COMPARING SCORE-BASED DIFFUSION MODELS AND STOCHASTIC INTERPOLANTS ON APPROXIMATING GAUSSIAN MIXTURES

KAI CHANG *

Abstract. In this report, we focus on comparing the score-based generative modeling with stochastic differential equations (SDE-SBGM) [9] and the stochastic interpolant (SI) [1], two flow-based methods for approximating and sampling from high-dimensional probability densities. Falling under the umbrella of transport maps, both methods seek to construct a mapping between a tractable base density and the target density. While SDE-SBGMs model the mapping as a reversed Ornstein-Uhlenbeck process, SIs employ a simple ordinary differential equation and characterize the mapping by a velocity field. As such, SDE-SBGMs can be interpreted as stochastic flow-based methods and SIs as deterministic flow-based methods. Moreover, both SDE-SBGMs and SIs adopt a least-squares type training objective.

The different but closely related formulations of the two methods motivate us to conduct a comparative study on their performance. In particular, we apply the two methods on a two-dimensional and a ten-dimensional four-mode Gaussian mixtures and record the following quantities: the Euclidean norm of the difference between the approximate mean and the true mean, the Frobenius norm of the difference between the approximate covariance and the true variance, and the entropy-regularized Wasserstein-2 distance between samples from the approximate distribution and those from the true distribution. We investigate the effect of the number of discretization steps on the approximation performance of the methods.

Key words. Sampling, generative modeling, transport maps, diffusion models

1. Introduction.

We consider the problem of sampling from and approximating an unknown probability distribution with the sole knowledge of some independent and identically distributed (IID) data from the distribution. Before rigorously defining the problem, let us set up the mathematical notations first.

Let $(\Omega, \mathcal{F}, P)$ be a probability space, where Ω is a sample space, $\mathcal{F}$ is a σ-algebra, and P is a probability measure on Ω. Let $X : \Omega \to \mathbb{R}^d$ be an $\mathcal{F}$-measurable function (in other words, a random variable). X and P induce a measure μ on $\mathbb{R}^d$ which is defined as

$$\mu(A) = P(X^{-1}(A))$$

where $A \in \mathbb{R}^d$ is a set and $X^{-1}(A) = \{\omega \in \Omega : X(\omega) \in A\}$. We further assume the existence of the probability density function (PDF) p of X under the measure μ, which is defined as the Radon-Nikodym derivative of μ with respect to the Lebesgue measure λ on $\mathbb{R}^d$:

$$p = \frac{d\mu}{d\lambda},$$

where

$$\mu(A) = \int_A p \, d\lambda.$$

The existence of the PDF is generally a weak assumption. Under such an assumption, we refer $X \sim p$ as "X follows the distribution of p" or equivalently, "the PDF of X is p". Note that since we are working with probability measures, we have

$$\int_{\mathrm{Supp}(p)} p \, d\lambda = 1$$

where $\mathrm{Supp}(p)$ denotes the support of p.

Now, to formalize the problem of interest, let $\{x_i\}_{i=1}^N$ be a set of independent realizations of some random variable $\boldsymbol{x}_0 \in \mathbb{R}^d$ where $\boldsymbol{x}_0 \sim p_*$ for some unknown PDF p_*. The goal is to approximate and sample from p_*. When d is large, such as in computer vision and natural language processing tasks, the problem can be extremely difficult to solve.

Over the years, a number of methods have been developed to improve the approximation accuracy and sampling efficiency for high-dimensional random variables. Here, we focus on two methods, namely the score-based generative modeling with stochastic differential equations (SDE-SBGM) [9] and the stochastic interpolant (SI) [1], that not only have shown superior results in challenging problems, but also present neat and intriguing mathematical formulations. On the high level, both SDE-SBGM and SI attempt to find a

*Center for Computational Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 (kaichang@mit.edu)

mapping T from a tractable base density p_0, from which sampling is easy, to the target density p_*. Once T is found, sampling from p_* reduces to sampling from p_0 and applying T on those samples.

The different but closely related formulations of the two methods motivate us to conduct a comparative study on their performance. We apply the two methods on a two-dimensional and a ten-dimensional four-mode Gaussian mixtures and record the following quantities: the Euclidean norm of the difference between the approximate mean and the true mean, the Frobenius norm of the difference between the approximate covariance and the true variance, and the entropy-regularized Wasserstein-2 distance between samples from the approximate distribution and those from the true distribution. We investigate the effect of the number of discretization steps on the approximation performance of the methods.

2. Methodological Formulation. In this section, we give a brief review on the score-based generative modeling with stochastic differential equations (SDE-SBGM) [9] and the stochastic interpolant (SI) [1]. In particular, to introduce the SDE-SBGMs, we first formalize denoising diffusion probabilistic models (DDPMs) due to DDPMs' deep connections with SDE-SBGM and the insights DDPMs have to offer on the training objective of SDE-SBGMs. SIs are introduced subsequently. Despite the major difference between the formulation of SIs and SDE-SBGMs, the ideas behind them are highly correlated: to model a mapping T from a tractable target density ρ_0 to the target density ρ_1 as a continuous process governed by some differential equation.

