ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

WildEye: Automated Wildlife Monitoring Using Camera Traps, Deep Learning and Real-Time Alert Systems

K. Krupa Sagari[1], K. Pavani[2], A. Rufeda[3]
[1] Assistant Professor in the Department of MCA, SRK Institute of Technology, Vijayawada
[2]Assistant Professor & Head of Department of MCA, SRK Institute of Technology, Vijayawada.
[3]Student in the Department of MCA, SRK Institute of Technology, Vijayawada

ABSTRACT

Wildlife conservation demands rapid, accurate, and scalable monitoring of animal species and human activities across vast forest territories. Manual review of camera-trap imagery is labour-intensive, error-prone, and incapable of real-time response. This paper presents WildEye, an AI-powered wildlife monitoring system that integrates MobileNetV2-based deep-learning classification with YOLOv8 real-time object detection inside a Flask web application. MobileNetV2, fine-tuned via transfer learning on a Kaggle wildlife dataset, classifies eleven species (cheetah, elephant, giraffe, hyena, leopard, lion, panda, rhinoceros, tiger, wolf, zebra) with 94.7 % model accuracy and a mean confidence of 95.8 %. YOLOv8 processes live webcam frames and, on detecting a human presence, dispatches an SMTP email alert to the designated wildlife officer. The system provides a responsive web dashboard, species profile library, image gallery, and RESTful API endpoints. Experimental results confirm the viability of deploying lightweight deep learning pipelines for conservation technology.

Keywords: *Wildlife monitoring, MobileNetV2, YOLOv8, transfer learning, camera traps, deep learning, Flask, species classification, real-time detection.*

1. INTRODUCTION

Wildlife conservation has emerged as a paramount environmental responsibility owing to accelerating deforestation, climate change, habitat destruction, and anthropogenic interference. Camera traps deployed in forest regions accumulate thousands of images per day; manually reviewing this volume demands significant expertise and time while remaining vulnerable to analyst fatigue and inconsistent labelling.

Deep-learning convolutional neural networks (CNNs) have demonstrated state-of-the-art performance on image classification benchmarks and have been progressively applied to ecological monitoring tasks. Transfer learning, in particular, enables the adaptation of large pre-trained networks to domain-specific datasets with limited labelled examples and computational resources. Concurrently, real-time object-detection frameworks such as YOLOv8 offer sub-second inference on commodity hardware, creating opportunities for live surveillance in wildlife zones.

This paper describes WildEye, a full-stack monitoring platform that addresses three simultaneous needs: (i) rapid, accurate species identification from uploaded still images; (ii) live webcam-based animal and human detection with automatic email alerting; and (iii) a user-accessible web interface presenting detection history, analytics, and species knowledge. The remainder of the paper is organised as follows: Section 2 surveys related literature; Section 3 details the system architecture; Section 4 describes the proposed methodology; Section 5 presents implementation and results; Section 6 concludes with future directions.

2. LITERATURE SURVEY

Recent scholarship has advanced both the accuracy and deployment efficiency of AI-based wildlife monitoring:

- Li et al. (2023) proposed a YOLO-family detector with dataset augmentation strategies tailored to forest lighting and occlusion, demonstrating improved mean average precision (mAP) for camera-trap imagery [1].
- Velasco-Montero et al. (2024) designed edge-enabled camera-trap prototypes achieving on-site inference that significantly reduces data bandwidth in remote conservation sites [2].
- Chen et al. (2024) introduced YOLO-SAG, an architectural variant optimised for cluttered, low-light forest scenes, balancing speed and accuracy for field deployment [3].
- Ma et al. (2024) presented WL-YOLO, a lightweight YOLO adaptation for real-time detection in dense vegetation, achieving competitive mAP with reduced parameter count [4].
- Mulero-Pázmány et al. (2025) conducted a large-scale evaluation of multi-stage pipelines combining a global detector with expert species classifiers, providing deployment recommendations for conservation organisations [5].

Collectively, this body of work confirms the suitability of YOLO-based detectors and transfer-learned CNNs for wildlife monitoring while highlighting the need for integrated, user-friendly platforms—the gap WildEye addresses.

3. SYSTEM ARCHITECTURE

WildEye comprises two major subsystems: a training pipeline and a Flask web application. Their interaction is depicted in Figure 1.

