Versatile Scene-Consistent Traffic Scenario Generation as Optimization with Diffusion

Zixu Zhang[‡*], Zhiyu Huang[†*], Ameya Vaidya[§*], Chen Lv[†], Jaime Fernández Fisac[‡§]
[‡]Department of Electrical and Computer Engineering, Princeton University, United States
[†]School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore
[§]Department of Computer Science, Princeton University, United States

***Abstract*—Generating realistic and controllable agent behaviors in traffic simulation is crucial for the development of autonomous vehicles. This problem is often formulated as imitation learning (IL) from real-world driving data by either directly predicting future trajectories or inferring cost functions with inverse optimal control. In this paper, we draw a conceptual connection between IL and diffusion-based generative modeling and introduce a novel framework Versatile Behavior Diffusion (VBD) to simulate interactive scenarios with multiple traffic participants. Our model not only generates scene-consistent multi-agent interactions but also enables scenario editing through multi-step guidance and refinement. Experimental evaluations show that VBD achieves state-of-the-art performance on the Waymo Sim Agents benchmark. In addition, we illustrate the versatility of our model by adapting it to various applications. VBD is capable of producing scenarios conditioning on priors, integrating with model-based optimization, sampling multi-modal scene-consistent scenarios by fusing marginal predictions, and generating safety-critical scenarios when combined with a game-theoretic solver. Project website: https://sites.google.com/view/versatile-behavior-diffusion**

I. Introduction

Simulation plays a crucial role in validating the performance of autonomous driving systems. A primary challenge is generating diverse, realistic, and interactive traffic behaviors in a scalable and human-like manner. Conventional model or rule-based methods [59, 33] are inadequate to deliver realism and capture human interactions in the real world. Leveraging readily available driving logs [63, 2, 39], many studies have turned to data-driven methods and apply imitation learning (IL) techniques to model more realistic behaviors for traffic agents [16, 30, 70, 57, 67]. However, these methods mostly focus on a single agent or employ a shared policy across all traffic participants, leading to a lack of scene consistency and causing collisions when deployed in a highly interactive scenario [7]. To improve consistency, one can formulate the multi-agent behavior generation as a joint trajectory optimization problem, and explicitly learn a model to rationalize traffic interaction. Modeling optimization objective is a standard inverse optimal control (IOC) or inverse reinforcement learning (IRL) problem [41, 76]. Strategies vary from directly training a mapping from scenario to cost [48, 61, 28], or learning the cost weights for a set of handcraft heuristics through differentiable optimization layers [26, 24, 12]. However, these methods struggle to scale to general scenes or require cumbersome cost function design.

Our objective is to leverage diffusion models (a.k.a score-based models) [51, 53, 56, 23], a class of generative modeling methods that gradually recover structured data from random noise, for traffic scenario generation. Diffusion models enable effective behavior modeling for multi-agent joint futures and allow for iterative refinement. A critical aspect of diffusion models is controllability, which enables generating scenarios or editing agent behaviors to meet specific user requirements (e.g., cooperative or adversarial). Although diffusion models have been increasingly employed for generating agent behaviors [73, 32, 5, 72, 20, 65], training and controlling diffusion models for traffic agent interaction modeling remain challenging, and its connection with the classic formulation of this task under imitation learning has been overlooked. Therefore, we aim to bridge the conceptual gap between scenario generation and diffusion and develop a practical framework to model scene-consistent interactive traffic scenarios and enable user-specified behavior generation through structured guidance.

In this paper, we propose the **Versatile Behavior Diffusion** (VBD), which utilizes both the map and historical states of agents as conditional inputs to generate realistic and controllable traffic scenarios. VBD consists of three main components. First, we employ a query-centric Transformer-based [50, 75] scene context encoder, which encodes the states of agents and map polylines in their local coordinates and preserves relative information in attention, thus enhancing the multi-agent modeling performance. Second, we introduce a Transformer-based denoiser to generate scene-level joint behaviors of agents from noise. Third, we incorporate a Transformer-based multi-modal trajectory predictor to forecast the individual agents' intentions as behavior priors. While this predictor is not required for scenario generation, we find it improves training stability and can be incorporated with the denoiser to sample diverse scene-consistent scenarios. The versatility of our model can be realized through directly predicting scenarios through the denoiser in one step, controllable sampling of various tasks with user-specific objectives, further improving the generation quality by fusing the behavior priors, and generating long-tail safety-critical scenarios with game-theoretic guidance. The primary contributions of this paper are summarized as follows:

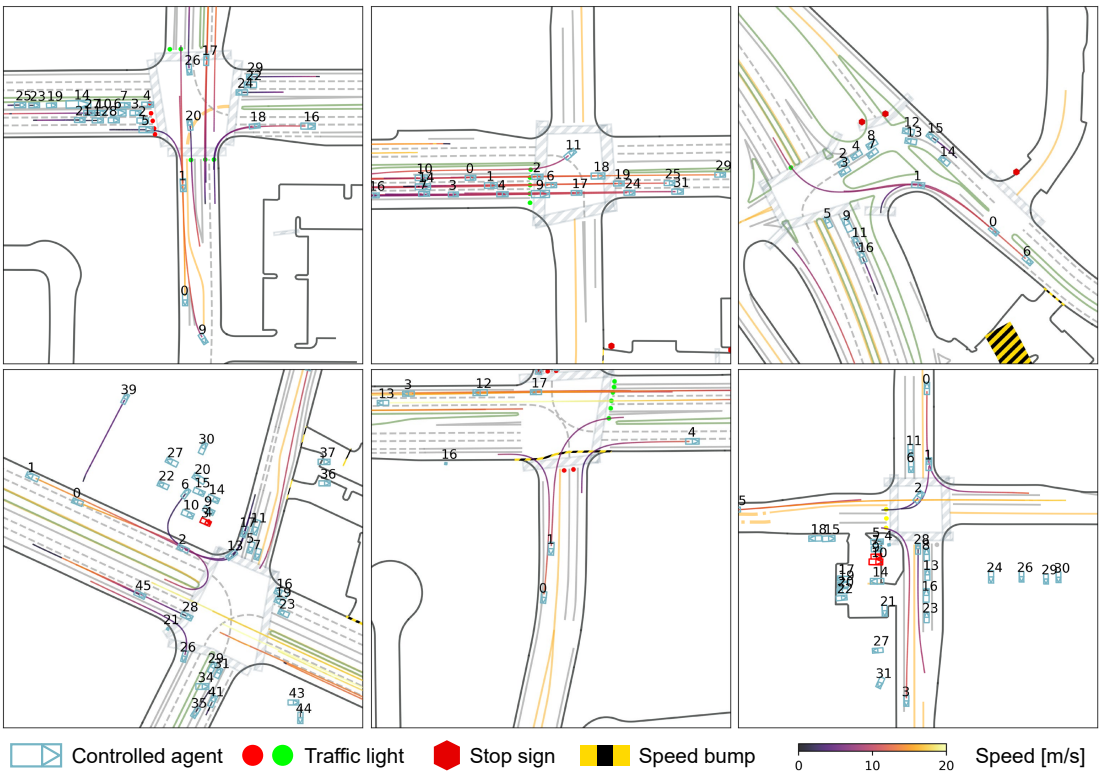

Figure 1. Performance of our VBD model on the Waymo Sim Agents task. The multi-agent diffusion policy is capable of controlling a large number of agents in an interactive and map-adherent manner for traffic simulation.

1) We draw conceptual insights to connect diffusion generative modeling with the classic imitation learning formulation of traffic scenario generation.
2) We propose the Versatile Behavior Diffusion model to facilitate realistic and controllable traffic simulation, which achieves the state-of-the-art performance of multi-agent interaction modeling on the Waymo Sim Agents Benchmark.
3) We demonstrate the versatility of the proposed model through various guidance methods, including cost function, goal priors, and game theory structure, to generate diverse, realistic traffic scenarios *within and beyond* the training data distribution.

II. Related Work

Traffic Simulation. There has been a growing shift towards learning-based methods to enhance the realism and interactivity of traffic simulations [58]. BITS [66] employs imitation learning to simulate agent behaviors by inferring high-level intent and replicating low-level driving actions. The socially-controllable behavior generation model proposed in [6] focuses on simulating social and interactive behaviors. Symphony [30] integrates learning-based policies with parallel beam search to further enhance realism. Trajeglish [44] introduces a multi-agent sequence of motion tokens using a GPT-like encoder-decoder architecture, achieving state-of-the-art realism. Another line of research focuses on generating safety-critical or adversarial scenarios to test the robustness of driving systems. STRIVE [47] generates challenging scenarios that can induce collisions with the ego planner through optimization in latent space. Similarly, AdvDO [3] and KING [21] utilize optimization-based methods to generate adversarial trajectories for robust planning. TrafficBots [70] introduces a multi-agent policy conditioned on specific goals to generate configurable behaviors, though it faces challenges in goal formation. CAT [68] chooses conflicting trajectories from the predicted behavior distribution of agents. However, those methods cannot produce interactions during the safety-critical scenario, as the trajectory of the agent under attack are often assumed to be fixed or known to the adversarial counterpart. Moreover, current simulation models often lack versatility since they are trained for either maximum likelihood (realistic) behaviors or adversarial scenarios. We aim to build a unified framework for both tasks to obtain realistic and controllable traffic simulation.

Multi-modal Behavior Prediction. Behavior prediction is closely related to traffic simulation or behavior cloning tasks [70, 57, 16]. Recent advances in learning-based behavior prediction models have significantly increased the accuracy of both agent-wise motion prediction [27, 40, 75] and scene-level multi-agent joint prediction [38, 50, 25]. Leveraging a large amount of real-world data, they are capable of generating accurate multi-modal distributions of possible behaviors for multiple agents in a scene. In addition, diffusion models

have been applied in behavior prediction and generation tasks [32, 10, 43], demonstrating superior results in multi-agent motion prediction. Our proposed model integrates multi-modal behavior prediction as action priors or feasible high-level intentions for traffic agents, which aligns with realistic distributions and can be used in guided diffusion to generate specific agent behavior.

