SceneDiffuser: Efficient and Controllable Driving Simulation Initialization and Rollout

Chiyu Max Jiang   Yijing Bai* Andre Cornman*   Christopher Davis*   Xiukun Huang*
Hong Jeon*   Sakshum Kulshrestha*   John Lambert*   Shuangyu Li*   Xuanyu Zhou*
Carlos Fuertes   Chang Yuan   Mingxing Tan   Yin Zhou   Dragomir Anguelov

Waymo LLC

Abstract

Realistic and interactive scene simulation is a key prerequisite for autonomous vehicle (AV) development. In this work, we present SceneDiffuser, a scene-level diffusion prior designed for traffic simulation. It offers a unified framework that addresses two key stages of simulation: scene initialization, which involves generating initial traffic layouts, and scene rollout, which encompasses the closed-loop simulation of agent behaviors. While diffusion models have been proven effective in learning realistic and multimodal agent distributions, several challenges remain, including controllability, maintaining realism in closed-loop simulations, and ensuring inference efficiency. To address these issues, we introduce amortized diffusion for simulation. This novel diffusion denoising paradigm amortizes the computational cost of denoising over future simulation steps, significantly reducing the cost per rollout step (16x less inference steps) while also mitigating closed-loop errors. We further enhance controllability through the introduction of generalized hard constraints, a simple yet effective inference-time constraint mechanism, as well as language-based constrained scene generation via few-shot prompting of a large language model (LLM). Our investigations into model scaling reveal that increased computational resources significantly improve overall simulation realism. We demonstrate the effectiveness of our approach on the Waymo Open Sim Agents Challenge, achieving top open-loop performance and the best closed-loop performance among diffusion models.

1 Introduction

Simulation environments allow efficient and safe evaluation of autonomous driving systems [1, 8, 15, 22, 31, 32, 46, 50–52, 54]. Simulation involves initialization (determining starting conditions for agents) and rollout (simulating agent behavior over time), typically treated as separate problems [44]. Inspired by diffusion models' success in generative media, such as video generation [2, 10] and video editing (inpainting [21, 24, 28], extension, uncropping etc.), we propose SceneDiffuser, a unified spatiotemporal diffusion model that addresses both initialization and rollout for autonomous driving, trained end-to-end on logged driving scenes. To our knowledge, SceneDiffuser is the first model to jointly enable scene generation, controllable editing, and efficient learned closed-loop rollout (Fig. 1).

One challenge in simulation is evaluating long-tail safety-critical scenarios [1, 8, 22, 32, 46]. While data mining can help, such scenarios are often rare. We address this by learning a generative scene realism prior that allows editing logged scenes or generating diverse scenarios. Our model supports scene perturbation (modifying a scene while retaining similarity) and agent injection (adding agents to create challenging scenarios). We also enable synthetic scene generation on roadgraphs with realistic

---

*Equal contribution core technical contributors (alphabetically ordered).

Scene Initialization (Generation and Editing)

| Log Scenario | Log Perturbation | Agent Injection | Fully Synthetic Scenario |
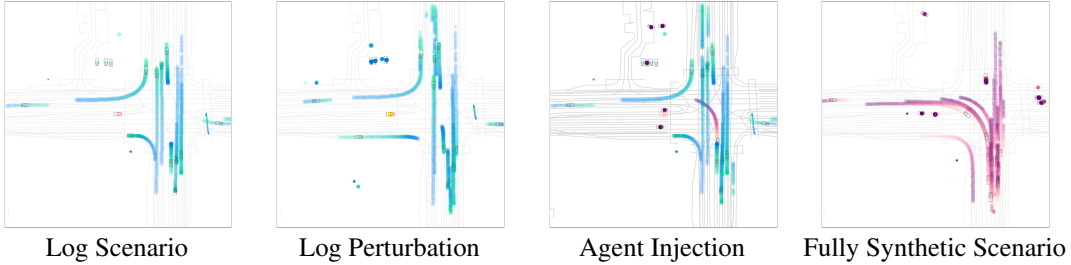
Scene Rollout (Closed-loop with Amortized Diffusion)

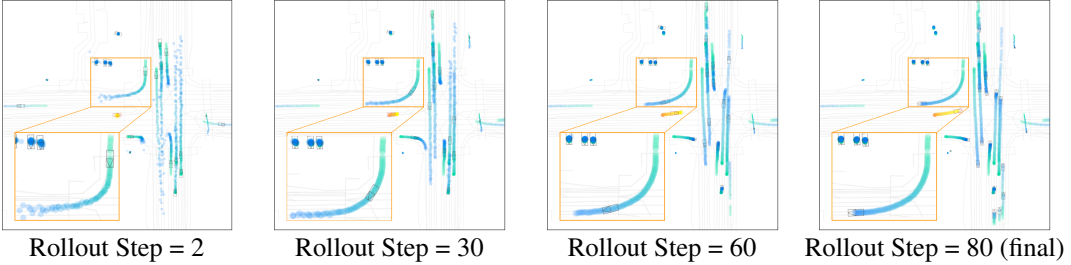| Rollout Step = 2 | Rollout Step = 30 | Rollout Step = 60 | Rollout Step = 80 (final) |

Figure 1: SceneDiffuser: a generative prior for simulation initialization via log perturbation, agent injection, and synthetic scene generation, and for efficient closed-loop simulation at 10Hz via amortized diffusion. It progressively refines initial trajectories throughout the rollout. Environment sim agents are in green-blue gradient (temporal progression), AV agent in orange-yellow, and synthetic agents in red-purple.

layouts. We design a protocol for specifying scenario constraints, enabling scalable generation, and demonstrate how a few-shot prompted LLM can generate constraints from natural language.

Given a scene, realistically simulating agents and AV behavior is challenging [15, 31, 50–52, 54]. Unlike motion prediction tasks [18, 25, 26, 38, 41, 48] where entire future trajectories are jointly predicted in a single inference, simulator predictions are iteratively fed back into the model, requiring realism at each step. This poses challenges: distributional drift from compounding errors, high computational cost for models like diffusion, and the need to simulate various perception attributes realistically.

We propose Amortized Diffusion for simulation rollout generation, a novel approach for amortizing the cost of the denoising inference over a span of physical steps that effectively addresses the challenges of simulation realism due to closed-loop drift and inference efficiency. Amortized diffusion iteratively carries over prior predictions and refines them over the course of future physical steps (see Sec. 3.2 and Fig. 4). This allows our model to produce stable, consistent, and realistic simulated trajectories, while requiring only a *single* denoising function evaluation at each physical step while jointly simulating all perception attributes at each step. Experiments show that Amortized Diffusion not only requires 16x less model inferences per step, but is also significantly more realistic.

In summary, SceneDiffuser's main contributions are:

- A unified generative model for scene initialization and rollout, jointly learning distributions for agents, timesteps, and perception features including pose, size and type.

- A novel amortized diffusion method for efficient and realistic rollout generation, significantly improving trajectory consistency and reducing closed-loop error.

- Controllable scene initialization methods, including log perturbation, agent injection, and synthetic generation with a novel hard constraint framework and LLM.

- Investigation of model scaling, showing increased compute effectively improves realism.

- Demonstration of effectiveness on the Waymo Open Sim Agents Challenge, achieving top open-loop performance and the best closed-loop performance among diffusion models.
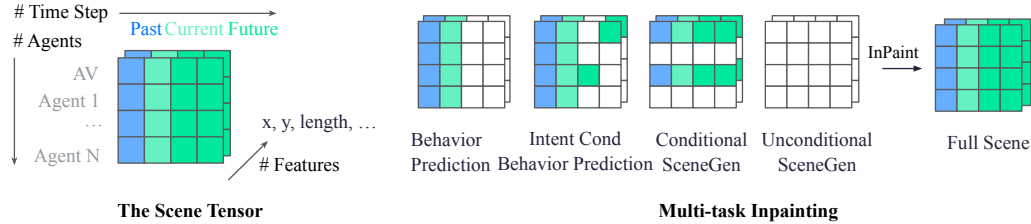
Figure 2: We formulate various different tasks, including behavior prediction, conditional scenegen and unconditional scenegen as inpainting tasks on the scene tensor. We represent the scene tensor as a normalized tensor $x \in \mathbb{R}^{A \times \mathcal{T} \times D}$, for the number of agents, timesteps and feature dimensions.

2   Related Work

2.1   Data-driven Agent Simulation

A variety of generative models have been explored for scene initialization and simulation, including autoregressive models [8, 22, 46], cVAEs [45], cGANs [1], and Gaussian Mixture Models (GMMs) [8, 47]. For closed-loop rollouts, these models have been extended with GMMs [51], GANs [15], AR models over discrete motion vocabularies [31], cVAE [54], and deterministic policies [50, 52]. Open-loop rollouts have also been explored using cVAE [35].

2.2   Diffusion Models for Agent Simulation

Open-loop Sim Open-loop simulation generates behavior for agents that all lie within one's control, i.e. does not receive any external inputs between steps. Open-loop simulation thus cannot respond to an external planner stack (AV), the evaluation of which is the purpose of simulation. Diffusion models have recently gained traction in multi-agent simulation, particularly in open-loop scenarios (multi-agent trajectory forecasting) [31, 39], using either single-shot or autoregressive (AR) generation. Single-shot approaches employ spatiotemporal transformers in ego-centric [6, 18] or scene-centric frames with motion/velocity deltas [9, 53]. Soft guidance techniques enhance controllability [17, 56]. DJINN [27] uses 2d condition masks for flexible generation.

