TrafficBots: Towards World Models for Autonomous Driving Simulation and Motion Prediction

Zhejun Zhang[1], Alexander Liniger[1], Dengxin Dai[1,2], Fisher Yu[1], Luc Van Gool[1,3]

***Abstract*— Data-driven simulation has become a favorable way to train and test autonomous driving algorithms. The idea of replacing the actual environment with a learned simulator has also been explored in model-based reinforcement learning in the context of world models. In this work, we show data-driven traffic simulation can be formulated as a world model. We present TrafficBots, a multi-agent policy built upon motion prediction and end-to-end driving, and based on TrafficBots we obtain a world model tailored for the planning module of autonomous vehicles. Existing data-driven traffic simulators are lacking configurability and scalability. To generate configurable behaviors, for each agent we introduce a destination as navigational information, and a time-invariant latent personality that specifies the behavioral style. To improve the scalability, we present 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 dot-product attention. As a result, we can simulate all traffic participants seen in dense urban scenarios. Experiments on the Waymo open motion dataset show TrafficBots can simulate realistic multi-agent behaviors and achieve good performance on the motion prediction task.**

I. Introduction

To realize autonomous driving (AD) in the urban environment, the planning module of autonomous vehicles has to address highly interactive driving scenarios involving human drivers, pedestrians and cyclists. Despite being a necessary step, the validation of planning algorithms on public roads is often too expensive and dangerous. Therefore, simulations have been widely adopted and efforts have been made to develop photo-realistic driving simulators [1]. While the full-stack simulators are popular for testing AD stacks and training visuomotor policies, they are not the best choice for developing planning algorithms because the simulated scenarios are not as sophisticated and realistic as those encountered in the real world. Moreover, the computationally demanding rendering is redundant for AD planning modules that expect intermediate-level representations as input.

Therefore, simulators tailored for AD planning should have a different design and rely on real-world datasets. As shown in Fig. 1, the player agent, i.e., the planning module, generates a motion plan by observing some intermediate-level representations. Then the simulator updates its internal states and generates a new observation based on the actions taken by the player agent. The internal states of the simulation can be separated into two categories depending on whether they are reactive to the player agent. The scenario contexts, including the map and traffic controls, are non-reactive states loaded from the datasets. The states of the player agent are reactive and can be updated using vehicle dynamics. Of the most importance for the simulation fidelity are the bot agents, i.e., the non-player agents. The behaviors of bot agents fall into three categories: the non-reactive log-replay, the scripted behavior based on heuristics, and the learned behavior which is our focus. To generate human-like behaviors for bot agents, we present TrafficBots, a multi-agent policy built upon two established research fields: multi-modal motion prediction and end-to-end (E2E) driving.

As shown in Fig. 2, the TrafficBots policy is conditioned on the *destination* of each agent, which approximates the output of a navigator available in the problem formulation of E2E driving [2]. To learn diverse behaviors from demonstrations, each TrafficBot has a *personality* learned using conditional variational autoencoder (CVAE) [3] following multi-modal motion prediction. Compared to other methods, TrafficBots consume less memory, scale to more agents, and run faster than real time. This is achieved by using a vectorized representation [4] for the context and sharing it among all bots. A new scheme of positional encoding (PE) is introduced for angles such that the memory-efficient dot-product attention can be used to retrieve local information from the shared context that lies in the global coordinate.

Using TrafficBots and a differentiable observation generator, the simulator in Fig. 1 is fully differentiable and it summarizes the player agent's past experience, hence it can be trained and used like a *world model* [5]. In this paper we focus on the TrafficBots and leave the training of player agents as future work. We evaluate TrafficBots on both the simulation and motion prediction tasks. We show that motion prediction can be formulated as the *a priori simulation*, hence it is a legit surrogate task for the evaluation of simulation fidelity. While prior works on traffic simulation introduce their own metrics, baselines and datasets, evaluation with motion prediction ensures an open and fair comparison. Although our performance is not comparable to the state-of-the-art open-loop methods, TrafficBots shows the potential of solving motion prediction with a closed-loop policy.

Our contributions are summarized as follows: We address data-driven traffic simulation using world models and we present TrafficBots, a multi-agent policy built upon motion prediction and E2E driving. We improve the simulation configurability by introducing the navigational destination and the latent personality, as well as the scalability by introducing a new PE for angles. Based on the public dataset and leaderboard, we propose a comprehensive and reproducible evalu-

This work is funded by Toyota Motor Europe via TRACE-Zürich.

[1]Computer Vision Lab, ETH Zurich, Switzerland. {zhejun.zhang, alex.liniger,vangool}@vision.ee.ethz.ch, i@yf.io

[2]MPI for Informatics, Germany. ddai@mpi-inf.mpg.de

[3]PSI, KU Leuven, Belgium.

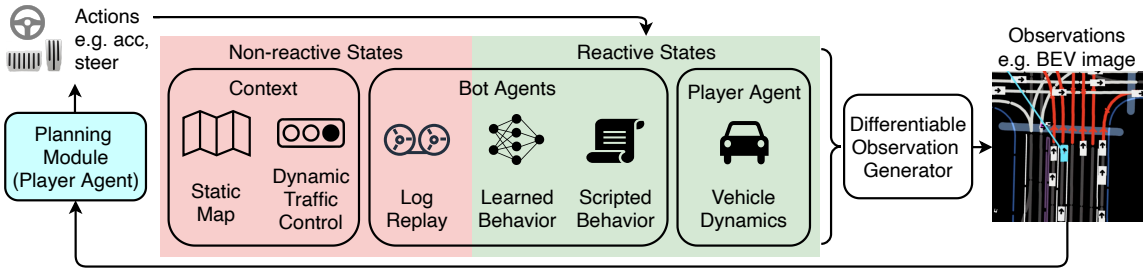


Fig. 1: World model for AD planning modules. The simulator is fully data-driven and differentiable.

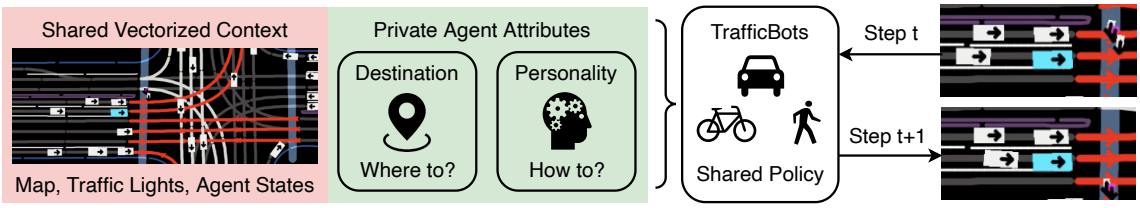


Fig. 2: TrafficBots, a multi-agent policy that generates realistic behaviors for bot agents by learning from real-world data.

ation protocol for traffic simulation. Our repository is available at `https://github.com/zhejz/TrafficBots`

II. Related Work

World models [5] are action-conditional dynamics models learned from observational data. As a differentiable substitute of the actual environment, world models can be used for planning [6] and policy learning [7]. In this paper, we use world models to address a new problem: traffic simulation. We seek to obtain a world model realistic enough to replace the real world or full-stack simulators for developing AD planning algorithms. Training world models is often formulated as a video prediction problem such that the method can generalize to all image-based environments, like Atari [8] and highway driving [9]. Although the same approach can be applied to urban driving via rasterization, this would cause unnecessary complexity because most dynamics of driving can be explicitly modeled without deep learning. In fact, only the decision-dynamics of the bot agents that have a potential to interact with the player agent have to be learned. To this end, we introduce the multi-agent policy TrafficBots and based on it we build a world model for AD planning.

