International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 04 | April - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Animal Detection in Farms Using OpenCV

Lokesh Kumar A , Aravind C , Sabareesh R , Sathiya B

ABSTRACT

A farmer cannot afford the enormous losses in agricultural earnings caused by animal incursion on their land. In the agriculture industry, computer vision is being used more and more to automate operations and increase production. Our proposal is an artificial intelligence system that uses cameras to keep an eye out for animal intrusions in the field. If an animal is spotted, the system may notify the farmer or even act independently. As the name implies, real-time object detection is the skill of identifying different items at that same moment. Identifying objects has always required courage. Faster processing power is needed to identify an item for this reason. The camera is completely utilized in this application. The camera in this application must be configured before we can identify animals. The camera will operate nonstop. The user has total control over this. The picture will be taken by the camera. The video is recorded and split up into several frames using the frame extraction modules. Using the extraction function, every frame from the movie is extracted. Usually, this extraction operates within a set time frame. If an animal is discovered in the video, the animal detection feature finds it and sounds a buzzer. But every system operating in real-time produces unlabelled data, and for effective training, a sizable collection of labeled data is needed. This work presents an application that was constructed using OpenCV libraries to detect certain things, such as animals.

Keywords: Animal detection, Farms, OpenCV, Computer vision, Real-time object detection, Python, MySQL, Intrusion detection, Buzzer alert, Frame extraction, AI-based system, Agricultural productivity.

CHAPTER 1

INTRODUCTION

Humans can comfortably detect and classify objects residing in an image. Complicated tasks such as the identification of numerous objects and the detection of obstacles can be performed by the human visual system with a meager vigilant thought. Moving objects are difficult to track on real-time video sequences, and their detection is a challenging task. This paper deals with designing a system that can detect various objects in a site.

The data gathered here may undoubtedly assist security cameras in transmitting real-time reports on items observed. These days, wildlife monitoring is crucial because many animal species—some of which have already vanished—are in danger of going extinct. Consequently, it is essential to document wild animal activities. Without technology, maintaining these documents becomes an overwhelming chore.

Conventional techniques, like placing camera traps in the forest, taking pictures by hand, and then categorizing and labeling animals based only on the photos, take too much time and energy. Furthermore, it is not feasible for someone to snap images in the forest all day long. New species can also be identified by the technique. A new species' data is kept in a separate database for future research when it is found. If there is internet access nearby, controlling the device from a distance is also feasible. Pretrained categorization models are included in the Python-based code used by the system.

Optimizing agricultural operations is essential for effective and sustainable output in modern agriculture. A crucial element is keeping an eye on and managing animals. Conventional techniques can be labour-intensive and prone to human mistakes, which results in inefficiencies and higher expenses. The incorporation of computer vision technology offers a potential remedy to these problems.

The goal of our project, "Animal Detection in Farms using OpenCV," is to use computer vision to transform farm management. We suggest a system that can automatically identify and keep an eye on animals in agricultural settings by utilising the OpenCV (Open-Source Computer Vision Library) framework.

Our system will continually record live video by using cameras that are strategically positioned across the farm. Next, OpenCV's sophisticated image processing algorithms will be applied to the recorded video stream. These algorithms will make it possible to track and identify different animal species, such as pigs, sheep, and cows.

There are several advantages to putting such a system in place. First of all, it gives farmers priceless insights into the health and behavior of their animals, making it possible to identify any problems like disease or injury early on. Furthermore, our system's data collection may be used for additional analysis and decision-making. We can create predictive models that foresee patterns in animal behavior, optimize feeding schedules, and raise overall farm output by using machine learning approaches.

In conclusion, our research embodies a novel approach to farm management by fusing state-of-the-art computer vision technology with the pragmatic requirements of contemporary agriculture. We want to provide farmers with the necessary skills to prosper in a highly competitive business by optimizing the monitoring process and offering practical insights.

Technological developments have caused a tremendous revolution in the agriculture industry in recent years. Among these developments, computer vision has become a formidable instrument that might completely change agricultural methods. There are many of intriguing prospects to improve production, efficiency, and animal welfare in agricultural systems by integrating OpenCV (Open-Source Computer Vision Library). In this project, we explore the field of farm animal detection with OpenCV to create a reliable automated system for this important part of livestock care.

1.1 BACKGROUND

Our civilization is based on agriculture, which provides billions of people worldwide with a stable source of food and employment. The lack of labor, limited resources, and environmental concerns, however, frequently pose problems for conventional farming practices. Precision agriculture is becoming more and more popular despite these obstacles. It uses data-driven methods to optimize crop production and livestock husbandry.

Technological interventions might have a major positive impact, especially in livestock production. Surveillance of the environment, behavior, and health of livestock is necessary for effective management. These have historically required a lot of labor and been error-prone. The development of computer vision and machine learning, however, has made automated methods for tracking and detecting animals more and more practical.

1.2 SIGNIFICANCE AND POTENTIAL IMPACT:

The successful implementation of an animal detection system in farms using OpenCV holds several significant implications:

1.2.1 Enhanced productivity and efficiency:

Farmers may optimize their operations and more efficiently use their resources by automating the animal detection procedure. Labor costs are decreased and productivity is raised as a result.

1.2.2 Improved animal welfare:

Early identification of anomalies or health problems in cattle allows for timely intervention, which improves the well-being of the animals. Animal suffering can be reduced and illness or injury escalation can be stopped early.

1.2.3 Data-driven decision-making:

Farmers may now get actionable data and insightful information on cattle behavior, movement patterns, and environmental conditions thanks to the integration of computer vision technology. This enables farmers to optimize their management techniques for better results and make well-informed decisions.

1.2.4 Sustainability and resource optimization:

Precision livestock management enhances sustainable agricultural methods by maximizing resource use and reducing waste. Thus, agriculture's environmental impact is lessened and long-term ecological equilibrium is supported.

1.3 SCOPE AND LIMITATIONS:

While the goal of the proposed project is to use OpenCV to construct a complete animal identification system for farms, it is important to recognize some limits and things to keep in mind:

1.3.1 Environmental factors:

Environmental factors including illumination, weather, and topography might affect how well the system performs. When developing a system, robustness to a variety of environmental variables will be crucial.

