Maximum Likelihood Training of Implicit Nonlinear Diffusion Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Whereas diverse variations of diffusion models exist, expanding the linear diffusion into a nonlinear diffusion process is investigated only by a few works. The nonlinearity effect has been hardly understood, but intuitively, there would be more promising diffusion patterns to optimally train the generative distribution towards the data distribution. This paper introduces such a data-adaptive and nonlinear diffusion process for score-based diffusion models. The proposed Implicit Nonlinear Diffusion Model (INDM) learns the nonlinear diffusion process by combining a normalizing flow and a diffusion process. Specifically, INDM implicitly constructs a nonlinear diffusion on the *data space* by leveraging a linear diffusion on the *latent space* through a flow network. This flow network is the key to forming a nonlinear diffusion as the nonlinearity fully depends on the flow network. This flexible nonlinearity is what improves the learning curve of INDM to nearly MLE training, compared against the non-MLE training of DDPM++, which turns out to be a special case of INDM with the identity flow. Also, training the nonlinear diffusion empirically yields a sampling-friendly latent diffusion that the sample trajectory of INDM is closer to an optimal transport than the trajectories of previous research. In experiments, INDM achieves the state-of-the-art FID on CelebA.

1 Introduction

Diffusion models have recently achieved success on a task of sample generation, and some works [1, 2] claim the state-of-the-art performance over Generative Adversarial Networks (GAN) [3]. This success is highlighted particularly in likelihood-based models, including normalizing flows [4], autoregressive models [5], and Variational Auto-Encoders (VAE) [6]. Moreover, this success is noteworthy because it is achieved merely using linear diffusion processes, such as Variance Exploding (VE) Stochastic Differential Equation (SDE) [7] and Variance Preserving (VP) SDE [8].

This paper expands these linear diffusions of VE/VP SDEs to a data-adaptive trainable nonlinear diffusion. To motivate the expansion, though there are structural similarities between diffusion models and VAEs, the inference part of a linear diffusion process has not been trained while its counterpart of VAE (the encoder) is being trained. We introduce Implicit Nonlinear Diffusion Models (INDM) to train its *forward* diffusion SDE, which corresponds to training the inference part in diffusion models. INDM constructs nonlinearity of data diffusion by transforming a linear diffusion process on the latent space back to the data space, where the transformation is implemented with a normalizing flow.

INDM provides two main advantages in its logical mechanism over the existing models. First, this implicit way of modeling nonlinear data diffusion achieves fast and tractable optimization. Second, INDM learns both drift and diffusion terms of the nonlinear forward SDE unlike the previous drift-only learning. Additionally, INDM demonstrated two empirical advantages over the existing models. First, INDM shows a reduced variational gap to more closely approximate the MLE training. Second,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

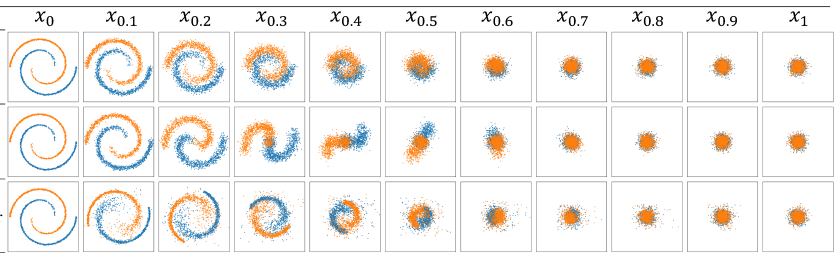


Figure 1: Examples of linear (top row) and nonlinear (middle/bottom rows) diffusion processes.

the sampling from the probability flow Ordinary Differential Equation (ODE) [1] of INDM is robust on the tolerance of ODE solvers, which implies that INDM constructs a sampling-friendly generative process. Eventually, in experiments, we achieve the state-of-the-art performance on CelebA [9].

2 Preliminary

A diffusion model is constructed with bidirectional *forward* and *reverse* stochastic processes.

Forward and Reverse Diffusions A forward diffusion process diffuses an input data variable, $\mathbf{x}_0 \sim p_r$, to a noise variable, and the corresponding reverse diffusion process [10] of this forward diffusion denoises a noise variable to regenerate the input variable. The forward diffusion is fully described by an SDE of $\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t, t)\,\mathrm{d}t + \mathbf{G}(\mathbf{x}_t, t)\,\mathrm{d}\boldsymbol{\omega}_t$, and the corresponding reverse SDE becomes $\mathrm{d}\mathbf{x}_t = \left[\mathbf{f}(\mathbf{x}_t, t) - \mathrm{div}(\mathbf{G}\mathbf{G}^T)(\mathbf{x}_t, t) - (\mathbf{G}\mathbf{G}^T)(\mathbf{x}_t, t)\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\right] \mathrm{d}\bar{t} + \mathbf{G}(\mathbf{x}_t, t)\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$. Here, $\boldsymbol{\omega}_t \in \mathbb{R}^d$ is an abstraction of a random walk process with independent increments, and $\mathrm{d}\bar{\boldsymbol{\omega}}_t$ is the standard Wiener processes with backwards in time, where d is the data dimension.

Generative Diffusion Having that the drift ($\mathbf{f} \in \mathbb{R}^d$) and the diffusion ($\mathbf{G} \in \mathbb{R}^{d\times d}$) terms are given a-priori, diffusion models [1] estimate the data score, $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$, with the score network, $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$. By plugging the score network in the data score, we obtain another diffusion process, called the *generative* SDE, described by $\mathrm{d}\mathbf{x}_t^{\boldsymbol{\theta}} = \left[\mathbf{f}(\mathbf{x}_t^{\boldsymbol{\theta}}, t) - \mathrm{div}(\mathbf{G}\mathbf{G}^T)(\mathbf{x}_t^{\boldsymbol{\theta}}, t) - (\mathbf{G}\mathbf{G}^T)(\mathbf{x}_t^{\boldsymbol{\theta}}, t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t^{\boldsymbol{\theta}}, t)\right] \mathrm{d}\bar{t} + \mathbf{G}(\mathbf{x}_t^{\boldsymbol{\theta}}, t)\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$. Starting from a prior distribution of $\mathbf{x}_T^{\boldsymbol{\theta}} \sim \pi$ and solving the SDE time backwards, Song et al. [1] constructs the generative stochastic process of $\{\mathbf{x}_t^{\boldsymbol{\theta}}\}_{t=0}^T$ that perfectly reconstructs the reverse process of $\{\mathbf{x}_t\}_{t=0}^T$ under two conditions. The first condition is $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$; and the second condition is $\mathbf{x}_T \sim \pi$. We define a generative distribution, $p_{\boldsymbol{\theta}}$, as the distribution of $\mathbf{x}_0^{\boldsymbol{\theta}}$.

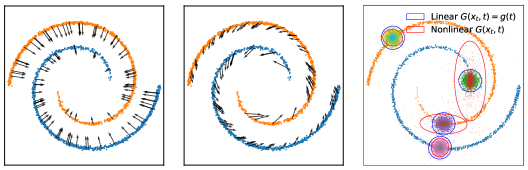
Score Estimation The diffusion model estimates the data score with the score network by minimizing the denoising score loss [1], given by $\mathcal{L}(\{\mathbf{x}_t\}_{t=0}^T, \lambda; \boldsymbol{\theta}) = \int_0^T \lambda(t)\mathbb{E}_{\mathbf{x}_0,\mathbf{x}_t}[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{0t}(\mathbf{x}_t|\mathbf{x}_0)\|_2^2]\,\mathrm{d}t$, where $p_{0t}(\mathbf{x}_t|\mathbf{x}_0)$ is a transition probability of $\mathbf{x}_t$ given $\mathbf{x}_0$; and λ is the weighting function that determines the level of contribution for each diffusion time. When $\mathbf{G}(\mathbf{x}_t, t) = g(t)$, Song et al. [11], Huang et al. [12] proved that this loss with the likelihood weighting ($\lambda = g^2$) turns out to be a variational bound of the negative log-likelihood: $\mathbb{E}_{\mathbf{x}_0}[-\log p_{\boldsymbol{\theta}}(\mathbf{x}_0)] \leq \mathcal{L}(\{\mathbf{x}_t\}_{t=0}^T, g^2; \boldsymbol{\theta}) - \mathbb{E}_{\mathbf{x}_T}[\log \pi(\mathbf{x}_T)]$, up to a constant, see Appendix A.1 for a detailed discussion.

