

MDG: Masked Denoising Generation for Multi-Agent Behavior Modeling in Traffic Environments

Zhiyu Huang, Zewei Zhou, Tianhui Cai, Yun Zhang, Jiaqi Ma
University of California, Los Angeles

{zhiyuh, zeweizhou, tianhui, yun666, jiaqima}@ucla.edu

Abstract

*Modeling realistic and interactive multi-agent behavior is critical to autonomous driving and traffic simulation. However, existing diffusion and autoregressive approaches are limited by iterative sampling, sequential decoding, or task-specific designs, which hinder efficiency and reuse. We propose **Masked Denoising Generation (MDG)**, a unified generative framework that reformulates multi-agent behavior modeling as the reconstruction of independently noised spatiotemporal tensors. Instead of relying on diffusion time steps or discrete tokenization, MDG applies continuous, per-agent and per-timestep noise masks that enable localized denoising and controllable trajectory generation in a single or few forward passes. This mask-driven formulation generalizes across open-loop prediction, closed-loop simulation, motion planning, and conditional generation within one model. Trained on large-scale real-world driving datasets, MDG achieves competitive closed-loop performance on the Waymo Sim Agents and nuPlan Planning benchmarks, while providing efficient, consistent, and controllable multi-agent trajectory generation. These results position MDG as a simple yet versatile paradigm for multi-agent behavior modeling.*

1. Introduction

Multi-agent behavior modeling is a cornerstone for enabling safe and interactive autonomous systems in complex real-world environments [25]. Accurate and controllable behavior generation supports a range of downstream tasks, from open-loop motion prediction [36, 60, 63] to traffic simulation [35, 56] and closed-loop planning [5, 47]. These tasks are typically addressed using distinct models and objectives: prediction emphasizes accuracy and diversity, simulation demands controllability and interactivity, and planning prioritizes consistency and efficiency. This separation prevents models from generalizing or being reused across tasks [18, 22, 57], hindering scalable autonomy development.

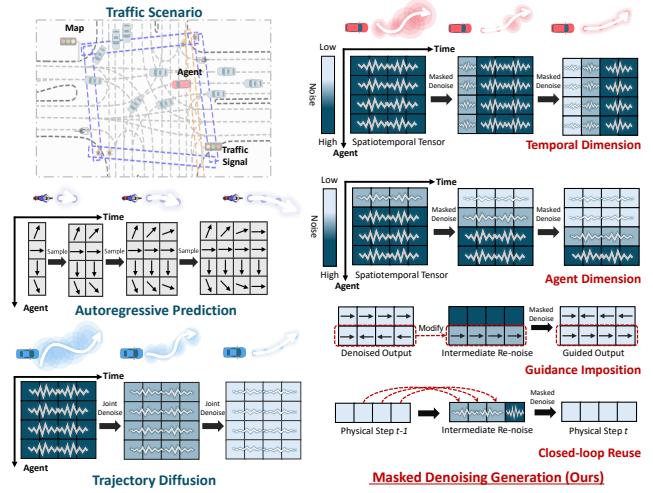


Figure 1. Comparison of MDG with existing trajectory generation paradigms. MDG denoises masked spatiotemporal tensors under varied noise-masking patterns, enabling temporal-wise, agent-wise generation, guided conditioning, and closed-loop reuse. Unlike autoregressive models, MDG predicts full-sequence multi-agent futures in a single step, and unlike joint trajectory diffusion, it supports fine-grained control with efficient, flexible sampling.

Recent advances in generative modeling have improved realism and diversity by learning joint scene-level distributions over agent behaviors [43, 44, 55]. Two major paradigms have emerged: diffusion-based [2, 58] and autoregressive (AR) approaches [32, 49, 64]. However, both paradigms exhibit inherent limitations. Joint trajectory diffusion models [18, 21, 59] operate through iterative noise removal at the scene level, which limits controllability, increases computational cost when guidance is used, and may produce out-of-distribution actions as denoising steps accumulate [18]. AR generative models, which discretize continuous states or actions into tokens and rely on sequential sampling, struggle to capture continuous dynamics and enforce long-term guidance for specific agents. Recent extensions such as Diffusion Forcing [3] and Masked Diffusion [33, 37] introduce per-element noise or partial mask-

ing, but still rely on diffusion time-stepping or categorical data. Consequently, existing approaches struggle to generate multi-agent behaviors that are efficient, consistent, and easily controllable across diverse tasks.

To address these limitations, we propose **Masked Denoising Generation (MDG)**, a continuous, structured, and noise mask-driven generative paradigm for multi-agent behavior modeling. MDG reformulates multi-agent trajectory generation as the reconstruction of independently noised spatiotemporal tensors rather than as iterative diffusion or token decoding. Each agent-time position receives an independent mask intensity that determines how much noise is applied, enabling the model to perform localized denoising and targeted conditioning within a single or a few forward passes. This design combines the controllability and efficiency of masked modeling with the expressiveness of continuous denoising, while avoiding the diffusion time axis [42]. As illustrated in Fig. 1, MDG supports several inference modes, such as temporal or agent-wise denoising, conditional guidance, and closed-loop reuse. By introducing a continuous mask field over structured multi-agent trajectories, MDG provides a simple yet general framework to perform open-loop prediction, interactive simulation, and motion planning. The main contributions of this paper are summarized as follows:

1. We introduce Masked Denoising Generation (MDG), a generative paradigm that models multi-agent behaviors through mask-based denoising of spatiotemporal tensors, supporting diverse behavior modeling tasks.
2. We propose a per-agent, per-timestep mask field that regulates localized denoising and enables flexible inference modes, including temporal-wise, agent-wise, condition-guided generation, and closed-loop reuse.
3. We demonstrate that MDG is a general framework for behavior modeling, achieving competitive closed-loop results on Waymo Sim Agents and nuPlan benchmark, while supporting efficient and controllable generation.

2. Related Work

Multi-agent Behavior Modeling in Traffic Scenarios. Modeling the joint behaviors of multiple interacting agents is a central challenge in autonomous driving [1, 14, 15, 19, 20, 27, 34]. Traditional predictive methods, such as MTR [40] and GameFormer [16], directly decode future trajectories from historical states, achieving accurate but often marginalized predictions that fail to capture inter-agent dependencies. As generative modeling advances, recent approaches have shifted toward learning scene-level distributions over all agents’ futures, enabling richer and more interactive behaviors. Models such as MotionLM [38] and MotionDiffuser [21] improve behavioral diversity but remain limited to pairwise or partially joint dependencies. More recent generative frameworks, including autoregres-

sive Transformers (*e.g.*, Trajeglish [32], BehaviorGPT [64], SMART [49]) and diffusion-based models (*e.g.*, VBD [18], SceneDiffuser [22]), have advanced multi-agent prediction and simulation. Advanced training methods, such as closed-loop fine-tuning [53] and reinforcement fine-tuning [17, 31] have further enhanced performance. Despite this progress, both AR and diffusion paradigms have intrinsic limitations. Autoregressive models rely on discrete sequential decoding, which hinders long-horizon guidance and temporal coherence, while diffusion-based models require slow iterative sampling and struggle with fine-grained control. MDG unifies these paradigms by replacing sequential or iterative generation with a continuous, mask-conditioned denoising process that reconstructs the spatiotemporal tensor in a single step. This formulation enables MDG to produce consistent, diverse, and controllable multi-agent futures.

Trajectory Diffusion Models. Diffusion models [13, 24, 41] have demonstrated strong performance in generating continuous and temporally consistent behaviors [46, 61]. In autonomous driving, diffusion models have been applied to trajectory prediction (*e.g.*, MotionDiffuser [22]), scene simulation (*e.g.*, SceneDiffuser [22], SceneDM [12], VBD [18]), and planning (*e.g.*, Gen-Drive [17], Diffusion-ES [51], Diffusion-Planner [57]). However, existing approaches typically apply uniform noise across entire multi-agent sequences and rely on iterative denoising or integration of ordinary differential equations for sample generation, which increases computational cost and yields limited benefits under strong conditioning tasks (*e.g.*, behavior generation in a traffic context). Recent flow-based variants for trajectory generation, such as Leapfrog [29], TrajFlow [50], and MoFlow [9], still depend on such iterative processes. In contrast, MDG performs single-step masked reconstruction with per-timestep, per-agent noise, enabling localized conditioning and efficient generation while maintaining the expressive capacity.

Masked Generative Modeling. Masked generative modeling has recently shown strong scalability across modalities such as language [33, 37], video [28], and images [10]. These approaches train models to reconstruct masked inputs, offering an efficient alternative to stepwise generation. Inspired by this, MDG adopts a masked denoising formulation, introducing independent per-token noise levels for localized reconstruction and adaptive conditioning. Masked modeling has been explored in trajectory prediction and generation (*e.g.*, masked trajectory model [48] and Forecast-MAE [4]), which demonstrate that randomized masking improves robustness and inference flexibility. MDG generalizes these ideas by introducing a continuous mask field over structured spatiotemporal tensors, combining the efficiency of masked modeling with the expressive power of denoising.

3. Masked Denoising Generation

3.1. Preliminary

Diffusion Models. Diffusion models [13] are a family of generative models that learn to recover structured data from noise through iterative denoising. Let $q(\mathbf{x})$ represent the data distribution, and $\mathbf{x} \sim q(\mathbf{x})$ denote clean data sampled from this distribution. The forward diffusion process gradually adds Gaussian noise over K steps:

$$\mathbf{z}_k = \sqrt{\bar{\alpha}_k} \mathbf{x} + \sqrt{1 - \bar{\alpha}_k} \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}), \quad (1)$$

where $\bar{\alpha}_k$ is a noise schedule controlling the magnitude of noise at step k . The reverse process learns to sequentially denoise \mathbf{z}_k to recover the original data. In our context, \mathbf{x} is structured as a spatiotemporal tensor x_a^t with temporal axis t and agent axis a .

Diffusion Forcing. Diffusion forcing [3] extends standard diffusion by assigning an independent noise level to each element in a sequence rather than a single global timestep. This allows the model to treat some elements as nearly clean while others remain highly corrupted, effectively interpolating between next-token prediction and full-sequence diffusion. However, it still relies on multi-step denoising and is primarily applied to one-dimensional token sequences (*e.g.*, text or video tokens), without modeling of structured spatiotemporal data.

Masked Discrete Diffusion. Masked diffusion [33, 37] represents another form of partial corruption, typically used in discrete domains. These methods replace missing elements with a [MASK] token and train a model to reconstruct them from the visible context. While computationally efficient, they operate on categorical data and cannot handle continuous-valued signals or partially noised inputs.

3.2. Masked Denoising

We propose Masked Denoising Generation (MDG), a continuous spatiotemporal generalization of masked and diffusion-based paradigms. The key idea is to represent noise as a *continuous mask field* applied independently to each agent-time position in the trajectory tensor. This allows single-step or few-step denoising, localized conditioning, and controllable scene generation.

- Compared to standard diffusion, MDG supports single-step reconstruction for efficient prediction and also admits controlled iterative refinement when required.
- Compared to diffusion forcing, MDG generalizes per-element noise to continuous, structured spatiotemporal tensors with per-agent and per-timestep masking, enabling more flexible control.
- Compared to masked discrete diffusion, MDG represents masking as continuous Gaussian corruption (soft masks) instead of discrete mask tokens, permitting partial corruption and continuous outputs.

Mask-driven Corruption Process. Each element of the trajectory tensor is assigned a noise level (mask intensity) indicating how strongly it should be perturbed. Let $\mathbf{m} \in [0, K]^{T \times N}$ (N agents, T timesteps) be a noise-level mask, where m_a^t specifies the noise magnitude applied to position x_a^t . The forward-noising process is defined as:

$$\mathbf{z} = \sqrt{\alpha(\mathbf{m})} \odot \mathbf{x} + \sqrt{1 - \alpha(\mathbf{m})} \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}), \quad (2)$$

where \mathbf{z} is the noised states and $\alpha : [0, K] \rightarrow [0, 1]$ maps noise level values to corresponding noise scales, where K is the maximum noise level. During training, MDG samples noise masks across agent-time positions, and the denoiser \mathcal{D} learns to reconstruct the clean trajectories from the corresponding noised inputs:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}, \mathbf{m}, \epsilon} \left[\|\mathcal{D}(\sqrt{\alpha(\mathbf{m})} \odot \mathbf{x} + \sqrt{1 - \alpha(\mathbf{m})} \odot \epsilon, \mathbf{m}) - \mathbf{x}\|^2 \right], \quad (3)$$

where $\mathbf{m} \sim p_{\text{mask}}$ is sampled from a predefined distribution.

