ARTIFICIAL INTELLIGENCE

NATURAL DISASTER INTENSITY ANALYSIS AND CLASSIFICATION

ASSIGNMENT 3

Input:

```
from zipfile import ZipFile
file_name = "Flowers.zip"

with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print('Done')
```

Output:

```
Done
```

Input:

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

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

test_datagen=ImageDataGenerator(rescale=1./255)


x_train=train_datagen.flow_from_directory(r"/content/flowers",target_size=(64,64), class_mode="categorical",batch_size=24)
```

Output:

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

Input:

```
x_test=test_datagen.flow_from_directory(r"/content/flowers",target_size=(64,64), class_mode="categorical",batch_size=24)
```

Output:

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

Input:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
model=Sequential()
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(5,activation="softmax"))
```

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

```
len(x_train)
```

Output:

```
180
```

Input:

```
model.fit(x_train,epochs=5,validation_data=x_test,steps_per_epoch=len(x_train),validation_steps=len(x_test))
```

Output:

```
Epoch 1/5
180/180 [==============================] - 53s 289ms/step - loss: 1.3252 -
accuracy: 0.4677 - val_loss: 1.2957 - val_accuracy: 0.4971
Epoch 2/5
180/180 [==============================] - 51s 285ms/step - loss: 1.0926 -
accuracy: 0.5617 - val_loss: 1.0207 - val_accuracy: 0.6032
Epoch 3/5
180/180 [==============================] - 53s 292ms/step - loss: 1.0192 -
accuracy: 0.5956 - val_loss: 0.9547 - val_accuracy: 0.6421
Epoch 4/5
180/180 [==============================] - 51s 285ms/step - loss: 0.9546 -
accuracy: 0.6278 - val_loss: 0.9099 - val_accuracy: 0.6604
Epoch 5/5
180/180 [==============================] - 51s 286ms/step - loss: 0.9053 -
accuracy: 0.6532 - val_loss: 1.0180 - val_accuracy: 0.6176
```

Input:

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

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

```
model=load_model("/content/daisy.h5")
```

```
img=image.load_img("/content/flowers/daisy/100080576_f52e8ee070_n.jpg",target_size=(64,64))
```

```
img
```

Output:

Input:

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

Output:

```
array([[[141., 141., 139.],
        [149., 149., 149.],
        [152., 152., 154.],
        ...,
        [162., 161., 166.],
        [154., 154., 152.],
        [153., 153., 153.]],

       [[136., 135., 131.],
        [146., 145., 143.],
        [169., 168., 174.],
        ...,
        [159., 158., 163.],
        [155., 155., 153.],
        [149., 149., 149.]],

       [[125., 125., 117.],
        [138., 140., 137.],
        [152., 152., 152.],
        ...,
        [156., 156., 156.],
        [157., 157., 155.],
        [143., 142., 140.]],

       ...,

       [[ 41.,  44.,  23.],
        [ 43.,  46.,  25.],
        [ 49.,  51.,  37.],
        ...,
        [128., 124., 121.],
        [125., 121., 118.],
        [125., 122., 117.]],

       [[ 43.,  46.,  25.],
        [ 43.,  46.,  25.],
        [ 54.,  55.,  37.],
        ...,
        [130., 126., 125.],
        [129., 125., 124.],
        [127., 123., 122.]],

       [[ 44.,  47.,  26.],
        [ 45.,  48.,  27.],
        [ 53.,  55.,  34.],
...,
        [137., 133., 132.],
        [133., 129., 128.],
        [130., 126., 125.]]], dtype=float32)
```

Input:

```
x.ndim
```

Output:

```
3
```

Input:

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

Output:

```
4
```

Input:

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

Output:

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

Input:

```
labels=["daisy","dandelion","rose","sunflower","tulip"]
np.argmax(pred)
```

Output:

```
0
```

Input:

```
labels[4]
```

Output:

```
'tulip'
```

Input:

```
labels[np.argmax(pred)]
```

Output:

```
'daisy'
```