2.1. Denoising Diffusion Probabilistic Models. Let $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ be a series of random variables where $\boldsymbol{x}_0 \sim p_*$, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}$ are latent variables whose distributions are to be defined, and $\boldsymbol{x}_T \sim p_0 := \mathcal{N}(0, I)$ is a Gaussian noise. We use the notation $\boldsymbol{x}_{i:j}$ to denote the set $\{\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_j\}$. A denoising diffusion probabilistic model (DDPM) is modeled in two steps: a forward process and a reverse process.

In the forward process, a DDPM "encodes" $\boldsymbol{x}_0$ into $\boldsymbol{x}_T$ by gradually adding Gaussian noise to it until it reaches $\boldsymbol{x}_T$. Mathematically, it is defined through a series of conditional distributions that read as follows: for any t such that $1 \leq t \leq T-1$, the distribution of $\boldsymbol{x}_t$ conditioned on $\boldsymbol{x}_{t-1}$ is modeled as

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}\left(\sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\,\mathbf{I}\right) \tag{2.1}$$

where $\mathbf{I}$ is the identity matrix and $\alpha_t \in \mathbb{R}$ is a potentially learnable scalar that varies with t. It is worth noting that with (2.1), we may derive that

$$\boldsymbol{x}_t \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\,\mathbf{I}\right) \tag{2.2}$$

where $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i$. The detailed derivation of (2.2) can be found in [6]. This expression is crucial as it directly relates $\boldsymbol{x}_t$ and $\boldsymbol{x}_0$ without consulting the middle terms. It appears in the objective function of DDPMs.

In the reverse process, a DDPM attempts to map $\boldsymbol{x}_T$ back to $\boldsymbol{x}_0$ through a series of transformations defined by the reverse conditional probability model $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$, the form of which and how it is parameterized by $\boldsymbol{\theta}$ is discussed later. It does so by maximizing a proxy of $\log p_*(\boldsymbol{x}_0)$, namely the evidence lower bound (ELBO), during training for all the training data $\{\boldsymbol{x}_i\}_{i=1}^N$. The idea is to look for the density function that makes all the data "seen" by the model mostly likely to occur if the density function were the true one. The ELBO can be written as

$$\log p(\boldsymbol{x}_0) \geq \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1 \mid \boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}\left(q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \,\|\, p_0(\boldsymbol{x}_T)\right)}_{\text{prior matching term}} \tag{2.3}$$

$$- \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)}\left[D_{\mathrm{KL}}\left(q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \,\|\, p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right)\right]}_{\text{denoising matching term}}. \tag{2.4}$$

where how $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ is defined will be discussed later. The detailed derivation of (2.3) can be found in [6].

We make the following comments on the maximization of the ELBO: 1) maximizing the ELBO is equivalent to simultaneously maximizing the reconstruction term, minimizing the prior matching term, and minimizing the denoising matching term; 2) maximizing the reconstruction term makes sure that the latent variable $\boldsymbol{x}_1$ is effective in encoding the original data; 3) minimizing the prior matching term ensures that

the distribution of the final latent variable $\boldsymbol{x}_T$ matches the prior distribution $p_0(\boldsymbol{x}_T)$, which is assumed to be a standard Gaussian in this case; and 4) minimizing the denoising matching term makes sure that the learned reverse conditional probability approximates the ground-truth reverse conditional probability as well as possible. Now, note that the prior matching term does not involve trainable parameters, assuming that α_t is fixed for all t. Therefore, only the reconstruction and the denoising matching terms will require training.

To derive a simpler form for the denoising matching term, notice that by Bayes' Theorem, we have

$$q\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) = \frac{q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, \boldsymbol{x}_0\right) q\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_0\right)}{q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right)} \tag{2.5}$$

$$\propto \mathcal{N}\left(\frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}\left(1-\alpha_t\right)\boldsymbol{x}_0}{1-\bar{\alpha}_t}, \frac{\left(1-\alpha_t\right)\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{I}\right) \tag{2.6}$$

where (2.6) follows from (2.1) and (2.2). Let us define

$$\mu_q\left(\boldsymbol{x}_t, \boldsymbol{x}_0\right) = \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}\left(1-\alpha_t\right)\boldsymbol{x}_0}{1-\bar{\alpha}_t} \tag{2.7}$$

and

$$\boldsymbol{\Sigma}_q(t) = \sigma_q^2(t)\mathbf{I} \tag{2.8}$$

where

$$\sigma_q^2(t) = \frac{\left(1-\alpha_t\right)\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}.$$

Let the reverse conditional probability model, $p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right)$, be parameterized as $\mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right), \boldsymbol{\Sigma}_q(t)\right)$ such that

