]:
```
'daisy'
```