Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors

Nidhal Jegham, Chan Young Koh, Marwan Abdelatti, and Abdeltawab Hendawi

Abstract—**This study presents a comprehensive benchmark analysis of various YOLO (You Only Look Once) algorithms, from YOLOv3 to the newest addition. It represents the first research to comprehensively evaluate the performance of YOLO11, the latest addition to the YOLO family. It evaluates their performance on three diverse datasets: Traffic Signs (with varying object sizes), African Wildlife (with diverse aspect ratios and at least one instance of the object per image), and Ships and Vessels (with small-sized objects of a single class), ensuring a comprehensive assessment across datasets with distinct challenges. To ensure a robust evaluation, we employ a comprehensive set of metrics, including Precision, Recall, Mean Average Precision (mAP), Processing Time, GFLOPs count, and Model Size. Our analysis highlights the distinctive strengths and limitations of each YOLO version. For example: YOLOv9 demonstrates substantial accuracy but struggles with detecting small objects and efficiency whereas YOLOv10 exhibits relatively lower accuracy due to architectural choices that affect its performance in overlapping object detection but excels in speed and efficiency. Additionally, the YOLO11 family consistently shows superior performance in terms of accuracy, speed, computational efficiency, and model size. YOLO11m achieved a remarkable balance of accuracy and efficiency, scoring mAP50-95 scores of 0.795, 0.81, and 0.325 on the Traffic Signs, African Wildlife, and Ships datasets, respectively, while maintaining an average inference time of 2.4ms, a model size of 38.8Mb, and around 67.6 GFLOPs on average. These results provide critical insights for both industry and academia, facilitating the selection of the most suitable YOLO algorithm for diverse applications and guiding future enhancements.**

Index Terms—**YOLO (You Only Look Once), YOLO11, YOLOv10, Object detection, Ultralytics, Benchmark Analysis.**

I. INTRODUCTION AND POSITIONING

Object detection is an essential component of computer vision systems, enabling automated identification and localization of objects within images or video frames [34]. Its applications span from autonomous driving and robotics [16] [5] [20] [56] to inventory management, video surveillance, and sports analysis [4] [23] [55] [69].

Over the years, object detection has developed significantly. Initially, traditional methods such as the Viola-Jones algorithm [63] and the Deformable Part-based Model (DPM) [15] used handcrafted features and were effective for applications such as face detection [63], pedestrian detection [10], and video surveillance [3]. However, these methods had limitations in robustness and generalization. With the advancement of deep learning, network-based methods have since become the primary approach. These methods are usually classified into two categories: one-stage and two-stage approaches.

One-stage methods such as RetinaNet [32] and SSD (Single Shot MultiBox Detector) [35] perform detection in a single pass, balancing speed and accuracy. In contrast, two-stage methods, such as Region-based Convolutional Neural Networks (R-CNN) [19], generate region proposals and then perform classification, offering high precision but being computationally intensive.

Among one-stage object detection methods, YOLO (You Only Look Once) stands out for its robustness and efficiency. Initially introduced in 2015 by Redmon et al. [21], YOLO redefined object detection by predicting bounding boxes and class probabilities directly from full images in a single evaluation [47]. This innovative approach allowed YOLOv1 to achieve real-time object detection with impressive accuracy. Building upon this foundation, YOLOv2 [48] incorporated several key enhancements. It integrated the Darknet-19 framework, a 19-layer convolutional neural network that improved feature extraction. YOLOv2 also introduced batch normalization and employed data augmentation techniques inspired by the VGG architecture [57] to enhance the model's generalization. YOLOv3 [49] further advanced the model with the Darknet-53 framework, a deeper network that significantly improved feature extraction capabilities. This version also utilized a Feature Pyramid Network (FPN)-inspired design, which allowed for better detection across various object scales by combining high-level semantic features with low-level detailed features, and a Three-Scale detection mechanism that improved accuracy for objects of different sizes.

Following YOLOv3, the model's development branched into various communities, leading to several notable iterations. YOLOv4 [6], developed by Bochkovskiy et al., introduced enhancements such as Spatial Pyramid Pooling (SPP) and the Path Aggregation Network (PAN). SPP aggregates features from multiple scales, preserving spatial information, while PAN improves the fusion of features between layers, resulting in improved speed and accuracy. YOLOv5 [60] marked a significant transition by moving from the Darknet framework to PyTorch, a popular deep learning library. This transition made the model more accessible and easier to customize. The architecture incorporated strided convolution layers, which reduced computational load, and Spatial Pyramid Pooling Fast (SPPF) layers, optimizing memory usage while maintaining high performance. YOLOv6 and YOLOv7 continued this trajectory with innovative architectures. YOLOv6 [29] introduced

YOLOv3
Farhadi, A., & Redmon, J. "Yolov3: An incremental improvement"

Scaled-YOLOv4
Wang, C. Y., Bochkovskiy, A., et al. "Scaled-YOLOv4: Scaling cross stage partial network"

YOLOv6
Li, Chuyi et al. "YOLOv6: A single-stage object detection framework for industrial applications."

YOLO-NAS
DECI-AI
YOLO-NAS - GitHub

YOLOv11
Glenn Jocher (Ultralytics)
YOLOv11 GitHub

YOLO is introduced
Redmon, J. "You only look once: Unified, real-time object detection".

YOLOv5
Glenn Jocher (Ultralytics)
YOLOv5 GitHub

YOLOS
Fang, Y. et al. "You only look at one sequence: Rethinking transformer in vision through object detection.

YOLOv8
Glenn Jocher (Ultralytics)
YOLOv8 GitHub

YOLOv9
Wang, C. Y. et al. "Yolov9: Learning what you want to learn using programmable gradient information"

| Jun 8, 2015 | Apr 8, 2018 | Jun 9, 2020 | Nov 16, 2020 | Jun 1, 2021 | Jun, 2022 | Jan 10, 2023 | May 2, 2023 | Feb 21, 2024 | Sep 30, 2024 |

| Dec 25, 2016 | Apr 23, 2020 | Jul 23, 2020 | May 10, 2021 | Jul 18, 2021 | Jul 6, 2022 | Jan 13, 2023 | Jan 30, 2024 | May 23, 2024 |

YOLOv2 (YOLO9000)
Redmon, J., & Farhadi, A. "YOLO9000: better, faster, stronger"

PP-YOLO
Long, X. et al. "PP-YOLO: An effective and efficient implementation of object detector"

YOLOX
Zheng Ge et al. "YOLOX: Exceeding YOLO Series in 2021"

YOLOv6 3.0
Li, Chuyi et al. "Yolov6 v3. 0: A full-scale reloading"

YOLOv10
Wang, A., Chen H. et al. "Yolov10: Real-time end-to-end object detection"

YOLOv4
Bochkovskiy, A., et al. "Yolov4: Optimal speed and accuracy of object detection"

YOLOR
Wang, C. Y. et al. "You only learn one representation: Unified network for multiple tasks"

YOLOv7
Wang, C. Y., Bochkovskiy, A., et al. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors"

YOLO-World
Cheng, T. et al. "Yolo-world: Real-time open-vocabulary object detection"
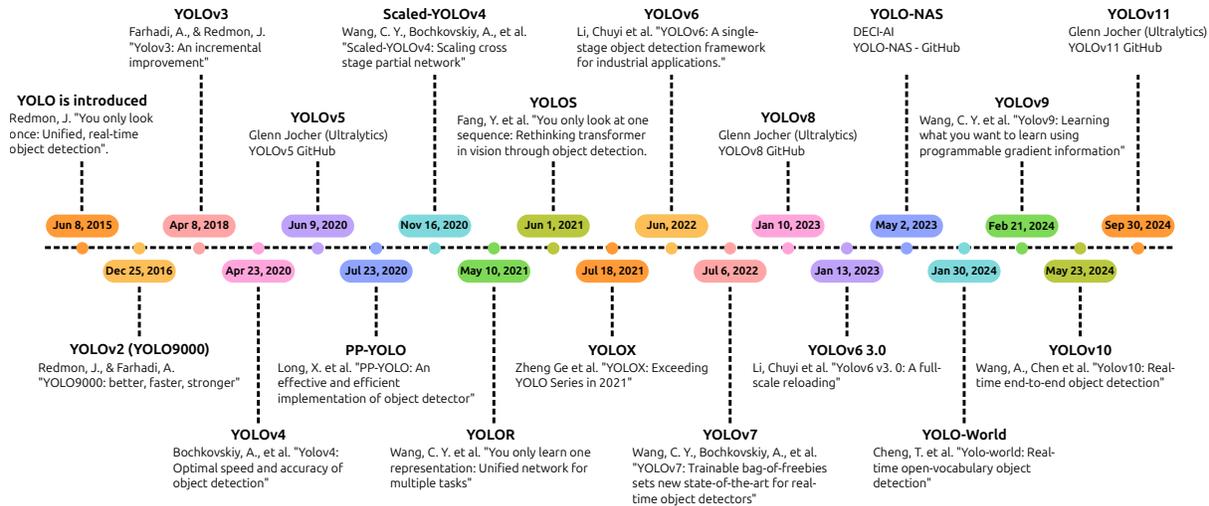
Fig. 1. Evolution of YOLO Algorithms throughout the years.

RepVGG, an architecture that simplified convolutional layers during inference, and CSPStackRep blocks, which improve accuracy by splitting the feature map into two parts to process them separately. In addition, YOLOv6 employed a hybrid channel strategy for better feature representation. YOLOv7 [65] leveraged the Extended Efficient Layer Aggregation Network (E-ELAN), a novel architecture that improved efficiency and effectiveness by enhancing information flow between layers.

The most recent versions of YOLO, including YOLOv8, YOLOv9, YOLOv10, and YOLO11 represent the forefront of the model's development. YOLOv8 [58], released by Ultralytics, introduced semantic segmentation capabilities, allowing the model to classify each pixel of an image, and provided scalable versions to meet various application needs, from resource-constrained environments to high performance systems alongside other tasks such as pose estimation, image classification, and oriented object detection (OOB). YOLOv9 [66] built on its predecessors' architectural advancements with Programmable Gradient Information (PGI), which optimizes gradient flow during training, and the Generalized Efficient Layer Aggregation Network (GELAN), which further improved performance by enhancing layer information flow. YOLOv10 [64], developed by Tsinghua University, eliminated the need for Non-Maximum Suppression (NMS) used by its predecessors, a technique used to eliminate duplicate predictions and pick the bounding boxes with the most confidence, by introducing a dual assignment strategy in its training protocol. Additionally, YOLOv10 features lightweight classification heads, spatial-channel decoupled downsampling, and rank-guided block design, making it one of the most efficient and effective YOLO models to date. Lastly, YOLO11 [26], also introduced by Ultralytics, retains the capabilities of YOLOv8 with applications such as Instance Segmentation, Pose Estimation, and Oriented Object Detection while providing 5 scalable versions for different use cases. YOLO11 replaces the C2f block from YOLOv8 with the more efficient C3k2

block, delivering improved performance without compromising speed. Additionally, it introduces the C2PSA (Cross Stage Partial with Spatial Attention) module, which improves spatial attention in feature maps, increasing accuracy, especially for small and overlapping objects.

This object detection algorithm has undergone several developments as seen in Figure 1 achieving competitive results in terms of accuracy and speed, making it the preferred algorithm in various fields such as ADAS (Advanced Driver-Assist System) [47], video surveillance [38], face detection [39], and many more [18]. For instance, YOLO plays a crucial role in the agriculture field by being implemented in numerous applications such as crop classification [1] [17], pest detection [33], automated farming [67] [37], and virtual fencing [62]. Moreover, YOLO has been utilized on numerous occasions in the field of healthcare such as cancer detection [?] [45], ulcer detection [2], medicine classification [36] [42], and health protocols enforcement [11].

In recent years, Ultralytics has played a crucial role in the advancement of YOLO by maintaining, improving, and making these models more accessible [46]. Notably, Ultralytics has streamlined the process of fine-tuning and customizing YOLO models, a task that was considerably more complex in earlier iterations. The introduction of user-friendly interfaces, comprehensive documentation, and pre-built modules has greatly simplified essential tasks such as data augmentation, model training, and evaluation. Moreover, the development of scalable model versions allows users to select models tailored to specific resource constraints and application requirements, thereby facilitating more effective fine-tuning. For instance, YOLOv8n is favorable over YOLOv8m in scenarios where speed and computational efficiency are prioritized over accuracy, making it ideal for resource-constrained environments. The integration of advanced tools for hyperparameter tuning, automated learning rate scheduling, and model pruning has further refined the customization process. Continuous updates and robust community support have also contributed to making

YOLO models more accessible and adaptable for a wide range of applications.

This paper aims to present a comprehensive comparative analysis of the YOLO algorithm's evolution. It makes a significant contribution to the field by offering the first comprehensive evaluation of YOLO11, the newest member of the YOLO family. By leveraging pre-trained models and fine-tuning them, we evaluate their performance across three diverse custom datasets, each with varying sizes and objectives. Consistent hyperparameters are applied to ensure a fair and unbiased comparison. The analysis delves into critical performance metrics, including speed, efficiency, accuracy, and computational complexity, as measured by GFLOPs count and model size. In addition, we explore the real-world applications of each YOLO version, highlighting their strengths and limitations across different use cases. Through this comparative study, we aim to provide valuable insights for researchers and practitioners, offering a deeper understanding of how these models can be effectively applied in various scenarios.

The rest of this paper is organized as follows: Section 2 covers related work. Section 3 describes the datasets, the models, and the experimental setup, including the hyperparameters and evaluation metrics used. Section 4 presents the experimental results and comparative analysis alongside a discussion. Finally, Section 5 concludes with insights drawn from the study.

