# AUTOMATICALLY DETECTING ANIMAL USING SINGLE SHOT DETECTION

PROJECT PHASE II REPORT

*submitted in partial fulfillment*

*of requirements for the award of the degree in*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**by**

**YASH RAJ PANDEY** **(191061101214)**

**SURAN PAL SINGH BALI** **(191061101189)**

**SRIKANT M** **(191061101184)**


**Dr. M.G.R.**
**EDUCATIONAL AND RESEARCH INSTITUTE**
**DEEMED TO BE UNIVERSITY**
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**APRIL 2023**

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## BONAFIDE CERTIFICATE

This is to certify that this project phase-II report is the bonafide work of

| | |
|---|---|
| **YASH RAJ PANDEY** | **(191061101214)** |
| **SURAN PAL SINGH BALI** | **(191061101189)** |
| **SRIKANT M** | **(191061101184)** |

who carried out the project entitled "**AUTOMATICALLY DETECTING ANIMAL USING SINGLE SHOT DETECTION**" under our supervision from December 2022 to April 2023.

| **Supervisor** | **Project Coordinator** | **Department Head** |
|---|---|---|
| **Mrs. O. Vasantha Kumari** | **Dr. P. Dinesh Kumar** | **Dr. S. Geetha** |
| Assistant Professor<br>Department of CSE<br>Dr. MGR Educational and Research Institute,<br>Deemed to be University | Assistant Professor<br>Department of CSE<br>Dr. M.G.R Educational and Research Institute,<br>Deemed to be University | Professor<br>Department of CSE<br>Dr. M.G.R Educational and Research University,<br>Deemed to be University |

**Submitted for the Viva Voce Examination held on _____**

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

# DECLARATION

We **Mr. YASH RAJ PANDEY (191061101214), Mr. SURANPAL SINGH BALI (191061101189), Mr. SRIKANTH .M (191061101185),** hereby declare that the project phase 2 report entitled **"AUTOMATICALLY DETECTING ANIMALS USING SINGLE SHOT DETECTION"** is done by us under the guidance of **Mrs. O. VASANTHA KUMARI** and is submitted in partial fulfilment of the requirements for the award of the degree in **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING**.

**1.**

**2.**

**DATE:** **3.**

**PLACE: CHENNAI**

SIGNATURE OF THE CANDIDATE(S)

# ACKNOWLEDGEMENT

We would first like to thank our beloved Chancellor **Dr. A.C. SHANMUGAM, B.A., B.L.** and President **Er. A.C.S. Arunkumar, B.Tech., M.B.A.,** and for all the encouragement and support extended to us during the tenure of this project and also our years of studies in his wonderful University.

We express my heartfelt thanks to our Vice Chancellor **Prof. Dr. S. GEETHALAKSHMI** in providing all the support of my Project (Project Phase-II).

We express my heartfelt thanks to our Head of the Department, **Prof. Dr. S. Geetha,** who has been actively involved and very influential from the start till the completion of our project.

Our sincere thanks to our Project Coordinator **Dr. VICTO SUDHA GEORGE** and **Dr. P. DINESH KUMAR** and our Project guide **Mrs. O. VASNTHA KUMARI** for their continuous guidance and encouragement throughout this work, which has made the project a success. I would also like to thank all the teaching and non-teaching staffs of Computer Science and Engineering and Information Technology department, for their constant support and the encouragement given to us while we went about to achieving my project goals.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

Crop damage caused by animal attacks is one of the major threats in reducing the crop yield. Due to the expansion of cultivated land into previous wildlife habitat, crop raiding is becoming one of the most antagonizing human-wildlife conflicts. Farmers in India face serious threats from pests, natural calamities & damage by animals resulting in lower yields. Traditional methods followed by farmers are not that effective and it is not feasible to hire guards to keep an eye on crops and prevent wild animals. Since safety of both human and animal is equally vital, it is important to protect the crops from damage caused by animal as well as divert the animal without any harm. Thus, in order to overcome above problems and to reach our aim, we use machine learning to detect animals, entering into our farm by using deep neural network concept, a division in computer vision. In this project, we will monitor the entire farm at regular intervals through a camera which will be recording the surrounding throughout the day. With the help of a machine learning model, we detect the entry of animals and we play appropriate sounds to drive the animal away. This report specifies various libraries and concepts of convolutional neural networks used to create the model.

# CHAPTER 1

# INTRODUCTION

Agriculture meets the food demands of the population and also provides various raw materials for different industries. Interference of animals in agricultural lands causes a huge loss of crops. Crop damage due to raiding wild animals has become a major issue of concern these days. Animals like wild boars, macaques, porcupines, deer, monkeys and bears are extremely destructive and have also caused human casualties in certain occasions. Small farmers can even lose up to half of their yield to animals and they cannot take any harsh measures due to the strict wildlife laws. Human-elephant conflict is rising intensely as elephants are a highly conflict prone wildlife species, especially in India. Thus, there is need for a system to detect any intrusion which can help the farmers to drive away these animals as soon as they learn about their intrusion.

Computer vision is applicable to many fields like medical field, robotics, remote sensing, machine vision, content-based image retrieval. Computer vision solves many problems in different disciplines. Computer vision also applied in the security field to perform automatic surveillance and access control and attendance management. The computer vision can be applied in agriculture field in many ways like disease detection of a tree by examining leaves or flowers or fruits and quality control of agricultural products.

The computer vision techniques can be applied in order to provide security from wild animals in agriculture. In agriculture fields near to forest areas have a severe threat from wild animals, which attacks regularly on farms. These attacks causing huge damage to agricultural crops subsequently causes significant financial losses to farmers.

Some measures are taken by the farmers by installing electrical fences to the farms, big flood lights in the farm. Some even resort to hiring guards. Installing an electrical fence is much costlier to equip huge farms and kills so many animals, which is even illegal in certain places and affects the biodiversity. Other existing techniques also are not effective due to several reasons, cost being one of them.

In this project, we proposed a new and cost effective solution for agriculture security from animals. It is a proactive solution which gives alerts to the farmers when animals come near

to the farms. It also causes certain siren to be played whenever any animals are detected and is directed towards the animal in an attempt to scare them away. Here, we are implementing a solution that recognizes animals when it is captured on camera. Furthermore, it increases the efficiency and productivity of laboratory staff by reducing the time spent in direct monitoring of animals, also, to have a better understanding of animal behavior. Deep (CNN) or ConvNet has accomplished important success in the computer vision field, like target detection, target tracking, image classification, and semantic image segmentation. It is a multi-layered neural network with a special architecture to detect complex features in data. Object detection means to predict the location of an object along with its category in static images which is one of the most critical computer vision problems, It often uses extracted characteristics and learning algorithms to identify the object belonging to a certain category of objects in a static image . Several types of CNNs are existing; Mobile Net is considered as one of the important applications for this type of CNN networks. The architecture of "Depthwise Separable Convolutions" greatly decreases the number of parameters compared to the normal CNN that has the same depth. This results in lightweight neural networks. The MobileNetSSD Model is a Single-Shot Multibox Detection (SSD) used as the pre-trained models for detecting and classification multi animals in difficult cases like detect part of an animal in lighting/illumination conditions, This combining between them gives us fast and sufficient performance to detect and classify the object. This architecture is more suitable for mobile and vision-based mobile applications where there is a lack of computing power. SSD approach utilizes deep neural network technique to detect and classify the target objects and this limits the output area for bounding boxes in the default boxes set over different aspect ratios and scales per each spatial position of the feature map. At the prediction time, the network introduces scores to the presence of each object class in every default box. As well, it inserts modification to these boxes for getting the best matching of the object's shape. Then, the current network collects predictions from multi-feature maps with various scales for handling objects of different volumes. Biodiversity is the abundance and diversity of life on the earth. It is our planet's most complicated and vital feature. Biodiversity is important for both ecological and economic reasons. From genes to ecosystems, it comprises the biological, ecological, and sociocultural processes that keep life on Earth going. However, the world's biodiversity is threatened and is at high risk due to technological and economical advancements, water pollution and air pollution, and some other factors such as demographic changes. The number of threatened species is increasing at an alarming rate. As of 2019, the International Union for Conservation of Nature has classified 13,868 animal species and

14,360 plant species as endangered, up from 1,102 animal species and 1,197 plant species listed in 1996. According to the WWF's Living Planet Report 2020, global population proportions of amphibians, birds, fish mammals, and reptiles decreased by 68 percent between 1970 and 2016. Many organizations, however, are currently attempting to improve the situation. Nowadays, technologies like machine learning (ML) and Deep Learning (DL) have the potential to teach computers to identify a wide range of items from images. Visual information, according to wildlife researchers, gives irrefutable evidence of an animal's presence by capturing image frames and recognizing them as endangered species without human intervention or support, which would aid researchers, ecologists, and scientists in classifying and preserving these animals. These animal detection systems can further help in preventing animal-vehicle collisions which result in death, injury, and also property damage. This chapter provides a comparative analysis of different algorithms for the identification and detection of animals within their natural habitat, this would indeed help beginners in their research work.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 **Parikh, M. et al. "Wild-Animal Recognition in Agriculture Farms Using W-COHOG for Agro-Security." (2017).**

Computer Vision is applied in agriculture field for food grading, disease identification of the plants and agro-farms security. Huge crop damage is caused by the wild animal attacks on the agriculture farms. Here are some traditional techniques followed by the local farmers, but which are not effective. This problem can be solved using computer vision techniques. In this paper, we proposed an algorithm to detect animals in a given image. WCoHOG is a Histogram oriented gradients based feature vector with better accuracy. It is an extension of Co-occurrence Histograms of Oriented Gradients (CoHOG). In this paper LIBLINEAR classifier is used in order to get better accuracy for high dimensional data. The experiments were conducted on two benchmark datasets called Wild-Anim and CamaraTrap dataset. Experimental results prove that W-CoHOG performs better than existing state of the art methods.

