Rethinking Closed-loop Training for Autonomous Driving

Chris Zhang[⋆1,2], Runsheng Guo[⋆†3], Wenyuan Zeng[⋆1,2], Yuwen Xiong[1,2], Binbin Dai[1], Rui Hu[1], Mengye Ren[†4], and Raquel Urtasun[1,2]

[1] Waabi [2] University of Toronto
[3] University of Waterloo [4] New York University
{czhang,wzeng,yxiong,bdai,rhu,urtasun}@waabi.ai
r9guo@uwaterloo.ca@uwaterloo.ca mengye@nyu.edu

Abstract. Recent advances in high-fidelity simulators [22,82,44] have enabled closed-loop training of autonomous driving agents, potentially solving the distribution shift in training v.s. deployment and allowing training to be scaled both safely and cheaply. However, there is a lack of understanding of how to build effective training benchmarks for closed-loop training. In this work, we present the first empirical study which analyzes the effects of different training benchmark designs on the success of learning agents, such as how to design traffic scenarios and scale training environments. Furthermore, we show that many popular RL algorithms cannot achieve satisfactory performance in the context of autonomous driving, as they lack long-term planning and take an extremely long time to train. To address these issues, we propose trajectory value learning (TRAVL), an RL-based driving agent that performs planning with multistep look-ahead and exploits cheaply generated imagined data for efficient learning. Our experiments show that TRAVL can learn much faster and produce safer maneuvers compared to all the baselines.

Keywords: Closed-loop Learning, Autonomous Driving, RL

1 Introduction

Self-driving vehicles require complex decision-making processes that guarantee safety while maximizing comfort and progress towards the destination. Most approaches have relied on hand-engineered planners that are built on top of perception and motion forecasting modules. However, a robust decision process has proven elusive, failing to handle the complexity of the real world.

In recent years, several approaches have been proposed, aiming at exploiting machine learning to learn to drive. Supervised learning approaches such as behavior cloning [47,3,16,17,62,56] that learn from human demonstrations are amongst

⋆ Denotes equal contribution.
† Work done during affiliation with Waabi.

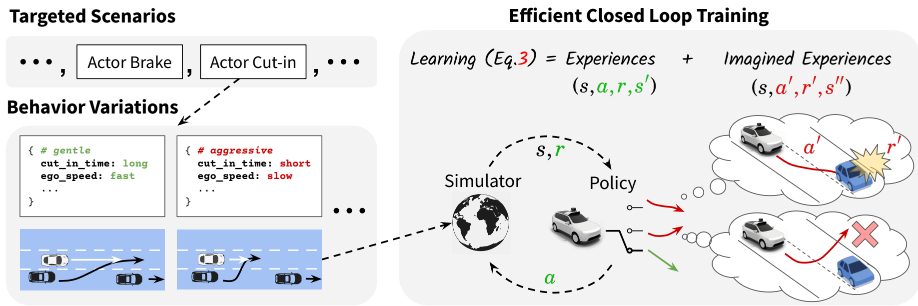

Fig. 1: We use behavioral variations on top of scenarios designed to target specific traffic interactions as the basis for learning. TRAVL efficiently learns to plan trajectories from both real and imagined experience.

the most popular, as large amounts of driving data can be readily obtained by instrumenting a vehicle to collect data while a human is driving. However, learning in this open-loop manner leads to distribution shift between training and deployment [59,17,18], as the model does not understand the closed-loop effects of its actions when passively learning from the expert data distribution.

Closed-loop training is one principled way to tackle this, by enabling the agent to continuously interact with the environment and thus learn to recover from its own mistakes. However, closing the loop while driving in the real world comes with many safety concerns as it is dangerous to update the software on the fly without proper safety verification . Furthermore, it is unethical to expose the self-driving vehicle (SDV) to safety-critical situations, which is necessary for learning to handle them. Finally, rare scenarios can take extremely long to capture in the wild and is impractical to scale. An appealing alternative is to learn to drive in a virtual environment by exploiting simulation systems [22,82]. While encouraging results have been demonstrated [39,73,62], there is still a lack of understanding of how to build training benchmarks for effective closed-loop training.

In this paper we aim to shed some light on this by studying the following questions: *What type of scenarios do we need to learn to drive safely?* Simple scenarios used in previous works such as car racing [78,38,53] and empty-lanes [33] are insufficient in capturing the full complexity of driving. Should we instead simulate complex free-flow traffic[5] [27,29] or design scenarios targeting particular traffic situations [22] that test specific self-driving capabilities? We have seen in the context of supervised learning [19,41,32] that large scale data improves generalization of learned models. Is this also the case in closed-loop? *How many scenarios do we need?* What effect does scaling our scenarios have on the quality of the learned agents? How should we scale the dataset to best improve performance?

To better understand these questions, we present (to our knowledge) the first study that analyzes the effect of different training benchmark designs on

[5] This is similar to how we encounter random events when collecting data.

the success of learning neural motion planners. Towards this goal, we developed a sophisticated highway driving simulator that can create both realistic free-flow traffic as well as targeted scenarios capable of testing specific self-driving capabilities. In particular, the latter is achieved by exploiting procedural modeling which composes variations of unique traffic patterns (*e.g.*, lead actor breaking, merging in from an on ramp). Since each of these patterns is parameterized, we can sample diverse variations and generate a large set of scenarios automatically. Under this benchmark, we show:

1. Scenarios designed for specific traffic interactions provide a richer learning signal than generic free-flow traffic simulations. This is likely because the former ensures the presence of interesting and safety-critical interactions.
2. Training on smaller scenario variations leads to more unsafe driving behaviors. This suggests that crafting more variations of traffic situations is key when building training benchmarks.
3. Existing RL-based approaches have difficulty learning the intricacies of many scenarios. This is likely because they typically learn a direct mapping from observations to control signals (*e.g.*, throttle, steering). Thus, they regrettably lack multi-step lookahead reasoning into the future, which is necessary to handle complex scenarios such as merging into crowded lanes. Furthermore, learning these agents in a model-free manner can be extremely slow, as the agents have to learn with trial and error through costly simulations.

