

Real-Time Wildlife Poaching Detection System Using YOLOv12 and Lightweight CNN Architecture

Authors: Harsha Naik, Laxmi Madiwalar, Amulya Chougale,
Basalingappa Patil, Sharada Shiragudikar

Table of Contents

Real-Time Wildlife Poaching Detection System using YOLOv12 and Lightweight CNN Architecture



- Introduction
- Motivation
- Problem Statement
- Literature Survey
- Objectives
- Research Gap
- Proposed Workflow
- Implementation
- Results / Outcomes Mapping
- Publication

Introduction

Real-Time Wildlife Poaching Detection System using YOLOv12 and Lightweight CNN Architecture

What is Wildlife Poaching?

Poaching:

- Illegal killing of wild animals for profit
- Driven by organized crime networks
- Threatens species extinction globally

Impact:

- Species face critical population decline
- Ecosystem disruption and food chain collapse
- \$23 billion annual loss from illegal trade
- Conservation crisis in remote reserves

Why it Matters:

- Automated detection enables rapid ranger response
- Bridges critical time gap for intervention
- Technology-driven conservation essential for survival



Introduction

Traditional Monitoring Challenges

Current Systems:

- Manual patrol covers small areas only
- Fixed cameras lack real-time alerts
- 2-4 hour response delay allows poachers to escape
- Cannot monitor 24/7 continuously

Why They Fail:

- Cannot scale across vast terrain
- Human fatigue reduces night detection
- Lack instant threat identification
- Limited to single reserve coverage

AI Solution:

- Continuous 24/7 monitoring without fatigue
- Real-time detection with instant alerts
- Scalable across multiple reserves
- Improved threat identification accuracy



Introduction

AI Revolution in Wildlife Conservation

Computer Vision Evolution:

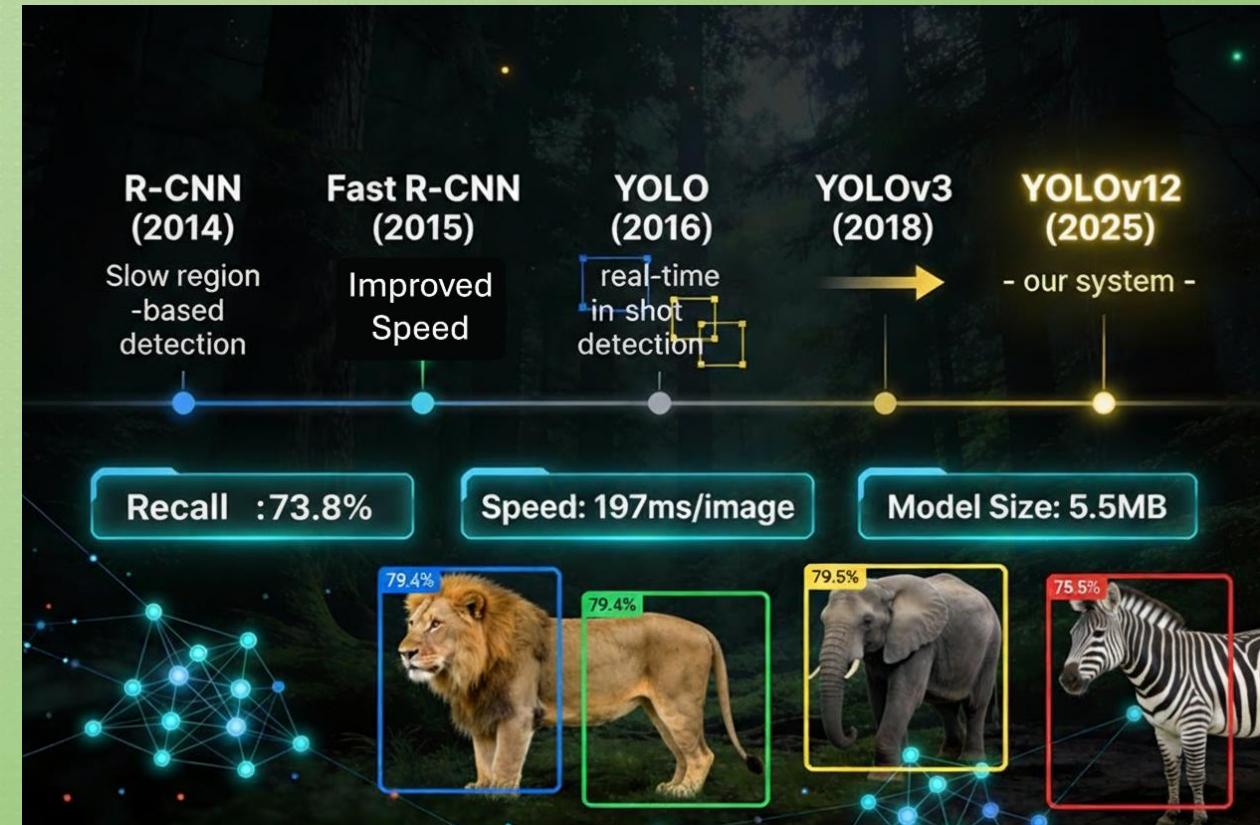
- Deep learning: end-to-end features.
- R-CNN / Faster R-CNN: accurate, slow.
- YOLO: single-stage, real-time.
- YOLOv12n: improved backbone + losses.

