

Versatile Behavior Diffusion for Generalized Traffic Agent Simulation

Journal Title
XX(X):1–23
©The Author(s) 2024
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/



Zhiyu Huang^{1*}, Zixu Zhang^{2*}, Ameya Vaidya², Yuxiao Chen³, Chen Lv¹, and Jaime F. Fisac²

Abstract

Existing traffic simulation models often fail to capture the complexities of real-world scenarios, limiting the effective evaluation of autonomous driving systems. We introduce **Versatile Behavior Diffusion** (VBD), a novel traffic scenario generation framework that utilizes diffusion generative models to predict scene-consistent and controllable multi-agent interactions in closed-loop settings. VBD achieves state-of-the-art performance on the Waymo Sim Agents Benchmark and can effectively produce realistic and coherent traffic behaviors with complex agent interactions under diverse environmental conditions. Furthermore, VBD offers inference-time scenario editing through multi-step refinement guided by behavior priors and model-based optimization objectives. This capability allows for controllable multi-agent behavior generation, accommodating a wide range of user requirements across various traffic simulation applications. Despite being trained solely on publicly available datasets representing typical traffic conditions, we introduce conflict-prior and game-theoretic guidance approaches that enable the creation of interactive, long-tail safety-critical scenarios—essential for comprehensive testing and validation of autonomous vehicles. Lastly, we provide in-depth insights into effective training and inference strategies for diffusion-based traffic scenario generation models, highlighting best practices and common pitfalls. Our work significantly advances the ability to simulate complex traffic environments, offering a powerful tool for the development and assessment of autonomous driving technologies. Project website: <https://sites.google.com/view/versatile-behavior-diffusion>.

Keywords

Autonomous Driving, Traffic Simulation, Multi-agent Interaction, Scenario Generation, Diffusion Model

1. Introduction

Comprehensive testing is crucial for ensuring the safe deployment of autonomous driving systems. Traditionally, these assessments rely on replaying logged trajectories from driving datasets in simulation environments (Li et al. 2022). However, because other traffic participants in the replay do not react to the autonomous vehicle’s actions, log-replay methods often fail to account for interactive traffic scenarios, such as pedestrians stepping into crosswalks or vehicles negotiating right-of-way at four-way stops. This limitation introduces a significant simulation-to-reality gap, where simulated test results may not translate to real-world performance, undermining the validity of simulation testing.

To address this challenge, traffic simulators that incorporate reactive traffic behavior models, which account for interactions between agents and their surroundings, have become increasingly popular. These models enable the synthesis of scenarios that better reflect the complex dynamics of real-world traffic, including lane changes, merges, and responses to traffic controls. Conventional heuristic-based traffic modeling approaches (Treiber et al. 2000; Kesting et al. 2007) often fail to scale to the complexity of realistic traffic maneuvers, such as subtle navigation around construction zones, pedestrians stepping into crosswalks, or vehicles inching forward at congested intersections. As a result, recent efforts (Shi et al. 2022; Igl et al. 2022; Nayakanti et al. 2023; Zhang et al. 2023c; Feng et al. 2023) have increasingly shifted towards data-driven methods that leverage large-scale

driving datasets (Caesar et al. 2020; Wilson et al. 2023; Montali et al. 2023) and behavior cloning (BC) techniques to more accurately model traffic behaviors. Despite these advancements, existing methods still focus on predicting marginal open-loop trajectories for individual agents, potentially resulting in a lack of scene consistency. This means that when multiple agents’ most likely trajectories are rolled out together, unnatural interactions and even collisions can occur (Chen et al. 2022), compromising the realism and reliability of simulated scenarios. One common approach to improve scene consistency is by controlling agents’ motions in a receding horizon fashion or utilizing an auto-regressive policy (Zhou et al. 2024; Wu et al. 2024), relying on the model to re-plan and adapt to each other’s actions. However, by controlling all agents with a single shared policy, the planned trajectories often lack diversity. Consequently, generated scenarios struggle to effectively model complex, joint interactions among multiple agents. Additionally, this practice may be inefficient, as the computational requirements scale linearly with the number of traffic participants

¹Nanyang Technological University, Singapore

²Princeton University, USA

³NVIDIA Research, USA

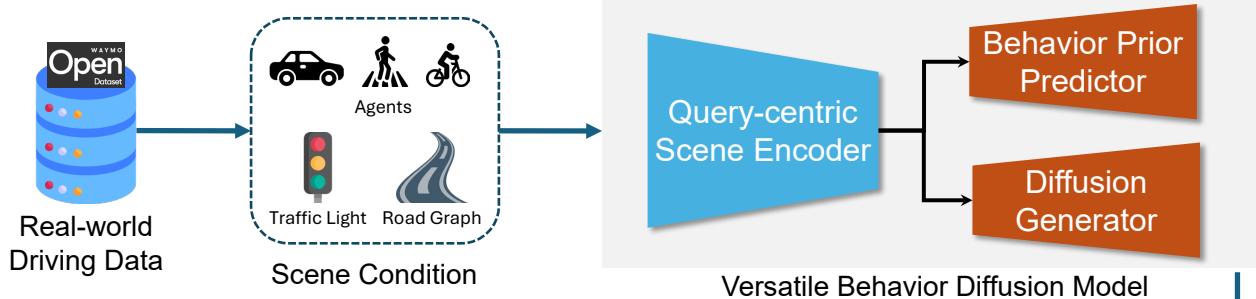
*Equal Contribution

Corresponding author:

Jaime Fernández Fisac and Chen Lv

Email: jfisac@princeton.edu, lyuchen@ntu.edu.sg

Train from Normal Offline Data



Deploy as Generalized Simulation Engine

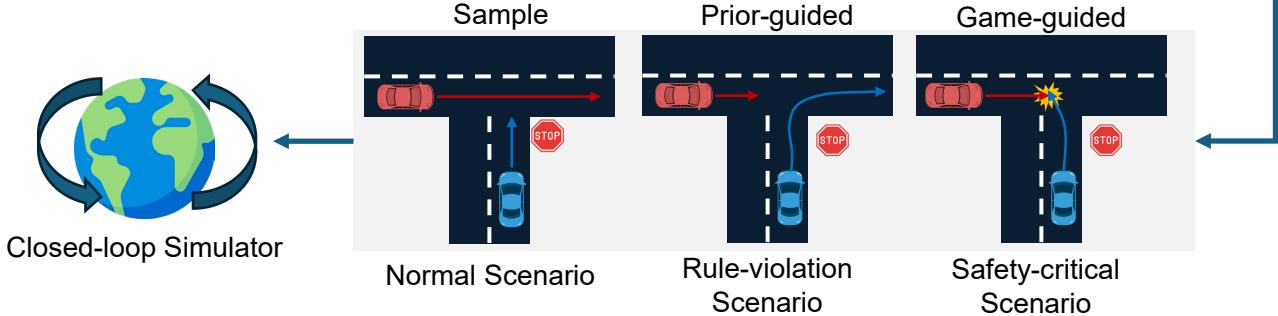


Figure 1. Illustration of the VBD framework. The model consists of a query-centric Transformer scene condition encoder, a marginal multi-modal behavior predictor for individual agents, and a diffusion generator for joint multi-agent behavior. In training, the model only utilizes commonly used driving datasets collected from public roads. In testing, the model's versatility is showcased through various purposes enabled by simple sampling and different guidance structures (e.g., prior guidance for sophisticated target agent's behavior control and game-theoretic cost guidance for adversarial behaviors). The VBD model functions as a generalized simulation engine and operates effectively in a closed-loop traffic simulator.

in the scene. Therefore, scenario generation models in traffic simulators should be designed for scene-centric prediction, where future trajectories of all agents are jointly generated within a unified and interactive framework, enabling more scalable and realistic simulation of complex traffic scenarios.

Furthermore, traffic simulators should offer versatility and enable controllable scenario generation to satisfy both explicit (e.g., goals or targets) and implicit (e.g., collision avoidance) requirements. Existing models are often specialized in generating nominal behaviors conditioned on high-level objectives, such as navigation goals (Zhang et al. 2023c). However, scenario generation models should also address the “long-tail” challenge in most datasets, where rare and dangerous situations are underrepresented (Liu and Feng 2024), potentially leading to inadequate testing in critical scenarios. Previous works in safety-critical traffic scenario generation (Cao et al. 2022; Hanselmann et al. 2022; Rempe et al. 2022; Zhang et al. 2023a) have tailored training objectives to encourage adversarial behaviors (e.g., inducing collisions with the ego agent), which limits their ability to capture the nuances of real-world traffic interactions. For instance, a model trained solely to generate aggressive driving behaviors may struggle to produce scenarios where surrounding agents attempt to avoid such adversarial actions, highlighting the constraints imposed by its learning objectives.

Therefore, a more generalized traffic simulation framework is needed to produce a wide spectrum of scenarios accommodating various requirements at inference time without necessitating retraining or — a level of flexibility that current models frequently lack. This flexibility is crucial in traffic

simulation, as it allows users to assign different modes to agents, enables the evaluation of edge cases, and facilitates the exploration of novel situations, ultimately providing a more comprehensive and customizable simulation experience. To achieve both realism and controllability in multi-agent behavior modeling, one potential solution is to first learn an optimization objective that captures nominal driving behavior, and then align scenarios to user specifications by solving an optimization problem that combines the learned objectives with additional heuristics. This approach involves modeling optimization objectives — a standard problem in inverse optimal control (IOC) or inverse reinforcement learning (IRL) — which aims to infer the underlying cost function or reward structure governing an agent’s behavior (Ng et al. 2000; Ziebart et al. 2008). Existing IOC/IRL methods have been applied to modeling trajectories of traffic agents, ranging from directly mapping scenarios to costs (Rosbach et al. 2019; Wang et al. 2023a; Huang et al. 2021) to learning cost weights for a set of handcrafted heuristics through differentiable optimization layers (Huang et al. 2023c,a; Diehl et al. 2023). However, these methods have not matched the performance achieved by behavior cloning methods (Shi et al. 2022; Nayakanti et al. 2023; Wu et al. 2024) in capturing realistic traffic behaviors.

To overcome these limitations, we leverage diffusion models (also known as score-based models) (Sohl-Dickstein et al. 2015; Song and Ermon 2019; Song et al. 2020c; Ho et al. 2020), a class of generative modeling methods that gradually recover structured data from random noise. We develop **Versatile Behavior Diffusion** (VBD), a flexible and scalable model for traffic behavior generation. To

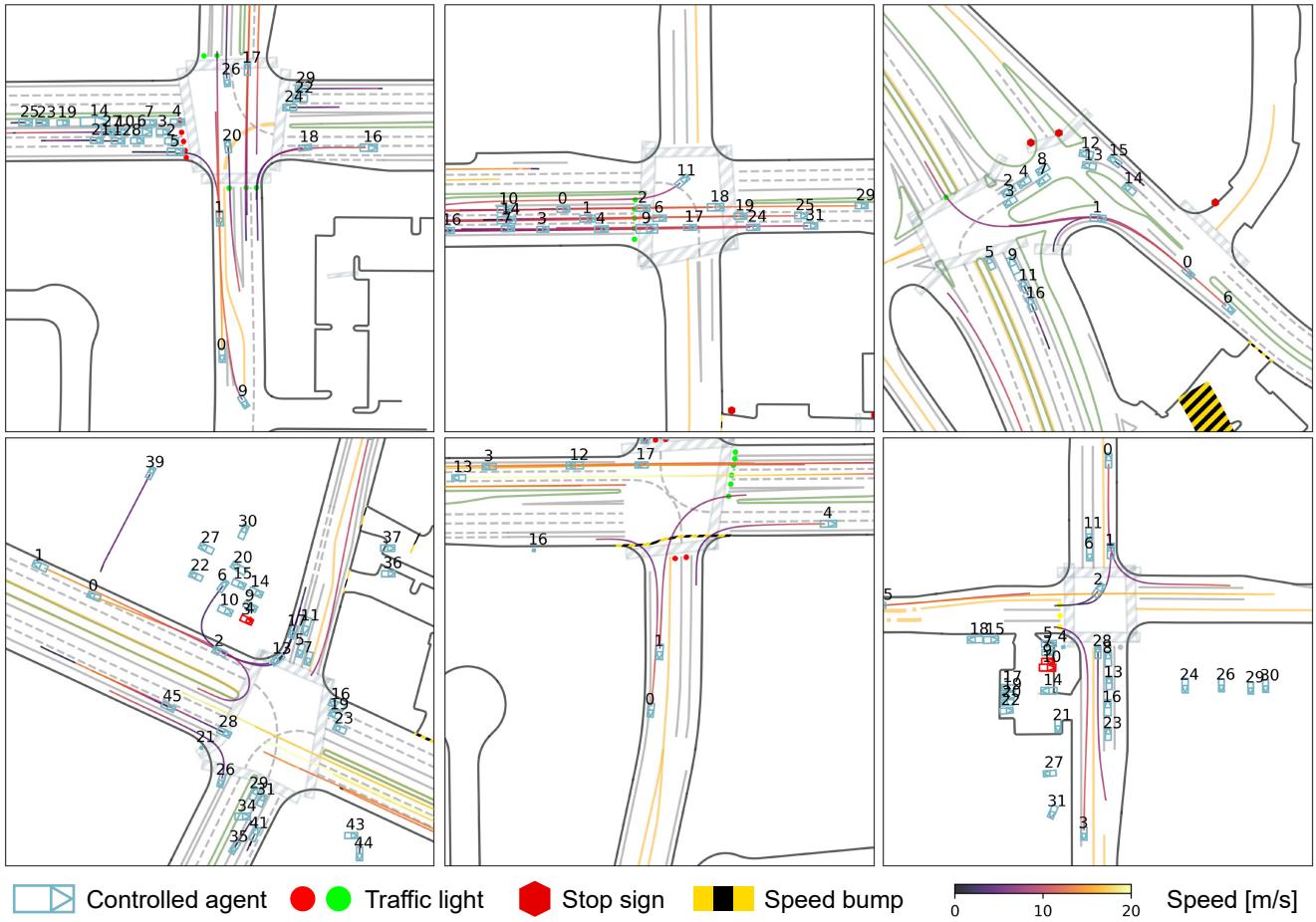


Figure 2. Performance of our VBD model on the Waymo Sim Agents task. The multi-agent diffusion policy is capable of controlling a large number of agents in an interactive and map-adherent manner for traffic simulation.

the best of our knowledge, VBD is the first open-source framework capable of generating realistic, interactive, and controllable traffic scenarios for any number of agents in a closed-loop setting. As illustrated in Figure 1, despite being trained only on commonly used public road traffic datasets, VBD can accommodate a variety of tasks—such as normal or adversarial behavior generation, counterfactual reasoning, and reactive simulation—through guided sampling combined with user-specified objectives. Our model demonstrates state-of-the-art performance on the large-scale simulated agent benchmark, as shown in Figure 2, and offers versatile generation capabilities. Additionally, we elucidate both training and inference-time design choices of VBD to provide valuable insights for future work. The primary contributions of this paper are threefold:

1. We propose **Versatile Behavior Diffusion** for generalized traffic simulation, capable of generating realistic and controllable traffic agent behaviors with excellent closed-loop simulation performance.
2. We demonstrate the versatility of VBD in simulations, showcasing its ability to generate diverse user-specified scenarios through flexible guidance schemes compatible with optimization objectives, behavior priors, and game-theoretic structures.
3. We conduct extensive empirical studies to investigate the effects of various training and inference settings on

multi-agent behavior generation using diffusion models, providing valuable insights for future research.

In the rest of this manuscript, we first establish a conceptual connection between diffusion generative modeling and imitation learning for multi-agent traffic behavior modeling, as well as compositional scenario generation in Section 3. Then, we detail the model architecture and training methodology of VBD in Section 4. Section 5 illustrates that VBD can generate realistic and interactive traffic scenarios without guidance in closed-loop simulations. Furthermore, we demonstrate multiple real-world use cases of VBD to enhance generation quality and sample user-desired behaviors through guidance in Section 6, and generate safety-critical scenarios using guidance in Section 7. Finally, we ablate important design choices necessary for high-quality generation in Section 8.

2. Related Work

2.1 Traffic Simulation

Traditional model-based or heuristic-based methods (Treiber et al. 2000; Kesting et al. 2007) often fail to capture complex agent interactions in real-world scenarios, resulting in a significant gap between simulation and reality. Consequently, there has been a growing shift towards learning-based methods to enhance the realism and interactivity of traffic simulations (Suo et al. 2021). For

example, BITS (Xu et al. 2023b) employs imitation learning to simulate agent behaviors by inferring high-level intentions and replicating low-level driving actions. The socially-controllable behavior generation model proposed in Chang et al. (2023b) focuses on simulating social and interactive behaviors. Symphony (Igl et al. 2022) integrates learning-based policies with parallel beam search to further enhance realism. TrafficBots (Zhang et al. 2023c) introduces a shared policy conditioned on specific goals to generate configurable behaviors. Trajeglish (Phlion et al. 2023) introduces a multi-agent sequence of motion tokens using a GPT-like encoder-decoder architecture, achieving state-of-the-art realism metrics. Another research direction focuses on generating safety-critical or adversarial scenarios to test the robustness of driving systems. STRIVE (Rempe et al. 2022) generates challenging scenarios that can induce collisions with the ego planner through optimization in latent space. Similarly, AdvDO (Cao et al. 2022) and KING (Hanselmann et al. 2022) utilize optimization-based methods to generate adversarial trajectories for robust planning. CAT (Zhang et al. 2023a) selects conflicting trajectories from agents' predicted behavior distributions to construct adversarial scenarios, supporting robust training of RL driving agents. However, most existing simulation models lack versatility, as they are specifically trained either for normal behaviors (maximum likelihood) or adversarial scenarios. This limitation restricts their applicability in general-purpose traffic simulators. We aim to bridge this gap by developing a generative framework that supports various tasks and requirements to enable generalizable, controllable, and realistic traffic simulation.

