

# YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions

Nidhal Jegham<sup>a,b</sup>, Chan Young Koh<sup>b</sup>, Marwan Abdelatti<sup>b,c</sup>, Abdeltawab Hendawi<sup>b,\*</sup>

<sup>a</sup>Tunis Business School, University of Tunis, Tunis, Tunisia

<sup>b</sup>Computer Science Department, University of Rhode Island, Kingston, RI, USA

<sup>c</sup>Department of Computer Science, Providence College, Providence, RI, USA

---

## Abstract

This study presents a comprehensive benchmark analysis of various YOLO (You Only Look Once) algorithms. It represents the first comprehensive experimental evaluation of YOLOv3 to the latest version, YOLOv12, on various object detection challenges. The challenges considered include varying object sizes, diverse aspect ratios, and 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 maintaining a remarkable balance of accuracy and efficiency. However, YOLOv12 delivered underwhelming results, with its complex architecture introducing computational overhead without significant performance gains. 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.

**Keywords:** YOLOv12, YOLO11, Benchmark, Evaluation, Architectural Review

---

## 1. Introduction

Object detection is an essential component of computer vision systems, enabling automated identification and localization of objects within images or video frames [44]. Its applications span from autonomous driving and robotics [20, 7, 26, 69] to inventory management, video surveillance, and sports analysis [6, 30, 68, 87].

---

\*Corresponding author

Email addresses: jeghamnidhal7@gmail.com, nidhal.jegham@uri.edu (Nidhal Jegham), ckoh04@uri.edu (Chan Young Koh), mabdelrazik@uri.edu, mabdelat@providence.edu (Marwan Abdelatti), hendawi@uri.edu (Abdeltawab Hendawi)

Preprint submitted to Image and Vision Computing

March 19, 2025

Over the years, object detection has developed significantly. Initially, traditional methods such as the Viola-Jones algorithm [78] and the Deformable Part-based Model (DPM) [19] used handcrafted features and were effective for applications such as face detection [78], pedestrian detection [14], and video surveillance [4]. 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 [41] and SSD (Single Shot MultiBox Detector) [45] perform detection in a single pass, balancing speed and accuracy. In contrast, two-stage methods, such as Region-based Convolutional Neural Networks (R-CNN) [24], generate region proposals and then perform classification, offering high precision but being computationally intensive.

YOLO (You Only Look Once) is a pioneering one-stage object detection framework that predicts bounding boxes and class probabilities in a single evaluation, enabling real-time performance. First introduced by Redmon et al. in 2015 [59], YOLOv1 redefined object detection with its efficiency. YOLOv2 [60] improved upon this by integrating Darknet-19, batch normalization, and data augmentation techniques inspired by the VGG architecture [70], leading to better generalization. YOLOv3 [61] adopted Darknet-53, a deeper network with enhanced feature extraction, and incorporated a Feature Pyramid Network (FPN)-inspired design for multi-scale detection, significantly improving accuracy for objects of varying sizes. Subsequent iterations diversified under different development groups. YOLOv4 [10] introduced Spatial Pyramid Pooling (SPP) for multi-scale feature fusion and the Path Aggregation Network (PAN) to refine feature integration. YOLOv5 [74] marked a shift from the Darknet framework to PyTorch, increasing accessibility and efficiency through strided convolution layers and Spatial Pyramid Pooling Fast (SPPF) layers. YOLOv6 [38] implemented RepVGG for simplified inference and CSPStackRep blocks for improved accuracy. YOLOv7 [81] introduced Extended Efficient Layer Aggregation Networks (E-ELAN) to optimize information flow and enhance detection performance.

More recent versions further pushed the boundaries of efficiency and accuracy. YOLOv8 [71], released by Ultralytics, introduced scalable models tailored for various hardware constraints and expanded capabilities to tasks like semantic segmentation, pose estimation, and oriented bounding box (OBB) detection. YOLOv9 [82] introduced Programmable Gradient Information (PGI) for optimized gradient flow and Generalized Efficient Layer Aggregation Networks (GELAN) for enhanced feature fusion. YOLOv10 [79] eliminated Non-Maximum Suppression (NMS) using a dual assignment strategy, while also introducing lightweight classification heads and spatial-channel decoupled downsampling for increased efficiency. YOLOv11 [33] retained YOLOv8's multi-task versatility while improving efficiency with the C3k2 block and introducing the C2PSA module for better spatial attention, particularly benefiting small and overlapping object detection. The latest iteration, YOLOv12, builds on prior advancements by incorporating an attention-centric approach with Area Attention (A2), which enhances feature aggregation while maintaining real-time performance. It also introduces Residual Efficient Layer Aggregation Networks (R-ELAN) to improve optimization stability and convergence [73]. These innovations solidify YOLO as a leading framework in real-time object detection, continuously improving accuracy, speed, and adaptability across various tasks.

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) [59], video surveillance [49], face detection [50], and many more [23]. For instance, YOLO plays a crucial role in the agriculture field by being implemented in numerous applications such as crop classification [2] [22],

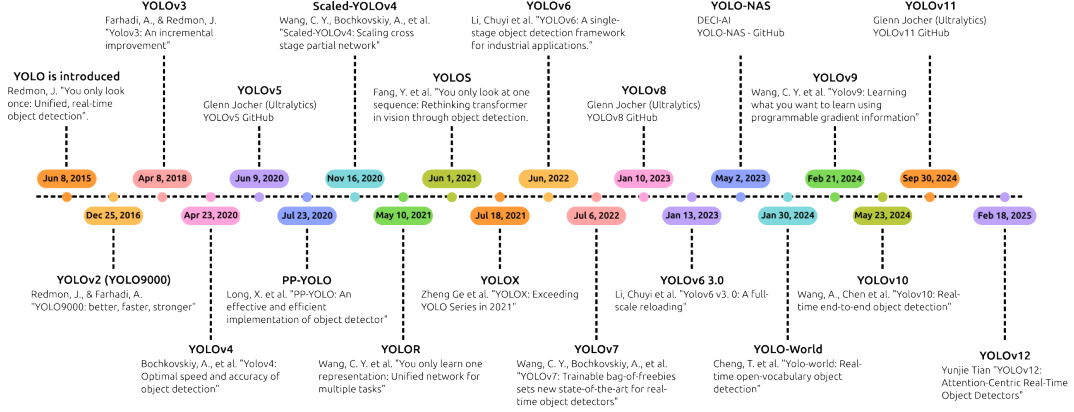


Figure 1: Evolution of YOLO Algorithms throughout the years

pest detection [42], automated farming [85, 48], and virtual fencing [77]. Moreover, YOLO has been utilized on numerous occasions in the field of healthcare, such as cancer detection [57], ulcer detection [3], medicine classification [47, 53], and health protocols enforcement [15].

In recent years, Ultralytics has played a crucial role in the advancement of YOLO by maintaining, improving, and making these models more accessible [58]. Notably, Ultralytics has streamlined the process of fine-tuning and customizing YOLO models, a considerably more complex task 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. 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 helped YOLO models to be more accessible and adaptable for a wide range of applications.

This paper presents 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 YOLOv12, 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 focuses on numerous 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, models, experimental setup, hyperparameters, and evaluation metrics used. Section 4 presents the experimental results, comparative analysis, and discussion, highlighting key discoveries, trends, architectural review, and real-life applications. Finally, Section 7 concludes with insights drawn from the study.

## 2. 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 [18] 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 with cloud segmentation using a few spectral bands, including RGB and RGBN-IR combinations.

Highlighting advancements in UAV-based object detection, the study [36] introduces SFFEF-YOLO to tackle challenges posed by varying target scales and small object prevalence in aerial imagery. By replacing the large prediction head with a tiny one, integrating a Fine-Grained Information Extraction Module (FIEM), and refining multi-scale fusion with the Multi-Scale Feature Fusion Module (MFFM), the model achieves an increase of 9.9% on VisDrone2019-DET and 3.6% on UAVDT in terms of mAP50 over YOLOv8. However, using an older YOLO version may limit its full potential. Incorporating newer YOLO models such as YOLO11 could further enhance accuracy, efficiency, and generalizability, a gap this paper addresses through broader benchmarks.

The authors of [28] review the evolution of the YOLO variants from version 1 to version 8, examining their internal architecture, key innovations, and benchmarked performance metrics. 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 [16] 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 and applications utilizing two-stage detectors. However, not including newer YOLO models limits its comprehensiveness, leaving a gap in understanding the advancements and improvements introduced in more recent versions.

The authors of [66] 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 study [37] presents a lightweight rotational object detection algorithm to address challenges in remote sensing and surveillance, particularly variations in object size and orientation. By integrating an angle prediction branch and the Circular Smooth Label (CSL) method, along with a Channel and Spatial Attention (CSA) module, the model enhances feature extraction and detection accuracy. Achieving 57.86 mAP50 on the DOTA v2 dataset while maintaining a low computational footprint, it demonstrates efficiency for real-time deployment. However, its reliance on YOLOv5 limits potential performance gains. However, maintaining the backbone architecture of YOLOv5 can lead to limitations in terms of both computational efficiency and accuracy. Therefore, incorporating newer YOLO architectures could further improve the scope of this paper,

The authors in the work in [34] 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 evaluating the most recent models, YOLOv9, YOLOv10, and YOLO11.

This paper makes several key contributions: (i) It pioneers a comprehensive comparison of YOLOv12 against its predecessors across their scaled variants from nano- to extra-large; (ii) It provides deep insights into the architectural advancements of YOLO by analyzing key structural developments across versions. (iii) It evaluates YOLO models using three diverse datasets, reflecting various object properties and real-world applications, including Smart Cities, Satellite Imaging, and Wildlife Conservation. (iv) The performance evaluation extends beyond mAP and FPS, incorporating critical metrics such as Precision, Recall, Preprocessing Time, Inference Time, Postprocessing Time, GFLOPs, and model size. (v) These comprehensive metrics provide valuable insights for selecting the most optimal YOLO models, benefiting industry professionals and academics. (vi) It offers specific use case recommendations, identifying the most suitable YOLO models for different scenarios and environments, such as resource-constrained deployments, real-time applications, and the detection of small or overlapping objects.

### 3. Benchmark Setup

#### 3.1. Datasets

This study conducts in-depth benchmark research and assesses 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 [59, 67, 39, 11], satellite imagery [40, 12, 56], and wildlife conservation [62, 86, 63]. 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 to ensure a comprehensive evaluation.

