RESOLVING MAN-ANIMAL CONFLICT WITH ELEPHANT DETECTION USING CONVOLUTIONAL NEURAL NETWORK

A PROJECT REPORT

Submitted by

SHAHANA A

15CSR191

VIDHYA C

15CSR227

YUVARAJ S M

15CSR241

in partial fulfillment of the requirements

for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMMUNICATION AND COMPUTER SCIENCES

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 060

APRIL 2019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KONGU ENGINEERING COLLEGE
(Autonomous)

PERUNDURAI ERODE – 638060

APRIL 2019

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **RESOLVING MAN-ANIMAL CONFLICT WITH ELEPHANT DETECTION USING CONVOLUTIONAL NEURAL NETWORK** is the bonafide record of project work done by **SHAHANA A(Register no.: 15CSR191), VIDHYA C (Register no.: 15CSR227) and YUVARAJ S M(Register no.: 15CSR241)** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Science and Engineering** of Anna University, Chennai during the year 2018 - 2019.

SUPERVISOR **HEAD OF THE DEPARTMENT**

(Signature with seal)

Date:

Submitted for the end semester viva voce examination held on________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KONGU ENGINEERING COLLEGE
(Autonomous)

PERUNDURAI ERODE – 638060

APRIL 2019

DECLARATION

We affirm that the project report titled **RESOLVING MAN-ANIMAL CONFLICT WITH ELEPHANT DETECTION USING CONVOLUTIONAL NEURAL NETWORK**being submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering is the original work carried out by us. It has not formed part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Date:

SHAHANA A
(Reg.No.:15CSR191)

VIDHYA C
(Reg.No.:15CSR227)

YUVARAJ S M
(Reg.No.:15CSR241)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date **Name & Signature of the Supervisor with seal**

ABSTRACT

Nowadays farmers who cultivate crops adjoining forest areas face a serious threat from wild animals like elephants and wild boars. These animals trespass into fields and sometimes into villages in search of food and water. This is mainly due to deforestation and intrusion of humans into the perimeter of the forests and corridors used by wild animals for foraging and migration.

Such human-animal conflicts leads to colossal damage of agricultural crops and also at times causes loss of precious human life who are working in the field. The solution for this problem is to devise an early warning system that could detect the arrival of elephants sufficiently before the actually raid a field. The objective of this project is to detect elephants by using Convolution Neural Networks and running Deep Learning algorithms on images that are captured by cameras installed at likely routes of elephants' movement.

The system once it detects an elephant will notify through SMS a set of mobile numbers that will serve the purpose of an early warning system. This system is trained with 637 images of positive and 250 negative samples of Color Images. The system gives 98% accuracy on the test set.

ACKNOWLEDGEMENT

We express our sincere thanks and gratitude to our beloved correspondent **Thiru.K.Venkatachalam B.Sc.,** for giving us the opportunity to pursue this course. We areextremely thankful with no words of formal nature to the dynamic principal **Prof.S.Kuppuswami B.E., M.Sc(Engg)., Dr.Ing(France).,** for providing the necessaryfacilities to complete our work.

We would like to express our sincere gratitude to our respected Head of the Department **Dr.R.R.Rajalaxmi M.E., Ph.D.,** for providing necessary facilities.

Our debt is owed to **Mr.K.Devendran M.E.,** the project coordinator for his encouragement and valuable advice that made us to carry out the project work successfully.

We extend our gratitude to our supervisor, **Dr.R.Thangarajan M.E.,Ph.D.,** for his valuable ideas and suggestions, which have been very helpful in the project. We are grateful to all the faculty members of the Computer Science and Engineering Department, for their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolution Neural Networks
ANN	Artificial Neural Networks
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
LSTMs	Long Short-Term Memory’s
RL	Reinforcement Learning

CHAPTER 1

INTRODUCTION

1.1 DEEP LEARNING

Deep learning is part of machine learning methods based on learning datarepresentations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social networkfiltering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human experts.

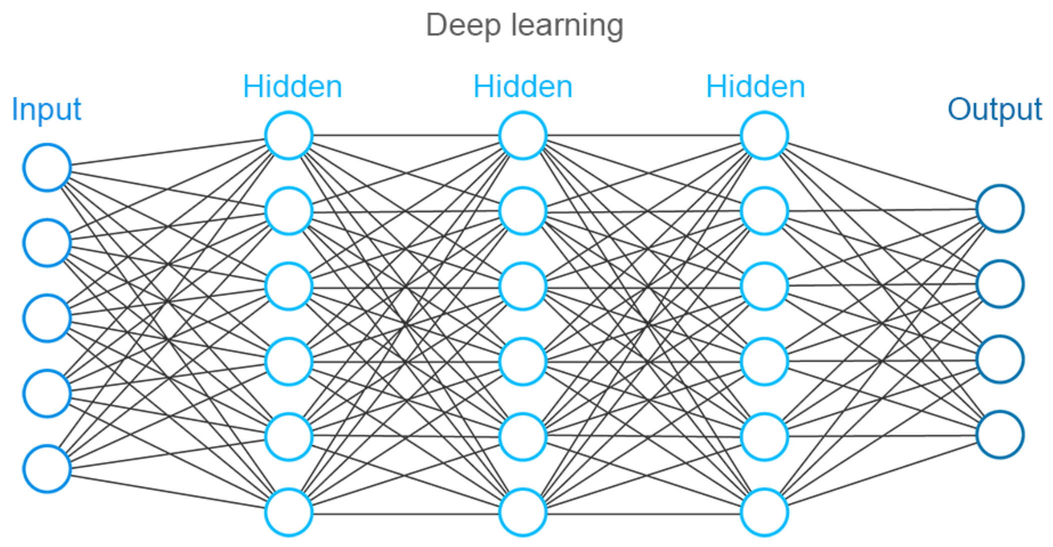


Figure 1.1

The neural networks a methodology to connect human brains and computer to increase the interaction of machines and human. It is used in various applications such as visual processing, signal processing and many more.Initially, the network has to be trained by feeding the information and sequences to it for the purpose to make the machine work precisely.

