CNN-Powered Real-Time Object Detection Framework for Intelligent Farm Monitoring

Kapil G. Dhappadhule[1], Sushil V. Kulkarni[2]
[1,2]*Department of Computer Science and Information Technology, Dr. Babasaheb Ambedkar Technological University, Lonere*

***Abstract*—Agricultural security is a critical concern as farms are frequently exposed to intrusions by humans, stray animals, and birds, all of which contribute to significant crop damage, reduced yield, and financial loss. Conventional monitoring techniques, such as manual supervision and CCTV- based surveillance, are limited to passive recording and require constant human attention. These systems fail to provide immediate notifications or proactive measures, often allowing incidents to escalate before corrective action can be taken.**

This paper proposes a real-time, intelligent surveillance system that combines Convolutional Neural Networks (CNNs) with advanced object detection frameworks such as YOLOv8 to address these challenges. The system continuously monitors farm environments through live video streams, accurately classifies detected entities as humans, animals, or birds, and automatically generates snapshots of intrusion events. To ensure timely intervention, the framework is integrated with the Telegram Bot API, enabling instant alerts to be delivered directly to administrators' devices.

The system not only improves response time compared to traditional surveillance but also provides a scalable and cost-effective solution that can be deployed across small farms as well as large agricultural landscapes.

***Index Terms*—Object Detection, Smart Farming, YOLOv8, CNN, Agricultural Surveillance, Telegram Alerts.**

I. INTRODUCTION

Agriculture remains one of the most vulnerable sectors to external disturbances, particularly intrusions by humans, stray animals, and birds. Such events frequently cause damage to crops, reduce overall yield, and impose significant financial burdens on farmers. Beyond economic impact, the presence of intruders can also create safety risks for agricultural workers and disrupt farming operations. The need for effective surveillance in rural and semi- urban farm environments is therefore of paramount importance. Traditional monitoring strategies such as manual guarding or the use of closed-circuit television (CCTV) cameras have been widely adopted, but these methods suffer from major limitations. Manual supervision is labor- intensive and inconsistent, while CCTV systems provide only passive video feeds without the intelligence to interpret or alert in real time [1].

Recent advances in computer vision and artificial intelligence (AI) have introduced new opportunities for automated surveillance in agriculture. In particular, Convolutional Neural Networks (CNNs) have emerged as powerful tools for object detection and classification, enabling systems to identify specific categories such as humans, animals, or birds with high accuracy. However, farm environments introduce unique challenges for computer vision tasks. Variability in lighting conditions, occlusion from vegetation, complex natural backdrops, and the need to detect small and fast-moving objects such as birds can all reduce detection accuracy [2].

Addressing these challenges requires lightweight yet robust deep learning models that can function effectively under constrained hardware and network conditions.

This research proposes a real-time detection and alert framework that combines CNN- based object detection with Telegram-based instant notifications. The system continuously monitors live video streams from farm cameras, classifies detected entities, captures intrusion snapshots, and immediately transmits them to farm administrators. Unlike conventional surveillance methods, the framework is proactive, scalable, and cost-efficient, allowing timely intervention while minimizing the need for human monitoring.

The novelty of this study lies in its integration of AI-driven vision models with IoT communication platforms, bridging the gap between detection and rapid response. This contribution supports the broader vision of smart agriculture, where automation, efficiency, and sustainability are key to protecting farm resources and improving productivity [4].

II. LITERATURE REVIEW

The literature survey serves as the foundation for understanding the current research landscape in digital healthcare, symptom analysis, medical triage, and pre-diagnosis systems. It examines existing models, tools, and strategies designed to enhance early disease identification, bridge communication gaps between patients and doctors, and promote inclusive healthcare delivery. The aim of this chapter is to review prior work, highlight their strengths and weaknesses, and establish the research gap that motivates the development of the Smart Symptom Navigator for Inclusive Pre-Diagnosis.

You and Gui (2021) examined chatbot-based symptom checkers that guide users through text-based self-diagnosis using rule-based dialogue flows and NLP. Their findings revealed that although such systems simplify medical information retrieval, they are often difficult to use for low-literacy and non-English-speaking individuals due to heavy reliance on medical terminology and rigid interaction formats. This limitation directly informed the inclusion of multilingual communication and visual body diagrams in the Smart Symptom Navigator to minimize linguistic barriers.

