BENCHMARKING DATASET FOR LATENCY-ACCURACY TRADEOFFS IN AUTONOMOUS VEHICLE SYSTEMS

BY

SAWAN PARUTHI

[MT23003]

Under the supervision of

Dr. Arani Bhattacharya

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

May , 2025

BENCHMARKING DATASET FOR LATENCY-ACCURACY TRADEOFFS IN AUTONOMOUS VEHICLE SYSTEMS

BY

SAWAN PARUTHI
[MT23003]

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology, Research

TO

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY DELHI

May , 2025

Certificate

This is to certify that the thesis titled *Benchmarking Dataset for Latency-Accuracy Tradeoffs in Autonomous Vehicle Systems* being submitted by *Sawan Paruthi* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Master of Technology, Research, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

May , 2025

Arani Bhattacharya

Dr. Arani Bhattacharya
Indraprastha Institute of Information Technology Delhi
New Delhi 110020

Everything can be taken from a man but the last of the human freedoms—to choose one's attitude in any given set of circumstances, to choose one's own way.

Viktor E. Frankl

The world will ask you who you are, and if you don't know, the world will tell you.

Carl Jung

Meaning is only found in the journey uphill, and never in the fleeting sense of satisfaction awaiting at the next peak.

Unknown

Acknowledgements

I would like to sincerely thank my advisor, Dr. Arani Bhattacharya, whose constant guidance, availability, and support have enabled me to navigate every obstacle encountered in the course of this project. I have learned and grown immensely through their constructive criticism and feedback on my work.

I am also grateful for my time at IIIT Delhi, which has proved to be a turning point in my life and has provided me with the opportunity to learn from exceptionally knowledgeable professors to better myself at my craft. Every lecture has been a privilege.

Sawan Paruthi

Indraprastha Institute of Information Technology, Delhi

Department of Computer Science and Engineering

Abstract

Autonomous Vehicle (AV) systems demand precise and timely processing of perception, prediction, and planning components to ensure safe and efficient operation. However, evaluating these components in real-world environments remains challenging. This thesis presents a novel framework for benchmarking AV systems, featuring a scalable gRPC-based data pipeline and fine-tuned models for tasks including object detection, semantic segmentation, tracking and audio detection. Hosted across distributed locations to emulate edge-cloud architectures, the pipeline captures latency-accuracy tradeoffs under diverse network and computational constraints. We created a comprehensive dataset of standardized performance metrics, such as latency, throughput, resource consumption, and timely accuracies, across various algorithms and scenarios, extending the Pylot platform's evaluation framework. Experimental results show that fine-tuned models outperform baseline implementations, while distributed hosting highlights network-induced latency impacts on safety and efficiency. This work provides a robust, network-aware benchmarking resource, fostering reproducible, context-dependent AV research and advancing safer, scalable autonomous driving systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid advancement of Autonomous Vehicle (AV) systems has ushered in a new era of transportation, promising safer roads, reduced human error, and enhanced mobility for individuals worldwide. At the heart of these systems lie complex components such as perception, prediction, and planning, which must operate seamlessly and in real time to ensure safe navigation through dynamic and unpredictable environments. Perception modules, responsible for tasks like object detection, semantic segmentation, tracking, and audio detection, provide the sensory foundation for AVs, while prediction and planning components enable the vehicle to anticipate future states and make informed decisions. However, the performance of these components is heavily influenced by a multitude of factors, including computational resources, network conditions, and the inherent variability of real-world scenarios. Evaluating the efficacy of AV systems in such conditions remains a significant challenge, as traditional benchmarking approaches often fail to capture the intricate interplay between latency, accuracy, and resource constraints in distributed, network-dependent architectures. This thesis addresses these challenges by proposing a novel dataset for benchmarking AV systems, with a focus on creating a scalable, network-aware evaluation pipeline that emulates real-world edge-cloud deployments, captures comprehensive performance metrics, and provides actionable insights for improving the safety and efficiency of autonomous driving systems.

A critical aspect of AV systems is the diversity of tasks they must perform, each with distinct

computational and latency requirements, which precludes the use of a single algorithm across all scenarios. For instance, when an AV operates at high speeds on a highway, the perception system must prioritize rapid object detection to identify obstacles in real time, necessitating lightweight models that sacrifice some accuracy for speed to ensure timely decision-making. Conversely, at lower speeds in urban environments, the system can afford to employ more computationally intensive models that offer higher accuracy for tasks like semantic segmentation, as the slower pace allows for greater processing time without compromising safety. This trade-off underscores the need for a flexible benchmarking framework that can evaluate a range of algorithms tailored to specific operational contexts, ensuring that AV systems can adapt dynamically to varying conditions while maintaining optimal performance.

The motivation for this work stems from the critical need to bridge the gap between controlled laboratory testing and the complexities of real-world AV deployment. While simulation environments and closed-course testing have been instrumental in the development of AV technologies, they often overlook the impact of network-induced latencies, computational bottlenecks, and resource constraints that arise in distributed systems. For instance, an AV operating in an urban environment may rely on edge servers for real-time processing of sensor data, while cloud servers handle more computationally intensive tasks such as model updates or long-term planning. The latency introduced by network communication between these layers can significantly affect the timeliness of critical decisions, potentially compromising safety. Moreover, the lack of standardized, reproducible benchmarking frameworks makes it difficult to compare the performance of different AV algorithms across diverse scenarios, hindering progress in the field. To address these issues, this work introduces a scalable gRPC-based data pipeline designed to facilitate the evaluation of AV components under realistic conditions. By hosting the pipeline across distributed locations, we emulate the edge-cloud architectures commonly employed in AV systems, allowing us to systematically study the tradeoffs between latency and accuracy under varying network and computational constraints.

A key contribution of this thesis is the development of a comprehensive dataset that captures standardized performance metrics, including latency, throughput, resource consumption, and timely accuracies, across a range of algorithms and scenarios. This dataset extends the evaluation

capabilities of the Pylot [1] platform, a widely used framework for AV research, by integrating new metrics and scenarios that reflect real-world challenges. To ensure the robustness of our benchmarking framework, we fine-tuned models for critical AV tasks, including object detection, semantic segmentation, tracking, and audio detection. These fine-tuned models serve as a benchmark for evaluating the performance of AV components, revealing significant improvements over baseline implementations. Experimental results from our framework highlight the superior performance of these fine-tuned models, while also underscoring the impact of network-induced latencies on safety and efficiency. For example, we observed that delays in data transmission between edge and cloud servers can lead to degraded performance in time-sensitive tasks such as object detection, emphasizing the need for network-aware design in AV systems.

The broader implications of this work lie in its potential to foster reproducible, context-dependent research in the AV domain. Going forward, we will review related works, identify gaps, outline service and task requirements, list models used, describe the gRPC [2] pipeline and dataset structure, and conclude with future aspects. By providing a robust benchmarking resource that accounts for network dynamics, computational constraints, and scenario variability. The framework and dataset presented herein offer researchers and practitioners a standardized toolset for evaluating AV components, enabling fair comparisons across algorithms and facilitating the identification of performance bottlenecks. Furthermore, the insights gained from our experiments—particularly regarding the interplay between network latency and system performance—can inform the design of future AV architectures, ensuring that they are better equipped to handle the demands of real-world deployment. Ultimately, this work seeks to contribute to the ongoing effort to make autonomous vehicles a reliable and safe mode of transportation, capable of navigating the complexities of the modern world with precision and efficiency.

