Appendix

In the appendix, we provide more details about the experiments discussed in the main text. Section A introduces the implementation details of the diffusion model and the specific content and form of the guide function. Section B details the implementation of the system and showcases the visualization of scenarios in the simulator. Section C covers the relevant content of the SenseTime driving dataset, while Section D delves into the detailed experimental configurations for multi-style reinforcement learning experiments. Code and Demos are available at `https://github.com/tsinghua-fib-lab/LCSim`.

A Diffusion Model

The process of the diffusion model generating vehicle action sequences is shown in Figure 7. With the road network topology and vehicle historical states as input, the model generates future action sequences for the vehicle through a denoising diffusion process.

Due to the relevant regulations of the Waymo Open Motion Dataset (WOMD) [26], we cannot provide the parameters of the model trained on it. In this section, we introduce the implementation details of the diffusion model and the hyperparameters used for training and inference in detail to ensure that the relevant experimental results can be easily reproduced.

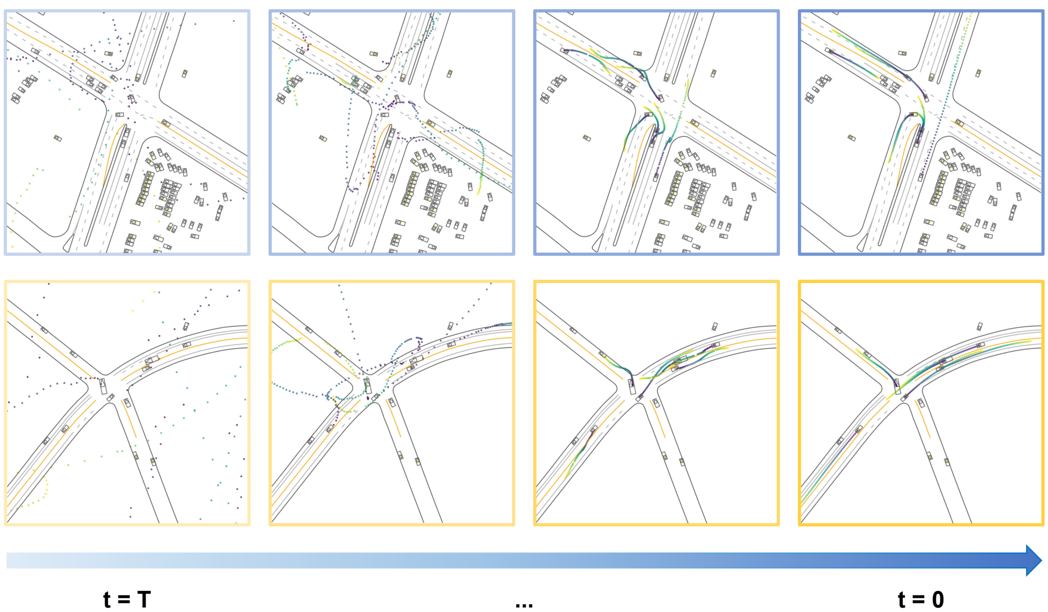

Figure 7: The process of generating vehicle action sequences by diffusion model.

A.1 Problem Formulation

Similar to [23], we denote a traffic scenario as $\omega = (M, A_{1:T})$, where M contains the information of a High-Definition (HD) map and $A_{1:T} = [A_1, ..., A_T]$ is the state sequence of all traffic participates. Each element m_i of $M = \{m^1, ..., m^{N_m}\}$ represents the map factor like road lines, road edges, centerline of lanes, etc. And each element a_i^t of $A_t = \{a_t^1, ..., a_t^{N_a}\}$ represents the state of the ith traffic participate at time step t including position, velocity, heading, etc.

Given the map elements $M = \{m^1, ..., m^{N_m}\}$ and the historical states of agents $A_{t_c - T_h : t_c}$, where T_h is the number historical steps and $0 < T_h < t_c$, the model generates the future states of agents in the scenario $A_{t_c : t_c + T_f}$, where T_f is the number of future steps.

Table 4: The attention mechanisms of scene encoder.

