Optimizing Diffusion Models for Joint Trajectory Prediction and Controllable Generation

Yixiao Wang[1], Chen Tang[1,2], Lingfeng Sun[1], Simone Rossi[3], Yichen Xie[1], Chensheng Peng[1], Thomas Hannagan[3], Stefano Sabatini[3], Nicola Poerio[4], Masayoshi Tomizuka[1], and Wei Zhan[1]

[1] University of California, Berkeley, USA
[2] The University of Texas at Austin, USA
[3] Stellantis, France
[4] Stellantis, Italy
{yixiao_wang, wzhan}@berkeley.edu

Abstract. Diffusion models are promising for joint trajectory prediction and controllable generation in autonomous driving, but they face challenges of inefficient inference steps and high computational demands. To tackle these challenges, we introduce Optimal Gaussian Diffusion (OGD) and Estimated Clean Manifold (ECM) Guidance. OGD optimizes the prior distribution for a small diffusion time T and starts the reverse diffusion process from it. ECM directly injects guidance gradients to the estimated clean manifold, eliminating extensive gradient backpropagation throughout the network. Our methodology streamlines the generative process, enabling practical applications with reduced computational overhead. Experimental validation on the large-scale Argoverse 2 dataset demonstrates our approach's superior performance, offering a viable solution for computationally efficient, high-quality joint trajectory prediction and controllable generation for autonomous driving. Our project webpage is at https://yixiaowang7.github.io/OptTrajDiff_Page/

Keywords: Diffusion model · Autonomous driving · Trajectory Prediction · Controllable Trajectory Generation

1 Introduction

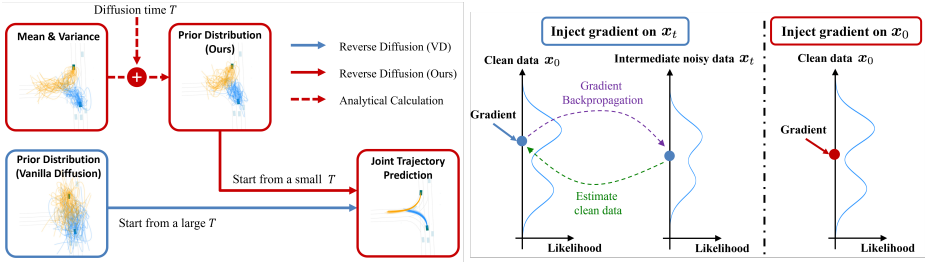
The diffusion model is a class of generative models capable of representing high-dimensional data distributions. In particular, it has demonstrated strong performance in trajectory prediction and generation for autonomous driving [10, 11, 16, 28, 49]. In contrast to traditional trajectory prediction [18, 50] and generative models [41, 45], the unique advantage of diffusion models lies in their ability to deform the generated trajectory distribution to comply with additional requirements at inference stage via gradient-based guided sampling. Notably, it is achieved without extra model training costs. This ability to perform controllable trajectory generation enables various useful applications, such as enforcing additional realism constraints on predicted trajectories, generating directed and user-specified simulation scenarios.

However, computational efficiency is a crucial bottleneck hindering the practical application of diffusion models in autonomous driving. Real-time inference is essential for trajectory prediction, as it provides timely forecasts of surrounding agents' behavior, enabling safe and efficient planning in dynamic traffic scenarios. The high demand for inference speed, coupled with limited onboard computational resources, makes it infeasible to deploy diffusion models for onboard trajectory prediction. While simulations do not occur onboard, lightweight models are still preferred to streamline the closed-loop training and evaluation pipelines. The heavy computational cost is mainly attributed to the following two aspects:

Computationally Expensive Reverse Diffusion. At the inference stage, the diffusion model samples from standard Gaussian distribution and gradually denoises the sample through dynamics described by a Stochastic Differential Equation (SDE) [39], aiming to eventually obtain a sample as if drawn from a target data distribution. The target data distribution can be significantly different from a standard Gaussian distribution, necessitating a large number of denoising steps to yield good performance. Prior works have attempted to reduce the reverse diffusion steps through adaptive noise schedule [17, 32], fast samplers [21,35,43,46], or distillation [31,37]. However, the fixed standard Gaussian prior poses a challenge in accelerating the reverse diffusion process without violating the SDE, which can inevitably compromise the generation quality.

Computationally Expensive Guided Sampling. Controllable generation is typically achieved by guiding the denoising process with the gradient of a guidance cost function. The guidance cost function encodes the desired characteristics of the generated data, which is typically defined on the clean data manifold in trajectory prediction and controllable generation problems. However, guided sampling intends to inject the gradient of the guidance cost function into the series of noisy data at intermediate diffusion steps. It requires a forward pass to estimate the clean data first and then back-propagating throughout the entire network to estimate the gradient with respect to the intermediate noisy data [16, 20, 28], which is very computationally intensive.

Targeted at these challenges, we take a step further to improve the computational efficiency of diffusion models while maintaining their performance for joint trajectory prediction and controllable generation tasks. Specifically, we propose two novel solutions for efficient reverse diffusion and guided sampling respectively. First, we present *Optimal Gaussian Diffusion* (OGD) to accelerate the reverse diffusion process. At the inference stage, instead of a standard Gaussian distribution far away from the desired data distribution, OGD starts from an optimal Gaussian distribution, which minimizes the distance to intermediate data distribution at a specific noise level, cutting down the diffusion steps before that. We show that we can analytically estimate such an optimal Gaussian distribution and also, an optimal perturbation kernel distribution, *at any noise level* from the statistics of the data distribution. It allows flexible tuning of the diffusion steps at the inference stage, without the need to train any additional models [48]. We further derive a practical implementation of OGD for joint trajectory prediction and generation, where the optimal Gaussian prior is computed



(a) Optimal Gaussian Diffusion (OGD) **(b)** Estimated Clean Manifold Guidance (ECM)

Fig. 1: Overview of Optimal Gaussian Diffusion (OGD) and Estimated Clean Manifold Guidance (ECM). **(a)** OGD uses the mean and variance of the data distribution to calculate the optimal prior distribution at a small T. It can largely reduce the diffusion time compared with vanilla diffusion. **(b)** ECM directly injects the gradient of guidance into the clean data manifold to mitigate computational complexity.

using the mean and variance of marginal trajectory distributions estimated with a pre-trained marginal trajectory prediction [18, 24, 50] model.

Second, we propose *Estimated Clean Manifold Guidance* (ECM) to accelerate guided sampling for controllable generation. To save the computational cost due to estimating the guided gradient on the noisy data, we aim to directly inject the gradient into the clean data manifold without lengthy backpropagation. ECM is motivated by the insight that guided sampling can be regarded as a multi-objective optimization problem on the clean data manifold: The first objective is to maximize the likelihood of the samples on the estimated real data distribution; the second objective is to achieve low guidance cost. ECM hierarchically solves this multi-objective problem without backpropagation throughout the entire diffusion model. We show that it leads to faster inference time and much better performance than existing approaches. Also, to tackle the challenges imposed by the multi-modal nature of vehicle interactions, we propose to warm-start the multi-objective optimization problem with reference joint trajectory points estimated using a marginal trajectory predictor. We refer to the complete algorithm as *Estimated Clean Manifold with Reference Joint Trajectory* (ECMR).

To evaluate the proposed OGD and ECM methods on real-world tasks, we implement the OGD model leveraging a pre-trained marginal prediction model, QCNet [50], and conduct extensive experiments on the Argoverse 2 dataset. We show that OGD can achieve better joint trajectory prediction performance than vanilla diffusion with significantly fewer diffusion steps—it only takes $1/12$ of the diffusion steps used by the vanilla diffusion model. OGD also achieves outstanding prediction accuracy compared to non-diffusion joint prediction models. In addition, ECMR, coupled with OGD, can generate samples with significantly lower guidance costs with the same level of realism scores compared to conducting controllable generation on the vanilla diffusion model using existing guided sampling approaches used in autonomous driving [16, 28], but using around $1/5$ of their inference step.

2 Related Work

Diffusion models. Diffusion models have demonstrated their ability to produce high-quality, diverse samples in a variety of applications, such as image, video, and 3D generation [13, 22, 29]. Recently, diffusion models have been applied to trajectory prediction in autonomous driving. It shows great performance on representing future trajectory distribution [10, 16]. However, diffusion models need to run lengthy reverse diffusion processes to generate high-quality samples [5, 10]. This makes it hard to apply diffusion models to trajectory prediction in autonomous driving since it requires in-time prediction for the downstream planning module to promptly respond to dynamically changing traffic scenarios. Previous works [17, 21, 31, 32, 35, 37, 43, 46] mostly focused on mitigating this issue by investigating how to solve reverse SDE in a faster manner. In addition, similar to ours, some works sought alternative initialization of the reverse diffusion process to achieve faster inference. For example, [3] initialized the reverse diffusion process with samples generated by another generative backbone network. However, the backbone network is not deliberately trained for the reverse diffusion process. [48] proposed to learn the initial diffusion with a generative adversarial network (GAN) [8]. However, it requires training a large additional model with a complex training procedure. Also, it requires specifying the reverse diffusion time as a hyperparameter prior to GAN training, which is hard to tune.

Guided sampling. Diffusion models have been successfully used to tackle controllable tasks through guided sampling, such as image inpainting [36] and motion planning [15]. A notable feature of diffusion models with guided sampling is their ability to condition the generation process on the user's preference, which was not available during the training phase. In the driving domain, recent works used guided sampling to generate controllable traffic [16, 49], and controllable pedestrian animation [28]. Our work belongs to the prior category, where user-specified guidance cost is used to guide generation in the trajectory space. In this case, the guidance cost function encodes certain desired properties of the generated trajectories. Thus, it is normally defined on the realistic trajectory samples, which are on the so-called clean manifold, instead of the noisy manifold containing the intermediate noisy data. Some works attempted to learn the guidance cost function on the noisy manifold [15]. Otherwise, it will lead to numerical instability when evaluating guidance cost at intermediate noisy data [28, 49]. To avoid the additional computational costs introduced by a learned guidance cost function, [16, 28] proposed to project the intermediate noisy data into the clean manifold through the diffusion model, and evaluate the guidance cost on the projected point. This approach requires back-propagating throughout the entire diffusion model, which is also computationally intensive.

Trajectory prediction. In autonomous driving [26, 27], it is vital to precisely forecast how other participants in traffic will move in the future so that ego vehicle can plan a safe and efficient trajectory to execute in the future. Marginal trajectory prediction is used to predict the trajectory distribution for single vehicle and recent works involve kinematic constraints of the vehicles, restrictions of complex topology of roads, and interaction from the surrounding

vehicles [18, 24, 50]. Recently, joint trajectory prediction has attracted attention from several researchers. It consists in predicting the joint future trajectories for all agents so that these trajectories are consistent with one another [25], an aspect which marginal trajectory prediction does not consider. Scene-Transformer [25] uses a fixed set of learnable scene embeddings to generate corresponding joint trajectories for all the vehicles in the given scene. Models like M2I [40], and FJMP [30] adopt a conditional approach, predicting the motions of other agents based on the movements of controlled agents. Diffusion model [16] has also been used to predict the joint trajectory. However, joint trajectory prediction is still a challenging problem since the complexity increases exponentially with the number of vehicles in the scene. The efficiency problem becomes more serious when using diffusion model to predict the joint trajectory distribution [16], and largely limits the application of diffusion models in autonomous driving.

