المؤتمر العلمي الدولي الثالث للعلوم و الهندسة
The 3RD Scientific International Conference in Science & Engineering

http://bwu.edu.ly/icse2024
Received 25/07/2024 Revised 17/08/2024 Published 10/09/2024
icse@bwu.edu.ly *ترسل الورقات كاملة علي

Object Detection Using Artificial Intelligence in Autonomous Vehicles

Bader N. Awedat[a]
[a]Computer science, faculty of Information Technology / Azzaytuna University, Libya
*Crosspnding author: bader_najep@yahoo.com

Abstract: This research paper focuses on object detection, such as bicycles, motorcycles, persons, traffic lights, traffics signs, and vehicles within the framework of autonomous driving systems in the CARLA environment. Currently, object detection in autonomous driving primarily relies on actual autonomous vehicles, which face challenges such as high costs and real-time implementation difficulties. The open-source CARLA system enables precise and cost-effective experimentation. In this paper, the deep learning model YOLOv5 was used, yielding good results in both training and validation datasets. A total of 1560 different images were used in the training process, divided into 1120 images for training, 160 images for testing, and 320 images for validation. The training results showed a Precision (P) of 0.898, Recall (R) of 0.827, mAP@50 of 0.900, and mAP@50-95 of 0.583. In the validation results, the Precision (P) was 0.891, Recall (R) was 0.801, mAP@50 was 0.880, and mAP@50-95 was 0.542. These results indicate that the model is capable of accurately detecting and retrieving objects effectively.

Keywords: (Object Detection, Artificial Intelligence, Autonomous Vehicles, CARLA, mean Average Precision, traffic lights, traffics signs.

Introduction

Self-driving cars are living in a technologically transformative era, as scientists and engineers strive to enhance the performance of these systems to ensure the safety and efficiency of autonomous navigation. Object detection emerges as a crucial element in this context, requiring leading vehicles to interact efficiently with these objects to ensure safe and effective traffic control. Amid the rapid advancements in artificial intelligence-based autonomous vehicle technologies, understanding the behavior of objects in diverse weather scenarios has become imperative to ensure the safety of vehicles and pedestrians. Simulation systems such as CARLA provide valuable means to realistically and effectively address these challenges.

RELATED WORK

1. In this research paper by S. Malik, M.A. Khan, and H. El-Sayed (2022), an in-depth exploration of open-source autonomous driving simulator CARLA was conducted. CARLA stands out as a robust tool for development and testing of autonomous driving systems within a virtual urban environment. Simulator is distinguished by its simulation engine, which is built on Unreal Engine 4, and it supports crucial features such as sensor configuration and definition of environmental conditions. Paper provided a comprehensive overview of simulation's structure, shedding light on intricacies of virtual environment, encompassing elements like roads, buildings, and various other components. It also detailed configurability of weather conditions and terrains. Research extensively covered array of available sensors in CARLA, including cameras, LiDAR, and radar, elucidating their roles in training autonomous driving systems. Special emphasis was placed on distinctive features within CARLA, such as Traffic Manager, Scenario Runner, and Responsibility Sensitive Safety. Paper delved into different releases of CARLA, showcasing their updates and

enhanced features. In conclusion carried out a comparative analysis between CARLA and other simulators. It effectively demonstrated how CARLA serves as a valuable tool for training intelligent models for autonomous driving systems. Paper also highlighted diverse applications of CARLA in areas like imitation learning, cooperative driving, and interactions within driving environments. Throughout discussion, paper underscored significance of recognizing that while simulation cannot replace real-road experiences, it indeed provides a safe and effective environment for testing and advancing autonomous driving technologies [1].

2. In a study conducted by Raymond Muller and others (2022), research was carried out using DRIVETRUTH system. DRIVETRUTH is a data collection framework that relies on CARLA simulator to gather and automatically label autonomous driving data. It extends capabilities of Semantic LiDAR to compute three-dimensional bounding boxes for static objects, including traffic signals, and also retrieves customized bounding boxes for dynamic objects. DRIVETRUTH enables tracking of context for each object, providing additional support for autonomous driving systems, particularly in safety and security applications. Research demonstrates that DRIVETRUTH can be used to automatically collect simulation data that simulates real-world driving conditions, with a focus on classifying and tracking traffic signals [2]

3. In paper by Dosovitskiy, A. et al. (2017), an open autonomous driving simulator called CARLA was introduced. This simulator is built on Unreal Engine 4 and provides an interactive environment for development and testing of autonomous driving systems. CARLA enables researchers to assess performance and train autonomous driving systems using three main approaches: a classical modular pipeline, neural network training through imitation, and neural network training through reinforcement learning. Simulator's architecture strikes a balance between configuration flexibility and simulation realism, leveraging high-quality graphics and realistic physics provided by Unreal Engine 4. System includes an environmental design with three-dimensional models of both static and dynamic objects, along with support for diverse weather conditions and lighting. Client sensors, including cameras and depth sensors, can be configured to obtain precise readings. Performance of three different autonomous driving methods was evaluated using CARLA, including a classical modular pipeline, imitation learning, and reinforcement learning. Results revealed that performance is not perfect even in simple tasks, and challenges escalate in new environments and varied weather conditions. Findings shed light on difficulties of generalizing to new environments and different weather conditions, providing a comparison between three methods used in paper. In conclusion, CARLA simulator offers research community an opportunity for active engagement in exploring and developing autonomous driving technologies [3]

