```
!unzip /content/drive/MyDrive/dataset.zip
  inflating: Trainingset/Wild animal/OIP-XWGEslsdy_OCPIME7p_BKQHaH2.jpeg
  inflating: Trainingset/Wild animal/OIP-XwkSC1VAzv9T5D1iXCKu0AHaEo.jpeg
  inflating: Trainingset/Wild animal/OIP-xwWrIzvhJ1vaokUaLyZ50AHaFc.jpeg
  inflating: Trainingset/Wild animal/OIP-xwzQyekc4HrqPcSpS6DlyAHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-xZD2U2o_YLgZ4G2bTDndzAHaE6.jpeg
  inflating: Trainingset/Wild animal/OIP-y_k9wnt_j5pZZZe0yfd6dwHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-y26yPV9EgCbFMmfxfVaiQQHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-Y2pVAQuRkOTpUaVOJzOdPwHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-Y6BQuvYoeS0tmqv4NnBLYgHaHk.jpeg
  inflating: Trainingset/Wild animal/OIP-Y9Lc3aBXRdO_I3lyNwIecQAAAA.jpeg
  inflating: Trainingset/Wild animal/OIP-YA0uxaxtoKflISNLy4bqfQHaE9.jpeg
  inflating: Trainingset/Wild animal/OIP-YApir7af--L_6FQOC9tijQAAAA.jpeg
  inflating: Trainingset/Wild animal/OIP-YCGPKLwC-fbOp2x18Fe80AHaLI.jpeg
  inflating: Trainingset/Wild animal/OIP-yciJzcz9CgBokAvOk-hL7QHaFb.jpeg
  inflating: Trainingset/Wild animal/OIP-yDn1ham7DjZ5oS4zGmiJLgAAAA.jpeg
  inflating: Trainingset/Wild animal/OIP-yeIFN-pyg-Ewti8esESt9AHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-YHhNFMosfdT6P1QMpjXxRAHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-yi3kvE7_uxui2a4F_PYyPQHaEK.jpeg
  inflating: Trainingset/Wild animal/OIP-Yio1IWV3YK1BJOiY2HLEwwHaE1.jpeg
  inflating: Trainingset/Wild animal/OIP-Ykv4UIvJtjedkWi9QPbfGgHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-YLAWtT7J2BluyhLcvxdk5AHaEK.jpeg
  inflating: Trainingset/Wild animal/OIP-yNDf-ruD3aWAA14mJWpB1AHaEK.jpeg
  inflating: Trainingset/Wild animal/OIP-YnKLqp8AjcA2RfMYDNukAQHaE7.jpeg
  inflating: Trainingset/Wild animal/OIP-YoLyyzHee6BHTx01rbqSEQHaFh.jpeg
  inflating: Trainingset/Wild animal/OIP-yoZ1Q3dD3VlNtSnBurtbswHaFR.jpeg
  inflating: Trainingset/Wild animal/OIP-YrkykgfBRezEr041yzYScQHaE-.jpeg
  inflating: Trainingset/Wild animal/OIP-yRWJgpu2dvuRPKY1eDgNfAHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-ysaCOurFtmn4JKvx1sFyhgHaFc.jpeg
  inflating: Trainingset/Wild animal/OIP-YSBJtsDgNOfzJfuk4UHDsAHaEK.jpeg
  inflating: Trainingset/Wild animal/OIP-yU-CsRDvGX7_8UYbtKi-MQHaE1.jpeg
  inflating: Trainingset/Wild animal/OIP-YXYCzbuMYSZXgxcdGwGI0wHaE1.jpeg
