ICICSET-2026
https://doi.org/10.65091/icicset.v3i1.105
Crossref

Edge-AI Detection and Adaptive Non-Lethal Deterrence of Rhesus Macaques using YOLO26n

Munna Pajiyar
Nepal College of Information Technology, (Affiliated to Pokhara University),
Lalitpur, Nepal
munna.221452@ncit.edu.np

Bishal Neupane
Nepal College of Information Technology, (Affiliated to Pokhara University),
Lalitpur, Nepal
bishal.221419@ncit.edu.np

Neesha Basnet
Nepal College of Information Technology, (Affiliated to Pokhara University),
Lalitpur, Nepal
neesha.221427@ncit.edu.np

Samikshya Ghimire
Nepal College of Information Technology, (Affiliated to Pokhara University),
Lalitpur, Nepal
samikshya.221436@ncit.edu.np

Roshan Chitrakar
Nepal College of Information Technology, (Affiliated to Pokhara University),
Lalitpur, Nepal
roshanchi@ncit.edu.np

***Abstract*-Rhesus macaque (Macaca mulatta) crop raiding is a widespread and economically damaging form of human-wildlife conflict in Nepal, with field studies reporting average crop losses of 24.62% and raiding frequency peaking at 45% in spring and 39% during the monsoon. Traditional mitigation manual guarding, scarecrows, and static acoustic devices provides no automated early warning and is prone to rapid habituation, leaving smallholder farmers with limited recourse. This paper presents the design, implementation, and bench-level evaluation of an edge-based AI-IoT system that detects macaques in real time and responds with an adaptive, non-lethal audio deterrent while logging and notifying farmers of every event. A YOLO26n object detector, transfer-learned from COCO-pretrained weights, was trained on a curated single-class dataset of 10,886 images (8,229 macaque images and 2,657 negative samples of humans, cattle, scarecrows, and empty fields, added after bench testing exposed human false positives) and exported to ONNX for CPU inference on a Raspberry Pi 4. A confidence-threshold decision module (≥ 0.60) triggers a Bluetooth-driven PAM8403/TS-250 audio actuator with rotating clips designed to resist habituation, alongside Telegram and Flutter notifications and an SQLite event log. The retrained model achieved 89.84% precision, 80.90% recall, 89.9% mAP@50, and 65.75% mAP@50–95, with an average edge inference latency of approximately 320 ms. Indoor bench testing confirmed correct end-to-end operation across detection, deterrence, notification, and logging; field deployment and quantitative field validation remain the principal direction for future work.**

Keywords-Edge AI, Object Detection, YOLO, ONNX, Non-Lethal Deterrence, IoT, Precision Agriculture

I. INTRODUCTION

Agriculture underpins the livelihood of a large share of Nepal's rural population, yet farmers face persistent losses from human-wildlife conflict, most notably crop raiding by Rhesus macaques (Macaca mulatta) [1]. Field studies along the Budhi Gandaki corridor and surrounding settlements report average crop losses of 24.62%, with raiding intensity peaking at 45% in spring and 39% during the monsoon, and maize identified as the most heavily targeted crop [1]. Beyond direct crop loss, sustained conflict has been linked to reduced farm income and rural-to-semi-urban migration among affected households [2].

The cultural and religious status of monkeys in Nepali and wider South Asian tradition constrains aggressive or lethal control measures, so farmers are largely limited to non-lethal strategies such as manual guarding, scarecrows, and passive noisemakers [1]. These measures share a common weakness: they are static or reactive, provide no automated early warning, and as behavioral research on acoustic deterrents consistently shows lose effectiveness once the target animal habituates to a predictable, repeated stimulus [4]. Rhesus macaques in particular exhibit high cognitive flexibility and social learning capacity, which allows troops to quickly identify and disregard non-threatening deterrents [7].

Advances in lightweight convolutional object detectors and low-cost embedded computing have made it feasible to pair automated, species-level detection with a triggered physical response at the point of intrusion, rather than relying on a farmer's continuous presence. YOLO-family detectors deployed on single-board computers have been used to detect stray animals [5], to combine detection with predator-sound deterrence for durian plantations [6], and to drive general agricultural-field surveillance and animal-intrusion alerts [8], [9], [13], [14]. However, most of this prior work is either multi-species and generic rather than tuned to a single, locally dominant species [13], targets a different geography and crop system [6], [9], or evaluates detection accuracy in isolation without carrying the result through to an integrated, farmer-facing deterrent-and-notification loop [8]. No system in the reviewed literature was designed for, or evaluated against, the Nepal Rhesus-macaque smallholder context specifically.

This paper addresses that gap with an edge-deployed, species-specific detection-to-deterrence pipeline: a YOLO26n detector trained specifically for Rhesus macaques, deployed via ONNX Runtime on a Raspberry Pi 4, coupled to a confidence-gated, habituation-resistant audio deterrent and a farmer-facing notification and logging layer.

The research objectives were to (i) design and train a lightweight, high-accuracy computer-vision model capable of detecting Rhesus macaque intrusions from camera imagery; (ii) deploy the trained model on a resource-constrained edge device and verify reliable on-device inference; and (iii) integrate the edge-deployed detector with

an automated deterrent, farmer-notification, and event-logging subsystem, validating the complete pipeline end-to-end.

