```
from zipfile import ZipFile


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

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

```
!unzip '/content/drive/MyDrive/Animal_Dataset.zip'
```

```
       inflating: dataset/Training/rats/images (51).jpeg
       inflating: dataset/Training/rats/images (52).jpeg
       inflating: dataset/Training/rats/images (53).jpeg
       inflating: dataset/Training/rats/images (54).jpeg
       inflating: dataset/Training/rats/images (55).jpeg
       inflating: dataset/Training/rats/images (56).jpeg
       inflating: dataset/Training/rats/images (57).jpeg
       inflating: dataset/Training/rats/images (58).jpeg
       inflating: dataset/Training/rats/images (59).jpeg
       inflating: dataset/Training/rats/images (6).jpeg
       inflating: dataset/Training/rats/images (60).jpeg
       inflating: dataset/Training/rats/images (61).jpeg
       inflating: dataset/Training/rats/images (62).jpeg
       inflating: dataset/Training/rats/images (63).jpeg
       inflating: dataset/Training/rats/images (64).jpeg
       inflating: dataset/Training/rats/images (65).jpeg
       inflating: dataset/Training/rats/images (66).jpeg
       inflating: dataset/Training/rats/images (67).jpeg
       inflating: dataset/Training/rats/images (68).jpeg
       inflating: dataset/Training/rats/images (69).jpeg
       inflating: dataset/Training/rats/images (7).jpeg
       inflating: dataset/Training/rats/images (70).jpeg
       inflating: dataset/Training/rats/images (71).jpeg
       inflating: dataset/Training/rats/images (72).jpeg
       inflating: dataset/Training/rats/images (73).jpeg
       inflating: dataset/Training/rats/images (74).jpeg
       inflating: dataset/Training/rats/images (75).jpeg
       inflating: dataset/Training/rats/images (76).jpeg
       inflating: dataset/Training/rats/images (77).jpeg
       inflating: dataset/Training/rats/images (78).jpeg
       inflating: dataset/Training/rats/images (79).jpeg
       inflating: dataset/Training/rats/images (8).jpeg
       inflating: dataset/Training/rats/images (80).jpeg
       inflating: dataset/Training/rats/images (81).jpeg
       inflating: dataset/Training/rats/images (82).jpeg
       inflating: dataset/Training/rats/images (83).jpeg
       inflating: dataset/Training/rats/images (84).jpeg
       inflating: dataset/Training/rats/images (85).jpeg
       inflating: dataset/Training/rats/images (86).jpeg
       inflating: dataset/Training/rats/images (87).jpeg
       inflating: dataset/Training/rats/images (88).jpeg
       inflating: dataset/Training/rats/images (89).jpeg
       inflating: dataset/Training/rats/images (9).jpeg
       inflating: dataset/Training/rats/images (90).jpeg
       inflating: dataset/Training/rats/images (91).jpeg
       inflating: dataset/Training/rats/images (92).jpeg
       inflating: dataset/Training/rats/images (93).jpeg
       inflating: dataset/Training/rats/images (94).jpeg
       inflating: dataset/Training/rats/images (95).jpeg
       inflating: dataset/Training/rats/images (96).jpeg
       inflating: dataset/Training/rats/images (97).jpeg
       inflating: dataset/Training/rats/images (98).jpeg
       inflating: dataset/Training/rats/images (99).jpeg
       inflating: dataset/Training/rats/images - 2019-05-23T115933.485.jpeg
       inflating: dataset/Training/rats/images - 2019-05-23T115940.497.jpeg
       inflating: dataset/Training/rats/images - 2019-05-23T120017.550.jpeg
       inflating: dataset/Training/rats/images.jpeg
       inflating: dataset/Training/rats/images.png
```

watch the first video "for unizip!" path bold text

```
# Data Augmentation

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

```
train_gen = ImageDataGenerator(rescale=(1./255),horizontal_flip=True,shear_range=0.2)
test_gen = ImageDataGenerator(rescale=(1./255))  #--> (0 to 255) convert to (0 to 1)



train = train_gen.flow_from_directory('/content/dataset/Training',
                                      target_size=(120, 120),
                                      class_mode='categorical',
                                      batch_size=8)
test = test_gen.flow_from_directory('/content/dataset/Testing',
                                    target_size=(120, 120),
                                      class_mode='categorical',
                                      batch_size=8)
```

```
Found 1238 images belonging to 4 classes.
Found 326 images belonging to 4 classes.
```

```
train.class_indices
```