1.3.2 Species and variability:

There is a lot of variation among farm animals in terms of species, breeds, sizes, and colors. The detection system has to be able to handle this variation and have good cross-species generalization.

1.3.3 Hardware constraints:

Using cameras or other imaging equipment in agricultural settings might present problems with connectivity, power supply, and longevity. For practical applications, designing hardware solutions that are reliable and affordable will be essential.

CHAPTER – II

LITERATURE SURVEY

Sabeenian et al., (2020) Discussed the important problem of wild animal incursions, which may cause problems in a variety of settings, including protected regions, agricultural fields, and populated areas. Because wildlife behavior is complicated and dynamic, traditional intrusion detection techniques may not be able to effectively identify and mitigate such invasions. The authors suggested a unique strategy based on deep learning methods to solve this. The study specifically looked at the automated identification of wild animals using deep neural networks. Convolutional neural networks (CNNs), a kind of deep learning model well-suited for tasks involving images, were used by the authors.

Panda, Prabhat Kumar et al., (2022) Centered on the creation and use of a model for utilizing Internet of Things (IoT) technologies to identify wild animal invasions. To strengthen security protocols in regions vulnerable to wildlife incursions, the study investigated the integration of IoT devices for real-time monitoring and detection of wild animals.

Ravoor et al., (2021) Provided a thorough system for warning communication, tracking, prevention, and detection of animal intrusions. All the parts of this system are carefully examined and optimized to guarantee a successful deployment on embedded devices. As a result, a precisely planned, distributed, reliable, effective, and scalable system is produced. Although re-identification has often required human involvement, this research presents a unique application that makes use of it for following individual animals across many cameras. The proof-of-concept for real-time system operation is provided by the prototype, which is based on Raspberry Pi devices and exhibits a frame rate of 2-3 frames per second. The techniques presented in this research have the potential to create a robust system for detecting animal intrusions in locations that are sensitive to animal attacks.

Sayagavi et al., (2017) Created a scale-out architecture made possible by the separation of duties between edge and central servers, off-site feature extraction, centralized identity association, and autonomous end-point device operation. The deployment of this completely distributed system is simple. To improve system adaptability, the technique may be improved in several ways, for as by adding infrared and wide-angle cameras. The authors proposed using cutting-edge algorithms like LSTM and Recurrent Neural Networks to demonstrate improved in-camera tracking. Tensor Processing Unit (TPU) use can improve object detection performance at the edge server. For broad acceptance, there are still operational issues to be resolved. Alarm noises may cause animals to get used to them, decreasing their usefulness. A further obstacle is the physical security of field-deployed units, which are vulnerable to theft or damage.

Priya Sharma et al., A technique to protect crops from possible animal dangers was introduced. Their creative method makes use of a Raspberry Pi and a PIR sensor. When an animal moves, the PIR sensor detects it, which triggers the camera to turn on and start taking pictures. After being taken, these photos are sent to the Raspberry Pi so that it may analyze and categorize the animal that was found. The system takes the necessary precautions to safeguard the crops based on this classification.

Prajna. P et al., Have created a field observation platform. The system works by first using a sensor to detect any encroachments surrounding the field, and then it turns on a camera to take a picture of the invader. It then uses image processing algorithms to classify the intruder and, depending on the categorization, it initiates appropriate action to deal with the intrusion. Lastly, to alert the farm owner, the system sends them a notification.

Sk. Almas Tabassum et al., Implemented a system meant to shield crops from the dangers caused by animals including elephants, buffaloes, and cows. These creatures can occasionally be seen in agricultural areas, where they can trample and eat crops, causing serious harm. The purpose of the system is to recognize these creatures and

notify the farmer with a snapshot of the animal as soon as it is found. In addition, it notifies the farmer of the intruder's presence by continuously buzzing if the animal is visible to the camera.

Howard et al., The authors provide Mobile-Nets, a class of effective models designed for embedded and mobile vision applications. Deep neural networks that are lightweight are achieved by utilizing depth-wise separable convolutions in the construction of Mobile Nets, an efficient design. Two simples global hyperparameters that successfully balance the trade-off between computational performance and model correctness are also introduced by the authors. With the help of these hyperparameters, model builders may choose a model that is the right size and fits within the limitations of their application. A thorough set of experiments assessing the trade-offs between model accuracy and resource utilization are included in the study. When compared to other popular models in the context of ImageNet classification, the findings show how well Mobile-Nets perform.

Yiting Li et al., Presented this research study with the goal of utilizing deep learning approaches to provide real-time and extremely accurate surface fault identification. The Single Shot Multibook Detector (SSD) design was used in the study to achieve this aim. It was included in the Mobile Net architecture at the bottom convolutional neural network (CNN) level, creating what is known as Mobile Net SSD. The system uses a mix of charge-coupled devices (CCD cameras), digital cameras, and parallel cameras to gather target pictures for the detection process. These cameras are used to take pictures of the target items, identify pertinent aspects, and create mathematical models for examination. To accomplish its goals, Mobile Net SSD mostly uses paired cameras, digital cameras, depth cameras, and CCD cameras.

Anita Chaudhari et al., Presented a CNN (Convolutional Neural Network) based object identification system in her research paper. The purpose of this system is to take pictures of different fruits and vegetables, classify them into different groups, and provide nutritional data. The program takes a picture of your plate of food and utilizes it to make comparisons. It then produces a list of the products that are most likely to be included in your meal. Rather than displaying a single option that includes everything on your plate, it shows every recognized item separately. For example, if there are oranges on your plate, the system's object class recognition component recognizes and provides the oranges' form, colour, texture, and other characteristics.

CHAPTER - III

OBJECTIVES & METHODOLOGY

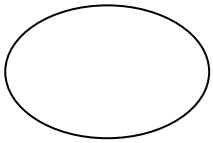
The main aim of the project is to help the farmer to save the crops without harming the animals. The proposed system effectively detects animal entry into the farm area using the Yolo Model. After the success of the animal detection application automatically enables sound alerts based on alerts that animals may run away from the farm such that the crops are prevented from damage. A farmer cannot afford the enormous losses in agricultural earnings caused by animal incursion on their property. The use of computer vision in agriculture is growing as a means of work automation and increased productivity. We suggest an artificial intelligence (AI) system that uses cameras to scan the field for animal incursions and notify the farmer or even do some tasks autonomously. The camera is completely utilized in this application. The camera is always functioning and taking pictures. The video is recorded and split up into several frames using the frame extraction modules. Using the extraction function, every frame from the movie is extracted. Usually, this extraction operates within a set time frame.