II. RELATED WORK

The YOLO (You Only Look Once) algorithm is considered one of the most prominent object detection algorithms. It achieves state-of-the-art speed and accuracy, and its various applications have made it indispensable in numerous fields and industries. Numerous researchers have shown interest in this object detection algorithm by publishing papers reviewing its evolution, fine-tuning its models, and benchmarking its performance against other computer vision algorithms. This widespread interest underscores YOLO's important role in advancing the field of computer vision.

The paper in [14] examines seven semantic segmentation and detection algorithms, including YOLOv8, for cloud segmentation from remote sensing imagery. It conducts a benchmark analysis to evaluate their architectural approaches and identify the most performing ones based on accuracy, speed, and potential applications. The research aims to produce machine learning algorithms that can perform cloud segmentation using only a few spectral bands, including RGB and RGBN-IR combinations.

The authors of the paper in [22] review the evolution of the YOLO variants from version 1 to version 8, examining their internal architecture, key innovations, and benchmarked performance metrics. However, YOLOv9, YOLOv10, and YOLO11 are not considered in the analysis. The paper highlights the models' applications across domains like autonomous driving and healthcare and proposes incorporating federated learning to improve privacy, adaptability, and generalization in collaborative training. The review, however, limits its focus to mAP (mean Average Precision) for accuracy evaluation,

neglecting other key metrics such as Recall and Precision. Additionally, it considers FPS (frames per second) as the sole measure of computational efficiency, excluding the impact of preprocessing, inference, postprocessing times, GFLOPs, and size.

The paper in [12] thoroughly analyzes single-stage object detectors, particularly YOLOs from YOLOv1 to YOLOv4, with updates to their architecture, performance metrics, and regression formulation. Additionally, it provides an overview of the comparison between two-stage and single-stage object detectors, several YOLO versions from version 1 to version 4, applications utilizing two-stage detectors, and future research prospects.

The authors of the paper in [53] explore the evolution of the YOLO algorithms from version 1 to 10, highlighting their impact on automotive safety, healthcare, industrial manufacturing, surveillance, and agriculture. The paper highlights incremental technological advances and challenges in each version, indicating a potential integration with multimodal, context-aware, and General Artificial Intelligence systems for future AI-driven applications. However, the paper does not include a benchmarking study or a comparative analysis of the YOLO models, leaving out performance comparisons across the versions.

The paper in [61] explores the development of the YOLO algorithm till the fourth version. It highlights its challenges and suggests new approaches, highlighting its impact on object detection and the need for ongoing study.

The authors in the work in [27] analyze the YOLO algorithm, focusing on its development and performance. They conduct a comparative analysis of the different versions of YOLO till the 8th version, highlighting the algorithm's potential to provide insights into image and video recognition and addressing its issues and limitations. The paper focuses exclusively on the mAP metric, overlooking other accuracy measures such as Precision and Recall. Additionally, it neglects speed and efficiency metrics limiting the scope of the comparative study. The paper also omits the evaluation of the most recent models, YOLOv9, YOLOv10, and YOLO11.

This paper makes several key contributions: (i) It pioneers a comprehensive comparison of YOLO11 against its predecessors across their scaled variants from nano- to extra-large; (ii) it offers deep insights into the structural evolution of these algorithms by evaluating their performance across three diverse datasets of various object properties; and (iii) our performance evaluation extends beyond mAP and FPS to include critical metrics such as Precision, Recall, Preprocessing, Inference, and Postprocessing Time, GFLOPs, and model size. These metrics provide valuable insights to guide the selection of the optimal YOLO algorithm for specific use cases for both industry professionals and academics.

III. BENCHMARK SETUP

A. Datasets

This study aims to conduct in-depth benchmark research and assess the YOLO algorithms provided by the Ultralytics

library. The main goal is to provide a thorough and comparative analysis of these models and explain their strengths, deficiencies, and possible applications.

This paper is made possible using several publicly accessible datasets on Kaggle and Roboflow. The selection of the datasets is based on the increasing implementation of the YOLO algorithms in the fields of Autonomous driving [47] [54] [30] [7], satellite imagery [31] [8] [44], and wildlife conservation [50] [68] [51]. Moreover, each picked dataset presents unique difficulties and situations for object detection with varying image sizes and number of observations alongside the number of classes.

1) Traffic Signs Dataset: The Traffic Signs dataset by Radu Oprea is an open-source dataset on Kaggle that contains around 55 classes across 3253 training and 1128 validation images of traffic signs in different sizes and environments [40]. All of the images in the dataset are initially in a size of 640×640 with no labels for False Positives detection. Undersampling techniques were applied to this dataset to balance the different classes. This dataset is crucial for applications in autonomous driving, traffic management, road safety, and intelligent transportation systems. However, it presents challenges due to the varying sizes of target objects and the similarities in patterns across different classes, complicating the detection process.

2) Africa Wild Life Dataset: The Africa Wild Life dataset is an open-source Kaggle dataset by Bianca Ferreira, designed for real-time animal detection in nature reserves [59]. It features four common African animal classes: Buffalo, elephant, rhino, and zebra. Each class is represented by at least 376 images collected via Google image searches and manually labeled in the YOLO format. The challenges of this dataset are the varying aspect ratios, with each image containing at least one instance of the specified animal class and potentially multiple instances or occurrences of other classes. Moreover, overlapping these target objects makes the detection process more challenging. This dataset is essential for applications in wildlife conservation, anti-poaching efforts, biodiversity monitoring, and ecological research.

3) Ships/Vessels Dataset: The Ships dataset is an extensive open-source collection containing approximately 13.5k images, collected by Siddharth Sah from numerous Roboflow datasets, curated explicitly for ship detection [52]. Each image has been manually annotated with bounding boxes in the YOLO format, ensuring precise and efficient detection of ships. This dataset features a single class, "ship," allowing for streamlined and focused analysis. However, the relatively small size of the target objects and their varying rotations pose challenges for detection, particularly for the YOLO algorithm, which often struggles with small object detection and objects with varying orientations. The dataset is essential for various practical applications such as maritime safety, fisheries management, marine pollution monitoring, defense, maritime security, and more.

B. Models

1) Comparative Analysis: Ultralytics vs. Original YOLO Models: In this subsection, we will conduct a comparative

TABLE I
ULTRALYTICS-SUPPORTED LIBRARY TASKS AND MODELS

YOLO Version	Inference	Validation	Training	Pre-trained
YOLOv1	No	No	No	No
YOLOv2	No	No	No	No
YOLOv3-u (Ultralytics)	Yes	Yes	Yes	Yes
YOLOv4	No	No	No	No
YOLOv5-u (Ultralytics)	Yes	Yes	Yes	Yes
YOLOv6	Yes	Yes	Yes	No
YOLOv7	No	No	No	No
YOLOv8	Yes	Yes	Yes	Yes
YOLOv9	Yes	Yes	Yes	Yes
YOLOv10	Yes	Yes	Yes	Yes
YOLO11	Yes	Yes	Yes	Yes

analysis between the models provided by Ultralytics and their original counterparts on the Traffic Signs dataset provided by Radu Oprea [40] using the same hyperparameters in Table V. The objective is to highlight the differences between Ultralytics models and the original versions, which justifies the exclusion of YOLOv4 [6], YOLOv6 [29], and YOLOv7 [65] from this paper due to the lack of support for these models by Ultralytics. This analysis will demonstrate why focusing exclusively on Ultralytics-supported models ensures a fair and consistent benchmark evaluation.

a) Ultralytics Supported Models and Tasks:: Ultralytics library provides researchers and programmers various YOLO models for inference, validation, training, and export. Based on the results of Table I, we notice that Ultralytics does not support YOLOv1, YOLOv2, YOLOv4, and YOLOv7. Concerning YOLOv6, the library only supports the configuration *.yaml files without the pre-trained *.pt models.

b) Performance Comparison of Ultralytics and Original Models:: Based on the results of our comparative analysis of the Ultralytics models and their original counterparts on the Traffic Signs dataset presented in Table II, we observe significant discrepancies between the performance of the Ultralytics models and their original counterparts. Notably, Ultralytics' versions of YOLOv5n (nano) and YOLOv3 demonstrate superior performance, underscoring the enhancements and optimizations implemented by Ultralytics. Conversely, the original YOLOv9c (compact) slightly outperforms its Ultralytics version, potentially due to the lack of extensive optimization for this newer model by Ultralytics. These observations highlight that the Ultralytics models have undergone substantial modifications, making a direct comparison with the original versions inequitable. Consequently, the noticeable performance discrepancy between the two models, including the original and Ultralytics models in the same benchmarking study, would not provide a fair or accurate assessment. Therefore, this paper will focus exclusively on the Ultralytics-supported versions to ensure consistent and fair benchmarks.

2) YOLOv3u: YOLOv3, based on its predecessors, aims to improve localization errors and detection efficiency, particularly for smaller objects. It uses the Darknet-53 framework, which has 53 convolutional layers and achieves double the speed of ResNet-152 [49]. YOLOv3 also incorporates elements from the Feature Pyramid Network (FPN), such as residual blocks, skip connections, and up-sampling, to enhance

TABLE II
ULTRALYTICS AND ORIGINAL YOLO PERFORMANCE COMPARISON

Version	Source	mAP50	mAP50-95
YOLOv9c (compact)	Ultralytics	0.845	0.748
	Github	0.881	0.786
YOLOv5n (nano)	Ultralytics	0.756	0.663
	Github	0.429	0.367
YOLOv3	Ultralytics	0.766	0.67
	Github	0.562	0.471

its ability to detect objects efficiently across varying scales, as seen in Figure 2. The algorithm generates feature maps at three distinct scales, down-sampling the input at factors of 32, 16, and 8, and uses a three-scale detection mechanism to detect large, medium, and small-sized objects using distinct feature maps. Despite its improvements, YOLOv3 still faces challenges in achieving precise results for medium and large-sized objects, so Ultralytics released YOLOv3u. YOLOv3u is an improved version of YOLOv3 that utilizes an anchor-free detection method used later in YOLOv8 and improves upon the accuracy and speed of YOLOv3, especially for medium and large-sized objects.

Fig. 2. YOLOv3 architecture showcasing the residual blocks and the upsampling layers to enhance object detection efficiency through different scales [9].

3) YOLOv5u: YOLOv5, proposed by Glenn Jocher, transitions from the Darknet framework to PyTorch, retaining many improvements from YOLOv4 [60] [24] and utilizing CSPDarknet as its backbone. CSPDarknet is a modified version of the original Darknet architecture that incorporates Cross-Stage Partial connections by splitting feature maps into separate paths, allowing for more efficient feature extraction and reduced computational costs. YOLOv5 features a strided convolution layer with a large window size, aiming to reduce memory and computational costs, as showcased in Figure 3. Moreover, this version adopts the Spatial Pyramid Pooling Fast (SPPF) module to provide a multiscale representation of the input feature maps. The SPPF module works by pooling features at different scales and concatenating them, allowing the network to capture fine and coarse information. This helps recognize objects of various sizes more effectively. In addition, YOLOv5 implements several augmentations, such as Mosaic, copy-paste, random affine, MixUp, HSV augmentation, and random horizontal flip. YOLOv5 is available in five variants, varying in width and depth of convolution modules. Ultralytics is actively improving this model through YOLOv5u, adopting

an anchor-free detection method and achieving better overall performance, especially on complex objects of different sizes.
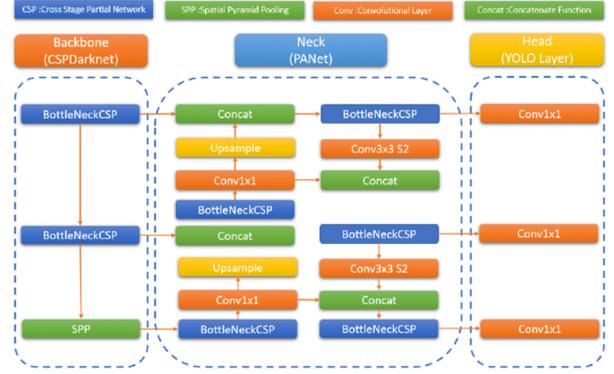

Fig. 3. Detailed architecture of YOLOv5 including the CSPDarknet Backbone, PANet Neck, and YOLO Layer Head [13].

4) YOLOv8: Ultralytics has introduced YOLOv8, a significant evolution in the YOLO series, with five scaled versions [58] [25]. Alongside object detection, YOLOv8 also provides various applications such as image classification, pose estimation, instance segmentation, and oriented object detection (OOB). Key features include a backbone similar to YOLOv5, with adjustments in the CSPLayer, now known as the C2f module, which combines high-level features with contextual information for enhanced detection accuracy highlighted in Figure 4. YOLOv8 also introduces a semantic segmentation model called YOLOv8-Seg, which combines a CSPDarknet53 feature extractor with a C2F module, achieving state-of-the-art results in object detection and semantic segmentation benchmarks while maintaining high efficiency.
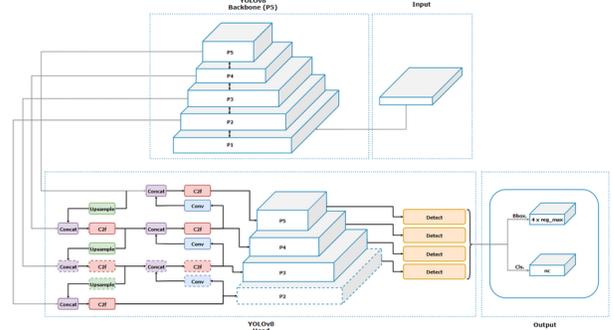

Fig. 4. Detailed architecture of YOLOv8 showcasing the backbone network's multiple convolutional layers to extract hierarchical features, the Feature Pyramid Network (FPN) for enhances detection at different scales, and the network head to perform final predictions, incorporating convolutional blocks and upsample blocks to refine features [28].

