Don't be so Negative! Score-based Generative Modeling with Oracle-assisted Guidance

Saeid Naderiparizi [*1 2] Xiaoxuan Liang [*1 2] Setareh Cohan [1] Berend Zwartsenberg [2] Frank Wood [1 2 3]

Abstract

Score-based diffusion models are a powerful class of generative models, widely utilized across diverse domains. Despite significant advancements in large-scale tasks such as text-to-image generation, their application to constrained domains has received considerably less attention. This work addresses model learning in a setting where, in addition to the training dataset, there further exists side-information in the form of an oracle that can label samples as being outside the support of the true data generating distribution. Specifically we develop a new denoising diffusion probabilistic modeling methodology, Gen-neG, that leverages this additional side-information. Gen-neG builds on classifier guidance in diffusion models to guide the generation process towards the positive support region indicated by the oracle. We empirically establish the utility of Gen-neG in applications including collision avoidance in self-driving simulators and safety-guarded human motion generation.

1. Introduction

What should we do when we train a generative model that generates samples known to be invalid within the constraints of the data domain? For instance, when generating traffic scenes, road users cannot overlap each other. Likewise, in robotics, adherence to numerous physics-based constraints is essential for maintaining the appropriate motion and configuration of the robot. Typically, generative models are only trained to maximize the likelihood of a set of "good" training data samples. Nevertheless, when sampling from a fully trained, highly expressive model, some fraction of generated samples fall into the category of "bad" samples. Here we consider the problem of generative modeling where in addition to the conventional training dataset of good samples, we are also given access to constraints in the form of an oracle, which provides insights into whether a given sample is considered bad. Such oracles are ubiquitous in practice and are often a simple function implemented by domain experts.

Modern deep generative models are sufficiently parameterized that they can effectively be trained to result in a model placing a mixture of Dirac measures directly on the training data (Somepalli et al., 2023a;b; Carlini et al., 2023). However, training such models on large amounts of data (Rombach et al., 2022) or imposing regularization (such as smaller architectures or fewer integration steps) ensures that they generalize rather than memorize (Arpit et al., 2017; Zhang et al., 2021). This also results in these models placing mass in the invalid part of the support (Hanneke et al., 2018). In this paper, we assume a modeling regime where the model generalizes effectively. Within this context, our objective is to reduce the probability mass assigned to invalid outputs while avoiding overfitting. Consequently, our contribution can be viewed as a method for controlling the specific type of the model's generalization.

The simplest way to use an oracle is to deploy the model with a rejection sampling loop in which the oracle is used to filter the output and return the constraint-satisfying samples (Kim et al., 2023). Depending on circumstances this may constitute an acceptable final "generative model", but this solution comes at a (potentially unacceptable) computational cost. Consider the concrete example of real-time autonomous vehicle path planning and model predictive control (Zhong et al., 2022). This task involves generating the next control action for an autonomous vehicle based the past and current state of itself and its surroundings. The generated action must avoid collisions and other types of invalid behavior, collectively referred to as "infractions." This is an extremely challenging task that requires low latency and high success rates. To guarantee low latency, it is essential to generate a sufficiently large number of parallel samples to ensure obtaining at least one valid sample with high probability. Assume that a generative model trained

[*]Equal contribution [1]Department of Computer Science, University of British Columbia, Vancouver, Canada [2]Inverted AI, Vancouver, Canada [3]Montréal Institute for Learning Algorithms (MILA). Correspondence to: Saeid Naderiparizi <saeidnp@cs.ubc.ca>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

on trajectories with no infractions produces infracting trajectories for all vehicles with probability ϵ (state of the art models (Lee et al., 2017; Djuric et al., 2018; Gupta et al., 2018; Cui et al., 2019; Ngiam et al., 2021; Ścibior et al., 2021; Niedoba et al., 2023) can have high infraction rates.) Generating at least one non-infracting sample with $1-\delta$ probability without looping the rejection sampler requires $\frac{\log \delta}{\log \epsilon}$ parallel samples. Depending on the specific concrete value of $1-\delta$ required (e.g. 1 chance in a billion of having latency arising from rejection sampling looping imposed) and the baseline trajectory model rejection rate (e.g. 30-50% is not atypical) this could require running many parallel samplers (in this concrete example around 30). Depending on model size and available realtime edge computational capacity, this quantity may be prohibitively large. Other examples of this nature arise in many control as inference problems (Levine, 2018).

Minimizing ϵ directly i.e., restricting the generative model to only place mass on the positive support region indicated by the oracle, is the most natural approach to combat this problem. Working towards this goal includes a body of work on amortized rejection sampling (Warrington et al., 2020; Naderiparizi et al., 2022) and the body of related work on generative adversarial networks (GANs) (Goodfellow et al., 2014). Of course in the GAN setting, the discriminator (which can be used in a rejection sampling loop for improved performance (Azadi et al., 2018; Che et al., 2020)) is learned rather than being a fixed, pre-defined oracle as in the case we consider.

We focus specifically on learning with constraints in score-based models. What we reveal in this study is a *necessary condition*, essential for the accurate functioning of classifier guidance in this problem domain, which, to the best of our knowledge, has been surprisingly overlooked until now. Following recent findings on discriminator guidance in diffusion processes (Kim et al., 2022), we introduce a new methodology for classifier guidance. Our approach involves training and employing a series of differentiable classifiers, trained on synthetic samples generated from a sequence of classifier-guided diffusion models and labeled by the oracle. The resulting sequence of multiply classifier-guided diffusion models effectively reduce the rejection rate while empirically maintaining a competitive probability mass assigned to validation samples. We demonstrate that comparable performance can be achieved with a reduced computational overhead by distilling the sequence of classifiers.

We evaluate our proposed methodology, which we call **Gen**erative modeling with **neG**ative examples (Gen-neG) on several problems, including modeling motion capture sequence data in a way that eliminates ground plane violations, and static traffic scene vehicle arrangements that avoid collisions and off-road placements.

2. Background

2.1. Score-based Diffusion Models

Score-based diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2021), also referred to as diffusion models (DMs) are a class of generative models that are defined through a stochastic process which gradually adds noise to samples from a data distribution $q_0(\mathbf{x}_0)$, such that when simulated forward from $t=0$ the marginal distribution at time T is $q_T(\mathbf{x}_T) \approx \pi(\mathbf{x}_T)$ for some known $\pi(\mathbf{x}_T)$ typically equal to $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. This is known as the "forward process" and is formulated as an SDE

$$d\mathbf{x}_t = f(\mathbf{x}_t, t)dt + g(t)d\mathbf{w}, \quad \mathbf{x}_0 \sim q_0(\mathbf{x}_0), \tag{1}$$

where f and g are predefined drift and diffusion coefficients of $\mathbf{x}_t$ and $\mathbf{w}$ is the standard Wiener process. DMs define another stochastic process known as the "reverse process" defined as

$$d\mathbf{x}_t = [f(\mathbf{x}_t, t) - g(t)^2 s_\theta(\mathbf{x}_t; t)]dt + g(t)d\bar{\mathbf{w}}, \ \mathbf{x}_T \sim \pi(\mathbf{x}_T), \tag{2}$$

where $\bar{\mathbf{w}}$ is the infinitesimal reverse time and reverse Wiener process, respectively. If s_θ matches the score function of the marginals of the forward process, the terminal distribution of the reverse process coincides with $q_0(\mathbf{x}_0)$ (Anderson, 1982). Formally,

$$s_\theta(\mathbf{x}_t; t) = \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t) \Rightarrow p_\theta(\mathbf{x}_0; 0) = q_0(\mathbf{x}_0), \tag{3}$$

where $p_\theta(\mathbf{x}_t; t)$ is the marginal distribution of the reverse process.

In order to approximate the score function $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$, DMs minimize the following score matching objective function (Hyvärinen & Dayan, 2005; Vincent, 2011; Song & Ermon, 2019):

$$\mathcal{L}_\theta^{\text{DM}} = \mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_t} \left[\gamma_t \left\| s_\theta(\mathbf{x}_t; t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) \right\|^2 \right], \tag{4}$$

where $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$, t is sampled from a distribution over $[0, T]$, and γ_t is a positive weighting term. Importantly, the Wiener process in Equation (1) allows direct sampling from the marginals of the forward distributions (Song et al., 2021), i.e. $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\alpha_t \mathbf{x}_0, \sigma_t)$, with α_t and σ_t determined by the drift and diffusion coefficients in Equation (1). This formulation moreover allows the evaluation of the conditional score function $(\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|\mathbf{x}_0))$ in closed form.

Many of the DMs reported in the literature operate on discrete time steps (Ho et al., 2020; Nichol & Dhariwal, 2021),

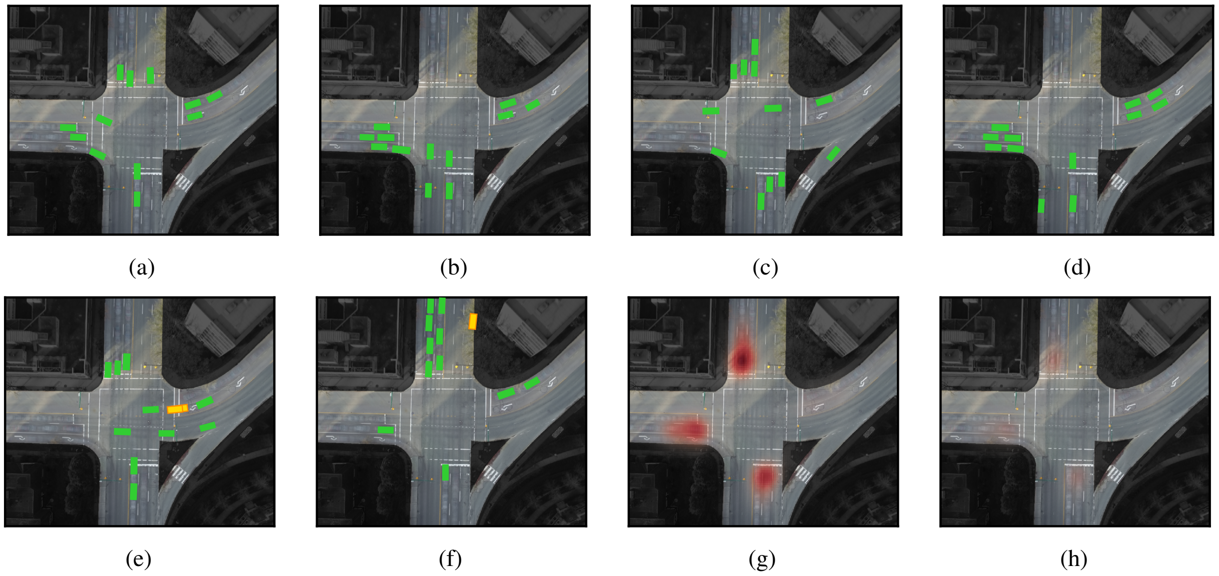


Figure 1. Gen-neG applied to a diffusion model of non-infracting static vehicle placements (i.e. a set of oriented rectangles) for the efficient initialization of autonomous vehicle planning simulators (see Zwartsenberg et al. (2023) for a similar model and full problem description). The top row show samples (green "cars") that are not colliding (non-overlapping) and not off-road (stay within the unshaded area of road surfaces) from a baseline diffusion model improved by Gen-neG. The second row shows the kind of infractions our oracle identifies as not being in the support of the true distribution. (e) shows a collision (yellow overlapping cars) and (f) shows an off-road car in yellow. (g) and (h) graphically illustrate the reduction in infractions per unit area before and after Gen-neG is applied to the baseline model (both plots are normalized to the same maximum value). Quantitative results corresponding to this plot appear later in Table 1.

and can be considered as particular discretizations of the presented framework.

In the remainder of this paper we use q to denote the forward process, s_θ for the score function of the reverse process and p_θ as the distribution generated by running Equation (2) backward in time. This applies to the marginals, conditionals, and posteriors as well. Furthermore, to reduce notational clutter throughout the rest of the paper, we omit the explicit mention of θ and ϕ and t when their meaning is evident from the context.

2.2. Classifier Guidance

A distinctive and remarkable property of DMs is the ability to utilize an unconditional diffusion model to draw samples from its class-conditional distributions at inference time without requiring re-training or fine-tuning (Dhariwal & Nichol, 2021; Song et al., 2021). However, doing so typically utilizes a time-dependent classifier $q(y|\mathbf{x}_t) = \int q(y|\mathbf{x}_0)q(\mathbf{x}_0|\mathbf{x}_t)\, d\mathbf{x}_0$ (alternative approaches include (Wu et al., 2023)). Here, $q(y|\mathbf{x}_0)$ is a traditional classifier, that predicts the class probabilities for each y given a datum $\mathbf{x}_0$ from the dataset. While $q(y|\mathbf{x}_t)$ classifies a noisy datum $\mathbf{x}_t$ sampled from $q_t(\mathbf{x}_t) = \int q(\mathbf{x}_t|\mathbf{x}_0)q(\mathbf{x}_0)\, d\mathbf{x}_0$.