Implement the suggested system YOLO is an object identification technique that is used to locate and identify animals in farm photos or video frames in real-time. This procedure is enhanced by OpenCV, which offers a strong foundation for image processing and makes it possible to do operations like feature extraction, picture preprocessing, and camera interface. By combining YOLO with OpenCV, the goal is to develop a dependable and effective system for real-time animal detection that will give farmers insightful information about livestock behaviors and enable timely decision-making. The suggested method addresses issues with animal monitoring in agricultural areas while improving farm management applications and resulting in higher output.

3.1.1 Data Flow Diagram

The data flow diagram shows the interactions between data and the system. Data flow diagrams model a system by using external entities from which data flow a process that transmits the data and creates output data which goes to other processes on external entities of files. They are very helpful in modeling many aspects of a business function because they methodically divide a task into basic parts, helping the analyst understand the system that they are trying to model. Additionally, data may enter a process as input.

The symbols appearing in the Data Flow Diagram have been explained below:



- Represents a process

- Which shows data flow

- Designation of the data

- Shows Data source

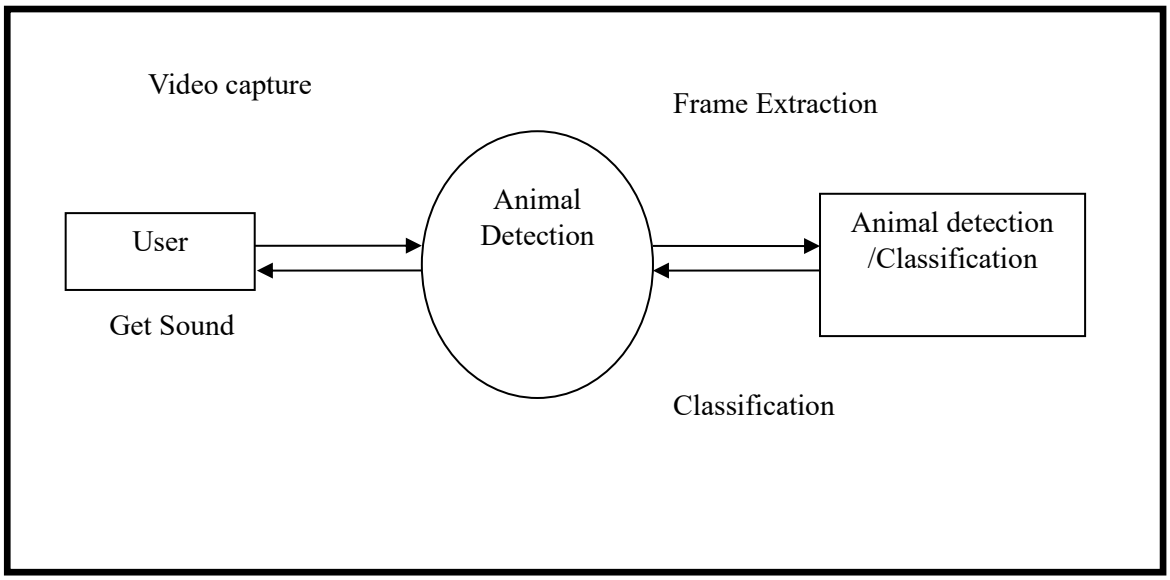


Fig 3.1 Level-1

3.1.2 Enhancing Farm Monitoring Efficiency:

Conventional farm animal monitoring techniques frequently depend on limiting sensor-based devices or manual observation. Through the introduction of a computer vision-based method that can reliably and constantly identify and categorize different kinds of farm animals, this research aims to transform farm surveillance.

3.1.3 Improving Animal Welfare:

The technology helps raise the bar for animal care on farms by automating the process of animal detection and monitoring. Early identification of problems like animal discomfort, wounds, or behavioral abnormalities can result in timely treatment and action, protecting farm animals' well-being.

3.1.4 Optimizing Resource Allocation:

Optimal resource allocation, including feed, water, and medical care, is necessary for effective farm management. Farmers will be able to make data-driven decisions on resource allocation and utilization thanks to the created system's insights into animal presence, movement patterns, and population dynamics.

