SEBHA UNIVERSITY CONFERENCE PROCEEDINGS, VOL.03 NO. 3 2024
DOI: 10.51984/SUCP.V3I3.3694

المؤتمر السادس للعلوم الهندسية والتقنية
The Sixth Conference for Engineering Sciences and Technology (CEST-6)
Confrence Proceeeding homepage: https://cest.org.ly

Efficient Object Detection in Autonomous Driving Systems Using YOLOv5 and CARLA Simulator

Bader N. Awedat

Al-Zaytouna University, Faculty of Information Technology, Tarhuna, Libya

Keywords:
Autonomous Driving
Object Detection
CARLA Simulator
YOLOv5
Deep Learning Models

A B S T R A C T

One of primary challenges in autonomous driving is high cost of electronic components, which can hinder widespread adoption and experimentation necessary for advancements in this field. Open-source CARLA simulator provides a cost-effective and realistic environment for conducting experiments in autonomous driving, allowing for precise and efficient testing without need for expensive hardware .In this study, we focus on object detection within autonomous driving systems using CARLA simulator. Deep learning model YOLOv5 was employed to detect ten different objects: bike, motorcycle, person, traffic light green, traffic light orange, traffic light red, traffic sign 30, traffic sign 60, traffic sign 90, and vehicle. Model was trained for 150 epochs using a dataset of 1864 images, divided into 1600 images for training, 264 images for testing. Training results for all classes were Precision (P) of 0.934, Recall (R) of 0.908, mAP@50 of 0.935 and mAP@50-95 of 0.689. Test results for all classes were Precision (P) of 0.93, Recall (R) of 0.892, mAP@50 of 0.93 and mAP@50-95 of 0.675. These results demonstrate model's capability to accurately detect and retrieve objects. Additionally, external testing on model with new images showed good performance, successfully recognizing objects in various scenarios .This research highlights potential of using CARLA simulator and YOLOv5 model for efficient and effective object detection in autonomous driving systems, paving way for further advancements in this critical field.

كشف الكائنات الفعّال في أنظمة القيادة الذاتية باستخدام YOLOv5 ومحاكي CARLA

بدر نجيب عويدات

جامعة الزيتونة، كلية تقنية المعلومات، ترهونة، ليبيا

الكلمات المفتاحية:
القيادة الذاتية
كشف الكائنات
محاكي *CARLA*
YOLOv5
نماذج التعلم العميق

الملخص

تُعد تكلفة المكونات الإلكترونية العالية من التحديات الأساسية في القيادة الذاتية، مما قد يعيق الاستخدام الواسع والتجارب اللازمة لتحقيق التقدم في هذا المجال. يوفر محاكي CARLA مفتوح المصدر بيئة فعّالة من حيث التكلفة وواقعية لإجراء التجارب في القيادة الذاتية، مما يسمح بإجراء اختبارات دقيقة وفعّالة دون الحاجة إلى أجهزة باهظة الثمن. في هذه الدراسة، نركز على كشف الكائنات داخل أنظمة القيادة الذاتية باستخدام محاكي CARLA. تم استخدام نموذج التعلم العميق YOLOv5 لكشف عشرة كائنات مختلفة: دراجة، دراجة نارية، أشخاص، إشارة مرور خضراء، إشارة مرور صفراء، إشارة مرور حمراء، علامة مرور 30، علامة مرور 60، علامة مرور 90، ومركبات. تم تدريب النموذج 150 دورة باستخدام مجموعة بيانات تتكون من 1864 صورة، تم تقسيمها إلى 1600 صورة للتدريب و264 صورة للاختبار. كانت نتائج التدريب لجميع الفئات هي: الدقة 0.934، الاسترجاع 0.908، متوسط الدقة عند 50 0.935، ومتوسط الدقة عند 50-95 0.689. أما نتائج الاختبار لجميع الفئات فكانت: الدقة 0.93، الاسترجاع 0.892، متوسط الدقة عند 50 0.93، ومتوسط الدقة عند 50-95 0.675. تُظهر هذه النتائج قدرة النموذج على كشف واسترجاع الكائنات بدقة. بالإضافة إلى ذلك، أظهرت الاختبارات الخارجية على النموذج باستخدام صور جديدة أداءً جيدًا، حيث تم التعرف بنجاح على الكائنات في سيناريوهات متنوعة. تُبرز هذه الدراسة إمكانيات استخدام محاكي CARLA

*Corresponding author:
E-mail addresses: bader_najep@yahoo.com
Article History : Received 14 March 2024 - Received in revised form 29 August 2024 - Accepted 21 October 2024

ونموذج YOLOv5 للكشف الفعّال والناجح عن الكائنات في أنظمة القيادة الذاتية، مما يمهد الطريق لمزيد من التقدم في هذا المجال الحيوي.

1. Introduction

Object detection for autonomous driving is a cornerstone technology critical for ensuring safe, efficient, and reliable vehicle operation. Autonomous vehicles rely on accurate detection and classification of objects in their environment to make informed decisions, essential for collision avoidance and safe driving. This task encompasses several critical requirements, including high accuracy, real-time inference speed, small model size, and energy efficiency. Each of these factors significantly impacts effectiveness and safety of autonomous systems.

2. Previous Advancements

2.1 Weber et al. (2016) introduced DeepTLR, a deep convolutional network designed for real-time detection and classification of traffic lights [1]. This model established a foundation for subsequent enhancements in traffic light detection, addressing initial limitations of earlier models.

2.2 Choi et al. (2019) proposed Gaussian YOLOv3, an improvement that incorporates Gaussian parameters into bounding box modeling to enhance detection accuracy and support real-time operation [2]. This advancement addressed several limitations observed in prior models, paving way for more robust object detection in complex environments.

2.3 Sharma et al. (2022) showcased YOLOv5's effectiveness for object detection and scene perception under diverse weather conditions, including challenging scenarios like heavy rain [3]. Their study highlighted model's adaptability in real-time scenarios, demonstrating its capability to handle variations in environmental conditions.