Diffusion Models. Score-based model, a.k.a Diffusion models [51, 53, 23, 56, 52], have gained widespread popularity in various generative tasks, including image [69], audio [34], and video [14] generation. Recently, diffusion models have shown great potential in traffic scenario generation due to their diversity and controllability. SceneDM [20] utilizes a diffusion model to generate joint and consistent future motions of all agents in a scene. MotionDiffuser [32] employs a diffusion-based representation for joint multi-agent motion prediction and introduces a constrained sampling framework for controlled trajectory sampling. CTG [73] combines diffusion modeling and STL rules to enforce traffic rules on generated trajectories. CTG++ [72] leverages Large Language Models to translate user queries into loss functions, guiding the diffusion model toward generating query-compliant scenarios. TRACE [46] proposes a guided diffusion model to generate future trajectories for pedestrians, employing analytical loss functions to impose trajectory constraints. DiffScene [65] and [5] utilizes guided diffusion with adversarial optimization objectives to simulate safety-critical scenarios. However, a conceptual understanding of Diffusion models under traffic simulation settings has been overlooked in previous works. We aim to explore the optimal training strategies for traffic behavior modeling using diffusion model and propose various sampling strategies, to enhance realism and versatility.

III. Problem Formulation

Traffic Scenario Generation as Optimization. Consider a traffic scenario $\mathcal{S} = (\mathbf{x}, \mathbf{u}, \mathbf{c})$ with 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 $\mathbf{u} = (\mathbf{u}^1, \cdots, \mathbf{u}^A) \in \mathbb{R}^{A\times T\times D_u}$. The context of the scene $\mathbf{c} \in \mathbb{R}^{D_c}$ includes information regarding the road map, the status of traffic lights, 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:

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

where f represents the discrete-time joint dynamics. If our goal is to generate realistic (statistically representative) scenarios, the objective $\mathcal{J}_\theta$ should be designed to incentivize real-world driving behaviors, whether through a statistical loss metric (behavior cloning) or by using inverse reinforcement learning (IRL). Once $\mathcal{J}_\theta$ is established, we need to tractably find an optimal joint control sequence $\mathbf{u}$, either through numerical optimization or reinforcement learning.

Generative Modeling as Trajectory Optimization. Instead of solving the aforementioned IL problem in two steps, prior works [22, 18] have shown a strong connection between IL and generative modeling under a generative-adversarial training framework. Extending the analysis of [13, 36, 8], 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 $\mathbf{x}$ is a rollout of control $\mathbf{u}$ with known dynamics f, we can factorize the probability density function as $p(\mathcal{S}) = p(\mathbf{u}|\mathbf{c})p(\mathbf{c})$. Under Maximum Entropy IRL [76] formulation, we aim to approximate $p(\mathbf{u}|\mathbf{c})$ as the Boltzmann distribution of an optimization objective:

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

where Z_θ is the partition function. Eq. (2) resembles the Energy-Based Models (EBM) [35, 55]. Specifically, we want to learn the parameter θ of the optimization objective that maximizes the conditional log-likelihood of the dataset $\mathbb{D}$:

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

Ideally, we can employ score-matching [29, 60, 53, 54] to directly learn the gradient of $\mathcal{J}_\theta$ w.r.t the control (our random variable of interest) as the score function:

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

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 [13, 54, 23, 56, 52] 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 [23, 52] and are later generalized as score-based models by [56]. 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:

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

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, $\mathbf{s}_\theta(\mathbf{u}|\mathbf{c}, k \to 0) \approx -\nabla_{\mathbf{u}} \mathcal{J}_\theta(\mathbf{x}, \mathbf{u}; \mathbf{c})$, 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) [23], which is also known as the discrete-time variance-preserving score-based SDE (VP-SDE) [56]. The equivalence between the original DDPM training objective and the score-matching loss (Eq. (5)) has been shown in [37].

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 (Eq. (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 [13] and Classifier Guidance in diffusion model [11]. Specifically, we can sample the reverse process with a conditional score function:

$$\begin{aligned} \nabla_{\tilde{\mathbf{u}}} \log p_k(\tilde{\mathbf{u}}|\mathbf{c}, y) &\approx \mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, y, k) \\ &= \mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c}, k) + \nabla_{\tilde{\mathbf{u}}} \log p_k(y|\tilde{\mathbf{u}}, \mathbf{c}), \end{aligned} \tag{6}$$

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 [11, 31]. Prior works [73, 72, 32] proposed 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 objective without training. We analyze these guidance methods in Section IV.

IV. Versatile Behavior Diffusion Model

Model Structure. The Versatile Behavior Diffusion model consists of three main components as illustrated in Fig. 2. 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 [50]. 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 control sequence $\hat{\mathbf{u}}$ from $\hat{\mathbf{c}}$ and noised control $\tilde{\mathbf{u}}$ at step k. This allows our model to perform one-step generation, while still maintaining the capability of iterative sampling and refinement. 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 [50]. All three modules utilize a stack of query-centric self-attention and cross-attention blocks for flexibility and scalability. Details regarding the model architecture can be found in the supplementary materials.

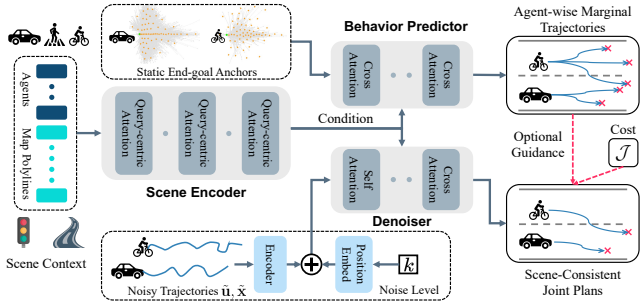

Figure 2. Overview of the proposed VBD model. The input scenario tokens are encoded through a query-centric Transformer-based scenario encoder. The behavior predictor generates marginal multi-modal trajectories. The denoiser predicts the joint multi-agent future trajectories while attending to themselves 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.

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:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}_\theta} = &\mathbb{E}_{\mathcal{S}\sim p, k\sim\mathcal{U}(0,K)} \mathbb{E}_{\tilde{\mathbf{u}}\sim p_k(\cdot|\mathbf{u})} \big[\\ &\lambda(k) \mathcal{S}\mathcal{L}_1(\hat{\mathbf{x}}(\mathcal{D}_\theta(\hat{\mathbf{c}}, \tilde{\mathbf{u}}, k)) - \mathbf{x}) \big], \end{aligned} \tag{7}$$

which is defined as the the Smooth L1 loss between ground-truth trajectories $\mathbf{x}$ and the trajectories $\hat{\mathbf{x}}$ rollout from $\hat{\mathbf{u}}$.

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 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 $\mathcal{P}_\psi$, we first select the mode m^* that most closely matches the ground truth trajectory of each agent and minimize the predictor loss defined as:

$$\mathcal{L}_{\mathcal{P}_\psi} = \mathbb{E}_{\mathcal{S}\sim p}\Bigg[\sum_{a=1}^{A} \mathcal{SL}_1\left(\hat{\mathbf{x}}(\hat{\mathbf{u}}^{a,m^*}) - \mathbf{x}^a\right) + \beta CE(m^*, \hat{\omega}^a)\Bigg], \tag{8}$$

which penalizes the smooth $L1$ difference between the ground truth trajectory of each agent and the trajectory from the best mode m^*, and encourages a higher probability to be assigned on this mode through a Cross-Entropy loss.

Scenario Generation with VBD. To employ the VBD model for scenario generation or behavior modeling, we can use the following implementations.

- *One-step generation.* We directly query the denoiser with a randomly sampled noised $\tilde{\mathbf{u}}$. Since the encoder provides rich scene context information, we empirically find one-step generation can sample high-quality joint scenarios. However, due to strong conditioning on the scene context, one-step sampling often collapses to a single mode of scenario.
- *Multi-step sampling*. To improve sample diversity, we utilize a standard DDPM sampling approach by iteratively querying the denoiser and updating the noised $\tilde{\mathbf{u}}$.
- *Guided Sampling*. To impose constraints or targets on specific agents, we use the classifier guidance method [11] by alternatively updating the noised $\tilde{\mathbf{u}}$ with denoiser output and $\nabla_{\tilde{\mathbf{u}}} \log p_k(y|\tilde{\mathbf{u}}, \mathbf{c})$. We evaluate two approaches for approximation: 1. CTG [73, 72] directly approximates $\log p_k(y|\tilde{\mathbf{u}}, \mathbf{c}) \approx \log p(y|\tilde{\mathbf{u}}, \mathbf{c}) = -\mathcal{J}_y(\mathbf{x}(\tilde{\mathbf{u}}), \tilde{\mathbf{u}}; \mathbf{c})$; 2. MotionDiffuser (MD) [32] approximates $\log p_k(y|\tilde{\mathbf{u}}, \mathbf{c}) \approx \mathcal{J}_y(\mathbf{x}(\mathcal{D}_\theta(\tilde{\mathbf{u}})), \mathcal{D}_\theta(\tilde{\mathbf{u}}); \mathbf{c})$, where $\mathcal{D}_\theta(\tilde{\mathbf{u}})$ is the one-step generation result from denoiser.

Diverse Scene-consistent Scenario Generation. Sampling diverse outputs from a conditional diffusion model is challenging, especially when the denoising strongly relies on the context information [49]. On the other hand, behavior predictors capture the multi-modal trajectories of individual agents but will result in scene inconsistency if marginal trajectories are naively combined, because the predictor alone cannot ensure the collective coherence necessary for realistic multi-agent scenario generation. VBD can used as an effective scenario optimizer and produce diverse and scene-consistent scenarios by first sampling goal positions from the behavior predictor (or any other models) and generating joint trajectories matching individual goals using guided sampling with the denoiser.

