

ASSIGNMENT 3

Assignment Date	27-09-2022
Student Name	P.Rajalekshmi
Student Roll.No	960519104062
Maximum Marks	2 Marks

```
1. import os
2. print(os.listdir('../input/flowers/flowers'))
3. # Ignore the warnings
4. import warnings
5. warnings.filterwarnings('always')
6. warnings.filterwarnings('ignore')
7.
8. # data visualisation and manipulation
9. import numpy as np
10. import pandas as pd
11. import matplotlib.pyplot as plt
12. from matplotlib import style
13. import seaborn as sns
14.
15. #configure
16. # sets matplotlib to inline and displays graphs below the corresponding cell.
17. %matplotlib inline
18. style.use('fivethirtyeight')
19. sns.set(style='whitegrid',color_codes=True)
20.
21. #model selection
22. from sklearn.model_selection import train_test_split
23. from sklearn.model_selection import KFold
24. from sklearn.metrics import
    accuracy_score,precision_score,recall_score,confusion_matrix,roc_curve,roc_auc_score
25. from sklearn.model_selection import GridSearchCV
26. from sklearn.preprocessing import LabelEncoder
27.
28. #preprocess.
29. from keras.preprocessing.image import ImageDataGenerator
30.
31. #dl librairaies
32. from keras import backend as K
33. from keras.models import Sequential
34. from keras.layers import Dense
35. from tensorflow.keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
```

```

36. #from keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
37. from tensorflow.keras.utils import to_categorical
38.
39. # specifically for cnn
40. from keras.layers import Dropout, Flatten,Activation
41. from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
42.
43. import tensorflow as tf
44. import random as rn
45.
46. # specifically for manipulating zipped images and getting numpy arrays of pixel values of
    images.
47. import cv2
48. import numpy as np
49. from tqdm import tqdm
50. import os
51. from random import shuffle
52. from zipfile import ZipFile
53. from PIL import Image
54.
55. X=[]
56. Z=[]
57. IMG_SIZE=150
58. FLOWER_DAISY_DIR='../input/flowers/flowers/daisy'
59. FLOWER_SUNFLOWER_DIR='../input/flowers/flowers/sunflower'
60. FLOWER_TULIP_DIR='../input/flowers/flowers/tulip'
61. FLOWER_DANDI_DIR='../input/flowers/flowers/dandelion'
62. FLOWER_ROSE_DIR='../input/flowers/flowers/rose'
63.
64. def assign_label(img,flower_type):
65.     return flower_type
66.
67. def make_train_data(flower_type,DIR):
68.     for img in tqdm(os.listdir(DIR)):
69.         label=assign_label(img,flower_type)
70.         path = os.path.join(DIR,img)
71.         img = cv2.imread(path,cv2.IMREAD_COLOR)
72.         img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
73.
74.         X.append(np.array(img))
75.         Z.append(str(label))
76.
77. make_train_data('Daisy',FLOWER_DAISY_DIR)
78. print(len(X))
79.
80. make_train_data('Sunflower',FLOWER_SUNFLOWER_DIR)
81. print(len(X))
82.

```

```

83. make_train_data('Tulip',FLOWER_TULIP_DIR)
84. print(len(X))
85.
86. make_train_data('Dandelion',FLOWER_DANDI_DIR)
87. print(len(X))
88.
89. make_train_data('Rose',FLOWER_ROSE_DIR)
90. print(len(X))
91.
92. fig,ax=plt.subplots(5,2)
93. fig.set_size_inches(15,15)
94. for i in range(5):
95.     for j in range (2):
96.         l=mn.randint(0,len(Z))
97.         ax[i,j].imshow(X[l])
98.         ax[i,j].set_title('Flower: '+Z[l])
99.
100.         plt.tight_layout()
101.
102.         le=LabelEncoder()
103.         Y=le.fit_transform(Z)
104.         Y=to_categorical(Y,5)
105.         X=np.array(X)
106.         X=X/255
107.
108.         x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,random_state=42)
109.
110.         np.random.seed(42)
111.         mn.seed(42)
112.         tf.set_random_seed(42)np.random.seed(42)
113.         mn.seed(42)
114.         tf.set_random_seed(42)
115.
116.         ## modelling starts using a CNN.
117.
118.         model = Sequential()
119.         model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',activation
            ='relu', input_shape = (150,150,3)))
120.         model.add(MaxPooling2D(pool_size=(2,2)))
121.
122.
123.         model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation
            ='relu'))
124.         model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
125.
126.
127.         model.add(Conv2D(filters =96, kernel_size = (3,3),padding = 'Same',activation
            ='relu'))

