Real-Time Vehicle Tracking in CARLA Simulator

Jagadeesh Prasad Batchu
Department of Computer Science
University of Exeter
Exeter, UK
jb1473@exeter.ac.uk

Han Wu
Department of Computer Science
University of Exeter
Exeter, UK
hw630@exeter.ac.uk

Berrisford Liam
Department of Computer Science
University of Exeter
Exeter, UK
l.berrisford2@exeter.ac.uk

***Abstract*—Real-time vehicle tracking is a critical component in the development of autonomous vehicles because it facilitates a precise and reliable understanding of the surrounding traffic environment. In this project, a novel approach is presented to implement real-time 2D vehicle tracking in the CARLA simulator, utilizing the robust StrongSORT tracking algorithm in conjunction with the state-of-the-art YOLOv8 object detection model. The proposed methodology leverages the capability of training a model on YOLOv8 to accurately detect and classify vehicles in 2D within the CARLA Simulator, providing a rich stream of data for the subsequent tracking process. The StrongSORT algorithm then effectively handles the dynamic challenges of multiple object tracking, ensuring consistent and accurate vehicle trajectory estimation in real-time.**

The experimental results demonstrate the efficacy of the combined StrongSORT and YOLOv8 approach, showcasing its capability to deliver real-time vehicle tracking performance with high accuracy and efficiency in 2D. Moreover, the integration of this tracking system into the CARLA Simulator opens up new possibilities for simulating and validating the behavior of autonomous vehicles in complex and dynamic traffic scenarios.

Keywords - YOLOV8, StrongSORT, Multi-Object Tracking, CARLA

I. Introduction

The global autonomous vehicles market is currently experiencing rapid growth and innovation [1], driven by the convergence of automotive and artificial intelligence sectors, alongside advancements in technology. Established automakers, tech giants, and emerging startups are key players in this dynamic landscape, investing heavily in the research, development, and testing of self-driving cars equipped with cutting-edge sensors, AI algorithms, and connectivity features [2]. While challenges such as safety, regulation, and public acceptance persist, autonomous vehicles are gradually transitioning from controlled testing environments to real-world scenarios, with some vehicles already achieving partial automation levels [3]. Beyond personal transportation, the potential applications of autonomous vehicles extend to goods delivery, public transportation, and the industrial sectors, making it a focal point for global investment and innovation [4].

The rise of modern road networks has resulted in significant growth in the use of autonomous vehicles. Established car manufacturers are pushing the boundaries of self-driving technology to outperform their competitors and capture consumer interest. At the core of this technological evolution lies the crucial requirement for real-time object detection and precise vehicle tracking, serving as the foundation for autonomous driving [5]. As a result, the pursuit of dependable object detection and vehicle tracking systems has swiftly risen to prominence within the industry's agenda.

The focus of this project is to develop a real-time multi-vehicle tracking system that can accurately and efficiently monitor vehicles in the CARLA simulator. CARLA, an open-source driving simulator, is renowned for its contribution to shaping autonomous driving technologies and serves as the virtual environment for the pursuits. The core of the methodology focuses on achieving accurate vehicle tracking through the utilization of CARLA's RGB camera. This integration uses depth perception to estimate the spatial coordinates and orientation of the vehicle in relation to the road. To accomplish this, leverage the capabilities of the CARLA simulator, which provides highly realistic urban layouts, a diverse range of vehicle models, pedestrians, and a myriad of environmental conditions [6].

The use of an RGB camera in the CARLA simulator is a popular and widely accepted method for image recognition in computer vision. It provides crucial color information necessary for object detection and tracking [6]. In the CARLA simulator, it is possible to generate a real-time vehicle image stream using an RGB sensor camera. This feature offers easy installation and usage in experimental setups, making it well-suited for vehicle tracking within the CARLA simulator environment.

The approach encompasses two primary goals: identifying vehicles (2D) and tracking them subsequently. To achieve vehicle detection, utilizing a powerful deep learning-based object detection algorithm that specializes in accurately recognizing vehicles within stereo images. This involves leveraging advanced frameworks such as YOLOv8, known for its exceptional performance in accurately detecting vehicles in various scenarios. The use of YOLOv8 for vehicle detection in the CARLA simulator is beneficial due to its exceptional performance in accurately recognizing vehicles within Camera images.

A state-of-the-art multi-object tracking system was implemented for this project in the field of vehicle tracking. It accurately determines the coordinates and orientation of vehicles by utilizing detection information from the camera feed. The widely acclaimed StrongSORT tracking algorithm is used, known for its exceptional real-time performance in monitoring multiple objects. Its effectiveness has been demonstrated in

various scenarios, affirming its ability to ensure precise and dynamic tracking.

The versatility of this approach is demonstrated by its ability to track multiple vehicles simultaneously in the CARLA simulator. This makes it highly adaptable for real-world implementation in various domains, including traffic management, surveillance, and autonomous driving. By accurately and consistently tracking vehicles, the methodology contributes to improving safety, reducing congestion, and optimizing the efficiency of the transportation system. This has the potential to transform road safety, improve traffic flow, and increase overall efficiency in transportation networks.

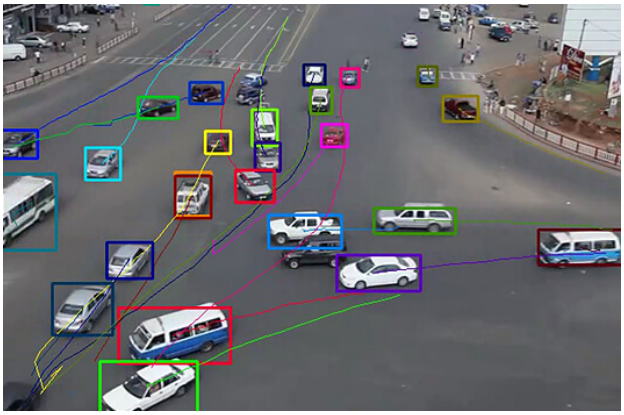


Fig. 1: From Aidetic Blog about Object Tracking in Videos: Introduction and Common Techniques [7], shows the Multi-Object Tracking of vehicles from a camera along with the tracks.

II. LITERATURE REVIEW

A. 2D Object detection

Accurate identification and localization of objects within a two-dimensional image or frame, known as 2D object detection, is critical in computer vision. This task is essential for autonomous driving, surveillance, robotics, and augmented reality applications. The objective of 2D object detection is to precisely determine the location and classification of each object in an image by creating bounding boxes around them [8]. This foundational information forms the basis for more advanced tasks such as tracking, scene comprehension, and decision-making.

Modern 2D object detection pipelines often leverage deep learning techniques, such as convolutional neural networks (CNNs), due to their exceptional ability to learn intricate patterns and features directly from raw image data [9]. These networks are typically trained on large labeled datasets, enabling them to recognize a wide array of objects and variations in real-world scenarios. One of the key challenges in 2D object detection is achieving a balance between precision and speed, as real-time applications demand rapid processing of frames without sacrificing detection accuracy. To address this, researchers have explored various architectures and optimization techniques, leading to advances in single-stage and two-stage detectors. [10]

Two-stage detectors, like Faster R-CNN [11], involve a multi-step process where regions of interest are first proposed and then refined to predict the final bounding boxes and class labels. In contrast, single-stage detectors, like YOLO (You Only Look Once) [12] and SSD (Single Shot MultiBox Detector) [13], predict object attributes directly from the image in a single pass, thus offering faster inference times. As the field evolves, efforts are being directed toward improving detection accuracy in challenging scenarios, such as occlusions, scale variations, and crowded environments. The ongoing research in 2D object detection continues to push the boundaries of what's possible in terms of real-time, accurate, and robust object detection, contributing to advancements across various domains that rely on visual understanding and interpretation [10].