Generation via Masked Denoising. In inference, MDG follows a predefined denoising schedule $\{\bar{\mathbf{m}}_L, \bar{\mathbf{m}}_{L-1}, \dots, \bar{\mathbf{m}}_0\}$, where L is the number of denoising steps. Each mask $\bar{\mathbf{m}}_\ell$ defines the desired noise level for every spatiotemporal position (agent and timestep) at step ℓ . The process begins from a fully noised state $\mathbf{z}_L \sim \mathcal{N}(0, \mathbf{I})$ corresponding to the highest noise mask $\bar{\mathbf{m}}_L$, and progressively reduces the noise level toward $\bar{\mathbf{m}}_0$, representing the clean output.

At step ℓ , the model receives the current noisy state \mathbf{z}_ℓ and the mask $\bar{\mathbf{m}}_\ell$, and produces a clean estimate $\hat{\mathbf{x}}_\ell = \mathcal{D}(\mathbf{z}_\ell, \bar{\mathbf{m}}_\ell)$. If iterative refinement is desired, the clean estimate is then re-noised according to the next mask in the schedule for the subsequent denoising step:

$$\mathbf{z}_{\ell-1} = \sqrt{\alpha(\bar{\mathbf{m}}_{\ell-1})} \odot \hat{\mathbf{x}}_\ell + \sqrt{1 - \alpha(\bar{\mathbf{m}}_{\ell-1})} \odot \epsilon. \quad (4)$$

At each iteration, the model predicts a clean reconstruction, which is then re-noised to the level specified by $\bar{\mathbf{m}}_{\ell-1}$ before the next call. After the final iteration, the model outputs \mathbf{z}_0 , the fully denoised trajectory ready for use.

3.3. Problem Formulation

We consider a multi-agent traffic scene involving N agents observed over a past temporal window H . The historical trajectories of all agents are denoted by $\mathcal{A} = \{x_a^t\}_{a=1:N}^{t=-H:0}$, where $t = 0$ corresponds to the current timestep. The environmental context, including high-definition map features and traffic signal states, is represented by \mathcal{M} and \mathcal{S} , respectively. These components form the scene context $\mathbf{c} = \{\mathcal{A}, \mathcal{M}, \mathcal{S}\}$. Given \mathbf{c} , the goal is to predict the joint future trajectories of all agents over a horizon T : $\mathbf{x} = \{x_a^t\}_{a=1:N}^{t=1:T}$. Formally, the model learns a conditional generative mapping $p_\theta(\mathbf{x}|\mathbf{c})$. This unifies multiple downstream tasks: **Open-loop prediction:** the model generates

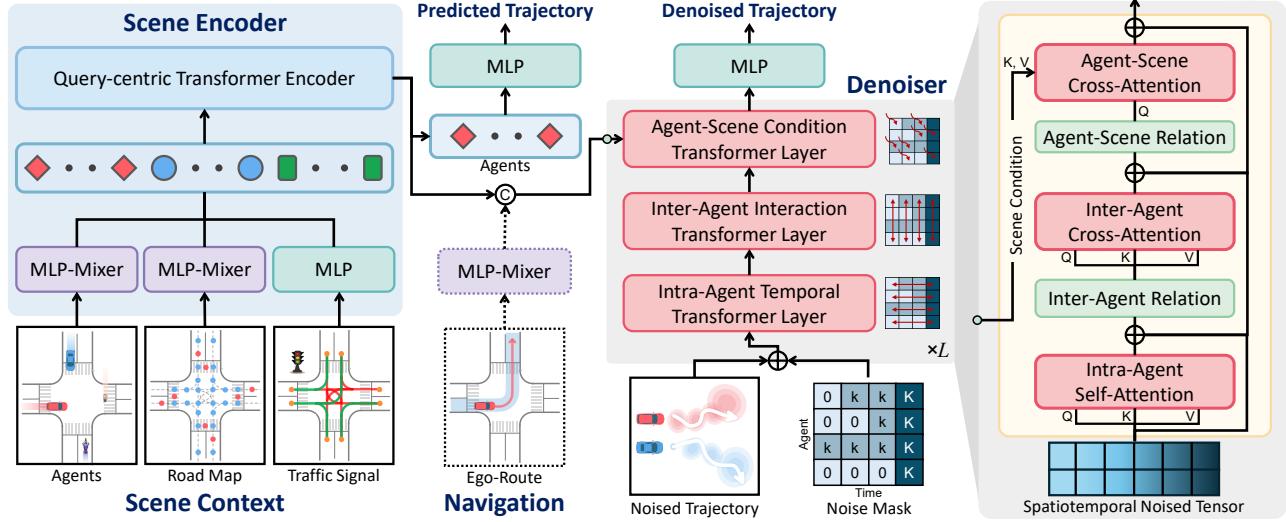


Figure 2. Overview of the MDG model structure. The Scene Encoder integrates scene context, including agent states, map poly-lines, and traffic lights, using modality-specific networks and a query-centric Transformer to produce a unified scene representation. An auxiliary MLP head decodes from the representation of agents to predict trajectories as regularization. The ego route poly-lines are encoded via an MLP-Mixer network. The Denoiser processes mask-conditioned noised spatiotemporal trajectory tensor through stacked Transformer blocks with: intra-agent temporal self-attention, inter-agent interaction cross-attention, and agent-scene condition cross-attention, where only the ego agent attends to its route context for planning tasks. A final MLP head outputs clean, denoised trajectories.

the entire future sequence \mathbf{x} in a single or few denoising steps; **Multi-agent simulation**: only a short segment of the predicted multi-agent future is executed before re-querying the model with updated agent histories; **Closed-loop Planning**: the model conditions generation on constraint, allowing consistent ego-agent motion planning.

3.4. Model Structure

The MDG model is based on a Transformer encoder-decoder architecture, designed to denoise spatiotemporal tensors and fuse scene context for multi-agent trajectory generation. Fig. 2 provides an overview of the model structure, and a detailed description is provided as follows. Additional details are provided in the supplementary material.

Scene Encoder. MDG encodes a multimodal scene context comprising agent states $\mathcal{A} \in \mathbb{R}^{N \times H \times D_a}$, vectorized map poly-lines $\mathcal{M} \in \mathbb{R}^{N_m \times N_w \times D_w}$, and traffic light states $\mathcal{S} \in \mathbb{R}^{N_s \times D_s}$. For planning tasks, we additionally condition on the ego agent’s navigation route $\mathcal{R} \in \mathbb{R}^{N_r \times N_w \times D_w}$. Here, N_m , N_r , N_w , and N_s denote the numbers of road map poly-lines, ego-route poly-lines, waypoints, and traffic lights, respectively; D_a , D_w , and D_s represent their corresponding feature dimensions. To process these inputs, MDG adopts three modality-specific MLP-Mixer encoders [45] for agents, maps, and optional ego-routes, and an MLP encoder for traffic lights. Temporal features in \mathcal{A} and spatial features in \mathcal{M} and \mathcal{R} are aggregated through adaptive max-pooling along the temporal and waypoint axes, producing compact yet informative latent embeddings. The resulting unified scene representation $\mathcal{C} \in \mathbb{R}^{(N+N_m+N_s) \times D}$

encodes all entities in a shared latent space. Then, query-centric Transformer layers [18, 40, 62] model high-order dependencies among agents, maps, and signals, yielding refined context features $\mathcal{C}' \in \mathbb{R}^{(N+N_m+N_s) \times D}$. For the ego agent, route encoding is concatenated to preserve navigation intent. To stabilize encoder learning, an auxiliary MLP head decodes \mathcal{C}' to predict future trajectories, providing regularization signals.

Denoiser. The denoiser operates on a noised future trajectory tensor $\mathbf{z} \in \mathbb{R}^{N \times T \times D_z}$ and its associated noise mask $\mathbf{m} \in \mathbb{N}_0^{N \times T}$, which specifies per-timestep, per-agent noise magnitudes. Each token in \mathbf{z} represents an action (acceleration, yaw rate), which is propagated through a differentiable motion model into continuous physical states (x, y, θ, v). The resulting state tensor is encoded with an MLP, while the noise mask is embedded separately and fused downstream. The denoiser consists of stacked Transformer blocks that interleave three specialized attention mechanisms: (1) *intra-agent temporal self-attention* to model temporal dependencies within each agent’s trajectory, (2) *inter-agent interaction cross-attention* with learned inter-agent relation encoding to capture multi-agent interactions, and (3) *agent-scene condition cross-attention* with agent-scene relation encoding to inject spatial and semantic context into trajectory refinement. All cross-attention modules employ relative relation encodings to preserve local coordinate invariance. For planning tasks, only the ego agent can access route-conditioned context. The stacked denoising layers progressively reconstruct clean trajectories, followed by an MLP decoder that outputs the final denoised trajectories.

3.5. Training

MDG is trained to reconstruct clean trajectories from arbitrarily noised inputs. The training objective jointly optimizes denoising and auxiliary prediction losses:

$$\mathcal{L} = \mathcal{L}_d + \lambda \mathcal{L}_p, \quad (5)$$

where \mathcal{L}_d is the denoising loss and \mathcal{L}_p is the prediction loss, and λ is the balance weight.

Given the noise mask \mathbf{m} applied to the action sequence, the corrupted input is \mathbf{z} , and its corresponding physical states (derived through the differentiable dynamics model) are represented as $\hat{\mathbf{s}} = g(\mathcal{D}(\mathbf{z}, \mathbf{m}))$. The per-sample denoising loss encourages the model to directly reconstruct clean states for all spatiotemporal positions:

$$\mathcal{L}_d = \frac{1}{NT} \sum_{i=1}^N \sum_{t=1}^T |\hat{\mathbf{s}}_i^t - \mathbf{s}_i^t|^2. \quad (6)$$

Details of the auxiliary prediction loss \mathcal{L}_p are provided in the supplementary material.

We employ a linear α -scheduler that linearly distributes noise variance across discrete noise levels in the range $\alpha \in [0.99, 0.01]$. Higher noise levels correspond to larger variance (lower α), while lower noise levels retain a stronger signal component. This scheduling ensures stable learning across denoising difficulty levels.

Training MDG with strong noise or random noise can obscure inter-agent dependencies and hinder learning. To address this, we introduce an adaptive masking strategy that varies the masking rate δ across samples in a batch and applies noise either along the temporal or agent dimension. For each training sample: if *temporal masking*, a δ fraction of later timesteps for each agent is fully noised, while remaining timesteps receive progressively increasing random noise levels; if *agent masking*, a δ fraction of agents is fully noised, and the rest are assigned lower, uniform noise levels across timesteps. The masking rate δ is uniformly distributed across samples within a batch, ensuring diverse exposure to corruption patterns. This encourages the denoiser to generalize across both temporal degradation and inter-agent corruption, improving its ability to recover structured multi-agent dynamics during inference. Further details are provided in the supplementary material.

3.6. Inference

Our MDG model supports flexible inference through customized denoising strategies tailored to diverse downstream scenarios, as illustrated in Fig. 1.

One-step Denoising. Given the highly conditional nature of traffic-agent interactions, MDG can generate realistic trajectories in a single denoising step ($L = 1$). Starting from

a fully noised mask, the model produces clean rollouts directly. This strategy is well-suited for closed-loop planning, offering better runtime efficiency.

Denoising along Time. Denoising can proceed along the temporal axis, gradually reducing noise over time steps. The denoised granularity can be flexibly adjusted, with noise levels decreasing progressively for traversed timesteps and remaining consistent across agents. This approach is effective for open-loop forecasting, promoting diversity and iterative refinement of multi-agent futures.

Denoising along Agent. Alternatively, denoising can be performed agent-wise, selectively reconstructing subsets of agents at each step. Noise levels are consistent across time for individual agents but vary between agents. This enables conditional behavior prediction and target-agent planning, facilitating interactive and controllable simulation.

Long-horizon Guidance. MDG enables long-horizon control by perturbing modified trajectories (with certain objective functions) with small additive noise and re-denoising. This approach efficiently refines trajectory adjustments while constraining future behaviors within desired bounds, outperforming guided diffusion methods [21, 59].

Closed-loop Result Reuse. To ensure temporal consistency during continuous rollouts, MDG can reuse previous-step results by shifting the latest actions and adding small perturbations. This supports high-frequency planning and reduces distributional drift caused by accumulated errors.