Why Deep Learning Works:

- Detects animals and humans together.
- Handles clutter, scale, illumination.
- Robust to day/night and occlusion.
- Enables continuous automated monitoring.

YOLO Advantage:

- Single-pass detection pipeline.
- Good accuracy with low compute.
- ~5.5 MB YOLOv12n model for edge.



Introduction

Real-Time Wildlife Poaching Detection System using YOLOv12 and Lightweight CNN Architecture

Real-Time Wildlife Detection System

Goals:

- Automated detection of wildlife and humans.
- Lightweight model suitable for edge devices.
- Support timely ranger intervention.

Detection Scope:

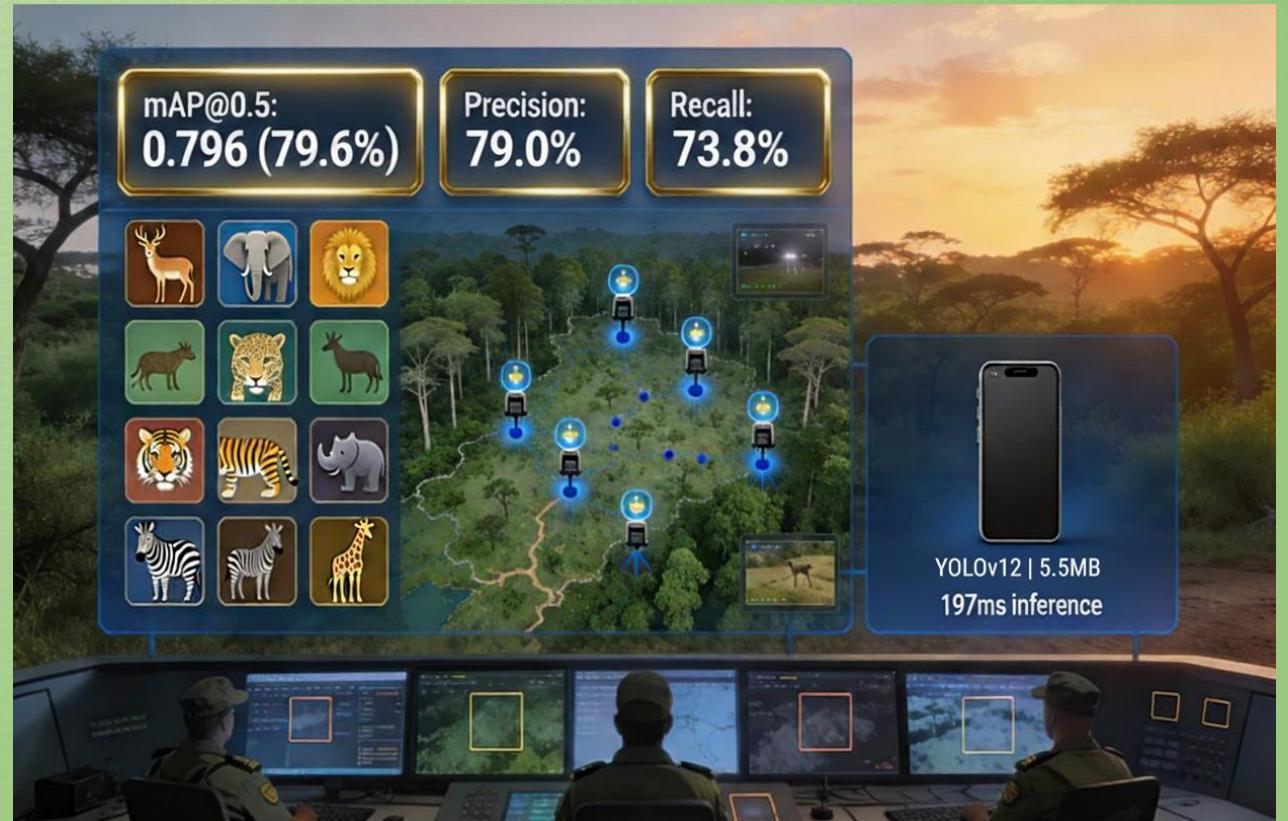
- 38 object classes.
- 5,204 labelled images in forest imagery.

