INTELLIGENT ALERT SYSTEM FOR FOREST TRIBAL PEOPLE USING IBM WATSON STUDIO

1. INTRODUCTION

Intelligent Alert System for Forest Tribal People Using IBM Watson Studio. This project aims to save the lives of tribal by notifying about wildlife predation with the help of Artificial Intelligence.

Tribal people constitute 8.6% of the nation's total population, and most of them spend the greater part of their lives in the proximity of trees and villages or clans near to the forest. There are so many challenges faced by these people out of which Human-wildlife conflict is a serious challenge undermining the protection of tribal regions. The major types of human-wildlife conflict in the area include crop-raiding, livestock predation, increased risk of livestock diseases, and direct threats to human life. so Active measures are to be implemented to mitigate these problems and safeguard the future of the wildlife. Hence, we came up with this Project "Intelligent Alert System". This ensures the complete safety of humans who lives near the forests by notifying the wild animal predation before it enters the clan.

1.1OVERVIEW

- Data collection
- Image pre processing
- Model building
- Video analysis
- Train CNN model on IBM

1.2 PURPOSE

- Tribal people constitute 8.6% of the nation's total population, and most of them spend the greater part of their lives in the proximity of trees and villages or clans near to the forest.
- There are so many challenges faced by these people out of which Human-wildlife conflict is a serious challenge undermining the protection of tribal regions.
- The major types of human-wildlife conflict in the area include crop-raiding, livestock predation, increased risk of livestock diseases, and direct threats to human life.
- Hence, we came up with this Project "Intelligent Alert System". This ensures the complete safety of humans who lives near the forests by notifying the wild animal predation before it enters the clan.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

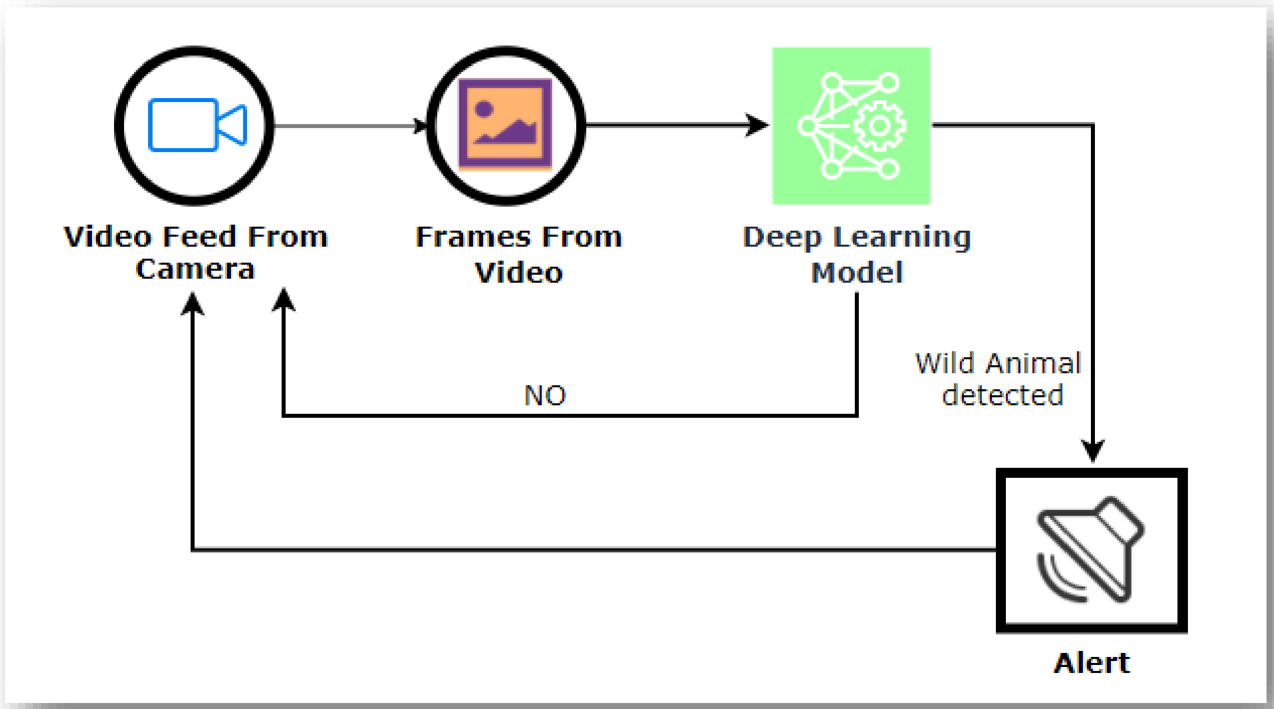
Agriculture is the most important sector of Indian Economy but the issue of damage to crops by wild creatures has turned into a noteworthy social issue in current occasions. So far there is no effective solution to this problem and therefore requires earnest consideration. This project provides a smart solution to resolve this problem. In this framework, image is captured when an animal intrudes and then image is classified as domestic or wild animal using Convolution Neural Network (CNN) and deep learning technique. This classification helps in alerting the farmer by alerting in case of intrusion of wild animal. The smart farm protection system gives reliable security and safety to crops. This system guarantees the wellbeing of creatures while warding them off It likewise diminishes the exertion made by man in securing the field and all the tribal populations of India were traditionally closely associated with forests, and there are some who even today spend the greater part of their lives in the proximity of trees and villages or clans near to forest. If any dangerous predators when entered into a village or clan may lead to loss of resources or in extreme cases leads to loss of life. Here we come up with an artificial intelligence-based technique which detects the animals before its gets enter into villages.

2.2 PROPOSED SYSTEM

The whole system can be divided in to three main part1. Animal classification and recognition from real time video.2. Alarm unit A. Real time Animal classification and recognition in the literature large amount of approaches have been proposed to accomplish the task of animal recognition that has different aims, strengths and limitations. Here new approach called "Transfer learning" is used for animal detection and recognition. Before going deep in to transfer learning need to know about neural network and Convolutional Neural Network. Pooling Layer, and Fully-Connected Layer. Stacking of these layers gives a full1) Convolutional neural network. A simple CNN is a sequence of layers. It mainly uses three main types of layers to build CNN architectures. That are Convolutional Layer, Convolutional neural network architecture. a) Convolutional Layer: The Convolution layer is the core building block of CNN that does most of the computations. The Convolution layer consist of a set of learnable filters Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a Convolution layer might have size 3 x 3 x 3 (i.e., 3 pixels width and height, and 3 colour channels). We convolve by sliding each filter across the width and height of the input volume and compute dot products between the filter coefficient.

3. THEORETICAL ANALYSIS

3.1 BLOCK DIAGRAM



3.2 HARDWARE AND SOFTWARE DESIGNING

➢ **Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It was created by Guido van Rossum, and first released on February 20, 1991. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

- **Anaconda Navigator**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, RStudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder.

- **Jupyter Notebook**

The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter. Jupyter Notebooks are a spin-off project from the IPython project, which used to have an I Python Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

- **Spyder**

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features. Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. Spyder is extensible with first-party and third-party plugins includes support for interactive tools for data inspection and embeds Python specific code. Spyder is also pre-installed in Anaconda Navigator, which is included in Anaconda.

- **Flask**

Web framework used for building. It is a web application framework written in python which will be running in local browser with a user interface. In this application, whenever the user interacts with UI and selects emoji, it will suggest the best and top movies of that genre to the user.

- **CNN**

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision.

- **Twilio**

Twilio is an American company based in San Francisco, California, which provides programmable communication tools for making and receiving phone calls, sending and receiving text messages, and performing other communication functions using its web service APIs.

- **TensorFlow**

TensorFlow is an end-to-end open-source platform for machine learning with a particular focus on deep neural networks. Deep learning is a subset of machine learning that involves the analysis of large-scale unstructured data. Deep learning differs from traditional machine learning in that the latter typically deals with structured data. TensorFlow boasts of a flexible and comprehensive collection of libraries, tools, and community resources.