Motion prediction for AD is a popular research topic. Here we only discuss the most relevant works and refer the reader to [10] for a detailed review. Our TrafficBots use a network architecture based on Transformers [11] and vectorized representations [4] because they achieve top performance [12], [13] while being computationally more efficient [14]. To improve the multi-agent performance, our Transformer-based architecture uses a new PE for angles. Goal-conditioning can improve the performance of AD planning [15], [16] and motion prediction [17]–[20], but it leads to causal confusions if applied to closed-loop policy. This problem is solved by replacing the goal, which is associated with the prediction horizon, with the destination, which is time-independent and emulates a navigator. Once conditioned on the destination, the behavior of TrafficBots agent is characterized by a time-invariant personality. The personality is represented as the latent variable of a CVAE, which is used to address the multi-modality of motion forecasting [21]–[24]. Unlike other works, we use a time-invariant personality, i.e., a fixed sample is used throughout the simulation horizon. Finally, TrafficBots is related to [25]–[27] in the sense that a recurrent policy is learned and combined with vehicle dynamics. However, our method is recurrent and closed-loop, whereas motion prediction methods are open-loop.

Data-driven simulation can reduce the sim-to-real gap while being more efficient and scalable than manually developing a simulator. While many works on data-driven simulation focus on the photo-realism [28]–[30], we study the behavior-realism of bot agents. Compared to the hand-crafted rules [1], [31], [32], more realistic behaviors can be generated through log-replay [33], [34] or learning from demonstrations [35]. The problem of learning realistic behaviors is formulated as generative adversarial imitation learning [36] in [37], as behavioral cloning in [38] and as flow prediction in [39]. Most related to our method is TrafficSim [40], an auto-regressive extension of the motion prediction method ILVM [22]. Compared to our method, TrafficSim is not based on world models or E2E driving, it uses rasterization and it does not factorize the uncertainty into personality and destination. Finally, our simulation shown in Fig. 1 can be considered a data-driven extension of SMARTS [41], and TrafficBots shown in Fig. 2 can be used as a sub-module to control bot agents in other simulators [1], [32], [41].

III. Problem Formulation

We use motion prediction datasets to train a policy, which can be used for simulation if a complete episode is given, and for motion prediction if only the history is available.

Data representation. Each episode in the motion prediction dataset includes the static map $\mathbf{M} \in \mathbb{R}^{N_\mathrm{M} \times N_\mathrm{node} \times 4}$, traffic lights $\mathbf{C} \in \mathbb{R}^{T \times N_\mathrm{C} \times 4}$, agent states $\hat{\mathbf{s}} \in \mathbb{R}^{T \times N_\mathrm{A} \times 6}$ and agent attributes $\mathbf{u} \in \mathbb{R}^{N_\mathrm{A} \times 4}$, where N_M is the number of map polylines, N_node is the number of nodes per polyline, N_C is the number of traffic lights and N_A is the number of agents. We define $t = 0$ to be the current step, T_h to be the history length and T_f to be the future length. A polyline node or a

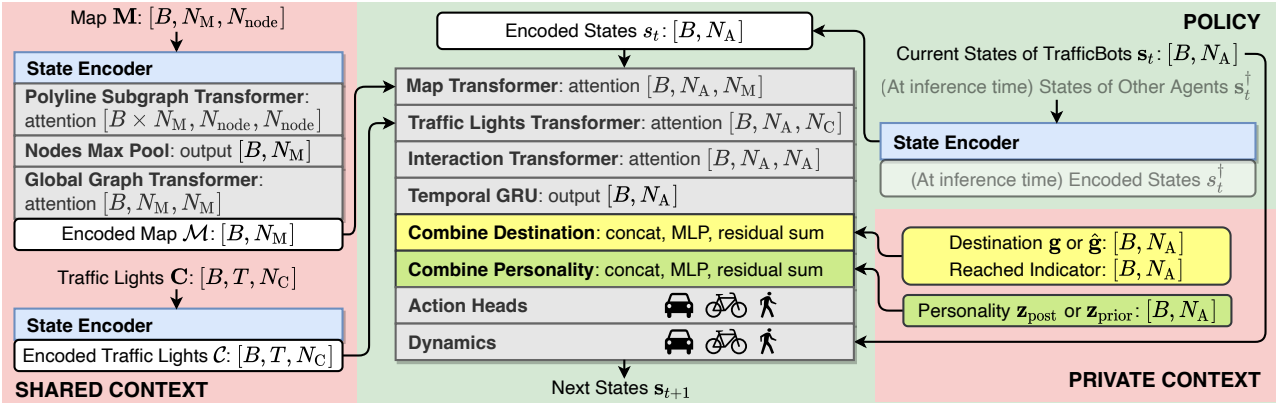


Fig. 3: Network architecture of TrafficBots. In the brackets are the tensor shapes where B is the batch size. The hidden/feature dimensions are omitted for conciseness. The shared and private contexts are encoded only once at the start of an episode.

traffic light is represented by (x, y, θ, u) where x, y are the positions, θ is the yaw angle and u is the polyline type or light state. The ground truth (GT) state of agent i at step t is denoted by $\hat{\mathbf{s}}_t^i = (x, y, \theta, \dot{\theta}, v, a)$ where $\dot{\theta}$ is the yaw rate, v is the speed and a is the acceleration. The time-invariant agent attribute $\mathbf{u}$ includes the agent size and type of each agent. We use a scene-centric, vectorized representation [12] to ensure the efficiency of the simulation.

Simulation. We denote the states of TrafficBots agents as $\mathbf{s}$ and the states of other agents, including the player and other bots, as $\mathbf{s}^{\dagger}$. Given a complete episode, we initialize the simulation with the history $t \in [-T_{\mathrm{h}}, 0]$ and rollout for the future steps $t \in [1, T_{\mathrm{f}}]$. We assume all uncertainties can be explained by the GT future, thus the simulation can be formulated as predicting a single-modal next state $\mathbf{s}_{t+1}$ given $\mathbf{s}_t$ and $\mathbf{s}_t^{\dagger}$. Given the GT future, the simulation has two formulations: counterfactual and a posteriori. In *counterfactual simulation* the behavior of some agents, e.g. the player agent, might deviate from the GT, i.e., $\mathbf{s}^{\dagger} \neq \hat{\mathbf{s}}^{\dagger}$. In this case TrafficBots should be reactive to the change and behave naturally. In the second case, if all agents are either controlled by TrafficBots or log-replay agents, the simulation should ideally reconstruct the same episode. In the spirit of world models, we refer to this as the *a posteriori simulation*.

Motion prediction. We formulate motion prediction as the *a priori simulation*, a special case of the a posteriori simulation where TrafficBots control all agents and the GT is given for $t \in [-T_{\mathrm{h}}, 0]$. In this case, rolling out for $t \in [1, T_{\mathrm{f}}]$ is equivalent to predicting $\mathbf{s}_{1:T_f}^{1:N_{\mathrm{A}}}$, the joint future of all agents. Since the GT future is unavailable and multiple futures are possible given the same history, the a priori simulation is multi-modal and each rollout represents one possible way of how the scenario could evolve. In fact, a priori simulation is equivalent to the multi-modal joint future prediction, which is a more difficult and hence less common task in comparison to the multi-modal marginal motion prediction that considers the prediction independently for each agent.