4. Zang et al. (2018) proposed an approach for traffic sign recognition using deep learning techniques for unmanned autonomous vehicles. Study utilized MASTIF dataset, which contains video clips captured by a camera mounted on vehicle. Traffic sign detection tool employed in this study is Faster R-CNN approach, which is used to sense traffic signs in each frame of video clips. Results were aggregated using Mean Shift method for temporal matching, representing each traffic sign with quaternion numbers. Two quaternion neural networks (QCNNs) were used to extract spatial and temporal features, and these features were fused to achieve a high-level description of traffic signs in both domains.

Experimental results demonstrated effectiveness of proposed approach, comparing favorably with other methods such as color model, SVM-based decision support method, and another deep learning approach. Proposed approach showed superior performance in traffic sign detection and recognition, improving accuracy and efficiency compared to other methods [4].

W. Gao, J. Tang, and T. Wang. (2021) study divided 1512 images into training, validation, and test sets in an 8:1:1 ratio. Evaluation metrics used were mAP (mean Average Precision) and FPS (Frames Per Second). experiment was conducted on a system with a single GeForce GTX 1080Ti graphics card with 11GB video memory, equipped with CUDA 10.0 and an Intel i5-9400F CPU with 16GB of RAM. Python 3.6 was used as the coding environment. YOLOv4 was implemented using the Keras framework with TensorFlow backend, while CenterNet and Faster-RCNN were implemented using PyTorch. For YOLOv4, parameters were tuned using transfer learning, with specific settings for frozen and global training phases. CenterNet utilized a two-step training method with pre-trained Hourglass backbone network weights fine-tuned on the dataset. Faster-RCNN utilized a VGG16 backbone with specific anchor settings and training parameters. YOLOv4 achieved the highest mAP across all object classes but had the lowest FPS. CenterNet and Faster-RCNN had lower mAP but higher FPS compared to YOLOv4.

Overall, YOLOv4 demonstrated higher detection accuracy but lower processing speed compared to CenterNet and Faster-RCNN [8].

Table 1: Previous Studies Summary

Raymond Muller and others (2022)

Model	Dataset	Results
YOLOv3	- DRIVETRUTH: Sample dataset with 500 objects	- Achieved an average precision of 47.2% at 50% Intersection over Union (IoU) with ground truth.
DaSiamRPN	- DRIVETRUTH: Randomly sampled dataset with 500 objects	- Achieved an average precision of 77% at 50% IoU with ground truth.
Contextual Data Usage	- DRIVETRUTH: Dataset with contextual information	- Enhances accuracy and robustness in object tracking. - Improves Kalman filter predictions for bounding boxes.

Dosovitskiy, A. et al. (2017)

Model	Best Dataset Condition	Overall Success Rate
Modular Pipeline (MP)	New Weather	67.25%
Imitation Learning (IL)	Training	77.75%
Reinforcement Learning (RL)	Training	36.25%

W. Gao, J. Tang, and T. Wang. (2021)

Model	AP-person	AP-car	AP-trafficLight	AP-motorcycle	mAP	FPS
YOLOv4	75.37%	88.41%	66.56%	33.48%	65.96%	16.31
CenterNet	49.56%	83.14%	50.96%	35.68%	54.83%	12.25
Faster-RCNN	53.12%	69.90%	58.28%	29.37%	52.67%	7.43

Zang et al. (2018)

Model	Dataset	No. of Training Samples	No. of Test Samples	No. of Classes	No. of Correct Detections	No. of False Detections	Detection Rate (%)	No. of Correctly Classified Signs	Classification Rate (%)
QCNN	TS2011 (Detection)	1015	-	-	-	-	-	-	-
QCNN	TS2010 (Detection)	-	587	-	583	7	99.31	-	-
QCNN	TS2009 (Classification)	6430	-	277	-	-	-	582	99.15

Raymond Muller and others (2022)

Model		Dataset			Results			
QCNN	TS2010 (Classification)	-	590	-	-	-	-	-	-

CARLA (Car Learning to Act)

CARLA, an open-source simulator for autonomous driving research. CARLA has been developed from the ground up to support development, training, and validation of autonomous urban driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely [3].

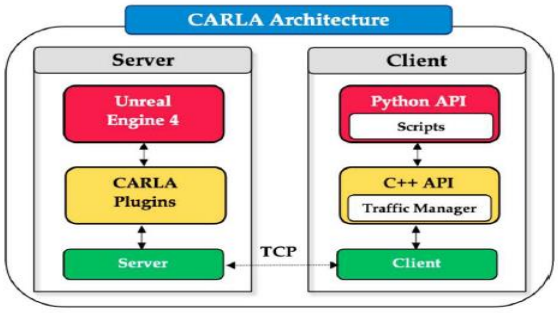

Figure. 1: CARLA client-server architecture [4].