Chapter 2

Related Work in Autonomous Vehicle Benchmarking

2.1 Literature Review

The development of Autonomous Vehicle (AV) systems relies heavily on robust datasets and evaluation frameworks that enable researchers to benchmark and refine algorithms for perception, prediction, and planning. Over the past decade, several seminal works have contributed to this domain by providing standardized datasets with annotated sensor data and simulation-based frameworks for testing AV components. These resources have been instrumental in advancing AV research, particularly in areas such as object detection, semantic segmentation, and motion planning. However, as AV systems transition from controlled environments to real-world deployments, new challenges emerge, particularly related to network-induced latencies, distributed computational architectures, and the need for comprehensive Quality of Service (QoS) metrics. This chapter reviews key contributions in AV datasets and benchmarking frameworks, focusing on the KITTI [3], Cityscapes [4], and WSDream [5] datasets, as well as the Pylot [1] framework, and identifies the gaps that necessitate the novel benchmarking approach proposed in this thesis.

2.1.1 Datasets for Autonomous Vehicle Research

The KITTI dataset, introduced by Geiger et al. (2012) [3], is a foundational benchmark for AV research, offering a comprehensive collection of sensor data and images captured in real-world driving scenarios in Karlsruhe, Germany. The dataset includes stereo camera images, LiDAR point clouds, GPS/IMU data, and 3D object annotations, covering over 200,000 objects across various tasks such as object detection, tracking, stereo matching, and optical flow estimation. KITTI's real-world data, collected using a vehicle equipped with multiple sensors, provides a realistic testbed for evaluating perception algorithms, making it a widely adopted resource in the AV community. However, KITTI [3] is a static dataset that does not include QoS metrics—such as latency, throughput, or resource consumption—when running different models on its data, limiting its ability to assess the practical performance of AV systems under dynamic, network-dependent conditions.

Similarly, the Cityscapes dataset, presented by Cordts et al. (2016) [4], focuses on urban environments and provides a detailed collection of sensor data and images for semantic and instance segmentation tasks. Comprising 5,000 finely annotated images and 20,000 coarsely annotated images across 50 cities, Cityscapes [4] captures diverse weather and lighting conditions, with pixel-level annotations for 30 classes, including vehicles, pedestrians, and road surfaces. This dataset has become a standard benchmark for segmentation algorithms, which are critical for AV navigation in complex urban settings. Like KITTI [3], however, Cityscapes [4] lacks data on QoS metrics when running various models on its annotated images, restricting its utility for evaluating the real-world performance of AV systems in distributed architectures where network latency and computational constraints play a significant role.

Another dataset relevant to QoS evaluation is the WSDream dataset, which focuses on Quality of Service metrics for web services, as described by Zheng et al. (2010) [5]. WSDream [5] includes real-world QoS evaluation results from 339 users on 5,825 web services, utilizing SOAP-based pipelines to collect metrics such as response time and throughput, along with location information (e.g., IP, latitude, longitude) for users and services. While WSDream [5] provides valuable QoS data, the many of its services are not relevant to AV systems, as they

primarily pertain to general web service applications rather than the specific sensor-driven, real-time requirements of autonomous driving. This mismatch in domain relevance, coupled with its focus on web service interactions rather than AV-specific tasks, limits its direct applicability to AV research, highlighting the need for a dataset tailored to AV systems with a focus on distributed, network-aware QoS metrics.

2.1.2 Limitations of Existing Datasets

While KITTI [3], Cityscapes [4], and WSDream [5] have contributed significantly to their respective fields, their limitations hinder their applicability to real-world AV deployment scenarios. KITTI [3] and Cityscapes [4], as static datasets of sensor data and images, do not provide insights into the QoS metrics—such as latency, throughput, or resource consumption—required to evaluate the performance of different models under varying computational and network conditions. WSDream [5], although rich in QoS metrics, is not tailored for AV systems, as its services are primarily general web services rather than AV-specific applications, and it is bean implemented over SOAP based APIs which are slow and outdated for AV perception tasks and it also lacks the sensor-driven context necessary for autonomous driving tasks. In contrast, the dataset developed in this thesis addresses these limitations by focusing on QoS metrics obtained by running a diverse set of algorithms on AV-specific tasks, offering a practical perspective on model performance in distributed, network-aware environments.

In the domain of AV system evaluation, Gog et al. (2019) introduced Pylot [1], a modular and scalable framework for benchmarking AV components within the CARLA simulator [6]. Pylot [1] enables researchers to test a set of services, including object detection, semantic segmentation, tracking, motion planning, and prediction, either independently or as part of an integrated pipeline. The framework collects performance metrics such as latency and accuracy in a controlled simulation environment, making it a valuable tool for AV research. Through our engagement with Pylot [1], we gained a deeper understanding of the relevant tasks in AV systems, such as the interplay between perception and planning, which directly informed the design of our evaluation framework. However, Pylot's [1] simulation-based approach does not

account for network-induced latencies or the challenges of distributed edge-cloud architectures, which are critical for practical AV deployment.

Despite its strengths, Pylot's [1] focus on simulation limits its ability to evaluate AV systems in real-world scenarios where network dynamics and distributed computational setups are prevalent. The framework lacks mechanisms to assess the impact of network latencies on performance, a critical factor for AV systems that rely on edge-cloud architectures for real-time processing. This limitation highlights the need for a benchmarking framework that integrates network-aware evaluation and distributed computational constraints, a gap that this thesis aims to address.

2.1.3 Research Gap and Motivation

The reviewed literature reveals a significant gap in the evaluation of AV systems: the absence of a comprehensive framework that integrates real-world network dynamics, distributed computational architectures, and QoS-focused evaluation across varied scenarios. Datasets like KITTI [3] and Cityscapes [4] provide rich sensor data and annotations for algorithm development but do not capture the QoS metrics necessary to assess the practical performance of AV models in dynamic, network-dependent environments. WSDream [5], while offering QoS metrics, is not suited for AV systems due to its focus on general web services, which are largely irrelevant to the sensor-driven, real-time requirements of autonomous driving. Similarly, frameworks like Pylot [1], which relies on the CARLA simulator [6], while effective for simulation-based testing, fails to address the challenges of network-induced latencies and distributed systems, instead, it allows us to simulate various AV algorithm setups within its environment but does not provide insights into the comparison of latency-accuracy tradeoffs across different sets of algorithms. This thesis addresses these gaps by introducing a novel gRPC-based data pipeline, distributed hosting to emulate edge-cloud interactions, and a dataset that captures QoS metrics—such as latency, throughput, resource consumption, and timely accuracies—across a range of algorithms and scenarios tailored for AV systems. By extending Pylot's [1] capabilities to include real-world scenarios and network-aware metrics, this work provides a more holistic and practical evaluation of AV systems, paving the way for safer and more scalable autonomous driving technologies.