  inflating: Trainingset/Wild animal/OIP-z_vKf7Z8Gjs2JjkUTbbnrwHaGl.jpeg
  inflating: Trainingset/Wild animal/OIP-z1WgJuO8vFGlaA1fLtRTogHaE8.jpeg
  inflating: Trainingset/Wild animal/OIP-z4BZGwSwbr5mRB41zPhlrwHaE8.jpeg
  inflating: Trainingset/Wild animal/OIP-z6azUmkPF2nT-MsFP3jp3AHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-Z9wGusnpjaWEUwcU_gcMHgHaEA.jpeg
  inflating: Trainingset/Wild animal/OIP-ZapRpPspWsg92JPyPw9mmgHaE7.jpeg
  inflating: Trainingset/Wild animal/OIP-zaY33YSyzBcvJq_2Fb504QHaE-.jpeg
  inflating: Trainingset/Wild animal/OIP-zbPZNqeqEZ8Nj5QmqSV23gHaE6.jpeg
  inflating: Trainingset/Wild animal/OIP-zcxByXad8au5aZndi6SyqgEsDI.jpeg
  inflating: Trainingset/Wild animal/OIP-ZcyFicUrQxAEJwL7c4LdTwHaEI.jpeg
  inflating: Trainingset/Wild animal/OIP-zdLC6J2HBU8m5uGhQNupOgAAAA.jpeg
  inflating: Trainingset/Wild animal/OIP-ZdUyIxcRqX2nAxxNWFgF2wHaKD.jpeg
  inflating: Trainingset/Wild animal/OIP-zfw5W7yP-utIh75Wrmaq4AHaE7.jpeg
  inflating: Trainingset/Wild animal/OIP-zgqo-Q3nZAJiB-4_4V_OUgHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-zjO3Dp6jg-zilWTaQ184UgAAAA.jpeg
  inflating: Trainingset/Wild animal/OIP-ZjrjjZM8y9QuxjNDA-PiGQHaFA.jpeg
  inflating: Trainingset/Wild animal/OIP-ZlsZOeXQTrzCVP73WfyTQgAAAA.jpeg
  inflating: Trainingset/Wild animal/OIP-ZmVr7z14oXAlYojDFOAWUwHaEU.jpeg
  inflating: Trainingset/Wild animal/OIP-zN7aZgPbLjN9SRlIuoJruAAAAA.jpeg
  inflating: Trainingset/Wild animal/OIP-ZOLNyUI9fOd-lnACXOdnWAHaFj.jpeg
  inflating: Trainingset/Wild animal/OIP-ZPRJ85QAzQ7hC4vp7vzErwHaE8.jpeg
  inflating: Trainingset/Wild animal/OIP-ZR-NuMDLR_C-1CHKM7ST8wHaE6.jpeg
  inflating: Trainingset/Wild animal/OIP-ZUZnitzqZNls7psG83OlywHaLI.jpeg

  inflating: Trainingset/Wild animal/OIP-ZVPwvKY3N2tqg00m_z92rQHaE8.jpeg
  inflating: Trainingset/Wild animal/OIP-Zw7kOwinhlE9mol5L4SJJQHaEo.jpeg
  inflating: Trainingset/Wild animal/OIP-zwsmvOpywbUHPDIiA9vLBgHaEK.jpeg
```

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

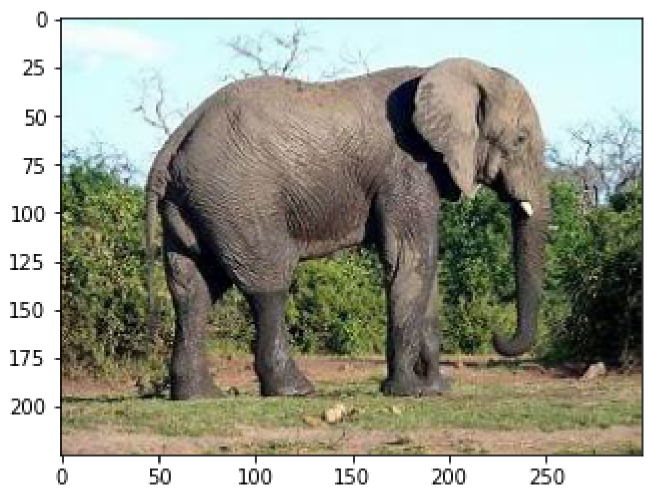
```
    Mounted at /content/drive
```

```
pip install twilio
```

```
    Collecting twilio
      Downloading https://files.pythonhosted.org/packages/5a/ee/65693a0094667b21a21ed273
         |████████████████████████████████| 481kB 4.1MB/s
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from t
    Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from
    Collecting PyJWT==1.7.1
      Downloading https://files.pythonhosted.org/packages/87/8b/6a9f14b5f781697e51259d81
    Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-p
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pa
    Building wheels for collected packages: twilio
      Building wheel for twilio (setup.py) ... done
      Created wheel for twilio: filename=twilio-6.58.0-py2.py3-none-any.whl size=1267711
      Stored in directory: /root/.cache/pip/wheels/0c/c3/36/584246f48bce8d3a8b314c5ecfe7
    Successfully built twilio
    Installing collected packages: PyJWT, twilio
    Successfully installed PyJWT-1.7.1 twilio-6.58.0
```

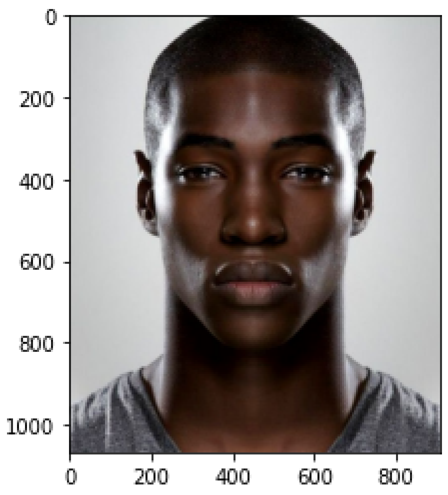
```
%pylab inline
import matplotlib.pyplot as plt
import matplotlib.image as nping
img=nping.imread('/content/Trainingset/Wild animal/OIP--3aF2OpzGKcdI6FHil50qQHaFj.jpeg')
imgplot=plt.imshow(img)
plt.show()
```

```
    Populating the interactive namespace from numpy and matplotlib
```



```
img=nping.imread('/content/Trainingset/Human/1 (1001).jpg')
```

```
imgplot=plt.imshow(img)
plt.show()
```

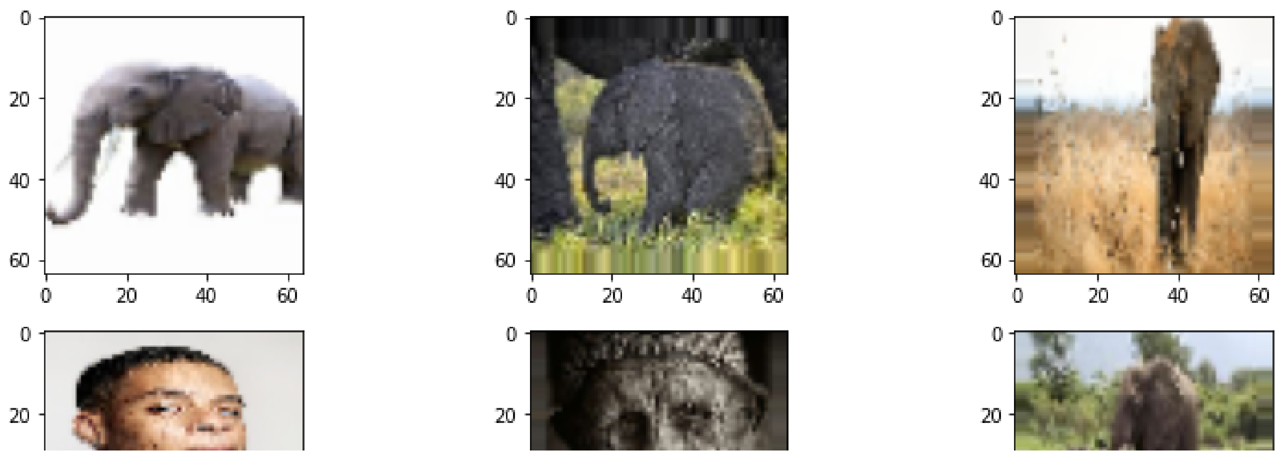


```
import tensorflow
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,Flatten,Dropout,MaxPooling2D,Activation
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import matplotlib.image as nping


img_width,img_height=150,150
train_datagen=r"/content/Trainingset"
validation_datagen=r"/content/Testset"
#nb_train_sample=100
#nb_validation_samples=100
#epochs=5
#batch_size=10



import tensorflow.keras.backend as k
if k.image_data_format()=='channels_first':
  input_shape=(3,img_width,img_height)
else:
  input_shape=(img_width,img_height,3)


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

train_set = train_datagen.flow_from_directory('/content/Trainingset',
                                              target_size = (64, 64),
                                              batch_size = 32,
                                              class_mode = 'binary')

#Test images
test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
test_set = test_datagen.flow_from_directory('/content/Testset',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 2527 images belonging to 2 classes.
