DISS. ETH NO. 30472

NEURAL POLICIES FOR PROSOCIAL NAVIGATION

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by

ZHEJUN ZHANG

M.Sc. in Electrical Engineering and Information Technology
ETH Zurich

born on 25.08.1994

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Marco Pavone, co-examiner
Prof. Dr. Igor Gilitschenski, co-examiner
Dr. Alexander Liniger, co-examiner

2024

Dedicated to my wife Lin.

Abstract

In order for an autonomous vehicle to navigate safely and prosocially in dense urban traffic, it needs to predict the potential motion of surrounding traffic participants and apply a policy that incorporates the predicted trajectories of other agents. As the complexity of this task scales up rapidly, learning-based methods become more preferable compared to traditional planners. In this thesis, we explore techniques for training neural policies for both autonomous vehicles and non-playable simulated traffic agents. Our goal is twofold: to enhance the generalizability of the autonomous driving policy through data, and to narrow the sim-to-real gap in terms of behavioral realism without adding excessive computational overhead.

Firstly, we introduce Reinforcement Learning Coach (Roach), a two-stage approach to end-to-end driving that combines reinforcement learning and imitation learning. In the first stage, we train a reinforcement learning expert that maps bird's-eye view images to continuous low-level actions. While setting a new performance upper-bound on the CARLA simulator, our expert also serves as a better coach, providing informative supervision signals for imitation learning agents to learn from. In the second stage, supervised by the Roach expert, a baseline end-to-end agent with monocular camera-input achieves expert-level performance while generalizing to a new map and new weather.

Secondly, we introduce a novel multiplicative value function for model-free reinforcement learning algorithms to alleviate the poor sample efficiency suffered by the Roach expert. The multiplicative value function consists of a safety critic and a reward critic. The safety critic predicts the probability of constraint violation and discounts the reward critic, which solely estimates constraint-free returns. Splitting responsibilities facilitates the learning task, resulting in increased sample efficiency when training Proximal Policy Optimization and Soft Actor Critic in safety-critic environments.

Thirdly, we address the issue of poor behavioral realism in existing simulators by training simulated agents to exhibit human-like behaviors using real-world data. We show data-driven traffic simulation can be formulated as a world model, and we present TrafficBots, a multi-agent policy built upon motion prediction and end-to-end driving. Based on TrafficBots, we develop a world model tailored for the planning module of autonomous vehicles. Compared to existing data-driven traffic simulators, TrafficBots demonstrates superior configurability and scalability.

Finally, we delve into the trade-off between the accuracy and efficiency of motion prediction and traffic simulation methods. We introduce the K-nearest neighbor attention with relative pose encoding (KNARPE), a novel attention mechanism allowing the pairwise-relative representation to be used by Transformers. Then, based on KNARPE we present the Heterogeneous Polyline Transformer with Relative pose encoding (HPTR), a hierarchical framework enabling asynchronous token update during the online inference. By sharing

contexts among agents and reusing the unchanged contexts, our approach is as efficient as scene-centric methods, while performing as accurate as state-of-the-art agent-centric methods. Hence, it is a superior architecture for motion prediction networks and neural policies for autonomous vehicles and robots.

Zusammenfassung

Um ein autonomes Fahrzeug sicher und prosozial im dichten städtischen Verkehr zu navigieren, muss es die potenzielle Bewegung der umliegenden Verkehrsteilnehmer voraussagen und eine Richtlinie anwenden, die die vorhergesagten Trajektorien anderer Agenten berücksichtigt. Da die Komplexität dieser Aufgabe rapide zunimmt, werden lernbasierte Methoden gegenüber traditionellen Planern bevorzugt. In dieser Arbeit untersuchen wir Techniken zur Schulung neuronaler Richtlinien für sowohl autonome Fahrzeuge als auch nicht-spielbare simulierten Verkehrsagenten. Unser Ziel ist zweigleisig: die Verallgemeinerbarkeit der autonomen Fahrtrichtlinie durch Daten zu verbessern und die Lücke zwischen Simulation und Realität in Bezug auf Verhaltensrealismus zu verringern, ohne dabei eine übermäßige Rechenlast hinzuzufügen.

Zunächst führen wir Reinforcement Learning Coach (Roach) ein, einen zweistufigen Ansatz für das End-to-End-Fahren, der Verstärkungslernen und Imitationslernen kombiniert. In der ersten Stufe trainieren wir einen Verstärkungslernexperten, der Vogelperspektivenbilder auf kontinuierliche, niedrigstufige Aktionen abbildet. Während er einen neuen Leistungsrekord auf dem CARLA-Simulator setzt, fungiert unser Experte auch als besserer Trainer, der informative Überwachungssignale für Imitationslernagenten liefert. In der zweiten Stufe erreicht ein Baseline-End-to-End-Agent mit monokularem Kameraeingang unter der Aufsicht des Roach-Experten Expertenleistung und generalisiert auf eine neue Karte und neues Wetter.

Zweitens führen wir eine neuartige multiplikative Wertefunktion für modellfreie Verstärkungslernalgorithmen ein, um die geringe Proben-Effizienz zu mildern, unter der der Roach-Experte leidet. Die multiplikative Wertefunktion besteht aus einem Sicherheitskritiker und einem Belohnungskritiker. Der Sicherheitskritiker sagt die Wahrscheinlichkeit einer Einschränkungsverletzung voraus und reduziert den Belohnungskritiker, der ausschließlich freiheitsbeschränkte Rückgaben schätzt. Die Aufteilung der Verantwortlichkeiten erleichtert die Lernaufgabe und führt zu einer erhöhten Proben-Effizienz beim Training von Proximal Policy Optimization und Soft Actor Critic in Sicherheitskritikumgebungen.

Drittens gehen wir das Problem des schlechten Verhaltensrealismus in bestehenden Simulatoren an, indem wir simulierte Agenten mit realen Daten trainieren, um menschenähnliches Verhalten zu zeigen. Wir zeigen, dass datengesteuerte Verkehrssimulation als Weltmodell formuliert werden kann, und stellen TrafficBots vor, eine Multi-Agenten-Richtlinie, die auf Bewegungsvorhersage und End-to-End-Fahren basiert. Basierend auf TrafficBots entwickeln wir ein Weltmodell, das speziell für das Planungsmodul autonomer Fahrzeuge zugeschnitten ist. Im Vergleich zu vorhandenen datengesteuerten Verkehrssimulatoren demonstriert TrafficBots eine überlegene Konfigurierbarkeit und Skalierbarkeit.

Schließlich gehen wir auf den Kompromiss zwischen Genauigkeit und Effizienz von Bewegungsvorhersage- und Verkehrssimulationsmethoden ein. Wir stellen das K-nearest-neighbor attention with relative pose encoding (KNARPE) vor, einen neuartigen Aufmerksamkeitsmechanismus, der es ermöglicht, die paarweise-relative Darstellung von Transformers zu verwenden. Basierend auf KNARPE präsentieren wir dann den Heterogeneous Polyline Transformer mit relativer Positions-Codierung (HPTR), ein hierarchisches Rahmenwerk, das eine asynchrone Token-Aktualisierung während der Online-Inferenz ermöglicht. Durch das Teilen von Kontexten zwischen Agenten und die Wiederverwendung der unveränderten Kontexte ist unser Ansatz so effizient wie szenezentrierte Methoden und dabei so genau wie agentenzentrische Methoden auf dem neuesten Stand der Technik. Daher ist es eine überlegene Architektur für Bewegungsvorhersagenetzwerke und neuronale Richtlinien für autonome Fahrzeuge und Roboter.

Publications

The following publications are included as a whole or in parts in this thesis:

- Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool. "End-to-end urban driving by imitating a reinforcement learning coach". In: *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*. 2021.
- N. Bührer, Z. Zhang, A. Liniger, F. Yu, and L. Van Gool. "A Multiplicative Value Function for Safe and Efficient Reinforcement Learning". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2023.
- Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. Van Gool. "Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.
- Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. Van Gool. "TrafficBots: Towards World Models for Autonomous Driving Simulation and Motion Prediction". In: *International Conference on Robotics and Automation (ICRA)*. 2023.

Acknowledgements

The research presented in this thesis is funded by Toyota Motor Europe via the research project TRACE-Zürich.

The last four years at CVL have been an enjoyable experience. Firstly, I would like to thank my supervisor, Prof. Dr. Luc Van Gool, for giving me the opportunity to start my PhD and the freedom during my PhD. I truly believe that research freedom and a positive working atmosphere are the most important aspects for a research team.

I am deeply grateful to Dr. Alexander Liniger, my second supervisor and the most significant contributor to my PhD, for his consistent guidance and feedback throughout the entire duration of my doctoral studies. Brainstorming with Alex is always so fruitful and inspiring. Additionally, I would like to thank my co-authors, Dr. Dengxin Dai, Dr. Christos Sakaridis, and Nick Bührer, for their contributions to my publications.

Next, I want to express my appreciation to my co-examiners, Prof. Dr. Marco Pavone and Prof. Dr. Igor Gilitschenski. It was fortunate for me to meet you at the conference, and it is an honor to have you as members of my PhD committee. I would like to thank Prof. Dr. Marco Pavone also for providing me with the opportunity to join his research team at NVIDIA for an internship. And I want to thank everyone from the industry whom I met at conferences, specifically Dr. Yuxiao Chen, Dr. Maximilian Igl, Dr. Boris Ivanovic, Dr. Peter Karkus, Dr. Sushant Veer, and Jonah Philion from NVIDIA; as well as Xieyuan Zhang, Dr. Long Chen, and Dr. Anthony Hu from Wayve. The conversations we had were so enjoyable and intriguing.

I want to express my gratitude to everyone at CVL. In particular, I appreciate my office mate Mengya Liu, Dr. Prune Truong, and Dr. Janis Postels for the limited but lovely dates we met in the office. Special thanks to TRACE members Dr. Jan-Nico Zaech, Dr. Ozan Ünal, Dr. Deng-Ping Fan, Dr. Simon Hecker, Dr. Anton Obukhov, Dr. Vaishakh Patil, and Dr. Martin Hahner for the unforgettable time we spent on Zoom and in Belgium for the update meetings. Lastly, I want to say thank you to Dr. Yigit Baran Can, Dr. Menelaos Kanakis, and Dr. David Brüggemann for the interesting and informative coffee talks.

I would also like to thank Yan Wu, Siyuan Li, Roy Yang, Mattia Segù, and our always-happy Italian friend Luigi Piccinelli for joining the CVL family and adding more fun to our experiences. And I want to pay my deepest tribute to Dr. Yifan Liu. I will never forget the sunny days in London when we attended ICRA together, and I will do my best to prevent the misfortune that happened to you from happening again to someone else.

Finally, I want to thank my wife, Dr. Lin Zhang, for her constant support over the past twelve years, starting from the first semester of our Bachelor's study at TU Munich back in 2012. And, of course, I have to give my biggest thanks to my four-legged furry friends, Pepino and Panini! I cannot imagine how I would have survived the depressive days during my PhD, especially during the pandemic, without them.

Contents

List of Figures

List of Tables

1

Introduction

Navigation in the proximity of humans and human-driven vehicles is one of the most fundamental tasks for large-scale deployment of autonomous vehicles (AV) and robots in our daily lives. Traditionally, the navigation task has been tackled using a modular pipeline that typically consists of four modules, arranged from upstream to downstream as follows:

- **Perception:** Generating detection or tracking results from sensor measurements, such as camera images and Lidar point clouds.
- **Prediction:** Forecasting the future motion of surrounding objects based on their historical trajectories and environmental contexts, such as a high-definition map.
- **Planning:** Generating the future trajectory for the ego agent given the outputs from perception and prediction modules.
- **Control:** Controlling the actuators of the ego agent to follow the planned trajectory.

In practice, the prediction and the control module can be simplified or even neglected, depending on the complexity of the operational design domain (ODD) and the mechanical dynamics of the ego vehicle or robot.

Over the past decade, the remarkable achievements of deep learning have led to an increased adoption of data-driven approaches across various segments of this pipeline. Notably, it has been observed that upstream modules extensively use big data and deep learning techniques, while downstream modules place a greater emphasis on physical models and manually crafted rules. In fact, today's perception modules are almost completely based on deep learning [1], whereas the control modules are predominantly built upon physics [2]. This phenomenon can be explained by the complexity of different tasks and the generalizability of different methods. Considering that the ODD for today's AV and robots encompasses unstructured and consistently changing outdoor environments, conventional model-based methods are incapable of addressing the perception challenge due to the lack of generalizability. On the contrary, addressing the control problem through a model-based approach is feasible because, in most cases, the robot and vehicle dynamics are well-defined, except when the ODD involves off-road navigation under extreme road conditions.

While this conclusion is evident for the perception module at the most upstream level and the control module at the most downstream level, the preference between data-driven and model-based methods remains controversial for the intermediate modules,

namely prediction and planning. In recent years, there has been a trend in transitioning from model-based to data-driven approaches in prediction and planning modules, or, more radically, replacing either the complete or part of the modular pipeline with an end-to-end (E2E) model [3, 4]. This shift is not only because of the success of deep learning in perception tasks, but also because the ODD of AV and robots has extended to environments involving interaction with humans, such as navigating through dense urban traffic. This task of prosocial navigation demands high generalizability and human-acceptable behavior from the prediction and planning algorithms, which can be more elegantly addressed through large-scale datasets and deep neural networks. This trend is also observed in the recent progress in the AV industry, where many companies have explored E2E approaches, such as those employed by Waymo [4, 5], NVIDIA [6, 7], Waabi [8], Tesla [9], and Wayve [10].

The scope of E2E driving is not restricted to a policy that maps sensor measurements to low-level actions. It usually also includes "mid-to-mid" methods, which map the output of perception module to motion trajectories. This relaxation is valid because the problem formulation remains unchanged: generating a motion plan for the ego agent based on past observations. Importantly, the learned model must function as a policy, i.e., the E2E model will be rolled out auto-regressively at inference time. Auto-regressive rollout means the current inputs to the policy can be influenced by its own outputs at previous time steps. This will lead to error accumulation due to covariate shift [11] if the policy is trained using an expert dataset in a supervised fashion, specifically through behavior cloning. This is because the learned policy is most likely imperfect, and the pre-collected expert demonstrations used for training cannot provide guidance on recovering from situations caused by this imperfection. One of the most promising ways to alleviate this problem is to use on-policy data, i.e., experiences collected by executing the policy, to refine the policy through imitation learning (IL) or reinforcement learning (RL). In IL, the on-policy data has to be annotated by an expert with the correct actions, whereas in RL, a reward is needed to quantify how well a policy performs. In any case, an environment is necessary for collecting the on-policy data. Since the policy is imperfect, executing it in the real world could be dangerous, especially when the environment involves humans in proximity. Therefore, it is preferable to use simulation as a substitute to train and test neural policies for AV and robots.

Policies trained and tested solely in simulation are prone to overfitting to the simulated environment. To effectively deploy such policies in the real world, it is essential to tackle two challenges: minimizing the sim-to-real gap inherent in the simulation and facilitating the sim-to-real transfer for the policy. In this thesis, our focus will be on the first challenge, as a high-fidelity simulator is not only valuable for training neural policies but also for testing all kinds of AV algorithms, including those that are not data-driven or E2E. For this challenge, there is also a trend towards employing more data-driven approaches, particularly in enhancing the photo-fidelity and behavior-fidelity of the simulator. In the past, achieving photo-fidelity depended on advanced game engines and skilled 3D artists [12], a process that was economically expensive and lacked scalability. Recently, however, numerous data-driven real-to-sim methods have been introduced to reconstruct real-world observations and synthesize sensor measurements [13], including camera images [14] and Lidar point clouds [15]. Similarly, behavior-fidelity used to rely on sophisticated rules heuristically designed by engineers [16]. This was not a challenging

task in the past when the ODD mainly consisted of scenarios with scarce interaction with humans, such as highways. However, when the environment expands to include densely populated urban scenarios, the complexity of manually crafting rules becomes overwhelming, making this approach infeasible. To address this problem, various data-driven traffic simulation methods [17, 18] have been proposed to generate human-like behavior for bot agents in the simulation through neural policies learned from real-world datasets.

In the following Section 1.1, we will review existing works on autonomous driving, with a focus on learning neural policies for the ego vehicle. Then in Section 1.2, we will review related works on traffic simulation, with a focus on learning neural policies for simulated traffic participants. Finally, in Section 1.3, we will present an outline of this thesis, which addresses several critical problems in learning neural policies for prosocial navigation.

1.1 Autonomous Driving

The first part of this thesis focuses on training a neural policy for the AV to navigate safely and prosocially in dense urban traffic. Following the E2E driving problem formulation, we define the input end of the policy to be either the sensor measurements or the intermediate representations generated by the perception module, and the output end to be either planned trajectories or actions for the ego vehicle. To train such a policy, we will leverage techniques from previous works on E2E driving, IL, and RL.

1.1.1 End-to-End Driving

Although the first E2E driving approach was proposed in the 1980s [19], most AV solutions deployed on public roads today still rely on a modular pipeline [2, 20, 21, 22]. However, current AV solutions do not perform well enough for massive deployment in cities due to the complexity and the long tail in urban driving scenarios [23, 24, 25]. Hence, there is a recent trend in the AV industry [4, 9, 10] and research [26, 27, 28, 29] to revisit data-driven E2E autonomous driving. Typical E2E driving methods directly map sensor measurements to low-level actions, thereby replacing the entire modular AV stack with a single deep neural network [3, 6, 30, 31, 32, 33, 34]. However, this approach demands a large amount of data and suffers from poor generalizability [35]. This problem can be addressed by using more structured representations for input and output, which means the E2E model now replaces only some parts of the modular pipeline [7, 36, 37, 38, 39, 40, 41]. We refer to [42, 43] for a comprehensive review of E2E driving.

1.1.2 Imitation Learning

Imitating the expert demonstrations is the most straight-forward approach to train E2E driving policies [3, 4, 30, 34]. However, training policies via supervised learning, specifically behavior cloning, often leads to covariate shift [11, 32], which can be partially mitigated through data augmentation techniques [6, 19, 44]. A more effective solution to

this problem is applying data aggregation (DAGGER) [11], which demands an interactive environment such as the CARLA simulator [12] and on-policy demonstrations from humans [45, 46] or automated experts [39, 40, 47, 48]. The automated experts can be either hand-crafted [32] or trained using IL [39] or RL [35]. In the absence of on-policy interactions with the environment, alternatives such as GAIL [49] and inverse RL [50] can be employed, although these methods tend to be less efficient and may not perform as well.

1.1.3 Reinforcement Learning

Motivated by the successes of model-free RL in diverse gaming scenarios [51, 52, 53, 54], numerous studies have extended model-free RL techniques [55, 56, 57, 58] to autonomous driving, both in simulated environments [12, 59, 60, 61, 62] and real-world settings [44, 45, 63, 64]. In [65], RL was combined with IL, while in [66] studied multi-agent RL is studied for E2E driving in simulation. A more detailed review of RL for AV can be found in [67].

Given that collision avoidance, a core function of AV, is modeled as constraint satisfaction within the RL framework, and considering that constraint violations in real-world scenarios may lead to catastrophic accidents, extensive research has focused on developing safe RL methodologies [68, 69, 70, 71]. These methodologies can be broadly classified into four categories: Lagrangian, primal, Lyapunov and intervention. The Lagrangian-based approach [68, 72, 73, 74, 75] is the most popular approach used in most of the RL for AV publications mentioned above. This approach converts the constrained problem into an unconstrained one by relaxing the safety constraint with a Lagrange multiplier. The other three approaches are less popular because they are more difficult to implement and their performance is not superior. The primal approach [76, 77, 78, 79, 80] computes a policy gradient that satisfies the constraints. The Lyapunov-based approach [81, 82, 83, 84] leverages Lyapunov functions, which are used in control theory to prove the stability of a dynamical system. The intervention approach [85, 86, 87, 88] uses backup policies to ensure safe actions. We refer to [89, 90] for an overview of the recent advances in safe RL.

1.2 Traffic Simulation

The second part of this thesis focuses on training a neural policy for bot agents in the simulation. To minimize the behavioral sim-to-real gap, simulated traffic participants surrounding the AV should exhibit human-like behavior, as observed in daily lives on public roads. To achieve that, we will leverage techniques from previous works on data-driven simulation, world models and motion prediction.

1.2.1 Data-Driven Simulation

Various data-driven approaches have been introduced to reduce the sim-to-real gap in perception, realized specifically through the synthesis of camera images [14, 44, 91, 92, 93], LiDAR point clouds [15, 94], and 3D assets [95, 96]. For the planning modules, however,

the sim-to-real gap in terms of the behavior of bot agents is more important. Compared to the hand-crafted rules [12, 97, 98], more realistic behaviors can be generated through log-replay [99, 100, 101, 102, 103, 104] or data-driven traffic simulation. The problem of learning realistic behaviors is formulated as generative adversarial IL [49] in [105, 106, 107, 108], as behavior cloning in [109], as flow prediction in [110], as diffusion model in [111], and as model-based IL in [18, 112]. While the behavioral realism of vehicles considers only the trajectories, the behavioral realism of pedestrians also includes animation [113]. In order to stress test the planner, methods like [114, 115] have been proposed to generate challenging driving scenarios. We refer to [116] for a more detailed review of data-driven traffic simulation.

1.2.2 World Models

While the aforementioned data-driven simulation approaches address only part of a full-stack simulator, a world model can be considered a fully data-driven E2E simulator. World models [117] are action-conditional dynamics models learned from observational data. As a differentiable substitute of the actual environment or a full-stack simulator, world models can be used for planning [118] and policy learning [119, 120]. Training world models is often formulated as a video prediction problem such that the method can generalize to all image-based environments, like Atari [121] and highway driving [122]. Recently, encouraged by the success of generative models, specifically diffusion models [123], for image [124, 125] and video [126, 127], numerous world models have been proposed for more challenging applications such as AV [128] and robots [129, 130, 131, 132]. A recent survey on world models for autonomous driving can be found in [133].

1.2.3 Motion Prediction

Data-driven traffic simulation often relies on the network architecture of motion prediction methods. In fact, traffic simulation can be considered as a closed-loop counterpart of the open-loop motion prediction [18, 112] and the same datasets [134, 135, 136] are used for both tasks. However, while the problem formulation of motion prediction could be either marginal for individual agent [137] or joint for all agents in the same scene [137, 138, 139], traffic simulation is always joint [140]. In contrast to policy learning for AV, traffic simulation [140] and motion prediction [137] often require multi-modal outputs, which can be addressed using goal conditioning [141, 142, 143, 144] and conditional variational autoencoder [145, 146, 147, 148]. While previous motion prediction approaches [146, 149] used recurrent neural networks [148, 150, 151, 152] and rasterized images [147, 153, 154, 155] with convolutional neural networks [156, 157], better performance is achieved by recent methods using vectorized representations [158] as input and Transformers [159, 160, 161] as the network backbone.

The vectorized representation falls into three categories: agent-centric [158], scene-centric [162] and pairwise-relative [163]. The agent-centric representation is the most popular one because of its good performance [164, 165, 166, 167]. By transforming all inputs to the local coordinates of each target agent, it ensures rotation and translation invariance but at the cost of poor scalability as the number of target agents grows. The

scene-centric representation is the most efficient and scalable approach, because it shares the contexts among all target agents by transforming all inputs to a global coordinate [162]. However, it performs poorly due to its lack of rotation and translation invariance. The pairwise-relative representation combines the best of both worlds by introducing a pair of global pose and local attribute for each object [163, 168, 169]. Methods using this representation can achieve high accuracy and scalability at the same time [170, 171]. This advantage is crucial for large-scale data-driven simulation and onboard deployment of AV and robots.

1.3 Thesis Outline

In this thesis, we address several issues that arise while learning E2E policy for AV and learning multi-agent policies for traffic simulation.

In Chapter 2, we propose Roach [35], the RL coach, which maps bird's-eye view images to continuous actions, and we use Roach to supervise an E2E IL driving policy. Roach eliminates the need for human experts and outperforms previous automated experts. Moreover, it also serves as a better teacher, providing more informative supervision signals for the IL student policy to learn from. We validate the effectiveness of our approach in the CARLA simulator.

Then, in Chapter 3, we address a limitation of Roach: the poor sample efficiency during the training of the mid-to-end RL policy. Specifically, we investigate more stable and efficient methods to train model-free RL algorithms in safety-critical environments such as autonomous driving. To this end, we propose a novel multiplicative value function [71] comprising a safety critic, which predicts the probability of constraint violation, and a reward critic, which estimates only constraint-free returns. By splitting responsibilities, we facilitate the learning task, leading to increased sample efficiency when applying to Proximal Policy Optimization (PPO) [172] and Soft Actor Critic (SAC) [58] in safety-critical environments

In Chapter 4, we address another limitation of Roach: the unnatural behavior exhibited by simulated agents in the CARLA simulator we used for training and testing our algorithms. To this end, we introduce TrafficBots [112], a multi-agent nueral policy built upon motion prediction and E2E driving. Then we draw a parallel between data-driven traffic simulation and world models. Based on TrafficBots we obtain a world model tailored for the planning module of AV. To address the configurability and multi-modality of the simulation, TrafficBots assigns a destination and a personality to each agent. To ensure the scalability, TrafficBots use the scene-centric representation and a new scheme of positional encoding.

Finally, in Chapter 5, we address the limitation of TrafficBots, which is the poor performance caused by the usage of the scene-centric representation. To achieve that, we introduce the K-nearest neighbor attention with relative pose encoding (KNARPE), a novel attention mechanism allowing the pairwise-relative representation to be used by Transformers. Then based on KNARPE we present the Heterogeneous Polyline Transformer with Relative pose encoding (HPTR), a hierarchical framework enabling asynchronous token update during the online inference [170]. The new motion prediction architecture,

HPTR, performs on par with state-of-the-art agent-centric methods, while maintaining the efficiency and scalability of scene-centric methods. As a result, it is suitable for on-board and real-time applications, as well as large-scale traffic simulation.

2

End-to-End Urban Driving by Imitating a Reinforcement Learning Coach

End-to-end approaches to autonomous driving commonly rely on expert demonstrations. Although humans are good drivers, they are not good coaches for end-to-end algorithms that demand dense on-policy supervision. On the contrary, automated experts that leverage privileged information can efficiently generate large scale on-policy and off-policy demonstrations. However, existing automated experts for urban driving make heavy use of hand-crafted rules and perform suboptimally even on driving simulators, where ground-truth information is available. To address these issues, we train a reinforcement learning expert that maps bird's-eye view images to continuous low-level actions. While setting a new performance upper-bound on CARLA, our expert is also a better coach that provides informative supervision signals for imitation learning agents to learn from. Supervised by our reinforcement learning coach, a baseline end-to-end agent with monocular camera-input achieves expert-level performance. Our end-to-end agent achieves a 78% success rate while generalizing to a new town and new weather on the NoCrash-dense benchmark and state-of-the-art performance on the challenging public routes of the CARLA LeaderBoard.

2.1 Introduction

Even though nowadays, most autonomous driving (AD) stacks [173, 174] use individual modules for perception, planning and control, end-to-end approaches have been proposed since the 80's [19] and the success of deep learning brought them back into the research spotlight [6, 175]. Numerous works have studied different network architectures for this task [3, 4, 34], yet most of these approaches use supervised learning with expert demonstrations, which is known to suffer from covariate shift [11, 32]. While data augmentation based on view synthesis [6, 19, 44] can partially alleviate this issue, in this paper, we tackle the problem from the perspective of expert demonstrations.

This chapter was published as a conference article: Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, Luc Van Gool, "End-to-End Urban Driving by Imitating a Reinforcement Learning Coach", Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, doi: 10.1109/ICCV48922.2021.01494

Expert demonstrations are critical for end-to-end AD algorithms. While imitation learning (IL) methods directly mimic the experts' behavior [4, 30], reinforcement learning (RL) methods often use expert demonstrations to improve sample efficiency by pre-training part of the model via supervised learning [60, 64]. In general, expert demonstrations can be divided into two categories: *(i) Off-policy*, where the expert directly controls the system, and the state/observation distribution follows the expert. Off-policy data for AD includes, for example, public driving datasets [136, 176, 177]. *(ii) On-policy*, where the system is controlled by the desired agent and the expert "labels" the data. In this case, the state/observation distribution follows the agent, but expert demonstrations are accessible. On-policy data is fundamental to alleviate covariate shift as it allows the agent to learn from its own mistakes, which the expert in the off-policy data does not exhibit. However, collecting adequate on-policy demonstrations from humans is non-trivial. While trajectories and actions taken by the human expert can be directly recorded during off-policy data collection, labeling these targets given sensor measurements turns out to be a challenging task for humans. In practice, only sparse events like human interventions are recorded, which, due to the limited information it contains, is hard to use for training and better suited for RL [44, 45, 63] than for IL methods.

In this work we focus on automated experts, which in contrast to human experts can generate large-scale datasets with dense labels regardless of whether they are on-policy or off-policy. To achieve expert-level performance, automated experts may rely on exhaustive computations, expensive sensors or even ground truth information, so it is undesirable to deploy them directly. Even though some IL methods do not require on-policy labeling, such as GAIL [49] and inverse RL [50], these methods are not efficient in terms of on-policy interactions with the environment.

On the contrary, automated experts can reduce the expensive on-policy interactions. This allows IL to successfully apply automated experts to different aspects of AD. As a real-world example, Pan et al. [47] demonstrated end-to-end off-road racing with a monocular camera by imitating a model predictive control expert with access to expensive sensors. In the context of urban driving, [32] showed that a similar concept can be applied to the driving simulator CARLA [12]. Driving simulators are an ideal proving ground for such approaches since they are inherently safe and can provide ground truth states. However, there are two caveats. The first regards the "expert" in CARLA, commonly referred to as the Autopilot (or the roaming agent). The Autopilot has access to ground truth simulation states, but due to the use of hand-crafted rules, its driving skills are not comparable to a human expert's. Secondly, the supervision offered by most automated experts is not informative. In fact, the IL problem can be seen as a knowledge transfer problem and just learning from expert actions is inefficient.

To tackle both drawbacks and motivated by the success of model-free RL in Atari games [178] and continuous control [58], we propose Roach (RL coach), an RL expert that maps bird's-eye view (BEV) images to continuous actions (Fig. 2.1 bottom). After training from scratch for 10M steps, Roach sets the new performance upper-bound on CARLA by outperforming the Autopilot. We then train IL agents and investigate more effective training techniques when learning from our Roach expert. Given that Roach uses a neural network policy, it serves as a better coach for IL agents also based on neural networks. Roach offers numerous informative targets for IL agents to learn from, which go far beyond deterministic action provided by other experts. Here we demonstrate

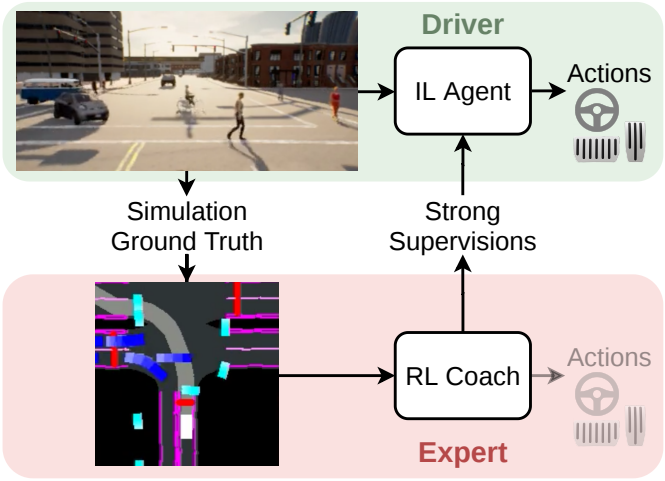


Figure 2.1: Roach: RL Coach allows IL agents to benefit from dense and informative on-policy supervisions.

the effectiveness of using action distributions, value estimations and latent features as supervisions.

Fig. 2.1 shows the scheme of learning from on-policy supervisions labeled by Roach on CARLA. We also record off-policy data from Roach by using its output to drive the vehicle on CARLA. Leveraging 3D detection algorithms [179, 180] and extra sensors to synthesize the BEV, Roach could also address the scarcity of on-policy supervisions in the real world. This is feasible because on the one hand, BEV as a strong abstraction reduces the sim-to-real gap [181], and on the other hand, on-policy labeling does not have to happen in real-time or even onboard. Hence 3D detection becomes easier given the complete sequences [182].

In summary, this paper presents Roach, an RL expert that sets a new performance upper-bound on CARLA. Moreover, we demonstrate the state-of-the-art performance on both the NoCrash benchmark and the public routes of CARLA LeaderBoard using a single camera based end-to-end IL agent, which is supervised by Roach using our improved training scheme. Our repository is available at `https://github.com/zhejz/carla-roach`

2.2 Related Work

Since our methods are trained and evaluated on CARLA, we mainly focus on related works also done on CARLA.

End-to-End IL: Dosovitskiy et al. [12] introduced the CARLA driving simulator and demonstrated that a baseline end-to-end IL method with single camera input can achieve a performance comparable to a modular pipeline. After that, CIL [30] and CILRS [31] addressed directional multi-modality in AD by using branched action heads where the branch is selected by a high-level directional command. While the aforementioned methods are trained via behavior cloning, DA-RB [32] applied DAGGER [11] with critical state sampling to CILRS. Most recently, LSD [33] increased the model capacity of CILRS by learning a mixture of experts and refining the mixture coefficients using evolutionary optimization. Here, we use DA-RB as the baseline IL agent to be supervised by Roach.

Mid-to-X IL: Directly mapping camera images to low-level actions requires a large amount of data, especially if one wants generalization to diverse weather conditions. Mid-to-X approaches alleviate this issue by using more structured intermediate representation as input and/or output. CILRS with coarse segmentation masks as input was studied in [36]. CAL [37] combines CIL and direct perception [38] by mapping camera images to driving affordances which can be directly used by a rule-based low-level controller. LBC [39] maps camera images to waypoints by mimicking a privileged mid-to-mid IL agent similar to Chauffeurnet [4], which takes BEV as input and outputs future waypoints. Similarly, SAM [40] trained a visuomotor agent by imitating a privileged CILRS agent that takes segmentation and affordances as inputs. Our Roach adopts BEV as the input representation and predicts continuous low-level actions.

RL: As the first RL agent on CARLA, an A3C agent [59] was demonstrated in [12], yet its performance is lower than that of other methods presented in the same paper. CIRL [60] proposed an end-to-end DDPG [55] agent with its actor network pre-trained via behavior cloning to accelerate online training. To reduce the problem complexity, Chen et al. [61] investigated DDQN [56], TD3 [57] and SAC [58] using BEV as an input and pre-trained the image encoder with a variational auto-encoder [183] on expert trajectories. State-of-the-art performance is achieved in [64] using Rainbow-IQN [62]. To reduce the number of trainable parameters during online training, its image encoder is pre-trained to predict segmentation and affordances on an off-policy dataset. IL was combined with RL in [65] and multi-agent RL on CARLA was discussed in [66]. In contrast to these RL methods, Roach achieves high sample efficiency without using any expert demonstrations.

IL with Automated Experts: The effectiveness of automated experts was demonstrated in [47] for real-world off-road racing, where a visuomotor agent is trained by imitating on-policy actions labeled by a model predictive control expert equipped with expensive sensors. Although CARLA already comes with the Autopilot, it is still beneficial to train a proxy expert based on deep neural networks, as shown by LBC [39] and SAM [40]. Through a proxy expert, the complex to solve end-to-end problem is decomposed into two simpler stages. At the first stage, training the proxy expert is made easier by formulating a mid-to-X IL problem that separates perception from planning. At the second stage, the end-to-end IL agent can learn more effectively from the proxy expert given the informative targets it supplies. To provide strong supervision signals, LBC queries all branches of the proxy expert and backpropagates all branches of the IL agent given one data sample, whereas SAM matches latent features of the proxy expert and the end-to-end IL agent. While the proxy expert addresses planning, it is also possible to address perception at the first stage as shown by FM-Net [48]. Overall, two-stage

approaches achieve better performance than direct IL, but using proxy experts inevitably lowers the performance upper-bound as a proxy expert trained via IL cannot outperform the expert it imitates. This is not a problem for Roach, which is trained via RL and outperforms the Autopilot.

2.3 Method

In this section we describe Roach and how IL agents can benefit from diverse supervisions supplied by Roach.

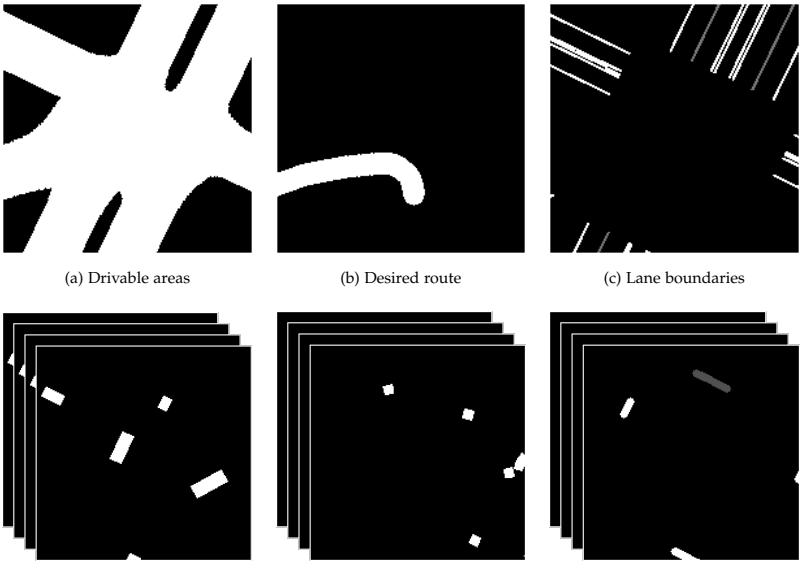
2.3.1 RL Coach

Our Roach has three features. Firstly, in contrast to previous RL agents, Roach does not depend on data from other experts. Secondly, unlike the rule-based Autopilot, Roach is end-to-end trainable, hence it can generalize to new scenarios with minor engineering efforts. Thirdly, it has a high sample efficiency. Using our proposed input/output representation and exploration loss, training Roach from scratch to achieve top expert performance on the six LeaderBoard maps takes less than a week on a single GPU machine.

Roach consists of a policy network $\pi_\theta(\mathbf{a}|\mathbf{i}_{\text{RL}}, \mathbf{m}_{\text{RL}})$ parameterized by θ and a value network $V_\phi(\mathbf{i}_{\text{RL}}, \mathbf{m}_{\text{RL}})$ parameterized by ϕ. The policy network maps a BEV image $\mathbf{i}_{\text{RL}}$ and a measurement vector $\mathbf{m}_{\text{RL}}$ to a distribution of actions $\mathbf{a}$. Finally the value network estimates a scalar value v, while taking the same inputs as the policy network.

Input Representation: We use a BEV semantic segmentation image $\mathbf{i}_{\text{RL}} \in [0,1]^{W\times H\times C}$ to reduce the problem complexity, similar to the one used in [4, 39, 61]. It is rendered using ground-truth simulation states and consists of C grayscale images of size $W \times H$. The ego-vehicle is heading upwards and is centered in all images at D pixels above the bottom, but it is not rendered. Fig. 2.2 illustrates each channel of $\mathbf{i}_{\text{RL}}$. Drivable areas and intended routes are rendered respectively in Fig. 2.2a and 2.2b. In Fig. 2.2c solid lines are white and broken lines are grey. Fig. 2.2d is a temporal sequence of K grayscale images in which cyclists and vehicles are rendered as white bounding boxes. Fig. 2.2e is the same as Fig. 2.2d but for pedestrians. Similarly, stop lines at traffic lights and trigger areas of stop signs are rendered in Fig. 2.2f. Red lights and stop signs are colored by the brightest level, yellow lights by an intermediate level and green lights by a darker level. A stop sign is rendered if it is active, i.e. the ego-vehicle enters its vicinity and disappears once the ego-vehicle has made a full stop. By letting the BEV representation memorize if the ego-vehicle has stopped, we can use a network architecture without recurrent structure and hence reduce the model size of Roach. A colored combination of all channels is visualized in Fig. 2.1. We also feed Roach a measurement vector $\mathbf{m}_{\text{RL}} \in \mathbb{R}^6$ containing the states of the ego-vehicle not represented in the BEV, these include ground-truth measurements of steering, throttle, brake, gear, lateral and horizontal speed.

Output Representation: Low-level actions of CARLA are *steering* $\in [-1,1]$, *throttle* $\in [0,1]$ and *brake* $\in [0,1]$. An effective way to reduce the problem complexity is predicting



(a) Drivable areas (b) Desired route (c) Lane boundaries

(d) Vehicles (e) Pedestrians (f) Lights and stops

Figure 2.2: The BEV representation used by our Roach.

waypoint plans which are then tracked by a PID-controller to produce low-level actions [39, 65]. However, a PID-controller is not reliable for trajectory tracking and requires excessive parameter tuning. A model-based controller would be a better solution, but CARLA's vehicle dynamics model is not directly accessible. To avoid parameter tuning and system identification, Roach directly predicts action distributions. Its action space is $\mathbf{a} \in [-1, 1]^2$ for steering and acceleration, where positive acceleration corresponds to throttle and negative corresponds to brake. To describe actions we use the Beta distribution $\mathcal{B}(\alpha, \beta)$, where $\alpha, \beta > 0$ are respectively the concentration on 1 and 0. Compared to the Gaussian distribution, which is commonly used in model-free RL, the support of the Beta distribution is bounded, thus avoiding clipping or squashing to enforce input constraints. This results in a better behaved learning problem since no tanh layers are needed and the entropy and KL-divergence can be computed explicitly. Further, the modality of the Beta distribution is also suited for driving, where extreme maneuvers may often be taken, for example, emergency braking or taking a sharp turn.