2.2 Background

2.2.1 Relevant Tasks in AVs

Autonomous Vehicles (AVs) operate through a complex pipeline of tasks that enable them to perceive their environment, predict future states, and plan safe navigation paths. These tasks are critical for ensuring the reliability, safety, and efficiency of AV systems in dynamic, real-world scenarios. Pylot [1], a modular framework for AV benchmarking, provides a comprehensive set of tasks that are widely recognized in the AV research community. Within the CARLA simulator [6], Pylot supports the evaluation of key tasks such as object detection, semantic segmentation, tracking, motion prediction, and motion planning. Object detection involves identifying and localizing objects (e.g., vehicles, pedestrians) in the AV's surroundings using sensor data like LiDAR and camera images. Semantic segmentation assigns a class label to each pixel in an image, enabling the AV to understand the scene at a granular level, such as distinguishing between roads, sidewalks, and obstacles. Tracking ensures that detected objects are consistently monitored over time, while motion prediction forecasts the future trajectories of dynamic objects, such as other vehicles or pedestrians. Finally, motion planning generates safe and efficient paths for the AV to navigate, balancing constraints like traffic rules and obstacle avoidance. These tasks, as implemented in Pylot [1], are interdependent, with the output of one task (e.g., object detection) serving as input to another (e.g., tracking), making their seamless integration crucial for overall system performance. Understanding these tasks is foundational for evaluating AV systems, particularly when assessing their performance under real-world constraints like network latencies and distributed computational architectures.

These tasks are the backbone of AV systems because they collectively enable the vehicle to operate autonomously in unpredictable environments, directly impacting safety and reliability. For instance, a failure in object detection could result in the AV missing a cyclist, while inaccurate motion prediction might lead to an unsafe lane change. The interdependence of these tasks—where the output of one task (e.g., object detection) serves as input to another (e.g., tracking)—means that errors in one stage can propagate through the pipeline, potentially leading to catastrophic outcomes. Moreover, audio perception tasks add an additional layer of safety by

addressing scenarios where visual sensors alone are insufficient, such as detecting an emergency vehicle approaching from behind a corner. Figure 2.1 illustrates the AV task pipeline, showing the flow from perception (visual and audio) to tracking, prediction, and planning, highlighting their interconnected roles in ensuring safe navigation.

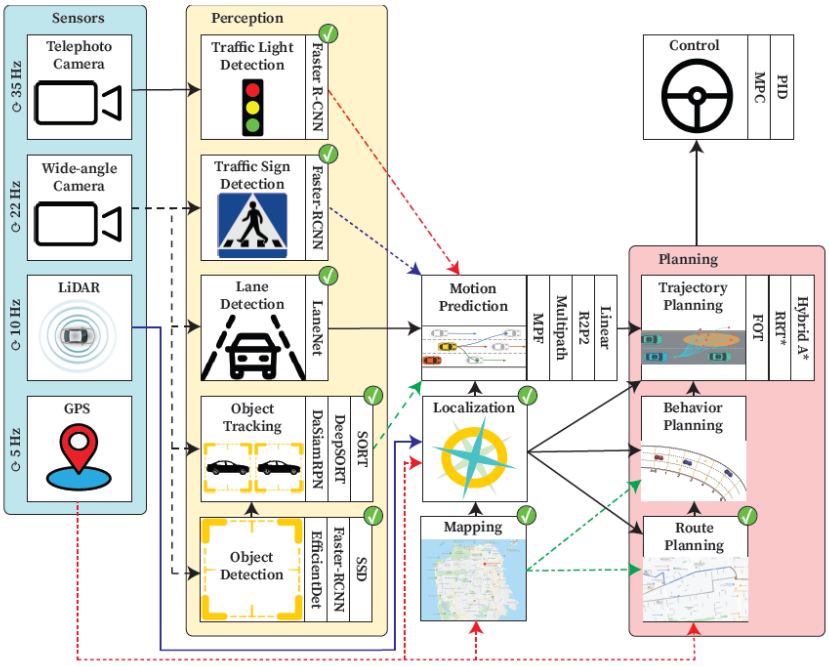

Figure 2.1: An AV pipeline consists of several interconnected modules (e.g. perception, planning).

The critical nature of these tasks underscores their importance for safety in AV systems. In real-world scenarios, AVs must handle diverse challenges, such as adverse weather conditions, occlusions, and unpredictable pedestrian behavior, all of which test the robustness of perception, prediction, and planning. For example, semantic segmentation ensures the AV can navigate a rain-soaked road by accurately identifying lane markings despite reflections, while motion planning ensures the vehicle slows down in a school zone to account for potential child pedestrians. Pylot's [1] implementation of these tasks provides a standardized framework for evaluation, but understanding their performance under real-world constraints—such as network latencies and distributed computational architectures—remains a challenge, as discussed in later sections. By studying these tasks, researchers can identify bottlenecks, optimize algorithms, and ensure that AV systems meet the stringent safety requirements necessary for widespread adoption.

2.2.2 QoS Parameters

Quality of Service (QoS) parameters are essential metrics for evaluating the performance of AV systems, especially in distributed and network-dependent environments. In the context of AVs, key QoS parameters include latency, throughput, resource consumption, and accuracy, each of which directly impacts the system's ability to operate safely and efficiently. Latency refers to the time taken for data to travel between components, such as from sensors to a processing unit or between edge and cloud servers in a distributed setup. For AVs, low latency is critical, as delays in processing sensor data can lead to outdated decisions, potentially compromising safety. Throughput measures the rate at which data is processed, such as the number of frames per second an object detection algorithm can handle, which determines the system's ability to handle high-frequency sensor inputs. Resource consumption encompasses the computational resources (e.g., CPU, memory, bandwidth) required by AV algorithms, which is particularly relevant in distributed architectures where resources are often constrained, such as on edge devices. These QoS parameters are interdependent; for instance, achieving higher accuracy in object detection might increase latency or resource consumption, leading to tradeoffs that must be carefully managed. Evaluating these parameters provides a holistic view of an AV system's performance, especially under the real-world conditions that this thesis aims to address.

2.2.3 Challenges in Evaluating QoS for AV Systems

Evaluating QoS parameters in AV systems presents several challenges, particularly when transitioning from simulation to real-world deployment. Simulation-based frameworks, which operates within the CARLA simulator [6], provide controlled environments to test AV tasks, but they often fail to capture the complexities of real-world network dynamics and distributed architectures. Network-induced latencies, such as those caused by intermittent connectivity or bandwidth limitations in edge-cloud setups, are difficult to simulate accurately and can significantly impact latency and throughput. Moreover, distributed systems introduce additional challenges, such as data synchronization between edge and cloud components, which can lead to inconsistencies in QoS measurements. Another challenge is the lack of standardized datasets that capture QoS

metrics across a range of AV algorithms and scenarios; existing datasets like KITTI [3] and Cityscapes [4] provide sensor data but do not include QoS-related information, while datasets like WSDream [5] focus on web services rather than AV-specific tasks. Finally, the interdependence of QoS parameters—such as the latency-accuracy tradeoff—complicates evaluation, as optimizing one parameter (e.g., reducing latency) may degrade another (e.g., accuracy). These challenges highlight the need for a comprehensive benchmarking framework that can evaluate QoS in real-world, network-aware scenarios, a gap that this thesis aims to address.

2.2.4 Importance of Studying QoS in AV Systems

Studying QoS in AV systems is crucial for several reasons, directly tied to the goals of safety, efficiency, and scalability in autonomous driving. First, understanding QoS parameters like latency and accuracy ensures that AV systems can make timely and correct decisions, which is essential for safety-critical applications where delays or errors can lead to accidents. For example, a motion planning algorithm that experiences high latency may fail to adjust the AV's trajectory in time to avoid an obstacle, endangering passengers and pedestrians. Second, evaluating throughput and resource consumption enables the optimization of AV systems for resource-constrained environments, such as edge devices, which is vital for scalability. As AVs are deployed at scale, efficient resource usage ensures that computational demands do not overwhelm available infrastructure, particularly in distributed setups where edge-cloud communication is prevalent. Third, studying QoS provides insights into the tradeoffs between parameters, such as latency versus accuracy, allowing researchers to design algorithms that balance these factors effectively. For instance, an object detection algorithm might achieve high accuracy at the cost of increased latency, which could be mitigated by optimizing the algorithm or offloading computation to the cloud—decisions informed by QoS analysis. Finally, QoS evaluation is essential for building trust in AV technology; by demonstrating that AV systems can meet stringent performance requirements under real-world conditions, researchers can pave the way for broader adoption and regulatory approval. This thesis emphasizes the importance of QoS by developing a framework that captures these metrics in distributed, network-aware scenarios, addressing a critical need in AV research.

