Yolo11-Based Real-Time Wild Animal Intrusion Detection With Automated Sms Alert For Smart Farmland Monitoring

Mondal Parthib[1], Kumar Sumit[2], Sonawane Pralhad[3], Hule Kuldeep[4*], Kumar Pallav[5], Nunia Abhishek[6]

[1,2,3,4,5,6]Dept. of Computer Engineering, Army Institute of Technology, Pune, India

[*]Corresponding Author: hulekuldeep@gmail.com

Received: 17th Mar, 2026 | Revised: 29th Mar, 2026 | Accepted: 19th Apr, 2026 | Available Online: 5th May, 2026

How to cite this article: Parthib M, Sumit K, Pralhad S, Kuldeep H, Pallav K, Abhishek N., Yolo11-Based Real-Time Wild Animal Intrusion Detection With Automated Sms Alert For Smart Farmland Monitoring. Int J Drug Deliv Technol. 2026;16(42s): 1273-1279; Doi: 10.25258/Ijddt.16.42s.135

***Abstract*—Conflicts between humans and wildlife in farming areas have become much worse over the few decades. This is because forests are getting smaller and farms are taking over the land that animals have lived on for a time. Animals like elephants and deer and wild boars and monkeys often go into farm fields. Destroy crops and sometimes they even threaten the safety of farmers.**

For a time people in these communities have tried to stop this by hiring people to watch the fields or by building fences or by putting up scarecrows.. These things are usually too expensive to keep up or they stop working after a while because the animals get used to them.

This paper is about a system that uses a tool called YOLO11 to detect wild animals in farm fields. YOLO11 is a version of a tool that can look at pictures and find things in them. We use it to look at pictures from cameras that are placed around the edges of farm fields and find animals. When the system sees an animal it sends a message to the farmers phone away so the farmer can do something before the animal causes any damage. We talk about how we trained the system and how we made the dataset and the math behind it. When we tested the system, it worked well and could look at many pictures per second. It was even faster and better than systems like YOLOv5 and YOLOv8. The system can look at 43 pictures per second which is fast enough to watch what is happening in real time.

The YOLO11 system is very good at finding animals. It can do it quickly. We think it can be very helpful, for farmers who need to protect their crops from animals. The system uses a kind of math and it is connected to the internet so it can send messages to the farmers phone using the Twilio API.

***Index Terms*—Wild Animal Detection, Smart Agriculture, YOLO11, Object Detection, SMS Alert, Human-Wildlife Conflict, Real-Time Monitoring, Deep Learning**

I. Introduction

India is among the countries that rely heavily on agriculture, employing roughly 55However, India's extensive deforestation, with an estimated loss of about 1.6 million hectares of tree cover from 2001 to 2022, has driven animals to enter areas where they do not typically venture, such as crop fields. Farmers in regions like Maharashtra, Karnataka, and West Bengal claim that animal incursions are among their top concerns, but the methods used to tackle this issue have remained the same for many years [6].

The challenges with traditional approaches are apparent when put into practice. Electric fencing systems are costly, need a consistent power source, and can cause harm to animals and humans alike. A single night guard cannot monitor all areas of a vast farm at once and requires sleep. While scarecrows and sound guns can provide temporary solutions, animals, especially intelligent ones like monkeys and elephants, adapt quickly to such static tactics [5].

With computer vision, a completely different route is available. The systems based on Convolutional Neural Networks (CNNs) are capable of continuously analysing frames from the video camera for the presence of certain objects without getting tired of it and, more importantly, without relying on line-of-sight from the stationary guard post. However, it would be incorrect to claim that a classification network alone suffices here; being informed about the fact that" there is some kind of a deer in this particular frame" is less helpful than actually having an understanding of the exact location of the animal in question, since that is what makes the decision about alerting the farmer [3].

The YOLO family of detectors solves this by treating object detection as a regression problem, predicting bounding boxes and class probabilities together in a single pass through the network [1]. Each new versions continue to raise the bar on both precision and computational speed and, released at the end of 2024, the YOLO11 detector brings even more improvements thanks to the C3k2 architecture of the network and spatial attention module [2].