4. Experiments

4.1. Experimental Setup

Datasets. For the simulation and prediction tasks, we employ the Waymo Open Motion Dataset (WOMD) [8], which contains 486,995 training scenarios, each covering 9 seconds of agent trajectories with the corresponding map data. To enable closed-loop simulation, we use the Waymax Simulator [11] for roll-out. For planning evaluation, we adopt the nuPlan dataset [23], which comprises approximately 1,300 hours of real-world driving data. During training, we include all scenario types from the nuPlan dataset while limiting each type to a maximum of 4,000 scenarios, resulting in a total of 176,218 training samples.

Implementation Details. Our MDG model consists of six query-centric Transformer encoder layers and two denoiser blocks, each containing three Transformer layers. The hidden dimension is set to $D = 256$, leading to a total of approximately 10 million parameters. We adopt a five-level noise schedule ($K = 5$), where the schedule parameter α in Eq. (2) is linearly distributed from 0.99 to 0.01. Additional training details are provided in the supplementary material.

Overview. MDG adopts a *unified formulation and modeling* for multi-agent behavior generation, enabling a single model to handle diverse tasks without task-specific adapta-

Table 1. Closed-loop Multi-Agent Simulation Results on the Waymo Sim Agents Benchmark. * denotes results from the 2024 benchmark.

Agent Policy	Realism Meta Metric (\uparrow)	Kinematic Metric (\uparrow)	Interactive Metric (\uparrow)	Map-based Metric (\uparrow)	minADE [m] (\downarrow)
VBD* [18]	0.7200	0.4169	0.7819	0.7207	1.4743
BehaviorGPT* [64]	0.7473	0.4333	0.7997	0.7636	1.4147
SMART-Large* [49]	0.7614	0.4786	0.8066	0.7682	1.3728
DRoPE* [56]	0.7625	0.4779	0.8065	0.7715	1.2626
UniMM [26]	0.7829	0.4914	0.8089	0.9161	1.2949
SMART-CLSFT [53]	0.7846	0.4931	0.8106	0.9177	1.3065
TrajTok [54]	0.7852	0.4887	0.8116	0.9207	1.3179
SMART-R1 [31]	0.7858	0.4944	0.8110	0.9201	1.2885
MDG (1-step, closed-loop)	0.7844	0.4928	0.8099	0.9183	1.3123

tions. Its *one-stage training* and *versatile denoising* facilitate both efficient closed-loop rollout and diverse open-loop prediction. The following experiments demonstrate the *generality and effectiveness* of MDG across a wide range of traffic behavior modeling tasks.

4.2. Closed-loop Tasks

4.2.1. Multi-Agent Simulation

Task Description. The objective of this task is to generate 32 future trajectories for up to 128 agents per scenario, each spanning 8 seconds and conditioned on 1 second of historical context. Agent trajectories are produced in a closed-loop manner using our MDG model in one-step denoising mode, which directly predicts clean samples and executes them within the Waymax simulator. Simulations are performed with a replanning frequency of 1 Hz. We follow the official evaluation protocol of [30], which includes metrics evaluating motion realism, agent interactions, map compliance, and displacement error (minADE). The overall realism meta-metric is then derived as the primary metric.

Results. As shown in Tab. 1, MDG achieves competitive performance on the Waymo Sim Agents Benchmark, closely matching the best results across core metrics, with only marginal differences ($< 0.2\%$). Unlike SMART-based models (AR) such as SMART-R1 and SMART-CLSFT, which rely on multi-stage training and partially open-loop rollouts, MDG operates fully in closed-loop with a single-stage training. These results demonstrate that the proposed one-step denoising mode effectively captures multi-agent dynamics without requiring complex supervision or objectives. Additional closed-loop simulation results are provided in the supplementary material, and a qualitative example is shown in Fig. 3.

4.2.2. Ego-Agent Planning

Task Description. This task evaluates the ability of navigating the ego agent along a predefined route in a closed-loop simulation environment. We evaluate on the nuPlan Val14 [7] and Test14 [5] benchmarks under both non-reactive and reactive agent behavior modes. The *closed-*

Table 2. Closed-loop Motion Planning Results on the nuPlan Val14 and Test14 Benchmarks. NR and R denote simulation with non-reactive and reactive agent settings, respectively. * denotes methods that incorporate prior knowledge or rules about scoring.

Planner	Val14		Test14	
	NR (\uparrow)	R (\uparrow)	NR (\uparrow)	R (\uparrow)
IDM	75.60	77.33	70.39	74.42
GameFormer [16]	79.94	79.78	83.88	82.05
PlanTF [6]	84.27	76.95	85.62	79.58
PLUTO [5]	88.89	78.11	89.90	78.62
Diffusion Planner [57]	89.87	82.80	89.19	82.93
PDM-Closed* [7]	92.84	92.12	90.05	91.63
CarPlanner* [52]	91.45	–	94.07	91.10
MDG (1-step)	88.85	81.32	88.43	81.10
MDG (1-step, reuse)	90.45	83.89	90.16	83.21

loop score (CLS) is computed as the average across all scenarios, incorporating metrics such as Ego Progress, No At-Fault Collisions, and Drivable Area Compliance, where higher values indicate better performance. Each simulation scenario lasts 15 seconds and runs at 10 Hz.

Results. As shown in Tab. 2, MDG demonstrates strong closed-loop planning performance on the nuPlan benchmarks. The *MDG (reuse 1-step)* variant achieves the best overall scores, surpassing the one-step version by selectively reusing ego-agent actions from the previous step, thereby enhancing trajectory consistency. Compared to previous diffusion-based planners, MDG attains higher scores, indicating improved performance in dynamic environments. While methods that incorporate explicit priors or rule-based scoring (*i.e.*, CarPlanner and PDM-Closed) achieve marginally higher scores, MDG delivers comparable performance without relying on such heuristics, highlighting its generality and scalability.

4.3. Open-loop Tasks

4.3.1. Multi-modal Motion Prediction

Task Description. This task evaluates the model’s ability to predict six joint future trajectories for all agents in a scene,

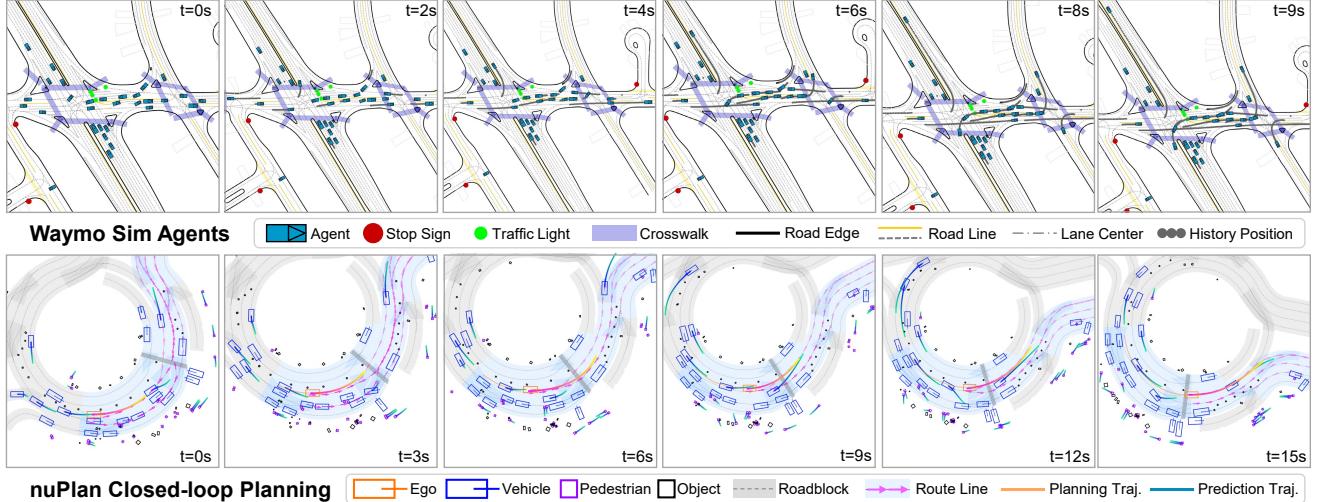


Figure 3. Qualitative results of MDG in closed-loop multi-agent simulation and ego-agent planning tasks. MDG controls all agents in an interactive, map-compliant manner and navigates the ego vehicle effectively in complex scenarios.

with a fixed prediction horizon of 8 seconds. Experiments are conducted on the WOMD validation split. We report three evaluation metrics: *collision rate (CR)* to assess scene-level consistency of predictions; *scene average displacement error (SADE)*, which quantifies the overall accuracy of the predicted trajectories; and *minimum SADE (minSADE)*, which reflects the quality of the most accurate prediction. All metrics are computed over agents labeled as *modeled*. The difference between SADE and minSADE can provide insight into the diversity of the prediction results. We also explore different inference modes for our MDG model, such as denoising along time and agent axes.

Results. Quantitative results in Tab. 3 demonstrate that MDG achieves the best overall performance on the WOMD validation set. The MTR baseline shows the highest collision rate, indicating limited scene-level consistency due to marginal predictions. Both VBD and SMART achieve strong results, while MDG (1-step) achieves the lowest collision rate, and MDG (temporal, 5-step) achieves the lowest minSADE metric. Increasing the number of denoising steps (along temporal or agent dimensions) generally enhances prediction diversity, as reflected by higher SADE and lower minSADE, although excessive denoising steps may still slightly degrade accuracy due to out-of-distribution sampling. Qualitative results in Fig. 4 further illustrate that MDG with five-step temporal denoising generates richer and more diverse multi-agent behaviors compared to the one-step setting. Additional results are provided in the supplementary material.

4.3.2. Controllable Generation

Task Description. This task evaluates controllable scenario generation, where the behaviors of target agents are conditioned on predefined goals. To evaluate both controllability

Table 3. Open-loop Prediction Results on WOMD Validation Set

Method	CR [%] (\downarrow)	SADE (\downarrow)	minSADE (\downarrow)
MTR [40]	9.990	3.171	2.014
VBD [18]	6.427	2.988	2.006
SMART [53]	4.993	2.896	1.849
MDG (1-step)	4.415	2.755	1.951
MDG (time 5-step)	4.875	3.122	1.840
MDG (agent 5-step)	4.641	2.994	1.863
MDG (time 10-step)	5.026	3.233	1.964
MDG (agent 10-step)	4.981	3.107	1.916
MDG (time 20-step)	5.071	3.279	2.050

and scene consistency, we adopt an open-loop generation setting and produce three samples with different random seeds. Specifically, ground-truth goals are assigned to the labeled target agents, and the model generates trajectories for all agents in the scene. For MDG with multi-step guidance, we apply denoising along the agent axis. We use the following evaluation metrics: *collision rate (CR)*, *SADE*, and *goal-reach rate (GR)*. These metrics quantify how effectively the assigned agents achieve the desired target behaviors while preserving the coherence of the scene. We randomly select 1K scenarios from the WOMD validation set for evaluation. The detailed guidance procedure is provided in the supplementary material.

Results. The results in Tab. 4 indicate that the MDG model achieves better controllability and scene consistency compared to the guided VBD model. MDG attains higher goal-reach rates and lower collision rates, indicating a better balance between goal satisfaction and scene consistency. For the MTR model, directly selecting the trajectory closest to the target goal yields the highest GR and lowest SADE, but this heuristic approach significantly degrades scene con-

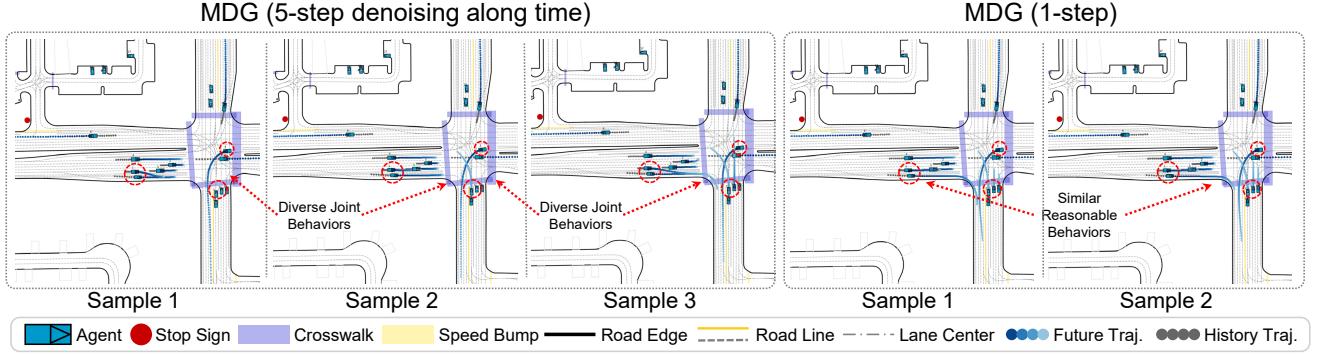


Figure 4. Qualitative results of MDG in multi-agent open-loop prediction. The one-step denoising mode can produce plausible and interactive scenarios, but with limited sample diversity. By using multi-step denoising along the temporal axis, the generated scenarios exhibit greater diversity, and agents display obvious multimodal behaviors.

