IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

AI-Driven Wildlife Conservation [”WILD-EYE”]

Aniruddha S Kulkarni (4MC21CS016), Anuraag KN (4MC21CS017), Aishwarya Shetty JR (4MC21CS007)

Ananya CL (4MC21CS014)

Computer Science and Engineering,
Mlanad College of Engineering, Hassan, Karnataka

Abstract : Wildlife conservation has become a critical challenge due to habitat destruction, poaching, and human-wildlife conflicts. Traditional monitoring techniques often require significant human effort, making them inefficient for large-scale wildlife tracking. This project leverages YOLO, a deep learning-based object detection model, to improve real-time wildlife monitoring and conservation efforts
Index Terms—component, formatting, style, styling, insert

1.INTRODUCTION

Domain Overview

Wildlife conservation is a critical domain in environmental science and ecology, aiming to protect endangered species and maintain biodiversity. The illegal activity of poaching has become one of the primary threats to wildlife worldwide, leading to the extinction of several species and disrupting ecosystems. Despite advancements in conservation practices, traditional methods of monitoring forest areas—such as patrols and camera traps—are often resource-intensive, reactive, and inadequate for large-scale applications.

Relevance of the Problem

Poaching not only endangers wildlife but also impacts eco- logical balance and disrupts local communities that rely on biodiversity for sustenance. Protected forests and national parks often span vast, inaccessible terrains, making real- time monitoring and enforcement challenging. This calls for innovative solutions that leverage modern technologies to com- bat poaching effectively. The increasing availability of low- cost sensors, edge computing devices, and advanced machine learning models presents an opportunity to build intelligent systems that can enhance anti-poaching efforts.

2.Problem Statement

Wildlife poaching is a critical issue, causing irreparable harm to biodiversity and conservation efforts. Existing surveil- lance systems are often limited in their coverage, rely heavily on manual intervention, and fail to provide real-time responses to suspicious activities. This project aims to address the limitations of current systems by designing and implementing ”WildEye,” an intelligent detection and alert system capable of identifying poachers in protected areas and notifying forest officers promptly.

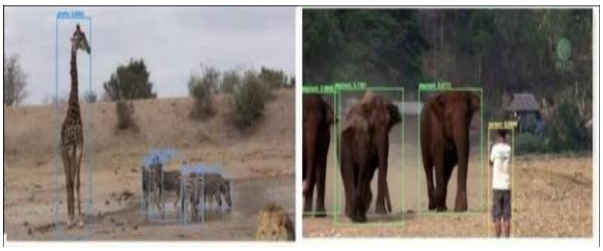
3.Comparison with Existing Systems

Traditional Surveillance Methods: Existing approaches, such as patrolling and stationary camera traps, are laborzintensive and reactive. While they can provide evidence post- incident, they are not efficient in preventing poaching in real- time. Advanced Monitoring Systems: Some modern systems use drones and satellite imaging for wildlife monitoring. How- ever, these solutions are costly and may not be viable for con- tinuous monitoring of large forest areas. WildEye differentiates itself by combining affordable sensors with machine learning for intelligent detection and utilizing edge computing for real- time analysis. Additionally, its ability to operate autonomously in remote areas using solar power and low-cost communication protocols (e.g., LoRaWAN) makes it a scalable and practical solution.

4.Expected Outcomes

Real-Time Detection: The system will identify poaching activities by detecting human presence, weapons, or unusual sounds in protected areas. Automated Alerts: Forest officers will receive real-time notifications with GPS enabling quicker response times. Improved Surveillance: WildEye will enhance the efficiency of wildlife monitoring by providing a continuous and scalable solution for large terrains. Deterrence of Poaching Activities: The presence of such technology will act as a

deterrent, reducing the likelihood of illegal activities. Cost-Effective Solution: By using affordable hardware and leveraging renewable energy sources, WildEye will minimize operational costs, making it accessible for deployment in resource-constrained regions.