Choice of Drift ($\mathbf{f}$) and Diffusion ($\mathbf{G}$) Terms The original diffusion model strictly limits the scope of diffusion process to be a family of linear diffusions that $\mathbf{f}$ is a linear function of $\mathbf{x}_t$ and $\mathbf{G}$ is an identity matrix multiplied by a t-function. For instance, VESDE [1] satisfies $\mathbf{f} \equiv 0$ with $\mathbf{G} = \sqrt{\mathrm{d}\sigma^2(t)/\,\mathrm{d}t}\mathbf{I}$ and VPSDE [1] satisfies $\mathbf{f} = -\frac{1}{2}\beta(t)\mathbf{x}_t \propto \mathbf{x}_t$ with $\mathbf{G} = \sqrt{\beta(t)}\mathbf{I}$. Few concurrent works have expanded linear diffusions to nonlinear diffusions by 1) transforming the data diffusion into a latent diffusion using VAE in LSGM [13], 2) approximating the nonlinear diffusion to a discretized diffusion with a nonlinear drift using a flow model in DiffFlow [14], and 3) reformulating the diffusion model into a Schrodinger Bridge Problem (SBP) [15–17]. We further analyze these approaches in Section 5.

3 Motivation of Nonlinear Diffusion Process

Figure 1 illustrates various diffusion processes on a spiral toy dataset. The top row represents the process of VPSDE. The diffused manifold keeps its overall structure of the initial data manifold during the data deformation procedure to $\mathcal{N}(0, \mathbf{I})$. Concretely, the drift vector field illustrated in Figure 2-(a) as black arrows presents that VPSDE *linearly* deforms its data distribution.

Unlike the linear diffusion, the middle row of Figure 1 with a nonlinear drift shows that the data is not linearly deformed to $\mathcal{N}(0, \mathbf{I})$. The corresponding vector field is illustrated in Figure 2-(b), in which two distinctive components (orange/blue) are forced to separate each other. The nonlinearity of the drift term represented as rotating black arrows is the source of such nonlinear deformation at the intermediate steps, $\mathbf{x}_{0.2} \sim \mathbf{x}_{0.6}$. When it comes to the diffusion term, the last row of Figure 1 presents the process with nonlinear $\mathbf{G}$. Figure 2-(c) illustrates the covariance matrices of the noise distribution at $t = 0$ with linear and nonlinear diffusion terms, where the noise distribution induced by the diffusion term is $\mathcal{N}(0, \mathbf{G}(\mathbf{x}_t, t)\mathbf{G}^T(\mathbf{x}_t, t))$[1]. It shows the non-diagonal and data-dependent covariances of $\mathbf{G}\mathbf{G}^T$ in red ellipses of a nonlinear diffusion term, and the isotropic blue circles of linear diffusions.



(a) Linear $\mathbf{f} \propto \mathbf{x}_t$ (b) Nonlinear $\mathbf{f}$ (c) Nonlinear $\mathbf{G}$

Figure 2: Vector fields on various SDEs at $t = 0$.

4 Implicit Nonlinear Diffusion Model

Imposing nonlinearity with the explicit parametrization of $\mathbf{f}_\phi$ and $\mathbf{G}_\phi$ incurs a significant increase in the training time [14, 17] because the transition probability of $p_{0t}(\mathbf{x}_t|\mathbf{x}_0)$ would be intractable. We, therefore, parametrize $\mathbf{f}_\phi$ and $\mathbf{G}_\phi$ in an *implicit* way for the fast and tractable optimization. Implicit parametrization can be achieved by leveraging a linear diffusion on the latent space to the data space through an invertible transformation. Figure 3 provides a schematic view of our model. We choose a normalizing flow as the invertible transformation between the data and the latent spaces.

Latent Diffusion Processes Let us define $\mathbf{z}_0^\phi$ to be a transformed latent variable, $\mathbf{z}_0^\phi = \mathbf{h}_\phi(\mathbf{x}_0)$, where $\mathbf{h}_\phi$ is a transformation of the normalizing flow. Then, a forward linear diffusion,

$$\mathrm{d}\mathbf{z}_t^\phi = -\frac{1}{2}\beta(t)\mathbf{z}_t^\phi \,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\omega}_t,$$

starting at $\mathbf{z}_0^\phi = \mathbf{h}_\phi(\mathbf{x}_0)$ with $\mathbf{x}_0 \sim p_r$ describes the forward diffusion process on the latent space. The corresponding reverse latent diffusion is given by $\mathrm{d}\mathbf{z}_t^\phi = [-\frac{1}{2}\beta(t)\mathbf{z}_t^\phi - g^2(t)\nabla_{\mathbf{z}_t^\phi} \log p_t^\phi(\mathbf{z}_t^\phi)]\,\mathrm{d}\bar{t} + g(t)\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$, where p_t^ϕ is the probability distribution of $\mathbf{z}_t^\phi$.

Forward Data Diffusion Since the transformation is invertible, we could define random variables on the data space by transforming the latent linear diffusion back to the data space: $\mathbf{x}_t^\phi = \mathbf{h}_\phi^{-1}(\mathbf{z}_t^\phi)$ for any $t \in [0, T]$. Then, from the Ito's formula [18], the dynamics of a stochastic process, $\{\mathbf{x}_t^\phi\}_{t=0}^T$, is governed by an SDE of

$$\mathrm{d}\mathbf{x}_t^\phi = \mathbf{f}_\phi(\mathbf{x}_t^\phi, t)\,\mathrm{d}t + \mathbf{G}_\phi(\mathbf{x}_t^\phi, t)\,\mathrm{d}\boldsymbol{\omega}_t,$$

starting with $\mathbf{x}_0^\phi = \mathbf{h}_\phi^{-1}(\mathbf{z}_0^\phi)$. Since $\mathbf{z}_0^\phi$ is the transformed variable of $\mathbf{z}_0^\phi = \mathbf{h}_\phi(\mathbf{x}_0)$, we obtain $\mathbf{x}_0^\phi = \mathbf{h}_\phi^{-1}(\mathbf{h}_\phi(\mathbf{x}_0)) = \mathbf{x}_0 \sim p_r$, which follows the original data distribution. Therefore, the forward data diffusion (illustrated as a green path in Figure 3) leveraged from the latent space is *a nonlinear diffusion process that starts from the data variable*. Here, we derive drift and diffusion terms to be

$$\begin{cases} \mathbf{f}_\phi(\mathbf{x}_t^\phi, t) & = -\frac{1}{2}\beta(t)\big[\nabla \mathbf{h}_\phi(\mathbf{x}_t^\phi)\big]^{-1}\mathbf{h}_\phi(\mathbf{x}_t^\phi) + \frac{1}{2}g^2(t)\mathrm{tr}\big[\nabla^2 \mathbf{h}_\phi^{-1}\big](\mathbf{h}_\phi(\mathbf{x}_t^\phi)) \\ \mathbf{G}_\phi(\mathbf{x}_t^\phi, t) & = g(t)\big[\nabla \mathbf{h}_\phi(\mathbf{x}_t^\phi)\big]^{-1}, \end{cases}$$

see Appendix A.2 for the derivation. This diffusion process is an *induced diffusion* on the data space that is defined indirectly by transforming a linear diffusion on the latent space back to the data space. We emphasize that the nonlinear diffusion collapses to the vanilla linear diffusion of VPSDE (or VESDE) when $\mathbf{h}_{\phi_{id}} = id$ is an identity transformation.

