QUASI-CONSERVATIVE SCORE-BASED GENERATIVE MODELS

Chen-Hao Chao†, Wei-Fang Sun‡, Bo-Wun Cheng†, and Chun-Yi Lee†∗
† Elsa Lab, Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
‡ NVIDIA AI Technology Center, NVIDIA Corporation

ABSTRACT

Existing Score-based Generative Models (SGMs) can be categorized into constrained SGMs (CSGMs) or unconstrained SGMs (USGMs) according to their parameterization approaches. CSGMs model the probability density functions as Boltzmann distributions, and assign their predictions as the negative gradients of some scalar-valued energy functions. On the other hand, USGMs employ flexible architectures capable of directly estimating scores without the need to explicitly model energy functions. In this paper, we demonstrate that the architectural constraints of CSGMs may limit their score-matching ability. In addition, we show that USGMs’ inability to preserve the property of conservativeness may lead to serious sampling inefficiency and degraded sampling performance in practice. To address the above issues, we propose Quasi-Conservative Score-based Generative Models (QCSGMs) for keeping the advantages of both CSGMs and USGMs. Our theoretical derivations demonstrate that the training objective of QCSGMs can be efficiently integrated into the training processes by leveraging the Hutchinson trace estimator. In addition, our experimental results on the Cifar-10, Cifar-100, ImageNet, and SVHN datasets validate the effectiveness of QCSGMs. Finally, we justify the advantage of QCSGMs using an example of a one-layered autoencoder.

1 INTRODUCTION

Score-based Generative Models (SGMs) are parameterized functions for estimating scores, which are vector fields corresponding to the gradients of log probability density functions. According to their parameterization approaches, SGMs can be categorized into constrained or unconstrained SGMs (Salimans & Ho, 2021).

Constrained SGMs (CSGMs), also known as Energy-Based Models (EBMs), model probability density functions as Boltzmann distributions, and assign their predictions as the negative gradients of some scalar-valued energy functions (Salimans & Ho, 2021). CSGMs are able to ensure the conservativeness of their output vector fields. This property is essential in guaranteeing that the Metropolis-Hasting decisions used in the sampling process are determined based on the probability ratio between two consecutive sampling steps (Salimans & Ho, 2021). This, in turn, is necessary to ensure that the sample distribution converges to the true data distribution. Such a concept has been explored by the researchers of (Salimans & Ho, 2021; Alain & Bengio, 2014; Nguyen et al., 2017; Chen et al., 2014). However, this parameterization approach requires specific model designs, limiting the choices of model architectures for CSGMs. For example, the authors of (Vincent, 2011; Kamyshanska & Memisevic, 2013) proposed to restrict a CSGM to be a one-layered autoencoder with symmetric weights in its linear layer, which hinders its ability to be extended to more sophisticated architectures such as convolution neural networks. On the other hand, the authors of (Salimans & Ho, 2021; Saremi et al., 2018; Song et al., 2019) divided a CSGM into two halves: the first half explicitly parameterizes the negative energy function, while the second half is generated by automatic differentiation tools (Martens et al., 2012) to output the estimated scores. Nevertheless, these methods limit that the output of the first half can only be a scalar, and the second half has to be generated using automatic differentiation tools.

In contrast, unconstrained SGMs (USGMs) employ flexible architectures capable of directly estimating the scores without the need of modeling the energy functions. Due to their architectural...
flexibility, USGMs have been extensively utilized in contemporary machine learning tasks such as image generation (Song & Ermon 2019; Ho et al. 2020; Song & Ermon 2020; Song et al. 2021b; Nichol & Dhariwal, 2021) and audio generation (Lam et al., 2022; Kong et al., 2021; Chen et al. 2021). Among these works, the authors in (Song et al. 2021b) proposed a unified framework based on a USGM, which achieved remarkable performance on several benchmarks. Their success demonstrated that architectural flexibility can be beneficial for SGMs. However, in spite of their empirical benefit, our analyses in Section 3 indicate that USGMs’ inability to ensure conservativeness may lead to degraded sampling performance.

To preserve both the conservativeness of CSGMs and the architectural flexibility of USGMs, we propose Quasi-Conservative Score-based Generative Models (QCSGMs). Instead of constraining the model architecture, QCSGMs resort to enhancing the conservativeness of USGMs through minimizing a regularization loss. Our theoretical derivations demonstrate that such a regularization term can be integrated into the training processes of SGMs efficiently through the Hutchinson trace estimator (Hutchinson, 1989). Moreover, our experimental results showcase that the performance of Noise Conditional Score Network++ (NCSN++) (Song et al., 2021b), which is considered the state-of-the-art USGM, can be further improved by incorporating our regularization method on the Cifar-10, Cifar-100, ImageNet-32x32, and SVHN datasets.

2 BACKGROUND AND RELATED WORKS

In this section, we walk through the background material and the related works for understanding the contents of this paper. We first introduce a number of score matching methods for training an SGM. Next, we describe the sampling algorithms for generating samples through an SGM. Lastly, we elaborate on the conservative property of SGMs, and the differences between CSGMs and USGMs.

2.1 Score Matching Methods

Given a true data distribution \( p_{\text{data}} \), its empirical distribution \( p_0(x) \) is established through sampling \( M \) independent and identically distributed \( D \)-dimensional vectors \( \{x^{(i)} : x^{(i)} \in \mathbb{R}^D \}_{i=1}^M \), represented as a Dirac delta distribution, i.e., \( p_0(x) \triangleq \frac{1}{M} \sum_{i=1}^{M} \delta(\|x - x^{(i)}\|) \). To ensure the probability density function (pdf) is everywhere non-zero and differentiable, previous literature (Vincent, 2011) proposed a closed form \( \frac{1}{\sqrt{2\pi} \sigma^D} e^{-\frac{1}{2\sigma^2} \|\tilde{x} - x\|^2} \) is an isotropic Gaussian smoothing kernel with a standard deviation \( \sigma \). When \( \sigma > 0 \), the score function of \( p_\sigma(\tilde{x}) \) has a closed form (Chao et al., 2022), which can be formulated as:

\[
\frac{\partial \log p_\sigma(\tilde{x})}{\partial \tilde{x}} = \sum_{i=1}^{M} \frac{1}{\sigma}(x^{(i)} - \tilde{x})p_\sigma(\tilde{x}|x^{(i)}) \sum_{i=1}^{M} p_\sigma(\tilde{x}|x^{(i)}). \tag{1}
\]

Score matching (Hyvärinen, 2005) describes the learning process to approximate the score function \( \frac{\partial}{\partial x} \log p_\sigma(\tilde{x}) \) in Eq. (1) using a neural network \( s(\cdot; \theta) : \mathbb{R}^D \rightarrow \mathbb{R}^D \), which is parameterized by \( \theta \) and is trained through minimizing the Explicit Score Matching (ESM) objective expressed as follows:

\[
\mathcal{L}_{\text{ESM}}(\theta) = \mathbb{E}_{p_\sigma(\tilde{x})} \left[ \frac{1}{2} \left\| s(\tilde{x}; \theta) - \frac{\partial \log p_\sigma(\tilde{x})}{\partial \tilde{x}} \right\|^2 \right]. \tag{2}
\]

Eq. (2) involves the explicit calculation of Eq. (1), which suffers from serious training inefficiency when the dataset size \( M \) is large. To address this issue, an alternative method called Implicit Score Matching (ISM) (Hyvärinen, 2005), which excludes \( \frac{\partial}{\partial x} \log p_\sigma(\tilde{x}) \) in the training objective, was introduced to efficiently train \( s(\tilde{x}; \theta) \). ISM employs an equivalent loss \( \mathcal{L}_{\text{ISM}} \) expressed as follows:

\[
\mathcal{L}_{\text{ISM}}(\theta) = \mathbb{E}_{p_\sigma(\tilde{x})} \left[ \frac{1}{2} \left\| s(\tilde{x}; \theta) \right\|^2 + \text{tr} \left( \frac{\partial s(\tilde{x}; \theta)}{\partial \tilde{x}} \right) \right], \tag{3}
\]

where \( \frac{\partial s(\tilde{x}; \theta)}{\partial \tilde{x}} \) corresponds to the Jacobian matrix of \( s(\tilde{x}; \theta) \), and \( \text{tr} (\cdot) \) denotes the trace of a matrix. Although \( \mathcal{L}_{\text{ISM}} \) avoids the calculation of Eq. (1), the calculation of \( \text{tr} \left( \frac{\partial s(\tilde{x}; \theta)}{\partial \tilde{x}} \right) \) in Eq. (3) still requires \( D \) times of backpropagations (Song et al., 2019), which hinders \( \mathcal{L}_{\text{ISM}} \)'s ability of being utilized in high-dimensional context. To alleviate it, a scalable objective, called Sliced Score Matching (SSM), was proposed...
Matching (SSM) (Song et al., 2019) loss, was proposed to approximate $\text{tr} \left( \frac{\partial}{\partial x} s(\tilde{x}; \theta) \right)$ in $L_{\text{SSM}}$ with the Hutchinson trace estimator (Hutchinson, 1989). Given a random vector $v$ drawn from a distribution $p(v)$ satisfying $E_{p(v)}[v v^T] = I$, the Hutchinson trace estimator replaces the trace of a square matrix $A$ with $E_{p(v)}[v v^T A v]$, which can be derived as:

$$\text{tr} (A) = \text{tr} (A I) = \text{tr} (AE_{p(v)}[v v^T]) = E_{p(v)}[\text{tr} (A v v^T)] = E_{p(v)}[\text{tr} (v^T A v)].$$

(4)

The above derivation suggests that $tr \left( \frac{\partial}{\partial x} s(\tilde{x}; \theta) \right)$ in Eq. (3) can be substituted with $E_{p(v)}[v^T \frac{\partial}{\partial x} s(\tilde{x}; \theta) v]$, resulting in an equivalent objective $L_{\text{SSM}}$ expressed as follows:

$$L_{\text{SSM}}(\theta) = E_{p_v(x)} \left[ \frac{1}{2} \left\| s(\tilde{x}; \theta) \right\|^2 + E_{p_v(v)} \left[ v^T \frac{\partial}{\partial \tilde{x}} s(\tilde{x}; \theta) \right] \right].$$

(5)

The vector-Jacobian product $v^T \frac{\partial}{\partial x} s(\tilde{x}; \theta)$ can be calculated with a single backward propagation using automatic differentiation (Martins et al., 2012), and the expectation can be approximated using $K$ independently sampled vectors $\{v(i)\}_{i=1}^K$. Therefore, the computation of $E_{p_v(v)} [v^T \frac{\partial}{\partial x} s(\tilde{x}; \theta) v]$ in Eq. (5) can be less expensive than $\text{tr} \left( \frac{\partial}{\partial x} s(\tilde{x}; \theta) \right)$ in Eq. (3) when $K \ll D$. The Denoising Score Matching (DSM) (Vincent, 2011) loss is another scalable objective formulated based on the Parzen density estimator, which further prevents the computational overhead incurred by the backward propagation in $L_{\text{SSM}}$:

$$L_{\text{DSM}}(\theta) = E_{p_v(x|\tilde{x})} p_v(\tilde{x}) \left[ \frac{1}{2} \left\| s(\tilde{x}; \theta) - \frac{\partial \log p_\sigma(\tilde{x}|x)}{\partial \tilde{x}} \right\|^2 \right],$$

(6)

where $\frac{\partial \log p_\sigma(\tilde{x}|x)}{\partial \tilde{x}} = \frac{1}{\sqrt{2 \pi}}(x - \tilde{x})$. Since the computational cost of $L_{\text{DSM}}$ is relatively low in comparison to the other score matching losses, it has been widely adopted in contemporary modeling methods (Song & Ermon, 2019, 2020; Song et al., 2021b) that pursue training efficiency.

