SFBD Flow: A Continuous-Optimization Framework for Training Diffusion Models with Noisy Samples

Haoye Lu
University of Waterloo
Vector Institute

Darren Lo
University of Waterloo

Yaoliang Yu
University of Waterloo
Vector Institute

Abstract

Diffusion models achieve strong generative performance but often rely on large datasets that may include sensitive content. This challenge is compounded by the models' tendency to memorize training data, raising privacy concerns. SFBD (Lu et al., 2025) addresses this by training on corrupted data and using limited clean samples to capture local structure and improve convergence. However, its iterative denoising and fine-tuning loop requires manual coordination, making it burdensome to implement. We reinterpret SFBD as an alternating projection algorithm and introduce a continuous variant, SFBD flow, that removes the need for alternating steps. We further show its connection to consistency constraint-based methods, and demonstrate that its practical instantiation, Online SFBD, consistently outperforms strong baselines across benchmarks.

1 INTRODUCTION

Diffusion-based generative models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021a,b, 2023) have attracted growing interest and are now regarded as one of the most powerful frameworks for modelling high-dimensional distributions. They have enabled remarkable progress across various domains (Croitoru et al., 2023), including image (Ho et al., 2020; Song et al., 2021a,b; Rombach et al., 2022), audio (Kong et al., 2021; Yang et al., 2023), and video generation (Ho et al., 2022).

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

Diffusion models can be efficiently trained using the conditional score-matching loss, making them relatively easy to scale. This scalability enables the training of very large models on web-scale datasets – a crucial factor in achieving high performance. This approach has driven recent breakthroughs in image generation, exemplified by models such as Stable Diffusion (-XL) (Rombach et al., 2022; Podell et al., 2024) and DALL-E (Betker et al., 2023). However, this success comes with challenges: large-scale datasets often include copyrighted material, and diffusion models are more prone than earlier generative methods like GANs (Goodfellow et al., 2014, 2020) to memorizing training data, potentially reproducing entire samples (Carlini et al., 2023; Somepalli et al., 2023).

A recently proposed strategy to address memorization and copyright concerns involves training or fine-tuning diffusion models on corrupted data (Daras et al., 2023b; Somepalli et al., 2023; Daras and Dimakis, 2023; Daras et al., 2024). In this setting, the model never has direct access to the original data. Instead, each sample is transformed via a known, non-invertible corruption process, such as pixel-wise additive Gaussian noise in image datasets, ensuring that the original content cannot be reconstructed or memorized at the individual sample level. Remarkably, under mild conditions, such corruptions - although irreversible at the sample level – can induce a bijection between the original and corrupted data distributions (Bora et al., 2018). Specifically, the corrupted data distribution has a density equal to the convolution of the true data density with the corruption noise distribution (Meister, 2009; Lu et al., 2025). As a result, it is theoretically possible to recover the original data distribution by first estimating the corrupted (noisy) density from samples, and then performing density deconvolution to approximate the underlying true data density.

We refer to this task – recovering the true data distribution from noisy observations – as the *deconvolution problem*. Motivated by this formulation, several works (Daras et al., 2024, 2025; Lu et al., 2025) have shown

that diffusion models can effectively address the deconvolution problem either by applying iterative denoising and fine-tuning, as in SFBD (Lu et al., 2025), or by enforcing consistency constraints (CCs) during training (Daras et al., 2023a). Specifically, when paired with a small set of copyright-free clean data, both SFBD and CC-based methods have been shown to guide diffusion models toward generating high-quality images. However, SFBD relies on costly iterative denoising and fine-tuning, while CC-based methods require solving backward stochastic differential equations (SDEs) at each training step, making both approaches computationally expensive in different ways.