```

```

128.     model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
129.
130.     model.add(Conv2D(filters = 96, kernel_size = (3,3),padding = 'Same',activation
    ='relu'))
131.     model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
132.
133.     model.add(Flatten())
134.     model.add(Dense(512))
135.     model.add(Activation('relu'))
136.     model.add(Dense(5, activation = "softmax"))
137.
138.     batch_size=128
139.     epochs=50
140.
141.     from keras.callbacks import ReduceLROnPlateau
142.     red_lr= ReduceLROnPlateau(monitor='val_acc',patience=3,verbose=1,factor=0.1)
143.
144.     datagen = ImageDataGenerator(
145.         featurewise_center=False, # set input mean to 0 over the dataset
146.         samplewise_center=False, # set each sample mean to 0
147.         featurewise_std_normalization=False, # divide inputs by std of the dataset
148.         samplewise_std_normalization=False, # divide each input by its std
149.         zca_whitening=False, # apply ZCA whitening
150.         rotation_range=10, # randomly rotate images in the range (degrees, 0 to 180)
151.         zoom_range = 0.1, # Randomly zoom image
152.         width_shift_range=0.2, # randomly shift images horizontally (fraction of total
    width)
153.         height_shift_range=0.2, # randomly shift images vertically (fraction of total
    height)
154.         horizontal_flip=True, # randomly flip images
155.         vertical_flip=False) # randomly flip images
156.
157.
158.     datagen.fit(x_train)
159.
160.     model.compile(optimizer=Adam(lr=0.001),loss='categorical_crossentropy',metrics=['
    accuracy'])
161.     model.summary()
162.
163.     History = model.fit_generator(datagen.flow(x_train,y_train, batch_size=batch_size),
164.         epochs = epochs, validation_data = (x_test,y_test),
165.         verbose = 1, steps_per_epoch=x_train.shape[0] // batch_size)
166.     # model.fit(x_train,y_train,epochs=epochs,batch_size=batch_size,validation_data =
    (x_test,y_test))
167.
168.     plt.plot(History.history['loss'])
169.     plt.plot(History.history['val_loss'])
170.     plt.title('Model Loss')

```

```

171.     plt.ylabel('Loss')
172.     plt.xlabel('Epochs')
173.     plt.legend(['train', 'test'])
174.     plt.show()
175.
176.     plt.plot(History.history['acc'])
177.     plt.plot(History.history['val_acc'])
178.     plt.title('Model Accuracy')
179.     plt.ylabel('Accuracy')
180.     plt.xlabel('Epochs')
181.     plt.legend(['train', 'test'])
182.     plt.show()
183.
184.     # getting predictions on val set.
185.     pred=model.predict(x_test)
186.     pred_digits=np.argmax(pred,axis=1)
187.
188.     # now storing some properly as well as misclassified indexes'.
189.     i=0
190.     prop_class=[]
191.     mis_class=[]
192.
193.     for i in range(len(y_test)):
194.         if(np.argmax(y_test[i])==pred_digits[i]):
195.             prop_class.append(i)
196.             if(len(prop_class)==8):
197.                 break
198.
199.     i=0
200.     for i in range(len(y_test)):
201.         if(not np.argmax(y_test[i])==pred_digits[i]):
202.             mis_class.append(i)
203.             if(len(mis_class)==8):
204.                 break
205.
206.     warnings.filterwarnings('always')
207.     warnings.filterwarnings('ignore')
208.
209.     count=0
210.     fig,ax=plt.subplots(4,2)
211.     fig.set_size_inches(15,15)
212.     for i in range (4):
213.         for j in range (2):
214.             ax[i,j].imshow(x_test[prop_class[count]])
215.             ax[i,j].set_title("Predicted Flower :
"+str(le.inverse_transform([pred_digits[prop_class[count]]]))+"\n"+"Actual Flower :
"+str(le.inverse_transform(np.argmax([y_test[prop_class[count]]])))
216.             plt.tight_layout()

```

```
217.         count+=1
218.
219.     warnings.filterwarnings('always')
220.     warnings.filterwarnings('ignore')
221.
222.     count=0
223.     fig,ax=plt.subplots(4,2)
224.     fig.set_size_inches(15,15)
225.     for i in range (4):
226.         for j in range (2):
227.             ax[i,j].imshow(x_test[mis_class[count]])
228.             ax[i,j].set_title("Predicted Flower :
"+str(le.inverse_transform([pred_digits[mis_class[count]]]))+"\n"+"Actual Flower :
"+str(le.inverse_transform(np.argmax([y_test[mis_class[count]]]))))
229.             plt.tight_layout()
230.             count+=1
```