The contributions of this work are:

1) A curated, single-class Rhesus macaque dataset incorporating negative examples to address false positive detection during bench testing. (Section IV).
2) A lightweight YOLO26n-to-ONNX edge -inference pipeline validated on Raspberry Pi 4 CPU hardware.
3) A confidence-gated, habituation-resistant non -lethal audio deterrence mechanism grounded in the acoustic -deterrent literature [4], [7].
4) An integrated farmer notification and event -logging layer (Telegram, Flutter, SQLite, Flask) bench -validated end-to-end. Field deployment and quantitative field validation remain the immediate next step and are discussed in Section VII.

II. RELATED WORK

Rai and Rai [1] quantified crop-raiding patterns of Rhesus macaques along the Budhi Gandaki corridor, finding average losses of 24.62%, seasonal peaks in spring (45%) and monsoon (39%), and identifying macaques among Nepal's top ten crop-raiding species with no systematic national response mechanism in place. Their work establishes the scale and seasonality of the problem but does not propose or evaluate a technological countermeasure. Lamichhane and Lamichhane [2] documented the socioeconomic consequences in Nuwakot district, linking chronic crop damage to reduced income and rural-to-semi-urban migration, confirming that the problem is not merely agronomic but also a driver of demographic change; again, no mitigation technology is proposed. Together, these studies motivate the Nepal-specific framing adopted here but leave open the question of how an automated, farmer-deployable countermeasure could be built and validated the gap this paper addresses.

Hemalatha [3] designed and tested an Arduino-based ultrasonic repellent circuit sweeping 10–65 kHz, reporting some deterrent effect but noting that monkeys habituate to a purely electronic, single-mode stimulus with repeated exposure. Biedenweg et al. [4], working with kangaroos, found that unpredictable, varied acoustic stimuli combining artificial and natural cues significantly delayed habituation compared with a repetitive single-tone signal. Maestripieri and Hoffman [7] provide behavioral grounding for why static deterrents fail against macaques specifically: the species exhibits high cognitive flexibility and social learning, enabling troops to rapidly assess and disregard stimuli that carry no real consequence. These three works jointly motivate but do not themselves implement a detection-triggered, variably-timed deterrent; the present system operationalizes this behavioral insight directly in its audio-actuation module (Section III-I).

Goyal [5] combined YOLOv5 with IoT hardware to detect and track stray animals in agricultural fields, reporting over 95% detection accuracy and a real-time notification component, but the system is multi-species and generic, and reports high initial hardware cost and internet dependency as open limitations. Shahrir et al. [6] developed a hybrid YOLOv5–SSD system for durian-plantation protection with Telegram notification and a predator-sound deterrent architecturally close to the present work but report a steep night-time accuracy drop, with monkey detection falling to roughly 70%, and call explicitly for lighting-independent detection; this is a gap the present system's conditional low-light enhancement module (Section III-G) was designed to address, though its field effect remains to be quantified (Section VI). Ramya et al. [8] compared YOLOv3, R-CNN, and Random Forest for crop-protection animal detection, reporting approximately 95% accuracy with the deep-learning approach and confirming the general superiority of camera-based automated detection over passive monitoring, without addressing deterrence or notification. Lekhadia et al. [13] deployed YOLOv8 on a Raspberry Pi 5 for multi-species dangerous-wildlife alerts (lions, elephants, leopards, monkeys), achieving 90.0% mAP@50 with a web dashboard, but the system is explicitly multi-species and generalist rather than tuned to a single locally dominant species, and does not incorporate a physical deterrent response.

Delwar et al. [9] implemented real-time farm surveillance using YOLOv8 and IoT hardware, achieving over 99% accuracy on curated data but reporting reduced performance under low-light and adverse-weather conditions a limitation echoed in [6] and directly relevant to the present system's still-unvalidated field robustness (Section VI). Singh and Krishnamurthi [14] combined an ESP32-CAM and Raspberry Pi with an optimized YOLOv8 model and Firebase Cloud Messaging for farmer notification, reporting precision and recall above 96%, demonstrating the practicality of low-power, edge-first crop-security architectures and reinforcing the architectural choice of on-device rather than cloud-dependent inference adopted here. The Ultralytics YOLO framework [10], ONNX Runtime [11], and Roboflow [12] used as the underlying training, export, and dataset-management tooling for the present system are themselves widely adopted across this literature, supporting the reproducibility of the described pipeline on equivalent open-source tools.

Across these themes, three gaps recur: (1) detection systems are typically multi-species and generic rather than tuned to a single, locally dominant conflict species [5], [13]; (2) deterrence, where present at all, uses a fixed or single-mode stimulus without an explicit habituation-avoidance design informed by behavioral literature [3], [6]; and (3) no reviewed system was designed for, or evaluated in, the Nepal Rhesus-macaque smallholder context [1], [2], [5], [6], [8], [9], [13], [14]. The system presented here addresses all three: a single-class, Nepal-specific macaque detector; a confidence-gated deterrent with rotating stimuli grounded

directly in the habituation literature [4], [7]; and a complete, bench-validated farmer-facing notification and logging loop.