### 2.2 Sampling Process

Given an optimal SGM $s(\tilde{x}; \theta) = \frac{\partial}{\partial x} \log p_\sigma(\tilde{x})$, $\forall \tilde{x} \in \mathbb{R}^D$ that minimizes the score-matching objectives (i.e., Eqs. (2), (3), (5), and (6)), Langevin dynamics (Roberts & Tweedie, 1996; Roberts & Rosenthal, 1998) enables $p_\sigma(\tilde{x})$ to be iteratively approximated through the following equation:

$$\tilde{x}_{t+1} = \tilde{x}_t + \alpha s(\tilde{x}_t; \theta) + \sqrt{2\alpha} z_t,$$

(7)

where $\alpha$ is the step size, $t$ is the timestep, $z_t \in \mathbb{R}^D$ is a noise vector sampled from a normal distribution $\mathcal{N}(0, I)$. Under the condition where $\alpha \to 0$ and $T \to \infty$, $\tilde{x}_T$ can be generated as if it is directly sampled from $p_\sigma(\tilde{x})$ (Roberts & Rosenthal, 1998; Welling & Teh, 2011). In practice, however, $\alpha > 0$ and $T < \infty$, which violate the convergence guarantee. Fortunately, the convergence guarantee can be recovered by the Metropolis-Hasting decisions (Roberts & Tweedie, 1996; Roberts & Rosenthal, 1998), which adjust the updates in Eq. (7) by the acceptance ratio $p_{\sigma}(\tilde{x}_{t+1}) / p_{\sigma}(\tilde{x}_t)$. This can be estimated using the discretized line integral through $s(\cdot; \theta)$ (Alain & Bengio, 2014) as:

$$\frac{p_{\sigma}(\tilde{x}_{t+1})}{p_{\sigma}(\tilde{x}_t)} = \exp \left( - \int_0^1 s(\gamma(t); \theta) \cdot \gamma'(t) dt \right) \approx \exp \left( - \frac{1}{N} \sum_{i=1}^{N} s(\tilde{x}_i(\gamma(t)); \theta) \cdot (\tilde{x}_{t+1} - \tilde{x}_t) \right),$$

(8)

where $\gamma(\cdot) : [0, 1] \to \mathbb{R}^D$ is a smooth curve, $N$ is the number of discretized points, $\{\tilde{x}_i(\gamma(t))\}_{i=1}^{N}$ is a set of equally-spaced discretized points with $\tilde{x}_i(1) = \tilde{x}_t$ and $\tilde{x}_i(N) = \tilde{x}_{t+1}$. The Metropolis-Hastings acceptance decisions are determined when the value of $\min \{1, p_{\sigma}(\tilde{x}_{t+1}) / p_{\sigma}(\tilde{x}_t)\}$ is greater than a random noise sampled from a uniform distribution. Otherwise, the update is rejected.

The sampling process that involves the iterative updates in Eq. (7) and the adjustment in Eq. (8) is called Metropolis-Adjusted Langevin Algorithm (MALA). This method was adopted in traditional score-based generative modeling methods (Alain & Bengio, 2014; Nguyen et al., 2017), but becomes less popular in contemporary deep neural network (DNN) based methods due to the computational overhead of Eq. (8). In fact, recent studies (Song & Ermon, 2019, 2020; Song et al., 2021b) followed a different route and proposed to extend the Langevin dynamics to time-inhomogeneous variants by making $s(\cdot; \theta)$ and $\alpha$ dependent on $t$. They demonstrated that such time-inhomogeneous variants can achieve superior results on several benchmarks within limited timesteps without the assistance of the Metropolis-Hasting adjustment. In Appendix A.4 we elaborate on the implementation details of such a time-inhomogeneous sampling process.
2.3 CONSERVATIVENESS AND ROTATION DENSITY OF A SCORE-BASED GENERATIVE MODEL

A vector field is said to be conservative if it can be written as the gradient of a scalar function [Im et al. 2016]. As proved in [Im et al. 2016], the output vector field of an SGM \( s(\tilde{x}; \theta) \) is said to be conservative over a smooth and simply-connected domain \( \mathbb{S} \subseteq \mathbb{R}^D \) if and only if its Jacobian is symmetry for all \( \tilde{x} \in \mathbb{S} \), which can be equivalently expressed as the zero-rotation-density (ROT) (Glotzli & Richters 2020) condition expressed as follows:

\[
\text{ROT}_{ij} s(\tilde{x}; \theta) = \frac{\partial s(\tilde{x}; \theta)_i}{\partial x_j} - \frac{\partial s(\tilde{x}; \theta)_j}{\partial x_i} = 0, \quad 1 \leq i, j \leq D, \tag{9}
\]

where \( \frac{\partial}{\partial x_j} s(\tilde{x}; \theta)_i \) corresponds to the gradient of the \( i \)-th element of \( s(\tilde{x}; \theta) \) with respect to the \( j \)-th element of \( \tilde{x} \). \( \text{ROT}_{ij} s(\tilde{x}; \theta) \) in Eq. (9) describes the infinitesimal circulation of \( s(\tilde{x}; \theta) \) around \( \tilde{x} \).

For CSGMs, \( p_\sigma(\tilde{x}) \) is modeled as a Boltzmann distribution \( p(\tilde{x}; \theta) = \exp \left(- E(\tilde{x}; \theta) \right) / Z(\theta) \), where \( \exp (\cdot) \) indicates the exponential function, \( E(\cdot ; \theta) : \mathbb{R}^D \to \mathbb{R} \) represents a scalar-valued energy function, and \( Z(\theta) \) refers to the partition function. Therefore, the output vector field of a CSGM can be represented as \( s(\tilde{x}; \theta) = \frac{\partial}{\partial x} \log p(\tilde{x}; \theta) = - \frac{\partial}{\partial x} E(\tilde{x}; \theta) \). This implies that \( s(\tilde{x}; \theta) \) is conservative. In other words, \( s(\tilde{x}; \theta) \) satisfies the zero-rotation-density condition in Eq. (9), since the mixed second derivatives of \( E(\tilde{x}; \theta) \) are equivalent (Alain & Bengio 2014), which can be shown as the following:

\[
\text{ROT}_{ij} s(\tilde{x}; \theta) = \frac{\partial^2 E(\tilde{x}; \theta)}{\partial x_i \partial x_j} - \frac{\partial^2 E(\tilde{x}; \theta)}{\partial x_j \partial x_i} = 0, \quad 1 \leq i, j \leq D. \tag{10}
\]

On the other hand, since USGMs do not follow the aforementioned modeling procedure, the conservativeness of USGMs is not guaranteed. Although it is possible to ensure the conservativeness of an USGM under an ideal scenario that \( s(\tilde{x}; \theta) \) perfectly models \( \frac{\partial}{\partial x} \log p_\sigma(\tilde{x}) \) for all \( \tilde{x} \in \mathbb{R}^D \), a trained USGM typically contains approximation errors in practice. This suggests that USGMs are non-conservative in most cases, and do not satisfy the zero-rotation-density condition.

3 MOTIVATIONAL EXAMPLES

In this section, we demonstrate the importance of preserving the conservativeness as well as the architectural flexibility of SGMs. In addition, we provide the motivation behind the adoption of QCSGMs through two motivational experiments.

3.1 THE INFLUENCES OF NON-CONSERVATIVENESS ON SAMPLING EFFICIENCY

The sampling processes described in Section 2.2 are formulated under an ideal scenario that \( s(\tilde{x}; \theta) = \frac{\partial}{\partial x} \log p_\sigma(\tilde{x}) \), \( \forall \tilde{x} \in \mathbb{R}^D \). In practice, however, a trained USGM contains approximation errors, which could lead to its failure in preserving its conservativeness, as stated in Section 2.3. In the following example, we show that the non-conservativeness of a USGM may cause serious sampling inefficiency even in the presence of slight approximation errors. Given a target distribution \( p_\sigma(\tilde{x}) = \mathcal{N}(0; \sigma^2 I) \) and a small constant \( \epsilon > 0 \), a USGM \( s_\epsilon(\tilde{x}) \) with its approximation errors less than \( \epsilon \), i.e., \( L_{\text{ESM}} < \epsilon \), can be expressed as the following non-conservative vector field:

\[
s_\epsilon(\tilde{x}) = \frac{1}{\sigma^2} \begin{bmatrix} -x_1 \\ -x_2 \\ \frac{2 \epsilon \mu(\tilde{x})}{p_\sigma(\tilde{x})(\|\tilde{x}\|^2 + \epsilon)} \end{bmatrix} \begin{bmatrix} -x_2 \\ x_1 \\ \|\tilde{x}\|^2 + \epsilon \end{bmatrix}, \tag{11}
\]

where \( \mu(\tilde{x}) = \frac{1}{10} \sum_{i=1}^{10} N([2 \cos(\frac{2 \pi i}{10}), 2 \sin(\frac{2 \pi i}{10})]^T; I) \) is a mixture of isotropic Gaussians, and \( \tilde{x} = [x_1, x_2]^T \) is a two-dimensional vector. The derivation of Eq. (11) and the rationale behind the formulation of \( s_\epsilon(\tilde{x}) \) are provided in Appendix A.2.1. To quantitatively evaluate the non-conservativeness of \( s_\epsilon(\tilde{x}) \), we measure the magnitude of \( \text{ROT}_{ij} s_\epsilon(\tilde{x}) \) using two metrics: the asymmetry metric \( \text{Asym} \in [0, \infty) \) and its normalized variant \( \text{NAsym} \in [0, 1] \), defined as follows:

\[
\text{Asym} = \mathbb{E}_{p_\sigma(\tilde{x})} \left[ \frac{1}{2} \sum_{i,j=1}^{D} \left( \text{ROT}_{ij} s_\epsilon(\tilde{x}) \right)^2 \right] = \mathbb{E}_{p_\sigma(\tilde{x})} \left[ \frac{1}{2} \left\| J - J^T \right\|_F^2 \right], \tag{12}
\]
Recall
Precision
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0
0 20 40 60 80 100
0 200 400 600 800 1,000

Asym = 0.00
NAsym = 7.50e-3
= 2.00e-2
= 1.50e-2
= 2.25e-2

Fig. 1 (a) depicts the trends of precision and recall during the sampling processes under different choices of $\epsilon$. The table on the right-hand side reports the results measured using the Asym and NAsym metrics. (b) The visualized trajectories of Langevin dynamics under different choices of $\epsilon$.

$$NAsym = \mathbb{E}_{p_x(\tilde{x})} \left[ \frac{1}{4} \| J - J^T \|^2_F \right],$$  (13)

where $J = \frac{\partial}{\partial \tilde{x}_i} s_x(\tilde{x})$, $\| \cdot \|_F$ is the Frobenius norm. The details of NAsym are elaborated in Appendix A.3. We leverage the precision (P) and recall (R) metrics (Kynkänniemi et al., 2019) to examine the sampling performance.

Fig. 1(a) depicts the trends of precision and recall during the sampling processes under different choices of $\epsilon$. It is observed that the number of steps required for achieving the same sampling performance increases rapidly as the values of Asym and NAsym grow. This suggests that the non-conservativeness of $s_x(\tilde{x})$ may decelerate the sampling processes even when $L_{ESM}$ is bounded by $\epsilon$. In addition, the visualized trajectories shown in Fig. 1(b) further reveal that the existence of approximation errors in $s_x(\tilde{x})$ may incur rotational vector fields tangent to the true score, leading to inefficient updates during the sampling processes. The above experimental evidences thus demonstrate that the non-conservativeness of a USGM may potentially cause negative impacts on the sampling efficiency. In other words, the non-conservativeness of a USGM may lead to degraded sampling performance within limited timesteps.

3.2 The Impacts of Architectural Flexibility on Score-Matching Accuracy and Sampling Performance

In this experiment, we inspect the influence of architectural flexibility on the training and sampling processes. For a fair evaluation, a USGM $s_U$ and a CSGM $s_C$ are implemented as neural networks consisting of the same number of parameters. Following the approach described in (Salimans & Ho, 2021), these two models are represented as follows:

$$s_U(\tilde{x}; \theta_U) = f(\tilde{x}; \theta_U), \quad s_C(\tilde{x}; \theta_C) = -\frac{1}{2\sigma} \frac{\partial}{\partial \tilde{x}} f(\tilde{x}; \theta_C) - \frac{\partial}{\partial \tilde{x}} \log p_x(\tilde{x}),$$  (14)

where $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$ is a neural network, and $\theta_U$ and $\theta_C$ are the parameters. We then compare the conservativeness, the score-matching ability, and the sampling performance of both $s_U$ and $s_C$, which are trained independently on three two-dimensional datasets. The conservativeness is measured using Asym and NAsym. The sampling performance is measured using the precision and recall metrics, where the number of the sampling timesteps is fixed at 1,000. Finally, the score-matching ability is evaluated using the score-matching error $\mathbb{E}_{s(x)(\theta)} \| s(x; \theta) - \frac{\partial}{\partial x} \log p_x(x) \|$, where $s$ can be either $s_C$ or $s_U$, and $\frac{\partial}{\partial x} \log p_x(x)$ is calculated based on Eq. (1). In this experiment, $f(\cdot; \theta_U)$ and $f(\cdot; \theta_C)$ are modeled using three-layered multilayer perceptrons (MLPs), which are the same as those used.
In this section, we introduce QCSGMs and present an efficient implementation of them. In Section 4.1, we describe the learning objective of QCSGMs, and derive its scalable variant. In Section 4.2, we elaborate on the formulation and our implementation of QCSGMs.