**Summary:** This journal discusses about the Computer Vision and its application in the field of agricultural safety.

## 2.2 S. Yadahalli, A. Parmar and A. Deshpande, "Smart Intrusion Detection System for Crop Protection by using Arduino," *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2020, pp. 405-408, doi: 10.1109/ICIRCA48905.2020.9182868.

Agriculture is still one of the most crucial sectors of the Indian economy. It is important for human survival as well as economic growth. Traditional systems like humanoid scarecrows are used even today in an agricultural field to stop birds and animals from disturbing and feeding on growing crops. There are many loopholes in such ideas and so enhancing agricultural security has become a major issue these days. Thus, this paper focuses on proposing a system which detects the intruders, monitors any malicious activity and then

reports it to the owner of the system. It acts as an adaptable system which provides a practicable system to the farmers for ensuring complete safety of their farmlands from any attacks or trespassing activities.

**Summary:** This journal helps us in understanding the use of AI and its integration with motion detector and IR sensor to the agricultural field.

## 2.3 Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications."

We present a class of efficient models called MobileNets for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks. We introduce two simple global hyperparameters that efficiently tradeoff between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. We present extensive experiments on resource and accuracy tradeoffs and show strong performance compared to other popular models on ImageNet classification. We then demonstrate the effectiveness of MobileNets across a wide range of applications and use cases including object detection, finegrain classification, face attributes and large scale geo-localization.

**Summary:** In this paper, we learn the about MobileNet SSD (Single Shot Detection) which we have extensively used in our project. It offers massive gains in efficiency over models like ResNet, etc and therefore can be used in low end hardware devices.

## 2.4 Deshpande, Abhinav. (2016). Design and Implementation of an Intelligent Security System for Farm Protection from Wild Animals. 5. pp.2319-7064.

Crops are vulnerable to wild animals. Therefore, it is very important to monitor the nearby presence of animals. Then the actuation of various devices should follow to repel the hazardous animals. Traditional methods have been widely applied depending on the kinds of produce and imperiling animals. In this paper, we propose a method to protect farms from wild animals via ubiquitous wired network devices, which is applied to farm along with traditional methods to improve the protection performance. Operational amplifier circuits are utilized mainly for the detection of animal intrusion from the outside of farms. The proposed monitoring scheme is to provide an early warning about possible intrusion and damage by wild animals.

**Summary:** In this paper, we learn about animal detection and protection of crops using embedded systems.

# CHAPTER 3

# REQUIREMENT ANALYSIS

## HARDWARE & SOFTWARE REQUIREMENTS

**H/W Configuration:**

- Processor - I3/Intel Processor
- RAM - 4GB (min)
- Hard Disk - 128 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - Any

**S/W Configuration:**

- Operating System : Windows 7+
- Server side Script : Python 3.6+
- IDE : PyCharm
- Libraries Used : Pandas, Numpy, playsound, collections, time, os, imutils, OpenCV.
- Dataset : MS COCO Image Dataset.

### 3.1 SOFTWARE COMPONENTS

***3.1.1PYTHON :***

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know

what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

• Open-source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/Obj C/Java/Fortran

• Easy-is to interface with C++ (via SWIG)

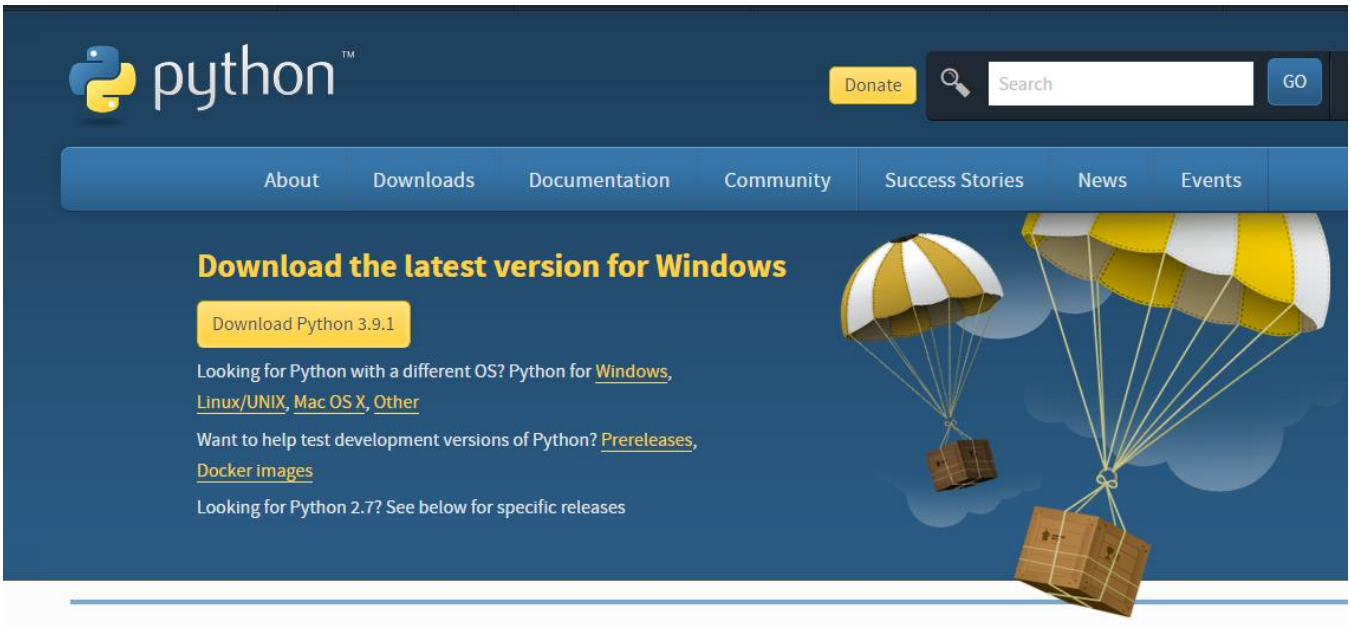
• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## Installing Python:

- To download and install Python visit the official website of Python https://www.python.org/downloads/ and choose your version.




- Once the download is complete, run the exe for install Python. Now click on Install Now.
- You can see Python installing at this point.
- When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

### *3.1.2 PYCHARM*

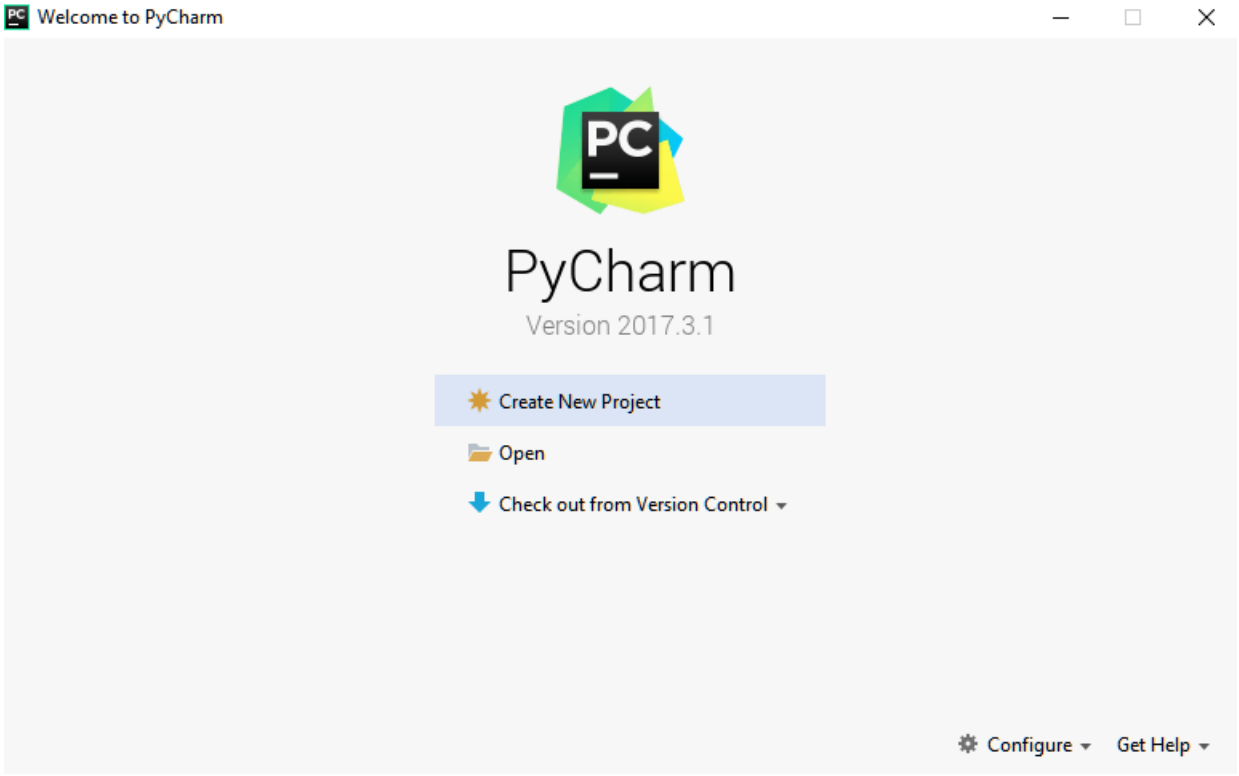
PyCharm is a hybrid platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE!

