Assignment -3
Python Programming

Assignment Date	06 October 2022
Student Name	Ms.Ruthradevi M
Student Roll Number	812419104710
Maximum Marks	2 Marks

Question-1:

1.DOWNLOAD THE DATASET

Solution:

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

OUTPUT

```
Mounted at /content/drive
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPool2D,Flatten,Dense

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

Question-2:

2.IMAGE AUGUMENTATION

Solution:

```
train_datagen =
ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_f
lip=True,vertical_flip=True)

test_datagen=ImageDataGenerator(rescale=1./255)

x_train =
train_datagen.flow_from_directory(r"/content/drive/MyDrive/dataset/Training",
target_size=(64,64),batch_size=32,class_mode="categorical")
```

OUTPUT

```
Found 1238 images belonging to 4 classes.
```

```
#Load your images data

x_test =
```

```
test_datagen.flow_from_directory(r"/content/drive/MyDrive/dataset/Testing",ta
rget_size=(64,64),batch_size=32,class_mode="categorical")
```

OUTPUT

```
Found 326 images belonging to 4 classes.
```

```
x_train.class_indices
```

OUTPUT

```
{'bears': 0, 'crows': 1, 'elephants': 2, 'rats': 3}
```

Question-3:

3.CREATE MODEL

Solution:

```
#initialize the model
model=Sequential()
```

Question-4:

4.ADD LAYERS(Convolution,MxPooling,Flatten,Dense-(Hidden Layers),Output)

Solution:

```
#add convolution layer
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation='relu'))

#add max pooling layer
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

#hidden layers
model.add(Dense(units=300,kernel_initializer="random_uniform",activation="rel
u"))
model.add(Dense(units=200,kernel_initializer="random_uniform",activation="rel
u"))
#output layer
model.add(Dense(units=4,kernel_initializer="random_uniform",activation="softm
ax"))
```

Question-5:

5.COMPILE THE MODEL

Solution:

```
#compile the model
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=['accu
racy'])
```

Question-6:

6.FIT THE MODEL

Solution:

```
model.fit_generator(x_train,steps_per_epoch=39,epochs=25,validation_data=x_te
st,validation_steps=10)
```

OUTPUT

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.

Epoch 1/25
39/39 [==============================] - 213s 5s/step - loss: 1.3571 -
accuracy: 0.3086 - val_loss: 1.2797 - val_accuracy: 0.3844
Epoch 2/25
39/39 [==============================] - 31s 796ms/step - loss: 1.2132 -
accuracy: 0.4338 - val_loss: 0.9831 - val_accuracy: 0.5469
Epoch 3/25
39/39 [==============================] - 31s 794ms/step - loss: 0.9853 -
accuracy: 0.5792 - val_loss: 0.8243 - val_accuracy: 0.6500
Epoch 4/25
39/39 [==============================] - 31s 790ms/step - loss: 0.8966 -
accuracy: 0.6284 - val_loss: 0.7700 - val_accuracy: 0.6781
Epoch 5/25
39/39 [==============================] - 31s 793ms/step - loss: 0.8226 -
accuracy: 0.6656 - val_loss: 0.6223 - val_accuracy: 0.7656
Epoch 6/25
39/39 [==============================] - 31s 800ms/step - loss: 0.7507 -
accuracy: 0.6922 - val_loss: 0.5325 - val_accuracy: 0.8344
Epoch 7/25
39/39 [==============================] - 31s 796ms/step - loss: 0.7334 -
accuracy: 0.6931 - val_loss: 0.6391 - val_accuracy: 0.7563
Epoch 8/25
39/39 [==============================] - 31s 800ms/step - loss: 0.6739 -
accuracy: 0.7246 - val_loss: 0.4539 - val_accuracy: 0.8188
Epoch 9/25
39/39 [==============================] - 31s 795ms/step - loss: 0.6430 -
accuracy: 0.7528 - val_loss: 0.5661 - val_accuracy: 0.7250
Epoch 10/25
39/39 [==============================] - 31s 793ms/step - loss: 0.5744 -
accuracy: 0.7617 - val_loss: 0.3414 - val_accuracy: 0.8875
Epoch 11/25
39/39 [==============================] - 31s 792ms/step - loss: 0.5035 -
accuracy: 0.8013 - val_loss: 0.5984 - val_accuracy: 0.7781
Epoch 12/25
39/39 [==============================] - 31s 790ms/step - loss: 0.4987 -
```

```
accuracy: 0.8053 - val_loss: 0.3194 - val_accuracy: 0.8781
Epoch 13/25
39/39 [==============================] - 31s 794ms/step - loss: 0.4479 -
accuracy: 0.8183 - val_loss: 0.2687 - val_accuracy: 0.8906
Epoch 14/25
39/39 [==============================] - 31s 793ms/step - loss: 0.3554 -
accuracy: 0.8740 - val_loss: 0.2047 - val_accuracy: 0.9312
Epoch 15/25
39/39 [==============================] - 31s 796ms/step - loss: 0.3572 -
accuracy: 0.8667 - val_loss: 0.3596 - val_accuracy: 0.8313
Epoch 16/25
39/39 [==============================] - 31s 791ms/step - loss: 0.3545 -
accuracy: 0.8708 - val_loss: 0.1499 - val_accuracy: 0.9625
Epoch 17/25
39/39 [==============================] - 31s 794ms/step - loss: 0.3031 -
accuracy: 0.8885 - val_loss: 0.1655 - val_accuracy: 0.9406
Epoch 18/25
39/39 [==============================] - 31s 794ms/step - loss: 0.3006 -
accuracy: 0.8990 - val_loss: 0.1121 - val_accuracy: 0.9656
Epoch 19/25
39/39 [==============================] - 31s 796ms/step - loss: 0.2436 -
accuracy: 0.9063 - val_loss: 0.0975 - val_accuracy: 0.9563
Epoch 20/25
39/39 [==============================] - 31s 793ms/step - loss: 0.2332 -
accuracy: 0.9233 - val_loss: 0.0822 - val_accuracy: 0.9844
Epoch 21/25
39/39 [==============================] - 31s 788ms/step - loss: 0.1828 -
accuracy: 0.9346 - val_loss: 0.0978 - val_accuracy: 0.9625
Epoch 22/25
39/39 [==============================] - 31s 791ms/step - loss: 0.2079 -
accuracy: 0.9330 - val_loss: 0.2019 - val_accuracy: 0.9312
Epoch 23/25
39/39 [==============================] - 31s 796ms/step - loss: 0.1691 -
accuracy: 0.9410 - val_loss: 0.0647 - val_accuracy: 0.9781
Epoch 24/25
39/39 [==============================] - 31s 798ms/step - loss: 0.1361 -
accuracy: 0.9491 - val_loss: 0.0550 - val_accuracy: 0.9750
Epoch 25/25
39/39 [==============================] - 31s 795ms/step - loss: 0.1839 -
accuracy: 0.9346 - val_loss: 0.1726 - val_accuracy: 0.9312

<keras.callbacks.History at 0x7f42189f8dd0>
```