III. PROPOSED METHODOLOGY

The methodology is described at the level of a conceptual pipeline and its constituent modules so that an equivalent system could be reproduced with comparable open-source tools (Python, an ONNX-compatible detector, a single-board computer, and a Bluetooth-capable audio actuator).

A. Overall Research Framework

The system follows an eight-stage pipeline: Capture → Quality Check → Conditional Enhancement → Detection → Decision → Deterrence → Notification → Logging. Fig. 1 shows the corresponding system architecture: an edge camera feeds a Raspberry Pi 4, which runs the YOLO26n/ONNX detector and a threshold-based decision module; a positive decision fans out to four subsystems an audio deterrent actuator, a Telegram notification channel, an SQLite event log, and a Flask/Flutter notification and review layer.

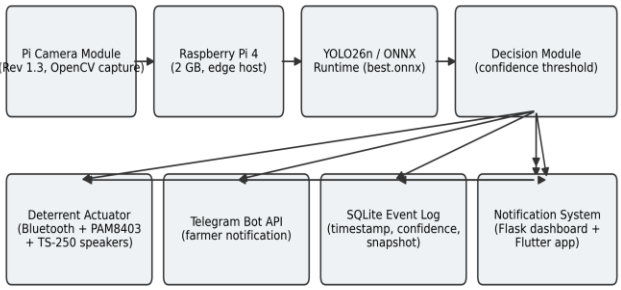

Fig. 1. Overall system architecture: acquisition, edge detection, decision, and the four farmer-facing response subsystems.

B. Data Acquisition

A single target class, 'macaque,' was used throughout. The base dataset comprised 8,229 field-representative and web-sourced macaque images, organized and bounding-box annotated using Roboflow. Bench testing of the resulting detector (Section IV) revealed a recurring false-positive failure mode the detector frequently fired when a human, rather than a macaque, appeared in frame a well-documented consequence of training a single-class detector on positive examples only. To correct this, 2,657 negative/background images containing no macaque (humans, cattle, scarecrows, and empty field scenes) were added to the same Roboflow project and annotated with zero bounding boxes, bringing the total dataset to 10,886 images. The dataset was partitioned 80% / 10% / 10% into training, validation, and test splits (Table I).

TABLE I. DATASET COMPOSITION AND SPLIT

Subset	Images	Share
Macaque (positive)	8,229	
Negative (background)	2,657	
Training	8,709	80%
Validation	1,089	10%
Test	1,088	10%
Total	10,886	100%

C. Data Preparation

Images were normalized in resolution and augmented (horizontal flipping, brightness adjustment, and cropping) to improve generalization across lighting, viewing angle, and partial occlusion before being exported in a YOLO-compatible input format. The negative images contribute background-only training signal rather than a second class, concentrating network capacity on precise localization of the target species while still teaching the detector the visual boundary against common non-macaque subjects.

D. Model Selection

YOLO26n the nano-scale variant of the YOLO family was selected from among candidate lightweight architectures (other YOLO variants and MobileNet-SSD) on the basis of detection accuracy, inference speed on resource-constrained hardware, and memory footprint. Training initialized from COCO-pretrained weights via transfer learning, reducing the data and compute required to reach a usable detector on a domain-specific, single-class dataset of moderate size.

E. Model Training

The model was trained for 50 epochs at an input resolution of 640×640 px on GPU-enabled cloud infrastructure (Kaggle, with Google Colab as backup), evaluated using precision, recall, mAP@50, and mAP@50–95 on the held-out test split. The best-performing checkpoint (best.pt) was exported to ONNX format (best.onnx) using the Ultralytics export utility, decoupling the trained model from the PyTorch training framework so that inference could run through the lightweight ONNX Runtime library on ARM-based edge hardware.

F. Edge Deployment and Inference

The exported model was deployed to a Raspberry Pi 4 Model B (2 GB RAM) running a 64-bit Raspberry Pi OS, inside a Python virtual environment using the ONNX Runtime CPU execution provider and OpenCV for camera capture. Each frame is resized and letterboxed to 640×640, normalized, and reshaped into the NCHW tensor layout expected by the model before being passed to an ONNX Runtime InferenceSession; Non-Maximum Suppression (NMS) removes duplicate or overlapping boxes before detections are handed to the decision module (Fig. 2).

G. Image Quality Assessment and Conditional Enhancement

Bench testing showed detection quality was noticeably affected by motion blur and low ambient brightness. Rather than retraining on a wider variety of degraded imagery, a lightweight, rule-based pre-processing module runs upstream of inference on every frame: the variance of the Laplacian of the grayscale frame is used as a sharpness score, and frames falling below a calibrated threshold are passed through an unsharp-mask sharpening filter; the mean pixel intensity of the V channel (HSV) is checked against a minimum-brightness threshold, and frames falling below it are enhanced using Contrast-Limited Adaptive Histogram Equalization (CLAHE) on the luminance channel. Both checks run on every frame, but enhancement is applied only when a frame actually falls below its threshold, adding an average of approximately 18 ms per frame when triggered and negligible overhead otherwise (Fig. 2).

