Path Diffuser: Diffusion Model for Data-Driven Traffic Simulator

Da Saem Lee, Akash Karthikeyan, Yash Vardhan Pant, Sebastian Fischmeister

***Abstract*—Simulating diverse and realistic traffic scenarios is critical for developing and testing autonomous planning. Traditional rule-based planners lack diversity and realism, while learning-based simulators often replay, forecast, or edit scenarios using historical agent trajectories. However, they struggle to generate new scenarios, limiting scalability and diversity due to their reliance on fully annotated logs and historical data. Thus, a key challenge for a learning-based simulator's performance is that it requires agents' past trajectories and pose information in addition to map data, which might not be available for all agents on the road. Without which, generated scenarios often produce unrealistic trajectories that deviate from drivable areas, particularly under out-of-distribution (OOD) map scenes (e.g., curved roads). To address this, we propose Path Diffuser (PD): a two-stage, diffusion model for generating agent pose initializations and their corresponding trajectories conditioned on the map, free of any historical context of agents' trajectories. Furthermore, PD incorporates a motion primitive-based prior, leveraging Frenet frame candidate trajectories to enhance diversity while ensuring road-compliant trajectory generation. We also explore various design choices for modeling complex multi-agent interactions. We demonstrate the effectiveness of our method through extensive experiments on the Argoverse2 Dataset and additionally evaluate the generalizability of the approach on OOD map variants. Notably, Path-Diffuser outperforms the baseline methods by $1.92\times$ on distribution metrics, $1.14\times$ on common-sense metrics, and $1.62\times$ on road compliance from adversarial benchmarks[1].**

I. Introduction

Traffic scenario simulations that reflect the real world are crucial for evaluating autonomous driving systems [7]. Scenario simulation can be decomposed into two steps: (1) agent pose initialization and (2) trajectory generation. While trajectories can technically originate from any location on the map, realistic agent initialization is necessary to reflect real-world behavior. For instance, at the intersection in the real world, some vehicles may stop behind the stop line to allow other vehicles to pass through. Further, given an initialization, the trajectories should be generated by following the traffic rules (e.g., driving on the correct side of the road) and not colliding with the other agents to follow a real-world scenario.

Industry practitioners manually specify scenarios for scenario-based testing [1], [7] and use vehicle simulators, such as SUMO [13] and CARLA [5]. Although these frameworks are helpful, they often lack the realism and diversity necessary to fully reflect real-world conditions. Recent works [12], [14], [19], [25] have adopted learning-based techniques to mitigate this limitation and capture the underlying distributions in complex and diverse traffic scenarios.

Although data-driven traffic simulators have made significant progress with rasterized representation of a scene [3], [17], [20], several challenges remain, including scalability to the longer horizon and the complexity to incorporating control signals or constraints. To address these limitations, recent works [12], [19], [25] utilizes vectorized representation of a scene, which offer a more compact, interpretable, and flexible encoding of map and agent features. However, these methods often depend on agent history to generate future trajectories, leading to models that overfit and replay logged behaviors rather than generalizing to diverse scenarios. Moreover, absense of historical trajectories of other road agents in driving logs makes it challenging to scale these approaches. In the absence of historical context, generated scenarios often produce unrealistic behaviors, such as the vehicles stay still or deviate from drivable areas, particularly under out-of-distribution (OOD) map conditions (e.g., curved roads).

Agent initialization remains a key challenge. A traffic scenario can be generated by randomly placing vehicles in the scene, though this raises concerns about realism. In order to generate realistic traffic scenario, it is critical to have realistic initialization that aligns with the map structure and meets common sense, such as agents inside the drivable area or not in collision. To address this, recent work [14] utilizes the transformer to generate agent initializations given a map. However, this method requires the guidance sampling for collision and map constraints to generate realistic placements.

Contributions of this work. To address these challenges, we propose Path-Diffuser, a diffusion-based framework for traffic scenario generation that models agent initialization and trajectory generation. Our key contributions are as follows:

- We introduce a two stage diffusion framework that jointly generates agent initializations and their trajectories, conditioned on the map without requiring historical data.
- For Agent Initialization, we incorporate the differential transformer to suppress attention noise and generate realistic, spatially consistent scene configurations by attending relevant map context.
- For trajectory generation, we introduce Frenet-frame candidates, enabling the model to capture diverse trajectories

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Canada Foundation for Innovation - John R. Evans Leaders Fund (CFI JELF).

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. ds3lee@uwaterloo.ca, a9karthi@uwaterloo.ca, yash.pant@uwaterloo.ca, sfischme@uwaterloo.ca

[1]https://github.com/CL2-UWaterloo/PathDiffuser/

that are more robust to map perturbations.

In this framework, trajectory generation is designed to be independent of historical context, as such information may not be available for every agent in the scene. Moreover, we validate our framework on the Argoverse v2 dataset through qualitative and quantitative evaluation. To demonstrate the robustness to the OOD map variants, we also evaluated on the perturbed map.

Outline of the paper. Section II provides a brief overview of existing methods for agent initialization, motion forecasting, and end-to-end approaches for traffic scenario generation and simulation. Section III presents the required background, notation, and introduces the problem statement for this work. Section IV describes the proposed agent initialization and trajectory generation approaches. Section V presents quantitative and qualitative evaluations that benchmark the performance of the proposed approach against baselines. Finally, we discuss limitations and future work in Section VI.

