TrafficBots V1.5: Traffic Simulation via Conditional VAEs and Transformers with Relative Pose Encoding

Zhejun Zhang
ETH Zürich
zhejun.zhang@vision.ee.ethz.ch

Christos Sakaridis
ETH Zürich
csakarid@vision.ee.ethz.ch

Luc Van Gool
ETH Zürich, INSAIT Sofia
vangool@vision.ee.ethz.ch

Abstract

In this technical report we present TrafficBots V1.5, a baseline method for the closed-loop simulation of traffic agents. TrafficBots V1.5 achieves baseline-level performance and a 3rd place ranking in the Waymo Open Sim Agents Challenge (WOSAC) 2024. It is a simple baseline that combines TrafficBots, a CVAE-based multi-agent policy conditioned on each agent's individual destination and personality, and HPTR, the heterogeneous polyline transformer with relative pose encoding. To improve the performance on the WOSAC leaderboard, we apply scheduled teacher-forcing at the training time and we filter the sampled scenarios at the inference time. The code is available at: https://github.com/zhejz/TrafficBotsV1.5

1. Introduction

The problem of closed-loop multi-agent traffic simulation can be addressed by learning a policy for each traffic participants. Specifically, at each time step, the policy predicts the next action of each agent, given the historical observations form previous time steps, including the map, traffic lights and agent trajectories. Based on TrafficBots [20], the TrafficBots V1.5 policy is shared by all agents. Different behaviors are generated by conditioning the policy on the individual destination and personality of each agent. In contrast to the SceneTransformer [9] network architecture and input representation, which are not rotation and translation invariant, TrafficBots V1.5 uses the pairwise-relative representation and the HPTR [21] architecture. This greatly improves the accuracy of TrafficBots without sacrificing its efficiency and scalability. Moreover, instead using a recurrent neural network (RNN) to encode the temporal axis, TrafficBots V1.5 uses stacked historical observation as input, such that its architecture is solely based on Transformers [14].

1.1. TrafficBots

TrafficBots [20] is a multi-agent policy built upon motion prediction and end-to-end (E2E) driving. Compared to previous data-driven traffic simulators [13], TrafficBots demonstrates superior configurability and scalability. To generate configurable behaviors, for each agent TrafficBots introduces a destination as navigational information, and a time-invariant latent personality that specifies the behavioral style. Unlike the goal which depends on the prediction horizon and hence leads to causal confusions, the destination approximates the output of a navigator which is available in the problem formulation of E2E driving [19]. Importantly, the destination indicates where the agent wants to reach eventually, i.e., not necessarily at a specific future time step. In order to capture the diverse behaviors from human demonstrations, the personality is learned using the conditional variational autoencoder (CVAE) [12] following prior works on multi-modal motion prediction [2]. To ensure the scalability, TrafficBots uses the scene-centric representation [9] and presents a new scheme of positional encoding for angles, allowing all agents to share the same vectorized context and the use of an architecture based on Transformers. However, due to the lack of rotation and translation invariance induced by the scene-centric representation, TrafficBots does not achieve superior performance compared to methods using the agent-centric representation.

1.2. HPTR

Depending on how the coordinate system of the vectorized representation is selected, motion prediction methods fall into three categories: agent-centric [8], scene-centric [9] and pairwise-relative [3]. While agent-centric methods achieve top accuracy but lack scalability, and scene-centric methods show superior scalability but suffer from poor accuracy, the pairwise-relative methods get the best of both worlds. However, previous pairwise-relative methods are mostly using graph neural networks [3], which are often less efficiently implemented on the graphics processing unit (GPU) compared to Transformers with dot-product attention. To address this problem, a novel attention module called **K**-nearest **N**eighbor **A**ttention with **R**elative **P**ose **E**ncoding (KNARPE) is introduced in [21], which allows the pairwise-relative representation to be used by Transformers. KNARPE projects the relative pose encoding (RPE) and adds them to the keys and values to obtain $\mathbf{z}_i$, the output of letting token i attend to its K nearest neighbors κ_i^K,