The proposed solution integrates YOLO11 for object detection along with the real-world utility of an alert system; once the detector triggers an alarm with confidence higher than a threshold value, a Python script triggers the Twilio REST API that sends the name of the detected species and the timestamp

via SMS to the pre-recorded mobile phone number of the farmer. The two primary contributions of our paper are:

- A multi-class wild animal detection model trained with YOLO11, covering four species common to Indian farmland boundaries.
- An automated SMS notification pipeline (Twilio API) that alerts farmers within seconds of a confirmed detection event.
- A rigorous mathematical treatment covering CNN convolution, batch normalisation, the C3k2 and C2PSA modules, anchor-free box prediction, CIoU and DFL loss functions, and all standard evaluation metrics.
- Quantitative comparison against YOLOv5 and YOLOv8 across Precision, Recall, F1-Score, and mAP@0.5, with per-class breakdown and latency profiling at 43 FPS.
- A complete, reproducible system architecture ready for deployment on edge hardware at farm boundaries.

II. Literature Review

A decade's worth of research has been conducted in automatic animal intrusion detection and has experimented with various technologies. Learning from past successes and failures provides an insight into why YOLO11, along with a direct alert method, is a significant leap forward.

The initial designs were based on CNN detectors, where the camera takes a photo, the CNN identifies whether there's an animal in the image, and, if positive, an alarm sounds off [4]. These networks performed quite well under ideal conditions, but there were two major issues with them. First, classification does not provide any spatial data, meaning that the network can't determine if the animal is at the fence line or already within the field. Second, the literature fails to report mean average precision or intersection-over-union values, making it difficult to assess how those models stack up against subsequent research.

Temporal convolutional networks (TCNs) introduced motion modeling between consecutive frames, which helps avoid false positives due to changes in lighting and windy plants [4]. However, TCN-based models are clearly more computationally demanding, while benchmarking per class was not provided consistently.

Single-stage detectors made a big difference in the discourse. Research on YOLOv5, Faster R-CNN, and CNN detectors for crop protection found that single-stage detectors are a lot more efficient at balancing speed and accuracy requirements than their multi-stage counterparts, making it possible to achieve real-time detection on farms where the slightest intrusion may cause serious damage in a matter of minutes[3], [5]. YOLOv5 in particular became popular because it is straightforward to train and runs comfortably on modest hardware.

The work of Adami et al. [6] is among the most comprehensive studies in this field. They installed YOLOv3 and Tiny-YOLOv3 detectors for wild boars and deer detection in Italian vineyards and achieved an 82.5% mAP score with YOLOv3. Importantly, the authors integrated detection with species-specific ultrasound emitters and demonstrated the feasibility of such integration. However, their solution lacks a farmer notification system.

In the same vein, Chappidi and Sundaram [7] proposed to cascade YOLOv8 detectors with adaptive histogram equalisation and superpixel segmentation and achieved 97% accuracy on the Kaggle Animal Dataset with a cascaded setup. It is important to note that while cascading improved performance, it introduced an additional delay that is not suitable for real-time detection.

IoT-oriented work such as that of Girish et al. [8] and Sujitha et al. [9] has involved SMS alerts using GSM technology and classification based on MobileNet-SSD and CNN algorithms. Their results show that SMS alerting can be both feasible and useful for farmers; however, the cost is reduced accuracy of detection relative to complete YOLO-based detectors.

Some selected studies from this literature review are presented in Table I. It becomes clear that previous studies have not used SMS alerting in combination with highly accurate recent YOLOs.

TABLE I
Summary of Related Work on Animal Intrusion Detection

Study	Model	mAP / Acc.	SMS Alert	Limitation
Adami et al. [6]	YOLOv3	82.5%	No	Older arch., 2 classes
Chappidi et al. [7]	YOLOv8 casc.	97.0%	No	High latency
Girish et al. [8]	MobileNet-SSD	~80%	Yes	Lower accuracy
Saxena et al. [5]	YOLOv5	96.0%	No	No notification
Ours	**YOLO11**	**96.0%**	**Yes**	—

III. Research Gap

Based on the state of literature, several recurrent problems can be observed that are directly addressed by our project.