To address the struggles of current RL approaches, we propose trajectory value learning (TRAVL), a method which learns *long-term reasoning efficiently in closed-loop*. Instead of myopically outputting control commands independently at each timestep, TRAVL can perform decision-making with explicit multistep look-ahead by *planning* in trajectory space. Our model learns a deep feature map representation of the state which can be fused with trajectory features to directly predict the Q-value of following that trajectory. Inference amounts to selecting the maximum value trajectory plan from a sampled trajectory set. Unlike conventional model-based planning, this bypasses the need to explicitly model and predict all state variables and transitions, as not all of them are equally important (*e.g.*, a far away vehicle is of less interest in driving). Furthermore, our trajectory-based formulation allows us to cheaply produce additional *imagined* (*i.e.*, counterfactual) experiences, resulting in significantly better learning efficiency compared to model-free methods which need to rely solely on interacting in the environment.

Summary of contributions: In this paper, we present an in-depth empirical study on how various design choices of training data generation can affect the success of learning driving policies in closed-loop. This allows us to identify a number of guidelines for building effective closed-loop training benchmarks. We further propose a new algorithm for efficient and effective learning of long horizon driving policies, that better handle complex scenarios which mimic the complexity of the real-world. We believe our work can serve as a starting point to rethink how we shall conduct closed-loop training for autonomous driving.

2 Related Work

Open-loop training: In open-loop training, the agent does not take any actions and instead learns passively by observing expert states and actions. ALVINN [55] first explored behavior cloning as an open-loop training method for end-to-end visuomotor self-driving. Since then, several advances in data augmentation [3,16], neural network architecture [47,16,56] and auxiliary task design [17,62] have been made in order to handle more complex environments. Additionally, margin-based learning approaches [79,80,61,60] incorporate structured output spaces, while offline RL [68,37] exploits reward functions. The primary challenge in open-loop training is the distribution shift encountered when the predicted actions are rolled out in closed-loop. While techniques such as cleverly augmenting the training data [3] partially alleviate this issue, challenges remain.

Closed-loop Training: The most popular paradigm for closed-loop training is reinforcement learning (RL). In contrast to open-loop learning, online RL approaches [40,65,66] do not require pre-collected expert data, but instead learn through interacting with the environment. However, such methods have prohibitively low sample efficiency [13] and can take several weeks to train a single model [73]. To address this issue, auxiliary tasks have been used as additional sources of supervision, such as predicting affordances [12,62,73], scene reconstruction [33] or imitation-based pre-training [39]. Note that our learning approach is orthogonal to these tasks and thus can be easily combined. Model-based RL approaches are more sample efficient. They assume access to a world model, which provides a cheaper way to generate training data [72,23,8] in addition to the simulator. It can also be used during inference for planning with multi-step lookaheads [48,26,15,69,64,51,25,70]. Yet, not many works have been explored in the context of self-driving. [13] uses an on-rails world model to generate data for training and empirically shows better efficiency. [52] uses a learned semantic predictor to perform multistep look-ahead during inference. Our work enjoys both benefits with a unified trajectory-based formulation.

When an expert can be queried online, other closed-loop training techniques such as DAgger style approaches [59,53,14] can be applied. When the expert is only available during offline data collection, one can apply closed-loop imitation learning methods [31,36]. However, building an expert can be expensive and difficult, limiting the applications of these methods.

Closed-loop Benchmarking: Directly closing the loop in the real world [33] provides the best realism but is unsafe. Thus, leveraging simulation has been a dominant approach for autonomous driving. Environments focused on simple car racing [78,7] are useful in prototyping quickly but can be over-simplified for real-world driving applications. More complex traffic simulators [43,11,2,4] typically use heuristic-based actors to simulate general traffic flow. Learning-based traffic models have also been explored recently [5,71]. However, we show that while useful for evaluating the SDV in nominal conditions, general traffic flow provides

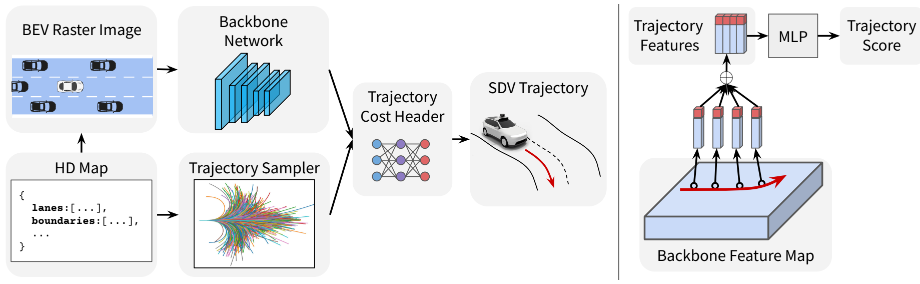

Fig. 2: TRAVL leverages rich backbone features to predict the cost of following a trajectory. The lowest costed trajectory is selected as the SDV's plan

limited learning signal as interesting interactions are not guaranteed to happen frequently. As the majority of traffic accidents can be categorized into a few number of situations [49], recent works focus on crafting traffic scenarios specifically targeting these situations for more efficient coverage [67,1,22,21,81,58]. However, visual diversity is often more stressed than behavioral diversity. For example, the CARLA benchmark [22] has 10 different scenario types and uses geolocation for variation, resulting in visually diverse backgrounds but fixed actor policies [6]. Other approaches include mining real data [9], or using adversarial approaches [75,35,20,24]. Multi-agent approaches that control different actors with different policies [82,6] have also been proposed. However, these works have not studied the effects on learning in detail.

3 Learning Neural Planners in Closed-loop

Most model-free RL-based self-driving approaches parametrize the action space as instant control signals (e.g., throttle, steering), which are directly predicted from state observations. While simple, this parameterization hampers the ability to perform long-term reasoning into the future, which is necessary in order to handle complex driving situations. Furthermore, this approach can lead to inefficient learning as it relies solely on experiences collected from the environment, which may contain only sparse supervisory signals. Model-based approaches address these issues by explicitly a predictive model of the world. However, performing explicit model rollouts online during inference can be prohibitively expensive especially if the number of potential future trajectories considered is very large.