2.2 Behavior Prediction

Behavior or trajectory prediction models are widely utilized in both traffic simulations and driving policies (Zhang et al. 2023c; Sun et al. 2023; Feng et al. 2023). Recent advances in learning-based behavior prediction models have significantly improved the accuracy of predicting single-agent motions (Huang et al. 2022; Nayakanti et al. 2023; Zhou et al. 2023) as well as joint multi-agent interactions at the scene level (Mo et al. 2022; Shi et al. 2023; Huang et al. 2023b). Leveraging a large amount of real-world data, these models are capable of generating multi-modal distributions of possible behaviors for multiple agents in a scene. However, these prediction models often face limitations in scalability and efficiency when applied to large-scale scenarios with numerous agents. Therefore, more powerful generative models, including diffusion models (Jiang et al. 2023; Niedoba et al. 2023) and autoregressive Transformer models (Seff et al. 2023; Zhou et al. 2024), have been increasingly applied in behavior prediction and generation tasks, demonstrating superior performance in large-scale predictions and simulations. Our proposed VBD model integrates multi-modal behavior prediction as high-level intention priors, enabling the estimation of realistic behavior distributions and sophisticated control over target agents' behaviors, which facilitates the diffusion model in generating specific scenarios based on the intended behaviors of particular agents.

2.3 Diffusion Models for Traffic Simulation

Diffusion models, also known as score-based models (Sohl-Dickstein et al. 2015; Song and Ermon 2019; Ho et al. 2020; Song et al. 2020c,a), have gained widespread popularity in a variety of generative tasks, including image (Zhang et al. 2023b), audio (Kong et al. 2020), and video (Esser et al. 2023) generation. Recently, their application has extended to traffic scenario generation, where their ability to generate diverse and controllable outputs is particularly valuable. For instance, SceneDM (Guo et al. 2023) utilizes a diffusion model to generate consistent joint motions for all agents in a scene. MotionDiffuser (Jiang et al. 2023) employs a diffusion-based representation for joint multi-agent motion prediction and introduces a constrained sampling framework for controlled trajectory sampling. CTG (Zhong et al. 2023b) combines diffusion modeling with signal temporal logic (STL) rules to ensure compliance with traffic rules in generated trajectories. CTG++ (Zhong et al. 2023a) leverages Large Language Models (LLMs) to translate user queries into loss functions, guiding the diffusion model toward generating query-compliant scenarios. TRACE (Rempe et al. 2023) proposes a guided diffusion model to generate future trajectories for pedestrians, employing analytical loss functions to impose trajectory constraints. Other models, such as DiffScene (Xu et al. 2023a) and Safe-Sim (Chang et al. 2023a), adopt guided diffusion techniques with adversarial optimization to generate safety-critical traffic scenarios. Concurrent work from Waymo (Jiang et al. 2024) also explored the idea of generating joint agent trajectories for traffic scenario generation. They proposed to use amortized diffusion to improve the quality and computational efficiency for closed-loop roll-outs. While our work is closely related to these guided diffusion models, these approaches often remain limited to specific tasks and lack versatility across different simulation applications. Additionally, there has been limited conceptual exploration of diffusion models within the context of traffic simulation. We aim to advance this area by proposing training strategies for diffusion-based behavior generation models and exploring various inference strategies to enhance both realism and versatility.

3. Problem Formulation

3.1 Traffic Scenario Generation as Optimization

Consider a traffic scenario $\mathcal{S} = (\mathbf{x}, \mathbf{u}, \mathbf{c})$ with an episode length T containing a tensor of A agents' trajectories $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^A) \in \mathbb{R}^{A \times T \times D_x}$ and control sequences $\mathbf{u} = (\mathbf{u}^1, \dots, \mathbf{u}^A) \in \mathbb{R}^{A \times T \times D_u}$. The context of the scene $\mathbf{c} \in \mathbb{R}^{D_c}$ includes information regarding the road map, traffic light status, and the initial joint state of all agents x_0 , etc. Given an optimization objective $\mathcal{J}_\theta(\mathbf{x}, \mathbf{u}; \mathbf{c})$, we formulate scenario generation as a finite-horizon optimal control problem by:

$$\begin{aligned} & \min_{\mathbf{u} \in \mathbb{R}^{A \times T \times D_u}} \mathcal{J}_\theta(\mathbf{x}, \mathbf{u}; \mathbf{c}), \\ \text{s.t. } & \mathbf{x}_0 = x_0, \\ & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \forall t \in \{0, \dots, T-1\}, \end{aligned} \tag{1}$$

where f represents the discrete-time joint dynamics. If our goal is to generate realistic (statistically representative)

scenarios, the objective \mathcal{J}_θ should incentivize real-world driving behaviors, which can be achieved through statistical loss metrics or inverse reinforcement learning (IRL). Once \mathcal{J}_θ is established, we need to tractably find an optimal joint control sequence \mathbf{u} through numerical optimization or reinforcement learning.

3.2 Generative Modeling as Trajectory Optimization

Instead of solving the aforementioned IL problem in two steps, prior works (Ho and Ermon 2016; Finn et al. 2016) have shown a strong connection between IL and generative modeling under a generative-adversarial training framework. Extending the analysis by Du and Mordatch (2019), Liu et al. (2022), and Chi et al. (2023), we show that synthesizing a diffusion generative model in this IL setting can be viewed as learning the gradient descent step of a particular optimal control solver.

Consider a dataset \mathbb{D} with scenario triplets sampled independently from an unknown distribution p . Since we are interested in scenario generation given a scene context and the recorded trajectory \mathbf{x} as a control sequence \mathbf{u} under known dynamics f , we can factorize the probability density function as $p(\mathcal{S}) = p(\mathbf{u}|\mathbf{c})p(\mathbf{c})$. Under the Maximum Entropy IRL (Ziebart et al. 2008) formulation, we aim to approximate $p(\mathbf{u}|\mathbf{c})$ as the Boltzmann distribution of an optimization objective:

$$p(\mathbf{u}|\mathbf{c}) \approx p_\theta(\mathbf{u}|\mathbf{c}) := \frac{1}{Z_\theta} \exp(-\mathcal{J}_\theta(\mathbf{x}(\mathbf{u}), \mathbf{u}; \mathbf{c})), \quad (2)$$

where Z_θ is the partition function. Equation 2 resembles the Energy-Based Models (EBM) (LeCun et al. 2006; Song and Kingma 2021). Specifically, we want to learn the parameter θ of the optimization objective that maximizes the conditional log-likelihood of the dataset \mathbb{D} :

$$\theta = \arg \max_{\hat{\theta}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\theta}}(\mathbf{u}|\mathbf{c})]. \quad (3)$$

Ideally, we can employ score-matching (Hyvärinen and Dayan 2005; Vincent 2011; Song and Ermon 2019; Song et al. 2020b) to directly learn the gradient of \mathcal{J}_θ w.r.t the control (our random variable of interest) as the score function:

$$\begin{aligned} \nabla_{\mathbf{u}} \log p(\mathbf{u}|\mathbf{c}) &\approx \mathbf{s}_\theta(\mathbf{u}|\mathbf{c}) \\ &:= \nabla_{\mathbf{u}} \log p_\theta(\mathbf{u}|\mathbf{c}) \\ &= -\nabla_{\mathbf{u}} \mathcal{J}_\theta(\mathbf{x}(\mathbf{u}), \mathbf{u}; \mathbf{c}) - \cancel{\nabla_{\mathbf{u}} \log Z}^0 \end{aligned} \quad (4)$$

If $\nabla_{\mathbf{u}} \mathcal{J}_\theta$ was obtained over the entire action space, we could use it for gradient descent. However, since the dataset contains mostly near-optimal scenarios, the gradient estimation in suboptimal regions of the action space (away from demonstration data) may be inaccurate or not well-defined. To overcome this issue, a class of approaches (Du and Mordatch 2019; Song et al. 2020b; Ho et al. 2020; Song et al. 2020c,a) utilize a stochastic process to gradually diffuse p into noised distributions p_k for k steps until it becomes a known distribution $p_K = \pi$. These methods are commonly known as Diffusion models (Ho et al. 2020; Song et al. 2020a) and are later generalized as score-based models by

Song et al. (2020c). Specifically, we train a step-conditioned score function $\mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, k)$ to approximate the gradient of the log noised distribution $\nabla_{\tilde{\mathbf{u}}} \log p_k(\tilde{\mathbf{u}})$ by:

$$\theta = \arg \max_{\hat{\theta}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}, k \sim \mathcal{U}(0, K)} \mathbb{E}_{\tilde{\mathbf{u}} \sim p_k(\cdot|\mathbf{u})} [\lambda(k) \| \nabla_{\tilde{\mathbf{u}}} \log p_k(\tilde{\mathbf{u}}|\mathbf{u}) - \mathbf{s}_{\hat{\theta}}(\tilde{\mathbf{u}}|\mathbf{c}, k) \|], \quad (5)$$

where $\lambda(k)$ is a positive weighting function. At inference time, we can generate scenarios by first randomly selecting $\tilde{\mathbf{u}}$ from the known distribution π and sampling through the reverse diffusion process.

Connecting this formulation of generative modeling with trajectory optimization, we can view the forward diffusion as uplifting original data distribution into a higher-dimensional space augmented by diffusion step k . By injecting noise, we achieve good coverage over the entire action space in the final step K so that $\mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, K)$ are well defined for random $\tilde{\mathbf{u}}$. Sampling through reverse diffusion can be interpreted as stochastic gradient descent towards high-probability regions with a fixed descent direction along the diffusion step, analogous to the direct shooting method in optimal control. We note that at low noise level k , as p_k is close to the original data distribution p , $\mathbf{s}_\theta(\mathbf{u}|\mathbf{c}, k \rightarrow 0) \approx -\nabla_{\mathbf{u}} \mathcal{J}_\theta(\mathbf{x}, \mathbf{u}; \mathbf{c})$, which is the gradient we originally try to model. Therefore, the generative modeling of scenarios can be viewed as an explicit solution of IL by learning the gradient steps of trajectory optimization and solving the optimal control problem through reverse diffusion sampling.

In the remainder of the paper, without loss of generality, we consider a specific form of Diffusion model, diffusion-denoising probabilistic models (DDPM) (Ho et al. 2020), which is also known as the discrete-time variance-preserving score-based SDE (VP-SDE) (Song et al. 2020c). The equivalence between the original DDPM training objective and the score-matching loss (Equation 5) has been shown in Luo (2022).

3.3 Denoising Diffusion Probabilistic Model

The discrete-time formulation of the forward diffusion process of DDPM (Nichol and Dhariwal 2021) can be described as:

$$q(\tilde{\mathbf{u}}_1, \dots, \tilde{\mathbf{u}}_K | \mathbf{u}) := \prod_{k=1}^K q(\tilde{\mathbf{u}}_k | \tilde{\mathbf{u}}_{k-1}), \quad (6)$$

$$q(\tilde{\mathbf{u}}_k | \tilde{\mathbf{u}}_{k-1}) := \mathcal{N} \left(\tilde{\mathbf{u}}_k; \sqrt{1 - \beta_k} \tilde{\mathbf{u}}_{k-1}, \beta_k \mathbf{I} \right), \quad (7)$$

where $\beta_k \in (0, 1)$ is the k -th noise scale from a predefined noise scheduling, \mathbf{u} is the clean control action sampled from data distribution, and $\tilde{\mathbf{u}}_k$ is the noisy action samples at diffusion step k . In the final step K , the data distribution approaches an isotropic Gaussian distribution $q(\tilde{\mathbf{u}}_K) \approx \mathcal{N}(\tilde{\mathbf{u}}_K; 0, \mathbf{I})$. Let $\alpha_k = 1 - \beta_k$ and $\bar{\alpha}_k = \prod_{i=0}^k \alpha_i$, we can directly sample $\tilde{\mathbf{u}}_i$ from data \mathbf{u} without iterative diffusion via reparameterization trick:

$$\begin{aligned} \tilde{\mathbf{u}}_k &= \sqrt{\alpha_k} \tilde{\mathbf{u}}_{k-1} + \sqrt{1 - \alpha_k} \epsilon_{k-1} \\ &= \sqrt{\alpha_k} \mathbf{u} + \sqrt{1 - \bar{\alpha}_k} \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \end{aligned} \quad (8)$$

The generation process is accomplished by learning to reverse the forward diffusion process based on context

information \mathbf{c} . The reverse diffusion process starts with an isotropic Gaussian noise $q(\tilde{\mathbf{u}}_K)$ and can be expressed as:

$$p_\theta(\tilde{\mathbf{u}}_0, \dots, \tilde{\mathbf{u}}_K | \mathbf{c}) := q(\tilde{\mathbf{u}}_K) \prod_{i=1}^K p_\theta(\tilde{\mathbf{u}}_{k-1} | \tilde{\mathbf{u}}_k, \mathbf{c}), \quad (9)$$

$$p_\theta(\tilde{\mathbf{u}}_{k-1} | \tilde{\mathbf{u}}_k, \mathbf{c}) := \mathcal{N}(\tilde{\mathbf{u}}_{k-1}; \mu_\theta(\tilde{\mathbf{u}}_k, \mathcal{D}_\theta(\tilde{\mathbf{u}}_k, k, \mathbf{c})), \sigma_k^2 \mathbf{I}), \quad (10)$$

where μ_θ calculates the posterior mean of the noise at $k-1$ step from $\tilde{\mathbf{u}}_k$ and DDPM denoiser output $\mathcal{D}_\theta(\cdot)$, and σ_k is the standard deviation according to the fixed noise schedule. Specifically, the denoiser $\mathcal{D}_\theta(\cdot)$ estimates the clean action trajectory sample $\hat{\mathbf{u}}_k$ from the current noisy sample $\tilde{\mathbf{u}}_k$, according to which the mean of the previous noisy sample $\tilde{\mathbf{u}}_{k-1}$ can be derived as follows:

$$\mu_k := \frac{\sqrt{\bar{\alpha}_{k-1}} \beta_k}{1 - \bar{\alpha}_k} \hat{\mathbf{u}}_k + \frac{\sqrt{\alpha_k}(1 - \bar{\alpha}_{k-1})}{1 - \bar{\alpha}_k} \tilde{\mathbf{u}}_k. \quad (11)$$

3.4 Controllable and Compositional Generation

In many applications, we want to generate scenarios that satisfy a specific user requirement y without retraining the model. For example, y can be defined as the goal or the reference path for individual agents, or it can describe a soft constraint, such as obeying the speed limit or avoiding collisions. From the perspective of optimal control (Equation 1), we modify the optimization objective to: $\mathcal{J}_\theta(\mathbf{x}, \mathbf{u}; \mathbf{c}) + \mathcal{J}_y(\mathbf{x}, \mathbf{u}; \mathbf{c})$. Plugging into the EBM representation, we obtain a new conditional distribution: $p(\mathbf{u}|\mathbf{c}, y) \propto p(\mathbf{u}|\mathbf{c})p(y|\mathbf{u}, \mathbf{c})$, where $p(\mathbf{u}|\mathbf{c})$ is the data distribution we approximated through generative modeling and $p(y|\mathbf{u}, \mathbf{c})$ is the likelihood of y . This immediately resembles the compositionality in EBM (Du and Mordatch 2019) and Classifier Guidance in diffusion model (Dhariwal and Nichol 2021). Specifically, we can sample the reverse process with a conditional score function:

$$\begin{aligned} \nabla_{\tilde{\mathbf{u}}} \log p_k(\tilde{\mathbf{u}}|\mathbf{c}, y) &\approx \mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, y, k) \\ &= \mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, k) + \nabla_{\tilde{\mathbf{u}}} \log p_k(y|\tilde{\mathbf{u}}, \mathbf{c}), \end{aligned} \quad (12)$$

where $p_k(y|\tilde{\mathbf{u}}, \mathbf{c})$ is the likelihood of y given the noised action $\tilde{\mathbf{u}}$ at step k . It is important to note that $p_k(y|\tilde{\mathbf{u}}, \mathbf{c})$ is not equivalent to the likelihood of y in the data distribution $p(y|\mathbf{u}, \mathbf{c})$, therefore it is typically required to train a separate model (Dhariwal and Nichol 2021; Janner et al. 2022). However, we can utilize practical approximation to the gradient of noised likelihood with the gradient of \mathcal{J}_y for guidance, which enables flexible composition and controllability with additional objectives without training.

In the reverse diffusion process, the guidance is implemented by subtly altering the predicted mean of the model at each denoising step (Zhong et al. 2023b,a). Initially, we can directly approximate $\log p_k(y|\tilde{\mathbf{u}}, \mathbf{c}) \approx \log p(y|\tilde{\mathbf{u}}, \mathbf{c}) = \mathcal{J}_y(\mathbf{x}(\tilde{\mathbf{u}}), \tilde{\mathbf{u}}; \mathbf{c})$; therefore, the denoising mean is modified to impose guidance in the form of a score function \mathcal{J}_y as:

$$\tilde{\mu}_k = \mu_k + \lambda \sigma_k \nabla_{\mu_k} \mathcal{J}_y(\mu_k), \quad (13)$$

where λ is a parameter that controls the strength of the guidance.

