1. INTRODUCTION:

Deep learning is a subset of machine learning. In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. All the tribal populationof 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.

The proposed Intelligent alert system for forest tribal people model is based on neural networks (CNN) incorporated with an alerting system. This system will monitor the entire villages of surrounding forests at regular intervals through a camera. Once any dangerous animal is detected then it will send information to the people in the village and it will produce an appropriate sound or alarm in the village to alert the people.

1.1. Scope:

Human–wildlife conflict is a serious challenge undermining the protection of tribal regions. The major types of human–wildlife conflict are crop raiding, livestock predation, increased risk of livestock diseases and direct threats to human life. Active measures are to be implemented to mitigate these problems and safeguard the future of the wildlife. Hence we came up with this “Intelligent Alert System”.

Developing effective human–wildlife conflict mitigation strategies requires an understanding of the conflict patterns, species involved and attitudes of local people living along protected area boundaries.

an authorized person using RFID’s

1.2. Existing System:

A design of a wireless sensor network to detect elephants as an early warning system which can act as a virtual barrier covering elephant corridors or villages would be presented in this paper. The elephant habitats are being continuously reduced due to increasing population and changes in the land-use patterns. Thus human elephant conflict has been increasing and remains unsolved regardless of various solutions developed. The proposed system is a virtual barrier rather than a physical one. A prototype system to detect elephants and send a warning message to the authorities and the villagers has been implemented. The wireless sensor network (WSN) has been setup as a virtual barrier covering elephant corridors or villages.

Disadvantages of Existing System:

In this section, we present some of the limitations that are present in the existing system.

- Very Slow response
- Since it consists of sensors, they are sensitive to extreme environmental changes.
- Expensive
- Low accuracy
- Poor results with success rate of classification around 50% only.
- Sensing range

1.3. Proposed System:

The proposed Intelligent alert system for forest tribal people model is based on neural networks (CNN) incorporated with an alerting system. This system will monitor the entire villages of surrounding forests at regular intervals through a camera. Once any dangerous animal is detected then it will send information to the people in the village and

it will produce an appropriate sound or alarm in the village to alert the people.

Advantages:

- Low cost
- No need for manual guarding
- Can ensure

  a. no crop raiding

  b. no loss of domestic life due to livestock predation

  c. decrease the risk of livestock diseases
- Safeguarding human life and wildlife as well.

2. SYSTEM ANALYSIS:

Wildlife conservation and the management of human–wildlife conflicts require cost-effective methods of monitoring wild animal behavior. Still and video camera surveillance can generate enormous quantities of data, which is laborious and expensive to screen for the species of interest. In the present study, we describe a state-of-the-art, deep learning approach for automatically identifying and isolating species-specific activity from still images and video data.

Efficient and reliable monitoring of wild animals in their natural habitat is essential. This project develops an algorithm to detect the animals in wild life. Since there are large number of different animals manually identifying them can be a difficult task. This algorithm classifies animals based on their images so we can monitor them more efficiently. Animal detection and classification can help to prevent animal-vehicle accidents, trace animals and prevent theft. This can be achieved by applying effective deep learning algorithms.