Autonomous Vehicle System Architecture in CARLA Environment:

1. Global Planner: The Global Planner is responsible for determining the overall path of the journey based on the set destination and road conditions.

2. Waypoint Position Decision: Controls the direction of the vehicle and specifies specific locations along the path.

3. Obstacle Detection and Perception: Recognizes obstacles in the environment and enhances the vehicle's understanding of its surroundings.

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

5. Self-Localization: Allows the vehicle to accurately determine its position within the environment.

6. Motion Control: Manages the vehicle's control and regulates its movement based on the decisions made.

7. Local Planner: Determines a short and local path to assist in overcoming immediate obstacles.

8. Collision Avoidance: Controls the vehicle's movement to avoid collisions with obstacles.

These components collaborate to achieve a self-driving system in the CARLA environment, enabling the vehicle to control itself and adapt its movement based on the surrounding conditions and specified objectives.

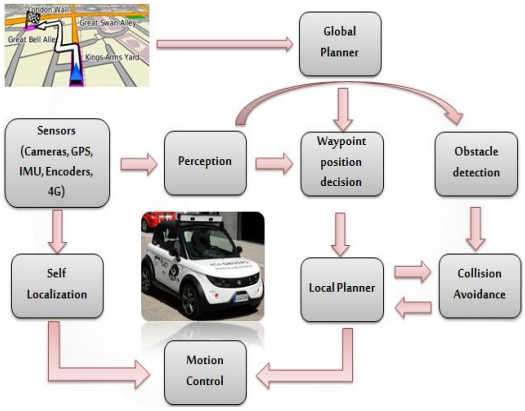

Figure. 2. System architecture of autonomous vehicle [1].

Weather Presets:

In CARLA, weather and lighting conditions can be customized through a selection of predefined settings. To choose a specific preset, adjust the "WeatherId" key in the configuration file "CarlaSettings.ini." The available presets are as follows:

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 offer a range of weather conditions and lighting scenarios for simulation purposes in the CARLA environment. To apply a specific setting, refer to the corresponding numerical identifier when configuring the WeatherId parameter.

Figure.3. Some weather conditions in CARLA.

CAMERA IN CARLA:

CARLA uses a variety of camera types to simulate driving scenes and artificial intelligence applications, including:

1. RGB Camera: Type: RGB. Usage: Reproduces images using tricolor technology. Applications: Used for general vision and object recognition.

2. Depth Camera: Type: Depth. Usage: Generates a depth map showing distance between objects. Applications: Distance measurement and object classification based on depth.

3. Semantic Segmentation Camera: Type: Semantic Segmentation. Usage: Reproduces an image where a unique color is assigned to each object category.

Applications: Accurate classification of objects using colors.

4. DVS Camera: Type: Dynamic Vision Sensor (DVS). Usage: Records dynamic changes in lighting only. Applications: Efficient motion tracking.

5. Grayscale Camera: Type: Grayscale. Usage: Reproduces images in shades of gray. Applications: Used for general vision with less processing complexity.

6. Distorted RGB Camera: Type: Distorted RGB. Usage: Reproduces images using tricolor technology with intentional distortion to simulate potential distortion effects. Applications: Used to simulate image distortion effects in realistic environments.

Cameras in CARLA are used to generate simulation data for training artificial intelligence models for self-driving systems [6].

Yolov5:

YOLOv5 was proposed in 2020 by a person named Glenn Jocher. The 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 the fifth iteration of this model and is considered one of the most efficient and fastest models in this field.

Key features of YOLOv5:

1. Speed: YOLOv5 is characterized by its high speed in image processing and object detection compared to many other models.
2. Accuracy: It offers a good balance between speed and accuracy, making it suitable for real-time object detection applications.
3. Ease of use: It provides an easy-to-use interface and comes with numerous ready-to-use examples, facilitating the training and application process.
4. Customization: The model can be easily modified to suit a wide range of applications by changing parameters and retraining the model on customized datasets.

In summary, YOLOv5 is a robust and efficient model for object detection, widely used in various practical applications.

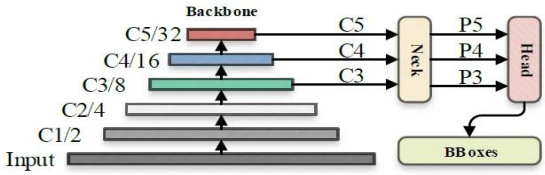

Fig 4. default inference flowchart of YOLOv5[9].

Network Structure of YOLOv5

Generally speaking, the network structure of YOLOv5 refers to the backbone and neck.

1. Backbone: The backbone of YOLOv5 is shown in Figure 5. The main structure is the stacking of multiple CBS (Conv + BatchNorm + SiLU) modules and C3 modules, and finally one SPPF module is connected. CBS module is used to assist C3 module in feature extraction, while SPPF module enhances the feature expression ability of the backbone[9].