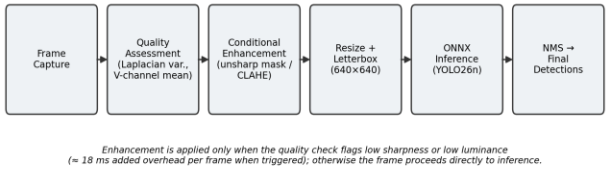

Fig. 2. Edge-side frame processing and detection pipeline.

H. Decision Mechanism

Detections are gated by a confidence threshold, calibrated at 0.60 through bench testing, above which the deterrence, notification, and logging actions are triggered (Fig. 3). This threshold was set empirically for the completed prototype and is not claimed to be globally optimal; formal threshold sensitivity analysis is identified as future work (Section VII).

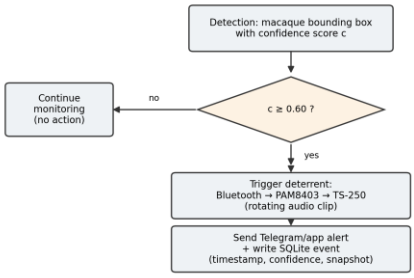

Fig. 3. Confidence-based decision, deterrence, and logging flow.

I. Non-Lethal Deterrence Design

When a detection crosses the confidence threshold, the Raspberry Pi 4 triggers the non-lethal deterrent by streaming a selected clip over Bluetooth to a dedicated Bluetooth audio decoder, which feeds a PAM8403 amplifier driving a pair of TS-250 tweeters. Rather than a single repeated tone, the deterrent module plays back a small rotating set of distress-call and broadband-noise clips (managed through pygame), with playback order and inter-trigger spacing varied to reduce the risk of rapid habituation directly operationalizing the habituation-avoidance findings of Biedenweg et al. [4] and the behavioral characterization of macaque cognitive flexibility by Maestripieri and Hoffman [7]. Long-term deterrence effectiveness under repeated real encounters has not been measured and is not claimed here (Section VI).

J. IoT Hardware Components and Operating Mechanism

For clarity and reproducibility, this subsection separates out the physical IoT hardware used across acquisition and deterrent actuation (Sections III-B and III-I) and states the working mechanism of each component. Eight hardware items were used: (1) a Raspberry Pi 4 Model B (2 GB RAM), (2) a Raspberry Pi Camera Module Rev 1.3, (3) a Bluetooth audio decoder, (4) a PAM8403 audio amplifier, (5) a pair of TS-250 dome tweeters, (6) jumper wires, (7) a breadboard, and (8) a battery. Fig. 4 shows how power, video, and audio signals flow between them.

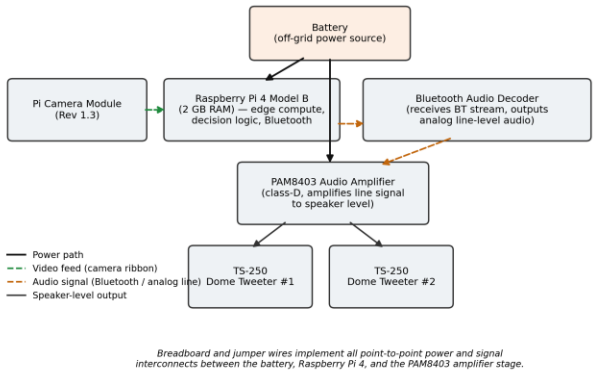

Fig. 4. IoT hardware components and their power / video / audio signal flow.

Raspberry Pi 4 Model B (2 GB RAM): the single edge-compute unit for the entire system. It runs the operating system, the ONNX Runtime inference session, the frame quality-assessment and conditional-enhancement logic (Section III-G), and the confidence-threshold decision module (Section III-H). When a detection crosses the confidence threshold, the Pi streams the selected deterrent audio clip over its onboard Bluetooth radio; it is powered from the battery.

Raspberry Pi Camera Module (Rev 1.3): the image-acquisition sensor for the system, connected to the Pi over its CSI ribbon interface and read via OpenCV (Section III-F). It supplies the live frame stream that feeds the quality-assessment, enhancement, and YOLO26n detection stages.

Bluetooth audio decoder: a Bluetooth-receiver module that pairs with the Raspberry Pi and decodes the streamed audio clip into an analog line-level signal. Routing the deterrent audio over Bluetooth rather than a wired connection lets the decoder/amplifier/speaker assembly be mounted at the field-edge actuation point, physically separated from the Pi and

camera, which simplifies enclosure design and cabling for an outdoor installation.

PAM8403 audio amplifier: a compact class-D stereo amplifier module that takes the low-power analog signal from the Bluetooth decoder and amplifies it to a level sufficient to drive the tweeters at an effective deterrent volume.

TS-250 dome tweeter (×2): the acoustic output transducers that physically reproduce the rotating distress-call and broadband-noise clips (Section III-I) at the field edge. Two units are used in parallel off the amplifier output to increase effective coverage and loudness across the monitored field-edge zone.