Artificial Neural Network (ANN) has a capability to feed-forward and hence it is called standard feed-forward network. Convolution Neural Network(CNN) is a new technique to solve large dataset recognition effectively as it has fewer parameters and connections than the Artificial neural networks. It has capacity to process more than million images in a quick time accurately.

Tensor flow is an open source library designed by Google to train,design and bulit large datasets. Sign language is nothing but a one way of communication to help deaf-dumb community mostly by hand gestures for better understanding.

Overfitting is a disturbance to the models in machine learning that will negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model.

Predictions from different models and combining the results of it will reduce the test errors, but it is costliest and takes much longer to train.So, to overcome this the technique called dropout consists of setting to zero is used. The output of each hidden neuron with probability 0.5. The neurons which are“dropped out” in this way do not contribute to the forward pass and do not participate in backpropagation. Dropout reduces complex co-adaptations of neurons,since a neuron cannot rely on the presence of particular other neurons.

1.2 TECHINIQUES IN DEEP LEARNING

- ✓ **Convolutional Neural Network**

  Convolutional Neural Networks are very similar to ordinary Neural Networks .They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.In deep learning, a **Convolutional Neural Network**(**CNN**, or**ConvNet**) is a class ofdeep, feed-forward artificial neural networks, mostcommonly applied to analyzing visualimagery.CNNs use a variation of multilayer perceptrons designed to require minimalpreprocessing. Theyare also known as**shift invariant or space invariant artificialneural networks** (**SIANN**), based on their shared-weights architecture andtranslationinvariance characteristics.

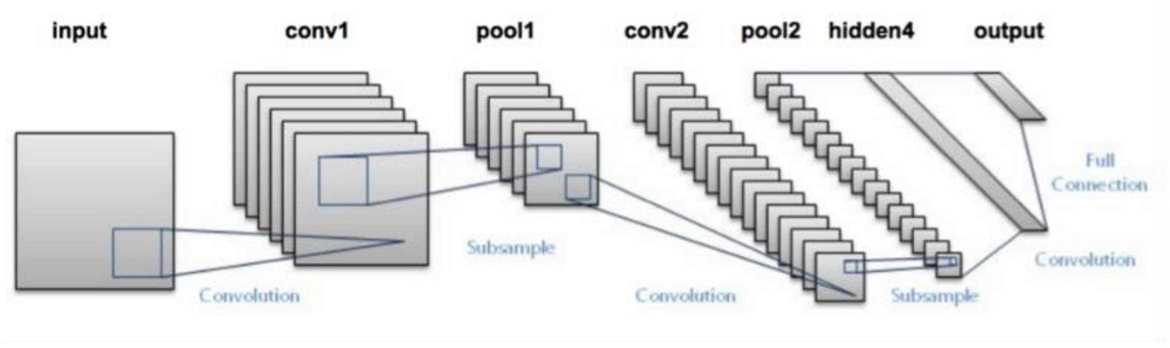


Figure 1.2

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual corticalneurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

- ✓ **Fully Connected Neural Network**

After several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

A linear operation in which every input is connected to every output by a weight (so there are n_inputs * n_outputs weights - which can be a lot!). Generally followed by a non-linear activation function.

The basic idea behind a fully convolutional network is that it is “fully convolutional”, that is, all of its layers are convolutional layers.FCNs don’t have any of the fully-connected layers at the end, which are typically use for classification. Instead, FCNs use convolutional layers to classify each pixel in the image. So the final output layer will be the same height and width as the input image, but the number of channels will be equal to the number of classes. If we’re classifying each pixel as one of fifteen different classes, then the final output layer will be height x width x 15 classes. Using a softmax probability function, we can find the most likely class for each pixel.

In a fully connected layer each neuron is connected to every neuron in the previous layer, and each connection has its own weight. This is a totally general purpose connection pattern andmakes no assumptions about the features in the data. It's also very expensive in terms of memory (weights) and computation.Each neuron in a neural network contains an activation function that changes the output of a neuron given its input. These activation functions are:

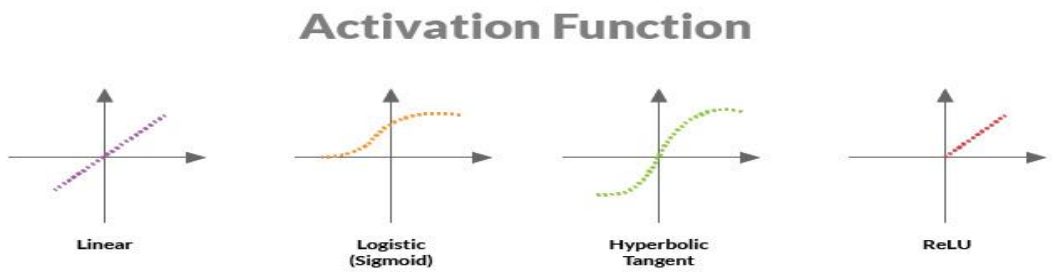


Figure 1.3

- **Linear function -** It is a straight line that essentially multiplies the input by a constant value.
- **Non-Linear Function**

A)Sigmoid function- It is an S-shaped curve ranging from 0 to 1.

