Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 246 (2024) 2792–2801

Procedia Computer Science

www.elsevier.com/locate/procedia

28th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2024)

Real-time Object Detection in Autonomous Vehicles with YOLO

Nusaybah M.Alahdal[a], Felwa Abukhodair[a,b], Leila Haj Meftah[c], Asma Cherif[a,b,*]

[a]*Department of Information Technology, King Abdulaziz University, Jeddah 21589, KSA*
[b]*Center of Excellence in Smart Environment Research, King Abdulaziz University, Jeddah 22252, KSA*
[c]*PRINCE Research Lab ISITCom H- Sousse, Sousse University, Sousse 4000, Tunisia*

Abstract

AI analytics enables autonomous cars to detect and recognize objects, such as other vehicles, pedestrians, traffic signs, and obstacles, in real-time. Deep learning models, notably the You Only Look Once (YOLO) model, have demonstrated accuracy and speed in obstacle avoidance. However, current datasets are limited, lacking diversity and labeling, hindering their ability to represent real-world scenarios accurately. Besides, previous studies have focused extensively on specific object classes, such as pedestrians and vehicles, often neglecting other objects like bikes and road signs. To address this, we introduce a novel dataset tailored for AV environments, encompassing various road object types under different conditions. Our innovative methodology relies on self-supervised learning using the late YOLO version to improve model robustness with limited labeled data and AI-driven adaptive model optimization based on real-time feedback. We evaluate three YOLO architectures—YOLOv5, YOLOv7, and YOLOv8—customized for AV object detection. Our assessment covers everyday AV objects such as cars, pedestrians, bicycles, and road signs, emphasizing early detection. We employ the VSim-AV simulator dataset to ensure robust evaluation, augmented with preprocessing techniques to optimize data quality and model generalization. The study reveals that YOLOv5 and YOLOv8 outperform YOLOv7 regarding precision and recall across various object classes, with YOLOv5 leading at 1.3 ms/image and YOLOv8 at 3.3 ms/image. The mean average precision was 0.94 for YOLOv5, 0.441 for YOLOv7, and 0.927 for YOLOv8, highlighting the limitations in current literature and challenges in YOLO model performance.

Keywords: Smart city; Simulation environment; Deep learning; Artificial intelligence; YOLO; Object detection; Autonomous vehicles.

1. Introduction

AI analytics provide several key benefits to Autonomous Vehicles (AVs), enhancing their safety and efficiency [1]. This requires robust object detection in various environments. Object detection and avoidance in autonomous vehicles is a critical aspects of their development, enabling them to perceive and understand their surroundings. The essential decision-making processes of AV systems heavily depend on their ability to recognize and categorize items like cars,

* Corresponding author. Tel.: +966-012-695-1603 ; fax: +966-67434 -67293.
E-mail address: nalahdal0012@stu.kau.edu.sa, fabukhodair@kau.edu.sa, leila.hajmeftah@isitc.u-sousse.tn, acherif@kau.edu.sa

1877-0509
10.1016/j.procs.2024.09.392

people, bikes, and road signs, which directly impact road safety and navigation [2]. This involves using many Internet of Things (IoT) and AI technologies to enhance object detection in AV scenarios.

The development of obstacle avoidance in AVs has been dramatically influenced by advancements in sensor technologies, including LiDAR (Light Detection and Ranging), radar, and cameras, which provide crucial input for detecting and assessing potential obstacles in the vehicle's path. These sensors generate large volumes of data, which are processed and analyzed using AI and machine learning algorithms to identify and classify obstacles in the vehicle's vicinity. Obstacle avoidance relies heavily on computer vision to perceive and interpret the environment. Computer vision, a field of artificial intelligence, enables AVs to understand and analyze visual information from the surrounding environment using cameras and other optical sensors. Computer vision algorithms process visual data to detect and recognize obstacles such as pedestrians, vehicles, cyclists, and other relevant objects in the vehicle's path. This is essential for identifying potential hazards and obstacles.

Deep learning-based methods have become famous for object detection due to their ability to extract intricate patterns and characteristics from large datasets [3, 4]. Obstacle avoidance capability is achieved through sensor data processing, real-time decision-making algorithms, and intelligent control systems. Methods like You Only Look Once (YOLO) are more and more used to enhance the performance of AVs in recognizing and identifying objects on the road [6, 7]. YOLO series has garnered significant attention because of its excellent accuracy and real-time processing capabilities [5]. YOLO models provide fast inference times without sacrificing detection performance due to their single-pass design [7]. Over time, many versions of the YOLO architecture have been created, each introducing enhancements to improve detection accuracy and efficiency. The original YOLO model, proposed by Redmon et al. [9], transformed object detection by framing it as a single regression problem, significantly improving speed. YOLOv2 and YOLOv3, also by Redmon and Farhadi [10, 11], introduced batch normalization, anchor boxes, and multiscale training to enhance accuracy and versatility. YOLOv3, further refined by Redmon and Farhadi, incorporated Darknet-53 for feature extraction, making it more robust and accurate for detecting smaller objects. These foundational studies set the stage for subsequent advancements in the YOLO series. YOLO series incorporates enhancements in network design, feature extraction, and optimization techniques. These foundational studies set the stage for subsequent advancements in the YOLO series. The YOLO series incorporates enhancements in network design, feature extraction, and optimization techniques. For instance, YOLOv5 implemented a Focus layer for better feature extraction and AutoAnchor for dataset-specific optimization [12]. YOLOv7 introduced novel architectural changes that significantly enhanced accuracy and efficiency [13]. YOLOv8 further refined these techniques, improving both speed and detection accuracy. [14] For AVs to proactively respond to any risks and obstructions on the road, it is essential to compare the performance of different designs in early object detection tasks.

