Masterarbeit

Addressing the Fundamental Barriers towards End-to-End Driving in Simulation

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Long Nguyen, long.nguyen@student.uni-tuebingen.de, 2025

Bearbeitungszeitraum: 01.05.2025-31.10.2025

Betreuer/Gutachter: Prof. Dr. Andreas Geiger, Universität Tübingen
Zweitgutachter: Prof. Dr. Georg Martius, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Long Nguyen (Matrikelnummer 5709676), December 27, 2025

Abstract

Despite remarkable progress in deep learning for autonomous driving, the open-source development of end-to-end (E2E) driving systems in simulation remains constrained by several overlooked barriers. This thesis identifies and addresses three of the most critical ones: overly omniscient rule-based expert supervision, incomplete navigation conditioning, and rigid evaluation protocols.

We redesign the E2E driving stack, from expert policy to dataset and model training, around the TransFuser++ baseline. Our contributions include a new sensor-aware expert aligned with the student's perceptual limitations, a curated dataset covering 73 hours of diverse driving across all public CARLA towns, and an improved architecture that removes shortcut conditioning on the target point whose inductive bias previously led to the so called target-point bias. Lastly, we also conduct an investigation into how to integrate radar sensing into our model baseline.

Together, these improvements substantially mitigate the target-point bias while preserving strong driving performance. Beyond quantitative gains, our analysis reveals that many long-route failures stem from weak navigation signals and unrealistic route design rather than model capacity, thus highlighting the need for improved evaluation protocols and post-training policy refinement in future work.

We release code and data to the community to accelerate progress in simulation-based E2E driving. The system achieved state-of-the-art performance on all publicly available CARLA closed-loop benchmarks, reaching 95 DS on Bench2Drive, 62 DS on Longest6 v2, and 5DS on the Town13 validation routes.

Acknowledgments

I would like to express my deepest gratitude to Prof. Andreas Geiger for giving me the opportunity to conduct my Master's thesis in his group. I am equally grateful to Kashyap Chitta, whose guidance, technical insight, and patience shaped every stage of this project. I have learned immensely from his curiosity, goal-driven mentality, and constant pursuit of deeper understanding - qualities that shaped not only this thesis but also my own approach to research.

A special thank you to Bernhard Jaeger for our weekly discussions and his constructive feedback on my ideas, which helped me refine my thinking and push this work forward. I would also like to thank my Waymo teammate, Micha, for his constant helpfulness and collaboration during the project, as well as Prof. Georg Martius for kindly agreeing to serve as the second examiner of this thesis. I am also grateful to Simon for assisting with the evaluation of HiP-AD, as well as to Daniel for his help in setting up NavSim.

On a personal note, I would like to thank my cousin Trang for always lending me an ear and offering her honest advice whenever I needed it. Her support has meant a lot throughout this journey.

And to Lara: thank you for your unconditional love, patience, and support.

Contents

1 Introduction

Autonomous driving has the potential to address some of the most pressing challenges in the transportation domain. Every year, more than one million people die in traffic accidents, the majority caused by human error [28]. Automated vehicles could drastically reduce the number of fatalities and enhance mobility for individuals unable to operate a vehicle independently. Deep learning has become the dominant paradigm for autonomous driving due to its conceptual simplicity, scalability, and the rapid progress of modern hardware. In particular, end-to-end (E2E) learning, in which a network is trained to map raw observations directly to actions, eliminates hand-engineered mid-level abstractions and optimizes the driving task holistically. However, the current development of E2E driving models within the open-source and academic research community faces several fundamental issues that hinder progress.

Recent research in development and evaluation of E2E driving models has progressed in two main directions:

- real-world evaluation [14, 8, 42], often leveraging 3D reconstruction and generative sensor simulation
- high-fidelity simulation environments, such as CARLA [15]

Real-world evaluation typically focuses on short-horizon reactive behavior predictions, while high-fidelity simulation environments enable true closed-loop evaluation over long routes with interactive traffic. Under optimal conditions, a competent driving model should behave reliably in both real-world replay settings and simulations, demonstrating strong generalization across domains and the ability to reason about long-term interactions.

As for evaluation in simulation, collecting high-quality human driving demonstrations in simulation is costly and yields limited real-world value, whereas companies have a stronger incentive to gather real-world data since it directly improves deployed systems and product reliability. Simulation-based development therefore faces several open challenges, ranging from brittle rule-based expert supervision that relies heavily on privileged state information to insufficient navigation conditioning and overly strict evaluation protocol, which will be examined in more detail in later chapters. As a result, many failures observed in long-horizon evaluation stem not from model capacity, but from a lack of high-quality supervision and incomplete conditioning that prevents the policy from expressing its full potential.

For E2E driving to evolve from research prototypes to Level-4 autonomy (driverless autonomy in restricted domains [33]) and beyond, long-route evaluation must become the norm rather than the exception. To ensure high-impact long-route evaluation, models must have access to improved driving demonstrations, and evaluation protocols must be equipped with more intelligent metrics, in addition to robust navigation conditioning that supports complex road topology and driving routes.

In this case, **navigation** refers to the ability to follow a *global* route and make correct *high-level* decisions. For example, in Figure 1.1, a good navigation system tells ego to take the highway exit timely - decisions that depend more on structured map information and route planning than on raw perception. These behaviors can be described by hand-engineered algorithms when explicit map and route information are available, and they are notoriously difficult to learn from raw sensor data alone, especially in the context of academic research.

A robust navigation system should do the hard-to-learn but simple-to-engineer part of the task - precise routing - allowing learning-based models to focus on what they do best: recognize patterns, understand context, and make nuanced decisions. While meaningful progress has been made despite imperfect navigation support, a clear separation of concerns - where robust navigation handles routing and learning-based models focus on perception and decision-making - would accelerate advances in E2E driving even further.

Given the considerable engineering effort required to build robust navigation systems, the E2E driving community should instead focus on improving **evaluation protocols and metrics**. In cases like the one shown in Figure 1.1, missing the highway exit may be an entirely reasonable outcome, especially when visual cues are ambiguous and the maneuver requires abrupt lane changes. Yet, current evaluation schemes treat this as a full failure and terminate the episode immediately. A more balanced protocol should distinguish between minor navigational misses and truly unsafe behavior, ensuring that models are not overly penalized for outcomes that stem from imperfect map design rather than poor driving ability.

The key **contribution** of this project is to provide the research community with an improved E2E driving stack, including data collection, model training, and closed-loop driving evaluation. In the pursuit of this goal, we identified multiple pressing issues that hinder current research progress in this highly exciting field. We propose simple yet targeted model improvements and a comprehensive dataset that together eliminate numerous bottlenecks within the CARLA-based E2E driving community. Our main contributions are:

(1) An enhanced model that incorporates local route structure into planning conditions, helping to mitigate the previously identified target point bias [20] while still achieving high route completion.

(2) An improved expert policy tailored to a sensory driving student model,

addressing the current lack of high-quality driving data in CARLA.

(3) A modernized training pipeline and dataset of 73 driving hours, which covers more spatial locations and scenario types.

(4) Identification of certain weakness in the evaluation protocols and metrics which over-penalize safe behavior.

After supervision and conditioning were sufficiently repaired, our baseline model TransFuser++ [31, 11, 20] obtained state-of-the-art performances on all available CARLA benchmarks: Bench2Drive, Longest6 V2, and Town13 Validation [21, 11, 4].

We release the code and dataset to the community to facilitate follow-up research and hope to set a new standard for closed-loop E2E driving in CARLA.

Thesis Structure: Chapter 2 reviews relevant work. Chapter 3 outlines our preliminary results on the inaugural Waymo Vision Based E2E Driving Challenge. Chapter 4 details the modification of the model, expert, and datasets in CARLA for our main experiments. Chapter 5 presents closed-loop evaluation and ablations. Chapter 6 summarizes the findings and outlines implications for future research.

Figure 1.1: Example of a limited navigation system in CARLA. The driving model sees only the red points, the output of the navigation system. The lane change is abrupt, and the fork offers ambiguous visual cues about whether to stay on the main road or take the exit. Attempting the lane change too early or too late often results in an out-of-lane violation or collisions. Even worse, missing the exit would immediately terminate the evaluation. This highlights a limitation of current long-route evaluation protocols, where overly strict success criteria penalize reasonable driving behavior, which mostly come from insufficient navigation conditioning.

2 Related Work

This chapter presents an overview of relevant research, starting with an introduction to end-to-end self-driving and its advantages over traditional modular architectures. We discuss imitation learning as a scalable and effective method for training agents to navigate complex environments by imitating an expert. Finally, we situate IL within the context of autonomous driving, covering datasets, trends in output representations for driving and the common issue of covariate shift.

2.1 End-to-End Learning for Autonomous Driving

Autonomous driving systems have traditionally been implemented as modular pipelines, where perception, prediction, planning, and control are separated into distinct components connected by hand-designed interfaces [10]. Each module is developed and optimized independently: perception produces discrete object detections and semantic maps, prediction forecasts agent trajectories, planning generates motion primitives or spatiotemporal paths, and control executes low-level commands. This modularity provides interpretability, enables module-level debugging, and allows safety monitors to be inserted at interface boundaries. However, the separation introduces several limitations.

Errors propagate and compound across module boundaries, as each downstream component must operate on potentially noisy or incomplete outputs from upstream modules. To optimize the driving objective holistically, end-to-end (E2E) methods learn a direct mapping from raw sensor observations to control commands or short-horizon trajectory plans [7, 13]. Rather than decomposing the task into perception, prediction, and planning stages with discrete intermediate representations, E2E models process sensor inputs through a single learned policy that outputs either steering and throttle commands or a sequence of waypoints.

The primary advantage of end-to-end learning is holistic optimization: the model is trained jointly across all components, without introducing hand-crafted intermediate representations or post-processing steps that may discard information, as is often the case with rule-based hard decision layers. However, E2E methods introduce new challenges themselves. They require high-quality supervision, as the learned policy directly inherits the biases and limitations of the expert demonstrations or reward signals.

2.2 Simulation for E2E Driving Development

The development and evaluation of E2E driving policies requires extensive closed-loop interaction with realistic driving environments, making simulation an essential tool for research and development [15, 18, 14, 8]. Physical testing is prohibitively expensive, dangerous during the early stages of development, and limited in the diversity of scenarios that can be systematically explored. Simulation addresses these limitations by offering a safe, replicable, and scalable environment in which policies can be trained and evaluated under controlled conditions.

CARLA has emerged as the dominant open-source simulator for E2E driving research [15]. The CARLA Leaderboard [4] was introduced to standardize evaluation and enable reproducible comparisons across E2E driving methods. Leaderboard 1.0 (LB1) established a fixed set of routes with predefined traffic scenarios and introduced metrics including Driving Score (DS), Route Completion (RC), and Infraction Score (IS) to quantify closed-loop performance. However, LB1 has become saturated, with multiple methods achieving near-perfect scores, revealing that the diversity and difficulty of the benchmark's scenarios are insufficient to differentiate between approaches. Leaderboard 2.0 addresses these limitations by introducing substantially longer routes (up to 12 km per route) and a richer set of traffic scenarios, including complex multi-agent interactions [4].

While simulation provides a controlled and scalable development environment, it also introduces limitations. Simulated sensor data, vehicle dynamics, and agent behaviors are approximations of real-world conditions, and policies trained in simulation often exhibit performance degradation when deployed in physical environments due to the sim-to-real gap. Nevertheless, simulation remains indispensable for early-stage development, systematic ablation studies, and large-scale benchmarking, as it enables rapid iteration and exploration of design choices that would be infeasible in physical testing.

2.3 Imitation Learning for E2E Driving

Imitation learning (IL) trains a policy to replicate expert demonstrations by minimizing the difference between predicted and observed actions [30, 7]. Early work demonstrated camera-to-steering mappings on simple tasks, such as lane following, using relatively shallow networks. Recent IL approaches in simulation have achieved substantially higher performance by scaling model capacity, incorporating multi-modal sensor fusion, and introducing auxiliary perception losses to stabilize training [11, 31, 35, 36, 32].

The primary advantage of IL is training stability and sample efficiency when expert demonstrations are of high quality. The supervised objective is straightforward, training converges reliably, and the policy can achieve competent performance

without requiring extensive reward shaping or safety constraints during exploration. However, IL suffers from several well-documented limitations. First, the learned policy inherits all biases present in the expert demonstrations, including behaviors that rely on privileged information unavailable to the sensory student [44]. Second, IL is vulnerable to covariate shift: when the policy deviates from the expert's state distribution during deployment, it encounters situations absent from the training data and may fail catastrophically [34].

In the context of our work, we opt for behavior cloning, a subfield of IL that reduces the learning to supervised learning, which allows us to leverage the benefits of a simple training approach at a large model and data scale.

2.4 Expert Supervision and Model Conditioning

Expert supervision and model conditioning are two critical factors that influence the performance of IL policies for end-to-end autonomous driving.

Regarding the driving expert, Jaeger [19] shows that the quality of demonstrations is a critical bottleneck for the performance of IL in CARLA. By upgrading the default rule-based driving expert of CARLA to leverage more privileged information to avoid crashes, including precise agent positions, velocities, and future intentions, Jaeger improved the expert's performance and provided evidence that downstream student performance also improved substantially. This establishes that demonstration quality, rather than model architecture alone, can be a limiting factor in IL performance.

Zimmerlin [44] extends this analysis by documenting specific expert behaviors in CARLA, showing that naively making the expert better does not always yield high-quality demonstrations. For example, the expert may slow down preemptively for pedestrians crossing the road while they are mostly invisible or heavily occluded in the camera and LiDAR streams, acting on privileged knowledge of their precise position and trajectory that the sensory student cannot observe. When these demonstrations are used for training, the student learns to replicate the action pattern but lacks the causal trigger, resulting in brittle policies. Zimmerlin shows that correcting the expert to act based only on observable cues improves closed-loop driving performance on Leaderboard 2.0.

As for navigation conditioning, Codevilla et al. [13] introduced conditional IL, demonstrating that explicit navigation conditioning is essential for E2E driving policies to execute directional maneuvers correctly. Without conditioning on high-level commands (e.g., turn left, turn right, go straight), the policy observes identical visual inputs at intersections but must produce different actions depending on the intended route, leading to ambiguous training signals and failures at evaluation time.

