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*—The implementation of real-time vehicle tracking is a critical component in the development of autonomous vehicles, as it facilitates a precise and reliable understanding of the surrounding traffic environment. In this thesis, we present a novel approach to 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 capabilities of 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 results of our experiments demonstrate the efficacy of the combined StrongSORT and YOLOv8 approach, showcasing its ability 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.

As autonomous vehicles continue to expand in prominence and reshape the transportation landscape, the significance of robust real-time vehicle tracking cannot be overstated. The successful implementation of our tracking system not only advances the state-of-the-art in autonomous driving research but also contributes to the safe and reliable deployment of self-driving vehicles on real-world roads.

In conclusion, this thesis makes a substantial contribution to the field of autonomous driving, providing a comprehensive solution for real-time vehicle tracking within the CARLA Simulator. The combination of StrongSORT and YOLOv8 proves to be a potent toolkit for monitoring and understanding traffic dynamics, supporting the development and testing of autonomous vehicles, and ultimately, paving the way for their widespread adoption in the near future.

keywords - YOLOV8, StrongSORT, MOT(Multi Object Tracking)

I. Introduction

The global autonomous vehicles market is currently experiencing rapid growth and innovation, 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. 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. 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.

Geographically, the autonomous vehicle market spans regions such as North America, Europe, and Asia, each with its own regulatory framework and technological progress. Leading companies are in a race to achieve higher levels of automation, with the goal of providing safer and more efficient mobility solutions. As technology continues to evolve and regulatory hurdles are addressed, the autonomous vehicle market is on track to reshape transportation and mobility, ushering in an era of smart, self-driving vehicles that could revolutionize industries and improve the overall quality of life.

The proliferation of modern road networks has led to a notable increase in the adoption 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. As a result, the pursuit of dependable object detection and vehicle tracking systems has swiftly risen to prominence within the industry's agenda.

The real-time monitoring of vehicles has taken a central role in ensuring the secure and efficient operation of self-driving cars. The identification and continuous monitoring of vehicles have gained significant traction as a focal point of exploration within the fields of computer vision and machine learning, as emphasized by research in the realm of autonomous Internet of Things (IoT) systems.

Our endeavor revolves around the development of a dynamic real-time vehicle tracking system, specifically designed to meticulously and efficiently monitor vehicles within 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 our pursuits. The foundation of our approach lies in achieving pinpoint precision in tracking through the ingenious fusion of stereo cameras. This integration harnesses the power of depth perception, enabling the estimation of the vehicle's spatial coordinates and orientation with respect to the road.

Stereo cameras have emerged as a preferred and widely adopted solution for depth estimation in computer vision. Their functionality is based on their ability to capture dual images from slightly distinct vantage points. By analyzing the disparity between corresponding pixels across these images, depth estimation is derived reliably. This depth of insight forms the cornerstone for the precise monitoring of objects' spatial presence and orientation across both two-dimensional and three-dimensional spaces. Stereo cameras play a pivotal role in tracking vehicles, as highlighted in Szeliski's work "Computer Vision: Algorithms and Applications." [1]

Our strategy encompasses two main objectives: vehicle detection(2D) and subsequent tracking. For vehicle detection, employing a robust deep learning-driven object detection algorithm focused on identifying vehicles within stereo images with precision. This involves utilizing cutting-edge object detection architectures, particularly frameworks like YOLOv8, known for their remarkable performance in discerning vehicles across diverse scenarios.

In the realm of vehicle tracking, we have developed an innovative multi-object tracking system. This framework extrapolates precise spatial coordinates and orientation of vehicles, leveraging detection insights from the camera feed. We adopt the renowned StrongSORT tracking algorithm, recognized for its exceptional performance in real-time monitoring of multiple objects. Its efficacy in ensuring accurate and dynamic tracking is evidenced by its success in various scenarios.

The versatility of this approach shines through its capability to track multiple vehicles concurrently in CARLA Simulator, making it highly adaptable for real-world implementation across domains such as traffic management, surveillance, and autonomous driving. By accurately discerning and consistently tracking vehicles, our methodology acts as a catalyst to advance safety, reduce congestion, and optimize the efficiency of the transportation system. This potential transformation paves the way for improved road safety, streamlined traffic flow, and increased efficiency in transportation networks.

