Rolling Ahead Diffusion for Traffic Scene Simulation

Yunpeng Liu[1,2] **Matthew Niedoba**[1,2] **William Harvey**[1] **Adam Ścibior**[2]
Berend Zwartsenberg[2] **Frank Wood**[1,2,3]

[1]University of British Columbia, [2]Inverted AI, [3]Amii

Abstract

Realistic driving simulation requires that NPCs not only mimic natural driving behaviors but also react to the behavior of other simulated agents. Recent developments in diffusion-based scenario generation focus on creating diverse and realistic traffic scenarios by jointly modelling the motion of all the agents in the scene. However, these traffic scenarios do not react when the motion of agents deviates from their modelled trajectories. For example, the ego-agent can be controlled by a stand along motion planner. To produce reactive scenarios with joint scenario models, the model must regenerate the scenario at each timestep based on new observations in a Model Predictive Control (MPC) fashion. Although reactive, this method is time-consuming, as one complete possible future for all NPCs is generated per simulation step. Alternatively, one can utilize an autoregressive model (AR) to predict only the immediate next-step future for all NPCs. Although faster, this method lacks the capability for advanced planning. We present a rolling diffusion based traffic scene generation model which mixes the benefits of both methods by predicting the next step future and simultaneously predicting partially noised further future steps at the same time. We show that such model is efficient compared to diffusion model based AR, achieving a beneficial compromise between reactivity and computational efficiency.

Introduction

Traffic simulation is essential in the development of autonomous driving systems, particularly for addressing sim-to-real issues during real-world deployment. A key component is the realistic behavior of simulated surrounding agents or non-playable characters (NPCs), as noted by (Gulino et al. 2024). One approach involves replaying recorded driving behaviors, obtained either from overhead drones (Zhan et al. 2019) or onboard vehicle recordings (Sun et al. 2020). According to (Gulino et al. 2024), reinforcement learning policies trained on such driving logs outperform those trained with NPCs controlled by the rule-based reactive planner like the Intelligent Driver Model (IDM) (Treiber, Hennecke, and Helbing 2000). However, this log-replay approach has two drawbacks, the collection of driving records is both costly and time-consuming (Rempe et al. 2022; Liu et al. 2023). In addition, real-world agents react to the ego agent's behavior with diverse actions, a dynamic not captured by IDM.

An increasingly popular alternative involves controlling simulated NPCs with generative driving behavior models (Suo et al. 2021; Ścibior et al. 2021; Xu et al. 2023), which learn to generate realistic driving trajectories from recorded behaviors. These models facilitate the generation of a diverse set of synthetic driving logs. More recently, diffusion model based traffic scene prediction models have become more popular (Guo et al. 2023; Jiang et al. 2023; Niedoba et al. 2024). These models generate joint future trajectories for all agents based on initial observations and allow for flexible conditioning at test time with classifier guidance for tasks like adversarial scenario generation (Zhong et al. 2023) and scenario editing (Niedoba et al. 2024).

While these diffusion based models are suitable for open-loop simulations, employing them in closed-loop simulations is time-consuming due to the inherently slow generation times of diffusion models. In such setups, one would need to replan at each simulation timestep to maintain maximum reactivity to the uncontrolled ego agent. Current methods utilizing diffusion models for closed-loop traffic scene planning typically generate a small window of steps ahead, then replan (Chang et al. 2023) to create adversarial scenarios. However, this approach may be inefficient, our method improves upon this by partially denoising the future plan and only predicting a clean subsequent state after realizing all agents' states at the last simulation step.

We propose a rolling diffusion-based traffic simulation planner that plans for all agents in the scene in an autoregressive manner, which is four times faster than a typical autoregressive diffusion model in terms of the number of denoising steps, while remaining reactive to the adversarial agent, as demonstrated in our empirical evaluation. Since the denoising operation cannot benefit from parallel computing, this iterative process becomes the bottleneck in simulation speed. Furthermore, our experiments, which involve training on real-world driving logs (Zhan et al. 2019), demonstrate that our rolling-ahead traffic planner produces more realistic scenes than our diffusion-based AR baseline, as measured qualitatively and by scene-level displacement metrics.

Workshop on Machine Learning for Autonomous Driving at AAAI 2025 (ML4AD@AAAI 2025), Philadelphia, PA USA. https://ml4ad.github.io

Background

Diffusion Models

Diffusion models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020; Kingma et al. 2021; Karras et al. 2022; Song et al. 2020) are probabilistic generative models which have recently been applied to the problem of traffic scenario modelling. They are defined based on a forward process which gradually adds Gaussian noise to data. The amount of noise added at any point is based on the "time" in the forward process, τ. We will refer to the copy of the data at a certain diffusion timestep as $\boldsymbol{x}_\tau$. When τ equals zero, $\boldsymbol{x}_\tau = \boldsymbol{x}_0$ is the original data with no noise. For positive τ, the distribution of $\boldsymbol{x}_\tau$ given $\boldsymbol{x}$ is

$$q(\boldsymbol{x}_\tau|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_\tau; \alpha_\tau \boldsymbol{x}_0, \sigma_\tau^2 \boldsymbol{I}), \tag{1}$$

where α_τ and σ_τ are τ-dependent scalars and $\tau \in [0, 1]$. In this paper, we will set $\alpha_\tau = 1$ for all τ following EDM (Karras et al. 2022). We will define σ_τ to be an increasing function which is zero when τ is zero and large when $\tau = 1$. The signal-to-noise ratio is defined by

$$\text{SNR}(\tau) = \frac{\alpha_\tau^2}{\sigma_\tau^2}. \tag{2}$$

Given our choices of α_τ and σ_τ, the signal to noise ratio is small for $\tau = 1$. This means that $q(\boldsymbol{x}_1|\boldsymbol{x}_0)$ will be well-approximated by a zero-mean Gaussian, and therefore the marginal $q(\boldsymbol{x}_1|\boldsymbol{x}_0) = \int q(\boldsymbol{x}_1|\boldsymbol{x}_0) p_{\text{data}}(\boldsymbol{x}_0) \mathrm{d}\boldsymbol{x}_0$ will also be roughly Gaussian.