2.1. Software requirements:

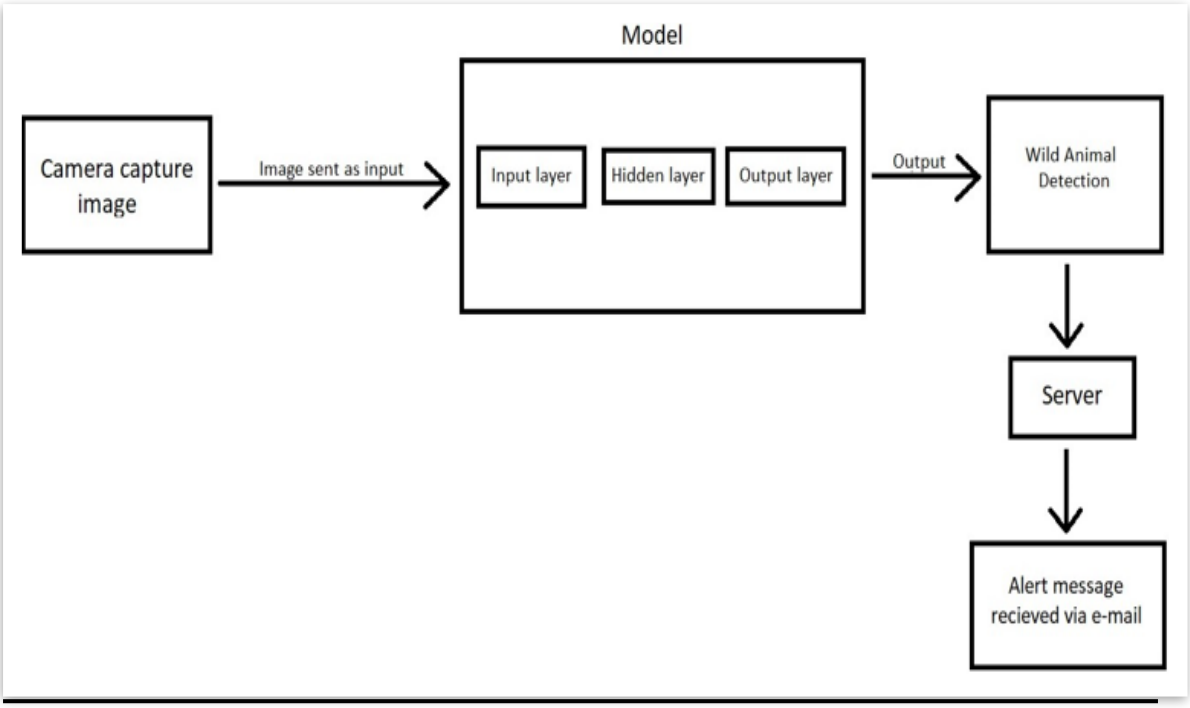
- Operating System : Windows 7 , Windows 8 or higher versions
- Language : Python 3 (Developer: Anaconda – Jupyter notebook)

2.2. Hardware requirements:

- RAM : 4GB RAM or higher
- Hard disk : 40GB or higher
- Cameras

3. SYSTEM DESIGN:

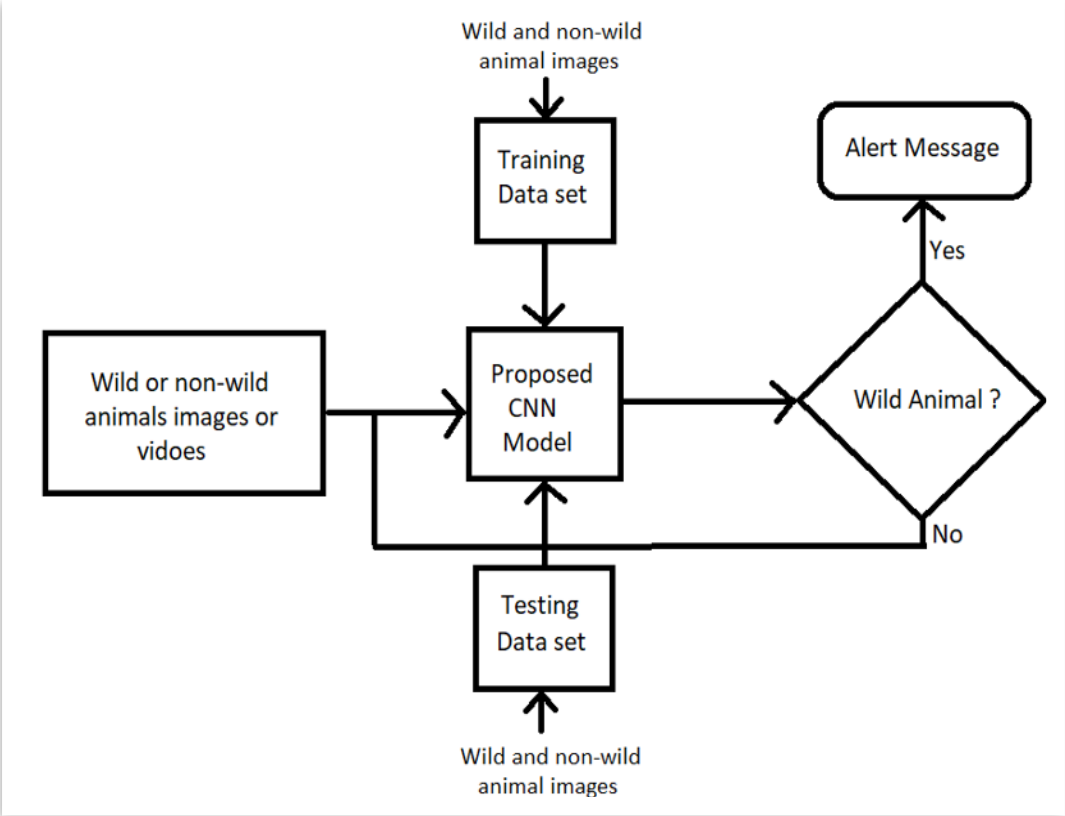
Systems design is the process of defining the architecture and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.



