Quasi-Conservative Score-based Generative Models

Anonymous authors
Paper under double-blind review

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 modeling ability. In addition, we show that USGMs' inability to preserve the property of *conservativeness* may lead to 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 each updates 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_{data}, its empirical distribution $p_0(\boldsymbol{x})$ is established through sampling M independent and identically distributed D-dimensional vectors $\{\boldsymbol{x}^{(i)} : \boldsymbol{x}^{(i)} \in \mathbb{R}^D\}_{i=1}^M$, represented as a Dirac delta distribution, i.e., $p_0(\boldsymbol{x}) \triangleq \frac{1}{M}\sum_{i=1}^M \delta(\|\boldsymbol{x} - \boldsymbol{x}^{(i)}\|)$. To ensure the probability density function (pdf) is everywhere non-zero and differentiable, previous literature (Vincent, 2011) proposed to replace $p_0(\boldsymbol{x})$ with Parzen density estimator $p_\sigma(\tilde{\boldsymbol{x}}) \triangleq \int_{\boldsymbol{x}} p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) p_0(\boldsymbol{x}) d\boldsymbol{x}$, where $p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \triangleq \frac{1}{(2\pi)^{D/2}\sigma^D} e^{\frac{-1}{2\sigma^2}\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|^2}$ is an isotropic Gaussian smoothing kernel with a standard deviation σ. When $\sigma > 0$, the score function of $p_\sigma(\tilde{\boldsymbol{x}})$ has a closed form (Chao et al., 2022), which can be formulated as:

$$\frac{\partial \log p_\sigma(\tilde{\boldsymbol{x}})}{\partial \tilde{\boldsymbol{x}}} = \frac{\sum_{i=1}^M \frac{1}{\sigma^2}(\boldsymbol{x}^{(i)} - \tilde{\boldsymbol{x}}) p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}^{(i)})}{\sum_{i=1}^M p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}^{(i)})}. \tag{1}$$

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

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

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 $\frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}})$ in the training objective, was introduced to efficiently train $s(\tilde{\boldsymbol{x}};\theta)$. ISM employs an equivalent loss $\mathcal{L}_{\text{ISM}}$ expressed as follows:

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

where $\frac{\partial}{\partial \tilde{\boldsymbol{x}}} s(\tilde{\boldsymbol{x}};\theta)$ corresponds to the Jacobian matrix of $s(\tilde{\boldsymbol{x}};\theta)$, and $\operatorname{tr}(\cdot)$ denotes the trace of a matrix. Although $\mathcal{L}_{\text{ISM}}$ avoids the calculation of Eq. (1), the calculation of $\operatorname{tr}\left(\frac{\partial}{\partial \tilde{\boldsymbol{x}}} s(\tilde{\boldsymbol{x}};\theta)\right)$ 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) (Song et al., 2019) loss, was proposed to approximate $\operatorname{tr}\left(\frac{\partial}{\partial \tilde{\boldsymbol{x}}} s(\tilde{\boldsymbol{x}};\theta)\right)$ in $\mathcal{L}_{\text{ISM}}$

with the Hutchinson trace estimator (Hutchinson, 1989). Given a random vector $\boldsymbol{v}$ drawn from a distribution $p(\boldsymbol{v})$ satisfying $\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}\boldsymbol{v}^T\right] = I$, the Hutchinson trace estimator replaces the trace of a square matrix A with $\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^T A \boldsymbol{v}\right]$, which can be derived as:

$$\text{tr}\,(A) = \text{tr}\,(AI) = \text{tr}\,\left(A\mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{v}^T]\right) = \mathbb{E}_{p(\boldsymbol{v})}[\text{tr}\,\left(A\boldsymbol{v}\boldsymbol{v}^T\right)] = \mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}^T A\boldsymbol{v}]. \tag{4}$$

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

$$\mathcal{L}_{\text{SSM}}(\theta) = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|s(\tilde{\boldsymbol{x}};\theta)\right\|^2 + \mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^T\frac{\partial s(\tilde{\boldsymbol{x}};\theta)}{\partial\tilde{\boldsymbol{x}}}\boldsymbol{v}\right]\right]. \tag{5}$$

The vector-Jacobian product $\boldsymbol{v}^T\frac{\partial}{\partial\tilde{\boldsymbol{x}}}s(\tilde{\boldsymbol{x}};\theta)$ can be calculated with a single backward propagation using automatic differentiation (Martens et al., 2012), and the expectation can be approximated using K independently sampled vectors $\{\boldsymbol{v}^{(i)}\}_{i=1}^K$. Therefore, the computation of $\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^T\frac{\partial}{\partial\tilde{\boldsymbol{x}}}s(\tilde{\boldsymbol{x}};\theta)\boldsymbol{v}\right]$ in Eq. (5) can be less expensive than $\text{tr}\left(\frac{\partial}{\partial\tilde{\boldsymbol{x}}}s(\tilde{\boldsymbol{x}};\theta)\right)$ 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 $\mathcal{L}_{\text{SSM}}$:

$$\mathcal{L}_{\text{DSM}}(\theta) = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})p_0(\boldsymbol{x})}\left[\frac{1}{2}\left\|s(\tilde{\boldsymbol{x}};\theta) - \frac{\partial\log p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x})}{\partial\tilde{\boldsymbol{x}}}\right\|^2\right], \tag{6}$$

where $\frac{\partial}{\partial\tilde{\boldsymbol{x}}}\log p_\sigma(\tilde{\boldsymbol{x}}|\boldsymbol{x}) = \frac{1}{\sigma^2}(\boldsymbol{x} - \tilde{\boldsymbol{x}})$. Since the computational cost of $\mathcal{L}_{\text{DSM}}$ 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;a) that pursue training efficiency.

2.2 Sampling Process

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

$$\tilde{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{x}}_t + \alpha s(\tilde{\boldsymbol{x}};\theta) + \sqrt{2\alpha}\boldsymbol{z}_t, \tag{7}$$

where α is the step size, t is the timestep, $\boldsymbol{z}_t \in \mathbb{R}^D$ is a noise vector sampled from a normal distribution $\mathcal{N}(0, I)$. Under the condition where $\alpha \to 0$ and $T \to \infty$, $\tilde{\boldsymbol{x}}_T$ can be generated as if it is directly sampled from $p_\sigma(\tilde{\boldsymbol{x}})$ (Roberts & Rosenthal, 1998; Welling & Teh, 2011). Despite the theoretical guarantee of Langevin dynamics, it empirically suffers from the slow mixing issue as discussed by (Song & Ermon, 2019), which limits its ability of being utilized in practical data generation scenarios. To resolve this issue, a recent study (Song et al., 2021b) proposed to extend Eq. (7) to a time-inhomogeneous variant by making the noise scale σ, the score model $s(\cdot\,;\theta)$, and step size α dependent on t. Specifically, they consider a continuous sampling process defined using a stochastic differential equation (SDE) as follows:

$$d\mathbf{x}(t) = [\mathbf{f}(\mathbf{x}(t), t) - g(t)^2 s(\mathbf{x}(t), t;\theta)]dt + g(t)d\bar{\mathbf{w}}, \tag{8}$$

where $\{\mathbf{x}(t)\}_{t=0}^T$ is a set of time dependent variables, dt is an infinitesimal negative timestep, $\bar{\mathbf{w}}$ represents the Wiener process, $\mathbf{f}(\cdot, t)$ is the drift coefficient, and $g(t)$ is the diffusion coefficient. Contemporary score-based generation frameworks (Ho et al., 2020; Song et al., 2021b;a; Nichol & Dhariwal, 2021; Xu et al., 2022) implement such a sampling process in two different ways according to the discretization method used. One branch of them (Ho et al., 2020; Song et al., 2021b) follows the concept of Eq. (7) to discretize Eq. (8) using equal-sized steps. The other branch of them (Song et al., 2021a; Xu et al., 2022) leverages an ordinary differential equation (ODE) solver to solve the deterministic variant of Eq. (8) using adaptive sampling step sizes.

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 $s(\tilde{\boldsymbol{x}};\theta)$ 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 $\tilde{\boldsymbol{x}} \in \mathbb{S}$, which can be equivalently expressed as the zero-rotation-density ($\overline{\mathrm{ROT}}_{ij}$) (Glotzl & Richters, 2020) condition expressed as follows:

$$\overline{\mathrm{ROT}}_{ij} s(\tilde{\boldsymbol{x}};\theta) = \frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j} - \frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i} = 0, \;\; 1 \leq i,j \leq D, \tag{9}$$

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

For CSGMs, $p_\sigma(\tilde{\boldsymbol{x}})$ is modeled as a Boltzman distribution $p(\tilde{\boldsymbol{x}};\theta) = \exp\left(-E(\tilde{\boldsymbol{x}};\theta)\right)/Z(\theta)$, 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 $s(\tilde{\boldsymbol{x}};\theta) = \frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p(\tilde{\boldsymbol{x}};\theta) = -\frac{\partial}{\partial \tilde{\boldsymbol{x}}} E(\tilde{\boldsymbol{x}};\theta)$. This implies that $s(\tilde{\boldsymbol{x}};\theta)$ is conservative. In other words, $s(\tilde{\boldsymbol{x}};\theta)$ satisfies the zero-rotation-density condition in Eq. (9), since the mixed second derivatives of $E(\tilde{\boldsymbol{x}};\theta)$ are equivalent (Alain & Bengio, 2014), which can be shown as the following:

$$\overline{\mathrm{ROT}}_{ij} s(\tilde{\boldsymbol{x}};\theta) = \frac{\partial^2 E(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}_j \partial \tilde{\boldsymbol{x}}_i} - \frac{\partial^2 E(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}_i \partial \tilde{\boldsymbol{x}}_j} = 0, \;\; 1 \leq i,j \leq D. \tag{10}$$

On the other hand, since USGMs do not follow the aforementioned modeling procedure to assign their output vector field as the gradients of a scalar-valued function, the conservativeness of USGMs is not guaranteed. Although it is possible to ensure the conservativeness of an USGM under an ideal scenario that $s(\tilde{\boldsymbol{x}};\theta)$ perfectly models $\frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}})$ for all $\tilde{\boldsymbol{x}} \in \mathbb{R}^D$, 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.

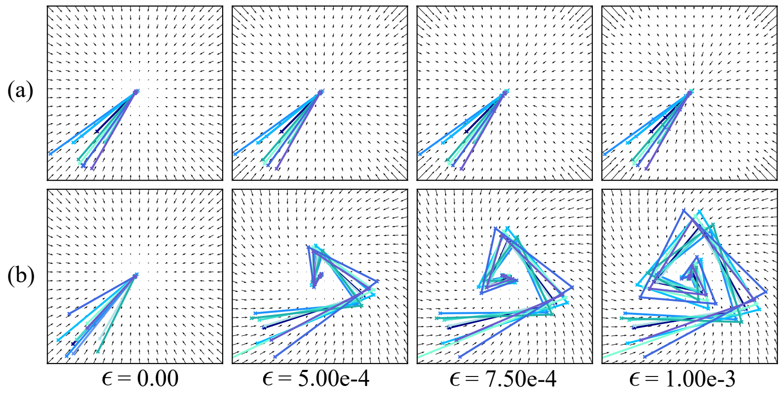
3.1 The Influences of Non-Conservativeness on Sampling Efficiency

The sampling processes described in Section 2.2 are formulated under an ideal scenario that $s(\tilde{\boldsymbol{x}};\theta) = \frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}}), \forall \tilde{\boldsymbol{x}} \in \mathbb{R}^D$. 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 this example, we inspect the impact of the non-conservativeness of USGMs on the sampling process by comparing the sampling efficiency of a USGM and a CSGM under the same approximation error ϵ, i.e., $\mathcal{L}_{\mathrm{ESM}} = \epsilon$. To quantitatively evaluate the non-conservativeness of these SGMs, we measure the magnitude of $\overline{\mathrm{ROT}}_{ij} s(\tilde{\boldsymbol{x}};\theta)$ using the asymmetry metric *Asym* $\in [0, \infty)$ defined as:

$$\textit{Asym} = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\sum_{i,j=1}^{D}\left(\overline{\mathrm{ROT}}_{ij} s(\tilde{\boldsymbol{x}};\theta)\right)^2\right] = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|J - J^T\right\|_F^2\right], \tag{11}$$

where $J = \frac{\partial}{\partial \tilde{\boldsymbol{x}}} s(\tilde{\boldsymbol{x}})$, $\|\cdot\|_F$ is the Frobenius norm. We also measure the normalized asymmetry metric *NAsym* $\in [0, 1]$ defined as $\mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\left\|J - J^T\right\|_F^2 / (4\left\|J\right\|_F^2)\right]$. To evaluate the sampling efficiency, we calculate the number of function evaluation (NFE) required for all sample points to move to the target during the sampling process. In this example, the USGM and the CSGM are denoted as s_{U} and s_{C}, and constructed based on Eq. (A6) presented in Appendix A.5.1.