Generative Data Diffusion A diffusion model estimates the forward latent score, $\mathbf{s}_\phi(\cdot, t) := \nabla \log p_t^\phi(\cdot)$, with the score network, $\mathbf{s}_\theta(\cdot, t)$, in order to mimic the forward linear diffusion on the latent space with the score network. Then, the generative SDE on the latent space becomes $\mathrm{d}\mathbf{z}_t^\theta = \big[-\frac{1}{2}\beta(t)\mathbf{z}_t^\theta - g^2(t)\mathbf{s}_\theta(\mathbf{z}_t^\theta, t)\big]\,\mathrm{d}\bar{t} + g(t)\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$ with a starting variable of $\mathbf{z}_T^\theta \sim \pi$. Thus, the

[1]The covariance is $\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}_{\mathbf{x}_{t+\mathrm{d}t}|\mathbf{x}_t}[(\mathbf{x}_{t+\mathrm{d}t} - \mathbf{x}_t - \mathbf{f}\,\mathrm{d}t)(\mathbf{x}_{t+\mathrm{d}t} - \mathbf{x}_t - \mathbf{f}\,\mathrm{d}t)^T] = \mathbf{G}(\mathbf{x}_t, t)\mathbf{G}^T(\mathbf{x}_t, t)$.

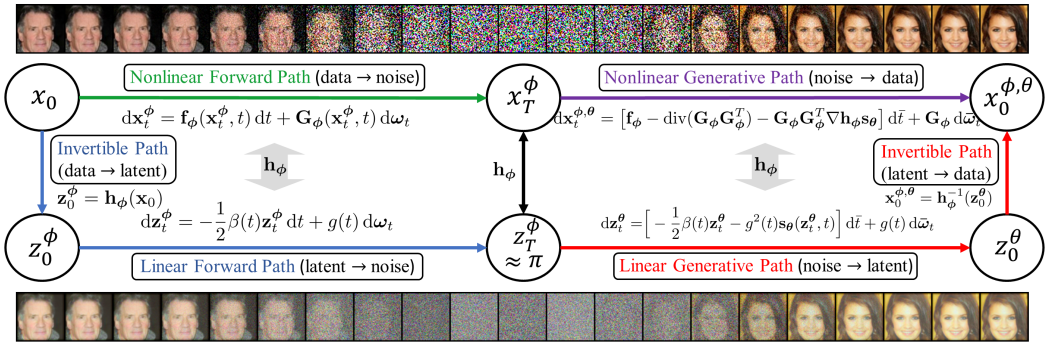


Figure 3: INDM attains a ladder structure between the data space and the latent space. The latent vector is visualized by normalizing the latent value.

stochastic process, $\{\mathbf{x}_t^{\phi,\boldsymbol{\theta}}\}_{t=0}^T$, defined by $\mathbf{x}_t^{\phi,\boldsymbol{\theta}} = \mathbf{h}_\phi^{-1}(\mathbf{z}_t^{\boldsymbol{\theta}})$ for any $t \in [0, T]$ becomes a generative nonlinear process on the data space (purple path in Figure 3), where the generative SDE is given by

$$\mathrm{d}\mathbf{x}_t^{\phi,\boldsymbol{\theta}} = \big[\mathbf{f}_\phi - \mathrm{div}(\mathbf{G}_\phi\mathbf{G}_\phi^T) - (\mathbf{G}_\phi\mathbf{G}_\phi^T\nabla\mathbf{h}_\phi)\mathbf{s}_{\boldsymbol{\theta}}\big(\mathbf{h}_\phi(\mathbf{x}_t^{\phi,\boldsymbol{\theta}}), t\big)\big]\,\mathrm{d}\bar{t} + \mathbf{G}_\phi\,\mathrm{d}\bar{\boldsymbol{\omega}}_t.$$

Score Estimation Normalizing flows enable tractable training of nonlinear diffusions, and we train INDM with Negative Evidence Lower Bound (NELBO) in Theorem 1. See Appendix B.6 for proof.

Theorem 1. *Suppose that $p_{\phi,\boldsymbol{\theta}}$ is the likelihood of a generative random variable $\mathbf{x}_0^{\phi,\boldsymbol{\theta}}$. Then, the negative log-likelihood is upper bounded by*

$$\mathbb{E}_{\mathbf{x}_0}\big[-\log p_{\phi,\boldsymbol{\theta}}(\mathbf{x}_0)\big] \le \mathcal{L}\big(\{\mathbf{x}_t\}_{t=0}^T, g^2; \{\phi,\boldsymbol{\theta}\}\big),$$

where

$$\mathcal{L}\big(\{\mathbf{x}_t\}_{t=0}^T, g^2; \{\phi,\boldsymbol{\theta}\}\big) = -\mathbb{E}_{\mathbf{x}_0}\big[\log\big|\det\big(\nabla\mathbf{h}_\phi(\mathbf{x}_0)\big)\big|\big] + \mathcal{L}\big(\{\mathbf{z}_t^\phi\}_{t=0}^T, g^2; \boldsymbol{\theta}\big) - \mathbb{E}_{\mathbf{z}_T^\phi}\big[\log\pi(\mathbf{z}_T^\phi)\big],$$

with $\mathcal{L}\big(\{\mathbf{z}_t^\phi\}_{t=0}^T, g^2; \boldsymbol{\theta}\big) := \frac{1}{2}\int_0^T g^2(t)\mathbb{E}_{\mathbf{z}_0^\phi,\mathbf{z}_t^\phi}\big[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t^\phi, t) - \nabla\log p_{0t}(\mathbf{z}_t^\phi|\mathbf{z}_0^\phi)\|_2^2\big]\,\mathrm{d}t$, up to a constant. Here, $p_{0t}(\mathbf{z}_t^\phi|\mathbf{z}_0^\phi)$ is the transition probability of the forward linear diffusion on the latent space.

NELBO in Theorem 1 is equivalent to $D_{KL}(\boldsymbol{\mu}_\phi\|\boldsymbol{\nu}_{\phi,\boldsymbol{\theta}})$, where $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\nu}_{\phi,\boldsymbol{\theta}}$ are the path measures of the forward diffusion (green path in Figure 3) and generative diffusion (purple path in Figure 3) on the data space. This means that even though the flow network only acts at the initial diffusion time ($t = 0$) to transform $\mathbf{x}_0$ into $\mathbf{z}_0^\phi$ (or $\mathbf{z}_0^{\boldsymbol{\theta}}$ into $\mathbf{x}_0^{\phi,\boldsymbol{\theta}}$), the effect of flow training covers the whole forward and generative paths across all diffusion time. Algorithm 1 describes the line-by-line algorithm for training. We train the flow network with $\mathcal{L}_f = \mathcal{L}\big(\{\mathbf{x}_t\}_{t=0}^T, g^2; \{\phi,\boldsymbol{\theta}\}\big)$ and the score network with $\mathcal{L}_s = \mathcal{L}\big(\{\mathbf{x}_t\}_{t=0}^T, \lambda; \{\phi,\boldsymbol{\theta}\}\big)$, separately. These two losses coincide if $\lambda = g^2$.