$$\mathbf{z}_i = \text{KNARPE}\left(\mathbf{u}_i, \mathbf{u}_j, \mathbf{r}_{ij} \mid j \in \kappa_i^K\right) \tag{1}$$

$$= \sum_{j\in\kappa_i^K} \alpha_{ij}\left(\mathbf{u}_j\mathbf{W}^v + \mathbf{b}^v + \text{RPE}(\mathbf{r}_{ij})\hat{\mathbf{W}}^v + \hat{\mathbf{b}}^v\right) \tag{2}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k\in\kappa_i^K}\exp(e_{ik})} \tag{3}$$

$$e_{ij} = \frac{(\mathbf{u}_i\mathbf{W}^q + \mathbf{b}^q)(\mathbf{u}_j\mathbf{W}^k + \mathbf{b}^k + \text{RPE}(\mathbf{r}_{ij})\hat{\mathbf{W}}^k + \hat{\mathbf{b}}^k)}{\sqrt{D}} \tag{4}$$

where $\mathbf{u}_i, \mathbf{u}_j$ are the local attributes of token i and j represented in their local coordinates, $\mathbf{r}_{ij}$ is the relative pose between token i and j, α_{ij} are the attention weights, e_{ij} are the logits, $W^{\{q,k,v\}}, b^{\{q,k,v\}}$ are the learnable projection matrices and biases for query, key and value, and $\hat{W}^{\{k,v\}}, \hat{b}^{\{k,v\}}$ are the learnable projection matrices and biases for RPE. The RPE is defined as follows:

$$\text{RPE}(\mathbf{r}_{ij}) = \text{concat}(\text{PE}(x_{ij})\text{PE}(y_{ij}), \text{AE}(\theta_{ij})) \tag{5}$$

$$\text{PE}_{2i}(x) = \sin(x\cdot\omega^{\frac{2i}{D}}) \tag{6}$$

$$\text{PE}_{2i+1}(x) = \cos(x\cdot\omega^{\frac{2i}{D}}) \tag{7}$$

$$\text{AE}_{2i}(\theta) = \sin\left(\theta\cdot(i+1)\right) \tag{8}$$

$$\text{AE}_{2i+1}(\theta) = \cos\left(\theta\cdot(i+1)\right) \tag{9}$$

$$i \in \{0,\dots,D/2-1\}, \tag{10}$$

where $(x_{ij}, y_{ij}, \theta_{ij})$ are the 2D location and yaw heading of token j represented in the coordinate of token i, ω is the base frequency, and D is the embedding dimension.

Based on KNARPE, a pure Transformer-based framework called **H**eterogeneous **P**olyline **T**ransformer with **R**elative pose encoding (HPTR) [21] is presented, which uses a hierarchical architecture to enable asynchronous token update and avoid redundant computations. By sharing contexts among traffic agents and reusing the unchanged contexts in driving scenarios, HPTR is as efficient as scene-centric methods, while performing on par with state-of-the-art agent-centric methods on marginal motion prediction tasks. Since our method TrafficBots V1.5 is entirely based on KNARPE and HPTR, we refer the readers to the HPTR [21] paper for more details about the mathematical formulation and the network architecture.

2. Method

TrafficBots V1.5 updates TrafficBots [20] with the HPTR [21] input representation and network architecture. This section describes the changes we have made while combining TrafficBots and HPTR.

2.1. Architecture

The network architecture of TrafficBots V1.5 is illustrated in Fig. 1. We make minimal changes while applying HPTR to TrafficBots. We remove the temporal RNN from TrafficBots, and follow HPTR to use stacked historical observations as input and PointNet [4, 10] to aggregate the temporal axis. The policy network, the personality predictor, and the destination predictor of TrafficBots are now all based on the pairwise-relative representation and KNARPE attention module. We keep the intra-map, enhance-traffic-light and enhance-agent Transformers of HPTR. Following TrafficBots, multi-modal outputs are generated by conditioning the policy on each agent's individual destination and personality. Therefore, the anchors and the anchor-to-all Transformer of HPTR are discarded. Instead of a learnable prior personality of TrafficBots, we use a standard Gaussian for the prior personality. We also tried to add a traffic light state predictor, but its accuracy was not good enough to improve the overall simulation performance on the leaderboard.

