

Closed-Loop Supervised Fine-Tuning of Tokenized Traffic Models

Supplementary Material

7. Supplementary videos

We provide the following videos as part of the supplementary material. All videos are carefully edited and thoroughly annotated, offering additional qualitative results to support our paper. Please note that all videos are without sound.

- `tldr_highlights.mp4`: The most interesting behaviors generated by our model. If you are short on time, we recommend watching this video.
- `parking_lot.mp4`: The busy parking lot scenario shown in the main paper.
- `ped_cyc.mp4`: Interesting behaviors for pedestrians and cyclists.
- `lane_changing.mp4`: Lane-changing scenario involving interactions between multiple agents.
- `bc_compounding_error.mp4`: This video provides a real-world example of covariate shift, demonstrating how the BC policy suffers from compounding errors.
- `more_behaviors.mp4`: If you have time, enjoy additional interesting behaviors generated by our model. These include exiting parking lots, making U-turns, stopping at stop signs, obeying traffic lights, near-accident scenarios, and more.

Additional videos are available on the homepage of this paper¹.

8. Full algorithm

To complement the algorithm in the main paper, we provide the detailed and complete algorithm for BC pre-training, followed by CAT-K fine-tuning, in Algorithm 2. Our BC pre-training follows Trajenglish and SMART, which are identical when no data augmentation is applied.

9. Implementation details

Our implementation is based on the open-source repository of SMART². We made the following changes, as we believe they may improve performance:

- Preprocessing agent trajectories using linear interpolation.
- Adding additional HD map elements, such as speed bumps.
- Setting the learning rate decay to 1% instead of to 0%.
- Resolving duplicated tokens in the action token vocabulary.
- Removing data augmentation applied to the tokenization of map polylines and agent trajectories, as it only im-

Algorithm 2 BC pre-training and CAT-K fine-tuning

```
1: Input: Policy  $\pi_\theta$ , action token vocabulary  $V$ , dataset  $\mathcal{D}$ 
2: Pre-train  $\pi_\theta(\mathbf{c}_t \mid \hat{\mathbf{h}}_t, \mathcal{M})$  with BC until convergence
3: repeat ▷ BC pre-training
4:   Sample a traffic scenario  $\{\hat{\mathbf{s}}_{0:T}, \mathcal{M}\}$ .
5:   Init rollout state  $\mathbf{s}_0 = \hat{\mathbf{s}}_0$ . ▷ Sequential tokenization following Trajenglish
6:   for  $t$  in  $[0, \dots, T-1]$  do
7:     Tokenization for each agent  $i \in \{1, \dots, N\}$ 
        $c_t^i = \arg \min_{c \in \{1, \dots, |V|\}} d(f(s_t^i, x_c) - s_{t+1}^i)$ .
8:     Save  $\mathbf{c}_t$  as the GT labels  $\hat{\mathbf{c}}_t$ .
9:     Get next rollout state  $s_{t+1}^i$ . (Eq. 3)
10:  end for
11:  Batched forward pass with causal masking
        $\pi_\theta(\mathbf{c}_{1:T} \mid \hat{\mathbf{c}}_{0:T-1}, \mathcal{M})$ .
12:  Update  $\theta$  by minimizing the cross entropy loss Eq. 5
       with GT labels  $\hat{\mathbf{c}}$ .
13: until convergence
14: repeat ▷ Closed-loop supervised fine-tuning
15:   Sample a traffic scenario  $\{\hat{\mathbf{s}}_{0:T}, \mathcal{M}\}$ 
16:   Init rollout state  $\mathbf{s}_0 = \hat{\mathbf{s}}_0$  ▷ CAT-K Rollout
17:   for  $t$  in  $[0, \dots, T-1]$  do ▷  $T$  steps
18:     for  $i$  in  $[1, \dots, N]$  do ▷  $N$  agents
19:       One step forward pass policy  $\pi$  with previous rollout states.
20:       Get action index for rollout  $c_t^i$ . (Eq. 1)
21:       Get next rollout state  $s_{t+1}^i$ . (Eq. 3)
22:       Compute target  $\hat{c}_t^i$ . (Eq. 4)
23:       Save forward-pass output logits of this step
       for later training.
24:     end for
25:   end for
26:   Update  $\theta$  by minimizing  $\mathcal{L}_\theta(\mathbf{s}_{0:T}, \hat{\mathbf{c}}_{1:T}, \mathcal{M})$ . (Eq. 5)
27: until convergence
```

proves performance for zero-shot transfer from NuPlan to WOSAC but significantly decreases performance when the model is both trained and validated on WOSAC [35].