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right) = \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}\left(1-\alpha_t\right)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right)}{1-\bar{\alpha}_t}, \tag{2.9}$$

where $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right)$ is a Neural Network (NN) parameterized by $\boldsymbol{\theta}$ that takes $\boldsymbol{x}_t$ and t as inputs. Note that $\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right)$ is defined in such a way to match $\mu_q\left(\boldsymbol{x}_t, \boldsymbol{x}_0\right)$ the formula of which is shown in (2.7). As such, $\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right)$ seeks to predict $\boldsymbol{x}_0$ from x_t at time t. With these, the integrand in the denoising matching term in (2.3) can be rewritten as

$$D_{\mathrm{KL}}\left(q\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) \| p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right)\right) = \frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1}\left(1-\alpha_t\right)^2}{\left(1-\bar{\alpha}_t\right)^2}\left[\left\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right) - \boldsymbol{x}_0\right\|_2^2\right]. \tag{2.10}$$

With (2.10), maximizing the ELBO defined in (2.3) can be equivalently written as

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{t \sim U\{2,T\}}\left[\mathbb{E}_{q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right)}\left[D_{\mathrm{KL}}\left(q\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) \| p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right)\right)\right]\right], \tag{2.11}$$

which can be optimized using gradient-descent type methods, such as Adam [5], after estimating the value of the objective by drawing samples from $t \sim U\{2, T\}$ and $\boldsymbol{x}_t \sim q\left(\boldsymbol{x}_t \mid \boldsymbol{x}_0\right)$ and taking sample average.

2.2. Score-Based Diffusion Modeling through Stochastic Differential Equations. The SDE-SBDM is closely related to the DDPM in that the training objective of SDE-SBDM can be directly derived from (2.10) by using the so-called Tweedie's formula [6], and conversely, the DDPM can also be formulated as a discretized SDE-SBDM. For a normally distributed random variable $\boldsymbol{z} \sim \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z\right)$, the Tweedie's formula reads

$$\mathbb{E}\left[\boldsymbol{\mu}_z \mid \boldsymbol{z}\right] = \boldsymbol{z} + \boldsymbol{\Sigma}_z \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}) \tag{2.12}$$

Applying the formula above to (2.2), we obtain that

$$\mathbb{E}\left[\boldsymbol{\mu}_{x_t} = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 \mid \boldsymbol{x}_t\right] = \boldsymbol{x}_t + \left(1-\bar{\alpha}_t\right)\nabla_{\boldsymbol{x}_t} \log p_t\left(\boldsymbol{x}_t\right),$$

where p_t is the underlying true distribution of $\boldsymbol{x}_t$. Rearranging terms yields

$$\boldsymbol{x}_0 = \frac{\boldsymbol{x}_t + (1-\bar{\alpha}_t)\,\nabla_{\boldsymbol{x}_t} \log p_t\left(\boldsymbol{x}_t\right)}{\sqrt{\bar{\alpha}_t}}. \tag{2.13}$$

Plugging (2.13) into (2.7) gives

$$\boldsymbol{\mu}_q\left(\boldsymbol{x}_t, \boldsymbol{x}_0\right) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla_{\boldsymbol{x}_t} \log p_t\left(\boldsymbol{x}_t\right). \tag{2.14}$$

Then, $\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right)$ as in

$$p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right), \boldsymbol{\Sigma}_q(t)\right)$$

can be parameterized as

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right),$$

where $\boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right)$ seeks to approximate the score function of the true distribution of $\boldsymbol{x}_t$. The form of $D_{\mathrm{KL}}\left(q\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) \| p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right)\right)$, in contrast to (2.10), can then be shown to be

$$D_{\mathrm{KL}}\left(q\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0\right) \| p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t\right)\right) = \frac{1}{2\sigma_q^2(t)}\frac{\left(1-\alpha_t\right)^2}{\alpha_t}\left[\left\|\boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t, t\right) - \nabla_{\boldsymbol{x}_t} \log p_t\left(\boldsymbol{x}_t\right)\right\|_2^2\right], \tag{2.15}$$

and the objective function (2.11) of the DDPM can be reformulated accordingly.

The above argument shows that the objective function of the DDPM can be formulated as the score-matching objective in (2.15). SDE-SBGM extends this "DDPM via score matching" method to a continuous process. In particular, the forward process, which maps from $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$, in SDE-SBGM is defined as the following Ornstein-Uhlenbeck (OU) process

$$\mathrm{d}\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}, \quad t \in [0, T], \tag{2.16}$$

where $\boldsymbol{f}(\boldsymbol{x}_t, t)$ is the drift coefficient which has the same dimension as $\boldsymbol{x}_t$, $g(t) \in \mathbb{R}$ is the diffusion coefficient, and $\mathbf{w}$ is a standard Wiener process (and $\mathrm{d}\mathbf{w}$ is therefore a white noise). The reverse process remarkably admits a closed-form SDE in terms of the forward OU process:

$$\mathrm{d}\boldsymbol{x}_t = \left[\boldsymbol{f}(\boldsymbol{x}_t, t) - g(t)^2\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)\right]\mathrm{d}t + g(t)\mathrm{d}\overline{\mathbf{w}}, \tag{2.17}$$

where $\boldsymbol{f}$ and g are the same as those in (2.16), and $\mathrm{d}\overline{\mathbf{w}}$ is a standard Wiener process. Since $\boldsymbol{f}$ and g are modeling assumptions, finding (2.17) amounts to learning $\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t)$ from data. To do this, it is proposed to solve a continuous generalization of (2.15), which reads

