CarPlanner: Consistent Auto-regressive Trajectory Planning for Large-scale Reinforcement Learning in Autonomous Driving

Dongkun Zhang[1,2] Jiaming Liang[2] Ke Guo[2] Sha Lu[1] Qi Wang[2]
Rong Xiong[1,✉] Zhenwei Miao[2,†] Yue Wang[1]
[1]Zhejiang University [2]Cainiao Network
[1] {zhangdongkun, lusha, rxiong, ywang24}@zju.edu.cn
[2] {liangjiaming.ljm, muguo.gk, ruifeng.wq, zhenwei.mzw}@alibaba-inc.com

Abstract

Trajectory planning is vital for autonomous driving, ensuring safe and efficient navigation in complex environments. While recent learning-based methods, particularly reinforcement learning (RL), have shown promise in specific scenarios, RL planners struggle with training inefficiencies and managing large-scale, real-world driving scenarios. In this paper, we introduce ***CarPlanner****, a* ***C****onsistent* ***a****uto-****r****egressive* ***Planner*** *that uses RL to generate multi-modal trajectories. The auto-regressive structure enables efficient large-scale RL training, while the incorporation of consistency ensures stable policy learning by maintaining coherent temporal consistency across time steps. Moreover, CarPlanner employs a generation-selection framework with an expert-guided reward function and an invariant-view module, simplifying RL training and enhancing policy performance. Extensive analysis demonstrates that our proposed RL framework effectively addresses the challenges of training efficiency and performance enhancement, positioning CarPlanner as a promising solution for trajectory planning in autonomous driving. To the best of our knowledge, we are the first to demonstrate that the RL-based planner can surpass both IL- and rule-based state-of-the-arts (SOTAs) on the challenging large-scale real-world dataset nuPlan. Our proposed CarPlanner surpasses RL-, IL-, and rule-based SOTA approaches within this demanding dataset.*

1. Introduction

Trajectory planning [36] is essential in autonomous driving, utilizing outputs from perception and trajectory prediction modules to generate future poses for the ego vehicle. A controller tracks this planned trajectory, producing control commands for closed-loop driving. Recently, learning-based trajectory planning has garnered attention due to its

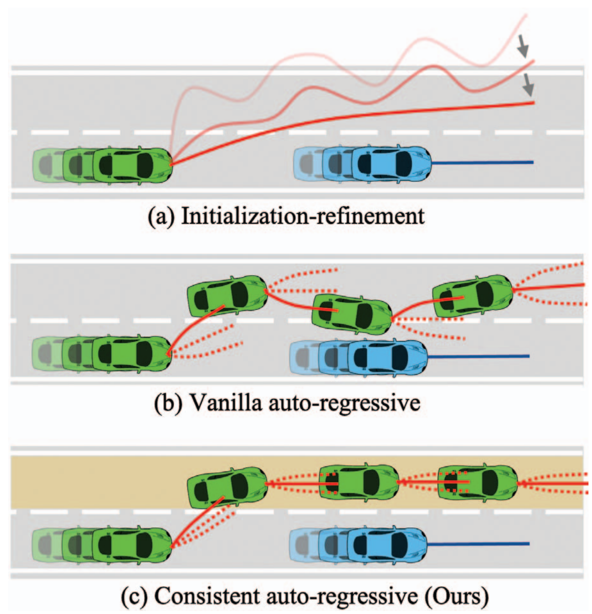

Figure 1. Frameworks for multi-step trajectory generation. (a) Initialization-refinement that generates an initial trajectory and refines it iteratively. (b) Vanilla auto-regressive models that decode subsequent poses sequentially. (c) Our consistent auto-regressive model that integrates time-consistent mode information.

potential to automate algorithm iteration, eliminate tedious rule design, and ensure safety and comfort in diverse real-world scenarios [36].

Most existing researches [11, 16, 29] employ imitation learning (IL) to align planned trajectories with those of human experts. However, this approach suffers from distribution shift [28] and causal confusion [8]. Reinforcement learning (RL) offers a potential solution, addressing these challenges and providing richer supervision through reward functions. Although RL shows effectiveness in domains such as games [34], robotics [19], and language models [25], it still struggles with training inefficiencies and performance issues in the large-scale driving task. To the ex-

[†]Project lead. [✉] Corresponding author.

tent of our knowledge, no RL methods have yet achieved competitive results on large-scale open datasets such as nuPlan [2], which features diverse real-world scenarios.

Thus, this paper aims to tackle two key challenges in RL for trajectory planning: 1) training inefficiency and 2) poor performance. Training inefficiency arises from the fact that RL typically operates in a model-free setting, necessitating an inefficient simulator running on a CPU to repeatedly roll out a policy for data collection. To overcome this challenge, we propose an efficient model-based approach utilizing neural networks as transition models. Our method is optimized for execution on hardware accelerators such as GPUs, rendering our time cost comparable to that of IL-based methods.

To apply RL to solve the trajectory planning problem, we formulate it as a multi-step sequential decision-making task utilizing a Markov Decision Process (MDP). Existing methods that generate the trajectory[†] in multiple steps generally fall into two categories: initialization-refinement [17, 20, 33, 45] and auto-regressive models [27, 32, 41, 46].

The first category, illustrated in Fig. 1 (a), involves generating an initial trajectory estimate and subsequently refining it through iterative applications of RL. However, recent studies, including Gen-Drive [18], suggest that it continues to lag behind SOTA IL and rule-based planners. One notable limitation of this approach is its neglect of the temporal causality inherent in the trajectory planning task. Additionally, the complexity of direct optimization over high-dimensional trajectory space can hinder the performance of RL algorithms. The second category consists of auto-regressive models, shown in Fig. 1 (b), which generate the poses of the ego vehicle recurrently using a single-step policy within a transition model. In this category, ego poses at all time steps are consolidated to form the overall planned trajectory. As taking temporal causality into account, current auto-regressive models allow for interactive behaviors. However, a common limitation is their reliance on auto-regressively random sampling from action distributions to generate multi-modal trajectories. This vanilla auto-regressive procedure may compromise long-term consistency and unnecessarily expand the exploration space in RL, leading to poor performance.

