International Journal of Research in Engineering, Science and Management
Volume 9, Issue 8, August 2026
www.ijresm.com | E-ISSN: 2581-5792 | RESAIM Publishing | www.resaim.com

Realtime Animal Intrusion Detection System Using YOLOv5

G. T. Shayana Shree[1*], N. B. Shankar[1], Basavaraj S. Pol[1]
[1]*Department of Computer Science and Engineering, R. L. Jalappa Institute of Technology, Kodigehalli, India*

***Abstract*: Farmers working land near forests or open grazing tracts deal with a problem that rarely makes it into agricultural planning discussions: animals wandering into their fields at night and destroying weeks of work in minutes. Fences get pushed over, scarecrows get ignored, and a farmer cannot realistically stay awake every night watching the boundary. This paper walks through a system built to take over that watching job. It pairs a small camera and motion sensor sitting at the field edge with a compact deep learning model that can tell, in under two seconds, whether something moving near the crop is actually an animal worth worrying about. Once it decides yes, it fires off a text message or app alert with a photo attached, so the farmer knows exactly what showed up and where. The hardware is built around a Raspberry Pi, a PIR sensor that keeps the unit asleep until something moves, a YOLO-based detector trained on wild boar, nilgai, deer, and stray cattle, and a GSM or Wi-Fi link for sending the alert itself. Testing across a labelled image set and a two-week trial run at a field boundary produced detection accuracy above ninety percent for the animals it was trained on, with the whole detect-and-alert cycle finishing in under two seconds when Wi-Fi was available. The paper lays out how the pieces fit together, why certain design trade-offs were made to keep cost and power draw low, and what would need to change to scale this up more cameras, solar charging, and eventually some kind of automated deterrent that fires before the farmer even reaches the field.**

***Keywords*: animal intrusion detection, deep learning, YOLO, convolutional neural network, IoT, real-time monitoring, smart agriculture, embedded systems, wildlife-human conflict.**

1. Introduction

For farmers working land next to a forest edge or an open grazing corridor, one of the biggest threats to a harvest has nothing to do with rainfall or pests, it is a wild boar, a nilgai, or a stray herd of cattle wandering in overnight and flattening a field that took months to grow. When it happens often enough, some farmers stop trying to coexist with the wildlife around them and start reacting with hostility, which is bad for the animals and, longer term, bad for the surrounding ecosystem too. The usual defences thorn fences, tin-can noise-makers, someone sitting up all night with a torch ask a lot of a person and still miss plenty of intrusions, especially the ones that happen at 3 a.m. on the far side of a large plot.

What has changed recently is cost. Single-board computers, small camera modules, and pretrained detection models are now cheap enough that automated field monitoring is no longer something only a research institute or a large estate could afford. A device that sits quietly at the boundary, tells the difference between a genuine animal and a branch swaying in the wind, and pings the farmer's phone the moment it sees something real, could cut reaction time from hours to seconds. That is the system this paper describes built from off-the-shelf parts, trained on a retrainable model rather than a fixed one, and kept cheap enough that a single farmer could realistically buy and run it, not just a cooperative or a government pilot programme.

The rest of the paper is laid out as follows. Section 2 looks at what has already been tried in this space. Section 3 sketches the system at a high level before Section 4 gets into the actual hardware and software. Section 5 covers how detection and alerting work under the hood. Section 6 reports what happened when the system was tested, Section 7 is an honest look at what it does well and where it still falls short, and Section 8 wraps up with where this could go next.

2. Literature Survey

The earliest attempts at this problem leaned on PIR motion sensors wired to a simple alarm. Cheap and easy to install, sure, but a strong gust of wind or a stray cat sets them off just as easily as a wild boar does, and the sensor itself has no way of telling the farmer which one it was. That meant walking out to check every single alert, which defeats a good part of the purpose.

As computer vision got better, camera traps paired with image classifiers became the next step, and they were genuinely good at telling one species from another. The catch is that most of this work was built for ecological surveys and population counts, not for stopping a boar mid-raid. Someone still had to sit down and go through the images, often a day or more after they were captured, by which point the damage was already done.