2.4 Kim et al. (2023) addressed challenge of detecting objects in heavy rain by developing a novel dataset from experimental raindrop data. This dataset, evaluated using CARLA simulator, facilitated extensive testing of YOLO-Series models under different precipitation conditions. Study emphasized persistent difficulties encountered by YOLO-Series models in extreme weather scenarios [4].

3. Research Significance

Ongoing research in object detection for autonomous driving focuses not only on refining existing technologies but also on tackling new challenges arising from diverse environmental conditions. Addressing these challenges is crucial for advancing autonomous vehicle systems and enhancing road safety. This research aims to contribute to field by exploring innovative solutions and refining current methodologies to improve object detection performance under various conditions.

Table 1: Performance Metrics from Weber et al. (2016).

Model DeepTLR	Precision (640×480)	Recall (640×480)	F1-Score (640×480)	Precision (1280×960)	Recall (1280×960)	F1-Score (1280×960)
Base	70.4%	66.1%	68.2%	93.8%	73.1%	82.2%
Pretrained	70.6%	68.3%	69.4%	94.8%	78.1%	85.6%
Small Samples	78.6%	87.4%	82.7%	93.7%	91.0%	92.4%
Full	85.6%	90.7%	88.1%	95.6%	91.4%	93.5%

Table 2: Performance Metrics from Choi et al. (2019).

Dataset	Input Size	mAP (%)	FPS
KITTI Validation	512×512	83.61	43.13
KITTI Validation	704×704	86.79	24.91
BDD Test	512×512	18.4	42.5
BDD Test	736×736	20.8	22.5
COCO	N/A	36.1	N/A
COCO (AP75)	N/A	39.0	N/A

Table 3: Performance Metrics from Sharma et al. (2022).

Class	Images	Labels	Precision	Recall	mAP (0.5)
all	239	1520	0.474	0.337	0.258
biker	239	27	0.438	0.0741	0.0811
car	239	1066	0.528	0.726	0.723
pedestrian	239	156	0.22	0.308	0.186
Traffic Light	239	41	0.308	0.415	0.297
Traffic Light-Green	239	42	0.133	0.476	0.0956
Traffic Light Green Left	239	4	1	0	0.00756
Traffic Light Red	239	91	0.378	0.714	0.468
Traffic Light Red Left	239	24	0.2	0.0833	0.128
Traffic Light Yellow	239	12	1	0	0.0169
truck	239	57	0.532	0.578	0.573

Table 4: Comparison of KITTI and CARLA dataset on normal weather condition from Kim et al. (2023).

Model	Class	KITTI AP	KITTI F1	CARLA AP	CARLA F1
YOLOv4	Car	0.805	0.555	0.64	0.628
YOLOv4	Person	0.569	0.379	0.771	0.699
YOLOv5x	Car	0.805	0.784	0.652	0.623
YOLOv5x	Person	0.559	0.602	0.738	0.746
YOLOv7x	Car	0.808	0.792	0.669	0.663
YOLOv7x	Person	0.561	0.603	0.731	0.786

Table 5: AP score degradation of object detection performance according to precipitation from Kim et al. (2023).

Model	Condition	Car	Person	Bicycle	Motorcycle
YOLOv4	Normal	0.64	0.771	0.913	0.263
YOLOv4	100mm/h	0.424	0.469	0.677	0.0043
YOLOv5x	Normal	0.652	0.738	0.813	0.425
YOLOv5x	100mm/h	0.452	0.427	0.614	0.0031
YOLOv7x	Normal	0.669	0.731	0.936	0.198
YOLOv7x	100mm/h	0.46	0.51	0.693	0.00025

Table 6: F1 score degradation of object detection performance according to precipitation from Kim et al. (2023).

Model	Condition	Car	Person	Bicycle	Motorcycle
YOLOv4	Normal	0.629	0.699	0.802	0.179
YOLOv4	100mm/h	0.477	0.414	0.627	0
YOLOv5x	Normal	0.623	0.746	0.724	0.415
YOLOv5x	100mm/h	0.514	0.485	0.535	0
YOLOv7x	Normal	0.663	0.786	0.859	0.159
YOLOv7x	100mm/h	0.533	0.549	0.522	0

4. CARLA (CAR LEARNING ACT)

CARLA, an open-source simulator for autonomous driving research. CARLA has been developed from 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 [5].

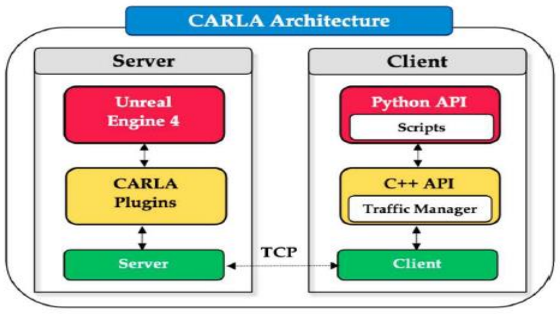


Fig. 1: CARLA client-server architecture [6].

5. Architecture of Autonomous Vehicle Systems in CARLA Environment

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

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

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

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

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

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

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

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

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

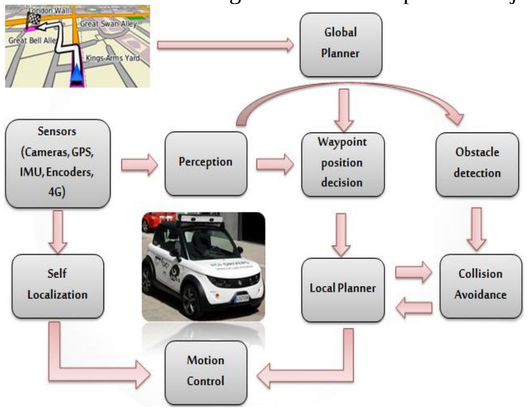


Fig. 2: System architecture of autonomous vehicle [7].