### 4.1 Quasi-Conservative Score-based Generative Models

Instead of following the concept of CSGMs to ensure the conservativeness through architecture constraints, QCSGMs resort to penalizing the non-conservativeness through a regularization loss. The training objective for QCSGMs is defined as $\mathcal{L}_{\text{Total}}$, which is expressed as the following equation:

$$
\mathcal{L}_{\text{Total}}(\theta) = \mathcal{L}_{\text{SM}}(\theta) + \lambda \mathcal{L}_{\text{QC}}(\theta),
$$

where $\mathcal{L}_{\text{SM}}$ can be any one of the score-matching objectives (i.e., Eqs. (2), (5), (5), or (6)), $\mathcal{L}_{\text{QC}}$ represents the regularization term reflecting the non-conservativeness, and $\lambda$ is a balancing factor. As discussed in Section 4.1, the non-conservativeness of an SG can be measured using the magnitude of its rotation densities in the Frobenius norm (i.e., Eq. (12)), suggesting a formulation of $\mathcal{L}_{\text{QC}}$ as:

$$
\mathcal{L}_{\text{QC}}(\theta) = \mathbb{E}_{p_r(\tilde{x})} \left[ \frac{1}{2} \left\| J - J_T \right\|_F^2 \right],
$$

where $\mathbb{E}_{p_r(\tilde{x})}$ denotes the expectation over the sampling distribution $p_r(\tilde{x})$. This formulation allows QCSGMs to adaptively penalize the non-conservativeness and improve the sampling performance.

### Table 1: The evaluation results of CSGMs, USGMs, and QCSGMs in terms of their means and variances of five independent runs on the '8-Gaussian,' 'Spirals,' and 'Checkerboard' datasets, which are detailed in Appendix A.5.1. L denotes ‘Langevin dynamics,’ and M represents ‘MALA.’ The arrow symbols $\uparrow / \downarrow$ indicate that higher / lower values correspond to better performance, respectively.

| Dataset | Model | $\text{Asym} (\|)$ | $\text{Maxym} (\|)$ | Score Error (\%) | Precision (L) (\%) | Precision (M) (\%) | Recall (L) (\%) | Recall (M) (\%) |
|---------|-------|----------------------|----------------------|-------------------|-------------------|-------------------|-----------------|------------------|
| 8-Gaussian | CSGM  | 1.79 ± 00. e-8  | 7.51 ± 00. e-10 | 4.62 ± 00. e-1  | 9.729 ± 00. e-1 | 9.795 ± 00. e-1 | 9.879 ± 00. e-1 | 9.975 ± 00. e-1 |
|          | USGM  | 2.95 ± 00. e-2  | 3.70 ± 00. e-3  | 4.099 ± 00. e-1 | 9.696 ± 00. e-1 | 9.922 ± 00. e-1 | 9.794 ± 00. e-1 | 9.880 ± 00. e-1 |
|          | QCSGM | 1.30 ± 00. e-1  | 1.72 ± 00. e-4  | 3.233 ± 00. e-1 | 9.805 ± 00. e-1 | 9.982 ± 00. e-1 | 9.982 ± 00. e-1 | 9.982 ± 00. e-1 |
| Spirals  | CSGM  | 1.17 ± 00. e-7  | 9.71 ± 00. e-10 | 4.976 ± 00. e-1 | 4.487 ± 00. e-1 | 5.740 ± 00. e-1 | 9.992 ± 00. e-1 | 9.857 ± 00. e-1 |
|          | USGM  | 1.99 ± 00. e-1  | 1.02 ± 00. e-2  | 4.907 ± 00. e-1 | 4.487 ± 00. e-1 | 5.797 ± 00. e-1 | 9.992 ± 00. e-1 | 9.906 ± 00. e-1 |
|          | QCSGM | 6.67 ± 00. e-3  | 3.76 ± 00. e-4  | 5.874 ± 00. e-1 | 4.503 ± 00. e-1 | 5.829 ± 00. e-1 | 9.992 ± 00. e-1 | 9.924 ± 00. e-1 |
| Checkerboard | CSGM | 2.58 ± 00. e-8  | 1.67 ± 00. e-9  | 3.866 ± 00. e-1 | 8.375 ± 00. e-1 | 9.382 ± 00. e-1 | 9.988 ± 00. e-1 | 9.985 ± 00. e-1 |
|          | USGM  | 1.27 ± 00. e-1  | 1.93 ± 00. e-2  | 3.804 ± 00. e-1 | 8.411 ± 00. e-1 | 9.393 ± 00. e-1 | 9.988 ± 00. e-1 | 9.981 ± 00. e-1 |
|          | QCSGM | 2.73 ± 00. e-3  | 6.69 ± 00. e-4  | 2.753 ± 00. e-1 | 8.423 ± 00. e-1 | 9.401 ± 00. e-1 | 9.988 ± 00. e-1 | 9.980 ± 00. e-1 |

Table 1 reports the results of the above setting. The column ‘Score Error’ in Table 1 demonstrates that the USGMs consistently deliver lower score-matching errors in comparison to the CSGMs, suggesting that their architectural flexibility is potentially beneficial to their score-matching. On the other hand, due to the impact of their non-conservativeness, USGMs are unable to consistently achieve superior results on the precision and recall metrics, as shown in the last four columns of Table 1. The above observations thus indicate that the architectural flexibility of a USGM may enhance its score-matching ability. Nevertheless, its non-conservativeness may deteriorate its sampling performance.

The experimental insights in Sections 3.1 and 3.2 shed light on two essential issues to be further explored and addressed. First, although USGMs benefit from their architectural flexibility, their non-conservativeness may lead to serious sampling inefficiency and degraded sampling performance in practice. Second, despite that CSGMs are conservative, their architectural requirement may limit their score-matching ability. Based on the above observations, this paper intends to investigate a new type of SGMs, called Quasi-Conservative Score-based Generative Models (QCSGMs), which are developed to maintain both the conservativeness as well as the architectural flexibility. As revealed in Table 1, QCSGMs are able to achieve improved results in terms of score-matching accuracy and sampling performance in comparison to CSGMs and USGMs under most of the experimental setups. In the next section, we elaborate on the formulation and implementation of QCSGMs.
where $J = \frac{\partial}{\partial x} s(\tilde{x}; \theta)$. This objective function, however, requires $D$ times of backpropagations to explicitly calculate the Jacobian matrix of $s(\tilde{x}; \theta)$. In order to reduce the computational cost, we first formulate an equivalent objective $L_{QC}^{tr}$, and then utilize the Hutchinson trace estimator to approximate $L_{QC}^{tr}$. The loss of $L_{QC}^{tr}$ is derived in Appendix A.2.2 and is formulated as follows:

$$L_{QC}^{tr}(\theta) = \mathbb{E}_{p(x)} \left[ \text{tr} \left( J J^T \right) - \text{tr} \left( J J \right) \right].$$

(17)

By applying the Hutchinson trace estimator to both $\text{tr} \left( J J^T \right)$ and $\text{tr} \left( J J \right)$ according to Eq. (4), $L_{QC}^{tr}$ can be equivalently replaced by an another objective $L_{QC}^{est}$, which is expressed as the following:

$$L_{QC}^{est}(\theta) = \mathbb{E}_{p(x)} \left[ \mathbb{E}_{p(v)} \left[ v^T J J^T v \right] - \mathbb{E}_{p(v)} \left[ v^T J J v \right] \right].$$

(18)

Eq. (18) suggests that $\mathbb{E}_{p(v)} \left[ v^T J J^T v - v^T J J v \right]$ can be approximated using $K$ random vectors $\{v^{(i)}\}_{i=1}^K$ independently sampled from $p(v)$. Additionally, the computational graph of $v^T J J^T v - v^T J J v$ can be efficiently constructed without increasing the asymptotic time complexity with respect to $D$, which is later explained in Section 4.2. As a result, under such an implementation, the computational cost of $L_{QC}^{est}$ is significantly lower than $L_{QC}$ and $L_{QC}^{tr}$ when $K \ll D$.

4.2 THE TRAINING PROCEDURE OF QCSGMs

In this subsection, we walk through the proposed training procedure of QCSGMs. This procedure is detailed in Algorithm 1 and the corresponding computational graph is illustrated in Fig. 2. The entire training procedure is divided into five steps, denoted as Steps (1)–(5), respectively. Steps (1)–(3) describe the forward propagation process of $L_{Total}(\theta)$, which is depicted by the black arrows in Fig. 2. Steps (4) and (5) correspond to the backpropagation processes of the two gradient components comprising $\frac{\partial}{\partial \theta} L_{Total}(\theta)$, which are named the primary and secondary components, and are depicted as the blue and red arrows in Fig. 2, respectively. The detailed formulations for these two components and the rationale behind such a two-step backpropagation process are further elaborated in Appendix A.2.3. The details of Steps (1)–(5) are explained as follows.

(1) Computing $v^T J J^T v$. First, $v^T J$ is computed by performing backpropagation of $v^T s(\tilde{x}; \theta)$ with respect to $\tilde{x}$ via automatic differentiation, which is depicted as the upper ‘Auto. Diff.’ block in Fig. 2. Then, $v^T J J^T v$ is calculated by taking the squared L2 norm on $v^T J$ according to the relationship: $\|v^T J\|^2 = v^T J (v^T J)^T = v^T J J^T v$.

(2) Computing $v^T J J v$. First, $\text{sg} \left[ v^T J \right] s(\tilde{x}; \theta)$ is calculated by taking the inner product between $\text{sg} \left[ v^T J \right]$ and $s(\tilde{x}; \theta)$, where the stop-gradient operator $\text{sg} \left[ \cdot \right]$ is applied to $v^T J$ to detach it from the computational graph built in Step (1). Then, $\text{sg} \left[ v^T J \right] J$ is calculated by differentiating $\text{sg} \left[ v^T J \right] s(\tilde{x}; \theta)$ via performing backpropagation. Stopping the gradient of $v^T J$ is necessary to ensure that the automatic differentiation (i.e., the lower ‘Auto. Diff.’) block in Fig. 2 excludes the computational graph used for differentiating $v^T J$, allowing $v^T J$ to be correctly derived. Lastly, $\text{sg} \left[ v^T J \right] J v$ is obtained by taking the inner product of $\text{sg} \left[ v^T J \right] J$ and $v$. 