For an illustrative purpose, we present the visualization of the sampling processes as well as the evaluation results under different choices of ϵ for two specific designs of s_{U} and s_{C} in Figs. 1 (a) and (b), respectively. As demonstrated in the visualized trajectories in Fig. 1 (b), the existence of the non-conservativeness in s_{U} incurs rotational vector fields tangent to the true score function, leading to inefficient updates during the sampling processes. In addition, the evaluation results on the *Asym*, *NAsym*, and NFE metrics further reveal that s_{U} requires more function evaluations during the sampling process than s_{C} under the same score-matching error ϵ. The above experimental evidences thus demonstrate that the non-conservativeness of s_{U} may decelerate the sampling processes.



ϵ	*Asym*	*NAsym*	NFE
0.00	0.00	0.00	2
5.00e-4	0.00	0.00	2
7.50e-4	0.00	0.00	3
1.00e-3	0.00	0.00	4
1.50e-3	0.00	0.00	4
1.75e-3	0.00	0.00	5
2.25e-3	0.00	0.00	26

ϵ	*Asym*	*NAsym*	NFE
0.00	0.00	0.00	2
5.00e-4	4.41e-7	1.77e-2	4
7.50e-4	6.60e-7	2.65e-2	5
1.00e-3	8.78e-7	3.53e-2	7
1.50e-3	1.31e-6	5.28e-2	57
1.75e-3	1.53e-6	6.15e-2	121
2.25e-3	1.96e-6	7.89e-2	238

Figure 1: The visualized examples of (a) the conservative s_C and (b) the non-conservative s_U under different choices of ϵ. The table on the right-hand side reports the results measured using NFE as well as the *Asym* and *NAsym* metrics. For a better data visualization, the vector fields are normalized with the maximum norm of s_U and s_C in each plot.

3.2 The Impacts of Architectural Flexibility on Modeling Ability and Sampling Performance

To ensure the conservative property of an SGM, previous literature (Saremi et al., 2018; Salimans & Ho, 2021) proposed to constrain the architecture such that its output vector field can be described as the gradients of a scalar-valued function. This design, however, may limit the modeling ability of an SGM. In this experiment, we examine the influence of architectural flexibility on both 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:

$$s_{\mathrm{U}}(\tilde{\boldsymbol{x}}, t; \theta_{\mathrm{U}}) = \frac{1}{\sigma_t}(\tilde{\boldsymbol{x}} - f(\tilde{\boldsymbol{x}}, t; \theta_{\mathrm{U}})), \quad s_{\mathrm{C}}(\tilde{\boldsymbol{x}}, t; \theta_{\mathrm{C}}) = -\frac{1}{2\sigma_t}\frac{\partial \left\| \tilde{\boldsymbol{x}} - f(\tilde{\boldsymbol{x}}, t; \theta_{\mathrm{C}}) \right\|^2}{\partial \tilde{\boldsymbol{x}}}, \tag{12}$$

where $f : \mathbb{R}^D \to \mathbb{R}^D$ is a neural network, and θ_{U} and θ_{C} are the parameters. The former is the USGM architecture used in NCSN (Song & Ermon, 2019), while the latter is its conservative variant explored by (Salimans & Ho, 2021). We then compare the conservativeness, the score-matching ability, the likelihood-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 score-matching ability is evaluated using $\mathcal{L}_{\mathrm{ESM}}$. The likelihood-matching ability is measured using the negative log likelihood (NLL) metric, which can be calculated based on the instantaneous change of variable formula (Chen et al., 2018). Finally, the sampling performance is evaluated using the Precision and Recall metrics (Kynkäänniemi et al., 2019), which measures the distances between the true samples and the generated samples based on k-nearest neighbor algorithm.

Table 1 reports the results of the above setting. The columns ‘Score Error’ and ‘NLL’ in Table 1 demonstrate that the USGMs consistently deliver better modeling performance in comparison to the CSGMs, suggesting that their architectural flexibility is beneficial to the training process. On the other hand, due to the potential impact of their non-conservativeness, USGMs are unable to consistently achieve superior results on the precision and recall metrics, as shown in the last two columns of Table 1. The above observations thus indicate that the architectural flexibility of a USGM may enhance its score-matching and likelihood-matching abilities. 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 sampling inefficiency and degraded sampling performance. Second, despite that CSGMs are conservative, their architectural requirement may limit their score-matching and likelihood-matching abilities in practice. 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 their

Table 1: The evaluation results of CSGMs, USGMs, and QCSGMs in terms of their means and confidence intervals of three independent runs on the '8-Gaussian,' 'Spirals,' and 'Checkerboard' datasets, which are detailed in Appendix A.5.2. The arrow symbols ↑ / ↓ indicate that higher / lower values correspond to better performance, respectively.

Dataset	Model	*Asym* (↓)	*NAsym* (↓)	Score Error (↓)	NLL (↓)	Precision (↑)	Recall (↑)
8-Gaussian	CSGM	**0.00±0.00 e-4**	**0.00±0.00 e-2**	2.49±0.00 e+1	4.86±0.00 e+0	9.78±0.00 e-1	9.31±0.00 e-1
	USGM	9.64±0.00 e-3	3.80±0.03 e-1	2.28±0.00 e+1	4.56±0.00 e+0	9.74±0.00 e-1	9.52±0.00 e-1
	QCSGM	1.52±0.00 e-3	7.16±0.00 e-2	**2.26±0.01 e+1**	**4.55±0.00 e+0**	**9.79±0.00 e-1**	**9.64±0.00 e-1**
Spirals	CSGM	**0.00±0.00 e-4**	**0.00±0.00 e-2**	3.88±0.07 e+1	5.40±0.00 e+0	4.48±0.00 e-1	9.96±0.00 e-1
	USGM	2.61±0.00 e-2	9.23±0.01 e-1	3.62±0.01 e+1	5.27±0.01 e+0	4.53±0.00 e-1	9.94±0.00 e-1
	QCSGM	6.92±0.00 e-4	8.04±0.00 e-2	**3.55±0.01 e+1**	**5.24±0.00 e+0**	**4.55±0.00 e-1**	**9.97±0.00 e-1**
Checkerboard	CSGM	**0.00±0.00 e-4**	**0.00±0.00 e-2**	3.38±0.06 e+1	5.15±0.05 e+0	8.87±0.00 e-1	9.98±0.00 e-1
	USGM	1.58±0.00 e-2	6.08±0.01 e-1	3.34±0.03 e+1	5.05±0.01 e+0	8.93±0.00 e-1	9.98±0.00 e-1
	QCSGM	6.14±0.00 e-3	3.83±0.00 e-2	**3.32±0.02 e+1**	**5.03±0.00 e+0**	**8.96±0.00 e-1**	**9.99±0.00 e-1**

conservativeness as well as sampling performance without sacrificing its modeling ability. In the next section, we elaborate on the formulation and our implementation of QCSGMs.

4 METHODOLOGY

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 detail the training procedure for QCSGMs, and discuss the time complexity of this implementation.

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:

$$\mathcal{L}_{\text{Total}}(\theta) = \mathcal{L}_{\text{SM}}(\theta) + \lambda \mathcal{L}_{\text{QC}}(\theta), \tag{13}$$

where $\mathcal{L}_{\text{SM}}$ can be any one of the score-matching objectives (i.e., Eqs. (2), (3), (5), or (6)), $\mathcal{L}_{\text{QC}}$ represents the regularization term reflecting the non-conservativeness, and λ is a balancing factor. As discussed in Section 3.1, the non-conservativeness of an SGM can be measured using the magnitude of its rotation densities in the Frobenius norm (i.e., Eq. (11)), suggesting a formulation of $\mathcal{L}_{\text{QC}}$ as:

$$\mathcal{L}_{\text{QC}}(\theta) = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|J - J^T\right\|_F^2\right], \tag{14}$$

where $J = \frac{\partial}{\partial \tilde{\boldsymbol{x}}} s(\tilde{\boldsymbol{x}};\theta)$. This objective function, however, requires D times of backpropagations to explicitly calculate the Jacobian matrix of $s(\tilde{\boldsymbol{x}};\theta)$. In order to reduce the computational cost, we first formulate an equivalent objective $\mathcal{L}_{\text{QC}}^{\text{tr}}$, and then utilize the Hutchinson trace estimator to approximate $\mathcal{L}_{\text{QC}}^{\text{tr}}$. The loss of $\mathcal{L}_{\text{QC}}^{\text{tr}}$ is derived in Appendix A.2.2, and is formulated as follows:

$$\mathcal{L}_{\text{QC}}^{\text{tr}}(\theta) = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\operatorname{tr}\left(JJ^T\right) - \operatorname{tr}\left(JJ\right)\right]. \tag{15}$$

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

$$\begin{aligned}\mathcal{L}_{\text{QC}}^{\text{est}}(\theta) &= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^T JJ^T \boldsymbol{v}\right] - \mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^T JJ\boldsymbol{v}\right]\right] \\ &= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^T JJ^T \boldsymbol{v} - \boldsymbol{v}^T JJ\boldsymbol{v}\right]\right].\end{aligned} \tag{16}$$

Eq. (16) suggests that $\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^T JJ^T \boldsymbol{v} - \boldsymbol{v}^T JJ\boldsymbol{v}\right]$ can be approximated using K random vectors $\{\boldsymbol{v}^{(i)}\}_{i=1}^K$ independently sampled from $p(\boldsymbol{v})$. Additionally, the computational graph of $\boldsymbol{v}^T JJ^T \boldsymbol{v} - \boldsymbol{v}^T JJ\boldsymbol{v}$ 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 $\mathcal{L}_{\text{QC}}^{\text{est}}$ is significantly lower than $\mathcal{L}_{\text{QC}}$ and $\mathcal{L}_{\text{QC}}^{\text{tr}}$ when $K \ll D$.

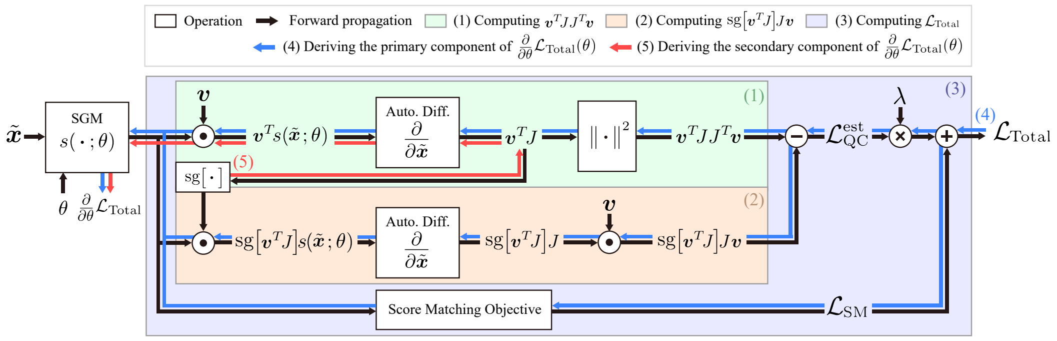

Operation
Forward propagation
(1) Computing v^TJJ^Tv
(2) Computing $\mathrm{sg}[v^TJ]Jv$
(3) Computing $\mathcal{L}_{\text{Total}}$
(4) Deriving the primary component of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$
(5) Deriving the secondary component of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$
$\tilde{x}$
SGM $s(\cdot;\theta)$
θ
$\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}$
v
$v^Ts(\tilde{x};\theta)$
Auto. Diff. $\frac{\partial}{\partial\tilde{x}}$
v^TJ
$\|\cdot\|^2$
v^TJJ^Tv
(1)
$\mathcal{L}_{\text{QC}}^{\text{est}}$
λ
(3)
(4)
$\mathcal{L}_{\text{Total}}$
$\mathrm{sg}[\cdot]$
(5)
(2)
$\mathrm{sg}[v^TJ]s(\tilde{x};\theta)$
Auto. Diff. $\frac{\partial}{\partial\tilde{x}}$
$\mathrm{sg}[v^TJ]J$
v
$\mathrm{sg}[v^TJ]Jv$
Score Matching Objective
$\mathcal{L}_{\text{SM}}$

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

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 $\mathcal{L}_{\text{Total}}(\theta)$, 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 $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$, 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. Please note that the symbol sg $[\cdot]$ used in Algorithm 1 represents the 'stop gradient' operation, which is adopted to disconnect the computational graph.