Training: We use proximal policy optimization (PPO) [172] with clipping to train the policy network π_θ and the value network V_ϕ. To update both networks, we collect trajectories by executing π_{θ_k} on CARLA. A trajectory $\tau = \{(\mathbf{i}_{\text{RL},k}, \mathbf{m}_{\text{RL},k}, \mathbf{a}_k, r_k)_{k=0}^{T}, z\}$ includes BEV images $\mathbf{i}_{\text{RL}}$, measurement vectors $\mathbf{m}_{\text{RL}}$, actions $\mathbf{a}$, rewards r and a terminal

event $z \in \mathcal{Z}$ that triggers the termination of an episode. The value network is trained to regress the expected returns, whereas the policy network is updated via

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{\tau \sim \pi_{\theta_k}} \left[\mathcal{L}_{\text{ppo}} + \mathcal{L}_{\text{ent}} + \mathcal{L}_{\text{exp}} \right] .$$

The first objective $\mathcal{L}_{\text{ppo}}$ is the clipped policy gradient loss with advantages estimated using generalized advantage estimation [184]. The second objective $\mathcal{L}_{\text{ent}}$ is a maximum entropy loss commonly employed to encourage exploration

$$\mathcal{L}_{\text{ent}} = -\lambda_{\text{ent}} \cdot \mathrm{H}\left(\pi_\theta(\cdot|\mathbf{i}_{\text{RL}}, \mathbf{m}_{\text{RL}})\right) .$$

Intuitively $\mathcal{L}_{\text{ent}}$ pushes the action distribution towards a uniform prior because maximizing entropy is equivalent to minimizing the KL-divergence to a uniform distribution,

$$\mathrm{H}\left(\pi_\theta\right) = -\mathrm{KL}\left(\pi_\theta \parallel \mathcal{U}(-1, 1)\right) ,$$

if both distributions share the same support. This inspires us to propose a generalized form of $\mathcal{L}_{\text{ent}}$, which encourages exploration in sensible directions that comply with basic traffic rules. We call it the exploration loss and define it as

$$\mathcal{L}_{\text{exp}} = \lambda_{\text{exp}} \cdot \mathbb{1}_{\{T-N_z+1,\ldots,T\}}(k) \cdot \mathrm{KL}(\pi_\theta(\cdot|\mathbf{i}_{\text{RL},k}, \mathbf{m}_{\text{RL},k}) \parallel p_z) ,$$

where $\mathbb{1}$ is the indicator function and $z \in \mathcal{Z}$ is the event that ends the episode. The terminal condition set $\mathcal{Z}$ includes collision, running traffic light/sign, route deviation and being blocked. Unlike $\mathcal{L}_{\text{ent}}$ which imposes a uniform prior on the actions at all time steps regardless of which z is triggered, $\mathcal{L}_{\text{exp}}$ shifts actions within the last N_z steps of an episode towards a predefined exploration prior p_z which encodes an "advice" to prevent the triggered event z from happening again. In practice, we use $N_z = 100, \forall z \in \mathcal{Z}$. If z is related to a collision or running traffic light/sign, we apply $p_z = \mathcal{B}(1, 2.5)$ on the acceleration to encourage Roach to slow down while the steering is unaffected. In contrast, if the car is blocked we use an acceleration prior $\mathcal{B}(2.5, 1)$. For route deviations, a uniform prior $\mathcal{B}(1, 1)$ is applied on the steering. Despite being equivalent to maximizing entropy in this case, the exploration loss further encourages exploration on steering angles during the last 10 seconds before the route deviation.

Implementation Details: Our implementation of PPO-clip is based on [185] and the network architecture is illustrated in Fig. 2.3a. We use six convolutional layers to encode the BEV and two fully-connected (FC) layers to encode the measurement vector. Outputs of both encoders are concatenated and then processed by another two FC layers to produce a latent feature $\mathbf{j}_{\text{RL}}$, which is then fed into a value head and a policy head, each with two FC hidden layers. Trajectories are collected from six CARLA servers at 10 FPS, each server corresponds to one of the six LeaderBoard maps. At the beginning of each episode, a pair of start and target location is randomly selected and the desired route is computed using the A* algorithm. Once the target is reached, a new random target will be chosen, hence the episode is endless unless one of the terminal conditions in $\mathcal{Z}$ is met. We use the reward of [62] and additionally penalize large steering changes to prevent oscillating maneuvers. To avoid infractions at high speed, we add an extra penalty proportional to the ego-vehicle's speed. More details are in the appendix.

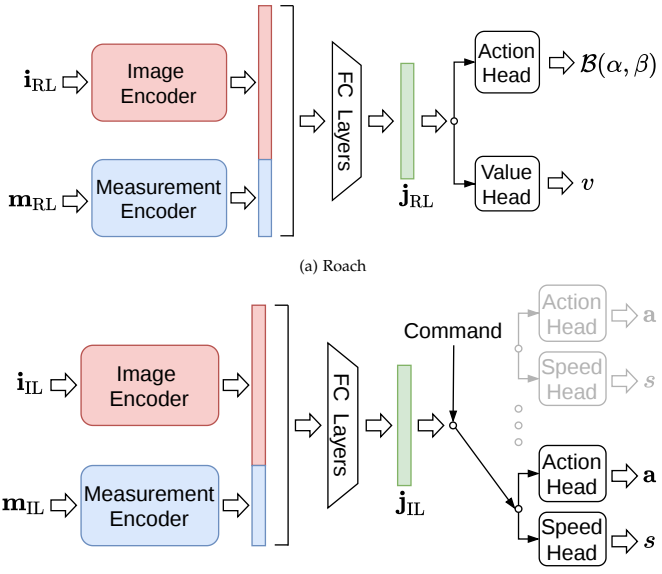


Figure 2.3: Network architecture of Roach, the RL expert, and CILRS, the IL agent.

2.3.2 IL Agents Supervised by Roach

To allow IL agents to benefit from the informative supervisions generated by Roach, we formulate a loss for each of the supervisions. Our training scheme using Roach can be applied to improve the performance of existing IL agents. Here we use DA-RB [32] (CILRS [31] + DAGGER [11]) as an example to demonstrate its effectiveness.

CILRS: The network architecture of CILRS is illustrated in Fig. 2.3b, it includes a perception module that encodes the camera image $\mathbf{i}_{\mathrm{IL}}$ and a measurement module that encodes the measurement vector $\mathbf{m}_{\mathrm{IL}}$. Outputs of both modules are concatenated and processed by FC layers to generate a bottleneck latent feature $\mathbf{j}_{\mathrm{IL}}$. Navigation instructions are given as discrete high-level commands and for each kind of command a branch is constructed. All branches share the same architecture, while each branch contains an action head predicting continuous actions $\mathbf{a}$ and a speed head predicting the current speed s of the ego-vehicle. The latent feature $\mathbf{j}_{\mathrm{IL}}$ is processed by the branch selected by the command. The imitation objective of CILRS consists of an L1 action loss

$$\mathcal{L}_{\mathrm{A}} = \|\hat{\mathbf{a}} - \mathbf{a}\|_1$$

and a speed prediction regularization

$$\mathcal{L}_{\mathrm{S}} = \lambda_{\mathrm{S}} \cdot |\hat{s} - s|\,,$$

where λ_s is a scalar weight, $\hat{\mathbf{a}}$ is the expert's action, $\hat{s}$ is the measured speed, $\mathbf{a}$ and s are action and speed predicted by CILRS. Expert actions $\hat{\mathbf{a}}$ may come from the Autopilot, which directly outputs deterministic actions, or from Roach, where the distribution mode is taken as the deterministic output. Besides deterministic actions, Roach also predicts action distributions, values and latent features. Next we will formulate a loss function for each of them.

Action Distribution Loss: Inspired by [186] which suggests soft targets may provide more information per sample than hard targets, we propose a new action loss based on the action distributions as a replacement for $\mathcal{L}_A$. The action head of CILRS is modified to predict distribution parameters and the loss is formulated as a KL-divergence

$$\mathcal{L}_K = \mathrm{KL}(\hat{\pi} \| \pi)$$

between the action distribution $\hat{\pi}$ predicted by the Roach expert and π predicted by the CILRS agent.

Feature Loss: Feature matching is an effective way to transfer knowledge between networks and its effectiveness in supervising IL driving agents is demonstrated in [40, 48]. The latent feature $\mathbf{j}_{RL}$ of Roach is a compact representation that contains essential information for driving as it can be mapped to expert actions using an action head consists of only two FC layers (cf. Fig. 2.3a). Moreover, $\mathbf{j}_{RL}$ is invariant to rendering and weather as Roach uses the BEV representation. Learning to embed camera images to the latent space of $\mathbf{j}_{RL}$ should help IL agents to generalize to new weather and new situations. Hence, we propose the feature loss

$$\mathcal{L}_F = \lambda_F \cdot \|\mathbf{j}_{RL} - \mathbf{j}_{IL}\|_2^2 .$$

Value Loss: Multi-task learning with driving-related side tasks could also boost the performance of end-to-end IL driving agents as shown in [175], which used scene segmentation as a side task. Intuitively, the value predicted by Roach contains driving relevant information because it estimates the expected future return, which relates to how dangerous a situation is. Therefore, we augment CILRS with a value head and regress value as a side task. The value loss is the mean squared error between $\hat{v}$, the value estimated by Roach, and v, the value predicted by CILRS,

$$\mathcal{L}_V = \lambda_V \cdot (\hat{v} - v)^2 .$$

Implementation Details: Our implementation follows DA-RB [32]. We choose a Resnet-34 pretrained on ImageNet as the image encoder to generate a 1000-dimensional feature given $\mathbf{i}_{RL} \in [0,1]^{900\times256\times3}$, a wide-angle camera image with a 100° horizontal FOV. Outputs of the image and the measurement encoder are concatenated and processed by three FC layers to generate $\mathbf{j}_{IL} \in \mathbb{R}^{256}$, which shares the same size as $\mathbf{j}_{RL}$. More details are found in the appendix.

2.4 Experiments

Benchmarks: All evaluations are completed on CARLA 0.9.11. We evaluate our methods on the NoCrash [31] and the offline LeaderBoard benchmark[1] [187]. Each benchmark specifies its training towns and weather, where the agent is allowed to collect data, and evaluates the agent in new towns and weather. The NoCrash benchmark considers generalization from Town 1, a European town composed of solely one-lane roads and T-junctions, to Town 2, a smaller version of Town 1 with different textures. By contrast, the LeaderBoard considers a more difficult generalization task in six maps that cover diverse traffic situations, including freeways, US-style junctions, roundabouts, stop signs, lane changing and merging. Following the NoCrash benchmark, we test generalization from four training weather types to two new weather types. But to save computational resources, only two out of the four training weather types are evaluated. The NoCrash benchmark comes with three levels of traffic density (empty, regular and dense), which defines the number of pedestrians and vehicles in each map. We focus on the NoCrash-dense and introduce a new level between regular and dense traffic, NoCrash-busy, to avoid congestion that often appears in the dense traffic setting. For the offline LeaderBoard the traffic density in each map is tuned to be comparable to the busy traffic setting.

Metrics: Our results are reported in success rate, the metric proposed by NoCrash, and driving score, a new metric introduced by the CARLA LeaderBoard. The success rate is the percentage of routes completed without collision or blockage. The driving score is defined as the product of route completion, the percentage of route distance completed, and infraction penalty, a discount factor that aggregates all triggered infractions. For example, if the agent ran two red lights in one route and the penalty coefficient for running one red light was 0.7, then the infraction penalty would be 0.7^2=0.49. Compared to the success rate, the driving score is a fine-grained metric that considers more kinds of infractions and it is better suited to evaluate long-distance routes. More details about the benchmarks and the complete results are found in the appendix.

2.4.1 Performance of Experts

We use CARLA 0.9.10.1 to train RL experts and fine-tune our Autopilot, yet all evaluations are still on 0.9.11.

Sample Efficiency: To improve the sample efficiency of PPO, we propose to use BEV instead of camera images, Beta instead of Gaussian distributions, and the exploration loss in addition to the entropy loss. Since the benefit of using a BEV representation is obvious, here we only ablate the Beta distribution and the exploration loss. As shown in Fig. 2.4, the baseline PPO with Gaussian distribution and entropy loss is trapped in a local minimum where staying still is the most rewarding strategy. Leveraging the exploration loss, PPO+exp can be successfully trained despite relatively high variance and low sample efficiency. The Beta distribution helps substantially, but without the exploration loss the training still collapsed in some cases due to insufficient exploration (cf. dashed blue line

1 In contrast to the Leaderboard online ranking, this benchmark is evaluated offline on the Leaderboard public routes (50 training, 26 testing).

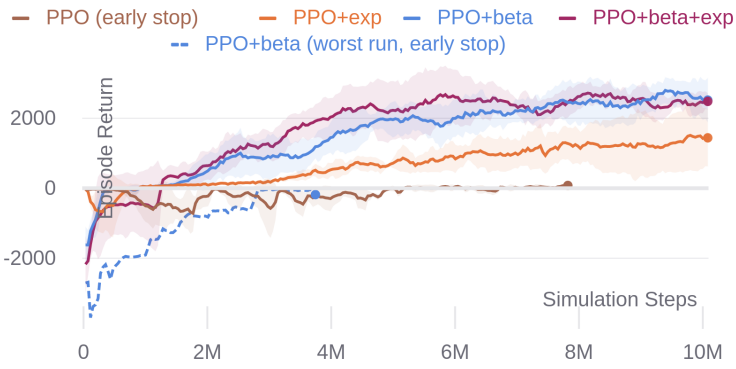


Figure 2.4: Learning curves of RL experts trained in CARLA Town 1-6. Solid lines show the mean and shaded areas show the standard deviation of episode returns across 3 seeds. The dashed line shows an outlier run that collapsed.

in Fig. 2.4). Our Roach (PPO+beta+exp) uses both Beta distribution and exploration loss to ensure stable and sample efficient training. The training takes around 1.7M steps in each of the six CARLA servers, this accounts for 10M steps in total, which takes roughly a week on an AWS EC2 g4dn.4xlarge or 4 days on a 2080 Ti machine with 12 cores.

Driving Performance: Table 2.1 compares different experts on the NoCrash-dense and on all 76 LeaderBoard routes under dynamic weather with busy traffic. Our Autopilot is a strong baseline expert that achieves a higher success rate than the Autopilot used in LBC and DA-RB. We evaluate three RL experts - (1) Roach, the proposed RL coach using Beta distribution and exploration prior. (2) PPO+beta, the RL coach trained without using the exploration prior. (3) PPO+exp, the RL coach trained without using the Beta distribution. In general, our RL experts achieve comparable success rates and higher driving scores than Autopilots because RL experts handle traffic lights in a better way (cf. Table 2.3). The two Autopilots often run red lights because they drive over-conservatively and wait too long at the junction, thus missing the green light. Among RL experts, PPO+beta and Roach, the two RL experts using a Beta distribution, achieve the best performance, while the difference between both is not significant. PPO+exp performs slightly worse, but it still achieves better driving scores than our Autopilot.

2.4.2 Performance of IL Agents

The performance of an IL agent is limited by the performance of the expert it is imitating. If the expert performs poorly, it is not sensible to compare IL agents imitating that expert. As shown in Table 2.1, this issue is evident in the NoCrash new town with dense traffic, where Autopilots do not perform well. To ensure a high performance upper-bound and hence a fair comparison, we conduct ablation studies (Fig. 2.5 and Table 2.3) under the

Suc. Rate % ↑	NCd-tt	NCd-tn	NCd-nt	NCd-nn	LB-all
PPO+exp	86 ± 6	86 ± 6	79 ± 6	77 ± 5	67 ± 3
PPO+beta	$\mathbf{95} \pm 3$	$\mathbf{95} \pm 3$	83 ± 5	$\mathbf{87} \pm 6$	72 ± 5
Roach	91 ± 4	90 ± 7	$\mathbf{83} \pm 3$	83 ± 3	72 ± 6
AP (ours)	$\mathbf{95} \pm 3$	$\mathbf{95} \pm 3$	83 ± 5	81 ± 2	$\mathbf{75} \pm 8$
AP-lbc [39]	86 ± 3	83 ± 6	60 ± 3	59 ± 8	N/A
AP-darb [32]	71 ± 4	72 ± 3	41 ± 2	43 ± 2	N/A
Dri. Score % ↑	NCd-tt	NCd-tn	NCd-nt	NCd-nn	LB-all
PPO+exp	92 ± 2	92 ± 2	88 ± 3	86 ± 1	83 ± 0
PPO+beta	$\mathbf{98} \pm 2$	$\mathbf{98} \pm 2$	90 ± 3	$\mathbf{92} \pm 2$	$\mathbf{86} \pm 2$
Roach	95 ± 2	95 ± 3	$\mathbf{91} \pm 3$	90 ± 2	85 ± 3
AP (ours)	86 ± 2	86 ± 2	70 ± 2	70 ± 1	78 ± 3

Table 2.1: Success rate and driving score of experts. Mean and standard deviation over 3 evaluation seeds. NCd: NoCrash-dense. tt: train town & weather. tn: train town & new weather. nt: new town & train weather. nn: new town & weather. LB-all: all 76 routes of LeaderBoard with dynamic weather. AP: CARLA Autopilot. For RL experts the best checkpoint among all training seeds and runs is used.

busy traffic setting such that our Autopilot can achieve a driving score of 80% and a success rate of 90%. In order to compare with the state-of-the-art, the best model from the ablation studies is still evaluated on NoCrash with dense traffic in Table 2.2.

The input measurement vector $\mathbf{m}_{\text{IL}}$ is different for the NoCrash and for the LeaderBoard. For NoCrash, $\mathbf{m}_{\text{IL}}$ is just the speed. For the LeaderBoard, $\mathbf{m}_{\text{IL}}$ contains additionally a 2D vector pointing to the next desired waypoint. This vector is computed from noisy GPS measurements and the desired route is specified as sparse GPS locations. The LeaderBoard instruction suggests that it is used to disambiguate situations where the semantics of left and right are not clear due to the complexity of the considered map.

Ablation: Fig. 2.5 shows driving scores of experts and IL agents at each DAGGER iteration on NoCrash and offline LeaderBoard with busy traffic. The baseline $\mathcal{L}_{\text{A}}(\text{AP})$ is our implementation of DA-RB$^+$ supervised by our Autopilot. Given our improved Autopilot, it is expected that $\mathcal{L}_{\text{A}}(\text{AP})$ can achieve higher success rates than those reported in the DA-RB paper, but this is not observed in Table 2.2. The large performance gap between the Autopilot and $\mathcal{L}_{\text{A}}(\text{AP})$ (cf. Fig. 2.5), especially while generalizing to a new town and new weather, indicates the limitation of this baseline.

By replacing the Autopilot with Roach, $\mathcal{L}_{\text{A}}$ performs better overall than $\mathcal{L}_{\text{A}}(\text{AP})$. Further learning from the action distribution, $\mathcal{L}_{\text{K}}$ generalizes better than $\mathcal{L}_{\text{A}}$ on the NoCrash but not on the offline LeaderBoard. Feature matching only helps when $\mathbf{j}_{\text{IL}}$ is provided with the necessary information needed to reproduce $\mathbf{j}_{\text{RL}}$. In our case, $\mathbf{j}_{\text{RL}}$ contains navigational information as the desired route is rendered in the BEV input. For the LeaderBoard,

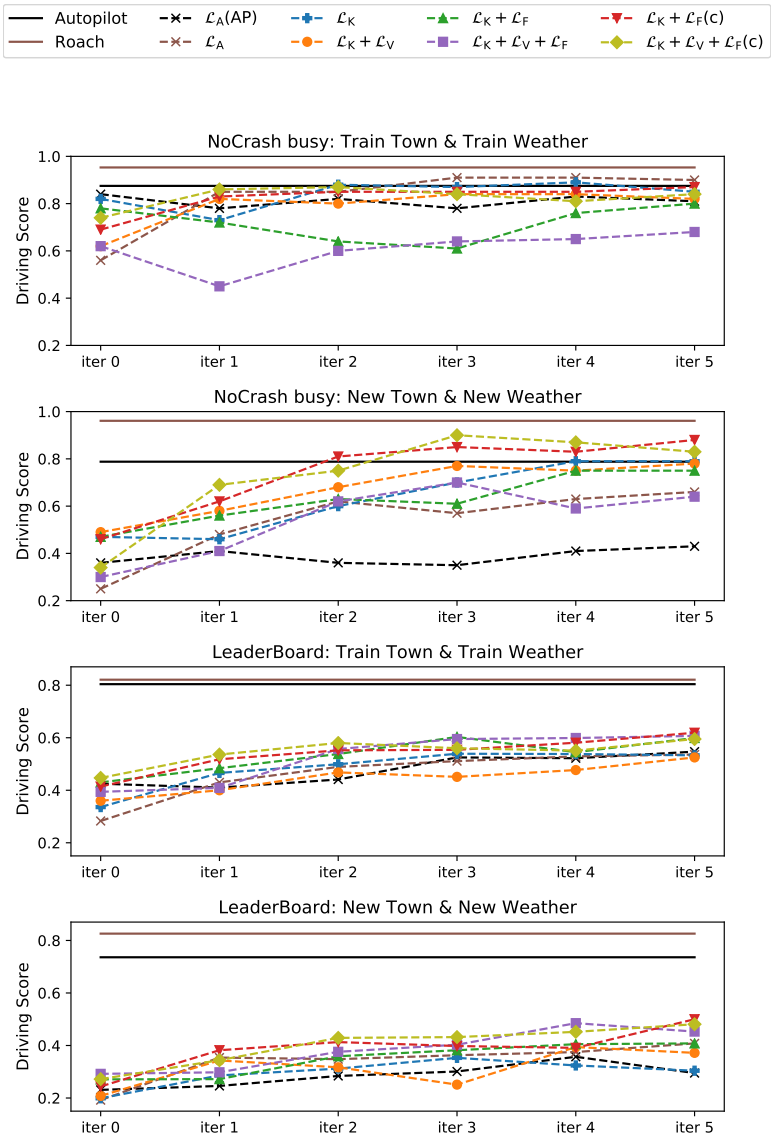


Figure 2.5: Driving score of experts and IL agents. All IL agents (dashed lines) are supervised by Roach except for $\mathcal{L}_A(\text{AP})$, which is supervised by our Autopilot. For IL agents at the 5th iteration on NoCrash and all experts, results are reported as the mean over 3 evaluation seeds. Others are evaluated with one seed. The offline Leaderboard benchmark is used here.

Success Rate % ↑	NCd-tt	NCd-tn	NCd-nt	NCd-nn
LBC [39] (0.9.6)	71 ± 5	63 ± 3	51 ± 3	39 ± 6
SAM [40] (0.8.4)	54 ± 3	47 ± 5	29 ± 3	29 ± 2
LSD [33] (0.8.4)	N/A	N/A	30 ± 4	32 ± 3
DA-RB$^+$(E) [32]	66 ± 5	56 ± 1	36 ± 3	35 ± 2
DA-RB$^+$ [32] (0.8.4)	62 ± 1	60 ± 1	34 ± 2	25 ± 1
Our baseline, $\mathcal{L}_A$(AP)	$\mathbf{88} \pm 4$	29 ± 3	32 ± 11	28 ± 4
Our best, $\mathcal{L}_K + \mathcal{L}_F$(c)	86 ± 5	$\mathbf{82} \pm 2$	$\mathbf{78} \pm 5$	$\mathbf{78} \pm 0$

Table 2.2: Success rate of camera-based end-to-end IL agents on NoCrash-dense. Mean and standard deviation over 3 seeds. Our models are from DAGGER iteration 5. For DA-RB, + means triangular perturbations are added to the off-policy dataset, (E) means ensemble of all iterations.

	Success rate	Driving score	Route compl	Infrac. penalty
iter 5	%, ↑	%, ↑	%, ↑	%, ↑
$\mathcal{L}_A$(AP)	31 ± 7	43 ± 2	62 ± 6	77 ± 4
$\mathcal{L}_A$	57 ± 7	66 ± 3	84 ± 3	76 ± 1
$\mathcal{L}_K$	74 ± 3	79 ± 0	91 ± 2	86 ± 1
$\mathcal{L}_K + \mathcal{L}_F$(c)	$\mathbf{87} \pm 5$	$\mathbf{88} \pm 3$	$\mathbf{96} \pm 0$	$\mathbf{91} \pm 3$
Roach	95 ± 2	96 ± 3	100 ± 0	96 ± 3
Autopilot	91 ± 1	79 ± 2	98 ± 1	80 ± 2

	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Agent blocked
iter 5	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
$\mathcal{L}_A$(AP)	0.54 ± 0.53	$\mathbf{0} \pm 0$	0.63 ± 0.50	3.33 ± 0.58	19.4 ± 14.4
$\mathcal{L}_A$	2.07 ± 1.37	$\mathbf{0} \pm 0$	1.36 ± 1.10	1.4 ± 0.2	2.82 ± 1.45
$\mathcal{L}_K$	0.50 ± 0.25	$\mathbf{0} \pm 0$	0.53 ± 0.18	0.68 ± 0.08	3.39 ± 0.20
$\mathcal{L}_K + \mathcal{L}_F$(c)	$\mathbf{0.08} \pm 0.04$	0.01 ± 0.02	$\mathbf{0.23} \pm 0.08$	$\mathbf{0.61} \pm 0.23$	$\mathbf{0.84} \pm 0.04$
Roach	0 ± 0	0.11 ± 0.07	0.04 ± 0.05	0.16 ± 0.20	0 ± 0
Autopilot	0 ± 0	0 ± 0	0.18 ± 0.08	1.93 ± 0.23	0.18 ± 0.08

Table 2.3: Driving performance and infraction analysis of IL agents on NoCrash-busy, new town & new weather. Mean and standard deviation over 3 evaluation seeds.

navigational information is partially encoded in $\mathbf{m}_{\text{IL}}$, which includes the vector to the next desired waypoint, so better performance is observed by using $\mathcal{L}_{\text{F}}$. But for NoCrash this information is missing as $\mathbf{m}_{\text{IL}}$ is just the speed, hence it is impractical for $\mathbf{j}_{\text{IL}}$ to mimic $\mathbf{j}_{\text{RL}}$ and this causes the inferior performance of $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{F}}$ and $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{F}} + \mathcal{L}_{\text{V}}$. To confirm this hypothesis, we evaluate a single-branch network architecture where the measurement vector $\mathbf{m}_{\text{IL}}$ is augmented by the command encoded as a one-hot vector. Using feature matching with this architecture, $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{F}}(\text{c})$ and $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{V}} + \mathcal{L}_{\text{F}}(\text{c})$ achieve the best driving score among IL agents in the NoCrash new town & weather generalization test, even outperforming the Autopilot.

Using value supervision in addition to feature matching helps the DAGGER process to converge faster as shown by $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{V}} + \mathcal{L}_{\text{F}}$ and $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{V}} + \mathcal{L}_{\text{F}}(\text{c})$. However, without feature matching, using value supervision alone $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{V}}$ does not demonstrate superior performance. This indicates a potential synergy between feature matching and value estimation. Intuitively, the latent feature of Roach encodes the information needed for value estimation, hence mimicking this feature should help to predict the value, while value estimation could help to regularize feature matching.

Comparison with the State-of-the-art: In Table 2.2 we compare the baseline $\mathcal{L}_{\text{A}}(\text{AP})$ and our best performing agent $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{F}}(\text{c})$ with the state-of-the-art on the NoCrash-dense benchmark. Our $\mathcal{L}_{\text{A}}(\text{AP})$ performs comparably to DA-RB^{+} except when generalizing to the new weather, where there is an incorrect rendering of after-rain puddles on CARLA 0.9.11 (see appendix for visualizations). This issue does not affect our best method $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{F}}(\text{c})$ due to the stronger supervision of Roach. By mimicking the weather-agnostic Roach, the performance of our IL agent drops by less than 10% while generalizing to the new town and weather. Hence if the Autopilot is considered the performance upper-bound, it is fair to claim our approach saturates the NoCrash benchmark. However, as shown in Fig. 2.5, there is still space for improvement on NoCrash compared to Roach and the performance gap on the offline LeaderBoard highlights the importance of this new benchmark.

Performance and Infraction Analysis: Table 2.3 provides the detailed performance and infraction analysis on the NoCrash benchmark with busy traffic in the new town & weather setting. Most notably, the extremely high "Agent blocked" of our baseline $\mathcal{L}_{\text{A}}(\text{AP})$ is due to reflections from after-rain puddles. This problem is largely alleviated by imitating Roach, which drives more naturally, and $\mathcal{L}_{\text{A}}$ shows an absolute improvement of 23% in terms of driving score. In other words this is the gain achieved by using a better expert, but the same imitation learning approach. Further using the improved supervision from soft targets and latent features results in our best model $\mathcal{L}_{\text{K}} + \mathcal{L}_{\text{F}}(\text{c})$, which demonstrates another 22% absolute improvement. By handling red lights in a better way, this agent achieves 88%, an expert-level driving score, using a single camera image as input.

2.5 Conclusion

We present Roach, an RL expert, and an effective way to imitate this expert. Using the BEV representation, Beta distribution and the exploration loss, Roach sets the new

performance upper-bound on CARLA while demonstrating high sample efficiency. To enable a more effective imitation, we propose to learn from soft targets, values and latent features generated by Roach. Supervised by these informative targets, a baseline end-to-end IL agent using a single camera image as input can achieve state-of-the-art performance, even reaching expert-level performance on the NoCrash-dense benchmark. Future works include performance improvement on simulation benchmarks and real-world deployment. To saturate the LeaderBoard, the model capacity shall be increased [4, 33, 188]. To apply Roach to label real-world on-policy data, several sim-to-real gaps have to be addressed besides the photorealism, which is partially alleviated by the BEV. For urban driving simulators, the realistic behavior of road users is of utmost importance [18].

Appendices

2.A Summary

In the appendix, we provide (1) an overview of supplementary videos and codes, (2) implementation details of the RL experts and the IL agents, (3) details regarding benchmarks, and (4) additional experimental results.

2.B Other Supplementary Materials

2.B.1 Videos

To investigate how different agents actually drive, we provide three videos. **roach.mp4** shows the driving performance of Roach, and highlights that it has a natural driving style and that it can handle complex traffic scenes. In **autopilot.mp4** we demonstrate the rule-based CARLA Autopilot. This agent uses unnatural brake actuation, i.e. it only uses emergency braking. Further, this video also highlights that in dense traffic, the rule-based agent can get stuck due to conservative danger predictions. For more details about the Autopilot and changes we made see Section 2.C.3. Finally, in **il_agent.mp4** we demonstrate our best roach-supervised IL agent, showing that the agent can handle complex traffic scenes but also highlighting failure cases. In detail:

- **roach.mp4** is an *uncut* evaluation run recorded from Roach driving in Town03 (LeaderBoard-busy under dynamic weather). This video demonstrates the natural driving style of Roach even in challenging situations such as US-style traffic lights, unprotected left turns, roundabouts and stop signs.
- **autopilot.mp4** is an *uncut* evaluation run recorded from Autopilot driving in Town02 (NoCrash-dense, new town & new weather). This video demonstrates the over-conservative behavior of the Autopilot while driving in dense traffic. This often leads to red light infractions and blockage (both are present in the video).
- **il_agent.mp4** is a *highlight* video recorded from our best roach-supervised IL agent $\mathcal{L}_{\mathrm{K}} + \mathcal{L}_{\mathrm{F}}(\mathrm{c})$. This video includes multiple challenging situations often encountered during urban driving, such as EU and US-style junctions, unprotected left turns, roundabouts and reacting to pedestrians walking into the street. Furthermore, we highlight some of the failure modes of our camera-based IL agent, including not coming to a full stop for stop signs, collisions at overcrowded intersections and oscillation in the steering if the lane markings are not visible due to sun glare. We believe that including memory in the IL agent policy can help in most of these issues, due to a better understanding of the ego-motion (stop sign and oscillations) and other agents' motion (collisions).

2.B.2 Code

To reproduce our results, we provide four python scripts:

- *train_rl.py* for training Roach.
- *train_il.py* for training DA-RB (CILRS + DAGGER).
- *benchmark.py* for benchmarking agents.
- *data_collect.py* for collecting on/off-policy data.

It is recommended to run our scripts through bash files contained in the folder *run*. All configurations are in the folder *config*. Our repository is composed of two modules:

- *carla_gym*, a versatile OpenAI gym [189] environment for CARLA. It allows not only RL training with synchronized rollouts, but also data collection and evaluation. The environment is configurable in terms of weather, number of background pedestrians and vehicles, benchmarks, terminal conditions, sensors, rewards for the ego-vehicle and etc.
- *agents*, which includes our implementation of Autopilot (in *agents/expert*), Roach (in *agents/rl_birdview*) and DA-RB (in *agents/cilrs*).

2.B.3 Rendering issues

As illustrated in Fig. 2.6, on CARLA 0.9.11 reflections from after-rain puddles are sometimes wrongly rendered as black pixels. When the black pixels are accumulated, for example in the middle of Fig. 2.6a, they are often recognized as obstacles by the camera-based agents. Since this kind of reflection only appears under the testing weather but not under the training weather, generalizing to testing weather is exceptionally hard on CARLA 0.9.11 for the camera-based end-to-end IL agents.

2.C Implementation Details

2.C.1 Roach

The network architecture of Roach can be found in Table 2.4 and the hyper-parameter values are listed in Table 2.5.

BEV: Cyclists and pedestrians are rendered larger than their actual sizes, this allows us to use a smaller image encoder with less parameters for Roach. Additionally, increasing the size naturally adds some caution when dealing with these vulnerable road users.

Update: The policy network and the value network are updated together using one Adam optimizer with an initial learning rate of 1e-5. The learning rate is scheduled based on the empirical KL-divergence between the policy before and after the update. If the KL-divergence is too large after an update epoch, the update phase will be interrupted

(a) Reflections from after-rain puddles in front of the ego-vehicle are incorrectly rendered as black pixels.

(b) Reflections are correctly rendered if the puddle is not directly in front of the ego-vehicle.

Figure 2.6: Rendering issue of CARLA 0.9.11 running on Ubuntu with OpenGL.

and a new rollout phase will start. Furthermore, a patience counter will be increased by one and the learning rate will be reduced once the patience counter reaches a threshold.

Rollout: Before each update phase a fixed-size buffer will be filled with trajectories collected on six CARLA servers, each corresponds to one of the six LeaderBoard maps (Town1-6).

Terminal Condition: An episode is terminated if and only if one of the following event happens.

- Run red light: examination code taken from the public repository of LeaderBoard. Terminal reward: $-1 - s$.
- Run stop sign: examination code taken from the public repository of LeaderBoard. Terminal reward: $-1 - s$.
- Collision registered by CARLA: based on the physics engine. Any collision with intensity larger than 0 is considered. Terminal reward: $-1 - s$.
- Collision detected by bounding box overlapping in the BEV. Terminal reward: $-1 - s$.
- Route deviation: triggered if the lateral distance to the lane centerline of the desired route is larger than 3.5 meters. Terminal reward: -1.
- Blocked: speed of the ego-vehicle is slower than 0.1 m/s for more than 90 consecutive seconds. Terminal reward: -1.

with s is the ego-vehicle's speed. The terminal reward is the reward given to the very last observation/action pair before the termination. For non-terminal samples, the terminal reward is 0.

Reward Shaping: The reward is the sum of the following components.

- r_speed: equals to $1.0 - |s - s_{\text{desired}}|/s_{\text{max}}$, where s is the measured speed of the ego-vehicle, s_{max} is the maximum speed and s_{desired} is the desired speed. We use a constant maximum speed $s_{\text{max}} = 6$ m/s. The desired speed is a variable and is explained below.
- r_position: equals to $-0.5\Delta_{\text{p}}$, where Δ_{p} is the lateral distance (in meters) between the ego-vehicle's center and the center line of the desired route.
- r_rotation: equals to $-\Delta_{\text{r}}$, where Δ_{r} is the absolute value of the angular difference (in radians) between the ego-vehicle's heading and the heading of the center line of the desired route.
- r_action: equals to -0.1 if the current steering differs more than 0.01 from the steering applied in the previous step.
- r_terminal: the aforementioned terminal reward.

The desired speed, as proposed in [62], depends on rule-based obstacle detections. If there's no obstacle detected, the desired speed equals to the maximum speed. If an obstacle is detected, based on the distance to the obstacle the desired speed is linearly decreased to 0. As obstacle detector we use the hazard detection of Autopilot (cf. Section 2.C.3). As a dense and informative reward, r_speed helps substantially to train our Roach and the camera-based end-to-end RL agent [62]. However, using rule-based obstacle detections inevitable introduces bias, the trained RL agent can be over-aggressive or over-conservative depending on the false positive and false negative rate of the detector. For example, during multi-lane freeway driving, our Roach decelerates for vehicles on the neighboring lanes because those vehicles are detected as obstacles during training. Another example, Roach tends to collide after a right turn, this is related to the sector shaped (around 40 degrees) detection area used by the obstacle detection; vehicles and pedestrians on the right are not covered in the detection area. To further improve the performance of Roach, this r_speed should be modified, either using a better obstacle detector, or completely remove the rule-based obstacle detection, and build a less artificial reward based on simulation states.

Mode of Beta Distribution: We take the distribution mode as a deterministic output. The mode of the Beta distribution $\mathcal{B}(\alpha, \beta)$ is defined as

$$\text{M} = \begin{cases} \frac{\alpha-1}{\alpha+\beta-2} & \text{if } \alpha > 1, \beta > 1 \\ 0 & \text{if } \alpha \leq 1, \beta > 1 \\ 1 & \text{if } \alpha > 1, \beta \leq 1 \\ \text{bimodal } \{0,1\} & \text{if } \alpha < 1, \beta < 1 \\ \text{any value in } [0,1] & \text{if } \alpha = 1, \beta = 1 \end{cases}$$

For a natural driving behavior, we use the mean $\frac{\alpha}{\alpha+\beta}$ as the deterministic output when the mode is not uniquely defined, i.e. when $\alpha < 1, \beta < 1$ or $\alpha = 1, \beta = 1$.

Layer Type	Filters	Size	Strides	Activation
		Image Encoder		
Conv2d	8	5x5	2	ReLU
Conv2d	16	5x5	2	ReLU
Conv2d	32	5x5	2	ReLU
Conv2d	64	3x3	2	ReLU
Conv2d	128	3x3	2	ReLU
Conv2d	256	3x3	1	-
Flatten				
		Measurement Encoder		
Dense	256			ReLU
Dense	256			ReLU
		FC Layers after Concatenation		
Dense	512			ReLU
Dense	256			ReLU
		Action Head		
Dense (shared)	256			ReLU
Dense (shared)	256			ReLU
Dense (for α)	2			Softplus
Dense (for β)	2			Softplus
		Value Head		
Dense	256			ReLU
Dense	256			ReLU
Dense	1			-

Table 2.4: The network architecture used for Roach. Around 1.53M trainable parameters.

Notation	Description	Value
	BEV Representation	
W	Width	192 px
H	Height	192 px
C	Number of channels	15
K	Size of the temporal sequence	4
	Timestamps of images in the temporal sequence	{-1.5, -1, -0.5, 0} sec
D	Distance from the ego-vehicle to the bottom	40 px
	Pixels per meter	5 px/m
	Minimum width/height of rendered bounding boxes	8 px
	Scale factor for bounding box size of pedestrians	2
	Rollout	
	Buffer size for six environments	12288 frames
	Value bootstrap for the last non-terminal sample	True
	Synchronized	True
	Reset at the beginning of a new phase	False
	Weather	dynamic
	Range of vehicle/pedestrian number in Town 1	$[0, 150] / [0, 300]$
	Range of vehicle/pedestrian number in Town 2	$[0, 100] / [0, 200]$
	Range of vehicle/pedestrian number in Town 3	$[0, 120] / [0, 120]$
	Range of vehicle/pedestrian number in Town 4	$[0, 160] / [0, 160]$
	Range of vehicle/pedestrian number in Town 5	$[0, 160] / [0, 160]$
	Range of vehicle/pedestrian number in Town 6	$[0, 160] / [0, 160]$
	Update	
	Number of epochs	20
λ_{ent}	Weight for the entropy loss	0.01
λ_{exp}	Weight for the exploration loss	0.05
	Weight for value loss	0.5
	γ for GAE	0.99
	λ for GAE	0.9
	Clipping range for PPO-clip	0.2
	Max norm for gradient clipping	0.5
	Batch size	256
	Initial learning rate	1e-5

KL-divergence threshold for learning rate schedule	0.15
Patience for learning rate schedule	8
Factor for learning rate schedule	0.5

Table 2.5: The hyper-parameter values used for Roach.

2.C.2 IL Agent Supervised by Roach

The network architecture of our IL agent is found in Table 2.6 and the hyper-parameter values are listed in Table 2.7.

Network Architecture: We use six branches: turning left, turning right and going straight at the junction, following lane, changing to the left lane and changing to the right lane.