Algorithm 1 Implicit Nonlinear Diffusion Model

repeat
  Get latent with flow by $\mathbf{z}_0^\phi = \mathbf{h}_\phi(\mathbf{x}_0)$ for $\mathbf{x}_0 \sim p_r$
  Compute $-\mathbb{E}_{\mathbf{x}_0}\big[\log\big|\det\big(\nabla\mathbf{h}_\phi(\mathbf{x}_0)\big)\big|\big]$
  Get diffused latents $\{\mathbf{z}_t^\phi\}_{t=0}^T$ with a linear SDE
  Compute $\mathcal{L}\big(\{\mathbf{z}_t^\phi\}_{t=0}^T, \lambda; \boldsymbol{\theta}\big) - \mathbb{E}_{\mathbf{z}_T^\phi}\big[\log\pi(\mathbf{z}_T^\phi)\big]$
  Compute flow loss $\mathcal{L}_f = \mathcal{L}\big(\{\mathbf{x}_t\}_{t=0}^T, g^2; \{\phi,\boldsymbol{\theta}\}\big)$
  Update $\phi \leftarrow \phi - \frac{\partial\mathcal{L}_f}{\partial\phi}$
  Compute $\mathcal{L}\big(\{\mathbf{z}_t^\phi\}_{t=0}^T, g^2; \boldsymbol{\theta}\big) - \mathbb{E}_{\mathbf{z}_T^\phi}\big[\log\pi(\mathbf{z}_T^\phi)\big]$
  Compute score loss $\mathcal{L}_s = \mathcal{L}\big(\{\mathbf{x}_t\}_{t=0}^T, \lambda; \{\phi,\boldsymbol{\theta}\}\big)$
  Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\partial\mathcal{L}_s}{\partial\boldsymbol{\theta}}$
until converged

Sampling Starting from a prior distribution (π), we denoise $\mathbf{z}_T^{\boldsymbol{\theta}} \sim \pi$ to $\mathbf{z}_0^{\boldsymbol{\theta}}$ by solving the generative process backward on the latent space. Afterwards, we transform the fully denoised latent of $\mathbf{z}_0^{\boldsymbol{\theta}}$ back to the data space $\mathbf{x}_0^{\phi,\boldsymbol{\theta}} = \mathbf{h}_\phi^{-1}(\mathbf{z}_0^{\boldsymbol{\theta}})$, as in the red path of Figure 3. Note that sampling through the red path is much faster than the purple path as the red path feed-forwards the flow only once.

5 Related Work: Comparison to Data Diffusion Learning Models

LSGM [13] also puts a linear diffusion on the latent space like INDM, but it uses VAE instead of the flow network, so the forward diffusion process on the *data* space (green path in Figure 3) is not

Table 1: Comparison of INDM with previous works.

Model	Nonlinear Diffusion	Implemented Data Diffusion	Latent Diffusion	Nonlinear $\mathbf{f}$-Modeling	Nonlinear $\mathbf{G}$-Modeling	Explicit $\mathbf{f}$&$\mathbf{G}$ Derived	Training Complexity	Sampling Cost
DDPM++	✗	Continuous	✗	✗	✗	✓	$O(1)$	$\downarrow$
LSGM	✗	✗	Continuous	✗	✗	✗	$O(1)$	$\downarrow$
SBP	△	Discrete	✗	Explicit	✗	✓	$O(N)$	$\downarrow$
DiffFlow	✓	Discrete	✗	Explicit	✗	✓	$O(N)$	$\uparrow$
INDM	✓	Continuous	Continuous	Implicit	Implicit	✓	$O(1)$	$\downarrow$

guaranteed to exist as the inverse of an encoder map does not exist in general, see Appendix C.1 for the details. Therefore, LSGM optimizes the divergence between the forward and generative diffusion processes only on the latent space, not necessarily on the data space, in contrast to INDM which optimizes the divergence between the bidirectional diffusion paths (green and purple paths in Figure 3) on the data space, directly. This leads LSGM to lose its theoretical ground for categorizing it as a family of data diffusion learning models. Although the LSGM's forward data diffusion could be visualized by leveraging the forward latent diffusion with the decoder as suggested in Vahdat et al. [13], it needs to be understood as a *reconstructed* data diffusion, which starts from the reconstructed variable. This reconstructed variable is distant from the data variable at the initial stage of training, so this discrepancy leads to the distinctive learning curves of INDM and LSGM, see Table 1.

DiffFlow [14] explicitly models the drift term, $\mathbf{f}_{\boldsymbol{\phi}}$, as a flow network, so the forward diffusion becomes $\mathrm{d}\mathbf{x}_t = \mathbf{f}_{\boldsymbol{\phi}}\,\mathrm{d}t + g\,\mathrm{d}\boldsymbol{\omega}_t$. Unlike INDM, modeling a nonlinear diffusion term ($\mathbf{G}_{\boldsymbol{\phi}}$) in DiffFlow is a nontrivial task as the transition probability is no longer a Gaussian distribution [1, 8], nor a Gamma distribution [19]. A challenge of DiffFlow is the slow training because the estimation of the denoising loss, $\mathbb{E}_{\mathbf{x}_0,\mathbf{x}_t^{\boldsymbol{\phi}}}[\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t^{\boldsymbol{\phi}},t) - \nabla \log p_{0t}(\mathbf{x}_t^{\boldsymbol{\phi}}|\mathbf{x}_0)\|_2^2]$, requires a pair of heavy computations: 1) sampling from $\mathbf{x}_t^{\boldsymbol{\phi}}$, and 2) computation of $\nabla \log p_{0t}(\mathbf{x}_t^{\boldsymbol{\phi}}|\mathbf{x}_0)$. Intractable transition probability, $p_{0t}(\mathbf{x}_t^{\boldsymbol{\phi}}|\mathbf{x}_0)$, is the major bottleneck of such slow training. One potential approach to stepside computing $\nabla \log p_{0t}$ is using another loss for training, such as the sliced score loss [20]. DiffFlow optimizes its network with DDPM-style loss [8], rather than the intractable denoising score loss, by discretizing the continuous diffusion into a discrete diffusion of N variables, see Appendix C.2. Under the discretization, however, the forward sampling of $\mathbf{x}_t^{\boldsymbol{\phi}}$ (green path in Figure 3) still takes $O(N)$ flow evaluations for every network update as $\mathbf{f}_{\boldsymbol{\phi}}$ is explicitly modeled with normalizing flows. Having that flow evaluation is generally more expensive than score evaluation given same number of parameters, a fast sampling is achievable if we reduce N, but it hurts the flexibility of a diffusion process, so DiffFlow needs to handle the trade-off between training speed and model flexibility. On the other hand, INDM allows to train continuous nonlinear diffusions with only a single flow evaluation ($O(1)$) per network update, so INDM is free from such trade-off. Analogously, the reverse sampling of DiffFlow requires $O(N)$ flow evaluations, contrastive to INDM with a single flow evaluation.