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_t\left\{\lambda(t)\mathbb{E}_{\boldsymbol{x}_0}\mathbb{E}_{\boldsymbol{x}_t|\boldsymbol{x}_0}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{x}_t|\boldsymbol{x}_0}(\boldsymbol{x}_t \mid \boldsymbol{x}_0)\right\|_2^2\right]\right\}, \tag{2.18}$$

where $\lambda(t)$ taking positive real values is a weighting function, $t \sim U\{[0, T]\}$, $\boldsymbol{x}_0 \sim p_0(\boldsymbol{x})$, and $\boldsymbol{x}_t \sim p_{\boldsymbol{x}_t|\boldsymbol{x}_0}(\boldsymbol{x}_t \mid \boldsymbol{x}_0)$. To see the equivalence between (2.18) and the score-matching version of (2.11), we comment that the missing of the $\mathbb{E}_{\boldsymbol{x}_0}$ operator in (2.11) is due to the fact that in the derivation of (2.11), we assumed that $\boldsymbol{x}_0$ stands for "all the training data available". As such (2.11) should be interpreted as the objective function for each realization of $\boldsymbol{x}_0$, which is equivalent to (2.18) because to apply $\mathbb{E}_{\boldsymbol{x}_0}$ in (2.18), we use an empirical uniform distribution on all the available realizations of $\boldsymbol{x}_0$.

2.3. Stochastic Interpolants. In contrast to SDE-SBGMs, SIs attempt to model the mapping from p_0 to p through a continuous process governed by the ordinary differential equation

$$\dot{\boldsymbol{x}}_t = v_t\left(\boldsymbol{x}_t\right), \quad \boldsymbol{x}_0 \sim p_0 \text{ and } t \in [0, 1] \tag{2.19}$$

where $\boldsymbol{x}_t$ is the decoded latent variable of $\boldsymbol{x}_0$ at time t, similar to that in SDE-SBGMs but with a reversed indexing to be consistent with the notations in the original paper [1]. This Lagrangian (sample-level) view of the process is equivalent to the following Eulerian (distribution-level) formulation:

$$\partial_t p_t + \nabla \cdot (v_t p_t) = 0 \quad \text{with } p_{t=0} = p_0 \text{ and } p_{t=1} = p_* \tag{2.20}$$

where p_t is the probability density of $\boldsymbol{x}_t$ such that $\boldsymbol{x}_t$ is governed by (2.19).

Using (2.19) per se to tackle generative modeling has been known as method of continuous normalizing flows (CNFs) in the literature [4, 2]. In contrast to the conventional CNFs that adopt maximum-likelihood estimation during training, SIs further take advantage of (2.20) to derive a least-squares-type training objective, and it is formulated as follows.

Let $I_t : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}^d$ be a mapping such that

$$I_{t=0}(\boldsymbol{x}_0, \boldsymbol{x}_1) = \boldsymbol{x}_0$$

and

$$I_{t=1}(\boldsymbol{x}_0, \boldsymbol{x}_1) = \boldsymbol{x}_1.$$

Define $\boldsymbol{x}_t := I_t(\boldsymbol{x}_0, \boldsymbol{x}_1)$ as the so-called stochastic interpolant. Then, by noticing that

$$p_t(\boldsymbol{x}_t) = \int_{\mathbb{R}^d \times \mathbb{R}^d} \delta\left(\boldsymbol{x} - I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right) p_0\left(\boldsymbol{x}_0\right) p_1\left(\boldsymbol{x}_1\right) d\boldsymbol{x}_0 d\boldsymbol{x}_1,$$

we obtain, using the chain rule,

$$\partial_t p_t(\boldsymbol{x}) = -\int_{\mathbb{R}^d \times \mathbb{R}^d} \partial_t I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right) \cdot \nabla_{\boldsymbol{x}} \delta\left(\boldsymbol{x} - I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right) p_0\left(\boldsymbol{x}_0\right) p_1\left(\boldsymbol{x}_1\right) d\boldsymbol{x}_0 d\boldsymbol{x}_1 := -\nabla_{\boldsymbol{x}} \cdot j_t(\boldsymbol{x}) \tag{2.21}$$

where

$$j_t(\boldsymbol{x}) := \int_{\mathbb{R}^d \times \mathbb{R}^d} \partial_t I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right) \delta\left(\boldsymbol{x} - I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right) p_0\left(\boldsymbol{x}_0\right) p_1\left(\boldsymbol{x}_1\right) d\boldsymbol{x}_0 d\boldsymbol{x}_1.$$

Followed by this, define

$$v_t = \begin{cases} j_t / p_t & \text{if } p_t > 0 \\ 0 & \text{otherwise} \end{cases}.$$

It is then not hard to see from (2.21) that (2.20) is satisfied under such a definition.