However, calculating the gradient based on the mean of noisy actions is difficult and inaccurate, and thus manipulating the noise mean using the noisy gradient can result in errors and instability. To address this, we utilize an alternative approach (Jiang et al. 2023) to approximate $\log p_k(y|\tilde{\mathbf{u}}, \mathbf{c}) \approx \mathcal{J}_y(\mathbf{x}(\mathcal{D}_\theta(\tilde{\mathbf{u}})), \mathcal{D}_\theta(\tilde{\mathbf{u}}); \mathbf{c})$. This means calculating the objective function using the one-step generation result from the denoiser rather than the noisy mean. The modification in the denoising step is expressed as:

$$\tilde{\mu}_k = \mu_k + \lambda \sigma_k \nabla_{\tilde{\mathbf{u}}_k} \mathcal{J}_y(\mathcal{D}_\theta(\tilde{\mathbf{u}}_k)), \quad (14)$$

where the gradient $\nabla_{\tilde{\mathbf{u}}_k}$ is calculated with respect to the noisy actions $\tilde{\mathbf{u}}_k$ and necessitates differentiation through the denoiser \mathcal{D}_θ .

4. Versatile Behavior Diffusion

4.1 Model Structure

The **Versatile Behavior Diffusion** (VBD) model consists of three main components, as illustrated in Figure 3. The scene encoder $\mathcal{E}_\phi : \mathbf{c} \mapsto \hat{\mathbf{c}}$ encodes the scene context \mathbf{c} into its latent representation $\hat{\mathbf{c}}$ using query-centric attention Transformers (Shi et al. 2023). Leveraging rich scene context information from encoder, the denoiser $\mathcal{D}_\theta : (\hat{\mathbf{c}}, \tilde{\mathbf{u}}, k) \mapsto \hat{\mathbf{u}}$ directly predict a joint clean control sequence $\hat{\mathbf{u}}$ from $\hat{\mathbf{c}}$ and noised control $\tilde{\mathbf{u}}$ at step k . The behavior predictor $\mathcal{P}_\psi : (\hat{\mathbf{c}}, \{\zeta^i\}_{i=1}^M) \mapsto \{\text{Cat}_M(\hat{\mathbf{u}}^a, \hat{\omega}^a)\}_{a=1}^A$ predicts an M -mode *marginal* categorical trajectory distribution of *each agent* from $\hat{\mathbf{c}}$ with the help of a set of representative static end-point anchors $\{\zeta^i\}_{i=1}^M$ extracted from data (Shi et al. 2023). All three modules utilize a stack of query-centric self-attention and cross-attention blocks for flexibility and scalability. Details regarding the model architecture are illustrated below.

Scene Context. The scene conditions are divided into three categories: agents $\mathbf{c}_a \in \mathbb{R}^{A \times T_h \times D_a}$, map polylines $\mathbf{c}_{m,pl} \in \mathbb{R}^{M_l \times M_p \times D_p}$ and traffic lights $\mathbf{c}_{m,tl} \in \mathbb{R}^{M_t \times D_t}$. Here, T_h denotes the number of historical steps, M_l the number of polylines, M_p the number of waypoints per polyline, M_t the number of traffic lights, and $M_l + M_t = M$ represents the combined count of map elements. The feature sizes for agents, polylines, and traffic lights are represented by D_a , D_p , and D_t , respectively. The agent history tensor records each agent's historical state, including x, y coordinates, heading angle (ψ), velocities (v_x, v_y), and bounding box dimensions (l, w, h), along with the agent type. Each map polyline, comprising M_p waypoints, includes attributes like x, y coordinates, direction angle, the traffic light state controlling the lane, and lane type. The traffic lights tensor encompasses x, y coordinates of stop points and the state of each traffic light. Before encoding, positional attributes of all elements are converted into their local coordinate systems; for agents, the reference point is their last recorded state, and for map polylines, it is the location of the first waypoint.

Scene Encoder. We encode the agent history tensor utilizing a shared GRU network, which is then combined with the agent type embedding. For map polylines, an MLP is employed for encoding, followed by max-pooling along the waypoint axis; for traffic lights, we only encode their light status using an MLP. These tensors are then concatenated to

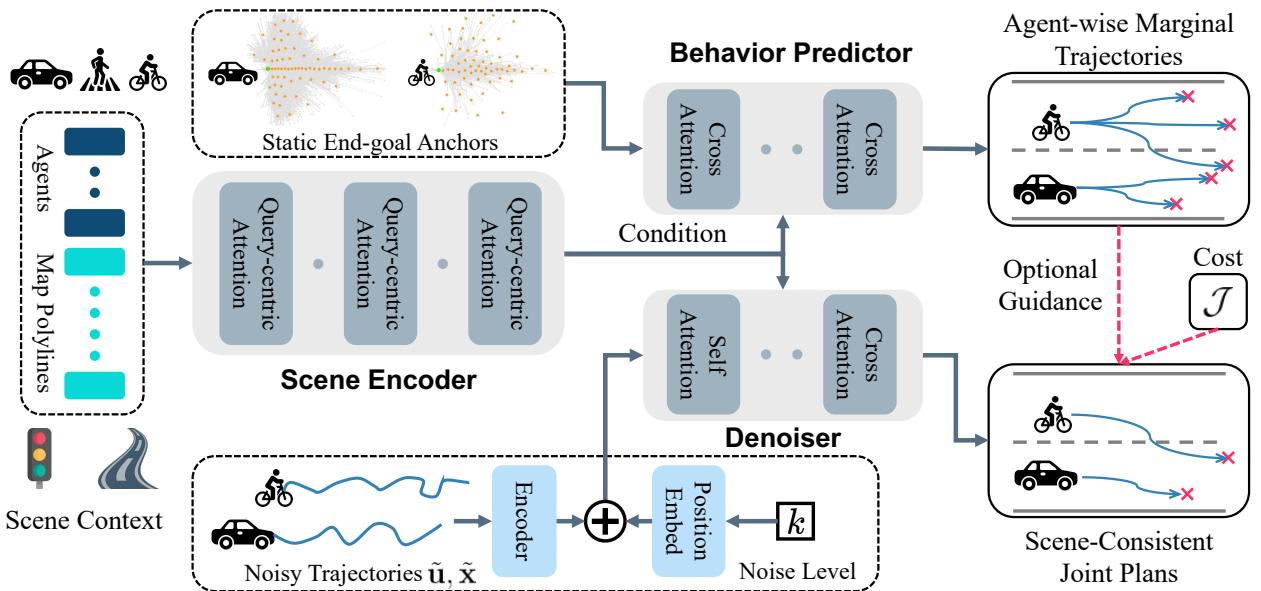


Figure 3. Neural network structure of the proposed VBD model. The input scenario tokens are encoded through a query-centric Transformer scene context encoder. The behavior predictor generates marginal multi-modal trajectories. The denoiser predicts the joint multi-agent future trajectories while attending to themselves and the condition tokens. During inference, the predicted behavior priors or user-defined model-based objectives \mathcal{J} can be used to guide the denoising process to generate desired scenarios.

form the initial scene encoding. The initial scene encoding is further processed using L_E query-centric Transformer layers to symmetrically encode the interrelationships among scene components, resulting in a uniform dimension for scene features $\hat{\mathbf{c}} \in \mathbb{R}^{(A+M) \times D}$. In this approach, each scene element is translated into its local coordinate system and encoded with query-centric features, and the relative position of each pair of scene elements is calculated and encoded as edge attributes.

Denoiser. The denoiser $\mathcal{D}_\theta(\tilde{\mathbf{u}}(k), k, \hat{\mathbf{c}})$ receives as input the noised agent actions $\tilde{\mathbf{u}}(k) \in \mathbb{R}^{A \times T \times 2}$ from the clean actions $\mathbf{u}(0) = f^{-1}(\mathbf{x}(0))$, which are computed using an inverse dynamic model f^{-1} from the ground truth states $\mathbf{x}(0) \in \mathbb{R}^{A \times T \times 4}$. Each agent's action at every timestep $u = [\dot{v}, \psi]^T$ consists of acceleration and yaw rate, while the state comprises coordinates, heading, and velocity (x, y, ψ, v) . The denoiser also takes in the noise level k and the encoded scene conditions $\hat{\mathbf{c}}$ as inputs. It encompasses several Transformer layers with both self-attention and cross-attention. Initially, the noised actions $\tilde{\mathbf{u}}_k$ are converted to noised states $\tilde{\mathbf{x}}(k) = f(\tilde{\mathbf{u}}(k))$, encoded via an MLP and concatenated with the noise level embedding. Then, a self-attention Transformer module is employed to model the joint distribution of future plans across agents. To maintain closed-loop rollout causality, a causal relationship mask (Huang et al. 2023a) is used in the self-attention module, which ensures that information from future timesteps cannot be utilized at the current timestep. Furthermore, a cross-attention Transformer module is used to model the scene-conditional distribution, by relating the noisy trajectories to the encoded scene conditions. This decoding block can be repeated several times, and the final embedding is fed into an MLP decoder to predict the denoised actions $\hat{\mathbf{u}}(0)$ and clean states $\hat{\mathbf{x}}(0)$ through a dynamic function f .

Behavior Predictor. The behavior predictor \mathcal{P}_ψ generates the marginal distributions of possible behaviors for

individual agents by directly decoding from the encoded scene conditions $\hat{\mathbf{c}}$. The predictor is composed of L_P cross-attention Transformer decoder layers. To accurately predict the probabilities of possible goals, we utilize type-wise static anchors (end goals) $\{\zeta^i\}_{i=1}^M$ extracted from data (Shi et al. 2023) as the modality query inputs to the initial Transformer decoder layer. The predictor iteratively refines its predictions through several decoding layers and finally generates M trajectories over the future time for each agent along with their associated scores, represented as $\{\text{Cat}_M(\hat{\mathbf{u}}^a, \hat{\omega}^a)\}_{a=1}^A \in \mathbb{R}^{A \times M \times T \times 5}$. To ensure the kinematic feasibility of these goals, the model predicts action trajectories, which are converted into state trajectories using the same dynamic function f . Each waypoint in the state trajectory contains the state (x, y, ψ, v) and the probability of the trajectory $\hat{\omega}$.

4.2 Model Training

We implement a multi-task learning framework that concurrently trains the encoder, denoiser, and predictor components of our model. To train the denoiser, we aim to minimize the denoising loss:

$$\mathcal{L}_{\mathcal{D}_\theta} = \mathbb{E}_{S \sim p, k \sim \mathcal{U}(0, K)} \mathbb{E}_{\tilde{\mathbf{u}} \sim p_k(\cdot | \mathbf{u})} [\lambda(k) \mathcal{SL}_1(\hat{\mathbf{x}}(\mathcal{D}_\theta(\hat{\mathbf{c}}, \tilde{\mathbf{u}}, k)) - \mathbf{x})], \quad (15)$$

which is defined as the Smooth L1 loss between ground-truth trajectories \mathbf{x} and the trajectories $\hat{\mathbf{x}}$ rollout from $\hat{\mathbf{u}}$. Essentially, the denoiser is trained to recover the clean trajectories under various noise levels. At each training step, noise level k is sampled and applied to corrupt the ground-truth trajectories, and the denoiser is optimized to predict the denoised trajectories from the corrupted trajectories. Since the model predicts scene-level joint trajectories, all agent trajectories are affected by the same noise level. The

design of the cost function plays a crucial role in generation performance and is later ablated in [Section 8.1](#).

In addition, we utilize a novel log noise schedule defined as:

$$\bar{\alpha}_k = \frac{f(k)}{f(0)}, f(k) = \log\left(\frac{K + K\delta}{k + K\delta}\right), k = [0, 1, \dots, K], \quad (16)$$

where K is the maximum diffusion steps and δ is a scale factor that controls the variance changing rate of the log noise schedule. This schedule ensures that the signal-to-noise ratio (SNR) remains sufficiently low, preventing degradation in inference performance due to short-cut learning. Empirically, we found the performance of this noise schedule is significantly better than the traditional Cosine scheduler ([Nichol and Dhariwal 2021](#)). A more detailed discussion of the noise schedule choice can be found in [Section 8.2](#). The full training procedure of the denoiser is described in [Algorithm 1](#).

Algorithm 1 Training process of denoiser

Require: Denoiser \mathcal{D}_θ , dataset D , denoising steps K , dynamics function f , inverse dynamics function f^{-1}

- 1: **for** each training iteration **do**
- 2: $\mathbf{x}, \mathbf{c} \sim D$ ▷ Sample from dataset
- 3: Get action trajectory: $\mathbf{u} = f^{-1}(\mathbf{x})$
- 4: $k \sim \mathcal{U}(0, K)$, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ ▷ Sample noise level and Gaussian noise
- 5: Add noise to ground-truth: $\tilde{\mathbf{u}}_k = \sqrt{\bar{\alpha}_k} \mathbf{u} + \sqrt{1 - \bar{\alpha}_k} \epsilon$
- 6: Predict denoised trajectory: $\hat{\mathbf{u}} = \mathcal{D}_\theta(\tilde{\mathbf{u}}_k, k, \mathbf{c}, f)$, $\hat{\mathbf{x}} = f(\hat{\mathbf{u}})$
- 7: Compute loss: $\mathcal{L}_{\mathcal{D}_\theta} = \mathcal{SL}_1(\hat{\mathbf{x}} - \mathbf{x})$ ▷ Use smooth L1 loss
- 8: Update denoiser parameters θ
- 9: **end for**

Training a denoiser with the scene encoder directly can be unstable, partially because the denoiser focuses on structured data from the noisy trajectories rather than the information from context encoding. To address this issue, we suggest incorporating an additional task in the model to predict marginal multi-modal trajectories, which can more effectively attend to the context encoding. This setting not only stabilizes training and enhances the overall learning performance but also provides behavior priors for individual agents. To train the behavior predictor \mathcal{P}_ψ , we follow the multi-trajectory-prediction (MTP) loss setting. This involves selecting the best-predicted mode m^* that most closely matches the ground truth trajectory of each agent. To determine the best-predicted mode for an agent, the following criterion is applied:

$$m^* = \begin{cases} \arg \min_i \|\mathbf{a}\mathbf{c}^i - \mathbf{x}_T\|, & \text{if } \mathbf{x}_T \text{ is valid,} \\ \arg \min_i \|\sum_t (\hat{\mathbf{x}}_t^i - \mathbf{x}_t)\|, & \text{otherwise,} \end{cases} \quad (17)$$

where $\mathbf{a}\mathbf{c}^i$ is the static anchor point, \mathbf{x}_t is the ground-truth point of the trajectory, and $\hat{\mathbf{x}}_t^i$ is the predicted trajectory point. This means that if the ground-truth trajectory endpoint is invalid, the predicted trajectory with the smallest average displacement error is selected; otherwise, the trajectory corresponding to the closest anchor point is selected.

Subsequently, trajectories are chosen from the multi-modal predictions based on the indices m^* , and the Smooth L1 loss is computed between these selected trajectories and the ground-truth trajectories. For the training of the scoring function, cross-entropy loss is utilized, comparing the predicted logits with the given modes. The prediction loss is formulated as:

$$\mathcal{L}_{\mathcal{P}_\psi} = \mathbb{E}_{\mathcal{S} \sim p} \left[\sum_{a=1}^A \mathcal{SL}_1(\hat{\mathbf{x}}(\hat{\mathbf{u}}^{a, m^*}) - \mathbf{x}^a) + \beta CE(m^*, \hat{\omega}^a) \right], \quad (18)$$

where β is a hyperparameter. Note that this loss function is computed marginally, and time steps lacking ground-truth data or invalid agents are excluded from the loss calculation.

The total loss function for the multi-task learning model is formulated as:

$$\mathcal{L} = \mathcal{L}_{\mathcal{D}_\theta} + \gamma \mathcal{L}_{\mathcal{P}_\psi}, \quad (19)$$

where γ is a hyperparameter to balance the importance of tasks.

4.3 Experimental Setup

System Dynamics. The diffusion model operates in the action space, and we assume that there is a dynamic function that can translate actions to physical states $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$. We utilize unicycle dynamics as the system dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ to roll out actions into states. This model is shared across all types of agents, including vehicles, pedestrians, and cyclists. The current state of an agent is defined by its global coordinates (x, y) , yaw angle ψ , and velocities v_x, v_y . Given the action of an agent, including acceleration \dot{v} and yaw rate $\dot{\psi}$, and the time length for one step Δt , the next-step state of the agent is calculated using the following forward dynamics f , expressed as:

$$\begin{aligned} x(t+1) &= x_t + v_x(t)\Delta t, \\ y(t+1) &= y_t + v_y(t)\Delta t, \\ \psi(t+1) &= \psi(t) + \dot{\psi}\Delta t, \\ v(t+1) &= \sqrt{v_x(t)^2 + v_y(t)^2} + \dot{v}\Delta t, \\ v_x(t+1) &= v(t+1) \cos \psi(t+1), \\ v_y(t+1) &= v(t+1) \sin \psi(t+1). \end{aligned} \quad (20)$$

Since each operation in the dynamics function is differentiable, it can be integrated as a layer in the network to convert predicted actions into states. Furthermore, we employ the inverse dynamics function f^{-1} to calculate actions from ground-truth states, which is formulated as:

$$\begin{aligned} \dot{v}(t) &= \frac{v(t+1) - v(t)}{\Delta t}, \\ v(t) &= \sqrt{v_x(t)^2 + v_y(t)^2}, \\ \dot{\psi}(t) &= \frac{\psi(t+1) - \psi(t)}{\Delta t}. \end{aligned} \quad (21)$$

Throughout our experiment, we normalize the acceleration by 1 m/s^2 and the yaw rate by 0.15 rad/s .