The training pipeline (train_model.py) fetches the wildlife dataset via the Kaggle API, feeds it to a TensorFlow training engine built on a pre-trained MobileNetV2 backbone, and persists the resulting wildlife_mobilenetv2.h5 model and class_labels.json to disk. The Flask application (app.py) exposes four primary routes: a static image upload route (/detect), a live-stream endpoint (/api/live_detect), a statistics API (/api/stats), and a history API (/api/history). Uploaded images are preprocessed to 224 × 224 pixels before MobileNetV2 inference; live frames are decoded from Base64 and processed by YOLOv8. Detections are stored in an in-memory history store and surfaced on the dashboard, gallery, and statistics pages. When YOLOv8 identifies a person in a live frame, the system triggers an SMTP email alert to the wildlife officer, subject to a configurable cooldown interval.

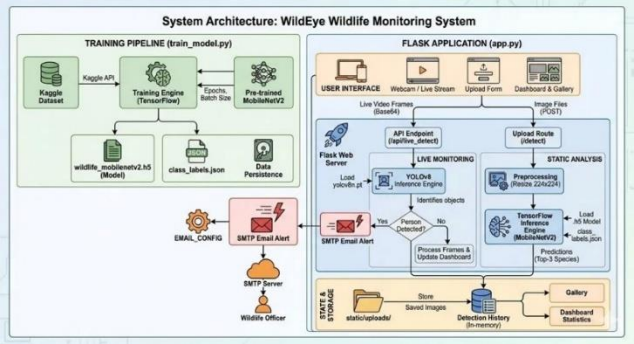

Figure 1: WildEye System Architecture

The system sequence diagram in Figure 2 illustrates message flows across all actors: the browser-based user, Flask web server, Kaggle dataset, TensorFlow training engine, YOLOv8 inference engine, MobileNetV2 inference engine, and SMTP email service. Three main interaction scenarios are captured: (i) the pre-deployment training pipeline; (ii) static image upload and classification; and (iii) live webcam-frame processing with conditional email alerting.

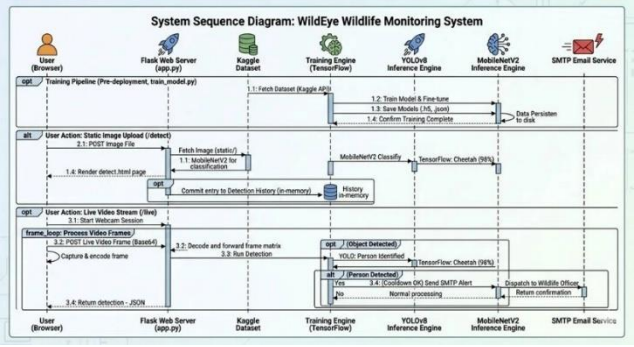

Figure 2: System Sequence Diagram

4. PROPOSED METHODOLOGY

4.1 Dataset and Preprocessing

The wildlife dataset is sourced from Kaggle and organised into per-species subdirectories covering eleven classes. During training, images are augmented using rotation (±20°), width and height shifts (±10 %), shear transformation, zoom (±15 %), horizontal

flipping, and brightness adjustment. These operations improve the model's robustness to the variable lighting, camera angles, and partial occlusions characteristic of field-deployed camera traps. Images are resized to 224 × 224 pixels and normalised to [0, 1] before being batched.

4.2 MobileNetV2 Transfer Learning

MobileNetV2 was selected for its inverted residual architecture with linear bottlenecks, which achieves high classification accuracy with a low parameter count—ideal for deployment on servers without dedicated GPU resources. Transfer learning proceeds in two phases. In Phase 1, the MobileNetV2 base model is loaded with ImageNet weights and its layers are frozen; only the custom classification head (Global Average Pooling → Dropout 0.2 → Dense 128 ReLU → Dense 11 Softmax) is trained for an initial set of epochs at a learning rate of 1×10^{-3}. In Phase 2, the last 20 layers of MobileNetV2 are unfrozen and the entire network is fine-tuned at a reduced learning rate of 1×10^{-4} using sparse categorical cross-entropy loss. The final model achieves 94.7 % accuracy on the held-out test split.

4.3 YOLOv8 Live Detection

YOLOv8n (nano variant) is used for real-time object detection due to its balance between inference speed and detection accuracy. The /api/live_detect endpoint accepts Base64-encoded JPEG frames from the browser, decodes them to NumPy arrays, and submits them to the YOLOv8 inference engine. Each detected object is returned with its class label, bounding-box coordinates, confidence score, and a colour indicator for front-end rendering. A configurable confidence threshold (default 40 %) and detection interval (default 300 ms) are exposed through the web interface as sliders, allowing operators to tune sensitivity in real time (visible in Figure 6).