Classifier guidance follows from the identity $\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t)$. Since the score function of the DM $s_\theta(\mathbf{x}_t; t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t)$, we have

$$s_\theta(\mathbf{x}_t|y; t) = s_\theta(\mathbf{x}_t; t) + \nabla_{\mathbf{x}_t} \log q(y|\mathbf{x}_t). \tag{5}$$

Binary classification A special case of the above classifier guidance that we use in this paper is when there are only two classes. We provide here a brief overview of such a binary classification task and the notation associated with it. Let $q(\mathbf{x}|y = 1)$ and $q(\mathbf{x}|y = 0)$ be the distribution of positive and negative examples, respectively. Let $\alpha = q(y = 1)$ and $1 - \alpha = q(y = 0)$ be the prior probabilities $q(y)$ of positive and negative examples. We then have $q(\mathbf{x}) = \alpha q(\mathbf{x}_t|y = 1) + (1 - \alpha)q(\mathbf{x}_t|y = 0)$. A binary classifier $C_\phi : \mathcal{X}, [0, T] \to [0, 1]$, can then be trained to approximate $q(y = 1|\mathbf{x}_t)$ by minimizing the expected cross-entropy loss

$$\begin{aligned}\mathcal{L}_\phi^{\text{CE}} = -\mathbb{E}_{t,\mathbf{x}_t}\Big[&q(y = 1|\mathbf{x}_t) \log C_\phi(\mathbf{x}_t; t)+ \\ &q(y = 0|\mathbf{x}_t) \log(1 - C_\phi(\mathbf{x}_t; t))\Big],\end{aligned} \tag{6}$$

where $\mathbf{x}_t \sim q(\mathbf{x}_t)$. Minimizing the cross-entropy loss between the classifier output and the true label is equivalent to minimizing the KL divergence between the classifier output and the Bayes optimal classifier (Sugiyama et al., 2012, Chapter 4). Therefore, Equation (6) is minimized when

$$\begin{aligned}C_{\phi^*}(\mathbf{x}_t; t) &= q(y = 1|\mathbf{x}_t) \\ &= \frac{\alpha q(\mathbf{x}_t|y = 1)}{\alpha q(\mathbf{x}_t|y = 1) + (1 - \alpha)q(\mathbf{x}_t|y = 0)}.\end{aligned} \tag{7}$$

Hence, $s_\theta(\mathbf{x}_t|y=1;t) = s_\theta(\mathbf{x}_t;t) + \nabla_{\mathbf{x}_t} \log C_{\phi^*}(\mathbf{x}_t;t)$. Note that this minimizer critically depends on the class prior probabilities α and $1-\alpha$. Gen-neG works by ensuring that these are properly accounted for.

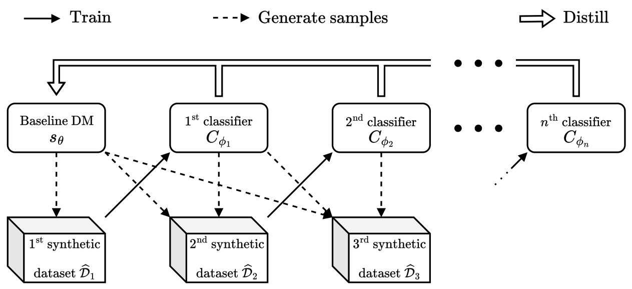
3. Methodology

In this section, we describe **Gen**erative modeling with **neG**ative examples (Gen-neG), a method for guiding the sampling of diffusion models to satisfy the constraints imposed by an oracle function. Gen-neG has two stages. It starts by training a DM on available training data following standard DM training procedures (e.g. Section 2.1) without utilizing the oracle. We refer to this model as the "baseline DM" throughout. In the second stage of Gen-neG, we draw samples from this baseline DM, label them using the oracle, and train a binary classifier using those samples, which we later use for guidance. Next, we use the obtained classifier to guide our baseline DM, and the combination of both constitutes a new generative model. Gen-neG establishes this combined model as a new DM and then repeats the process of sampling, classifying using the oracle, and training a time dependent classifier to form yet another model. Refinement using this iterative process can be repeated until the desired performance is obtained. We refer to this type of refinement as "stacking". Optionally, if better computational performance is desired, the stacked model can be distilled into a new model at any desired stage. As mentioned before, an important feature of Gen-neG is to properly account for the prior class probabilities α and $1-\alpha$ in the training of all classifiers, which is formalized later in this section and demonstrated later in Section 4. An overview of Gen-neGis shown in Figure 2 and Algorithm 1.

Problem formulation and notation Let $\mathcal{D} = \{\mathbf{x}^i\}_{i=1}^N \sim q(\mathbf{x})$ be a dataset of i.i.d. samples from an unknown data distribution q. Furthermore, let $\mathcal{O} : \mathcal{X} \to \{0,1\}$ be an oracle function that assigns each point in the data space $\mathcal{X}$ a binary label. In other words, this oracle partitions the data space into two disjoint sets $\mathcal{X} = \Omega \cup \Omega^{\complement}$ such that $\mathcal{O}(\mathbf{x}) = \mathbf{1}_\Omega(\mathbf{x})$. Moreover, assume $\mathcal{D} \subseteq \Omega$ i.e., all training examples satisfy the oracle constraints. Our objective is to learn a score-based diffusion model that (i) maximizes the likelihood of $\mathcal{D}$ and (ii) avoids allocating probability to $\Omega^{\complement}$.

3.1. Bayes Optimal Classifier Guidance for Diffusion models

The core component of Gen-neG is in the second stage where a *Bayes optimal* diffusion-time dependent classifier that discriminates between positive and negative samples in Ω and $\Omega^{\complement}$ respectively, is used to guide the baseline DM. There are two main questions that Gen-neG answers. (i) Which data and class distribution should the classifier be Bayes optimal with respect to such that classifier guidance does not modify the sampling distribution in the oracle approved region? (ii) How can one train such a classifier?



Train
Generate samples
Distill
Baseline DM s_θ
1st classifier C_{ϕ_1}
2nd classifier C_{ϕ_2}
nth classifier C_{ϕ_n}
1st synthetic dataset $\widehat{\mathcal{D}}_1$
2nd synthetic dataset $\widehat{\mathcal{D}}_2$
3rd synthetic dataset $\widehat{\mathcal{D}}_3$

Figure 2. Overview of Gen-neG. The process begins with a baseline diffusion model. In each iteration, a synthetic training dataset is generated from the current model and labeled by the oracle function $\mathcal{O}$. A time-dependent classifier is trained on this dataset and then used to update the model by guiding it towards the positive support region (see Equation (10)). The guided model (with multiple classifiers) can be distilled into a new, improved baseline diffusion model at any iteration by minimizing Equation (12).

Classifier guidance in score-based models is typically used to generate samples from a specific pre-defined class on the training dataset. As such, the classifier and generative model share the same training distribution. In our case, however, there is no pre-defined dataset of positive and negative classes. Even the training dataset $\mathcal{D}$ only includes samples from one class, since they all satisfy the oracle constraints. Despite that, as we show later in Section 4, classifier guidance using a Bayes optimal classifier for the binary classification task on the fully-synthetic data generated by the same baseline DM leads to (i) zero infraction and (ii) improved likelihood estimation on the data distribution, including a *held-out test set*.

The goal of Gen-neG is therefore to solve this classification task on synthetic data. Formally, the data is distributed as $p_\theta(\mathbf{x}_0)$, the labels are $y = \mathcal{O}(\mathbf{x}_0)$, and the noise distribution is $p_\theta(\mathbf{x}_t|\mathbf{x}_0)$. One can obtain the Bayes optimal classifier for this task using a cross-entropy objective similar to Equation (6) in which q is replaced by p_θ. This objective can be equivalently written as

$$\mathcal{L}^{\mathrm{CE}}_{\phi,\theta} = -\mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_t}\Big[\mathcal{O}(\mathbf{x}_0)\log C_\phi(\mathbf{x}_t;t) + (1-\mathcal{O}(\mathbf{x}_0))\log(1-C_\phi(\mathbf{x}_t;t))\Big], \tag{8}$$

where $(\mathbf{x}_0,\mathbf{x}_t) \sim p_\theta(\mathbf{x}_0,\mathbf{x}_t)$ (see the Appendix for details). In order to avoid the computational cost of sampling from the baseline DM to compute this objective, Gen-neG approximates $p_\theta(\mathbf{x}_0,\mathbf{x}_t) \approx p_\theta(\mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)$. This is similar to the approximation in Kim et al. (2022). The objective function of Gen-neG is therefore

$$\mathcal{L}_\phi^{\text{cls}} = -\mathbb{E}_{t,p_\theta(\mathbf{x}_0)}\Big[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\Big[\mathcal{O}(\mathbf{x}_0)\log C_\phi(\mathbf{x}_t;t) + (1-\mathcal{O}(\mathbf{x}_0))\log(1-C_\phi(\mathbf{x}_t;t))\Big]\Big]. \tag{9}$$

For notational simplicity, we drop the dependence of $\mathcal{L}_\phi^{\text{cls}}$ on θ, including in the equation above. Once trained, the classifier is incorporated into the baseline DM by

$$s_{\theta,\phi}(\mathbf{x}_t;t) = s_\theta(\mathbf{x}_t;t) + \nabla_{\mathbf{x}_t}\log C_\phi(\mathbf{x}_t;t). \tag{10}$$

We denote the marginal distributions generated by the oracle-assisted DM, implicitly defined through Equation (10) as $p_{\theta,\phi}(\mathbf{x}_t;t)$.

Training the classifier Training the classifier in our approach presents a noteworthy challenge due to the major label imbalance within the synthetic dataset $\widehat{\mathcal{D}}$ generated by the model. This imbalance emerges when the baseline is already close to the target distribution, resulting in a scarcity of negative examples. Meanwhile, these negative examples play a crucial role in guiding the model at the boundary between positive and negative examples, where the model requires the most guidance.

Gen-neG addresses this challenge by sampling a balanced dataset $\widehat{\mathcal{D}}$ from the model, ensuring the same number of positive and negative examples. However, this changes the class prior probabilities from the true marginal distribution over labels α and $1-\alpha$ which in turn changes the optimal classifier the cross-entropy objective targets (see Equation (7)). We show evidence of this happening in Figure 3. Gen-neG crucially employs importance sampling in the classifier's training objective to rectify the bias introduced by having to balance the dataset to achieve high classifier accuracy in training.

Given a balanced dataset $\widehat{\mathcal{D}} = \widehat{\mathcal{D}}^+ \cup \widehat{\mathcal{D}}^-$ where $\widehat{\mathcal{D}}^+ \sim p(\mathbf{x}_0|y=1)$, $\widehat{\mathcal{D}}^- \sim p(\mathbf{x}_0|y=0)$, $N = |\widehat{\mathcal{D}}^+| = |\widehat{\mathcal{D}}^-|$, and $\alpha = p_\theta(y=1)$,

$$\begin{aligned}&\hat{\mathcal{L}}_\phi^{\text{cls}}(\alpha,\widehat{\mathcal{D}}^+,\widehat{\mathcal{D}}^-)\\ :=&\frac{1}{N}\sum_{\mathbf{x}_0\in\widehat{\mathcal{D}}^+}\alpha\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log C_\phi(\mathbf{x}_t;t)\right]\\ +&\frac{1}{N}\sum_{\mathbf{x}_0\in\widehat{\mathcal{D}}^-}(1-\alpha)\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log(1-C_\phi(\mathbf{x}_t;t))\right],\end{aligned} \tag{11}$$

is an importance sampling estimator of the objective function in Equation (9); proof in Appendix A.5.

3.2. Iterative Training by Stacking Classifiers

With an optimal classifier minimizing Equation (8), the DM with score function $s_{\theta,\phi}$ will have improved likelihood and

Algorithm 1 Gen-neG

```
1:  Input: dataset $\mathcal{D}$, oracle $\mathcal{O}$, synthetic dataset size $N$
2:  $i \leftarrow 0$
3:  $\theta_i \leftarrow \arg\min_\theta \mathcal{L}_\theta^{\text{DM}}$ {train baseline DM, Eq. (4)}
4:  $s_i \leftarrow s_{\theta_i}(\mathbf{x}_t;t)$
5:  repeat
6:      $\widehat{\mathcal{D}}_i^+, \widehat{\mathcal{D}}_i^- \leftarrow \varnothing$
7:      repeat
8:          $\widehat{\mathcal{D}}^+, \widehat{\mathcal{D}}^- \leftarrow$ generate more samples from DM with
            score function $s_i$ and label with $\mathcal{O}$
9:          $\widehat{\mathcal{D}}_i^+ \leftarrow \widehat{\mathcal{D}}_i^+ \cup \widehat{\mathcal{D}}^+$, $\widehat{\mathcal{D}}_i^- \leftarrow \widehat{\mathcal{D}}_i^- \cup \widehat{\mathcal{D}}^-$
10:     until $\min(|\widehat{\mathcal{D}}_i^+|, |\widehat{\mathcal{D}}_i^-|) < N$
11:     $\alpha_i \leftarrow |\widehat{\mathcal{D}}_i^+|/(|\widehat{\mathcal{D}}_i^+| + |\widehat{\mathcal{D}}_i^-|)$ {Estimate class prior
        probabilities for Bayes optimal classifier}
12:     $\widehat{\mathcal{D}}_i^+ \leftarrow \text{select}(N, \widehat{\mathcal{D}}_i^+)$, $\widehat{\mathcal{D}}_i^- \leftarrow \text{select}(N, \widehat{\mathcal{D}}_i^-)$ {bal-
        ance dataset for IS classifier training}
13:     $\phi_i \leftarrow \arg\min_\phi \hat{\mathcal{L}}_\phi^{\text{cls}}(\alpha_i, \widehat{\mathcal{D}}_i^+, \widehat{\mathcal{D}}_i^-)$ {train guidance
        classifier, Eq. (11)}
14:     $i \leftarrow i+1$
15:     if distill then
16:         $\psi \leftarrow \arg\min_\psi \mathcal{L}_\psi^{\text{dtl}}$ {See Eq. (12)}
17:         $s_i \leftarrow s_\psi(\mathbf{x}_t;t)$
18:     else
19:         $s_i \leftarrow s_{i-1} + \nabla_{\mathbf{x}_t}\log C_{\phi_i}(\mathbf{x}_t;t)$ {See Eq. (10)}
20:     end if
21: until done
22: Output: DM score function $s_i$
```

zero infraction (see Corollary 3.2). However, in practice the trained classifier is only an estimate because learning the true decision boundary would require an infeasible synthetic dataset size and optimization budget, thus infractions may not be entirely eliminated.

