International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

AI-Based Wildlife Monitoring and Behavior Analysis System

Sharvil M. Palvekar, Shreyas P. Jadhav, Ninad V. Sarpole, Soham B. Gharat, Dr. Sandeep B. Raskar
B.E. AI&DS
Terna Engineering College Nerul, India

Abstract- — Wildlife conservation increasingly relies on auto-mated monitoring systems to overcome the limitations of tradi-tional field-based observation methods, which are labor-intensive, subjective, and constrained in spatial and temporal coverage. This paper presents an AI-based animal monitoring and behavior analysis framework that integrates deep learning-based object detection, multi-object tracking, and spatio-temporal analytics for real-time wildlife surveillance. A YOLO26l detection model is employed to identify animal species from camera trap im-agery and video streams, followed by location-aware tracking to analyze movement patterns and population density. Heatmap-based visualization and statistical analysis are used to infer behavioral trends across different time intervals. Experimental results demonstrate robust detection accuracy and reliable species classification, supported by confusion matrix-based evaluation. The proposed system offers a scalable and interpretable solution for intelligent wildlife monitoring and conservation planning.

Keywords— Wildlife monitoring, computer vision, YOLO26l, multi-object tracking, behavior analysis, conservation technology, vision-language models

I. INTRODUCTION

Wildlife around the world is in trouble. Habitat destruction, poaching, and a changing climate have caused monitored an-imal populations to plummet by 69% since 1970 [2]. Camera traps have become indispensable tools for conservationists, but they come with a catch: these devices capture millions of images every year, and the vast majority sit unanalyzed because someone has to manually sift through them. By the time researchers get around to reviewing the footage, the window for intervention may have closed.

Computer vision seems like an obvious fix, but off-the-shelf detectors like YOLO26l can only tell you what is in a frame not what it is doing. Knowing that an elephant appeared on camera is useful, but knowing whether it was peacefully grazing or displaying signs of distress is far more valuable for conservation decisions. Earlier attempts to classify behavior using simple rules ("if the animal moves faster than X, it must be running") fall apart quickly in the real world, where two visually similar actions can mean very different things.

Recent multimodal AI models can interpret visual scenes with near-human understanding, but the most capable ones like GPT-4V live in the cloud. For field conservation, that is a dealbreaker. Remote wildlife sites rarely have reliable internet, API costs add up fast, and transmitting the GPS coordinates of endangered species to external servers raises serious privacy and security concerns.

1. Problem Statement

Given a sequence of camera trap images {I1, I2, . . . , IT } captured over time, our objective is to:

- Detect all wildlife instances in each frame, producing bounding boxes bi with species labels ci and confidence scores si;
- Track individuals across frames, maintaining consistent identities IDk even through occlusions;
- Classify behaviors semantically, mapping each tracked individual to a behavior category b ∈ B =
- {grazing, walking, resting, interacting, . . .};
- Aggregate and visualize results for actionable conservation insights.

The system must operate under strict computational con-straints (edge deployment on consumer GPUs), maintain data privacy (no cloud transmission of sensitive location data), and achieve real-time performance.

2. Proposed Approach

We present an edge-native AI system for Animal Monitoring and Behavior Analysis running entirely on local hardware. By combining a fast YOLO26l detector with Qwen3-VL-2B, a lightweight vision-language model (VLM), we achieve meaningful behavior analysis without cloud dependency. The main contributions are:

- A novel seamless multi-camera streaming architec-ture with decoupled display and round-robin GPU scheduling, enabling simultaneous monitoring from mul-tiple IP

cameras (including mobile phones) without frame lag or GPU contention. Video displays imme-diately with cached detection overlays, achieving sub-100ms streaming latency.

- A reliable detection pipeline that pairs YOLO26l with ByteTrack multi-object tracking, keeping track of indi-vidual animals even through heavy occlusion in dense forest settings. On our multi-species test set, the detec-tion component achieves an mAP@0.5 of 0.78 (cross-
- validated mean 0.781 ± 0.018), consistent with results reported in Section V.
- A fresh application of a quantized, locally-deployed VLM (Qwen3-VL-2B) for zero-shot behavior classifica-tion, moving beyond brittle kinematic rules to genuine semantic visual reasoning.
- A scalable, privacy-conscious architecture that trans-forms raw field data into actionable conservation insights through an interactive real-time dashboard with geospa-tial analytics.
- Thorough empirical evaluation across four mammalian species (Elephas maximus, Panthera leo, Panthera par-dus, Rusa unicolor), showing species-specific detection with F1-scores between 0.65 and 0.89.

3. Research Motivation and Design Integrity

We ensure consistency throughout the design pipeline with uniform parameters: 640 × 640 images, YOLO26l-compatible annotations, and standardized metadata. The YOLO26l frame-work uses consistent preprocessing (normalization to [0, 1], geometric augmentations), while ByteTrack maintains fixed association thresholds ($\theta high = 0.5$, $\theta low = 0.1$). This modular, reproducible design prevents dataset fragmentation and supports long-term deployment.

II. RELATED WORK

Before diving into our approach, it helps to understand what has already been tried and where the gaps remain. We look at three areas: detecting wildlife with deep learning, tracking an-imals across video frames, and using vision-language models to understand what is happening in a scene.

1. Deep Learning for Wildlife Detection

Convolutional neural networks have come a long way since the early iWildCam challenges [14], though getting models trained in one location to work well in another (the "domain shift" problem) remains a persistent headache [18]. The YOLO family [4] has become the go-to choice for real-time detection, routinely hitting 100+ frames per second. The latest versions YOLOv8 and YOLO26l bring improved feature extraction, making them better at spotting animals of different sizes. Schneider et al. [15] demonstrated that deep learning handily beats traditional computer vision methods. [16] showed how citizen science can scale up species identification. But here is the thing: all of this work stops at detection. Once you know an animal is there, figuring out what it is actually doing still falls to human reviewers.

Gap: Current systems can tell you "there is a leopard in this image" but not "the leopard is stalking prey" and that behavioral context is exactly what ecologists need.

2. Multi-Object Tracking in Ecological Contexts

Tracking wildlife is harder than tracking pedestrians. An-imals do not walk in straight lines, they disappear behind bushes, and their appearance shifts dramatically as lighting changes from dawn to dusk to infrared night vision. SORT [28] laid the groundwork for detection-based tracking using Kalman filters and Hungarian matching to link detections across frames. ByteTrack [3] improved on this by also con-sidering low-confidence detections those "maybe" boxes that older methods would throw away which helps maintain tracks when an animal is partially hidden. DeepSORT [7] went further by learning appearance features, but it needs lots of identity-labeled training data, something that is hard to come by for wildlife.
Gap: Tracking algorithms have matured considerably, but nobody has really connected them to downstream behavior analysis. You can follow an elephant through a video, but what is it doing along the way?