- **Keras**

Keras runs on top of open-source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

- **Layersopen cv**

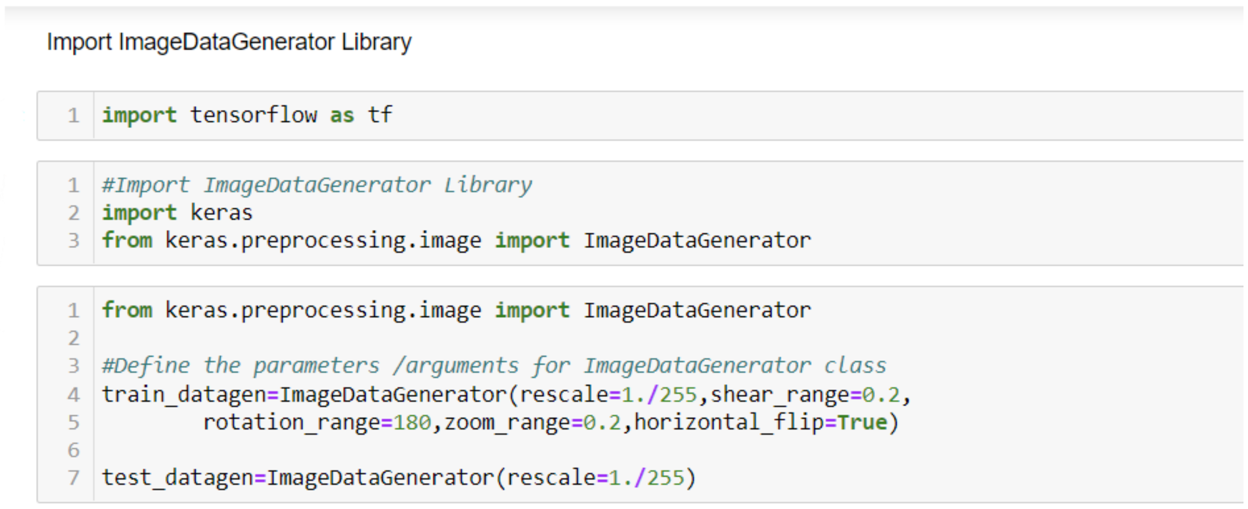
OpenCV is created to implement various operations including recognising and detecting faces, analysing human tasks in videos, identifying objects, recording camera movements, tracking moving objects, and combining images to create a high-resolution image for the accurate scene. Let's see the topic defining the term "Computer Vision."

❖ Hardware Requirements:

➢ Operating System	:	windows 7 and above with 64 bits
➢ Processor Type	:	Intel Core i3-3220 and above
➢ RAM	:	4Gb and above
➢ Hard Disk	:	minimum 100GB

4. EXPERIMENTAL INVESTIGATION

The text data need to be organized before proceeding with the project. The original dataset has a single folder. We will be using the HDI.csv file to fetch the text data of training data. The data need to be unique and all fields need to be filled. The dataset images are to be pre-processed before giving to the model. We will create a function that uses the pre-trained model for predicting custom outputs. Then we have to test and train the model. After the model is build, we will be integrating it to a web application.

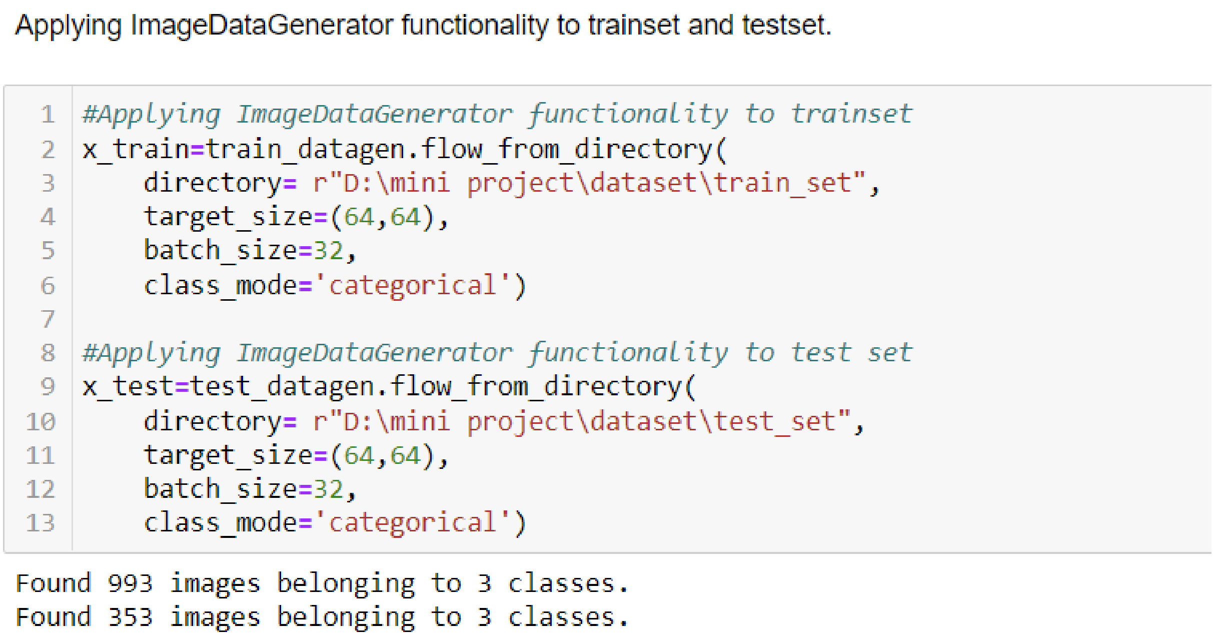


Import ImageDataGenerator Library

```
import tensorflow as tf
```

```
#Import ImageDataGenerator Library
import keras
from keras.preprocessing.image import ImageDataGenerator
```

```
from keras.preprocessing.image import ImageDataGenerator

#Define the parameters /arguments for ImageDataGenerator class
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,
        rotation_range=180,zoom_range=0.2,horizontal_flip=True)

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



Applying ImageDataGenerator functionality to trainset and testset.

```
#Applying ImageDataGenerator functionality to trainset
x_train=train_datagen.flow_from_directory(
    directory= r"D:\mini project\dataset\train_set",
    target_size=(64,64),
    batch_size=32,
    class_mode='categorical')

#Applying ImageDataGenerator functionality to test set
x_test=test_datagen.flow_from_directory(
    directory= r"D:\mini project\dataset\test_set",
    target_size=(64,64),
    batch_size=32,
    class_mode='categorical')
```

```
Found 993 images belonging to 3 classes.
Found 353 images belonging to 3 classes.
```

Importing the model building libraries

```
'''Importing the model building libraries'''
#to define linear initializations import Sequential
from keras.models import Sequential
#To add layers import Dense
from keras.layers import Dense
# to create a convolution kernel import Convolution2D
from keras.layers import Convolution2D
# Adding Max pooling Layer
from keras.layers import MaxPooling2D
# Adding Flatten Layer
from keras.layers import Flatten
```

Initializing the model

Initializing the model

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

Adding Convolutional Layer

```
# Adding CNN layers
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),
                        activation='relu'))
```

Adding Max pooling Layer

```
# Adding Max pooling Layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

Adding Flatten Layer

```
# Adding Flatten Layer
model.add(Flatten())
```

Adding Dense layers

```python
# Adding Hidden Layers
model.add(Dense( kernel_initializer='uniform',activation='relu',units=300))
```

```python
# Adding 2nd hidden layer
model.add(Dense( kernel_initializer='uniform',activation='relu',units=100))
```

```python
# Adding 3rd hidden layer
model.add(Dense( kernel_initializer='uniform',activation='relu',units=60))
```

Adding output layer

```python
# Adding output layer
model.add(Dense( kernel_initializer='uniform',activation='softmax',units=3))
```

Configure the learning process or compile the model

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

Configure the learning process or compile the model

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

Training the model

```
# Training the model
model.fit_generator(x_train,steps_per_epoch=31,
                    epochs=15,
                    validation_data=x_test,
                    validation_steps=11)
```