Off-policy Data Collection: Following CILRS [31], triangular perturbations on actions are applied while collecting the off-policy expert dataset to alleviate the covariate shift. The off-policy dataset for NoCrash includes 80 episodes and for LeaderBoard it includes 160 episodes. Each episode is at most 300 seconds and at least 30 seconds long. The episode will be terminated if the expert violates any traffic rules, including red light infractions, stop sign infractions and collisions. In such a case, we remove the last 30 seconds of that episode so as to ensure that the off-policy dataset includes only correct demonstrations. Data is not collected using the given training routes but from randomly spawned start and target locations.

On-policy Data Collection: We follow DA-RB [32] for DAGGER with critical state sampling and replay buffer. New DAGGER-data will replace the old data in the replay buffer, while the buffer size is fixed. The same number of frames are contained in the replay buffer as in the off-policy dataset. At each DAGGER iteration, around 15-25% of the replay buffer is filled with new DAGGER-data, whereas at least 20% of the replay buffer is filled with off-policy data. Identical to the off-policy data collection, we use randomly spawned start and target locations while collecting DAGGER datasets. Following DA-RB, we did not use a mixed agent/expert policy to collect DAGGER datasets. However, our code allows this kind of rollout for DAGGER.

Training Details: Since we take the ResNet-34 pre-trained on ImageNet, the input image is normalized as suggested. In case the IL agent uses a distributional action head and/or a value head, the corresponding weights will be loaded from the Roach model at the first training iteration (the behavior cloning iteration). At each DAGGER iteration, the training continuous from the last epoch of the previous DAGGER iteration. We apply image augmentations using code modified from CILRS. The image augmentation methods are applied in random order and include Gaussian blur, additive Gaussian noise, coarse and block-wise dropouts, additive and multiplicative noise to each channel, randomized contrast and grayscale. All models are trained for 25 epochs using the ADAM optimizer with an initial learning rate of 2e-4. The learning rate is halved if the validation loss has not decreased for more than 5 epochs.

Layer Type	Filters	Activation	Dropout
Image Encoder			
ResNet-34			
Measurement Encoder			
Dense	128	ReLU	
Dense	128	ReLU	
FC Layers after concatenation			
Dense	512	ReLU	
Dense	512	ReLU	
Dense	256	ReLU	
Speed Head			
Dense	256	ReLU	
Dense	256	ReLU	0.5
Dense	1		
Value Head			
Dense	256	ReLU	
Dense	256	ReLU	0.5
Dense	1		
Deterministic Action Head			
Dense	256	ReLU	
Dense	256	ReLU	0.5
Dense	2		
Distributional Action Head			
Dense (shared)	256	ReLU	
Dense (shared)	256	ReLU	0.5
Dense (for α)	2	Softplus	
Dense (for β)	2	Softplus	

Table 2.6: The network architecture used for our IL agent. Around 23.4M trainable parameters.

Description	Value
Inputs	
Camera type	RGB
Camera image width	900 px
Camera image height	256 px
Camera $[x, y, z]$ relative to the ego-veh.	$[-1.5, 0, 2]$
Camera $[roll, pitch, yaw]$ relative to the ego-veh.	$[0, 0, 0]$
Camera horizontal FOV	100°
Mean for image normalization	$[0.485, 0.456, 0.406]$
Standard deviation for image normalization	$[0.229, 0.224, 0.225]$
Speed measurement	Forward speed in m/s
Normalization factor for speed	12
Data Collection	
Episode length	300 sec
Triangular perturbation for off-policy data	20%
# episodes (NoCrash, off-policy)	80
# episodes (LB, off-policy)	160
# episodes (NoCrash, on-policy, Autopilot)	80
# episodes (LB, on-policy, Autopilot)	160
# episodes (NoCrash, on-policy, Roach)	40
# episodes (LB, on-policy, Roach)	80
DA-RB critical state sampling criterion	difference in acceleration
DA-RB critical state sampling threshold	0.2
Weather	Same as NoCrash train weathers
Range of veh./ped. number in NoCrash train town 1	$[0, 150]$ / $[0, 200]$
Range of veh./ped. number in LB train town 1	$[80, 160]$ / $[80, 160]$
Range of veh./ped. number in LB train town 3	$[40, 100]$ / $[40, 100]$
Range of veh./ped. number in LB train town 4	$[100, 200]$ / $[40, 120]$
Range of veh./ped. number in LB train town 6	$[80, 160]$ / $[40, 120]$
Training	
Number of epochs at each DAGGER iteration	25
λ_S, weight for the speed regularization	0.05
λ_V, weight for the value loss, if applied	0.05
λ_F, weight for the feature loss, if applied	0.001

Batch size	48
Initial learning rate (LR)	0.0002
Patience for reduce-on-plateau LR schedule	5
Factor for LR schedule	0.5
Pre-trained distributional action head	True
Pre-trained value head	True
Image augmentation	True

Table 2.7: The hyper-parameter values used for our IL agent.

2.C.3 Autopilot

The CARLA Autopilot (also called roaming agent) is a simple but effective automated expert based on hand-crafted rules and ground-truth simulation states. The Autopilot is composed of two PID controllers for trajectory tracking and hazard detectors for emergency brake. Hazards include

- pedestrians/vehicles detected ahead,
- red lights/stop sings detected ahead,
- negative ego-vehicle speed, for handling slopes.

Locations and states of pedestrians, vehicles, red lights and stop signs are provided as ground-truth by the CARLA API. If any hazard appears in a trigger area ahead of the ego-vehicle, Autopilot will make an emergency brake with $\mathit{throttle} = 0$, $\mathit{steering} = 0$, $\mathit{brake} = 1$. If no hazard is detected, the ego-vehicle will follow the desired path using two PID controllers, one for speed and one for steering control. The PID controller takes as input the location, rotation and speed of the ego-vehicle and the desired route specified as dense (1 meter interval) waypoints. The speed PID yields $\mathit{throttle} \in [0, 1]$ and the steering PID yields $\mathit{steering} \in [-1, 1]$. We tuned the parameters for PID controllers and hazard detectors manually, such that the Autopilot is a strong baseline. The target speed is 6 m/s.

2.D Benchmarks

Scope: The scope of the NoCrash and the offline LeaderBoard benchmark are illustrated in Table 2.8. The offline LeaderBoard benchmark considers more traffic scenarios and longer routes in six different maps.

Weather: Following the NoCrash benchmark, we use *ClearNoon*, *WetNoon*, *HardRainNoon* and *ClearSunset* as the training weather types, whereas new weather types are *SoftRainSunset* and *WetSunset*. To save computational resources, only two out of the four training weather types are evaluated, they are *WetNoon* and *ClearSunset*.

Map	# Routes	Total Km	# Traffic lights	# Stop signs
		NoCrash Train		
Town01	25	17.4	110	0
		NoCrash Test		
Town02	25	8.9	94	0
		LeaderBoard Train		
Town01	10	7.9	47	0
Town03	20	30.7	140	63
Town04	10	24.1	72	13
Town06	10	19.5	58	1
		LeaderBoard Test		
Town02	6	5.5	54	0
Town04	10	24.1	72	13
Town05	10	12.4	82	29

Table 2.8: Scope of the NoCrash benchmark and the offline LeaderBoard benchmark. Total kilometers, number of traffic lights and stop signs are measured using Roach.

Map	# Vehicles	# Pedestrians
NoCrash dense		
Town01	100	250
Town02	70	150
NoCrash busy		
Town01	120	120
Town02	70	70
LeaderBoard busy		
Town01	120	120
Town02	70	70
Town03	70	70
Town04	150	80
Town05	120	120
Town06	120	80

Table 2.9: Background traffic settings for different benchmarks.

Background Traffic: The number of vehicles and pedestrians spawned in each map of different benchmarks are listed in Table 2.9. Vehicles and pedestrians are spawned randomly from the complete blueprint library of CARLA 0.9.11. This stands in contrast to several previous works where for example two-wheeled vehicles are disabled.

Pros and cons of the online and the offline Leaderboard:

Online Leaderboard:
(+) All methods are evaluated under exactly the same condition.
(+) No need to re-evaluate other methods.
(−) No restriction on how the method is trained and how the training data is collected.

Offline Leaderboard:
(+) Strictly prescribes both the training and testing environment.
(+) Full control and observation over the benchmark.
(−) You will have to re-evaluate other methods, if any setup of the benchmark has changed, for example CARLA version and etc.

One can use the offline Leaderboard if a thorough study on the generalization ability of the method is desired.

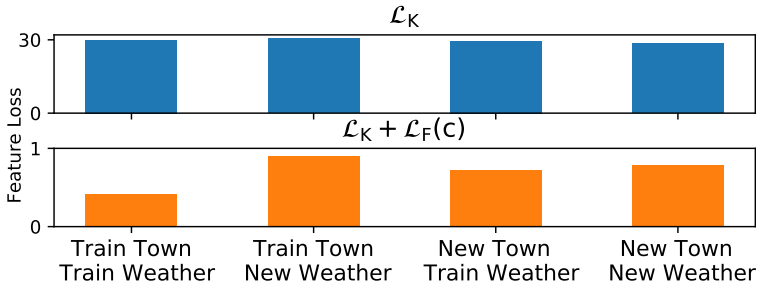
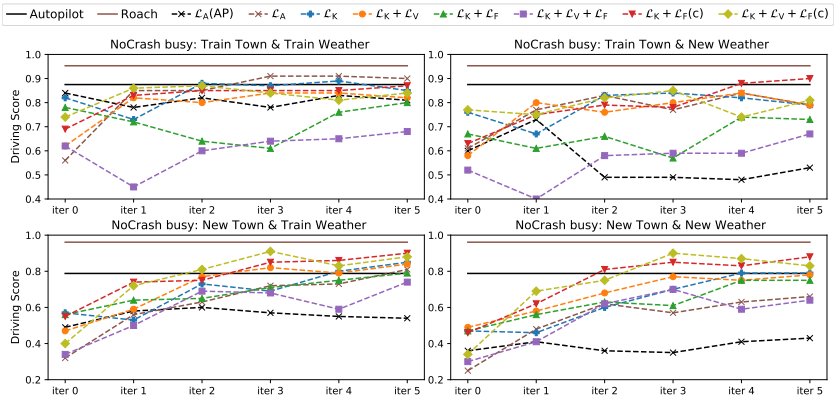


Figure 2.7: Feature loss w.r.t. Roach on one of the NoCrash-dense route. The y-axis of both charts have different scale.

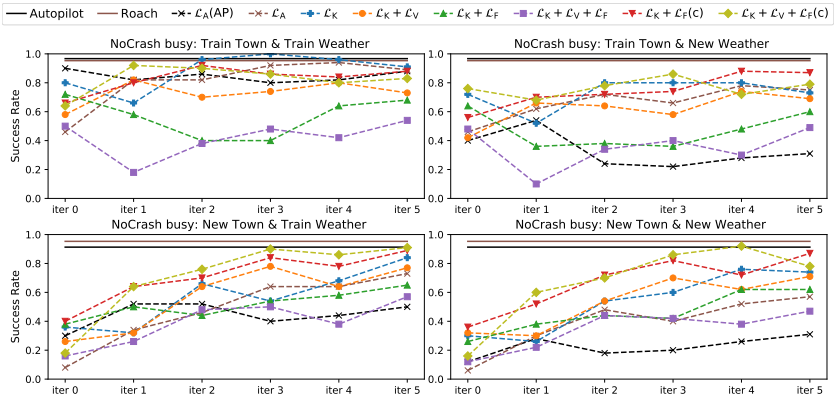
2.E Additional Experimental Results

To verify IL agents trained using the feature loss indeed embed camera images to the latent space of Roach, we report the feature loss at test time in Fig. 2.7. In the first row of Fig. 2.7, the IL agent trained without feature loss, $\mathcal{L}_K$, learns a latent space independent of the one of Roach. Hence, the test feature loss is effectively noise that is invariant to the test condition. In the second row, $\mathcal{L}_K + \mathcal{L}_F(c)$ is trained with the feature loss. The test feature loss of this agent is much smaller (less than 1) and increases as expected during the generalization tests.

To complete Fig. 5 of the main paper, driving scores of experts and IL agents at each DAGGER iterations are in Fig. 2.8 (NoCrash-busy) and Fig. 2.9 (LeaderBoard-busy).

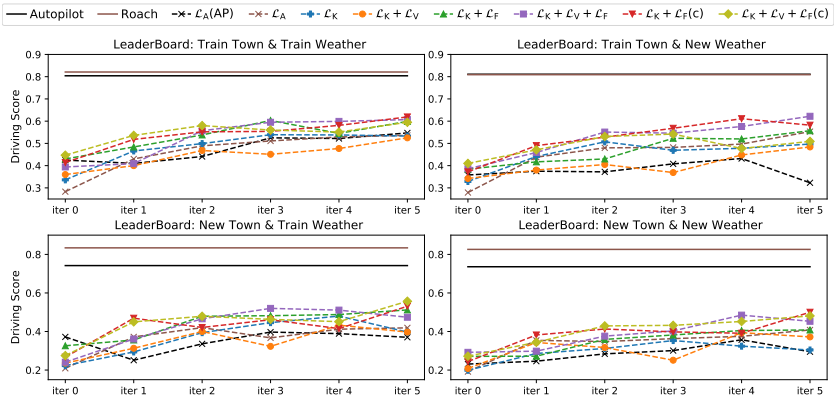


(a) Driving Score

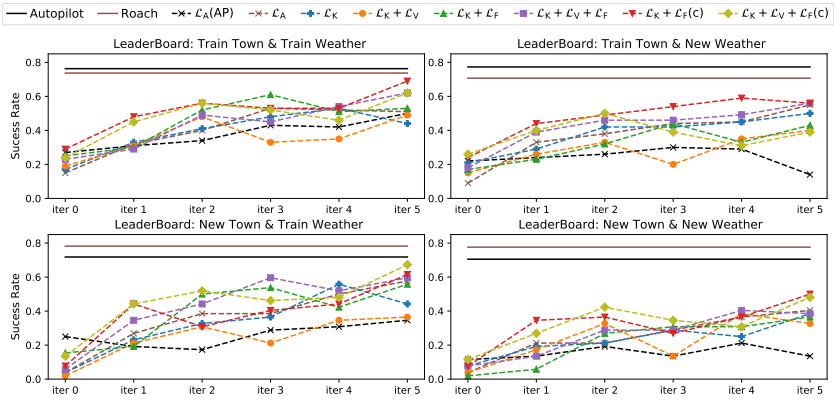


(b) Success Rate

Figure 2.8: Driving performance of experts and IL agents on the NoCrash-busy benchmark. All IL agents (dashed lines) are supervised by Roach except for $\mathcal{L}_A(\text{AP})$, which is supervised by the CARLA Autopilot. For IL agents at the 5th iteration and all experts, results are reported as the mean over 3 evaluation seeds. Others agents are evaluated only once.



(a) Driving Score



(b) Success Rate

Figure 2.9: Driving performance of experts and IL agents on the offline LeaderBoard-busy benchmark. All IL agents (dashed lines) are supervised by Roach except for $\mathcal{L}_A$(AP), which is supervised by the CARLA Autopilot. For all experts, results are reported as the mean over 3 evaluation seeds. Results of IL agents are evaluated only once.

3

A Multiplicative Value Function for Safe and Efficient Reinforcement Learning

An emerging field of sequential decision problems is safe Reinforcement Learning (RL), where the objective is to maximize the reward while obeying safety constraints. Being able to handle constraints is essential for deploying RL agents in real-world environments, where constraint violations can harm the agent and the environment. To this end, we propose a safe model-free RL algorithm with a novel multiplicative value function consisting of a safety critic and a reward critic. The safety critic predicts the probability of constraint violation and discounts the reward critic that only estimates constraint-free returns. By splitting responsibilities, we facilitate the learning task leading to increased sample efficiency. We integrate our approach into two popular RL algorithms, Proximal Policy Optimization and Soft Actor-Critic, and evaluate our method in four safety-focused environments, including classical RL benchmarks augmented with safety constraints and robot navigation tasks with images and raw Lidar scans as observations. Finally, we make the zero-shot sim-to-real transfer where a differential drive robot has to navigate through a cluttered room. Our code can be found at `https://github.com/nikeke19/Safe-Mult-RL`.

3.1 Introduction

Reinforcement Learning (RL) has made significant progress in recent years. Breakthroughs have been achieved on playing Atari [51] and board games like Go [52], as well as multi-agent RL on Starcraft [53] and Dota [54]. While some works like Miki et al. [190] deploy RL agents on real-world systems, most of the research is still conducted in simulation [35, 191]. On the one hand, the sim-to-real transfer requires high-fidelity simulators and robust models. On the other hand, there is the safety aspect. In simulation, the agent can perform any action without real consequences. However, when robots are deployed in reality, not every action is admissible. This can be due to damage to the agent, e.g., when a robot crashes into an obstacle, or damage to the environment, e.g., when the obstacle is a human. Thus, it is necessary to consider safety by design. We can formulate safety requirements using Constrained Markov Decision Process (CMDP) [192], where in

This chapter was published as a conference article: Nick Bührer, Zhejun Zhang, Alexander Liniger, Fisher Yu, Luc Van Gool, "A Multiplicative Value Function for Safe and Efficient Reinforcement Learning", International Conference on Intelligent Robots and Systems (IROS), 2023, doi: 10.1109/IROS55552.2023.10342288

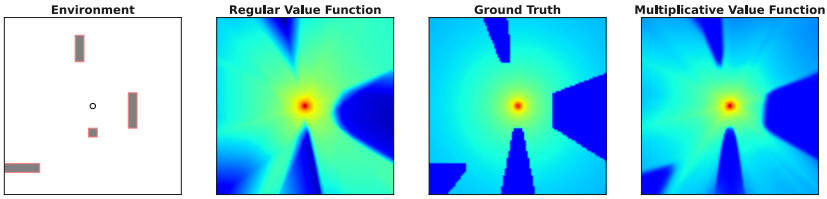


Figure 3.1: Linear Quadratic Regulator policy evaluation of a point robot navigating towards the goal in the middle while avoiding the gray obstacles. The ground truth return shows sharp discontinuities that are better replicated by the multiplicative value function. We use a deterministic policy and obtain the ground truth return by running Monte Carlo rollouts on a fine grid.

addition to maximizing the reward, the RL agent has to fulfill constraints on the expected accumulated safety cost.

Usually, the expected safety cost is estimated with a value function, called the safety critic. With a particular choice of safety cost, the cumulative cost constraint can be transformed into a chance constraint and be relaxed with a Lagrange multiplier. In contrast to previous works [68, 69, 70, 72], we additionally propose a novel multiplicative value function where the safety critic explicitly addresses constraints and discounts a reward critic that only estimates constraint-free returns. The multiplicative value function has several advantages. A standard RL algorithm can learn safe behavior by specifying a penalty for constraint-violating actions. However, the performance can be sensitive to the magnitude of the penalty [76]. In contrast, we do not need to specify the magnitude: the reward critic neglects constraint violating returns, and the safety critic learns a binary decision. Moreover, penalties are often large in magnitude to discourage standard RL agents from constraint violating actions. As a result, there can be sharp discontinuities in the value landscape as shown in Fig. 3.1, which makes it difficult for a regular neural network to learn the value function. In our approach, the reward critic does not have to learn these discontinuities. Instead, the responsibility is shifted to the safety critic that estimates the probability of constraint violation. This makes the model optimization better behaved and leads to faster convergence with increased stability.

We combine our approach with two popular algorithms: Proximal Policy Optimization (PPO) [172] and Soft Actor-Critic (SAC) [58], but in general, it can be combined with any on-policy or off-policy method that relies on a value or advantage function. To evaluate the effectiveness of our approach, we construct four safety-critical environments ranging from low to high dimensional observations based on images and Lidar scans. Finally, we deploy our algorithm on a real robot and perform map-less navigation.

Our contributions are summarized as follows:

- We introduce a novel multiplicative value function, that combines a regular value function and a safety critic in a multiplicative fashion. We integrate this multiplicative value function into PPO and SAC.
- We test our methods on a suit of safety-critical environments and show that our methods outperform competing safe RL methods.

- We conduct experiments on a real-world robot navigation task in a cluttered environment and show zero-shot sim-to-real transfer.

3.2 Related Work

An overview of the recent advances in safe RL can be found in the latest surveys [89, 90]. In this section, we will focus on the prior works most related to our approach.

Primal-dual approaches, i.e., Lagrangian-based approaches, convert the CMDP into an unconstrained problem by relaxing the safety constraint with a Lagrange multiplier. This is the most straightforward way to solve a CMDP and is also the most related to our work. A Lagrange multiplier with heuristic update rules was first proposed in [72]. More recent works established theoretical groundwork by proving convergence guarantees [68, 74] and zero duality gap [75]. Practically, the Lagrangian approach has been integrated with PPO/TRPO [193] and SAC [73, 194]. Our PPO implementation is similar to [193] except that our multiplicative value function adds a secondary mechanism that improves the learning stability. For our SAC integration, the gradient of the multiplicative value function already naturally results in something similar to a Lagrange multiplier. Commonly, the safety constraint is formulated as a constraint budget over the expected safety cost, e.g., [68, 69, 70, 193, 194, 195]. However, only considering the expected safety at each time step can cause the realized cost to exceed the constraint budget [73]. To provide better constraint satisfaction, worst-case analysis can be performed with the conditional value at risk [68, 73]. We tackle this issue by using reachability analysis and imposing zero constraint violation probability in our experiments. Another line of research improves the stability of the learning process by using derivatives and integrals of the constraint function yielding a PID approach [70, 195]. Similarly, our multiplicative value function improves stability by simplifying the learning task. Lastly, there are works that ensure state-wise safety with a learned Lagrange multiplier [194] and introduce safety transfer learning [69].

Primal approaches solve the CMDP by computing a policy gradient that satisfies the constraints. Prior works have achieved this in different ways, for example by searching the feasible policy in the trust region [76], projecting the unconstrained policy [77], projecting the optimum of the constrained non-parametric policy optimization [78], restricting the policy via log-barrier functions [79], alternating between objective improvement and constraint satisfaction [80] or deriving an equivalent unconstrained problem [196, 197]. Since the primal algorithms are harder to implement but not superior in terms of performance, they are less popular than the primal-dual algorithms.

Lyapunov approaches address the CMDP by leveraging Lyapunov functions, which are used in control theory to prove the stability of a dynamical system. In terms of model-based RL, the Lyapunov function can be used to guarantee that an agent can be brought back to a region of attraction [81]. More recently, this approach has been applied to model-free RL [82] and extended in [83] by an exploratory policy that maximizes its knowledge about safety. In [84], the policy optimization is constrained on the Lyapunov decrease condition, which is then relaxed with a Lagrange multiplier.

Intervention approaches use backup policies to ensure safe actions. Wagener et al. [85] defines an intervention rule based on the safety advantage between the action proposed by the backup policy and the RL agent. Another possibility is to construct a safe set using model-based or learning-based approaches, or a combination of both. Examples are control barrier functions [86, 87], reachability methods [88, 198] or predictive safe set algorithms [199, 200, 201].

3.3 Preliminaries

Lagrangian methods. Let us consider the optimization problem $\min_x f(x)$, st. $g(x) \leq c$, using Lagrangian primal-dual methods this problem can be cast to an unconstrained problem

$$(x^*, \lambda^*) = \min_x \max_{\lambda \geq 0} f(x) + \lambda(g(x) - c)\,,$$

where λ denotes the dual variable or Lagrange multiplier [202].

CMDP formulation. The interaction between the agent and the environment can be modeled with a Markov Decision Process (MDP). The MDP is defined by the Tuple $(\mathcal{S}, \mathcal{A}, r, P, s_0)$. Here $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space respectively, the transition probability is $P(\cdot|s,a)$ and the reward is $r(s,a)$. Finally, $s_0 \in \mathcal{S}$ is the initial state. For simplicity, we consider deterministic rewards and initial states, but our results can be easily generalized to random initial state distributions and rewards. Extending an MDP with constraints yields a CMDP which is described by the tuple $(\mathcal{S}, \mathcal{A}, r, P, s_0, r_c, c_{\max})$. Here, r_c is a safety cost and $c_{\max} \in \mathbb{R}_{\geq 0}$ is an upper bound on the expected cumulative safety cost. This yields the safety constraint $\mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma_c^t r_c(s_t) \middle| s_0\right] \leq c_{\max}$. The objective of the CMDP is to find an optimal policy π^* according to

$$\max_{\pi \in \Delta} \mathbb{E}^\pi \left[\sum_{t=0}^\infty \gamma^t r(s_t, a_t) \middle| s_0 \right], \quad \text{s.t. } \mathbb{E}^\pi \left[\sum_{t=0}^\infty \gamma_c^t r_c(s_t) \middle| s_0 \right] \leq c_{\max}\,. \tag{3.1}$$

If the feasibility set induced by the safety constraint in Eq. 3.1 is non-empty, then there exists an optimal policy π^* in the class of stationary Markovian policies Δ [192].

Reachability. To make the safety constraint in Eq. 3.1 more tangible, we specify the (un)safety as $P(\exists k : s_k \in \mathcal{C} | s_0 = s_t)$, i.e., the probability of visiting states in the constraint set $\mathcal{C} \subseteq \mathcal{S}$. We consider constraint violations as catastrophic, thus states $s \in \mathcal{C}$ are terminal states. Coming back to the CMDP, the reachability problem can be cast to a value function by setting $r_c(s_t) = \mathbb{1}_{\mathcal{C}}(s_t)$,

$$P(\exists k : s_k \in \mathcal{C} | s_0 = s_t) := \Phi^\pi(s_t) = \mathbb{E}^\pi \left[\sum_{k=0}^\infty \gamma_c^k \mathbb{1}_{\mathcal{C}}(s_k) \middle| s_0 = s_t \right], \tag{3.2}$$

where $\mathbb{1}_{\mathcal{C}}$ is the indicator function of the constraint set $\mathcal{C}$.

For the proof, we closely follow [72]. First, we note that the sum $R_c := \sum_{k=0}^\infty \gamma_c^k r_c(s_k)$ is finite and at most 1, namely when a constraint violation occurs and we reach a terminal state. Thus, when setting $\gamma_c = 1$, it holds that $R_c = 1$ if $\exists t : s_t \in \mathcal{C}$ else 0. We note

that R_c is a Bernoulli Random variable and define $P(R_c = 1) := q$. From the Bernoulli distribution we know that $\mathbb{E}[R_c] = q = \Phi^\pi(s_t)$.

Practically, a lower discount factor can increase the learning stability when used in an RL setup [72]. Furthermore, we denote $\Phi^\pi(s_t)$ as the safety critic and similar to Eq. 3.2, we define the action value safety critic as $\Psi^\pi(s_t, a_t)$.

3.4 Methods

Environment structure. We assume the following bounded reward structure of the environment

$$r(s_t, a_t) = \begin{cases} r_{\text{constraint}} & \text{if } s_t \in \mathcal{C} \\ r_{\text{constraint\_free}}(s_t, a_t) & \text{else} \end{cases}, \tag{3.3}$$

with $r_{\text{constraint}} \ll \min_{s,a} r_{\text{constraint\_free}}(s, a)$ and the constraint set $\mathcal{C}$ being a terminal state. The low reward for violating the constraint discourages standard RL agents from executing constraint violating actions. However, as shown in Fig. 3.1, the difference of magnitude between $r_{\text{constraint}}$ and $r_{\text{constraint\_free}}$ can cause discontinuities in the value landscape that are difficult to learn.

Multiplicative value function. The motivation behind our multiplicative value function is to facilitate the learning by splitting responsibilities. The safety critic $\Phi^\pi(s_t)$ explicitly handles constraints, whereas the reward critic $\bar{V}^\pi(s_t)$ only estimates constraint-free returns. As argued in Sec. 3.1, it is neither favorable to specify a magnitude for $r_{\text{constraint}}$ nor to learn $r_{\text{constraint}}$ with the reward critic $\bar{V}^\pi(s_t)$. Instead, we propose to clip the reward in Eq. 3.3, and learn the reward critic with this constrain neglecting reward,

$$\bar{r}(s_t, a_t) = \begin{cases} \min_{s,a} r_{\text{constraint\_free}}(s, a) & \text{if } s_t \in \mathcal{C} \\ r_{\text{constraint\_free}}(s_t, a_t) & \text{else} \end{cases},$$

$$\bar{V}^\pi(s_t) = \mathbb{E}^\pi \left[\sum_{k=0}^{\infty} \gamma^k \bar{r}(s_k, a_k) \middle| s_0 = s_t \right].$$

By taking the minimum in case of a constraint violation, we lightly discourage the policy from constraint violating actions. This can be especially useful when approximation errors cause the safety critic to be overly optimistic. Finally, the multiplicative value function $V^\pi_{\text{mult}}(s_t)$ is obtained by discounting the reward critic with the probability of constrained satisfaction:

$$V^\pi_{\text{mult}}(s_t) := (\bar{V}^\pi(s_t) - \bar{v}_{\min}) \cdot (1 - \Phi^\pi(s_t)) + \bar{v}_{\min},$$

where $\bar{v}_{\min} := \min_s \bar{V}^\pi(s)$ is the lower bound on the reward critic, such that $(\bar{V}^\pi(s_t) - \bar{v}_{\min}) \geq 0$. Practically, we set $\bar{v}_{\min}$ to the minimum encountered $\bar{V}^\pi$ value during training. The multiplicative combination of the two critics allows a hyperparameter-free fusion, where a constraint violating state is associated with the value of $\bar{v}_{\min}$ and

for save states, the value is $\bar{V}^\pi(s_t)$. Similarly, we define $Q^\pi_{\text{mult}}(s_t, a_t)$ with $\bar{q}_{\min}$ as the multiplicative action value function:

$$Q^\pi_{\text{mult}}(s_t, a_t) := [\bar{Q}^\pi(s_t, a_t) - \bar{q}_{\min}] \cdot (1 - \Psi^\pi(s_t, a_t)) + \bar{q}_{\min}\,. \tag{3.4}$$

Note that the offset terms $\bar{v}_{\min}$ and $\bar{q}_{\min}$ could be avoided by assuming positive rewards. Nevertheless, introducing this term allows our formulation to handle arbitrary reward functions.

Multiplicative advantage. We also want to consider advantage-based policy gradient methods. The advantage $A^\pi(s_t, a_t)$ is defined as,

$$A^\pi = Q^\pi - V^\pi = [r(s_t, a_t) + \gamma V^\pi(s_{t+1})] - V^\pi(s_t)\,. \tag{3.5}$$

From this, we derive three versions of the multiplicative advantage A^π_{mult}:

$$\begin{aligned}
&\text{V1: } [\bar{r}_t + \gamma V^\pi_{\text{mult}}(s_{t+1})] - V^\pi_{\text{mult}}(s_t) \\
&\text{V2: } Q^\pi_{\text{mult}}(s_t, a_t) - V^\pi_{\text{mult}}(s_t) \\
&\text{V3: } [\bar{Q}^\pi(s_t, a_t) - \bar{q}_{\min}]\,[1 - (r_{c,t} + \gamma_c \Phi^\pi(s_{t+1}))] + \bar{q}_{\min} - V^\pi_{\text{mult}}(s_t)
\end{aligned} \tag{3.6}$$

In V1 and V2, we consider Eq. 3.5 and replace all value functions with their multiplicative counterparts. Finally, V3 is similar to V2, but in Eq. 3.4 we use temporal difference bootstrapping for the safety critic.

Integration into SAC. We integrate the multiplicative value function Q^π_{mult} into the actor objective of SAC by replacing Q^π with Q^π_{mult},

$$\max_\theta \mathbb{E}_{a_\theta \sim \pi_\theta}[Q^{\pi_\theta}_{\text{mult}}(s, a_\theta) - \alpha \log \pi_\theta(a_\theta|s)]\,. \tag{3.7}$$

We call this version SAC Mult. For a compact notation, we drop the dependency on (s, a_θ, π) in the following. To get a better intuition about Eq. 3.7, we investigate the gradient of the SAC Mult objective

$$\nabla_\theta Q_{\text{mult}} = (1 - \Psi) \cdot \nabla_\theta \bar{Q} - (\bar{Q} - \bar{q}_{\min}) \cdot \nabla_\theta \Psi,$$

which has two terms. The first term is the gradient of the $\bar{Q}$-function discounted by the probability of constraint satisfaction. The second term is the gradient of the safety critic $\nabla_\theta \Psi$ discounted by $(\bar{Q} - \bar{q}_{\min})$, which can be understood as q-weighted multiplier. The disadvantage of this formulation is that in states where $\bar{Q}$ is high, the q-weighted multiplier becomes large and the gradient of the safety critic dominates the overall gradient. This can yield overly conservative behaviors. To mitigate this issue, we additionally propose two heuristics, SAC Mult Clipped

$$\nabla_\theta Q_{\text{mult}} \approx (1 - \Psi) \cdot \nabla_\theta \bar{Q} - \min(\bar{Q} - \bar{q}_{\min}, \lambda_{\max}) \cdot \nabla_\theta \Psi$$

and SAC Mult Lagrange

$$\nabla_\theta Q_{\text{mult}} \approx (1 - \Psi) \cdot \nabla_\theta \bar{Q} - \lambda \cdot \nabla_\theta \Psi\,.$$

In Mult Clipped, we limit the magnitude of the multiplier with $\lambda_{\max}$ which is a hyperparameter. In Mult Lagrange, we replace the q-weighted multiplier with a Lagrange multiplier that is optimized using primal-dual optimization. Under mild assumptions, this is

guaranteed to converge to a local optimum of the CMDP [68]. The Mult Lagrange objective is $\max_\theta \min_{\lambda \geq 0} \mathbb{E}_{a_\theta \sim \pi_\theta}[(1 - \Psi(s, a_\theta)).\text{detach}() \cdot \bar{Q}(s, a_\theta) - \alpha \log \pi_\theta(a_\theta|s) - \lambda \cdot (\Psi(s, a_\theta) - c_{\max})]$, where .detach() denotes the operation of detaching the variable from the computational graph. For the exact implementation, we refer to our code.

Integration into PPO. Given the three versions of the multiplicative advantage A^π_{mult} in Eq. 3.6, we can integrate them into the actor objective of PPO and extend it with a Lagrange multiplier

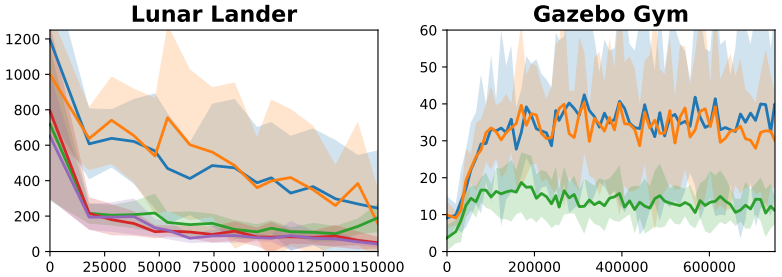
$$\max_\theta \min_{\lambda \geq 0} \mathbb{E}_{s,a \sim \pi_{\theta_k}}[\min\left\{\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}_{\text{mult}},\ g(\epsilon, A^{\pi_{\theta_k}}_{\text{mult}})\right\} - \lambda \cdot \mathbb{E}_{a_\theta \sim \pi_\theta}[\Psi^\pi(s, a_\theta) - c_{\max}]],$$

with $g(\epsilon, A) = (1 + \epsilon)A \cdot \mathbf{1}_{A \geq 0} + (1 - \epsilon)A \cdot \mathbf{1}_{A < 0}$. For the optimization of the Lagrange multiplier, we again proceed as in [68] to guarantee convergence to a locally optimal policy.

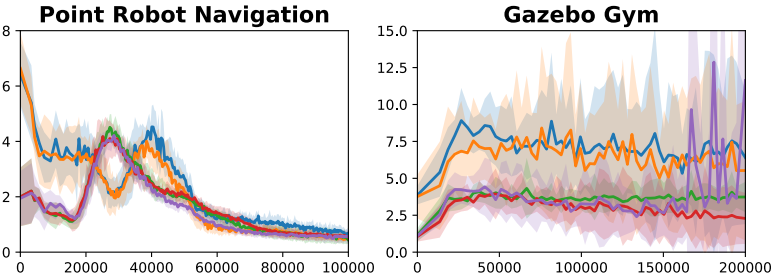
3.5 Experimental Results

We evaluate our methods in four environments: Lunar Lander Safe, Car Racing Safe, Point Robot Navigation, and Gazebo Gym. The first two are derived from OpenAI's gym [203] and extended with safety constraints. For Lunar Lander Safe, we impose the constraint that the agent can only land within the landing zone. In Car Racing Safe, we test how our algorithm deals with both hard and soft constraints. We set the hard constraint that the agent is not allowed to leave the track. For the soft constraint, we encourage the agent to drive below 50 km/h. If the soft constraint is violated, the agent gets a small negative reward, but the episode still continues. The agent observes the environment via an agent centered bird's-eye-view image and a vector containing information about the steering angle, yawing rate, and velocity. The last two environments focus on robotic navigation of ground robots. In Point Robot Navigation, the agent is a point robot that has to navigate towards the goal by choosing the 2D velocity while avoiding obstacles. At each iteration, a new set of random obstacles is spawned and the agent starts at a random position. The agent perceives its environment via a local occupancy grid and the vector from the current position to the goal. The Gazebo Gym is similar but the agent is a Jackal differential drive robot modeled in Gazebo [204] with speed and yaw rate as input and the occupancy grid is replaced by a 1D-Lidar scan. Again, the task is to navigate to the goal that lies in the middle of a cluttered room.

For the tuning, we use a sequential grid search on the learning rate, entropy coefficient, the number of optimization steps and experiment with the Beta distribution for the policy. For FOCOPS [78], we additionally tune lambda, the batch size and the KL divergence target. Furthermore, we tune the KL target and the clip range for PPO and the training frequency for SAC. After the baseline tuning, we keep the same hyperparameters for our multiplicative versions and additionally tune the safety discount factor y_c and the initial value of the Lagrange multiplier λ_{init}.



(a) Value losses for PPO. Blue: Base, Orange: Lagrange, Green: Mult V1, Violet: Mult V2, Red: Mult V3.



(b) Value losses for SAC. Blue: Base, Orange: Lagrange, Green: Mult, Red: Mult Clipped, Violet: Mult Lagrange.

Figure 3.2: Qualitative Results

3.5.1 Results and Comparisons

With our experiments, we want to answer two questions: Firstly, can the integration of the multiplicative value function facilitate the learning, leading to faster convergence and improved stability? For this, we integrate our approach into the SAC and PPO and compare against its Lagrangian counterpart. Secondly, we ask, can the integration of the multiplicative value function into Lagrangian approaches yield comparable performance to recent approaches, e.g., FOCOPS? All the results are shown in Table 3.1, where we evaluate each model at an intermediate checkpoint and at the end of the training. Each evaluation is over 10 seeds with 100 episodes.

Increased sample efficiency. One of the main motivations for the multiplicative value function is to simplify the learning task. This is supported by Fig. 3.2, where we observe a lower mean value loss and reduced variance across environments for both SAC and PPO. Having a simpler learning task allows our multiplicative versions to achieve significantly fewer constraint violations and higher rewards at the first evaluation checkpoint, indicating greater sampling efficiency. At the final evaluation checkpoint, our method achieves similar constraint satisfaction as the Lagrangian baseline. This is expected since both are Lagrangian methods and with enough training samples and model capacity, the regular value function can properly learn the value landscape.

Constraint satisfaction. In simpler environments, like Lunar Lander and Point Robot Navigation, our approach nearly achieves the target of zero constraint violations with PPO and SAC. In Car Racing Safe, the Lagrangian baseline and PPO V1 achieve 91% constraint satisfaction. The imperfect performance could be caused by the challenging environment setup where minor driving errors can result in a crash. Due to the long run-time, we stopped the Gazebo Gym experiments in Table 3.1 before convergence. In fact, SAC Mult Lagrange trained for 2 days as done for the sim-to-real transfer in Sec. 3.5.2 achieves a constraint satisfaction of 100%.

PPO vs. SAC. Overall, we achieve better constraint satisfaction with SAC agents in the navigation tasks Point Robot Navigation and Gazebo Gym. The only caveat is that we were not able to successfully train any SAC (nor FOCOPS) algorithm on Car Racing Safe because the agents never “make" the first corner. Overall for PPO, we observe an increased variance in the training reward if the maximum allowed KL divergence is not explicitly tuned. The tuning is necessary because the multiplicative value function together with the Lagrange multiplier yields more aggressive policy updates which can cause instabilities.

Soft constraint satisfaction. In Car Racing Safe, we additionally imposed the soft constraint to keep the velocity below 50 km/h. Even though PPO V1 and V2 achieve similar constraint satisfaction to the Lagrangian baseline, the reward is higher. This is because V1 and V2 violate the soft constraint with 20% and 12% respectively, whereas the Lagrangian Baseline violates the constraint in 32% of the steps. We credit the multiplicative value function for the improved soft-constraint satisfaction. By facilitating the learning, the reward critic has more capacity to learn the fine details in the reward structure, like the soft constraint on the velocity.