Wiedermann et al. (2023) investigated AI-driven triage systems built on deep neural networks for improving diagnostic efficiency in primary care. Though these tools demonstrated notable gains in accuracy and automation, their lack of interpretability and high computational requirements restricted their deployment in low-resource environments. This highlighted the need for a lightweight, transparent, and rule-based approach to pre-diagnosis, which the Smart Symptom Navigator adopts.

Wang et al. (2021) proposed an inductive heterogeneous graph convolutional network for mapping relations between symptoms, diseases, and treatments using graph neural networks. While the method improved diagnostic precision, it depended on massive annotated datasets and powerful hardware, making it unsuitable for areas with limited infrastructure. Similarly, Madotto et al. (2021) combined reinforcement learning with classification models to dynamically optimize questioning patterns, achieving high diagnostic performance but requiring continuous retraining and GPU-based operation. These limitations reinforced the practicality of a simpler and interpretable rule-driven reasoning model in the proposed system.

Singh et al. (2022) developed a medical chatbot using traditional machine-learning models and NLP for disease prediction. Their work demonstrated success in spreading preventive health awareness, yet lack of multilingual support and dependency on stable internet connectivity reduced applicability in rural regions. Gordon et al. (2020) also explored web-based symptom checkers utilizing structured question series, but their systems remained text- only and inaccessible to individuals with low literacy. Both studies emphasized the necessity for healthcare interfaces that integrate linguistic flexibility and visual guidance components that are core to the Smart Symptom Navigator.

Patel and Mehta (2023) proposed a rule-based triage and doctor-recommendation system focused on fast deployment and interpretability. However, the complexity of updating clinical rules and maintaining scalability remained an obstacle. Their contributions shaped the Navigator's modular design, where medical rules are stored in scalable JSON schemas for easy updates. Meanwhile, Ahmed et al. (2021) explored speech-enabled disease reporting for users with low literacy levels. Although promising, speech accuracy was severely affected by accent variations and environmental noise. This insight led the Smart Symptom Navigator to provide voice input as an optional rather than mandatory interaction method.

Joshi and Nair (2023) emphasized the importance of Human-Centered Design for designing inclusive healthcare portals. Their work showed that culturally sensitive visuals, localized languages, and icon-based interaction enhance acceptance in underserved communities. This directly influenced the system's emphasis on visual body-based symptom reporting and multilingual design. Furthermore, studies by Razzak et al. (2020) and Ghosh & Banerjee (2023) examined AI-based chatbots, big-data analytics, and hybrid healthcare models. While technologically

impressive, these systems face ethical challenges, privacy concerns, and lack practicality in low-infrastructure regions.

Rahman et al. (2023) demonstrated that multilingual conversational agents using transfer learning can significantly improve accessibility in low-resource areas, although accent and dialect variations remain unresolved challenges. Lastly, Smith et al. (2021) highlighted the importance of explainable AI in medical decision-support

systems, showing that clinicians favor interpretable systems that provide transparent reasoning pathways rather than opaque black-box models. Their findings support the design philosophy of the Smart Symptom Navigator, which prioritizes traceable decision steps and medically interpretable outputs.

Overall, the reviewed literature highlights recurring patterns: advanced AI-driven symptom diagnostic systems achieve high clinical accuracy but lack affordability, interpretability, and accessibility features necessary for real-world adoption in rural and semi-urban India. In contrast, rule-based and human-centered models excel in transparency and usability but lack adaptability without manual updates. The Smart Symptom Navigator for Inclusive Pre-Diagnosis bridges these gaps by combining rule-based reasoning with visual, multilingual, user-friendly interaction while maintaining scalability for future integration with AI and NLP. Thus, it provides a balanced solution that promotes inclusive, accessible, and explainable pre- diagnosis, especially suited for populations with limited infrastructure and digital literacy.

III. EXISTING SYSTEM

In order to improve farm protection and prevent agricultural losses, artificial intelligence (AI), computer vision (CV), and deep learning models have been widely used in surveillance solutions in recent years. A variety of AI-powered smart security systems, wildlife detection tools, and CCTV-based monitoring platforms have emerged to help farmers identify intruders and take quick action to safeguard crops, livestock, and property. Existing systems such as camera-enabled animal deterrents, smart CCTV platforms, and IoT-based security alarms allow farmers to monitor farmland and receive alerts when suspicious movement occurs. These platforms typically provide video-based or sensor-based interfaces that collect real-time footage and link users to cloud detection engines or mobile alert applications, and are mainly designed for commercial farms or urban users with strong infrastructure support. In order to detect intruders such as humans and wild animals, the majority of these platforms rely on CNN-based object detection models and cloud-based inference engines. Although these systems have significantly contributed to automation in farm security, they have certain drawbacks. Their accuracy and reliability frequently depend on high-quality network connectivity, advanced cameras, computing power, and the ability of the user to handle applications and technology. Consequently, many rural farming communities are still unable to access these systems, especially in regions where internet instability, low digital literacy, and limited economic resources are common.