Though many studies addressed object detection for AVs, they have focused extensively on specific object classes, such as pedestrians and vehicles, often neglecting other objects like bikes and road signs. This narrow focus limits the comprehensive understanding of AV environments. Additionally, most studies rely on datasets such as KITTI, which, while valuable, do not capture the full range of real-world driving conditions and scenarios that AVs might encounter. These limitations hinder the generalization of findings to more diverse and challenging environments.

Our research aims to address these gaps by covering a wide range of object classes, including pedestrians, vehicles, bikes, and road signs. Utilizing the VSim-AV simulator dataset, which includes diverse scenarios and lighting conditions, provides a more holistic evaluation of object detection performance. Furthermore, our study is the first to evaluate the latest YOLO architectures, including YOLOv8, in real-time AV scenarios. This comprehensive approach enhances the understanding of each model's strengths and weaknesses and provides insights into their applicability in practical AV systems. We use the VSim-AV simulator dataset, explicitly created for modeling AV environments, to assess the three designs. The dataset offers a robust assessment platform for object identification algorithms since it contains a variety of scenarios, lighting conditions, and item variances found in everyday driving situations. The dataset was generated and prepared using RoboFlow and VSim-AV, ensuring diverse and comprehensive training and testing images. We employed preprocessing and augmentation techniques to address common variations and challenges in real-world audio-visual (AV) scenarios. Preprocessing steps, such as auto-orientation, grayscale conversion, and auto-adjusting contrast, were utilized to maintain consistent input images. Augmentation techniques were introduced to introduce controlled variations and improve the model's ability to handle noise levels, changes in illumination, and orientation variations. The study evaluated YOLOv5, YOLOv7, and YOLOv8 models using precision, recall, and mean average precision metrics. The execution environment used NVIDIA GPU acceleration, specifically

Tesla T4, with NVIDIA GPU drivers version 535.104.05 and CUDA version 12.2, providing a robust platform for deep learning tasks. The original hyperparameters were preserved for comparison baselines, prioritizing accuracy over adjustments or additional techniques. Code implementations were exclusively Python-crafted and sourced from official repositories for reliability and adherence to field standards.

Our research contributes to advancing computer vision for object detection in AV environments by providing insights for system enhancements and guiding future research endeavors. Specifically, our contributions include:

- Creating a novel dataset tailored for AV environments to enhance the robustness of our evaluations.
- Implementing deep learning technique to assess the performance of object detection models in AV scenarios.
- Comparing multiple YOLO architectures namely YOLOv5, YOLOv7, and YOLOv8 and highlighting each model's strengths and weaknesses.
- Discussing the difficulties encountered during the evaluation process and suggesting future research to improve object detection in AV scenarios.

Notably, to our knowledge, this study represents the first attempt to evaluate the performance of various YOLO architectures, including the latest version, YOLOv8, in real-time AV scenarios for detecting multiple classes of objects in a simulation environment. The remainder of this paper is organized as follows: Section 2 reviews related work in the field, Section 3 describes the research methodology, Section 4 presents the results and discusses their implications, and finally, Section 5 provides the conclusion and suggests directions for future research.

2. Related Works

Since real-time object detection provides critical information for advanced driver assistance systems (ADAS) and decision-making, it has become an essential part of AV systems. This section will examine several relevant research studies, approaches, and real-time AV object identification strategies. Deep learning techniques have remarkably succeeded in object detection tasks in recent years. YOLOv1 [19], YOLOv2 [20], and YOLOv3 [21] make up the YOLO series [22], which is one of the most widely used frameworks. YOLOv1 introduced a single-pass detection pipeline that treated object detection as a regression issue and enabled real-time object detection. YOLOv2's addition of features, including multiscale training and anchor boxes, achieved better accuracy. By using varied-sized anchors for different object sizes and integrating a feature extraction network named Darknet-53, YOLOv3 considerably improved performance.