With I_t satisfying the distribution-level transport equation (2.20), the velocity field v_t that connects p_0 and p_* through (2.19) and (2.20) is then the solution to the optimization problem

$$\begin{aligned} v_t &= \arg\min_{v_t} \mathbb{E}_{t \sim U([0,1])} \left[\left|v_t\left(I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right) - \partial_t I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right|^2\right] \\ &= \arg\min_{v_t} \left(\left|v_t\left(I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right)\right|^2 - 2\partial_t I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right) \cdot v_t\left(I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right)\right) + \text{constant}, \end{aligned}$$

the derivation of which relies on mild requirements on the continuity of p_0 and p_* [1].

Parameterizing v_t^* as a NN with parameter $\boldsymbol{\theta}$, the following optimization problem is solved to find v_t:

$$v_t^* = \arg\min_{\boldsymbol{\theta}} \left(\left|v_t^{(\boldsymbol{\theta})}\left(I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right)\right|^2 - 2\partial_t I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right) \cdot v_t^{(\boldsymbol{\theta})}\left(I_t\left(\boldsymbol{x}_0, \boldsymbol{x}_1\right)\right)\right). \tag{2.22}$$

It is worth noting that the objective in (2.22) is lower bounded by the Wasserstein-2 distance between the p_* and the approximate $\hat{p}_*$ obtained by solving (2.22). This tells that, in principle, the stochastic interpolant method controls certain distance between the approximate distribution and the true distribution, despite not using maximum-likelihood type training.

3. Metrics for Comparison. Assume for our test cases, we have access to the true mean and covariance of the target distribution, which we denote as μ_* and Σ_*. We consider the following metrics to compare the performance of SDE-SBGMs and SIs, itemized for clarity.

- After each model is trained, we draw N_s samples $\{\hat{x}_i\}_{i=1}^{N_s}$ from the approximate target distribution, compute the sample average

$$\hat{\mu} := \frac{1}{N_s} \sum_{i=1}^{N_s} \hat{x}_i$$

as an estimate of the true mean and the sample covariance

$$\hat{\Sigma} := \frac{1}{N_s - 1} \sum_{i=1}^{N_s} (\hat{x}_i - \hat{\mu})(\hat{x}_i - \hat{\mu})^T$$

as an estimate of the true covariance. We record the Euclidean norm of the difference between $\hat{\mu}$ and μ_*, i.e.,

$$\|\hat{\mu} - \mu_*\|_2$$

and the Frobenius norm of the difference between $\hat{\Sigma}$ and Σ_*, i.e.,

$$\|\hat{\Sigma} - \Sigma_*\|_F.$$

- Again, after each model is trained, we draw N_s samples $\{\hat{x}_i\}_{i=1}^{N_s}$ from the approximate target distribution. We then record the (discrete) entropy-regularized 2-Wasserstein distance between the approximate density $\hat{p}$ and the true density p_*. The entropy-regularized 2-Wasserstein distance is defined as

$$W_{2,\varepsilon}(\alpha, \beta) \overset{\text{def.}}{=} \min_{\pi \in \Pi(\alpha,\beta)} \int_{\mathcal{X}\times\mathcal{Y}} \|x - y\|^2 \mathrm{d}\pi(x, y) + \varepsilon \int_{\mathcal{X}\times\mathcal{Y}} \log\left(\frac{\mathrm{d}\pi(x, y)}{\mathrm{d}\alpha(x)\mathrm{d}\beta(y)}\right) \mathrm{d}\pi(x, y)$$

where α and β are probability measures on $\mathcal{X}$ and $\mathcal{Y}$ respectively and $\Pi(\alpha, \beta) = \{\pi : \pi(A, \mathcal{Y}) = \alpha(A), \pi(\mathcal{X}, B) = \beta(B), \forall A \subset \mathcal{X} \text{ and } B \subset \mathcal{Y}\}$ is the set of measure couplings. As optimal transport is not a focus of this writeup, we refer the readers to [8] for more details. In our case, $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ and we are interested in $W_{2,\varepsilon}(\hat{p}, p_*)$ where $\hat{p}$ is the approximate density given by SDE-SBGM or SI. The quantity is computed with the Sinkhorn's Algorithm [3].

4. Numerical Results. In the following set of experiments, we investigate the effect of the number of discretization steps used in training and sampling on the performance of the methods.

We first consider a two-dimensional four-mode mixture of Gaussians as the target density. Let

$$\mu_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \mu_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mu_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

$$\Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

and define the target density as

$$p_* = \sum_{i=1}^{4} \frac{1}{4} \mathcal{N}(\mu_i, \Sigma_i).$$

We use $\mathcal{N}(0, I)$ as the base density. We draw 10000 IID samples from the base density to be passed through a trained model and 10000 IID samples from the target density as the training data. An illustration of the base and target density is provided in Figure 4.1.