To address the limitations of auto-regressive models, we introduce **CarPlanner**, a **C**onsistent **a**uto-**r**egressive model designed for efficient, large-scale RL-based **Planner** training (see Fig. 1 (c)). The key insight of CarPlanner is its incorporation of consistent mode representation as conditions for the auto-regressive model. Specifically, we leverage a longitudinal-lateral decomposed mode representation, where the longitudinal mode is a scalar that captures average speeds, and the lateral mode encompasses all possible routes derived from the current state of the ego vehicle along with map information. This mode remains constant across time steps, providing stable and consistent guidance during policy sampling.

Furthermore, we propose a universal reward function that suits large-scale and diverse scenarios, eliminating the need for scenario-specific reward designs. This function consists of an expert-guided and task-oriented term. The first term quantifies the displacement error between the ego-planned trajectory and the expert's trajectory, which, along with the consistent mode representation, narrows down the policy's exploration space. The second term incorporates common senses in driving tasks including the avoidance of collision and adherence to the drivable area. Additionally, we introduce an Invariant-View Module (IVM) to supply invariant-view input for policy, with the aim of providing time-agnostic policy input, easing the feature learning and embracing generalization. To achieve this, IVM preprocesses state and lateral mode by transforming agent, map, and route information into the ego's current coordinate and by clipping information that is distant from the ego.

To our knowledge, we are the first to demonstrate that RL-based planner outperforms state-of-the-art (SOTA) IL and rule-based approaches on the challenging large-scale nuPlan dataset. In summary, the key contributions of this paper are highlighted as follows:

- We present **CarPlanner**, a consistent auto-regressive planner that trains an RL policy to generate consistent multi-modal trajectories.
- We introduce an expert-guided universal reward function and IVM to simplify RL training and improve policy generalization, leading to enhanced closed-loop performance.
- We conduct a rigorous analysis on the characteristics of IL and RL training, providing insights into their strengths and limitations, while highlighting the advantages of RL in tackling challenges such as distribution shift and causal confusion.
- Our framework showcases exceptional performance, surpassing all RL-, IL-, and rule-based SOTAs on the nuPlan benchmark. This underscores the potential of RL in navigating complex real-world driving scenarios.

2. Related Work

2.1. Imitation-based Planning

The use of IL to train planners based on human demonstrations has garnered significant interest recently. This approach leverages the driving expertise of experienced drivers who can safely and comfortably navigate a wide range of real-world scenarios, along with the added advantage of easily collectible driving data at scale [2, 9, 15]. Nu-

[†]In this paper, the term "trajectory" refers to the future poses of the ego vehicle or traffic agents. To avoid confusion, we use the term "state (action) sequence" to refer to the "trajectory" in the RL community.

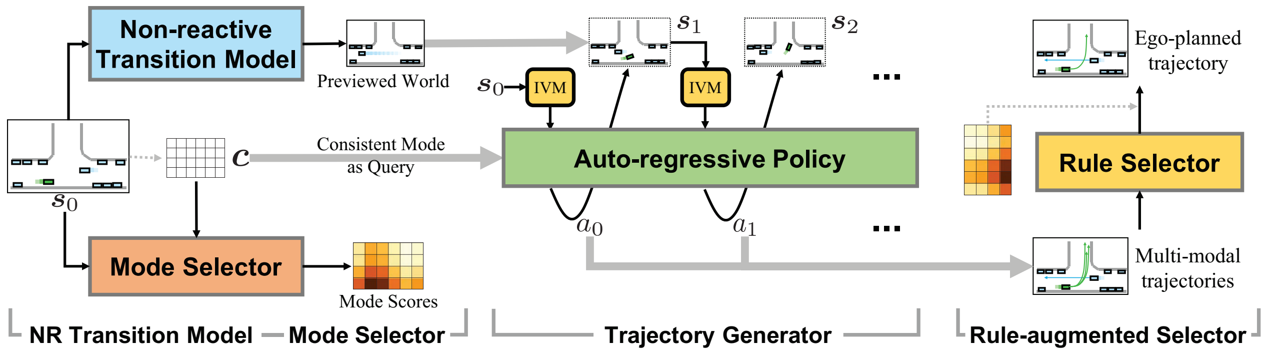

Figure 2. CarPlanner contains four parts. (1) The non-reactive transition model takes initial state $\boldsymbol{s}_0$ as input and predicts the future trajectories of traffic agents. (2) The mode selector outputs scores based on the initial state and the modes $\boldsymbol{c}$. (3) The trajectory generator obeys an auto-regressive structure condition on the consistent mode and produces mode-aligned multi-modal trajectories. (4) The rule-augmented selector compensates the mode scores by safety, comfort, and progress metrics.

merous studies [5, 17, 29] have focused on developing innovative networks to enhance open-loop performance in this domain. However, the ultimate challenge of autonomous driving is achieving closed-loop operation, which is evaluated using driving-oriented metrics such as safety, adherence to traffic rules, comfort, and progress. This reveals a significant gap between the training and testing phases of planners. Moreover, IL is particularly vulnerable to issues such as distribution shift [28] and causal confusion [8]. The first issue results in suboptimal decisions when the system encounters scenarios that are not represented in the training data distribution. The second issue arises when networks inadvertently capture incorrect correlations and develop shortcuts based on input information, primarily due to the reliance on imitation loss from expert demonstrations. Despite efforts in several studies [1, 3, 4, 42] to address these challenges, the gap between training and testing remains substantial.