Installing PyCharm:

1. To download PyCharm visit the website https://www.jetbrains.com/pycharm/download/ and Click the "DOWNLOAD" link under the Community Section.

2. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.

3. On the next screen, Change the installation path if required. Click “Next”.
4. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
5. Choose the start menu folder. Keep selected JetBrains and click on “Install”.
6. Wait for the installation to finish.
7. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.
8. After you click on "Finish," the Following screen will appear.

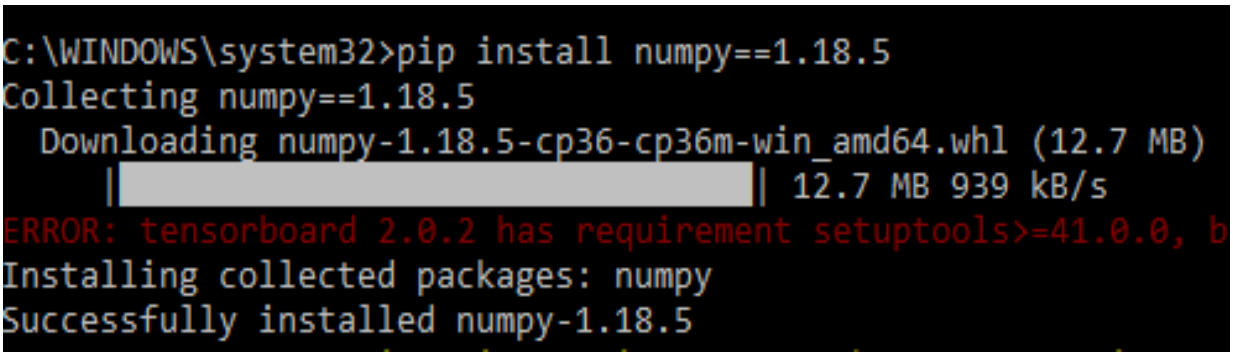



9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like numpy, pandas, seaborn, scikit-learn, matplotlib. pyplot)

Ex: pip install numpy




## 3.2 HARDWARE COMPONENTS

### *3.2.1 I3 Processor*

An Intel Corei3 is an Intel proprietary processor that is built on the framework of multiprocessor architecture. It is a type of dual-core processor with an integrated graphic processing unit (GPU). It is a successor of the Core 2 series of processors produced by Intel. It can be installed within mobile, desktop and embedded devices.

### *3.2.2 8GB RAM*

With 8 GB of RAM, **you will have enough memory to run several programs at once**. You can open lots of browser tabs at once, use photo or video editing programs, stream content, and play mid-to-high-end games. Many Windows 10 and macOS computers or laptops come with 8 GB of memory installed these days.

# CHAPTER 4

# EXISTING SYSTEM

Since most of the farms in India are small, most farmers relies on medieval techniques like using a scare crow, or relying on guards to monitor crops. More recently, crops are also being protected using electric fencing but it can be highly cost inefficient which a small farmer cannot afford. Even if they can afford it, in most cases it is illegal to use such fences which governments uses as a measure to conserve the wildlife populations.

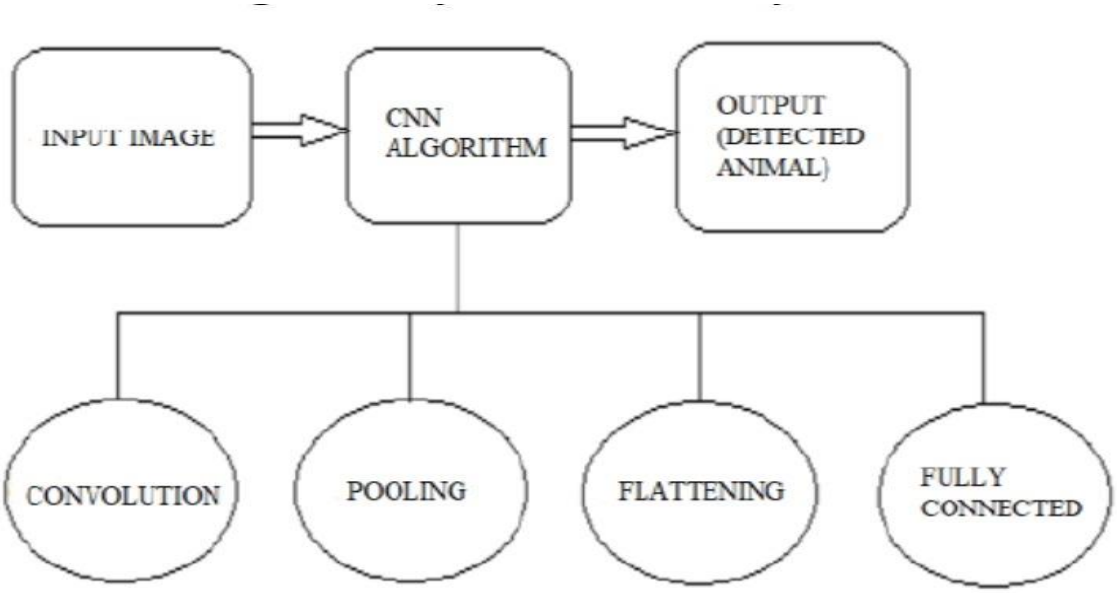
Also, in busy seasons like the harvesting time, it can get difficult to have a guard, guarding and monitoring the crops from animals.

**DISADVANTAGES:**

- High cost.
- Prone to seasonality.
- Requires costly equipment’s and infrastructure.
- Are highly inefficient.

# CHAPTER 5
# ARCHITECTURE DESIGN SPECIFICATION

## 5.1 ARCHITECTURE DIAGRAM




**5.1 Architecture Diagram of the proposed system**

## 5.2 UML DIAGRAM :

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

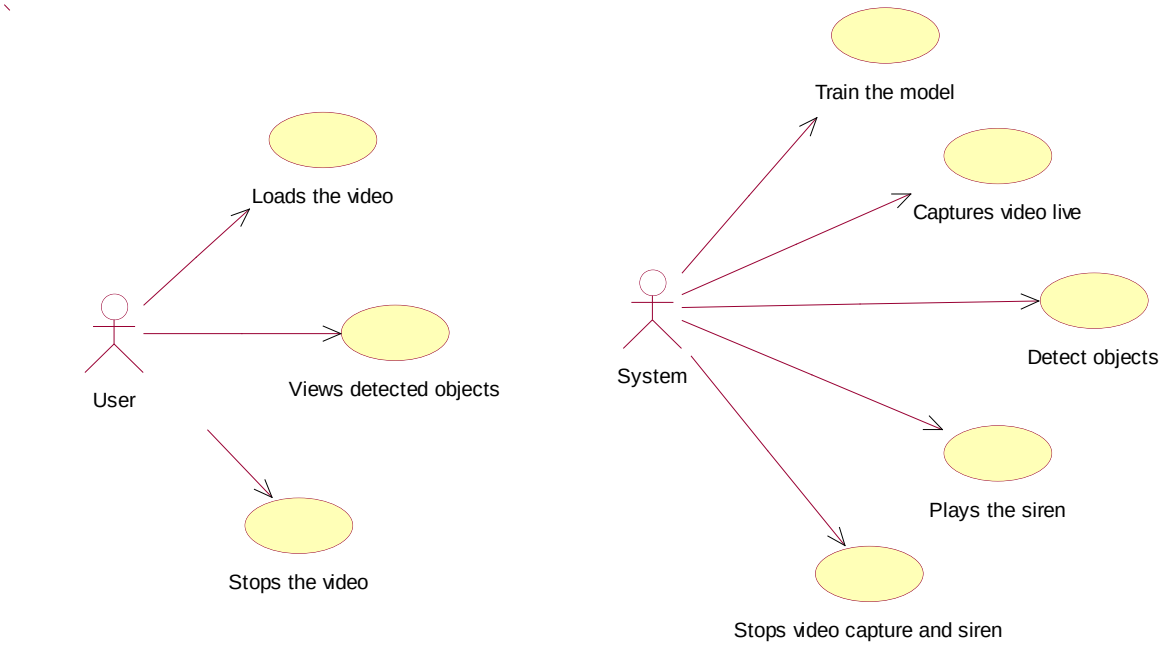
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