A real-time multi-task framework for detecting vehicles and pedestrians was developed by the authors of [23], utilizing YOLOv5. The proposed architecture performed both tasks using a single neural network, significantly reducing computational complexity and increasing detection accuracy. The experiments conducted on the KITTI dataset demonstrated that this framework outperforms current methods in both speed and accuracy. The system's real-time ability to identify vehicles and people significantly enhances AV safety. Still, its limitation is its focus on detecting only cars and people, excluding other road items. Jia, X., and team [24] developed an improved YOLOv5-based object recognition system for self-driving cars. The method improved the model's accuracy and speed through structural re-parameterization, neural architecture exploration, a layer for small object detection, and a coordinate attention tool. When tested using the KITTI dataset, this method surpassed others in accuracy and speed. However, it required extensive training data and was sensitive to noise and occlusion. Choudhury and colleagues proposed a hybrid deep-learning structure for object detection and tracking in self-driving cars [25]. This structure included a dual-function object detector and tracker model. The detector used a Faster R-CNN model, while the tracker relied on the Kalman filter. Tests on the KITTI dataset demonstrated that this new structure was superior to previous methods in accuracy and speed. The ability to detect and track objects in real-time enhances AV safety, but this structure's complexity may demand significant computational resources. Existing literature on autonomous vehicles focuses primarily on specific

Table 1. Comparison of methodologies and results.

Study	Methodology	Limitations
[23]	Multi-task framework using YOLOv5	Limited to vehicles and pedestrians
[24]	Structural re-parameterization, neural architecture exploration	Requires extensive training data, sensitive to noise and occlusion
[25]	Hybrid structure with Faster R-CNN and Kalman filter	High computational complexity
Proposal	YOLOv5,v7, v8 with IoT integration	Comprehensive evaluation across multiple classes and scenarios

objects, often excluding other essential objects on the road.

Our study aims to address this limitation by including a more comprehensive range of object classes. We use the VSim-AV simulator dataset to capture various events and environmental conditions. Every study we have discussed has used the same dataset. Recall that the dataset has limitations; our research aims to provide a more thorough analysis of self-driving car object detection systems. By considering a broader range of object classes and environmental scenarios, our approach enhances the reliability of autonomous vehicle systems in real-world applications. We compare related work in Table 1 and highlight our study's specific contributions to addressing the gaps found in existing literature.

3. Research Methodology

In this section, we outline the methodology employed to evaluate the performance of YOLOv5, YOLOv7, and YOLOv8 architectures in object detection tasks within AV environments see Fig. 1. In the following subsections, we will elaborate on each step in detail.

3.1. Dataset Generation and Preparation

The dataset utilized in this study originated from simulations of AV environments conducted using the virtual simulation platform VSim-AV [18]. These simulated environments encompass a range of scenarios typically encountered in real-world driving conditions, such as urban streets, highways, and intersections (see Fig. 2) that show some Samples of the VSim-AV dataset. The VSim-AV simulator dataset distinguishes itself from existing datasets in several key ways:

- Data Diversity: VSim-AV offers a diverse dataset of urban, suburban, and rural scenarios, including different angles, conditions, traffic densities, and times of day, providing a comprehensive training and evaluation environment.
- Interactive Elements: VSim-AV, unlike other simulators, incorporates interactive elements like dynamic pedestrians, complex traffic behaviors, and construction zones, enhancing realism and preparing AV systems for real-world scenarios.

By addressing these aspects, VSim-AV aims to fill gaps left by existing simulators, providing a more versatile and detailed dataset to advance the field of autonomous vehicle research. Following the generation of simulated data, the data was uploaded to Roboflow, an online platform specializing in managing and annotating computer vision datasets [26]. The collected dataset comprised images captured within the simulated AV environments, each annotated with bounding boxes indicating four objects: bike, car, human, and road sign. Subsequently, the dataset is divided into three subsets for training, validation, and testing purposes. The training set constituted (82%) of the total images, totaling 4542. At the same time, the validation set consisted of (12%) of the images, totaling 653 images, and the test set encompassed (6%) of the images, totaling 347 images. Before training, the dataset underwent preprocessing to enhance its quality and improve the performance of object detection models. These preprocessing steps included:

- Auto-Orient: Ensures consistent image orientation by correcting deviations, which helps the model learn from uniformly oriented images.
- Grayscale Conversion: Reduces computational complexity by converting images to grayscale, focusing the model on shape and structure rather than color, which is particularly useful for detecting objects in monochrome scenarios.
- Auto-Adjust Contrast: Applies adaptive equalization to enhance object visibility, improving the model's ability to detect features in images with varying contrast levels.
- Tile: Splits images into smaller tiles to diversify training examples, helping the model to learn from different parts of the image and improving its ability to detect small objects.
- Filter Null: Eliminates images with inadequate annotations to ensure high-quality training data, which is crucial for accurate model learning.

Data augmentation techniques are also applied to augment the training dataset and enhance model generalization. These techniques included:

- Hue Adjustment: Varying the hue of images between -15° and +15° enhances the model's robustness to changes in color, such as different lighting conditions or camera settings.

