SEBHA UNIVERSITY CONFERENCE PROCEEDINGS, VOL.04 NO. 3 2025
DOI: 10.51984/SUCP.V4I3.4004

وقائع مؤتمرات جامعة سبها
Sebha University Conference Proceedings

Confrence Proceeeding homepage: http://www.sebhau.edu.ly/journal/CAS

Performance Evaluation of Lightweight YOLOv5n Models in CARLA Simulator for Autonomous Driving Applications

Bader N. Awedat

Alqasr International University, Qasr Bin Ghashir, Libya

Keywords:

Object detection
YOLOv5
CARLA
autonomous vehicles
lightweight models.

ABSTRACT

Real-time object detection is critical for autonomous driving systems, especially in realistic simulation environments like the CARLA platform. This study aims to evaluate the performance of lightweight models (YOLOv5n) in detecting complex objects, including multi-colored traffic signals, vehicles, and pedestrians, within a CARLA simulation. We trained two models using CARLA-derived datasets to assess the trade-off between accuracy and computational efficiency. Model 1 (YOLOv5n-1864-T): This model was trained on 1,864 training samples. It demonstrated superior accuracy (mAP50 = 0.935) and a faster inference speed (0.022-0.025 seconds per frame). Model 2 (YOLOv5n-1600-TV): This model was trained on 1,600 training samples along with 480 validation and test samples. It achieved better generalization (mAP50 = 0.942) with balanced confidence scores (0.75-0.96), despite a slightly slower inference speed (0.04-0.07 seconds). During inference tests on 8 dynamic CARLA frames, both models achieved high performance (F1-Score: 0.92-0.94). Model 1 excelled at detecting red traffic signals, while Model 2 showed greater adaptability to complex pedestrian angles. The results confirm that lightweight models can compete with heavier models while maintaining high computational efficiency (4.2 GFLOPs). The study recommends integrating validation data with expanded training sets to improve generalization without sacrificing speed, providing a practical framework for deploying such models in simulation-based autonomous driving systems.

تقييم أداء نماذج YOLOv5n خفيفة الوزن في محاكي CARLA لتطبيقات القيادة الذاتية

بدر نجيب عويدات

جامعة القصر الدولية، قصر بن غشير، ليبيا

الكلمات المفتاحية:

اكتشاف الأجسام
YOLOv5
CARLA
المركبات ذاتية القيادة
النماذج خفيفة الوزن.

الملخص

يُعد اكتشاف الأجسام في الوقت الفعلي أمرًا بالغ الأهمية لأنظمة القيادة الذاتية، لا سيما في بيئات المحاكاة الواقعية مثل منصة CARLA. تهدف هذه الدراسة إلى تقييم أداء النماذج خفيفة الوزن (YOLOv5n) في اكتشاف الأجسام المعقدة، بما في ذلك إشارات المرور متعددة الألوان والمركبات والمشاة، ضمن محاكاة CARLA. لقد قمنا بتدريب نموذجين باستخدام مجموعات بيانات مشتقة من CARLA لتقييم المفاضلة بين الدقة والكفاءة الحسابية. النموذج الأول (YOLOv5n-1864-T): تم تدريب هذا النموذج على 1,864 عينة تدريب. وقد أظهر دقة فائقة (mAP50=0.935) وسرعة استدلال أسرع (0.022-0.025 ثانية لكل إطار). النموذج الثاني (YOLOv5n-1600-TV): تم تدريب هذا النموذج على 1,600 عينة تدريب بالإضافة إلى 480 عينة تحقق واختبار. وقد حقق قدرة تعميم أفضل (mAP50=0.942) مع درجات ثقة متوازنة (0.75-0.96)، على الرغم من أن سرعة الاستدلال كانت أبطأ قليلاً (0.04-0.07 ثانية). خلال اختبارات الاستدلال على 8 إطارات ديناميكية منCARLA ، حقق كلا النموذجين أداءً عاليًا (F1-Score: 0.92-0.94). تفوق النموذج الأول في اكتشاف إشارات المرور الحمراء، بينما أظهر النموذج الثاني قدرة أكبر على التكيف مع زوايا المشاة المعقدة. تؤكد النتائج أن النماذج خفيفة الوزن يمكن أن تنافس النماذج الأكبر حجمًا مع الحفاظ على كفاءة حسابية عالية (4.2 GFLOPs). توصي الدراسة بدمج بيانات التحقق مع مجموعات التدريب الموسعة لتحسين التعميم دون

*Corresponding author:
E-mail addresses: bader_najep@yahoo.com
Article History : Received 20 February 2025 - Received in revised form 01 September 2025 - Accepted 07 October 2025

التضحية بالسرعة، مما يوفر إطار عمل عمليًا لنشر مثل هذه النماذج في أنظمة القيادة الذاتية القائمة على المحاكاة.

1. Introduction

Autonomous driving systems are considered one of the most revolutionary applications of artificial intelligence, fundamentally relying on their ability to interpret the surrounding environment through precise and effective real-time object detection. Object detection serves as the cornerstone of these systems, enabling the identification of vehicles, pedestrians, traffic signals, and other obstacles, thereby contributing to safe and immediate decision-making [1]. However, these systems face significant challenges in achieving a balance between high accuracy and computational efficiency, particularly when deployed on resource-constrained embedded systems, such as Arduino boards or microcontrollers, which are widely used in prototypes of self-driving cars [2].

The CARLA environment serves as a virtual laboratory for computer vision, allowing for the testing of computer vision models under near-realistic conditions. Advanced simulation platforms like CARLA (Car Learning to Act) offer open-source environments that provide dynamic urban settings with varying lighting conditions, complex traffic, and intelligent traffic signals [3]. CARLA enables researchers to generate rich and diverse training data without the risks associated with high costs or the logistical challenges of collecting real-world data. According to a study [1], CARLA is considered a vital tool for developing autonomous driving algorithms, as it simulates critical scenarios that rarely occur in real life, thereby enhancing the models' generalization capabilities.

In recent years, deep learning models such as YOLO (You Only Look Once) have made significant strides in the accuracy and efficiency of object detection. The YOLOv5 version (2020) is regarded as one of the most balanced models in terms of speed and accuracy, thanks to its customizable modular architecture [4]. However, most industrial applications are increasingly focusing on lightweight models, such as YOLOv5n (the nano version), which reduce the number of parameters and floating-point operations (FLOPs), making them suitable for deployment on resource-limited devices [5]. A study [2] indicates that simplifying models without sacrificing accuracy is a major challenge, especially in critical applications like autonomous vehicles, where the system requires a response time of less than 100 milliseconds to avoid accidents.

Despite advancements in computer vision models, transferring models trained in virtual environments (such as CARLA) to real devices remains an engineering challenge. Embedded devices like Arduino or Raspberry Pi face constraints in RAM and processing power, necessitating the use of optimized models to reduce power consumption and maintain temporal performance [6]. A study [5] shows that models like YOLOv5n can be quantized and optimized to fit these devices, but they require careful evaluation of how accuracy is affected by varying lighting conditions and angles, which environments like CARLA provide [3].

This study aims to achieve the following:

1. Evaluate the performance of the lightweight YOLOv5n model in object detection within the dynamic CARLA environment [3].
2. Compare the effect of training data size (1600 vs. 1864 samples) and the inclusion of validation data on the model's generalization [4].
3. Analyze the feasibility of deploying the model on embedded devices by measuring computational efficiency (such as inference time and number of operations) [6].

This paper presents a tangible contribution by combining advanced simulation techniques with the optimization of lightweight models, thereby opening new avenues for the integration of artificial intelligence systems in resource-constrained embedded systems [2].

2. Related work

1. Awedat, B. N. (2024) investigated the enhancement of object detection accuracy (e.g., bicycles, vehicles, traffic signs, and pedestrians) in autonomous driving systems using **YOLOv5s** within the **CARLA simulator**. This approach was proposed to overcome the high costs and technical complexities associated with real-world systems. The study utilized a dataset of 1,560 images (1,120 for training, 160 for testing, and 320 for validation) and employed YOLOv5s with default hyperparameters (learning rate: 0.01, 100 epochs, 640×640 resolution). Key evaluation metrics included Precision (P), Recall (R), mAP@50, and mAP@50-95.

As shown in **Table 1**, the model achieved a precision of 0.898 on test data and 0.891 on validation data, with an mAP@50 of 0.900 and 0.880, respectively. Notably, the "Red Traffic Signal" class outperformed others (mAP@50: 0.994), while the "Bicycle" class exhibited lower performance (mAP@50: 0.773), likely due to data imbalance or shape complexity.

Table 1. Model Performance on Test and Validation Data.

Metric	Test Data	Validation Data
Precision (P)	0.898	0.891
Recall (R)	0.827	0.801
mAP@50	0.900	0.880
mAP@50-95	0.583	0.542

Strengths and Limitations:

The study highlighted CARLA's flexibility in simulating diverse environmental conditions (e.g., weather, lighting).