It is possible to "invert" this forward process, yielding a reverse process that can transport samples from $q(\boldsymbol{x}_1)$ to $q(\boldsymbol{x}_0)$, as shown by (Song et al. 2020). This reverse process maps a roughly-Gaussian distribution to the data distribution, allowing us to make realistic samples out of noise. Doing so only requires an approximation of the score function $\nabla_{\boldsymbol{x}_\tau} \log q(\boldsymbol{x}_\tau)$. Such an approximation can be learned by optimizing the loss (Song et al. 2020)

$$\begin{aligned} &\mathcal{L}_{\text{diffusion}}(\theta) = \\ &\mathbb{E}_{p_{\text{data}}(\boldsymbol{x}_0) q(\boldsymbol{x}_\tau|\boldsymbol{x}_0) u(\tau)} \left[\omega(\tau) \| D_\theta(\boldsymbol{x}_\tau; \tau) - \boldsymbol{x}_0 \|_2^2 \right] \end{aligned} \tag{3}$$

where $u(\tau)$ is a distribution over diffusion times to use during training, $\omega(\boldsymbol{\tau})$ is a function that weights the contribution of each timestep to the loss, and $D_\theta(\cdot, \cdot)$ represents a neural network that produces an output of the same shape as $\boldsymbol{x}_0$. This neural network learns to estimate clean data given noisy data, and such an estimate can be used to produce an estimate of the score function (Song et al. 2020). This can be used to simulate the SDE that produces sampled clean data $\boldsymbol{x}_0$ given samples from a Gaussian approximating $q(\boldsymbol{x}_1)$. Through out the paper we use τ as the time in the diffusion process and t as the chronological time within a scenario. Note that the diffusion model can be made conditional on any extra information by inputting the extra information into the neural network in Eq. (3) and providing it to the neural network in the same way at test-time (Tashiro et al. 2021).

Temporally Correlated Diffusion Models

The above formulation of diffusion models becomes less efficient in terms of memory and computational resources as the dimensions of $\boldsymbol{x_0}$ grow, particularly for long sequence modeling. The authors of the rolling diffusion model (RDM) (Ruhe et al. 2024) introduce a method by modelling a sliding window of $\boldsymbol{x_0}$, given the assumptions that elements that are far in the past from the sliding window are irrelevant.The authors propose assigning temporally correlated noise levels to the elements within the sliding window, introducing a temporal inductive bias to the model. Previous work has shown that this particular inductive bias helps generating diverse and high quality samples in long term human motion predictions (Zhang et al. 2023) and being efficient in NLP tasks like summarization and translation (Wu et al. 2024). RDM formalizes the temporal noise correlation approach and provides a local and global perspective of modelling long sequences of videos.

We begin our discussion by examining a general rule for diffusion models that incorporate temporal noise correlation for sequences. We then delve into the specific mechanisms of the RDM, reviewing both its forward and reverse processes along with its objective function, and the two operating stage as depicted in Figure 2.

Given a long sequence of data with length T, RDM approaches the problem from a local perspective by examining a single sliding window of length W. Within this framework, RDM defines a function g that maps a global diffusion step τ to a local diffusion step $\tau_w \in [0, 1]$ given the local window index w. The fundamental rule for diffusion models with temporal noise correlation is

$$\text{SNR}(\tau_{w+1}) < \text{SNR}(\tau_w),$$

indicating the temporal nature of the sequence results in increased uncertainty as time progresses. As shown in Figure 2, darker-colored circles represent higher uncertainty because they are at a higher noise level.

Given a sampled sequence index t, The forward process within the local window in RDM is

$$q(\boldsymbol{x}_\tau^{t:t+W}|\boldsymbol{x}_0^{t:t+W}) = \prod_{w=t}^{t+W-1} \mathcal{N}(\boldsymbol{x}_\tau^w; \alpha_{\tau_w} \boldsymbol{x}_0^w, \sigma_{\tau_w}^2 \boldsymbol{I}), \tag{4}$$

where the diffusion parameter α_{τ_w} and σ_{τ_w} are local diffusion step τ_w-dependent scalars rather than τ in Eq. (1). This represents the fact that more noise is added to the later frame as α_τ and σ_τ is monotonically increasing with respect to τ.

For the reverse process, RDM defines

$$\begin{aligned} &p_\theta(\boldsymbol{x}_{\tau-1}^{t:t+W}|\boldsymbol{x}_\tau^{t:t+W}) := \\ &\prod_{w=t}^{t+W-1} q(\boldsymbol{x}_{\tau-1}^w|\boldsymbol{x}_\tau^{t:t+W}, \quad \boldsymbol{x}^w = f_\theta(\boldsymbol{x}_\tau, \tau_w)). \end{aligned} \tag{5}$$

The benefit arising from such formulation is that instead of training on the full sequence T which is memory inefficient and complex, RDM's objective function is defined only within a sampled sliding window with length W,

$$\mathbb{E}_{\substack{\tau \sim \mathcal{U}(0,1), \\ \boldsymbol{x}_\tau^{t:t+W} \sim q}} \left[\sum_{w=t}^{t+W-1} \omega(\tau_w) \| D_\theta(\boldsymbol{x}_\tau^{t:t+W}; \tau_w) - \boldsymbol{x}_0^w \|_2^2 \right]. \tag{6}$$