##### 3.1.1. Traffic Signs Dataset

The Traffic Signs dataset by Radu Oprea is an open-source dataset on Kaggle containing around 55 classes across 3253 training and 1128 validation images of traffic signs in various

sizes and environments [51]. All images in the dataset are initially sized 640×640, with no labels for False Positives detection. To balance the classes, undersampling techniques were applied, as shown in Figure 2. After preprocessing the dataset by removing classes with less than 50 observation counts, 24 classes remained, with 3233 images split into 70% training, 20% validation, and 10% testing, with no data augmentation techniques applied. This dataset is vital for applications in autonomous driving, traffic management, road safety, and intelligent transportation systems. Additionally, it presents challenges due to the varying sizes of target objects and the similarities in patterns across different classes, which complicate the detection process.



Figure 2: Classes Distribution of the Traffic Signs Dataset

### 3.1.2. Africa Wild Life Dataset

The Africa Wildlife dataset is an open-source Kaggle dataset by Bianca Ferreira, designed for real-time animal detection in nature reserves [72]. It features four common African animal classes: Buffalo, elephant, rhino, and zebra. Each class is represented by at least 376 images, which were collected via Google image searches without any data augmentation techniques applied. The dataset is split into 70% training, 20% validation, and 10% testing, with all images manually labeled in the YOLO format, as shown in Figure 3. The dataset presents challenges such as varying aspect ratios, with each image containing at least one instance of the specified animal class and potentially multiple occurrences of other classes. Furthermore, the overlap of target objects increases the difficulty of detection. This dataset is crucial for applications in wildlife conservation, anti-poaching efforts, biodiversity monitoring, and ecological research.

### 3.1.3. Ships/Vessels Dataset

The Ships/Vessels 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 [65]. Each image has been manually annotated with bounding boxes in

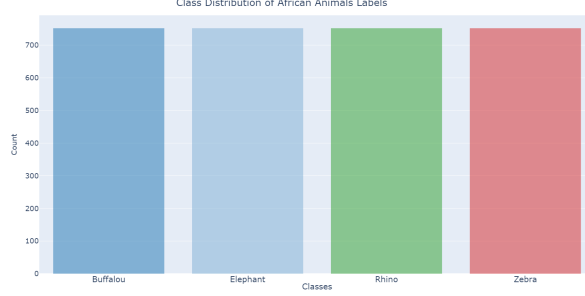


Figure 3: Classes Distribution of the Africa Wildlife Dataset

the YOLO format, ensuring precise and efficient detection of ships. This dataset features a single class, "ship," and is divided into 70% training, 20% validation, and 10% testing. Notably, no data augmentation techniques were applied during the experiment, which ensures that the model's performance is evaluated on the raw, unaltered dataset. However, the relatively small size of the target objects, varying aspect ratios, 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.

### 3.2. Models

#### 3.2.1. Ultralytics vs. Original YOLO

In this subsection, we will conduct a comparative analysis between the models provided by Ultralytics and their original counterparts on the Traffic Signs dataset provided by Radu Oprea [51] using the same hyperparameters in Table 3. The objective is to highlight the differences between Ultralytics models and the original versions, which justifies the exclusion of YOLOv4 [10], YOLOv6 [38], and YOLOv7 [81] 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.

**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 1, 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.

**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 2, 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

Table 1: Ultralytics-supported library tasks and models

<b>YOLO Version</b>	<b>Inference</b>	<b>Validation</b>	<b>Training</b>
YOLOv1	No	No	No
YOLOv2	No	No	No
<b>YOLOv3u</b>	Yes	Yes	Yes
YOLOv4	No	No	No
<b>YOLOv5u</b>	Yes	Yes	Yes
YOLOv6	Yes	Yes	Yes
YOLOv7	No	No	No
<b>YOLOv8</b>	Yes	Yes	Yes
<b>YOLOv9</b>	Yes	Yes	Yes
<b>YOLOv10</b>	Yes	Yes	Yes
<b>YOLO11</b>	Yes	Yes	Yes
<b>YOLOv12</b>	Yes	Yes	Yes

<b>Version</b>	<b>Scaled Version</b>
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)
YOLOv12	YOLO11n (nano)
	YOLOv12s (small)
	YOLOv12m (medium)
	YOLOv12l (large)
	YOLOv12x (extra-large)

Figure 4: YOLO versions and scaled versions



Table 2: Ultralytics and original YOLO performance comparison

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

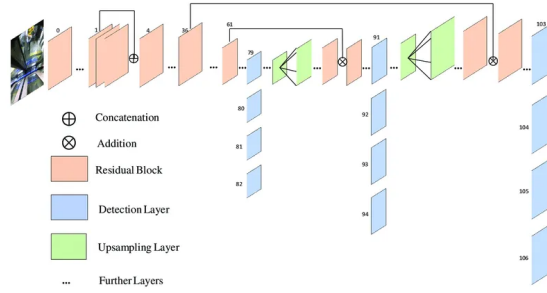


Figure 5: YOLOv3 architecture showcasing the residual blocks and the upsampling layers [13]

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.

In total, 33 models from 7 different YOLO versions were trained on three different datasets, as seen in Figure 4. Each YOLO version includes several scaled models (e.g., YOLOv5u, YOLOv5un, YOLOv5us, and YOLOv5ux), with the suffixes denoting model size and complexity, such as "n" for nano, "s" for small, "m" for medium, "l" for large, "x" for extra-large, "t" for tiny, "c" for compact, "b" for balanced, and "e" for extended. These models offer a variety of trade-offs between accuracy and inference speed, as discussed in the following sections.

### 3.2.2. YOLOv3u

YOLOv3 enhances localization and detection efficiency, particularly for small objects, using the Darknet-53 framework, which offers double the speed of ResNet-152 [61]. It integrates Feature Pyramid Network (FPN) elements, including residual blocks, skip connections, and up-sampling, to improve multi-scale detection, as illustrated in Figure 5. The model generates feature maps at three scales (down-sampling at factors of 32, 16, and 8) for detecting objects of varying sizes. However, YOLOv3 struggles with medium and large objects, leading Ultralytics to introduce YOLOv3u, which adopts an anchor-free detection method, later used in YOLOv8, improving both accuracy and speed.

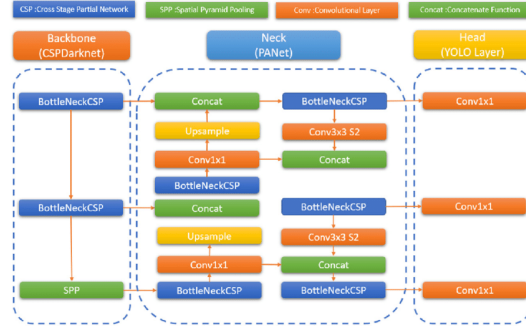


Figure 6: Detailed architecture of YOLOv5 including the CSPDarknet Backbone, PANet Neck, and YOLO Layer Head [17]

### 3.2.3. YOLOv5u

YOLOv5, developed by Glenn Jocher, transitions from Darknet to PyTorch, incorporating CSPDarknet as its backbone for improved feature extraction and reduced computational cost [74] [31]. It features Cross-Stage Partial (CSP) connections, a strided convolution layer, and the Spatial Pyramid Pooling Fast (SPPF) module, which enhances multi-scale feature representation, as illustrated in Figure 6. YOLOv5 also integrates data augmentation techniques (Mosaic, copy-paste, MixUp, HSV augmentation) to improve robustness. Available in five scaled variants, it continues to evolve with YOLOv5u, which introduces anchor-free detection for enhanced accuracy and efficiency, particularly on complex objects of varying sizes.

### 3.2.4. YOLOv8

Ultralytics has introduced YOLOv8, a significant evolution in the YOLO series, with five scaled versions [71] [32]. Alongside object detection, YOLOv8 also provides various applications such as image classification, pose estimation, instance segmentation, and OBB. 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 7. 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.

### 3.2.5. YOLOv9

YOLOv9, developed by Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao, enhances object detection using the Information Bottleneck Principle and Reversible Functions, ensuring efficient gradient propagation and improved model convergence [82]. Reversible functions enable lossless information retention, benefiting lightweight models prone to under-parameterization. A key advancement, Programmable Gradient Information (PGI), dynamically adjusts gradient flow during training, prioritizing informative gradients to prevent feature loss. Additionally, YOLOv9 integrates Gradient Enhanced Lightweight Architecture Network (GELAN), optimizing computational pathways for better parameter utilization, high-speed inference, and accurate detection, as illustrated in Figure 8. With five scalable versions, YOLOv9

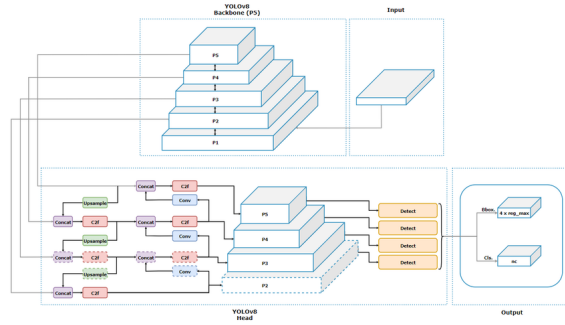


Figure 7: Detailed architecture of YOLOv8 showcasing the backbone’s multiple convolutional layers to extract hierarchical features, the Feature Pyramid Network (FPN) enhances detection at different scales. The network head performs final predictions, incorporating convolutional blocks and upsample blocks to refine features [35]

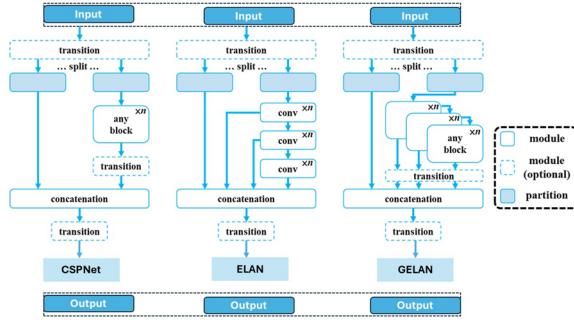


Figure 8: YOLOv9 architecture featuring CSPNet, ELAN, and GELAN modules. CSPNet optimizes gradient flow and reduces computational complexity via feature map partitioning. ELAN enhances learning efficiency by linearly aggregating features, and GELAN extends this concept by integrating features from various depths and pathways [82]

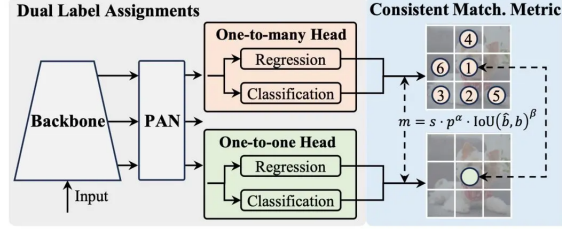


Figure 9: YOLOv10 architecture showcasing the dual label assignment strategy for improving accuracy. The PAN layer enhances feature representation alongside one-to-many head for regression and classification tasks and one-to-one head for precise localization [79]

balances efficiency and feature retention, making it highly adaptable for various real-world applications.