We address these issues by combining aspects of model-free and model-based approaches. In particular, we learn to reason into the future by directly costing trajectories without explicit model rollouts, resulting in more efficient inference. In addition to using real experience from the environment, our trajectory output representation allows us to learn from imagined (*i.e.*, counterfactual) experiences collected from an approximate world model, greatly improving sample efficiency.

[6] https://github.com/carla-simulator/scenario_runner

3.1 Preliminaries on RL

The goal of an SDV is to make safe decisions sequentially. This can be modeled as a Markov Decision Process (MDP) : $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent state and action spaces respectively, such as raw observations of the scene and control signals for the ego-vehicle. $P(s'|s,a)$ and $R(s,a,s')$ represent the transition dynamics and reward functions respectively, and $\gamma \in (0,1)$ is the discount factor. We are interested in learning an optimal policy that maximizes the expected discounted return,

$$\pi^*(a|s) = \arg\max_{\pi} \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{T} \gamma^t R(s^t, a^t, s^{t+1})\right]$$

Off-policy RL algorithms are popular solutions due to their high data efficiency since they are agnostic to the data collecting policy and thus do not constantly require fresh data. A general form of the off-policy learning process can be described as iteratively alternating between *policy evaluation* and *policy improvement* steps. Specifically, one can maintain a learnable Q-function $Q^k(s,a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{T} \gamma^t R(s^t, a^t, s^{t+1})\right]$, which captures the expected future return when executing $\pi^k(a|s)$, with k being the learning iteration. In the policy evaluation step, the Bellman operator $\mathcal{B}_\pi Q \coloneqq R + \gamma \mathcal{P}_\pi Q$ is applied to update Q based on simulated data samples. Here, the transition matrix $\mathcal{P}_\pi$ is a matrix coupled with the policy π, *i.e.*, $\mathcal{P}_\pi Q(s,a) = \mathbb{E}_{s'\sim P(s'|s,a), a'\sim\pi(a'|s')}[Q(s',a')]$. We can then improve the policy π to favor selecting actions that maximize the expected Q-value. However, both steps require evaluations on all possible (s,a) pairs, and thus this is intractable for large state and action spaces. In practice, one can instead apply empirical losses over a replay buffer, *e.g.*, minimizing the empirical ℓ_2 loss between the left and right hand side of the Bellman operator. The replay buffer is defined as the set $\mathcal{D} = \{(s,a,r,s')\}$ holding past experiences sampled from $P(s'|s,a)$ and $\pi(a|s)$. Putting this together, the updating rules can be described as follows,

$$\begin{aligned} Q^{k+1} &\leftarrow \arg\min_{Q} \mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\left((r + \gamma \mathbb{E}_{a'\sim\pi^k}[Q^k(s',a')]) - Q(s,a)\right)^2\right] \text{ (evaluation)}, \\ \pi^{k+1} &\leftarrow (1-\epsilon) \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi}[Q^{k+1}(s,a)] + \epsilon U(a), \quad \text{(improvement)} \end{aligned} \tag{1}$$

where U is the uniform distribution and ϵ is introduced for epsilon-greedy exploration, *i.e.*, making a greedy action under Q with probability $1-\epsilon$ and otherwise randomly exploring other actions with probability ϵ. Note that Eq. 1 reduces to standard Q-learning when we use $\pi^{k+1}(s) = \arg\max_a Q^{k+1}(s,a)$ as the policy improvement step instead.

3.2 Planning with TRAVL

Our goal is to design a driving model that can perform long term reasoning into the future by *planning*. To this end, we define our action as a trajectory

$\tau = \{(x^0, y^0), (x^1, y^1), \cdots, (x^T, y^T)\}$, which navigates the ego-vehicle for the next T timesteps. Here (x^t, y^t) is the spatial location in birds eye view (BEV) at timestep t. Inspired by humans, we decompose the cost of following a trajectory into a short-term *cost-to-come*, $R_\theta(s, \tau)$, defined over the next T timesteps, and a long-term *cost-to-go* $V_\theta(s, \tau)$ that operates beyond that horizon. The final Q-function is defined as

$$Q_\theta(s, \tau) = R_\theta(s, \tau) + V_\theta(s, \tau)$$

Note that both R_θ and V_θ are predicted with a neural network. In the following, we describe our input state representation, the backbone network and cost predictive modules used to predict Q_θ follow by our planning inference procedure.

Input Representation: Following [3,32,57], our state space $\mathcal{S}$ contains an HD map as well as the motion history of the past T' seconds of both the ego-vehicle and other actors. To make the input data amenable to standard convolutional neural networks (CNNs), we rasterize the information into a BEV tensor, where for each frame within the history horizon T', we draw bounding boxes of all actors as 1 channel using a binary mask. The ego-vehicle's past positions are also rasterized similarly into T' additional channels. We utilize an M channel tensor to represent the HD map, where each channel encodes a different map primitive, such as centerlines or the target route. Finally, we include two more channels to represent the (x, y) coordinates of BEV pixels [42]. This results in a input tensor of size $\mathbb{R}^{H \times W \times (2T'+M+2)}$, where H and W denotes the size of our input region around the SDV.

Backbone Network: To extract useful contextual information, we feed the input tensor to a backbone network. As the input modality is a 2D image, we employ a CNN backbone adapted from ResNet [28]. Given an input tensor of size $\mathbb{R}^{H \times W \times (2T'+M+2)}$, the backbone performs downsampling and computes a final feature map $\mathbf{F} \in \frac{H}{8} \times \frac{W}{8} \times C$, where C is the feature dimension. More details on the architecture are provided in the supplementary.