Question-7:

7.SAVE THE MODEL

Solution:

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

Question-8

8.TEST THE MODEL

Solution:

```
#CNN prediction
from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

import numpy as np

model = load_model('animal.h5')
img = image.load_img('/content/drive/MyDrive/dataset/Testing/crows/Z1
(28).jpg',target_size=(64,64))

img
```

OUTPUT

```
type(img)
```

OUTPUT

```
PIL.Image.Image

x=image.img_to_array(img)

x
```

OUTPUT

```
array([[[230., 238., 240.],
        [235., 239., 242.],
        [235., 239., 242.],
        ...,
        [241., 242., 244.],
        [242., 241., 246.],
        [242., 241., 246.]],

       [[234., 238., 241.],
        [235., 239., 242.],
        [235., 239., 242.],
        ...,
        [240., 241., 243.],
        [241., 240., 245.],
```

```
        [242., 241., 246.]],

       [[234., 238., 241.],
        [234., 238., 241.],
        [234., 238., 241.],
        ...,
        [242., 241., 246.],
        [242., 242., 244.],
        [242., 242., 244.]],

       ...,

       [[136.,  97.,  30.],
        [147., 112.,  56.],
        [168., 128.,  59.],
        ...,
        [161., 122.,  53.],
        [159., 124.,  58.],
        [171., 132.,  63.]],

       [[136.,  99.,  29.],
        [147., 112.,  44.],
        [176., 132.,  71.],
        ...,
        [166., 128.,  65.],
        [164., 126.,  53.],
        [176., 131.,  64.]],

       [[148., 109.,  50.],
        [151., 115.,  55.],
        [191., 143.,  79.],
        ...,
        [168., 130.,  67.],
        [156., 122.,  48.],
        [160., 121.,  46.]]], dtype=float32)
```

```
x.shape
```

OUTPUT

```
(64, 64, 3)
```

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

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

```
pred_prob
```

OUTPUT

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

```
class_name=['Bear','Crow','Elephant','Rat']
```

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

```
pred_id
```

OUTPUT

```
1
```

```
print('Predicted animal is',str(class_name[pred_id]))
```

OUTPUT

```
Predicted animal is Crow
```