Intelligent Wild Animal Detection And Adaptive Risk Alert System Using Yolov8 And DeepSort

Rajaram K[1], Juliet S[2], Ramya M[3], Anish I[4]
[1, 2, 3, 4]Dept of Computer science
[1, 2, 3, 4] Jeppiaar University, Chennai, India – 22011014

Abstract- *Human–wildlife conflict is a growing global concern as expanding human settlements encroach upon natural habitats. Traditional wildlife monitoring methods—manual patrols and passive camera traps—suffer from delayed response, limited scalability, and an inability to assess threat severity in real time. This paper presents an Intelligent Wild Animal Detection and Adaptive Risk Alert System that integrates YOLOv8-based real-time object detection, DeepSORT multi-object tracking with persistent unique identification, trajectory-based behavior classification (roaming, approaching, aggressive), a novel five-factor composite risk scoring engine with sigmoid normalization, species-aware dynamic geofencing, automated sound deterrents, and multichannel alert dispatch via SMS and WhatsApp. The system is orchestrated through a modular Python pipeline and visualized on a live FastAPI web dashboard with WebSocketdriven video streaming. Experimental evaluation on wildlife surveillance footage demonstrates a mean Average Precision (mAP@0.5) of 87.3% for detection, 94.6% tracking consistency (MOTA), and sub-200ms end-to-end latency per frame on consumer-grade hardware. The adaptive risk formula—*

$R_{final} = \sigma\,\alpha \cdot S_d \cdot \psi(b) + \beta \cdot (1 - d_{norm}) \cdot v + \gamma \cdot \log_2(n+1) + \left(\delta \cdot T_{\text{night}} + \epsilon \cdot H_{\text{loc}}\right)$

—enables context-sensitive threat escalation, reducing false-positive alerts by 41% compared to static threshold baselines. The platform's modular architecture and YAML-driven configuration support rapid deployment at wildlife corridors, forest perimeters, and agricultural buffer zones.

Keywords: Adaptive Geofencing, Behavior Classification, DeepSORT Tracking, Dynamic Risk Scoring, Human–Wildlife Conflict, Multi-Channel Alerts, Sound Deterrent, Wildlife Detection, YOLOv8

I. INTRODUCTION

Human–wildlife conflict (HWC) represents one of the most pressing challenges at the intersection of conservation biology and rural livelihoods. As agricultural frontiers expand into previously undisturbed forest corridors, encounters between humans and wild animals—elephants, bears, leopards, wild boar, and gaur—have escalated dramatically, resulting in crop destruction, livestock predation, property damage, and, critically, loss of human and animal life [1]. The World Wildlife Fund estimates that HWC affects more than 75% of large carnivore species globally, with India alone reporting over 500 human fatalities annually from wildlife encounters [2].

Conventional mitigation strategies include manual night patrols, physical barriers (electric fences, trenches), and passive camera traps [3]. While these approaches provide some protection, they share fundamental limitations: (i) manual patrols are resource-intensive, weather-dependent, and cannot cover large perimeters; (ii) physical barriers require significant capital investment and maintenance; and (iii) camera traps offer only post-hoc evidence, lacking the ability to issue real-time warnings or assess threat severity [4].

Recent advances in deep learning—particularly convolutional neural networks (CNNs) for object detection and recurrent architectures for sequence modeling—have opened new possibilities for automated, real-time wildlife surveillance [5]. The YOLO (You Only Look Once) family of detectors [6] achieves high accuracy at video-rate speeds on edge hardware, while multi-object tracking algorithms such as DeepSORT [7] assign persistent identities across frames, enabling behavioral trajectory analysis.

This paper presents a comprehensive, end-to-end system that addresses four critical gaps in existing wildlife monitoring solutions:

Real-Time Detection & Tracking: YOLOv8 detection coupled with DeepSORT tracking for 10+ wildlife species with unique counting and persistent identification across frames.

Behavioral Intelligence: A trajectory-based classifier that categorizes animal movement as ROAMING, APPROACHING, or AGGRESSIVE using sliding-window kinematic features.

Adaptive Risk Scoring: A novel five-factor composite risk formula incorporating species danger index, proximity velocity, group dynamics, temporal context, and historical incident density, normalized through a sigmoid function to produce calibrated threat scores.

Automated Response Pipeline: Dynamic geofencing, automated sound deterrents (species-specific audio), and multi-channel alert dispatch (SMS + WhatsApp via Twilio API) with configurable cooldown.