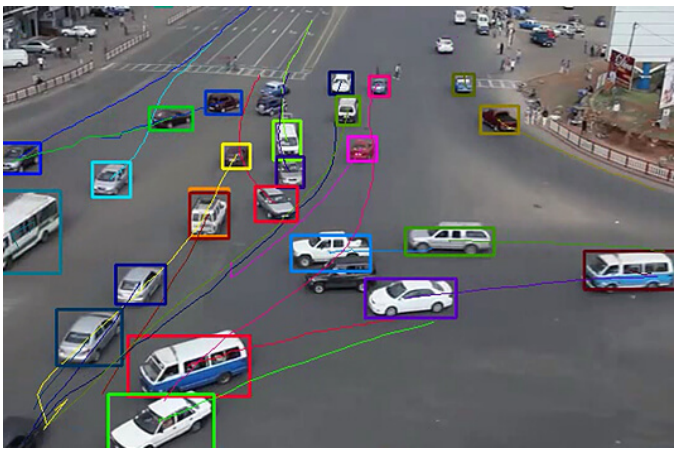

Fig. 1. Multi Object Tracking from a Traffic camera

II. Related Work or Literature Review

A. 2D Object detection

Two-dimensional (2D) object detection is a crucial task in computer vision that involves identifying and localizing objects within a two-dimensional image or frame. This process plays a pivotal role in various applications, including autonomous driving, surveillance, robotics, and augmented reality. The goal of 2D object detection is to accurately pinpoint the location and class label of each object present in an image, represented as a bounding box around the object. This information serves as a fundamental building block for higher-level tasks such as tracking, scene understanding, 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. 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. [2]

Two-stage detectors, like Faster R-CNN [3], 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) [4] and SSD (Single Shot MultiBox Detector) [5], 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 [2].

B. Stereo Camera

Stereo cameras are a pivotal technology in the realm of object detection, offering a perspective that significantly enhances the accuracy and reliability of identifying and localizing objects within a scene. These cameras consist of two lenses positioned slightly apart, capturing images from slightly different viewpoints. This disparity allows specialized algorithms to calculate depth information, generating a depth map that represents the distance of objects from the camera. This depth of insight is invaluable in various applications, especially where understanding spatial relationships is crucial.

One of the primary advantages of stereo cameras in object detection is their ability to provide accurate depth estimation, enabling precise distance calculations between the camera and

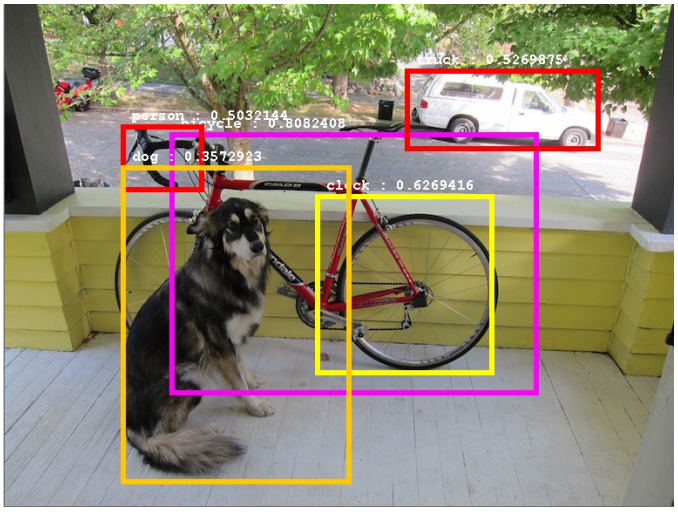

Fig. 2.

objects in the scene. This accuracy translates into improved obstacle detection, where the technology aids in identifying potential collision risks by distinguishing between objects close to the camera and those further away. Moreover, the depth information contributes to generating more accurate and well-fitted bounding boxes around detected objects. This precision is particularly significant in scenarios with occlusion challenges, ensuring that obscured objects are correctly identified.

Stereo cameras also play a key role in real-time monitoring and tracking of objects. The continuous stream of depth-enhanced images facilitates dynamic tracking of moving objects, making them well-suited for surveillance and traffic management applications. Additionally, stereo cameras provide a three-dimensional understanding of the environment, aiding in the creation of detailed maps that encompass not only horizontal but also vertical dimensions. This feature is especially beneficial in applications like autonomous navigation, where vehicles need to comprehend the full spatial layout of their surroundings to navigate safely and effectively.

In essence, stereo cameras revolutionize object detection by introducing depth perception, enabling more accurate and reliable identification, localization, and tracking of objects. Their ability to provide a three-dimensional understanding of the environment opens doors to various applications, from autonomous driving and robotics to augmented reality and environmental mapping, ultimately reshaping how machines interact with and interpret the world around them.

C. Multi-Object Tracking

Multi-object tracking serves as a pivotal pillar in the landscape of vehicle tracking for autonomous vehicles, furnishing the capability to adeptly recognize, classify, and concurrently monitor various entities like pedestrians, other vehicles, and obstacles. In real-time, this capability empowers the system to holistically engage with multiple objects, a quintessential ingredient for secure and efficient autonomous navigation. By foreseeing the actions of these entities and formulating appropriate responses, the autonomous vehicle operates in harmony with its environment, ensuring safety and effectiveness [6].