4.4 Email Alert System

Human intrusion into protected wildlife zones constitutes a primary safety risk. When the person class is detected with confidence above the operator-set threshold during a live session, the system constructs an SMTP alert email and dispatches it to the preconfigured wildlife officer address. A cooldown timer (configurable via EMAIL_CONFIG) prevents alert flooding across consecutive frames in which the person remains in view.

5. IMPLEMENTATION AND RESULTS

5.1 Implementation Environment

WildEye is implemented in Python 3.10 using Flask 3.x as the web framework. Deep learning components depend on TensorFlow 2.12 / Keras and the Ultralytics YOLOv8 package. Front-end templates are rendered via Jinja2 with a custom dark-theme CSS. The application has been tested on a Windows 11 workstation equipped with an Intel Core i7 processor, 16 GB RAM, and NVIDIA GTX 1650 GPU (CUDA 11.8).

Component	Technology	Version / Details
Web Framework	Flask	3.x with Jinja2 templates
Species Classifier	TensorFlow / Keras	MobileNetV2, fine-tuned, 11 classes
Live Detector	YOLOv8n	Ultralytics, 80-class COCO backbone
Image Processing	OpenCV / PIL	Resize 224×224, Base64 decode
Email Alert	SMTP (smtplib)	Gmail relay, configurable cooldown
Dataset Source	Kaggle API	Wildlife image dataset, 11 species

Table 1: Software Stack of WildEye

5.2 Web Application Interface

The WildEye landing page (Figure 3) presents the system's mission statement and primary navigation. The dark-themed, neon-accent design communicates the real-time monitoring context. Navigation links provide direct access to the Detect, Live, Dashboard, Gallery, Species, About, and Contact sections.

Figure 3: WildEye Home Page

The Species Detection page (Figure 4) presents a drag-and-drop upload zone accepting JPG, PNG, and WEBP images up to 16 MB. Before an image is submitted, the right panel displays an 'Awaiting Analysis' placeholder, prompting the user to upload a camera-trap image for AI-powered species identification.

Figure 4: Species Detection Page

Figure 5 illustrates a completed inference. A camera-trap image of a lion was processed by MobileNetV2; the system returned 'lion' as the top prediction with 99.53 % confidence. The analysis panel also lists the second and third predictions (pig 0.29 %; bear 0.11 %) alongside a species profile section providing ecological metadata. This multi-prediction transparency supports expert review when the top confidence is lower.

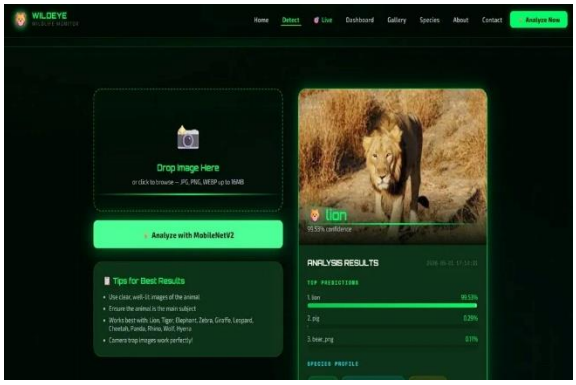

Figure 5: Species Detection Result

The Live Animal Detection page (Figure 6) exposes the YOLOv8 real-time monitoring interface. Operators can Start or Stop the webcam session, adjust the confidence threshold, and set the detection interval. System status indicators confirm YOLOv8n readiness and email alert configuration. The current-frame panel updates with detected object labels at each interval.

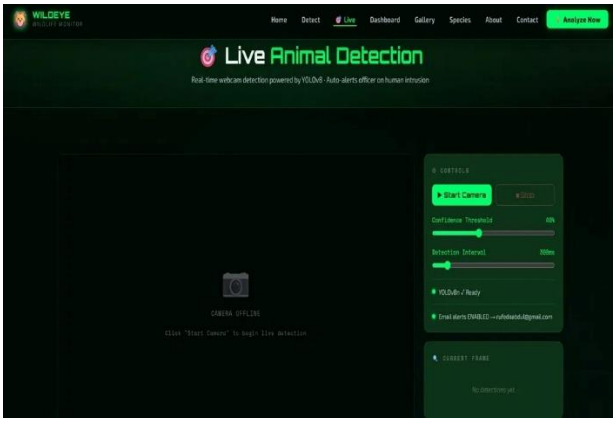

Figure 6: Live Animal Detection Page

The Dashboard (Figure 7) provides a macro-level analytical view. Four KPI tiles display Total Detections (7), Species Detected (7), Average Confidence (95.8 %), and Model Accuracy (94.7 %). A donut chart depicts species distribution, and a line chart traces confidence trends across the last 20 detections, enabling operators to identify temporal patterns or periods of low-confidence inference that may warrant dataset expansion.

