# 1. INTRODUCTION

Effective wildlife monitoring is crucial for conservation efforts, yet traditional methods often fall short in providing timely and comprehensive data, particularly in vast natural habitats. Periodic surveys and camera traps, while valuable, can be labor intensive, costly, and lack real-time capabilities. This poses significant challenges for forest officers and conservationists in promptly addressing threats to biodiversity and ecosystems.

To overcome these limitations, this paper proposes an innovative approach: an intelligent wildlife detection and alert system that combines deep learning techniques with instant messaging technology. By leveraging the advancements in object detection algorithms, particularly YOLOv5, our system aims to automatically identify and classify wildlife species in real-time.

The integration of YOLOv5 enables our system to operate autonomously, eliminating the need for manual surveillance and significantly reducing resource requirements and operational costs. Moreover, by leveraging edge computing capabilities, the system facilitates on-site analysis, ensuring real-time responsiveness without relying on centralized processing units.

One of the key features of our system is its integration with Telegram, a widely used instant messaging platform. By transmitting real-time alerts to forest officers and stakeholders, including precise location data and identified animal species, the system enables timely intervention in case of potential threats or illegal activities.

This paper presents a comprehensive overview of our proposed intelligent wildlife detection and alert system, highlighting its technological framework, operational advantages, and potential impact on conservation efforts. Through the seamless integration of deep learning and instant messaging technologies, our system aims to revolutionize wildlife monitoring, facilitating proactive conservation measures and safeguarding biodiversity in natural habitats.

# 2. LITERATURE SURVEY

## 2.1 EXISTING SYSTEM

The author Zhang et al. proposed wild animal detection using a multi-level graph cut approach for investigating spatial details and a cross-frame temporal patch verification technique for temporal details. The model analyzes the foreground and background details of the camera trap videos. This approach uses a Camera trap and Change Detection net dataset for segmenting the animal object from natural scenes based on cluttered background videos. Although the model produces a high detection rate, fails to perform well in detecting crucial details like location details, and human interruptions. The author proposed animal detection using Convolutional Neural Network (CNN), and the author proposed animal detection using Iterative Embedded Graph Cut (IEGC) techniques to form regions over images and DeepCNN features and machine learning classification algorithms for classification purposes. Although these models verify the extracted patches are background or animal, still need improvements in classification performance.

Object Detection using deep learning methods attained new heights in computer vision applications. The detection of objects present in images or videos by using object localization and classification techniques gives higher support in detecting various objects present in an image or video. From the extracted results, we can count the number of objects and their activity. This technique is highly used in video surveillance and security-based applications, tracking objects in hidden boxes, monitoring fraudulent activity in public and crowded areas, traffic monitoring and identification of vehicle theft, vehicle number plate recognition, and Object Character Recognition (OCR) .

This paper aims to identify the movements of animals around forest space, provides alert information to the forest officers in case of hunting, crossing the forest lines, any hindrance to villagers and tourists people, and detection of trespassing activity. The development of various methods for employing object detection in different environments and diverse applications shows the progress and importance of object detection in research fields and gained more attention. Moreover, further research works in this area provide useful insights into numerous applications and construct powerful frameworks for detecting objects in different scenarios. The Fast R-CNN techniques are widely used for object detection due to their high accuracy and improved training performance. The introduction of the Faster R-CNN technique rapidly improves the detection performance of the model by employing full image-based convolution features and region-based networks. The Histogram of Oriented Gradients (HOG) feature descriptors uses the Region of Interest (ROI) techniques to identify the objects faster than earlier approaches. The conventional R-CNN technique introduces efficient detection methods by incorporating region proposal networks and ConvNet. This method detects the thousands of object classes in an image or video using annotated information. The R-CNN techniques do not use any approximation techniques and hashing methods for predicting the object regions. R-

FCN techniques use weighted full convolution layers to detect object’s region and finds ROI to detect the category of objects and its background details.Object detection techniques also sounds good with the help of deep learning techniques in the field of autonomous vehicles and traffic scene object detection also.

The Single Short Detector (SSD) methodology uses bounding boxes based discretization techniques to effectively handle feature map information and large volume data. The Spatial Pyramid Pooling (SPP-net) computes the feature maps in single computations and provides high robustness to the object detection tasks using sub-region-based fixed length representations. The You Only Look Once (YOLO) architecture achieves faster results by processing 155 frames per second in real-time cases. This technique uses an end to end approach to detect the objects using regression and probabilistic computations instead of considering classification approaches and produces remarkable results in object detection with a lower falsepositive rate. The detailed investigation is done by the researchers with respect to background subtraction and elimination. The authors used different approaches to detect the background details such as estimating multiple hypotheses, non-parametric model , and global statistic- based methods , background cut .

## 2.2 LIMITATIONS ON EXISTING SYSTEM

Certainly, here are some limitations of the existing systems for wild animal activity detection:

Limited Accuracy: Existing systems often suffer from limited accuracy in detecting and classifying wild animal activity, particularly in complex and cluttered environments such as forested areas. This can lead to false alarms or missed detections, compromising the effectiveness of the overall monitoring system.

High False Positive Rates: Many existing systems exhibit high false positive rates, resulting in unnecessary alerts and increasing the workload for monitoring personnel. False alarms can diminish trust in the system and lead to complacency among users.

Slow Response Times: Due to the lack of real-time alert mechanisms, existing systems may experience delays in notifying authorities of detected animal activity. Slow response times can hinder timely intervention and increase the risk of human-wildlife conflicts.

Limited Scalability: Some existing systems may struggle to scale effectively to cover large geographical areas or accommodate a growing volume of data. This limitation can restrict the system's applicability in expansive or densely populated regions.

## 2.3 LITERATURE SURVEY

**1. Connected segmentation tree—A joint representation of region layout and hierarchy**

Authors:N. Ahuja and S. Todorovic

This paper proposes a new object representation, called connected segmentation tree (CST), which captures canonical characteristics of the object in terms of the photometric, geometric, and spatial adjacency and containment properties of its constituent image regions. CST is obtained by augmenting the object silas segmentation tree (ST) with inter-region neighbor links, in addition to their recursive embedding structure already present in ST. This makes CST a hierarchy of region adjacency graphs. A region's neighbors are computed using an extension to regions of the Voronoi diagram for point patterns. Unsupervised learning of the CST model of a category is formulated as matching the CST graph representations of unlabeled training images, and fusing their maximally matching subgraphs. A new learning algorithm is proposed that optimizes the model structure by simultaneously searching for both the most salient nodes (regions) and the most salient edges (containment and neighbor relationships of regions) across the image graphs. Matching of the category model to the CST of a new image results in simultaneous detection, segmentation and recognition of all occurrences of the category, and a semantic explanation of these results.