II. Related Works

Although rule-based traffic simulation methods, such as [1], [7], are widely used for testing autonomous driving systems, given the scope of this work, we restrict this section to data-driven methods for traffic scenario generation and simulation.

a) Initialization of a traffic scenario: SceneGen [21] and SceneControl [14] generate agent bounding boxes, while other approaches [3], [20] generate both agent boxes and lane graphs. However, SLEDGE [3] and DriveSceneGen [20] rely on rasterized images for initialization, resulting in parameter-heavy architectures. In contrast, SceneControl proposes a controllable scene generation framework that produces initial 2D poses and vehicle attributes (e.g., size, speed) using guided diffusion sampling [4]. SceneControl enforces realism and controllability through constraints such as collision avoidance and off-lane penalties. However, without careful balancing of these constraints, performance degrades. In contrast, we aim to generate initial scenes to produce more realistic scenarios without relying on guided sampling.

b) Motion Forecasting: For autonomous vehicles, predicting the motion of non-ego agents is crucial for safe navigation. Consequently, extensive research has focused on trajectory prediction. Similar to next token prediction for language model, Trajeglish [16] formulates trajectory forecasting as a token prediction task using a trajectory codebook. Similarly, MotionDiffuser [12] shows that representing trajectories in a compact and expressive latent space enables efficient and high-quality trajectory generation. Building on MotionDiffuser, OptTrajDiff [25] adapts latent diffusion and introduces Optimal Gaussian Diffusion, which incorporates marginal mode predictions from QCNet [28] as a diffusion prior to reduce the number of diffusion steps. By leveraging QCNet predictions, OptTrajDiff achieves approximately $5\times$ computational efficiency compared to other diffusion-based baselines. However, these approaches rely on past trajectories, which may not be available for all road agents. In their absence, generated scenarios often exhibit unrealistic behaviors, or fail to produce novel and diverse interactions. In contrast, our method does not depend on historical trajectories and instead generates agent behaviors solely from their initial states and map context.

c) End-to-End Model for Traffic Scenario Generation: For end-to-end traffic simulation, TrafficGen [6] uses a Gaussian Mixture Model (GMM) to generate initial placements by sequentially placing agents one by one. Trajectory generation is then treated as a motion-forecasting task, conditioned on the agents' historical context. SceneDiffuser [11] adopts a diffusion-based model to generate full traffic scenes end-to-end, supporting operations such as perturbing scenes while preserving realism, injecting agents, and synthesizing new scenarios with realistic layouts. Additionally, SceneDiffuser introduces a protocol for scalable scenario generation by incorporating constraint specifications derived from Large Language Models (LLMs). However, to generate trajectories for newly initialized agents, it still requires partial scene information, such as the state or history of other agents.

III. Problem Statements and Preliminaries

A. Notations

We begin by defining the initial state of each traffic agent i as a vector $\vec{x}_i = [x_i, y_i, \theta_i, v_i, c_i] \in \mathbb{R}^5$, where $(x_i, y_i) \in \mathbb{R}^2$ denotes the initial 2D position, $\theta_i \in [-\pi, \pi)$ the heading angle, $v_i \in \mathbb{R}_+$ the velocity, and $c_i \in \mathbb{Z}_+$ the agent type. The initial states of all N agents are denoted as $\vec{\mathbf{x}} = \{\vec{x}_1, \ldots, \vec{x}_N\} \in \mathbb{R}^{N\times 5}$. Each agent's trajectory over horizon H is $\tau_i = \{(x, y)_{i,1}, \ldots, (x, y)_{i,H}\}$, and the joint trajectory is $\boldsymbol{\tau} = \{\tau_1, \ldots, \tau_N\} \in \mathbb{R}^{N\times H\times 2}$. Let $\mathcal{M}$ denote the map, a set of 2D map points $m \in \mathbb{R}^2$, and lanes $l \in \mathbb{R}^{K\times 2}$, each represented as an ordered sequence of K such map points. Diffusion steps t are denoted by superscripts, e.g., $\vec{x}_i^t$ with $t \in \{1, 2, \ldots, T\}$. We use $\mathcal{U}[1, T]$ for the uniform distribution over $\{1, \ldots, T\}$, and $\mathcal{N}$ for the Gaussian distribution. The notation $\sim$ indicates sampling, e.g., $\epsilon \sim \mathcal{N}(0, I)$.

B. Problem Statement

Given a map $\mathcal{M}$ and the number of agents in the map, we aim to develop a generative model that produces traffic scenarios aligned with the real-world distributions $q(\vec{\mathbf{x}}^0)$ and $q(\boldsymbol{\tau}^0)$, corresponding to agent initializations and their subsequent trajectories, respectively. To this end, we decompose the problem into two sub-tasks:

Problem 1: (Scene Initialization) Given a map $\mathcal{M}$ and number of agents N, the goal is to learn a generative model that samples initial states $\hat{\vec{\mathbf{x}}} \sim \vec{\mathbf{x}}^0$, such that the generated samples resemble the real data.

Problem 2: (Trajectory Generation) Given a map $\mathcal{M}$ and agent initializations $\vec{\mathbf{x}}^0$, we aim to learn a generative model that produces trajectories $\boldsymbol{\tau}$ consistent with the underlying road structure and behaviors.

C. Diffusion Model

Denoising diffusion probabilistic models (DDPM) [9] are an expressive class of generative models. Latent diffusion