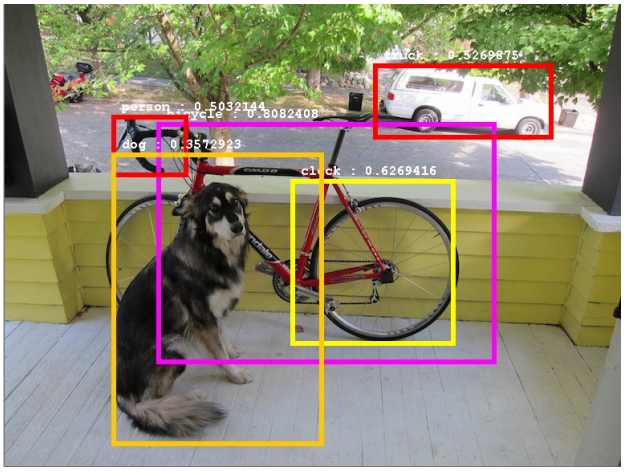


Fig. 2: The image from JetBrains blog [14] shows the 2D object detection with bounding boxes, classification names, and confidence scores. And the misrepresentation of the wheel with the clock can be seen.

B. Regular Camera

Regular cameras play a crucial role in object detection systems, capturing visual input that enables the identification and detection of objects within various environments. Incorporating regular cameras into object detection has significant practical implications in domains such as video surveillance, smart environments, and robotics. In this project, CARLA's RBG camera is used as the chosen regular camera to capture images and transmit the video stream to the models for object detection and tracking. Securely affixed to the vehicle within the CARLA environment, this regular camera allows us to acquire large amounts of visual data that can be processed in real-time for tasks like multi-object detection and tracking.

Regular cameras are commonly employed in 2D object detection to capture images, which are then processed using computer vision algorithms for identifying and localizing

objects within a scene [15]. These cameras find wide applications in both indoor and outdoor environments, as well as in humanoids, air drones, ground vehicles, and mobile devices. The use of regular cameras in object detection systems has practical implications across various areas such as video surveillance for monitoring suspicious activities or intrusions.

Although traditional cameras lack depth perception, they still hold valuable applications. They aid self-driving cars in perceiving their surroundings, enable surveillance systems to detect individuals or objects, and facilitate robots in navigating and engaging with their environment. However, the lack of depth information may pose challenges when determining the size and proximity of objects captured by these cameras. To overcome this limitation, intelligent algorithms can analyze the dimensions of objects in 2D images to make informed assumptions about distance.

C. Multi-Object Tracking

Multi-object tracking plays a crucial role in the field of autonomous vehicle tracking, providing the ability to accurately identify, categorize, and simultaneously monitor different entities such as pedestrians, other vehicles, and obstacles. This real-time capability enables the system to effectively interact with multiple objects, which is essential for safe and efficient navigation in autonomous vehicles. By anticipating the behaviours of these entities and generating appropriate responses, autonomous vehicles can operate harmoniously within their environment while ensuring safety and efficiency [16].

Various approaches exist for multi-object tracking. Classic techniques involve the use of Kalman filters and particle filters, which have proven to be reliable in estimating object trajectories. Modern advancements in deep learning, particularly with the utilization of convolutional neural networks, have also contributed to significant progress in object detection and tracking. In recent years, hybrid strategies that combine traditional methods with deep learning capabilities have emerged, exploiting the respective strengths of both domains [17]. These multi-object tracking algorithms are designed to handle the challenges presented in complex and dynamic environments, such as occlusions, appearance changes, and object interactions.

Ongoing research is devoted to improving the accuracy and effectiveness of multiobject tracking in autonomous vehicles, particularly in complex and dynamic settings such as busy urban environments [16]. This highlights the continuous quest to equip self-driving cars with the ability to navigate intricate scenarios smoothly while prioritizing road safety for all involved people. The accuracy of multi-object tracking in autonomous vehicles is essential for their safe and efficient operation.

In the domain of autonomous vehicles, multi-object tracking assumes great significance due to its pivotal role in comprehending the intricate interactions among various entities present on the road. As an autonomous vehicle navigates through complex traffic scenarios, it becomes crucial to accurately interpret the movements and trajectories of pedestrians, cyclists, and other drivers. This heightened awareness enables the autonomous system not only to comply with traffic regulations but also to forecast and effectively respond to unexpected behaviors. By leveraging the insights derived from multi-object tracking observations, autonomous vehicles transcend their mere transportation function by evolving into sophisticated collaborators that co-exist harmoniously with humans and other objects on the road, thereby facilitating a safer and more efficient ecosystem for shared mobility. In summary, multi-object tracking plays a crucial role in autonomous driving, as it allows the vehicle to understand the surrounding environment and make informed decisions [18].

Fig. 3: The Image from youtube by bibozidan [19] represents the Multi-object tracking on people and the generated distinct tracking IDs.

D. CARLA Simulator

The CARLA Simulator [20] is a widely recognized open source software that plays a vital role in the field of autonomous driving research. Developed collaboratively by the Computer Vision Center and Universitat Autonoma de Barcelona, this state-of-the-art creation serves as an invaluable tool for refining cutting-edge algorithms. With its realistic environment, aptly referred to as a "simulator," CARLA enables researchers to conduct comprehensive testing and validation within a highly accurate virtual setting. This highly acclaimed software has gained significant acclaim in both academic and industry spheres, making it instrumental in fostering innovation in autonomous vehicles. CARLA simulator is widely used in the field of autonomous driving research due to its flexibility and the ability to modify the source code to add new sensors.

The CARLA simulator offers a range of impressive features. It provides a realistic 3D environment with various sensors, such as LiDAR, radar, and cameras, allowing for extensive experimentation. Its adaptable nature is supported by an API that developers can customize to meet their specific research requirements. This flexibility extends to various aspects of autonomous driving, including perception, planning, and control tasks. The CARLA Simulator is highly attractive due to its extensive collection of meticulously crafted scenarios. These scenarios serve as effective testing environments for

a wide range of algorithms, covering diverse terrains such as busy city streets and expansive highways. Within this dynamic environment, algorithms undergo rigorous testing and continuously evolve to improve their ability to navigate through challenging conditions. The CARLA Simulator's significance in autonomous driving research stems from its lifelike virtual setting, versatile array of sensors, customizable API, and abundant selection of pre-existing scenarios.

Fig. 4: The image from UnrealEngine [21] represents the CARLA environment with roads, vehicles, traffic lights, and buildings

E. YOLOV8

YOLO (You Only Look Once) has emerged as a notable breakthrough in the field of object detection and image segmentation. Developed by Joseph Redmon and Ali Farhadi at the University of Washington in 2015, YOLO quickly gained recognition for its impressive combination of speed and accuracy. The latest version, YOLOv8, led by Ultralytics, builds on this success by incorporating new features and enhancements. Positioned as an advanced state-of-the-art model, YOLOv8 extends the achievements of its predecessors to deliver improved performance, adaptability, and efficiency. Its wide-ranging capabilities span various vision AI tasks including detection, segmentation, pose estimation, tracking, and classification which can be seen in Fig 5. These extensive functionalities empower users to leverage the powerhouse that is YOLOv8 across diverse applications and domains, reaffirming its pivotal role in contemporary computer vision and AI endeavors [22].

The Ultralytics YOLOv8 model represents the latest in real-time object detection and image segmentation. It builds on deep learning and computer vision progress, combining speed and accuracy. The YOLOv8 architecture is versatile, enabling it to be used across a range of applications and platforms, including edge devices and cloud APIs. It also provides a pre-trained model to perform object detection. The model can predict 80 different classes. [22].

F. StrongSORT

The StrongSORT algorithm, an enhanced version of the DeepSORT algorithm, which is a simple online real-time tracking algorithm, offers a simple but effective solution for real-time object tracking in video frames. This approach

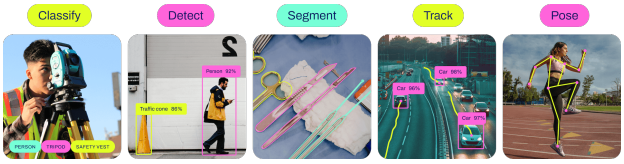


Fig. 5: The image from Ultrlytics [23] shows the different applications Yolo can perform such as Image classification, detection, segmentation, tracking, and pose estimation.

considers various aspects, such as object detection, feature embedding, and trajectory association, in order to achieve reliable tracking outcomes. With its focus on online and real-time applications, the StrongSORT algorithm efficiently associates objects between successive frames, overcoming one of the key challenges in object-tracking tasks. The StrongSORT algorithm builds on the existing DeepSORT algorithm, which has gained significant attention in the field of multiobject tracking [24]. The implementation of the StrongSORT algorithm takes into consideration the advances in object detection methods based on deep learning that have significantly improved the accuracy and efficiency of object detection.