**2 . Change detection with weightless neural networks**

Authors:M. De Gregorio and M. Giordano

In this paper a pixel -- based Weightless Neural Network (WNN) method to face the problem of change detection in the field of view of a camera is proposed. The main features of the proposed method are 1) the dynamic adaptability to background change due to the WNN model adopted and 2) the introduction of pixel color histories to improve system behavior in videos characterized by (des)appearing of objects in video scene and/or sudden changes in lightning and background brightness and shape. The WNN approach is very simple and straightforward, and it gives high rank results in competition with other approaches applied to the ChangeDetection.net 2014 benchmark dataset.

**3. Development of an end-to end deep learning framework for sign language recognition, translation,and video generation**

Authors:B. Natarajan, E. Rajalakshmi, R. Elakkiya,

The recent developments in deep learning techniques evolved to new heights in various domains and applications. The recognition, translation, and video generation of Sign Language (SL) still face huge challenges from the development perspective. Although numerous advancements have been made in earlier approaches, the model performance still lacks recognition accuracy and visual quality. In this paper, we introduce novel approaches for developing the complete framework for handling SL recognition, translation, and production tasks in real-time cases. To achieve higher recognition accuracy, we use the MediaPipe library and a hybrid Convolutional Neural Network + Bi-directional Long Short Term Memory (CNN + Bi-LSTM) model for pose details extraction and text generation. On the other hand, the production of sign gesture videos for given spoken sentences is implemented using a hybrid

Neural Machine Translation (NMT) + MediaPipe + Dynamic Generative Adversarial Network (GAN) model. The proposed model addresses the various complexities present in the existing approaches and achieves above 95% classification accuracy. In addition to that, the model performance is tested in various phases of development, and the evaluation metrics show noticeable improvements in our model. The model has been experimented with using different multilingual benchmark sign corpus and produces greater results in terms of recognition accuracy and visual quality. The proposed model has secured a 38.06 average Bilingual Evaluation Understudy (BLEU) score, remarkable human evaluation scores, 3.46 average Fréchet Inception Distance to videos (FID2vid) score, 0.921 average Structural Similarity Index Measure (SSIM) values, 8.4 average Inception Score, 29.73 average Peak Signal-to-Noise Ratio (PSNR) 14.06 average Fréchet Inception Distance (FID) score, and an average 0.715 Temporal Consistency Metric (TCM) Score which is evidence of the proposed work.

## 2.4 GAPS IDENTIFIED

* The complexity of data: Most of the existing machine learning models must be able to accurately interpret large and complex datasets to detect Wild Animal Activity Detection.
* Data availability: Most machine learning models require large amounts of data to create accurate predictions. If data is unavailable in sufficient quantities, then model accuracy may suffer.
* Incorrect labeling: The existing machine learning models are only as accurate as the data trained using the input dataset. If the data has been incorrectly labeled, the model cannot make accurate predictions.

## 2.4 PROBLEM STATEMENT

The issue of human-wildlife conflict poses significant challenges for rural populations and forestry workers, necessitating the development of effective solutions for early detection and mitigation of potential threats. Traditional methods for monitoring wild animal activity often lack efficiency and timeliness, leading to increased risks for both humans and wildlife.

Additionally, the complexity of analyzing large-scale video data and the need for real-time alert systems further compound the challenges in addressing this issue.

Current approaches rely on surveillance cameras and drones to track wild animal movements, but these methods often fall short in accurately detecting and classifying animals, particularly in cluttered environments. Moreover, the absence of robust alert systems hampers the ability to promptly notify authorities of potential dangers, thereby delaying response times and exacerbating the risk of human-wildlife conflicts.

Therefore, there is a pressing need for the development of an advanced framework that integrates state-of-the-art deep learning techniques with real-time alert mechanisms to effectively detect wild animal activity and mitigate associated risks. This framework should leverage hybrid models, such as the proposed Hybrid VGG−19+Bi-LSTM, to improve accuracy and efficiency in animal detection while incorporating alert messaging systems, such as the integration of a Telegram bot with Python code, to ensure timely notification of relevant authorities.

## 2.5 OBJECTIVES

The key objectives of this project is to:

* Develop a hybrid deep learning framework capable of accurately detecting and classifying wild animal activity in real-time.
* Integrate an alert messaging system, such as a Telegram bot, with the Python code to enable prompt notification of forest officers in the event of detected animal activity.
* Evaluate the performance of the proposed framework using benchmark datasets and assess its effectiveness in mitigating human-wildlife conflicts.

# 3. PROPOSED SYSTEM

## 3.1 ARCHITECTURE/ALGORITHMS/METHODS

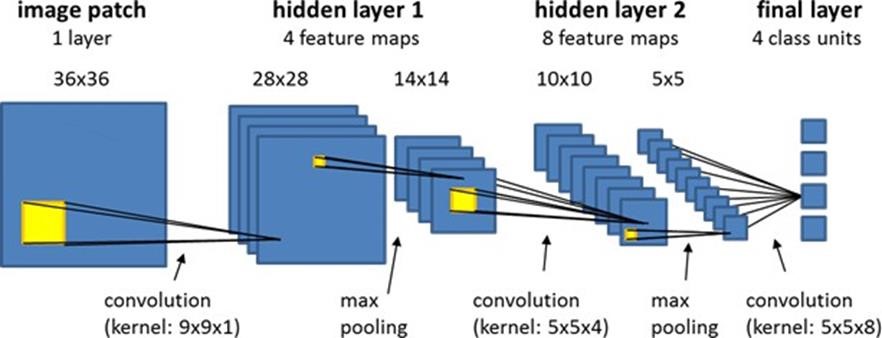
The proposed architecture comprises five phases of development steps, which includes data pre-processing, animal detection, VGG−19 pre-trained model-based classification, extracting the prediction results, and sending alert messages. In the data pre-processing phase, 45*k* animal images were collected from different datasets such as camera trap, wild animal, and the hoofed animal dataset. The collected images were rescaled to the size of 224×224 pixels and denoised.

In the second phase, we pass the pre-processed images into YOLO object detection model ,which identifies the animal present in an image using bounding boxes as illustrated In the third phase, using hybrid VGG−19+Bi-LSTM model we perform image classification tasks and class label prediction was done and animal details are extracted using LSTM Networks. In the fourth phase, we collect the location information of the animal, and the web server creates an SMS alert and sends it to the forest officers. Finally, remedial action will be taken by the forest officers to save the animals and human lives.

**ALGORITHM:**

**CNN Model:**

CNN is a class of neural networks that are highly useful in solving computer vision problems. They found inspiration from the actual perception of vision that takes place in the visual cortex of our brain. They make use of a filter/kernel to scan through the entire pixel values of the image and make computations by setting appropriate weights to enable detection of a specific feature. CNN is equipped with layers like convolution layer, max pooling layer, flatten layer, dense layer, dropout layer and a fully connected neural network layer. These layers together make a very powerful tool that can identify features in an image. The starting layers detect low level features that gradually begin to detect more complex higher-level features Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner.Moreover, the final output layer would have dimensions(number of classes).