	Query	Key	Value
Agent Temporal	$\mathbf{v}_{i,t_c}^a$	$\mathbf{v}_{i,t}^a$	$\mathbf{v}_{i,t}^a \oplus Pos(t-t_c)$
Map-Map	$\mathbf{v}_i^m$	$\mathbf{v}_j^m$	$\mathbf{v}_j^m \oplus \mathbf{e}_{ij}^{m\to m}$
Agent-Map	$\mathbf{v}_{i,t_c}^a$	$\mathbf{v}_j^m$	$\mathbf{v}_j^m \oplus \mathbf{e}_{ij}^{a\to m}$
Agent-Agent	$\mathbf{v}_{i,t_c}^a$	$\mathbf{v}_{j,t_c}^a$	$\mathbf{v}_{j,t_c}^a \oplus \mathbf{e}_{ij}^{a\to a}$

A.2 Model Architecture

Scene Encoder. We implemented our scene encoder based on MTR [32] and QCNet [51]. As mentioned before, at each time step t_c, the input to the scene encoder includes the map elements $M = \{m^1, ..., m^{N_m}\}$ and the historical states of agents $A_{t_c - T_h : t_c}$. First, we construct a heterogeneous graph $G = (V, E)$ based on the geometric relationships among input features. The node set V contains two kinds of node v^a and v^m and the edge set E consists of three kinds of edge $e^{a\to a}$, $e^{a\to m}$ and $e^{m\to m}$. Connectivity is established between nodes within a certain range of relative distances. For nodes like v_i^a and v_j^m, their node features contain attributes independent of geographical location like lane type, agent type, agent velocity, etc. The position information of nodes is stored in the relative form within the edge features like $e_{ij}^{a\to m} = [\mathbf{p}_j^m - \mathbf{p}_i^a, \theta_j^m - \theta_i^a]$, where $\mathbf{p}$ and θ are position vector and heading angle of each node at current time step t_c. For each category of elements in the graph, we use an MLP to map their features into the latent space with dimension N_h to get the node embedding $\mathbf{v}_{i,t}^a (t_c - T_h \leq t \leq t_c)$, $\mathbf{v}_j^m$ and edge embedding $\mathbf{e}_{ij}^{a\to a}$, $\mathbf{e}_{ij}^{a\to m}$, $\mathbf{e}_{ij}^{m\to m}$. Then we apply four attention mechanisms in Table 4 to them to get the final scene embedding. The scene embedding consists of two components: the map embedding with a shape of $[M, N_h]$, and the agent embedding with a shape of $[A, T_h, N_h]$.

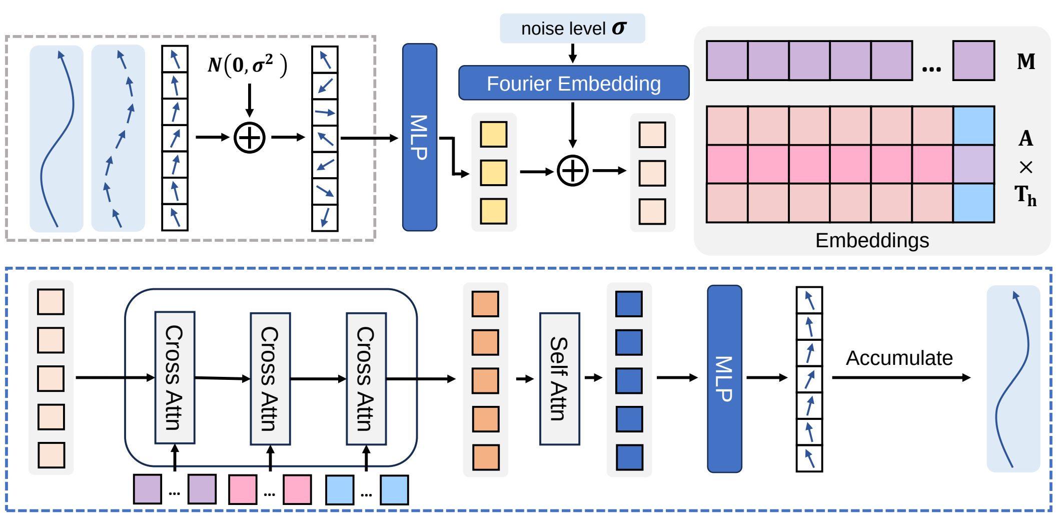

Figure 8: The architecture of diffusion decoder.

Diffusion Decoder. Figure 8 shows the whole architecture of the diffusion decoder. Similar to [52], we implemented a DETR-like decoder to model the joint distribution of multi-agent action sequences. Denote the generation target as $\boldsymbol{x} \in \mathbb{R}^{A \times T_f \times N_a}$, which represents future T_f steps' actions of agents in the scenario. Firstly, noise $\boldsymbol{z} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\right)$ is added to the input sequence. Subsequently, the action sequence with noise for each agent is mapped to a latent space via an MLP, serving as the query embedding for that agent. The query is then added to the Fourier Embedding with noise level σ, similar to positional encoding, to inform the model about the current noise level. Next, the query