Found 2527 images belonging to 2 classes.
```

```
plt.figure(figsize=(12,12))
for i in range(0,15):
  plt.subplot(5,3,i+1)
  for X_batch,Y_batch in train_set:
    image=X_batch[0]
    plt.imshow(image)
    break
plt.tight_layout()
plt.show()
```



```
model=Sequential()
model.add(Conv2D(64,(3,3),input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 64)      1792
_________________________________________________________________
activation (Activation)      (None, 148, 148, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 350464)            0
_________________________________________________________________
dense (Dense)                (None, 64)                22429760
_________________________________________________________________
activation_1 (Activation)    (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_2 (Activation)    (None, 1)                 0
=================================================================
Total params: 22,431,617
Trainable params: 22,431,617
Non-trainable params: 0
_________________________________________________________________
```

```
#Building CNN
#1. initializing CNN
import tensorflow as tf
cnn = tf.keras.models.Sequential()
#2. Convolution
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[

#3. Pooling
```

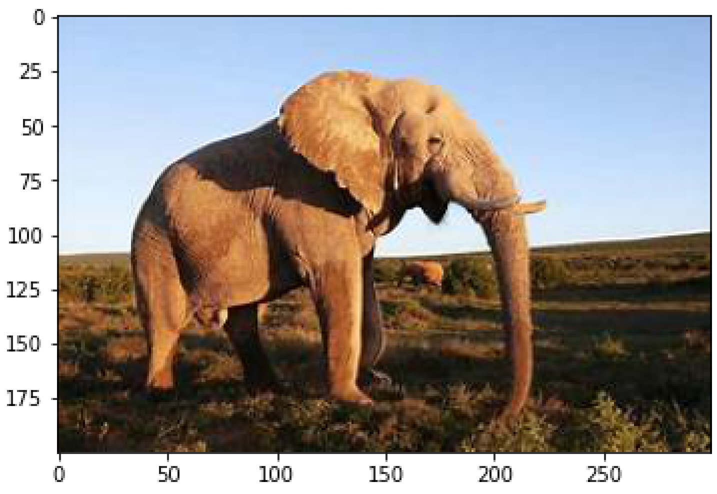
```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

#4. 2nd convolution layer
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

#5. Flattening
cnn.add(tf.keras.layers.Flatten())

#6. Full connection
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

#7. Output layer
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

#Training CNN
#CompilingCNN
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])



cnn.fit(x = train_set, validation_data = test_set, epochs = 25)
```

```
     Epoch 1/25
     79/79 [==============================] - 99s 890ms/step - loss: 0.4309 - accuracy: 0
     Epoch 2/25
     79/79 [==============================] - 70s 885ms/step - loss: 0.1524 - accuracy: 0
     Epoch 3/25
     79/79 [==============================] - 69s 881ms/step - loss: 0.1169 - accuracy: 0
     Epoch 4/25