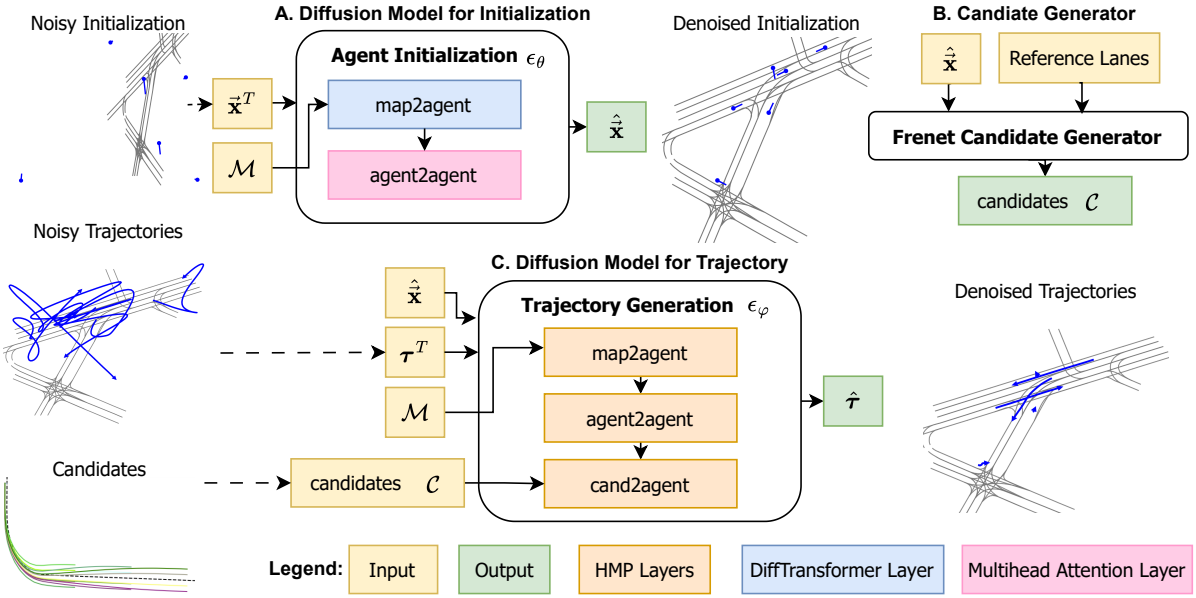

Fig. 1. **Overview of Path-Diffuser**. In **(A)**, Agent initialization takes noisy initialization $\vec{\mathbf{x}}^T$ and predicts the noise $\boldsymbol{\epsilon}_\theta$ (See Algorithm 1). Frenet candidate trajectories are generated in **(B)** based on the initial pose as a prior (details in Algorithm 3). **(C)** Trajectory denoiser takes the initialization and noisy latent trajectory $\boldsymbol{\tau}^T$ and denoises by taking the candidate trajectories.

models [18] extend this framework by learning a generative model in latent space, reducing computational overhead. The diffusion framework transforms the data distribution into an isotropic Gaussian (prior) through a forward process, then iteratively denoises the samples from prior distribution to recover the original data through the reverse process.

Forward Process. Let $q(\vec{\mathbf{x}}^0)$ and $q(\boldsymbol{\tau}^0)$ denote the data distributions of agent initial states and trajectories, respectively. Gaussian noise is incrementally added over T steps according to a variance schedule $\beta_1, \ldots, \beta_T$, such that the samples $\vec{\mathbf{x}}^T, \boldsymbol{\tau}^T$ approach $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The transition kernels at each step are defined as:

$$q(\vec{\mathbf{x}}^t|\vec{\mathbf{x}}^{t-1}) := \mathcal{N}\left(\vec{\mathbf{x}}^t; \sqrt{1-\beta_t}\,\vec{\mathbf{x}}^{t-1}, \beta_t\,\mathbf{I}\right), \quad (1)$$

$$q(\boldsymbol{\tau}^t|\boldsymbol{\tau}^{t-1}) := \mathcal{N}\left(\boldsymbol{\tau}^t; \sqrt{1-\beta_t}\,\boldsymbol{\tau}^{t-1}, \beta_t\,\mathbf{I}\right), \quad (2)$$

By the Markov property of the forward process, the marginal distribution of the noisy samples at any step t can be expressed as a conditional distribution given the original data:

$$q(\vec{\mathbf{x}}^t|\vec{\mathbf{x}}^0) = \mathcal{N}\left(\vec{\mathbf{x}}^t; \sqrt{\bar{\alpha}_t}\,\vec{\mathbf{x}}^0, (1-\bar{\alpha}_t)\,\mathbf{I}\right), \quad (3)$$

$$q(\boldsymbol{\tau}^t|\boldsymbol{\tau}^0) = \mathcal{N}\left(\boldsymbol{\tau}^t; \sqrt{\bar{\alpha}_t}\,\boldsymbol{\tau}^0, (1-\bar{\alpha}_t)\,\mathbf{I}\right), \quad (4)$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Using the reparameterization trick, we can sample noisy input at any t as follows:

$$\vec{\mathbf{x}}^t = \sqrt{\bar{\alpha}_t}\,\vec{\mathbf{x}}^0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\tau}^t = \sqrt{\bar{\alpha}_t}\,\boldsymbol{\tau}^0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad (5)$$

Reverse Process. The joint distribution $p_\theta(\vec{\mathbf{x}}_{0:T})$ defines the reverse process. Starting from the prior distribution $p_\theta(\vec{\mathbf{x}}^T) = \mathcal{N}(\vec{\mathbf{x}}^T; \mathbf{0}, \mathbf{I})$, the model learns to denoise by parameterizing the transitions as $p_\theta(\vec{\mathbf{x}}^{t-1}|\vec{\mathbf{x}}^t) = \mathcal{N}(\vec{\mathbf{x}}^{t-1}; \boldsymbol{\mu}_\theta(\vec{\mathbf{x}}^t, t), \sigma_t^2\mathbf{I})$, where $\sigma_k^2 = \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}$, to approximate the true posterior $q(\vec{\mathbf{x}}^{t-1}|\vec{\mathbf{x}}^t, \vec{\mathbf{x}}^0)$. Similarly, the reverse process for trajectories can be parameterized as $p_\varphi(\boldsymbol{\tau}^{t-1}|\boldsymbol{\tau}^t) = \mathcal{N}(\boldsymbol{\tau}^{t-1}; \boldsymbol{\mu}_\varphi(\boldsymbol{\tau}^t, t), \sigma_t^2\mathbf{I})$. Here, θ and φ denote the learnable parameters.

D. Frenet Frame based Candidate Trajectories

The Frenet-Serret Frame (Frenet Frame) [8] defines a local, path-relative coordinate system in which a vehicle's position is described by the arc-length s, the distance traveled along the reference lane $l_\mathcal{R} \in \mathcal{M}$ and the lateral offset d, representing the deviation from the centerline. This representation is invariant under the action of the special Euclidean group $\mathrm{SE}(2) = \mathrm{SO}(2) \times \mathbb{R}^2$. Naturally adapting to curvy lanes as in Fig. 2, the Frenet Frame simplifies trajectory generation and is widely used in motion planning [10], [23].

A trajectory in the Frenet frame is defined by:

$$s(h) = s_0 + vh, \quad d(h) = d_0 + a_3h^3 + a_4h^4 + a_5h^5 \quad (6)$$

where (s_0, d_0) denotes the initial position in the Frenet frame with respect to the reference lane, h is time step, and a_i are the coefficients of the quintic polynomial that govern the lateral deviation. We assume a constant longitudinal velocity v, which enables efficient exploration over a grid of initial speeds and lateral offsets to generate a diverse set of feasible trajectories originating from the Cartesian point (x_i, y_i).

IV. Path-Diffuser: Approach

An overview of our method is shown in Fig. 1. Building on the concepts introduced in Section III, we now describe our overall framework and its components in detail.