Table 5: Model parameters

Parameter	Value
Input Size	2
Output Size	2
Embedding Size	128
Num Historical Steps	10
Num Future Steps	80
Num Polygon Types	20
Num Freq Bands	64
Map Encoder	
Hidden Dim	64
Num Layers	5
Num Pre Layers	3
Agent Encoder	
Time Span	10
a2a Radius	50
pl2a Radius	50
Num Layers	2
Num Heads	8
Head Dim	64
Dropout	0.1
Diff Decoder	
Output Head	False
Num t2m Steps	10
pl2m Radius	150
a2m Radius	150
Num Layers	2
Num Recurrent Steps	2
Num Heads	8
Head Dim	64
Dropout	0.1

Table 6: Training parameters

Parameter	Value
Batch Size	16
Num Epochs	200
Weight Decay	0.03
Learning Rate	0.0005
Learning Rate Schedule	OneCycleLR
σ_{data}	0.1
$c_{in}(\sigma)$	$1/\sqrt{\sigma^2+\sigma_{data}^2}$
$c_{skip}(\sigma)$	$\sigma_{data}^2/(\sigma^2+\sigma_{data}^2)$
$c_{out}(\sigma)$	$\sigma\cdot\sigma_{data}/\sqrt{\sigma^2+\sigma_{data}^2}$
$c_{noise}(\sigma)$	$\frac{1}{4}\ln\sigma$
Noise Distribution	$\ln(\sigma)\sim\mathcal{N}\left(P_{\text{mean}}, P_{\text{std}}^2\right)$
P_{mean}	-1.2
P_{std}	1.2

vector undergoes cross-attention with map embeddings, embeddings of other agents in the scenario, and the historical state embedding of the current agent, resulting in a fused agent feature vector incorporating environmental information. Following this, self-attention is applied to the feature vectors of each agent to ensure the authenticity of interaction among the action sequences generated for each agent. Finally, the feature vectors from the latent space are mapped back to the agent's action space via an MLP to obtain the de-noised agent action sequence.

A.3 Training Details

Training Target. Diffusion model estimates the distribution of generation target $\boldsymbol{x}\sim p(\boldsymbol{x})$ by sampling from $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ with learnable model parameter $\boldsymbol{\theta}$. Normally we have $p_{\boldsymbol{\theta}}(\boldsymbol{x})=\frac{-f_{\boldsymbol{\theta}}(\boldsymbol{x})}{Z_{\boldsymbol{\theta}}}$, and use max-likelihood $\max_{\boldsymbol{\theta}}\sum_{i=1}^{N}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ to get parameter $\boldsymbol{\theta}$. However, to make the max likelihood training feasible, we need to know the normalization constant $Z_{\boldsymbol{\theta}}$, and either computing or approximating it would be a rather computationally expensive process, So we choose to model the score function $\nabla_{\boldsymbol{x}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x};\sigma)$ rather than directly model the probability density, with the score function, one can get data sample $\boldsymbol{x}_0\sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ by the following equation [17]:

$$\boldsymbol{x}_0=\boldsymbol{x}(T)+\int_T^0-\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}(t);\sigma(t))dt \quad \text{where } \boldsymbol{x}(T)\sim\mathcal{N}\left(\mathbf{0},\sigma_{\max}^2\boldsymbol{I}\right) \tag{1}$$

On this basis, we add a condition $\boldsymbol{c}$ composed of scene embeddings and use our model to approximate the score function $\nabla_{\boldsymbol{x}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x};\boldsymbol{c},\sigma)\approx\left(D_{\boldsymbol{\theta}}(\boldsymbol{x};\boldsymbol{c},\sigma)-\boldsymbol{x}\right)/\sigma^2$, which leads to the training target [17]:

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{c}\sim\chi_c}\mathbb{E}_{\sigma\sim q(\sigma)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\sigma^2\boldsymbol{I})}\left\|D_{\boldsymbol{\theta}}(\boldsymbol{x}+\boldsymbol{\epsilon};\boldsymbol{c},\sigma)-\boldsymbol{x}\right\|_2^2 \tag{2}$$

χ_c is the training dataset combined with embeddings computed by the scene encoder, and $q(\sigma)$ represents the schedule of the noise level added to the original data sample. For better performance, we introduce the precondition as described in [18] to ensure that the input and output of the model both follow a standard normal distribution with unit variance:

$$D_{\boldsymbol{\theta}}(\boldsymbol{x};\boldsymbol{c},\sigma)=c_{\text{skip}}(\sigma)\boldsymbol{x}+c_{\text{out}}(\sigma)F_{\boldsymbol{\theta}}\left(c_{\text{in}}(\sigma)\boldsymbol{x};\boldsymbol{c},c_{\text{noise}}(\sigma)\right) \tag{3}$$

Here, $F_{\boldsymbol{\theta}}(\cdot)$ represents the original output of the diffusion decoder. In the experiment, we used the magnitude and direction of vehicle speed as the target for generation.