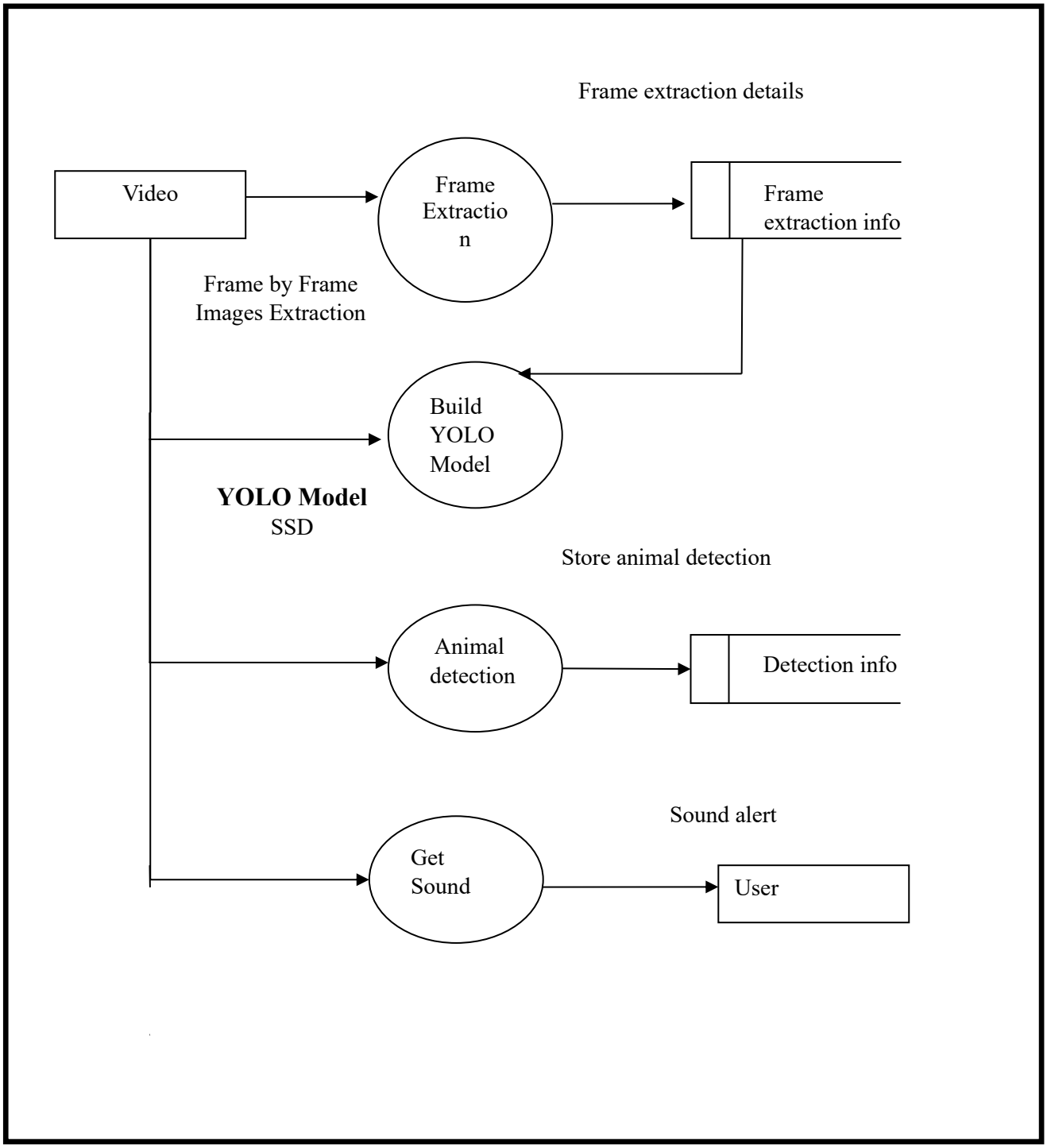


Fig 3.2 Level-2

3.1.5 Facilitating Precision Livestock Farming:

The goal of precision livestock farming is to increase agricultural output while reducing resource waste and its negative effects on the environment. This research advances the precision farming paradigm by enabling targeted interventions and management methods based on real-time animal data through the integration of powerful computer vision techniques with farm monitoring systems.

3.1.6 Enabling Scalable and Sustainable Solutions:

Due to its scalability, the suggested system may be used on farms of all shapes and sizes, ranging from huge commercial operations to tiny family farms. Reduced physical labour, resource optimization, and the advancement of technical innovation to support productive farming techniques are the three main components of sustainability.

3.1.7 Empowering Farm Operators:

Providing farm managers and operators with an effective tool that improves their capacity to make decisions, eases their workload, and increases farm operations' overall productivity, profitability, and sustainability is the aim.

3.2 METHODOLOGY:

YOLO Model:

Train the YOLO object identification model using the annotated dataset. Set up the network architecture, number of classes, anchor boxes, and other YOLO characteristics. Using the annotated data for supervised learning, train the model to correctly identify animals. Utilizing the pre-processed and annotated dataset, train the YOLO model. Optimize the model using the dataset unique to the farm to increase detection precision. The dataset was gathered from online resources such as Kaggle.

The following section describes the algorithm steps in YOLO.

Step 1. Image Input: The YOLO algorithm takes an input image, denoted as I, where I represents the image captured from the video V.

Step 2. Grid Division: The input image is divided into a grid of cells. Each cell is represented by its grid coordinates (i,j) where i and j are the row and column indices, respectively.

Step 3. Anchor Boxes: in the proposed animal detection, the YOLO uses anchor boxes, denoted as B, representing pre-defined bounding box shapes. The anchor boxes are defined as $B=\{(w1,h1),(w2,h2),\ldots\}$, where w and h are width and height.

Step 4. Bounding Box Prediction: For each grid cell (i,j), it predicts multiple bounding boxes, denoted as Bij, with their coordinates represented as(xij,yij,wij,hij). Each bounding box has an associated confidence score Cij, indicating the probability of an object being present.

Step 5. Object Confidence Score: The object confidence score, denoted as Cij, represents the likelihood that a bounding box contains an object. It is a value between 0 and 1.

Step 6. Class Probability: proposed algorithm predicts the class probabilities for each bounding box. Let Pij(Ck) be the probability of class Ck for the bounding box Bij.

Step 7. Non-Maximum Suppression: After predictions, YOLO applies non-maximum suppression to select the bounding box with the highest confidence score for each object. This is denoted as NMS(Bij).

Step 8. Output: The final output consists of a set of selected bounding boxes Bij, each associated with a class label and a confidence score.