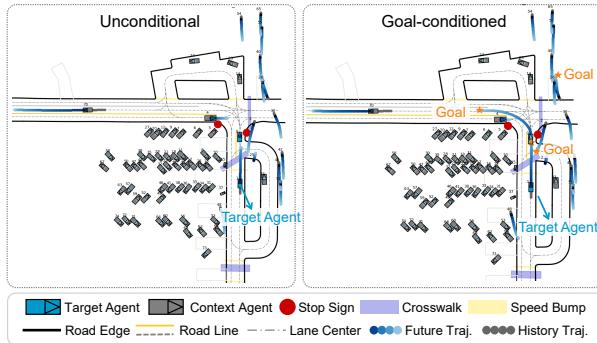


Figure 5. Illustration of the controllable generation task. Goals are assigned to target agents, and MDG guides them to reach the designated goals while maintaining reactions for surrounding agents.

sistency, as reflected in increased collision rates. Among different inference modes, the 5-step guidance offers the best trade-off between controllability and consistency, outperforming 1-step and 10-step settings. Runtime comparisons in Tab. 5 further show that MDG significantly reduces computation overhead compared to diffusion-based (VBD) guidance, highlighting its efficiency for controllable scenario generation.

Table 4. Results on Controllable Scenario Generation

Method	CR [%] (↓)	GR [%] (↑)	SADE (↓)
MTR (goal)	15.13±0.08	69.3±0.05	2.386±0.002
VBD (guide)	9.83±0.09	43.8±0.06	3.406±0.006
MDG (guide 1-step)	5.03±0.20	36.4±0.16	2.886±0.002
MDG (guide 5-step)	5.42±0.03	48.9±0.21	2.919±0.007
MDG (guide 10-step)	6.39±0.26	48.5±0.23	3.357±0.001

4.4. Ablation Study

Effect of Training Noise Masking. We investigate the effect of different noise-masking strategies during MDG training, including random, binary, and multi-level masking. All models are tested with 5-step denoising, and re-

Table 5. Comparison of Runtime Performance (ms, mean ± std)

Model	AR	1-step	5-step	5-step Guidance
SMART	253.8±26.9	–	–	–
VBD	–	282.2±41.8	1155.0±94.6	9076.6±62.3
MDG	–	252.6±49.4	1115.2±45.1	1160.7±48.4

sults are reported in Tab. 6. When random noise levels are used, the model fails to capture consistent spatiotemporal patterns, leading to degraded performance. Binary masking ($K=1$) shows suboptimal performance due to limited masking diversity, while excessive noise levels ($K=10$) add unnecessary complexity without performance gain. Our used configuration ($K=5$) achieves an effective balance between noise variability and testing performance. Additional ablation results are provided in the supplementary materials.

Table 6. Influence of Noise Masking on Open-loop Prediction

Noise Mask	CR [%] (↓)	SADE (↓)	minSADE (↓)
Random	12.477	4.085	3.982
Multi $K = 10$	4.956	3.274	1.956
Multi $K = 5$	4.875	3.122	1.840
Binary $K = 1$	5.635	3.374	2.222

5. Conclusions

We introduce MDG, a masked denoising generative model for multi-agent behavior modeling in traffic scenarios. MDG leverages noise-based masking in a multi-agent spatiotemporal tensor and a Transformer-based denoiser to reconstruct clean samples from arbitrarily masked inputs. MDG achieves competitive performance in closed-loop simulation on the Waymo Sim Agents Benchmark and in planning on the nuPlan benchmark. MDG also proves effective in open-loop tasks such as trajectory prediction and goal-conditioned controllable generation. Future work will focus on improving runtime efficiency and extending MDG to integrate user-provided text or other conditioning inputs.

References

- [1] Mustafa Baniodeh, Kratarth Goel, Scott Ettinger, Carlos Fuertes, Ari Seff, Tim Shen, Cole Gulino, Chenjie Yang, Ghassen Jerfel, Dokook Choe, et al. Scaling laws of motion forecasting and planning—a technical report. *arXiv preprint arXiv:2506.08228*, 2025. 2
- [2] Wei-Jer Chang, Francesco Pittaluga, Masayoshi Tomizuka, Wei Zhan, and Manmohan Chandraker. Safe-sim: Safety-critical closed-loop traffic simulation with diffusion-controllable adversaries. In *European Conference on Computer Vision*, pages 242–258. Springer, 2025. 1
- [3] Boyuan Chen, Diego Martí Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *arXiv preprint arXiv:2407.01392*, 2024. 1, 3
- [4] Jie Cheng, Xiaodong Mei, and Ming Liu. Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8679–8689, 2023. 2
- [5] Jie Cheng, Yingbing Chen, and Qifeng Chen. Pluto: Pushing the limit of imitation learning-based planning for autonomous driving. *arXiv preprint arXiv:2404.14327*, 2024. 1, 6, 14
- [6] Jie Cheng, Yingbing Chen, Xiaodong Mei, Bowen Yang, Bo Li, and Ming Liu. Rethinking imitation-based planners for autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14123–14130. IEEE, 2024. 6
- [7] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning*, pages 1268–1281. PMLR, 2023. 6
- [8] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021. 5
- [9] Yuxiang Fu, Qi Yan, Lele Wang, Ke Li, and Renjie Liao. Moflow: One-step flow matching for human trajectory forecasting via implicit maximum likelihood estimation based distillation. *arXiv preprint arXiv:2503.09950*, 2025. 2
- [10] Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23164–23173, 2023. 2
- [11] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinglei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36, 2024. 5
- [12] Zhiming Guo, Xing Gao, Jianlan Zhou, Xinyu Cai, and Botian Shi. Scenedm: Scene-level multi-agent trajectory generation with consistent diffusion models. *arXiv preprint arXiv:2311.15736*, 2023. 2
- [13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 3
- [14] Yihan Hu, Siqi Chai, Zhenning Yang, Jingyu Qian, Kun Li, Wenxin Shao, Haichao Zhang, Wei Xu, and Qiang Liu. Solving motion planning tasks with a scalable generative model. In *European Conference on Computer Vision*, pages 386–404. Springer, 2024. 2
- [15] Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with transformer-based neural network for autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2605–2611. IEEE, 2022. 2
- [16] Zhiyu Huang, Haochen Liu, and Chen Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3903–3913, 2023. 2, 6
- [17] Zhiyu Huang, Xinshuo Weng, Maximilian Igl, Yuxiao Chen, Yulong Cao, Boris Ivanovic, Marco Pavone, and Chen Lv. Gen-Drive: Enhancing diffusion generative driving policies with reward modeling and reinforcement learning fine-tuning. *arXiv preprint arXiv:2410.05582*, 2024. 2
- [18] Zhiyu Huang, Zixu Zhang, Ameya Vaidya, Yuxiao Chen, Chen Lv, and Jaime Fernández Fisac. Versatile behavior diffusion for generalized traffic agent simulation. *arXiv preprint arXiv:2404.02524*, 2024. 1, 2, 4, 6, 7, 13, 17
- [19] Xiaosong Jia, Liting Sun, Hang Zhao, Masayoshi Tomizuka, and Wei Zhan. Multi-agent trajectory prediction by combining egocentric and allocentric views. In *Conference on Robot Learning*, pages 1434–1443. PMLR, 2022. 2
- [20] Xiaosong Jia, Penghao Wu, Li Chen, Yu Liu, Hongyang Li, and Junchi Yan. Hdgt: Heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding. *IEEE transactions on pattern analysis and machine intelligence*, 45(11):13860–13875, 2023. 2
- [21] Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2023. 1, 2, 5, 16
- [22] Chiyu Max Jiang, Yijing Bai, Andre Cornman, Christopher Davis, Xiukun Huang, Hong Jeon, Sakshum Kulshrestha, John Wheatley Lambert, Shuangyu Li, Xuanyu Zhou, et al. SceneDiffuser: Efficient and controllable driving simulation initialization and rollout. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1, 2
- [23] Napat Karnchanachari, Dimitris Geromichalos, Kok Seang Tan, Nanxiang Li, Christopher Eriksen, Shakiba Yaghoubi, Noushin Mehdipour, Gianmarco Bernasconi, Whye Kit Fong, Yiluan Guo, et al. Towards learning-based planning: The nuplan benchmark for real-world autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 629–636. IEEE, 2024. 5, 16

- [24] Yiheng Li, Heyang Jiang, Akio Kodaira, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Immiscible diffusion: Accelerating diffusion training with noise assignment. *arXiv preprint arXiv:2406.12303*, 2024. 2
- [25] Yiming Li, Zhiheng Li, Nuo Chen, Moonjun Gong, Zonglin Lyu, Zehong Wang, Peili Jiang, and Chen Feng. Multiagent multitraversal multimodal self-driving: Open mars dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22041–22051, 2024. 1
- [26] Longzhong Lin, Xuewu Lin, Kechun Xu, Haojian Lu, Lichao Huang, Rong Xiong, and Yue Wang. Revisit mixture models for multi-agent simulation: Experimental study within a unified framework. *arXiv preprint arXiv:2501.17015*, 2025. 6
- [27] Haochen Liu, Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Multi-modal motion prediction with group-wise modal assignment transformer for autonomous driving. In *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1533–1538. IEEE, 2024. 2
- [28] Haozhe Liu, Shikun Liu, Zijian Zhou, Mengmeng Xu, Yanping Xie, Xiao Han, Juan C Pérez, Ding Liu, Kumara Khatapitiya, Menglin Jia, et al. MarDini: Masked autoregressive diffusion for video generation at scale. *arXiv preprint arXiv:2410.20280*, 2024. 2
- [29] Weibo Mao, Chenxin Xu, Qi Zhu, Siheng Chen, and Yanfeng Wang. Leapfrog diffusion model for stochastic trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5517–5526, 2023. 2
- [30] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nicholas Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, et al. The waymo open sim agents challenge. *Advances in Neural Information Processing Systems*, 36, 2024. 6, 16
- [31] Muleilan Pei, Shaoshuai Shi, and Shaojie Shen. Advancing multi-agent traffic simulation via r1-style reinforcement fine-tuning. *arXiv preprint arXiv:2509.23993*, 2025. 2, 6
- [32] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Traffic modeling as next-token prediction. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2
- [33] Jarrid Rector-Brooks, Mohsin Hasan, Zhangzhi Peng, Zachary Quinn, Chenghao Liu, Sarthak Mittal, Nouha Dziri, Michael Bronstein, Yoshua Bengio, Pranam Chatterjee, et al. Steering masked discrete diffusion models via discrete denoising posterior prediction. *arXiv preprint arXiv:2410.08134*, 2024. 1, 2, 3
- [34] Luke Rowe, Martin Ethier, Eli-Henry Dykhne, and Krzysztof Czarnecki. Fjmp: Factorized joint multi-agent motion prediction over learned directed acyclic interaction graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13745–13755, 2023. 2
- [35] Luke Rowe, Roger Girgis, Anthony Gosselin, Bruno Carréz, Florian Golemo, Felix Heide, Liam Paull, and Christopher Pal. Ctrl-sim: Reactive and controllable driving agents with offline reinforcement learning. *arXiv preprint arXiv:2403.19918*, 2024. 1
- [36] Hongzhi Ruan, Haibao Yu, Wenxian Yang, Siqi Fan, and Zaiqing Nie. Learning cooperative trajectory representations for motion forecasting. *Advances in Neural Information Processing Systems*, 37:13430–13457, 2024. 1
- [37] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *arXiv preprint arXiv:2406.07524*, 2024. 1, 2, 3
- [38] 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*, pages 8579–8590, 2023. 2
- [39] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35:6531–6543, 2022. 17
- [40] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. MTR++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2, 4, 7
- [41] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2
- [42] Qiao Sun, Zhicheng Jiang, Hanhong Zhao, and Kaiming He. Is noise conditioning necessary for denoising generative models? *arXiv preprint arXiv:2502.13129*, 2025. 2
- [43] Shuhan Tan, Kelvin Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 892–901, 2021. 1
- [44] Shuhan Tan, Boris Ivanovic, Yuxiao Chen, Boyi Li, Xinshuo Weng, Yulong Cao, Philipp Krähenbühl, and Marco Pavone. Promptable closed-loop traffic simulation. *arXiv preprint arXiv:2409.05863*, 2024. 1
- [45] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021. 4
- [46] Julen Urain, Ajay Mandlekar, Yilun Du, Mahi Shafullah, Danfei Xu, Katerina Fragkiadaki, Georgia Chalvatzaki, and Jan Peters. Deep generative models in robotics: A survey on learning from multimodal demonstrations. *arXiv preprint arXiv:2408.04380*, 2024. 2
- [47] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132, 2022. 1