Game-theoretic Safety-critical Scenario Generation. It is essential to expose AVs to a variety of simulated safety-critical scenarios to stress-test and improve their planning ability. To generate *interactive* safety-critical scenarios, we model the two-agent interaction as a map-constrained pursuit-evasion game, where the pursuer aims to cause a collision with the evader, while the evader attempts to avoid it. One approach to solving such a game is iterative best response (IBR), such as gradient descent–ascent (GDA). Specifically, we can apply τ-GDA to guarantee local convergence to a stable *minimax equilibrium* of the pursuit-evasion game [17]. We update the pursuer more frequently than the evader, providing information advantage to the adversarial agents.

Generating a scenario with a game-theoretic solver alone often leads to unrealistic results that disregard the scene context. Instead, leveraging realistic traffic behavior modeled by VBD, we propose a *game-guided diffusion* scheme (**??**) by alternative denoising, performing gradient descent and ascent for agents, and updating the noised $\tilde{\mathbf{u}}$ 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 only performing gradient updates on selected timesteps.

V. Experiments

Platform. We conduct the experiments on the large-scale Waymo Open Motion Dataset (WOMD) [15], which includes 486,995 9-second logged real-world scenarios for training and 44,097 scenarios for validation. The dataset provides tracks of all agents and corresponding vectorized maps in each scenario. We employ the Waymax simulator [19] as the interface for closed-loop traffic simulation. In the Waymo Sim Agents benchmark [39], we evaluate our model on 44,920 testing scenarios. To improve closed-loop rollout performance and stability [4], trajectories are replanned in the receding horizon fashion.

Implementation Details. During training, we consider $A = 32$ agents, 256 map polylines (each containing 30 waypoints), and 16 traffic lights in the scene. VBD generates $T = 80$ steps of future control sequences with step size $0.1s$ based on (up to) 11 steps of past trajectories. The scene encoder contains 6 query-centric-attention Transformer layers, and the embedding dimension is 256. The behavior predictor comprises 4 cross-attention Transformer layers and generates $M = 64$ possible trajectories for each agent along with respective probability estimates. The denoiser includes two decoding blocks with a total of 4 Transformer layers. A cosine variance schedule is adopted in the diffusion process, employing $K = 10$ diffusion steps, and the maximum value of $\beta(k)$ is set to 0.999. The predicted raw actions are standardized during the diffusion process, with the mean and standard deviation of actions set to 0 and 1. The scalability of

Table I
TESTING RESULTS ON THE 2024 WAYMO SIM AGENTS BENCHMARK

Model	Realism Meta (↑)	Kinematic (↑)	Interactive (↑)	Map-based (↑)	minADE (↓)
SMART[64]	0.7511	0.4445	0.8050	0.8571	1.5447
BehaviorGPT[74]	0.7473	0.4333	0.7997	0.8593	1.4147
MVTE[62]	0.7302	0.4503	0.7706	0.8381	1.6770
TrafficBotsV1.5[71]	0.6988	0.4304	0.7114	0.8360	1.8825
VBD (Ours)	0.7200	0.4169	0.7819	0.8137	1.4743

Table II
VBD IMPROVES SCENE-CONSISTENCY FROM MARGINAL BEHAVIOR PRIORS

Method	Collision [%] ↓	Off-road [%] ↓	Wrong-way [%] ↓	Kin. [%] ↓	ADE [m] ↓
Marginal Prediction Only	5.61±0.27	6.19±0.05	0.86±0.09	**0.31±0.02**	1.113±0.012
Post-Optimization via Denoiser	**2.23±0.15**	**1.26±0.10**	**0.52±0.11**	0.32±0.01	**0.974±0.005**

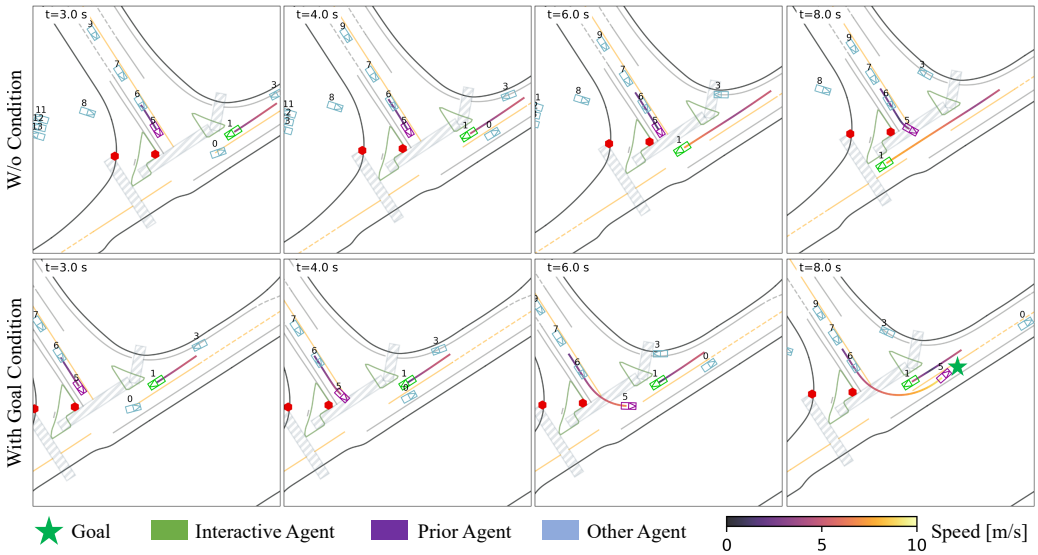

Figure 3. VBD produces scene-consistent traffic scenarios when certain agents are conditioned on a given prior. **Top**: Nominal VBD rollout without guidance generates a scene-consistent scenario, where Vehicle 5 (in purple) waits at the stop sign and then precedes. **Bottom**: Using goal-guided diffusion 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 III
COST-GUIDED SAMPLING IMPROVES GENERATION QUALITY

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 (CTG)	1.67±0.58	1.51±0.18	0.86±0.08	0.25±0.01	1.711±0.020
VBD + Collision (MD)	**1.11±0.23**	**1.15±0.17**	**0.60±0.12**	0.25±0.01	1.113±0.010

Transformer blocks allows VBD to be adapted to any number of agents during inference.

Evaluation Metrics. We follow the official evaluation metrics of Waymo Sim Agents benchmark [39] encompassing kinematic, interactive, and map-based features, and a meta realism metric is calculated as a weighted sum of these features. For the assessment of controllable scenario generation, we utilize a subset of 500 WOMD interactive validation scenarios selected by prior work [68]. We employ metrics provided by the Waymax simulator [19], such as off-road incidents, collisions, wrong-way, kinematic infeasibility, and average displacement error (ADE) of the rollout trajectories w.r.t the ground-truth ones (log divergence).

Results on Sim Agents Benchmark. The Waymo Sim Agents benchmark requires 32 independent rollouts of simulation from one scenario, and each rollout contains the $x/y/z$ coordinates and headings for up to 128 agents over an 8-second future horizon. We evaluate the performance of

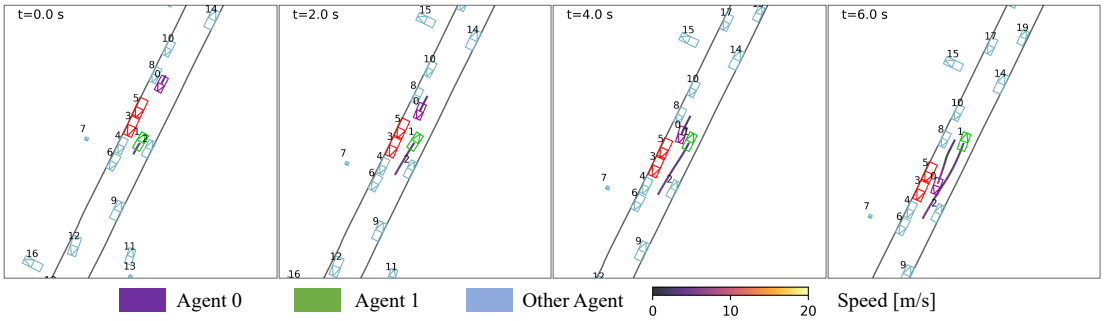

Figure 4. Composition of denoiser model-based objective can improve generation quality. Two vehicles interact and coordinate in a narrow passage scenario with collision cost function guidance. (Note: Vehicles 3 and 5 have been in collision since the initial step.)