Jumper wires and breadboard: the physical prototyping hardware used to implement all point-to-point power and signal interconnects between the battery, the Raspberry Pi 4, and the PAM8403 amplifier stage. This breadboard-based wiring was appropriate for a bench-validated prototype; a soldered or enclosure-mounted wiring harness would be a natural hardening step for field deployment (Section VII).

Battery: provides off-grid power to the Raspberry Pi and, via the breadboard/jumper-wire wiring, to the amplifier and tweeter assembly, since mains electricity is not generally available at the field-edge deployment point. Capacity, voltage, and expected field runtime were not specified in the project source material and are not stated here [INFORMATION REQUIRED]; sizing the battery for a target continuous-operation duration is a practical prerequisite for the field deployment identified in Section VII.

TABLE II. IOT HARDWARE COMPONENT SUMMARY

Component	Role in System	Interface
Raspberry Pi 4 (2 GB)	Edge compute: inference, decision logic, orchestration	Battery in; Bluetooth to decoder
Pi Camera Module (Rev 1.3)	Image acquisition	CSI ribbon to Pi
Bluetooth audio decoder	BT audio stream → analog line-level audio	BT in; line-out to amplifier
PAM8403 amplifier	Amplifies line signal to speaker level	Line-in; battery power; speaker-out
TS-250 tweeter (×2)	Acoustic output of the deterrent	Speaker-level in from amplifier
Jumper wires / breadboard	Physical power and signal interconnects	Battery ↔ Pi ↔ amplifier
Battery	Off-grid power for the full chain	Direct to Pi and amplifier via wiring

K. Notification and Event Logging

The Telegram Bot API dispatches a real-time farmer notification upon a confirmed detection event, including a timestamp, confidence score, and detection snapshot. An SQLite-backed event log persists every event with the same fields, providing a queryable field-activity history. A Flask backend serves both the decision/notification logic and a web dashboard for reviewing historical events, while a companion web application (Provider state management, periodic HTTP polling) surfaces the same event data detection time, confidence, snapshot, and action taken as a phone-based alternative to the Telegram channel.

L. Algorithm

Algorithm 1 summarizes the per-frame detection-to-response logic implemented by the system.

```
Algorithm 1: Detection-to-Deterrence Loop
Input: live camera frame stream
1:  for each captured frame f do
2:      s ← LaplacianVariance(f); b ← MeanVChannel(f)
3:      if s < s_min then f ← UnsharpMask(f)
4:      if b < b_min then f ← CLAHE(f)
5:      f' ← LetterboxResize(f, 640, 640); normalize(f')
6:      D ← ONNXInference(f');  D ← NMS(D)
7:      for each detection d in D do
8:          if d.class = 'macaque' and d.confidence ≥ 0.60 then
9:              Trigger-Deterrent()      // rotating audio clip
10:             Send-Notification(d)     // Telegram
11:             LogEvent(d)              // SQLite: t, conf, snapshot
12:         end if
13:     end for
14: end for
```

M. Reproducibility Details

To support reproducibility, Table VIII summarizes software versions, resolved training hyperparameters, and augmentation settings used for the final training run.

TABLE III. REPRODUCIBILITY DETAILS

Component	Value
Python	3.12.13
PyTorch	2.10.0+cu128
Ultralytics	8.4.144
OpenCV	4.13.0
NumPy	2.0.2
CUDA	12.8
Optimizer	AdamW
Epoch	50
LR scheduler	Linear decay
Weight decay	0.0005
Warmup epochs / momentum	3 / 0.8

Component	Value
Random seed	0 (deterministic = True)
Mixed Precision (AMP)	Enabled
Batch size / image size	16 / 640x640
NMS IoU threshold	0.7
Confidence threshold (F1-optimal, benchmarking)	0.35
Confidence threshold (deployed decision module)	0.60 (Section III-H; see Section IV-B for rationale)
Patience / close_mosaic	100 / 10
HSV hue / saturation / value	0.015 / 0.700 / 0.400
Horizontal / vertical flip	0.500 / 0.000
Translation / scale	0.100 / 0.500
Mosaic / mixup	1.000 / 0.000
Random erasing	0.400
Auto-augment	RandAugment
Dataset source	Roboflow Universe, “Macaque Detection First Addition v5”
Hardware	2x Tesla T4(Kaggle Cloud, 14.9 GB/GPU
Training time	0.689 hours (50 epochs)

The two confidence thresholds reported above serve different purpose: 0.35 is the empirically F1-optiml threshold from the sensitivity sweep in Section IV, used for benchmarking; 0.60 is the operational threshold adopted in the deployed decision module, chosen to prioritize precision over recall for the reasons discussed in Section IV-B.

IV. EXPERIMENTS AND RESULTS

The retrained YOLO26n model (10,886-image dataset, 50 epochs, 640×640 resolution) achieved the test-set performance reported in Table IV.

TABLE IV. FINAL MODEL PERFORMANCE (HELD-OUT TEST SPLIT)

Metric	Value
Precision	89.84%
Recall	80.90%
mAP@50	89.9%
mAP@50–95	65.75%

Table V reports edge-device and end-to-end system performance measured during bench testing.

