VBD: Versatile Behavior Diffusion Model for Smart Agent Simulation

Zhiyu Huang[1], Zixu Zhang[2], Jaime Fernández Fisac[2], Chen Lv[1]
[1]Nanyang Technological University, Singapore
[2]Princeton University
zhiyu001@e.ntu.edu.sg, {zixuz, jfisac}@princeton.edu, lyuchen@ntu.edu.sg

Abstract

Generating realistic and controllable agent behaviors in traffic simulation is crucial for the development of autonomous vehicles. In this report , we introduce a novel framework ***Versatile Behavior Diffusion*** *(VBD) to simulate interactive scenarios with multiple traffic participants. The VBD model consists of three main parts: a query-centric Transformer encoder, a Transformer denoiser to generate scene-level joint behaviors of agents using the diffusion process, and a Transformer-based multi-modal marginal trajectory predictor. Our model not only generates scene-consistent multi-agent interactions but also enables scenario editing. Experimental evaluations indicate that the VBD model achieves competitive performance on the Waymo 2024 Sim Agents benchmark. Project website:* https://sites.google.com/view/versatile-behavior-diffusion

1. Introduction

Simulation plays a crucial role in validating the performance of autonomous driving systems. One primary challenge is generating diverse, realistic, and interactive traffic behaviors at scale. Conventional model-based methods often fail to deliver the necessary realism and accurately capture human interactions within real-world contexts. Consequently, many studies have shifted towards data-driven methods, leveraging readily available driving logs and applying imitation learning techniques to model more realistic behaviors for traffic agents. However, these methods typically focus on a single agent or use a shared policy across all traffic participants, resulting in a lack of scene consistency and increasing the likelihood of collisions in highly interactive scenarios.

We aim to leverage diffusion models [2, 7, 8], 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 have been increasingly employed in generating agent behaviors [4, 9]. However, training and controlling diffusion models for traffic agent interaction modeling remain challenging. In this paper, we propose **Versatile Behavior Diffusion** (VBD), which utilizes maps and historical states of agents as conditional inputs to generate realistic and scene-consistent traffic scenarios.

We demonstrated that our model is capable of directly generating scenarios via denoising, controllable sampling for various tasks with user-specific objectives, improving generation quality by integrating behavior priors, and generating long-tail safety-critical scenarios using game-theoretic guidance. In this report, we focus on directly simulating traffic agents' behaviors in a closed-loop environment without guidance, as our submission to the 2024 Waymo Open Dataset SimAgent Challenge. Detailed descriptions of our model and additional applications can be found in our extended paper.

2. Method

2.1. Overview

The VBD model consists of three main components as illustrated in Fig. 1. 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 [6, 10]. Leveraging rich scene context information from encoder, the denoiser $\mathcal{D}_\theta : (\hat{\mathbf{c}}, \tilde{\mathbf{u}}, k) \mapsto \hat{\mathbf{u}}$ directly predicts a joint control sequence $\hat{\mathbf{u}}$ from $\hat{\mathbf{c}}$ and noised control $\tilde{\mathbf{u}}$ at step k. The behavior predictor $\mathcal{P}_\psi : (\hat{\mathbf{c}}, \{\zeta^i\}_{i=1}^M) \mapsto \{\text{Cat}_M(\hat{\mathbf{u}}^a, \hat{\omega}^a)\}_{a=1}^A$ predicts an M-mode *marginal* categorical trajectory distribution of *each agent* from $\hat{\mathbf{c}}$ with the help of a set of representative static end-point anchors $\{\zeta^i\}_{i=1}^M$ extracted from data [6]. All three modules utilize a stack of query-centric self-attention and cross-attention blocks for flexibility and scalability.