Increased stability. We observe better training stability across seeds with the multiplicative value function. This is most prominent in Lunar Lander Safe, where we observe a large variance in the Lagrangian baseline rewards. This is because the Lagrangian agent only lands in 80% of the seeds reliably, both for SAC and PPO. In contrast, SAC Mult Lagrange, Clipped and PPO Mult V2, V3 agents manage to land in all seeds, PPO Mult V1 in 90% of the seeds. The poor performance of SAC Mult is not due to missing stability, in fact, SAC Mult performs badly across seeds. The issue is caused by the potentially large q-weighted multiplier, which makes the policy updates overly conservative leading to high timeout rates without ever landing.

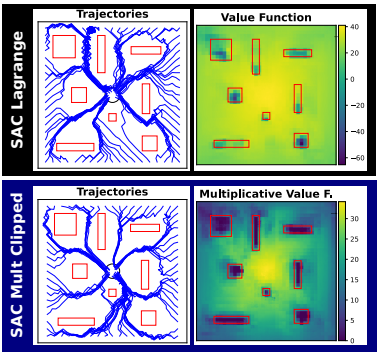
	Reward ↑	% Constraint violations ↓	Reward ↑	% Constraint violations ↓
	Lunar Lander Safe			
SAC	50k		150k	
SAC base	90 ± 108	10 ± 16	181 ± 117	3 ± 6
Lagrange	111 ± 105	17 ± 13	184 ± 128	2 ± 3
Mult	-35 ± 27	$\mathbf{3 \pm 5}$	-34 ± 22	3 ± 4
Mult Clipped	$\mathbf{134 \pm 94}$	14 ± 13	243 ± 49	8 ± 15
Mult Lagrange	125 ± 59	29 ± 15	$\mathbf{251 \pm 20}$	$\mathbf{2 \pm 2}$
PPO	50k		150k	
PPO base	-126 ± 158	77 ± 29	225 ± 100	10 ± 30
Lagrange	-24 ± 146	54 ± 39	204 ± 116	12 ± 24
V1	101 ± 84	41 ± 19	205 ± 78	7 ± 16
V2	89 ± 122	44 ± 34	251 ± 28	5 ± 9
V3	$\mathbf{144 \pm 4}$	$\mathbf{26 \pm 22}$	$\mathbf{264 \pm 5}$	$\mathbf{1 \pm 2}$
FOCOPS	-129 ± 21	64 ± 24	117 ± 80	30 ± 19
	Point Robot Navigation			
SAC	50k		100k	
SAC base	22 ± 19	1 ± 1	$\mathbf{38 \pm 1}$	1 ± 1
Lagrange	29 ± 8	$\mathbf{0 \pm 0}$	$\mathbf{38 \pm 1}$	$\mathbf{0 \pm 0}$
Mult	34 ± 3	$\mathbf{0 \pm 0}$	$\mathbf{38 \pm 1}$	$\mathbf{0 \pm 0}$
Mult Clipped	33 ± 4	$\mathbf{0 \pm 0}$	$\mathbf{38 \pm 1}$	$\mathbf{0 \pm 0}$
Mult Lagrange	$\mathbf{37 \pm 2}$	$\mathbf{0 \pm 0}$	37 ± 2	$\mathbf{0 \pm 0}$
PPO	250k		500k	
PPO base	15 ± 15	3 ± 3	31 ± 3	3 ± 2
Lagrange	15 ± 16	3 ± 3	24 ± 5	3 ± 2
V1	25 ± 5	3 ± 3	30 ± 4	2 ± 1
V2	11 ± 21	3 ± 3	17 ± 15	3 ± 1
V3	$\mathbf{27 \pm 5}$	5 ± 2	29 ± 3	3 ± 3
FOCOPS	17 ± 11	$\mathbf{0 \pm 0}$	$\mathbf{33 \pm 2}$	$\mathbf{0 \pm 0}$
	Gazebo Gym			

SAC	100k		200k	
SAC base	34 ± 6	7 ± 9	35 ± 5	6 ± 8
Lagrange	30 ± 7	11 ± 12	35 ± 6	5 ± 10
Mult	34 ± 11	5 ± 12	35 ± 6	$\mathbf{3 \pm 7}$
Mult Clipped	$\mathbf{36 \pm 5}$	$\mathbf{3 \pm 8}$	$\mathbf{36 \pm 5}$	3 ± 8
Mult Lagrange	36 ± 6	4 ± 9	$\mathbf{36 \pm 5}$	3 ± 8
PPO	**250k**		**750k**	
PPO base	27 ± 6	18 ± 11	$\mathbf{31 \pm 5}$	12 ± 8
Lagrange	26 ± 5	19 ± 9	31 ± 6	12 ± 9
V1	$\mathbf{30 \pm 6}$	$\mathbf{14 \pm 9}$	$\mathbf{31 \pm 5}$	$\mathbf{11 \pm 8}$
FOCOPS	13 ± 12	19 ± 8	29 ± 5	15 ± 9
	Car Racing Safe			
PPO	**500k**		**1000k**	
PPO base	43 ± 23	28 ± 25	89 ± 25	9 ± 15
Lagrange	43 ± 23	28 ± 25	89 ± 25	9 ± 15
V1	$\mathbf{74 \pm 19}$	$\mathbf{17 \pm 14}$	98 ± 13	$\mathbf{9 \pm 7}$
V2	65 ± 24	26 ± 15	$\mathbf{100 \pm 9}$	10 ± 6
V3	73 ± 21	25 ± 15	78 ± 17	22 ± 16
FOCOPS	-2 ± 2	93 ± 2	-2 ± 2	93 ± 2

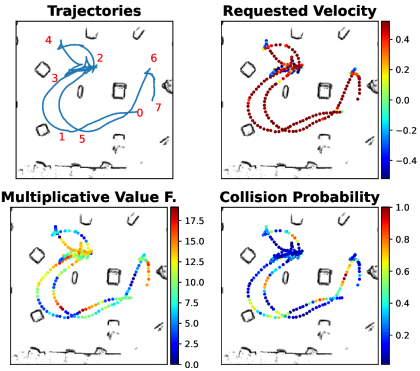
Table 3.1: SAC and PPO evaluation results. Overall, PPO Mult V1 delivers most consistently good performance across environments. This might be because V1 is based on the Generalized Advantage Estimation [184] which has shown to work particularly well for standard PPO. Among the SAC derivatives, Mult Clipped and Mult Lagrange perform the most consistent. For SAC Mult, the q-weighted safety multiplier yields overly conservative behavior in Lunar Lander, where the agent rarely lands, but instead times out. Consequently the reward is low.

Qualitative results. In Fig. 3.3a, we show the qualitative results for Point Robot Navigation. Of most interest is the multiplicative value function, which can better represent the obstacles highlighted by red boxes. Furthermore, the trajectories of SAC Mult Clipped seem more directed towards the goal compared to the Lagrangian baseline. In Lunar Lander Safe, we observe the Mult agents land faster than the Lagrangian agents by having greater downward speeds high above the landing pad, while at lower altitudes, landing as cautiously as the Lagrangians.

Theoretical guarantees vs. heuristics. Based on [68], we have theoretical safety guarantees for all PPO Multiplicative versions as well as for SAC Mult Lagrange. The SAC Mult and Clipped inhibit a q-weighted multiplier from the gradient of the multiplicative



(a) Evaluation of SAC Mult Lagrange and the Lagrangian baseline after 100k steps on Point Robot Navigation. The multiplicative value function better represents obstacles.



(b) Trajectories of real-world experiment with a differential drive robot and SAC Mult Lagrange. Goal regions are marked with numbers 1-7. Starting point is 0. Here, success rate is 100%.

Figure 3.3: Qualitative results on robot navigation environments.

value function. However, this multiplier is not a Lagrangian multiplier, thus has no theoretical guarantees. Practically, we observe that SAC Mult and Mult Clipped have similar constraint satisfaction as SAC Mult Lagrange.

Comparison to FOCOPS. For Lunar Lander, the FOCOPS evaluation result at 150k steps is significantly worse than for any PPO Mult algorithm. We suspect this is caused by poor sample efficiency. Therefore, we train FOCOPS up to 300k steps where it obtains a reward of 215 ± 65 and a constraint violation rate of $13 \pm 18\%$. This is still worse than any Mult algorithm at 150k steps and is due to two seeds showing high constraint violation rates of 53% and 38%. In Point Robot Navigation, our algorithms converge faster but finally, FOCOPS outperforms all PPO Mult agents and is only beaten by SAC. In Gazebo Gym, FOCOPS performs worse than PPO in both evaluations, however, longer training could yield more comparable performance. Finally, in Car Racing Safe the FOCOPS agent never gets passed the first corner similar to SAC.

3.5.2 Real-World Experiments

Based on the good performance of SAC Mult Lagrange on Gazebo Gym, the question arises of how the learned policy performs on a real Jackal robot? Can we tackle navigation, one of the fundamental robotic problems, in a safe way while only given sparse Lidar observations and a direction to the goal? To this end, we trained the policy for 4M steps in simulation with action noise, a smaller goal region of 0.3m and noisy Lidar observations. Furthermore, we included a cross-attention encoder for both the policy and value nets as depicted in Fig 3.4. This attention mechanism allows the networks to focus on the latent representation of certain Lidar rays, for example the rays that are pointing forward. Those

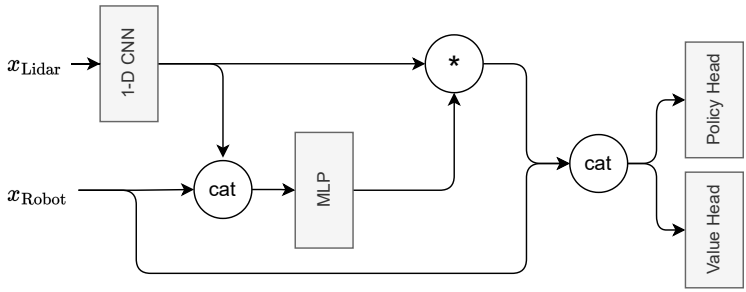


Figure 3.4: Attention Encoder used for the real world experiments. Here x_{Robot} is the vector from current position to the goal and x_{Lidar} is the measurement from the 1D-Lidar.

changes helped the policy to achieve a 100% success rate with 0% constraint violations in the simulation.

One of the main differences between simulation and reality is that the ground friction in the real world is larger and the velocity controller behaves differently, i.e., given the same velocity command, the robot moves faster in simulation. In reality, if the robot is stationary and the velocity command is chosen too small, the robot can remain stationary due to the increased friction and different behavior of the velocity controller. Another real-world difficulty is the delay of 0.3s from the Lidar rays being recorded, send to the off-board computer, and the policy commands being sent back and executed on the robot. However, the fact that the real robot moves slower mitigates the time delay to an extent.

In the lab, we constructed a cluttered obstacle course and directly deployed the policy trained in simulation on the robot. We defined seven goal regions and let the robot pass them four times. One of the four runs is depicted in Fig. 3.3b. The robot drives with a direct trajectory from start 0 to goals 1 and 2, where it needs to reverse its direction by 180 degrees to approach goal 3. Interestingly, the policy did not learn to turn on the spot a differential drive robot would allow, but instead, the trajectories from goal 2-4 resemble more a kinematic bicycle model. This unfortunately caused the robot to get stuck once between goals 2-3, however, without violating a safety constraint. Overall, we report a success rate of 96% with 100% constraint satisfaction meaning that no box was touched.

Encouraged by the demonstrated safety of our algorithm, we wanted to see if our agent can generalize from static box obstacles as encountered in the simulation to moving obstacles like humans in reality. For this, we arranged the goals in a circle and sequentially let the robot pass them. In the first experiment, we suddenly put a box in front of the agent as shown in Fig. 3.5 (a). The robot reacted in a safe manner and came to an immediate stop. After a few seconds, we removed the box and the robot continued its original trajectory. In the second experiment shown in Fig. 3.5 (b) we wanted to go a step further and see how the robot deals with obstacle shapes it has never seen before, i.e., human legs. Additionally, we wanted to know if the robot could naturally interact with a human walking next to it. For this, we started behind the robot, and then walked next to it at a certain safety distance, see Fig. 3.5 (b1). This did not visibly influence the robot's trajectory. When we overtook the robot, we positioned ourselves close to a box

Figure 3.5: (a): We dynamically put an obstacle in the trajectory of the robot which causes it to stop. The robot then waits until the obstacle is removed and continues its trajectory. (b1, b2): Dynamic interactions with the robot by first walking next to it (b1), then overtaking it and standing to the side of the box (b2). While the person walks next to it the robot continues its trajectory. When the person overtook the robot, it swings to the right to avoid the collision. (c): The person walks towards the robot. The avoid the collision, the robot drives backwards. The complete video is available at `https://youtu.be/gAcETwOWTM4`.

such that we were in the trajectory of the robot, see Fig. 3.5 (b2). Because of that, the robot corrected its trajectory and steered away from us. The final interaction is shown in Fig. 3.5 (c). Here we wanted to investigate what happens if we actively provoke a collision by moving towards the robot. In that case, the robot started to move backward to keep a certain safety distance from us. The only drawback is that when relentlessly chasing the robot one can cause a rear collision with another obstacle. An explanation for this is that the robot has a Lidar blind spot in the back due to the mounted robot arm. All the interactions can be found in the supplementary video.

3.6 Conclusions and Limitations

In this work, we introduced a safety critic to yield a multiplicative value function. We started with the CMDP formulation, derived the safety critic from reachability analysis and integrated our approach into the SAC and PPO framework. We proposed several versions of SAC and PPO using our multiplicative value function and showed increased sample efficiency and stability compared to the Lagrangian and FOCOPS baselines. Furthermore, the multiplicative value function can help to learn the fine details in the reward structure, like soft constraints. To show the real-world potential of our method, we took a SAC Mult Lagrange agent trained in simulation and successfully deployed the policy on a real robot in a zero-shot sim-to-real fashion. The robot showed safe behavior and was able to generalize to dynamic obstacles of novel shape. In future work, we would like to investigate further theoretical justification for our multiplicative value function.

Limitations. One of the main limitations is that our Lagrangian approach encourages safety during training but cannot guarantee it. This issue can be mitigated by adding an intervention mechanism, which could however cause problems when learning the safety critic as it requires reaching the constraint set. Future work will investigate the feasibility of using the intervention as a terminal state and more theoretical analysis of the multiplicative value function. Moreover, our method adds the initial value of the Lagrange multiplier and the safety discount factor γ_c as hyperparameters to which the algorithm can be sensitive in some environments.

Appendices

3.A Supplementary Video

The supplementary video for the paper is found at `https://youtu.be/gAcETwOWTM4`. This video demonstrates the qualitative outcomes of the interaction experiment with the Jackal differential drive robot. In the first part of the video, we dynamically block the path of the agent with a box. Encouraged by the demonstrated safety, we try interactions with a human in the second part of the video. This is challenging in two ways: First, the agent only perceives the legs of the human, which are significantly thinner than the obstacles encountered in training. Second, the agent only encountered static objects in training while the human is a dynamic obstacle. Still, the agent shows safe behavior.

3.B Hyperparameter Tuning

For the tuning, we start with the default parameters of [185] or the suggested parameters of [205] for Car Racing and Lunar Lander. We then tune the PPO and SAC baseline algorithms by varying the parameters of the initial learning rate and schedule, entropy coefficient and schedule, state-dependent exploration [206] vs. standard action sampling, and Gaussian vs. Beta distribution for the actor policy. For PPO, we additionally tune the clip range, the KL divergence target, the initial variance parameter of the Gaussian policy, and the number of epochs of optimization. As for SAC, we vary the training frequency, the number of gradient steps, how many samples to collect before starting the training, and the buffer size. Once satisfied with the baseline performance, we keep the same hyperparameters for our multiplicative versions and additionally tune the safety discount factor y_c and the initial value of the Lagrange multiplier λ_{init}. To best compare the effect of the multiplicative value function, we use the same initial Lagrange multiplier λ_{init} for our multiplicative versions and the Lagrange baseline.

3.C Complete Algorithms for SAC and PPO Multiplicative

Algorithm 1 is the complete version of SAC Multiplicative, whereas Algorithm 2 is the complete version of the PPO Multiplicative. The differences between our proposed methods and the standard algorithms are highlighted in blue.

Algorithm 1 SAC Multiplicative

Init: policy parameters θ, $\bar{Q}$-function parameters ξ_1, ξ_2, Safety critic parameters ψ_1, ψ_2, empty replay buffer $\mathcal{D}$.
Set target parameters equal to main parameters $\xi_{\text{targ},i} \leftarrow \xi_i$, $\psi_{\text{targ},i} \leftarrow \psi_i$, for $i \in \{1,2\}$
repeat
  Observe state s and sample action $a \sim \pi_\theta(\cdot|s)$.
  Observe next state s' and done signal d.
  Observe clipped reward $\bar{r}$ and constraint cost r_c.
  Store $(s, a, \bar{r}, r_c, s', d)$ in replay buffer $\mathcal{D}$.
  Reset environment states **if** s' is terminal.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions from $\mathcal{D}$, $B = \{(s, a, \bar{r}, r_c, s', d)\}$.
      Compute targets $y(\bar{r}, s', d)$ for $\bar{Q}$-functions:

$$\bar{r} + \gamma(1-d)(\min_i \bar{Q}_{\xi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')), \text{ where } \tilde{a}' \sim \pi_\theta(\cdot|s')$$

      Compute targets $y_c(r_c, s', d)$ for safety critic:

$$r_c + \gamma_c(1-d)(\max_i \Psi_{\psi_{\text{targ},i}}(s', \tilde{a}')), \text{ where } \tilde{a}' \sim \pi_\theta(\cdot|s')$$

      Update $\bar{Q}$-functions for $i \in \{1,2\}$:

$$\nabla_{\xi_i} \frac{1}{|B|} \sum_{b \in B} \left(\bar{Q}_{\xi_i}(s,a) - y(\bar{r}, s', d)\right)^2, \text{ where } b = (s, a, \bar{r}, r_c, s', d)$$

      Update safety critic for $i \in \{1,2\}$, BCE denotes the binary cross-entropy:

$$\nabla_{\psi_i} \frac{1}{|B|} \sum_{b \in B} \text{BCE}\left(\Psi_{\psi_i}(s,a), y_c(r_c, s', d)\right),$$

      Update policy by one step of gradient ascent:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\text{mult},\xi,\psi}(s, \tilde{a}_\theta) - \alpha \log \pi_\theta(\tilde{a}_\theta|s),$$

      where $\tilde{a}_\theta(s)$ is sampled from $\pi_\theta(\cdot|s)$ via reparametrization trick.
      Update target networks:

$$\xi_{\text{targ},i} \leftarrow \rho \cdot \xi_{\text{targ},i} + (1-\rho)\xi_i\,, \;\; \psi_{\text{targ},i} \leftarrow \rho_c \cdot \psi_{\text{targ},i} + (1-\rho_c)\psi_i\,, i \in \{1,2\}$$

    end for
  end if
until convergence

Algorithm 2 PPO Multiplicative

init: policy parameters θ_0, value function parameters ξ_0, safety critic parameters $\psi_{1,0}$, $\psi_{2,0}$, maximum unsafety probability target $c_{\max}$, and Lagrange multiplier λ.

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

Compute clipped rewards-to-go $\hat{R}_t$ and constraint cost-to-go $\hat{C}_t$.

Compute advantage estimates, $\hat{A}_{\text{mult}}$ with current value function $\bar{V}_{\xi_k}$ and safety critic Ψ_{ψ_k}.

Approximate $\hat{\Phi}^{\pi_\theta}(s) := \mathbb{E}_{a_\theta \sim \pi_\theta}[\Psi(s, a_\theta) - c_{\max}]$ the expectation in the PPO Mult objective by sampling from the policy

$$\hat{\Phi}^{\pi_\theta}(s) \approx \frac{1}{N} \sum_{i=0}^{N} \max_{j=1,2} \Psi_{\psi_j}(s, a_{\theta_i}) - c_{\max},$$

where $a_{\theta_i} \sim \pi_\theta$.

Update the policy θ by maximizing the PPO-Clip Mult objective:

$$\frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \Big(\frac{\pi_\theta(a_t|x_t)}{\pi_{\theta_k}(a_t|x_t)} A_{\text{mult}}^{\pi_{\theta_k}}(x, a), \\ g(\epsilon, A_{\text{mult}}^{\pi_{\theta_k}}(x, a)) \Big) - \lambda \hat{\Phi}^{\pi_\theta}(s_t),$$

typically via stochastic gradient ascent with Adam.

Update the Lagrange Multiplier by

$$\lambda \leftarrow \max \left(0, \lambda + \alpha \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \hat{P}_c^{\pi_\theta}(x_t) \right),$$

where α is a learning rate.

Fit value function using mean-squared error:

$$\xi_{k+1} = \arg\min_{\xi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(\bar{V}_\xi(s_t) - \hat{R}_t \right)^2$$

Update safety critic parameter $\psi_{i,k+1}$ by minimizing the binary cross entropy:

$$\frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \text{BCE}\left(\Psi_{\psi_i}(x, a), \hat{C}_t \right),$$

where $i \in \{1, 2\}$.

end for

3.D Detailed Experimental Description

To recapitulate, we assume the following reward structure of the environment

$$r(s_t, a_t) = \begin{cases} r_{\text{constraint}} & \text{if } s_t \in \mathcal{C} \\ r_{\text{constraint\_free}}(s_t, a_t) & \text{else} \end{cases}$$

Here, we associate $r_{\text{constraint}}$ with hard constraints which cause the episode to end in case of constraint violations. Additionally, there can be soft constraints encoded in $r_{\text{constraint\_free}}$. Violating a soft constraint causes a negative reward but the episode continues.

Lunar Lander Safe is a continuous control task where the agent has to land a rocket on the moon's surface [203]. Once the rocket lands, the episode ends. The agent receives a reward for minimizing the distance to the landing pad and it can land anywhere as long as its down velocity is slow enough when touching the ground. The observation is

$$\vec{x}_{\text{observation}} = [\vec{d}, \vec{v}, \phi, \dot{\phi}, \mathbb{1}_{\text{contact\_left}}, \mathbb{1}_{\text{contact\_right}}],$$

where $\vec{d}$ denotes the vector from current position to landing pad, $\vec{v}$ is the linear velocity of the agent, ϕ the roll angle and $\mathbb{1}_{\text{contact}}$ denotes if the corresponding left or right leg has ground contact. We want to make the environment more safety-critical and go by the concept "the floor is lava". This means, we keep the original constraint on the maximum allowed landing velocity but expand the constraint set $\mathcal{C}$ such that landing outside the landing pad is not allowed anymore. The rewards are

$$\begin{aligned} r_{\text{constraint\_free}} = &- 100 \cdot \left(||\vec{d}_t||_2 - ||\vec{d}_{t-1}||_2 \right) \\ &- 100 \cdot \left(||\vec{v}_t||_2 - ||\vec{v}_{t-1}||_2 \right) \\ &- 100 \cdot (\phi_t - \phi_{t-1}) \\ &+ 10 \cdot \left(\mathbb{1}_{\text{contact\_left,t}} - \mathbb{1}_{\text{contact\_left,t-1}} \right) \\ &+ 10 \cdot \left(\mathbb{1}_{\text{contact\_right,t}} - \mathbb{1}_{\text{contact\_right,t-1}} \right) \\ &- \text{fuel\_spend} \\ &+ 100 \cdot \mathbb{1}_{\text{contact\_right,t}} \mathbb{1}_{\text{contact\_left,t}} \mathbb{1}_{||\vec{v}_t||_2 < 0.01}, \\ r_{\text{constraint}} = &- 100, \end{aligned}$$

where the last line in $r_{\text{constraint\_free}}$ denotes the state of a successful landing. Furthermore, we introduce a timeout if the agent cannot land within 1000 steps. Since no measure of time is present in the observation, the agent does not know about the timeout. The action space is two dimensional, representing the impulse the agent can apply to the left/right and up/down using "rocket engines".

Car Racing Safe environment is based on CarRacing from OpenAI [203] where the agent is rewarded for driving around a race track. Each iteration, a new random track is spawned. The episode terminates if the agent has visited all track tiles or leaves the playground, which extends far beyond the track. Again, we want to make this environment more safety critic. For this, we tighten the hard constraint such that the

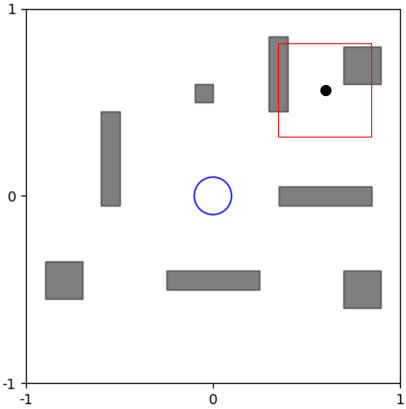


Figure 3.6: Point Robot Navigation. The obstacles are shown in gray, the goal region in blue, the agent is a black dot, and the field of view of the local occupancy grid is marked by the red box. The environment has the boundaries $[-1, 1]$ m.

agent enters the constraint set $\mathcal{C}$ if it leaves the racetrack. Also, the agent has to drive faster than 0.1 km/h after an initial time period. Additionally, we impose a soft constraint that encourages the agent to drive below 50 km/h, which is encoded in the reward

$$r_{\text{constraint\_free}} = \frac{0.3}{n_{tiles}} \cdot \mathbb{1}_{\text{new\_track}} + \begin{cases} 0.005v & \text{if } v < 50 \\ -0.01v & \text{if } v \geq 50 \end{cases}, \quad r_{\text{constraint}} = -10\,,$$

where v denotes the longitudinal velocity. The observation space of the agent consists of an agent-centered bird's-eye-view image, longitudinal velocity v, yawing rate ψ and steering angle of the wheels δ, $\vec{x}_{\text{observation}} = [\text{img}, v, \dot{\psi}, \delta]$. The action space is continuous and two-dimensional. The actions are between accelerating/braking and steering to the left/right. We use the CNN structure of [51] to process the birds-eye-view image.

3.E Point Robot Navigation

In the Point Robot Navigation environment, the agent has to navigate to a goal region of 0.1 m while avoiding obstacles and staying inside the environment boundaries. At each iteration, both the starting position of the robot and the set of obstacles are random. An example of the environment can be taken from Fig. 3.6. As the observation, the agent receives the vector $\vec{d}$ from the current position to the goal and an agent-centered occupancy grid,

$$\vec{x}_{\text{observation}} = [\vec{d}, \text{oc\_grid}].$$

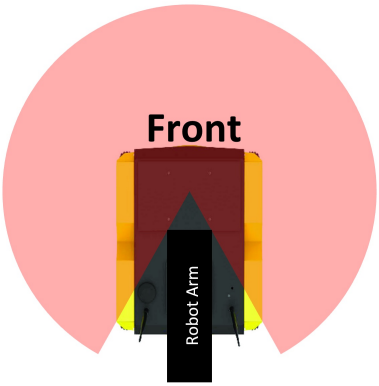


Figure 3.7: Top-down view of the Jackal Robot. The laser scan is occluded due to the robot arm yielding a field of view of 300° as depicted with the red circle.

The action space is two dimensional, representing the percentage of a step size the agent can travel in x and y-direction. The reward is

$$r_{\text{constraint\_free}} = \begin{cases} 40 & \text{if } ||\vec{d}||_2 < 0.1 \\ -0.1 & \text{else} \end{cases}, \quad r_{\text{constraint}} = -20\,. \tag{3.8}$$

This corresponds to a sparse reward setting. The step penalty of -0.1 encourages the agent to reach the goal in a low number of steps. Furthermore, we choose the reward for reaching the goal higher than the penalty for constraint violation. With a lower goal reward, we have experienced baseline agents that try to crash immediately into obstacles to avoid the accumulation of step penalties. To process the occupancy grid, we use a small CNN encoder. When initially training SAC and PPO baselines, we experienced instability. This was related to the model not understanding the occupancy grid. To mitigate the issue, we added a decoder module to the CNN and posed an auxiliary loss on the reconstruction error. This improved the performance and stability of PPO. For SAC, we had to additionally use separate CNN encoders for actor, reward and safety critic.

Gazebo Gym is similar to the Point Robot Navigation meaning that it shares the same task and constraint set $\mathcal{C}$ but resembles a realistic environment. The agent is a Jackal differential drive robot. Furthermore, the environment size is increased from 1x1 to 8x8 m. Similarly, the goal region is enlarged to 1 m in the easy and to 0.3 m in the hard setting of Gazebo Gym. The observation of the agent is given as

$$\vec{x}_{observation} = [\vec{d}, \sin(\psi), \cos(\psi), v_{\text{long}}, \dot{\psi}, \vec{d}_{\text{laser}}], \tag{3.9}$$

where $\vec{d}$ denotes the vector from the current robot position to the goal, ψ the yawing angle of the robot and v_{long} the longitudinal velocity, assuming zero slip. The environment is encoded in $\vec{d}_{\text{laser}}$ which is a 120-dimensional vector containing the 1D range measurements

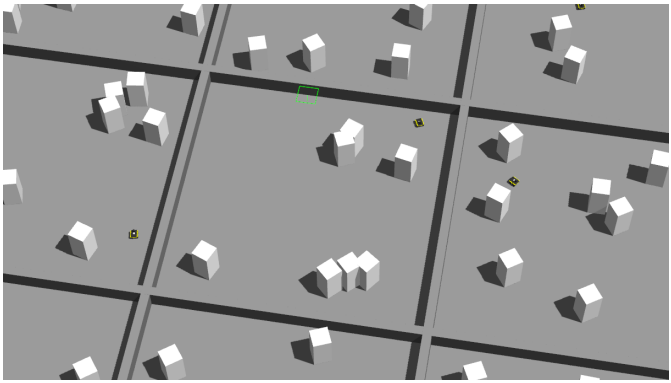


Figure 3.8: Gazebo Gym. This is an 8x8 m world where the task is to navigate a cluttered environment. The picture shows the training of PPO.

of each laser ray with small additive Gaussian noise. The laser scan has a field of view of 300° as depicted in Fig. 3.7. We first have been experiencing with the sparse reward setting in Eq. 3.8 in Point Robot Navigation. However, both SAC and PPO baseline performed poorly with success rates around 10% at 250k steps. To account for the more challenging dynamics of a differential drive robot, we implemented the dense reward

$$r_{\text{constraint\_free}} = \begin{cases} 40 & \text{if } ||\vec{d}||_2 < 0.1 \\ -0.1||\vec{d}||_2 & \text{else} \end{cases}, \quad r_{\text{constraint}} = -20\,.$$

The term $-0.1||\vec{d}||_2$ is a dense signal that directly connects the observation of the robot in Eq. 3.9 to the reward. The environment is implemented in Gazebo [204]. For PPO, we train 9 agents simultaneously, whereas for SAC, only one agent is used. The sampling time is 0.1s. At each iteration, the agent starts at a random position, and every forty iterations, a new set of obstacles is randomly spawned. In the hard mode, the agent is subject to Gaussian additive action noise. An example of the environment is shown in Fig. 3.8.

3.F Additional Experimental Results

We provide the training curves of SAC in Fig. 3.9 and of PPO in Fig. 3.10. Compared to the evaluation results in Table 1 and 2 of the main paper, the algorithms violate the constraints more frequently in training. However, this is expected since the randomness of the training policy can cause the agent to reach potentially dangerous states. Furthermore, in Fig. 3.10, we provide the soft constraint violation rates in Car Racing Safe. Interestingly, the agent drives faster with the deterministic policy such that the soft constraint is violated more frequently in evaluation.

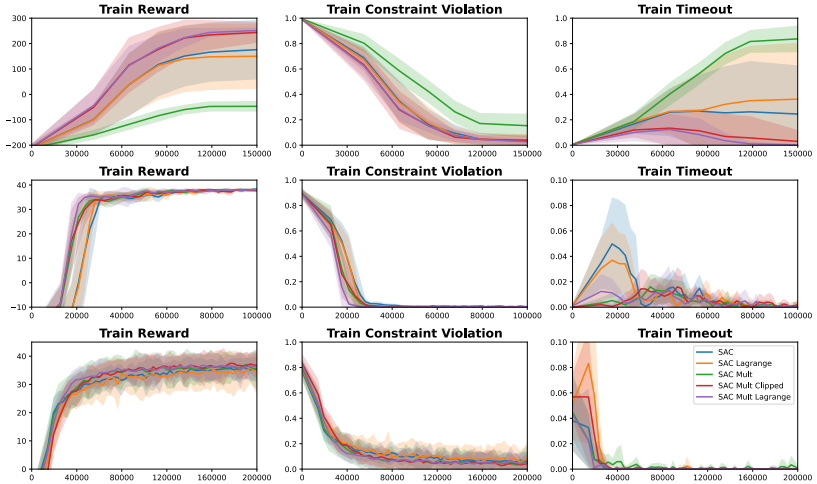


Figure 3.9: SAC training curves. Top to bottom: Lunar Lander Safe, Point Robot Navigation, Gazebo Gym.

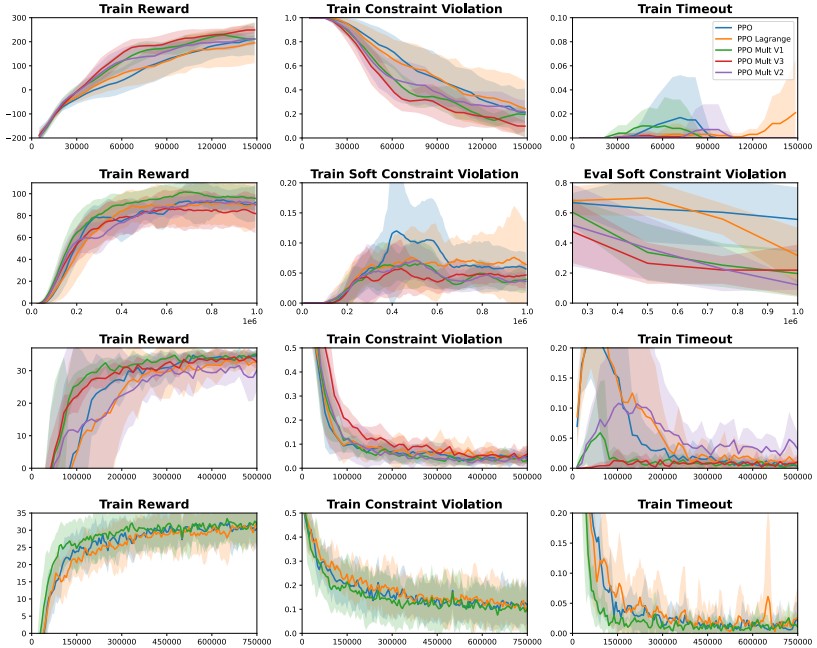


Figure 3.10: PPO training curves. Top to bottom: Lunar Lander Safe, Car Racing Safe, Point Robot Navigation, Gazebo Gym.

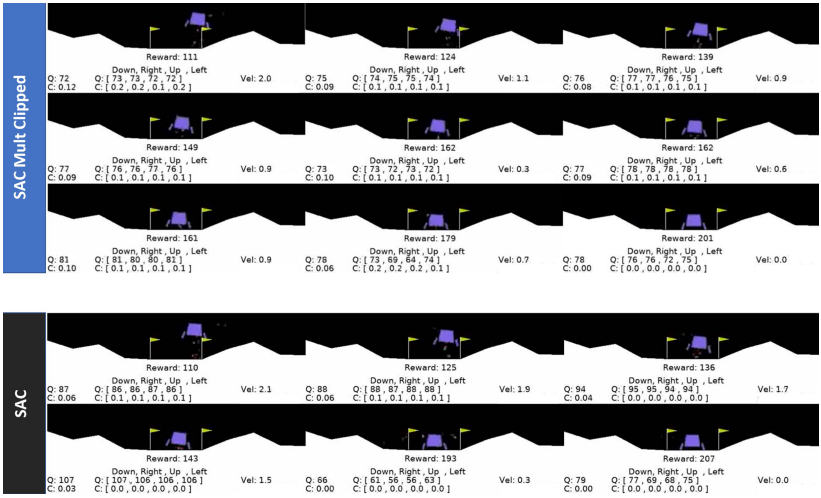


Figure 3.11: Qualitative comparison of SAC vs SAC Mult Clipped on seed six after 300k steps on Lunar Lander Safe. We depict every fifth frame of a one-episode video evaluation. The images have to be read from left to right, top to bottom. In each frame, the first column with Q and C denotes the estimated value and constraint violation probability of the next suggested action. The second column shows the action values and the constraint violation probabilities for basic actions like going up, left, right or down. The last column depicts the current downwards velocity of the lander.

We furthermore present qualitative results for Lunar Lander Safe, Car Racing Safe and Gazebo Gym. In Fig. 3.11, we compare the landing of SAC Mult Clipped against SAC baseline on Lunar Lander Safe. In the first row, the flights of baseline and Mult Clipped agent look similar. However, in the third frame, we see that the downward velocity of the baseline lander is 1.7 m/s, which is much higher than 0.9 m/s of the multiplicative version. When continuing with the baseline plot, second row, first image, we see that the lander slows down by 0.2 m/s and uses its right leg to establish ground contact. This makes the agent decelerate from 1.5 to 0.3 m/s, shown in the next frame. In contrast, the multiplicative agent lands more conservatively. Starting from the second row, the next five frames show the agent decelerating. Shortly before landing, the agent has a velocity of 0.7 m/s and lands with both legs simultaneously. This is much safer from a real-world perspective: Slowing down from 0.7→0 m/s with two legs compared to 1.5→0.3 m/s with one leg like the baseline.

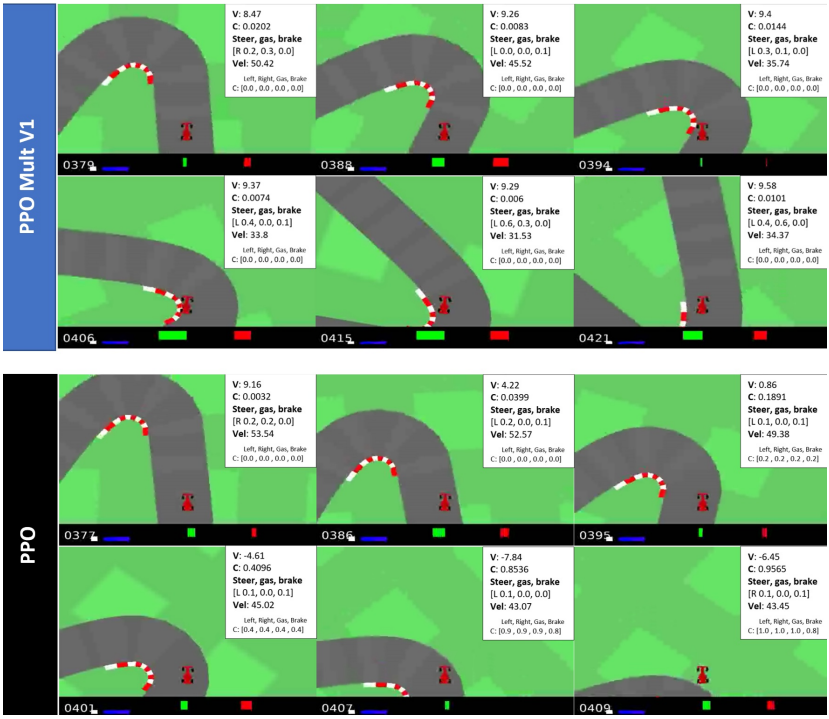


Figure 3.12: Qualitative comparison of PPO base vs PPO Mult V1 on seed six after 1M steps. We depict every fifth frame of a one-episode video evaluation. The images have to be read from left to right, top to bottom. In each frame, the first two rows with V and C denote the estimated value and constraint violation probability at the current state with the next suggested action. The third and fourth row depict the next suggested action. The last row contains the constraint violation probabilities for basic actions like steering left, right, accelerating and braking.

Next, we regard the first frame of Car Racing Safe in Fig. 3.12. Both the multiplicative and baseline agents steer to the right to open up the corner. When proceeding to the third frame, we can see that the baseline agent has opened up the corner more but also has a higher velocity of 49 km/h than the multiplicative agent. On the other side the multiplicative agent slows down to 36 km/h and cuts the inside. Continuing with frame four, we see that the velocity of the baseline agent is still high at 45 km/h. This limits the agent's ability to steer to the left without losing traction. At this point, the safety critic estimates a crashing probability of 40% (note that we train a safety critic also in the baseline version for visualization purposes, but to not use it in any way for the policy training and stop all possible gradients). As we proceed, the agent cannot make the corner and leaves the track. We now regard the multiplicative policy. In the fourth frame, we can see that the velocity is low at 34 km/h. This allows the agent to steer more aggressively to the left. In the next frame, the agent suggests further steering to the left

but also starts to accelerate. This is a real-world racing technique where one starts to accelerate after the apex of a corner to increase traction.

Finally, we switch to the Gazebo Gym results shown in Fig. 3.13 and start with the best case, where both SAC baseline and Mult Clipped achieve a success rate of 100%. Furthermore, the trajectories of the multiplicative version seem smoother and more natural compared to the baseline. Another difference is that our approach strictly drives with the front forward, whereas the baseline sometimes drives back first. This is shown in the velocity plots, where a velocity smaller than zero means driving back-first. This is potentially dangerous since the agent has a 60° blind spot in the back as shown in Fig. 3.7. Note that the behavior of driving back or front first is an emerging behavior that was not incentivized by the reward.