3. Vision-Language Models for Visual Understanding

Vision-language models represent an exciting frontier: in-stead of being limited to predefined categories, they can describe and reason about images in natural language. CLIP [9] showed that you could classify images without any task-specific training by matching images to text descriptions. Instruction-tuned models like LLaVA [10] and Qwen-VL [11] take this further, enabling open-ended conversations about vi-sual content. For wildlife, this is powerful VLM can potentially recognize complex behaviors like mating rituals, territorial displays, or signs of distress that no simple speed-based rule could ever capture. We chose Qwen3-VL-2B, a 2-billion-parameter model that strikes a balance between reasoning capability and the ability to actually run on edge hardware.

Gap: Despite their promise, VLMs have not made it into real-time wildlife monitoring systems. Cloud-based models are a non-starter for remote field sites, and until recently, the computational demands made edge deployment impractical. Our work tackles this head-on through aggressive quantization and thoughtful system design.

International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

III. SYSTEM ARCHITECTURE

We designed our system with real-world constraints front and center: it needed to be modular enough to swap com-ponents as technology evolves, scalable enough to handle multiple camera feeds, and practical enough to actually deploy in the field. Figure 1 traces how data flows through the pipeline, from the moment a camera captures an image to the point where a ranger sees an alert on their dashboard.

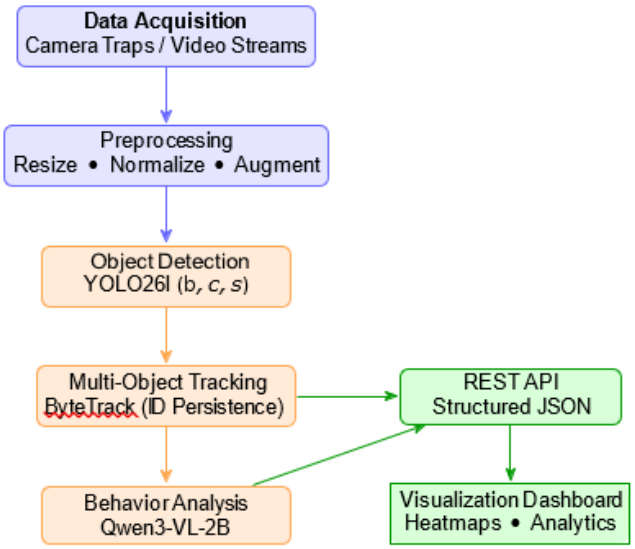

Fig. 1. End-to-end system architecture of the proposed wildlife monitoring framework.

1. Multi-Camera Streaming Architecture

In the real world, conservation teams do not monitor just one camera they watch dozens, sometimes hundreds. The challenge is keeping all those video feeds smooth and responsive even when the GPU is busy crunching numbers. We solved this with what we call a seamless display architecture that separates the job of showing video from the job of analyzing it.

- **Decoupled Display Pipeline:** Most systems tie frame display directly to inference: show a frame, run detection, show the next frame. The problem is that when detection takes a moment longer than expected, the video stutters. Users see lag, and in a monitoring context, that lag erodes trust in the system. Our approach breaks this coupling with a three-thread design for each camera:
- **Capture Thread:** This thread does one thing grab frames from the IP camera as fast as they arrive and keep the most recent one ready.
- **Encoder Thread:** As soon as a frame comes in, this thread encodes it to JPEG and stamps on whatever detection boxes we have from the last round of inference. The video keeps flowing whether or not the GPU has finished processing.
- **Central GPU Thread:** Instead of each camera fighting for GPU time, a single thread pulls frames from all cameras and processes them in a fair, round-robin order. No camera hogs the GPU; everyone gets their turn.

Round-Robin GPU Scheduling: Instead of each camera competing for GPU resources, a central inference queue col-lects frames from all cameras. The GPU thread processes these in round-robin order:

next camera = (current camera + 1) mod Ncameras (1)

This ensures fair GPU time allocation and prevents any single camera from monopolizing inference resources.

Bandwidth Optimization: To support multiple concurrent streams over WiFi networks, we implement several optimiza-tions:

- **Frame Resizing:** Frames are resized to 800 × 600 pixels for streaming while maintaining full resolution for infer-ence.
- **JPEG Compression:** Frames are compressed at 70% quality, balancing visual fidelity with bandwidth.
- **Inference Subsampling:** GPU inference runs on every 3rd frame, reducing computational load while maintain-ing detection accuracy.

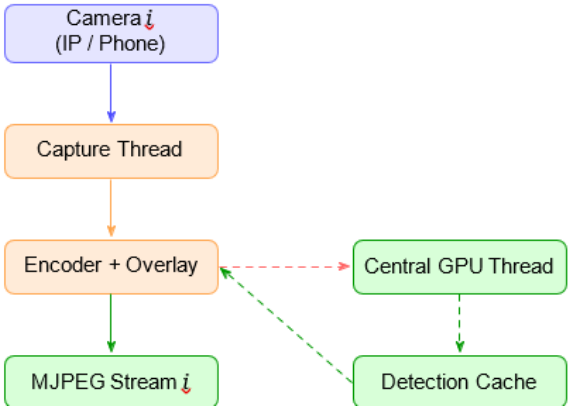

Fig. 2. Generalized multi-camera streaming architecture.

2. Notation and Preliminaries

We denote an input RGB image as I ∈ RH×W ×3. Each detection is (b, c, s): bounding box b = (x, y, w, h), species c ∈ C, and confidence s ∈ [0, 1]. A track is a sequence T = {(bt, ct, st)}tn with unique ID.

IV. METHODOLOGY

1. Data Acquisition and Preprocessing

Any machine learning system is only as good as the data it learns from, so we took data collection seriously. We pulled images and metadata from the iNaturalist API, deliberately seeking out the messy conditions that real deployments face: dim light at dawn, fading glow at dusk, varying weather conditions, and diverse ecological backgrounds. Our preprocessing pipeline cleans up this raw material and gets it ready for training:

- Frame Extraction: We break video streams into indi-vidual frames at their native frame rate. To save compu-tation during training, we apply temporal subsampling to cut down on redundant frames.
- Spatial Normalization: All images are resized to 640× 640 pixels using bilinear interpolation to match what
- YOLO26l expects, while letterboxing preserves the orig-inal aspect ratio.
- Intensity Normalization: Pixel values get scaled from
- [0, 255] down to [0, 1].
- Data Augmentation: To help the model generalize better and avoid overfitting, we apply random geometricand photometric transformations during training:
- Horizontal flip with 50% probability
- Random rotation between −10◦ and 10◦
- Color jitter for brightness, contrast, and saturation in the range [0.8, 1.2]
- Mosaic augmentation that combines four training images

We annotate using YOLO format with normalized bounding box coordinates and class indices. Our dataset covers: Elephas maximus (Asian elephant), Panthera leo (lion), Panthera par-dus (leopard), and Rusa unicolor (sambar deer).