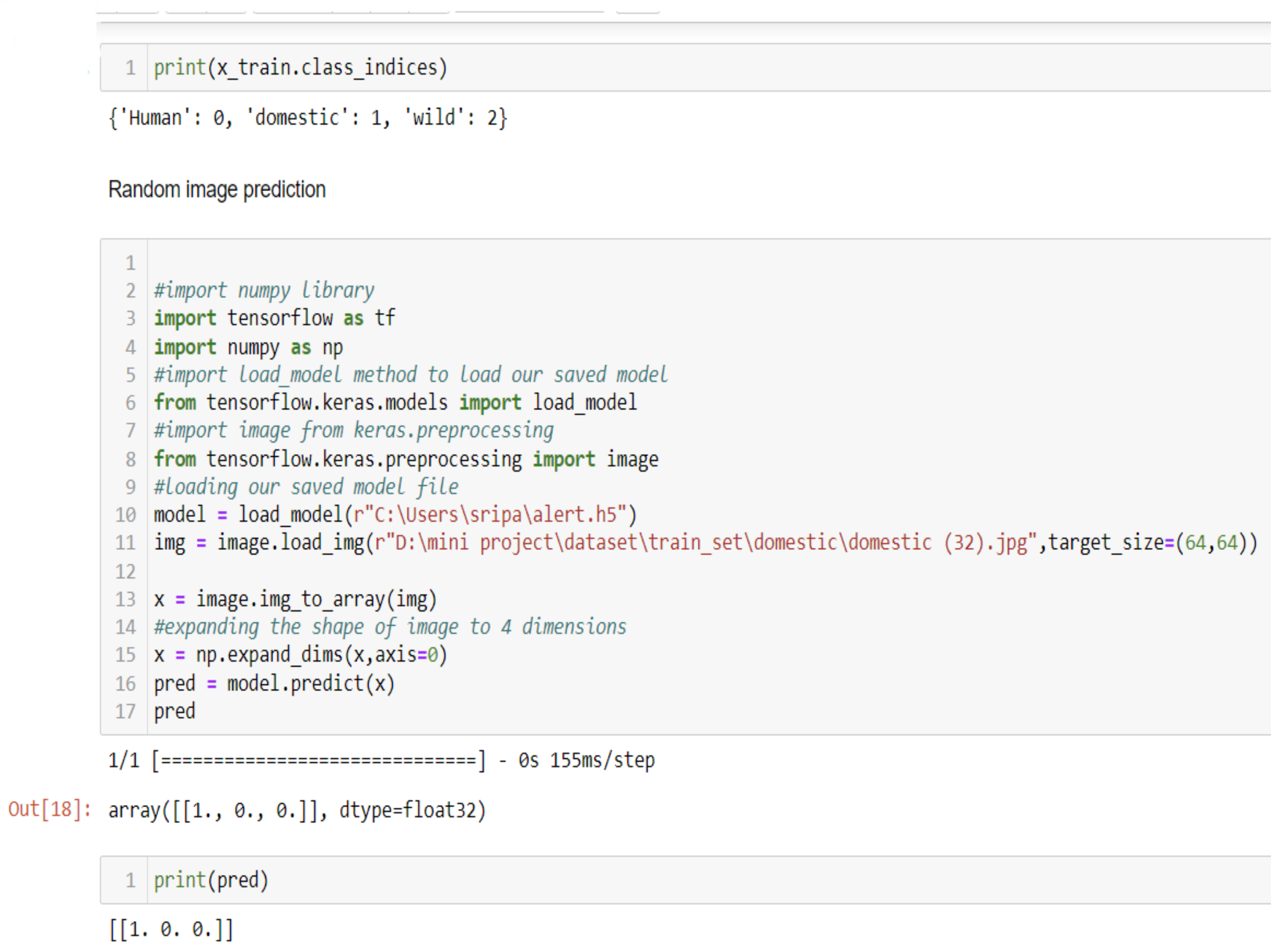
```
C:\Users\sripa\AppData\Local\Temp\ipykernel_17344\4250337181.py:2: UserWarning: `Model.fit_generator` is deprecated and will be
removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(x_train,steps_per_epoch=31,
```

```
Epoch 1/15
31/31 [==============================] - 10s 284ms/step - loss: 1.0905 - accuracy: 0.3600 - val_loss: 1.0201 - val_accuracy: 0.
4233
Epoch 2/15
31/31 [==============================] - 8s 247ms/step - loss: 1.0786 - accuracy: 0.4058 - val_loss: 1.0096 - val_accuracy: 0.4
489
Epoch 3/15
31/31 [==============================] - 7s 225ms/step - loss: 1.0122 - accuracy: 0.4370 - val_loss: 1.0001 - val_accuracy: 0.4
602
Epoch 4/15
31/31 [==============================] - 7s 246ms/step - loss: 0.9559 - accuracy: 0.4984 - val_loss: 1.0304 - val_accuracy: 0.4
176
Epoch 5/15
31/31 [==============================] - 7s 238ms/step - loss: 0.9537 - accuracy: 0.5203 - val_loss: 0.9702 - val_accuracy: 0.4
801
Epoch 6/15
31/31 [==============================] - 7s 219ms/step - loss: 1.0469 - accuracy: 0.4391 - val_loss: 1.0494 - val_accuracy: 0.5
114
Epoch 7/15
31/31 [==============================] - 7s 221ms/step - loss: 0.9584 - accuracy: 0.5036 - val_loss: 0.9518 - val_accuracy: 0.5
341

Epoch 8/15
31/31 [==============================] - 7s 236ms/step - loss: 0.9424 - accuracy: 0.5432 - val_loss: 0.9217 - val_accuracy: 0.5
227
Epoch 9/15
31/31 [==============================] - 7s 222ms/step - loss: 0.8596 - accuracy: 0.5827 - val_loss: 0.9354 - val_accuracy: 0.5
511
Epoch 10/15
31/31 [==============================] - 7s 225ms/step - loss: 0.8503 - accuracy: 0.5723 - val_loss: 1.2838 - val_accuracy: 0.3
977
Epoch 11/15
31/31 [==============================] - 7s 228ms/step - loss: 0.8321 - accuracy: 0.6004 - val_loss: 0.9756 - val_accuracy: 0.5
398
Epoch 12/15
31/31 [==============================] - 7s 221ms/step - loss: 0.8401 - accuracy: 0.5765 - val_loss: 0.8945 - val_accuracy: 0.5
256
Epoch 13/15
31/31 [==============================] - 7s 226ms/step - loss: 0.8653 - accuracy: 0.5806 - val_loss: 1.0613 - val_accuracy: 0.5
511
Epoch 14/15
31/31 [==============================] - 7s 223ms/step - loss: 0.8085 - accuracy: 0.6202 - val_loss: 0.9650 - val_accuracy: 0.5
568
Epoch 15/15
31/31 [==============================] - 7s 222ms/step - loss: 0.7988 - accuracy: 0.6275 - val_loss: 0.9516 - val_accuracy: 0.5
795
```

```
<keras.callbacks.History at 0x2121c0432e0>
```

Save the trained the model

```
#save model
model.save('alert.h5')
```



```python
print(x_train.class_indices)
```

```
{'Human': 0, 'domestic': 1, 'wild': 2}
```

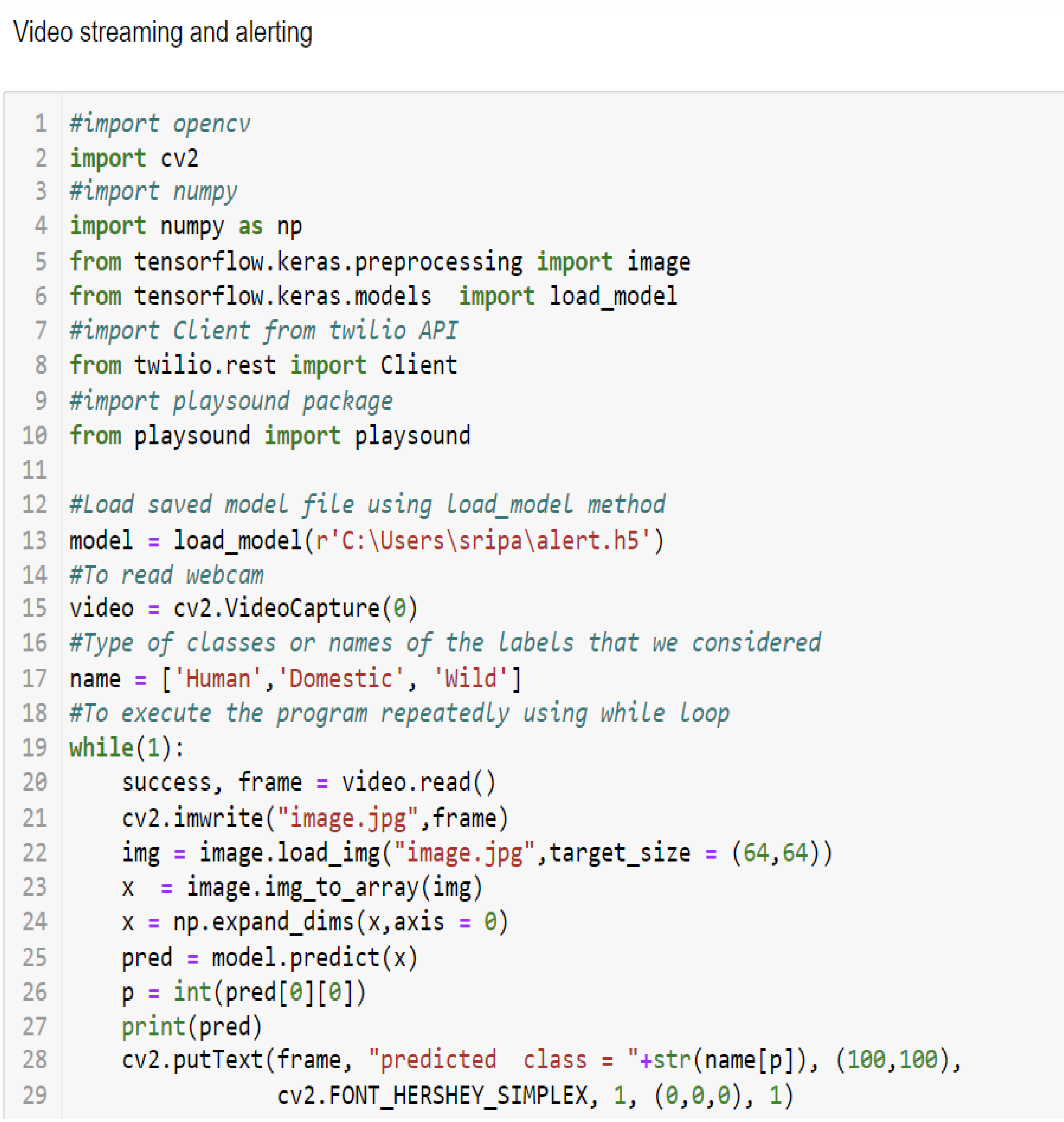
Random image prediction

```python

#import numpy library
import tensorflow as tf
import numpy as np
#import load_model method to load our saved model
from tensorflow.keras.models import load_model
#import image from keras.preprocessing
from tensorflow.keras.preprocessing import image
#loading our saved model file
model = load_model(r"C:\Users\sripa\alert.h5")
img = image.load_img(r"D:\mini project\dataset\train_set\domestic\domestic (32).jpg",target_size=(64,64))

x = image.img_to_array(img)
#expanding the shape of image to 4 dimensions
x = np.expand_dims(x,axis=0)
pred = model.predict(x)
pred
```

```
1/1 [==============================] - 0s 155ms/step
Out[18]: array([[1., 0., 0.]], dtype=float32)
```