2.2. 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 dy-

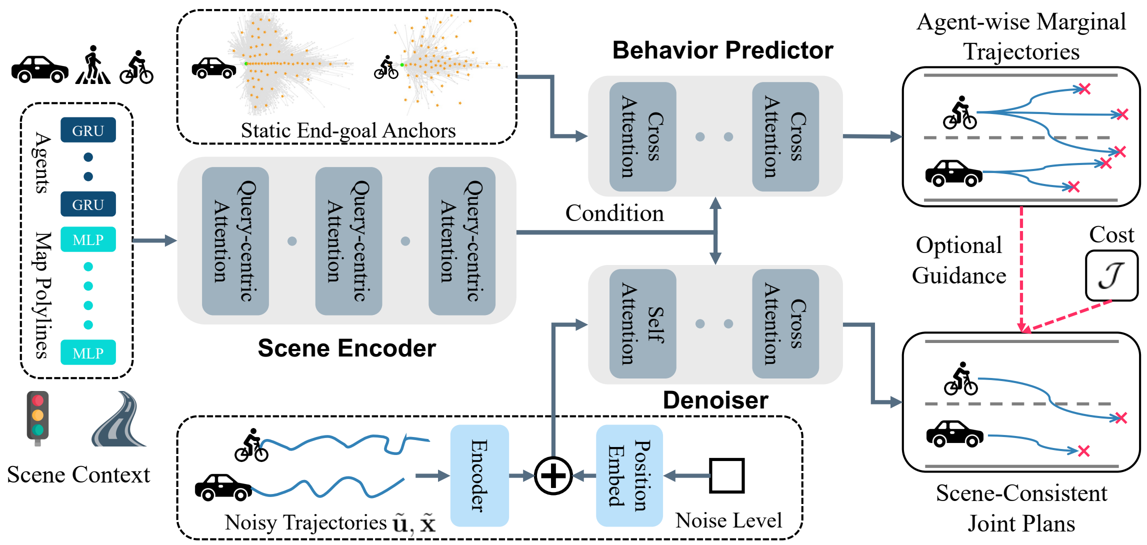

Figure 1. Overview of the proposed VBD model. The input scenario tokens are encoded through a query-centric Transformer-based scenario encoder. The behavior predictor generates marginal multi-modal trajectories. The denoiser predicts the joint multi-agent future trajectories while attending to themselves 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.

namics 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:

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

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:

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

2.3. 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 $A = 64$ 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 [6, 10]. 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 e^{ij} with a MLP. For each query element $\mathbf{q}^i$ operates, the query-centric transformer adapted a modified multi-head attention layer [5] as follows:

$$QCA(Q^i, K, V, \mathbf{e}) = \text{softmax}\Big(\frac{\mathbf{q}^i}{\sqrt{D}}[\{\mathbf{k}^j + \mathbf{e}^{ij}\}_{j\in\Omega(j)}]^T\Big)(\{\mathbf{v}^j + \mathbf{e}^{ij}\}_{j\in\Omega(j)}), \tag{3}$$

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. Our encoder consists of 6 post-norm query-centric Transformer layers with GELU activations to process the initial scene encoding. Each transformer layer has 8 attention heads and the hidden embedding dimension of $D = 256$. 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 [3] 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 4 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 $[A, Mo]$.

2.4. Model Training

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

$$\mathcal{L}_{\mathcal{D}_\theta} = \mathbb{E}_{\mathcal{S}\sim p, k\sim\mathcal{U}(0,K)}\mathbb{E}_{\tilde{\mathbf{u}}\sim p_k(\cdot|\mathbf{u})}\left[\lambda(k)\mathcal{SL}_1(\hat{\mathbf{x}}(\mathcal{D}_\theta(\hat{\mathbf{c}}, \tilde{\mathbf{u}}, k)) - \mathbf{x})\right], \tag{4}$$

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

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

Algorithm 1 Training of denoiser

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

of the denoiser is described in Algorithm 1.

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:

$$\mathcal{L}_{\mathcal{P}_\psi} = \mathbb{E}_{\mathcal{S}\sim p}\left[\sum_{a=1}^{A} \mathcal{SL}_1\left(\hat{\mathbf{x}}(\hat{\mathbf{u}}^{a,m^*}) - \mathbf{x}^a\right) + \beta CE(m^*, \hat{\omega}^a)\right], \tag{5}$$

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. To determine the best-predicted indices for an agent, the following criterion is applied:

$$m^* = \begin{cases} \arg\min_i ||ac^i - x_T||, & \text{if } x_T \text{ is valid,} \\ \arg\min_i || \sum_t (\hat{x}_t^i - x_t)||, & \text{otherwise,} \end{cases} \tag{6}$$

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.

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

$$\mathcal{L} = \mathcal{L}_{\mathcal{D}_\theta} + \gamma \mathcal{L}_{\mathcal{P}_\psi}, \tag{7}$$

where γ is a hyperparameter to balance the importance of tasks. The hyperparameters used in the total loss function are $\gamma = 0.5$, and $\beta = 0.05$ is used in the predictor loss.

A cosine variance schedule is adopted in the diffusion process, employing $K = 5$ 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 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 4 NVIDIA A100 GPUs, with a batch size of 14 per GPU.

2.5. Implementation details

The model was trained and validated on the Waymo Open Motion Dataset (WOMD) and we employ the Waymax simulator [1] as the interface for closed-loop traffic simulation. To improve closed-loop roll-out performance and stability, agent behaviors are replanned with a frequency of 1 Hz.

We evaluate the simulation performance of the multi-agent diffusion policy. We execute the VBD model's output in the Waymax simulator, and only let the policy/model control 64 agents near the self-driving car in the scene, while the rest of the agents follow a constant velocity policy. At each replanning step, we sample from Gaussian noise to initialize the diffusion generation process. The simulation horizon is 8 seconds, requiring 8 replanning steps to generate a scenario. We generate a total of 32 scenarios from the same initial state. The detailed steps of scenario generation can be found in Algorithm 2.