5) YOLOv9: YOLOv9, developed by Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao, uses the Information Bottleneck Principle and Reversible Functions to preserve essential data across the network's depth, ensuring reliable gradient generation and improved model convergence and performance [66]. Reversible functions, which can be inverted without loss of information, are another cornerstone

of YOLOv9's architecture. This property allows the network to retain a complete information flow, enabling more accurate updates to the model's parameters. Moreover, YOLOv9 offers five scaled versions for different uses, focusing on lightweight models, which are often under-parameterized and prone to losing significant information during the feedforward process.

Programmable Gradient Information (PGI) is a significant advancement introduced in YOLOv9. PGI is a method that dynamically adjusts the gradient information during training to optimize learning efficiency. By selectively focusing on the most informative gradients, PGI helps preserve crucial information that might otherwise be lost in lightweight models. This advancement ensures the model retains the essential features for accurate object detection, improving overall performance.

In addition, YOLOv9 incorporates GELAN (Gradient Enhanced Lightweight Architecture Network), a new architectural advancement designed to improve parameter utilization and computational efficiency as illustrated in Figure 5. GELAN achieves this by optimizing the computational pathways within the network, allowing for better resource management and adaptability to various applications without compromising speed or accuracy.
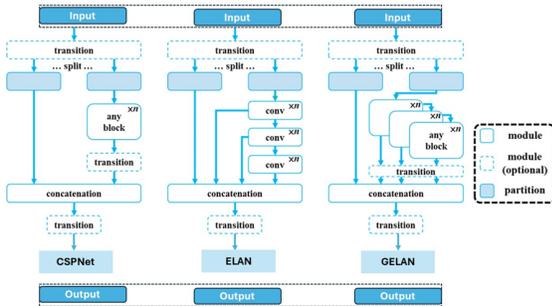

Fig. 5. YOLOv9 architecture featuring CSPNet, ELAN, and GELAN modules. CSPNet optimizes gradient flow and reduces computational complexity via feature map partitioning. ELAN enhances learning efficiency by linearly aggregating features, and GELAN extends this concept by integrating features from various depths and pathways, offering increased flexibility and accuracy in feature extraction [66].

6) YOLOv10: YOLOv10, developed by Tsinghua University researchers, builds upon previous models' strengths with key innovations [64]. The architecture has an enhanced CSP-Net (Cross Stage Partial Network) backbone for improved gradient flow and reduced computational redundancy, as seen in Figure 6. The network is structured into three main components: the backbone, the neck, and the detection head. The neck includes PAN (Path Aggregation Network) layers for effective multiscale feature fusion. PAN is designed to enhance information flow by aggregating features from different layers, enabling the network to better capture and combine details at various scales, which is crucial for detecting objects of different sizes. At the same time, the One-to-Many Head generates multiple predictions per object during training to provide rich supervisory signals and improve learning accuracy. Moreover, this version also offers five scaled versions, from nano to extralarge.

For inference, the One-to-One Head generates a single best prediction per object, eliminating the need for Non-Maximum Suppression (NMS). By removing the need for NMS, YOLOv10 reduces latency and improves the post-processing speed. In addition, YOLOv10 includes NMS-Free Training, which uses consistent dual assignments to reduce inference latency, and a model design that optimizes various components from both efficiency and accuracy perspectives. This includes lightweight classification heads, spatial-channel decoupled downsampling, and rank-guided block design. In addition, the model incorporates large-kernel convolutions and partial self-attention modules to enhance performance without significant computational costs.
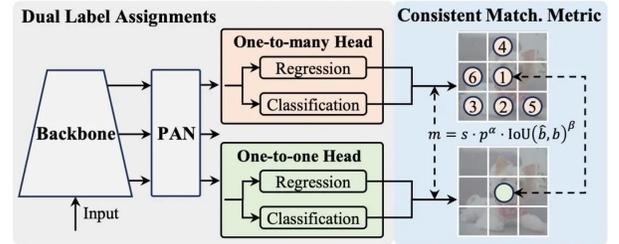

Fig. 6. YOLOv10 architecture showcasing the dual label assignment strategy for improving accuracy and the PAN layer for enhancing feature representation alongside one-to-many head for regression and classification tasks and one-to-one head for precise localization [64].

7) YOLO11: YOLO11 [26] is the latest innovation in the YOLO series developed by Ultralytics, building upon the developments of its predecessors, especially YOLOv8. This iteration offers five scaled models from nano to extra large, catering to various applications. Like YOLOv8, YOLO11 includes numerous applications such as object detection, instance segmentation, image classification, pose estimation, and oriented object detection (OBB).

Key improvements in YOLO11 include the introduction of the C2PSA (Cross-Stage Partial with Self-Attention) module, as seen in Figure 7, which combines the benefits of cross-stage partial networks with self-attention mechanisms. This enables the model to capture contextual information more effectively across multiple layers, improving object detection accuracy, especially for small and colluded objects. Additionally, in YOLO11, the C2f block has been replaced by C3k2, a custom implementation of the CSP Bottleneck that uses two convolutions, unlike YOLOv8's use of one large convolution. This block uses a smaller kernel, retaining accuracy while improving efficiency and speed.

C. Hardware and Software Setup

Table III showcases the libraries and packages used throughout this paper. During this experiment, 23 models were trained of 5 different YOLO versions found in Table IV. To ensure a fair comparative analysis, similar hyperparameters were used throughout the whole experiment on all models found below in table V. We have used 2 NVIDIA RTX 4090 GPUs for the training and evaluation, each with 16,384 CUDA cores.

For the traffic signs dataset, undersampling techniques were applied to ensure a balanced dataset, reducing the number of
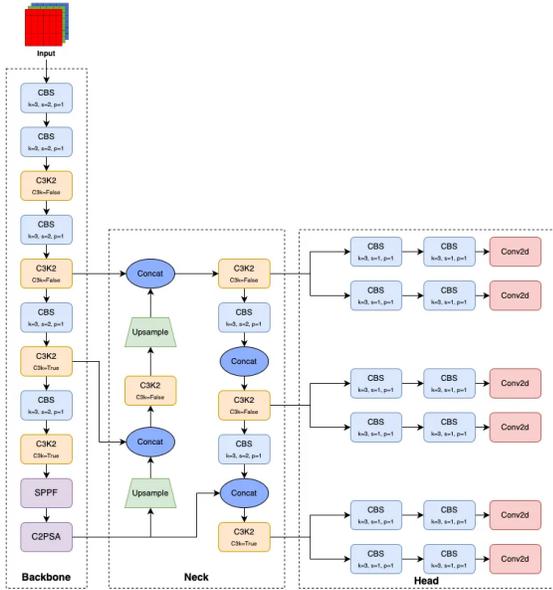
Fig. 7. YOLO11 architecture showcasing the new C3k2 blocks and the C2PSA module. [26] [43].

TABLE IV
YOLO VERSIONS AND SCALED VERSIONS

Version	Scaled Version
YOLOv3u	YOLOv3u-tiny
	YOLOv3u
YOLOv5u	YOLOv5un (nano)
	YOLOv5us (small)
	YOLOv5um (medium)
	YOLOv5ul (large)
	YOLOv5ux (extra-large)
YOLOv8	YOLOv8n (nano)
	YOLOv8s (small)
	YOLOv8m (medium)
	YOLOv8l (large)
	YOLOv8x (extra-large)
YOLOv9	YOLOv9t (tiny)
	YOLOv9s (small)
	YOLOv9m (medium)
	YOLOv9c (compact)
	YOLOv9e (extended)
YOLOv10	YOLOv10n (nano)
	YOLOv10s (small)
	YOLOv10m (medium)
	YOLOv10b (balanced)
	YOLOv10l (large)
	YOLOv10x (extra-large)
YOLO11	YOLO11n (nano)
	YOLO11s (small)
	YOLO11m (medium)
	YOLO11l (large)
	YOLO11x (extra-large)

TABLE V
TABLE OF PARAMETERS

Parameter	Value
Epochs	100
Optimizer	AdamW
Batch Size	16
Image Size	(640, 640)
Initial & Final Learning rate	(0.0001, 0.01)
Dropout rate	0.15
Data Split	(70, 20, 10)

images from 4381 to 3233 images split into 70% training, 20% validation, and 10% testing. After balancing the dataset, 24 classes remained, with an average of 100 images per class in the training dataset. This dataset contains numerous images of traffic sizes of different sizes that render it suitable for diverse object detection.

The Africa Wildlife dataset contains 1504 images distributed equally among all 4 classes following the 70% training, 20% validation, and 10% testing split. This dataset is used for large object detection in this section,

The Ships dataset contains 13.4k images and is divided into 70% training, 20% validation, and 10% testing. It has only one class (ship) and is focused on small object detection.

TABLE III
SOFTWARE SETUP

Name	Version	Description
Python	3.12	Programming language
Ubuntu	22.04	Linux operating system
CUDA	12.5	Platform for GPU based processing
cuDNN	8.9.7	CUDA library for deep neural networks
Ultralytics	8.2.55	YOLO Object Detection Library
WandB	0.17.4	ML experiment tracking

D. Metrics

This study evaluates the performance of YOLO models using three primary metrics: accuracy, computational efficiency, and size. The accuracy metrics include Precision, Recall, mAP50 (Mean Average Precision at an IoU (Intersection over Union) threshold of 0.50), and mAP50-95 (Mean Average Precision across IoU (Intersection over Union) thresholds from 0.50 to 0.95). Precision [41] measures the ratio of correctly predicted observations to the total predicted observations, thus

highlighting the occurrence of False Positives. Conversely, Recall [41] measures the ratio of correctly predicted observations to all actual observations, thus emphasizing the occurrence of False Negatives. Both mAP50 and mAP50-95 [70] provide a comprehensive summary of Precision and Recall. While mAP50 calculates the Mean Average Precision at an IoU threshold of 0.50, mAP50-95 extends this calculation across multiple IoU thresholds from 0.50 to 0.95, with a step size of 0.05.

Regarding computational efficiency metrics, Preprocessing Time, Inference Time, and Postprocessing Time will be utilized to evaluate the model's speed. Preprocessing Time refers to the duration taken to prepare raw data for input into the model. Inference Time is the duration required for the model to process the input data and generate predictions. Postprocessing Time denotes the time needed to convert the model's raw predictions into a final, usable format. These metrics were measured using a sample of images for testing after training the models. Additionally, the GFLOPs (Giga Floating-Point Operations Per Second) measure the computational power for the model training, reflecting its efficiency. In contrast, the

TABLE VI
EVALUATION RESULTS FOR THE TRAFFIC SIGNS DATASET.

Versions	Precision	Recall	mAP50	mAP50-95	Preprocess Time	Inference Time	Postprocess Time	Total Time	GFLOPs	Size
YOLOv3u	0.75	0.849	0.874	0.781	0.7	8.5	0.4	9.6	207.86	282.4
YOLOV3u tiny	0.845	0.667	0.772	0.682	1.4	0.7	0.3	2.4	24.44	19
YOLOv5un	0.805	0.679	0.749	0.665	0.6	6.6	0.4	7.6	5.65	7.1
YOLOv5us	0.85	0.777	0.827	0.744	**0.5**	7.8	0.4	8.7	18.58	23.9
YOLOv5um	0.849	0.701	0.83	0.744	1.1	9.5	0.4	11	50.54	64.1
YOLOv5ul	0.831	0.836	0.886	**0.799**	0.6	9.7	0.4	10.7	106.85	134.9
YOLOv5ux	0.863	0.795	0.867	0.777	1.1	9.8	0.4	11.3	195.2	246.3
YOLOv8n	0.749	0.688	0.777	0.689	0.6	6.8	0.4	7.8	6.55	8.1
YOLOv8s	0.766	0.788	0.806	0.718	0.6	7.8	0.4	8.8	22.59	28.6
YOLOv8m	0.838	0.805	0.845	0.763	1.6	9.1	0.4	11.1	52.12	78.9
YOLOv8l	0.771	0.789	0.853	0.767	0.6	9.2	0.4	10.2	87.77	165
YOLOv8x	**0.902**	0.744	0.874	0.78	0.6	9.4	0.4	10.4	136.9	257.7
YOLOv9t	0.792	0.748	0.812	0.731	**0.5**	10	0.4	10.9	**4.93**	7.7
YOLOv9s	0.763	0.81	0.828	0.75	0.6	11.1	0.4	12.1	15.33	26.8
YOLOv9m	0.864	0.796	0.864	0.784	1	12.1	0.4	13.5	40.98	76.7
YOLOv9c	0.827	0.807	0.852	0.769	1.3	11.6	0.4	13.3	51.8	102.6
YOLOv9e	0.819	0.824	0.854	0.764	0.8	16.1	0.4	17.3	117.5	189.4
YOLOv10n	0.722	0.602	0.722	0.64	1	0.8	**0.2**	**2**	5.59	8.3
YOLOv10s	0.823	0.742	0.834	0.744	1.2	1.1	0.2	2.5	15.9	24.7
YOLOv10m	0.834	0.843	0.88	0.781	1.2	2.4	0.2	3.8	32.1	63.8
YOLOv10b	0.836	0.764	0.859	0.765	1	3.1	0.2	4.3	39.7	98.4
YOLOv10l	0.873	0.807	0.866	0.771	1.1	3.8	0.2	5.1	50	126.8
YOLOv10x	0.773	**0.854**	0.88	0.787	1	6.3	0.2	7.5	61.4	170.4
YOLO11n	0.768	0.695	0.757	0.668	1.2	**0.6**	0.4	2.2	5.35	**6.4**
YOLO11s	0.819	0.758	0.838	0.742	1.2	1	0.4	2.6	18.4	21.4
YOLO11m	0.898	0.826	**0.893**	0.795	1.2	2.4	0.4	4	38.8	67.9
YOLO11l	0.862	0.839	0.889	0.794	1.2	3	0.4	4.6	49	86.8
YOLO11x	0.819	0.816	0.885	0.784	0.9	6.1	0.4	7.4	109	194.8

size metric reflects the actual disk size of the model and the number of its parameters.