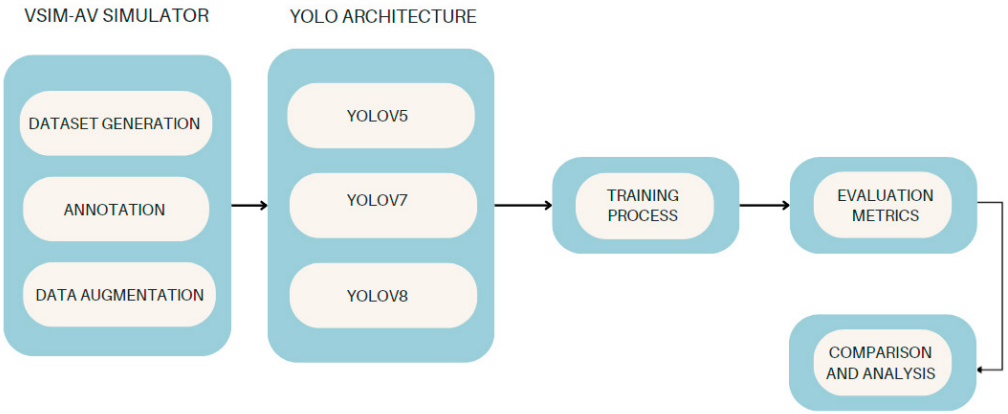

Fig. 1. Methodology flow chart.

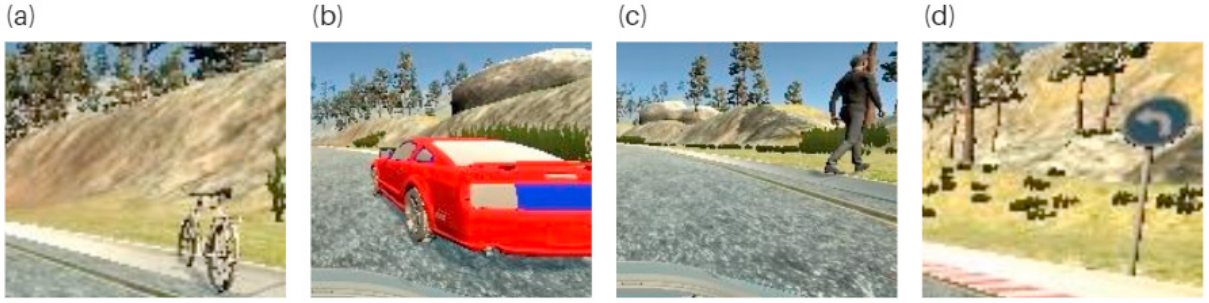

Fig. 2. (a) bike on side road; (b) car on road; (c) human on side road; (d) road-sign.

- Saturation Adjustment: Adjusting saturation between -25% and +25% improves the model's ability to detect objects under varying color intensities, enhancing its adaptability to real-world scenarios.
- Brightness Adjustment: Modifying brightness between -15% and +15% ensures accurate object detection under different illumination levels, making the model more reliable in diverse lighting conditions.
- Exposure Adjustment: Altering exposure between -10% and +10% helps the model handle images with different brightness levels, contributing to its robustness in varying lighting environments.
- Blur: Applying a blur of up to 2.5 pixels helps the model generalize better by simulating motion or out-of-focus effects commonly encountered in real-world images.
- Noise Addition: Introducing random noise of up to 0.1% of pixels improves the model's performance in handling noisy images, enhancing its capability in challenging environments with varying noise levels.

Preprocessing and augmentation techniques are used to address common variations and challenges in real-world AV scenarios. Preprocessing steps like auto-orient, grayscale conversion, and auto-adjust contrast ensure consistent input images, while augmentation techniques introduce controlled variations to improve generalization. These methods also enhance the model's resilience to noise levels, illumination, and orientation variations. The training dataset, consisting of 5542 images, was divided into three subsets: training, validation, and test. These strategies aim to enhance object recognition and classification efficiency in various AV scenarios.

3.2. Selected YOLO Models

In this section, we examine YOLOv5, YOLOv7, and YOLOv8, highlighting their improvements in architecture design, optimization techniques, and performance for object identification. See Table 2 for some of their salient characteristics.

Table 2. Summary of key features in YOLOv5, YOLOv7, and YOLOv8 models.

Model	Architecture	Features
YOLOv5	CSPDarknet53	Lightweight, streamlined architecture
YOLOv7	Deeper, complex	Curriculum learning, progressive resizing
YOLOv8	Highly scalable	Modular design, knowledge distillation

3.2.1. YOLOv5

With its simplified architecture and enhanced performance, YOLOv5 significantly improves the YOLO series [8]. YOLOv5, created by Ultralytics, has a lightweight backbone network built on the CSPDarknet53 architecture, balancing computational efficiency and model complexity [12]. This model uses sophisticated data augmentation techniques to improve resilience and generalization abilities, such as mosaic augmentation, mix-up augmentation, and auto-augmentation. In addition, YOLOv5 optimizes the model architecture, training hyperparameters, and data loading to speed up convergence and real-time inference in different applications [27, 28].