Experiment Setting. We trained our diffusion model on the Waymo Open Motion Dataset (WOMD) [26]. Each traffic scenario in the dataset has a duration of 9 seconds. We used the map information and the historical state of the previous 1 second as input to the model and generated future vehicle action sequences for the next 8 seconds. The training was conducted on a server with 4 $\times$ Nvidia 4090 GPUs. We set the batch size for training to 16 and trained with the OneCycleLR learning rate schedule for 200 epochs. The entire training process lasted approximately 20 days. The detailed parameters of the model and the training process are shown in Table 5 and Table 6.

A.4 Guide Functions

Following [50, 17], we calculate the cost function $\mathcal{L}:\mathbb{R}^{A\times T_f\times N_a}\mapsto\mathbb{R}$ based on the intermediate results of the generation process and propagate gradients backward to guide the final generation outcome. In our experiments, the control objectives include the vehicle's maximum acceleration, target velocity, time headway, and relative distance to the preceding car during car-following, and generating adversarial behavior by controlling nearby vehicles to approach the current vehicle. Denote vehicle i at timestep t has states $acc_{i,t}, v_{i,t}, x_{i,t}, y_{i,t}, heading_{i,t}$, and $dis_t(i,j)$ computes the relative distance between vehicle i and vehicle j at timestep t when vehicle i is followed by vehicle j on the same lane. Table 7 shows the details of the cost functions.

Table 7: The cost functions used in the guided generation process.

Guide Target	Cost Function
max acceleration	$\sum_{i=1}^{A}\sum_{t=1}^{T_f}\max(0,\lvert acc_{i,t}\rvert-acc_{max})$
target velocity	$\sum_{i=1}^{A}\sum_{t=1}^{T_f}\parallel v_{i,t}-v_{target}\parallel_2^2$
time headway	$\sum_{t=1}^{T_f}\sum_{i\neq j}\lvert\frac{dis_t(i,j)}{\lVert v_{j,t}\rVert_2^2}-thw_{target}\rvert$ where i is followed by j at t
relative distance	$\sum_{t=1}^{T_f}\sum_{i\neq j}\lvert dis_t(i,j)-dis_{target}\rvert$ where i is followed by j at t
goal point	$\sum_{i=1}^{A}\sum_{t=1}^{T_f}\parallel(x_{i,t},y_{i,t})-(x_{goal_{i,t}},y_{goal_{i,t}})\parallel_2^2$
no collision	$\sum_{t=1}^{T_f}\sum_{i\neq j}\mathbb{I}[\parallel(x_{i,t},y_{i,t})-(x_{j,t},y_{j,t})\parallel_2^2\leq\epsilon]$
no off-road	$\sum_{i=1}^{A}\sum_{t=1}^{T_f}\mathbb{I}[\parallel(x_{i,t},y_{i,t})-(x_{\text{off-road}},y_{\text{off-road}})\parallel_2^2\leq\epsilon]$

B Simulation System

B.1 Scenario Generator

We defined a unified map and vehicle Origin-Destination (OD) format based on Protobuf[4]. Additionally, we have developed format conversion tools designed for the Waymo and Argoverse datasets, the conversion results can be seen in Figure 9.