These metrics are essential for providing a comprehensive overview of YOLO models' performance, allowing for effective comparison and evaluation. By employing these metrics, we can thoroughly assess the accuracy and efficiency of different YOLO model versions, ensuring a robust benchmark for their performance and application in various real-world scenarios.

IV. BENCHMARK RESULTS AND DISCUSSION

A. Results

1) Traffic Signs Dataset: Table VI presents a comparative analysis of the YOLO algorithms' performance on the Traffic Signs dataset, evaluated based on accuracy, computational efficiency, and model size. The Traffic Signs dataset is a medium-sized dataset with varied object sizes, making it favorable for benchmarking. The results highlight the effectiveness of YOLO models in detecting traffic signs, demonstrating a range of precision. The highest mAP50-95 was 0.799, while the lowest recorded precision was 0.64. On the other hand, the highest mAP50 is 0.893 while the lowest is 0.722. The substantial gap between the mAP50 and mAP50-95 results suggests that the models encounter difficulties in uniformly handling traffic signs with different sizes at higher thresholds, reflecting areas for potential improvement in their detection algorithms.

a) Accuracy:: As illustrated in Figure 8, YOLOv5ul demonstrates the highest accuracy, achieving a mAP50 of 0.866 and a mAP50-95 of 0.799. This is followed by YOLO11m with a mAP50-95 of 0.795 and YOLO11l with a mAP50-95 of 0.794. In contrast, YOLOv10n exhibits the lowest precision, with a mAP50 of 0.722 and a mAP50-95 of 0.64, closely followed by YOLOv5un with a mAP50-95 of 0.665, as evidenced by the data points in Figure 8.

b) Precision and Recall:: Figure 9 elucidates the trade-off between precision and recall taking the size of the models into consideration. Models such as YOLO11m, YOLO10l, YOLOv9m, YOLOv5ux, and YOLO11l exhibit high precision and recall, specifically with YOLO11m achieving a precision of 0.898 and a recall of 0.826 while having a size of 67.9Mb, and YOLOv10l achieving a precision of 0.873 and a recall of 0.807 with a significantly bigger size (126.8 Mb). In contrast, smaller models such as YOLOv10n (precision 0.722, recall 0.602), YOLOv8n (precision 0.749, recall 0.688), and YOLO11n (precision 0.768, recall 0.695) underperform in both metrics. This underscores the superior performance of larger models on the Traffic Signs dataset. Moreover, the high precision (0.849) and low recall (0.701) of YOLOv5um indicate a propensity for false negatives, while YOLOv3u's high recall (0.849) and low precision (0.75) suggest a tendency for false positives.

c) Computational Efficiency:: In terms of computational efficiency, YOLOv10n is the most efficient, with a processing time of 2ms per image and a GFLOPs count of 8.3, as shown in Figures 10 and 11. YOLO11n closely trails this at 2.2ms with a 6.4 GFLOPs count, and YOLOv3u-tiny with a processing time of 2.4ms and a GFLOPs count of 19, making it relatively computationally inefficient compared to the other fast models. However, the data indicates that YOLOv9e, YOLOv9m, YOLOv9c, and YOLOv9s are the least
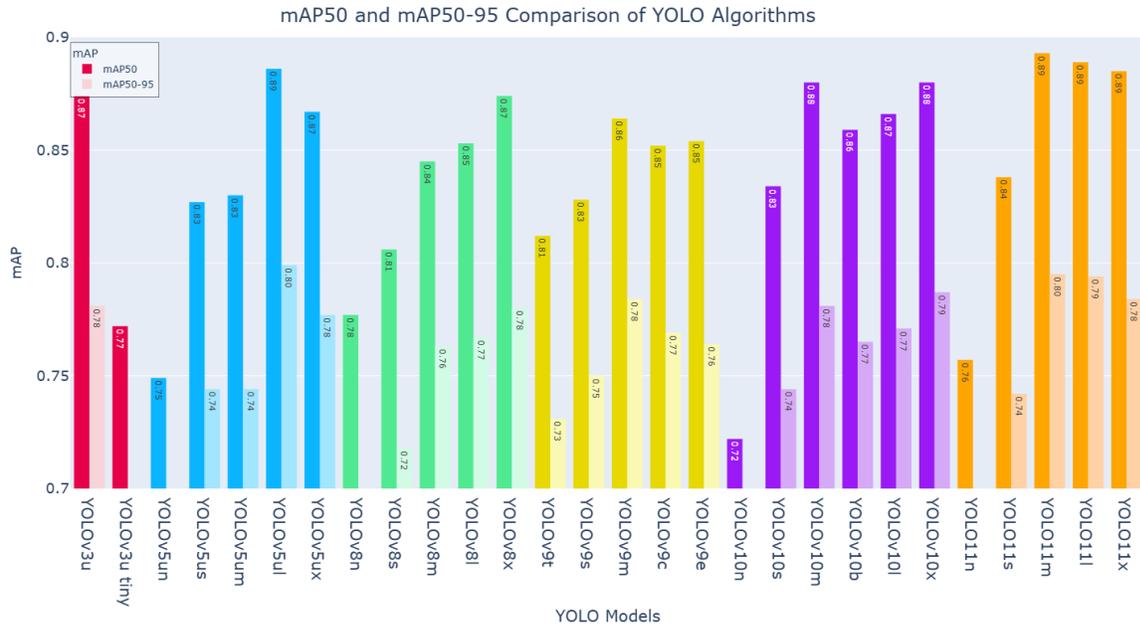
Fig. 8. mAP50 and mAP50-95 YOLO results on traffic signs dataset. Each model is represented by two bars: the left bar shows the mAP50 score, while the right bar represents the mAP50-95 score.
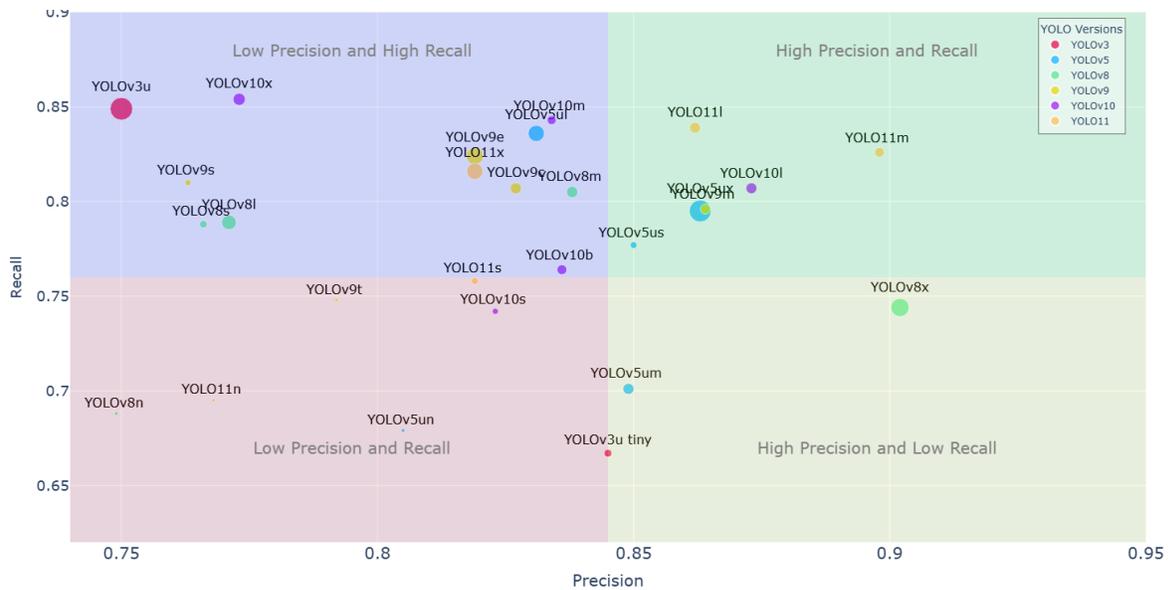

Fig. 9. Precision vs. Recall based on size results on traffic signs dataset. The size of each circle represents the size of the model, with larger circles indicating larger models.

efficient, with inference times of 16.1ms, 12.1ms, 11.6ms, and 11.1ms, and GFLOPs count of 189.4, 76.7, 102.6, and 26.8 respectively. These findings delineate a clear trade-off between accuracy and computational efficiency.

d) Overall Performance:: When evaluating overall performance, which includes accuracy, size, and model efficiency, YOLO11m emerges as a consistently top-performing model. It achieves a mAP50-95 of 0.795, an inference time of 2.4ms, a model size of 38.8Mb, and a 67.9 GFLOPs count, as detailed in Figures 8, 10, 11, and Table VI. This is followed by YOLO11l (mAP50-95 of 0.794, inference time of 4.6ms, size

of 49Mb, and 86.8 GFLOPs count), and YOLOv10m (mAP50-95 of 0.781, inference time of 2.4ms, size of 32.1Mb, 63.8 GFLOPs count). These results highlight the robustness of these models in detecting traffic signs of various sizes while maintaining short inference times and small model sizes. Notably, the YOLO11 and YOLOv10 families significantly outperform other YOLO families, in terms of accuracy and computational efficiency in this dataset, as their models consistently surpass counterparts from other families.

2) Africa Wildlife Dataset: The results in Table VII showcase the performance of the YOLO models on the Africa
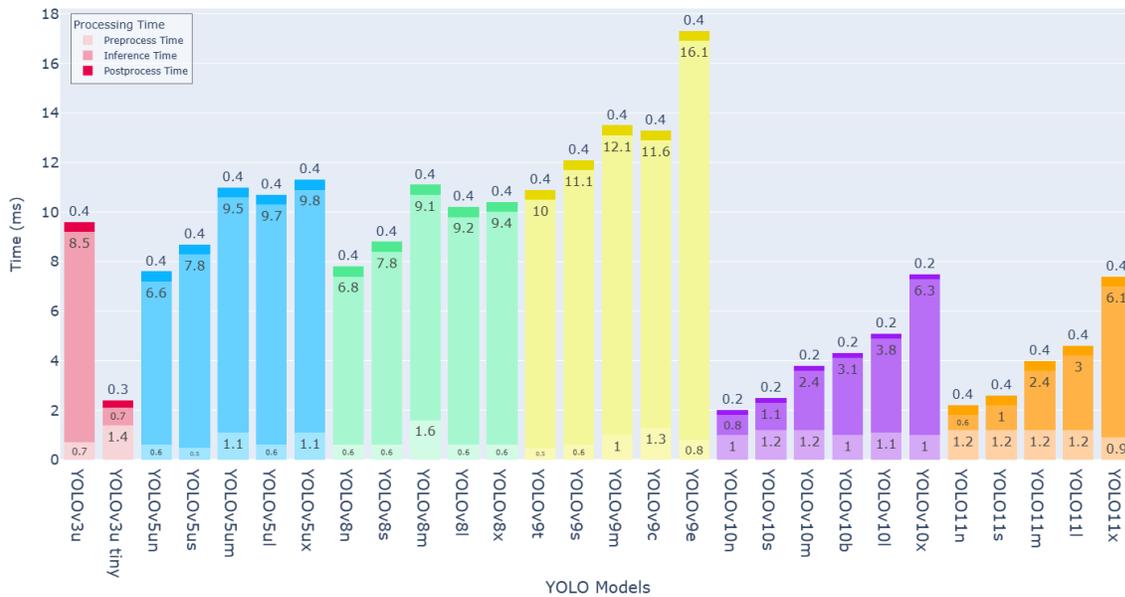
Fig. 10. Total processing time results on traffic signs dataset. Each bar represents the total processing time, divided into three sections: Preprocessing Time (bottom), Inference Time (middle), and Postprocessing Time (top).
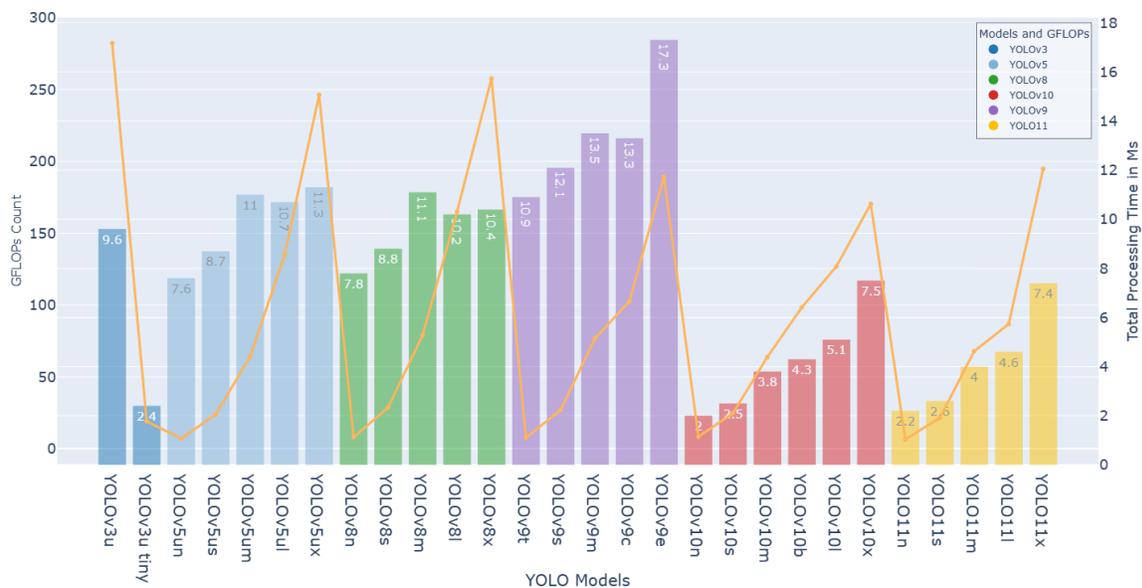

Fig. 11. Total processing time and GFLOPs count results on traffic signs dataset.