In the second-worst case, the behavior of occasionally driving back first causes the baseline to crash, as can be seen in the bottom right corner of the trajectory plot. On the other hand, our approach drives head-first and achieves a success rate of 96%. For the 4% of trajectories in which the agent crashes, there is an anomaly in the reward critic: the estimated return is overly optimistic, especially close to obstacles. The worst case is obtained with seed two, where the combination of driving back first and anomalies in the value functions cause baseline and multiplicative agents to crash every second trajectory. However, since the second-worse case achieves significantly higher performance, we consider this second seed an outlier.

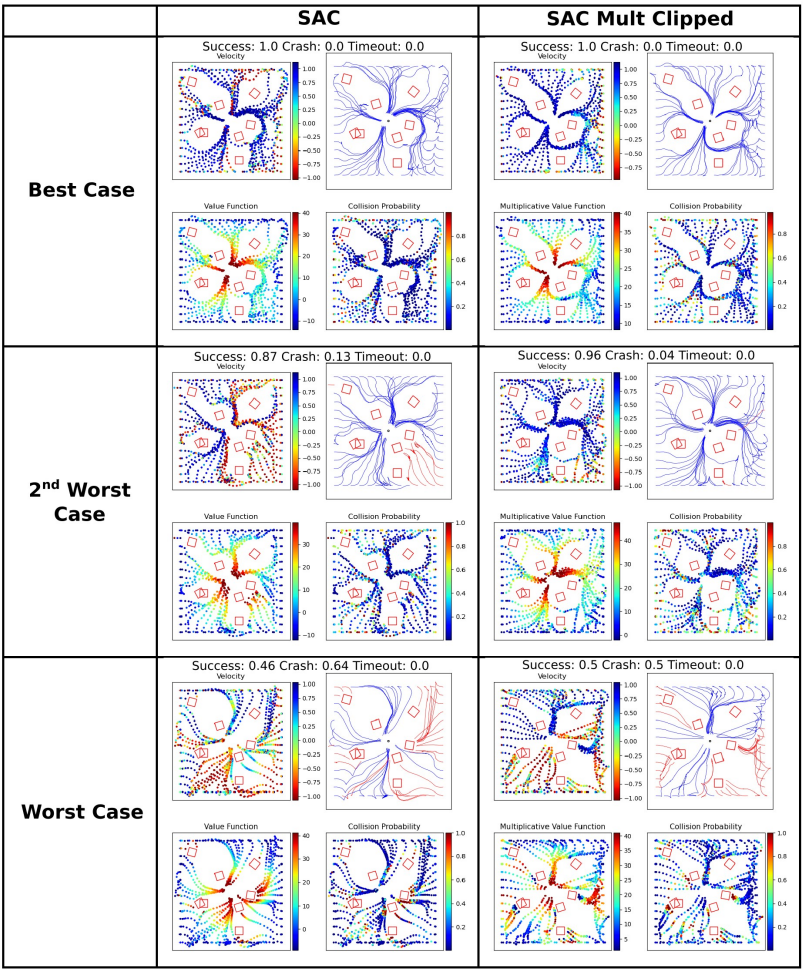
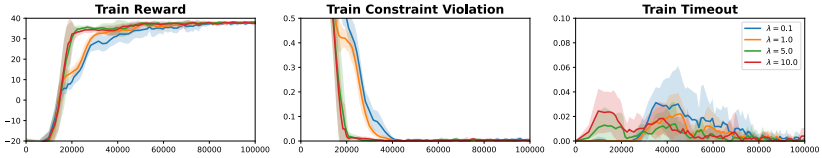


Figure 3.13: Qualitative results on Gazebo Gym with SAC after 200k steps. The Velocity, Value Function, and Collision Probability plots show the corresponding metric at each third trajectory point. The best and worst case distinction are with respect to the best and worst success rates out of ten seeds.

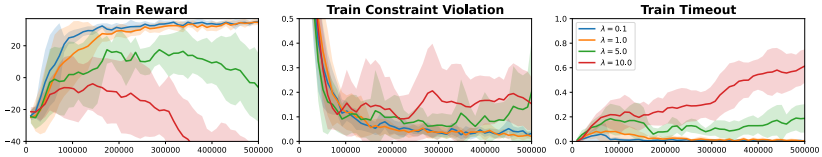
3.G Additional Ablation Studies

Finally, we want to show the effects of the choice of initial Lagrange multiplier λ_{init} and safety discount factor γ_c which is depicted in Fig. 3.14. For SAC in Fig. 3.14a, the different choices of the initial Lagrange multiplier do not significantly affect the performance at convergence but can result in different sample efficiencies. However, Fig. 3.14b shows PPO is more sensitive where multipliers with magnitude five and higher cause training instabilities with rising timeout rates. Note that in the stable-baselines3 implementation [185] upon which we build our code, the advantage in PPO is normalized whereas the reward in SAC is not. This means that the different magnitudes of the Lagrange multiplier have a greater effect on PPO than on SAC in Point Robot Navigation.

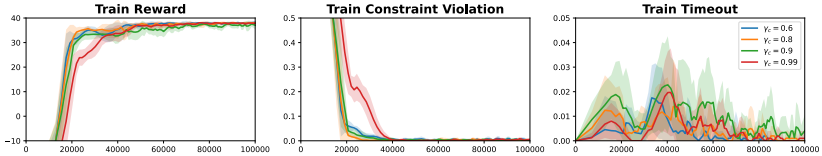
Furthermore, we experiment with different safety discount factors γ_c shown in Fig. 3.14c and 3.14d. Both for SAC and PPO the safety discount factor seems to have little effect on the performance. This can be explained by the fact that the "dynamics" of the point robot allow for an instantaneous change of direction such that a short safety horizon with low γ_c is sufficient for obstacle avoidance.



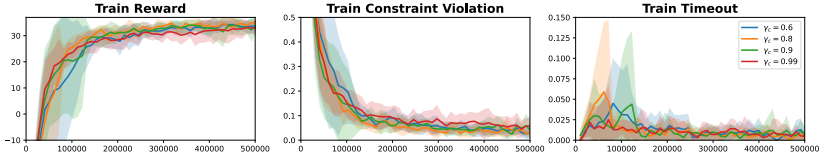
(a) Varying λ_{init} for SAC Mult Lagrange, $\gamma_c = 0.8$



(b) Varying λ_{init} for PPO Mult V1, $\gamma_c = 0.8$.



(c) Varying γ_c for SAC Mult Lagrange, $\lambda_{\text{init}} = 5.0$.



(d) Varying γ_c for PPO Mult V1, $\lambda_{\text{init}} = 0.1$.

Figure 3.14: Ablation experiments in point robot navigation.

4

TrafficBots: Towards World Models for Autonomous Driving Simulation and Motion Prediction

Data-driven simulation has become a favorable way to train and test autonomous driving algorithms. The idea of replacing the actual environment with a learned simulator has also been explored in model-based reinforcement learning in the context of world models. In this work, we show data-driven traffic simulation can be formulated as a world model. We present TrafficBots, a multi-agent policy built upon motion prediction and end-to-end driving, and based on TrafficBots we obtain a world model tailored for the planning module of autonomous vehicles. Existing data-driven traffic simulators are lacking configurability and scalability. To generate configurable behaviors, for each agent we introduce a destination as navigational information, and a time-invariant latent personality that specifies the behavioral style. To improve the scalability, we present a new scheme of positional encoding for angles, allowing all agents to share the same vectorized context and the use of an architecture based on dot-product attention. As a result, we can simulate all traffic participants seen in dense urban scenarios. Experiments on the Waymo open motion dataset show TrafficBots can simulate realistic multi-agent behaviors and achieve good performance on the motion prediction task.

4.1 Introduction

To realize autonomous driving (AD) in the urban environment, the planning module of autonomous vehicles has to address highly interactive driving scenarios involving human drivers, pedestrians and cyclists. Despite being a necessary step, the validation of planning algorithms on public roads is often too expensive and dangerous. Therefore, simulations have been widely adopted and efforts have been made to develop photo-realistic driving simulators [12]. While the full-stack simulators are popular for testing AD stacks and training visuomotor policies, they are not the best choice for developing planning

This chapter was published as a conference article: Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, Luc Van Gool, "TrafficBots: Towards World Models for Autonomous Driving Simulation and Motion Prediction", International Conference on Robotics and Automation (ICRA), 2023, doi: 10.1109/ICRA48891.2023.10161243

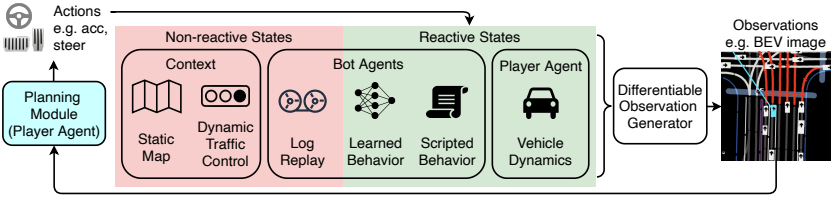


Figure 4.1: World model for AD planning modules. The simulator is fully data-driven and differentiable.

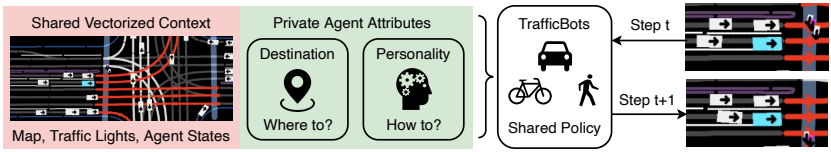


Figure 4.2: TrafficBots, a multi-agent policy that generates realistic behaviors for bot agents by learning from real-world data.

algorithms because the simulated scenarios are not as sophisticated and realistic as those encountered in the real world. Moreover, the computationally demanding rendering is redundant for AD planning modules that expect intermediate-level representations as input.

Therefore, simulators tailored for AD planning should have a different design and rely on real-world datasets. As shown in Fig. 4.1, the player agent, i.e., the planning module, generates a motion plan by observing some intermediate-level representations. Then the simulator updates its internal states and generates a new observation based on the actions taken by the player agent. The internal states of the simulation can be separated into two categories depending on whether they are reactive to the player agent. The scenario contexts, including the map and traffic controls, are non-reactive states loaded from the datasets. The states of the player agent are reactive and can be updated using vehicle dynamics. Of the most importance for the simulation fidelity are the bot agents, i.e., the non-player agents. The behaviors of bot agents fall into three categories: the non-reactive log-replay, the scripted behavior based on heuristics, and the learned behavior which is our focus. To generate human-like behaviors for bot agents, we present TrafficBots, a multi-agent policy built upon two established research fields: multi-modal motion prediction and end-to-end (E2E) driving.

As shown in Fig. 4.2, the TrafficBots policy is conditioned on the *destination* of each agent, which approximates the output of a navigator available in the problem formulation of E2E driving [35]. To learn diverse behaviors from demonstrations, each TrafficBot has a *personality* learned using conditional variational autoencoder (CVAE) [207] following multi-modal motion prediction. Compared to other methods, TrafficBots consume less memory, scale to more agents, and run faster than real time. This is achieved by using

a vectorized representation [158] for the context and sharing it among all bots. A new scheme of positional encoding (PE) is introduced for angles such that the memory-efficient dot-product attention can be used to retrieve local information from the shared context that lies in the global coordinate.

Using TrafficBots and a differentiable observation generator, the simulator in Fig. 4.1 is fully differentiable and it summarizes the player agent's past experience, hence it can be trained and used like a *world model* [117]. In this paper we focus on the TrafficBots and leave the training of player agents as future work. We evaluate TrafficBots on both the simulation and motion prediction tasks. We show that motion prediction can be formulated as the *a priori simulation*, hence it is a legit surrogate task for the evaluation of simulation fidelity. While prior works on traffic simulation introduce their own metrics, baselines and datasets, evaluation with motion prediction ensures an open and fair comparison. Although our performance is not comparable to the state-of-the-art open-loop methods, TrafficBots shows the potential of solving motion prediction with a closed-loop policy.

Our contributions are summarized as follows: We address data-driven traffic simulation using world models and we present TrafficBots, a multi-agent policy built upon motion prediction and E2E driving. We improve the simulation configurability by introducing the navigational destination and the latent personality, as well as the scalability by introducing a new PE for angles. Based on the public dataset and leaderboard, we propose a comprehensive and reproducible evaluation protocol for traffic simulation. Our repository is available at `https://github.com/zhejz/TrafficBots`

4.2 Related Work

World models [117] are action-conditional dynamics models learned from observational data. As a differentiable substitute of the actual environment, world models can be used for planning [118] and policy learning [119]. In this paper, we use world models to address a new problem: traffic simulation. We seek to obtain a world model realistic enough to replace the real world or full-stack simulators for developing AD planning algorithms. Training world models is often formulated as a video prediction problem such that the method can generalize to all image-based environments, like Atari [121] and highway driving [122]. Although the same approach can be applied to urban driving via rasterization, this would cause unnecessary complexity because most dynamics of driving can be explicitly modeled without deep learning. In fact, only the decision-dynamics of the bot agents that have a potential to interact with the player agent have to be learned. To this end, we introduce the multi-agent policy TrafficBots and based on it we build a world model for AD planning.

Motion prediction for AD is a popular research topic. Here we only discuss the most relevant works and refer the reader to [166] for a detailed review. Our TrafficBots use a network architecture based on Transformers [159] and vectorized representations [158] because they achieve top performance [162, 208] while being computationally more efficient [209]. To improve the multi-agent performance, our Transformer-based architecture uses a new PE for angles. Goal-conditioning can improve the performance of AD planning [210, 211] and motion prediction [141, 142, 143, 144], but it leads to causal confusions

if applied to closed-loop policy. This problem is solved by replacing the goal, which is associated with the prediction horizon, with the destination, which is time-independent and emulates a navigator. Once conditioned on the destination, the behavior of TrafficBots agent is characterized by a time-invariant personality. The personality is represented as the latent variable of a CVAE, which is used to address the multi-modality of motion forecasting [145, 146, 147, 148]. Unlike other works, we use a time-invariant personality, i.e., a fixed sample is used throughout the simulation horizon. Finally, TrafficBots is related to [155, 212, 213] in the sense that a recurrent policy is learned and combined with vehicle dynamics. However, our method is recurrent and closed-loop, whereas motion prediction methods are open-loop.

Data-driven simulation can reduce the sim-to-real gap while being more efficient and scalable than manually developing a simulator. While many works on data-driven simulation focus on the photo-realism [44, 91, 92], we study the behavior-realism of bot agents. Compared to the hand-crafted rules [12, 97, 98], more realistic behaviors can be generated through log-replay [99, 100] or learning from demonstrations [106]. The problem of learning realistic behaviors is formulated as generative adversarial imitation learning [49] in [105], as behavioral cloning in [109] and as flow prediction in [110]. Most related to our method is TrafficSim [18], an auto-regressive extension of the motion prediction method ILVM [146]. Compared to our method, TrafficSim is not based on world models or E2E driving, it uses rasterization and it does not factorize the uncertainty into personality and destination. Finally, our simulation shown in Fig. 4.1 can be considered a data-driven extension of SMARTS [101], and TrafficBots shown in Fig. 4.2 can be used as a sub-module to control bot agents in other simulators [12, 98, 101].

4.3 Problem Formulation

We use motion prediction datasets to train a policy, which can be used for simulation if a complete episode is given, and for motion prediction if only the history is available.

Data representation. Each episode in the motion prediction dataset includes the static map $\mathbf{M} \in \mathbb{R}^{N_\mathrm{M} \times N_\mathrm{node} \times 4}$, traffic lights $\mathbf{C} \in \mathbb{R}^{T \times N_\mathrm{C} \times 4}$, agent states $\hat{\mathbf{s}} \in \mathbb{R}^{T \times N_\mathrm{A} \times 6}$ and agent attributes $\mathbf{u} \in \mathbb{R}^{N_\mathrm{A} \times 4}$, where N_M is the number of map polylines, N_node is the number of nodes per polyline, N_C is the number of traffic lights and N_A is the number of agents. We define $t = 0$ to be the current step, T_h to be the history length and T_f to be the future length. A polyline node or a traffic light is represented by (x, y, θ, u) where x, y are the positions, θ is the yaw angle and u is the polyline type or light state. The ground truth (GT) state of agent i at step t is denoted by $\hat{\mathbf{s}}_t^i = (x, y, \theta, \dot{\theta}, v, a)$ where $\dot{\theta}$ is the yaw rate, v is the speed and a is the acceleration. The time-invariant agent attribute $\mathbf{u}$ includes the agent size and type of each agent. We use a scene-centric, vectorized representation [162] to ensure the efficiency of the simulation.

Simulation. We denote the states of TrafficBots agents as $\mathbf{s}$ and the states of other agents, including the player and other bots, as $\mathbf{s}^\dagger$. Given a complete episode, we initialize the simulation with the history $t \in [-T_\mathrm{h}, 0]$ and rollout for the future steps $t \in [1, T_\mathrm{f}]$. We assume all uncertainties can be explained by the GT future, thus the simulation can be formulated as predicting a single-modal next state $\mathbf{s}_{t+1}$ given $\mathbf{s}_t$ and $\mathbf{s}_t^\dagger$. Given the GT future, the simulation has two formulations: counterfactual and a posteriori. In

counterfactual simulation the behavior of some agents, e.g. the player agent, might deviate from the GT, i.e., $\mathbf{s}^{\dagger} \neq \hat{\mathbf{s}}^{\dagger}$. In this case TrafficBots should be reactive to the change and behave naturally. In the second case, if all agents are either controlled by TrafficBots or log-replay agents, the simulation should ideally reconstruct the same episode. In the spirit of world models, we refer to this as the *a posteriori simulation*.

Motion prediction. We formulate motion prediction as the *a priori simulation*, a special case of the a posteriori simulation where TrafficBots control all agents and the GT is given for $t \in [-T_{\mathrm{h}}, 0]$. In this case, rolling out for $t \in [1, T_{\mathrm{f}}]$ is equivalent to predicting $\mathbf{s}_{1:T_f}^{1:N_{\mathrm{A}}}$, the joint future of all agents. Since the GT future is unavailable and multiple futures are possible given the same history, the a priori simulation is multi-modal and each rollout represents one possible way of how the scenario could evolve. In fact, a priori simulation is equivalent to the multi-modal joint future prediction, which is a more difficult and hence less common task in comparison to the multi-modal marginal motion prediction that considers the prediction independently for each agent.

4.4 TrafficBots

As shown in Fig. 4.3, the TrafficBots policy is conditioned on shared and private contexts which are encoded beforehand and explain all uncertainties, thus the rollout is deterministic.

4.4.1 Policy

The policy predicts agent states at the next step $\mathbf{s}_{t+1}$, based on the current states $\mathbf{s}_t$ and the contexts. After encoding $\mathbf{s}_t$, the contexts are sequentially injected into the encoded states s_t. We use Transformer encoder layers with cross-attention to update s_t by attending to the encoded map $\mathcal{M}$ and the encoded traffic lights $\mathcal{C}_t$. The interaction Transformer uses self-attention across the agent dimension to allow agents to attend to each other. At inference time, states of non-TrafficBots agents $s_t^{\dagger}$ will also be processed by these Transformers such that TrafficBots can react to them. After incorporating the map, traffic lights and states of other agents, each agent has a recurrent unit to aggregate its history because the simulation states are not Markovian. Then the outputs are combined with the agent's individual destination and personality via concatenation and residual MLP. Finally, the actions of each agent are predicted by the action heads and $\mathbf{s}_{t+1}$ is computed by the dynamics module based on the actions and $\mathbf{s}_t$.

4.4.2 Contexts

State encoder. Following [162], all shared contexts, i.e., the map $\mathbf{M}$, traffic lights $\mathbf{C}$ and agent states $\mathbf{s}$, are represented in the global coordinates and incorporated via dot-product attention. This approach is computationally more efficient than transforming the global information to the local coordinate of each agent. However, the dot-product attention alone cannot efficiently model a global to local coordinate transform. To remedy this

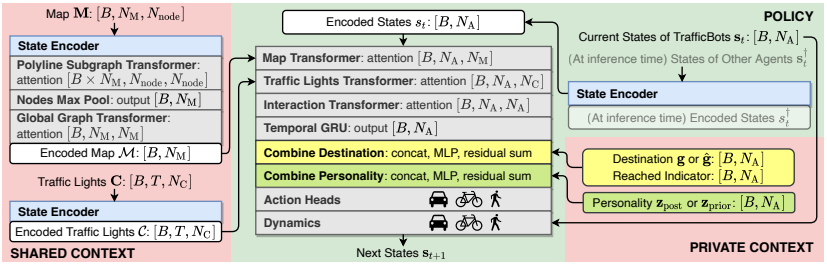


Figure 4.3: Network architecture of TrafficBots. In the brackets are the tensor shapes where B is the batch size. The hidden/feature dimensions are omitted for conciseness. The shared and private contexts are encoded only once at the start of an episode.

issue, PE is introduced. Without PE, VectorNet [158] has to transform all contexts to the local coordinate of each agent. SceneTransformer [162] concatenates the PE for position with the unit vector for direction and other attributes u, and then feeds it to an MLP:

$$s = \text{MLP}(\text{cat}(\text{PE}(x), \text{PE}(y), \cos\theta, \sin\theta, u)), \tag{4.1}$$
$$\text{with } \text{PE}_{2i}(x) = \sin(x \cdot \omega^{\frac{2i}{d_{\text{emb}}}}),\ \text{PE}_{2i+1}(x) = \cos(x \cdot \omega^{\frac{2i}{d_{\text{emb}}}}),$$

where $i \in [0, \ldots, d_{\text{emb}}/2]$, ω is the base frequency and d_{emb} is the embedding dimension. This state encoder can be improved by using PE also for the direction vector [214] and adding the PE after the MLP [159]. This ends up with

$$s = \text{cat}(\text{PE}(x), \text{PE}(y), \text{PE}(\cos\theta), \text{PE}(\sin\theta)) + \text{MLP}(u). \tag{4.2}$$

However, empirically we observe TrafficBots using this state encoder is not sensitive to directional information. To address this issue, we propose the following state encoder

$$s = \text{cat}(\text{PE}(x), \text{PE}(y), \text{AE}(\theta), \text{MLP}(u)) \tag{4.3}$$
$$\text{with } \text{AE}_{2i}(\theta) = \sin(\theta \cdot i),\ \text{AE}_{2i+1}(\theta) = \cos(\theta \cdot i)$$

where $i \in [1, \ldots, d_{\text{emb}}/2 + 1]$ and AE stands for angular encoding, a special case of sinusoidal PE we introduced to encode the radian yaw θ. Compared to PE that has to use a small ω to avoid overloading the 2π period, AE can use integer frequency because it encodes an angle. Moreover, the addition is replaced by concatenation because other states, e.g. velocity, are highly correlated to the pose encoded by PE and AE. We use the state encoders to encode the map, traffic lights and agent states. Our map encoder follows [158], except that we use Transformers for the polyline sub-graph.

Destination. Fig. 4.4 highlights the difference between our destination and the goal proposed in prior works [141, 142, 143, 144]. The GT goal, which is associated to the last observed position, does not reflect an agent's intention. In Fig. 4.4, the vehicle stops because of the red light, whereas the pedestrian does not intend to stay in the middle of a crosswalk. Although this is not a problem for open-loop motion prediction, the driving policy would learn a wrong causal relationship if conditioned on the goal. This

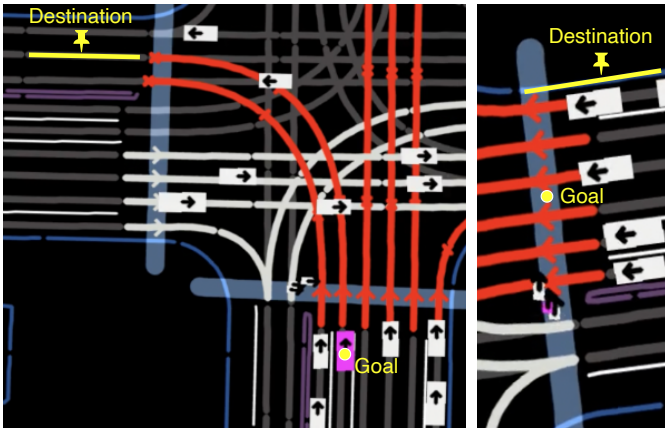


Figure 4.4: GT destination and goal of the magenta agent.

problem can be solved by introducing a navigator, which specifies the next goal once the current one is reached. However, running an online navigator for every agent is computationally demanding. For simulation with a short horizon and small maps, it is sufficient to estimate one destination for the near future, and switch to an unconditioned policy once that destination is reached. Since the GT destination is not available in any motion prediction datasets, we approximate it with a map polyline heuristically selected by extending the recorded agent trajectory based on the map topology. For training and simulation we use the approximated GT destination $\hat{\mathbf{g}}$, whereas for motion prediction we predict $\mathbf{g}$. Predicting the destination is formulated as a multi-class classification task where the logit for polyline i and agent j is predicted by $\text{MLP}(\text{cat}(\mathcal{M}^i, \text{GRU}(s^j_{-T_h:0})))$, i.e., the destination of an agent depends only on the map and its own history.

Personality. In order to address the remaining uncertainties not explained by the destination and to learn diverse behaviors of different human drivers, pedestrians and cyclists, we introduce a latent personality for each agent which is learned using CVAE. Similar ideas have been applied to world models [117, 118, 119] and motion prediction [145, 146, 147]. The personality encoder has a similar architecture as the policy network in Fig. 4.3. For training and simulation, we use the posterior $\mathbf{z}_{\text{post}}$ which is estimated from the complete episode $t \in [-T_h, T_f]$, whereas for motion prediction we use the prior $\mathbf{z}_{\text{prior}}$ that encodes only the history $t \in [-T_h, 0]$. In contrast to TrafficSim [18] which updates the latent at each time step to address all uncertainties, our personality is time-invariant because the behavioral style of an agent will not change in a short time horizon if the destination is determined.

4.4.3 Training

Similar to world models [117], our training uses reparameterization gradients and back-propagation through time (BPTT). Given a complete episode, we first encode the map $\mathbf{M}$, traffic lights $\mathbf{C}$ and GT agent states $\hat{\mathbf{s}}$. Then we predict $\mathbf{z}_{\text{post}}$, $\mathbf{z}_{\text{prior}}$ and the destination $\mathbf{g}$. Conditioned on the GT destination $\hat{\mathbf{g}}$ and a sample of $\mathbf{z}_{\text{post}}$ we rollout the policy. For $t \in [-T_{\text{h}}, 0]$ we warm-start using teacher-forcing with GT agent states, whereas for $t \in [1, T_{\text{f}}]$ the rollout is auto-regressive. All components are trained simultaneously using the weighted sum of three losses: the reconstruction loss with smoothed L1 distance for the states (x, y, θ, v), the KL-divergence between $\mathbf{z}_{\text{post}}$ and $\mathbf{z}_{\text{prior}}$ clipped by free nats [118], and the cross-entropy loss for destination prediction. Following [119], we stop the gradient from the action and allow only the gradient from the states during the BPTT. We train with all agents so as to generate realistic behaviors for all traffic participants, not just for the interested ones heuristically selected by the dataset.

4.4.4 Implementation Details

We use a 16-dim diagonal Gaussian for the personality. The action heads and dynamics have the same architecture but different parameters for vehicles, cyclists and pedestrians. We use a unicycle model with constraints on maximum yaw rate and acceleration for all types of agents. With a hidden dimension of 128 our model has less than 3M parameters. Considering 64 agents, 1024 map polylines and a sampling time of 0.1 second, we can parallelize 16 simulations on one 2080Ti GPU while each rollout step takes around 10 ms, which is a magnitude faster than other methods [18, 105, 109, 110].

4.5 Experiments

Dataset. We use the Waymo Open Motion Dataset (WOMD) [134] because compared to other datasets it has longer episode lengths and more diverse and complex driving scenarios, such as busy intersections with pedestrians and cyclists. The WOMD is also one of the largest motion prediction datasets, consisting of 487K episodes for training, 44K for validation and 45K for testing. With a fixed sampling time of 0.1 second, each episode is 9 seconds long and contains 91 steps: $T_{\text{h}} = 10$ for the history, one for the current $t = 0$, and $T_{\text{f}} = 80$ future steps that shall be predicted.

Tasks. Ultimately we want to verify the fidelity of the *counterfactual simulation*, such that the simulator can be used for training and testing planning modules. However, once the scenario diverges from the factual recording, the GT trajectories can no longer be used for evaluation metrics. To this end, different surrogate metrics have been proposed, such as traffic rule compliance [18] and distribution of curvatures [105]. But these metrics cannot fully reflect the behavioral fidelity because they consider only vehicles and neglect pedestrians and cyclists. Moreover, performing well on these metrics does not mean the behavior is human-like, in fact good performance can be achieved by a hand-crafted policy. Alternatively we can verify the fidelity of the *a posteriori simulation*, where the scenario should be reconstructed and the performance can be quantified by the distance to the

test	mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓	overlap rate ↓
DenseTNT [142]	0.328	1.039	1.551	0.157	0.178
SceneTransformer [162]	0.279	0.612	1.212	0.156	0.147
MultiPath [149] static	0.236	0.880	2.044	0.345	0.166
Waymo LSTM [134]	0.176	1.007	2.355	0.375	0.190
TrafficBots (a priori)	0.212	1.313	3.102	0.344	**0.145**
valid TrafficBots	↑	↓	↓	↓	↓
a priori (*K=6*)	0.210	1.291	3.117	0.346	0.143
GT sdc future (*what-if*)	0.214	1.281	3.095	0.342	0.142
GT traffic light (*v2x*)	0.209	1.288	3.100	0.345	0.143
GT destination (*v2v*)	0.217	1.292	3.123	0.345	0.142
a posteriori (*K=1*)	0.332	0.962	2.034	0.339	0.129

Table 4.1: Results on the WOMD (marginal) leaderboard.

GT trajectories. But since the GT future is given, a model can achieve good performance by misusing the posterior latent to memorize the GT future, instead of learning the underlying human-like behavior. In fact, the best possible performance can be simply achieved via log-replay. We argue the *a priori simulation*, i.e., motion prediction, together with the a posteriori simulation is a better evaluation setup. For a priori simulation, the model predicts multiple futures of how an episode might evolve. While all predictions should demonstrate natural behaviors, at least one of them should reconstruct the GT future. Importantly, motion prediction is usually formulated as an open-loop problem. Although TrafficBots can be used for motion prediction by formulating it as the a priori simulation, the performance will be affected by the covariant shift and compounding errors [11] caused by the closed-loop rollout. Nevertheless, we show the potential of solving motion prediction with a multi-agent policy.

Metrics. For motion prediction we follow the metrics of WOMD [134], including the accuracy metrics mAP, the distance-based minADE/FDE and miss rate, and the surrogate metric overlap rate. Inspired by [148], we further examine the sampling-based negative log-likelihood (NLL) of the GT scene. The WOMD specifies up to 8 agents that shall be predicted and allows up to K=6 predictions. Accordingly, we generate 6 rollouts, i.e., the joint future of all agents, by sampling the destination and the prior personality. For a posteriori simulation, only one rollout is generated using the most likely posterior personality and the GT destination. The simulation fidelity is evaluated using traffic rule violation rate and distance to GT trajectories. The differences in position and rotation are averaged over all steps and agents, whereas the rates of collision, running a red light and passiveness (stop moving for no reason) are for vehicles only.

Comparison with motion prediction methods. In the first half of Table 4.1 we compare TrafficBots with open-loop motion prediction methods on the Waymo (marginal) motion prediction leaderboard. In terms of mAP we are better than the Waymo LSTM baseline [134], but worse than other methods because TrafficBots is not optimized to generate diverse predictions which is favored by the mAP metrics. Although the miss rates are comparable, the minADE/FDE of our method are significantly higher than other methods. This can be explained by the compounding errors caused by the auto-regressive policy rollout. While this drawback is well-known for closed-loop methods, TrafficBots still has its advantage which is shown by the reduced overlap rate. Compared to the open-loop methods, it is easier for a policy to learn the correct causal relationship. The second half of Table 4.1 shows that the prediction performance can be improved given additional information. Since the predictions are generated via rollout, we can set some of the future observations to their GT. For example, for conditional motion prediction (*what-if*) the future trajectory of the self-driving-car is given. Furthermore, the future traffic light states and the destinations could be obtained via vehicle-to-everything (*v2x*) or vehicle-to-vehicle (*v2v*) communication. Having access to all future information, the a posteriori simulation achieves the best performance with a single (*K=1*) prediction.

Ablations. We ablate the state encoders, personality, destination and world-model training techniques on the a priori simulation in Table 4.2, and on the a posteriori simulation in Table 4.2. Our state encoder Eq. 4.3 with AE performs overall better than *Eq. 4.1* and *Eq. 4.2*. *Without the personality*, the policy is unable to capture the diverse behaviors of different traffic participants. If we allow a *larger KL* divergence by downweighting the KL loss, the performance is better for a posterior simulation but worse for motion prediction. Then we have TrafficBots *w/o destination* where the latent captures all uncertainties. In this case the model performs worse on motion prediction because the Gaussian latent suffers from mode averaging. If the policy is conditioned on the *goal*, i.e., the polyline associated with the last observed pose, then the model will learn a wrong causal relationship and the traffic rule violation rates will increase even though the minADE/FDE are smaller. If we use the *goal w/o navigator* module that drops the goal once it is reached, the policy learning will fail completely and the performances are overall inferior. Finally, we show world-model training techniques can improve the performance of TrafficBots. To further compare with prior works on traffic simulation, we ablate more design differences between our method and SimNet [109], which uses behavioral cloning (BC) without personality or destination, as well as TrafficSim [18]. Generally, TrafficBots performs better but there are three interesting exceptions: Firstly, if we allow a *larger KL* or *resample pers.*, the posterior will memorize the GT future and the prior will fail to infer the personality. Consequently the model performs better for a posterior simulation but worse for motion prediction, and the traffic rule violation rates are higher because the model masters the memorization rather than the driving skills. This highlights the importance of using a time-invariant personality and the advantage of evaluating with both a priori and a posteriori simulation. Secondly, models without personality have smaller NLL. This is reasonable because models without CVAE generate less diverse predictions, hence the NLL is smaller. Finally, the model with an *interactive decoder* following TrafficSim [18] shows a smaller miss rate during a posteriori simulation. This is achieved by adding the private contexts before the interaction Transformer, such that private contexts are shared among all agents. However, this requires the personality

		mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓	overl. rate ↓	NLL ↓ ($\times 10^{-7}$)
Our best	TrafficBots	**0.18**	1.49	3.66	**0.39**	**0.15**	1.37
Encoder	Eq. 4.1	0.12	1.74	4.48	0.48	0.18	1.90
	Eq. 4.2	0.14	1.62	4.12	0.46	0.17	1.48
Personality	w/o persona	0.06	1.66	4.09	0.48	0.15	1.16
	larger KL	0.15	1.65	4.19	0.42	0.17	1.88
Destination	w/o dest.	0.16	1.53	3.80	0.40	0.15	1.44
	goal	0.17	**1.47**	**3.44**	0.40	0.16	2.02
	goal w/o navi	0.14	1.57	3.83	0.45	0.17	3.39
World Model	w/o free nats	0.18	1.52	3.74	0.40	0.16	1.39
	w/ action grad.	0.17	1.51	3.71	0.41	0.16	1.39
SimNet	BC w/o pers. & dest.	0.01	2.76	7.77	0.76	0.21	2.64
	w/o pers. & dest.	0.02	1.91	4.95	0.55	0.15	**1.10**
	BC	0.09	3.11	9.24	0.73	0.21	3.34
TrafficSim	w/o dynamics	0.14	1.81	4.37	0.46	0.17	1.68
	inter. decoder	0.17	1.52	3.73	0.41	0.16	1.66
	resample pers.	0.14	1.81	4.74	0.47	0.16	1.56

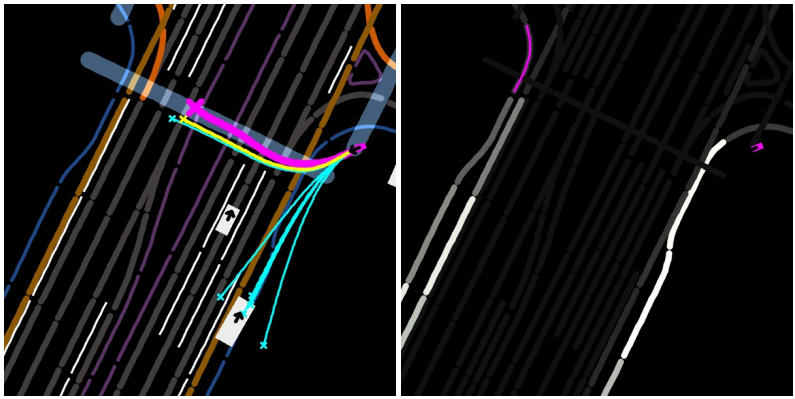
Table 4.2: Ablation on the WOMD validation split, a priori simulation K=6 (motion prediction). All models are trained for 24K iterations (48 hours).

and destination of all agents to be known before the rollout, which is infeasible if the simulation includes a player agent whose future actions are undetermined.

		dif. pos (m) ↓	dif. rot (deg) ↓	veh col (%) ↓	run red (%) ↓	passive (%) ↓	miss rate ↓
Our best	TrafficBots	0.80	2.84	**11.5**	1.31	19.1	0.42
Encoder	Eq. 4.1	0.74	3.05	14.7	1.47	19.4	0.49
	Eq. 4.2	0.74	3.02	13.8	1.46	19.3	0.48
Personality	w/o persona	1.29	3.63	13.6	1.50	19.2	0.53
	larger KL	**0.47**	**2.39**	12.9	1.56	19.1	0.24
Destination	w/o dest.	0.74	2.63	11.8	**1.29**	19.3	0.41
	goal	0.78	2.68	12.3	1.35	20.2	0.42
	goal w/o navi	0.79	2.97	15.1	1.40	23.3	0.49
World Model	w/o free nats	0.86	3.00	12.6	1.31	19.1	0.44
	w/ action grad.	0.90	2.82	12.6	1.30	19.1	0.46
SimNet	BC w/o pers. & dest.	2.27	7.37	21.9	1.59	19.6	0.76
	w/o pers. & dest.	1.34	3.69	13.6	1.46	19.2	0.54
	BC	2.99	7.56	33.4	4.27	19.3	0.76
TrafficSim	w/o dynamics	0.72	55.18	48.0	1.73	**18.9**	0.45
	inter. decoder	0.75	2.85	12.8	1.46	19.2	**0.22**
	resample pers.	0.49	2.45	12.8	1.55	19.5	0.29

Table 4.3: Ablation on the WOMD validation split, a posteriori simulation K=1. All models are trained for 24K iterations (48 hours).

(a) A vehicle entering the parking lots.



(b) A cyclist crossing the road through the crosswalk.

Figure 4.5: Qualitative results of TrafficBots. In each sub-figure, left: predicted trajectories; right: heat map of predicted destinations. Agent of interest and GT are in magenta. A priori predictions are in cyan. A posteriori simulated trajectory is in yellow. The brightness is proportional to the probability in the destination heat map.

Qualitative results. Fig. 4.5 shows two examples of the prediction and simulation results. In both cases, one of the a priori predictions matches the GT, whereas the a posteriori simulation reconstructs the scenario with less deviation. With similar destinations but sampled personalities, five predictions in Fig. 4.5a follow the lane with different speeds and lane selections. With predicted destinations on both sides of the road, the cyclist in Fig. 4.5b is predicted to either cross the road or follow the road edge.

4.6 Conclusions and Future Works

This paper presented TrafficBots, a multi-agent policy learned from motion prediction datasets. Based on the shared, vectorized context and the individual personality and destination, TrafficBots can generate realistic multi-agent behaviors in dense urban scenarios. Besides the simulation, TrafficBots can also be used for motion prediction. Evaluating on motion prediction tasks allows us to verify the simulation fidelity and benchmark on a public leaderboard. Based on TrafficBots, we build a differentiable, data-driven simulation framework, which in the future can serve as a platform to develop AD planning algorithms, or as a world model to train E2E driving policies via reinforcement learning [35] or model-based imitation learning [99]. Moreover, TrafficBots could also be integrated as a module to generate human-like behaviors for bot agents in a game or a full-stack AD simulator. Future work will investigate better network architectures and training techniques, the downstream tasks, and combining data-driven traffic simulation with neural rendering.

Appendices

4.A Supplementary Video

The supplementary video for the paper is found here at `https://youtu.be/2idvJ0qbXeo`. This video contains more experimental results generated by TrafficBots. The video is nicely edited and exhaustively commented. It includes two episodes, the first episode highlights a vehicle making an U-turn on a narrow street, whereas the second episode is at a busy intersection with traffic lights and a large number of traffic participants. For each episode, we first show the results of a posteriori simulation and a priori motion prediction, and then we inspect agents demonstrating the most interesting behaviors. Besides the good cases, this video also presents the bad cases where our method failed to generated realistic behavior.

4.B Dataset and Pre-Processing

We use the unfiltered 9-second datasets (*scenario*, not the filtered *tf_example*) from the WOMD, these are: *testing*, *testing_interactive*, *training*, *validation*, *validation_interactive*. The WOMD also provides a full-length training dataset *training_20s*, which includes the original 20-second-long episodes. In contrast to the 9-second datasets which are clipped from the 20-second-long episodes, episodes in the *training_20s* do not always have a fixed length. Although we did not use the 20-second dataset, future works can take advantage of it for simulation with a longer time horizon. We pre-process the dataset by first filtering the map polylines:

1. Split the original map polylines into shorter polylines with maximum $N_{\text{node}} = 20$ nodes one meter away from each other.
2. Remove polylines too far away from any agents.
3. Remove polylines that contain too few nodes.
4. Continue removing polylines based on the distance to agents, until the number of remaining polylines is smaller than a threshold $N_{\text{M}} = 1024$.