3 Preliminaries

3.1 Diffusion Models

The diffusion process continuously perturbs the unknown data distribution p_{data} with a perturbation kernel and generates intermediate data with a given diffusion time T. Denote the distribution of the time-dependent intermediate nosiy data $\boldsymbol{x}_t$ as $p_t(\boldsymbol{x}_t)$, $t \in [0, T]$, where $p_0(\boldsymbol{x}_0) = p_{data}$ is the clean data distribution and $\boldsymbol{x}_0$ is the clean data. The series of intermediate data $\boldsymbol{x}_t$ are generated through the Stochastic Differential Equation (SDE) [39]:

$$\mathrm{d}\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}, \boldsymbol{x}_0 \sim p_{data} = p_0(\boldsymbol{x}_0), \tag{1}$$

where $f(\cdot, t)$ is the drift coefficient, $g(t)$ is the diffusion coefficient, and $\boldsymbol{w}$ is the Wiener process. We can recover p_{data} from reverse-time SDE

$$\mathrm{d}\boldsymbol{x}_t = [f(\boldsymbol{x}_t, t) - g(t)^2 \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)]\mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}}, \boldsymbol{x}_T \sim p_T(\boldsymbol{x}_T), \tag{2}$$

where $\bar{\boldsymbol{w}}$ is another Wiener process, $\mathrm{d}t$ is negative timestep.

In order to solve Eq. (2), we first need to approximate $p_T(\boldsymbol{x}_T)$. In previous works, $p_T(\boldsymbol{x}_T)$ is typically approximated by some prior distributions p_{prior} which contain no information of p_{data}. In this paper, we adopt the setting of Variance Preserving (VP) SDE [12, 39]. In VP-SDE, $p_T(\boldsymbol{x}_T) \approx p_{prior}$ when $T \to \infty$. The perturbation kernels are in the form of $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1 - \bar{\alpha}_t)\boldsymbol{\Sigma}_p)$ where scalar $\bar{\alpha}_t$ is diffusion schedule parameter, $|\boldsymbol{\Sigma}_p| = 1$. A common choice for $\boldsymbol{\Sigma}_p$ is the identity matrix $\mathbf{I}$ [12, 39]. Second, we need to approximate $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ for all $t \in (0, T]$. Some works solve score-matching problem [14, 38] and learn a score function $\boldsymbol{s}_\theta(\boldsymbol{s}_t, t)$ to approximate $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$. In this paper, we follow the practice in DDPM [12] to learn the noise $\boldsymbol{\epsilon}$ using a network $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$:

$$\arg\min_\theta \mathbb{E}_{\boldsymbol{x}_0 \sim p_{data}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)} ||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)||_2^2 \tag{3}$$

where $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) = -\sqrt{1 - \bar{\alpha}_t}\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$. With these two approximation, we can learn $q_\theta(\boldsymbol{x}_0)$ to estimate unknown data distribution $p_0(\boldsymbol{x}_0)$ solving Eq. (2) from $t = T$ to $t = 0$.

3.2 Guided Sampling

In prior diffusion-based controllable generation frameworks [15,16,20,28,49], controllable generation is achieved via biasing of the score function for sampling:

$$\nabla_{\boldsymbol{x}_t} \log \left[p_t(\boldsymbol{x}_t) \exp(-\mathcal{C}(\boldsymbol{x}_t))\right] = \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) - \nabla_{\boldsymbol{x}_t} \mathcal{C}(\boldsymbol{x}_t) \tag{4}$$

where $\mathcal{C}(\cdot)$ is the guidance function. It requires estimating the guidance gradient with respect to the noisy data $\boldsymbol{x}_t$. Some approaches introduce an additional neural network to approximate $\mathcal{C}(\cdot)$ at different noisy levels [4,15,49]. The additional neural network imposes additional training costs and heavier computational burden at the inference stage. To this end, some works define an analytical guidance function $\mathcal{J}(\cdot)$ on the clean data $\boldsymbol{x}_0$. They first estimate $\hat{\boldsymbol{x}}_0 = f_\theta(\boldsymbol{x}_t)$ based on $\boldsymbol{x}_t$ with the diffusion model, then calculate $\mathcal{C}(\boldsymbol{x}_t)$ as $\mathcal{J}(\hat{\boldsymbol{x}}_0)$. However, when taking the gradient of $\mathcal{J}(\hat{\boldsymbol{x}}_0)$ with respect to $\boldsymbol{x}_t$, we get

$$\nabla_{\boldsymbol{x}_t} \mathcal{J}(\hat{\boldsymbol{x}}_0) = \nabla_{\hat{\boldsymbol{x}}_0} \mathcal{J}(\hat{\boldsymbol{x}}_0) \cdot \nabla_{\boldsymbol{x}_t} f_\theta(\boldsymbol{x}_t). \tag{5}$$

Estimating $\nabla_{\boldsymbol{x}_t} f_\theta(\boldsymbol{x}_t)$ requires backpropagating throughout the entire diffusion model, i.e., $f_\theta(\boldsymbol{x}_t)$. It requires heavy computing resources and GPU memory.

3.3 Trajectory Prediction and Controllable Generation

Joint trajectory prediction aims to predict the future joint trajectories $\boldsymbol{x}_0$ for all the vehicles in the scene, conditioned on context information $\boldsymbol{c}$. It can be regarded as a conditional generation task where the goal is to train a generative model to approximate the distribution $p_0(\boldsymbol{x}_0|\boldsymbol{c})$. For simplicity, we omit $\boldsymbol{c}$ and represent $p_0(\boldsymbol{x}_0|\boldsymbol{c})$ as $p_0(\boldsymbol{x}_0)$. We denote n as the number of vehicles in the same scene and $\boldsymbol{x}_{0,i}$ as the future trajectory for vehicle i, $i \in \{1, 2, ..., n\}$, so $\boldsymbol{x}_0 = [\boldsymbol{x}_{0,1}, \boldsymbol{x}_{0,2}, ..., \boldsymbol{x}_{0,n}]$. Joint trajectory prediction can be very challenging. The complex interactions among vehicles, especially in highly interactive and dense traffic, result in a complicated high-dimensional $p_0(\boldsymbol{x}_0)$, which is difficult to accurately model with lightweight and computationally efficient models. A simplified solution is to approximately decompose the joint trajectory distribution into marginal ones, i.e., $p_0(\boldsymbol{x}_0) \approx \prod_i^n p_0(\boldsymbol{x}_{0,i})$. It leads to the marginal trajectory prediction task, which has been extensively studied with mature solutions [18,24,50]. One drawback is that it omits the interactions among vehicles in the predicted horizon, which leads to large errors in highly interactive scenes.

Controllable trajectory generation is closely related to trajectory prediction. In addition to generating realistic trajectory samples resembling the ground-truth $p_0(\boldsymbol{x}_0)$, controllable generation imposes an additional objective—the generated trajectories should comply with specified guidance cost function $\mathcal{J}(\cdot)$. We term the former objective as *realism* and the latter as *guidance effectiveness*. The guidance cost function $\mathcal{J}(\cdot)$ can be some goal points of the vehicles, kinematic constraints, etc, which are mostly defined on the clean manifold rather than on the noisy data when it comes to diffusion-based controllable generation.

4 Methodology

4.1 Optimal Gaussian Diffusion

As discussed in Sec. 3.1, diffusion models typically select a non-informative prior distribution p_{prior}, such as a standard Gaussian, as the initial data distribution for the reverse diffusion process. Such a non-informative p_{prior} is reasonable since $p_T(\boldsymbol{x}_T)$ converges to it when $T \to \infty$. However, it also means that a large T is required at inference time to yield good performance, which undermines computational efficiency and limits its wide real-time applications in autonomous driving. In this section, we aim to investigate a practical solution to tackle this challenge for joint trajectory prediction. Given the inherent limitation imposed by a non-informative prior, the key question we look into is: *can we instead adopt an informative prior so that we can obtain the same level of performance with much smaller reverse diffusion steps?*

First, we still consider a Gaussian prior, but with learnable parameters, i.e., we parameterize p_{prior} as $q_\phi(T) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are learnable. We aim to optimize $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to enhance the generation performance for small T. We got inspiration from [5], where an upper bound of the Kullback–Leibler divergence of the clean data distribution $p_0(\boldsymbol{x}_0)$ and learned distribution $q_\theta(\boldsymbol{x}_0)$ was derived as a function of the diffusion time T:

$$\mathrm{KL}[p_0(\boldsymbol{x}_0)||q_\theta(\boldsymbol{x}_0)] \leq \mathcal{G}(\boldsymbol{x}_\theta, T) + \mathrm{KL}[p_T(\boldsymbol{x}_T)||p_{prior}] \tag{6}$$

where $\mathcal{G}(\boldsymbol{x}_\theta, T)$ is the positive accumulated error between $\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}_t)$ and $\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$ from 0 to T [5]. Note that $\mathcal{G}(\boldsymbol{x}_\theta, T)$ is an accumulated error so $\mathcal{G}(\boldsymbol{x}_\theta, T_1) \leq \mathcal{G}(\boldsymbol{x}_\theta, T_2)$, $T_1 \leq T_2$. If we can achieve lower $\mathrm{KL}[p_{T_1}(\boldsymbol{x}_{T_1})||q_\phi(T_1)]$, then a tighter upper bound can be obtained. This opens up the possibility to achieve better performance with less diffusion time. Thus, we propose to optimize the prior distribution by minimizing $\mathrm{KL}[p_T(\boldsymbol{x}_T)||q_\phi(T)]$. As shown in Proposition 1 (See proof in Appendix A), it turns out that the optimal μ and Σ can be expressed analytically as functions of the ground-truth data statistics. In addition, we find that we can further minimize the target KL divergence if we set a learnable $\boldsymbol{\Sigma}_p$ in the perturbation kernel $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{\Sigma}_p)$, whose optimal value can also be expressed as a function of the data statistics.

Proposition 1. *Denote $\boldsymbol{\mu}_d$ and $\boldsymbol{\Sigma}_d$ as the mean and variance of p_{data}. Denote $\boldsymbol{\Sigma}^*(i,j)$ and $\boldsymbol{\Sigma}_p^*(i,j)$ as the element at ith row and jth column of matrix $\boldsymbol{\Sigma}^*$ and $\boldsymbol{\Sigma}_p^*$. The optimal solution to* $\min KL[p_T(\boldsymbol{x}_T)||q_\phi(\boldsymbol{x}_T, T)]$ *is*

$$\begin{cases} \boldsymbol{\mu}^* \approx \sqrt{\bar{\alpha}_T}\boldsymbol{\mu}_d \\ \boldsymbol{\Sigma}_p^*(i,j) \approx \frac{1}{|\boldsymbol{\Sigma}_d|}\boldsymbol{\Sigma}_d(i,j) \\ \boldsymbol{\Sigma}^*(i,j) \approx \bar{\alpha}_T\boldsymbol{\Sigma}_d + (1-\bar{\alpha}_T)^2\boldsymbol{\Sigma}_p^* = (\bar{\alpha}_T^2 + \frac{(1-\bar{\alpha}_T)^2}{|\boldsymbol{\Sigma}_d|})\boldsymbol{\Sigma}_d(i,j) \end{cases} \tag{7}$$

Thus, if we are able to estimate the mean and variance of the ground-truth data distribution, we can then analytically determine the optimal prior distribution $\mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ and the optimal perturbation kernel variance $\boldsymbol{\Sigma}_p^*$ at *any noise*

level $T \in [0, \infty)$. It leads to a crucial advantage against prior efforts on prior learning [48], where a target noise level has to be determined at training time in order to train an additional neural network to represent the prior distribution at the pre-selected noise level. In contrast, our method enables flexible tuning of the number of diffusion steps at inference time, without additional training costs. Since the learnable $q_\phi(T)$ and perturbation kernel are both Gaussian, we refer to our proposed diffusion framework as Optimal Gaussian Diffusion (OGD).

