International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 11s, 2025
https://www.theaspd.com/ijes.php

Advancing Environmental Monitoring: YOLO Algorithm For Real- Time Detection Of Greater One-Horned Rhinos

Nelson R Varte[1*], Kaustubh Bhattacharyya[2], Navajit Saikia[3]

[1*]*Department of Computer Application, Assam Engineering College, Jalukbari, Guwahati, India, Email: nelson.varte@gmail.com*
[2]*Department of ECE, Assam Don Bosco University, Azara, Guwahati, India Email: kaustubh.d.electronics@gmail.com*
[3]*Department of ETE, Assam Engineering College, Jalukbari, Guwahati, India Email: navajit.ete@aec.ac.in*

Abstract
Object detection is a key challenge in computer vision, with applications that span security, surveillance, autonomous driving, and wildlife conservation. You Only Look Once (YOLO) has emerged as a state-of-the-art framework for real-time object detection, and its models are typically trained on standard datasets such as Common Objects in Context (COCO) and PASCAL Visual Object Classes (VOC), which often lack the diversity needed for real-world applications. Conservation efforts for endangered species, such as the Greater One-horned Rhino, require specialized solutions due to their ecological importance and vulnerability.
Assam and Northeast India, home to the largest population of the endangered Greater One-horned Rhino, faces severe environmental threats including annual monsoon floods, habitat encroachment, and human-wildlife conflicts. These challenges highlight the urgent need for AI-powered monitoring solutions that can aid in conservation efforts. This work evaluated the performance of various You Only Look Once (YOLO) models, from YOLOv5 to YOLOv9, to detect one-horned rhinos. It also enhances the existing rhino dataset by improving data quality, diversity, and relevance and tunes hyper-parameters for optimal performance. The best-performing model achieved a mean average precision (mAP) of 98.9% and an F1 score of 98%.
Our findings underline the potential of tailored deep-learning models for wildlife monitoring, offering a scalable and effective approach to mitigating human-animal conflicts. By integrating AI into conservation practices, we can enhance real-time tracking, improve habitat protection strategies, and contribute to the long-term survival of endangered species.

Key Words : *Computer Vision, Machine Learning, Deep Learning, Convolutional Neural Network, YOLO Algorithm, Object Detection, Wildlife Monitoring, Human-Animal Conflicts*

1 Introduction

Detecting wild animals is crucial for combating poaching and mitigating human-animal conflicts, both of which have escalated due to urbanization and habitat encroachment. Many species, including the endangered one-horned rhino, face extinction, underscoring the urgency of effective conservation tools. Assam and Northeast India, a critical habitat for this species, frequently experience challenges such as recurrent flooding, habitat loss, and increasing instances of human-wildlife conflict, making real-time monitoring essential for mitigation efforts.

Advances in deep learning, particularly YOLO models, have revolutionized object detection, enabling real-time identification with high accuracy and speed. YOLO is a convolutional neural network (CNN)-based object detector that performs regression and classification in a single pass, making it efficient in detecting multiple objects of varying sizes and shapes. Its variants have introduced features such as anchor boxes, spatial pyramid pooling, and mosaic augmentation to enhance performance. However, these models are typically trained on datasets such as COCO and PASCAL VOC, which lack the quality and diversity needed for specialized tasks such as wildlife monitoring.

Object detection technologies have become an indispensable tool in the conservation and management of endangered species, providing precise and efficient methods for monitoring wildlife populations. These systems leverage advanced computer vision techniques to identify and track individual animals within their natural habitats, facilitating ecological studies with minimal human interference. By enabling automated population assessments, habitat utilization analysis, and behavioral studies, object

detection enhances the scientific understanding of species distribution and dynamics. Furthermore, real- time object detection plays a crucial role in mitigating human-wildlife conflicts, particularly in regions where habitat encroachment and environmental changes disrupt traditional animal migration patterns. The deployment of AI-powered monitoring solutions strengthens anti-poaching efforts by providing conservationists and authorities with actionable intelligence, thus improving the enforcement of wildlife protection laws. Additionally, object detection supports habitat preservation by offering valuable insights into environmental threats, such as deforestation and climate-induced changes, allowing for proactive conservation strategies.

This work explores the detection of the one-horned rhino using YOLO models. The key contributions include the following:

- Evaluation of various YOLO versions on a one-horned rhino dataset.
- Improvement of the dataset by collecting and annotating additional images to enhance its size,
- quality, and diversity.
- Training and comparison of YOLO models using a custom dataset to assess their performance.
- Analysis of the results to derive insights for wildlife detection applications.

Integrating these technologies into conservation practices not only enhances wildlife monitoring but also informs policy decisions and resource allocation for biodiversity conservation. As deep learning models continue to evolve, their application in ecological research and conservation management is expected to expand, further contributing to the protection and sustainability of endangered species and their habitats. The remainder of this paper is structured as follows: Section 2 review related works and theories.

Section 3 details the methodology and presents the results. Section 4 discusses the findings as a conclusion, and finally concludes with future scope.

