# **Shyama Harihar CB.EN.U4CSE19147 GROUP 11** CBIR - CNN Image Classification

## Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
from keras.layers import Dense,Flatten
from keras.models import Model
from keras.applications.inception_v3 import InceptionV3,preprocess_input
from keras.preprocessing.image import ImageDataGenerator
import keras
```

```
input_dir='/content/drive/MyDrive/Data/Classes'
```

## Build the base model - Inception v3

```
base_model=InceptionV3(input_shape=(512,512,3),include_top=False)
```

```
for layer in base_model.layers:
  layer.trainable=False
```

```
X=Flatten()(base_model.output)
X=Dense(units=6,activation='sigmoid')(X)

#The Final model after Attaching the Dense Layer

model=Model(base_model.input,X)

#Compile the Model
model.compile(optimizer='adam',loss=keras.losses.binary_crossentropy,metrics=['accuracy']

#Summary
model.summary()
```

```
      conv2d_277 (Conv2D)            (None, 14, 14, 448)  917504      ['mixed9[0][0]']

      batch_normalization_277 (Batch  (None, 14, 14, 448)  1344       ['conv2d_277[0][
      Normalization)

      activation_277 (Activation)    (None, 14, 14, 448)  0           ['batch_normaliz

      conv2d_274 (Conv2D)            (None, 14, 14, 384)  786432      ['mixed9[0][0]']

      conv2d_278 (Conv2D)            (None, 14, 14, 384)  1548288     ['activation_277

      batch_normalization_274 (Batch  (None, 14, 14, 384)  1152       ['conv2d_274[0][
      Normalization)
```

```
batch_normalization_278 (Batch  (None, 14, 14, 384)  1152       ['conv2d_278[0][
Normalization)

activation_274 (Activation)    (None, 14, 14, 384)  0          ['batch_normaliz

activation_278 (Activation)    (None, 14, 14, 384)  0          ['batch_normaliz

conv2d_275 (Conv2D)            (None, 14, 14, 384)  442368     ['activation_274

conv2d_276 (Conv2D)            (None, 14, 14, 384)  442368     ['activation_274

conv2d_279 (Conv2D)            (None, 14, 14, 384)  442368     ['activation_278

conv2d_280 (Conv2D)            (None, 14, 14, 384)  442368     ['activation_278

average_pooling2d_26 (AverageP  (None, 14, 14, 2048  0          ['mixed9[0][0]']
ooling2D)                      )

conv2d_273 (Conv2D)            (None, 14, 14, 320)  655360     ['mixed9[0][0]']

batch_normalization_275 (Batch  (None, 14, 14, 384)  1152       ['conv2d_275[0][
Normalization)

batch_normalization_276 (Batch  (None, 14, 14, 384)  1152       ['conv2d_276[0][
Normalization)

batch_normalization_279 (Batch  (None, 14, 14, 384)  1152       ['conv2d_279[0][
Normalization)


batch_normalization_280 (Batch  (None, 14, 14, 384)  1152       ['conv2d_280[0][
Normalization)

conv2d_281 (Conv2D)            (None, 14, 14, 192)  393216     ['average_poolin

batch_normalization_273 (Batch  (None, 14, 14, 320)  960        ['conv2d_273[0][
Normalization)

activation_275 (Activation)    (None, 14, 14, 384)  0          ['batch_normaliz

activation_276 (Activation)    (None, 14, 14, 384)  0          ['batch_normaliz

activation_279 (Activation)    (None, 14, 14, 384)  0          ['batch_normaliz

activation_280 (Activation)    (None, 14, 14, 384)  0          ['batch_normaliz
```

```
train_datagen=ImageDataGenerator(featurewise_center=True,
                                 rotation_range=0.4,
                                 width_shift_range=0.3,
                                 horizontal_flip=True,
                                 preprocessing_function=preprocess_input,
                                 zoom_range=0.4,
                                 shear_range=0.4)
train_data=train_datagen.flow_from_directory(directory=input_dir,
                                             target_size=(512,512),
                                             batch_size=36)
```

```
Found 420 images belonging to 6 classes.
```

```
train_data.class_indices
```

```
{'Cheetah': 0, 'Fox': 1, 'Lion': 2, 'Lioness': 3, 'Tiger': 4, 'WhiteTiger': 5}
```

## See the data preprocessed