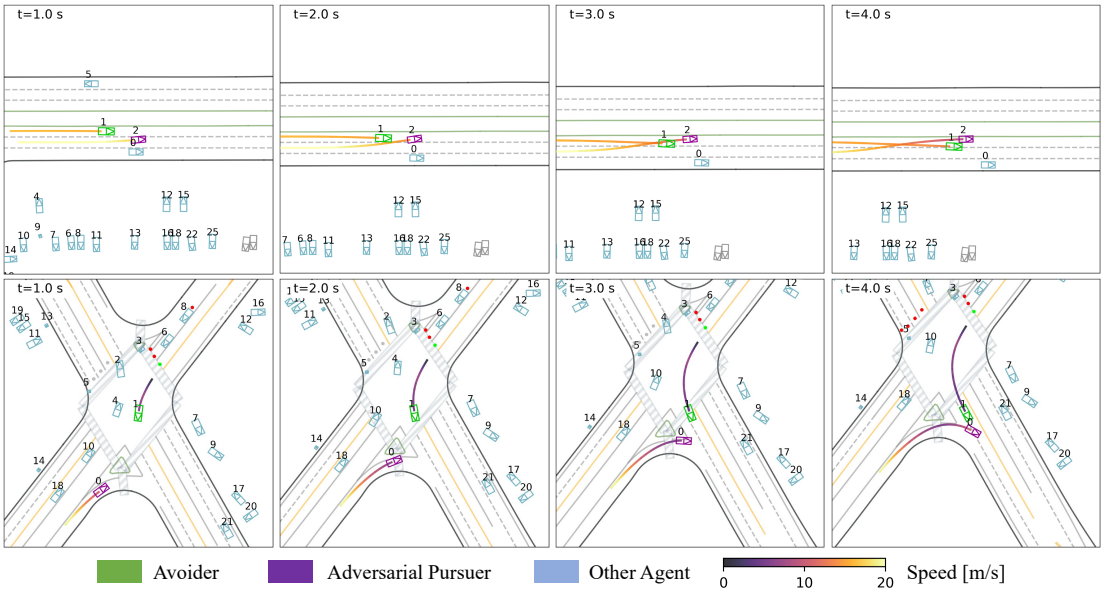

Figure 5. Results of game theoretical guided generation. **Top**: The adversarial pursuer merges in front of the evader, performs a brake check, and attempts to cause a rear-end crash. The evader immediately switches its lane and avoids the collision. **Bottom**: The adversarial pursuer merges aggressively to the adjacent lane, and the evader yields to the pursuer by slowing down.

scenario generation from VBD's joint diffusion policy by initiating the denoising process sampling from unit Gaussian noise. As summarized in Table I, we demonstrate that the behavior generation performance from VBD closely matches state-of-the-art models. We present a selection of qualitative simulation results in Fig. 1, showcasing the model's ability to generate diverse and realistic traffic scenarios. Further analysis suggests VBD is capable of facilitating realistic agent interactions through its joint multi-agent diffusion policy.

Denosing as Scene-consistent Scenario Optimization. In this experiment, we evaluate the effectiveness of VBD as a scenario optimization tool to generate scene-consistent interactions from marginal behavior priors. Specifically, the baseline (marginal) method directly samples actions from the behavior predictor in a receding horizon fashion with the replan frequency as 1 Hz. For each agent in the scene, we select the final positions of agents' most likely predicted trajectories as the goal prior and apply guided sampling to minimize the mean $L2$ distance between the denoised results and goals. We compare their performance with 500 scenarios of the WOMD interactive subset and test each method with three different random seeds. The average $L2$ distance between the goals and rollout trajectories is 2.1833 m, which shows VBD can reach goal points better than marginal samples. The results in Table II indicate VBD significantly reduces the collision rate and our model better captures interactions between agents. Moreover, under the circumstance when priors were selected from suboptimal samples, e.g. off-road or wrong-way, VBD can alleviate these cases and generate scenarios that adhere to scene context.

Additionally, we find that VBD is capable of generating interactive conditional predictions, when there is only one or a subset of agents are conditioned by marginal priors. Here, a human user supervises the simulation process and manually selects the target agent and its target goal. As seen in Fig. 3, by conditioning on the behavior that vehicle 5

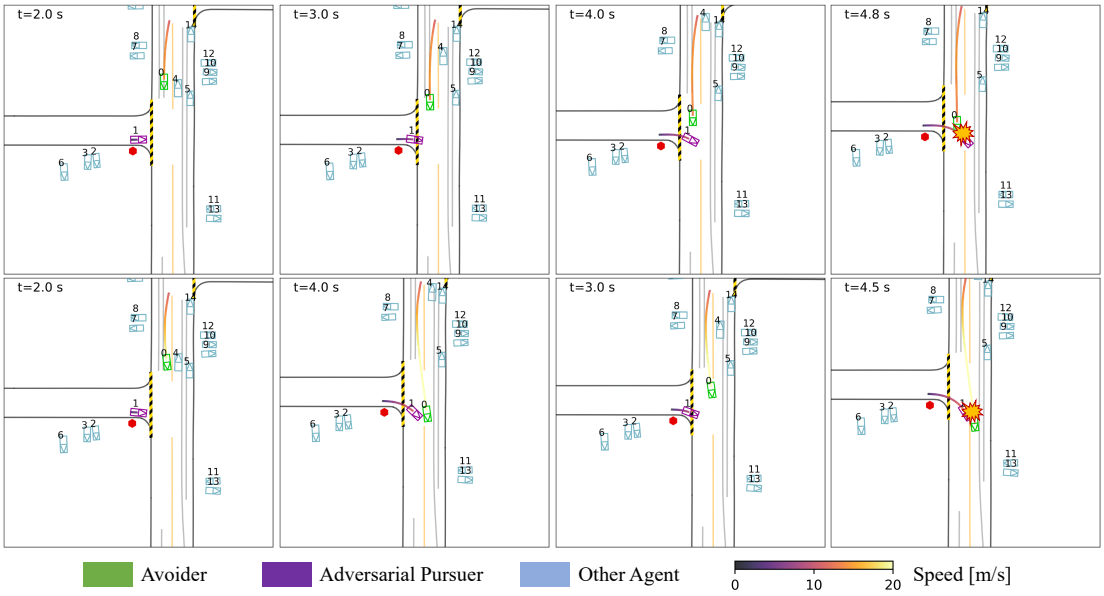

Figure 6. VBD with game-theoretic guidance can generate reactive and long-tail safety-critical scenarios. **Top**: The adversarial pursuer suddenly pulls out and causes a rear-end crash with the evader. **Bottom**: Stochastic sampling allows us to generate diverse safety-critical behaviors. The pursuer forces the green vehicle to drive to the opposite side of the road while causing a collision.

makes a left turn ignoring the stop sign, VBD's diffusion policy can generate a scenario such that vehicle 1 is forced to yield for the other vehicle. On the other hand, the nominal diffusion policy rollout without any guidance follows a completely different but still scene-consistent traffic ordering. Such behaviors can be observed in a variety of scenes, and additional examples can be found in the Supplementary Materials.

Improving Generation Quality via Composition with Model-Based Objectives. The performance of VBD can be further improved when combined with a model-based optimization objective. In this experiment, we introduce a simple collision avoidance cost to maximize each agent's minimum distance from others along the horizon. We compare the performance of different guidance methods including CTG guidance [73] and MotionDiffuser (MD) guidance [32]. As shown in Table III, this simple objective further reduces the collision rate. We observe that under the same cost, CTG has worse performance. One explanation is that noised $\tilde{\mathbf{u}}$ sampled from $\mathbf{u}$ with low $\mathcal{J}_y$ may have high costs, and assuming the p_k is equivalent to the data distribution p in CTG could lead to incorrect gradient at high noise level. The qualitative results in Fig. 4 also illustrate that, by composing with a model-based optimization objective, the model can generate collision-free interaction in the challenging narrow-passage scene.

Reactive Safety-critical Scenario Generation. As illustrated in Fig. 5 and Fig. 6, our model is capable of generating safety-critical scenarios with the proposed game-theoretic guided sampling framework in various scenes. Distinct from previous works [47, 5, 68], which sample adversarial behaviors given fixed trajectories of vehicles being pursued, our guidance strategy actively optimizes the actions of both the purser and evader and leads to highly-interactive scenarios. Our proposed method facilitates the generation of highly realistic scenarios, especially with regard to adversarial behavior, by ensuring it remains proportionately adversarial and responsive to the ego vehicle's actions. This strategy overcomes the shortcomings of previous methods, which tend to generate unrealistically aggressive adversarial tactics and are less useful in stress-testing AVs. Detailed formulations can be found in the Supplementary Materials.

VI. Conclusions

In this paper, we introduce a scene-consistent scenario optimizer leveraging the diffusion model, called Versatile Behavior Diffusion. This model integrates a denoiser for diffusion inference, which accounts for the joint futures of all agents, as well as a behavior predictor that provides the prior distribution of the agents' multi-modal trajectories. The experiment results demonstrate that our model is capable of generating diverse, realistic, and interactive scenarios, achieving state-of-the-art performance on the Waymo Sim Agents benchmark. Moreover, we show the model's versatility in various tasks employing different structured guidance strategies, such as model-based optimization objectives, prior-based scene editing or conditional generation, and game-theoretical guidance for crafting safety-critical scenarios. Future research could aim to enhance the model's runtime efficiency and apply it to AV planning tests.

References

[1] Anderson, B.D.: Reverse-time diffusion equation models. Stochastic Processes and their Applications **12**(3), 313–326 (1982)

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

[3] Cao, Y., Xiao, C., Anandkumar, A., Xu, D., Pavone, M.: Advdo: Realistic adversarial attacks for trajectory prediction. In: European Conference on Computer Vision. pp. 36–52. Springer (2022)

[4] Chang, W.J., Hu, Y., Li, C., Zhan, W., Tomizuka, M.: Analyzing and enhancing closed-loop stability in reactive simulation. In: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC). pp. 3665–3672. IEEE (2022)

[5] Chang, W.J., Pittaluga, F., Tomizuka, M., Zhan, W., Chandraker, M.: Controllable safety-critical closed-loop traffic simulation via guided diffusion. arXiv preprint arXiv:2401.00391 (2023)

[6] Chang, W.J., Tang, C., Li, C., Hu, Y., Tomizuka, M., Zhan, W.: Editing driver character: Socially-controllable behavior generation for interactive traffic simulation. arXiv preprint arXiv:2303.13830 (2023)

[7] Chen, Y., Ivanovic, B., Pavone, M.: 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 (June 2022)

[8] Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. arXiv preprint arXiv:2303.04137 (2023)

[9] Chiu, H.k., Smith, S.F.: Collision avoidance detour for multi-agent trajectory forecasting. arXiv preprint arXiv:2306.11638 (2023)

[10] Choi, Y., Mercurius, R.C., Shabestary, S.M.A., Rasouli, A.: Dice: Diverse diffusion model with scoring for trajectory prediction. arXiv preprint arXiv:2310.14570 (2023)

[11] Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in neural information processing systems **34**, 8780–8794 (2021)

[12] Diehl, C., Klosek, T., Krueger, M., Murzyn, N., Osterburg, T., Bertram, T.: Energy-based potential games for joint motion forecasting and control. In: 7th Annual Conference on Robot Learning (2023)