Cost Predictive Header: We use a cost predictive header that takes an arbitrary trajectory τ and backbone feature map $\mathbf{F}$ as inputs, and outputs two scalar values representing the cost-to-come and cost-to-go of executing τ. As τ is represented by a sequence of 2D waypoints $\{(x^0, y^0), (x^1, y^1), \cdots, (x^T, y^T)\}$, we can extract context features of τ by indexing the t channel of the backbone feature $\mathbf{F}$ at position (x^t, y^t) for each timestep t. We then concatenate the features from all timesteps into a single feature vector $\mathbf{f}_\tau$. Note that the backbone feature $\mathbf{F}$ encodes rich information about the environment, and thus such an indexing operation is expected to help reason about the goodness of a trajectory, *e.g.*, if it is collision-free and follows the map. We also include kinematic information of τ into $\mathbf{f}_\tau$ by concatenating the position (x^t, y^t), velocity (v^t), acceleration (a^t), orientation (θ_t) and curvature $(\kappa_t, \dot{\kappa}_t)$. We use two shallow (3 layer) multi-layer perceptrons (MLPs) to regress the cost-to-come and cost-to-go from $\mathbf{f}_\tau$ before finally summing them to obtain our estimate of $Q(s, \tau)$.

Efficient Inference: Finding the optimal policy $\tau^* := \arg\max_\tau Q(s, \tau)$ given a Q-function over trajectories is difficult as τ lies in a continuous and high-dimensional space that has complex structures (*e.g.*, dynamic constraints). To make inference efficient, we approximate such an optimization problem using a sampling strategy [63,77,61,79]. Towards this goal, we first sample a wide variety of trajectories $\mathcal{T}$ that are physically feasible for the ego-vehicle, and then pick the one with maximum Q-value

$$\tau^* = \arg\max_{\tau \in \mathcal{T}} Q(s, \tau).$$

To obtain a set of trajectory candidates $\mathcal{T}$, we use a map-based trajectory sampler, which samples a set of lane following and lane changing trajectories following a bicycle model [54]. Inspired by [61], our sampling procedure is in the Frenet frame of the road, allowing us to easily sample trajectories which consider map priors, *e.g.*, follow curved lanes. Specifically, longitudinal trajectories are obtained by fitting quartic splines to knots corresponding to varying speed profiles, while lateral trajectories are obtained by first sampling sets of various lateral offsets (defined with respect to reference lanes) at different longitudinal locations and then fitting quintic splines to them. In practice, we find embedding map priors in this manner can greatly improve the model performance. In our experiments we sample roughly 10k trajectories per state. Note that despite outputting an entire trajectory as the action, for inference we use an MPC style execution [10] where the agent only executes an initial segment of the trajectory before replanning with the latest observation. Further discussion on this method of planning can be found in the supplementary.

3.3 Efficient Learning with Counterfactual Rollouts

The most straightforward way to learn our model is through classical RL algorithms. We can write the policy evaluation step in Eq. 1 as

$$Q^{k+1} \leftarrow \arg\min_{Q_\theta} \mathbb{E}_{\mathcal{D}} \left[\left(Q_\theta - \mathcal{B}^k_\pi Q^k \right)^2 \right], \quad s.t. \quad Q_\theta = R_\theta + V_\theta, \tag{2}$$

where $\mathcal{B}_\pi Q := R + \gamma \mathcal{P}_\pi Q$ is the Bellman operator. However, as we show in our experiments and also demonstrated in other works [76], such model-free RL algorithms learn very slowly. Fortunately, our trajectory-based formulation and decomposition of Q_θ into R_θ and V_θ allow us to design a more efficient learning algorithm that follows the spirit of model-based approaches.

One of the main benefits of model-based RL [72] is the ability to efficiently sample imagined data through the world dynamics model, as this helps bypass the need for unrolling the policy in simulation which can be computationally expensive. However, for complex systems the learned model is likely to be also expensive, *e.g.*, neural networks. Therefore, we propose a simple yet effective world model where we assume the actors are not intelligent and do not react to different SDV trajectories. Suppose we have a replay buffer $\mathcal{D} = \{(s^t, \tau^t, r^t, s^{t+1})\}$

collected by interacting our current policy π with the simulator. To augment the training data with cheaply generated imagined data (s^t, τ', r', s'), we consider a counterfactual trajectory τ' that is different from τ. The resulting counterfactual state s' simply modifies s^{t+1} such that the ego-vehicle follows τ', while keeping the actors' original states the same. The counterfactual reward can then be computed as $r' = R(s^t, \tau', s')$. We exploit our near limitless source of counterfactual data for dense supervision on the short term predicted cost-to-come R_θ. Efficiently learning R_θ in turn benefits the learning of the Q-function overall. Our final learning objective (policy evaluation) is then

$$Q^{k+1} \leftarrow \arg\min_{Q_\theta} \mathbb{E}_{\mathcal{D}} \left[\underbrace{\left(Q_\theta(s,\tau) - \mathcal{B}^k_\pi Q^k(s,\tau)\right)^2}_{\text{Q-learning}} + \alpha_k \underbrace{\mathbb{E}_{\tau' \sim \mu(\tau'|s)} \left(R_\theta(s,\tau') - r'\right)^2}_{\text{Counterfactual Reward Loss}} \right],$$
$$s.t. \quad Q_\theta = R_\theta + V_\theta, \quad V_\theta = \gamma \mathcal{P}^k_\pi Q^k. \tag{3}$$

Here, μ is an arbitrary distribution over possible trajectories and characterizes the exploration strategy for counterfactual supervision. We use a uniform distribution over the sampled trajectory set $\mathcal{T}$ in all our experiments for simplicity. To perform an updating step in Eq. 3, we approximate the optimal value of Q_θ by applying a gradient step of the empirical loss of the objective. In practice, the gradients are applied over the parameters of R_θ and V_θ, whereas Q_θ is simply obtained by $R_\theta + V_\theta$. We also find that simply using the Q-learning policy improvement step suffices in practice. We provide more implementation details in the supplementary material. Note that we use counterfactual rollouts, and thus the simplified non-reactive world dynamics, only as supervision for short term reward/cost-to-come component. This can help avoid compounding errors of using such an assumption for long-term imagined simulations.