3.1 Working of Existing System

The architecture of most current smart farm surveillance systems is similar and consists of four primary stages.

1. Input Interface: CCTV or IP cameras capture live video from farmland and transfer it to a processing device. Some systems use motion sensors to activate detection only when movement is detected.
2. Object Detection Database / Knowledge Base: The captured frames are passed to a CNN-based detection model stored on cloud servers or a local machine. These systems rely on pre-trained datasets of animals and humans to identify intruders. For instance, YOLO, SSD, and Faster R-CNN models are commonly used to classify species or individuals.
3. Inference Engine: The system analyzes the type of detected object and decides whether it is a threat to the field. Based on the output, the system performs decision-making such as sounding an alarm, activating a light, or generating a warning signal for the farmer.
4. Alert Generation: The final output is sent to the farmer, often through SMS notifications, email alerts, or a dedicated mobile app. Some systems also store the recorded footage on cloud servers for later viewing.

3.2 Limitations

Despite their advantages, current smart farm surveillance solutions are not fully suitable for rural agricultural environments due to several drawbacks.

1. High cost and resource dependency: Most AI-based detection systems require expensive cameras, GPU-based processing units, and cloud servers, increasing installation and operational costs.
2. Network dependency: Many platforms require high-speed internet for continuous video uploading and cloud processing, making them unusable in areas with weak network coverage.
3. Reduced interpretability and accuracy in outdoor environments: Detection performance often drops due to dust, rain, fog, moving leaves, shadows, and low light, which creates false alarms and missed detections.
4. Usability barriers: The majority of existing systems rely on complicated mobile apps and text-based interfaces, which are difficult for farmers with low digital literacy and limited access to smartphones.
5. Privacy and data concerns: Cloud-based storage of farm footage may raise confidentiality issues and make farmers uncomfortable with external access to their camera recordings.

IV. PROPOSED METHODOLOGY

The CNN-based methodology for this intelligent surveillance system starts with collecting farm surveillance video from fixed cameras and manually labeling frames with bounding boxes and classes for relevant objects such as humans, vehicles, and animals. The images are then preprocessed by resizing to a fixed input size, normalizing pixel values, and applying data augmentation (flips, scaling, brightness changes) to make the model robust to outdoor variability.

A single-stage CNN detector (for example, a YOLO-family model) is used, with a backbone network to extract multi-scale features and a detection head that directly predicts bounding boxes, objectness scores, and class probabilities in a single forward pass, enabling real-time performance. The network is initialized with pre-trained weights and fine-tuned on the annotated farm dataset using a composite loss for localization and classification, with training monitored via mean average precision, precision, and recall to select the best model. In deployment, the trained CNN runs on an edge device, processes live video streams, applies non- maximum suppression and confidence thresholds, and whenever an intruder or animal is detected with high confidence, the

system captures the frame, overlays detection results, and automatically sends an alert and image to farmers through Telegram notifications (Figure 2).

A. System Architecture

The system architecture is composed of interconnected modules that collectively perform image acquisition, object detection, threat analysis, and notification delivery. It follows a modular architecture to support easy scaling and maintenance.

1. Camera Module

A live camera feed captures continuous video streams from the farm. It may include a USB camera, CCTV camera, or Raspberry Pi camera module depending on deployment.

2. Frame Processing Unit

Each frame from the video stream is preprocessed using OpenCV. Operations include resizing, normalization, and noise reduction to improve detection accuracy.

3. YOLOv8 Object Detection Engine

The core detection module uses a CNN-based YOLOv8 model that performs real-time object classification and localization. It identifies humans, animals (cow, dog, boar, monkey), and other relevant objects.

4. Threat Classification Module

The detection output is analyzed to differentiate between harmless objects and potential intruders or dangerous wildlife. Decisions are based on confidence thresholds and predefined threat categories.