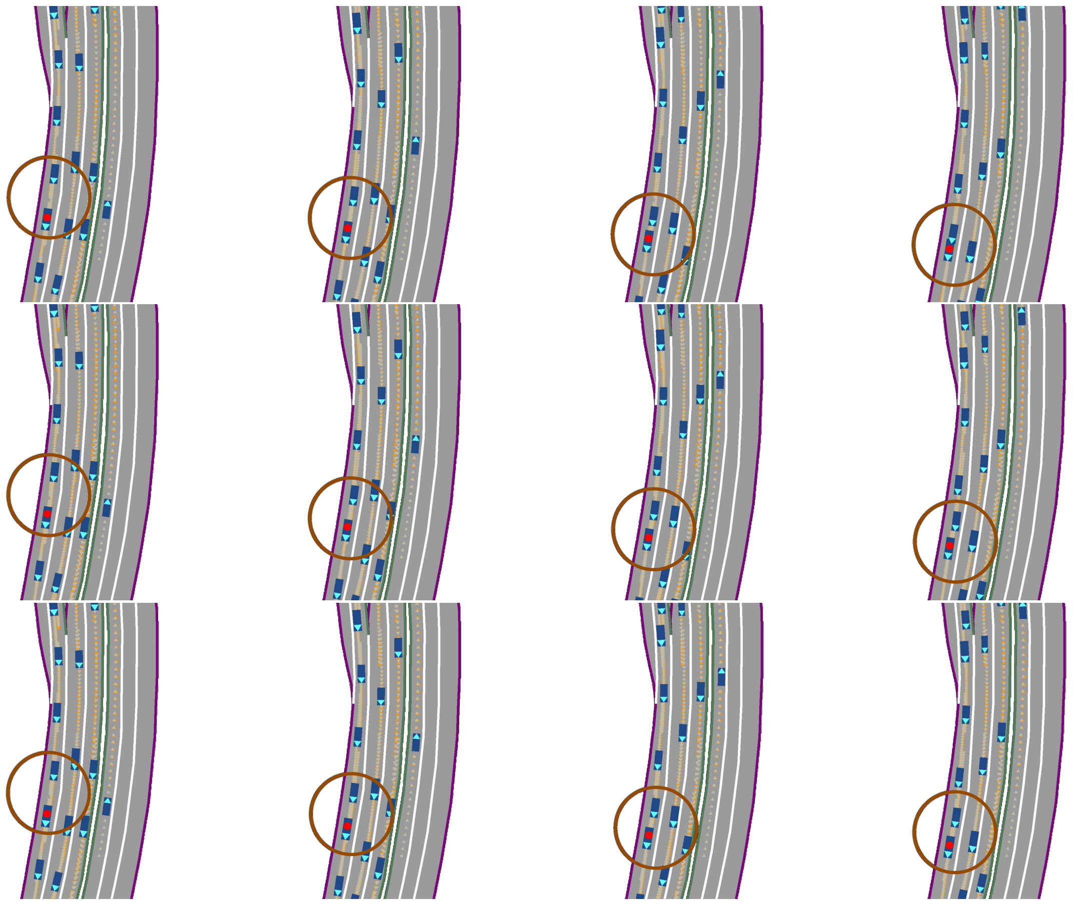

Figure 1: From top to bottom row: DJINN (Niedoba et al. 2024), autoregressive (AR), and RoAD (Ours). The adversarial agent, marked with a red dot, follows its replay log and slows down to reach only half its trajectory by the end of the simulation. Brown circles highlight the interaction region. The agents controlled by the RoAD and AR models slow down to react to the adversarial agent, while agents controlled by the DJINN model do not. Ground truth trajectories are shown in gray, and predicted trajectories are shown in orange.

There are two stages of RDM, the warm-up stage and the rolling stage. In the warm-up stage, the model handles the initial boundary condition by generating from white noise, as shown in the first row of Figure 2 (left), and denoises it to produce one clean element and partially denoised future elements in the sliding window as shown in the bottom row. Once it reaches the temporal correlated noise stage (Bottom row of Figure 2 left), RDM takes few denoising steps for the next step prediction shown in Figure 2 (right). This requires the model to train two tasks, where β controls the training task distribution and for each tasks, RDM designs an associated function g for calculating the local diffusion time τ_w given τ and window index w. In addition, we can condition n number of clean observations within the sliding window, g is defined for warm-up and rolling stage as

$$g_{\text{warm-up}}(\tau, w) := \max(\min(\frac{w}{W} + \tau, 1.0), 0.0) \tag{7}$$

$$g_{\text{rolling}}(\tau, w) := \max(\min(\frac{w + \tau - n}{W - n}, 1.0), 0.0), \tag{8}$$

where n,W are application-dependent hyperparameters.

Related Work

Traffic Simulation with Diffusion Models

Predicting the motion of road users is a critical task for autonomous vehicle driving or simulation. For this reason, the number of methods which have attempted to model traffic behavior is vast. The literature contains a variety of techniques for modelling the distribution of driving behavior, including mixture models (Chai et al. 2019; Cui et al. 2019; Nayakanti et al. 2023), variational autoencoders (Ścibior et al. 2021; Suo et al. 2021), and generative adversarial networks (Zhao et al. 2019).

Our work builds upon recent methods which model driving behavior using diffusion models. In CTG (Zhong et al. 2023), the authors model the motion of each agent in the scene independently with a Diffuser (Janner et al. 2022) based diffusion model. The authors of (Chang et al. 2023) also model agent motions via diffusion, with a focus on controllability. By contrast, most other diffusion based traffic models model entire traffic scenes. This includes MotionDiffuser (Jiang et al. 2023), Scenario Diffusion (Pronovost

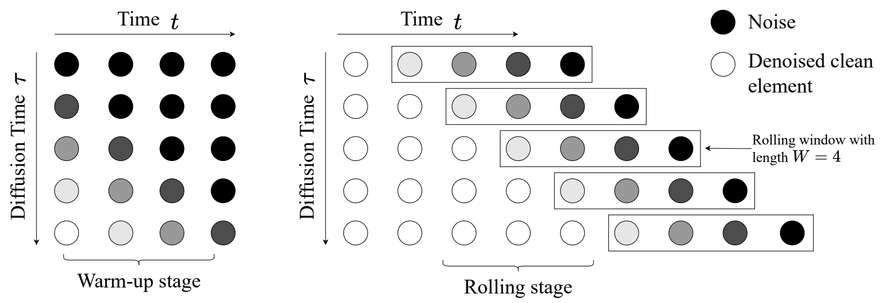

Figure 2: Rolling Diffusion Model. Columns represent sequence timesteps and rows represent diffusion timesteps. Circles are shown in white if the corresponding sequence timestep is fully denoised; black if the sequence timestep is pure noise; and grey if in between. During the denoising process, the SNR for each element in the rolling window depends on the local diffusion time τ_w which can be calculated using Eq. (7) or Eq. (8), depending on whether it is in the warm-up or rolling stage.

et al. 2023) and SceneDM (Guo et al. 2023) which all diffuse the joint motion of all agents in the scene. Our work builds directly on that of DJINN (Niedoba et al. 2024), which utilizes a transformer based network to generate joint traffic scenarios based on a variable set of agent state observations. Crucially, due to the expensive computational cost of diffusion model sampling, only CTG (Zhong et al. 2023) utilize their model for closed-loop scenario simulation. Twice per second, they incorporate new state observations and resample trajectories for each agent. By comparison, our method does not require iterative replanning, greatly improving simulation speed.