2.2. Training

We found two techniques that improve the performance of TrafficBots V1.5. Firstly, we adopt a larger free nats [5] equals to 1.0 for the KL-divergence between the posterior and the prior personality. This allows the posterior personality to encode more information. Secondly, we use scheduled sampling [1] and apply teacher-forcing to 30% agents at the beginning of the training, i.e., these agents will be trained via open-loop behavior cloning. The percentage of teacher-forcing decreases linearly to 0 during the training. Due to limited computational resources, we train our models for 5 days on 4 NVIDIA RTX 4090 GPUs. Training for longer time or using more GPUs should improve the performance further. We use a total batch size of 8 and we batch over scenarios. Each scenario contains 91 time steps and a maximum of 64 agents. Following TrafficBots, the training uses back-propagation through time and the training loss includes the following terms:

1. Reconstruction loss that trains the model to reconstruct the ground-truth (GT) states using the posterior personality and the GT destination. It is a weighted sum of:
   - A smoothed L1 loss between the predicted and the GT (x, y) positions.

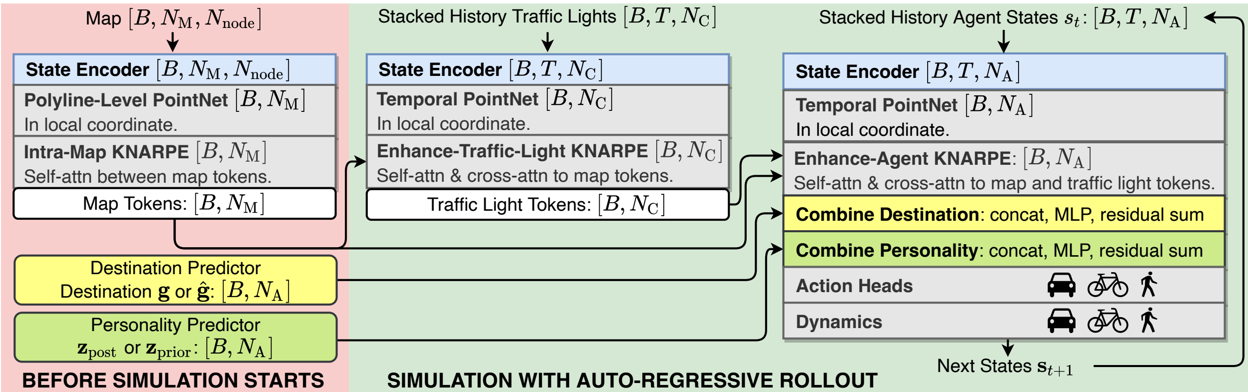

Figure 1. **Network architecture of TrafficBots V1.5**. In the brackets are the tensor shapes, where the hidden dimensions are omitted for conciseness. B is the batch size, which is also the number of episodes. $N_\mathrm{M}, N_\mathrm{C}, N_\mathrm{A}$ are, respectively, the number of map polylines, traffic light polylines and agent trajectories. N_node is the number of segments in each polyline. T is the length of the stacked historical observations. The destination predictor and personality predictor are not visualized. They have a similar structure to the policy network.

- A cosine distance between the predicted and the GT yaw heading.
- A smoothed L1 loss between the predicted and the GT velocities.

2. The KL divergence between the posterior and the prior personality. We use free nats [5] to clip the KL divergence, i.e., if $KL(\mathbf{z}_\mathrm{post}, \mathbf{z}_\mathrm{prior})$ is smaller than the free nats, then the KL loss is not applied.
3. The cross entropy loss for destination classification. Since the GT destination is a single class, this loss boils down to a maximum likelihood loss, i.e., the destination distribution is trained to maximize the log-likelihood of the polyline index of the GT destination.