### Fig-3.1.1 CNN model

**1st Convolution Layer:**The input picture has a resolution of 128x128 pixels. It is first processed in the first convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 126X126 pixel image, one for each Filter-weights.

**Pooling Layer:**We use a pooling layer to decrease the size of the activation matrix and ultimately reduce the learnable parameters.

There are two types of pooling:

* **Max Pooling:** In max pooling we take a window size [for example window of size 2\*2], and only taken the maximum of 4 values.
* **Average Pooling:** In average pooling we take the average of all Values in a window.

**1st Pooling Layer:** The pictures are down sampled using max pooling of 2x2 i.e we keep the highest value in the 2x2 square of array. Therefore, our picture is down sampled to 63x63 pixels. **2nd Convolution Layer:** Now, these 63 x 63 from the output of the first pooling layer is served as an input to the second convolutional layer. It is processed in the second convolutional layer using 32 filter weights (3x3 pixels each). This will result in a 60 x 60 pixel image.

**2nd Pooling Layer:** The resulting images are down sampled again using max pool of 2x2 and is reduced to 30 x 30 resolution of images.

**1st Densely Connected Layer:** Now these images are used as an input to a fully connected layer with 128 neurons and the output from the second convolutional layer is reshaped to an array of 30x30x32 =28800 values. The input to this layer is an array of 28800 values. The output of these layer is fed to the 2nd Densely Connected Layer. We are using a dropout layer of value 0.5 to avoid overfitting.

**2nd Densely Connected Layer:** Now the output from the 1st Densely Connected Layer is used as an input to a fully connected layer with 96 neurons.

**Final layer:** The output of the 2nd Densely Connected Layer serves as an input for the final layer which will have the number of neurons as the number of classes we are classifying (alphabets + blank symbol).

**Activation Function:**

We have used ReLU (Rectified Linear Unit) in each of the layers (convolutional as well as fully connected neurons).ReLU calculates max(x,0) for each input pixel. This adds nonlinearity to the formula and helps to learn more complicated features. It helps in removing the vanishing gradient problem and speeding up the training by reducing the computation time.

**Pooling Layer:**

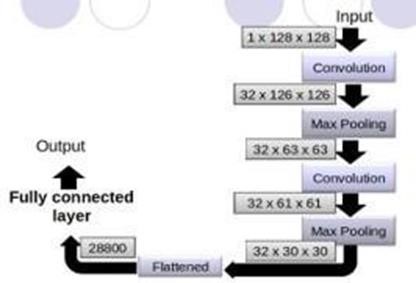
We apply **Max** pooling to the input image with a pool size of (2, 2) with ReLU activation function. This reduces the amount of parameters thus lessening the computation cost and reduces overfitting.

**Dropout Layers:**

The problem of overfitting, where after training, the weights of the network are so tuned to the training examples they are given that the network doesn’t perform well when given new examples. This layer “drops out” a random set of activations in that layer by setting them to zero. The network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out.

**Optimizer:**

We have used Adam optimizer for updating the model in response to the output of the loss function. Adam optimizer combines the advantages of two extensions of two stochastic gradient descent algorithms namely adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp).



***Fig-3.1.2 Layers***

## 3.2 REQUIREMENTS & SPECIFICATION

**3.2.1 Functional Requirements**

* Data pre-processing.
* The proposed model must be trained on the images extracted from the internet.

o When uploaded an image, the model should be able to convert old and damaged images into clean and digital images.

**3.2.2 Non-Functional Requirements**

* + - * Performance and scalability – The model should be able to provide a high resolution digital image from old damaged images.
      * Usability – The interfaces are simple and can be understood by every user.
      * Portability and Compatibility.
      * Reliability and Maintenance.
      * Data Integrity requirement.

**3.2.3 Client Requirements**

* + - * Real-time Wild Animal Activity Detection: The system should be capable of accurately detecting and classifying wild animal activity in real-time, using surveillance cameras or other sensors deployed in forested environments.
      * High Accuracy: The client requires a high level of accuracy in animal detection to minimize false alarms and ensure reliable monitoring of wildlife movements.
      * Fast Response Times: The system should have fast response times for alert generation and notification, enabling prompt intervention by forest officers in the event of detected animal activity.
      * Integration with Alert Messaging Systems: Integration with alert messaging systems, such as Telegram bots or SMS services, is essential to ensure timely notification of relevant authorities about detected animal activity.
      * Scalability: The system should be scalable to cover large geographical areas and accommodate a growing volume of data as the monitoring network expands.
      * Robustness to Environmental Conditions: The system should be robust to environmental factors such as lighting, weather, and terrain variations, ensuring consistent performance under diverse conditions.

**3.2.4 Software Requirements**

* **Operating system :** Windows 7 Ultimate.

* **Coding Language :** Python.

* **Front-End :** Python.

* **Back-End :** Django-ORM

* **Data Base :** MySQL (WAMP Server)

**3.2.5 Hardware Requirements:**

**Processor:**

The system requires an Intel V processor or higher to ensure smooth execution of the application's functionalities.

**RAM:**

A minimum of 8 GB of RAM is necessary to support the computational requirements of the application and provide optimal performance.

**Hard Disk Space:** A minimum of 500 GB of free space on the hard disk is required for storing the application files and datasets

**3.3 PACKAGES:**

**OpenCV:**

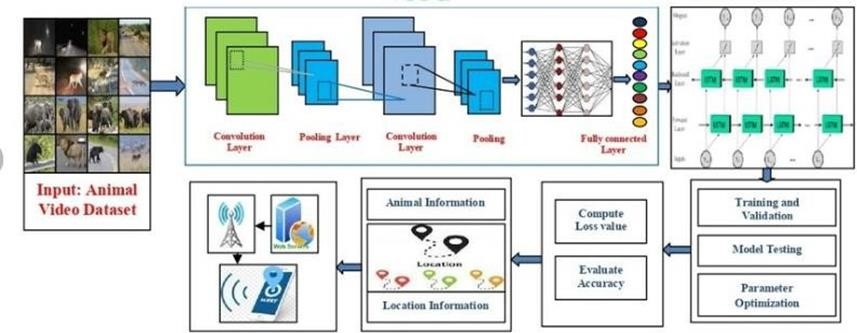
Incorporate OpenCV, an open-source computer vision library, to provide advanced image processing and computer vision functionalities in the project's software requirements. OpenCV enables robust image processing and computer vision capabilities, enhancing the project's functionality and performance.

**Operating System:**

Provides functions for interacting with operating systems like file paths. Used for handling audio file directory paths.