3.2.2. YOLOv7

Building upon its predecessors, YOLOv7 introduces architectural refinements and optimization strategies to enhance performance and robustness further [13]. This model features a more profound and complex architecture, facilitating better feature representation and context modeling. YOLOv7 optimizes the training pipeline through curriculum learning, progressive resizing, and label smoothing, improving generalization and object detection accuracy [8]. Integrating contextual information through feature pyramid networks (FPN) and spatial pyramid pooling (SPP) enables YOLOv7 to capture multiscale features, particularly in challenging scenarios effectively [29, 30].

3.2.3. YOLOv8

It is the most recent iteration of the YOLO architecture series and combines state-of-the-art developments in architectural design and deep learning research [14]. Because of its great flexibility and scalability, this model can handle a wide range of computing resources and application needs. To improve efficiency and performance [31], YOLOv8 uses sophisticated optimization techniques, including knowledge distillation, attention mechanisms, and dynamic routing. Because its modular architecture comprises reusable building components, experimentation and customization for specific use cases are made simple. Across many benchmark datasets and application domains, YOLOv8 exhibits superior performance [32, 33].

3.3. Performance Metrics

In our evaluation of the three YOLO architectures (YOLOv5, v7, and v8) for object detection tasks within AV environments, we utilized several performance metrics to assess each architecture's effectiveness. These metrics include precision, recall, and mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.5.

Precision

It is a metric that measures the accuracy of optimistic predictions made by the model, calculated as the ratio of true positive detections to the total number of positive predictions (true positives and false positives). A higher precision value indicates fewer false positives, implying that the model makes fewer incorrect detections [34]. A definition of precision in mathematics is expressed by the formula shown in Equation 1, where TP stands for the quantity of adequately predicted positive cases. The number of instances correctly forecasted as positive is known as False Positives (FP).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall

This metric, also known as sensitivity, measures the ability of the model to detect all positive instances of a class, calculated as the ratio of true positive detections to the total number of actual positive instances (true positives and

false negatives) [35]. Less false negatives are shown by a more excellent recall score, indicating that the model is capturing the majority of positive occurrences. In mathematical terms, it follows in Equation 2, where FN denotes the number of mistakenly predicted cases and TP denotes the number of adequately predicted positive occurrences.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Mean Average Precision (mAP)

It is a composite metric that evaluates the precision-recall trade-off across all object classes by averaging the precision values at different recall levels for each class. A mAP score closer to 1 indicates better overall performance in object detection tasks [36]. The definition of precision in mathematics is shown in Equation 3, where precision(r) is the precision at a specific recall level (r). A higher mAP denotes superior object detection performance when recall and precision are considered.

$$AP = \int_0^1 precision(r)dr \tag{3}$$

3.4. Execution Environment

The implementation environment for this study relies on NVIDIA GPU acceleration, specifically the Tesla T4 model. Complemented by NVIDIA GPU drivers version 535.104.05 and CUDA version 12.2, this setup provides a robust platform for deep learning tasks. We preserved the original hyperparameters for each model to create a comparison baseline, prioritizing this over seeking optimal results through adjustments or additional techniques. Our code implementations are exclusively crafted in Python and sourced from official repositories to ensure reliability and adherence to standard practices within the field.

3.5. Training and Evaluation

Training and evaluation are fundamental steps in machine learning. They ensure that models are effectively trained and perform well in real-world applications.

3.5.1. YOLOv5

YOLOv5 is a highly efficient and popular version of the YOLO series, known for its balance between speed and accuracy. This model was trained for about 1.740 hours and consisted of 214 layers with 7,030,417 parameters. Data augmentation methods, such as blur and grayscale conversion, were applied along with input images standardized to a size of 416x416 pixels. The Stochastic Gradient Descent (SGD) optimizer, renowned for its simplicity and effectiveness in training neural networks and its role in iteratively updating model parameters, is employed with a learning rate of 0.01, momentum of 0.937, and weight decay of 0.0005. The model leveraged the AutoAnchor feature for anchor determination, achieving a Best Possible Recall (BPR) of 0.992. After validation, the model demonstrated strong performance with a mean Average Precision (mAP@0.5) of 0.94, recall of 0.933, and an overall precision of 0.899. Additionally, YOLOv5 showed an impressive inference speed of 1.3 milliseconds per image.