```
t_img,label=train_data.next()
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data_generato
  warnings.warn('This ImageDataGenerator specifies '
```

```
t_img
```

```
         [ 0.32459605,  0.37165487,  0.37165487],
         ...,
         [ 0.23921573,  0.30980396,  0.28627455],
         [ 0.23874962,  0.31026995,  0.28627455],
         [ 0.2329917 ,  0.316028  ,  0.28627455]],

        [[ 0.32822073,  0.37527955,  0.37527955],
         [ 0.32549024,  0.37254906,  0.37254906],
         [ 0.3245479 ,  0.3716067 ,  0.3716067 ],
         ...,
         [ 0.23921573,  0.30980396,  0.28627455],
         [ 0.23870146,  0.31031823,  0.28627455],
         [ 0.23294342,  0.31607616,  0.28627455]],

        [[ 0.32817268,  0.3752315 ,  0.3752315 ],
         [ 0.32549024,  0.37254906,  0.37254906],
         [ 0.32449973,  0.37155855,  0.37155855],
         ...,
         [ 0.23921573,  0.30980396,  0.28627455],
         [ 0.23865318,  0.3103664 ,  0.28627455],
         [ 0.23289537,  0.31612432,  0.28627455]],

        ...,

        [[-0.16434938, -0.3037433 , -0.62281644],
         [-0.25222003, -0.39339656, -0.70712197],
         [-0.22208595, -0.36326241, -0.6769879 ],
         ...,
         [ 0.4933009 ,  0.54035985,  0.5089872 ],
         [ 0.9529412 ,  1.        ,  0.96862745],
         [ 0.9529412 ,  1.        ,  0.96862745]],

        [[-0.164253  , -0.30369514, -0.6226237 ],
         [-0.25332826, -0.39450473, -0.70823026],
         [-0.21905035, -0.3602268 , -0.67395234],
         ...,
         [ 0.4984566 ,  0.5455154 ,  0.5141429 ],
         [ 0.9529412 ,  1.        ,  0.96862745],
         [ 0.9529412 ,  1.        ,  0.96862745]],

        [[-0.16415662, -0.30364698, -0.6224309 ],
         [-0.25443655, -0.395613  , -0.7093384 ],
         [-0.2160148 , -0.35719126, -0.6709167 ],
         ...,
         [ 0.5036123 ,  0.5506711 ,  0.51929855],
         [ 0.9529412 ,  1.        ,  0.96862745],
         [ 0.9529412 ,  1.        ,  0.96862745]]],


       [[[-0.50561875, -0.53699136, -0.56052077],
         [-0.49028122, -0.5216538 , -0.5451832 ],
         [-0.47981995, -0.51119256, -0.534722  ],
         ...,
         [ 0.4401058 ,  0.37614298,  0.39023983],
         [ 0.53728914,  0.4784789 ,  0.5073745 ],
         [ 0.6262486 ,  0.58089006,  0.6171627 ]],

        [[-0.5079623 , -0.53933483, -0.56286424],
         [-0.4969496 , -0.5283221 , -0.5518515 ],
```

```
t_img.shape
```

```
(36, 512, 512, 3)
```

## Model CheckPoint

```
from keras.callbacks import ModelCheckpoint,EarlyStopping
mc=ModelCheckpoint(filepath="./best_model.h5",
                   monitor="accuracy",
                   verbose=1,
                   save_best_only=True
                   )
es=EarlyStopping(monitor="accuracy",
                 min_delta=0.01,
                 patience=5,
                 verbose=1)
cb=[mc,es]
```

```
his=model.fit_generator(train_data,
steps_per_epoch=10,
epochs=10,
validation_freq=1,
callbacks=cb)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: UserWarning: `Model.
  """
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data_generato
  warnings.warn('This ImageDataGenerator specifies '
Epoch 1/10
10/10 [==============================] - ETA: 0s - loss: 4.6167 - accuracy: 0.6111
Epoch 00001: accuracy improved from -inf to 0.61111, saving model to ./best_model.h5
10/10 [==============================] - 53s 5s/step - loss: 4.6167 - accuracy: 0.61
Epoch 2/10
10/10 [==============================] - ETA: 0s - loss: 0.9902 - accuracy: 0.9195
Epoch 00002: accuracy improved from 0.61111 to 0.91954, saving model to ./best_model
10/10 [==============================] - 35s 4s/step - loss: 0.9902 - accuracy: 0.91
Epoch 3/10
```