6. Weather Presets:

In CARLA, weather and lighting conditions can be customized through a selection of predefined settings. To choose a specific preset, adjust "WeatherId" key in configuration file "CarlaSettings.ini." 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 CARLA environment. To apply a specific setting, refer to corresponding numerical identifier when configuring WeatherId parameter.

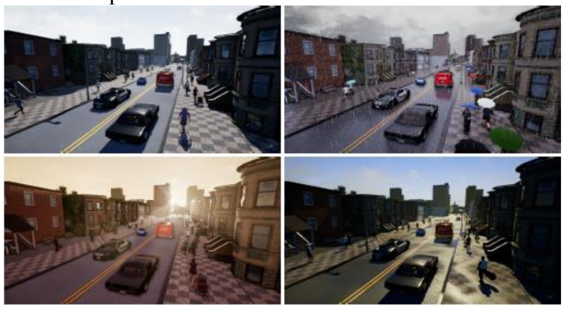


Fig.3: Some weather conditions in CARLA.

7. Camera In Carla:

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

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

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

7.3 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.

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

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

7.6 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 [8].

8. YOLOV5:

YOLOv5 was proposed in 2020 by a person named Glenn Jocher. Model used in 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:

8.1 Speed: YOLOv5 is characterized by its high speed in image processing and object detection compared to many other models.

8.2 Accuracy: It offers a good balance between speed and accuracy, making it suitable for real-time object detection applications.

8.3 Ease of use: It provides an easy-to-use interface and comes with numerous ready-to-use examples, facilitating training and application process.

8.4 Customization: Model can be easily modified to suit a wide range of applications by changing parameters and retraining model on customized datasets.

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

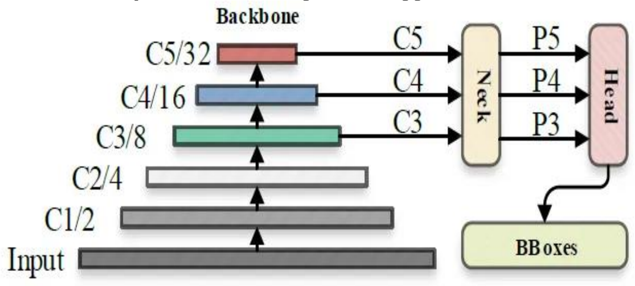


Fig. 4: default inference flowchart of YOLOv5[9].

9. Network Structure of Yolov5:

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

9.1 Backbone: Backbone of YOLOv5 is shown in Figure 5. Main structure is 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 feature expression ability of backbone [9].

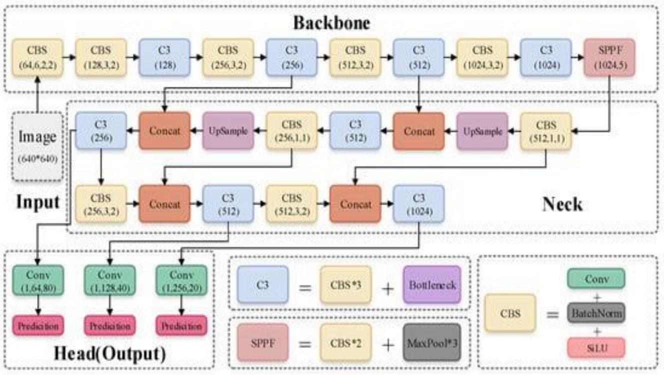


Fig. 5: Default network structure of YOLOv5[9].

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

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

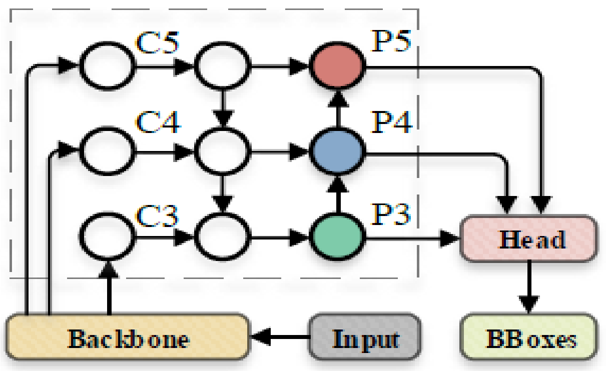


Fig. 6. Neck [9].

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].

10. Experiment and Result Analysis:

10.1 Parameter settings: Default hyperparameters for model were:

Table 7: hyperparameters for model.

epochs	VALUES
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	8
epochs	150

10. YOLOv5s model summary:

YOLOv5s model summary is as follows is:

10.1 Backbone: backbone is core feature extractor of YOLOv5 model, responsible for processing input images through a series of convolutional and residual layers to extract hierarchical features used for object detection.

Tabl 8: Backbone summary.

Index	Module	Number	Arguments	From
0	Conv	1	[64, 6, 2, 2]	-1
1	Conv	1	[128, 3, 2]	-1
2	C3	3	[128]	-1
3	Conv	1	[256, 3, 2]	-1
4	C3	6	[256]	-1
5	Conv	1	[512, 3, 2]	-1
6	C3	9	[512]	-1
7	Conv	1	[1024, 3, 2]	-1
8	C3	3	[1024]	-1
9	SPPF	1	[1024, 5]	-1

10.2 Head: head of YOLOv5 model takes extracted features from backbone and performs object detection tasks, involving upsampling, concatenation, and detection layers to predict bounding boxes and class probabilities.

Table 9: Head Summary.