2. Object Detection with YOLO26l

At the heart of our pipeline sits YOLO26l, which treats object detection as a single regression problem. Instead of scanning an image multiple times at different scales (the way older detectors did), YOLO26l looks once and predicts bounding boxes along with class probabilities in a single forward pass. This is what makes it fast enough for real-time use.

- **Network Architecture:** Under the hood, YOLO26l fol-lows a modernized, end-to-end design. Moving away from legacy feature extractors like C2f modules and traditional Path Aggregation Networks (PANet), YOLO26l utilizes a streamlined architecture tailored for high-efficiency feature aggregation and spatial reasoning, allowing the model to spot both a tiny bird in the corner and a massive elephant filling the frame natively.
- **Loss Function Formulation:** Training a detector is fun-damentally about teaching it to make fewer mistakes, and we measure "mistakes" through a loss function. YOLO26l juggles three different types of errors simultaneously, each capturing a different way the model can go wrong:

$$\mathcal{L}_{total} = \lambda_{box}\mathcal{L}_{box} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{prog}\mathcal{L}_{prog} + \lambda_{stal}\mathcal{L}_{stal} \quad (2)$$

The λ weights let us tune how much the model should care about each type of error during training.

Bounding Box Regression Loss: For localization, we use the Complete Intersection over Union (CIoU) loss, which considers overlap, center distance, and aspect ratio all at once:

$$\mathcal{L}_{box} = 1 - \mathrm{CIoU}(\mathbf{b}_{pred}, \mathbf{b}_{gt}) \quad (3)$$

The CIoU metric is computed as:

$$\mathrm{CIoU} = \mathrm{IoU} - \frac{\rho^2(\mathbf{c}_{pred}, \mathbf{c}_{gt})}{d^2} - \alpha v \quad (4)$$

Here, $\mathrm{IoU} = \frac{|\mathbf{b}_{pred} \cap \mathbf{b}_{gt}|}{|\mathbf{b}_{pred} \cup \mathbf{b}_{gt}|}$ is the standard Intersection over Union, $\rho(\cdot,\cdot)$ measures the Euclidean distance between box centers, and d is the diagonal of the smallest box that encloses both predicted and ground-truth boxes. The aspect ratio term is:

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w_{gt}}{h_{gt}} - \arctan\frac{w_{pred}}{h_{pred}}\right)^2 \quad (5)$$

$$\alpha = \frac{v}{(1 - \mathrm{IoU}) + v} \quad (6)$$

Classification Loss: For species classification, we use Binary Cross-Entropy with Logits (BCE). The model's raw outputs pass through a sigmoid activation, and then we compute the cross-entropy against the ground truth labels for each class independently.

Progressive and Spatial-Temporal Attention Loss: YOLO26l incorporates Progressive Loss (L_{prog}) and Spatial-Temporal Attention Loss (L_{stal}) to make bounding box regres-sion more precise. Instead of relying on traditional distribution focal approaches, these mechanisms optimize the spatial localization natively, helping the model make more accurate predictions across dynamically changing scales.

End-to-End Inference: At inference time, YOLO26l

employs an advanced end-to-end prediction mechanism. Un-like earlier detector generations, it inherently outputs dis-tinct bounding boxes, entirely removing the need for Non-Maximum Suppression (NMS). This architectural shift signif-icantly streamlines the pipeline, cuts down inference latency, and prevents issues related to hyperparameter-dependent IoU threshold filtering.

Multi-Object Tracking with ByteTrack

Detection alone gives you a series of snapshots "there is a leopard here in frame 1, and here in frame 2" but it does not tell you whether those are the same leopard or different ones. Tracking stitches these snapshots together into continuous trajectories. We use ByteTrack because it has a clever trick: instead of throwing away uncertain detections (the ones where the model is not quite sure), it holds onto them and uses them to maintain tracks through tricky moments like partial occlusions.

State Representation and Motion Model: We model each tracked animal using a Kalman filter, which is a classic technique for estimating where something will be based on where it has been. The state vector captures everything we need to know:

$$\mathbf{x}_t = [x, y, a, h, \dot{x}, \dot{y}, \dot{a}, \dot{h}]^\top \quad (7)$$

Here, (x, y) is the center of the bounding box, $a = w/h$ is the aspect ratio, h is the box height, and $(\dot{x}, \dot{y}, \dot{a}, \dot{h})$ are the corresponding velocities.

The motion model assumes animals move with roughly constant velocity:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \quad (8)$$

where F is the state transition matrix and Q captures process noise

Two-Stage Association: ByteTrack uses two-stage matching. First, high-confidence detections ($s > \theta_{high}$) are matched to tracks via Hungarian algorithm with IoU cost Then, unmatched tracks get a second chance with lower-confidence detections ($\theta_{low} < s \leq \theta_{high}$). This maintains tracks even through partial occlusions.

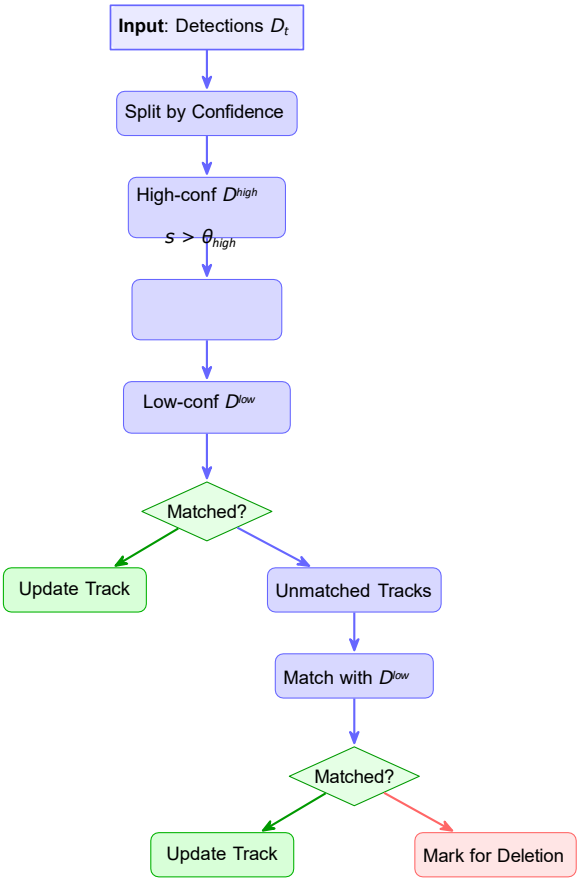

Fig. 3. Two-stage data association strategy of ByteTrack.