2 Literature Review

2.1 Related Works

Recent advancements in machine learning have significantly contributed to object detection and tracking in wildlife monitoring and road safety applications. This literature review summarizes key studies, grouping similar objectives and methods to provide a streamlined perspective on the field.

Animal Detection for Road Safety:

The work in [1] & [2] focuses on detecting animals to prevent road accidents. In [2], a system using Histogram of Oriented Gradients (HOG) and a cascade classifier detects cows on highways with an 82.5% accuracy, but its effectiveness is limited to daylight and speeds below 35 km/h. Similarly, [2] employs Faster R-CNN, trained on the PASCAL VOC 2012 dataset, to detect on-road obstacles, including animals. The study emphasizes robustness to varying environmental conditions and suggests dataset customization for Indian road scenarios.

Camera Trap-Based Animal Detection:

Detecting animals in cluttered backgrounds from camera trap images has been a major challenge, addressed in [3], [4]. The authors in [3] propose multilevel graph cuts for object segmentation and introduce deep learning with HOG features for enhanced accuracy, outperforming Faster R-CNN by 4.5%. Meanwhile, [4] integrates dynamic background modeling with deep convolutional neural networks (DCNN), achieving high accuracy while reducing classification time by 14 times.

Deep Learning for Wildlife Monitoring:

Several studies leverage deep learning for large-scale automated wildlife monitoring. In [5], a model trained on the Wildlife Spotter dataset achieves 96% accuracy in recognizing common species like birds and rats. Similarly, [6] trains deep CNNs on the Snapshot Serengeti dataset, identifying 48 species with 93.8% accuracy and reducing manual annotation time by 99.3%. In [7], an ensemble graph-cut segmentation method is used for animal object detection in dynamic scenes, improving detection rates while minimizing false positives.

UAV and Thermal Imaging for Animal Detection:
UAVs and thermal cameras have been employed for improved wildlife detection in [8] & [9]. Study [8] uses thermal imaging and a dynamic thresholding approach to detect animals, extracting features via Discrete Cosine Transform (DCT) and classifying them with a KNN classifier, achieving up to 93.3% balanced accuracy. In [9], UAV-acquired sub-decimeter images enable high-precision animal detection, surpassing Fast R-CNN in reducing false positives while processing over 72 images per second for real- time applications.

Individual Animal Identification and Action Recognition:
Recognizing specific animals or their actions is explored in [10] & [11]. Study [10] employs Faster R-CNN with AlexNet features and SVM classification for identifying patterned species like tigers and zebras. In [11], an I3D-based model is tested for animal action recognition, achieving 36% accuracy, with recommendations for pre-trained models to enhance performance.

Specialized Challenges in Animal Detection:
Challenges like detecting animals behind obstructions or in occluded environments are addressed in [12] & [13]. In [12], M2Det, utilizing a Multi-Level Feature Pyramid Network (MLFPN), improves detection for animals behind cage bars. Study [13] presents a unified model integrating feature extraction, occlusion handling, and classification, reducing the average miss rate by 9% on pedestrian detection benchmarks.

One-Horned Rhino Detection:
Building on these advancements, [14] focuses on detecting the Greater One-Horned Rhino using YOLOv3, introducing architectural modifications to improve performance. The study also presents the first deep learning dataset specific to one-horned rhinos, setting the groundwork for future research in rhino detection and behavior analysis.

2.2 Related Theories
Object detection, a fundamental task in computer vision, has seen significant advancements through the evolution of deep learning models. The YOLO (You Only Look Once) series has emerged as a state-of- the-art framework, excelling in real-time detection applications due to its speed and accuracy [15]. Unlike traditional two-stage detectors, YOLO adopts a single-stage approach, segmenting images into grids and predicting bounding boxes and class probabilities simultaneously, thereby enabling efficient real-time detection [16].

YOLOv5 introduced multiple model variants (n, s, m, l, x) with different sizes and complexities, optimizing detection performance using anchor boxes refined through k-means clustering [17]. Further improvements were seen in YOLOv6, which incorporated anchor-free detection mechanisms and novel loss functions such as varifocal loss (VFL) and distribution focal loss (DFL), improving classification precision and localization accuracy [18].

YOLOv7 extended these advancements by enhancing small object recognition through multi- anchor training and focal loss reduction techniques, further improving robustness in cluttered environments [19]. YOLOv8, developed by Ultralytics, expanded its capabilities by integrating object detection, segmentation, and classification, employing anchor-free detection and mosaic augmentation for improved generalization [20].

The most recent iteration, YOLOv9, introduced cutting-edge enhancements such as Programmable Gradient Information (PGI) and Generalized Efficient Layer Aggregation Networks (GELAN), optimizing efficiency and accuracy across varying detection scenarios [21]. These progressive improvements underscore the adaptability of YOLO models in wildlife conservation, where the ability to detect, track, and monitor animals in real-time is critical for mitigating human-animal conflicts and supporting conservation efforts.

The YOLO series represents a substantial advancement in object detection, emphasizing real-time performance and high accuracy. The original YOLO framework introduced a single-pass detection approach, segmenting images into grids and simultaneously predicting bounding boxes and class probabilities. Its evolution through YOLOv5 to YOLOv9 has resulted in various enhancements, including anchor-free detection, spatial pyramid pooling, mosaic augmentation, and novel loss functions to improve precision and recall. These developments have solidified the YOLO series as a dominant approach for real-time object detection in diverse applications, particularly wildlife conservation, where real-time monitoring and accuracy are paramount.