Index	From	Number	Module	Arguments
10	-1	1	Conv	[512, 1, 1]
11	-1	1	nn.Upsample	[None, 2, "nearest"]
12	[-1, 6]	1	Concat	[1]
13	-1	3	C3	[512, False]
14	-1	1	Conv	[256, 1, 1]
15	-1	1	nn.Upsample	[None, 2, "nearest"]
16	[-1, 4]	1	Concat	[1]
17	-1	3	C3	[256, False]
18	-1	1	Conv	[256, 3, 2]
19	[-1, 14]	1	Concat	[1]
20	-1	3	C3	[512, False]
21	-1	1	Conv	[512, 3, 2]
22	[-1, 10]	1	Concat	[1]
23	-1	3	C3	[1024, False]
24	[17, 20, 23]	1	Detect	[80, anchors]

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

11.1 Hardware

11.1.1 GPU: T4 GPU provided by Google Colab.

11.1.2 RAM: 16GB (Google Colab environment).

11.2 Software:

11.2.1 CUDA Version: 10.0 (provided by Google Colab).

11.2.2 Python Version: 3.7 for run CARLA Environment.

11.2.3 Deep Learning Framework: PyTorch for YOLOv5.

11.3 Dataset: Dataset used for training and evaluation consisted of 1864 images, which were divided into training and test sets as follows:

11.3.1 Training Set: 1600 images (80%)

11.3.2 Test Set: 264 images (20%)

11.3.3 Validation Set: A diverse external set of images was used for validation to assess model's performance and ensure it generalizes well across various scenarios and object instances. Dataset included annotations for various object categories, which were used to train and evaluate model's performance.

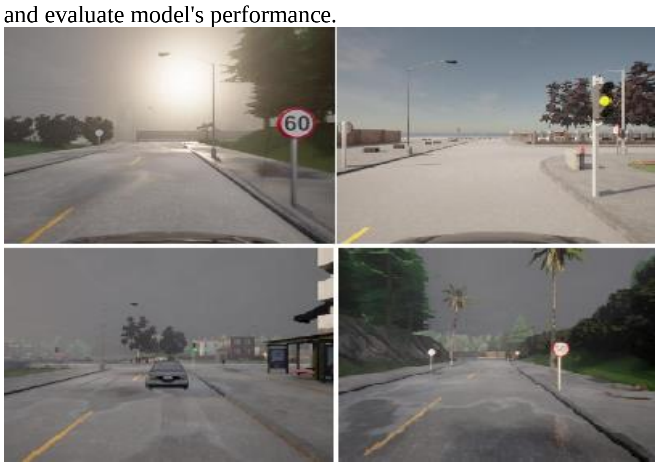


Fig. 7: Sample Images from Dataset.

11. EVALUATION METRICS:

Performance of YOLOv5s model was evaluated using following metrics:

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

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

11.3 mAP50 (mean Average Precision at IoU 0.50): These 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.

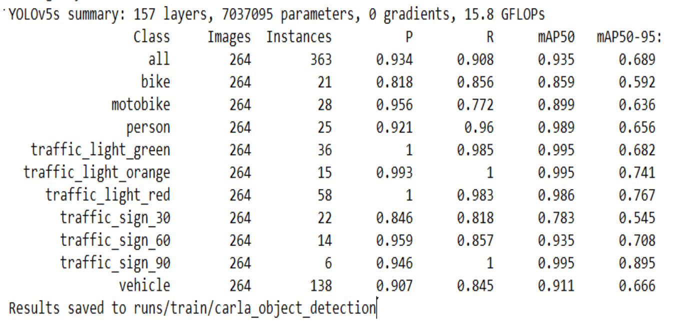
11.4 mAP50-95 (mean Average Precision at IoU 0.50 to 0.95): These metric averages precision scores at multiple IoU thresholds (from 0.50 to 0.95 in increments of 0.05). It gives a comprehensive evaluation of model's performance across various levels of localization accuracy.

12. Results: Trained YOLOv5s model was evaluated on training set. Detailed performance metrics, including precision, recall, mAP50, and mAP50-95 for each class, are summarized below:

Table 10: Performance Metrics on Test Data.

Metric	Value
Precision (P)	0.934
Recall (R)	0.908
mAP50	0.935
mAP50-95	0.689

Table 10 shows overall performance of YOLOv5s 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.



```
YOLOv5s summary: 157 layers, 7037095 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95:
                   all        264        363      0.934      0.908      0.935      0.689
                  bike        264         21      0.818      0.856      0.859      0.592
              motobike        264         28      0.956      0.772      0.899      0.636
                person        264         25      0.921       0.96      0.989      0.656
   traffic_light_green        264         36          1      0.985      0.995      0.682
  traffic_light_orange        264         15      0.993          1      0.995      0.741
     traffic_light_red        264         58          1      0.983      0.986      0.767
       traffic_sign_30        264         22      0.846      0.818      0.783      0.545
       traffic_sign_60        264         14      0.959      0.857      0.935      0.708
       traffic_sign_90        264          6      0.946          1      0.995      0.895
               vehicle        264        138      0.907      0.845      0.911      0.666
Results saved to runs/train/carla_object_detection
```

Fig. 8: Class-Specific Performance on Training.

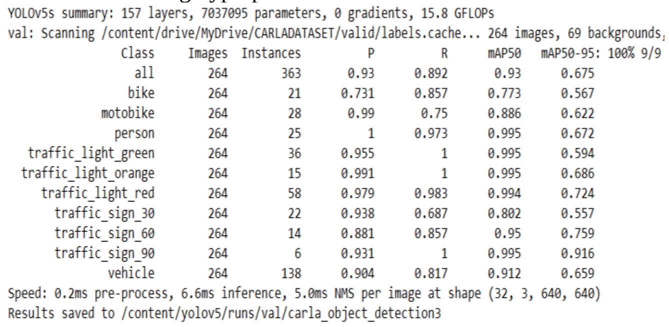
Figure 8 illustrates precision, recall, and mAP metrics for each object class in Training dataset. Figure helps in understanding how well model performs for each specific category, highlighting strengths and potential areas for improvement. Trained YOLOv5s model was also evaluated on validation set.

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

Table 11: Performance Metrics on Test Data.