Figure 2: The computational graph of $L_{Total}$ in QCSGMs. The ‘Auto. Diff.’ blocks represent the operation of differentiating $u^T s(\tilde{x}; \theta)$, where $u$ is a constant vector with respect to $\tilde{x}$. 

| Operation | Forward propagation | (1) Computing $u^T J J^T u$ | (2) Computing $\text{sg}[u^T J] s(\tilde{x}; \theta)$ | (3) Computing $\text{sg}[u^T J] J$ | (4) Deriving the primary component of $\frac{\partial}{\partial \theta} L_{Total}(\theta)$ | (5) Deriving the secondary component of $\frac{\partial}{\partial \theta} L_{Total}(\theta)$ |
|-----------|---------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|

- $\mathbb{E}_{p(v)}$ denotes the expectation operator with respect to $p(v)$.
- $\text{sg} \left[ \cdot \right]$ denotes the stop-gradient operator.
- $\text{tr} \left( A \right)$ denotes the trace of matrix $A$.
- $\| \cdot \|^2$ denotes the squared L2 norm.
- $\| \cdot \|$ denotes the L2 norm.
- $\mathbb{E}_{p(x)}$ denotes the expectation operator with respect to $p(x)$.
- $\mathbb{E}_{p(v)} \left[ \cdot \right]$ denotes the expectation operator with respect to $p(v)$.
- $\text{sg} \left[ \cdot \right]$ denotes the stop-gradient operator.
- $\text{tr} \left( A \right)$ denotes the trace of matrix $A$.
- $\| \cdot \|^2$ denotes the squared L2 norm.
- $\| \cdot \|$ denotes the L2 norm.
(3) Computing \( \mathcal{L}_{\text{Total}}(\theta) \). Based on the results of Steps (1) and (2), \( \mathcal{L}_{\text{Total}}^{\text{QC}}(\theta) \) is computed by taking the expectation of \( (v^T J^T v - \text{sg}[v^T J] v) \). Meanwhile, the score matching loss \( \mathcal{L}_{\text{SM}}(\theta) \) can be derived using any one of the Eqs. (2), (3), (5), and (6). Finally, \( \mathcal{L}_{\text{Total}}(\theta) \) is calculated by adding \( \mathcal{L}_{\text{SM}}(\theta) \) and \( \lambda \mathcal{L}_{\text{QC}}^{\text{est}}(\theta) \), as described in Eq. (15).

(4) Deriving the primary component of \( \frac{\partial}{\partial \theta} \mathcal{L}_{\text{Total}}(\theta) \). Based on the computational graph built in Steps (1)\textendash}(3), the primary component of \( \frac{\partial}{\partial \theta} \mathcal{L}_{\text{Total}}(\theta) \) is computed by performing backward propagation through the paths in the computational graph highlighted by the blue arrows in Fig. 2 using automatic differentiation. Note that these gradients are not equal to \( \frac{\partial}{\partial \theta} \mathcal{L}_{\text{Total}}(\theta) \) due to the adoption of the stop-gradient operator \( \text{sg} [\cdot] \) in Step (2). As a result, an additional secondary gradient component, which is derived in Step (5), is included to compensate it.

(5) Deriving the secondary component of \( \frac{\partial}{\partial \theta} \mathcal{L}_{\text{Total}}(\theta) \). The secondary component of \( \frac{\partial}{\partial \theta} \mathcal{L}_{\text{Total}}(\theta) \) is derived by performing backward propagation through the paths in the computational graph highlighted by the red arrows in Fig. 2 using automatic differentiation. By accumulating the gradients of the primary and the secondary components, the gradients \( \frac{\partial}{\partial \theta} \mathcal{L}_{\text{Total}}(\theta) \) can be correctly calculated.

Based on the above implementation, the computation of \( \mathcal{L}_{\text{est}}^{\text{QC}} \) does not require \( D \) times of backpropagation, justifying the computational efficiency of \( \mathcal{L}_{\text{est}}^{\text{QC}} \) over \( \mathcal{L}_{\text{QC}} \) and \( \mathcal{L}_{\text{est}}^{\text{SM}} \). We summarize this section with the time complexity of different training objectives discussed in this paper in Table 2. In this table, \( H \) denotes the dimension of the largest hidden layer in \( s(\cdot ; \theta) \), \( L \) denotes the number of layers in \( s(\cdot ; \theta) \), \( M \) denotes the dataset size, \( D \) denotes the data dimension, and \( K \) denotes the number of random vectors used in the Hutchinson trace estimator. Please note that a reasonable assumption for deep generative tasks is \( M \gg D \gg K \) (Song et al., 2019; Grathwohl et al., 2019).

### Algorithm 1 Training Procedure of QC-SGM

| Input: \( \hat{x}, v, s(\cdot ; \theta), \lambda \) |
| --- |
| // (1) Computing \( v^T J^T v \). |
| 1: \( v^T s(\hat{x}; \theta) \leftarrow v \cdot s(\hat{x}; \theta) \) |
| 2: \( v^T J \leftarrow \frac{\partial}{\partial \theta} [v^T s(\hat{x}; \theta)] \) |
| 3: \( v^T J^T v \leftarrow [v^T J]^2 \) |
| // (2) Computing \( v^T J^T J v \). |
| 4: \( v^T J^T J^T v \leftarrow \text{sg} [v^T J] \cdot s(\hat{x}; \theta) \) |
| 5: \( v^T J^T J^T v \leftarrow \frac{\partial}{\partial \theta} [v^T J^T J^T v] \) |
| 6: \( v^T J^T J^T v \leftarrow \text{sg} [v^T J] \cdot s(\hat{x}; \theta) \) |
| // (3) Computing \( \mathcal{L}_{\text{Total}}(\theta) \). |
| 7: \( \mathcal{L}_{\text{est}}^{\text{QC}}(\theta) \leftarrow E_{p_{D}(\theta)}[v^T J^T v - \text{sg} [v^T J] v] \) |
| 8: \( \mathcal{L}_{\text{SM}}(\theta) \leftarrow \text{Eqs. (2), (5), (6)} \) |
| 9: \( \mathcal{L}_{\text{Total}}(\theta) \leftarrow \mathcal{L}_{\text{est}}^{\text{QC}}(\theta) + \lambda \mathcal{L}_{\text{QC}}^{\text{est}}(\theta) \) |
| // (4) Deriving the primary component. |
| 10: Perform Backpropagation through the blue arrows. |
| 11: Perform Backpropagation through the red arrows. |
| 12: Update \( \theta \) |

### Table 2: The asymptotic computational complexity of different objectives discussed in this paper.

| Objective | Complexity |
| --- | --- |
| \( \mathcal{L}_{\text{SM}} \) | \( O(DH + M) \) |
| \( \mathcal{L}_{\text{est}}^{\text{SM}} \) | \( O(DH + L) \) |
| \( \mathcal{L}_{\text{est}}^{\text{QC}} \) | \( O(DH + L \times K) \) |
| \( \mathcal{L}_{\text{est}}^{\text{QC-SGM}} \) | \( O(DH + L \times K) \) |

5 EXPERIMENTAL RESULTS ON THE REAL-WORLD DATASETS

In this section, we examine the effectiveness of the proposed QC-SGMs on four real-world datasets: Cifar-10, Cifar-100 (Krizhevsky & Hinton, 2009), ImageNet-32x32 (Van Oord et al., 2016), and SVHN (Netzer et al., 2011) datasets. We employ the unconstrained architecture as well as the training procedure adopted by NCSN++ (VE) (Song et al., 2021b) as our baseline. On the other hand, QC-NCSN++ (VE), a variant of NCSN++ (VE) regularized by \( \mathcal{L}_{\text{est}}^{\text{QC}} \), is compared against NCSN++ (VE) using the precision (P), recall (R), Asym, NAsym, Fréchet Inception Distance (FID) (Heusel et al., 2017), and Inception Score (IS) (Barratt & Sharma, 2018) metrics. The Asym and NAsym metrics are calculated based on Eqs. (A12) and (A13) for inspecting the non-conservativeness of the SGMs. Please note that the details of the experimental setups are provided in Appendix A.5.2.

Table 3 presents the quantitative results of NCSNs++ (VE) and QC-NCSNs++ (VE) on the Cifar-10, Cifar-100, ImageNet-32x32, and SVHN datasets. It is observed that QC-NCSN++ (VE) is able to achieve better results on the Asym and NAsym metrics than the NCSN++ (VE) baseline on both datasets, indicating that minimizing the proposed \( \mathcal{L}_{\text{est}}^{\text{QC}} \) indeed enhances the conservativeness of QC-NCSN++ (VE). Moreover, QC-NCSN++ (VE) can outperform the baseline in terms of the FID, IS, P, and R metrics under a majority of the experimental setups, suggesting that minimizing the proposed \( \mathcal{L}_{\text{est}}^{\text{QC}} \) is able to improve the sampling performance. The above results thus demonstrate the effectiveness of the proposed regularization term \( \mathcal{L}_{\text{est}}^{\text{QC}} \) on the real-world datasets. In Appendix A.5.2, we further extend the empirical studies on the effectiveness of the proposed regularization term on other real-world datasets.
Table 3: The FID, IS, precision (P), recall (R), Asym, and NAsym of NCSN++ (VE) and QC-NCSN++ (VE) evaluated on the Cifar-10, Cifar-100, ImageNet-32x32, and SVHN datasets. The arrow symbols ↑ / ↓ indicate that higher / lower values correspond to better performance, respectively.

| Method            | Method          | FID ↑ | IS ↑ | P ↑ | R ↑ | Asym ↓ | NAsym ↓ |
|-------------------|-----------------|------|------|-----|-----|--------|---------|
| Cifar-10          | Cifar-100       |      |      |     |     |        |         |
| NCSN++ (VE)       | FID ↑ | 2.20 | 0.00 | 9.89 | 0.7656 | 0.0635 | 1.38e-03 | 1.90e-03 |
| QC-NCSN++ (VE)    | FID ↑ | 2.23 | 0.00 | 9.91 | 0.6759 | 0.0636 | 5.03e-07 | 1.10e-03 |
| SVHN              | FID ↑ | 19.22 | 0.00 | 10.10 | 0.6357 | 0.5186 | 2.05e-07 | 7.17e-04 |
| NCSN++ (VE)       | FID ↑ | 19.48 | 0.00 | 10.40 | 0.6261 | 0.5245 | 1.13e-07 | 5.47e-04 |
| QC-NCSN++ (VE)    | FID ↑ | 13.88 | 0.00 | 11.88 | 0.6372 | 0.6242 | 9.31e-07 | 1.44e-03 |

Figure 3: The trends of \( ||WR^T - RW^T||_F \) and \( ||W - R||_F \) during the minimization process of \( \mathcal{L}_{QC} \). The ‘steps’ on the x-axes refer to the training steps.

6 QCSGM IMPLEMENTED AS A ONE-LAYERED AUTOENCODER

A line of research (Vincent, 2011; Kamyshanska & Memisevic, 2013; Im et al., 2016; Kamyshanska & Memisevic, 2015) focuses on a type of SGM constructed as a one-layered autoencoder, since its property of conservativeness can be systematically analyzed. Such an SGM is represented as \( s(x; \theta) = R h(W^T x + b) + c \), where \( h(\cdot) \) is an activation function, \( b, c \in \mathbb{R}^D \), \( W \in \mathbb{R}^{D \times H} \) are the weights of \( s(\cdot; \theta) \) (i.e., \( \theta = \{R, W, b, c\} \)), and \( H \) is the width of the hidden-layer. As proved in (Im et al., 2016), the output vector field of \( s(\cdot; \theta) \) is conservative if and only if \( WR^T = RW^T \). To ensure the conservativeness of such an SGM, a number of works (Vincent, 2011; Kamyshanska & Memisevic, 2013; 2015) follow the concept of CSGMs and restrict the weights of \( s(\cdot; \theta) \) to be ‘tied,’ i.e., \( W = R \). An SGM with tied weights, however, is only a sufficient condition for its conservativeness, rather than a necessary one. This implies that there must exist some conservative \( s(\cdot; \theta) \) that cannot be modeled using tied weights (Im et al., 2016).

Instead of enforcing an SGM’s weights to be tied (i.e., \( W = R \)), QCSGMs indirectly learn to satisfy the conservativeness condition (i.e., \( WR^T = RW^T \)) through minimizing \( \mathcal{L}_{QC} \). Fig. 3 depicts the trends of \( \mathcal{L}_{QC} \) and \( ||W - R||_F \) during the minimization process of \( \mathcal{L}_{QC} \). As the training progresses, the values of \( \mathcal{L}_{QC} \) approach zero, indicating that \( s(\cdot; \theta) \) learns to output a conservative vector field through minimizing \( \mathcal{L}_{QC} \). In contrast, the values of \( ||W - R||_F \) do not decrease to zero, revealing that minimizing \( \mathcal{L}_{QC} \) does not necessarily lead to \( W = R \). The experimental results thus suggest that QCSGMs can learn to output conservative vector fields that cannot be modeled by one-layered autoencoders with tied weights. This justifies the advantage of QCSGMs over CSGMs. In Appendix A.6.1, we offer more examples to support this observation.