In this paper, we eliminate the iterative denoising and fine-tuning required by SFBD by introducing a continuous variant, *SFBD flow*. We reinterpret SFBD as alternating projections between two sets of stochastic processes, framing it as a stochastic process optimization problem. Inspired by Schrödinger bridge flow (Bortoli et al., 2024), this perspective yields a generalized family of diffusion-based deconvolution methods, γ-SFBD for $\gamma \in (0, 1]$, that guide the model toward the clean data distribution. Standard SFBD is recovered when $\gamma = 1$, while letting $\gamma \to 0$ turns the discrete sequence into a continuous evolution, giving rise to the SFBD flow. We further show that SFBD flow corresponds to steepest descent in function space, with γ-SFBD as its discrete approximation using step size γ. This interpretation motivates Online SFBD, a practical diffusion-based deconvolution method that avoids repeated fine-tuning (Sec 5). By adaptively adjusting γ, Online SFBD accommodates approximation errors from imperfect training, achieving faster and more stable convergence. We also reveal a close connection to CC-based methods, providing a unified perspective. Empirical results validate our analysis, with Online SFBD consistently outperforming strong baselines across benchmarks. The implementation is available at https://github.com/watml/SFBD-flow.

2 PRELIMINARIES

In this section, we review diffusion models, the deconvolution problem, and two typical methods for training diffusion models on data corrupted by Gaussian noise.

Diffusion Models. Diffusion models generate data by progressively adding Gaussian noise to input samples and then learning to reverse this process through a sequence of denoising steps. Formally, given an initial data distribution p_0 over $\mathbb{R}^d$, the forward process is defined by the SDE

$$\mathrm{d}\mathbf{x}_t = \mathrm{d}\mathbf{w}_t, \quad \mathbf{x}_0 \sim p_0, \quad t \in [0, T], \tag{1}$$

where T is a fixed positive constant. $\{\mathbf{w}_t\}_{t\in[0,T]}$ is the standard Brownian motion.[1]

Eq (1) induces a transition kernel $p_{t|s}(\mathbf{x}_t|\mathbf{x}_s)$ for $0 \le s \le t \le T$. For $s = 0$,

$$p_{t|0}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0, t\,\mathbf{I}), \qquad t \in [0, T]. \tag{2}$$

When T is large, the terminal state $\mathbf{x}_T$ closely approximates a sample from the isotropic Gaussian distribution $\mathcal{N}(\mathbf{0}, T\,\mathbf{I})$. Let $p_t(\mathbf{x}_t) = \int p_{t|0}(\mathbf{x}_t|\mathbf{x}_0)\, p_0(\mathbf{x}_0)\, \mathrm{d}\mathbf{x}_0$ denote the marginal distribution of $\mathbf{x}_t$, where $p_T \approx \mathcal{N}(\mathbf{0}, T\,\mathbf{I})$. Anderson (1982) showed that the time-reversed dynamics of the forward SDE can be expressed as the backward SDE:

$$\mathrm{d}\mathbf{x}_t = -\mathbf{s}_t(\mathbf{x}_t)\,\mathrm{d}t + \mathrm{d}\bar{\mathbf{w}}_t, \ \mathbf{x}_T \sim p_T, \tag{3}$$

where $\bar{\mathbf{w}}_t$ is standard Brownian motion in reverse time and $\mathbf{s}_t = \mathbf{s}_t^* := \nabla \log p_t$ is the score function. Crucially, this reverse SDE induces transition kernels that match the posterior of the forward process: $p_{s|t}(\mathbf{x}_s|\mathbf{x}_t) = \frac{p_{t|s}(\mathbf{x}_t|\mathbf{x}_s)p_s(\mathbf{x}_s)}{p_t(\mathbf{x}_t)}$ for $s \le t$ in $[0, T]$. It is known that $\mathbf{s}_t^*(\mathbf{x}_t) = \frac{1}{t}(\mathbb{E}_{p_{0|t}}[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t)$, where the conditional expectation $\mathbb{E}_{p_{0|t}}[\mathbf{x}_0 \mid \mathbf{x}_t]$ is typically approximated in practice by a neural network-denoiser $D_{\boldsymbol{\phi}}(\mathbf{x}_t)$ (Karras et al., 2022), trained by minimizing

$$\mathcal{L}_d(\boldsymbol{\phi}) = \mathbb{E}_{t\sim\mathcal{T}}\, \mathbb{E}_{p_0}\, \mathbb{E}_{p_{t|0}} \left[w(t) \| D_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{x}_0 \|^2 \right], \tag{4}$$

where $w(t)$ is a time-dependent weighting function and $\mathcal{T}$ denotes a sampling distribution over $[0, T]$. With a well-trained denoiser $D_{\boldsymbol{\phi}}$, $\mathbf{s}_t^*$ can be approximated by

$$\mathbf{s}_t^{\boldsymbol{\phi}}(\mathbf{x}_t) := \tfrac{1}{t}\big(D_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{x}_t\big). \tag{5}$$