CNN:

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vison that includes image and video recognition, along with recommender systems and natural language processing (NLP).

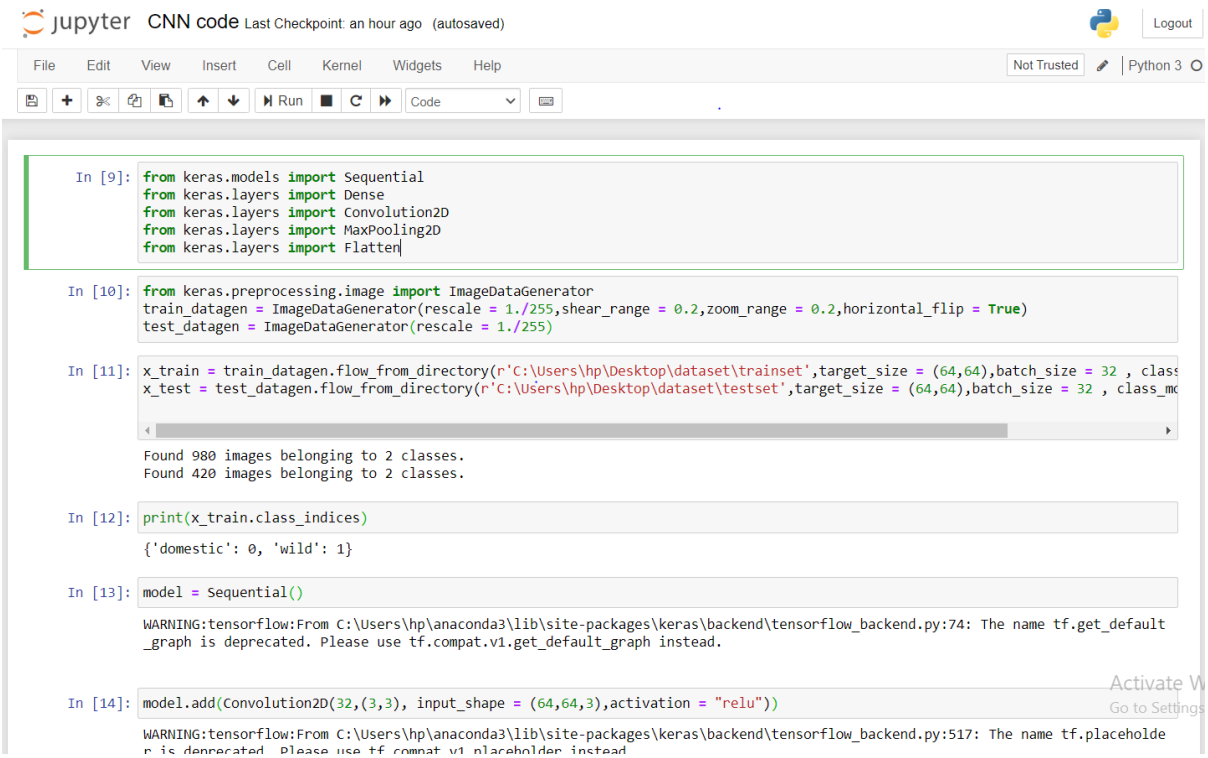


4. SYSTEM IMPLEMENTATION:

The implementation stage of any project is a true display of the defining moments that make a project a success or a failure. The implementation stage is defined as the system or system modifications being installed and made operational in a production environment. The

phase is initiated after the system has been tested and accepted by the user. This phase continues until the system is operating in production in accordance with the defined user requirements.