Apart from these changes, we use the same model architecture, hyperparameters, and other settings as provided in the open-source repository. While SMART-tiny was originally trained on 32 NVIDIA TESLA V100 GPUs for 23 hours, we use 8 NVIDIA A100 GPUs for all our experiments. Our reproduced SMART-tiny model is trained for 32 hours (32 epochs) with BC. We finetune this BC baseline model with CAT-K rollout for 25 hours (10 epochs) to obtain our final model, SMART-tiny-CLSFT, which is sub-

¹<https://zhejz.github.io/catk>

²<https://github.com/rainmaker22/SMART>

Leaderboard, test split Method	Realism meta metric↑	Linear speed likeli.↑	Linear acc. likeli.↑	Angular speed likeli.↑	Angular acc. likeli.↑	Distance to nearest object likeli.↑	Collision likeli. ↑	Time to collision likeli.↑	Distance to road edge likeli.↑	Offroad likeli. ↑	min ADE ↓
SMART-tiny-CLSFT (ours)	0.7702	0.3868	0.4066	0.5201	0.6589	0.3923	0.9702	0.8356	0.6814	0.9523	1.3068
UniMM [16]	0.7683	0.3836	0.4159	0.5168	0.6491	0.3911	0.9679	0.8347	0.6791	0.9506	1.2947
SMART-large [35]	0.7614	0.3786	0.4134	0.4952	0.6270	0.3872	0.9632	0.8346	0.6761	0.9403	1.3728
KiGRAS [43]	0.7597	0.3704	0.3784	0.4962	0.6314	0.3867	0.9619	0.8373	0.6723	0.9431	1.4383
SMART-tiny [35]	0.7591	0.3733	0.4082	0.4945	0.6277	0.3835	0.9601	0.8338	0.6709	0.9401	1.4062
FDriver-tiny	0.7584	0.3661	0.3669	0.4876	0.6248	0.3840	0.9641	0.8366	0.6688	0.9446	1.4475
SMART [35]	0.7511	0.3646	0.4057	0.4231	0.5845	0.3769	0.9655	0.8318	0.6590	0.9363	1.5447
BehaviorGPT [47]	0.7473	0.3615	0.3365	0.4806	0.5544	0.3834	0.9537	0.8308	0.6702	0.9349	1.4147
GUMP [9]	0.7431	0.3569	0.4111	0.5089	0.6353	0.3707	0.9403	0.8276	0.6686	0.9028	1.6031
SMART-tiny (we reproduced not on the public leaderboard)	0.7671	0.3781	0.4026	0.5183	0.6571	0.3899	0.9653	0.8346	0.6788	0.9507	1.3587

Table 4. **Results on the WOSAC 2024 leaderboard [17]** accessed on March 14, 2025. Realism Meta Metric is the key metric used for ranking. All other metrics contribute to the realism meta metric, except for the minADE, which has no effect on the ranking. Note that on the public leaderboard [17] our method appears under the name “SMART-tiny-CLSFT” (Closed-Loop Supervised Fine-Tuning), and our reproduced SMART-tiny is not published to the public leaderboard. Here likeli. is the abbreviation of likelihood, and acc. stands for acceleration.

mitted to the leaderboard. Performing inference and generating the submission file for the validation split (44,097 scenarios) together requires 3 hours, the same as for the test split (44,920 scenarios).