These improvements are achieved through the integration of various techniques and advanced modules, which are detailed below.

Detection Model: Instead of using the faster R-CNN detector, StrongSORT employs YOLOX-X for object detection. YOLOX-X is a state-of-the-art object detection model that provides accurate and efficient detection results.

Appearance Feature Extractor: StrongSORT incorporates BoT (Bottleneck Transformers) as the appearance feature extractor. BoT is a powerful architecture capable of extracting highly discriminative features, enhancing tracking performance.

Exponential Moving Average (EMA): To mitigate the effects of detection noise, StrongSORT replaces the feature bank mechanism of DeepSORT with an Exponential Moving Average (EMA) strategy to update the appearance states. This involves updating the appearance embeddings in an EMA manner using the current matched detection appearance embedding and a momentum term (α = 0.9). EMA uses inter-frame feature changes to reduce detection noise and enhance matching quality.

Enhanced Correlation Coefficient (ECC): StrongSORT adopts the Enhanced Correlation Coefficient (ECC) model to compensate for camera motion. ECC is a technique for parametric image alignment that estimates the global rotation and translation between adjacent frames. This compensates for motion noise caused by camera movement during the tracking process.

NSA Kalman: To improve the Kalman filter's performance in the presence of low-quality detections and varying detection noise scales, StrongSORT incorporates the NSA (Noise Scaling Adaptation) Kalman algorithm. NSA Kalman adaptively adjusts the noise covariance based on the detection confidence score, improving the accuracy of state updates.

Motion Cost: StrongSORT extends the matching cost calculation in the first association stage by considering both appearance and motion information. The cost matrix is a weighted combination of the costs of appearance and movement, with the weight factor λ determining the balance between the two. This approach improves the association accuracy.

Vanilla Matching: StrongSORT replaces the complex matching cascade algorithm of DeepSORT with a simpler vanilla global linear assignment method. This change removes additional constraints, allowing the tracker to leverage its improved capabilities more effectively. Overall, StrongSORT takes advantage of these advances to provide superior tracking performance compared to DeepSORT, making it a robust and accurate choice for multiple object tracking tasks. The comparison can be seen in Fig. [24]

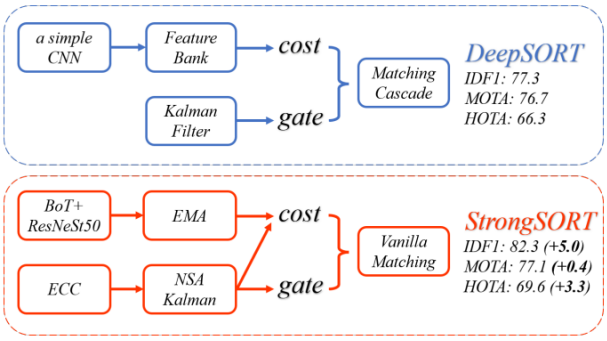


Fig. 6: The Diagram from StrongSORT Documentation [24] compares DeepSORT and StrongSORT algorithms and performance scores.

III. AIMS AND OBJECTIVES

A. AIM

The primary aim of this project is to develop a dynamic and real-time vehicle tracking system by synergizing the capabilities of YOLOv8, a state-of-the-art object detection model, and StrongSort, an advanced multi-object tracking algorithm. This system is designed to seamlessly integrate into the CARLA simulator, a high-fidelity environment for autonomous driving research.

B. OBJECTIVES

1) Train YOLOv8 and Detection: The first objective entails training the YOLOv8 model using a tailor-made dataset, allowing it to proficiently detect vehicles in various scenarios. This involves harnessing the power of deep learning to enable accurate and rapid vehicle detection within the simulated environment. And detect vehicles accurately.

2) Implement StrongSort: : The project's second objective revolves around the implementation of the StrongSort algorithm, a robust solution for multi-object tracking. This involves the development and integration of a system that can accurately follow and predict the trajectories of multiple vehicles simultaneously, contributing to an enhanced understanding of dynamic road scenarios.

3) CARLA Simulator Integration: : The final objective involves seamlessly integrating the YOLOv8-based vehicle detection and the StrongSort-based tracking system into the CARLA Simulator. This step ensures that real-time tracking capabilities are effectively embedded within the simulated environment, providing a holistic and interactive testing ground for autonomous vehicle technologies.

By achieving these objectives, the project aims to create a cutting-edge vehicle tracking solution that can effectively identify, monitor, and predict vehicle movements within the CARLA Simulator, thus contributing to the advancement of autonomous driving research and technology.

IV. DATASET AND RESOURCES

A. Image Dataset

The dataset used in this project was collected from the Roboflow website. It consists of around 3,800 images that have been carefully divided into training and validation sets. The collection process involved merging data from different versions of datasets to ensure completeness. This dataset is crucial for training and evaluating object detection models, as it plays a vital role in improving the accuracy and reliability of the implemented system. Using this extensive data set, the objective of this project is to develop an advanced object detection solution capable of accurately detecting and localizing various objects within a given visual field.

The dataset comprises various object categories, each assigned to distinct classes. These classes include "bike," "motorbike," "traffic lights," and the broad category of "vehicle" which encompasses a diverse range of automobiles such as cars, trucks, and buses. This comprehensive classification system ensures that the object detection model can accurately identify and localize a wide variety of relevant objects encountered in real-world situations.

The dataset utilizes the YOLO format, which is widely used in object detection tasks. This format includes crucial details for each object instance, such as the class name and bounding-box coordinates that define the spatial extent of the object within the image. By offering this organized and standardized labeling system, the dataset enhances training by enabling accurate alignment between model predictions and ground truth information. The incorporation of these annotations greatly contributes to a more efficient training process for deep learning algorithms in panoramic imagery.

B. CARLA Simulator

The object detection and tracking process in this project relies on the image stream extracted from the CARLA simulator. For this project, the latest version of CARLA simulator is used which is 0.9.14 The CARLA simulator provides various scenarios suited for experimental purposes, including traffic management related to climate change. To generate vehicles for experimentation, they were simulated within the environment of the CARLA simulator. A camera was attached to a selected vehicle, which was then placed on autopilot mode and guided through the simulated landscape. As a result of this

autonomous journey, the images were captured and resized before being fed into the detection model for further analysis.

V. Experimentation

To enhance object detection and tracking abilities, a thorough experimental phase was conducted involving the use of advanced techniques. This section provides an in-depth analysis of the implementation process and results obtained from custom object training using YOLOv8 for detection purposes, followed by employing the robust StrongSORT algorithm for object tracking.

Note: The CARLA Simulator requires a minimum of 6GB Graphical Processing Unit(GPU) to run.

A. Custom Object Training with YOLOv8

The first phase of the experimentation involved custom object training using the YOLOv8 architecture. This cutting-edge deep learning model has garnered significant attention for its remarkable object detection performance across diverse scenarios. The custom dataset, meticulously curated from the Roboflow resource, contained a diverse array of objects, including bikes, motorbikes, traffic lights, and vehicles encompassing various automobile types.

To conduct this study, a substantial dataset was gathered from the Roboflow website, comprising approximately 3800 images.

1) Data Preparation: As part of the data preparation phase, irrelevant categories are excluded such as traffic lights from the dataset. Additionally, merged the "bike" and "motorcycle" classes into one category. Moreover, broadened the definition of the "vehicle" class to encompass different types of vehicles including cars, trucks, and buses. Eliminated unnecessary categories from the dataset and divided the data into training and validation sets.

2) Selection of YOLO Architecture: For this experiment, YOLOv8 is selected, a variant of the YOLO architecture. The model was initialized with pre-trained weights on the image dataset to facilitate faster convergence.