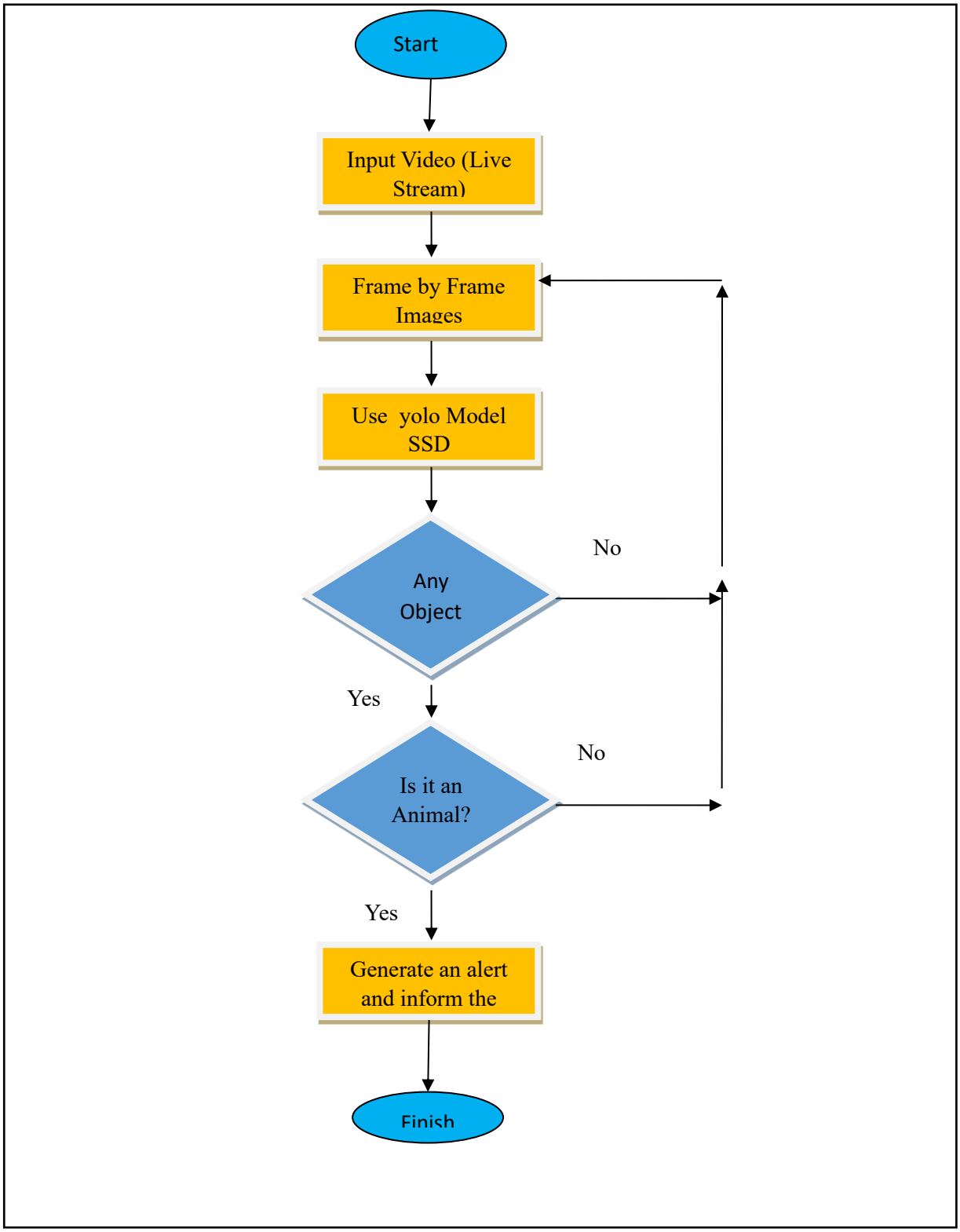


Fig 3.2.1 Proposed Methodology (Flow chart)

3.2.1 Data Collection and Preprocessing:

The process begins with gathering a representative and varied collection of photos of farm animals. Several species that are frequently seen on farms, including sheep, goats, pigs, chickens, and cows, will be included in this dataset. There are several sources from which the data-collecting procedure might be accomplished:

Publicly Available Datasets: Make use of pre-existing datasets, such as the COCO (Common items in Context) collection, which includes annotations for photos of various items, including animals.

Farm Surveillance Cameras: Work with agricultural organizations or farm owners to gain access to recorded or live footage from these cameras.

Custom Data Collection: Use cameras or drones to take pictures, especially for this project, making sure to get a variety of backdrops, lighting setups, and stances.

Preprocessing procedures are essential to guarantee data quality and applicability for training machine learning models after it has been gathered. Among the preprocessing methods are:

Image Resizing: To make model training and inference easier, standardize the image dimensions to a common size, such as 224x224 pixels.

Data augmentation: To improve variability and minimize overfitting, apply changes like rotation, scaling, and flipping to the dataset.

Normalization: To enhance model convergence during training, normalize pixel values to a common scale (such as 0 to 1).

3.2.2 Object Detection using OpenCV:

The core of the animal identification system is object detection, a basic problem in computer vision. A variety of object detection algorithms and methods are available in OpenCV, such as:

Haar Cascade Classifiers: To recognize faces, bodies, or certain patterns in animals, use pre-trained Haar Cascade models.

YOLO (You Only Look Once): Use the deep learning-based YOLOv3 or YOLOv4 object detection models, which are renowned for their quickness and precision in real-time applications.

Faster R-CNN (Region-based Convolutional Neural Networks): To recognize animals with high precision and localization accuracy, either start with a blank and train a Faster R-CNN model, or utilize pre-trained weights.

Object identification entails the following steps:

Model Selection: To choose which object detection model is best for agricultural situations, compare several models and consider criteria like speed, accuracy, and resource

Training and Fine-Tuning: Use the annotated farm animal dataset to train from scratch or refine already-trained models. Use strategies such as transfer learning to make use of your understanding of generic object detection assignments.

Model Optimisation: Prepare the selected model for deployment on resource-constrained devices, including embedded systems or edge computing devices, that are frequently seen in farm settings.

3.2.3 Animal Classification:

Classifying animals into distinct groups, such as cows, pigs, chickens, etc., comes next once they have been identified in farm photos. This entails applying deep learning methods to create a reliable animal categorization model:

Convolutional neural networks, or CNNs: Create and hone CNN designs specifically for categorization problems involving animals. When evaluating architectures for image recognition problems, consider models like VGG, ResNet, or EfficientNet.

Transfer Learning: To speed up training and increase classification accuracy, use pre-trained CNN models (such as ImageNet) and refine them on the dataset of farm animals.

Data Balancing: To guarantee that all animal groups are equally represented during training, address class imbalance concerns by using approaches like oversampling, under-sampling, or class weighting.

The pipeline for categorization consists of:

Data splitting: To train, validate, and assess the performance of the model, separate the annotated dataset into test, training, and validation sets.
Model Training: Using the training set, optimize the hyperparameters and keep an eye on metrics like accuracy, loss, and validation performance while you train the animal classification model.

Model Evaluation: Using the validation set, evaluate the trained model to see how well it generalizes and make any required adjustments to the model's parameters.
Metrics of Performance: Determine classification metrics to measure the model's effectiveness on several animal categories, such as accuracy, precision, recall, F1 score, and confusion matrix.