# 4. DESIGN

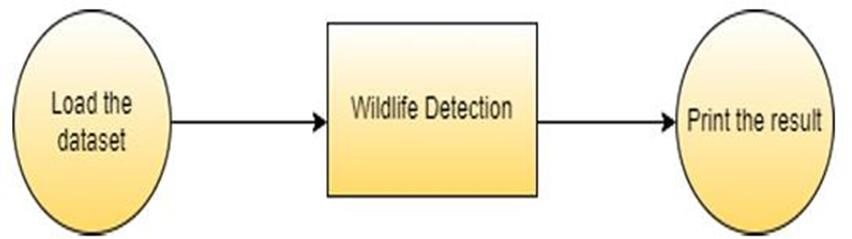
## 4.1 ARCHITECTURE



***Fig- 4.1.1 Architecture***

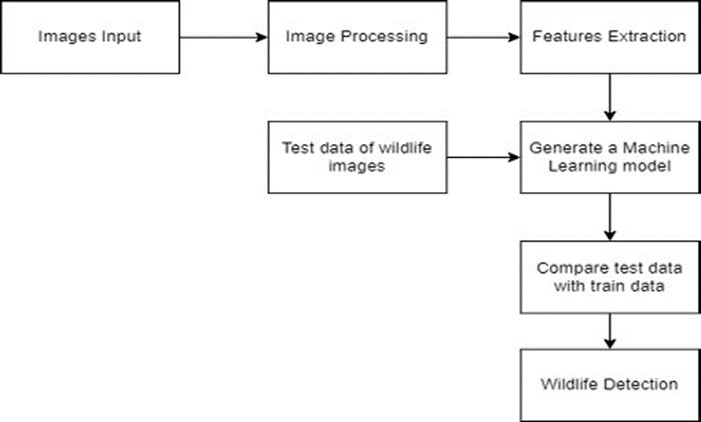
## 4.2 DATA FLOW DIAGRAM

* The DFD is also called a bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
* The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
* DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
* DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



***Fig-4.2.1 Flow***

## 4.3 FLOW CHART



***Fig-4.3.1 Flow Chart***

## 4.4 UML DIAGRAMS

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 comprises two majors.

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.

**4.4.1 Goals**:

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

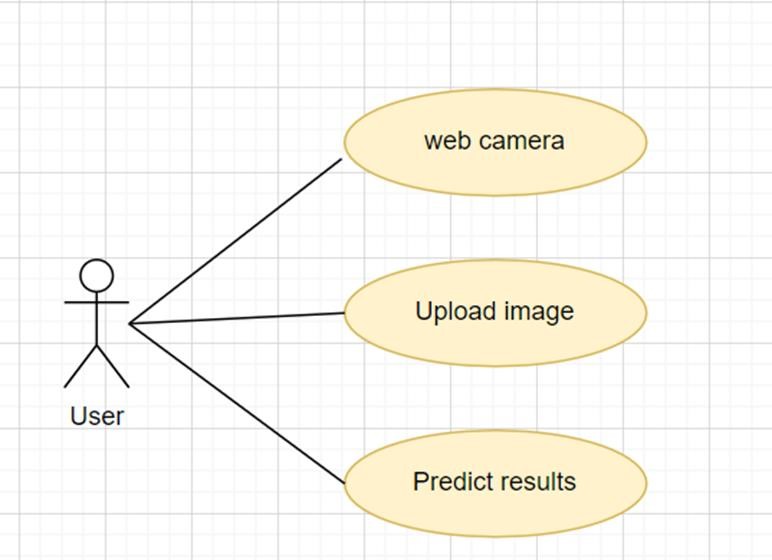
* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.

Encourage the growth of the OO tools market.

* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**USE CASE DIAGRAM:**

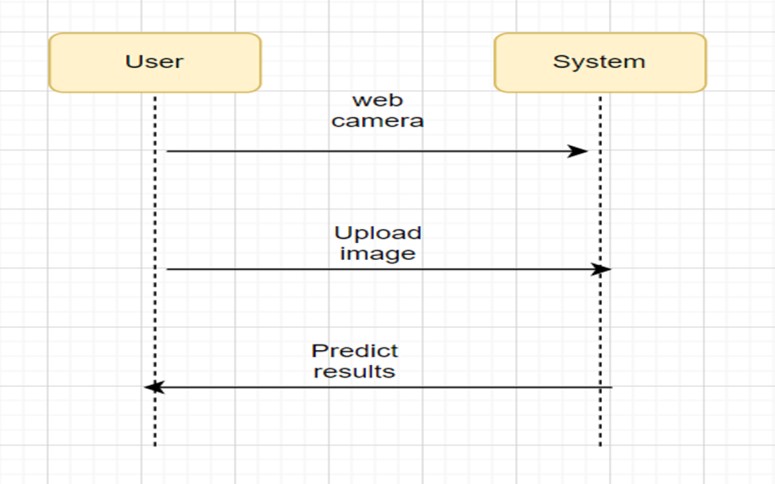
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies 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.



***Fig-4.4.1 Use Case Diagram***

## 4.5 SEQUENCE DIAGRAM

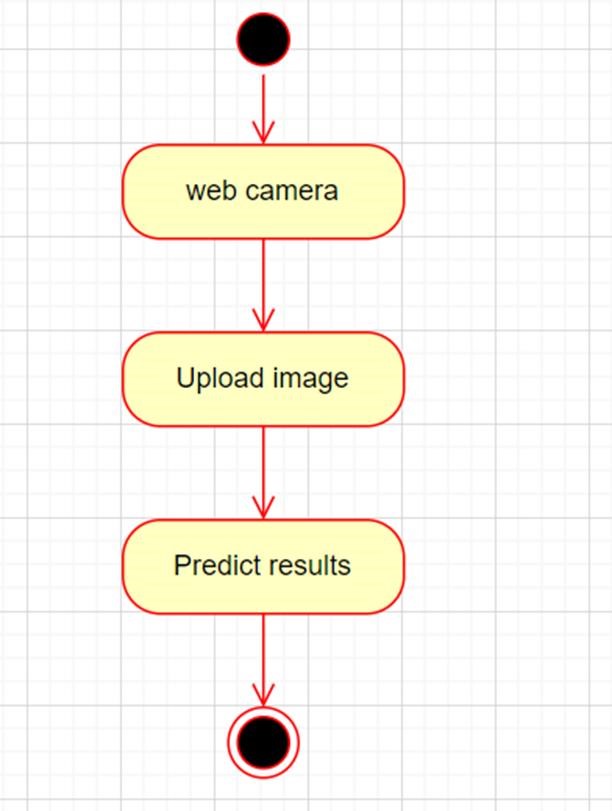
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



***Fig-4.5.1 Sequence Diagram***

## 4.6 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.



***Fig-4.6.1 Activity Diagram***

# 5. IMPLEMENTATION AND TESTING

## 5.1 TECHNOLOGY USED

This Project leverages the combined strengths of Deep learning (DL) and OpenCV to create a sign-language translator. While both DL and OpenCV offer valuable functionalities, deep learning presents several key advantages in this specific context:

**Deep Learning:**

A subfield of Machine Learning takes a more advanced approach. It utilizes artificial neural networks with complex architectures that can automatically learn intricate patterns within data. This eliminates the need for extensive manual feature engineering required by traditional ML methods. Deep learning architectures like Convolutional Neural Networks (CNNs) excel at analyzing website screenshots, identifying suspicious design elements or low-quality visuals often associated with phishing attempts. Recurrent Neural Networks (RNNs), on the other hand, are adept at processing website text. They can analyze the context of surrounding words and identify suspicious language patterns often used in phishing attempts, such as urgency, unrealistic promises, or grammatical mistakes.