### 5.2.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

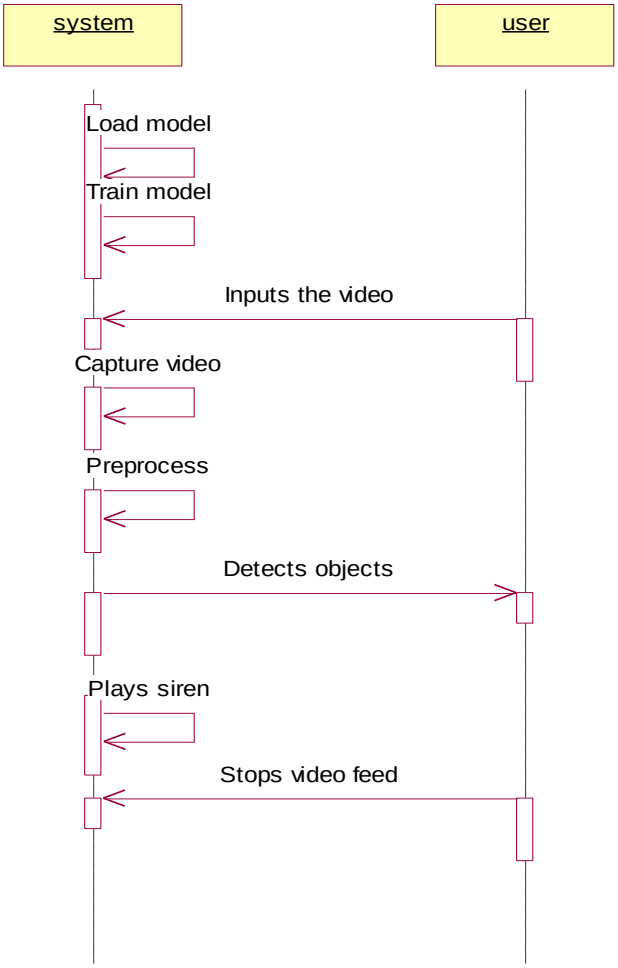



### *5.2.2 CLASS DIAGRAM:*

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

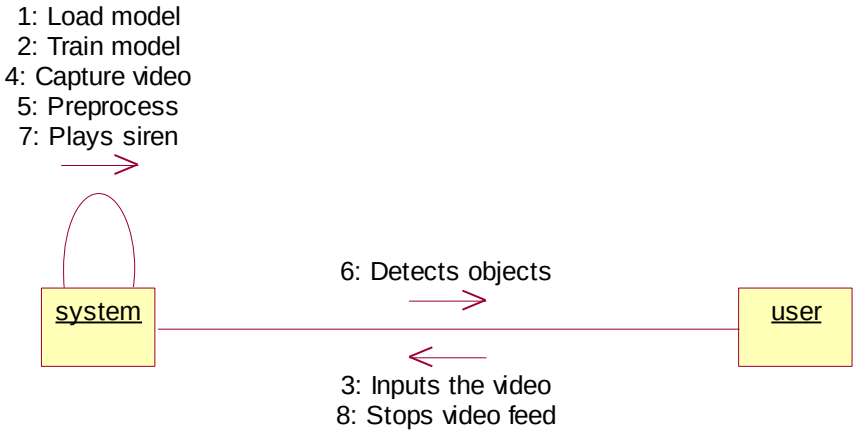
### *5.2.3 SEQUENCE DIAGRAM:*

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



### *5.2.4 COLLABORATION DIAGRAM:*

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.



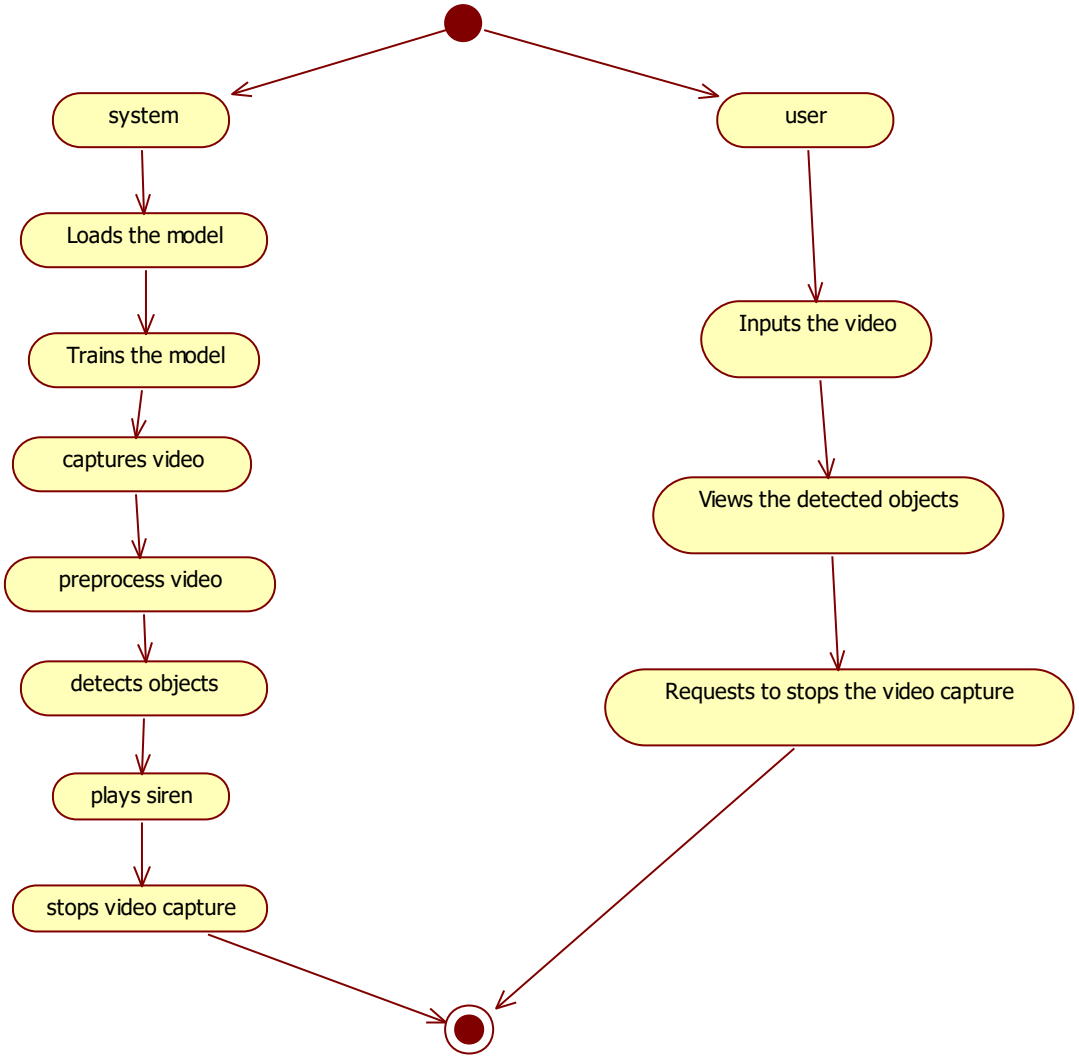

### *5.2.5 DEPLOYMENT DIAGRAM*

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.


### *5.2.6 ACTIVITY DIAGRAM:*

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-

step workflows of components in a system. An activity diagram shows the overall flow of control.



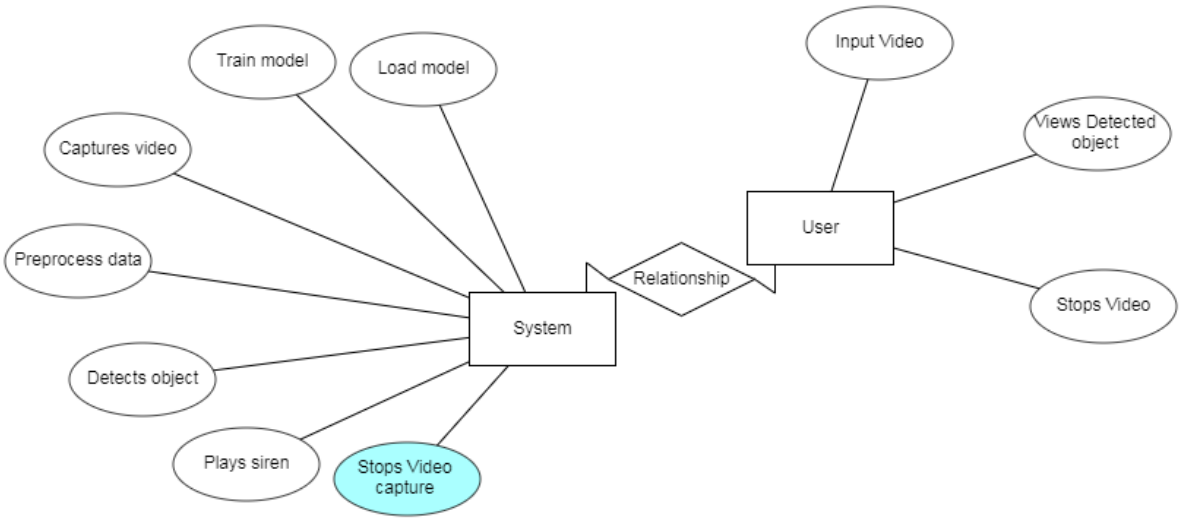
## 5.2.7 COMPONENT DIAGRAM

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical **c**omponents in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.