3) Configuration of Model Size and Iterations: In this experiment, among various sizes of the yolov8 model: nano, small, medium, large, and extra-large, Smaller sizes were used to enhance detection speed while larger sizes were used to improve detection accuracy. In order to ensure computational convenience in this experimental setup, only the nano and small architectures were implemented.

The nano model used 2.21 gigabytes of GPU memory, and the Small model used 3.67 gigabytes of GPU memory for each iteration.

4) Training Loop (Iterations): Batch Predictions: The dataset was processed in smaller groups known as mini-batches, with each batch consisting of loaded and preprocessed images. The YOLO algorithm made predictions for bounding boxes, class names, and confidence scores based on the provided images.

Loss calculation: The model's weights were updated through backpropagation using gradient descent.

Backpropagation: The model's weights were updated through backpropagation using gradient descent.

Validation: Throughout the experimentation process, the model periodically assesses the performance of the model using a separate validation set. This enabled us to calculate evaluation metrics such as precision, recall, and mean Average Precision (mAP) in order to evaluate the accuracy and generalizability of the model.

5) Training Completion: The model was trained for an adequate number of epochs or until a plateau or convergence in the validation metrics was observed. Alternatively, training ceased if there was no alteration in parameters after a specific number of iterations. At the end of the training process, the current model and the best model achieved so far were saved.

6) Training Process Details: The model was subjected to 1000 iterations(epochs) for this particular training, employing YOLOvn and YOLOv8s models. The training process stopped at 406 epochs for the small model as indicated in the provided Fig 7. and the nano model stopped at 271 epochs. The size of the nano model is 6 megabytes, the small model is 22 megabytes.

B. CARLA Image Extraction

An image stream extraction from the CARLA simulator is a key step in the detection and tracking process, it involves a series of ordered actions:

1) Initiation of the simulation environment by launching CARLA.
2) Establishment of a connection to the CARLA simulator through utilization of the CARLA client.
3) Entry into the world environment embedded within CARLA.
4) Retrieval of blueprints and spawn points residing in CARLA's world.
5) Selection of a vehicle, designated as the primary element, along with an RGB camera, both sourced from the blueprints.
6) Configuration of camera attributes that include dimensions such as height, width, and field of view.
7) Attachment of the camera to the vehicle and subsequent adjustment of the camera's perspective to align with the desired view.

C. Yolo Detection

The subsequent stage of this experimentation pertains to object detection.

The image acquired from the CARLA simulator is divided into individual frames, which are then directed to the object detection model. In this particular scenario, employ the pre-trained Yolov8 model to successfully accomplish the task of object detection.

1) Initialization and Setup: Commencement involves configuring specific parameters. These encompass the path leading to the pre-trained model, numeric identifiers assigned to distinct object classes, and corresponding labels for these categories (examples being "bike," "vehicle," etc.).

Model	Size (pixels)	mAPval 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	Params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

TABLE I: The table from ultralytics [25] shows the different available Yolo models and their respective speeds. From this table it can be seen that the speed is decreasing when the size of the model is increased

```
Stopping training early as no improvement observed in last 50 epochs. Best results observed at epoch 356, best model saved as b
est.pt.
To update EarlyStopping(patience=50) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStoppin
g.

406 epochs completed in 11.168 hours.
Optimizer stripped from runs\detect\train11\weights\last.pt, 22.5MB
Optimizer stripped from runs\detect\train11\weights\best.pt, 22.5MB

Validating runs\detect\train11\weights\best.pt...
Ultralytics YOLOv8.0.152  Python-3.9.12 torch-1.13.1 CUDA:0 (NVIDIA GeForce RTX 3070 Laptop GPU, 8192MiB)
YOLOv8s summary (fused): 168 layers, 11126358 parameters, 0 gradients
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 5/5 [00:04<0
                   all        144        260      0.955      0.967      0.987      0.835
             motobike        144         61      0.949      0.967      0.985      0.797
              vehicle        144        199       0.96      0.966      0.989      0.874
Speed: 1.1ms preprocess, 5.6ms inference, 0.0ms loss, 1.7ms postprocess per image
Results saved to runs\detect\train11
```

Fig. 7: After initializing the model for 1000 epochs the Yolo small model stopped early at 406 epochs due to no improvement in the last 50 epochs. And shows the final performance metrics of precision, Recall, and mean Average Precision for individual classes of bikes and vehicles.

2) Model Loading: The subsequent step entails loading the pretrained model into the system's memory. This model has already acquired the capability to recognize various objects within images.

3) Frame Prediction: Each frame of the image, having been properly resized, is employed as input for the loaded model's prediction. The model then processes this frame and endeavors to discern the objects present within it.

4) Analysis and Output: Post-analysis, the model provides insights regarding the identified objects. This information encompasses various details, such as the precise positions of objects (bounding boxes), the category of each object (class identifiers), and the confidence level associated with the model's prediction.

Additionally, let's briefly delve into the details of Step 3, which is a crucial component of the process:

Preprocessing: The initial phase involves subjecting the preprocessed image to a Convolutional Neural Network (CNN) model to derive a feature map. A convolutional layer is subsequently applied to this feature map. This layer serves to predict attributes like objectness scores, bounding box coordinates, and class probabilities.

Anchor Boxes: A collection of anchor boxes, each with distinct shapes and sizes, is defined. These anchor boxes correspond to the potential dimensions of object bounding boxes. They play a pivotal role in predicting precise bounding box coordinates.

Bounding Box Prediction: For each anchor box, forecasts are made regarding offset values pertaining to the bounding box coordinates (minimum x, minimum y, maximum x, maximum y) in relation to the dimensions of the anchor box.

Objectness Score: The model estimates an objectness score (confidence score) for each anchor box. This score signifies the likelihood that an object is present within the box.

Class Prediction: Forecasts are generated for class probabilities associated with each anchor box. These probabilities indicate the likelihood that the detected object belongs to particular predefined classes.

Non-Maximum Suppression (NMS): Non-maximum suppression techniques are employed to eliminate superfluous and overlapping bounding box predictions. Bounding boxes with low objectness scores or meager class probabilities are discarded. Only the most assured bounding boxes are retained, ensuring that each detected object corresponds to a distinct bounding box.

Final Output: Ultimately, the process culminates in the presentation of the definitive array of detected object bounding boxes, accompanied by their respective class labels and confidence scores.

The steps for visualizing the results of the object detection process are as follows:

1) Retrieve Detection Outputs: Obtain the output from the object detection model, which includes information about detected objects such as bounding box coordinates, class IDs, and confidence scores.
2) Image Display: Display the original input image, which was used for object detection. This will serve as the base

for visualizing the detection results.

3) Bounding Box Overlay: Overlay the bounding boxes on the displayed image according to the coordinates provided by the object detection model. Each bounding box represents the location of a detected object.
4) Class Label Annotation: Annotate the bounding boxes with class labels corresponding to the detected objects. Use the class IDs provided by the model to map to human-readable class labels (e.g., "bike," "vehicle").
5) Confidence Score Indication: Indicate the confidence score associated with each detection by displaying it near or within the respective bounding boxes. This provides insight into the model's certainty about its predictions.
6) Color Coding: Apply color coding to the bounding boxes to differentiate between different classes of objects. Use consistent colors for the same class across different frames or images.
7) Display or Save: Choose whether to display the annotated image with detection results in a graphical user interface using Python opencv library or save it as an output file for later reference and analysis.
8) Iterate for Multiple Frames(Optional): If working with an image stream or video, repeat the above steps for each frame to visualize object detection results over time.

D. StrongSORT Integration

The code utilized in this project is obtained from the original strongSORT repository developed by Du, Yunhao et al. [24] Some modifications were made to ensure compatibility with the track file, and used the torchreid package to calculate the distances for the nearest neighbors matching. Developed the tracker file to integrate the CARLA image stream with the detection model and tracker to perform tracking.