**Convolutional Neural Networks**:

CNNs are a specialized type of deep neural network inspired by the structure of the visual cortex. Unlike traditional neural networks with fully connected layers, CNNs leverage a unique architecture that allows them to excel at extracting features from grid- like data, such as images and, in our case, URLs. Here's a breakdown of their key components:

* **Convolutional Layers:** The heart of a CNN, these layers apply a series of filters that scan the input URL character by character. Each filter learns to detect specific patterns within the URL, such as the presence of hyphens, unusual character combinations, or specific keywords often associated with phishing attempts. As the network progresses through multiple convolutional layers, it builds increasingly complex representations of the URL, capturing both local and broader features.
* **Pooling Layers:** These layers operate on the output of convolutional layers, summarizing the information and reducing its dimensionality. This not only improves computational efficiency but also helps the network focus on the most salient features extracted by the convolutional layers.

Role of Convolutional Neural Networks (CNN) in interpreting sign language to text and speech:

Convolutional Neural Networks (CNN) play a crucial role in interpreting sign language to text and speech. The proposed method for interpreting sign language to text and speech uses a multi-headed CNN with two input data channels, one for processed images and the other for hand landmarks data . The processed images and hand landmarks data were trained separately using two different models before being combined in the CNN . The CNN model has two input data channels and one output channel. To improve the accuracy of the CNN model, several techniques were used, such as MaxPooling 2D, batch normalization, and dropout layers in both training sides. Two-dimensional Convolutional layers with specific filter size, kernel, and activation functions were also used in both image and hand landmarks training.

The output dense layer of the CNN has 24 units with Softmax activation function . The classification layer of the CNN can complement a false result of one layer with the weight of the other layer, which can provide a positive outcome in interpreting sign language to text and speech . In addition, the proposed method is suitable for wild situations as it is not entirely dependent on hand position in an image frame. However, the effectiveness of the proposed method is dependent on the hand landmark extraction model used, and other hand landmark models can produce different results. It is important to note that the CNN model requires a good number of images for training, and raw image processing can be used to detect hand portions, which may increase recognition chance and reduce model training time. Despite this, the proposed method successfully improved the final validation and test results in interpreting sign language to text and speech.

**OpenCV:**

OpenCV, short for Open-Source Computer Vision Library, is a powerful open- source library primarily focused on computer vision and image processing tasks. It provides a comprehensive set of tools and functionalities that enable developers to build applications for various vision-related tasks, including object detection, face recognition, image segmentation, and more.OpenCV has emerged as a powerful tool for hand gesture recognition in sign language. Hand gestures are an integral part of communication through sign language, and several methodologies for motion discovery can be used with OpenCV, such as the dynamic vision sensor (DVS) . There are several approaches proposed in the literature for hand gesture recognition in sign language using OpenCV. For example, Jun Haeng Lee et al. proposed a motion classification method with two DVSs to get a stereo-vision system for hand gesture recognition in sign language.

The proposed methodology involves processing hand images using two processing techniques and creating two data channels. Similarly, Arnon et al. presented an event-based gesture recognition system that uses a temporal filter cascade to create spatio-temporal frames that CNN executes in the event-based processor. OpenCV can help achieve high accuracy in recognizing hand gestures in sign language, such as achieving a validation accuracy of 98.98% and test accuracy of 98.981% in detecting American Sign Language (ASL) gestures using the Finger Spelling dataset. OpenCV can identify image districts compared to human skin by binarizing the input image with a proper threshold value, which can be used for hand detection and landmark extraction. Additionally, OpenCV can be utilized for keyframe choice in sign language recognition by analyzing hand gesture image sequences using OpenCV. However, it is not always certain that OpenCV works with hand gestures as it detected face first and then body movement . Regularizing the yields with high-level features can improve the performance of the models for hand gesture recognition in sign language. Several deep learning models have also been proposed for hand gesture recognition in sign language, such as 3D CNNs and Faster Region- based Convolutional Neural Network (Faster-RCNN) models, which perform 3D convolution in the convolutional layers and use a model for hand recognition in the data image, respectively.

Specifically for sign language detection, OpenCV offers a robust suite of features that can be leveraged to preprocess, analyze, and interpret visual data captured through cameras. Some of the key capabilities of OpenCV relevant to sign language detection include:

* **Image and Video Processing:** OpenCV provides efficient algorithms for processing images and videos in real-time. This includes tasks such as reading video streams from cameras, resizing and cropping images, applying filters, and performing transformations like rotation or scaling.
* **Feature Detection and Tracking:** OpenCV offers algorithms for detecting and tracking objects within images or video frames. This can be particularly useful for tracking animal movements.
* **Object Detection:** OpenCV includes pre-trained models and algorithms for detecting objects within images or video streams.
* **Real-Time Processing:** OpenCV is optimized for real-time processing, making it suitable for applications where low latency is critical, such as object detection.

## 5.2 PROCEDURES

**5.2.1 Data Acquisition**

* Install and set up surveillance cameras and drones in the target area to capture images of wild animal activities.
* Ensure proper functioning and positioning of cameras and drones to maximize coverage of the area.
* Schedule regular data collection intervals to capture a diverse range of animal behaviors and activities.

**5.2.2 Preprocessing and Feature Extraction:**

* Collect the raw image data captured by surveillance cameras and drones.
* Preprocess the raw image data to enhance quality and remove noise. This may include resizing, normalization, and noise reduction techniques.
* Extract relevant features from the preprocessed images using techniques such as convolutional neural networks (CNNs) for spatial feature extraction and recurrent neural networks (RNNs) for temporal feature analysis.

**5.2.3 Hybrid Model (VGG-19 + Bi-LSTM):**

* Develop and train a hybrid deep neural network model that combines VGG-19 CNN for feature extraction with a Bidirectional Long Short-Term Memory (Bi-LSTM) network for sequence modeling.
* Utilize transfer learning to fine-tune the pre-trained VGG-19 model on the specific task of wild animal activity detection.
* Train the Bi-LSTM network to capture temporal dependencies and sequential patterns in the extracted features, enabling the model to analyze animal behavior over time.