Chapter 3

gRPC-Based Services: Design and Rationale

In this chapter, we first explain the need of benchmarking, then methodology of benchmarking and finally the results. We then utilize these results to reach our conclusion.

3.1 Importance of Benchmarking Datasets for Autonomous Vehicles

The development of autonomous vehicle (AV) systems requires thorough evaluation of components like detection, segmentation, and tracking to ensure safety and efficiency in dynamic environments. Existing platforms, such as Pylot, focus on latency-accuracy tradeoffs but lack comprehensive, standardized benchmarks for modern models. This work addresses this gap by introducing a gRPC-based data pipeline and a robust benchmarking dataset, enabling detailed evaluation of fine-tuned AV models across diverse scenarios. The dataset standardizes performance metrics, including timely accuracies, fostering reproducible research. By providing optimized models and a scalable framework, it extends Pylot's capabilities, offering insights into context-dependent component performance and driving outcomes. This work is crucial for advancing safer, more efficient AV systems, bridging the gap between simulation and real-world applications.

3.2 Technical Specifications of Data Pipeline

The data pipeline developed for this thesis is a critical component of the proposed framework, enabling efficient and scalable communication. This section outlines the technical specifications, focusing on the application programming interface (API) and the rationale for adopting gRPC as the underlying framework.

3.2.1 Application Programming Interface

The data pipeline is built using gRPC, a high-performance remote procedure call (RPC) framework, implemented in Python. The gRPC-based APIs enable seamless data streaming between AV Services (e.g., object detection, tracking) via modular, service-oriented independent endpoints. Using the grpcio library, service definitions are specified in Protocol Buffers (protobuf) .proto files, which generate Python stubs for efficient, type-safe communication.

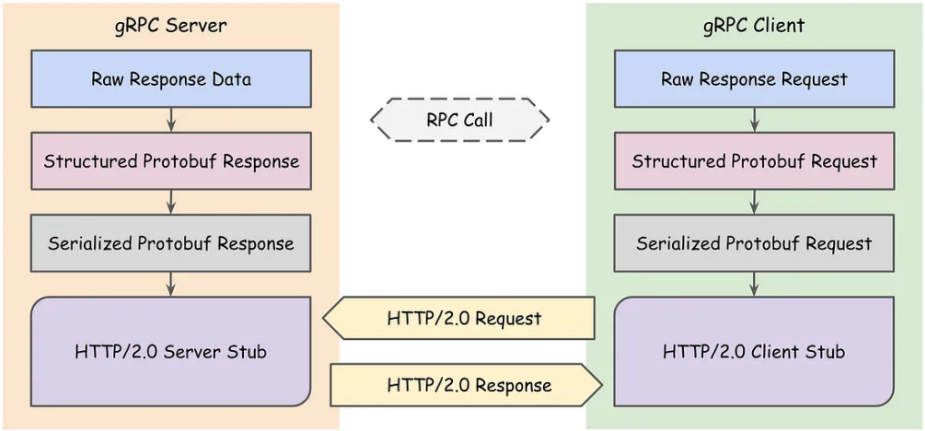

Figure 3.1: The process of gRPC call

3.2.2 Rationale for Choosing gRPC

GRPC was selected over REST APIs for its superior performance and suitability for AV systems. Key advantages include:

Performance - gRPC uses Protocol Buffers (protobuf), a binary serialization format, instead

of JSON, reducing overhead and enabling faster data transmission. This is critical for latency-sensitive AV applications.

HTTP/2 Support - Built on HTTP/2, gRPC supports multiplexing and bidirectional streaming, minimizing latency in high-throughput scenarios compared to REST's HTTP/1.1.

Scalability - gRPC's lightweight communication model supports distributed setups, aligning with the thesis's evaluation of latency-accuracy tradeoffs in varied conditions.

Comparison with REST APIs - REST APIs, typically reliant on JSON for data exchange, incur higher serialization overheads compared to gRPC, making them less suitable for real-time autonomous vehicle (AV) pipelines. While modern REST APIs can leverage HTTP/2 to reduce latency they still face inefficiencies due to JSON's text-based format, which requires more bandwidth and parsing time than gRPC's compact, binary Protocol Buffers (protobuf). gRPC's type-safe design, enabled by protobuf schemas, enhances developer productivity and debugging by ensuring strict data contracts, aligning with Pylot's emphasis on debuggability and modularity. This efficiency makes gRPC an optimal choice for scalable, high-performance AV benchmarking.

By leveraging gRPC, the pipeline minimizes latency, supports timely accuracies, and extends Pylot's framework for robust, scalable AV benchmarking.

3.3 AV Perception Services

3.3.1 Object Detection Service

Our object detection service comprises a comprehensive suite of 60 pretrained models, designed to support robust benchmarking of autonomous vehicle (AV) perception tasks. These models, pretrained on the COCO dataset, include various YOLO[7] architectures—namely YOLOv3, YOLOv5, YOLOv6, YOLOv8, YOLOv9, YOLOv11, and YOLOv12—alongside TorchVision-based models[8] such as Faster R-CNN, ResNet, MobileNet, and SSD. This diverse set enables a thorough evaluation of detection performance across latency-accuracy tradeoffs, aligning with the Pylot platform's focus on timely accuracies, which measure accuracy adjusted for runtime.

Integrated into our gRPC-based data pipeline, the service ensures efficient, scalable inference for AV benchmarking.

3.3.2 License Plate Detection Service

We fine-tuned the aforementioned 60 detection models for license plate detection, utilizing a dataset of over 2,000 images. The dataset encompasses a wide variety of license plates, including those for 2-wheelers and 4-wheelers, with colors such as green, white, and yellow, captured in both daytime and nighttime conditions. This fine-tuning optimized the models—comprises Ultralytics YOLO [7] and TorchVision-based models [8] enhanced performance in AV perception tasks. The fine-tuned models achieved an F1 score exceeding 90% for most models, demonstrating their effectiveness for real-world license plate detection within the proposed benchmarking framework.

3.3.3 Brake Light Detection Service

We fine-tuned the aforementioned 60 detection models for brake light detection, using a dataset comprising over 1,500 images. The dataset captures a diverse set of brake lights from both 2-wheelers and 4-wheelers, encompassing various designs such as short and long bar lights, and colors like red and amber. These images in dataset consist of varying lighting conditions, including both daytime and nighttime scenarios. The fine-tuned models—based on Ultralytics YOLO [7] and TorchVision frameworks—were able to effectively distinguish between glowing backlights at night and actual brake lights activated during braking. This fine-tuning process significantly enhanced the models performance in AV's perception tasks. Most models achieved an F1 score exceeding 90%, highlighting their robustness and effectiveness in real-world brake light detection within the proposed benchmarking framework.