For joint trajectory prediction and generation, we need to estimate the mean and variance of the joint trajectory distribution $p_0(\boldsymbol{x}_0)$. It is not straightforward as only a limited number of trajectory samples exist in the dataset under the same context. Considering there exists mature and accurate marginal trajectory prediction models [18, 24, 50], we can conveniently extract statistics of the marginal trajectory distributions, i.e., $p_0(\boldsymbol{x}_{0,i}), i \in \{1, 2, ..., n\}$ from a pre-trained marginal trajectory predictor. Since Proposition 1 provides element-wise optimal value, we can easily get:

Corollary 1. *Denote* $\boldsymbol{\mu}_d(\boldsymbol{x}_{0,i})$ *and* $\boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,i})$ *as the mean and variance of marginal distribution for vehicle* i *and set both* $\boldsymbol{\Sigma}$ *and* $\boldsymbol{\Sigma}_p$ *are block-diagonal matrices where each block represents the marginal characteristics of one single vehicle. The optimal solution to* $\min KL[p_T(\boldsymbol{x}_T)||q_\phi(\boldsymbol{x}_T, T)]$ *is*

$$\begin{cases} \boldsymbol{\mu}^* = [\sqrt{\bar{\alpha}_T}\boldsymbol{\mu}_d(\boldsymbol{x}_{0,1}), ..., \sqrt{\bar{\alpha}_T}\boldsymbol{\mu}_d(\boldsymbol{x}_{0,n})]^T \\ \boldsymbol{\Sigma}_p^* = \frac{1}{\sum_{i=1}^n |\boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,i})|} diag[\boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,1}), ..., \boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,n})] \\ \boldsymbol{\Sigma}^* = \bar{\alpha}_T diag[\boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,1}), ..., \boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,n})] + (1 - \bar{\alpha}_T)^2 \boldsymbol{\Sigma}_p^* \end{cases} \tag{8}$$

Corollary 1 implies that, if we further confine $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}_p$ to be block-diagonal without covariance between the states of different vehicles, then we can determine their optimal values purely from the estimated marginal statistics $\boldsymbol{\mu}_d(\boldsymbol{x}_{0,i})$ and $\boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,i})$, $i \in \{1, 2, ..., n\}$, which enables a practical implementation of the proposed OGD model for joint trajectory prediction and generation tasks. Specifically, for vehicle i, we leverage a pre-trained marginal trajectory predictor [18, 24, 50], predict diverse marginal trajectory sample set $\mathcal{R}_i = \{\boldsymbol{r}_i^l\}_{l=1}^L$ and corresponding likelihood set $\{p(\boldsymbol{r}_i^l)\}_{l=1}^L$, and estimate $\boldsymbol{\mu}_d(\boldsymbol{x}_{0,i})$ and $\boldsymbol{\Sigma}_d(\boldsymbol{x}_{0,i})$. For example, $\boldsymbol{\mu}_d(\boldsymbol{x}_{0,i})$ can be estimated as $\frac{1}{L}\sum_{l=L} p(\boldsymbol{r}_i^l)\boldsymbol{r}_i^l$.

4.2 Estimated Clean Manifold Guidance with Reference

As discussed in Sec. 3.2, the intensive computation required for guided sampling comes from the calculation of $\nabla_{\boldsymbol{x}_t} f_\theta(\boldsymbol{x}_t)$. Previous guided sampling approaches bias the score function defined on the intermediate noisy data $\boldsymbol{x}_t$. In this section, we aim to investigate whether we can inject the gradient directly into $\boldsymbol{x}_0$ to avoid the gradient propagation process. We first reformulate the controllable generation as a multi-objective optimization problem directly over $\boldsymbol{x}_0$ and propose an iterative algorithm to solve the formulated problem. In addition, we use reference trajectory points to create the region of interest, which helps solve the local optimal problem caused by multi-modal joint trajectory distribution and accelerate the guided sampling process.

Estimated Clean Manifold Guidance. The objective of controllable generation of sample $\boldsymbol{x}_0$ includes two different objectives. The most important is the negative likelihood, ensuring the sample lies in the clean manifold. The second important is the guidance cost representing the user preference on the generated sample $\boldsymbol{x}_0$. This multi-objective optimization problem can be represented as

$$\min_{\boldsymbol{x}_0} \quad [-\log q_\theta(\boldsymbol{x}_0), \mathcal{J}(\boldsymbol{x}_0)]^T \tag{9}$$

Inspired by lexicographic optimization [33], we solve this multi-objective optimization problem hierarchically. The main idea is to optimize each objective in the order of importance regardless of the degradation of the other less significant objectives. We first optimize the most important objective, $-\log q_\theta(\boldsymbol{x}_0)$, to generate realistic and high-likelihood samples. The diffusion model achieves this goal effectively by reversing the diffusion process from noisy samples at specific noise level. However, exact noise level of current sample $\boldsymbol{x}_0$ is unknown. To address this, we inject noise at level t into $\boldsymbol{x}_0$, and then denoise it from t with learned diffusion model. This approach, similar to *noise injection and denoising* [1,23,37], improves the desired sample quality. We iteratively repeat this process K times, injecting guidance at each iteration to strengthen user preference.

Specifically, denote $\boldsymbol{x}_0(k)$ as the sample at iteration k. We first regenerate high-likelihood sample $\hat{\boldsymbol{x}}_0(k)$ by diffusion model:

$$\hat{\boldsymbol{x}}_0(k) \leftarrow \mathbb{E}[q_\theta(\boldsymbol{x}_0(k)|\boldsymbol{x}_{t_k}(k))] = \frac{1}{\sqrt{\bar{\alpha}_{t_k}}}(\boldsymbol{x}_{t_k} - \sqrt{1-\bar{\alpha}_{t_k}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{t_k}(k), t_k)), \tag{10}$$

where $\boldsymbol{x}_{t_k}(k) = \sqrt{\bar{\alpha}_{t_k}}\boldsymbol{x}_0(k) + \sqrt{1-\bar{\alpha}_{t_k}}\boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)$. $t_k \in [0, T]$ is a tunable parameter. Then we minimize the guidance cost function with a small degradation of the most important objective $-\log q_\theta(\boldsymbol{x}_0)$,

$$\boldsymbol{x}_0(k-1) \leftarrow \hat{\boldsymbol{x}}_0(k) - \zeta \nabla_{\hat{\boldsymbol{x}}_0(k)} \mathcal{J}(\hat{\boldsymbol{x}}_0(k)) \tag{11}$$

where ζ is the step size. Small degradation of $-\log q_\theta(\boldsymbol{x}_0)$ is realized by one-step gradient update and proper step size ζ. See the derivation of the optimization process and the parameter tuning in Appendix B.

During the iterations, $\boldsymbol{x}_0(k)$, $\forall k \in \{0, 1, ..., K-1\}$ are not exactly on the clean manifold. We minimize its negative log-likelihood through diffusion model, resulting in it lying on an estimated clean manifold. Thus, we call our method Estimated Clean Manifold (ECM) Guidance.

Reference Joint Trajectories. To generate trajectories with low guidance cost, we are essentially searching low-cost trajectories within the high-likelihood region. At the same time, joint trajectory distribution is a multi-modal distribution resulting from road topologies and different decision variables, meaning the likelihood has multiple peaks. This leads to the optimal solution of Problem 9 having multiple local optimals, and each optimal is far away from the other. Guided sampling methods [15,16,28,49], including our method ECM, suffer from

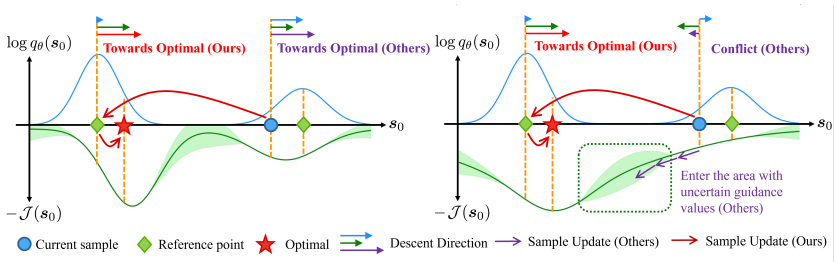


Fig. 2: Two challenges with multi-peak function optimization: 1) Gradients may lead to suboptimal local optima (left); 2) There exist regions with low likelihood but high guidance cost uncertainty, leading to instability (right). Our approach can bypass the lengthy paths between peaks, search for better optima, and avoid uncertain areas.

two challenges (See Fig. 2): 1) it can be trapped at the local optimal around the initial position; 2) it takes massive efforts to drag the sample from one peak to another and transferring from one modal to another will need to pass through the region of low-likelihood (off clean-manifold), leading to numerical instability.

To overcome this, we generate high-likelihood reference joint trajectories, choose the best one as the initialization. Note that combinations of samples with high marginal likelihood tend to exhibit high joint likelihood. Therefore, we can utilize the marginal sample set $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{n}$ obtained from pre-trained marginal models to generate the references. Specifically, for iteration k, we construct candidate joint trajectory set $\mathcal{R} \bigotimes \hat{\boldsymbol{x}}_0(k) = \{[\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_n] | \boldsymbol{w}_i \in \mathcal{R}_i \cup \{\hat{\boldsymbol{x}}_{0,i}\}, i = 1, 2, ..., n\}$. We calculate the guidance cost of all possible combinations, $\mathcal{J}(w), w \in \mathcal{R} \bigotimes \hat{\boldsymbol{x}}_0(k)$, and choose the minimal-cost one as the reference. The guided sampling algorithm, ECM with reference joint trajectories (ECMR), is introduced in Algorithm 1.

Algorithm 1: ECMR

Input: $\mathcal{J}(\cdot)$, $\{\bar{\alpha}_t, \beta_t\}_{t=0}^{T-1}$, $\{t_k\}_{k=0}^{K-1}$
$\boldsymbol{x}_0(K) \sim \mathcal{N}(0, \boldsymbol{\Sigma}_p)$
for $k = K-1, ..., 1$ **do**
  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_p)$
  $\boldsymbol{x}_{t_k}(k) = \sqrt{\bar{\alpha}_{t_k}}\boldsymbol{x}_0(k) + \sqrt{1-\bar{\alpha}_{t_k}}\boldsymbol{\epsilon}$
  $\hat{\boldsymbol{x}}_0(k) = \frac{1}{\sqrt{\bar{\alpha}_{t_k}}}(\boldsymbol{x}_{t_k}(k) - \sqrt{1-\bar{\alpha}_{t_k}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{t_k}(k), t))$
  $\hat{\boldsymbol{x}}_0(k) = \arg\min \mathcal{J}(w), w \in \mathcal{R} \bigotimes \hat{\boldsymbol{x}}_0(k)$
  $\boldsymbol{x}_0(k-1) \leftarrow \hat{\boldsymbol{x}}_0(k) - \zeta \nabla_{\hat{\boldsymbol{x}}_0(k)} \mathcal{J}(\hat{\boldsymbol{x}}_0(k))$
end for
Output: $\boldsymbol{x}_0(0)$

5 Experiments

5.1 Experimental setup

Dataset. We use Argoverse 2 [44], a widely used and large-scale trajectory prediction dataset, to test the effectiveness of our approaches for joint trajectory prediction and controllable generation. It has a large observation window of 5s and a long prediction horizon of 6s.