CNN CODE FOR TRAINING THE MODEL WITH WILD AND NON-WILD ANIMAL DATASET



jupyter CNN code Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```
WARNING:tensorflow:From C:\Users\hp\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use tf.random.uniform instead.
```

In [15]:
```
model.add(MaxPooling2D(pool_size = (2,2)))
```

```
WARNING:tensorflow:From C:\Users\hp\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3976: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [16]:
```
model.add(Flatten())
```

In [17]:
```
model.add(Dense(units = 128,init = "random_uniform",activation = "relu"))
```

```
C:\Users\hp\anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(units=128, activation="relu", kernel_initializer="random_uniform")`
  """Entry point for launching an IPython kernel.
```

In [18]:
```
model.add(Dense(units =2,init = "random_uniform",activation = "softmax"))
```

```
C:\Users\hp\anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `Dense` call to the Keras 2 API: `Dense(units=2, activation="softmax", kernel_initializer="random_uniform")`
  """Entry point for launching an IPython kernel.
```

In [19]:
```
model.compile(loss = "categorical_crossentropy",optimizer = "adam",metrics = ["accuracy"])
```

```
WARNING:tensorflow:From C:\Users\hp\anaconda3\lib\site-packages\keras\optimizers.py:790: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\hp\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3295: The name tf.log is deprecated. Please use tf.math.log instead.
```


jupyter CNN code Last Checkpoint: an hour ago (autosaved)

Logout

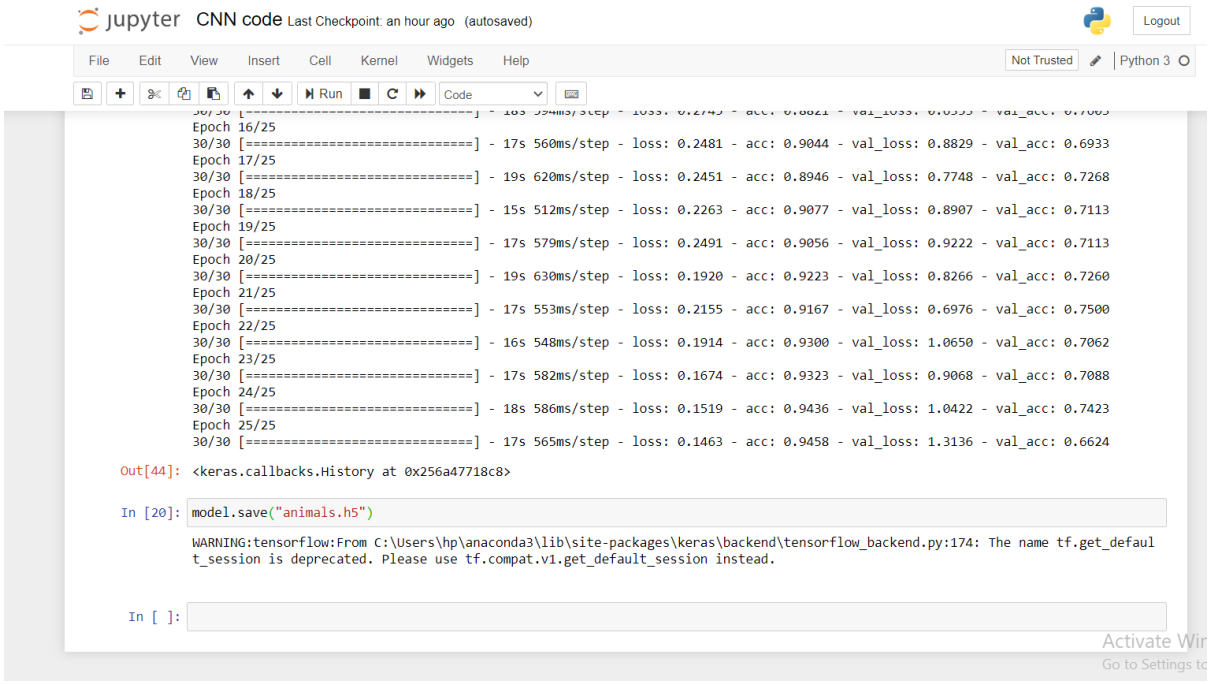
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