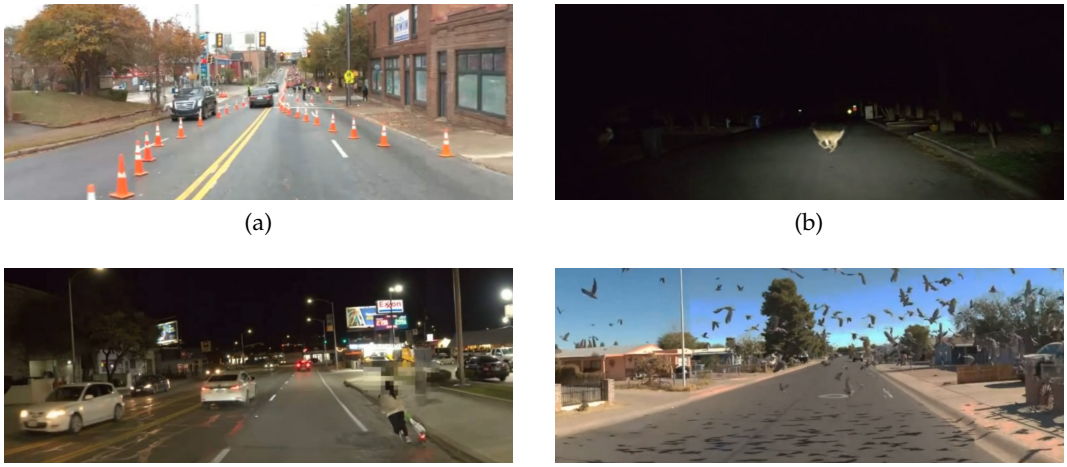
Jaeger et al. [20] further identified that E2E models can exhibit over-reliance on

navigation conditioning signals when perceptual representations are insufficiently expressive. In scenarios where the bird's-eye view (BEV) feature representation does not adequately capture local scene geometry or agent interactions, the model compensates by attending disproportionately to the target point, effectively using it as a substitute for missing scene understanding rather than as a navigational guide.

In our work, we extend both directions by tailoring the expert to the sensory limitations of the student and by systematically exploring the trade-off between navigation conditioning and scene understanding, pushing the Pareto frontier further.

3 Preliminary Results on the Waymo Open Dataset

The Waymo Open Dataset End-to-End Driving Challenge [41] provided a controlled benchmark for evaluating vision-based E2E policies in the context of rare, high-impact real-world events. The challenge isolated long-tail situations and evaluated models in an open-loop waypoint-prediction setting using 360-degree camera inputs, ego history, and routing signals. Although it operates in an open-loop rather than a closed-loop setting, the challenge offered a unique opportunity to stress-test models on infrequent but safety-critical events that typical open-loop benchmarks do not expose. Figure 3.1 depicts selected examples of the dataset.



(a) (b)

(c) (d)

Figure 3.1: Waymo E2E Demo Scenarios demonstrating the long tail and reactive nature of the benchmark.

This challenge was particularly informative, not for measuring driving competence, but for revealing what current open-loop benchmarks truly test. They assess reactive alignment with a logged trajectory under a rich sensory context, which is an important building block that needs to be considered when transferring policy from simulation to the real world.

Although this thesis primarily focuses on closed-loop driving in CARLA, we also participated in the Waymo E2E challenge as an educational detour—to better under-

stand how TransFuser++ behaves under curated, real-world long-tail conditions without diving into the complexities of closed-loop control. This exercise proved valuable not for leaderboard performance, but for deepening our understanding of how data curation, scenario design, and supervision shape model behavior. In that sense, the experiment served as a diagnostic tool: it helped us analyze the generalization and limitations of TransFuser++ in an open-loop setting, complementing the closed-loop investigations that form the core of our work.

3.1 Challenge Overview

The challenge evaluated predicted trajectories using the Rater Feedback Score (RFS), as depicted in Figure 3.2. RFS evaluates predictions against human expert judgment using a 0-10 scale. Expert raters define trust regions around acceptable driving behavior using lateral and longitudinal thresholds at timestamps T=3s and T=5s. To receive an expert's score, predicted trajectories must fall within both trust regions; otherwise, they receive exponentially decreasing penalties based on distance to the nearest reference. As a secondary metric, we report Final Displacement Error (FDE), measured as the L2 distance between predicted and ground truth trajectory endpoints at T=5s.

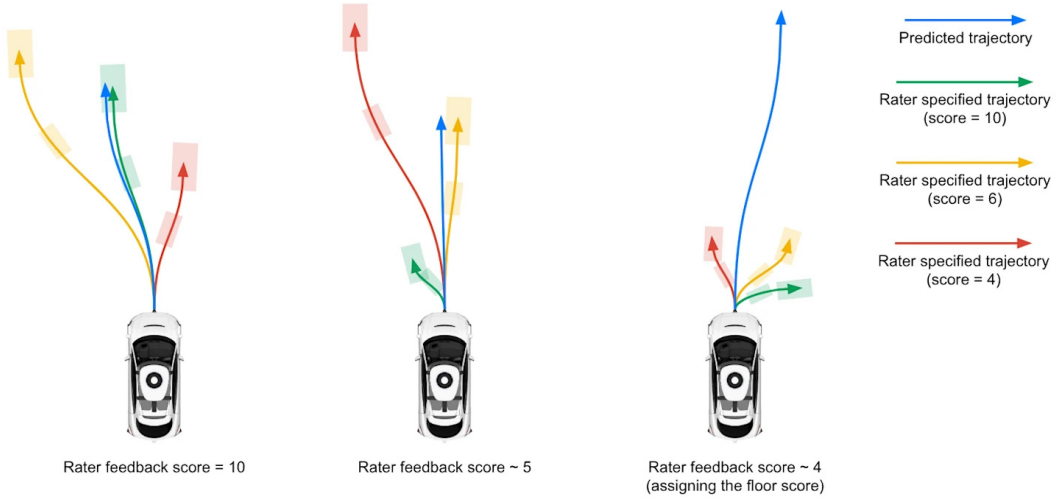


Figure 3.2: Rater Feedback Score (RFS) Metric.

3.2 Our Approach

Our approach adopted a two-stage training workflow that leveraged the distinct strengths of multiple, diverse data sources. In the first stage, we performed perception pre-training on large-scale datasets with rich auxiliary labels (CARLA [15], NAVSIM

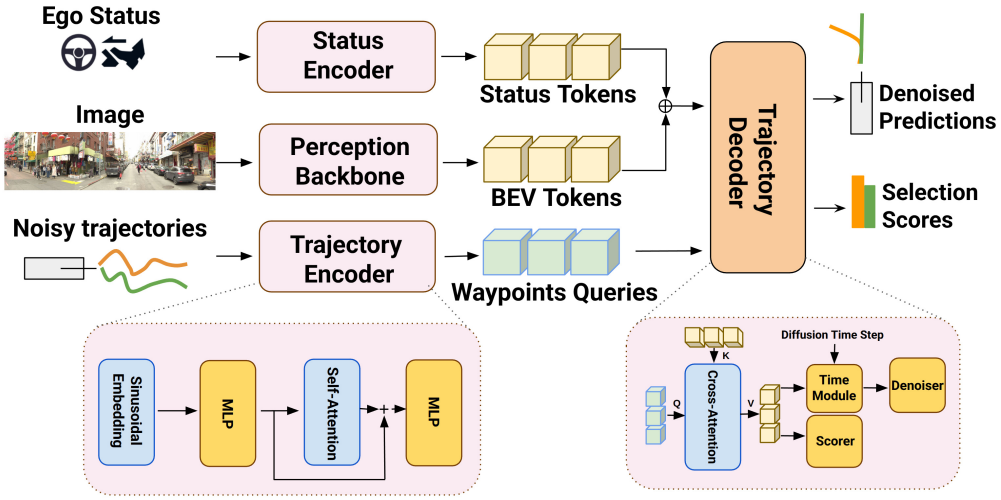


Figure 3.3: Adaption of TF++ [20] for Waymo E2E Driving Challenge 2025.

[14], WOD-P [16]) to build a strong representational foundation. In the second stage, we conducted post-training on the curated planning-centric WOD-E2E data, which exposed the model to challenging and rarely observed driving scenarios.

We employed a Latent TransFuser (LTF) architecture, see Figure 3.3, with a ResNet34 backbone for vision-only E2E driving, extending it with a diffusion-based trajectory generation head (DiffusionLTF) to handle multimodal trajectory distributions. The diffusion decoder used a discrete vocabulary of representative driving patterns derived from training data, resulting in the generation of diverse trajectory proposals. For final predictions, we implemented a hybrid ensemble approach: by drawing 10 trajectory proposals from DiffusionLTF, grouping them by mode, averaging within groups, and combining the highest-scoring mode with the deterministic LTF output.

3.3 Final Results

Table 3.1 presents our dataset mixture analysis on the WOD-E2E validation set. Our experiments revealed that pre-training with auxiliary datasets, even those synthetic from CARLA, consistently improves model performance compared to training solely on WOD-E2E data.

Among individual datasets, the Waymo Open Dataset-Perception split (WOD-P) emerges as the most effective single pre-training source, achieving the highest RFS score of 7.85.

Interestingly, post-training exhibits a contrasting pattern. Incorporating additional datasets during post-training generally degraded performance compared to training exclusively on WOD-E2E data, with the baseline achieving the best RFS score of

Pre-training Datasets			Metrics	
CARLA	NAVSIM	WOD-P	FDE ↓	RFS ↑
			6.33 ± 0.46	7.52 ± 0.01
✓			6.02 ± 0.14	7.75 ± 0.02
	✓		5.94 ± 0.16	7.81 ± 0.03
		✓	5.82 ± 0.10	**7.85 ± 0.04**
✓	✓		5.74 ± 0.17	7.83 ± 0.04
✓		✓	5.84 ± 0.20	7.74 ± 0.03
	✓	✓	5.90 ± 0.09	7.81 ± 0.04
✓	✓	✓	**5.73 ± 0.21**	7.84 ± 0.01

(a) Impact of pre-training with diverse data.

Post-training Datasets			Metrics	
CARLA	NAVSIM	WOD-P	FDE ↓	RFS ↑
			5.73 ± 0.21	**7.84** ± 0.01
✓			6.25 ± 0.19	7.76 ± 0.12
	✓		6.17 ± 0.05	7.77 ± 0.04
		✓	5.82 ± 0.21	7.83 ± 0.03
✓	✓		5.91 ± 0.20	7.81 ± 0.07
✓		✓	**5.65** ± 0.18	7.80 ± 0.05
	✓	✓	5.88 ± 0.10	7.80 ± 0.04
✓	✓	✓	5.74 ± 0.08	**7.84** ± 0.07

(b) Joint post-training perception and planning.

Table 3.1: Dataset mixture analysis on the WOD-E2E validation set.

7.84. This indicated the importance of curated planning-centric data in the fine-tuning phase, where exposure to the specific long-tail scenarios present in WOD-E2E appears more valuable than increased data diversity.

Our final test submission achieved an RFS of 7.71, demonstrating competitive performance with only a ResNet34 backbone and requiring only one day of training on a single A100 GPU.

3.4 Discussion

One of the main lessons from participating in the Waymo E2E challenge is the importance of label curation and annotation quality in large-scale perception datasets. While model architecture and training strategy often receive most of the attention, the consistency and precision of labels ultimately determine the ceiling of achievable performance and the validity of any resulting conclusions.

Another finding of the challenge shaped our research direction in CARLA. Typical E2E driving environments provide navigation signals as discrete categorical commands indicating the intended action at the current timestamp: left, right, or straight.

These commands do not encode route structure or spatial relationships to the goal. Figure 3.4 visualizes these commands as red arrows overlaid on consecutive frames from the same intersection scenario.

The top-performing method in the challenge did not even include the navigation command as input and still achieved competitive results.

This highlights a critical limitation in the current state of research in E2E driving: typical short-route evaluation tests focus on reactive capability alone, not real planning capability.

(a) (b)

(c) (d)

Figure 3.4: Temporal inconsistency in automatically generated navigation commands. Red arrows visualize the discrete command at each timestamp (straight, left, or right). All four frames show the same intersection scenario captured at consecutive timestamps, yet the commands flicker between different directions, exposing the lack of temporal coherence in the navigation signal.

The success of models that ignore the provided navigation command entirely indicate that the limitation of the benchmark lies in its weak goal specification. If models can achieve competitive results without explicitly knowing where to drive, it implies that the benchmark primarily rewards short-horizon reactive alignment rather than true goal-directed reasoning. Yet, goal-directed reasoning is precisely what enables long-route completion in realistic driving: understanding where to go, anticipating future maneuvers, and maintaining consistency over time. Without evaluating this capability, the benchmark remains biased toward short-term imitation rather than long-horizon planning, making it a poor proxy for true autonomous driving competence.

4 Methods

This chapter outlines the methods used in our approach and provides the necessary background. We start by introducing foundational concepts, practical tools and evaluation methods used to develop and evaluate our approach. In particular, we describe the functioning principle of the driving simulator, the evaluation metrics. Then we describe our adaptation to a baseline driving model to help it navigate more competently in the virtual environment. Last, we describe our adaptation of the driving expert, training dataset and training pipeline that we tailored to better support our goals.

4.1 Preliminaries

This section introduces the research and development environment we setup and the evaluation metrics we used. We also present the baseline model architecture and the driving expert we used to collect demonstrations. To close, we highlight equally important technical tools that we used along the way.

4.1.1 Imitation Learning (IL)

IL is a widely used technique for training autonomous agents to perform complex tasks by learning from expert demonstrations [3]. Expert demonstrations are recordings of a competent agent performing the desired task, consisting of sequences of observations and corresponding actions. For autonomous driving, those observations could be camera images, LiDAR point clouds, and radar readings. While actions include steering angles, throttle, and brake commands. Our autonomous driving-agent, in most cases a deep neural network, maps observations to actions. The goal of IL is to find the best student policy that mimics the expert's behavior on a fixed dataset as close as possible. In case of a deep neural network, this optimization is achieved through supervised learning, which effectively optimizes the network parameters in a differentiable manner.

For autonomous driving, it is common to predict intermediate representations such as waypoints that the vehicle should follow [20, 44] rather than low-level control commands. Those representations can then be converted to control commands using a separate controller module, for example a PID controller. This decomposition

allows the student policy to focus on high-level decision-making and planning, while the controller handles low-level actuation.

4.1.2 Research Plattform: CARLA Simulator and Leaderboard 2.0

CARLA is an open-source simulator for autonomous driving research [15]. It provides a high-fidelity virtual environment with realistic urban layouts, dynamic traffic participants, and diverse weather conditions.

CARLA Leaderboard 2.0 is a standardized closed-loop evaluation benchmark for autonomous driving agents in the CARLA simulator[15]. The protocol comes with two official benchmarks, Town13 validation routes and the main Leaderboard 2.0 test routes. Further benchmarks can be contributed by the community by designing new route sets and distributing those routes. A benchmark typically consists of pre-defined routes across multiple town environments, each route containing a sequence of navigation waypoints and scripted traffic scenarios that the agent must navigate safely and efficiently, see Figure 4.1. The new benchmark protocol is a significant upgrade over the previous Leaderboard 1.0, featuring more complex scenarios. The two official benchmarks also feature longer routes, with each individual route reaching up to 13.5 km in length.

Figure 4.1: In CARLA Leaderboard 2.0, agents are expected to follow navigation instructions (red target points) while reacting to dynamic traffic and scenario events.

The CARLA Leaderboard 2.0 evaluation protocol provides agents with a sensor budget consisting of 8 RGB cameras, 2 rotating LiDAR units, and 4 radar sensors, as well as IMU, GNSS, and a speedometer. Agents receive sensor observations at 20 Hz

and must produce control commands at the same frequency. Navigation is specified through a target point interface: the agent is provided with a sequence of sparse waypoints in world coordinates at the beginning of each route, representing key locations along the planned route such as turn points and lane change positions. The agent also receives GPS measurements of its current position in world coordinates.