[4] `https://github.com/tsinghua-fib-lab/LCSim/blob/main/lcsim/protos`

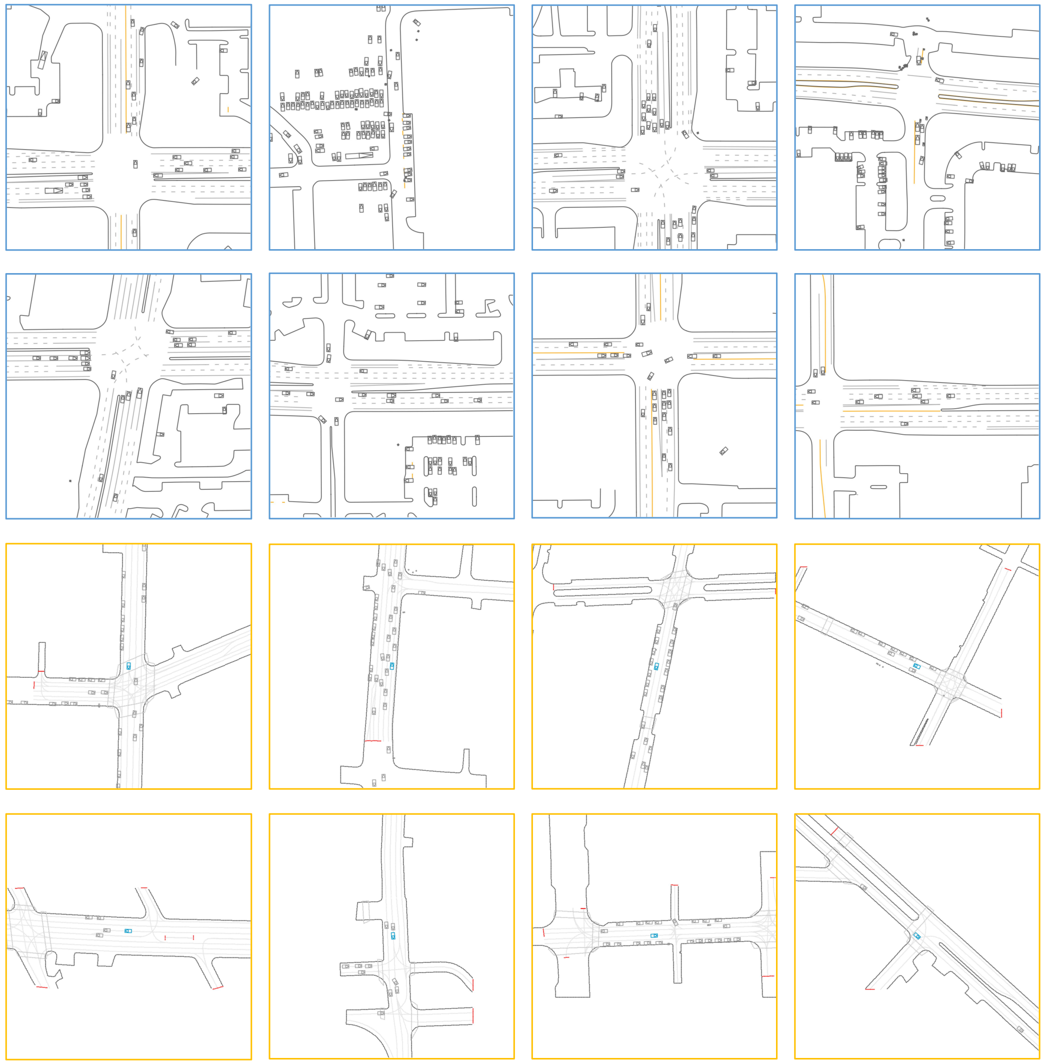

Figure 9: Traffic scenarios from WOMD (blue box) and Argoverse (yellow box).

B.2 Policy Details

We implemented five different policies to support traffic simulation in various scenarios:

- *ExpertPolicy*: The vehicles strictly follow the given action sequences to proceed.
- *BicycleExpertPolicy*: Based on the expert policy, we impose kinematic constraints on the vehicle's behavior using a bicycle model to prevent excessive acceleration and steering. By default, we set max acceleration to 6.0 m/s^2 and max steering angle to 0.3 rad.
- *LaneIDMPolicy*: Under this policy, vehicles ignore the action sequences and proceed along the center line of their current lane. The vehicle's acceleration is calculated using the IDM model and lane-changing behavior is generated using the Mobil model.
- *TrajIDMPolicy*: Vehicles move along the trajectories computed based on the action sequence, but their acceleration is controlled by the IDM Mode to prevent collisions.
- *RL-based Policy*: A PPO [31] agent trained based on our simulator, its observation space contains the scene embedding and the action sequence. The action space consists of acceleration and steering values. The training environment of this agent is the second one, enabling diffusive simulation with Waymo-style vehicle behavior.

For the IDM model in these policies, the default configuration is that $acc_{max} = 5m/s^2, thw = 2.0s, v_{target} = 20m/s$.

C SenseTime Driving Dataset

C.1 Dataset Overview

SenseTime driving dataset comprises about 426.26 hours of vehicle driving logs collected from vehicles based on SenseAuto[5] in the Beijing Yizhuang area and the whole dataset is split into 765 scenarios. The data is presented in a format similar to vehicle trajectories in the Waymo dataset with a sampled rate of 10 Hz. However, the road networks of the scenarios are not provided in this dataset, so we can not train our model on it, but due to the sufficient duration of the data, we can analyze the behavioral characteristics of vehicles within the data collection area. This analysis provides a reference for constructing driving scenarios with different styles.

Understandably, due to confidentiality regulations, the complete dataset cannot be released. However, we will share the statistical distribution data of vehicle behaviors obtained from the dataset.