Closed-loop Sim Closed-loop simulation with diffusion remains challenging due to compounding errors and efficiency concerns. Chang *et al.* [3] explore route and collision avoidance guidance in closed-loop diffusion, while VBD [14] combines denoising and behavior prediction losses with a query-centric Transformer encoder [42]. VBD found it computationally infeasible to replan at a 1Hz frequency in a receding horizon fashion over the full WOSAC test split due to the high diffusion inference cost, therefore testing in open-loop except over 500 selected scenarios.

Initial Condition Generation Diffusion-based initial condition generation has also been studied [20]. Pronovost *et al.* [32, 33] adapt the LDM framework to rendered scene images, while SLEDGE [5] and DriveSceneGen [44] diffuse initial lane polylines, agent box locations, and AV velocity.

2.3   Diffusion for Temporal World Modeling and Planning

Outside of the autonomous driving domain, diffusion models have proven effective for world simulation through video and for planning. Various diffusion models for 4d data have been proposed, often involving spatiotemporal convolutions and attention mechanisms [11, 12, 43]. In robotics, diffusion-based temporal models leverage Model Predictive Control (MPC) for closed-loop control [4] and have shown state-of-the-art performance for imitation learning [29].

Similar to our Amortized Diffusion approach, TEDi [55] proposes to entangle the physical timestep and diffusion steps for human animation, thereby reducing $O(T \cdot \mathcal{T})$ complexity for $\mathcal{T}$ physical timesteps and T denoising steps to $O(\mathcal{T})$. However, we are the first work to demonstrate the effectiveness of this approach for reducing closed-loop simulation errors, and the first to extend it to a multi-agent simulation setting.
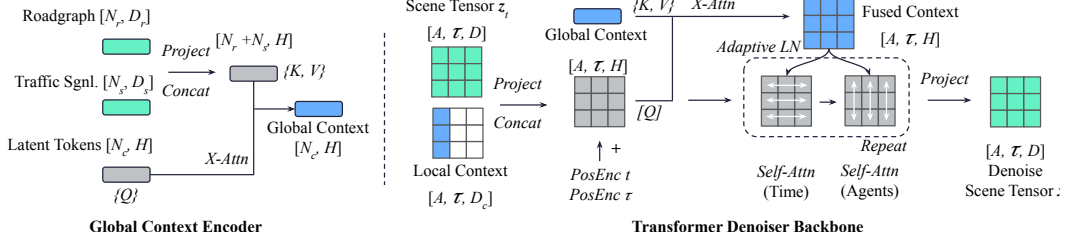
Figure 3: SceneDiffuser architecture. Global scene context is encoded into a fixed number of N_c tokens via a Perceiver IO [16] encoder. The noisy scene tokens are fused with local and global context, then used to condition a spatiotemporal transformer-based backbone [49] via Adaptive LayerNorm (AdaLN) [30]. Input/output tensor are in green, context tensors in blue, and ops in *italics*.

3 Method

3.1 Scene Diffusion Setup

We denote the scene tensor as $\boldsymbol{x} \in \mathbb{R}^{A \times \mathcal{T} \times D}$, where A is the number of agents jointly modeled in the scene, $\mathcal{T}$ is the total number of modeled physical timesteps, and D is the dimensionality of all the features that are jointly modeled. We learn to predict the following attributes for each agent: positional coordinates x, y, z, heading γ, bounding box dimensions l, h, w, and object type $k \sim \{AV, car, pedestrian, cyclist\}$. We model all tasks considered in SceneDiffuser as multi-task inpainting on this scene tensor. Given an inpainting mask $\bar{\boldsymbol{m}} \in \mathbb{B}^{A \times \mathcal{T} \times D}$, the corresponding inpainting context values $\bar{\boldsymbol{x}} := \bar{\boldsymbol{m}} \odot \boldsymbol{x}$, a set of global context $\boldsymbol{c}$ (such as roadgraph and traffic signals), and a validity mask for a given agent at a given timestep $\bar{\boldsymbol{v}} \in \mathbb{B}^{A, \mathcal{T}}$ (to account for there being $< A$ agents in the scene or for occlusion), we train a diffusion model to learn the conditional probability $p(\boldsymbol{x}|\mathcal{C})$, where $\mathcal{C} := \{\bar{\boldsymbol{m}}, \bar{\boldsymbol{x}}, \boldsymbol{c}, \bar{\boldsymbol{v}}\}$. See Fig. 2 for an illustration of the scene tensor.

Feature Normalization To simplify the diffusion model's learning task, we normalize all feature channels before concatenating them along D to form the scene tensor. We first encode the entire scene in a scene-centric coordinate system, namely the AV's coordinate frame just before the simulation commences. We then scale x, y, z by fixed constants, l, h, w by their standard deviation, and one-hot encode k. See Appendix A.6 for more details. This simple yet generalizable process allows us to jointly predict float, boolean, and even categorical attributes by converting into a normalized space of floats. After generating a scene tensor $\boldsymbol{x}$, we apply a reverse process to obtain the generated features.

Diffusion Preliminaries We adopt the notation and setup for diffusion models from [13]. The forward diffusion process gradually adds Gaussian noise to $\boldsymbol{x}$. The noisy scene tensor at diffusion step t can be expressed as $\mathbf{q}(\boldsymbol{z}_t|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_t|\alpha_t \boldsymbol{x}, \sigma_t^2 \boldsymbol{I})$, where α_t and σ_t are parameters which control the magnitude and variances of the noise schedule under a variance-preserving model. Therefore $\boldsymbol{z}_t = \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \boldsymbol{I})$. One major departure from the classic diffusion setup in our amortized diffusion regime is that we do not assume a uniform noise level $t \in \mathbb{R}$ for the entire scene tensor $\boldsymbol{x}$. Instead, we have $t \in \mathbb{R}^{\mathcal{T}}$ where t can be relaxed to have a different value per physical timestep in the scene tensor as described in Sec. 3.2. We utilize the commonly used α-cosine schedule where $\alpha_t = \cos(\pi t/2)$ and $\sigma_t = \sin(\pi t/2)$. At the highest noise level of $t = 1$, the forward diffusion process completely destroys the initial scene tensor $\boldsymbol{x}$ resulting in $\boldsymbol{z}_t = \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \boldsymbol{I})$. Assuming a Markovian transition process, we have the transition distributions $q(\boldsymbol{z}_t|\boldsymbol{z}_s) = \mathcal{N}(\boldsymbol{z}_t|\alpha_{ts}\boldsymbol{z}_s, \sigma_{ts}^2 \boldsymbol{I})$, where $\alpha_{ts} = \alpha_t/\alpha_s$ and $\sigma_{ts}^2 = \sigma_t^2 - \alpha_{ts}^2 \sigma_s^2$ and $t > s$. In the denoising process, conditioned on a single datapoint $\boldsymbol{x}$, the denoising process can be written as

$$q(\boldsymbol{z}_s|\boldsymbol{z}_t, \boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_{t \to s}, \sigma_{t \to s}^2 \boldsymbol{I}), \tag{1}$$

where $\boldsymbol{\mu}_{t \to s} = \frac{\alpha_{ts}\sigma_s^2}{\sigma_t^2}\boldsymbol{z}_t + \frac{\alpha_s \sigma_{ts}^2}{\sigma_t^2}\boldsymbol{x}$ and $\sigma_{t \to s} = \frac{\sigma_{ts}^2 \sigma_s^2}{\sigma_t^2}$. In the denoising process, $\boldsymbol{x}$ is approximated using a learned denoiser $\hat{\boldsymbol{x}}$. Following [13] and [37], we adopt the commonly used *v prediction*, defined as $\boldsymbol{v}_t(\boldsymbol{\epsilon}_t, \boldsymbol{x}) = \alpha_t \boldsymbol{\epsilon}_t - \sigma_t \boldsymbol{x}$. We trained a model parameterized by $\boldsymbol{\theta}$ to predict $\boldsymbol{v}_t$ given $\boldsymbol{z}_t$, t and context $\mathcal{C}$: $\hat{\boldsymbol{v}}_t := \hat{\boldsymbol{v}}_\theta(\boldsymbol{z}_t, t, \mathcal{C})$. The predicted $\hat{\boldsymbol{x}}_t$ can be recovered via $\hat{\boldsymbol{x}}_t = \alpha_t \boldsymbol{z}_t - \sigma_t \hat{\boldsymbol{v}}_t$. The model is end-to-end trained with a single loss:

$$\mathbb{E}_{(\boldsymbol{x}, \mathcal{C}) \sim \mathcal{D}, t \sim \{\mathcal{U}(0,1); \hat{t}\}, \boldsymbol{m} \sim \mathcal{M}, \epsilon_t \sim \mathcal{N}(0, \boldsymbol{I})}[||\hat{\boldsymbol{v}}_\theta(\boldsymbol{z}_t, t, \mathcal{C}) - \boldsymbol{v}_t(\boldsymbol{\epsilon}_t, \boldsymbol{x})||_2^2], \tag{2}$$
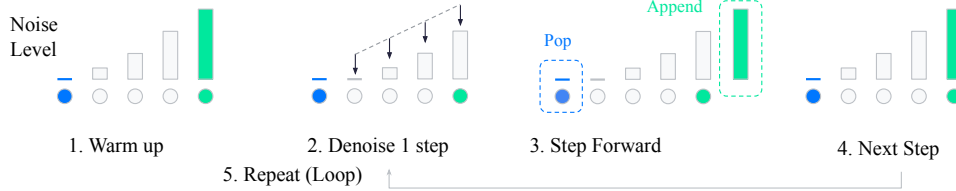
4

Figure 4: Amortized diffusion rollout procedure. The warm up step initializes the future predictions for the entire future horizon, which is then perturbed by a monotonic noise schedule $\hat{t}$. The trajectory is iteratively denoised by one step at each simulation step.