Implementation Details. We use the fixed scene context encoder of pre-trained QCNet [50] to extract compact and representative context features from context information $\boldsymbol{c}$. Then, we utilize a cross-attention layer to update the intermediate noisy data $\boldsymbol{x}_t$ with multiple contexts, including the history encodings of the target agent, the map encodings, the neighboring agents' encodings. Inspired by [7], we also add a cross-attention layer to update $\boldsymbol{x}_t$ with the diverse marginal trajectory samples $\boldsymbol{r}_i^l$ and its corresponding likelihood. In addition, we use self-attention to allow the interaction between $\boldsymbol{x}_{t,i}$ and $\boldsymbol{x}_{t,j}$. Then the model predicts the noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$. According to [16], compact trajectory representation helps the diffusion model to generate high-quality trajectories efficiently. Inspired by this, we also learn a linear mapping between the 10-dimensional latent and 120-dimensional trajectories. Similar to [16], we design a rapid sample clustering algorithm so that we can generate a representative joint trajectory set. To increase the efficiency of sampling, we use DDIM [35] to accelerate the inference, and the DDIM step stride is 10. See Appendix C for details and analysis.

5.2 Joint trajectory prediction

We now evaluate OGD for joint trajectory prediction. Given K joint trajectories, the evaluation metrics are 1) $\textbf{avgMinFDE}_K$/$\textbf{avgMinADE}_K$: the average of lowest final/average displacement error (FDE/ADE) of joint trajectory samples; 2) $\textbf{actorMR}_K$: the rate of trajectory predictions that are considered to be "missed" (>2m FDE) in the lowest minFDE joint trajectory samples; 3) $\textbf{actorCR}_K$: the rate of collisions across "best" (lowest avgMinFDE) joint trajectory samples; 4) $\textbf{avgBrierMinFDE}_K$: calculated similarly to avgMinFDE_K but scaled by the probability score of joint trajectory samples. We denote metrics with superscript "*" as those after sample clustering (see Appendix C.3).

As a baseline, we train a vanilla diffusion (VD) baseline that shares the same neural network architecture as OGD. Specifically, we train an Optimal Gaussian Diffusion model with diffusion time $T_{train} = 100$ (OGD), vanilla diffusion with two different diffusion times $T_{train} = 100, 500$ (VD_{100}, VD_{500}). Note that T is denoted as the diffusion time from which the reverse diffusion process starts. First, during the inference, we change T and evaluate OGD and VD_{100} who have the same $T_{train} = 100$. Figure 3 shows that, with the decrease of the reverse steps, avgMinFDE_{128} of OGD keeps lower than VD_{100} and it is more stable with the change of T. It is also interesting to note that the performance of OGD even becomes better in the early stage. We hypothesize that it is because that $\text{KL}[p_T(\boldsymbol{x}_T)||p_{prior}]$ of OGD is small and the accumulated error of $\mathcal{G}(\boldsymbol{x}_\theta, T)$ is sufficiently reduced by a small T. It also shows an advantage of OGD, which is that it is easy for OGD to tune a suitable T for reverse diffusion without training an additional model for every T [48]. Table 1 shows that OGD outperforms VD_{500} with only 40 diffusion steps.

We also compare our OGD with the other state-of-the-art methods on the Argoverse multi-world leaderboard; our approach OGD ranks 4^{th} on the leaderboard ranked by avgBrierMinFDE_K, which demonstrates the effectiveness of our

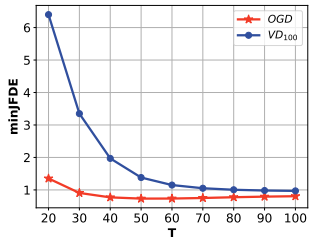


Fig. 3: Evaluation of Optimal Gaussian Diffusion and vanilla diffusion over reverse steps T.

Table 1: Evaluation on joint trajectory prediction task. For each metric, the best result is in **bold** and the second best result is underlined. $T = 70$ is the best T from Fig. 3. $T = 40$ is the minimal diffusion time when OGD outperforms VD_{500} on all metrics.

Model	T	avgMinFDE$^*_6$	avgMinADE$^*_6$	avgMinFDE$_{128}$	avgMinADE$_{128}$
VD_{100}	100	0.62	1.38	0.49	0.97
VD_{500}	500	0.61	1.36	0.48	0.91
OGD	100	0.60	1.32	0.43	0.81
OGD	70	**0.59**	**1.31**	**0.42**	**0.75**
OGD	40	0.61	1.34	0.47	0.77

OGD framework. Note that we only list the entries with publications or technical reports in Tab. 2. Please refer to the official website for the full leaderboard.

5.3 Controllable Generation

Tasks. Future behaviors of vehicles can be effectively can be effectively represented by a set of goal points [9, 47] such as acceleration, braking, and right or left turn. Generating such diverse modes of joint trajectories is a typical motivation for using controllable generation. Thus, we study the controllable generation task where the goal is to reach diverse goal points at a specific time to fully test our guided sampling method. Considering that goal points should lie in realistic routes, and the trajectories to reach such goal points should also lie in such routes, we generate such target goal points in some realistic routes $\text{Route}(\tau_g)$. The guidance cost function can be expressed as

$$\mathcal{J}(\boldsymbol{x}_0) = \frac{1}{n}||\text{Position}(\boldsymbol{x}_0, \tau_d) - \text{Route}(\tau_g)||_2^2, \tag{12}$$

where $\text{Position}(\boldsymbol{x}_0, \tau_d)$ is the positions of $\boldsymbol{x}_0$ at time τ_d. We choose ground-truth trajectories and a random combination set of diverse marginal samples, i.e., $\mathcal{U} = \{[\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n] | \boldsymbol{u}_i \in \mathcal{R}_i, i = 1, 2, ..., n\}$ as the realistic routes. We denote

Table 2: Quantitative results on the Argoverse 2 Multi-world Forecasting leaderboard. For each metric, the best result is in **bold** and the second best result is underlined.

Model	avgMinFDE$^*_6$	avgMinFDE$^*_1$	actorMR$^*_6$	avgMinADE$^*_6$	avgMinADE$^*_1$	avgBrierMinFDE$^*_6$	actorCR$^*_6$
QCXet [51]	**1.02**	**2.29**	**0.13**	**0.50**	**0.94**	**1.65**	**0.01**
Gnet [6]	1.46	3.05	0.19	0.69	1.23	2.12	0.01
Forecast-MAE [2]	1.55	3.33	0.19	0.69	1.30	2.24	0.01
FJMP [30]	1.89	4.00	0.23	0.81	1.52	2.59	0.01
OGD (Ours)	1.31	2.71	0.17	0.60	1.08	1.95	0.01

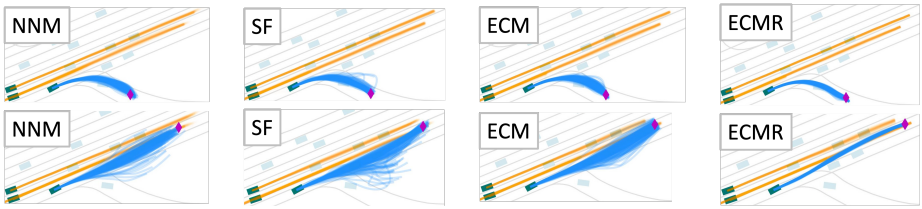


Fig. 4: Evaluation on controllable generation: route set **U** and **Deceleration**. Magenta diamonds represent goal points. In the first (second) row, goal points are set at the fork lane (right lane). NNM [49] and SF [16, 28] struggle to drag samples from one modal to another. Our methods can achieve better guidance effectiveness and realism.

the former as **GT** set and the latter as **U** set. The underlying assumption is that diverse samples from a good marginal trajectory predictor are realistic. We design different velocity settings to cover the diverse controllable generation tasks in autonomous driving: first is **Normal Speed** (**N**), $\tau_d = \tau_g = 6s$; second is **Acceleration** (**A**), $\tau_d = 5s < \tau_g = 6s$; third is **Deceleration** (**D**), $\tau_d = 6s < \tau_g = 5s$.

Metrics. We use the following metrics to evaluate controllable generation performance: Joint Route Deviation Error (JRDE), which measures the displacements to the routes to evaluate the realism, and Joint Final Displacement Error (JFDE), which evaluates the guidance effectiveness. We also evaluate from the "min" and "mean" perspectives: The "min" metric considers the best sample's performance, while the "mean" metric assesses the ratio of valid samples.

Baselines. Guided sampling in controllable generation is mainly divided into two approaches: the first is to directly calculate $\nabla_{\boldsymbol{x}_t}\mathcal{J}(\boldsymbol{x}_t)$ [49]; the second is to calculate $\nabla_{\boldsymbol{x}_t}\mathcal{J}(\hat{\boldsymbol{x}}_0)$ [16, 28]. We denote the former as Next Noisy Mean Guidance (NNM) and the latter as Score Function Guidance (SF). For a fair comparison, we use one guidance step followed by one DDIM step. We also tune the gradient step size for different guided sampling with Optimal Gaussian Diffusion and vanilla diffusion and report the results with the optimal step size. See Appendix D for the details of baseline derivation and step size tuning. We evaluate the following experiments with 128 joint trajectory samples.

Evaluation. First, we evaluate the performance and efficiency with our diffusion model (OGD) and guided sampling method (ECM and ECMR) in Tab. 3, which demonstrates our methods can generate more realistic and effective samples with 5 times less DDIM steps. In addition, with reference joint trajectories, ECM significantly improves 'mean' metrics, indicating it addresses the issues discussed in Sec. 4.2 to a certain extent. Second, we compare solely on different guided sampling methods using the same OGD model shown in Fig. 5. Our ECM achieves better performance both in guidance effectiveness and realism. In Fig. 4, our ECM can generate trajectories that reach goal points more closely than NNM and SF. And with reference joint trajectory, ECMR can easily move samples from one modal to another . Third, we evaluate the average inference time and average GPU memory usage in Tab. 4. Our methods can generate

Table 3: Evaluation on controllable generation: route set **U** and **Deceleration**. For each metric, the best result is in **bold** and the second best result is underlined.

Model	Sampling	DDIM Steps	Guidance Effectiveness		Realism	
			minJFDE	meanJFDE	minJRDE	meanJRDE
VD_{500}	No Guid	50	1.961	5.229	0.165	0.492
VD_{500}	NNM [49]	50	0.778	2.913	0.130	0.309
VD_{500}	SF [16, 28]	50	0.538	2.339	0.158	0.500
OGD	No Guid	10	1.772	5.172	0.138	0.469
OGD	ECM (Ours)	10	0.072	0.237	0.128	0.236
OGD	ECMR (Ours)	10	**0.053**	**0.146**	**0.110**	**0.154**

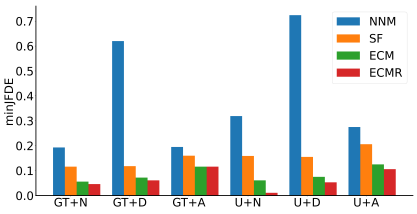


Fig. 5: Evaluation of different guided sampling methods on various tasks. See Appendix E for other metrics.

Table 4: We evaluate the average inference time per step and GPU incremental memory on **U** + **Deceleration**. We test on single RTX A6000, batch size is 16 and number of samples is 128.

Sampling	minJFDE	minJRDE	time(ms)	memory (GB)
NNM [49]	0.724	0.119	113	3.21
SF [16, 28]	0.155	0.126	247	7.96
ECM(Ours)	0.075	0.116	111	3.21
ECMR(Ours)	0.053	0.110	116	3.22

realistic trajectories satisfying guidance quite well with low inference time and GPU memory usage. More results on controllable generation can be found in Appendix E.