The agent is expected to follow these target points while reacting appropriately to dynamic traffic and traffic rules. Beyond basic safety constraints such as collision avoidance, stopping at red lights and stop signs, further constraints penalize insufficient progress, forcing agents to maintain a minimum speed. Deviation from the planned route beyond a threshold distance results in route deviation infractions and episode termination. Routes are evaluated individually, and performance is aggregated across all routes to produce final benchmark scores. The evaluation system monitors infractions, as each contributes to a penalty that reduces the agent's overall score.

4.1.3 Development Benchmarks: Bench2Drive and Longest6 v2

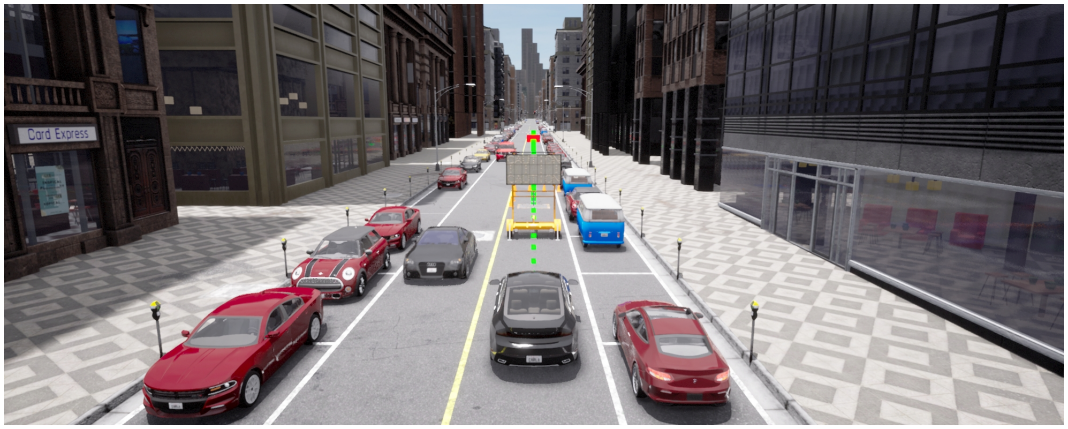
The challenging nature of CARLA Leaderboard 2.0, with its long routes and complex scenarios, makes it less suitable for rapid iteration during development. To address this, we utilize two complementary closed-loop benchmarks, Bench2Drive [21] and Longest6 v2 [11], designed for training and intermediate evaluation of autonomous driving agents in CARLA. For an overview, see Table 4.1.

Attribute	Bench2Drive	Longest6 v2	Town13	LB 2.0
# Routes	220	36	78	10
Length	50–200 m	1–2.5 km	11-13.5 km	10.295 km [44]
Towns	All publicly available	T01–06	T13	Undisclosed
Focus	Scenarios	Navigation	Validation	Generalization
Min. speed	No	Yes	Yes	Yes
Purpose	Fast Iteration	Fast Iteration	Final Validation	Ranking

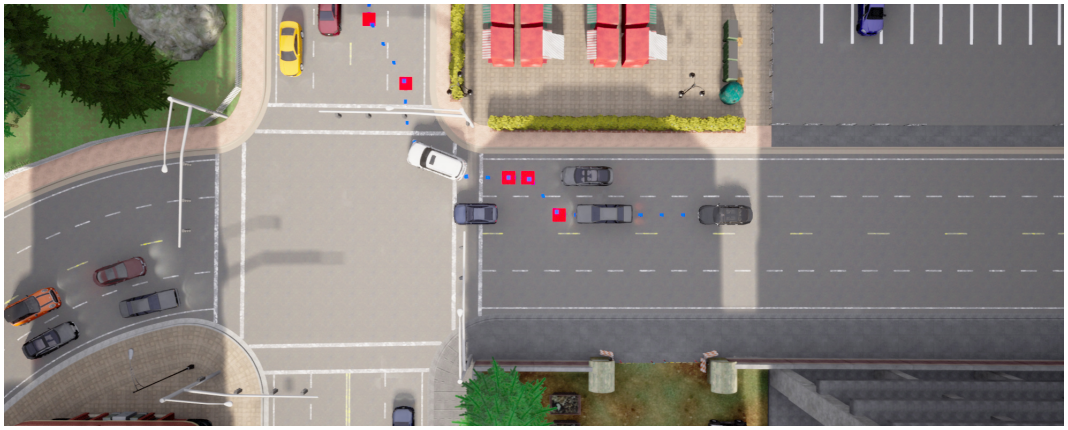
Table 4.1: Comparison of popular CARLA closed-loop benchmarks.

Bench2Drive consists of 220 short routes distributed across multiple CARLA towns. Routes are approximately 50–200 m in length and focus on specific scenario types such as unprotected turns, pedestrian crossings, and traffic light interactions. The shorter route length enables faster iteration during development, as complete benchmark runs require less simulation time than Leaderboard 2.0 evaluation. Scenarios are drawn from the same categories as Leaderboard 2.0 but with different spatial distributions.

Longest6 v2 comprises 36 medium-length routes in a total of six towns (Town01 to Town06), with each route ranging from 1.0 km to 2.5 km. The benchmark emphasizes route diversity and navigation complexity, including challenging intersection geometries, dense traffic flows, and complex route changing maneuvers. Longest6 v2 serves



(a)



(b)

Figure 4.2: **(a)** A typical Bench2Drive scenario focuses on evaluating short-horizon decision-making, sensor and perception robustness. Almost every route consists of a single hard challenge to be solved. **(b)** On the other hand, a Longest6 v2 route, with its complex route structure poses a challenge for agents to stay on the path and complete the route.

as an intermediate evaluation point between the short scenarios of Bench2Drive and the long routes of Leaderboard 2.0. For a qualitative overview, see Figure 4.2.

Both benchmarks use the same sensor setup and target point interface as Leaderboard 2.0. Bench2Drive, excluding minimum speed penalties as it consists of mostly short routes, instead sets its focus on evaluating the agent's ability to handle corner cases and safety-critical scenarios rather than navigation efficiency. Our end goal is to develop agents that perform well on Leaderboard 2.0, while the complementary benchmarks facilitate efficient development and evaluation at multiple scales.

4.1.4 Evaluation Metrics

Average Displacement Error (ADE) and **Final Displacement Error (FDE)** are standard metrics for evaluating trajectory prediction accuracy in open-loop settings. ADE measures the average distance between predicted and ground truth trajectory points over the entire prediction horizon, while FDE measures the distance at the final time step. Those two metrics are useful for debugging training pipeline, but less relevant for closed-loop driving performance, as they do not capture the agent's ability to react to dynamic traffic and maintain safety.

Closed-loop performance in CARLA benchmarks is quantified through three primary metrics: Route Completion, Infraction Score, and Driving Score.

Route Completion (RC) measures the percentage of the route distance successfully traversed before the episode terminates. An episode ends when the agent reaches the destination, collides with an object and becomes immobilized, deviates too far from the planned route, or exceeds the time limit. RC is computed as the ratio of distance traveled to total route length, expressed as a percentage between 0 and 100.

Infraction Score (IS) quantifies driving quality by penalizing safety violations and rule infractions. The score starts at 1.0 and is multiplied by a penalty coefficient for each infraction encountered during the route. For more details on infraction types and penalty values, we refer to Zimmerlin [44]. Multiple infractions of the same type on a single route compound multiplicatively, see Figure 4.3. The final IS ranges from 0 to 1, where 1 indicates a completely infraction-free drive. *Since the infraction score does not increase or decrease linearly, significance of improvements should always be treated with care. For example, an improvement from 0.1 to 0.4 requires much fewer infractions than an improvement from 0.4 to 0.7.*

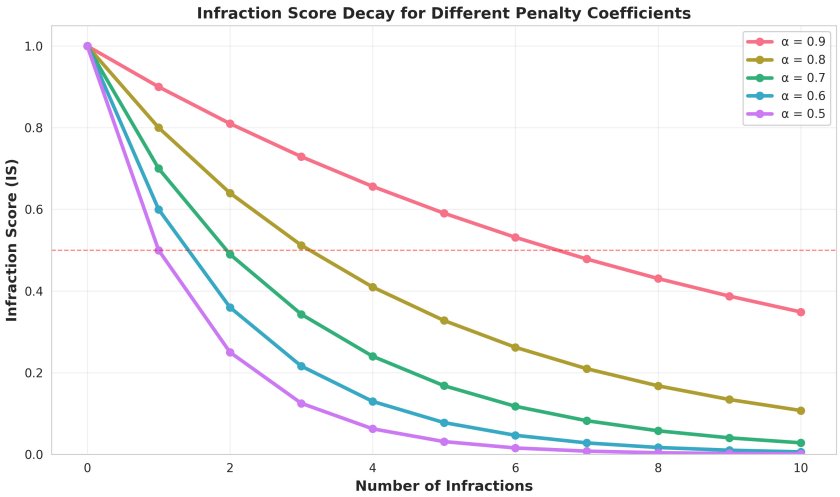


Figure 4.3: Infraction score decay for different penalty coefficients α.

Driving Score (DS) combines route completion and infraction score into a single performance metric: DS = RC × IS. This ensures that both navigation progress and driving safety contribute to the final score. An agent that completes the full route (RC = 100) with no infractions (IS = 1.0) achieves the maximum DS of 100. Conversely, an agent that drives safely but fails to make progress, or completes the route while incurring numerous infractions, receives a proportionally reduced score.

On short routes like those of Bench2Drive, another interesting metric is the **Success Rate (SR)**, which measures the percentage of routes completed without any infractions. Perfect success rate on a large number of short routes is a neccessary condition for strong performances on longer routes, as infractions tend to terminate episodes early. *On short routes, one should be cautious with models achieving high DS but low SR, which is possible if infractions happen mostly at the end of the routes.*

4.1.5 Baseline Model Architecture

TransFuser++ is a multi-modal sensor fusion architecture for end-to-end autonomous driving that serves as our baseline student model [31, 11, 20, 44]. Central to the model is a transformer-based architecture that fuses RGB camera and LiDAR sensor inputs on multiple resolutions with the self-attention mechanism. While features of each modality are extracted independently through modality-specific convolutional backbones, the transformer layers enable rich interactions between modalities. Beyond cross-sensor interactions, the self-attention mechanism also enables long-range spatial reasoning within each modality, effectively enhancing the feature extractors of each modality with global context.

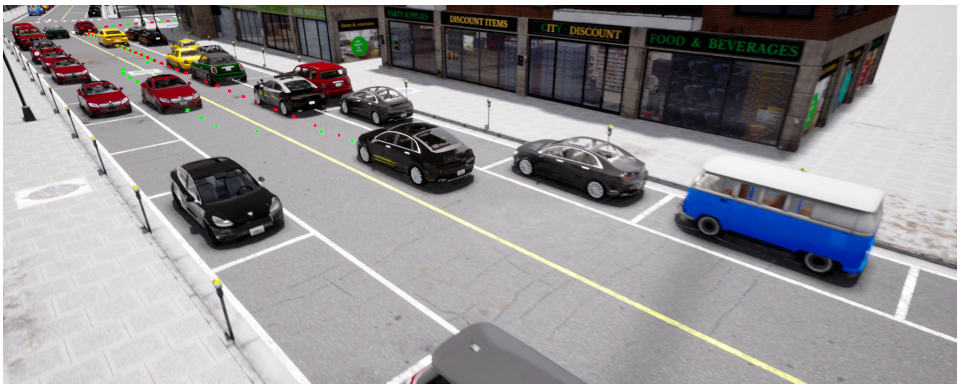
As for the planning, TransFuser++ focuses strongly on the dense BEV grid produced by the LiDAR branch. Those tokens represent the scene in a metric 2D space, which is a more suitable space for planning than the perspective image space of cameras. The BEV tokens are shaped by light-weight auxiliary perception heads, which predict 2D bounding boxes and HD-Map. This design in particular benefits from the rich pixel-level annotations available in CARLA.

The planning decoder, designed around a multi-layer transformer decoder [40], takes the BEV tokens context to predict the dense spatial route, scalar target speed and future positions. Besides scene tokens, the planner incorporates navigation conditioning through the current target point, the ego's speed, and the navigation command. The *target point* in particular is a subject of importance as it indicates the desired direction of travel and influences trajectory planning.

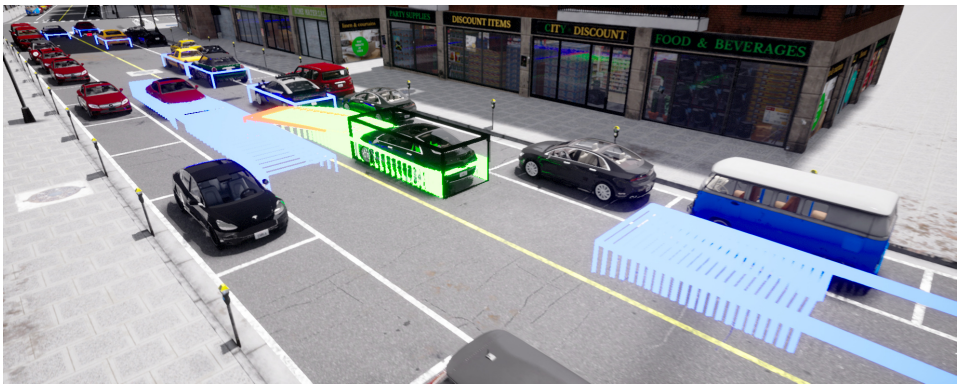
To stabilize training and avoid shortcut learning in the early stages, the model is trained in two stages. In the first stage, only the perception auxiliary heads are trained to establish strong scene representations. In the second stage, both perception and planning losses are jointly optimized.

4.1.6 Driving Expert for Collecting Demonstrations

IL requires expert demonstrations to supervise the student model. Collecting those demonstrations from human drivers is expensive and time-consuming; instead, we leveraged a rule-based expert planner that operates with privileged access to the simulator state [6]. The expert maintains a global representation of the environment including precise positions, velocities, and semantic classes of all agents in the scene, as well as complete map topology and traffic light states. Note that this privileged information is unavailable to sensor-based agents. A visualization of the expert's internal working mechanism can be seen in Figure 4.4.



(a)



(b)

Figure 4.4: Driving expert in action. **(a)** In this situation, the program detects an accident site and performs a lane change maneuver around the obstacle (green points). **(b)** The initial target speed proposal considers the speed limit and the Intelligent Driver Model. Further heuristics, such as bounding box collision prediction, reduce the target speed in case of a predicted collision. Final driving commands are produced by two PID controllers, one for steering along the planned route and one to throttle/brake to meet the target speed. [6]

4.1.7 Doppler Radar Sensors

Radar sensors emit electromagnetic waves and measure the reflections from surrounding objects to estimate their distance and relative motion. Unlike cameras or LiDAR sensors, which mainly provide spatial or visual information, radar also directly measures the radial velocity of moving objects via the Doppler effect, making it a valuable sensor for autonomous driving [27]. An overview and comparision of the three sensors can be found in Table 4.2.

	Cost	Noise Robustn.	Range	Resolution	Weather Robustn.	Velocity
Camera	✓	✓	●	✓	✗	✗
LiDAR	✗	✗	✓	●	✗	●
Radar	✓	✓	✓	✗	✓	✓

Table 4.2: Comparison of sensor types, Camera, LiDAR, and Radar, across various attributes. Green checkmarks indicate favorable traits, yellow circles indicate moderate traits, and red crosses indicate unfavorable traits.