$\mathcal{D} = \{(\boldsymbol{x}_i, \mathcal{C}_i) | i = 1, 2, \cdots, |\mathcal{D}|\}$ is the dataset containing paired agents and scene context data, t is probabilistically either sampled from a uniform distribution, or sampled as a monotonically increasing temporal schedule $\hat{\boldsymbol{t}}$, where $\hat{\boldsymbol{t}}_\tau = \max\left(0, (\tau - \mathcal{T}_{\text{history}})/\mathcal{T}_{\text{future}}\right)$ to facilitate amortized rollout which will be discussed in Sec. 3.2. Each is sampled with 50% probability. $\mathcal{M} = \{\bar{\boldsymbol{m}}_{\text{bp}} \odot \bar{\boldsymbol{m}}_{\text{control}}, \bar{\boldsymbol{m}}_{\text{scenegen}} \odot \bar{\boldsymbol{m}}_{\text{control}}\}$ is the set of inpainting masks for the varied tasks.

Scene Diffusion Tasks Different tasks are fomulated as inpainting problems (Fig. 2).

Scene Generation (SceneGen): Given the full trajectory of some agents, generate the full trajectory of other agents. We have $\bar{\boldsymbol{m}}_{\text{scenegen}} \in \mathbb{R}^{A,1,1}$ (broadcastable to $\mathcal{T}$ timesteps and D features), where $\bar{\boldsymbol{m}}_{\text{scenegen, a}} \sim Pr(X = A_{\text{select}}/A_{\text{valid}})$, where $A_{\text{select}} \sim \mathcal{U}(0, A_{\text{valid}})$ is the number of agents sampled to be selected as inpainting conditions out of A_{valid} valid agents in the scene.

Behavior Prediction (BP): Given past and current data for all agents, predict the future for all agents. We have $\bar{\boldsymbol{m}}_{\text{bp}} \in \mathbb{R}^{1,\mathcal{T},1}$ (broadcastable to A agents and D features), where $\bar{\boldsymbol{m}}_{\text{bp},\tau} = \mathcal{I}(\tau < \mathcal{T}_{history})$.

Conditional SceneGen and Behavior Prediction: Both scenegen and behavior prediction masks are multiplied by a control mask at training time to enable controllable scenegen and controllable behavior prediction at inference time. We have $\bar{\boldsymbol{m}}_{\text{control}} \in \mathbb{R}^{A,\mathcal{T},D}$, where $\bar{\boldsymbol{m}}_{\text{control},(a,\tau,d)} = I_a \cdot I_\tau \cdot I_d, I_a \sim Pr(X = A_{\text{control}}/A_{\text{valid}}), I_\tau \sim Pr(X = \mathcal{T}_{\text{control}}/\mathcal{T}), I_d \sim Pr(X = p_d)$ where p_d of the corresponding feature channel. This allows us to condition on certain channels, such as positions x, y with or without specifying other features such as type and heading.

Architecture We present a schematic for the SceneDiffuser architecture in Fig. 3, consisting of two end-to-end trained models: a global context encoder and a transformer denosier backbone. Validity $\bar{\boldsymbol{v}}$ is used as a transformer attention mask within the transformer denoiser backbone.

Diffusion Sampler We use DPM++ [19] with a Heun solver. We utilize 16 denoising steps for our one-shot experiments and for our amortized diffusion warmup process.

3.2 Scene Rollout

Future prediction with no replanning ('One-Shot') is not used in simulation due to its non-reactivity, and forward scene inference, under the standard diffusion paradigm ('Full AR'), is computationally intensive due to the double for-loop over both physical rollout steps and denoising diffusion steps [55]. Moreover, executing only the first step while discarding the remainder leads to inconsistent plans that result in compounding errors. We adopt an amortized autoregressive ('Amortized AR') rollout, aligning the diffusion steps with physical timesteps to amortize diffusion steps over physical time, requiring a single diffusion step at each simulation step while reusing previous plans.

We illustrate the three algorithms in Algorithm 1-3 using the same model trained with a noise mixture $t \sim \{\mathcal{U}(0,1); \hat{\boldsymbol{t}}\}$ (Eqn. 2). We also illustrate Algorithm 3 in Fig. 4. We denote the total number of timesteps $\mathcal{T} = H + F$, where H, F denote the number of past and future steps. We denote $\boldsymbol{x} := \boldsymbol{x}^{[-H:F]}$ to be the temporal slicing operator where $\boldsymbol{x}^{[0]}$ is the final history step.

Input: Global context $\mathbf{c}$ (roadgraph and traffic signals), history states $\boldsymbol{x}^{[-H:0]}$, validity $\bar{\boldsymbol{v}}$.

Output: Simulated observations for unobserved futures $\hat{\boldsymbol{x}}^{[1:F]}$.

---

[2] We omit L/2 due to training collapse.

Algorithm 1 One-Shot (Open-Loop)

1: $\text{OneShot}(\boldsymbol{x}^{[-H:0]}, \mathcal{C})$:
2: $\boldsymbol{z}_1 \sim \mathcal{N}(0, I)$
3: **for** $t = 1, \cdots, \Delta t, 0$ **do**   ▷ For each diffusion timestep
4:    $\hat{\boldsymbol{x}}_t \leftarrow \alpha_t \boldsymbol{z}_t - \sigma_t \hat{\boldsymbol{v}}_\theta(\boldsymbol{z}_t, t, \mathcal{C})$ ▷ V-prediction
5:    $\hat{\boldsymbol{x}}_t \leftarrow \boldsymbol{x}^{[-H:0]} \odot \bar{\boldsymbol{m}}_{\text{bp}} + \hat{\boldsymbol{x}}_t \odot (\sim \bar{\boldsymbol{m}}_{\text{bp}})$ ▷ Apply inpainting
6:    $\boldsymbol{z}_s \sim q(\boldsymbol{z}_s | \boldsymbol{z}_t, \hat{\boldsymbol{x}}_t)$
7:    $\boldsymbol{z}_t \leftarrow \boldsymbol{z}_s$
   $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{z}_0$
8: **return** $\hat{\boldsymbol{x}}^{[1:F]}$

Algorithm 2 Full AR (Closed-Loop)

1: $\text{FullAR}(\boldsymbol{x}^{[-H:0]}, \mathcal{C})$:
2: $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}^{[-H:0]}$
3: **for** $\tau = 0, ..., \mathcal{T} - 1$ **do**   ▷ For each physical timestep
4:    $\hat{\boldsymbol{x}}^{[\tau+1:\tau+F]} \leftarrow \text{OneShot}(\hat{\boldsymbol{x}}^{[\tau-H:\tau]}, \mathcal{C})$ ▷ Update buffer at indices
5: **return** $\hat{\boldsymbol{x}}^{[1:F]}$

Algorithm 3 Amortized AR (Closed-Loop)

1: $\text{AmortizedAR}((\boldsymbol{x}^{[-H:0]}, \mathcal{C}))$:
2: $\hat{\boldsymbol{x}} \leftarrow \text{OneShot}(\boldsymbol{x}^{[-H:0]}, \mathcal{C})$ ▷ Warm-Up
3: $\hat{\boldsymbol{x}}^{[0:F]} \leftarrow \alpha_{\hat{\boldsymbol{t}}} \hat{\boldsymbol{x}}^{[0:F]} + \sigma_{\hat{\boldsymbol{t}}} \boldsymbol{\epsilon}$ ▷ Add noise $\hat{\boldsymbol{t}}$
4: **for** $\tau = 1, ..., \mathcal{T}$ **do**   ▷ For each physical timestep
5:    $\hat{\boldsymbol{x}}^{[\tau:\tau+F]} \leftarrow \alpha_{\hat{\boldsymbol{t}}} \hat{\boldsymbol{x}}^{[\tau:\tau+F]} - \sigma_{\hat{\boldsymbol{t}}} \hat{\boldsymbol{v}}_\theta(\hat{\boldsymbol{x}}^{[\tau:\tau+F]}, \hat{\boldsymbol{t}}, \mathcal{C})$ ▷ Recover solution from v-prediction
6:    $\hat{\boldsymbol{x}}^{[\tau:\tau+F]} \leftarrow \alpha_{\hat{\boldsymbol{t}}} \hat{\boldsymbol{x}}^{[\tau:\tau+F]} + \sigma_{\hat{\boldsymbol{t}}} \boldsymbol{\epsilon}$ ▷ Add noise $\hat{\boldsymbol{t}}$
7:    $\hat{\boldsymbol{x}}^{[\tau-H:\tau+F]} \leftarrow \hat{\boldsymbol{x}}^{[\tau-H:\tau]} \odot \bar{\boldsymbol{m}}_{\text{bp}} + \hat{\boldsymbol{x}}^{[\tau-H:\tau+F]} \odot (\sim \bar{\boldsymbol{m}}_{\text{bp}})$ ▷ Apply inpainting
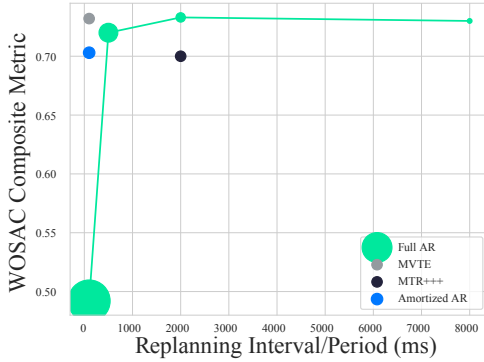8: **return** $\hat{\boldsymbol{x}}^{[1:F]}$

Figure 5: We compare the influence of replan rate on performance for our Full AR and Amortized AR models. Circle radius $\propto$ # inference calls over the simulation. At 10Hz, Amortized AR requires 16x less model inference per step and is more realistic compared to Full AR.
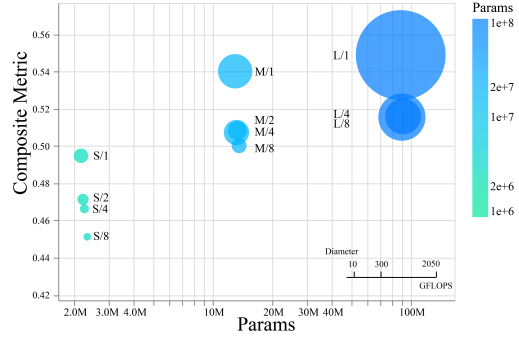
Figure 6: Scene generation realism with model parameter and resolution scaling[2]. Decreased temporal patch sizes (i.e. increased temporal resolution) and increased parameters are both effective for improving realism via compute scaling. Circle radius $\propto$ compute GFLOPs.