[13] Du, Y., Mordatch, I.: Implicit generation and modeling with energy based models. Advances in Neural Information Processing Systems **32** (2019)

[14] Esser, P., Chiu, J., Atighehchian, P., Granskog, J., Germanidis, A.: Structure and content-guided video synthesis with diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7346–7356 (2023)

[15] Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C.R., Zhou, Y., Yang, Z., Chouard, A., Sun, P., Ngiam, J., Vasudevan, V., McCauley, A., Shlens, J., Anguelov, D.: 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 (2021)

[16] Feng, L., Li, Q., Peng, Z., Tan, S., Zhou, B.: Trafficgen: Learning to generate diverse and realistic traffic scenarios. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 3567–3575. IEEE (2023)

[17] Fiez, T., Ratliff, L., Mazumdar, E., Faulkner, E., Narang, A.: Global convergence to local minmax equilibrium in classes of nonconvex zero-sum games. Advances in Neural Information Processing Systems **34**, 29049–29063 (2021)

[18] Finn, C., Christiano, P., Abbeel, P., Levine, S.: A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. arXiv preprint arXiv:1611.03852 (2016)

[19] Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., et al.: Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. arXiv preprint arXiv:2310.08710 (2023)

[20] Guo, Z., Gao, X., Zhou, J., Cai, X., Shi, B.: Scenedm: Scene-level multi-agent trajectory generation with consistent diffusion models. arXiv preprint arXiv:2311.15736 (2023)

[21] Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., Geiger, A.: King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In: European Conference on Computer Vision. pp. 335–352. Springer (2022)

[22] Ho, J., Ermon, S.: Generative adversarial imitation learning. Advances in neural information processing systems **29** (2016)

[23] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)

[24] Huang, Z., Karkus, P., Ivanovic, B., Chen, Y., Pavone, M., Lv, C.: Dtpp: Differentiable joint conditional prediction and cost evaluation for tree policy planning in autonomous driving. arXiv preprint arXiv:2310.05885 (2023)

[25] Huang, Z., Liu, H., Lv, C.: 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 (October 2023)

[26] Huang, Z., Liu, H., Wu, J., Lv, C.: Differentiable integrated motion prediction and planning with learnable

cost function for autonomous driving. IEEE transactions on neural networks and learning systems (2023)
[27] Huang, Z., Mo, X., Lv, C.: Multi-modal motion prediction with transformer-based neural network for autonomous driving. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 2605–2611. IEEE (2022)
[28] Huang, Z., Wu, J., Lv, C.: Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning. IEEE transactions on intelligent transportation systems **23**(8), 10239–10251 (2021)
[29] Hyvärinen, A., Dayan, P.: Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research **6**(4) (2005)
[30] Igl, M., Kim, D., Kuefler, A., Mougin, P., Shah, P., Shiarlis, K., Anguelov, D., Palatucci, M., White, B., Whiteson, S.: Symphony: Learning realistic and diverse agents for autonomous driving simulation. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 2445–2451. IEEE (2022)
[31] Janner, M., Du, Y., Tenenbaum, J., Levine, S.: Planning with diffusion for flexible behavior synthesis. In: International Conference on Machine Learning. pp. 9902–9915. PMLR (2022)
[32] Jiang, C., Cornman, A., Park, C., Sapp, B., Zhou, Y., Anguelov, D., et al.: Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9644–9653 (2023)
[33] Kesting, A., Treiber, M., Helbing, D.: General lane-changing model mobil for car-following models. Transportation Research Record **1999**(1), 86–94 (2007)
[34] Kong, Z., Ping, W., Huang, J., Zhao, K., Catanzaro, B.: Diffwave: A versatile diffusion model for audio synthesis. In: International Conference on Learning Representations (2020)
[35] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. Predicting structured data **1**(0) (2006)
[36] Liu, N., Li, S., Du, Y., Torralba, A., Tenenbaum, J.B.: Compositional visual generation with composable diffusion models. In: European Conference on Computer Vision. pp. 423–439. Springer (2022)
[37] Luo, C.: Understanding diffusion models: A unified perspective. arXiv preprint arXiv:2208.11970 (2022)
[38] Mo, X., Huang, Z., Xing, Y., Lv, C.: Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. IEEE Transactions on Intelligent Transportation Systems (2022)
[39] Montali, N., Lambert, J., Mougin, P., Kuefler, A., Rhinehart, N., Li, M., Gulino, C., Emrich, T., Yang, Z., Whiteson, S., et al.: The waymo open sim agents challenge. arXiv preprint arXiv:2305.12032 (2023)
[40] Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K.S., Sapp, B.: Wayformer: Motion forecasting via simple & efficient attention networks. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 2980–2987. IEEE (2023)
[41] Ng, A.Y., Russell, S., et al.: Algorithms for inverse reinforcement learning. In: Icml. vol. 1, p. 2 (2000)
[42] Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning. pp. 8162–8171. PMLR (2021)
[43] Niedoba, M., Lavington, J.W., Liu, Y., Lioutas, V., Sefas, J., Liang, X., Green, D., Dabiri, S., Zwartsenberg, B., Scibior, A., et al.: A diffusion-model of joint interactive navigation. In: Thirty-seventh Conference on Neural Information Processing Systems (2023)
[44] Philion, J., Peng, X.B., Fidler, S.: Trajeglish: Learning the language of driving scenarios. arXiv preprint arXiv:2312.04535 (2023)
[45] Qian, C., Xiu, D., Tian, M.: The 2nd place solution for 2023 waymo open sim agents challenge. arXiv preprint arXiv:2306.15914 (2023)
[46] Rempe, D., Luo, Z., Bin Peng, X., Yuan, Y., Kitani, K., Kreis, K., Fidler, S., Litany, O.: 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 (2023)
[47] Rempe, D., Philion, J., Guibas, L.J., Fidler, S., Litany, O.: 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 (2022)
[48] Rosbach, S., James, V., Großjohann, S., Homoceanu, S., Roth, S.: 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). pp. 2658–2665. IEEE (2019)
[49] Sadat, S., Buhmann, J., Bradely, D., Hilliges, O., Weber, R.M.: Cads: Unleashing the diversity of diffusion models through condition-annealed sampling. arXiv preprint arXiv:2310.17347 (2023)
[50] Shi, S., Jiang, L., Dai, D., Schiele, B.: Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. arXiv preprint arXiv:2306.17770 (2023)
[51] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International conference on machine learning. pp. 2256–2265. PMLR (2015)
[52] Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
[53] Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems **32** (2019)
[54] Song, Y., Garg, S., Shi, J., Ermon, S.: Sliced score matching: A scalable approach to density and score estimation. In: Uncertainty in Artificial Intelligence. pp. 574–584. PMLR (2020)

[55] Song, Y., Kingma, D.P.: How to train your energy-based models. arXiv preprint arXiv:2101.03288 (2021)
[56] Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
[57] Sun, S., Gu, Z., Sun, T., Sun, J., Yuan, C., Han, Y., Li, D., Ang Jr, M.H.: Drivescenegen: Generating diverse and realistic driving scenarios from scratch. arXiv preprint arXiv:2309.14685 (2023)
[58] Suo, S., Regalado, S., Casas, S., Urtasun, R.: Trafficsim: Learning to simulate realistic multi-agent behaviors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10400–10409 (2021)
[59] Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical review E **62**(2), 1805 (2000)
[60] Vincent, P.: A connection between score matching and denoising autoencoders. Neural computation **23**(7), 1661–1674 (2011)
[61] Wang, T., Dhiman, V., Atanasov, N.: Inverse reinforcement learning for autonomous navigation via differentiable semantic mapping and planning. Autonomous Robots **47**(6), 809–830 (2023)
[62] Wang, Y., Zhao, T., Yi, F.: Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023. arXiv preprint arXiv:2306.11868 (2023)
[63] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., et al.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. arXiv preprint arXiv:2301.00493 (2023)
[64] Wu, W., Feng, X., Gao, Z., Kan, Y.: Smart: Scalable multi-agent real-time simulation via next-token prediction. arXiv preprint arXiv:2405.15677 (2024)
[65] Xu, C., Zhao, D., Sangiovanni-Vincentelli, A., Li, B.: Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles. In: The Second Workshop on New Frontiers in Adversarial Machine Learning (2023)
[66] Xu, D., Chen, Y., Ivanovic, B., Pavone, M.: Bits: Bi-level imitation for traffic simulation. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 2929–2936. IEEE (2023)
[67] Zhang, C., Tu, J., Zhang, L., Wong, K., Suo, S., Urtasun, R.: Learning realistic traffic agents in closed-loop. In: 7th Annual Conference on Robot Learning (2023)
[68] Zhang, L., Peng, Z., Li, Q., Zhou, B.: Cat: Closed-loop adversarial training for safe end-to-end driving. In: Conference on Robot Learning. pp. 2357–2372. PMLR (2023)
[69] Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3836–3847 (2023)
[70] Zhang, Z., Liniger, A., Dai, D., Yu, F., Van Gool, L.: Trafficbots: Towards world models for autonomous driving simulation and motion prediction. arXiv preprint arXiv:2303.04116 (2023)
[71] Zhang, Z., Sakaridis, C., Van Gool, L.: Trafficbots v1. 5: Traffic simulation via conditional vaes and transformers with relative pose encoding. arXiv preprint arXiv:2406.10898 (2024)
[72] Zhong, Z., Rempe, D., Chen, Y., Ivanovic, B., Cao, Y., Xu, D., Pavone, M., Ray, B.: Language-guided traffic simulation via scene-level diffusion. arXiv preprint arXiv:2306.06344 (2023)
[73] Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., Pavone, M.: Guided conditional diffusion for controllable traffic simulation. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 3560–3566. IEEE (2023)
[74] Zhou, Z., Hu, H., Chen, X., Wang, J., Guan, N., Wu, K., Li, Y.H., Huang, Y.K., Xue, C.J.: Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. arXiv preprint arXiv:2405.17372 (2024)
[75] Zhou, Z., Wang, J., Li, Y.H., Huang, Y.K.: Query-centric trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17863–17873 (2023)
[76] Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K., et al.: Maximum entropy inverse reinforcement learning. In: Aaai. vol. 8, pp. 1433–1438. Chicago, IL, USA (2008)