C.2 Vehicle Behavior Analysis

We conducted statistical analysis on the dataset, focusing on metrics such as max acceleration, usual brake acceleration, velocity, relative distance, relative velocity, and time headway during the car following process, Figure 10 shows the results. This analysis allowed us to derive the driving behavior characteristics of vehicles in the Yizhuang area.

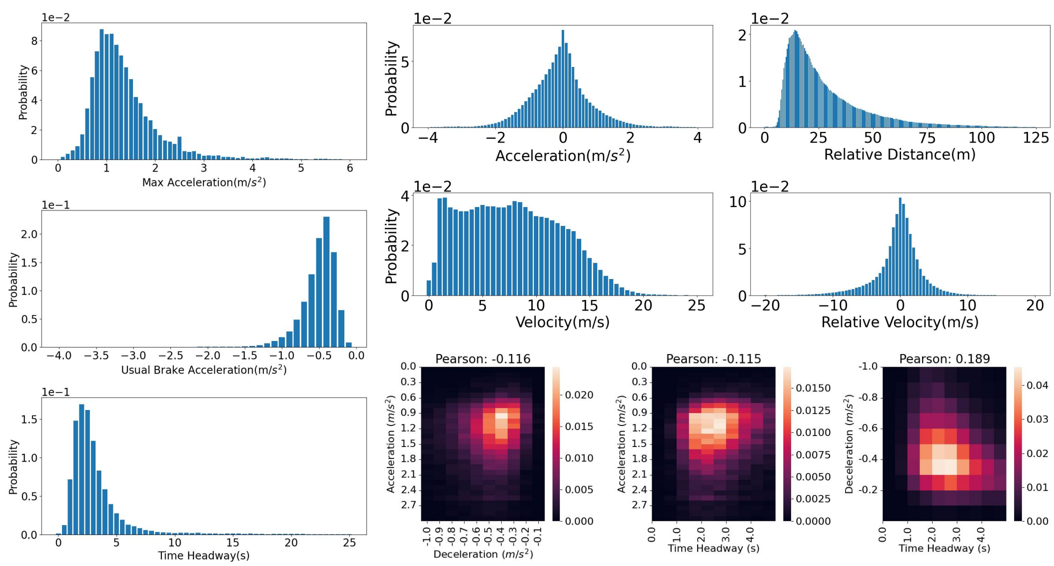

Figure 10: The analysis of SenseTime driving dataset.

D Multi-Style Reinforcement Learning

We constructed single-agent reinforcement learning experiments based on the Waymo traffic scenarios with our guided diffusive simulation to see the influence of styles of scenarios on policy learning.

[5] https://www.senseauto.com/

D.1 Reinforcement Learning Setup

We constructed a reinforcement learning environment based on the validation set of the Waymo dataset. 4,400 scenarios are selected from the validation set and further divided into a training set containing 4,000 scenarios and a test set containing 400 scenarios. We trained a PPO [31] agent on the training set and evaluated its performance on the test set.

Observation Spec. Observation of the agent consists of two parts:

- Scene Embedding: Embedding computed by scene encoder of the diffusion model with size of $[N_h]$, by applying cross attention to map polygons and agent states, this feature contains information about surrounding vehicles, road elements, and the vehicle's own historical states. In this experiment, we use $N_h = 128$ following the setup of the diffusion model.
- Route: We sampled the vehicle's trajectory points within the next 1 second at a frequency of 10Hz and projected them into a relative coordinate system based on the vehicle's current position and orientation. The shape of the route data is $[10, 2]$, representing the reference path of the vehicle's forward movement. If the vehicle behavior in the driving environment is generated by a diffusion model, then this path will be accumulated from the behavior sequences generated by the model for the vehicle.

Action Spec. We let the agent directly control the throttle and steering angle of the vehicle. The agent's output is a two-dimensional vector with a range $[-1, 1]$. This vector is multiplied by the maximum range of acceleration and steering angle, resulting in the final vehicle action. In this experiment, the maximum acceleration and steering angle of the vehicle are set to 6.0 and 0.3, respectively.

Rollout Setting. To let the agent explore every scenario in the training set, we randomly divided the 4000 scenes in the training set into 20 parts, each containing 200 different scenarios. We used 20 parallel threads to rollout episodes, with each thread pre-loading and pre-calculating map embeddings for 200 different training scenarios. During the rollout process, after the current episode ends, the environment automatically switches to the next scenario, and this cycle continues iteratively.