Algorithm 1 Training Procedure of QCSGM

```
Input. $\tilde{x}$, $v$, $s(\cdot\,;\theta)$, $\lambda$
    // (1) Computing $v^TJJ^Tv$.
 1: $v^TJ \leftarrow \frac{\partial}{\partial\tilde{x}}[v^Ts(\tilde{x};\theta)]$
 2: $v^TJJ^Tv \leftarrow \|v^TJ\|^2$
    // (2) Computing $v^TJJv$.
 3: $\mathrm{sg}\left[v^TJ\right]J \leftarrow \frac{\partial}{\partial\tilde{x}}[\mathrm{sg}\left[v^TJ\right]s(\tilde{x};\theta)]$
 4: $\mathrm{sg}\left[v^TJ\right]Jv \leftarrow \mathrm{sg}\left[v^TJ\right]J\cdot v$
    // (3) Computing $\mathcal{L}_{\text{Total}}(\theta)$.
 5: $\mathcal{L}_{\text{QC}}^{\text{est}}(\theta) \leftarrow \mathbb{E}_{p_\sigma(\tilde{x})p(v)}[v^TJJ^Tv - \mathrm{sg}\left[v^TJ\right]Jv]$
 6: $\mathcal{L}_{\text{SM}}(\theta) \leftarrow$ Eqs. (2), (3), (5), (6)
 7: $\mathcal{L}_{\text{Total}}(\theta) \leftarrow \mathcal{L}_{\text{SM}}(\theta) + \lambda\mathcal{L}_{\text{QC}}^{\text{est}}(\theta)$
    // (4) Deriving the primary component.
 8: Perform Backpropagation through the blue arrows.
    // (5) Deriving the secondary component.
 9: Perform Backpropagation through the red arrows.
10: Update $\theta$
```

Based on the above implementation, the computation of $\mathcal{L}_{\text{QC}}^{\text{est}}$ does not require D times of backpropagation, justifying the computational efficiency of $\mathcal{L}_{\text{QC}}^{\text{est}}$ over $\mathcal{L}_{\text{QC}}$ and $\mathcal{L}_{\text{QC}}^{\text{tr}}$. 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).

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

$\mathcal{L}_{\text{SM}}$	$\mathcal{L}_{\text{ESM}}$ $O(DHL+M)$	$\mathcal{L}_{\text{ISM}}$ $O(D^2HL)$	$\mathcal{L}_{\text{SSM}}$ $O(KDHL)$	$\mathcal{L}_{\text{DSM}}$ $O(DHL)$
$\mathcal{L}_{\text{QC}}$	$\mathcal{L}_{\text{QC}}$ $O(D^2HL)$	$\mathcal{L}_{\text{QC}}^{\text{tr}}$ $O(D^2HL)$	$\mathcal{L}_{\text{QC}}^{\text{est}}$ $O(KDHL)$	

5 Experimental Results on the Real-World Datasets

In this section, we examine the effectiveness of the proposed QCSGMs 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, and denote this method as 'U-NCSN++' in our experiments. On the other hand, C-NCSN++ and QC-NCSN++,

Table 3: The sampling performance and NFE of C-NCSN++, U-NCSN++, and QC-NCSN++ with an ODE sampler. The arrow symbols ↑ / ↓ indicate that higher / lower values correspond to better performance, respectively.

	CIFAR-10					ImageNet-32x32				
Method	NFE (↓)	FID (↓)	IS (↑)	Precision (↑)	Recall (↑)	NFE (↓)	FID (↓)	IS (↑)	Precision (↑)	Recall (↑)
C-NCSN++	442	16.43	8.19	0.5587	0.6117	422	23.91	8.64	0.5416	0.5391
U-NCSN++	170	7.48	9.24	0.6083	0.6204	148	17.09	9.80	0.5541	0.5488
QC-NCSN++	**124**	**7.21**	**9.25**	**0.6099**	**0.6205**	**115**	**16.62**	**9.85**	**0.5556**	**0.5515**
	CIFAR-100					SVHN				
Method	NFE (↓)	FID (↓)	IS (↑)	Precision (↑)	Recall (↑)	NFE (↓)	FID (↓)	IS (↑)	Precision (↑)	Recall (↑)
C-NCSN++	381	17.79	8.40	0.5650	0.6146	498	25.09	2.93	0.5314	0.5494
U-NCSN++	168	8.95	10.09	0.5890	0.6321	209	16.08	3.17	0.5553	0.6268
QC-NCSN++	**131**	**8.90**	**10.12**	**0.5903**	**0.6373**	**126**	**15.15**	**3.24**	**0.5865**	**0.6512**

Table 4: The sampling performance and NFE of C-NCSN++, U-NCSN++, and QC-NCSN++ with the PC sampler. The arrow symbols ↑ / ↓ indicate that higher / lower values correspond to better performance, respectively.

		CIFAR-10				ImageNet-32x32			
Method	NFE	FID (↓)	IS (↑)	Precision (↑)	Recall (↑)	FID (↓)	IS (↑)	Precision (↑)	Recall (↑)
U-NCSN++	500	3.77	9.41	0.6573	0.6010	24.85	9.32	0.6113	0.5014
QC-NCSN++		**3.75**	**9.52**	**0.6603**	**0.6028**	**24.78**	**9.45**	**0.6345**	**0.5048**
U-NCSN++	1000	2.50	9.58	0.6682	0.6026	19.82	9.89	0.6048	0.5202
QC-NCSN++		**2.48**	**9.70**	**0.6686**	**0.6048**	**19.62**	**9.94**	**0.6108**	**0.5216**
U-NCSN++	2000	**2.20**	9.89	0.6756	0.6035	**19.22**	10.14	0.6157	0.5186
QC-NCSN++		2.33	**9.91**	**0.6759**	**0.6036**	19.48	**10.40**	**0.6261**	**0.5245**

which are variants of U-NCSN++ constructed by Eq. (12) and regularized by $\mathcal{L}_{\text{QC}}^{\text{est}}$, are compared against U-NCSN++ using the NLL, *Asym*, *NAsym*, Fréchet Inception Distance (FID) (Heusel et al., 2017), and Inception Score (IS) (Barratt & Sharma, 2018), Precision, and Recall metrics. Please note that the details of the experimental setups are provided in Appendix A.5.3.

Table 5: The NLL, *Asym*, and *NAsym* of C-NCSN++, U-NCSN++, and QC-NCSN++ evaluated on the CIFAR-10, CIFAR-100, ImageNet-32x32, and SVHN datasets.

	CIFAR-10			ImageNet-32x32		
Method	NLL	*Asym*	*NAsym*	NLL	*Asym*	*NAsym*
C-NCSN++	3.89	**0.00**	**0.00**	4.26	**0.00**	**0.00**
U-NCSN++	3.46	1.88 e8	1.90 e-3	3.96	2.05 e7	7.17 e-4
QC-NCSN++	**3.38**	5.03 e7	1.10 e-3	**3.83**	1.13 e7	5.47 e-4
	CIFAR-100			SVHN		
Method	NLL	*Asym*	*NAsym*	NLL	*Asym*	*NAsym*
C-NCSN++	3.69	**0.00**	**0.00**	2.74	**0.00**	**0.00**
U-NCSN++	3.50	2.98 e8	2.25 e-3	2.15	3.06 e7	6.54 e-4
QC-NCSN++	**3.44**	9.31 e7	1.44 e-3	**2.01**	1.69 e7	4.80 e-4

Likelihood and Conservativeness Evaluation. Table 5 reports the evaluation results of U-NCSN++, C-NCSN++, and QC-NCSN++ in terms of NLL, *Asym*, and *NAsym* on the four real-world datasets. The evaluation results of C-NCSN++ is inferior to those of U-NCSN++ and QC-NCSN++ on the NLL metric, which aligns with our observation in Section 3, suggesting that the modeling flexibility is influential to the final performance on the NLL metric. In addition, we observe that the evaluation results on the NLL metric can be further improved when $\mathcal{L}_{\text{QC}}^{\text{est}}$ is incorporated into the training process. As demonstrated in the table, QC-NCSN++, which can achieve superior performance in terms of *Asym* and *NAsym* metrics, also has improved results on the NLL metric.

Sampling with an ODE Solver. In this experiment, we examine the sampling performance and efficiency of U-NCSN++, C-NCSN++, and QC-NCSN++ based on NFE and the FID/IS/Precision/Recall metrics. The sampler is implemented using the RK45 (Dormand & Prince, 1980) ODE solver, and NFE here refers to the number of function evaluations performed during the sampling process. Table 3 presents the evaluation results of the above setting. It is observed that C-NCSN++ is inferior to U-NCSN++ and QC-NCSN++, suggesting that modeling errors (i.e., NLL) can be influential to the sampling performance. On the other hand, QC-NCSN++ is able to outperform U-NCSN++ in terms of the sampling performance metrics with fewer function evaluations, indicating that QC-NCSN++ is able to deliver a better sampling efficiency. The above experimental results thus demonstrate the effectiveness of the proposed $\mathcal{L}_{\text{QC}}^{\text{est}}$.

Sampling under a Fixed NFE. In this experiment, we further compare the sampling performance of U-NCSN++ and QC-NCSN++ under fixed NFE using the Predictor-Corrector (PC) sampler (Song

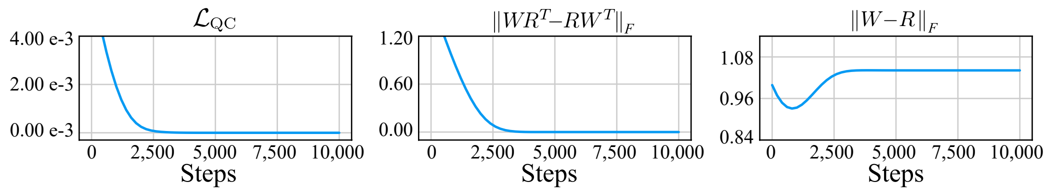


Figure 3: The trends of $\left\|WR^T - RW^T\right\|_F$ and $\|W - R\|_F$ during the minimization process of $\mathcal{L}_{\mathrm{QC}}$. The 'steps' on the x-axes refer to the training steps.

et al., 2021b). Different from the ODE sampler presented above, PC sampler discretizes the sampling process described Eq. (8) with equal-sized steps according to a predetermined value of NFE. This design allows us to control NFE in the sampling process and compare the sampling performance under a fixed NFE. Table A8 presents the evaluation results of U-NCSN++ and QC-NCSN++ when NFE is equal to 500, 1,000, and 2,000. It is observed that QC-NCSN++ can outperform U-NCSN++ in terms of the FID/IS/Precion/Recall metrics when NFE is equal to 500, 1,000. Nonetheless, QC-NCSN++ has inferior FID results when NFE is equal to 2,000. The results suggest that the impact of the non-conservativeness may also be influenced by the sampling step size. A larger step size, which corresponds to a smaller value of NFE, is able to magnify the effect of the rotational vector field incurred by approximation errors.

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 $s(\tilde{\boldsymbol{x}};\theta) = Rh(W^T\tilde{\boldsymbol{x}} + \boldsymbol{b}) + \boldsymbol{c}$, where $h(\cdot)$ is an activation function, $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^D$, $R, W \in \mathbb{R}^{D\times H}$ are the weights of $s(\cdot\,;\theta)$ (i.e., $\theta = \{R, W, \boldsymbol{b}, \boldsymbol{c}\}$), 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 $\mathcal{L}_{\mathrm{QC}}$. Fig. 3 depicts the trends of $\left\|WR^T - RW^T\right\|_F$ and $\|W - R\|_F$ during the minimization process of $\mathcal{L}_{\mathrm{QC}}$. As the training progresses, the values of $\left\|WR^T - RW^T\right\|_F$ approach zero, indicating that $s(\cdot\,;\theta)$ learns to output a conservative vector field through minimizing $\mathcal{L}_{\mathrm{QC}}$. In contrast, the values of $\|W - R\|_F$ do not decrease to zero, revealing that minimizing $\mathcal{L}_{\mathrm{QC}}$ 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.