Things shifted with the arrival of single-stage detectors like YOLO, which made it realistic to run detection directly on modest hardware and get an answer almost instantly rather than waiting on a server round trip. Smaller variants YOLO-tiny, MobileNet-based detectors pushed that even further down to devices barely more powerful than a microcontroller. Around the same time, separate work on IoT-based farm monitoring

*Corresponding author: shayanashree96@gmail.com

showed that pairing a sensor trigger with a GSM or Wi-Fi alert was entirely workable. What is harder to find is research that treats detection and alerting as one connected pipeline rather than two separate problems solved in isolation, tested under real field conditions where power is unreliable, lighting is inconsistent, and the network signal comes and goes. That is really the gap this paper is trying to close building the model, the trigger logic, and the alert channel together as one system instead of stitching together parts that were each designed on their own.

3. Proposed System

The design rests on one simple idea: stay asleep and save power until something actually moves, then wake up just enough to check whether that something is an animal worth an alert, and only bother the farmer once that's confirmed. This two-step approach sidesteps the two problems that simpler designs run into a raw motion sensor that cries wolf constantly, or a camera-plus-model combo that burns through battery by running non-stop.

In practice, the pipeline breaks down into four steps. A PIR sensor at the boundary picks up movement and wakes the unit out of its low-power idle state. The camera then grabs a short burst of frames, which get handed to the detection model running on the device itself no cloud round trip needed. The model figures out whether whatever it's looking at belongs to one of the animal classes it was trained on, along with a bounding box and a confidence score. If that confidence clears a set threshold, the system stamps the image with a time and location and sends it off to the farmer's phone, either as an SMS through the GSM module or a push notification over Wi-Fi, whichever the deployment site actually has available.

4. System Architecture

A. Hardware Components

At the centre of the edge unit sits a Raspberry Pi 4 Model B enough headroom to run a quantised YOLO model at a usable frame rate without needing anything bulkier or more power-hungry. Vision comes from either the standard Raspberry Pi Camera Module or a USB night-vision camera with built-in infrared illumination, which matters since most intrusions during the field trial happened after dark. A PIR sensor handles the wake-up job, keeping the rest of the unit powered down between events. Where the site has no dependable Wi-Fi, a SIM800L GSM module sends alerts as plain SMS; where Wi-Fi is available, the unit instead pushes a notification over MQTT to a companion phone app. Power comes from a 12V battery topped up by a small solar panel, which matters given that most of these field boundaries sit well away from any grid connection.

B. Software Components

On the software side, the Pi runs a stripped-down Linux build suited to single-board computers. Detection itself uses a YOLOv5-nano model, quantised down so it runs comfortably on the Pi's CPU alone, with a USB neural compute stick as an optional speed boost where one is available. A small Python service ties everything together listening for the PIR interrupt, triggering the camera, running inference, and firing off the alert. Every event that gets flagged, along with its frame, timestamp, and confidence score, gets logged to a local SQLite database and synced to a cloud dashboard whenever a connection is available, giving the farmer a running record of when and where intrusions tend to happen.

C. Data Flow

Table 1 lays out how data moves from the moment motion is first sensed through to the alert landing on the farmer's phone, along with roughly how much time each stage took during testing.

Table 1
Event-to-alert data flow

Stage	Function	Typical Time
Trigger	PIR sensor detects motion, wakes edge unit	< 0.1 s
Capture	Camera records a short frame burst	~0.3 s
Inference	YOLO model detects and classifies object	~0.7 s
Alert	SMS / push notification sent with snapshot	~0.5–1 s

5. Methodology

A. Dataset Preparation

The image set used for training combined publicly available camera-trap photos with additional pictures taken in the field of the species farmers in the area actually complained about — wild boar, nilgai, deer, and stray cattle. Each image was hand-labelled with bounding boxes using an open-source annotation tool. Because most intrusions happen around dusk or dawn rather than midday, the set was also augmented with random brightness shifts, horizontal flips, and synthetic low-light colour changes, so the model wouldn't fall apart the moment lighting got dim.

B. Model Training

Rather than training from scratch, the detector started from weights pretrained on a general object-detection benchmark and was then fine-tuned on the custom animal dataset. Standard augmentation, a cosine learning-rate schedule, and early stopping based on validation loss kept the model from overfitting to what is, admittedly, a fairly small custom dataset. Once training settled, the model was exported in a quantised integer format so it would fit comfortably in memory and run fast enough on the Pi's CPU alone.

C. Detection and Alerting Logic

Once the PIR sensor fires, the camera captures a short burst of frames rather than a single shot, and each one gets run through the model. Detections are aggregated across that whole burst instead of trusting any one frame, which cuts down on false positives caused by something like a shadow flickering across the lens for an instant. An alert only goes out once at least two frames in a row agree on the same animal class above the confidence threshold and when it does, the highest-confidence frame from the burst gets attached, so the farmer can look at the photo and confirm it for themselves rather than taking the system's word for it.