3.2.4 Integration with Farm Monitoring Systems:

The method for detecting and classifying animals that have been created will either be used as a stand-alone solution or incorporated into the current farm monitoring systems. Integration involves a few crucial elements:

API Development: Create and put into use APIs (Application Programming Interfaces) to easily integrate with centralized monitoring systems, IoT devices, and farm management software.

Real-time Processing: To enable prompt identification and reaction to animal-related incidents, optimize the system for real-time processing of farm surveillance footage. Investigate edge computing options to install the

system on farm premises' edge devices, lowering latency and bandwidth needs.

User Interface: Provide an intuitive user interface, complete with warnings, reporting features, and visualizations, so that farm operators and managers can easily access and engage with the animal detection system.

3.2.5 Evaluation and Performance Analysis:

There will be thorough testing and performance analysis done to assess the produced system's efficacy:

Testing in the real world: To evaluate the system's resilience and dependability, install it in real farm surroundings with varied lighting, weather, and animal behaviors.

Metrics of Performance: To measure the system's performance, have a look at important performance measures including processing speed, resource utilization, detection accuracy, and classification accuracy.

Comparative Analysis: To demonstrate the benefits and enhancements of the system, and measure its performance against industry standards or baseline techniques.

User feedback: To confirm the system's usefulness, influence on farm management, and usability, get input from farmers, veterinarians, and agricultural specialists.

3.2.6 Documentation and Reporting:

All phases of the development process, including data gathering, model training, integration procedures, performance assessment, and user input, should be documented. Make an in-depth project report that emphasizes:

Project Goals and Scope: Give a thorough explanation of the project's aims, targets, and intended results.

Technique Overview: Give a summary of the technical strategies, methods, and tools used in the technique.

Results and Analysis: Summarise the findings from user feedback analysis, comparison analysis, performance measurements, and system testing.

Suggestions and Upcoming Work: Provide suggestions for system upgrades, scalability improvements, and possible future paths for farm animal identification and management research.

3.3 TYPES OF TESTING:

Regarding your project on "Animal Detection in Farms using OpenCV," there are several testing methods that may be used to assess the functionality and performance of the system that has been constructed. The following are some crucial test kinds that apply to our project:

3.3.1 Unit Testing:

Unit testing is a software testing technique used in computer programming that tests individual source code units, sets of one or more computer program modules along with related control data, use instructions, and operating procedures, to see if they are suitable for use. Every bit of code in this chatbot is tested, and each module's accuracy is guaranteed.

3.3.2 Integration Testing:

Software testing's integration testing phase, often known as integration and testing, or I&T for short, is when separate software components are put together and tested together. It takes place before validation testing but following unit testing. Unit-tested modules provide the input for integration testing, which then collects them into bigger aggregates, performs tests specified in an integration test plan to those aggregates, and outputs an integrated system that is prepared for system testing.

3.3.3 Functional Testing:

Functional testing is a kind of black-box testing that is part of the quality assurance (QA) process. It builds its test cases around the requirements of the software component that is being tested. The internal program structure is rarely considered while testing functions; instead, input is fed into the function and the result is examined (unlike white-box testing). Typically, functional testing explains the functionality of the system. It is not implied by functional testing that you are testing a module or class function (method). A portion of the system's overall functioning is tested through functional testing.
Functional testing has many types:

- Smoke testing
- Sanity testing
- Regression testing
- Usability testing

3.3.4 White Box Testing:

Testing a software solution's internal coding and architecture is known as "white box" testing. Its main objectives are to enhance security, optimize input and output flow inside the program, and enhance design and usability. Clear box testing, Open box testing, structural testing, transparent box testing, code-based testing, and glass box testing are other names for white box testing. It is one of two components of the software testing "box testing" methodology. Black box testing, its opposite, is testing from an external or end-user perspective. Conversely, White Box testing centers around internal testing and is based on an application's internal workings. Because of the see-through box concept, the phrase "white box" was adopted. The ability to look through the software's exterior (or "box") and into its internal workings is symbolized by the term "clear box" or "white box." Similarly, "black box"

in "black box testing" refers to the fact that only the end-user experience can be assessed since the inner workings of the program are hidden from view.

3.3.5 BlackBox Testing:

Software testing known as "black box" testing involves verifying the functionality of the software under test (SUT) without examining the underlying code structure, implementation specifics, or internal program routes. The software specifications and requirements serve as the sole foundation for this kind of testing.

CHAPTER – IV

PROPOSED WORK MODULES

Using the previously described approach for the project "Animal Detection in Farms using OpenCV," the following work modules are suggested, which may be organized to make our project's execution easier:

4.1 Camera Configuration and Capture Process:

The camera is completely utilized in this application. The camera in this application must be configured before we can identify animals. The camera will operate nonstop. The user has total control over this. The picture will be taken by the camera.

4.1.1 Video Capturing:

The first module handles setting up the video and carrying out the capturing procedure. Using the webcam, the video-capturing module records video.

4.1.2 Frame Extraction:

The video is recorded and split up into several frames using the frame extraction modules. Using the extraction function, every frame from the movie is extracted. Usually, this extraction operates within a set time frame.

4.1.3 Detect Object Boundary Boxes:

Boundary boxes are typically used to indicate the spatial position of animal images. Given the coordinates of the upper-left corner of the rectangle and the lower-right corner, the bounding box has a rectangular shape. The width and height of the box, together with the (x,y)-axis coordinates of the bounding box center, are two more often used bounding box representations.