Performance (Test Split – 517 images):

- mAP@0.5: 0.796.
- mAP@0.5:0.95: 0.615.
- Precision: 0.790.
- Recall: 0.738.
- Model size: \approx 5.5 MB (YOLOv12n checkpoint).

Impact:

- Scalable tool for multi-species wildlife monitoring.
- Helps detect both animals and potential poachers.
- Bridges AI models with practical field deployments



1. Need for Faster, Automated Wildlife Protection:

- Poaching crisis needs continuous monitoring.
- Manual patrols cannot cover large reserves.
- Require timely detection of animals and humans.

2. Limitations of Conventional Conservation:

- Patrols and fixed cameras give delayed response.
- No automatic analysis of camera streams.
- High risk of missing poaching activity.

3. AI as a Catalyst for Wildlife Security:

- Real-time multi-species and human detection.
- Compact YOLOv12n model for edge devices.
- Supports quicker ranger decisions in the field.

Problem Statement

Real-Time Wildlife Poaching Detection System using YOLOv12 and Lightweight CNN Architecture



- Develop a real-time system to detect wildlife species and human presence in forest environments. The system should ensure high detection accuracy and fast response to effectively prevent poaching.

Objectives

1. Detect wildlife species and human presence in forest environments.
2. Design a real-time detection system using YOLOv12.
3. Develop a lightweight architecture suitable for edge deployment.
4. Evaluate detection accuracy and real-time performance.

Limited Real-Time Multi-Species Detection

Existing wildlife monitoring systems lack accurate, simultaneous detection of multiple species and human threats in real forest conditions with varying lighting and occlusion.

Inadequate Edge Deployment Solutions

Most detection models are too large for deployment on resource-constrained edge devices in remote reserves without electrical infrastructure.

Poor Night/Low-Light Performance

Conventional models struggle with infrared and night-vision camera feeds common in nocturnal poaching scenarios.

Lack of Explainable Detection Insights

Current systems rarely provide ranger-friendly insights about detected threats, confidence levels, or threat patterns for better decision-making.

Proposed Workflow

Real-Time Wildlife Poaching Detection System using YOLOv12 and Lightweight CNN Architecture

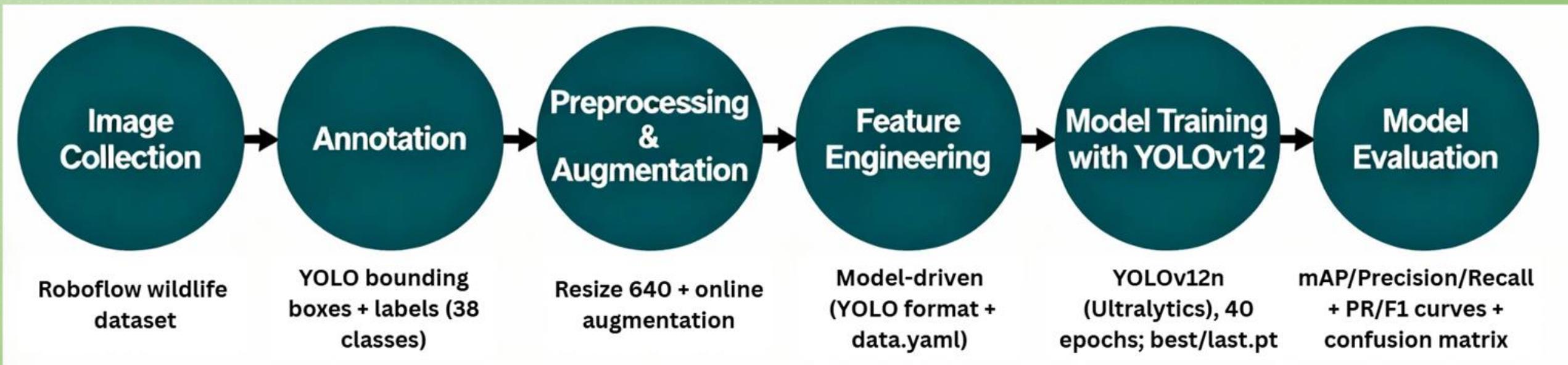


Figure 1: The workflow of the proposed model

Implementation

● Image Collection

Dataset & Sourcing:

- Roboflow “Animal & Human Detection” wildlife dataset (v6); 37 animal/vehicle classes + 1 person class.
- 5,204 labelled images; forest and camera-trap style scenes in JPG/PNG format.
- Strong class imbalance: many common classes, few samples for rare ones.

Data Split & Normalization:

- Predefined Roboflow split: 3,650 train / 1,037 validation / 517 test.
- Standard YOLO layout: images/{train,val,test} and labels/{train,val,test}.
- data.yaml specifies image path, splits, class names (0–37) and nc = 38.
- Image–label consistency check; any missing/extralabels ignored during training.