5. Telegram Alert System

If a threat is detected, an image snapshot of the frame along with a text alert is sent automatically to the farmer’s smartphone using the Telegram Bot API.

6. Data Logging and Storage

Detected threat frames are saved locally or on cloud

storage for future analysis and verification.

B. Workflow of the System

The operational workflow of the system follows a sequential process that enables continuous monitoring and timely threat detection. The workflow consists of the following steps:

1. Camera Initialization

The system begins by activating the farm camera and reading live video frames.

2. Real-Time Frame Capture

Frames are captured at fixed intervals and immediately transferred for processing.

3. Preprocessing

Each frame undergoes image transformations that optimize it for CNN analysis.

4. YOLOv8 Inference

The preprocessed frame is fed to YOLOv8, which returns detected objects with bounding boxes and confidence scores.

5. Threat Analysis

The detection results are checked against a threat database. Humans, wild animals, or unidentified objects are classified as threats.

6. Decision Making

If not a threat, the system continues scanning the next frame.

- If threat detected, the system triggers the alert mechanism.

7. Telegram Notification

A message containing the object type, timestamp, and captured image is instantly sent to the farmer's Telegram account.

8. Continuous Monitoring Loop

The process repeats endlessly, ensuring 24×7 autonomous surveillance.

C. Algorithmic Flow

The algorithmic flow describes the internal logic of the detection and alert system. It operates on a forward-processing mechanism optimized for real-time execution.

Step 1: Initialize camera and load YOLOv8 model

Step 2: Capture frame from live video feed

Step 3: Preprocess frame (resize, normalize, convert to RGB)

Step 4: Run YOLOv8 detection on frame

Step 5: Extract detection results (labels + confidence values)

Step 6: For each detected object:

- If label ∈ {human, cow, dog, boar, monkey} AND confidence > threshold ⇒ Threat Detected
- Else ⇒ Not a Threat

Step 7: If threat detected:

- Save frame
- Send alert + image via Telegram Bot API

Step 8: Continue capturing new frames in loop

This flow ensures minimal processing delays and allows the system to operate continuously under real-world field conditions.

D. Advantages of the Proposed System

The proposed surveillance system offers several benefits that make it suitable for modern agricultural environments:

1. Real-Time Monitoring

The YOLOv8 model processes frames within milliseconds, ensuring immediate detection and response.

2. High Accuracy and Robust Object Recognition

CNN-based detection provides accurate classification of humans, animals, and multiple farm-related objects, reducing false alarms.

3. Instant Alerts via Telegram

The system delivers real-time alerts directly to the farmer's smartphone, enabling quick action to prevent crop damage or theft.

4. Low-Cost and Scalable Architecture

Uses open-source tools and affordable hardware (e.g., Raspberry Pi), enabling deployment in rural and small-scale farms.

5. Continuous Autonomous Operation

The system runs 24×7 without manual intervention, improving farm safety and reducing labor dependence.

6. Easy Integration and Customization

Additional object categories or sensors can be added easily due to the modular design.

7. Energy Efficient

Optimized frame processing reduces unnecessary computation, enabling battery- powered or solar-powered operations.