SBP [15–17] is a problem of $\min_{\boldsymbol{\rho}_{\boldsymbol{\theta}} \in \mathcal{P}(p_r,\pi)} D_{KL}(\boldsymbol{\rho}_{\boldsymbol{\theta}} \| \boldsymbol{\mu})$, where $\mathcal{P}(p_r,\pi)$ is the collection of path measure with p_r and π as its marginal distributions at $t=0$ and $t=T$, respectively. It is a bi-constrained optimization problem as any path measure on a search space should satisfy boundary conditions at both $t=0$ and $t=T$ with $p_0 = p_r$ and $p_T = \pi$, respectively. $\boldsymbol{\mu}$ is the reference measure of linear diffusion with $\mathrm{d}\mathbf{x}_t = \mathbf{f}(\mathbf{x}_t,t)\,\mathrm{d}t + g(t)\boldsymbol{\omega}_t$; and the forward and reverse SDEs of $\boldsymbol{\rho}_{\boldsymbol{\theta}}$ are $\mathrm{d}\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t,t) + g^2(t)\nabla \log \Psi(\mathbf{x}_t,t)]\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\omega}_t$ and $\mathrm{d}\mathbf{x}_t = [\mathbf{f}(\mathbf{x}_t,t) - g^2(t)\nabla \log \hat{\Psi}(\mathbf{x}_t,t)]\,\mathrm{d}\bar{t} + g(t)\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$, respectively, where $(\Psi,\hat{\Psi})$ is the solution of a coupled PDE, called Hopf-Cole transform [21]. Solving this coupled PDE is intractable, so the estimation target of SBP is $\nabla \log \Psi$ and $\nabla \log \hat{\Psi}$. As $\mathbf{f}$ and g are assumed to be linear functions, the nonlinearity of SBP is fully determined by $(\Psi,\hat{\Psi})$.

Analogous to DiffFlow, sampling from $\mathbf{x}_t$ in SBP needs long time to solve the forward SDE, and existing works on SBP discretize the continuous diffusion for the feasible training, which causes the aforementioned trade-off of DiffFlow. Few works [16, 17] in SBP detour this issue using experience replay memory, but the training time still diverges by N, in contrast to INDM with invariant training time by N, see Table 18 of the Appendix. Also, the experience replay memory could potentially bring a suboptimal training or the training instability. Aside from the training aspects, the KL minimization puts the global optimal nonlinear diffusion, $\boldsymbol{\rho}_{\boldsymbol{\theta}^*}$, near a neighborhood of the linear diffusion, $\boldsymbol{\mu}$. In other words, the optimal $\boldsymbol{\rho}_{\boldsymbol{\theta}^*}$ is the closest path measure on $\mathcal{P}(p_r,\pi)$ to $\boldsymbol{\mu}$, so the inferred nonlinear diffusion would be the *most* linear diffusion on the space of $\mathcal{P}(p_r,\pi)$. For the demonstration, we illustrate

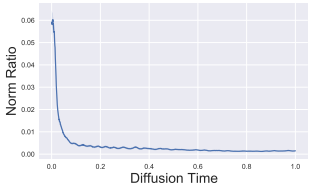


Figure 4: Norm Ratio of SBP.

$\|g^2(t)\nabla \log \Psi(\mathbf{x}_t, t)\Delta t\|_2 / \|g(t)\Delta \boldsymbol{\omega}_t\|_2$ in Figure 4. We used the released checkpoint of SB-FBSDE [17], which is an algorithm for solving SBP, trained with VESDE on CIFAR-10. As $\mathbf{f} \equiv 0$ in VESDE, this norm ratio measures how much nonlinearity is counted on a diffusion trajectory, compared to the linear effect. Figure 4 shows that the ratio approaches to zero except at the small range around $t \approx 0$, meaning that the nonlinear effect is virtually ignorable than the linear effect. Therefore, Figure 4 implies that the diffusion process is nearly linear in most of diffusion time. We give detailed discussion of SBP in Appendix C.3.

6 Discussion

6.1 Better Optimization

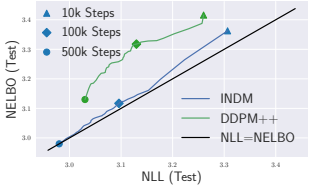
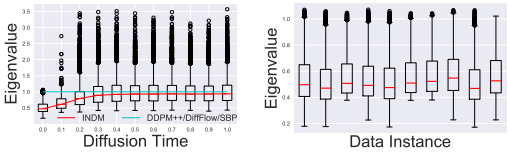


Figure 5: Training Curve.

Having that DDPM++ is a collapsed INDM with a specific choice of $\phi = \phi_{id}$, the difference of DDPM++ and INDM lies in whether to train ϕ or not. This trainable transformation turns out to be the key for the better optimization of INDM as evidenced in Figure 5. It shows that 1) INDM achieves a faster training, compared to DDPM++, and 2) the learning curve of INDM is asymptotically MLE training on CIFAR-10. With $\mathbf{s}_{\boldsymbol{\phi}}(\mathbf{z}_t^{\boldsymbol{\phi}}, t) := \nabla \log p_t^{\boldsymbol{\phi}}(\mathbf{z}_t^{\boldsymbol{\phi}})$, the optimization of INDM NELBO, $\int_0^T g^2(t)\mathbb{E}[\|\mathbf{s}_{\boldsymbol{\phi}} - \mathbf{s}_{\boldsymbol{\theta}}\|_2^2]\,\mathrm{d}t/2 + D_{KL}(p_T^{\boldsymbol{\phi}}\|\pi) - \mathbb{E}[\log\det(\nabla \mathbf{h}_{\boldsymbol{\phi}})]$, w.r.t $\boldsymbol{\phi}$ updates the forward score, $\mathbf{s}_{\boldsymbol{\phi}}$, towards the generative score, $\mathbf{s}_{\boldsymbol{\theta}}$. Therefore, on top of score training that forces $\mathbf{s}_{\boldsymbol{\theta}} \to \mathbf{s}_{\boldsymbol{\phi}}$, INDM drags $\mathbf{s}_{\boldsymbol{\phi}}$ pointing towards $\mathbf{s}_{\boldsymbol{\theta}}$. This *bidirectional* attraction between $\mathbf{s}_{\boldsymbol{\theta}}$ and $\mathbf{s}_{\boldsymbol{\phi}}$ (or between the green and purple paths in Figure 3) is what the flow learning does in the optimization procedure, where DDPM++ has no such bidirectional attraction, see Appendix B.1 for further theoretical analysis.

6.2 Nonlinear Diffusion Term



(a) Eigenvalue by t (b) Eigenvalue by $\mathbf{x}_t$

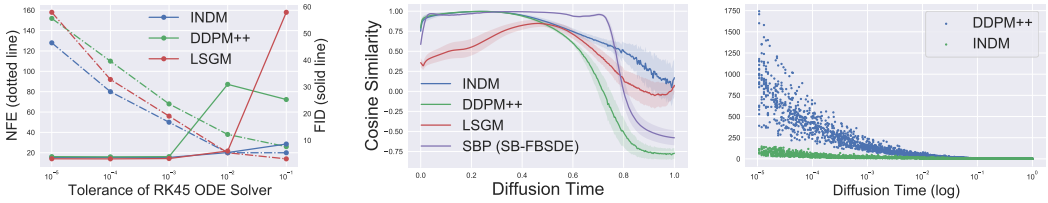
Figure 6: Eigenvalue of $\mathbf{G}\mathbf{G}^T/g^2$ on CIFAR-10.

INDM trains the diffusion term, $\mathbf{G}_{\boldsymbol{\phi}}$, which was fixed across previous research, except LSGM. The exact form of $\mathbf{G}_{\boldsymbol{\phi}}$ in LSGM, however, is not derivable, so we exclude comparing LSGM in this section. As stated in Section 3, the noise distribution of a diffusion process is $\mathcal{N}(0, \mathbf{G}(\mathbf{x}_t, t)\mathbf{G}^T(\mathbf{x}_t, t))$, which is anisotropic by the input data, $\mathbf{x}_t$, and time, t. The influence of diffusion time on this covariance matrix is illustrated in Figure 6-(a). It presents the box plot of the eigenvalue distribution of the (normalized) covariance, $\mathbf{G}_{\boldsymbol{\phi}}(\mathbf{x}_t, t)\mathbf{G}_{\boldsymbol{\phi}}^T(\mathbf{x}_t, t)/g^2(t)$, on CIFAR-10, from $t = 0$ to $t = T$. All the eigenvalues of previous research collapse to one as they share the isotropic covariance matrix, $g^2(t)\mathbf{I}$. On the other hand, the eigenvalues of INDM is dispersive throughout the diffusion time. As the distribution becomes more dispersive, the covariance matrix becomes more unisotropic, and Figure 6-(a) implies that the learnt diffusion process is under a highly nonlinear noise perturbation in a range of large diffusion time. The covariance matrix also depends on the input data, and Figure 6-(b) illustrates the eigenvalue distribution of the covariance at distinctive data instances of $\mathbf{x}_t$ at $t = 0$. The eigenvalue distribution varies by instance, implying that data is diffused inhomogeneously by its location.