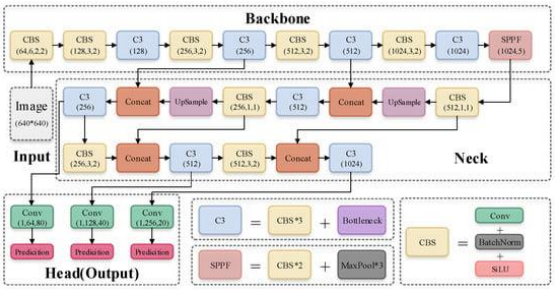

Fig 5. Default network structure of YOLOv5[9].

Therefore, in the backbone of YOLOv5, the most important layer is the C3 module. The basic idea of C3 comes from CSPNet (cross stage partial networks. C3 can actually be regarded as the specific implementation of CSPNet. YOLOv5 uses the idea of CSPNet to build the C3 module, which not only ensures that the backbone has excellent feature extraction ability, but also curbs the problem of gradient information duplication in the backbone [9].

2. Neck: In neck, YOLOv5 uses the methods of FPN and PAN, as shown in Figure 6. The basic idea of FPN is to up-sampling the output feature map (C3, C4, and C5) generated by multiple convolutions down sampling operations from the feature extraction network to generate multiple new feature maps (P3, P4, and P5) for detecting different scales targets [9].

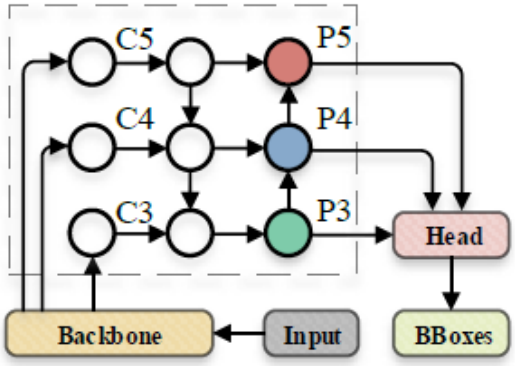

Fig 6. Neck [9].

Feature fusion path of FPN is top-down. On this basis, PAN reintroduces a new bottom-up feature fusion path, which further enhance detection accuracy for different scales objects [9].

Experiment and result analysis

1. Parameter settings: The default hyperparameters for model were:

Table 2: hyperparameters for model.

Parameters	Value
lr0	0.01
lrf	0.01
momentum	0.937
weight_decay	0.0005
warmup_epochs	3.0
warmup_momentum	0.8
warmup_bias_lr	0.1
box	0.05
cls	0.5
cls_pw	1.0
obj	1.0
obj_pw	1.0
iou_t	0.2
anchor_t	4.0
hsv_h	0.015
hsv_s	0.7
hsv_v	0.4
translate	0.1
scale	0.5
fliplr	0.5
mosaic	1.0
batch	32
epochs	100

The YOLOv5s model summary is as follows:

Table 3: model summary

Parameters	weight loss
Layers	157
Parameters	7,037,095
Gradients	0
GFLOPs	15.8

2. Experimental Setup:

The experiments were conducted using Google Colab with a T4 GPU to ensure efficient training and evaluation of the model. The details of the hardware and software environment are as follows:

Hardware:

GPU: T4 GPU provided by Google Colab.

RAM: 16GB (Google Colab environment).

Software:

CUDA Version: 10.0 (provided by Google Colab).

Python Version: 3.6.

Deep Learning Framework: PyTorch for YOLOv5.

3. Dataset:

The dataset used for training and evaluation consisted of 1600 images, which were divided into training, validation, and test sets as follows:

Training Set: 1120 images (70%)

Validation Set: 320 images (20%)

Test Set: 160 images (10%)

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

Fig 7: Sample Images from the Dataset

4. Evaluation Metrics:

The performance of the YOLOv5s model was evaluated using the following metrics:

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

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

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

mAP50-95 (mean Average Precision at IoU 0.50 to 0.95): This metric averages the precision scores at multiple IoU thresholds (from 0.50 to 0.95 in increments of 0.05). It gives a comprehensive evaluation of the model's performance across various levels of localization accuracy.

5. Results: The trained YOLOv5s model was evaluated on the test set. The detailed performance metrics, including precision, recall, mAP50, and mAP50-95 for each class, are summarized below:

Table 4: Performance Metrics on Test Data

Metric	Value
Precision (P)	0.898
Recall (R)	0.827
mAP50	0.900
mAP50-95	0.583

Table 3 shows the overall performance of the YOLOv5s model on the test data. Precision and recall values indicate the model's accuracy in

detecting objects correctly, while mAP50 and mAP50-95 metrics provide a comprehensive measure of the model's detection performance across different IoU thresholds.