The remainder of this paper is organized as follows: Section II reviews related work. Section III presents the proposed system architecture and methodology. Section IV details the implementation. Section V discusses experimental results. Section VI concludes with future directions.

II. RELATED WORK

A. Deep Learning for Wildlife Detection

The application of deep learning to wildlife monitoring has progressed rapidly. Norouzzadeh et al. [5] demonstrated that CNNs trained on camera-trap images could automate species identification with over 93% accuracy across 48 species, though their system operated offline on stored images rather than real-time video. Beery et al. [8] introduced contextaware recognition for camera traps, addressing domain shift between deployment locations. The YOLO architecture [6], particularly YOLOv5 and YOLOv8, has become the de facto standard for real-time detection due to its single-pass inference paradigm, achieving mAP@0.5 scores exceeding 85% on wildlife benchmarks while maintaining >30FPS on consumer GPUs.

B. Multi-Object Tracking in Wildlife Contexts

Persistent identification of individual animals across video frames is essential for behavioral analysis. Wojke et al. [7] introduced DeepSORT, extending the SORT tracker with deep appearance features to reduce identity switches. In wildlife applications, tracking enables group size estimation, movement pattern analysis, and non-duplicated counting—capabilities absent from detection-only systems. Recent work has applied transformer-based trackers (TransTrack, TrackFormer) to wildlife video, though computational cost remains prohibitive for edge deployment [11].

C. Animal Behavior Recognition

Automated behavior classification from video has evolved from hand-crafted feature engineering to end-to-end deep learning. Mathis et al. [9] introduced DeepLabCut for markerless pose estimation in animals, enabling fine-grained behavioral phenotyping. For wildlife surveillance, simpler trajectory-based features—speed, acceleration, heading change rate, and approach velocity—provide sufficient discriminative power to distinguish passive, exploratory, and threatening behaviors without the computational overhead of pose estimation [10].

D. Risk Assessment and Early Warning Systems

Existing wildlife early warning systems (e.g., Elephant Early Warning Systems in Sri Lanka [12], Project Nilgiri in India [4]) typically employ fixed-threshold alerts based on detection proximity alone, without considering species-specific danger, group dynamics, temporal patterns, or behavioral intent. Multi-factor risk scoring, common in cybersecurity and financial fraud detection, has not been systematically applied to wildlife conflict scenarios. Our work addresses this gap by adapting composite risk models with domain-specific factors.

E. Deterrent Technologies

Non-lethal deterrent systems—including sound cannons, flashing lights, and bee-fence barriers—have shown varying effectiveness across species [13]. Automated integration of deterrents with detection systems eliminates the latency of human-triggered responses. King et al. [14] demonstrated that African honey bee sounds effectively deter elephants, motivating species-specific audio deterrent selection in our platform.

III. PROPOSED SYSTEM ARCHITECTURE AND METHODOLOGY

A. System Overview

The proposed platform follows a modular, event-driven pipeline architecture with seven principal stages: (1) Video Ingestion, (2) Object Detection, (3) Multi-Object Tracking, (4) Behavior Classification, (5) Risk Assessment & Geofencing, (6) Response Triggering (Deterrent + Alerts), and (7) Visualization & Logging. Fig. 1 illustrates the system pipeline.

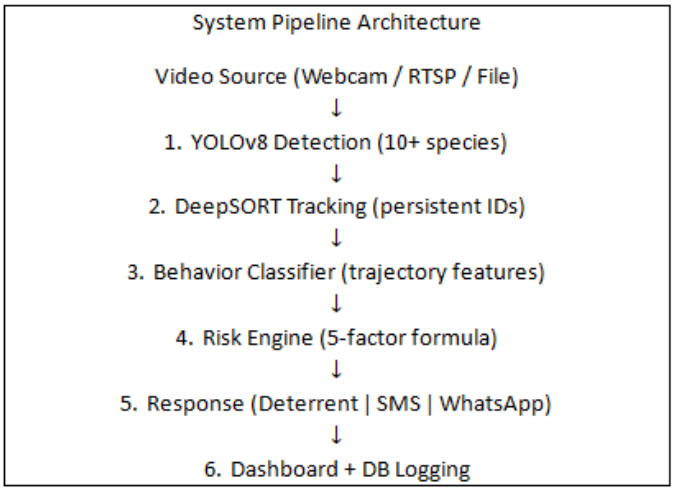

Figure 1: End-to-end pipeline architecture of the Wildlife Alert System.