7 CONCLUSION

In this paper, we unveiled the underlying issues of CSGMs and USGMs, and highlighted the importance of preserving both of the architectural flexibility and the property of conservativeness through two motivational experiments. We proposed a new category of SGMs, named QCSGMs, in which the magnitudes of their rotation densities are minimized through a regularization loss for enhancing their property of conservativeness. We showed that such a regularization loss can be reformulated as a scalable variant based on the Hutchinson trace estimator, and demonstrated that it can be efficiently incorporated into the training procedure of SGMs. Finally, we validated the effectiveness of QCSGMs through the experimental results on the real-world datasets, and showcased the advantage of QCSGMs over CSGMs using the example of a one-layered autoencoder.
G. Alain and Y. Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research (JMLR)*, 15(1):3563–3593, 2014.

S. Andrilli and D. Hecker. Chapter 1 - vectors and matrices. In S. Andrilli and D. Hecker (eds.), *Elementary Linear Algebra (Fifth Edition)*, pp. 1–83. Academic Press, Boston, fifth edition edition, 2016. doi: https://doi.org/10.1016/B978-0-12-800853-9.00001-3.

S. Barratt and R. Sharma. A note on the inception score. *arXiv preprint arXiv:1801.01973*, 2018.

C.-H. Chao, W.-F. Sun, B.-W. Cheng, Y.-C. Lo, C.-C. Chang, Y.-L. Liu, Y.-L. Chang, C.-P. Chen, and C.-Y. Lee. Denoising likelihood score matching for conditional score-based data generation. In *Int. Conf. on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=LcF-EEt8cCC

N. Chen, Y. Zhang, H. Zen, R. J Weiss, M. Norouzi, and W. Chan. Wavegrad: Estimating gradients for waveform generation. In *Int. Conf. on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=NsMLjcFaO8O

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1683–1691, 22–24 Jun 2014.

E. Glotzl and O. Richters. Helmholtz decomposition and rotation potentials in n-dimensional cartesian coordinates. 2020.

W. Grathwohl, R. T. Q. Chen, J. Bettencourt, and D. Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *Int. Conf. on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:6840–6851, 2020.

M. F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research (JMLR)*, 6(24):695–709, 2005. URL http://jmlr.org/papers/v6/hyvarinen05a.html

D. J. Im, M. I. Belghazi, and R. Memisevic. Conservativeness of untied auto-encoders. In *Thirtyith AAAI Conf. on Artificial Intelligence (AAAI)*, 2016.

H. Kamysphanska and R. Memisevic. On autoencoder scoring. In *Int. Conf. on Machine Learning (ICML)*, pp. 720–728. PMLR, 2013.

H. Kamysphanska and R. Memisevic. The potential energy of an autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37:1261–1273, 2015.

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

Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *Int. Conf. on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.
T. Kynkänneni, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, abs/1904.06991, 2019.

M. W. Y. Lam, J. Wang, D. Su, and D. Yu. BDDM: Bilateral denoising diffusion models for fast and high-quality speech synthesis. *In Int. Conf. on Learning Representations (ICLR)*, 2022. URL [https://openreview.net/forum?id=L7wzpQttN0](https://openreview.net/forum?id=L7wzpQttN0).

J. Martens, I. Sutskever, and K. Swersky. Estimating the hessian by back-propagating curvature. *arXiv preprint arXiv:1206.6464*, 2012.

M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo. Reliable fidelity and diversity metrics for generative models. 2020.

Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

A. M Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3510–3520, 2017.

A. Quinn Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. *In Int. Conf. on Machine Learning (ICML)*, pp. 8162–8171. PMLR, 2021.

P. Ramachandran, B. Zoph, and Q. V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

G. O Roberts and J. S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(1):255–268, 1998.

G. O Roberts and R. L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363, 1996. doi: [10.3233/Bj/1178291835](https://doi.org/10.3233/Bj/1178291835).

Y. Salimans and J. Ho. Should EBMs model the energy or the score? *In Energy Based Models Workshop at the Int. Conf. on Learning Representations (ICLR)*, 2021. URL [https://openreview.net/forum?id=9AS-TF2JRnb](https://openreview.net/forum?id=9AS-TF2JRnb).

S. Saremi, A. Mehrjou, B. Schölkopf, and A. Hyvärinen. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018.

HM Shemy. *Div, grad curl, and all that: An informal text on vector calculus*, the third edition, 1997.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *In Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, 2019.

Y. Song and S. Ermon. Improved techniques for training score-based generative models. *In Proc. of Conf. on Neural Information Processing Systems (NeurIPS)*, 2020.

Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. *In Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pp. 204, 2019. URL [http://auai.org/uai2019/proceedings/papers/204.pdf](http://auai.org/uai2019/proceedings/papers/204.pdf).

Y. Song, C. Durkan, I. Murray, and S. Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021a.

Y. Song, J. Sohl-Dickstein, D. P Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *In Int. Conf. on Learning Representations (ICLR)*, 2021b. URL [https://openreview.net/forum?id=PxTIG12RRHS](https://openreview.net/forum?id=PxTIG12RRHS).

A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *In Int. Conf. on Machine Learning (ICML)*, pp. 1747–1756, 2016.

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

M. Welling and Y. W Teh. Bayesian learning via stochastic gradient langevin dynamics. *In Proc. of the Int. Conf. on Machine Learning (ICML)*, pp. 681–688. Citeseer, 2011.
Appendix

In this Appendix, we first provide the definitions for the symbols used in the main manuscript and the Appendix in Section A.1. Next, we detail the backpropagation processes described in Section 4.2, and provide the derivations for Eqs. (11) and (17) in Section A.2. Then, we offer a discussion on the normalized asymmetry metric in Section A.3. Subsequently, in Section A.4, we describe the approach to extend a QCSGM to the time-inhomogeneous variant, i.e., QC-NCSN++ (VE) described in Section 5 of the main manuscript. Finally, we provide the detailed experimental configurations in Section A.5, and a number of qualitative and quantitative experimental results in Section A.6.

A.1 List of Notations

In this section, we offer the list of notations used throughout the main manuscript and the Appendix. These notations and their descriptions are summarized in Tables A1 and A2.

| Symbol | Description |
|--------|-------------|
| $M$    | the dataset size. |
| $D$    | the data dimension. |
| $K$    | the number of random vectors used in the Hutchinson trace estimator. |
| $H$    | the dimension of the largest hidden layer in an SGM. |
| $L$    | the number of layers in an SGM. |
| $N$    | the number of discretized points for the estimation of the line integral. |
| $T$    | the number of discretized timesteps for the sampling algorithm. |
| $\alpha$ | the step size used in Langevin dynamics. |
| $\epsilon$ | the score-matching error described in Section 3.1. |
| $\sigma$ | the standard deviation for Gaussian distribution. |
| $\theta$ | the parameters of an SGM. |
| $x \in \mathbb{R}^D$ | a data sample. |
| $\tilde{x} \in \mathbb{R}^D$ | a perturbed data sample. |
| $z \in \mathbb{R}^D$ | a noise vector used in Langevin dynamics. |
| $v \in \mathbb{R}^D$ | a random vector used in the Hutchinson trace estimator. |
| $b, c \in \mathbb{R}^D$ | the bias for the one-layered autoencoder described in Section 6. |
| $W, R \in \mathbb{R}^{D \times H}$ | the weights for the one-layered autoencoder described in Section 6. |
| \{$x^{(i)}\}_{i=1}^M$ | a dataset. |
| \{$\tilde{x}_t\}_{t=1}^T$ | a set of discretized timesteps in Langevin dynamics. |
| \{$x_t^{(i)}\}_{i=1}^N$ | a set of discretized points of the $t$-th timestep used for estimating line integral. |
| \{$v^{(i)}\}_{i=1}^K$ | a set of random vectors drawn from $p(v)$. |
| $p_{\text{data}}$ | the unknown true probability density function (pdf). |
| $p_0(x) = \frac{1}{m} \sum_{i=1}^m \delta(||x - x^{(i)}||)$ | the empirical distribution of a dataset. |
| $p_\sigma (\tilde{x} | x) = \frac{1}{(2\pi)^{D/2}\sigma} e^{-\frac{1}{2\sigma^2}||\tilde{x} - x||^2}$ | the smoothing kernel with mean $x$ and standard deviation $\sigma$ used in Parzen density estimator. |
| $p_\sigma (\tilde{x}) = \int p_\sigma (\tilde{x} | x)p_0(x)dx$ | Parzen density estimator for $p_0(x)$. |
| $p(v)$ | a distribution satisfying $E_{p(v)}[vv^T] = I$ such as a Gaussian or a Radamacher distribution. |

Table A1: The list of symbols used in this paper.
\( \frac{\partial}{\partial \theta} \log p_\sigma(\tilde{x}) \) the gradient of \( \log p_\sigma(\tilde{x}) \) w.r.t. \( \tilde{x} \).

\( \frac{\partial}{\partial \theta} E(\tilde{x}; \theta) \) the gradient of \( E(\tilde{x}; \theta) \) w.r.t. \( \tilde{x} \).

\( \frac{\partial}{\partial \theta} s(\tilde{x}; \theta) \) the Jacobian matrix of \( s(\tilde{x}; \theta) \).

\( J \) the simplified notation for \( \frac{\partial}{\partial \theta} s(\tilde{x}; \theta) \).

\( L_{\text{ESM}} \) Explicit Score Matching (ESM) loss defined in Eq. (2).

\( L_{\text{ISM}} \) Implicit Score Matching loss (ISM) loss defined in Eq. (3).

\( L_{\text{SSM}} \) Sliced Score Matching loss (SSM) loss defined in Eq. (5).

\( L_{\text{DSM}} \) Denoising Score Matching loss (DSM) loss defined in Eq. (6).

\( L_{\text{Total}} \) the total loss of QCSGMs defined in Eq. (15).

\( L_{\text{QC}} \) the proposed regularization loss defined in Eq. (16).

\( L_{\text{QC}}^{\text{Tr}} \) the equivalent variant of \( L_{\text{QC}} \) defined in Eq. (17).

\( L_{\text{QC}}^{\text{sat}} \) the approximated variant of \( L_{\text{QC}}^{\text{Tr}} \) defined in Eq. (18).

\( u^T v = u \cdot v = \sum_i u_i v_i \) inner product between two vectors \( u, v \).

\( \text{tr}(A) = \sum_i A_{i,i} \) trace of a matrix \( A \).

\( \|u\| = \sqrt{\sum_i u_i^2} \) Euclidean norm of a vector \( u \).

\( \|A\|_F = \sqrt{\sum_{i,j} A_{i,j}^2} \) Frobenius norm of a matrix \( A \).

\( \exp(\cdot) \) an exponential function.

\( \text{sg}[\cdot] \) a stop gradient operator.

---

### A.2 Derivations

#### A.2.1 The Derivation of \( s_\epsilon(\tilde{x}) \) in Eq. (11)

Section 3.1 stated that \( s_\epsilon(\tilde{x}) \) can be written as Eq. (11) given \( p_\sigma(\tilde{x}) = \mathcal{N}(0; \sigma^2 I) \) and \( L_{\text{ESM}} < \epsilon \). To justify this, we first demonstrate the derivation of \( s_\epsilon(\tilde{x}) \), and show that \( s_\epsilon(\tilde{x}) \) satisfies \( L_{\text{ESM}} < \epsilon \).