```
10/10 [==============================] - ETA: 0s - loss: 0.3927 - accuracy: 0.9425
Epoch 00003: accuracy improved from 0.91954 to 0.94253, saving model to ./best_model
10/10 [==============================] - 27s 3s/step - loss: 0.3927 - accuracy: 0.94
Epoch 4/10
10/10 [==============================] - ETA: 0s - loss: 0.1877 - accuracy: 0.9598
Epoch 00004: accuracy improved from 0.94253 to 0.95977, saving model to ./best_model
10/10 [==============================] - 28s 3s/step - loss: 0.1877 - accuracy: 0.95
Epoch 5/10
10/10 [==============================] - ETA: 0s - loss: 0.1486 - accuracy: 0.9667
Epoch 00005: accuracy improved from 0.95977 to 0.96667, saving model to ./best_model
10/10 [==============================] - 29s 3s/step - loss: 0.1486 - accuracy: 0.96
Epoch 6/10
10/10 [==============================] - ETA: 0s - loss: 0.1332 - accuracy: 0.9741
Epoch 00006: accuracy improved from 0.96667 to 0.97414, saving model to ./best_model
10/10 [==============================] - 28s 3s/step - loss: 0.1332 - accuracy: 0.97
Epoch 7/10
10/10 [==============================] - ETA: 0s - loss: 0.0922 - accuracy: 0.9770
Epoch 00007: accuracy improved from 0.97414 to 0.97701, saving model to ./best_model
10/10 [==============================] - 27s 3s/step - loss: 0.0922 - accuracy: 0.97
Epoch 8/10
10/10 [==============================] - ETA: 0s - loss: 0.0730 - accuracy: 0.9828
Epoch 00008: accuracy improved from 0.97701 to 0.98276, saving model to ./best_model
10/10 [==============================] - 28s 3s/step - loss: 0.0730 - accuracy: 0.98
Epoch 9/10
10/10 [==============================] - ETA: 0s - loss: 0.0450 - accuracy: 0.9943
Epoch 00009: accuracy improved from 0.98276 to 0.99425, saving model to ./best_model
10/10 [==============================] - 28s 3s/step - loss: 0.0450 - accuracy: 0.99
Epoch 10/10
10/10 [==============================] - ETA: 0s - loss: 0.0495 - accuracy: 0.9885
Epoch 00010: accuracy did not improve from 0.99425
10/10 [==============================] - 26s 3s/step - loss: 0.0495 - accuracy: 0.98
```

```
from keras.models import load_model
model=load_model("/content/best_model.h5")
```

```
h=his.history
h.keys
```

```
<function dict.keys>
```

```
plt.plot(h['loss'])
plt.plot(h['accuracy'],c='red')
```

```
[<matplotlib.lines.Line2D at 0x7f113f703910>]
```

```
import tensorflow as tf
```

```
import os
import cv2
```

Custom Functions for displaying images

```
import cv2
from matplotlib import pyplot as plt
def showtiger():
  fig = plt.figure(figsize=(10, 7))

# setting values to rows and column variables
  rows = 2
  columns = 2

# reading images
  Image1 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image2 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image3 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image4 =tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Data

# Adds a subplot at the 1st position
  fig.add_subplot(rows, columns, 1)

# showing image
  plt.imshow(Image1/255.)
  plt.axis('off')
  plt.title("First")

# Adds a subplot at the 2nd position
  fig.add_subplot(rows, columns, 2)

# showing image
  plt.imshow(Image2/255.)
  plt.axis('off')
  plt.title("Second")

# Adds a subplot at the 3rd position
  fig.add_subplot(rows, columns, 3)

# showing image
  plt.imshow(Image3/255.)
  plt.axis('off')
  plt.title("Third")

# Adds a subplot at the 4th position
  fig.add_subplot(rows, columns, 4)
```

```
# showing image
  plt.imshow(Image4/255.)
  plt.axis('off')
  plt.title("Fourth")

def showwhitetiger():
  fig = plt.figure(figsize=(10, 7))

# setting values to rows and column variables
  rows = 2
  columns = 2

# reading images
  Image1 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image2 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image3 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image4 =tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Data