6 Conclusion

In this work, we introduce Optimal Gaussian Diffusion (OGD) and Estimated Clean Manifold (ECM) Guidance to significantly improve the computational efficiency and performance of diffusion models in autonomous driving. These methodologies enable a substantial reduction in inference steps and computational demands while ensuring enhanced joint trajectory prediction and controllable generation capabilities. Our approaches and experimental results underscore the potential of diffusion models for real-time applications in dynamic environments, marking a pivotal advancement in the deployment of diffusion models for autonomous driving. One limitation of the current implementation is that the performance is affected by the accuracy of marginal trajectory predictors. Enhancements could be achieved with superior marginal models or by directly learning joint predictions' mean and variance, areas for future work.

Acknowledgement

This work was supported by Berkeley DeepDrive. [5]

References

1. Avrahami, O., Lischinski, D., Fried, O.: Blended diffusion for text-driven editing of natural images. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 18208–18218 (2022)
2. Cheng, J., Mei, X., Liu, M.: Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8679–8689 (2023)
3. Chung, H., Sim, B., Ye, J.C.: Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12413–12422 (2022)
4. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in neural information processing systems **34**, 8780–8794 (2021)
5. Franzese, G., Rossi, S., Yang, L., Finamore, A., Rossi, D., Filippone, M., Michiardi, P.: How much is enough? a study on diffusion times in score-based generative models. Entropy **25**(4), 633 (2023)
6. Gao, X., Jia, X., Li, Y., Xiong, H.: Dynamic scenario representation learning for motion forecasting with heterogeneous graph convolutional recurrent networks. IEEE Robotics and Automation Letters **8**(5), 2946–2953 (2023)
7. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Thomas: Trajectory heatmap output with learned multi-agent sampling. arXiv preprint arXiv:2110.06607 (2021)
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
9. Gu, J., Sun, C., Zhao, H.: Densetnt: End-to-end trajectory prediction from dense goal sets. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15303–15312 (2021)
10. Gu, T., Chen, G., Li, J., Lin, C., Rao, Y., Zhou, J., Lu, J.: Stochastic trajectory prediction via motion indeterminacy diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17113–17122 (2022)
11. Guo, Z., Gao, X., Zhou, J., Cai, X., Shi, B.: Scenedm: Scene-level multi-agent trajectory generation with consistent diffusion models. arXiv preprint arXiv:2311.15736 (2023)
12. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
13. Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., Fleet, D.J.: Video diffusion models (2022)
14. Hyvärinen, A., Dayan, P.: Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research **6**(4) (2005)
15. Janner, M., Du, Y., Tenenbaum, J.B., Levine, S.: Planning with diffusion for flexible behavior synthesis. arXiv preprint arXiv:2205.09991 (2022)

[5] https://deepdrive.berkeley.edu

16. Jiang, C., Cornman, A., Park, C., Sapp, B., Zhou, Y., Anguelov, D., et al.: Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9644–9653 (2023)
17. Kingma, D., Salimans, T., Poole, B., Ho, J.: Variational diffusion models. Advances in neural information processing systems **34**, 21696–21707 (2021)
18. Lan, Z., Jiang, Y., Mu, Y., Chen, C., Li, S.E., Zhao, H., Li, K.: Sept: Towards efficient scene representation learning for motion prediction. arXiv preprint arXiv:2309.15289 (2023)
19. Laumanns, M., Thiele, L., Zitzler, E.: An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. European Journal of Operational Research **169**(3), 932–942 (2006)
20. Lin, H., Wang, Y., Huo, M., Peng, C., Liu, Z., Tomizuka, M.: Joint pedestrian trajectory prediction through posterior sampling. arXiv preprint arXiv:2404.00237 (2024)
21. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. Advances in Neural Information Processing Systems **35**, 5775–5787 (2022)
22. Luo, S., Hu, W.: Diffusion probabilistic models for 3d point cloud generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2837–2845 (2021)
23. Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: Sdedit: Guided image synthesis and editing with stochastic differential equations (2022), https://arxiv.org/abs/2108.01073
24. Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K.S., Sapp, B.: Wayformer: Motion forecasting via simple & efficient attention networks. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 2980–2987. IEEE (2023)
25. Ngiam, J., Caine, B., Vasudevan, V., Zhang, Z., Chiang, H.T.L., Ling, J., Roelofs, R., Bewley, A., Liu, C., Venugopal, A., et al.: Scene transformer: A unified architecture for predicting multiple agent trajectories. arXiv preprint arXiv:2106.08417 (2021)
26. Peng, C., Wang, G., Lo, X.W., Wu, X., Xu, C., Tomizuka, M., Zhan, W., Wang, H.: Delflow: Dense efficient learning of scene flow for large-scale point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16901–16910 (2023)
27. Peng, C., Xu, C., Wang, Y., Ding, M., Yang, H., Tomizuka, M., Keutzer, K., Pavone, M., Zhan, W.: Q-slam: Quadric representations for monocular slam. arXiv preprint arXiv:2403.08125 (2024)
28. Rempe, D., Luo, Z., Bin Peng, X., Yuan, Y., Kitani, K., Kreis, K., Fidler, S., Litany, O.: Trace and pace: Controllable pedestrian animation via guided trajectory diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13756–13766 (2023)
29. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
30. Rowe, L., Ethier, M., Dykhne, E.H., Czarnecki, K.: 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. pp. 13745–13755 (2023)

31. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512 (2022)
32. San-Roman, R., Nachmani, E., Wolf, L.: Noise estimation for generative diffusion models. arXiv preprint arXiv:2104.02600 (2021)
33. Sherali, H.D., Soyster, A.L.: Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples. Journal of Optimization Theory and Applications **39**, 173–186 (1983)
34. Shi, S., Jiang, L., Dai, D., Schiele, B.: Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. arXiv preprint arXiv:2306.17770 (2023)
35. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
36. Song, J., Vahdat, A., Mardani, M., Kautz, J.: Pseudoinverse-guided diffusion models for inverse problems. In: International Conference on Learning Representations (2022)
37. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. arXiv preprint arXiv:2303.01469 (2023)
38. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems **32** (2019)
39. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 (2020)
40. Sun, Q., Huang, X., Gu, J., Williams, B.C., Zhao, H.: M2i: From factored marginal trajectory prediction to interactive prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6543–6552 (2022)
41. Suo, S., Regalado, S., Casas, S., Urtasun, R.: Trafficsim: Learning to simulate realistic multi-agent behaviors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10400–10409 (2021)
42. Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K.S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C.P., Anguelov, D., et al.: Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 7814–7821. IEEE (2022)
43. Watson, D., Chan, W., Ho, J., Norouzi, M.: Learning fast samplers for diffusion models by differentiating through sample quality. arXiv preprint arXiv:2202.05830 (2022)
44. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., et al.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. arXiv preprint arXiv:2301.00493 (2023)
45. Xu, D., Chen, Y., Ivanovic, B., Pavone, M.: Bits: Bi-level imitation for traffic simulation. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 2929–2936. IEEE (2023)
46. Zhang, Q., Chen, Y.: Fast sampling of diffusion models with exponential integrator. arXiv preprint arXiv:2204.13902 (2022)
47. Zhao, H., Gao, J., Lan, T., Sun, C., Sapp, B., Varadarajan, B., Shen, Y., Shen, Y., Chai, Y., Schmid, C., et al.: Tnt: Target-driven trajectory prediction. In: Conference on Robot Learning. pp. 895–904. PMLR (2021)
48. Zheng, H., He, P., Chen, W., Zhou, M.: Truncated diffusion probabilistic models. arXiv preprint arXiv:2202.09671 (2022)

49. Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., Pavone, M.: Guided conditional diffusion for controllable traffic simulation. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 3560–3566. IEEE (2023)
50. Zhou, Z., Wang, J., Li, Y.H., Huang, Y.K.: Query-centric trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17863–17873 (2023)
51. Zhou, Z., Wen, Z., Wang, J., Li, Y.H., Huang, Y.K.: Qcnext: A next-generation framework for joint multi-agent trajectory prediction (2023)

A Proof of Proposition 1

In this section, we prove the Proposition 1.

Proof. We would like to minimize $\mathrm{KL}[p_T(\boldsymbol{x}_T)||q_\phi(T)]$ in the VP-SDE [39] setting. We expand VP-SDE setting into the setting when the perturbation kernel is in the format of $p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; a_t\boldsymbol{x}_0 + b_t, c_t^2\boldsymbol{\Sigma}_p)$. For VP-SDE, $a_t = \sqrt{\bar{\alpha}_t}$, $b_t = 0$, $c_t = 1 - \bar{\alpha}_t$. For Variance Exploding (VE) SDE [39], $a_t = 1$, $b_t = 0$, $c_t = \sigma_t^2$. In the following content, we are going to minimize $\mathrm{KL}[p_T(\boldsymbol{x}_T)||q_\phi(T)]$ in a more general form when $\boldsymbol{x}_T = a_T\boldsymbol{x}_0 + b_T + \boldsymbol{\epsilon}_T$ and $\mathrm{Var}[\boldsymbol{\epsilon}_T] = c_T^2\boldsymbol{\Sigma}_p$. Then, we use the results on VP-SDE.

Minimizing $\mathrm{KL}[p_T(\boldsymbol{x}_T)||q_\phi(T)]$ is equivalent to maximizing the log likelihood $\max_{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\Sigma}_p} \mathbb{E}[\log q_\phi(\boldsymbol{x}_T, T)]$, where $q_\phi(\boldsymbol{x}_T, T)$ is denoted as the probability density of $q_\phi(T) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ at $\boldsymbol{x}_T$. Assume the dataset contains N data points, which are $\boldsymbol{x}_0^i, i = 1, 2, ..., N$. And $\boldsymbol{x}_T^i = a_T\boldsymbol{x}_0^i + b_T + \boldsymbol{\epsilon}_T^i, i = 1, 2, ..., N$. Thus, $\mathbb{E}[\log q_\phi(\boldsymbol{x}_T, T)]$ can be approximated as $\frac{1}{N}\sum_{i=1}^N \log q_\phi(\boldsymbol{x}_T^i, T)$, denoted as $l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p)$. Our optimization problem is formulated as

$$\begin{aligned} \max_{\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{\Sigma}_p} \quad & l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p) \\ s.t. \quad & |\boldsymbol{\Sigma}_p| = 1 \end{aligned} \tag{13}$$

We formulate the Lagrange function

$$\begin{aligned} h(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p, \lambda) &= l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p) + \lambda(\log|\boldsymbol{\Sigma}_p| - \log 1) \\ &= l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p) + \lambda\log|\boldsymbol{\Sigma}_p| \end{aligned} \tag{14}$$

First, we express explicitly $l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p)$ as

$$\begin{aligned} l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p) =& \frac{1}{N}\sum_{i=1}^N \log q_\phi(\boldsymbol{x}_T^i, T) \\ =& -\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2N}\sum_{i=1}^N (\boldsymbol{x}_T^i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_T^i - \boldsymbol{\mu}) \\ =& -\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2N}\sum_{i=1}^N (a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu}) \\ & - \frac{1}{N}\sum_{i=1}^N (a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\epsilon}_T^i - \frac{1}{2N}\sum_{i=1}^N {\boldsymbol{\epsilon}_T^i}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\epsilon}_T^i \end{aligned} \tag{15}$$

Take derivative with respect to $\boldsymbol{\mu}$, we get

$$\begin{aligned} \frac{\partial h(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p, \lambda)}{\partial\boldsymbol{\mu}} =& \frac{1}{N}\sum_{i=1}^N (a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1} + \frac{1}{N}\sum_{i=1}^N {\boldsymbol{\epsilon}_T^i}^T\boldsymbol{\Sigma}^{-1} \\ =& \frac{1}{N}\sum_{i=1}^N (a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1} = 0 \end{aligned} \tag{16}$$