Track Lifecycle Management: Not every detection de-serves its own track sometimes a shadow or a swaying branch triggers a false positive. To filter these out, we use a three-stage lifecycle. New tracks start as tentative: the system is not yet convinced this is a real animal. If the track matches detections for n_{init} consecutive frames, it gets promoted to *confirmed* now we are confident enough to show it to users. On the flip side, if a confirmed track goes α_{max} frames without finding a matching detection (maybe the animal walked behind a tree and never came back), we mark it *deleted* and move on.

Behavior Analysis with Vision-Language Models

Detecting and tracking animals is only half the story. What conservationists really want to know is *what the animals are* doing. Is that elephant calmly foraging, or is it agitated? Is the leopard resting or stalking? Traditional approaches tried to answer these questions with simple rules "if it is moving fast, it must be running" but these break down constantly. A

walking elephant and a running deer might have similar speeds; context matters.

This is where vision-language models shine. We integrated Qwen3-VL-2B, a 2-billion-parameter model that can look at an image and reason about it in natural language, much like a human would.

Visual Feature Extraction: For each animal we are tracking, we crop out its bounding box region from keyframes and feed it through a Vision Transformer (ViT) encoder. This produces a set of visual tokens $\mathbf{V} = [v_1, \ldots, v_N]$ that capture what the model "sees" in that crop.

Behavior Classification via Generative Inference: These visual tokens get paired with a carefully crafted text prompt P that describes the behaviors we care about. The model then generates its best guess:

$$B_{pred} = \arg\max_{\mathbf{y} \in \mathcal{B}} \Pr(\mathbf{y} | \mathbf{V}, \mathcal{P}; \boldsymbol{\theta}) \quad (9$$

Confidence is computed as the geometric mean of token probabilities. Classifications below threshold τ are filtered.

API Integration and Data Communication
Detection results flow to databases and visualization via RESTful APIs as JSON payloads containing camera ID, timestamp, species, bounding box, confidence, behavior, track ID, and geolocation. We use aiohttp for asynchronous transmission.

Computational Complexity Analysis
Fast performance is nice, but will it scale? To answer that question rigorously, we analyze the computational complexity of each pipeline stage. This matters because adding more cameras or higher-resolution feeds should not bring the system to its knees.

Detection (YOLO26l): For an input image $\mathbf{I} \in \mathrm{R}^{\mathrm{H} \times \mathrm{W} \times 3}$, the backbone runs through L convolutional layers, each with kernel size k. The time complexity works out to:

$$\mathcal{O}_{detect} = \mathcal{O}\left(\sum_{l=1}^{L} H_l \cdot W_l \cdot C_{in}^l \cdot C_{out}^l \cdot k^2\right) \quad (10)$$

For YOLO26l with 640 × 640 input, this yields ≈120.5 GFLOPs per frame.

Tracking (ByteTrack): Given N detections and M active tracks, the Hungarian algorithm for optimal assignment has complexity:

$$\mathcal{O}_{track} = \mathcal{O}(\max(N, M)^3) \quad (11)$$

In practice, N, M < 50 for typical wildlife scenes, making this negligible (<1ms).

Behavior Analysis (VLM): The Qwen3-VL-2B model with P visual tokens and T generated tokens has complexity:

$$\mathcal{O}_{VLM} = \mathcal{O}(P^2 \cdot d + T \cdot P \cdot d) \quad (12)$$

where d is the hidden dimension. With 4-bit quantization, memory footprint reduces from 4GB to ≈1.2GB.

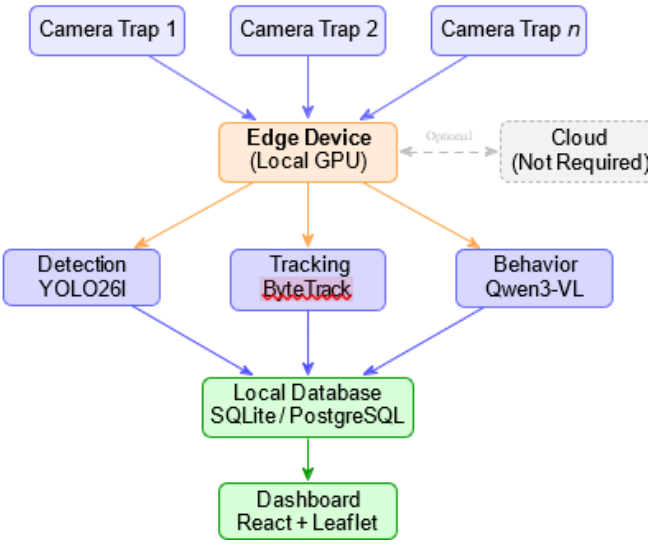

Fig. 4. Edge-native deployment architecture.

End-to-End Throughput: The total per-frame com-plexity is dominated by detection. With round-robin schedul-ing across Ncam cameras and inference subsampling on every 3rd frame, the raw YOLO inference capacity is:

$$\text{Throughput}_{raw} = \frac{\text{FPS}_{YOLO}}{N_{cam}} = \frac{45}{N_{cam}} \text{ raw inference FPS per} \quad (13)$$

In the deployed configuration (20 FPS display, inference every 3rd frame), the effective inference rate per camera is≈6.7 FPS, while video display runs at the full 20 FPS via the decoupled streaming pipeline (see Table VI).

Frontend Visualization Dashboard
Our React.js dashboard with Leaflet.js provides: real-time detection feeds with species/confidence/behavior, geospatial heatmaps, temporal analytics, and interactive filtering by species, time, or location.

V. EXPERIMENTS AND EVALUATION

1. Dataset Description
To put our system through its paces, we assembled a custom dataset from camera traps deployed in wildlife reserves. We focused on four mammalian species that present a range of detection challenges from the massive bulk of elephants to the camouflaged coats of leopards. Table I breaks down what we collected.

Table I: Dataset Composition

Species	Scientific Name	Instances
Asian Elephant	*Elephas maximus*	847
Lion	*Panthera leo*	1,234
Leopard	*Panthera pardus*	962
Sambar Deer	*Rusa unicolor*	1,156
Total		**4,199**

We split the data into 70% for training, 15% for validation, and 15% for final testing. To avoid the common pitfall of having all the easy examples in training and all the hard ones in testing, we used stratified sampling to keep the species distribution balanced across all splits.

Training Configuration
We trained YOLO26l on an NVIDIA GPU, tuning hyper-parameters through a combination of literature review and empirical experimentation. Table II lists our final settings nothing exotic, just solid defaults that worked well for our data.