To alleviate this problem, we note that once the classifier is trained, the guided score function $s_{\theta,\phi}(\mathbf{x})$ itself defines a new diffusion model. Consequently, we can employ a similar procedure to train a new classifier on $s_{\theta,\phi}$, aiming to further lower its infraction rate. This iterative approach involves training successive classifiers and incorporating them into the model, progressively enhancing its performance and reducing the infraction rate.

3.3. Model Distillation

Adding a stack of classifiers to the model linearly increases its computational cost, since each new classifier requires a forward and backward pass each time the score function is evaluated. To avoid this, we show that it is possible and sometimes beneficial to distill the classifiers into a combined diffusion model.

Let $s_{\theta,\mathbf{\Phi}}$ be a "teacher model" consisting of a baseline model s_θ and a stack of classifiers $\{C_\phi\}_{\phi\in\mathbf{\Phi}}$. We distill $s_{\theta,\mathbf{\Phi}}$ into

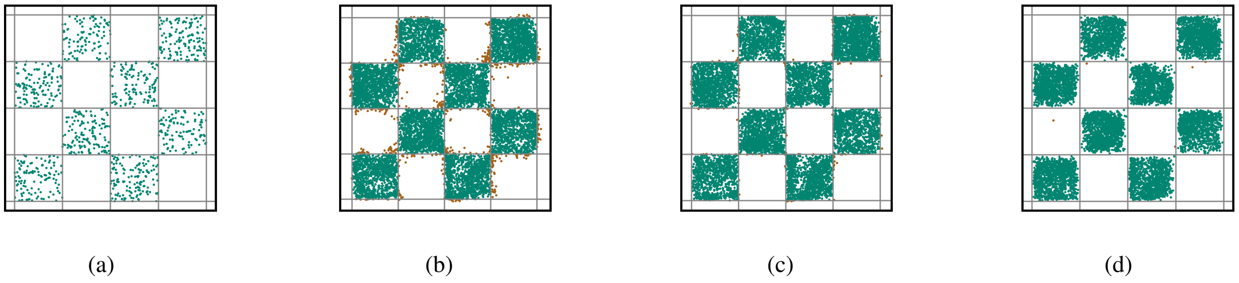


Figure 3. Samples from the checkerboard experiment. Samples with infraction (i.e. $\mathcal{O}(\mathbf{x}) = 0$) are shown in brown. (a) The baseline training dataset; (b) baseline DM; (c) first iteration of Gen-neG using a Bayes optimal classifier trained on a balanced dataset and correct α; (d) a classifier trained on a balanced dataset without employing importance sampling results in suboptimal density estimation. We see samples are suboptimally pushed inwards from the boundaries. We also have observed that validation ELBOs in these kinds of cases are significantly worse.

a new "student model" s_ψ^{dtl}, possibly with the same architecture as the baseline model, by minimizing the following distillation loss

$$\mathcal{L}_\psi^{\text{dtl}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), t} \left[\gamma_t \left\| s_{\theta, \mathbf{\Phi}}(\mathbf{x}_t; t) - s_\psi^{\text{dtl}}(\mathbf{x}_t; t) \right\|^2 \right], \tag{12}$$

where γ_t is the weight term, similar to the training objective of diffusion models. Here, $\mathcal{L}^{\text{dtl}}$ makes the outputs of the student model match that of the teacher. Algorithm 1 summarises Gen-neG.

3.4. Theory

Here we provide the theoretical grounding for why the classifier Gen-neG targets results in improved likelihood estimation and avoids violating the constraints.

Theorem 3.1. *Let $p_\theta(\mathbf{x})$ be the distribution learned by a baseline DM with marginal distributions denoted by $p_\theta(\mathbf{x}_t; t)$ and let $p_\theta(y = 1|\mathbf{x}_0) = \mathcal{O}(\mathbf{x}_0)$. Further, let $C_{\phi^*} : \mathcal{X}, [0, T] \to [0, 1]$ be the Bayes-optimal time-dependent binary classifier arising from perfectly optimizing the following cross-entropy objective*

$$\begin{aligned} \mathcal{L}_{\phi,\theta}^{CE} = -\mathbb{E}_t \Big[\mathbb{E}_{p_\theta(\mathbf{x}_0, \mathbf{x}_t)} \big[\mathcal{O}(\mathbf{x}_0) \log C_\phi(\mathbf{x}_t; t) + \\ (1 - \mathcal{O}(\mathbf{x}_0)) \log(1 - C_\phi(\mathbf{x}_t; t)) \big] \Big] \end{aligned} \tag{13}$$

then

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t | y = 1; t) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t; t) + \\ \nabla_{\mathbf{x}_t} \log C_{\phi^*}(\mathbf{x}_t; t). \end{aligned} \tag{14}$$

In other words, by using a Bayes-optimal binary classifier for guidance, we target exactly the score function of positive (oracle-approved) examples, without modifying the underlying distribution in the oracle-approved region.

Corollary 3.2. *For an optimal classifier C_{ϕ^*},*

1. *$p_{\theta,\phi^*}(\mathbf{x}_t) = p_\theta(\mathbf{x}_t | y = 1)$,*
2. *There is no mass on $\Omega^\complement$, i.e. $\int_{\mathbf{x} \in \Omega^\complement} p_{\theta,\phi^*}(\mathbf{x}_t) = 0$,*
3. *For any dataset $\mathcal{D} \subseteq \Omega$, $p_{\theta,\phi^*}(\mathcal{D}) \geq p_\theta(\mathcal{D})$.*

Corollary 3.2 suggests that our Gen-neG methodology can improve the baseline DM in terms of both infraction rate and test dataset likelihood.

See the proofs for the Theorem 3.1 and Corollary 3.2 in Appendices A.1 and A.2.

4. Experiments

We evaluate Gen-neG on three datasets: a 2D checkerboard, collision avoidance in traffic scenario generation, and safety-guarded human motion generation. In each experiment we report a likelihood-based metric on a held out dataset to measure distributional shifts and a form of infraction metric which measures faithfulness to the oracle. We release the source code for the checkerboard and motion generation experiments[1].

4.1. Checkerboard

To develop some insight, we start by demonstrating the principles and performance of Gen-neG on a dataset of 2-dimensional points uniformly distributed on a checkerboard grid as shown in Figure 3(a). We apply EDM (Karras et al., 2022), a continuous-time DM, to this problem. The training dataset only contains 1000 points. This makes the model prone to over-fitting. As shown in Figure 4 and further explored in Appendix D.5(blue dots), training the model for long causes strong overfitting. We therefore stop training of the baseline DM before it starts overfitting measured by the evidence lower bound (ELBO) on a held-out validation set. Figure 3(b) shows samples from this baseline DM and the blue dot with a red edge in Figure 4 shows its measured performance.

Figure 3(c) shows samples from the first iteration of Gen-neG and the orange stars in Figure 4 show the metrics for the first five iterations of Gen-neG. These results show Gen-

[1]https://github.com/plai-group/gen-neg

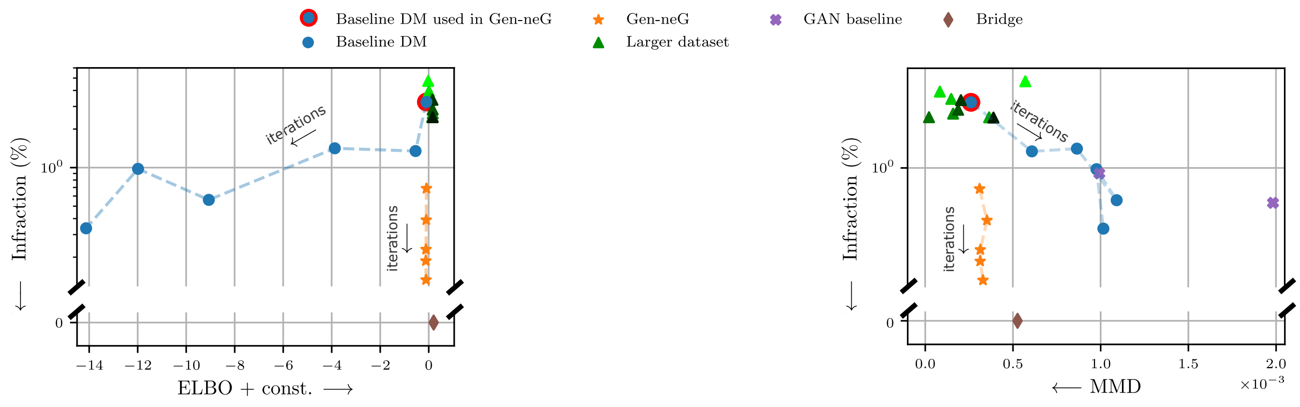


Figure 4. Infraction, ELBO and MMD estimates from the checkerboard experiment. Dashed lines connect different iterations of the same method. For the baseline DM, it corresponds to training iterations. For Gen-neG it corresponds to different iterations of our algorithm. The plot shows (i) prolonged training of the *baseline DM* reduces the infraction rate but leads to overfitting. (ii) Our experiments on *larger datasets*, denoted by triangles in varying shades of green (up to 1000× larger than the original dataset; darker shades indicate larger datasets), maximum likelihood training even on substantially larger datasets is strongly outperformed by Gen-neG. (iii) Various iterations of Gen-neG consistently decrease the infraction rate while maintaining fidelity to the data distribution. (iv) Diffusion bridges perfectly achieve zero infraction rate and improved likelihood estimation. However, they require analytical access to constraints and are not generalizable to complex constraints.

neG dramatically reduces the rate of infractions while still matching the data distribution for non-infracting regions. Figure 7 in the appendix shows distilling various Gen-neG models maintains a comparable performance.

We test the effectiveness Gen-neG against training on a larger dataset. The green triangles in Figure 4 show the performance of models trained on datasets up to 10^6 points (1000× larger than our original dataset). Even the first iteration of Gen-neG achieves significantly lower infraction rates compared to any of these models. This also emphasizes the importance of negative samples, consistent with Giannone et al. (2023).

An alternative approach to learning constrained distributions with diffusion models is diffusion bridges (Liu et al., 2023b) that provably produce no infraction. However, it requires analytical access to the constraints and quantities that are only tractable under very simple constraints. As such it is not applicable to our problem setting. Therefore, it should be treated as an upper bound to Gen-neG's performance. As shown by the brown diamond in Figure 4, this method achieves zero infractions with a better ELBO.

Kong & Chaudhuri (2023) proposed an algorithm for data redaction in GANs that is applicable to our oracle-based constraints. Since GANs do not provide ELBO estimates, we only compute MMD and infraction rates for this baseline. We train this model on our problem and choose the two checkpoints with the best value for either metric. The purple crosses in the right panel of Figure 4 show our results. The MMD scores are significantly worse than those of diffusion-based models, including Gen-neG. Additionally, Gen-neG quickly outperforms this baseline in infraction rate.

The Gen-neG models in this experiment reach near-zero infraction rates. This makes the balanced synthetic dataset generation step slow. We explore an importance sampling-based approach to avoid this slowdown. Our approach and its results are presented in Figure 9 but we leave further exploration for future research.

4.2. Traffic Scene Generation

We continue to the task of traffic scene generation where vehicles of varying sizes are placed on a given two-dimensional map. Traditionally implemented by rule based systems (Yang & Koutsopoulos, 1996; Lopez et al., 2018), this task has recently been approached using generative modeling techniques (Tan et al., 2021; Zwartsenberg et al., 2023). In both of these prior works, the common approach has been to discard any samples that violate predefined rules, such as a vehicle being outside the designated driving area ("offroad") or overlapping with another vehicle ("collision"). Rejecting such samples, while effective, can be computationally wasteful, particularly when rule violations occur frequently. Hence, in this context, we employ Gen-neG to enhance performance. The specific task we consider is to generate 12 vehicles in a given scene, conditioned on a rendered representation of the roadway. Each vehicle is represented by its position, length, width, orientation and velocity for a total of 7 dimensions per vehicle. Vehicles are sampled *jointly*, meaning that the features are in $\mathbb{R}^{N\times 7}$. We train the baseline DM employing the formalism in DDPM (Ho et al., 2020) with a transformer-based denoising network (Vaswani et al., 2017) on a private dataset. Our architecture consists

Table 1. Results for traffic scene generation, in terms of collision, offroad, and overall infractions as well as reweighted ELBO (r-ELBO). We compare Gen-neG against a normalizing flow baseline (Zwartsenberg et al., 2023), a classifier trained on imbalanced synthetic dataset, and a classifier without importance sampling and a time-independent classifier. The final two rows provide the results of distilling the models labeled with † and *.