- [48] Philipp Wu, Arjun Majumdar, Kevin Stone, Yixin Lin, Igor Mordatch, Pieter Abbeel, and Aravind Rajeswaran. Masked trajectory models for prediction, representation, and control. In *International Conference on Machine Learning*, pages 37607–37623. PMLR, 2023. 2
- [49] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. SMART: Scalable multi-agent real-time simulation via next-token prediction. *arXiv preprint arXiv:2405.15677*, 2024. 1, 2, 6, 17
- [50] Qi Yan, Brian Zhang, Yutong Zhang, Daniel Yang, Joshua White, Di Chen, Jiachao Liu, Langechuan Liu, Binnan Zhuang, Shaoshuai Shi, et al. Trajflow: Multi-modal motion prediction via flow matching. *arXiv preprint arXiv:2506.08541*, 2025. 2
- [51] Brian Yang, Huangyuan Su, Nikolaos Gkanatsios, Tsung-Wei Ke, Ayush Jain, Jeff Schneider, and Katerina Fragkiadaki. Diffusion-es: Gradient-free planning with diffusion for autonomous and instruction-guided driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15342–15353, 2024. 2
- [52] Dongkun Zhang, Jiaming Liang, Ke Guo, Sha Lu, Qi Wang, Rong Xiong, Zhenwei Miao, and Yue Wang. Carplanner: Consistent auto-regressive trajectory planning for large-scale reinforcement learning in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17239–17248, 2025. 6
- [53] Zhejun Zhang, Peter Karkus, Maximilian Igl, Wenhao Ding, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Closed-loop supervised fine-tuning of tokenized traffic models. *arXiv preprint arXiv:2412.05334*, 2024. 2, 6, 7, 17
- [54] Zhiyuan Zhang, Xiaosong Jia, Guanyu Chen, Qifeng Li, and Junchi Yan. Trajtok: Technical report for 2025 waymo open sim agents challenge. *arXiv preprint arXiv:2506.21618*, 2025. 6
- [55] Jianbo Zhao, Jiaheng Zhuang, Qibin Zhou, Taiyu Ban, Ziyao Xu, Hangning Zhou, Junhe Wang, Guoan Wang, Zhiheng Li, and Bin Li. Kigras: Kinematic-driven generative model for realistic agent simulation. *IEEE Robotics and Automation Letters*, 2024. 1
- [56] Jianbo Zhao, Taiyu Ban, Zhihao Liu, Hangning Zhou, Xiyang Wang, Qibin Zhou, Hailong Qin, Mu Yang, Lei Liu, and Bin Li. Droke: Directional rotary position embedding for efficient agent interaction modeling. *arXiv preprint arXiv:2503.15029*, 2025. 1, 6
- [57] Yinan Zheng, Ruiming Liang, Kexin ZHENG, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *The Thirteenth International Conference on Learning Representations*, 2025. 1, 2, 6, 14, 18
- [58] 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 *Conference on Robot Learning*, pages 144–177. PMLR, 2023. 1
- [59] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3560–3566. IEEE, 2023. 1, 5, 16
- [60] Yang Zhou, Hao Shao, Letian Wang, Steven L Waslander, Hongsheng Li, and Yu Liu. Smartpretrain: Model-agnostic and dataset-agnostic representation learning for motion prediction. *arXiv preprint arXiv:2410.08669*, 2024. 1
- [61] Yunsong Zhou, Naisheng Ye, William Ljungbergh, Tianyu Li, Jiazhi Yang, Zetong Yang, Hongzi Zhu, Christoffer Petersson, and Hongyang Li. Decoupled diffusion sparks adaptive scene generation. *arXiv preprint arXiv:2504.10485*, 2025. 2
- [62] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023. 4, 12
- [63] Zikang Zhou, Zihao Wen, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Qcnext: A next-generation framework for joint multi-agent trajectory prediction. *arXiv preprint arXiv:2306.10508*, 2023. 1
- [64] Zikang Zhou, Haibo Hu, Xinhong Chen, Jianping Wang, Nan Guan, Kui Wu, Yung-Hui Li, Yu-Kai Huang, and Chun Jason Xue. BehaviorGPT: Smart agent simulation for autonomous driving with next-patch prediction. *arXiv preprint arXiv:2405.17372*, 2024. 1, 2, 6

MDG: Masked Denoising Generation for Multi-Agent Behavior Modeling in Traffic Environments

Supplementary Material

A. Model Details

A.1. Scene Encoder

Agent Encoder. The agent state tensor contains per-timestep states for all agents, including their (x, y) coordinates, heading, velocity, and bounding box dimensions (length, width, height). All agent trajectories are transformed into their respective local coordinate frames, using the final observed state as the origin. An MLP-Mixer encoder is employed to capture both temporal dynamics and feature interactions within each agent’s state sequence. Specifically, the agent state tensor is processed by stacked MLP-Mixer blocks that alternately mix information along the temporal and feature dimensions, producing a latent tensor of shape $[N, H, D]$, where N is the number of agents, H the historical timesteps, and D the hidden dimension. Temporal max-pooling is then applied to summarize motion patterns across time. Each agent type is embedded via a learnable embedding vector and added to its feature representation. The final agent encoding has the shape $[N, D]$.

Map Encoder. The map tensor consists of N_m polylines (e.g., road centerlines, lane boundaries, crosswalks), each with N_w waypoints described by (x, y) coordinates and heading. All map elements are transformed into local coordinates using the first waypoint of each polyline as the origin. Each polyline is processed by an MLP-Mixer that models both intra-polyline spatial dependencies and cross-feature correlations. The Mixer operates on a tensor of shape $[N_m, N_w, D]$, and its outputs are aggregated by max-pooling along the waypoint axis to produce a polyline-level feature tensor $[N_m, D]$. Polyline type and associated traffic-signal state are encoded via learnable embeddings and added to the feature representation. The final map encoding has the shape $[N_m, D]$.

Ego-Route Encoder. For planning tasks, we include route polylines for the ego agent, consisting of N_r polylines, each with N_w waypoints defined by (x, y) coordinates and heading. All waypoints are transformed into local coordinates using the first waypoint of each polyline as the origin. An MLP-Mixer encoder processes these route polylines in the same manner as the map encoder, capturing both waypoint-wise spatial relations and inter-feature dependencies. The resulting ego-route encoding has shape $[N_r, D]$.

Traffic Signal Encoder. Traffic signals are represented by their stop points. Since we adopt relative spatial encoding, only the current signal phase (e.g., red, yellow, green) is encoded. An MLP Embedding layer converts each signal

state into a feature tensor of shape $[N_s, D]$.

Relative Relation Encoder. To model spatial relationships among all scene elements, we compute pairwise relative attributes between every pair (i, j) . The relative distance and heading are computed as follows: (1) for agents, using the last observed position; (2) for map elements, using the first waypoint of each polyline; and (3) for traffic signals, using the stop point position. Self-relations are assigned a small constant. An MLP-based Fourier Embedding relation encoder [62] encodes these pairwise relations into a relation tensor of shape $[N + N_m + N_s, N + N_m + N_s, D]$, enabling subsequent attention modules to incorporate spatial context and topology-aware reasoning.

Query-centric Transformer Encoder. The Transformer encoder is employed to extract relationships among all scene elements. It comprises a stack of query-centric attention layers, which use the same attention mechanism in Eq. (S1). The relative relation encodings are incorporated into this encoder. All scene elements (agents, map polylines, and traffic signals) are concatenated into a tensor of shape $[N + N_m + N_s, D]$, which is input to the Transformer. After passing through several attention layers, the Transformer learns to capture the interactions and relationships between the scene elements. Invalid elements are masked out during the attention calculations. The final scene context encoding retains its original shape $[N + N_m + N_s, D]$. The relative position and heading differences between elements i and j are encoded as edge attributes, forming a relation encoding tensor $\mathbf{e}^{i \rightarrow j}$. The relative cross-attention (RCA) is defined as:

$$RCA(Q^i, K, V, \mathbf{e}) = \text{softmax} \left(\frac{\mathbf{q}^i}{\sqrt{D}} \left[\left\{ \mathbf{k}^j + \mathbf{e}^{i \rightarrow j} \right\}_{j \in \Omega(j)} \right]^T \right) \times \left(\left\{ \mathbf{v}^j + \mathbf{e}^{i \rightarrow j} \right\}_{j \in \Omega(j)} \right), \quad (\text{S1})$$

where $\mathbf{q}^i, \mathbf{k}^j, \mathbf{v}^j$ represent the query, key, and value elements respectively, each containing relevant element-centric information, and $j \in \Omega(j)$ defines the set of indices corresponding to neighboring elements.

A.2. Decoder

Noised Future Encoder. We first compute each agent’s control actions (acceleration and yaw rate) from their logged trajectories. These actions are normalized to form a clean action tensor of shape $[N, T_a, 2]$, where T_a denotes the reduced temporal dimension obtained via action chunking, which improves computational and memory efficiency. Gaussian noise is then applied to the clean actions, which

are subsequently transformed into noisy physical states (positions, headings, and velocities) through a differentiable dynamics function, resulting in a tensor of shape $[N, T_a, 5]$. The noised states are encoded using an MLP. Mask and timestep embeddings are then added to produce the noised future query tensor of shape $[N, T_a, D]$, serving as the input to the denoising Transformer.

Transformer Denoiser. The denoiser consists of stacked Transformer blocks, each containing complementary attention layers that operate over different relational dimensions:

- **Intra-Agent Temporal Attention.** A multi-head self-attention layer extracts temporal dependencies within each agent’s future trajectory.
- **Inter-Agent Interaction Attention.** The outputs are passed through a cross-attention layer that models inter-agent interactions. Invalid or missing agents are masked. Relation encodings from the encoder are used as positional and relational priors (inter-agent relations) in this attention computation.
- **Agent-Scene Condition Cross-Attention.** A cross-attention layer fuses the agent features with scene context representations from the encoder (map, traffic lights, and agents). For planning tasks, the ego agent receives additional conditioning through cross-attention with the ego-route encoding, which provides route-level geometric and semantic guidance. This route information is exclusively accessible to the ego agent. Relation encodings for agents and scene elements (map polylines, traffic agents, traffic lights, and optional ego-route polylines) are incorporated into this attention process.

Residual connections are applied across all Transformer layers to stabilize optimization and preserve gradient flow. The final denoised latent tensor is decoded with an MLP that outputs denoised action sequences for each agent.

Trajectory Decoder. An auxiliary MLP decoder operates directly on the agent-specific scene context encodings to predict future trajectories of shape $[N, M, T, 3]$, where M is the number of trajectory modalities. This auxiliary prediction branch stabilizes training and encourages the scene encoder to learn trajectory-relevant representations.

Differentiable Dynamic Function. We adopt the differentiable dynamics function from [18] to convert predicted agent actions (acceleration and yaw rate) into corresponding physical states (positions and headings), conditioned on each agent’s current state. As this function is differentiable, it is integrated into the model as a trainable layer, enabling end-to-end gradient propagation through both the kinematic transformation and the denoising process.

A.3. Model Parameters

The primary parameters used in our MDG model are summarized in Tab. S1 and Tab. S2. Separate configurations are adopted for the Waymo and nuPlan experiments to account

for differences in scene complexity and task requirements. Unless otherwise specified, parameters in the nuPlan setup are identical to those in the Waymo configuration.

B. Training Details

The training objective combines two loss components and invalid agents and timesteps are excluded:

$$\mathcal{L} = \mathcal{L}_d + \lambda \mathcal{L}_p, \quad (\text{S2})$$

where \mathcal{L}_d represents the denoising loss, and \mathcal{L}_p is an auxiliary prediction loss, and $\lambda = 5$ is the balance weight. The prediction loss \mathcal{L}_p is defined as:

$$\mathcal{L}_p = \frac{1}{N} \sum_{i=1}^N \mathcal{SL}_1(\hat{\mathbf{s}}_i^* - \mathbf{s}_i^{gt}), i^* = \arg \min_m \|\hat{\mathbf{s}}_i^m - \mathbf{s}_i^{gt}\|_2, \quad (\text{S3})$$

where \mathcal{SL}_1 denotes the smooth L_1 loss applied to the best-predicted trajectory $\hat{\mathbf{s}}_i^*$, defined as the trajectory with the minimal L_2 from the ground truth \mathbf{s}_i^{gt} .