Dataset. We conduct the experiments on the Waymo Open Motion Dataset (WOMD) ([Ettinger et al. 2021](#)),

which comprises 486,995 nine-second logged real-world traffic scenarios for training and 44,097 scenarios for validation. The dataset provides trajectories for all agents and corresponding vectorized maps for each scenario. In the Waymo Sim Agents task (Montali et al. 2023), we evaluate our model’s closed-loop simulation performance across 44,920 testing scenarios. In addition, we select 500 scenarios from the WOMD Validation Interactive split for additional experiments and ablation studies.

Model Inputs. For the scene conditions, we consider $A = 32$ agents, $M_l = 256$ map polylines (each containing $M_p = 30$ waypoints), and $M_t = 16$ traffic lights in the scene. The flexibility of Transformer blocks allows VBD to be adapted to any number of agents during inference. VBD generates $T = 80$ steps of future control sequences with step size 0.1s based on only the current state and up to 1 s of past trajectories ($T_h = 11$). Empirically, we observe discarding entire past trajectories and only keeping the current state during inference leads to better closed-loop performance by addressing potential causal confusion issue in closed-loop testing (Cheng et al. 2024). The effect of history dropout is ablated in Section 8.3.

Model Details. The scene encoder contains $L_E = 6$ query-centric-attention Transformer layers, and the embedding dimension is $D = 256$. The behavior predictor comprises $L_P = 4$ cross-attention Transformer layers and generates 64 possible trajectories for each agent along with respective probability estimates. The denoiser includes two decoding blocks with four Transformer layers in total. Action sequences are effectively shortened to $T_f = 40$ from T , by repeating actions over two time steps, which considerably reduces computational demands while ensuring high accuracy.

Training Details. The proposed log variance schedule is adopted in the diffusion process, employing $K = 50$ diffusion steps, $\bar{\alpha}_{min} = 1e - 9$, and $\delta = 0.0031$. For model training, the hyperparameters include a total loss function coefficient $\gamma = 0.5$ and a predictor loss coefficient $\beta = 0.05$. The model is trained using an AdamW optimizer with a weight decay of 0.01. The initial learning rate is set at 0.0002 and decays by 0.02 every 1,000 training steps, and a linear warm-up is employed for the first 1,000 steps. The total number of epochs for training is 16. Gradient clipping is implemented with a norm limit set to 1.0. The training of the model utilizes BFloat16 Mixed precision on 8 NVIDIA L40 GPUs, with an effective batch size of 96 across all GPUs.

5. VBD as a Closed-loop Traffic Simulator

In this section, we demonstrate the VBD’s capability and generality in generating interactive and realistic traffic scenarios through a closed-loop evaluation using the large-scale Waymo Open Motion Dataset (WOMD) (Ettinger et al. 2021). For each scene, we initialize agent states from WOMD and leverage the Waymax simulator (Gulino et al. 2023) for scenario roll-out. At each time of inference, VBD generates 8-second plans for all agents in the scene via the standard DDPM denoising scheme. All agents replan at 1 Hz using a receding horizon approach. For details on the closed-loop traffic simulation process, refer to Algorithm 2. Furthermore, our qualitative simulation results in Figure 2

highlight VBD’s ability to simulate high-quality traffic scenarios in complex intersections with diverse agent counts, ensuring adherence to maps and traffic controls.

Algorithm 2 Closed-loop traffic simulation with VBD

Require: Encoder \mathcal{E}_ϕ , Denoiser \mathcal{D}_θ , Initial Scene \mathbf{c}_0 , Denoising Steps K , Simulation Steps T_s , Replan Steps Δt_r .

```

1: for  $t \leftarrow 0, \dots, T_s$  do
2:    $t_r = t \bmod \Delta t_r$             $\triangleright$  Relative time for replanning
3:   if  $t_r = 0$  then           $\triangleright$  Check if time to replan
4:      $\hat{\mathbf{c}}_t \leftarrow \mathcal{E}_\phi(\mathbf{c}_t)$        $\triangleright$  Encode scene context
5:      $\tilde{\mathbf{u}}_K \sim \mathcal{N}(0, \mathbf{I})$          $\triangleright$  Sample random noise
6:     for  $k \leftarrow K, \dots, 1$  do
7:        $\hat{\mathbf{u}}_0 \leftarrow \mathcal{D}_\theta(\tilde{\mathbf{u}}_k, \hat{\mathbf{c}}_t, k)$        $\triangleright$  Denoise
8:       Sample  $\tilde{\mathbf{u}}_{k-1}$  using Equation 10
9:     end for
10:    else
11:      Continue using previous control inputs  $\tilde{\mathbf{u}}_0$ 
12:    end if
13:     $\mathbf{c}_{t+1} = \text{Step}(\mathbf{c}_t, \tilde{\mathbf{u}}_0^{t_r})$        $\triangleright$  Step simulator with
14:    generated actions
end for

```

5.1 Results on Waymo Sim Agents Challenge

We begin by evaluating the scalability of VBD’s generation quality and interaction modeling capabilities using the Waymo Open Dataset Sim Agents Benchmark (Montali et al. 2023), which comprises 44,920 testing scenes from WOMD. In each scene, we are required to simulate 32 independent closed-loop rollouts of future scenarios for up to 128 agents over an 8-second horizon, conditioning on their trajectories from the preceding 1-second and the corresponding map contexts. Throughout the evaluation, VBD controls up to 64 agents surrounding the labeled self-driving vehicle in each scene, while a constant velocity policy governs any additional agents. By adopting this approach, we significantly reduce computational requirements when sampling millions of scenarios, leveraging the fact that most WOMD scenes contain fewer than 64 dynamic agents, with the remainder being stationary. In practice, we only utilize the current state of all agents by dropping out their past trajectories to overcome the causal confusion. The effectiveness of this strategy will be discussed in Section 8.4.

Evaluation Metrics. We follow the official evaluation metrics of the Waymo Sim Agents benchmark (Montali et al. 2023) encompassing kinematic, interactive, and map-based realism metrics, and a meta realism metric is calculated as a weighted sum of these features. Specifically, kinematic tests measure the likelihood of simulated agents’ linear and angular speed and acceleration compared to the ground truth. The minimum average displacement error (minADE) is also utilized to quantify the positional accuracy relative to the original scenarios.

Results. We evaluate the performance of both diffusion generator and behavior predictor in the Waymo Sim Agents benchmark, and the results are presented in Table 1. We compare the performance of VBD with state-of-the-art autoregressive generative models, including BehaviorGPT

Table 1. Testing Results on the 2024 Waymo Sim Agents Benchmark

Model	Kinematics ↑						Interaction ↑			Map-Based ↑		
	Realism Meta ↑	Linear Speed	Linear Accel	Ang. Speed	Ang. Accel	Dist to Obj	Collision	TTC	Dist to roadedge	Offroad	minADE [m]↓	
<i>Logged Oracle</i>	-	0.476	0.478	0.578	0.694	0.476	1.000	0.883	0.715	1.000	-	
SMART	0.751	0.365	0.406	0.423	0.584	0.377	0.966	0.832	0.659	0.936	1.545	
BehaviorGPT	0.747	0.361	0.337	0.481	0.554	0.383	0.954	0.831	0.670	0.935	1.415	
GUMP	0.743	0.357	0.411	0.509	0.635	0.371	0.940	0.828	0.669	0.903	1.604	
MVTE	0.730	0.351	0.353	0.497	0.600	0.374	0.905	0.831	0.666	0.907	1.677	
TrafficBotsV1.5	0.698	0.336	0.350	0.451	0.584	0.360	0.808	0.821	0.642	0.913	1.882	
SceneDiffuser	0.703	0.310	0.389	0.459	0.560	0.349	0.917	0.815	0.634	0.833	1.767	
VBD (Ours)	0.720	0.359	0.366	0.420	0.522	0.368	0.934	0.815	0.651	0.879	1.474	

Table 2. Closed-loop Simulation Results of VBD on Reactive Simulation

Ego planner	Model	Collision w/ ego [%]	Off-road [%]	ADE [m]
Log-playback	Marginal Predictor	10.60	4.49	0.979
	VBD Denoiser	4.80	1.43	1.082
IDM-route	Marginal Predictor	13.20	5.38	1.070
	VBD Denoiser	8.40	2.26	1.107

(Zhou et al. 2024), SMART (Wu et al. 2024), and GUMP (Hu et al. 2024), behavior cloning models, such as MVTE (Wang et al. 2023b) and TrafficBotsV1.5 (Zhang et al. 2024), and concurrent diffusion-based model SceneDiffuser (Jiang et al. 2024). We demonstrate that the simulation performance of the VBD model is comparable to state-of-the-art autoregressive generative models and outperforms other motion prediction models. In addition, we observe better metrics across almost all categories compared to SceneDiffuser (Jiang et al. 2024) baseline from Waymo. Notably, our VBD model achieves this with fewer parameters than these GPT-style models, achieving a balance between performance and computational efficiency. We present a selection of qualitative simulation results in Figure 2, showcasing the model’s ability to generate interactive and realistic traffic scenarios. The results in the Waymo Sim Agents task suggest that our VBD model is reliable at generating realistic agent interactions through its joint multi-agent diffusion policy. Additional simulation videos can be found on the [Project Website](#).

5.2 Reactive Simulation with AV Planner

A critical aspect of traffic simulation is ensuring realistic agent responses to the ego vehicle’s maneuvers, regardless of the ego’s underlying policy. This experiment aims to quantify the impact of scene-centric representation in VBD on simulation realism. We conduct reactive simulation tests, where the ego vehicle is controlled by an external planner while surrounding agents are controlled by our models. The evaluation uses a subset of 500 WOMD interactive validation scenarios, previously selected by Zhang et al. (2023a). Each simulation selects up to 32 agents from the original scene, excluding any additional agents, and simulations run for 8 seconds with a 1 Hz replanning frequency. For a fair comparison, we use the co-trained marginal behavior predictor as a baseline. This predictor shares the same environment context encoding and similar architecture as

the denoiser. In the baseline policy, we sample maximum likelihood trajectories for each agent and replan at 1 Hz.

Evaluation Metrics. We employ several metrics from the Waymax simulator (Gulino et al. 2023) to evaluate the performance of our model. Specifically, we track three key indicators:

- *Off-road incidents*: The percentage of agents that deviate from designated drivable areas within the map.
- *Collisions with ego*: The frequency of collisions between agents and the ego vehicle, reflecting the responsiveness of the behavior simulation engine.
- *Log divergence*: A measure of the discrepancy between simulated and recorded behaviors, calculated using the average displacement error (ADE).

Results. We assessed the reactivity of our model to the ego vehicle’s actions under two distinct control scenarios: a log-playback from WOMD and an IDM-route planner. According to our findings, summarized in Table 2, both the marginal predictor and VBD denoiser demonstrated comparable levels of log divergence (ADE), indicating their ability to mimic realistic trajectories. However, the diffusion joint policy showed notable superiority over the marginal behavior prediction approach in terms of reactivity, yielding fewer collisions with the ego vehicle. Interestingly, when utilizing the IDM-route planner – known for its deviations from human-like driving patterns and actual trajectories – the performance suffered. Conversely, employing a log-playback planner, which closely resembles the training data distribution, enabled our VBD model to exhibit highly reactive behaviors in response to the ego vehicle’s actions.

6. VBD as a Controllable Scenario Generator

A key strength of the VBD model lies in its capacity to generate customizable traffic scenarios without requiring retraining. This flexibility stems from the guided generation, allowing the model to adapt behaviors based on user-specified objectives, such as avoiding collisions or creating

adversarial scenarios. By integrating these objectives into the denoising process, the VBD model provides a versatile and potent tool for simulating diverse traffic scenarios on demand. As outlined in Algorithm 3, users can define cost objectives \mathcal{J}_y , enabling compositional optimization via the classifier guidance method (Dhariwal and Nichol 2021). This involves iteratively refining the noisy $\tilde{\mathbf{u}}$ using both denoiser outputs and gradients of the form $\nabla_{\tilde{\mu}_k} \mathcal{J}_y(\mathcal{D}\theta(\tilde{\mu}_k))$. Throughout this process, the denoiser ensures scenario realism, while the user's objective encourages the model to explore desired modes. Subsequent sections will demonstrate the application of VBD's capabilities across various simulation tasks. Specifically, our guided generation tasks utilize $N_g = 5$ gradient steps and a fixed scaling parameter $\lambda = 0.1$ during all denoising operations.

Algorithm 3 Guided sampling with objective function

Require: Denoiser \mathcal{D}_θ , objective function \mathcal{J}_y , diffusion steps K , gradient steps N_g , scaling parameter λ , standard deviation σ_k

- 1: $\tilde{\mathbf{u}}_K \sim \mathcal{N}(0, \mathbf{I})$ ▷ Sample initial trajectory
- 2: **for** $k \leftarrow K$ to 1 **do**
- 3: $\hat{\mathbf{u}} \leftarrow \mathcal{D}_\theta(\tilde{\mathbf{u}}_k, k, \mathbf{c})$ ▷ Predict denoised control sequence
- 4: $\tilde{\mu}_k \leftarrow \frac{\sqrt{\alpha_k(1-\bar{\alpha}_{k-1})}}{1-\bar{\alpha}_k} \tilde{\mathbf{u}}_k + \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k} \hat{\mathbf{u}}$ ▷ Calculate unguided posterior $\tilde{\mu}_k$
- 5: **for** $i \leftarrow 1$ to N_g **do**
- 6: $\tilde{\mu}_k \leftarrow \tilde{\mu}_k + \lambda\sigma_k \nabla_{\tilde{\mu}_k} \mathcal{J}_y(\mathcal{D}\theta(\tilde{\mu}_k))$ ▷ Guidance gradient step
- 7: **end for**
- 8: $\tilde{\mathbf{u}}_{k-1} \sim \mathcal{N}(\tilde{\mu}_k, \sigma_k^2 \mathbf{I})$ ▷ Sample previous-step noised control sequence
- 9: **end for**
- 10: **Return:** Final control sequence $\mathbf{u} \leftarrow \tilde{\mathbf{u}}_0$

Setup. We showcase the controllable scenario generation capabilities of our model in fulfilling various simulation tasks. To test this, we leverage the same subset of 500 WOMD interactive validation scenarios employed in Section 5.2. Within each scene, we select the 32 agents closest to the labeled ego agent for evaluation, excluding all others. Our simulations span an 8-second horizon with a replanning frequency of 1 Hz, utilizing metrics from the Waymax simulator including off-road incidents, collisions, wrong-way occurrences, kinematic infeasibilities, and log divergences. Furthermore, we analyze all quantitative results across three distinct random seeds, averaging outcomes to assess the generality and consistency of VBD.

Evaluation Metrics. To evaluate the performance of our proposed model, we provide both qualitative and quantitative results. Additional qualitative results including more illustrative video examples can be found on the [Project Website](#). Furthermore, We employ with following metrics to thoroughly test our proposed method quantitatively:

- *Off-road:* A binary metric indicating whether a vehicle drives off the road, determined by its position relative to oriented road graph points. A vehicle is considered off-road if it strays from the right side of an oriented road edge. The off-road rate is calculated as the percentage of vehicles transitioning from on-road to

off-road across all simulated agents and generated scenarios.

- *Collision:* A binary metric identifying collisions between agents. Collisions are detected when the 2D bounding boxes of two objects overlap at the same time step. The collision rate is calculated as the percentage of agents involved in collisions across all simulated agents and generated scenarios.
- *Wrong-way:* A binary metric measuring whether a vehicle deviates from its intended driving direction. A wrong-way movement is flagged if the vehicle's heading angle deviates more than 90 degrees from its closest lane direction for over 1 second. The wrong-way rate is calculated as the percentage of vehicles entering the wrong way across all simulated vehicles and generated scenarios.
- *Kinematic infeasibility (Kin.):* A binary metric assessing whether a vehicle's transition steps are kinematically plausible. The limits for acceleration magnitude (6 m/s^2) and steering curvature magnitude (0.3 m^{-1}) are set empirically. The kinematic infeasibility rate is calculated as the percentage of vehicles with infeasible trajectories across all simulated vehicles and generated scenarios.
- *Log divergence (ADE and FDE):* These metrics quantify the deviation from ground truth behavior using displacement error. Average Displacement Error (ADE) is the L2 distance between an agent's simulated and ground-truth positions at each time step. Final Displacement Error (FDE) is the L2 distance between an agent's simulated and ground-truth positions at $t = 8$ seconds. ADE and FDE are reported as averages across all time steps and valid agents in all generated scenarios.
- *Minimum Log divergence (minADE and minFDE):* These metrics quantify the minimum deviation from ground truth behavior using displacement error among all generated scenarios for each scene. MinADE and minFDE are reported as averages across all time steps and valid agents for all scenes.