2.2. RL in Autonomous Driving

In the field of autonomous driving, RL has demonstrated its effectiveness in addressing specific scenarios such as highway driving [22, 39], lane changes [14, 23], and unprotected left turns [22, 40]. Most methods directly learn policies over the control space, which includes throttle, brake, and steering commands. Due to the high frequency of control command execution, the simulation process can be time-consuming, and exploration can be inconsistent [40]. Several works [40, 44] have proposed learning trajectory planners with actions defined as ego-planned trajectories, which temporally extend the exploration space and improve training efficiency. However, a trade-off exists between the trajectory horizon and training performance, as noted in ASAP-RL [40]. Increasing the trajectory horizon results in less reactive behaviors and a reduced amount of data, while a smaller trajectory horizon leads to challenges similar to those encountered in control space. Additionally, these methods typically employ a model-free setting, making them difficult to apply to the complex, diverse real-world scenarios found in large-scale driving datasets. In this paper, we propose adopting a model-based formulation that can facilitate RL training on large-scale datasets. Under this formulation, we aim to overcome the trajectory horizon trade-off by using a transition model, which can provide a preview of the world in which our policy can make multi-step decisions during testing.

3. Method

3.1. Preliminaries

MDP is used to model sequential decision problems, formulated as a tuple $\langle \mathcal{S}, \mathcal{A}, P_\tau, R, \rho_0, \gamma, T \rangle$. $\mathcal{S}$ is the state space. $\mathcal{A}$ is the action space. $P_\tau : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ [†] is the state transition probability. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function and is bounded. $\rho_0 \in \Delta(\mathcal{S})$ is the initial state distribution. T is the time horizon and γ is the discount factor of future rewards. The state-action sequence is defined as $\tau = (\boldsymbol{s}_0, a_0, \boldsymbol{s}_1, a_1, \ldots, \boldsymbol{s}_T)$, where $\boldsymbol{s}_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ are the state and action at time step t. The objective of RL is to maximize the expected return:

$$\max_{\pi} \mathbb{E}_{\boldsymbol{s}_t \sim P_\tau, a_t \sim \pi} [\sum_{t=0}^{T} \gamma^t R(\boldsymbol{s}_t, a_t)]. \tag{1}$$

Vectorized state representation. State $\boldsymbol{s}_t$ contains map and agent information in vectorized representation [10]. Map information m includes the road network, traffic lights, etc, which are represented by polylines and polygons. Agent information includes the current and past poses of ego vehicle and other traffic agents, which are represented by polylines. The index of ego vehicle is 0 and the indices of traffic agents range from 1 to N. For each agent i, its history is denoted as $s^i_{t-H:t}, i \in \{0, 1, \ldots, N\}$, where H is the history time horizon.

[†] $\Delta(\mathcal{X})$ denotes the set of probability distribution over set $\mathcal{X}$.

3.2. Problem Formulation

We model the trajectory planning task as a sequential decision process and decouple the auto-regressive models into policy and transition models. The key to connect trajectory planning and auto-regressive models is to define the action as the next pose of ego vehicle, i.e., $a_t = s^0_{t+1}$. Therefore, after forwarding the auto-regressive model, the decoded pose is collected to be the ego-planned trajectory. Specifically, we can reduce the state-action sequence to the state sequence under this definition and vectorized representation:

$$
\begin{aligned}
&P(\boldsymbol{s}_0, a_0, \boldsymbol{s}_1, a_1, \ldots, \boldsymbol{s}_T) \\
&= P(m, s^{0:N}_{-H:0}, s^0_1, m, s^{0:N}_{1-H:1}, s^0_2, \ldots, m, s^{0:N}_{T-H:T}) \\
&= P(m, s^{0:N}_{-H:0}, m, s^{0:N}_{1-H:1}, \ldots, m, s^{0:N}_{T-H:T}) \\
&= P(\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_T).
\end{aligned}
\tag{2}
$$

The state sequence can be further formulated in an auto-regressive fashion and decomposed into policy and transition model:

$$
\begin{aligned}
P(\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_T) &= \rho_0(\boldsymbol{s}_0) \prod_{t=0}^{T-1} P(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t) \\
&= \rho_0(\boldsymbol{s}_0) \prod_{t=0}^{T-1} P(s^0_{t+1}, s^{1:N}_{t+1} | \boldsymbol{s}_t) \\
&= \rho_0(\boldsymbol{s}_0) \prod_{t=0}^{T-1} \underbrace{\pi(a_t | \boldsymbol{s}_t)}_{\text{Policy}} \underbrace{P_\tau(s^{1:N}_{t+1} | \boldsymbol{s}_t)}_{\text{Transition Model}}.
\end{aligned}
\tag{3}
$$

From Eq. (3), we can clearly identify the inherent problem associated with the typical auto-regressive approach: inconsistent behaviors across time steps arise from the policy distribution, which depends on random sampling from the action distribution.

To solve the above problem, we introduce consistent mode information $\boldsymbol{c}$ that remains unchanged across time steps into the auto-regressive fashion:

$$
\begin{aligned}
&P(\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_T) = \int_{\boldsymbol{c}} P(\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_T, \boldsymbol{c}) d\boldsymbol{c} \\
&= \rho_0(\boldsymbol{s}_0) \int_{\boldsymbol{c}} P(\boldsymbol{c} | \boldsymbol{s}_0) P(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T | \boldsymbol{c}) d\boldsymbol{c} \\
&= \rho_0(\boldsymbol{s}_0) \prod_{t=0}^{T-1} \underbrace{P_\tau(s^{1:N}_{t+1} | \boldsymbol{s}_t)}_{\text{Transition Model}} \int_{\boldsymbol{c}} \underbrace{P(\boldsymbol{c} | \boldsymbol{s}_0)}_{\text{Mode Selector}} \prod_{t=0}^{T-1} \underbrace{\pi(a_t | \boldsymbol{s}_t, \boldsymbol{c})}_{\text{Policy}} d\boldsymbol{c}.
\end{aligned}
\tag{4}
$$

Since we focus on the ego trajectory planning, the consistent mode $\boldsymbol{c}$ does not impact transition model.

This consistent auto-regressive formulation defined in Eq. (4) reveals a generation-selection framework where the mode selector scores each mode based on the initial state s_0 and the trajectory generator generates multi-modal trajectories via sampling from the mode-conditioned policy.