B)Hyperbolic tangent (tanH) function- It is an S-shaped curve ranging from -1 to +1

C)Rectified linear unit(ReLU) function- It is a piecewise function that outputs a 0 if the input is less than a certain value, or linear multiple if the input is greater than a certain value.

Each type of activation function has pros and cons, so we use them in various layers in a deep neural network based on the problem each is designed to solve.

- ✓ **Recurrent Neural Network**

A **recurrent neural network** (**RNN**) is a class of artificial neural network where connections between nodes form a directed graph a long a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected **handwriting recognition** or **speech recognition.**

The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infiniteimpulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictlyfeedforward neural network, while an infinite impulse recurrent network is a directed cyclicgraph that can not be unrolled.Both finite impulse and infinite impulse recurrent networks can have additional stored state, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of **longshort-term memory(LSTMs)** and gated recurrent units.

- ✓ **Deep Reinforcement Learning**

Reinforcement learning (**RL**) is an area of machine learning concerned with how softwareagents ought to take **actions**in an **environment**so as to maximize some notion of cumulative reward. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In the operations research and control literature, reinforcement learning is called approximate dynamic programming, or neuro-dynamic programming.The problems of interest in reinforcement learning have also been studied in the theory ofoptimal control, which is concerned mostly with the existence and characterization of optimal solutions, and algorithms for their exact computation, and less with learning or approximation, particularly in the absence of a mathematical model of the environment.

CHAPTER 2

LITERATURE REVIEW

2.1 EXISTING SYSTEM

Naveen et alin the paper titled as “WARNING SYSTEM TO PREVENT ELEPHANT INTRUSION” havediscussed a system uses PIR sensor and vibration sensor.When heavy vibration is caused by elephant movement, the sensor sends the electrical pulses.If it matches the threshold value powerful flashing lights are activated.If the elephant still continues in path PIR sensor is activated producing Bee Sound and with the help of GSM messages are sent to the village representatives.By this method they alert the people about the elephant intrusion.

Wireless sensor technology experiences high cost which includes the cost of hardware components. In this case,the alarm was provided only when the vibration was caused by the Elephant. The system fails to detect if the Elephant doesn’t cause vibration at all. This can be avoided by detecting the Elephants in deep learning approach. Here the Elephants are detected by capturing the images and processing it,thus predicting the Elephants raid into the field.

CHAPTER 3

SYSTEM REQUIREMENTS

The system requirements deal with the hardware and software needed for the development and implementation of the system.

3.1 SOFTWARESPECIFICATION

Coding Language	: Python
IDE	: Anaconda, Jupyter notebook
Framework	: Tensor Flow,Keras

3.2SOFTWARE DESCRIPTIONS

3.2.1 Anaconda

Anaconda is a free - open source distribution of Python and R programming languages. It is used for large-scale data processing, predictive analytics and scientific computing which aims to simplify package management and deployment. Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

3.2.2 Jupyter Notebook

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. It can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the internet.In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard”

(Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

3.2.3 Tensor flow

Tensorflow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library used for machine learning applications such as neural networks. Among the applications for which TensorFlow is the foundation, are automated image captioning software, such as DeepDream. RankBrain now handles a substantial number of search queries, replacing and supplementing traditional static algorithm-based search results.

3.2.4 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. The principlesinclude,

- User friendliness. Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- Modularity. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.
- Easy extensibility. New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.
- Work with Python. No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

3.3 LANGUAGES USED

3.3.1 Python

Python has become one of the most popular programming languages among developers and programmers. Python is known for its clean syntax and code readability. Python is a general-purpose high-level programming language. Python is both object oriented and imperative and it can be even used in a functional style as well. In Image processing, it has modules like Numpy, Scipy, Matplotlib and other special modules provide the optimal functionality to be able to cope with the flood of pictures.

CHAPTER 4

PROPOSED SYSTEM

4.1 INTRODUCTION

The Elephant Intrusion leads to the loss in Sugarcane production and also human life.The solution for this to deter the raid of the elephants into the field.This can be achieved by classifying the image captured using Camera in the background.The detection of Elephant provides an early warning to the Farmers.The process of Classification and detection is proposed using Convolutional Neural Network in Deep Learning Approach.

4.2 MODULE DESCRIPTION

4.2.1 DATASET COLLECTION

The dataset for the training and testing have been generated manually. The images of an elephant are collected from various websites. The positive images trained were about 637 and negative images were about 250 in count. This includes the images of an elephant in various postures and gestures. After collecting the images, it undergoes the process of pre –processing. Here downsizing of elephant images were done to $256\ x\ 256$.

4.2.2 LAYERS IN NETWORK

Convolutional Neural Networks (CNN) is a type of deep neural network architecture designed for specific tasks like image classification. CNNs were inspired by the organization of neurons in the visual cortex of the animal brain. A CNN is composed of an input layer. However, for basic image processing, this input is typically a two-dimensional array of neurons which correspond to the pixels of an image. It also contains an output layer which is typically a one- dimensional set of output neurons. CNN uses a combination of sparsely connected convolution layers, which perform image processing on their inputs. In addition, they contain down sampling layers called pooling layers to further reduce the number of neurons necessary in subsequent layers of the network. And finally, CNNs typically contain one or more fully connected layers to connect our pooling layer to our output layer.