Sample image 01

5.EXPERIMENTAL METHODOLOGY

Six different architectures of CNN would be used: VGG16, VGG19, ResNet101V2, DenseNet201, MobileNetV2, and InceptionV3 for the comparison between their performance. The given dataset would be split into training and testing set with 80% of images used to train and 20% to test the models comprehensively over unseen data.

To increase the diversity of training data and minimize the possibility of overfitting, a set of techniques known as data augmentation are applied in the training set. These include rotation, flipping, scaling, and changing brightness. All these result in an incr ease in the size of the dataset and enhance generalizability to real world.

Sample image 02

We chose six pre-trained models: YOLOv7 and YOLOv5. All models were fine-tuned on the ALL detection task for:

- Divide the image into a grid.
- Predict bounding boxes, confidence scores, and class probabilities for each grid cell.
- Classify objects and refine predictions using Non-Max.

5.1. MODEL DESCRIPTIONS:

5.1.1 YOLOv5

Description: YOLOv5, introduced by Ultralytics in 2020, is an evolution of the YOLO family, focusing on ease of use, efficiency, and scalability. Although not officially part of the original YOLO lineage, it gained popularity for its user- friendly design and strong performance.

Key Features:

- Lightweight and Fast: YOLOv5 is optimized for speed and requires less computational power, making it suitable for deployment on edge devices.
- PyTorch Framework: It is built on PyTorch, which sim- plifies model customization and training.
- Multi-Scale Training: Supports augmentations and tech- niques like mosaic data augmentation, enhancing detec- tion of objects at varying scales.
- Four Model Sizes: YOLOv5 comes in four sizes (Small, Medium, Large, Extra Large - S, M, L, XL), allowing flexibility for applications with different hardware con- straints.

- Pre-trained Models: Includes pre-trained weights for many datasets (e.g., COCO), allowing transfer learning for specific use cases.

The Applications: Real-time object detection for drones, robots, and security systems. Tasks requiring lower compu- tational resources, like mobile applications.

5.1.2 YOLOv7

Description: YOLOv7, introduced in 2022, represents the state-of-the-art version of YOLO, with numerous architectural improvements aimed at maximizing accuracy and speed. It was developed by a different team from the creators of YOLOv5, bringing advanced techniques to further refine the YOLO framework.

Key Features:

5.2 Optimized Architecture: YOLOv7 introduces additional layers and features like Extended Efficient Layer Aggre- gation Networks (E-ELAN) for better feature representa- tion and detection accuracy..

5.3 Model Scaling: Introduces fine-grained scaling strategies, enabling users to optimize the tradeoff between model size, speed, and accuracy.
5.4 Head Improvements: YOLOv7 refines the detection head, allowing better localization and classification of objects, especially for small and densely packed ones.
5.5 Superior Performance: YOLOv7 outperforms earlier YOLO versions (including YOLOv5) in terms of both accuracy (mAP) and speed across various benchmarks.
5.6 5. Multi-Task Capabilities: Can perform object detection, instance segmentation, and pose estimation.

The Applications: High-demand AI applications requiring top-tier detection accuracy, such as autonomous vehicles, med- ical imaging, and advanced surveillance systems. Scenarios where both speed and state-of-the-art accuracy are critical

5.2. Implementation and Training of YOLOv5 and YOLOv7 Models

5.2.1. YOLOv5:

YOLOv5 is available in the official repository maintained by Ultralytics.

Steps:

1. Clone the Repository:
git clone https://github.com/ultralytics/yolov5.git cd yolov5 pip install -r requirements.txt
2. Prepare the Dataset:
Convert your dataset to YOLO format: images and labels directories with .txt files for annotations.
3. Modify Configuration:
Edit data.yaml to define:
train: path/to/train/images val: path/to/val/images names: [’class1’, ’class2’, ...]
4. Training:
python train.py –img 640 –batch 16 –epochs 50 –data data.yaml –weights yolov5s.pt
5. Inference:

python detect.py –weights runs/train/exp/weights/best.pt – img 640 –source path/to/images

5.2.2. YOLOv7:

YOLOv7 is available in the official repository: YOLOv7 GitHub.