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	320	557	0.898	0.827	0.9	0.583
bike	320	29	0.912	0.715	0.794	0.446
motobike	320	26	0.907	0.754	0.912	0.605
person	320	71	0.928	0.912	0.956	0.551
traffic_light_green	320	59	0.965	0.941	0.976	0.52
traffic_light_orange	320	24	1	0.758	0.937	0.535
traffic_light_red	320	63	0.919	0.723	0.934	0.521
traffic_sign_30	320	50	0.967	0.96	0.977	0.771
traffic_sign_60	320	36	0.911	0.853	0.909	0.711
traffic_sign_90	320	9	0.58	0.778	0.678	0.506
vehicle	320	190	0.892	0.874	0.925	0.663

Fig 8: Class-Specific Performance on Test Data.

Figure 8 illustrates precision, recall, and mAP metrics for each object class in test dataset. Figure helps in understanding how well model performs for each specific category, highlighting strengths and potential areas for improvement. Trained YOLOv5s model was also evaluated on validation set.

Detailed performance metrics, including precision, recall, mAP50, and mAP50-95 for each class, are summarized below:

Table 5: Performance Metrics on Validation Data.

Metric	Value
Precision (P)	0.891
Recall (R)	0.801
mAP50	0.880
mAP50-95	0.542

Results indicate that the YOLOv5s model achieves high performance in terms of precision, recall, and mAP metrics across various object categories. Table 4 presents overall performance of the YOLOv5s model on the validation data. Similar to the test data results, these metrics provide insights into the model's accuracy and detection capabilities on unseen data used for tuning hyperparameters and early stopping.

Class	Images	Instances	P	R	mAP50	mAP50-95:
all	160	292	0.891	0.801	0.88	0.542
bike	160	15	0.824	0.467	0.65	0.336
motobike	160	16	0.905	0.812	0.872	0.52
person	160	36	0.844	0.889	0.94	0.466
traffic_light_green	160	37	1	0.917	0.943	0.518
traffic_light_orange	160	11	1	0.806	0.906	0.475
traffic_light_red	160	29	0.919	0.785	0.91	0.44
traffic_sign_30	160	19	0.894	0.947	0.947	0.756
traffic_sign_60	160	14	0.868	0.937	0.961	0.714
traffic_sign_90	160	5	0.764	0.6	0.778	0.626
vehicle	160	110	0.887	0.855	0.889	0.57

Fig 9: Class-Specific Performance on Validation Data**.**

Figure 9 depicts the precision, recall, and mAP metrics for each object class in the validation dataset. This figure helps visualize the model's performance across different categories, ensuring the model is well-tuned and generalizes well on new data.

Training Performance: In addition to evaluating the model on test and validation sets, various performance metrics and curves were analyzed during the training process.

Confusion Matrix: Figure 10 shows confusion matrix, providing a detailed look at model's predictions versus actual labels, which helps identify specific classes that may be causing confusion.

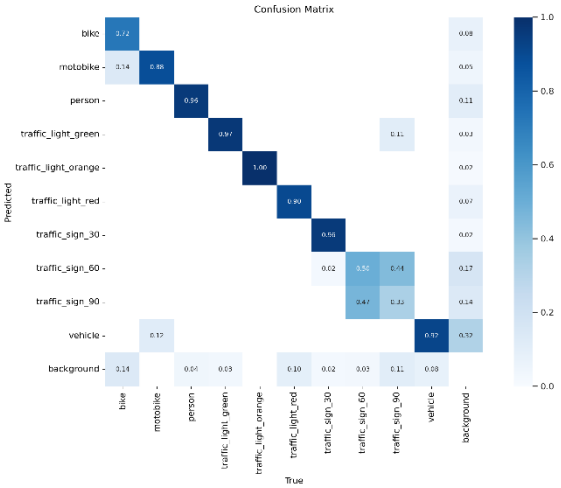

Fig 10: Confusion Matrix.

Precision-Recall Curve: Figure 11 illustrates the precision-recall (PR) curve, showing the trade-off between precision and recall for different thresholds.

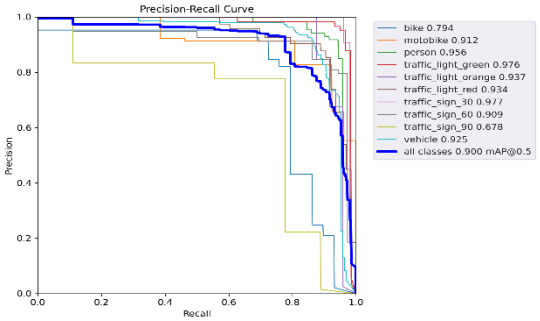

Fig 11: PR Curve.

F1 Curve: Figure 12 shows the F1 curve, highlighting the harmonic mean of precision and recall across different thresholds.

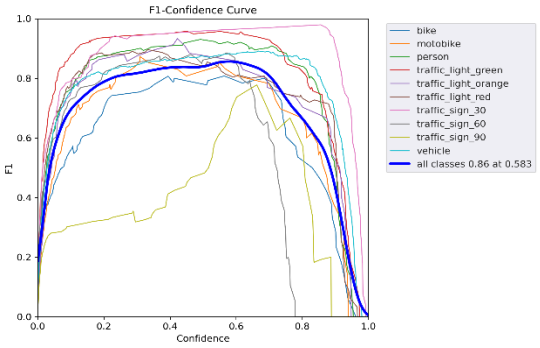

Fig 12: F1 Curve.