Then we filter the traffic lights which are associated with map polylines. A traffic light will be filtered if its map polyline is removed. Finally we filter agents as follows:

1. Remove agents that are tracked for too few steps.
2. Remove agents that have small displacement and large distance to any of the relevant agents marked by the WOMD or any of the map polylines. These agents are mostly parking vehicles.
3. Remove vehicles that have small displacement but large yaw change, which are caused by tracking errors.
4. Continue removing irrelevant agents based on the distance to relevant agents, until the number of remaining agents is smaller than a threshold $N_{\text{A}} = 64$.

After the filtering, we center the episode such that the position of the self-driving-car is at $(0, 0)$. The training episodes are randomly rotated by an angle between $-\pi$ and π, whereas the validation and testing episodes are unaffected. We smooth the agent trajectories and fill in the missing steps via temporal linear interpolation. A pre-processed episode has $T = 91$ steps and includes the following data:

1. Agent states
   - *agent/valid*: $[T, N_{\text{A}}]$, Boolean mask.
   - *agent/pos*: $[T, N_{\text{A}}, 2]$, x, y positions.
   - *agent/vel*: $[T, N_{\text{A}}, 2]$, velocities in x, y directions.
   - *agent/spd*: $[T, N_{\text{A}}, 1]$, m/s
   - *agent/acc*: $[T, N_{\text{A}}, 1]$, m/s^2
   - *agent/yaw_bbox*: $[T, N_{\text{A}}, 1]$, rad.
   - *agent/yaw_rate*: $[T, N_{\text{A}}, 1]$, rad/s.
2. Agent attributes
   - *agent/type*: $[N_{\text{A}}, 3]$; vehicle, pedestrian, cyclist.
   - *agent/role*: $[N_{\text{A}}, 3]$, 3 types of role; self-driving-car, agent of interest, agent to predict.
   - *agent/size*: $[N_{\text{A}}, 3]$, length, width, height.
3. Map
   - *map/valid*: $[N_{\text{M}}, N_{\text{node}}]$, Boolean mask.
   - *map/type*: $[N_{\text{M}}, 11]$, 11 types of polylines. They are *freeway*, *surface_street*, *stop_sign*, *bike_lane*, *road_edge_boundary*, *road_edge_median*, *solid_single*, *solid_double*, *passing_double_yellow*, *speed_bump* and *crosswalk*.
   - *map/pos*: $[N_{\text{M}}, N_{\text{node}}, 2]$, x, y position of nodes.
   - *map/dir*: $[N_{\text{M}}, N_{\text{node}}, 2]$, a 2D vector pointing to the next node.
4. Stop point of traffic lights
   - *tl_stop/valid*: $[T, N_{\text{C}}]$, Boolean mask.
   - *tl_stop/state*: $[T, N_{\text{C}}, 5]$, 5 types of states; *unknown*, *stop*, *caution*, *go* and *flashing*.
   - *tl_stop/pos*: $[T, N_{\text{C}}, 2]$, position of the stop point.
   - *tl_stop/dir*: $[T, N_{\text{C}}, 2]$, direction of the stop point.

4.C Ground-Truth Destination

The GT destinations are not available in any motion prediction datasets. Therefore, we use the following heuristics to approximate the GT destination of an agent:

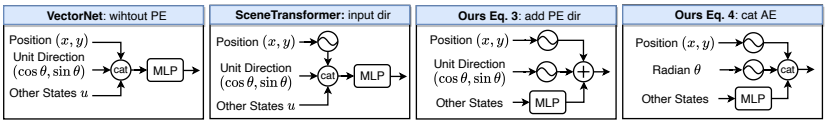


Figure 4.6: State encoders with different architectures.

- If the agent is a **vehicle on a lane**, i.e. the last observed pose of the agent is close enough to a driving lane in terms of position and direction, then we find the destination by randomly selecting one of the successors of that lane base on the map topology. This step will be repeated multiple times. These agents are vehicles driving on the road. In this case the type of the destination is either *freeway*, *surface_street* or *stop_sign*.
- If the agent is a **vehicle not on lane**, then we extend the last observed pose with constant velocity for 5 seconds. After that the *road_edge_boundary* polyline closest to that extended position will be selected. These agents are mostly vehicles in parking lots.
- For **cyclists on bike lanes**, we extend the last position with constant velocity and find the closest *bike_lane*.
- For **cyclists not on bike lanes** or **pedestrians**, we find the *road_edge_boundary* polyline closest to the position extended using constant velocity.

For the ablation we have a model trained with *goal* instead of destination. In this case the goals are still polylines and the GT goals are still approximated using the aforementioned method, with the exception that we do not extend the last observed position using map topology or constant velocity. The map polyline closest to the last observed position will be directly used as the *goal*. Unlike motion prediction methods [141, 142] that predict an accurate goal and then simply fit a smooth trajectory towards the goal, our destination is less informative such that the motion profile is determined solely by the policy. The destinations are pre-processed and saved as agent attributes $agent/dest : [N_A]$. We save the indices of the corresponding map polyline, hence the value of $agent/dest$ ranges from 1 to N_M.

4.D Detailed Network Architecture

We use dropout probability 0.1 and ReLU activation.

State Encoders. The architecture of the state encoders discussed in the main paper are visualized in Fig. 4.6.

Transformers. We use the Transformer encoder layer with cross-attention as shown in Fig. 4.7. The layer norm is inside the residual blocks [215]. If the query, key and value share the same tensor, then the cross-attention boils down to self-attention which is used by the interaction Transformer.

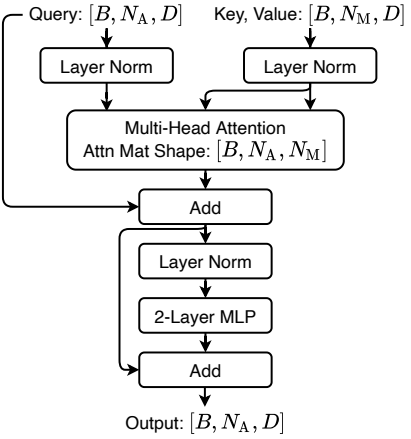


Figure 4.7: Transformer encoder layer with pre-layer-norm.

Combine Personality and Destination. As shown in Fig. 4.8, the personality, or the destination, is injected to the intermediate state via concatenation, MLP and residual sum. Since the personality is always valid, the masking is unnecessary for combining personality. In terms of destination, the masking is based on a reached indicator. If the destination is reached, then the output of the residual block will be masked such that the intermediate states remain unchanged and the destination no longer affects the policy.

Action Heads. We use a two-layer MLP to predict the acceleration and the yaw rate of each agent. We instantiate three action heads with the same architecture; one for each type of agent. The outputs of action heads are normalized to $[-1, 1]$ via the tanh activation.

Dynamics. Following MultiPath++ [166] we use a unicycle dynamics with constraints on maximum yaw rate and acceleration for all types of agents. For vehicles the acceleration is limited to $\pm 5\,\mathrm{m/s}$ and the yaw rate is limited to $\pm 1.5\,\mathrm{rad/s}$. For cyclists we use $\pm 6\,\mathrm{m/s}$, $\pm 3\,\mathrm{rad/s}$ and for pedestrians $\pm 7\,\mathrm{m/s}$, $\pm 7\,\mathrm{rad/s}$. The outputs of action heads are multiplied by the maximum allowed acceleration or yaw rate to obtain the final actions.

Personality Encoder. The inputs to the map, traffic lights and interaction Transformer of the personality encoder are reshaped differently. For the map encoder, we flatten the agent states tensor with shape $[B, T, N_A]$ to $[B, T \times N_A]$ and use it to query the map with shape $[B, N_M]$. This allows each agent at each time step to attend to the map independently. For the traffic lights Transformer, the agent states tensor with shape $[B, T, N_A]$ is flattened to $[B \times T, N_A]$ and the traffic lights with shape $[B, T, N_C]$ is flattened to $[B \times T, N_C]$. In this case, the agents states can only attend to the traffic lights from the same time step. Similarly, inputs to the interaction Transformer are reshaped from $[B, T, N_A]$ to $[B \times T, N_A]$, such that an agent can only attend to other agents' states from

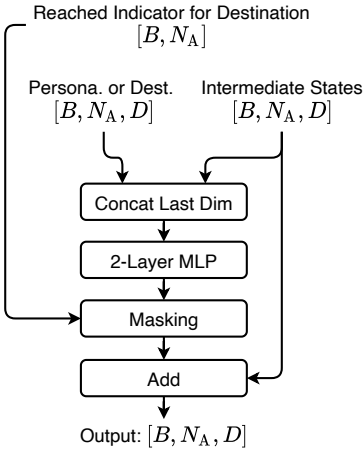


Figure 4.8: Combine personality/destination.

the same time step. We use two personality encoders with the same architecture to encode the posterior and the prior personality respectively.

Latent Distribution of Personality. The personality encoder predicts the mean of a 16-dimensional diagonal Gaussian for each agent. The standard deviation is a learnable parameter independent of any inputs. We initialize the log standard deviation to -2 for all the 16 dimensions. The standard deviation parameter is shared by agents from the same type (vehicle, pedestrian, cyclist).

Predicting Destinations. The destination of agent j depends only on the encoded map $\mathcal{M}$ and its own encoded history states $s^j_{-T_h:0}$. Given $\mathcal{M}^i$, the hidden feature of the ith polyline of the encoded map $\mathcal{M}$, the logit p^j_i for polyline i and agent j is predicted by

$$p^j_i = \text{MLP}(\mathcal{M}^i, \text{GRU}(s^j_{-T_h:0})), \text{ where } i \in \{1, \dots, N_M\}, j \in \{1, \dots, N_A\}.$$

Based on these logits, the destinations of agent j are represented by a categorical distribution with N_M classes and the probability is obtained via softmax. After obtaining the polyline index i, the predicted destination $\mathbf{g}$ is the encoded polyline feature $\mathcal{M}^i$ indexed by i. The polyline indices of the GT destinations are saved during the dataset pre-processing.

4.E Training Details

We use six 2080Ti GPUs for the training with a batch size of 4 on each GPU, i.e. the total batch size is $B = 24$. Due to the large size of the WOMD training dataset, in each epoch we randomly select 15% from the complete training and validation datasets. We

use the Adam optimizer with a learning rate of 4e-4. The learning rate is halved every 7 epochs. The model converges after about 30 epochs, that is almost a week. We predict the posterior personality $\mathbf{z}_{\text{post}}$ using the posterior personality encoder and information from $t \in [-T_{\text{h}}, T_{\text{f}}]$. Similarly $\mathbf{z}_{\text{prior}}$ is predicted using the prior personality encoder and information from $t \in [-T_{\text{h}}, 0]$. The logits of destinations are predicted using the encoded map $\mathcal{M}$ and the GT agent states $\hat{s}_{-T_{\text{h}}:0}$ from the past. From the logits we use softmax to obtain a multi-class categorical distribution of the destination of each agent $P_{\text{dest}}^{1:N_{\text{A}}}$, which has N_{M} classes; one for each map polyline. During the training we rollout with the GT destination and the posterior personality $\mathbf{z}_{\text{post}}$. Our training loss has the following terms:

1. Reconstruction loss, which trains the model to reconstruct the GT states using the posterior personality and the GT destination. It is a weighted sum of:
   - A smoothed L1 loss between the predicted (x, y) positions and the GT positions.
   - A cosine distance between the predicted yaw θ and the GT yaw $\hat{\theta}$, i.e. $0.5 \cdot (1 - \cos(\theta - \hat{\theta}))$.
   - A smoothed L1 loss between the predicted velocity and the GT velocity.
2. The KL divergence between the posterior and the prior personality, which trains the prior to match the posterior and regularize the posterior at the same time. We use free nats [118] to clip the KL divergence, i.e. if $KL(\mathbf{z}_{\text{post}}, \mathbf{z}_{\text{prior}})$ is smaller than the free nats, then the KL loss is not applied. We use a free nats of 0.01.
3. The cross entropy loss for destination classification. Since the GT destination is a single class, this loss boils down to a maximum likelihood loss, i.e. the destination distribution is trained to maximize the log-likelihood of the polyline index of the GT destination.

4.F Inference Details

We use the GT destination and the most likely posterior personality for the a posteriori simulation, hence the simulation is single modal in this case. For a priori simulation, i.e. motion prediction, we generate multiple modes by randomly sampling the destination distribution and the prior personality of each agent. For WOMD we generate $K = 6$ predictions. The first mode K_0 is deterministic, which is generated using the most likely destination and prior personality. We use this mode to inspect the most likely mode of the joint future prediction. The score of each prediction, which is required by the WOMD leaderboard, is the joint probability of the destination and the personality. We normalize the score using softmax with temperature. The scores are computed with respect to agents, not the joint future of all agents. For motion prediction where the future traffic light states are not available, we use the last observed (i.e. from the current step $t = 0$) light states for all prediction steps.

test	soft mAP ↑	mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓	overlap rate ↓
DenseTNT	N/A	0.165	1.142	2.490	0.535	0.231
SceneTransformer (J)	N/A	0.119	0.977	2.189	0.494	0.207
Air2	N/A	0.096	1.317	2.714	0.623	0.247
HeatIRm4	N/A	0.084	1.420	3.260	0.722	0.284
Waymo LSTM	N/A	0.052	1.906	5.028	0.775	0.341
TrafficBots (a priori)	0.113	0.111	1.669	4.514	0.681	0.220
valid TrafficBots	soft mAP ↑	mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓	overlap rate ↓
a priori (*K=6*)	0.102	0.100	1.670	4.514	0.677	0.221
GT sdc future (*what-if*)	0.110	0.108	1.577	4.317	0.651	0.215
GT traffic light (*v2x*)	0.102	0.100	1.663	4.485	0.675	0.221
GT destination (*v2v*)	0.106	0.103	1.640	4.440	0.668	0.223
a posteriori (*K=1*)	0.188	0.188	1.085	2.313	0.602	0.165

Table 4.4: Performance on the Waymo (joint) interactive prediction leaderboard.

4.G More Experimental Results

In Table 4.4 we compare TrafficBots with other open-loop motion prediction methods on the Waymo (joint) interactive prediction leaderboard, where the joint future of exactly two agents shall be predicted and the metrics are evaluated at the scene-level, i.e. for both agents at the same time. For a more detailed description on the task and the metrics, please refer to the publication [134] or the homepage of the WOMD. Since our method is essentially solving the joint future prediction, TrafficBots significantly outperforms the baselines on this task. As shown in Table 4.4, we achieve overall better performance than the *LSTM* baseline [134]. TrafficBots also perform better than *HeatIRm4* [216], the winner of the 2021 WOMD challenge, and *Air2* [217], the honorable mention of the 2021 WOMD challenge, in terms of the mAP and the overlap rate, which are the most relevant metrics used for the ranking. Our performance is comparable to *SceneTransformer (J)*, the joint version of SceneTransformer [162]. Compared to *DenseTNT* [142], we achieve a lower overlap rate. As discussed in the main paper, our method suffers from larger minADE/FDE and the performance can be improved given additional GT information. These trends are also observed in Table 4.4. Although the (joint) interactive prediction is a more favorable task for our method, we do not include Table 4.4 in the main paper because this leaderboard is partially deprecated and hence less active, and the predictions are restricted to two agents which significantly limits its application in the real world.

5

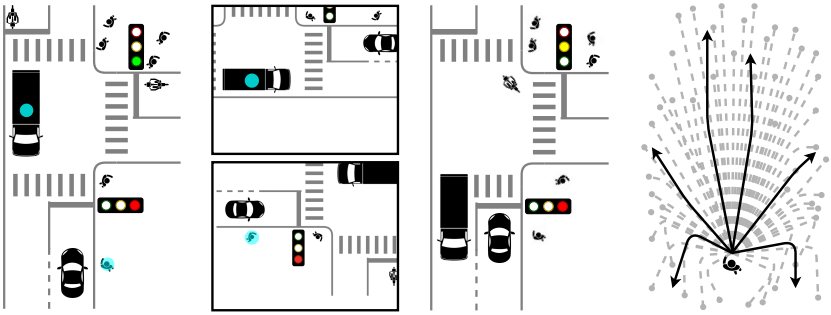
Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding

The real-world deployment of an autonomous driving system requires its components to run on-board and in real-time, including the motion prediction module that predicts the future trajectories of surrounding traffic participants. Existing agent-centric methods have demonstrated outstanding performance on public benchmarks. However, they suffer from high computational overhead and poor scalability as the number of agents to be predicted increases. To address this problem, we introduce the K-nearest neighbor attention with relative pose encoding (KNARPE), a novel attention mechanism allowing the pairwise-relative representation to be used by Transformers. Then, based on KNARPE we present the Heterogeneous Polyline Transformer with Relative pose encoding (HPTR), a hierarchical framework enabling asynchronous token update during the online inference. By sharing contexts among agents and reusing the unchanged contexts, our approach is as efficient as scene-centric methods, while performing on par with state-of-the-art agent-centric methods. Experiments on Waymo and Argoverse-2 datasets show that HPTR achieves superior performance among end-to-end methods that do not apply expensive post-processing or model ensembling. The code is available at `https://github.com/zhejz/HPTR`.

5.1 Introduction

Motion prediction is an important component of modular autonomous driving stack [2]. As the downstream module of perception [182, 218, 219] and the upstream module of planning [100, 220, 221], the task of motion prediction [134, 135] is to predict the multi-modal future trajectories of other agents next to the self-driving vehicle (SDV) based on the heterogeneous observations, including for example high-definition (HD) maps, traffic lights and other vehicles, pedestrians and cyclists. This is an essential task because

This chapter was published as a conference article: Zhejun Zhang, Alexander Liniger, Christos Sakaridis, Fisher Yu, Luc Van Gool, "Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding", Advances in Neural Information Processing Systems (NeurIPS), 2023, doi: 10.3929/ethz-b-000643424



(a) Dense traffic scenario (b) Agent-centric ROIs (c) Online inference (d) Trajectory aggregation

Figure 5.1: To efficiently predict the multi-modal future of numerous agents in dense traffic (5.1a), HPTR minimizes the computational overhead by: (5.1b) Sharing contexts among target agents. (5.1c) Reusing static contexts during online inference. (5.1d) Avoiding expensive post-processing and ensembling.

without accurate and real-time prediction results [222], the planning module cannot safely and comfortably navigate the SDV through highly interactive driving environments.

To achieve top performance on public motion prediction leaderboards [137, 138], state-of-the-art (SOTA) methods [164, 165, 166, 167] leverage agent-centric vectorized representations and Transformer-based network architectures. However, the good performance of these approaches comes at a cost of high computational overhead as illustrated in Figure 5.1. The most well-known problem of agent-centric approaches is the poor scalability as the number of target agents grows in urban driving environments with dense traffic; for example the busy intersection in Figure 5.1a. Although the agent-centric regions of interest (ROI) in Figure 5.1b are largely overlapping, the same context is transformed to the coordinate system of each target agent and independently saved and processed. This causes a huge waste of computational resources, which is often neglected by prior works because their experiments focus on offline inference that queries the prediction only once given a scenario from the dataset. In contrast to offline inference, the motion prediction module of a real-world SDV is continuously queried with streaming inputs during online inference. For example, in Figure 5.1c, the prediction module is queried again shortly after Figure 5.1a. Although most contexts remain unchanged after this short period of time, existing methods would start inference from scratch without reusing the encoded features of static contexts. Moreover, prior works often predict a massive number of redundant trajectories and ensemble the outputs of numerous models, such that the final output can be adjusted in favor of prediction diversity during the post-processing, as illustrated in Figure 5.1d. Although these techniques significantly boost the performance on public benchmarks, they should be sparingly used on a real-world SDV where the computational resources are scarce and the raw predictions, rather than the heuristically aggregated ones, are preferred by the downstream planning module.

To address these problems, our first step is to represent everything as heterogeneous polylines [158]. Then we adopt the pairwise-relative representation [163, 169] and separate

the local attribute from the global pose of each polyline. The local attribute specifies what the polyline actually is and it will be shared or reused if possible. The global pose specifies where the polyline is and it will be used to derive the pairwise-relative pose before being processed by the neural networks. Considering polylines as tokens, our second step is to introduce the **K**-nearest **N**eighbor **A**ttention with **R**elative **P**ose **E**ncoding (KNARPE) module, which allows Transformers [159] to aggregate the local contexts for each token via the local attributes and relative poses. While KNARPE enables context sharing among agents, we further propose the **H**eterogeneous **P**olyline **T**ransformer with **R**elative pose encoding (HPTR), which emphasizes the heterogeneous nature of polylines in motion prediction tasks. Based on KNARPE, HPTR uses hierarchical Transformer encoders and decoders to separate the intra-class and inter-class attention, such that tokens can be updated asynchronously during online inference. More specifically, for static polylines, such as HD maps, their features will be reused, whereas for dynamic polylines, such as traffic lights and agents, their features will be updated on demand.

Our contributions are summarized as follows: (1) We introduce KNARPE, a novel attention mechanism that enables the pairwise-relative representation to be used by Transformer-based architectures. (2) Based on KNARPE we propose HPTR, a hierarchical framework that minimizes the computational overhead via context sharing among agents and asynchronous token update during online inference. (3) Compared to SOTA agent-centric methods, we achieve similar performance while reducing the memory consumption and the inference latency by 80%. By caching the static map features during online inference, HPTR can generate predictions for 64 agents in real time at 40 frames per second. Experiments on Waymo and Argoverse-2 datasets show that our approach compares favorably against other end-to-end methods which do not apply expensive post-processing and ensembling.

5.2 Related work

Motion prediction is a popular research topic because it is an essential component of modular autonomous driving stacks [2]. The task of motion prediction has been formulated in different ways. In this paper, we focus on the most popular one: the marginal motion prediction [145, 146, 148, 149, 223] where the multi-modal future trajectories are predicted individually for each target agent [137, 138, 139]. In contrast to the marginal formulation, joint motion prediction [209, 224, 225, 226] requires the multi-modal futures to be simultaneously generated for all target agents from the same scenario [227, 228, 229]. Beyond the open-loop motion prediction tasks, behavior simulation [18, 112, 114] is formulated in a closed-loop fashion where the future agent trajectories are simulated by rolling out a learned policy [100, 140]. Numerous works have also investigated the possibility to combine motion prediction with other modules, such as perception and planning [41, 110, 213, 230], or to learn an end-to-end driving policy [26, 35, 131] mapping sensor measurements directly to the actions or motion plans of the SDV.

Vectorized representation proposed by VectorNet [158] is widely used in recent works because of its promising performance [165, 166, 167]. Depending on how the coordinate system is selected, the vectorized representation falls into three categories: agent-centric, scene-centric and pairwise-relative. The most popular one is the agent-centric representa-

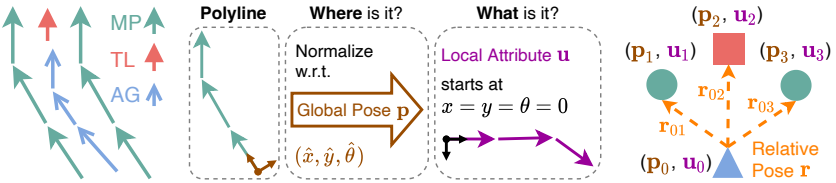
tion [164, 165, 166, 167] that transforms all inputs to the local coordinates of each target agent. Despite its good performance, this approach suffers from poor scalability as the number of target agents grows. To reduce the computational overhead, scene-centric representation [162] shares the contexts among all target agents by transforming all inputs to a global coordinate system, which is not tied to any specific agent but to the whole scene. However, its performance is poor due to the lack of rotation and translation invariance. In this paper we use the pairwise-relative representation, which is less often discussed in prior works because it has not demonstrated any clear advantage. To the best of our knowledge, there are three works using the pairwise-relative representation: HDGT [169], GoRela [163] and HiVT [168]. HDGT and GoRela are based on Graph Neural Networks (GNNs), and their implementation requires message passing and GNN libraries. However, these libraries are often less efficiently implemented on Graphics Processing Units (GPUs) when compared to the basic matrix operations that Transformers rely on. HiVT augments the agent-centric encoders with a pairwise-relative interaction decoder in order to realize multi-agent prediction. In contrast to HiVT which uses agent-centric vectors and the standard Transformer, we formulate all inputs as pairwise-relative polylines and introduce the KNARPE attention mechanism. As a result, our HPTR demonstrates clear advantages in terms of accuracy and efficiency. Replacing the vanilla attention with our KNARPE, we can borrow ideas from other Transformer-based methods and adapt them to the pairwise-relative representation. For example, the hierarchical architecture of our HPTR is inspired by Wayformer [164]. More sophisticated architectures and techniques, such as goal-based decoding [141, 142], can also be incorporated into our framework to further boost the performance.

Rasterized representation was widely used in early works on motion prediction [147, 153, 154, 155]. These methods use convolutional neural networks (CNNs) to process the rasterized image of agent-centric ROI. To improve the inference efficiency, some works [146, 231, 232, 233] pre-compute the static map features and use rotated ROI alignment [234] to retrieve the local contexts around the target agent. In this paper, our KNARPE realizes this operation for pure Transformer-based architectures.

Transformers with attention mechanism [159] have achieved great success in natural language processing [160, 235] and computer vision tasks [161, 236, 237]. Inspired by the vision Transformers, we treat polylines as high-dimensional pixels; the local attribute corresponds to color channels, whereas the global pose corresponds to the pixel-wise location. We further extend the relative position encoding [238, 239] to higher dimensions and rename it to relative pose encoding (RPE) in this paper. Although the benefit of RPE is still controversial for other tasks [240, 241], its value for motion prediction is demonstrated in this paper. Instead of using RPE, previous motion prediction Transformers use everything as input, which is equivalent to concatenating the pixel-wise location to the RGB channels. This is a very uncommon practice from the perspective of vision Transformers.

5.3 Method

In Sec. 5.3.1 we introduce the pairwise-relative polyline representation which enjoys the advantages of both the agent-centric and the scene-centric representation. Then in



(a) Polylines. (b) Local attribute represented in the global pose's frame. (c) Obtain relative poses.

Figure 5.2: Pairwise-relative polyline representation. MP: Map. TL: Traffic lights. AG: Agents.

Sec. 5.3.2 we present the **K**-nearest **N**eighbor **A**ttention with **R**elative **P**ose **E**ncoding (KNARPE) which enables the pairwise-relative polyline representation to be used by Transformers. Based on KNARPE, we propose the **H**eterogeneous **P**olyline **T**ransformer with **R**elative pose encoding (HPTR) in Sec. 5.3.3. HPTR uses a hierarchical architecture to prevent redundant computations and to realize the asynchronous update of heterogeneous tokens. Finally in Sec. 5.3.4 we discuss our output representation and training strategies.

5.3.1 Pairwise-Relative Polyline Representation

Based on prior works [163, 169], we formulate the pairwise-relative polyline representation as the third type of input representation for motion prediction. As shown in Figure 5.2a, all data relevant to motion prediction can be represented as an ordered list of consecutive vectors, i.e. polylines. By transforming the polyline to the coordinate frame of its global pose as done in Figure 5.2b, a polyline is fully described by a pair of global pose and local attribute. The global pose specifies the location and heading of the polyline in the global coordinate system, whereas the local attribute describes the polyline in its local coordinates, for example it is a yellow solid lane, 8 meters long, slightly curved to the right. In practice, it does not matter where the lane is on the earth; we only care about where the lane is relative to us. Hence, we obtain the relative poses from the global poses as shown in Figure 5.2c, before feeding them to the prediction network. Specifically, the global pose and local attribute of polyline i are denoted as $(\mathbf{p}_i, \mathbf{u}_i)$. The global pose $\mathbf{p}_i$ has 3 degrees of freedom (x, y, θ), i.e. the 2D position and the heading yaw angle. The local attribute $\mathbf{u}_i$ is derived from $\mathbf{c}_i$, the intrinsic characteristics of the polyline, and $\mathbf{l}_i$, the polyline vectors represented in the local coordinate.

The task of marginal motion prediction is to predict the future 2D positions individually for each target agent based on the static HD map (MP), the history trajectories of all agents (AG) and the traffic lights (TL). For each scenario to be predicted, we consider a maximum number of N_{MP} map polylines, N_{TL} traffic lights and N_{AG} agents. We define $t = 0$ to be the current step, $\{T_{\text{h}} - 1, \dots, 0\}$ to be the observed history steps and $\{1, \dots, T_{\text{f}}\}$ to be the predicted future steps. The static HD map are spatial polylines $(\mathbf{p}_i^{\text{MP}}, \mathbf{l}_i^{\text{MP}}, \mathbf{c}_i^{\text{MP}}), i \in \{1, \dots, N_{\text{MP}}\}$ where $\mathbf{p}_i^{\text{MP}} \in \mathbb{R}^3$ is its starting vector, $\mathbf{l}_i^{\text{MP}} \in \mathbb{R}^{N_{\text{node}} \times 4}$ are the 2D positions and directions of the N_{node} segments normalized against $\mathbf{p}_i^{\text{MP}}$, and $\mathbf{c}_i^{\text{MP}} \in \mathbb{R}^{C_{\text{MP}}}$ is the one-hot encoding of C_{MP} different lane types. Polygonal elements, such as crosswalks, are converted to a group of parallel polylines across the polygon. Similar

to the map, history trajectories of agents are represented as spatial-temporal polylines $(\mathbf{p}_i^{\mathrm{AG}}, \mathbf{l}_i^{\mathrm{AG}}, \mathbf{c}_i^{\mathrm{AG}}), i \in \{1, \dots, N_{\mathrm{AG}}\}$ where $\mathbf{p}_i^{\mathrm{AG}} \in \mathbb{R}^3$ is the last observed agent pose, $\mathbf{l}_i^{\mathrm{AG}} \in \mathbb{R}^{T_{\mathrm{h}} \times (6+3)}$ contains the history 2D positions, directions and velocities normalized against $\mathbf{p}_i^{\mathrm{AG}}$ as well as the 1D speed, yaw rate and acceleration which do not need to be normalized, and $\mathbf{c}_i^{\mathrm{AG}} \in \mathbb{R}^6$ is the 3D agent size and the one-hot encoding of 3 agent types (vehicle, pedestrian and cyclist). Following VectorNet [158], we use PointNet [242] with masked max-pooling to aggregate $(\mathbf{l}_i^{\mathrm{MP}}, \mathbf{c}_i^{\mathrm{MP}})$ into $\mathbf{u}_i^{\mathrm{MP}} \in \mathbb{R}^D$ and $(\mathbf{l}_i^{\mathrm{AG}}, \mathbf{c}_i^{\mathrm{AG}})$ into $\mathbf{u}_i^{\mathrm{AG}} \in \mathbb{R}^D$, where D is the hidden dimension. Since current datasets contain only the detection results rather than the tracking results of traffic lights, we consider only the traffic lights observed at $t = 0$ and represent them as singular polylines $(\mathbf{p}_i^{\mathrm{TL}}, \mathbf{c}_i^{\mathrm{TL}}), i \in \{1, \dots, N_{\mathrm{TL}}\}$ where $\mathbf{p}_i^{\mathrm{TL}} \in \mathbb{R}^3$ is the pose of the stop point and $\mathbf{c}_i^{\mathrm{TL}} \in \mathbb{R}^{C_{\mathrm{TL}}}$ is the one-hot encoding of C_{TL} different states of traffic lights. We use a multi-layer perceptron (MLP) to encode $\mathbf{c}_i^{\mathrm{TL}}$ into $\mathbf{u}_i^{\mathrm{TL}} \in \mathbb{R}^D$.

Because the local attributes are normalized and the global poses are used to compute the relative poses before being consumed by the network, the pairwise-relative representation preserves the viewpoint invariance of the agent-centric representation. Additionally, since the local attributes have higher dimensions compared to the 3-dimensional global poses, sharing the local attributes enables the pairwise-relative representation to maintain the good scalability of the scene-centric representation. However, so far this representation has only be exploited by GNNs, which are not comparable to Transformers in terms of accuracy and efficiency on the motion prediction benchmarks [137, 138].

5.3.2 KNARPE: K-Nearest Neighbors Attention with Relative Pose Encoding

After encoding the local attributes via the polyline-level encoders, the scenario is now described as $(\mathbf{p}_i, \mathbf{u}_i), i \in \{1, \dots, N\}$ where $N = N_{\mathrm{MP}} + N_{\mathrm{AG}} + N_{\mathrm{TL}}$ is the total number of polylines. However, this representation cannot be handled by standard self-attention because on the one hand modeling the all-to-all attention is prohibitively expensive [164] due to the large N, and on the other hand the performance deteriorates when the input is scene-centric [162]. To address both problems, we introduce the **K**-nearest **N**eighbors **A**ttention with **R**elative **P**ose **E**ncoding (KNARPE). Similar to MTR [165], KNARPE limits the attention to the K-nearest neighbors of each token. Specifically, $\kappa_i^K(d_{i1}, \dots, d_{iN}) \subseteq \{1, \dots, N\}$ is a set that contains the indices of K tokens closest to token i. For simplicity, we use the L2 distance to measure the distance d_{ij} between the global poses of token i and j. Instead of directly processing global poses, KNARPE uses the relative pose encoding (RPE), i.e. the positional encoding for pairwise-relative poses. Denoting $\mathbf{r}_{ij} = (x_{ij}, y_{ij}, \theta_{ij})$ to be the global pose of token j represented in the coordinate of token i, i.e. $\mathbf{p}_j$ transformed to the coordinate of $\mathbf{p}_i$, the RPE of $\mathbf{r}_{ij}$ is computed using sinusoidal positional encoding (PE) and angular encoding (AE) [112],

$$
\begin{aligned}
&\mathrm{RPE}(\mathbf{r}_{ij}) = \mathrm{concat}(\mathrm{PE}(x_{ij}), \mathrm{PE}(y_{ij}), \mathrm{AE}(\theta_{ij})), \\
&\mathrm{PE}_{2i}(x) = \sin(x \cdot \omega^{\frac{2i}{D}}),\ \mathrm{PE}_{2i+1}(x) = \cos(x \cdot \omega^{\frac{2i}{D}}), \\
&\mathrm{AE}_{2i}(\theta) = \sin\left(\theta \cdot (i+1)\right),\ \mathrm{AE}_{2i+1}(\theta) = \cos\left(\theta \cdot (i+1)\right),\ i \in \{0, \dots, D/2-1\},
\end{aligned}
$$

where ω is the base frequency. Following [239], the RPE is projected and added to the keys and values to obtain $\mathbf{z}_i$, the output of letting token i attend to its K neighbors κ_i^K,

$$\mathbf{z}_i = \textsc{Knarpe}\left(\mathbf{u}_i, \mathbf{u}_j, \mathbf{r}_{ij} \mid j \in \kappa_i^K\right) = \sum_{j \in \kappa_i^K} \alpha_{ij} \left(\mathbf{u}_j \mathbf{W}^v + \mathbf{b}^v + \text{RPE}(\mathbf{r}_{ij}) \hat{\mathbf{W}}^v + \hat{\mathbf{b}}^v\right),$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \kappa_i^K} \exp(e_{ik})}, \quad e_{ij} = \frac{(\mathbf{u}_i \mathbf{W}^q + \mathbf{b}^q)(\mathbf{u}_j \mathbf{W}^k + \mathbf{b}^k + \text{RPE}(\mathbf{r}_{ij}) \hat{\mathbf{W}}^k + \hat{\mathbf{b}}^k)}{\sqrt{D}},$$

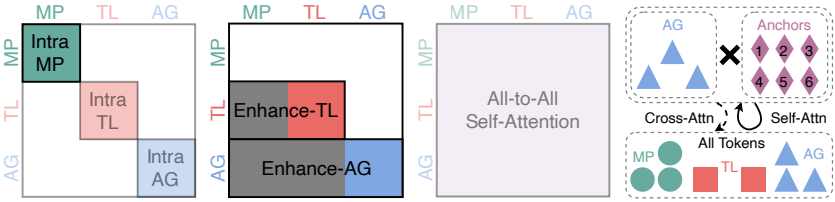
where α_{ij} are the attention weights, e_{ij} are the logits, $W^{\{q,k,v\}}, b^{\{q,k,v\}}$ are the learnable projection matrices and biases for query, key and value, and $\hat{W}^{\{k,v\}}, \hat{b}^{\{k,v\}}$ are the learnable projection matrices and biases for RPE. We do not apply RPE to query because doing this does not boost the performance in our experiments.

Efficient implementation of KNARPE can be achieved using basic matrix operations such as matrix indexing, summation and element-wise multiplication. See the appendix for more details. KNARPE allows the pairwise-relative representation to be used by pure Transformer-based architectures. Self-attention with KNARPE aggregates the local context for each token in the same way as CNN aggregates the context around each pixel via convolutional kernels. The pixel corresponds to the token, whereas the kernel size of a CNN corresponds to the number of neighbors of KNARPE. Cross-attention with KNARPE enables rotated ROI alignment of pre-computed features, e.g. the static map features. Previously, this was only possible for CNN-based [146, 232] and GNN-based [163, 169] methods.

5.3.3 HPTR: Heterogeneous Polyline Transformer with Relative Pose Encoding

By replacing the standard multi-head attention [159] with our KNARPE, we construct Transformer encoders and decoders which can model the interactions between heterogeneous polylines via relative poses and local attributes. To address the marginal motion prediction task involving HD map, traffic lights and agents, we propose the **H**eterogeneous **P**olyline **T**ransformer with **R**elative pose encoding (HPTR) as shown in Figure 5.3. Since the temporal dimension is eliminated by the polyline-level encoder [158], HPTR models only the spatial relationship between tokens from different classes.

Firstly in Figure 5.3a, the intra-class Transformer encoders build a block diagonal attention matrix that models the interactions within each class of tokens. Then in Figure 5.3b, the inter-class Transformer decoders enhance the traffic lights tokens by making them attend to the map tokens, whereas the agent tokens are enhanced by attending to both the traffic lights and map tokens. The intuition behind this is the hierarchical nature of traffic; first there is only the map, then the traffic lights are added and finally the agents join. Intuitively, the map influences the interpretation of traffic lights but not vice versa, whereas map and traffic lights together influence the behavior of agents but not vice versa. After the inter-class Transformer decoders, the all-to-all Transformer encoder in Figure 5.3c builds a full attention matrix allowing tokens to attend to each other irrespective of their class. Finally, each agent token is concatenated with N_{AC} learnable anchors, which are shared among each type of agent. Since the task is marginal motion prediction, we batch over agents and anchors, i.e. now the number of anchor



(a) Intra-class TF encoder (b) Inter-class TF decoder (c) All-to-all TF encoder (d) AC-to-all TF decoder

Figure 5.3: The hierarchical architecture of HPTR. Transformers are applied in sequential order from left to right, top to down. Some attentions are redundant and can be skipped for better efficiency. We propose the lower triangular attention matrix, which excludes the intra-TL, intra-AG and all-to-all self-attentions (the transparent parts). By removing the redundant attentions, this specific lower triangular architecture enables asynchronous token update during online inference. TF: Transformer. Attn: Attention. MP: Map. TL: Traffic lights. AG: Agents.

tokens is $N_{\text{AG}} \cdot N_{\text{AC}}$. Then the anchor-to-all Transformer decoder in Figure 5.3d lets each of these anchor tokens attend to all tokens so as to aggregate more contextual information. The final output is denoted as $\hat{\mathbf{z}}_i^{\text{AG}}, i \in \{1, \dots, N_{\text{AG}} \cdot N_{\text{AC}}\}$, and it is used to generate the multi-modal future trajectories. We set the number of neighbors to K for the intra-class and all-to-all Transformers. This number K is multiplied by γ_{TL}, γ_{AG} and γ_{AC} respectively for the enhance-TL, enhance-AG and AC-to-all Transformers, such that these Transformers can have a larger receptive field.

HPTR organizes the Transformers in a hierarchical way such that some of the intermediate results can be cached and reused during the online inference; for example the outputs of the intra-MP Transformer, i.e. the static map features. This allows tokens from different classes to be updated asynchronously, which significantly reduces the online inference latency. We can further improve the efficiency without sacrificing the performance by trimming the redundant attentions. The attention matrices shown in Figure 5.3a, 5.3b and 5.3c are overlapping, which means some relationships are repeatedly modeled, such as the agent-to-agent self-attention. We propose to remove the intra-TL, intra-AG and all-to-all Transformer, while keeping the intra-MP, enhance-TL and enhance-AG Transformer. These three Transformers together build a lower triangular attention matrix which is necessary and sufficient to model the relationship between map, traffic lights and agents. Our approach can be seen as an extension to Wayformer [164] which does not introduce the inter-class Transformer decoders. We can trim Figure 5.3 differently and cast HPTR into the Wayformer. Specifically, Wayformer with late fusion corresponds to HPTR with diagonal attention matrix (only intra-class TF), Wayformer with early fusion corresponds to HPTR with full attention matrix (only all-to-all TF) and Wayformer with hierarchical fusion corresponds to HPTR with the diagonal followed by the full attention matrix.