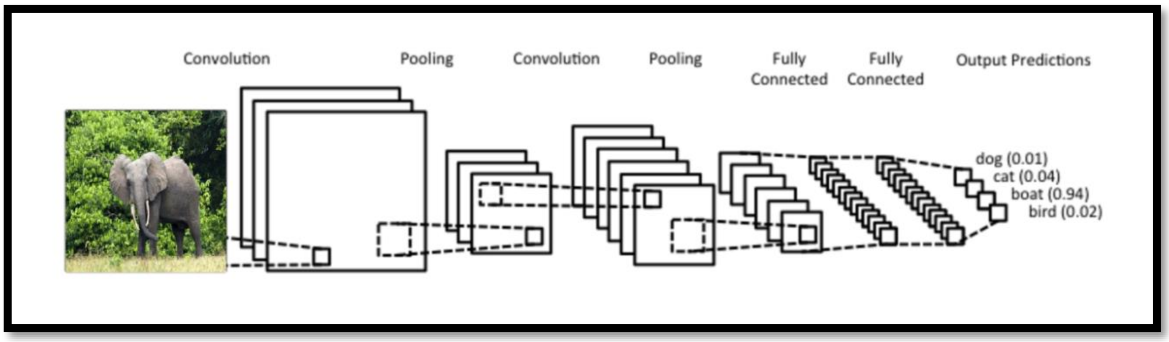


Figure 4.1 Layers of convolution

CONVOLUTIONAL NEURAL NETWORK LAYERS

- Input layer
- Pooling layer
- Fully connected layer
- Dropout layer
- Output layer

➢ INPUT LAYER

The input image is placed into this layer. It can be a single-layer 2D image (grayscale), 2D 3-channel image (RGB colour) or 3D. The main difference between how the inputs are arranged comes in the formation of the expected kernel shapes

➢ POOLING LAYER

The pooling layer is key to making sure that the subsequent layers of the CNN are able to pick up larger-scale detail than just edges and curves. It does this by merging pixelregions in the convolved image together (shrinking the image) before attempting to learn kernels on it. effectively, this stage takes another kernel, say [2 x 2] and passes it over the entire image, just like in convolution. It is common to have the stride and kernel size equal i.e. a [2 x 2] kernel has a stride of 2. Thus the pooling layer returns an array with the same depth as the convolution layer.

- **FULLY CONNECTED LAYER**

The fully connected (FC) layer in the CNN represents the feature vector for the input. When the network gets trained, this feature vector is then further use for classification.It is also being used as a encoded vector. During training, this feature vector is being used to determine the loss, and help the network to get train.The convolution layers before the FC layer(s) hold information regarding local features in the input image such as edges, blobs, shapes, etc. Each conv layer hold several filters that represent one of the local features. The FC layer holds composite and aggregated information from all the conv layers .

- **DROPOUT LAYER**

Fully-connected layer is connected to all weights therefore this can be a very large number. As such, an FC layer is prone to **overfitting** meaning that the network won't generalise well to new data. There are a number of techniques that can be used to reduce overfitting though the most commonly seen in CNNs is the dropout layer. This layer causes the network to 'drop' some nodes on each iteration with a particular probability. The **keepprobability** is between 0 and 1.This is the probability that a particular node is dropped duringtraining. When back propagation occurs, the weights connected to these nodes are not updated

- **OUTPUT LAYER**

The output layer consists of a number of nodes which have a high value if they are 'true' or activated. Some output layers are probabilities and as such will sum to 1, while others will just achieve a value which could be a pixel intensity in the range 0-255. The output can also consist of a single node.

4.2.3 DEFINING BATCH SIZE AND EPOCH

A convolutional neural network (CNN) doesn't process its inputs one-at-a-time: to increase throughput, it will process the data in batches. Batch size defines number of samples that going to be propagated through the network.It refers to the number of training examples utilised in one iteration.An epoch is a single pass through the full training set. You don't just run through the training set once, it can take thousands of epochs for your backpropagation algorithm to converge on a combination of weights with an acceptable level of accuracy.

4.2.4 DATA FORMAT

Keras has three backend implementations available the TensorFlow backend, the Theano backend, and the CNTK backend.For tensorflow backend the data format is Channels_ last and Channels_first in the case of Theano backend.The backend of the implemenation is checked, and the splitting of datasets are done by specifying the channel size at the first in the case of Channels_first whereas at last for Channels_last.

4.2.5 BUILDING A MODEL

In this Convolution is done on the image with the 3 x 3 matrix by activating the ReLu function.In the past, nonlinear functions like tanh and sigmoid were used, but researchers found out that ReLU layers work far better because the network is able to train a lot faster without making a significant difference to the accuracy.The ReLU layer applies the function $f(x) = max(0, x)$ to all of the values in the input volume. In basic terms, this layer just changes all the negative activations to 0.The output from the ReLu unit will be taken into max-pool layer for more classification.The Pool - size is specified here,in this case (2,2).It is repeated again.Then comes the Dropout layer,an approach in Regularization.Dropout layers have a very specific function in neural networks . Drop out is added to prevent Over-fitting.Flattening of values to a long vector array is done with the help of flatten().The output of this function is provided to the fully- connected layer.The ReLu activation function is applied on the previous output.At the final layer,softmax function converts a vector of raw scores into class probabilities at the output layer of a Neural Network used for classification.

4.2.6 TRAINING AND SAVING THE MODEL

Compile method configures the model for training.The optimizer,loss and metrics are passed as the function arguments.Here the optimizer is AdaDelta which doesn't force to initialize the learning rate.The model is then trained for the specified number of Epochs.The trained model is then saved in the json file and the weights on the Network is saved in the model.Thus the saved model can be invoked for the future use in the prediction on the Video Input.