Implementation

● Annotation

Label Format & Standards:

- YOLO TXT format: <class_id><x_center><y_center><width><height> (normalized 0–1).
- Class indices follow the Roboflow data.yaml mapping exactly (0–37).
- Center–width–height boxes support scale-invariant training and augmentation.

Quality Control:

- Multi-object scenes: one bounding box per visible animal/person.
- Orphan label files without images are ignored during preprocessing.
- Final cleaned set: 5,204 images with valid YOLO labels used for training/validation/testing.

Implementation

● Preprocessing & Augmentation

Input Processing:

- Images resized to 640×640 with letterbox padding (aspect ratio preserved).
- Ultralytics YOLO default normalization and color space handling applied.

Augmentation (augment=True):

- Built-in YOLO transforms: horizontal flip, HSV jitter, translation and random erasing.
- Additional light Albumentations: Blur, MedianBlur, ToGray and CLAHE with low probabilities.

Implementation

● Feature Engineering

Anchor & Scale Management:

- Anchor-free, decoupled head design of YOLOv12n used without custom anchors.
- Multi-scale feature maps detect small, medium and large objects in a single pass.
- Letterbox resizing keeps aspect ratio, so bounding box proportions remain stable.

Class Balance & Hard Examples:

- Heavy class imbalance handled implicitly by focal-style classification loss.
- Rare classes get larger gradients because of higher loss on misclassifications.
- Hard examples (occluded, distant animals) contribute more to loss during training.

Implementation

- **Model Training with YOLOv12 + LiteCNN**

Architecture:

- YOLOv12n backbone—head as provided by Ultralytics; $\sim 2.5M$ parameters, ≈ 5.5 MB checkpoint, pre-trained on COCO.
- Single lightweight detection head suitable for edge deployment on resource-constrained devices.

Training Configuration:

- Epochs: 40 | Batch size: 16 | Base learning rate: 0.01 with cosine annealing schedule.
- Optimizer: AdamW (auto-selected), effective lr $\approx 2.38 \times 10^{-4}$, momentum 0.9, weight decay 0.0005.
- Loss: GIoU box loss + objectness (BCE) + classification loss with focal behaviour for hard examples.

Implementation

● Model Evaluation

Evaluation Protocol:

- Test set: 517 images from Roboflow split (held-out, unseen during training).
- Inference thresholds: confidence = 0.25, NMS IoU = 0.7 (Ultralytics defaults).
- Evaluation via `model.val(data="wildlife_detection.yaml")` after training.

Metrics (COCO-style):

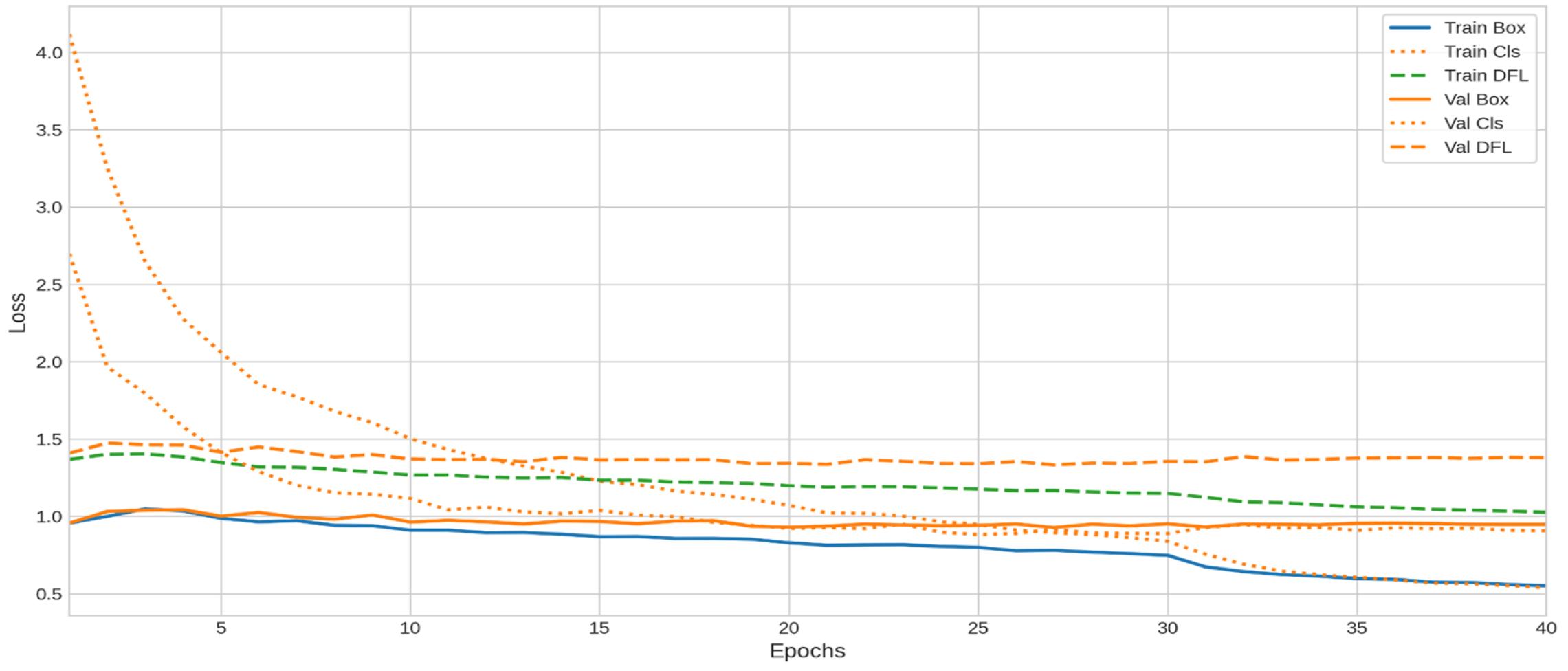
- mAP@0.5 (primary) and mAP@0.5:0.95 (stricter).
- Precision (TP/TP+FP) and Recall (TP/TP+FN) for the test split.
- Per-class AP, Precision, Recall reported from YOLO summary tables.