Supplementary Material

APPENDIX

A. 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 $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$. A unicycle dynamics function is utilized to transform agent actions into states, which is adopted to approximate the dynamics of all agent types, 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:

$$\begin{aligned} x(t+1) &= x_t + v_x(t)\Delta t, \\ y(t+1) &= y_t + v_y(t)\Delta t, \\ \psi(t+1) &= \psi(t) + \dot{\psi}\Delta t, \\ v(t+1) &= \sqrt{v_x(t)^2 + v_y(t)^2} + \dot{v}\Delta t, \\ v_x(t+1) &= v(t+1)\cos\psi(t+1), \\ v_y(t+1) &= v(t+1)\sin\psi(t+1). \end{aligned} \tag{S1}$$

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:

$$\begin{aligned} \dot{v}(t) &= \frac{v(t+1) - v(t)}{\Delta t}, \ v(t) = \sqrt{v_x(t)^2 + v_y(t)^2}, \\ \dot{\psi}(t) &= \frac{\psi(t+1) - \psi(t)}{\Delta t}. \end{aligned} \tag{S2}$$

B. Model Structure

Scene Encoder. The encoder processes three main inputs: the agent history tensor ($[A, T_h, D_a]$), the map polyline tensor ($[Ml, Mp, D_p]$), and the traffic lights tensor ($[Mt, D_t]$). Here, T_h denotes the number of historical steps, Ml the number of polylines, Mp the number of waypoints per polyline, Mt the number of traffic lights, and $Ml + Mt = M$ 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 $Mp = 30$ waypoints, includes attributes like x, y coordinates, direction angle, the traffic light state controlling the lane, and lane type. The traffic lights tensor encompasses the 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. We choose $A = 32$ agents, $T_h = 11$ historical steps, $Ml = 256$ map polylines, and $M_t = 16$ traffic lights.

We first encode the agent history tensor, utilizing a shared GRU network to produce a tensor of shape $[A, D]$, which is then combined with the agent type embedding. For map polylines, an MLP is employed for encoding, resulting in a tensor of shape $[Ml, Mp, D]$. This is followed by max-pooling along the waypoint axis to produce a tensor of shape $[Ml, D]$. For traffic lights, we only encode their light status using an MLP, yielding a tensor of shape $[Mt, D]$. These tensors are then concatenated to form the initial scene encoding tensor with shape $[A + M, D]$. The initial scene encoding is further processed using query-centric Transformer layers to symmetrically encode the interrelationships among scene components. 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. For example, for elements i and j, the relative position $\Delta_{ij} = [\Delta_x, \Delta_y, \Delta_{heading}]$ is computed and then encoded into the edge attribute using an MLP, resulting in relation encoding tensor $\mathbf{e}^{ij} = MLP(\Delta_{ij})$. The query-centric attention mechanism for a query element $\mathbf{q}^i$ operates as follows:

$$\begin{aligned} QCA(Q^i, K, V, \mathbf{e}) = \text{softmax}\Big(\frac{\mathbf{q}^i}{\sqrt{D}}[\{\mathbf{k}^j \\ +\mathbf{e}^{ij}\}_{j\in\Omega(j)}]^T\Big)(\{\mathbf{v}^j + \mathbf{e}^{ij}\}_{j\in\Omega(j)}), \end{aligned} \tag{S3}$$

where $\mathbf{k}^j, \mathbf{v}^j$ represent the key and value elements respectively, each containing relevant element-centric information, and $j \in \Omega(j)$ indicates the index of other tokens. Other standard operations in Transformers, such as multi-head attention, feed-forward networks, and layer normalization, remain unchanged. In practice, batch operations can be applied to implement the multi-head query-centric attention mechanism efficiently. The encoder consists of 6 query-centric Transformer layers to process the initial scene encoding, the embedding dimension $D = 256$, and the final output tensor retains the same shape as $[A + M, D]$.

Denoiser. The denoiser part processes three types of input: noised actions ($[A, T_f, 2]$) derived from the ground-truth state and action trajectories, the noise level, and the scene encoding tensor. Each agent's action at every timestep $a = [\dot{v}, \dot{\psi}]^T$ consists of acceleration and yaw rate, while the state comprises coordinates, heading, and velocity (x, y, ψ, v). Here, actions are reduced to a shorter length T_f from T, by replicating the same action over multiple timesteps, denoted as T_a. The noise level is encoded via an embedding layer to a tensor of shape $[1, 1, D]$. Noised actions are converted into noisy states using a forward dynamics model and subsequently encoded into a tensor of shape $[A, T_f, D]$ using an MLP. The encoded noisy states are combined with the noise level embedding and a temporal embedding ($[1, T_f, D]$) to create the initial trajectory embedding. For the denoising process, two decoding

blocks, each comprising two Transformer decoder layers, are applied to predict clean actions. Within the decoding block, 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 [24] 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. Since the elements are encoded in a query-centric manner, the decoding layers still require relative positional information between elements, which can be obtained from the encoder.

Following the Transformer decoding stage, the resulting trajectory embedding is fed into an MLP to decode the clean actions tensor of shape $[A, T_f, 2]$. Subsequently, clean states ($[A, T, 3]$, encompassing x, y, ψ) are deduced from these predicted actions using a differentiable dynamics model. In this work, we choose $T = 80$ and $T_a = 2$, resulting in $T_f = 40$ and thus significantly reducing computational demands while maintaining high accuracy.

Behavior predictor. The behavior predictor generates the marginal distributions of possible behaviors for agents by directly decoding from the encoded scene conditions. To accurately predict the probabilities of possible goals, the predictor takes as input the static anchors for agents in local coordinates with shape $[A, Mo, 2]$ as the modality query inputs. The anchors contain $Mo = 64$ typical x, y coordinates at $T = 80$ extracted from data using the K-means algorithm, and vary across different agent types such as vehicles, pedestrians, and cyclists. We utilize an MLP encoder to encode these anchors into a tensor of shape $[A, Mo, 256]$. This encoding is then combined with the agent encoding of shape $[A, 1, 256]$, to form an initial query tensor with dimensions $[A, Mo, 256]$. Then we employ four cross-attention Transformer layers where relative relation encoding is still used in the attention mechanism. The predictor iteratively refines its predictions through several decoding layers and finally, an MLP decoding head is added to decode the possible action sequences for all agents, resulting in a tensor of $[A, Mo, T_f, 2]$. These action trajectories are transformed into state trajectories of shape $[A, Mo, T, 4]$ using the same differentiable dynamics model, and each waypoint in the trajectory contains the state (x, y, ψ, v). Another MLP layer decodes the embedding after the Transformer layers to derive marginal scores (probabilities) of these predicted trajectories, with shape $[A, Mo]$.

C. Model Training

Predictor. The training of the predictor follows the multi-trajectory-prediction (MTP) loss setting. This involves selecting the best-predicted trajectories and computing the loss relative to the ground-truth trajectories. To determine the best-predicted indices for an agent, the following criterion is applied:

$$m^* = \begin{cases} \arg\min_i ||ac^i - x_T||, & \text{if } x_T \text{ is valid,} \\ \arg\min_i || \sum_t (\hat{x}_t^i - x_t)||, & \text{otherwise,} \end{cases} \tag{S4}$$

where ac^i is the static anchor point, x_t is the ground-truth point of the trajectory, and $\hat{x}_t^i$ 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 head, cross-entropy loss is utilized, comparing the predicted logits with the given indices. Note that this loss function is computed marginally, and time steps lacking ground-truth data or invalid agents are excluded from the loss calculation.

Denoiser. The denoiser is trained to recover the clean trajectories under various noise levels. At each training step, noise level k and Gaussian noise ϵ are sampled and applied to corrupt the ground-truth trajectories. 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 training procedure of the denoiser is described in Algorithm 1.

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

$$\mathcal{L} = \mathcal{L}_{\mathcal{D}_\theta} + \gamma \mathcal{L}_{\mathcal{P}_\psi}, \tag{S5}$$

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

The hyperparameters used in the total loss function is $\gamma = 0.5$, and $\beta = 0.05$ is used in the predictor loss. 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 and is executed on four NVIDIA A100 GPUs, with a batch size of 14 per GPU.

The guided diffusion algorithm using a score function is illustrated in Algorithm 2. In practice, multiple gradient steps are employed to more effectively perturb the predicted means. The score function $\mathcal{J}_y$ varies with the structure y, which can be defined as the goal for individual agents or a soft constraint, such as obeying the speed limit or avoiding collisions.

Algorithm 1 Training of denoiser

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

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

Algorithm 2 Guided diffusion with score function

Require: Denoiser $\mathcal{D}_\theta$, score function $\mathcal{J}_y$, diffusion steps K, gradient steps N_g, scaling parameter α, standard deviation σ_k

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

D. Prior Guidance

Consider a scenario where we need to specify desired behaviors for N_a agents. We can utilize the predicted behavior priors from the VBD model to heuristically determine the target behaviors or goals for each target agent, represented by $\{g^i\}_{i=1:N_a}$. Without sophisticated cost and feature design, the score function based on behavior priors is significantly simplified as follows:

$$\mathcal{J}_{goal} = -\sum_{i=1:N_a} \mathcal{SL}_1(g^i - x_T^i), \tag{S6}$$

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.