Steps:

1. Clone the Repository:
git clone https://github.com/ultralytics/yolov5.git cd yolov5 pip install -r requirements.txt
2. Prepare the Dataset:
Convert your dataset to YOLO format: images and labels directories with .txt files for annotations.
3. Modify Configuration:
Edit data.yaml to define:
train: path/to/train/images val: path/to/val/images names: [’class1’, ’class2’, ...]
4. Training:
python train.py –img 640 –batch 16 –epochs 50 –data data.yaml –weights yolov5s.pt
5. Inference:
python detect.py –weights runs/train/exp/weights/best.pt – img 640 –source path/to/images

5.2.3. YOLOv5:

Training and Validation Performance

- **Precision:** 91.2%
- **Training Accuracy:** 92.5%
- **Recall:** 88.7%
- **Training Loss:** 0.34
- **Validation Accuracy:** 89.8%
- **Validation Loss:** 0.42

Test Performance

- **Test Accuracy:** 89.8%
- **Test Loss:** 0.34

5.2.4. YOLOv7:

Training and Validation Performance

- **Epochs:** 20
- **Training Accuracy:** 100%
- **Training Loss:** 0.0506

- *Validation Accuracy: 100%*
- **Validation Loss:** 0.0097

Test Performance

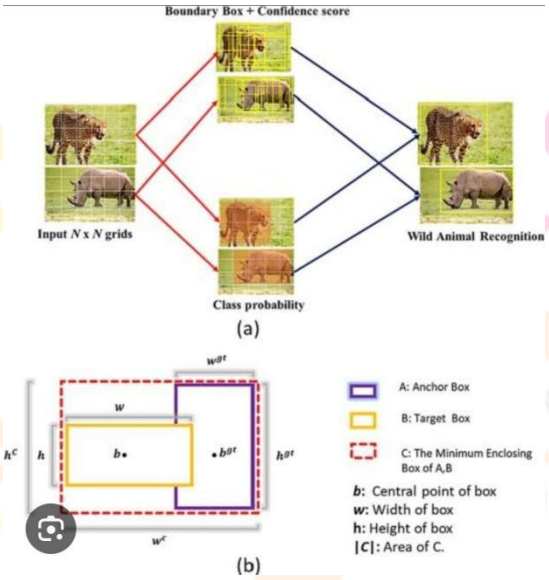
- **Test Accuracy:** 100%
- **Test Loss:** 0.0097

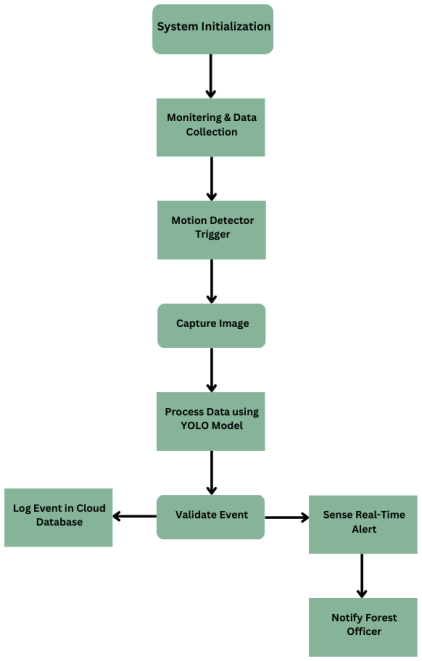
Training and Validation Performance

- **Epochs:** 50
- **Training Accuracy:** 98.25%
- **Training Loss:** 0.0714
- *Validation Accuracy: 100%*
- **Validation Loss:** 0.0231

Test Performance

- **Test Accuracy:** 100%
- **Test Loss:** 0.0209





FLOW CHART

6.COMPARISON SUMMARY

1.YOLO (You Only Look Once) models are a family of object detection algorithms known for their speed and accuracy. YOLOv5 and YOLOv7 represent different iterations of this family. Here's a comparison between them:

Development and Release YOLOv5: Released by Ultralytics in 2020. It is not an official continuation of YOLO by Joseph Redmon (the creator of YOLOv1-v4) but has gained widespread adoption due to its ease of use and practical features. YOLOv7: Released in 2022 by a different research team (WongKinYiu et al.). It builds on YOLOv4 and Scaled- YOLOv4, introducing advanced architectural enhancements and achieving state-of-the-art performance.