Results vs. Roboflow Baseline:

- mAP@0.5: 0.796 | mAP@0.5:0.95: 0.615.
- Precision: 0.790 | Recall: 0.738 on the test set.
- Example classes with strong AP: rhino, giraffe, elephant; weaker AP for rare classes due to few samples.
- Inference time: \approx 197 ms per 640×640 image on T4 GPU (real-time for edge use after optimization).

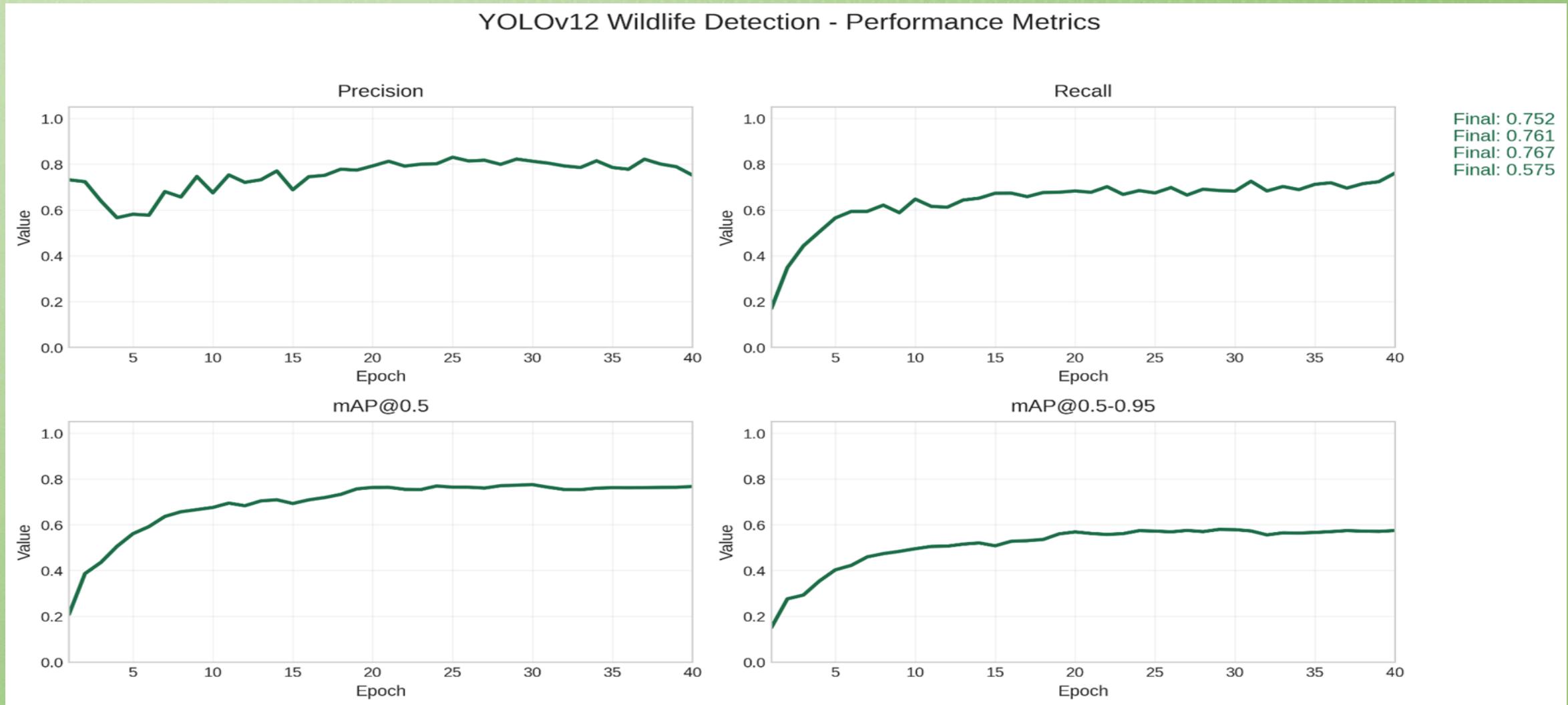
Results

YOLOv12 Wildlife - Train Results



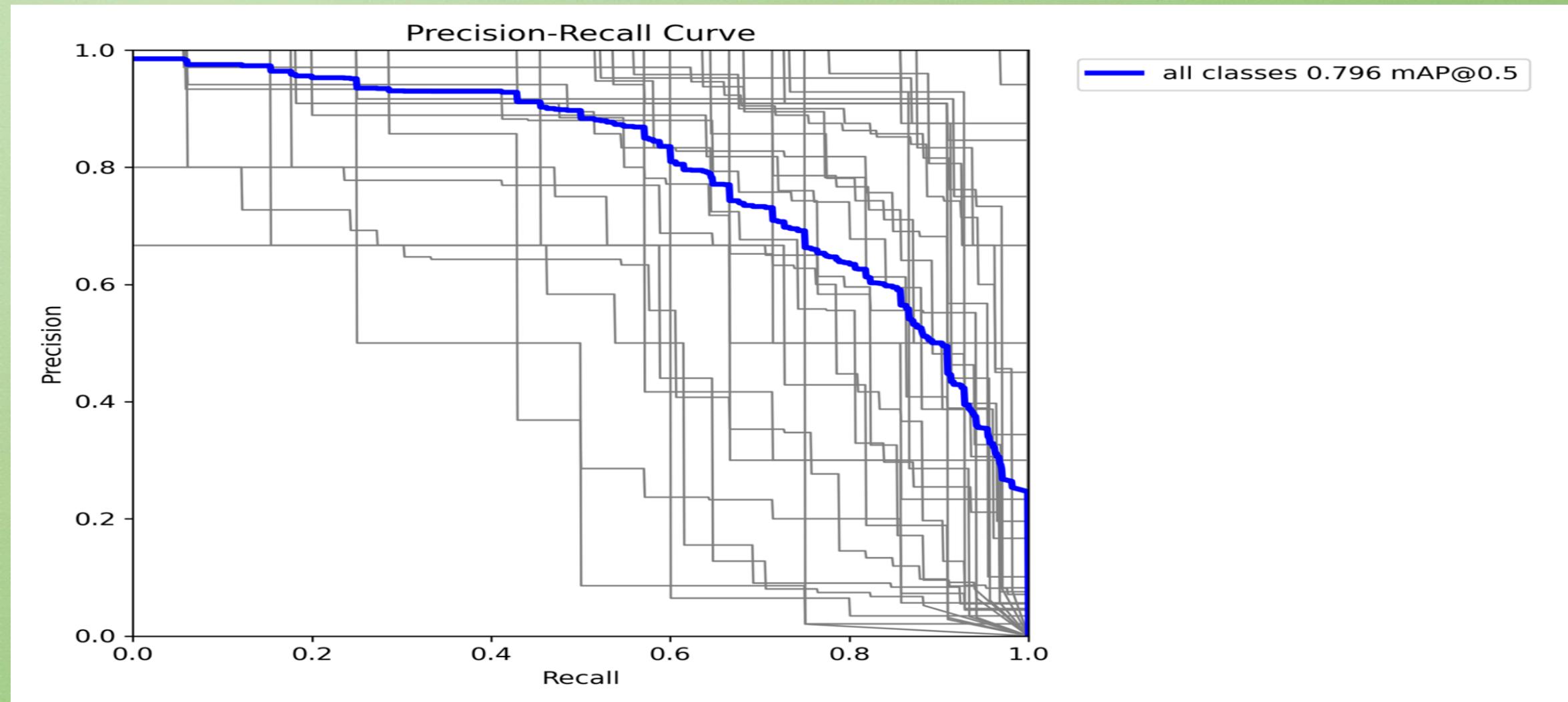
Training and validation losses (box, classification, and DFL) steadily decrease across epochs with a small train-validation gap, indicating stable convergence and good generalization without overfitting.