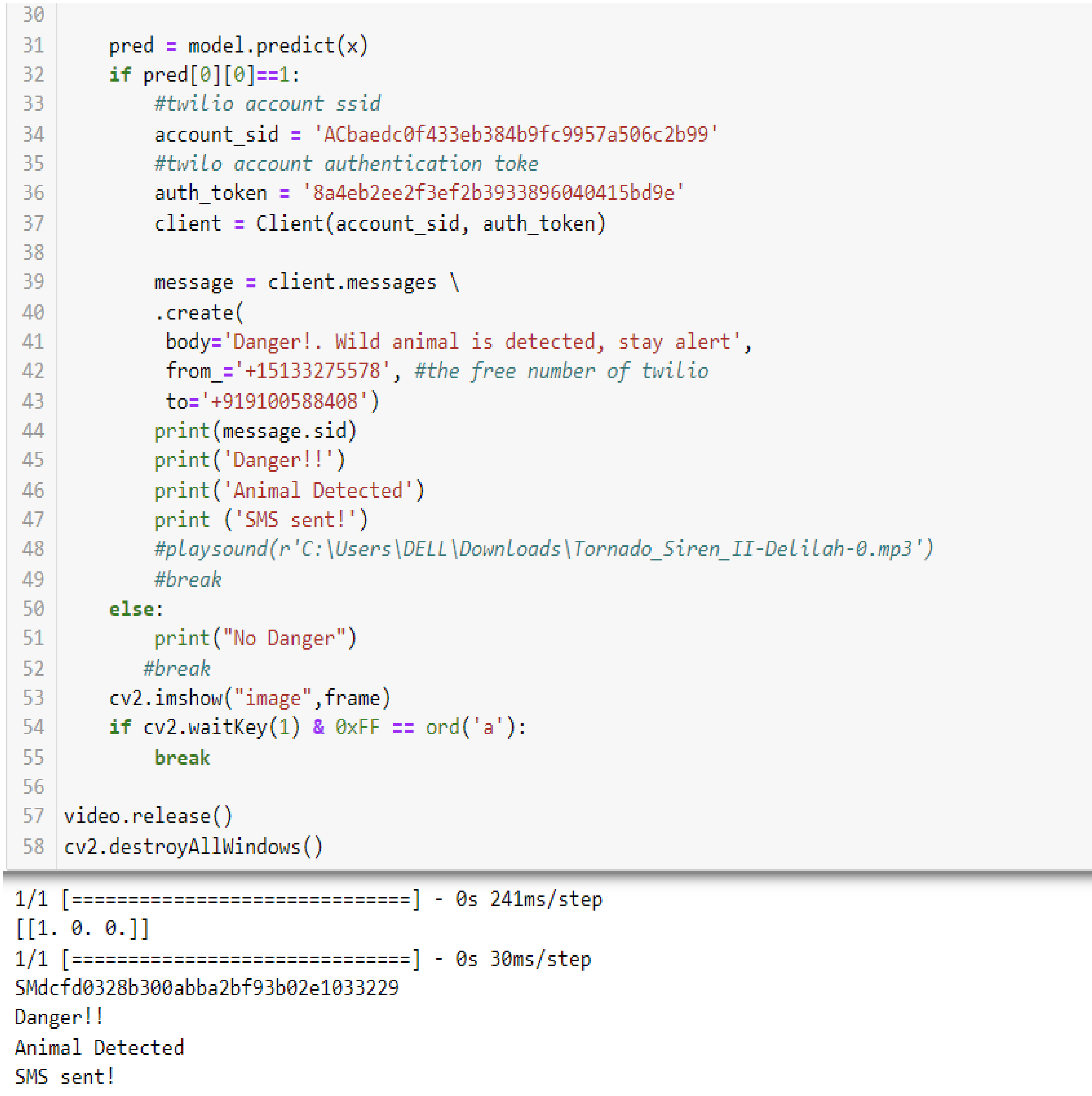
```python
print(pred)
```

```
[[1. 0. 0.]]
```



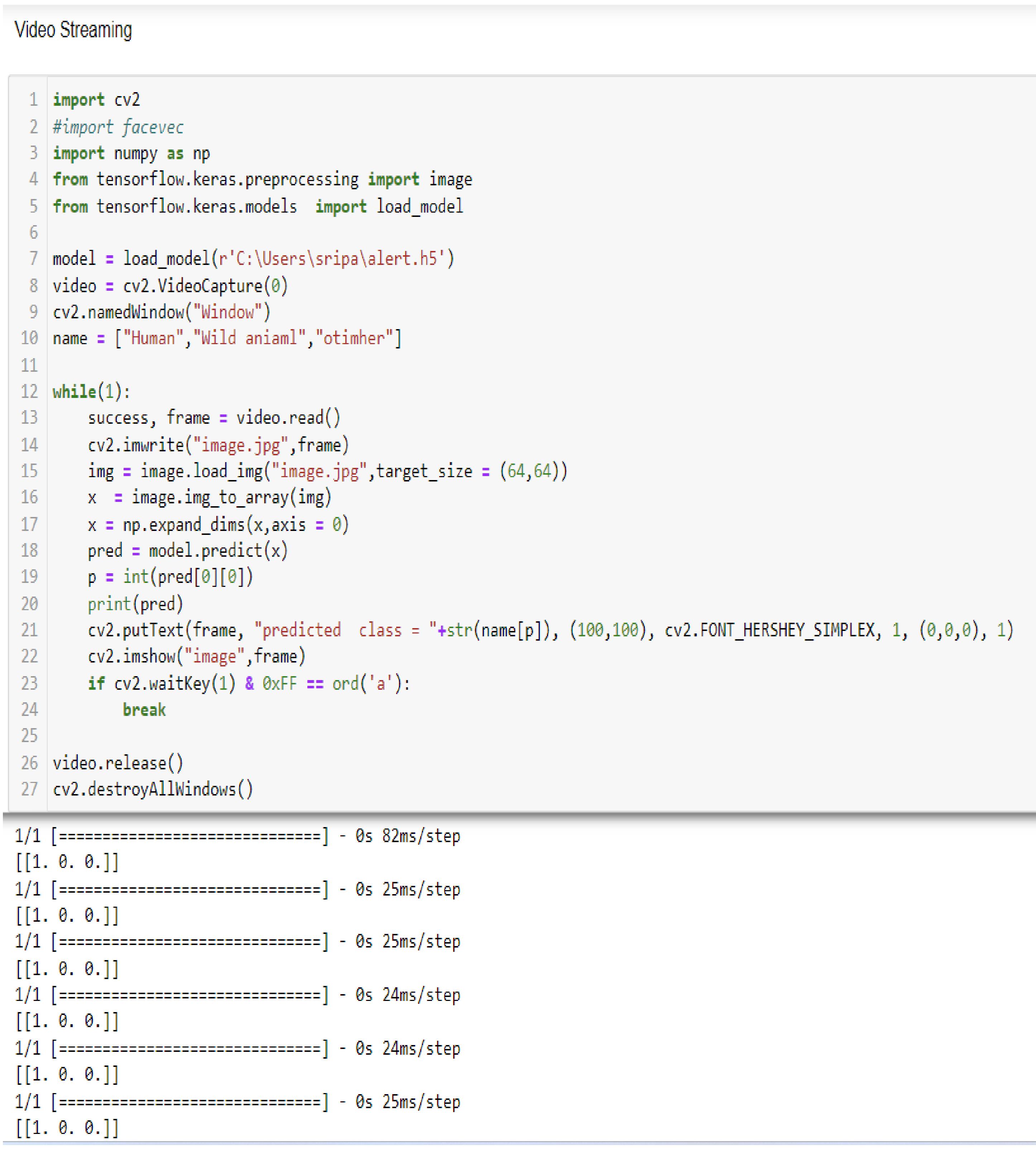
Video streaming and alerting

```python
#import opencv
import cv2
#import numpy
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models  import load_model
#import Client from twilio API
from twilio.rest import Client
#import playsound package
from playsound import playsound

#Load saved model file using load_model method
model = load_model(r'C:\Users\sripa\alert.h5')
#To read webcam
video = cv2.VideoCapture(0)
#Type of classes or names of the labels that we considered
name = ['Human','Domestic', 'Wild']
#To execute the program repeatedly using while loop
while(1):
    success, frame = video.read()
    cv2.imwrite("image.jpg",frame)
    img = image.load_img("image.jpg",target_size = (64,64))
    x  = image.img_to_array(img)
    x = np.expand_dims(x,axis = 0)
    pred = model.predict(x)
    p = int(pred[0][0])
    print(pred)
    cv2.putText(frame, "predicted  class = "+str(name[p]), (100,100),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)
```



```python

    pred = model.predict(x)
    if pred[0][0]==1:
        #twilio account ssid
        account_sid = 'ACbaedc0f433eb384b9fc9957a506c2b99'
        #twilo account authentication toke
        auth_token = '8a4eb2ee2f3ef2b3933896040415bd9e'
        client = Client(account_sid, auth_token)

        message = client.messages \
        .create(
         body='Danger!. Wild animal is detected, stay alert',
         from_='+15133275578', #the free number of twilio
         to='+919100588408')
        print(message.sid)
        print('Danger!!')
        print('Animal Detected')
        print ('SMS sent!')
        #playsound(r'C:\Users\DELL\Downloads\Tornado_Siren_II-Delilah-0.mp3')
        #break
    else:
        print("No Danger")
       #break
    cv2.imshow("image",frame)
    if cv2.waitKey(1) & 0xFF == ord('a'):
        break

video.release()
cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 241ms/step
[[1. 0. 0.]]
1/1 [==============================] - 0s 30ms/step
SMdcfd0328b300abba2bf93b02e1033229
Danger!!
Animal Detected
SMS sent!
```

