

Closed-Loop Supervised Fine-Tuning of Tokenized Traffic Models

Zhejun Zhang¹ Peter Karkus¹ Maximilian Igl¹
 Wenhao Ding¹ Yuxiao Chen¹ Boris Ivanovic¹ Marco Pavone^{1,2}
¹NVIDIA Research ²Stanford University

{zhejunz, pkarkus, migl, wenhaod, yuxiaoc, bivanovic, mpavone}@nvidia.com
 pavone@stanford.edu

Abstract

Traffic simulation aims to learn a policy for traffic agents that, when unrolled in closed-loop, faithfully recovers the joint distribution of trajectories observed in the real world. Inspired by large language models, tokenized multi-agent policies have recently become the state-of-the-art in traffic simulation. However, they are typically trained through open-loop behavior cloning, and thus suffer from covariate shift when executed in closed-loop during simulation. In this work, we present *Closest Among Top-K* (CAT-K) rollouts, a simple yet effective closed-loop fine-tuning strategy to mitigate covariate shift. CAT-K fine-tuning only requires existing trajectory data, without reinforcement learning or generative adversarial imitation. Concretely, CAT-K fine-tuning enables a small 7M-parameter tokenized traffic simulation policy to outperform a 102M-parameter model from the same model family, achieving the top spot on the Waymo Sim Agent Challenge leaderboard at the time of submission. The code is available at <https://github.com/NVlabs/catk>.

1. Introduction

Traffic modeling is a cornerstone of autonomous driving simulation and evaluation, typically formulated as learning a multi-agent policy that imitates the behavior of traffic participants in the real world. Given a set of historical agent trajectories and scene context (map, traffic light states, etc.), the policy generates actions for all simulated agents. The task gives rise to an imitation learning (IL) problem, with two key challenges: *multimodality* and *covariate shift*.

Traffic agent behavior is highly multimodal, and faithfully recovering accurate behavior distributions is a key challenge in the field. Inspired by large language models [2, 32], recent works introduce next-token-prediction (NTP) models where the policy reduces to a classifier over a discrete set of trajectory tokens, which makes it easier to represent highly-multimodal distributions. Accordingly,

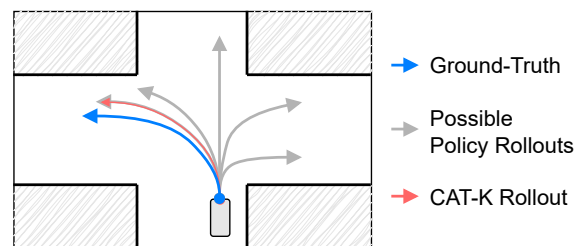


Figure 1. **Closest Among Top-K (CAT-K) rollouts.** The key idea of our approach is to unroll the policy during fine-tuning in a way that visited states remain close to the GT. At each time step, CAT-K first takes the top-K most likely action tokens according to the policy, then chooses the one leading to the state closest to the GT. As a result, CAT-K rollouts follow the mode of the GT (e.g., turning left), while random or top-K rollouts can lead to large deviations (e.g., going straight or right). Since the policy is essentially trained to minimize the distance between the rollout states and the GT states, the GT-based supervision remains effective for CAT-K rollouts, but not for random or top-K rollouts.

the Waymo Open Sim Agent Challenge (WOSAC) leaderboard [21] is heavily populated by tokenized traffic models [9, 35, 43, 47].

Covariate shift is a well-known challenge of IL arising from the gap between open-loop training and closed-loop deployment. When a model is trained on a fixed dataset of expert demonstrations, it can face a distribution mismatch between the states seen during training and those encountered during deployment, as small errors compound and lead to unseen states where the policy performs poorly. A classic approach to tackle this problem is Dataset Aggregation (DAgger) [27], which unrolls the policy and queries an expert to generate new demonstrations, but querying experts is not readily available for traffic simulation. Prior work has proposed closed-loop training using hand-crafted recovery controllers [1] or reinforcement learning (RL) [19, 23, 37]. However, it is inherently difficult to design rewards with high behavioral realism or recovery controllers robust to divergent modes. Consequently, such approaches are not currently competitive on WOSAC realism metrics.

Contributions: We introduce Closest Among Top-K (CAT-K) rollouts, a simple yet highly efficient fine-tuning strategy to address the open-loop to closed-loop gap. The key idea, illustrated in Fig. 1, is to unroll the multimodal policy during training in a way that the policy-visited states remain close to the ground-truth (GT) demonstration. CAT-K achieves this by first finding the K most likely modes of the policy and then choosing the mode closest to the GT. At inference time, actions are sampled from the policy according to the predicted likelihoods. During training, however, random sampling would lead to large deviations from the demonstrations, making them invalid and degrading the final policy performance. Our CAT-K rollout strategy balances being on-policy and staying close to GT demonstrations, such that they remain a valid supervision signal.

Experiments on the Waymo Open Motion Dataset (WOMD) demonstrate the efficacy of CAT-K. Notably, fine-tuning the SMART-7M [35] next-token-prediction traffic model enables it to outperform the 14x larger State-of-The-Art (SoTA) SMART-102M from the same model family, achieving the #1 spot on the public WOSAC leaderboard at the time of submission. To further demonstrate the potential of employing CAT-K fine-tuning for different tasks and policy representations, we apply it to an ego-motion planning task using a Gaussian Mixture Model (GMM) policy, yielding significant gains in closed-loop behavior, reducing collisions by 25.7% and off-road driving by 33.9%.

2. Related work

2.1. Traffic simulation

Prior work explored various architectures for traffic models, including conditional variational autoencoders [13, 31, 36], motion forecasting Transformers [6, 22, 30, 46], and diffusion models [11, 15, 18, 26, 44, 45]. However, long-term stability is an open challenge for these models due to covariate shift when transitioning from open-loop training to closed-loop deployment. Various ways have been proposed to mitigate covariate shift. BITS [36] and Symphony [12] introduce hierarchy, with high-level intent and low-level behavior prediction. TrafficBots [40, 42] incorporates configurable behaviors through destination and personality. Most diffusion models use guiding to generate rule-abiding behavior; however, they often struggle with computational efficiency and long horizons. The latest advances come from NTP models, which predict the next action as a token, e.g., Trajenglish [24], GUMP [9], KiGRAS [43], MotionLM [28], and SMART [35]. Notably, SMART is the current SoTA on the WOSAC leaderboard. In addition to the strong scalability and flexibility, NTP models also show better closed-loop stability than regression-based models, thanks to their discrete action space. However, achieving generalization and reducing compounding errors continue to be challenges.