Results



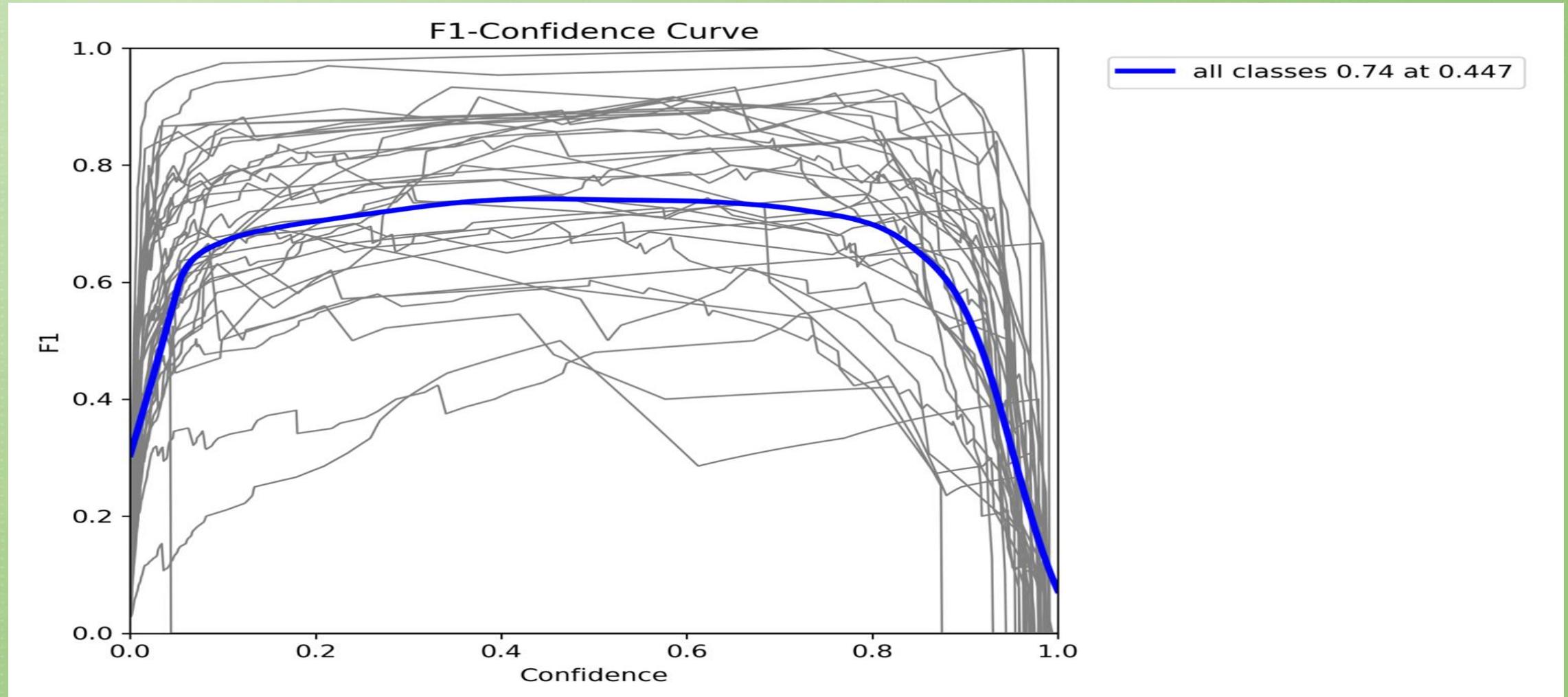
Precision, recall, and mAP values steadily improve during training and stabilize after ~25 epochs, confirming effective convergence and optimal training duration.