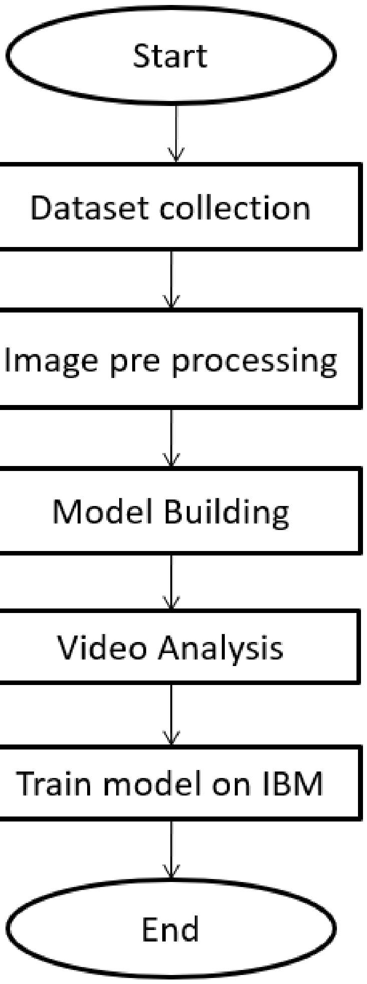


Video Streaming

```python
import cv2
#import facevec
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models  import load_model

model = load_model(r'C:\Users\sripa\alert.h5')
video = cv2.VideoCapture(0)
cv2.namedWindow("Window")
name = ["Human","Wild aniaml","otimher"]

while(1):
    success, frame = video.read()
    cv2.imwrite("image.jpg",frame)
    img = image.load_img("image.jpg",target_size = (64,64))
    x  = image.img_to_array(img)
    x = np.expand_dims(x,axis = 0)
    pred = model.predict(x)
    p = int(pred[0][0])
    print(pred)
    cv2.putText(frame, "predicted  class = "+str(name[p]), (100,100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 1)
    cv2.imshow("image",frame)
    if cv2.waitKey(1) & 0xFF == ord('a'):
        break

video.release()
cv2.destroyAllWindows()
```

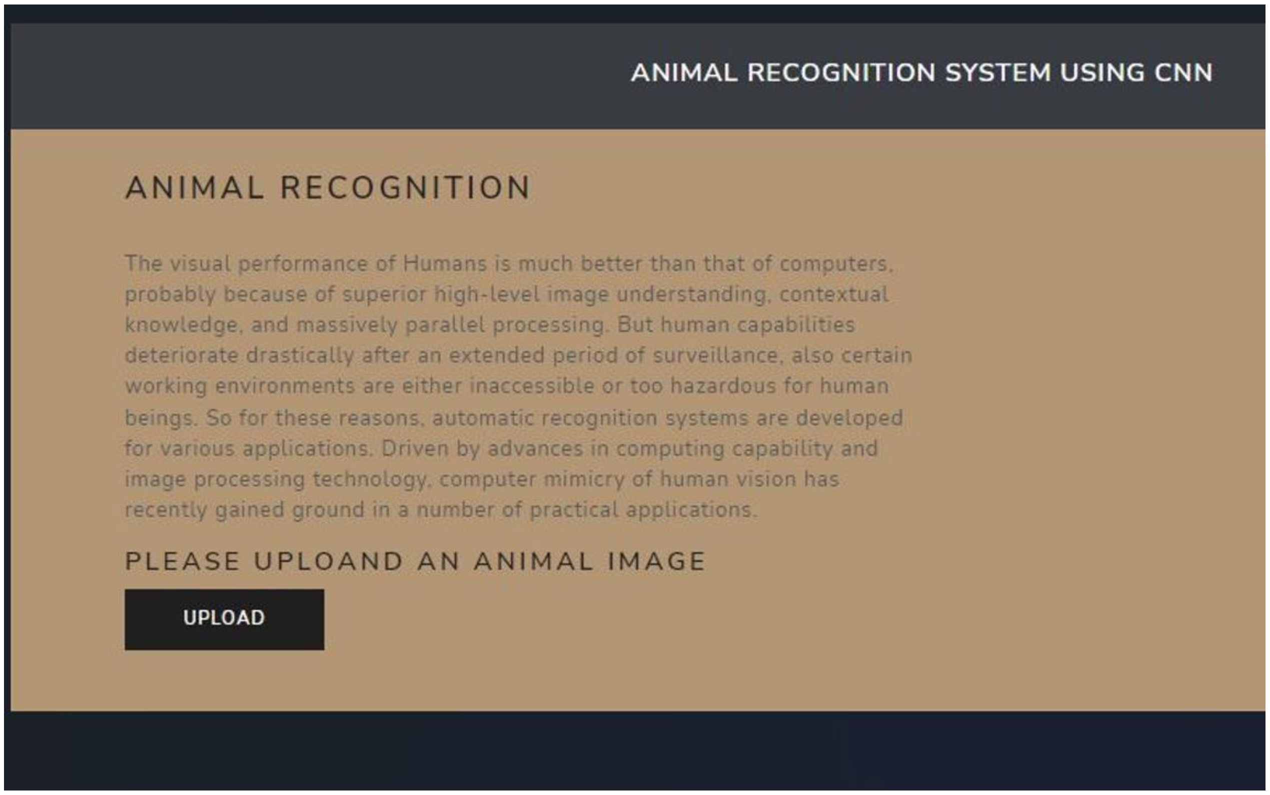
```
1/1 [==============================] - 0s 82ms/step
[[1. 0. 0.]]
1/1 [==============================] - 0s 25ms/step
[[1. 0. 0.]]
1/1 [==============================] - 0s 25ms/step
[[1. 0. 0.]]
1/1 [==============================] - 0s 24ms/step
[[1. 0. 0.]]
1/1 [==============================] - 0s 24ms/step
[[1. 0. 0.]]
1/1 [==============================] - 0s 25ms/step
[[1. 0. 0.]]
```

5. FLOWCHART


Start
Dataset collection
Image pre processing
Model Building
Video Analysis
Train model on IBM
End

6.RESULT

As the first step datasets are collected. We know that the image taken from the cameras in the forest contain animal coming in different orientation. So, the system should detect animal in any orientation. Data augmentation is performed for each images.