Methods

Problem Formulation

We refer to the motion of A agents across T discrete times in an environment $\mathcal{M}$ as a traffic scenario. Formally, we define the scenario as $\boldsymbol{x} \in \mathbb{R}^{A\times T\times 3}$, where we represent the state of each agent $a \in A$ at time $t \in T$ as the combination of its 2D position and 1D orientation. We introduce a probabilistic planner π_{sim} which jointly predicts the future states for all agents, conditioned on static map information $\mathcal{M}$ and previously observed agent states $x_{obs} \in \mathbb{R}^{A\times t_{obs}\times 3}$.

A more difficult form of this planning problem is closed-loop traffic simulation. In closed-loop simulation one agent, known as the *ego agent* a^{ego}, is typically controlled by a standalone motion planner π_{ego} which may be a black-box and which may cause the ego agent to drive very differently to any agents in the training data and thus is not amenable to accurate prediction by the traffic scenario planner π_{sim}. At each time step t, the standalone motion planner π_{ego} plans the single next step for the ego-agent given the entire history of the scenario, $\boldsymbol{x}^{0:t}$ and the map $\mathcal{M}$. The closed-loop traffic simulation problem is to model the behavior of every other agent in this scene, including potential interactions with the ego agent. Since the state of the ego agent is neither controllable nor known in advance, the traffic scenario planner π_{sim} must continually update its plan to be continuously conditioned on the most recent state and actions of the ego agent.

Replanning with a joint prediction model

Our baseline planner relies on a conditional diffusion model $p(\boldsymbol{x}^{t_{obs}:T}|\boldsymbol{x}^{0:t_{obs}}, \mathcal{M}, \boldsymbol{c})$, which jointly predicts the scenario for all agents in the scene up to time T given the map $\mathcal{M}$ and additional conditioning information c. Although diffusing the joint states of all agents is a flexible way of modelling the distribution of traffic scenarios, the model does not respond to the ego agent trajectories which deviate from modelled behavior. To mitigate this, one option is to regenerate the traffic scenario after each simulator step to incorporate new ego agent state observations. We select DJINN (Niedoba et al. 2024) as our conditional diffusion model and we denote this method of iterative planning as DJINN-MPC as it resembles a traditional model predictive control loop. This allows the scenario simulation planner π_{sim} to adjust its predictions at every simulation step in response to the standalone ego agent in the scene.

Diffusion based autoregressive model (Diff-AR)

One key drawback of DJINN-MPC is that we must fully diffuse a new traffic scenario at every simulator step, at significant cost. As an alternative, one can train an diffusion based autoregressive model as the simulation planner, which factorizes the conditional probability as

$$p(\boldsymbol{x}^{t_{obs}:T}|\boldsymbol{x}^{0:t_{obs}}, \mathcal{M}, \boldsymbol{c}) = \prod_{t=t_{obs}}^{T-1} p(\boldsymbol{x}_t|\boldsymbol{x}_0^{0:t-1}, \mathcal{M}, \boldsymbol{c}).$$

Given the past observations, the model only predicts one subsequent step. In practice, the history of past observations $\boldsymbol{x}_0^{0:t-1}$ is truncated to a fixed length. Compared to the previous method, Diff-AR is slightly more efficient as it only denoises the single-step future from scratch. However, Diff-AR cannot anticipate other agents' long-term behaviors beyond the immediate next step which is important for effective planning in many traffic scenarios.

Rolling ahead autoregressive model

We propose a rolling diffusion based model (RoAD) for traffic scenario planning based on RDM. We start by providing an overview of our autoregressive traffic planner, then discuss some details of our design choices on the diffusion process and the model with our updated objective function.

We utilize a sliding window of length W, which is much smaller than the scenario length T. This sliding window includes t_{obs} clean observations for all agents. Within this window, only the t_{obs+1}th state is fully denoised at each scenario time for all agents, while the remainder of the sequence undergoes partial denoising. At the next simulation step, we then shift the sliding window and repeat the previous process. By focusing on a smaller window and selectively denoising, our approach maintains computational efficiency while preserving the ability to adaptively plan for the immediate future.

We follow the design choices from EDM (Karras et al. 2022) in designing our diffusion process. Given a local diffusion time τ_w, during training, our σ_τ is a continuous version of the sampling noise schedule in EDM,

$$\sigma_\tau = ({\sigma_{\max}}^{\frac{1}{\rho}} + \tau({\sigma_{\min}}^{\frac{1}{\rho}} - {\sigma_{\max}}^{\frac{1}{\rho}}))^{\rho},$$

where we keep the default hyper-parameter choice for $\sigma_{\max}$, $\sigma_{\min}$ and ρ from (Karras et al. 2022). We apply the Heun 2nd order sampler to sample at prediction time with the same hyper-parameter reported in EDM. We referred the reader to EDM for the detailed denoising algorithms.

As we are interested in modelling a joint traffic planner for all agents in the scene, we diffuse in a global coordinate. We adopt the map representation from (Niedoba et al. 2024) where $\mathcal{M}$ is represented as an unordered set of polylines, each polylines describing lane centers and normalized for length and scale to match the agent states. Our model, built on a transformer-based architecture utilizes a feature tensor shaped $[A, W, F]$ to process agent trajectories and map information. It embeds noisy and observed states, temporal indices, and the local diffusion step τ_w into high-dimensional vectors with feature dimension F. We apply per-agent positional embeddings to these feature vectors, which are then fed into a series of transformer blocks that perform self-attention in the time and agent dimensions, as well as cross-attention with the map features.

Rather than denoise the sequence one by one as in Eq. (6), our transformer architecture jointly predicts the score for all noisy states in the window. Denote $\boldsymbol{x}^W$ as the sliding window of interest. Our score estimator D_θ takes in $\boldsymbol{x}^W_\tau$, $\boldsymbol{\tau}$, and the map $\mathcal{M}$, along with additional conditional information $\boldsymbol{c}$ that includes the dimensions of each agent. Our updated objective function is

$$\mathbb{E}_{\boldsymbol{x}_0^W, \boldsymbol{\tau}, \boldsymbol{x}_\tau^W}[\omega(\boldsymbol{\tau})\|D_\theta(\boldsymbol{x}_\tau^W, \mathcal{M}, \boldsymbol{c}, \boldsymbol{\tau}) - \boldsymbol{x}_0^W\|_2^2]. \quad (9)$$

Note that $\boldsymbol{x}_\tau^W$ is sampled from RDM forward process defined in Eq. (4) that contains states with noise level according to τ_w. While local diffusion time τ_w depends on the window index w and the global diffusion steps τ, all agents in the scene has a consistent local diffusion time τ_w. Therefore, our score estimator D_θ takes a vector $\boldsymbol{\tau} = \{\tau_w\}_{w=0}^W$ to reflect the temporal correlation of different nose levels in $\boldsymbol{x}_\tau^W$. The weighting term is also a vector that takes vector $\boldsymbol{\tau}$ as input then assign different weights according to each τ_w.