Model configurations read as follows. For SDE-SBGMs, we choose the forward process as

$$\mathrm{d}\boldsymbol{x} = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^t \sqrt{2 \log \frac{\sigma_{\max}}{\sigma_{\min}}} \mathrm{d}\mathbf{w}$$

where $\sigma_{\min}$ is selected as 0.01 and $\sigma_{\max}$ is selected as 10. The corresponding forward conditional probability model reads

$$p_{\boldsymbol{x}_t|\boldsymbol{x}_0}(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_0, \sigma_{\min}^2 \left(\frac{\sigma_{\max}}{\sigma_{\min}}\right)^{2t} \mathbf{I}\right),$$

which is a common choice in practice [9]. We use a neural network of 4 linear layers, each of which is combined with a rectified linear unit [7], to parameterize $\boldsymbol{s}_{\boldsymbol{\theta}}$. We use the Euler-Maruyama method as our discretization scheme; the number of discretization steps is to be discussed. For SI, we choose to use the linear interpolant,

$$I_t(\boldsymbol{x}_0, \boldsymbol{x}_1) = (1 - t)\boldsymbol{x}_0 + \boldsymbol{x}_1$$

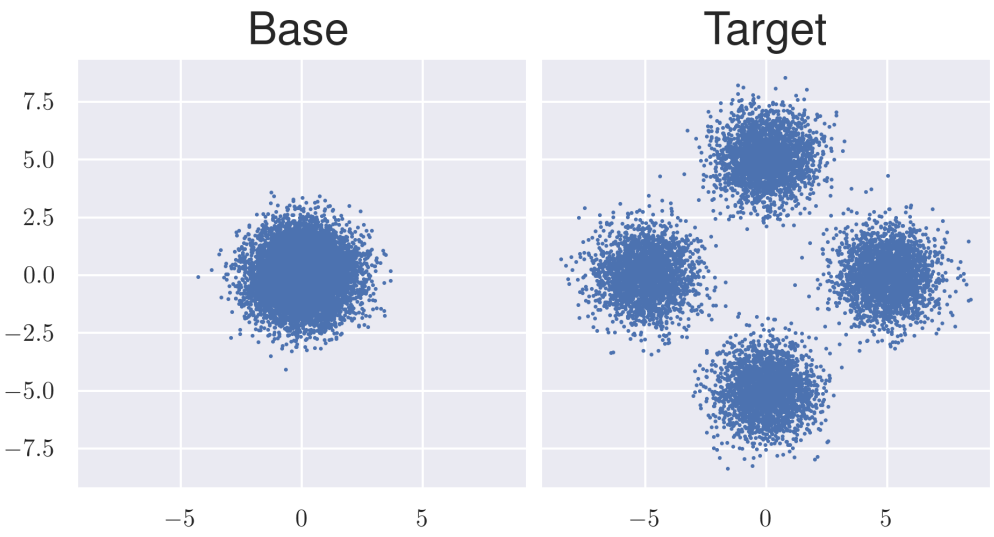

Fig. 4.1: 2-dimensional mixture of Gaussians. Left: 10000 samples from base density. Right: 10000 samples from the 2-dimensional mixture of Gaussians.

which can be easily checked to satisfy the requirements of the interpolant function. We use the explicit Euler's method as our numerical integrator. In both cases, we use Adam [5] as our optimizer with a decay of learning rate, initialized as 0.001, for every 500 steps of training. We train all of our models for 8000 iterations with a batch size of 128.

Suppose N_t is the number of discretization steps used when discretizing (2.17) in SDE-SBGM and (2.19) in SI. Note that N_t is the same for training and sampling. For each $N_t \in \{10, 20, 40, 80, 160, 320, 640, 1280\}$, we train a model with the configuration discussed above. After a model is trained, we draw $N_s = 10000$ samples with it and compute the quantities discussed in Section 3. In particular, for the entropy-regularized 2-Wasserstein distance, ϵ is selected as 0.001, and it is computed with the Sinkhorn's Algorithm [3]. We remark that due to the constraint in computational resources, we were only able to obtain results for N_t up to 160 for the SDE-SBGM. For $N_t \geq 320$, SDE-SBGM requires too long of a time to train on our machine, potentially due to sub-optimal implementations.

The mean error, covariance error, and the entropy-regularized 2-Wasserstein distance are presented in Figures 4.2, 4.3, and 4.4. We make the following observations regarding the numerical results. First, for all of the three metrics, the behavior of SIs is uniformly better than SDE-SBGMs. We suspect that this is due to the fact that SIs in fact admit a closed-form solution as derived in [1]. This could make it easier for SIs to learn the better velocity field. Second, SDE-SBGMs' behavior does not follow the intuition that propagating the base samples for a longer time gives better performance. This can be due to sub-optimal hyper-parameter combinations of the models. In our experiments, we have managed to fine tune the models as much as possible by monitoring the evolution of the training objective. However, making the models work better than they are is still possible. Because of time constraints, we were not able to further make improvements. Third, the small value of the entropy-regularized 2-Wasserstein distance for SIs verifies the 2-Wasserstein lower bound of SIs' training objective mentioned in Section 2.3.