IV. TrafficBots

As shown in Fig. 3, the TrafficBots policy is conditioned on shared and private contexts which are encoded beforehand and explain all uncertainties, thus the rollout is deterministic.

A. Policy

The policy predicts agent states at the next step $\mathbf{s}_{t+1}$, based on the current states $\mathbf{s}_t$ and the contexts. After encoding $\mathbf{s}_t$, the contexts are sequentially injected into the encoded states s_t. We use Transformer encoder layers with cross-attention to update s_t by attending to the encoded map $\mathcal{M}$ and the encoded traffic lights $\mathcal{C}_t$. The interaction Transformer uses self-attention across the agent dimension to allow agents to attend to each other. At inference time, states of non-TrafficBots agents $s_t^{\dagger}$ will also be processed by these Transformers such that TrafficBots can react to them. After incorporating the map, traffic lights and states of other agents, each agent has a recurrent unit to aggregate its history because the simulation states are not Markovian. Then the outputs are combined with the agent's individual destination and personality via concatenation and residual MLP. Finally, the actions of each agent are predicted by the action heads and $\mathbf{s}_{t+1}$ is computed by the dynamics module based on the actions and $\mathbf{s}_t$.

B. Contexts

State encoder. Following [12], all shared contexts, i.e., the map $\mathbf{M}$, traffic lights $\mathbf{C}$ and agent states $\mathbf{s}$, are represented in the global coordinates and incorporated via dot-product attention. This approach is computationally more efficient than transforming the global information to the local coordinate of each agent. However, the dot-product attention alone cannot efficiently model a global to local coordinate transform. To remedy this issue, PE is introduced. Without PE, VectorNet [4] has to transform all contexts to the local coordinate of each agent. SceneTransformer [12] concatenates the PE for position with the unit vector for direction and other attributes u, and then feeds it to an MLP:

$$s = \mathrm{MLP}(\mathrm{cat}(\mathrm{PE}(x), \mathrm{PE}(y), \cos\theta, \sin\theta, u)), \tag{1}$$

$$\text{with } \mathrm{PE}_{2i}(x) = \sin(x \cdot \omega^{\frac{2i}{d_{\mathrm{emb}}}}),\ \mathrm{PE}_{2i+1}(x) = \cos(x \cdot \omega^{\frac{2i}{d_{\mathrm{emb}}}}),$$

where $i \in [0, \ldots, d_{\mathrm{emb}}/2]$, ω is the base frequency and d_{emb} is the embedding dimension. This state encoder can be improved by using PE also for the direction vector [42] and adding the PE after the MLP [11]. This ends up with

$$s = \mathrm{cat}(\mathrm{PE}(x), \mathrm{PE}(y), \mathrm{PE}(\cos\theta), \mathrm{PE}(\sin\theta)) + \mathrm{MLP}(u). \tag{2}$$

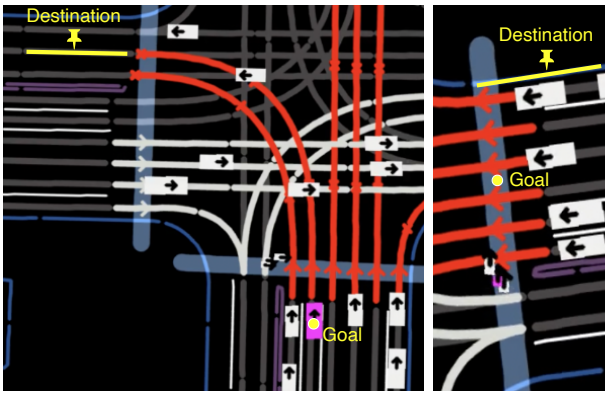


Fig. 4: GT destination and goal of the magenta agent.

However, empirically we observe TrafficBots using this state encoder is not sensitive to directional information. To address this issue, we propose the following state encoder

$$s = \text{cat}(\text{PE}(x), \text{PE}(y), \text{AE}(\theta), \text{MLP}(u)) \quad (3)$$
$$\text{with } \text{AE}_{2i}(\theta) = \sin(\theta \cdot i),\ \text{AE}_{2i+1}(\theta) = \cos(\theta \cdot i)$$

where $i \in [1, \ldots, d_{\text{emb}}/2 + 1]$ and AE stands for angular encoding, a special case of sinusoidal PE we introduced to encode the radian yaw θ. Compared to PE that has to use a small ω to avoid overloading the 2π period, AE can use integer frequency because it encodes an angle. Moreover, the addition is replaced by concatenation because other states, e.g. velocity, are highly correlated to the pose encoded by PE and AE. We use the state encoders to encode the map, traffic lights and agent states. Our map encoder follows [4], except that we use Transformers for the polyline sub-graph.

Destination. Fig. 4 highlights the difference between our destination and the goal proposed in prior works [17]–[20]. The GT goal, which is associated to the last observed position, does not reflect an agent's intention. In Fig. 4, the vehicle stops because of the red light, whereas the pedestrian does not intend to stay in the middle of a crosswalk. Although this is not a problem for open-loop motion prediction, the driving policy would learn a wrong causal relationship if conditioned on the goal. This problem can be solved by introducing a navigator, which specifies the next goal once the current one is reached. However, running an online navigator for every agent is computationally demanding. For simulation with a short horizon and small maps, it is sufficient to estimate one destination for the near future, and switch to an unconditioned policy once that destination is reached. Since the GT destination is not available in any motion prediction datasets, we approximate it with a map polyline heuristically selected by extending the recorded agent trajectory based on the map topology. For training and simulation we use the approximated GT destination $\hat{\mathbf{g}}$, whereas for motion prediction we predict $\mathbf{g}$. Predicting the destination is formulated as a multi-class classification task where the logit for polyline i and agent j is predicted by $\text{MLP}(\text{cat}(\mathcal{M}^i, \text{GRU}(s^j_{-T_\text{h}:0})))$, i.e., the destination of an agent depends only on the map and its own history.

Personality. In order to address the remaining uncertainties not explained by the destination and to learn diverse behaviors of different human drivers, pedestrians and cyclists, we introduce a latent personality for each agent which is learned using CVAE. Similar ideas have been applied to world models [5]–[7] and motion prediction [21]–[23]. The personality encoder has a similar architecture as the policy network in Fig. 3. For training and simulation, we use the posterior $\mathbf{z}_\text{post}$ which is estimated from the complete episode $t \in [-T_\text{h}, T_\text{f}]$, whereas for motion prediction we use the prior $\mathbf{z}_\text{prior}$ that encodes only the history $t \in [-T_\text{h}, 0]$. In contrast to TrafficSim [40] which updates the latent at each time step to address all uncertainties, our personality is time-invariant because the behavioral style of an agent will not change in a short time horizon if the destination is determined.