Theoretical analysis: The counterfactual reward loss component can introduce modeling errors due to our approximated world modeling. As the Q-learning loss term in Eq. 3 is decreasing when k is large, such errors can have significant effects, hence it is non-obvious whether our learning procedure Eq. 3 can satisfactorily converge. We now show that iteratively applying policy evaluation in Eq. 3 and policy update in Eq. 1 indeed converges under some mild conditions.

Lemma 1. *Assuming R is bounded by a constant R_{max} and α_k satisfies*

$$\alpha_k < \left(\frac{1}{\gamma^k C} - 1\right)^{-1} \left(\frac{\pi^k}{\mu}\right)_{min}, \tag{4}$$

with C an arbitrary constant, iteratively applying Eq. 3 and the policy update step in Eq. 1 converges to a fixed point.

Furthermore, it converges to the optimal Q-function, Q^*. We refer the reader to the supplementary material for detailed proofs.

Theorem 1. *Under the same conditions as Lemma 1, our learning procedure converges to Q^*.*

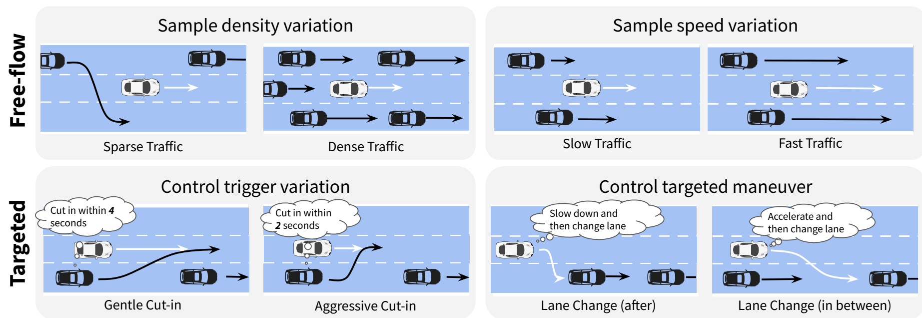

Fig. 3: **Top row:** Free-flow scenarios generated by sampling parameters such as density and actor speed. **Bottom row:** Targeted scenarios generated by enacting fine-grained control on the actors to target specific traffic situations.

4 Large Scale Closed-loop Benchmark Dataset

We now describe how we design our large scale closed-loop benchmark. In our simulator the agent drives on a diverse set of highways, which contain standard, on-ramp, merge and fork map topologies with varying curvature and number of lanes. We use IDM [74] and MOBIL [34] policies to control actors. As our work is focused on learning neural planners in closed-loop, we simulate bounding boxes and trajectory information of actors, and leave sensor simulation for future work.

There are two popular ways to testing an autonomous driving system: 1) uncontrolled traffic environments and 2) controlled scenarios that test certain self-driving capabilities such as reacting to a cut-in. However, there has been no analysis in the literature of the effects of *training* in these different environments. Towards this goal, we construct two training benchmarks based on these two different paradigms.

Free-flow Scenario Set: Free-flow scenarios are similar to what we observe in real-world data, where we do not enact any fine-grained control over other actors. We define a generative model which samples from a set of parameters which define a scenario. Parameters which vary the initial conditions of actors include density, initial speed, actor class, and target lane goals. Parameters which vary actor driving behavior include actor target speed, target gap, speed limit, maximum acceleration and maximum deceleration. We also vary the map topology, road curvature, geometry, and the number of lanes. We use a mixture of truncated normal distributions for continuous properties and categorical distribution for discrete properties. More details are provided in the supplementary.

Targeted Scenario Set: Targeted scenarios are those which are designed to test autonomous vehicles in specific traffic situations. These scenarios are designed to ensure an autonomous vehicle has certain capabilities or meets certain

Method		Pass Rate ↑	Col. Rate ↓	Prog. ↑	MinTTC↑	MinDist↑
Imit. Learning	C	0.545	0.177	240	0.00	2.82
PPO [66]	C	0.173	0.163	114	0.00	5.56
A3C [46]	C	0.224	0.159	284	0.03	4.65
RAINBOW[7] [30]	C	0.435	0.270	234	0.00	1.38
Imit. Learning	T	0.617	0.261	**286**	0.00	1.49
PPO [66]	T	0.273	0.249	200	0.00	1.73
A3C [46]	T	0.362	0.137	135	0.30	6.14
RAINBOW[7] [30]	T	0.814	0.048	224	0.45	9.70
TRAVL (ours)		**0.865**	**0.026**	230	**0.82**	**12.62**

Table 1: We compare our approach against several baselines. Here C is using the standard control setting and T is using our proposed trajectory-based architecture and formulation. We see that trajectory-based approaches outperforms their control-based counterparts. We also see that our proposed method is able to learn more efficiently and outperform baselines.

requirements (*e.g.*, the ability to stop for a leading vehicle braking, the ability to merge onto the highway). In this work, we identified 3 ego-routing intentions (lane follow, lane change, lane merge) and 5 behavior patterns for other agents (braking, accelerating, blocking, cut-in, negotiating). Combining these options along with varying the number of actors and where actors are placed relative to the ego gives us a total of 24 scenario types (*e.g.* lane change with leading and trailing actor on the target lane). Each scenario type is then parameterized by a set of scenario-specific parameters such as heading and speed of the ego at initialization, the relative speed and location of other actors at initialization, time-to-collision and distance thresholds for triggering reactive actions (*e.g.* when an actor performs a cut-in), IDM parameters of other actors as well as map parameters. We then procedurally generate variations of these scenarios by varying the values of these parameters, which result in diverse scenario realizations with actor behaviors that share similar semantics.

Benchmarking: We briefly explain how we sample parameters for scenarios. As the free-flow scenarios aim to capture nominal traffic situations, we simply sample *i.i.d* random parameter values and hold out a test set. In contrast, each targeted scenario serves for benchmarking a specific driving capability, and thus we should prevent training and testing on the same (or similar) scenarios. To this end, we first generate a test set of scenarios aiming to provide thorough evaluations over the entire parameterized spaces. Because enumerating all possible combinations of parameter is intractable, we employ an all-pairs generative approach [50] which provides a much smaller set that contains all possible combinations for any pair of discrete parameters. This is expected to provide efficient testing while covering a significant amount of failure cases [45]. More details on this approach are in the supplementary. Finally, we hold out those test parameters when drawing random samples for the training and validation set.