In CARLA, radar is implemented as a LiDAR-like sensor with raycast for collision detection and additionally reports the relative velocity of each detection with respect to the ego vehicle. Each radar unit produces up to 75 detections per frame and the agent is allowed to utilize up to 4 radar units.

4.1.8 Camera Model

A camera model describes how 3D points in the world are projected onto a 2D image plane. A particularly popular model is the pin-hole camera model, which describes how light rays from objects in the 3D scene pass through a single point and create an image on the sensor plane. With (X, Y, Z) as the 3D world coordinates, (u, v) as the image coordinates, $\mathbf{K} \in \mathcal{R}^{3\times3}$ as the intrinsic matrix, and $\mathbf{R} \in \mathcal{R}^{3\times3}, \mathbf{t} \in \mathcal{R}^{3\times1}$ as the extrinsic rotation and translation of the camera, the projection from 3D world points to 2D image points can be expressed as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{4.1}$$

It is important to note, that this projection loses depth information. This means that multiple 3D points along the same ray from the camera will project to the same pixel location. As a result, only the nearest point along each ray is visible in the image. More importantly, if we know the depth Z of a pixel, we can reverse this process and recover the 3D positions of points from their 2D image coordinates as following

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = Z\,\mathbf{R}^{\top}\mathbf{K}^{-1}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \mathbf{R}\mathbf{t}, \tag{4.2}$$

For our work, the pin-hole camera model is of particular significance to equip the expert with better scene understanding and more accurate label generation. Given the intrinsic and extrinsic parameters of the camera, we can unproject the 2D image points with depth information and obtain their 3D coordinates in the world frame. The obtained camera point cloud is particularly useful for occlusion checks and generating accurate semantic labels for training data.

4.1.9 Kinematic Bicycle Model

We use the standard kinematic bicycle model with state x, control input u, and wheelbase length L:

$$x = \begin{bmatrix} X \\ Y \\ \psi \\ v \end{bmatrix}, \qquad u = \delta, \qquad L > 0 \tag{4.3}$$

In a time step k, given current state X_k, control input δ_k, and time step duration Δt, the next state X_{k+1} of the vehicle can be predicted as:

$$X_{k+1} = X_k + v\cos\psi_k\,\Delta t, \qquad Y_{k+1} = Y_k + v\sin\psi_k\,\Delta t, \qquad \psi_{k+1} = \psi_k + \frac{v}{L}\tan\delta_k\,\Delta t \tag{4.4}$$

As for v_{k+1}, we fit a simple function based on current speed and control input to approximate CARLA vehicle dynamics. For our expert, the kinematic bicycle model is used for trajectory rollout and collision checking. Given another vehicle's current position, velocity, heading and control inputs, we can predict its future trajectory and check for potential collisions with the ego vehicle's planned path.

4.1.10 Navigation and Motion Planning in CARLA

As for navigation, PDM-Lite utilizes A*, a graph-based path planning algorithm that finds the shortest path between target points on a road network. The algorithm requires access to a complete map representation. This privileged map information is unavailable to the sensor-based student model, that has to infer navigation decisions solely from sensors and sparse target points.

Besides navigation, the dense A* output trajectory is also used for local planning and motion control by PDM-Lite. While A* is optimal in sense of distances, it does not consider dynamic traffic, vehicle dynamic constraints, or other real-world factors. Paths produced by A* can produce uncomfortable driving behavior when changing lanes. For highway exit situations, the A* path can sometimes be impossible to follow due to sharp turns, current speed and current traffic. For the sake of simplicity, and because our main goal is to minimize collisions and not maximize comfort, the A* path is generally good enough for both navigation and motion control.

4.1.11 Relevant Neural Network Building Blocks

Transformer Decoder: A transformer decoder [40] consists of multiple stacked layers, each containing three main components: a self-attention block, a cross-attention block, and a feed-forward network (FFN). In TransFuser++, the transformer decoder processes a set of learned queries, where each query corresponds to one future trajectory point. Through multiple steps, these queries iteratively refine their representation by incorporating information from the previous queries and spatial cues.

The *self-attention* mechanism allows query tokens (output trajectory points) to exchange information among themselves. Given a set of N input tokens $X = [x_1, \ldots, x_N] \in \mathbb{R}^{N \times d}$, self-attention computes three projections:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{4.5}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_h}$ are learnable matrices for queries, keys, and values. The output of the self-attention layer is the weighted sum of the value vectors:

$$\mathrm{SA}(X) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V \tag{4.6}$$

This formulation allows the queries to interact with each other. In the subsequent *cross-attention* layer, the query tokens Q interact with context features C (e.g., BEV embeddings, target point, ego speed),

$$Q = XW_Q, \quad K = CW_K, \quad V = CW_V \tag{4.7}$$

The combination of self- and cross-attention thus supports both internal reasoning and external grounding. Finally, the *feed-forward network* (FFN) applies non-linear transformations to each query independently.

Gated Recurrent Unit (GRU): A GRU is a type of recurrent neural network that processes sequential data by maintaining a hidden state that is updated at each point [29, 12]. To be concrete, h_t is the hidden state at trajectory point t, x_t is the input at point t, $h_{(t-1)}$ is the hidden state of the previous point, r_t, z_t, and n_t are the reset, update, and new gates, respectively, and $\odot$ is the Hadamard product. For each element x_t in the input sequence, the GRU computes the following functions:

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}) \tag{4.8}$$
$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}) \tag{4.9}$$
$$n_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn})) \tag{4.10}$$
$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{(t-1)} \tag{4.11}$$

The hidden state h_t is also treated as output of the GRU for the input x_t. In the original architecture of TransFuser++, the GRU is used on top of the planning transformer decoder to process the queries outputs before transforming them into trajectory points. To be concrete, we encode the current target point with a linear layer and take the embedding as the initial hidden state. The inputs are learned queries coming from the transformer decoder. The outputs of the GRU are linearly transformed into trajectory offsets and the final trajectories are obtained by cumulatively summing those offsets.

Since the connection between the target point input and the first output points of the trajectory is very shallow in the GRU, changing the magnitude and position of the target point most likely will directly affect the scale and direction of the output offsets.

Detection Transformer (DETR): DETR [9] is an object detection architecture that uses learned queries to predict objects in an image. Assuming a maximal number of objects in any picture, DETR uses a fixed set of learned queries that are passed through a transformer decoder to predict object properties. The decoder outputs predictions for each query, including whether an object is present and the object's properties, such as bounding box coordinates and class labels.

During training, DETR uses Hungarian matching to establish a one-to-one correspondence between predicted objects and ground truth objects. This matching process finds the optimal assignment that minimizes the total matching cost across all predictions and ground truths. The cost function for matching and the loss function for optimization typically consists of two main components: a classification loss on the presence prediction of an object, and another loss that penalizes errors in the predicted object properties. The total loss is computed over the matched pairs, ensuring each ground truth object is assigned to exactly one prediction.

In our work, we adapt DETR-style processing for radar detections, where a learned query corresponds to a radar point cluster of a dynamic object.

4.2 Data Collection and Training Pipeline Contributions

Below we describe the contributions we made to advance the overall downstream driving performance of TransFuser++.

4.2.1 Modernizing Model Architecture

Removing GRU: The decision to remove the GRU from the planning head was motivated by the desire to enhance the model's flexibility of conditioning. The transformer decoder alone is capable of modeling complex dependencies and interactions between trajectory points through its attention mechanism. Furthermore, the GRU's inductive bias to guide the planning explicitly with the target point may limit the model's ability to adapt to different conditioning signals.

To further stabilize training, we normalize target points by constant scaling factors to avoid the large variance in their magnitudes. Those two modifications encourage the model to learn planning behavior more from robust features that are actually relevant rather than relying on explicit geometric cues.

For this purpose, we apply a linear layer directly on the transformer decoder outputs to produce differenced outputs representing offsets between two steps. Besides predicting only the spatial trajectory and target speed as the original TF++ does, we also predict future positions. This provides additional supervision that encourages the model to learn temporal consistency and motion dynamics. For closed-loop evaluation, only the spatial trajectory and target speed outputs are used. Further experiments on additional supervision signals, such as direct controls and different planning frequencies, did not yield significant improvements.

Integration of Radar: We incorporated radar sensor data into the model to provide additional motion-salient information that complements the visual and LiDAR modalities. Since there are only a few radar detections per frame, we could either tokenize each detection individually and let the transformer attend to them directly, or cluster the detections into a fixed number of representative points.

We experimented with both approaches and found that clustering yields better performance. To be specific, we encoded each radar detection by applying grid sampling on BEV features where the detection was located. This feature vector was concatenated with the radial velocity measurement and passed through a linear layer to produce radar detection embeddings.

The learned queries, representing a detected vehicle, then attend to these radar detection embeddings together with BEV features and tokenized ego velocity through 4 cross-attention layers in the transformer decoder.

We trained the radar detection head using a DETR-style loss with Hungarian matching, where each query predicts object presence, bounding box coordinates,

and velocity vector. The detector is trained in the perception stage alongside other auxiliary heads and will be trained end-to-end with the planning head in the joint training stage.

As for planning, we used the radar queries as additional context for trajectory prediction.

Using Multiple Target Points as Local Navigation Conditioning: The original TF++ model receives only the next immediate target point as a conditioning signal for trajectory planning. This single-point representation lacks information about the upcoming trajectory structure, limiting the model's ability to perform anticipatory maneuvers. Our implementation utilizes the previous target point (already visited), the current target point (immediate goal), and the next target point (subsequent goal).

Direct application of this change did not lead to an improvement in performance. Inspecting the data revealed that for most samples, the current target point alone provided sufficient directional information, making the additional points redundant.

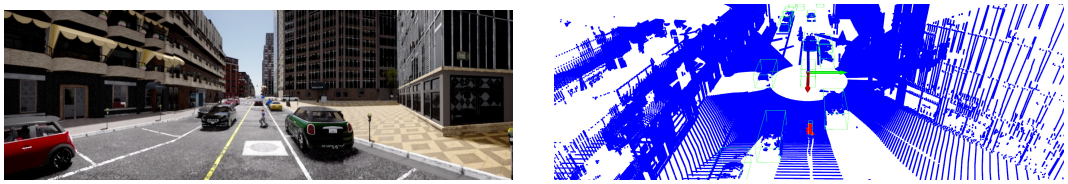
Behind this observation is the fact that the current target point is marked as visited as soon as the ego vehicle is within 7.5m of it. To address this, we reduced the visited threshold to 3m in training, ensuring that the next target point remains relevant for a longer duration. This change increased the qualitative numbers of samples where the subsequent target point provides a qualitatively significant contextual cue.

4.2.2 Redesigning the Expert Driving Style

Speed Limit Computation: We refined the expert's speed limit computation to ensure a more grounded and context-aware driving profile. While raw simulator API queries can sometimes result in aggressive velocity targets that exceed the intended safety margins, our updated approach harmonizes these queries with the real-time behavior of the surrounding environment.

Specifically, we integrated a hybrid estimation logic that considers both the official speed limit and the second-highest speed of nearby dynamic actors. By utilizing the minimum of these two values as the effective speed limit, the expert prevents unrealistic acceleration while naturally adapting to the observed traffic flow. This modification ensures that the expert demonstrates a more realistic, visibility-aware driving style—adhering to traffic regulations while avoiding the overly aggressive profiles often produced by omniscient planners.

Enhancing the Agent's Occlusion Check: The previous occlusion detection relied solely on LiDAR point clouds. To be concrete, to determine whether an agent in the scene should be considered in the planning process, we test whether there are sufficient LiDAR points within the agent's bounding box.



(a) (b)

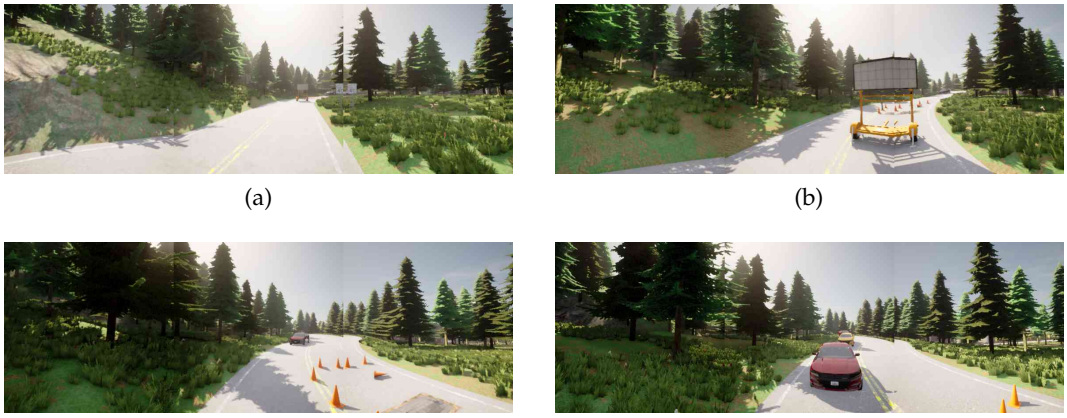
Figure 4.5: **(a)** Small pedestrians are often hit by a few LiDAR points **(b)** Camera point clouds are denser and semantically richer than LiDAR point clouds, making them more suitable for human-like occlusion detection.

We extended this approach by incorporating camera point clouds in addition to LiDAR data. The camera-based point cloud is generated using instance segmentation and depth map labels from the simulator and provides bounding boxes with visibility properties such as the number of visible pixels per object. Pedestrians and small objects are detected more reliably through camera data than through LiDAR alone, as cameras provide denser sampling and better capture the visual appearance of these kinds of objects. See Figure 4.5 for an illustration.

Adapting the Expert to the Student's Limitation: The TF++ student model is ultimately a BEV planner that learns to act in a limited BEV space. While the planner theoretically has access to everything the camera sees, practically, the spatially limited BEV supervision constrains what the planner can effectively see. Acknowledging this limitation, we modified the expert to also operate with a limited BEV FOV during data collection. For an effect of this change, see Figure 4.6.

Improving Driving Style in Uncertain Situations: In Leaderboard 1.0, occlusion handling was mostly unnecessary since the town environments were small and simple, allowing the student policy to maintain full visibility of all relevant agents. With Leaderboard 2.0, the larger and more complex town layouts introduce many occlusion scenarios, particularly at junctions and in dense traffic. To address this, we enhanced the expert's behavior in several key areas to better handle uncertainty and partial observability.

- *Weather-adaptive behavior:* The previous expert implementation did not adjust its behavior based on weather conditions, maintaining the same driving speed regardless of rain, fog, or clear conditions. We incorporated weather-adaptive behavior so that the expert drives more conservatively in adverse weather, reflecting how a human driver would respond to reduced visibility and decreased traction.
- *Occluded junction handling:* Beyond detecting which agents are visible, we implemented several mechanisms to assess the degree of occlusion in the scene, particularly near junctions where visibility is often limited. Using the camera point cloud, we compute an occlusion score for each camera that quantifies



(a) (b)

(c) (d)

Figure 4.6: **(a, b)** In this scene, the red car is outside of the expert's field of view, so the expert ignores that agent and initiates the construction site overtake sequence. **(c, d)** Since the estimation is not perfect, the gap was too small for smooth overtaking. To avoid a collision, the expert has to slow down and stop mid-overtake as soon as they detect the oncoming car.

visibility based on the distance and density of observable road-relevant objects. When the expert approaches a junction and detects high occlusion levels, it reduces speed to account for potentially hidden hazards.