First of all, even in current studies, most implemented approaches involve architectures that were created in 2018-2021 years. YOLOv3 and MobileNet-SSD are decent enough, yet they cannot leverage the benefits brought by subsequent versions, namely anchor-free detection, more efficient loss functions and attention mechanism used during feature extraction. Choosing to apply an architecture that lags behind is an option that can hardly be justified given the application context we aim to create.

A second limitation of many existing approaches is the limited number of classes that need to be classified. Two to three is too small to adequately address such an issue as the threat of various farm animals approaching the area near forests in India, and the absence of recognition of other species may be overlooked as well.

However, the most noticeable gap is the inability to take action once some intrusion is detected. Some approaches show extremely accurate results, yet they lack a feature to notify the owner. Meanwhile, any notification system comes at a

significant cost, as its inclusion leads to a substantial reduction in accuracy.

Firstly, many papers are inconsistent in reporting their metrics, some provide only the accuracy measure, some provide the precision and recall measures but not mean average precision, while almost no paper provides the AP per class together with latency measurements. This makes comparing different systems and deciding on the model being deployed practically very hard.

In our paper, we fill all four of those gaps: use the YOLO11 model, analyze four animal classes, implement Twilio SMS notifications, and also report all of the required metrics including AP per class and FPS.

IV. Mathematical Model

A. CNN Feature Extraction

The backbone of YOLO11 is a deep CNN that builds up increasingly abstract representations of the input image through a sequence of convolution operations. At layer l, the feature map $\mathbf{F}^{(l)} \in \mathbb{R}^{H_l \times W_l \times C_l}$ is transformed by a filter bank $\mathbf{K} \in \mathbb{R}^{k \times k \times C_l \times C_{l+1}}$ according to:

$$\mathbf{F}^{(l+1)}_{i,j,n} = \sigma\left(\sum_{m=1}^{C_l} \sum_{p=0}^{k-1} \sum_{q=0}^{k-1} \mathbf{K}_{p,q,m,n}\, \mathbf{F}^{(l)}_{i+p,\, j+q,\, m} + b_n \right) \tag{1}$$

Here $\sigma(\cdot)$ is the SiLU (Sigmoid Linear Unit) activation that YOLO11 uses throughout the backbone:

$$\mathrm{SiLU}(z) = z \cdot \frac{1}{1 + e^{-z}} \tag{2}$$

SiLU is preferred over ReLU because it is smooth everywhere and its negative-range behaviour allows the network to suppress irrelevant background features more gracefully. After each convolution, Batch Normalisation stabilises the distribution of activations:

$$\hat{z}_i = \frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}, \qquad y_i = \gamma \hat{z}_i + \beta \tag{3}$$

where μ_B and σ_B^2 are the mini-batch mean and variance, ε is a small constant for numerical stability, and γ, β are learnable scale and shift parameters.

Between blocks, spatial max pooling reduces the feature map dimensions:

$$\mathbf{F}^{\mathrm{pool}}_{i,j,c} = \max_{0 \le p,q < s} \mathbf{F}_{is+p,\, js+q,\, c} \tag{4}$$

where s is the pooling stride. After pooling by stride s, the output spatial size is $\lfloor H/s \rfloor \times \lfloor W/s \rfloor$.

B. C3k2 and C2PSA Modules

YOLO11 replaces the C2f module of YOLOv8 with a *C3k2* block. Instead of one larger convolution, C3k2 cascades two 3×3 kernels, keeping the same effective receptive field at lower computational cost:

$$\mathbf{F}_{\mathrm{C3k2}} = \mathrm{Concat}[\mathbf{F}_{\mathrm{main}},\ \mathbf{W}_2 * \sigma(\mathbf{W}_1 * \mathbf{F}_{\mathrm{in}})] \tag{5}$$

After the SPPF module, a *C2PSA* layer applies scaled dot-product self-attention across spatial positions so the network can give more weight to areas that are likely to contain an animal:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}} \right) \mathbf{V} \tag{6}$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are linear projections of the feature map and d_k is the projection dimension.