4.2.7VIDEO CLASSIFICATION

4.2.7.1 FRAME EXTRACTION

In the Video Classification phase, Video file is given as an Input to the Model. The saved model is now invoked and proceeds further for the classification of Images from the video. Frame rate is defined such that the frames are extracted from the video in a second. Here the frame rate is 5 per second. About 5 frames are extracted from the Video Input and each of the images undergo the process of preprocessing and then the Classification.

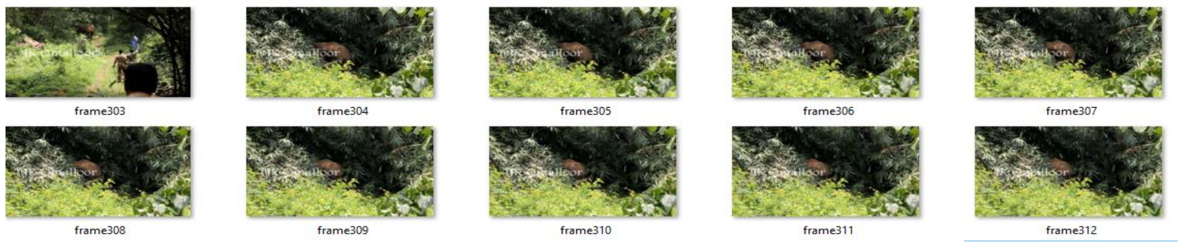
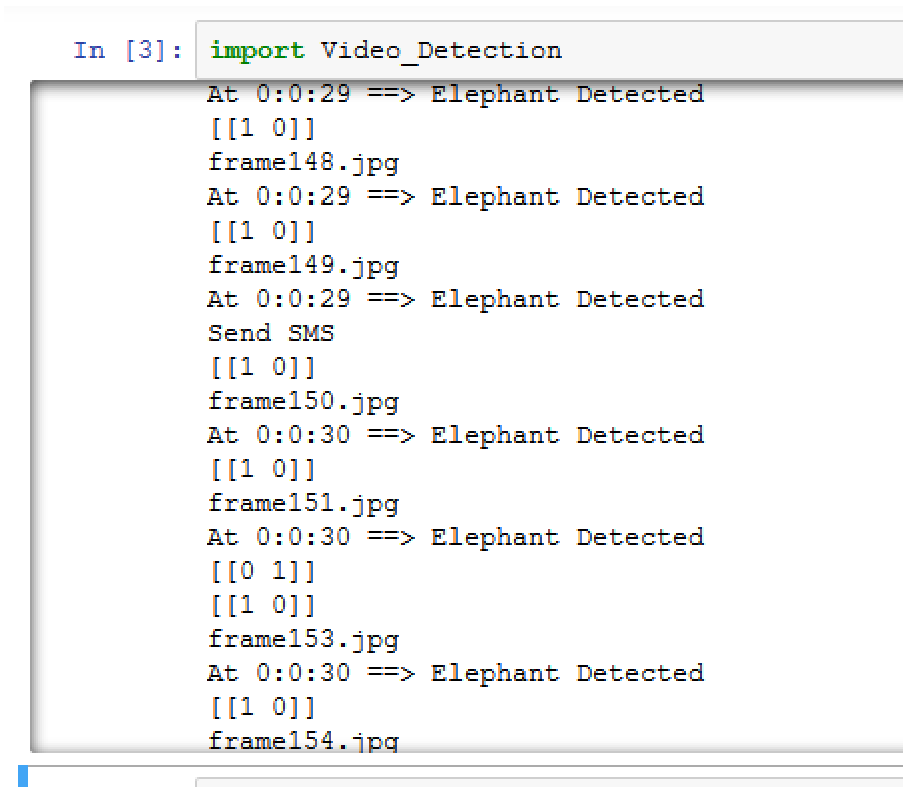


Figure 4.2

4.2.7.2 IMAGE CLASSIFICATION

The extracted frames undergo the preprocessing. The image is converted into 256 x 256 resolution since the trained images mismatch occurs. Then conversion of image into a pixel array for the prediction of image takes place. Then the model invoked is used for the prediction of the current frame. The predicted value is compared with the threshold and classified as an Elephant.

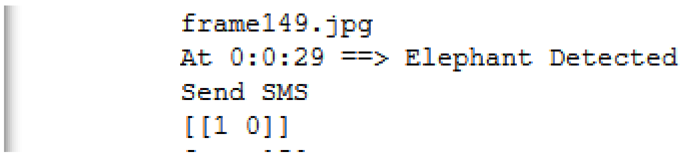


```
In [3]: import Video_Detection
        At 0:0:29 ==> Elephant Detected
        [[1 0]]
        frame148.jpg
        At 0:0:29 ==> Elephant Detected
        [[1 0]]
        frame149.jpg
        At 0:0:29 ==> Elephant Detected
        Send SMS
        [[1 0]]
        frame150.jpg
        At 0:0:30 ==> Elephant Detected
        [[1 0]]
        frame151.jpg
        At 0:0:30 ==> Elephant Detected
        [[0 1]]
        [[1 0]]
        frame153.jpg
        At 0:0:30 ==> Elephant Detected
        [[1 0]]
        frame154.jpg
```

Figure 4.3

4.2.7.3 ALERT ON DETECTION

The frames which are predicted as Elephant are counted. If the occurrence count is greater than 5 in 2 seconds, an alert SMS will be sent to those who are living along the region of the area.



```
frame149.jpg
At 0:0:29 ==> Elephant Detected
Send SMS
[[1 0]]
```

4.3 ARCHITECTURAL DIAGRAM

The fig 4.1 represents the general architecture of Convolution Neural Network. In this project, CNN needs to construct two conventional layers, two max-pooling layers and two fully connected layers in the convolution network.Finally,soft-max layers are used to aggregate and get the results. The first convolution layer filters the image with 3 x 3 matrix and sent to ReLu layer.It is then sent to max-pool layer for analyzing the image and then to fully-connected. For more accurate classification, the input of first convolution layeris taken in second convolution layer with the filter of 3 x 3 matrix and to ReLu layer to make the arguments positive and then to second max-pool layer for further analyzing and then sends to fully-connected layer.