     79/79 [==============================] - 70s 890ms/step - loss: 0.1223 - accuracy: 0
     Epoch 5/25
     79/79 [==============================] - 70s 876ms/step - loss: 0.0829 - accuracy: 0
     Epoch 6/25
     79/79 [==============================] - 70s 889ms/step - loss: 0.0721 - accuracy: 0
     Epoch 7/25
     79/79 [==============================] - 69s 880ms/step - loss: 0.0556 - accuracy: 0
     Epoch 8/25
     79/79 [==============================] - 69s 883ms/step - loss: 0.0509 - accuracy: 0
     Epoch 9/25
     79/79 [==============================] - 69s 881ms/step - loss: 0.0471 - accuracy: 0
     Epoch 10/25
     79/79 [==============================] - 69s 875ms/step - loss: 0.0532 - accuracy: 0
     Epoch 11/25
     79/79 [==============================] - 69s 877ms/step - loss: 0.0311 - accuracy: 0
     Epoch 12/25
     79/79 [==============================] - 69s 879ms/step - loss: 0.0359 - accuracy: 0
     Epoch 13/25
     79/79 [==============================] - 68s 872ms/step - loss: 0.0273 - accuracy: 0
     Epoch 14/25
     79/79 [==============================] - 69s 874ms/step - loss: 0.0323 - accuracy: 0
     Epoch 15/25
     79/79 [==============================] - 69s 879ms/step - loss: 0.0477 - accuracy: 0
     Epoch 16/25
     79/79 [==============================] - 69s 880ms/step - loss: 0.0321 - accuracy: 0
     Epoch 17/25
     79/79 [==============================] - 69s 876ms/step - loss: 0.0152 - accuracy: 0
     Epoch 18/25
     79/79 [==============================] - 69s 874ms/step - loss: 0.0416 - accuracy: 0
```

```
    Epoch 19/25
    79/79 [==============================] - 69s 877ms/step - loss: 0.0222 - accuracy: 0
    Epoch 20/25
    79/79 [==============================] - 69s 880ms/step - loss: 0.0152 - accuracy: 0
    Epoch 21/25
    79/79 [==============================] - 69s 882ms/step - loss: 0.0219 - accuracy: 0
    Epoch 22/25
    79/79 [==============================] - 68s 873ms/step - loss: 0.0381 - accuracy: 0
    Epoch 23/25
    79/79 [==============================] - 68s 873ms/step - loss: 0.0152 - accuracy: 0
    Epoch 24/25
    79/79 [==============================] - 69s 873ms/step - loss: 0.0248 - accuracy: 0
    Epoch 25/25