Architecture Diagram:

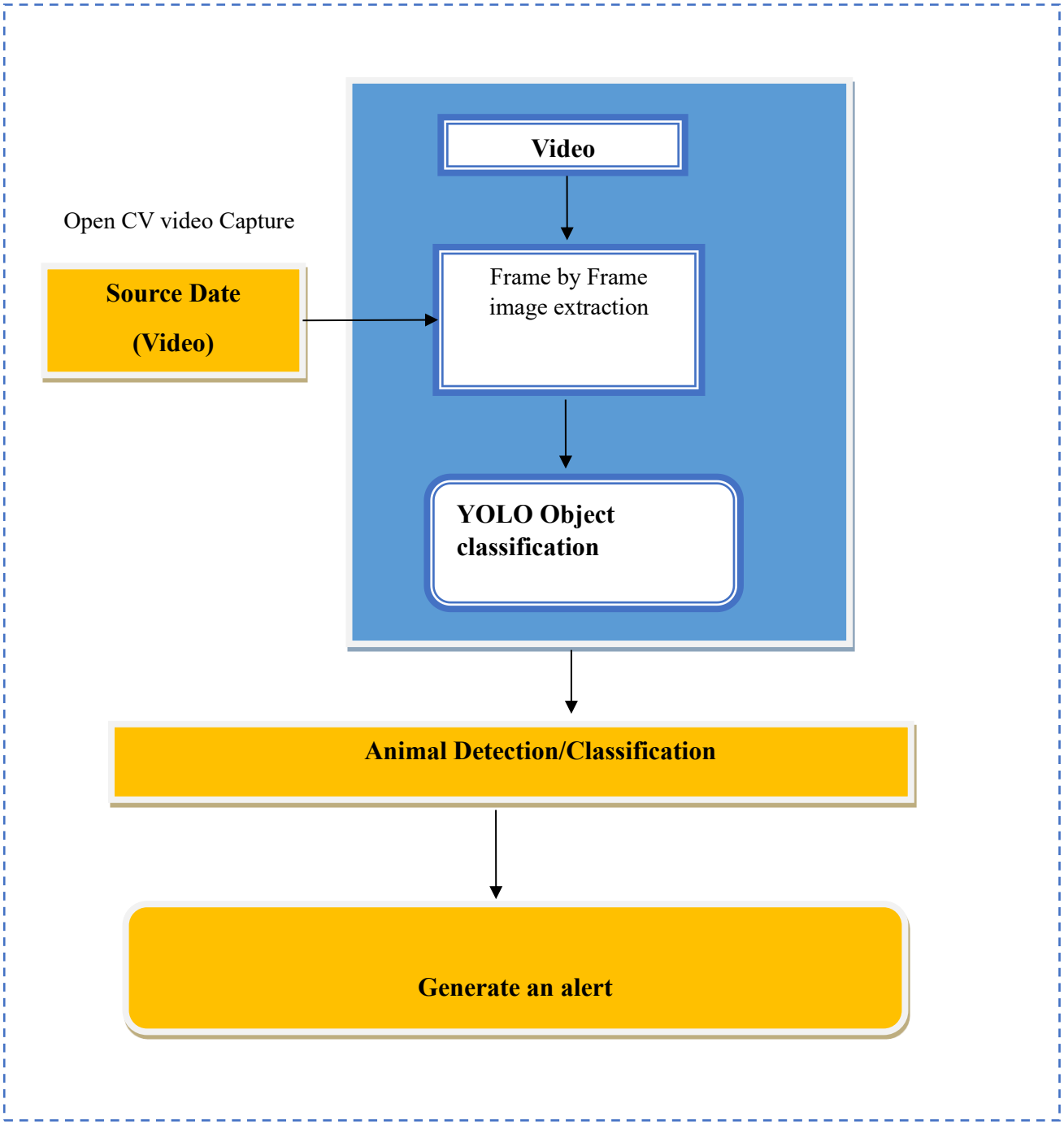


Fig 4.1 Model of the Architecture

4.1.4 YOLO Animal Classification:

Yolo Object detection is the capacity to accurately recognize or detect objects in any given picture. To put it simply, Yolo Image Classification determines the class that an item belongs to. Ultimately, the animal information from the captured image is classified using this module.

4.1.5 Sound Alert:

These modules will assist in providing audible alarm notifications following YOLO Animal Classification. The easiest module to employ to play sound is the play sound module. It has just one method, play sound (), and one parameter to get the name of the audio file to play.

CHAPTER - V

RESULTS & DISCUSSION

Implementation is the stage of the project when the theoretical design is turned into a working system. Thus, it can be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The web application developed in this project proves that the detection of objects done using the desired training model gives a decent measure of accuracy. The power of this technique is that it is often trained any sort of object to be identified in several circumstances. An extension to the present work would be to set up a strong microprocessor and install a high end. This system is intended for the identification of animals and this program is tested over many data set images.

5.1 OUTPUT:

Fig 5.1 Capture Image

5.1.1 Sample Coding:

```
import cv2
import time
import os
import pyttsx3
engine = pyttsx3.init()
engine.say("Welcome To all")
engine.runAndWait()
classNames = []
classFile = "coco.names"
with open(classFile,"rt") as f:
    classNames = f.read().rstrip("\n").split("\n")
configPath = "ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt"
weightsPath = "frozen_inference_graph.pb"
net = cv2.dnn_DetectionModel(weightsPath,configPath)
net.setInputSize(320,320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
def getObjects(img, thres, nms, draw=True, objects=[]):
    classIds,confs,bbox=net.detect(img,confThreshold=thres,nmsThreshold=nms)
    #print(classIds,bbox)
    if len(objects) == 0: objects = classNames
    objectInfo =[]
    if len(classIds) != 0:
        for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
```

```
            className = classNames[classId - 1]
            if className in objects:
                objectInfo.append([box,className])
                if (draw):
                    cv2.rectangle(img,box,color=(0,255,0),thickness=2)
                    cv2.putText(img,classNames[classId-1].upper(),(box[0]+10,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
cv2.putText(img,str(round(confidence*100,2)),(box[0]+200,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
                    print(className)
                    if className=='elephant':
                        print("hai")
                        engine.say("elephant detected")
                        engine.runAndWait()
    return img,objectInfo
if __name__ == "__main__":
    cap = cv2.VideoCapture(0)
    cap.set(3,640)
    cap.set(4,480)
    #cap.set(10,70)
    while True:
        success, img = cap.read()
        result, objectInfo = getObjects(img,0.45,0.2,objects=['elephant'])
        #print(objectInfo)
        cv2.imshow("Output",img)
```

```
        cv2.waitKey(1)
import cv2
import time
import os
import pyttsx3
engine = pyttsx3.init()
engine.say("Welcome To all")
engine.runAndWait()
classNames = []
classFile = "coco.names"
with open(classFile,"rt") as f:
    classNames = f.read().rstrip("\n").split("\n")
configPath = "ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt"
weightsPath = "frozen_inference_graph.pb"

net = cv2.dnn_DetectionModel(weightsPath,configPath)
net.setInputSize(320,320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
def getObjects(img, thres, nms, draw=True, objects=[]):
    classIds,confs,bbox=et.detect(img,confThreshold=thres,nmsThreshold=nms)
    #print(classIds,bbox)
    if len(objects) == 0: objects = classNames
    objectInfo =[]
```

```
    if len(classIds) != 0:
        for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
            className = classNames[classId - 1]
            if className in objects:
                objectInfo.append([box,className])
                if (draw):
                    cv2.rectangle(img,box,color=(0,255,0),thickness=2)
                    cv2.putText(img,classNames[classId-1].upper(),(box[0]+10,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
cv2.putText(img,str(round(confidence*100,2)),(box[0]+200,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
                    print(className)
                    if className=='elephant':
                        print("hai")
                        engine.say("elephant detected")
                        engine.runAndWait()
    return img,objectInfo
if __name__ == "__main__":
    cap = cv2.VideoCapture(0)
    cap.set(3,640)
    cap.set(4,480)
    #cap.set(10,70)
    while True:
        success, img = cap.read()
        result, objectInfo = getObjects(img,0.45,0.2,objects=['elephant'])
```

```
#print(objectInfo)

cv2.imshow("Output",img)

cv2.waitKey(1)
```