Substituting this estimate into Eq (3), one can simulate the reverse-time SDE starting from $\tilde{\mathbf{x}}_T \sim \mathcal{N}(\mathbf{0}, T\,\mathbf{I})$, yielding a sample $\tilde{\mathbf{x}}_0$ that serves as an approximation sampled from p_0.

Deconvolution Problem. We follow the setup of Daras et al. (2024); Lu et al. (2025), where corrupted samples $\mathcal{Y} = \{\mathbf{y}^{(i)}\}_{i=1}^n$ are generated as $\mathbf{y}^{(i)} = \mathbf{x}^{(i)} + \boldsymbol{\epsilon}^{(i)}$, with $\mathbf{x}^{(i)} \sim p_{\text{data}}$ and $\boldsymbol{\epsilon}^{(i)} \sim h = \mathcal{N}(\mathbf{0}, \tau\mathbf{I})$ drawn independently, where $\tau \in (0, T)$ is known and fixed. The resulting samples $\mathbf{y}^{(i)}$ follow a distribution with density $p_\tau^* = p_{\text{data}} * h$, where $*$ denotes the convolution operator (Lu et al., 2025). In addition, we assume access to a small set of clean samples $\mathcal{D}_{\text{clean}} = \{\mathbf{x}^{(i)}\}_{i=1}^M$ with $\mathbf{x}^{(i)} \sim p_{\text{data}}$.

While deconvolution theory (Meister, 2009; Lu et al., 2025) and related empirical results in the context of GANs (Bora et al., 2018) have demonstrated the theoretical and practical feasibility of learning the true

[1] Despite its simplicity, nearly all existing diffusion models adopt a forward process equivalent to Eq (1), as shown in Sec A.1.

data distribution from noisy samples, a key challenge remains: how to effectively train a diffusion model on corrupted data to generate clean samples.

Consistency Constraint-based Method. Daras et al. (2024) first addressed this problem using CCs (Daras et al., 2023a). With noisy samples $\mathbf{x}_\tau \sim p^*_\tau$, they trained a network $\mathbf{s}^{\boldsymbol{\phi}}_t$ to approximate score $\mathbf{s}^*_t$ for $t > \tau$ via a modified loss called ambient score matching (ASM). Specifically, $\mathbf{s}^{\boldsymbol{\phi}}_t$ is implemented through Eq (5), where $D_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$ approximates $\mathbb{E}_{p_{0|t}}[\mathbf{x}_0 \mid \mathbf{x}_t]$. For $t \le \tau$, score matching is inapplicable, and instead $D_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$ is trained to satisfy the CCs:

$$\mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s] = \mathbb{E}_{p_{r|s}}\big[\mathbb{E}_{p_{0|r}}[\mathbf{x}_0|\mathbf{x}_r]\big], \text{ for } 0 \le r \le s \le T$$

by jointly minimizing the *consistency loss*:

$$\mathcal{L}_{\text{con}}(\boldsymbol{\phi}, r, s) = \mathbb{E}_{p_s}\big\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{r|s}}[D_{\boldsymbol{\phi}}(\mathbf{x}_r, r)]\big\|^2 \quad (6)$$

where r and s are sampled from predefined distributions. Sampling from $p_{r|s}$ is performed by solving Eq (3) backward from $\mathbf{x}_s$, using the network-estimated drift $\mathbf{s}^{\boldsymbol{\phi}}_t$ from Eq (5). To sample from p_s, one first draws $\mathbf{x}_\tau$ for $\tau > s$, then samples $\mathbf{x}_s \sim p_{s|\tau}$ analogously. If $D_{\boldsymbol{\phi}}$ minimizes the consistency loss for all r, s and satisfies $\mathbf{s}^{\boldsymbol{\phi}}_t = \mathbf{s}^*_t$ for $t > \tau$, then $\mathbf{s}^{\boldsymbol{\phi}}_t$ exactly recovers $\mathbf{s}^*_t$ for all $t \in [0, T]$, allowing $p_0 = p_{\text{data}}$ to be sampled via Eq (3) (Daras et al., 2024).