Method	Collision (%) $\downarrow$	Offroad (%) $\downarrow$	Infraction (%) $\downarrow$	r-ELBO ($\times 10^{-2}$) $\uparrow$
baseline DM	28.3 ± 0.70	1.3 ± 0.14	29.3 ± 0.64	-27.5 ± 0.01
Normalizing flow (Zwartsenberg et al., 2023)	91.2 ± 0.27	13.1 ± 0.48	91.9 ± 0.25	—
Time-independent classifier	20.7 ± 0.59	0.9 ± 0.09	21.4 ± 0.63	-244 ± 30.4
Imbalanced classifier (ablation)	17.8 ± 1.21	0.9 ± 0.16	18.6 ± 1.30	-27.7 ± 0.01
w/o IS classifier (ablation)	14.6 ± 0.49	0.8 ± 0.13	15.2 ± 0.50	-28.0 ± 0.01
Gen-neG† (iteration 1)	16.4 ± 0.5	0.9 ± 0.12	17.2 ± 0.44	-27.7 ± 0.01
Gen-neG* (iteration 2)	11.6 ± 0.65	0.6 ± 0.10	12.2 ± 0.60	-28.0 ± 0.01
Gen-neG (distillation of †)	12.2 ± 0.42	0.8 ± 0.06	12.9 ± 0.36	$\mathbf{-26.8 \pm 0.01}$
Gen-neG (distillation of *)	$\mathbf{5.1 \pm 0.24}$	$\mathbf{0.5 \pm 0.09}$	$\mathbf{5.6 \pm 0.20}$	-27.0 ± 0.01

of self-attention layers and map-conditional cross-attention layers in an alternating order. We use relative positional encodings (RPEs) (Shaw et al., 2018; Wu et al., 2021). Further details are provided in Appendix B.2. Relevant examples (including infracting, and non-infracting ones) and road geometry can be seen in Figure 1.

Table 1 summarizes the results of this experiment. We compare against the baseline DM model and various guided models. Further, we compare against a prior work on this problem based on normalizing flows (NFs) (Zwartsenberg et al., 2023). Comparing to the baseline DM, Gen-neG lowers the infraction rates while maintaining a comparable distribution match. Gen-neG significantly outperforms the NF baseline because DMs are much more expressive generative models. The infraction rates reported here for the NF baseline are worse than those of Zwartsenberg et al. (2023). This can be attributed to the higher average traffic density in our dataset compared to the INTERACTION dataset (Zhan et al., 2019) which Zwartsenberg et al. (2023) uses. In Appendix D.3 we empirically verify this by training the baseline DM on the INTERACTION dataset.

In the second section of Table 1 we report results of various guided diffusion models using a synthetic dataset generated by the baseline DM and labelled by the oracle. First, we consider a common approach of classifier guidance in which a time-independent classifier pre-trained *on clean data* is utilized to perform (approximate) classifier guidance (Wu et al., 2023; Bansal et al., 2023). In this approach, one-step estimate of $\mathbf{x}_0 \approx \frac{\mathbf{x}_t + \sigma_t^2 s_\theta(\mathbf{x}_t; t)}{\alpha_t}$ is obtained using the diffusion model. This estimate is then passed to the pre-trained classifier. This method enhances the infraction rate but exhibits a significant decrease in ELBO, indicating a strong distributional misalignment. We also present ablations where we omit the importance sampling ("w/o IS") step or forego balancing the dataset ("imbalanced classifier") in Gen-neG. The "w/o IS" ablation improves infraction rates, but they both deteriorate the ELBO.

Different iterations of Gen-neG, however, shows even better infraction rates. The presence of lower ELBO can be justified by the approximations in Gen-neG's objective function and classifiers not being trained to optimality. This is why the results deviate from theory to some extent. On the other hand, training the baseline DM is the only stage where we explicitly maximize the ELBO. Classifiers trained on all the other iterations only implicitly improve ELBO through guiding the model to not allocate probability mass on the invalid region. Finally, in the third section of Table 1 we demonstrate that our approach of distilling the resulting models back into a single one works well here too, sometimes even surpassing their teacher models. This can be attributed to knowledge distillation effects (Hinton et al., 2015). Overall we find that Gen-neG works as expected, and provides a competitive infraction rate boost over our baseline model, without shifting the distribution. To relate this to the introduction, as explained in Appendix D.1, using Gen-neG in producing non-infracting scenes for autonomous vehicle synthetic data generation would reduce GPU costs by 57% on average.

4.3. Motion Diffusion

Our final experiment focuses on human motion generation. Diffusion models have been successfully applied to motion generation and editing tasks (Tevet et al., 2023; Zhang et al., 2024; Shafir et al., 2023; Xie et al., 2023; Cohan et al., 2024). While these models produce diverse and realistic results, they often lack physical plausibility (Yuan et al., 2022). For instance, issues like ground penetration frequently occur in the generated examples. Such imperfections can affect the quality of the generated motions and limit the model's applicability in real-world scenarios.

For our baseline DM, we use the pre-trained checkpoints provided by Motion Diffusion Model (MDM) (Tevet et al.,

Table 2. Results of the Motion Diffusion experiment. "Inf. per step" is the average rate of generated motion frames with infraction while "infraction" is the average rate of generated motions that at least including one infracting frame. r-ELBO is a reweighted ELBO with the same weighting as in diffusion loss.

Method	Infraction (%) $\downarrow$	Inf. per step (%) $\downarrow$	r-ELBO ($\times 10^{-2}$) $\uparrow$	FID $\downarrow$	KID ($\times 10^{-3}$) $\downarrow$
MDM (baseline DM)	27.66 ± 0.77	7.84 ± 0.27	-1.06 ± 0.02	0.445 ± 0.040	8.27 ± 2.14
Gen-neG (Ours)	24.25 ± 0.35	6.12 ± 0.19	$\mathbf{-1.01 \pm 0.03}$	$\mathbf{0.414 \pm 0.030}$	$\mathbf{6.99 \pm 0.78}$
w/o IS (ablation)	$\mathbf{22.85 \pm 0.18}$	$\mathbf{5.47 \pm 0.18}$	-1.13 ± 0.06	$\mathbf{0.415 \pm 0.030}$	8.40 ± 1.83

2023), tailored for text-conditioned motion generation. MDM is a DDPM model with a transformer-based architecture trained on the HumanML3D dataset (Guo et al., 2022). It uses a pre-trained CLIP embedding module (Radford et al., 2021) for conditioning on the text descriptions. To address the issue of ground penetration, we implement an oracle that labels motions with ground penetration at any point in their duration as negative. We employ Gen-neG with a classifier having the same architecture as MDM, but the CLIP encoder, as the classifier is not text-conditional.

Table 2 summarizes our results of one iteration of Gen-neG on this dataset. We report infraction rate, reweighted ELBO (referring to a uniform schedule of γ_t in Equation (4)). We also report Fréchet Inception Distance (FID) (Heusel et al., 2017), and Kernel Inception Distance (KID) (Bińkowski et al., 2018) to measure quality of samples. Gen-neG improves on all metrics. While the ablation of omitting the IS weighting produces lower infraction rates compared to Gen-neG, it worsens the reweighted ELBO and KID. Hence, Gen-neG improves infraction, with a improved model likelihood and sample quality. We conjecture the relatively smaller improvement in the motion experiment is because the baseline DM predicts $\mathbf{x}_0$ (Zhong et al., 2022).

5. Related Work

Hanneke et al. (2018) proposes a theoretical framework for oracle-based constraints. They however, do not provide practical considerations. More recently, use of negative and invalid data have been explored to improve the training of generative models. Sinha et al. (2021) uses heuristic functions to augment the training set of GANs with negative samples, while (Giannone et al., 2023) utilizes a pre-defined negative set. Meanwhile, data redaction approaches propose methods for removing undesirable learned concepts from pre-trained generative models in safety and security applications (Gandikota et al., 2023; Schramowski et al., 2023). Similarly, (Kong & Chaudhuri, 2023) explores various data redaction methods, with the validity-based approach being the most relevant to our oracle-assisted guidance, although in the context of GAN literature. They implicitly carry out data redaction by integrating it into the discriminator and fine-tuning the generator. Another approach to constrained generative modeling explicitly incorporates the constraints in the model, similar to a final layer that maps to the constraint set (Stoian et al., 2024).

Several studies have explored constrained score-based modeling employing techniques such as diffusion bridges (Wu et al., 2022; Liu et al., 2023b), barrier methods (Fishman et al., 2023), or reflected diffusion (Lou & Ermon, 2023; Fishman et al., 2023) or mirror diffusion (Liu et al., 2023a). Despite being effective, they rely on constraint-specific information such as closed form, linear, or convex constraints. This imposes strong limitations, making them impractical for general problems where such information is unavailable.

6. Conclusion

We have proposed a framework to incorporate constraints into diffusion models. These constraints are defined through an oracle function that categorizes samples as either *good* or *bad*. Importantly, such a flexibility allows for simple integration with human feedback. We have demonstrated our model on different modalities demonstrating how it can benefit safety constraints.

The current limitations we recognize, and the possible future directions for this work are (i) incorporating the true training dataset into the later iterations of the method, as the training dataset only affects the baseline DM. The next stages solely use synthetic data. Although we show theoretically that our guidance only improves the model, this lack of revisiting the true dataset in presence of practical errors and approximations poses challenges for large-scale adoption of our method. Our preliminary experiments of visiting the true dataset at the distillation time have not been successful yet. (ii) Avoiding stacking of classifiers, instead directly learning an artifact that can replace the previous classifier in our method, similar to (De Bortoli et al., 2021), is vital to the computational complexity of the method as the current computational cost scales linearly with the number of classifiers. (iii) Extending Gen-neG on tabular diffusion to support tabular data (Kotelnikov et al., 2023), which compasses a mixture of continuous and categorical data to generalize our work is an inspiring area to future research. (iv) Bridging the gap the diffusion bridge-based approaches and our work which is practically applicable to a larger set of applications is another avenue for future developments.

Acknowledgments

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

Impact Statement

Our work is intended to *improve* generative modeling performance by eliciting improved generalization performance. Better generalization performance will lead to energy savings as less computation is required to generate "good" samples and better performance of systems that can be deployed for societal good such as self-driving cars that crash less often, robotic exoskeletons that are more safely and accurately performant.

References

Anderson, B. D. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982. ISSN 0304-4149. doi: https://doi.org/10.1016/0304-4149(82)90051-5. URL https://www.sciencedirect.com/science/article/pii/0304414982900515.

Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pp. 233–242. PMLR, 2017.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. Discriminator rejection sampling. In *International Conference on Learning Representations*, 2018.

Bansal, A., Chu, H.-M., Schwarzschild, A., Sengupta, S., Goldblum, M., Geiping, J., and Goldstein, T. Universal guidance for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 843–852, 2023.

Bello, I., Zoph, B., Vaswani, A., Shlens, J., and Le, Q. V. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3286–3295, 2019.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying mmd gans. In *International Conference on Learning Representations*, 2018.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pp. 213–229. Springer, 2020.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., and Wallace, E. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270, 2023.

Che, T., Zhang, R., Sohl-Dickstein, J., Larochelle, H., Paull, L., Cao, Y., and Bengio, Y. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. *Advances in Neural Information Processing Systems*, 33:12275–12287, 2020.

Cohan, S., Tevet, G., Reda, D., Peng, X. B., and van de Panne, M. Flexible motion in-betweening with diffusion models. *arXiv preprint arXiv:2405.11126*, 2024.

Cui, H., Radosavljevic, V., Chou, F.-C., Lin, T.-H., Nguyen, T., Huang, T.-K., Schneider, J., and Djuric, N. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2090–2096. IEEE, 2019.

De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, pp. 8780–8794, 2021.

Djuric, N., Radosavljevic, V., Cui, H., Nguyen, T., Chou, F.-C., Lin, T.-H., and Schneider, J. Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks. *arXiv preprint arXiv:1808.05819*, 1(2):6, 2018.

Fishman, N., Klarner, L., De Bortoli, V., Mathieu, E., and Hutchinson, M. Diffusion models for constrained domains. *arXiv preprint arXiv:2304.05364*, 2023.

Gandikota, R., Materzynska, J., Fiotto-Kaufman, J., and Bau, D. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023.

Giannone, G., Regenwetter, L., Srivastava, A., Gutfreund, D., and Ahmed, F. Learning from invalid data: On constraint satisfaction in generative models. *arXiv preprint arXiv:2306.15166*, 2023.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Guo, C., Zou, S., Zuo, X., Wang, S., Ji, W., Li, X., and Cheng, L. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5152–5161, 2022.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2255–2264, 2018.

Hanneke, S., Kalai, A. T., Kamath, G., and Tzamos, C. Actively avoiding nonsense in generative models. In *Conference On Learning Theory*, pp. 209–227. PMLR, 2018.

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

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Kim, D., Kim, Y., Kang, W., and Moon, I.-C. Refining generative process with discriminator guidance in score-based diffusion models. *arXiv preprint arXiv:2211.17091*, 2022.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kong, Z. and Chaudhuri, K. Data redaction from pre-trained gans. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 638–677. IEEE, 2023.

Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pp. 17564–17579. PMLR, 2023.

Lee, N., Choi, W., Vernaza, P., Choy, C. B., Torr, P. H., and Chandraker, M. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 336–345, 2017.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Liu, G.-H., Chen, T., Theodorou, E., and Tao, M. Mirror diffusion models for constrained and watermarked generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Liu, X., Wu, L., Ye, M., et al. Learning diffusion bridges on constrained domains. In *The Eleventh International Conference on Learning Representations*, 2023b.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pp. 2575–2582. IEEE, 2018.

Lou, A. and Ermon, S. Reflected diffusion models. *arXiv preprint arXiv:2304.04740*, 2023.

Naderiparizi, S., Scibior, A., Munk, A., Ghadiri, M., Baydin, A. G., Gram-Hansen, B. J., De Witt, C. A. S., Zinkov, R., Torr, P., Rainforth, T., et al. Amortized rejection sampling in universal probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, pp. 8392–8412. PMLR, 2022.