3.3 Controllable Scene Generation

To simulate long-tail scenarios such as rare behavior of other agents, it is important to effectively insert controls into the scene generation process. To do so, we input an inpainting context scene tensor $\bar{\boldsymbol{x}}$, where some pixels are pre-filled. Through pre-filled feature values in $\bar{\boldsymbol{x}}$, we can specify a particular agent of a specified type to be appear at a specific position at a specific timestamp.

Data Augmentation via Log Perturbation The diffusion framework makes it straightforward to produce additional perturbed examples of existing ground truth (log) scenes. Instead of starting from pure noise $\boldsymbol{z}_t \sim \mathcal{N}(0, \boldsymbol{I})$ and diffusing backwards from $t \to 0$, we take our original log scene $\boldsymbol{x}'$ and add noise to it such that our initial $\boldsymbol{z}_t = \alpha_t \boldsymbol{x}' + \boldsymbol{\epsilon}_t$ where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \sigma_t \boldsymbol{I})$. Starting the diffusion process at $t = 0$ yields the original data, while $t = 1$ produces purely synthetic data. For $t \in (0, 1)$, higher values increase diversity and decrease resemblance to the log. See Figs. 1 and 12 (Appendix).

Language-based Few-shot Scene Generation The diffusion model inpaint constraints can be defined through structured data such as a Protocol Buffer[3] ('proto'). Protos can be converted into inpainting values, and we leverage the off-the-shelf generalization capabilities of a publicly accessible

---

[3] https://protobuf.dev/

Full AR @ 0.125 HZ   Full AR @ 2 Hz

Full AR @ 10 Hz   Amortized AR @ 10 Hz

(a) No Hard Constraints

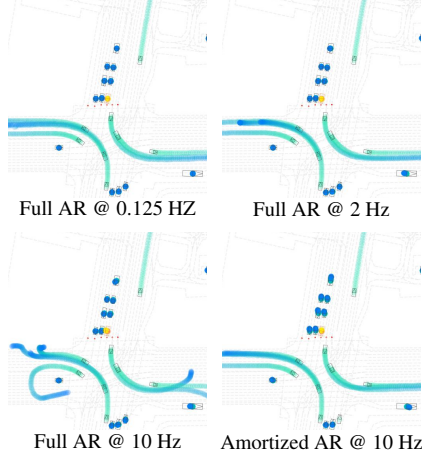(b) Post-Diffusion GHC   (c) In-Diffusion GHC

Figure 7: Full AR quality deteriorates at increasing replan rates due to compounding errors. Amortized AR retains a high level of realism even at 10 Hz while being more efficient.
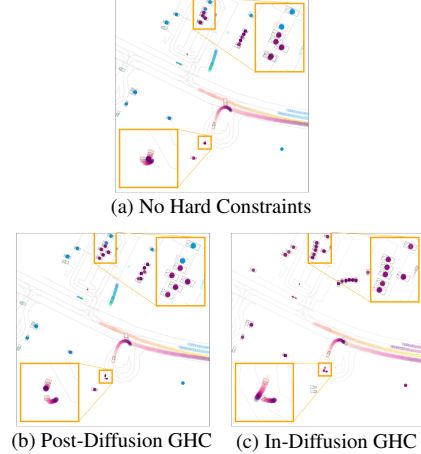
Figure 8: Applying no-collision constraints prevents collisions (red-purple) in generated scenes (b, c). Iteratively applying constraints with every diffusion step further enhances realism (c vs b).

chat app powered by a large language model (LLM)[4], to generate new Scene Diffusion constraints protos solely using natural language via few-shot prompt engineering. We show example results generated by the LLM in Fig. 10. Details in the Appendix (A.7).

3.4 Generalized Hard Constraints

Users of simulation often require agents to have specific behaviors while maintaining realistic trajectories. However, diffusion soft constraints [27, 56, 57] require a differentiable cost for the constraint and do not guarantee constraint satisfaction. Diffusion hard constraints [21] are modeled as inpainting values and are limited in their expressivity.

Inspired by dynamic thresholding [36] in the image generation domain, where intermediate images are dynamically clipped to a range at every denoising step, we introduce *generalized hard constraints* (GHC), where a generalized clipping function is iteratively applied at each denoising step. We modify Eqn. 1 such that at each denoising step $\boldsymbol{\mu}_{t \rightarrow s} = \frac{\alpha_{ts}\sigma_s^2}{\sigma_t^2}\boldsymbol{z} + \frac{\alpha_s\sigma_{ts}^2}{\sigma_t^2}\text{clip}(\boldsymbol{x})$, where $\text{clip}(\cdot)$ denotes the GHC-specific clipping operator. See more details on constraints in Appendix A.9.

We qualitatively demonstrate the effect of hard constraints for unconditional scene generation in Fig. 8. Applying hard constraints post-diffusion removes overlapping agents but results in unrealistic layouts, while applying the hard constraints after each diffusion step both removes the overlapping agents and takes advantage of the prior to improve the realism of the trajectories. We find that the basis on which the hard constraints operate is important: a good constraint will modify a significant fraction of the scene tensor (for example, shifting an agent's entire trajectory rather than just the overlapping waypoints), or else the model "rejects" the constraint on the next denoising step.

4 Experimental Results

Dataset We use the Waymo Open Motion Dataset (WOMD)[7] for both our scene generation and agent simulation experiments. WOMD includes tracks of all agents and corresponding vectorized maps in each scenario, and offers a large quantity of high-fidelity object behaviors and shapes produced by a state-of-the-art offboard perception system.

4.1 Simulation Rollout

Benchmark We evaluate our closed-loop simulation models on the Waymo Open Sim Agent Challenge (WOSAC) [23] metrics (see Appendix A.1), a popular sim agent benchmark used in many recent works [9, 14, 31, 51, 53]. Challenge submissions consist of x/y/z/γ trajectories representing centroid coordinates and heading of the objects' boxes that must be generated in closed-loop and

METRICS	M/1	M/1+GHC	L/1	LOG
COMPOSITE METRIC	0.516	**0.558**	0.549	0.593
LINEAR SPEED	0.326	0.327	**0.331**	0.339
LINEAR ACCEL.	**0.387**	0.383	0.378	0.445
ANGULAR SPEED	0.529	**0.562**	0.534	0.572
ANGULAR ACCEL.	0.595	**0.608**	0.588	0.625
DIST. TO OBJ.	0.154	**0.176**	0.174	0.192
COLLISION	0.692	**0.841**	0.794	0.875
TIME TO COLLISION	0.826	**0.827**	**0.827**	0.842
DIST. TO RD. EDGE	0.164	0.165	**0.176**	0.204
OFFROAD	0.546	0.549	**0.566**	0.605

Table 1: Realism metrics on WOMD *val.* for scene generation for the M/1 model, M/1 with hard constraints, L/1, and oracle log distribution matching (see Sec. 4.3). Realism can be improved through hard constraints or scaling.
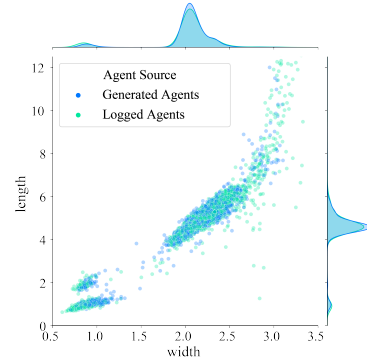
Figure 9: Generated vs logged distribution. SceneDiffuser learns realistic joint distributions across modeled features such as length and width.

	ONE-SHOT	FULL AR (10 Hz)	AMORTIZED AR (10 Hz)
COMPOSITE SCORE (M/1)	0.730	0.492	0.673
COMPOSITE SCORE (L/1)	0.736	-	0.703
# FN EVALS	16	$16 \cdot 80 = 1280$	$80 + 16 = 96$

	COMPOSITE METRIC ($\uparrow$)	COLLISION RATE ($\downarrow$)	OFFROAD RATE ($\downarrow$)
-ADALN-ZERO	-7.99%	+65.2%	+29.3%
-SPATIAL-ATTN	-14.5%	+209%	+11.8%
-MULTITASK	-2.04%	+39.6%	+3.24%
-SIZE,TYPE	0.68%	-6.85%	+2.90%

Table 2: Distrib. realism metrics on WOSAC. L/1 denotes the Large model of patch size 1.

Table 3: Design analysis and ablation studies.

with factorized AV vs. agent models. WOSAC uses the test data from the Waymo Open Motion Dataset (WOMD)[7]. Up to 128 agents (one of which must represent the AV) must be simulated in each scenario for the 8 second future (comprising 80 steps of simulation), producing 32 rollouts per scenario for evaluation. In a small departure from the official setting, we utilize the logged validity mask as input to our transformer and unify the AV and agents' rollout step for simplicity.