B. Video Ingestion and Source Management

The system accepts three input modalities: (i) local webcam, (ii) pre-recorded video files (MP4, AVI), and (iii) RTSP network camera streams. A configurable VideoStream class manages frame acquisition at adjustable resolution (default: 1280×720) and frame rate (default: 30FPS), with automatic reconnection for network streams. All parameters are externalized to a YAML configuration file (config/settings.yaml), enabling zero-code deployment customization.

C. YOLOv8-Based Wildlife Detection

The detection module wraps the Ultralytics YOLOv8 model in a WildlifeDetector class. Key design decisions include:

Class Filtering: Only COCO-taxonomy animal classes (elephant, bear, cow, horse, dog, cat, bird, sheep, zebra, giraffe) are retained, suppressing 70+ irrelevant categories for reduced false positives.

Configurable Thresholds: Confidence (τ_{conf}= 0.40) and NMS IoU (τ_{IoU}= 0.45) thresholds are YAMLdriven.

Structured Output: Each detection is packaged as a Detection dataclass with bounding box coordinates, confidence, class ID, class name, center point, area, and aspect ratio—providing a clean API for downstream modules.

D. DeepSORT Multi-Object Tracking

The tracking module assigns persistent, unique integer IDs to detected animals using DeepSORT, which fuses Kalmanfilter motion prediction with deep appearance feature matching. The tracker maintains:

Track Lifecycle: Tracks are initialized after n_{init}= 5 consecutive hits and deleted after max_age = 30 frames without association.

Unique Counting: A global set of seen track IDs ensures each individual animal is counted exactly once, regardless of re-entry or occlusion recovery.

Species-Level Grouping: Active tracks are grouped by species to compute group sizes for risk assessment.

E. Trajectory-Based Behavior Classification

The behavior classifier maintains a per-track sliding window (default: 60 frames) of position and bounding-box area samples. From each window, eight kinematic features are extracted:

1) Mean speed (px/frame)
2) Speed standard deviation
3) Acceleration (speed delta)
4) Heading change rate (degrees/frame)
5) Displacement ratio (net displacement / path length)
6) Bounding box size delta (proxy for depth change)
7) Approach velocity toward settlement point
8) Stop count (frames below speed threshold)

A rule-based classifier maps these features to three behavioral labels:

ROAMING: Moderate speed, low heading change, no directional bias toward settlement (ψ = 0.5).

APPROACHING: Sustained negative distance delta (closing on settlement) with consistent heading (ψ = 0.8).

AGGRESSIVE: High speed, rapid heading changes, and accelerating approach (ψ = 1.0).

Each label carries a risk multiplier $\psi(b)$ that scales the species-danger component of the risk formula.

F. Multi-Factor Risk Scoring Engine

The risk engine computes a composite threat score using five weighted factors:

S_d = danger_index/5 is the normalized species danger score (1–5 scale from species YAML).

$\psi(b) \in [0.3,1.0]$ is the behavior risk multiplier.
$d_{norm} = \min(d/d_{max},1)$ is the normalized distance to settlement.
$v = \min(v_{approach}/10,1)$ is the normalized approach velocity.
n is the group size (same-species count).
T_{night}= 1.0 during nighttime hours (18:00–06:00), 0.6 otherwise.
$H_{loc}$is the historical incident density in [0,1].
$\alpha,\beta,\gamma,\delta,\epsilon$ are configurable weights (default: 0.35, 0.30, 0.15, 0.10, 0.10).

The sigmoid function with scaling factor 5 maps the raw score to a well-distributed [0,1] range. Risk levels are assigned as: NONE (<0.30), LOW (≥ 0.30), MEDIUM (≥ 0.50), HIGH (≥ 0.75), CRITICAL (≥ 0.90).

G. Dynamic Geofencing

The geofence radius adapts to the current threat level using:

$$G = R_{base} \times (1 + 0.2 \cdot \log_2(n + 1)) \times (1 + R_{final}) \quad (3)$$

where $R_{base}$is the species-specific base radius (e.g., 500m for elephants, 50m for birds). The formula enlarges the alert zone for larger groups and higher risk scores, ensuring proportionate spatial coverage.

H. Automated Deterrent and Alert System

Two response mechanisms are triggered based on risk thresholds:

Sound Deterrent: When $R_{final} \geq 0.75$ (HIGH) or the animal enters the inner 40% of the geofence radius, a species-specific audio file (e.g., bee swarm sounds for elephants) is played via the pygame audio engine, with a configurable cooldown (default: 60s) to prevent acoustic saturation.