During training, we auto-regressively rollout the policy using the GT destination and the posterior personality. Before the auto-regressively rollout, we use the reparameterization trick to sample the posterior personality, such that it can be trained to reconstruct the GT trajectories. For 10% of the episodes we rollout with the prior personality instead of the posterior personality. During the auto-regressively rollout, we take the mean of the actions such that the gradient can be back-propagated through the differentiable vehicle dynamics. We do not apply a loss that encourages collision avoidance [3, 6, 18] because this will bias the model, even though it could improve the performance on the leaderboard.

2.3. Inference

Instead of bias the model directly, we apply a milder approach to bias the model's outputs towards safer behavior and hence improve the collision-based metrics on the WOSAC leaderboard. Specifically, we sample 128 scenarios at the inference time and select 32 scenarios that contain the least collision events. To sample a scenario, we first sample the personality and destination for each agent, after that we start the auto-regressive rollout. We use the mean of the predicted action distribution, hence the rollout is completely deterministic given the sampled personality and destination. Except the episode filtering, we do not apply any other post-processing or model ensembling techniques. At the inference time, HPTR allows different types of tokens to be updated asynchronously at different frequency. Therefore, an accurate analysis of FLOPS turns out to be difficult. For simplicity, we only profile the map encoder, which is the computationally heaviest module in our model. For a single episode, the map encoder uses 20 GFLOPS. Since HPTR allows the map features to be cached and reused during the rollout, the map is encoded only once before the simulation starts. Other modules, e.g. the personality encoder, the traffic light encoder and the policy, are computationally much lighter than the map encoder. Based on the FLOPS of the map encoder, we estimate that each iteration of the auto-regressive policy rollout should require approximately an order of magnitude fewer FLOPS, i.e., around 2 GFLOPS.

2.4. Implementation details

Overall, we use a hidden dimension of 128. We use Transformer with pre-layer normalization, and the attention module has 4 heads and a feed-forward dimension of 512. The sampling rate is fixed to 10 FPS at both the training and inference time. At each time step, our method predicts the actions one step ahead, based on the previous observations. We do not differentiate between the ego-agent, i.e., the autonomous vehicle agent, and other agents. Similar to TrafficBots, TrafficBots V1.5 allows agents that are not controlled by itself. Since TrafficBots V1.5 predicts the action for all observed agents, the actions will still be predicted for those agents, but then these actions will be discarded; the behavior of those agents will be overridden by some other control modules, e.g. an AV planner software or log-reply. Each episode contains at most 1024 map tokens, 128 traf-

Table 1. Results on the WOSAC leaderboard 2024 [16].

Method name	Realism meta metric ↑	Kinematic metrics ↑	Interactive metrics ↑	Map-based metrics ↑	minADE ↓
SMART-large [17]	0.7564	0.4769	0.7986	0.8618	1.5501
BehaviorGPT [22]	0.7473	0.4333	0.7997	0.8593	1.4147
GUMP	0.7431	0.4780	0.7887	0.8359	1.6041
model_predictive_submission	0.7417	0.4182	0.7942	0.8591	1.4842
MVTE [15]	0.7302	0.4503	0.7706	0.8381	1.6770
VBD	0.7200	0.4169	0.7819	0.8137	1.4743
TrafficBots V1.5 (Ours)	0.6988	0.4304	0.7114	0.8360	1.8825
cogniBot v1.5	0.6288	0.3293	0.7129	0.6918	N/A
linear_extrapolation_baseline	0.3985	0.2253	0.4327	0.4533	7.5148

fic light tokens and 64 agent tokens, where invalid tokens are masked. Each map token corresponds to a polyline consisting of up to 20 segments, each 1 meter in length. Each traffic light token corresponds to the polyline it associated to and the traffic light state. Each agent token corresponds to an agent trajectory. We use a sliding window approach and stack the historical observations from the last 11 steps. For the Transformers with K-nearest-neighbor attention, each map token attends to the 32 nearest map tokens. Each traffic light token attends to 24 nearest map tokens and 24 nearest traffic light tokens. Each agent token attends to 64 map tokens, 25 traffic light tokens and 25 agent tokens in its proximity. More details about the network architecture can be found in the open-source repository of TrafficBots V1.5, as well as in the TrafficBots [20] and HPTR [21] papers. Our model does not take the z axis, i.e., the altitude dimension, into account. During inference, we assume that the z dimension of agent trajectories remains constant and is equal to its last observed value.