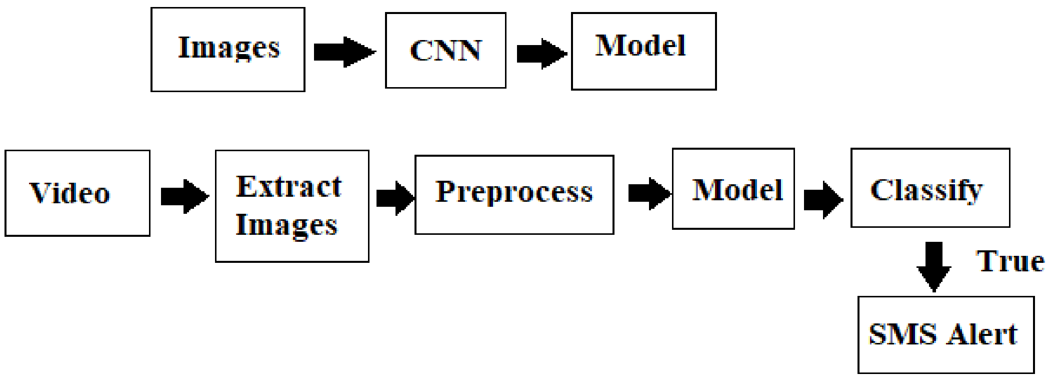
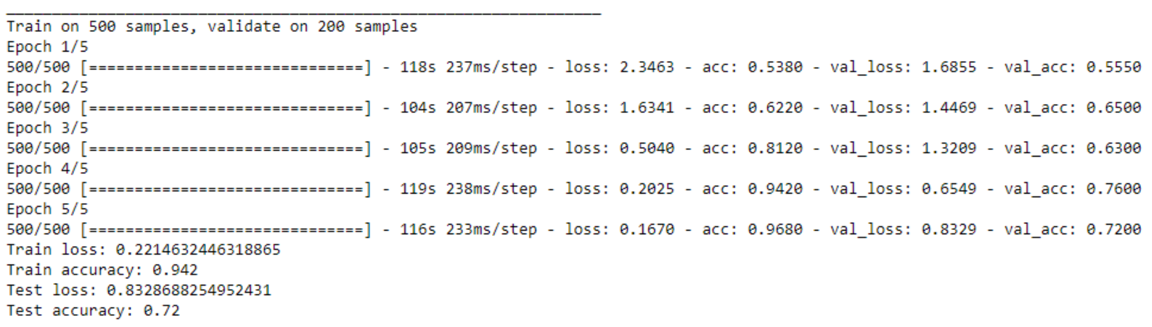


Figure 4.4

4.4 RESULTS AND DISCUSSION

➢ DATASET DESCRIPTION

The dataset for the training and testing have been generated manually. The images of an elephant are collected from various websites. This includes the images of an elephant in various postures and gestures. After collecting the it undergoes the process of pre – processing. Here downsizing of elephant images were done to 256 x 256.The training images consists of about 500 samples(200 negative samples and 300 positive samples) . The testing images consists of about 200 images(100 positive samples and 100 negative samples).



```
_________________________________________________________________
Train on 500 samples, validate on 200 samples
Epoch 1/5
500/500 [==============================] - 118s 237ms/step - loss: 2.3463 - acc: 0.5380 - val_loss: 1.6855 - val_acc: 0.5550
Epoch 2/5
500/500 [==============================] - 104s 207ms/step - loss: 1.6341 - acc: 0.6220 - val_loss: 1.4469 - val_acc: 0.6500
Epoch 3/5
500/500 [==============================] - 105s 209ms/step - loss: 0.5040 - acc: 0.8120 - val_loss: 1.3209 - val_acc: 0.6300
Epoch 4/5
500/500 [==============================] - 119s 238ms/step - loss: 0.2025 - acc: 0.9420 - val_loss: 0.6549 - val_acc: 0.7600
Epoch 5/5
500/500 [==============================] - 116s 233ms/step - loss: 0.1670 - acc: 0.9680 - val_loss: 0.8329 - val_acc: 0.7200
Train loss: 0.2214632446318865
Train accuracy: 0.942
Test loss: 0.8328688254952431
Test accuracy: 0.72
```

Figure 4.5

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

Thus, with much efficiency and effectiveness the Elephant images has been trained and put forwarded in convolution neural networks. It gets out the precise results whether the image represents Elephant or not. From this, specificity and accuracy of the image classification have been done precisely.

5.2 FUTURE WORK

The results show that a large, deep convolutional neural network is capable of achieving record-breakingresults on a highly challenging dataset using purely supervised learning. Ultimately, it can be used in very large and deep convolutional nets on video sequences.