TABLE I: Results on the WOMD (marginal) leaderboard.

test	mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓	overlap rate ↓
DenseTNT [18]	0.328	1.039	1.551	0.157	0.178
SceneTransformer [12]	0.279	0.612	1.212	0.156	0.147
MultiPath [43] static	0.236	0.880	2.044	0.345	0.166
Waymo LSTM [44]	0.176	1.007	2.355	0.375	0.190
TrafficBots (a priori)	0.212	1.313	3.102	0.344	**0.145**
valid TrafficBots	↑	↓	↓	↓	↓
a priori (*K=6*)	0.210	1.291	3.117	0.346	0.143
GT sdc future (*what-if*)	0.214	1.281	3.095	0.342	0.142
GT traffic light (*v2x*)	0.209	1.288	3.100	0.345	0.143
GT destination (*v2v*)	0.217	1.292	3.123	0.345	0.142
a posteriori (*K=1*)	0.332	0.962	2.034	0.339	0.129

C. Training

Similar to world models [5], our training uses reparameterization gradients and back-propagation through time (BPTT). Given a complete episode, we first encode the map $\mathbf{M}$, traffic lights $\mathbf{C}$ and GT agent states $\hat{\mathbf{s}}$. Then we predict $\mathbf{z}_\text{post}$, $\mathbf{z}_\text{prior}$ and the destination $\mathbf{g}$. Conditioned on the GT destination $\hat{\mathbf{g}}$ and a sample of $\mathbf{z}_\text{post}$ we rollout the policy. For $t \in [-T_\text{h}, 0]$ we warm-start using teacher-forcing with GT agent states, whereas for $t \in [1, T_\text{f}]$ the rollout is auto-regressive. All components are trained simultaneously using the weighted sum of three losses: the reconstruction loss with smoothed L1 distance for the states (x, y, θ, v), the KL-divergence between $\mathbf{z}_\text{post}$ and $\mathbf{z}_\text{prior}$ clipped by free nats [6], and the cross-entropy loss for destination prediction. Following [7], we stop the gradient from the action and allow only the gradient from the states during the BPTT. We train with all agents so as to generate realistic behaviors for all traffic participants, not just for the interested ones heuristically selected by the dataset.

D. Implementation Details

We use a 16-dim diagonal Gaussian for the personality. The action heads and dynamics have the same architecture but different parameters for vehicles, cyclists and pedestrians. We use a unicycle model with constraints on maximum yaw rate and acceleration for all types of agents. With a hidden dimension of 128 our model has less than 3M parameters. Considering 64 agents, 1024 map polylines and a sampling time of 0.1 second, we can parallelize 16 simulations on one 2080Ti GPU while each rollout step takes around 10 ms, which is a magnitude faster than other methods [37]–[40].

TABLE II: Ablation on the WOMD validation split. All models are trained for 24K iterations (48 hours).

		a priori simulation K=6 (motion prediction)						a posteriori simulation K=1					
		mAP ↑	min ADE ↓	min FDE ↓	miss rate ↓	overl. rate ↓	NLL ↓ ($\times 10^{-7}$)	dif. pos (m) ↓	dif. rot (deg) ↓	veh col (%) ↓	run red (%) ↓	passive (%) ↓	miss rate ↓
Our best	TrafficBots	**0.18**	1.49	3.66	**0.39**	**0.15**	1.37	0.80	2.84	**11.5**	1.31	19.1	0.42
Encoder	Eq. 1	0.12	1.74	4.48	0.48	0.18	1.90	0.74	3.05	14.7	1.47	19.4	0.49
	Eq. 2	0.14	1.62	4.12	0.46	0.17	1.48	0.74	3.02	13.8	1.46	19.3	0.48
Personality	w/o persona	0.06	1.66	4.09	0.48	0.15	1.16	1.29	3.63	13.6	1.50	19.2	0.53
	larger KL	0.15	1.65	4.19	0.42	0.17	1.88	**0.47**	**2.39**	12.9	1.56	19.1	0.24
Destination	w/o dest.	0.16	1.53	3.80	0.40	0.15	1.44	0.74	2.63	11.8	**1.29**	19.3	0.41
	goal	0.17	**1.47**	**3.44**	0.40	0.16	2.02	0.78	2.68	12.3	1.35	20.2	0.42
	goal w/o navi	0.14	1.57	3.83	0.45	0.17	3.39	0.79	2.97	15.1	1.40	23.3	0.49
World Model	w/o free nats	0.18	1.52	3.74	0.40	0.16	1.39	0.86	3.00	12.6	1.31	19.1	0.44
	w/ action grad.	0.17	1.51	3.71	0.41	0.16	1.39	0.90	2.82	12.6	1.30	19.1	0.46
SimNet [38]	BC w/o pers. & dest.	0.01	2.76	7.77	0.76	0.21	2.64	2.27	7.37	21.9	1.59	19.6	0.76
	w/o pers. & dest.	0.02	1.91	4.95	0.55	0.15	**1.10**	1.34	3.69	13.6	1.46	19.2	0.54
	BC	0.09	3.11	9.24	0.73	0.21	3.34	2.99	7.56	33.4	4.27	19.3	0.76
TrafficSim [40]	w/o dynamics	0.14	1.81	4.37	0.46	0.17	1.68	0.72	55.18	48.0	1.73	**18.9**	0.45
	inter. decoder	0.17	1.52	3.73	0.41	0.16	1.66	0.75	2.85	12.8	1.46	19.2	**0.22**
	resample pers.	0.14	1.81	4.74	0.47	0.16	1.56	0.49	2.45	12.8	1.55	19.5	0.29

V. Experiments

Dataset. We use the Waymo Open Motion Dataset (WOMD) [44] because compared to other datasets it has longer episode lengths and more diverse and complex driving scenarios, such as busy intersections with pedestrians and cyclists. The WOMD is also one of the largest motion prediction datasets, consisting of 487K episodes for training, 44K for validation and 45K for testing. With a fixed sampling time of 0.1 second, each episode is 9 seconds long and contains 91 steps: $T_{\mathrm{h}} = 10$ for the history, one for the current $t = 0$, and $T_{\mathrm{f}} = 80$ future steps that shall be predicted.

Tasks. Ultimately we want to verify the fidelity of the *counterfactual simulation*, such that the simulator can be used for training and testing planning modules. However, once the scenario diverges from the factual recording, the GT trajectories can no longer be used for evaluation metrics. To this end, different surrogate metrics have been proposed, such as traffic rule compliance [40] and distribution of curvatures [37]. But these metrics cannot fully reflect the behavioral fidelity because they consider only vehicles and neglect pedestrians and cyclists. Moreover, performing well on these metrics does not mean the behavior is human-like, in fact good performance can be achieved by a hand-crafted policy. Alternatively we can verify the fidelity of the *a posteriori simulation*, where the scenario should be reconstructed and the performance can be quantified by the distance to the GT trajectories. But since the GT future is given, a model can achieve good performance by misusing the posterior latent to memorize the GT future, instead of learning the underlying human-like behavior. In fact, the best possible performance can be simply achieved via log-replay. We argue the *a priori simulation*, i.e., motion prediction, together with the a posteriori simulation is a better evaluation setup. For a priori simulation, the model predicts multiple futures of how an episode might evolve. While all predictions should demonstrate natural behaviors, at least one of them should reconstruct the GT future. Importantly, motion prediction is usually formulated as an open-loop problem. Although TrafficBots can be used for motion prediction by formulating it as the a priori simulation, the performance will be affected by the covariant shift and compounding errors [45] caused by the closed-loop rollout. Nevertheless, we show the potential of solving motion prediction with a multi-agent policy.