A. Scene Initialization

Given samples from driving logs, our goal is to learn the underlying distribution of agent initializations $\vec{\mathbf{x}}$. To address Problem 1, we adopt a DDPM-based formulation. The reverse diffusion process is intractable, and we therefore train a neural network $\boldsymbol{\epsilon}_\theta(\vec{\mathbf{x}}^t, \mathcal{M}, t) = \frac{\sqrt{1-\bar{\alpha}_t}}{\beta_t}\left(\vec{\mathbf{x}}^t - \sqrt{\alpha_t}\,\boldsymbol{\mu}_\theta(\vec{\mathbf{x}}^t, \mathcal{M}, t)\right)$ to approximate it. This training maximizes the Variational Lower Bound (VLO) [9] $\mathcal{L}_{\mathbb{VLO}} = \mathbb{E}_{q(\vec{\mathbf{x}}^{0:T})}\left[\log \frac{p_\theta(\vec{\mathbf{x}}^{0:T})}{q(\vec{\mathbf{x}}^{1:T}|\vec{\mathbf{x}}^0)}\right]$. We additionally condition the model on map features to guide generation toward valid initializations. In practice, $\mathcal{L}_{\mathbb{VLO}}$ reduces to the following conditional score matching loss:

$$\min_\theta \mathcal{L}_{\mathbb{I}}(\theta) = \mathbb{E}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\vec{\mathbf{x}}^t, \mathcal{M}, t\right)\right\|^2\right] \tag{7}$$

1) Centralized and Decentralized Behavior (CDB): Modeling multi-agent systems requires capturing inter-agent interactions while allowing for decentralized decision-making. Following [29], we regularize learning by alternating between centralized and decentralized modes. In the centralized mode, agents attend to one another via a full attention mask $\mathbf{1}_{n\times n}$ to enable coordinated scene generation. In the decentralized mode, attention is masked using the identity matrix $\mathbf{I}_n$, allowing agents to evolve independently. This stochastic masking strategy is integrated into our training pipeline and reflected in the training objective (Eq. 7) and Line 5 of Algorithm 1.

2) Design Choices: To model map-agent and agent-agent interactions, we follow [14], [28], encoding the vectorized map with point-level features such as normalized location, heading, and lane curvature and pairwise connections to capture map geometry. However, as shown in Table I, standard Multi-Head Attention (MHA) [24] struggles to prioritize relevant tokens in scenarios with dense map components which is reflected in low variance across diffusion steps for map-agent attention. This indicates that the agent attends indiscriminately to map tokens regardless of spatial locality. Conversely, agent-agent attention exhibits high variance, suggesting that the model selectively focuses on a subset of agents. With existence of the vast amount of map tokens, attention patterns result in noisy and unstable distributions that degrade quality.

To address this, we incorporate Differential Transformer (DIFFT) [27], a transformer variant that cancels attention noise, promotes sparsity and enhances semantic relevance. Table I shows that using DIFFT for the map-agent attention significantly increases variance of map-agent attention score, emphasizing local spatial context attention.

TABLE I
ATTENTION SCORE VARIANCE ACROSS DIFFUSION STEPS

Attention Type	Layer	Diffusion Steps 1	201	401
map-agent	MHA	1.83e-4	1.89e-4	1.59e-4
	DIFFT	**1.82e-2**	**1.80e-2**	**1.76e-2**
agent-agent	**MHA**	**2.74e-3**	**2.65e-3**	**2.67e-3**
	DIFFT	1.83e-4	1.89e-4	1.59e-4

Algorithm 1 Agent Initialization Training

1: **Input:** $\mathcal{M}, \vec{\mathbf{x}}^0 \sim q(\vec{\mathbf{x}}^0), \bar{\alpha}_t$
2: **for** each iteration $i = 1$ to $maxiter$ **do**
3: $\quad\gamma \sim$ Bernoulli(0.5), $t \sim \mathcal{U}[0, \ldots, T]$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\quad\vec{\mathbf{x}}^t \leftarrow \sqrt{\bar{\alpha}_t}\vec{\mathbf{x}}^0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
5: $\quad M_{a2a} \leftarrow \gamma\mathbf{1}_{n\times n} + (1-\gamma)\mathbf{I}_n$
6: $\quad\mathcal{L}_{\mathbb{I}}(\theta) \leftarrow ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\vec{\mathbf{x}}^t \odot M_{a2a}, \mathcal{M}, t)||^2$
7: $\quad\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}_{\mathbb{I}}(\theta)$
8: **end for**
9: **return** $\boldsymbol{\epsilon}_\theta$

Algorithm 2 Agent Initialization Sampling

1: **Input:** $\mathcal{M}, N, \vec{\mathbf{x}}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** each iteration $t \in \{T, T-1, \ldots, 1\}$ **do**
3: $\quad\hat{\boldsymbol{\epsilon}} \leftarrow \boldsymbol{\epsilon}_\theta(\vec{\mathbf{x}}^t, \mathcal{M}, t)$
4: $\quad\vec{\mathbf{x}}^0 \leftarrow \frac{\vec{\mathbf{x}}^t - \sqrt{1-\bar{\alpha}_t}\,\hat{\boldsymbol{\epsilon}}}{\sqrt{\bar{\alpha}_t}}$
5: $\quad\vec{\mathbf{x}}^{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\,\vec{\mathbf{x}}^0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\hat{\boldsymbol{\epsilon}}$
6: **end for**
7: **return** $\vec{\mathbf{x}}^0$

3) Order Enforcing: To resolve ambiguity from arbitrary permutations in generated agent initializations, we impose a canonical left-to-right, top-to-bottom ordering and add sinusoidal positional embeddings after input projection. This preserves spatial ordering while allowing permutation-invariant attention when needed.

B. Trajectory Generation

Once agent initializations are obtained, we proceed to generate trajectories. As in Problem 1, we adopt a DDPM-based VLO objective, now applied over a low-dimensional latent representation of trajectories to reduce computational overhead. The denoising process is conditioned on the diffusion step t, map $\mathcal{M}$, agent initializations $\vec{\mathbf{x}}^0$ (Section IV-A), and agent type. To encourage diversity while ensuring realistic behavior, we additionally condition on a set of candidate trajectories $\mathcal{C}$ derived from the Frenet Frame, detailed in the next subsection. The corresponding training objective is given in Equation 8.