    79/79 [==============================] - 70s 890ms/step - loss: 0.0164 - accuracy: 0
    <tensorflow.python.keras.callbacks.History at 0x7f5130537b10>
```

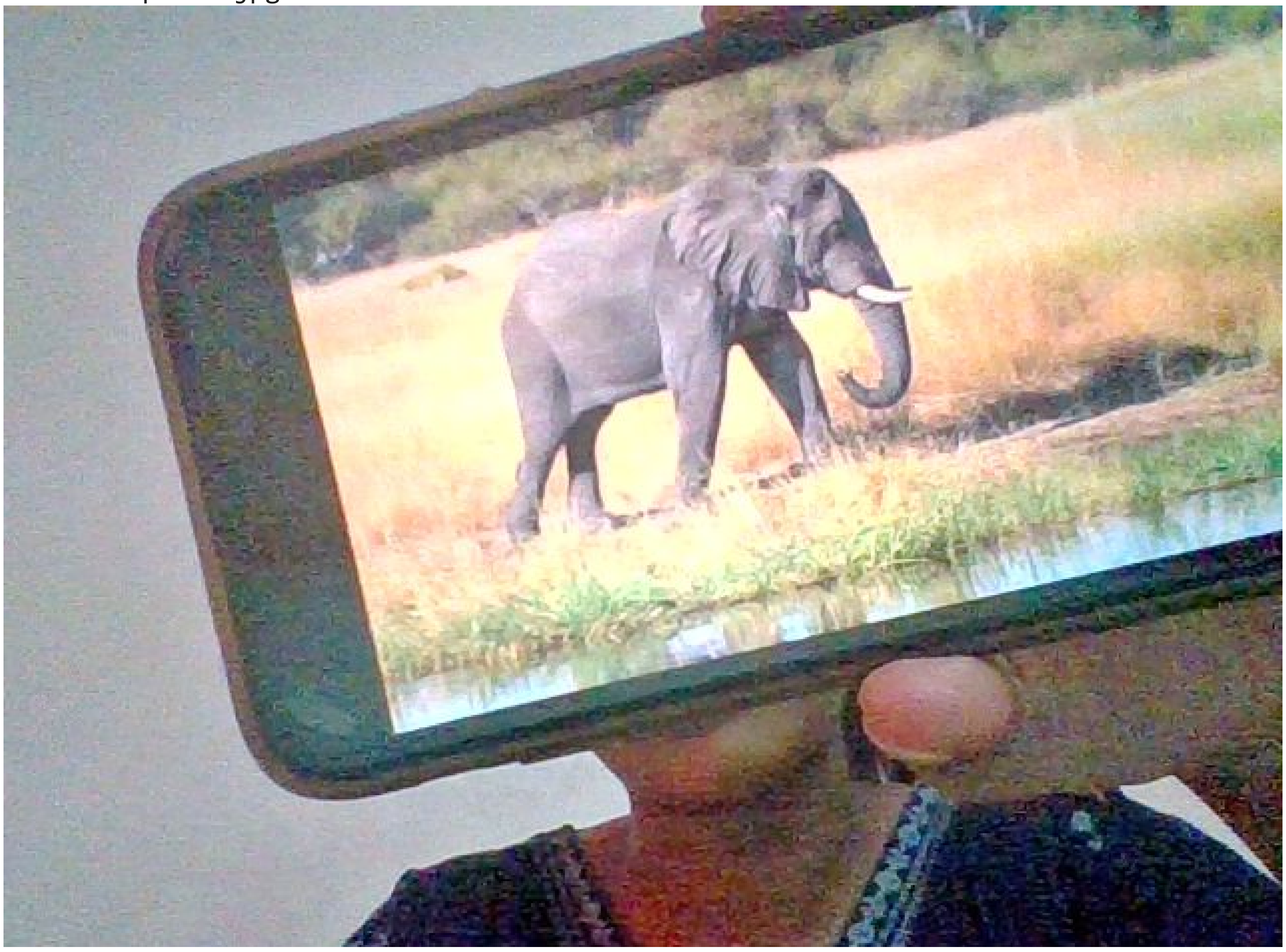
```
#results
result = cnn.predict(test_set)
k2 = np.argmax(result, axis = 1)

#Making one prediction
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('/content/Testset/wildanimal/OIP--J8rTRP-zHNg0b0dWPYdzwHaE7.jp
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
train_set.class_indices
if result[0][0] == 1:
  prediction = 'wild animal'
else:
   prediction = 'human'
print("prediction:",prediction)
img=nping.imread('/content/Testset/wildanimal/OIP--J8rTRP-zHNg0b0dWPYdzwHaE7.jpeg')
imgplot=plt.imshow(img)
plt.show()
```

```
    prediction: wild animal
```



```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename


from IPython.display import Image
try:
  filename = take_photo()
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they do not
  # grant the page permission to access it.
  print(str(err))
```

```
Saved to photo.jpg
```



```
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('photo.jpg', target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
train_set.class_indices
if result[0][0] == 1:
  prediction = 'human'
else:
   prediction = 'wild animal'
print("prediction",prediction)
if prediction == "wild animal":
  from twilio.rest import Client

# Your Account SID from twilio.com/console
  account_sid = "AC4286c279f8fd6380ee4bdffd0b337e2e"
# Your Auth Token from twilio.com/console
  auth_token  = "47a3676810517ff278db4c8815847455"

  client = Client(account_sid, auth_token)

  message = client.messages.create(
    to="+918897929782",
    from_="+19196705517",
    body="wild animal detected in your field")
  call = client.calls.create(
                      twiml='<Response><Say>Wild animal detected in your crop......... n
                      to='+918897929782'
```

```
                        to='+918897929782',
                        from_='+19196705517')

print(message.sid)
print(call.sid)

   prediction wild animal
   SMd778fa9c34da44cfadd4621b95890119
   CA8c8736d8b7e4c460ae5839065c37189a
```

✓ 34s completed at 9:38 PM