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

```
# CNN

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


model = Sequential()
model.add(Convolution2D(20,(3,3),activation='relu',input_shape=(120, 120, 3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(45,activation='relu'))
model.add(Dense(4,activation='softmax'))


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


model.fit(train,batch_size=8,validation_data=test,epochs=10)
```

```
Epoch 1/10
155/155 [==============================] - 26s 162ms/step - loss: 1.4863 - accuracy: 0.2981 - val_loss: 1.3830 - val_accuracy: 0.2607
Epoch 2/10
155/155 [==============================] - 22s 145ms/step - loss: 1.3728 - accuracy: 0.2948 - val_loss: 1.3774 - val_accuracy: 0.2945
Epoch 3/10
155/155 [==============================] - 23s 146ms/step - loss: 1.3657 - accuracy: 0.3231 - val_loss: 1.4030 - val_accuracy: 0.2822
Epoch 4/10
155/155 [==============================] - 24s 152ms/step - loss: 1.3587 - accuracy: 0.3255 - val_loss: 1.3756 - val_accuracy: 0.2945
Epoch 5/10
155/155 [==============================] - 21s 138ms/step - loss: 1.3572 - accuracy: 0.3288 - val_loss: 1.3815 - val_accuracy: 0.2914
Epoch 6/10
155/155 [==============================] - 23s 148ms/step - loss: 1.3580 - accuracy: 0.3231 - val_loss: 1.4078 - val_accuracy: 0.2577
Epoch 7/10
155/155 [==============================] - 23s 149ms/step - loss: 1.3594 - accuracy: 0.3247 - val_loss: 1.3832 - val_accuracy: 0.2914
Epoch 8/10
155/155 [==============================] - 23s 150ms/step - loss: 1.3555 - accuracy: 0.3223 - val_loss: 1.3840 - val_accuracy: 0.2914
Epoch 9/10
155/155 [==============================] - 23s 148ms/step - loss: 1.3543 - accuracy: 0.3231 - val_loss: 1.3847 - val_accuracy: 0.2914
Epoch 10/10
155/155 [==============================] - 23s 147ms/step - loss: 1.3536 - accuracy: 0.3231 - val_loss: 1.3854 - val_accuracy: 0.2914
<keras.src.callbacks.History at 0x7ddfe675f130>
```

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

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `m
  saving_api.save_model(
```

```
# Testing

import numpy as np
from tensorflow.keras.preprocessing import image



img = image.load_img('/content/drive/MyDrive/crow.jpeg',target_size=(120,120))


img
```

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

```
array([[[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [252., 247., 244.],
        [251., 246., 243.],
        [251., 246., 243.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [252., 247., 244.],
        [251., 246., 243.],
        [251., 246., 243.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [252., 247., 244.],
        [251., 246., 243.],
        [251., 246., 243.]],

       ...,

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [254., 253., 251.],
        [254., 253., 251.],
        [254., 253., 251.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 254., 252.],
        [254., 253., 251.],
        [254., 253., 251.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 254., 252.],
        [254., 253., 251.],
        [254., 253., 251.]]], dtype=float32)
```

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

```
array([[[[255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.],
         ...,
         [252., 247., 244.],
```

```
        [251., 246., 243.],
        [251., 246., 243.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [252., 247., 244.],
        [251., 246., 243.],
        [251., 246., 243.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [252., 247., 244.],
        [251., 246., 243.],
        [251., 246., 243.]],

       ...,

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [254., 253., 251.],
        [254., 253., 251.],
        [254., 253., 251.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 254., 252.],
        [254., 253., 251.],
        [254., 253., 251.]],

       [[255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.],
        ...,
        [255., 254., 252.],
        [254., 253., 251.],
        [254., 253., 251.]]], dtype=float32)
```

```
np.argmax(model.predict(img))
```

```
1/1 [==============================] - 0s 104ms/step
0
```