Metric	Value
Precision (P)	0.93
Recall (R)	0.892
mAP50	0.93
mAP50-95	0.675

Results indicate that YOLOv5s model achieves high performance in terms of precision, recall, and mAP metrics across various object categories. Table 11 presents overall performance of YOLOv5s model on validation data. Similar to test data results, these metrics provide insights into model's accuracy and detection capabilities on unseen data used for tuning hyperparameters.



```
YOLOv5s summary: 157 layers, 7037095 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning /content/drive/MyDrive/CARLADATASET/valid/labels.cache... 264 images, 69 backgrounds,
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 9/9
                   all        264        363       0.93      0.892       0.93      0.675
                  bike        264         21      0.731      0.857      0.773      0.567
              motobike        264         28       0.99       0.75      0.886      0.622
                person        264         25          1      0.973      0.995      0.672
   traffic_light_green        264         36      0.955          1      0.995      0.594
  traffic_light_orange        264         15      0.991          1      0.995      0.686
     traffic_light_red        264         58      0.979      0.983      0.994      0.724
       traffic_sign_30        264         22      0.938      0.687      0.802      0.557
       traffic_sign_60        264         14      0.881      0.857       0.95      0.759
       traffic_sign_90        264          6      0.931          1      0.995      0.916
               vehicle        264        138      0.904      0.817      0.912      0.659
Speed: 0.2ms pre-process, 6.6ms inference, 5.0ms NMS per image at shape (32, 3, 640, 640)
Results saved to /content/yolov5/runs/val/carla_object_detection3
```

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

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

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

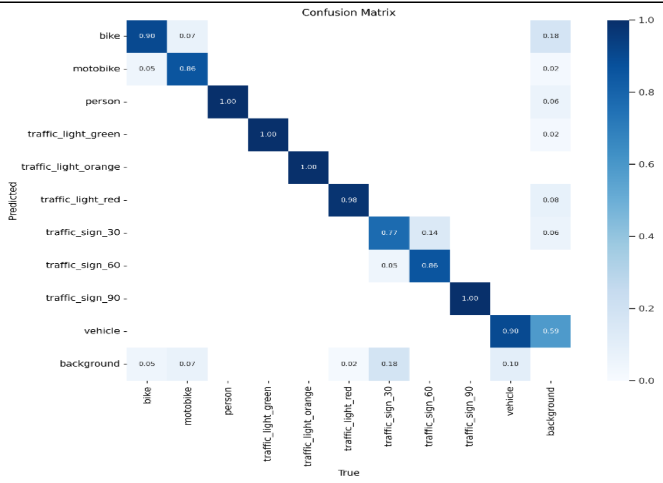


Fig. 10: Confusion Matrix.

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

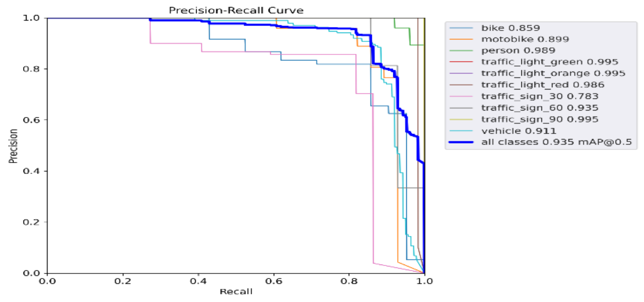


Fig. 11: PR Curve.

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

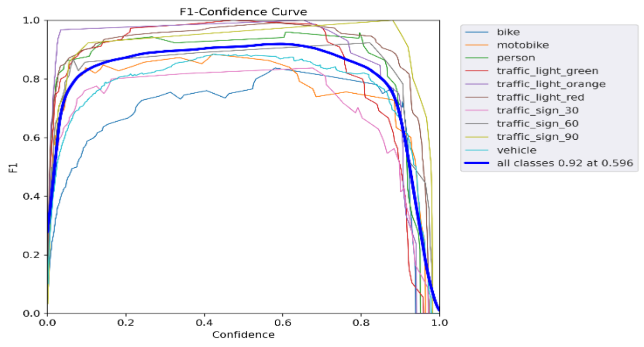


Fig. 12: F1 Curve.

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

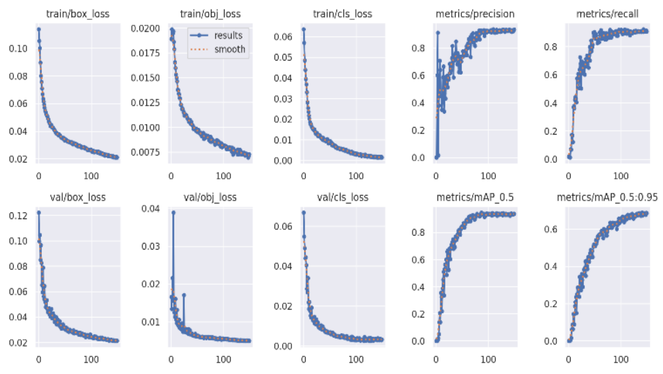


Fig. 13: Training Results.

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

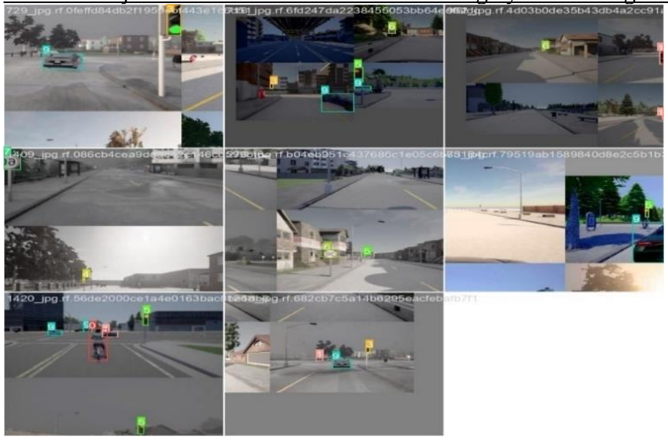


Fig. 14: Sample Images from Training Batches.