## 5.2.8 ER DIAGRAM:

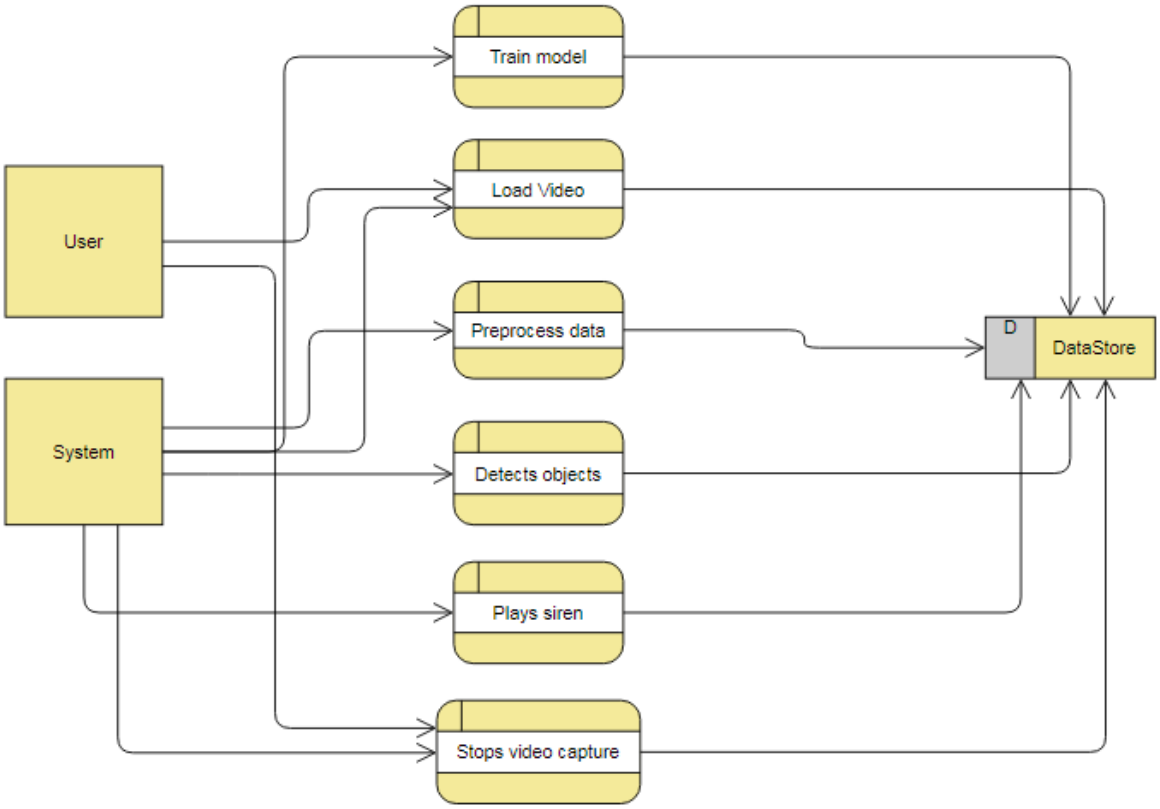
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

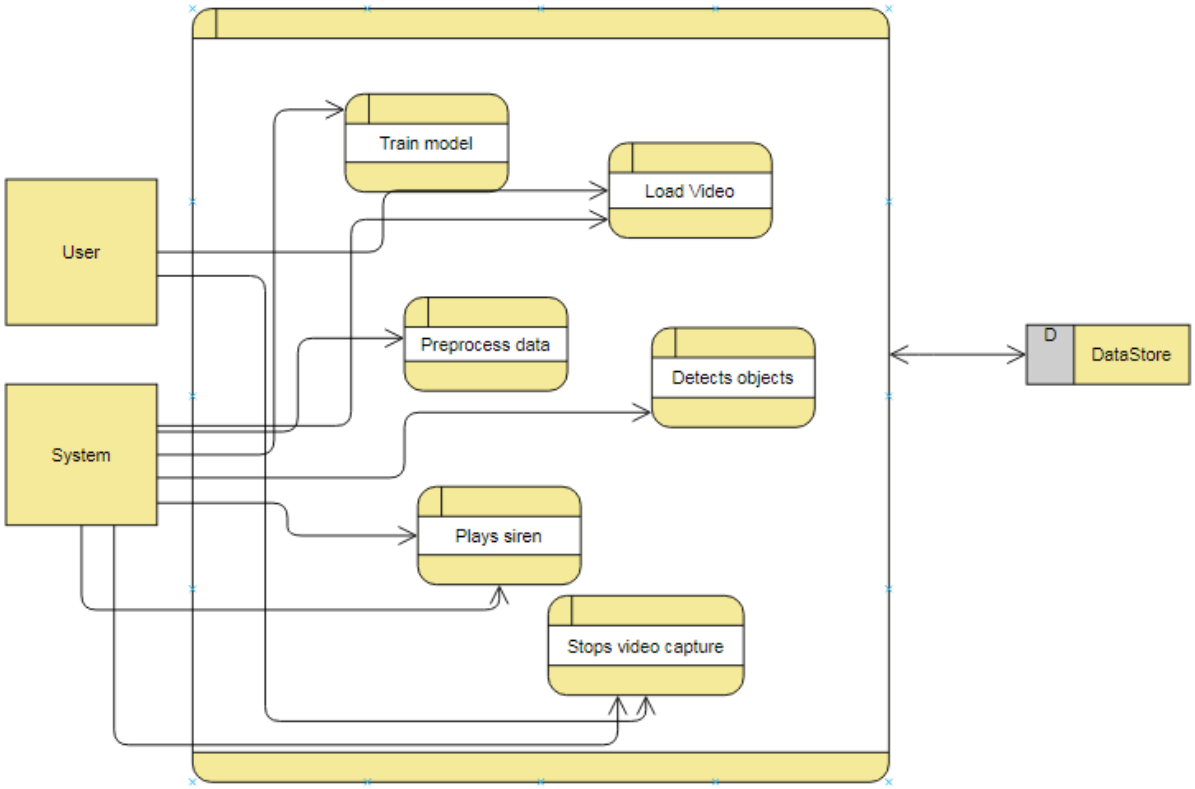
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let’s have a look at a simple ER diagram to understand this concept.




### 5.2.9 DFD DIAGRAM:

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.


Train model
Load Video
User
Preprocess data
D
DataStore
System
Detects objects
Plays siren
Stops video capture


Train model
Load Video
User
Preprocess data
Detects objects
D
DataStore
System
Plays siren
Stops video capture

# CHAPTER 6
# PROPOSED SYSTEM

We propose an AI based surveillance system to detect and monitor the presence of any animal. A camera can be placed conveniently at location(s) where any possible animal might enter from. The system uses computer vision using OpenCV to process the feed from the camera. Pre-trained model Mobile Net SSD (Single Shot Detector) is used to detect the animals in the farms. The model is trained on MS COCO image dataset. A siren is fired on detecting an animal which can act as a deterrent to the animal. It can also notify the farmer so that he/she can take the concerned action as required in time.

## 6.1 UPLOAD(LIVE):

Upload a video as a live feed using a webcam (or any camera attached in a farm).

## 6.2 VIEW:

Video can be viewed live in a dialog box.

## 6.3 DATA PREPROCESSING:

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. Cleaning the data refers to removing the null values, filling the null values with meaningful value, removing duplicate values, removing outliers, removing unwanted attributes. If dataset contains any categorical records means convert those categorical variables to numerical values.

In this case, we are taking a live video feed in the form of images and resizing them to a standard size.

### 6.4 IDENTIFYING FEATURES:

We use Mobile Net SSD pretrained model which identifies features in any image using a Convolution Neural Network (CNN) model.

### 6.5 THE MODEL:

- SSD (Single Shot Detector) is a popular algorithm in object detection.
- It's generally faster than RCNN.
- SSD has two components: a backbone model and SSD head. *Backbone* model usually is a pre-trained image classification network as a feature extractor.
- Here, we will use Mobile Net SSD model to detect the objects.
- Here, VGG Net is used as a backbone model to extract the features from the images.
- Convolution layers (CNN) are then used for object detection in the images using the feature map generated by VGG net layer.
- The model is able to detect multiple objects in any given image.
- For the purpose of classification, the model uses softmax in the last layer.
- SoftMax takes in a vector of numbers and converts them to probabilities which are then used for image generating results.
- SoftMax converts logits into probabilities by taking the exponents from every output and then normalize each of these numbers by the sum of such exponents, such that the entire output vector adds up to one.