Wildlife dataset. This dataset contains large object sizes focusing on the ability of YOLO models to predict large objects and their risk of overfitting due to the size of the dataset. The models demonstrate robust accuracy across the board, with the highest-performing models achieving a mAP50-95 ranging from 0.832 to 0.725. This relatively shorter range reflects the effectiveness of the models in detecting and classifying large wildlife objects by maintaining high accuracy.

a) Accuracy:: As illustrated in Figure 12, YOLOv9s demonstrates exceptional performance with a high mAP50-95 of 0.832 and a mAP50 of 0.956, showcasing its robust accuracy across various IoU thresholds. YOLOv9c and YOLOv9t follow closely, with mAP50 scores of 0.96 and 0.948 and

mAP50-95 scores of 0.83 and 0.825, respectively. These results highlight the YOLOv9 family's ability to effectively learn patterns from a small sample of images, making it particularly suited for smaller datasets. In contrast, YOLOv5un, YOLOv10n, and YOLOv3u-tiny show lower mAP50-95 scores of 0.791, 0.786, and 0.725, indicating their limitations in accuracy. The underperformance of larger models like YOLO11x, YOLOv5ux, YOLOv5ul, and YOLOv10l can be attributed to overfitting, especially given the small dataset size.

b) Precision and Recall:: Figure 13 reveals that YOLO8l and YOLO11l achieve the highest precision and recall, with values of 0.942 and 0.937 for precision, and 0.898 and 0.896 for recall, respectively. Notably, YOLOv8n achieves

TABLE VII
EVALUATION RESULTS FOR THE AFRICA WILDLIFE DATASET.

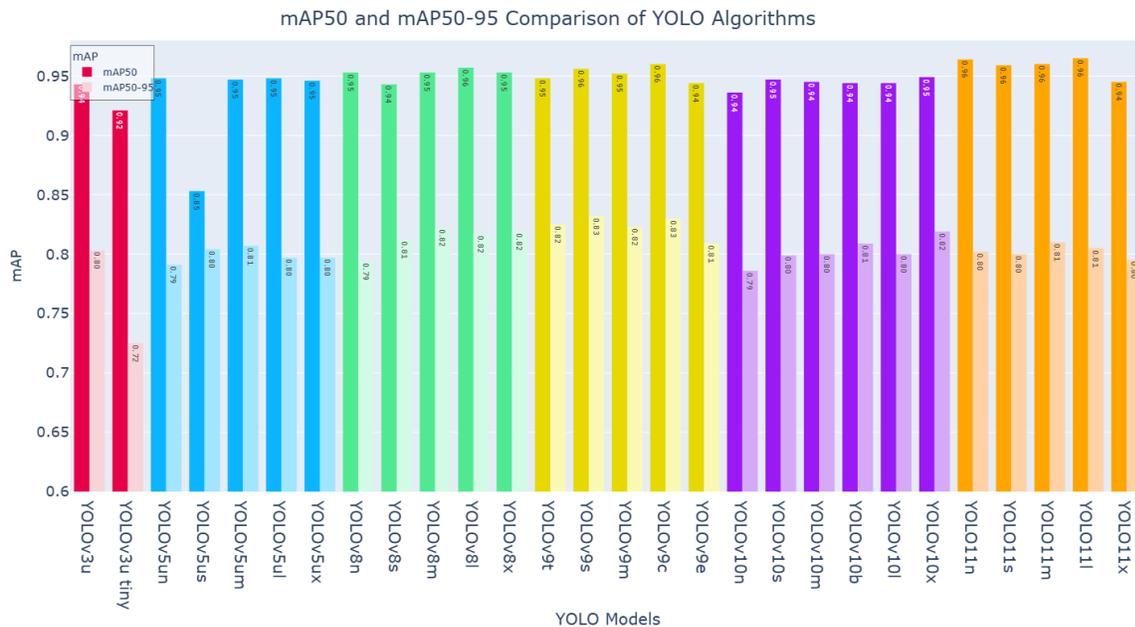| Versions	Precision	Recall	mAP50	mAP50-95	Preprocess Time	Inference Time	Postprocess Time	Total Time	Size	GFLOPs
YOLOv3u	0.91	0.88	0.943	0.803	**0.5**	6.2	0.4	7.1	207.86	282.2
YOLOV3u tiny	0.897	0.866	0.921	0.725	0.7	0.7	0.4	**1.8**	24.44	19.1
YOLOv5un	0.949	0.862	0.948	0.791	1.1	**0.5**	0.4	2	5.65	7
YOLOv5us	0.924	0.882	0.853	0.804	1	0.8	0.4	2.2	18.58	23.8
YOLOv5um	0.935	0.887	0.947	0.807	0.6	2.1	0.4	3.1	50.54	64
YOLOv5ul	0.916	0.881	0.948	0.797	0.7	3.3	0.4	4.4	106.85	135
YOLOv5ux	0.932	0.867	0.946	0.797	**0.5**	6.6	0.4	7.5	195.2	246.2
YOLOv8n	0.932	0.908	0.953	0.794	1.1	**0.5**	0.4	2	6.55	8.2
YOLOv8s	0.962	0.88	0.943	0.812	0.9	1	0.4	2.3	22.59	28.7
YOLOv8m	0.928	**0.909**	0.953	0.822	0.8	2.5	0.4	3.7	52.12	78.9
YOLOv8l	0.942	0.898	**0.957**	0.817	0.9	3.9	0.4	5.2	87.77	165.1
YOLOv8x	0.912	0.906	0.953	0.819	**0.5**	7.1	0.4	8	136.9	257.6
YOLOv9t	0.944	0.875	0.948	0.825	1.3	1.1	0.4	2.8	**4.93**	**7.7**
YOLOv9s	0.921	0.897	0.956	**0.832**	1	1.2	0.4	2.6	15.33	26.9
YOLOv9m	0.924	0.901	0.952	0.823	0.9	2.8	0.4	4.1	40.98	76.5
YOLOv9c	0.934	0.897	0.96	0.83	0.9	3.4	0.4	4.7	51.8	102.7
YOLOv9e	0.932	0.864	0.944	0.809	**0.5**	7.6	0.4	8.5	117.5	189.3
YOLOv10n	0.901	0.9	0.936	0.786	1.1	0.7	**0.2**	**2**	5.59	8.2
YOLOv10s	0.929	0.888	0.947	0.799	0.9	1.1	**0.2**	2.2	15.9	24.5
YOLOv10m	0.91	0.88	0.945	0.8	1	2.4	**0.2**	3.6	32.1	63.4
YOLOv10b	0.905	0.899	0.944	0.809	0.8	3.2	**0.2**	4.2	39.7	98
YOLOv10l	0.922	0.894	0.944	0.8	0.7	3.8	**0.2**	4.7	50	126.3
YOLOv10x	0.96	0.862	0.949	0.819	0.8	6.3	**0.2**	7.3	61.4	169.8
YOLO11n	**0.964**	0.877	0.964	0.802	1.1	0.7	0.4	2.2	5.35	**6.3**
YOLO11s	0.952	0.892	0.959	0.8	1.1	1	0.4	2.5	18.4	21.3
YOLO11m	0.922	0.906	0.96	0.81	0.9	2.4	0.4	3.7	38.8	67.7
YOLO11l	0.937	0.896	**0.965**	0.805	0.9	3	0.4	4.3	49	86.6
YOLO11x	0.908	0.886	0.945	0.795	0.6	6.1	0.4	7.1	109	194.4

Fig. 12. mAP50 and mAP50-95 YOLO results on Africa wildlife dataset. Each model is represented by two bars: the left bar shows the mAP50 score, while the right bar represents the mAP50-95 score.
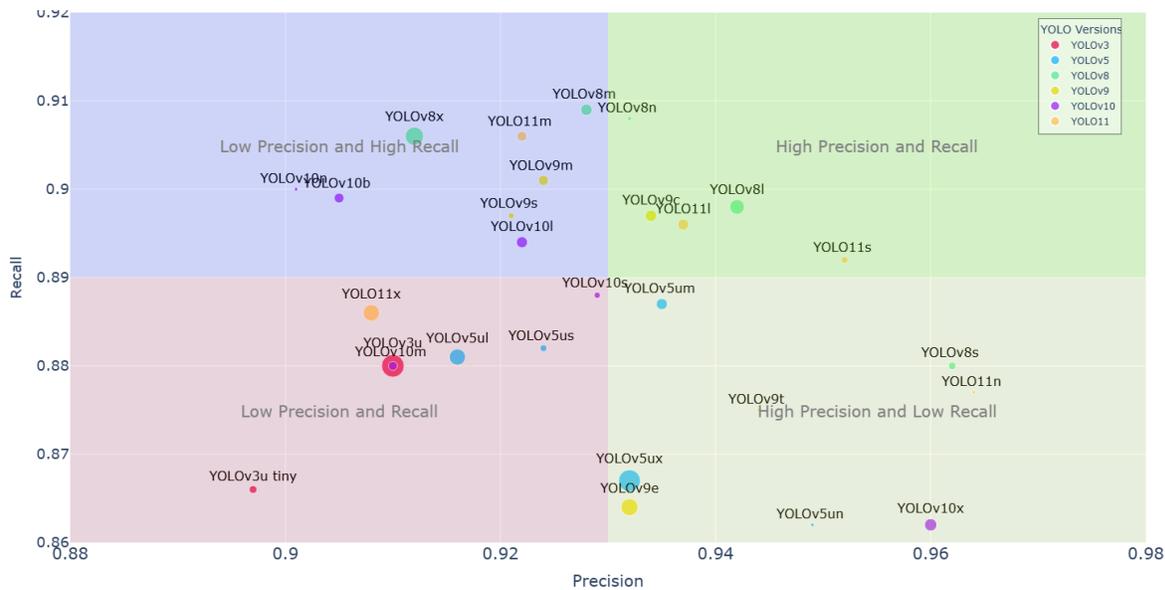
Fig. 13. Precision vs. Recall based on size results on Africa wildlife dataset. The size of each circle represents the size of the model, with larger circles indicating larger models.
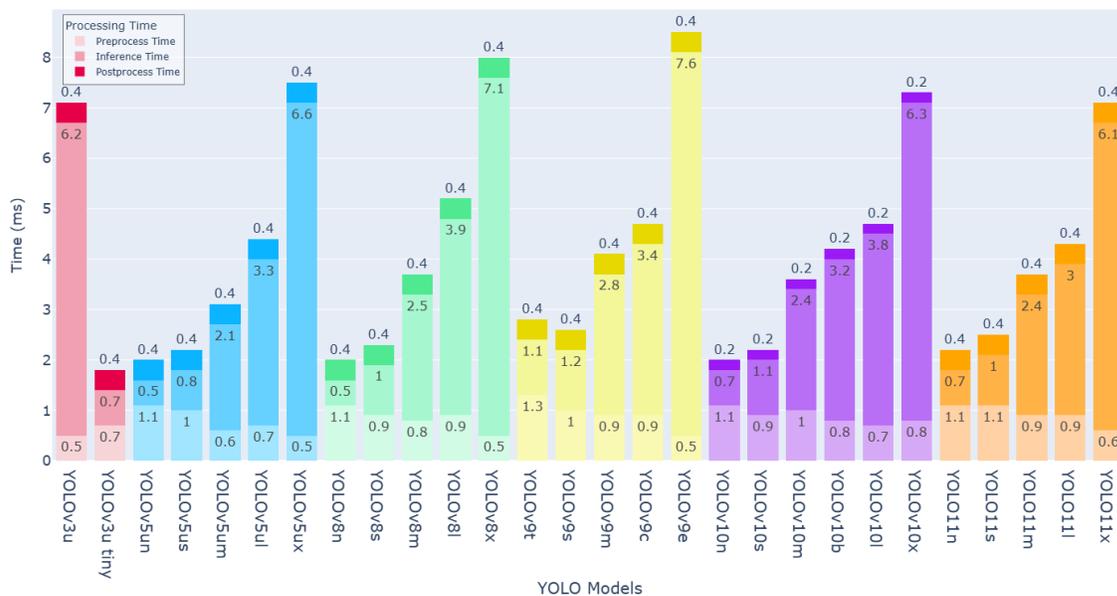

Fig. 14. Total processing time results on Africa wildlife dataset. Each bar represents the total processing time, divided into three sections: Preprocessing Time (bottom), Inference Time (middle), and Postprocessing Time (top).