6.1 Composition with Behavior Priors

Sampling diverse outputs from a conditional diffusion model is challenging, especially when the denoising strongly relies on the context information (Sadat et al. 2023). On the other hand, the behavior predictor in VBD captures the multi-modal trajectories of individual agents but will result in scene inconsistency if marginal trajectories are naively combined. This is because the predictor alone cannot ensure the coherence necessary for realistic multi-agent scenario generation. The denoiser in VBD can used as an effective scenario optimizer and produce diverse and scene-consistent scenarios by first selecting goal positions for some target agents and generating joint multi-agent trajectories matching individual goals using guided sampling. Consider a scenario where we need to specify desired behaviors for A^t agents, we can heuristically determine the target behaviors or goals for each target agent, represented by $\{\mathbf{g}^i\}_{i=1:A^t}$. The guidance

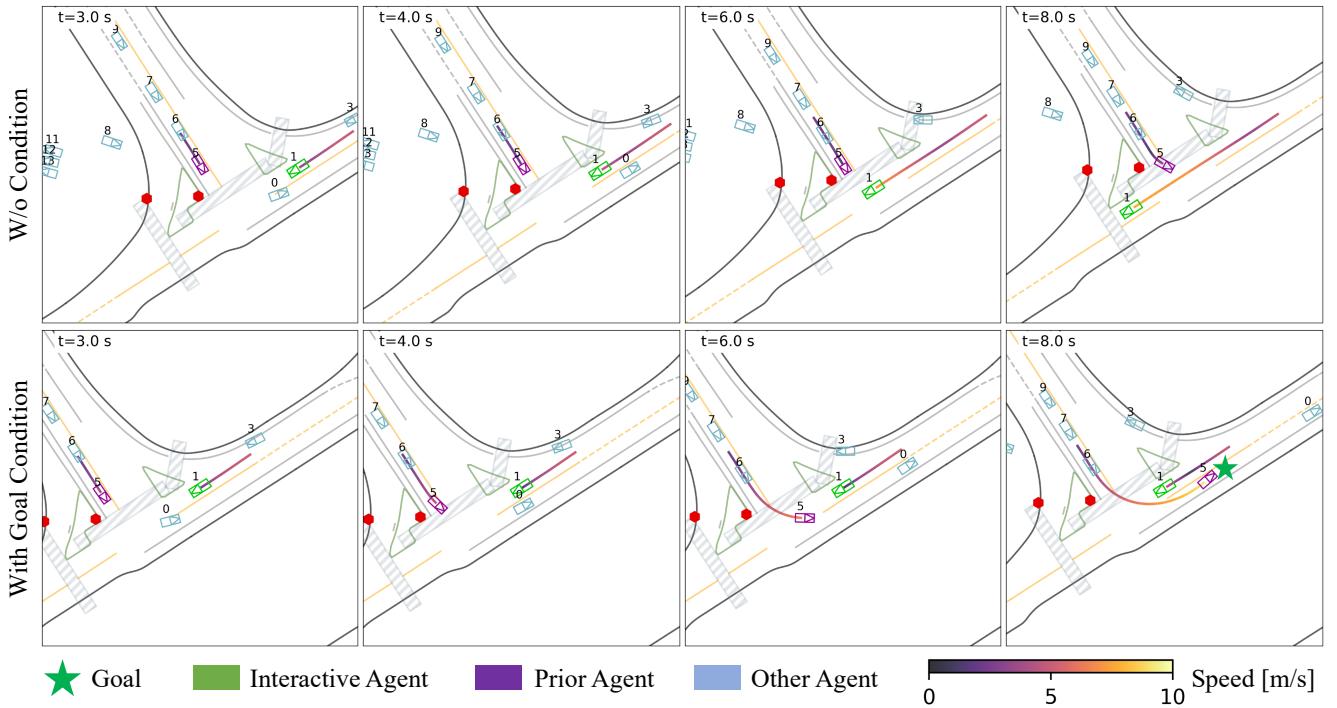


Figure 4. VBD produces scene-consistent traffic scenarios when target agents are conditioned on specific behavior priors. **Top:** Nominal VBD rollout without guidance generates a scene-consistent scenario, where Vehicle 5 (in purple) waits at the stop sign and then precedes. **Bottom:** Using goal-guided diffusion to minimize Vehicle 5’s final position w.r.t to a given goal, we enforce Vehicle 5 to run the stop sign. VBD model generates a scene-consistent scenario with Vehicle 1 yielding to Vehicle 5.

Table 3. Simulation Results of Scenario Editing with Marginal Predictor and Diffusion Policy over Three Different Seeds

Method	Collision [%] ↓	Off-road [%] ↓	Wrong-way [%] ↓	Kin. [%] ↓	ADE [m] ↓
Marginal Prediction	5.61±0.27	6.19±0.05	0.86±0.09	0.31±0.02	1.113±0.012
Diffusion	2.47±0.09	1.21±0.14	0.57±0.02	0.24±0.01	1.010±0.007
Post-Optimization via Diffusion	2.23±0.15	1.26±0.10	0.52±0.11	0.32±0.01	0.974±0.005

function based on behavior priors is:

$$\mathcal{J}_{goal} = - \sum_{i=1:A^t} \mathcal{SL}_1(\mathbf{g}^i - \mathbf{x}_T^i), \quad (22)$$

where \mathcal{SL}_1 denotes the Smooth L1 loss, x_T^i is the state of an agent derived from actions using a differentiable dynamic function, and T is the planning horizon. The other agents in the scene will not be directly influenced by the guidance.

Case Study 1: Scene-consistent Scenario Editing We evaluate the effectiveness of VBD as a scenario editing and optimization tool to generate scene-consistent interactions based on marginal behavior priors for target agents. Specifically, for each target agent, we select its most likely predicted trajectory as the goal. The baseline method (marginal prediction) directly executes the trajectory for target agents. For the post-optimization with diffusion, these trajectories serve as goal priors, and we employ guided sampling to minimize the mean L_2 distance between the diffusion results and the selected goals. We compare their performance in generating specific behaviors (goal reaching) and test each method with three different random seeds. The results in Table 3 indicate the diffusion policy enhances scene-consistency in the generated scenarios, given specific behaviors from marginal behavior priors. It significantly reduces the collision rate and better captures interactions

between agents in the scenes for goal-reaching behaviors of target agents. Moreover, even when priors are selected from suboptimal samples, e.g., off-road or wrong-way, VBD can alleviate these cases and generate scenarios that conform to the scene context.

Figure 4 showcases the VBD model’s capability in scene-consistent scenario editing where one agent is assigned a target point based on marginal priors. In this setup, a human user oversees the simulation, manually selecting the target agent and its goal. By conditioning on the behavior that vehicle 5 makes a left turn ignoring the stop sign, our VBD model’s diffusion policy can generate a scenario such that vehicle 1 is compelled to yield to the other vehicle. Conversely, the nominal diffusion policy rollout without guidance follows a completely different but still scene-consistent traffic ordering.

6.2 Composition with Cost Functions

Incorporating collision avoidance guidance into the diffusion generation process can enhance the quality and realism of the generated scenarios and create nuanced, collision-avoiding behaviors among agents. This is implemented through a differentiable cost function, expressed as follows:

$$\mathcal{J}_{overlap} = \sum_{t=1}^T \sum_{i,j}^A d_{ij}(\mathbf{x}_t) \mathbb{1}(d_{ij}(\mathbf{x}_t) < \epsilon_d), \quad (23)$$

Table 4. Simulation Results of Collision Avoidance Cost-guided Diffusion over Three Different Seeds

Method	Collision [%] ↓	Off-road [%] ↓	Wrong-way [%] ↓	Kin. [%] ↓	ADE [m] ↓
VBD	2.47±0.09	1.21±0.14	0.57±0.02	0.24±0.01	1.010±0.007
VBD + Collision Avoid	1.11±0.23	1.15±0.17	0.60±0.12	0.25±0.01	1.113±0.010

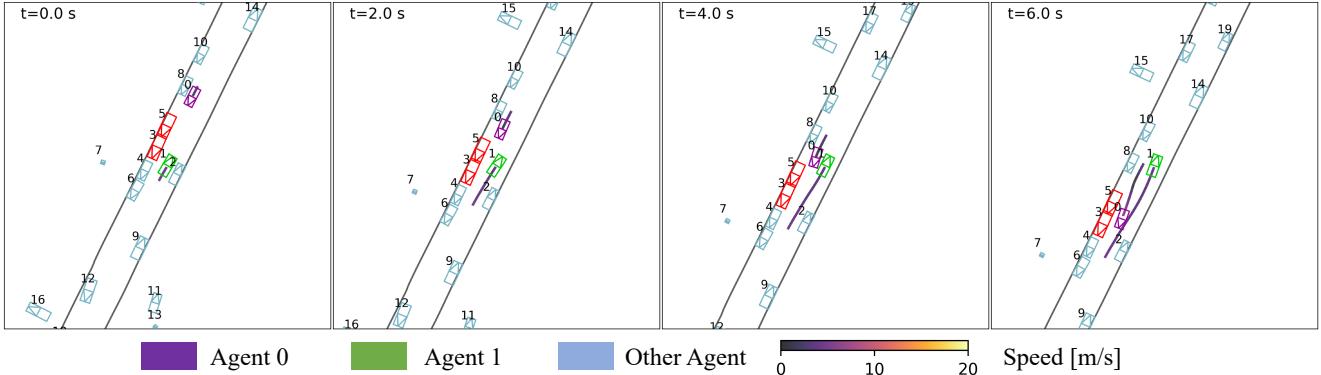


Figure 5. Composition of collision avoidance cost-based objective in diffusion can improve generation quality. Two vehicles dynamically interact and coordinate in a narrow passage scenario with collision cost function guidance. (Note: Vehicles 3 and 5 have been in collision since the initial step.)

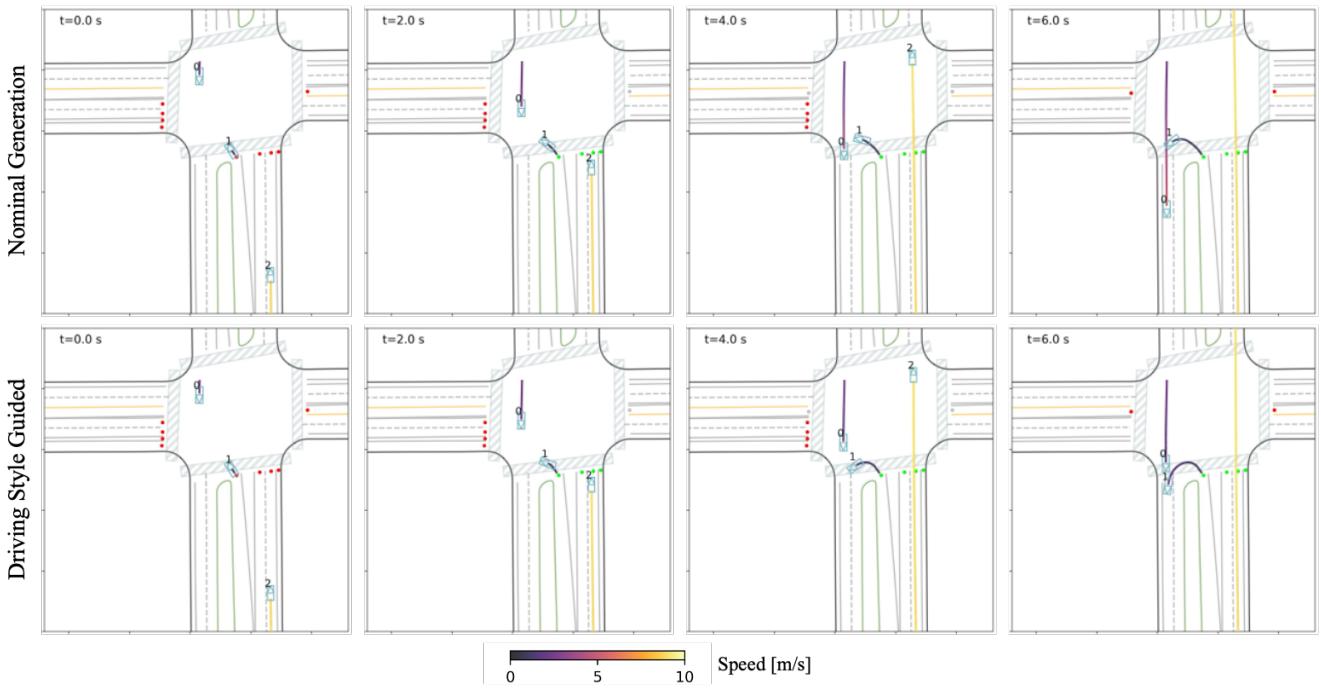


Figure 6. Composition of rush cost-based objective in diffusion can effectively change the driving style of an agent. **Top:** Nominal VBD rollout without guidance, where Vehicle 1 yields to Vehicle 2 before the U-turn maneuver. **Bottom:** Using rush cost-guided diffusion on Vehicle 1 makes it more aggressive, forcing Vehicle 2 to slow down and yield.

where d_{ij} represents the Minkowski distance between the footprints of agents i and j at time t . The parameter ϵ_d is the threshold for defining potential collision.

Further, to modify the driving behavior of a target agent (i.e., to make it more aggressive), we introduce a rush cost function that penalizes speed reduction, which is:

$$\mathcal{J}_{rush} = - \sum_{t=1}^T \|a_t^i\|^2 \mathbb{1}(a_t^i < 0), \quad (24)$$

where a_t^i represents acceleration action of agent i at time t . By adding this function and the collision avoidance function

together, we can implicitly modify the driving behavior of a specific agent within traffic flows.

Case Study 2: Enhancing Generation Quality Since neural networks still exhibit unpredictability in generation, they may be prone to errors in some scenarios (e.g., off-roads or collision behaviors). We aim to use a collision avoidance cost-based optimization objective in guided sampling to further improve the performance and generation quality of VBD. Specifically, we introduce a collision avoidance cost to maximize each agent's minimum distance from others across the horizon. As shown in Table 4, this collision avoidance objective effectively reduces the collision rate.

The qualitative results in [Figure 5](#) further illustrate that incorporating a cost-based objective enables the model to generate collision-free interactions in the challenging narrow-passage scenario.

Case Study 3: Modifying Driving Style In simulation settings, it is often desirable to alter the driving behaviors of certain agents to generate different driving styles. To achieve this, we combine a collision avoidance cost and a rush cost to change the driving behavior of a target agent in the scene generation process using guided diffusion. This composite cost guidance can effectively change the behaviors of specific agents to show aggressive driving styles, while the VBD model ensures that surrounding agents remain responsive to these behavioral changes. We manually select a target agent in a scenario and consistently apply the guidance to change the driving style of the agent in closed-loop simulation. A qualitative example of modifying an agent's driving style is illustrated in [Figure 6](#), where we modify the driving style of Agent 1 to exhibit an aggressive U-turn maneuver. Concurrently, the VBD model generates scene-consistent behaviors for Agent 2, who adapts by yielding to the aggressive agent.

7. Generation Beyond Data Distribution

A crucial use case of simulations is to create safety-critical scenarios to stress test the robustness of AV planning systems, which must be both realistic and reactive. We employ two guidance approaches (conflict prior guided diffusion and game-theoretic guided diffusion) in our VBD model to achieve this. Examples of the generated safety-critical and adversarial scenarios using our VBD model, implemented with these two composition methods, are showcased on the [Project Website](#). These examples demonstrate the efficacy of our methods in generating realistic scenarios that introduce collision risks for the AV, without requiring risk-specific driving datasets or any retraining or fine-tuning of the model.

7.1 Safety-Critical Scenario Generation via Conflict Priors

We can utilize conflict-prior guidance to facilitate the generation of safety-critical scenarios. The behavior prior prediction significantly aids in identifying potentially unsafe agents, which is difficult for existing methods, and enhances the realism of the unsafe behaviors. By using the behavior predictor in our VBD model, we can obtain prior distributions of the possible movements of surrounding agents. From this distribution, agents that could conflict with the ego vehicle's plans are identified, and these selected priors are then used to guide diffusion policy to generate expected scenarios. The process of selecting conflicting agents is represented as follows:

$$\hat{\mathbf{x}}^{adv} = \arg \max_{i,j} \hat{\omega}_j^i [col(\hat{\mathbf{x}}_j^i, \hat{\mathbf{x}}_*^{ego}) \mathbf{1}(\hat{\omega}_j^i > \epsilon_p)], \forall i \neq ego, \quad (25)$$

where $\hat{\mathbf{x}}_j^i, \hat{\omega}_j^i$ denote the trajectory and probability estimation for the prior mode j of agent i , and $\hat{\mathbf{x}}_*^{ego}$ is the most-likely prior mode of the ego vehicle. col is a collision probability function, which decreases linearly with the

temporal proximity to a collision, and ϵ_p is a threshold controlling the scenario's possibility.

The core idea is to select the highest posterior probability of other agents' behaviors that conflict with the ego agent's normal driving behavior. The selected conflicting behaviors are then fed into the diffusion policy as guidance inputs. Meanwhile, the behaviors of other agents are controlled by the policy without guidance to respond to that situation. Notably, utilizing prior guidance in the diffusion policy results in more realistic safety-critical behavior compared to direct trajectory rollout, as well as robustness against selected priors that are less likely in real-world conditions. The proposed conflict prior guidance for safety-critical simulation is illustrated in [Figure 7](#), and some generated safety-critical scenarios are presented in [Figure 8](#).

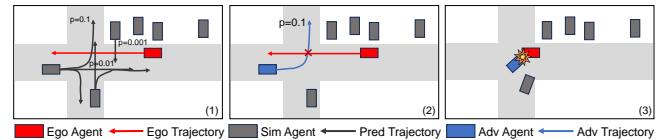


Figure 7. Illustration of conflict prior-guided safety-critical scenario generation. (1) The behavior predictor generates multimodal trajectories for each sim agent; (2) From the prior distribution, select the agent behavior with the highest probability of conflicting with the most likely ego plan; (3) Generate the expected scenario using the diffusion policy.

Algorithm 4 Game-theoretic-guided adversarial scenario generation with gradient descent-ascent

Require: Denoiser \mathcal{D}_θ , diffusion steps K , evader index e , pursuer index p , guidance steps N_g , descent steps N_e , ascent steps N_p , evader gradient mask M_e , pursuer gradient mask M_p , scaling parameter λ , standard deviation σ_k .