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.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Niedoba, M., Lavington, J. W., Liu, Y., Lioutas, V., Sefas, J., Liang, X., Green, D., Dabiri, S., Zwartsenberg, B., Scibior, A., et al. A diffusion-model of joint interactive navigation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. Image transformer. In *International conference on machine learning*, pp. 4055–4064. PMLR, 2018.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Risken, H. and Risken, H. *Fokker-planck equation*. Springer, 1996.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and 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.

Schramowski, P., Brack, M., Deiseroth, B., and Kersting, K. Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22522–22531, 2023.

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

Shafir, Y., Tevet, G., Kapon, R., and Bermano, A. H. Human motion diffusion as a generative prior. *arXiv preprint arXiv:2303.01418*, 2023.

Shaw, P., Uszkoreit, J., and Vaswani, A. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Sinha, A., Ayush, K., Song, J., Uzkent, B., Jin, H., and Ermon, S. Negative data augmentation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ovp8dvB8IBH.

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

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6048–6058, 2023a.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Understanding and mitigating copying in diffusion models. *Advances in Neural Information Processing Systems*, 36:47783–47803, 2023b.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 11918–11930, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Stoian, M. C., Dyrmishi, S., Cordy, M., Lukasiewicz, T., and Giunchiglia, E. How realistic is your synthetic data? constraining deep generative models for tabular data. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tBROYsEz9G.

Sugiyama, M., Suzuki, T., and Kanamori, T. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

Tan, S., Wong, K., Wang, S., Manivasagam, S., Ren, M., and Urtasun, R. Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 892–901, 2021.

Tevet, G., Raab, S., Gordon, B., Shafir, Y., Cohen-or, D., and Bermano, A. H. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Warrington, A., Naderiparizi, S., and Wood, F. Coping with simulators that don't always return. In *International Conference on Artificial Intelligence and Statistics*, pp. 1748–1758. PMLR, 2020.

Wu, K., Peng, H., Chen, M., Fu, J., and Chao, H. Rethinking and improving relative position encoding for vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10033–10041, 2021.

Wu, L., Gong, C., Liu, X., Ye, M., and Liu, Q. Diffusion-based molecule generation with informative prior bridges. *Advances in Neural Information Processing Systems*, 35: 36533–36545, 2022.

Wu, L., Trippe, B. L., Naesseth, C. A., Blei, D., and Cunningham, J. P. Practical and asymptotically exact conditional sampling in diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Xie, Y., Jampani, V., Zhong, L., Sun, D., and Jiang, H. Omnicontrol: Control any joint at any time for human motion generation. *arXiv preprint arXiv:2310.08580*, 2023.

Yang, Q. and Koutsopoulos, H. N. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3):113–129, 1996.

Yuan, Y., Song, J., Iqbal, U., Vahdat, A., and Kautz, J. Physdiff: Physics-guided human motion diffusion model. *arXiv preprint arXiv:2212.02500*, 2022.

Zhan, W., Sun, L., Wang, D., Shi, H., Clausse, A., Naumann, M., Kummerle, J., Konigshof, H., Stiller, C., de La Fortelle, A., et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Zhang, M., Cai, Z., Pan, L., Hong, F., Guo, X., Yang, L., and Liu, Z. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., and Pavone, M. Guided conditional diffusion for controllable traffic simulation. *arXiv preprint arXiv:2210.17366*, 2022.

Zwartsenberg, B., Scibior, A., Niedoba, M., Lioutas, V., Sefas, J., Liu, Y., Dabiri, S., Lavington, J. W., Campbell, T., and Wood, F. Conditional permutation invariant flows. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=DUsgPi3oCC.

A. Proofs

A.1. Proof of Theorem 3.1

Proof. Let α and $(1-\alpha)$ be the prior probabilities of positive and negative examples under $p_\theta(\mathbf{x}_t;t)$. Note that α remains independent of t because

$$\begin{aligned}\alpha &= \int p_\theta(\mathbf{x}_t;t)p_\theta(\mathbf{x}_0|\mathbf{x}_t)p_\theta(y=1|\mathbf{x}_0)\,d\mathbf{x}_0\,d\mathbf{x}_t = \int p_\theta(\mathbf{x}_t;t)p_\theta(\mathbf{x}_0|\mathbf{x}_t)\mathcal{O}(\mathbf{x}_0)\,d\mathbf{x}_0\,d\mathbf{x}_t \\ &= \int p_\theta(\mathbf{x}_0,\mathbf{x}_t;t)\mathcal{O}(\mathbf{x}_0)\,d\mathbf{x}_0 d\mathbf{x}_t = \int p_\theta(\mathbf{x}_0;0)\mathcal{O}(\mathbf{x}_0)\,d\mathbf{x}_0.\end{aligned}$$

The objective in Equation (13) is equal to

$$\begin{aligned}&-\mathbb{E}_t\left[\mathbb{E}_{p_\theta(\mathbf{x}_t)}\left[\mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t)}\left[\mathcal{O}(\mathbf{x}_0)\right]\log C_\phi(\mathbf{x}_t;t) + \mathbb{E}_{p_\theta(\mathbf{x}_0|\mathbf{x}_t)}\left[(1-\mathcal{O}(\mathbf{x}_0))\right]\log(1-C_\phi(\mathbf{x}_t;t))\right]\right] \\ &\qquad = -\mathbb{E}_t\left[\mathbb{E}_{p_\theta(\mathbf{x}_t)}\left[p(y=1|\mathbf{x}_t)\log C_\phi(\mathbf{x}_t;t) + p(y=0|\mathbf{x}_t)\log(1-C_\phi(\mathbf{x}_t;t))\right]\right] \quad (15)\end{aligned}$$

This is equivalent to Equation (6) after replacing q with p_θ, i.e. sampling from the reverse process instead of the forward. Therefore, its optimal solution follows Equation (7). Hence,

$$\begin{aligned}&\nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t;t) + \nabla_{\mathbf{x}_t}\log C_{\phi^*}(\mathbf{x}_t;t) \\ &= \nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t;t) + \nabla_{\mathbf{x}_t}\log \alpha p_\theta(\mathbf{x}_t|y=1;t) \\ &\qquad - \nabla_{\mathbf{x}_t}\log\Big[\overbrace{\alpha p_\theta(\mathbf{x}_t|y=1;t)+(1-\alpha)p_\theta(\mathbf{x}_t|y=0;t)}^{p_\theta(\mathbf{x}_t;t)}\Big] \\ &= \nabla_{\mathbf{x}_t}\log \alpha p_\theta(\mathbf{x}_t|y=1;t) = \nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t|y=1;t)\end{aligned}$$

□

A.2. Proof of Corollary 3.2

Proof. Since p_{θ,ϕ^*} is defined as the distribution generated by simulating the SDE in equation 2, its score function $\nabla_{\mathbf{x}_t}\log p_{\theta,\phi^*}(\mathbf{x}_t;t)$ is by definition equal to $s_{\theta,\phi^*}(\mathbf{x}_t;t)$ (Risken & Risken, 1996; Song & Ermon, 2019). Similarly for the baseline DM we have $s_\theta(\mathbf{x}_t;t) = \nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t;t)$. Therefore,

$$\nabla_{\mathbf{x}_t}\log p_{\theta,\phi^*}(\mathbf{x}_t;t) = s_{\theta,\phi^*}(\mathbf{x}_t;t) \overset{\text{Equation (10)}}{=\!=} s_\theta(\mathbf{x}_t;t) + \nabla_{\mathbf{x}_t}\log C_{\phi^*}(\mathbf{x}_t;t) \quad (16)$$

$$= \nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t;t) + \nabla_{\mathbf{x}_t}\log C_{\phi^*}(\mathbf{x}_t;t) \overset{\text{Thm. 3.1}}{=\!=} \nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t|y=1) \quad (17)$$

Here we derived $s_{\theta,\phi^*}(\mathbf{x}_t;t) = \nabla_{\mathbf{x}_t}\log p_\theta(\mathbf{x}_t|y=1)$. By (Anderson, 1982), we proved the first statement.

The second statement follows by decomposing $p_\theta(\mathbf{x}_t|y=1)$:

$$p_{\theta,\phi^*}(\mathbf{x}) = p_\theta(\mathbf{x}|y=1) \propto p_\theta(\mathbf{x})\mathcal{O}(\mathbf{x}) \quad\Rightarrow\quad p_{\theta,\phi^*}(\mathbf{x}) = 0 \quad \forall \mathbf{x}\in\Omega^{\complement}.$$

For the last statement, we have

$$\left.\begin{aligned}p_{\theta,\phi^*}(\mathbf{x}) &= \tfrac{p_\theta(\mathbf{x})\mathcal{O}(\mathbf{x})}{\int p_\theta(\mathbf{x})\mathcal{O}(\mathbf{x})\,d\mathbf{x}} \\ \int p_\theta(\mathbf{x})\mathcal{O}(\mathbf{x})\,d\mathbf{x} &\le 1\end{aligned}\right\} \Rightarrow p_{\theta,\phi^*}(\mathbf{x}) \ge p_\theta(\mathbf{x}) \quad \forall\mathbf{x}\in\Omega$$

$$\overset{\mathcal{D}\subseteq\Omega}{\Longrightarrow} \log p_{\theta,\phi^*}(\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}}\log p_{\theta,\phi^*}(\mathbf{x}) \ge \sum_{\mathbf{x}\in\mathcal{D}}\log p_\theta(\mathbf{x}) = \log p_\theta(\mathcal{D}).$$

□

We demonstrate this corollary with a one-dimension density in Figure 5. We show a "base distribution" $p(x)$ and the positive and negative regions Ω and $\Omega^{\complement}$, respectively. We can see that the distribution $p(x|y=1)$ assigns no mass to $\Omega^{\complement}$ and has a larger mass assigned to any point in Ω.

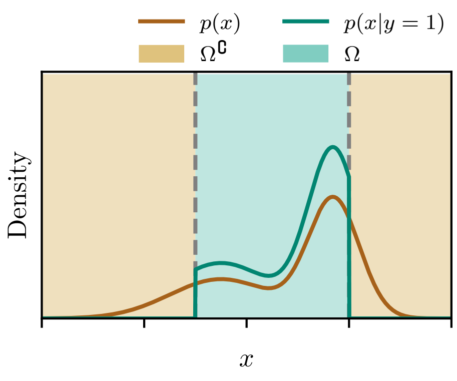


Figure 5. We use this one-dimensional density plot to show how we guide the generation process towards the positive support region indicated by the oracle. The base distribution $p(x)$ is a mixture of two Gaussian distributions shown by the brown curve. We show the regions allowed and disallowed by the oracle respectively by the cyan and light brown shaded areas. Gen-neG with a Bayes optimal classifier targets the distribution $p(x|y=1)$ that has probability mass in the allowed region and assigns zero probability outside this region.

A.3. Equivalence of cross-entropy loss minimization and KL divergence minimization

This is a well-known result in the literature. Nonetheless, we include the result and its proof here for completeness and ease of reference.

Claim: minimizing the cross-entropy loss between the classifier output and the true label is equivalent to minimizing the KL divergence between the classifier output and the Bayes optimal classifier.

Proof. The Bayes optimal classifier $C^*(\mathbf{x}_t;t)$ approximates $q(y=1|\mathbf{x}_t)$. Let $p_\phi(y|\mathbf{x}_t)$ be the distribution represented by the learned classifier $C_\phi(\mathbf{x}_t;t)$ i.e., $p_\phi(y=1|\mathbf{x}_t)=C_\phi(\mathbf{x}_t;t)$. For an arbitrary diffusion time step t, the expected KL divergence between the Bayes optimal and the learned classifier therefore is

$$\begin{aligned}
&\mathbb{E}_{q(\mathbf{x}_t)}\left[\mathrm{KL}\left(q(y|\mathbf{x}_t)||p_\phi(y|\mathbf{x}_t)\right)\right] \\
&= \mathbb{E}_{q(\mathbf{x}_t)}\left[\mathbb{E}_{q(y|\mathbf{x}_t)}\left[\log\frac{q(y|\mathbf{x}_t)}{p_\phi(y|\mathbf{x}_t)}\right]\right] && (18)\\
&= \mathbb{E}_{q(\mathbf{x}_t)}\left[q(y=1|\mathbf{x}_t)\log\frac{q(y=1|\mathbf{x}_t)}{C_\phi(\mathbf{x}_t;t)} + q(y=0|\mathbf{x}_t)\log\frac{q(y=0|\mathbf{x}_t)}{1-C_\phi(\mathbf{x}_t;t)}\right] && (19)\\
&= \mathbb{E}_{q(\mathbf{x}_t)}\left[H(q(y|\mathbf{x}_t))\right] - \mathbb{E}_{q(\mathbf{x}_t)}\left[q(y=1|\mathbf{x}_t)\log C_\phi(\mathbf{x}_t;t) + q(y=0|\mathbf{x}_t)\log(1-C_\phi(\mathbf{x}_t;t))\right]. && (20)
\end{aligned}$$

The first term is the expected entropy of the optimal classifier and is independent of ϕ. Therefore,

$$\begin{aligned}
&\arg\min_\phi \mathbb{E}_{q(\mathbf{x}_t)}\left[\mathrm{KL}\left(q(y|\mathbf{x}_t)||p_\phi(y|\mathbf{x}_t)\right)\right] \\
&= \arg\min_\phi -\mathbb{E}_{q(\mathbf{x}_t)}\left[q(y=1|\mathbf{x}_t)\log C_\phi(\mathbf{x}_t;t) + q(y=0|\mathbf{x}_t)\log(1-C_\phi(\mathbf{x}_t;t))\right] && (21)\\
&= \arg\min_\phi \mathrm{CE}(q(y|\mathbf{x}_t), p_\phi(y|\mathbf{x}_t)). && (22)
\end{aligned}$$

□

Note that Equation (6) is the expected cross entropy for different time steps t.