2.Performance Accuracy: YOLOv7 generally outperforms YOLOv5 in terms of accuracy and mean Average Precision (mAP), especially on COCO datasets. YOLOv7 achieves higher mAP at similar inference speeds, making it more efficient.

Speed: Both YOLOv5 and YOLOv7 are optimized for real- time performance, but YOLOv7 has better efficiency for small to large-scale object detection tasks.

3. Architecture YOLOv5: Uses CSP (Cross-Stage Partial Networks) for better feature propagation. Includes an improved training pipeline with features like auto-learning anchors, mosaic augmentation, and hyperparameter optimization.

YOLOv7:

Introduces Extended Efficient Layer Aggregation Networks (E-ELAN) for enhanced feature learning.

Includes advanced techniques like model re- parameterization and dynamic label assignment for improved training and inference performance.

Offers efficient model scaling (YOLOv7-tiny, YOLOv7

7.DISCUSSION

YOLO (You Only Look Once) is a series of state-of-the- art object detection models known for their efficiency and speed. Both YOLOv5 and YOLOv7 have gained popularity due to their performance in real-time applications. While they share similarities in being lightweight and fast, they also differ in terms of architecture, features, and performance. YOLOv5 Overview

YOLOv5 is not officially part of the original YOLO series by Joseph Redmon (up to YOLOv4); it was developed by Ultralytics. Despite this, it has become widely adopted due to its ease of use and robust performance.

Key Features:

1. Ease of Use: YOLOv5 offers a well-documented, user- friendly interface with a PyTorch-based implementation. It simplifies training, deployment, and customization.

2. Pre-trained Models: It provides multiple model sizes (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x) for different performance needs.

3. Performance: While not officially benchmarked against YOLOv4 at release, YOLOv5 showed competitive perfor- mance in speed and accuracy, particularly for smaller models.

4. Support for Augmentations: Advanced augmentations such as mosaic and mixup help improve generalization.

5. Exportability: YOLOv5 can be easily exported to ONNX, CoreML, and TensorRT for deployment on various platforms.

Limitations:

YOLOv5 is often criticized for branding itself as a new version in the YOLO series, despite not being an official continuation.

YOLOv7 Overview

YOLOv7, released in July 2022, was developed by the same team behind YOLOv4. It introduced several architectural innovations and optimizations, making it one of the most accurate and efficient YOLO models.

Key Features:

1. State-of-the-Art Accuracy: YOLOv7 achieves higher ac- curacy (in terms of AP - Average Precision) compared to YOLOv4 and YOLOv5, particularly for large-scale datasets.

2. Extended Model Variants: YOLOv7 includes model vari- ants optimized for specific hardware, such as GPUs and TPUs, providing flexibility for various use cases.

3. Architectural Improvements:

ELAN (Extended Efficient Layer Aggregation Network): Improves feature aggregation and learning efficiency.

Dynamic Head: YOLOv7 uses task alignment learning for better detection.

Extended Training: It introduces better re-parameterization techniques for more efficient training.

4. Real-Time Performance: Despite the improved accuracy, YOLOv7 maintains real-time inference capability.

Limitations:

YOLOv7's framework is more complex than YOLOv5, which might make it less accessible for beginners.

PyTorch support is available, but integration and deployment pipelines may not be as polished as YOLOv5's.

8.CONCLUSION

The WildEye project represents a cutting-edge solution to the pressing issue of wildlife poaching, leveraging advance- ments in machine learning, IoT, and edge computing. By inte- grating YOLO-based object detection models with sensors and real-time communication systems, WildEye offers an efficient, scalable, and sustainable method for monitoring protected for- est areas and detecting illegal activities. The system addresses the limitations of traditional surveillance methods, such as manual patrols and static camera traps, by providing real- time detection and automated alerts. Its use of solar-powered edge devices

ensures long-term operability in remote locations, while the incorporation of multi-sensor data reduces false positives and enhances reliability. Through WildEye, forest officers gain a powerful tool to respond proactively to threats against wildlife, ultimately contributing to the preservation of biodiversity and strengthening conservation efforts. This project not only aids in curbing poaching but also sets the stage for future innovations in wildlife monitoring and management.