In order to take derivative with respect to $\boldsymbol{\Sigma}$, we reformulate $l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p)$ as

$$\begin{aligned}
l(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p) =& \frac{1}{2}\log|\boldsymbol{\Sigma}^{-1}| - \frac{1}{2N}\sum_{i=1}^{N}\mathrm{tr}[(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}] \\
& - \frac{1}{N}\sum_{i=1}^{N}\mathrm{tr}[\boldsymbol{\epsilon}_T^i(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}] - \frac{1}{2N}\sum_{i=1}^{N}\mathrm{tr}[\boldsymbol{\epsilon}_T^i{\boldsymbol{\epsilon}_T^i}^T\boldsymbol{\Sigma}^{-1}] \\
=& \frac{1}{2}\log|\boldsymbol{\Sigma}^{-1}| - \frac{1}{2N}\mathrm{tr}[(\sum_{i=1}^{N}(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T)\boldsymbol{\Sigma}^{-1}] \\
& - \frac{1}{N}\mathrm{tr}[(\sum_{i=1}^{N}\boldsymbol{\epsilon}_T^i(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T)\boldsymbol{\Sigma}^{-1}] - \frac{1}{2N}\mathrm{tr}[\sum_{i=1}^{N}(\boldsymbol{\epsilon}_T^i{\boldsymbol{\epsilon}_T^i}^T)\boldsymbol{\Sigma}^{-1}] \\
\approx& \frac{1}{2}\log|\boldsymbol{\Sigma}^{-1}| - \frac{1}{2N}\mathrm{tr}[(\sum_{i=1}^{N}(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T)\boldsymbol{\Sigma}^{-1}] \\
& - \mathrm{tr}[\mathrm{Cov}[\boldsymbol{\epsilon}, \boldsymbol{x}_T - \boldsymbol{\mu}]\boldsymbol{\Sigma}^{-1}] - \frac{1}{2}\mathrm{tr}[c_T^2\boldsymbol{\Sigma}_p\boldsymbol{\Sigma}^{-1}] \\
\approx& \frac{1}{2}\log|\boldsymbol{\Sigma}^{-1}| - \frac{1}{2N}\mathrm{tr}[(\sum_{i=1}^{N}(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T)\boldsymbol{\Sigma}^{-1}] \\
& - \frac{c_T^2}{2}\mathrm{tr}[\boldsymbol{\Sigma}_p\boldsymbol{\Sigma}^{-1}]
\end{aligned} \tag{17}$$

where $\mathrm{Cov}[\cdot,\cdot]$ is the covariance matrix of two random variables and $\mathrm{Cov}[\boldsymbol{x}_T - \boldsymbol{\mu}, \boldsymbol{\epsilon}_T] \approx 0$ because $\boldsymbol{\epsilon}_T$ is a independent random variable.

Then take derivative of $\boldsymbol{\Sigma}^{-1}$, we get:

$$\frac{\partial h(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p, \lambda)}{\partial \boldsymbol{\Sigma}^{-1}} \approx \frac{1}{2}\boldsymbol{\Sigma} - \frac{1}{2N}\sum_{i=1}^{N}(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})(a_T\boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T - \frac{c_T^2}{2}\boldsymbol{\Sigma}_p = 0 \tag{18}$$

Take derivative of $\boldsymbol{\Sigma}_p$

$$\begin{aligned}
\frac{\partial h(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p, \lambda)}{\partial \boldsymbol{\Sigma}_p} \approx& \frac{\partial[-\frac{c_T^2}{2}\mathrm{tr}[\boldsymbol{\Sigma}_p\boldsymbol{\Sigma}^{-1}] + \lambda\log|\boldsymbol{\Sigma}_p|]}{\partial \boldsymbol{\Sigma}_p} \\
=& \frac{\partial[-\frac{c_T^2}{2}\mathrm{tr}[\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}_p] + \lambda\log|\boldsymbol{\Sigma}_p|]}{\partial \boldsymbol{\Sigma}_p} \\
=& -\frac{c_T^2}{2}\boldsymbol{\Sigma}^{-1} + \lambda\boldsymbol{\Sigma}_p^{-1} = 0
\end{aligned} \tag{19}$$

Take derivative of λ

$$\frac{\partial h(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}_p, \lambda)}{\partial \lambda} = \log|\boldsymbol{\Sigma}_p| = 0 \tag{20}$$

In this way, we get

$$\begin{cases} \boldsymbol{\mu}^* = b_T + a_T \frac{1}{N} \sum_{i=1}^N \boldsymbol{x}_0^i \\ \boldsymbol{\Sigma}^* \approx \frac{1}{N} \sum_{i=1}^N (a_T \boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})(a_T \boldsymbol{x}_0^i + b_T - \boldsymbol{\mu})^T + c_T^2 \boldsymbol{\Sigma}_p^* \\ \boldsymbol{\Sigma}_p^* \approx \frac{2\lambda}{c_T^2} \boldsymbol{\Sigma}^* \\ |\boldsymbol{\Sigma}_p^*| = 1 \end{cases} \tag{21}$$

Given that $\boldsymbol{\mu}_d \approx \frac{1}{N} \sum_{i=1}^N \boldsymbol{x}_0^i$ and $\boldsymbol{\Sigma}_d \approx \frac{1}{N} \sum_{i=1}^N (\boldsymbol{x}_0^i - \boldsymbol{\mu}_d)(\boldsymbol{x}_0^i - \boldsymbol{\mu}_d)^T$, we have $\boldsymbol{\Sigma}^* \approx a_T^2 \boldsymbol{\Sigma}_d + c_T^2 \boldsymbol{\Sigma}_p^*$ and

$$\begin{cases} \boldsymbol{\mu}^* \approx b_T + a_T \boldsymbol{\mu}_d \\ \boldsymbol{\Sigma}_p^* \approx \frac{1}{|\boldsymbol{\Sigma}_d|} \boldsymbol{\Sigma}_d \\ \boldsymbol{\Sigma}^* \approx a_T^2 \boldsymbol{\Sigma}_d + c_T^2 \boldsymbol{\Sigma}_p^* = (a_T^2 + \frac{c_T^2}{|\boldsymbol{\Sigma}_d|}) \boldsymbol{\Sigma}_d \end{cases} \tag{22}$$

where the approximation becomes increasingly close to an equality when N is very large.

Denote $\boldsymbol{\Sigma}^*(i,j)$ and $\boldsymbol{\Sigma}_p^*(i,j)$ as the element at ith row and jth column of matrix $\boldsymbol{\Sigma}^*$ and $\boldsymbol{\Sigma}_p^*$, and we can have element-wise optimal values for $\boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}_p$,

$$\begin{cases} \boldsymbol{\mu}^* \approx b_T + a_T \boldsymbol{\mu}_d \\ \boldsymbol{\Sigma}_p^*(i,j) \approx \frac{1}{|\boldsymbol{\Sigma}_d|} \boldsymbol{\Sigma}_d(i,j) \\ \boldsymbol{\Sigma}^*(i,j) \approx a_T^2 \boldsymbol{\Sigma}_d + c_T^2 \boldsymbol{\Sigma}_p^* = (a_T^2 + \frac{c_T^2}{|\boldsymbol{\Sigma}_d|}) \boldsymbol{\Sigma}_d(i,j) \end{cases} \tag{23}$$

The element-wise version is important because perturbation models typically use a diagonal matrix as the diffusion kernel, implying that $\boldsymbol{\Sigma}_p$ is a diagonal matrix. The element-wise version can handle not only diagonal matrix diffusion kernels but also any other variance matrix diffusion kernels. In a multi-agent setting, if we disentangle the relationships between different agents in the latent space and allow the model to learn these relationships, the variance matrix can be a block diagonal matrix. This means that there are correlations within each agent, but no correlation exists between agents.

Apply this to VP-SDE and we get Proposition 1 and Corollary 1.

B Derivation of ECM and Parameter Tuning

B.1 Derivation of ECM

As discussed in Sec. 4.2, we solve the multi-objective problem 9 by K iterations. In this section, we derive the optimization process for each iteration.

At iteration $k \in \{0, 1, ..., K-1\}$, we first minimize the most important objective $-\log q_\theta(\boldsymbol{x}_0(k))$ where $\boldsymbol{x}_0(k)$ is the intermediate sample at iteration k. Since our ultimate goal in optimizing $-\log q_\theta(\boldsymbol{x}_0(k))$ is to ensure that the sample lies on the clean manifold, we can employ a diffusion model to regenerate the high-likelihood sample based on the current sample $\boldsymbol{x}_0(k)$. We add the noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)$ to the current sample $\boldsymbol{x}_0(k)$ and get $\boldsymbol{x}_{t_k}(k) = \sqrt{\bar{\alpha}_{t_k}} \boldsymbol{x}_0(k) + \sqrt{1 - \bar{\alpha}_{t_k}} \boldsymbol{\epsilon}$

where $t_k \in [0, T]$ is a tunable parameter. In order to obtain $q_\theta(\boldsymbol{x}_0(k)|\boldsymbol{x}_{t_k}(k))$, we can we can iteratively run the reverse diffusion process until $t = 0$. However, this approach is time-consuming. Therefore, we use Tweedie's formula to perform a one-step estimation. Specifically, $q_\theta(\boldsymbol{x}_0(k)|\boldsymbol{x}_{t_k}(k))$ can be approximated as a Gaussian distribution and its mean has the highest likelihood. Thus, given $\boldsymbol{x}_{t_k}(k)$, we can update $\boldsymbol{x}_0(k)$ to the highest likelihood position, i.e., $\mathbb{E}[q_\theta(\boldsymbol{x}_0(k)|\boldsymbol{x}_{t_k}(k))]$,

$$\hat{\boldsymbol{x}}_0(k) \leftarrow \mathbb{E}[q_\theta(\boldsymbol{x}_0(k)|\boldsymbol{x}_{t_k}(k))] = \frac{1}{\sqrt{\bar{\alpha}_{t_k}}}(\boldsymbol{x}_{t_k} - \sqrt{1-\bar{\alpha}_{t_k}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{t_k}(k), t_k)) \tag{24}$$

Then, inspired by ϵ-Constraint method [19], we minimize the guidance cost under the constraint of small degradation of the primary objective value,

$$\begin{aligned} \min_{\boldsymbol{x}_0(k-1)} \quad & \mathcal{J}(\boldsymbol{x}_0(k-1)) \\ s.t. \quad & -\log q_\theta(\boldsymbol{x}_0(k-1)) < -\log q_\theta(\hat{\boldsymbol{x}}_0(k)) + \gamma \end{aligned} \tag{25}$$

where γ is a small positive scalar. We assume that if the deviation of $\boldsymbol{x}_0(k-1)$ from $\hat{\boldsymbol{x}}_0(k)$ is small enough, the constrain in Problem 25 can be satisfied. Thus, we propose a simple approach to solve this problem by one gradient update

$$\boldsymbol{x}_0(k-1) \leftarrow \hat{\boldsymbol{x}}_0(k) - \zeta \nabla_{\hat{\boldsymbol{x}}_0(k)} \mathcal{J}(\hat{\boldsymbol{x}}_0(k)) \tag{26}$$

where ζ is the step size.

B.2 Parameter tuning and analysis

In this section, we analyze the impact of t_k, where $t_k \in [0, T]$, and injected noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)$ on primary objective: the negative log-likelihood.