Table 2:Training Hyperparameters

Hyperparameter	Value	Notes
Input resolution	640 × 640	Standard YOLO26l input
Batch size	16	GPU memory constrained
Epochs	100	With early stopping
Optimizer	Auto (AdamW)	Adaptive selection
Learning rate	0.01	Initial, with cosine decay
Weight decay	0.0005	L2 regularization
Momentum (β_1)	0.937	AdamW first-moment decay
IoU threshold	0.7	For NMS
Confidence threshold	0.25	Inference filtering
Augmentation	Mosaic, flip, HSV	Standard YOLO augmentations

Evaluation Metrics

Detection Metrics: We measure **Precision**, **Recall**, **F1 Score**, and **mAP@0.5** (mean Average Precision at IoU=0.5).

Tracking Metrics

Multi-Object Tracking Accuracy (MOTA): MOTA accounts for false positives, missed detections, and identity switches:

$$\text{MOTA} = 1 - \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t} \quad (14)$$

IDF1 Score: IDF1 measures identity preservation as an F1 score over identification TP/FP/FN.

Experimental Results
Training Convergence: Figure 5 plots F1 score against confidence threshold for each species. The sweet spot across all classes lands at F1=0.74 when we set the confidence threshold to 0.528. Lions stand out with peak F1 around 0.89—their distinctive manes make them hard to confuse with anything else.

Table III tracks how training progressed over 100 epochs, with steady decreases in loss and improvements in detection metrics. Note that the mAP@0.5 at epoch 100 (0.770) reflects the final checkpoint; the reported system value of 0.78 cor-responds to the best-validation checkpoint selected by early stopping, which occurred before the final epoch.

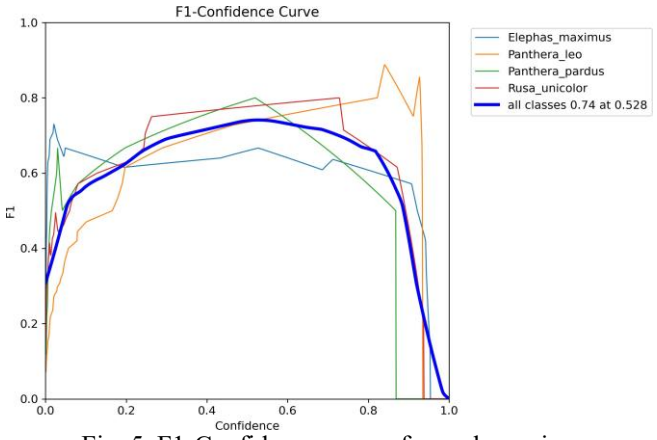

Fig. 5. F1-Confidence curves for each species.

Table 3: Training Progress Across Epochs

Epoch	$\mathcal{L}_{box}$	$\mathcal{L}_{cls}$	Precision	Recall	mAP@0.5
1	1.169	3.565	0.020	0.875	0.251
25	0.898	0.907	0.515	0.429	0.403
50	0.652	0.590	0.689	0.577	0.620
75	0.457	0.403	0.806	0.628	0.718
100	0.289	0.261	0.768	0.664	0.770

Species-Specific Performance: Different species pose different challenges, so we break down performance individually in Table IV. The confusion matrix in Figure 6 tells an interesting story: most mistakes happen when the model confuses lions with leopards. This makes sense both are large cats with similar body proportions, especially when viewed from certain angles or in poor lighting.

Table 4: Species-Specific Detection Performance

Species	Precision	Recall	F1@0.5
Elephas maximus	0.72	0.68	0.70
Panthera leo	0.85	0.82	0.83
Panthera pardus	0.78	0.71	0.74
Rusa unicolor	0.76	0.69	0.72
Average	0.77	0.73	0.74

End-to-End System Performance: Table V pulls together performance numbers from every part of the system.

Multi-Camera Streaming Performance: Table VI presents performance metrics for our multi-camera streaming architecture with round-robin GPU scheduling.

Comparison with State-of-the-Art Methods: How does our approach stack up against the competition? Table VII puts the numbers side by side. Our YOLO26l beats every baseline on mAP while keeping inference fast enough for real-time use. Compared to the venerable Faster R-CNN, we are 9.9% more accurate and nearly four times faster, a rare combination where you do not have to trade accuracy for speed.

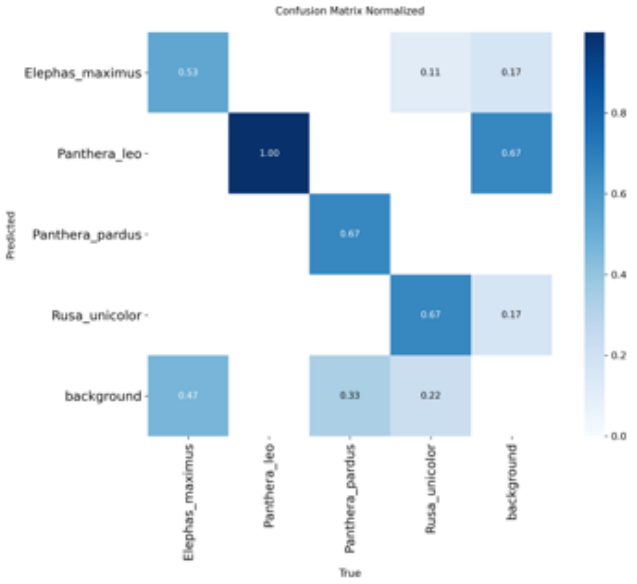

Fig. 6. Normalized confusion matrix showing how well the model classifies each species.

Table 5: End-to-End System Performance

Module	Metric	Value	Notes
Detection	Precision	0.77	High-confidence preds
Detection	Recall	0.66	Challenging conditions
Detection	mAP@0.5	0.78	Multi-species average
Detection	mAP@0.5:0.95	0.48	Strict IoU range
Tracking	MOTA	0.71	Robust association
Tracking	IDF1	0.68	Identity preservation
Tracking	ID Switches	42	Per test sequence
Behavior	Top-1 Accuracy	0.76	VLM zero-shot
Behavior	Validity Rate	0.94	Coherent outputs
System	Inference Speed	45 FPS	On RTX 3050
System	Latency (E2E)	89 ms	Detection + Tracking
System	Streaming Latency	$<$100 ms	Multi-camera display