StrongSORT incorporates various advanced modules, including YOLOX-X for detection, BoT for appearance feature extraction, EMA for noise reduction, ECC for camera motion compensation, NSA Kalman for adaptive noise covariance. Additionally, StrongSORT utilizes a motion cost calculation and replaces the matching cascade with a vanilla global linear assignment to enhance performance.

The steps involved in the StrongSORT algorithm are,

1) Forward the detected objects, class identifiers, and confidence scores from the initial detection stage along with the original image frame.
2) Update image frames (previous and current frames)
3) Execute the tracking process.
4) The system will generate tracked bounding boxes that include tracking IDs, tracking classes, and confidence scores.

The main step in the process, Step 3 of tracking execution, can be further represented in the following way.

1) Create Tracked Objects: Establish a collection of tracked objects by assigning them initial bounding boxes and distinctive identifiers. For each individual object, generate a Kalman filter to estimate its state (including position and velocity) as well as covariance.

2) Frame Processing:: For each frame received, the detected bounding boxes and confidence scores of objects are obtained. In order to eliminate duplicate or low-confidence detections, Non-Maximum Suppression is applied.

3) Association: For each identified object bounding box in the current frame: Calculate appearance characteristics using a convolutional neural network that transforms the object image patch into a feature vector. Measure the pairwise similarity between the appearance features of the identified object and the monitored objects. Determine Mahalanobis distance by comparing the forecasted Kalman filter state of each monitored object with an identified object's bounding box. Integrate appearance and motion data to calculate an affinity score reflecting the probability of association between an identified object and a monitored object.

4) Data Association: Formulate data association as a bipartite graph matching problem, where the nodes in separate sets represent detected objects and tracked objects. Compute the affinity matrix by utilizing the computed affinity scores. Resolve the linear assignment problem through an optimization algorithm such as the Hungarian algorithm. This will identify the optimal assignment that maximizes the total affinity score.

5) Update Tracked Objects: For each assigned pair of detected and tracked objects, the Kalman filter state is updated using information from the object's bounding box. This update refines both position and velocity estimates. The appearance features of the tracked object are then updated using a deep neural network. Subsequently, the tracking score is adjusted based on these newly updated appearance features.

6) Create New Tracks: For each detected object that is not assigned to a track, a new track is established using the bounding box and appearance features of the object. A unique identifier will be assigned to this new track, along with the initialization of a Kalman filter.

7) Predictions: Utilizing the Kalman filter to estimate the state of each tracked object, forecasting its position in the subsequent frame. Adjust the tracking score by considering motion prediction.

8) Track Management:: Eliminate tracks that have not been linked to detection for a specific number of frames, which are considered lost tracks. Keep and keep up-to-date records of recent object detections and tracked objects to handle temporary occlusions or missed detections.

VI. Experimental Results

A. Yolo Model Training Results

The results after training the Yolo model on a custom dataset are as follows.

Results from Table II give the bounding boxes, class, and Distribution Focal Loss values for the nano and small models respectively. From the values, both model has good scores but the small model has better accuracy which can be also seen in Table II which represents the precision, recall, and Mean Average Precision(mAP) scores for the Intersection of

Union(IoU) at (0-50) and (50-95) for the nano and small models.

models	box loss	class loss	dfl loss
nano	0.5996	0.3377	0.9165
small	0.4531	0.2588	0.8637

TABLE II: The bounding boxes, class, and Distribution Focal Loss values of yolo models. The small model has less loss values compared to the nano model in all three loss cases.

models	precision	recall	mAP50	mAP50-90
nano	0.962	0.924	0.983	0.797
small	0.954	0.967	0.987	0.836

TABLE III: The precision, recall, and Mean Average Precision(mAP) scores for the Intersection of Union(IoU) at (0-50) and (50-95) for the nano and small models. From the Table, the Yolo small model has a significant increase in scores.

The small model performed well compared to the nano model but the frame rate decreased by 40% while executing the tracking algorithm. The nano model is fast and performs with good accuracy. Both models are used for detection in this experiment. Based on the computational equipment and Graphical Processing unit increase the model size for better and more accurate results.

B. Yolo object detection Results

After passing the image stream from the CARLA simulator through the Yolo detection model, Fig. 9 represents an output image displaying detected vehicles. Each vehicle is represented by a bounding box accompanied by confidence scores and class names.

Fig. 8: Output of yolo detection in CARLA simulator with bounding boxes, class names, and confidence scores.

C. StrongSORT Results

After incorporating the detections and tracking, the image in Figure 10 displays the resulting frame from the generated video. This frame presents individual bounding boxes, distinct tracking IDs, class names, and confidence scores assigned to each vehicle present.

Fig. 9: Final Output of object tracking in CARLA simulator with bounding boxes, tracking ids, class names, and confidence scores.

D. Ground Truth

For generating ground truth the code is used from the CARLA simulator Documentation, and motmetrics package by Heindl [26] which uses Intersection over Union (IOU) and Squared Euclidean distance to calculate Multiple Object Tracking Accuracy and Multiple Object Tracking Precision, the package is used to evaluate the ground truth of the CARLA Simulator to the outputs of the implemented tracking algorithm. The results can be seen in Table IV.

mota	motp
0.623	0.348

TABLE IV: The table presents the recorded scores for Multiple Object Tracking Accuracy and Multiple Object Tracking Precision, indicating an accuracy of 62.3 percent and a precision of 34.8 percent. The underlying factors contributing to these relatively low evaluation scores are elaborated upon in the subsequent discussion section.

VII. Discussion

While running the ground truth, the CARLA simulator is extracting the bounding boxes of the hidden vehicles which are not in sight of the camera; due to this, the measurement of ground truth is difficult to determine. Also, the distance metric affects the ground truth results. For example, if the distance of 50 meters is set and the Yolo model could predict the image further than 50 meters it comes under false positive. If the distance is increased, then the bounding boxes of vehicles not in sight also increase. Due to this evaluating the ground truth is a difficult task in CARLA Simulator.

The following sections explore potential areas for future expansion of the project.

A. Advancing to 3D Object Tracking

The project's scope can be expanded to include the development of a more advanced tracking system that incorporates three-dimensional (3D) object tracking. This would involve integrating additional sensor modalities such as LiDAR, which would enable accurate tracking of objects both in image space and in-depth. By combining data from cameras and LiDAR sensors, the tracking system could achieve higher precision, particularly in complex scenarios with obstacles and varying distances between objects [27].

B. Incorporating the Latest Tracking Algorithms

Improving the tracking system's performance can be achieved by integrating and evaluating cutting-edge tracking algorithms. The use of deep learning methods or hybrid approaches may enhance both the accuracy and stability of the system. Additionally, incorporating algorithms that consider object interactions, predict object trajectories, and adapt to different situations can further enhance the capabilities of the tracking system.

C. Training Detection Models with Alternate Architectures

To broaden the scope of the project, it is recommended to include various architectural models along with YOLOv8 to train the detection model. Models such as Fast R-CNN, Faster R-CNN, or EfficientDet can offer valuable insights into their performance in object detection within specific contexts. By conducting a thorough comparison and analysis of the strengths and limitations of different architectures, well-informed decisions can be made about the selection of the most appropriate model for specific application scenarios [28].

D. Semantic Segmentation and Contextual Understanding

The use of semantic segmentation techniques has the potential to improve object tracking by providing a more holistic understanding of the surrounding environment. This approach allows the tracking system to distinguish between different categories of objects and understand their relationships. By incorporating contextual information, the tracking system can make more informed decisions and predictions about object movements and interactions [29].

E. Real-time Fusion of Multiple Sensors

Developing a comprehensive perception system that integrates data from various sensors, including cameras, LiDAR, radar, and inertial sensors in real-time can greatly enhance the accuracy of object detection and tracking. By combining input from multiple sources and addressing the limitations of individual sensors, this approach improves overall performance [30].

This project lays the foundation for future advances in object detection and tracking. As technology progresses, these developments have the potential to greatly enhance the capabilities of the project, allowing it to handle more complex scenarios with improved accuracy. This will broaden its applications across different industries and contribute to the advancement of intelligent and adaptable systems

VIII. CONCLUSION

In conclusion, this project marks a notable advancement in the field of object detection and tracking. It uses state-of-the-art technologies and methods to improve the perception and comprehension of dynamic environments. By employing custom object training with YOLOv8 for precise detection, as well as integrating StrongSORT for reliable object tracking, the project showcases the potential for developing intelligent systems capable of real-time object analysis.