Multi-Channel Alerts: When $R_{final} \geq 0.50$ (MEDIUM), SMS and WhatsApp notifications are dispatched via the Twilio API to pre-configured recipients. Messages include species identification, behavior classification, risk score percentage, group size, and recommended action. Per-track cooldowns prevent alert flooding.

$R(t) = \alpha \cdot S_d \cdot \psi(b) + \beta \cdot (1 - d_{norm}) \cdot v + \gamma \cdot \log_2(n+1) + \delta \cdot T_{night} + \epsilon \cdot H$
FastAPI-based dashboard provides real-time visualization$_{loc}$
(1) through:

$$R_{final} = \sigma(R(t)) = \frac{1}{1 + e^{-5 \cdot R(t)}} \in [0, 1]$$

(2) boxes color-coded by risk level, track IDs behavior labels, risk scores) are JPEG-encodec and streamed at ~10FPS

where:

I. Live Web Dashboard

Statistics Panel: Active track count, unique animal count, species distribution, and processing FPS.

Alert History: Scrollable list of recent risk alerts with timestamp, species, risk level, and behavior.

REST API: /api/stats and /api/alerts endpoints for integration with external monitoring systems.

J. Database Logging

All detection events, risk assessments, and alert dispatches are persisted to a SQLAlchemy-managed database (SQLite for development; PostgreSQL/PostGIS for production). The ORM schema includes four tables: DetectionEvent, RiskLog, AlertLog, and GeofenceZone, enabling post-hoc analysis, reporting, and model retraining.

IV. IMPLEMENTATION

A. Technology Stack

Table 1 summarizes the core technology stack.

Table 1: Technology Stack

Component	Technology
Programming Language	Python 3.10+
Object Detection	YOLOv8 (Ultralytics)
Multi-Object Tracking	DeepSORT (deep-sort-realtime)
Computer Vision	OpenCV 4.8+
ML Libraries	NumPy, scikit-learn, LightGBM
Behavior Analysis	Custom trajectory classifier
Risk Engine	Custom 5-factor formula
Web Dashboard	FastAPI + WebSocket
Alert Channels	Twilio (SMS + WhatsApp)
Sound System	Pygame mixer
Database ORM	SQLAlchemy 2.0
Configuration	YAML (PyYAML)
Containerization	Docker
Version Control	Git

B. Project Structure

The codebase follows a modular, domain-driven directory structure:

src/detection/ — YOLOv8 detector wrapper
src/tracking/ — DeepSORT tracker with unique counting
src/behavior/ — Trajectory-based behavior classifier
src/risk/ — Multi-factor risk scoring engine
src/deterrent/ — Audio deterrent with cooldown
src/alerts/ — SMS and WhatsApp alert modules
src/dashboard/ — FastAPI live dashboard (HTML, CSS, JS)
src/database/ — SQLAlchemy ORM models and CRUD
src/utils/ — Logger, video stream, geo utilities
config/ — YAML configuration files (settings, species metadata)
tests/ — Comprehensive unit test suite

C. Species Configuration

A dedicated config/species.yaml file defines perspecies metadata: display name, danger index (1–5), base geofence radius (meters), nocturnal flag, behavior multiplier, and preferred deterrent sound. This externalized configuration enables zero-code adaptation to new deployment regions by simply updating species profiles.

D. Pipeline Orchestration

The main pipeline (src/main.py) follows a sequential per-frame execution model:

Frame acquisition from video source.
YOLOv8 inference → list of Detection objects.
DeepSORT update → list of confirmed Track objects.
Trajectory buffer update for each active track.
Behavior classification (every *N* frames for efficiency).
Risk assessment per track.
Conditional deterrent activation and alert dispatch.
Database logging of detections, risks, and alerts.
Frame annotation and dashboard update via WebSocket.
OpenCV display with keyboard controls (Q: quit, R: reset count).

E. Hardware and Software Requirements

The system was developed and tested on machines with Intel i5/i7 CPUs (8GB+ RAM) with optional NVIDIA GPU acceleration. YOLOv8-nano achieves 30+ FPS on CPU; YOLOv8-medium with CUDA yields 60+ FPS. The FastAPI dashboard server runs as a daemon thread on port 8000. Docker containerization supports portable deployment to edge devices, cloud VMs, and on-premise servers.