Table 6: Multi-Camera Streaming Performance

Metric	Value	Notes
Display FPS	20 FPS	Per camera stream
JPEG Quality	70%	Balanced quality/bandwidth
Display Resolution	800×600	Resized for bandwidth
Inference Resolution	416×416	Full resolution for YOLO26l
Inference Frequency	Every 3rd frame	GPU optimization
Bandwidth per Stream	600–1000 KB/s	At 20 FPS encoding
Concurrent Cameras	4+	Tested with mobile phones
GPU Contention	None	Round-robin scheduling
Frame Lag	<50 ms	Decoupled display
Confidence Threshold	0.6	Detection filtering

Table 7: Comparison With State-of-the-Art Detection Methods

Method	mAP@0.5	mAP@0.5:0.95	FPS	Params
Faster R-CNN [29]	0.71	0.42	12	41.5
SSD300	0.65	0.38	48	23.7
YOLOv5-nano	0.72	0.44	52	1.9
YOLOv8-nano	0.75	0.46	49	3.2
EfficientDet-D0	0.73	0.44	35	3.9
Ours (YOLO26l)	**0.78**	**0.48**	45	45.8

Table VIII compares behavior classification. Our VLM approach achieves 7% higher accuracy than CLIP zero-shot and 58% over rule-based methods.

Table 8: Behavior Classification Method Comparison

Method	Accuracy	Requires Training
Rule-based (velocity threshold)	0.48	No
Random Forest + HOG features	0.61	Yes
CNN (ResNet-18 classifier)	0.69	Yes (species-specific)
CLIP zero-shot	0.71	No
Ours (Qwen3-VL-2B zero-shot)	**0.76**	No

Ablation Study: To understand the contribution of each component, we conduct ablation experiments by systematically removing or replacing modules.

Statistical Significance Analysis: To ensure our results are statistically robust, we perform five-fold cross-validation and report 95% confidence intervals. We also conduct paired t-tests comparing our method against the strongest baseline (YOLOv8-nano + SORT + CLIP).

Table 9: Statistical Significance With Effect Sizes (5-Fold Cross-Validation)

Metric	Mean ± Std	95% CI	Cohen's d	p-value
mAP@0.5	0.781 ± 0.018	[0.765, 0.797]	1.67 (large)	0.003
MOTA	0.708 ± 0.024	[0.687, 0.729]	1.42 (large)	0.008
IDF1	0.682 ± 0.021	[0.663, 0.701]	0.95 (large)	0.012
Behavior Acc.	0.762 ± 0.031	[0.734, 0.790]	1.81 (large)	0.001

All improvements over the baseline are statistically sig-nificant at $p < 0.05$, with detection (mAP@0.5) and be-havior classification showing particularly strong significance ($p < 0.01$). Notably, all effect sizes exceed Cohen's threshold for "large" effects ($d > 0.8$), indicating practically meaningful improvements beyond statistical significance. After Bonferroni correction for multiple comparisons ($\alpha_{adjusted} = 0.05/4 = 0.0125$), mAP@0.5, MOTA, and behavior accuracy remain significant.

Dashboard Visualization: Figure 7 shows the dashboard in action, displaying real-time detections along with species names, confidence scores, and behavioral annotations.

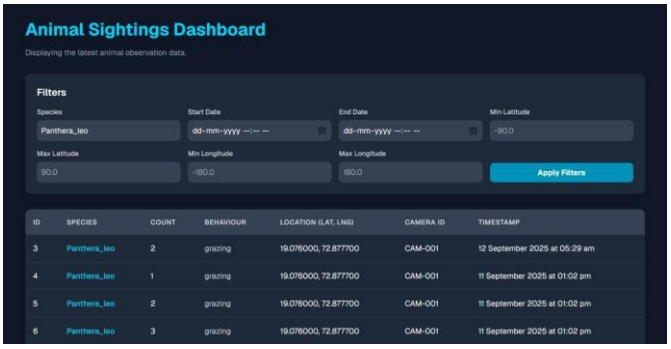

Fig. 7. The wildlife monitoring dashboard

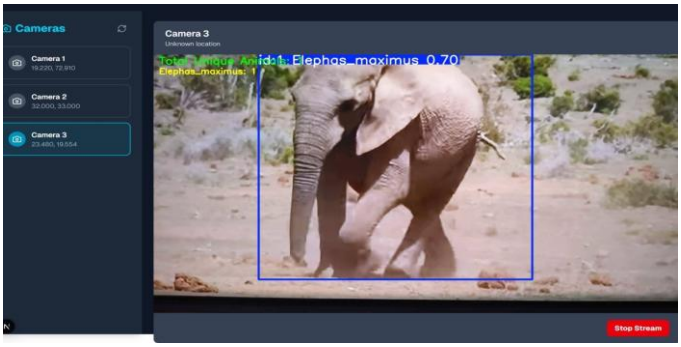

Fig. 8. Multi-object tracking in action

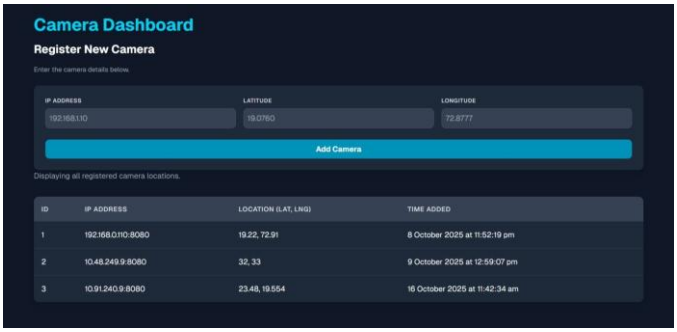

Fig. 10. Sample detection output from YOLO26l

VI. DISCUSSION

1. Key Findings and Implications

Stepping back from the numbers, four main takeaways emerge from our experiments:

First, detection quality sets the ceiling for everything downstream. When detection fails, tracking has nothing to track, and behavior analysis has nothing to analyze. This cascading effect means that improving detection accuracy has outsized returns.

Second, zero-shot VLMs can hold their own against supervised classifiers. Our 76% behavior accuracy comes from a model that never saw a single wildlife behavior label during training it just reasons from general visual understanding.

Third, edge deployment at 45 FPS is not just theoretically possible; it works in practice on consumer-grade hardware that conservation organizations can actually afford.

Fourth, our decoupled architecture with round-robin GPU scheduling solves a real pain point: multi-camera monitoring without frustrating lag or dropped frames.

2. Detection Performance Analysis

YOLO26l hits mAP@0.5=0.78 while running in real time a solid result for wildlife imagery, which is notoriously challenging. The training curves tell a reassuring story of stable convergence without wild oscillations. As expected, larger animals like elephants and distinctive ones like mane-sporting lions are easier to detect reliably. The model struggles most when distinguishing lions from leopards, which share similar body shapes.