As suggested in [Schey, 1997](#), a non-conservative vector field can be constructed by perturbing a conservative vector field with a unit tangent vector field, which can be established through rotating each vector in the conservative vector field by 90 degrees clockwise. For example, given the true score function \( \frac{\partial}{\partial \tilde{x}} \log p_\sigma(\tilde{x}) = \frac{1}{\sigma^2} [-x_1, -x_2]^T \) of a Gaussian distribution, the associated unit tangent vector field can be written as \( u(\tilde{x}) = \frac{1}{\|\tilde{x}\|} [-x_2, x_1]^T \). Based on such an approach, the non-conservative vector field \( s_\epsilon(\tilde{x}) \) can be constructed by mixing \( \frac{\partial}{\partial \tilde{x}} \log p_\sigma(\tilde{x}) \) and \( u(\tilde{x}) \), expressed as the following:

\[
    s_\epsilon(\tilde{x}) = \frac{1}{\sigma^2} \begin{bmatrix} -x_1 \\ -x_2 \end{bmatrix} + \sqrt{\frac{2\epsilon}{\|\tilde{x}\|^2}} \begin{bmatrix} -x_2 \\ x_1 \end{bmatrix}.
\]

(A1)

In Eq. (A1), the unit tangent vector field \( u(\tilde{x}) \) is scaled by a scaling factor \( \sqrt{2\epsilon} \) such that \( s_\epsilon(\tilde{x}) \) satisfies the constraint \( L_{\text{ESM}} = \epsilon \). However, \( s_\epsilon(\tilde{x}) \) in Eq. (A1) is undefined at \( \tilde{x} = [0, 0]^T \). To resolve this issue, we substitute the denominator of \( u(\tilde{x}) \) with \( \|\tilde{x}\|^2 + \epsilon \), and reformulate \( s_\epsilon(\tilde{x}) \) as the following:

\[
    s_\epsilon(\tilde{x}) = \frac{1}{\sigma^2} \begin{bmatrix} -x_1 \\ -x_2 \end{bmatrix} + \sqrt{\frac{2\epsilon}{\|\tilde{x}\|^2 + \epsilon}} \begin{bmatrix} -x_2 \\ x_1 \end{bmatrix}.
\]

(A2)

This modification causes \( L_{\text{ESM}} < \epsilon \). To redistribute the errors of \( s_\epsilon(\tilde{x}) \) in Eq. (A2), a factor \( \sqrt{\mu(\tilde{x})/p_\sigma(\tilde{x})} \) is included to adjust the magnitude of \( u(\tilde{x}) \), where \( \mu(\tilde{x}) \) is an arbitrary probability...
To show that the equivalence holds, we provide a formal derivation as follows.

**Proposition 1.** Given $\epsilon > 0$, $p_\sigma(\hat{x}) = \mathcal{N}(0; \sigma^2 I)$, and an arbitrary pdf $\mu(\hat{x})$, $s_\epsilon(\hat{x}) = \frac{1}{\sigma^2}[-x_1, -x_2]^T + \sqrt{\frac{2\epsilon\mu(\hat{x})}{p_\sigma(\hat{x})}}[-x_2, x_1]^T$ satisfies the condition $L_{\text{ESM}} < \epsilon$.

**Proof.**

\[
0 < \epsilon \\
\implies x_1^2 + x_2^2 < x_1^2 + x_2^2 + \epsilon, \ \forall \hat{x} \in \mathbb{R}^2 \\
\implies \frac{x_1^2 + x_2^2}{x_1^2 + x_2^2 + \epsilon} < 1, \ \forall \hat{x} \in \mathbb{R}^2 \\
\implies \epsilon \left( \frac{x_1^2 + x_2^2}{x_1^2 + x_2^2 + \epsilon} \right) < \epsilon, \ \forall \hat{x} \in \mathbb{R}^2 \\
\implies \epsilon \left( \frac{x_1^2 + x_2^2}{x_1^2 + x_2^2 + \epsilon} \right) \int_{\hat{x}} \mu(\hat{x}) d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} \frac{\epsilon \mu(\hat{x})}{x_1^2 + x_2^2 + \epsilon} (x_1^2 + x_2^2) d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} (x_1^2 + x_2^2 + \epsilon) (((-x_2)^2 + x_1^2) d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{2\epsilon \mu(\hat{x})}{2p_\sigma(\hat{x})(x_1^2 + x_2^2 + \epsilon)} ((-x_2)^2 + x_1^2) d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{2\epsilon \mu(\hat{x})}{2p_\sigma(\hat{x})(\|\hat{x}\|^2 + \epsilon)} \left\| \frac{-x_2}{x_1} \right\|^2 d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{1}{2} \left\| \frac{2\epsilon \mu(\hat{x})}{p_\sigma(\hat{x})(\|\hat{x}\|^2 + \epsilon)} \left[ \frac{-x_2}{x_1} \right] \right\|^2 d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{1}{2} \left\| \frac{2\epsilon \mu(\hat{x})}{p_\sigma(\hat{x})(\|\hat{x\|}^2 + \epsilon)} \left[ \frac{-x_2}{x_1} \right] \right\|^2 d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{1}{2} \left\| \frac{2\epsilon \mu(\hat{x})}{p_\sigma(\hat{x})(\|\hat{x\|}^2 + \epsilon)} \left[ \frac{-x_2}{x_1} \right] \right\|^2 d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{1}{2} \left\| \frac{2\epsilon \mu(\hat{x})}{p_\sigma(\hat{x})(\|\hat{x\|}^2 + \epsilon)} \left[ \frac{-x_2}{x_1} \right] \right\|^2 d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{1}{2} \left\| \frac{2\epsilon \mu(\hat{x})}{p_\sigma(\hat{x})(\|\hat{x\|}^2 + \epsilon)} \left[ \frac{-x_2}{x_1} \right] \right\|^2 d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{1}{2} \left\| s_\epsilon(\hat{x}) - \frac{1}{\sigma^2} \left[ \frac{-x_1}{-x_2} \right] \right\|^2 d\hat{x} < \epsilon \\
\implies \int_{\hat{x}} p_\sigma(\hat{x}) \frac{1}{2} \left\| s_\epsilon(\hat{x}) - \frac{1}{\sigma^2} \left[ \frac{-x_1}{-x_2} \right] \right\|^2 d\hat{x} < \epsilon \\
\implies L_{\text{ESM}} < \epsilon \]

\[\square\]

### A.2.2 The Derivation of $L_{\text{QC}}^{\text{tr}}$ in Eq. (17)

In Section 4.1, we derived the computationally efficient objective $L_{\text{QC}}^{\text{eff}}$, based on the assumption that $L_{\text{QC}}$ equals $L_{\text{QC}}^{\text{tr}}$. To show that the equivalence holds, we provide a formal derivation as follows.
Proposition 2. $L_{QC}(\theta) = L_{QC}^{\mathcal{U}}(\theta)$.

Proof.

$L_{QC}(\theta) = \mathbb{E}_{p_\theta}(\frac{1}{2} \| J - J^T \|^2_F)$

$= \mathbb{E}_{p_\theta}(\frac{1}{2} \left\| \left( \frac{\partial s(\hat{x}; \theta)}{\partial x} \right) - \left( \frac{\partial s(\hat{x}; \theta)}{\partial x} \right)^T \right\|^2_F)$

$= \mathbb{E}_{p_\theta}(\frac{1}{2} \sum_{i=1}^{D} \sum_{j=1}^{D} \left( \left( \frac{\partial s(\hat{x}; \theta)}{\partial x_j} \right) \right)^2 + \left( \frac{\partial s(\hat{x}; \theta)}{\partial x_i} \right)^2 - 2 \left( \frac{\partial (\hat{x}; \theta)}{\partial x_j} \frac{\partial s(\hat{x}; \theta)}{\partial x_i} \right)]]]$\]

Remark 1. Proposition can also be proved by utilizing the properties of trace, i.e., $\| A \|^2_F = \text{tr} (A^T A)$ and $\text{tr} (AB) = \text{tr} (BA)$, leading to a simplified proof as follows:

$L_{QC}(\theta) = \mathbb{E}_{p_\theta}(\frac{1}{2} \| J - J^T \|^2_F)$

$= \mathbb{E}_{p_\theta}(\frac{1}{2} \text{tr} (J - J^T)^T (J - J^T))$

$= \mathbb{E}_{p_\theta}(\frac{1}{2} \text{tr} ((J^T - J)(J - J^T))$

$= \mathbb{E}_{p_\theta}(\frac{1}{2} \text{tr} ((J^T J - J J^T + J J^T - J J))$

$= \mathbb{E}_{p_\theta}(\frac{1}{2} \left( \text{tr} \left( J^T J \right) - \text{tr} \left( J J^T \right) + \text{tr} \left( J J^T \right) - \text{tr} \left( J J \right) \right)$

$= \mathbb{E}_{p_\theta}(\text{tr} \left( J J^T \right) - \text{tr} \left( J J \right) )$

$= L_{QC}^{\mathcal{U}}(\theta)$

A.2.3 THE DERIVATION OF THE PRIMARY AND SECONDARY COMPONENTS OF $\frac{\partial}{\partial \theta} L_{Total}(\theta)$

In Section 4.2, we decompose $\frac{\partial}{\partial \theta} L_{Total}(\theta)$ as the primary and secondary components, and separately derive them in Steps (4) and (5). To further elaborate on such a backward propagation process, we
offer a detailed description in this subsection. For the sake of notational simplicity, we assume that both the batch size and the number of random vectors $K$ are 1.

According to the rule of sum and the rule of product from vector calculus, the gradient of the total loss $\frac{\partial}{\partial \theta} L_{\text{Total}}(\theta)$ can be decomposed as the sum of $\frac{\partial}{\partial \theta} L_{\text{SM}}(\theta), \lambda \frac{\partial}{\partial \theta} v^T J J^T v, -\lambda (\frac{\partial}{\partial \theta} v^T J) J v$, and $-\lambda v^T J(\frac{\partial}{\partial \theta} v) J v$, indexed as (i)~(iv) respectively. The derivation is shown as the following:

$$
\frac{\partial L_{\text{Total}}(\theta)}{\partial \theta} = \frac{\partial (L_{\text{SM}}(\theta) + \lambda L_{\text{est}}^{\text{ROT}}(\theta))}{\partial \theta} \\
= \frac{\partial L_{\text{SM}}(\theta)}{\partial \theta} + \lambda \frac{\partial L_{\text{est}}^{\text{ROT}}(\theta)}{\partial \theta} \\
= \frac{\partial L_{\text{SM}}(\theta)}{\partial \theta} + \lambda \frac{\partial (v^T J J^T v - v^T J J v)}{\partial \theta} \\
= \frac{\partial L_{\text{SM}}(\theta)}{\partial \theta} + \lambda \frac{\partial v^T J J^T v}{\partial \theta} - \lambda \frac{\partial v^T J J v}{\partial \theta} \\
= \frac{\partial L_{\text{SM}}(\theta)}{\partial \theta} + \lambda \frac{\partial v^T J J^T v}{\partial \theta} + (-\lambda) \frac{\partial v^T J J v}{\partial \theta} + (-\lambda) \frac{\partial v^T J}{\partial \theta} J v.
$$

We name the sum of (i)~(iii) the primary component of $\frac{\partial}{\partial \theta} L_{\text{Total}}(\theta)$, and the term (iv) the secondary component of $\frac{\partial}{\partial \theta} L_{\text{Total}}(\theta)$. Such a decomposition suggests that $\frac{\partial}{\partial \theta} L_{\text{Total}}(\theta)$ can be separately computed based on the computational graph built in Steps (1)~(3) as shown in the upper subplot of Fig. A1. For the primary component, the sum of (i)~(iii) is computed by performing backward propagation through the paths in the computational graph highlighted by the blue arrows in the lower subplot of Fig. A1 using automatic differentiation. For the secondary component, the term (iv) is calculated by performing backward propagation through the red arrows in the lower subplot of Fig. A1. Through these two steps, $\frac{\partial}{\partial \theta} L_{\text{Total}}(\theta)$ can be correctly derived.

A.2.4 THE EQUIVALENCE BETWEEN ZERO-ROTATION-DENSITY CONDITION AND THE CONSERVATIVENESS OF A SCORE MODEL

Lemma 1. $\text{ROT}_{ij}s(\bar{x}; \theta) = 0$ for all $1 \leq i, j \leq D$ if and only if the Jacobian of $s(\bar{x}; \theta)$ (i.e., $J$) is symmetric.

Proof. As defined in Eq. (9), the following holds:

$$
\text{ROT}_{ij}s(\bar{x}; \theta) = 0, \ \forall \ 1 \leq i, j \leq D \\
\Leftrightarrow \frac{\partial s(\bar{x}; \theta)_i}{\partial x_j} - \frac{\partial s(\bar{x}; \theta)_j}{\partial x_i} = 0, \ \forall \ 1 \leq i, j \leq D \\
\Leftrightarrow J_{ij} - J_{ji} = 0, \ \forall \ 1 \leq i, j \leq D \\
\Leftrightarrow J_{ij} = J_{ji}, \ \forall \ 1 \leq i, j \leq D \\
\Leftrightarrow J \text{ is symmetric}.
$$

Lemma 2. According to [Im et al., 2016], the Jacobian of $s(\bar{x}; \theta)$ (i.e., $J$) is symmetric if and only if $s(\bar{x})$ is conservative.