3 Methodology

This section outlines four key parts of our approach: data creation and preparation, model selection, model training, and model evaluation and comparison. This study follows a structured approach for developing and evaluating a YOLO-based object detection model for the Greater One-Horned Rhino. The methodology is divided into four key stages: 3.1 Dataset Preparation, 3.2 Model Selection, 3.3 Model Training, and 3.4 Model Evaluation Results and Comparison.

3.1 Data Creation and Preparation

The first step involved creating and preparing a custom dataset of endangered, Greater One-Horned Rhino for object detection. This effort improves existing datasets by incorporating high-quality images with accurate annotations, diverse scenarios (varied poses, lighting, and backgrounds), and images addressing challenges such as occlusion and scale variation.

3.1.1 Initial Dataset and Analysis

- The dataset used in this study initially consisted of 4,649 training and 418 testing images.
- A performance comparison was conducted using YOLOv5, YOLOv6, YOLOv7, and YOLOv8, evaluating models at default IoU (Intersection over Union) of 0.5.
- A modification to the IoU metric is also considered to explore the impact on detection performance. As increasing the IoU threshold, the model becomes more stringent in accepting predictions as true positives leading to a higher precision. Hence, an IoU threshold of 0.85 is considered in the following experiment on YOLOv5, YOLOv6, YOLOv7 and YOLOv8 network architectures on the one-horned rhino RGB dataset.
- The best-performing model at this stage was YOLOv8L, achieving 0.975 mAP and an F1-Score of 0.95 at IoU threshold of 0.5, showing a slight improvement over the prior work.

3.1.2 Dataset Expansion and Augmentation

To further improve detection performance, the dataset was expanded and balanced:

- Additional 2,000 training and 200 testing images were collected, increasing the dataset to 6,649 training and 618 testing images.
- Data Preprocessing & Annotation: The images were resized to a fixed dimension of 640×360 pixels to ensure compatibility with the YOLO models. The bounding boxes and class labels were manually annotated using the LabelImg software. The classes included rhino and group of rhinos. The annotations followed the YOLO dataset format, specifying the image ID, class ID, and bounding box coordinates (x, y, width, height). The annotations were validated for their accuracy and consistency.
- A significant class imbalance was observed, with the "group of rhinos" class underrepresented more than rhino.
- Augmentation techniques were applied, including flipping, rotation, cropping, scaling, brightness adjustment, and blurring on manually selected images having "group of rhino" class to improve class distribution and model generalization. Some sample of image augmentation on one-horned rhino dataset is shown in Figure 1. The ImgAug Python library was utilized for advanced augmentations such as affine transformations, Gaussian noise, and saturation changes. Annotation files were updated after augmentation.

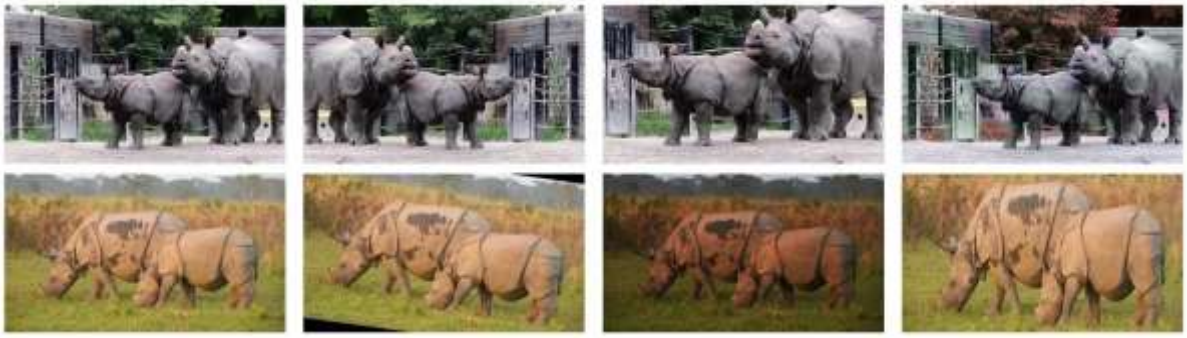


Figure 1: Sample Image Augmentation using basic image manipulation

3.1.3 Final Dataset

The final dataset after expansion and augmentation consists of:

- 6,849 training and 618 testing
- 8,096 instances of the "rhino" class.
- 1,338 instances of the "group of rhinos" class.Some sample images of the final dataset of one-horned rhino is shown in Figure 2.

Figure 2: Sample images of one-horned rhino dataset

3.2 Model Selection

The second step involves selecting object detection models based on criteria, such as speed, accuracy, robustness, and compatibility. As discussed in Section 2.2, we focus on the latest iterations of the YOLO series, which are known for their efficiency and scalability. The study explored multiple YOLO versions to determine the most effective model for one-horned rhino detection:

YOLOv5: A streamlined version of YOLOv4 built using the PyTorch framework, focusing on architectural optimization. YOLOv6: Enhances YOLOv5 with features such as self-adversarial training and cross-stage partial networks.