The successful integration of these methods highlights their efficacy in various domains such as autonomous driving, surveillance, and robotics. The seamless combination of object detection and tracking processes has the potential to significantly transform machine interactions with the environment by improving navigation safety, optimizing decision-making capabilities, and improving general situational awareness.

Moreover, the examination of potential future directions highlights the project's adaptability and scope. By incorporating advances such as 3D object tracking, cutting-edge tracking algorithms, and comprehensive sensor fusion, the project can improve its capabilities to address even more complex and challenging scenarios. These upcoming enhancements have considerable potential in advancing the field of object detection and tracking, thereby making noteworthy contributions to ongoing progress in intelligent systems.

This project showcases the significant advancements made through the integration of advanced computer vision techniques and innovative algorithms. The findings from this project provide a strong basis for future research, development, and innovation in object detection and tracking. These advancements contribute to the advancement of safer, smarter, and more capable systems in various fields.

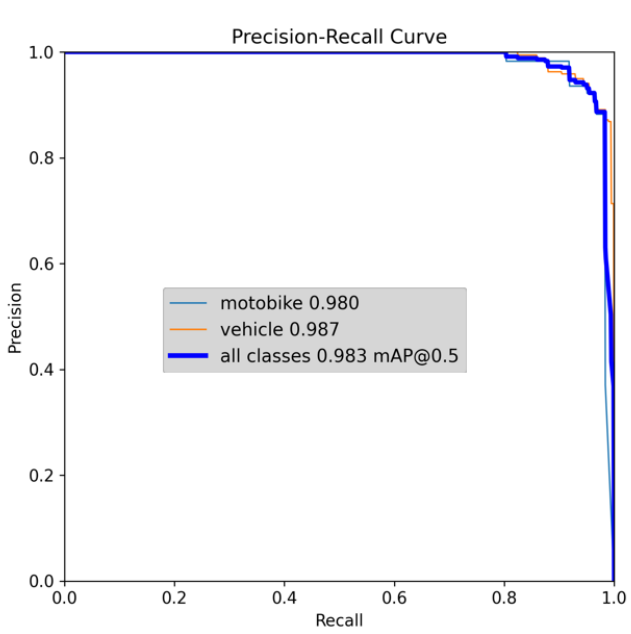
IX. DECLARATION

Declaration of Originality. I am aware of and understand the University of Exeter's policy on plagiarism and I certify that this assignment is my own work, except where indicated by reference, and that I have followed good academic practices.

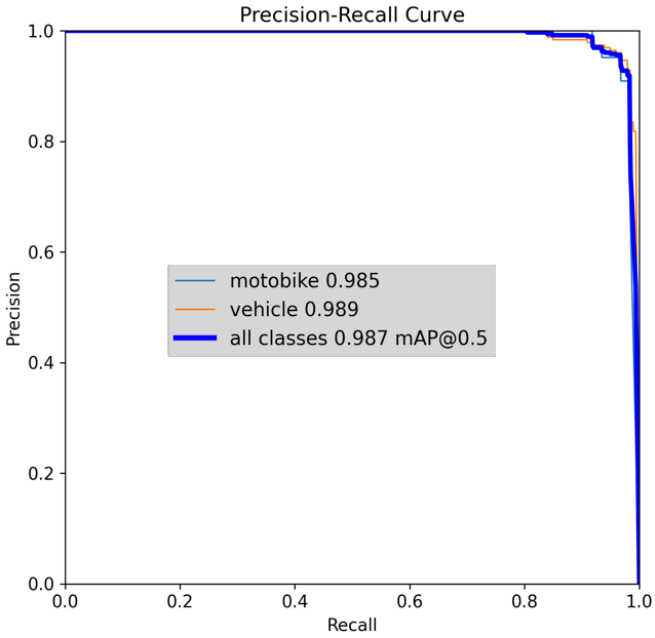
Declaration of Ethical Concerns. This work does not raise any ethical issues. No human or animal subjects are involved nor has the personal data of human subjects been processed. Additionally, no security or safety-critical activities have been carried out.

X. APPENDIX

In this appendix, the output graphs of the trained Yolo models of sizes small and nano are represented with respect to precision, recall, F1 scores, and Precision-recall curves.

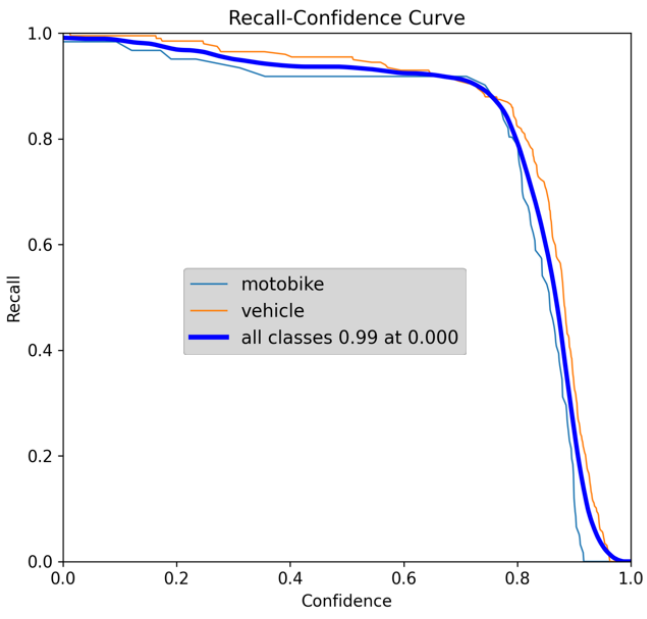


(a) nano model Precision-Recall Curve

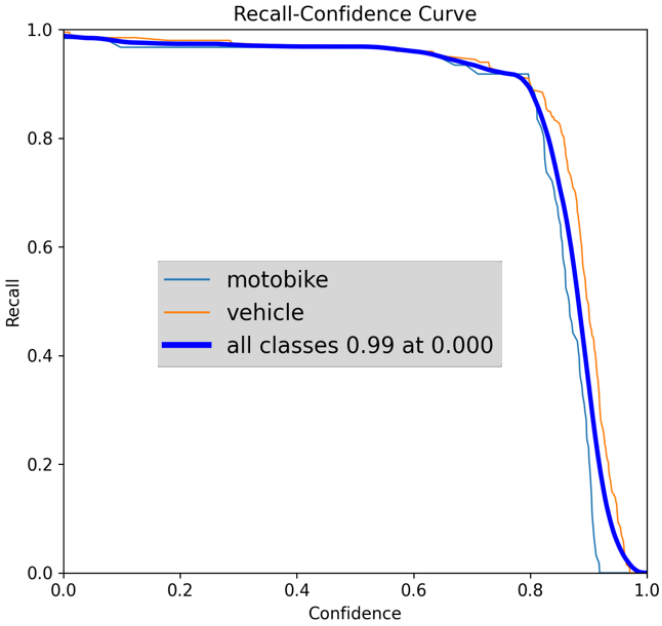


(b) small Precision-Recall Curve

Fig. 10: The PR Curve represents the trade-off between precision and recall. From the curves, the Yolo small model has a better mean Average Precision0-50 with an overall improvement of 0.003