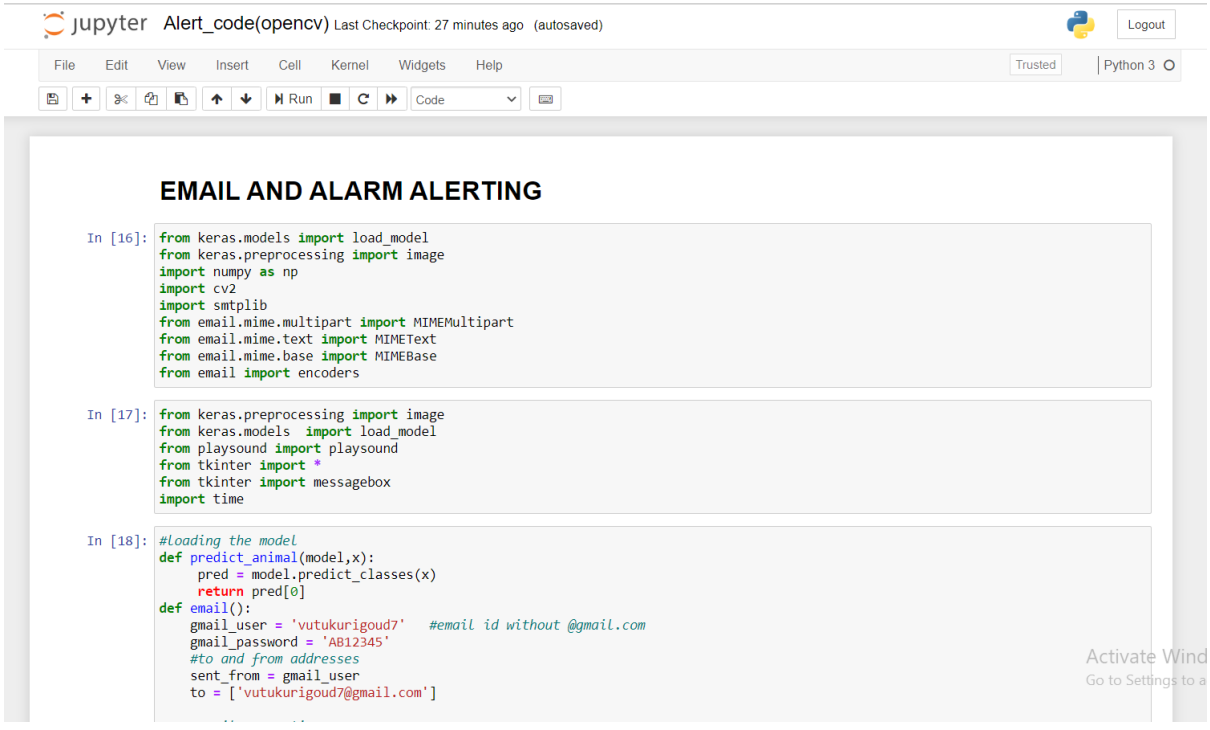
Run Code

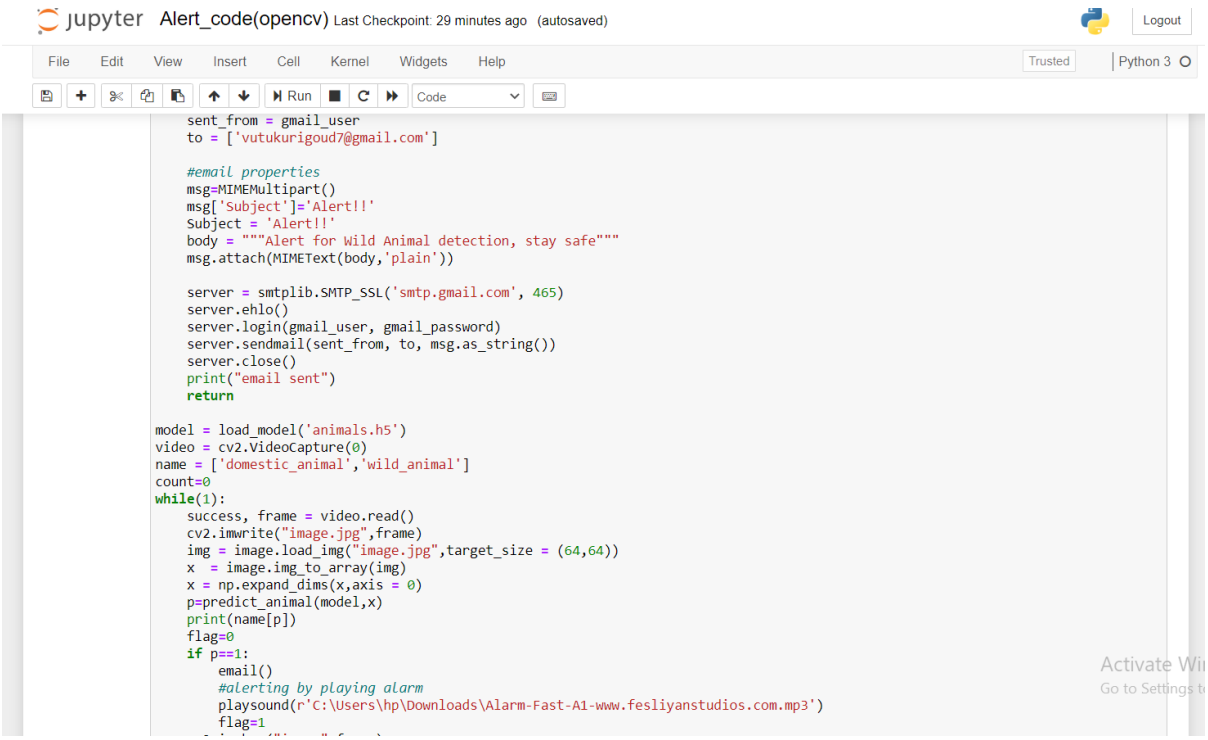
In [44]:
```
model.fit_generator(x_train , steps_per_epoch =  30 , epochs = 25, validation_data = x_test, validation_steps =13)
```

```
Epoch 1/25
30/30 [==============================] - 19s 636ms/step - loss: 0.5000 - acc: 0.7498 - val_loss: 0.6839 - val_acc: 0.6649
Epoch 2/25
30/30 [==============================] - 16s 535ms/step - loss: 0.4694 - acc: 0.7677 - val_loss: 0.5501 - val_acc: 0.6959
Epoch 3/25
30/30 [==============================] - 19s 622ms/step - loss: 0.4165 - acc: 0.8085 - val_loss: 0.6200 - val_acc: 0.7113
Epoch 4/25
30/30 [==============================] - 17s 553ms/step - loss: 0.3782 - acc: 0.8267 - val_loss: 0.5946 - val_acc: 0.7448
Epoch 5/25
30/30 [==============================] - 18s 588ms/step - loss: 0.4073 - acc: 0.8044 - val_loss: 0.7229 - val_acc: 0.6753
Epoch 6/25
30/30 [==============================] - 19s 618ms/step - loss: 0.4006 - acc: 0.8268 - val_loss: 0.6191 - val_acc: 0.7188
Epoch 7/25
30/30 [==============================] - 18s 593ms/step - loss: 0.3544 - acc: 0.8400 - val_loss: 0.6265 - val_acc: 0.7191
Epoch 8/25