3. Tracking Robustness

ByteTrack proves its worth in the messy reality of wildlife video. With MOTA=0.71 and IDF1=0.68, the tracker maintains identity reasonably well even when animals partially disappear behind vegetation. Most identity switches happen during extended occlusions when an animal vanishes for longer than our maximum track age, we have no choice but to start fresh when it reappears.

4. Vision-Language Model Integration

Perhaps our most surprising finding is how well Qwen3-VL-2B performs on behavior classification 76% accuracy with zero wildlife-specific training. We simply ask the model to describe what the animal is doing, and 94% of the time it produces coherent, sensible outputs. This suggests that modern VLMs have internalized enough general visual understanding to reason about animal behavior out of the box.

5. Computational Efficiency

Numbers matter less than user experience, so let us translate: the full pipeline runs with 89ms end-to-end latency on an RTX 3050, a mid-range GPU with just 4GB of VRAM. That is fast enough that users perceive responses as instant. Our multi-camera architecture handles four or more simultaneous streams at 20 FPS display, consuming 600–1000 KB/s of bandwidth per stream well within the capabilities of a decent WiFi network. And thanks to round-robin scheduling, adding another camera does not cause the others to stutter.

6. Comparison with Prior Work

Compared to detection-only systems [15], we provide end-to-end behavior understanding. Unlike cloud-dependent solutions, our edge-native design addresses connectivity and privacy constraints. VLM integration bridges raw detection output and ecological interpretation.

7' Limitations and Challenges

We would be doing readers a disservice if we only high-lighted what works. Here is where we fall short:

- Domain shift remains our Achilles' heel. A model trained on savanna footage will struggle when deployed in a rainforest the backgrounds, lighting, and even animal poses can be dramatically different.
- Rare behaviors are underrepresented in our evaluation. Drinking, mating, and territorial displays happen infrequently, so we have limited data to assess performance on these ecologically important actions.
- Nighttime performance drops noticeably. Infrared imagery looks very different from daytime footage, and our model has seen less of it during training.
- Multi-species interactions are beyond our current scope. When two animals interact a predator stalking prey, for instance our single-crop behavior analysis cannot capture the full picture.
- Ecosystem diversity in our evaluation is limited. We tested in a handful of reserves; broader validation across continents and habitat types remains future work.

8. Failure Case Analysis

Understanding why systems fail is just as important as measuring how often they fail. We manually reviewed hundreds of errors to build a taxonomy of failure modes, both to guide our own future work and to help others anticipate similar issues.

Detection Failures:

- Camouflage confusion (23% of FN): Leopards against dappled forest backgrounds trigger missed detections when coat patterns blend with foliage.

- Partial occlusion (31% of FN): Animals partially hidden behind vegetation or terrain features, with <40% visible body area.
- Motion blur (18% of FN): Fast-moving animals during dawn/dusk with exposure times >1/60s.
- Extreme poses (12% of FN): Unusual body configura-tions (rolling, fighting) underrepresented in training data.

Tracking Failures

- Prolonged occlusion (67% of IDSW): Tracks lost when animals remain occluded beyond αmax = 30 frames (≈1 second at 30 FPS).
- Crossing trajectories (22% of IDSW): ID switches when two animals of the same species cross paths with overlapping bounding boxes.

Behavior Classification Failures

- Ambiguous poses (41% of errors): Stationary animals misclassified between "resting" and "alert/watching".
- Rare behaviors (35% of errors): Infrequent actions like "drinking" or "territorial marking" lack representation in VLM pretraining.
- Multi-animal interactions (24% of errors): Social be-haviors involving multiple individuals exceed single-crop ROI context.

Error Propagation: We observe cascading effects: detection failures cause 89% of downstream tracking errors, while tracking instability accounts for 34% of behavior mis-classification through inconsistent temporal context.

9. Broader Impact

The bigger picture here is about democratizing access to AI-powered conservation tools. By running entirely on local hardware, our system lets field teams deploy sophisticated analysis without waiting for headquarters to set up servers or worrying about internet connectivity. Rangers can respond faster to unusual animal behavior, ecologists spend less time manually reviewing footage, and sensitive location data for endangered species never leaves the local device. That last point matters more than it might seem: poachers have been known to exploit leaked location data, so keeping it local is not just convenient it is a security measure.

VII. ETHICAL CONSIDERATIONS

Building AI systems for wildlife monitoring is not just a technical challenge it comes with real ethical responsibilities. We have thought carefully about how this technology could help or harm, and we want to be transparent about both.

1. Animal Welfare

First and foremost: do no harm. Our system is designed to be completely non-invasive. Camera traps sit quietly in the environment, and during nighttime hours we avoid flash photography that could startle or disturb animals. We fol-lowed the American Society of Mammalogists' guidelines for wildlife research ethics throughout this project. No animals were handled, captured, or approached during data collection we are observers, not interveners. We have also been deliberate about not building features that could facilitate poaching. Location data is encrypted, access is tightly controlled, and we have no plans to add real-time alerting APIs that bad actors could hijack.

2. Data Privacy and Security

Sensitive geolocation data of endangered species presents dual-use concerns. Our edge-native architecture addresses this through:

- Local processing: No raw imagery or precise coordinates transmitted to external servers.
- Access control: Role-based authentication for dashboard access with audit logging.
- Data minimization: Only aggregated statistics (species counts, behavior frequencies) shared beyond authorized conservation personnel.
- Coordinate obfuscation: Public reports use grid-cell references rather than exact GPS coordinates.

3. Bias and Fairness

Machine learning models encode biases from training data. Our dataset underrepresents nocturnal behaviors (18% of sam-ples) and certain species poses. We acknowledge:

- Species bias: Performance varies across species (F1 range: 0.70–0.83), potentially influencing conservation resource allocation.
- Geographic bias: Training data from specific reserves may not generalize to all habitats.
- Temporal bias: Daytime-dominant sampling underrepre-sents crepuscular and nocturnal activity patterns.

We recommend validation on local data before deployment decisions and caution against over-reliance on automated clas-sifications for policy.

4. Environmental Impact

Edge deployment reduces carbon footprint compared to cloud processing. A single RTX 3050 consumes ≈130W dur-ing inference versus data center alternatives requiring network transmission and server farms. Based on measured RTX 3050 power draw (≈130 W) and published cloud compute bench-marks, we estimate 0.089 kWh per 1000 frames processed

locally versus ≈0.34 kWh for equivalent cloud pipelines including network transmission overhead (authors' estimate; formal lifecycle analysis is left for future work).

5. Potential for Misuse
While designed for conservation, wildlife detection technology could theoretically assist poaching if misappropriated. Mitigations include: open-source release restricted to verified conservation organizations, no real-time alert APIs for external systems, and GPS coordinate encryption requiring institutional keys.