While our rolling ahead autoregressive model is efficient for long traffic scenario planning, the partially denoised future plan affects the reactivity of our model. In traffic simulation, such degradation may cause a higher collision rate with the uncontrolled ego agent in the scene. The reactivity of the model depends on the SNR of future states. We empirically evaluate the reactivity of our model compared to the AR baseline in our experiment section.

Conditioning Augmentation

We have empirically found that noise conditioning augmentation, as described by (Ho et al. 2022b), is essential for all models operating in an autoregressive manner. This augmentation is critical for autoregressive human motion generation (Yin et al. 2023), cascaded diffusion models for class-conditional generation, and super-resolution video generation (Ho et al. 2022a). Noise conditioning augmentation enhances the model's robustness against generated noise, which serves as observations for subsequent predictions (Ho et al. 2022b), and it mitigates the risk of the model overfitting to its autoregressive nature. In the context of traffic simulation, this augmentation aids in generating smooth trajectories and ensures that the model does not ignore other conditional factors, such as the presence of other agents and, importantly, the map M of the environment. Previous work on diffusion-based traffic simulation (Zhang et al. 2023; Chang et al. 2023) circumvents the issue of noisy observations by relying on a kinematic model to produce smooth trajectories; however, our autoregressive traffic simulation planner does not require such a kinematic model.

We follow (Ho et al. 2022b) to employ conditioning augmentation for our rolling ahead traffic planner with an important modification. During training, given a sampled training segment $\boldsymbol{x}^W$ with length W, and n observations $\boldsymbol{x}^{obs}$ within this segment, we apply Gaussian noise augmentation to the $\boldsymbol{x}^{obs}$, where the noise level $\sigma_{\tau_{ca}}$ is sampled uniformly between $\sigma_{\min}$ and $\sigma_{\max}$. Unlike (Ho et al. 2022b), we found jointly predicting all elements within the sliding window, including the noised observations is enssential for our application. At testing time, we apply Gaussian noise with $\sigma_{\min}$ to our observations for minimal level of augmentation.

From a broader perspective, conditioning augmentation addresses a well-understood issue in imitation learning with autoregressive-style methods, where at test-time the model must condition on samples that it produced earlier. Throughout a roll-out, the distribution of these samples may shift so that they appear out-of-distribution relative to the training data. One solution to this problem allows the model to learn from its own mistakes using a differentiable simulator (Ścibior et al. 2021). In the diffusion model context, though, this requires sampling from the reverse process which is expensive. We denote this type of augmentation as reverse process conditioning augmentation, where the noise originates from the model's prediction. Existing work (Ho et al. 2022b) on cascaded diffusion models has achieved comparable performance through both reverse process conditioning augmentation and forward process conditioning augmentation for high-resolution image generation conditioned on a low-resolution image. Therefore, we opt for the more efficient forward process conditioning augmentation approach.

Experiments

We evaluate our rolling ahead scene generation model (RoAD) on the INTERACTION dataset (Zhan et al. 2019), which contains 16.5 hours of driving records across 11 locations. Our baselines include an autoregressive diffusion model (AR), which takes observations of length 10 and predicts the next step future for all agents. Another baseline, DJINN, is a scene generation model that takes 10 observations and jointly predicts the next 30 steps at 10Hz for all agents in a one-shot manner. We have also trained a version of DJINN that predicts 10 future steps ahead jointly for all agents (DJINN-10). As our RoAD model with window size 20 (RoAD-20) predicts 10 partially denoised future steps as well, we believe DJINN-10 provides a reasonable comparison. DJINN-10 (MPC-X) is a variant of DJINN-10 that has been trained with conditioning augmentation and deployed in an MPC style, enabling us to replan after executing X steps of predictions for all agents.

We first compare RoAD with AR, DJINN, and DJINN-10 (MPC) using standard scene-level displacement metrics such as minSceneADE and minSceneFDE to demonstrate the quality of samples generated by RoAD. We then assess the reactivity of DJINN, DJINN-10 (MPC-1), AR and RoAD with an adversarial agent, which is not controlled by the scene generation model, by measuring the collision rates with the adversarial agent.

Implementation Details We adopt the same transformer architecture from DJINN (Niedoba et al. 2024) for all of our models. We apply 0.2 conditional augmentation for AR and DJINN-10 and 0.5 for RoAD, as we found that higher conditional augmentation ratio for AR and DJINN-10 results in worse performance. We train our RoAD planner with observation length 10 and task ratio β=0.1.

Evaluation Metrics We measure the accuracy of our generated trajectories with standard displacement metrics. To measure the joint motion forecasting quality, we follow (Ngiam et al. 2021) reporting minSceneADE and minSceneFDE. Both metrics capture the minimum joint displacements error for all agents across 6 joint traffic scenario samples. To measure per-agent motion forecasting performance, we report the miss rate; the rate of agents where none of the six predicted trajectories have a final displacement error less than 2 meters. To measure the reactivity of each model, we report the collision rate, the number of collisions divided by the total number of simulated scenarios.

Motion Forecasting We compare RoAD with AR and DJINN-10 on the motion forecasting task using the validation set of the INTERACTION dataset (Zhan et al. 2019). We generate three seconds of driving behavior at 10 hertz, conditioned on one second of observations. We consider the performance of DJINN as the upper bound for this task since DJINN is trained only at this fixed time horizon and is not an autoregressive model by nature. Following (Niedoba et al. 2024), displacement metrics are calculated by generating 24 samples for each scenario and fit a 6 component Gaussian mixture model to cover all future modes. DJINN-10 (MPC-1) achieves slightly better results than AR but performs worse than RoAD due to larger accumulated errors from replanning at each simulation step.

RoAD models with window sizes of 15 (RoAD-15) and 20 (RoAD-20) achieved lower displacement metrics than AR models, as RoAD also considers the noisy future steps beyond the next immediate one. Additionally, RoAD models exhibit a slightly lower miss rate. We also observed that RoAD models with larger window sizes demonstrate better displacement metrics. Displacement metrics are one indicator of the quality of the generated samples. We show qualitatively in Figure 3 that RoAD reconstructs to ground truth trajectories marked in grey better than AR.