1: $\tilde{\mathbf{u}}_K \sim \mathcal{N}(0, \mathbf{I})$ ▷ Sample initial trajectory
 2: **for** $k \leftarrow K$ to 1 **do**
 3: $\hat{\mathbf{u}} \leftarrow \mathcal{D}_\theta(\tilde{\mathbf{u}}_k, k, \mathbf{c})$ ▷ Predict denoised control sequence
 4: $\tilde{\mu}_k \leftarrow \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k} \tilde{\mathbf{u}}_k + \frac{\sqrt{\bar{\alpha}_k-\bar{\alpha}_k}\beta_k}{1-\bar{\alpha}_k} \hat{\mathbf{u}}$ ▷ Calculate unguided posterior $\tilde{\mu}_k$
 5: **for** $i \leftarrow 1$ to N_g **do**
 6: **for** $i_d \leftarrow 1$ to N_e **do**
 7: $\hat{\mathbf{x}} \leftarrow f(\mathcal{D}_\theta(\tilde{\mu}_k))$
 8: $\mathcal{J} \leftarrow \min_t d_{e,p}(\hat{\mathbf{x}}_t) + \min_t d_r(\hat{\mathbf{x}}_t^e)$
 9: $\tilde{\mu}_k^e \leftarrow \tilde{\mu}_k - \lambda \sigma_k M_e \nabla_{\tilde{\mu}_k^e} \mathcal{J}$ ▷ Gradient descent for evader
 10: **end for**
 11: **for** $i_d \leftarrow 1$ to N_p **do**
 12: $\hat{\mathbf{x}} \leftarrow f(\mathcal{D}_\theta(\tilde{\mu}_k))$
 13: $\mathcal{J} \leftarrow \min_t d_{e,p}(\hat{\mathbf{x}}_t) + \min_t d_r(\hat{\mathbf{x}}_t^p)$
 14: $\tilde{\mu}_k^p \leftarrow \tilde{\mu}_k + \lambda \sigma_k M_p \nabla_{\tilde{\mu}_k^p} \mathcal{J}$ ▷ Gradient ascent for pursuer
 15: **end for**
 16: **end for**
 17: $\tilde{\mathbf{u}}_{k-1} \sim \mathcal{N}(\tilde{\mu}_k, \sigma_k^2 \mathbf{I})$ ▷ Sample previous-step noised control sequence
 18: **end for**
 19: **Return:** Final control sequence $\mathbf{u} \leftarrow \tilde{\mathbf{u}}_0$

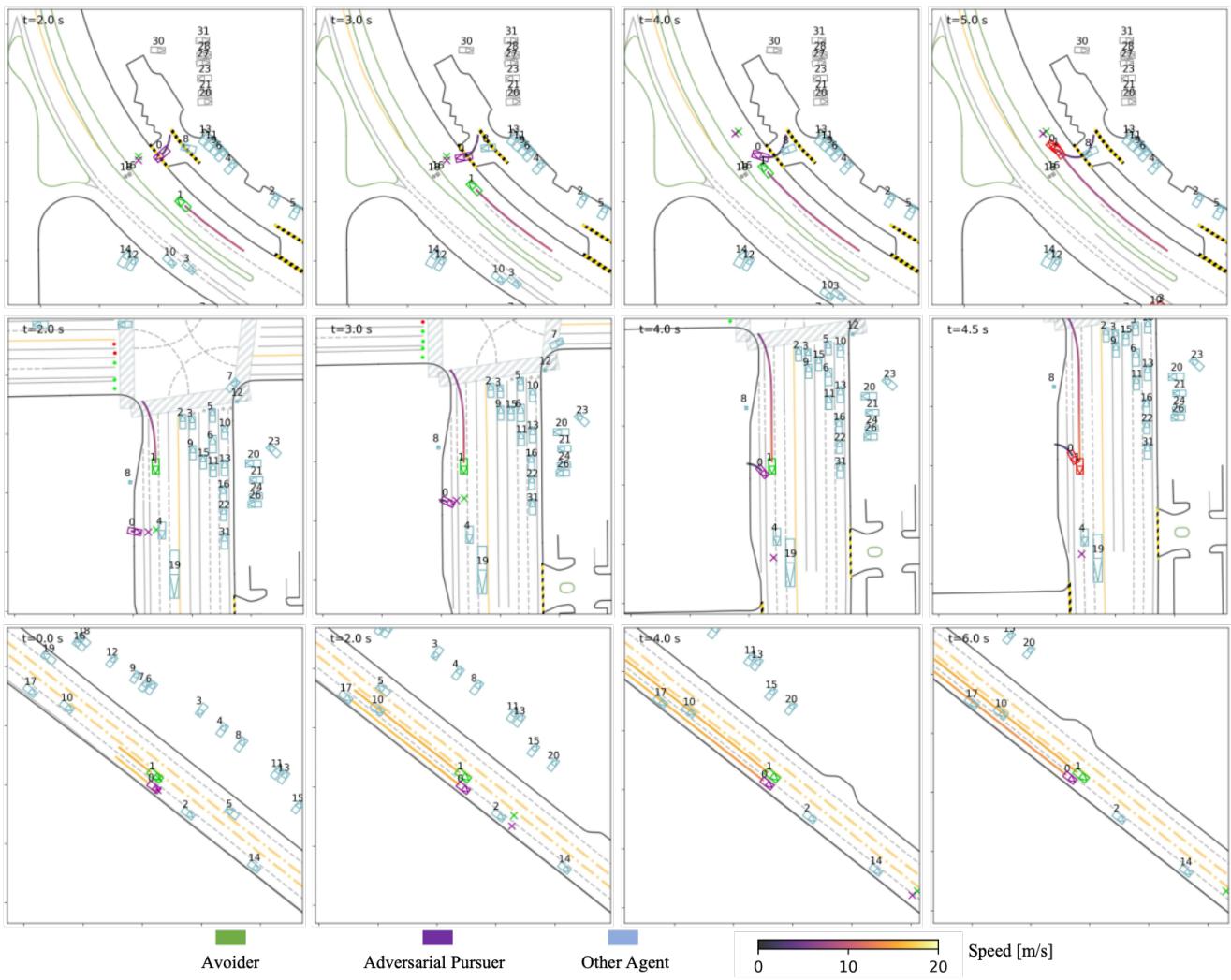


Figure 8. Safety-critical scenario generation through conflict priors guidance. **Top and Middle:** By selecting the highest posterior probability of adversarial agents' trajectories that conflict with the ego agent's most likely prediction, we are able to guide the denoising process to generate safety-critical scenarios. **Bottom:** Failure case of conflict priors guidance when there were no collision pairs from behavior priors.

7.2 Safety-Critical Scenario Generation via Game-theoretic Guidance

Conflict prior method is an effective sampling approach when agents' paths intersect with each other. However, this method may fail to capture the potential safety-critical scenario when no collision pairs can be automatically selected from behavior priors (e.g., the bottom row of Figure 8). To simulate interactions in more extreme conditions, such as adversarial driving behaviors, we model the interaction between two agents as a map-constrained pursuit-evasion game, where the pursuer aims to cause a collision with the evader and stay within the road boundary, while the evader attempts to avoid it. An effective approach to solving this game is through iterative best response (IBR), such as gradient descent-ascent (GDA). Specifically, we can apply τ -GDA with time-scale separation to guarantee local convergence to a stable *minimax equilibrium* of the pursuit-evasion game (Fiez et al. 2021). We update the pursuer more frequently than the evader to provide information advantage to the adversarial agents.

Generating a scenario with a game-theoretic solver alone often leads to unrealistic results. Instead, leveraging realistic

traffic behaviors modeled by VBD, we propose a *game-theoretic-guided diffusion* approach (Algorithm 4). This method involves alternating denoising steps, performing gradient descent and ascent for agents, and updating the noised \tilde{u} until convergence. In addition, we can further enhance the realism of the scenario with *optional* gradient masks M_e and M_p , which allow us to adjust the adversity of the pursuer and the responsiveness of the evader by selectively updating gradients at certain timesteps.

We illustrate generating adversarial driving scenarios with the proposed game-theoretic guidance in various scenes in Figure 9, where the proposed game-theoretic guidance can generate highly interactive maneuvers on both traffic merge and lane change scenarios. In contrast to previous works (Rempe et al. 2022; Chang et al. 2023a; Zhang et al. 2023a), which primarily sample adversarial behaviors based on fixed trajectories of the pursued vehicle, our approach proactively optimizes the actions of both the pursuer and evader. This results in highly interactive scenarios, enriching the realism of adversarial encounters. Our proposed method facilitates the generation of highly realistic scenarios, especially with regard to adversarial behavior, by ensuring that they are

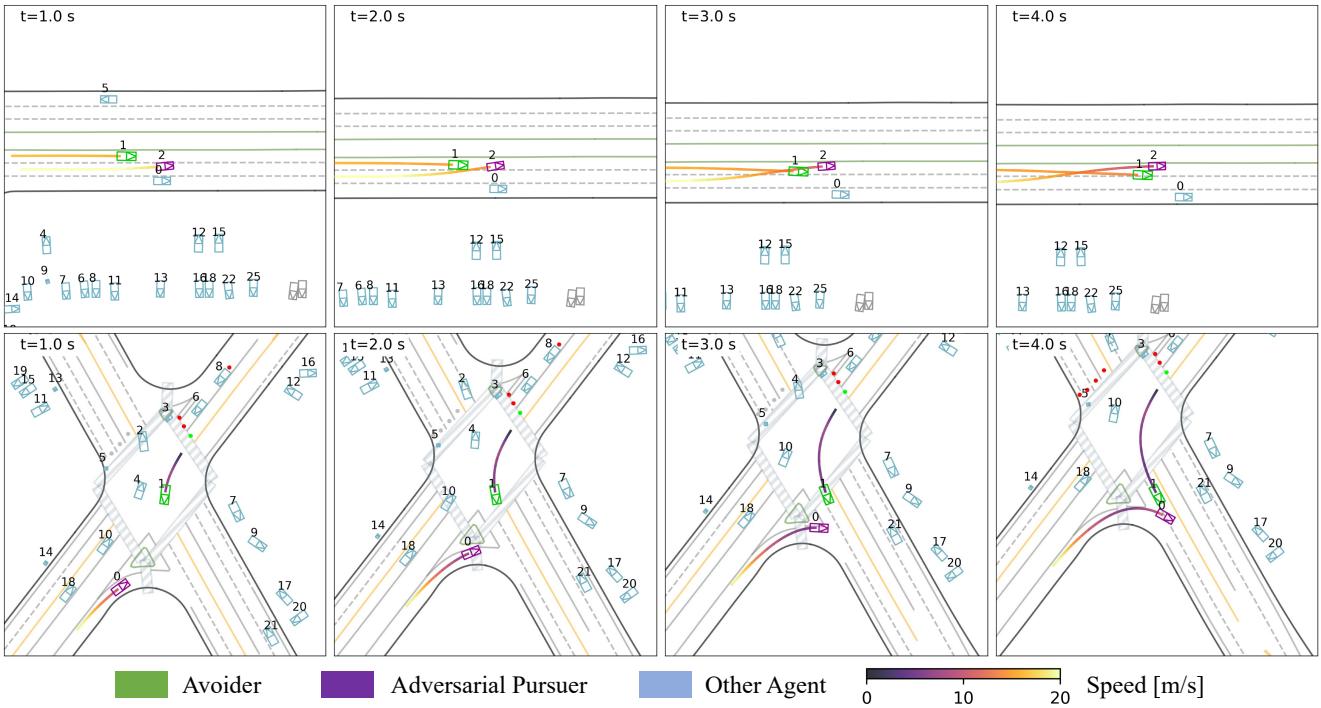


Figure 9. Results of game-theoretic adversarial behavior generation. **Top:** The adversarial pursuer merges in front of the evader, performs a brake check, and attempts to cause a rear-end crash. The evader immediately switches its lane and avoids the collision. **Bottom:** The adversarial pursuer merges aggressively to the adjacent lane, and the evader yields to the pursuer by slowing down.

proportionately challenging and reactive to the maneuvers of the ego vehicle. This strategy overcomes the shortcomings of previous methods, which tend to generate unrealistically aggressive adversarial strategies.

8. Effective Training and Inference Strategies

This section outlines the key strategies employed during the training and testing phases of our VBD model to enhance its effectiveness in behavior modeling and generation. We conducted closed-loop simulation experiments to validate the importance of these strategies. These experiments are evaluated with the same 500 scenarios and metrics used in Section 6. To further stress test our algorithm, we sample 32 independent closed-loop roll-outs for every scene. Each scene contains up to 32 agents operating with a replanning frequency of 1 Hz. Performance metrics were computed as the average across all rollouts.

8.1 Importance of the Training Objective for Behavior Generation

Traditional image generation tasks using diffusion models typically involve regressing either the injected noise or the ground truth image from a noised input. Notably, research has shown that predicting noise yields superior generation quality (Ho et al. 2020). However, our VBD model is trained with the loss function defined in Equation 15, which minimizes the L_1 distance between the trajectory rollouts derived from denoised action sequences and the corresponding ground-truth trajectories.

Our experimental results demonstrate that supervision through trajectory rollouts is crucial for generating high-quality traffic scenarios. Unlike image generation, where

errors in pixel prediction do not compound, scenario generation involves sequential decision-making, where errors accumulate across time steps. Supervising either the control sequence \mathbf{u} or the noise ϵ in an open-loop manner fails to capture the temporal dynamics of traffic behaviors.

To illustrate this, we trained a diffusion model with the same architecture as VBD for 20 epochs, supervising directly on the generated control sequences \mathbf{u} and the noise ϵ . As shown in Figure 10, models trained under these settings failed to produce coherent and realistic traffic scenarios, highlighting the necessity of trajectory rollouts as the training objective.

8.2 Effective Noise Scheduling Design

Prior studies have emphasized the significance of selecting an optimal noise schedule for enhancing generation quality (Nichol and Dhariwal 2021; Chen 2023). While the cosine noise schedule has gained widespread adoption in various domains, including image generation, offline reinforcement learning (Janner et al. 2022), robot trajectory planning (Chi et al. 2023), and other traffic scenario generation works (Zhong et al. 2023b,a; Jiang et al. 2024), our findings suggest that customizing the noise schedule to suit the problem formulation substantially affects generation quality.

During training, we sample a noise variance level $\bar{\alpha}_k$ and introduce random noise into the ground truth control sequence \mathbf{u} . Then, we supervise the trajectory rollout from the predicted control sequence to align with the ground-truth one. Considering a simplified example of a vehicle approaching an intersection with three possible paths (straight, left turn, and right turn), we visualize the trajectory rollout $\tilde{\mathbf{x}}_k$ from the noised action sequence $\tilde{\mathbf{u}}_k$

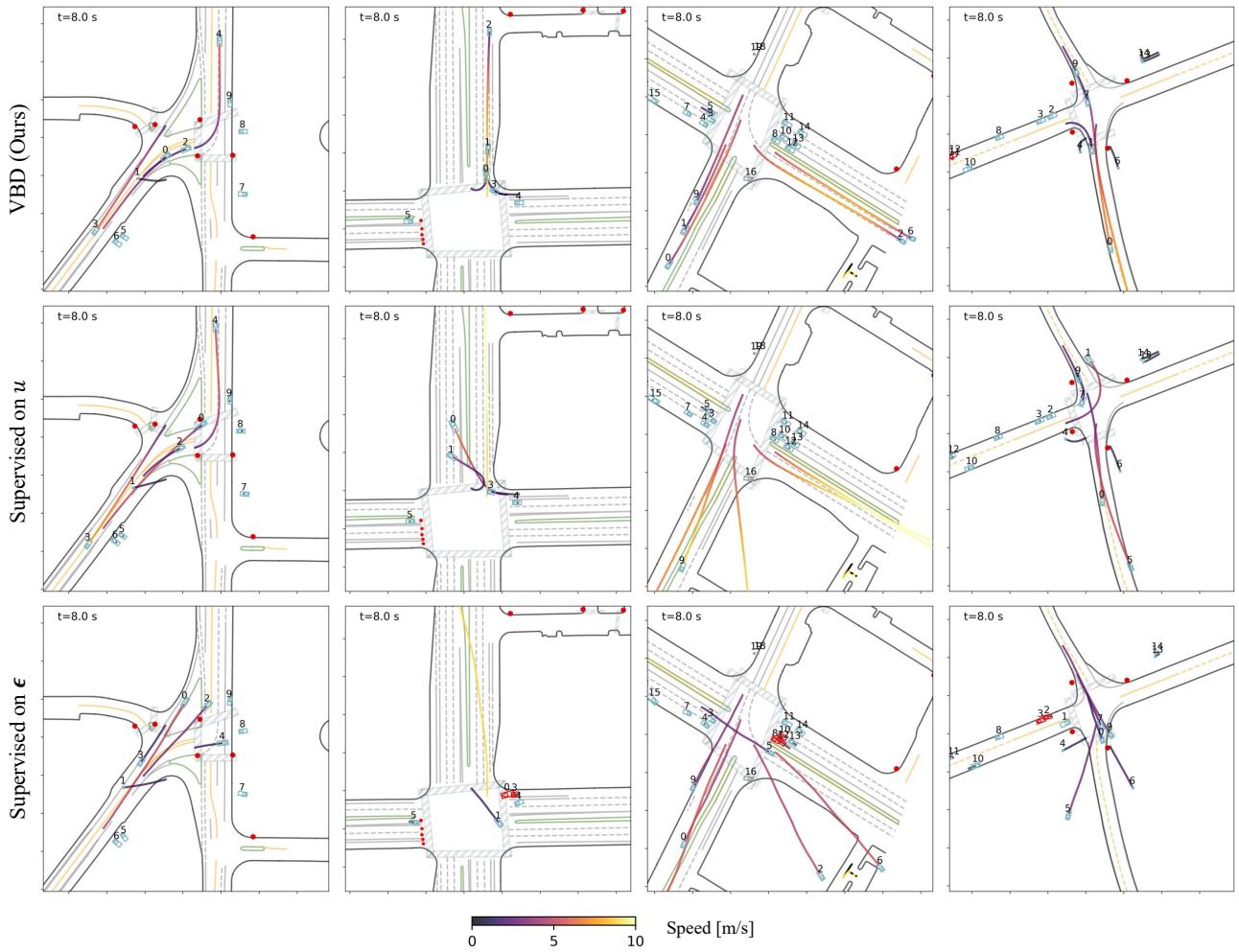


Figure 10. Comparison of training objectives in VBD by rolling out entire 8-second open-loop trajectories. The impact of training objectives is illustrated by examining the generated agent trajectories. **Bottom:** The model trained to predict noise ϵ fails to produce meaningful agent behaviors. **Middle:** The model trained to imitate control sequences u fails to align with map features. **Top:** Supervised on roll-out trajectories, VBD significantly improves planning quality with the same architecture and training steps.