A range of methodologies is available for multi-object tracking. Traditional techniques encompass Kalman filters and particle filters, trusted tools for estimating object trajectories. In recent times, deep learning has unfurled novel avenues, with convolutional neural networks (CNNs) propelling strides in object detection and tracking. In a symbiotic synergy, hybrid strategies have emerged, amalgamating the strengths of traditional methods with the prowess of deep learning [7].

Emerging research is dedicated to refining the precision and efficiency of multi-object tracking within autonomous vehicles, especially in intricate, dynamic settings like bustling urban landscapes [6]. This underscores the ongoing endeavor to equip autonomous vehicles with the acumen to seamlessly navigate intricate scenarios while ensuring the safety of all stakeholders on the road.

In the realm of autonomous vehicles, the significance of multi-object tracking is magnified by its pivotal role in unraveling the complex interplay of diverse entities populating the road. As the vehicle maneuvers through intricate traffic scenarios, the ability to seamlessly decipher the intentions and trajectories of pedestrians, cyclists, and fellow motorists becomes indispensable. This dynamic awareness empowers the autonomous system to not only adhere to traffic rules but also to anticipate and respond effectively to unexpected behaviors. By harnessing the insights gleaned from multi-object tracking, autonomous vehicles transcend mere transportation, evolving into sophisticated partners that harmoniously navigate alongside humans and other objects, forging a safer and more efficient road-sharing ecosystem.

Fig. 3.

D. CARLA Simulator

The Carla Simulator [8] stands as an open-source software marvel, hailing from the collaborative efforts of the Computer Vision Center (CVC) and the Universitat Autonoma de Barcelona (UAB). This robust creation serves as a potent ally in the realm of autonomous driving research, providing a dynamic arena for honing and validating cutting-edge algorithms. Aptly dubbed a ”simulator,” Carla offers a realm of

realism, enabling comprehensive testing and refinement within a high-fidelity virtual environment. Renowned across academic circles and industry domains alike, Carla Simulator's prowess facilitates the unfurling of autonomous vehicle innovation[5].

At its core, Carla Simulator boasts an array of compelling attributes. From a lifelike 3D backdrop to a suite of diverse sensors including LiDAR, radar, and cameras, the simulator offers a playground teeming with possibilities. The heartbeat of its adaptability lies in its flexible and customizable API, a gateway through which developers can shape and mold the virtual world to fit their experimental needs. This extends to encompass various facets of autonomous driving, spanning perception, planning, and control tasks. Amplifying its allure, Carla Simulator comes pre-equipped with a repository of meticulously designed scenarios. These serve as testing grounds for an eclectic array of algorithms, spanning terrains from bustling urban streets to sweeping highways. In this dynamic crucible, algorithms are put to the test, evolving and enhancing their capabilities to navigate a spectrum of challenging conditions.

Fig. 4.

E. YOLOV8

YOLO (You Only Look Once) stands as a celebrated breakthrough in the realm of object detection and image segmentation. Conceived by Joseph Redmon and Ali Farhadi at the University of Washington and introduced in 2015, YOLO swiftly gained acclaim for its exceptional blend of speed and accuracy. YOLOv8, the latest iteration spearheaded by Ultralytics, amplifies this legacy by introducing new features and refinements. Operating as a cutting-edge state-of-the-art model, YOLOv8 builds upon the triumphs of its predecessors to deliver heightened performance, adaptability, and efficiency. Its versatile prowess spans a spectrum of vision AI tasks, encompassing detection, segmentation, pose estimation, tracking, and classification. This expansive functionality empowers users to harness the might of YOLOv8 across diverse applications and domains, underscoring its role as a pivotal force in modern computer vision and AI pursuits [9].

Ultralytics YOLOv8 takes the mantle as the latest embodiment of the acclaimed real-time object detection and image segmentation paradigm. Its bedrock is rooted in the vanguard of deep learning and computer vision advancements, culminating in a formidable fusion of swiftness and precision. With its lean and adaptable architecture, YOLOv8 seamlessly spans a gamut of applications and platforms, traversing from edge devices to cloud APIs.

The realm of vehicle tracking represents a formidable challenge within the purview of computer vision and autonomous driving. This challenge's magnitude underscores its paramount importance in advancing the frontiers of technology. A myriad of approaches have been conceived over time to address the complexities of tracking, with 2D and 3D techniques emerging as the cornerstone. While the surge of deep learning-powered object detection algorithms has showcased remarkable strides in object identification, the task of accurately tracking vehicles in 3D space remains an intricate puzzle. This complexity is further compounded by real-time environmental dynamics and the inherent limitations of cameras, including restricted perspectives and potential obstructions. In this context, an exploration of the literature unfolds, shedding light on diverse vehicle tracking methodologies, and the burgeoning prospects of stereo-camera-based tracking [9].