Non-reactive transition model. The transition model formulated in Eq. (4) needs to be employed in every time step since it produces the poses of traffic agents at time step $t+1$ based on current state $\boldsymbol{s}_t$. In practice, this process is time-consuming and we do not observe a performance improvement by using this transition model, therefore, we use trajectory predictors $P(s^{1:N}_{1:T} | \boldsymbol{s}_0)$ as non-reactive transition model that produces all future poses of traffic agents in one shot given initial state $\boldsymbol{s}_0$.

3.3. Planner Architecture

The framework of our proposed **CarPlanner** is illustrated in Fig. 2, comprising four key components: 1) the non-reactive transition model, 2) the mode selector, 3) the trajectory generator, and 4) the rule-augmented selector.

Our planner operates within a generation-selection framework. Given an initial state s_0 and all possible N_{mode} modes, the trajectory selector evaluates and assigns scores to each mode. The trajectory generator then produces N_{mode} trajectories that correspond to their respective modes. For trajectory generator, the initial state s_0 is replicated N_{mode} times, each associated with one of the N_{mode} modes, effectively creating N_{mode} parallel worlds. The policy is executed within these previewed worlds. During the policy rollout, a trajectory predictor acts as the state transition model, generating future poses of traffic agents across all time horizons.

3.3.1. Non-reactive Transition Model

This module takes the initial state s_0 as input and outputs the future trajectories of traffic agents. The initial state is processed by agent and map encoders, followed by a self-attention Transformer encoder [38] to fuse the agent and map features. The agent features are then decoded into future trajectories.

Agent and map encoders. The state s_0 contains both map and agent information. The map information m consists of $N_{m,1}$ polylines and $N_{m,2}$ polygons. The polylines describe lane centers and lane boundaries, with each polyline containing $3N_p$ points, where 3 corresponds to the lane center, the left boundary, and the right boundary. Each point is with dimension $D_m = 9$ and includes the following attributes: x, y, heading, speed limit, and category. When concatenated, the points of the left and right boundaries together with the center point yield a dimension of $N_{m,1} \times N_p \times 3D_m$. We leverage a PointNet [26] to extract features from the points of each polyline, resulting in a dimensionality of $N_{m,1} \times D$, where D represents the feature dimension. The polygons represent intersections, crosswalks, stop lines, etc, with each polygon containing N_p points. We utilize another PointNet to extract features from the points of each polygon, producing a dimension of $N_{m,2} \times D$. We then concatenate the features from both polylines and polygons to form the overall map features, resulting in a dimension of $N_m \times D$. The agent information A consists of N agents, where each agent maintains poses for the past H time steps. Each pose is with dimension $D_a = 10$ and includes the following attributes: x, y, heading, velocity, bounding box, time step, and category. Consequently, the agent information has a dimension of $N \times H \times D_a$. We apply another PointNet to extract features from the poses of each agent, yielding an agent feature dimension of $N \times D$.

3.3.2. Mode Selector

This module takes s_0 and longitudinal-lateral decomposed mode information as input and outputs the probability of each mode. The number of modes $N_{\text{mode}} = N_{\text{lat}} N_{\text{lon}}$.

Route-speed decomposed mode. To capture the longitudinal behaviors, we generate N_{lon} modes that represent the average speed of the trajectory associated with each mode. Each longitudinal mode $c_{\text{lon},j}$ is defined as a scalar value of $\frac{j}{N_{\text{lon}}}$, repeated across a dimension D. As a result, the dimensionality of the longitudinal modes is $N_{\text{lon}} \times D$. For lateral behaviors, we identify N_{lat} possible routes from the map using a graph search algorithm. These routes correspond to the lanes available for the ego vehicle. The dimensionality of these routes is $N_{\text{lat}} \times N_r \times D_m$. We employ another PointNet to aggregate the features of the N_r points along each route, producing a lateral mode with a dimension of $N_{\text{lat}} \times D$. To create a comprehensive mode representation $\boldsymbol{c}$, we combine the lateral and longitudinal modes, resulting in a combined dimension of $N_{\text{lat}} \times N_{\text{lon}} \times 2D$. To align this mode information with other feature dimensions, we pass it through a linear layer, mapping it back to $N_{\text{lat}} \times N_{\text{lon}} \times D$.

Query-based Transformer decoder. This decoder is employed to fuse the mode features with map and agent features derived from s_0. In this framework, the mode serves as the query, while the map and agent information act as the keys and values. The updated mode features are decoded through a multi-layer perceptron (MLP) to yield the scores for each mode, which are subsequently normalized using the softmax operator.

3.3.3. Trajectory Generator

This module operates in an auto-regressive manner, recurrently decoding the next pose of the ego vehicle a_t, given the current state $\boldsymbol{s}_t$, and consistent mode information $\boldsymbol{c}$.

Invariant-view module (IVM). Before feeding the mode and state into the network, we preprocess them to eliminate time information. For the map and agent information in state $\boldsymbol{s}_t$, we select the K-nearest neighbors (KNN) to the ego current pose and only feed these into the policy. K is set to the half of map and agent elements respectively. Regarding the routes that capture lateral behaviors, we filter out the segments where the point closest to the current pose of the ego vehicle is the starting point, retaining K_r points. In this case, K_r is set to a quarter of N_r points in one route. Finally, we transform the routes, agent, and map poses into the coordinate frame of the ego vehicle at the current time step t. We subtract the historical time steps $t - H : t$ from the current time step t, yielding time steps in range $-H : 0$.

Query-based Transformer decoder. We employ the same backbone network architecture as the mode selector, but with different query dimensions. Due to the IVM and the fact that different modes yield distinct states, the map and agent information cannot be shared among modes. As a result, we fuse information for each individual mode. Specifically, the query dimension is $1 \times D$, while the dimensions of the keys and values are $(N + N_m) \times D$. The output feature dimension remains $1 \times D$. Note that Transformer decoder can process information from multiple modes in parallel, eliminating the need to handle each mode sequentially.

Policy output. The mode feature is processed by two distinct heads: a policy head and a value head. Each head comprises its own MLP to produce the parameters for the action distribution and the corresponding value estimate. We employ a Gaussian distribution to model the action distribution, where actions are sampled from this distribution during training. In contrast, during inference, we utilize the mean of the distribution to determine the actions.