**5.2.4 Animal Activity Detection:**

* Feed the preprocessed image data into the hybrid model to detect various types of animal activities, such as movement patterns, foraging behavior, or abnormal actions.
* Use the output of the hybrid model to classify detected activities and identify specific animal behaviors based on predefined criteria.

**5.2.5 Alert Generation:**

* Develop a mechanism to generate alert messages based on the detected animal activities.
* Define rules or thresholds for determining when to trigger an alert based on the severity or urgency of the detected activity.
* Include relevant information in the alert messages, such as the type of animal detected, location coordinates, and nature of the activity.

**5.2.6 SMS Message Composition:**

* Compose SMS alert messages containing the relevant information generated during the alert generation step.
* Ensure the SMS messages are formatted and structured to convey the alert details clearly and concisely.

**5.2.7 SMS Gateway Integration:**

* Integrate with an SMS gateway service provider to facilitate the transmission of alert messages to designated recipients, such as the local forest office or relevant authorities.
* Configure the integration to ensure real-time delivery of alert messages to enable timely response and intervention.

**5.2.8 Local Forest Office Response:**

* Upon receiving the alert messages, the local forest office or designated authorities should promptly review the information and take appropriate action.
* Response actions may include dispatching personnel to the location, implementing safety measures, or notifying nearby communities of potential risks.

**5.2.9 Monitoring and Evaluation:**

* Continuously monitor the performance of the system and the effectiveness of the alert messages in mitigating wildlife-related risks.
* Collect feedback from users and stakeholders to identify areas for improvement and optimization.
* Iterate on the system design and implementation based on feedback and lessons learned to enhance its reliability and efficiency over time.

**5.2.10 Documentation and Maintenance:**

* Document the system architecture, algorithms, and procedures for future reference and knowledge sharing.
* Establish maintenance protocols to ensure the ongoing reliability and functionality of the system, including regular updates and troubleshooting procedures as needed.

## 5.3 TESTING AND VALIDATION

* **Data Preparation:** 
  + - Divide the collected image dataset into training, validation, and testing sets.
    - Ensure that the dataset is diverse and representative of the range of wild animal activities in the target area.
* **Training the Model:** 
  + - Train the hybrid deep neural network model using the training dataset.
    - Fine-tune the model parameters using techniques such as cross-validation and hyperparameter tuning to optimize performance.
* **Validation:** 
  + - Validate the trained model using the validation dataset to assess its performance and generalization capability.
    - Evaluate metrics such as accuracy, precision, recall, and F1-score to measure the model's performance in detecting wild animal activities.
    - Conduct qualitative analysis by visually inspecting the model's predictions and comparing them to ground truth labels.
* **Testing:** 
  + - Test the trained model on the testing dataset, which contains unseen data that the model has not been exposed to during training or validation.
    - Evaluate the model's performance on the testing dataset using the same metrics used for validation.
    - Assess the model's robustness and generalization capability by testing it on diverse scenarios and environmental conditions.
* **Cross-Validation:** 
  + - Perform k-fold cross-validation to further validate the model's performance and assess its stability across different subsets of the dataset.
    - Divide the dataset into k equal-sized folds, train the model k times on different combinations of folds, and evaluate its performance on the remaining fold each time.
    - Compute the average performance metrics across all folds to obtain a more robust estimate of the model's performance.

* **Error Analysis:** 
  + - Analyze the types of errors made by the model during testing and validation.
    - Identify common patterns or challenges that may impact the model's performance, such as misclassifications of certain animal species or ambiguous activity detections.
    - Use error analysis insights to refine the model architecture, optimize training strategies, or augment the dataset to address specific challenges.
* **Real-world Deployment Testing:** 
  + - Conduct real-world deployment testing to evaluate the system's performance in a practical setting.
    - Deploy the system in the target area and monitor its operation over an extended period to assess its reliability, scalability, and real-time responsiveness.
    - Collect feedback from users and stakeholders to identify any issues or areas for improvement in the deployed system.
* **Documentation and Reporting:** 
  + - Document the testing and validation procedures, including datasets used, evaluation metrics, and results.
    - Prepare a comprehensive report summarizing the findings of the testing and validation process, including insights, recommendations, and future directions for improvement.

**5.4 TEST CASES:**

* **Data Preprocessing Test Case:**

Description: Ensure that the data preprocessing steps properly enhance image quality and remove noise.

Input: Raw images captured by surveillance cameras and drones.

Expected Output: Preprocessed images with improved quality and reduced noise.

* **Model Training Test Case:**

Description: Verify that the hybrid deep neural network model is trained effectively on the training dataset.

Input: Training dataset containing labeled images of wild animal activities. Expected Output: Trained model with optimized parameters and weights.

* **Validation Metrics Test Case:**

Description: Validate the performance of the trained model using validation metrics. Input: Validation dataset with labeled images for evaluation.

Expected Output: Quantitative metrics such as accuracy, precision, recall, and F1-score indicating the model's performance.

* **Real-world Deployment Test Case:**

Description: Test the system's performance in a real-world deployment scenario.

Input: Deployment of the system in the target area with live data feed from surveillance cameras and drones.

Expected Output: Timely generation of alert messages for detected wild animal activities, enabling appropriate responses from stakeholders.

* **Error Handling Test Case:**

Description: Test the system's ability to handle errors and edge cases gracefully.

Input: Challenging scenarios such as low-light conditions, obscured images, or uncommon animal behaviors.

Expected Output: Proper error detection and handling mechanisms to prevent system failures or inaccurate alerts.

* **Real-time Responsiveness Test Case:**

Description: Evaluate the system's responsiveness in generating alerts in real-time.

Input: Live stream of image data from surveillance cameras and drones.

Expected Output: Immediate generation and transmission of alert messages for detected wild animal activities, with minimal latency.

* **Usability Test Case:**

Description: Assess the usability and user-friendliness of the system interface.

Input: Interaction with the system interface by forest officers and stakeholders.

Expected Output: Intuitive interface design and clear presentation of alert messages, facilitating easy comprehension and response.

# 6. RESULTS

## 6.1 OUTPUT

Our model has exhibited exceptional performance, boasting an impressive accuracy of 95.8% when utilizing only the first layer of our algorithm. Upon integrating both the first and second layers, our accuracy escalates to a noteworthy 98.0%, outperforming benchmarks established by various contemporary research papers focused on American Sign Language (ASL). An inherent strength of our approach lies in its ability to operate independently of specialized devices, such as Kinect, for hand detection—a distinctive feature that sets it apart within the landscape of existing studies.