First, we examine the impact of t_k where $t_k \in [0, T]$. We first inject noise at level t_k and denoise it from t_k using learned diffusion model. Generally, a larger t_k but highly distinguished samples will negate the guidance efforts made in previous steps. Moreover, a larger t_k reduces sample quality because we use a one-step estimation, and the estimation error increases with a larger t_k. Conversely, a smaller t_k introduces minimal noise, facilitating denoising to a high-likelihood sample with only a few denoising steps. Thus, updating with mean of Tweedie's formula is more accurate. Therefore, we design t_k to decrease as k decrease, enhancing diversity in the initial stage and refining the sample in the final stage.

Second, we analyze the impact of injected noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_p)$ in $\boldsymbol{x}_{t_k}(k) = \sqrt{\bar{\alpha}_{t_k}}\boldsymbol{x}_0(k) + \sqrt{1-\bar{\alpha}_{t_k}}\boldsymbol{\epsilon}$. Note that $\boldsymbol{\epsilon}$ at different iterations are independent with each other, which injects excessive stochasticity. More importantly, directly injecting stochastic noise at each iteration also diffuse the guidance from previous iterations, resulting in lower guidance effectiveness. Inspired by DDIM [35], we can choose to use $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_{t_{k+1}}(k+1), t_{k+1})$ as the deterministic noise $\boldsymbol{\epsilon}$ at iteration k. We use Estimated Clean Manifold (ECM) Guidance with the optimal step

Table 5: Comparison of different choices of injected noise. **GT** denotes the ground truth route set. **N** signifies normal speed, **D** indicates deceleration, and **A** stands for acceleration.

Task	Noise	Guidance Effectiveness		Realism	
		minJFDE	meanJFDE	minJRDE	meanJRDE
GT+N	Stochastic	0.129	0.416	0.101	0.218
	Deterministic	0.056	0.209	0.113	0.232
GT+D	Stochastic	0.145	0.468	0.124	0.223
	Deterministic	0.072	0.237	0.128	0.236
GT+A	Stochastic	0.444	0.979	0.139	0.218
	Deterministic	0.117	0.438	0.137	0.232

size tuned in Appendix D.2, and compare the performance differences when injecting stochastic versus deterministic noise. The results are shown in Tab. 5. We find that ECM with stochastic noise can generate samples as realistic as that with deterministic noise, but the guidance is excessively diffused. We use the deterministic injected noise in this paper.

C Additional Implementation Details and Analysis

In this section, we begin by detailing the network structure and parameters such as the diffusion schedule parameters. Subsequently, we describe the process of learning compact trajectory representations through Linear Mapping (LM). Next, we introduce the sample clustering algorithm utilized in this study. Finally, we compare the influence of different pre-trained marginal trajectory predictors.

C.1 Network Structure and parameters

We use the fixed scene context encoder of pre-trained QCNet [50] to extract compact and representative context features from context information $\boldsymbol{c}$. Then, we utilize a cross-attention layer to update the intermediate noisy data $\boldsymbol{x}_t$ with multiple contexts, including the history encodings of the target agent, the map encodings, the neighboring agents' encodings. Inspired by [7], we also add a cross-attention layer to update $\boldsymbol{x}_t$ with the diverse marginal trajectory samples $\boldsymbol{r}_i^l$ and its corresponding likelihood (generated by pre-trained QCNet [50]). In addition, we use self-attention to allow the interaction between $\boldsymbol{x}_{t,i}$ and $\boldsymbol{x}_{t,j}$. Then the model predicts the noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$.

For the diffusion parameters, we set $\beta_0 = 0.0001$ and $\beta_T = 0.05$ for all the models we trained. And then we calculate $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{t=0}^{T} \alpha_t$. We train the models for 64 epochs.

For all experiments, we generate 128 samples by default. To facilitate comparison with other state-of-the-art methods on the Argoverse 2 multi-world leaderboard, we generate 2048 joint trajectory samples.

C.2 Compact Latent Representation

According to [16], compact and expressive latent representation helps the diffusion model to generate high-quality trajectories efficiently. In this paper, we learn a low-dimensional latent of the trajectory then apply diffusion model in the latent space. For clearer presentation, we use the notation $\boldsymbol{x}$ in this section to represent the trajectory. Denote the latent as $\boldsymbol{z} \in \mathbb{R}^{Z\times 1}$ and the mapping functions between $\boldsymbol{x}$ and $\boldsymbol{z}$ are $\boldsymbol{x} = \mathcal{F}(\boldsymbol{z})$ and $\boldsymbol{z} = \mathcal{G}(\boldsymbol{x})$.

First, we minimize the reconstruction loss, $\mathcal{L}_{rec}$:

$$\mathcal{L}_{rec} = ||\mathcal{F}(\mathcal{G}(\boldsymbol{x})) - \boldsymbol{x}||_2^2 \tag{27}$$

Second, we constrain mapping between between $\boldsymbol{x}$ and $\boldsymbol{z}$ should be distance-preserving transformations. The intuition behind is that the latent diffusion model aims to minimize expected error in latent space, though the ultimate objective is reducing error in trajectory space. Consequently, it disproportionately prioritizes samples with significant latent space errors, which may not correspond to those with substantial errors in trajectory space. Thus, we minimize this regularization term $\mathcal{L}_{reg}$:

$$\mathcal{L}_{reg} = (\boldsymbol{x}^T\boldsymbol{x} - \mathcal{G}(\boldsymbol{x})^T\mathcal{G}(\boldsymbol{x}))^2 \tag{28}$$

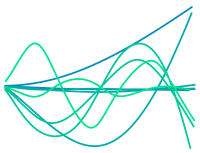
In addition, extreme variance in different dimension of the latent will cause instability of the training. Thus, we minimize the variance term $\mathcal{L}_{var}$

$$\mathcal{L}_{var} = ||\text{std}(\mathcal{G}(\boldsymbol{x})) - \mathbf{v}||_2^2 \tag{29}$$

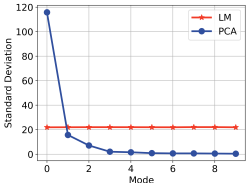
where $\mathbf{v} \in \mathbb{R}^{Z\times 1}$ whose elements are all η, a learnable variable.

To further improve the efficiency during the inference, we need to find a mapping $\mathcal{G}(\cdot)$ as simple as possible. [16] shows that PCA can have relatively low reconstruction error, and it is only a linear mapping which is computational efficient. Inspired by this, we use two linear mapping $U \in \mathbb{R}^{X\times Z}$ and $V \in \mathbb{R}^{Z\times X}$ to formulate the mapping function, $\mathcal{G}(\boldsymbol{x}) = \boldsymbol{x}U$ and $\mathcal{F}(\boldsymbol{z}) = \boldsymbol{z}V$. Since it is linear mapping, we call this method as Linear Mapping (LM). We set $Z = 10$ and it works well both in reconstruction and performance of diffusion model. To satisfy distance-preserving requirements, columns in V should be orthogonal to each other so each dimension in $\boldsymbol{z}$ is independent with each other. Thus, if this mapping is well-trained, variance of $\boldsymbol{z}$ is diagonal matrix. In our implementation, we assume we can learn such a good linear mapping and marginal variances in Corollary 1 are all diagonal matrices. We visualize the Top-10 components of PCA [16] and 10 components in V in Fig. 6. The modes learned from the LM exhibit greater diversity and expressiveness. In addition, latent variances derived from the LM across different dimensions are consistent, whereas PCA-derived

(a) Modes from LM.



(b) Modes from PCA.



(c) Variances of latents from PCA and LM.

Fig. 6: Analysis on PCA and Linear Mapping (Ours).

Table 6: Quantitative results on the Argoverse 2 validation dataset.

Latent	avgMinADE_6^*	avgMinFDE_6^*	avgMinADE_{128}	avgMinFDE_{128}
PCA [16]	0.60	1.33	0.46	0.80
LM (Ours)	0.60	1.32	0.43	0.81

latent variances show extreme variability, with a large variance in the main component and significantly lower variance in the remaining components. We also compare the performance of latent diffusion model with PCA and LM in Tab. 6, using Optimal Gaussian Diffusion. Our linear mapping approach can improve the performance.

C.3 Sample Clustering Algorithm

We design a sample clustering algorithm to to generate a representative set from a batch of joint trajectory samples. Specifically, we first designate $\mathcal{U}$ as the central point for the groups. Then for each vehicle i, we map the trajectory samples $\boldsymbol{x}_{0,i}$, $i = 1, 2, ..., N$ into closest $\boldsymbol{r}_i^l$, $l = 1, 2, ..., L$. Thus, we assign each joint trajectory into $\mathcal{U}$. Subsequently, we sort the $\mathcal{U}$ by the size of its group memberships. Third, we prune the group based on the center of the groups represented by the combination of reference trajectories $\boldsymbol{r}_i^l$. Specifically, if the centers of two groups are close to each other, the group with the larger number of members will absorb the other. We define 'close enough' as a scenario where the maximum endpoint deviation between these two joint trajectories is less than 2.5 meters, aligning with the threshold used in NMS [34]. After pruning, we calculate the probability of each group by its size. Since our clustering algorithm does not need to calculate the exact log probability [16], it is less computationally demanding compared to more intensive techniques such as Non-Maximum Suppression (NMS) [34] or Expectation-Maximization (EM) [42]. However, since we do not explicitly to calculate the likelihood of a sample, the likelihood estimation is not very accurate. This has resulted in our outperforming the rank-3rd benchmark on the Argoverse 2 Multi-world Forecasting leaderboard across all metrics, with

Table 7: Evaluation on different number of samples on the Argoverse 2 Multi-world Forecasting leaderboard.

# of Samples	avgMinFDE$^*_6$	avgMinFDE$^*_1$	actorMR$^*_6$	avgMinADE$^*_6$	avgMinADE$^*_1$	avgBrierMinFDE$^*_6$	actorCR$^*_6$
32	1.34	2.78	0.18	0.61	1.11	2.01	0.01
128	1.32	2.74	0.17	0.60	1.09	1.97	0.01
512	1.31	2.72	0.17	0.60	1.08	1.96	0.01
1024	1.31	2.71	0.17	0.60	1.08	1.95	0.01
2048	1.31	2.71	0.17	0.60	1.08	1.95	0.01

the exception of avgBrierMinFDE$_K$, which is calculated based on the likelihood of the samples. Since we are focusing on optimizing diffusion model, we leave fast and accurate likelihood estimation for the future research. Table 7 demonstrates the influence of the number of samples. With more samples, all metrics improve. The performance saturates when the number of samples exceeds 1024.

C.4 Ablation Study on Marginal Predictors

We froze the trained Optimal Gaussian Diffusion (OGD) model used in Sec. 5.2 and replaced QCNet [50] with Forecast-MAE [2] as the marginal predictor. We downloaded their pre-trained weights from their official website [6] and fine-tuned them for multiple epochs. Fine-tuning is necessary because the pre-trained weights were trained on a subset of target agents for the joint prediction task [44]. We compare the marginal and joint metrics in Tab. 8. The Forecast-MAE version of our model also yielded good performance, showing the flexibility of our approach. Note that the joint metrics of OGD are correlated with the marginal metrics. It indicates that we could improve OGD's performance without re-training if a better marginal predictor is available.

Table 8: Comparison on different marginal predictors.

Predictor	Epoch	Marginal Metrics		Joint Metrics	
		MinADE$_6$	MinFDE$_6$	avgMinADE$_{128}$	avgMinFDE$_{128}$
QCNet	-	**0.37**	**0.60**	**0.43**	**0.81**
Forecast-mae	7	0.37	0.70	0.47	0.85
Forecast-mae	1	0.39	0.74	0.49	0.91

[6] https://github.com/jchengai/forecast-mae

D Guided Sampling Baseline and Step Size Tuning

In this section, we first derive two guided sampling baselines: Next Noisy Mean (NNM) Guidance and Score Function (SF) Guidance. We also introduce the comparison settings for controllable generation tasks. Secondly, we present the step size tuning experiments.