3. Results

We have not run any ablation studies for our method. However, we would like to point out some promising directions for ablations, including the discrete action space, weights of different losses, and the collision avoidance loss. We have done a manual and rough parameter tuning for our model. The performance of our method on the WOSAC leaderboard [16] is shown in Table 1. More details about the challenge and the metrics can be found in the WOSAC paper [7]. Our method achieves baseline-level performance in terms of the realism meta metric, which is a weighted sum of other metrics except the minADE. We apply a unicycle model for the dynamics of all types of agents and select the parameters heuristically. This allows our simulation to generate smooth trajectories, but it also affects the kinematic metrics negatively. Our TrafficBots V1.5 is outperformed by other methods in terms of interactive metrics which involve collision avoidance. We believe adding a loss that encourages collision avoidance would help, but it is controversial if a collision-free simulation would be useful for the development of autonomous driving algorithms. In terms of map-based metrics that consider off-road driving, our model performs comparably to other methods The minADE of TrafficBots V1.5 is significantly larger than other methods on the leaderboard, which is a known problem inherited from TrafficBots. Overall, from Table 1 we observe that GPT-based [11] architectures that rely on tokenization and next-token prediction, such as SMART and BehaviorGPT, achieve top performance. Interestingly, none of these GPT-based architectures uses goal or personality conditioning, but they are still able to generate multi-modal outputs. It seems that the multi-modality in traffic simulation can be addressed using tokenization and the cross-entropy loss. This indicates that the poor performance of TrafficBots might be caused by the CVAE and regression losses on continuous states and actions. Another interesting thing is that the GPT-based methods achieve the best performance without considering the traffic lights. This indicates that the dataset might be imbalanced and the evaluation metrics might be flawed.

4. Conclusion

In this technical report we present TrafficBots V1.5, which is a baseline method that combines TrafficBots, a prior work on the closed-loop traffic simulation using CVAE, and HPTR, a prior work on the Transformer-based motion prediction using the pairwise-relative representation. Our method is the only CVAE-based method on the WOSAC leaderboard 2024. The performance of our method is slightly worse than the GPT-based methods. However, as a baseline method that involves minor novelty, it achieves the performance we expected, and there are many possibilities to improve this simple baseline.

References

[1] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems (NeurIPS)*, 28, 2015. 2

[2] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *European Conference on Computer Vision (ECCV)*, 2020. 1

[3] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. 1, 2, 3

[4] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[5] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning (ICML)*, 2019. 2, 3

[6] 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 *International Conference on Intelligent Robots and Systems (IROS)*, 2023. 3

[7] 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)*, 2024. 4

[8] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *ICRA*, 2023. 1

[9] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, 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 future trajectories of multiple agents. In *ICLR*, 2021. 1

[10] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[11] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. 4

[12] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 1

[13] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multiagent behaviors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1

[15] Yu Wang, Tiebiao Zhao, and Fan Yi. Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023. *arXiv preprint arXiv:2306.11868*, 2023. 4

[16] Waymo. Waymo open sim agents challenge leaderboard 2024. `https://waymo.com/open/challenges/2024/sim-agents/`, 2024. Accessed: 2024-06-03. 4

[17] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time simulation via next-token prediction. *arXiv preprint arXiv:2405.15677*, 2024. 4

[18] 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. 3

[19] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *International Conference on Computer Vision (ICCV)*, 2021. 1

[20] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. Trafficbots: Towards world models for autonomous driving simulation and motion prediction. In *International Conference on Robotics and Automation (ICRA)*, 2023. 1, 2, 4

[21] 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. 1, 2, 4

[22] 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. 4