# Adds a subplot at the 1st position
  fig.add_subplot(rows, columns, 1)

# showing image
  plt.imshow(Image1/255.)
  plt.axis('off')
  plt.title("First")

# Adds a subplot at the 2nd position
  fig.add_subplot(rows, columns, 2)

# showing image
  plt.imshow(Image2/255.)
  plt.axis('off')
  plt.title("Second")

# Adds a subplot at the 3rd position
  fig.add_subplot(rows, columns, 3)

# showing image
  plt.imshow(Image3/255.)
  plt.axis('off')
  plt.title("Third")

# Adds a subplot at the 4th position
  fig.add_subplot(rows, columns, 4)

# showing image
  plt.imshow(Image4/255.)
  plt.axis('off')
  plt.title("Fourth")


def showfox():
  fig = plt.figure(figsize=(10, 7))

# setting values to rows and column variables
```

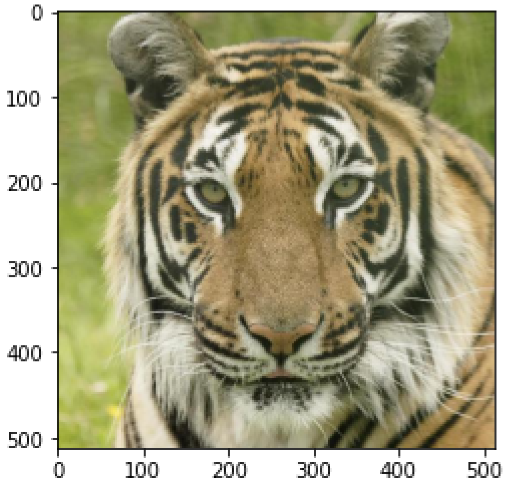
```
  rows = 2
  columns = 2

# reading images
  Image1 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image2 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image3 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image4 =tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Data

# Adds a subplot at the 1st position
  fig.add_subplot(rows, columns, 1)

# showing image
  plt.imshow(Image1/255.)
  plt.axis('off')
  plt.title("First")

# Adds a subplot at the 2nd position
  fig.add_subplot(rows, columns, 2)

# showing image
  plt.imshow(Image2/255.)
  plt.axis('off')
  plt.title("Second")

# Adds a subplot at the 3rd position
  fig.add_subplot(rows, columns, 3)

# showing image
  plt.imshow(Image3/255.)
  plt.axis('off')
  plt.title("Third")

# Adds a subplot at the 4th position
  fig.add_subplot(rows, columns, 4)

# showing image
  plt.imshow(Image4/255.)
  plt.axis('off')
  plt.title("Fourth")



def showLioness():
  fig = plt.figure(figsize=(10, 7))

# setting values to rows and column variables
  rows = 2
  columns = 2

# reading images
  Image1 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image2 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image3 = tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Dat
  Image4 =tf.keras.utils.img_to_array(tf.keras.utils.load_img('/content/drive/MyDrive/Data
# Adds a subplot at the 1st position
  fig.add_subplot(rows, columns, 1)