3.3.4. Rule-augmented Selector

This module is only utilized during inference and takes as input the initial state s_0, the multi-modal ego-planned trajectories, and the predicted future trajectories of agents. It calculates driving-oriented metrics such as safety, progress, comfort. A comprehensive score is obtained by the weighted sum of rule-based scores and the mode scores provided by the mode selector. The ego-planned trajectory with the highest score is selected as the output of the planner.

3.4. Training

We first train the non-reactive transition model and freeze the weights during the training of the mode selector and trajectory generator. Instead of feeding all modes to the generator, we apply a winner-takes-all strategy, wherein a positive mode is assigned based on the ego ground-truth trajectory and serves as a condition for the trajectory generator.

Mode assignment. For the lateral mode, we assign the route closest to the endpoint of ego ground-truth trajectory as the positive lateral mode. For the longitudinal mode, we partition the longitudinal space into N_{lon} intervals and assign the interval containing the endpoint of the ground-truth trajectory as the positive longitudinal mode.

Reward function. To handle diverse scenarios, we use the negative displacement error (DE) between the ego future pose and the ground truth as a universal reward. We also introduce additional terms to improve trajectory quality: collision rate and drivable area compliance. If the future pose collides or falls outside the drivable area, the reward is set to -1; otherwise, it is 0.

Mode dropout. In some cases, there are no available routes for ego to follow. However, since routes serve as queries in Transformer, the absence of a route can lead to unstable or hazardous outputs. To mitigate this issue, we implement a mode dropout module during training that randomly masks routes to prevent over-reliance on this information.

Loss function. For the selector, we use cross-entropy loss that is the negative log-likelihood of the positive mode and a side task that regresses the ego ground-truth trajectory. For the generator, we use PPO [31] loss that consists of three parts: policy improvement, value estimation, and entropy. Full description can be found in supplementary.

4. Experiments

4.1. Experimental Setup

Dataset and simulator. We use nuPlan [2], a large-scale closed-loop platform for studying trajectory planning in autonomous driving, to evaluate the efficacy of our method. The nuPlan dataset contains driving log data over 1,500 hours collected by human expert drivers across 4 diverse cities. It includes complex, diverse scenarios such as lane follow and change, left and right turn, traversing intersections and bus stops, roundabouts, interaction with pedestrians, etc. As a closed-loop platform, nuPlan provides a simulator that uses scenarios from the dataset as initialization. During the simulation, traffic agents are taken over by log-replay (non-reactive) or an IDM [37] policy (reactive). The ego vehicle is taken over by user-provided planners. The simulator lasts for 15 seconds and runs at 10 Hz. At each timestamp, the simulator queries the planner to plan a trajectory, which is tracked by an LQR controller to generate control commands to drive the ego vehicle.

Benchmarks and metrics. We use two benchmarks: Test14-Random and Reduced-Val14 for comparing with other methods and analyzing the design choices within our method. The Test14-Random provided by PlanTF [4] contains 261 scenarios. The Reduced-Val14 provided by PDM [7] contains 318 scenarios.

We use the closed-loop score (CLS) provided by the official nuPlan devkit[†] to assess the performance of all methods. The CLS score comprehends different aspects such as safety (S-CR, S-TTC), drivable area compliance (S-Area), progress (S-PR), comfort, etc. Based on the different behavior types of traffic agents, CLS is detailed into CLS-NR (non-reactive) and CLS-R (reactive).

Implementation details. We follow PDM [7] to construct our training and validation splits. The size of the training set is 176,218 where all available scenario types are used, with a number of 4,000 scenarios per type. The size of the validation set is 1,118 where 100 scenarios with 14 types are selected. We train all models with 50 epochs in 2 NVIDIA 3090 GPUs. The batch size is 64 per GPU. We use AdamW optimizer with an initial learning rate of 1e-4 and reduce the learning rate when the validation loss stops decreasing with a patience of 0 and decrease factor of 0.3. For RL training, we set the discount $\gamma = 0.1$ and the GAE parameter $\lambda = 0.9$. The weights of value, policy, and entropy loss are set to 3, 100, and 0.001, respectively. The number of longitudinal modes is set to 12 and a maximum number of lateral modes are set to 5.

[†] https://github.com/motional/nuplan-devkit

Type	Planners	CLS-NR	CLS-R
Rule	IDM [37]	70.39	72.42
	PDM-Closed [7]	90.05	**91.64**
IL	RasterModel [2]	69.66	67.54
	UrbanDriver [29]	63.27	61.02
	GC-PGP [12]	55.99	51.39
	PDM-Open [7]	52.80	57.23
	GameFormer [17]	80.80	79.31
	PlanTF [4]	86.48	80.59
	PEP [42]	91.45	89.74
	PLUTO [3]	<u>91.92</u>	90.03
RL	CarPlanner (Ours)	**94.07**	<u>91.1</u>

Table 1. Comparison with SOTAs in Test14-Random. Based on the type of trajectory generator, all methods are categorized into Rule, IL, and RL. The best result is in **bold** and the second best result is <u>underlined</u>.

Type	Planners	CLS-NR	S-CR	S-PR
Rule	PDM-Closed [7]	<u>91.21</u>	**97.01**	<u>92.68</u>
IL	GameFormer [17]	83.76	94.73	88.12
	PlanTF [4]	83.66	94.02	92.67
	Gen-Drive (Pretrain) [18]	85.12	93.65	86.64
RL	Gen-Drive (Finetune) [18]	87.53	95.72	89.94
	CarPlanner (Ours)	**91.45**	<u>96.38</u>	**95.37**

Table 2. Comparison with SOTAs in Reduced-Val14 with non-reactive traffic agents.