Reactivity The RoAD models efficiently roll out long scenarios by partially denoising future states. However, this limits its ability to adapt to perturbations, such as an agent controlled by a different motion planner while being observed by our model. This is a typical setup in closed-loop simulation. To evaluate this, we evaluate the RoAD models' adaptation to an adversarial agent. We select one agent per scene and control it using its replay log, slowing it down to reach only half its trajectory by the end of the simulation. The simulation runs for 40 time steps at 10 Hz, given initial 10-step observations, which makes the performance of DJINN one-shot a lower bound since it is blind to the adversarial agent during the simulation.

In total, we select 1,440 scenes from the INTERACTION validation set, focusing on the top six locations with the largest number of scenarios. Scenes with a low number of participants are filtered out, as agents in these scenarios are less likely to interact. We take three samples per scenario and for each model, then report the average collision rate in Table 1. In addition, we reported the prediction time for a single sample on a RTX2080Ti GPU in Table 2 for each model to highlight the efficiency of our RoAD models.

DJINN-10 (MPC-1) achieves the lowest collision rate compared to other models while having the longest total prediction time. AR reduces the collision rate by 3x compared to the lower bound (DJINN). DJINN-10 (MPC-1) achieving better results than AR aligns with our expectations, as DJINN-10 (MPC-1)can look ahead ten steps into the future, whereas AR predicts only one step ahead. Our proposed RoAD-15 performs close to AR which reduces the collision rate over 2.5x compared to DJINN while having half of the prediction time compared to AR and DJINN-10 (MPC-1).

As an ablation, we measured the collision rate for RoAD-20 in the reactivity experiment. We observed that increasing the window size can reduce the prediction time further while having a slightly higher collision rate. Figure S1 in the Supplementary Materials shows an example where RoAD-20 failing to react, while RoAD-15 avoids a collision in the same scenario. This feature provides practitioners with the

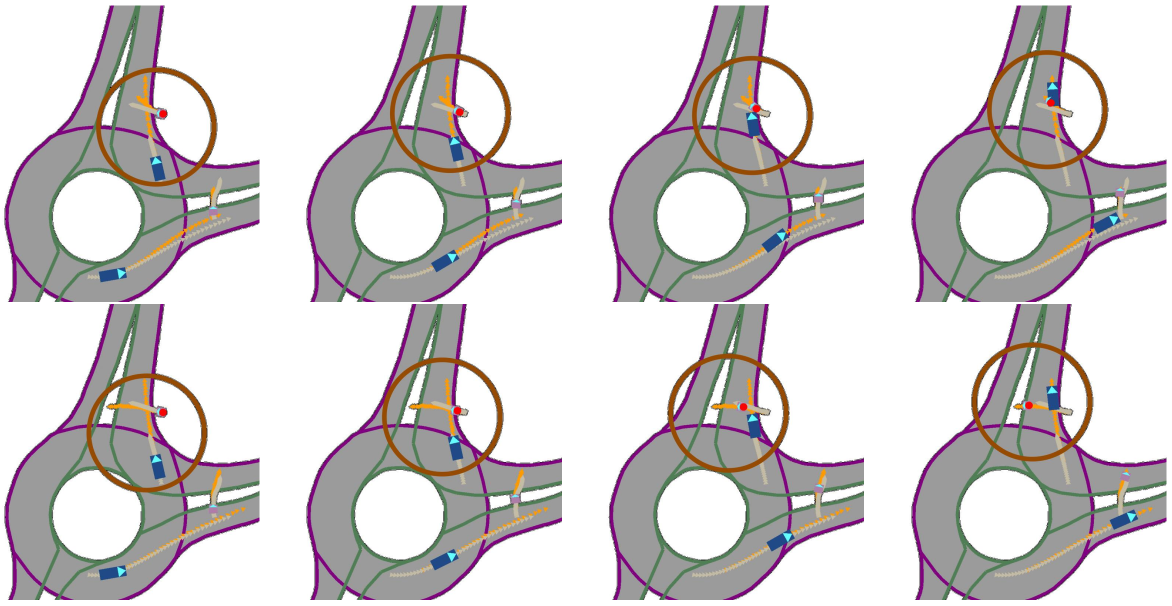
Figure 3: From Top to Bottom row, AR, RoAD-20. By looking ahead of the subsequent step, the pedestrian marked with a red dot controlled by RoAD-20 planner avoided colliding with the vehicle. Brown circles highlight the interaction region. Grey trajectories denote replay logs and orange trajectories are the full predicted future. This example demonstrates that RoAD-20, with a longer planning horizon compared to AR can anticipate and mitigate interactions with other agents effectively.

Table 1: Comparison of displacement metrics and collision rate.

Location		**All**		
Model	**Type**	**minSceneADE**	**minSceneFDE**	**Miss Rate**
DJINN	(One shot)	0.388	1.004	0.049
DJINN-10	(MPC-1)	0.692	1.675	0.166
AR		0.695	1.670	0.168
RoAD-15		0.673	1.596	0.160
RoAD-20		**0.654**	**1.553**	**0.142**

Table 2: Performance with an adversarial ego agent.

Model	**Collision Rate**	**Prediction time** (min)
DJINN	0.052	0.07
DJINN-10 (MPC-1)	**0.014**	0.69
AR	0.016	0.68
RoAD-15	0.019	0.34
RoAD-20	0.024	**0.20**

flexibility to decide whether reactivity or computational efficiency is more important for their simulation needs.

Ablation on conditioning augmentation We show the significance of conditioning augmentation by measuring the displacement metrics for two RoAD models trained with same configuration but one without conditioning augmentation in Table 4. We can see the displacement errors increased significantly without conditioning augmentation.

Table 3: Ablation on conditioning argumentation (CA) across six locations from INTERACTION dataset.

Metrics	RoAD w/o CA	RoAD w CA
minSceneADE	0.930	**0.663**
minSceneFDE	2.197	**1.579**
ego_minADE	0.693	**0.475**

Conclusion

In conclusion, we have proposed a rolling diffusion-based traffic scene planning framework that strikes a beneficial compromise between reactivity and computational efficiency. We believe this work addresses a gap in the community by enabling the autoregressive generation of traffic scenarios for all agents jointly, and it offers insights into the crucial role of conditioning augmentation techniques. For future work, we aim to explore test-time conditioning with this model and seek to enhance model performance through flexible conditioning on past observations (Harvey et al. 2022).