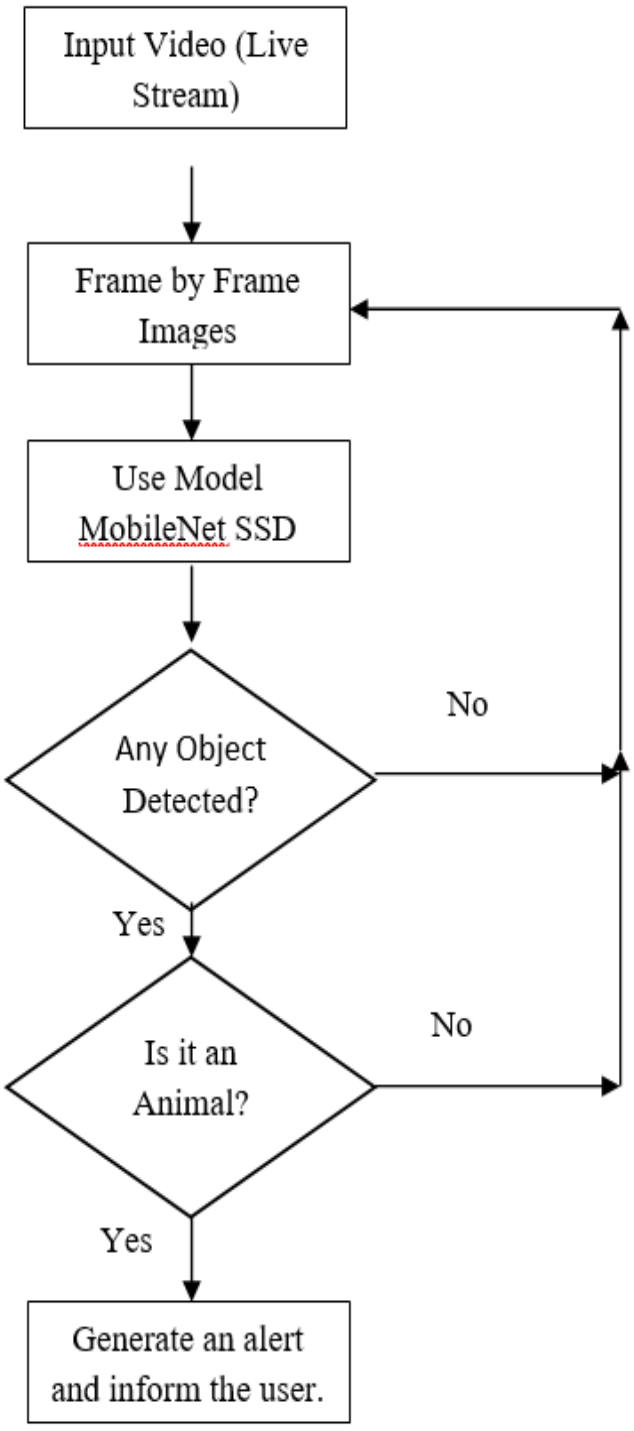
### 6.6 PREDICTION:

A live video feed is taken in frame by frame as individual images. These images the then fed into the model after preprocessing to detect animals (if any exists).

### 6.7 USER INTERFACE:

A dialog box opens up while taking in the live video feed. The frames or images from the video are used to detect objects. The objects are then bounded in a bounding box along with a label and the probability of success in also displayed in there. A siren is then played if any animal is detected for a while.

**FLOW OF DIAGRAM:**



## STEPS FOR EXECUTING THE PROJECTS

1. Import all the Libraries/packages.
2. Load the pretrained ImageNet SSD model.
3. Load MobileNetSSD_deploy.prototxt.txt which defines all the layers in the model.
4. Start the live video feed.
5. Take the live video frame by frame as images.
6. The images are then preprocessed.
7. We use OpenCV's blobFromImage which performs certain preprocessing to convert it as a 4-dimensional blob.
8. All the objects (if any) are detected in that blob using the model.
9. If the probability of those detected exceeds certain threshold (which is 0.2 in our case), only then we will consider that the object is present.
10. The objects is then bounded in a bounding box.
11. A label along with the probability of success is displayed to the user in the live feed itself.
12. A siren is played if an animal is detected for a while.
13. The live video feed can be closed using the key 'q'.

# CHAPTER 7

## ALGORITHM:

## OpenCV:

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

Using OpenCV library, we can −

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

**Convolutional Neural Network**

- A convolutional neural network (CNN) uses a variation of the multilayer perceptron's. A CNN contains one or more than one convolutional layers. These layers can either be completely interconnected or pooled.
- Before passing the result to the next layer, the convolutional layer uses a convolutional operation on the input. Due to this convolutional operation, the network can be much deeper but with much fewer parameters.
- Due to this ability, convolutional neural networks show very effective results in image and video recognition, natural language processing, and recommender systems.

- Convolutional neural networks also show great results in semantic parsing and paraphrase detection. They are also applied in signal processing and image classification.

**Let us consider the basic part of Convolutional Neural Network:**

**1. Input Layer:** It is the layer where we provide the input for the model. The number of features our input has is equal to the number of neuron in the input layer.

2. **Hidden Layer:** The input features are transferred to the hidden layer(s) where different processes/activities takes place. There can be multiple hidden layers. The layers undergoes mathematical operations like matrix multiplication, convolutions, pooling etc. along with an activation function.

3**. Output Layer:** They layer which is used to generate probability scores using sigmoid or SoftMax functions which is then converted to the output of our model.

**Image Input Layer:**

Create an image input layer using image input layer. An image input layer inputs images to a network and applies data normalization. Specify the image size using the input Size argument. The size of an image corresponds to the height, width, and the number of color channels of that image. For example, for a grayscale image, the number of channels is 1, and for a color image it is 3.

**Convolution Layer:**

Convolutional layers are the major building blocks used in convolutional neural networks.

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

**Pooling Layer:**

It is common to periodically insert a Pooling layer in-between successive Convolution layers in a CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

**Input Sequence Layer:**

A sequence input layer inputs sequence data to a network. In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup, which is what people commonly refer to when mentioning "sequence to sequence models" with no further context. Here's how it works:

In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup, which is what people commonly refer to when mentioning "sequence to sequence models". Here's how it works:

- A RNN layer (or stack thereof) acts as "encoder": it processes the input sequence and returns its own internal state. Note that we discard the outputs of the encoder RNN, only recovering the state.
- Another RNN layer (or stack thereof) acts as "decoder": it is trained to predict the next characters of the target sequence, given previous characters of the target sequence.

**Output Layers:**

**SoftMax and Classification Layers:**

SoftMax converts logits into probabilities by taking the exponents from every output and then norms each of these numbers by the sum of such exponents, such that the entire output

vector adds up to one – every probability should be one. Generally, cross-entropy loss is the loss of such a problem in several classes. In the last layer of an image classification network such as CNN (e.g. VGG16) used in ImageNet competitions, softmax is also applied.

A SoftMax layer applies a SoftMax function to the input. A classification layer computes the cross-entropy loss for multi-class classification problems with mutually exclusive classes. Create a classification layer using classification Layer. For classification problems, a softmax layer and then a classification layer must follow the final fully connected layer. The softmax function is also known as the normalized exponential and can be considered the multi-class generalization of the logistic sigmoid function. For typical classification networks, the classification layer must follow the softmax layer. In the classification layer, train Network takes the values from the Softmax function.

# CHAPTER 8

## CODE:

### DDN_DETECTION LIBRARIES

```
# Import necessary libraries
import numpy as np
import argparse
import cv2

# Construct the argument parse and parse the arguments
ap = argparse.ArgumentParser(
    description='Script to run MobileNet-SSD object detection network')
ap.add_argument('-v', '--video', type=str, default='',
                help='Path to video file. If empty, web cam stream will be
used')
ap.add_argument('-p', '--prototxt', required=True,
                help="Path to Caffe 'deploy' prototxt file")
ap.add_argument('-m', '--model', required=True,
                help='Path to weights for Caffe model')
ap.add_argument('-l', '--labels', required=True,
                help='Path to labels for dataset')
ap.add_argument('-c', '--confidence', type=float, default=0.2,
                help='Minimum probability to filter weak detections')
args = vars(ap.parse_args())

# Initialize class labels of the dataset
CLASSES = [line.strip() for line in open(args['labels'])]
print('[INFO]', CLASSES)

# Generate random bounding box colors for each class label
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# Load Caffe model from disk
print("[INFO] Loading model")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# Open video capture from file or capture device
print("[INFO] Starting video stream")
if args['video']:
    cap = cv2.VideoCapture(args['video'])
else:
    cap = cv2.VideoCapture(0)

while cap.isOpened():
    # Capture frame-by-frame
    ret, frame = cap.read()
```

```
    if not ret:
        break

    (h, w) = frame.shape[:2]

    # MobileNet requires fixed dimensions for input image(s)
    # so we have to ensure that it is resized to 300x300 pixels.
    # set a scale factor to image because network the objects has differents
size.
    # We perform a mean subtraction (127.5, 127.5, 127.5) to normalize the
input;
    # after executing this command our "blob" now has the shape:
    # (1, 3, 300, 300)
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 0.007843, (300,
300), 127.5)

    # Pass the blob through the network and obtain the detections and
predictions
    net.setInput(blob)
    detections = net.forward()

    for i in range(detections.shape[2]):
        # Extract the confidence (i.e., probability) associated with the
prediction
        confidence = detections[0, 0, i, 2]

        # Filter out weak detections by ensuring the `confidence` is
        # greater than the minimum confidence
        if confidence > args["confidence"]:
            # Extract the index of the class label from the `detections`,
            # then compute the (x, y)-coordinates of the bounding box for
            # the object
            class_id = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype('int')

            # Draw bounding box for the object
            cv2.rectangle(frame, (startX, startY), (endX, endY),
COLORS[class_id], 2)

            # Draw label and confidence of prediction in frame
            label = "{}: {:.2f}%".format(CLASSES[class_id], confidence * 100)
            print("[INFO] {}".format(label))
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                          COLORS[class_id], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[class_id], 2)
```

```
    # Show fame
    cv2.imshow("Frame", frame)

    key = cv2.waitKey(1) & 0xFF

    # Press `q` to exit
    if key == ord("q"):
        break

# Clean-up
cap.release()
cv2.destroyAllWindows()
```