4.2. Comparison with SOTAs

SOTAs. We categorize the methods into Rule, IL, and RL based on the type of trajectory generator. (1) PDM [7] wins the nuPlan challenge 2023, its IL-based and rule-based variants are denoted as PDM-Open and PDM-Closed, respectively. PDM-Closed follows the generation-selection framework where IDM is used to generate multiple candidate trajectories and rule-based selector considering safety, progress, and comfort is used to select the best trajectory. (2) PLUTO [3] also obeys the generation-selection framework and uses contrastive IL to incorporate various data augmentation techniques and trains the generator. (3) Gen-Drive [18] is a concurrent work that follows a pretrain-finetune pipeline where IL is used to pretrain a diffusion-based planner and RL is used to finetune the denoising process based on a reward model trained by AI preference.

Results. We compare our method with SOTAs in Test14-Random and Reduced-Val14 benchmark as shown in Tab. 1 and Tab. 2. Overall, our CarPlanner demonstrates superior performance, particularly in non-reactive environments.

In the non-reactive setting, our method achieves the highest scores across all metrics, with an improvement of 4.02 and 2.15 compared to PDM-Closed and PLUTO, establishing the potential of RL and the superior performance

Design Choices				Closed-loop metrics (↑)					Open-loop losses (↓)	
Reward DE	Reward Quality	Coord Trans	KNN	CLS-NR	S-CR	S-Area	S-PR	S-Comfort	Loss Selector	Loss Generator
✗	✓	✓	✓	31.79	95.74	98.45	33.10	48.84	1.03	**30.3**
✓	✗	✓	✓	90.44	97.49	96.91	93.33	90.73	**0.99**	1221.6
✓	✓	✗	✓	90.78	96.92	98.46	91.37	**94.23**	1.00	2130.7
✓	✓	✓	✗	92.73	98.07	98.46	94.69	93.44	1.03	2083.6
✓	✓	✓	✓	**94.07**	**99.22**	**99.22**	**95.06**	91.09	1.03	1624.5

Table 3. Ablation studies on the design choices in RL training. Results are in Test14-random non-reactive benchmark.

of our proposed framework. Moreover, CarPlanner reveals substantial improvement in the progress metric S-PR compared to PDM-Closed in Tab. 2 and comparable collision metric S-CR, indicating the ability of our method to improving driving efficiency while maintaining safe driving. Importantly, we do not apply any techniques commonly used in IL such as data augmentation [3, 4] and ego-history masking [11], underscoring the intrinsic capability of our approach to solving the closed-loop task.

In the reactive setting, while our method performs well, it falls slightly short of PDM-Closed. This discrepancy arises because our model was trained exclusively in non-reactive settings and has not interacted with the IDM policy used by reactive settings; as a result, our model is less robust to disturbances generated by reactive agents during testing.

4.3. Ablation Studies

We investigate the effects of different design choices in RL training. The results are shown in Tab. 3.

Influence of reward items. When using the quality reward only, the planner tends to generate static trajectories and achieves a low progress metric. This occurs because the ego vehicle begins in a safe, drivable state, but moving forward is at risk of collisions or leaving the drivable area. On the other hand, when the quality reward is incorporated alongside the DE reward, it leads to significant improvements in closed-loop metrics compared to using the DE reward alone. For instance, the S-CR metric rises from 97.49 to 99.22, and the S-Area metric rises from 96.91 to 99.22. These improvements indicate that the quality reward encourages safe and comfortable behaviors.

Effectiveness of IVM. The results show that the coordinate transformation and KNN techniques in IVM notably improve closed-loop metrics and generator loss. For instance, with the coordinate transformation technique, the overall closed-loop score increases from 90.78 to 94.07, and S-PR rises from 91.37 to 95.06. These improvements are attributed to the enhanced accuracy of value estimation in RL, leading to generalized driving in closed-loop.

4.4. Extention to IL

In addition to designing for RL training, we also extend the CarPlanner to incorporate IL. We conduct rigorous analysis to compare the effects of various design choices in IL and RL training, as summarized in Tab. 4. Our findings indicate that while mode dropout and selector side task contribute to both IL and RL training, ego-history dropout and backbone sharing, often effective in IL, are less suitable for RL.

Ego-history dropout. Previous works [1, 3, 4, 11] suggest that planners trained via IL may rely too heavily on past poses and neglect environmental state information. To counter this, we combine techniques from ChauffeurNet [1] and PlanTF [4] into an ego-history dropout module, randomly masking ego history poses and current velocity to alleviate the causal confusion issue.

Our experiments confirm that ego-history dropout benifits IL training, as it improves performance across closed-loop metrics like S-CR and S-Area. However, in RL training, we observe a negative impact on advantage estimation due to ego-history dropout, which significantly affects the value part of generator loss, leading to closed-loop performance degradation. This suggests that RL training naturally addresses the causal confusion problem inherent in IL by uncovering causal relationships that align with the reward signal, which explicitly encodes task-oriented preferences. This capability highlights the potential of RL to push the boundaries of learning-based planning.

Backbone sharing. This choice, often used in IL-based multi-modal planners, promotes feature sharing across tasks to improve generalization. While backbone sharing helps IL by balancing losses across trajectory generator and selector, we find it adversely affects RL training. Specifically, backbone sharing leads to higher losses for both the trajectory generator and selector in RL, indicating that gradients from each task interfere. The divergent objectives in RL for trajectory generation and selection tasks seem to conflict, reducing overall policy performance. Consequently, we avoid backbone sharing in our RL framework to maintain task-specific gradient flow and improve policy quality.