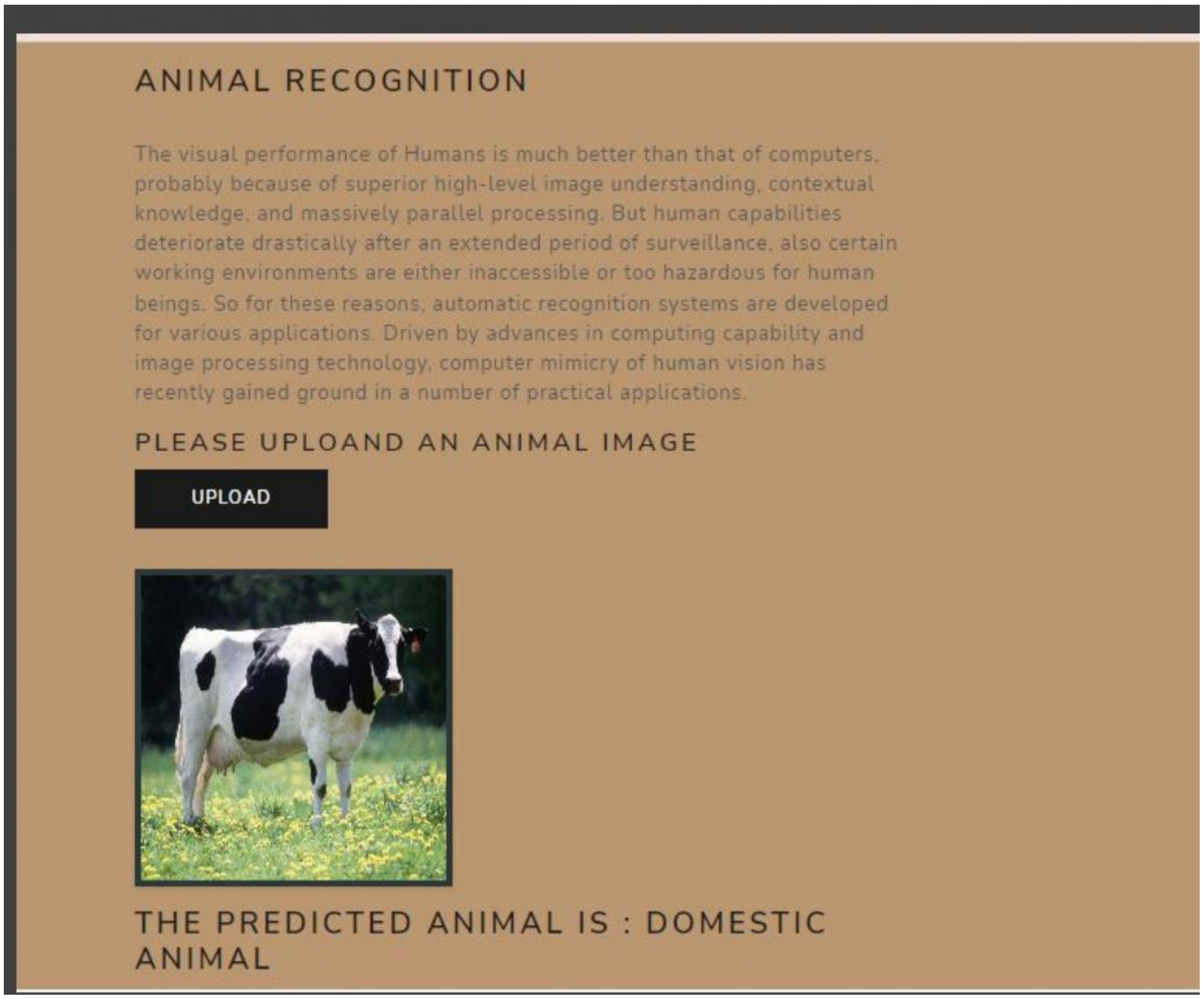


ANIMAL RECOGNITION SYSTEM USING CNN

ANIMAL RECOGNITION

The visual performance of Humans is much better than that of computers, probably because of superior high-level image understanding, contextual knowledge, and massively parallel processing. But human capabilities deteriorate drastically after an extended period of surveillance, also certain working environments are either inaccessible or too hazardous for human beings. So for these reasons, automatic recognition systems are developed for various applications. Driven by advances in computing capability and image processing technology, computer mimicry of human vision has recently gained ground in a number of practical applications.

PLEASE UPLOAND AN ANIMAL IMAGE



ANIMAL RECOGNITION

The visual performance of Humans is much better than that of computers, probably because of superior high-level image understanding, contextual knowledge, and massively parallel processing. But human capabilities deteriorate drastically after an extended period of surveillance, also certain working environments are either inaccessible or too hazardous for human beings. So for these reasons, automatic recognition systems are developed for various applications. Driven by advances in computing capability and image processing technology, computer mimicry of human vision has recently gained ground in a number of practical applications.

PLEASE UPLOAND AN ANIMAL IMAGE

THE PREDICTED ANIMAL IS : DOMESTIC ANIMAL

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Features are automatically deduced and optimally tuned for desired outcome. Features are not required to be extracted ahead of time.
- The same neural network-based approach can be applied to many different application and data type.

DISADVANTAGES

- Screen is difficult to read in low light or bright sunlight
- Keeping track of cameras and images (no meta-data)
- Losing settings during transit
- Walk-by test requires downloading image on laptop in the field.

8. APPLICATIONS

- Object Classification and Detection in Photographs
- Automatic Handwriting Generation
- Character Text Generation
- Automatic Machine Translation
- Image Caption Generation
- Automatic Game Playing.

9. CONCLUSION

The paper discusses in detail all advances in the area of automated wildlife monitoring. Animal detection methods are very useful for many real time applications. Automated animal detection, localization, and species recognition lie in the heart of automated camera-trap image analysis. Animal detection approaches are helpful to prevent dangerous situations caused by the wild animals in residential area. Detection of animal in the border region helps the people living in the border region to take safety precaution. Hence this system ensures the security of both wild animal and human beings in the border region.

10. FUTURE SCOPE

This work can be further extended by sending an alert in the form of a message when the animal is detected to the nearby forest office. Furthermore, it can be used to reduce human wildlife conflict and also animal accidents.