# showing image
  plt.imshow(Image1/255.)
  plt.axis('off')
  plt.title("First")

# Adds a subplot at the 2nd position
  fig.add_subplot(rows, columns, 2)

# showing image
  plt.imshow(Image2/255.)
  plt.axis('off')
  plt.title("Second")

# Adds a subplot at the 3rd position
  fig.add_subplot(rows, columns, 3)

# showing image
  plt.imshow(Image3/255.)
  plt.axis('off')
  plt.title("Third")

# Adds a subplot at the 4th position
  fig.add_subplot(rows, columns, 4)

# showing image
  plt.imshow(Image4/255.)
  plt.axis('off')
  plt.title("Fourth")
```

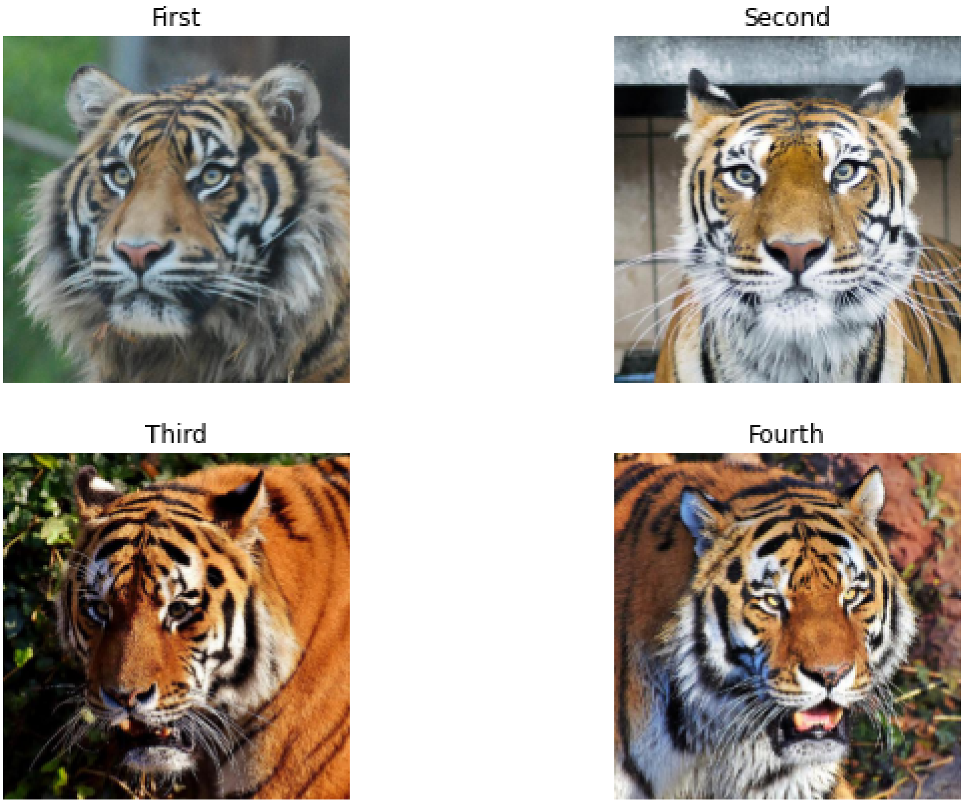
## Testing the model

```
path='/content/drive/MyDrive/Data/test/156.jpg'
img=tf.keras.utils.load_img(path,target_size=(512,512))
i=tf.keras.utils.img_to_array(img)
plt.imshow(i/255.)
i=preprocess_input(i)
#print(i)
input_arr=np.array([i])
pred=np.argmax(model.predict(input_arr))
if pred==0:
  print('Cheetah')
elif pred==1:
  print('Fox')
elif pred==2:
  print('Lion')
elif pred==3:
  print('Lioness')
elif pred==4:
  print('Tiger')
  showtiger()
```

```
elif pred==5:
  print('WhiteTiger')
```

```
Tiger
```




First

Second

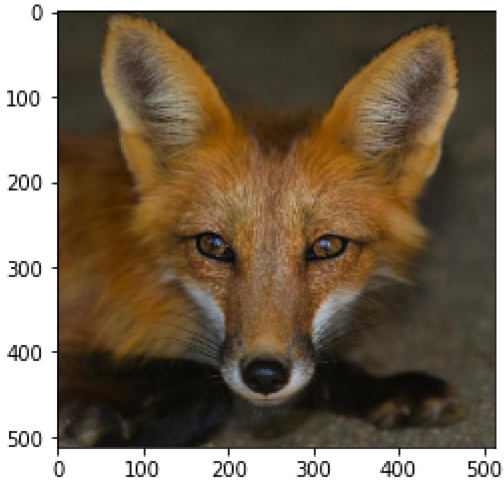
Third

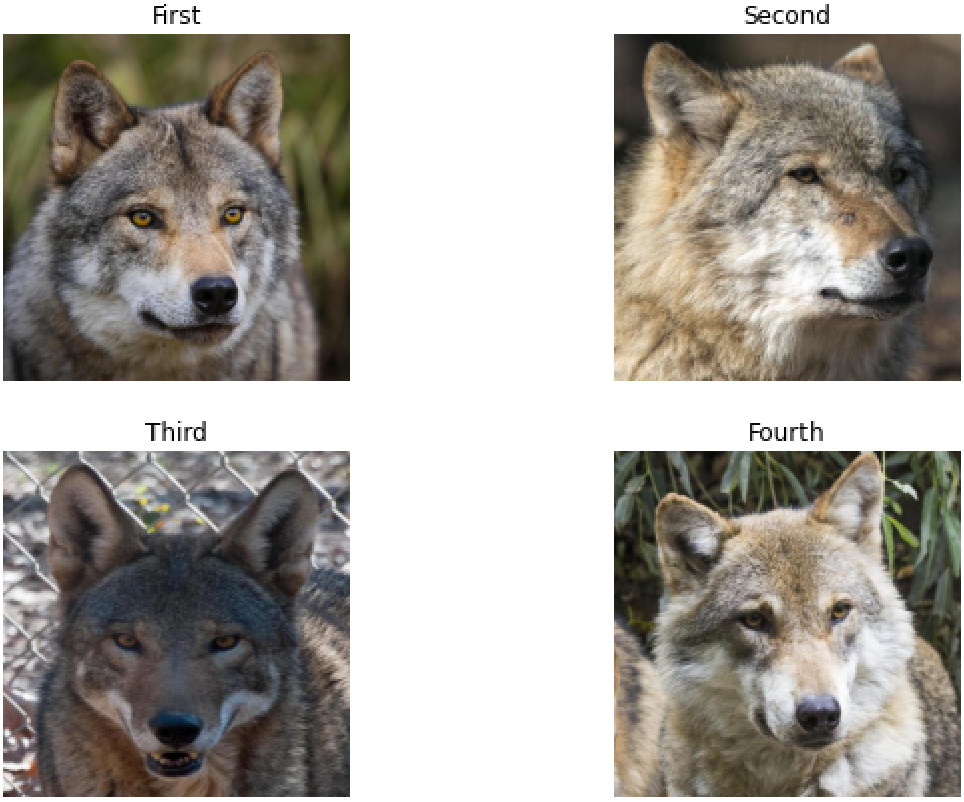
Fourth