4.5. Qualitative Results

We provide qualitative results as shown in Fig. 3. In this scenario, ego vehicle is required to execute a right turn while navigating around pedestrians. In this case, Our method shows a smooth, efficient performance. From $t_{\text{sim}} = 0s$ to $t_{\text{sim}} = 9s$, all methods wait for the pedestrians to cross the road. At $t_{\text{sim}} = 10s$, an unexpected pedestrian goes back and prepares to re-cross the road. PDM-Closed is

Loss Type	Mode Dropout	Selector Side Task	Ego-history Dropout	Backbone Sharing	CLS-NR	S-CR	S-Area	S-PR	S-Comfort	Loss Selector	Loss Generator
	Design Choices				Closed-loop metrics (↑)					Open-loop losses (↓)	
IL	✗	✗	✗	✗	90.82	97.29	98.45	92.15	94.57	1.04	**147.5**
	✓	✗	✗	✗	91.21	96.54	98.46	91.44	**96.92**	1.07	153.0
	✓	✓	✗	✗	91.51	96.91	98.46	**95.30**	96.91	1.04	162.3
	✓	✓	✓	✗	92.72	98.06	98.84	94.88	95.35	1.04	167.5
	✓	✓	✓	✓	93.41	98.85	98.85	93.87	96.15	1.04	174.3
RL	✗	✗	✗	✗	91.67	98.84	98.84	91.69	90.73	1.04	1812.6
	✓	✗	✗	✗	93.46	98.07	**99.61**	94.26	92.28	1.09	2254.6
	✓	✓	✗	✗	**94.07**	**99.22**	99.22	95.06	91.09	**1.03**	1624.5
	✓	✓	✓	✗	89.51	97.27	98.44	90.93	83.20	1.05	5424.3
	✓	✓	✓	✓	88.66	95.54	98.84	92.82	86.05	1.21	1928.1

Table 4. Effect of different components on IL and RL loss using our CarPlanner. Results are in Test14-random non-reactive benchmark.

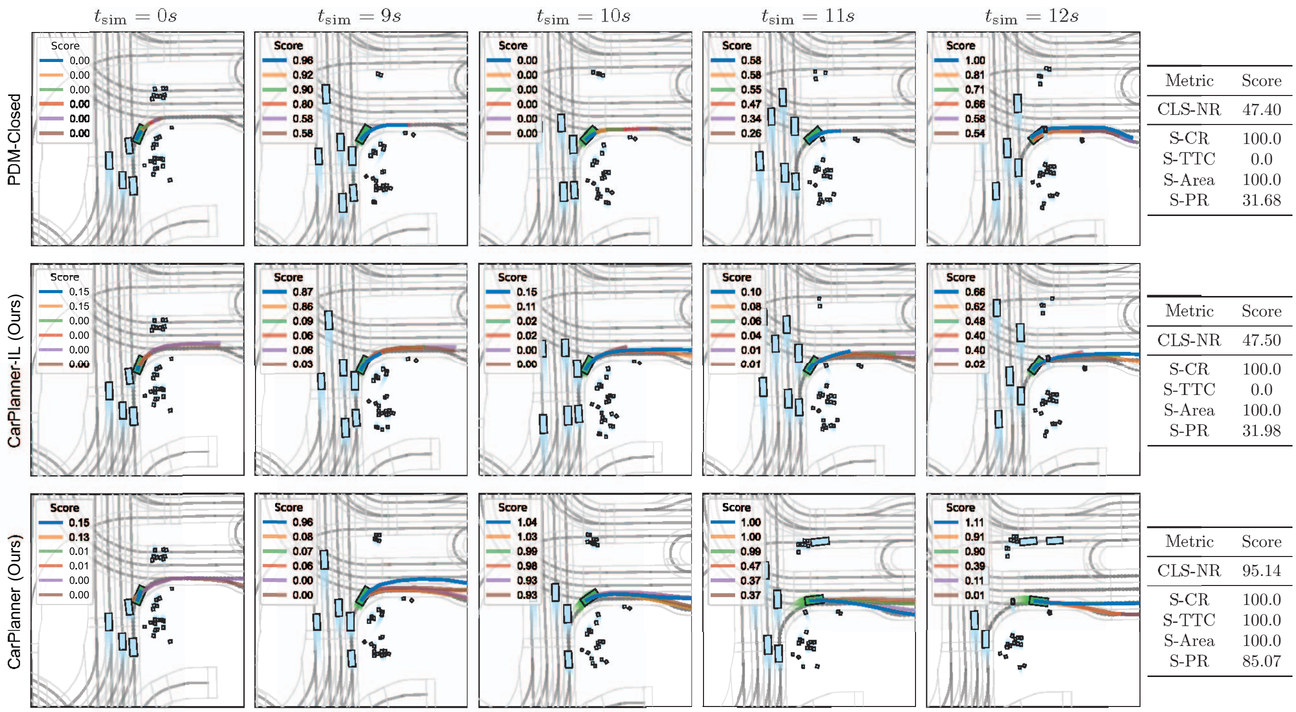

Figure 3. Qualitative comparison of PDM-Closed and our method in non-reactive environments. The scenario is annotated as `waiting_for_pedestrian_to_cross`. In each frame shot, ego vehicle is marked as green. Traffic agents are marked as sky blue. Lineplot with blue is the ego planned trajectory.

unaware of this situation and takes an emergency stop, but it still intersects with this pedestrian. In contrast, our IL variant displays an awareness of the pedestrian's movements and consequently conducts a braking maneuver. However, it still remains close to the pedestrian. Our RL method avoids this hazard by starting up early up to $t_{\text{sim}} = 9s$ and achieves the highest progress and safety metrics.

5. Conclusion

In this paper, we introduce CarPlanner, a consistent auto-regressive planner aiming at large-scale RL training. Thanks to the proposed framework, we train an RL-based planner that outperforms existing RL-, IL-, and rule-based SOTAs. Furthermore, we provide analysis indicating the characteristics of IL and RL, highlighting the potential of RL to take a further step toward learning-based planning.

Limitations and future work. RL needs delicate design and is prone to input representation. RL can overfit its training environment and suffer from performance drop in unseen environments [21]. Our method leverages expert-aided reward design to guide exploration. However, this approach may constrain the full potential of RL, as it inherently relies on expert demonstrations and may hinder the discovery of solutions that surpass human expertise. Future work aims to develop robust RL algorithms capable of overcoming these limitations, enabling autonomous exploration and generalization across diverse environments.