REFERENCES

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`.

R. TQ Chen, Y. Rubanova, J. Bettencourt, and D. K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

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.

J. R. Dormand and P. J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6:19–26, 1980.

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 *Thirtieth AAAI Conf. on Artificial Intelligence (AAAI)*, 2016.

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

H. Kamyshanska 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. Kynkäänniemi, 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=L7wzpQttNO`.

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: bj/1178291835. URL `https://doi.org/`.

T. 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`.

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

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`.

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`.

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.

Y. Xu, Z. Liu, M. Tegmark, and T. Jaakkola. Poisson flow generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

A 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 derivation for (15) in Section A.2. Then, we offer a discussion on the normalized asymmetry metric as well as the detailed setups for the motivational experiments 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++ 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.
α	the step size used in Langevin dynamics.
ϵ	the score-matching error described in Section 3.1.
σ	the standard deviation for Gaussian distribution.
θ	the parameters of an SGM.
$\boldsymbol{x} \in \mathbb{R}^D$	a data sample.
$\tilde{\boldsymbol{x}} \in \mathbb{R}^D$	a perturbed data sample.
$\boldsymbol{z} \in \mathbb{R}^D$	a noise vector used in Langevin dynamics.
$\boldsymbol{v} \in \mathbb{R}^D$	a random vector used in the Hutchinson trace estimator.
$\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^D$	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.
$\{\boldsymbol{x}^{(i)}\}_{i=1}^M$	a dataset.
$\{\tilde{\boldsymbol{x}}_t\}_{t=1}^T$	a set of discretized timesteps in Langevin dynamics.
$\{\tilde{\boldsymbol{x}}_t^{(i)}\}_{i=1}^N$	a set of discretized points of the t-th timestep used for estimating line integral.
$\{\boldsymbol{v}^{(i)}\}_{i=1}^K$	a set of random vectors drawn from $p(\boldsymbol{v})$.
p_{data}	the unknown true probability density function (pdf).
$p_0(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^m \delta(\Vert\boldsymbol{x} - \boldsymbol{x}^{(i)}\Vert)$	the empirical distribution of a dataset.
$p_\sigma(\tilde{\boldsymbol{x}}\vert\boldsymbol{x}) = \frac{1}{(2\pi)^{D/2}\sigma^D} e^{\frac{-1}{2\sigma^2}\Vert\tilde{\boldsymbol{x}}-\boldsymbol{x}\Vert^2}$	the smoothing kernel with mean $\boldsymbol{x}$ and standard deviation σ used in Parzen density estimator.
$p_\sigma(\tilde{\boldsymbol{x}}) = \int_{\boldsymbol{x}} p_\sigma(\tilde{\boldsymbol{x}}\vert\boldsymbol{x}) p_0(\boldsymbol{x}) d\boldsymbol{x}$	Parzen density estimator for $p_0(\boldsymbol{x})$.
$p(\boldsymbol{v})$	a distribution satisfying $\mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{v}^T] = I$ such as a Gaussian or a Radamacher distribution.

Table A1: The list of symbols used in this paper.

$Z(\theta)$	a partition function of a Boltzmann distribution.
$E(\cdot\,;\theta):\mathbb{R}^D\to\mathbb{R}$	an energy model parameterized by θ.
$s(\cdot\,;\theta):\mathbb{R}^D\to\mathbb{R}^D$	a score model parameterized by θ.
$\frac{\partial}{\partial\tilde{\boldsymbol{x}}}\log p_\sigma(\tilde{\boldsymbol{x}})$	the gradient of $\log p_\sigma(\tilde{\boldsymbol{x}})$ w.r.t. $\tilde{\boldsymbol{x}}$.
$\frac{\partial}{\partial\tilde{\boldsymbol{x}}}E(\tilde{\boldsymbol{x}};\theta)$	the gradient of $E(\tilde{\boldsymbol{x}};\theta)$ w.r.t. $\tilde{\boldsymbol{x}}$.
$\frac{\partial}{\partial\tilde{\boldsymbol{x}}}s(\tilde{\boldsymbol{x}};\theta)$	the Jacobian matrix of $s(\tilde{\boldsymbol{x}};\theta)$.
J	the simplified notation for $\frac{\partial}{\partial\tilde{\boldsymbol{x}}}s(\tilde{\boldsymbol{x}};\theta)$.
$\mathcal{L}_{\text{ESM}}$	Explicit Score Matching (ESM) loss defined in Eq. (2).
$\mathcal{L}_{\text{ISM}}$	Implicit Score Matching loss (ISM) loss defined in Eq. (3).
$\mathcal{L}_{\text{SSM}}$	Sliced Score Matching loss (SSM) loss defined in Eq. (5).
$\mathcal{L}_{\text{DSM}}$	Denoising Score Matching loss (DSM) loss defined in Eq. (6).
$\mathcal{L}_{\text{Total}}$	the total loss of QCSGMs defined in Eq. (13).
$\mathcal{L}_{\text{QC}}$	the proposed regularization loss defined in Eq. (14).
$\mathcal{L}_{\text{QC}}^{\text{tr}}$	the equivalent variant of $\mathcal{L}_{\text{QC}}$ defined in Eq. (15).
$\mathcal{L}_{\text{QC}}^{\text{est}}$	the approximated variant of $\mathcal{L}_{\text{QC}}^{\text{tr}}$ defined in Eq. (16).
$\boldsymbol{u}^T\boldsymbol{v}=\boldsymbol{u}\cdot\boldsymbol{v}=\sum_i\boldsymbol{u}_i\boldsymbol{v}_i$	inner product between two vectors $\boldsymbol{u}$, $\boldsymbol{v}$.
$\text{tr}\,(A)=\sum_i A_{i,i}$	trace of a matrix A.
$\lVert\boldsymbol{u}\rVert=\sqrt{\sum_i\boldsymbol{u}_i^2}$	Euclidean norm of a vector $\boldsymbol{u}$.
$\lVert A\rVert_F=\sqrt{\sum_{i,j}A_{i,j}^2}$	Frobenius norm of a matrix A.
$\exp\,(\cdot)$	an exponential function.
$\text{sg}\,[\cdot]$	a stop gradient operator.

Table A2: The list of symbols used in this paper (cont.).

A.2 Derivations

A.2.1 The Equivalence between Zero-rotation-density Condition and the Conservativeness of a Score Model

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

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

$$\begin{aligned}&\overline{\text{ROT}}_{ij}s(\tilde{\boldsymbol{x}};\theta)=0,\ \forall\,1\le i,j\le D\\ \Leftrightarrow\ &\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial\tilde{\boldsymbol{x}}_j}-\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial\tilde{\boldsymbol{x}}_i}=0,\ \forall\,1\le i,j\le D\\ \Leftrightarrow\ &J_{ij}-J_{ji}=0,\ \forall\,1\le i,j\le D\\ \Leftrightarrow\ &J_{ij}=J_{ji},\ \forall\,1\le i,j\le D\\ \Leftrightarrow\ &J\text{ is symmetric.}\end{aligned}$$

□

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

Proposition 1. $\overline{\text{ROT}}_{ij}s(\tilde{\boldsymbol{x}})=0$ *for all* $1\le i,j\le D$ *if and only if* $s(\tilde{\boldsymbol{x}})$ *is conservative.*

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

A.2.2 The Derivation of $\mathcal{L}^{\text{tr}}_{\text{QC}}$ in Eq. (15)

In Section 4.1, we derived the computationally efficient objective $\mathcal{L}^{\text{est}}_{\text{QC}}$ based on the assumption that $\mathcal{L}_{\text{QC}}$ equals $\mathcal{L}^{\text{tr}}_{\text{QC}}$. To show that the equivalence holds, we provide a formal derivation as follows.

Proposition 2. $\mathcal{L}_{\text{QC}}(\theta) = \mathcal{L}^{\text{tr}}_{\text{QC}}(\theta)$.

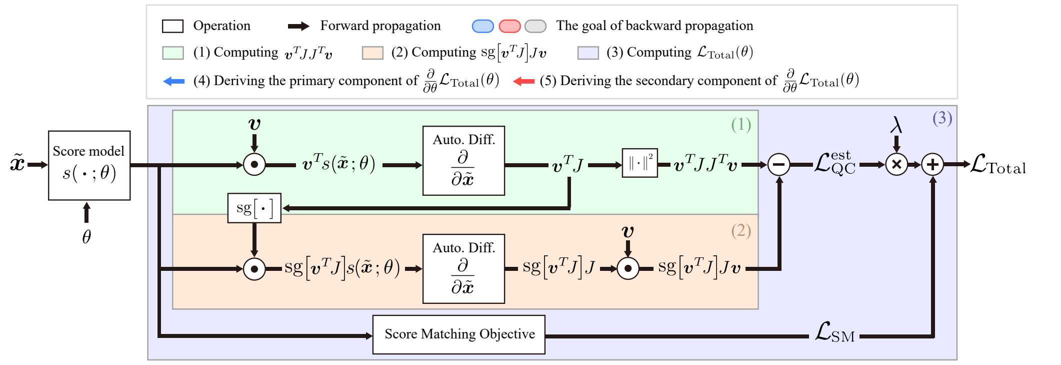
Proof.

$$
\begin{aligned}
\mathcal{L}_{\text{QC}}(\theta) &= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|J - J^T\right\|_F^2\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}}\right) - \left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}}\right)^T\right\|_F^2\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j} - \frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i}\right)^2\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\sum_{i=1}^D\sum_{j=1}^D\left(\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\right)^2 + \left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i}\right)^2 - 2\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i}\right)\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\right)^2 + \frac{1}{2}\sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i}\right)^2 - \frac{1}{2}\cdot 2\sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i}\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\cdot 2\sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\right)^2 - \frac{1}{2}\cdot 2\sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i}\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\right)^2 - \sum_{i=1}^D\sum_{j=1}^D\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_i}{\partial \tilde{\boldsymbol{x}}_j}\frac{\partial s(\tilde{\boldsymbol{x}};\theta)_j}{\partial \tilde{\boldsymbol{x}}_i}\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\operatorname{tr}\left(\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}}\right)\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}}\right)^T\right) - \operatorname{tr}\left(\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}}\right)\left(\frac{\partial s(\tilde{\boldsymbol{x}};\theta)}{\partial \tilde{\boldsymbol{x}}}\right)\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\operatorname{tr}\left(JJ^T\right) - \operatorname{tr}\left(JJ\right)\right] \\
&= \mathcal{L}^{\text{tr}}_{\text{QC}}(\theta)
\end{aligned}
$$

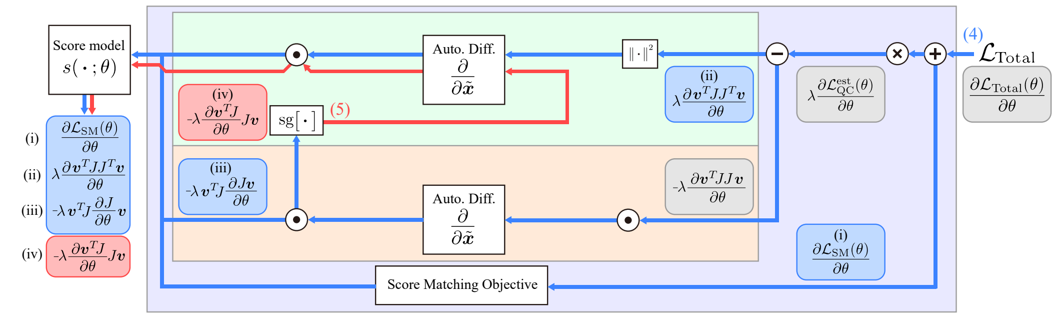
□

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

$$
\begin{aligned}
\mathcal{L}_{\text{QC}}(\theta) &= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|J - J^T\right\|_F^2\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\operatorname{tr}\left((J - J^T)^T(J - J^T)\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\operatorname{tr}\left((J^T - J)(J - J^T)\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\operatorname{tr}\left(J^TJ - J^TJ^T + JJ^T - JJ\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left(\operatorname{tr}\left(J^TJ\right) - \operatorname{tr}\left(J^TJ^T\right) + \operatorname{tr}\left(JJ^T\right) - \operatorname{tr}\left(JJ\right)\right)\right] \\
&= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\operatorname{tr}\left(JJ^T\right) - \operatorname{tr}\left(JJ\right)\right] \\
&= \mathcal{L}^{\text{tr}}_{\text{QC}}(\theta)
\end{aligned}
$$