However, both Daras et al. and Lu et al. showed that using CCs alone is insufficient to recover the drift below τ due to poor sample complexity (Lu et al., 2025; Daras et al., 2025). Daras et al. addressed this by adding the denoising loss Eq (4) on $\mathcal{D}_{\text{clean}}$, achieving strong empirical results.

Stochastic Forward-backward Deconvolution. Instead of relying on CCs to recover the distribution for $t \le \tau$, Lu et al. (2025) proposed an iterative scheme, SFBD, that alternates between finetuning and denoising steps. Given a sample set $\mathcal{E}$, let $p_{\mathcal{E}}$ denote the empirical distribution induced by $\mathcal{E}$. Starting from a pretrained model $D_{\boldsymbol{\phi}_0}$ trained on $\mathcal{D}_{\text{clean}}$, the algorithm proceeds as follows for $k = 1, 2, \ldots, K$:

(Denoise) $\mathcal{E}_k \leftarrow \{\mathbf{y}^{(i)}_0$: solve Eq (3) from $t = \tau$ to 0 with $\mathbf{s}_t(\mathbf{x}_t) = \frac{D_{\boldsymbol{\phi}_k}(\mathbf{x}_t, t) - \mathbf{x}_t}{t}, \mathbf{x}_\tau = \mathbf{y}^{(i)}_\tau \in \mathcal{E}_{\text{noisy}}\}$.

(Finetune) Update $D_{\boldsymbol{\phi}_k}$ to obtain $D_{\boldsymbol{\phi}_{k+1}}$ by miminizing Eq (4) with $p_0 = p_{\mathcal{E}_k}$.

Lu et al. (2025) showed that as $K \to \infty$, $p_{\mathcal{E}_K}$ converges to the true distribution p_{data}. While SFBD outperforms DDIM (Song et al., 2021a) trained solely on clean data (e.g., on CelebA (Liu et al., 2015)), its iterative nature makes implementation challenging.

3 SFBD AS ALTERNATIVE PROJECTIONS

In this section, we show that SFBD can be interpreted as an alternating projection algorithm.

Notation. Let $\mathcal{M}$ be the set of path measures on $t \in [0, \tau]$ induced by the backward process Eq (3), with arbitrary drift $\mathbf{s} : [0, \tau] \times \mathbb{R}^d \to \mathbb{R}^d$ and fixed terminal distribution p^*_τ at $t = \tau$. We write $M(\mathbf{s}) \in \mathcal{M}$ for the path measure corresponding to drift $\mathbf{s}$. Likewise, let $\mathcal{D}$ be the set of path measures on $t \in [0, \tau]$ induced by the forward process Eq (1) with arbitrary initial distribution p_0, and denote by $D(q) \in \mathcal{D}$ the measure induced by $p_0 = q$. In particular, let $P^* = D(p_{\text{data}})$ and $\mathbf{s}^*_t = \nabla \log p_t$ be the score function of the forward diffusion process initialized at $p_0 = p_{\text{data}}$. Then $P^* = D(p_{\text{data}}) = M(\mathbf{s}^*)$, showing that P^* lies in $\mathcal{M} \cap \mathcal{D}$, and is in fact the unique element therein.