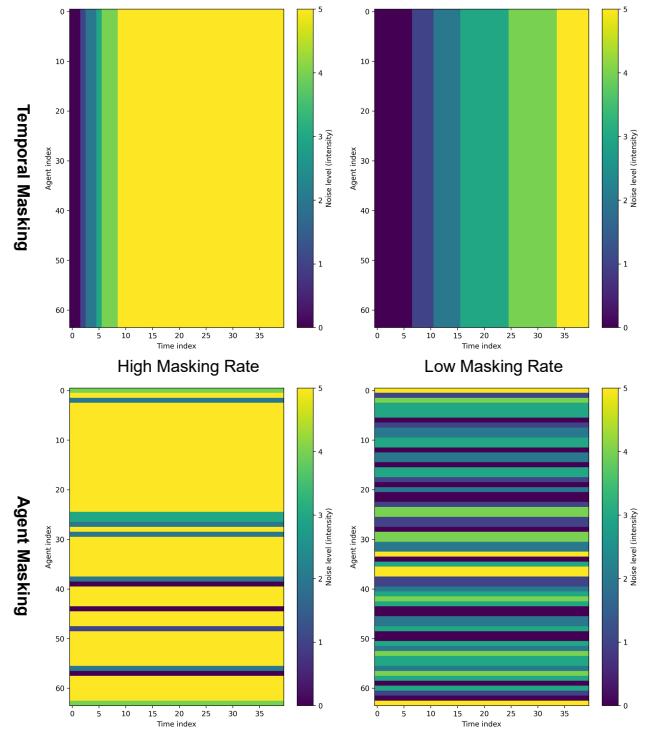


Figure S1. Illustration of the proposed training noise masking strategy. Top: Temporal masking; Bottom: Agent masking. Each column shows an example under either a high or low masking rate.

We propose a novel masking strategy that distributes the masking rate across training batches and applies masking randomly along either the temporal or agent axis. This approach facilitates adaptive denoising generation during inference and guidance. The training procedure is detailed

Table S1. Model Parameters for Waymo Experiments

Parameter	Value	Description
N	128	Number of agents per scene
N_m	320	Number of map polylines
N_w	16	Number of waypoints per polyline
N_s	16	Number of traffic lights
H	11	History steps
T	80	Future steps
T_a	40	Reduced future timesteps
D	256	Hidden dimension
M	6	Number of trajectory modalities in predictor
Modality encoder layers	2	Number of MLP-Mixer layers in the encoder
Encoder layers	6	Number of query-centric Transformer layers
Decoder blocks	2	Number of Transformer denoising blocks
Attention heads	8	Number of attention heads per layer
Dropout rate	0.1	Dropout probability
Feed-forward dimension	1024	Dimension of the feed-forward network
Action chunk	2	Action sequence granularity
Action mean	[0.0, 0.0]	Mean of action distribution [acceleration, yaw rate]
Action std	[1.0, 0.5]	Standard deviation of action distribution [acceleration, yaw rate]

Table S2. Model Parameters for nuPlan Experiments

Parameter	Value	Description
N	32	Number of agents per scene
N_m	256	Number of map polylines
N_m	32	Number of ego-route polylines
N_w	20	Number of waypoints per polyline
H	20	History steps
T	80	Future steps

in Algorithm 1. The strategy involves randomly selecting masking patterns: masking over time or masking over agents. For each training sample, a masking rate δ is assigned. When masking over time, a δ fraction of the later timesteps for each agent are fully noised, while other timesteps are randomly assigned noise levels that progressively increase. When masking over agents, a δ fraction of the agents are fully noised with the highest noise levels, and the remaining agents are randomly assigned lower noise levels, while the noise levels for an individual agent remain consistent across timesteps. Importantly, the masking rate δ is evenly distributed across all samples in a batch to ensure robust learning. Fig. S1 illustrates both masking types under high and low masking rates.

For Waymo experiments, to balance computational efficiency and model performance, we limit training to the 64 agents nearest to the labeled self-driving car (including itself). This reduces GPU memory usage while accommodating most scenarios, as they typically involve fewer than 64 valid agents. During testing, the model can scale up to 128 agents, leveraging the flexible attention mechanisms of

the Transformer architecture.

For the nuPlan experiments, following established practice [5, 57], we apply random perturbations to the current state as data augmentation. A short interpolation segment is inserted to ensure a physically plausible transition, allowing the model to remain robust to perturbations and converge back to the ground-truth trajectory.

Training is performed on eight NVIDIA L40S GPUs using the AdamW optimizer with a weight decay of 0.01 and BFloat16 precision. The set of training hyperparameters is provided in Tab. S3.

C. Inference Details

Multi-step Denoising. Our proposed MDG framework offers flexibility during inference by enabling multi-step generation along either the temporal axis or the agent axis. Below, we elaborate on the inference process of our masked denoising model. The inference begins by initializing the process with full Gaussian noise, corresponding to a mask \bar{m}_L that fully covers the data with noise. This noisy in-

Table S3. Hyperparameters for Model Training

Hyperparameter	Value	Description
Noise levels	5	Number of distinct noise levels
Learning rate	0.0002	Initial learning rate
LR decay step	2000	Step interval for learning rate decay
LR warmup step	1000	Warmup steps at the beginning
LR decay factor	0.98	Multiplicative decay factor
Batch size	4	Number of samples per GPU
Training epochs	20	—
Gradient clip	1.0	Maximum gradient norm for clipping

Algorithm 1 Training Procedure for MDG

```

1: Input: Data  $X$ , Noise schedule  $\alpha$ , Noise level  $K$ , Denoising model  $\mathcal{D}_\theta$ ,
2: Output: Trained model  $\mathcal{D}_{\theta^*}$ 
3: Initialize model parameters  $\theta$ 
4: for Each training batch  $X_b$  do
5:   for Each sample  $x \in X_b$  do
6:     Randomly select masking type: time-axis or agent-axis
7:     Assign masking rate  $\delta \in [0, 1]$ 
8:     if masking over time then
9:       Apply full noise to  $\delta$  fraction of later timesteps
10:      Assign random progressively increasing noise levels to remaining timesteps
11:    else
12:      Apply full noise to  $\delta$  fraction of agents
13:      Randomly assign noise levels across timesteps for each agent
14:    end if
15:    Generate noise mask  $\mathbf{m}$  and add noise to  $x$  according to schedule  $\alpha(\mathbf{m})$ 
16:  end for
17:  Perform a forward pass of the model on  $X_b$ 
18:  Compute loss  $\mathcal{L}$ 
19:  Backpropagate the loss and update  $\theta$  using an optimizer
20: end for

```

put, along with the corresponding mask, is fed into the denoising model to generate an intermediate clean sample. In the subsequent iteration, the generated clean sample is re-noised according to the noise mask at the next step $\bar{\mathbf{m}}_{L-1}$. This noised sample is then passed through the model for denoising. This iterative process continues until the final step, where the final denoised sample is obtained. The complete inference procedure is illustrated in Algorithm 2.

The design of denoising schedules at inference time is important. We propose three strategies for this purpose. 1) *Single-step Denoising*: The model produces a clean out-

Algorithm 2 Inference Procedure for MDG

```

1: Input: Denoising step  $L$ , Noise mask schedule  $\{\bar{\mathbf{m}}_\ell\}_{\ell=L}^0$ , Noise schedule  $\alpha$ , Denoising model  $\mathcal{D}_\theta$ 
2: Output: Denoised sample  $\hat{\mathbf{x}} = \mathbf{z}_0$ 
3: Initialize  $\mathbf{z}_L \sim \mathcal{N}(0, \mathbf{I})$  {Start with full Gaussian noise}
4: for  $\ell = L$  to 1 do
5:    $\hat{\mathbf{x}}_\ell \leftarrow \mathcal{D}_\theta(\mathbf{z}_\ell, \bar{\mathbf{m}}_\ell)$  {Denoise the current sample}
6:    $\mathbf{z}_{\ell-1} \leftarrow \sqrt{\alpha(\bar{\mathbf{m}}_{\ell-1})}\hat{\mathbf{x}}_\ell + \sqrt{1 - \alpha(\bar{\mathbf{m}}_{\ell-1})}\epsilon$ ,  $\epsilon \sim \mathcal{N}(0, \mathbf{I})$  {Re-noise for next step}
7: end for
8: Return  $\mathbf{z}_0$  {Final denoised output}

```

put in a single pass starting from a fully masked noisy input. 2) *Temporal Axis Denoising*: Denoising proceeds progressively along the temporal dimension, analogous to autoregressive next-step generation. The noise level at each step is adjusted based on the remaining timesteps and the predefined noise scale, enabling flexible and fine-grained temporal refinement. 3) *Agent Axis Denoising*: Similar to temporal-axis denoising, this strategy iterates along the agent dimension instead. It offers flexibility in the ordering and progression of agents, which can be advantageous in multi-agent scenarios with heterogeneous importance or interaction structures. Fig. S2 illustrates an example of the temporal-axis inference schedule. These flexible strategies allow the MDG framework to tailor its denoising process to diverse tasks and operational constraints.

Guidance Imposition. The training mechanism of the MDG model facilitates a straightforward yet effective guidance imposition method to change the behavior of selected target agents while preserving the reactivity of other agents and maintaining overall scene consistency. This method involves replacing the denoised trajectories of the target agents with modified ones and adding small controlled noise. The adjusted trajectories are then fed into the next denoising iteration. To implement this, we define a guidance noise mask $\bar{\mathbf{g}}$, assigning a fixed low noise level $\alpha = 0.8$ to all timesteps of the target agents while leaving other agents

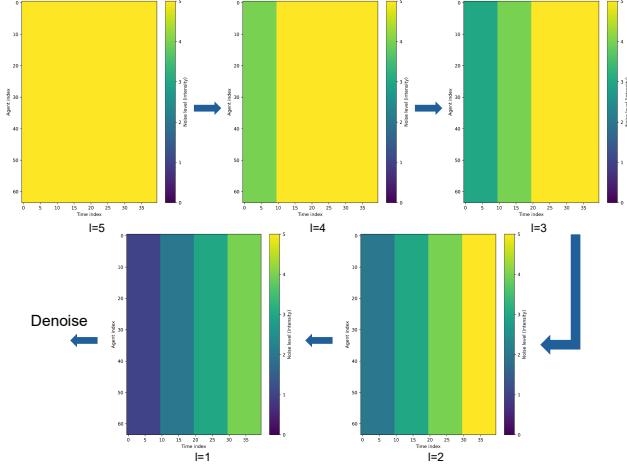


Figure S2. Example of a five-step temporal-axis denoising schedule used during inference.

unaffected. This mask remains consistent throughout the denoising generation process, and the noise mask schedule controls the denoising of other agents. This design allows us to explicitly optimize the trajectories of the target agents or replace them with prior prediction results, without compromising the overall scene dynamics. The guidance procedure is illustrated in Algorithm 3. Notably, unlike diffusion-based models [21, 59], our method avoids differentiating through the denoiser, which shows significantly faster inference speeds, enhancing its practical applicability.

Algorithm 3 Guidance Imposition Procedure for MDG

- 1: **Input:** Noise mask schedule $\{\bar{m}_\ell\}_{\ell=L}^0$, Guidance noise mask \bar{g} , Denoising schedule α , Denoising model \mathcal{D}_θ , Objective function \mathcal{J}
- 2: **Output:** Denoised sample $\hat{x} = z_0$
- 3: Initialize $z_L \sim \mathcal{N}(0, I)$ {Start with full Gaussian noise}
- 4: **for** $\ell = L$ to 1 **do**
- 5: $\hat{x}_\ell \leftarrow \mathcal{D}_\theta(z_\ell, \bar{m}_\ell)$ {Denoise the current sample}
- 6: $\hat{x}_\ell^M \leftarrow \mathcal{J}(\hat{x}_\ell)$ {Modify the target agents' trajectories on the clean sample}
- 7: $z_{\ell-1} \leftarrow \sqrt{\alpha(\max(\bar{m}_{\ell-1}, \bar{g}))} \hat{x}_\ell^M + \sqrt{1 - \alpha(\max(\bar{m}_{\ell-1}, \bar{g}))} \epsilon, \epsilon \sim \mathcal{N}(0, I)$ {Apply guidance mask and re-noise for the next step}
- 8: **end for**
- 9: **Return** z_0 {Final denoised output}

D. Experiment Details

D.1. Evaluation Metrics

The evaluation metrics employed in the Waymo Sim Agents benchmark are detailed in [30]. These metrics are designed to quantify the distributional divergence between ground-truth agent trajectories and simulated agent trajectories. The evaluation contains three key aspects: kinematic features, interactive features (e.g., time-to-collision (TTC), collision rate, distance to the nearest object), and map-based features (e.g., off-road and distance to road edge). A meta-composite realism score aggregates these components and serves as the primary evaluation metric.