- *Narrow streets with many parked cars:* In urban environments with narrow streets and parked vehicles, visibility can be severely limited. The expert now identifies such scenarios by analyzing the density of static obstacles and reducing target speed accordingly.
- *Approaching obstacles:* When approaching an obstacle, the expert reduces speed progressively and adjusts its safety margin dynamically based on visibility conditions: under good daylight conditions, it maintains a greater clearance distance for early gap detection and reduces speed earlier. Under reduced visibility (fog, rain, or nighttime), it approaches at a higher speed and a tighter margin.

Improving Driving Style when Encountering Pedestrians: The previous implementation relied on bounding box forecasting and collision prediction to determine when to stop for pedestrians, which could lead to late or unsafe reactions. In particular, the expert often does not fully stop for pedestrians until they are very close or already in the crosswalk, which can be confusing for the student model to learn from.

We simplified and improved this behavior by using visibility-based criteria derived from camera point clouds. The expert now stops when a pedestrian is detected with a sufficient number of visible pixels and exhibits motion above a certain speed

threshold. This approach provides a more direct and reliable stopping criterion that is grounded in observable visual features, making it more learnable for the sensor-based student model.

Improving Driving Style when Encountering Emergency Vehicles: A critical scenario in CARLA Leaderboard 2.0 involves emergency vehicles that take priority at junctions, often approaching at high speed. Collisions with these vehicles typically displace the ego vehicle off the road, effectively terminating the evaluation run prematurely. The previous expert did not need to consider those scenarios because of its privileged access to the full simulator state, allowing it to avoid collisions reliably.

In fact, the previous driving expert solved most of the scenarios by slowing down very slightly to let the emergency vehicle pass first, before proceeding through the junction itself.

To make the demonstrations clearer to student models, when an emergency vehicle is detected with a high number of visible pixels, or when a vehicle exhibits a very high approaching speed around a junction, our new expert triggers an immediate emergency stop regardless of its current trajectory. To further strengthen the training signal, the model only continues driving after the danger has fully passed. This defensive behavior teaches the student to recognize and yield to priority vehicles under time-critical conditions, preventing catastrophic failures.

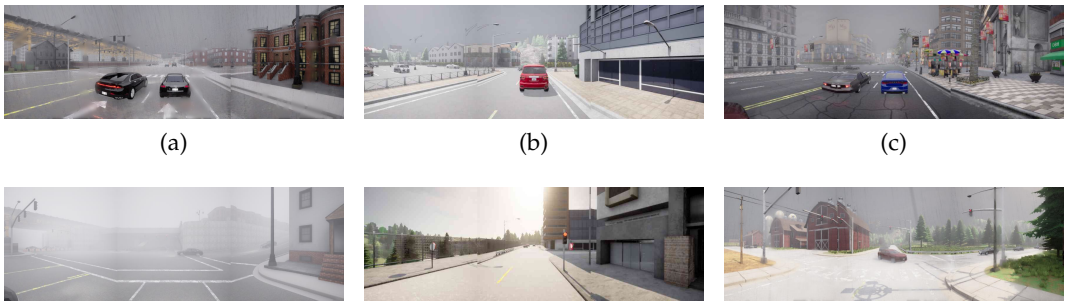
4.2.3 Polishing Training Dataset

Adjusting Perception Labels for Leaderboard 2.0: To better align the model's features with new diverse scenarios, we made several refinements to the perception labels.

- *Semantic segmentation labels:* We introduced new classes for several driving-relevant object categories like construction sites, emergency vehicles, and stop signs that were not available in the original camera semantic segmentation labels. On the other hand, we move non-driving-relevant classes, such as sidewalks, to the background class.
- *BEV segmentation labels:* Similarly, we removed sidewalks from the BEV segmentation and added construction sites, accident obstacles, and parking obstacles. New traffic lights requiring recognition for special stopping behavior were also added as a separate class.
- *Bounding box labels:* The new scenarios in Leaderboard 2.0 also required new bounding box labels. We refined the labeling for more subclasses at training time to account for this fact.
- *Handling of small objects:* For pedestrians and cyclists, we increased the BEV segmentation areas and bounding box sizes by 4 times to ensure they are large enough to be reliably detected by the model.

The previous implementation of BEV segmentation provides labels with every class drawn at collection time. We switched to collecting only road maps and drawing other classes at training time to allow for more flexible label changes.

Custom Scenario, Stopping at Red Lights: Analysis of the collected data revealed that in most red light scenarios, the ego vehicle stops behind other vehicles already queued at the intersection; see Figure 4.7. To mitigate this imbalance, we introduce a new scenario type that extends the red phase duration for all traffic lights in the town and removes all vehicles positioned in front of the ego. This ensures that the model learns the direct causal relationship between observing a red traffic light and performing a stop.



(a) (b) (c)

(d) (e) (f)

Figure 4.7: **(a,b,c)**: stopping for red gets easier with a lead vehicle. **(d,e,f)**: adverse weather and no lead vehicle forces the model to learn robust causal relationships between red light and stopping behavior.

Custom Scenario, Defective Traffic Lights: We introduced a scenario in which all junction directions simultaneously display green signals; see Figure 4.8. The ego vehicle must navigate through an intersection while facing continuous streams of cross-traffic from the left and/or right, requiring gap detection and assertive merging behavior to reach the opposite side.

Custom Scenario, Unprotected Left Turn with Competing Flow: We introduced a more complex variant of unprotected left turns, where the ego attempts to merge into the oncoming traffic flow; see Figure 4.9. The ego must not only find a gap in the existing traffic stream but also compete with other vehicles trying to enter the same lane from different approach angles. This teaches the model to handle competitive merging situations where multiple agents compete for limited gaps in dense traffic, requiring more sophisticated gap selection and assertive execution.

Reusing Route Descriptions for More Effective Data Labeling: We observed that certain scenario types, such as accident obstacles and construction sites, share identical XML scenario descriptions except for their naming and geographical distribution across the map; see Figure 4.10. This insight leads to two opportunities for more efficient data labeling.

Figure 4.8: Defective traffic lights. The ego has to find gaps in slow but continuous cross traffic to proceed through the intersection.

For existing route descriptions that are underrepresented in the dataset, we duplicated the XML files and renamed the scenario types to create new routes. This allowed us to increase the frequency of rare scenarios without additional labeling efforts.

Furthermore, for the actual labeling process, a single labeled XML file from manual laboring could be reused for multiple different scenario types that share the same structure, effectively saving time and increase the throughput.

Adding Missing Towns: The dataset used for training was missing route definitions for Town06 and Town07, which in turn limited the diversity of map environments available for training. To address this gap, we adapted XML route files from CARLA Garage [5] and translated them to the Leaderboard 2.0 format.

Diversifying Weather Presets: The original dataset used 21 weather presets that lacked sufficient diversity to capture the range of environmental conditions encountered during evaluation. To improve the model's robustness to weather variations, we introduced 30 additional weather presets, including foggy conditions, several adversarial weather scenarios, and dawn lighting.

One preset is randomly sampled for each route. The sampled preset incorporates slight randomization to the sun position, fog density, and rain intensity to further increase the weather diversity of the dataset.

Adding Routes for Town15: To address missing routes in the new Town15, where no official routes existed, we used an open-source labeling tool to manually author new route definitions for this town [1]. Combined with the route duplication technique described above, this effort resulted in 800 additional routes that significantly increased the coverage of these challenging scenario types. The trigger point of each route file also shifted a few meters to increase visual diversity, see Figure 4.11.

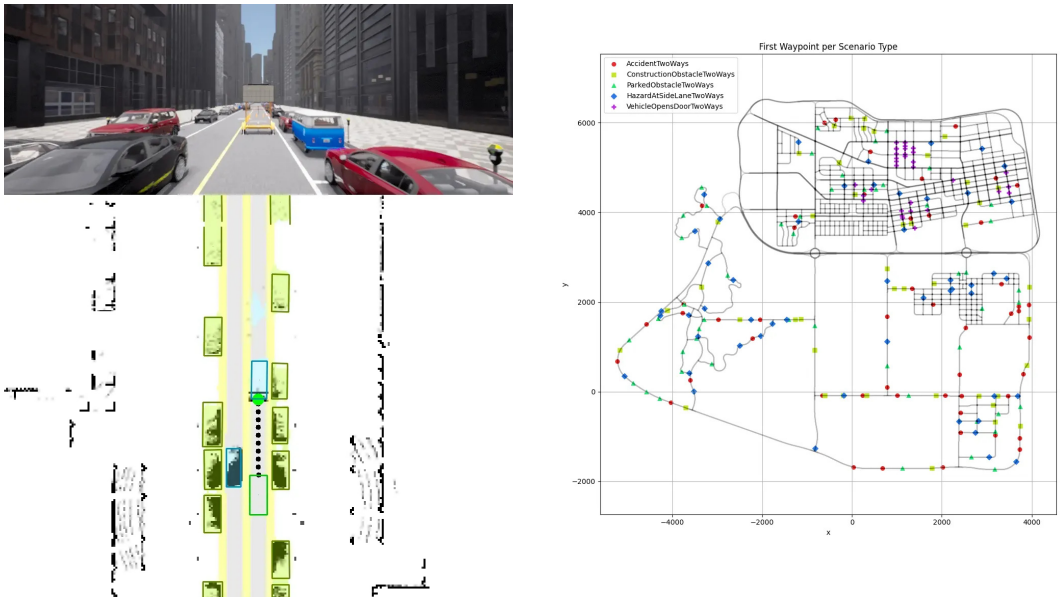
Enhancing the Existing Sensor Rig: The previous setup relied on a single front-

Figure 4.9: Unprotected left turn with competing Flow. Both front and rear collisions are possible if the model fails to find a suitable gap in time.

facing camera and one LiDAR sensor. We expanded this to two LiDAR sensors, three front-facing cameras, and eventually six cameras to provide 360-degree coverage. Each camera has a FOV of 60 degrees with some overlap between adjacent views, creating a wide frontal perception range without introducing significant distortion.

The two LiDAR sensors are positioned to cover complementary half-spaces around the vehicle. For the LiDAR data, we stack the point clouds from the previous five timesteps at 20 Hz, providing temporal context that encodes velocity cues and motion information.

Extending the Sensor Rig with Radar: Radar sensors in CARLA provide at most 75



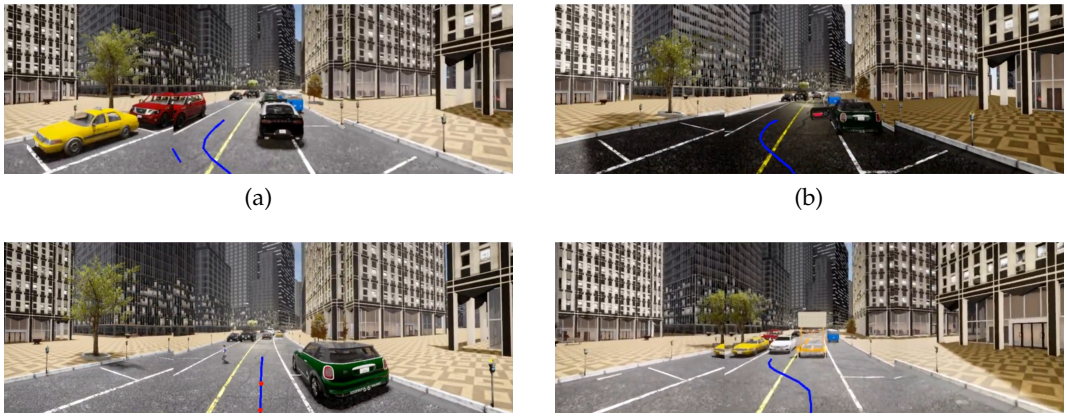
(a) (b)

Figure 4.10: **(a)** Inspections show that most failures in ConstructionObstaclesTwoWays come from lack of spatial coverage in urban locations. Here the construction site is misclassified as vehicle, leading to wrong planning decisions. **(b)** Analysis of route descriptions shows that many scenario types with the same structure have complementary spatial distribution. For example, VehicleOpensDoorTwoWays can be reused as ConstructionObstaclesTwoWays without any further change while contributing more geographical diversity to the latter.

detections per frame. To maximize the sensor's coverage, we set the vertical FOV to be 1 degree and the horizontal FOV to be 90 degrees. We mount two radars on the front left and front right corners of the ego vehicle, each turned at a 45-degree angle outward to cover a wide frontal area. Similarly, we mount two radars on the rear left and rear right corners. An illustration of a scenario that could benefit from radar can be found in Figure 4.12

Generating More Realistic Sensor Perturbations: To improve the model's robustness to positioning errors and lane tracking deviations, the previous system applied random perturbations to camera and LiDAR poses during data collection. The sensor outputs gave impression of slight translated and rotated ego vehicle in world frame while keeping the ground truth trajectory unchanged.

We increased the magnitude of those perturbations to better simulate real-world localization noise and ego motion uncertainty. On narrow streets, the translation magnitudes are lowered to avoid physically implausible situations where the ego vehicle appears to be on the opposite lane, almost crashing other vehicles, while the



(a) (b)

(c) (d)

Figure 4.11: Labeling custom routes is time-consuming. Exploiting scenario structure similarities for efficient route labeling. A labeled XML file **(a)** can be reused for multiple scenario types **(b,c,d)**.

ground truth trajectory continues driving.

The rotation parameters are adapted based on the translation parameter to avoid situation where the intentions can be misinterpreted from the sensor parameters. In particular, we want to avoid situations where it seems like ego is doing a lane change but the ground truth trajectory continues to stay on current lane.

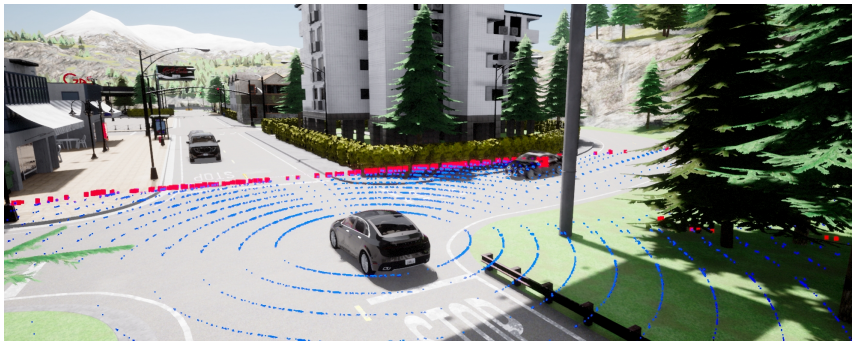
4.2.4 Adapting Data and Training Pipeline

Caching Data for Higher Training Throughput: Data preprocessing for sensor inputs and labels is computationally expensive, and loading times on distributed training systems can become a significant bottleneck. To mitigate this, we implemented a Pickle-based caching system where all data required for a single training sample is preprocessed once and stored in a single cache file [39].