Evaluation In Tab. 2, we show results on WOSAC. We show that Amortized AR (10 Hz) not only requires 16x fewer model inference calls, but is also significantly more realistic than Full AR at a 10Hz replan rate. In Amortized AR, we re-use the plan from the previous step, leading to increased efficiency and consistency. The one-shot inference setting is equivalent to Full AR with no replanning (0.125 Hz) and achieves comparably higher realism, though as it is not executed in closed-loop, it is not reactive to external input in simulation, and thus not a valid WOSAC entry.

In Figs. 5 and 7, we investigate the effects of varied replan rates to simulation realism. While high replan frequency leads to significant degradation in realism under the Full AR rollout paradigm, Amortized AR significantly reduces error accumulation while being $16\times$ more efficient.

In Tab. 4, we compare against the WOSAC leaderboard with the aforementioned modifications. We achieve top open-loop performance and the best closed-loop performance among diffusion models.

4.2 Scene Generation

Unconstrained Scene Generation We use the unconditional scene generation task as a means to quantitatively measure the distributional realism of our model. We condition the scene using the same logged road graph and traffic signals, as well as the logged agent validity to control for the same number of agents generated per scene. All agent attributes are generated by the model.

Due to a lack of public benchmarks for this task, we adopt a slightly modified version of the WOSAC [23] metrics, where different metrics buckets are aggregated per-scene instead of per-agent, due to the lack of one-to-one correspondence between agents in the generated scene versus the logged scene (see Appendix A.2 for more details). Metrics are aggregated over all agents that are ever valid in the 9 second trajectory.

We show our model's realism metrics in Tab. 1. Even compared to the oracle performance (comparing logged versus logged distributions), our model achieves comparable realism scores in every realism bucket. Introducing hard constraints on collisions can significantly improve the composite metric

Agent Policy f	Replan Rate (Hz)	Linear Speed (↑)	Linear Accel. (↑)	Ang. Speed (↑)	Ang. Accel. (↑)	Dist. to Obj. (↑)	Collision (↑)	TTC (↑)	Dist. to Road Edge (↑)	Offroad (↑)	Composite Metric (↑)	ADE (↓)	minADE (↓)	Collision Rate (↓)	Offroad Rate (↓)
Random Agent	10	0.002	0.116	0.014	0.034	0.000	0.000	0.735	0.148	0.191	0.144	50.739	50.706	1.000	0.613
Const. Velocity	10	0.043	0.067	0.252	0.439	0.202	0.355	0.739	0.455	0.451	0.381	7.923	7.923	0.314	0.293
MTR+++ [34]	2	0.321	0.247	0.428	0.533	0.340	0.886	0.797	0.655	0.893	0.700	**2.125**	1.679	0.080	**0.135**
MVTE [51]	10	0.353	0.354	0.496	0.599	**0.392**	0.913	**0.833**	0.642	**0.907**	0.732	3.859	1.674	0.090	0.158
Trajeglish [31]	10	**0.356**	**0.399**	**0.509**	**0.654**	0.378	**0.925**	**0.834**	**0.660**	0.884	**0.735**	3.158	**1.615**	0.076	0.170
SceneDMF [9]	0.125	0.343	0.395	0.381	0.366	0.362	0.760	0.812	0.623	0.735	0.628	4.158	2.414	0.217	0.285
VBD [14]	0.125	0.359	0.366	0.420	0.522	0.368	0.934	0.815	0.651	0.879	0.720	2.257	1.474	0.036	0.152
Ours (One-Shot)	0.125	0.399	0.225	0.510	0.605	0.401	0.934	0.837	0.686	0.893	0.736	2.436	1.103	0.070	0.172
Ours (Amortized)	10	0.310	0.389	0.459	0.560	0.349	**0.917**	0.815	0.634	0.833	0.703	2.619	1.767	**0.064**	0.177
Logged Oracle	-	0.476	0.478	0.578	0.694	0.476	1.000	0.883	0.715	1.000	0.819	0.000	0.000	0.028	0.111

Table 4: Per-component WOSAC metric results on the *test* split of WOMD, representing likelihoods. Methods are ranked by composite metric on the 2024 Challenge scores; Closed-loop results within 1% of the best are in bold (for models with 10Hz replan). Diffusion-based methods marked in blue.

(M) Cut-In (M) Tailgater
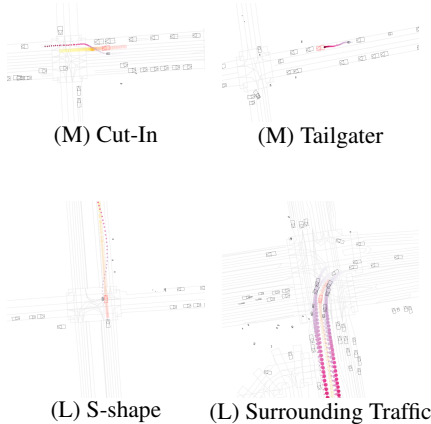
(L) S-shape (L) Surrounding Traffic

Figure 10: Long-tail synthetic scenes generated via control points either explicitly defined by a manually defined (M) or LLM-generated (L) config. Magenta indicates generated motorcyclists/car agents.
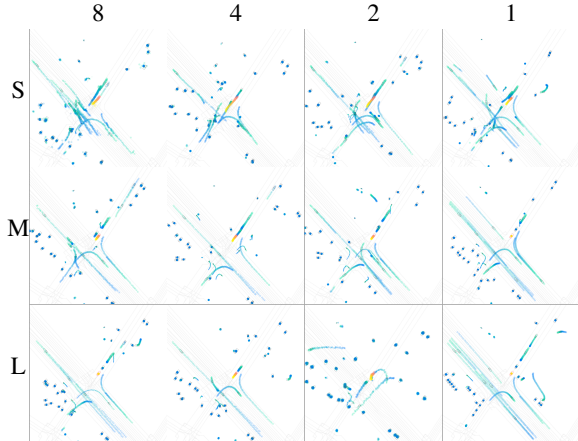

Figure 11: Fully synthetic scene generation quality comparison via scaling model parameters ($S \rightarrow L$) and increasing temporal resolution (patch size $8 \rightarrow 1$). Increasing compute by scaling either the model size or temporal resolution improves the overall realism.

by preventing collisions, while scaling the model without hard constraints improves most realism metrics as the model learns to generate more realistic trajectories. The realism metrics only apply to trajectories and do not account for generated agent type and size distributions. We compare the generated size distributions versus log distributions in Fig. 9 and find the marginal and joint distributions both closely track the logged distribution. We show more examples of diverse, unconstrained scene generation when conditioning on the same global context in Appendix A.8 Fig. 13.

Constrained Scene Generation and Augmentation The controllability we possess in the scene generation process as a product of our diffusion model design can be useful for targeted generation and augmentation of scenes. In Fig. 10, we show qualitative results of scenes with constrained agents generated either via manually defined configs or by a few-shot prompted LLM. Extended qualitative results are listed in Appendix A.7.3.

4.3 Model Design Analysis and Ablation Studies

Scaling Analysis Given two options of scaling model compute, either by increasing transformer temporal resolution by decreasing temporal patch sizes, or increasing the number of model parameters, we investigate the performance of multiple transformer backbones: {Model Size} × {Temporal Patch Size} = {L, M, S} × {8, 4, 2, 1}. We vary model size by jointly scaling the number of transformer layers, hidden dimensions, and attention heads (see Sec. A.6 of Appendix for details). We show quantitative results from this model scaling in Fig. 6 and qualitative comparisons in Fig. 11. Increasing both temporal resolution and number of model parameters improves realism of the simulation.

Multi-task Compatibility We find that multitask co-training across BP, SceneGen and with random control masks improves performance compared to a single-task, BP only model on the sim agent rollout task, notably reducing collision and offroad rates. We find that jointly learning multiple agent features $(x, y, z, \gamma, \text{size}, \text{type})$ achieves on-par performance with a pose-only (x, y, z, γ) model.

9

Model Architecture Ablation As shown in Tab. 3, replacing AdaLN-Zero conditioning with cross attention leads to a 7.99% decrease in realism performance, largely due to significantly higher collision rates and offroad rates. Removing the agent-wise spatial attention layer very significantly increases collision rate, as it removes the mechanism for agents to learn a joint distribution.

5 Conclusion

We have introduced SceneDiffuser, a scene-level diffusion prior designed for traffic simulation. SceneDiffuser combines scene initialization with scene rollout to provide a diffusion-based approach to closed-loop agent simulation that is efficient (through amortized autoregression) and controllable (through generalized hard constraints). We performed scaling and ablation studies and demonstrated model improvements with computational resources. On WOSAC, we demonstrate competitive results with the leaderboard and state-of-the-art performance among diffusion methods.

Limitations While our amortized diffusion approach is, to our knowledge, the only and best performing closed-loop diffusion-based agent model with competitive performance, we do not exceed current SOTA performance for other autoregressive models. We do not explicitly model validity masks and resort to logged validity in this work. Future work looks to also model the validity mask.

Broader Impact This paper aims to improve AV technologies. With our work we aim to make AVs safer by providing more realistic and controllable simulations. The generative scene modeling techniques developed in this work could have broader social implications regarding generative media and content generation, which poses known social benefits as well as risks of misinformation.

References

[1] Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Błażej Osiński, Hugo Grimmet, and Peter Ondruska. SimNet: Learning reactive self-driving simulations from real-world observations. In *ICRA*, 2021.