2.2. Data augmentation for behavior cloning

Data augmentation is a simple yet effective way to improve the generalizability of traffic simulation models. ChauffeurNet [1] showed that carefully perturbing the vehicle trajectory and designing a recovery trajectory could alleviate the covariate shift suffered by behavior cloning (BC). However, this technique is difficult to apply to traffic simulation with complicated scenarios, including pedestrians and cyclists; and adding handcrafted recovery trajectories may negatively impact behavioral realism. Recently, NTP works such as Trajenglish [24] and SMART [35] have explored similar ideas by using noisy tokenization to perturb trajectories during training, but their data augmentation did not lead to significant improvements in performance. Our method is related to Trajenglish’s noisy tokenization, but importantly, instead of blindly sampling tokens close to the GT without considering the policy, our CAT-K rollout selects the token from the most likely K tokens predicted by the policy that is closest to the GT. It can be considered as a closed-loop variants of the “winner-takes-all” training strategy [20] widely applied to open-loop motion prediction models [22, 29, 41].

2.3. Closed-loop fine-tuning

As mentioned above, covariate shift is a major challenge faced by traffic simulation models, since most of them are trained in open-loop and evaluated in closed-loop. Even with data augmentation, the issue is not fully resolved as the augmented noisy data does not reflect the compounded error during closed-loop rollout. Therefore, some existing works explored the use of closed-loop fine-tuning.

As a classic remedy for the covariate shift, DAgger has been applied to end-to-end driving [25, 38, 39], yet its application in traffic simulation is limited, as it requires interactive demonstrations from human drivers or an expert policy. As an expert-free variant of DAgger, Data As Demonstrator (DAD) [34] obtains recovery actions using a pair of trajectories: a forward simulation (i.e., rollout) trajectory and a GT trajectory. At each time step, the policy is trained to return to the GT next state from the simulated current state. However, DAD fails when the rollout deviates from the GT trajectory. Our CAT-K rollout directly addresses this problem, enabling the use of DAD for closed-loop fine-tuning of traffic simulation and ego-vehicle policies.

Another popular approach is to use RL. BC is combined with RL to improve the robustness of a policy in [19]. Yet, it also exposes RL’s weakness in improving realism as it is difficult to handcraft a reward that promotes realism. Its follow-up work [23] learns a joint traffic model capable of rolling out the entire scenario by itself. The reward/loss are typically handcrafted, and can be distilled from explicit traffic rules [37]. With human preference data, RL from human feedback (RLHF) has also been applied to traffic model training for better user alignment [4, 10]. Finally, Sym-

phony [12] adds a Generative Adversarial Imitation Learning (GAIL) loss to encourage the rollout states to stay in distribution; however, the well-known issue of training stability and discriminator overfitting remains a challenge.

3. Background

3.1. Problem formulation

A multi-agent traffic simulation policy can be typically formulated as $\pi_\theta(\mathbf{a}_t | \mathbf{h}_t, \mathcal{M})$, where θ denotes the trainable model parameters, $\mathbf{h}_t = \mathbf{s}_{t-H:t}$ is the state history of length H , \mathcal{M} is the context, including for example high-definition (HD) maps and traffic light states, t is the current time step, $\mathbf{a}_t = [a_t^1, \dots, a_t^N]$ and $\mathbf{s}_t = [s_t^1, \dots, s_t^N]$ are respectively the actions and states of N agents at the current time step. The dimensions of actions a and states s are respectively denoted as D_a and D_s , i.e., $a \in \mathbb{R}^{D_a}$ and $s \in \mathbb{R}^{D_s}$. From the current states and actions at step t , the next states are computed using the per-agent forward dynamics $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t) = [f(s_t^i, a_t^i)]_{i=1}^N$. We assume that $f(s_t^i, a_t^i)$ is deterministic, and can be queried during training, which is the case for traffic simulation. Extensions to stochastic dynamics would be possible in future work. We define a rollout of T steps starting at $t = 0$ as a sequence of states $\mathbf{s}_{0:T} = [\mathbf{s}_0, \dots, \mathbf{s}_T]$, while the GT trajectories of all agents are denoted as $\hat{\mathbf{s}}_{0:T}$. For the training, we are given a dataset $\mathcal{D} = \{\hat{\mathbf{s}}_{0:T}^j, \mathcal{M}^j\}_{j=1}^{|\mathcal{D}|}$ of such real-world trajectories that we want to emulate with their corresponding contexts.

3.2. Next Token Prediction (NTP) policies

NTP policies, such as SMART [35] and Trajenglish [24], are parameterized as a probability distribution over a vocabulary of action tokens denoted as $V = \{x_c \mid c = 1, 2, \dots, |V|\}$, where $|V|$ is the size of the vocabulary, $x_c \in \mathbb{R}^{D_a}$ are template actions and $c \in \mathbb{N}$ is the token index. Hence, an autoregressive NTP policy for traffic simulation can equivalently be written as an agent-factorized categorical distribution at each timestep t , i.e.,

$$\pi_\theta(\mathbf{c}_t | \mathbf{h}_t, \mathcal{M}) = \prod_{i=1}^N \pi_\theta(c_t^i | \mathbf{h}_t, \mathcal{M}) = \prod_{i=1}^N \text{Cat}(c_t^i),$$

where $\text{Cat}(c_t^i)$ is the categorical distribution over the action token index for agent i (and not to be confused with our method CAT-K). Given the sampled output $\mathbf{c}_t = [c_t^1, \dots, c_t^N]$, the actions $\mathbf{a}_t = [x_{c_t^1}, \dots, x_{c_t^N}]$ are obtained using the token vocabulary V .