3. Results

Tab. 1 presents the quantitative results in comparison to other methods on the Waymo 2024 Sim Agents Benchmark. The proposed VBD model (diffusion policy) demonstrates strong performance across all metrics, particularly the minADE metric, which assesses proximity to human driving trajectories. With only 12M parameters and fewer diffusion steps, our model enables faster runtime (<100 ms per step), facilitating traffic simulation in practice. Fig. 2 showcases some qualitative results of the VBD model, highlighting its ability to generate interactive behaviors among agents within complex scenarios.scene-level

4. Conclusions

We introduce the **VBD** model for smart agent behavior simulation. The VBD model utilizes query-centric attention Transformers for scene encoding, capturing both local and relative information to enhance performance in multi-agent prediction. The Transformer-based denoiser head creates a diffusion-based multi-agent behavior policy to better model joint agent behaviors and scene-level interactions. Additionally, the model includes a marginal behavior prediction head to stabilize training and provide priors

Table 1. Testing performance on the Waymo Sim Agents Leaderboard

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

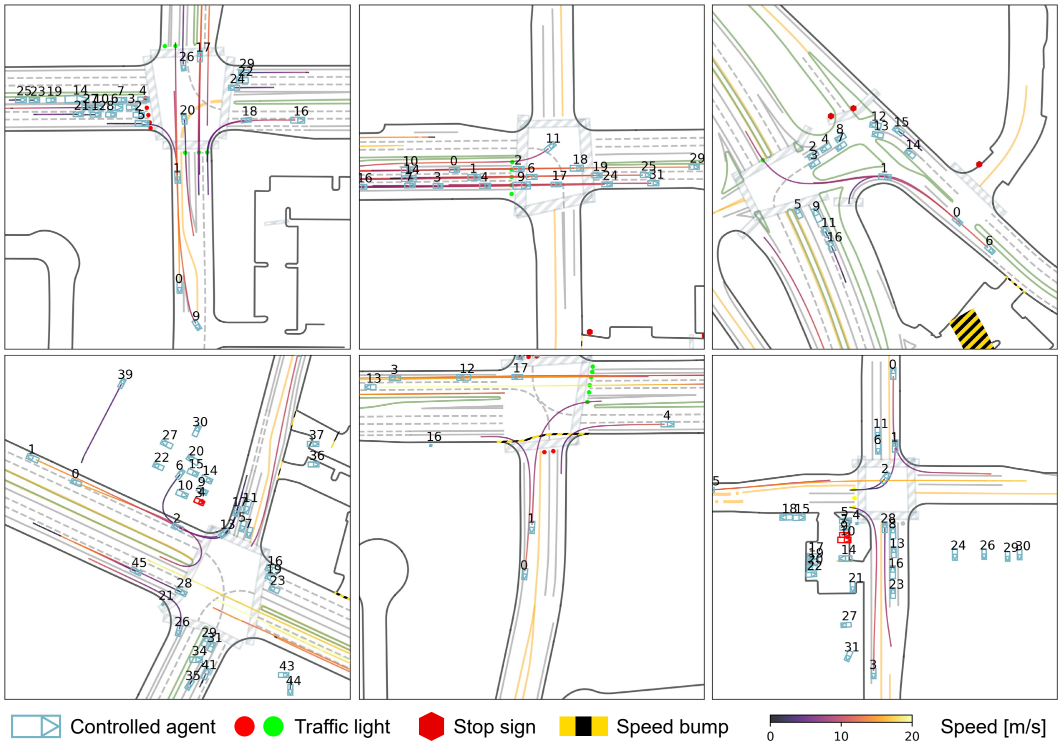

Figure 2. Performance of our VBD model on the Waymo Sim Agents task. The multi-agent diffusion policy is capable of controlling a large number of agents in an interactive and map-adherent manner for traffic simulation.

Algorithm 2 Generating Scenario

```
Require: Encoder $\mathcal{E}_\phi$, Denoiser $\mathcal{D}_\theta$, Initial Observation $\mathbf{c}_0$,
    Denoising Steps $K$
1: for $t = 0, \ldots, 8$ do
2:     $\hat{\mathbf{c}}_t = \mathcal{E}_\phi(\mathbf{c}_t)$                    ▷ Encode Scene Context
3:     $\tilde{\mathbf{u}}_K \sim \mathcal{N}(0, \mathbf{I})$                  ▷ Sample Random Noise
4:     for k=K...1 do
5:         $\tilde{\mathbf{u}}_{k-1} = \mathcal{D}_\theta(\tilde{\mathbf{u}}_k, \hat{\mathbf{c}}_t, k)$          ▷ Denoise
6:     end for
7:     $\mathbf{c}_{t+1} = \text{Step}(\mathbf{c}_t, \tilde{\mathbf{u}}_0^{0:1})$      ▷ Step Waymax with the
    first second control sequence
8: end for
```

for scene editing and controllable generation. Our model is trained on the large-scale Waymo Open Motion Dataset and achieves competitive closed-loop simulation performance on the Waymo 2024 Sim Agents Benchmark.

References

[1] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36, 2024. 4

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[3] Zhiyu Huang, Peter Karkus, Boris Ivanovic, Yuxiao Chen, Marco Pavone, and Chen Lv. DTPP: Differentiable

joint conditional prediction and cost evaluation for tree policy planning in autonomous driving. *arXiv preprint arXiv:2310.05885*, 2023. 3

[4] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023. 1

[5] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, 2018. 3

[6] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1, 3

[7] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 1

[8] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 1

[9] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3560–3566. IEEE, 2023. 1

[10] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023. 1, 3