Next, we consider the following ten-dimensional four-mode mixture of Gaussians as the target density. Suppose e_i is the i^{th} column of the 10-by-10 identity matrix. The mixture of Gaussian is defined as

$$p_* = \sum_{i=1}^{4} \frac{1}{4} \mathcal{N}(e_i, I).$$

We again use $\mathcal{N}(0, I)$ as our base density. The experiment set-up is the same as the previous case. The mean error, covariance error, and the entropy-regularized 2-Wasserstein distance are presented in Figures 4.5, 4.6, and 4.7. The behavior of the errors does not deviate much from what we observed in the 2-dimensional case.

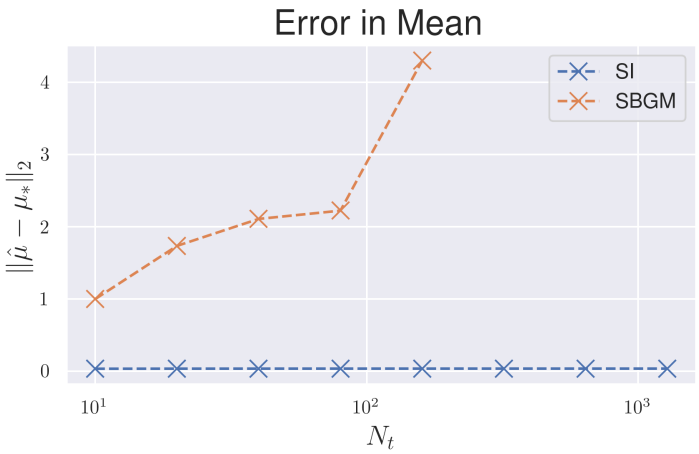

Fig. 4.2: Approximation error in mean for the 2-dimensional case.

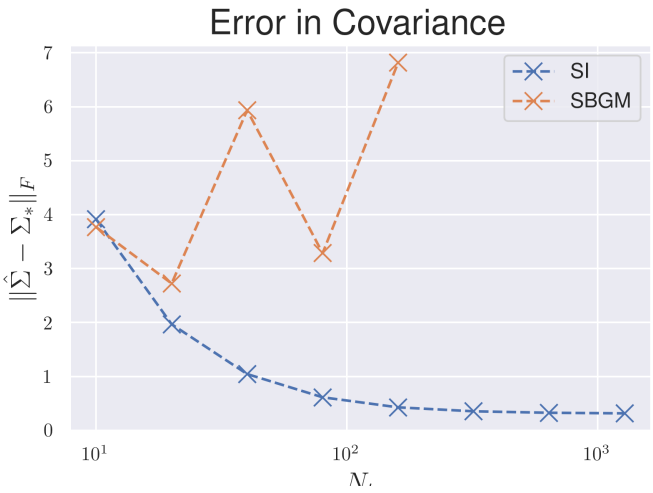

Fig. 4.3: Approximation error in covariance for the 2-dimensional case.

5. Conclusion and Discussion. In this report, we have reviewed two representative methods of the popular diffusion models, namely the denoising diffusion probablistic model (DDPM) and score-based generative modeling through stochastic differential equations (SDE-SBGM), and the recently proposed stochastic interpolant (SI). We pointed out the connection between DDPM and SDE-SBGM and how the training objective of SDE-SBGM can be derived from that of DDPM. We highlighted the difference and similarity between SDE-SBGM and SI; that is, while SDE-SBGM is a stochastic flow-based approach and SI is a deterministic flow-based approach, both of the methods admit a least-squares type training objective. We investigated the effect of the number of discretization steps in the training and sampling performance by applying them to a two-dimensional four-mode Gaussian mixture problem and a ten-dimensional four-mode Gaussian mixture problem. Our result shows that as the number of discretization steps gets bigger, it becomes harder for SDE-SBGM to achieve better performance in these two cases. In contrast, SIs do not have this problem. We suspect that this is because SIs admit a closed-form solution for Gaussian mixture problems, and this could potentially make it easier for SIs to learn the better velocity field. We caution that the conclusion we drew from these two problems may not be generalized to a wider range of problems in that although 10 dimensions is high in the physical sense, it is incomparably small with respect to problems in, for instance,

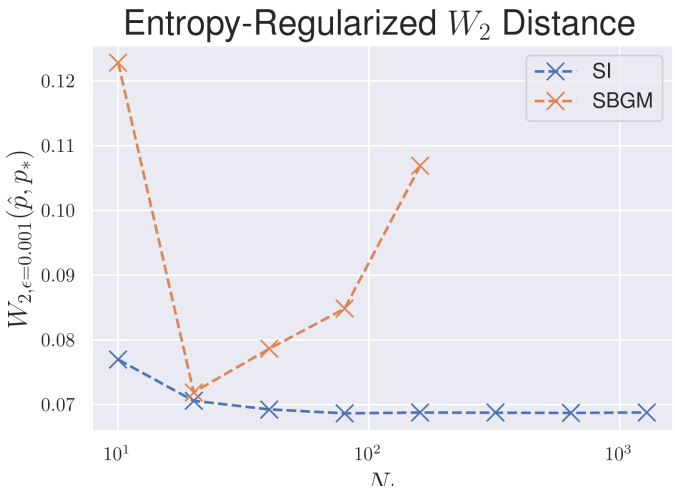

Fig. 4.4: The (discrete) entropy-regularized 2-Wasserstein distance between the approximate density and the true density for the 2-dimensional case. Here $\epsilon = 0.001$.