4. Method

Two key challenges in learning a policy from real-world trajectories are the *multimodal* nature of the trajectory distribution and the problem of *covariate shift* when policies are

Algorithm 1 CAT-K fine-tuning

```

1: Input: Policy  $\pi_\theta$ , action token vocabulary  $V$ , dataset  $\mathcal{D}$ 
2: Pre-train  $\pi_\theta(\mathbf{c}_t | \hat{\mathbf{h}}_t, \mathcal{M})$  with BC until convergence
3: repeat ▷ Closed-loop supervised fine-tuning
4:   Sample a traffic scenario  $\{\hat{\mathbf{s}}_{0:T}, \mathcal{M}\}$ 
5:   Init rollout state  $\mathbf{s}_0 = \hat{\mathbf{s}}_0$  ▷ CAT-K Rollout
6:   for  $t$  in  $[0, \dots, T-1]$  do ▷  $T$  steps
7:     for  $i$  in  $[1, \dots, N]$  do ▷  $N$  agents
8:       Get action index for rollout  $c_t^i$ . (Eq. 1)
9:       Get next rollout state  $s_{t+1}^i$ . (Eq. 3)
10:      Compute target  $\hat{c}_t^i$ . (Eq. 4)
11:    end for
12:  end for
13:  Update  $\theta$  by minimizing  $\mathcal{L}_\theta(\mathbf{s}_{0:T}, \hat{\mathbf{c}}_{1:T}, \mathcal{M})$ . (Eq. 5)
14: until convergence

```

trained open-loop, resulting in a distribution mismatch between expert states seen during training and states visited during policy deployment. Covariate shift can be overcome by closed-loop training, i.e., by training on trajectories sampled from the learned policy. However, this requires the generation of expert actions (or other notions of optimality) to be used as training targets along those trajectories [27]. Querying a human expert is infeasible at scale, RL-based methods require hard-to-define rewards, and methods such as GAIL [7] are prone to mode collapse. Consequently, they fail to achieve good performance in the WOSAC challenge.

An alternative strategy for generating “expert” actions is to construct recovery actions that bring the agent back to the available GT trajectory. However, this is complicated by the multimodal nature of the data, as the available GT trajectory might not be a valid recovery target for the generated trajectory. For example, as shown in Fig. 1, the GT trajectory $\hat{\mathbf{s}}_{0:T}$ turns left at the intersection, while the sampled trajectory $\mathbf{s}_{0:T} \sim \pi_\theta$ might go straight or turn right. As a result, while some SoTA traffic models [24, 35] augment the training data with recovery actions to reduce the covariate shift, they do so only from states that were reached by injecting small amounts of noise into the GT trajectory. This does guarantee that the GT trajectory remains a valid recovery target, but it *completely ignores the learned policy* and the state distribution induced by it. Instead, our method, *Closest Among Top-K* (CAT-K) rollout, informs the sampling process by the learned policy, but biases it towards the GT trajectory to guarantee the validity of the recovery actions. While simple to implement, it yields significant performance improvements in our experiments.

4.1. Closest Among Top-K (CAT-K) rollout

To facilitate the formulation, we define a top^K operator:

$$\{\xi_1, \dots, \xi_K\} = \text{top}^K_{c \in \{1, \dots, |V|\}} (\text{Cat}(c)),$$

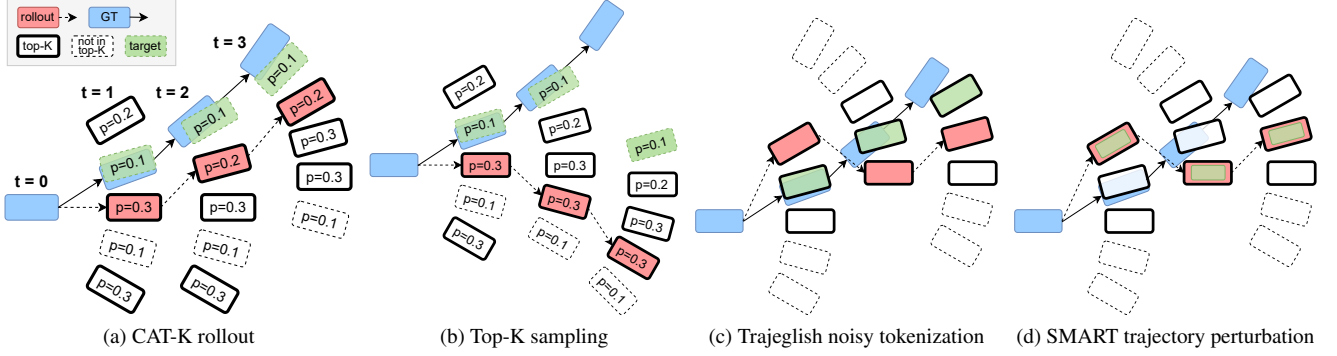


Figure 2. **Schematic comparison of CAT-K rollout, top-K sampling, and data augmentation techniques of Trajeglisch and SMART.** In this example, the token vocabulary has a size of 5. We rollout three steps from $t = 0$ to $t = 3$. For CAT-K rollout and top-K sampling, the top-K is w.r.t the probabilities p of tokens predicted by the policy. For the data augmentations used by Trajeglisch and SMART, the policy is unavailable, and the top-K selection is based on the negative distances between tokens and GT.

where $\text{Cat} : \mathbb{N} \rightarrow \mathbb{R}$ is the probability density of a categorical distribution on the vocabulary index, and $\{\xi_1, \dots, \xi_K\}$ are the K most probable indices. The top^K operator can be considered as a variation of the $\arg \max$ operator that returns multiple indices, with top^1 equivalent to $\arg \max$.

At time step t , the policy $\pi_\theta(c_t | \mathbf{h}_t, \mathcal{M})$ outputs independent categorical distributions over the token vocabulary for each agent. Our method, CAT-K rollout, deterministically rolls out the policy by selecting, at each time step and for each agent, the one action among the top-K likeliest according to π_θ that brings the agent closest to the GT next state. Using a distance metric $d(\cdot, \cdot)$ on the states, this is formally expressed as follows:

$$c_t^i = \arg \min_{c \in \{\xi_1, \dots, \xi_K\}} d(f(s_t^i, x_c), \hat{s}_{t+1}^i), \quad (1)$$

$$\{\xi_1, \dots, \xi_K\} = \text{top}^K_{c_t^i \in \{1, \dots, |V|\}} [\pi(c_t^i | \mathbf{h}_t, \mathcal{M})], \quad (2)$$

where c_t^i is the action token indices of agent i at step t for the CAT-K rollout, and $\{\xi_1, \dots, \xi_K\}$ are the top-K likeliest token index according to the policy. Given c_t^i , the next state is obtained using the vocabulary V and the dynamics as:

$$s_{t+1}^i = f(s_t^i, x_{c_t^i}). \quad (3)$$

These rollout states will be used as the input \mathbf{h} to the policy at the next time step. By doing this sequentially from $t = 0$ to $t = T - 1$ and repeating for all N agents, we obtain the CAT-K rollout trajectories $s_{0:T}$.