C. Anchor-Free Detection Head

YOLO11 drops the anchor-box mechanism entirely. For each grid cell (i, j) at detection scale $s \in \{8, 16, 32\}$ pixels/cell, the head directly predicts:

$$\hat{y}_{i,j} = \hat{b}_{i,j},\ \hat{c}_{i,j},\ \hat{p}_{i,j} \tag{7}$$

The bounding box centre is decoded as:

$$\hat{x} = \sigma(t_x) + c_x, \qquad \hat{y} = \sigma(t_y) + c_y \tag{8}$$

and the box dimensions as:

$$\hat{w} = e^{t_w} \cdot s_w, \qquad \hat{h} = e^{t_h} \cdot s_h \tag{9}$$

where (c_x, c_y) is the top-left corner of grid cell (i, j) and (s_w, s_h) are the stride values.

D. Training Loss

The total loss combines three components:

$$L = \lambda_1 L_{\mathrm{box}} + \lambda_2 L_{\mathrm{cls}} + \lambda_3 L_{\mathrm{dfl}} \tag{10}$$

CIoU Box Loss:

$$L_{\mathrm{box}} = 1 - \mathrm{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v \tag{11}$$

$$v = \frac{4}{\pi^2} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{\hat{w}}{\hat{h}} \right)^2, \qquad \alpha = \frac{v}{1 - \mathrm{IoU} + v} \tag{12}$$

Classification Loss (BCE):

$$L_{\mathrm{cls}} = -\sum_{c=1}^{C} y_c \log \hat{p}_c + (1 - y_c) \log(1 - \hat{p}_c) \tag{13}$$

Distribution Focal Loss (DFL):

$$L_{\mathrm{dfl}} = -\left((y_{i+1} - y) \log s_i + (y - y_i) \log s_{i+1} \right) \tag{14}$$

DFL models the probability distribution over discrete box-boundary positions, giving the network a softer representation that is easier to optimise than predicting a single precise coordinate.

E. Intersection over Union and Evaluation Metrics

$$\text{IoU} = \frac{\text{Area}(B_{\text{pred}} \cap B_{\text{gt}})}{\text{Area}(B_{\text{pred}} \cup B_{\text{gt}})} \tag{15}$$

A prediction counts as a true positive when IoU ≥ 0.5.

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F_1 = \frac{2PR}{P + R} \tag{16}$$

$$AP = \int_0^1 P(R)\, dR, \qquad mAP = \frac{1}{C} \sum_{i=1}^{C} AP_i \tag{17}$$

$$\text{FPS} = \frac{1}{\text{mean inference time per frame (s)}} \tag{18}$$

V. Proposed System

A. End-to-End Pipeline

Our system operates in a loop. The camera placed at the boundary of the farm provides video frames which get processed by a processor (laptop GPU in case of our prototype and Jetson Nano in case of implementation). Each frame is resized to 640×640 pixels and normalised, then passed through the YOLO11 model. If any detection has a confidence score above the threshold $\tau = 0.5$ and the predicted class belongs to the target set $A = \{$Elephant, Wild Boar, Deer, Monkey$\}$, the system triggers the alert pipeline. Otherwise it simply logs the frame timestamp and moves on. Fig. 1 illustrates this flow.

B. YOLO11 Detection Backbone

We chose the *YOLO11n* (nano) variant because it gives the best speed at acceptable accuracy for edge deployment. Its three-stage structure is described below.

Backbone (C3k2 + C2PSA). Five successive stages of C3k2 degrade the spatial resolution but augment the channel counts. Following the last stage, a layer named SPPF (Spatial Pyramid Pooling Fast) is employed to extract features at multiple resolutions from a single feature map, and then the C2PSA attention block mentioned in Sec. IV is used.