3.5.2. YOLOv7

YOLOv7 is an advanced version with numerous enhancements to improve performance. This model consists of 415 layers with 37,212,738 parameters and rained for approximately 1.638 hours. Input images standardized to 640x640 pixels. The optimizer used was Stochastic Gradient Descent (SGD) with scaled weight decay to balance effective parameter updates and regularization for optimal training performance and generalization with a learning rate of 0.01, a momentum of 0.937, and a weight decay of 0.0005. The AutoAnchor functionality provided a Best Possible Recall (BPR) of 0.9835. Despite these enhancements, the model exhibited a lower overall precision of 0.513, a recall of 0.429, and a mean Average Precision (mAP@0.5) of 0.441. Its inference speed was 4.93 milliseconds per image. These results indicate areas for improvement in the training or tuning process for YOLOv7.

3.5.3. YOLOv8

YOLOv8 represents the latest iteration in the YOLO series, focusing on enhanced accuracy and efficiency. With 11,137,148 parameters spread over 225 layers, this model took around 0.781 hours to train. It utilized the AdamW optimizer, known for its ability to handle sparse gradients and large parameter spaces, with a learning rate of 0.00125.

Data augmentation techniques are applied to improve robustness, including random scaling, rotation, and flipping. The model successfully passed Automatic Mixed Precision (AMP) testing, optimizing performance and memory usage. Input images standardized to 800x800 pixels. With an overall precision of 0.918, recall of 0.903, and a mean Average Precision (mAP@0.5) of 0.927 after validation, YOLOv8 demonstrated excellent performance. Its inference time was 3.3 milliseconds per image, slightly slower than YOLOv5 but still competitive (see Table 3).

Table 3. Summary of parameters and inference speed for YOLOv5, v7, and v8 models.

Model	Layers	Parameters	Optimizer	No. of epochs	Learning Rate	Input Size	Inference Speed
YOLOv5	214	7,030,417	SGD	50	0.01	416x416 pixels	1.3 ms/image
YOLOv7	415	37,212,738	SGD with scaled weight decay	50	0.01	640x640 pixels	4.93 ms/image
YOLOv8	225	11,137,148	AdamW	50	0.00125	800x800 pixels	3.3 ms/image

4. Results and Discussion

In this section, we present the results of our evaluation of three YOLO architectures, YOLOv5, YOLOv7, and YOLOv8, in object detection tasks within AV environments. The performance metrics calculated for each architecture across multiple object classes, including precision, recall, and mean Average Precision (mAP) at an Intersection over a Union (IoU) threshold of 0.5. We will also discuss the limitations observed during the evaluation process.

4.1. Testing Results

Our evaluation showed that the YOLO architectures performed noticeably differently from one another. With an overall mAP@0.5 score of 0.94, YOLOv5 demonstrated the best precision and recall across object classes, including Bikes, Cars, Humans, and Road signs. YOLOv8 closely followed, achieving overall mAP@0.5 of 0.927 and excellent precision-recall couples for every object type. On the other hand, YOLOv7 performed noticeably less effectively than expected, with an overall mAP@0.5 of 0.441 and noticeably poor precision-recall values for most object classes. Car identification consistently showed excellent recall and accuracy scores across all YOLO architectures. Likewise, human detection showed comparatively good precision and recall ratings across all designs. Table 4 shows notable differences in the recognition of other object classes, such as Bike and Road-Sign, where YOLOv8 and YOLOv5 performed significantly better than YOLOv7.Furthermore, concerning inference speed, YOLOv5 showed an inference speed of 1.3 milliseconds for each image, but YOLOv8 showed an inference speed of 3.3 milliseconds for each image. YOLOv7's inference speed per image was 4.93 milliseconds, slower than YOLOv8 and YOLOv5. Furthermore, concerning inference speed, YOLOv5 showed an inference speed of 1.3 milliseconds for each image, but YOLOv8 showed an inference speed of 3.3 milliseconds for each image. YOLOv7's inference speed per image was 4.93 milliseconds, slower than YOLOv8 and YOLOv5.

The superior performance of YOLOv5 and YOLOv8 can be attributed to their sophisticated architectural designs, feature extraction methods, and optimization techniques. These designs have been enhanced to yield more dependable object-detecting capabilities by increasing detection efficiency and accuracy. YOLOv5 benefits from improvements such as the Focus layer for efficient feature extraction, AutoAnchor for dataset-specific optimization, and hyperparameter evolution, all of which contribute to its high performance and adaptability to various object detection tasks [12]. YOLOv8, in particular, benefits from the latest advancements in deep learning, such as the AdamW optimizer and automatic mixed precision (AMP), contributing to its high performance despite a larger input size [14]. On the other hand, subpar optimization methods or an antiquated design can cause YOLOv7's performance constraints. The observed performance disparities underscore the significance of architectural developments in deep learning models for object identification tasks, particularly in AV contexts where precise and timely detection is critical to maintaining security. Although YOLOv8 and YOLOv5 show encouraging results, more investigation is required to fix YOLO's shortcomings and investigate novel ideas for improving object identification in AV settings. This includes exploring more diverse datasets, integrating advanced augmentation techniques, and refining hyperparameters to enhance model robustness and accuracy further. The performance of models trained on the VSim-AV simulator dataset demonstrates notable strengths compared to those trained on real-world datasets such as:

- Controlled Environment: The VSim-AV simulator allows for precise control over various environmental variables such as lighting, weather, and traffic density, providing a diverse and comprehensive set of training scenarios. This can result in well-prepared models for a wide range of conditions.