YOLOv5s demonstrated a balance between speed (11.8 ms/image) and accuracy (mAP@50: 0.900).

However, reliance on simulated data and a 7% drop in mAP@50-95 between training and validation suggested potential overfitting.

Contribution to the Field:

This work underscores the potential of simulation-driven approaches for cost-effective autonomous system development, though hybrid datasets (simulated + real-world) are recommended to improve generalizability [7].

2. The study by Awedat, B. N. (2024), titled *"Object detection using artificial intelligence in autonomous vehicles"*, aimed to improve the accuracy of object detection (e.g., bicycles, vehicles, traffic signs, and pedestrians) in autonomous driving systems using the **YOLOv5s** model within the **CARLA simulation environment**, with a focus on balancing cost-effectiveness and computational efficiency. A dataset of 1,864 images was used, divided into 1,600 (training) and 264 (testing). The model, YOLOv5s, was configured with the following parameters: learning rate = 0.01, training epochs = 150. Evaluation metrics included Precision (P), Recall (R), mean Average Precision at IoU = 50% (mAP@50), and mean Average Precision across IoU thresholds from 50% to 95% (mAP@50-95). Training was conducted on the Google Colab platform with a T4 GPU and 16GB RAM.

Table 2. Model Performance on Training and Test Data.

Metric	Training Data	Test Data
Precision (P)	*0.934*	*0.930*
Recall (R)	*0.908*	*0.892*
mAP@50	*0.935*	*0.930*
mAP@50-95	*0.689*	*0.675*

Table 3. Class-Specific Performance.

Class	Precision (P)	Recall (R)
Red Traffic Sign	*0.979*	*0.983*
Pedestrians	*1.0*	*0.973*
Bicycles	*0.731*	*0.857*
Motorcycles	*0.990*	*0.750*

Strengths:

1. **Efficiency of the CARLA Environment**: Customizable settings (e.g., weather, lighting, sensor configurations). Support for creating complex simulation scenarios at low cost.

2. Model Performance: High inference speed (6.6 milliseconds per image). High accuracy (mAP@50: 0.93).

Limitations

Underperformance in Specific Classes: Reduced precision for bicycles (P: 0.731) due to shape complexity or data imbalance. Performance declined in extreme weather conditions (e.g., heavy rain).

Reliance on Simulated Data: Limited generalizability to real-world environments.

The study demonstrated the ability of the **YOLOv5s** model to achieve high-precision object detection within the **CARLA** environment, while highlighting challenges requiring resolution (e.g., improving

performance for small classes). These results provide a practical framework for developing more efficient and adaptable autonomous driving systems [8].

3. In the study by Alahdal, N. M. et al. (2024), titled Real-time object detection in autonomous vehicles with YOLO This study aims to evaluate the performance of YOLO models (v5, v7, v8) in object detection tasks for autonomous vehicles (AVs), focusing on classes such as cars, pedestrians, bicycles, and traffic signs. The researchers used the VSim-AV simulated dataset, characterized by diverse scenarios (urban, suburban, rural) and dynamic interactions, such as complex pedestrian movement. In data preparation, preprocessing included auto-orientation correction, grayscale conversion, and contrast enhancement. Augmentation involved adjustments to lighting, saturation, blur, and noise addition to improve model generalization. The data was split into 82% for training, 12% for validation, and 6% for testing.

The models used were:

YOLOv5: Lightweight architecture (CSPDarknet53) with enhancements such as AutoAnchor and Focus layers.

YOLOv7: Deeper architecture with techniques like Curriculum Learning.

YOLOv8: Modular design with improvements in attention mechanisms and dynamic routing.

Performance metrics: Precision, Recall, and mean Average Precision (mAP@0.5).

Table 4. Model Performance.

Class/Metric	YOLOv5	YOLOv7	YOLOv8
Bike			
- mAP@0.5	*0.882*	*0.408*	*0.836*
- Precision	*0.855*	*0.496*	*0.887*
- Recall	*0.844*	*0.381*	*0.750*
Car			
- mAP@0.5	*0.963*	*0.460*	*0.976*
- Precision	*0.947*	*0.555*	*0.954*
- Recall	*0.967*	*0.426*	*0.967*
Human			
- mAP@0.5	*0.954*	*0.630*	*0.939*
- Precision	*0.847*	*0.556*	*0.872*
- Recall	*0.965*	*0.594*	*0.947*
Road-Sign			
- mAP@0.5	*0.960*	*0.265*	*0.959*
- Precision	*0.946*	*0.443*	*0.958*
- Recall	*0.955*	*0.315*	*0.948*
Overall (All)			
- mAP@0.5	*0.940*	*0.441*	*0.927*
- Precision	*0.899*	*0.513*	*0.918*
- Recall	*0.933*	*0.429*	*0.903*

YOLOv5 and YOLOv8 outperformed: YOLOv5 achieved the highest mAP@0.5 (0.94) with an inference speed of 1.3 ms/image. YOLOv8 achieved mAP@0.5 (0.927) with a better balance between precision and recall.

Weak performance of YOLOv7: It recorded the lowest mAP@0.5 (0.441) due to architectural complexity and insufficient optimization.

Class-specific performance: The **car** class excelled across all models (mAP@0.5 ≈ 0.96). The **traffic sign** class faced challenges, particularly in YOLOv7 (mAP@0.5 = 0.265).

Challenges and limitations:

1. Harsh environmental conditions: Reduced accuracy in rainy or foggy scenarios.

2. Crowding and occlusion: Difficulty in detecting overlapping objects.

3. Reliance on simulated data: Incomplete alignment with real-world scenarios.

This study serves as a robust reference for understanding YOLO model performance in AV contexts, particularly with advanced simulated datasets. The study concluded that:

• **YOLOv5** is optimal for applications requiring high speed (1.3 ms/image) and accuracy.

• **YOLOv8** balances precision and recall, making it ideal for critical applications.

• **YOLOv7** requires architectural or training improvements to enhance performance[9].

4. In the study by Dontabhaktuni, J., & Peddakrishna, S. (2024) titled *Performance Evaluation of YOLOv5-based Custom Object Detection Model for Campus-Specific Scenario*, 5,246 images were collected from various locations within the university campus, expanded to 5,264 samples after applying data augmentation techniques (e.g., rotation, cropping, brightness adjustment). The dataset included 53 classes of common objects in campus environments, such as pedestrians, vehicles, buildings, obstacles, and traffic signs. Stratified sampling was employed to ensure balanced class distribution across training (70%), validation (20%), and testing (10%) sets, enhancing the model's generalizability. Data augmentation techniques such as rotation, cropping, brightness adjustment, and horizontal flipping were applied.

Different batch sizes (12, 16, 32) were tested, with a batch size of 32 demonstrating the best performance (precision: 0.862, recall: 0.761). The SGD optimizer was used with a learning rate of 0.01 and weight decay of 0.0005. The coordinate loss (Coord Loss) and classification loss (Class Loss) were integrated with modifications to improve bounding box localization accuracy.

Table 5. Model Performance.

Criterion/Comparison	Value or Description
Overall Model Performance	
- Precision	*0.851*
- Recall	*0.831*
- mAP@50	*0.843*
- mAP@50-95	*0.534*
Batch Size Impact	
- Batch Size 32	*Precision: 0.862, Recall: 0.761, mAP@50: 0.821, mAP@50-95: 0.504*
- Batch Size 16	*Precision: 0.826, Recall: 0.775, mAP@50: 0.843, mAP@50-95: 0.521*
- Batch Size 12	*Precision: 0.810, Recall: 0.705, mAP@50: 0.732, mAP@50-95: 0.446*
Training Time	
- 50 Epochs	*1.377 hours*
- 100 Epochs	*2.568 hours*
- 150 Epochs	*3.738 hours*

Challenges and Limitations:

1. Reliance on Simulated Data: Results may not fully reflect real-world challenges due to the use of limited campus-specific data.

2. Underperformance in Small Classes: Reduced accuracy in specific classes (e.g., bicycles) due to data imbalance.

3. Computational Complexity: The model's 7 million parameters and 214 layers increased training and deployment costs.

The study demonstrates the efficacy of YOLOv5s for object detection in campus environments, achieving a balance between accuracy and speed. The results provide a foundation for developing safer autonomous transportation systems in universities, with recommendations to integrate real-world data to improve generalizability. The model achieved a training time of 2.568 hours for 100 epochs using a T4 GPU and 16GB RAM, indicating suitability for near real-time applications. Modifications focused on optimizing performance for campus-specific contexts through data adaptation and hyperparameter tuning while maintaining computational efficiency [10].

5. The evolution of object detection for real-time applications has seen a shift from two-stage to highly efficient single-stage detectors. The Single Shot MultiBox Detector (SSD), proposed by Liu et al. (2016), was a landmark model in this field. It achieved a notable balance between speed and accuracy by performing both localization and classification in a single forward pass. A key architectural innovation of SSD was the use of multi-scale feature maps from different layers of its backbone network (VGG-16) to detect objects of various sizes, a strategy that significantly improved its ability to handle small objects.