For the nuPlan benchmark, we employ the closed-loop evaluation metrics defined in [23]. These include No At-Fault Collisions, Drivable Area Compliance, Making Progress, Driving Direction Compliance, TTC Within Bound, Progress Along Route Ratio, Speed Limit Compliance, and Comfort. The individual scores are weighted to produce the final Closed-Loop Score (CLS).

In addition to these benchmark-specific metrics, we report several supplementary measures to compare our model against state-of-the-art methods in open-loop settings. These metrics include:

Collision Rate (CR). This metric measures the average collision rate per scene (the number of colliding agents divided by the total number of agents), averaged by the total number of testing scenarios:

$$CR = \frac{1}{N_T} \sum_{N_T} \frac{1}{N_S} \frac{1}{N_A} \sum_{j=1}^{N_S} \sum_{i=1}^{N_A} \mathbf{1}_c(s_i^j), \quad (S4)$$

where N_T is the number of test scenarios, N_S is the number of samples, N_A is the total number of modeled agents in a scenario, and s_i^j is the trajectory of agent i of sample j . $\mathbf{1}_c(s_i^j)$ is an indicator function that equals 1 if the simulated trajectory results in a collision, and 0 otherwise.

Off-road Rate (OR). This metric quantifies the proportion of agents deviating off-road. It is calculated as the average off-road rate per scene (samples and agents), averaged by the total number of test scenarios:

$$OR = \frac{1}{N_T} \sum_{N_T} \frac{1}{N_S} \frac{1}{N_A} \sum_{j=1}^{N_S} \sum_{i=1}^{N_A} \mathbf{1}_o(s_i^j), \quad (S5)$$

where $\mathbf{1}_o(s_i^j)$ is an indicator function that equals 1 if the simulated trajectory veers off-road (e.g., outside the road boundary), and 0 otherwise. Note that we exclude those agents that are already off-road or labeled as pedestrians.

Scene Average Displacement Error (SADE). SADE computes the average Euclidean distance (L2 norm) between logged ground-truth trajectories and simulated trajectories

Table S4. Influence of the auxiliary trajectory predictor on closed-loop performance

Task	Predictor	Main Metric (\uparrow)	minADE (m) (\downarrow)
Waymo Sim Agents	✓	0.7842 (Realism Meta)	1.3123
	–	0.7682	1.4117
nuPlan Val14-NR	✓	89.75 (Closed-loop Score)	–
	–	86.90	–

Table S5. Influence of action-to-state conversion on open-loop multi-agent trajectory prediction

Method	CR (%) \downarrow	OR (%) \downarrow	SADE (m) \downarrow	minSADE (m) \downarrow
W/ conversion	4.875	3.274	3.122	1.840
W/o conversion	8.875	6.066	4.353	2.853

across all agents and scenarios:

$$\text{SADE} = \frac{1}{N_T} \sum_{N_T} \frac{1}{N_S} \frac{1}{N_A} \sum_{j=1}^{N_S} \sum_{i=1}^{N_A} \|s_i^j - s_i^{gt}\|_2, \quad (\text{S6})$$

where s_i^{gt} represents the ground-truth trajectory of an agent. **Minimum SADE (minSADE).** This metric captures the minimum SADE (averaged over objects and minimum over samples), providing a measure of the best-case alignment with ground truth.

$$\text{minSADE} = \frac{1}{N_T} \sum_{N_T} \min_j \frac{1}{N_A} \sum_{i=1}^{N_A} \|s_i^j - s_i^{gt}\|_2. \quad (\text{S7})$$

Goal-reach Rate (GR). This metric measures the average number of agents that successfully reach their assigned goals across all scenarios. An agent is considered to have reached its goal if the distance between its position and the goal position is less than 1 meter. The final value is obtained by averaging over the total number of scenarios.

D.2. Baseline Methods for Open-loop Tasks

SMART [49] is a state-of-the-art autoregressive next-token generation model for traffic agent simulation, achieving top performance on the Waymo Sim Agents benchmark. In our experiments, we use the tiny variant of SMART with 7M parameters. We adopt the open-source implementation provided in CAT-K [53] and follow the default training and inference settings. To ensure a fair comparison, we do not perform closed-loop fine-tuning and only use the first-stage behavior cloning training.

VBD [18] is a diffusion-based generative model for traffic agent simulation and ranked second in the 2024 Waymo Sim Agents Challenge. VBD performs joint trajectory diffusion, where all agents at a given diffusion step share the same noise level. We reproduce the model using the authors' open-source implementation and evaluate it with five

diffusion steps, generating the final predictions under this configuration.

MTR [39] is a high-performance motion prediction model achieving state-of-the-art results on the Waymo Motion Prediction benchmark. We adapt MTR for multi-agent simulation by using the same encoder as in our model and extending the decoder to produce predictions for all agents simultaneously. Predefined trajectory anchors serve as initial queries for each agent, allowing marginal predictions for all agents in a single forward pass without iterative querying.

E. Additional Results

E.1. Ablation Studies

Influence of the Auxiliary Predictor. To examine the effect of the auxiliary trajectory predictor in the model, we conduct an ablation study by removing the predictor head from the model. As summarized in Tab. S4, excluding the predictor leads to a consistent decline in performance across both benchmarks. On the Waymo Sim Agents evaluation, the realism meta-metric drops from 0.7842 to 0.7682, while minADE increases from 1.3123 m to 1.4117 m, reflecting degraded simulation realism and motion accuracy. Similarly, on the nuPlan Val14 benchmark, CLS decreases from 89.75 to 86.90, indicating reduced planning performance. These results demonstrate that the auxiliary predictor enhances representation learning within the scene encoder by providing trajectory-level supervision.

Influence of Action-to-State Conversion. The Transformer-based denoiser operates on a spatiotemporal representation of agent behaviors. To better capture motion dynamics, a differentiable kinematic function converts agent actions (e.g., acceleration and yaw rate) into corresponding physical states (x, y, θ, v) , which are then encoded and denoised. We evaluate the impact of this action-to-state conversion on multi-agent open-loop trajectory prediction. All models are trained with five noise levels and evaluated using five-step temporal denoising

Table S6. Influence of model scale on open-loop multi-agent trajectory prediction

Scale	# Model Params	CR (%) ↓	OR (%) ↓	SADE (m) ↓	minSADE (m) ↓
D=128	2.5M	5.477	4.585	3.714	2.358
D=256	10.0M	4.875	3.274	3.122	1.840
D=512	39.4M	4.905	3.425	3.289	1.921

Table S7. Ablation on attention mechanisms in the MDG denoiser on open-loop multi-agent trajectory prediction

Attention Modules			Metrics		
Intra-Agent	Inter-Agent	Agent-Scene	CR (%) ↓	SADE (m) ↓	minSADE (m) ↓
✓	✓	✓	4.875	3.122	1.840
✓	—	✓	6.821	3.580	2.060
—	✓	✓	5.612	3.460	1.990
—	—	✓	9.964	4.070	2.310

to ensure consistent settings. As shown in Tab. S5, integrating the conversion module substantially improves all evaluation metrics. Specifically, incorporating physical state transformation reduces collision and off-road rates by more than 40% and decreases both SADE and minSADE by a significant margin. This improvement highlights the importance of embedding kinematic consistency within the denoiser: the conversion enables the model to reason over sequential dynamics rather than isolated action tokens, resulting in more physically coherent and accurate trajectory generation.

Influence of Model Scale. We examine the impact of model scale on open-loop prediction performance by varying the hidden dimension of the Transformer while keeping the number of layers fixed. All models are trained for the same number of epochs, with batch sizes adjusted to fit GPU memory constraints. As shown in Tab. S6, enlarging the hidden dimension from 128 to 256 consistently improves performance across all metrics. However, further scaling to 512 yields no additional gains, indicating diminishing returns. This saturation may stem partly from smaller batch sizes used at larger scales, but the primary factor is likely the limited training data, which restricts the model’s ability to fully leverage its increased capacity.

Influence of Attention in Denoiser. We analyze the contribution of each attention module in the Transformer denoiser. As reported in Tab. S7, removing any of the three components consistently degrades prediction performance across all metrics, while the full model achieves the best overall results. These findings highlight the complementary roles of intra-agent, inter-agent, and agent-scene attention in the denoiser.

E.2. Visualization

Denoising Process. We provide visualizations of the denoising process to illustrate its behavior. Fig. S3 depicts

the temporal denoising process over five steps. Initially, the denoising results exhibit similarities across predictions. However, as the process progresses, subsequent denoising steps yield increasingly distinct outcomes for future predictions. Temporal denoising primarily focuses on the agents’ dynamic behaviors, and increasing the number of denoising steps enables greater variation in intermediate actions. This, in turn, enhances the behavioral diversity across the entire time horizon. In Fig. S4, we demonstrate the denoising process along the agents over five steps. Unlike temporal denoising, agent-axis denoising emphasizes iterative trajectory-level refinement. Early denoising results are sub-optimal for agents with high noise levels. However, as the denoising progresses, the trajectories become increasingly refined. This iterative process leads to a gradual determination of agents’ behaviors, ultimately producing diverse and multi-modal joint agent trajectories.

Closed-loop Reuse. An example of the closed-loop reuse denoising method in the Waymo dataset is illustrated in Fig. S5. In the case of a simple one-step denoising method, the ego agent’s planning results exhibit significant variability across consecutive frames, even with small intervals, indicating poor temporal consistency. In contrast, the closed-loop one-step reuse method demonstrates improved temporal consistency, yielding smoother planning trajectories and better overall planning performance.

Comparison on nuPlan Data. We compare MDG and the Diffusion Planner [57] in a representative nuPlan scenario, as shown in Fig. S6. The Diffusion Planner generates trajectories that lack temporal coherence across planning steps, causing the ego vehicle to drift from a safe course and ultimately collide with surrounding traffic. In contrast, MDG preserves temporal consistency through its closed-loop denoising mechanism, enabling stable motion planning and safe scenario completion.

Table S8. Detailed performance metrics of MDG across different benchmark splits

Benchmark	Score	Collision	TTC	Drivable area	Driving direction	Comfort	Ego progress	Speed limit
Val14 Non-Reactive	90.45	96.10	91.80	98.77	99.69	94.87	94.30	96.95
Val14 Reactive	83.89	93.54	88.33	98.33	99.58	90.01	86.58	97.82
Test14 Non-Reactive	90.16	95.93	91.46	98.78	99.69	94.51	94.08	96.85
Test14 Reactive	83.21	93.23	87.70	98.15	99.59	89.34	86.08	97.85