- Data Diversity: The VSim-AV dataset includes a broader range of object types and scenarios that might be underrepresented in real-world datasets. This variety can enhance the model's ability to generalize to different situations.
- Quality of Annotations: Simulated datasets can have perfectly accurate annotations without human error, potentially leading to better training results.

Overall, our results highlight the importance of choosing suitable deep-learning architectures for object identification in AV contexts and offer insightful information for creating AV systems that are more dependable and resilient.

Table 4. Performance results of different versions of YOLO in AV object detection.

	mAP@0.5			Precision			Recall		
	YOLOv5	YOLOv7	YOLOv8	YOLOv5	YOLOv7	YOLOv8	YOLOv5	YOLOv7	YOLOv8
Bike	**0.882**	0.408	0.836	0.855	0.496	**0.887**	**0.844**	0.381	0.75
Car	0.963	0.46	**0.976**	0.947	0.555	**0.954**	**0.967**	0.426	**0.967**
Human	**0.954**	0.63	0.939	0.847	0.556	**0.872**	**0.965**	0.594	0.947
Road- Sign	**0.96**	0.265	0.959	0.946	0.443	**0.958**	**0.955**	0.315	0.948
All	**0.94**	0.441	0.927	0.899	0.513	**0.918**	**0.933**	0.429	0.903

4.2. Limitations

The YOLO models, like YOLOv5, YOLOv7, and YOLOv8, excel in AV object recognition but have limitations. They struggle in challenging conditions like rain, fog, or low light, reducing accuracy and higher false positives. They also face difficulties in crowded or occluded scenarios and have high computational resource requirements. Addressing these issues requires coordinated efforts to improve model architectures, training techniques, datasets, and real-world testing. Collaboration among researchers, industry stakeholders, policymakers, and regulatory agencies is essential for responsible integration into AV systems.

5. Conclusion and Future Directions

Accurate object detection is critical to autonomous vehicles' safe and efficient operation (AVs). We compared the item identification tasks in the AV settings, focusing on the YOLOv5, YOLOv7, and YOLOv8 designs. We discovered differences in performance, with YOLOv5 demonstrating quicker inference speed than YOLOv8 despite both models exhibiting vital accuracy and recall across various item classes. Additionally, YOLOv7 exhibited considerably reduced precision-recall levels, highlighting the importance of selecting appropriate deep-learning architectures for object identification in AV systems. These performance discrepancies underscore the need for further research and development in critical areas. One such area is the enhancement of Model Architectures, leveraging attention mechanisms and multiscale feature fusion for improved robustness. Moreover, integrating these models into Federated Learning frameworks holds promise for decentralizing the learning process, enhancing privacy and scalability in AV applications, and addressing the shortcomings of centralized systems. Furthermore, integrating IoT technologies and methodologies such as self-supervised learning and AI-driven analytics can further enhance object detection capabilities in AV environments. Our work lays the groundwork for optimizing detection effectiveness, fortifying AV security, and realizing the full promise of autonomous driving technology by advancing computer vision for object detection in AV environments.

References

[1] Balasubramaniam, Abhishek, and Sudeep Pasricha. "Object detection in autonomous vehicles: Status and open challenges." arXiv preprint arXiv:2201.07706 (2022).

[2] Haris, Malik, and Adam Glowacz. "Road object detection: A comparative study of deep learning-based algorithms." Electronics 10, no. 16 (2021): 1932.

[3] Jiao, Licheng, et al. "A survey of deep learning-based object detection." IEEE Access 7 (2019): 128837-128868.

[4] Aziz, Lubna, et al. "Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review." IEEE Access 8 (2020): 170461-170495.

[5] Jiang, Peiyuan, et al. "A Review of Yolo algorithm developments." Procedia Computer Science 199 (2022): 1066-1073.

[6] Kang, Chang Ho, and Sun Young Kim. "Real-time object detection and segmentation technology: an analysis of the YOLO algorithm." JMST Advances 5, no. 2 (2023): 69-76.