Alternative Projections. SFBD then can be formulated as an algorithm alternating between two projections: the Markov projection (M-Proj) and the diffusion projection (D-Proj):

$$\text{(M-Proj) } M^k = \operatorname*{proj}_{\mathcal{M}} P^k := \operatorname*{argmin}_{M \in \mathcal{M}} D_{\text{KL}}(P^k \,\|\, M) \quad (7)$$

$$\text{(D-Proj) } P^{k+1} = \operatorname*{proj}_{\mathcal{D}} M^k := \operatorname*{argmin}_{P \in \mathcal{D}} D_{\text{KL}}(M^k \,\|\, P) \quad (8)$$

for $k = 0, 1, 2, \ldots, K$, with initial path measure $P^0 = \mathcal{D}(p_{\mathcal{E}_{\text{clean}}})$. Since each $M \in \mathcal{M}$ is fully determined by a backward drift $\mathbf{s}$, we denote the drift of M^k by $\mathbf{s}^k$, i.e., $M^k = M(\mathbf{s}^k)$. Thus, the M-Proj can be equivalently written as $\operatorname{argmin}_{\mathbf{s}} D_{\text{KL}}(P^k \,\|\, M(\mathbf{s}))$.

The M-Proj corresponds to the finetuning step in SFBD. To see this, by Lem 1 in Sec A.8,

$$\begin{aligned} D_{\text{KL}}(P^k \,\|\, M^k) =& D_{\text{KL}}(p^k_\tau \,\|\, p^*_\tau) \\ &+ \mathbb{E}_{P^k}\big[\tfrac{1}{2}\textstyle\int_0^\tau \|\nabla \log p^k_t(\mathbf{x}_t) - \mathbf{s}^k_t(\mathbf{x}_t)\|^2 \,\mathrm{d}t\big] \end{aligned}$$

where p^k_t denotes the marginal density of P^k at time t. Since the first term is independent of M^k, minimizing the KL reduces to setting $\mathbf{s}^k_t(\mathbf{x}_t) = \nabla \log p^k_t(\mathbf{x}_t)$, i.e., performing score matching. This corresponds to the fine-tuning step that minimizes Eq (4) with $p_0 = p^k_0$ (Karras et al., 2022).

Likewise, D-Proj corresponds to the denoising step. By the disintegration theorem (Vargas et al., 2021),

$$D_{\text{KL}}(M^k \,\|\, P) = D_{\text{KL}}(m^k_0 \,\|\, p_0) + \mathbb{E}_{M^k}\Big[\log \tfrac{\mathrm{d}M^k(\cdot|\mathbf{x}_0)}{\mathrm{d}P(\cdot|\mathbf{x}_0)}\Big] \quad (9)$$

where m^k_0 is the marginal of M^k at $t = 0$. Since $P \in \mathcal{D}$ is determined by the forward SDE in Eq (1), its conditional path measure given $\mathbf{x}_0$ is fixed, making the second term constant. Therefore, minimizing the

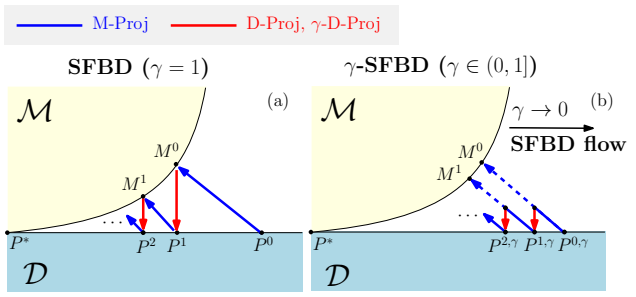


Figure 1: SFBD uses alternative projection to guide the stochastical process sequences P^k and M^k converge to the optimal P^*. When $\gamma \to 0$, the changes of P^k and M^k become smooth and we obtain γ-SFBD.

KL divergence reduces to matching the marginals, i.e., $p_0 = m_0^k$ and thus $P^{k+1} = D(m_0^k)$. In other words, D-Proj sets p_0 to the distribution of the denoised samples in the denoising step. Fig 1a illustrates how SFBD applies the alternative projection algorithm to find P^*.