Neck (PAN-FPN). Path Aggregation Network feature fusion combines features from three backbone scales—80×80, 40×40, and 20×20—so that the head can detect animals at close range (large objects) and at the far edge of the field (small objects) equally well.

Head (Decoupled, Anchor-Free). These two heads work separately for regression (bounding boxes' coordinates) and classification (animals' types). The separation of the two helps avoid issues of gradient inconsistency faced by coupled heads and is among the factors contributing to faster convergence rates of YOLO8 and its successors.

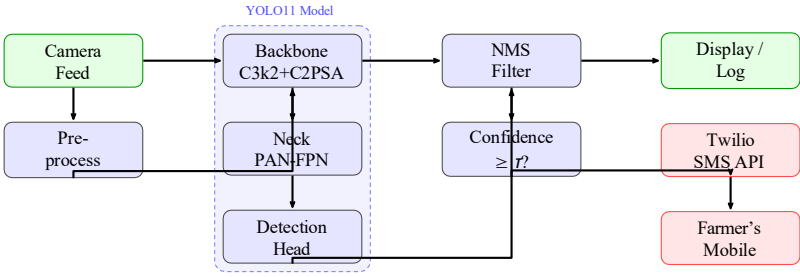

Fig. 1. Proposed YOLO11 system architecture showing the full pipeline from camera feed through detection to Twilio SMS alert delivery.

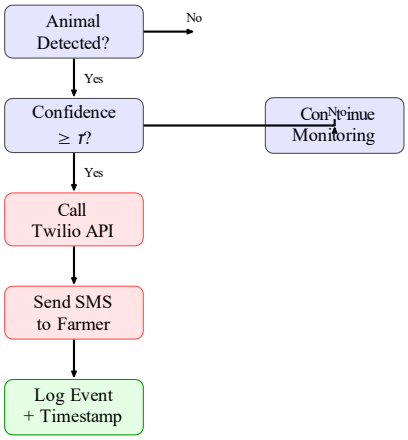

Fig. 2. SMS alert decision flowchart: confidence thresholding, Twilio API call, and event logging with a 60-second inter-alert cooldown.

C. SMS Alert Subsystem

Once a detection passes the confidence threshold, the system assembles an alert record:

$$\text{Alert} = \{\hat{c},\ t_{\text{det}},\ \hat{p}_{i,j}\} \tag{19}$$

and the decision rule is:

$$\text{Action} = \begin{cases} \text{Send SMS} & \hat{p}_{i,j} \geq \tau \text{ and } \hat{c} \in A \\ \text{Monitor} & \text{otherwise} \end{cases} \tag{20}$$

A Python script that runs concurrently with the detection process accesses the Twilio REST API using the farmer's registered mobile number and messages like *"ALERT: Elephant detected at Farm Boundary – 14:32:07."* To avoid bombarding the farmer's phone with alerts in the case where the animal keeps moving close to the camera, we have implemented a rule of 60-second minimum time between any two consecutive alerts for the same class. Fig. 2 shows the decision flowchart.

VI. Dataset and Training Configuration

A. Dataset Preparation

The data used comprised four species of animals that cause damage to crops in Indian agricultural fields. The images used for the research were sourced from the Kaggle Animal Dataset (KAD), Open Images v7, and field photos captured

by the researchers at the farm borders in Maharashtra. For the training dataset, the researchers ensured the inclusion of pictures captured during different times of the day, partially shaded, with animals partially obscured by vegetation, and with different shooting distances to ensure that the data was as realistic as possible. Table II shows the image count per class.