[7] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition (2016).
[8] Terven, Juan, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS." Machine Learning and Knowledge Extraction 5, no. 4 (2023): 1680-1716.
[9] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[10] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[11] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).
[12] Ultralytics. "Comprehensive Guide to Ultralytics YOLOv5." Retrieved from https://docs.ultralytics.com/yolov5/, Last accessed on February 20, 2024.
[13] Solawetz, J. "What is YOLOv7? A Complete Guide." Roboflow Blog. They were retrieved from https://blog.roboflow.com/yolov7-breakdown/, Last accessed on February 20, 2024.
[14] Solawetz, J. "What is YOLOv8? The Ultimate Guide." Roboflow Blog. They were retrieved from https://blog.roboflow.com/whats-new-in-yolov8/, Last accessed on February 20, 2024.
[15] Jocher, Glenn, et al. "ultralytics/yolov5: v3. 0." Zenodo (2020).
[16] Olorunshola, Oluwaseyi Ezekiel, Martins Ekata Irhebhude, and Abraham Eseoghene Evwiekpaefe. "A Comparative Study of YOLOv5 and YOLOv7 Object Detection Algorithms." Journal of Computing and Social Informatics 2, no. 1 (2023): 1-12.
[17] Sohan, Mupparaju, et al. "A Review on YOLOv8 and Its Advancements." International Conference on Data Intelligence and Cognitive Informatics. Springer, Singapore, 2024.
[18] Meftah, Leila Haj, and Rafik Braham. "VSim-AV: A Virtual Simulation Platform for Autonomous Vehicles." International Conference on Intelligent Systems Design and Applications. Cham: Springer International Publishing, 2021.
[19] Faisal, Mustafa M., et al. "Object Detection and Distance Measurement Using AI." 2021 14th International Conference on Developments in eSystems Engineering (DeSE). IEEE, 2021.
[20] Dazlee, Nurin Mirza Afiqah Andrie, et al. "Object detection for autonomous vehicles with sensor-based technology using YOLO." International Journal of Intelligent Systems and Applications in Engineering 10, no. 1 (2022): 129-134.
[21] Mohanapriya, S., et al. "Object and lane detection for autonomous vehicle using YOLO V3 algorithm." AIP Conference Proceedings 2387, no. 1 (2021): 020005.
[22] Sarda, Abhishek, Shubhra Dixit, and Anupama Bhan. "Object detection for autonomous driving using YOLO [you only look once] algorithm." 2021 Third international conference on intelligent communication technologies and virtual mobile networks (ICICV). IEEE, 2021.
[23] Razali, Haziq, Taylor Mordan, and Alexandre Alahi. "Pedestrian intention prediction: A convolutional bottom-up multi-task approach." Transportation research part C: emerging technologies 130 (2021): 103259.
[24] Jia, Xiang, et al. "Fast and accurate object detector for autonomous driving based on improved YOLOv5." Scientific reports 13, no. 1 (2023): 1-13.
[25] Choudhury, Sanjoy, et al. "A Hybrid CNN Real-Time Object Identification and Classification Approach for Autonomous Vehicles." Intelligent Systems: Proceedings of ICMIB 2021. Singapore: Springer Nature Singapore, 2022, pp. 485-497.
[26] Roboflow. "Give your software the power to see objects in images and video." Retrieved from https://roboflow.com/, Last accessed on February 20, 2024.
[27] Mathew, Midhun P., and Therese Yamuna Mahesh. "Leaf-based disease detection in bell pepper plant using YOLO v5." Signal, Image and Video Processing (2022): 1-7.
[28] Zendehdel, Niloofar, Haodong Chen, and Ming C. Leu. "Real-time tool detection in smart manufacturing using You-Only-Look-Once (YOLO) v5." Manufacturing Letters 35 (2023): 1052-1059.
[29] Gallo, Ignazio, et al. "Deep object detection of crop weeds: Performance of YOLOv7 on a real case dataset from UAV images." Remote Sensing 15, no. 2 (2023): 539.
[30] Wang, C. Y., A. Bochkovskiy, and H. Y. M. Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors—arXiv 2022." arXiv preprint arXiv:2207.02696 (2022).
[31] Ultralytics. "Home. Ultralytics YOLOv8 Docs." Retrieved from https://docs.ultralytics.com/, Last accessed on February 20, 2024.
[32] Talaat, Fatma M., and Hanaa ZainEldin. "An improved fire detection approach based on YOLO-v8 for smart cities." Neural Computing and Applications 35, no. 28 (2023): 20939-20954.
[33] Yang, Guoliang, et al. "A lightweight YOLOv8 tomato detection algorithm combining feature enhancement and attention." Agronomy 13, no. 7 (2023): 1824.
[34] Akbarnezhad, E. "YOLOv8 Projects #1 "Metrics, Loss Functions, Data Formats, and Beyond." Retrieved from https://www.linkedin.com/pulse/yolov8-projects-1-metrics-loss-functions-data-formats-akbarnezhad, Last accessed on February 20, 2024.
[35] Ultralytics. "YOLO Performance Metrics." Ultralytics YOLOv8 Docs. Retrieved from https://docs.ultralytics.com/guides/yolo-performance-metrics/, Last accessed on February 20, 2024.
[36] Cochard, D. "mAP: Evaluation metric for object detection models." Medium. Retrieved from https://medium.com/axinc-ai/map-evaluation-metric-of-object-detection-model-dd20e2dc2472, Last accessed on February 20, 2024.