3.3.4 Traffic Lights Detection Service

For Traffic Light Detection, We also fine-tuned the aforementioned 60 detection models for Traffic light detection, using a dataset comprising over 2,500 labled images. The dataset captures a diverse set of traffic lights, encompassing various designs such as short and long bar lights, and images were labled for 4 classes, Red, Green, Orange and Off. These images in dataset consist of varying lighting conditions, including both daytime and nighttime scenarios. The fine-tuned models—based on Ultralytics YOLO [7] and TorchVision frameworks—were able to effectively distinguish between Red, Green, Orange and Off Traffic lights. This fine-tuning process significantly enhanced the models performance in AV's perception tasks. Most models achieved an F1 score exceeding 90%, highlighting their robustness and effectiveness in real-world brake light detection within the proposed benchmarking framework.

3.3.5 Object Tracking Service

Object tracking involves continuously monitoring a detected object across multiple frames in a video segment to ensure consistent identification in dynamic AV scenarios. Our object tracking service employs eight established tracking algorithms: BoTSORT [9], SORT [10], DeepSORT [11], StrongSORT [12], ByteTrack [13], NorFair [14], TrackPy [15], and OCSORT [16]. Each algorithm can be paired with any of the 60 previously mentioned detection models, yielding over 300 unique tracking combinations. The service processes video clips from diverse AV contexts as input, recording Quality of Service (QoS) parameters such as response time, latency, and resource consumption. These metrics enable a detailed evaluation of tracking performance, supporting latency-accuracy tradeoff analysis, including Pylot's timely accuracies, within our AV benchmarking framework.

3.3.6 Depth Estimation Service

Depth estimation involves generating depth maps from input images to approximate the depth of objects, providing critical spatial understanding for autonomous vehicle (AV) systems. This

task is essential for AVs, as depth information enables the estimation of action response times, improving navigation, obstacle avoidance, and overall safety in dynamic environments. Our depth estimation service employs MiDaS [17], a deep learning framework utilizing PyTorch-based Vision Transformers (ViTs) [18], a monocular depth estimation algorithm, that generates accurate depth maps using images from a single camera, eliminating the need for stereo setups. We utilized 18 pretrained MiDaS [17] models, covering various architectures and configurations, to ensure robust depth estimation across diverse scenarios. Integrated into our gRPC-based pipeline, this service supports comprehensive evaluation of depth estimation performance, enhancing AV perception tasks within our benchmarking framework.

3.3.7 Audio Segmentation and Classification Service

Audio processing is a critical task for autonomous vehicles (AVs), enabling the detection and interpretation of environmental sounds to enhance situational awareness and safety. Our audio processing service focuses on two primary tasks: segmentation and augmentation. Segmentation involves detecting and classifying audio events along with their timestamps in a clip, while augmentation identifies the presence of various classified audio types within the clip. We employed PANNs Inference [19], a deep learning framework utilizing PyTorch-based convolutional neural networks (CNNs), for audio event detection. The service leverages 27 pretrained PANNs Inference [19] models, facilitating robust audio analysis to support AV perception within our benchmarking framework.

Table 3.1: Models Used for Each Task in the Autonomous Vehicle Benchmarking Dataset

Task	**Models**
Object Detection[1]	• **Ultralytics** [7]: YOLOv3, YOLOv5, YOLOv6, YOLOv8, YOLOv9, YOLOv10, YOLOv11 • **TensorFlow** [**?**]: EfficientDet D0, D1, D2, D3, D4, D5; Faster R-CNN ResNet50, ResNet50v2, ResNet101; RetinaNet ResNet50; SSD MobileNet V2 • **TorchVision** [8]: FasterRCNN-ResNet50-FPN, FasterRCNN-MobileNet-V3-Large-FPN, FasterRCNN-ResNet50-FPN-V2, RetinaNet-ResNet50-FPN, SSD300-VGG16, SSDLite320-MobileNet-V3-Large, FCOS-ResNet50-FPN
Object Tracking	BoTSORT [9], SORT [10], DeepSORT [11], StrongSORT [12], OCSORT [16], NorFair [14], TrackPy [15] (each paired with all detection models)
Depth Estimation	**MiDaS** [17]: dpt_beit_large_512, dpt_beit_large_384, dpt_beit_base_384, dpt_swin2_large_384, dpt_swin2_base_384, dpt_swin2_tiny_256, dpt_swin_large_384, dpt_next_vit_large_384, dpt_levit_224, dpt_large_384, dpt_hybrid_384, midas_v21_384, midas_v21_small_256, openvino_midas_v21_small_256
Audio Detection	**PANNs** [19]: Cnn6, Cnn10, Cnn14_8k, Cnn14_16k, Cnn14_DecisionLevelAtt, Cnn14_DecisionLevelMax, Cnn14_emb32, Cnn14_emb128, Cnn14_emb512, Cnn14, DaiNet19, LeeNet11, LeeNet24, MobileNetV1, MobileNetV2, Res1dNet31, Res1dNet51, ResNet22, ResNet38, ResNet54, Wavegram_Cnn14, Wavegram_Logmel_Cnn14

[1]Object Detection includes sub-tasks such as General Detection, License Plate Detection, Traffic Lights Detection, and Brake Light Detection. General Detection uses all listed models, while the sub-tasks use only Ultralytics and TorchVision models.

3.4 QoS Parameters in AV Benchmarking: Analysis and Measurement

3.4.1 Latency

Network latency is a critical Quality of Service (QoS) parameter for autonomous vehicle (AV) systems, affecting the timeliness of communication between distributed components. We measure network latency as the round-trip time for a request to travel from the client to the server and the response to return from the server to the client, excluding the server's processing time. This metric is evaluated at the client side using Python's time library. Once measured, the latency

time is sent to the server in a separate request for further analysis.

3.4.2 Response Time

Response time, in the context of our system, refers to the total duration from when a user sends a request to the server, the server processes it, and the user receives the response. We measure this parameter by capturing the latency at the client side, which includes network transit time and we add server processing time in it. Response time largely depends on the configuration of server and the algorithm that is being used in the service.

3.4.3 CPU Consumption

CPU usage, in the context of our system, refers to the computational power consumed by the server while processing requests for various AV services. We measure CPU usage at the server side with Python's psutil library, which provides detailed insights into resource utilization by tracking CPU percentage during request processing. This measurement enables the analysis of resource requirements for different algorithms across services like object detection and tracking, facilitating an optimized resource allocation strategy in an AV systems.

3.4.4 Memory Consumption

Memory usage, in the context of our system, refers to the amount of memory consumed by the server while executing AV service requests, such as object tracking or depth estimation. We measure memory usage with Python's psutil library, which tracks memory allocation by monitoring the resident set size (RSS) during request processing. This measurement provides insights into the memory demands of various algorithms across different services, enabling efficient resource management and optimization for real-time AV applications.

3.4.5 Throughput

Throughput, in the context of our system, refers to the rate at which the server processes data for AV services, calculated as the size of the input image divided by the processing time, expressed in megabits per second (Mbps). We measure throughput by recording the image size and the server's processing time in seconds for each request, focusing on services like object detection and depth estimation. This metric is crucial for AV systems, as high throughput ensures the server can handle large volumes of data—such as high-resolution camera feeds—in real time, directly impacting the ability to perform time-sensitive tasks like obstacle detection or navigation. By evaluating throughput, we gain insights into the server's capacity to support scalable, efficient performance, enabling the optimization of resource allocation for real-world AV deployments.