[2] Tim Brooks, Bill Peebles, Connor Homes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Wing Yin Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL https://openai.com/research/video-generation-models-as-world-simulators.

[3] Wei-Jer Chang, Francesco Pittaluga, Masayoshi Tomizuka, Wei Zhan, and Manmohan Chandraker. Controllable safety-critical closed-loop traffic simulation via guided diffusion, 2023.

[4] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[5] Kashyap Chitta, Daniel Dauner, and Andreas Geiger. Sledge: Synthesizing simulation environments for driving agents with generative models, 2024.

[6] Younwoo Choi, Ray Coden Mercurius, Soheil Mohamad Alizadeh Shabestary, and Amir Rasouli. Dice: Diverse diffusion model with scoring for trajectory prediction, 2023.

[7] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurélien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *ICCV*, 2021.

[8] Lan Feng, Quanyi Li, Zhenghao Peng, Shuhan Tan, and Bolei Zhou. Trafficgen: Learning to generate diverse and realistic traffic scenarios. In *ICRA*, 2023.

[9] Zhiming Guo, Xing Gao, Jianlan Zhou, Xinyu Cai, and Botian Shi. SceneDM: Scene-level multi-agent trajectory generation with consistent diffusion models, 2023.

[10] Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Li Fei-Fei, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models, 2023.

[11] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models, 2022.

[12] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, volume 35, pages 8633–8646, 2022.

[13] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *ICML*, pages 13213–13232. PMLR, 2023.

[14] Zhiyu Huang, Zixu Zhang, Ameya Vaidya, Yuxiao Chen, Chen Lv, and Jaime Fernández Fisac. Versatile scene-consistent traffic scenario generation as optimization with diffusion, 2024.

[15] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. In *ICRA*, 2022.

[16] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[17] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

[18] Chiyu "Max" Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, and Dragomir Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *CVPR*, 2023.

[19] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[20] Jack Lu, Kelvin Wong, Chris Zhang, Simon Suo, and Raquel Urtasun. Scenecontrol: Diffusion for controllable traffic scene generation. In *ICRA*, 2024.

[21] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, June 2022.

[22] Reza Mahjourian, Rongbing Mu, Valerii Likhosherstov, Paul Mougin, Xiukun Huang, Joao Messias, and Shimon Whiteson. Unigen: Unified modeling of initial agent states and trajectories for generating autonomous driving scenarios. In *ICRA*, 2024.

[23] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nick Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, Brandyn White, and Dragomir Anguelov. The waymo open sim agents challenge. In *Advances in Neural Information Processing Systems Track on Datasets and Benchmarks*, 2023.

[24] Jiteng Mu, Michaël Gharbi, Richard Zhang, Eli Shechtman, Nuno Vasconcelos, Xiaolong Wang, and Taesung Park. Editable image elements for controllable synthesis. *arXiv preprint arXiv:2404.16029*, 2024.

[25] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*, 2022.

[26] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, David J Weiss, Benjamin Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *ICLR*, 2022.

[27] Matthew Niedoba, Jonathan Lavington, Yunpeng Liu, Vasileios Lioutas, Justice Sefas, Xiaoxuan Liang, Dylan Green, Setareh Dabiri, Berend Zwartsenberg, Adam Scibior, and Frank Wood. A diffusion-model of joint interactive navigation. In *NeurIPS*, 2023.

[28] Yotam Nitzan, Zongze Wu, Richard Zhang, Eli Shechtman, Daniel Cohen-Or, Taesung Park, and Michaël Gharbi. Lazy diffusion transformer for interactive image editing, 2024.

[29] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *ICLR*, 2023.

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, pages 4195–4205, October 2023.

[31] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Learning the language of driving scenarios. In *ICLR*, 2024.

[32] Ethan Pronovost, Meghana Reddy Ganesina, Noureldin Hendy, Zeyu Wang, Andres Morales, Kai Wang, and Nick Roy. Scenario diffusion: Controllable driving scenario generation with diffusion. In *Advances in Neural Information Processing Systems*, 2023.

[33] Ethan Pronovost, Kai Wang, and Nick Roy. Generating driving scenes with diffusion. In *ICRA Workshop on Scalable Autonomous Driving*, June 2023.

[34] Cheng Qian, Di Xiu, and Minghao Tian. A simple yet effective method for simulating realistic multi-agent behaviors. Technical report, 2023.

[35] Davis Rempe, Jonah Philion, Leonidas J Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *CVPR*, June 2022.

[36] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[37] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022.

[38] Benjamin Sapp, Yuning Chai, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*, pages 86–99. PMLR, 2020.

[39] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S. Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8579–8590, October 2023.

[40] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*, 2018.

[41] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr-a: 1st place solution for 2022 waymo open dataset challenge–motion prediction. *arXiv preprint arXiv:2209.10033*, 2022.

[42] 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*, 46(5):3955–3971, 2024.

[43] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-a-video: Text-to-video generation without text-video data. In *ICLR*, 2023.

[44] Shuo Sun, Zekai Gu, Tianchen Sun, Jiawei Sun, Chengran Yuan, Yuhang Han, Dongen Li, and Marcelo H Ang. Drivescenegen: Generating diverse and realistic driving scenarios from scratch. *IEEE Robotics and Automation Letters*, 2024.

[45] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *CVPR*, 2021.

[46] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *CVPR*, June 2021.

[47] Shuhan Tan, Boris Ivanovic, Xinshuo Weng, Marco Pavone, and Philipp Krähenbühl. Language conditioned traffic generation. *7th Annual Conference on Robot Learning (CoRL)*, 2023.

[48] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. *arXiv preprint arXiv:2111.14973*, 2021.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[50] Eugene Vinitsky, Nathan Lichtlé, Xiaomeng Yang, Brandon Amos, and Jakob Foerster. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. In *NeurIPS Datasets and Benchmarks Track*, 2022.

[51] Yu Wang, Tiebiao Zhao, and Fan Yi. Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023. Technical report, Pegasus, 2023.

[52] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. In *ICRA*, 2023.

[53] Chen Yang, Aaron Xuxiang Tian, Dong Chen, Tianyu Shi, and Arsalan Heydarian. Wcdt: World-centric diffusion transformer for traffic scene generation, 2024.

[54] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. Trafficbots: Towards world models for autonomous driving simulation and motion prediction. In *ICRA*, 2023.

[55] Zihan Zhang, Richard Liu, Kfir Aberman, and Rana Hanocka. Tedi: Temporally-entangled diffusion for long-term motion synthesis, 2023.

[56] Ziyuan Zhong, Davis Rempe, Yuxiao Chen, Boris Ivanovic, Yulong Cao, Danfei Xu, Marco Pavone, and Baishakhi Ray. Language-guided traffic simulation via scene-level diffusion. In *CoRL*, 2023.

[57] 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 *ICRA*, 2023.

A Appendix / supplemental material

A.1 WOSAC Metrics

Suppose there are $N \approx 500k$ scenarios, each of length $T = 80$ steps, each containing $A \leq 128$ agents (objects). For each scenario, we generate $K = 32$ samples (conditioned on the true initial state), which is a set of trajectories for each object for each time step, where each point in the trajectory is a $D = 4$-dim vector recording location (x, y, z) and orientation θ. Let all this generated data be denoted by $x(1 : N, 1 : A, 1 : K, 1 : T, 1 : D)$. Let the ground truth data be denoted $x^*(1 : N, 1 : A', 1 : T, 1 : D)$. Below we discuss how to evaluate the likelihood of the true (test) dataset x^* under the distribution induced by the simulated dataset x.

(Note that we may have $A' > A$, since the ground truth (GT) can contain cars that enter the scene after the initial prefix used by the simulator; this is handled by defining a validity mask, $v(1 : N, 1 : T, 1 : A')$, which is set to 0 if we want to exclude a GT car from the evaluation, and is set to 1 otherwise.)

Rather than evaluating the realism of the full trajectories in the raw (x, y, z, θ) state space, WOSAC defines $M = 9$ statistics (scalar quantities of interest) from each trajectory. Let $F_j(x(i, a, :))$ represent the set of statistics/features (of type j) derived from $x(i, a, 1 : K, 1 : T)$ by pooling over T, K. This is used to compute a histogram $p_{ija}(.)$ for the empirical distribution of F_j for scenario i. Let $F_j(x^*(i, a, t))$ be the value of this statistic from the true trajectory i for vehicle a at time t . Then we define the negative log likelihood to be

$$NLL(i, a, t, j) = -\log p_{ija}(F_j(x^*(i, a, t))) \tag{3}$$

The j'th metric for scenario i is defined as

$$m(a, i, j) = \exp\left(-\left[\frac{1}{N(i, a)}\right]\sum_t v(i, a, t)NLL(i, a, t, j)\right)$$

$$m(i, j) = \frac{1}{A}\sum_a m(a, i, j) \tag{4}$$

$$N(i, a) = \sum_t v(i, a, t) \text{ is the number of valid points.}$$

Finally an aggregated metric, used to rank entries, is computed as

$$score = \frac{1}{N'}\frac{1}{M}\sum_{i=1}^{N'}\sum_{j=1}^{M} w_j m(i, j) \tag{5}$$

where $0 \leq w_j \leq 1$.

The 9 component metrics are defined as linear speed, linear acceleration magnitude, angular speed, angular acceleration magnitude, distance to nearest object, collisions, time-to-collision (TTC), distance to road edge, and road departures.

A.2 SceneGen Metrics

We instead let $F_j(x(i, :))$ represent the set of statistics/features (of type j) derived from $x(i, 1 : A', 1 : K, -H : T)$ by pooling over T, A', K. This is used to compute a histogram $p_{ij}(.)$ for the empirical distribution of F_j for scenario i.