Operation
Forward propagation
The goal of backward propagation
(1) Computing v^TJJ^Tv
(2) Computing $\mathrm{sg}[v^TJ]Jv$
(3) Computing $\mathcal{L}_{\text{Total}}(\theta)$
(4) Deriving the primary component of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$
(5) Deriving the secondary component of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$
Score model $s(\cdot;\theta)$
Auto. Diff. $\frac{\partial}{\partial\tilde{x}}$
Score Matching Objective
$\mathcal{L}_{\text{SM}}$
$\mathcal{L}^{\text{est}}_{\text{QC}}$
$\mathcal{L}_{\text{Total}}$

(a) Forward Propagation (Steps (1)~(3))


Score model $s(\cdot;\theta)$
(i) $\frac{\partial\mathcal{L}_{\text{SM}}(\theta)}{\partial\theta}$
(ii) $\lambda\frac{\partial v^TJJ^Tv}{\partial\theta}$
(iii) $-\lambda v^TJ\frac{\partial J}{\partial\theta}v$
(iv) $-\lambda\frac{\partial v^TJ}{\partial\theta}Jv$
$\lambda\frac{\partial\mathcal{L}^{\text{est}}_{\text{QC}}(\theta)}{\partial\theta}$
$\frac{\partial\mathcal{L}_{\text{Total}}(\theta)}{\partial\theta}$
$-\lambda\frac{\partial v^TJJv}{\partial\theta}$
(4)
(5)
Score Matching Objective

(b) Backward Propagation (Steps (4) and (5))

Figure A1: The computational graphs of $\mathcal{L}_{\text{Total}}$ in QCSGMs. The upper and lower subplots depict the forward and backward propagation processes, respectively. The 'Auto. Diff.' blocks represent the operation of differentiating $\boldsymbol{u}^T s(\tilde{\boldsymbol{x}};\theta)$, where $\boldsymbol{u}$ is a constant vector with respect to $\tilde{\boldsymbol{x}}$.

A.2.3 A Detailed Description of the Training Process

The entire training procedure is divided into five steps, denoted as Steps (1)~(5), respectively. Steps (1)~(3) describe the forward propagation process of $\mathcal{L}_{\text{Total}}(\theta)$, which is depicted by the black arrows in Fig. A1 (a). Steps (4) and (5) correspond to the backpropagation processes of the two gradient components comprising $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$, which are depicted in Fig. A1 (b). In the following paragraphs, we elaborate on the details of Steps (1)~(5).

(1) Computing $\boldsymbol{v}^TJJ^T\boldsymbol{v}$. First, $\boldsymbol{v}^TJ$ is computed by performing backpropagation of $\boldsymbol{v}^Ts(\tilde{\boldsymbol{x}};\theta)$ with respect to $\tilde{\boldsymbol{x}}$ via automatic differentiation, which is depicted as the upper 'Auto. Diff.' block in Fig. A1 (a). Then, $\boldsymbol{v}^TJJ^T\boldsymbol{v}$ is calculated by taking the squared L2 norm on $\boldsymbol{v}^TJ$ according to the relationship: $\left\|\boldsymbol{v}^TJ\right\|^2 = \boldsymbol{v}^TJ(\boldsymbol{v}^TJ)^T = \boldsymbol{v}^TJJ^T\boldsymbol{v}$.

(2) Computing $\boldsymbol{v}^TJJ\boldsymbol{v}$. First, $\mathrm{sg}\left[\boldsymbol{v}^TJ\right]s(\tilde{\boldsymbol{x}};\theta)$ is calculated by taking the inner product between $\mathrm{sg}\left[\boldsymbol{v}^TJ\right]$ and $s(\tilde{\boldsymbol{x}};\theta)$, where the stop-gradient operator $\mathrm{sg}\left[\cdot\right]$ is applied to $\boldsymbol{v}^TJ$ to detach it from the computational graph built in Step (1). Then, $\mathrm{sg}\left[\boldsymbol{v}^TJ\right]J$ is calculated by differentiating $\mathrm{sg}\left[\boldsymbol{v}^TJ\right]s(\tilde{\boldsymbol{x}};\theta)$ via performing backpropagation. Stopping the gradient of $\boldsymbol{v}^TJ$ is necessary to ensure that the automatic differentiation (i.e., the lower 'Auto. Diff.' block in Fig. A1 (a)) excludes the computational graph used for differentiating $\boldsymbol{v}^TJ$, allowing $\boldsymbol{v}^TJJ$ to be correctly derived. Lastly, $\mathrm{sg}\left[\boldsymbol{v}^TJ\right]J\boldsymbol{v}$ is obtained by taking the inner product of $\mathrm{sg}\left[\boldsymbol{v}^TJ\right]J$ and $\boldsymbol{v}$.

(3) Computing $\mathcal{L}_{\text{Total}}(\theta)$. Based on the results of Steps (1) and (2), $\mathcal{L}^{\text{est}}_{\text{QC}}(\theta)$ is computed by taking the expectation of $(\boldsymbol{v}^TJJ^T\boldsymbol{v} - \mathrm{sg}\left[\boldsymbol{v}^TJ\right]J\boldsymbol{v})$. 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 $\lambda\mathcal{L}^{\text{est}}_{\text{QC}}(\theta)$, as described in Eq. (13).

(4) Deriving the primary component of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$. Based on the computational graph built in Steps (1)~(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. A1 (b) 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. A1 (b) 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.

The Derivation of the primary and secondary components of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$. In Steps (4) and (5), we decompose $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$ as the primary and secondary components, and separately derive them. 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 $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$ can be decomposed as the sum of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{SM}}(\theta)$, $\lambda\frac{\partial}{\partial\theta}\boldsymbol{v}^TJJ^T\boldsymbol{v}$, $-\lambda(\frac{\partial}{\partial\theta}\boldsymbol{v}^TJ)J\boldsymbol{v}$, and $-\lambda\boldsymbol{v}^TJ(\frac{\partial}{\partial\theta}J\boldsymbol{v})$, indexed as (i)$\sim$(iv) respectively. The derivation is shown as the following:

$$\begin{aligned}
\frac{\partial\mathcal{L}_{\text{Total}}(\theta)}{\partial\theta} &= \frac{\partial(\mathcal{L}_{\text{SM}}(\theta)+\lambda\mathcal{L}^{\text{est}}_{\text{QC}}(\theta))}{\partial\theta}\\
&= \frac{\partial\mathcal{L}_{\text{SM}}(\theta)}{\partial\theta}+\lambda\frac{\partial\mathcal{L}^{\text{est}}_{\text{QC}}(\theta)}{\partial\theta}\\
&= \frac{\partial\mathcal{L}_{\text{SM}}(\theta)}{\partial\theta}+\lambda\frac{\partial(\boldsymbol{v}^TJJ^T\boldsymbol{v}-\boldsymbol{v}^TJJ\boldsymbol{v})}{\partial\theta}\\
&= \frac{\partial\mathcal{L}_{\text{SM}}(\theta)}{\partial\theta}+\lambda\frac{\partial\boldsymbol{v}^TJJ^T\boldsymbol{v}}{\partial\theta}-\lambda\frac{\partial\boldsymbol{v}^TJJ\boldsymbol{v}}{\partial\theta}\\
&= \underbrace{\frac{\partial\mathcal{L}_{\text{SM}}(\theta)}{\partial\theta}}_{\text{(i)}}+\underbrace{\lambda\frac{\partial\boldsymbol{v}^TJJ^T\boldsymbol{v}}{\partial\theta}}_{\text{(ii)}}+\underbrace{(-\lambda)\boldsymbol{v}^TJ\frac{\partial J\boldsymbol{v}}{\partial\theta}}_{\text{(iii)}}+\underbrace{(-\lambda)\frac{\partial\boldsymbol{v}^TJ}{\partial\theta}J\boldsymbol{v}}_{\text{(iv)}}.
\end{aligned}$$

We name the sum of (i)$\sim$(iii) the *primary* component of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$, and the term (iv) the *secondary* component of $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$. Such a decomposition suggests that $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$ can be separately computed based on the computational graph built in Steps (1)$\sim$(3) as shown in the upper subplot of Fig. A1. For the primary component, the sum of (i)$\sim$(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, $\frac{\partial}{\partial\theta}\mathcal{L}_{\text{Total}}(\theta)$ can be correctly derived.

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.

Derivation of the *NAsym* Metric. As described in (Andrilli & Hecker, 2016), any matrix A can be uniquely decomposed into a symmetric matrix A_{sym} and a skew-symmetric matrix A_{skew} as follows:

$$A = A_{\text{sym}} + A_{\text{skew}} = \frac{A+A^T}{2}+\frac{A-A^T}{2}. \tag{A1}$$

Based on Eq. (A1), the Jacobian J of an SGM $s(\tilde{\boldsymbol{x}};\theta)$ 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 *NAsym* metric introduced in Section 3.1 can be formulated as follows:

$$\textit{NAsym} = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{\|J_{\text{skew}}\|_F^2}{\|J\|_F^2}\right] = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{\left\|\frac{1}{2}(J-J^T)\right\|_F^2}{\|J\|_F^2}\right] = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{4}\frac{\|J-J^T\|_F^2}{\|J\|_F^2}\right]. \tag{A2}$$

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 $[0,1]$. *NAsym* $= 1$ corresponds to the condition where J_{skew} dominates J, implying that J is skew-symmetric. On the contrary, *NAsym* $= 0$ indicates that J only contains the symmetric component J_{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 $s(\tilde{\boldsymbol{x}};\theta)$, i.e., $\|J_{\text{skew}}\|_F^2 = \left\|(J-J^T)/2\right\|_F^2 = \frac{1}{4}\sum_{i,j=1}^{D}(\frac{\partial}{\partial\tilde{\boldsymbol{x}}_j}s(\tilde{\boldsymbol{x}};\theta)_i - \frac{\partial}{\partial\tilde{\boldsymbol{x}}_i}s(\tilde{\boldsymbol{x}};\theta)_j)^2 = \frac{1}{4}\sum_{i,j=1}^{D}\left(\overline{\text{ROT}}_{ij}s(\tilde{\boldsymbol{x}};\theta)\right)^2$, *NAsym* can be adopted to measure the non-conservativeness of $s(\tilde{\boldsymbol{x}};\theta)$, as mentioned in Section 3.1.