10. Additional experiment results

10.1. WOSAC leaderboard

In Tab. 4 we provide the results of all metrics for leading entries on the WOSAC leaderboard³, accessed before the camera-ready deadline of CVPR 2025 (March 14, 2025). We also provide the results of our reproduced SMART-tiny, trained via BC and used as the starting point for our fine-tuning experiments. Our method achieves the best performance across nearly all metrics. Notably, a concurrent work, UniMM [16] (previously called MM-GPT), has recently made multiple submissions and achieved a high ranking on the leaderboard. Nevertheless, our method still outperforms the best UniMM model in the majority of the metrics.

10.2. Ablation

In Tab. 5, we provide additional ablation studies we conducted. Our method, CAT-K fine-tuning with $K = 32$, achieves the overall best performance. Only on the map-based metrics we are slightly outperformed by “Trajenglish top-5, sampled w/ policy prob.”, but the difference is insignificant. The “sampled w/ policy prob.” version of Trajenglish’s noisy tokenization and SMART’s trajectory perturbation is an on-policy variation of the original data augmentation, where the K closest-to-GT tokens are sampled using

the probability predicted by the policy rather than using the negative distance. These on-policy versions perform better than the off-policy data augmentation, but their performance is still worse than our CAT-K fine-tuning. For top-K sampling, adding distance based filtering or distance based sampling improves the performance, but they still cannot match the performance of our method. For the original versions of Trajenglish’s and SMART’s data augmentation, a thorough search of the hyperparameters confirms the conclusion drawn in the Trajenglish and SMART papers: Off-policy data augmentation does not significantly improve the performance on the WOSAC leaderboard.

Our CAT-K rollout can be seen as a special case of top-K with distance based sampling, where a very low temperature is used in the distance-based sampling, ensuring that the closest-to-GT token is selected deterministically. For example, “Top-32 + distance based sampling” with a sampling temperature $\tau \rightarrow 0$ is equivalent to CAT-32 rollout.

10.3. GMM-based ego policy

In Tab. 6 we present additional ablation studies for training and fine-tuning the GMM ego policy. Inspired by the training strategy used in multimodal motion prediction, we experimented with applying hard-assignment to train the BC policy, aiming to mitigate the mode-averaging problem in the GMM. Specifically, at each time step and for each agent, we train only the Gaussian mixture component that is closest to the GT, leaving the other components untrained. However, this approach did not work, and the training diverged. We then investigate the impact of Trajenglish’s and SMART’s data augmentation on fine-tuning the ego policy. The results indicate that the effectiveness of these off-policy data augmentation methods is marginal: the