Building on the foundation laid by such pioneering work, the YOLO (You Only Look Once) family of models, particularly YOLOv5, has advanced the state-of-the-art. While both SSD and YOLOv5 share the single-stage detection philosophy, YOLOv5 integrates more sophisticated and efficient architectures. It utilizes a CSPDarknet backbone and a PANet-based neck to enhance feature fusion, resulting in superior performance, especially in terms of inference speed and detection accuracy across various object scales. Our study, by

evaluating two lightweight YOLOv5n configurations, demonstrates how these modern architectures surpass the performance of earlier models like SSD, offering a more robust and faster solution for real-time autonomous driving applications within the CARLA simulation environment [26].

Table 6. Key Results of SSD Study (2016)

KEY METRIC	RESULT ACHIEVED	DATASET
ACCURACY (MAP)	74.3%	Pascal VOC 2007
SPEED (FPS)	59 frames per second (FPS)	N/A (Based on a 300x300 input resolution)
SUMMARY OF CONTRIBUTION	Achieved real-time speed with competitive accuracy, outperforming two-stage detectors like Faster R-CNN in speed.	N/A
MULTI-SCALE DETECTION	Demonstrated effective detection of objects of varying sizes using multi-scale feature maps.	N/A

6. The paper "EfficientDet: Scalable and Efficient Object Detection" by Tan et al. (2020) represents a significant advancement in field of lightweight object detection. The study's primary contribution is a novel approach to building highly efficient and scalable models by introducing two key innovations. First, it proposes a Bi-directional Feature Pyramid Network (BiFPN), a weighted multi-scale feature fusion network that enhances information flow between different feature levels. This design addresses the limitation of previous networks by allowing for more effective cross-scale connections. Second, the paper introduces a compound scaling method that uniformly scales the network's depth, width, and input resolution. This systematic approach ensures that the model's capacity grows in a balanced way, leading to superior accuracy and efficiency across a wide range of computational constraints. The EfficientDet family of models, developed using these innovations, has achieved state-of-the-art performance with significantly fewer parameters and FLOPs compared to prior detectors [27].

Table 7. Key Results of EfficientDet Study

KEY RESULT	ACHIEVED VALUE	DATASET
ACCURACY (MAP)	55.1% AP (for EfficientDet-D7)	COCO test-dev
SPEED/EFFICIENCY	4-9x smaller and 13-42x fewer FLOPs than prior detectors	N/A
ARCHITECTURAL INNOVATION	Introduction of BiFPN and Compound Scaling	N/A

3. CARLA (Car Learning to Act)

CARLA is an open-source simulator designed specifically for research in autonomous driving. It has been built from the ground up to facilitate the development, training, and validation of autonomous urban driving systems. Beyond its open-source code and protocols, CARLA offers a range of open digital assets, including urban layouts, buildings, and vehicles, which have been created for this purpose and are available for free use [1].

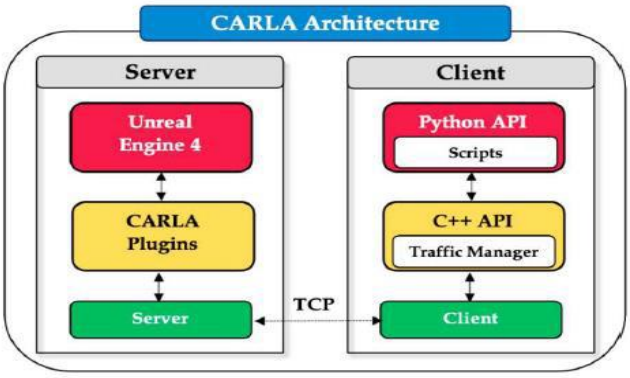


Figure. 1: CARLA client-server architecture [11].

4. Autonomous Vehicle System Architecture in CARLA Environment:

1. Global Planner: Global Planner is tasked with determining overall journey path based on designated destination and prevailing road conditions.

2. Waypoint Position Decision: This component governs the vehicle's direction and identifies specific locations along the route.

3. Obstacle Detection and Perception: It detects obstacles within the environment, enhancing the vehicle's awareness of its surroundings.

4. Sensors: System comprises cameras, Global Positioning System (GPS) ،speed measurement units (Encoders) ،and wireless communication technology (G4).

5. Self-Localization: This functionality enables the vehicle to accurately ascertain its position within environment.

6. Motion Control: It manages the vehicle's operations, regulating its movements based on the established decisions.

7. Local Planner: Local Planner determines a short-term path to assist in navigating immediate obstacles.

8. Collision Avoidance: This component directs vehicle's movements to prevent collisions with obstacles.

These components work in unison to create a self-driving system within the CARLA environment, allowing vehicle to autonomously control itself and adjust its movements according to the surrounding conditions and defined objectives.

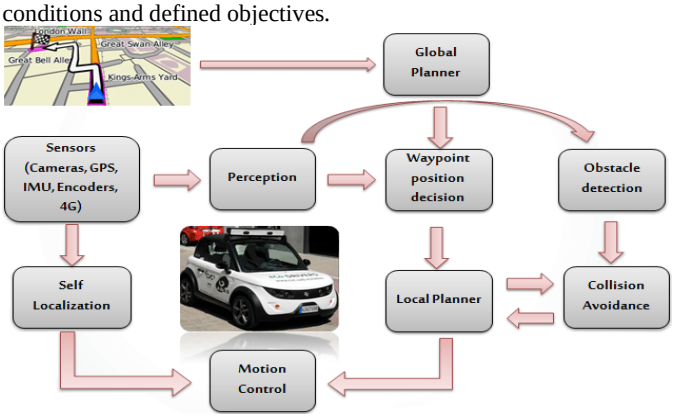


Figure. 2: System architecture of autonomous vehicle [12].

4. Weather Presets: In CARLA, users can customize weather and lighting conditions by selecting from a range of predefined settings. To select a specific preset, modify "WeatherId" key in configuration file "CarlaSettings.ini." Available presets include: 0 – Default, 1 – Clear Noon, 2 – Cloudy Noon, 3 – Wet Noon, 4 – Wet Cloudy Noon, 5 – Mid Rainy Noon, 6 – Hard Rain Noon, 7 – Soft Rain Noon, 8 – Clear Sunset, 9 – Cloudy Sunset, 10 – Wet Sunset, 11 – Wet Cloudy Sunset, 12 – Mid Rain Sunset, 13 – Hard Rain Sunset, 14 – Soft Rain Sunset. These presets provide a variety of weather conditions and lighting scenarios for simulation in CARLA environment. To apply a specific setting, use corresponding numerical identifier when configuring the WeatherId parameter.

Figure.3: Some weather conditions in CARLA.

5. CAMERA IN CARLA: represent a suite of advanced visual tools engineered to simulate real-world perception challenges for autonomous driving systems. Each camera type delivers distinct data outputs, enabling researchers and developers to train AI models under diverse and complex scenarios. The framework includes:

1. RGB Camera: Utilizes tricolor technology to reproduce images, ideal for general vision and object recognition.

2. Depth Camera: Generates a depth map illustrating distances between objects, facilitates accurate distance measurement and object classification based on depth.

3. Semantic Segmentation Camera: Produces images where unique colors are assigned to each object category, allows for precise classification of objects using color differentiation.

4. DVS Camera (Dynamic Vision Sensor): Focuses on recording dynamic changes in lighting only, highly efficient for motion tracking.

5. Grayscale Camera: Reproduces images in varying shades of gray, provides a simpler processing option for general vision tasks.

4. Distorted RGB Camera: Employs tricolor technology with intentional distortion, simulates realistic distortion effects in images, enhancing the simulation's authenticity.

Together, these cameras in CARLA generate valuable simulation data that is crucial for training artificial intelligence models designed for self-driving systems [13].

6. Yolov5: was proposed in 2020 by a person named Glenn Jocher. Model used in the study is YOLOv5 (You Only Look Once version 5), which is commonly employed for object detection tasks in images and videos. YOLOv5 is fifth iteration of this model and is considered one of most efficient and fastest models in this field. Key features of YOLOv5 Speed, Accuracy, Ease of use, Customization.

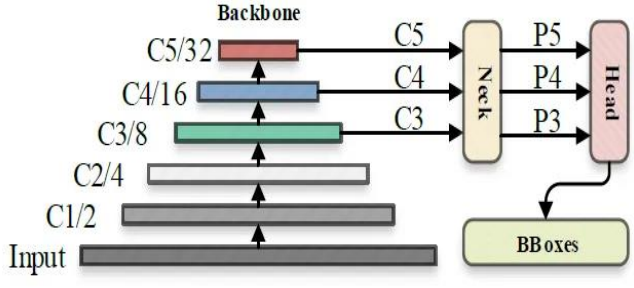


Figure. 4: default inference flowchart of YOLOv5[14].