### 3.2.6. YOLOv10

YOLOv10, developed by Tsinghua University researchers, introduces key innovations, enhancing gradient flow and computational efficiency through an improved Cross Stage Partial Network (CSPNet) backbone [79]. The architecture, illustrated in Figure 9, consists of three main components: the backbone, the neck, and the detection head. The Path Aggregation Network (PAN) neck improves multi-scale feature fusion, allowing for better detection of objects at different sizes. The One-to-Many Head generates multiple predictions per object during training, enhancing learning accuracy.

For inference, YOLOv10 replaces Non-Maximum Suppression (NMS) with a One-to-One Head, producing a single best prediction per object, reducing latency and post-processing time. Additionally, NMS-Free Training leverages dual assignments, optimizing both speed and accuracy. Other enhancements include lightweight classification heads, spatial-channel decoupled downsampling, large-kernel convolutions, and partial self-attention modules, all improving efficiency without significantly increasing computational costs. YOLOv10 is available in five scaled versions, from nano to extra-large, catering to diverse use cases.

### 3.2.7. YOLO11

YOLO11 [33] is one of the latest additions to 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 OBB.

Key improvements in YOLO11 include the introduction of the Cross-Stage Partial with Self-Attention (C2PSA) module, as seen in Figure 10, 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.

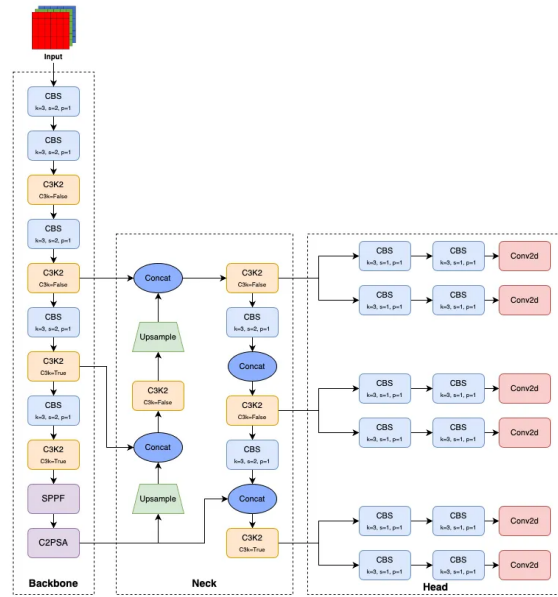


Figure 10: YOLO11 architecture showcasing the new C3k2 blocks and the C2PSA module [33] [54]

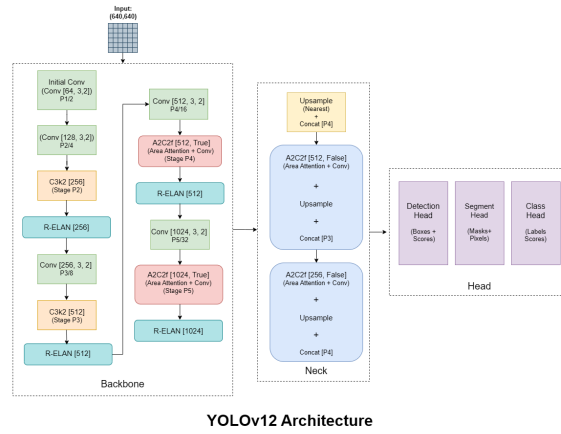


Figure 11: YOLOv12 architecture showcasing the new Area Attention (A2) module and Residual Efficient Layer Aggregation Networks (R-ELAN)

Table 3: 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
Augmentation Techniques	None
Data Split	(70, 20, 10)

### 3.2.8. YOLOv12

YOLOv12 [73] is the latest evolution in the YOLO series, introducing an attention-centric design that significantly enhances both speed and accuracy. It continues the trend of offering five scalable models (Nano to Extra Large), making it adaptable for a wide range of applications such as object detection, instance segmentation, and OBB. As seen in Figure 11, YOLOv12 introduces the Area Attention (A2) module, which maintains a large receptive field while drastically reducing computational complexity, allowing the model to enhance speed without compromising accuracy. Additionally, it features Residual Efficient Layer Aggregation Networks (R-ELAN), which improve training stability and model convergence through block-level residual design and optimized feature aggregation.

Moreover, YOLOv12 uses FlashAttention to minimize memory access overhead, closing the speed gap with CNNs. It adjusts the MLP ratio from 4 to 1.2, enhancing runtime efficiency, and removes positional encodings for a cleaner, faster model without losing detection accuracy. YOLOv12 also reduces complexity by using a single R-ELAN block in the last stage of the backbone instead of stacking three attention/CNN blocks. It replaces linear layers with convolutional layers and batch normalization, maximizing computational efficiency and achieving state-of-the-art latency-accuracy trade-offs.

### 3.3. Hardware and Software Setup

The experiments were conducted with Python 3.12, Ubuntu 22.04, CUDA 12.4, and cuDNN 8.9.7 for GPU acceleration. Ultralytics 8.2.55 was used for model training, while WandB 0.17.4 was utilized for tracking experiments. To ensure a fair comparison, similar hyperparameters were used across all models, as outlined in Table 3. The experiments were carried out using 2 NVIDIA RTX 4090 GPUs, each with 16,384 CUDA cores.

### 3.4. Metrics

This study evaluates YOLO models based on accuracy, computational efficiency, and size. Accuracy metrics include Precision, Recall, mAP50, and mAP50-95. Precision measures the ratio of correctly predicted observations to total predictions, indicating False Positives, while Recall highlights False Negatives by comparing correct predictions to actual observations [52]. mAP50 calculates Mean Average Precision at an IoU threshold of 0.50, whereas mAP50-95 extends this across thresholds from 0.50 to 0.95 with a 0.05 step size [88].

Computational efficiency is assessed using Preprocessing Time, Inference Time, and Postprocessing Time. Preprocessing Time refers to data preparation, Inference Time measures prediction generation, and Postprocessing Time converts raw outputs into final results. Additionally, GFLOPs indicate computational power, while model size reflects storage requirements and parameter count. These metrics enable a comprehensive performance comparison, ensuring a robust benchmark for real-world applications.

## 4. Benchmark Results

### 4.1. Results

#### 4.1.1. 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	282.4	207.86
YOLOv3u tiny	0.845	0.667	0.772	0.682	1.4	0.7	0.3	2.4	19	24.44
YOLOv5un	0.805	0.679	0.749	0.665	0.6	6.6	0.4	7.6	7.1	5.65
YOLOv5us	0.85	0.777	0.827	0.744	<b>0.5</b>	7.8	0.4	8.7	23.9	18.58
YOLOv5um	0.849	0.701	0.83	0.744	1.1	9.5	0.4	11	64.1	50.54
YOLOv5ul	0.831	0.836	0.886	<b>0.799</b>	0.6	9.7	0.4	10.7	134.9	106.85
YOLOv5ux	0.863	0.795	0.867	0.777	1.1	9.8	0.4	11.3	246.3	195.2
YOLOv8n	0.749	0.688	0.777	0.689	0.6	6.8	0.4	7.8	8.1	6.55
YOLOv8s	0.766	0.788	0.806	0.718	0.6	7.8	0.4	8.8	28.6	22.59
YOLOv8m	0.838	0.805	0.845	0.763	1.6	9.1	0.4	11.1	78.9	52.12
YOLOv8l	0.771	0.789	0.853	0.767	0.6	9.2	0.4	10.2	165	87.77
YOLOv8x	<b>0.902</b>	0.744	0.874	0.78	0.6	9.4	0.4	10.4	257.7	136.9
YOLOv9t	0.792	0.748	0.812	0.731	<b>0.5</b>	10	0.4	10.9	<b>7.7</b>	<b>4.93</b>
YOLOv9s	0.763	0.81	0.828	0.75	0.6	11.1	0.4	12.1	26.8	15.33
YOLOv9m	0.864	0.796	0.864	0.784	1	12.1	0.4	13.5	76.7	40.98
YOLOv9c	0.827	0.807	0.852	0.769	1.3	11.6	0.4	13.3	102.6	51.8
YOLOv9e	0.819	0.824	0.854	0.764	0.8	16.1	0.4	17.3	189.4	117.5
YOLOv10n	0.722	0.602	0.722	0.64	1	0.8	<b>0.2</b>	<b>2</b>	8.3	5.59
YOLOv10s	0.823	0.742	0.834	0.744	1.2	1.1	0.2	2.5	24.7	15.9
YOLOv10m	0.834	0.843	0.88	0.781	1.2	2.4	0.2	3.8	63.8	32.1
YOLOv10b	0.836	0.764	0.859	0.765	1	3.1	0.2	4.3	98.4	39.7
YOLOv10l	0.873	0.807	0.866	0.771	1.1	3.8	0.2	5.1	126.8	50
YOLOv10x	0.773	<b>0.854</b>	0.88	0.787	1	6.3	0.2	7.5	170.4	61.4
YOLO11n	0.768	0.695	0.757	0.668	1.2	<b>0.6</b>	0.4	2.2	<b>6.4</b>	<b>5.35</b>
YOLO11s	0.819	0.758	0.838	0.742	1.2	1	0.4	2.6	21.4	18.4
YOLO11m	0.898	0.826	<b>0.893</b>	0.795	1.2	2.4	0.4	4	67.9	38.8
YOLO11l	0.862	0.839	0.889	0.794	1.2	3	0.4	4.6	86.8	49
YOLO11x	0.819	0.816	0.885	0.784	0.9	6.1	0.4	7.4	194.8	109
YOLOv12n	0.883	0.684	0.802	0.708	1.2	4.6	0.4	6.2	<b>6.4</b>	<b>5.34</b>
YOLOv12s	0.761	0.839	0.877	0.78	1.6	4.3	0.5	6.4	21.4	18.1
YOLOv12m	0.868	0.811	0.89	0.79	0.9	4.9	0.4	6.2	67.3	39
YOLOv12l	0.826	0.836	0.877	0.781	0.9	5.5	0.4	6.8	89.6	51.2
YOLOv12x	0.832	0.849	0.889	0.793	1	9.5	0.5	11	198.9	114

Figure 12: Evaluation results for the traffic signs dataset.