5.3.4 Output Representation and Training Strategies

We follow the common practice [164, 165, 166] to represent the outputs as a mixture of Gaussians and train with hard assignment. The multi-modal future trajectories for each agent are generated by decoding $\hat{\mathbf{z}}_i^{\text{AG}}, i \in \{1, \dots, N_{\text{AG}} \cdot N_{\text{AC}}\}$ via two MLPs; the

confidence head and the trajectory head. The confidence head predicts a scalar confidence of each trajectory. Besides the Gaussian parameters of 2D positions $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$ at each future time step, our trajectory head predicts also the yaw angles, speeds and 2D velocities. We use the cross entropy loss for confidences, negative log-likelihood loss for 2D positions, negative cosine loss for yaw angles and Huber loss for speeds and 2D velocities. The final training loss is the unweighted sum of all losses. Following the hard-assignment strategy, for each agent we optimize only the predicted trajectory that is closest to the ground truth in terms of 2D average displacement error. Please refer to the appendix for more details.

5.4 Experiments

5.4.1 Experimental Setup

Benchmarks. We benchmark our method on the two most popular datasets: the Waymo Open Motion Dataset (WOMD) [134] and the Argoverse-2 motion forecasting dataset (AV2) [135]. Both datasets have an online leaderboard for marginal motion prediction. However, their task descriptions are slightly different. For each scenario, WOMD evaluates the predictions of up to 8 agents, whereas AV2 evaluates only one agent. Both datasets require exactly 6 futures to be predicted for each target agent. The sampling time is 0.1 seconds for both datasets. The history length is 11 steps for WOMD and 50 steps for AV2, whereas the future length is 80 steps for WOMD and 60 steps for AV2. We use the official evaluation tool of the leaderboards to compute the metrics. For the WOMD leaderboard [137], the ranking metric is soft mAP; for the AV2 leaderboard [138], it is brier-minFDE. Please refer to the leaderboard homepages for more details about the dataset and evaluation metrics.

Implementation details. We use Transformer with pre-layer normalization [215] for HPTR. For each episode, we consider $N_{\text{MP}} = 1024$ map polylines, $N_{\text{TL}} = 40$ traffic light stop points and $N_{\text{AG}} = 64$ agents. Each polyline contains up to $N_{\text{node}} = 20$ one-meter-long segments. The base number of neighbors considered by KNARPE is $K = 36$. This number is multiplied by $\gamma_{\text{TL}} = 2$, $\gamma_{\text{AG}} = 4$ and $\gamma_{\text{AC}} = 10$ respectively for the enhance-TL, enhance-AG and AC-to-all Transformer. We set $N_{\text{AC}} = 6$ to predict exactly 6 futures as specified by the leaderboard. We do not apply ensembling or expensive post-processing such as trajectory aggregation. Our post-processing manipulates only the confidences. To improve the soft mAP, we use the non-maximum suppression of MPA [243] for the WOMD leaderboard. To improve the brier-minFDE, we set the softmax temperature to 0.5 for the AV2 leaderboard. More details are provided in the appendix.

Training details. Thanks to the viewpoint invariance of the pairwise-relative representation, no data augmentation or input permutation is needed for the training of HPTR. We use AdamW optimizer with an initial learning rate of 1e-4 and decaying by 0.5 every 25 epochs. We train with a total batch size of 12 episodes on 4 RTX 2080Ti GPUs. For WOMD, we randomly sample 25% from all training episodes at each epoch; for AV2 we use 50%. Our final models are trained for 120 epochs for WOMD and 150 epochs for AV2. The complete training takes 10 days. For WOMD, the SDV agents, agents of interest

WOMD *test*	repr.	*soft mAP* ↑	mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓
*† MTR-Adv-ens [165]	AC	0.4594	0.4492	0.5640	1.1344	0.1160
*† Wayformer [164]	AC	0.4335	0.4190	0.5454	1.1280	0.1228
* MTR [165]	AC	0.4216	0.4129	0.6050	1.2207	0.1351
*† MultiPath++ [166]	AC	N/A	0.4092	0.5557	1.1577	0.1340
HPTR (Ours)	PR	0.3968	0.3904	0.5565	1.1393	0.1434
MPA [243] (MultiPath++)	AC	0.3930	0.3866	0.5913	1.2507	0.1603
HDGT [169]	PR	0.3709	0.3577	0.7676	1.1077	0.1325
Gnet [244]	AC	0.3396	0.3259	0.6207	1.2391	0.1718
SceneTransformer [162]	SC	N/A	0.2788	0.6117	1.2116	0.1564
WOMD *valid*	repr.	*soft mAP* ↑	mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓
HPTR (Ours)	PR	0.4222	0.4150	0.5378	1.0923	0.1326
MTR-e2e	AC	N/A	0.3245	0.5160	1.0404	0.1234

AV2 *test*	repr.	*brier-minFDE*$_6$ ↓	minFDE$_6$ ↓	minFDE$_1$ ↓	minADE$_6$ ↓	miss rate$_6$ ↓
* ProphNet [167]	AC	1.88	1.33	4.74	0.68	0.18
† Gnet [244]	AC	1.90	1.34	4.40	0.69	0.18
† TENET [245]	AC	1.90	1.38	4.69	0.70	0.19
* MTR [165]	AC	1.98	1.44	4.39	0.73	0.15
GoRela [163]	PR	2.01	1.48	4.62	0.76	0.22
HPTR (Ours)	PR	2.03	1.43	4.61	0.73	0.19
THOMAS [224]	AC	2.16	1.51	4.71	0.88	0.20
HDGT [169]	PR	2.24	1.60	5.37	0.84	0.21

Table 5.1: Results on the marginal motion prediction leaderboards of WOMD and AV2. Both tables are sorted according to the ranking metric such that the best performing method is on the top. The ranking metric is *soft mAP* for WOMD, and *brier-minFDE*$_6$ for AV2. † denotes ensemble. * denotes predicting more futures than required. SC: scene-centric. AC: agent-centric. PR: pairwise-relative.

and agents to be predicted are used for optimization. For AV2, the SDV agents, scored agents and focal agents are used for optimization. To address the imbalance between agent types, for WOMD we additionally optimize for pedestrians and cyclists which are tracked for at least 4 seconds.

5.4.2 Benchmark Results

In Table 5.1 we benchmark our approach on the public leaderboards of WOMD and AV2. From the leaderboard we observe that using ensemble and predicting redundant trajectories can increase the prediction diversity and hence improve the soft mAP, brier-minFDE and miss rate. For example, MTR-Adv-ens aggregates the outputs of 7 ablation models, each predicting 64 futures. Besides the exaggerated large number of redundant predictions, it is also non-trivial to aggregate them into 6 predictions. Some works use K-means clustering while the others use non-maximum suppression; both involve heuristic parameter fine-tuning. Since our framework is dedicated to real-world autonomous driving, applying these computationally expensive techniques is contradictory to our motivation. For a fair comparison, we focus on end-to-end methods which do not apply these techniques. On the WOMD dataset, we achieve SOTA performance among the end-to-end methods, including MTR-e2e, the end-to-end version of MTR, and MPA, the end-to-end version of MultiPath++. Specifically, HPTR outperforms the pairwise-relative HDGT and the scene-centric SceneTransformer by a large margin in all metrics. We also show competitive performance on the AV2 leaderboard. We achieve better performance in all metrics except the brier-minFDE compared to GoRela, which uses GNNs to tackle the pairwise-relative representation. Since there exists a performance gap while adapting from one dataset to another and we choose WOMD to be our main benchmark, our performance on the AV2 leaderboard could be further improved by fine-tuning the hyper-parameters for the AV2 dataset.

5.4.3 Ablation Study

In Table 5.2 we ablate different input representations and hierarchical architectures. The scene-centric baseline, HPTR SC, uses the architecture of our HPTR and the input representation of SceneTransformer [162]. The agent-centric baseline, WF baseline, is our reimplementation of the Wayformer [164] with multi-axis attention and early fusion. Both baselines achieve their expected performances compared to their original implementations. The large performance gap between HPTR SC and other models confirms the disadvantage of scene-centric representation. The agent-centric baseline requires more training iterations. After convergence, its performance is on par with our HPTR. To ablate the hierarchical architecture, we implement three variations of HPTR; each corresponds to a fusion strategy investigated by Wayformer. The full attention corresponds to the early fusion, diagonal corresponds to late fusion, and HPTR with diagonal followed by full attention corresponds to Wayformer with hierarchical fusion. While the early fusion performs the best for Wayformer, for HPTR the lower triangular attention we proposed outperforms both the early and the late fusion by a significant margin. The performance

WOMD *valid*	input repr.	intra MP	intra TL/AG	enhance TL/AG	all 2all	minFDE ↓	*soft mAP* ↑
HPTR (Ours)	PR	✓	×	✓	×	1.145 ± 0.016	0.399 ± 0.010
WF baseline (100-epoch)	AC	×	×	×	✓	1.161 ± 0.006	0.397 ± 0.007
HPTR diag+full (WF hier)	PR	✓	✓	×	✓	1.156 ± 0.014	0.391 ± 0.002
HPTR diag (WF late)	PR	✓	✓	×	×	1.169 ± 0.013	0.387 ± 0.012
HPTR full (WF early)	PR	×	×	×	✓	1.158 ± 0.093	0.386 ± 0.041
WF baseline	AC	×	×	×	✓	1.212 ± 0.019	0.378 ± 0.014
HPTR SC	SC	✓	×	✓	×	1.687 ± 0.046	0.246 ± 0.005

Table 5.2: Ablation on the valid split of WOMD. The table is sorted according to the *soft mAP* such that the best-performing method is on the top. Performances are reported as the mean plus-minus 3 standard deviations over 3 training seeds. Models are trained for 60 epochs if not otherwise mentioned. WF: Wayformer. SC: scene-centric, AC: agent-centric, PR: pairwise-relative.

of hierarchical fusion is slightly worse than ours, but its inference latency is significantly longer because of the redundancy in its attention matrices.

5.4.4 Efficiency Analysis and Qualitative Results

In Figure 5.4 we compare the computational efficiency of the ablation models presented in Table 5.2. As discussed in prior works [164, 165], one of the major drawbacks of agent-centric approaches is the poor scalability, which is reflected by the large slope of the memory and latency curves of our WF baseline. On the RTX 2080Ti, it can handle only up to 48 agents while the inference latency is 140ms. On the contrary, the scene-centric baseline is extremely efficient, but it suffers from poor accuracy. Our HPTR takes the best of both approaches. In terms of efficiency, our HPTR is comparable to the scene-centric approaches, while our performance is on par with the agent-centric approaches. Compared to the scene-centric baseline, the GPU memory consumption and the inference latency of HPTR are slightly higher because for each new agent, we have to compute its relative pose to all existing tokens. Compared to the hierarchical architectures introduced by Wayformer, HPTR with our lower triangular attention achieves the best trade-off between accuracy and latency. By reusing the static map features during the online inference, our HPTR predicts the multi-modal futures for 64 agents in 37ms without the use of inference libraries. Narrowing the receptive field, for example by reducing λ_{AC} from 10 to 8, can improve the inference speed but the accuracy might be affected in some cases. Using half precision at inference time can reduce the latency to 25ms (40 fps) without affecting the accuracy. Compared to the most efficient agent-centric method Wayformer, we reduce the memory consumption and the online inference latency by 80% without sacrificing the accuracy.

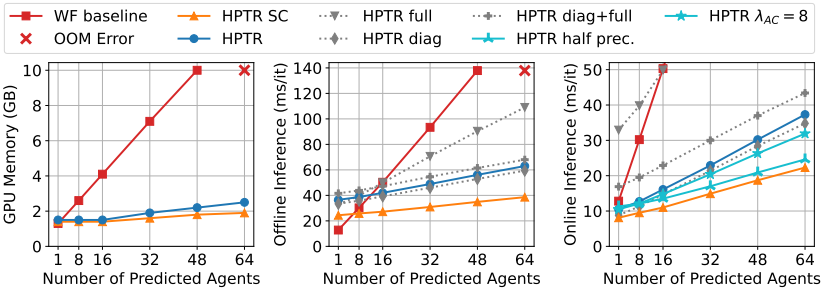


Figure 5.4: HPTR efficiency analysis. HPTR is as efficient as scene-centric methods in terms of GPU memory consumption and inference latency, while being as accurate as agent-centric methods. We use standard Ubuntu, Python and Pytorch without optimizing for real-time deployment. We predict one scenario at each inference time on one 2080Ti, i.e. we batch over scenarios and the batch size is 1. The number of context agents is the same as the number of predicted agents for all experiments. During offline inference, every inference starts from scratch, while during online inference, we cache and reuse the static map features. Specifically, we repeat the inference of the same scenario for 100 times to simulate online inference, where the static map features are computed at the first step and reused for the next 99 steps.

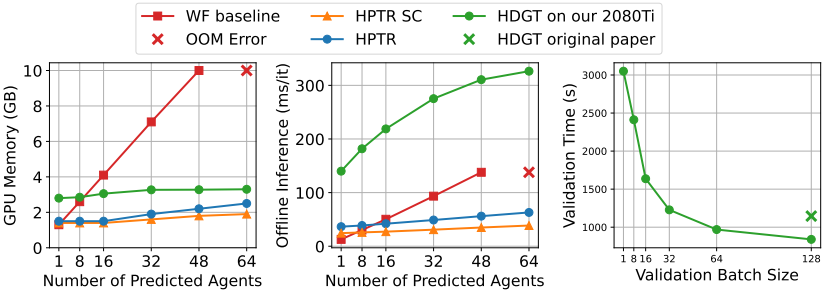
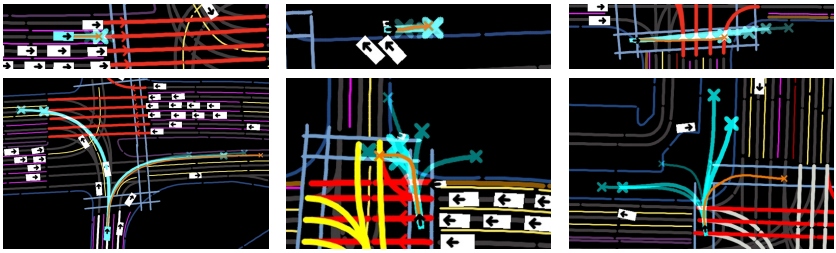


Figure 5.5: Efficiency comparison with HDGT. We run their official repository on our machine with a single 2080Ti. The left plot shows that HDGT achieves good scalability in terms of GPU memory consumption as expected. The middle plot shows that the offline inference latency (with batch size 1) of HDGT scales well, but it is significantly larger than that of other Transformer-based methods. The right plot shows the inference times for the complete WOMD validation split (64 agents per episode) with different validation batch sizes. It confirms the inference speed reported in the original HDGT paper is correctly reproduced on our setup. Our setup is faster because it has a more powerful CPU.



(a) Vehicle (b) Pedestrian (c) Cyclist

Figure 5.6: Qualitative results of HPTR. For each type of agent we show a successful case (top) and a failure case (bottom). The intersections are cluttered because we visualize traffic lights by overlaying the lanes they control with their color. The ground truth is in orange. The target agent and the predictions are in cyan. The most confident prediction has the least transparent color, the thickest line and the biggest cross. Please refer to the appendix for a detailed explanation of the visualization.

In Figure 5.5 we compare the computational efficiency of our method with the GNN-based pairwise-relative methods. Currently there are two such methods, GoRela [163] and HDGT [169]. While HDGT does not match the prediction accuracy of GoRela or our method, it is worth noting that GoRela is not open-sourced, whereas HDGT is. Therefore, we use HDGT in this efficiency comparison. The left plot of Figure 5.5 confirms the good scalability of HDGT in terms of GPU memory. This is expected because it uses the pairwise-relative representation. In the middle plot, we can observe that HDGT is slower than both our HPTR and our agent-centric Wayformer baseline in terms of offline inference speed. To confirm that HDGT runs correctly on our setup, in the right plot we reproduce the inference time of HDGT on the complete WOMD validation split with different validation batch size and we compare the reproduce numbers with the reported number in the HDGT paper. The slow inference speed of GNN-based methods such as HDGT is mainly because GNN libraries cannot utilize the GPU as efficiently as the basic matrix operations do. Our KNARPE is implemented with the most basic matrix operations, hence it is better suited for real-time and on-board applications.

Figure 5.6 illustrates the qualitative results of our HPTR. For each type of agent, we select a successful case where the most confident prediction matches the ground truth, and a failure case to show the limitation of our method. In the successful cases, we observe the vehicle stops at the red light, the pedestrian walks along the road edge with another person, and the cyclist rides across the road via the crosswalk. Although in the failure cases the most confident prediction deviates from the ground truth, we are encouraged to see that our predictions are still reasonable. The failure cases can be addressed by improving the prediction diversity via goal-conditioning, which is orthogonal to our contributions.

5.5 Conclusion

In this paper we introduce a novel attention module, KNARPE, that allows the pairwise-relative representation to be used by Transformers. Based on KNARPE, we present a pure Transformer-based framework called HPTR, which uses hierarchical architecture to enable asynchronous token update and avoid redundant computations. While agent-centric methods suffer from poor scalability and scene-centric methods suffer from poor accuracy, our HPTR gets the best from both worlds. Experiments on the two most popular benchmarks show that our approach achieves superior performance, while satisfying the real-time and on-board requirements of real-world autonomous driving.

Limitations. The poses in this work reside on a 2D plane, which can be extended to the 3D space in the future. For simplicity, we use L2 distance to obtain the K-nearest neighbors. Future works can explore more sophisticated distances which involve map topology. Currently we use the most basic anchor-based decoder which has limited diversity. This can be improved by using more advanced decoding techniques, such as goal-conditioning. In this paper we focus on marginal motion prediction. It would be interesting to investigate the potential of our approach in other prediction tasks.

Appendices

5.A Output Representation and Training Strategies

For each anchor token $\hat{\mathbf{z}}_i^{\text{AG}}, i \in \{1, \ldots, N_{\text{AG}} \cdot N_{\text{AC}}\}$, the confidence head predicts the logits $p_k, k \in \{1, \ldots, 6\}$, whereas the trajectory head predicts 6 trajectories, each of which is represented as $(\mu_x^t, \mu_y^t, \log \sigma_x^t, \log \sigma_y^t, \rho^t, v_x^t, v_y^t, \theta^t, s^t), t \in \{1, \ldots, T_f\}$, i.e. the mean of the Gaussian in x, y, the log standard deviation of the Gaussian in x, y, the correlation of the Gaussians, the velocity in x, y, the heading angle and the speed. We denote the ground truth as $(\hat{x}, \hat{y}, \hat{v}_x, \hat{v}_y, \hat{\theta}, \hat{s})$. The negative log-likelihood loss for position is formulated as

$$L_{\text{pos}} = -\log \mathcal{N}(\hat{x}, \hat{y} \mid \mu_x, \mu_y, \sigma_x, \sigma_y, \rho).$$

The negative cosine loss for the heading angle is formulated as

$$L_{\text{rot}} = -\cos(\hat{\theta} - \theta).$$

The Huber loss for velocities and speed is formulated as

$$L_{\text{vel}} = \mathcal{L}_\delta(\hat{v}_x - v_x) + \mathcal{L}_\delta(\hat{v}_y - v_y) + \mathcal{L}_\delta(\hat{s} - s),$$

where $\mathcal{L}_\delta$ is the Huber loss. We use $\delta = 1$ for all Huber losses. The final regression loss for a trajectory is the unweighted sum

$$L_{\text{traj}} = L_{\text{pos}} + L_{\text{rot}} + L_{\text{vel}},$$

which is averaged over the future time steps where the ground truth is available. We use a hard assignment strategy, i.e. among the 6 predictions of each agent we select the one that is closest to the ground truth in terms of average displacement error and optimize only for that prediction. Denoting the index of this prediction as $\hat{k}$, we train the confidence head via the cross entropy loss by taking $\hat{k}$ as the ground truth:

$$L_{\text{conf}} = -\log \frac{\exp(p_{\hat{k}})}{\sum_{i=1}^{6} \exp(p_i)}.$$

The final training loss for the complete model is the unweighted sum

$$L = L_{\text{traj}} + L_{\text{conf}}.$$

Our output representation and training strategies are the same as prior works [164, 165, 166], except for the auxiliary losses on velocities, speeds and heading angles.

5.B Implementation Details

5.B.1 Knarpe Implementation

Figure 5.7 shows how KNARPE is implemented with the most basic matrix operations, i.e. matrix indexing, summation and element-wise multiplication.

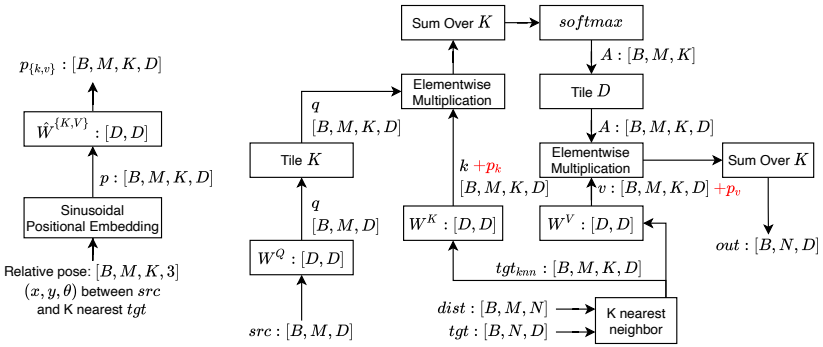


Figure 5.7: KNARPE implementation. In contrast to most GNNs, KNARPE is implemented with the most basic matrix operations. The tensor size is shown in the brackets, where B is the batch size, M is the source ($src = q$) sequence length, N is the target ($tgt = k = v$) sequence length, K in the number of neighbors, D is the hidden dimension.

5.B.2 Network Architectures

We use ReLU activation and set the hidden dimension D to 256. Our KNARPE is implemented with multi-head attention with 4 heads. We use Transformer with pre-layer normalization [215] with a dropout rate of 0.1. The feed-forward hidden dimension of Transformers is set to 1024. The base frequency ω of the sinusoidal positional encoding is set to 1000. We train a single model for all types of agents, while each type of agent has its own anchors. The polyline-line level PointNet and MLPs have 3 layers, the intra-MP Transformer encoder has 6 layers, and the inter-class as well as the AC-to-all Transformer decoders, have 2 layers. Our HPTR has 15.2M trainable parameters in total. The same setup is used for both the WOMD dataset and the AV2 dataset.

In the following, we report the configuration of ablation models. The HPTR with diagonal attention has 6 layers of intra-MP, 3 layers of intra-TL and 3 layers of intra-AG Transformer. It has 15.4M trainable parameters. The HPTR with full attention has 6 layers of all-to-all and 6 layers of AC-to-all Transformer. It has 15.2M parameters. The HPTR with diagonal followed by full attention has 6 layers of intra-MP, 2 layers of intra-TL, 2 layers of intra-AG, 2 layers of all-to-all and 2 layers of AC-to-all Transformer. It has 15.4M trainable parameters. Both the HPTR with full attention and the HPTR with diagonal followed by full attention have to be trained on GPUs with 24GB of VRAM (RTX 3090 in our case) because they require more GPU memory at training time. The scene-centric baseline uses the scene-centric representation and the standard Transformer. Following SceneTransformer [162], the input 2D positions and 2D directions are pre-processed using sinusoidal positional encoding. The base frequency is set to 1000 for 2D positions and 10 for 2D directions. The output dimension of the positional encoding is 256. This model has 13M trainable parameters. The agent-centric baseline closely follows Wayformer [164]. It has 6 layers of all-to-all Transformer and 8 layers of AC-to-all Transformer. The number of latent queries is 192. The learning rate starts at 2e-4 and it is multiplied by 0.5 every 20

epochs. The training of the agent-centric baseline takes 100 epochs to converge. We do not use auxiliary losses on velocities, speeds and yaw angles to train this model. This model has 15.6M trainable parameters.

5.B.3 Pre-Processing and Post-Processing

Our pre-processing and post-processing closely follow MPA [243]. The post-processing manipulates only the confidences via greedy non-maximum suppression. The distance threshold is 2.5m for vehicles, 1m for pedestrians and 1.5m for cyclists. We use the average displacement error to compute the distance between predicted trajectories. For AV2 we simply use softmax with a temperature of 0.5 instead of doing non-maximum suppression.

Training Details

Due to the large size of motion prediction datasets, each epoch would take a very long time if trained on the complete training split. In order to track losses more frequently, we randomly sample a fraction of all training data in each epoch. This is equivalent to using the complete training dataset if the training runs for many epochs. We observe a statistically significant correlation between the model performance and the initialization of anchors. We recommend to use a large variance for the initialization distributions. Specifically, we use Xavier initialization and multiply the initialized values by 5.

Our final models for the leaderboard submission are trained for 10 days and models for ablation and development are trained for 5 days. This long training time is because on the one hand WOMD is a very large-scale dataset, and on the other hand we only use 4 RTX 2080Ti GPUs for the training. While comparing the wall time duration of training, the computational resources should be taken into consideration. As a reference, HDGT [169] uses 8 V100 and trains for 4-5 days, GoRela [163] uses 16 GPUs (model not specified but most likely A100/V100), MTR [165] uses 8 RTX 8000, Wayformer [164] uses 16 TPU v3 cores and ProphNet [167] uses 16 V100. All of these methods use a much higher number of more powerful GPUs than we use. Given comparable computational resources, the training time of our method could be reduced to 1-2 days. In terms of sample efficiency, our method is on par with other methods. As shown in Figure 5.8, our HPTR converges after 15 epochs (5 days) and our final model is trained for 30 epochs (10 days) on WOMD. As a reference, HDGT is trained for 30 epochs, MTR is trained for 30 epochs and ProphNet is trained for 60 epochs on WOMD.

5.C Explanation of the Visualization

We use white line for lane centers of freeway, aluminium line for lane centers of surface street, dark orange line for stop sign, chocolate line for lane centers of bike lane, dark blue line for road edges boundary, dark plum line for road edges median, butter line for all types of broken road lines, magenta line for all types of solid single road lines, scarlet

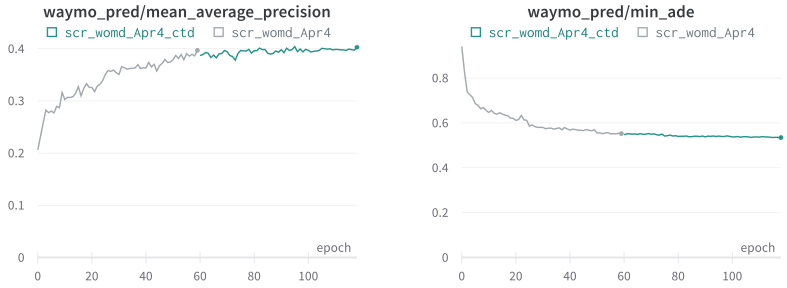


Figure 5.8: The validation mAP and minFDE logged during the training of our HPTR. The training runs on 4 RTX 2080Ti GPUs. The gray curve is trained for 5 days and the green curve continues the training for another 5 days. At each epoch, we sample 25% of the complete training split so as to do validation and log metrics more frequently. As a consequence, the effective epoch of these plots should be divided by 4, i.e. in the first 5 days we actually go through the complete WOMD training split 15 times. As we can see, training models for 5 days is enough for development. Only the models for the final leaderboard submission are trained for 10 days. The training can be speeded up given more computational resources.

red line for all types of double road lines, chameleon line for speed bumps and entrances to driveways, sky blue line for crosswalks.

The intersections are cluttered because we visualize traffic lights by overlaying the lanes they control with their color: red line for stop light state, yellow line for caution light state, green line for go light state, light aluminum line for unknown light state, violet line for flashing light state.

The ground truth is in orange. The target agent and the predictions are in cyan. Confidence are reflected by the transparency and thickness of the trajectory. The most confident prediction has the least transparent color, the thickest line and the biggest cross.

model description	concat RPE	query RPE	train on 2080ti	mem% on 3090	Min FDE ↓	Soft mAP ↑
ours	×	×	✓	58.2	1.143 ± 0.039	0.401 ± 0.007
ours without q, k, v bias	×	×	✓	58.2	1.140 ± 0.021	0.396 ± 0.009
add proj. RPE to proj. q, k, v	×	✓	OOM	66.2	1.144 ± 0.036	0.397 ± 0.006
concat. RPE to k, v	✓	×	OOM	71.6	1.138 ± 0.026	0.395 ± 0.006
concat. RPE to q, k, v	✓	✓	OOM	90.5	1.133 ± 0.024	0.396 ± 0.006

Table 5.3: Ablation on WOMD valid split. We study different ways to incorporate the RPE into the dot-product attention. Performance is reported as the mean plus-minus 3 standard deviations over 3 training seeds. Models are trained for 60 epochs. OOM: out of memory. q: query. k: key. v: value.

minFDE_6 ↓	minFDE_1 ↓	minADE_6 ↓	minADE_1 ↓	Miss Rate_6 ↓	Miss Rate_1 ↓	brier-minFDE_6 ↓
1.43	4.61	0.73	1.84	0.19	0.61	2.03

Table 5.4: Complete results of our HPTR on the AV2 *test* split.

5.D Additional Ablation Studies

In Table 5.3 we ablate different ways to incorporate RPE into the dot-product attention. The differences are insignificant in terms of performance. However, our approach, i.e. adding projected RPE to projected key and value, consumes less memory at training time. We use this setup in our main paper because it can be trained on the RTX 2080 Ti GPUs (12GB VRAM), which are more accessible than the RTX 3090 GPUs (24GB VRAM) in practice.

5.E Additional Quantitative Results

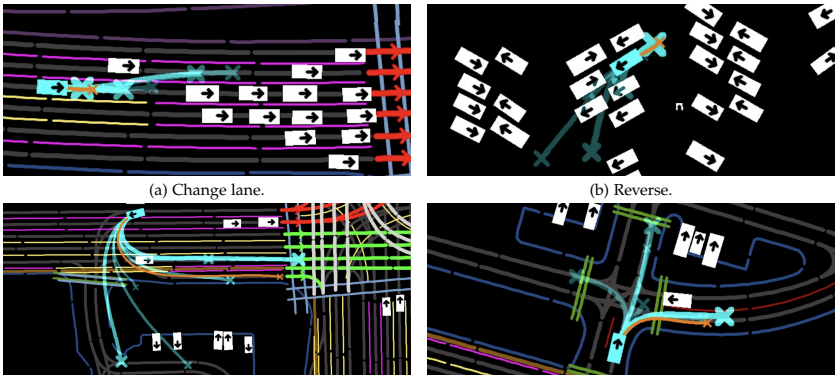
In Tables 5.5, 5.6, and 5.4, we provide the complete results of our HPTR on the AV2 *test* split, the WOMD *test* split, and the WOMD *valid* split, respectively.

Object Type	Measurement Time (s)	Soft mAP ↑	mAP ↑	Min ADE ↓	Min FDE ↓	Miss Rate ↓	Overlap Rate ↓
Vehicle	3	0.5631	0.5475	0.2795	0.4997	0.0927	0.0190
Vehicle	5	0.4687	0.4623	0.5714	1.1020	0.1297	0.0415
Vehicle	8	0.3697	0.3664	1.0739	2.2753	0.1787	0.0915
Vehicle	Avg	0.4671	0.4587	0.6416	1.2923	0.1337	0.0507
Pedestrian	3	0.4534	0.4427	0.1637	0.3111	0.0676	0.2408
Pedestrian	5	0.3422	0.3370	0.3220	0.6616	0.0938	0.2648
Pedestrian	8	0.2792	0.2751	0.5722	1.2778	0.1248	0.2952
Pedestrian	Avg	0.3582	0.3516	0.3526	0.7502	0.0954	0.2669
Cyclist	3	0.4334	0.4267	0.3266	0.6078	0.1859	0.0494
Cyclist	5	0.3587	0.3552	0.6166	1.2085	0.1922	0.0900
Cyclist	8	0.3025	0.3006	1.0825	2.3096	0.2250	0.1369
Cyclist	Avg	0.3649	0.3608	0.6752	1.3753	0.2011	0.0921
Avg	3	0.4833	0.4723	0.2566	0.4729	0.1154	0.1030
Avg	5	0.3899	0.3848	0.5033	0.9907	0.1386	0.1321
Avg	8	0.3171	0.3140	0.9095	1.9543	0.1762	0.1745
Avg	Avg	0.3968	0.3904	0.5565	1.1393	0.1434	0.1366

Table 5.5: Complete results of our HPTR on the WOMD *test* split.

Object Type	Measurement Time (s)	Soft mAP ↑	mAP ↑	Min ADE ↓	Min FDE ↓	Miss Rate ↓	Overlap Rate ↓
Vehicle	3	0.5611	0.5451	0.2796	0.4988	0.0934	0.0186
Vehicle	5	0.4704	0.4637	0.5698	1.0986	0.1297	0.0405
Vehicle	8	0.3678	0.3644	1.0731	2.2909	0.1824	0.0909
Vehicle	Avg	0.4664	0.4577	0.6408	1.2961	0.1352	0.0500
Pedestrian	3	0.4923	0.4802	0.1454	0.2674	0.0478	0.2358
Pedestrian	5	0.4055	0.3993	0.2782	0.5488	0.0661	0.2605
Pedestrian	8	0.3639	0.3577	0.4834	1.0157	0.0813	0.2901
Pedestrian	Avg	0.4206	0.4124	0.3023	0.6106	0.0651	0.2621
Cyclist	3	0.4606	0.4519	0.3309	0.6021	0.1779	0.0523
Cyclist	5	0.3822	0.3788	0.6136	1.1843	0.1905	0.0897
Cyclist	8	0.2962	0.2943	1.0661	2.3242	0.2239	0.1433
Cyclist	Avg	0.3797	0.3750	0.6702	1.3702	0.1974	0.0951
Avg	3	0.5047	0.4924	0.2519	0.4561	0.1064	0.1022
Avg	5	0.4194	0.4139	0.4872	0.9439	0.1288	0.1302
Avg	8	0.3427	0.3388	0.8742	1.8769	0.1626	0.1748
Avg	Avg	0.4222	0.4150	0.5378	1.0923	0.1326	0.1357

Table 5.6: Complete results of our HPTR on the WOMD *valid* split.



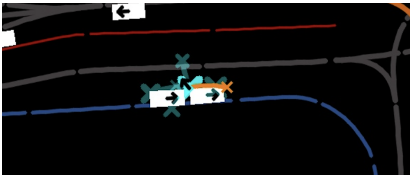
(a) Change lane. (b) Reverse.

(c) U-Turn. (d) Multi-modal futures.

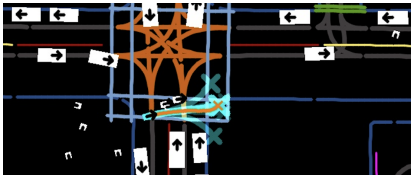
Figure 5.9: Qualitative results of HPTR predicting *vehicles*. Scenarios are selected from the WOMD validation dataset. The ground truth is in orange. The target agent and the predictions are in cyan. The most confident prediction has the least transparent color, the thickest line and the biggest cross.

5.F Additional Qualitative Results

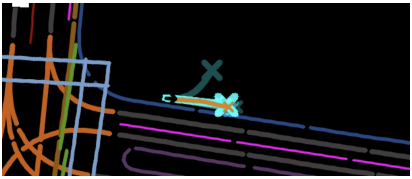
In Figures 5.9, 5.10, and 5.11, we provide more qualitative results on WOMD *valid* of our HPTR predicting vehicles, pedestrians, and cyclists, respectively.



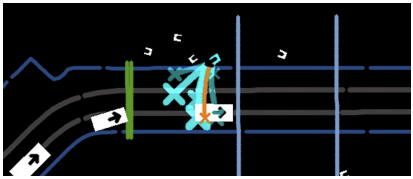
(a) Interact with parked cars.



(b) On the crosswalk with other people.

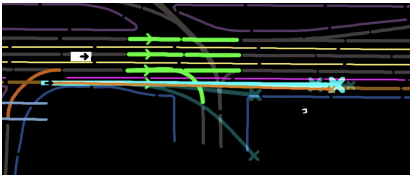


(c) Walk alone on the road edge.

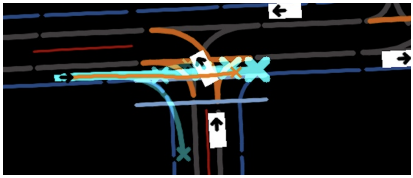


(d) Cross the street without using crosswalk.

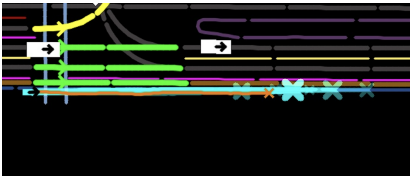
Figure 5.10: Qualitative results of HPTR predicting *pedestrians*. Read as Figure 5.9.



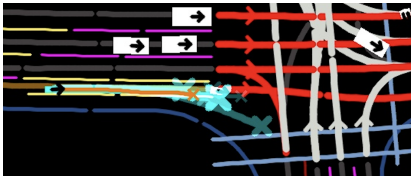
(a) Ride on the bike lane (in brown).



(b) Ride on a road without bike lane.



(c) Ride on the road edge.



(d) Stop at red light.

Figure 5.11: Qualitative results of HPTR predicting *cyclists*. Read as Figure 5.9.

6

Summary and Perspectives

In this thesis, we explore techniques for training neural policies for both the autonomous vehicles (AV) and non-playable simulated traffic agents. Here, we provide an overview of our key contributions, and a discussion on future research directions.

6.1 Summary of Contributions

In Chapter 2, we presented Roach, a reinforcement learning (RL) expert, and an effective way to imitate this expert. Using the bird's-eye view representation, Beta distribution and the exploration loss, Roach set the new performance upper-bound on CARLA while demonstrating high sample efficiency. To enable a more effective imitation, we proposed to learn from soft targets, values and latent features generated by Roach. Supervised by these informative targets, a baseline end-to-end (E2E) imitation learning agent using a single camera image as input can achieved state-of-the-art performance, even reaching expert-level performance on the NoCrash-dense benchmark.

In Chapter 3, we alleviated the poor sample efficiency of the Roach expert by introducing a safety critic to yield a multiplicative value function. We started with the constrained Markov decision process formulation, derived the safety critic from reachability analysis and integrated our approach into the Soft Actor-Critic (SAC) and the Proximal Policy Optimization (PPO) framework. We proposed several versions of SAC and PPO using our multiplicative value function and showed increased sample efficiency and stability compared to the baselines. Furthermore, the multiplicative value function can help to learn the fine details in the reward structure, like soft constraints. To show the real-world potential of our method, we took a SAC Mult Lagrange agent trained in simulation and successfully deployed the policy on a real robot in a zero-shot sim-to-real fashion. The robot showed safe behavior and was able to generalize to dynamic obstacles of novel shape.

To improve the behavioral realism of the CARLA simulator, in Chapter 4 we presented TrafficBots, a multi-agent policy learned from real-world motion prediction datasets. Based on the shared, vectorized context and the individual personality and destination, TrafficBots can generate realistic multi-agent behaviors in dense urban scenarios. Besides the simulation, TrafficBots can also be used for motion prediction. Evaluating on motion prediction tasks allows us to verify the simulation fidelity and benchmark on a public leaderboard. Based on TrafficBots, we built a differentiable, data-driven simulation

framework, which in the future can serve as a platform to develop AV planning algorithms, or as a world model to train E2E driving policies via RL [35] or model-based imitation learning [99]. Moreover, TrafficBots could also be integrated as a module to generate human-like behaviors for bot agents in a game or a full-stack autonomous driving simulator.

Finally, in Chapter 5, we proposed a novel network architecture that resolved the trade-off between the accuracy and efficiency of motion prediction methods. Specifically, we introduced a novel attention module, KNARPE, which allows the pairwise-relative representation to be used by Transformers. Based on KNARPE, we presented a pure Transformer-based framework called HPTR, which uses hierarchical architecture to enable asynchronous token update and avoid redundant computations. While agent-centric methods suffer from poor scalability and scene-centric methods suffer from poor accuracy, our HPTR gets the best from both worlds. Experiments on the two most popular benchmarks showed that our approach achieves superior performance, while satisfying the real-time and on-board requirements of real-world autonomous driving.

6.2 Discussion and Future Perspectives

In this thesis, we have taken a step forward in the field of learning neural policies to facilitate prosocial navigation for AV and robots. However, there is still a long way to go before AV can drive safely in dense urban scenarios and behave like human experts on public roads. In the following, we briefly discuss what we consider to be some of the most promising and important directions for the future of E2E driving and AV planning.

Training ego policy in world models: In this thesis, we presented a world model and focused on improving its configurability, fidelity, scalability and efficiency. However, we have yet to use this world model to train a policy for the ego vehicle. Recent studies [246, 247] have explored this direction, but it remains unclear whether training the AV policy against data-driven reactive agents is more effective than training against log-replay or rule-based simulated agents [248, 249, 250]. Besides planning-oriented world models, image-based world models [128, 132], and video generation models [251, 252] are also promising future directions, but they have not been utilized for policy training or testing yet.

Offline RL: In addition to world models, another promising approach to leverage the vast offline datasets of AV is offline RL, i.e., RL algorithms that utilize previously collected data, without additional online data collection or explicitly training a model [253]. Using sequence modeling [254, 255] or diffusion models [256], significant progress has been made by recent studies for robotics applications. However, applying offline RL to more complex and safety-critical environments involving close interaction with humans, such as autonomous driving in dense urban scenarios, remains an open challenge.