11.BIBILOGRAPHY

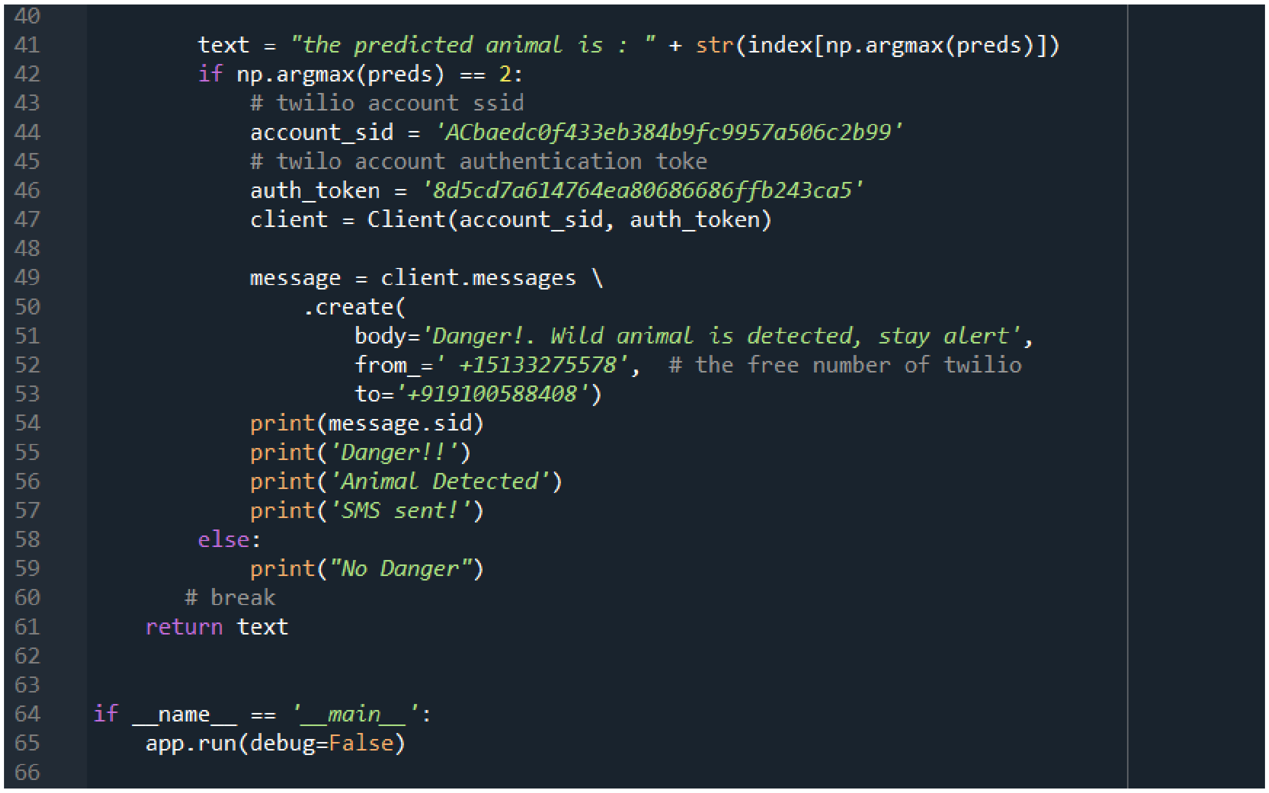
1.www.google.com

2.www.wikipedia.org

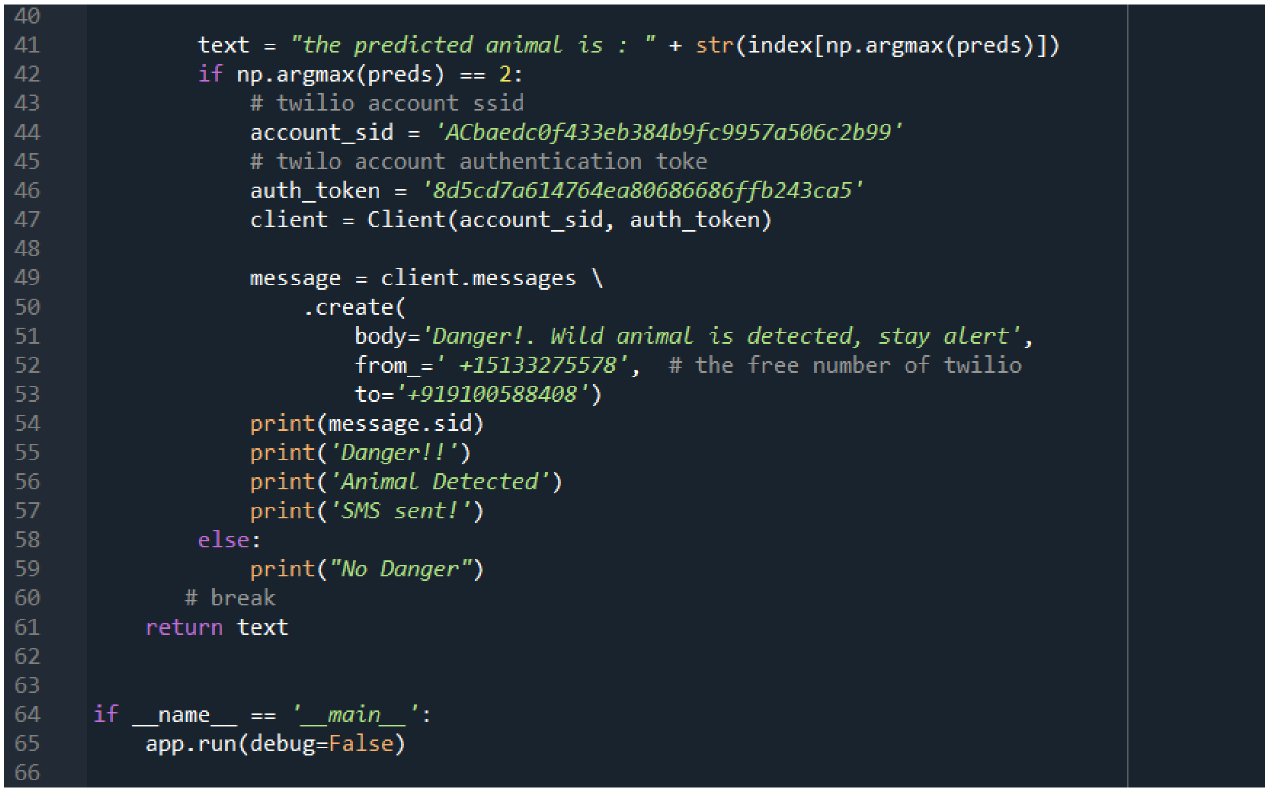
3.https://ieeexplore.ieee.org

12.APPENDIX

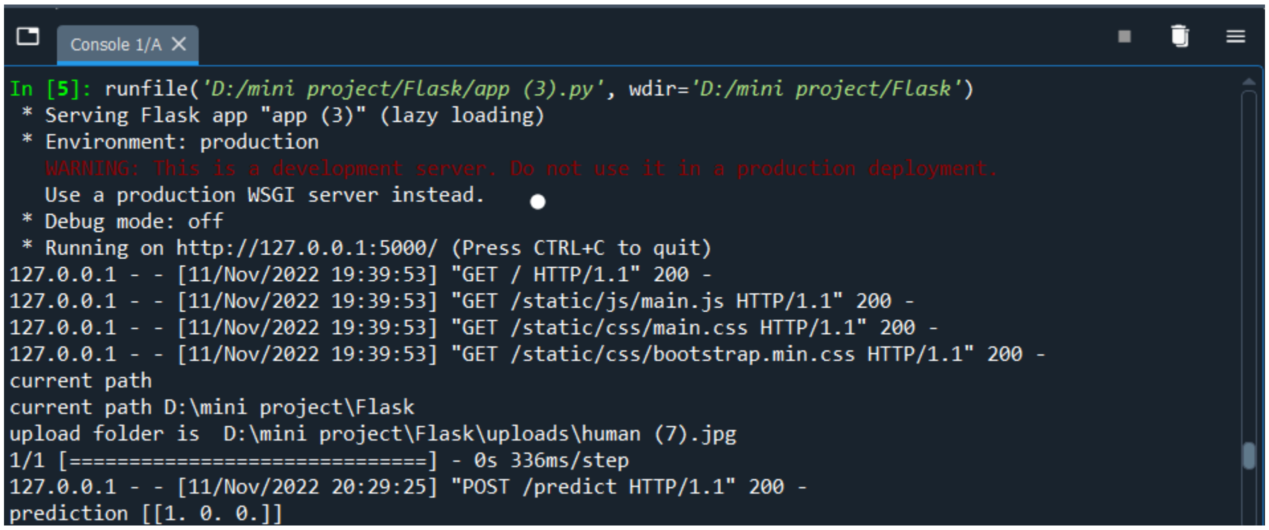
SOURCE CODE OF FLASK



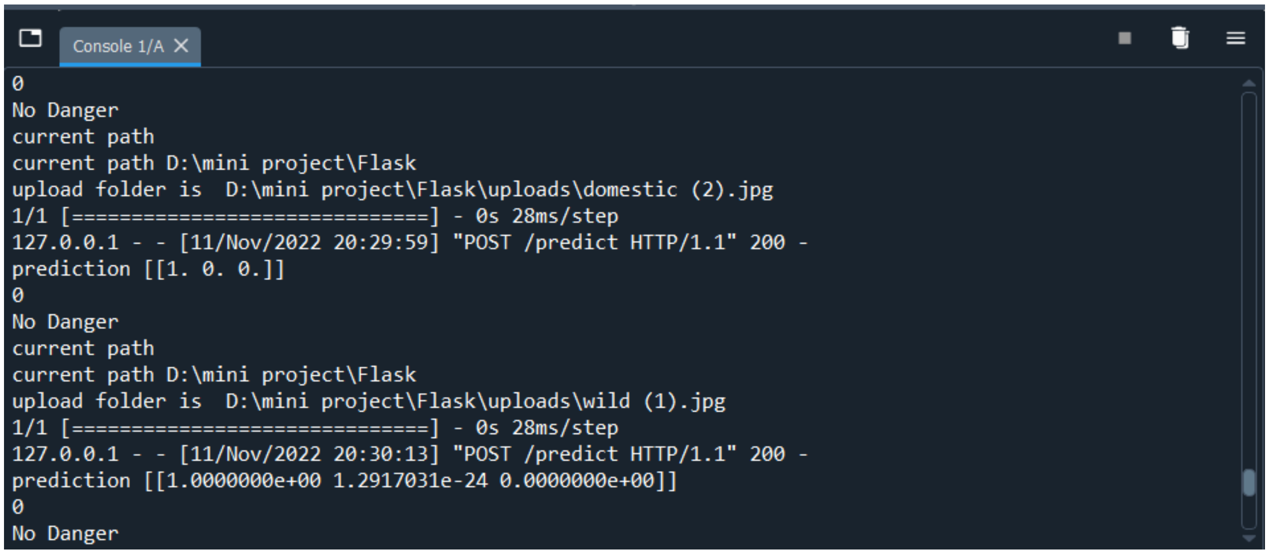
```python

        text = "the predicted animal is : " + str(index[np.argmax(preds)])
        if np.argmax(preds) == 2:
            # twilio account ssid
            account_sid = 'ACbaedc0f433eb384b9fc9957a506c2b99'
            # twilo account authentication toke
            auth_token = '8d5cd7a614764ea80686686ffb243ca5'
            client = Client(account_sid, auth_token)

            message = client.messages \
                .create(
                    body='Danger!. Wild animal is detected, stay alert',
                    from_=' +15133275578',  # the free number of twilio
                    to='+919100588408')
            print(message.sid)
            print('Danger!!')
            print('Animal Detected')
            print('SMS sent!')
        else:
            print("No Danger")
       # break
    return text


if __name__ == '__main__':
    app.run(debug=False)

```


```
        text = "the predicted animal is : " + str(index[np.argmax(preds)])
        if np.argmax(preds) == 2:
            # twilio account ssid
            account_sid = 'ACbaedc0f433eb384b9fc9957a506c2b99'
            # twilo account authentication toke
            auth_token = '8d5cd7a614764ea80686686ffb243ca5'
            client = Client(account_sid, auth_token)

            message = client.messages \
                .create(
                    body='Danger!. Wild animal is detected, stay alert',
                    from_=' +15133275578',  # the free number of twilio
                    to='+919100588408')
            print(message.sid)
            print('Danger!!')
            print('Animal Detected')
            print('SMS sent!')
        else:
            print("No Danger")
       # break
    return text


if __name__ == '__main__':
    app.run(debug=False)

```

CONSOLE:

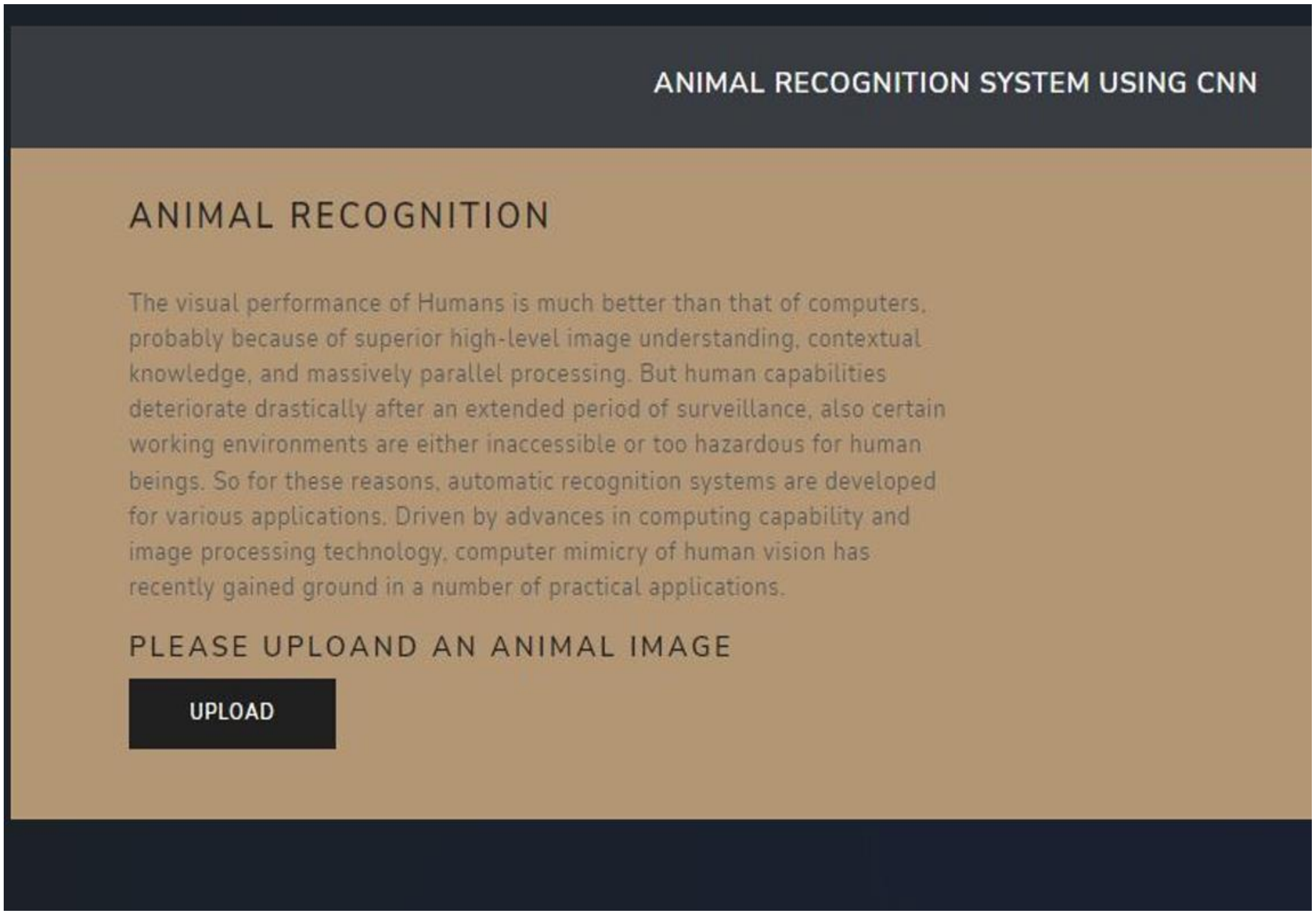


```
In [5]: runfile('D:/mini project/Flask/app (3).py', wdir='D:/mini project/Flask')
 * Serving Flask app "app (3)" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [11/Nov/2022 19:39:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2022 19:39:53] "GET /static/js/main.js HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2022 19:39:53] "GET /static/css/main.css HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2022 19:39:53] "GET /static/css/bootstrap.min.css HTTP/1.1" 200 -
current path
current path D:\mini project\Flask
upload folder is  D:\mini project\Flask\uploads\human (7).jpg
1/1 [==============================] - 0s 336ms/step
127.0.0.1 - - [11/Nov/2022 20:29:25] "POST /predict HTTP/1.1" 200 -
prediction [[1. 0. 0.]]
```



```
0
No Danger
current path
current path D:\mini project\Flask
upload folder is  D:\mini project\Flask\uploads\domestic (2).jpg
1/1 [==============================] - 0s 28ms/step
127.0.0.1 - - [11/Nov/2022 20:29:59] "POST /predict HTTP/1.1" 200 -
prediction [[1. 0. 0.]]
0
No Danger
current path
current path D:\mini project\Flask
upload folder is  D:\mini project\Flask\uploads\wild (1).jpg
1/1 [==============================] - 0s 28ms/step
127.0.0.1 - - [11/Nov/2022 20:30:13] "POST /predict HTTP/1.1" 200 -
prediction [[1.0000000e+00 1.2917031e-24 0.0000000e+00]]
0
No Danger
```

OUTPUT

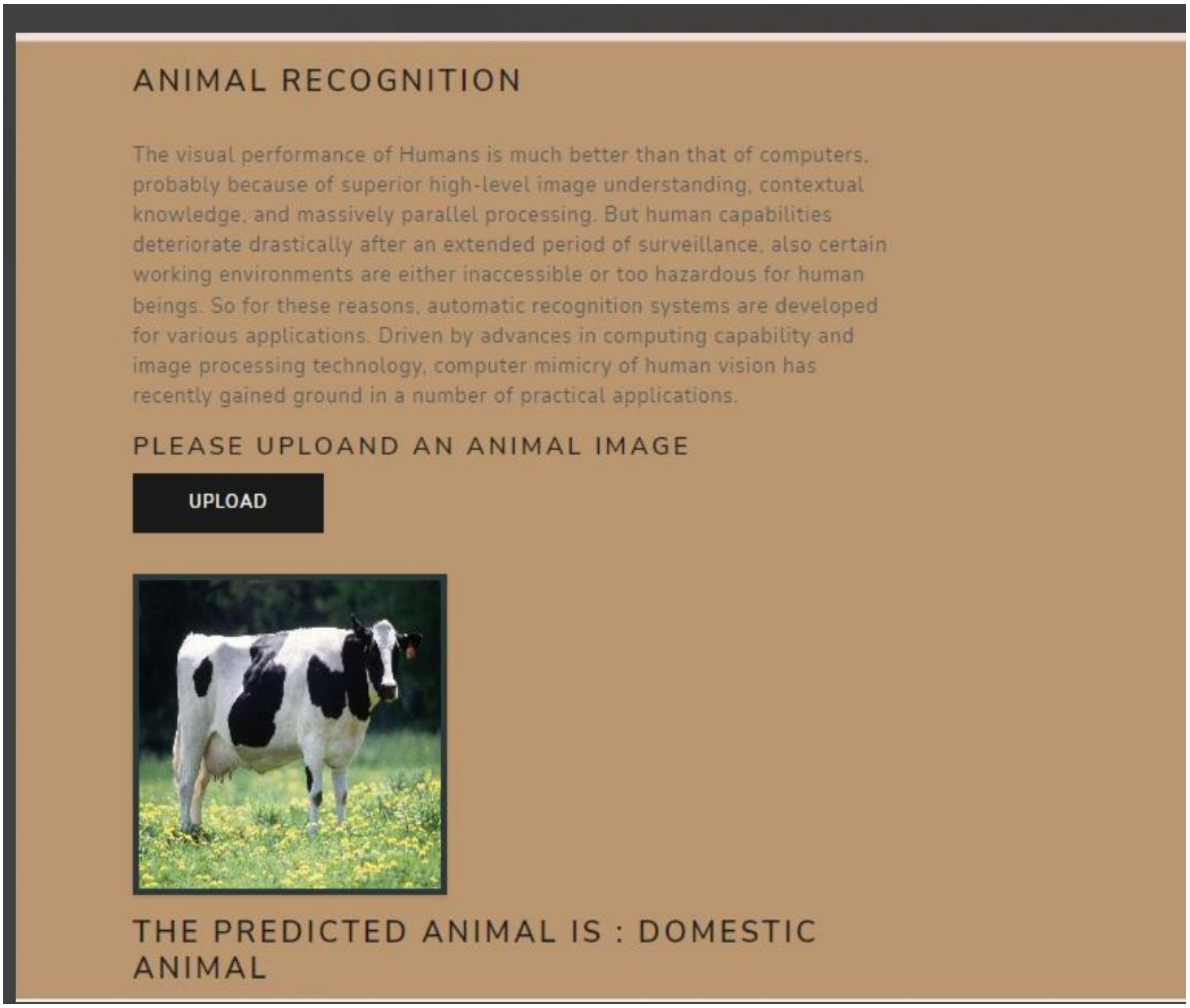


ANIMAL RECOGNITION SYSTEM USING CNN

ANIMAL RECOGNITION

The visual performance of Humans is much better than that of computers, probably because of superior high-level image understanding, contextual knowledge, and massively parallel processing. But human capabilities deteriorate drastically after an extended period of surveillance, also certain working environments are either inaccessible or too hazardous for human beings. So for these reasons, automatic recognition systems are developed for various applications. Driven by advances in computing capability and image processing technology, computer mimicry of human vision has recently gained ground in a number of practical applications.

PLEASE UPLOAND AN ANIMAL IMAGE



ANIMAL RECOGNITION

The visual performance of Humans is much better than that of computers, probably because of superior high-level image understanding, contextual knowledge, and massively parallel processing. But human capabilities deteriorate drastically after an extended period of surveillance, also certain working environments are either inaccessible or too hazardous for human beings. So for these reasons, automatic recognition systems are developed for various applications. Driven by advances in computing capability and image processing technology, computer mimicry of human vision has recently gained ground in a number of practical applications.

PLEASE UPLOAND AN ANIMAL IMAGE

THE PREDICTED ANIMAL IS : DOMESTIC ANIMAL