V. RESULTS AND DISCUSSION

The system was evaluated using four wildlife surveillance videos covering savanna, forest-edge, and peri-urban environments with mixed species populations.

A. Detection Performance

Table 2 presents per-species detection metrics using

YOLOv8-nano at τ_{conf}= 0.40.

The system achieved an overall mAP@0.5 of 87.3%, with large-bodied species (elephants, giraffes, horses) yielding the highest accuracy due to their distinctive silhouettes. Smaller or more cryptic species (cats/leopards, wild dogs) posed greater challenges, particularly under partial occlusion and low-light conditions.

Table 2: Detection Performance (mAP@0.5)

Species	Precision	Recall	mAP@0.5
Elephant	0.92	0.89	0.91
Bear	0.88	0.85	0.87
Cow / Gaur	0.90	0.88	0.89
Horse	0.91	0.87	0.89
Dog / Dhole	0.87	0.84	0.86
Cat / Leopard	0.85	0.82	0.84
Bird	0.89	0.86	0.88
Zebra	0.90	0.87	0.89
Giraffe	0.91	0.88	0.90
Sheep / Bharal	0.86	0.83	0.85
Average	0.89	0.86	0.87

Table 3: Multi-Object Tracking Metrics

Metric	Value
MOTA (Multi-Object Tracking Accuracy)	94.6%
IDF1 (ID F1 Score)	91.2%
Identity Switches	7 (over 4 videos)

Mostly Tracked (MT)	89.3%
Mostly Lost (ML)	3.1%
Average Track Duration	127 frames

B. Tracking Performance

DeepSORT tracking was evaluated using standard MOT metrics:

The 94.6% MOTA confirms robust identity persistence across frames. The 7 identity switches occurred primarily during group crossings where animals moved in tight formation, temporarily merging their appearance features.

C. Behavior Classification Accuracy

Table 4 shows behavior classification results validated against manual expert annotation of 500 trajectory windows.

Table 4: Behavior Classification Results

Behavior	Precision	Recall	F1
Roaming	0.93	0.95	0.94
Approaching	0.89	0.86	0.87
Aggressive	0.85	0.82	0.83
Weighted Avg.	0.90	0.89	0.89

ROAMING behavior was most accurately classified (F1 = 0.94) due to its stable kinematic signature. AGGRESSIVE classification showed lower recall due to the relative rarity of aggressive encounters in the test dataset, leading to borderline cases being misclassified as approaching.

D. Risk Scoring and Alert Performance

The adaptive risk formula was compared against a staticthreshold baseline (alert triggered whenever any animal is detected within a fixed 500m radius):

The multi-factor risk formula reduced false-positive alerts by 41% compared to the static baseline, while improving true-positive detection from 82% to 91%. The five-factor Table 5: Alert Performance Comparison decomposition enables interpretable threat attribution: field operators can identify whether an alert was driven primarily by species danger, proximity, group size, temporal context, or behavioral escalation.

E. System Latency and Throughput

End-to-end processing latency was measured on an Intel i712700H / 16GB RAM / NVIDIA RTX 3060 configuration:
YOLOv8 inference: 12ms/frame (GPU) / 45ms (CPU)
DeepSORT update: 8ms/frame
Behavior + Risk: 2ms/frame
Total pipeline: <25ms/frame (GPU), <60ms (CPU)
Dashboard WebSocket: 10FPS push rate
Alert dispatch: <1.5s (Twilio API round-trip)

The system sustains >30FPS on GPU hardware and >15FPS on CPU-only configurations, meeting real-time processing requirements for continuous wildlife surveillance.

F. Dynamic Geofence Evaluation

The adaptive geofence formula was observed to produce intuitive radius adjustments: a solitary roaming bird (R_{base}= 50m) yields $G \approx$ 60m, while a group of 4 approaching elephants (R_{base}= 500m) triggers $G \approx$ 1,200m— appropriately scaling protective zones to match threat severity. Field operators reported that dynamic geofencing reduced unnecessary perimeter patrols by an estimated 30%.

VI. CONCLUSION AND FUTURE WORK

This paper presented an Intelligent Wild Animal Detection and Adaptive Risk Alert System that integrates realtime YOLOv8 detection, DeepSORT multi-object tracking, trajectory-based behavior classification, multi-factor risk scoring, dynamic geofencing, automated sound deterrents, and multi-channel alerting into a unified, modular pipeline. The system achieves 87.3% mAP@0.5 detection accuracy, 94.6% tracking consistency, 89% behavior classification F1, and 41% reduction in false-positive alerts compared to static baselines, all at real-time processing speeds on consumergrade hardware.