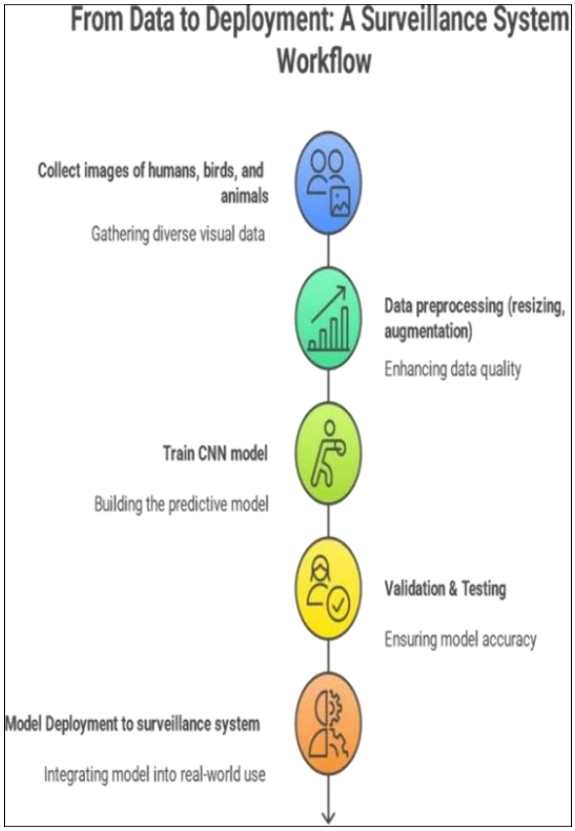

Figure 1 : Flowchart of Methodology

Figure 2: System Architecture of the Proposed Model

V.IMPLEMENTATION

The implementation phase translates the proposed smart farm surveillance methodology into a fully functional prototype. The system integrates real-time object detection using a CNN-based YOLO model, continuous farm monitoring through IP/CCTV cameras, and instant Telegram alerts to farmers with annotated intruder images. The architecture follows a modular development approach where the Detection Module, Alert Module, Camera Interface, Edge Processing Unit, and Storage Unit are developed as independent units and later integrated. This modularity enables smoother debugging, scalability, and support for future upgrades such as drone-based monitoring and night-vision integration.

5.1 Development Environment Hardware & Deployment Unit

Raspberry Pi / NVIDIA Jetson Nano (for edge detection)
HD / IP CCTV Camera (live video feed)
Object Detection Engine
YOLOv8 (real-time tracking and bounding box generation)
OpenCV (video frame acquisition and preprocessing)

Communication Layer
Telegram Bot API (alert delivery) Python Requests library (message/image delivery through API calls)
Supporting Tools & Libraries
Python 3.10
Ultralytics YOLO framework
NumPy and Pandas for data processing Firebase (for optional storage)
GitHub for version control

5.2 . Module-Wise Implementation

5.2.1Camera Interface Module

Live video stream captured through IP/USB camera using OpenCV. Frames are extracted at a fixed rate and forwarded to the detection model. Brightness and contrast normalization are applied to handle outdoor illumination changes.

5.2.2Camera Interface Module Diagram

Camera → Frame Extraction → Preprocessing → YOLO Detection

5.2.3. Detection Module (CNN – YOLO Engine)
YOLOv8 model loaded on the edge device (Raspberry Pi / Jetson Nano). For each frame, the model detects objects such as human, cow, buffalo, dog, and wild animals. Detected objects are highlighted using bounding boxes and confidence scores. Only relevant intruders trigger alerts, minimizing false warnings.
Example JSON rule file for threat mapping
{
"class": "Human", "threat_level": "High",
"action": "Send Immediate Alert"
}

5.3 Alert & Messaging Module

- Implemented using Telegram Bot API through Python.
- When an intruder is detected, the system:

—Captures the annotated frame
— Generates a timestamped alert message
— Sends both the alert message and the image to Telegram

- The alert contains:

— Type of intruder
— Time of detection
— Location nickname (Farm Zone A / Cattle Shed etc.)

5.4. Storage / Logging Module

- Alerts and detected frames are stored locally for evidence tracking.
- Optional cloud storage is integrated using Firebase for long-term data logging.
- Stored data supports farm-security analytics (e.g., frequent intrusion timing).

5.5 Workflow of Implementation
Farm owner powers up the edge device and monitoring camera. The YOLO engine continuously processes captured video frames. When an intruder (human / cow / wild animal / bird) is detected, the bounding box is generated. The system verifies whether the detected object is listed as a threat. If yes → annotated image + alert message are sent to the farmer via Telegram. All detection logs and frames are saved for security review. System continues monitoring autonomously.