APPENDIX

CODING

LoadDataset :

```
from numpy import load

import numpy as np

def load_data():

    data = np.load('Array.npz')

    lst = data.files



    #print("hai")

    print(np.size(data['x_test']))

    for item in lst:

        x_train,y_train=data['x_train'],data['y_train']

        x_test,y_test=data['x_test'],data['y_test']

        #y_train=data['y_train']

        #y_test=data['y_test']

        #print(x_train)

        #print(np.size(x_train))

        #print(data[item])

        #print("hai")



        print(len(x_train))
    return (x_train,y_train),(x_test,y_test)
```

Preprocess

```
#IMPORTING PACKAGES

import os
from matplotlib import pyplot as plt
import numpy as np
from PIL import Image
import fnmatch
import glob
import cv2
file_name2="Array"

#PATH SPECIFICATION
path3="D:\\15CSR191_227_241\\Dataset\\Train images"
path1="D:\\15CSR191_227_241\\Dataset\\Test images"
count=0
train=[]
train1=[]
train2=[]
test=[]
```

```
x_train=[]
x_train1=[]
x_train2=[]
x_test=[]
i=0
i1=0
i2=0
j=1
#ARRAY CONVERSION FOR TRAIN IMAGES
for file in os.listdir(path3):
    if file.endswith(".jpg"):
        list=os.path.join(path3,file)
        img=Image.open(list)
        x=np.array(img)
        train2.append((x))
        i2=i2+1
print(i2)
for file in os.listdir(path1):
    if file.endswith(".jpg"):
        list=os.path.join(path1,file)
        img=Image.open(list)
        x=np.array(img)
        test.append((x))
        j=j+1
```

```
#print(j)

#CLASS LABEL FOR TRAIN IMAGES


cls_train=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,


1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,


0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,


1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,


1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,


0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,


1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,


1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,


0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,


1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,


0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,


1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]


#print(cls_train)

print(cls_train.count(0))

print(cls_train.count(1))

print(len(cls_train))

cls_test=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
```

```
#print len([name for name in os.listdir(path2) if os.path.isfile(os.path.join(path2, name))])

#print(len(cls_test))
#SAVING NPZ FILE
np.savez_compressed(file_name2+'.npz',x_train=train2,y_train=cls_train,x_test=test,y_test=cls_tes
t)
print("Succesfully array conversion done");
```

Training and Saving the Model

```
#IMPORTING PACKAGES import Pre_processing_video

import numpy as np
import keras
import LoadDataset
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras import backend as K
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictions
from keras.models import Model
import timeit
from keras.layers.normalization import BatchNormalization
```

```
import pylab as plt

import warnings

warnings.filterwarnings('ignore')

import json

from keras.models import model_from_json

from keras.utils.vis_utils import plot_model

 #PREPARING DATASET


batch_size = 100

num_classes = 2

epochs = 20




#INPUT IMAGES DIMENSION


img_rows, img_cols = 256, 256


#LOADING DATASET

(x_train, y_train), (x_test, y_test) = LoadDataset.load_data()


if K.image_data_format() == 'channels_first':

    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)

    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)

    input_shape = (1, img_rows, img_cols)
```

```
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 3)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 3)
    input_shape = (img_rows, img_cols, 3)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

#convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

#BUILDING A MODEL
model = Sequential()
model.add(Conv2D(filters=6,input_shape=(256,256,3),kernel_size=(3,3),strides=(2,2),activation='r
elu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))
model.add(BatchNormalization())
model.add(Conv2D(filters=8,kernel_size=(3,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))

model.add(BatchNormalization())

model.add(Conv2D(filters=12,kernel_size=(3,3),strides=(1,1),padding="valid",activation='relu'))

model.add(BatchNormalization())

model.add(Conv2D(filters=16,kernel_size=(3,3),strides=(1,1),padding="valid",activation='relu'))

model.add(BatchNormalization())

model.add(Flatten())

model.add(Dense(4096,input_shape=(img_rows,img_cols,3),activation='relu'))

model.add(Dropout(0.4))

model.add(BatchNormalization())

model.add(Dense(4096,activation='relu'))

model.add(Dropout(0.4))

model.add(BatchN

        ormalization())

model.add(Dense(1000,activation='relu'))

model.add(Dropout(0.4))

model.add(BatchNormalization())

#plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)


#OUTPUT LAYER

model.add(Dense(num_classes,activation='softmax'))

model.summary()


#MODEL TRAINING
```

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])


model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
#plot(history)
#TRANING EVALUATION
score = model.evaluate(x_train, y_train, verbose=0)
print('Train loss:', score[0])
print('Train accuracy:', score[1])


#TESTING EVALUATION
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])


# serialize model to JSON
model_json = model.to_json()
with open("Model5.json", "w") as json_file:
    json_file.write(model_json)
```

```
# serialize weights to HDF5
model.save_weights("Model5.h5")
print("Saved model to disk")


# later...


# load json and create model
json_file = open('Model5.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights("Model5.h5")
print("Loaded model from disk")


# evaluate loaded model on test data
loaded_model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
score = loaded_model.evaluate(x_test, y_test, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
'''#PREDICTION
obj=open("Testing.txt","w")
for i in range(0,200):
    plt.imshow(x_test[i].reshape(256,256,3))
    plt.show()
    prediction = model.predict(x_test[i:i+1])
    print('Prediction Score:\n',prediction[0])
    thresholded = (prediction>0.5)*1
    print(thresholded)
    print('\nThresholded Score:\n',thresholded[0])
    Pi=np.where(thresholded == 1)[1][0]
    print('\nPredicted Image:\n',Pi)
    obj.write(str(i))
    obj.write("->")
    obj.write(str(Pi))
    obj.write("\n")
obj.close()'''
```

Detecting the Elephant from the Video