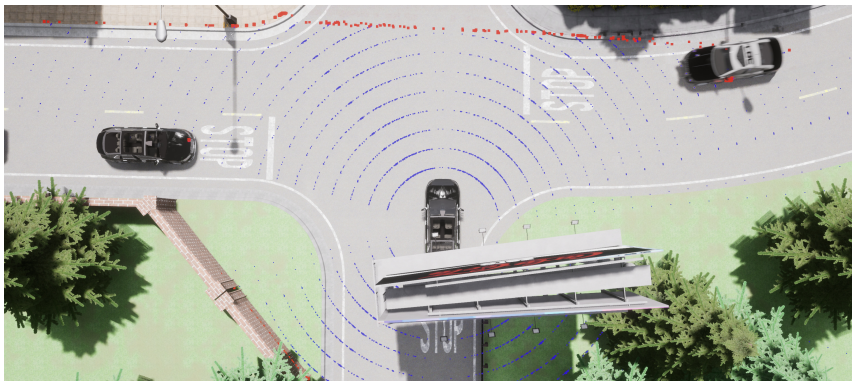
This approach eliminates redundant preprocessing operations and reduces I/O overhead during training, as each sample can be loaded directly from its cached representation and therefore requires only one disk access.

Employing Mixed Precision Training: To reduce memory consumption and accelerate training, we employ mixed-precision training using the BF16 (bfloat16) numerical format for the majority of operations. However, to maintain numerical stability, we enforced FP32 (float32) precision for critical operations, including normalization layers, softmax activations, and cross-entropy loss computations.

Designing a More Effective Data Compression Scheme: To reduce storage requirements and improve data loading efficiency, we implemented several compression



(a)



(b)

Figure 4.12: A dangerous situations where the ego has to brake sharply: Blue points indicate free space while red points represent obstacles with a size proportional to the detected relative velocity. Static images or LiDAR alone would struggle to infer motion cues that radar provides natively.

optimizations for the training data.

- *Model-based LiDAR filtering:* We applied a filtering step to the raw LiDAR point clouds to remove points that are unlikely to contribute to driving decisions, such as points on the ground or distant background structures.
- *Depth label downsampling:* We reduced the resolution of depth labels by a factor of 4, as high-resolution depth maps are not necessary for effective spatial reasoning.
- *Efficient serialization and compression:* We replaced the original GZIP compression with a JSON serialization scheme, with a more efficient combination of LZMA compression, Pickle serialization, and float16 encoding for numerical data.
- *Storing camera images:* We stored camera images in JPEG format, with quality levels depending on time of day and weather conditions. For example, images captured during nighttime or foggy weather are stored at a higher quality to

preserve important visual details.

These combined optimizations reduced the dataset size from approximately 500 GB to 100 GB, achieving a 5× compression ratio.

5 Experiments

We evaluated our approaches in CARLA 0.9.15 following largely the experimental setup of [44]. As a deviation, we trained on *all available routes*. This corresponds to an L4 driving environment, where the agent is expected to operate in locations that it has already encountered during training.

5.1 New Expert

Table 5.1: New expert on Bench2Drive.

	DS ↑	SR ↑
Baseline	83.56 ± 0.34	0.66 ± 0.01
New Dataset	84.94 ± 0.50	0.67 ± 0.01

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	15	21	183	18	15	0	3	12
New Dataset	46	12	134	45	11	5	0	23

Note: **CL**: Collision Layout; **CP**: Collision Pedestrian; **CV**: Collision Vehicle; **OL**: Outside Lane; **RL**: Red Light; **RD**: Route Deviation; **SI**: Stop Infraction; **VB**: Vehicle Blocked.

Table 5.2: New expert on Longest6 v2.

	DS ↑	RC ↑
Baseline	22.51 ± 4.42	70.68 ± 8.24
New Dataset	34.05 ± 1.50	62.68 ± 11.39

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	57	6	155	48	65	23	23	27
New Dataset	48	3	95	35	8	14	7	34

Furthermore, we evaluated the combined effect of all modifications made to the expert and the dataset described in Chapter 4.2.2 and Chapter 4.2.3. We trained on

the training routes described in [44] and used the same TransFuser++ architecture. The results are shown in Table 5.1 and Table 5.2.

The results show mixed improvements on both benchmarks. For Bench2Drive, the driving score improved slightly from 83.56 to 84.94, accompanied by a noticeable reduction in vehicle collisions. The increase in outside route lane violations is partly attributable to obstacle avoidance behavior: for scenarios with static blockages, the new expert must leave the assigned lane to bypass the obstacle safely. This behavior incurs a minor infraction penalty but prevents more severe collisions, reflecting a trade-off inherent in the scenario design.

On Longest6 v2, the driving score increased from 28.88 to 34.05 despite a decrease in route completion from 74.82% to 62.68%. This apparent contradiction is explained by the multiplicative structure of the driving score: a decrease in route completion leads to fewer infractions, which increases the infraction score component.

The infraction breakdown shows substantial improvements in traffic rule compliance: red light violations drop from 65 to 8 total occurrences, and stop sign violations decrease from 23 to 7 occurrences. Vehicle collisions also improved significantly, decreasing from 155 to 95 occurrences.

Although individual infraction improvement could be attributed to the fact that the ego vehicle has a lower route completion, drives less distance, and thus has fewer opportunities to collide, the overall DS improvement is a strong indicator that the expert modifications successfully transferred to the learned policy.

5.2 Removing GRU

Table 5.3: Effect of Removing GRU on Bench2Drive.

	DS ↑	SR ↑
Baseline	84.94 ± 0.50	0.67 ± 0.01
Removing GRU	87.26 ± 0.47	0.70 ± 0.01

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	46	12	134	45	11	5	0	23
Removing GRU	21	9	121	13	9	37	0	4

Note: **CL**: Collision Layout; **CP**: Collision Pedestrian; **CV**: Collision Vehicle; **OL**: Outside Lane; **RL**: Red Light; **RD**: Route Deviation; **SI**: Stop Infraction; **VB**: Vehicle Blocked.

Following the approach discussed in Section 4.2.1, we first removed the GRU on top of the transformer decoder. Additionally, we separated the ego speed and driving command, which had previously been concatenated into a single conditioning token,

Table 5.4: Effect of Removing GRU on Longest6 v2.

	DS ↑	RC ↑
Baseline	34.05 ± 1.50	62.68 ± 11.39
Removing GRU	40.70 ± 2.86	76.82 ± 6.89

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	48	3	95	35	8	14	7	34
Removing GRU	14	5	120	20	14	34	14	6

into two distinct tokens. This allows the model to respond independently to speed and navigation intent, reducing potential interference between two sources of control information. The results are shown in Table 5.3 and Table 5.4.

The removal of the GRU and the transition to token-based navigation conditioning yielded substantial improvements across both benchmarks. On Bench2Drive, the driving score increased from 84.94 to 87.26. The infraction breakdown shows broad reductions across multiple categories: layout collisions decrease from 46 to 21 occurrences, outside route lanes violations reduce from 45 to 13 occurrences, and vehicle blocked events lower from 23 to 4 occurrences. However, the route deviation infractions increased substantially from 5 to 37 occurrences. As previously discussed in Section 4.1.11, the original TransFuser++ modeled an almost linear relationship between the input target point and the model's output trajectory. This strong inductive bias gets disrupted, leading the model to occasionally prioritize local scene context over strict adherence to the planned route. Normalizing the target point further reduces the model's sensitivity to target point locations, which could also contribute to increased route deviation infractions.

On Longest6 v2, the improvement is even more pronounced: the driving score increased from 34.05 to 40.70, and route completion improved substantially from 62.68% to 76.82%. Notably, the improvements in several infraction categories occur despite the longer driving distance: layout collisions decrease dramatically from 48 to 14 occurrences, lane departure violations drop from 35 to 20 occurrences, and vehicle blocked events are nearly eliminated, decreasing from 34 to 6 occurrences.

However, several types of infractions show absolute increases: vehicle collisions rise from 95 to 120 occurrences, red light violations increase from 8 to 14 occurrences, stop sign violations double their occurrences from 7 to 14, and route deviation grows from 14 to 34 occurrences. Because the model now drives 23% further on average, part of this increase may be attributed to greater exposure to traffic scenarios rather than degraded decision-making. The more substantial increase in route deviation aligns with the Bench2Drive results.

Despite the mixed infraction profile, overall performance improvements, especially

the significant gains in driving score and route completion on Longest6 v2, confirm that relying on a transformer decoder's reasoning capacity is more effective than the strong but brittle inductive bias of the original GRU-based architecture. The increase in route deviation infractions represents a manageable trade-off that can be addressed through subsequent refinements to the conditioning mechanism or on the navigation system, as we will discuss.

5.3 Multiple Target Points Conditioning

Table 5.5: Multiple Target Points Conditioning on Bench2Drive.

	DS ↑	SR ↑
Baseline	87.26 ± 0.47	0.70 ± 0.01
Multiple Target Points	89.29 ± 0.54	0.76 ± 0.01

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	21	9	121	13	9	37	0	4
Multiple Target Points	15	1	91	11	10	22	0	2

Note: **CL**: Collision Layout; **CP**: Collision Pedestrian; **CV**: Collision Vehicle; **OL**: Outside Lane; **RL**: Red Light; **RD**: Route Deviation; **SI**: Stop Infraction; **VB**: Vehicle Blocked.

Table 5.6: Multiple Target Points Conditioning on Longest6 v2.

	DS ↑	RC ↑
Baseline	40.70 ± 2.86	76.82 ± 6.89
Multiple Target Points	42.13 ± 0.75	62.87 ± 2.07

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	14	5	120	20	14	34	14	6
Multiple Target Points	3	0	77	22	10	60	7	2

As discussed in Section 4.2.1, we modified the navigation conditioning mechanism to provide the model with three sequential target points instead of a single target point. By exposing the model to the local trajectory structure, this modification enables anticipatory behavior: the agent can initiate lane changes earlier when it detects that the upcoming target point requires repositioning.

The results are shown in Table 5.5 and Table 5.6. The introduction of three-point trajectory conditioning produced clear improvements on Bench2Drive, but mixed results on Longest6 v2.

On Bench2Drive, the driving score increased from 87.26 to 89.29. The infraction breakdown shows broad reductions across multiple categories: vehicle collisions decrease from 121 to 91 occurrences, route deviation drops substantially from 37 to 22 occurrences, and layout collisions reduce from 21 to 15 occurrences.

On Longest6 v2, the results are less conclusive. The driving score shows a modest increase from 40.70 to 42.13 (a gain of 3.5%), while route completion drops from 76.82% to 62.87%. The infraction breakdown reveals substantial improvements in several safety-critical categories: layout collisions decrease dramatically from 14 to 3 occurrences, and vehicle collisions drop from 120 to 77 occurrences. However, route deviation increases substantially from 34 to 60 occurrences.

The mixed performance on Longest6 v2 indicates that while three-point conditioning provides clear benefits for safety and collision avoidance, other system limitations may prevent the full potential of this modification from being realized. Video analysis of the evaluation runs suggested that untuned controller parameters and insufficient data coverage for certain scenario types and Town06 represent potential bottlenecks that could mask the benefits of improved trajectory reasoning. Despite the reduction in route completion, the overall trend suggests that three-point conditioning does improve driving safety, particularly by substantially reducing vehicle collisions and layout collisions.

5.4 Radar Fusion

Table 5.7: Radar fusion on Bench2Drive.

	DS ↑	SR ↑
Baseline	89.29 ± 0.54	0.76 ± 0.01
Radar Sensing	90.01 ± 0.42	0.76 ± 0.01

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	15	1	91	11	10	22	0	2
Radar Sensing	19	8	86	14	6	35	0	5

Note: **CL**: Collision Layout; **CP**: Collision Pedestrian; **CV**: Collision Vehicle; **OL**: Outside Lane; **RL**: Red Light; **RD**: Route Deviation; **SI**: Stop Infraction; **VB**: Vehicle Blocked.

The baseline model processes only single frames from camera and LiDAR sensors, relying on implicit temporal reasoning to infer velocity and motion dynamics. Although velocity and temporal cues can, in principle, be extracted from static observations, for example, fast-moving vehicles typically have more space around them, while slow or stopped vehicles are often surrounded by other traffic. Such indirect inference is less principled and more error-prone than directly observing

Table 5.8: Radar fusion on Longest6 v2.

Configuration	DS ↑	RC ↑
Baseline	42.13 ± 0.75	62.87 ± 2.07
Radar Sensing	42.60 ± 2.56	65.30 ± 3.38

	CL↓	CP↓	CV↓	OL↓	RL↓	RD↓	SI↓	VB↓
Baseline	3	0	77	22	10	60	7	2
Radar Sensing	10	2	78	38	12	50	9	6

velocity information through dedicated sensors.

To provide the model with explicit velocity information, we incorporated radar sensor data into the perception pipeline. To the best of our knowledge, this represents the first work to integrate radar sensing into an end-to-end learning system within the CARLA ecosystem. The results are shown in Table 5.7 and Table 5.8.

The integration of radar sensing yields modest improvements across both benchmarks, although the results remain inconclusive. On Bench2Drive, the driving score increased slightly from 89.29 to 90.01. Vehicle collisions decreased marginally from 91 to 86 occurrences. On Longest6 v2, the driving score shows a minimal increase from 42.13 to 42.60, with route completion slightly improving from 62.87% to 65.30%. The infraction profile similarly shows mixed trends.

The modest performance gains suggest that other system bottlenecks now dominate the overall behavior. At this stage of development, limitations in controller tuning, scenario coverage, and dataset diversity may mask the potential benefits of velocity-aware reasoning. However, since the integration of radar is relatively straightforward and computationally inexpensive and provides measurable improvement, it remains a valuable addition to the perception stack. Furthermore, future benchmarks that emphasize dynamic agent interaction may better showcase the advantages of radar sensing.

5.5 Further experiments

In the following sections, we describe additional experiments conducted to further improve closed-loop performance. Those experiments build on the previous sections without changing the architecture or training procedure. We summarize the key findings in Tables 5.9, 5.10, 5.11 in Section 5.6. Bench2Drive further provides multi-ability metrics, which we report in Table 5.12

5.5.1 Expanded Dataset

We included all routes from Town06, Town07, and Town15 in addition to the existing data. This increased the size of the dataset from approximately 48 hours to 72 hours of expert driving data. No changes are made to architecture, loss, optimizer, or expert; therefore, any downstream difference is due solely to additional data.

5.5.2 Calibrating Controllers

The PID controller was originally tuned for the baseline model's speed profile, and we hypothesize that suboptimal controller parameters may limit performance when the model's speed profile changes.

To address this, we retuned the lateral controller by adding a parameter that increases the steering sensitivity when the predicted route has high curvature.

As for the target point controller, we decreased the threshold to pop a target point of the queue in case the ego vehicle is in proximity of multiple target points from 5 meters to 4 meters. In this situation, the ego vehicle is likely approaching a complex intersection or sharp turn, and popping target points later allows the controller to better follow the intended path.