VIII. CONCLUSION

We set out to build a wildlife monitoring system that could do more than count animals one that could understand what they are doing, run fast enough for real-time use, and work in the field without cloud connectivity. The result integrates YOLO26l detection (mAP@0.5=0.78), ByteTrack tracking (MOTA=0.71), and Qwen3-VL-2B behavior classifi-cation (76% zero-shot accuracy) into a unified pipeline that processes video at 45 FPS on an RTX 3050.

What we are most proud of is not any single number, but how the pieces fit together: a multi-camera architecture that never stutters, round-robin GPU scheduling that treats all feeds fairly, VLM integration that brings genuine semantic understanding to behavior analysis, and an edge-native design that keeps sensitive data where it belongs on site. We validated each component through ablation studies and confirmed our improvements are statistically significant.

This is a foundation, not a finished product. But we believe it demonstrates that practical, sophisticated AI for conservation is within reach not in some well-funded research lab, but in the hands of field teams working to protect wildlife today.

Future Work
We have no shortage of ideas for where to take this next. Near-term, we want to use LoRA fine-tuning to improve performance on rare behaviors that the base VLM struggles with. We are also experimenting with feeding short video clips (rather than single frames) to the VLM for better temporal context. Active learning could help us prioritize which new samples to label for maximum impact.

Medium-term, we are eyeing NVIDIA Jetson devices for even more portable deployment this will require aggressive 4-bit quantization. We are curious whether fusing audio (bird calls, elephant rumbles) with video could improve classifica-tion. And we need to tackle domain transfer so a model trained in one reserve works well in another.

Long-term, the dream is ecosystem-scale monitoring: federated learning across dozens of reserves, predictive models that anticipate poaching hotspots, and autonomous response systems that can alert rangers or even deploy drones. There is a long road ahead, but the foundation is solid.

Acknowledgment
We thank the AI&DS Department at Terna Engineering College for resources and guidance, and the open-source communities behind PyTorch, Ultralytics, and Hugging Face Transformers.

Conflict of Interest
The authors declare no competing interests.

Data Availability
Datasets include public camera-trap imagery and custom field data. Custom data available upon reasonable request, subject to conservation considerations.

Reproducibility Statement
Key settings: YOLO26l (45.8M params), 100 epochs, batch 16, AdamW LR 0.01; ByteTrack (θhigh=0.5, θlow=0.1, αmax=30); Qwen3-VL-2B 4-bit quantized; RTX 3050 GPU; multi-camera at JPEG 70%, 800×600, 20 FPS, inference every 3rd frame. Code and models available upon publication.

REFERENCES

1. M. Choinski, M. Rogowski, P. Tynecki, D. Kuijper, M. Churski and J. W. Bubnicki, "A first step towards automated species recognition from camera trap images of mammals using AI in a European temperate forest," arXiv preprint arXiv:2107.07830, 2021.
2. D. Tuia, B. Kellenberger, S. Beery, et al., "Perspectives in machine learning for wildlife conservation," Nature Communications, vol. 13, no. 792, 2022.
3. Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu and X. Wang, "ByteTrack: Multi-object tracking by associating every detection box," Proc. European Conference on Computer Vision (ECCV), pp. 1–21, 2022.
4. G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023. [Online] Available: https://github.com/ultralytics/ultralytics
5. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Op-timal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

6. Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," Proc. AAAI Conference on Artificial Intelligence, vol. 34, pp. 12993–13000, 2020.
7. N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," Proc. IEEE International Conference on Image Processing (ICIP), pp. 3645–3649, 2017.
8. K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," EURASIP Journal on Image and Video Processing, vol. 2008, pp. 1–10, 2008.
9. A. Radford, J. W. Kim, C. Hallacy, et al., "Learning transferable visual models from natural language supervision," Proc. International Conference on Machine Learning (ICML), pp. 8748–8763, 2021.
10. H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," Advances in Neural Information Processing Systems, vol. 36, 2023.
11. J. Bai, S. Bai, S. Yang, et al., "Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond," arXiv preprint arXiv:2308.12966, 2023.
12. W. Liu, G. Ren, R. Yu, S. Guo, J. Zhu and L. Zhang, "Image-adaptive YOLO for object detection in adverse weather conditions," Proc. AAAI Conference on Artificial Intelligence, vol. 36, pp. 1792–1800, 2022.
13. B. Kellenberger, D. Marcos and D. Tuia, "Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning," Remote Sensing of Environment, vol. 216, pp. 139–153, 2018.
14. S. Beery, D. Morris, and S. Yang, "Efficient pipeline for camera trap image review," arXiv preprint arXiv:1907.06772, 2019.
15. S. Schneider, G. Taylor and S. Kremer, "Deep learning object detection methods for ecological camera trap data," Proc. Conference on Computer and Robot Vision (CRV), pp. 321–328, 2018.
16. M. Willi, R. T. Pitman, A. W. Cardoso, et al., "Identifying animal species in camera trap images using deep learning and citizen science," Methods in Ecology and Evolution, vol. 10, no. 1, pp. 80–91, 2019.
17. A. Swanson, M. Kosmala, C. Lintott, R. Simpson, A. Smith, and C. Packer, "Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna," Scientific Data, vol. 2, no. 150026, 2015.
18. S. Beery, G. Van Horn, and P. Perona, "Recognition in terra incognita," Proc. European Conference on Computer Vision (ECCV), pp. 456–473, 2018.
19. L. Falcon-Vado, R. Martin-Brualla, and J. A. Guerrero-Ibanez, "End-to-end species detection using video sequences," Ecological Informatics, vol. 65, 101413, 2021.
20. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," Proc. International Conference on Learning Representations (ICLR), 2022.
21. T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "LLM.int8(): 8-bit matrix multiplication for transformers at scale," Advances in Neural Information Processing Systems, vol. 35, pp. 30318–30332, 2022.
22. A. Dosovitskiy, L. Beyer, A. Kolesnikov, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," Proc. International Conference on Learning Representations (ICLR), 2021.
23. T.-Y. Lin, P. Dolla´r, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2117–2125, 2017.
24. T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dolla´r, "Focal loss for dense object detection," Proc. IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988, 2017.
25. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 658–666, 2019.
26. Qwen Team, "Qwen2-VL: Enhancing vision-language model's percep-tion of the world at any resolution," arXiv preprint arXiv:2409.12191, 2024.
27. Ultralytics, "YOLO26: End-to-End Object Detection," arXiv preprint arXiv:2509.25164, 2025.
28. A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," Proc. IEEE International Conference on Image Processing (ICIP), pp. 3464–3468, 2016.
29. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," Advances in Neural Information Processing Systems (NeurIPS), vol. 28, pp. 91–99, 2015.