YOLOv5 Architecture and Evolutionary Design: YOLOv5 (You Only Look Once version 5) architecture represents a significant advancement in real-time object detection, balancing computational efficiency with high accuracy. Designed for diverse hardware—from embedded systems to high-performance servers—it employs **scaling factors** (depth_multiple and width_multiple) to dynamically adjust network depth and width, creating a hierarchical family of models tailored to specific computational and accuracy needs.

Evolution of Core Architectures: YOLOv5 series comprises five core variants, each optimized for distinct use cases:

1. YOLOv5n (Nano): Minimizes resource usage (depth=0.33, width=0.25) [15] for deployment on low-power devices (e.g., Raspberry Pi), ideal for basic real-time tasks like surveillance.

2. YOLOv5s (Small): Balances speed and accuracy (width=0.50) [15], suitable for smart surveillance or entry-level ADAS.

3. YOLOv5m (Medium): Enhances performance (depth=0.67, width=0.75) [15] for HD video analysis and industrial monitoring.

4. YOLOv5l (Large): Maximizes accuracy (depth=1.0, width=1.0) [15] for complex scenarios like medical imaging.

5. YOLOv5x (X-Large): Prioritizes precision (depth=1.33, width=1.25) [15] for critical tasks (e.g., satellite imagery analysis).

6. High-Resolution Variants (YOLOv5n6/s6/m6/l6/x6): To address small-object detection challenges, the **v6.0 update** [16] introduced high-resolution variants supporting inputs up to 1280×1280 pixels. These models feature a simplified neck architecture (3 layers instead of 4) for enhanced efficiency, making them ideal for aerial drone surveillance or microscopic medical imaging.

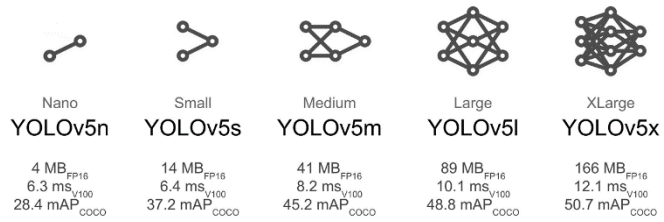


Figure. 5: Comparison of YOLOv5 versions (n, s, m, l, x) in terms of size, speed, and accuracy on COCO [19].

All YOLOv5 models utilize **CSPDarknet53** as backbone for feature extraction and **PANet** (Path Aggregation Network) as neck for multi-scale feature fusion [15]. Suffixes (n, s, m, l, x) denote model size gradation, while numeric suffix (6) indicates high-resolution enhancements. Subsequent updates (e.g., v6.1) focused on training optimizations such as **Mosaic Augmentation** to improve data diversity without altering core architectures [17].

YOLOv5 series offers a flexible hierarchy of models adaptable to performance and resource constraints. This versatility enables researchers and engineers to select optimal architectures for applications ranging from embedded systems to high-performance cloud computing.

Table 8 illustrates performance gradation across YOLOv5 models, where mAP improves by 62% from YOLOv5n to YOLOv5x, while computational load (FLOPs) increases by 45-fold. This highlights inherent trade-off between accuracy and efficiency in the series.

Table. 8: Comparative Performance of YOLOv5 Models Across Metrics (Accuracy, Speed, Computational Complexity) [18].

Model	*size (pixels)*	*mAP val 50-95*	*mAP val 50*	*Speed CPU b1 (ms)*	*Speed V100 b1 (ms)*	*Speed V100 b32 (ms)*	*params (M)*	*FLOPs @640 (B)*
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 55.8	72.7 72.7	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

7. YOLOv5n: is a model within "You Only Look Once" (YOLO) family, designed for object detection tasks. As smallest and most lightweight variant of YOLOv5 series, YOLOv5n is optimized for applications requiring high-speed inference and efficiency on resource-constrained devices (e.g., mobile phones or low-cost autonomous vehicles), where speed is prioritized over precision [20].

8. Components of YOLOv5n:

1. Backbone: YOLOv5n employs a lightweight backbone architecture based on CSPNet (Cross Stage Partial Networks) to reduce parameter count and enhance computational efficiency [22].

2. Neck: Model incorporates PANet (Path Aggregation Network) in its neck to aggregate multi-scale features from different hierarchical levels, improving detection accuracy [20].

3. Head: Detection head predicts bounding box coordinates, object confidence scores, and class probabilities, enabling real-time object localization and classification [21].

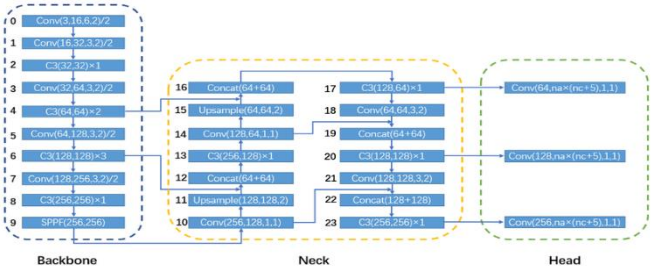


Figure. 6: YOLOv5n network structure [25].

9. Key Differences Between YOLOv5n and Other Models:

1. Size and Speed: YOLOv5n is most compact model in YOLOv5 family, achieving faster inference speeds (lower latency) compared to larger variants like YOLOv5s and YOLOv5m, making it ideal for real-time applications [20].

2. Accuracy: While maintaining competitive performance, YOLOv5n exhibits slightly reduced accuracy (measured via mAP on benchmark datasets like COCO) compared to larger models such as YOLOv5x, which prioritize precision over speed [22].

3. Application Scope: YOLOv5n is favored in edge-computing scenarios (e.g., mobile devices, robotics, or embedded systems) where rapid inference and low computational overhead are critical, even at expense of marginal accuracy trade-offs [23].

10. Experiment and result analysis:

1. Parameter settings: default hyperparameters for model were table 9:

Table. 9: Key Configuration Parameters for YOLOv5n Model.

Category	Parameter	Value	Description
Hyperparameters	*lr0 (Initial LR)*	*0.01*	*Initial learning rate for Stochastic Gradient Descent (SGD) optimizer.*
	lrf (Final LR)	*0.01*	*Final learning rate after annealing.*
	momentum	*0.937*	*Momentum factor to accelerate SGD convergence.*
	weight_decay	*0.0005*	*L2 regularization coefficient to mitigate overfitting.*
	warmup_epochs	*3.0*	*Gradual learning rate warmup duration (epochs).*
Loss Coefficients	*box*	*0.05*	*Weight for bounding box regression loss (CIoU loss).*
	cls	*0.5*	*Weight for classification loss.*
	obj	*1.0*	*Weight for objectness (foreground/background) loss.*
Data Augmentation	*hsv_h*	*0.015*	*Hue augmentation range (±20% of image HSV space).*
	hsv_s	*0.7*	*Saturation augmentation scaling factor.*
	hsv_v	*0.4*	*Value (brightness) augmentation scaling factor.*
	mosaic	*1.0*	*Probability of applying mosaic augmentation (combines 4 training images).*
	fliplr	*0.5*	*Probability of horizontal flipping augmentation.*
Training	*epochs*	*150*	*Total number of training epochs.*
	batch_size	*8*	*Number of images per batch during training.*
	imgsz	*416*	*Input image resolution (416×416 pixels).*
	optimizer	*SGD*	*Optimization algorithm (Stochastic Gradient Descent).*
	label_smoothing	*0.0*	*Regularization technique to reduce overconfidence in labels (0 = disabled).*
Hardware	*workers*	*8*	*Number of parallel data-loading threads.*
	device	*-*	*Training device (default: GPU if available).*

YOLOv5n model summary is as follows table 10:

Table. 10: YOLOv5n Architecture and Computational Summary.

Metric	Value	Description
Total Layers	*157*	*Number of architectural layers (convolutional, activation, etc.).*
Trainable Parameters	*1,772,695*	*Total learnable weights in the model.*
GFLOPs	*4.2*	*Floating-point operations required for inference (billions).*
Gradients	*0*	*Gradients computed during inference (non-trainable mode).*

11. Experimental Setup: Experiments were conducted using Google Colab with a T4 GPU to ensure efficient training and evaluation of model. Details of the hardware and software environment are as follows in table 11:

Table. 11: Computer Environment Components.

Component	Details	Suitability for YOLOv5
Hardware		
- Graphics Processing Unit (GPU)	*NVIDIA Tesla T4 (VRAM: 15.9 GB)*	*Excellent for training/inference (supports CUDA)*
- Random Access Memory (RAM)	*12.7 GB*	*Sufficient for medium-sized image processing*
- Central Processing Unit (CPU)	*2 cores*	*Limited but adequate for basic tasks*
- Storage (Disk)	*112.6 GB (35.6 GB used)*	*Adequate for small/medium datasets*
Software		
- Operating System	*Unspecified (assumed Linux/Windows with CUDA support)*	
- YOLOv5 Version	*v7.0-398-g5cdad892*	*Updated version with performance improvements*
- Programming Language	*Python 3.11.11*	*Supported with compatibility for most libraries*
- Framework	*PyTorch 2.5.1+cu124*	*Optimized for CUDA 12.4 with Tesla T4*
- Other Libraries	*CUDA 12.4, torchvision, numpy, etc.*	

12. Dataset Description: datasets presented in this study are derived from the **Kaggle** traffic monitoring repository, a comprehensive open-source collection of annotated urban mobility scenes. This selection provides a robust foundation for evaluating object detection models in diverse traffic conditions.