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, Inverted AI, MITACS, and Google. This research was enabled in part by technical support and computational resources provided by the Digital Research Alliance of Canada Compute Canada (alliancecan.ca), the Advanced Research Computing at the University of British Columbia (arc.ubc.ca), and Amazon.

References

Austin, J.; Johnson, D. D.; Ho, J.; Tarlow, D.; and Van Den Berg, R. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34: 17981–17993.

Chai, Y.; Sapp, B.; Bansal, M.; and Anguelov, D. 2019. MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction. In Kaelbling, L. P.; Kragic, D.; and Sugiura, K., eds., *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings*, volume 100 of *Proceedings of Machine Learning Research*, 86–99. PMLR.

Chang, W.-J.; Pittaluga, F.; Tomizuka, M.; Zhan, W.; and Chandraker, M. 2023. Controllable Safety-Critical Closed-loop Traffic Simulation via Guided Diffusion. *arXiv preprint arXiv:2401.00391*.

Cui, H.; Radosavljevic, V.; Chou, F.; Lin, T.; Nguyen, T.; Huang, T.; Schneider, J.; and Djuric, N. 2019. Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks. In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*, 2090–2096. IEEE.

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

Guo, Z.; Gao, X.; Zhou, J.; Cai, X.; and Shi, B. 2023. SceneDM: Scene-level multi-agent trajectory generation with consistent diffusion models. *arXiv preprint arXiv:2311.15736*.

Han, B.; Peng, H.; Dong, M.; Ren, Y.; Shen, Y.; and Xu, C. 2024. AMD: Autoregressive Motion Diffusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 2022–2030.

Harvey, W.; Naderiparizi, S.; Masrani, V.; Weilbach, C.; and Wood, F. 2022. Flexible diffusion modeling of long videos. *Advances in Neural Information Processing Systems*, 35: 27953–27965.

Ho, J.; Chan, W.; Saharia, C.; Whang, J.; Gao, R.; Gritsenko, A.; Kingma, D. P.; Poole, B.; Norouzi, M.; Fleet, D. J.; et al. 2022a. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Ho, J.; Saharia, C.; Chan, W.; Fleet, D. J.; Norouzi, M.; and Salimans, T. 2022b. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47): 1–33.

Hoogeboom, E.; Gritsenko, A. A.; Bastings, J.; Poole, B.; Berg, R. v. d.; and Salimans, T. 2021. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*.

Janner, M.; Du, Y.; Tenenbaum, J. B.; and Levine, S. 2022. Planning with Diffusion for Flexible Behavior Synthesis. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvári, C.; Niu, G.; and Sabato, S., eds., *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, 9902–9915. PMLR.

Jiang, C.; Cornman, A.; Park, C.; Sapp, B.; Zhou, Y.; Anguelov, D.; et al. 2023. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9644–9653.

Karras, T.; Aittala, M.; Aila, T.; and Laine, S. 2022. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577.

Kingma, D.; Salimans, T.; Poole, B.; and Ho, J. 2021. Variational diffusion models. *Advances in neural information processing systems*, 34: 21696–21707.

Liu, Y.; Lioutas, V.; Lavington, J. W.; Niedoba, M.; Sefas, J.; Dabiri, S.; Green, D.; Liang, X.; Zwartsenberg, B.; Ścibior, A.; et al. 2023. Video Killed the HD-Map: Predicting Multi-Agent Behavior Directly From Aerial Images. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 3261–3267. IEEE.

Nayakanti, N.; Al-Rfou, R.; Zhou, A.; Goel, K.; Refaat, K. S.; and Sapp, B. 2023. Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, 2980–2987. IEEE.

Ngiam, J.; Caine, B.; Vasudevan, V.; Zhang, Z.; Chiang, H.-T. L.; Ling, J.; Roelofs, R.; Bewley, A.; Liu, C.; Venugopal, A.; et al. 2021. Scene transformer: A unified architecture for predicting multiple agent trajectories. *arXiv preprint arXiv:2106.08417*.

Niedoba, M.; Lavington, J.; Liu, Y.; Lioutas, V.; Sefas, J.; Liang, X.; Green, D.; Dabiri, S.; Zwartsenberg, B.; Scibior, A.; et al. 2024. A Diffusion-Model of Joint Interactive Navigation. *Advances in Neural Information Processing Systems*, 36.

Pronovost, E.; Ganesina, M. R.; Hendy, N.; Wang, Z.; Morales, A.; Wang, K.; and Roy, N. 2023. Scenario Diffusion: Controllable driving scenario generation with diffusion. *Advances in Neural Information Processing Systems*, 36: 68873–68894.

Rempe, D.; Philion, J.; Guibas, L. J.; Fidler, S.; and Litany, O. 2022. Generating Useful Accident-Prone Driving Scenarios via a Learned Traffic Prior. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ruhe, D.; Heek, J.; Salimans, T.; and Hoogeboom, E. 2024. Rolling Diffusion Models. *arXiv preprint arXiv:2402.09470*.

Ścibior, A.; Lioutas, V.; Reda, D.; Bateni, P.; and Wood, F. 2021. Imagining the road ahead: Multi-agent trajectory prediction via differentiable simulation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 720–725. IEEE.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, 2256–2265. PMLR.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; et al. 2020. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2446–2454.

Suo, S.; Regalado, S.; Casas, S.; and Urtasun, R. 2021. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10400–10409.

Tashiro, Y.; Song, J.; Song, Y.; and Ermon, S. 2021. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34: 24804–24816.

Treiber, M.; Hennecke, A.; and Helbing, D. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2): 1805.

Uria, B.; Murray, I.; and Larochelle, H. 2014. A deep and tractable density estimator. In *International Conference on Machine Learning*, 467–475. PMLR.

Wu, T.; Fan, Z.; Liu, X.; Zheng, H.-T.; Gong, Y.; Jiao, J.; Li, J.; Guo, J.; Duan, N.; Chen, W.; et al. 2024. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36.

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

Yin, W.; Tu, R.; Yin, H.; Kragic, D.; Kjellström, H.; and Björkman, M. 2023. Controllable Motion Synthesis and Reconstruction with Autoregressive Diffusion Models. In *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 1102–1108. IEEE.