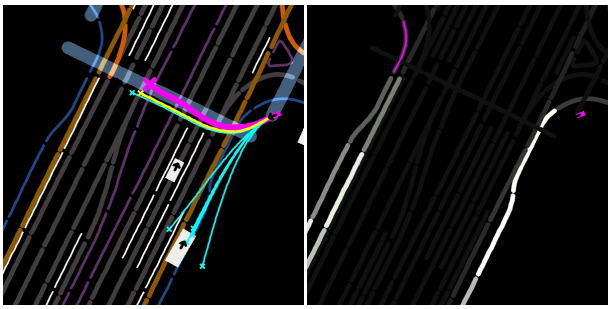
Metrics. For motion prediction we follow the metrics of WOMD [44], including the accuracy metrics mAP, the distance-based minADE/FDE and miss rate, and the surrogate metric overlap rate. Inspired by [24], we further examine the sampling-based negative log-likelihood (NLL) of the GT scene. The WOMD specifies up to 8 agents that shall be predicted and allows up to K=6 predictions. Accordingly, we generate 6 rollouts, i.e., the joint future of all agents, by sampling the destination and the prior personality. For a posteriori simulation, only one rollout is generated using the most likely posterior personality and the GT destination. The simulation fidelity is evaluated using traffic rule violation rate and distance to GT trajectories. The differences in position and rotation are averaged over all steps and agents, whereas the rates of collision, running a red light and passiveness (stop moving for no reason) are for vehicles only.

Comparison with motion prediction methods. In the first half of Table I we compare TrafficBots with open-loop motion prediction methods on the Waymo (marginal) motion prediction leaderboard. In terms of mAP we are better than the Waymo LSTM baseline [44], but worse than other methods because TrafficBots is not optimized to generate diverse predictions which is favored by the mAP metrics. Although the miss rates are comparable, the minADE/FDE of our method are significantly higher than other methods. This can be explained by the compounding errors caused by the auto-regressive policy rollout. While this drawback is well-known for closed-loop methods, TrafficBots still has its advantage which is shown by the reduced overlap rate.

Compared to the open-loop methods, it is easier for a policy to learn the correct causal relationship. The second half of Table I shows that the prediction performance can be improved given additional information. Since the predictions are generated via rollout, we can set some of the future observations to their GT. For example, for conditional motion prediction (*what-if*) the future trajectory of the self-driving-car is given. Furthermore, the future traffic light states and the destinations could be obtained via vehicle-to-everything (*v2x*) or vehicle-to-vehicle (*v2v*) communication. Having access to all future information, the a posteriori simulation achieves the best performance with a single ($K=1$) prediction.

Ablations. In Table II we ablate the state encoders, personality, destination and world-model training techniques on both the a priori and the a posteriori simulation. Our state encoder Eq. 3 with AE performs overall better than *Eq. 1* and *Eq. 2*. *Without the personality*, the policy is unable to capture the diverse behaviors of different traffic participants. If we allow a *larger KL* divergence by downweighting the KL loss, the performance is better for a posterior simulation but worse for motion prediction. Then we have TrafficBots *w/o destination* where the latent captures all uncertainties. In this case the model performs worse on motion prediction because the Gaussian latent suffers from mode averaging. If the policy is conditioned on the *goal*, i.e., the polyline associated with the last observed pose, then the model will learn a wrong causal relationship and the traffic rule violation rates will increase even though the minADE/FDE are smaller. If we use the *goal w/o navigator* module that drops the goal once it is reached, the policy learning will fail completely and the performances are overall inferior. Finally, we show world-model training techniques can improve the performance of TrafficBots. To further compare with prior works on traffic simulation, we ablate more design differences between our method and SimNet [38], which uses behavioral cloning (BC) without personality or destination, as well as TrafficSim [40]. Generally, TrafficBots performs better but there are three interesting exceptions: Firstly, if we allow a *larger KL* or *resample pers.*, the posterior will memorize the GT future and the prior will fail to infer the personality. Consequently the model performs better for a posterior simulation but worse for motion prediction, and the traffic rule violation rates are higher because the model masters the memorization rather than the driving skills. This highlights the importance of using a time-invariant personality and the advantage of evaluating with both a priori and a posteriori simulation. Secondly, models without personality have smaller NLL. This is reasonable because models without CVAE generate less diverse predictions, hence the NLL is smaller. Finally, the model with an *interactive decoder* following TrafficSim [40] shows a smaller miss rate during a posteriori simulation. This is achieved by adding the private contexts before the interaction Transformer, such that private contexts are shared among all agents. However, this requires the personality and destination of all agents to be known before the rollout, which is infeasible if the simulation includes a player agent whose future actions are undetermined.

(a) A vehicle entering the parking lots.



(b) A cyclist crossing the road through the crosswalk.

Fig. 5: In each sub-figure, left: predicted trajectories; right: heat map of predicted destinations. Agent of interest and GT are in magenta. A priori predictions are in cyan. A posteriori simulated trajectory is in yellow. The brightness is proportional to the probability in the destination heat map.

Qualitative results. Fig. 5 shows two examples of the prediction and simulation results. In both cases, one of the a priori predictions matches the GT, whereas the a posteriori simulation reconstructs the scenario with less deviation. With similar destinations but sampled personalities, five predictions in Fig. 5a follow the lane with different speeds and lane selections. With predicted destinations on both sides of the road, the cyclist in Fig. 5b is predicted to either cross the road or follow the road edge.

VI. Conclusions and Future Works

This paper presented TrafficBots, a multi-agent policy learned from motion prediction datasets. Based on the shared, vectorized context and the individual personality and destination, TrafficBots can generate realistic multi-agent behaviors in dense urban scenarios. Besides the simulation, TrafficBots can also be used for motion prediction. Evaluating on motion prediction tasks allows us to verify the simulation fidelity and benchmark on a public leaderboard. Based on TrafficBots, we build a differentiable, data-driven simulation framework, which in the future can serve as a platform to develop AD planning algorithms, or as a world model to train E2E driving policies via reinforcement learning [2] or model-based imitation learning [33]. Moreover, TrafficBots could also be integrated as a module to generate human-like behaviors for bot agents in a game or a full-stack AD simulator. Future work will investigate better network architectures and training techniques, the downstream tasks, and combining data-driven traffic simulation with neural rendering.

REFERENCES

[1] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[2] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, "End-to-end urban driving by imitating a reinforcement learning coach," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 222–15 232.

[3] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[4] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[5] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in neural information processing systems*, vol. 31, 2018.

[6] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International conference on machine learning*. PMLR, 2019, pp. 2555–2565.

[7] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[8] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering atari with discrete world models," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[9] M. Henaff, A. Canziani, and Y. LeCun, "Model-Predictive Policy Learning with Uncertainty Regularization for Driving in Dense Traffic," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[10] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7814–7821.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[12] J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal, *et al.*, "Scene transformer: A unified architecture for predicting future trajectories of multiple agents," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[13] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7577–7586.

[14] R. Girgis, F. Golemo, F. Codevilla, M. Weiss, J. A. D'Souza, S. E. Kahou, F. Heide, and C. Pal, "Latent Variable Sequential Set Transformers for Joint Multi-Agent Motion Prediction," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[15] S. V. Albrecht, C. Brewitt, J. Wilhelm, B. Gyevnar, F. Eiras, M. Dobre, and S. Ramamoorthy, "Interpretable goal-based prediction and planning for autonomous driving," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1043–1049.