## FINAL CODE:

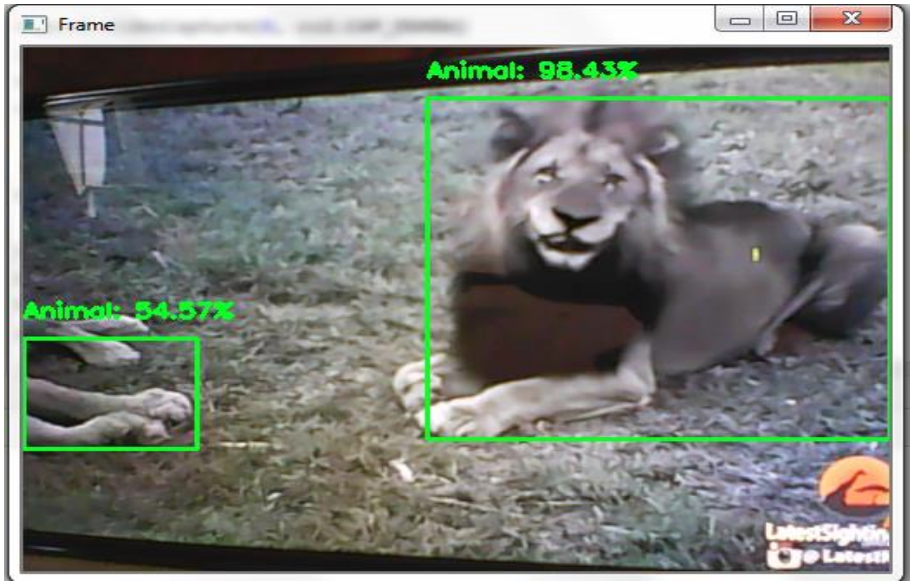
```
#Import packages
import numpy as np
#import argparse
import cv2
import imutils
import time
import os
from collections import Counter
import pygame
import random
#from imutils.video import VideoStream
from imutils.video import FPS

# construct the argument parse and parse the arguments
#ap = argparse.ArgumentParser()
#ap.add_argument("-p", "--prototxt", required=True, help="path to Caffe 'deploy'
prototxt file")
#ap.add_argument("-m", "--model", required=True, help="path to Caffe pre-trained
model")
#ap.add_argument("-c", "--confidence", type=float, default=0.2, help="minimum
probability to filter weak detections")
#args = vars(ap.parse_args())

protext="DNN_Object_Detection-master/DNN_Object_Detection-
master/MobileNetSSD_deploy.prototxt.txt"
model="DNN_Object_Detection-master/DNN_Object_Detection-
master/MobileNetSSD_deploy.caffemodel"

# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
```

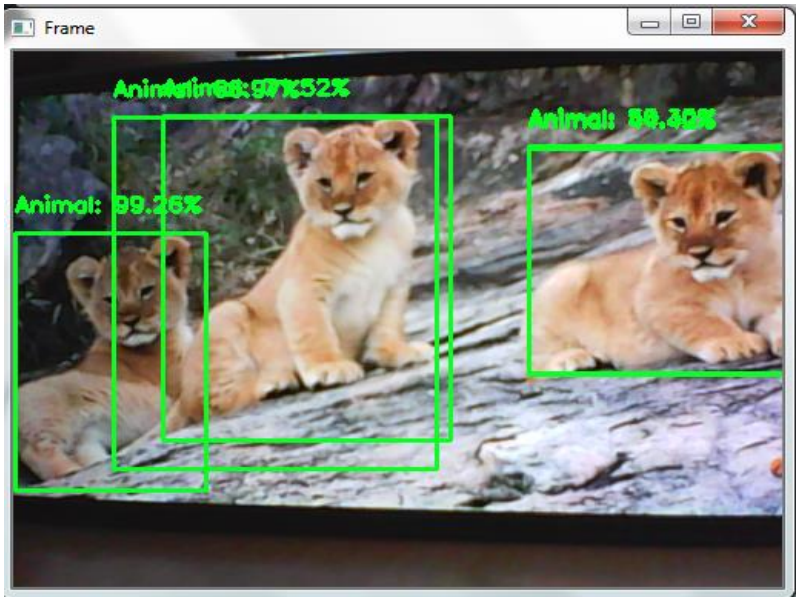
```
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat", "bottle",
"bus", "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike",
"person", "pottedplant", "sheep", "sofa", "train", "tvmonitor"]
REQ_CLASSES=["bird","cat","cow","dog","horse","sheep"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

#Load model
print("Loading Model...")
net=cv2.dnn.readNetFromCaffe(protext, model)
print("Starting camera feed...")

vs=cv2.VideoCapture(0, cv2.CAP_DSHOW)
#vs=VideoStream(src=0).start()
time.sleep(2)
fps=FPS().start()

#Set confidence threshold
conf_thresh=0.2

#Animal detection counter
count=[]
flag=0
siren_loc=os.path.join(os.getcwd(),'Siren.wav')

#Read frame by frame
while vs:
    success, frame = vs.read()
    if not success:
        break
    frame = imutils.resize(frame, width=500)
    #Take the frame dimentions and convert it to a blob
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300,300)), 0.007843,
(300,300), 127.5)

    net.setInput(blob)
    detections = net.forward()
    #frame detection flag
    det=0
    for i in np.arange(0, detections.shape[2]):
        #Probability associated with predictions
        confidence = detections[0,0,i,2]
        if confidence > conf_thresh:
            #Extract the class labels and dimentions of bounding box
            idx = int(detections[0,0,i,1])
            box = detections[0,0,i,3:7]*np.array([w,h,w,h])
            (startX, startY, endX, endY) = box.astype("int")

            #draw the box with labels
```

```
            if CLASSES[idx] in REQ_CLASSES:
                det=1
                label="{}: {:.2f}%".format("Animal",confidence*100)
                cv2.rectangle(frame, (startX, startY), (endX, endY),
(36,255,12), 2)
                if (startY-15) > 15:
                    y = (startY - 15)
                else:
                    y = (startY+15)
                cv2.putText(frame, label, (startX, y), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (36,255,12), 2)

    #Show the frames
    cv2.imshow("Frame", frame)
    count.append(det)
    #Alerts only if at least 15 frames sucessfully detects animals in the last
36 frames (appx 2 sec)

    #if flag==1 and len(count) > c+36:
    #   flag=0
    if flag==1 and len(count) > c +(11*18):
        flag=0
    if Counter(count[len(count)-36:])[1] > 15 and flag==0:
        print(f"Animal Intrusion Alert...!!! {len(count)}")
        path = r"siren/alert"
        file = os.path.join(path, random.choice(os.listdir(path)))
        pygame.mixer.init()
        pygame.mixer.music.load(file)
        pygame.mixer.music.play()
        flag=1
        c=len(count)

    key = cv2.waitKey(1)
    if key == ord("q"):
        break
    fps.update()

fps.stop()
print("Elapsed time: {:.2f}".format(fps.elapsed()))
print("Approximate FPS: {:.2f}".format(fps.fps()))
vs.release()
cv2.destroyAllWindows()
#vs.stop()
```

# CHAPTER 9

## OUTPUT:





The accuracy of animal detection over each dataset for both the MobileNetv2- SSD and tiny-YOLO is provided in Table . Boldface indicates better results. The table also indicates the rate of false positives i.e., where an animal is either mislabelled or bounding box placement is incorrect. The model is trained over the detection dataset, and evaluated over both the re-identification and the detection sets. Although YOLO performs better than SSD for the detection datasets, but the SSD model generalizes better as observed from its higher accuracy over the re-identification set. This could be due to the training parameters used, rather than being a characteristic of the models themselves. Table 4 illustrates the object detection inference times,

and tiny-YOLO is much slower than SSD . The difference is due to use of quantization training of the SSD

TABLE 6.1 EFFICIENT ALGORITHM TO PERFORM THE DATASET

| Dataset | Type | MobileNetv2-SSD | | tiny-YOLO | |
|---|---|---|---|---|---|
| | | Acc. (%) | FPR (%) | Acc. (%) | FPR (%) |
| Tiger | Detection | 80.00 | 4.76 | **89.00** | **0.00** |
| | Re-Id | **99.79** | **0.21** | 85.00 | 0.08 |
| Elephant | Detection / Re-Id | 92.56 | 0.1 | **97.0** | **0.09** |
| Jaguar | Detection | **89.47** | **4.06** | 74.00 | 0.00 |
| | Re-Id | **97.05** | **3.04** | 72.00 | 0.00 |

Table shows the performance of the suggested CNN-based technique. We achieved 97.87% accuracy using five-layer CNN, which is impressive. We tested with a variety of layer counts, but the differences in results were not statistically significant when employing our five-layer CNN model. When we increased the number of layers, we saw an increase in computing time, the complexity of the technique batch size, and steps per second. Furthermore, we set the dropout value to 0.2 but did not calibrate the model since the accuracy flattened. As a consequence, this model has the highest accuracy without the use of dropout.