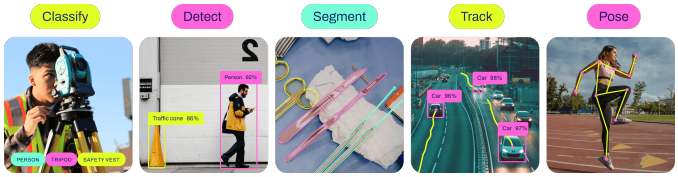

Fig. 5.

Model	size (pixels)	mAP^{val} 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

Fig. 6.

F. StrongSORT

StrongSORT is an advanced multiple object tracking algorithm that builds upon the foundation of DeepSORT while introducing several enhancements to improve tracking accuracy and robustness. These improvements are achieved through the integration of various techniques and modules, which are detailed below.

Advanced Modules: 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 for updating appearance states. This involves updating appearance embeddings in an EMA manner using the current matched detection's appearance embedding and a momentum term (= 0.9). EMA leverages 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 global rotation and translation between adjacent frames. This compensates for motion noise caused by camera movement in 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.

Two-Branch AFLink Model: StrongSORT employs a two-branch AFLink model to predict association scores between tracklets. The model integrates information from both temporal and feature dimensions using convolutional layers. This enhanced association prediction contributes to more accurate and reliable tracking.

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 appearance and motion costs, with the weight factor 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 leverages these advancements to provide superior tracking performance compared to DeepSORT, making it a robust and accurate choice for multiple object tracking tasks.

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 be seamlessly integrated into the Carla Simulator, a high-fidelity environment for autonomous driving research.

B. Objectives

Train YOLOv8: 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.

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.

Carla Simulator Integration: The final objective involves seamlessly integrating the YOLOv8-based vehicle detection and StrongSort-based tracking system into the Carla Simulator. This step ensures that the 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 the movements of vehicles within the Carla Simulator, thus contributing to the advancement of autonomous driving research and technology.

IV. DATASET AND RESOURCES

A. Image Dataset

The dataset employed in this project was sourced from the Roboflow website, comprising approximately 1700 images meticulously divided between training and validation sets. The dataset serves as the cornerstone for training and evaluating object detection models, playing a pivotal role in enhancing the accuracy and robustness of the deployed system. Leveraging this rich dataset, the project seeks to develop a cutting-edge object detection solution capable of accurately identifying and localizing various objects within the visual field.

The dataset is categorized into distinct classes, each catering to specific object categories. These classes encompass "bike," "motorbike," "traffic lights," and the expansive category of "vehicle," which encompasses a diverse array of automobiles such as cars, trucks, and buses. This meticulous classification of objects ensures that the object detection model is capable of discerning and precisely localizing a wide spectrum of relevant entities encountered in real-world scenarios.

The dataset's annotations follow the YOLO (You Only Look Once) format, a widely adopted approach for object detection tasks. This format includes essential information for each object instance, such as the class name and the bounding box coordinates that delineate the object's spatial extent within the image. By providing this structured and standardized labeling, the dataset empowers the training process by facilitating

the alignment of the model's predictions with ground truth information.

B. CARLA Simulator

The image stream utilized for object detection and tracking originates from the CARLA simulator, which offers a range of scenarios such as climate-change traffic management for experimentation. In this project, vehicles were generated within the simulator's environment. A camera was affixed to a chosen vehicle, which was then set in autopilot mode and navigated through the simulated landscape. This autonomous journey resulted in the creation of images, which were subsequently resized and forwarded to the detection model for analysis.

V. Experimentation or Implementation

In the pursuit of advancing object detection and tracking capabilities, a comprehensive experimentation phase was undertaken, involving the utilization of state-of-the-art techniques. This section delves into the implementation details and outcomes of custom object training using YOLOv8 for detection and the subsequent object tracking process utilizing the powerful StrongSORT algorithm.

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.

The YOLOv8 model was trained on this dataset, and the training process involved optimizing the model's weights to identify and localize objects within images accurately. By leveraging the YOLO (You Only Look Once) version 8 approach, the model exhibited real-time object detection capabilities, making it suitable for applications requiring swift and precise object identification.

Object Tracking Using StrongSORT: The second phase of the experimentation focused on object tracking, a crucial aspect of dynamic scene analysis. The StrongSORT (Strong Simple Online and Realtime Tracking) algorithm was employed for this purpose. StrongSORT is renowned for its exceptional real-time object-tracking performance, making it an optimal choice for seamlessly monitoring multiple objects within complex and dynamic environments.

By integrating StrongSORT with the object detection outputs from the YOLOv8 model, the experimentation aimed to achieve continuous and accurate tracking of detected objects across consecutive frames. The algorithm's ability to handle occlusions, track objects through cluttered scenes, and maintain tracking consistency contributed to the robustness of the overall tracking system.

Each process is explained in the following sections

A. Yolo model Training

The first step of the experimentation is the training of the model.

For this, a relatively large amount of data was collected from the roboflow website, consisting of around 1700 images as mentioned in the earlier section on data sources.