6.3 Robust Sampling by Discretization

Figure 7-(a) compares INDM with DDPM++ experimented on CIFAR-10 using VPSDE w.r.t Fréchet Inception Distance (FID) [25] and Number of Function Evaluations (NFE) under various tolerances of the numerical solver. This section analyzes this robustness with a pair of arguments. First, we conjecture this sample robustness arises because the sample trajectory is less curvy in INDM than DDPM++. When a sample trajectory is straight, then a sampler would be robust on the tolerance. Empirically, Figure 7-(b) provides strong evidence of our conjecture. Figure 7-(b) shows the cosine similarity between the direction of the particle trajectory and the direction of the straight line, where the straight direction is pointing towards a vector connecting the starting and the ending particle locations (hence the direction is fixed for each sample trajectory), see Appendix B.3 for the details.

More theoretically, the sample robustness could be analyzed in view of Theorem 1 of De Bortoli et al. [16], which is restated in Theorem 4 of Appendix B.4. Theorem 4 decomposes the Total



(a) FID by Tolerance (b) Cosine Similarity (c) Scaled score norm

Figure 7: (a) Sample performance on CIFAR-10. (b) Cosine similarity of forward diffusion trajectories on CIFAR-10. (c) Score norm scaled by $1 + \|\mathbf{x}_t\|$ (DDPM++) or $1 + \|\mathbf{z}_t\|$ (INDM) on CIFAR-10.

Variation (TV) norm between the sample distribution and the data distribution into two terms: the first term is the error coming from the discrepancy between p_T and π; and the second term is the error originating from the sampling discretization. As the first term is insignificant in general (e.g., $D_{KL}(p_T\|\pi) \approx 10^{-5}$ in VPSDE [8]), the second error fundamentally determines the sampling error. Theorem 4 derives that the sensitivity of the sampling error by discretization is proportional to $A_1^{data} := \sup \frac{\|\nabla \log p_r(\mathbf{x}_0)\|}{1+\|\mathbf{x}_0\|}$, implying that a sampling is robust on the discretization step size when A_1^{data} is small. Not only on the data diffusion, Theorem 4 is directly applicable to the latent diffusion, so $A_1^{latent} := \sup \frac{\|\nabla \log p_0^{\boldsymbol{\phi}}(\mathbf{z}_0^{\boldsymbol{\phi}})\|}{1+\|\mathbf{z}_0^{\boldsymbol{\phi}}\|}$ characterizes the sensitivity of discretization in INDM. Figure 7-(c) illustrates $\frac{\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)\|}{1+\|\mathbf{x}_t\|}$ of DDPM++ and $\frac{\|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{z}_t^{\boldsymbol{\phi}},t)\|}{1+\|\mathbf{z}_t^{\boldsymbol{\phi}}\|}$ of INDM, and it shows that the scaled score norm of INDM consistently lower bounds that of DDPM++ throughout diffusion time. In particular, at $t = 0$, A_1^{latent} of INDM is 12x times smaller than A_1^{data} of DDPM++, implying that the latent diffusion of INDM is more robust on the discretization step size than the data diffusion of DDPM++. Figure 7 combined with Theorem 4 partially elucidates the mechanism of the sample robustness of INDM.

6.4 Relative Energy

Table 2: *Relative* Energy.

Model	*Relative* Energy
DDPM++	1.60
INDM	1.23

Each flow network parameter constructs a different latent trajectory, so training the flow network has the effect of shifting the diffusion bridge. To check if learning the flow network is helpful for the transportation cost or not, recall the Benamou-Brenier formula [26, 23], which is a dual formulation of the Wasserstein distance [23, 27] that the optimal transportation cost is the least kinetic energy out of all admissible transportation plans: $W_2^2(p,q) = \inf_{\{p_t,\mathbf{v}_t\}_t} \left\{ \mathcal{K}(\{p_t,\mathbf{v}_t\}_t); \underbrace{\frac{\partial p_t}{\partial t} + \mathrm{div}(p_t\mathbf{v}_t) = 0}_{\text{continuity equation}}, \underbrace{p_0 = p, p_T = q}_{\text{boundary conditions}} \right\}$, where $\mathcal{K}(\{p_t,\mathbf{v}_t\}_t) := \int_0^T \int p_t(\mathbf{x})\|\mathbf{v}_t(\mathbf{x})\|_2^2 \,\mathrm{d}\mathbf{x}\,\mathrm{d}t/T$ is the kinetic energy of the transportation. The continuity equation (that guarantees the conservation of mass along time transition [28]) and the boundary conditions determine the set of admissible trajectories, and the forward diffusion constructs an admissible trajectory, see Appendix B.5. We *quantify* how much a trajectory is close to the optimal transport as the *relative* energy, given by $R(\boldsymbol{\phi}) = \frac{\mathcal{K}(\{p_t^{\boldsymbol{\phi}},\mathbf{v}_t^{\boldsymbol{\phi}}\}_t)}{W_2^2(p_0^{\boldsymbol{\phi}},p_T^{\boldsymbol{\phi}})}$. Table 2 shows that INDM's latent diffusion is more close to the optimal transport than DDPM++ on CIFAR-10.

7 Experiments

This section quantitatively analyzes suggested INDM on CIFAR-10 [36] and CelebA 64×64. Throughout the experiments, we use NCSN++ for VESDE and DDPM++ for VPSDE [1] as backbones of diffusion models, and a ResNet-based flow model [37, 38] as a backbone of the flow model. See Appendix E for experimental details. We experiment with a pair of weighting functions. One is so-called the likelihood weighting [11] with $\lambda(t) = g^2(t)$, and we denote INDM (NLL) for this weighing. The other is the variance weighting $\lambda(t) = \sigma^2(t)$ with the emphasis on FID, and we denote INDM (FID) for this weighting. We compute Negative Log-Likelihood (NLL) and NELBO for density estimation performances in Bits Per Dimension (BPD) scale. We compute NLL with the uniform dequantization, and we exclude reporting NLL with the variational dequantization [39] because it requires training an auxiliary network [11] only for the evaluation, after the model training.

Table 3: Performance comparison to linear/nonlinear diffusion models on CIFAR-10. We report both before/after correction of density estimation performances. We report the baseline performances of linear diffusions by training our PyTorch implementation based on Song et al. [1, 11] with identical hyperparameters and networks on both linear/nonlinear diffusions in order to quantify the effect of nonlinearity in a fair setting. Boldface numbers represent the best performance in a column.