(denoiser input) at varying noise variance levels on the right-hand side of Figure 11. At lower values of k , \tilde{x}_k closely approximates the original ground truth. Moreover, even at moderate noise levels ($\bar{\alpha} = 0.6$), the modes of ground truth actions remain distinguishable. However, traditional cosine scheduling tends to favor low noise variance regions during the initial diffusion steps, potentially allowing the model to bypass proper scene representation learning by relying on shortcuts through less noisy inputs. Consequently, when encountering out-of-distribution actions during testing, the model struggles to adapt and instead produces degraded output by directly passing the actions through the final denoising steps. This insight motivated us to develop a log noise schedule, presented in Equation 16, which maintains a suitable signal-to-noise ratio (SNR) throughout the diffusion process, ensuring effective action denoising dependent on scene context. Additionally, experimental results in Figure 12 validate the efficacy of our proposed approach, showcasing improved trajectory diffusion performance under the new noise schedule. By adopting the log scheduling strategy, VBD successfully generates diverse scenarios without compromising generation quality.

Interestingly, the result in Figure 12 reveals that the generation quality is satisfactory after just the first denoising

step, especially regarding collision rate, off-road rate, ADE, and FDE. Further denoising steps seem to degrade these metrics. One possible explanation is that traffic behavior diffusion is heavily influenced by scene context; hence, the initial denoising step can yield plausible outcomes adhering to both agent interactions and map constraints. Nevertheless, subsequent denoising steps play a vital role in striking a balance between fidelity and diversity. With increasing numbers of steps, we notice a decline in minADE and minFDE metrics, suggesting that one of the generated samples draws closer to the ground truth. Simultaneously, average ADE and FDE metrics rise, highlighting increased sample diversity. This outcome demonstrates that additional denoising steps empower the model to explore a wider range of feasible outcomes while preserving high-quality trajectories. Furthermore, these extra steps are crucial for facilitating control and guidance over agent behaviors, particularly in applications such as traffic scenario generation as we have previously elaborated in Section 5 and 6. The diffusion process helps the model to produce realistic outputs while adjusting behaviors according to specific scene-level requirements.

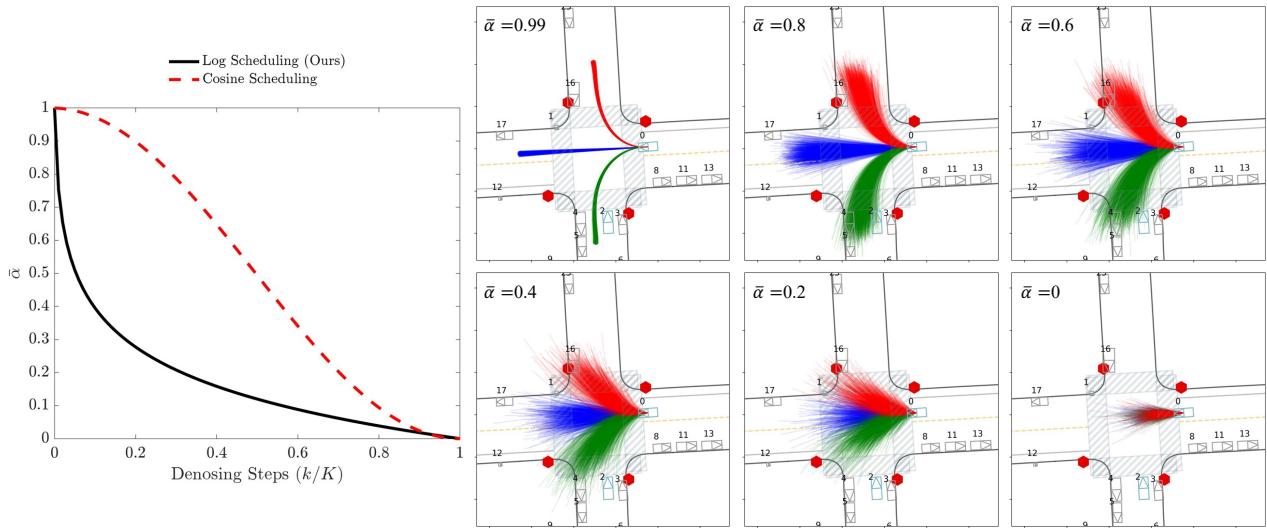


Figure 11. Illustration of the influence of noise levels. **Right:** Sample trajectory roll-outs from noisy action sequences at different noise levels for three different modes. At low noise level (high $\bar{\alpha}$), rollout trajectories resemble the ground truth trajectories and show clear distinctions between modes. **Left:** Our proposed log scheduling biases more training steps towards high noise levels. Empirically, extensive training of denoiser at low noise levels leads to short-cut learning in VBD.

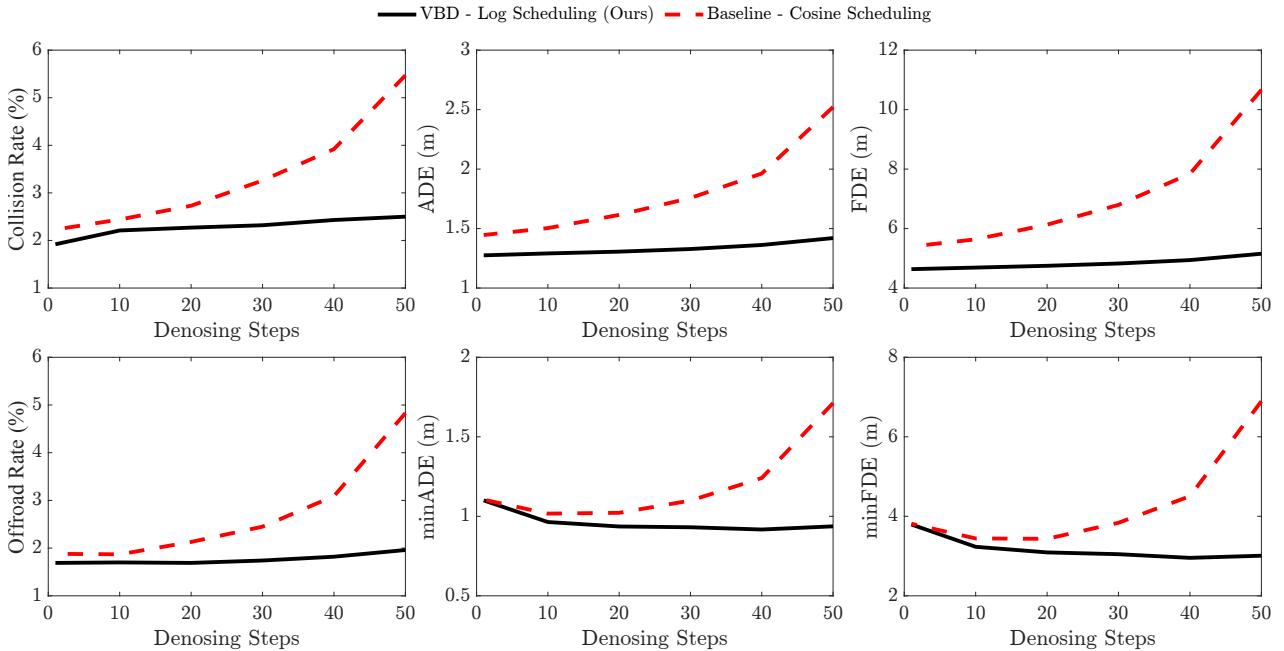


Figure 12. Effects of noise scheduling on closed-loop simulation performance. The metrics except for minADE and minFDE are average values over 32 samples of closed-loop rollout. The VBD model trained with cosine scheduling exhibits significant performance degradation across all metrics. In contrast, VBD trained with the proposed log scheduling maintains low collision and off-road rates across all denoising steps. Additionally, minADE and minFDE metrics decrease as the number of denoising steps increases, reflecting the model's ability to generate scenarios closer to the ground truth. A slight increase in ADE and FDE metrics indicates improved sample diversity with more denoising steps.

8.3 Influence of History Encoding

Causal confusion presents a considerable obstacle when applying imitation learning-based models to closed-loop simulations. Consistent with prior studies (De Haan et al. 2019; Cheng et al. 2024; Bansal et al. 2018), our findings suggest that discarding historical information from agent representations and focusing solely on their current states substantially alleviates this problem. Specifically, we employ an aggressive dropout rate (50%) for the past agent trajectories at training time and only supply the current joint state to the model during inference. As shown in

Table 5, experimental results reveal that models trained and tested with history dropout surpass those using full historical states encoding across all metrics, underscoring its effectiveness in mitigating causal confusion. Specifically, we observe the baseline with full past trajectories overly accelerates or decelerates in closed-loop simulation and leads to significant high displacement errors, highlighting its copycatting behavior through kinematic history. By adopting history dropout, we achieve enhanced performance in closed-loop simulations, highlighting the significance of suitable state representation in such contexts.

Table 5. Impact of History Encoding and Integrating Behavior Prior Prediction in VBD

Predictor	No History	Collision [%] ↓	Off-road [%] ↓	ADE [m] ↓	FDE [m] ↓	minADE [m] ↓	minFDE [m] ↓
✓	✗	7.44	4.44	4.23	17.94	3.55	14.57
✗	✓	3.33	2.42	1.46	5.22	0.99	3.14
✓	✓	2.50	1.96	1.42	5.15	0.94	3.01

Table 6. Closed-loop Simulation Results with DDIM Sampling

Step	Unguided Scenario Generation				Guided Scenario Generation			
	Collision [%] ↓	Off-road [%] ↓	ADE [m] ↓	Runtime [s] ↓	Collision [%] ↓	Off-road [%] ↓	ADE [m] ↓	Runtime [s] ↓
5	2.26	1.62	1.23	0.16 ± 0.023	1.01	1.14	1.32	0.55 ± 0.021
10	2.39	1.72	1.25	0.31 ± 0.001	1.06	1.35	1.35	1.19 ± 0.034
25	2.41	1.77	1.29	0.73 ± 0.002	1.01	1.34	1.54	2.87 ± 0.078
DDPM	2.50	1.96	1.42	1.48 ± 0.005	1.11	1.58	1.68	6.33 ± 0.130

8.4 Integration of Behavior Prediction

Another key design choice in VBD is the integration of a multi-modal behavior predictor within the model architecture as a co-training task. The results shown in Table 5 indicate the inclusion of the behavior predictor significantly enhances performance in closed-loop simulation tests compared to a baseline approach lacking prediction capabilities. The integration of the predictor contributes to the refinement of the diffusion policy by facilitating more effective learning within the model’s encoder. This interaction strengthens the encoder’s capacity to extract meaningful features, which are critical for downstream tasks. Moreover, the predictor provides behavior priors for individual agents, thereby enabling flexible behavior editing and enhancing the controllability of the diffusion policy.

8.5 Accelerating Inference

Employing Denoising Diffusion Implicit Models (DDIM) (Song et al. 2020a) can significantly accelerate the inference process of the VBD model while maintaining performance. To evaluate the impact of diffusion steps on simulation and runtime performance, we conduct experiments with varying numbers of diffusion steps in DDIM, and the detailed results are summarized in Table 6. The runtime of each inference step is measured on a system equipped with an AMD 7950X CPU and an NVIDIA RTX 4090 GPU. The results reveal that the runtime grows linearly with the number of diffusion steps. Notably, employing 5 or 10 diffusion steps achieves a favorable balance between generation quality and real-time performance (with a feasible replan frequency of 1 Hz). Especially when combined with collision cost guidance, we can significantly reduce collision rate and enhance the quality of generated behaviors. Specifically, using 5 diffusion steps in DDIM guided sampling yields a notable reduction in collision rates and improves scenario realism without compromising computational efficiency. Moreover, increasing the number of diffusion steps results in diminishing returns for collision avoidance and a negative impact on the log divergence (ADE) metric. This suggests the performance and efficiency of VBD can be further enhanced with DDIM sampling to balance computational efficiency and performance and retain a certain level of controllability.

9 Conclusions

9.1 Summary

In this work, we present **Versatile Behavior Diffusion** (VBD), a novel traffic scenario generation framework that integrates a diffusion generative model to produce realistic and controllable multi-agent interactions in closed-loop settings. Our approach has demonstrated state-of-the-art performance on the Waymo Sim Agents Benchmark, showcasing its capability to generate diverse, coherent, and interactive traffic scenarios that accommodate complex agent interactions and varied environmental conditions. The flexibility of VBD is highlighted through its support for controllable inference-time scenario editing and behavior guidance, allowing users to generate scenarios according to specific requirements. Our extensive evaluation and analysis reveal the effectiveness of VBD in handling a wide range of traffic simulation applications, including those involving safety-critical scenarios. Moreover, we provide valuable insights into successful training and inference strategies for diffusion-based traffic scenario generation models, shedding light on best practices and common pitfalls.

9.2 Future Work

While VBD has achieved impressive results, there are several avenues for future research and development. One important direction involves optimizing the model’s runtime efficiency to facilitate seamless integration with autonomous vehicle planning and training pipelines. Another exciting area of exploration is the automation of scenario generation using large language models (LLMs) that can learn to create context-dependent guidance objectives and prior distributions. Furthermore, expanding the scope of VBD to incorporate diverse traffic datasets, such as those from highway environments, will be crucial for improving generation quality and scalability. By pursuing these research directions, we envision VBD becoming an important tool for advancing traffic simulation and autonomous driving system development.

References

- Anderson BD (1982) Reverse-time diffusion equation models. *Stochastic Processes and their Applications* 12(3): 313–326.

- Bansal M, Krizhevsky A and Ogale A (2018) Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*.
- Caesar H, Bankiti V, Lang AH, Vora S, Lioig VE, Xu Q, Krishnan A, Pan Y, Baldan G and Beijbom O (2020) nuscenes: A multimodal dataset for autonomous driving. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 11621–11631.
- Cao Y, Xiao C, Anandkumar A, Xu D and Pavone M (2022) Advdo: Realistic adversarial attacks for trajectory prediction. In: *European Conference on Computer Vision*. Springer, pp. 36–52.
- Chang WJ, Pittaluga F, Tomizuka M, Zhan W and Chandraker M (2023a) Controllable safety-critical closed-loop traffic simulation via guided diffusion. *arXiv preprint arXiv:2401.00391*.
- Chang WJ, Tang C, Li C, Hu Y, Tomizuka M and Zhan W (2023b) Editing driver character: Socially-controllable behavior generation for interactive traffic simulation. *arXiv preprint arXiv:2303.13830*.
- Chen T (2023) On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*.
- Chen Y, Ivanovic B and Pavone M (2022) Scept: Scene-consistent, policy-based trajectory predictions for planning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 17103–17112.
- Cheng J, Chen Y, Mei X, Yang B, Li B and Liu M (2024) Rethinking imitation-based planners for autonomous driving. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 14123–14130.
- Chi C, Feng S, Du Y, Xu Z, Cousineau E, Burchfiel B and Song S (2023) Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*.
- De Haan P, Jayaraman D and Levine S (2019) Causal confusion in imitation learning. *Advances in neural information processing systems* 32.
- Dhariwal P and Nichol A (2021) Diffusion models beat gans on image synthesis. *Advances in neural information processing systems* 34: 8780–8794.
- Diehl C, Klosek T, Krueger M, Murzyn N, Osterburg T and Bertram T (2023) Energy-based potential games for joint motion forecasting and control. In: *7th Annual Conference on Robot Learning*.
- Du Y and Mordatch I (2019) Implicit generation and modeling with energy based models. *Advances in Neural Information Processing Systems* 32.
- Esser P, Chiu J, Atighehchian P, Granskog J and Germanidis A (2023) Structure and content-guided video synthesis with diffusion models. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 7346–7356.
- Ettinger S, Cheng S, Caine B, Liu C, Zhao H, Pradhan S, Chai Y, Sapp B, Qi CR, Zhou Y, Yang Z, Chouard A, Sun P, Ngiam J, Vasudevan V, McCauley A, Shlens J and Anguelov D (2021) Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 9710–9719.
- Feng L, Li Q, Peng Z, Tan S and Zhou B (2023) Trafficgen: Learning to generate diverse and realistic traffic scenarios. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3567–3575.
- Fiez T, Ratliff L, Mazumdar E, Faulkner E and Narang A (2021) Global convergence to local minmax equilibrium in classes of nonconvex zero-sum games. *Advances in Neural Information Processing Systems* 34: 29049–29063.
- Finn C, Christiano P, Abbeel P and Levine S (2016) A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*.
- Gulino C, Fu J, Luo W, Tucker G, Bronstein E, Lu Y, Harb J, Pan X, Wang Y, Chen X et al. (2023) Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *arXiv preprint arXiv:2310.08710*.
- Guo Z, Gao X, Zhou J, Cai X and Shi B (2023) Scenedm: Scene-level multi-agent trajectory generation with consistent diffusion models. *arXiv preprint arXiv:2311.15736*.
- Hanselmann N, Renz K, Chitta K, Bhattacharyya A and Geiger A (2022) King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In: *European Conference on Computer Vision*. Springer, pp. 335–352.
- Ho J and Ermon S (2016) Generative adversarial imitation learning. *Advances in neural information processing systems* 29.
- Ho J, Jain A and Abbeel P (2020) Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33: 6840–6851.
- Hu Y, Chai S, Yang Z, Qian J, Li K, Shao W, Zhang H, Xu W and Liu Q (2024) Solving motion planning tasks with a scalable generative model. *arXiv preprint arXiv:2407.02797*.
- Huang Z, Karkus P, Ivanovic B, Chen Y, Pavone M and Lv C (2023a) Dtpp: Differentiable joint conditional prediction and cost evaluation for tree policy planning in autonomous driving. *arXiv preprint arXiv:2310.05885*.
- Huang Z, Liu H and Lv C (2023b) Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 3903–3913.
- Huang Z, Liu H, Wu J and Lv C (2023c) Differentiable integrated motion prediction and planning with learnable cost function for autonomous driving. *IEEE transactions on neural networks and learning systems*.
- Huang Z, Mo X and Lv C (2022) Multi-modal motion prediction with transformer-based neural network for autonomous driving. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2605–2611.
- Huang Z, Wu J and Lv C (2021) Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning. *IEEE transactions on intelligent transportation systems* 23(8): 10239–10251.
- Hyvärinen A and Dayan P (2005) Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research* 6(4).
- Igl M, Kim D, Kuefler A, Mougim P, Shah P, Shiarlis K, Anguelov D, Palatucci M, White B and Whiteson S (2022) Symphony: Learning realistic and diverse agents for autonomous driving simulation. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2445–2451.
- Janner M, Du Y, Tenenbaum J and Levine S (2022) Planning with diffusion for flexible behavior synthesis. In: *International Conference on Machine Learning*. PMLR, pp. 9902–9915.