Data Preparation:

During data preparation, unnecessary classes such as traffic lights were removed from the dataset. Additionally, the "bike" and "motorcycle" classes were merged together. The "vehicle" class was expanded to encompass all types of vehicles, including cars, trucks, and buses.

The images and labels mentioned above were eliminated from the dataset, and the data was divided into training and validation sets. The training data was employed to train the YOLOv8 model.

Selection of YOLO Architecture:

A YOLO architecture variant, namely YOLOv8, was chosen for this experiment.

Model Initialization:

To expedite convergence, the model was initialized with pretrained weights on the image dataset.

Configuration of Model Size and Iterations:

For this experiment, model sizes were designated as nano, small, medium, large, and extra-large. Detection speed increases with smaller sizes, whereas larger sizes enhance detection accuracy. For the sake of computational convenience in this experiment, the nano and v8s architectures were utilized.

Training Loop (Iterations):

The dataset was iterated through in mini-batches:

Batches of images were loaded and preprocessed. The YOLO model was employed to derive predictions for bounding boxes, class probabilities, and objectness scores from the images. Loss calculation entailed a comparison between the predictions and the ground truth annotations. Backpropagation was performed to update the model's weights through gradient descent. Validation:

At periodic intervals, the model's performance was assessed using a distinct validation set. This allowed for the computation of evaluation metrics such as precision, recall, and mean Average Precision (mAP) to evaluate the model's accuracy and generalization.

Training Completion:

The model underwent training for an adequate number of epochs or until a plateau or convergence in validation metrics was observed. Alternatively, training ceased if there was no alteration in parameters after a specific number of iterations. At the conclusion of the training process, the current model and the best model achieved thus far were saved.

Training Process Details:

For this particular training, the model was subjected to 1000 iterations, employing YOLOvN and YOLOv8s models. The training process concluded around 200 epochs, as indicated in the provided figure 8.

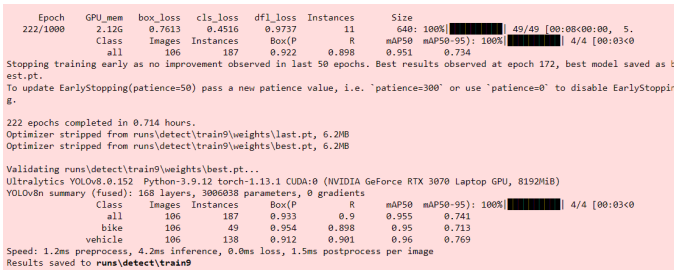

Fig. 7.

B. Carala Image Extraction

An image stream extraction from the Carla simulator involves a series of ordered actions:

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

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, we employ the pre-trained yolov8 model to successfully accomplish the task of object detection.

Step 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.). Step 2: Model Loading

The subsequent step entails loading the pre-trained model into the system's memory. This model has already acquired the capability to recognize various objects within images. Step 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. Step 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 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 of an object being present within the box. Class Prediction:

Forecasts are generated for class probabilities associated with each anchor box. These probabilities indicate the likelihood of the detected object belonging 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:

Step 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.

Step 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.

Step 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.

Step 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").

Step 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. Step 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.

Step 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.

Step 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.

——place object detection image

D. StrongSORT

After object detection, the next step is tracking using the StrongSORT algorithm.

It is the improved version of DeepSORT

step 1: send the detections, class ids, and confidence scores after the detection step. along with the original image frame.

step 2. Update image frames (previous and current frame)

Step 3: perform tracking

Step 4: output tracking bounding boxes with tracking ids, tracking classes, and confidence scores

step 3 of tracking can be done in the following way.

Initialize Tracked Objects: Initialize a set of tracked objects with their initial bounding boxes and unique identifiers. For each object, create a Kalman filter to estimate its state (position and velocity) and covariance.

Frame Processing: For each incoming frame: Receive the detected object bounding boxes and confidence scores. Perform Non-Maximum Suppression (NMS) to remove duplicate or low-confidence detections.

Association: For each detected object bounding box in the current frame: Compute appearance features using a deep neural network that maps the object image patch into a feature vector. Compute the pairwise similarity between the appearance features of the detected object and the tracked objects. Compute the Mahalanobis distance between the predicted Kalman filter state of each tracked object and the detected object's bounding box. Combine appearance and motion information to compute an affinity score that represents the likelihood of association between a detected object and a tracked object.

Data Association: Formulate data association as a bipartite graph matching problem, where detected objects and tracked objects are nodes in separate sets. Compute the affinity matrix using the computed affinity scores. Solve the linear assignment problem using an optimization algorithm (Hungarian algorithm) to find the best assignment that maximizes the total affinity score.

Update Tracked Objects: For each assigned pair of detected and tracked objects: Update the Kalman filter state using the detected object's bounding box, refining the position and velocity estimates. Update the appearance features of the tracked object using the deep neural network. Update the tracking score based on the updated appearance features.