1) Latent Representation: While learning directly in trajectory space is possible, it results in (1) increased computational overhead and (2) high variance due to differences in map scale, position, and orientation. To mitigate this, we follow [12] by applying Principal Component Analysis (PCA) [15] for dimensionality reduction on local coordinate frame to enable efficient modeling of long-horizon motion. We denote the resulting latent representation as $\boldsymbol{\tau}_z$.

2) Frenet Candidates: While conditioning on map features alone can model the trajectory distribution, we observe that this often leads to unrealistic outputs e.g., straight-line trajectories that fail to follow curves or respect traffic semantics. To address this, we generate a set of candidate trajectories $\mathcal{C}$ using the Frenet Frame and treat them as motion primitives.

When an agent's initial position $\vec{\mathbf{x}}^0$ is close to multiple lanes, all relevant lanes are included in $l_{\mathcal{R}}$ as illustrated in

Algorithm 3 Frenet Frame Candidate Generation

Input: $(\mathcal{M}_{\mathcal{R}}, \hat{\vec{\mathbf{x}}}) \sim$ Data, initial velocity set $\mathbb{V}$, lateral deviation set $\mathbb{D}$, time interval $\mathbf{h}$
$\mathcal{C} \leftarrow \{\}$
Identify lanes $\mathbf{l}_r$ in which $\hat{\vec{\mathbf{x}}}$ lies
for each iteration $l_{\mathcal{R}} \in \mathbf{l}_{\mathcal{R}}$ **do**
    Compute s_0, d_0 based on $l_{\mathcal{R}}$
    for each iteration $v_i \in \mathbb{V}$, $d_H \in \mathbb{D}$ **do**
        $(\mathbf{s}, \mathbf{d}) \leftarrow (s_0 + v_i\mathbf{h},\ d_0 + d_H\mathbf{h})$
        `#Frenet to Cartesian Coordinate`
        $\boldsymbol{\tau}_f \leftarrow Fren2Cart(\mathbf{s}, \mathbf{d})$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{\tau}_f\}$
    end for
end for
return $\mathcal{C}$

Fig. 2. As detailed in Algorithm 3, diverse trajectory candidates are generated using the initial positions $\vec{\mathbf{x}}^0$, reference lanes $l_{\mathcal{R}}$, and a predefined grid of initial speeds $\mathbb{V}$ and lateral offsets $\mathbb{D}$. These candidates improve trajectory diversity and reduce off-road violations by aligning the predictions with map topology. Similar to agent trajectories, we also transform the candidate trajectories into the latent space.

3) Heterogeneous Message Passing (HMP): The trajectory denoiser must capture both agent-agent and agent-map interactions. Each noisy latent trajectory $\boldsymbol{\tau}_z^t$ attends to its corresponding Frenet candidates $\mathcal{C}$, which are derived from the same initial state. To integrate multi-agent context, we alternate cross-attention and self-attention layers over agents and the vectorized map, enabling rich interaction modeling (see Fig. 1C). The training objective for trajectory generation is then defined as:

$$\min_{\varphi} \mathcal{L}_{\mathbb{T}}(\varphi) = \mathbb{E}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\varphi}\left(\boldsymbol{\tau}_z^t, \mathcal{M}, \vec{\mathbf{x}}^0, \mathcal{C}_z, t\right)\right\|^2\right] \tag{8}$$

Trajectory training and sampling follow the same procedure as scene initialization (Algorithms 1, 2).

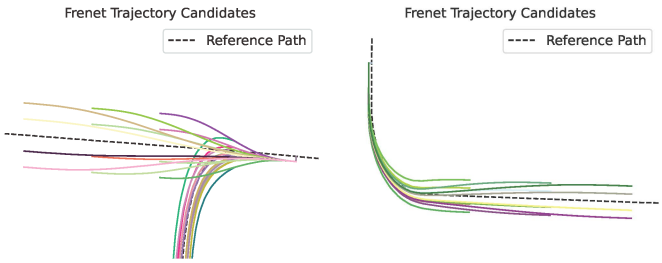

Fig. 2. **Trajectory Candidates.** Candidates (colored) and reference lane (black). Multiple trajectory candidates around a reference path, are created by exploring a set of initial velocities and lateral deviations.

V. Experiments

Our experiments aim to answer the following questions:

- How does PD compare with existing baselines in terms of distributional realism and common-sense metrics for agent initialization?
- How does PD perform compared to baselines on trajectory generation metrics?
- How do the individual components of PD influence overall performance?
- How robust is the framework to map perturbations?

For additional visualizations and high-resolution plots, please refer to our project page.[2]

A. Experimental Setup

1) Dataset: For both agent initialization and trajectory generation, we use the Argoverse 2 Motion Forecasting Dataset [26], a large-scale benchmark for evaluating motion prediction and planning in complex urban environments.

2) Agent Initialization: For agent initialization, the map is scaled so that the region which contains the agents is scaled to -1 to 1. Each agent's initial pose is represented by $\vec{\mathbf{x}}$ and contains position x, y, heading θ and initial speed. The position and initial speed attributes are normalized with minimum and maximum value from the training set.

3) Trajectory Generation: For evaluation, we consider a 6-second window of trajectories for each scene. To assess overall performance, we evaluate the agent initialization and trajectory generation models both independently and jointly, where the output of the initialization model is used to condition the trajectory generation model.

B. Agent Initialization

1) Baselines: We compare against SceneControl [14] , a diffusion-based framework for generating initial traffic scenes, which we refer to as **Multi-Head Attention (MHA)**. This baseline is trained using the objective defined in Equation 7. In contrast, our method **PD** includes additional design components (Section IV-A). We also evaluate an ablated variant, **PD⊖CDB**, without DIFFT and CDB to assess their impact. SceneControl outperforms popular autoregressive baselines such as those presented in [22].

2) Metrics: To evaluate the realism of generated traffic behaviors, we report (1) *Common-sense Metrics*: measured by the **Collision Rate**, which measures if the agents collided, distance to the nearest lane (**Near. Edge**), which measures the distance from the nearest center lane, and **Off-Road Rate**, which shows ratio of agents that are outside of the road. (2) *Distributional Fidelity*: evaluated by using the Jensen-Shannon Divergence (**JSD**) between the generated and ground-truth distributions using histograms over key behavioral attributes as seen in [14]. We report inter-agent spacing via nearest-agent distance (**Near. Dist.**), and inter-space spacing of 5 closest agents (**Local Density**). Moreover, we assess alignment to road geometry via lateral deviation (**Lat. Dev.**) and angular deviation (**Ang. Dev.**). Lastly, we compare the overall speed distribution(**Speed**).