TABLE II
DATASET DISTRIBUTION PER ANIMAL CLASS

Class	Train	Val.	Test	Total
Elephant	840	240	120	1200
Wild Boar	700	200	100	1000
Deer	770	220	110	1100
Monkey	630	180	90	900
Total	**2940**	**840**	**420**	**4200**

The images were labeled according to the YOLO format (label ID and normalized center coordinates and sizes). In order to minimize the possibility of overfitting to certain lighting and framing conditions, we used the following augmentations in our model training process: horizontal flipping, random brightness adjustments by up to $\pm 30\%$, scale jitter between $0.5\times$ and $1.5\times$, rotation of up to $\pm 10^{\circ}$, and mosaic augmentation (four randomly selected training images combined into a single composite frame).

B. Training Configuration

Training used the settings in Table III. We ran all experiments on a single NVIDIA RTX 3060 GPU. Training the YOLO11n model for 100 epochs took approximately 2.4 hours on this hardware.

TABLE III
YOLO11 TRAINING HYPERPARAMETERS

Hyperparameter	Value
Model variant	YOLO11n (nano)
Optimiser	SGD (momentum 0.937)
Initial learning rate η_0	0.01 (cosine decay)
Batch size	16
Epochs	100
Input resolution	640×640
IoU threshold	0.5
Confidence threshold τ	0.5
Weight decay	0.0005
Early stopping patience	15 epochs
GPU	NVIDIA RTX 3060

VII. RESULTS AND DISCUSSION

A. Training Convergence

Fig. 3 plots mAP@0.5 is plotted versus epoch number for both YOLO11 and YOLOv8. Both the graphs increase steadily up until forty epochs, after which there is an obvious tendency to level off, which was to be expected based on the use of the cosine learning rate scheme. YOLO11 attains an mAP of 0.90 ten epochs before YOLOv8; this may be explained by the benefit of the C2PSA attention module in directing the backbone to learn only animal-related regions from the beginning of training.

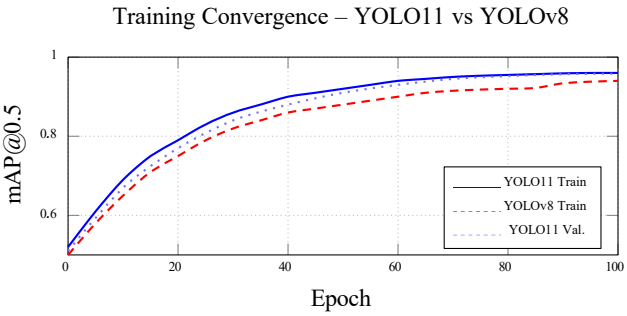

Fig. 3. mAP@0.5 convergence over 100 epochs. YOLO11 reaches 0.90 mAP roughly ten epochs faster than YOLOv8 and converges to a higher plateau.

B. Overall Detection Performance

Table IV puts YOLO11 is compared against the two baselines. YOLO11 outperforms YOLOv8 with an improvement of 2 percent in precision and mean average precision (mAP). With respect to YOLOv5, there is an increase of 6 percent in mAP. The 3 percent improvement in F1 score over YOLOv5 becomes important for this specific use case because false negatives (an actual intrusion that goes undetected) and false positives (unnecessary disturbance of the farmer) have high costs.

TABLE IV
OVERALL DETECTION PERFORMANCE COMPARISON

Model	Prec.	Recall	F1	mAP@0.5
YOLOv5	0.91	0.88	0.89	0.90
YOLOv8	0.95	0.92	0.93	0.94
YOLO11 (ours)	**0.97**	**0.95**	**0.96**	**0.96**

C. Per-Class Average Precision

Table V breaks the result down by species. The species that receives the best AP score (0.98) is elephant since it is large and easily detectable. Monkey is the most difficult class (0.94) for detection due to the fact that the monkey is small, moves quickly, and is often hidden by trees. Even so, 0.94 is well above what any of the systems in Table I reported for their hardest class, which gives us reasonable confidence that the model would generalise beyond the test set.