An Efficient Implementation of *NAsym*. Since Eq. (A2) involves the explicit calculation of the Jacobian matrix J, evaluating the *NAsym* metric for a single instance $\tilde{\boldsymbol{x}}$ 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 the *NAsym* metric based on the following derivation:

$$\begin{aligned} \textit{NAsym} &= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{4}\frac{\left\|J-J^T\right\|_F^2}{\|J\|_F^2}\right] = \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\frac{\operatorname{tr}\left(JJ^T\right)-\operatorname{tr}\left(JJ\right)}{\operatorname{tr}\left(JJ^T\right)}\right] \\ &= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\frac{\mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}^TJJ^T\boldsymbol{v}] - \mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}^TJJ\boldsymbol{v}]}{\mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}^TJJ^T\boldsymbol{v}]}\right] \\ &= \mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\mathbb{E}_{p(\boldsymbol{v})}\left[\frac{\boldsymbol{v}^TJJ^T\boldsymbol{v}-\boldsymbol{v}^TJJ\boldsymbol{v}}{\boldsymbol{v}^TJJ^T\boldsymbol{v}}\right]\right], \end{aligned} \tag{A3}$$

where $p(\boldsymbol{v})$ satisfies $\mathbb{E}_{p(\boldsymbol{v})}[\boldsymbol{v}\boldsymbol{v}^T] = I$. The expectation $\mathbb{E}_{p(\boldsymbol{v})}[\cdot]$ can be approximated using K random vectors. In addition, the terms $\boldsymbol{v}^TJJ^T\boldsymbol{v}$ and $\boldsymbol{v}^TJJ\boldsymbol{v}$ in Eq. (A3) can be efficiently calculated based on Steps (1) and (2) described in Section 4.2. This suggests that the computational cost of evaluating *NAsym* can be significantly reduced when $K \ll D$. In Section A.5.3, we describe an approach to measure *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++ adopts the variance exploding (VE) diffusion process identical to that employed in NCSN++ (VE) (Song et al., 2021b), which is a time-inhomogeneous sampling algorithm. In this sampling algorithm, SGM and step size are respectively represented as $s(\cdot\,;\theta,\sigma_t)$ and $\alpha_t = \frac{\partial}{\partial t}\sigma_t^2$, where σ_t is a time-dependent standard deviation. In C-NCSN++, U-NCSN++, and QC-NCSN++, σ_t is set to $\sigma_{\min}(\sigma_{\min}/\sigma_{\max})^{\frac{t}{T}}$ (Song et al., 2021b), where T is the total number of timesteps in the sampling process, $\sigma_{\min}$ is a constant representing a minimal noise scale, and $\sigma_{\max}$ 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 σ_t, the training objectives of $s(\cdot\,;\theta,\sigma_t)$ have to be modified accordingly. For example, the score-matching objective $\mathcal{L}_{\text{DSM}}$ used in C-NCSN++, U-NCSN++, and QC-NCSN++ is modified as follows:

$$\mathbb{E}_{\mathcal{U}(t)}\left[\lambda(t)\mathbb{E}_{p_{\sigma_t}(\tilde{\boldsymbol{x}}|\boldsymbol{x})p_0(\boldsymbol{x})}\left[\left\|s(\tilde{\boldsymbol{x}};\theta,\sigma_t) - \frac{\partial\log p_{\sigma_t}(\tilde{\boldsymbol{x}}|\boldsymbol{x})}{\partial\tilde{\boldsymbol{x}}}\right\|^2\right]\right], \tag{A4}$$

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 $\mathcal{L}_{\text{QC}}^{\text{est}}$ used in QC-NCSN++ is adjusted according to $\lambda(t)$, which is formulated as follows:

$$\mathbb{E}_{\mathcal{U}(t)}\left[\lambda(t)\mathbb{E}_{p_{\sigma_t}(\tilde{\boldsymbol{x}})}\left[\mathbb{E}_{p(\boldsymbol{v})}\left[\boldsymbol{v}^TJJ^T\boldsymbol{v} - \boldsymbol{v}^TJJ\boldsymbol{v}\right]\right]\right], \tag{A5}$$

where $J = \frac{\partial}{\partial\tilde{\boldsymbol{x}}}s(\tilde{\boldsymbol{x}};\theta,\sigma_t)$.

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. The code implementation for the experiments is provided in the following anonymous repository: `https://anonymous.4open.science/r/qcsgm-review-B4FF/README.md`.

A.5.1 Experimental Setups for Section 3.1

In Section 3.1, we compare the sampling efficiency of a USGM and a CSGM with an approximation error ϵ. These SGMs are formulated based on the following equation:

$$s(\tilde{\boldsymbol{x}}) = \frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}}) + \sqrt{\frac{2\epsilon\mu(\tilde{\boldsymbol{x}})}{\|\tilde{\boldsymbol{x}}\|^2 p_\sigma(\tilde{\boldsymbol{x}})}}\boldsymbol{u}(\tilde{\boldsymbol{x}}), \tag{A6}$$

where p_σ is the target distribution, μ is an arbitrary distribution, and $\boldsymbol{u}(\tilde{\boldsymbol{x}}) \in \mathbb{R}^D$ is a vector function with its norm equal to the norm of its input (i.e., $\|\boldsymbol{u}(\tilde{\boldsymbol{x}})\| = \|\tilde{\boldsymbol{x}}\|$). To show that $s(\tilde{\boldsymbol{x}})$ in Eq. (A6) satisfies $\mathcal{L}_{\text{ESM}} = \epsilon$, we provide the following proposition.

Proposition 3. *Given $\epsilon > 0$, a target distribution p_σ, and an arbitrary pdf μ, s defined in Eq. (A6) satisfies $\mathcal{L}_{\text{ESM}} = \epsilon$.*

Proof.

$$\begin{aligned}
\mathcal{L}_{\text{ESM}} &= \int_{\tilde{\boldsymbol{x}}} p_\sigma(\tilde{\boldsymbol{x}}) \frac{1}{2} \left\| s(\tilde{\boldsymbol{x}}) - \frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}}) \right\|^2 d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} p_\sigma(\tilde{\boldsymbol{x}}) \frac{1}{2} \left\| \frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}}) + \sqrt{\frac{2\epsilon\mu(\tilde{\boldsymbol{x}})}{\|\tilde{\boldsymbol{x}}\|^2 p_\sigma(\tilde{\boldsymbol{x}})}}\boldsymbol{u}(\tilde{\boldsymbol{x}}) - \frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}}) \right\|^2 d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} p_\sigma(\tilde{\boldsymbol{x}}) \frac{1}{2} \left\| \sqrt{\frac{2\epsilon\mu(\tilde{\boldsymbol{x}})}{\|\tilde{\boldsymbol{x}}\|^2 p_\sigma(\tilde{\boldsymbol{x}})}}\boldsymbol{u}(\tilde{\boldsymbol{x}}) \right\|^2 d\tilde{\boldsymbol{x}} = \int_{\tilde{\boldsymbol{x}}} p_\sigma(\tilde{\boldsymbol{x}}) \frac{1}{2} \frac{2\epsilon\mu(\tilde{\boldsymbol{x}})}{\|\tilde{\boldsymbol{x}}\|^2 p_\sigma(\tilde{\boldsymbol{x}})} \|\boldsymbol{u}(\tilde{\boldsymbol{x}})\|^2 d\tilde{\boldsymbol{x}} \\
&= \int_{\tilde{\boldsymbol{x}}} \mu(\tilde{\boldsymbol{x}}) \frac{\epsilon \|\boldsymbol{u}(\tilde{\boldsymbol{x}})\|^2}{\|\tilde{\boldsymbol{x}}\|^2} d\tilde{\boldsymbol{x}} = \int_{\tilde{\boldsymbol{x}}} \mu(\tilde{\boldsymbol{x}}) \epsilon d\tilde{\boldsymbol{x}} = \epsilon
\end{aligned}$$

□

In the motivational example presented in Section 3.1, we choose $p_\sigma = \mathcal{N}(0; \sigma^2 I)$, and select $\mu = \frac{1}{10}\sum_{i=1}^{10} \mathcal{N}([3\cos(\frac{2i\pi}{10}), 3\sin(\frac{2i\pi}{10})]^T; I)$. We consider $\boldsymbol{u}(\tilde{\boldsymbol{x}}) = [-\tilde{x}_2, \tilde{x}_1]^T$ for s_{U}, and $\boldsymbol{u}(\tilde{\boldsymbol{x}}) = [\tilde{x}_1, \tilde{x}_2]^T$ for s_{C}, where $\tilde{\boldsymbol{x}} = [\tilde{x}_1, \tilde{x}_2]^T$. In particular, $[-\tilde{x}_2, \tilde{x}_1]^T$ is a rotational vector field with each vector tangent to the true score function $\frac{\partial}{\partial \tilde{\boldsymbol{x}}} \log p_\sigma(\tilde{\boldsymbol{x}}) = -1/\sigma^2[\tilde{x}_1, \tilde{x}_2]^T$. On the other hand, $[\tilde{x}_1, \tilde{x}_2]^T$ is a vector field with each vector pointing to the opposite direction against the true score function. For an illustrative purpose, we leverage a deterministic variant of Eq. (7) (i.e., $\tilde{\boldsymbol{x}}_{t+1} = \tilde{\boldsymbol{x}}_t + \alpha_t s(\tilde{\boldsymbol{x}}_t)$) as our sampler to generate samples based on s_{U} and s_{C}, and calculate the NFE required for all samples to move to the center of p_σ.

A.5.2 Experimental Setups for the Motivational Examples Presented in Section 3.2

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 $(\cos(\frac{\pi w}{4}), \sin(\frac{\pi w}{4}))$, where $w \in \{1, ..., 8\}$. 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 $(4w - 2, t - 2s + \lfloor w \rfloor \bmod 2)$, where $w \in [0, 1]$, $s \in \{0, 1\}$, $\lfloor \cdot \rfloor$ is a floor function, and mod represents the modulo operation. For all of these three datasets, $p_0(\boldsymbol{x})$ 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). The SGMs s_{U} and s_{C} are trained utilizing the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 1×10^{-2} and a batch size of 5,000 for 100,000 iterations. The balancing factor λ is fixed to 0.1. The maximal and minimal noise scales $\sigma_{\max}$ and $\sigma_{\min}$ are set to 10 and 0.01, respectively.

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

$$\mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\left\|s(\tilde{\boldsymbol{x}};\theta) - \frac{\partial \log p_\sigma(\tilde{\boldsymbol{x}})}{\partial \tilde{\boldsymbol{x}}}\right\|^2\right] \approx \sum_{\tilde{\boldsymbol{x}}\in\mathcal{D}} p_\sigma(\tilde{\boldsymbol{x}}) \left\|s(\tilde{\boldsymbol{x}};\theta) - \frac{\partial \log p_\sigma(\tilde{\boldsymbol{x}})}{\partial \tilde{\boldsymbol{x}}}\right\|^2, \tag{A7}$$

$$\mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|J - J^T\right\|_F^2\right] \approx \sum_{\tilde{\boldsymbol{x}}\in\mathcal{D}} p_\sigma(\tilde{\boldsymbol{x}}) \frac{1}{2}\left\|J - J^T\right\|_F^2, \tag{A8}$$

$$\mathbb{E}_{p_\sigma(\tilde{\boldsymbol{x}})}\left[\frac{1}{4}\frac{\left\|J - J^T\right\|_F^2}{\|J\|_F^2}\right] \approx \sum_{\tilde{\boldsymbol{x}}\in\mathcal{D}} p_\sigma(\tilde{\boldsymbol{x}}) \frac{1}{4}\frac{\left\|J - J^T\right\|_F^2}{\|J\|_F^2}, \tag{A9}$$

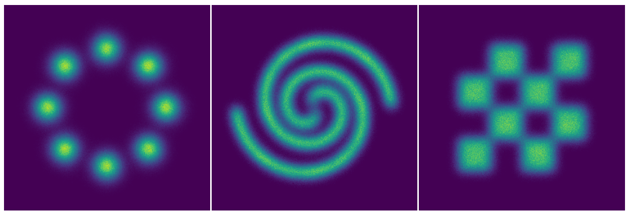
where $\mathcal{D}$ denotes a set of 1,600 grid points. A visualization of $\mathcal{D}$ is depicted in Fig. A2 (b).

A.5.3 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. C-NCSN++, U-NCSN++, and QC-NCSN++ are implemented using the `Pytorch` framework. C-NCSN++, U-NCSN++, and QC-NCSN++ are trained using the Adam optimizer with a learning rate of 2×10^{-4}. The batch size is fixed to 128, while the value of K for QC-NCSN++ is fixed to 1. The training procedure of QC-NCSN++ consists of two stages. In the first stage, QC-NCSN++ is optimized according to the score matching objective, which requires 600,000 training iterations for convergence. In the second stage, the regularization term $\mathcal{L}_{\text{QC}}^{\text{est}}$ is incorporated during the training process, which requires 150,000 training iterations for convergence. The maximal and minimal noise scales $\sigma_{\max}$ and $\sigma_{\min}$ 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 λ is set to 0.0001. The ODE sampler is implemented using the `scipy.integrate.solve_ivp` library.