A.4. Connection between Eq. (11) and Eq. (13)

In this section we make the connection between Equation (9) and Equation (13) more clear. We start with Equation (13), ignoring the outer expectation with respect to t, is equal to

$$
\begin{aligned}
&- \left(\mathbb{E}_{p_\theta(\mathbf{x}_0,\mathbf{x}_t)}\left[\mathcal{O}(\mathbf{x}_0)\log C_\phi(\mathbf{x}_t;t) + (1-\mathcal{O}(\mathbf{x}_0))\log(1-C_\phi(\mathbf{x}_t;t))\right]\right) && (23)\\
&= -\int p_\theta(\mathbf{x}_0,\mathbf{x}_t)\Big(p_\theta(y=1|\mathbf{x}_0)\log C_\phi(\mathbf{x}_t;t) + p_\theta(y=0|\mathbf{x}_0)\log(1-C_\phi(\mathbf{x}_t;t))\Big)\,d\mathbf{x}_0 d\mathbf{x}_t && (24)\\
&= -\int p_\theta(y=1)p_\theta(x_0|y=1)p_\theta(x_t|x_0)\log C_\phi(\mathbf{x};t)\,d\mathbf{x}_0 d\mathbf{x}_t\\
&\quad -\int p_\theta(y=0)p_\theta(x_0|y=0)p_\theta(x_t|x_0)\log(1-C_\phi(\mathbf{x}_t;t))\,d\mathbf{x}_0 d\mathbf{x}_t && (25)\\
&\approx -\int p_\theta(y=1)p_\theta(x_0|y=1)q(x_t|x_0)\log C_\phi(\mathbf{x};t)\,d\mathbf{x}_0 d\mathbf{x}_t\\
&\quad -\int p_\theta(y=0)p_\theta(x_0|y=0)q(x_t|x_0)\log(1-C_\phi(\mathbf{x}_t;t))\,d\mathbf{x}_0 d\mathbf{x}_t && (26)\\
&= \alpha\mathbb{E}_{p_\theta(\mathbf{x}_0|y=1)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log C_\phi(\mathbf{x}_t;t)\right]\right] + (1-\alpha)\mathbb{E}_{p_\theta(\mathbf{x}_0|y=0)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log(1-C_\phi(\mathbf{x}_t;t))\right]\right] && (27)\\
&:= \frac{1}{N}\sum_{\mathbf{x}_0\in\widehat{\mathcal{D}}^+}\alpha\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log C_\phi(\mathbf{x}_t;t)\right] + \frac{1}{N}\sum_{\mathbf{x}_0\in\widehat{\mathcal{D}}^-}(1-\alpha)\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log(1-C_\phi(\mathbf{x}_t;t))\right], && (28)
\end{aligned}
$$

which recovers Equation (11).

A.5. Gen-neG's objective function

Here we show why Equation (11) is an importance sampling estimator of the original objective function in Equation (9).

$$
\begin{aligned}
\mathcal{L}^{\text{cls}}_\phi(\alpha) :=\ & \alpha\mathbb{E}_{p_\theta(\mathbf{x}_0|y=1)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log C_\phi(\mathbf{x}_t;t)\right]\right]\\
&+ (1-\alpha)\mathbb{E}_{p_\theta(\mathbf{x}_0|y=0)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log(1-C_\phi(\mathbf{x}_t;t))\right]\right] && (29)\\
=\ & -\mathbb{E}_{p_\theta(y)}\left[\mathbb{E}_{p_\theta(\mathbf{x}_0|y)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[y\log C_\phi(\mathbf{x}_t;t) + (1-y)\log(1-C_\phi(\mathbf{x}_t;t))\right]\right]\right]. && (30)
\end{aligned}
$$

Now we apply importance sampling to $p_\theta(y)$ by sampling from $\pi(y)$ as the proposal distribution. Therefore,

$$
\begin{aligned}
\mathcal{L}^{\text{cls}}_\phi(\alpha) =\ & -\mathbb{E}_{p_\theta(y)}\left[\mathbb{E}_{p_\theta(\mathbf{x}_0|y)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[y\log C_\phi(\mathbf{x}_t;t) + (1-y)\log(1-C_\phi(\mathbf{x}_t;t))\right]\right]\right] && (31)\\
=\ & -\mathbb{E}_{\pi(y)}\left[\frac{p_\theta(y)}{\pi(y)}\mathbb{E}_{p_\theta(\mathbf{x}_0|y)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[y\log C_\phi(\mathbf{x}_t;t) + (1-y)\log(1-C_\phi(\mathbf{x}_t;t))\right]\right]\right] && (32)\\
=\ & \frac{p_\theta(y=1)}{\pi(y=1)}\mathbb{E}_{p_\theta(\mathbf{x}_0|y=1)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log C_\phi(\mathbf{x}_t;t)\right]\right]\\
&+ \frac{p_\theta(y=0)}{\pi(y=0)}\mathbb{E}_{p_\theta(\mathbf{x}_0|y=0)}\left[\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log(1-C_\phi(\mathbf{x}_t;t))\right]\right] && (33)\\
=\ & \mathbb{E}_{p_\theta(\mathbf{x}_0|y=1)}\left[\frac{\alpha}{\pi(y=1)}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log C_\phi(\mathbf{x}_t;t)\right]\right]\\
&+ \mathbb{E}_{p_\theta(\mathbf{x}_0|y=0)}\left[\frac{1-\alpha}{\pi(y=0)}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[-\log(1-C_\phi(\mathbf{x}_t;t))\right]\right] && (34)
\end{aligned}
$$

In our case, $\pi(y)$ is a uniform Bernoulli distribution i.e., $\pi(y=1)=\pi(y=0)=0.5$. Therefore, minimizing Equation (11) is equivalent to minimizing a Mone Carlo estimate of Equation (34).

A.6. Error bound on the prior probability of positive examples α

In practice, the prior probability of generating positive samples $\alpha = p(y=1)$ is not accessible, and we use an empirical $\hat{\alpha}$ obtained from the generated synthetic dataset to estimate it. We perform a statistical analysis to investigate the difference between joint log-probabilities that we want to maximize based on the true α and the empirically estimated α. We start with a naive Lemma A.1 about Delta method:

Lemma A.1. *(Delta method) Suppose $\hat{\theta}_n$ follows an asymptotic normal distribution, $\sqrt{n}(\hat{\theta}_n - \theta)$ converges to $\mathcal{N}(0, \sigma^2)$ in distribution as $n \to \infty$, then if g is a continuous function with a well-defined first derivative at θ and $g'(\alpha) \neq 0$, then*

$$\sqrt{n}(g(\hat{\theta}_n) - g(\theta)) \xrightarrow{D} \mathcal{N}(0, (g'(\theta))^2\sigma^2). \tag{35}$$

Proof. By Taylor approximation expansion,

$$g(\hat{\theta}_n) \approx g(\theta) + g'(\theta)(\hat{\theta}_n - \theta) \Rightarrow \sqrt{n}(g(\hat{\theta}_n) - g(\theta)) \approx \sqrt{n}g'(\theta)(\hat{\theta}_n - \theta) \tag{36}$$

by subtracting $g(\theta)$ and multiplying $\sqrt{n}$ on the both sides. Therefore, $\sqrt{n}(g(\hat{\theta}_n) - g(\theta)) \xrightarrow{D} \mathcal{N}(0, (g'(\theta))^2\sigma^2)$. □

Theorem A.2. *The difference of true log-joint probability and the estimated log-joint probability converges in distribution $\mathcal{N}\left(0, \frac{1}{\alpha} - \frac{p_\theta(\mathbf{x}_t|y=1;t) - p_\theta(\mathbf{x}_t|y=0;t)}{\alpha p_\theta(\mathbf{x}_t|y=1;t) + (1-\alpha)p_\theta(\mathbf{x}_t|y=0;t)}(\alpha - \alpha^2)\right)$.*

Given a fully synthetic dataset of size N, we estimate α with $\hat{\alpha} = \frac{1}{N}\sum_i \mathbb{1}[y_i = 1]$, and its expectation and variance are:

$$\mathbb{E}[\hat{\alpha}] = \mathbb{E}\left[\frac{1}{N}\sum_i \mathbb{1}[y_i = 1]\right] = \frac{1}{N}\sum_i \mathbb{E}[\mathbb{1}[y_i = 1]] = \frac{1}{N}\sum_i p_\theta(y_i = 1) = \alpha \tag{37}$$

$$\mathrm{Var}[\hat{\alpha}] = \mathrm{Var}\left[\frac{1}{N}\sum_i \mathbb{1}[y_i = 1]\right] = \frac{1}{N^2}\sum_i \left(\mathbb{E}\left[\mathbb{1}^2[y_i = 1]\right] - (\mathbb{E}[\mathbb{1}[y_i = 1]])^2\right) = \frac{\alpha - \alpha^2}{N} \tag{38}$$

Let $N \to \infty$, by central limit theorem (CLT), we have $\hat{\alpha} \xrightarrow{D} \mathcal{N}\left(\alpha, \frac{\alpha - \alpha^2}{N}\right)$. If $\frac{\alpha-\alpha^2}{N} \to 0$, we have $\hat{\alpha} \to \alpha$, then by Theorem 3.1, the joint log-probability is:

$$\begin{aligned}\log p_\theta(\mathbf{x}_t; t, \alpha) + \log C_{\phi^*}(\mathbf{x}_t; t, \alpha) &= \log(\alpha p_\theta(\mathbf{x}_t|y = 1, t) + (1 - \alpha)p_\theta(\mathbf{x}_t|y = 0, t)) \\ &\quad + \log\left(\frac{\alpha p_\theta(\mathbf{x}_t|y = 1, t)}{\alpha p_\theta(\mathbf{x}_t|y = 1, t) + (1 - \alpha)p_\theta(\mathbf{x}_t|y = 1, t)}\right) \qquad (39) \\ &= \log \alpha + \log p_\theta(\mathbf{x}_t|y = 1, t) \qquad (40)\end{aligned}$$

otherwise,

$$\begin{aligned}\log p_\theta(\mathbf{x}_t; t, \alpha) + \log C_{\phi^*}(\mathbf{x}_t; t, \hat{\alpha}) &= \log(\alpha p_\theta(\mathbf{x}_t|y = 1, t) + (1 - \alpha)p_\theta(\mathbf{x}_t|y = 0, t)) \\ &\quad + \log\left(\frac{\hat{\alpha} p_\theta(\mathbf{x}_t|y = 1, t)}{\hat{\alpha} p_\theta(\mathbf{x}_t|y = 1, t) + (1 - \hat{\alpha})p_\theta(\mathbf{x}_t|y = 1, t)}\right) \qquad (41) \\ &= \log\left(\frac{\alpha p_\theta(\mathbf{x}_t|y = 1; t) + (1 - \alpha)p_\theta(\mathbf{x}_t|y = 0; t)}{\hat{\alpha} p_\theta(\mathbf{x}_t|y = 1; t) + (1 - \hat{\alpha})p_\theta(\mathbf{x}_t|y = 0; t)}\right) \\ &\quad + \log \hat{\alpha} + \log p_\theta(\mathbf{x}_t|y = 1, t) \qquad (42)\end{aligned}$$

The difference between Equation (42) and Equation (40) is:

$$\ell(\mathbf{x}_t; t, \hat{\alpha}) = \left|\log\left(\frac{\hat{\alpha}}{\alpha}\right) - \log\left(\frac{\hat{\alpha} p_\theta(\mathbf{x}_t|y = 1; t) + (1 - \hat{\alpha})p_\theta(\mathbf{x}_t|y = 0; t)}{\alpha p_\theta(\mathbf{x}_t|y = 1; t) + (1 - \alpha)p_\theta(\mathbf{x}_t|y = 0; t)}\right)\right| \tag{43}$$

Let $g(\alpha) = \log \alpha - \log\Big(\alpha p_\theta(\mathbf{x}_t|y = 1; t) + (1 - \alpha)p_\theta(\mathbf{x}_t|y = 0; t)\Big)$, then

$$g'(\alpha) = \frac{1}{\alpha} - \frac{p_\theta(\mathbf{x}_t|y = 1; t) - p_\theta(\mathbf{x}_t|y = 0; t)}{\alpha p_\theta(\mathbf{x}_t|y = 1; t) + (1 - \alpha)p_\theta(\mathbf{x}_t|y = 0; t)}. \tag{44}$$

Based on Lemma A.1, we have

$$\sqrt{N}\left(\ell(\mathbf{x}_t; t, \hat{\alpha})\right) \xrightarrow{D} \mathcal{N}(0, g'(\alpha)(\alpha - \alpha^2)) \tag{45}$$

where $g'(\alpha)$ follows Equation (44).