SDE	Model	Nonlinear Data Diffusion	# Params	NLL (↓) after correction	NLL (↓) before correction	NELBO (↓) w/ residual (after)	NELBO (↓) w/o residual (before)	Gap (↓) (=NELBO-NLL) after	Gap (↓) (=NELBO-NLL) before	FID (↓) ODE	FID (↓) PC
VE	NCSN++ (FID)	✗	63M	4.86	3.66	4.89	4.45	0.03	0.79	-	2.38
	INDM (FID)	✓	76M	3.22	3.13	3.28	3.24	0.06	0.11	-	**2.29**
VP	DDPM++ (FID)	✗	62M	3.21	3.16	3.34	3.32	0.13	0.16	3.90	2.89
	INDM (FID)	✓	75M	3.17	3.11	3.23	3.18	0.06	0.07	**3.61**	2.90
	DDPM++ (NLL)	✗	62M	3.03	2.97	3.13	3.11	0.10	0.14	6.70	5.17
	INDM (NLL)	✓	75M	**2.98**	**2.95**	**2.98**	**2.97**	**0.00**	**0.02**	6.01	5.30

Table 5: Performance comparison on CIFAR-10.

SDE	Type	Model	# Params	NLL	FID
Linear		NCSN++ (FID) [1]	108M	4.85	2.20
		DDPM++ (FID) [1]	108M	3.19	2.64
		DDPM++ (NLL) [1]	108M	3.01	4.88
		VDM [29]	-	**2.65**	7.41
		CLD-SGM [30]	108M	3.31	2.25
Nonlinear	SBP	SB-FBSDE [17]	102M	2.98	3.18
	VAE -based	LSGM (FID) [13]	476M	3.45	**2.10**
		LSGM (NLL) [13]	269M	2.97	6.15
		LSGM (NLL) [13]	506M	2.87	6.89
		LSGM (balanced) [13]	109M	2.96	4.60
		LSGM (balanced) [13]	476M	2.98	2.17
	Flow -based	DiffFlow (FID) [14]	≈36M	3.04	14.14
		INDM (FID)	118M	3.09	2.28
		INDM (NLL)	121M	2.97	4.79
		INDM (NLL) + ST	75M	3.01	3.25

Table 6: Performance comparison on CelebA 64×64.

Model	NLL	NELBO	Gap	FID
UNCSN++ [31]	**1.93**	-	-	1.92
DDGM [32]	-	-	-	2.92
Efficient-VDVAE [33]	-	**1.83**	-	-
CR-NVAE [34]	-	1.86	-	-
DenseFlow-74-10 [4]	1.99	-	-	-
StyleFormer [35]	-	-	-	3.66
NCSN++ (VE, FID)	3.41	3.42	0.01	3.95
INDM (VE, FID)	2.31	2.33	0.02	2.54
DDPM++ (VP, FID)	2.14	2.21	0.07	2.32
INDM (VP, FID)	2.27	2.31	0.04	**1.75**
DDPM++ (VP, NLL)	2.00	2.09	0.09	3.95
INDM (VP, NLL)	2.05	2.05	**0.00**	3.06

A continuous diffusion model truncates the time horizon from $[0, T]$ to $[\epsilon, T]$ to avoid training instability [31]. This positive truncation could potentially misguide the model performance evaluation. Concretely, observe $\mathbb{E}_{\mathbf{x}_0}[-\log p_0^m(\mathbf{x}_0)] \leq \mathbb{E}_{\mathbf{x}_0,\mathbf{x}_\epsilon}\left[-\log p_\epsilon^m(\mathbf{x}_\epsilon) + \log \frac{p_{\epsilon 0}^m(\mathbf{x}_0|\mathbf{x}_\epsilon)}{p_{0\epsilon}(\mathbf{x}_\epsilon|\mathbf{x}_0)}\right]$ holds from the Jensen's inequality. Here, p_0^m and p_ϵ^m are the model probability distributions at $t = 0$ and $t = \epsilon$, respectively; and $p_{\epsilon 0}^m(\cdot|\mathbf{x}_\epsilon)$ is the reconstruction transition probability of $\mathbf{x}_0$ given $\mathbf{x}_\epsilon$ of a model at hand, where model varies by algorithms such as DDPM++, LSGM, and INDM. In NLL calculation, previous works [1, 11, 13] have simply approximated $\log p_0^m(\mathbf{x}_0)$ by $\log p_\epsilon^m(\mathbf{x}_0)$ with no theoretic guarantee for this approximation. Instead, we choose to compute $\log p_\epsilon^m(\mathbf{x}_\epsilon)$ and $\log \frac{p_{\epsilon 0}^m(\mathbf{x}_0|\mathbf{x}_\epsilon)}{p_{0\epsilon}(\mathbf{x}_\epsilon|\mathbf{x}_0)}$, separately, and add these values for NLL. This NLL computation differs to the previous way ($\log p_\epsilon^m(\mathbf{x}_0)$) about 0.03-0.06 in VPSDE, see Table 3. Also, for NELBO, we add the residual term of $\log \frac{p_{\epsilon 0}^m(\mathbf{x}_0|\mathbf{x}_\epsilon)}{p_{0\epsilon}(\mathbf{x}_\epsilon|\mathbf{x}_0)}$ in our computation. We report both previous/corrected ways in Table 3, and report corrected NLL/NELBO as default in the remaining tables, see Appendix D for theoretic and empirical justification of our NLL/NELBO corrections. We use either the Predictor-Corrector (PC) sampler [1] or a numerical ODE solver (RK45 [40]) of probability flow ODE [1].

We pretrain the score network as default to improve sample generation performance. The score network is pretrained by fixing the flow network to be an identity function, which corresponds to a linear diffusion model, such as NCSN++ or DDPM++. Table 4 compares INDM with and without the pretraining, where we denote DDPM++ as our pretraining performance. It shows that INDM training after the pretraining of 500k steps helps the model performance, which is the benefit of the flow training. Table 3 provides additional evidence of the effectiveness of the flow training in various linear diffusions and weighting functions. It compares the pretrained NCSN++/DDPM++ with INDM, of which the score network is initialized by these pretrained linear models, and the flow network is initialized by the identity mapping. The learnable nonlinear diffusion strategy is the major source of such a consistent performance improvement. In particular, there is a significant gain on NELBO by nonlinearizing the diffusion process throughout all experiments. Also, the *variational gap* between NELBO and NLL significantly reduces: e.g., the gap of 0.1 in DDPM++ (NLL) collapses to zero in the further training of INDM, as illustrated in Figure 5, see Appendix B.1 for further analysis.

Table 4: Effect of Pretraining.

Model	NLL	FID
DDPM++	3.03	6.70
INDM (w/ pre)	2.98	6.01
INDM (w/o pre)	2.98	8.49

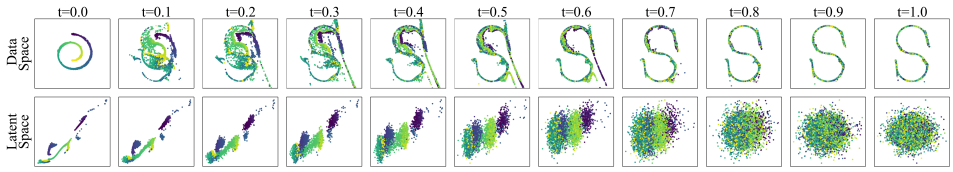


Figure 8: INDM enables to learn a diffusion bridge between two distinctive data distributions.

Table 5 compares INDM to other diffusion models with various SDEs on CIFAR-10. With the emphasis on sample generation, LSGM is the state-of-the-art model on CIFAR-10. INDM (FID) with 118M largely improves LSGM (FID) with 476M parameters with respect to NLL, at the slight expense of FID with less than a quarter number of parameters. Moreover, we use Soft Truncation [31] to compare with LSGM (balanced). Soft Truncation softens the smallest diffusion time as a random variable in the training stage to boost sample performance by improving the score accuracy particularly on large diffusion time. In the inference stage, Soft Truncation uses the fixed smallest diffusion time. INDM (NLL) + ST outperforms LSGM (balanced) with 109M parameters in terms of FID with comparable NLL. Also, Table 5 presents that INDM (NLL) with 121M parameters outperforms LSGM (NLL) with 269M parameters in FID with identical NLLs. Note that we report NLL with the uniform dequantization, which is known to perform worse than the variational dequantization about 0.15-0.2 in BPD scale [11]. Table 6 compares INDM to the current best models in various generative models on CelebA. INDM reports 1.75 FID, which outperforms the previous state-of-the-art model (UNCSN++). Analogous to CIFAR-10, INDM achieves the zero *variational gap* with $\lambda(t) = g^2(t)$.