4.2. Closed-loop supervised fine-tuning

Given the CAT-K rollout trajectory $s_{0:T}$, we can apply the idea of DAD [34] and construct the recovery action indices \hat{c}_t from the GT trajectories $\hat{s}_{0:T}$ by finding the action token that brings each agent closest to its original trajectory:

$$\hat{c}_t^i = \arg \min_{c \in \{1, \dots, |V|\}} d(f(s_t^i, x_c), \hat{s}_{t+1}^i). \quad (4)$$

Given these indices \hat{c}_t , the NTP policy is trained using the cross-entropy loss

$$\mathcal{L}_\theta = -\frac{1}{NT} \sum_{i=1}^N \sum_{t=0}^{T-1} \log \pi_\theta(\hat{c}_t^i | \mathbf{h}_t, \mathcal{M}). \quad (5)$$

Since CAT-K rollout is effective only when the top-K rollouts of the policy cover the GT mode, we adopt a two-stage training procedure, summarized in Algorithm 1. First, we obtain a reasonably well-trained policy through BC pre-training, then fine-tune it using CAT-K rollouts.

4.3. Comparison to previous methods

In Fig. 2, we compare our method with previous data augmentation approaches that alleviate the covariate shift for NTP traffic simulation policies.

Top-K sampling is a common approach used by NTP models for generating sequences *during inference* as it improves the sample quality. However, it is unsuitable for generating trajectories during training as it does not consider the distance to the GT trajectory, and hence the validity of generated recovery actions (c.f. Fig. 2b). This can be partially addressed by post-hoc filtering of trajectories that deviated too far from the GT, but the resulting method is sample-inefficient, as most rollouts will be discarded, and the distance threshold hyperparameter is difficult to tune, since its optimal value varies across scenarios (e.g., high-speed vs. low-speed situations). In contrast, CAT-K rollout (c.f. Fig. 2a) is sample efficient by choosing the closest among top-K actions at each timestep and also removes the need to tune a distance-based hyperparameter. As shown in Tab. 2, our hyperparameter K provides strong results across a large range of values and is hence much easier to tune.

Instead of sampling top-K trajectories from the policy, current SoTA methods such as Trajeglisch [24] and SMART [35] use forms of **trajectory noising** to address the issue of covariate shift. All trajectory noising approaches rely

on the injection of small perturbations into the tokenization of the GT trajectory, but implementations can vary from each other in details. For example, in Trajenglish, the likelihood of each noised token is a function of the resulting distance to the ground-truth trajectory, i.e. $q_t^i(c) \sim \exp(-d(f(s_t^i, x_c), \hat{s}_{t+1}^i)/\tau)$, while for SMART, tokens are sampled uniformly from those K tokens with the highest likelihood $q_t^i(c)$. Additionally, Trajenglish, similar to our methods, uses the DAD [34] recovery target that finds the token that would bring the agent back to the GT trajectory (c.f. Fig. 2c), while SMART uses the next, also noised, token as target - effectively treating the noise injection as data augmentation on the training trajectories (c.f. Fig. 2d).

However, Trajenglish’s noisy tokenization does not yield a significant improvement (see Fig. 9 in [24]). Similarly, while SMART’s trajectory perturbation enhances the performance of a zero-shot policy trained on NuPlan [3] and evaluated on WOSAC, it does not improve performance on WOSAC itself (see Tab. 4 in [35]). We believe this is because the state distributions generated by their sampling strategies are sub-optimal as they completely ignore what state distribution would be induced by the learned policy, hence likely oversampling irrelevant states and under-sampling states the learned policy would actually encounter. By incorporating the learned policy into the sampling strategy, CAT-K generates a state distribution more like the policy’s, and is hence better able to reduce the covariate shift between training and inference. Although sampling from the policy incurs higher data generation costs, these can be effectively mitigated through the two-phase training strategy with BC pre-training and closed-loop fine-tuning.

Lastly, note that CAT-K with $K = |V|$ is equivalent to noise-free BC, since in this case rollouts follow the GT as closely as the available token book allows. On the other hand, for $K = 1$, CAT-K is equivalent to deterministically rolling out the policy by always choosing the most likely token. Consequently, for CAT-K, the hyperparameter K trades off following the policy (for $K = 1$) vs. following the GT (for $K = |V|$). In Fig. 5 we show how various choices of K impact the Average Displacement Error (ADE) between the CAT-K rollouts and the GT. The intuition behind CAT-K is to find, among all the top- K T-step rollouts (with a total of K^T possibilities), the one with the minimum ADE (minADE) to the GT. However, since directly solving this optimization problem is difficult, we use CAT-K as a greedy approximation to solve it sequentially.

5. Experiments

To show the broad relevance of CAT-K fine-tuning, we evaluate its performance on two different tasks using two different types of policy architecture, namely a *traffic simulation* task using a NTP policy and an *ego-motion planning* task using a policy parameterized as GMM. We base both tasks

on the widely used WOMD [5] and follow the simulation setup proposed in WOSAC [21], namely providing 1 second of history and generating rollouts of 8 seconds length.

5.1. Traffic Simulation

Metrics. In the traffic simulation task, we follow the WOSAC protocol: for each scenario, we generate 32 simulated rollouts *for all agents in the scene*, at 10Hz, and evaluate how well their distribution matches that of the human demonstrations in the data. This distributional matching is evaluated along three dimensions, namely kinematics (e.g., velocities and accelerations), interactions (e.g., collisions), and map alignment (e.g., off-road). All three are summarized as weighted averages in the “Realism Meta-Metric” (RMM), the key performance indicator of the WOSAC leaderboard. For more details please see [21]. We also report the minADE, which is not used in the RMM, but widely applied for motion prediction and policy evaluation.