E. Cost Guidance

Another form of guidance is differentiable cost functions, which can be used to further improve the generation quality, such as collision avoidance and staying on-road. The formulation of these cost functions is outlined as follows. Specifically, the collision avoidance cost function is formulated as:

$$\mathcal{J}_{overlap} = \sum_{t=1}^{T}\sum_{i,j}^{A} d_{ij}(\mathbf{x}_t)\mathbb{1}(d_{ij}(\mathbf{x}_t) < \epsilon_d), \tag{S7}$$

where d_{ij} represents the Minkowski distance between the footrpints of agents i and j at time t. The parameter ϵ_d is the threshold for defining potential collision. The on-road guidance function is designed to prevent agents from straying off drivable roads. The on-road cost function is formulated as:

$$\mathcal{J}_{onroad} = -\sum_{t=1}^{T}\sum_{i}^{A} \text{relu}\left(d_r(\mathbf{x}_t^i)\right), \tag{S8}$$

where d_r denotes the signed distance between the bounding box of an agent and the nearest road edge. A negative distance indicates that the agent's position is on the road, while a positive distance suggests off-road. It is important to note that this cost is applied only to vehicles that are not off-road at the initial step.

F. Sim Agents

We evaluate the performance of both diffusion policy and behavior predictor in the Waymo Sim Agents benchmark. The testing of the behavior prediction model employs an open-loop planner, which means that the model directly generates 8-second multi-agent trajectories and these trajectories are rolled out in the Waymax simulator. Although this setting violates the closed-loop testing protocol of the Sim Agents benchmark, it serves as a useful indicator of our model's ability to accurately generate behavior distributions for multiple agents. For the diffusion policy, we directly roll out the policy's output in the simulator without replanning, primarily due to time constraints, as the benchmark requires testing in a large number of scenarios. It is important to note that despite the absence of replanning, the diffusion policy adheres to the benchmark's closed-loop protocol, because of the incorporation of a causal relationship mask in the attention-based decoding process. The decoding of an agent's state at a given timestep relies only on the static map and the states of other agents in preceding timesteps, without any sharing of information about their future intentions.

During testing, we only let the policy/model control the first 32 agents in the scene, while the remaining agents follow a constant velocity policy. For the evaluation of behavior priors, the model first generates marginal behavior distributions for the agents. Subsequently, we sample 32 times from these distributions to obtain 32 different scenario rollouts. For the diffusion policy, a batch of Gaussian noise is directly sampled, allowing the generation of 32 varied scenarios simultaneously, thereby expediting the sampling process.

The results in Table S2 indicate that both the diffusion policy and behavior prediction perform very well. Notably, the joint diffusion policy outperforms the marginal behavior prediction model in terms of interactive score, which highlights the advantage of the diffusion policy for joint behavior modeling. These findings suggest that our VBD model is reliable at generating accurate marginal behavior priors for agents and can facilitate realistic agent interactions through its joint multi-agent diffusion policy.

G. Evaluation Metrics

In addition to the Sim Agents benchmark evaluation metrics, we employ the following metrics provided by the Waymax simulation platform for the controllable traffic scenario generation testing.

Off-road. A binary metric indicates if a vehicle drives off the road, based on its position relative to oriented roadgraph points. If a vehicle is on the left side of an oriented road edge, it is considered on the road; otherwise, the vehicle is considered off-road. This metric is averaged over all valid agents in the scene.

Collision. A binary metric identifies whether an agent has collided with another agent. For each pair of objects, if their 2D bounding boxes overlap in the same timestep, they are considered as collision. This metric is computed as an average across all valid agents in the scene.

Wrong-way. A binary metric that measures 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 a duration exceeding 1 second. The calculation of this metric is an average across all valid agents in the scene.

Kinematic infeasibility. A binary metric that computes whether a transition step is kinematically feasible for the vehicle. The limit of acceleration magnitude is empirically set to be 6 m/s^2 and the steering curvature magnitude to be 0.3 m^{-1}. The metric is averaged for all valid agents in the scene.

Log divergence. This metric quantifies deviation from logged behavior using the average displacement error (ADE), defined as the L2 distance between an agent's current and logged positions at each timestep. The metric is averaged across all timesteps and all valid agents in the scene.

H. Guided Scenario Generation

For the guided diffusion testing, we employ $Ng = 5$ gradient steps and set the strength parameter $\alpha = 0.1$, and guidance is applied throughout all diffusion steps. We limit the simulation to a maximum of 32 agents to manage computational resources, and any additional agents present in the original scenarios, mostly static vehicles, are excluded. The experiments are conducted on selected 500 9-second scenarios from the WOMD validation interactive subset. The simulation starts at 1 second and extends over a horizon of 8 seconds, and the simulation model's replanning frequency is 1 Hz. We show additional results on guided scenario generation in video format.

I. Reactive Simulation

The primary requirement of simulation is to ensure that the agents respond realistically to the actions of the ego vehicle. To assess our model's performance, we conduct reactivity tests where the ego vehicle is decoupled from the VBD model and is instead controlled by another planner. We test the reactivity of our model across 500 scenarios where the labeled self-driving car (SDC) is controlled by a log-playback planner or an IDM-route planner. In this experiment, only the historical movements of the SDC are provided to the model, and thus the VBD model does not control the ego vehicle but rather coordinates the behaviors of the remaining agents. The results are presented in Table S3, which indicate that the reactivity of the diffusion policy is superior compared to marginal behavior prediction or behavior cloning method, as evidenced by a significant reduction in ego vehicle collisions. Employing an IDM-route planner, which may not mimic human-like driving and deviate substantially from the actual trajectory, results in poorer performance. Conversely, using a log-playback planner with better human likeness enables our VBD model to generate reactive behaviors to the ego vehicle. Visualizations of the reactive simulation outcomes with the log-playback planner are available in video format at https://sites.google.com/view/versatile-behavior-diffusion.

J. Safety-critical Scenario Generation

Using the VBD model with the proposed game guidance or unsafe prior guidance, we demonstrate additional results of safety-critical scenario generation in video format at https://sites.google.com/view/versatile-behavior-diffusion.

K. Ablation Studies

We investigate the factors that influence the training of the diffusion policy, focusing on the number of diffusion steps and the integration of a prior predictor in a multi-task learning framework. We train the denoiser under various settings and test the performance of the learned policy across identical testing scenarios. For the evaluation of diffusion steps, we only train the denoiser with a frozen the encoder. The results are presented in Table S4, where we found increasing number of diffusion steps does not imply better generation metrics. One explanation is that, since scenario generation is heavily conditioned on scene contexts, the initial denoising steps can already generate reasonable results. When sampled without guidance, extensive steps may inject excessive randomness and lead to undesirable behaviors, such as collision and off-road.

In addition, we also studied the effectiveness of auxiliary marginal prediction in Table S5. We found that the inclusion of marginal predictor in the joint training significantly improve the training of the encoder, and leading to better generation results with the denoiser.

The results suggest that increasing the number of diffusion steps has a negative effect on simulation performance. This is likely because, different from image generation, behavior generation is strongly conditioned on the scene context, which means that small variance gaps between steps are not necessary

Table S1
TESTING RESULTS ON THE 2023 WAYMO SIM AGENTS BENCHMARK

Method	Realism Meta	Kinematic	Interactive	Map-based	minADE (m)
VBD (Ours)	0.6342	0.4212	0.7256	**0.8200**	**1.3509**
Trajeglish [44]	**0.6437**	0.4157	**0.7646**	0.8104	1.6146
MVTA [62]	0.6361	0.4175	0.7390	0.8139	1.8698
MTR+++ [45]	0.6077	0.3597	0.7172	0.8151	1.6817
SceneDM [20]	0.5821	**0.4244**	0.6675	0.7000	2.4186
CAD [9]	0.5314	0.3357	0.5638	0.7688	2.3146
Joint-Multipath++[39]	0.4766	0.1792	0.6322	0.6833	2.0517

Table S2
TESTING RESULTS OF THE VBD MODEL ON THE WAYMO SIM AGENTS BENCHMARK

Method	Realism Meta	Kinematic	Interactive	Map-based	minADE
VBD-Diffusion	0.6342	0.4212	0.7256	0.8200	1.3509
VBD-BP	0.6315	0.4261	0.7177	0.8216	1.3400

Table S3
TESTING RESULTS OF THE VBD MODEL ON REACTIVE SIMULATION

Model	Ego planner	Collision w/ ego [%]	Off-road [%]	Log divergence [m]
BP	Log-playback	10.60	4.49	0.979
BP	IDM-route	13.20	5.38	1.070
Diffusion	Log-playback	4.80	1.43	1.082
Diffusion	IDM-route	8.40	2.26	1.107

Table S4
ABLATION RESULTS ON THE DIFFUSION POLICY

Diffusion	Collision [%] ↓	Off-road [%]↓	Wrong-way [%] ↓	ADE [m]	Guidance Collision [%]
step=1	**1.95±0.03**	1.24±0.02	0.67±0.03	**0.823±0.002**	–
step=5	2.18±0.16	1.23±0.04	0.64±0.06	0.961±0.004	1.30
step=10	2.47±0.09	**1.21±0.14**	**0.57±0.02**	1.010±0.007	**1.11**
step=25	2.68±0.15	1.52±0.06	0.64±0.09	1.061±0.010	1.27
step=50	3.60±0.45	1.67±0.17	0.67±0.01	1.123±0.012	1.51
step=100	3.68±0.63	1.81±0.24	0.69±0.05	1.163±0.011	1.50

for refining predictions. Moreover, adding more diffusion steps can introduce excessive randomness into the sampling process, leading to a decrease in behavior modeling performance. The result indicates that using 10 steps of diffusion can balance the performance of normal and guided sampling, which can also significantly improve the inference speed. Regarding the effect of the behavior predictor, the result reveals that it benefits the diffusion policy by improving the learning of the encoder part of the model.

L. Connection Between Score-based Model and Optimization

In this subsection, 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 [13, 36, 8].