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

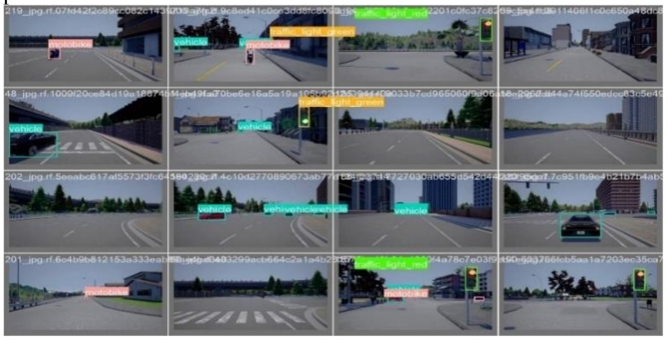


Fig. 15: Actual Labels in Validation Batches.

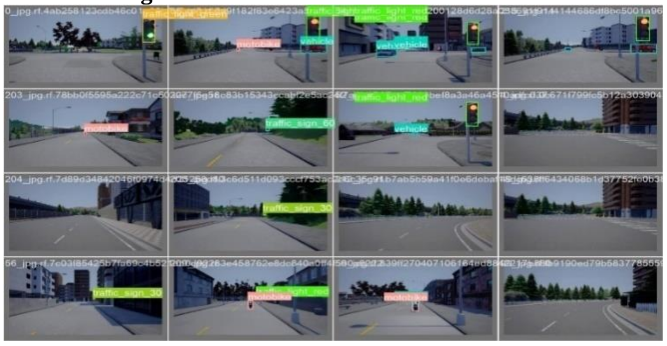


Fig. 16: Model Predictions in Validation Batches.

These figures provide a visual comparison of ground truth and predictions made by model, highlighting its performance and areas where it may need improvement.

Test Performance: During testing phase, model's performance was thoroughly assessed using various performance metrics and visualizations.

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

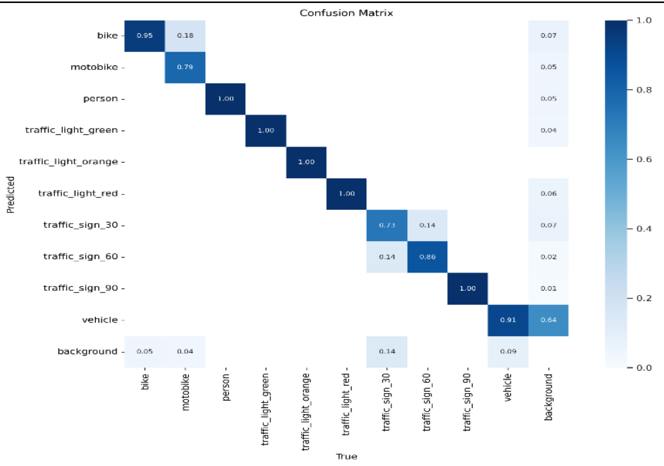


Fig. 17: Confusion Matrix.

Precision-Recall Curve: Figure 18 illustrates precision-recall (PR) curve computed during validation process. This curve showcases trade-off between precision and recall at different decision thresholds. It offers insights into how effectively model balances precision and recall for object detection tasks on validation dataset.

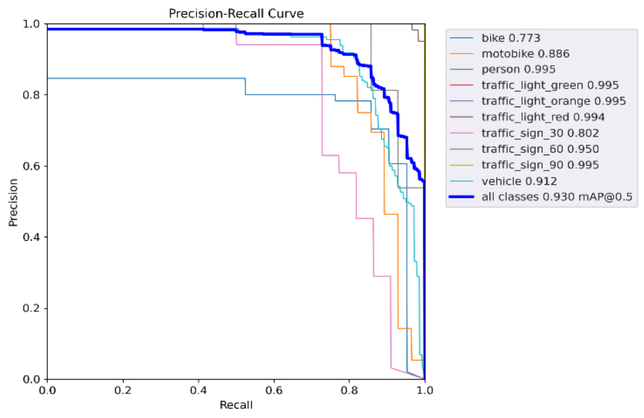


Fig. 18: Precision-Recall Curve.

F1 Curve: Figure 19 displays F1 curve calculated during validation phase. F1 curve depicts harmonic mean of precision and recall across various decision thresholds. Analyzing this curve helps gauge model's overall performance and convergence during validation.

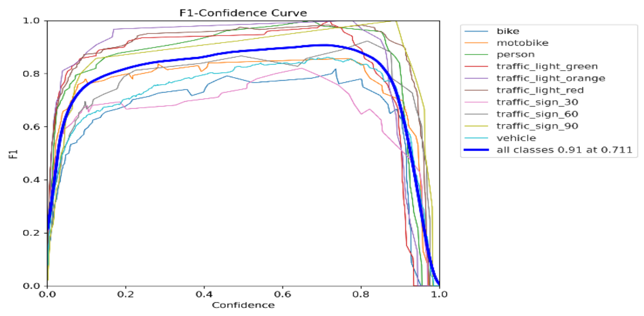


Fig. 19: F1 Curve.

Precision Curve: Figure 20 represents precision curve obtained from validation phase. This curve plots precision against different decision thresholds and provides a closer look at model's precision performance on validation dataset.

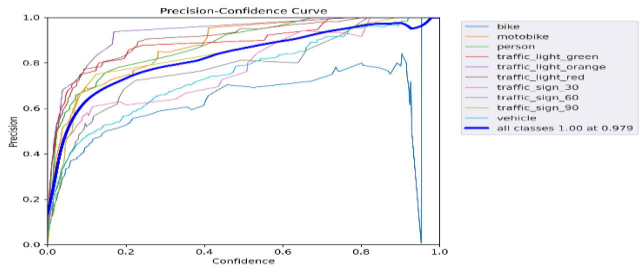


Fig. 20: Precision Curve.