D.1 Guided Sampling Baseline

Previous guided sampling approaches for controllable generation in autonomous driving are mainly focusing on guide the sample generation with an analytical guidance cost function $\mathcal{J}(\cdot)$ defined on clean data $\boldsymbol{x}_0$, such as goal point guidance and target speed guidance [49]. In this paper, we discuss guided sampling approach under the human-defined guidance cost function $\mathcal{J}(\cdot)$ with no need to train additional guidance function defined on noisy data $\boldsymbol{x}_t$, $t > 0$ [15].

The first baseline is to directly calculate $\nabla_{\boldsymbol{x}_t}\mathcal{J}(\boldsymbol{x}_t)$. Specifically, it first calculate the mean of $\boldsymbol{x}_{t-1}$ conditioned on $\boldsymbol{x}_t$,

$$\boldsymbol{m}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)) \tag{30}$$

Then, add the guidance into $\boldsymbol{m}_{t-1}$

$$\tilde{\boldsymbol{m}}_{t-1} = \boldsymbol{m}_{t-1} - \text{clip}(\zeta\nabla_{\boldsymbol{m}_{t-1}}\mathcal{J}(\boldsymbol{m}_{t-1}), \pm\beta_t\sigma_p) \tag{31}$$

where ζ is the step size, clip is elementwise clipping function, σ_p is the positive squared root of diag$[\boldsymbol{\Sigma}_p]$. Note that $\boldsymbol{m}_{t-1}$ is also noisy so $\mathcal{J}(\boldsymbol{m}_{t-1})$ suffers from numerical instability. Since this method directly inject the guidance into the mean of next noisy data distribution $\boldsymbol{x}_{t-1}$, we call it as Next Noisy Mean (NNM) Guidance.

The second baseline is to first project $\boldsymbol{x}_t$ into $\hat{\boldsymbol{x}}_0$ on the clean manifold and then calculate the guidance $\nabla_{\boldsymbol{x}_t}\mathcal{J}(\hat{\boldsymbol{x}}_0)$ [16,28]. Note that [16,28] directly train a denoiser to predict $\hat{\boldsymbol{x}}_0$. In our DDPM [12] formulation, $\hat{\boldsymbol{x}}_0$ is estimated through Tweedie's formula,

$$\hat{\boldsymbol{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\boldsymbol{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)) \tag{32}$$

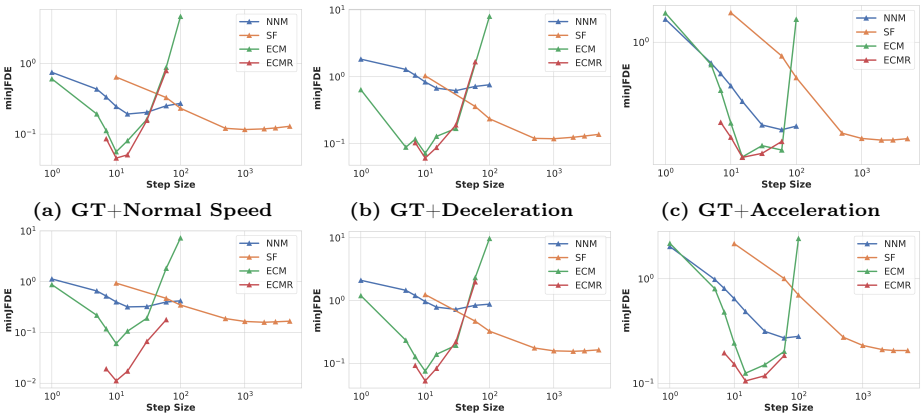
Then, add the guidance $\nabla_{\boldsymbol{x}_t}\mathcal{J}(\hat{\boldsymbol{x}}_0)$ to the score function,

$$\tilde{\boldsymbol{s}}_\theta(\boldsymbol{x}_t, t) = \boldsymbol{s}_\theta(\boldsymbol{x}_t, t) + \text{clip}(\zeta\sqrt{1-\bar{\alpha}_t}\nabla_{\boldsymbol{x}_t}\mathcal{J}(\hat{\boldsymbol{x}}_0), \pm\sigma_p)/\sqrt{1-\bar{\alpha}_t} \tag{33}$$

where ζ is the step size. Since $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) = -\sqrt{1-\bar{\alpha}_t}\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)$, we have

$$\tilde{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t) = \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) - \text{clip}(\zeta\sqrt{1-\bar{\alpha}_t}\nabla_{\boldsymbol{x}_t}\mathcal{J}(\hat{\boldsymbol{x}}_0), \pm\sigma_p) \tag{34}$$

We use DDIM [35] to accelerate the inference for controllable generation. Note that we replace β_t in Eq. (31) with $\sqrt{1-\bar{\alpha}_t/\bar{\alpha}_{t'}}$ for time index t', $t' < t$.



(a) GT+Normal Speed (b) GT+Deceleration (c) GT+Acceleration

(d) U+Normal Speed (e) U+Deceleration (f) U+Acceleration

Fig. 7: Evaluation on the impact of step size. In general, small step size causes low guidance effectiveness. Big step size will cause the optimization process unstable, leading to low guidance effectiveness.

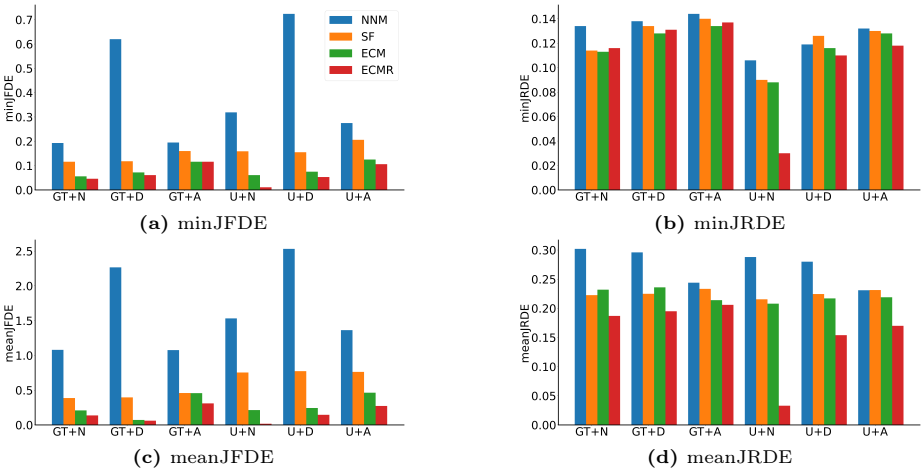
To fairly compare the performance and efficiency of different guided sampling approaches, we set one network inference step followed by one gradient guidance step. For our ECM and ECMR, we set t_k to be the same as the DDIM time step t. Specifically, we set DDIM step stride to 10 and T to 100. Then $t_k = 10(k+1)$, where $k = 0, 1, ..., 9$.

D.2 Step Size Tuning

We conduct a grid search to identify the optimal gradient step size for NNM, SF, ECM, and ECMR. For NNM, ECM and ECMR, we use the settings $\zeta \in \{1, 5, 7, 10, 15, 30, 60, 100\}$. For SF, $\zeta \in \{10, 500, 1000, 2000, 3000, 5000\}$. We find that both excessively small and large gradient step sizes result in low guidance effectiveness, similar to the inefficiency or instability seen in gradient-based optimization algorithms with overly small or large step sizes. The optimal step size lies in between. Based on Fig. 7, we select the optimal step size that yields the lowest minJFDE. We apply these optimal step sizes across all guided sampling methods and controllable generation tasks, and report the quantitative results using these sizes throughout the remainder of the paper.

E Evaluation on Controllable Generation Tasks

Figure 8 showcases the performance of various guided sampling methods across different tasks. ECM and ECMR consistently outperform others in minJFDE across all tasks and in minJRDE in nearly all tasks, indicating their superior guidance effectiveness and realism. Notably, ECMR, in particular, excels in meanJFDE/meanJRDE, suggesting the generated joint trajectories are



(a) minJFDE **(b)** minJRDE **(c)** meanJFDE **(d)** meanJRDE

Fig. 8: Evaluation on different guided sampling methods.

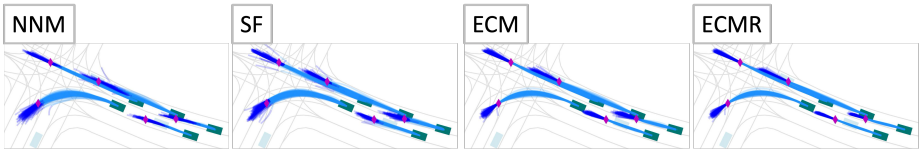


Fig. 9: Visualizations on controllable generation: route set **GT** and **Acceleration**. Magenta diamonds represent goal points. Dodgerblue curves represent the predicted joint trajectory from 0s to 5s. Blue curves represent the predicted joint trajectory from 5s to 6s.

of higher average quality. This implies our methods' ability to produce valid and high-quality joint trajectories with fewer samples. The experiment results are shown in Tab. 9. Our ECM and ECMR approaches can generate a better joint trajectory with a small number of samples.

Figure 9 and Fig. 10 demonstrate examples in different controllable generation tasks. Our guided sampling approach, ECM, can better satisfy the guidance while simultaneously maintaining realism. With reference joint trajectory, ECM can further enhance its performance.

Table 9: Evaluation on different number of samples in the controllable generation task: route set **GT** and **Normal Speed**.

Sampling	# of samples	minJFDE	minJRDE
NNM [49]	128	0.193	0.134
SF [16, 28]	128	0.116	0.114
ECM	128	0.056	0.113
ECM	96	0.068	0.113
ECM	64	0.135	0.118
ECM	32	0.351	0.143
ECMR	128	0.046	0.115
ECMR	96	0.046	0.116
ECMR	64	0.088	0.119
ECMR	32	0.368	0.993

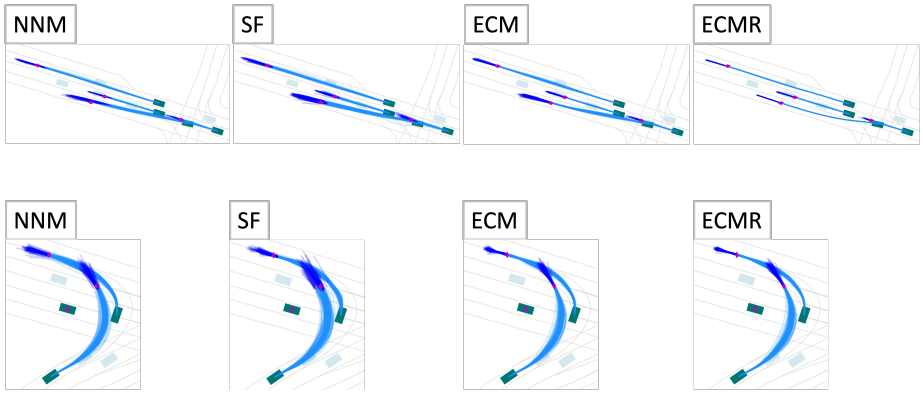


Fig. 10: Visualizations on controllable generation: route set **U** and **Acceleration**. Magenta diamonds represent goal points. Dodgerblue curves represent the predicted joint trajectory from 0s to 5s. Blue curves represent the predicted joint trajectory from 5s to 6s.