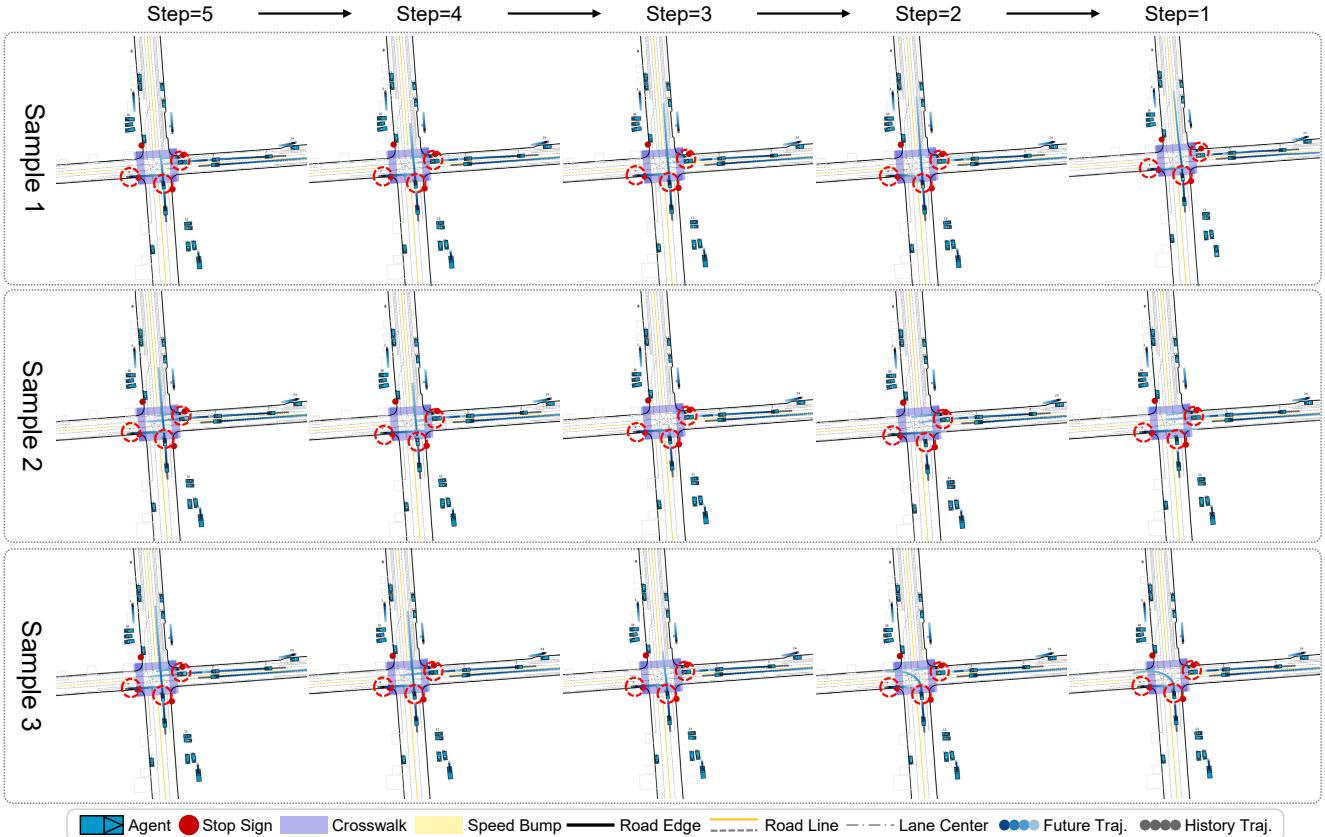


Figure S3. Visualization of the denoising process over time with five steps. Clean, denoised samples predicted by the MDG model at each step are shown. The process begins with similar predictions and gradually evolves to generate diverse future outcomes, emphasizing temporal behavior refinement and increased sample diversity across the time horizon.

E.3. Detailed Benchmark Results

Tab. S8 presents the detailed metrics of the MDG (1-step closed-reuse) method on the nuPlan benchmarks. Performance on reactive settings is lower primarily due to the simplistic IDM used for reactive agents, which produces unrealistic behaviors and artificial gaps.

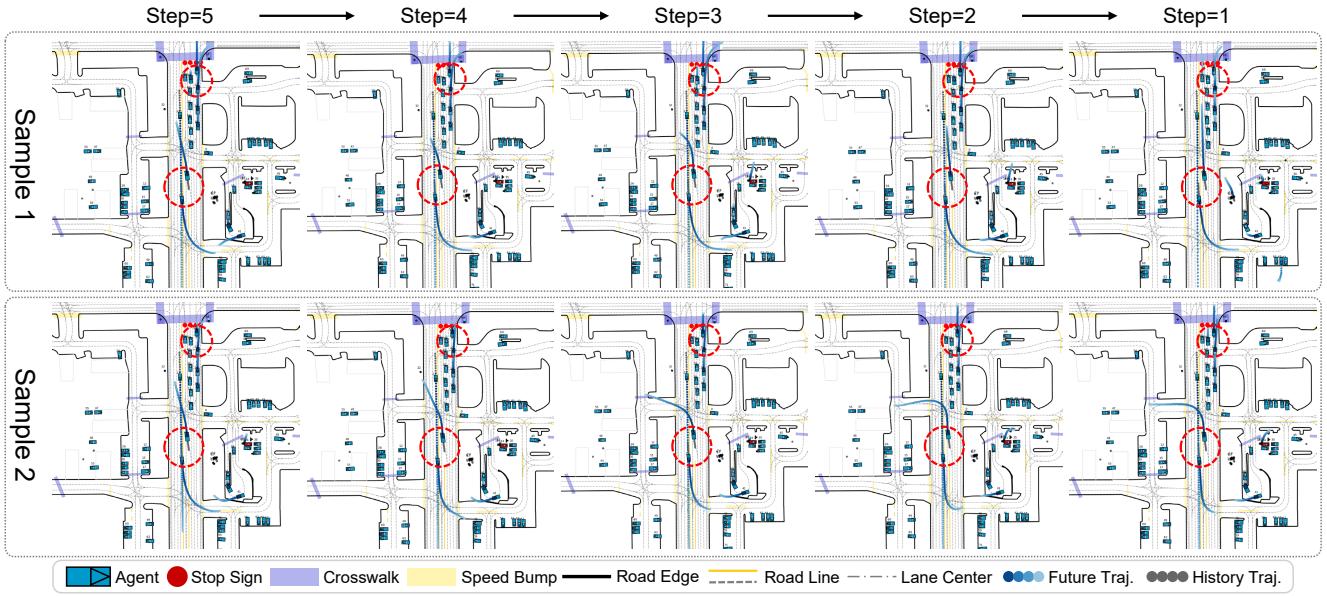


Figure S4. Visualization of the denoising process across agents over five steps. Clean, denoised samples predicted by the MDG model at each step are shown. Initial predictions for agents with high noise levels are suboptimal, but iterative refinement improves trajectory accuracy and enhances the diversity of behaviors.

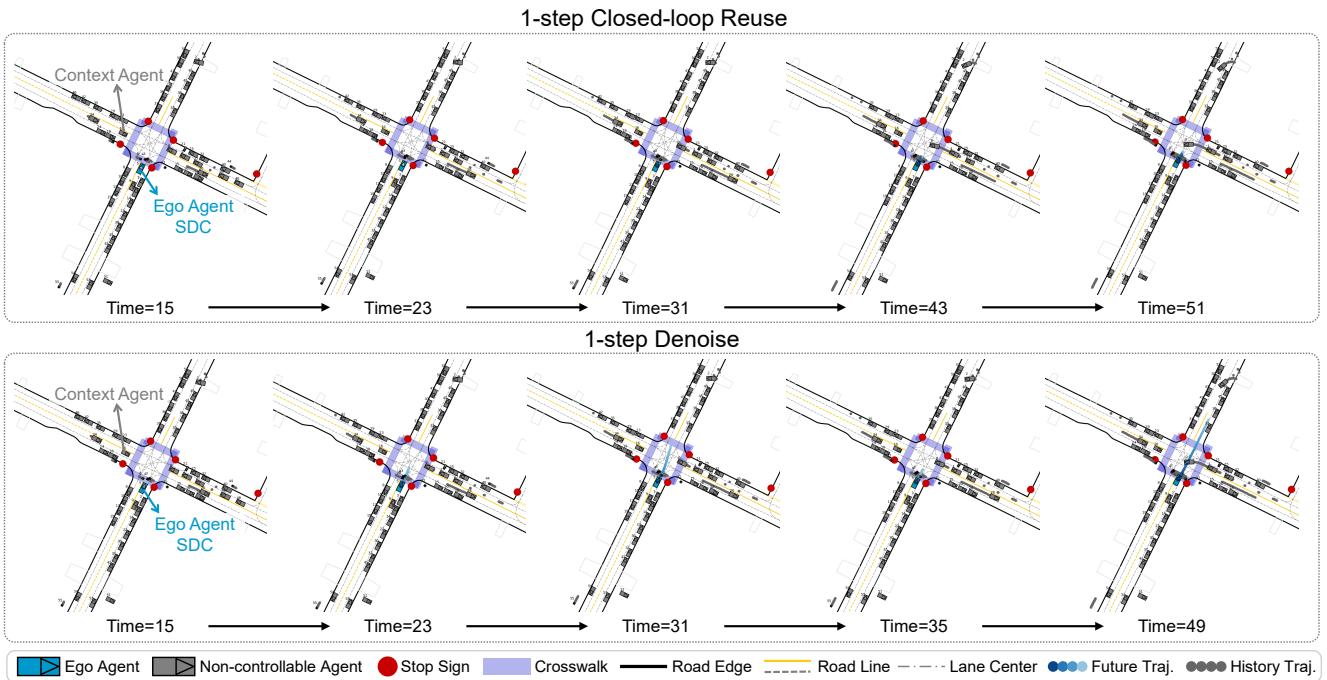


Figure S5. Comparison of a simple one-step denoising method and the closed-loop one-step reuse denoising method. The reuse method demonstrates improved temporal consistency and smoother planning trajectories for the ego agent.

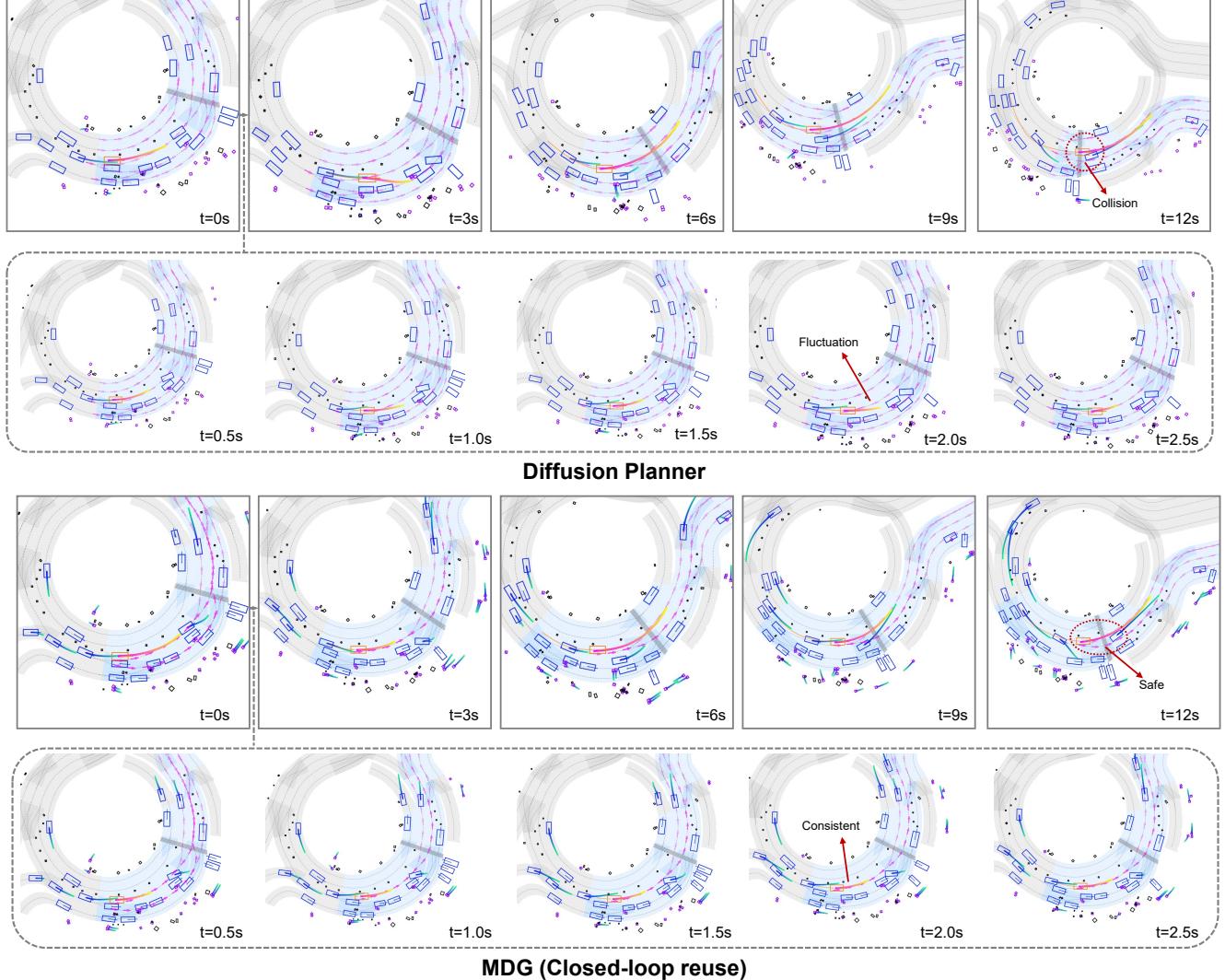


Figure S6. Comparison between MDG and Diffusion Planner in a nuPlan scenario. **Top:** The Diffusion Planner produces temporally inconsistent trajectories, leading to a collision. **Bottom:** MDG maintains consistent trajectories through a closed-loop denoising strategy and successfully completes the scenario without collisions.