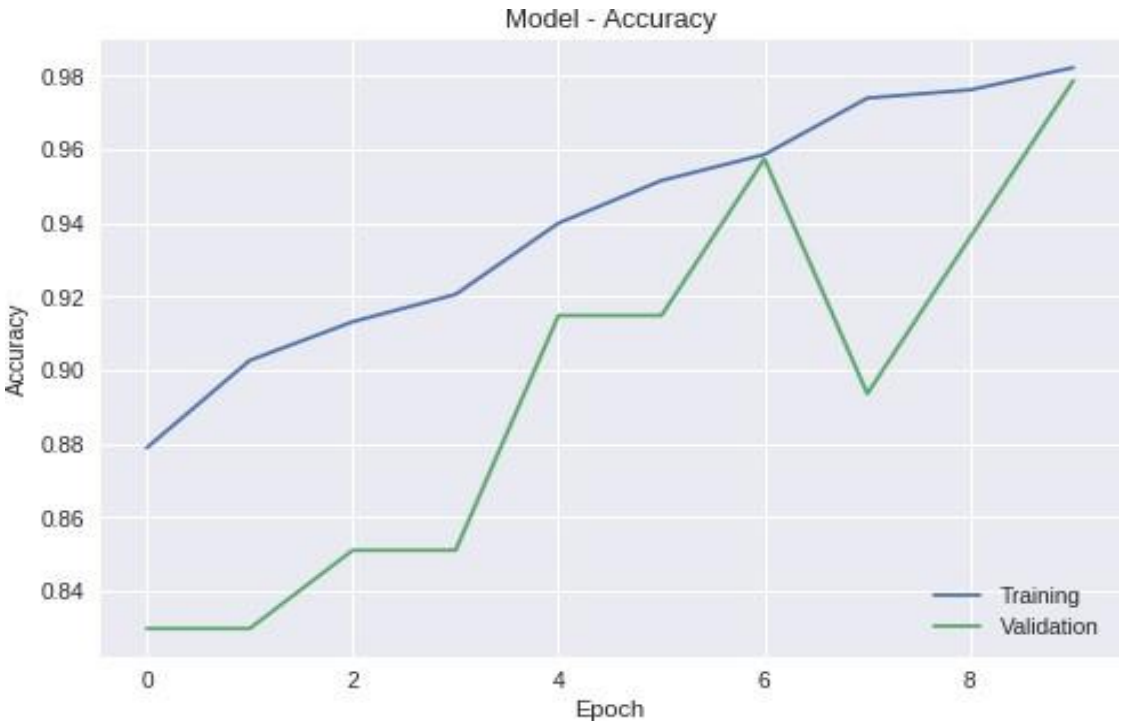



***Fig 6.1. Accuracy of the proposed CNN model***

Figure 6.1 displays our model's training and validation accuracy. Convolution is the mathematical operation which is central to the efficacy of this algorithm. The input to the red region is the image which we want to classify and the output is a set of features. Think of features in an image, for example, an image of a tiger might have features like whiskers, two ears, four legs etc. Convolution in CNN is performed on an input image using a filter or a kernel. The filter slides over the input image one pixel at a time starting from the top left. The filter multiplies its own values with the overlapping values of the image while sliding over it and adds all of them up to output a single value for each overlap.

# CHAPTER 10

## CONCLUSION:

The problem of damaging crops by wild animals has become a major social problem in the current time. It requires urgent attention and an effective solution. The proposed method allows us to detect any animal presence or intrusion in farms using video from any camera device placed in the farms. The object detection model worked almost consistently at 18 frames per second.

It is a cheap and robust system. The siren scares the intruders away as well as it can alert the farmer to act. Thus, this application can be used to protect crops in the farm. It might be very useful for agricultural purposes instead of traditional methods used today.

# CHAPTER 11

## REFERENCES:

[1] Parikh, M. et al. “Wild-Animal Recognition in Agriculture Farms Using W-COHOG for Agro-Security.” (2017).

[2] S. Yadahalli, A. Parmar and A. Deshpande, "Smart Intrusion Detection System for Crop Protection by using Arduino," *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2020, pp. 405-408, doi: 10.1109/ICIRCA48905.2020.9182868.

[3] Howard, Andrew & Zhu, Menglong & Chen, Bo & Kalenichenko, Dmitry & Wang, Weijun & Weyand, Tobias & Andreetto, Marco & Adam, Hartwig. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications."

[4] Deshpande, Abhinav. (2016). Design and Implementation of an Intelligent Security System for Farm Protection from Wild Animals. 5. pp.2319-7064.

[5] M. Gogoi and S.R. Philip, “Protection of Crops from Animals using Intelligent Surveillance System,” Journal of Applied and Fundamental Sciences, vol.1, no.2, pp.200-206, 2015.

[6] S. Pandey and S. B. Bajracharya, “Crop protection and its effectiveness against wildlife: A case study of two villages of shivapuri national park, nepal,” Nepal Journal of Science and Technology, vol. 16, no. 1, pp. 1– 10, 2015.

[7] V. Bavane, A. Raut, S. Sonune, A. Bawane, and P. Jawandhiya, “Protection of crops from wild animals using intelligent surveillance system.”

[8] R. Bhardwaj, K. Bera, O. Jadhav, P. Gaikwad, and T. Gupta, “Intrusion detection through image processing and getting notified via sms and image,” 2018.

[9] R. M. Antunes and F. L. Grilo, “Intruder alarm systems-the road ahead,” in Advanced Technologies. IntechOpen, 2009

10. JAY GE Jr, SOKOLOFF L. Natural history of degenerative joint disease in small laboratory animals. II. Epiphyseal maturation and osteoarthritis of the knee of mice of inbred strains. AMA Arch Pathol. 1956;62(2): 129-35.

11. Staines KA, Poulet B, Wentworth DN, Pitsillides AA. The STR/ort mouse model of spontaneous osteoarthritis—an update. Osteoarthritis Cartilage. 2017;25(6): 802-8.

12. Walton M. Degenerative joint disease in the mouse knee; histological observations. J Pathol. 1977;123(2): 109-22.

13. Brewster M, Lewis EJ, Wilson KL, Greenham AK, Bottomley KM. Ro 32-3555, an orally active collagenase selective inhibitor, prevents structural damage in the STR/ORT mouse model of osteoarthritis. Arthritis Rheum. 1998;41(9): 1639-44.

14. Glasson SS, Chambers MG, Van Den Berg WB, Little CB. The OARSI histopathology initiative—recommendations for histological assessments of osteoarthritis in the mouse. Osteoarthritis Cartilage. 2010;18(Suppl 3):S17-23.

15. Mulrane L, Rexhepaj E, Penney S, Callanan JJ, Gallagher WM. Automated image analysis in histopathology: a valuable tool in medical diagnostics. Expert Rev Mol Diagn. 2008;8(6): 707-25.

16. Sertel O, Dogdas B, Chiu CS, Gurcan MN. Microscopic image analysis for quantitative characterization of muscle fiber type composition. Computer Med Imaging Graph. 2011;35(7-8): 616-28.

17. Ghaznavi F, Evans A, Madabhushi A, Feldman M. Digital imaging in pathology: whole-slide imaging and beyond. Annu Rev Pathol. 2013;8:331-59.

18. Tiulpin A, Thevenot J, Rahtu E, Saarakkala S. A novel method for automatic localization of joint area on knee plain radiographs. In: Sharma P, Bianchi F editors. Image analysis. SCIA 2017. Lecture notes in computer science, vol. 10270. Cham, Switzerland: Springer; 2017. p. 290-301.

19. Liu FY, Chen CC, Cheng CT, Wu CT, Hsu CP, Fu CY, et al. Automatic hip detection in anteroposterior pelvic radiographsA labelless practical framework. J Pers Med. 2021;11:522.

20. von Schacky CE, Sohn JH, Liu F, Ozhinsky E, Jungmann PM, Nardo L, et al. Development and validation of a multitask deep learning model for severity grading of hip osteoarthritis features on radiographs. Radiology. 2020;295(1): 136-45.

21. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, et al. SSD: single shot multibox detector. In: Leibe B, Matas J, Sebe N, Welling M, editors. Computer vision—ECCV 2016. ECCV 2016. Lecture notes in computer science, vol. 9905. Cham, Switzerland: Springer; 2016. p. 21-37.

22. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: ICLR 2015. Available from: https://arxiv.org/abs/1409.1556.

23. Tzutalin D. LabelImg. Available from tzutalin/labelImg. Published Sepember 17, 2015. Accessed June 24, 2021.

24. Cai L, Gao J, Zhao D. A review of the application of deep learning in medical image classification and segmentation. Ann Transl Med. 2020;8(11): 713.

25. Chung SW, Han SS, Lee JW, Oh KS, Kim NR, Yoon JP, et al. Automated detection and classification of the proximal humerus fracture by using deep learning algorithm. Acta Orthop. 2018;89(4): 468-73.

26. Olczak J, Emilson F, Razavian A, Antonsson T, Stark A, Gordon M. Ankle fracture classification using deep learning: automating detailed AO Foundation/Orthopedic Trauma Association (AO/OTA) 2018 malleolar fracture identification reaches a high degree of correct classification. Acta Orthop. 2021;92(1): 102-8.

27. Tiulpin A, Thevenot J, Rahtu E, Lehenkari P, Saarakkala S. Automatic knee osteoarthritis diagnosis from plain radiographs: a deep learning-based approach. Sci Rep. 2018;8:1727.

28. Srinidhi CL, Ciga O, Martel AL. Deep neural network models for computational histopathology: a survey. Med Image Anal. 2021;67:101813.

29.S. Matuska, R. Hudec, P. Kamencay, M. Benco, and M. Zachariasova, “Classification of wild animals based on SVM and local descriptors,” AASRI Procedia, vol. 9, pp. 25–30, 2014.

30. A. Rivas, P. Chamoso, A. González-Briones, and J. Corchado, “Detection of cattle using drones and convolutional neural networks,” Sensors, vol. 18, no. 7, p. 2048, 2018.

.