Figure 12 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.

**Accuracy:** As illustrated in Figure 13, 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.

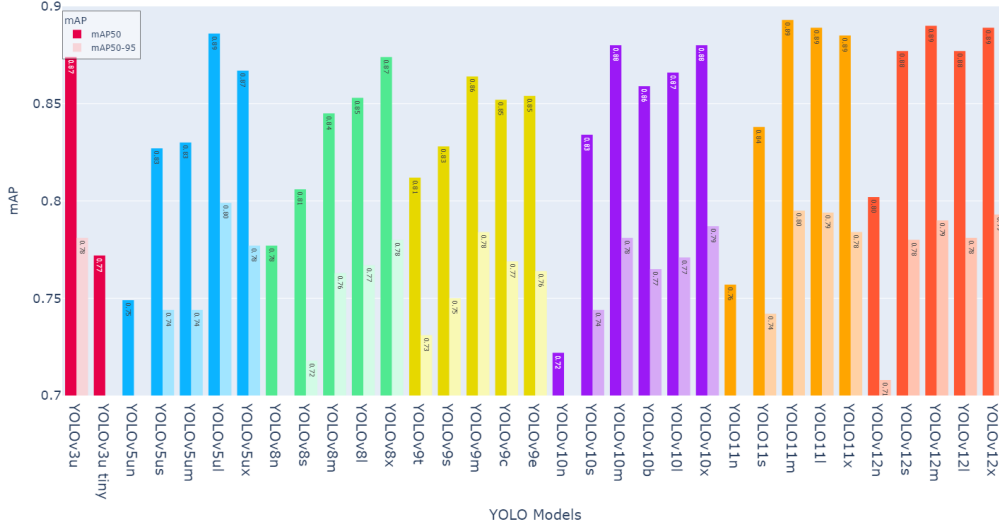


Figure 13: mAP50 and mAP50-95 YOLO results on traffic signs dataset. Each model is represented by two bars: the left bar shows the mAP50 score, while the right bar represents the mAP50-95 score

**Computational Efficiency:** Regarding 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 Figure 15. 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 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.

**Overall Performance:** When evaluating overall performance, which includes accuracy, size, and model efficiency, YOLO11m emerges as a consistently top-performing model as detailed in Figures 12, 13, 14, and 15. This is followed by YOLO11l and YOLOv10m showcasing the robustness of these models. On the contrary, YOLOv9e demonstrates poor performance overall in terms of accuracy, speed, GFLOPs, and model size. Additionally, YOLOv5um, YOLOv8m, and YOLOv8s performed suboptimally, further indicating the inconsistency in performance among earlier model families. Notably, the YOLO11 and YOLOv12 families significantly outperform other YOLO families regarding accuracy and computational efficiency. Their models consistently surpass counterparts from the YOLOv3u, YOLOv5u, YOLOv8, YOLOv9, and YOLOv10 families, demonstrating their ability to balance precision and speed effectively in the detection of objects with varying sizes.



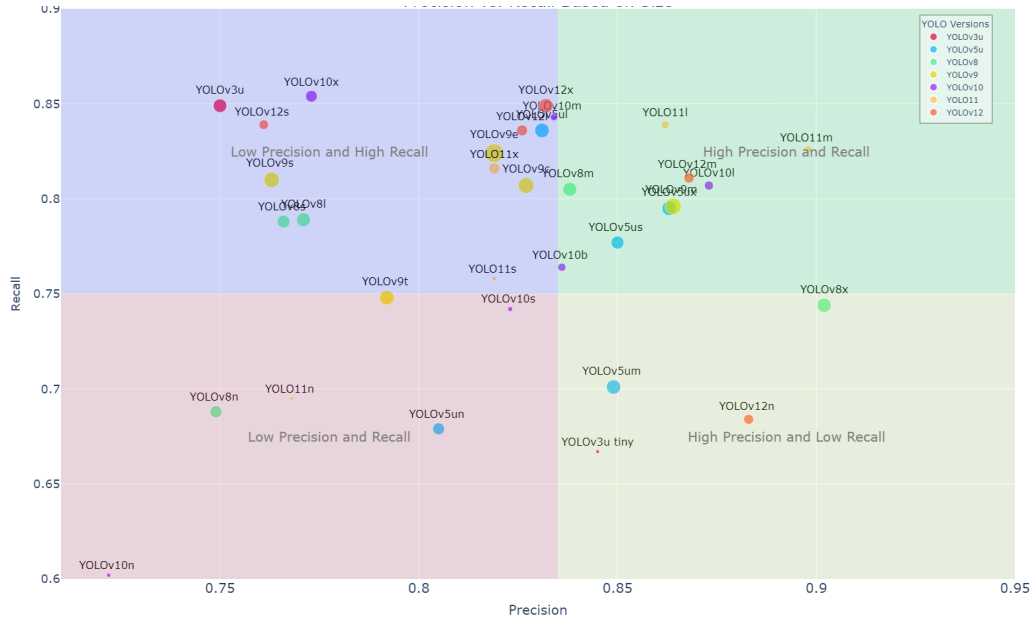


Figure 14: Precision vs. Recall based on size on traffic signs dataset, with larger circles indicating larger model sizes

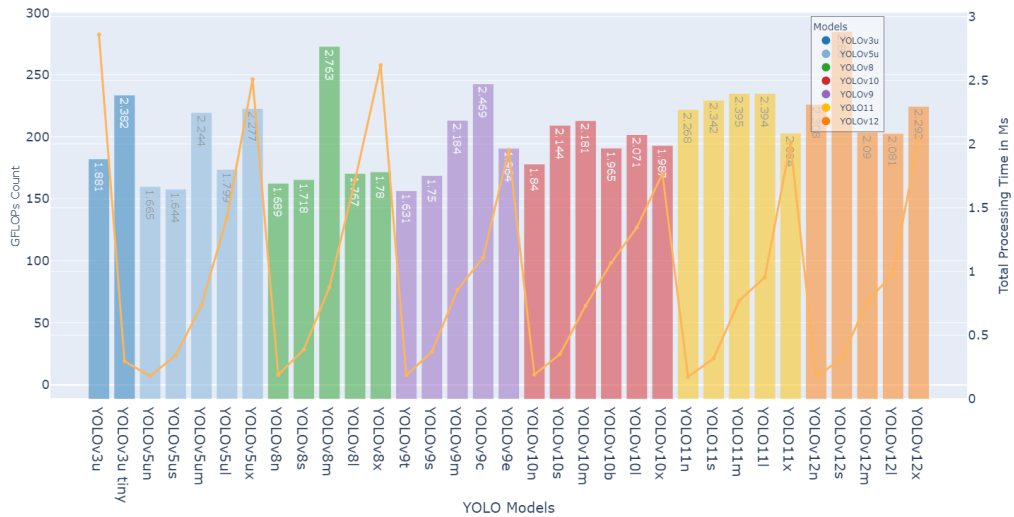


Figure 15: Total processing time and GFLOPs count results on traffic signs dataset