³<https://waymo.com/open/challenges/2024/sim-agents/>

<i>Local val. split</i> 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
CAT-32 (submitted to leaderboard)	prob	-	closest	GT	0.7616	0.4583	0.8105	0.8720	1.3105
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.	16	neg. dist.	GT	0.7486	0.4336	0.8031	0.8585	1.4811
	neg. dist.	16	uniform	GT	0.7481	0.4315	0.8033	0.8581	1.5012
	neg. dist.	32	neg. dist.	GT	0.7401	0.4174	0.7985	0.8493	1.6669
	neg. dist.	32	uniform	GT	0.7412	0.4177	0.7987	0.8521	1.6715
	neg. dist.	64	neg. dist.	GT	0.7303	0.4005	0.7906	0.8413	1.9083
	neg. dist.	64	uniform	GT	0.7295	0.3994	0.7890	0.8416	1.9307
SMART’s trajectory perturbation	neg. dist.	5	neg. dist.	RO	0.7560	0.4469	0.8069	0.8673	1.3514
	neg. dist.	5	uniform	RO	0.7553	0.4468	0.8074	0.8647	1.3566
	neg. dist.	16	neg. dist.	RO	0.7495	0.4329	0.8035	0.8609	1.4958
	neg. dist.	16	uniform	RO	0.7478	0.4317	0.8029	0.8576	1.4890
	neg. dist.	32	neg. dist.	RO	0.7407	0.4190	0.7985	0.8503	1.6472
	neg. dist.	32	uniform	RO	0.7403	0.4179	0.7986	0.8497	1.6568
	neg. dist.	64	neg. dist.	RO	0.7309	0.4012	0.7917	0.8411	1.8701
	neg. dist.	64	uniform	RO	0.7284	0.3962	0.7879	0.8417	1.9574
Top-16	prob	16	prob	GT	0.6439	0.3309	0.6912	0.7619	1.8744
Top-16 + distance filter	prob	16	prob	GT	0.6904	0.3375	0.7489	0.8169	1.7991
Top-16 + distance based sampling	prob	16	neg. dist.	GT	0.7233	0.3675	0.7808	0.8528	1.4876
Top-32	prob	32	prob	GT	0.6395	0.3324	0.6882	0.7522	1.8961
Top-32 + distance filter	prob	32	prob	GT	0.6950	0.3400	0.7560	0.8193	1.8194
Top-32 + distance based sampling	prob	32	neg. dist.	GT	0.7229	0.3663	0.7843	0.8477	1.6470
Top-64	prob	64	prob	GT	0.6381	0.3318	0.6846	0.7535	1.9117
Top-64 + distance filter	prob	64	prob	GT	0.6979	0.3407	0.7590	0.8234	1.8172
Top-64 + distance based sampling	prob	64	neg. dist.	GT	0.7208	0.3660	0.7823	0.8446	1.7260
Trajenglish top-5, sampled w/ policy prob.	neg. dist.	5	prob	GT	0.7596	0.4513	0.8089	0.8723	1.3116
Trajenglish top-32, sampled w/ policy prob.	neg. dist.	32	prob	GT	0.7526	0.4320	0.8069	0.8659	1.3569
SMART top-5, sampled w/ policy prob.	neg. dist.	5	prob	RO	0.7589	0.4510	0.8085	0.8709	1.3135
SMART top-32, sampled w/ policy prob.	neg. dist.	32	prob	RO	0.7580	0.4533	0.8093	0.8661	1.3325

Table 5. **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. RMM stands for the realism meta metric of WOSAC.

RMM shows slight improvement, while the collision and off-road rates are marginally worse. Next, we explore the use of top-K sampling for fine-tuning the BC policy. As expected, top-K sampling alone does not work. However, when combined with distance-based filtering or sampling, top-K sampling can significantly enhance the BC policy’s performance, achieving results comparable to those of our CAT-K fine-tuning approach. This justifies the effectiveness of this approach when the expert demonstrations are generally well-behaved and less diverse, which is consistent with prior work that applies this sampling strategy for fine-tuning to only vehicles [19], often within the context of highway scenarios [37]. Compared to top-K sampling with distance-based filtering or sampling, our method significantly outperforms in off-road rate and minADE, while other metrics remain on par. Overall, fine-tuning with CAT-

K rollout achieves the best performance, with peak performance at $K = 2$ or $K = 3$, which aligns with the fact that the ego vehicle’s behavior is less multimodal. For traffic simulations where demonstrations are highly multimodal and involve various traffic participants (vehicles, pedestrians, and cyclists) whose behaviors do not necessarily obey traffic rules, the advantage of our CAT-K rollout becomes more significant.

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 w/ hard-assignment (training diverged)	0.1574	0.0637	0.7409	5.3507	5.3447
Trajectory noisy tokenization ($K = 3$, neg. dist., GT)	0.0611	0.0057	0.8117	1.3563	1.3575
SMART trajectory perturbation ($K = 3$, uniform, RO)	0.0590	0.0057	0.8118	1.3713	1.3771
Top-3	0.0415	0.0140	0.8072	1.2004	0.8249
Top-3 + distance filter	0.0409	0.0076	0.8128	1.1639	0.8577
Top-3 + distance based sampling	0.0410	0.0070	0.8163	1.3245	0.7610
CAT-1 (Deterministic rollout)	0.0433	0.0138	0.8081	1.1799	0.7962
CAT-2	0.0437	0.0038	0.8147	1.5117	0.6323
CAT-3	0.0422	0.0035	0.8169	1.3096	0.6912
CAT-4	0.0500	0.0035	0.8137	1.5699	1.4840
CAT-8	0.0771	0.0045	0.8050	1.6775	1.6704

Table 6. **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. RMM stands for the realism meta metric of WOSAC.