Furthermore, we skip the subsequent target point if the distance to that point is larger than 50 meters. We do this by simply replacing the subsequent target point with the current target point. This prevents the large magnitude of the subsequent target point from destabilizing the controller. We acknowledge that a more principled approach would be to applied a more sophisticated normalization technique and train the model simply on more variability of target point distribution. However, this simple heuristic already yields noticeable improvements.

5.5.3 360 Camera

The baseline model configuration uses three front-facing cameras (front-left, front-center, front-right) to capture the forward FOV. To mirror a production vehicle's configuration and eliminate blind spots, we added three additional rear-facing cameras (back-left, back-center, back-right). Each camera retains the same resolution and frame rate as in the baseline configuration, while the overlapping FOV between adjacent cameras ensures seamless coverage.

5.6 State-of-the-Art Results

In the following comparison to prior work, we report Driving Score (DS) and Success Rate (SR) for Bench2Drive. For Longest6 V2 and Town13 Validation, we report only DS, as these are the quantities that are consistently available across methods. The

Bench2Drive table is a non-exhaustive summary; for a more complete list, we refer to [2].

We also want to note that the relative improvements are not linearly proportional to actual capability of the system. Examining the Figure 4.3 we can see that an improvement of the scores from 20 to 40 needs to reduces more infractions than to improve the scores from 60 to 80.

Table 5.9: State of the art on Bench2Drive.

Method	**DS ↑**	**Inputs**	**Expert**	**Backbone**
Hydra-NeXt [23]	73.86	2C	Think2Drive	ResNet50
VLR-Driver [22]	75.01	Undisclosed	Think2Drive	Undisclosed
ORION [17]	77.74	6C	Think2Drive	EVA-02-L
AutoVLA [43]	78.84	6C	Think2Drive	Qwen2.5-VL-3B
[1] Reproduced TF++	**83.56 ± 0.34**	**1C+1L**	**PDM-Lite**	RegNetY-032
TF++ [44]	84.21	1C+1L	PDM-Lite	RegNetY-032
[2] New Expert	**84.94 ± 0.50**	**3C+2L**	**PDM-Lite**	RegNetY-032
SimLingo [32]	85.07 ± 0.95	1C	PDM-Lite	InternViT-300M
R2SE [25]	86.28	1C	PDM-Lite	Undisclosed
HiP-AD [38]	86.77	6C	Think2Drive	ResNet50
BridgeDrive [26]	86.86 ± 0.88	1C+1L	PDM-Lite	RegNetY-032
[3] Removing GRU	**87.26 ± 0.47**	**3C+2L**	**PDM-Lite**	RegNetY-032
[4] Three TPs	**89.29 ± 0.54**	**3C+2L**	**PDM-Lite**	RegNetY-032
[5] Radar Fusion	**90.01 ± 0.42**	**3C+2L+4R**	**PDM-Lite**	RegNetY-032
[6] Dataset Expansion	**94.01 ± 1.48**	**3C+2L+4R**	**PDM-Lite**	ResNet34
[7] Calibrated Controller	**94.72 ± 0.72**	**3C+2L+4R**	**PDM-Lite**	ResNet34
[8] 360 Camera	**95.04 ± 0.71**	**6C+2L+4R**	**PDM-Lite**	ResNet34
[9] Final Model	**95.28 ± 0.36**	**3C+2L+4R**	**PDM-Lite**	RegNetY-032
PDM-Lite (expert) [6]	97.02	priv	–	–

C = camera, L = LiDAR, R = radar, priv = privileged state access.

Table 5.10: State of the art on Longest6v2.

Method	DS ↑	Inputs	Expert	Backbone
HiP-AD [38]	7	6C	Think2Drive	ResNet50
[1] Reproduced TF++	**28.88 ± 10.20**	**1C+1L**	**PDM-Lite**	**RegNetY-032**
[2] New Expert	**34.05 ± 3.52**	**3C+2L**	**PDM-Lite**	**RegNetY-032**
[3] Removing GRU	**40.70 ± 1.30**	**3C+2L**	**PDM-Lite**	**RegNetY-032**
[4] Three TPs	**42.13 ± 1.80**	**3C+2L**	**PDM-Lite**	**RegNetY-032**
[5] Radar Fusion	**42.60 ± 2.56**	**3C+2L+4R**	**PDM-Lite**	**RegNetY-032**
[6] Dataset Expansion	**50.01 ± 2.86**	**3C+2L+4R**	**PDM-Lite**	**ResNet34**
[8] 360 Camera	**54.16 ± 5.32**	**6C+2L+4R**	**PDM-Lite**	**ResNet34**
[7] Calibrated Controller	**57.74 ± 2.99**	**3C+2L+4R**	**PDM-Lite**	**ResNet34**
[9] Final Model	**62.92 ± 1.58**	**3C+2L+4R**	**PDM-Lite**	**RegNetY-032**
PDM-Lite (expert) [6]	73	priv	–	–

C = camera, L = LiDAR, R = radar, priv = privileged state access.

Table 5.11: Results on Town13 validation routes. Models were trained on all towns, and no early termination was applied.

Method	RC ↑	DS ↑	NDS ↑
TF++ [44]	68.53	0.96	4.94
[7] 3 Cameras ResNet34	**71.82 ± 8.04**	**5.01 ± 0.96**	**14.65 ± 1.29**
[8] 6 Cameras ResNet34	**53.95 ± 10.20**	**5.01 ± 0.97**	**9.49 ± 4.19**
[9] 3 Cameras RegNetY-032	**74.69 ± 9.05**	**4.32 ± 0.12**	**14.06 ± 4.13**
PDM-Lite	92.35	40.20	61.55

Table 5.12: Multi ability scores on Bench2Drive evaluation protocol.

Method	Merg. ↑	Overtak. ↑	Emer. Brake ↑	Give Way ↑	Traf. Sign ↑	Mean ↑
TCP-traj*	8.89	24.29	51.67	40.00	46.28	34.22
UniAD-Base	14.10	17.78	21.67	10.00	14.21	15.55
VAD	8.11	24.44	18.64	20.00	19.15	18.07
DriveTransformer	17.57	35.00	48.36	40.00	52.10	38.60
ORION	25.00	71.11	78.33	30.00	69.15	54.72
TransFuser++	58.75	57.77	83.33	40.00	82.11	64.39
HiP-AD	50.00	84.44	83.33	40.00	72.10	65.98
SimLingo	54.01	57.04	88.33	53.33	82.45	67.03
BridgeDrive	63.50	58.89	88.34	50.00	88.95	69.93
[9] Final Model	**72.50**	**97.77**	**91.66**	**40.00**	**89.47**	**78.28**

6 Discussion

6.1 Failure Modes of Models

Yielding to Emergency Vehicles: This scenario expects the model to switch lanes to yield to an emergency vehicle approaching from behind. We observed that most models fail to yield to the emergency vehicle. We hypothesize that this is due to the rarity of this event in the training data. A pursuit to solve this challenge with a single-frame model may be futile, as the model might struggle to return to the original lane after yielding, especially once the emergency vehicle is no longer visible. This could lead to route deviation infractions on longer benchmarks.

Missing Highway Exits: On highways, we observed that models often miss exits, particularly when multiple lane changes are required. This is an important failure mode to address in future work, as missing an exit directly terminates the episode. It also raises the question of metric fairness, since in real-world driving, missing an exit is often safer than attempting a high-risk late lane change. Moreover, the difficulty is partly due to limitations of the simulation environment: highway exits in CARLA are often poorly signaled, and the A* planner provides very little time for the model to initiate a lane change. Performing an early lane change can result in an "out-of-lane" infraction, constraining the model to switch lanes only when explicitly instructed by the navigation system.

Performing Multiple Lane Switches Before Intersections: We observed that models often struggle with performing multiple lane switches before intersections. This is a common scenario in the Longest6 V2 benchmark, which significantly reduces the model's success rate. We hypothesize that this is caused by the scarcity of such examples in the training data.

Navigating Dense Intersections: We observed that even the best of our models still frequently have issues with unprotected left and right turns, particularly in dense urban intersections with multiple dynamic agents. These situations require the model to predict the intentions of other vehicles and time its own acceleration precisely to avoid collisions or unnecessary delays.

Overall, many failure modes observed in our models are often related to their inability to recover from route deviations.

6.2 Further Research and Work Directions

Several promising directions emerge from our findings:

- **Improved Metrics and Evaluation Protocols:** Current metrics at times disproportionately penalize deviations from the reference route, especially over longer trajectories. A deviation at the start of a route can lead to a zero score, even if the model could have completed the remaining distance without any other infractions.

  In many cases, such early failures are not caused by the model itself but stem from an insufficiency in the navigation interface, or other stochastic factors unrelated to driving competence.

  This can obscure meaningful progress in driving behavior and exaggerate the effect of minor or unlucky route differences. Future evaluation protocols could therefore incorporate mechanisms that penalize route deviations while still allowing continued evaluation, making it possible to assess overall driving competence more reliably. In particular, such a protocol could allow route deviation under penalty. Upon deviation, the agent is returned to the nearest on-route waypoint. A recovery budget of n resets/km applies; exceeding this budget would terminate the episode.

- **Refined Benchmarks for Leaderboard 2.0:** While Bench2Drive provides a broad and diverse evaluation setting, we observed that even models with low performance on other benchmarks, for example Longest6 V2, can achieve competitive results on Bench2Drive. For example, HiP-AD performs well on Bench2Drive despite struggling elsewhere.

  Conversely, Longest6 V2 includes routes that are unrealistically difficult even for human drivers, while Town13 Validation over-penalizes early deviations, making it harder to evaluate models fairly.

  These findings suggest that current benchmarks could benefit from refinement to better balance difficulty, realism, and robustness of evaluation.

- **Reinforcement Learning Fine-Tuning:** Current sensor perturbation approaches enable recovery from small deviations. However, as the Longest6 V2 evaluation shows, when models operate far out of distribution and the target point bias is removed, they often fail to return to the driving route even when the target points provide a clear cue.

  This limitation arises because recovery behavior is not part of the training process. Fine-tuning models through closed-loop interaction could allow them to learn corrective behaviors that help mitigate such situations and improve robustness during long-horizon driving.

Taking everything into account, improving evaluation protocols and benchmark

design will be key to enabling researchers to perform more meaningful investigations and obtain conclusive insights, rather than being constrained by current system-level bottlenecks. In the meantime, post-training techniques can be investigated to increase the recovery capability of IL-trained models.

6.3 Limitations

Model Generalization: We acknowledge the fact that our work did not consider the generalization aspect of trained models. All benchmark evaluations were conducted on scenes present in the training dataset, meaning the model has seen the locations at least once. Evaluations on an internal generalization benchmark indicate that our models still struggle to generalize to unseen scenarios. This indicates that while our models perform well on the training distribution, they may not be robust to variations in the environment or task.

Nonetheless, we argue that current research in end-to-end driving still struggles to achieve consistent and reliable performance even in seen environments. Many recent works fail to reach full completion on standard benchmarks despite extensive overfitting to training towns and scenarios. Therefore, we believe that before tackling the more ambitious challenge of out-of-distribution generalization, it is crucial to first address the fundamental bottlenecks in perception, planning, and evaluation, which limit robustness within the training domain itself.

Expert Logic: The logic to handle pedestrians in our system remains fragile and unrealistic. Realistic pedestrian behavior is often ambiguous and highly variable, making it difficult for rule-based experts to reliably predict intent. For CARLA's simplified pedestrian behavior, the current "brake when visible and moving" heuristic might be sufficient, but it is overly cautious in urban environments. A more structured modeling of pedestrian intent and temporal reasoning could improve such interactions in future works.

Computational Requirements: Another limitation is the computational demand of our training setup. Training a single ResNet-34 model from scratch requires approximately three to four days on high-end hardware, and large-scale experiments demand considerable GPU resources. This prolongs the development cycle and limits the ability to perform systematic architecture or hyperparameter exploration. Efficient fine-tuning strategies, frozen backbones, and lightweight closed-loop evaluation proxies could make research more accessible and sustainable.

6.4 Other Experiments and Notes for Future Research

Dataset Distillation: To reduce training cost, we explored dataset distillation by enriching metadata for each sample and selecting a smaller but more representative

subset. The distilled dataset contained only 42% of the original samples, reducing post-training time to 17 hours, but resulting in performance drops of 2 DS on Bench2Drive and 10 DS on Longest6 V2. We hypothesize that the model was undertrained, since the number of epochs was not adjusted accordingly. The underlying infrastructure for dataset distillation has been implemented and remains available for future work to build upon and explore improved subset selection strategies.

Model-Predictive Controller for Waypoint Planning: We also experimented with a version of DiffusionDrive [24], where the diffusion model predicts future ego-positions that are then executed via a model-predictive controller. The performance approached that of TransFuser++ on Bench2Drive after enough fine-tuning of the controller.

Currently, all leading models in CARLA separate the future positions into spatial trajectory and target speed. In terms of implementation, the target speed is also simplified using vehicle dynamics, which further streamlines training but complicates sim2real research.

The results of this experiment suggest that a general waypoint-based trajectory representation (future positions) can achieve performance comparable to the currently dominant approach in CARLA. A well-tuned controller is sufficient to interpret these waypoints efficiently. We hope this insight will encourage further research on general trajectory representations, which are more transferable to real-world systems.

Enhanced Average-Pooling Tokenizer: In the TransFuser backbone, three out of four fusion stages rely on PyTorch's `AdaptiveAvgPool2d`, which may restrict representational expressiveness. To address this, we replaced the average-pooling layers with learnable convolutional filters that also increase the output dimensionality. This modification resulted in minor improvements on Longest6 V2, but the gains were not consistent across runs, leaving the results inconclusive.

Vision Transformer Backbone: Following the release of DINOv3 [37], we integrated its ViT-B variant into our system by replacing the self-attention fusion modules with cross-attention layers, in which the queries correspond to tokenized LiDAR feature maps. The resulting model, with roughly 200 M parameters, ran successfully within our framework, but required twice the training time and achieved a slightly lower DS (–2 DS) on Bench2Drive. We hypothesize that non-convolutional backbones may demand extensive hyperparameter tuning to match the inductive biases of driving datasets, an avenue that remains promising but was beyond our computational budget. This direction remains worth exploring, given the strong performance of Vision Transformers on real-world perception and driving benchmarks.

6.5 Qualitative Evaluation of Driving Performance

We refer the reader to our accompanying YouTube video for qualitative demonstrations of our policy in closed-loop execution. A complete set of videos demonstrating our model's performance on Longest6 V2 and Bench2Drive will be included in the repository release.

6.6 Conclusion

The work presented in this thesis is a comprehensive and systematic modernization of the closed-loop end-to-end driving pipeline in CARLA, focusing on improving its supervision, conditioning, and training infrastructure. Through these targeted yet pragmatic interventions, the TransFuser++ baseline was transformed into a robust, state-of-the-art system. At present, it leads across **all available** CARLA Leaderboard 2.0 benchmarks. Beyond performance improvements, this work revealed two key structural limitations in current end-to-end driving research.