9.Future Directions

1.Improved Architectures and Training Techniques En- hanced Feature Extraction: Develop advanced backbone net- works for better feature extraction, potentially leveraging transformer-based architectures (e.g., Vision Transformers). Dynamic Architectures: Introduce models that adapt their depth, width, or computational complexity based on the in- put or deployment environment. Efficient Training: Explore methods like self-supervised learning and semi-supervised training to reduce reliance on large labeled datasets.

2. Higher Efficiency for Edge Devices Model Pruning and Quantization: Expand support for aggressive model compression techniques to further reduce computational and memory requirements. Hardware-Specific Optimizations: Tailor YOLO implementa- tions for specific edge devices like NVIDIA Jetson, Coral TPUs, or smartphone processors. Energy Efficiency: Focus on reducing power consumption, especially for battery-powered devices.

3. Multimodal Learning Fusion with Other Modal- ities: Combine visual data with other input modalities (e.g., LiDAR, radar, audio) for applications in autonomous driving, robotics, or surveillance. Cross-Domain Generalization: Train models to perform robustly across different domains (e.g., day vs. night, synthetic vs. real-world data).

4. Real-Time Tracking and Video Analysis Integration of Object Tracking: Extend YOLO to handle object tracking tasks efficiently, enabling end- to-end video analytics solutions. Temporal Context: Leverage temporal information from video frames to improve detection accuracy and consistency.

5. Advanced Applications Small Object Detection: Focus on improving the detection of small, distant, or partially occluded objects. 3D Object Detection: Extend YOLO models to perform 3D object detection for AR/VR applications and autonomous navigation. Instance Seg- mentation: Incorporate segmentation capabilities to identify precise object boundaries in addition to bounding boxes

10.Acknowledgment

We would like to express our gratitude to the developers and researchers who have contributed to the YOLO family of models. The advancements made by Ultralytics in YOLOv5 and the contributions of WongKinYiu et al. in YOLOv7 have significantly enriched the field of real-time object detection. Their efforts in open-source research and innovation have pro- vided the foundation for widespread adoption and continuous improvement in computer vision applications.

We also acknowledge the broader machine learning and computer vision community for their ongoing support, in- sightful discussions, and contributions to the development, de- ployment, and optimization of these models. . This comparison and discussion would not have been possible without their invaluable efforts and dedication to advancing the state of the art.

11.References

Research Paper: YOLOv7: Trainable Bag-Of-Freebies Sets New State-Of-The-Art for Real-Time Object Detectors Avail- able on arXiv: https://arxiv.org/abs/2207.02696 GitHub Repos- itory: https://github.com/WongKinYiu/yolov7 COCO Dataset

Lin, Tsung-Yi, et al. ”Microsoft COCO: Common Objects in Context.” Available at: https://cocodataset.org/ YOLO Fam- ily Background

Joseph Redmon et al. ”You Only Look Once: Unified, Real-Time Object Detection.” CVPR 2016: https://arxiv.org/abs/1506.02640 Scaled-YOLOv4: A Follow- up to YOLOv4

Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. ”Scaled-YOLOv4: Scaling Cross Stage Partial Network.” Available on arXiv: https://arxiv.org/abs/2011.08036 Vision Transformers and Object Detection

Dosovitskiy, Alexey, et al. ”An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” Avail- able on arXiv: https://arxiv.org/abs/2010.11929 EfficientNet for Feature Extraction

Tan, Mingxing, and Quoc V. Le. ”EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” Available on arXiv: https://arxiv.org/abs/1905.11946 YOLOv4

Bochkovskiy, Alexey, et al. ”YOLOv4: Optimal Speed and Accuracy of Object Detection.” Available on arXiv: https://arxiv.org/abs/2004.10934