6. Acknowledgement

Many thanks to Jingke Wang for helpful discussions and all reviewers for improving the paper. This work was supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. LD24F030001, and by the National Nature Science Foundation of China under Grant 62373322.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Proc. of Robotics: Science and Systems*, 2019. 3, 7

[2] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021. 2, 6, 12

[3] Jie Cheng, Yingbing Chen, and Qifeng Chen. Pluto: Pushing the limit of imitation learning-based planning for autonomous driving. *arXiv preprint arXiv:2404.14327*, 2024. 3, 6, 7, 12, 13

[4] Jie Cheng, Yingbing Chen, Xiaodong Mei, Bowen Yang, Bo Li, and Ming Liu. Rethinking imitation-based planners for autonomous driving. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, pages 14123–14130. IEEE, 2024. 3, 6, 7

[5] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 2022. 3

[6] Ignasi Clavera, Yao Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. In *Proc. of the Int. Conf. on Learning Representations*, 2020. 15

[7] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Proc. of the Conf. on Robot Learning*, 2023. 6

[8] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Proc. of the Advances in Neural Information Processing Systems*, 32, 2019. 1, 3

[9] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 9710–9719, 2021. 2, 12

[10] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 3

[11] Ke Guo, Wei Jing, Junbo Chen, and Jia Pan. CCIL: Context-conditioned imitation learning for urban driving. In *Proc. of Robotics: Science and Systems*, 2023. 1, 7

[12] Marcel Hallgarten, Martin Stoll, and Andreas Zell. From prediction to planning with goal conditioned lane graph traversals. *arXiv preprint arXiv:2302.07753*, 2023. 6

[13] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proc. of the Intl. Conf. on Machine Learning*, pages 8387–8406. PMLR, 2022. 15

[14] Xiangkun He, Haohan Yang, Zhongxu Hu, and Chen Lv. Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach. *IEEE Transactions on Intelligent Vehicles*, 8(1):184–193, 2022. 3

[15] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. In *Proc. of the Conf. on Robot Learning*, pages 409–418, 2021. 2

[16] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023. 1

[17] Zhiyu Huang, Haochen Liu, and Chen Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 3903–3913, 2023. 2, 3, 6, 13

[18] Zhiyu Huang, Xinshuo Weng, Maximilian Igl, Yuxiao Chen, Yulong Cao, Boris Ivanovic, Marco Pavone, and Chen Lv. Gen-drive: Enhancing diffusion generative driving policies with reward modeling and reinforcement learning fine-tuning. *arXiv preprint arXiv:2410.05582*, 2024. 2, 6, 12

[19] Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *Intl. Journal of Robotics Research*, 40(4-5):698–721, 2021. 1

[20] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023. 2

[21] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023. 8

[22] Edouard Leurent and Jean Mercat. Social attention for autonomous decision-making in dense traffic. *arXiv preprint arXiv:1911.12250*, 2019. 3

[23] Guofa Li, Yifan Yang, Shen Li, Xingda Qu, Nengchao Lyu, and Shengbo Eben Li. Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness. *Transportation research part C: emerging technologies*, 134:103452, 2022. 3

[24] Quanyi Li, Zhenghao Mark Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *Proc. of the Advances in Neural Information Processing Systems*, 36:3894–3920, 2023. 12

[25] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Proc. of the Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022. 1

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 4

[27] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. Precog: Prediction conditioned on goals in visual multi-agent settings. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 2821–2830, 2019. 2

[28] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011. 1, 3

[29] Oliver Scheel, Luca Bergamini, Maciej Wolczyk, Błażej Osiński, and Peter Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Proc. of the Conf. on Robot Learning*, pages 718–728, 2022. 1, 3, 6

[30] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 11, 12

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 6, 11

[32] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision*, pages 8579–8590, 2023. 2

[33] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Proc. of the Advances in Neural Information Processing Systems*, 35:6531–6543, 2022. 2

[34] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 1

[35] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021. 13

[36] Ardi Tampuu, Tambet Matiisen, Maksym Semikin, Dmytro Fishman, and Naveed Muhammad. A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4): 1364–1384, 2020. 1

[37] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000. 6

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proc. of the Advances in Neural Information Processing Systems*, 30, 2017. 4

[39] Huanjie Wang, Shihua Yuan, Mengyu Guo, Xueyuan Li, and Wei Lan. A deep reinforcement learning-based approach for autonomous driving in highway on-ramp merge. *Proceedings of the Institution of Mechanical engineers, Part D: Journal of Automobile engineering*, 235(10-11):2726–2739, 2021. 3

[40] Letian Wang, Jie Liu, Hao Shao, Wenshuo Wang, Ruobing Chen, Yu Liu, and Steven L Waslander. Efficient Reinforcement Learning for Autonomous Driving with Parameterized Skills and Priors. In *Proc. of Robotics: Science and Systems*, Daegu, Republic of Korea, 2023. 3

[41] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time simulation via next-token prediction. *arXiv preprint arXiv:2405.15677*, 2024. 2

[42] Dongkun Zhang, Jiaming Liang, Sha Lu, Ke Guo, Qi Wang, Rong Xiong, Zhenwei Miao, and Yue Wang. Pep: Policy-embedded trajectory planning for autonomous driving. *IEEE Robotics and Automation Letters*, 2024. 3, 6

[43] Zhejun Zhang, Alexander Liniger, Christos Sakaridis, Fisher Yu, and Luc V Gool. Real-time motion prediction via heterogeneous polyline transformer with relative pose encoding. *Proc. of the Advances in Neural Information Processing Systems*, 36, 2024. 13

[44] Tong Zhou, Letian Wang, Ruobing Chen, Wenshuo Wang, and Yu Liu. Accelerating reinforcement learning for autonomous driving using task-agnostic and ego-centric motion skills. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 11289–11296. IEEE, 2023. 3

[45] Yang Zhou, Hao Shao, Letian Wang, Steven L Waslander, Hongsheng Li, and Yu Liu. Smartrefine: A scenario-adaptive refinement framework for efficient motion prediction. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, pages 15281–15290, 2024. 2

[46] Zikang Zhou, Haibo Hu, Xinhong Chen, Jianping Wang, Nan Guan, Kui Wu, Yung-Hui Li, Yu-Kai Huang, and Chun Jason Xue. Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. *arXiv preprint arXiv:2405.17372*, 2024. 2