Policy. We use SMART [35] as our policy architecture due to its strong performance and open-source implementation. Specifically, we use the SMART-tiny (7M) model and its open-sourced token vocabularies. The map polyline token vocabulary has a size of 1024, with each token representing a polyline consisting of 10 segments, each 0.5 meters long. The agent trajectory token vocabulary has a size of 2048, with each token representing a 0.5-second trajectory at 10Hz. SMART re-plans at 2Hz, hence the final output is at 10Hz when submitted to the WOSAC leaderboard. However, our experiments show that SMART’s open-sourced agent token vocabulary results in lower kinematic metrics. To address this issue, we reran the K-disk clustering [24] with a larger number of trajectories. By using our own agent token vocabulary, the performance on the WOSAC leaderboard is significantly improved (c.f. Tab. 1). Please note that all other experiments in this paper (i.e., except those in Tab. 1) still use SMART’s open-sourced token vocabularies. SMART uses Transformers [33] with query-centric representations [41, 46], and the distance metric $d(\cdot, \cdot)$ is the average Euclidean distance between the four pairs of corners of two bounding boxes. We disable SMART’s trajectory perturbation as it deteriorates performance on the WOSAC leaderboard (see [35] Tab. 4). We run BC pre-training for 32 epochs, each taking 1.7 hour, then continue with closed-loop supervised fine-tuning with CAT-32 for 10 epochs, each taking 2.6 hours. During fine-tuning, the map encoder is frozen to save GPU memory, allowing for a larger training batch size with negligible impact on performance. For both pre-training and fine-tuning, we use $8 \times$ A100 80GB GPUs with a total batch size of 80.

5.2. Ego-motion planning

Metrics. Unlike traffic simulation, in ego-motion planning only the ego-vehicle is controlled and evaluated. Moreover,

Leaderboard, test split Method	# model params	RMM ↑	RMM diff. to SMART-large	Kinematic metrics ↑	Interactive metrics ↑	Map-based metrics ↑	min ADE ↓
SMART-tiny fine-tuned w. CAT-K (ours)	7 M	0.7702	+0.0088	0.4931	0.8119	0.8749	1.3068
SMART-large [35]	102 M	0.7614	+0.0000	0.4786	0.8066	0.8648	1.3728
KiGRAS [43]	0.7 M	0.7597	−0.0014	0.4691	0.8064	0.8658	1.4384
SMART-tiny [35]	7 M	0.7591	−0.0023	0.4759	0.8039	0.8632	1.4062
FDriver-tiny	7 M	0.7584	−0.0030	0.4614	0.8069	0.8658	1.4475
SMART [35]	8 M	0.7511	−0.0103	0.4445	0.8050	0.8571	1.5447
BehaviorGPT [47]	3 M	0.7473	−0.0141	0.4333	0.7997	0.8593	1.4147
GUMP [9]	523 M	0.7431	−0.0183	0.4780	0.7887	0.8359	1.6041

Table 1. **Results on the WOSAC 2024 leaderboard [17]**. RMM stands for Realism Meta Metric, the key metric used for ranking. Note that on the public leaderboard [17] our method appears under the name “SMART-tiny-CLSFT” (Closed-Loop Supervised Fine-Tuning).

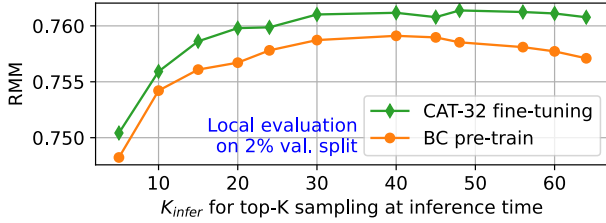


Figure 3. Influence of K_{infer} for inference-time top-K sampling.

the optimal policy is not the one that perfectly matches the GT trajectory *distribution*, but which minimizes planning metrics, such as collision rate, off-road rate, and ADE. For completeness, we also report RMM, applied only to ego trajectories, and minADE over 32 sampled rollouts. With the exception of minADE, and in contrast to the traffic simulation task, the rollouts for evaluation are sampled deterministically, by choosing the most-likely mode at each timestep.

Policy. To show that our fine-tuning approach can be applied to a more general class of policies beyond NTP, we parameterized the output of the ego-policy as GMM. Specifically, we replace the final layer of the SMART network with two heads, a classification head that predicts the mixture densities and a regression head that predicts the means of the Gaussian distributions with fixed standard deviations (similar to, e.g., [14]). Specifically, the GMM predicts 16 modes, each representing a 3-dimensional Gaussian distribution over the changes in x- and y-position and yaw heading. The final model comprises a total of 3.2M parameters. In contrast to the SMART traffic simulation policy that uses cross-entropy loss on all agents, the GMM-based ego-policy is trained using negative log-likelihood loss and only predicts actions for the ego-vehicle. When applying CAT-K rollout or top-K sampling to the GMM, we select the K most-likely modes of the GMM, and within each mode deterministically use the mean of the Gaussian. Please note that ego-motion planning is a separate experiment from traffic simulation; hence, the ego-policy is not fine-tuned on the traffic simulation policy. Also note that the ego-policy operates in a mid-to-end manner, taking tracking results and HD maps as inputs, rather than in a fully end-to-end fashion that maps raw sensor observations to actions.

5.3. Results

5.3.1. WOSAC leaderboard for traffic simulation

In Tab. 1 we compare our approach with other traffic simulation policies on the WOSAC leaderboard. Notably, all the top-ranking methods on the leaderboard are NTP policies trained via BC. Our SMART-tiny model using CAT-K fine-tuning outperforms the previous SoTA, SMART-large with 102M parameters, by a significant margin of +0.0088; and improves on SMART-tiny by +0.0111. As the first to perform closed-loop fine-tuning on the leaderboard, our approach improves all metrics and sets a new SoTA.