TABLE V. EDGE-DEVICE AND END-TO-END PERFORMANCE

Measure	Value
Avg. inference latency (RPi4 CPU)	≈ 320 ms
Avg. Alert response time	≈ 2.66 ms
Avg. API latency	≈ 253 ms
Telegram latency	≈ 1081 ms
End-to-End latency	≈ 1653 ms
Prototype hardware cost	NPR 41,440

C. Effect of Adding Negative Training Samples

An internal dataset development comparison was conducted across progressively assembled version of the dataset. The initial dataset contained 4,320 images. During dataset development the positive image collection was further expanded to 8,229 macaque images. The final dataset was then expanded to 10,886 images by incorporating 2,657 negative/background images containing humans, cattle, scarecrows, and empty field scenes. Therefore, the 4,320-image condition in Table IV represents an earlier dataset version rather than 8,229 image positive only dataset described in Section III-B. Table IV reports the available before/after performance values for the earlier and final dataset configurations.

TABLE VI. PERFORMANCE ACROSS DATASET DEVELOPMENT STAGES

Metric	Earlier (4,320 img)	Final (10,886 img)
Precision	91.22%	89.84%
Recall	88.01%	80.90%

mAP@50 and mAP@50–95 was reported qualitatively as “essentially unchanged” after the negative images were added, without a stated numeric baseline for the pre-negative-sample model; the exact before-value for these two metrics.

D. Threshold Sensitivity and Confusion Matrix Analysis

Threshold sensitivity and confusion matrix analyses were performed to evaluate the precision-recall trade-off and assess the model’s detection errors across different confidence thresholds.

TABLE VII. CONFIDECNE THRESHOLD SENSITIVITY

Confidence	Precision	Recall	F1 score
0.10	0.873	0.860	0.867
0.20	0.873	0.860	0.867
0.30	0.870	0.863	0.867
0.40	0.897	0.834	0.865
0.50	0.928	0.803	0.861
0.60	0.952	0.765	0.848
0.70	0.965	0.692	0.806
0.80	0.980	0.594	0.470
0.90	0.998	0.311	0.474

The statistically F1-optimal threshold (0.35) was not adopted for development. Instead, the confidence threshold was deliberately set higher, at 0.60 (Section III-H), to bias the operating point toward precision (95.2% at conf = 0.60) over recall (76.5%), a design choice made directly address the human false-positive failure mode identified during bench testing (Section III-B) because a spurious deterrent trigger carries a real cost (wasted audio actuation, farmer alert fatigue from false Telegram notifications) while an occasional missed detection is partially mitigated by the continuous frame by frame nature of video monitoring, the system trades some recall for a substantially lower false-trigger rate.

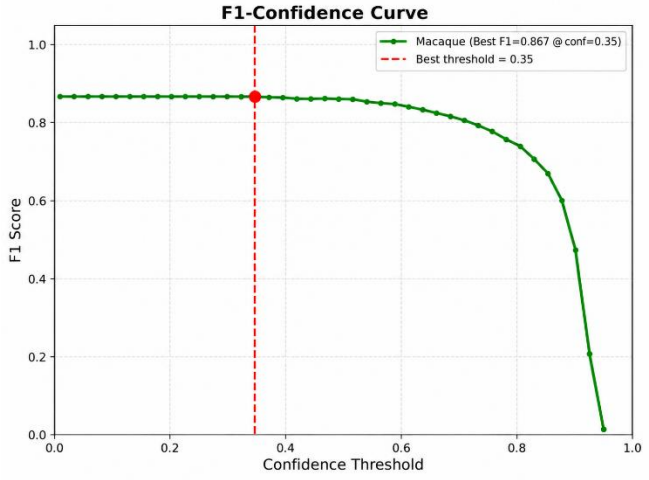

Fig 5: F1-confidence curve.

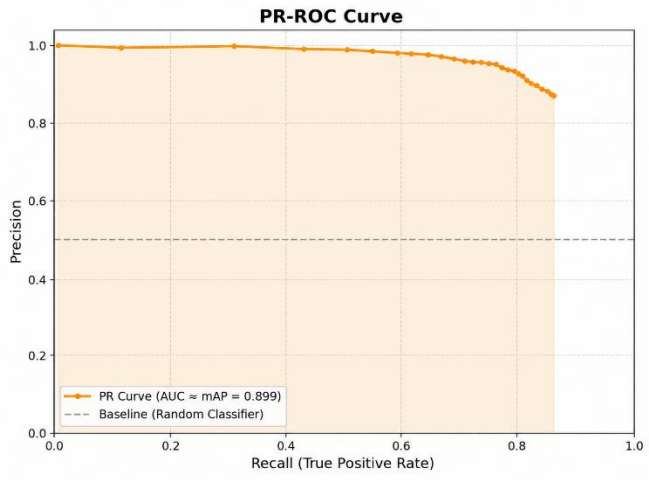

Fig 6: PR-ROC Curve

On-device testing confirmed that best.onnx loads and executes correctly under the ONNX Runtime CPU execution provider on the Raspberry Pi 4's ARM Cortex-A72 processor, and that the deployed model correctly and consistently identifies macaque presence in photos and video presented to the camera during indoor bench testing. With all subsystems integrated, presenting macaque imagery to the camera was confirmed to reliably trigger the Bluetooth-driven deterrent, dispatch Telegram notifications, and write a corresponding SQLite event record, in sequence.