Training Results: Figure 13 provides an overview of the training results, including loss and accuracy metrics over epochs, demonstrating the model's learning progress and convergence. These figures together provide a comprehensive overview of the model's performance, both during training and on the evaluation sets, allowing for a thorough analysis of the YOLOv5s model's strengths and areas for potential improvement.

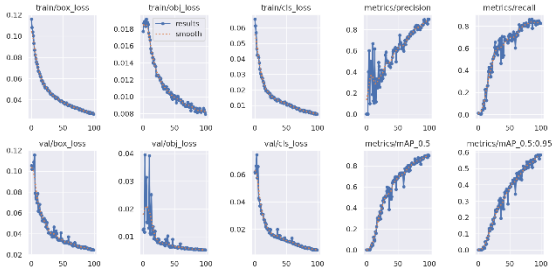

Fig 13: Training Results.

Training Data Samples: To provide an overview of the training data, we present a few samples from the training batches.

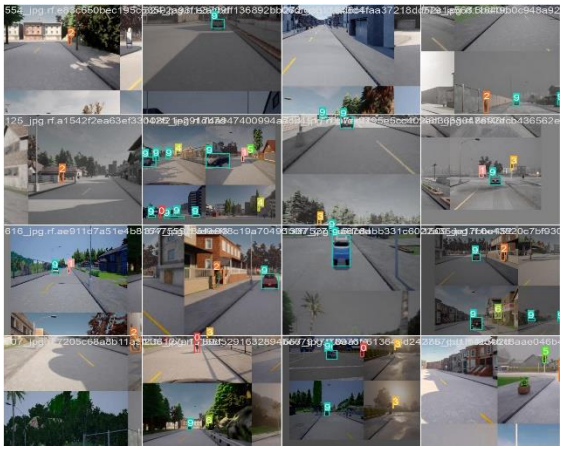

Fig 13: Sample Images from Training Batches.

Validation Data Labels and Predictions To evaluate the model's performance on the validation set, we compare the actual labels with the model's predictions.

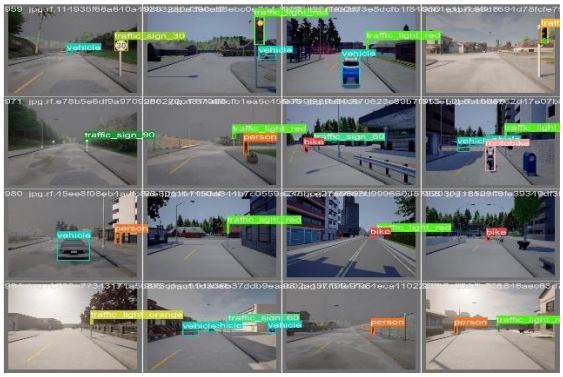

Fig 14: Actual Labels in Validation Batches.

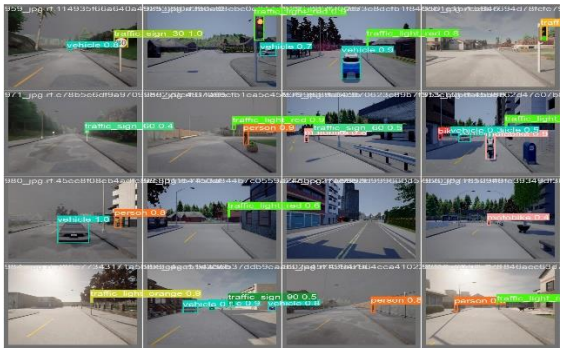

Fig 15: Model Predictions in Validation Batches.

These figures provide a visual comparison of the ground truth and the predictions made by the

model, highlighting its performance and areas where it may need improvement.

Validation Performance: During the validation phase, the model's performance was thoroughly assessed using various performance metrics and visualizations.

Confusion Matrix: Figure 16 presents the confusion matrix generated during the validation phase. This matrix provides a detailed overview of the model's predictions compared to the actual labels in the validation dataset. It helps identify classes where the model may be struggling or making errors, thereby guiding further refinement efforts.

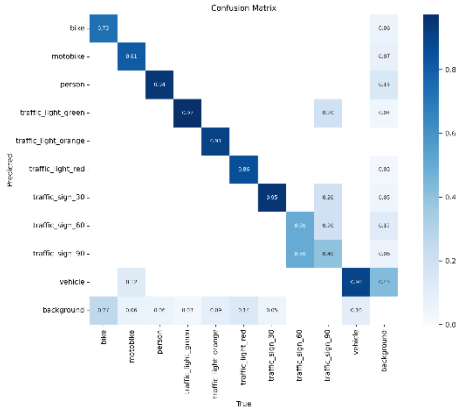

Fig 16: Confusion Matrix.