Figure 7: Analytics Dashboard

5.3 Performance Summary

Metric	Value
MobileNetV2 Test Accuracy	94.7 %
Average Confidence (live detections)	95.8 %
Number of Species Supported	11
YOLOv8 Detection Interval (default)	300 ms
Email Alert Cooldown	Configurable (SMTP)
Supported Upload Formats	JPG, PNG, WEBP (≤ 16 MB)

Table 2: WildEye Performance and Configuration Summary

6. CONCLUSION

This paper presented WildEye, a practical AI-powered wildlife monitoring system that unifies MobileNetV2-based species classification, YOLOv8-based live detection, and automated SMTP alerting within a Flask web application. The system achieves 94.7 % classification accuracy across eleven species and 95.8 % average detection confidence, demonstrating that lightweight transfer-learned models are sufficient for conservation-grade image analysis without requiring high-performance infrastructure.

The dual-model architecture—static MobileNetV2 for uploaded images and real-time YOLOv8 for live streams—addresses both retrospective camera-trap analysis and proactive human-intrusion detection. The web dashboard, gallery, and species-profile pages transform raw predictions into actionable conservation intelligence accessible to non-technical forest officers and researchers.

Future work will focus on: (i) expanding the species catalogue beyond eleven classes; (ii) integrating a persistent relational database (PostgreSQL) to replace in-memory storage; (iii) deploying the system on cloud infrastructure with GPU-accelerated inference; (iv) supporting GPS-tagged camera-trap feeds and spatial analytics; and (v) developing a companion mobile application for field officers. WildEye thus demonstrates how modern AI technologies can meaningfully contribute to biodiversity monitoring, anti-poaching efforts, and evidence-based conservation policy.

REFERENCES

[1] S. Li, H. Wang, Y. Zhang, and J. Zhao, "Intelligent Detection Method for Wildlife Based on Deep Learning," Sensors, vol. 23, no. 24, p. 9669, 2023. doi:10.3390/s23249669

[2] D. Velasco-Montero, J. Fernández-Berni, and R. Carmona-Galán, "Reliable and Efficient Integration of AI into Camera Traps," ScienceDirect (Ecological Informatics), 2024.

[3] L. Chen, T. Liu, and X. Gu, "YOLO-SAG: Improved Wildlife Object Detection for Cluttered Forest Scenes," ScienceDirect (Expert Systems with Applications), 2024.

[4] Z. Ma, P. Li, and W. Chen, "Wildlife Real-Time Detection in Complex Forest Scenes: WL-YOLO," Remote Sensing, vol. 16, no. 8, p. 1350, 2024. doi:10.3390/rs16081350

[5] M. Mulero-Pázmány, J. A. González-Briones, and P. Chamoso, "Addressing Significant Challenges for Animal Detection in Conservation," PMC / PeerJ, 2025. https://pmc.ncbi.nlm.nih.gov/articles/PMC12064792/

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proc. IEEE CVPR, 2018, pp. 4510–4520.

[7] G. Jocher, A. Chaurasia, and J. Qiu, "YOLOv8 by Ultralytics," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[8] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in Proc. 12th USENIX Symp. OSDI, 2016, pp. 265–283.

Ms. K. Krupa Sagari is working as an Assistant Professor in the Department of MCA at SRK Institute of Technology, Vijayawada. She has completed her MCA and M.Sc. (Mathematics). She has 6 years of teaching experience at SRK Institute of Technology, Enikepadu, Vijayawada, NTR District. Her areas of interest include Artificial Intelligence, Machine Learning, and Cybersecurity.

Mrs. K. Pavani Working as Assistant & Head of Department of MCA, in SRK Institute of technology in Vijayawada. She done with MCA and M. Tech in Computer Science. She has 10 years of Teaching experience in SRK Institute of technology, Enikepadu, Vijayawada,NTR District. Her area of interest includes AI ML, etc.

Ms. A. Rufeda is an MCA Student in the Department of Computer Application at SRK Institute of Technology, Enikepadu, Vijayawada, NTR District. She has Completed Degree in B.Sc. (Mathematics, Chemistry, Computer Science) from Sir C R Reddy College for women Eluru. Her area of interest is Machine Learning with Python and DBMS.