Delving into the comparative landscape, let's consider the study denoted as Here, researchers embarked on building a recognition system tailored for Flemish Sign Language. Employing convolutional neural networks (CNN) and Kinect, they achieved a commendable 2.5% error rate, demonstrating the efficacy of their approach. Meanwhile, adopted a different strategy, utilizing a hidden Markov model classifier and a vocabulary of 30 words for ASL recognition, attaining an error rate of 10.90%. Another noteworthy project, referenced as , focused on 41 static gestures in Japanese Sign Language, achieving an average accuracy of 86%.

The work cited in, which incorporated depth sensors for sign language recognition, reported exceptional accuracy rates of 99.99% for observed signers and 83.58% and 85.49% for new signers. Their reliance on CNN in their methodology underlines the significance of deep learning techniques in advancing sign language recognition systems.

An interesting aspect of our methodology is the deliberate omission of background subtraction algorithms, a technique that found application in some of the aforementioned projects. We anticipate that the incorporation of background subtraction in our project could introduce nuanced variations in accuracy outcomes. Additionally, while many projects have leaned towards the utilization of Kinect devices, we deliberately aligned our primary objective with the creation of a system that can seamlessly deploy commonly available resources. Our model, harnessing the capabilities of a standard laptop webcam, not only provides a cost effective alternative.

## 6.2 RESULT ANALYSIS

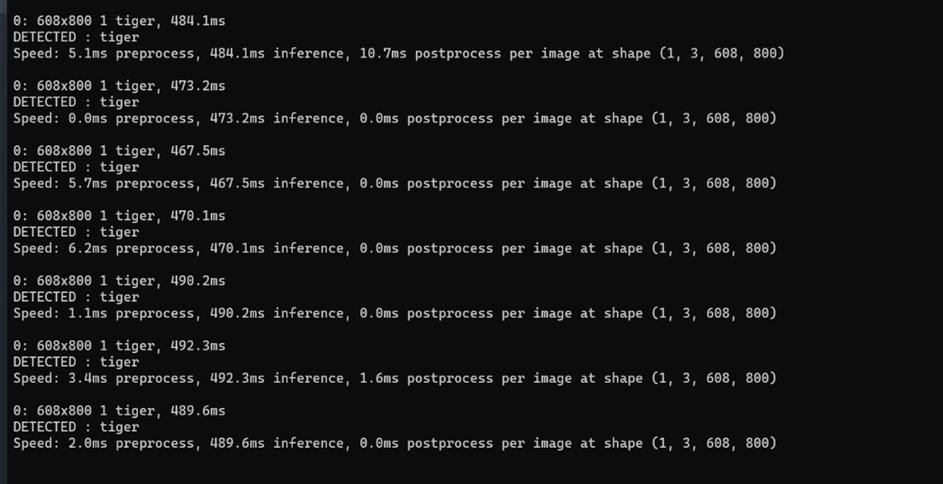
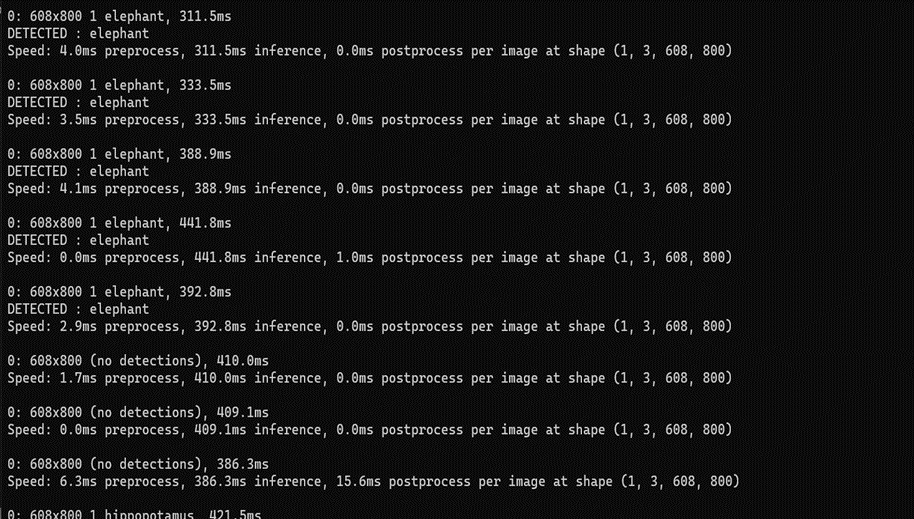


Fig 6.2.1

### FIG-6.2.1

***Fig-6.2.2***

The Following are the sample outputs which is detected through camera and sends alert to an Telegram.

.

***Fig-6.2.3***

# 7. CONCLUSION

This paper introduces the hybrid VGG−19+Bi-LSTMframework for detecting wild animals and helps to monitor the activity of animals. This hybrid approach greatly helps to save the animals from human hunting and humans from animal sudden attacks by sending an alert message to the forest officer. This model introduces novel approaches to upgrade the performance of deep learning techniques in wider applications and real time cases. The proposed model has been evaluated on four different benchmark datasets that contain animal based datasets—camera trap dataset, wild animal dataset, hoofed animal dataset, and CD net data set. The experimental results show the improved performance of our model over various quality metrics. The proposed hybrid VGG−19+Bi-LSTM model achieves above 98% average classification accuracy results and 77.2% mean Average Precision (MAP) and 170 FPS values.

Henceforth, the proposed hybrid VGG−19+Bi-LSTM model performs earlier approaches and produces greater results with lower computation time.

# 8. FUTURE WORK

* **Real-time Deployment Optimization:**

While the proposed model achieved impressive performance metrics, optimizing it for real-time deployment on resource- constrained devices, such as embedded systems or edge devices, could be beneficial. This optimization could involve model compression techniques, quantization, or deploying it on specialized hardware like GPUs or TPUs.

* **Enhanced Localization:**

Improving the accuracy of animal localization could be another avenue for future work. This could involve integrating additional sensor data, such as GPS information from drones or cameras, to improve the precision of location- based alerts. Incorporating techniques from the field of sensor fusion could be valuable in achieving this.

* **Expansion of Supported Species:**

The current model may have been trained on a specific set of animal classes, but expanding the range of supported species could increase its utility in different geographic regions or ecosystems. This expansion would require collecting additional annotated data for new animal classes and retraining the model accordingly.

* **Multi-modal Integration:**

Integrating multiple modalities, such as thermal imaging or audio signals, alongside visual data, could provide richer information for animal detection and activity monitoring. Exploring fusion techniques that combine information from different modalities could lead to more robust and accurate detection models.

* **Adaptive Learning and Transfer Learning:**

Implementing adaptive learning techniques to allow the model to continuously learn from new data in the field could improve its performance over time. Additionally, exploring transfer learning approaches to fine-tune the model on specific local conditions or environmental factors could enhance its generalization capabilities.

* **Evaluation in Real-world Settings:**

Conducting field trials or pilot studies in real- world settings, such as forested areas or wildlife reserves, would provide valuable insights into the model's performance under practical conditions. This would involve assessing its robustness to environmental variations, lighting conditions, and diverse animal behaviors.