[16] C. Brewitt, B. Gyevnar, S. Garcin, and S. V. Albrecht, "Grit: Fast, interpretable, and verifiable goal recognition with learned decision trees for autonomous driving," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1023–1030.

[17] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, and C. Schmid, "Tnt: Target-driven trajectory prediction," in *Conference on Robot Learning*, 2020.

[18] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 303–15 312.

[19] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2821–2830.

[20] N. Deo, E. Wolff, and O. Beijbom, "Multimodal trajectory prediction conditioned on lane-graph traversals," in *Conference on Robot Learning*. PMLR, 2022, pp. 203–212.

[21] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 336–345.

[22] S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, "Implicit latent variable model for scene-consistent motion forecasting," in *European Conference on Computer Vision*. Springer, 2020, pp. 624–641.

[23] C. Tang and R. R. Salakhutdinov, "Multiple futures prediction," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[24] B. Ivanovic and M. Pavone, "The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2375–2384.

[25] N. Rhinehart, K. M. Kitani, and P. Vernaza, "R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 772–788.

[26] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision*. Springer, 2020, pp. 683–700.

[27] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool, "Deep interactive motion prediction and planning: Playing games with motion prediction models," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 1006–1019.

[28] S. W. Kim, J. Philion, A. Torralba, and S. Fidler, "Drivegan: Towards a controllable high-quality neural simulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5820–5829.

[29] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[30] A. Amini, T.-H. Wang, I. Gilitschenski, W. Schwarting, Z. Liu, S. Han, S. Karaman, and D. Rus, "Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2419–2426.

[31] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2575–2582.

[32] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[33] O. Scheel, L. Bergamini, M. Wolczyk, B. Osiński, and P. Ondruska, "Urban driver: Learning to drive from real-world demonstrations using policy gradients," in *Conference on Robot Learning*. PMLR, 2022, pp. 718–728.

[34] H. Caesar, J. Kabzan, K. S. Tan, W. K. Fong, E. Wolff, A. Lang, L. Fletcher, O. Beijbom, and S. Omari, "nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.

[35] F. Behbahani, K. Shiarlis, X. Chen, V. Kurin, S. Kasewa, C. Stirbu, J. Gomes, S. Paul, F. A. Oliehoek, J. Messias, *et al.*, "Learning from demonstration in the wild," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 775–781.

[36] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[37] M. Igl, D. Kim, A. Kuefler, P. Mougin, P. Shah, K. Shiarlis, D. Anguelov, M. Palatucci, B. White, and S. Whiteson, "Symphony: Learning realistic and diverse agents for autonomous driving simulation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[38] L. Bergamini, Y. Ye, O. Scheel, L. Chen, C. Hu, L. Del Pero, B. Osiński, H. Grimmett, and P. Ondruska, "Simnet: Learning reactive self-driving simulations from real-world observations," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5119–5125.

[39] A. Kamenev, L. Wang, O. B. Bohan, I. Kulkarni, B. Kartal, A. Molchanov, S. Birchfield, D. Nistér, and N. Smolyanskiy, "Pre-

dictionnet: Real-time joint probabilistic traffic prediction for planning, control, and simulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8936–8942.

[40] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 400–10 409.

[41] M. Zhou, J. Luo, J. Villela, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, and Z. Chen, "SMARTS: Scalable multi-agent reinforcement learning training school for autonomous driving," in *Conference on Robot Learning*. PMLR, 2020.

[42] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[43] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *Conference on Robot Learning*, 2019.

[44] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. 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*, 2021, pp. 9710–9719.

[45] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[46] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 524–10 533.

[47] D. Wu and Y. Wu, "Air2 for interaction prediction," in *Workshop on Autonomous Driving, CVPR*, 2021.

[48] X. Mo, Z. Huang, and C. Lv, "Multi-modal interactive agent trajectory prediction using heterogeneous edge-enhanced graph attention network," in *Workshop on Autonomous Driving, CVPR*, 2021.

APPENDIX

A. Supplementary Video

The supplementary video for the paper is found here at `https://youtu.be/2idvJOqbXeo`. This video contains more experimental results generated by TrafficBots. The video is nicely edited and exhaustively commented. It includes two episodes, the first episode highlights a vehicle making an U-turn on a narrow street, whereas the second episode is at a busy intersection with traffic lights and a large number of traffic participants. For each episode, we first show the results of a posteriori simulation and a priori motion prediction, and then we inspect agents demonstrating the most interesting behaviors. Besides the good cases, this video also presents the bad cases where our method failed to generated realistic behavior.

B. Dataset and Pre-Processing

We use the unfiltered 9-second datasets (*scenario*, not the filtered *tf_example*) from the WOMD, these are: *testing, testing_interactive, training, validation, validation_interactive*. The WOMD also provides a full-length training dataset *training_20s*, which includes the original 20-second-long episodes. In contrast to the 9-second datasets which are clipped from the 20-second-long episodes, episodes in the *training_20s* do not always have a fixed length. Although we did not use the 20-second dataset, future works can take advantage of it for simulation with a longer time horizon. We pre-process the dataset by first filtering the map polylines:

1) Split the original map polylines into shorter polylines with maximum $N_{\text{node}} = 20$ nodes one meter away from each other.
2) Remove polylines too far away from any agents.
3) Remove polylines that contain too few nodes.
4) Continue removing polylines based on the distance to agents, until the number of remaining polylines is smaller than a threshold $N_{\text{M}} = 1024$.

Then we filter the traffic lights which are associated with map polylines. A traffic light will be filtered if its map polyline is removed. Finally we filter agents as follows:

1) Remove agents that are tracked for too few steps.
2) Remove agents that have small displacement and large distance to any of the relevant agents marked by the WOMD or any of the map polylines. These agents are mostly parking vehicles.
3) Remove vehicles that have small displacement but large yaw change, which are caused by tracking errors.
4) Continue removing irrelevant agents based on the distance to relevant agents, until the number of remaining agents is smaller than a threshold $N_{\text{A}} = 64$.

After the filtering, we center the episode such that the position of the self-driving-car is at $(0, 0)$. The training episodes are randomly rotated by an angle between $-\pi$ and π, whereas the validation and testing episodes are unaffected. We smooth the agent trajectories and fill in the missing steps via temporal linear interpolation. A pre-processed episode has $T = 91$ steps and includes the following data:

1) Agent states
   - $agent/valid$: $[T, N_{\text{A}}]$, Boolean mask.
   - $agent/pos$: $[T, N_{\text{A}}, 2]$, x, y positions.
   - $agent/vel$: $[T, N_{\text{A}}, 2]$, velocities in x, y directions.
   - $agent/spd$: $[T, N_{\text{A}}, 1]$, m/s
   - $agent/acc$: $[T, N_{\text{A}}, 1]$, m/s^2
   - $agent/yaw\_bbox$: $[T, N_{\text{A}}, 1]$, rad.
   - $agent/yaw\_rate$: $[T, N_{\text{A}}, 1]$, rad/s.
2) Agent attributes
   - $agent/type$: $[N_{\text{A}}, 3]$; vehicle, pedestrian, cyclist.
   - $agent/role$: $[N_{\text{A}}, 3]$, 3 types of role; self-driving-car, agent of interest, agent to predict.
   - $agent/size$: $[N_{\text{A}}, 3]$, length, width, height.
3) Map
   - $map/valid$: $[N_{\text{M}}, N_{\text{node}}]$, Boolean mask.
   - $map/type$: $[N_{\text{M}}, 11]$, 11 types of polylines. They are *freeway*, *surface_street*, *stop_sign*, *bike_lane*, *road_edge_boundary*, *road_edge_median*, *solid_single*, *solid_double*, *passing_double_yellow*, *speed_bump* and *crosswalk*.
   - map/pos: $[N_{\text{M}}, N_{\text{node}}, 2]$, x, y position of nodes.
   - map/dir: $[N_{\text{M}}, N_{\text{node}}, 2]$, a 2D vector pointing to the next node.
4) Stop point of traffic lights
   - $tl\_stop/valid$: $[T, N_{\text{C}}]$, Boolean mask.
   - $tl\_stop/state$: $[T, N_{\text{C}}, 5]$, 5 types of states; *unknown*, *stop*, *caution*, *go* and *flashing*.

- tl_stop/pos: $[T, N_{\mathrm{C}}, 2]$, position of the stop point.
- tl_stop/dir: $[T, N_{\mathrm{C}}, 2]$, direction of the stop point.

C. Ground-Truth Destination

The GT destinations are not available in any motion prediction datasets. Therefore, we use the following heuristics to approximate the GT destination of an agent:

- If the agent is a **vehicle on a lane**, i.e. the last observed pose of the agent is close enough to a driving lane in terms of position and direction, then we find the destination by randomly selecting one of the successors of that lane base on the map topology. This step will be repeated multiple times. These agents are vehicles driving on the road. In this case the type of the destination is either *freeway*, *surface_street* or *stop_sign*.
- If the agent is a **vehicle not on lane**, then we extend the last observed pose with constant velocity for 5 seconds. After that the *road_edge_boundary* polyline closest to that extended position will be selected. These agents are mostly vehicles in parking lots.
- For **cyclists on bike lanes**, we extend the last position with constant velocity and find the closest *bike_lane*.
- For **cyclists not on bike lanes** or **pedestrians**, we find the *road_edge_boundary* polyline closest to the position extended using constant velocity.

For the ablation we have a model trained with *goal* instead of destination. In this case the goals are still polylines and the GT goals are still approximated using the aforementioned method, with the exception that we do not extend the last observed position using map topology or constant velocity. The map polyline closest to the last observed position will be directly used as the *goal*. Unlike motion prediction methods [17], [18] that predict an accurate goal and then simply fit a smooth trajectory towards the goal, our destination is less informative such that the motion profile is determined solely by the policy. The destinations are pre-processed and saved as agent attributes $agent/dest$: $[N_{\mathrm{A}}]$. We save the indices of the corresponding map polyline, hence the value of $agent/dest$ ranges from 1 to N_{M}.

D. Detailed Network Architecture

We use dropout probability 0.1 and ReLU activation.

State Encoders. The architecture of the state encoders discussed in the main paper are visualized in Fig. 6.

Transformers. We use the Transformer encoder layer with cross-attention as shown in Fig. 7. The layer norm is inside the residual blocks [46]. If the query, key and value share the same tensor, then the cross-attention boils down to self-attention which is used by the interaction Transformer.

Combine Personality and Destination. As shown in Fig. 8, the personality, or the destination, is injected to the intermediate state via concatenation, MLP and residual sum. Since the personality is always valid, the masking is unnecessary for combining personality. In terms of destination, the masking is based on a reached indicator. If the destination is reached, then the output of the residual block will be masked such that the intermediate states remain unchanged and the destination no longer affects the policy.

Action Heads. We use a two-layer MLP to predict the acceleration and the yaw rate of each agent. We instantiate three action heads with the same architecture; one for each type of agent. The outputs of action heads are normalized to $[-1, 1]$ via the $\tanh$ activation.

Dynamics. Following MultiPath++ [10] we use a unicycle dynamics with constraints on maximum yaw rate and acceleration for all types of agents. For vehicles the acceleration is limited to $\pm 5\,\mathrm{m/s}$ and the yaw rate is limited to $\pm 1.5\,\mathrm{rad/s}$. For cyclists we use $\pm 6\,\mathrm{m/s}$, $\pm 3\,\mathrm{rad/s}$ and for pedestrians $\pm 7\,\mathrm{m/s}$, $\pm 7\,\mathrm{rad/s}$. The outputs of action heads are multiplied by the maximum allowed acceleration or yaw rate to obtain the final actions.

Personality Encoder. The inputs to the map, traffic lights and interaction Transformer of the personality encoder are reshaped differently. For the map encoder, we flatten the agent states tensor with shape $[B, T, N_{\mathrm{A}}]$ to $[B, T \times N_{\mathrm{A}}]$ and use it to query the map with shape $[B, N_{\mathrm{M}}]$. This allows each agent at each time step to attend to the map independently. For the traffic lights Transformer, the agent states tensor with shape $[B, T, N_{\mathrm{A}}]$ is flattened to $[B \times T, N_{\mathrm{A}}]$ and the traffic lights with shape $[B, T, N_{\mathrm{C}}]$ is flattened to $[B \times T, N_{\mathrm{C}}]$. In this case, the agents states can only attend to the traffic lights from the same time step. Similarly, inputs to the interaction Transformer are reshaped from $[B, T, N_{\mathrm{A}}]$ to $[B \times T, N_{\mathrm{A}}]$, such that an agent can only attend to other agents' states from the same time step. We use two personality encoders with the same architecture to encode the posterior and the prior personality respectively.

Latent Distribution of Personality. The personality encoder predicts the mean of a 16-dimensional diagonal Gaussian for each agent. The standard deviation is a learnable parameter independent of any inputs. We initialize the log standard deviation to -2 for all the 16 dimensions. The standard deviation parameter is shared by agents from the same type (vehicle, pedestrian, cyclist).

Predicting Destinations. The destination of agent j depends only on the encoded map $\mathcal{M}$ and its own encoded history states $s^j_{-T_{\mathrm{h}}:0}$. Given $\mathcal{M}^i$, the hidden feature of the ith polyline of the encoded map $\mathcal{M}$, the logit p^j_i for polyline i and agent j is predicted by

$$p^j_i = \mathrm{MLP}(\mathcal{M}^i, \mathrm{GRU}(s^j_{-T_{\mathrm{h}}:0})),$$
$$\text{where } i \in \{1, \ldots, N_{\mathrm{M}}\},\ j \in \{1, \ldots, N_{\mathrm{A}}\}.$$

Based on these logits, the destinations of agent j are represented by a categorical distribution with N_{M} classes and the probability is obtained via softmax. After obtaining the polyline index i, the predicted destination $\mathbf{g}$ is the encoded polyline feature $\mathcal{M}^i$ indexed by i. The polyline indices of the GT destinations are saved during the dataset pre-processing.

E. Training Details

We use six 2080Ti GPUs for the training with a batch size of 4 on each GPU, i.e. the total batch size is $B = 24$.