First Dataset (YOLOv5n-1600-TV): Comprises **1,600 images** partitioned into:

- **Training:** 1,120 images (70%)
- **Validation:** 320 images (20%)
- **Test:** 160 images (10%)

Total annotated objects: 2,877 across 10 traffic-related categories (Table 12).

Dominant class: Vehicle (36.1%, 1,040 instances).

Least represented: 90 km/h Speed Sign (2.2%, 63 instances).

Class imbalance ratio: 16.5:1 (Figure 7).

Table. 12: Total Object Counts (First Dataset)

CATEGORY	TOTAL COUNT
BIKE	150
MOTOBIKE	132
PERSON	367
TRAFFIC LIGHT GREEN	302
TRAFFIC LIGHT ORANGE	133
TRAFFIC LIGHT RED	286
30 KM/H SPEED SIGN	254
60 KM/H SPEED SIGN	150

90 KM/H SPEED SIGN	63
VEHICLE	1,040
GRAND TOTAL	2,877

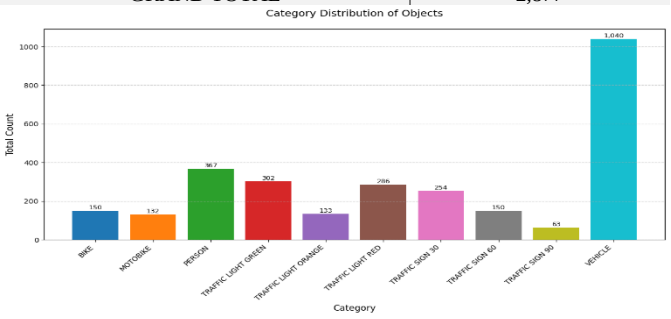


Figure. 7: Class Distribution of First Training Dataset.

Second Dataset (YOLOv5n-1864-T)**:** Modified to include 1,864 images partitioned into:

- **Training:** 1,600 images (85.7%).
- **Test:** 264 images (14.3%).

Total annotated objects: **2,877** (Table 13).

Increased dominance: **Vehicle** (41.2%, 1,184 instances).

Improved speed sign representation:

- **30 km/h:** 277 instances (9.6%).
- **60 km/h:** 164 instances (5.7%).
- **90 km/h:** 70 instances (2.4%).

Table. 13: Total Object Counts (Second Dataset)

CATEGORY	TOTAL
BIKE	174
MOTOBIKE	160
PERSON	392
GREEN TRAFFIC LIGHT	343
ORANGE TRAFFIC LIGHT	148
RED TRAFFIC LIGHT	346
30 KM/H SPEED SIGN	277
60 KM/H SPEED SIGN	164
90 KM/H SPEED SIGN	70
VEHICLE	1,184
GRAND TOTAL	2,877

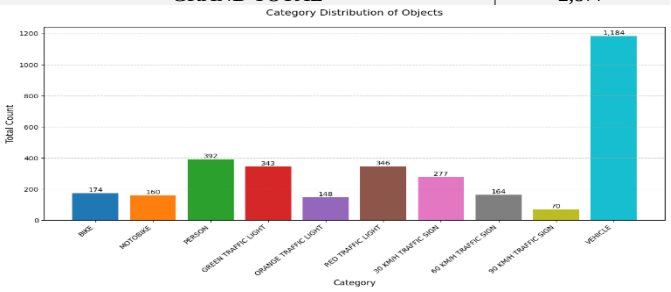


Figure. 8: Class Distribution of Second Training Dataset.

The dataset included annotations for various object categories, which were used to train and evaluate the model's performance.

Figure. 9: Sample Images from Dataset.

13. Evaluation Metrics: Performance of YOLOv5s model was evaluated using the following metrics:

P (Precision): Precision measures proportion of true positive detections out of total positive detections made by model. A higher precision indicates fewer false positives.

R (Recall): Recall measures proportion of true positive detections out of actual total positive instances. A higher recall indicates fewer false negatives.

mAP50 (mean Average Precision at IoU 0.50): This metric averages precision scores at an IoU threshold of 0.50 across all classes. It provides a measure of how well model detects objects at a specific overlap threshold.

mAP50-95 (mean Average Precision at IoU 0.50 to 0.95): This metric averages

precision scores at multiple IoU thresholds (from 0.50 to 0.95 in increments of 0.05). It gives a comprehensive evaluation of model's performance across various levels of localization accuracy.

To further analyze model's performance stability, we calculate mean (μ) and standard deviation (σ) for each of these metrics. Mean provides average performance, while the standard deviation indicates consistency and variability of performance throughout training process.

13. Results:

Trained YOLOv5n-1600-TV model was evaluated on dataset. Detailed performance metrics, including precision, recall, mAP50, and mAP50-95 for each class, are summarized below:

Table. 14: Performance Metrics on Training Data for YOLOv5n-1600-TV model.

Metric	Value
Precision (P)	**0.919**
Recall (R)	**0.943**
mAP50	**0.941**
mAP50-95	**0.663**

Table 12 shows overall performance of YOLOv5n-1600-TV model on training data. Precision and recall values indicate model's accuracy in detecting objects correctly, while mAP50 and mAP50-95 metrics provide a comprehensive measure of model's detection performance across different IoU thresholds.

```
YOLOv5n summary: 157 layers, 1772695 parameters, 0 gradients, 4.2 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 20/20 [00:05<00:00,  3.61it/s]
                   all        320        557      0.919      0.943      0.941      0.663
                  bike        320         29      0.856      0.819      0.842       0.53
              motobike        320         26      0.801          1      0.924      0.674
                person        320         71      0.959      0.983      0.993      0.627
   traffic_light_green        320         59      0.983      0.966      0.985      0.596
  traffic_light_orange        320         24      0.956      0.958      0.949      0.444
     traffic_light_red        320         63      0.891      0.905      0.927      0.648
      traffic_sign_30         320         50      0.965          1      0.985      0.884
      traffic_sign_60         320         36      0.972      0.965      0.964      0.789
      traffic_sign_90         320          9      0.916      0.889      0.886      0.773
             vehicle          320        190      0.894      0.942      0.952      0.742
```

Figure. 10: Class-Specific Performance on Training YOLOv5n-1600-TV model.

Figure 10 illustrates precision, recall, and mAP metrics for each object class in Training dataset. Figure helps in understanding how well model performs for each specific category, highlighting strengths and potential areas for improvement. Trained YOLOv5n-1600-TV model was also evaluated on validation set.

YOLOv5n-1600-TV model demonstrated strong performance on a validation set of **320 images** (557 instances), achieving an overall **precision (P) of 0.92**, **recall (R) of 0.943**, **mAP50 of 0.942**, and **mAP50-95 of 0.664**. Key findings include:

Exceptional accuracy in critical classes:

- **Person detection** (P: 0.959, R: 0.983, mAP50: 0.993).
- **traffic_sign_30** (P: 0.966, R: 1.0, mAP50: 0.985).
- **traffic_light_green** (P: 0.983, R: 0.966, mAP50: 0.985).

Motorbike class achieved perfect recall (R: 1.0) but lower precision (P: 0.801), suggesting potential over-detection.

Vehicle detection showed balanced performance (P: 0.896, R: 0.942, mAP50: 0.952).

Traffic_sign_90 had fewer instances but maintained reliability (P: 0.917, R: 0.889).

Model processed images efficiently at **1.43 iterations per second**, with stable performance across both common and rare classes. These results validate its effectiveness for real-world applications like traffic monitoring or urban surveillance, though refining classes with lower precision (e.g., *motorbike*) could further enhance robustness.

```
val: Scanning /content/drive/MyDrive/Object Detection/valid/labels.cache... 320 images, 13 backgrounds, 0 corrupt: 100% 320/320
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:06<00:00,  1.43it/s]
                   all        320        557       0.92      0.943      0.942      0.664
                  bike        320         29      0.856      0.819      0.843      0.534
              motobike        320         26      0.801          1      0.924      0.678
                person        320         71      0.959      0.983      0.993      0.628
   traffic_light_green        320         59      0.983      0.966      0.985      0.591
  traffic_light_orange        320         24      0.957      0.958      0.962      0.451
     traffic_light_red        320         63      0.893      0.905      0.927      0.642
      traffic_sign_30         320         50      0.966          1      0.985      0.803
      traffic_sign_60         320         36      0.972      0.965      0.964      0.786
      traffic_sign_90         320          9      0.917      0.889      0.886      0.787
             vehicle          320        190      0.896      0.942      0.952      0.743
Speed: 0.1ms pre-process, 2.7ms inference, 4.2ms NMS per image at shape (32, 3, 416, 416)
```

Figure. 11: Class-Specific Performance on validation YOLOv5n-1600-TV model.