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

(a)


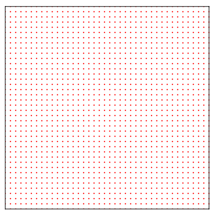
(b)


Figure A2: (a) The visualizations of the 8-Gaussian, Spirals, and Checkerboard datasets. (b) The grid points comprising $\mathcal{D}$.

calculated based on the following equation:

$$Asym = \sum_{t\in\mathcal{T}} \frac{1}{|\mathcal{T}|} \sum_{\tilde{\boldsymbol{x}}\in\mathcal{D}} \frac{1}{|\mathcal{D}|} \mathbb{E}_{p(\boldsymbol{v})} \left[\boldsymbol{v}^T JJ^T \boldsymbol{v} - \boldsymbol{v}^T JJ\boldsymbol{v} \right], \tag{A10}$$

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

$$NAsym = \sum_{t\in\mathcal{T}} \frac{1}{|\mathcal{T}|} \sum_{\tilde{\boldsymbol{x}}\in\mathcal{D}} \frac{1}{|\mathcal{D}|} \frac{1}{2} \mathbb{E}_{p(\boldsymbol{v})} \left[\frac{\boldsymbol{v}^T JJ^T \boldsymbol{v} - \boldsymbol{v}^T JJ\boldsymbol{v}}{\boldsymbol{v}^T JJ^T \boldsymbol{v}} \right]. \tag{A11}$$

The expectations $\mathbb{E}_{p(\boldsymbol{v})}[\cdot]$ in Eqs. (A10) and (A11) 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 (Kynkäänniemi et al., 2019; Naeem et al., 2020).

Sampling performance of U-NCSN++. Table A3 compares the sampling performance of the baseline method (i.e., U-NCSN++) reported in (Xu et al., 2022) and that reproduced by us on the CIFAR-10 dataset using an ODE sampler. It is observed that the reproduced results are improved in terms of the FID, IS, and NFE metrics. This reinforces our statement in Section 5, as QC-NCSN++ is able to achieve superior results to both the reproduced and reported performance of U-NCSN++.

Table A3: A comparison between the results reported in (Xu et al., 2022) and those reproduced by us for U-NCSN++.

	FID	IS	NFE
U-NCSN++ (Xu et al., 2022)	7.66	9.17	194
U-NCSN++ (Ours)	7.48	9.24	170
QC-NCSN++ (Ours)	**7.21**	**9.25**	**124**

Confidence Intervals of the Evaluation Results. Table A4 shows the 95% confidence intervals for the evaluation results of QC-NCSN++ in terms of the NLL, FID, IS, Precision, and Recall metrics on the CIFAR-10 dataset. For the evaluation results of the FID, IS, Precision, and Recall metrics, the PC sampler with NFE=1,000 is adopted. All of these results are obtained by three times of evaluations.

Table A4: The confidence interval of the evaluation results on the CIFAR-10 dataset.

NLL	FID	IS	Precision	Recall
±0.0014	±0.0143	±0.0065	±0.0024	±0.0011

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 C-NCSN++, U-NCSN++, and QC-NCSN++ in terms of their time and memory consumption for each training and sampling iteration. In Section A.6.3, we demonstrate the impact of the choices of λ on the performance of QC-NCSN++. In Section A.6.4, we offer the results of QC-NCSN++ with a PC sampler on the CIFAR-100 and SVHN datasets. Finally, in Section A.6.5, we provide additional 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 $s(\tilde{\boldsymbol{x}};\theta) = Rh(W^T\tilde{\boldsymbol{x}} + \boldsymbol{b}) + \boldsymbol{c}$ 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 $\left\|WR^T - RW^T\right\|_F$ and $\|W - R\|_F$ during the minimization process of $\mathcal{L}_{\text{QC}}$ with four different seeds. As the training progresses, $\mathcal{L}_{\text{QC}}$ and $\left\|WR^T - RW^T\right\|_F$ 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 evidences demonstrate that QCSGMs can learn to output conservative vector fields with $R \neq W$, and thus justify the advantage of QCSGMs over CSGMs.

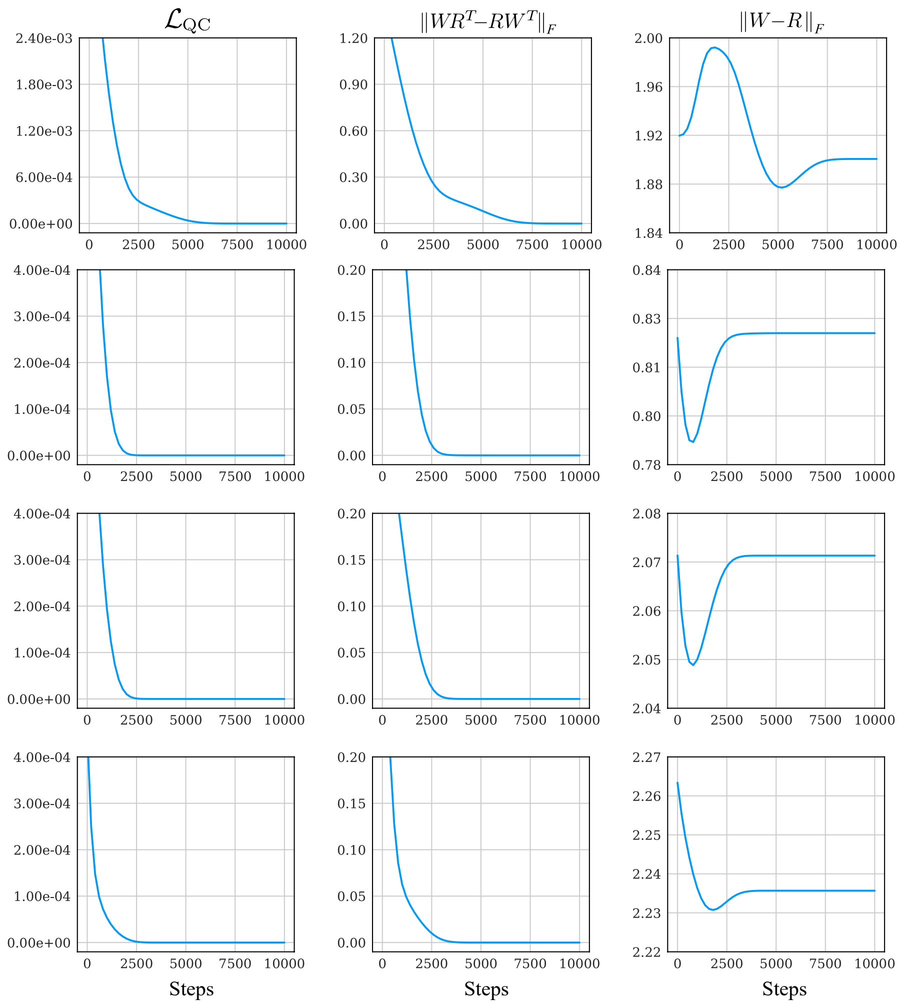


Figure A3: The trends of $\left\|WR^T - RW^T\right\|_F$ and $\|W - R\|_F$ during the minimization process of $\mathcal{L}_{\text{QC}}$. The 'steps' on the x-axes refer to the training steps.

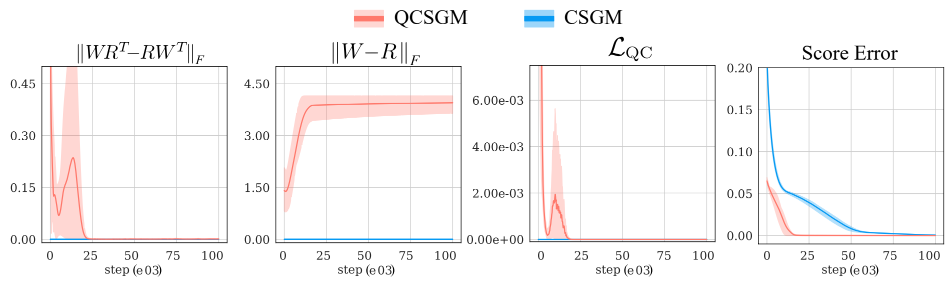


Figure A4: The trends of $\left\|WR^T - RW^T\right\|_F$, $\|W - R\|_F$, $\mathcal{L}_{\text{QC}}$, and the score error of QCSGM and CSGM during the minimization process of $\mathcal{L}_{\text{Total}}$. The 'step' on the x-axes stands for the training steps. The curves depict the mean and 95% confidence interval of three times of training. In this experiment, CSGM and QCSGM are implemented using one-layered autoencoders.

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.

A.6.2 A Comparison on the Time and Memory Consumption

In this section, we investigate the time and memory consumption of C-NCSN++, U-NCSN++, and QC-NCSN++ for each training and sampling iteration. The results are evaluated on a single NVIDIA V100 GPU with 32GB memory, and the batch size is fixed at 32. Table A5 reports the evaluation results of the above setting. The training time and memory requirement of QC-NCSN++ are 2.6× and 2.3× higher than U-NCSN++ as the calculation of $\mathcal{L}_{\text{QC}}^{\text{est}}$ requires two additional backward propagations. On the other hand, QC-NCSN++ and U-NCSN++ have the same time and memory requirement for each sampling iteration, which are 2.6× and 1.6× more efficient than those of C-NCSN++, respectively.

Table A5: The time and memory consumption of C-NCSN++, U-NCSN++, and QC-NCSN++. The time is reported as the mean over 100 evaluations.

	Training		Sampling	
Method	Time	Memory	Time	Memory
C-NCSN++	0.25 s	20.4 GB	0.23 s	18.1 GB
U-NCSN++	0.13 s	13.8 GB	0.09 s	11.5 GB
QC-NCSN++	0.34 s	32.0 GB	0.09 s	11.5 GB

To demonstrate the effectiveness of QC-NCSN++ over U-NCSN++ within the same training time, we compare the performance of 'QC-NCSN++' against both 'U-NCSN++', and 'U-NCSN++ (Extend)', where 'U-NCSN++ (Extend)' represents the U-NCSN++ model trained with an extended training period after convergence. As shown in Table A6, U-NCSN++ (Extend) has inferior performance in comparison to U-NCSN++ and QC-NCSN++, suggesting that even with an extended the training period, U-NCSN++ is unable to improve its performance.

Table A6: The evaluation results in terms of the FID and IS metrics of U-NCSN++, U-NCSN++ (Extend), and QC-NCSN++ on the CIFAR-10 dataset. In this experiment, a PC sampler with NFE=1,000 is adopted.

Method	Training Time	FID	IS
U-NCSN++	87 hours	2.50	9.58
U-NCSN++ (Extend)	144 hours	2.51	9.56
QC-NCSN++	144 hours	**2.48**	**9.70**

A.6.3 The Impact of the Choices of λ on the Performance of QC-NCSN++

Based on our preliminary results on the toy environment, we perform a hyperparameter sweep for $\lambda = \{$1e-3, 5e-4, 1e-4, 5e-5$\}$, and report the best results on the real-world experiments. Table A7 presents the evaluation results of FID and IS under different choices of λ. In this experiment, the PC sampler is adopted and NFE is fixed at 1,000. The experimental results presented on the rows 'FID' and 'IS' demonstrate that QC-NCSN++ achieves its best sampling performance when λ is selected as 0.0001. Based on this finding, we choose λ to equal to 0.0001 throughout the experiments in Section 5.

Table A7: The evaluation results of QC-NCSN++ with different choices of λ on the CIFAR-10 dataset.

λ	0.001	0.0005	0.0001	0.00005	0.0
Asym	**9.99 e6**	2.38 e7	5.03 e7	6.42 e7	1.88 e8
FID	2.75	2.53	**2.48**	**2.48**	2.50
IS	9.58	9.64	**9.70**	9.61	9.58

A.6.4 Additional Results of U-NCSN++ and QC-NCSN++ with PC Sampler

Table A8 presents the experimental results of U-NCSN++ and QC-NCSN++ using a PC sampler (NFE=2,000) on the SVHN and CIFAR-100 datasets. It is observed that QC-NCSN++ is able to outperform U-NCSN++ in terms of FID/IS/Precision/Recall, justifying the effectiveness of $\mathcal{L}_{\text{QC}}^{\text{est}}$.