```
path='/content/drive/MyDrive/Data/test/626.jpg'
img=tf.keras.utils.load_img(path,target_size=(512,512))
i=tf.keras.utils.img_to_array(img)
plt.imshow(i/255.)
i=preprocess_input(i)
#print(i)
input_arr=np.array([i])
pred=np.argmax(model.predict(input_arr))
if pred==0:
  print('Cheetah')
elif pred==1:
  print('Fox')
  showfox()
```

```
elif pred==2:
  print('Lion')
elif pred==3:
  print('Lioness')
elif pred==4:
  print('Tiger')
elif pred==5:
  print('WhiteTiger')
```

```
Fox
```

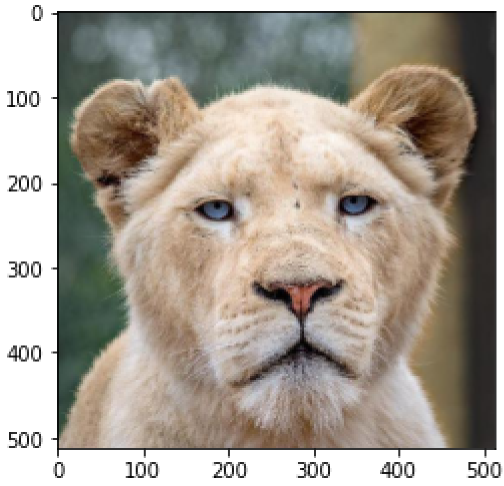


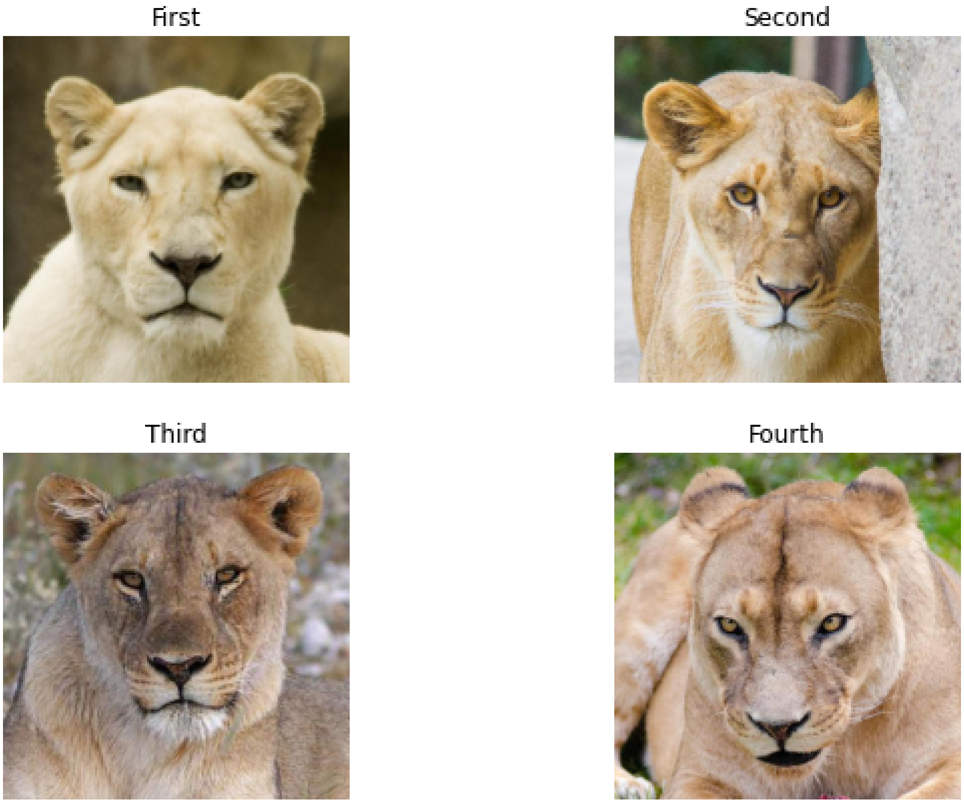