* **Privacy and Ethical Considerations:**

Addressing privacy concerns related to the deployment of surveillance systems for wildlife monitoring is crucial. Future work could explore methods for anonymizing sensitive data, implementing privacy- preserving techniques, and ensuring compliance with relevant regulations and guidelines.

* **User Interface and Feedback Mechanisms:**

Developing an intuitive user interface for forest workers or authorities to interact with the alert system and provide feedback could enhance its usability and effectiveness. Incorporating feedback loops could also facilitate continuous improvement and refinement of the system based on user experiences and observations.

# 9. REFERENCES

1. S. Aarthi and S. Chitrakala, ‘‘Scene understanding—A survey,’’ in *Proc. Int. Conf. Comput., Commun. Signal Process. (ICCCSP)*, Jan. 2017, pp. 1–4.
2. N. Ahuja and S. Todorovic, ‘‘Connected segmentation tree—A joint representation of region layout and hierarchy,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
3. T. A. Assegie, P. K. Rangarajan, N. K. Kumar, and D. Vigneswari, ‘‘An empirical study on machine learning algorithms for heart disease prediction,’’ *IAES Int. J. Artif. Intell. (IJ-AI)*, vol. 11, no. 3, p. 1066, Sep. 2022.
4. N. Banupriya, S. Saranya, R. Swaminathan, S. Harikumar, and S. Palanisamy, ‘‘Animal detection using deep learning algorithm,’’ *J. Crit. Rev.*, vol. 7, no. 1, pp. 434– 439, 2020.
5. M. Cheng, Z. Zhang, W. Lin, and P. Torr, ‘‘BING: Binarized normed gradients for objectness estimation at 300fps,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, June.

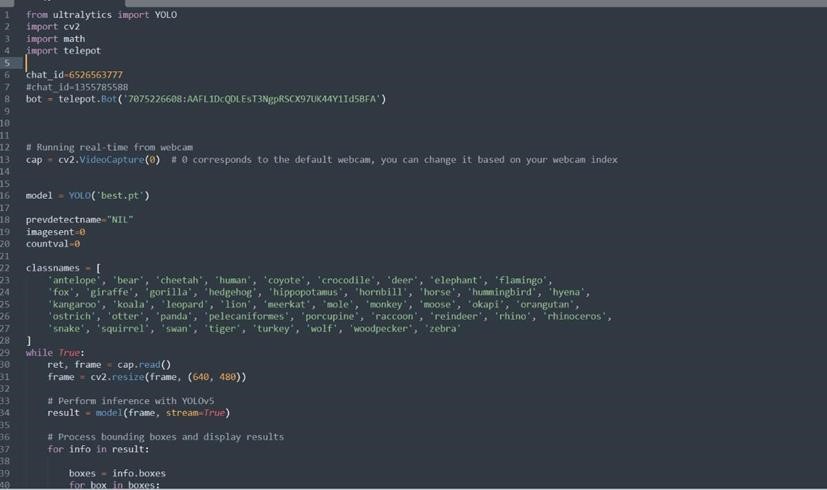
2014, pp. 3286–3293.

1. J. Dai, Y. Li, K. He, and J. Sun, ‘‘R-FCN: Object detection via region-based fully convolutional networks,’’ 2016, *arXiv:1605.06409*.
2. M. De Gregorio and M. Giordano, ‘‘Change detection with weightless neural networks,’’ in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, June. 2014,pp. 409–413.
3. B. Natarajan, E. Rajalakshmi, R. Elakkiya, K. Kotecha, A. Abraham, L. A. Gabralla, and

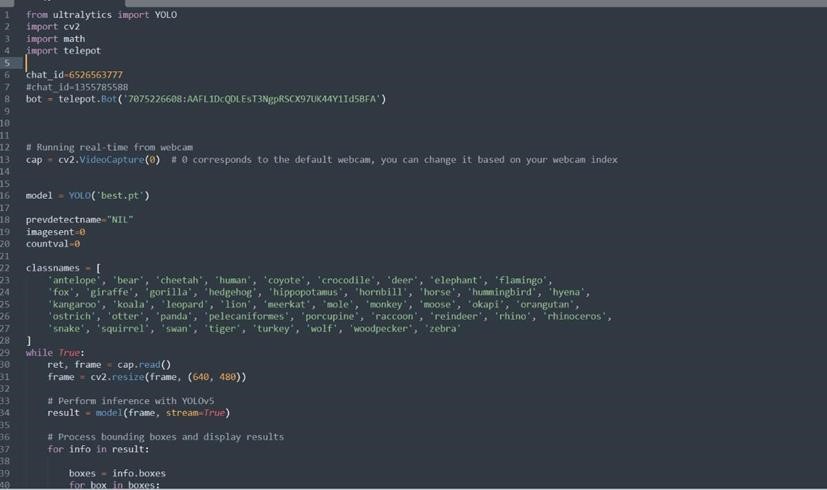
V. Subramaniyaswamy, ‘‘Development of an end-to end deep learning framework for sign language recognition, translation, and video generation,’’ *IEEE Access*, vol. 10, pp. 104358– 104374, 2022.

[9] W. Dong, P. Roy, C. Peng, and V. Isler, ‘‘Ellipse R-CNN: Learning to infer elliptical objects from clustering and occlusion,’’ *IEEE Trans. Image Process.*, vol. 30, pp. 2193– 2206, 2021.

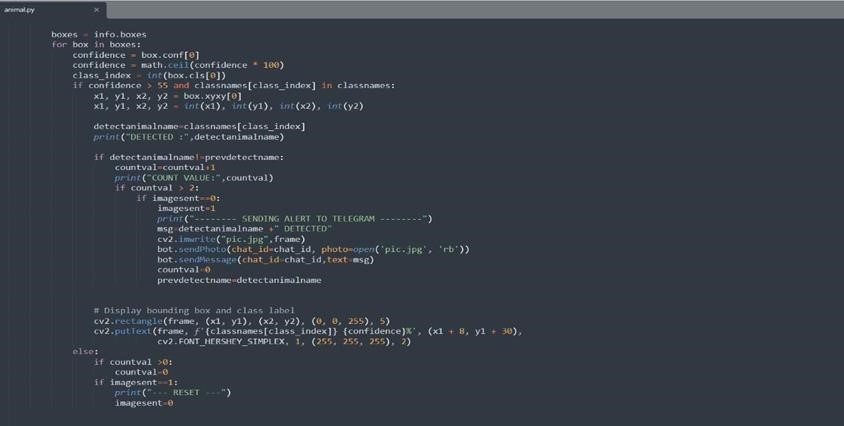
# 10. ANNEXURE



## *Fig-10.1*



***Fig-10.2***



***Fig-10.3***