similar results (0.932 for precision, 0.908 for recall) with a compact size of 6.55Mb, demonstrating its efficiency. In contrast, YOLOv3u and YOLOv5ul exhibit lower precision and recall scores (0.91 and 0.88 for YOLOv3u, 0.916 and 0.881 for YOLOv5ul), despite their larger sizes (204.86Mb for YOLOv3u, 106.85Mb for YOLOv5ul), which may be attributed to overfitting issues.

c) Computational Efficiency:: YOLOv10n, YOLOv8n, and YOLOv3u-tiny are the fastest models, achieving processing times of 2ms and 1.8ms, with GFLOPs counts of 8.2 and 19.1, respectively. The first two models share the same processing speed and GFLOPs count, as showcased in

Figures 14 and 15. Conversely, YOLOv9e exhibits the slowest processing time at 11.2ms and a GFLOPs count of 189.3, followed by YOLOv5ux at 7.5ms and 246.2 GFLOPs count. These results indicate that larger models tend to require more processing time and hardware usage compared to smaller models, emphasizing the trade-off between model size and processing efficiency.

d) Overall Performance:: YOLOv9t and YOLOv9s consistently excel across all metrics, delivering high accuracy while maintaining small model sizes, low GFLOPs, and short inference times, as shown in Table VII, and Figures 13, 14, and 15. This demonstrates the robustness of YOLOv9's smaller
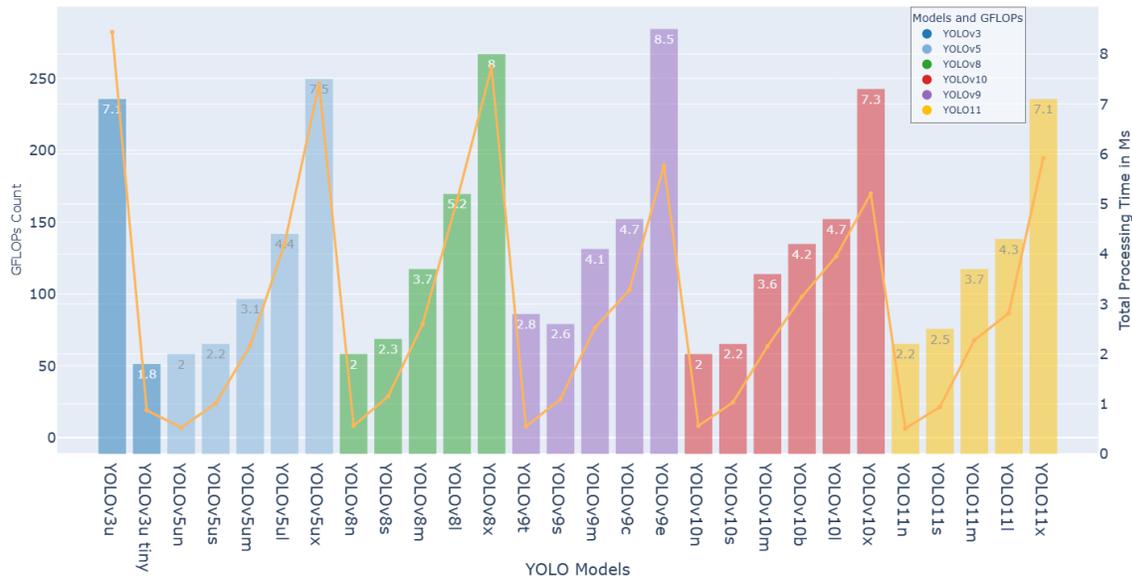
Fig. 15. Total processing time and GFLOPs count results on Africa wildlife dataset.

models and their effectiveness on small datasets. In contrast, YOLO5ux and YOLO11x show suboptimal accuracy despite their larger sizes and longer inference times, likely due to overfitting. Most large models underperformed on this dataset, with the exception of YOLOv10x, which benefited from a modern architecture that prevents overfitting.

3) Ships and Vessels Dataset: Table VIII presents the performance of YOLO models on the Ships and Vessels dataset, a large dataset featuring tiny objects with varying rotations. Overall, the models demonstrated moderate effectiveness in detecting ships and vessels, with mAP50-95 ranging from 0.273 to 0.327. This performance suggests that YOLO algorithms may face challenges in accurately detecting smaller objects, and the dataset's diversity in object sizes and rotations provides a comprehensive test of the models' capabilities in these conditions.

a) Accuracy: : The disparity between mAP50-95 and mAP50, illustrated in Figure 16, underscores the challenges YOLO models face with higher IoU thresholds when detecting small objects. Additionally, YOLO models struggle with detecting objects of varying rotations. Among the models, YOLO11x achieved the highest accuracy, with a mAP50 of 0.529 and a mAP50-95 of 0.327, closely followed by YOLO11l, YOLO11m, and YOLO11s, which recorded mAP50 values of 0.529, 0.528, and 0.53, and mAP50-95 values of 0.327, 0.325, and 0.325, respectively. These results highlight the robustness of the YOLO11 family in detecting small and tiny objects. In contrast, YOLOv3u-tiny, YOLOv8n, YOLOv3u, and YOLOv5n exhibited the lowest accuracy, with mAP50 scores of 0.489, 0.515, 0.519, and 0.514, and mAP50-95 scores of 0.273, 0.297, 0.298, and 0.298, respectively. This suggests the outdated architecture of YOLOv3u and the potential underfitting of smaller models due to the large dataset size.

b) Precision and Recall: : Figure 17 indicates that YOLOv5ux outperformed other models, achieving a precision

of 0.668 and a recall of 0.555. It was closely followed by YOLOv9m (precision of 0.668, recall of 0.551) and YOLOv8m (precision of 0.669, recall of 0.525), both of which are significantly smaller in size (40.98 Mb for YOLOv9m and 52.12 Mb for YOLOv8m). In contrast, YOLO11n and YOLOv10s exhibited lower performance, with precisions of 0.574 and 0.586 and recalls of 0.51 and 0.511, respectively, likely due to underfitting issues. Generally, YOLO11 models tended to produce false positives, reflected in their low precision and high recall. Meanwhile, YOLOv10 underperformed in both precision and recall, despite being one of the newest models in the YOLO family.

*c) Computational Efficiency:: * As illustrated in Figures 18 and 19, YOLOv3u-tiny achieved the fastest processing time at 2 ms, closely followed by YOLOv8n and YOLOv5un, both recording 2.3 ms. YOLOv10 and YOLO11 models also excelled in speed, with YOLOv10n and YOLO11n achieving rapid inference times of 2.4 ms and 2.5 ms, along with GFLOPs counts of 8.2 and 6.3, respectively. In contrast, YOLOv9e exhibited the slowest speed, with an inference time of 7.6 ms and a GFLOPs count of 189.3, highlighting the trade-off between accuracy and efficiency within the YOLOv9 family.

*d) Overall Performance:: * The results in Table VIII and Figures 16, 17, and 18 demonstrate that YOLO11s and YOLOv10s excelled in accuracy while maintaining compact sizes, low GFLOPs, and quick processing times. In contrast, YOLOv3u, YOLOv8x, and YOLOv8l fell short of expectations despite their larger sizes and longer processing times. These findings highlight the robustness and reliability of the YOLO11 family, particularly in improving the YOLO family's performance in detecting small and tiny objects while ensuring efficient processing. Additionally, the results reveal the underperformance of YOLOv9 models when faced with large datasets and small objects, despite their modern architecture.

TABLE VIII
EVALUATION RESULTS FOR THE SHIPS AND VESSELS DATASET.

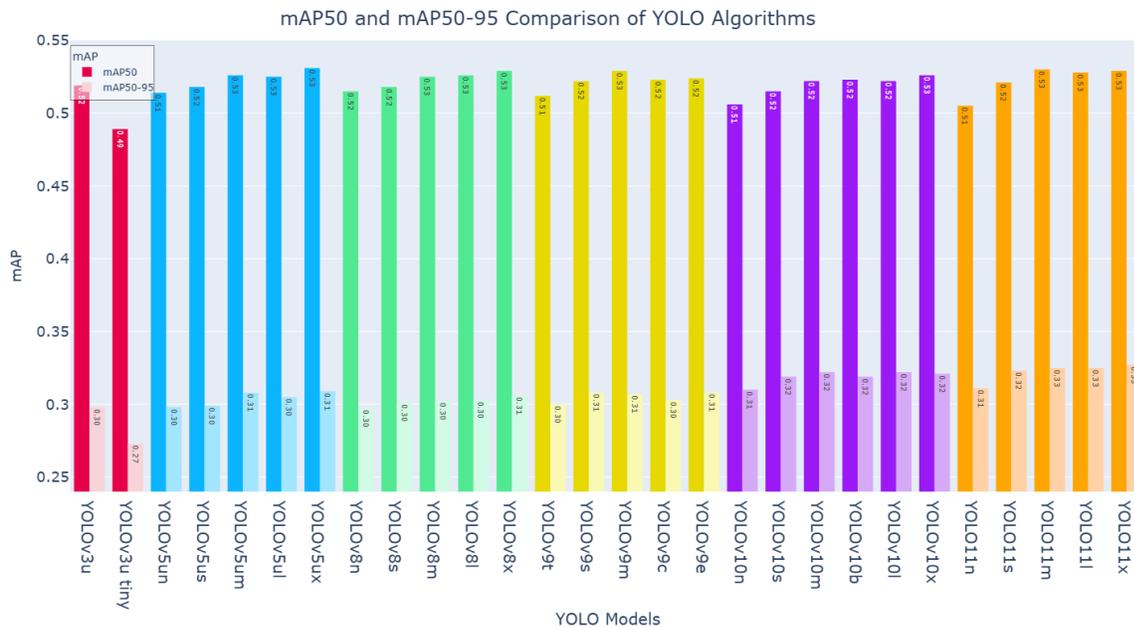| Versions	Precision	Recall	mAP50	mAP50-95	Preprocess Time	Inference Time	Postprocess Time	Total Time	Size	GFLOPs
YOLOv3u	**0.679**	0.534	0.519	0.298	**0.8**	6.2	0.3	7.3	207.86	282.5
YOLOV3u tiny	0.647	0.511	0.489	0.273	1	0.7	0.3	**2**	24.44	18.9
YOLOv5un	0.635	0.532	0.514	0.298	1.5	0.6	0.3	2.4	5.65	7.2
YOLOv5us	0.653	0.541	0.518	0.299	1.2	0.8	0.3	2.3	18.58	24
YOLOv5um	0.667	0.541	0.526	0.308	0.9	2.1	0.3	3.3	50.54	64
YOLOv5ul	0.654	0.545	0.525	0.305	0.9	3.3	0.3	4.5	106.85	134.8
YOLOv5ux	0.668	**0.555**	**0.531**	0.309	**0.8**	6.7	0.3	7.8	195.2	246.2
YOLOv8n	0.655	0.533	0.515	0.297	1.5	**0.5**	0.3	2.3	6.55	8
YOLOv8s	0.647	0.545	0.518	0.301	1.1	1	0.3	2.4	22.59	28.5
YOLOv8m	0.669	0.547	0.525	0.302	**0.8**	2.5	0.3	3.6	52.12	79
YOLOv8l	0.659	0.551	0.526	0.303	0.9	3.9	0.3	5.1	87.77	165
YOLOv8x	0.655	0.55	0.529	0.306	**0.8**	7.1	0.3	8.2	136.9	257.7
YOLOv9t	0.647	0.516	0.512	0.3	1.4	1.1	0.3	2.8	**4.93**	**7.5**
YOLOv9s	0.655	0.552	0.522	0.308	1.4	1.2	0.3	2.9	15.33	26.9
YOLOv9m	0.668	0.551	0.529	0.307	1.1	2.7	0.3	4.1	40.98	76.8
YOLOv9c	0.663	0.547	0.523	0.303	1.2	3.4	0.3	4.9	51.8	102.4
YOLOv9e	0.667	0.537	0.524	0.308	1.1	7.6	0.3	9	117.5	189.5
YOLOv10n	0.584	0.487	0.506	0.31	1.4	0.8	0.2	2.4	5.59	8.2
YOLOv10s	0.586	0.511	0.515	0.319	1.1	1.1	0.2	2.4	15.9	24.4
YOLOv10m	0.588	0.517	0.522	0.322	1	2.4	**0.1**	3.5	32.1	63.4
YOLOv10b	0.603	0.509	0.523	0.319	1.1	3.2	**0.1**	4.4	39.7	97.9
YOLOv10l	0.601	0.511	0.522	0.322	1.1	3.8	**0.1**	5	50	126.3
YOLOv10x	0.6	0.523	0.526	0.321	1	6.3	0.2	7.5	61.4	169.8
YOLO11n	0.574	0.51	0.505	0.311	1.5	0.7	0.3	2.5	5.35	**6.3**
YOLO11s	0.585	0.535	0.521	0.323	1.3	1	0.3	2.6	18.4	21.3
YOLO11m	0.588	0.541	0.53	0.325	1	2.4	0.3	3.7	38.8	67.6
YOLO11l	0.596	0.531	0.528	0.325	1.1	3	0.4	4.5	49	86.6
YOLO11x	0.596	0.538	0.529	**0.327**	**0.8**	6.1	0.3	7.2	109	194.4