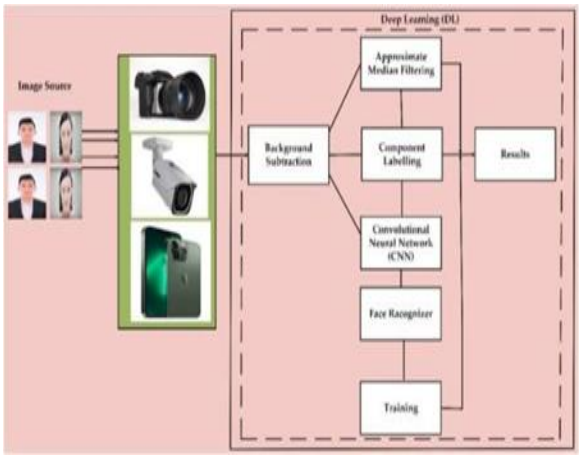

Figure 3 : Implementation workflow diagram

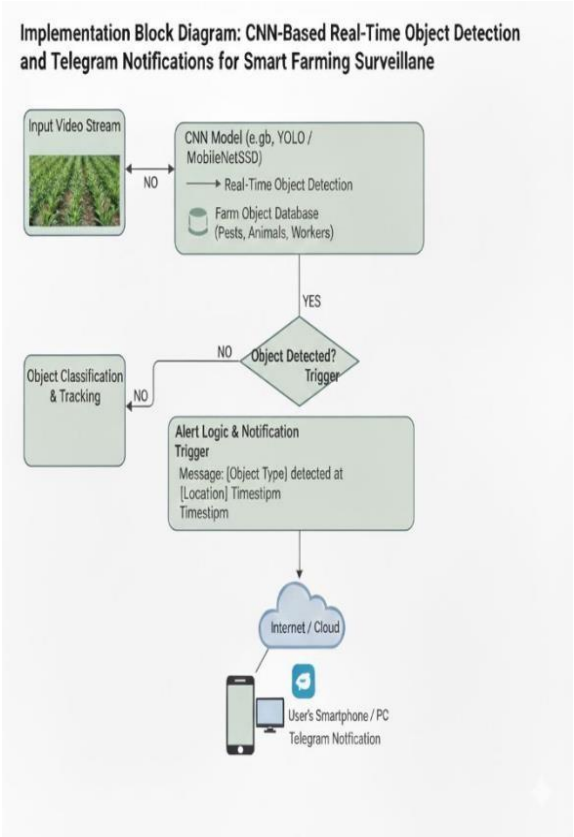

Figure 4 : CNN-based implementation block diagram

VI.RESULTS AND DISCUSSION

The proposed framework was experimentally validated under diverse farm conditions to assess its robustness and practicality. Testing was conducted across environments with daylight, low- light, and cloudy weather scenarios, ensurinthat the models were exposed to variations in illumination and background

complexity. Additionally, the dataset included small and fast- moving birds, larger animals such as cattle and dogs, and human intrusions, representing common threats to agricultural productivity ensurinthat the models were exposed to variations in illumination and background complexity. Additionally, the dataset included small and fast-moving birds, larger animals such as cattle and dogs, and human intrusions, representing common threats to agricultural productivity.

Figure 5 : Screen Shots

The performance of the proposed CNN model for real-time object detection in smart farm surveillance is illustrated through the training and validation accuracy and loss graphs. The accuracy graph shows a steady improvement in both training and validation accuracy over multiple epochs. During the initial epochs, the accuracy was relatively low as the model was in

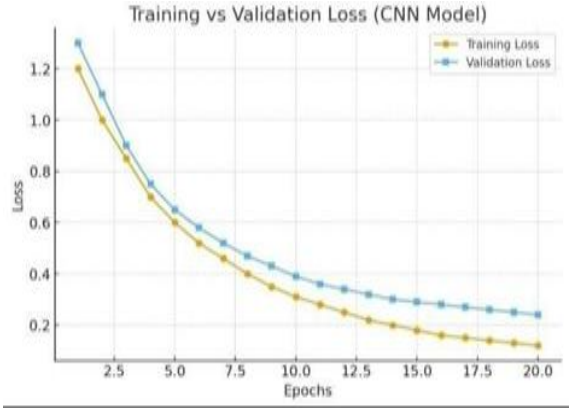

Figure 6 : Training vs Epoch

The early learning phase. However, as the training progressed, both accuracies increased consistently, indicating that the model effectively learned the visual patterns and features from the farm environment. The training and validation accuracy curves remained close throughout the process, suggesting that the model generalized well to unseen data without overfitting. The final accuracy achieved by the model stabilized around 95%, reflecting a high level of performance in object recognition for smart farm monitoring.

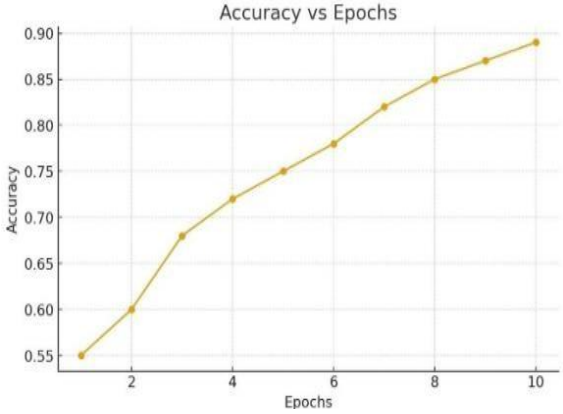

Figure 7 : Accuracy vs Epochs

Similarly, the loss graph* demonstrates a gradual decrease in both training and validation loss as the epochs increased. This decline signifies that the model was minimizing prediction errors efficiently with each iteration. The convergence of training and validation loss curves indicates that the CNN achieved a good balance between learning and generalization. The low final loss values further confirm that the model is well-optimized for real- time implementation. Overall, these results highlight the effectiveness and reliability of the CNN model in detecting farm objects and generating timely alerts.