Evolution of $\mathbf{s}^k$. In practice, the only component in SFBD requiring estimation is $\mathbf{s}^k = \nabla \log p_t^k(\mathbf{x}_t)$, approximated by a neural network $\hat{\mathbf{s}}^k$. As $k \to \infty$, $\mathbf{s}_t^k$ converges to the true score function $\mathbf{s}_t^*$ associated with the forward diffusion process Eq (1) initialized with $p_0 = p_{\text{data}}$ (Lu et al., 2025).

As the updates in Eqs (7) and (8) can be compactly written as $M^{k+1} = \text{argmin}_{M \in \mathcal{M}} D_{\text{KL}}(\text{proj}_{\mathcal{D}} M^k \,\|\, M)$ with $M^k = M(\mathbf{s}^k)$, we have

$$\mathbf{s}^{k+1} = \text{argmin}_{\mathbf{s}}\, \mathcal{L}^\dagger(\mathbf{s}, M^k), \tag{10}$$

where $\mathcal{L}^\dagger(\mathbf{s}, M^k) = D_{\text{KL}}(\text{proj}_{\mathcal{D}} M^k \,\|\, M(\mathbf{s}))$. As detailed in Sec A.2, this is equivalent to minimizing

$$\mathcal{L}(\mathbf{s}, M^k) = \int_0^\tau \mathcal{L}_t \, \mathrm{d}t \tag{11}$$

where $\mathcal{L}_t := \mathbb{E}_{D(m_0^k)} \frac{1}{2} \left\| \frac{\mathbf{x}_0 - \mathbf{x}_t}{t} - \mathbf{s}_t(\mathbf{x}_t) \right\|^2$ and m_0^k is the marginal of M^k at $t = 0$. Thus, SFBD can be interpreted as the iterative update:

$$\mathbf{s}^{k+1} = \text{argmin}_{\mathbf{s}}\, \mathcal{L}(\mathbf{s}, M(\mathbf{s}^k)), \tag{12}$$

with $\mathbf{s}^0 = \text{argmin}_{\mathbf{s}}\, D_{\text{KL}}(\mathcal{D}(p_{\mathcal{E}_{\text{clean}}}) \,\|\, M(\mathbf{s}))$.

In practice, estimating each $\mathbf{s}^k$ requires training a separate neural network, which is computationally expensive and hindered by manual intervention and ambiguous stopping criteria. While one could continue training from the previous step, the large shift in the target distribution m_0^k across iterations forces optimizers such as Adam (Kingma and Ba, 2015) to be reset, as stale momentum otherwise leads to divergence. This repeated reinitialization discards accumulated momentum and slows training. In Sec 4, we reformulate the update as a continuous steepest descent of $\mathcal{L}$, ensuring that P^k evolves smoothly and $u_{\boldsymbol{\theta}}$ can be optimized without resetting, allowing the optimizer to adapt seamlessly.

4 SFBD FLOW

In this section, we extend SFBD to a family of iterative deconvolution procedures, γ-SFBD with $\gamma \in (0, 1]$. It recovers SFBD at $\gamma = 1$, while as $\gamma \to 0$ the sequence $M^k, \mathbf{s}^k$ converges to continuous flows $M^\kappa, \mathbf{s}^\kappa$, $\kappa \geq 0$.

We show that γ-SFBD admits two derivations: a generalized D-Proj, showing how smaller γ yields smoother trajectories, and a discretized functional gradient descent on $\mathcal{L}(\mathbf{s}, M_0(\mathbf{s}))$, establishing convergence to a continuous flow.

Derive γ-SFBD Through a Generalized D-Proj. For $\gamma \in (0, 1]$, consider a generalized D-Proj:

$$P^{k+1,\gamma} = \underset{P \in \mathcal{D}}{\text{argmin}}\, (1-\gamma) D_{\text{KL}}(P^{k,\gamma} \,\|\, P) + \gamma D_{\text{KL}}(M^k \,\|\, P) \tag{13}$$