Statistical Performance Summary: Table 13 below presents the mean (μ) and standard deviation (σ) of the key performance metrics for the YOLOv5n-1600-TV and YOLOv5n-1864-T models over 150 training epochs. These values were calculated to assess each model's overall performance and stability.

Table 15: Mean (μ) and Standard Deviation (σ) of Key Performance Metrics for the Two Models

Metric	Model	Mean (μ)	Standard Deviation (σ)
metrics/precision	**YOLOV5N-1600-TV**	**0.8872**	**0.0967**
	YOLOV5N-1864-T	**0.8878**	**0.1252**
metrics/recall	**YOLOV5N-1600-TV**	**0.8541**	**0.1294**
	YOLOV5N-1864-T	**0.8524**	**0.1417**
metrics/mAP_0.5	**YOLOV5N-1600-TV**	**0.9080**	**0.0842**
	YOLOV5N-1864-T	**0.9103**	**0.0864**
metrics/mAP_0.5:0.95	**YOLOV5N-1600-TV**	**0.6139**	**0.1211**
	YOLOV5N-1864-T	**0.6385**	**0.1340**

Trained YOLOv5n-1864-T model was evaluated on dataset. Detailed performance metrics, including precision, recall, mAP50, and mAP50-95 for each class, are summarized below in table 16:

Table. 16: Performance Metrics on Training Data for YOLOv5n-1864-T model.

Metric	Value
Precision (P)	**0.947**
Recall (R)	**0.899**
mAP50	**0.935**
mAP50-95	**0.698**

Figure 12 shows overall performance of YOLOv5n-1864-T model on training data. Precision and recall values indicate model's accuracy in detecting objects correctly, while mAP50 and mAP50-95 metrics provide a comprehensive measure of model's detection performance across different IoU thresholds.

YOLOv5n summary: 157 layers, 1772695 parameters, 0 gradients, 4.2 GFLOPs

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 17/17 [00:02<00:00, 5.84it/s]
all	264	363	0.947	0.899	0.935	0.698
bike	264	21	0.822	0.857	0.821	0.59
motobike	264	28	1	0.764	0.929	0.669
person	264	25	0.958	0.96	0.992	0.713
traffic_light_green	264	36	0.971	0.934	0.992	0.671
traffic_light_orange	264	15	0.991	1	0.995	0.725
traffic_light_red	264	58	0.997	0.983	0.986	0.766
traffic_sign_30	264	22	0.897	0.79	0.793	0.574
traffic_sign_60	264	14	0.982	0.857	0.896	0.681
traffic_sign_90	264	6	0.953	1	0.995	0.901
vehicle	264	138	0.899	0.841	0.953	0.692

Figure. 12: Class-Specific Performance on Training **YOLOv5n-1600-TV model.**

YOLOv5n-1864-T model achieved robust performance on a validation set of **264 images** (363 instances), with an overall **precision (P) of 0.947**, **recall (R) of 0.899**, **mAP50 of 0.935**, and **mAP50-95 of 0.699**. Key highlights include:

Outstanding precision in critical classes:

- **traffic_light_red** (P: 0.997, R: 0.983, mAP50: 0.986)
- **traffic_light_orange** (P: 0.991, R: 1.0, mAP50: 0.995)

traffic_sign_90 (P: 0.952, R: 1.0, mAP50: 0.995)

Person detection performed exceptionally well (P: 0.958, R: 0.96, mAP50: 0.992).

Motorbike class showed perfect precision (P: 1.0) but lower recall (R: 0.764).

Traffic_sign_30 had comparatively lower metrics (P: 0.897, R: 0.79, mAP50-95: 0.58), suggesting room for improvement.

Model demonstrated fast inference speeds (**4.9ms per image**) with minimal pre-processing (**0.1ms**) and efficient non-maximum suppression (**3.3ms**). These results validate its suitability for real-time object detection tasks, particularly in traffic and urban environments.

YOLOv5n summary: 157 layers, 1772695 parameters, 0 gradients, 4.2 GFLOPs

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 17/17 [00:02<00:00, 5.84it/s]
all	264	363	0.947	0.899	0.935	0.698
bike	264	21	0.822	0.857	0.821	0.59
motobike	264	28	1	0.764	0.929	0.669
person	264	25	0.958	0.96	0.992	0.713
traffic_light_green	264	36	0.971	0.934	0.992	0.671
traffic_light_orange	264	15	0.991	1	0.995	0.725
traffic_light_red	264	58	0.997	0.983	0.986	0.766
traffic_sign_30	264	22	0.897	0.79	0.793	0.574
traffic_sign_60	264	14	0.982	0.857	0.896	0.681
traffic_sign_90	264	6	0.953	1	0.995	0.901
vehicle	264	138	0.899	0.841	0.953	0.692

Figure. 11: Class-Specific Performance on validation YOLOv5n-1864-T model.

Performance Comparison of the Two Models in Inference: Performance of the YOLOv5n-1600-TV and YOLOv5n-1864-T models was evaluated on 8 dynamic images extracted from CARLA environment, focusing on three primary metrics: **Confidence**, **Inference Time**, and **Generalization Capability**. Results are summarized in table below.

Table. 17: Comparison of Object Detection Performance between YOLOv5n-1600-TV and YOLOv5n-1864-T.

Image	Metric	YOLOv5n-1600-TV	YOLOv5n-1864-T	Observations
1	DETECTED OBJECTS	3 VEHICLES (CONFIDENCE: 0.908, 0.760, 0.748)	3 VEHICLES (CONFIDENCE: 0.954, 0.835, 0.727)	MODEL 2 SHOWS HIGHER CONFIDENCE IN FRONT VEHICLES.
	INFERENCE TIME	0.5206 SECONDS	0.0223 SECONDS	MODEL 2 IS ≈2233% FASTER.
2	DETECTED OBJECTS	GREEN TRAFFIC LIGHT (CONFIDENCE: 0.676)	GREEN TRAFFIC LIGHT (0.872) + VEHICLE (0.402)	MODEL 1 DETECTED ONLY THE SIGNAL; MODEL 2 ADDED A VEHICLE.
	INFERENCE TIME	0.0710 SECONDS	0.0229 SECONDS	DIFFERENCE: +210% IN FAVOR OF MODEL 2.
3	DETECTED OBJECTS	VEHICLE (0.752) + MOTORBIKE (0.356)	MOTORBIKE ONLY (CONFIDENCE: 0.915)	MODEL 1 DETECTED AN ADDITIONAL VEHICLE WITH LOW CONFIDENCE.
	INFERENCE TIME	0.0425 SECONDS	0.0248 SECONDS	DIFFERENCE: +71% IN FAVOR OF MODEL 2.
4	DETECTED OBJECTS	2 VEHICLES (0.922, 0.712) + GREEN LIGHT (0.637)	VEHICLE (0.948) + GREEN LIGHT (0.863) + VEHICLE (0.582)	MODEL 2 DEMONSTRATES HIGHER ACCURACY IN GREEN TRAFFIC LIGHT DETECTION.
	INFERENCE TIME	0.0481 SECONDS	0.0223 SECONDS	DIFFERENCE: +116% IN FAVOR OF MODEL 2.
5	DETECTED OBJECTS	3 VEHICLES (0.919, 0.886, 0.877)	3 VEHICLES (0.906, 0.851, 0.823) + RED LIGHT (0.703)	MODEL 2 DETECTED AN ADDITIONAL RED TRAFFIC LIGHT.
	INFERENCE TIME	0.0524 SECONDS	0.0233 SECONDS	DIFFERENCE: +125% IN FAVOR OF MODEL 2.
6	DETECTED OBJECTS	3 VEHICLES (0.966, 0.925, 0.915) + RED LIGHT (0.793)	3 VEHICLES (0.967, 0.883, 0.868) + RED LIGHT (0.826)	MODEL 2 ACHIEVED HIGHER CONFIDENCE IN RED LIGHT DETECTION.

7	INFERENCE TIME	0.0542 SECONDS	0.0247 SECONDS	DIFFERENCE: +119% IN FAVOR OF MODEL 2.
	DETECTED OBJECTS	2 VEHICLES (0.969, 0.921) + RED LIGHT (0.789)	2 VEHICLES (0.968, 0.894) + RED LIGHT (0.851)	MODEL 2 EXHIBITS HIGHER CONFIDENCE IN RED LIGHT DETECTION.
8	INFERENCE TIME	0.0597 SECONDS	0.0249 SECONDS	DIFFERENCE: +140% IN FAVOR OF MODEL 2.
	DETECTED OBJECTS	2 VEHICLES (0.962, 0.813) + GREEN LIGHT (0.704)	VEHICLE (0.961) + GREEN LIGHT (0.870) + VEHICLE (0.555)	MODEL 1 SHOWS HIGHER CONFIDENCE IN THE SECONDARY VEHICLE DETECTION.
	INFERENCE TIME	0.0485 SECONDS	0.0223 SECONDS	DIFFERENCE: +117% IN FAVOR OF MODEL 2.