- Jiang C, Cornman A, Park C, Sapp B, Zhou Y, Anguelov D et al. (2023) Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 9644–9653.
- Jiang CM, Bai Y, Cornman A, Davis C, Huang X, Jeon H, Kulshrestha S, Lambert JW, Li S, Zhou X et al. (2024) Scenediffuser: Efficient and controllable driving simulation initialization and rollout. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Kesting A, Treiber M and Helbing D (2007) General lane-changing model mobil for car-following models. *Transportation Research Record* 1999(1): 86–94.
- Kong Z, Ping W, Huang J, Zhao K and Catanzaro B (2020) Diffwave: A versatile diffusion model for audio synthesis. In: *International Conference on Learning Representations*.
- LeCun Y, Chopra S, Hadsell R, Ranzato M and Huang F (2006) A tutorial on energy-based learning. *Predicting structured data* 1(0).
- Li Q, Peng Z, Feng L, Zhang Q, Xue Z and Zhou B (2022) Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence* 45(3): 3461–3475.
- Liu HX and Feng S (2024) Curse of rarity for autonomous vehicles. *nature communications* 15(1): 4808.
- Liu N, Li S, Du Y, Torralba A and Tenenbaum JB (2022) Compositional visual generation with composable diffusion models. In: *European Conference on Computer Vision*. Springer, pp. 423–439.
- Luo C (2022) Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970* .
- Mo X, Huang Z, Xing Y and Lv C (2022) Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. *IEEE Transactions on Intelligent Transportation Systems* .
- Montali N, Lambert J, Mougin P, Kuebler A, Rhinehart N, Li M, Gulino C, Emrich T, Yang Z, Whiteson S et al. (2023) The waymo open sim agents challenge. *arXiv preprint arXiv:2305.12032* .
- Nayakanti N, Al-Rfou R, Zhou A, Goel K, Refaat KS and Sapp B (2023) Wayformer: Motion forecasting via simple & efficient attention networks. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2980–2987.
- Ng AY, Russell S et al. (2000) Algorithms for inverse reinforcement learning. In: *Icmi*, volume 1. p. 2.
- Nichol AQ and Dhariwal P (2021) Improved denoising diffusion probabilistic models. In: *International Conference on Machine Learning*. PMLR, pp. 8162–8171.
- Niedoba M, Lavington JW, Liu Y, Lioutas V, Sefas J, Liang X, Green D, Dabiri S, Zwartsenberg B, Scibior A et al. (2023) A diffusion-model of joint interactive navigation. In: *Thirty-seventh Conference on Neural Information Processing Systems*.
- Phlion J, Peng XB and Fidler S (2023) Trajeglish: Learning the language of driving scenarios. *arXiv preprint arXiv:2312.04535* .
- Rempe D, Luo Z, Bin Peng X, Yuan Y, Kitani K, Kreis K, Fidler S and Litany O (2023) Trace and pace: Controllable pedestrian animation via guided trajectory diffusion. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 13756–13766.
- Rempe D, Phlion J, Guibas LJ, Fidler S and Litany O (2022) Generating useful accident-prone driving scenarios via a learned traffic prior. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 17305–17315.
- Rosbach S, James V, Großjohann S, Homoceanu S and Roth S (2019) Driving with style: Inverse reinforcement learning in general-purpose planning for automated driving. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2658–2665.
- Sadat S, Buhmann J, Bradely D, Hilliges O and Weber RM (2023) Cads: Unleashing the diversity of diffusion models through condition-annealed sampling. *arXiv preprint arXiv:2310.17347* .
- Seff A, Cera B, Chen D, Ng M, Zhou A, Nayakanti N, Refaat KS, Al-Rfou R and Sapp B (2023) Motionlm: Multi-agent motion forecasting as language modeling. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 8579–8590.
- Shi S, Jiang L, Dai D and Schiele B (2022) Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems* 35: 6531–6543.
- Shi S, Jiang L, Dai D and Schiele B (2023) Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *arXiv preprint arXiv:2306.17770* .
- Sohl-Dickstein J, Weiss E, Maheswaranathan N and Ganguli S (2015) Deep unsupervised learning using nonequilibrium thermodynamics. In: *International conference on machine learning*. PMLR, pp. 2256–2265.
- Song J, Meng C and Ermon S (2020a) Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* .
- Song Y and Ermon S (2019) Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems* 32.
- Song Y, Garg S, Shi J and Ermon S (2020b) Sliced score matching: A scalable approach to density and score estimation. In: *Uncertainty in Artificial Intelligence*. PMLR, pp. 574–584.
- Song Y and Kingma DP (2021) How to train your energy-based models. *arXiv preprint arXiv:2101.03288* .
- Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S and Poole B (2020c) Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456* .
- Sun S, Gu Z, Sun T, Sun J, Yuan C, Han Y, Li D and Ang Jr MH (2023) Drivescenegen: Generating diverse and realistic driving scenarios from scratch. *arXiv preprint arXiv:2309.14685* .
- Suo S, Regalado S, Casas S and Urtasun R (2021) Trafficsim: Learning to simulate realistic multi-agent behaviors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 10400–10409.
- Treiber M, Hennecke A and Helbing D (2000) Congested traffic states in empirical observations and microscopic simulations. *Physical review E* 62(2): 1805.
- Vincent P (2011) A connection between score matching and denoising autoencoders. *Neural computation* 23(7): 1661–1674.
- Wang T, Dhiman V and Atanasov N (2023a) Inverse reinforcement learning for autonomous navigation via differentiable semantic mapping and planning. *Autonomous Robots* 47(6): 809–830.

- Wang Y, Zhao T and Yi F (2023b) Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023. *arXiv preprint arXiv:2306.11868*.
- Wilson B, Qi W, Agarwal T, Lambert J, Singh J, Khandelwal S, Pan B, Kumar R, Hartnett A, Pontes JK et al. (2023) Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*.
- Wu W, Feng X, Gao Z and Kan Y (2024) Smart: Scalable multi-agent real-time simulation via next-token prediction. *arXiv preprint arXiv:2405.15677*.
- Xu C, Zhao D, Sangiovanni-Vincentelli A and Li B (2023a) Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles. In: *The Second Workshop on New Frontiers in Adversarial Machine Learning*.
- Xu D, Chen Y, Ivanovic B and Pavone M (2023b) Bits: Bi-level imitation for traffic simulation. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2929–2936.
- Zhang L, Peng Z, Li Q and Zhou B (2023a) Cat: Closed-loop adversarial training for safe end-to-end driving. In: *Conference on Robot Learning*. PMLR, pp. 2357–2372.
- Zhang L, Rao A and Agrawala M (2023b) Adding conditional control to text-to-image diffusion models. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 3836–3847.
- Zhang Z, Liniger A, Dai D, Yu F and Van Gool L (2023c) Trafficbots: Towards world models for autonomous driving simulation and motion prediction. *arXiv preprint arXiv:2303.04116*.
- Zhang Z, Sakaridis C and Van Gool L (2024) TrafficBots V1.5: Traffic simulation via conditional VAEs and Transformers with relative pose encoding. *arXiv preprint arXiv:2406.10898*.
- Zhong Z, Rempe D, Chen Y, Ivanovic B, Cao Y, Xu D, Pavone M and Ray B (2023a) Language-guided traffic simulation via scene-level diffusion. *arXiv preprint arXiv:2306.06344*.
- Zhong Z, Rempe D, Xu D, Chen Y, Veer S, Che T, Ray B and Pavone M (2023b) Guided conditional diffusion for controllable traffic simulation. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3560–3566.
- Zhou Z, Hu H, Chen X, Wang J, Guan N, Wu K, Li YH, Huang YK and Xue CJ (2024) Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. *arXiv preprint arXiv:2405.17372*.
- Zhou Z, Wang J, Li YH and Huang YK (2023) Query-centric trajectory prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 17863–17873.
- Ziebart BD, Maas AL, Bagnell JA, Dey AK et al. (2008) Maximum entropy inverse reinforcement learning. In: *Aaai*, volume 8. Chicago, IL, USA, pp. 1433–1438.

Appendix

We provide an extended analysis to show that one interpretation of score-based or diffusion-based generative model under imitation learning setting is learning the gradient descent step of a particular optimal control solver based on previous results from [Du and Mordatch \(2019\)](#); [Liu et al. \(2022\)](#); [Chi et al. \(2023\)](#).

Consider a dataset \mathbb{D} with scenario triplets $\mathcal{S} := (\mathbf{x}, \mathbf{u}, \mathbf{c})$ sampled independently from an unknown distribution p , whose probability density function can be factorized as $p(\mathcal{S}) = p(\mathbf{u}|\mathbf{c})p(\mathbf{c})$. In particular, we are interested in modeling the conditional probability $p(\mathbf{u}|\mathbf{c})$ of the joint control sequence \mathbf{u} w.r.t the scene context \mathbf{c} by $p_\theta(\mathbf{u}|\mathbf{c})$, from which we can sample (high probability) \mathbf{u} and implement realistic trajectories \mathbf{x} with known dynamics f .

Under Maximum Entropy IRL ([Ziebart et al. 2008](#)) formulation, $p_\theta(\mathbf{u}|\mathbf{c})$ can be structured as the Boltzmann distribution of an optimization objective:

$$p(\mathbf{u}|\mathbf{c}) \approx p_\theta(\mathbf{u}|\mathbf{c}) := \frac{1}{Z_\theta} \exp(-\mathcal{J}_\theta(\mathbf{x}(\mathbf{u}), \mathbf{u}; \mathbf{c})), \quad (26)$$

where Z_θ is the partition function (normalizing factor). [Equation 26](#) resembles the Energy-Based Models (EBM), which can be trained through the estimation of maximum likelihood ([LeCun et al. 2006](#); [Song and Kingma 2021](#)).

Corollary 1. *The maximum likelihood estimation of θ can be obtained by maximizing conditional log-likelihood of the dataset \mathbb{D} :*

$$\theta = \arg \max_{\hat{\theta}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\theta}}(\mathbf{u}|\mathbf{c})]. \quad (27)$$

Proof. Assuming we want to model the unknown data distribution p by $p_{\theta, \phi}(\mathcal{S}) = p_\theta(\mathbf{x}, \mathbf{u}|\mathbf{c})p_\phi(\mathbf{c})$. To estimate both θ and ϕ , we can maximize the log-likelihood as:

$$\begin{aligned} \max_{\hat{\theta}, \hat{\phi}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\theta}, \hat{\phi}}] &= \max_{\hat{\theta}, \hat{\phi}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\theta}}(\mathbf{x}, \mathbf{u}|\mathbf{c}) + \log p_{\hat{\phi}}(\mathbf{c})] \\ &= \max_{\hat{\theta}, \hat{\phi}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\theta}}(\mathbf{x}, \mathbf{u}|\mathbf{c})] + \max_{\hat{\theta}, \hat{\phi}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\phi}}(\mathbf{c})] \\ &= \max_{\hat{\theta}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\theta}}(\mathbf{x}, \mathbf{u}|\mathbf{c})] + \max_{\hat{\phi}} \mathbb{E}_{\mathcal{S} \sim \mathbb{D}} [\log p_{\hat{\phi}}(\mathbf{c})] \end{aligned}$$

Therefore, we can separate the problem and learn the parameter θ of conditional distribution via maximizing log-likelihood independent of learning $p_\phi(\mathbf{c})$. \square

Ideally, we can train a score function $s_\theta(\mathbf{u}|\mathbf{c})$ by score-matching ([Hyvärinen and Dayan 2005](#); [Vincent 2011](#); [Song and Ermon 2019](#); [Song et al. 2020b](#)) to approximate the gradient of $\log p(\mathbf{u}|\mathbf{c})$ w.r.t the control sequence (our random variable of interest):

$$\begin{aligned} \nabla_{\mathbf{u}} \log p(\mathbf{u}|\mathbf{c}) &\approx s_\theta(\mathbf{u}|\mathbf{c}) := \nabla_{\mathbf{u}} \log p_\theta(\mathbf{u}|\mathbf{c}) \\ &= -\nabla_{\mathbf{u}} \mathcal{J}_\theta(\mathbf{x}(\mathbf{u}), \mathbf{u}; \mathbf{c}) - \underline{\nabla_{\mathbf{u}} \log Z}. \end{aligned} \quad (28)$$

With the score function, $\nabla_{\mathbf{u}} \mathcal{J}_\theta$ is naturally obtained and can be directly used for gradient descent. However, since the dataset contains mostly near-optimal scenarios, the gradient estimation in suboptimal regions of the action space (away from demonstration data) may be inaccurate. Given a context \mathbf{c} , if we initialize an arbitrary \mathbf{u} far away from the minimizer of \mathcal{J} , errors in the score function may lead to suboptimal generation. To overcome this issue, we follow [Song et al. \(2020c\)](#) to diffuse the control \mathbf{u} through a forward stochastic differential equation (SDE):

$$d\tilde{\mathbf{u}} = h(\tilde{\mathbf{u}}, k)dk + g(k)d\mathbf{w}, \quad (29)$$

where \mathbf{w} is the wiener process and $k \in [0, K]$ is the continuous diffusion step. This process allows p to gradually transform into noised distributions p_k until it becomes a known distribution $p_K = \pi$. While the probability density function of p_k is unknown, we can sample a noised control $\tilde{\mathbf{u}} \sim p_k$ by first sampling $\mathbf{u} \sim p$ and perturbing with $p_k(\tilde{\mathbf{u}}|\mathbf{u})$ through SDE. This allows us to train a step-conditioned score function $\mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, k)$ to approximate $\nabla_{\tilde{\mathbf{u}}} \log p_k(\tilde{\mathbf{u}})$ by:

$$\begin{aligned} \theta = \arg \max_{\hat{\theta}} & \mathbb{E}_{S \sim p, k \sim \mathcal{U}(0, K)} \mathbb{E}_{\tilde{\mathbf{u}} \sim p_k(\cdot|\mathbf{u})} \\ & [\lambda(k) \|\nabla_{\tilde{\mathbf{u}}} \log p_k(\tilde{\mathbf{u}}|\mathbf{u}) - \mathbf{s}_{\hat{\theta}}(\tilde{\mathbf{u}}|\mathbf{c}, k)\|], \end{aligned} \quad (30)$$

where $\lambda(k)$ is a positive weighting function. At inference time, we can generate scenarios by first sampling $\tilde{\mathbf{u}}$ from the known distribution π and iterating through the reverse SDE (Anderson 1982):

$$d\tilde{\mathbf{u}} = [h(\tilde{\mathbf{u}}, k) - g^2(k)\mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, k)] dk + g(k)d\bar{\mathbf{w}}. \quad (31)$$

To connect the score-based model with IRL and EBM, we can view the forward diffusion as uplifting original data distribution into a higher-dimensional space. By injecting noise, we achieve good coverage over the entire action space in the final step K so that $\mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, K)$ are well defined for random $\tilde{\mathbf{u}}$. Sampling through reverse SDE can be interpreted as stochastic gradient descent towards high-probability regions with a fixed descent direction along the diffusion step, analogous to the direct shooting method in optimal control. We note that at low noise level k , as p_k is close to the original data distribution p , $\mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, k) \approx -\nabla_{\tilde{\mathbf{u}}} \mathcal{J}_\theta(\mathbf{x}, \mathbf{u}; \mathbf{c})$, which recovers our learning objective. Therefore, the generative modeling of scenarios can be viewed as an implicit solution of IRL by learning the gradient steps of trajectory optimization and solving the optimal control problem through sampling.