6. Results and Discussion

Testing happened in two parts: first against a held-out set of images covering all four animal classes, and then through actual field trials over two weeks at the edge of a test plot, running through both day and night conditions. Table 2 shows what the held-out test set produced.

Table 2
Detection performance by species class

Species Class	Precision	Recall	F1-Score
Wild boar	0.94	0.91	0.92
Nilgai	0.90	0.88	0.89
Deer	0.93	0.90	0.91
Stray cattle	0.96	0.95	0.95
Overall (mean)	**0.93**	**0.91**	**0.92**

Over the two-week trial, the system caught the overwhelming majority of genuine intrusions. The handful it missed were mostly low-light cases where the animal was partly hidden behind vegetation not a huge surprise given how the model was trained. False positives, meanwhile, dropped sharply once the two-frame confirmation rule went in; an earlier single-frame version tested during development threw far more nuisance alerts. Latency stayed under two seconds end-to-end over Wi-Fi, and closer to six seconds over SMS though that delay comes almost entirely from the carrier network, not from anything happening on the device itself.

What this suggests is that a quantised model running locally can hold its own against a cloud-based setup on accuracy, while sidestepping the latency and connectivity headaches a cloud round trip brings with it. The two-frame rule was a small, almost obvious tweak, but it made a real difference to how often the system cried wolf and that matters more than it might sound, because a farmer who gets ignored by false alerts a few too many times stops trusting the device altogether.

7. Advantages and Limitations

A. Advantages

- Low per-unit cost relative to commercial wildlife-monitoring products, making it accessible to individual farmers rather than only large estates.
- On-device inference removes dependence on a constant internet connection for detection, with connectivity needed only at the alerting stage.
- Staged trigger-then-confirm design keeps power consumption low enough for solar-battery operation.
- The two-frame confirmation logic meaningfully reduces false alerts without adding perceptible delay for the user.

B. Limitations

- Detection accuracy degrades for species not represented in the training dataset, requiring periodic retraining as new intruder types are reported.
- Dense fog, heavy rain, or very low ambient light can reduce camera image quality enough to affect detection confidence.
- SMS-based alerting depends on cellular network coverage, which can be inconsistent in remote farmland.

8. Conclusion and Future Scope

What this comes down to is a fairly simple pitch: a motion-triggered camera and a small on-device model can catch animal intrusions and get word to a farmer within seconds, without needing constant internet or a person staying up all night. The numbers back it up over ninety percent detection accuracy on the trained species, under two seconds of latency over Wi-Fi, and it all runs on a solar-charged battery cheap enough for one farm to justify buying on its own.

There's plenty left to improve, though. Widening species coverage by pulling in images from other farmers who deploy the system would help it generalize better than a lab-collected dataset ever could. Coordinating multiple cameras would help with field boundaries that aren't a simple straight line. And there's an obvious next step in pairing the alert with some kind of humane deterrent a directional ultrasonic emitter or a light burst that fires automatically the moment an animal is confirmed, buying time before anyone physically gets there. Feeding all of this into a shared regional dashboard could also let agricultural extension offices spot intrusion hotspots across an area and plan things like targeted fencing or corridor management at a larger scale than any one farm could manage alone.

References

[1] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[4] M. Norouzzadeh *et al.*, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proc. Natl. Acad. Sci. U.S.A.*, vol. 115, no. 25, pp. E5716–E5725, Jun. 2018.

[5] N. Patel and D. Mehta, "Design of a GSM based alert system for agricultural field security," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2, pp. 4501–4505, Jul. 2019.

[6] P. Kumar and A. Singh, "A review of wildlife monitoring techniques using embedded systems and machine learning," *Int. J. Comput. Appl.*, vol. 178, no. 30, pp. 12–18, May 2019.

[7] Raspberry Pi Foundation, "Raspberry Pi 4 Model B — Product documentation," 2026. [Online]. Available: https://www.raspberrypi.com/documentation/

[8] S. Verma and R. Sharma, "IoT based smart farming system for crop protection from animals," *Int. J. Eng. Res. Technol.*, vol. 9, no. 4, pp. 245–248, Apr. 2020.

[9] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.

[10] Ultralytics, "YOLOv5 — PyTorch implementation of the YOLO object detector," 2021. [Online]. Available: https://github.com/ultralytics/yolov5