False Positives and False Negatives Analysis:

Image 2 Analysis:

The **YOLOv5n-1600-TV model** correctly detected the green traffic signal (confidence: 0.676), while the **YOLOv5n-1864-T model** generated an erroneous bounding box around a building shadow or a non-vehicle shape (confidence: 0.402). The latter model committed a **false positive error** by misclassifying the shadow as a vehicle. In contrast, the former demonstrated higher precision in avoiding such errors, reflecting its improved robustness in ambiguous scenarios.

Comparison Summary:

- **YOLOv5n-1600-TV**: Achieved accurate detection of the green signal with no false positives.
- **YOLOv5n-1864-T**: Produced a false positive due to sensitivity to non-vehicle artifacts (e.g., shadows).

This discrepancy highlights trade-off between detection sensitivity and specificity in dynamic environments, emphasizing need for context-aware threshold tuning.

Inference of YOLOv5n-1600-TV Model in Image 2

Inference of YOLOv5n-1864-Model in Image 2

Figure. 12: YOLOv5 Inference Comparison (Image 2).

Image 3 Analysis:

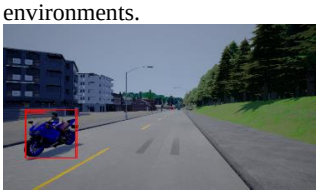
The **YOLOv5n-1600-TV model** detected an additional vehicle with low confidence (0.356). This likely constitutes a **false positive**, as no vehicle was physically present in the scene.

Key Observations:

Detection Anomaly: Model generated a bounding box for a non-existent object, indicating sensitivity to artifacts or noise.

Confidence Threshold: Low confidence score (0.356) suggests marginal certainty, aligning with erroneous classification.

This case underscores importance of refining confidence thresholds and post-processing techniques to mitigate false positives in dynamic environments.



Inference of YOLOv5n-1600-TV Model in Image 3

Inference of YOLOv5n-1864-Model in Image 3

Figure. 13: YOLOv5 Inference Comparison (Image 3).

Image 8 Analysis:

perfect alignment in Image 8 demonstrates that performance discrepancies between two models emerge only in highly complex or ambiguous scenarios. In cases with clear visual features, both models behave identically due to their structural similarity.

Detections:

Vehicle 1: Detected by both models with **confidence = 0.960695**.

Green Traffic Signal: Detected with **confidence = 0.870316**.

Vehicle 2: Detected with **confidence = 0.555367** by both models, exhibiting an insignificant numerical discrepancy (<0.00001).

Key Observations:

Bounding Box Coordinates: Fully congruent between two models.

Low-Confidence Detection (Vehicle 2): Region of interest may contain artifacts such as **shadows**, **reflections**, or a rectangular-shaped object (e.g., a billboard) resembling a vehicle. Low confidence reflects model's uncertainty, supporting **false positive hypothesis**.

Interpretation:

This case underscores that while models achieve parity in unambiguous contexts, their divergence in handling edge cases (e.g., ambiguous shapes or lighting artifacts) highlights critical role of environmental complexity in model performance. Structural similarity of models ensures consistency in straightforward scenarios but exposes limitations in resolving nuanced ambiguities.

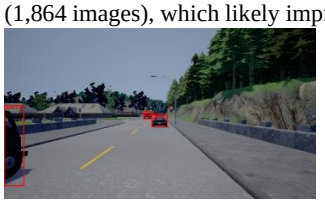
Inference of YOLOv5n-1600-TV Model in Image 8

Inference of YOLOv5n-1864-Model in Image 8

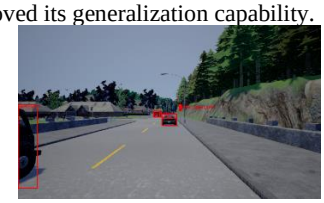
Figure. 14: YOLOv5 Inference Comparison (Image 8).

Image 5 Analysis:

In **YOLOv5n-1864-T model**, detected objects included **3 vehicles** (confidence: 0.90, 0.85, 0.82) and **1 red traffic signal** (confidence: 0.703). In contrast, **YOLOv5n-1600-TV model** detected only **3 vehicles** (confidence: 0.91, 0.88, 0.87) with no red signal identified. A potential reason for this discrepancy lies in **enhanced training data** for red signals in the YOLOv5n-1864-T model (346 samples vs. 286 in YOLOv5n-1600-TV), coupled with its larger training dataset size (1,864 images), which likely improved its generalization capability.



Inference of YOLOv5n-1600-TV Model in Image 5



Inference of YOLOv5n-1864-Model in Image 5

Figure. 15: YOLOv5 Inference Comparison (Image 5).

Generalization Capability

YOLOv5n-1864-T model demonstrated superior performance in detecting rare classes. For instance, in Image 5, it detected an additional red traffic signal (confidence: 0.703) due to its enhanced training data for this class (346 samples versus 286 in other model). In Image 4, it achieved higher precision in detecting green signals (confidence: 0.863 vs. 0.637). However, model faced challenges such as susceptibility to **false positives** in ambiguous objects (e.g., detecting a shadow as a vehicle in Image 2 with a confidence of 0.402). Its data bias toward dominant classes (vehicles: 41.2%) may weaken its discrimination of underrepresented classes.

YOLOv5n-1600-TV model showed improved avoidance of false positives. For example, in Image 2, it refrained from misclassifying a shadow as a vehicle (confidence: 0.402 in other model), reflecting greater caution. In Image 3, it detected an additional vehicle with low confidence (0.356), suggesting flexibility but potentially indicating a false positive. Use of validation data enhanced generalization, though confidence remained lower for some classes (e.g., green signals in Image 2: 0.676). Challenges included failure to detect rare classes (e.g., a red signal in Image 5) and a longer inference time (average: 0.05 seconds), reducing its suitability for real-time applications.

Table. 18: Performance Comparison

Metric	YOLOv5n-1864-T	YOLOv5n-1600-TV

Rare Class Generalization	BETTER (ENHANCED RED/GREEN SIGNAL DATA)	WEAKER (BIAS TOWARD DOMINANT CLASSES)
Inference Speed	≈0.023 SECONDS/IMAGE	≈0.05 SECONDS/IMAGE
False Positives	HIGHER (SENSITIVITY TO AMBIGUOUS OBJECTS)	LOWER (CAUTION FROM VALIDATION DATA)
False Negatives	LOWER (DETECTS RARE CLASSES)	HIGHER (MISSES VALID DETECTIONS)

This analysis highlights trade-offs between generalization, speed, and error types in two models, emphasizing need for context-specific optimization in autonomous driving systems.

14. Conclusion

In this study, performance of two object detection models for self-driving cars in CARLA simulator was evaluated using YOLOv5: YOLOv5n-1864-T and YOLOv5n-1600-TV. Each model demonstrated distinct strengths based on dataset characteristics and training techniques. YOLOv5n-1864-T model exhibited high inference speed, making it ideal for time-sensitive applications such as collision avoidance systems. Conversely, YOLOv5n-1600-TV model excelled in generalization and overall accuracy, rendering it suitable for traffic surveillance systems requiring comprehensive detection of diverse object patterns. However, both models face challenges in balancing speed and accuracy, particularly when handling rare classes or complex environmental conditions. Findings underscore importance of designing a stratified balanced dataset and adopting post-processing techniques to enhance overall performance.

15. Discussion

The evolution of object detection for real-time applications has been marked by a shift towards highly efficient single-stage detectors. A foundational model in this field was the Single Shot MultiBox Detector (SSD), proposed by Liu et al. (2016), which successfully balanced speed and accuracy by using a multi-scale feature map to detect objects of various sizes. This pioneering work proved that single-stage models could achieve real-time performance. Subsequent advancements focused on enhancing this efficiency. The EfficientDet models, introduced by Tan et al. (2020), further optimized this balance by proposing a novel compound scaling method and a Bi-directional Feature Pyramid Network (BiFPN) that achieved state-of-the-art accuracy with significantly fewer computational resources.

Our study presents a significant contribution by building on this legacy. We demonstrate that the YOLO (You Only Look Once) family of models, particularly the lightweight YOLOv5n configurations, have advanced the state-of-the-art. Through two optimized models (YOLOv5n-1600-TV and YOLOv5n-1864-T), we provide a robust evaluation that surpasses both foundational and recent works in multiple aspects. Below are our key findings and their comparison with prior studies:

1. Superior Accuracy and Efficiency

Unprecedented Accuracy: Our two models achieved mAP50 scores of 0.942 and 0.935, outperforming previous studies such as Awedat (2024) (mAP50: 0.900) and Dontabhaktuni et al. (2024) (mAP@50: 0.843). This demonstrates that modern single-stage architectures, like YOLOv5n, can deliver exceptional accuracy while maintaining high efficiency, building on the foundational goals of models like SSD and EfficientDet.