Reward Function. Our goal is to make the vehicle progress along the given route while avoiding collisions and staying within the road. Therefore, we provide the following formula for the reward:

$$R = R_{forward} + P_{collision} + P_{road} + P_{smooth} + R_{destination}. \tag{4}$$

The meanings of elements in the formula are as follows:

- $R_{forward}$: A dense reward to encourage the vehicle to drive forward along the given route. We project the current position and last position of the vehicle onto the Frenet coordinate of the route and calculate $d_t, d_{t-1}, s_t, s_{t-1}$, the value of the reward would be $0.1 \times ((s_t - s_{t-1}) - (d_t - d_{t-1}))$.
- $P_{collision}$: Penalty for collision, When the vehicle collides, the value will be -10, and the current episode terminates; otherwise, the value is 0.
- P_{road}: Penalty for driving off the road, when this happens, the value will be -5, and the current episode terminates; otherwise, the value is 0.
- P_{smooth}: Following [23], we implemented $P_{smooth} = min(0, 1/v_t - |a[0]|)$ to avoid a large steering value change between two timesteps.
- $R_{destination}$: When an episode ends, we check if the vehicle has reached the destination of the given route, which means the distance to the endpoint of the route is within 2.5 meters. If yes, the reward value is 10; otherwise, it's -5.

D.2 Multi-Style Environments Building

We build four kinds of environments with different driving styles using cost functions in Table 7:

- The original Waymo driving environment, in this environment, vehicles base their actions on real trajectories from the Waymo driving logs.
- The Waymo-style environment with diffusive simulation. This environment utilizes the diffusion model without guide functions, the vehicle behaviors are consistent with the Waymo dataset. With the diffusion model's nature, it generates diverse vehicle trajectories under the same initial conditions, exposing the agent to a broader range of traffic scenarios during training.
- The SenseTime-style environment with guided diffusive simulation. This environment follows the driving style observed in the SenseTime driving dataset, emphasizing a more "gentle" driving behavior compared to the Waymo-based environment. In this environment, we use cost functions on max acceleration with $acc_{max} = 3m/s^2$, and on time headway with $thw_{target} = 2.5s$.
- The adversarial environment. This environment is implemented by guiding nearby vehicles closer to the vehicle controlled by the RL agent. For vehicles in front of or alongside the main vehicle, we guide their action generation with $dis_{target} = 0$ to the main vehicle, thereby encouraging more sudden braking and cutting-in behaviors, increasing the aggressiveness of the environment.

References

[1] M. Bansal, A. Krizhevsky, and A. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. arXiv preprint arXiv:1812.03079, 2018.

[2] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo–simulation of urban mobility: an overview. In Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation. ThinkMind, 2011.

[3] 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), pages 5119–5125. IEEE, 2021.

[4] R. Bhattacharyya, B. Wulfe, D. J. Phillips, A. Kuefler, J. Morton, R. Senanayake, and M. J. Kochenderfer. Modeling human driving behavior through generative adversarial imitation learning. IEEE Transactions on Intelligent Transportation Systems, 24(3):2874–2887, 2022.

[5] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer. Multi-agent imitation learning for driving simulation, 2018.

[6] E. Brockfeld, R. D. Kühne, A. Skabardonis, and P. Wagner. Toward benchmarking of microscopic traffic flow models. Transportation research record, 1852(1):124–129, 2003.

[7] E. Bronstein, M. Palatucci, D. Notz, B. White, A. Kuefler, Y. Lu, S. Paul, P. Nikdel, P. Mougin, H. Chen, J. Fu, A. Abrams, P. Shah, E. Racah, B. Frenkel, S. Whiteson, and D. Anguelov. Hierarchical model-based imitation learning for planning in autonomous driving, 2022.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 11621–11631, 2020.

[9] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv preprint arXiv:1910.05449, 2019.

[10] P. De Haan, D. Jayaraman, and S. Levine. Causal confusion in imitation learning. Advances in neural information processing systems, 32, 2019.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator, 2017.

[12] C. Gulino, J. Fu, W. Luo, G. Tucker, E. Bronstein, Y. Lu, J. Harb, X. Pan, Y. Wang, X. Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. Advances in Neural Information Processing Systems, 36, 2024.

[13] K. T. e. a. H. Caesar, J. Kabzan. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In CVPR ADP3 workshop, 2021.

[14] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset, 2020.

[15] 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, 2022.

[16] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In 2018 IEEE international conference on robotics and automation (ICRA), pages 2034–2039. IEEE, 2018.

[17] C. M. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, and D. Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion, 2023.

[18] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models, 2022.

[19] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. In 2019 international conference on robotics and automation (ICRA), pages 8248–8254. IEEE, 2019.

[20] P. Kothari, C. Perone, L. Bergamini, A. Alahi, and P. Ondruska. Drivergym: Democratising reinforcement learning for autonomous driving, 2021.

[21] R. Krajewski, T. Moers, D. Nerger, and L. Eckstein. Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 2383–2390. IEEE, 2018.