YOLOv7: Introduces attention mechanisms and transformer blocks, improving YOLOv6. YOLOv8: Implements spatial attention, feature fusion, and context aggregation modules for improved detection.

YOLOv9: The latest YOLO iteration is designed with the Information Bottleneck Principle and Reversible Functions to mitigate information loss in deep networks, ensuring superior accuracy and efficiency.

Model Training

The third step of our methodology involves training the YOLO models and their variants on a custom dataset using optimized hyper-parameters and settings. This section details the hardware and software configurations, training workflow, and iterative training improvements across dataset versions.

3.3.1 Hardware and Software Specifications

Hardware:

- 12^{th} Gen Intel Core i7 processor

- 32 GB RAM
- NVIDIA RTX 3070Ti GPU with 8 GB VRAM and 6,144 CUDA cores Software:
- Programming Language: Python 3.8
- Deep Learning Framework: PyTorch 1.8
- Libraries: NumPy, OpenCV, Matplotlib, and additional libraries as required

3.3.2 Training Process

- Data Loading: The custom dataset was loaded into PyTorch using the torch .utils. data. Dataset and torch. utils. data. DataLoader classes. The dataset was split into training, validation, and testing sets at an 80:10:10 ratio.
- Model Initialization: The YOLO models were initialized with pretrained weights from the COCO dataset. These weights, which are available online, served as a foundation for fine-tuning the models for the rhino specific dataset. Initialization was handled using the torch.nn.Module class.
- Loss Function: The loss function comprises a weighted sum of: o Localization Loss: Measures the error in bounding box predictions using Mean Squared Error (MSE).

o Classification Loss: Computes errors in class predictions using Binary Cross-Entropy (BCE).
o Confidence Loss: Evaluates objectness scores using BCE.

- Optimization Algorithms

– YOLOv5 & YOLOv6: Used Stochastic Gradient Descent (SGD) with momentum and weight decay.
– YOLOv7: Emphasized architectural optimizations during training, leveraging SGD and dynamic learning rate schedules.
– YOLOv8: Combined AdamW optimizer for the first 10,000 iterations, followed by SGD.
– YOLOv9: Incorporated advanced features such as a Generalized Efficient Layer Aggregation Network (GELAN) and Programmable Gradient Information (PGI).

- Learning Rate Schedules: Each model followed tailored learning rate schedules as defined in their respective original implementations.

3.4 Results and Model Comparison

The fourth stage of the methodology evaluates the YOLO models and compares their performance with alternative methods using established benchmarks and metrics. This analysis helps to identify the most effective models and highlights areas of improvement.

Performance Metrics :

Performance evaluation of object detection models involves assessing detection validity by comparing predictions to ground truth data [22]. Key metrics include True Positive (TP) for valid detections, False Positive (FP) for incorrect detections, and False Negative (FN) for missed objects. Recall measures the model's ability to detect all ground truths, while Precision evaluates its accuracy in identifying relevant objects. Intersection over Union (IoU) quantifies localization accuracy by measuring the overlap between predicted and ground truth bounding boxes. F1-Score balances precision and recall. Mean Average Precision (mAP) aggregates class-specific Average Precision (AP) values, reflecting overall performance. Evaluations often apply an IoU threshold of 0.5, with higher thresholds improving precision but potentially reducing recall. For this work, AP values were calculated for two classes (rhino and group of rhinos), with their average representing mAP. Per-class analysis highlighted model performance on the minority class (group of rhinos), and statistical tests ensured the significance of performance differences among YOLO variants and other deep learning models. These evaluation metrics and processes ensured a rigorous comparison, focusing on precision, recall, and accurate localization for robust one-horned rhino detection.

3.4.1 Initial Training (YOLOv5 to YOLOv8 on Original Dataset)

Initially, training was conducted on the existing one-horned rhino dataset using YOLOv5, YOLOv6, YOLOv7, and YOLOv8. To investigate the impact of a higher IoU threshold on detection performance, an IoU threshold of 0.85 was also considered additionally. Upon increasing the IoU threshold, the model becomes more stringent in accepting predictions as true positives leading to a higher precision.

The Model Performances Evaluation with the default IoU threshold of 0.5 and additional IoU threshold of 0.85 are shown in Table 1.