Zhan, W.; Sun, L.; Wang, D.; Shi, H.; Clausse, A.; Naumann, M.; Kümmerle, J.; Königshof, H.; Stiller, C.; de La Fortelle, A.; and Tomizuka, M. 2019. INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088 [cs, eess]*.

Zhang, Z.; Liu, R.; Aberman, K.; and Hanocka, R. 2023. TEDi: Temporally-entangled diffusion for long-term motion synthesis. *arXiv preprint arXiv:2307.15042*.

Zhao, T.; Xu, Y.; Monfort, M.; Choi, W.; Baker, C.; Zhao, Y.; Wang, Y.; and Wu, Y. N. 2019. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 12126–12134.

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

Supplementary Materials

Introduction

We provide an extended discussion on related work regarding autoregressive diffusion models. We also detail the computational resources and datasets used in our experiments. Furthermore, we present additional results on accumulation errors caused by replanning for DJINN-10, as well as visualizations showcasing the reactivity of the RoAD model with varying window sizes.

Additional Related Work

Autoregressive Diffusion Models (ARDM) (Hoogeboom et al. 2021) introduce an order-agnostic autoregressive diffusion model that combines an order-agnostic autoregressive model (Uria, Murray, and Larochelle 2014) with a discrete diffusion model (Austin et al. 2021). The order-agnostic nature of this model eliminates the need for generating subsequent predictions in a specific order, thereby enabling faster prediction times through parallel sampling. Additionally, relaxing the causal assumption leads to a more efficient per-time-step loss function during training. However, such a model is not suitable for our application due to the sequential nature of traffic simulation. AMD (Han et al. 2024) proposes an auto-regressive motion generation approach for human motion given a text prompt, but unlike the Rolling Diffusion Model, it denoises one clean motion sample at a time, which is slow at prediction time. The Rolling Diffusion Model (RDM) (Ruhe et al. 2024) proposes a sliding window approach targeted at long video generation but does not specifically study its application in a multi-agent system, particularly for closed-loop traffic simulation. We investigate the level of reactivity when applying rolling diffusion models as a traffic scene planner.

Compute resources

We run all our experiments on four NVIDIA V100 GPUs hosted by a cloud provider. We trained our RoAD models for 9 days, and so 36 GPU-days. AR and DJINN were also trained for 36 GPU-days. In total, including preliminary runs and ablations, we estimate that the project required roughly 300 GPU-days.

Dataset

We experiment with the INTERACTION dataset (Zhan et al. 2019) which is available for non-commercial use following the guidelines at https://interaction-dataset.com/.

Accumulation Errors for DJINN-10

We observe that DJINN-10, even when trained with conditional augmentation, still experiences accumulation errors caused by autoregressive replanning. In Table 4, we compare the displacement error of DJINN-10 at a replanning rate of 10 Hz (MPC-1) and 2 Hz (MPC-5) for a prediction horizon of 40 with an observation length of 10. DJINN-10 (MPC-1) exhibits significantly higher displacement error. In contrast, RoAD utilizes a sliding window approach with decreasing SNR ratio within the window. Denoising for the next simulation step does not start from Gaussian noise, resulting in lower accumulation errors compared to DJINN-10 (MPC-1).

Table 4: Accumulation Errors caused by replanning for DJINN-10

Metrics	DJINN-10 (MPC-1)	DJINN-10 (MPC-5)
minSceneADE	0.692	**0.583**
minSceneFDE	1.675	**1.351**
Miss Rate	0.166	**0.091**

Additional Visualizations

In Figure 4, we demonstrate that the flexibility of our RoAD model by adjusting the sliding window size.

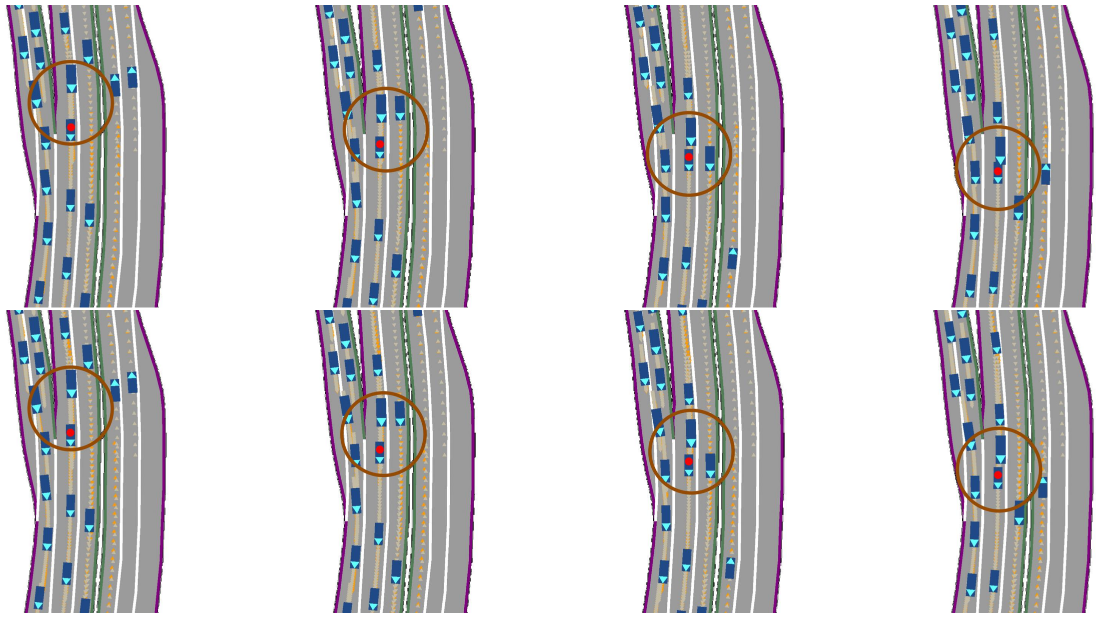
Figure 4: From top to bottom row: RoAD-20, RoAD-15. The adversarial agent, marked with a red dot, follows its replay log and slows down to reach only half its trajectory by the end of the simulation. Brown circles highlight the interaction region. RoAD-15 achieves better reactivity than RoAD-20, as reducing the window size causes the model to denoise the next element from a lower signal-to-noise ratio (SNR), which provides the model with greater flexibility to adjust to the adversarial agent.