Results



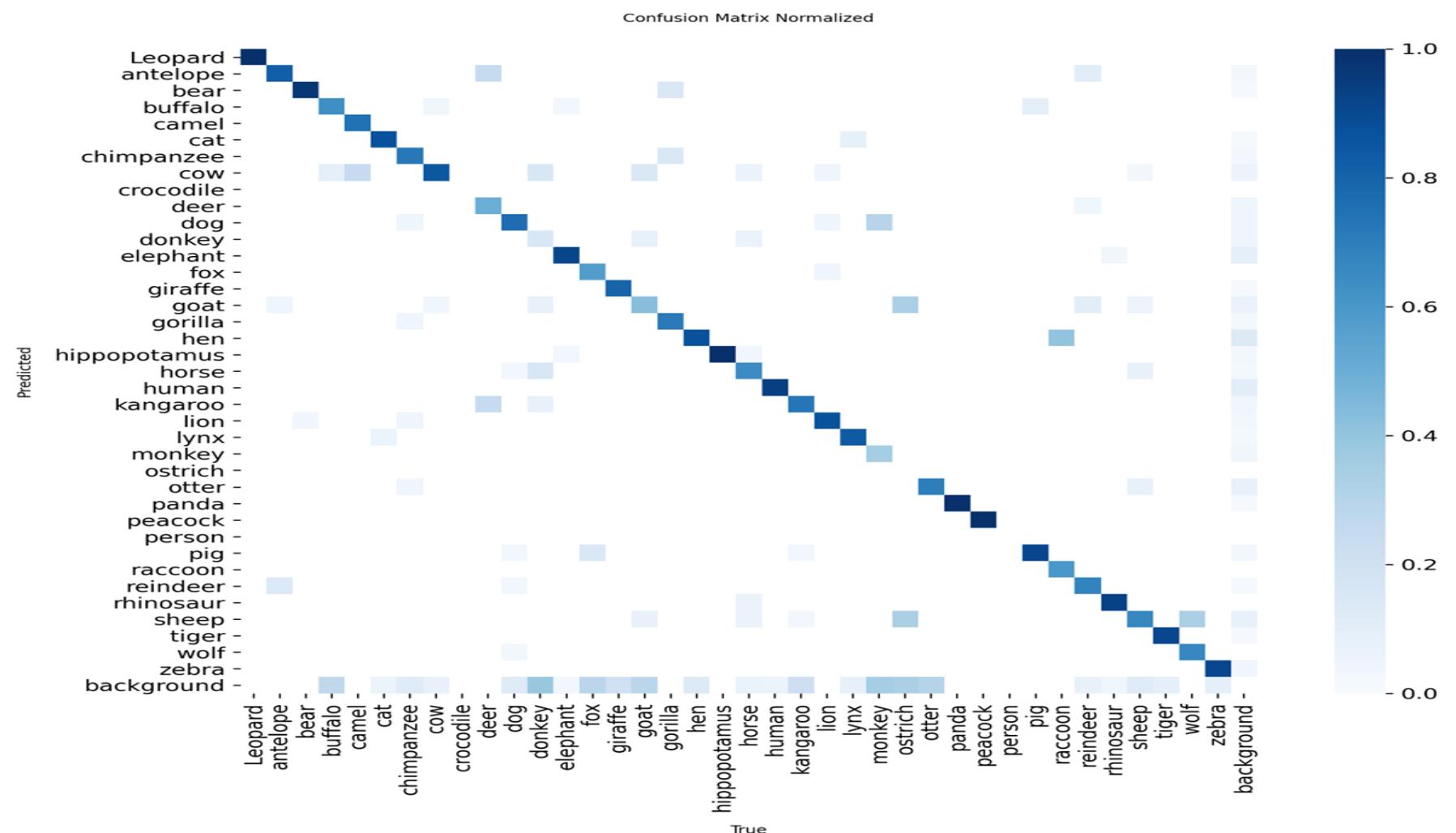
The Precision–Recall curve achieves an mAP@0.5 of 0.794, demonstrating strong detection performance across confidence thresholds with a stable precision–recall trade-off.

Results



F1-score peaks at 0.74 around a confidence threshold of 0.447, marking the optimal trade-off between precision and recall across all classes

Results



The confusion matrix shows strong diagonal dominance with high true positive rates across most classes and limited inter-class confusion, primarily among visually similar species such as lion and leopard.

Robust Multi-Class Wildlife Detection

- YOLOv12 achieved reliable detection of multiple wildlife species and human presence across diverse forest scenarios.
- Detection accuracy remained consistent under varying environmental and lighting conditions.

Stable Training and Model Convergence

- Training and validation losses decreased steadily and converged smoothly during optimization.
- Performance metrics stabilized after optimal epochs without signs of overfitting.

Edge-Optimized and Real-Time Deployment

- Model compression techniques reduced computational complexity while preserving detection accuracy.
- The optimized architecture enabled real-time inference on resource-constrained edge platforms.

Class-Level Performance and Error Analysis

- Confusion matrix analysis showed limited inter-class misclassification among detected species.
- Threshold and F1-score analysis supported reliable operating point selection for deployment.

Conclusion

- The work presented a real-time wildlife poaching detection system based on a YOLOv12 architecture integrated with a lightweight CNN for efficient multi-class detection.
- The proposed model effectively captured multi-scale visual features, enabling accurate detection of wildlife species and human presence in complex forest environments.
- Data-centric optimization and class-aware training strategies improved robustness under occlusion, class imbalance, and challenging lighting conditions.
- Model compression through pruning and quantization enabled edge-ready deployment, supporting practical real-time conservation monitoring applications.

Conference Publication

Real-Time Wildlife Poaching Detection System using YOLOv12 and Lightweight CNN Architecture



Paper Title: Real-Time Wildlife Poaching Detection System Using YOLOv12 and Lightweight CNN Architecture

Conference name: 8th International Conference on Communication and Computational Technologies (ICCCT 2026)

Indexing: Scopus

Location: National Forensic Sciences University Goa Campus, India

Date: 13-14 February, 2026