V. ANALYSIS AND DISCUSSION

Precision (89.84%) exceeds recall (80.90%) by a wider margin than in the pre-negative-sample model (where the gap was 91.22% vs. 88.01%), consistent with the intended effect of adding negative training examples: the retrained model is somewhat more conservative, more often correctly withholding a detection when it is not confident. Recall is the operationally more sensitive metric for this application, since a missed detection corresponds directly to an undeterred crop-raiding event; it is therefore notable that recall did not drop materially (88.01% → 80.90%) as a side effect of adding negatives.

The gap between mAP@50 (89.9%) and mAP@50–95 (65.75%) reflects a localization-precision characteristic rather than a detection-failure issue: the model reliably finds macaques, but its bounding boxes are not always tightly localized. Because the decision logic (Section III-H) only requires a confident detection not a tightly localized box to trigger the deterrent and notification pipeline, this characteristic is not expected to materially affect end-to-end system behavior, though it remains a target for improvement (Section VII).

The measured average inference latency of ≈ 320 ms on Raspberry Pi 4 CPU hardware, combined with Bluetooth streaming and audio playback startup, yields an estimated 1–2 s detection-to-deterrent response time. This is expected to be adequate for the walking-pace intrusion behavior typical of macaque crop raiding, but this expectation has not been confirmed under field conditions and is treated here as a hypothesis rather than a validated result.

The negative-sample finding (Table IV) is consistent with a well-known failure mode of single-class detectors: without counter-examples, a model tends to learn 'primate-shaped subject against this background' rather than the specific visual boundary between a macaque and other subjects (notably humans). Adding background and near-confusable negatives sharpened this boundary, improving precision without a corresponding loss of recall a pattern that also appears, in a different multi-species setting, in the negative-sampling strategy used by Lekhadia et al. [13].

The choice of a confidence-gated, audio-rotating deterrent over a single fixed alarm is directly supported by the habituation literature reviewed in Section II-B [3], [4], [7]: a predictable stimulus is expected to lose effectiveness against a behaviorally flexible species such as the Rhesus macaque, whereas varied timing and clip selection are expected to delay this decline. This expectation has not yet been tested against live macaques and is identified as a priority for field validation.

The reported precision and mAP figures are broadly comparable to accuracy levels reported for other YOLO-based animal-intrusion detectors in the literature reviewed [5], [6], [8], [9], [13], while being purpose-trained on a Nepal-relevant, single-species dataset with negative examples added specifically to address the false-positive behavior observed during bench testing. No controlled, quantitative comparison against alternative architectures (e.g., MobileNet-SSD, which was considered during model selection in Section III-D but not benchmarked on this dataset) was performed within this project.

All results reported here derive from indoor bench testing presenting photographs and short video clips of macaques to the camera under controlled desk conditions rather than from live animals in an agricultural field. Real-world factors that bench testing cannot capture, including outdoor lighting variation, camera-to-subject distance and angle in an open field, wind and vegetation motion, rural network connectivity, and genuine macaque behavioral response to the deterrent, remain unvalidated and are discussed further in Section VI.

VI. VALIDATION AND LIMITATIONS

Indoor bench testing confirmed the following on a qualitative basis:

- best.onnx loads and executes correctly under the ONNX Runtime CPU execution provider on the Raspberry Pi 4's ARM Cortex-A72 processor.
- The deployed model correctly and consistently identifies macaque presence in photos and video presented to the camera.
- A confident detection reliably triggers the Bluetooth-driven deterrent within approximately 1–2 seconds.
- Telegram and Flutter notifications are dispatched automatically upon a confirmed detection.
- The corresponding event (timestamp, confidence, snapshot) is correctly written to the SQLite log and reflected in the dashboard.

Real agricultural field deployment is the explicitly identified next step; the following claims are therefore not made: real-world field detection effectiveness, real-world false-positive rate, long-term deterrence effectiveness against live macaques, notification reliability under rural network conditions, and robustness to outdoor lighting and weather variation. Table V summarizes the field test plan designed to close this gap; these are planned test methods, not results, since field testing has not yet been conducted.

TABLE VIII. PLANNED FIELD VALIDATION TEST PLAN

Dimension	Planned Method
End-to-end response time	Time detection-to-deterrent and detection-to-notification delivery across repeated live intrusions at a field site.
False-positive rate	Log and categorize all field triggers as true/false positive, with attention to humans, cattle, scarecrows, and wind-blown vegetation.
Deterrent effectiveness	Observe macaque behavioral response (dispersal time, distance) following deterrent activation across multiple real encounters.
Notification reliability	Measure delivery success rate and latency for Telegram/app alerts under actual rural network conditions.
Environmental robustness	Compare detection recall across daylight, dusk/low-light, and weather conditions, with and without the enhancement module active.