#### 4.1.2. Africa Wildlife Dataset

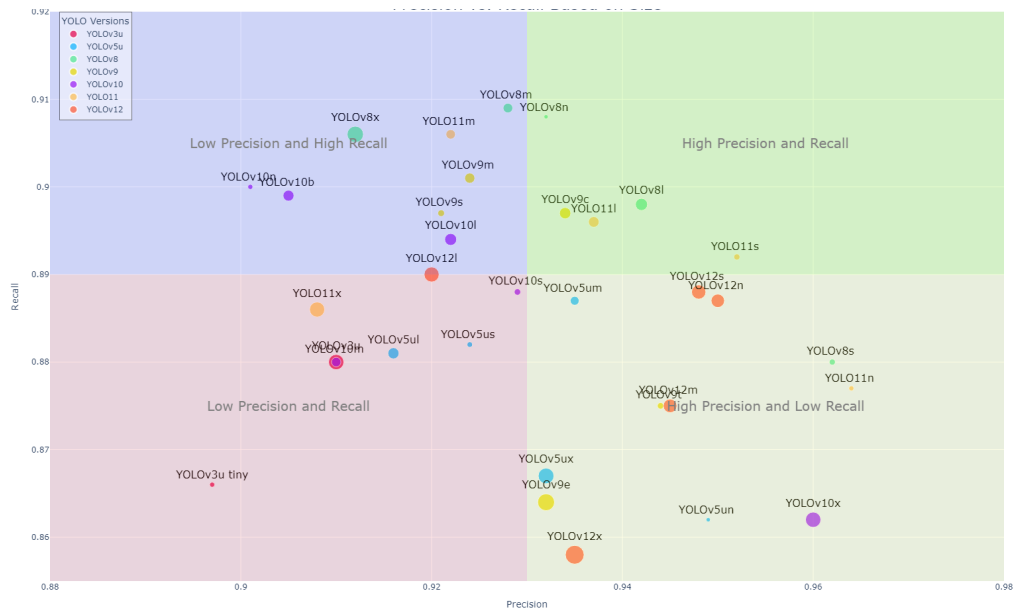
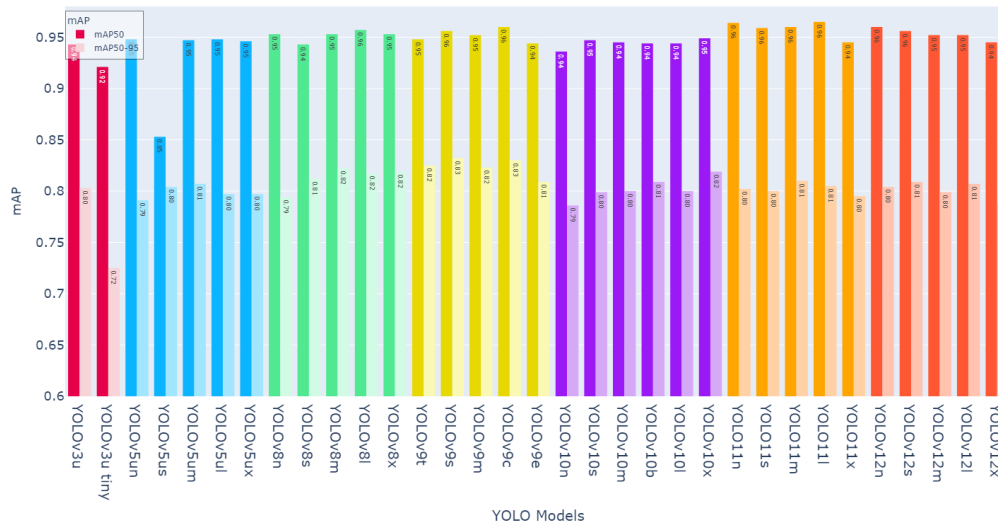
Versions	Precision	Recall	mAP50	mAP50-95	Preprocess Time	Inference Time	Postprocess Time	Total Time	GFLOPs	Size
YOLOv3u	0.91	0.88	0.943	0.803	<b>0.5</b>	6.2	0.4	7.1	<b>282.2</b>	<b>207.86</b>
YOLOv3u tiny	<b>0.897</b>	0.866	0.921	<b>0.725</b>	0.7	0.7	0.4	<b>1.8</b>	19.1	24.44
YOLOv5un	0.949	0.862	0.948	0.791	1.1	<b>0.5</b>	0.4	2	7	5.65
YOLOv5us	0.924	0.882	<b>0.853</b>	0.804	1	0.8	0.4	2.2	23.8	18.58
YOLOv5um	0.935	0.887	0.947	0.807	0.6	2.1	0.4	3.1	64	50.54
YOLOv5ul	0.916	0.881	0.948	0.797	0.7	3.3	0.4	4.4	135	106.85
YOLOv5ux	0.932	0.867	0.946	0.797	0.5	6.6	0.4	7.5	246.2	195.2
YOLOv8n	0.932	0.908	0.953	0.794	1.1	0.5	0.4	2	8.2	6.55
YOLOv8s	0.962	0.88	0.943	0.812	0.9	1	0.4	2.3	28.7	22.59
YOLOv8m	0.928	<b>0.909</b>	0.953	0.822	0.8	2.5	0.4	3.7	78.9	52.12
YOLOv8l	0.942	0.898	0.957	0.817	0.9	3.9	0.4	5.2	165.1	87.77
YOLOv8x	0.912	0.906	0.953	0.819	0.5	7.1	0.4	8	257.6	136.9
YOLOv9t	0.944	0.875	0.948	0.825	1.3	1.1	0.4	2.8	7.7	<b>4.93</b>
YOLOv9s	0.921	0.897	0.956	<b>0.832</b>	1	1.2	0.4	2.6	26.9	15.33
YOLOv9m	0.924	0.901	0.952	0.823	0.9	2.8	0.4	4.1	76.5	40.98
YOLOv9c	0.934	0.897	0.96	0.83	0.9	3.4	0.4	4.7	102.7	51.8
YOLOv9e	0.932	0.864	0.944	0.809	0.5	7.6	0.4	8.5	189.3	117.5
YOLOv10n	0.901	0.9	0.936	0.786	1.1	0.7	<b>0.2</b>	2	8.2	5.59
YOLOv10s	0.929	0.888	0.947	0.799	0.9	1.1	<b>0.2</b>	2.2	24.5	15.9
YOLOv10m	0.91	0.88	0.945	0.8	1	2.4	<b>0.2</b>	3.6	63.4	32.1
YOLOv10b	0.905	0.899	0.944	0.809	0.8	3.2	<b>0.2</b>	4.2	98	39.7
YOLOv10l	0.922	0.894	0.944	0.8	0.7	3.8	<b>0.2</b>	4.7	126.3	50
YOLOv10x	0.96	0.862	0.949	0.819	0.8	6.3	<b>0.2</b>	7.3	169.8	61.4
YOLO11n	<b>0.964</b>	0.877	0.964	0.802	1.1	0.7	0.4	2.2	<b>6.3</b>	5.35
YOLO11s	0.952	0.892	0.959	0.8	1.1	1	0.4	2.5	21.3	18.4
YOLO11m	0.922	0.906	0.96	0.81	0.9	2.4	0.4	3.7	67.7	38.8
YOLO11l	0.937	0.896	<b>0.965</b>	0.805	0.9	3	0.4	4.3	86.6	49
YOLO11x	0.908	0.886	0.945	0.795	0.6	6.1	0.4	7.1	194.4	109
YOLOv12n	0.95	0.887	0.96	0.804	0.9	4.8	0.5	6.2	<b>6.3</b>	5.34
YOLOv12s	0.948	0.888	0.956	0.809	1.1	5.4	<b>1.1</b>	7.6	21.2	18.1
YOLOv12m	0.945	0.875	0.952	0.799	1.1	5	0.8	6.9	67.1	39
YOLOv12l	0.92	0.89	0.952	0.807	1	6	0.6	7.6	88.6	51.2
YOLOv12x	0.935	<b>0.858</b>	0.945	0.788	<b>2.9</b>	<b>9.4</b>	0.5	<b>12.8</b>	198.5	114

Figure 16: Evaluation results for the Africa wildlife dataset.

The results in Figure 16 showcase the performance of the YOLO models on the Africa 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.

**Accuracy:** As illustrated in Figure 17, 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.

**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 Figure 19. Conversely, YOLOv12x exhibits the slowest processing time at 12.8 ms and a GFLOPs



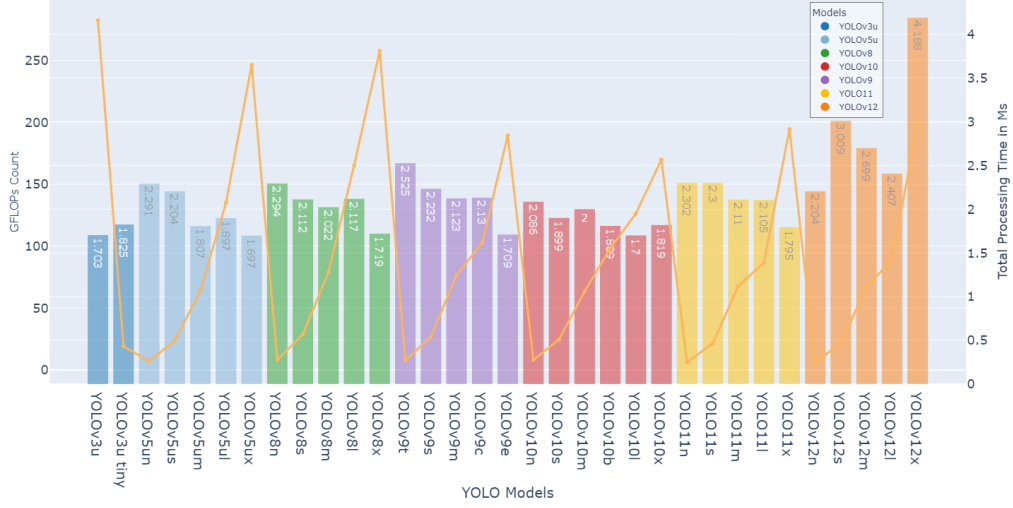


Figure 19: Total processing time and GFLOPs count results on Africa wildlife dataset

count of 198.5, followed by YOLOv9e at 8.5 ms and 189.3 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.

**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 Figures 16, 18, 18, and 19. This demonstrates the robustness of YOLOv9’s smaller models and their effectiveness on small datasets. In contrast, YOLOv12x and YOLO5ux 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.

#### 4.1.3. Ships and Vessels Dataset

Figure 20 presents the performance of YOLO models on the Ships and Vessels dataset, a large dataset featuring tiny objects with varying orientations.

**Accuracy:** The disparity between mAP50-95 and mAP50, illustrated in Figure 21, 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. 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 YOLOv3u architecture and the potential underfitting of smaller models with large datasets.