Precision-Recall Curve: Figure 17 illustrates the precision-recall (PR) curve computed during the validation process. This curve showcases the trade-off between precision and recall at different decision thresholds. It offers insights into how effectively the model balances precision and recall for object detection tasks on the validation dataset.

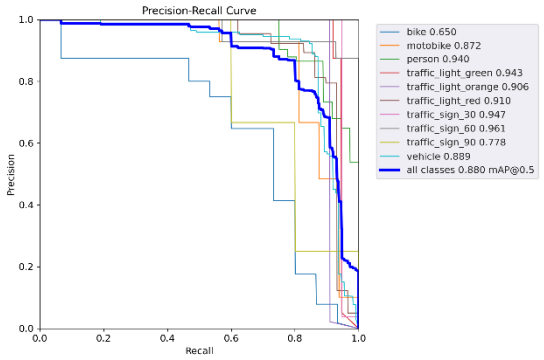

Fig 17: Precision-Recall Curve.

F1 Curve: Figure 18 displays the F1 curve calculated during the validation phase. The F1 curve depicts the harmonic mean of precision and recall across various decision thresholds. Analyzing this curve helps gauge the model's overall performance and convergence during validation.

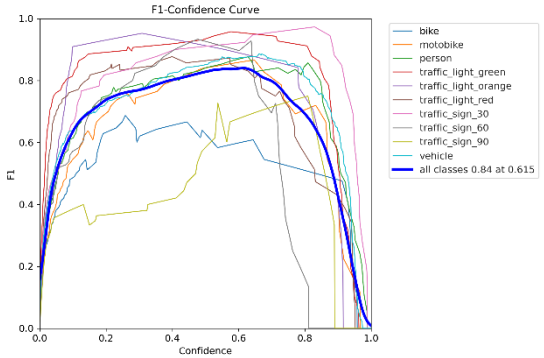

Fig 18: F1 Curve.

Precision Curve: Figure 19 represents the precision curve obtained from the validation phase. This curve plots precision against different decision thresholds and provides a closer look at the model's precision performance on the validation dataset.

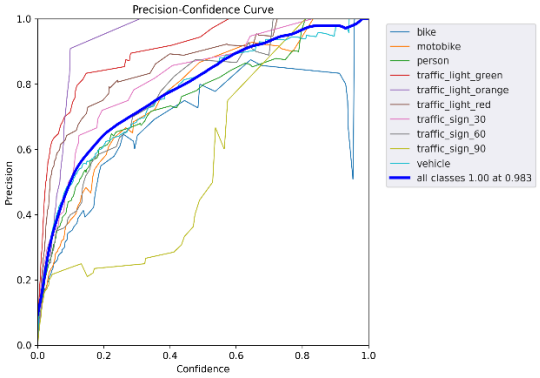

Fig 19: Precision Curve.

Recall Curve: Figure 20 showcases the recall curve derived from the validation phase. By plotting recall against different decision thresholds, this curve offers insights into the model's ability to correctly identify objects of interest across various recall levels during validation.

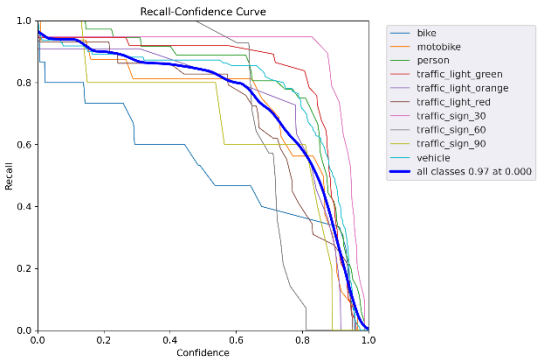

Fig 20: Recall Curve.

These visualizations collectively offer a comprehensive understanding of the YOLOv5s model's performance during the validation phase, facilitating an in-depth analysis of its strengths and areas for potential enhancement.

Results Analysis:

Overall Precision and Recall: We observed that both precision and recall in both training and validation results range around values of 0.89 and 0.8, respectively. This indicates that the model can effectively identify objects with moderate efficiency, but there are some gaps that can be improved.

Mean Average Precision (mAP): The values of mAP50 and mAP50-95 show that the model exhibits good performance in recognizing objects at 50% and 95% IoU (Intersection over Union). However, there seems to be a slight decline in validation performance compared to training.

Individual Class Analysis: The performance of individual classes ranges from good to excellent, with some classes like "traffic_sign_30" and "traffic_light_green" showing excellent performance in both training and validation results. However, there are some classes that appear to perform less well in validation compared to training, such as "bike" and "person".

Speed Analysis: The model appears to respond well with acceptable processing speed, ranging between 0.4 milliseconds for preprocessing and 10.5 milliseconds for inference.

Overall, the model demonstrates good performance on both datasets, but there are some gaps that can be improved, especially in certain classes like "bike" and "person". Performance can be enhanced by addressing some issues such as data imbalance and improving training settings.

Conclusion and Recommendations

In the realm of autonomous driving, accurate object detection is paramount for ensuring the safety and efficiency of vehicles. The YOLOv5 model exhibits commendable performance in identifying objects, laying a solid foundation for its integration into autonomous vehicle systems. However, to fully realize its potential, several key areas warrant attention.