VII. CONCLUSION AND FUTURE WORK

This paper presented the design, implementation, and bench-level evaluation of an edge-based AI-IoT system for detecting Rhesus macaque intrusions and triggering an automated, non-lethal, habituation-resistant deterrent response, integrated with farmer notification and event logging. A YOLO26n detector retrained on a 10,886-image, single-class dataset (with 2,657 negative samples added to correct a human false-positive failure mode) achieved 89.84% precision, 80.90% recall, 89.9% mAP@50, and 65.75% mAP@50–95, and was deployed via ONNX Runtime on a Raspberry Pi 4 with an average inference latency of ≈ 320 ms. Indoor bench testing confirmed that detection, deterrent actuation, notification, and event logging function correctly together as an integrated pipeline. These results demonstrate that an affordable (NPR 41,440), farmer-deployable, non-lethal detection-and-deterrence system is technically achievable for the Nepal smallholder context; whether this bench-validated foundation translates into real-world crop protection has not yet been established and is the necessary next step.

Future work, in the priority order identified from this project, comprises:

1) Field deployment and validation (highest priority): installing the system at an actual field site and executing the test plan in Table V end-to-end response time, false-positive rate, deterrent effectiveness, notification reliability, and environmental robustness under real conditions.
2) Quantifying low-light performance: the pre-processing module (Section III-G) was verified only on sample imagery; its real-world effect on recall at dusk and in low light is unmeasured and requires field data.
3) Increasing model capacity where justified: a modest upgrade to YOLO26s could improve bounding-box localization (mAP@50–95) within the Raspberry Pi 4's inference budget, once field data identifies whether this is the limiting factor.

REFERENCES

[1] S. Rai and R. Rai, "Monkey menace in Nepal: An analysis and proposed solutions," J. Multidiscip. Sci., vol. 6, no. 1, pp. 26–31, 2024, doi: 10.33888/jms.2024.614.

[2] B. Lamichhane and B. Lamichhane, "A story of farmer-monkey interactions towards their effects on semi-urban migration and livelihood," J. Popul. Dev., vol. 5, no. 1, pp. 18–27, Jun. 2024, doi: 10.3126/jpd.v5i1.67532.

[3] M. Hemalatha, "Smart monkey repellent circuit using Arduino-UNO," Sree Rama Engineering College, Tirupati, India.

[4] T. A. Biedenweg, M. H. Parsons, P. A. Fleming, and D. T. Blumstein, "Sounds scary? Lack of habituation following the presentation of novel sounds," PLoS ONE, vol. 6, no. 1, e14549, Jan. 2011, doi: 10.1371/journal.pone.0014549.

[5] N. Goyal, "A deep learning based model for detection and tracking of stray animals in the fields," Curr. Agric. Res. J., vol. 12, no. 3, pp. 1470–1476, 2024, doi: 10.12944/CARJ.12.3.38.

[6] A. S. Shahrir, Z. Ayop, S. Anawar, and N. M. Zainudin, "A hybrid YOLOv5–SSD IoT-based animal detection system for durian plantation protection," Int. J. Wireless Mobile Netw., vol. 17, no. 5, 2025, doi: 10.5121/ijwmn.2025.17501.

[7] D. Maestripieri and C. L. Hoffman, "Behavior and social dynamics of Rhesus macaques on Cayo Santiago," in Bones, Genetics, and Behavior of Rhesus Macaques, Q. Wang, Ed. New York: Springer, 2012, pp. 247–292, doi: 10.1007/978-1-4614-1046-1_12.

[8] G. Ramya, J. Sreeja, K. Jyothi, J. Radhika, and R. Vigneshwari, "Animal detection for crop protection using deep learning: Insights from YOLO V3, R-CNN, Random Forest," J. Data Sci., 2025.

[9] T. S. Delwar, S. Mukhopadhyay, A. Kumar, M. Singh, Y. Lee, J.-Y. Ryu, and A. S. M. S. Hosen, "Real-time farm surveillance using IoT and YOLOv8 for animal intrusion detection," Future Internet, vol. 17, no. 2, p. 70, Feb. 2025, doi: 10.3390/fi17020070.

[10] Ultralytics, "YOLO: Real-time object detection documentation," Ultralytics Docs. [Online]. Available: https://docs.ultralytics.com

[11] ONNX Runtime, "Cross-platform, high performance ML inferencing," Microsoft / ONNX Community. [Online]. Available: https://onnxruntime.ai

[12] Roboflow, "Computer vision dataset management and annotation platform," Roboflow Inc. [Online]. Available: https://roboflow.com

[13] S. Lekhadia, P. Ghoniya, S. B. V. Reddy, A. Panshala, D. Trivedi, and M. Singh, "An edge computing approach for real-time wildlife detection and alert system using YOLOv8 on Raspberry Pi 5," Int. J. Eng. Res. Technol., vol. 14, no. 8, Sep. 2025, doi: 10.5281/zenodo.18103444.

[14] P. Singh and R. Krishnamurthi, "IoT-based real-time object detection system for crop protection and agriculture field security," J. Real-Time Image Process., vol. 21, p. 106, 2024, doi: 10.1007/s11554-024-01488-8.