Table 3.2: QoS Parameters

Quality of Service Parameters	**Unit of Measure**
Latency	Milliseconds (ms)
Response Time	Milliseconds (ms)
CPU Consumption	Percentage (%)
Memory Consumption	Megabytes (MB)
Throughput	Megabit/Second (Mbps)

3.5 Geo-Location Parameters

Location-based performance metrics involve capturing the geo-locations of both the server and the client to analyze their impact on AV service performance. We collect geo-location data by recording the IP addresses of the client and server, using ip-api to extract parameters such as longitude, latitude, city, region, and country, and ipwhois to obtain the ASN. This data enables the study of geo-location effects on latency-accuracy tradeoffs, providing insights into optimal server-client placement strategies for real-world AV scenarios.

3.5.1 IP Address

An IP Address is a unique numerical label assigned to each device connected to a network, used for identification and communication. We record the IP addresses of the client and server to determine their geo-locations, facilitating the analysis of network latency in AV systems.

3.5.2 ASN and ASN Description

An Autonomous System Number or ASN is a unique identifier assigned to an autonomous system (a collection of IP routing prefixes) managed by a single entity, used for routing data across the internet. The ASN Description provides details about the autonomous system, such as its owner or operator. Using ipwhois, we extract the ASN and its description to understand the network infrastructure impacting AV service performance.

3.5.3 Subnet Mask and Subnet

A Subnet Mask is a 32-bit number that divides an IP address into network and host portions, defining the Subnet—a smaller network within a larger one for efficient routing. We use the Subnet Mask to identify the Subnet of the client and server, aiding in the study of localized network effects on latency.

3.5.4 Latitude, Longitude, City, Region, and Location

Latitude and Longitude are coordinates specifying a point on Earth (in degrees), while City, Region, and Location (a general term for the geographical area) provide contextual place information. Using ip-api, we extract latitude, longitude, city, region, and country to map the geographical locations of the client and server, enabling the analysis of geo-spatial impacts on AV performance.

Chapter 4

gRPC Pipeline workflow

The gRPC-based pipeline serves as the backbone of our autonomous vehicle (AV) benchmarking data pipeline, enabling efficient communication and data processing across various perception services, such as object detection, tracking, and depth estimation. This chapter details the workflow of the pipeline, focusing on how it handles requests, integrates services, and collects Quality of Service (QoS) parameters to evaluate performance. Understanding the workflow is essential for ensuring low-latency, scalable operations in real-world AV scenarios, where timely and accurate processing directly impacts safety and navigation. Going forward in this chapter we introduces foundational concepts and pipeline setup, explore an overview of the gRPC workflow, and then we discusses the request handling and service integration, followed by the pipeline execution methodology, outlining the step-by-step process of data processing and performance evaluation.

4.1 Preliminaries

This section provides the foundational concepts necessary to understand the gRPC workflow within our pipeline. The pipeline leverages gRPC, a high-performance remote procedure call (RPC) framework, to facilitate communication between distributed components, as discussed in Section 3.2. gRPC uses Protocol Buffers (protobuf) for efficient data serialization and supports

bidirectional streaming over HTTP/2, making it ideal for low-latency AV applications. Our pipeline is designed as a modular, service-oriented architecture, where each AV perception task—such as object detection, license plate detection, and audio segmentation—operates as an independent gRPC service. These services are hosted on a server, accessible by a client (e.g., an AV system or a benchmarking tool), and communicate via predefined protobuf messages. The workflow assumes a stable network connection and a server capable of handling high-volume requests, ensuring real-time performance for AV scenarios. This setup enables the pipeline to process diverse inputs, such as video frames and audio clips, while collecting QoS parameters like latency and throughput for performance evaluation.

4.2 Overview of the gRPC Pipeline Workflow

The gRPC pipeline workflow orchestrates the flow of data between the client and server to support AV perception tasks within our benchmarking framework. The process begins with the client—representing an AV system or a benchmarking tool—sending a request, such as a video frame for object detection or an audio clip for segmentation, to the server via a gRPC call. The request is serialized into a Protocol Buffers (protobuf) message, transmitted over HTTP/2, and received by the server. Each service processes the input, generates a response (e.g., detected objects, depth maps), and collects QoS parameters, such as latency and throughput, during execution. The response is then serialized back into a protobuf message and sent to the client, and as a next step client sends a log request containing the response time (calculated at client side) to the server, completing the cycle. This workflow ensures efficient, low-latency communication, enabling real-time processing for AV applications while facilitating performance evaluation through QoS metrics.

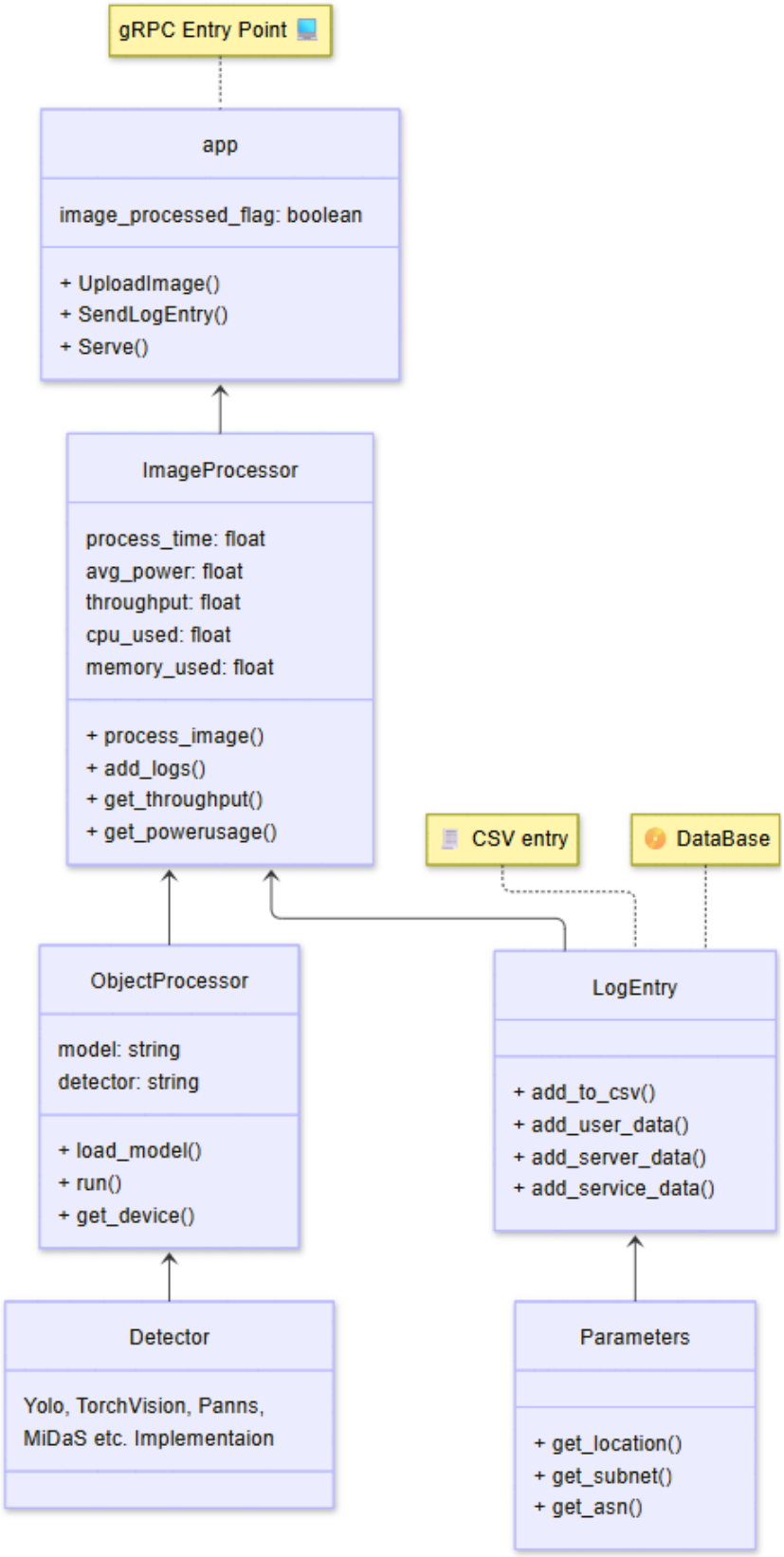

Figure 4.1: Low-level design of the gRPC server-side data flow.