1. Enhanced Safety: A robust object detection system is crucial for the safety of autonomous vehicles, as it enables them to perceive and react to their surroundings accurately. Addressing data imbalances and fine-tuning training settings can bolster the model's ability

to detect objects across diverse scenarios, mitigating risks on the road.

2. Efficient Navigation: Accurate object detection not only enhances safety but also facilitates smoother navigation. By refining class-specific performance and optimizing model complexity, the YOLOv5 model can better discern complex traffic scenarios, improving the vehicle's decision-making capabilities and ensuring seamless interactions with its environment.

3. Real-world Deployment: As autonomous driving technologies inch closer to widespread adoption, the deployment of reliable object detection systems becomes increasingly critical. Therefore, it's imperative to assess the model's performance in real-world scenarios, addressing deployment challenges and refining its effectiveness in practical applications.

In summary, the significance of precise object detection in autonomous vehicles cannot be overstated. By heeding these recommendations and continuously refining the YOLOv5 model, we pave the way for safer, more efficient autonomous driving experiences, bringing us closer to a future where road travel is not only autonomous but also remarkably secure and reliable.

Arabic section:

اكتشاف الأشياء باستخدام الذكاء الاصطناعي في المركبات ذاتية القيادة

بدر نجيب عويدات

الملخص: تركز هذه الورقة على الكشف عن الأجسام، مثل الدراجات الهوائية والدراجات النارية والأشخاص واشارات المرور وعلامات المرور والمركبات ضمن إطار نظام القيادة الذاتية في بيئة.CARLA حاليًا، يعتمد الكشف عن الأجسام في القيادة الذاتية بشكل أساسي على مركبات ذاتية القيادة الحقيقية، التي تواجه تحديات مثل التكاليف العالية وصعوبات التنفيذ في الوقت الفعلي. يمكن نظام CARLA وهو مفتوح المصدر من التجارب الدقيقة والفعّالة من حيث التكلفة. في هذه الورقة البحثية، تم استخدام نموذج التعلم العميقYOLOv5 ، وقد اعطى نتائج جيدة في مجموعات بيانات التدريب والتحقق. تم استخدام مجموعة متنوعة من الصور تبلغ 1560 صورة في عملية التدريب، تم تقسيمها إلى 1120 صورة للتدريب، و160 صورة للاختبار، و320 صورة للتحقق. أظهرت نتائج التدريب دقة (P) بنسبة 0.898، واستدعاء (R) بنسبة 0.827، و mAP@50 بنسبة 0.900، و mAP@50-95 بنسبة 0.583. في نتائج التحقق، كانت الدقة (P) هي 0.891، والاستدعاء (R) هو 0.801، و mAP@50كانت 0.880، و mAP@50-95 كانت 0.542. تشير هذه النتائج إلى أن النموذج قادر على كشف الأجسام بدقة واسترجاعها بفعالية.

الكلمات المفتاحية: كشف الكائنات، الذكاء الاصطناعي، المركبات الذاتية القيادة، كارلا، متوسط الدقة المتوسطة، إشارات المرور، علامات المرور.

Abbreviations and Acronyms

CARLA: Car Learning to Act, LiDAR: Light Detection and Ranging, MASTIF: MArine STation Imaging Facility, R-CNN: Region-based Convolutional Neural Network, QCNNs: Quaternion Convolutional Neural Networks, SVM: Support Vector Machine, mAP: mean Average Precision, FPS: Frames Per Second, YOLOv5: You Only Look Once, VGG16: Visual Geometry Group, CSPNet: Cross Stage Partial Network, P: Precision, R: Recall, mAP50: mean Average Precision: IoU: Intersection over Union,

References

[1] 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.

[2] R. Muller, “Drivetruth: Automated autonomous driving dataset generation for security applications,” in Workshop on Automotive and Autonomous Vehicle Security (AutoSec), 2022.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” in Conference on robot learning. PMLR, 2017, pp. 1–16.

[4] 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.

[5] E. Alcal´a, L. Sellart, V. Puig, J. Quevedo, J. Saludes, D. V´azquez, and A. L´opez, "Comparison of two non-linear model-based control strategies for autonomous vehicles," in 2016 24th Mediterranean Conference on Control and Automation (MED). IEEE, 2016, pp. 846–851.

[6] "CARLA documentation," Online, available: https://carla.readthedocs.io/en/stable/carlasettings/.

[7] M. H. Abdulabas and N. D. Al-Shakarchy, "Person identification based on facial biometrics in different lighting conditions," International Journal of Electrical and Computer Engineering (IJECE), vol. 13, no. 2, pp. 2086–2092, 2023.

[8] W. Gao, J. Tang, and T. Wang, "An object detection research method based on CARLA simulation," in *Journal of Physics: Conference Series*, vol. 1948, no. 1, pp. 012163, 2021, doi: 10.1088/1742-6596/1948/1/012163.

[9] 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