Fig. 16. mAP50 and mAP50-95 YOLO results on ships and vessel dataset. Each model is represented by two bars: the left bar shows the mAP50 score, while the right bar represents the mAP50-95 score.
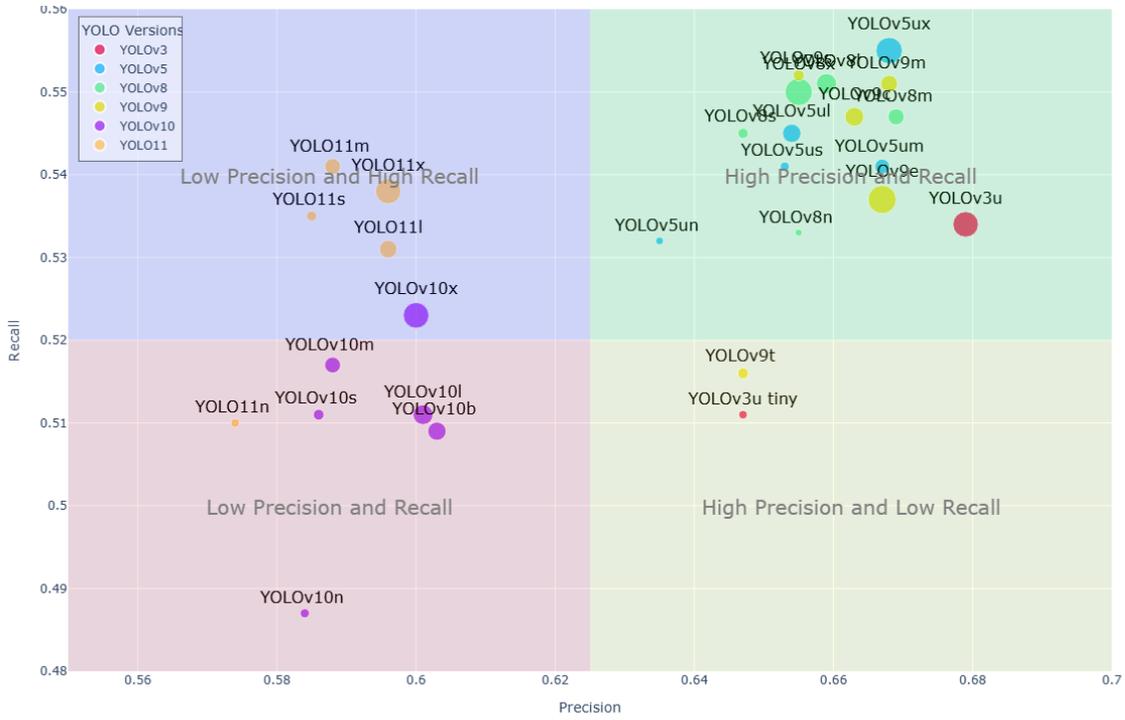
Fig. 17. Precision vs. Recall based on size results on ships and vessels dataset. The size of each circle represents the size of the model, with larger circles indicating larger models.

Fig. 18. Total processing time results on ships and vessels dataset. Each bar represents the total processing time, divided into three sections: Preprocessing Time (bottom), Inference Time (middle), and Postprocessing Time (top).
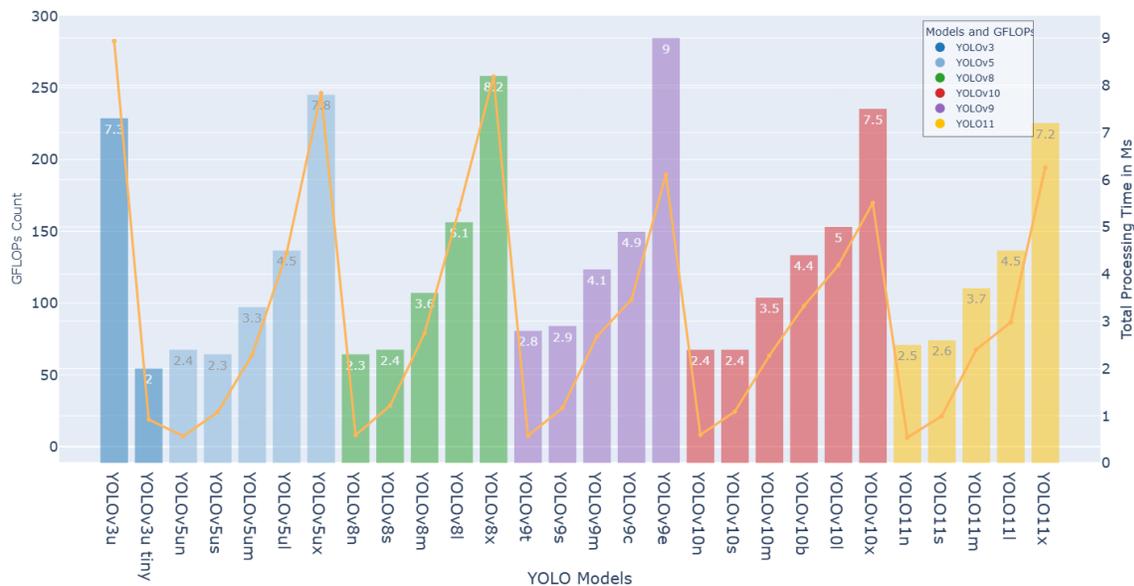
Fig. 19. Total processing time and GFLOPs count results on ships and vessels dataset.

B. Discussion

Based on the performance of the models across the three datasets, we ranked them by accuracy, speed, GFLOps count, and size, as shown in Table IX to facilitate a comprehensive evaluation. For accuracy, the mAP50-95 metric was employed due to its capacity to assess models across a range of IoU thresholds, thus providing a detailed insight into each model's performance. For speed, models were sorted based on the total processing time, which encompasses preprocessing, inference, and postprocessing durations. The rankings range from Rank 1, indicating the highest performance, to Rank 28, denoting the lowest, with the respective rankings highlighted in bold within the table.

The analysis of Table IX yields several critical observations:

1) Accuracy: YOLO11m consistently emerged as a top performer, frequently ranking among the highest, closely followed by YOLOv10x, YOLO11l, YOLOv9m, and YOLO11x. This underscores the robust performance of the YOLO11 family across varying IoU thresholds and object sizes, which can be attributed to their use of C2PSA for the preservation of contextual information, leading to improved convergence and overall performance. In addition, the implementation of large-kernel convolutions and partial self-attention modules helped increase the performance of the algorithm.

Conversely, YOLOv3u-tiny exhibited the lowest accuracy, particularly in the Africa Wildlife and Ships and Vessels datasets, with YOLOv5un and YOLOv8n showing slightly better but still sub-par results. This suggests that YOLO11 models are currently the most reliable for applications demanding high accuracy.

Closely following the performance of the YOLO11 family, the YOLOv9 models demonstrate their effectiveness in detecting objects across various sizes and different IoU thresholds. However, they may struggle with small objects, as seen in the Ships and Vessels dataset. In contrast, the YOLOv10

TABLE IX
OVERALL RANKING OF YOLO ALGORITHMS

Version	Accuracy Rank	Speed Rank	GFLOPs Rank	Size Rank
YOLOv3u-tiny	**28**	**1**	6	11
YOLOv3u	20	24	**28**	**28**
YOLOv5un	27	6	2	4
YOLOv5us	24	7	8	9
YOLOv5um	17	15	13	18
YOLOv5ul	14	19	21	23
YOLOv5ux	17	27	26	27
YOLOv8n	26	5	4	5
YOLOv8s	23	9	11	10
YOLOv8m	15	17	16	20
YOLOv8l	13	22	22	22
YOLOv8x	8	26	27	26
YOLOv9t	20	12	3	**1**
YOLOv9s	7	15	10	6
YOLOv9m	4	21	15	15
YOLOv9c	9	25	19	19
YOLOv9e	12	**28**	24	25
YOLOv10n	25	2	5	3
YOLOv10s	19	3	9	7
YOLOv10m	5	10	12	12
YOLOv10b	9	12	18	14
YOLOv10l	11	17	20	17
YOLOv10x	2	22	23	21
YOLO11n	22	3	**1**	2
YOLO11s	16	8	7	8
YOLO11m	**1**	11	14	13
YOLO11l	3	14	17	16
YOLO11x	5	19	25	24

family, despite its later introduction, exhibited relatively lower accuracy in the Traffic Signs and Africa Animals datasets, resulting in an average accuracy drop of 2.075% compared to the YOLOv9 models in those datasets. The slight under-performance of YOLOv10 can be attributed to its adoption of the One-to-One Head approach instead of Non-Maximum Suppression (NMS) for defining bounding boxes. This strategy can struggle to capture objects effectively, particularly when dealing with overlapping items, as it relies on a single pre-

diction per object. This limitation helps explain the relatively subpar results observed in the second dataset.

Similarly, the outdated architecture of YOLOv3u contributed to its inferior performance, averaging 6.5% lower accuracy than the YOLO11 models. This decline can be traced back to its reliance on the older Darknet-53 framework, first introduced in 2018, which may not adequately address contemporary detection challenges.

2) Computational Efficiency: YOLOv10n consistently outperformed other models in terms of speed and GFLOPs count, ranking among the top across all three datasets in terms of speed and 5th in terms of GFLOPs count. YOLOv3u-tiny, YOLOv10s, and YOLO11n also demonstrated notable computational efficiency.

YOLOv9e exhibited the slowest inference times and a very high GFLOPs count across the datasets, illustrating the trade-off between accuracy and efficiency. YOLO11's speed improvements, attributable to their use of the C3k2 block, make it suitable for applications where rapid processing is essential, surpassing YOLOv10 and YOLOv9 models, in terms of speed by %1.41 and %31 on average, respectively.

While YOLOv9 models excelled in accuracy, their inference times were among the slowest, making them less ideal for time-sensitive applications. In contrast, YOLOv10 models, though slightly slower than the YOLO11 variants, still offer a commendable balance between efficiency and speed. Their performance is well-suited for time-sensitive scenarios, providing rapid processing without significantly sacrificing accuracy, making them a viable option for real-time applications.

3) Model Size: YOLOv9t was the smallest model, ranking first across all three datasets, followed by YOLO11n and YOLOv10n. This efficiency in model size underscores the advancements in newer YOLO versions, especially YOLOv10, showcasing the effectiveness of implementing the Spatial-Channel Decoupled Downsampling for efficient parameter utilization.

YOLOv3u was the largest model, highlighting its inefficiency compared to its more modern counterparts due to its outdated architecture.

4) Overall Performance: Considering accuracy, speed, size, and GFLOPs, YOLO11m, YOLOv11n, YOLO11s, and YOLOv10s emerged as the most consistent performers. They achieved high accuracy, low processing time and power, and efficient disk usage, making them suitable for a wide range of applications where both speed and accuracy are crucial.

Conversely, YOLOv9e, YOLOv5ux, and YOLOv3u demonstrated poor results across all metrics, being computationally inefficient and underperforming relative to their sizes. YOLO11 models showed the best overall performance, likely due to recent enhancements such as the C3k2 block and C2PSA module. Following closely, YOLOv10 models, despite slightly underperforming in accuracy excelled in efficiency thanks to its use of implementation of One-to-One head for prediction. While YOLOv9 showed underperformance in computational efficiency, it remains competitive with YOLOv10 and YOLO11 in terms of accuracy, thanks to its PGI in-tegration. This positions YOLOv9 as a viable choice for applications where precision is prioritized over speed.

In addition, YOLOv8 and YOLOv5u exhibited competitive results, surpassing YOLOv3u in accuracy, which is likely due to YOLOv3u's older architecture. However, their accuracy still fell significantly short compared to the newer models, such as YOLOv9, YOLOv10, and YOLO11. While YOLOv8 and YOLOv5u had faster processing times than YOLOv9, their overall performance remains inferior to that of the newer models.

5) Object Size and Rotation Detection: The YOLO algorithm is effective in detecting large and medium-sized objects, as evidenced by high accuracy in the Africa Wildlife and Traffic Signs datasets. However, it struggles with small object detection, probably due to its division of images into grids, making identifying small, low-resolution objects challenging. Adding to that, YOLO faces challenges when handling objects of different rotations due to the inability to enclose rotated objects, leading to sub-par results overall.

To handle rotated objects, models such as YOLO11 OBB [26] and YOLOv8 OBB [25] (Oriented Bounding Box) can be implemented. Keeping the same foundational architecture as the standard YOLOv8 and YOLO11, YOLOv8 OBB and YOLO11 OBB replace the standard bounding box prediction head with one that predicts the four corner points of a rotated rectangle, allowing for more accurate localization and representation of arbitrarily oriented objects.

6) The Rise of YOLO11 Over YOLOv8: Although YOLOv8 [25] has been the algorithm of choice for its versatility in tasks such as pose estimation, instance segmentation, and oriented object detection (OBB), YOLO11 [26] has now emerged as a more efficient and accurate alternative. With its ability to handle the same tasks while offering improved contextual understanding and better architectural modules, YOLO11 sets a new standard in performance, surpassing YOLOv8 in both speed and accuracy across various applications.

7) Dataset Size: The size of the dataset significantly influences the performance of YOLO models. For instance, large models did not perform optimally on the small African wildlife dataset compared to their results on the Traffic Signs and Ships and Vessels datasets due to being more prone to overfitting. Conversely, small models such as YOLOv9t and YOLOv9s performed significantly better on the Africa Wildlife dataset compared to their results on the other datasets, showcasing the effectiveness of small-scaled models when handling limited datasets.

8) Impact of Training Datasets: The performance of YOLO models is influenced by the training datasets used, as shown in Tables VI, VII, and VIII. Different datasets yield varying results and top performers, indicating that dataset complexity affects algorithm performance. This underscores the importance of using diverse datasets during benchmarking to obtain comprehensive results on the strengths and limitations of each model.