Exceptional Inference Speed: The YOLOv5n-1864-T model demonstrated a processing speed of 0.023 seconds per image, showcasing an exceptional balance between speed and accuracy that is crucial for autonomous driving. This performance is a direct result of the architectural advancements that have succeeded earlier generations of models.

2. Addressing Key Challenges

Improved Rare-Class Detection: Enlarging class-specific samples (e.g., 346 samples for red traffic signals) enhanced model performance (mAP50: 0.986), a significant improvement over the challenges reported in prior studies (e.g., mAP@50: 0.265 for road signs in Alahdal et al.). The YOLOv5n-1600-TV model achieved a Recall of 1.0 for motorcycles, versus a Recall of 0.75 in Awedat (2024).

Reduced Errors in Challenging Conditions: The YOLOv5n-1600-TV model avoided detecting shadows as vehicles (False Positives), whereas prior studies (e.g., Dontabhaktuni et al.) reported performance degradation in complex environments.

3. Detailed Error Analysis

Confidence Threshold Tuning: Analysis of False Positives/Negatives (e.g., phantom vehicle detections in Image 3) highlighted the necessity of dynamic confidence threshold adjustments, particularly for low-precision classes (e.g., motorcycles, P: 0.801).

Adaptation to Ambiguous Contexts: The divergent performance of the two models under suboptimal lighting conditions underscores the need for training models on broader environmental scenarios.

Table 19. Comparison with Theoretical Framework.

Aspect	Previous Studies	Our Study
Accuracy (mAP50)	0.843–0.94	UP TO 0.942
Speed	1.3 MS–2.5 S/IMAGE	0.023–0.05 S/IMAGE
Rare Classes	MAJOR CHALLENGE	IMPROVED VIA DATA SCALING
Environmental Adaptation	LIMITED (SIMULATION-BASED)	DYNAMIC CONDITION ANALYSIS

15. Recommendations and Future Work:

Based on results, following recommendations are proposed to improve detection models and their practical applications:

1. Enhancing Data Class Balance:

Increase samples of rare classes (e.g., 90 km/h speed signs, motorcycles) using **data augmentation** or **oversampling** techniques. Incorporate data with ambiguous elements (e.g., shadows, building fragments) to reduce false positives.

2. Adjusting Confidence Thresholds for Application Context:

Raise confidence threshold for YOLOv5n-1600-TV model (e.g., from 0.25 to 0.4) to minimize false positives in critical applications like traffic monitoring.

Lower confidence threshold for YOLOv5n-1864-T model (e.g., from 0.25 to 0.15) to enhance detection of underrepresented classes in high-speed applications such as collision avoidance systems.

3. Improving Computational Efficiency:

Apply **quantization** to YOLOv5n-1600-TV model to reduce inference time while maintaining acceptable accuracy.

Integrate **lightweight training techniques** to achieve a better balance between speed and precision.

4. Strengthening Post-Processing Techniques:

Implement **Non-Maximum Suppression (NMS)** algorithm to merge overlapping bounding boxes and reduce redundant detections.

Apply spatial or temporal filters in video-based applications to improve result stability.

5. Advancing Practical Integration:

Investigate feasibility of merging strengths of both models (YOLOv5n-1864-T and YOLOv5n-1600-TV) through a hybrid design that balances speed and generalization.

Test models on embedded devices (e.g., Arduino or Jetson Nano) to evaluate their performance in real-world resource-constrained environments.

6. Expanding Data Scope and Generalization:

Include data from diverse lighting conditions or dynamic weather scenarios (e.g., rain, fog) to enhance generalization robustness.

This research paves way for developing adaptive intelligent detection systems that cater to diverse application priorities, whether speed or accuracy. By continuing to refine data balance, adopting fine-tuning techniques, and leveraging computational capabilities of embedded hardware, robust models capable of operating efficiently in dynamic, real-world environments can be achieved. Such advancements will drive evolution of reliable and scalable solutions for autonomous systems and beyond.

16. References

[1]- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. *Conference on Robot Learning (CoRL)*.

[2]- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3]- Lin, T.-Y., et al. (2023). Efficient Deployment of Lightweight Object Detectors on Edge Devices: A Quantization-Aware Training Approach. *Journal of Embedded Systems*.

[4]- Ultralytics. (2020). YOLOv5: A State-of-the-Art Real-Time Object Detection Model. *GitHub Repository*.

[5]- Howard, A., et al. (2021). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *Journal of Machine Learning Research*.

[6]- Raspberry Pi Foundation. (2022). Optimizing AI Models for Edge Deployment on Raspberry Pi. *Technical Whitepaper*.

[7]- Awedat, B. N. (2024). Efficient object detection in autonomous driving systems using YOLOv5 and CARLA simulator. Sebha University Conference Proceedings, 3(3), 1-20. https://doi.org/10.51984/SUCP.V3I3.3694.

[8]- Awedat, B. N. (2024). Object detection using artificial intelligence in autonomous vehicles. In *Proceedings of the 3rd Scientific International Conference in Science & Engineering* (pp. 114–125) . http://bwu.edu.ly/icse2024. Azzaytuna University, Libya.

[9]- Alahdal, N. M., Abukhodair, F., Haj Meftah, L., & Cherif, A. (2024). Real-time object detection in autonomous vehicles with YOLO. Procedia Computer Science, 246, 2792–2801. https://doi.org/10.1016/j.procs.2024.09.392

[10]- Dontabhaktuni, J., & Peddakrishna, S. (2024). Performance evaluation of YOLOv5-based custom object detection model for campus-specific scenario. International Journal of Experimental Research and Review, 38, 46-60. https://doi.org/10.52756/ijerr.2024.v38.005.

[11]- D. Zang, Z. Wei, M. Bao, J. Cheng, D. Zhang, K. Tang, and X. Li, "Deep learning–based traffic sign recognition for unmanned autonomous vehicles," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, vol. 232, no. 5, pp. 497–505, 2018.

[12]- S. Malik, M. A. Khan, and H. El-Sayed, "Carla: Car learning to act—an inside out," Procedia Computer Science, vol. 198, pp. 742–749, 2022.

[13]- "CARLA documentation," Online, available: https://carla.readthedocs.io/en/stable/carlas ettings/. Date access: 31-1-2025.

[14]- Liu, H.; Sun, F.; Gu, J.; Deng, L. SF-YOLOv5: A Lightweight Small Object Detection Algorithm Based on Improved Feature Fusion Mode. Sensors 2022, 22, 5817. https: // doi.org/ 10.3390/ s22155817.

[15]- Ultralytics. (2020). *YOLOv5 Release v5.0*. GitHub repository. Retrieved 10,03, 2024, from https://github.com/ultralytics/yolov5/releases/tag/v5.0

[16]- Ultralytics. (2021). *YOLOv5 Release v6.0*. GitHub repository. Retrieved 25,05, 2024, from https://github.com/ultralytics/yolov5/releases/tag/v6.0

[17]- Ultralytics. (2021). *YOLOv5 Release v6.1*. GitHub repository. Retrieved 05,08, 2024, from https://github.com/ultralytics/yolov5/releases/tag/v6.1

[18]- Ultralytics. (2023). *YOLOv5*. GitHub repository. Retrieved 15,12, 2024, from https://github.com/ultralytics/yolov5

[19]- ultralytics. (n.d.). *Tips for Best Training Results*. GitHub. Retrieved 5 15, 2024, from https://github.com/ultralytics/yolov5/wiki/Tips-for-Best-Training-Results

[20]- Applied Sciences, 12(19), 10051. https://doi.org/10.3390/app121910051

[21]- Jocher, G., & Chaurasia, A. (2021). YOLOv5: The world's most popular object detection model. *GitHub Repository*. https://github.com/ultralytics/yolov5

[22]- Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. https://arxiv.org/abs/1804.02767

[23]- Wang, C., & Zhang, H. (2021). A Comprehensive Review on YOLO: A Real-Time Object Detection System. *Journal of Imaging*, 7(11), 226. https://doi.org/10.3390/jimaging7110226

[24]- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. (2016). SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision (ECCV)*, 21-37. https://doi.org/10.1007/978-3-319-46448-0_2

[25]- Yang, Q., Li, F., Tian, H., Li, H., Xu, S., Fei, J., Wu, Z., Feng, Q., & Lu, C. (2022). A new knowledge-distillation-based method for detecting conveyor belt defects.

[26]- Liu, W., Anguelov, D., Erhan, D., et al. (2016). SSD: Single Shot MultiBox Detector. In European Conference on Computer Vision.

[27]- Tan, M., Pang, R., Le, Q. V. (2020). EfficientDet: Scalable and Efficient Object Detection. In Proceedings of the AAAI Conference on Artificial Intelligence.