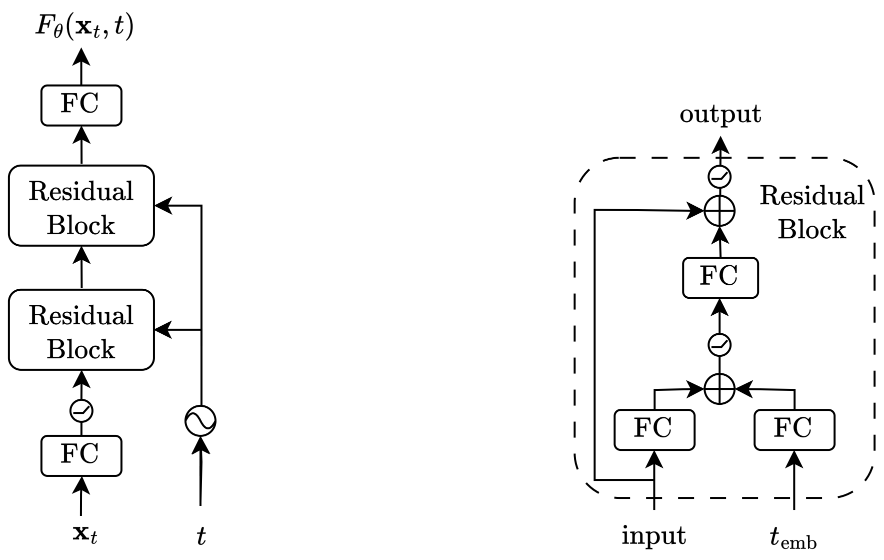


Figure 6. Architecture of the network in the checkerboard experiment. Left: the overall of the model. Right: detailed architecture of our "Residual Block". In this architecture, timestep is embedded using sinusoidal embedding and all nonlinearities are SiLU. The output of the network $F_\theta(\mathbf{x}_t, t)$ is then used in a preconditioning function to get an estimate of $\mathbf{x}_0$.

B. Experimental details

B.1. Checkerboard Experiment

Architecture We use a fully connected network with 2 residual blocks as shown in Figure 6. The hidden layer size in our experiment is 256 and timestep embeddings (output of the sinusiodal embedding layer) is 128. Our classifier has a similar architecture, the only difference is that the classifier has a different output dimension of one. Our baseline DM and classifier networks both have around 330k parameters.

Training the baseline DM We train our models baseline models on a single GPU, (we use either of GeForce GTX 1080 Ti or GeForce GTX TITAN X) for 30,000 iterations. We use the Adam optimizer (Kingma & Ba, 2014) with a batch size of 3×10^{-4} and full-batch training i.e., our batch size is 1000 which is the same as the training dataset size.

Training the classifiers Each classifier is trained on a fully-synthetic dataset of 100k samples which consists of 50k positive and 50k negative samples. This dataset is generated with 100 diffusion steps. We train the classifier for 20k iterations with a batch size of 8192. We use Adam optimizer with a learning rate of 3×10^{-3}.

Distillation The distilled models have the same architecture and hyperparameters as the baseline DM model. They are trained for 250k iterations on the true dataset with a batch size of 1000. We use Adam optimizer with a learning rate of 3×10^{-4}.

Diffusion process We use the EDM framework in this experiment with a preconditioning similar to the one proposed in Karras et al. (2022). In particular, the following precoditioning is applied to the the network in Figure 6, called $F_\theta(\mathbf{x}_t, t)$, to get $D_\theta(\mathbf{x}_t; t)$ which returns an estimate of $\mathbf{x}_0$.

$$D_\theta(\mathbf{x}_t; t) = \frac{\sigma_{\text{data}}^2}{\sigma(t)^2 + \sigma_{\text{data}}^2}\mathbf{x}_t + \sigma(t) F_\theta\left(\frac{1}{\sqrt{\sigma(t)^2 + \sigma_{\text{data}}^2}}\mathbf{x}_t; \frac{1}{4}\ln(\sigma(t))\right), \tag{46}$$

where $\sigma_{\text{data}} = 1$. Since smaller noise levels are important in our application, we changed the training distribution of t from the log-Normal used in Karras et al. (2022) to a log-uniform with the support of $\sigma_{\max} = 80$ and $\sigma_{\min} = 2 \times 10^{-3}$. In total, we spent around 250 GPU-hours for this experiment.

Sampling We use the second-order Heun solver of Karras et al. (2022) with 100 sampling steps and $S_{\text{churn}} = 10$. We modify the schedule of σ to a log-linear schedule from σ_{max} to σ_{min}.

B.2. Traffic Scene Generation

Overview This section provides additional details for the traffic scene generation task. The architectures for training the baseline DM model, classifiers and distillation models are majorly based on transformers introduced by Vaswani et al. (2017) . In particular, the architecture backbone consists of an encoder, a stack of attention residual blocks, and a decoder. Each of them will be discussed in detail later. The original data input shape is $[B, A, F]$ corresponding to A vehicles and F feature dimensions in a batch with B many scenes.

In terms of parameters, the attention layers comprise the major portion of the entire architectures. This leads to the difference in decoder being relatively minor, and the resulting architectures all contain approximately 6.3 million parameters. We use NVIDIA A100 GPUs for training and validating models, synthetic datasets generation with around 400 GPU-hours in total. We train each model with a batch size of 64 and Adam optimizer with a learning rate of 10^{-4}.

Encoder and time embeddings To generate input features, we use sinusoidal positional embeddings to embed the diffusion time step and 2-layer MLP with activation function SiLU to embed the original data separately into $H = 196$ hidden feature dimensions. The sum of the two embeddings is the input that is fed into the attention-based architecture.

Self-attention and cross-attention layers The major implementation of multi-head ($k = 4$) attention blocks is built on Transformer (Vaswani et al., 2017). Applying self-attention across agents enables model to learn the multi-agent interactions, while applying map-conditional cross-attention between agents and map allows agents to interact with the road representations. To prepare road image for model input, we use a convolutional neural network and a feed-forward network (Carion et al., 2020) to generate a lower-resolution map $m' \in \mathbb{R}^{196\times32\times32}$ from the original image $m \in \mathbb{R}^{3\times256\times256}$. Since the transformer architecture is permutation-invariant, we add a 2D positional encoding (Parmar et al., 2018; Bello et al., 2019) based on m' on the top of the map representation to preserve the spatial information of the image.

Relative Positional Encodings (RPEs) During experiments, we find the collision rate is much higher than the offroad rate. In order to effectively lower the frequency of or completely avoid vehicle collision occurrence, we manage to capture the relative positions by performing relative positional encodings (RPEs) in self-attention residual blocks and enforce the vehicles being aware of the other vehicles in close proximity in each scene. Following Shaw et al. (2018); Wu et al. (2021); Harvey et al. (2022), we compute the distances of each pair of vehicles and summarise into a tensor of shape $[B, A, A]$, where d_{ij}^b is the distance between vehicle i and j in the b^{th} scene. We choose to use sinusoidal embeddings (similar to how we embed diffusion time t) to parameterize d_{ij}^b rather than logarithm function $f_{\text{RPE}}(d_{ij}^b) = \log(1 + d_{ij}^b)$, as we need to adequately amplify the pairwise distances between vehicles when it is comparably small. We perform this operation together with diffusion time embedding at each diffusion time step, and we regard their sum as the complete pairwise distance embeddings. The resulting embedding tensor $\mathbf{p}$ is of the shape $[B, A, A, H]$, where $\mathbf{p}_{ij}^b$ is the encoding vector of length H representing the pairwise distance of vehicle i and j in the b^{th} scene.

In each scene, we have an input sequence, $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_A)$, and each $\mathbf{x}_i$ is linearly transformed to query $\mathbf{q}_i = W^Q\mathbf{x}_i$, key $\mathbf{k}_i = W^K\mathbf{x}_i$ and value $\mathbf{v}_i = W^V\mathbf{x}_i$. We also apply linear transformation onto RPEs to obtain query $\mathbf{p}_{ij}^Q = U^Q\mathbf{p}_{ij}$, key $\mathbf{p}_{ij}^K = U^K\mathbf{p}_{ij}$ and value $\mathbf{p}_{ij}^V = U^V\mathbf{p}_{ij}$. Then the add-on output from the self-attention residual block is the aggregated outputs of the vanilla transformer and the relative-position-aware transformer:

$$\mathbf{x}_i^{\text{output}} = \mathbf{x}_i + \sum_{j=1}^{A} \alpha_{ij}(\mathbf{v}_j + \mathbf{p}_{ij}^V) \tag{47}$$

$$\text{where} \quad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{A}\exp(e_{ik})} \text{ and } e_{ij} = \frac{\mathbf{q}_i^\top\mathbf{k}_j + \mathbf{p}_{ij}^{Q^\top}\mathbf{k}_j + \mathbf{q}_i^\top\mathbf{p}_{ij}^K}{\sqrt{d_{\mathbf{x}}}} \tag{48}$$

Decoder The settings for baseline, distillation models and classifiers are almost identical except the decoder for producing the final output. For baseline and distillation models, we apply 2-layer MLP and reconstruct the output of the shape $[B, A, H]$ from the final attention layer into $[B, A, D]$ through the decoder. To ensure we output individual label for each

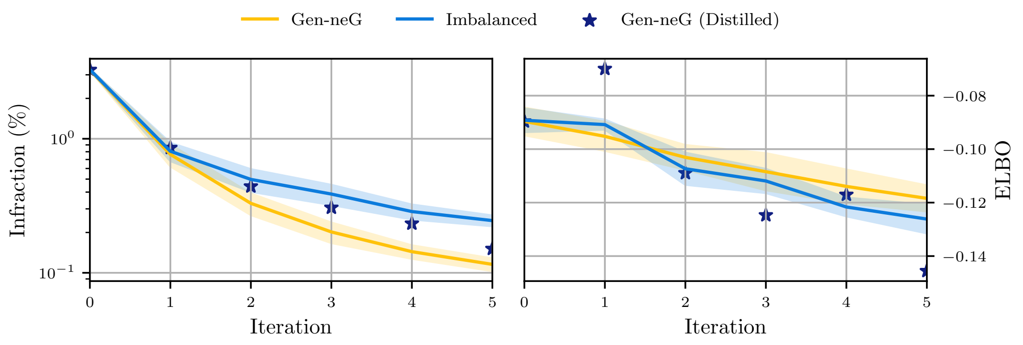


Figure 7. Infraction and ELBO estimations from different iterations of Gen-neG, distilled Gen-neG and an ablated version of it without label imbalance correction. Gen-neG achieves a lower infraction rate and a comparable ELBO.

vehicle with classifiers, we conduct the operations as follows. The decoder takes the hidden representation of the shape $[B, A, H]$ and produces a tensor with feature dimension $F' = 1$ with a 2-layer MLP, which is the predicted labels from the classifiers.

B.3. Motion Diffusion

For the task of text-conditional motion generation, we use the HumanML3D dataset (Guo et al., 2022). This dataset contains 14,616 human motions annotated by 44,970 textual descriptions. It includes motions between 2 and 10 seconds in length and their total length amounts to 28.59 hours. Each motion is between 36 and 196 frames, with the majority of them comprising 196 frames. Each frame is represented by a 263-dimensional feature vector, resulting in a dimensionality of over $51,000$ for the largest motions.

We used the official implementation of MDM[2] for our Motion Diffusion experiment. For the baseline DM, we used their officially released best pretrained checkpoint of text-to-motion task on HumanML3D dataset. We generate a synthetic dataset of around 250k positive and 250k negative examples from the baseline DM which is a DDPM-based model with 1000 diffusion steps. We then define our classifier architecture using their code base. Following our other experiments, our classifier architecture is the same as the baseline DM model. We train the classifier with a batch size of 128 and a learning rate of 10^{-4} for 300k iterations. Otherwise, we use the same hyperparameters as in Tevet et al. (2023). All the training and data generation is done on A100 GPUs.

To compute the FID scores, in accordance with Tevet et al. (2023), we generate one motion for each caption in the HumanML3D test set, resulting in a total of 4,626 generated motions.

The total compute used for this experiment (generating the datasets and training the classifiers) was around 600 GPU-hours.

C. Ablation studies

C.1. Label imbalance

As mentioned in Section 3.1, generating synthetic datasets for training the classifiers without careful planning leads to significant label imbalance issues, which impede the training process. In this section we perform an ablation study to empirically demonstrate its detrimental impact in our experiments.

As a reminder, Gen-neG requires an equal number of positive and negative examples in the synthetic datasets, and it employs importance sampling to address any distribution shift introduced. In the ablated experiment (referred to as "imbalanced" in the results), the ratio of positive to negative examples is model-dependent, being equal to the model's infraction rate. It results in a gradual increase in the dominance of positive examples as the model's infraction rate decreases.

[2] https://github.com/GuyTevet/motion-diffusion-model

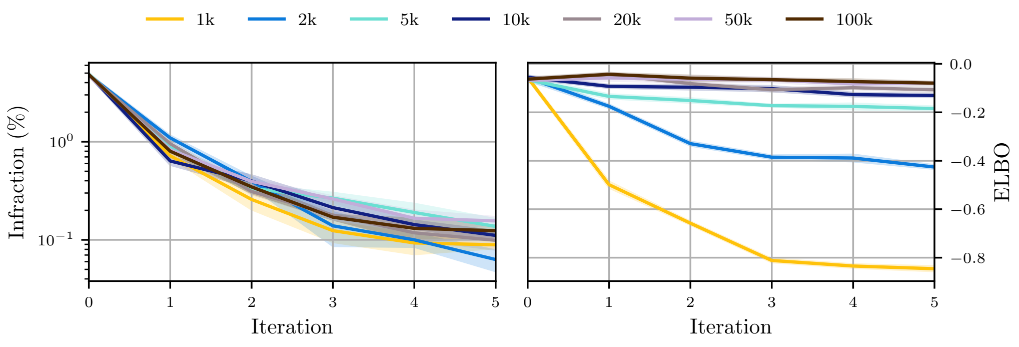


Figure 8. Ablation study for checkerboard experiment on synthetic dataset size

Table 3. Ablation study for the traffic scene generation experiment on synthetic dataset size

Classifier dataset size	Collision (%) $\downarrow$	Offroad (%) $\downarrow$	Infraction (%) $\downarrow$	r-ELBO ($\times 10^{-2}$) $\uparrow$
baseline DM	28.3 ± 0.7	1.3 ± 0.1	29.3 ± 0.6	-27.5 ± 0.01
$8,000$	23.8 ± 0.4	0.9 ± 0.2	24.5 ± 0.5	-28.0 ± 0.01
$40,000$	19.7 ± 0.8	0.8 ± 0.2	20.3 ± 0.9	-27.8 ± 0.01
$80,000$	17.6 ± 0.7	0.7 ± 0.2	18.2 ± 0.6	-27.7 ± 0.01
$400,000$	16.6 ± 0.7	0.8 ± 0.2	17.5 ± 0.7	-27.7 ± 0.01
$800,000$	16.4 ± 0.5	0.9 ± 0.1	17.2 ± 0.4	-27.7 ± 0.01

Checkerboard experiment We repeat our checkerboard experiment with and without Gen-neG's imbalance correction. In each iteration, we create a synthetic dataset of 20,000 samples and train a binary classifier for 10,000 iterations. The other details are the same as the experiment in the main text. We show the performance of these models in Figure 7. We observe that "imbalaced", the ablated version of Gen-neG, is consistently outperformed by Gen-neG in infraction rate. Furthermore, the performance gap between the two methods widens as the models improve. However, it is worth noting that Gen-neG achieves a comparable ELBO to that of "imbalanced" despite these infraction rate differences.