[22] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. IEEE transactions on pattern analysis and machine intelligence, 45(3):3461–3475, 2022.

[23] Q. Li, Z. M. Peng, L. Feng, Z. Liu, C. Duan, W. Mo, and B. Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. Advances in neural information processing systems, 36, 2024.

[24] C. Liang, Z. Huang, Y. Liu, Z. Liu, G. Zheng, H. Shi, K. Wu, Y. Du, F. Li, and Z. J. Li. Cblab: Supporting the training of large-scale traffic control policies with scalable traffic simulation. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 4449–4460, 2023.

[25] Z. Lin, G. Zhang, Z. He, J. Feng, W. Wu, and Y. Li. Vehicle trajectory recovery on road network based on traffic camera video data. In Proceedings of the 29th International Conference on Advances in Geographic Information Systems, pages 389–398, 2021.

[26] W. LLC. Waymo open dataset: An autonomous driving dataset, 2019.

[27] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7553–7560. IEEE, 2023.

[28] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 17305–17315, 2022.

[29] 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, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[30] F. Sagberg, Selpi, G. F. Bianchi Piccinini, and J. Engström. A review of research on driving styles and road safety. Human factors, 57(7):1248–1275, 2015.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[32] S. Shi, L. Jiang, D. Dai, and B. Schiele. Motion transformer with global intention localization and local movement refinement, 2023.

[33] 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, pages 10400–10409, 2021.

[34] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun. Scenegen: Learning to generate realistic traffic scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 892–901, 2021.

[35] C. Tang, W. Zhan, and M. Tomizuka. Exploring social posterior collapse in variational autoencoder for interaction modeling. Advances in Neural Information Processing Systems, 34:8481–8494, 2021.

[36] E. Vinitsky, N. Lichtlé, X. Yang, B. Amos, and J. Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world, 2023.

[37] P. Wang, C.-Y. Chan, and A. de La Fortelle. A reinforcement learning based approach for automated lane change maneuvers. In 2018 IEEE Intelligent Vehicles Symposium (IV), pages 1379–1384. IEEE, 2018.

[38] Waymo. Waymo safety report. Waymo Safety Repor, 2021.

[39] L. Wenl, D. Fu, S. Mao, P. Cai, M. Dou, Y. Li, and Y. Qiao. Limsim: A long-term interactive multi-scenario traffic simulator. In 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), pages 1255–1262. IEEE, 2023.

[40] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021.

[41] D. Xu, Y. Chen, B. Ivanovic, and M. Pavone. Bits: Bi-level imitation for traffic simulation. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 2929–2936, 2023.

[42] X. Yan, Z. Zou, S. Feng, H. Zhu, H. Sun, and H. X. Liu. Learning naturalistic driving environment with statistical realism. Nature communications, 14(1):2037, 2023.

[43] F. Yu, W. Ao, H. Yan, G. Zhang, W. Wu, and Y. Li. Spatio-temporal vehicle trajectory recovery on road network based on traffic camera video data. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 4413–4421, 2022.

[44] F. Yu, H. Yan, R. Chen, G. Zhang, Y. Liu, M. Chen, and Y. Li. City-scale vehicle trajectory data from traffic camera videos. Scientific data, 10(1):711, 2023.

[45] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, and Z. Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In The World Wide Web Conference, WWW '19, page 3620–3624, New York, NY, USA, 2019. Association for Computing Machinery.

[46] J. Zhang, W. Ao, J. Yan, C. Rong, D. Jin, W. Wu, and Y. Li. Moss: A large-scale open microscopic traffic simulation system, 2024.

[47] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool. Trafficbots: Towards world models for autonomous driving simulation and motion prediction. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 1522–1529. IEEE, 2023.

[48] G. Zheng, H. Liu, K. Xu, and Z. Li. Learning to simulate vehicle trajectories from demonstrations. In 2020 IEEE 36th International Conference on Data Engineering (ICDE), pages 1822–1825. IEEE, 2020.

[49] G. Zheng, H. Liu, K. Xu, and Z. Li. Objective-aware traffic simulation via inverse reinforcement learning, 2022.

[50] Z. Zhong, D. Rempe, D. Xu, Y. Chen, S. Veer, T. Che, B. Ray, and M. Pavone. Guided conditional diffusion for controllable traffic simulation. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 3560–3566. IEEE, 2023.

[51] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang. Query-centric trajectory prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 17863–17873, 2023.

[52] Z. Zhou, Z. Wen, J. Wang, Y.-H. Li, and Y.-K. Huang. Qcnext: A next-generation framework for joint multi-agent trajectory prediction, 2023.