3) Quantitative Results: Table II summarizes our results on the Argoverse 2 dataset. Overall, we observe that **PD** outperforms the baselines across most metrics. Specifically, introducing DIFFT in **PD⊖CDB** reduces the collision rate by

[2] https://github.com/CL2-UWaterloo/PathDiffuser/

TABLE II
AGENT INITIALIZATION RESULTS

	Distributional JSD ↓					Common Sense Metrics ↓		
	Near. Dist.	Local Density	Lat. Dev.	Ang. Dev.	Speed	Collision Rate (%)	Near. Edge (m)	Off-Road Rate (%)
GT	-	-	-	-		0.70	1.42	0.10
MHA	0.29	0.14	0.20	**0.11**	0.04	17.16	**1.56**	**4.26**
PD ⊖ CDB	0.15	0.08	**0.20**	**0.11**	0.14	6.27	1.59	5.70
PD (ours)	**0.15**	**0.08**	**0.21**	0.12	**0.13**	**6.09**	1.59	5.77

TABLE III
TRAJECTORY GENERATION RESULTS: REALISM AND ROAD COMPLIANCE

	No Map Perturbation				Map Perturbation			
	Realism ↓		Road compliance ↓		Realism ↓		Road compliance ↓	
	actorCR (%)	Off-road Rate (%)	Avg. Lat. Dev. (m)	Final Lat. Dev. (m)	actorCR (%)	Off-road Rate (%)	Avg. Lat. Dev. (m)	Final Lat. Dev. (m)
GT	1.82	5.49	12.17	1.13	1.82	5.49	12.75	1.16
VD	2.30	**5.14**	**11.77**	1.63	**2.25**	**4.98**	12.24	2.48
PD ⊖ P	**1.82**	5.67	12.24	**1.25**	2.40	5.52	13.01	1.90
PD (ours)	2.01	5.69	11.78	1.37	2.40	5.40	**12.14**	**1.52**

TABLE IV
TRAJECTORY GENERATION RESULTS: GROUDTRUTH COMPARISON

	No Map Perturbation			Map Perturbation		
	MR ↓ (%)	FDE ↓ (m)	ADE ↓ (m)	MR ↓ (%)	FDE ↓ (m)	ADE ↓ (m)
VD	52.21	21.61	7.84	53.44	23.33	8.32
PD ⊖ P	**41.89**	**5.84**	**1.86**	**47.65**	**8.14**	**2.60**
PD (ours)	43.84	7.16	2.57	48.69	8.63	3.07

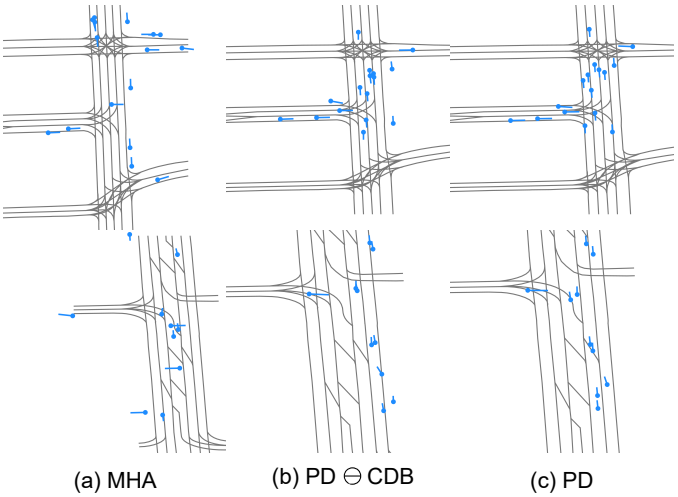

Fig. 3. **Qualitative Results of Agent Initialization**. Two different map to compare the agent initializations. Each dot indicates the initial position and the lines indicates the heading.

approximately 10%, albeit with a slight increase in off-road violations compared to **VD**, affecting map compliance. Further incorporating CDB-based regularization improves distributional metrics. Our variants offer localized spatial attention without relying on costly gradient-based guided sampling.

4) Qualitative Results: Fig. 3 shows the quality of the generated initializations. **PD** demonstrated the best map compliance and collisions compared to PD⊖CDB and MHA. We can see that in **PD⊖CDB**, map compliance improves while the collision still exists. Especially, initializations from **MHA** shows they are off-road and heading angle of some agents do not align with the road.

C. Trajectory Generation

Using the map and agent initializations, we generate trajectories that start from the given initial poses.

1) Baselines: OptTrajDiff [25] is most closely related to our work; however, it leverages agent history and frames the task as trajectory forecasting rather than generation. We refer to such baselines as **Vanilla Diffusion (VD)**, which are trained using the objective in Equation 8. In contrast, our approach **PD** and its ablated variant **PD without Primitives (PD⊖P)** (which omits Frenet candidates) is designed for trajectory generation, as detailed in Section IV-B.

2) Metrics: We evaluate the generated trajectories using the following criteria: (1) *Realism*: measured by the Actor Collision Rate (**actorCR**), which captures collisions between agents over the prediction horizon, and the **Off-road Rate**, which indicates whether trajectories deviate more than 4 meters from the road centerline. (2) *Road Compliance*: assessed by identifying reference lanes from each agent's initial position and computing the Average Lateral Deviation (**Avg. Lat. Dev.**) and Final Lateral Deviation (**Final Lat. Dev.**), which measure lateral deviation from the lane centerline across the prediction horizon H and at the final timestep, respectively. (3) *Ground-Truth Comparison*: we report standard motion forecasting

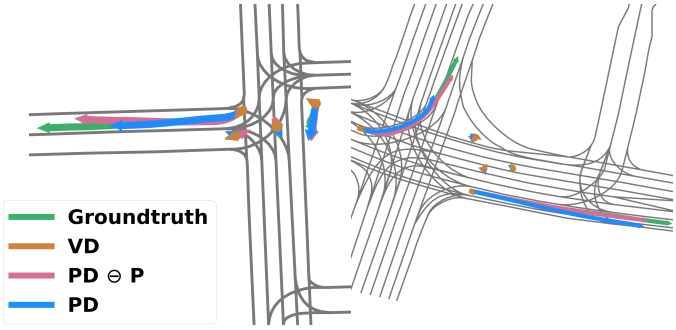

Fig. 4. **Qualitative Results of Trajectory Generation**. Generated trajectories from VD, PD⊖P, and PD from same map and overlayed for comparison.