30/30 [==============================] - 17s 564ms/step - loss: 0.3502 - acc: 0.8502 - val_loss: 0.6882 - val_acc: 0.7139
Epoch 9/25
30/30 [==============================] - 16s 547ms/step - loss: 0.3255 - acc: 0.8452 - val_loss: 0.7959 - val_acc: 0.6753
Epoch 10/25
30/30 [==============================] - 17s 580ms/step - loss: 0.3094 - acc: 0.8742 - val_loss: 0.6234 - val_acc: 0.7062
Epoch 11/25
30/30 [==============================] - 18s 593ms/step - loss: 0.3191 - acc: 0.8658 - val_loss: 0.6539 - val_acc: 0.7371
Epoch 12/25
30/30 [==============================] - 16s 527ms/step - loss: 0.3047 - acc: 0.8632 - val_loss: 0.7132 - val_acc: 0.6933
Epoch 13/25
30/30 [==============================] - 18s 584ms/step - loss: 0.2979 - acc: 0.8719 - val_loss: 0.6577 - val_acc: 0.7294
Epoch 14/25
30/30 [==============================] - 18s 595ms/step - loss: 0.2766 - acc: 0.8800 - val_loss: 0.8142 - val_acc: 0.6778
Epoch 15/25
30/30 [==============================] - 18s 594ms/step - loss: 0.2745 - acc: 0.8821 - val_loss: 0.6353 - val_acc: 0.7603
Epoch 16/25
30/30 [==============================] - 17s 560ms/step - loss: 0.2481 - acc: 0.9044 - val_loss: 0.8829 - val_acc: 0.6933
```