5.2 SIGNIFICANCE:

The project "Animal Detection in Farms using OpenCV" is a noteworthy development in the field of agriculture technology. The project seeks to construct an automated system for farm animal detection and monitoring using OpenCV and computer vision techniques. This has a great deal of real-world applications, such as raising agricultural output, protecting animal welfare, and managing animals better. A system like this can also help with manual labour reduction, resource allocation optimization, and giving farmers real-time insights. To address major issues in contemporary farming, the project title emphasizes how cutting-edge technology and agricultural techniques may be combined.

5.3 STRENGTHS:

First, it makes the project's focus—namely, the application of OpenCV technology for animal detection in agricultural environments—clearly defined. Because it establishes boundaries and expectations for the project's scope and results, this specificity is beneficial. Second, the addition of "Animal Detection" emphasizes how computer vision may be used practically to solve problems in the real world, such as keeping an eye on animals or cattle in agricultural settings. This feature highlights the project's potential influence on animal welfare and agricultural methods while also giving it more relevance and significance.

Furthermore, the title's reference to "Farms" highlights the target setting for the detection system and highlights how it may be useful in improving farm production, security, and management. The project's relevance and application to stakeholders in the agriculture industry are increased by this concentration on a particular topic. Furthermore, the title's usage of OpenCV, a popular open-source computer vision package, points to a theoretically sound and approachable method of putting the animal identification system into practice. The project's technological expertise is demonstrated by this option, which also suggests the project's potential for scalability and integration with current frameworks or systems.

5.4 LIMITATIONS OF THE PROPOSED WORK:

The "Animal Detection in Farms using OpenCV" project has several advantages, but it also has a lot of drawbacks. Its merits are found in its well-defined scope, which concentrates on applying OpenCV technology to the task of animal identification in agricultural environments. This precision not only establishes precise standards

but also highlights the usefulness of computer vision in solving real-world problems in agriculture. Furthermore, the project's connection to the agricultural industry is highlighted by the inclusion of "Farms" in the title, which also suggests the project's possible effects on farm production, security, and management. Furthermore, using OpenCV as the technological platform indicates a sound and user-friendly strategy that can help with integration and scalability with current farm systems.

But there are also some significant obstacles to the initiative. Animal species diversity, occlusions, and different lighting conditions can all reduce the accuracy of animal detection algorithms and need a great deal of training and fine-tuning. Moreover, the computing requirements for real-time detection on farms present difficulties in hardware resources, processing speed, and energy economy, particularly in large-scale operations. It takes much thought and testing to generalize the detection system across various farm types and environmental circumstances while guaranteeing compatibility with current farm procedures. Concerns about data protection and user acceptability might also surface during connection with agricultural management systems.

CHAPTER-VI

CONCLUSION

In conclusion, the "Animal Detection in Farms using OpenCV" study offers a great chance to improve animal management and monitoring procedures in farming environments. The project's goal is to create an improved animal identification system that can precisely identify and track animals in agricultural areas by using the power of computer vision and contemporary technologies like OpenCV and deep learning. The project's implementation phase is when the theoretical design is transformed into a functional system. As a result, it might be said to be the most important phase in creating a new system that is successful and instilling trust in the user regarding its functionality. The web application created for this project demonstrates that object detection carried out using the intended training model yields a respectable level of accuracy. This technique's strength is in its ability to identify any kind of item under a variety of conditions. It is often trained. A further development of the current effort would be to install high-end hardware and set up a powerful CPU. This program has been tested on many data set photos with the aim of identifying animals.

In summary, the "Animal Detection in Farms using OpenCV" project is a major step towards utilizing cutting-edge technology to modernize farm management techniques. The project seeks to provide farmers with actionable information through the delivery of a reliable, scalable, and user-friendly detection system, thereby promoting sustainable and ethical farming methods.

6.1 FURTHER SCOPE OF THE PROJECT:

Each application offers benefits and drawbacks of its own. Nearly all the requirements have been met by the project. Because the code is mostly organized or modular in nature, it is easy to add more needs and make enhancements. Append enhancements might involve introducing new modules or altering the current ones. The program may be improved further by using a real-time sensor for animal detection.

6.2 FUTURE WORK:

Multi-Species Detection and Tracking: Increase the system's capacity to concurrently identify and monitor a variety of species, such as pests, wildlife, and household pets. Identify target species and non-target things by developing algorithms.

IoT Integration for Sensor Data Fusion: Connect Internet of Things (IoT) devices to the animal detection system, including GPS trackers, humidity sensors, and temperature sensors. To improve situational awareness and decision-making in farm management, use sensor data fusion techniques.

Data Anonymization and Privacy Preservation: To preserve privacy and abide by data standards, look at ways to anonymize agricultural surveillance data. To safeguard sensitive data, use access control methods, data masking, and encryption.