			Test					
			Pass Rate ↑		Collision Rate ↓		Progress ↑	
			Free-flow	Targeted	Free-flow	Targeted	Free-flow	Targeted
Train	RB[7]+T	Free-flow	0.783	0.453	0.198	0.228	146	173
		Targeted	0.885	0.815	0.104	0.048	231	224
	TRAVL	Free-flow	0.784	0.696	0.198	0.177	229	219
		Targeted	0.903	0.865	0.089	0.026	172	230

Table 2: We train RAINBOW[7] and TRAVL on different sets and evaluate on different sets. We see that training on targeted scenarios performs better than training on free-flow scenarios, even when evaluated on free-flow scenarios.

5 Experiments

In this section, we showcase the benefits of our proposed learning method TRAVL by comparing against several baselines. We empirically study the importance of using targeted scenarios compared to free-flow scenarios for training and evaluation. Finally, we further study the effect of data diversity and scale and show that large scale, behaviorally diverse data is crucial in learning good policies.

Datasets and Metrics: The free-flow dataset contains 834 training and 274 testing scenarios. The targeted dataset contains 783 training and 256 testing scenarios. All scenarios last for 15 seconds on average.

We use a number of autonomy metrics for evaluating safety and progress. *Scenario Pass Rate* is the percentage of scenarios that pass, which is defined as reaching the goal (e.g., maintain a target lane, reach a distance in a given time) without collision or speeding violations. *Collision Rate* computes the percentage of scenarios where ego collides. *Progress* measures the distance traveled in meters before the scenario ends. *Minimum Time to Collision (MinTTC)* measures the time to collision with another actor if the ego state were extrapolated along its future executed trajectory, with lower values meaning closer to collision. *Minimum Distance to the Closest Actor (MinDistClAct)* computes the minimum distance between the ego and any other actor in the scene. We use the median when reporting metrics as they are less sensitive to outliers.

Closed-loop Benchmarking: We train and evaluate TRAVL and several baselines on our proposed targeted scenario set. We evaluate A3C [46], PPO [66] and a simplified RAINBOW[7] [30] as baseline RL algorithms, and experiment with control (C) and trajectory (T) output representations. We also include imitation learning (IL) baselines supervised from a well-tuned auto-pilot. In the control setting, the action space consists of 9 possible values for steering angle and 7 values for throttle, yielding a total of 63 discrete actions. In the trajectory setting,

[7] We only include prioritized replay and multistep learning as they were found to be the most applicable and important in our setting.

Fig. 4: Increasing scenario diversity improves performance across the board.

each trajectory sample is treated as a discrete action. All methods use the same backbone and header architecture, trajectory sampler, and MPC style inference when applicable. Our reward function consists of a combination of progress, collision and lane following terms. More details on learning, hyperparameters, reward function, and baselines are provided in the supplementary material.

As shown in Table. 1, trajectory-based methods outperform their control-based counterparts, suggesting our proposed trajectory-based formulation and architecture allow models to better learn long-term reasoning and benefit from the trajectory sampler's inductive bias. We also see that RAINBOW[7]+T outperforms other RL trajectory-based baselines. This suggests that trajectory *value learning* is easier compared to learning a policy directly over the trajectory set. Furthermore, its off-policy nature allows the use of a replay buffer for more efficient learning. In contrast, on-policy methods such as A3C and PPO have difficulty learning, *e.g.*, it takes 10x longer to overfit to a single scenario compared to off-policy approaches. This aligns with similar findings in [22]. Additionally, IL baselines outperform weaker RL baselines that suffer from issues of sample complexity, but falls short to more efficient RL baselines due to the problem of distribution shift. Finally, TRAVL outperforms the baselines. We believe this is because our model-based counterfactual loss provides denser supervisions and thus reduces noisy variances during learning. This is empirically validated in Fig. 5 as our method has the least variance and converges much faster.

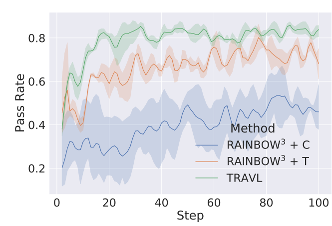

Fig. 5: Training curves for 3 runs. TRAVL has the least variance and converges faster.

Targeted vs Free-flow: We now study the effect of using different types of scenarios for training by comparing models learned on our *targeted* vs *free-flow* set. As shown in Table 2, a model learned on the targeted set performs the best. Notably this is *even true when evaluating on the free-flow test set*, which is closer in distribution to the free-flow train set. The effectiveness of the targeted set can be due to two reasons. Firstly, as scenarios and actors are carefully designed, each targeted scenario is more likely to provide interesting interactions, resulting in

Method	Pass Rate ↑	Col. Rate ↓	Prog. ↑	MinTTC ↑	MinDist ↑
Map Variation	0.738	0.070	230	0.53	8.97
Beh. Variation	**0.872**	**0.022**	228	0.60	9.82
Both	0.865	0.026	**231**	**0.82**	**12.6**

Table 3: We train a TRAVL agent on datasets with different axes of variation. Behavioral variation has larger effects than map for learning robust driving policies.

stronger learning signals. On the contrary, many of the free-flow scenarios can be relatively monotonous, with fewer interactions among actors. Secondly, the design of each targeted scenario is driven by autonomous driving capabilities which where determined with expert prior knowledge and are specifically meant to capture what is necessary to be able to drive in nearly all scenarios. As a result, each type of targeted scenario can be viewed as a basis over the scenario space. Sampling behavioral variations results in a scenario set that provides wide coverage.