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 the 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.tthe 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 [76] formulation, $p_\theta(\mathbf{u}|\mathbf{c})$ can be structured as the Boltzmann distribution of an optimization objective:

$$p(\mathbf{u}|\mathbf{c}) \approx p_\theta(\mathbf{u}|\mathbf{c}) := \frac{1}{Z_\theta} \exp(-\mathcal{J}_\theta(\mathbf{x}(\mathbf{u}), \mathbf{u}; \mathbf{c})), \tag{S9}$$

where Z_θ is the partition function (normalizing factor). Eq. (S9) resembles the Energy-Based Models (EBM), which can be trained through the estimation of maximum likelihood [35, 55].

Corollary 0.1. *The maximum likelihood estimation of θ can be obtained by maximizing **conditional** log-likelihood of the dataset $\mathbb{D}$:*

$$\theta = \arg\max_{\hat{\theta}} \mathbb{E}_{\mathcal{S}\sim\mathbb{D}}[\log p_{\hat{\theta}}(\mathbf{u}|\mathbf{c})]. \tag{S10}$$

Table S5
ABLATION RESULTS ON THE EFFECTIVENESS OF AUXILIARY PREDICTION TASK

Diffusion	Collision [%] ↓	Off-road [%]↓	Wrong-way [%] ↓	ADE [m]	Guidance Collision [%]
w. Predictor	2.47±0.09	**1.21±0.14**	**0.57±0.02**	1.010±0.007	**1.11**
w/o predictor	3.01±0.14	1.96±0.09	0.90±0.10	1.218±0.007	1.41

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 $p_\phi(\mathbf{c})$. ■

Ideally, we can train a score function $\mathbf{s}_\theta(\mathbf{u}|\mathbf{c})$ by score-matching [29, 60, 53, 54] to approximate the gradient of $\log p(\mathbf{u}|\mathbf{c})$ w.r.tthe control sequence (our random variable of interest):

$$\begin{aligned}\nabla_{\mathbf{u}} \log p(\mathbf{u}|\mathbf{c}) \approx \mathbf{s}_\theta(\mathbf{u}|\mathbf{c}) &:= \nabla_{\mathbf{u}} \log p_\theta(\mathbf{u}|\mathbf{c}) \\ &= -\nabla_{\mathbf{u}} \mathcal{J}_\theta(\mathbf{x}(\mathbf{u}),\mathbf{u};\mathbf{c}) - \cancelto{0}{\nabla_{\mathbf{u}} \log Z}.\end{aligned} \tag{S11}$$

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 [56] to diffuse the control $\mathbf{u}$ through a forward stochastic differential equation (SDE):

$$d\tilde{\mathbf{u}} = h(\tilde{\mathbf{u}},k)dk + g(k)d\mathbf{w}, \tag{S12}$$

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:

$$\begin{aligned}\theta = \arg\max_{\hat{\theta}} \mathbb{E}_{\mathcal{S}\sim p, k\sim\mathcal{U}(0,K)} \mathbb{E}_{\tilde{\mathbf{u}}\sim p_k(\cdot|\mathbf{u})}[\\ \lambda(k)\|\nabla_{\tilde{\mathbf{u}}} \log p_k(\tilde{\mathbf{u}}|\mathbf{u}) - \mathbf{s}_{\hat{\theta}}(\tilde{\mathbf{u}}|\mathbf{c},k)\|],\end{aligned} \tag{S13}$$

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 [1]:

$$d\tilde{\mathbf{u}} = \left[h(\tilde{\mathbf{u}},k) - g^2(k)\mathbf{s}_\theta(\tilde{\mathbf{u}}|\mathbf{c},k)\right]dk + g(k)d\bar{\mathbf{w}}. \tag{S14}$$

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, $\mathbf{s}_\theta(\mathbf{u}|\mathbf{c},k) \approx -\nabla_{\mathbf{u}}\mathcal{J}_\theta(\mathbf{x},\mathbf{u};\mathbf{c})$, 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.

M. DDPM: Denoising Diffusion Probabilistic Model

One popular method that uses the idea of learning reverse diffusion process to construct data is the Diffusion Denoising Probabilistic Model (DDPM) [23]. The discrete-time formulation of the forward diffusion process of DDPM can be described as [42]:

$$q(\tilde{\mathbf{u}}_1,\cdots,\tilde{\mathbf{u}}_K|\mathbf{u}) := \prod_{k=1}^{K} q(\tilde{\mathbf{u}}_k|\tilde{\mathbf{u}}_{k-1}), \tag{S15}$$

$$q(\tilde{\mathbf{u}}_k|\tilde{\mathbf{u}}_{k-1}) := \mathcal{N}\left(\tilde{\mathbf{u}}_k; \sqrt{1-\beta_k}\tilde{\mathbf{u}}_{k-1}, \beta_k\mathbf{I}\right), \tag{S16}$$

where $\beta_k \in (0,1)$ is the k-th noise scale from a predefined noise scheduling, $\mathbf{u}$ is the clean 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:

$$\begin{aligned}\tilde{\mathbf{u}}_k &= \sqrt{\alpha_k}\tilde{\mathbf{u}}_{k-1} + \sqrt{1-\alpha_k}\epsilon_{k-1} \\ &= \sqrt{\bar{\alpha}_k}\mathbf{u} + \sqrt{1-\bar{\alpha}_k}\epsilon,\ \epsilon \sim \mathcal{N}(0,\mathbf{I}).\end{aligned} \tag{S17}$$

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 follows:

$$p_\theta(\tilde{\mathbf{u}}_0,\cdots,\tilde{\mathbf{u}}_K|\mathbf{c}) := q(\tilde{\mathbf{u}}_K)\prod_{i=1}^{K} p_\theta(\tilde{\mathbf{u}}_{k-1}|\tilde{\mathbf{u}}_k,\mathbf{c}), \tag{S18}$$

$$p_\theta(\tilde{\mathbf{u}}_{k-1}|\tilde{\mathbf{u}}_k, \mathbf{c}) := \mathcal{N}\left(\tilde{\mathbf{u}}_{k-1}; \mu_\theta(\tilde{\mathbf{u}}_k, \mathcal{D}_\theta(\tilde{\mathbf{u}}_k, k, \mathbf{c})), \sigma_k^2 \mathbf{I}\right), \tag{S19}$$

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:

$$\mu_k := \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\hat{\mathbf{u}}_k + \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\tilde{\mathbf{u}}_k, \tag{S20}$$

A cosine schedule is employed for $\bar{\alpha}_k$, which is formulated as:

$$\bar{\alpha}_k = \frac{f_n(k)}{f_n(0)},\ f_n(k) = \cos\left(\frac{k/K + s}{1+s}\cdot\frac{\pi}{2}\right)^2, \tag{S21}$$

where K is the total number of diffusion steps, and $s = 0.008$ is a small offset to prevent β_k from being too small near $k = 0$. The variances β_k are calculated as $\beta_k = 1 - \frac{\bar{\alpha}_k}{\bar{\alpha}_{k-1}}$, with a maximum value capped at 0.999. Once the variance schedule is established, the values of α_k, β_k, and $\bar{\alpha}_k$ can be calculated.

In the reverse diffusion process, it is often necessary to incorporate constraints or objectives to guide the process towards desired outcomes, denoted as y. This guidance is implemented by subtly altering the predicted mean of the model at each denoising step, similar to the CTG method [73]. Therefore, the denoising step is modified to impose guidance in the form of a score function $\mathcal{J}_y$ as:

$$\tilde{\mu}_k = \mu_k + \alpha\sigma_k\nabla_{\mu_k}\mathcal{J}_y(\mu_k), \tag{S22}$$

where α 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 propose an alternative approach similar to the MotionDiffuser method [32]: calculating the objective function using the one-step generation result from denoiser rather than the noisy mean. This modification is expressed as:

$$\tilde{\mu}_k = \mu_k + \alpha\sigma_k\nabla_{\tilde{\mathbf{u}}_k}\mathcal{J}_y(\mathcal{D}_\theta(\tilde{\mathbf{u}}_k)), \tag{S23}$$

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 $\mathcal{D}$.

N. Connecting DDPM with Score-Based Model

In previous subsections, we consider a general form of score-based model with continuous-time SDE proposed by Song *et al.*[56]. To draw the connection, DDPM is a discrete-time variance-preserving SDE (VP-SDE) with

$$f(k,x) = -\frac{1}{2}\beta(k), \quad g(k) = \sqrt{\beta(k)}. \tag{S24}$$

whose proof is stated in Appendix B. of [56]. i-th diffusion step in DDPM can be viewed as discrete sampling along continuous time step k.

To train the denoiser, Ho *et al.*[23] used a simplified loss to minimize difference between sampled noise ϵ the predicted noise $\epsilon_\theta(\cdot)$ from the denoiser by:

$$\begin{aligned}\mathcal{L}_{\text{DDPM}} &= \mathbb{E}_{i\sim[1,N],\mathcal{S}\sim\mathbb{D},\epsilon\sim\mathcal{N}}\left[\|\epsilon - \epsilon_\theta(\tilde{\mathbf{u}}_i, i, \mathbf{c})\|^2\right] \\ &= \mathbb{E}_{i\sim[1,N],\mathcal{S}\sim\mathbb{D},\epsilon\sim\mathcal{N}}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_i}\mathbf{u} + \sqrt{1-\bar{\alpha}_i}\epsilon, i, \mathbf{c})\|^2\right].\end{aligned} \tag{S25}$$

The equivalence between the DDPM loss function (Eq. (S25)) and score-function loss (Eq. (S13)) has also shown in Section 3.2 [37] with:

$$\nabla_{\tilde{\mathbf{u}}_i} p_i(\tilde{\mathbf{u}}_i|\mathbf{c}) = -\frac{1}{\sqrt{1-\bar{\alpha}_i}}\epsilon, \quad \mathbf{s}(\tilde{\mathbf{u}}_i|\mathbf{c}, i) = -\frac{1}{\sqrt{1-\bar{\alpha}_i}}\epsilon_\theta(\tilde{\mathbf{u}}_i, \mathbf{c}, i). \tag{S26}$$