A.3 Additional Evaluation Details

Simulation step validity Due to the requirement of validity masks during inference, which is applied as an attention padding mask within the transformer, the model does not generate valid values for invalid steps. As the WOSAC challenge evaluates simulation agents for all steps, regardless of the step's validity, we use linear interpolation / extrapolation to impute values for all invalid steps in our simulations for the final evaluation.

A.4 Additional Dataset Information

WOSAC uses the v1.2.0 release of WOMD, and we treat WOMD as a set $\mathcal{D}$ of scenarios where each scenario is a history-future pair. This dataset offers a large quantity of high-fidelity object behaviors and shapes produced by a state-of-the-art offboard perception system. We use WOMD's 9 second 10 Hz sequences (comprising H = 11 observations from 1.1 seconds of history and 80 observations from 8 seconds of future data), which contain object tracks at 10 Hz and map data for the area covered by the sequence. Across the dataset splits, there exists 486,995 scenarios in train, 44,097 in validation, and 44,920 in test.

A.5 Additional Amortized Diffusion Algorithm Details

Warm up: At inference time, the rollout process is preceded by a warm up step. The warm up step is necessary for initializing a buffer of future timesteps before any diffusion iterations take place. The warm up entails a single iteration of a one-shot prediction process described in Algorithm 5a. This process samples pure noise for some future steps and conditions the denoising process on the set of past steps.

Amortized autoregressive rollout: In Fig. 4, we provide a visual illustration of our amortized autoregressive rollout procedure. We operate the rollout procedure using a buffer to track future steps in the trajectory. After the warm up, the future buffer contains $\mathcal{T}$ predicted steps with an increasing noise level. Note that step $\tau = 1$ has very little noise applied. The future buffer in this state is denoised for a single iteration using past steps to condition the process. After a single iteration, the clean step at $\tau = 1$ is popped off of the buffer, and it is added to the past steps. Before the next iteration, a step $\tau = \mathcal{T} + 1$ is sampled from a pure noise distribution and is appended to the end of the future buffer. The described rollout process can be repeated to generate trajectories of arbitrary length as clean steps are popped off the buffer.

A.6 Additional Implementation Details

Architecture Details : For our base model, our scene encoder architecture uses 256 latent queries. Each scene token is 256-dimensional, with 4 transformer layers and 4 transformer heads, with a transformer model dimension of 256. We train and run inference with all 128 agents.

Scaling Hyperparameters:
Small Model: Scene token dimension 128, 2 Transformer layers, 128 Transformer model dimensions, 2 Transformer Heads.

Medium Model: Scene token dimension 256, 4 Transformer layers, 256 Transformer model dimensions, 4 Transformer Heads.

Large Model: Scene token dimension 512, 8 Transformer layers, 512 Transformer model dimensions, 8 Transformer Heads.

Optimizer : We use the Adafactor optimizer [40], with EMA (exponential moving average). We decay using Adam, with $\beta_1 = 0.9$, $\text{decay}_{adam} = 0.9999$, weight decay of 0.01, and clip gradient norms to 1.0.

Training details Train batch size of 1024, and train for 1.2M steps. We select the most competitive model based on validation set performance, for which we perform a final evaluation using the test set. We use an initial learning rate of 3×10^{-4}. We use 16 diffusion sampling steps. When training, we mix the behavior prediction (BP) task with the scene generation task, with probability 0.5. The randomized control mask is applied to both tasks.

Additional Hyperparameters To preprocess features, we use scaling constants of $\frac{1}{80}$ for features x, y, z, and compute mean μ and standard deviation σ of features l, w, h.

We preprocess each agent feature f to produce normalized feature f' via $f' = \frac{f - \mu_f}{2 * \sigma_f}$, where:

$$\mu_l = 4.5, \quad \mu_w = 2.0, \quad \mu_h = 1.75, \quad \mu_k = 0.5. \tag{6}$$

and

$$\sigma_l = 2.5, \quad \sigma_w = 0.8, \quad \sigma_h = 0.6, \quad \sigma_k = 0.5. \tag{7}$$

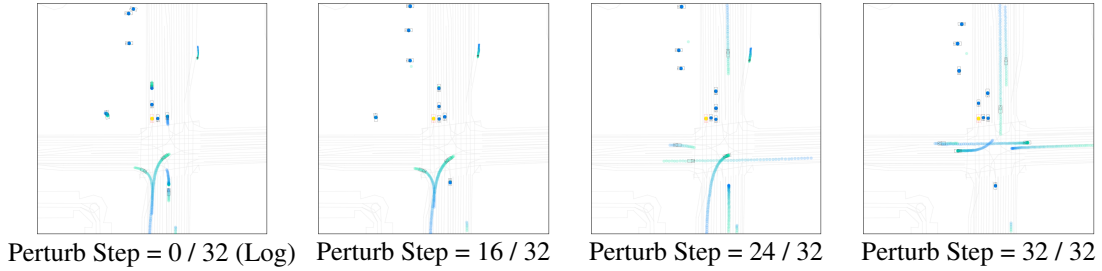| Perturb Step = 0 / 32 (Log) | Perturb Step = 16 / 32 | Perturb Step = 24 / 32 | Perturb Step = 32 / 32 |

Figure 12: Log-perturbation via noising and denoising the original logs to different noise levels. An increasing level of noise added and then removed results in scenes more and more dissimilar from the original log, yet increasingly diverse. The scenes are realistic regardless of the perturbation noise level.

We scale by twice the std σ values to allow sufficient dynamic range for high feature values for some channels.

A.7   Prompts used in Language-based few-shot Scene Generation

A.7.1   Prompt:

Prompt 1: Prompts used in Language-based few-shot Scene Generation.

```
You are writing proto to generate and control an agent's behavior
for an autonomous vehicle (AV) simulation. I will give the follow 2
examples of input and the generated proto MultiAgentMultiConstraint,
which will constrain the agent's position in either past, current, or
future timestep. 5 timesteps equals 1 second, so for example 15 steps
would equal 3 seconds. Given a natural language description of the
agent's desired behavior, please generate the corresponding
MultiAgentMultiConstraint.

Very Important limitations:
1. Only time_step_idx values in [0,8] are valid for PAST time step.
2. Only time_step_idx values in [0] are valid for CURRENT time step.
3. Only time_step_idx values in [0,49] are valid for FUTURE time step.
4. You may only use types POT_CAR, POT_MOTORCYCLIST, POT_PEDESTRIAN to
    generate these examples
5. No two agents should overlap each other at the same time step in
   the same time frame.

The following are 2 examples of a natural language input and the
output is a text file that creates the corresponding constraint.

Example 1
Input: Generate constraints where a motorcycle agent cuts in front of
the AV coming from the right side at some time in the future.
Output:
'''
agent_constraints {
  step_constraints {
    relative_step_constraint {
      time_frame: CURRENT
      time_step_idx: 0
      lat_distance: 3.7 # Slightly right of ADV
      long_distance: 0.0
      agent_type: POT_MOTORCYCLIST
    }
  }
  step_constraints {
    relative_step_constraint {
```

16

```
      time_frame: FUTURE
      time_step_idx: 30
      lat_distance: 0.0
      long_distance: 3.0 # ahead of AV
      agent_type: POT_MOTORCYCLIST
    }
  }
  agent_name: 'motorcycle_0'
}
scene_name: 'single_motorcycle_cut_in'
```

Example 2
Input: Generate constraints where a car agent is tailgating the AV by
following behind it closely.
Output:
```
agent_constraints {
  step_constraints {
    relative_step_constraint {
      time_frame: PAST
      time_step_idx: 0
      lat_distance: 0.0
      long_distance: -15.0
      agent_type: POT_CAR
    }
  }

  step_constraints {
    relative_step_constraint {
      time_frame: CURRENT
      time_step_idx: 0
      lat_distance: 0.0
      long_distance: -10.0
      agent_type: POT_CAR
    }
  }
  step_constraints {
    relative_step_constraint {
      time_frame: FUTURE
      time_step_idx: 30
      lat_distance: 0.0
      long_distance: -2.0
      agent_type: POT_CAR
    }
  }
  agent_name: 'car_0'
}
scene_name: 'single_car_tailgater'
```

Please output just the MultiAgentMultiConstraint in the response and
leave any explanation in the comments. Please double check the
important limitations described before are met.

Now take a deep breath and lets think step by step to write the proto
given the following input:

S-Shaped Maneuver: Create a motorcyclist agent that is doing a
S-shaped maneuver around the AV.