```
path='/content/drive/MyDrive/Data/test/1525.jpg'
img=tf.keras.utils.load_img(path,target_size=(512,512))
i=tf.keras.utils.img_to_array(img)
plt.imshow(i/255.)
i=preprocess_input(i)
#print(i)
input_arr=np.array([i])
pred=np.argmax(model.predict(input_arr))
```

```
if pred==0:
  print('Cheetah')
elif pred==1:
  print('Fox')
elif pred==2:
  print('Lion')
elif pred==3:
  print('Lioness')
  showLioness()
elif pred==4:
  print('Tiger')
elif pred==5:
  print('WhiteTiger')
```

```
Lioness
```

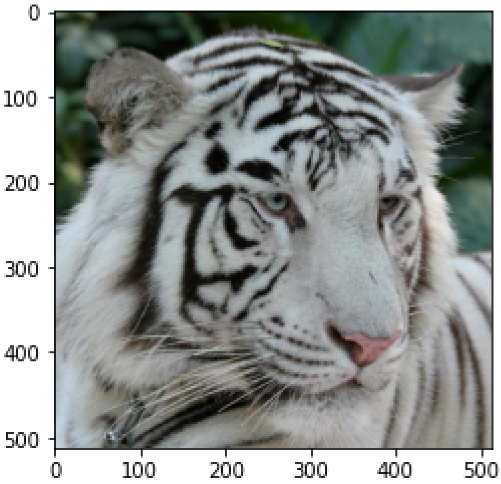


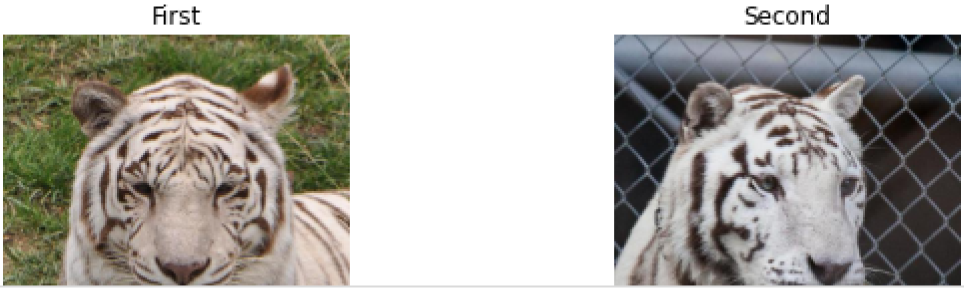


```
path='/content/drive/MyDrive/Data/test/1649.jpg'
img=tf.keras.utils.load_img(path,target_size=(512,512))
i=tf.keras.utils.img_to_array(img)
```

```
    plt.imshow(i/255.)
    i=preprocess_input(i)
    #print(i)
    input_arr=np.array([i])
    pred=np.argmax(model.predict(input_arr))
    if pred==0:
      print('Cheetah')
    elif pred==1:
      print('Fox')
    elif pred==2:
      print('Lion')
    elif pred==3:
      print('Lioness')
    elif pred==4:
      print('Tiger')
    elif pred==5:
      print('WhiteTiger')
      showwhitetiger()
```

```
WhiteTiger
```




First

Second

✓ 1s completed at 10:11 AM