

Assignment -3

Assignment Date	22 October 2022
Team Id	PNT2022TMID25540
Project Name	Natural Intensity Analysis and Classification

#Import necessary libraries

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

#Image augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
train_datagen =  
ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical  
_flip=True) test_datagen =  
  
ImageDataGenerator(rescale=1./255)
```

```
#Import necessary libraries  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
from tensorflow.keras.layers import Convolution2D  
from tensorflow.keras.layers import MaxPooling2D  
from tensorflow.keras.layers import Flatten  
  
#Image augmentation  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical_flip=True)  
test_datagen = ImageDataGenerator(rescale=1./255)
```

#data set

```
x_train =  
train_datagen.flow_from_directory(r"E:\Flowers\Training",target_size=(128,128),batch_size=32,class_  
s_mode="categorical")  
  
x_test =  
test_datagen.flow_from_directory(r"E:\Flowers\Testing",target_size=(128,128),batch_size=32,class_  
_mode="categorical") x_train.class_indices model = Sequential()
```

```
x_train = train_datagen.flow_from_directory(r"E:\Flowers\Training",target_size=(128,128),batch_size=32,class_mode="categorical")  
x_test = test_datagen.flow_from_directory(r"E:\Flowers\Testing",target_size=(128,128),batch_size=32,class_mode="categorical")  
x_train.class_indices
```

Found 3003 images belonging to 5 classes.

Found 1325 images belonging to 5 classes.

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

#Add layers

```
#Convolution layer model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

```
#Maxpooling layer model.add(MaxPooling2D(pool_size=(2,2)))
```

```
#flatten layer model.add(Flatten()) #hidden layer
```

```
model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))
```

```
model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))
```

```
model.add(Dense(units=5,kernel_initializer="random_uniform",activation="softmax"))
```

```
model.summary()
```

```

model = Sequential()
#Add layers
#Convolution layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
#Maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
#flatten layer
model.add(Flatten())
#hidden layer
model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))
model.add(Dense(units=5,kernel_initializer="random_uniform",activation="softmax"))
model.summary()

```

[4]

... Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
flatten (Flatten)	(None, 127008)	0
dense (Dense)	(None, 300)	38102700
dense_1 (Dense)	(None, 200)	60200
dense_2 (Dense)	(None, 5)	1005

```

=====
Total params: 38,164,801
Trainable params: 38,164,801
Non-trainable params: 0
=====

```

#compile the model

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

#Fit the model

```
model.fit_generator(x_train,steps_per_epoch=75,epochs=15,validation_data=x_test,validation_steps=80)
```

```
#compile the model
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])
#fit the model
model.fit_generator(x_train,steps_per_epoch=75,epochs=15,validation_data=x_test,validation_steps=80)

[4] Python
c:\Users\hp\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: UserWarning: "Model.fit_generator" is deprecated and will be removed in a future version. Please use "Model.fit",
which supports generators.
    after removing the cwd from sys.path.

Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/15
75/75 [=====] - ETA: 0s - loss: 1.0726 - accuracy: 0.5791WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or
generator can generate at least 'steps_per_epoch * epochs' batches (in this case, 80 batches). You may need to use the repeat() function when building your dataset.
75/75 [=====] - 83s 1s/step - loss: 1.0726 - accuracy: 0.5791 - val_loss: 1.2372 - val_accuracy: 0.5049
Epoch 2/15
75/75 [=====] - 68s 906ms/step - loss: 0.9907 - accuracy: 0.6125
Epoch 3/15
75/75 [=====] - 69s 917ms/step - loss: 0.8981 - accuracy: 0.6489
Epoch 4/15
75/75 [=====] - 70s 922ms/step - loss: 0.8850 - accuracy: 0.6522
Epoch 5/15
75/75 [=====] - 73s 962ms/step - loss: 0.8177 - accuracy: 0.6789
Epoch 6/15
75/75 [=====] - 75s 997ms/step - loss: 0.8101 - accuracy: 0.6917
Epoch 7/15
75/75 [=====] - 73s 966ms/step - loss: 0.8099 - accuracy: 0.6868
Epoch 8/15
75/75 [=====] - 72s 957ms/step - loss: 0.7574 - accuracy: 0.7229
Epoch 9/15
75/75 [=====] - 70s 926ms/step - loss: 0.7146 - accuracy: 0.7215
Epoch 10/15
Epoch 5/15
75/75 [=====] - 73s 962ms/step - loss: 0.8177 - accuracy: 0.6789
Epoch 6/15
75/75 [=====] - 75s 997ms/step - loss: 0.8101 - accuracy: 0.6917
Epoch 7/15
75/75 [=====] - 73s 966ms/step - loss: 0.8099 - accuracy: 0.6868
Epoch 8/15
75/75 [=====] - 72s 957ms/step - loss: 0.7574 - accuracy: 0.7229
Epoch 9/15
75/75 [=====] - 70s 926ms/step - loss: 0.7146 - accuracy: 0.7215
Epoch 10/15
75/75 [=====] - 69s 911ms/step - loss: 0.6867 - accuracy: 0.7446
Epoch 11/15
75/75 [=====] - 69s 920ms/step - loss: 0.6735 - accuracy: 0.7404
Epoch 12/15
75/75 [=====] - 70s 931ms/step - loss: 0.6735 - accuracy: 0.7562
...
Epoch 14/15
75/75 [=====] - 75s 995ms/step - loss: 0.6296 - accuracy: 0.7724
Epoch 15/15
75/75 [=====] - 75s 988ms/step - loss: 0.6024 - accuracy: 0.7775

<keras.callbacks.History at 0x218dffffd2b0>
```

#Save the model

model.save("flower.h5") from

tensorflow.keras.models import load_model from

tensorflow.keras.preprocessing import image

import numpy as np model =

load_model("Flower.h5")

```
#Save the model
model.save("flower.h5")

[7]
```

#Test the model:

```
img = image.load_img(r"C:\Users\hp\Downloads\rose.jpg",target_size=(128,128))
```

```
img type(img) x = image.img_to_array(img)
```

```
x
```

```
x.shape x =
```

```
np.expand_dims(x,axis=0)
```

```
x.shape
```

```
pred_prob = model.predict(x) pred_prob
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
model = load_model("Flower.h5")

#Testing with the image
img = image.load_img(r"C:\Users\hp\Downloads\rose.jpg",target_size=(128,128))
img
type(img)

PIL.Image.Image

x = image.img_to_array(img)
x
x.shape
x = np.expand_dims(x,axis=0)
x.shape

(1, 128, 128, 3)

pred_prob = model.predict(x)
pred_prob

array([[0., 0., 1., 0., 0.]], dtype=float32)
```

```
class_name = ["daisy","dandelion","rose","sunflower","tulip"]
```

```
pred_id = pred_prob.argmax(axis=1)[0] pred_id
```

```
print("Predicted flower is",str(class_name[pred_id]))
```

```
class_name = ["daisy", "dandelion", "rose", "sunflower", "tulip"]
```

```
pred_id = pred_prob.argmax(axis=1)[0]  
pred_id  
print("Predicted flower is", str(class_name[pred_id]))
```

```
Predicted flower is rose
```