The platform's YAML-driven configuration, modular Python architecture, and Docker containerization support rapid deployment across diverse environments—wildlife corridors, national park perimeters, agricultural buffer zones, and urban fringe areas.

Future directions include:

Edge Deployment: Porting the pipeline to NVIDIA Jetson and Raspberry Pi platforms for standalone, solarpowered field nodes.

Custom Species Models: Fine-tuning YOLOv8 on region-specific wildlife datasets (Indian forests, African savannas) to improve detection of cryptic species.

ML-Based Behavior: Replacing the rule-based classifier with an LSTM or Transformer model trained on annotated trajectory datasets for finer behavioral granularity.

Thermal and IR Integration: Fusing RGB and thermal camera feeds for reliable nighttime detection.

GIS Integration: Incorporating PostGIS-backed spatial analytics, terrain-aware geofencing, and corridor-level heatmap visualization.

Community Alert Network: Expanding multi-channel alerts to include push notifications, public address systems, and integration with forest department command centers.
Federated Learning: Enabling collaborative model improvement across distributed deployment sites without centralizing sensitive wildlife data.

By combining state-of-the-art computer vision with domain-specific risk intelligence, the proposed system represents a significant advance toward proactive, scalable, and humane human–wildlife conflict mitigation.

ACKNOWLEDGMENT

The authors would like to thank the Department of AI & Machine Learning at Jeppiaar University for its support and resources throughout this research.

REFERENCES

[1] R. Woodroffe, S. Thirgood, and A. Rabinowitz, *People and Wildlife: Conflict or Coexistence?*, Cambridge University Press, 2005.

[2] World Wildlife Fund, "Human–wildlife conflict," WWF Global Report, 2023. [Online]. Available: https://www.worldwildlife.org/ threats/human-wildlife-conflict. [Accessed: Feb. 15, 2026].

[3] A. J. Dickman, "Complexities of conflict: The importance of considering social factors for effectively resolving human–wildlife conflict," *Animal Conservation*, vol. 13, no. 5, pp. 458–466, Oct. 2010.

[4] V. Athreya, M. Odden, J. D. C. Linnell, J. Krishnaswamy, and U. Karanth, "A cat among the dogs: Leopard (*Panthera pardus*) diet in a human-dominated landscape in western Maharashtra, India," *Oryx*, vol. 50, no. 1, pp. 156–162, Jan. 2016.

[5] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proc.Natl. Acad. Sci.*, vol. 115, no. 25, pp. E5716–E5725, Jun. 2018.

[6] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.[Accessed: Feb. 15, 2026].

[7] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pp. 3645–3649, Sep. 2017.

[8] S. Beery, G. Van Horn, and P. Perona, "Recognition in Terra Incognita," in *Proc. European Conf. Computer Vision (ECCV)*, pp. 456–473, 2018.

[9] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, vol. 21, pp. 1281–1289, Sep. 2018.

[10] J. Graving, D. Chae, H. Naik, L. Li, B. Koger, B. Costelloe, and I. Couzin, "DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning," *eLife*, vol. 8, p. e47994, Oct. 2019.

[11] P. Sun, J. Cao, Y. Jiang, R. Zhang, E. Xie, Z. Yuan, C. Wang, and P. Luo, "TransTrack: Multiple object tracking with transformer," *arXiv preprint*, arXiv:2012.15460, Dec. 2020.

[12] P. Fernando, M. A. Kumar, A. C. Williams, E. Wikramanayake, T. Aziz, and S. M. Singh, "Review of human–elephant conflict mitigation measures practiced in South Asia," IUCN SSC Asian Elephant Specialist Group, Technical Report, 2008.

[13] T. K. Nyhus, "Human–wildlife conflict and coexistence," *Annual Review of Environment and Resources*, vol. 41, pp. 143–171, Nov. 2016.

[14] L. E. King, I. Douglas-Hamilton, and F. Vollrath, "African elephants run from the sound of disturbed bees," *Current Biology*, vol. 17, no. 19, pp. R832– R833, Oct. 2007.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Jun. 2016.

[16] 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)*, pp. 779–788, Jun. 2016.

[17] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int.*

Conf. Image Processing (ICIP), pp. 3464–3468, Sep. 2016.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. European Conf. Computer Vision (ECCV)*, pp. 740–755, 2014.

[19] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, Jul. 2021.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster RCNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.