4.3 Request Handling and Service Integration

The gRPC pipeline efficiently manages requests by routing them to the appropriate autonomous vehicle (AV) perception service, ensuring seamless integration across diverse tasks. When a client sends a request—such as a video frame for object tracking or an image for license plate detection—it invokes the `UploadImage()` stub to transmit the data to the target service. The server's gRPC endpoint deserializes the Protocol Buffers (protobuf) message and forwards the request to the corresponding algorithm/model the message (e.g.,Yolo, MiDaS, Panns etc.). The service processes the input using its fine-tuned models, such as YOLOv8 for object detection or MiDaS for depth estimation, as described in Section 3.3. During processing, the service collects Quality of Service (QoS) parameters, including latency and CPU usage, following the methodology outlined in Section 3.4. The service then generates a response (e.g., bounding boxes for detected objects, depth maps), which is serialized into a protobuf message and returned to the client via the gRPC endpoint. Subsequently, the client sends the QoS parameters calculated on its side, such as response time, in a separate request using the `SendLogEntry()` stub to the server. This design leverages asynchronous gRPC calls with a stateless backend to minimize latency and memory overhead, ensuring efficient operations. This modular integration enables the pipeline to support a wide range of AV perception tasks while maintaining low-latency communication and facilitating performance benchmarking through comprehensive QoS metrics.

4.4 Client-Side Interface

To support data collection at the client side, we implemented a script that automates the selection of services and models while introducing sparsity in the data collection process. The script randomly selects a service (e.g., object detection, depth estimation) and selects atleast 75% models randomly (e.g., YOLOv8 for object detection, MiDaS for depth estimation) from the available options outlined in Section 3.3. This randomness ensures a diverse dataset for benchmarking, capturing performance variations across different configurations. We provide end users with the script and input data, requiring them only to execute a batch file (`.bat`) to initiate the process.

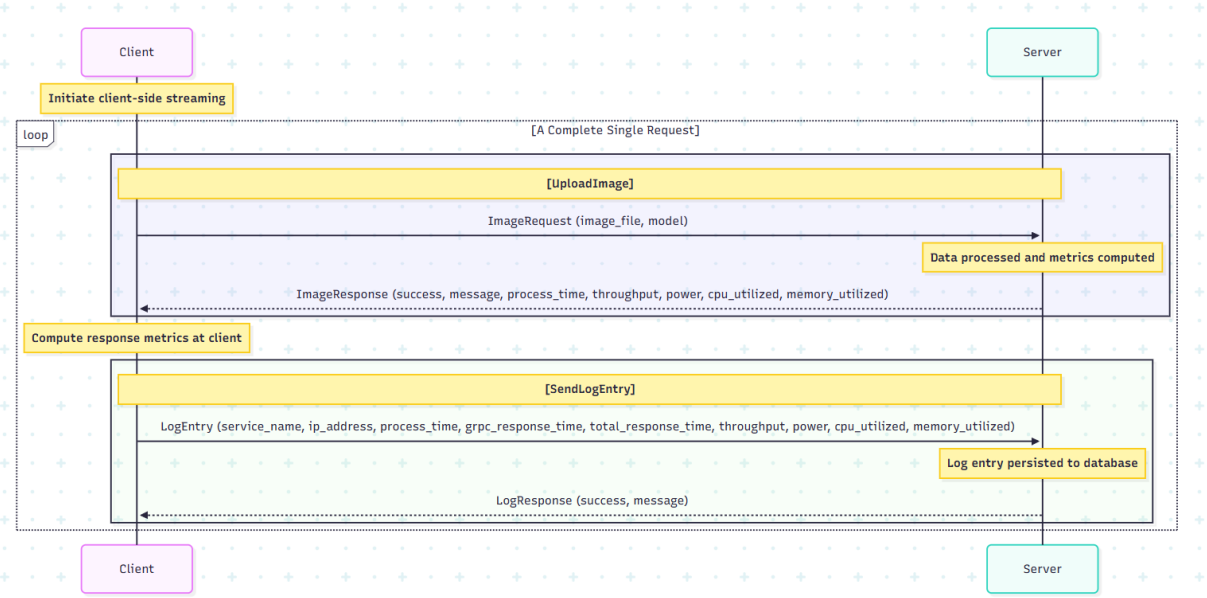

Figure 4.2: Request-response flow between client and server for a stateless gRPC system

The script automatically performs the following tasks: (1) creates a virtual environment (`venv`) to manage dependencies, (2) selects random services and models, (3) runs the selected service on the provided data (e.g., video frames, audio clips), and (4) cleans up the virtual environment upon completion. This automated workflow simplifies user interaction, ensures consistency in data collection, and supports the evaluation of AV services under varied conditions within our benchmarking framework.

Chapter 5

Data Collection Outcomes and Dataset Structure

This chapter presents the outcomes of our data collection pipeline, focusing on the metrics and tables generated to evaluate the performance of autonomous vehicle (AV) perception services within our benchmarking framework. Building on the gRPC workflow described in Chapter 4, we collect a comprehensive set of parameters to assess Quality of Service (QoS) and system efficiency across various AV tasks, such as object detection, tracking, and audio segmentation. These parameters are organized into structured tables to facilitate analysis and comparison, providing insights into latency-accuracy tradeoffs and resource utilization. The chapter is structured as first we explore the details of the parameters collected and the organization of the dataset into three primary tables—user data, service data, and server data— and moving on we describes the server-specific, service-specific, and user specific metrics recorded for benchmarking purposes.

5.1 Parameters in the Dataset

This section outlines the parameters collected as part of our data collection pipeline, which are essential for evaluating the performance of AV perception services. We primarily collect the

Quality of Service (QoS) parameters defined in Section 3.4, including latency, response time, CPU usage, memory usage, throughput, and location-based metrics. To organize these parameters systematically, we have structured the dataset into three tables: the user data table, the service data table, and the server data table. Each table contains parameters specific to its respective domain, enabling a comprehensive analysis of system performance across different components of the AV benchmarking framework. The following subsections detail the parameters recorded in each table.

5.1.1 User Data

The user data table captures geo-location and network-related parameters for the client, providing insights into the user's location and its impact on AV service performance. We record the following parameters: IP address, latitude, longitude, city, region, country, Autonomous System Number (ASN), ASN description, subnet, and subnet mask. These parameters are obtained using tools such as `ip-api` for geo-location data and `ipwhois` for ASN information, as described in Section 3.5. This data enables the analysis of location-based effects on latency and throughput, supporting the optimization of server placement strategies for real-world AV scenarios.