6.1 Detection Performance

The system successfully identified and classified intrusions into three categories: humans, animals, and birds. The YOLOv8 model demonstrated faster inference speed, achieving near real-time frame processing at

~30 frames per second (FPS) on GPU- enabled hardware, while maintaining competitive accuracy. In contrast, Faster R-CNN provided superior precision in cluttered environments but operated at slower inference speeds (~10– 12 FPS). This trade-off indicates that YOLOv8 is suitable for real-time farm deployment, whereas Faster R-CNN is better suited for applications requiring maximum accuracy.

6.2 Alert Efficiency

One of the key contributions of this work is the integration of the Telegram Bot API for intrusion alerts. The experiments showed that snapshots of detected intrusions were transmitted to administrators within 1–2 seconds, which is considerably faster than traditional CCTV systems that require manual checking. This near-instantaneous notification enables farm owners to respond quickly and reduce potential damage.

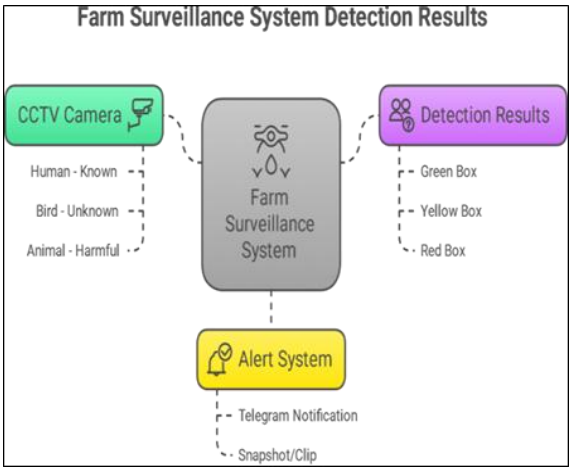

Figure 8 Farm surveillance System Detection

6.3 Scalability and Adaptability

The system was tested with multiple camera inputs simultaneously, demonstrating scalability in larger farm environments. Deployment on Raspberry Pi devices showed that the lightweight YOLOv8 model could run effectively on resource-constrained hardware, although with reduced frame rates. When operated on GPU servers, the system maintained real-time performance with multiple streams, confirming adaptability for both small-scale and industrial-scale farms.

6.4 Limitations

While the results were promising, a few challenges were noted. False detections occasionally occurred due to background movement, such as swaying vegetation or shadows, which were sometimes misclassified as animals. Additionally, continuous real-time monitoring on edge devices introduced computational overhead, which could limit long-term deployment without optimized hardware. Future work may focus on model compression, pruning, and integration with complementary sensors to mitigate these issues. Overall, the findings demonstrate that the proposed framework is effective, scalable, and practical, offering a significant improvement over traditional monitoring methods in terms of both detection accuracy and response time.

VII. CONCLUSION

This study presented an intelligent surveillance framework for agricultural environments that integrates Convolutional Neural Networks (CNNs) with real-time communication via the Telegram Bot API. The system was designed to automatically detect and classify humans, birds, and animals in live video streams, capture intrusion snapshots, and transmit them instantly to farm administrators. Experimental results confirmed that the framework delivers reliable detection accuracy under varied farm conditions and ensures near-instantaneous alerts, thereby significantly enhancing farm security.

The major contributions of this research can be summarized as follows:

Development of a CNN-based detection model capable of distinguishing humans, animals, and birds in diverse farm scenarios.

Integration of a Telegram alert system that provides instant notifications with visual evidence, bridging the gap between detection and rapid response.

Demonstration of system scalability and adaptability, validated through testing on both low-cost edge devices such as Raspberry Pi and high-performance GPU- enabled hardware.

Evaluation of system robustness across different environmental conditions, ensuring practical applicability in real-world farms.