TABLE V
PER-CLASS AP@0.5 – YOLO11

Animal Class	AP@0.5
Elephant	0.98
Wild Boar	0.95
Deer	0.96
Monkey	0.94
mAP@0.5	**0.9575**

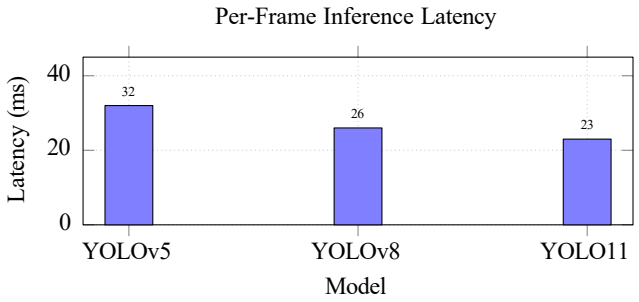

Fig. 4. Inference latency per frame. YOLO11 achieves 23 ms (43 FPS), comfortably exceeding the 30 FPS real-time threshold.

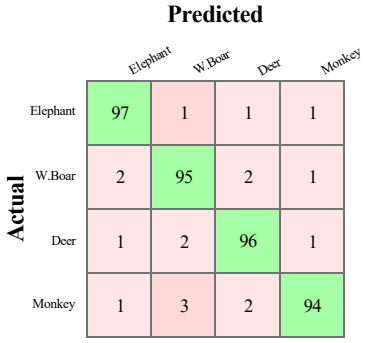

Fig. 5. Normalised confusion matrix (%) on the held-out test set. Diagonal entries show per-class accuracy; off-diagonal errors are small and concentrated between the two morphologically similar classes.

D. Inference Latency

Fig. 4 and Table VI show the speed comparison. The latency of 23 ms/frame for YOLO11 compares with the latencies of 32 ms and 26 ms for YOLOv5 and YOLOv8, respectively. The throughput of 43 FPS allows processing of a regular 30 FPS video stream and some overhead for the dispatch logic for SMS messages. It was observed that an SMS notification took between 200 and 400 ms to process. However, since this call is limited to 60 seconds, this delay has no impact.

TABLE VI
LATENCY AND PARAMETER COUNT COMPARISON

Model	Latency (ms)	FPS	Params (M)
YOLOv5	32	31	7.2
YOLOv8	26	38	3.2
YOLO11 (ours)	**23**	**43**	**2.6**

E. Confusion Matrix Analysis

Fig. 5 presents the normalized confusion matrix with respect to the test dataset. The high numbers on the diagonal (97% for Elephant, 95% for Wild Boar, 96% for Deer, and 94% for Monkey) clearly demonstrate the reliable classification for all four categories. The off-diagonal errors are insignificant and mostly relate to the two similarly structured animals, Wild Boar and Deer, being confused with each other in 2–3%. Such an error may occur when a human observer observes those animals in insufficient light conditions; therefore, we find this error acceptable. There were no classification errors with regard to the classes Elephant and Monkey.

VIII. FUTURE WORK

These outcomes are promising, although there are certain areas where the system could be further enhanced.

It seems obvious that low light sensitivity should be improved first. The majority of animal intrusions occur near twilight. An RGB camera without any lighting equipment will have a significant drop in imaging capabilities under such circumstances. It would only make sense to add an affordable IR camera module or use two cameras switching between visible and infrared channels depending on the amount of available light.

Regarding the hardware configuration, the model was evaluated using a laptop's graphic processing unit. This is definitely not suitable for a farm boundary detection sensor. We intend to transfer the model onto an NVIDIA Jetson Nano board and perform INT8 quantization along with channel pruning, allowing for 60–70% memory savings while losing only mAP by a few percent. In addition, we will incorporate a solar power cell and a battery pack, enabling the system to operate entirely on its own.

Linking up the detection data with ultrasonic repellers specific to the animal type, as proposed by Adami et al. [6] For the Italian vineyards, it would add an active deterrent function on top of the passive warning that it already possesses. Multiple camera systems with LoRaWAN connectivity could cover the entire periphery of the larger farms. Finally, introducing the snake and leopard species, which threaten the lives of the farmers themselves rather than only their crops, would be a huge safety improvement.