Behavioral scale and diversity: We now study how many scenarios we need for learning robust policies. Standard RL setups use only a single environment (*e.g.*, a fixed set of behavioral parameters for non-ego agents) and rely on the stochastic nature of the policy to collect diverse data. However, we find this is not enough. We train models on datasets with varying amount of scenario variations while keeping the total number of training simulation steps constant. As shown in Fig. 4, models trained with more diverse scenario variations exhibit better performance. In particular, we see that while metrics like pass rate and progress saturate quickly, safety-related metrics improve as we increase the number of variations. This suggests that adding in data diversity allows the model to be better prepared for safety-critical situations. We also study which axis of variation has the largest effect. Specifically, we disentangle map (*i.e.*, geolocation) variations (road curvature, number of lanes, topologies) and behavioral variation (actor triggers, target speeds) and construct datasets that only contain one source of diversity while keeping the total number of scenarios the same. Table 3 shows that TRAVL is able to perform well even without map variations as our trajectory-based formulation allows us to embed strong map priors into the model. However, the behavioral variation component is crucial in learning more robust policies.

6 Conclusion

We have studied how to design traffic scenarios and scale training environments in order to create an effective closed-loop benchmark for autonomous driving. We have proposed a new method to efficiently learn driving policies which can perform long-term reasoning and planning. Our method reasons in trajectory space and can efficiently learn in closed-loop by leveraging additional imagined experiences. We provide theoretical analysis and empirically demonstrate the advantages of our method over the baselines on our new benchmark.

References

1. Apollo simulation 5
2. Balmer, M., Rieser, M., Meister, K., Charypar, D., Lefebvre, N., Nagel, K.: Matsim-t: Architecture and simulation times. In: Multi-agent systems for traffic and transportation engineering (2009) 4
3. Bansal, M., Krizhevsky, A., Ogale, A.: Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. arXiv (2018) 1, 4, 7
4. Ben-Akiva, M., Koutsopoulos, H.N., Toledo, T., Yang, Q., Choudhury, C.F., Antoniou, C., Balakrishna, R.: Traffic simulation with mitsimlab. In: Fundamentals of traffic simulation (2010) 4
5. Bergamini, L., Ye, Y., Scheel, O., Chen, L., Hu, C., Del Pero, L., Osiński, B., Grimmett, H., Ondruska, P.: Simnet: Learning reactive self-driving simulations from real-world observations. In: ICRA (2021) 4
6. Bernhard, J., Esterle, K., Hart, P., Kessler, T.: Bark: Open behavior benchmarking in multi-agent environments. In: IROS (2020) 5
7. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. arXiv (2016) 4
8. Buckman, J., Hafner, D., Tucker, G., Brevdo, E., Lee, H.: Sample-efficient reinforcement learning with stochastic ensemble value expansion. NeurIPS (2018) 4
9. Caesar, H., Kabzan, J., Tan, K.S., Fong, W.K., Wolff, E., Lang, A., Fletcher, L., Beijbom, O., Omari, S.: nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. arXiv (2021) 5
10. Camacho, E.F., Alba, C.B.: Model predictive control (2013) 8
11. Casas, J., Ferrer, J.L., Garcia, D., Perarnau, J., Torday, A.: Traffic simulation with aimsun. In: Fundamentals of traffic simulation (2010) 4
12. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. In: ICCV (2015) 4
13. Chen, D., Koltun, V., Krähenbühl, P.: Learning to drive from a world on rails. In: ICCV (2021) 4
14. Chen, D., Zhou, B., Koltun, V., Krähenbühl, P.: Learning by cheating. In: CoRL (2020) 4
15. Chua, K., Calandra, R., McAllister, R., Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. NeurIPS (2018) 4
16. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: ICRA (2018) 1, 4
17. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: ICCV (2019) 1, 2, 4
18. De Haan, P., Jayaraman, D., Levine, S.: Causal confusion in imitation learning. NeurIPS (2019) 2
19. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009) 2
20. Ding, W., Chen, B., Li, B., Eun, K.J., Zhao, D.: Multimodal safety-critical scenarios generation for decision-making algorithms evaluation. IEEE Robotics and Automation Letters **6**(2), 1551–1558 (2021) 5
21. Ding, W., Xu, C., Lin, H., Li, B., Zhao, D.: A survey on safety-critical scenario generation from methodological perspective. arXiv preprint arXiv:2202.02215 (2022) 5
22. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: CoRL (2017) 1, 2, 5, 13

23. Feinberg, V., Wan, A., Stoica, I., Jordan, M.I., Gonzalez, J.E., Levine, S.: Model-based value estimation for efficient model-free reinforcement learning. arXiv (2018) 4
24. Feng, S., Yan, X., Sun, H., Feng, Y., Liu, H.X.: Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. Nature communications **12**(1), 1–14 (2021) 5
25. Finn, C., Levine, S.: Deep visual foresight for planning robot motion. In: ICRA (2017) 4
26. Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., Davidson, J.: Learning latent dynamics for planning from pixels. In: ICML (2019) 4
27. Halkias, J., Colyar, J.: Model-predictive policy learning with uncertainty regularization for driving in dense traffic. FHWA-HRT-06-137, Washington, DC, USA (2006) 2
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 7
29. Henaff, M., Canziani, A., LeCun, Y.: Model-predictive policy learning with uncertainty regularization for driving in dense traffic. arXiv (2019) 2
30. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: AAAI (2018) 11, 12
31. Ho, J., Ermon, S.: Generative adversarial imitation learning. NeurIPS (2016) 4
32. Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., Ondruska, P.: One thousand and one hours: Self-driving motion prediction dataset. arXiv (2020) 2, 7
33. Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.M., Lam, V.D., Bewley, A., Shah, A.: Learning to drive in a day. In: ICRA (2019) 2, 4
34. Kesting, A., Treiber, M., Helbing, D.: General lane-changing model mobil for car-following models. Transportation Research Record (2007) 10
35. Koren, M., Kochenderfer, M.J.: Efficient autonomy validation in simulation with adaptive stress testing. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). pp. 4178–4183. IEEE (2019) 5
36. Kuefler, A., Morton, J., Wheeler, T., Kochenderfer, M.: Imitating driver behavior with generative adversarial networks. In: 2017 IEEE Intelligent Vehicles Symposium (IV) (2017) 4
37. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv (2020) 4
38. Li, Y., Song, J., Ermon, S.: Infogail: Interpretable imitation learning from visual demonstrations. NeurIPS (2017) 2
39. Liang, X., Wang, T., Yang, L., Xing, E.: Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In: ECCV (2018) 2, 4
40. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv (2015) 4
41. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014) 2
42. Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., Yosinski, J.: An intriguing failing of convolutional neural networks and the coordconv solution. NeurIPS (2018) 7
43. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., Wießner, E.: Microscopic traffic simulation