Create New Tracks: For each unassigned detected object: Create a new track with the detected object's bounding box and appearance features. Initialize a Kalman filter for the new track and assign a unique identifier.

Predictions: Predict the state of each tracked object using the Kalman filter, projecting its position to the next frame. Update the tracking score based on motion prediction.

Track Management: Remove tracks that have not been associated with detection for a certain number of frames (lost track). Maintain and update a history of recent object detections and tracked objects to handle temporary occlusions or missed detections.

StrongSORT incorporates the described steps with its advanced modules, including the use of YOLOX-X for detection, BoT for appearance feature extraction, EMA for noise reduction, ECC for camera motion compensation, NSA Kalman for adaptive noise covariance, and a two-branch AFLink model for association score prediction. Additionally, StrongSORT employs a motion cost calculation and replaces the matching cascade with vanilla global linear assignment for improved performance.

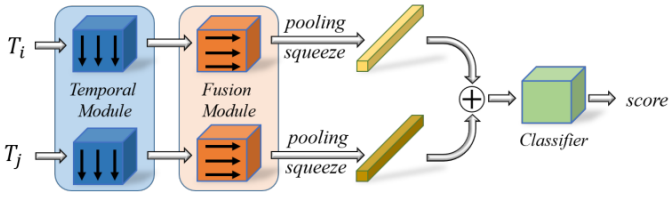

Fig. 8.

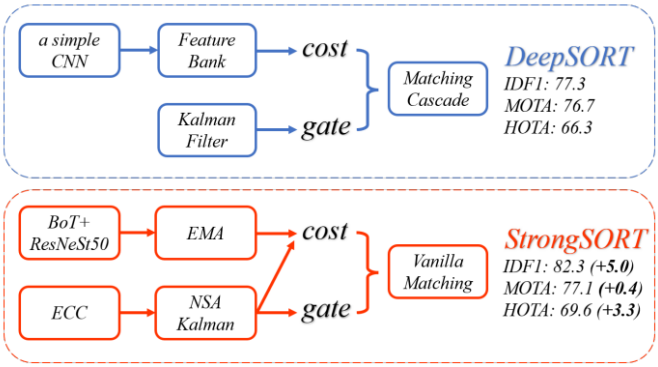

Fig. 9.

VI. Results

The results after training the Yolo model on custom dataset can be seen in the following figures

Note: nano model is represented as v8n and the small model is represented as v8s.

Labels: The classes that the YOLO (You Only Look Once) model was trained to detect are represented.

Labels Correlogram: A visualization that indicates the correlation between different labels or classes in your dataset, helping in understanding how often different classes are predicted together or are mutually exclusive, is presented.

Confusion Matrix: A table that illustrates how the models' predictions compare to the actual ground truth labels is shown.

Confusion Matrix Normalized: A version of the confusion matrix where the values are normalized to display percentages or proportions instead of raw counts is utilized for comparing the performance of different classes, especially when the class sizes vary significantly.

Results: A summary of key performance metrics for your model, such as accuracy, precision, recall, F1 score, and possibly others like mean average precision (mAP) in the case of object detection models, is provided for an overall assessment of your model's performance.

PR Curve (Precision-Recall Curve): The trade-off between precision and recall for different decision thresholds is illustrated by this curve. Precision represents the ratio of true positives to the total predicted positives, while recall is the ratio of true positives to the total actual positives. The PR curve assists in selecting an appropriate threshold that balances precision and recall for your specific use case.

R Curve (Recall Confidence Curve): The variation in your model's recall as the confidence thresholds for considering a detection as positive are adjusted is demonstrated by this curve.

P Curve (Precision Confidence Curve): Similar to the recall confidence curve, the change in your model's precision as you adjust the confidence threshold is visualized by this curve.

F1 Curve: The F1 score for different decision thresholds in your model's predictions is depicted by this curve. The F1 score, a balance between precision and recall, provides a single value that indicates the model's accuracy in classification tasks.

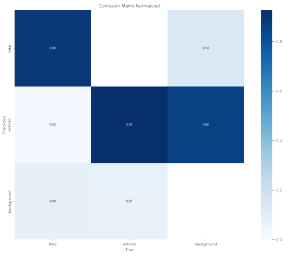

Fig. 10. v8n Normalised Confusion Matrix

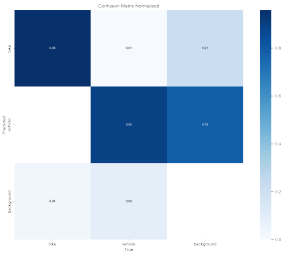

Fig. 11. v8s Normalised Confusion Matrix

From the figures, the Yolo models preformed well with score can see that the small model performed well in all the results compared to the nano model. but the frame rate is decreased by 40%. although both the models performed well small model has significant improvement the speed is getting affected. Based on the computational equipment and Graphical