For fine-tuning, we chose CAT-32 after preliminary hyperparameter explorations, a choice that was later confirmed to perform well in our ablation studies (see Tab. 2), though CAT-K improved performance for a wide range of values of K . We also found that choosing a sufficiently high K_{infer} for top-K sampling during inference is important for a high RMM [9], and we chose $K_{\text{infer}} = 48$ for our leaderboard submission based on local validation results (see Fig. 3). In all of our experiments, we fixed the inference time sampling temperature to 1.0 and did not use top-p (nucleus) sampling [8], though we expect that tuning these hyperparameters can lead to further performance gains.

To ensure that the observed performance gains with CAT-K are not due to improved hyperparameter tuning, we also conduct a large-scale hyperparameter grid search for the baseline method SMART-tiny. This includes training for more epochs, adjusting the learning rate and learning rate scheduler, experimenting with various data augmentation and data pre-processing strategies, and using various values of K_{infer} for the inference-time top-K sampling. This allowed us to push the performance of the BC baseline to 0.7671 RMM on the test split (not shown in Tab. 1), but still falling significantly short of the performance of our CAT-K fine-tuning method (0.7702 RMM). Note that even comparing against this improved baseline, our method improves the performance more (+0.0031) than scaling up the model size by a factor of 14 to 102M parameters (+0.0023).

Qualitatively, our method can generate diverse and realistic behavior over long periods of time. As shown in Fig. 6, our method can handle the subtle interactions between traf-

Local val. split Method	Criterion of top ^K	K for top ^K	Sampled from	Next target	RMM ↑	Kinematic metrics ↑	Interactive metrics ↑	Map-based metrics ↑	min ADE ↓
BC pre-training	-	-	-	GT	0.7581	0.4512	0.8076	0.8697	1.3152
BC fine-tuning	-	-	-	GT	0.7590	0.4514	0.8096	0.8700	1.3039
Trajenglish's noisy tokenization	neg. dist.	5 [†]	neg. dist. [†]	GT [†]	0.7562	0.4469	0.8074	0.8673	1.3459
	neg. dist.	5 [†]	uniform	GT [†]	0.7554	0.4467	0.8069	0.8655	1.3404
	neg. dist.	32	neg. dist. [†]	GT [†]	0.7401	0.4174	0.7985	0.8493	1.6669
SMART's trajectory perturbation	neg. dist.	5 [†]	uniform [†]	RO [†]	0.7556	0.4440	0.8082	0.8661	1.3177
	neg. dist.	5 [†]	neg. dist.	RO [†]	0.7560	0.4469	0.8069	0.8673	1.3514
	neg. dist.	32	uniform [†]	RO [†]	0.7314	0.4158	0.7949	0.8300	1.5380
Top-5	prob	5	prob	GT	0.6478	0.3313	0.6847	0.7528	1.8802
Top-5 + distance filter	prob	5	prob	GT	0.6860	0.3356	0.7466	0.8083	1.7627
Top-5 + distance based sampling	prob	5	neg. dist.	GT	0.7058	0.3536	0.7579	0.8400	1.5848
Deterministic rollout	-	-	max-prob	GT	0.6361	0.3291	0.6845	0.7492	1.8695
CAT-5	prob	-	closest	GT	0.7423	0.4251	0.7917	0.8601	1.4677
CAT-16	prob	-	closest	GT	0.7604	0.4592	0.8082	0.8709	1.3372
CAT-32 (leaderboard)	prob	-	closest	GT	0.7616	0.4583	0.8105	0.8720	1.3105
CAT-40	prob	-	closest	GT	0.7617	0.4567	0.8101	0.8738	1.2998
CAT-64	prob	-	closest	GT	0.7602	0.4552	0.8098	0.8707	1.3028

Table 2. **Ablation study on WOSAC 2% validation split.** We compare different ways to fine-tune the same base mode (BC pre-training). "Sampled from" indicates how the action is sampled during fine-tuning, either based on the distance to the GT ("neg. dist", "uniform", "closest") or based on the model outputs ("prob", "max-prob"). Here dist. is the abbreviation of distance. RO stands for rollout, i.e., the next target action is computed based on the rollout, not the GT state. † indicates original hyperparameter choices of baseline algorithms.

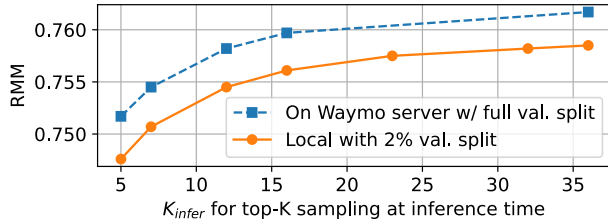


Figure 4. On server vs. local evaluation of SMART-tiny.

fic participants in a dense parking lot, which is arguably a more challenging scenario than intersections and highways for traffic simulation. Moreover, the behavior remains realistic at the end of the required simulation time of 8 seconds. Additional examples are in the supplementary videos.

5.3.2. Ablation studies on WOSAC

In Tab. 2 we provide a thorough ablation study. Due to the high cost of evaluation we use 2% of the validation split (880 out of 44097 scenarios). To verify the fidelity of this evaluation setting, we compare results with those on the full validation set in Fig. 4. We observe consistent differences indicating that our evaluation setting is reasonable. We begin with a SMART-tiny model trained with BC for 32 epochs (BC pre-training, row 1). All other models fine-tune the BC pre-training model for 5 epochs. During inference, we use top-K sampling with $K_{\text{infer}} = 40$ for all methods (the best K_{infer} for SMART-tiny according to Fig. 3).

In Tab. 2, CAT-K fine-tuning is the only method that significantly outperforms the BC pre-training model. Further fine-tuning with BC (row 2), using Trajenglish's noisy tokenization (rows 3-5), or SMART's trajectory perturbation

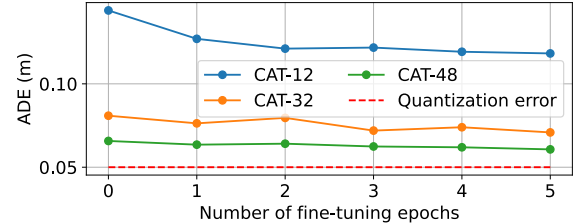


Figure 5. ADE between CAT-K rollouts and GT trajectories.