Table 1: Model Performance Evaluation

Model	Class	AP (0.5)	mAP (0.5)	F1-Score (0.5)	AP (0.85)	mAP (0.85)	F1-Score (0.85)
YOLOv5n	rhino	0.946	0.937	0.90	0.71	0.57	0.64
	group_of_rhino	0.929			0.43		
YOLOv5s	rhino	0.974	0.961	0.94	0.911	0.897	0.9
	group_of_rhino	0.948			0.883		
YOLOv5m	rhino	0.961	0.951	0.94	0.895	0.879	0.88
	group_of_rhino	0.941			0.863		
YOLOv5l	rhino	0.969	0.956	0.94	0.889	0.896	0.9
	group_of_rhino	0.942			0.903		
YOLOv5x	rhino	0.965	0.951	0.92	0.771	0.754	0.78
	group_of_rhino	0.937			0.738		
YOLOv6N	rhino	0.975	0.961	0.93	0.882	0.868	0.86
	group_of_rhino	0.947			0.855		
YOLOv6S	rhino	0.974	0.964	0.96	0.906	0.902	0.92
	group_of_rhino	0.954			0.897		
YOLOv6M	rhino	0.977	0.972	0.95	0.883	0.887	0.89
	group_of_rhino	0.967			0.891		
YOLOv6L	rhino	0.982	0.969	0.95	0.923	0.898	0.9
	group_of_rhino	0.956			0.874		
YOLOv7	rhino	0.959	0.95	0.92	0.854	0.845	0.84
	group_of_rhino	0.941			0.836		
YOLOv7-d6	rhino	0.982	0.961	0.92	0.896	0.887	0.88
	group_of_rhino	0.939			0.877		
YOLOv7-e6	rhino	0.978	0.954	0.92	0.901	0.888	0.87
	group_of_rhino	0.93			0.874		
YOLOv7-e6e	rhino	0.975	0.965	0.92	0.875	0.872	0.87
	group_of_rhino	0.955			0.868		
YOLOv7-tiny	rhino	0.972	0.953	0.91	0.868	0.84	0.84
	group_of_rhino	0.934			0.811		
YOLOv7-w6	rhino	0.983	0.966	0.94	0.908	0.892	0.88
	group_of_rhino	0.949			0.877		
YOLOv7x	rhino	0.97	0.957	0.93	0.893	0.888	0.89
	group_of_rhino	0.944			0.883		
YOLOv8N	rhino	0.977	0.969	0.95	0.903	0.902	0.89
	group_of_rhino	0.961			0.9		
YOLOv8S	rhino	0.966	0.96	0.93	0.914	0.91	0.89
	group_of_rhino	0.955			0.905		
YOLOv8M	rhino	0.974	0.969	0.94	0.924	0.931	0.91
	group_of_rhino	0.965			0.938		
YOLOv8L	rhino	0.979	0.975	0.95	0.927	0.927	0.9
	group_of_rhino	0.97			0.928		
YOLOv8X	rhino	0.964	0.971	0.95	0.912	0.91	0.9
	group_of_rhino	0.979			0.908		

The performance comparison revealed that YOLOv8L (large) delivered the highest performance, achieving a Mean Average Precision (mAP) of 0.975 and an F1-Score of 0.95, demonstrating a notable improvement over the previously evaluated models.

3.4.2 Training on Expanded dataset

In continuation of the previous step, the deep learning models YOLOv5, YOLOv6, YOLOv7 and YOLOv8 were used for performance tests on the expanded dataset.

Table 2 lists the performance on the new expanded one-horned rhino dataset. The results of the experiment on YOLOv6s (small) yielded the best result with a Mean Average Precision (mAP) of 0.976 and F1-Score of 0.96. followed closely by YOLOv8L, with an mAP of 0.976 and an F1-Score of 0.95.

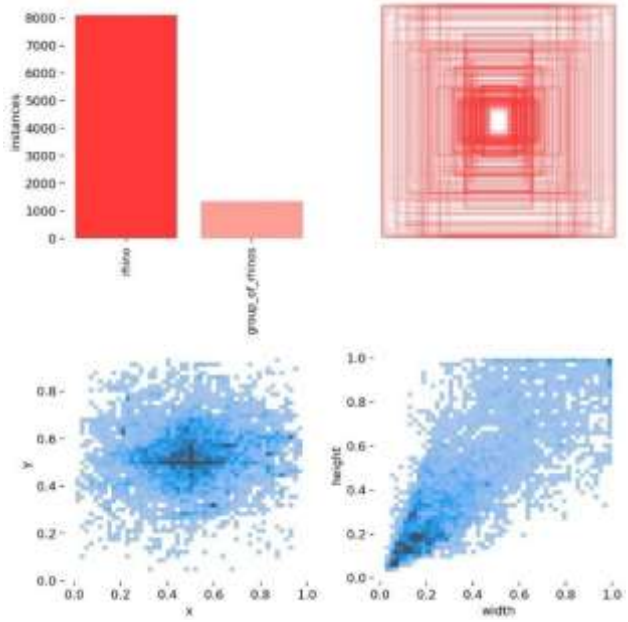
Table 2: Model Performance on Expanded dataset