7.1 Application: Dataset Interpolation

The linear VE/VP SDEs perturb data variable in a fixed manner, so such SDEs are designed to have the final distribution, $p_T(\mathbf{x}_T)$, as an easy-to-compute distribution, such as a Gaussian distribution. When the forward diffusion is nonlinearized, the diffusion process could be complex enough to transport a data distribution ($p_r^{(1)}$) to another data distribution ($p_r^{(2)}$). However, a diffusion model constrains only the starting variable by $\mathbf{x}_0^{\phi} \sim p_r^{(1)}$, and the ending variable of $\mathbf{x}_T^{\phi}$ is free to deviate from $p_r^{(2)}$. Imposing bi-constraints as De Bortoli et al. [16] of SBP with $\mathcal{P}(p_r, \pi)$ replaced by $\mathcal{P}(p_r^{(1)}, p_r^{(2)})$ is a natural approach for dataset interpolation, but we propose an alternative way with INDM.

On top of INDM NELBO in Theorem 1, we train the score and flow networks with a loss

$$\underbrace{D_{KL}(\boldsymbol{\mu}_{\phi} \| \boldsymbol{\nu}_{\phi,\theta})}_{\text{INDM NELBO}} + \underbrace{D_{KL}(p_r^{(2)} \| p_{\phi})}_{\text{Interpolation Loss}},$$

where p_{ϕ} is the probability distribution of $\mathbf{x}_T^{\phi}$. The additional interpolation loss forces the diffusion bridge of $\{\mathbf{x}_t^{\phi}\}_{t=0}^{T}$ to ahead towards the direction of $p_r^{(2)}$ by minimizing the distance between $\mathbf{x}_T^{\phi} \sim p_{\phi}$ and $p_r^{(2)}$. The destined variable of the bridge is constructed by $\mathbf{x}_T^{\phi} = \mathbf{h}_{\phi}^{-1}(\mathbf{z}_T) \approx \mathbf{h}_{\phi}^{-1}(\pi)$, so the flow network is what constructs the interpolated bridge between a couple of data variables, see Figures 8 and 9.By exchanging INDM NELBO with the LSGM loss and the interpolation loss with NELBO of VAE in LSGM, we could analogously train LSGM for the dataset interpolation task. Figure 9, however, illustrates that LSGM fails at interpolating two datasets. Also, INDM is modeling $p_r^{(1)}$ with $p_{\phi,\theta}$ and $p_r^{(2)}$ with p_{ϕ}, so we could compute NLL of each dataset *separately*: 1.10 BPD for MNIST and 1.56 BPD for EMNIST. In contrast, SBP cannot separately estimate densities on each dataset. We emphasize that no additional neural network is needed for this task with INDM.

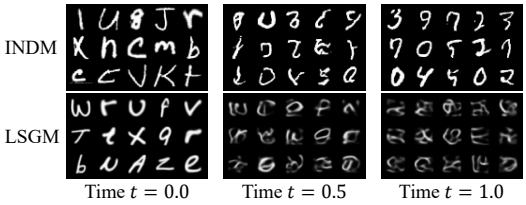


Figure 9: Image-to-image translation from EMNIST letters dataset to MNIST digits dataset without using conditional diffusion models [41, 42].

8 Conclusion

This paper expands the linear diffusion to trainable nonlinear diffusion by combining an invertible transformation and a diffusion model, where the nonlinear diffusion learns the forward diffusion out of variational family of inference measures. A limitation of INDM lies in the training/evaluation time. Potential risk from this work is the negative use of deep generative models, such as deepfake images.

References

[1] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

[2] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[4] Matej Grcić, Ivan Grubišić, and Siniša Šegvić. Densely connected normalizing flows. *Advances in Neural Information Processing Systems*, 34, 2021.

[5] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[6] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

[7] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Advances in neural information processing systems*, 33:12438–12448, 2020.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[9] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[10] Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[11] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34, 2021.

[12] Chin-Wei Huang, Jae Hyun Lim, and Aaron C Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34, 2021.

[13] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34, 2021.

[14] Qinsheng Zhang and Yongxin Chen. Diffusion normalizing flow. *Advances in Neural Information Processing Systems*, 34, 2021.

[15] Francisco Vargas, Pierre Thodoroff, Austen Lamacraft, and Neil Lawrence. Solving schrödinger bridges via maximum likelihood. *Entropy*, 23(9):1134, 2021.

[16] Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34, 2021.

[17] Tianrong Chen, Guan-Horng Liu, and Evangelos Theodorou. Likelihood training of schrödinger bridge using forward-backward SDEs theory. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nioAdKCEdXB.

[18] Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

[19] Eliya Nachmani, Robin San Roman, and Lior Wolf. Denoising diffusion gamma models. *arXiv preprint arXiv:2110.05948*, 2021.

[20] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

[21] Flavien Léger and Wuchen Li. Hopf–cole transformation via generalized schrödinger bridge problem. *Journal of Differential Equations*, 274:788–827, 2021.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[23] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[24] Valentin Khrulkov and Ivan Oseledets. Understanding ddpm latent codes through optimal transport. *arXiv preprint arXiv:2202.07477*, 2022.

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[26] Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

[27] Bharath K Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert RG Lanckriet. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11:1517–1561, 2010.

[28] Lawrence C Evans. Partial differential equations. *Graduate studies in mathematics*, 19(2), 1998.

[29] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

[30] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=CzceR82CYc.

[31] Dongjun Kim, Seungjae Shin, Kyungwoo Song, Wanmo Kang, and Il-Chul Moon. Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation, 2021. URL https://arxiv.org/abs/2106.05527.

[32] Eliya Nachmani, Robin San Roman, and Lior Wolf. Non gaussian denoising diffusion models. *arXiv preprint arXiv:2106.07582*, 2021.

[33] Louay Hazami, Rayhane Mama, and Ragavan Thurairatnam. Efficient-vdvae: Less is more. *arXiv preprint arXiv:2203.13751*, 2022.

[34] Samarth Sinha and Adji Bousso Dieng. Consistency regularization for variational auto-encoders. *Advances in Neural Information Processing Systems*, 34, 2021.

[35] Jeeseung Park and Younggeun Kim. Styleformer: Transformer based generative adversarial networks with style vector. *arXiv preprint arXiv:2106.07023*, 2021.

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[37] Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

[38] Xuezhe Ma, Xiang Kong, Shanghang Zhang, and Eduard H Hovy. Decoupling global and local representations via invertible generative flows. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=iWLByfvUhN.

[39] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019.

[40] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[41] Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=c7NBMfDXbW.

[42] Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models. *arXiv preprint arXiv:2104.05358*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have reflected the contribution of this work claimed in the abstract and introduction in the main sections.
   (b) Did you describe the limitations of your work? [Yes] See Section 7 and 8.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 8.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] I have read the ethics review guidelines and ensured that our paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will submit the code, data, and instructions needed to reproduce the main experimental results in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Figures 4, 6, 7-(b) report the error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix E.
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We will include a URL about the assets we used in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Our asset is a representative asset in the field of diffusion models.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]