Recall Curve: Figure 21 show cases recall curve derived from validation phase. By plotting recall against different decision thresholds, this curve offers insights into model's ability to correctly identify objects of interest across various recall levels during validation.

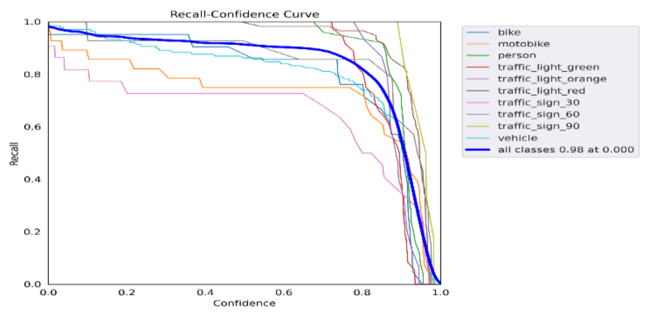
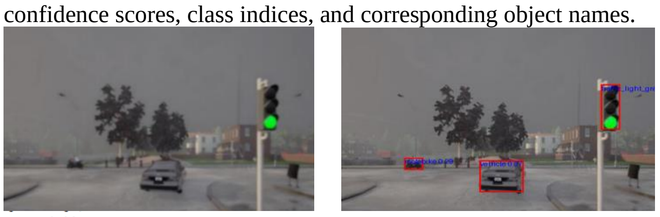


Fig. 21: Recall Curve.

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

External Validation using Trained Model on Unseen Images: In this section, we present validation of our trained YOLOv5 model using external images that were not part of training or testing datasets. This step is crucial to evaluate generalization capability of model in detecting objects in unseen data, ensuring its robustness and reliability in real-world scenarios.

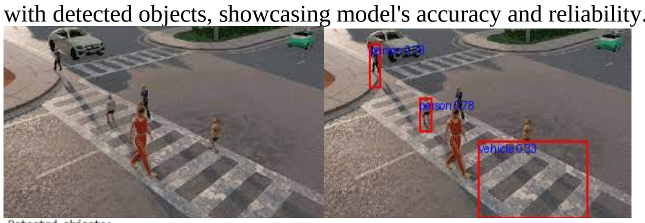
To perform validation, we loaded trained model and applied it to a set of external images. Model processed each image and produced bounding boxes around detected objects, along with confidence scores and class labels. Table below showcases results for a sample image, detailing coordinates of bounding boxes (xmin, ymin, xmax, ymax), confidence scores, class indices, and corresponding object names.



```
Detected objects:
      xmin       ymin       xmax       ymax   confidence   class                name
   183.728    218.180    241.207    270.370        0.968       9             vehicle
   344.912     93.631    369.925    167.828        0.932       3  traffic_light_green
    83.693    214.090    108.126    233.109        0.288       1            motobike
Inference time: 0.68343186378479 seconds
تم حفظ الصورة المعدلة في '/content/sample_data/output.png'.
```

Results indicate that model effectively detects and classifies objects such as vehicles, traffic lights, and motorbikes with high confidence. This external validation step demonstrates practical application of our model and its potential for deployment in real-world autonomous driving scenarios.

We have included representative images with annotated detections to visually illustrate performance of our model. Each image is labeled with detected objects, showcasing model's accuracy and reliability.



```
Detected objects:
      xmin       ymin       xmax       ymax   confidence   class      name
    93.026     61.935    105.408     91.470        0.779       2    person
    44.621     13.479     55.367     52.453        0.779       2    person
   150.689     99.900    257.885    168.000        0.331       9   vehicle
Inference time: 0.31244635581970215 seconds
تم حفظ الصورة المعدلة في '/content/sample_data/output.png'.
```

Fig. 23: Detected Objects with Confidence Scores and Bounding Boxes.

13. Results Analysis: Model demonstrates high precision and recall in both training and testing phases. A slight decrease in recall and a minor difference in mAP@50-95 between training and testing indicate consistent performance across different datasets, suggesting model's stability and minimal susceptibility to bias.

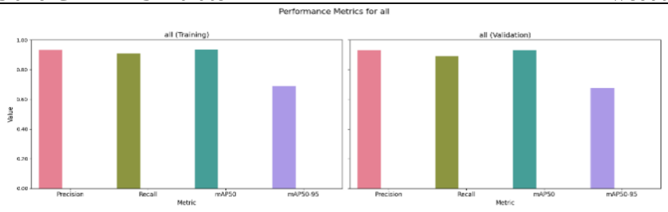


Fig. 24: Performance metrics for all.

13.1 Class-Specific Performance:

Following results highlight model's performance across various object classes, showcasing its precision and recall during both training and testing phases:

13.2 High Precision and Recall:

These results indicate that model performs exceptionally well in detecting certain objects, showing high accuracy in both training and testing phases.

13.2.1 Traffic light green:

Precision: 1.0 (Training), 0.955 (Testing); Recall: 0.985 (Training), 1.0 (Testing).

13.2.2 Traffic light orange:

Precision: 0.993 (Training), 0.991 (Testing); Recall: 1.0 (Training and Testing).

13.2.3 Traffic light red:

Precision: 1.0 (Training), 0.979 (Testing); Recall: 0.983 (Training and Testing).

13.2.4 Person:

Precision: 0.921 (Training), 1.0 (Testing); Recall: 0.96 (Training), 0.973 (Testing).