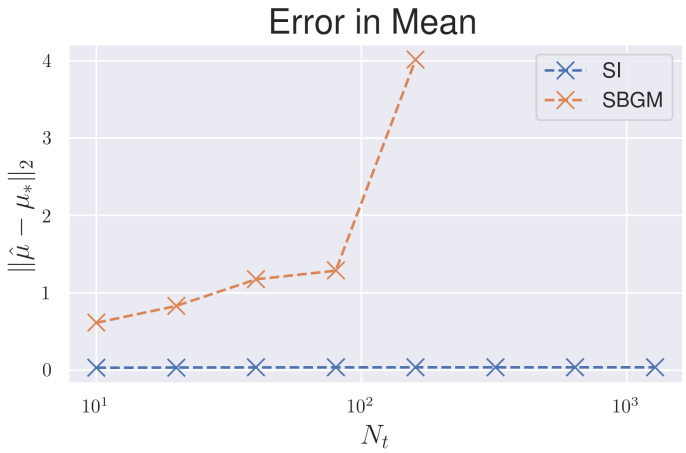

Fig. 4.5: Approximation error in mean for the 10-dimensional case.

computer vision and natural language processing, which are the scenarios that these methods are designed for in the first place. More fair suitable comparisons are in need, but due to the constraint in computational resources and the time limit in finishing this writeup, we are not able to conduct further investigations into these much more complicated problems. However, what we may conclude is SIs are likely to work better than SDE-SBGMs in relatively low-dimension Gaussian mixture type problems.

REFERENCES

[1] M. S. Albergo and E. Vanden-Eijnden, *Building normalizing flows with stochastic interpolants*, arXiv preprint arXiv:2209.15571, (2022).

[2] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*, Advances in neural information processing systems, 31 (2018).

[3] M. Cuturi, *Sinkhorn distances: Lightspeed computation of optimal transport*, Advances in neural information processing systems, 26 (2013).

[4] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, *Ffjord: Free-form continuous dynamics*

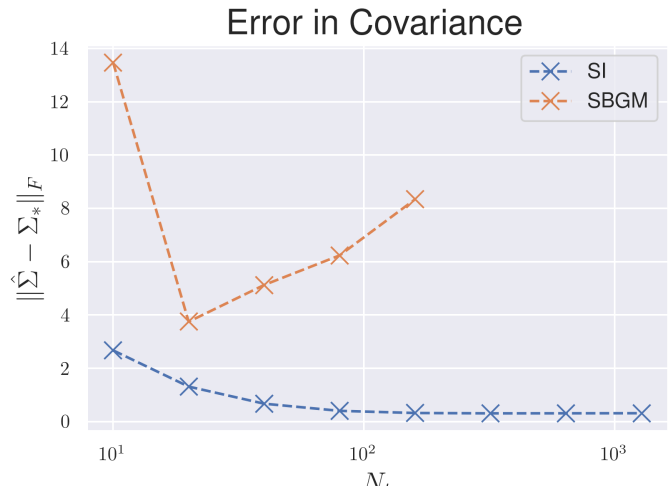

Fig. 4.6: Approximation error in covariance for the 10-dimensional case.

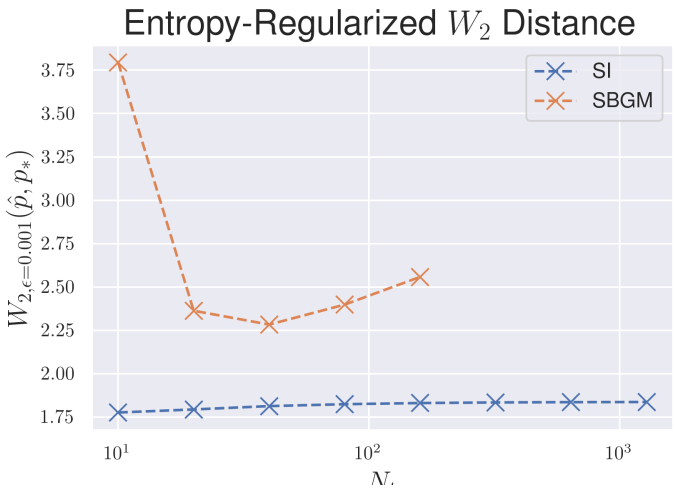

Fig. 4.7: The (discrete) entropy-regularized 2-Wasserstein distance between the approximate density and the true density for the 10-dimensional case. Here $\epsilon = 0.001$.

for scalable reversible generative models, arXiv preprint arXiv:1810.01367, (2018).

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

[6] C. Luo, *Understanding diffusion models: A unified perspective*, arXiv preprint arXiv:2208.11970, (2022).

[7] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[8] G. Peyré, M. Cuturi, et al., *Computational optimal transport*, Center for Research in Economics and Statistics Working Papers, (2017).

[9] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-based generative modeling through stochastic differential equations*, arXiv preprint arXiv:2011.13456, (2020).