Proposition 3. $\text{ROT}_{ij}s(\bar{x}) = 0$ for all $1 \leq i, j \leq D$ if and only if $s(\bar{x})$ is conservative.

Proof. Based on Lemma 1 and Lemma 2, the proof completes.

A.3 NORMALIZED ASYMMETRY METRIC

In this section, we elaborate on the formulation of the normalized asymmetry metric $NAsym$, which was introduced in Section 3.1 of the main manuscript. In addition, we derive a computationally efficient implementation of $NAsym$ using the Hutchinson trace estimator.
The goal of backward propagation becomes large. To reduce the evaluation cost, we utilize the Hutchinson trace estimator to approximate the squared Frobenius norm of the Jacobian matrix $J$.

This metric measures the ratio of the squared Frobenius norm of the skew-symmetric matrix $N_{\text{asym}}$, suggesting that $J$ becomes skew-symmetric. On the contrary, $N_{\text{asym}} = 0$ indicates that $J$ only contains the symmetric component $J_{\text{sym}}$, suggesting that $J$ is symmetric. Since the squared Frobenius norm of the skew-symmetric matrix can be written as the sum of the squared rotation densities of $s(\tilde{x}; \theta)$, as mentioned in Section 3.1.

Based on Eq. (A4), the Jacobian $J$ of an SGM $s(\tilde{x}; \theta)$ can be written as the sum of a symmetric matrix $J_{\text{sym}} = (J + J^T)/2$ and a skew-symmetric matrix $J_{\text{skew}} = (J - J^T)/2$. Under such a definition, the $N_{\text{asym}}$ metric introduced in Section 3.1 can be formulated as follows:

$$N_{\text{asym}} = \frac{1}{4} \sum_{i,j=1}^{D} \left( \begin{array}{c} \frac{\partial s(\tilde{x}; \theta)}{\partial x_i} - \frac{\partial s(\tilde{x}; \theta)}{\partial x_j} \end{array} \right) \left( \begin{array}{c} \frac{\partial s(\tilde{x}; \theta)}{\partial x_i} - \frac{\partial s(\tilde{x}; \theta)}{\partial x_j} \end{array} \right)^T. \quad (A5)$$

This metric measures the ratio of the squared Frobenius norm of the skew-symmetric matrix $\|J_{\text{skew}}\|_F^2$ to the squared Frobenius norm of the Jacobian matrix $\|J\|_F^2$, and falls within the range $[0, 1]$. $N_{\text{asym}} = 1$ corresponds to the condition where $J_{\text{skew}}$ dominates $J$, implying that $J$ is skew-symmetric. On the contrary, $N_{\text{asym}} = 0$ indicates that $J$ only contains the symmetric component $J_{\text{sym}}$, suggesting that $J$ is symmetric. Since the squared Frobenius norm of the skew-symmetric matrix can be written as the sum of the squared rotation densities of $s(\tilde{x}; \theta)$, as mentioned in Section 3.1.

**An Efficient Implementation of $N_{\text{asym}}$.** Since Eq. (A5) involves the explicit calculation of the Jacobian matrix $J$, evaluating the $N_{\text{asym}}$ metric for a single instance $\tilde{x}$ requires $D$ times of backward propagations. This indicates that the evaluation cost may grow significantly when $D$ and a test set size become large. To reduce the evaluation cost, we utilize the Hutchinson trace estimator to approximate...
Algorithm A1 Predictor-Corrector (PC) Sampler

1: \( \hat{x} = \mathcal{N}(0, \sigma^2_{\text{max}}) \)
2: for \( t = T - 1 \) to 0
3: // Predictor
4: \( z \sim \mathcal{N}(0, I) \)
5: \( \hat{x}_t \leftarrow \hat{x}_t + (\sigma^2_{t+1} - \sigma^2_t) s(\hat{x}_{t+1}; \theta, \sigma_{t+1}) + \sqrt{\sigma^2_{t+1} - \sigma^2_t} z \)
6: // Corrector
7: \( z \sim \mathcal{N}(0, I) \)
8: \( \tilde{x}_t \leftarrow \tilde{x}_t + \alpha_t s(\tilde{x}_t, \sigma_t) - \sqrt{2\alpha_t} z \)
9: return \( x_0 \)

the \( \text{NAsym} \) metric based on the following derivation:

\[
\text{NAsym} = E_{p(\hat{x})}\left[ \frac{1}{4} \left( \left\| J - J^T \right\|_F^2 \right) \right] = E_{p(\hat{x})}\left[ \frac{1}{2} \left( \frac{1}{\text{tr}(J^TJ)} - \text{tr}(JJ^T) \right) \right] \\
= E_{p(\hat{x})}\left[ \frac{1}{2} \left( E_{p(v)}[v^TJ^TJ^Tv] - E_{p(v)}[v^TJJ^Tv] \right) \right] \\
= E_{p(\hat{x})}\left[ \frac{1}{2} E_{p(v)}\left[ \frac{v^TJJ^Tv - v^TJJ^Tv}{v^TJ^Tv} \right] \right] ,
\]

where \( p(v) \) satisfies \( E_{p(v)}[vv^T] = I \). The expectation \( E_{p(v)}[\cdot] \) can be approximated using \( K \) random vectors. In addition, the terms \( v^TJJ^Tv \) and \( v^TJJ^Tv \) in Eq. (A6) can be efficiently calculated based on Steps (1) and (2) described in Section 4.2. This suggests that the computational cost of evaluating \( \text{NAsym} \) can be significantly reduced when \( K \ll D \). In Section A.5.2, we describe an approach to measure \( \text{NAsym} \) on the Cifar-10 and ImageNet-32x32 datasets in detail.

A.4 Time-Inhomogeneous QCSGMs

In this section, we demonstrate how a QCSGM is converted to its time-inhomogeneous variant QC-NCSN++ (VE), which was described in Section 5 of the main manuscript. We first explain the modifications made in the sampling process. Then, we elaborate on the corresponding adjustments in the score-matching objective and the regularization loss.

Sampling Process. QC-NCSN++ (VE) adopts the Predictor-Corrector (PC) sampler identical to that employed in NCSN++ (VE) (Song et al., 2021b), which is a time-inhomogeneous sampling algorithm as shown in Algorithm A1. In this sampler, SGM and step size are respectively represented as \( s(\cdot; \theta, \sigma_t) \) and \( \alpha_t = \frac{\partial}{\partial \sigma_t} \sigma_t \). In QC-NCSN++ (VE) and NCSN++ (VE), \( \sigma_t \) is set to \( \sigma_{\text{min}}(\sigma_{\text{min}}/\sigma_{\text{max}}) \) (Song et al., 2021b), where \( T \) is the total number of timesteps in the sampling process, \( \sigma_{\text{min}} \) is a constant representing a minimal noise scale, and \( \sigma_{\text{max}} \) is a constant denoting a maximal noise scale.

Training Objectives. Since the above time-inhomogeneous sampling process requires the SGM \( s(\cdot; \theta, \sigma_t) \) to be conditioned on a time-dependent standard deviation \( \sigma_t \), the training objectives of \( s(\cdot; \theta, \sigma_t) \) have to be modified accordingly. For example, the score-matching objective \( E_{\text{DSM}} \) used in QC-NCSN++ (VE) and NCSN++ (VE) is modified as follows:

\[
E_{\mathcal{U}(t)} \left[ \lambda(t) E_{p_{\sigma_t}(\hat{x}|x)p_{\theta}(x)} \left[ \left\| s(\hat{x}; \theta, \sigma_t) - \frac{\partial \log p_{\sigma_t}(\hat{x}|x)}{\partial \hat{x}} \right\|^2 \right] \right] ,
\]

where \( \mathcal{U}(t) \) is a uniform distribution defined on the interval \([0, T]\), and \( \lambda(t) \) is a time-dependent coefficient for balancing the loss functions of different \( t \). Meanwhile, the regularization term \( E_{\text{QC}} \) used in QC-NCSN++ (VE) is adjusted according to \( \lambda(t) \), which is formulated as follows:

\[
E_{\mathcal{U}(t)} \left[ \lambda(t) E_{\phi(\hat{x})} \left[ E_{p(v)} \left[ v^TJJ^Tv - v^TJJ^Tv \right] \right] \right] , \tag{A8}
\]

where \( J = \frac{\partial}{\partial \sigma_t} s(\hat{x}; \theta, \sigma_t) \).
Figure A2: (a) The visualizations of the 8-Gaussian, Spirals, and Checkerboard datasets. (b) The grid points comprising \( \mathcal{D} \).

A.5 EXPERIMENTAL SETUPS

In this section, we elaborate on the experimental configurations and provide the detailed hyperparameter setups for the experiments presented in Sections 3 and 5 of the main manuscript.

A.5.1 EXPERIMENTAL SETUPS FOR THE MOTIVATIONAL EXAMPLES

**Datasets.** The motivational experiments in Section 3 are performed on the 8-Gaussian, Spirals, and Checkerboard datasets as shown in Fig. A2 (a). The data points of the 8-Gaussian dataset are sampled from eight separate Gaussian distributions centered at \((\cos(\frac{w}{4}), \sin(\frac{w}{4}))\), where \(w \in \{1, \ldots, 8\}\). The data points of the Spirals dataset are sampled from two separate curves \((-\pi \sqrt{w} \cos(\pi \sqrt{w}), \pi \sqrt{w} \sin(\pi \sqrt{w}))\) and \((\pi \sqrt{w} \cos(\pi \sqrt{w}), -\pi \sqrt{w} \sin(\pi \sqrt{w}))\), where \(w \in [0, 1]\). Lastly, the data points of the Checkerboard dataset are sampled from 
\[ (4w - 2, t - 2s + \lceil w \rceil \mod 2) \]
where \(w \in [0, 1]\), \(s \in \{0, 1\}\), \(\lceil \cdot \rceil\) is a floor function, and \(\mod\) represents the modulo operation. For all of these three datasets, \(p_0(\tilde{x})\) is established by sampling 50,000 data points (i.e., \(M = 50,000\)).

**Training and Implementation Details.** The network architecture of \(f\) is a three-layered multilayer perceptron (MLP) with \((128, 64, 32)\) neurons and Swish (Ramachandran et al. 2017) as its activation function. This model architecture is the same as that used in the two-dimensional experiments of (Chao et al. 2022). Please note that we do not adopt the architecture in (Salimans & Ho 2021) since sophisticated architectures such as ResNet and U-Net employed in (Salimans & Ho 2021) are not suitable for low-dimensional datasets. The SGMs \(s_U\) and \(s_C\) are trained utilizing the Adam optimizer (Kingma & Ba 2014) with a learning rate of \(1 \times 10^{-2}\) and a batch size of 5,000 for 100,000 iterations. The balancing factor \(\lambda\) and the standard deviation \(\sigma\) are fixed to 5 and 0.5, respectively.

**Evaluation Method.** The precision and recall metrics are calculated using 10,000 sample points. On the other hand, the \(\text{Asym}, \text{NAsym}\) metrics, and the score errors are approximated based on the following formulas:

\[
\mathbb{E}_{p_\sigma(\tilde{x})} \left[ \left\| s(\tilde{x}; \theta) - \frac{\partial \log p_\sigma(\tilde{x})}{\partial \tilde{x}} \right\| \right] \approx \sum_{\tilde{x} \in \mathcal{D}} p_\sigma(\tilde{x}) \left\| s(\tilde{x}; \theta) - \frac{\partial \log p_\sigma(\tilde{x})}{\partial \tilde{x}} \right\|, \quad (A9)
\]

\[
\mathbb{E}_{p_\sigma(\tilde{x})} \left[ \frac{1}{2} \| J - J^T \|_F^2 \right] \approx \sum_{\tilde{x} \in \mathcal{D}} p_\sigma(\tilde{x}) \frac{1}{2} \| J - J^T \|_F^2, \quad (A10)
\]

\[
\mathbb{E}_{p_\sigma(\tilde{x})} \left[ \frac{1}{4} \| J - J^T \|_F^2 \right] \approx \sum_{\tilde{x} \in \mathcal{D}} p_\sigma(\tilde{x}) \frac{1}{4} \| J - J^T \|_F^2, \quad (A11)
\]

where \(\mathcal{D}\) denotes a set of 1,600 grid points. A visualization of \(\mathcal{D}\) is depicted in Fig. A2 (b).