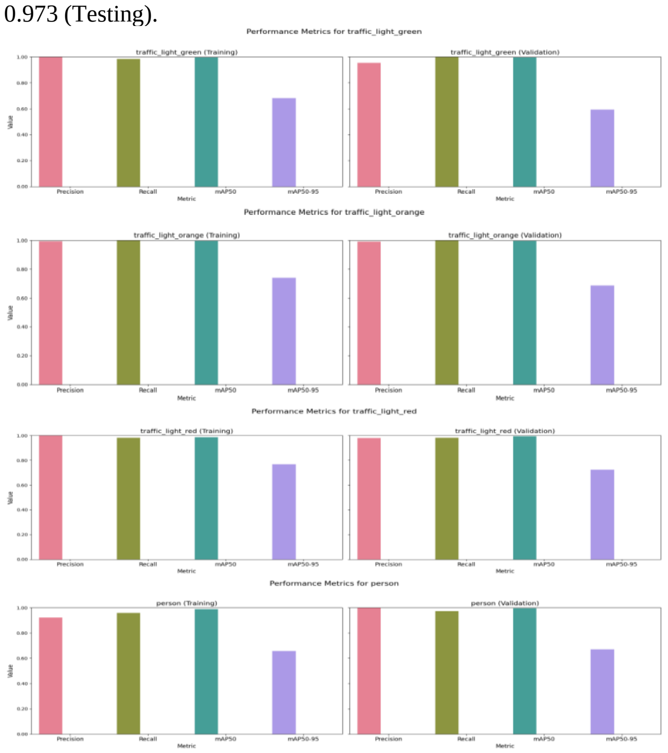


Fig. 25: High Precision and Recall.

13.3 Moderate Performance:

For certain classes, model shows moderate performance with slightly varying precision and recall between training and testing phases:

13.3.1 bike:

Precision: 0.818 (Training), 0.731 (Testing); Recall: 0.856 (Training), 0.857 (Testing).

13.3.2 Motobike:

Precision: 0.956 (Training), 0.99 (Testing); Recall: 0.772 (Training), 0.75 (Testing).

13.4 Inference Speed: model demonstrates excellent execution speed, making it well-suited for real-time applications such as autonomous driving. High speed ensures efficient image processing, a critical requirement for real-time systems.

13.5 Speed Details: Pre-processing takes 0.2ms, inference requires 6.6ms, and Non-Maximum Suppression (NMS) consumes 5.0ms per image at a resolution of (32, 3, 640, 640).

14. Discussion

Previous studies have made significant strides in enhancing object detection for autonomous driving. Weber et al. introduced DeepTLR, which demonstrated competitive performance metrics in real-time traffic light detection and classification [1]. This approach set a benchmark for subsequent advancements.

Choi et al. further advanced field with Gaussian YOLOv3, utilizing Gaussian parameters to refine detection accuracy and speed, especially in complex driving environments [2]. Sharma et al. expanded on this by showcasing YOLOv5's robustness under diverse weather conditions, including challenging scenarios like heavy rain [3].

In current study, training and testing YOLOv5 on CARLA dataset revealed strong performance metrics across multiple classes. High precision and recall were observed, particularly in detecting traffic lights and pedestrians. However, performance challenges persisted for smaller objects such as bicycles and motorcycles. These findings suggest that while YOLOv5 performs well in many scenarios, further optimization is needed to improve detection accuracy for smaller and more variable objects.

When comparing these results with Kim et al.'s findings on YOLO-Series models under heavy rain conditions, we observe similar challenges. Kim et al. identified significant performance degradation in adverse weather, a challenge that our study also encountered [4]. This parallel highlight ongoing difficulty in maintaining high detection accuracy in such conditions, underscoring need for continued research to enhance robustness.

In conclusion, advancements in deep learning models like YOLOv5 show considerable promise for real-time object detection in autonomous driving. Nonetheless, addressing specific challenges—such as detecting smaller objects and performing reliably under adverse weather conditions—remains crucial. Future research should focus on refining these aspects to enhance overall effectiveness and safety of autonomous driving systems.

15. Conclusion

Model demonstrates strong overall performance with high precision and recall in both training and testing phases. Excellent performance in traffic signal and pedestrian categories highlights model's capability in recognizing these classes. However, some categories like bicycles and motorcycles could benefit from further improvements. High execution speed enhances model's efficiency for real-time applications.

16. Recommendations

15.1 A. Utilizing Attention Mechanisms to Enhance Model Accuracy and Pruning and Trimming Techniques to Reduce Model Size: It is advisable to explore attention mechanisms to improve model's accuracy by focusing on relevant features. Additionally, using pruning and trimming techniques can reduce model's size, making it easier to deploy in autonomous driving environments like CARLA.

15.2 B. Training Model Locally on a Computer Instead of Google Colab: Training model on a local machine offers several advantages, including integrating trained model with graphical libraries such as tkinter or qt5. This increases model's portability and facilitates its integration into graphical interface applications, easing testing and deployment in simulation environments like CARLA.

15.3 C. Using Larger and Diverse Weather Data Sets: Increasing volume and diversity of data used in training is crucial to improve model's generalization capabilities under varied weather conditions. Gathering additional data that represents a wide range of weather conditions such as heavy rain, fog, and bright sunlight will enhance model's ability to accurately detect targets in diverse and changing road conditions.

References

[1]- M. Weber, P. Wolf, and J. M. Zöllner, "DeepTLR: A Single Deep Convolutional Network for Detection and Classification of Traffic Lights," in *Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 342-348. (IF: 4).

[2]- J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. [Online]. Available: https://arxiv.org/abs/1904.04620. [Accessed: Apr. 9, 2019]. [Revised: Aug. 12, 2019].

[3]- Sharma, T.; Debaque, B.; Duclos, N.; Chehri, A.; Kinder, B.; Fortier, P. Deep Learning-Based Object Detection and Scene Perception under Bad Weather Conditions. Electronics 2022, 11, 563. https://doi.org/10.3390/ electronics11040563.

[4]- T. Kim, H. Jeon, and Y. Lim, "Challenges of YOLO Series for Object Detection in Extremely Heavy Rain: CALRA Simulator based Synthetic Evaluation Dataset," arXiv e-prints, vol. 2312, Dec. 2023.the English citation first, followed by the original foreign-language citation [6].

[5]- 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.

[6]- 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.

[7]- 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.

[8]- "CARLA documentation," Online, available: https://carla.readthedocs.io/en/stable/carlas ettings/.

[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