```
import cv2    # for capturing videos
import math   # for mathematical operations
import matplotlib.pyplot as plt    # for plotting the images
import pandas as pd
from keras.preprocessing import image   # for preprocessing the images
import numpy as np    # for mathematical operations
from keras.utils import np_utils
from skimage.transform import resize   # for resizing images
import shutil
import os

#import LoadDataset

import json

from keras.models import model_from_json

import numpy as np

from matplotlib import pyplot as plt

from PIL import Image

import fnmatch

import glob


#Specifications

list_pos=[]

pos_c=0

file_name="video1"

width ,height= 256 , 256

dim = (width, height)

x=[]

pos_count=0

total_count= 0

img_rows, img_cols = 256, 256

f_count=0


#Path Specifications
```

```
path="D:\\15CSR191_227_241\\Video Detection"

path1="D:\\15CSR191_227_241\\Dataset\\Generated Images"



#Removing the existing files


for file in os.listdir(path1):

    if file.endswith(".jpg"):

        random=os.path.join(path1,file)

        os.remove(random)


# Read the video

videoFile = "D:\\15CSR191_227_241\\Video Detection\\Sample Video\\test.mp4"

cap = cv2.VideoCapture(videoFile)# capturing the video from the given path

frameRate = cap.get(5) #frame rate


# load json


json_file = open('modjan22hostel.json', 'r')

loaded_model_json = json_file.read()

loaded_model = model_from_json(loaded_model_json)


# load weights into new model
```

```
loaded_model.load_weights("modjan22hostel.h5")

print("Loaded model from disk")


#extract frames from it and save them as images


while(cap.isOpened()):
    frameId = cap.get(0) #current frame number
    ret, frame = cap.read()
    if (ret != True):
        break
    if (frameId % math.floor(frameRate) == 0):

        #Saving the images

        filename ="frame%d.jpg" % total_count;total_count+=1
        cv2.imwrite(filename, frame)
        f_count+=1
        #Resizing the image

        random=os.path.join(path,filename)
        img = cv2.imread(random, cv2.IMREAD_UNCHANGED)
        resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
```

```
#Array conversion

x_t=np.array(resized)

millis = cap.get( 0 )
millis = int(millis)
seconds=(millis/1000)%60
seconds = int(seconds)
minutes=(millis/(1000*60))%60
minutes = int(minutes)
hours=(millis/(1000*60*60))%24

#PREDICTION

x_t = x_t.reshape(1, img_rows, img_cols, 3)
prediction = loaded_model.predict(x_t)
#print(prediction)
thresholded = (prediction>0.5)*1
print(thresholded)
Pi=np.where(thresholded == 0)[1][0]
#print(Pi)
if(Pi==1):
    print(filename)
    print ("At %d:%d:%d ==> Elephant Detected" % (hours, minutes, seconds))
```

```
            pos_count=pos_count+1

            pos_c=pos_c+1

            plt.imshow(x_t.reshape(256,256,3))

            #plt.show()

            list_pos.append(filename)

            if(f_count%15==0 and pos_count>=5):

                if(len(list_pos)>=5):

                    print("Send SMS ")

                list_pos=[]

                pos_count=0


#Moving All the images to Generated folder


for file in os.listdir(path):

    if file.endswith(".jpg"):

        random=os.path.join(path,file)

        shutil.move(random,"D:\\15CSR191_227_241\\Dataset\\Generated Images")


print("Total:",total_count)

print("Total Detection as Positive:",pos_c)


print ("Video Extraction was successful!")

cap.release()

json_file.close()
```

REFERENCES

- Isha Dua,Pushkar Shukla, Ankush Mittae.,"A Vision based Human - Elepahant Collision Detection System ",Third International Conference on Image Infonnation Processing of the IEEE,2015.
- S. J. Sugumar,R. Jayaparvathy.,"An Improved Real Time Image Detection System for Elephant Intrusion along the Forest Border Areas",Hindawi Publishing Corporation.
- K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", ICLR, 2015.
- Guo lili, Ding shifei, "Deep learning research progress [J]", 2015.
- Tianmei Guo,Jiwen Dong,Henjian Li,Yunxing Gao.,"Simple convolutional neural network on image classification",2nd International Conference on Big Data Analysis (ICBDA) of the IEEE,2017.
- Nadia Jmour,Sehla Zayen,Afef Abdelkrim.,"Convolutional neural networks for image classification",International Conference on Advanced Systems and Electric Technologies (IC_ASET) of the IEEE, 2018.
- Xiaobin Li ,Shengjin Wang.,"Object Detection Using Convolutional Neural Networks in a Coarse-to-Fine Manner", IEEE Geoscience and Remote Sensing Letters (Volume: 14 , Issue: 11 , Nov. 2017).
- Duth P. Sudharshan ; Swathi Raj.,"Object recognition in images using convolutional neural network",2nd International Conference on Inventive Systems and Control (ICISC) of the IEEE, 2018.
- Luis Tobías,Aurélien Ducournau,François Rousseau,Grégoire Mercier, Ronan Fablet.,"Convolutional Neural Networks for object recognition on mobile devices: A case study",23rd International Conference on Pattern Recognition (ICPR) of the IEEE,2016.
- Fatima Zahra Ouadiay ; Hamza Bouftaih ; El Houssine Bouyakhf ; M. Majid Himmi.,"Simultaneous object detection and localization using convolutional neural networks",International Conference on Intelligent Systems and Computer Vision (ISCV) of the IEEE,2018.