Versions	Precision	Recall	mAP50	mAP50-95	Preprocess Time	Inference Time	Postprocess Time	Total Time	GFLOPs	Size
YOLOv3u	<b>0.679</b>	0.534	0.519	0.298	<b>0.8</b>	6.2	0.3	7.3	282.5	207.86
YOLOv3u tiny	0.647	0.511	0.489	0.273	1	0.7	0.3	<b>2</b>	18.9	24.44
YOLOv5un	0.635	0.532	0.514	0.298	1.5	0.6	0.3	2.4	7.2	5.65
YOLOv5us	0.653	0.541	0.518	0.299	1.2	0.8	0.3	2.3	24	18.58
YOLOv5um	0.667	0.541	0.526	0.308	0.9	2.1	0.3	3.3	64	50.54
YOLOv5ul	0.654	0.545	0.525	0.305	0.9	3.3	0.3	4.5	134.8	106.85
YOLOv5ux	0.668	<b>0.555</b>	<b>0.531</b>	0.309	<b>0.8</b>	6.7	0.3	7.8	246.2	195.2
YOLOv8n	0.655	0.533	0.515	0.297	1.5	<b>0.5</b>	0.3	2.3	8	6.55
YOLOv8s	0.647	0.545	0.518	0.301	1.1	1	0.3	2.4	28.5	22.59
YOLOv8m	0.669	0.547	0.525	0.302	<b>0.8</b>	2.5	0.3	3.6	79	52.12
YOLOv8l	0.659	0.551	0.526	0.303	0.9	3.9	0.3	5.1	165	87.77
YOLOv8x	0.655	0.55	0.529	0.306	<b>0.8</b>	7.1	0.3	8.2	257.7	136.9
YOLOv9t	0.647	0.516	0.512	0.3	1.4	1.1	0.3	2.8	<b>7.5</b>	<b>4.93</b>
YOLOv9s	0.655	0.552	0.522	0.308	1.4	1.2	0.3	2.9	26.9	15.33
YOLOv9m	0.668	0.551	0.529	0.307	1.1	2.7	0.3	4.1	76.8	40.98
YOLOv9c	0.663	0.547	0.523	0.303	1.2	3.4	0.3	4.9	102.4	51.8
YOLOv9e	0.667	0.537	0.524	0.308	1.1	7.6	0.3	9	189.5	117.5
YOLOv10n	0.584	0.487	0.506	0.31	1.4	0.8	0.2	2.4	8.2	5.59
YOLOv10s	0.586	0.511	0.515	0.319	1.1	1.1	0.2	2.4	24.4	15.9
YOLOv10m	0.588	0.517	0.522	0.322	1	2.4	<b>0.1</b>	3.5	63.4	32.1
YOLOv10b	0.603	0.509	0.523	0.319	1.1	3.2	<b>0.1</b>	4.4	97.9	39.7
YOLOv10l	0.601	0.511	0.522	0.322	1.1	3.8	<b>0.1</b>	5	126.3	50
YOLOv10x	0.6	0.523	0.526	0.321	1	6.3	0.2	7.5	169.8	61.4
YOLO11n	0.574	0.51	0.505	0.311	1.5	0.7	0.3	2.5	<b>6.3</b>	5.35
YOLO11s	0.585	0.535	0.521	0.323	1.3	1	0.3	2.6	21.3	18.4
YOLO11m	0.588	0.541	0.53	0.325	1	2.4	0.3	3.7	67.6	38.8
YOLO11l	0.596	0.531	0.528	0.325	1.1	3	0.4	4.5	86.6	49
YOLO11x	0.596	0.538	0.529	<b>0.327</b>	<b>0.8</b>	6.1	0.3	7.2	194.4	109
YOLOv12n	0.571	0.505	0.505	0.307	1.1	4.7	0.4	6.2	<b>6.3</b>	5.34
YOLOv12s	0.58	0.529	0.516	0.319	1.6	5	0.4	7	21.2	18.1
YOLOv12m	0.584	0.543	0.525	0.325	1.1	4.6	0.4	6.1	67.1	39
YOLOv12l	0.588	0.532	0.522	0.323	2.6	5.2	0.4	8.2	88.5	51.2
YOLOv12x	0.586	0.538	0.52	0.321	1.2	9.3	0.4	10.9	198.5	114

Figure 20: Evaluation results for the ships and vessels dataset

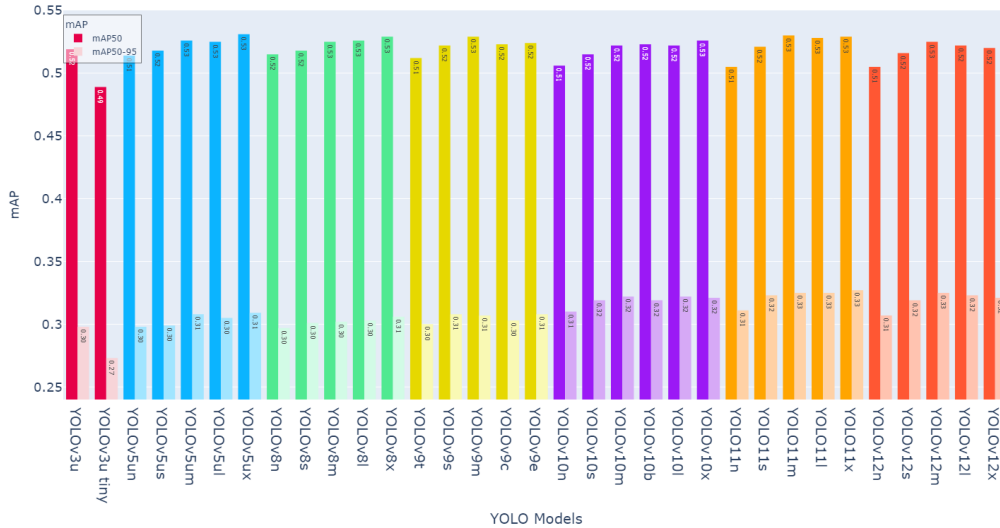


Figure 21: mAP50 and mAP50-95 YOLO results on ships and vessel dataset. Each model is represented by two bars: the left bar shows the mAP50 score, while the right bar represents the mAP50-95 score

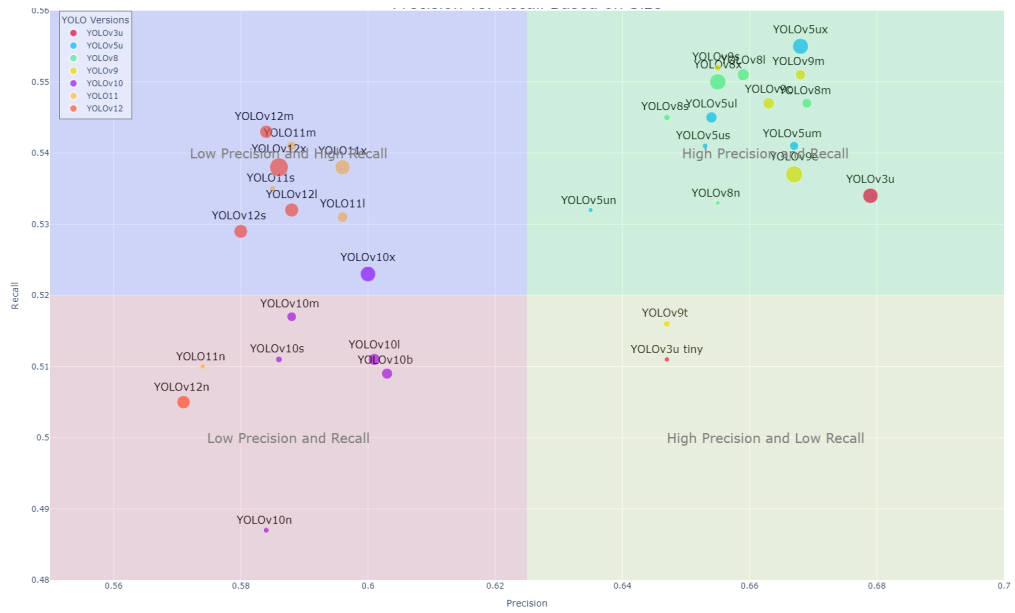


Figure 22: Precision vs. Recall based on size results on ships and vessels dataset, with larger circles indicating larger model sizes

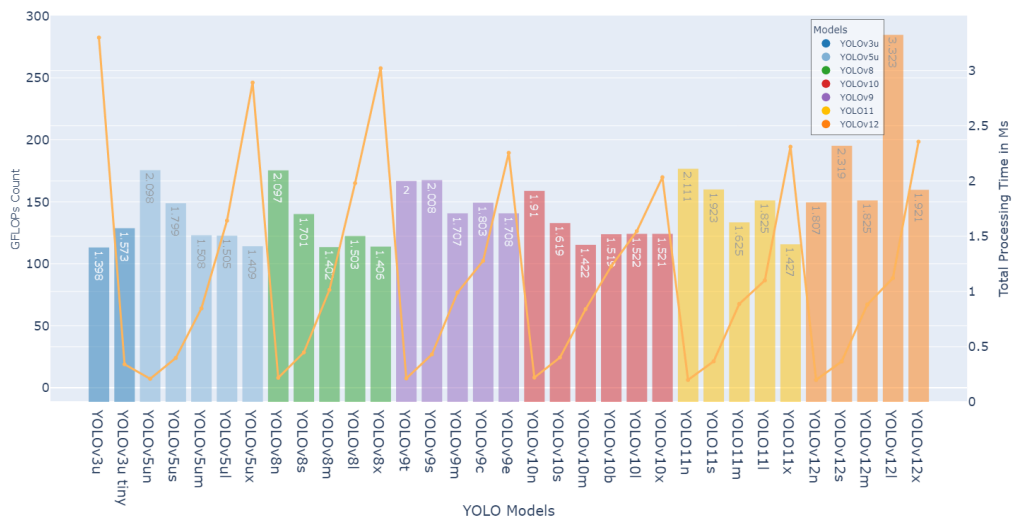


Figure 23: Total processing time and GFLOPs count results on ships and vessels dataset

Table 4: Overall ranking of YOLO algorithms based on family

Version	Acc.	Speed	GFLOPs	Size
YOLOv3u	<b>7</b>	<b>1</b>	4	<b>7</b>
YOLOv5u	6	4	4	5
YOLOv8	5	5	<b>7</b>	6
YOLOv9	3	6	6	4
YOLOv10	4	2	3	<b>1</b>
YOLO11	<b>1</b>	3	<b>1</b>	2
YOLOv12	<b>1</b>	<b>7</b>	2	3

**Computational Efficiency:** As illustrated in Figure 23, YOLOv3u tiny achieved the fastest processing time at 2 ms, closely followed by YOLOv8n and YOLOv5un, 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, YOLOv12x exhibited the lowest speed, with 10.9 ms inference time and 198.5 GFLOPs count, highlighting the shortcomings of the newest YOLO family.

**Overall Performance:** The results in Figures 20, 21, 22, and 23 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.

## 5. Discussion

Models were ranked by accuracy, speed, GFLOPs count, and size across three datasets, as shown in Figure 24. To ensure fair comparisons, models were categorized by scale. YOLOv3u tiny was classified as small, YOLOv9t as nano, YOLOv9c as medium, YOLOv9c and YOLOv10b as large, and YOLOv9e as extra-large. Additionally, family-wide performance was analyzed by averaging rankings across all scales to compare overall performance as seen in Table 4.

Accuracy was measured using mAP50-95 for a comprehensive performance assessment, while speed rankings were based on total processing time. Rankings range from Rank 1 (best) to the lowest performer, with results highlighted in bold. This approach ensures a fair evaluation of architectural improvements and computational trade-offs across model sizes and families.

### 5.1. Performance Analysis by Model Scale

**Nano Models.** Among nano models, YOLOv9t ranked highest in accuracy and speed, benefiting from its lightweight GELAN architecture, which optimizes parameter efficiency while maintaining strong detection capabilities. However, YOLOv10n, despite its computational efficiency, exhibited lower accuracy due to its reliance on the One-to-One head and NMS-Free training, which struggled with densely packed objects.