using sumo. In: 2018 21st international conference on intelligent transportation systems (ITSC) (2018) 4
44. Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., Yang, B., Ma, W.C., Urtasun, R.: Lidarsim: Realistic lidar simulation by leveraging the real world. In: CVPR (2020) 1
45. MIRA, H., Hillman, R.: Test methods for interrogating autonomous vehicle behaviour–findings from the humandrive project (2019) 11
46. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Harley, T., Lillicrap, T.P., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: ICML (2016) 11, 12
47. Muller, U., Ben, J., Cosatto, E., Flepp, B., Cun, Y.: Off-road obstacle avoidance through end-to-end learning. NeurIPS (2005) 1, 4
48. Nagabandi, A., Kahn, G., Fearing, R.S., Levine, S.: Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In: ICRA (2018) 4
49. Najm, W.G., Smith, J.D., Yanagisawa, M., et al.: Pre-crash scenario typology for crash avoidance research. Tech. rep. (2007) 5
50. Nie, C., Leung, H.: A survey of combinatorial testing. ACM Computing Surveys (CSUR) (2011) 11
51. Oh, J., Singh, S., Lee, H.: Value prediction network. NeurIPS (2017) 4
52. Pan, X., Chen, X., Cai, Q., Canny, J., Yu, F.: Semantic predictive control for explainable and efficient policy learning. In: ICRA (2019) 4
53. Pan, Y., Cheng, C.A., Saigol, K., Lee, K., Yan, X., Theodorou, E., Boots, B.: Agile autonomous driving using end-to-end deep imitation learning. arXiv (2017) 2, 4
54. Polack, P., Altché, F., d'Andréa Novel, B., de La Fortelle, A.: The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In: 2017 IEEE intelligent vehicles symposium (IV) (2017) 8
55. Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. NeurIPS (1988) 4
56. Prakash, A., Chitta, K., Geiger, A.: Multi-modal fusion transformer for end-to-end autonomous driving. In: CVPR (2021) 1, 4
57. Rhinehart, N., He, J., Packer, C., Wright, M.A., McAllister, R., Gonzalez, J.E., Levine, S.: Contingencies from observations: Tractable contingency planning with learned behavior models. arXiv (2021) 7
58. Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on scenario-based safety assessment of automated vehicles. IEEE access **8**, 87456–87477 (2020) 5
59. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: AAAI (2011) 2, 4
60. Sadat, A., Casas, S., Ren, M., Wu, X., Dhawan, P., Urtasun, R.: Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In: ECCV (2020) 4
61. Sadat, A., Ren, M., Pokrovsky, A., Lin, Y.C., Yumer, E., Urtasun, R.: Jointly learnable behavior and trajectory planning for self-driving vehicles. In: IROS (2019) 4, 8
62. Sauer, A., Savinov, N., Geiger, A.: Conditional affordance learning for driving in urban environments. In: CoRL (2018) 1, 2, 4
63. Schlechtriemen, J., Wabersich, K.P., Kuhnert, K.D.: Wiggling through complex traffic: Planning trajectories constrained by predictions. In: 2016 IEEE Intelligent Vehicles Symposium (IV) (2016) 8

64. Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al.: Mastering atari, go, chess and shogi by planning with a learned model. Nature (2020) 4
65. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: ICML (2015) 4
66. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv (2017) 4, 11, 12
67. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and service robotics (2018) 5
68. Shi, T., Chen, D., Chen, K., Li, Z.: Offline reinforcement learning for autonomous driving with safety and exploration enhancement. arXiv (2021) 4
69. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science (2018) 4
70. Srinivas, A., Jabri, A., Abbeel, P., Levine, S., Finn, C.: Universal planning networks: Learning generalizable representations for visuomotor control. In: ICML (2018) 4
71. Suo, S., Regalado, S., Casas, S., Urtasun, R.: Trafficsim: Learning to simulate realistic multi-agent behaviors. In: CVPR (2021) 4
72. Sutton, R.S.: Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In: Machine learning proceedings 1990 (1990) 4, 8
73. Toromanoff, M., Wirbel, E., Moutarde, F.: End-to-end model-free reinforcement learning for urban driving using implicit affordances. In: CVPR (2020) 2, 4
74. Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical review E (2000) 10
75. Wang, J., Pun, A., Tu, J., Manivasagam, S., Sadat, A., Casas, S., Ren, M., Urtasun, R.: Advsim: Generating safety-critical scenarios for self-driving vehicles. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9909–9918 (2021) 5
76. Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., Ba, J.: Benchmarking model-based reinforcement learning. arXiv (2019) 8
77. Werling, M., Ziegler, J., Kammel, S., Thrun, S.: Optimal trajectory generation for dynamic street scenarios in a frenet frame. In: ICRA (2010) 8
78. Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., Sumner, A.: Torcs, the open racing car simulator. Software available at http://torcs. sourceforge. net (2000) 2, 4
79. Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R.: End-to-end interpretable neural motion planner. In: CVPR (2019) 4, 8
80. Zeng, W., Wang, S., Liao, R., Chen, Y., Yang, B., Urtasun, R.: Dsdnet: Deep structured self-driving network. In: ECCV (2020) 4
81. Zhong, Z., Tang, Y., Zhou, Y., Neves, V.d.O., Liu, Y., Ray, B.: A survey on scenario-based testing for automated driving systems in high-fidelity simulation. arXiv preprint arXiv:2112.00964 (2021) 5
82. Zhou, M., Luo, J., Villella, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., Alban, M., Fadakar, I., Chen, Z., et al.: Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. arXiv (2020) 1, 2, 5