Model	Class	AP	mAP	F1-Score
YOLOV5N	rhino	0.960	0.948	0.92
	group_of_rhino	0.935		
YOLOV5S	rhino	0.964	0.955	0.94
	group_of_rhino	0.947		
YOLOV5M	rhino	0.967	0.961	0.93
	group_of_rhino	0.955		
YOLOV5L	rhino	0.971	0.957	0.94
	group_of_rhino	0.943		
YOLOV5X	rhino	0.972	0.961	0.96
	group_of_rhino	0.949		
YOLOV6N	rhino	0.977	0.953	0.91
	group_of_rhino	0.929		
YOLOV6S	rhino	0.986	0.972	0.96
	group_of_rhino	0.959		
YOLOV6M	rhino	0.980	0.972	0.94
	group_of_rhino	0.965		
YOLOV6L	rhino	0.982	0.965	0.94
	group_of_rhino	0.948		
YOLOV7	rhino	0.971	0.956	0.94
	group_of_rhino	0.940		
YOLOV7-D6	rhino	0.974	0.958	0.92
	group_of_rhino	0.943		
YOLOV7-E6	rhino	0.975	0.958	0.93
	group_of_rhino	0.942		
YOLOV7-E6E	rhino	0.978	0.960	0.93
	group_of_rhino	0.942		
YOLOV7-TINY	rhino	0.973	0.955	0.92
	group_of_rhino	0.938		
YOLOV7-W6	rhino	0.984	0.976	0.94
	group_of_rhino	0.968		
YOLOV7X	rhino	0.976	0.959	0.94
	group_of_rhino	0.941		
YOLOV8N	rhino	0.967	0.965	0.94
	group_of_rhino	0.962		
YOLOV8S	rhino	0.969	0.971	0.95
	group_of_rhino	0.974		
YOLOV8M	rhino	0.966	0.960	0.93
	group_of_rhino	0.955		
YOLOV8L	rhino	0.976	0.975	0.95
	group_of_rhino	0.974		
YOLOV8X	rhino	0.973	0.969	0.93
	group_of_rhino	0.964		

There is a slight improvement over the previous result of YOLOv8L (large), yielding a Mean Average Precision (mAP) of 0.975 and F1-Score of 0.95.

However, on examining the result, it is noticed that the AP scores of the "rhino" class is much higher than the "group of rhinos," thus resulting in a overall lower mAP. Analysis showed that the Average Precision (AP) score for the rhino class significantly outperformed that of the group of rhinos class, leading to lower mean Average Precision (mAP). This imbalance stems from the dataset distribution (Figure 3) which in this work is tackled using Augmentation techniques as discussed in 3.1.2.

Figure 3: Labels distribution



3.4.3 Training on Augmented dataset (Final dataset)

This augmented dataset was subsequently used to retrain the YOLO models. By this time, YOLOv9 was also released which is also included in further experiment. Therefore, the models used are YOLOv5, YOLOv6, YOLOv7, YOLOv8, and YOLOv9.

This iterative training methodology ensures robust detection capabilities while addressing challenges, such as dataset imbalance and hyperparameter optimization. Table 3 shows the performance result on the new one-horned rhino dataset after Image Augmentation and Optimizing of Hyper-parameters.

Table 3: Model Performance on Final Dataset

Model	Class	AP	mAP	F1-Score
YOLOV5N	rhino	0.960	0.958	0.92
	group_of_rhino	0.957		
YOLOV5S	rhino	0.977	0.979	0.97
	group_of_rhino	0.982		
YOLOV5M	rhino	0.975	0.975	0.96
	group_of_rhino	0.976		
YOLOV5L	rhino	0.975	0.976	0.96
	group_of_rhino	0.978		
YOLOV5X	rhino	0.978	0.978	0.97
	group_of_rhino	0.979		
YOLOV6N	rhino	0.984	0.983	0.96
	group_of_rhino	0.982		
YOLOV6S	rhino	0.987	0.988	0.97
	group_of_rhino	0.990		
YOLOV6M	rhino	0.986	0.987	0.98

	group_of_rhino	0.987		
YOLOV6L	rhino	0.990	0.989	0.98
	group_of_rhino	0.988		
YOLOV7	rhino	0.968	0.969	0.94
	group_of_rhino	0.970		
YOLOV7-W6	rhino	0.967	0.970	0.94
	group_of_rhino	0.973		
YOLOV7X	rhino	0.973	0.976	0.95
	group_of_rhino	0.978		
YOLOV8N	rhino	0.980	0.984	0.97
	group_of_rhino	0.987		
YOLOV8S	rhino	0.982	0.985	0.97
	group_of_rhino	0.988		
YOLOV8M	rhino	0.983	0.984	0.98
	group_of_rhino	0.985		
YOLOV8L	rhino	0.986	0.987	0.98
	group_of_rhino	0.988		
YOLOV8X	rhino	0.984	0.984	0.98
	group_of_rhino	0.984		
YOLOv9	rhino	0.977	0.979	0.95
	group_of_rhino	0.980		

The results of the experiment on YOLOv6L (large) yielded the best result, with a Mean Average Precision (mAP) of 0.989 (Figure 4) and F1-Score of 0.98 (Figure 5). The detection result (inference) on one-horned rhino dataset is shown in Figure 6.