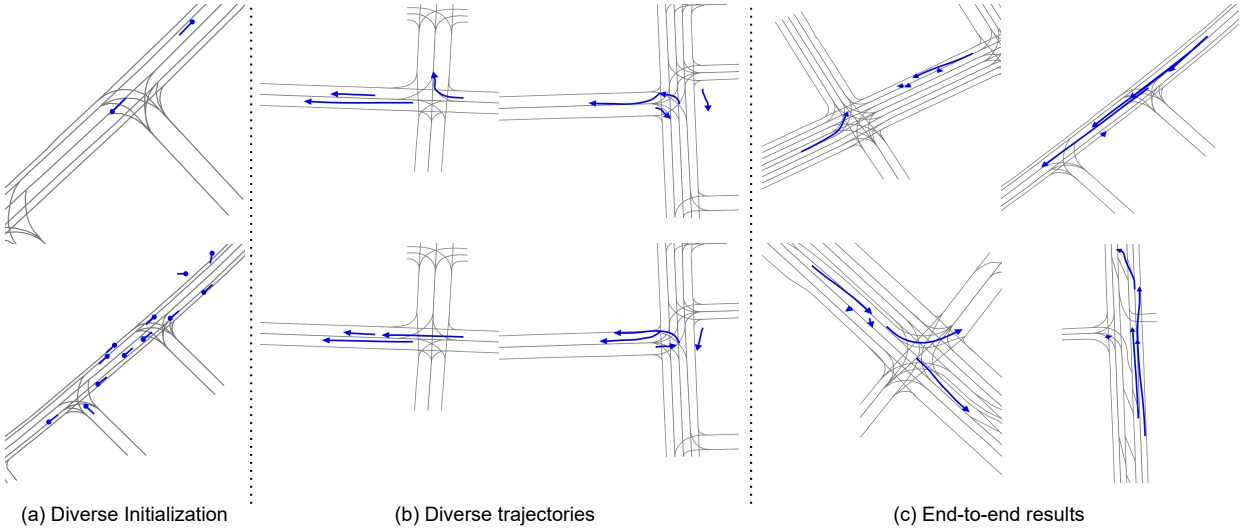

Fig. 5. **Diverse Generation Result**. (a) Initialization visualization with different number of agents in the same map. (b) Trajectories of samples from the PD. (c) End-to-end results which uses the generated initialization for trajectory generation.

metrics, Final Displacement Error (**FDE**), which quantifies the average displacement error at the final timestep; Average Displacement Error (**ADE**), the mean displacement error over the entire horizon H; and Miss Rate (**MR**), defined as the percentage of predicted trajectories whose FDE exceeds 2 meters. Note that we focus on trajectory generation rather than reconstruction. Therefore: (1) Existing autoregressive models rely on agent history encoding, making direct comparison infeasible; hence, they are excluded from our evaluation. (2) For quantitative evaluation, we use ground-truth initialization and provide all models with scene-level context, including agent type and attributes such as initial speed, to enable realistic and context-aware trajectory generation.

3) Quantitative Results: Table III and Table IV, summarizes the results of trajectory generation. We can observe that **PD⊖P** and **PD** shows comparable result when there is no map perturbation. By adding an additional token to VD, the model better captures agent dynamics compared to VD and improved Final Lateral Deviation metric by approximately 0.4 meters. While **VD** shows the best off-road rate, comparison to groundtruth shows that generated trajectories are not close from the groundtruth.

4) Qualitative Results: Fig. 4 shows the how well generated trajectories are aligned to the map and how realistic each trajectories are. As in quantitative results, **PD** and **PD⊖P** generated trajectories that are well aligned to the map structure and does not collide with other agents. However, trajectories generated by **VD** shows most of the trajectories are not moving from its initial position, which can explain why off-road rates are the lowest.

5) Diversity: In Fig. 5, it highlights the model's capacity for producing diverse and varied traffic scenarios. With both agent initialization and trajectory generation models, we can generate novel traffic scenarios given a map which aligns with its structure. Specifically, we seek stochasticity; i.e., the ability to generate multiple plausible traffic scenarios for the same scene. From Tables III and IV, diversity can be quantified by observing higher ground-truth errors and lower map compliance, indicating that the model generates plausible alternatives that deviate from the reference.

D. Robustness to Perturbations

We evaluate the robustness of our model and baselines under map perturbations. As shown in Fig. 6, the vectorized map is modified to test performance on out-of-distribution (OOD) map structures [2]. Under OOD map scenarios, we observe that the PD⊖P struggles due to the limited expressiveness of its low-dimensional PCA-based representations, often producing straight-line trajectories that fail to align with the perturbed map geometry. However, the Frenet candidates remain robust to map perturbations by consistently aligning with the structure of the perturbed lane. We use a guided sampling process [4] to enforce alignment between predicted trajectories and Frenet-frame candidates. At each sampling step, we compute the distance to all candidates and select the closest one as the reference for alignment. Table III (map perturbations) shows that PD outperforms baselines on road compliance metrics, largely due to the use of candidate trajectories generated in the Frenet frame. This conditioning anchors the generation process

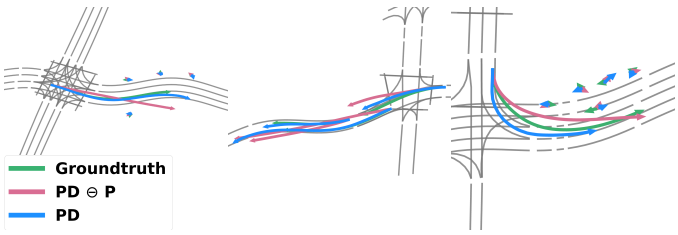

Fig. 6. **Scene-Attack Perturbations.** We leverage the Scene-Attack benchmark [2] to evaluate the robustness of our models under diverse, OOD map structures. The generation result with the map perturbation.

to the underlying map structure, which is also qualitatively evident in Fig. 6.

