1.DOWNLOAD THE DATASET

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

```
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
```

2.IMAGE AUGUMENTATION

```
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_f
```

```
test_datagen=ImageDataGenerator(rescale=1./255)
```

```
x_train = train_datagen.flow_from_directory(r"/content/drive/MyDrive/dataset/Training",target
```

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

```
#load your images data
```

```
x_test = test_datagen.flow_from_directory(r"/content/drive/MyDrive/dataset/Testing",target_si
```

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

```
x_train.class_indices
```

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

3.CREATE MODEL

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

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

```
#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="relu"))
model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))
#output layer
model.add(Dense(units=4,kernel_initializer="random_uniform",activation="softmax"))
```

5.COMPILE THE MODEL

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

6.FIT THE MODEL

```
model.fit_generator(x_train,steps_per_epoch=39,epochs=25,validation_data=x_test,validation_st
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_
  """Entry point for launching an IPython kernel.
Epoch 1/25
39/39 [==============================] - 213s 5s/step - loss: 1.3571 - accuracy: 0.3086
Epoch 2/25
39/39 [==============================] - 31s 796ms/step - loss: 1.2132 - accuracy: 0.433
Epoch 3/25
39/39 [==============================] - 31s 794ms/step - loss: 0.9853 - accuracy: 0.579
Epoch 4/25
39/39 [==============================] - 31s 790ms/step - loss: 0.8966 - accuracy: 0.628
Epoch 5/25
39/39 [==============================] - 31s 793ms/step - loss: 0.8226 - accuracy: 0.665
Epoch 6/25
39/39 [==============================] - 31s 800ms/step - loss: 0.7507 - accuracy: 0.692
Epoch 7/25
39/39 [==============================] - 31s 796ms/step - loss: 0.7334 - accuracy: 0.693
Epoch 8/25
39/39 [==============================] - 31s 800ms/step - loss: 0.6739 - accuracy: 0.724
Epoch 9/25
39/39 [==============================] - 31s 795ms/step - loss: 0.6430 - accuracy: 0.752
Epoch 10/25
39/39 [==============================] - 31s 793ms/step - loss: 0.5744 - accuracy: 0.761
Epoch 11/25
39/39 [==============================] - 31s 792ms/step - loss: 0.5035 - accuracy: 0.801
Epoch 12/25
39/39 [==============================] - 31s 790ms/step - loss: 0.4987 - accuracy: 0.805
Epoch 13/25
```

```
39/39 [==============================] - 31s 794ms/step - loss: 0.4479 - accuracy: 0.818
Epoch 14/25
39/39 [==============================] - 31s 793ms/step - loss: 0.3554 - accuracy: 0.874
Epoch 15/25
39/39 [==============================] - 31s 796ms/step - loss: 0.3572 - accuracy: 0.866
Epoch 16/25
39/39 [==============================] - 31s 791ms/step - loss: 0.3545 - accuracy: 0.876
Epoch 17/25
39/39 [==============================] - 31s 794ms/step - loss: 0.3031 - accuracy: 0.888
Epoch 18/25
39/39 [==============================] - 31s 794ms/step - loss: 0.3006 - accuracy: 0.899
Epoch 19/25
39/39 [==============================] - 31s 796ms/step - loss: 0.2436 - accuracy: 0.906
Epoch 20/25
39/39 [==============================] - 31s 793ms/step - loss: 0.2332 - accuracy: 0.923
Epoch 21/25
39/39 [==============================] - 31s 788ms/step - loss: 0.1828 - accuracy: 0.934
Epoch 22/25
39/39 [==============================] - 31s 791ms/step - loss: 0.2079 - accuracy: 0.933
Epoch 23/25
39/39 [==============================] - 31s 796ms/step - loss: 0.1691 - accuracy: 0.941
Epoch 24/25
39/39 [==============================] - 31s 798ms/step - loss: 0.1361 - accuracy: 0.949
Epoch 25/25
39/39 [==============================] - 31s 795ms/step - loss: 0.1839 - accuracy: 0.934
<keras.callbacks.History at 0x7f42189f8dd0>
```

7.SAVE THE MODEL

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

8.TEST THE MODEL

```
#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=


img
```

```
type(img)
```

```
PIL.Image.Image
```

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

```
x
```

```
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
```

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

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

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

```
pred_prob
```

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

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

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

```
pred_id
```

```
     1
```

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

```
     Predicted animal is Crow
```

Colab paid products - Cancel contracts here