Figure 4: Performance metrics of YOLOV6L – mAP

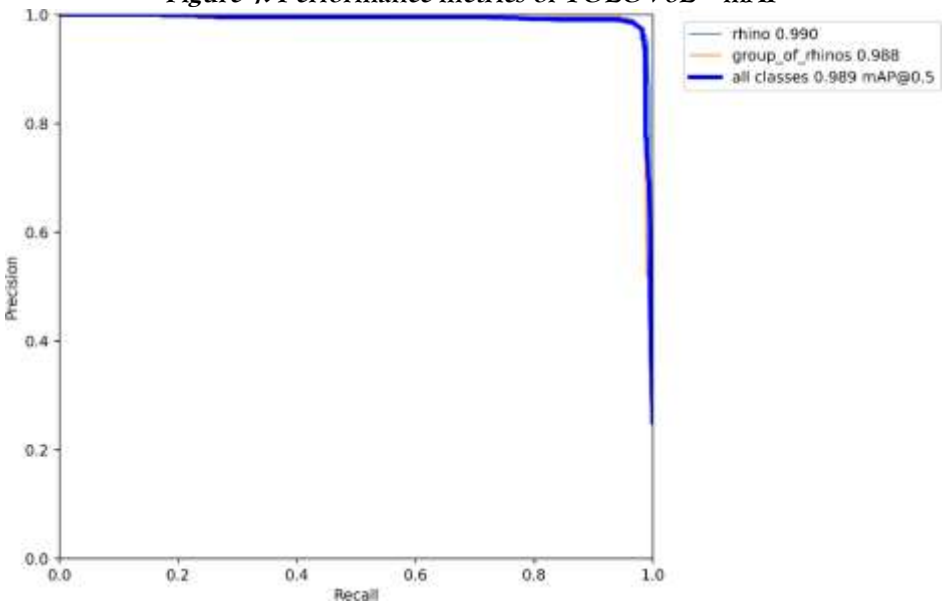


Figure 5: Performance metrics of YOLOV6L - F1-Score

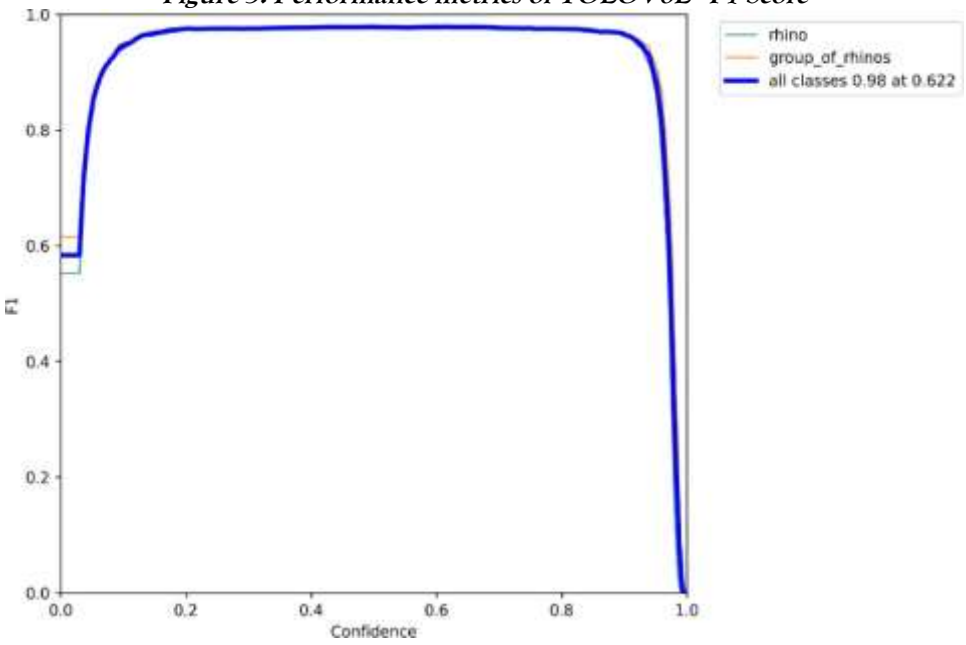


Figure 6: Detection

4 Conclusion and Future Scope

In conclusion, the experimental results demonstrate that YOLOv6, particularly the large variant (YOLOv6L), yielded the best performance for detecting the Greater One-horned Rhino, achieving a Mean Average Precision (mAP) of 0.989 and an F1-Score of 0.98 in the expanded and augmentation dataset. The consistent improvements observed across different YOLO models can be attributed to a combination of factors, including high-quality and diverse data, advanced model architectures, and robust training optimizations. YOLOv6's architectural refinements, such as CSPDarknet53 and specialized modules like SimCSPSPPF, proved especially effective in handling occlusions, class imbalances, and small object detection—key challenges in the rhino detection task.

While YOLOv8L also produced competitive results, YOLOv6 struck an optimal balance between detection quality and computational efficiency. The findings underscore the importance of integrating tailored data augmentation strategies and leveraging state-of-the-art detection frameworks when working with wildlife datasets.

This research contributes to enhancing automated monitoring systems for rhino conservation, providing a reliable approach for detecting both individual rhinos and groups under diverse and challenging

environmental conditions.

Looking ahead, while the proposed approach shows promising results in rhino detection, several areas remain for future research. First, limited data on rare behaviors like mating, territorial disputes, or aggression poses challenges for model generalization. Expanding datasets and collaborations with wildlife experts can improve detection of such behaviors. Second, ethical concerns regarding animal privacy and potential stress from monitoring technologies require attention. Developing non-invasive methods and clear ethical guidelines will ensure responsible AI deployment in conservation.