We observe that using Frenet-based candidates introduces meaningful stochasticity and enables the model to better capture road curvature, in contrast to baseline models that often default to straight-line paths.

VI. Conclusion

We study the problem of generating realistic and diverse traffic scenarios that remain robust under map perturbations, ensuring both semantic consistency and physical plausibility for high-fidelity traffic simulation. We propose Path-Diffuser, a two-stage diffusion framework for agent initialization and trajectory generation. By leveraging DIFFT for canceling attention noise and introducing localized spatial attention, our model significantly reduces collision rates. Furthermore, removing history dependence and incorporating Frenet-frame-based candidates enables diverse trajectory generation beyond simple driving log replay. By decoupling agent initialization from trajectory generation, our method enhances both distributional fidelity and scene controllability. Extensive experiments show that our approach outperforms baselines in realism and diversity, while maintaining robustness to map perturbations.

Limitations and Future Work. Due to the use of PCA-based dimensionality reduction, our current framework is limited to a fixed trajectory horizon of 6 seconds. In future work, we plan to explore more flexible representations that enable sampling of longer trajectories even when trained on shorter sequences. To assess error accumulation and traffic flow, we plan to evaluate the model in a closed-loop setting.

Additionally, we plan to explore diffusion model variants that offer faster convergence and greater test-time controllability. Despite the use of dimensionality reduction, training remains computationally intensive, taking approximately 10 hours. To improve inference efficiency, we propose using Frenet candidates as a learned diffusion prior replacing the standard $\mathcal{N}(\mathbf{0}, \mathbf{I})$ initialization to reduce the number of diffusion steps required. Since the candidate grid is currently predefined, we also plan to investigate its effect in edge cases and its adaptability to complex scenarios.

In the end-to-end process, trajectory generation model relies on the initialization output, making use of two models costly for both training and inference. As a future work, we plan to integrate the two models into an end-to-end framework to improve coherence and efficiency.

Finally, to improve the logical consistency of generated scenes, we intend to incorporate richer contextual cues such as traffic light signals, time of day, and weather conditions.

References

[1] M. Antkiewicz, M. Kahn, M. Ala, K. Czarnecki, P. Wells, A. Acharya, and S. Beiker. Modes of automated driving system scenario testing: Experience report and recommendations. *SAE International Journal of Advances and Current Practices in Mobility*, 2020.

[2] M. Bahari, S. Saadatnejad, A. Rahimi, M. Shaverdikondori, A.-H. Shahidzadeh, S.-M. Moosavi-Dezfooli, and A. Alahi. Vehicle trajectory prediction works, but not everywhere. In *CVPR*, 2022.

[3] K. Chitta, D. Dauner, and A. Geiger. Sledge: Synthesizing driving environments with generative models and rule-based traffic. In *ECCV*, 2024.

[4] P. Dhariwal and A. Q. Nichol. Diffusion models beat GANs on image synthesis. In , *NeurIPS*, 2021.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun. CARLA: an open urban driving simulator. In *CoRL*, 2017.

[6] L. Feng, Q. Li, Z. Peng, S. Tan, and B. Zhou. Trafficgen: Learning to generate diverse and realistic traffic scenarios. In *ICRA*, 2023.

[7] D. J. Fremont, E. Kim, Y. V. Pant, S. A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *ITSC*, 2020.

[8] F. Frenet. Sur les courbes à double courbure. *Journal de mathématiques pures et appliquées*, 1852.

[9] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In , *NeurIPS*, 2020.

[10] J. Huang, Z. He, Y. Arakawa, and B. Dawton. Trajectory planning in frenet frame via multi-objective optimization. *IEEE Access*, 2023.

[11] C. M. Jiang, Y. Bai, A. Cornman, C. Davis, X. Huang, H. Jeon, S. Kulshrestha, J. W. Lambert, S. Li, X. Zhou, C. Fuertes, C. Yuan, M. Tan, Y. Zhou, and D. Anguelov. Scenediffuser: Efficient and controllable driving simulation initialization and rollout. In *NeurIPS*, 2024.

[12] C. M. Jiang, A. Cornman, C. Park, B. Sapp, Y. Zhou, and D. Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *CVPR*, 2023.

[13] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *ITSC*, 2018.

[14] J. Lu, K. Wong, C. Zhang, S. Suo, and R. Urtasun. Scenecontrol: Diffusion for controllable traffic scene generation. In *ICRA*, 2024.

[15] A. Maćkiewicz and W. Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 1993.

[16] J. Philion, X. B. Peng, and S. Fidler. Trajeglish: Traffic modeling as next-token prediction. In *ICLR*, 2024.

[17] E. Pronovost, M. R. Ganesina, N. Hendy, Z. Wang, A. Morales, K. Wang, and N. Roy. Scenario diffusion: Controllable driving scenario generation with diffusion. In *NeurIPS*, 2023.

[18] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[19] L. Rowe, R. Girgis, A. Gosselin, L. Paull, C. Pal, and F. Heide. Scenario dreamer: Vectorized latent diffusion for generating driving simulation environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[20] S. Sun, Z. Gu, T. Sun, J. Sun, C. Yuan, Y. Han, D. Li, and M. H. Ang. Drivescenegen: Generating diverse and realistic driving scenarios from scratch. *IEEE RA-L*, 2024.

[21] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *CVPR*, 2021.

[22] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[23] R. Trauth, K. Moller, G. Würsching, and J. Betz. Frenetix: A high-performance and modular motion planning framework for autonomous driving. *IEEE Access*, 2024.

[24] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[25] Y. Wang, C. Tang, L. Sun, S. Rossi, Y. Xie, C. Peng, T. Hannagan, S. Sabatini, N. Poerio, M. Tomizuka, et al. Optimizing diffusion models for joint trajectory prediction and controllable generation. In *ECCV*, 2025.

[26] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[27] T. Ye, L. Dong, Y. Xia, Y. Sun, Y. Zhu, G. Huang, and F. Wei. Differential transformer. In *ICLR*, 2025.

[28] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang. Query-centric trajectory prediction. In *CVPR*, 2023.

[29] Z. Zhu, M. Liu, L. Mao, B. Kang, M. Xu, Y. Yu, S. Ermon, and W. Zhang. MADiff: Offline multi-agent learning with diffusion models. In *NeurIPS*, 2024.