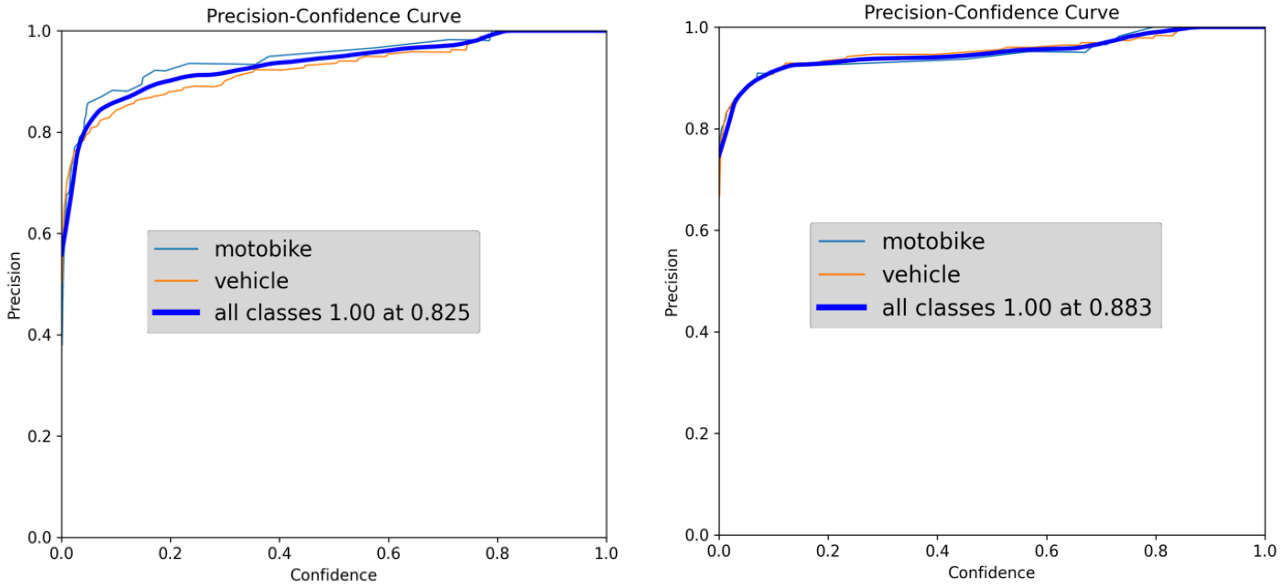


(a) nano model Recall-confidence curve



(b) small Recall-confidence curve

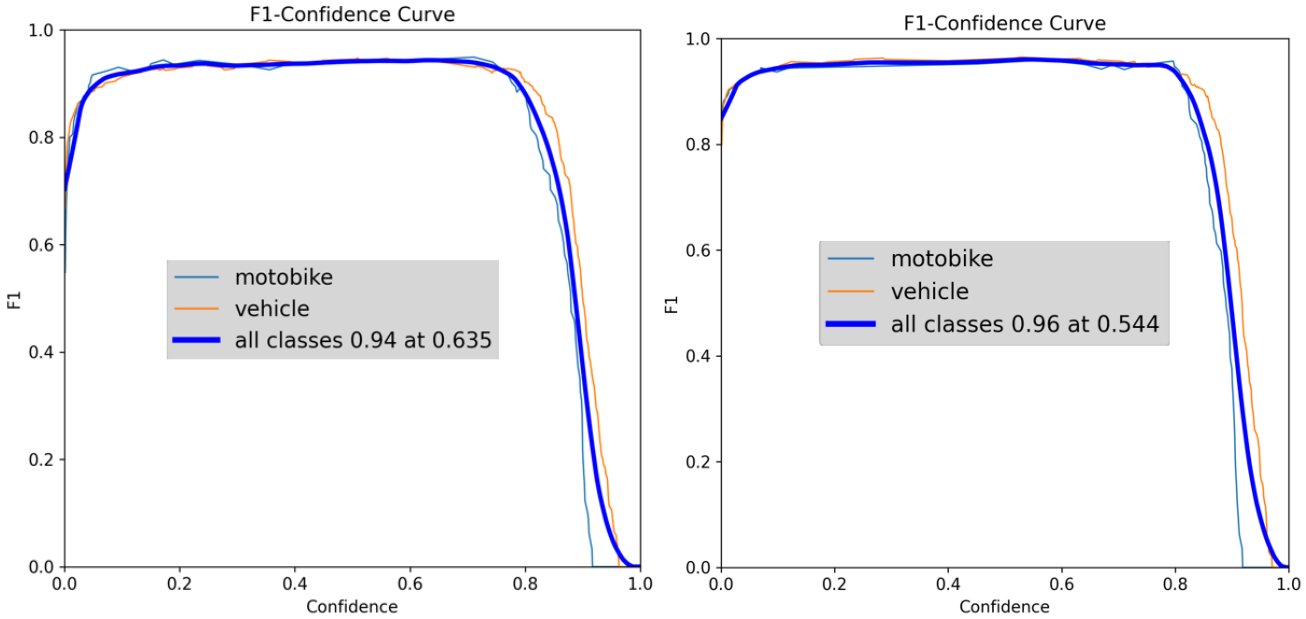
Fig. 11: The R Curve (Recall Confidence Curve) represents the variation in recall with respect to confidence thresholds. From the curves, both models have the same metrics.



(a) nano model Precision-Confidence curve

(b) small Precision-Confidence curve

Fig. 12: The P Curve (Precision Confidence Curve) is the precision of the model. The small model has a significant improvement in precision confidence by 0.58 to overall classes



(a) nano model F1 Confidence curve

(b) small F1 Confidence curve

Fig. 13: F1 Curve represents the combined precision and recall to give the model's classification accuracy. The small model has overall 0.2 more score than the nano model.

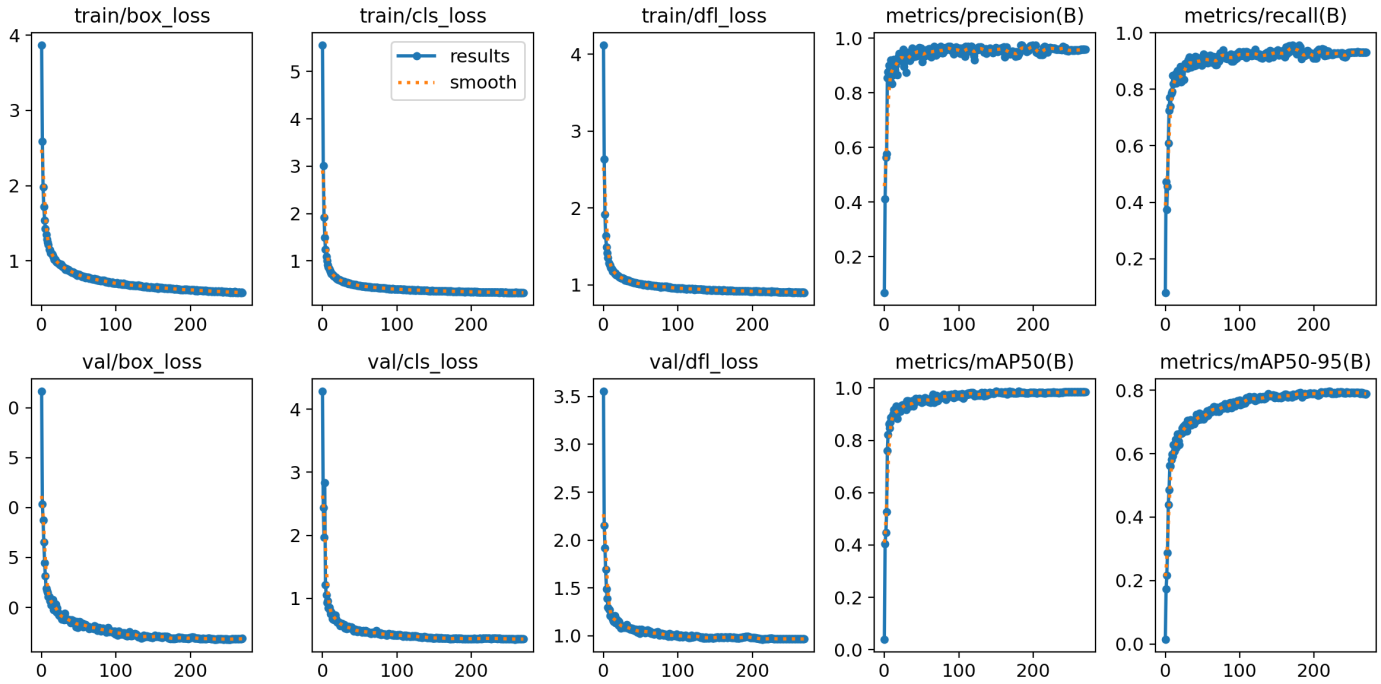


Fig. 14: The figure represents the curves of the Yolo nano model's training which includes the curves of the bounding box, class, and Deflection function losses for training and validation sets, and the metrics of precision, recall, and mean Average Precision. From the curves, it can be seen that at the end of iterations, the curves are becoming smooth which means the improvement is very less.