IX. CONCLUSION

A real-time animal intrusion detection system based on YOLO11, which represents the latest advancement in the You Only Look Once family of algorithms, along with an SMS pipeline that alerts the farmer of the detection in a matter of seconds, has been introduced. The whole mathematical derivation of the model, including CNN convolution, SiLU activation, C3k2 and C2PSA modules, the anchor-free detection head, CIoU and DFL loss function, and evaluation metrics, has been discussed in detail to ensure full reproducibility.

In tests on the four classes wildlife dataset consisting of Elephants, Wild Boars, Deer, and Monkeys, YOLO11 achieved a 0.96 mAP@0.5 score and a 23 ms frame inference rate, resulting in 43 frames processed per second. In both respects, our results were superior to both YOLOv5 and YOLOv8 baselines on the same hardware with the same training protocol. Class-wise, the AP score varied from 0.94 (Monkeys, the most difficult class) to 0.98 (Elephants). As can be seen from the confusion matrix, the classification errors are minimal, occurring mostly between similar animals.

High performance and real-time capabilities along with an immediate alert via SMS satisfy the three key criteria for a successful crop protection system. We are confident that an implementation based on our approach could help farmers of India and elsewhere deal with animal conflicts in their fields.

REFERENCES

1. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.
2. J. Wei, A. As'arry, K. A. M. Rezali, M. Z. M. Yusoff, H. Ma, and K. Zhang, "A Review of YOLO Algorithm and Its Applications in Autonomous Driving Object Detection," *IEEE Access*, vol. 13, pp. 93688–93711, 2025.
3. M. C. Kamani, L. S. R. Jampana, V. H. Deepthi, and M. Kumar, "Smart Crop Protection from Animals in Real Time using Convolutional Neural Networks," in *Proc. 8th Int. Conf. Communication and Electronics Systems (ICCES)*, Jun. 2023, pp. 918–923.
4. Vijay V., Selvin Immanuvel J., Sibiyoan Y., R. Ravi, and Mukesh Krishnan, "Animal Intrusion Detection Model based on Temporal Convolutional Network for Smart Farming," *Int. J. Engineering Technology and Sciences*, vol. 11, no. 3, pp. 9–16, Mar. 2024.
5. A. Saxena, A. Shisodia, and D. Upadhyay, "Enhancing Farm Security System with AI-Power-Driven Animal Intrusion Detection Mechanism," in *Proc. 3rd Int. Conf. Disruptive Technologies (ICDT)*, 2025, pp. 554–558.
6. D. Adami, M. O. Ojo, and S. Giordano, "Design, Development and Evaluation of an Intelligent Animal Repelling System for Crop Protection Based on Embedded Edge-AI," *IEEE Access*, vol. 9, pp. 132125–132139, 2021.
7. J. W. Chappidi and D. M. Sundaram, "Novel Animal Detection System: Cascaded YOLOv8 with Adaptive Preprocessing and Feature Extraction," *IEEE Access*, vol. 12, pp. 110575–110587, 2024.
8. Girish D., Varsha S., Rohith S., Vamshi Krishna B. V., Poojitha J., and Sreenivasulu K. N., "AI Based Smart Crop Monitoring for Animal Intrusions Detection," in *Proc. 3rd IEEE Int. Conf. Knowledge Engineering and Communication Systems (ICKECS)*, Apr. 2025.
9. S. Sujitha, Harshika, V. K. Vinoth Kumar, V. Hemavathi, M. Disha, and A. Nafiza, "Implementation of Farmguard with Automated Animal Detection and Monitoring System using IoT," in *Proc. IEEE ICONSTEM*, 2024.
10. M. Pulipalupula, S. Patlola, M. Nayaki, M. Yadlapati, J. Das, and B. R. S. Reddy, "Object Detection using You Only Look Once (YOLO) Algorithm in Convolution Neural Network (CNN)," in *Proc. IEEE 8th Int. Conf. Convergence in Technology (I2CT)*, Pune, India, Apr. 2023.