This discussion highlights the need for a balanced consideration of accuracy, speed, and model size when selecting YOLO models for specific applications. The consistent perfor-

mance of YOLO11 models across various metrics makes them highly recommended for versatile situations where accuracy and speed are essential. Meanwhile, YOLOv10 models can perform similarly while achieving faster processing times and with smaller model sizes. Additionally, YOLOv9 can deliver comparable results in terms of accuracy but sacrifices speed, making it suitable for applications where precision is prioritized over rapid processing.

V. CONCLUSION

This benchmark study thoroughly evaluates the performance of various YOLO algorithms. It pioneers a comprehensive comparison of YOLO11 against its predecessors, evaluating their performance across three diverse datasets: Traffic Signs, African Wildlife, and Ships and Vessels. The datasets were carefully selected to encompass a wide range of object properties, including varying object sizes, aspect ratios, and object densities. We showcase the strengths and weaknesses of each YOLO version and family by examining a wide range of metrics such as Precision, Recall, Mean Average Precision (mAP), Processing Time, GFLOPs count, and Model Size. Our study addresses the following key research questions:

- Which YOLO algorithm demonstrates superior performance across a comprehensive set of metrics?
- How do different YOLO versions perform on datasets with diverse object characteristics, such as size, aspect ratio, and density?
- What are the specific strengths and limitations of each YOLO version, and how can these insights inform the selection of the most suitable algorithm for various applications?

In particular, the YOLO11 family emerged as the most consistent, with YOLO11m striking an optimal balance between accuracy, efficiency, and model size. While YOLOv10 delivered slightly lower accuracy than YOLO11, it excelled in speed and efficiency, making it a strong choice for applications requiring efficiency and fast processing. Additionally, YOLOv9 performed well overall and particularly stood out in smaller datasets. These findings provide valuable insights for industry and academia, guiding the selection of the most suitable YOLO algorithms and informing future developments and enhancements. While the evaluated algorithms demonstrate promising performance, there is still room for refinement. Future research could focus on optimizing YOLOv10 to enhance its accuracy while preserving its speed and efficiency advantage. Additionally, continued advancements in architectural design may pave the way for even more groundbreaking YOLO algorithms. Our future work includes in-depth studies of the identified gaps in these algorithms, along with proposed improvements to demonstrate their potential impact on overall efficiency.

REFERENCES

[1] Oluibukun Ajayi, John Ashi, and BLESSED Guda. Performance evaluation yolo v5 model for automatic crop and weed classification on uav images. *Smart Agricultural Technology*, 5:100231, 04 2023.

[2] Bader Aldughayfiq, Farzeen Ashfaq, NZ Jhanji, and Mamoona Humayun. Yolo-based deep learning model for pressure ulcer detection and classification. In *Healthcare*, volume 11, page 1222. MDPI, 2023.

[3] Alaa Ali and Magdy A Bayoumi. Towards real-time dpm object detector for driver assistance. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3842–3846. IEEE, 2016.

[4] Isaiah Francis E Babila, Shawn Anthonie E Villasor, and Jennifer C Dela Cruz. Object detection for inventory stock counting using yolov5. In *2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA)*, pages 304–309. IEEE, 2022.

[5] Chetan Badgujar, Daniel Flippo, Sujith Gunturu, and Carolyn Baldwin. Tree trunk detection of eastern red cedar in rangeland environment with deep learning technique. *Croatian Journal of Forest Engineering*, 44, 06 2023.

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[7] Yining Cao, Chao Li, Yakun Peng, and Huiying Ru. Mcs-yolo: A multiscale object detection method for autonomous driving road environment recognition. *IEEE Access*, 11:22342–22354, 2023.

[8] Libo Cheng, Jia Li, Ping Duan, and Mingguo Wang. A small attentional yolo model for landslide detection from satellite remote sensing images. *Landslides*, 18(8):2751–2765, 2021.

[9] Yuan Dai, Weiming Liu, Haiyu Li, and Lan Liu. Efficient foreign object detection between psds and metro doors via deep neural networks. *IEEE Access*, PP:1–1, 03 2020.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[11] Sheshang Degadwala, Dhairya Vyas, Utsho Chakraborty, Abu Raihan Dider, and Haimanti Biswas. Yolo-v4 deep learning model for medical face mask detection. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pages 209–213. IEEE, 2021.

[12] Tausif Diwan, G Anirudh, and Jitendra V Tembhurne. Object detection using yolo: Challenges, architectural successors, datasets and applications. *multimedia Tools and Applications*, 82(6):9243–9275, 2023.

[13] Yunus Egi, Mortaza Hajyzadeh, and Engin Eyceyurt. Drone-computer communication based tomato generative organ counting model using yolo v5 and deep-sort. *Agriculture*, 12:1290, 08 2022.

[14] Loddo Fabio, Dario Piga, Michelucci Umberto, and El Ghazouali Safouane. Benchcloudvision: A benchmark analysis of deep learning approaches for cloud detection and segmentation in remote sensing imagery. *arXiv preprint arXiv:2402.13918*, 2024.

[15] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[16] Di Feng, Ali Harakeh, Steven L Waslander, and Klaus Dietmayer. A review and comparative study on probabilistic object detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):9961–9980, 2021.

[17] Rongli Gai, Na Chen, and Hai Yuan. A detection algorithm for cherry fruits based on the improved yolo-v4 model. *Neural Computing and Applications*, 35(19):13895–13906, 2023.

[18] Dweepna Garg, Parth Goel, Sharnil Pandya, Amit Ganatra, and Ketan Kotecha. A deep learning approach for face detection using yolo. In *2018 IEEE Punecon*, pages 1–4. IEEE, 2018.

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[20] Juan Guerrero-Ibáñez, Sherali Zeadally, and Juan Contreras-Castillo. Sensor technologies for intelligent transportation systems. *Sensors*, 18(4), 2018.

[21] Muhammad Hussain. Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, 11(7):677, 2023.

[22] Muhammad Hussain. Yolov1 to v8: Unveiling each variant–a comprehensive review of yolo. *IEEE Access*, 12:42816–42833, 2024.

[23] Rasheed Hussain and Sherali Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1275–1313, 2019.

[24] Glenn Jocher. Ultralytics yolov5, 2020.

[25] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[26] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024.

[27] Chang Ho Kang and Sun Young Kim. Real-time object detection and segmentation technology: an analysis of the yolo algorithm. *JMST Advances*, 5(2):69–76, 2023.

[28] Nyoman Karna, Made Adi Paramartha Putra, Syifa Rachmawati, Mideth Abisado, and Gabriel Sampedro. Toward accurate fused deposition modeling 3d printer fault detection using improved yolov8 with hyperparameter optimization. *IEEE Access*, PP:1–1, 01 2023.

[29] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022.

[30] Guofa Li, Zefeng Ji, Xingda Qu, Rui Zhou, and Dongpu Cao. Cross-domain object detection for autonomous driving: A stepwise domain adaptive yolo approach. *IEEE Transactions on Intelligent Vehicles*, 7(3):603–615, 2022.

[31] Min Li, Zhijie Zhang, Liping Lei, Xiaofan Wang, and Xudong Guo. Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of faster r-cnn, yolo v3 and ssd. *Sensors*, 20(17):4938, 2020.

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[33] Martina Lippi, Niccolò Bonucci, Renzo Fabrizio Carpio, Mario Contarini, Stefano Speranza, and Andrea Gasparri. A yolo-based pest detection system for precision agriculture. In *2021 29th Mediterranean Conference on Control and Automation (MED)*, pages 342–347. IEEE, 2021.

[34] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128:261–318, 2020.

[35] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37. Springer International Publishing, 2016.

[36] Jueal Mia, Hasan Imam Bijoy, Shoreef Uddin, and Dewan Mamun Raza. Real-time herb leaves localization and classification using yolo. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2021.

[37] Hamzeh Mirhaji, Mohsen Soleymani, Abbas Asakereh, and Saman Abdanan Mehdizadeh. Fruit detection and load estimation of an orange orchard using the yolo models through simple approaches in different imaging and illumination conditions. *Computers and Electronics in Agriculture*, 191:106533, 2021.

[38] Miand Mostafa and Milad Ghantous. A yolo based approach for traffic light recognition for adas systems. In *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pages 225–229. IEEE, 2022.

[39] Huy Hoang Nguyen, Thi Nhung Ta, Ngoc Cuong Nguyen, Hung Manh Pham, Duc Minh Nguyen, et al. Yolo based real-time human detection for smart video surveillance at the edge. In *2020 IEEE eighth international conference on communications and electronics (ICCE)*, pages 439–444. IEEE, 2021.

[40] Radu Oprea. Traffic signs detection europe dataset. https://universe.roboflow.com/radu-oprea-r4xnm/traffic-signs-detection-europe, feb 2024. visited on 2024-07-12.

[41] Rafael Padilla, Sergio L Netto, and Eduardo AB Da Silva. A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE, 2020.

[42] Govind S Patel, Ashish A Desai, Yogesh Y Kamble, Ganesh V Pujari, Priyanka A Chougule, and Varsha A Jujare. Identification and separation of medicine through robot using yolo and cnn algorithms for healthcare. In *2023 International Conference on Artificial Intelligence for Innovations in Healthcare Industries (ICAIIHI)*, volume 1, pages 1–5. IEEE, 2023.

[43] Paul Paul Tsoi. YOLO11: The cutting-edge evolution in object detection — a brief review of the latest in the yolo series. https://medium.com, October 2024. Accessed: 2024-10-17.

[44] Minh-Tan Pham, Luc Courtrai, Chloé Friguet, Sébastien Lefèvre, and Alexandre Baussard. Yolo-fine: One-stage detector of small objects under various backgrounds in remote sensing images. *Remote Sensing*, 12(15):2501, 2020.

[45] Francesco Prinzi, Marco Insalaco, Alessia Orlando, Salvatore Gaglio, and Salvatore Vitabile. A yolo-based model for breast cancer detection in mammograms. *Cognitive Computation*, 16(1):107–120, 2024.

[46] Sovit Rath. Yolov8 ultralytics: State-of-the-art yolo models. *LearnOpenCV–Learn OpenCV, PyTorch, Keras, TensorflowWith Examples and Tutorials*, 2023.

[47] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[48] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[49] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[50] Arunabha M Roy, Jayabrata Bhaduri, Teerath Kumar, and Kislay Raj. Wildect-yolo: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection. *Ecological Informatics*, 75:101919, 2023.

[51] Arunabha Mohan Roy, Jayabrata Bhaduri, Teerath Kumar, and Kislay Raj. A computer vision-based object localization model for endangered wildlife detection. *Ecological Economics, Forthcoming*, 2022.

[52] SIDDHARTH SAH. Ships/vessels in aerial images. https://www.kaggle.com/datasets/siddharthkumarsah/ships-in-aerial-images/data, july 2023. visited on 2024-07-12.

[53] Ranjan Sapkota, Rizwan Qureshi, Marco Flores Calero, Muhammad Hussain, Chetan Badjugar, Upesh Nepal, Alwin Poulose, Peter Zeno, Uday Bhanu Prakash Vaddevolu, Hong Yan, et al. Yolov10 to its genesis: A decadal and comprehensive review of the you only look once series. *arXiv preprint arXiv:2406.19407*, 2024.

[54] Abhishek Sarda, Shubhra Dixit, and Anupama Bhan. Object detection for autonomous driving using yolo [you only look once] algorithm. In *2021 Third international conference on intelligent communication technologies and virtual mobile networks (ICICV)*, pages 1370–1374. IEEE, 2021.

[55] Maged Shoman, Gabriel Lanzaro, Tarek Sayed, and Suliman Gargoum. Autonomous vehicle-pedestrian interaction modeling platform: A case study in four major cities. *Journal of Transportation Engineering Part A Systems*, 06 2024.

[56] Maged Shoman, Dongdong Wang, Armstrong Aboah, and Mohamed Abdel-Aty. Enhancing traffic safety with parallel dense video captioning for end-to-end event analysis, 2024.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[58] Mupparaju Sohan, Thotakura Sai Ram, Rami Reddy, and Ch Venkata. A review on yolov8 and its advancements. In *International Conference on Data Intelligence and Cognitive Informatics*, pages 529–545. Springer, 2024.

[59] suranaree university of technology. africa wild life dataset. https://universe.roboflow.com/suranaree-university-of-technology-wqhl6/africa-wild-life, feb 2023. visited on 2024-07-12.

[60] Ultralytics. YOLOv5: A state-of-the-art real-time object detection system. https://docs.ultralytics.com, 2021. Accessed: insert date here.

[61] Amir Ulykbek, Azamat Serek, and Magzhan Zhailau. A comprehensive review of object detection in yolo: Evolution, variants, and applications.

[62] NL Vidya, M Meghana, P Ravi, and Nithin Kumar. Virtual fencing using yolo framework in agriculture field. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 441–446. IEEE, 2021.

[63] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[64] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024.

[65] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[66] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616*, 2024.

[67] Yifan Wang, Lin Yang, Hong Chen, Aamir Hussain, Congcong Ma, and Malek Al-gabri. Mushroom-yolo: A deep learning algorithm for mushroom growth recognition based on improved yolov5 in agriculture 4.0. In *2022 IEEE 20th International Conference on Industrial Informatics (INDIN)*, pages 239–244. IEEE, 2022.

[68] Tingting Zhao, Xiaoli Yi, Zhiyong Zeng, and Tao Feng. Mobilenet-yolo based wildlife detection model: A case study in yunnan tongbiguan nature reserve, china. *Journal of Intelligent & Fuzzy Systems*, 41(1):2171–2181, 2021.

[69] Yifei Zheng and Hongling Zhang. Video analysis in sports by lightweight object detection network under the background of sports

industry development. *Computational Intelligence and Neuroscience*, 2022:1–10, 08 2022.

[70] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.