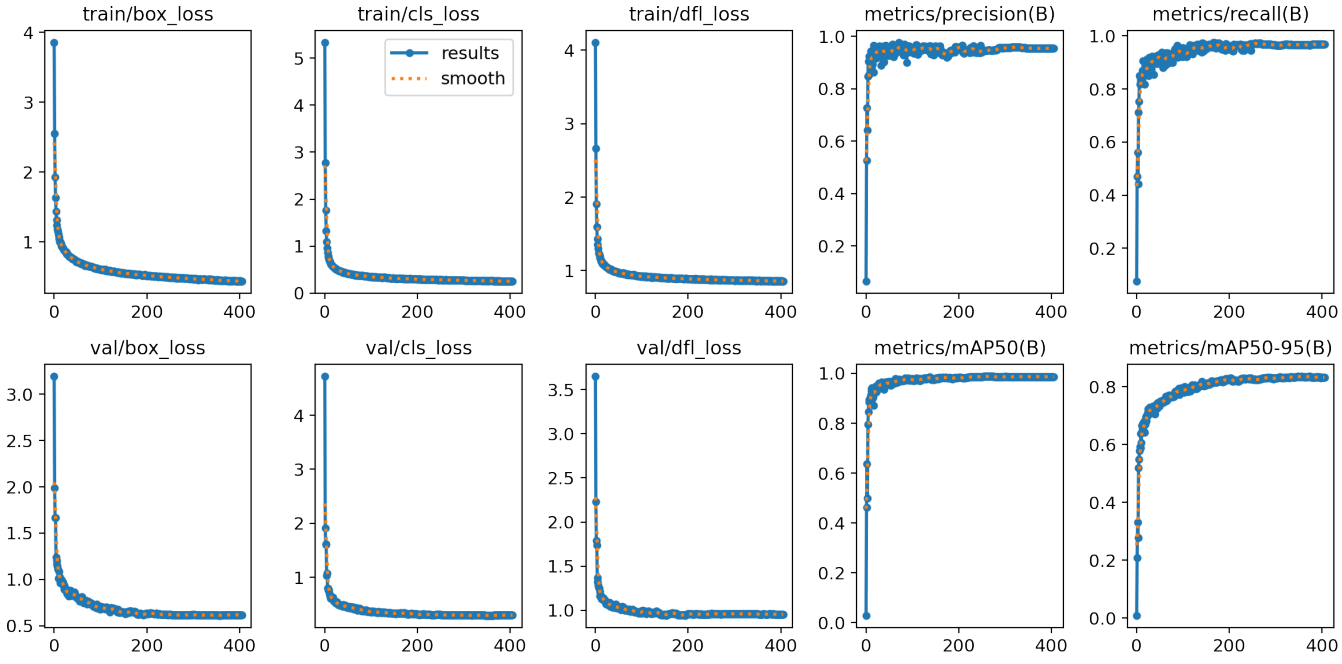


Fig. 15: The figure represents the curves of the Yolo small model's training which includes the curves of the bounding box, class, and Deflection function losses for training and validation sets, and the metrics of precision, recall, and mean Average Precision. From the curves, it can be seen that at the end of iterations, the curves are becoming smooth which means the improvement is very less. which is similar to yolo nano model.

REFERENCES

[1] L. D. Burns, "Sustainable mobility: A vision of our transport future," *Nature*, vol. 497, no. 7448, pp. 181–182, 2013.

[2] T. Litman, "Autonomous vehicle implementation predictions," Victoria Transport Policy Institute, Tech. Rep., 2017.

[3] S. Le Vine, A. Zolfaghari, and J. Polak, "Autonomous cars: The tension between occupant experience and intersection capacity," *Transportation Research Part C: Emerging Technologies*, vol. 52, pp. 1–14, 2015.

[4] J. Meyer, H. Becker, P. M. Bösch, and K. W. Axhausen, "Autonomous vehicles: The next jump in accessibilities?" *Research in Transportation Economics*, vol. 62, pp. 80–91, 2017.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[7] Mehul, "Object tracking in videos: Introduction and common techniques," Oct 2020. [Online]. Available: https://aidetic.in/blog/2020/10/05/object-tracking-in-videos-introduction-and-common-techniques/?ref=blog.roboflow.com

[8] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[10] W. Zhang, R. Li, and R. Nevatia, "Global context enhanced yolo: Recognizing objects in rgb-d indoor scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2016.

[14] JetBrains. (N.D.) Object detection with kotlindl and ktor. The Kotlin Blog. [Online]. Available: https://blog.jetbrains.com/kotlin/2022/01/object-detection-with-kotlindl-and-ktor/

[15] Y. J. Cho, W. Yu, J. H. Kim, and P. K. Rhee, "Real-time object detection and tracking on a moving vehicle in 2d images for automotive active safety systems," *International Journal of Automotive Technology*, vol. 14, no. 1, pp. 129–141, 2013.

[16] Y. Liu, Z. Liu, X. Wang, and X. Li, "Vehicle detection and tracking in autonomous driving: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 690–704, 2020.

[17] R. Chen, Z. Liao, H. Peng, H. Jiang, Y. Zhang, and Y. Liu, "Multi-object tracking using deep learning and visual data in autonomous vehicles," in *Proceedings of the 14th IEEE International Conference on Control and Automation (ICCA)*, 2018, pp. 764–769.

[18] J. Chen *et al.* (2021, October 12) Online multiple object tracking using a novel discriminative module for autonomous driving. [Online]. Available: https://scite.ai/reports/10.3390/electronics10202479

[19] YouTube. (N.D.) Multiple object tracking with dlib correlation tracker and sort. [Online]. Available: https://www.youtube.com/watch?v=tMuX5TP6uqA

[20] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78. PMLR, 2017, pp. 1–16.

[21] Unreal Engine. (N.D.) Carla democratizes autonomous vehicle rd with free open-source simulator. [Online]. Available: https://www.unrealengine.com/en-US/spotlights/carla-democratizes-autonomous-vehicle-r-d-with-free-open-source-simulator

[22] G. Jocher. (2020) Yolov8 documentation. [Online]. Available: https://docs.ultralytics.com/

[23] Ultralytics. (N.D.) Ultralytics yolov8 tasks. [Online]. Available: https://docs.ultralytics.com/tasks/

[24] Y. Du, S. Yang, B. Yang, and Y. Zhao. (n.d.) Strongsort: Make deepsort great again. [Online]. Available: http://arxiv.org/abs/2202.13514

[25] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[26] C. Heindl. (2023) py-motmetrics. [Online]. Available: https://github.com/cheind/py-motmetrics

[27] A. K. Patil, A. Balasubramanyam, J. Y. Ryu, B. Chakravarthi, and Y. Chai, "An open-source platform for human pose estimation and tracking using a heterogeneous multi-sensor system," *Sensors*, vol. 21, no. 7, p. 2340, 3 2021. [Online]. Available: https://scite.ai/reports/10.3390/s21072340

[28] M. Ghourabi, F. Mourad-Chehade, and A. Chkeir, "Eye recognition by yolo for inner canthus temperature detection in the elderly using a transfer learning approach," *Sensors*, vol. 23, no. 4, p. 1851, 2 2023. [Online]. Available: https://scite.ai/reports/10.3390/s23041851

[29] C. R. Qi, Y. Wu, H. Su, and L. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] D. Cho, "Autonomous driving assistance with dynamic objects using traffic surveillance cameras," *Applied Sciences*, vol. 12, no. 12, p. 6247, 6 2022. [Online]. Available: https://scite.ai/reports/10.3390/app12126247