A.5.2 EXPERIMENTAL SETUPS FOR THE EVALUATIONS ON THE REAL-WORLD DATASETS

**Datasets.** The experiments presented in Section 5 are performed on the Cifar-10, Cifar-100 (Krizhevsky & Hinton 2009), ImageNet-32x32 (Van Oord et al. 2016), and SVHN (Netzer et al. 2011) datasets. The training and test sets of Cifar-10 and Cifar-100 contain 50,000 and 10,000 images, respectively. The training and test sets of SVHN contain 73,257 and 26,032 images, respectively. On the other hand, the training and the test sets of ImageNet-32x32 consist of 1,281,149 and 49,999 images, respectively.
Training and Implementation Details. NCSN++ (VE) and QC-NCSN++ (VE) are both implemented using the Pytorch framework. NCSN++ (VE) and QC-NCSN++ (VE) are trained using the Adam optimizer with a learning rate of $2 \times 10^{-4}$. The batch size is fixed to 128, while the value of $K$ for QC-NCSN++ (VE) is fixed to 1. The training procedure of QC-NCSN++ (VE) consists of two stages. In the first stage, QC-NCSN++ (VE) is optimized according to the score matching objective, which requires 600,000 training iterations for convergence. In the second stage, the regularization term $L_{\text{reg}}$ is incorporated during the training process, which requires 150,000 training iterations for convergence. The maximal and minimal noise scales $\sigma_{\text{max}}$ and $\sigma_{\text{min}}$ are set to 50 and 0.01, respectively. The total number of timesteps $T$ in the sampling process is set to 1,000. The balancing factor $\lambda$ is set to 0.0001.

Evaluation Method. The asymmetry $Asym$ and normalized asymmetry $NAsym$ metrics are evaluated on 100 discretized timesteps on the test sets of both datasets. Specifically, the $Asym$ metric is calculated based on the following equation:

$$Asym = \frac{1}{|T|} \sum_{i \in T} \frac{1}{|D|} \mathbb{E}_{p(v)} \left[ v^T J J^T v - v^T J J^T v \right],$$  \hspace{1cm} (A12)

where $D$ represents the test set, and $T = \{ \frac{i}{100} | i = 1 \}^{100}$. On the other hand, the $NAsym$ metric is evaluated based on the following equation:

$$NAsym = \frac{1}{|T|} \sum_{i \in T} \frac{1}{|D|} \mathbb{E}_{p(v)} \left[ \frac{v^T J J^T v - v^T J J^T v}{v^T J J^T v} \right],$$  \hspace{1cm} (A13)

The expectations $\mathbb{E}_{p(v)}[\cdot]$ in Eqs. (A12) and (A13) are estimated with $K = 1$. The metrics for sampling performance (i.e., FID, IS, precision and recall) are evaluated using the tensorflow-gan library as well as the official evaluation package implemented by Kynkänneni et al. [2019] Naeem et al. [2020].

A.6 Additional Experimental Results

In this section, we provide a number of additional experimental results. In Section A.6.1, we present additional experimental results of QCSGMs implemented as one-layered autoencoders to support our observation presented in Section 6 of the main manuscript. In Section A.6.2, we provide a comparison between USGMs, CSGMs, and QCSGMs in terms of their performance during the training time. In Section A.6.3, we demonstrate the impact of the model architecture on the performance of USGMs and CSGMs. In Section A.6.4, we demonstrate the impact of the choices of $\lambda$ on the performance of QCSGMs. Finally, in Sections A.6.5 and A.6.6, we provide additional quantitative and qualitative results on the real-world datasets.

A.6.1 QCSGMs Implemented as One-Layered Autoencoders

In Section 6, we leveraged the example of an one-layered autoencoder $s(\hat{x}; \theta) = Rh(W^T \hat{x} + b) + c$ to demonstrate the advantage of QCSGMs over CSGMs. Our experimental results in Fig. 3 reveals that QCSGMs can learn to output conservative vector fields, which cannot be captured by CSGMs with tied weights (i.e., $R = W$). To further solidify our assumption, we provide additional examples in Fig. A3. Fig. A3 depicts the trends of $\|WR^T - RW^T\|_F$ and $\|W - R\|_F$ during the minimization process of $L_{\text{QC}}$ with four different seeds. As the training progresses, $L_{\text{QC}}$ and $\|WR^T - RW^T\|_F$ both approach zero in all of these four examples. In contrast, the values of $\|W - R\|_F$ do not approach zero, and the trends of $\|W - R\|_F$ for these four examples differ. The above experimental evidence demonstrates that QCSGMs can learn to output conservative vector fields with $R \neq W$, and thus justify the advantage of QCSGMs over CSGMs.

To further showcase the benefit of adopting QCSGMs over CSGMs, we include a comparison of QCSGMs and CSGMs in terms of their score matching ability. Fig. A4 shows that QCSGMs demonstrate lower score errors in comparison to CSGMs when both of them are implemented as one-layered autoencoders and trained on a Gaussian distribution.
Figure A3: The trends of $\|WR^T - RW^T\|_F$ and $\|W - R\|_F$ during the minimization process of $L_{QC}$. The ‘steps’ on the x-axes refer to the training steps.

Figure A4: The trends of $\|WR^T - RW^T\|_F$, $\|W - R\|_F$, $L_{QC}$, and the score error of QCSGM and CSGM during the minimization process of $L_{Total}$. The ‘step’ on the x-axes stands for the training steps. The curves depict the mean and 95% confidence interval of three times of training. In this experiment, CSGM and QCSGM are implemented using one-layered autoencoders.
Figure A5: The trend of score errors of USGM, CSGM, and QCSGM trained with $L_{\text{est}}^{\text{QC}}$ on the 8-Gaussian dataset. (a) The trend of score error when the training step is fixed. (b) The trend of score error when the training time (i.e., wall time) is fixed. The curves depict the mean and 95% confidence interval of three times of training. For a better visualization, the presented curves are smoothed by a moving average filter with its window size set to 15.

Table A3: The score errors of USGMs, CSGMs, and QCSGMs constructed with different model architectures. The results are evaluated on the 8-Gaussian dataset. The arrow symbol ↓ indicates that lower values correspond to better performance.

| Number of neurons in each layer | Model | Score Error (↓) | Model | Score Error (↓) | Model | Score Error (↓) |
|---------------------------------|-------|----------------|-------|----------------|-------|----------------|
| [64, 64, 64]                   | CSGM  | 5.54e-1        | USGM  | 3.98e-1        | QCSGM | 3.85e-1        |
| [64, 64, 128]                  |       |                |       |                |       |                |
| [64, 128, 64]                  |       |                |       |                |       |                |
| [64, 128, 128]                 |       |                |       |                |       |                |
| [128, 64, 64]                  |       |                |       |                |       |                |
| [128, 64, 128]                 |       |                |       |                |       |                |
| [128, 128, 64]                 |       |                |       |                |       |                |
| [128, 128, 128]                |       |                |       |                |       |                |

A.6.2 A Comparison on the Performance and Training Time

Figs. A5 (a) and (b) illustrate the trend of scores errors of USGMs, CSGMs, and QCSGMs on the 8-Gaussian dataset. The results revealed that QCSGMs have better performance in comparison to USGMs and CSGMs when either the training time or the wall time is fixed.

A.6.3 The Impact of the Model Architecture on the Performance of SGMs

In Section 3.2, we show that USGMs and QCSGMs have better empirical performance in terms of the score-matching accuracy in comparison to CSGMs. To demonstrate that this statement holds under different implementations of $f$, we provide a performance comparison of these SGMs with different architectural settings in Table A3.

Table A3 compares the performance of CSGMs, USGMs, and QCSGMs. All of them are constructed as three-layered MLPs with different number of neurons in each layer. It is observed that USGMs and QCSGMs are able to outperform CSGMs in terms of score errors with a noticeable margin. This result demonstrates the advantage of USGMs and QCSGMs over CSGMs, and highlights the importance of preserving the architectural flexibility.

A.6.4 The Impact of the Choices of $\lambda$ on the Performance of QCSGMs

Conservativeness. The first and second rows of Table A4 compare the conservativeness of USGM, CSGM, and QCSGMs trained under different values of $\lambda$ in terms of the $\text{Asym}$ and $\text{NAsym}$ metrics on the Checkerboard dataset. The experimental results demonstrate that the conservativeness of QCSGMs is improved (i.e., $\text{Asym}$ and $\text{NAsym}$ decrease) as the value of $\lambda$ increases, and therefore validate the effectiveness of $L_{\text{QC}}$.

Score Matching Ability. The third row of Table A4 compares the score errors of USGM, CSGM, and QCSGMs trained using different values of $\lambda$ on the Checkerboard dataset. It is observed that QCSGMs trained using $\lambda = \{1.0, 2.5, 5.0, 10.0, 50.0\}$ outperform CSGM and USGM by a noticeable
Table A4: The evaluation results of USGMs, CSGMs, and QCSGMs trained using different values of $\lambda$ on the Checkerboard dataset. The arrow symbol ↓ indicates that lower values correspond to better performance.

| $\lambda$ | USGM   | CSGM   | QCSGM  |
|-----------|--------|--------|--------|
| 0.0       |        |        |        |
| Asym (↓)  | 1.27±0.00 e-1 | 1.05±0.00 e-2 | 6.42±0.00 e-3 |
| NAsym (↓) | 1.93±0.00 e-2 | 2.73±0.00 e-3 | 6.69±0.00 e-4 |
| Score Error (↓) | 3.804±0.000 e-1 | 1.16±0.000 e-1 | 3.04±0.001 e-1 |

Table A5: The evaluation results of QC-NCSN++ (VE) on the precision and recall metrics on the Cifar-10 dataset under different choices of sampling steps.

| Metric                    | Step=1,000 | Step=500 | Step=100 |
|---------------------------|------------|----------|----------|
|                           | Asym | NAsym | precision | recall | precision | recall | precision | recall |
| NCSN++ (VE)               | 1.88e8 | 1.90e-3 | 0.6756    | 0.6035  | 0.6682    | 0.6026  | 0.4984    | 0.5139  |
| QC-NCSN++ (VE)            | 5.03e7 | 1.10e-3 | 0.6759    | 0.6036  | 0.6686    | 0.6048  | 0.5030    | 0.5141  |

margin. The results thus validate that the score matching ability of QCSGMs is not sensitive to the choices of $\lambda$.

A.6.5 A COMPARISON OF SAMPLING EFFICIENCY ON THE CIFAR-10 DATASET

The experimental results presented in Section 3.1 indicate that the non-conservativeness of an SGM may degrade its sampling performance under a fixed number of sampling steps. To further examine the sampling inefficiency issue, we perform experiments on the Cifar-10 dataset and compare the sampling performance of NCSN++ (VE) and QC-NCSN++ (VE) under different numbers of sampling steps. The evaluation results are shown in Table A5. It is observed that QC-NCSN++ (VE) are able to outperform NCSN++ (VE) in terms of the precision and recall metrics. The experimental results therefore justify the effectiveness of $L_{QC}$ on the real-world dataset.

A.6.6 VISUALIZED EXAMPLES

Figs. A6-A9 depict a few uncurated visualized examples that qualitatively demonstrate the sampling quality of NCSN++ (VE) and QC-NCSN++ (VE) on the real-world datasets.
Figure A6: The visualized examples generated using (a) NCSN++ (VE) and (b) QC-NCSN++ (VE) on the Cifar-10 dataset.
Figure A7: The visualized examples generated using (a) NCSN++ (VE) and (b) QC-NCSN++ (VE) on the Cifar-100 dataset.
Figure A8: The visualized examples generated using (a) NCSN++ (VE) and (b) QC-NCSN++ (VE) on the ImageNet-32x32 dataset.
Figure A9: The visualized examples generated using (a) NCSN++ (VE) and (b) QC-NCSN++ (VE) on the SVHN dataset.