5.1.2 Service Data

The service data table records performance metrics for each AV perception service, enabling the evaluation of efficiency and resource usage. We capture the following parameters: IP address, model ID, model name, service type, latency time, CPU usage, memory usage, throughput, energy required, power consumption (in watts), and response time. These metrics, aligned with the QoS parameters defined in Section 3.4, are collected during service execution, as described in Section 4.3. This data facilitates the analysis of service-specific performance, supporting the optimization of latency-accuracy tradeoffs and energy efficiency for AV applications.

5.1.3 Server Data

This section describes the parameters recorded in the server data table, which capture the server's geo-location and network characteristics to evaluate its impact on AV service performance. The table includes the following parameters: public IP address, local IP address, latitude, longitude, service provider, city, region, country, geo-location coordinates, Autonomous System Number (ASN), ASN description, subnet, and subnet mask. These parameters, collected using tools such as `ip-api` and `ipwhois` as described in Section 3.5, mirror the user data table but focus on the server's context. This data enables the analysis of server location effects on latency and throughput, providing insights into optimal server placement for efficient AV operations.

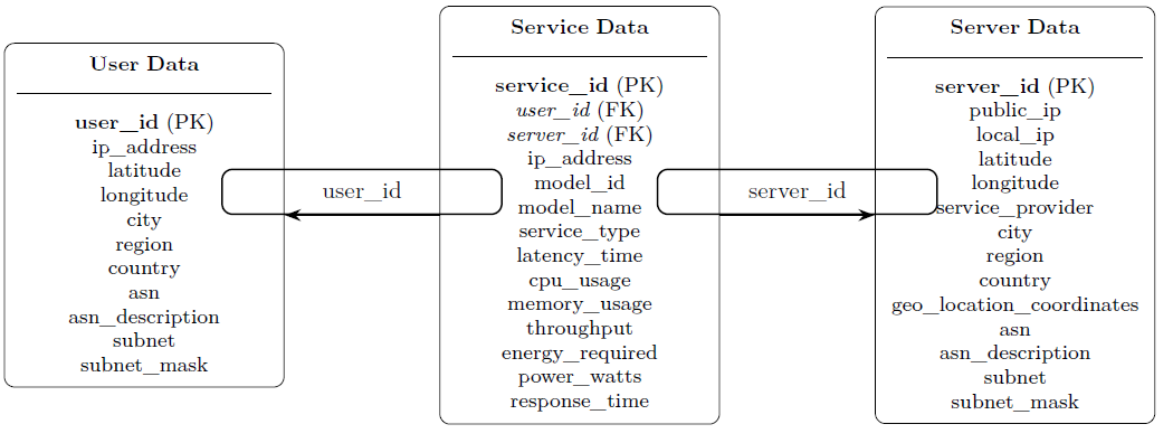

Figure 5.1: Entity-Relationship Diagram (ERD) illustrating the structure of the *User Data*, *Service Data*, and *Server Data* tables, and their relationships via *user_id* and *server_id*. Primary keys (PK) are shown in **bold**, and foreign keys (FK) are in *italics*.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This work addressed a critical gap in Autonomous Vehicle (AV) system evaluation by introducing a novel benchmarking dataset that integrates real-world network dynamics, distributed architectures, and Quality of Service (QoS)-focused evaluation. A review of existing datasets and frameworks, including KITTI [3], Cityscapes [4], WSDream [5], and Pylot [1], revealed their limitations in assessing AV performance under network-dependent conditions. KITTI and Cityscapes lack QoS metrics like latency and throughput, WSDream is not tailored for AVs, and Pylot, within the CARLA simulator [6], excels in simulation but cannot evaluate diverse algorithm sets under realistic network constraints.

To overcome these gaps, we created a dataset capturing QoS metrics—latency, throughput, resource consumption, and accuracy—across various AV algorithms and scenarios, using a gRPC-based data pipeline to emulate edge-cloud interactions for data collection. By extending Pylot's capabilities with network-aware evaluation, this work highlights QoS's role in ensuring AV safety, reliability, and scalability. Our framework enables researchers to identify bottlenecks and optimize algorithms, paving the way for safer and more scalable AV technologies in urban environments.

6.2 Future Work

While this thesis has made significant strides in evaluating QoS for AV systems, several avenues for future research remain unexplored, offering opportunities to further enhance the robustness and applicability of our framework. One promising direction is the development of predictive models for QoS parameters, such as latency, throughput, and accuracy, using machine learning techniques. By leveraging historical QoS data collected from our framework, predictive models could forecast these parameters under varying network conditions and algorithm configurations. This approach would build on the dataset introduced in this thesis, providing a foundation for training and validating predictive models tailored to AV-specific scenarios.

Another potential area of exploration is the creation of a recommendation system for AV algorithm selection and configuration based on QoS requirements. Such a system could analyze the QoS metrics captured by our framework and recommend the optimal set of algorithms for a given scenario, balancing tradeoffs between latency, accuracy, and resource consumption. By integrating this system into our framework, researchers and developers could automate the process of algorithm selection, improving the efficiency and adaptability of AV systems in diverse operational contexts.

Finally, extending our framework to support real-time QoS monitoring and adaptation in deployed AV systems could bridge the gap between research and practical implementation. By developing mechanisms for continuous QoS evaluation and dynamic reconfiguration—such as adjusting algorithm parameters or redistributing computational tasks between edge and cloud in response to network conditions—our framework could enable AVs to maintain optimal performance in real-world settings. This would involve integrating our benchmarking tools with on-board AV systems, allowing for live monitoring of QoS metrics and immediate corrective actions to ensure safety and efficiency. These future directions collectively aim to build on the foundation established in this thesis, advancing the field of AV benchmarking and contributing to the development of safer, more reliable autonomous driving technologies.

Bibliography

[1] I. Gog, S. Kalra, P. Schafhalter, and J. E. Gonzalez, "Pylot: A modular and scalable framework for autonomous driving research," in *Proceedings of the 2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1–8.

[2] Google Inc., "grpc: A high-performance, open-source universal rpc framework," https://grpc.io, 2023, accessed: May 14, 2025.

[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 3354–3361.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 3213–3223.

[5] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Ws-dream: A distributed reliability assessment mechanism for web services," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2010, pp. 1–10.

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

[7] J. Redmon and A. Farhadi, "Yolo - you look only once," *arXiv preprint arXiv:1804.02767*, 2018, extended by Ultralytics for YOLOv8 and YOLOv11, available at https://docs.ultralytics.com.

[8] TorchVision Team, "Torchvision: Pytorch's computer vision library," https://pytorch.org/vision/stable/index.html, 2023, accessed: May 14, 2025.

[9] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[10] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[11] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[12] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "Strongsort: Make deepsort great again," *IEEE Transactions on Multimedia*, vol. 25, pp. 8725–8737, 2023.

[13] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2022, pp. 1–17.

[14] Tryolabs Team, "Norfair: Lightweight python library for object tracking," https://github.com/tryolabs/norfair, 2022, accessed: May 14, 2025.

[15] D. B. Allan, T. A. Caswell, and N. C. Keim, "Trackpy: Fast, flexible particle-tracking toolkit in python," *Soft Matter*, vol. 12, no. 46, pp. 9568–9580, 2016.

[16] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric sort: Rethinking sort for robust multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023, pp. 9686–9696.

[17] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, 2022.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[19] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.