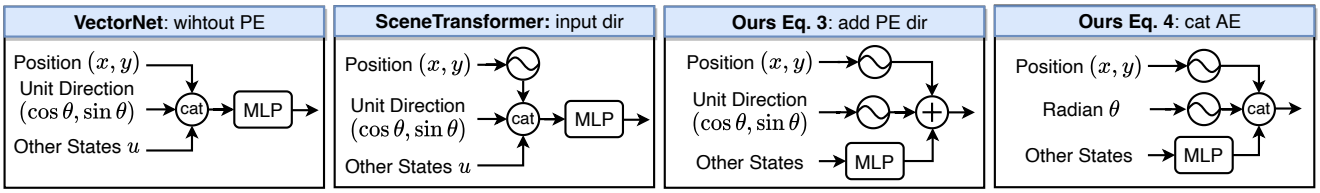


Fig. 6: State encoders with different architectures.

TABLE III: Performance on the Waymo (joint) interactive prediction leaderboard

test	soft mAP ↑	mAP ↑	minADE ↓	minFDE ↓	miss rate ↓	overlap rate ↓
DenseTNT [18]	N/A	0.165	1.142	2.490	0.535	0.231
SceneTransformer (J) [12]	N/A	0.119	0.977	2.189	0.494	0.207
Air2 [47]	N/A	0.096	1.317	2.714	0.623	0.247
HeatIRm4 [48]	N/A	0.084	1.420	3.260	0.722	0.284
Waymo LSTM [44]	N/A	0.052	1.906	5.028	0.775	0.341
TrafficBots (a priori)	0.113	0.111	1.669	4.514	0.681	0.220
valid TrafficBots	soft mAP ↑	mAP ↑	minADE ↓	minFDE ↓	miss rate ↓	overlap rate ↓
a priori (*K=6*)	0.102	0.100	1.670	4.514	0.677	0.221
GT sdc future (*what-if*)	0.110	0.108	1.577	4.317	0.651	0.215
GT traffic light (*v2x*)	0.102	0.100	1.663	4.485	0.675	0.221
GT destination (*v2v*)	0.106	0.103	1.640	4.440	0.668	0.223
a posteriori (*K=1*)	0.188	0.188	1.085	2.313	0.602	0.165

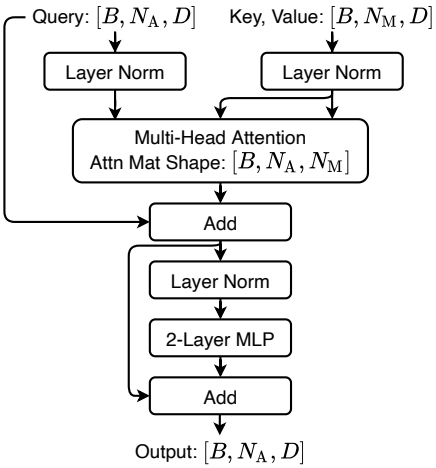


Fig. 7: Transformer encoder layer with pre-layer-norm.

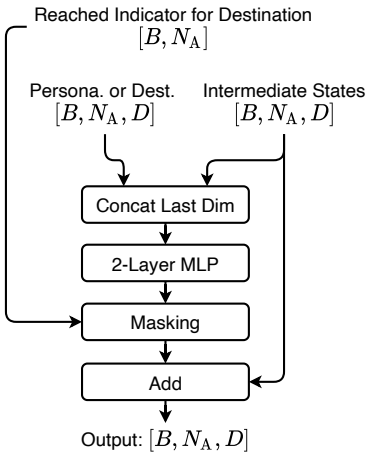


Fig. 8: Combine personality/destination.

Due to the large size of the WOMD training dataset, in each epoch we randomly select 15% from the complete training and validation datasets. We use the Adam optimizer with a learning rate of 4e-4. The learning rate is halved every 7 epochs. The model converges after about 30 epochs, that is almost a week. We predict the posterior personality $\mathbf{z}_{\text{post}}$ using the posterior personality encoder and information from $t \in [-T_{\text{h}}, T_{\text{f}}]$. Similarly $\mathbf{z}_{\text{prior}}$ is predicted using the prior personality encoder and information from $t \in [-T_{\text{h}}, 0]$. The logits of destinations are predicted using the encoded map $\mathcal{M}$ and the GT agent states $\hat{s}_{-T_{\text{h}}:0}$ from the past. From the logits we use softmax to obtain a multi-class categorical distribution of the destination of each agent $P_{\text{dest}}^{1:N_{\text{A}}}$, which has N_{M} classes; one for each map polyline. During the training we rollout with the GT destination and the posterior personality $\mathbf{z}_{\text{post}}$. Our training loss has the following terms:

1) Reconstruction loss, which trains the model to reconstruct the GT states using the posterior personality and the GT destination. It is a weighted sum of:
   - A smoothed L1 loss between the predicted (x, y) positions and the GT positions.
   - A cosine distance between the predicted yaw θ and the GT yaw $\hat{\theta}$, i.e. $0.5 \cdot (1 - \cos(\theta - \hat{\theta}))$.
   - A smoothed L1 loss between the predicted velocity and the GT velocity.
2) The KL divergence between the posterior and the prior personality, which trains the prior to match the posterior and regularize the posterior at the same time. We use free nats [6] to clip the KL divergence, i.e. if $KL(\mathbf{z}_{\text{post}}, \mathbf{z}_{\text{prior}})$ is smaller than the free nats, then the KL loss is not applied. We use a free nats of 0.01.
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.

F. Inference Details

We use the GT destination and the most likely posterior personality for the a posteriori simulation, hence the simulation is single modal in this case. For a priori simulation, i.e. motion prediction, we generate multiple modes by randomly sampling the destination distribution and the prior personality of each agent. For WOMD we generate $K = 6$ predictions. The first mode K_0 is deterministic, which is generated using the most likely destination and prior personality. We use this mode to inspect the most likely mode of the joint future prediction. The score of each prediction, which is required by the WOMD leaderboard, is the joint probability of the destination and the personality. We normalize the score using softmax with temperature. The scores are computed with respect to agents, not the joint future of all agents. For motion prediction where the future traffic light states are not available, we use the last observed (i.e. from the current step $t = 0$) light states for all prediction steps.

G. More Experimental Results

In Table III we compare TrafficBots with other open-loop motion prediction methods on the Waymo (joint) interactive prediction leaderboard, where the joint future of exactly two agents shall be predicted and the metrics are evaluated at the scene-level, i.e. for both agents at the same time. For a more detailed description on the task and the metrics, please refer to the publication [44] or the homepage of the WOMD. Since our method is essentially solving the joint future prediction, TrafficBots significantly outperforms the baselines on this task. As shown in Table III, we achieve overall better performance than the *LSTM* baseline [44]. TrafficBots also perform better than *HeatIRm4* [48], the winner of the 2021 WOMD challenge, and *Air2* [47], the honorable mention of the 2021 WOMD challenge, in terms of the mAP and the overlap rate, which are the most relevant metrics used for the ranking. Our performance is comparable to *SceneTransformer (J)*, the joint version of SceneTransformer [12]. Compared to *DenseTNT* [18], we achieve a lower overlap rate. As discussed in the main paper, our method suffers from larger minADE/FDE and the performance can be improved given additional GT information. These trends are also observed in Table III. Although the (joint) interactive prediction is a more favorable task for our method, we do not include Table III in the main paper because this leaderboard is partially deprecated and hence less active, and the predictions are restricted to two agents which significantly limits its application in the real world.