First, restricted navigation capabilities arise from both the model and the infrastructure itself. Our TransFuser++ implementation partially solved the first issue by conditioning the planner on local routing structure and significantly improved the downstream driving performance. As for infrastructure, many navigation failures are amplified by the simulator's limited route guidance and rigorous route deviation penalty. These kinds of system-level issues complicate reliable evaluation and fair comparison between models, highlighting the need for new evaluation metrics and protocols. In particular, a protocol that allows but penalizes route deviation while still giving the model a chance to get sent back to the route and continue its drive.

Second, limited recovery capability is an inherent shortcoming of current imitation learning models. Our experiments show that removing the target-point bias, a step necessary for improving driving score, reduces its ability to recover once deviations occur. This exposes an inherent trade-off between unbiased navigation and recovery robustness. While sensory perturbations have been explored as an orthogonal approach to improve resilience, they are currently insufficient to handle larger distribution shifts. Addressing this limitation likely requires new forms of interactive or closed-loop training that explicitly exposes models to a larger deviation.

By openly addressing these issues and releasing a transparent infrastructure, our work aims to provide the community with a strong foundation for further research.

Bibliography

[1] autonomousvision/carla_route_generator.

[2] Bench2Drive - a Hugging Face Space by CarlaLeaderboard.

[3] Self-Driving Cars | Universität Tübingen.

[4] CARLA Autonomous Driving Leaderboard, Feb. 2020.

[5] autonomousvision/carla_garage, Oct. 2025.

[6] J. Beißwenger. Pdm-lite: A rule-based planner for carla leaderboard 2.0. https://github.com/OpenDriveLab/DriveLM/blob/DriveLM-CARLA/pdm$_l$*ite*/*docs*/*report.pdf*, 2024.

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to End Learning for Self-Driving Cars, Apr. 2016.

[8] W. Cao, M. Hallgarten, T. Li, D. Dauner, X. Gu, C. Wang, Y. Miron, M. Aiello, H. Li, I. Gilitschenski, B. Ivanovic, M. Pavone, A. Geiger, and K. Chitta. Pseudo-Simulation for Autonomous Driving, Aug. 2025.

[9] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-End Object Detection with Transformers, May 2020.

[10] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li. End-to-end Autonomous Driving: Challenges and Frontiers, Aug. 2024.

[11] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger. TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving, May 2022.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Dec. 2014.

[13] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end Driving via Conditional Imitation Learning, Mar. 2018.

[14] D. Dauner, M. Hallgarten, T. Li, X. Weng, Z. Huang, Z. Yang, H. Li, I. Gilitschenski, B. Ivanovic, M. Pavone, A. Geiger, and K. Chitta. NAVSIM: Data-Driven Non-Reactive Autonomous Vehicle Simulation and Benchmarking, Oct. 2024.

[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator, Nov. 2017.

[16] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. Qi, Y. Zhou, Z. Yang, A. Chouard, P. Sun, J. Ngiam, V. Vasudevan, A. McCauley, J. Shlens, and D. Anguelov. Large Scale Interactive Motion Forecasting for Autonomous Driving : The Waymo Open Motion Dataset, Apr. 2021.

[17] H. Fu, D. Zhang, Z. Zhao, J. Cui, D. Liang, C. Zhang, D. Zhang, H. Xie, B. Wang, and X. Bai. ORION: A Holistic End-to-End Autonomous Driving Framework by Vision-Language Instructed Action Generation, Mar. 2025.

[18] C. Gulino, J. Fu, W. Luo, G. Tucker, E. Bronstein, Y. Lu, J. Harb, X. Pan, Y. Wang, X. Chen, J. D. Co-Reyes, R. Agarwal, R. Roelofs, Y. Lu, N. Montali, P. Mougin, Z. Yang, B. White, A. Faust, R. McAllister, D. Anguelov, and B. Sapp. Waymax: An Accelerated, Data-Driven Simulator for Large-Scale Autonomous Driving Research, Oct. 2023.

[19] B. Jaeger. Expert drivers for autonomous driving. Master's thesis, University of Tübingen, 2021.

[20] B. Jaeger, K. Chitta, and A. Geiger. Hidden Biases of End-to-End Driving Models, Aug. 2023.

[21] X. Jia, Z. Yang, Q. Li, Z. Zhang, and J. Yan. Bench2Drive: Towards Multi-Ability Benchmarking of Closed-Loop End-To-End Autonomous Driving, Nov. 2024.

[22] F. Kong, Y. Li, W. Chen, C. Min, Y. Li, Z. Gao, H. Li, Z. Guo, and H. Sun. Vlr-driver: Large vision-language-reasoning models for embodied autonomous driving. In *Proceedings of the 2025 IEEE/CVF International Conference on Computer Vision (ICCV) – Poster Session*, 2025. Poster 2205, Exhibit Hall I #2491, Thu 23 Oct 5:45–7:45 p.m. PDT.

[23] Z. Li, S. Wang, S. Lan, Z. Yu, Z. Wu, and J. M. Alvarez. Hydra-NeXt: Robust Closed-Loop Driving with Open-Loop Training, July 2025.

[24] B. Liao, S. Chen, H. Yin, B. Jiang, C. Wang, S. Yan, X. Zhang, X. Li, Y. Zhang, Q. Zhang, and X. Wang. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving, 2025.

[25] H. Liu, T. Li, H. Yang, L. Chen, C. Wang, K. Guo, H. Tian, H. Li, H. Li, and C. Lv. Reinforced refinement with self-aware expansion for end-to-end autonomous driving, 2025.

[26] S. Liu, W. Chen, W. Li, Z. Wang, L. Yang, J. Huang, Y. Zhang, Z. Huang, Z. Cheng, and H. Yang. BridgeDrive: Diffusion Bridge Policy for Closed-Loop Trajectory Planning in Autonomous Driving, Sept. 2025.

[27] P. Mishra, S. Srivastava, J. Li, K. Bansal, and D. Bharadia. *Demo Abstract: C-Shenron: A Realistic Radar Simulation Framework for CARLA*, page 726–727. Association for Computing Machinery, New York, NY, USA, 2025.

[28] W. H. Organization. *Global status report on road safety 2023*. World Health Organization, 2023.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, Dec. 2019.

[30] D. A. Pomerleau. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[31] A. Prakash, K. Chitta, and A. Geiger. Multi-Modal Fusion Transformer for End-to-End Autonomous Driving, Apr. 2021.

[32] K. Renz, L. Chen, E. Arani, and O. Sinavski. SimLingo: Vision-Only Closed-Loop Autonomous Driving with Language-Action Alignment, Mar. 2025.

[33] S. I. I. Report. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems, sae standard j3016_201401. Information Report J3016_201401, SAE International, Jan. 2014.

[34] S. Ross, G. J. Gordon, and J. A. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, Mar. 2011.

[35] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu. Safety-Enhanced Autonomous Driving Using Interpretable Sensor Fusion Transformer, Dec. 2022.

[36] H. Shao, L. Wang, R. Chen, S. L. Waslander, H. Li, and Y. Liu. ReasonNet: End-to-End Driving with Temporal and Global Reasoning, May 2023.

[37] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, F. Massa, D. Haziza, L. Wehrstedt, J. Wang, T. Darcet, T. Moutakanni, L. Sentana, C. Roberts, A. Vedaldi, J. Tolan, J. Brandt, C. Couprie, J. Mairal, H. Jégou, P. Labatut, and P. Bojanowski. Dinov3, 2025.

[38] Y. Tang, Z. Xu, Z. Meng, and E. Cheng. HiP-AD: Hierarchical and Multi-Granularity Planning with Deformable Attention for Autonomous Driving in a Single Decoder, Mar. 2025.

[39] G. van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need, Aug. 2023.

[41] R. Xu, H. Lin, W. Jeon, H. Feng, Y. Zou, L. Sun, J. Gorman, K. Tolstaya, S. Tang, B. White, B. Sapp, M. Tan, J.-J. Hwang, and D. Anguelov. Wod-e2e: Waymo open dataset for end-to-end driving in challenging long-tail scenarios, 2025.

[42] H. Zhou, L. Lin, J. Wang, Y. Lu, D. Bai, B. Liu, Y. Wang, A. Geiger, and Y. Liao. Hugsim: A real-time, photo-realistic and closed-loop simulator for autonomous driving, 2024.

[43] Z. Zhou, T. Cai, S. Z. Zhao, Y. Zhang, Z. Huang, B. Zhou, and J. Ma. AutoVLA: A Vision-Language-Action Model for End-to-End Autonomous Driving with Adaptive Reasoning and Reinforcement Fine-Tuning, June 2025.

[44] J. Zimmerlin, J. Beißwenger, B. Jaeger, A. Geiger, and K. Chitta. Hidden Biases of End-to-End Driving Datasets, Dec. 2024.

Erklärung

Laut Beschlüssen der Prüfungsausschüsse Bioinformatik, Informatik, Informatik Lehramt, Kognitionswissenschaft, Machine Learning, Medieninformatik und Medizininformatik der Universität Tübingen vom 05.02.2025. Gültig für Abschlussarbeiten (B.Sc./M.Sc./B.Ed./M.Ed.) in den zugehörigen Fächern. Bei Studienarbeiten und Hausarbeiten bitte nach Maßgabe des/der jeweiligen Prüfers/Prüferin.

1. Allgemeine Erklärungen

Hiermit erkläre ich:

- Ich habe die vorgelegte Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt.
- Ich habe alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet.
- Die Arbeit war weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens.
- Falls ich ein elektronisches Exemplar und eines oder mehrere gedruckte und gebundene Exemplare eingereicht habe (z.B., weil der/die Prüfer/in(nen) dies wünschen): Das elektronisch eingereichte Exemplar stimmt exakt mit dem bzw. den von mir eingereichten gedruckten und gebundenen Exemplar(en) überein.

2. Erklärung bezüglich Veröffentlichungen

Eine Veröffentlichung ist häufig ein Qualitätsmerkmal (z.B. bei Veröffentlichung in Fachzeitschrift, Konferenz, Preprint, etc.). Sie muss aber korrekt angegeben werden. Bitte kreuzen Sie die für Ihre Arbeit zutreffende Variante an:

☐ Die Arbeit wurde bisher weder vollständig noch in Teilen veröffentlicht.

☐ Die Arbeit wurde in Teilen oder vollständig schon veröffentlicht. Hierfür findet sich im Anhang eine vollständige Tabelle mit bibliographischen Angaben.

3. Nutzung von Methoden der künstlichen Intelligenz (KI, z.B. chatGPT, DeepL, etc.)

Die Nutzung von KI kann sinnvoll sein. Sie muss aber korrekt angegeben werden und kann die Schwerpunkte bei der Bewertung der Arbeit beeinflussen. Bitte kreuzen Sie alle für Ihre Arbeit zutreffenden Varianten an und beachten Sie, dass die Varianten 3.4 - 3.6 eine vorherige Absprache mit dem/der Betreuer/in voraussetzen:

☐ 3.1. Keine Nutzung: Ich habe zur Erstellung meiner Arbeit keine KI benutzt.

☐ 3.2. Korrektur Rechtschreibung & Grammatik: Ich habe KI für Korrekturen der Rechtschreibung und Grammatik genutzt, ohne dass es dabei zu inhaltlich relevanter Textgeneration oder Übersetzungen kam. Das heißt, ich habe von mir verfasste Texte in derselben Sprache korrigieren lassen. Es handelt sich um rein sprachliche Korrekturen, sodass die von mir ursprünglich intendierte Bedeutung nicht wesentlich verändert oder erweitert wurde. Im Zweifelsfall habe ich mich mit meinem/r Betreuer/in besprochen. Alle genutzten Programme mit Versionsnummer sind im Anhang meiner Arbeit in einer Tabelle aufgelistet.

☐ 3.3. Unterstützung bei der Softwareentwicklung: Ich habe KI als Unterstützung beim Schreiben von Code in der Softwareentwicklung genutzt. Es handelt sich hierbei lediglich um Unterstützung und nicht um die automatische Generierung von größeren Programm-Teilen. Im Zweifelsfall habe ich mich mit meinem/r Betreuer/in besprochen. Alle genutzten Programme mit Versionsnummer sind im Anhang meiner Arbeit in einer Tabelle aufgelistet.

☐ 3.4. Übersetzung: Ich habe *nach vorheriger Absprache und mit Erlaubnis meines/r Betreuer/in* KI zur Übersetzung von mir in einer anderen Sprache geschriebenen Texte genutzt. Jede derartige Übersetzung ist im laufenden Text gekennzeichnet und der Anhang meiner Arbeit enthält eine Tabelle mit einem vollständigen Nachweis aller übersetzten Textstellen und der verwendeten Programme mit Versionsnummer.

☐ 3.5. Code-Generierung: Ich habe *nach vorheriger Absprache und mit Erlaubnis meines/r Betreuer/in* KI zur Erzeugung von Code in der Softwareentwicklung genutzt. Der Anhang meiner Arbeit enthält eine Tabelle mit einem vollständigen Nachweis aller derartigen Nutzungen, der verwendeten Programme mit Versionsnummer und der verwendeten Prompts.

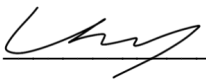
☐ 3.6. Text-Generierung: Ich habe *nach vorheriger Absprache und mit Erlaubnis meines/r Betreuer/in* KI zur Erzeugung von Text in meiner Arbeit genutzt. Jede derartige Verwendung von KI ist im laufenden Text gekennzeichnet und der Anhang meiner Arbeit enthält eine Tabelle mit einem vollständigen Nachweis aller derartigen Nutzungen, der verwendeten Programme mit Versionsnummer und der verwendeten Prompts.

Falls ich in irgendeiner Form KI genutzt haben (siehe oben), dann erkläre ich:

Mir ist bewusst, dass ich die Verantwortung trage, falls es durch die Verwendung von KI zu fehlerhaften Inhalten, zu Verstößen gegen das Datenschutzrecht, Urheberrecht oder zu wissenschaftlichem Fehlverhalten (z.B. Plagiaten) kommt.

4. Abschluss und Unterschrift(en)

Mir ist bekannt, dass ein Verstoß gegen diese Erklärung prüfungsrechtliche Konsequenzen haben und insbesondere dazu führen kann, dass die Prüfungsleistung mit „nicht ausreichend“ bzw. die Studienleistung mit „nicht bestanden“ bewertet wird und bei mehrfachem oder schwerwiegendem Täuschungsversuch eine Exmatrikulation erfolgen bzw. ein Verfahren zur Entziehung eines eventuell verliehenen akademischen Titels eingeleitet werden kann.

_____	_____	
Vorname, Nachname Student/in	Ort, Datum	Unterschrift

Die Punkte 3.4 - 3.6 erfordern eine Zustimmung des/r Betreuer/in. Sollten Sie einen dieser Punkte angekreuzt haben, dann sollte der/die Betreuer/in bitte hier unterschreiben:

Ich habe der oben genannten Nutzung von KI zur Erstellung der Arbeit zugestimmt.

_____	_____	_____
Vorname, Nachname Betreuer/in	Ort, Datum	Unterschrift

KI-System	**Anbieter**	**Version / Modell**	**Verwendungszweck**
ChatGPT (Pro)	OpenAI	GPT-5	Textüberarbeitung, Code
Claude	Anthropic	Claude 4.5 Sonnet	Textformulierung, Code

Table 6.1: Verwendete KI-Modelle im Rahmen der Thesis.