Traffic Scene Generation We conduct a similar ablation as described above, where we train classifiers on a imbalanced datasets of the same size as our method. The results of this ablated experiment are presented in Table 1 in the main text.

C.2. Synthetic dataset size

We conducted an ablation on the number of samples required on the checkerboard and the traffic scene generation tasks. Figure 8 and Table 3 show our results. We observe that the infraction rate constantly decreases irrespective of the dataset size. However, in order to avoid distribution shift we need a large enough dataset, as is evident from the ELBO plot. It is important to emphasize that the classifiers are trained on fully synthetic data generated by the model itself. Therefore, in principle we have access to an unbounded number of samples. As our results show, for the best performance, it is important to ensure the sample size is sufficiently large.

D. Additional results

D.1. Computational cost and sampling latency

Here we discuss the latency of different models considered in this paper. We first report the wall-clock time of generating samples from models. However, in order to deploy any of these models, one should ensure a generated sample is valid. Therefore, all the models should be used together with rejection sampling. We discuss the latency in conjunction with rejection sampling in the second part of this section.

Table 4. Wall-clock time of one denoising step

	Time per diffusion step (s) ($\times 10^{-3}$)		
Method	Checkerboard	Traffic Scenes	Motion Diffusion
Baseline DM / distilled Gen-neG	$0.2 \pm 1 \times 10^{-2}$	$32 \pm 9 \times 10^{-3}$	$83 \pm 2 \times 10^{-1}$
Gen-neG (iteration 1)	$0.5 \pm 2 \times 10^{-2}$	$86 \pm 2 \times 10^{-2}$	$171 \pm 8 \times 10^{-1}$
Gen-neG (iteration 2)	$0.8 \pm 2 \times 10^{-2}$	$138 \pm 3 \times 10^{-2}$	-

Table 5. Sampling time of the traffic scenes experiment (1000 steps)

Method	Parameters (M)	Latency (s)
Baseline DM	6.3	38.58 ± 0.20
Gen-neG (iteration 1)	12.6	86.71 ± 0.19
Gen-neG (iteration 2)	18.9	138.73 ± 0.42
Gen-neG (distilled)	6.3	38.43 ± 0.37
Time-Independent classifier	12.6	126.40 ± 0.20
Imbalanced classifier	12.6	86.88 ± 0.20
w/o IS classifier	12.6	86.81 ± 0.21

Latency We compute the average wall-clock time of the baseline models and iterations of Gen-neG for different experiments and present the results in Table 4. We observe that additional classifiers linearly increase the running time (for conciseness, we have not included further iterations of Gen-neG on checkerboard experiments in Table 4 as its runtime simply continues growing linearly). Moreover, in the Traffic Scenes and checkerboard experiments, the overhead of each classifier is larger than the runtime of the baseline model alone (almost 1.5 times). This is because the architecture of our classifier is the same as the baseline model and for each forward pass of a classifier-guided model, one forward and one backward pass through the classifier is required. However, in the Motion diffusion experiment, since the classifier is not text-conditional, this overhead is relatively smaller. We also report the total time to generate one batch of samples for all the models in Table 1. This table also includes the number of parameters for each model as a proxy for memory consumption. It is important to note that our distilled models have the exact same latency and memory consumption as the baseline diffusion model, since we use the same architecture for the distilled model.

Rejection sampling To ensure that a sample from a model is valid, deploying any of these models requires using them with rejection sampling. Assume a model with infraction rate of ϵ and sampling time of t seconds is given. A rejection sampling loop with this model takes $\frac{1}{1-\epsilon}t$ seconds. However, in a different, more realistic setting, we require an accepted sample in "one iteration" of running the model. Since all models have some infraction rate, we instead generate a batch of samples and require at least one non-infracting sample with high probability. As stated in Section 1, generating at least one non-infracting sample with $(1-\delta)$ probability requires $\frac{\log \delta}{\log \epsilon}$ parallel samples. Therefore, any improvement to ϵ leads to improved computational complexity. In particular, we get the following reductions in the required number of samples:

- 47% on the checkerboard experiment (baseline vs. distillation of 5 iterations of Gen-neG),
- 57% on the traffic scenes experiment (baselines vs. distillation of stacked Gen-neG),
- 9% on the motion diffusion experiment.

We conjecture the relatively smaller improvement in the motion diffusion experiment is because the baseline DM predicts $\mathbf{x}_0$. A follow up to MDM, argues that $\mathbf{x}_0$-prediction models are hard to guide (see appendix A of Zhong et al. (2022)).

D.2. Faster synthetic dataset generation

With the iterative stacking of the classifiers in Gen-neG, as the model becomes better in avoiding invalid samples, it becomes harder to generate a balanced synthetic dataset. To further reduce the computational cost for cases where the infraction rate is very small, one can employ Sequential Importance Sampling (SIS) using a proposal distribution with higher infraction

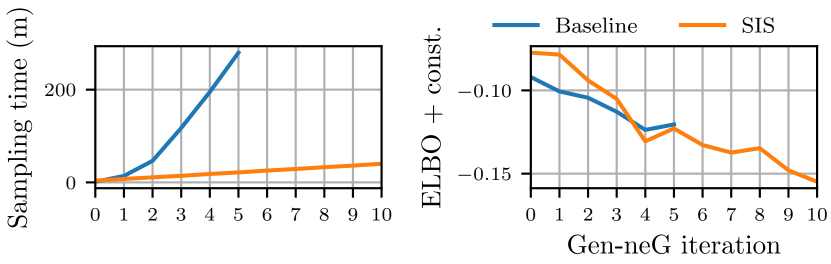


Figure 9. The results of the SIS approach for faster synthetic dataset generation for Gen-neG. SIS makes the sampling time grow linearly with iterations of Gen-neG while maintaining a comparable performance to the standard synthetic dataset sampling approach in Gen-neG.

Table 6. Comparison of a diffusion model and normalizing flow model trained on the INTERACTION dataset. The normalizing flow results are taken from Zwartsenberg et al. (2023).

Method	Collision (%) ↓	Offroad (%) ↓	Infraction (%) ↓
Diffusion model	7.34 ± 0.16	6.94 ± 0.06	13.62 ± 0.13
Normalizing flow (Zwartsenberg et al., 2023)	9.00 ± 1.00	12.00 ± 1.00	20.00 ± 1.00

rate. Concretely, assume we use an SDE solver with pre-defined timesteps $t_0 = 0 < t_1 < \ldots < t_T = 1$ to generate the samples. Therefore the loss function in Equation (9) becomes

$$\mathcal{L}_{\phi}^{\text{cls}} = \mathbb{E}_{p_\theta(\mathbf{x}_0)}\Big[l(\mathbf{x}_0)\Big] = \mathbb{E}_{p_\theta(\mathbf{x}_{t_0},\ldots,\mathbf{x}_{t_T})}\Big[l(\mathbf{x}_0)\Big] = \mathbb{E}_{\pi(\mathbf{x}_{t_0},\ldots,\mathbf{x}_{t_T})}\Big[\frac{p_\theta(\mathbf{x}_{t_0},\ldots,\mathbf{x}_{t_T})}{\pi(\mathbf{x}_{t_0},\ldots,\mathbf{x}_{t_T})} l(\mathbf{x}_0)\Big], \tag{49}$$

where $l(\mathbf{x}_0) = \mathbb{E}_{t,q(\mathbf{x}_t|\mathbf{x}_0)}\left[\mathcal{O}(\mathbf{x}_0)\log C_\phi(\mathbf{x}_t;t) + (1-\mathcal{O}(\mathbf{x}_0))\log(1-C_\phi(\mathbf{x}_t;t))\right]$ and π is a proposal distribution defined as another diffusion model for example, an earlier iteration of Gen-neG. Therefore, $\frac{p(\mathbf{x}_{t_0},\ldots,\mathbf{x}_{t_T})}{\pi(\mathbf{x}_{t_0},\ldots,\mathbf{x}_{t_T})} = \prod_{i=1}^{T} \frac{p(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\pi(\mathbf{x}_{t-1}|\mathbf{x}_t)}$ in which both the numerator and denominator are Gaussian distributions with the same variances but different means.

Figure 9 shows the results of this approach on the checkerboard experiment. It shows that SIS dataset sampling time grows linearly and independent of the infraction rate while maintaining a comparable performance. This linear growth is due to the linear growth in complexity of (non-distilled) Gen-neG models.

D.3. INTERACTION dataset

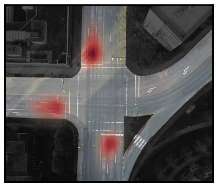
Here we report our results of training a baseline DM model on the INTERACTION dataset (Zhan et al., 2019). Table 6 shows the superior performance of our diffusion model compared to the normalizing flow results from Zwartsenberg et al. (2023). The lower infraction rates compared to Table 1 suggests that the INTERACTION dataset is a simpler dataset compared to the one we used in Section 4.2. One can further improve upon the diffusion model in Table 6 by using it as a baseline DM in Gen-neG.

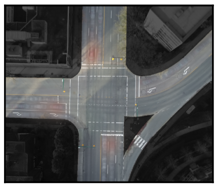
D.4. More visualization results for the traffic scene generation experiment

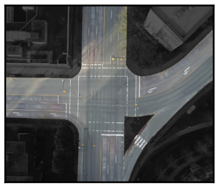
In Figure 10 we report more visualization results from our traffic scene generation experiment. This figure follows from and adds more details to Figure 1.

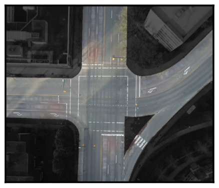
D.5. Overfitting in the checkerboard experiment

Here we present our results regarding the overfitting of the baseline DM in the checkerboard experiment. We run an experiment with 200,000 training iterations, much larger than the 30,000 iterations in the reported results. As we can see in Figure 11, the infraction rate keeps decreasing. However, the model starts overfitting after around 30,000 iterations, as measured by the ELBO on a held-out set. This suggests that the architecture is expressive enough to model sharp jumps in the learned density. However, simply training it on a small dataset without incorporating any prior on "where to allocate its


(a)


(b)


(c)


(d)

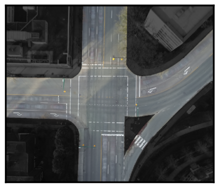

(e)

Figure 10. Complete visualization comparisons for infraction in traffic scene generation experiments. Subplots show infraction per unit area under different models. (a) baseline DM; (b) first iteration of Gen-neG; (c) second iteration of Gen-neG. (d) and (e) are the distillation models corresponding to (b) and (c) respectively. A clear reduction in terms of infractions per unit area can be observed from left to right.

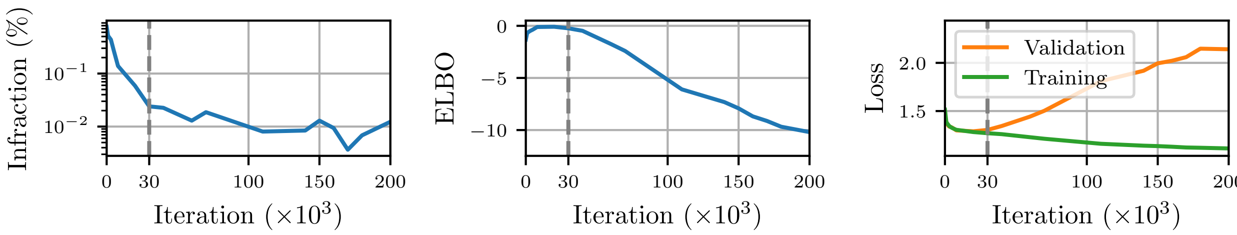


Figure 11. Overfitting results in the checkerboard experiment. The first two plots on the left respectively show the infraction rate and ELBO on a held-out validation set. We observe that training the baseline DM for longer can achieve much lower infraction rates. However, it quickly starts to overfit, leading to poor ELBO estimates on the held-out validation set. The last plot shows the training and validation loss of the model. These plots confirm that the checkpoint we used for baseline DM in Gen-neG at 30,000 iterations does not overfit.

capacity" fails because the model does not receive any signal on where the actual "sharp jump" is. Gen-neG, on the other hand, provides this kind of signal through the oracle-assisted guidance.