Lastly, future work can explore action detection models to classify behaviors such as aggression, normal movement, or social interactions, enhancing automated behavioral analysis for better wildlife monitoring.

These advancements will further strengthen the capabilities of AI-driven conservation tools, ensuring their effectiveness and ethical application in real-world scenarios.

Statements and Declarations

Conflict of Interest :

The authors declare that they have no conflict of interest. The authors have no competing interests to declare that are relevant to the content of this article.

Funding :

The research received no external funding. The authors have no relevant financial or non-financial interests to disclose.

Author Contributions:

- Nelson R Varte: Conceptualized the study, collected data, implemented the algorithms, and drafted the manuscript.
- Kaustubh Bhattacharyya: Provided critical insights, and contributed to manuscript revisions.
- Navajit Saikia: Oversaw the project, and approved the final manuscript for submission.

Acknowledgements:

No acknowledgements to declare.

Data Availability:

The data related to this study are available upon request from the corresponding author.

References

[1] Sharma, S., & Shah, D. J. (2017). A practical animal detection and collision avoidance system using computer vision technique. *IEEE Access*, 5, 347–358. https://doi.org/10.1109/ACCESS.2016.2642981

[2] Prabhakar, G., Kailath, B., Natarajan, S., & Kumar, R. (2017). Obstacle detection and classification using deep learning for tracking in high-speed autonomous driving. In *Proc. IEEE Region 10 Symposium (TEN-SYMP)* (pp. 1–6).

[3] Zhang, Z., He, Z., & Cao, G. (2016). Animal detection from highly cluttered natural scenes using spatiotemporal object region proposals and patch verification. *IEEE Trans. Multimedia*, 18(10), 2079–2092. https://doi.org/10.1109/TMM.2016.2594138

[4] Yousif, H., Yuan, J., & Kays, R. (2017). Fast human-animal detection from highly cluttered camera-trap images using joint background modeling and deep learning classification. In *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)* (pp. 1–4).

[5] Nguyen, H., et al. (2017). Animal recognition and identification with deep convolutional neural networks for automated wildlife monitoring. In *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)* (pp.40–49).

[6] Norouzzadeh, M. S., et al. (2017). Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *PNAS*.

[7] Zhang, Z., Han, T. X., & He, Z. (2015). Coupled ensemble graph cuts and object verification for animal

segmentation from highly cluttered videos. In *Proc. IEEE Int. Conf. Image Process.* (pp. 2830–2834).
[8] Sharma, S., et al. (2014). Automated detection of animals in context to Indian scenario. In *Proc. 5th Int. Conf. Intell. Syst. Model. Simul.* (pp. 334–338).
[9] Kellenberger, B., Volpi, M., & Tuia, D. (2017). Fast animal detection in UAV images using convolutional neural networks. In *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)* (pp. 866–869).
[10] Cheema, G. S., & Anand, S. (2017). Automatic detection and recognition of individuals in patterned species. In *Proc. Mach. Learn. Knowl. Discov. Databases, ECML PKDD* (Vol. 10536).
[11] Li, W., Swetha, S., & Shah, M. (2020). Wildlife action recognition using deep learning. *REU 2020*.
[12] Li, N., Kusakunniran, W., & Hotta, S. (2020). Detection of animal behind cages using convolutional neural network. In *Proc. 17th Int. Conf. Electr. Eng. Electron. Comput. Telecommun. Inf. Technol. (ECTI-CON)* (pp. 242–245).
[13] Ouyang, W., & Wang, X. (2013). Joint deep learning for pedestrian detection. In *Proc. IEEE Int. Conf. Comput. Vis.* (pp. 2056–2063).
[14] Choudhury, S., Bharti, N., Saikia, N., & Rajbongshi, S. C. (2020). Detection of one-horned rhino from green environment background using deep learning. *J. Green Eng.*, Oct. 2020.
[15] Redmon, J., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*.
[16] Ultralytics. (n.d.). YOLOv5. Available online: https://github.com/ultralytics/yolov5. Accessed 2023.
[17] Meituan AI. (2022). YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.
[18]Ultralytics Team. (n.d.).YOLOv6 GitHub repository. Available at: https://github.com/meituan/YOLOv6.
[19] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies setsnew state-of-the-art for real-time object detectors. *Inst. Inf. Sci., Academia Sinica, Taiwan*.
[20] Varghese, R., & S. M. (2024). YOLOv8: A novel object detection algorithm with enhanced performance and robustness. In *Proc. 2024 Int. Conf. Adv. Data Eng. Intell. Comput. Syst. (ADICS)* (pp.1–6). https://doi.org/10.1109/ADICS58448.2024.10533619
[21] Wang, C.-Y., Yeh, I.-H., & Liao, H.-Y. M. (2024). YOLOv9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616*. https://arxiv.org/abs/2402.13616
[22] Padilla, R., Netto, S. L., & Da Silva, E. A. (2020). A survey on performance metrics for object-detection algorithms. In *Proc. 2020 Int. Conf. Syst. Signals Image Process. (IWSSIP)* (pp. 237–242).