Surrounding Traffic: Create a scene with 8 agents that are surrounding
the AV and traveling along for a few seconds. There are 4 agents on
the left lane of AV in a straight line and 4 agents on the lane of AV
in a straight line. 2 of the front agents at motorcyclists while the
```

```
rest are cars.
```

Prompt 2: (Manual) Cut-in Constraint

```
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: PAST
        time_step_idx: 0
        lat_distance: 3.0
        long_distance: 0.0
        agent_type: POT_CYCLIST
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 9
        lat_distance: 0.0
        long_distance: 0.0
        agent_type: POT_CYCLIST
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 40
        lat_distance: -3
        long_distance: 10
        agent_type: POT_CYCLIST
      }
    }
    agent_name: 'cut_in_0'
}
```

Prompt 3: (Manual) Tailgate Constraint

```
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: PAST
        time_step_idx: 0
        lat_distance: 0
        long_distance: -20
        agent_type: POT_CYCLIST
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 20
        lat_distance: 0.0
        long_distance: -10
        agent_type: POT_CYCLIST
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 40
        lat_distance: 0
        long_distance: -3
```

```
        agent_type: POT_CYCLIST
      }
    }
    agent_name: 'tailgater_agent_0'
}
```

Prompt 4: (LLM Generated) S-shaped Constraint

```
agent_constraints {
    step_constraints {
      # Initial position slightly to the right of the AV (
    time_step_idx 0)
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: 2.0  # Adjust this value as needed for initial
    lateral offset
        long_distance: 0.0
        agent_type: POT_CYCLIST
      }
    }
    step_constraints {
      # Move the motorcycle diagonally to the left in the future (
    around 15 steps)
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 15
        lat_distance: -3.0  # Adjust this value for desired leftward
    movement
        long_distance: 5.0   # Adjust this value for desired forward
    movement
        agent_type: POT_CYCLIST
      }
    }
    step_constraints {
      # Move the motorcycle diagonally back to the right in the future
     (around 30 steps)
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 30
        lat_distance: 2.0  # Adjust this value for desired rightward
    movement
        long_distance: 10.0  # Adjust this value for desired forward
    movement
        agent_type: POT_CYCLIST
      }
    }
    agent_name: 'motorcycle_0'
}
```

Prompt 5: (LLM Generated) Surrounding Traffic Constraint

```
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: -3.7
        long_distance: 10.0
        agent_type: POT_MOTORCYCLIST
      }
    }
    step_constraints {
      relative_step_constraint {
```

```
            time_frame: FUTURE
            time_step_idx: 25
            lat_distance: -3.7
            long_distance: 30.0
            agent_type: POT_MOTORCYCLIST
        }
    }
    agent_name: 'motorcycle_left_0'
}
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: -3.7
        long_distance: 5.0
        agent_type: POT_CAR
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 25
        lat_distance: -3.7
        long_distance: 25.0
        agent_type: POT_CAR
      }
    }
    agent_name: 'car_left_0'
}
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: -3.7
        long_distance: -5.0
        agent_type: POT_CAR
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 25
        lat_distance: -3.7
        long_distance: -1.0
        agent_type: POT_CAR
      }
    }
    agent_name: 'car_left_1'
}
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: -3.7
        long_distance: -10.0
        agent_type: POT_CAR
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 25
```

```
          lat_distance: -3.7
          long_distance: -6.0
          agent_type: POT_CAR
        }
      }
    agent_name: 'car_left_2'
}
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: 3.7
        long_distance: 10.0
        agent_type: POT_MOTORCYCLIST
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 25
        lat_distance: 3.7
        long_distance: 30.0
        agent_type: POT_MOTORCYCLIST
      }
    }
    agent_name: 'motorcycle_right_0'
}
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: 3.7
        long_distance: 5.0
        agent_type: POT_CAR
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 25
        lat_distance: 3.7
        long_distance: 25.0
        agent_type: POT_CAR
      }
    }
    agent_name: 'car_right_0'
}
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: 3.7
        long_distance: -5.0
        agent_type: POT_CAR
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 25
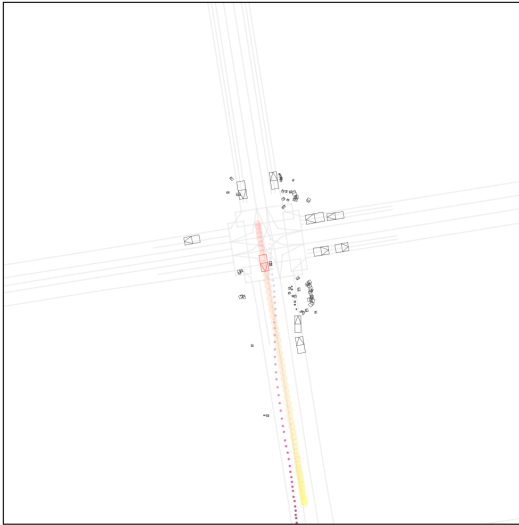        lat_distance: 3.7
        long_distance: -1.0
```

```
            agent_type: POT_CAR
        }
    }
    agent_name: 'car_right_1'
}
agent_constraints {
    step_constraints {
      relative_step_constraint {
        time_frame: CURRENT
        time_step_idx: 0
        lat_distance: 3.7
        long_distance: -10.0
        agent_type: POT_CAR
      }
    }
    step_constraints {
      relative_step_constraint {
        time_frame: FUTURE
        time_step_idx: 25
        lat_distance: 3.7
        long_distance: -6.0
        agent_type: POT_CAR
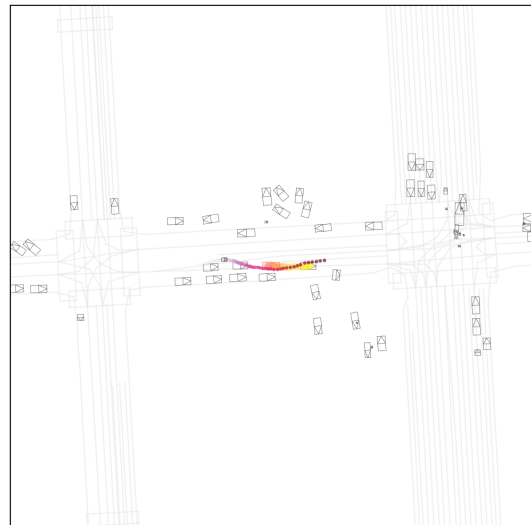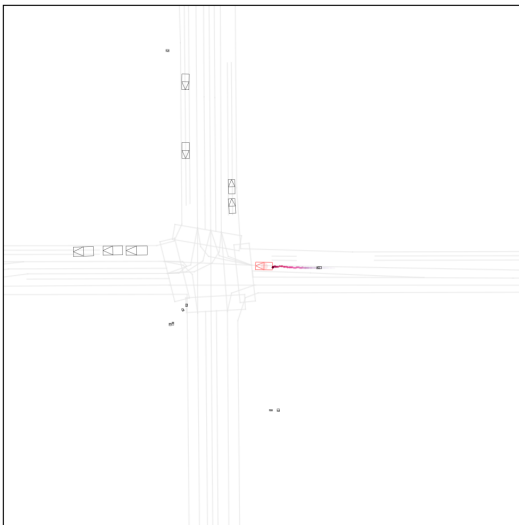      }
    }
    agent_name: 'car_right_2'
}
```

A.7.3 Controllable Scenegen Results

Qualitative results showing one successful and one failed example of applying the control points to scene generation task with the protos are listed in Appendix A.7.2. For measuring the success and failures of this scene generation task, we randomly selected 25 examples that were generated with each of the 4 control protos and qualitatively determined success on 1) if the new object does not overlap with any existing objects in the scene and 2) if the new object semantically behaves in the way intended by the control points. Otherwise, we considered it a failure. Overall, we measured a success rate of 40/100.

Cut-in (left: success, right: failure)

Tailgate (left: success, right: failure)

S-shape (left: success, right: failure)

Surrounding Traffic (left: success, right: failure)

A.8 Scene Generation

We show additional unconditioned scene generation results in Fig. 13.

A.9 Generalized Hard Constraint Definitions

Non-collision Constraints ensure the boxes of generated agents do not overlap. We define the potential field of agent a to be a rounded square potential $\phi_a(x,y) = \frac{1}{(x-x_a)^4+(y-y_a)^4+\epsilon}$ if $||(x - x_a, y - y_a)||_2 < 1.5$ else 0. We define $\text{clip}_{\text{collision}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \left(\sum_{a \in A} \sum_{i=\pm 0.5, j=\pm 0.5} \sum_{a' \in A, a' \neq a} \phi_{a'}(x_a + iw_a, y_a + jl_a) \right)$ that minimizes the potential of each agent's corners against all other agents. (x, y) is defined in the normalized space.

Range Constraints limit a certain feature $\boldsymbol{x}_d$ within the range of d_{min} and d_{max}. In the context of Scene Generation for example, this can be used to limit the length of a vehicle to an arbitrary range, e.g. between 7-9 meters. We have $\text{clip}_{\text{range}}(\boldsymbol{x}_d) = \min(\max(\boldsymbol{x}_d, d_{\min}), d_{\max})$.

Figure 13: Results of unconditioned scene generation for randomly selected road locations. For each example, we show the ground truth log along with 3 generated scenes.

Onroad Constraints ensure that the bounding boxes of specified generated agents stay on road. We define the offroad potential of road graph polyline i to be $\phi_i(x, y) = (x - x_i)^2 + (y - y_i)^2$ if $W_i(x, y) = 0$ else 0, where (x_i, y_i) is the closest point on the road graph with respect to (x, y) and $W_i(x, y)$ is the winding number of position (x, y) to polyline i, such that we only penalize a trajectory for going offroad. We only consider the closest road graph segment and only consider trajectories that are more than $> 20\%$ onroad. We define $\text{clip}_{\text{onroad}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \left(\min_{i \in RG} \phi_i(x, y) \right)$.

25

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: It accurately reflects the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes] .

   Justification: See Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical results included.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe in detail the model architecture, training details, and algorithmic pseudocode for our inference procedures.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes] and [No] .

Justification: We do not plan to release code in the near future. However our methods are clearly described and our dataset is based on the Waymo Open Motion Dataset, which is already publicly accessible.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [No]

Justification: It is not affordable to train multiple models for error bars in our experiments, especially the Large models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Due to the high computational cost it is infeasible to run multiple training runs of the large models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide number of model parameters as well as inference FLOPS count for our family of models in Figure 6.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have reviewed the Code of Ethics and confirm that we conform to the guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Discussed in Section 5.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We are not releasing data or models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We accessed and utilized public datasets (the Waymo Open Dataset) in compliance with the licenses.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] .

Justification: the paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA] .

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA] .

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.