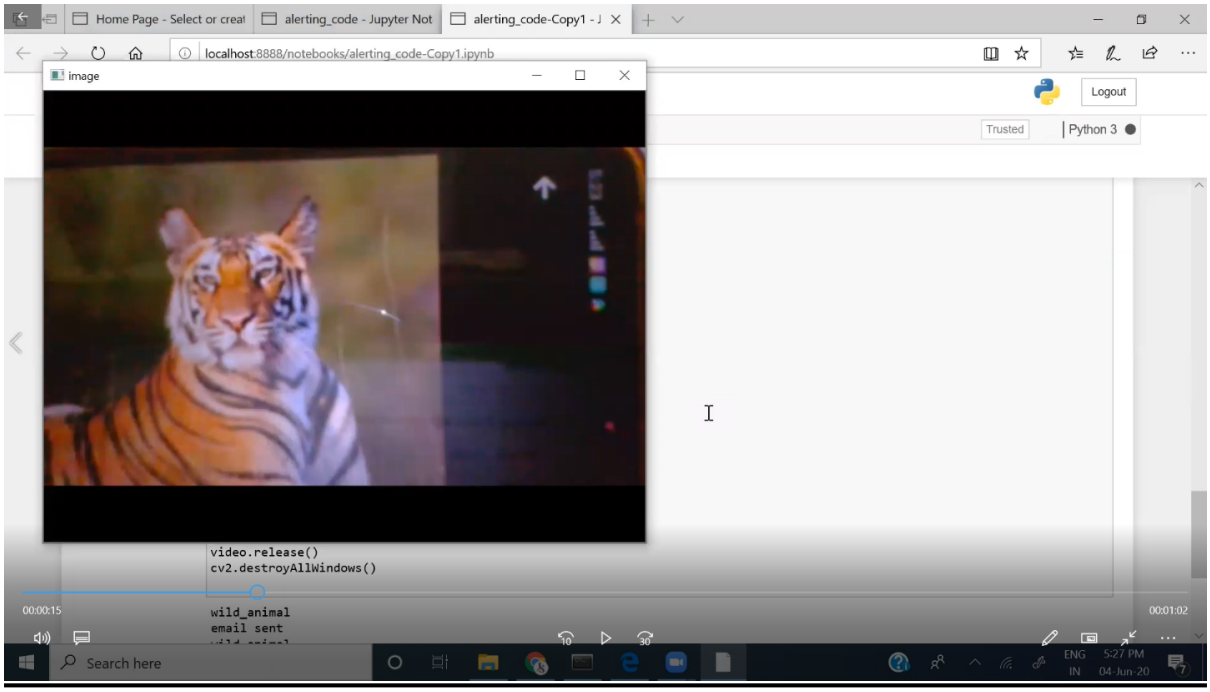
Alerting Code:

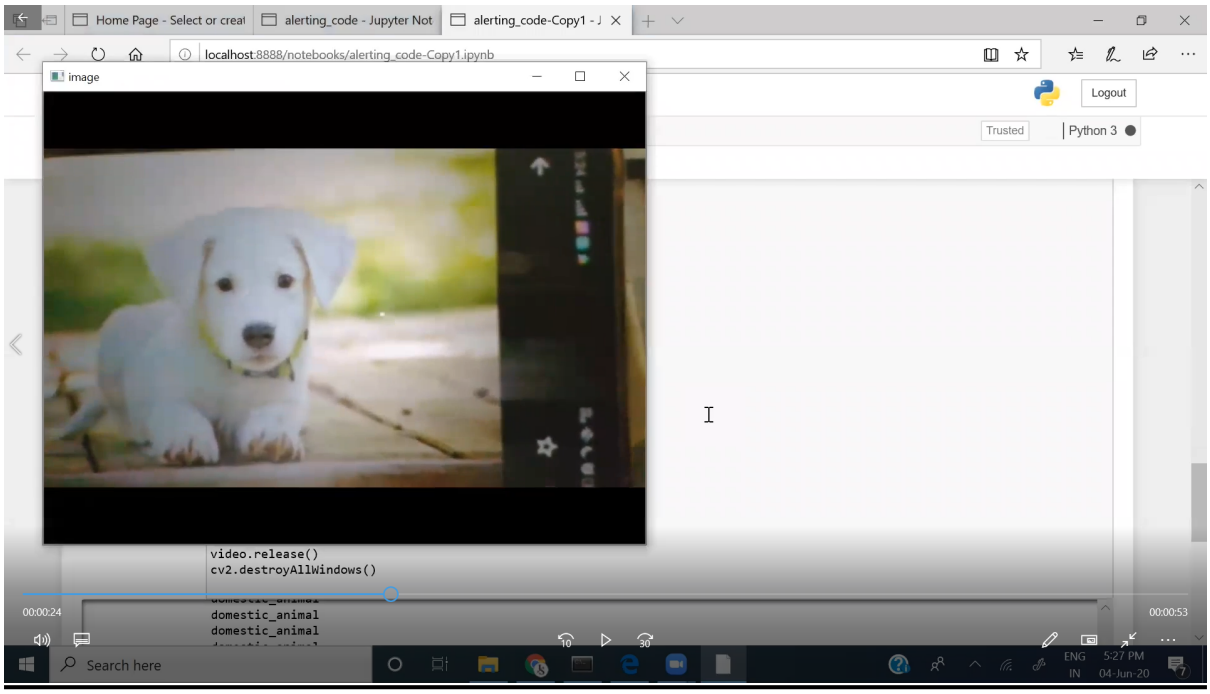





```
    p=predict_animal(model,x)
    print(name[p])
    flag=0
    if p==1:
        email()
        #alerting by playing alarm
        playsound(r'C:\Users\hp\Downloads\Alarm-Fast-A1-www.fesliyanstudios.com.mp3')
        flag=1
    cv2.imshow("image",frame)
    if cv2.waitKey(1) & 0xFF == ord('a'):
        break
    if flag==1:
        continue
video.release()
cv2.destroyAllWindows()
```

5. OUTPUT SCREENS:


Home Page - Select or creat
alerting_code - Jupyter Not
alerting_code-Copy1 -
localhost:8888/notebooks/alerting_code-Copy1.ipynb
Logout
Trusted
Python 3
image
video.release()
cv2.destroyAllWindows()
00:00:15
wild_animal
email sent
00:01:02
Search here
ENG
IN
5:27 PM
04-Jun-20


Home Page - Select or creat
alerting_code - Jupyter Not
alerting_code-Copy1 -
localhost:8888/notebooks/alerting_code-Copy1.ipynb
Logout
Trusted
Python 3
image
video.release()
cv2.destroyAllWindows()
00:00:24
domestic_animal
domestic_animal
00:00:53
Search here
ENG
IN
5:27 PM
04-Jun-20

6. CONCLUSION:

The animals, many of which are already threatened or endangered, are often killed in retaliation or to prevent future conflicts. So this zone is to be monitored continuously to prevent entry of wild animals. With regard to this problem, we have made an effort to develop the system which will monitor the field using sensor and camera and captured image of the intruder will be classified using image processing so that suitable action can be taken.

7. FUTURE SCOPE:

To be sure advances in hardware (high resolution cameras), storage, parallel processing architectures will enable even greater leaps in the functionality of this Intelligent Alert System thereby reducing human-wildlife conflict .However the field of Artificial Intelligence will remains in its infancy.

BIBLIOGRAPHY:

- Data collection :

https://www.kaggle.com/datasets

- Data preprocessing :

https://thesmartbridge.com/documents/spsaimldocs/CNNprep.pdf

- Model Building :

https://thesmartbridge.com/documents/spsaimldocs/CNNflow.pdf

- OpenCV for video processing :

https://opencv-python-tutrials.readthedocs.io/en/latest/py_tutorials/py_gui/py_ video_display/py_video_display.html

- Alerting through email:

https://www.youtube.com/watch?v=B1IsCbXp0uE