Version	Acc.	Speed	GFLOPs	Size
YOLOv5un	<b>6</b>	2	3	5
YOLOv8n	4	2	5	<b>6</b>
YOLOv9t	<b>1</b>	<b>6</b>	4	<b>1</b>
YOLOv10n	5	<b>1</b>	<b>6</b>	4
YOLO11n	3	4	<b>1</b>	3
YOLOv12n	2	5	<b>1</b>	2
YOLOv3u tiny	<b>7</b>	<b>1</b>	<b>1</b>	<b>7</b>
YOLOv5us	5	3	4	5
YOLOv8s	5	4	<b>7</b>	6
YOLOv9s	2	<b>7</b>	6	<b>1</b>
YOLOv10s	3	2	5	2
YOLO11s	3	4	2	4
YOLOv12s	<b>1</b>	6	2	3
YOLOv5um	<b>6</b>	2	2	6
YOLOv8m	5	4	<b>6</b>	<b>7</b>
YOLOv9m	2	5	5	5
YOLOv10m	4	<b>1</b>	<b>1</b>	<b>1</b>
YOLO11m	<b>1</b>	3	4	2
YOLOv12m	2	<b>6</b>	3	3
YOLOv5ul	4	3	6	<b>7</b>
YOLOv8l	<b>6</b>	6	<b>7</b>	6
YOLOv9c	3	5	4	5
YOLOv10b	<b>6</b>	<b>1</b>	3	<b>1</b>
YOLOv10l	4	4	5	3
YOLO11l	<b>1</b>	2	<b>1</b>	2
YOLOv12l	2	<b>7</b>	2	4
YOLOv3u	<b>5</b>	2	<b>7</b>	<b>7</b>
YOLOv5ux	<b>5</b>	4	5	6
YOLOv8x	4	4	6	5
YOLOv9e	<b>5</b>	<b>6</b>	2	4
YOLOv10x	<b>1</b>	3	<b>1</b>	<b>1</b>
YOLO11x	2	<b>1</b>	3	2
YOLOv12x	2	<b>6</b>	4	3

Figure 24: Overall ranking of YOLO algorithms based on size



*Small Models.* YOLOv12s emerged as the best small model in terms of accuracy, likely due to its R-ELAN-based feature aggregation. However, it performed poorly in speed, as the added complexity of Area Attention (A2) increased inference time. In contrast, YOLOv10s and YOLO11s provided a balanced performance, with YOLO11s leveraging the C3k2 block for improved feature extraction and YOLOv10s excelling in speed due to its efficient head design.

*Medium Models.* YOLO11m achieved the best balance of accuracy and efficiency in this category, owing to C2PSA, which enhances spatial feature capture. YOLOv10m outperformed all models in terms of computational efficiency but had lower accuracy due to its reliance on NMS-Free training. Meanwhile, YOLOv9m, while achieving strong accuracy, ranked lower in efficiency due to its reliance on gradient-based optimizations, which added computational overhead.

*Large Models.* YOLO11l outperformed other large models in accuracy and speed, thanks to the integration of C2PSA and refined convolutional operations. In contrast, YOLOv9c and YOLOv12l struggled with efficiency, with YOLOv9c being computationally expensive and YOLOv12l suffering from latency due to FlashAttention’s memory overhead.

*Extra-Large Models.* YOLO11x provided the best balance of accuracy and computational efficiency, with its C3k2-based optimization reducing inference overhead. However, YOLOv9e and YOLOv12x exhibited slower speeds and inefficient GFLOPs utilization, highlighting the limitations of their respective architectures when scaling up.

## 5.2. Performance Analysis by Family

*YOLOv12.* YOLOv12 delivered strong accuracy results, ranking first alongside YOLO11 in accuracy across models. This performance can be attributed to its integration of the Area Attention Module (A2) and Residual Efficient Layer Aggregation Networks (R-ELAN), which enhanced feature extraction and contextual understanding. However, this came at the cost of computational efficiency, as the added complexity introduced significant latency and reduced processing speed. FlashAttention, while designed to optimize memory access, further contributed to increased inference time. While YOLOv12 excelled in accuracy, its slow processing speed and higher computational demands limited its overall practicality, highlighting the trade-off between architectural complexity and real-world efficiency.

*YOLO11.* The YOLO11 family consistently ranked among the best due to its C3k2 block and C2PSA, enhancing efficiency and contextual understanding by ranking first in terms of accuracy and GFLOPs count and among the first in terms of speed and size. These innovations enabled YOLO11 to achieve high accuracy with low computational overhead, making it a well-balanced choice. While YOLOv8 was widely used for various tasks such as Pose Estimation and OBB, YOLO11 has emerged as a superior alternative, offering better feature extraction, inference speed, and accuracy. This study establishes YOLO11 as a new benchmark, demonstrating its high detection accuracy and low-latency processing, and various capabilities, making it ideal for real-world applications requiring both speed and precision.

*YOLOv10.* YOLOv10 models were highly efficient, particularly in speed and size, due to the One-to-One head and NMS-Free training. However, their accuracy was subpar compared to YOLOv9, YOLO11, and YOLOv12, particularly in scenarios requiring precise bounding box refinement, such as the Traffic Signs dataset.

*YOLOv9*. Despite strong accuracy due to PGI and GELAN, *YOLOv9* models suffered from slow inference times, limiting their real-time applicability. Their architectural choices prioritized precision over efficiency, making them suitable for applications where accuracy is paramount but speed is less critical.

*YOLOv8*, *YOLOv5u*, and *YOLOv3u*. While *YOLOv8*, *YOLOv5* surpassed *YOLOv3u* in accuracy, they lagged behind newer models in overall efficiency. *YOLOv8* introduced the C2k module for improved feature extraction, but its higher computational cost made it less efficient than more recent architectures. With *YOLO11* offering superior accuracy, speed, and efficiency, *YOLOv8* is no longer the go-to model for real-world applications, as *YOLO11* provides a better balance between performance and computational overhead. Meanwhile, *YOLOv5u* improved on *YOLOv3u* but lacked modern enhancements like self-attention, limiting its scalability.

Overall, the rankings highlight the dominance of *YOLO11* as the most balanced family, with *YOLOv10* excelling in efficiency and *YOLOv9* maintaining strong accuracy at the cost of speed. *YOLOv12*, despite its ambitious design and exceptional accuracy, failed to outperform its predecessors due to increased latency, reinforcing the importance of balancing architectural complexity with real-world feasibility.

### 5.3. 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 like *YOLOv9t* and *YOLOv9s* performed best on the Africa Wildlife dataset, showcasing the effectiveness of small-scaled models when handling limited datasets.

### 5.4. Impact of Training Datasets

The performance of YOLO models is influenced by the training datasets used, as shown in Figures 12, 16, and 20. 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. Additionally, this highlights the need for a balanced consideration of accuracy, speed, and model size when selecting YOLO models for specific applications.

## 6. Real-Life Applications

The advancements in YOLO models have enabled their application across diverse real-world scenarios, each with distinct demands for model performance, such as computational efficiency, accuracy, and adaptability to varying object complexities. Based on the benchmark of various YOLO models and related papers, this section highlights how different models excel under specific scenarios, making them suitable for deployment in various tasks across industries, as shown in Table 5.

Table 5: YOLO Model Recommendations for Real-Life Applications

Scenario	Recommended YOLO Models	Application Examples
Computationally Constrained Environments	YOLOv10(n/s/m) YOLO11(n/s/m)	Drone surveillance, embedded systems, battery-powered devices [29, 80, 25, 27, 9, 8]
Real-Time Monitoring and Rapid Response	YOLOv10(n/s/m) YOLO11(n/s/m)	Autonomous vehicles, disaster response, traffic monitoring [29, 80, 25, 75, 27, 9, 8]
Detection of Small Objects	YOLO11(m/l/x) YOLOv12(m/l/x)	Wildlife tracking, satellite imaging of ships and small vehicles [75, 27, 9, 8]
Detection of Large Objects	YOLOv8(s/m/l/x) YOLOv9(t/s/m/c/e)	Satellite imaging (buildings, forests, urban development) [83, 55, 43]
Overlapping and Densely Packed Objects	YOLOv9(t/s/m/c/e)	Traffic intersections, wildlife in dense foliage
Oriented and Rotated Object Detection	YOLO11 (OBB Support) YOLOv12 (OBB Support)	Satellite imagery, aerial monitoring of ships [46, 21]
Handling Large Datasets	YOLO11(m/l/x) YOLOv12(m/l/x)	City-wide monitoring, biodiversity mapping [5, 76]
Handling Small Datasets	YOLOv9(t/s/m/c)	Anti-poaching monitoring, niche industrial tasks [1, 64, 84]

### 6.1. Computationally Constrained Environments

In scenarios with limited computational resources, such as drones, embedded systems, and battery-powered devices, models must be lightweight and efficient to avoid draining power or overloading hardware. The nano, small, and medium-sized models of the YOLO11 and YOLOv10 families are optimal choices due to their high accuracy, compact model sizes, low inference times, and low GFLOPs count leading to minimal resource consumption, as also seen in related papers [29, 80, 25, 27, 9, 8].

### 6.2. Real-Time Monitoring and Rapid Response

For applications requiring real-time detection, such as autonomous driving, surveillance, and disaster response, models need to balance accuracy and speed. As illustrated by this study and related research papers [29, 80, 25, 75, 27, 9, 8], the nano, small, and medium-sized YOLO11 and YOLOv10 models demonstrate strong performance in maintaining fast inference while accurately detecting objects across various scales.

### 6.3. Detection of Small and Large Objects

Object detection models often face the challenge of accurately identifying small and large objects. Small objects, such as wildlife or distant vehicles, require models capable of capturing fine spatial details. Medium, large, and extra-large-sized YOLO11 and YOLOv12 models excel in these scenarios, as illustrated in our paper and related work [75, 27, 9, 8]. Meanwhile, all YOLOv9 and YOLOv8 models offer robust performance for detecting large objects, making them ideal for scenarios such as satellite imaging of buildings, forests, and urban development [83, 55, 43].

### 6.4. Overlapping and Densely Packed Objects

Detecting objects that overlap or appear densely packed is a major challenge in traffic intersections or wildlife monitoring scenarios. In these cases, models must efficiently distinguish between objects and avoid false positives. All YOLOv9 models are highly effective in handling overlapping objects, as seen in this benchmark and related work [46, 21]. This suits them, particularly for applications like detecting animals in dense foliage or distinguishing closely packed traffic signs.