(rows 6-8) remain on par with the original model or even reduce its performance. Closed-loop fine-tuning with top-K rollouts instead of CAT-K rollouts during training significantly reduces performance (rows 9-12), even when rollouts close to the GT are selected for training by either filtering them based on the distance to the GT or sampling among them from a distance-dependent distribution. The observation that data augmentation and fine-tuning with top-K sampling cannot improve the RMM is consistent with prior works [19, 23, 24, 35] and the WOSAC leaderboard. Ablations on different values of K for top-K sampling and other hyperparameters are provided in the appendix.

Next, we ablate the value of K used for CAT-K rollout during fine-tuning (rows 13-17). Results indicate that the performance improvement is robust to the choice of K after a reasonable minimum value. As discussed in Sec. 4.3, the hyperparameter K in CAT-K determines how closely the policy follows the GT trajectory, in a way that is more robust than a distance based threshold, for which the optimal value varies strongly based on the situation (vehicle speed, proximity to other cars, etc.). To give more insight into how

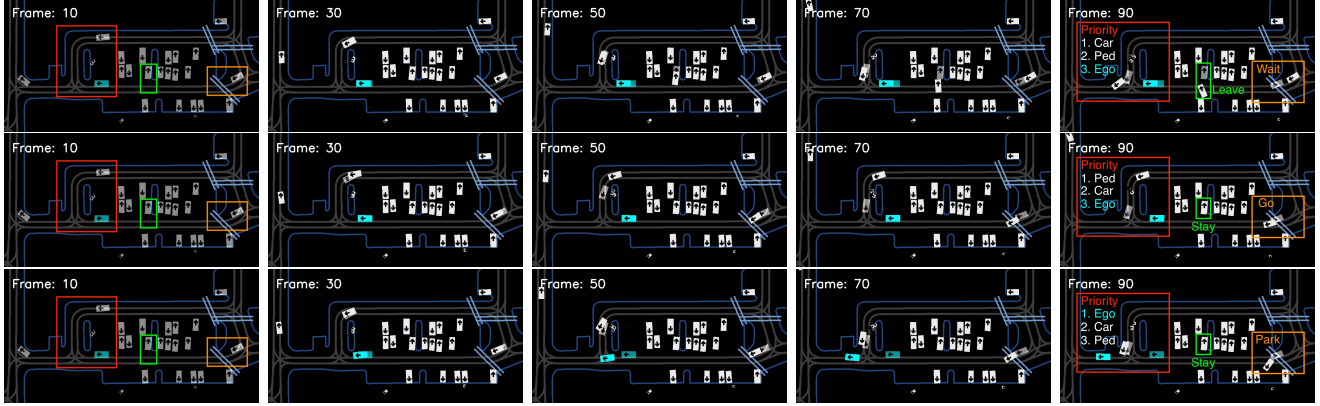


Figure 6. **Simulation results on WOSAC.** Our fine-tuned policy generates interesting and diverse behaviors rarely seen in prior works. Each row represents a different rollout of our model in the same scene. The transparent boxes show the GT agents in the dataset, while the solid boxes show the agents generated by our model. We highlight the agents within the red, green, and orange rectangles, across time steps and rollouts. The red rectangle shows different interactive negotiations emerging between a pedestrian and two vehicles. The green rectangle shows an initially parked vehicle, that leaves (row 1) or stays parked (row 2, 3). The orange rectangle shows a vehicle waiting (row 1) in front of a speed bump (visualized as two light-blue lines), proceeding (row 2), or entering a parking space (row 3).

Method (<i>Local val. split</i>)	Collision rate ↓	Off-road rate ↓	RMM ↑	ADE ↓	minADE ³² ↓
BC pre-training	0.0568	0.0053	0.8108	1.3623	1.3537
BC fine-tuning	0.0599	0.0058	0.8105	1.3520	1.3509
Deterministic rollout	0.0433	0.0138	0.8081	1.1799	0.7962
CAT-3	0.0422	0.0035	0.8169	1.3096	0.6912

Table 3. **Performance of ego policies on WOSAC with local evaluation on 2% validation split.** All models are fine-tuned for 5 epochs based on the BC pre-training model, which is trained for 32 epochs. We use deterministic rollout during inference and compute all metrics, except for the minADE³². For minADE³², we generate 32 rollouts by using top-3 sampling with a temperature of 1.0 to first sample the categorical distribution over the mixtures, then selecting the mean of the sampled Gaussian mixture.

K impacts rollouts, in Fig. 5, we inspect the average ADE between rollouts and GT trajectories over training epochs, for different K values. As expected, as $K \rightarrow |V|$, the ADE decreases towards the level of quantization error, induced by tokenization with a finite vocabulary size (dashed line). With more fine-tuning epochs the average ADE slightly reduces, highlighting how CAT-K fine-tuning improves the policy to follow all behavior modes more closely.

5.3.3. Fine-tuning a GMM-based ego-policy

Besides NTP traffic simulation policies, CAT-K fine-tuning also improves the performance of a GMM-based ego-policy with continuous action space. In Tab. 3 we compare our approach with fine-tuning using deterministic rollouts, as well as with continued BC. Our CAT-3 fine-tuning improves all metrics except for ADE, where deterministic rollout performs better. This is expected, as deterministic rollout aligns all modes towards the GT, resulting in mode averaging. While this reduces ADE, it negatively impacts other metrics. Additionally, CAT-K fine-tuning helps overcome the limitation of GMM trained with negative log-likelihood loss, which struggles to capture multimodality as effectively as NTP policies trained with cross-entropy loss. This is supported by the significant drop in minADE after fine-tuning.

6. Conclusion

In this paper, we introduce CAT-K rollouts, a closed-loop supervised fine-tuning technique for IL problems with highly multimodal demonstrations, such as traffic simulation. The CAT-K rollout approximately finds the rollout closest to the GT among likely rollouts of a policy, ensuring adherence to the policy while maintaining GT as a reliable reference for supervised learning. As the first method using closed-loop fine-tuning, it achieves the top spot on the WOSAC leaderboard.

In the future, we aim to incorporate modern sampling techniques into CAT-K rollouts, such as top-p sampling. Furthermore, instead of always selecting the closest token, we could loosen the “winner-takes-all” approach and sample from a group of nearby tokens. We also plan to explore CAT-K fine-tuning for a broader range of policy classes, such as variational autoencoders and diffusion models. Additionally, we want to apply it to other multimodal IL tasks, including end-to-end driving, motion generation for animation, and robot navigation and manipulation. Importantly, our results show that closed-loop supervised fine-tuning is a promising area of future research for policies trained in open-loop, such as the widely used NTP policies.