RL from human feedback: There are two reasons why RL from human feedback (RLHF) could be the next big thing for AV. The first reason is the success of RLHF applied to large language models (LLM) such as GPT [257] and Llama [258]. The second reason is that driving in proximity to humans requires prosocial navigation, where safety is not the only criterion; human acceptability has to be considered as well. RLHF for driving is also

aligned with how humans learn to drive in a driving school. It would be beneficial for AVs to learn from comments provided by human experts about their behavior, enabling them to minimize the probability of accidents without actually experiencing one, as well as improving driving skills without requiring concrete expert demonstration.

Vision-language models for AV planning: Current planning and E2E driving algorithms still operate at a relatively basic level, similar to human instinct, which considers short history and lacks high-level reasoning ability. In order to solve the long-tail problems encountered in real-world scenarios, the AV planning algorithms need the emergent ability [259] that has been observed in LLM and vision language models (VLM) [260]. Using VLM trained on internet resources as the brain of AV is also aligned with how humans use their common sense and theoretical knowledge for driving. Before the policy or the human actually maneuvers the vehicle, it should already be clear what needs to be done in principle, following common sense and the theoretical knowledge acquired beforehand. In fact, current VLMs have the common sense that can explain the situation given an example image of long-tail events on the road, such as unknown objects or uncommon behavior of pedestrians. In the near future, VLMs should also be able to pass the theory test of driving and prove its understanding of traffic rules. However, it is still an open question how to use VLMs and LLMs for real-time planing and E2E urban driving tasks [261, 262, 263, 264, 265, 266].

Dataset and benchmark for AV planning and E2E driving: Datasets and benchmarks are still a big bottleneck for data-driven planning and simulation for AV. Current AV datasets and benchmarks inherit the design for perception tasks, which is not optimal for planning tasks. While most motion prediction datasets today do not include sensor measurements, it is important for the planning tasks to know the visual details to reason about the behavior of surrounding vehicles and pedestrians, especially when dealing with long-tail events. As a result, we believe an ideal dataset for AV planning and E2E driving should have the following attributes:

- Collected in the real world.
- High-quality, temporally synchronized data, including various sensor measurements, intermediate perception outputs, as well as navigation information and the actions of the ego vehicle.
- Sufficient details for driving tasks, including, for example, signal lights of vehicles, gestures from pedestrians and cyclists.
- Large scale with high diversity, including, for example, dense urban driving scenarios involving interactions with humans, as well as uncommon behavior and confusing situations.

Building such a dataset is an ambitious but necessary task. None of the current AV datasets meet all these requirements, but datasets such as NuPlan [100] and WOMD [134] have been working towards these directions in their latest updates.

In terms of the benchmark, current real-world planning benchmarks adopt an offline setting and use offline metrics as a proxy for online metrics. While closed-loop E2E evaluation is desirable, it remains unclear how to achieve this with offline real-world datasets [104, 248]. One promising approach is to use real-world data to minimize the sim-to-real gap, with the hope that simulators can achieve sufficient fidelity to replace

real-world environment for driving tasks [128, 267]. In the future, we expect to see a standardized driving test in simulation that all AVs must pass before entering public roads.

Bibliography

[1] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. "A survey of deep learning techniques for autonomous driving". In: *Journal of field robotics* 2020.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. "A survey of autonomous driving: Common practices and emerging technologies". In: *IEEE access* 2020.

[3] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. "End-to-end interpretable neural motion planner". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[4] M. Bansal, A. Krizhevsky, and A. S. Ogale. "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst". In: *Robotics: Science and Systems XV*. 2019.

[5] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, et al. "Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2023.

[6] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. "End to end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316* 2016.

[7] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone. "Diffstack: A differentiable and modular control stack for autonomous vehicles". In: *Conference on robot learning (CoRL)*. 2023.

[8] Waabi. *Introducing the Waabi Driver*. `https://waabi.ai/introducing-the-waabi-driver/`. Accessed: 2024-03-13. 2022.

[9] electrek.co. *Tesla finally releases FSD v12, its last hope for self-driving*. `https://electrek.co/2024/01/22/tesla-releases-fsd-v12-last-hope-self-driving/`. Accessed: 2024-03-13. 2024.

[10] Wayve. *A new approach to self-driving: AV2.0*. `https://wayve.ai/thinking/a-new-approach-to-self-driving-av2-0/`. Accessed: 2024-03-13. 2021.

[11] S. Ross, G. Gordon, and D. Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. "CARLA: An Open Urban Driving Simulator". In: *Conference on Robot Learning (CoRL)*. 2017.

[13] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.-C. Ma, A. J. Yang, and R. Urtasun. "Unisim: A neural closed-loop sensor simulator". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[14] J. Yang, B. Ivanovic, O. Litany, X. Weng, S. W. Kim, B. Li, T. Che, D. Xu, S. Fidler, M. Pavone, et al. "Emernerf: Emergent spatial-temporal scene decomposition via self-supervision". In: *arXiv preprint arXiv:2311.02077* 2023.

[15] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, and R. Urtasun. "Lidarsim: Realistic lidar simulation by leveraging the real world". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[16] CARLA. *CARLA Autonomous Driving Leaderboard: Traffic Scenarios*. `https://leaderboard.carla.org/scenarios/`. Accessed: 2024-03-13. 2023.

[17] D. Xu, Y. Chen, B. Ivanovic, and M. Pavone. "Bits: Bi-level imitation for traffic simulation". In: *International Conference on Robotics and Automation (ICRA)*. 2023.

[18] S. Suo, S. Regalado, S. Casas, and R. Urtasun. "TrafficSim: Learning to Simulate Realistic Multi-Agent Behaviors". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[19] D. Pomerleau. "ALVINN: An Autonomous Land Vehicle In a Neural Network". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1989.

[20] S. Teng, X. Hu, P. Deng, B. Li, Y. Li, Y. Ai, D. Yang, L. Li, Z. Xuanyuan, F. Zhu, et al. "Motion planning for autonomous driving: The state of the art and future perspectives". In: *IEEE Transactions on Intelligent Vehicles* 2023.

[21] N. Webb, D. Smith, C. Ludwick, T. Victor, Q. Hommes, F. Favaro, G. Ivanov, and T. Daniel. "Waymo's safety methodologies and safety readiness determinations". In: *arXiv preprint arXiv:2011.00054* 2020.

[22] S. Ingle and M. Phute. "Tesla autopilot: semi autonomous driving, an uptick for future autonomy". In: *International Research Journal of Engineering and Technology* 2016.

[23] R. L. McCarthy. "Autonomous vehicle accident data analysis: California OL 316 reports: 2015–2020". In: *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering* 2022.

[24] J. Wang, X. Wang, T. Shen, Y. Wang, L. Li, Y. Tian, H. Yu, L. Chen, J. Xin, X. Wu, et al. "Parallel vision for long-tail regularization: Initial results from IVFC autonomous driving testing". In: *IEEE Transactions on Intelligent Vehicles* 2022.

[25] T. Guardian. *Cruise robotaxi service hid severity of accident, California officials claim*. `https://www.theguardian.com/business/2023/dec/04/california-cruise-robotaxi-san-francisco-accident-severity`. Accessed: 2024-03-13. 2023.

[26] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li. "Planning-oriented Autonomous Driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[27] A. Prakash, K. Chitta, and A. Geiger. "Multi-modal fusion transformer for end-to-end autonomous driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[28] S. Hu, L. Chen, P. Wu, H. Li, J. Yan, and D. Tao. "St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning". In: *European Conference on Computer Vision (ECCV)*. 2022.

[29] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger. "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 2022.

[30] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. "End-to-end driving via conditional imitation learning". In: *International Conference on Robotics and Automation (ICRA)*. 2018.

[31] F. Codevilla, E. Santana, A. M. López, and A. Gaidon. "Exploring the limitations of behavior cloning for autonomous driving". In: *International Conference on Computer Vision (ICCV)*. 2019.

[32] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger. "Exploring data aggregation in policy learning for vision-based urban autonomous driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[33] E. Ohn-Bar, A. Prakash, A. Behl, K. Chitta, and A. Geiger. "Learning situational driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[34] S. Hecker, D. Dai, A. Liniger, M. Hahner, and L. Van Gool. "Learning accurate and human-like driving using semantic maps and attention". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[35] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool. "End-to-end urban driving by imitating a reinforcement learning coach". In: *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*. 2021.

[36] A. Behl, K. Chitta, A. Prakash, E. Ohn-Bar, and A. Geiger. "Label Efficient Visual Abstractions for Autonomous Driving". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[37] A. Sauer, N. Savinov, and A. Geiger. "Conditional affordance learning for driving in urban environments". In: *Conference on Robot Learning (CoRL)*. 2018.

[38] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. "Deepdriving: Learning affordance for direct perception in autonomous driving". In: *International Conference on Computer Vision (ICCV)*. 2015.

[39] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. "Learning by cheating". In: *Conference on Robot Learning (CoRL)*. 2020.

[40] A. Zhao, T. He, Y. Liang, H. Huang, G. V. den Broeck, and S. Soatto. "SAM: Squeeze-and-Mimic Networks for Conditional Visual Driving Policy Learning". In: *Conference on Robot Learning (CoRL)*. 2020.

[41] S. Casas, A. Sadat, and R. Urtasun. "Mp3: A unified model to map, perceive, predict and plan". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[42] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li. "End-to-end autonomous driving: Challenges and frontiers". In: *arXiv preprint arXiv:2306.16927* 2023.

[43] P. S. Chib and P. Singh. "Recent advancements in end-to-end autonomous driving using deep learning: A survey". In: *IEEE Transactions on Intelligent Vehicles* 2023.

[44] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus. "Learning robust control policies for end-to-end autonomous driving from data-driven simulation". In: *IEEE Robotics and Automation Letters (RA-L)* 2020.

[45] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. "Learning to drive in a day". In: *International Conference on Robotics and Automation (ICRA)*. 2019.

[46] Z. M. Peng, W. Mo, C. Duan, Q. Li, and B. Zhou. "Learning from active human involvement through proxy value propagation". In: *Advances in neural information processing systems* 2024.

[47] Y. Pan, C. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots. "Agile Autonomous Driving using End-to-End Deep Imitation Learning". In: *Robotics: Science and Systems (RSS)*. 2018.

[48] Y. Hou, Z. Ma, C. Liu, and C. C. Loy. "Learning to steer by mimicking features from heterogeneous auxiliary networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2019.

[49] J. Ho and S. Ermon. "Generative Adversarial Imitation Learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[50] P. Abbeel and A. Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2004.

[51] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing atari with deep reinforcement learning". In: *arXiv* 2013.

[52] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. "Mastering the game of go without human knowledge". In: *Nature* 2017.

[53] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 2019.

[54] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. "Dota 2 with large scale deep reinforcement learning". In: *arXiv* 2019.

[55] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning." In: *International Conference on Learning Representations (ICLR)*. 2016.

[56] H. v. Hasselt, A. Guez, and D. Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2016.

[57] S. Fujimoto, H. Hoof, and D. Meger. "Addressing function approximation error in actor-critic methods". In: *International Conference on Machine Learning (ICML)*. 2018.

[58] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International Conference on Machine Learning (ICML)*. 2018.

[59] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2016.

[60] X. Liang, T. Wang, L. Yang, and E. Xing. "Cirl: Controllable imitative reinforcement learning for vision-based self-driving". In: *European Conference on Computer Vision (ECCV)*. 2018.

[61] J. Chen, B. Yuan, and M. Tomizuka. "Model-free deep reinforcement learning for urban autonomous driving". In: *IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019.

[62] M. Toromanoff, E. Wirbel, and F. Moutarde. "Is Deep Reinforcement Learning Really Superhuman on Atari?" In: *Deep Reinforcement Learning Workshop of the Conference on Neural Information Processing Systems*. 2019.

[63] G. Kahn, P. Abbeel, and S. Levine. "LaND: Learning to Navigate from Disengagements". In: *IEEE Robotics and Automation Letters (R-AL)* 2021.

[64] M. Toromanoff, E. Wirbel, and F. Moutarde. "End-to-end model-free reinforcement learning for urban driving using implicit affordances". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[65] N. Rhinehart, R. McAllister, and S. Levine. "Deep Imitative Models for Flexible Inference, Planning, and Control". In: *International Conference on Learning Representations (ICLR)*. 2020.

[66] P. Palanisamy. "Multi-agent connected autonomous driving using deep reinforcement learning". In: *International Joint Conference on Neural Networks (IJCNN)*. 2020.

[67] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez. "Deep reinforcement learning for autonomous driving: A survey". In: *IEEE Transactions on Intelligent Transportation Systems* 2021.

[68] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. "Risk-constrained reinforcement learning with percentile risk criteria". In: *The Journal of Machine Learning Research* 2017.

[69] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn. "Learning to be safe: Deep rl with a safety critic". In: *arXiv preprint arXiv:2010.14603* 2020.

[70] A. Stooke, J. Achiam, and P. Abbeel. "Responsive safety in reinforcement learning by pid lagrangian methods". In: *International Conference on Machine Learning (ICML)*. 2020.

[71] N. Bührer, Z. Zhang, A. Liniger, F. Yu, and L. Van Gool. "A Multiplicative Value Function for Safe and Efficient Reinforcement Learning". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2023.

[72] P. Geibel and F. Wysotzki. "Risk-sensitive reinforcement learning applied to control under constraints". In: *Journal of Artificial Intelligence Research* 2005.

[73] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan. "WCSAC: Worst-case soft actor critic for safety-constrained reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2021.

[74] C. Tessler, D. J. Mankowitz, and S. Mannor. "Reward constrained policy optimization". In: *International Conference on Learning Representations (ICLR)*. 2019.

[75] S. Paternain, L. Chamon, M. Calvo-Fullana, and A. Ribeiro. "Constrained reinforcement learning has zero duality gap". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[76] J. Achiam, D. Held, A. Tamar, and P. Abbeel. "Constrained policy optimization". In: *International Conference on Machine Learning (ICML)*. 2017.

[77] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. "Projection-Based Constrained Policy Optimization". In: *International Conference on Learning Representations (ICLR)*. 2020.

[78] Y. Zhang, Q. Vuong, and K. Ross. "First order constrained optimization in policy space". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[79] Y. Liu, J. Ding, and X. Liu. "IPO: Interior-point policy optimization under constraints". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2020.

[80] T. Xu, Y. Liang, and G. Lan. "Crpo: A new approach for safe reinforcement learning with convergence guarantee". In: *International Conference on Machine Learning (ICML)*. 2021.

[81] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. "Safe model-based reinforcement learning with stability guarantees". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[82] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh. "Safe Policy Learning for Continuous Control". In: *Conference on Robot Learning (CoRL)*. 2020.

[83] S. Huh and I. Yang. "Safe reinforcement learning for probabilistic reachability and safety specifications: A Lyapunov-based approach". In: *arXiv preprint arXiv:2002.10126* 2020.

[84] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, and Y. Zhao. "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles". In: *IEEE Transactions on Neural Networks and Learning Systems* 2021.

[85] N. C. Wagener, B. Boots, and C.-A. Cheng. "Safe reinforcement learning using advantage-based intervention". In: *International Conference on Machine Learning (ICML)*. 2021.

[86] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2019.

[87] K. P. Wabersich and M. N. Zeilinger. "Predictive control barrier functions: Enhanced safety mechanisms for learning-based control". In: *IEEE Transactions on Automatic Control* 2022.

[88] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac. "Sim-to-Lab-to-Real: Safe reinforcement learning with shielding and generalization guarantees". In: *Artificial Intelligence* 2023.

[89] Y. Liu, A. Halev, and X. Liu. "Policy Learning with Constraints in Model-free Reinforcement Learning: A Survey." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2021.

[90] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll. "A review of safe reinforcement learning: Methods, theory and applications". In: *arXiv preprint arXiv:2205.10330* 2022.

[91] S. W. Kim, J. Philion, A. Torralba, and S. Fidler. "Drivegan: Towards a controllable high-quality neural simulation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[92] A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus. "Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles". In: *International Conference on Robotics and Automation (ICRA)*. 2022.

[93] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar. "Block-nerf: Scalable large scene neural view synthesis". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[94] S. Huang, Z. Gojcic, Z. Wang, F. Williams, Y. Kasten, S. Fidler, K. Schindler, and O. Litany. "Neural lidar fields for novel view synthesis". In: *International Conference on Computer Vision (ICCV)*. 2023.

[95] B. Shen, X. Yan, C. R. Qi, M. Najibi, B. Deng, L. Guibas, Y. Zhou, and D. Anguelov. "Gina-3d: Learning to generate implicit neural assets in the wild". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[96] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun. "Geosim: Realistic video simulation via geometry-aware composition for self-driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[97] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. "Microscopic traffic simulation using sumo". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[98] E. Leurent. *An Environment for Autonomous Driving Decision-Making*. `https://github.com/eleurent/highway-env`. 2018.

[99] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska. "Urban driver: Learning to drive from real-world demonstrations using policy gradients". In: *Conference on Robot Learning (CoRL)*. 2022.

[100] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari. "nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles". In: *arXiv preprint arXiv:2106.11810* 2021.

[101] M. Zhou, J. Luo, J. Villela, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, and Z. Chen. "SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving". In: *Conference on Robot Learning (CoRL)*. 2020.

[102] C. Gulino, J. Fu, W. Luo, G. Tucker, E. Bronstein, Y. Lu, J. Harb, X. Pan, Y. Wang, X. Chen, et al. "Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2024.

[103] J. Bernhard, K. Esterle, P. Hart, and T. Kessler. "BARK: Open behavior benchmarking in multi-agent environments". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[104] N. Contributors. *NAVSim: Data-Driven Non-Reactive Autonomous Vehicle Simulation*. `https://github.com/autonomousvision/navsim`. 2024.

[105] M. Igl, D. Kim, A. Kuefler, P. Mougin, P. Shah, K. Shiarlis, D. Anguelov, M. Palatucci, B. White, and S. Whiteson. "Symphony: Learning Realistic and Diverse Agents for Autonomous Driving Simulation". In: *International Conference on Robotics and Automation (ICRA)*. 2022.

[106] F. Behbahani, K. Shiarlis, X. Chen, V. Kurin, S. Kasewa, C. Stirbu, J. Gomes, S. Paul, F. A. Oliehoek, J. Messias, et al. "Learning from demonstration in the wild". In: *International Conference on Robotics and Automation (ICRA)*. 2019.

[107] R. Bhattacharyya, B. Wulfe, D. J. Phillips, A. Kuefler, J. Morton, R. Senanayake, and M. J. Kochenderfer. "Modeling human driving behavior through generative adversarial imitation learning". In: *IEEE Transactions on Intelligent Transportation Systems* 2022.

[108] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer. "Multi-agent imitation learning for driving simulation". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2018.

[109] L. Bergamini, Y. Ye, O. Scheel, L. Chen, C. Hu, L. Del Pero, B. Osiński, H. Grimmett, and P. Ondruska. "Simnet: Learning reactive self-driving simulations from real-world observations". In: *International Conference on Robotics and Automation (ICRA)*. 2021.

[110] A. Kamenev, L. Wang, O. B. Bohan, I. Kulkarni, B. Kartal, A. Molchanov, S. Birchfield, D. Nistér, and N. Smolyanskiy. "Predictionnet: Real-time joint probabilistic traffic prediction for planning, control, and simulation". In: *International Conference on Robotics and Automation (ICRA)*. 2022.

[111] Z. Zhong, D. Rempe, D. Xu, Y. Chen, S. Veer, T. Che, B. Ray, and M. Pavone. "Guided conditional diffusion for controllable traffic simulation". In: *International Conference on Robotics and Automation (ICRA)*. 2023.

[112] Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. Van Gool. "TrafficBots: Towards World Models for Autonomous Driving Simulation and Motion Prediction". In: *International Conference on Robotics and Automation (ICRA)*. 2023.

[113] D. Rempe, Z. Luo, X. Bin Peng, Y. Yuan, K. Kitani, K. Kreis, S. Fidler, and O. Litany. "Trace and pace: Controllable pedestrian animation via guided trajectory diffusion". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[114] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany. "Generating useful accident-prone driving scenarios via a learned traffic prior". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[115] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun. "Advsim: Generating safety-critical scenarios for self-driving vehicles". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[116] D. Chen, M. Zhu, H. Yang, X. Wang, and Y. Wang. "Data-driven Traffic Simulation: A Comprehensive Review". In: *arXiv preprint arXiv:2310.15975* 2023.

[117] D. Ha and J. Schmidhuber. "Recurrent world models facilitate policy evolution". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[118] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. "Learning latent dynamics for planning from pixels". In: *International Conference on Machine Learning (ICML)*. 2019.

[119] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. "Dream to Control: Learning Behaviors by Latent Imagination". In: *International Conference on Learning Representations (ICLR)*. 2020.

[120] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. "Mastering diverse domains through world models". In: *arXiv preprint arXiv:2301.04104* 2023.

[121] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. "Mastering Atari with Discrete World Models". In: *International Conference on Learning Representations (ICLR)*. 2021.

[122] M. Henaff, A. Canziani, and Y. LeCun. "Model-Predictive Policy Learning with Uncertainty Regularization for Driving in Dense Traffic". In: *International Conference on Learning Representations (ICLR)*. 2019.

[123] J. Ho, A. Jain, and P. Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[124] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. "High-resolution image synthesis with latent diffusion models". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[125] W. Peebles and S. Xie. "Scalable diffusion models with transformers". In: *International Conference on Computer Vision (ICCV)*. 2023.

[126] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. "Video diffusion models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[127] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. "Imagen video: High definition video generation with diffusion models". In: *arXiv preprint arXiv:2210.02303* 2022.

[128] A. Hu, L. Russell, H. Yeo, Z. Murez, G. Fedoseev, A. Kendall, J. Shotton, and G. Corrado. "Gaia-1: A generative world model for autonomous driving". In: *arXiv preprint arXiv:2309.17080* 2023.

[129] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. "Daydreamer: World models for physical robot learning". In: *Conference on Robot Learning (CoRL)*. 2023.

[130] Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. "Masked world models for visual control". In: *Conference on Robot Learning (CoRL)*. 2023.

[131] A. Hu, G. Corrado, N. Griffiths, Z. Murez, C. Gurau, H. Yeo, A. Kendall, R. Cipolla, and J. Shotton. "Model-based imitation learning for urban driving". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[132] X. Wang, Z. Zhu, G. Huang, X. Chen, and J. Lu. "Drivedreamer: Towards real-world-driven world models for autonomous driving". In: *arXiv preprint arXiv:2309.09777* 2023.

[133] Y. Guan, H. Liao, Z. Li, G. Zhang, and C. Xu. "World Models for Autonomous Driving: An Initial Survey". In: *arXiv preprint arXiv:2403.02622* 2024.

[134] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou, et al. "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset". In: *International Conference on Computer Vision (ICCV)*. 2021.

[135] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al. "Argoverse 2: Next generation datasets for self-driving perception and forecasting". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[136] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. "nuScenes: A multimodal dataset for autonomous driving". In: *arXiv preprint arXiv:1903.11027* 2019.

[137] Waymo. *Waymo open dataset motion prediction challenge 2022*. `https://waymo.com/open/challenges/2022/motion-prediction/`. Accessed: 2023-05-10. 2022.

[138] Argoverse. *Argoverse 2: Motion forecasting competition*. `https://eval.ai/web/challenges/challenge-page/1719/leaderboard/4098`. Accessed: 2023-05-10. 2022.

[139] nuScenes. *nuScenes prediction challenge*. `https://eval.ai/web/challenges/challenge-page/591/leaderboard/1659`. Accessed: 2023-05-10. 2020.

[140] Waymo. *Waymo open dataset sim agent challenge 2023*. `https://waymo.com/open/challenges/2023/sim-agents/`. Accessed: 2023-05-10. 2023.

[141] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, and C. Schmid. "Tnt: Target-Driven Trajectory Prediction". In: *Conference on Robot Learning (CoRL)*. 2020.

[142] J. Gu, C. Sun, and H. Zhao. "Densetnt: End-to-end trajectory prediction from dense goal sets". In: *International Conference on Computer Vision (ICCV)*. 2021.

[143] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine. "Precog: Prediction conditioned on goals in visual multi-agent settings". In: *International Conference on Computer Vision (ICCV)*. 2019.

[144] N. Deo, E. Wolff, and O. Beijbom. "Multimodal trajectory prediction conditioned on lane-graph traversals". In: *Conference on Robot Learning (CoRL)*. 2022.

[145] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker. "Desire: Distant future prediction in dynamic scenes with interacting agents". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[146] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun. "Implicit latent variable model for scene-consistent motion forecasting". In: *European Conference on Computer Vision (ECCV)*. 2020.

[147] C. Tang and R. R. Salakhutdinov. "Multiple futures prediction". In: *Advances in Neural Information Processing Systems (NeurIPS)* 2019.

[148] B. Ivanovic and M. Pavone. "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs". In: *International Conference on Computer Vision (ICCV)*. 2019.

[149] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. "Multipath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction". In: *Conference on Robot Learning (CoRL)*. 2019.

[150] S. Hochreiter and J. Schmidhuber. "Long short-term memory". In: *Neural computation* 1997.

[151] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* 2014.

[152] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. "Social lstm: Human trajectory prediction in crowded spaces". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[153] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo. "Mantra: Memory augmented networks for multiple trajectory prediction". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[154] S. H. Park, G. Lee, J. Seo, M. Bhat, M. Kang, J. Francis, A. Jadhav, P. P. Liang, and L.-P. Morency. "Diverse and admissible trajectory forecasting through multimodal context understanding". In: *European Conference on Computer Vision (ECCV)*. 2020.

[155] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone. "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data". In: *European Conference on Computer Vision (ECCV)*. 2020.

[156] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[157] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[158] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid. "Vectornet: Encoding hd maps and agent dynamics from vectorized representation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[159] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 2017.

[160] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* 2018.

[161] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* 2020.

[162] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, et al. "Scene Transformer: A unified architecture for predicting future trajectories of multiple agents". In: *International Conference on Learning Representations (ICLR)*. 2021.

[163] A. Cui, S. Casas, K. Wong, S. Suo, and R. Urtasun. "Gorela: Go relative for viewpoint-invariant motion forecasting". In: *International Conference on Robotics and Automation (ICRA)*. 2023.

[164] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp. "Wayformer: Motion forecasting via simple & efficient attention networks". In: *International Conference on Robotics and Automation (ICRA)*. 2023.

[165] S. Shi, L. Jiang, D. Dai, and B. Schiele. "Motion Transformer with Global Intention Localization and Local Movement Refinement". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[166] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, et al. "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction". In: *International Conference on Robotics and Automation (ICRA)*. 2022.

[167] X. Wang, T. Su, F. Da, and X. Yang. "ProphNet: Efficient Agent-Centric Motion Forecasting with Anchor-Informed Proposals". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[168] Z. Zhou, L. Ye, J. Wang, K. Wu, and K. Lu. "Hivt: Hierarchical vector transformer for multi-agent motion prediction". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[169] X. Jia, P. Wu, L. Chen, Y. Liu, H. Li, and J. Yan. "HDGT: Heterogeneous Driving Graph Transformer for Multi-Agent Trajectory Prediction via Scene Encoding". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 2023.

[170] Z. Zhang, A. Liniger, C. Sakaridis, F. Yu, and L. Van Gool. "Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023.

[171] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang. "Query-centric trajectory prediction". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023.

[172] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* 2017.

[173] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. "Junior: The stanford entry in the urban challenge". In: *Journal of Field Robotics* 2008.

[174] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. "Autonomous driving in urban environments: Boss and the urban challenge". In: *Journal of Field Robotics* 2008.

[175] H. Xu, Y. Gao, F. Yu, and T. Darrell. "End-to-end learning of driving models from large-scale video datasets". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[176] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. `https://level5.lyft.com/dataset/`. 2020.

[177] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. "Bdd100k: A diverse driving dataset for heterogeneous multitask learning". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[178] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. "Rainbow: Combining improvements in deep reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[179] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. "Multi-task multi-sensor fusion for 3d object detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[180] D. Wang, C. Devin, Q.-Z. Cai, P. Krähenbühl, and T. Darrell. "Monocular Plan View Networks for Autonomous Driving". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2019.

[181] M. Mueller, A. Dosovitskiy, B. Ghanem, and V. Koltun. "Driving Policy Transfer via Modularity and Abstraction". In: *Conference on Robot Learning (CoRL)*. 2018.

[182] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov. "Offboard 3D Object Detection from Point Cloud Sequences". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[183] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* 2013.

[184] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *International Conference on Learning Representations (ICLR)*. 2016.

[185] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. *Stable Baselines3*. `https://github.com/DLR-RM/stable-baselines3`. 2019.

[186] G. Hinton, O. Vinyals, and J. Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* 2015.

[187] C. team. *CARLA Autonomous Driving Leaderboard*. `https://leaderboard.carla.org/`. Accessed: 2021-02-11. 2020.

[188] S. Hecker, D. Dai, and L. Van Gool. "End-to-end learning of driving models with surround-view cameras and route planners". In: *European Conference on Computer Vision (ECCV)*. 2018.

[189] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. *OpenAI Gym*. 2016.

[190] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. "Learning robust perceptive locomotion for quadrupedal robots in the wild". In: *Science Robotics* 2022.

[191] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr. "Super-human performance in gran turismo sport using deep reinforcement learning". In: *IEEE Robotics and Automation Letters (RA-L)* 2021.

[192] E. Altman. *Constrained Markov Decision Processes*. CRC Press, 1999.

[193] A. Ray, J. Achiam, and D. Amodei. *Benchmarking Safe Exploration in Deep Reinforcement Learning*. 2019.

[194] H. Ma, Y. Guan, S. E. Li, X. Zhang, S. Zheng, and J. Chen. "Feasible actor-critic: Constrained reinforcement learning for ensuring statewise safety". In: *arXiv preprint arXiv:2105.10682* 2021.

[195] B. Peng, Y. Mu, J. Duan, Y. Guan, S. E. Li, and J. Chen. "Separated proportional-integral lagrangian for chance constrained reinforcement learning". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2021.

[196] L. Zhang, L. Shen, L. Yang, S. Chen, X. Wang, B. Yuan, and D. Tao. "Penalized Proximal Policy Optimization for Safe Reinforcement Learning". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2022.

[197] A. Sootla, A. I. Cowen-Rivers, T. Jafferjee, Z. Wang, D. H. Mguni, J. Wang, and H. Ammar. "Sauté RL: Almost surely safe reinforcement learning using state augmentation". In: *International Conference on Machine Learning (ICML)*. 2022.

[198] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. "A general safety framework for learning-based control in uncertain robotic systems". In: *IEEE Transactions on Automatic Control* 2018.

[199] W. Zhao, T. He, and C. Liu. "Model-free safe control for zero-violation reinforcement learning". In: *Conference on Robot Learning (CoRL)*. 2021.

[200] K. P. Wabersich and M. N. Zeilinger. "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems". In: *Automatica* 2021.

[201] A. Wachi and Y. Sui. "Safe reinforcement learning in constrained Markov decision processes". In: *International Conference on Machine Learning (ICML)*. 2020.

[202] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[203] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. "Openai gym". In: *arXiv preprint arXiv:1606.01540* 2016.

[204] N. Koenig and A. Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2004.

[205] A. Raffin. *RL Baselines3 Zoo*. `https://github.com/DLR-RM/rl-baselines3-zoo`. 2020.

[206] A. Raffin, J. Kober, and F. Stulp. "Smooth exploration for robotic reinforcement learning". In: *Conference on Robot Learning (CoRL)*. 2022.

[207] K. Sohn, H. Lee, and X. Yan. "Learning structured output representation using deep conditional generative models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015.

[208] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou. "Multimodal motion prediction with stacked transformers". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[209] R. Girgis, F. Golemo, F. Codevilla, M. Weiss, J. A. D'Souza, S. E. Kahou, F. Heide, and C. Pal. "Latent Variable Sequential Set Transformers for Joint Multi-Agent Motion Prediction". In: *International Conference on Learning Representations (ICLR)*. 2022.

[210] S. V. Albrecht, C. Brewitt, J. Wilhelm, B. Gyevnar, F. Eiras, M. Dobre, and S. Ramamoorthy. "Interpretable goal-based prediction and planning for autonomous driving". In: *International Conference on Robotics and Automation (ICRA)*. 2021.

[211] C. Brewitt, B. Gyevnar, S. Garcin, and S. V. Albrecht. "GRIT: Fast, interpretable, and verifiable goal recognition with learned decision trees for autonomous driving". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2021.

[212] N. Rhinehart, K. M. Kitani, and P. Vernaza. "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting". In: *European Conference on Computer Vision (ECCV)*. 2018.

[213] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool. "Deep Interactive Motion Prediction and Planning: Playing Games with Motion Prediction Models". In: *Learning for Dynamics and Control Conference*. 2022.

[214] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 2021.

[215] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu. "On layer normalization in the transformer architecture". In: *International Conference on Machine Learning (ICML)*. 2020.

[216] X. Mo, Z. Huang, and C. Lv. "Multi-Modal Interactive Agent Trajectory Prediction Using Heterogeneous Edge-Enhanced Graph Attention Network". In: *Workshop on Autonomous Driving, CVPR*. 2021.

[217] D. Wu and Y. Wu. "AIR2 for Interaction Prediction". In: *Workshop on Autonomous Driving, CVPR*. 2021.

[218] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja. "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues". In: *Array* 2021.

[219] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[220] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser. "A review of motion planning for highway autonomous driving". In: *IEEE Transactions on Intelligent Transportation Systems* 2019.

[221] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. "A survey of motion planning and control techniques for self-driving urban vehicles". In: *IEEE Transactions on intelligent vehicles* 2016.

[222] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka. "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions". In: *Transportation Research Part C: Emerging Technologies* 2015.

[223] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun. "Learning lane graph representations for motion forecasting". In: *European Conference on Computer Vision (ECCV)*. 2020.

[224] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde. "Thomas: Trajectory heatmap output with learned multi-agent sampling". In: *International Conference on Learning Representations (ICLR)*. 2022.

[225] X. Mo, Y. Xing, and C. Lv. "Heterogeneous edge-enhanced graph attention network for multi-agent trajectory prediction". In: *arXiv preprint arXiv:2106.07161* 2021.

[226] Q. Sun, X. Huang, J. Gu, B. C. Williams, and H. Zhao. "M2I: From factored marginal trajectory prediction to interactive prediction". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[227] Argoverse. *Argoverse 2: Motion forecasting competition*. `https://eval.ai/web/challenges/challenge-page/1719/leaderboard/4761`. Accessed: 2023-05-10. 2023.

[228] Waymo. *Waymo open dataset interaction prediction challenge 2021*. `https://waymo.com/open/challenges/2021/interaction-prediction/`. Accessed: 2023-05-10. 2021.

[229] W. Zhan, L. Sun, H. Ma, C. Li, X. Jia, M. Tomizuka, et al. *INTERPRET: INTERACTION-Dataset-Based PREdicTion Challenge*. `http://challenge.interaction-dataset.com/leader-board`. Accessed: 2023-05-10. 2021.

[230] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall. "FIERY: future instance prediction in bird's-eye view from surround monocular cameras". In: *International Conference on Computer Vision (ICCV)*. 2021.

[231] S. Casas, C. Gulino, R. Liao, and R. Urtasun. "Spagnn: Spatially-aware graph neural networks for relational behavior forecasting from sensor data". In: *International Conference on Robotics and Automation (ICRA)*. 2020.

[232] A. Cui, S. Casas, A. Sadat, R. Liao, and R. Urtasun. "Lookout: Diverse multi-future prediction and planning for self-driving". In: *International Conference on Computer Vision (ICCV)*. 2021.

[233] W. Zeng, M. Liang, R. Liao, and R. Urtasun. "Lanercnn: Distributed representations for graph-centric motion forecasting". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2021.

[234] J. Huang, V. Sivakumar, M. Mnatsakanyan, and G. Pang. "Improving rotated text detection with rotation region proposal networks". In: *arXiv preprint arXiv:1811.07031* 2018.

[235] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 2019.

[236] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. "End-to-end object detection with transformers". In: *European Conference on Computer Vision (ECCV)*. 2020.

[237] F. Zeng, B. Dong, Y. Zhang, T. Wang, X. Zhang, and Y. Wei. "Motr: End-to-end multiple-object tracking with transformer". In: *European Conference on Computer Vision (ECCV)*. 2022.

[238] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. "Transformer-xl: Attentive language models beyond a fixed-length context". In: *Association for Computational Linguistics (ACL)*. 2019.

[239] P. Shaw, J. Uszkoreit, and A. Vaswani. "Self-attention with relative position representations". In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[240] B. Wang, L. Shang, C. Lioma, X. Jiang, H. Yang, Q. Liu, and J. G. Simonsen. "On position embeddings in bert". In: *International Conference on Learning Representations (ICLR)*. 2021.

[241] K. Wu, H. Peng, M. Chen, J. Fu, and H. Chao. "Rethinking and improving relative position encoding for vision transformer". In: *International Conference on Computer Vision (ICCV)*. 2021.

[242] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[243] S. Konev. "MPA: MultiPath++ Based Architecture for Motion Prediction". In: *arXiv preprint arXiv:2206.10041* 2022.

[244] X. Gao, X. Jia, Y. Li, and H. Xiong. "Dynamic Scenario Representation Learning for Motion Forecasting with Heterogeneous Graph Convolutional Recurrent Networks". In: *IEEE Robotics and Automation Letters (RA-L)* 2023.

[245] Y. Wang, H. Zhou, Z. Zhang, C. Feng, H. Lin, C. Gao, Y. Tang, Z. Zhao, S. Zhang, J. Guo, et al. "TENET: Transformer Encoding Network for Effective Temporal Flow on Motion Prediction". In: *arXiv preprint arXiv:2207.00170* 2022.

[246] C. Zhang, R. Guo, W. Zeng, Y. Xiong, B. Dai, R. Hu, M. Ren, and R. Urtasun. "Rethinking closed-loop training for autonomous driving". In: *European Conference on Computer Vision (ECCV)*. 2022.

[247] C. Zhang, J. Tu, L. Zhang, K. Wong, S. Suo, and R. Urtasun. "Learning Realistic Traffic Agents in Closed-loop". In: *Conference on Robot Learning (CoRL)*. 2023.

[248] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta. "Parting with Misconceptions about Learning-based Vehicle Motion Planning". In: *Conference on Robot Learning (CoRL)*. 2023.

[249] D. Chen, V. Koltun, and P. Krähenbühl. "Learning to drive from a world on rails". In: *International Conference on Computer Vision (ICCV)*. 2021.

[250] D. Chen and P. Krähenbühl. "Learning from all vehicles". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[251] OpenAI. *Video generation models as world simulators*. `https://openai.com/research/video-generation-models-as-world-simulators`. Accessed: 2024-03-13. 2024.

[252] OpenAI. *Sora: Creating video from text*. `https://openai.com/sora`. Accessed: 2024-03-13. 2024.

[253] S. Levine, A. Kumar, G. Tucker, and J. Fu. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems". In: *arXiv preprint arXiv:2005.01643* 2020.

[254] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. "Decision transformer: Reinforcement learning via sequence modeling". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[255] M. Janner, Q. Li, and S. Levine. "Offline Reinforcement Learning as One Big Sequence Modeling Problem". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[256] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. "Planning with Diffusion for Flexible Behavior Synthesis". In: *International Conference on Machine Learning (ICML)*. 2022.

[257] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[258] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* 2023.

[259] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. "Emergent abilities of large language models". In: *arXiv preprint arXiv:2206.07682* 2022.

[260] J. Li, D. Li, S. Savarese, and S. Hoi. "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models". In: *International Conference on Machine Learning (ICML)*. 2023.

[261] Y. Cui, S. Huang, J. Zhong, Z. Liu, Y. Wang, C. Sun, B. Li, X. Wang, and A. Khajepour. "DriveLLM: Charting the path toward full autonomous driving with large language models". In: *IEEE Transactions on Intelligent Vehicles* 2023.

[262] C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, P. Luo, A. Geiger, and H. Li. "Drivelm: Driving with graph visual question answering". In: *arXiv preprint arXiv:2312.14150* 2023.

[263] J. Mao, Y. Qian, H. Zhao, and Y. Wang. "Gpt-driver: Learning to drive with gpt". In: *arXiv preprint arXiv:2310.01415* 2023.

[264] J. Mao, J. Ye, Y. Qian, M. Pavone, and Y. Wang. *A Language Agent for Autonomous Driving*. 2023.

[265] Y. Ma, Y. Cao, J. Sun, M. Pavone, and C. Xiao. *Dolphins: Multimodal Language Model for Driving*. 2023.

[266] L. Chen, O. Sinavski, J. Hünermann, A. Karnsund, A. J. Willmott, D. Birch, D. Maund, and J. Shotton. "Driving with llms: Fusing object-level vector modality for explainable autonomous driving". In: *arXiv preprint arXiv:2310.01957* 2023.

[267] NVIDIA. *NVIDIA Supercharges Autonomous System Development with Omniverse Cloud APIs*. `https://blogs.nvidia.com/blog/omniverse-cloud-apis/?ncid=so-link-489278`. Accessed: 2024-03-19. 2024.