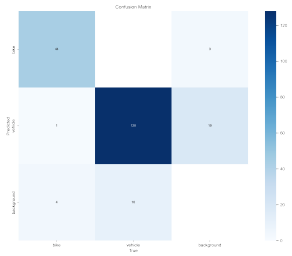

Fig. 12. v8n Confusion Matrix

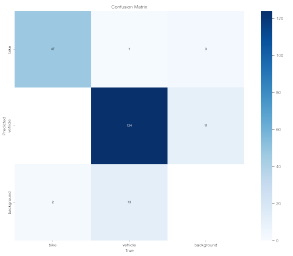

Fig. 13. v8s Confusion Matrix

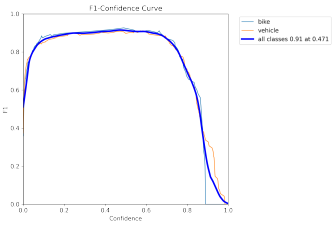

Fig. 14. v8n F1 curve

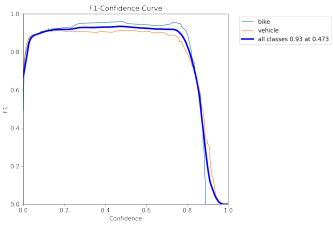

Fig. 15. v8s F1 curve

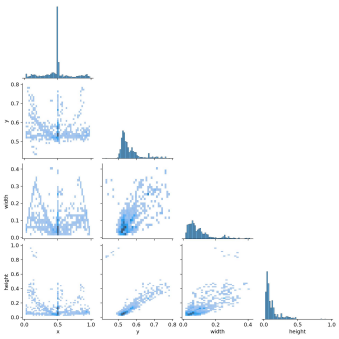

Fig. 16. v8n Labels Correlogram

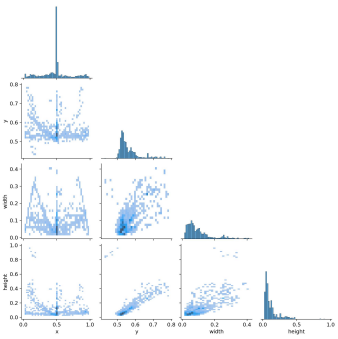

Fig. 17. v8s Labels Correlogram

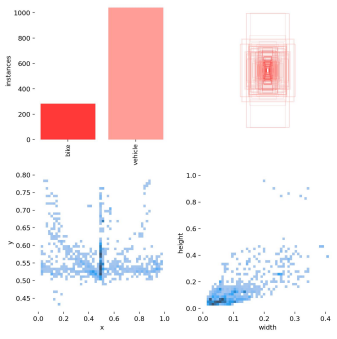

Fig. 18. v8n Labels

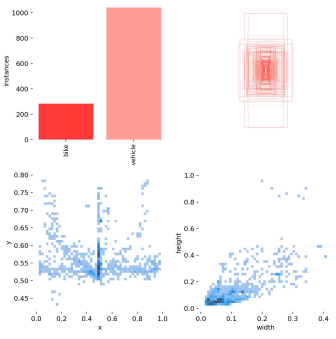

Fig. 19. v8s Labels

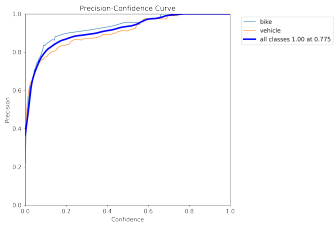

Fig. 20. v8n P curve

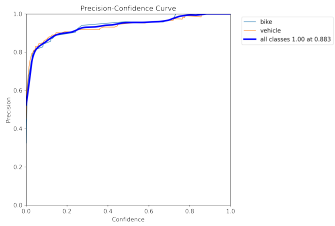

Fig. 21. v8s P curve

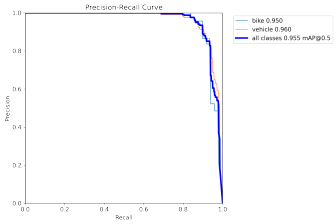

Fig. 22. v8n PR Curve

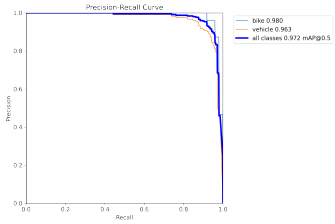

Fig. 23. v8s PR Curve

Processing unit increase the model size for better and accurate results.

VII. Discussion

The successful implementation of custom object training with YOLOv8 and object tracking using StrongSORT marks a significant milestone in advancing object detection and tracking capabilities. However, the project's potential for further enhancement and future directions remains substantial. This section delves into possible avenues for extending the project, incorporating more advanced techniques, and achieving even more accurate and comprehensive object tracking.