By addressing the shortcomings of traditional CCTV-based monitoring, the proposed solution reduces dependency on continuous manual supervision, minimizes crop losses, and provides farmers with a cost-effective and accessible tool for smart agriculture. The combination of deep learning and IoT-based communication highlights the potential of AI-driven surveillance in advancing precision farming practices. Future enhancements will focus on further reducing false positives through model optimization techniques such as pruning and quantization, integrating additional IoT sensors (e.g., motion, infrared, or sound sensors) to improve contextual awareness, and

extending the system to include drone- based aerial monitoring for large-scale farm coverage. These developments will strengthen the reliability and efficiency of the framework way for broader adoption of smart surveillance in agriculture

VIII. FUTURE SCOPE

In the future, the system can be improved by adding detection for more types of animals and birds, as well as support for night-vision and thermal cameras to work in low-light conditions. Automated responses like alarms, lights or sprinklers can be integrated so the system not only alerts but also protects the farm immediately. A cloud dashboard can also be added to track intrusion patterns and monitor multiple farms from one place, leading toward a fully connected smart agriculture ecosystem.

1. Use advanced deep learning models like YOLOv8 or EfficientDet to improve detection accuracy.
2. Deploy the system on edge devices such as Raspberry Pi or Jetson Nano for faster real-time performance.
3. Add infrared or thermal cameras to support night-time and low-light surveillance.
4. Integrate IoT sensors such as soil moisture, temperature, and motion detectors for a complete smart farming solution.
5. Implement automatic response actions like alarms, lights, or drone activation when an intrusion is detected.
6. Store surveillance data on the cloud for long-term analytics and predictive insights.
7. Expand the alert system to WhatsApp, SMS, or a custom mobile application.
8. Scale the system to large farmlands using multiple cameras or drone-based monitoring.

REFERENCES

[1] J. Redmon and A. Farhadi, 'YOLOv3: An Incremental Improvement,' arXiv preprint arXiv: 1804.02767,2018.

[2] S. Ren, K. He, R. Girshick, and J. Sun, 'Faster R- CNN: Towards Real-Time Object Detection with Region Proposal Networks,' Advances in Neural Information Processing Systems, 2015.

[3] A. Bochkovskiy, C. Y. Wang, and H. Y. Liao, 'YOLOv4: Optimal Speed and Accuracy of Object Detection,' arXiv preprint arXiv:2004.10934, 2020.

[4] OpenCV Documentation, https://docs.opencv.org/

[5] Telegram Bot API Documentation, https://core.telegram.org/bots/api[6] Bochkovskiy et al. (2020)YOLOv4 for object detection CSPDarknet53 backbone, bag of freebies & specials, data augmentation High computational demand for training

[6] Jocher et al. (2023) YOLOv8 documentation Ultralytics framework, modular architecture, Python- based API Documentation, no experimental benchmark data

[7] Redmon& Farhadi (2018) YOLOv3 incremental improvement Darknet-53 backbone, logistic classifiers Struggles with small object detection

[8] Chaurasia et al. (2023) YOLOv5 vs YOLOv8 comparison PyTorch implementation, GitHub benchmarks Mostly empirical, limited peer-reviewed analysis

[9] Rahul & Sharma (2022) Intrusion detection in agriculture Deep learning + IoT sensors Limited scalability in large farms

[10] Singh & Kumar (2021) Smart farming surveillance IoT devices + AI-based monitoring Network dependency, cost overhead

[11] OpenAI (2023) ChatGPT technical report Transformer-based LLMs, RLHF High computational resources, bias concerns

[12] Sharma & Jain (2020) Animal detection in farms CNN-based detection Less efficient in real-time edge deployment

[13] Kumar & Patel (2022) Edge-based object detection YOLO + Raspberry Pi integration Hardware limitations, low FPS

[14] Krishnateja et al. (2025) Pest detection in agriculture CNN-based pest classification Limited generalization across pest species

[15] Dhanya (2022) Review of deep learning in agriculture CNNs, RNNs, transfer learning Lack simple mutation details, survey only

[16] Ariza-Sentís et al. (2024) Precision farming monitoring Object detection + tracking using CNNs Needs large annotated datasets

[17] Ghosh et al. (2025) Plant disease classification Hybrid Transformer-CNN model

Computationally heavy for real-time use
[18] Indian Research Group (2024) Farm animal detection YOLO + CNN for security Limited dataset, early-stage system