### 6.5. Oriented Object Detection

Objects often appear at varying angles or orientations in satellite imaging and certain surveillance applications, requiring models with specialized detection capabilities. The YOLO11 and YOLOv12 families’ support for OBB enables effective detection of rotated objects, achieving higher accuracy than YOLOv8 models.

### 6.6. Handling Large and Very Small Datasets

Large-scale datasets with diverse object categories require models that generalize well without overfitting. Medium, large, and extra-large YOLO11 and YOLOv12 models excel in such scenarios, as showcased in our paper and related studies [5, 76]. For tiny datasets, where overfitting is a concern, all YOLOv9 models are optimal choices. Their compact sizes and efficient parameter usage enable effective learning from limited data [1, 64, 84].

## 7. Conclusion

This benchmark study thoroughly evaluates the performance of various YOLO algorithms. It pioneers a comprehensive comparison of YOLOv12 against its predecessors and assesses their performance across three diverse datasets: Traffic Signs, African Wildlife, and Ships and Vessels. The datasets were carefully selected to encompass various 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?
- What are each YOLO version’s specific strengths and limitations, and how can these insights inform the selection of suitable algorithms for various applications?
- Which YOLO models are best suited for specific real-world scenarios?

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.

Despite being the newest addition to the YOLO family, YOLOv12 delivered an underwhelming performance. Its ambitious architectural changes, including the Area Attention Module and R-ELAN, introduced complexity without translating into clear advantages. The model struggled to balance accuracy and speed, highlighting the challenges of effectively integrating advanced attention mechanisms within the YOLO framework.

While the evaluated algorithms show promising performance, there is room for improvement. Future research should focus on optimizing YOLOv10’s accuracy while maintaining its speed

and efficiency. Additionally, simplifying YOLOv12's complex attention mechanisms could enhance its speed without compromising accuracy. Further architectural advancements could lead to even more efficient and accurate YOLO models. Our future work will address these gaps and explore enhancements to maximize overall efficiency and real-world applicability.

## Declarations

- Funding: Not applicable
- Conflict of interest/Competing interests: The authors declare no conflict of interest.
- Ethics approval and consent to participate: Not applicable
- Consent for publication: The authors confirm that all necessary permissions have been obtained for the publication of this work, including the use of copyrighted material and any personal information or sensitive data.
- Data availability: Data is available on GitHub and will be publicly available upon publication of the final version
- Materials availability: Not applicable
- Code availability: Source code is available on GitHub and will be publicly available upon publication of the final version.
- Author contribution: All authors contributed to the study conception and design. Material preparation, data collection, and analysis were performed by Nidhal Jegham and Chan Young Koh. The first draft of the manuscript was written by Nidhal Jegham, and all authors commented on and edited previous versions of the manuscript. All authors read and approved the final manuscript. Dr. Abdeltawab Hendawi supervised the work of this study.

## References

- [1] Ammar Ahmed, Abdul Manaf, Ali Shariq Imran, Zenun Kastrati, and Sher Muhammad Daudpota. Small data, big impact: A multi-locale bone fracture detection on an extremely limited dataset via crack-informed yolov9 variants. In *2024 International Conference on Frontiers of Information Technology (FIT)*, pages 1–6. IEEE, 2024.
- [2] 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.
- [3] Bader Aldughayfiq, Farzeen Ashfaq, NZ Jhanjhi, and Mamoon Humayun. Yolo-based deep learning model for pressure ulcer detection and classification. In *Healthcare*, volume 11, page 1222. MDPI, 2023.
- [4] 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.
- [5] Mujadded Al Rabbani Alif. Yolov11 for vehicle detection: Advancements, performance, and applications in intelligent transportation systems, 2024.
- [6] Isaiah Francis E Babila, Shawn Anthonie E Villazor, 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.
- [7] 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.
- [8] Murat Bakirci and Irem Bayraktar. The cutting-edge yolo11 for advanced aircraft detection in synthetic aperture radar (sar) imagery. In *2024 8th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, pages 1–6. IEEE, 2024.

- [9] Murat Bakirci, Petro Dmytrovych, Irem Bayraktar, and Oleh Anatoliyovych. Multi-class vehicle detection and classification with yolo11 on uav-captured aerial imagery. In *2024 IEEE 7th International Conference on Actual Problems of Unmanned Aerial Vehicles Development (APUAVD)*, pages 191–196. IEEE, 2024.
- [10] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] Weizhi Feng, Meidong Liu, Yan Sun, Suyu Wang, and Jingli Wang. The use of a blueberry ripeness detection model in dense occlusion scenarios based on the improved yolov9. *Agronomy*, 14(8):1860, 2024.
- [22] 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.
- [23] 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.
- [24] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.
- [25] Sitong Guan, Yiming Lin, Guoyu Lin, Peisen Su, Siluo Huang, Xianrong Meng, Pingzeng Liu, and Jun Yan. Real-time detection and counting of wheat spikes based on improved yolov10. *Agronomy*, 14(9):1936, 2024.
- [26] Juan Guerrero-Ibáñez, Sherali Zeadally, and Juan Contreras-Castillo. Sensor technologies for intelligent transportation systems. *Sensors*, 18(4), 2018.
- [27] Jianwei Huang, Kangbo Wang, Yue Hou, and Jiahe Wang. Lw-yolo11: A lightweight arbitrary-oriented ship detection method based on improved yolo11. *Sensors*, 25(1):65, 2024.
- [28] Muhammad Hussain. YOLOv1 to v8: Unveiling each variant—a comprehensive review of yolo. *IEEE Access*, 12:42816–42833, 2024.
- [29] Muhammad Hussain. YOLOv5, yolov8 and yolov10: The go-to detectors for real-time vision, 2024.
- [30] Rasheed Hussain and Sherali Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1275–1313, 2019.
- [31] Glenn Jocher. Ultralytics yolov5, 2020.
- [32] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.
- [33] Glenn Jocher and Jing Qiu. Ultralytics yolo11, 2024.
- [34] 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.
- [35] 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.
- [36] Habib Khan, Inam Ullah, Mohammad Shabaz, Muhammad Faizan Omer, Muhammad Talha Usman, Mohammed Seghir Guellil, and JaKeoung Koo. Visionary vigilance: Optimized yolov8 for fallen person detection with large-scale benchmark dataset. *Image and Vision Computing*, 149:105195, 2024.

- [37] Habib Khan, Inam Ullah, Mohammad Shabaz, Muhammad Faizan Omer, Muhammad Talha Usman, Mohammed Seghir Guellil, and JaKeoung Koo. Visionary vigilance: Optimized yolov8 for fallen person detection with large-scale benchmark dataset. *Image and Vision Computing*, 149:105195, 2024.
- [38] 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.
- [39] Guofa Li, Zefeng Ji, Xingda Qu, Rui Zhou, and Dongpu Cao. Cross-domain object detection for autonomous driving: A stepwise domain adaptative yolo approach. *IEEE Transactions on Intelligent Vehicles*, 7(3):603–615, 2022.
- [40] 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.
- [41] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.
- [42] 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.
- [43] Jingjing LIU, Fengqian SUN, Haoxiang ZHANG, He JIANG, Junhui CHEN, Qiqi KOU, and Deqiang CHENG. Lightweight coal particle group instance segmentation method based on yolov8. *Journal of China Coal Society*, 2024.
- [44] 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.
- [45] 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.
- [46] Dunlu Lu and Yangxu Wang. Mar-yolov9: A multi-dataset object detection method for agricultural fields based on yolov9. *Plos one*, 19(10):e0307643, 2024.
- [47] 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.
- [48] 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.
- [49] 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.
- [50] 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.
- [51] 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.
- [52] 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.
- [53] 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.
- [54] 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.
- [55] Yue Peng, Alifu Kurban, and Mengmei Sang. An improved yolov8 method for measuring the body size of xinjiang bactrian camels. *International Journal of Advanced Computer Science & Applications*, 15(8), 2024.
- [56] 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.
- [57] 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.
- [58] Sovit Rath. Yolov8 ultralytics: State-of-the-art yolo models. *LearnOpenCV-Learn OpenCV, PyTorch, Keras, TensorflowWith Examples and Tutorials*, 2023.
- [59] 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.
- [60] 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.
- [61] Joseph Redmon and Ali Farhadi. YoloV3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [62] 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.
- [63] 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.
- [64] Wimonthip Saenprasert, Ei Ei Tun, Amir Hajian, Watchara Ruangsang, and Supavadee Aramvith. Yolo for small objects in aerial imagery: a performance evaluation. In *2024 21st International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 720–727. IEEE, 2024.
- [65] 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.
- [66] Ranjan Sapkota, Rizwan Qureshi, Marco Flores Calero, Muhammad Hussain, Chetan Badjugar, Upesh Nepal, Alwin Poulouse, 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.
- [67] 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.
- [68] 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.
- [69] 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.
- [70] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.
- [71] 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.
- [72] 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.
- [73] Yunjie Tian, Qixiang Ye, and David Doermann. YoloV12: Attention-centric real-time object detectors, 2025.
- [74] Ultralytics. YOLOv5: A state-of-the-art real-time object detection system. <https://docs.ultralytics.com>, 2021. Accessed: insert date here.
- [75] Ravi Teja Vempati. *Real-time detection and recognition of license plate using YOLO11 object detection model*. PhD thesis, KANSAS STATE UNIVERSITY, 2024.
- [76] Ravi Teja Vempati. *Real-time detection and recognition of license plate using YOLO11 object detection model*. PhD thesis, KANSAS STATE UNIVERSITY, 2024.
- [77] 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.
- [78] 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.
- [79] 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.
- [80] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, et al. YoloV10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, 37:107984–108011, 2025.
- [81] 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.
- [82] 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.
- [83] Yan Wang, Qianjie Rong, and Chunhua Hu. Ripe tomato detection algorithm based on improved yolov9. *Plants*, 13(22):3253, 2024.
- [84] Yan Wang, Qianjie Rong, and Chunhua Hu. Ripe tomato detection algorithm based on improved yolov9. *Plants*, 13(22):3253, 2024.
- [85] 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.
- [86] 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.
  - [87] 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.
  - [88] 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.