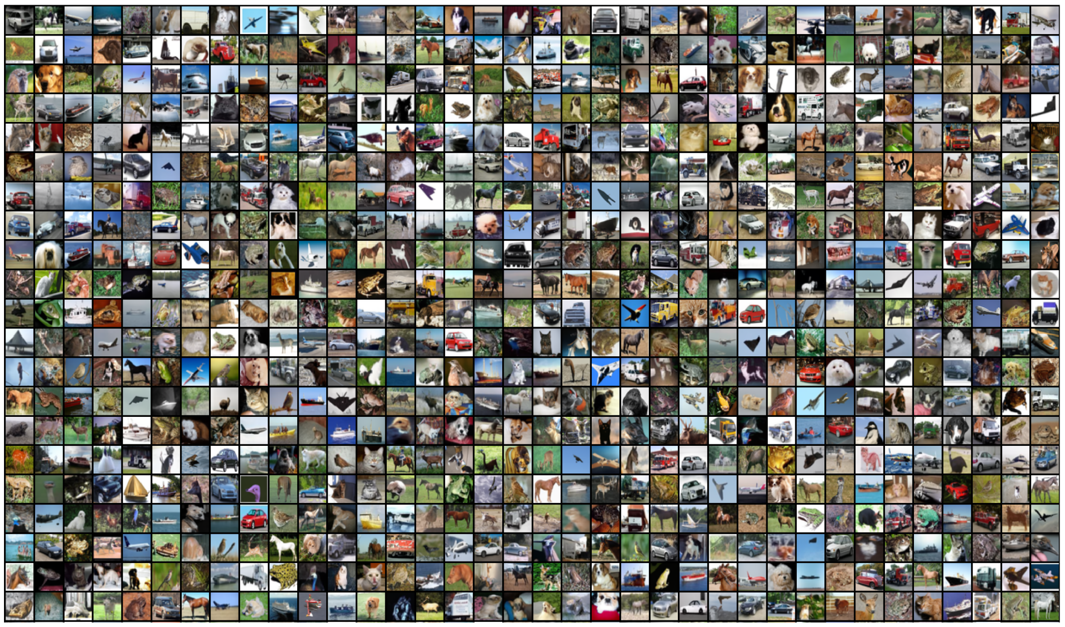
Table A8: The evaluation results of U-NCSN++ and QC-NCSN++ with PC sampler on the CIFAR-100 and SVHN datasets.

	Cifar-100			
Method	FID	IS	Precision	Recall
U-NCSN++	3.53	11.86	0.6341	0.6227
QC-NCSN++	**3.50**	**11.88**	**0.6372**	**0.6242**
	SVHN			
Method	FID	IS	Precision	Recall
U-NCSN++	14.34	3.10	0.6666	0.5970
QC-NCSN++	**13.88**	**3.12**	**0.6669**	**0.6057**

A.6.5 Visualized Examples

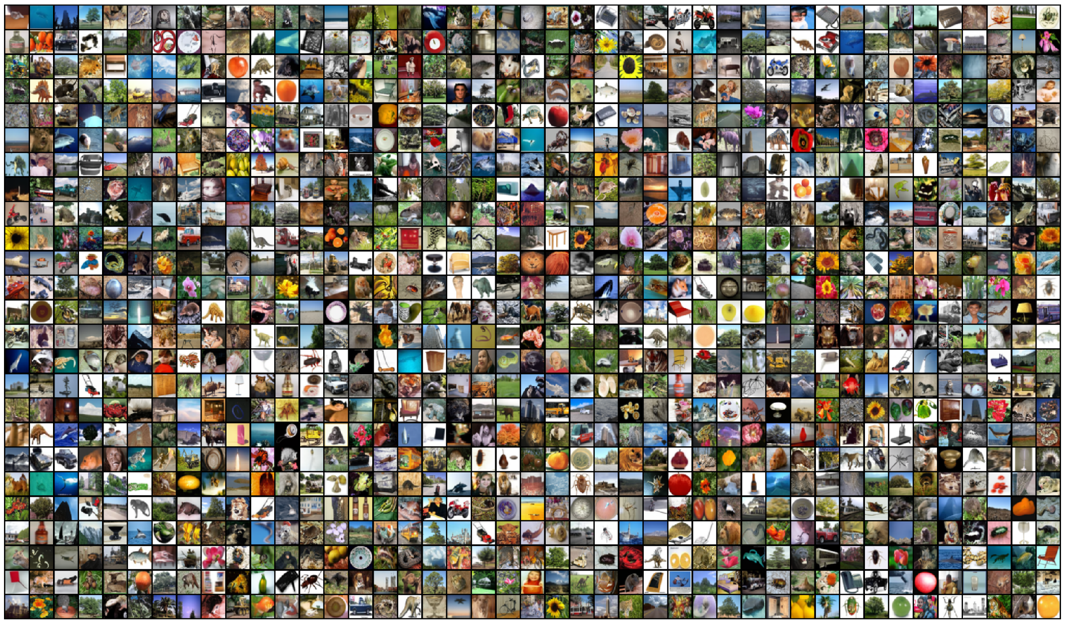
Figs. A5-A8 depict a few uncurated visualized examples that qualitatively demonstrate the sampling quality of NCSN++ and QC-NCSN++ on the real-world datasets.

(a) NCSN++ (VE)



(b) QC-NCSN++ (VE)

Figure A5: The visualized examples generated using (a) U-NCSN++ and (b) QC-NCSN++ on the CIFAR-10 dataset.


(a) NCSN++ (VE)

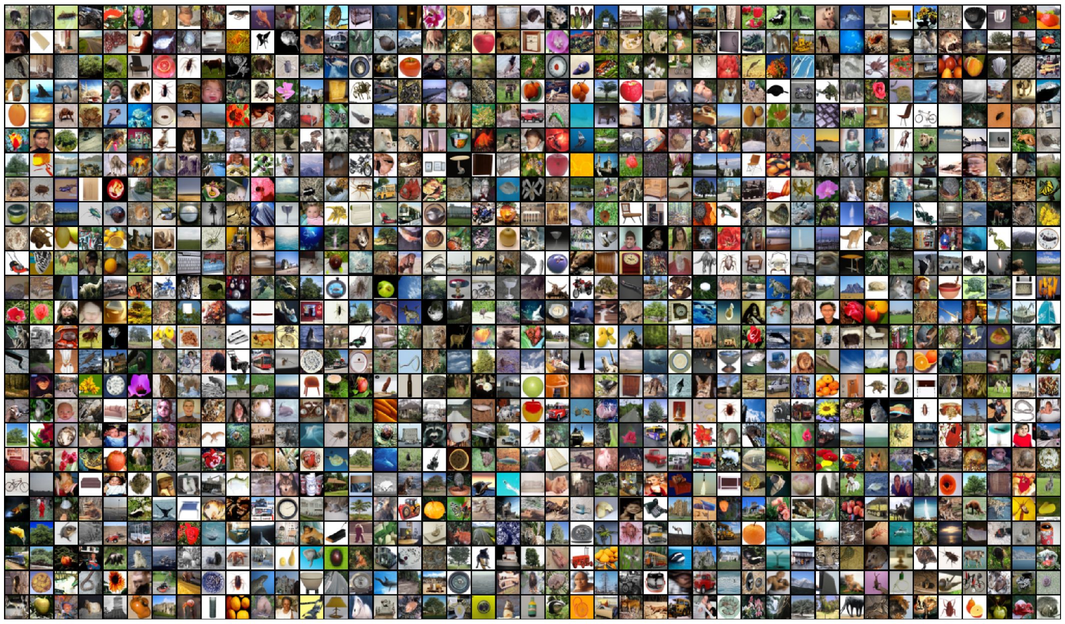
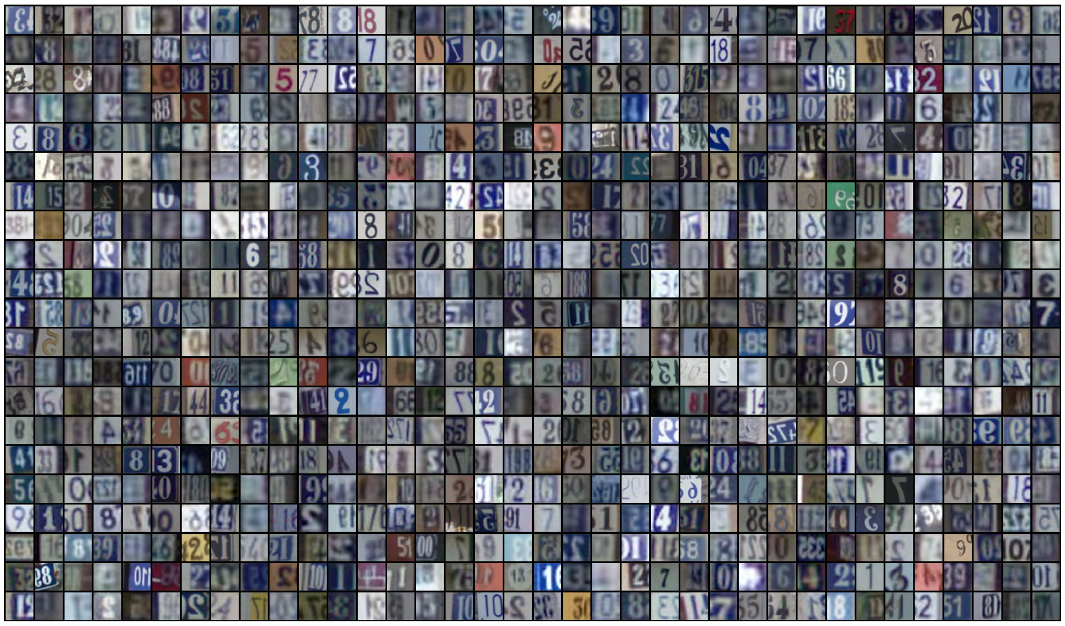

(b) QC-NCSN++ (VE)

Figure A6: The visualized examples generated using (a) U-NCSN++ and (b) QC-NCSN++ on the CIFAR-100 dataset.

(a) NCSN++ (VE)

(b) QC-NCSN++ (VE)

Figure A7: The visualized examples generated using (a) U-NCSN++ and (b) QC-NCSN++ on the ImageNet-32x32 dataset.


(a) NCSN++ (VE)

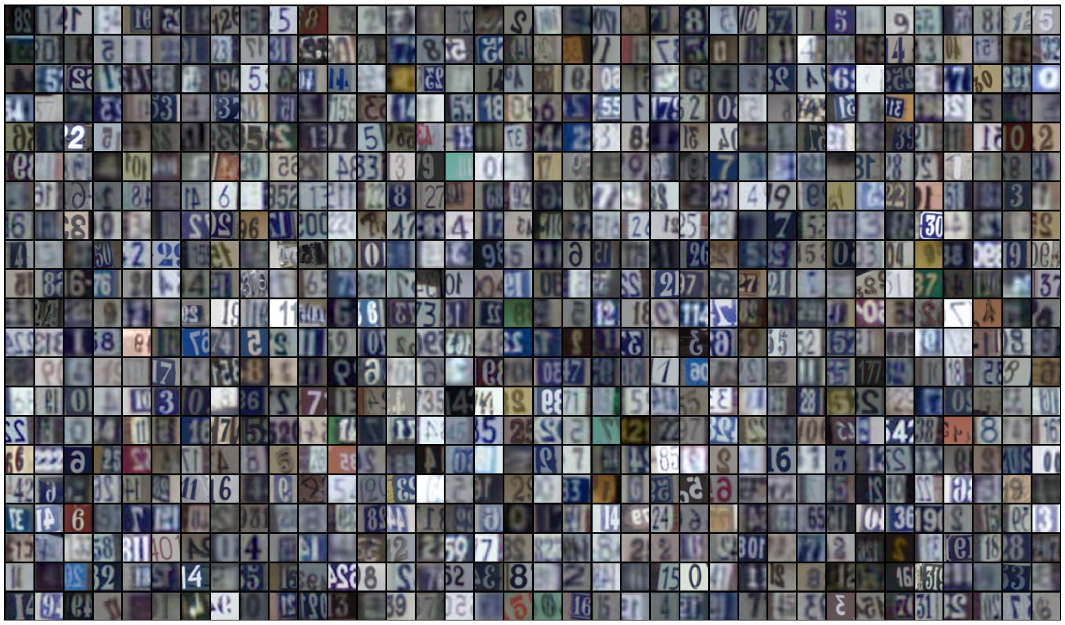

(b) QC-NCSN++ (VE)

Figure A8: The visualized examples generated using (a) U-NCSN++ and (b) QC-NCSN++ on the SVHN dataset.