References

- [1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 1, 2
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1877–1901. Curran Associates, Inc., 2020. 1
- [3] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric M. Wolff, Alex H. Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. NuPlan: A closed-loop ml-based planning benchmark for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR) ADP3 Workshop*, 2021. 5
- [4] Yulong Cao, Boris Ivanovic, Chaowei Xiao, and Marco Pavone. Reinforcement learning with human feedback for realistic traffic simulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14428–14434. IEEE, 2024. 2
- [5] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, 2021. 5
- [6] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D’Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. *arXiv preprint arXiv:2104.00563*, 2021. 2
- [7] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016. 3
- [8] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020. 6
- [9] Yihan Hu, Siqi Chai, Zhenning Yang, Jingyu Qian, Kun Li, Wenxin Shao, Haichao Zhang, Wei Xu, and Qiang Liu. Solving motion planning tasks with a scalable generative model. In *European Conference on Computer Vision (ECCV)*, 2025. 1, 2, 6
- [10] 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
- [11] Zhiyu Huang, Zixu Zhang, Ameya Vaidya, Yuxiao Chen, Chen Lv, and Jaime Fernández Fisac. Versatile scene-consistent traffic scenario generation as optimization with diffusion. *arXiv preprint arXiv:2404.02524*, 2024. 2
- [12] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2445–2451. IEEE, 2022. 2, 3
- [13] Maximilian Igl, Punit Shah, Paul Mougin, Sirish Srinivasan, Tarun Gupta, Brandyn White, Kyriacos Shiarlis, and Shimon Whiteson. Hierarchical imitation learning for stochastic environments. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1697–1704. IEEE, 2023. 2
- [14] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2375–2384, 2019. 6
- [15] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9644–9653, 2023. 2
- [16] Longzhong Lin, Xuewu Lin, Kechun Xu, Haojian Lu, Lichao Huang, Rong Xiong, and Yue Wang. Revisit mixture models for multi-agent simulation: Experimental study within a unified framework, 2025. 2
- [17] Waymo LLC. Waymo open sim agent challenge (wosac) 2024 leaderboard. <https://waymo.com/open/challenges/2024/sim-agents/>, 2024. Accessed: 2025-03-14. 6, 2
- [18] Jack Lu, Kelvin Wong, Chris Zhang, Simon Suo, and Raquel Urtasun. Scenecontrol: Diffusion for controllable traffic scene generation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16908–16914. IEEE, 2024. 2
- [19] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Rebecca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7553–7560. IEEE, 2023. 1, 2, 7, 3
- [20] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7144–7153, 2019. 2
- [21] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nicholas Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, et al. The waymo open

- sim agents challenge. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024. 1, 5
- [22] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting multiple agent trajectories. *arXiv preprint arXiv:2106.08417*, 2021. 2
- [23] Zhenghao Peng, Wenjie Luo, Yiren Lu, Tianyi Shen, Cole Gulino, Ari Seff, and Justin Fu. Improving agent behaviors with rl fine-tuning for autonomous driving. In *European Conference on Computer Vision (ECCV)*, pages 165–181. Springer, 2025. 1, 2, 7
- [24] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajenglish: Traffic modeling as next-token prediction. In *International Conference on Learning Representations (ICLR)*, 2024. 2, 3, 4, 5, 7
- [25] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11763–11773, 2020. 2
- [26] Ethan Pronovost, Meghana Reddy Ganesina, Noureldin Hendy, Zeyu Wang, Andres Morales, Kai Wang, and Nick Roy. Scenario diffusion: Controllable driving scenario generation with diffusion. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:68873–68894, 2023. 2
- [27] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 1, 3
- [28] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8579–8590, 2023. 2
- [29] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:6531–6543, 2022. 2
- [30] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024. 2
- [31] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. TrafficSim: Learning to simulate realistic multi-agent behaviors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10400–10409, 2021. 2
- [32] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1
- [33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017. 5
- [34] Arun Venkatraman, Martial Hebert, and J Bagnell. Improving multi-step prediction of learned time series models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2015. 2, 4, 5
- [35] Wei Wu, Xiaoxin Feng, Ziyang Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time simulation via next-token prediction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 1, 2, 3, 4, 5, 6, 7
- [36] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2929–2936. IEEE, 2023. 2
- [37] Chris Zhang, James Tu, Lunjun Zhang, Kelvin Wong, Simon Suo, and Raquel Urtasun. Learning realistic traffic agents in closed-loop. In *Conference on Robot Learning (CoRL)*, 2023. 1, 2, 3
- [38] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016. 2
- [39] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2
- [40] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. Trafficbots: Towards world models for autonomous driving simulation and motion prediction. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1522–1529. IEEE, 2023. 2
- [41] Zhejun Zhang, Alexander Liniger, Christos Sakaridis, Fisher Yu, and Luc V Gool. Real-time motion prediction via heterogeneous polyline transformer with relative pose encoding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 2, 5
- [42] Zhejun Zhang, Christos Sakaridis, and Luc Van Gool. Trafficbots v1. 5: Traffic simulation via conditional vaes and transformers with relative pose encoding. *arXiv preprint arXiv:2406.10898*, 2024. 2
- [43] Jianbo Zhao, Jiaheng Zhuang, Qibin Zhou, Taiyu Ban, Ziyao Xu, Hangning Zhou, Junhe Wang, Guoan Wang, Zhiheng Li, and Bin Li. Kigras: Kinematic-driven generative model for realistic agent simulation. *arXiv preprint arXiv:2407.12940*, 2024. 1, 2, 6
- [44] Ziyuan Zhong, Davis Rempe, Yuxiao Chen, Boris Ivanovic, Yulong Cao, Danfei Xu, Marco Pavone, and Baishakhi Ray. Language-guided traffic simulation via scene-level diffusion. *arXiv preprint arXiv:2306.06344*, 2023. 2
- [45] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3560–3566. IEEE, 2023. 2
- [46] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 17863–17873, 2023. [2](#), [5](#)

- [47] 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. [1](#), [6](#), [2](#)