1. Advancing to 3D Object Tracking: A natural progression of this project involves extending the tracking capabilities into the realm of three-dimensional (3D) object tracking. This entails integrating additional sensor modalities, such as

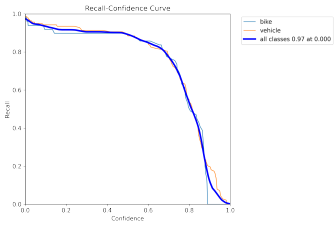

Fig. 24. v8n R curve

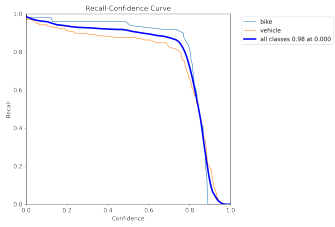

Fig. 25. v8s R curve

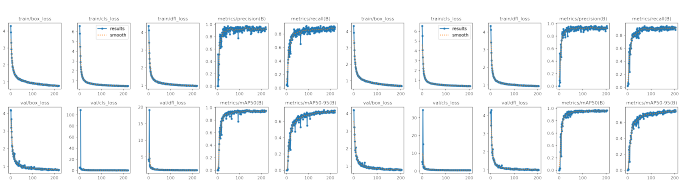

Fig. 26. v8n Final Reslts

Fig. 27. v8s Final Reslts

Fig. 28.

LiDAR, to enable precise tracking of objects in both image space and depth. By fusing the information from cameras and LiDAR sensors, the tracking system can achieve enhanced accuracy, particularly in scenarios with complex occlusions and varying distances between objects.

2. Incorporating Latest Tracking Algorithms: Continued improvement in tracking performance can be achieved by integrating and experimenting with the latest tracking algorithms. Exploring algorithms that leverage deep learning techniques or hybrid approaches can potentially enhance tracking accuracy and robustness. Algorithms that consider object interactions, anticipate object trajectories, and adapt to changing scenarios can further elevate the tracking system's capabilities.

3. Training Detection Models with Alternate Architectures: Expanding the project's scope involves training the detection model with alternate architectures beyond YOLOv8. Implementing models such as Fast R-CNN, Faster R-CNN, or EfficientDet can provide valuable insights into their performance in the specific context of object detection. Comparing and analyzing the strengths and weaknesses of different architectures can inform future decisions on selecting the most suitable model for specific application scenarios.

4. Semantic Segmentation and Contextual Understanding: Integrating semantic segmentation techniques can enhance object tracking by providing a deeper understanding of the scene's context. This approach allows the tracking system to differentiate between different object classes and understand the relationships between objects. By incorporating contextual information, the tracking system can make more informed decisions and predictions about object trajectories and interactions.

5. Real-time Fusion of Multiple Sensors: Incorporating real-time fusion of data from multiple sensors, including cameras, LiDAR, radar, and even inertial sensors, can provide a comprehensive perception system. This holistic approach enhances object detection and tracking accuracy by considering inputs from diverse sources and overcoming limitations of individual sensors.

In conclusion, the present project serves as a solid founda-

tion for future enhancements in the realm of object detection and tracking. As technology evolves, these directions have the potential to significantly advance the project's capabilities, making it more adept at handling complex scenarios, improving accuracy, and furthering its applicability in various industries, ultimately contributing to the advancement of intelligent and adaptive systems.

VIII. CONCLUSION

In conclusion, this project represents a significant step forward in the domain of object detection and tracking, harnessing cutting-edge technologies and methodologies to enhance the perception and understanding of dynamic environments. Through the utilization of custom object training with YOLOv8 for accurate detection and the integration of StrongSORT for robust object tracking, the project has demonstrated the potential for creating intelligent systems capable of real-time object analysis.

The successful implementation of these techniques underscores their effectiveness in applications such as autonomous driving, surveillance, and robotics. The seamless fusion of detection and tracking processes has the potential to revolutionize how machines interact with and respond to their surroundings, enabling safer navigation, efficient decision-making, and improved situational awareness.

Moreover, the discussions on potential future directions highlight the project's adaptability and scalability. By incorporating advancements such as 3D object tracking, latest tracking algorithms, and diverse sensor fusion, the project's capabilities can be extended to address even more complex and challenging scenarios. These future enhancements hold the promise of pushing the boundaries of object detection and tracking, contributing to the continued evolution of intelligent systems.

In essence, this project serves as a testament to the remarkable progress that can be achieved through the integration of advanced computer vision techniques and innovative algorithms. As technology continues to evolve, the insights gained from this project provide a solid foundation for further research, development, and innovation in the field of object detection and tracking, ultimately paving the way for safer, smarter, and more capable systems in the future.

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.

REFERENCES

[1] R. Szeliski, *Computer Vision*, ser. Texts in Computer Science. Springer International Publishing, 2022. [Online]. Available: https://doi.org/10.1007/978-3-030-34372-9

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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