We refer to SFBD with D-Proj replaced by γ-D-Proj as γ-SFBD. When $\gamma = 1$, it recovers the original SFBD. (Although M^k does depend on γ, we keep the original notation for simplicity.) To see how γ-D-Proj smooths the update, note that the denoised samples at iteration k follow a distribution with density (see Sec A.3):

$$p_0^{k+1,\gamma} = (1-\gamma)\, p_0^{k,\gamma} + \gamma\, m_0^k \tag{14}$$

where $p_0^{0,\gamma} = p_{\mathcal{E}_{\text{clean}}}$ and $P^{k+1,\gamma} = D(p_0^{k+1,\gamma})$. Basically, the parameter γ controls how much of the denoised set is updated using the latest model. When $\gamma = 1$, all samples are replaced, recovering standard SFBD. As $\gamma \to 0$, the updates become infinitesimal, leaving M^{k+1} – obtained by projecting $P^{k+1,\gamma}$ onto $\mathcal{M}$ – and its corresponding $\mathbf{s}^{k+1}$ nearly unchanged (see Fig 1b). Despite the smoothing effect, γ-SFBD guarantees convergence for all $\gamma \in (0, 1]$. In particular, let $\Phi_p(\mathbf{u}) = \mathbb{E}_p[\exp(i\, \mathbf{u}^\top \mathbf{x})]$ denote the characteristic function of p for $\mathbf{u} \in \mathbb{R}^d$. Under mild assumptions,

Proposition 1. *For $k \geq 0$, $D_{\text{KL}}(p_{\text{data}} \,\|\, p_0^{k+1,\gamma}) - D_{\text{KL}}(p_{\text{data}} \,\|\, p_0^{k,\gamma}) \leq -\gamma D_{\text{KL}}(p_\tau^* \,\|\, p_\tau^{k,\gamma})$. In addition,*

$$\min_{k=1,\ldots,K} \left| \Phi_{p_{\text{data}}}(\mathbf{u}) - \Phi_{p_0^{k,\gamma}}(\mathbf{u}) \right| \leq \exp\left(\frac{\tau}{2}\|\mathbf{u}\|^2\right) \left(\frac{2M}{\gamma K}\right)^{1/2}$$

for $K \geq 1$, $\mathbf{u} \in \mathbb{R}^d$, and $M = D_{\text{KL}}(p_{\text{data}} \,\|\, p_{\mathcal{E}_{\text{clean}}})$.

(All proofs are deferred to the appendix.) Prop 1 shows that for any $\gamma \in (0, 1]$, $p_0^{k,\gamma}$ converges to p_{data} as k grows, with convergence of characteristic functions ensuring convergence of distributions. Notably, although

we report convergence speed with a fixed γ for clarity, $\gamma \in (0, 1]$ may vary across iterations without compromising convergence to the optimal solution. In Prop 3, we show how to use an adaptive γ to accommodate the imperfect estimation of neural networks.

γ-SFBD as Functional Gradient Descent. In Sec 3, we showed that SFBD updates the backward drift $\mathbf{s}^k$ by solving $\operatorname{argmin}_{\mathbf{s}} \mathcal{L}(\mathbf{s}, M_0(\mathbf{s}^k))$. We now consider a relaxed version, where $\mathbf{s}$ is updated via a single gradient descent step in function space with step size $\gamma \in (0, 1]$. This update rule exactly recovers γ-SFBD algorithm.

Recall that for a functional $\ell : \mathcal{F} \to \mathbb{R}$ defined over a function space $\mathcal{F}$, its functional derivative at $\mathbf{u} \in \mathcal{F}$ with respect to a reference measure P is a function $\nabla_P \ell(\mathbf{u}) \in \mathcal{F}$ (when it exists) satisfying (Courant and Hilbert, 1989):

$$\int \langle \nabla_P \ell(\mathbf{u})(\mathbf{x}), \boldsymbol{\nu}(\mathbf{x}) \rangle \mathrm{d}\mu(\mathbf{x}) = \lim_{\lambda \to 0} \tfrac{1}{\lambda} \big(\ell(\mathbf{u} + \lambda \boldsymbol{\nu}) - \ell(\mathbf{u}) \big)$$

for all $\boldsymbol{\nu} \in \mathcal{F}$. Building on this, we have:

Proposition 2. *Let $\gamma \in (0, 1]$ and $k \in \mathbb{N}$. Let $P^{k,\gamma}$ and M^k denote the stochastic process sequences generated by γ-SFBD via the update rules in Eq (7) and Eq (13). Then the update of $M(\mathbf{s}^k) = M^k$ satisfies*

$$\mathbf{s}_t^{k+1}(\mathbf{x}) = \mathbf{s}_t^k(\mathbf{x}) - \gamma \nabla_{P_t^{k+1,\gamma}} \mathcal{L}_t(\mathbf{s}_t^k, m_0(\mathbf{s}^k))(\mathbf{x}) \quad (15)$$

for all $\mathbf{x} \in \mathbb{R}^d$ and $t \in [0, \tau]$.

As a result, γ-SFBD basically performs a discretized functional gradient descent on $\mathcal{L}(\mathbf{s}, M_0(\mathbf{s}))$ with step size γ, following the steepest descent under the reference distribution $P^{k,\gamma}$, updated via (14). Remarkably, Prop 1 shows that value γ does not affect convergence of $p_0^{k,\gamma}$ to p_{data}. Thus, *for any $\gamma \in (0, 1]$, $\mathbf{s}_t^k$ converges to the true score function $\mathbf{s}_t^*$ learned by a diffusion model trained on clean data, with $\gamma = 1$ recovering the original SFBD result (Lu et al., 2025).*

SFBD Flow. The functional gradient descent perspective shows that as $\gamma \to 0$, the discrete sequence $\{\mathbf{s}^k\}_{k \in \mathbb{N}}$ and the associated distributions $p_0^{k,\gamma}$ converge to continuous flows $\{\mathbf{s}^\kappa\}_{\kappa \geq 0}$ and p_0^κ, governed by the gradient flow of $\mathcal{L}(\mathbf{s}, M_0(\mathbf{s}))$. We refer to this continuous formulation as *SFBD flow*. To characterize the evolution of p_0^κ, fix $\kappa > 0$ and let $\{\gamma_i\} \to 0$ with $k_i = \kappa / \gamma_i \in \mathbb{N}$. Then $p_0^{k_i, \gamma_i} \to p_0^\kappa$ and $m_0^{k_i} \to m_0(\mathbf{s}^\kappa)$ via Euler approximation. Taking the limit,

$$\frac{\mathrm{d}}{\mathrm{d}\kappa} p_0^\kappa = \lim_{i \to \infty} \tfrac{1}{\gamma_i} (p_0^{k_i+1, \gamma_i} - p_0^{k_i, \gamma_i}) \overset{(14)}{=} m_0(\mathbf{s}^\kappa) - p_0^\kappa,$$

where $p_0^0 = p_{\mathcal{E}_{\mathrm{clean}}}$. Thus, p_0^κ evolves according to an ODE driven by the mismatch between the model's denoised output $m_0(\mathbf{s}^\kappa)$ and the current estimate p_0^κ. Under this flow formulation, the convergence of γ-SFBD reduces to:

Corollary 1. *For $\kappa > 0$, we have $\frac{\mathrm{d}}{\mathrm{d}\kappa} D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^\kappa) \leq -D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^\kappa)$. Additionally,*

$$\inf_{\kappa \in [0, \mathcal{K}]} \left| \Phi_{p_{data}}(\mathbf{u}) - \Phi_{p_0^\kappa}(\mathbf{u}) \right| \leq \exp\left(\tfrac{\tau}{2} \|\mathbf{u}\|^2 \right) \left(\tfrac{2M}{\mathcal{K}} \right)^{1/2}$$

for $\mathcal{K} > 0$, $\mathbf{u} \in \mathbb{R}^d$ and $M = D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_{\mathcal{E}_{\mathrm{clean}}})$.

Adaptive Step Size γ. In practice, $\mathbf{s}_t^k$ is approximated by a neural network $\hat{\mathbf{s}}_t^k$. The imperfect approximation may hinder convergence to the true data distribution as empirically shown in Sec 6. In Prop 3, we show the issue can be mitigated by adaptively updating γ using an upper bound δ_k on the approximation error, defined by

$$\delta_k > \max\Big\{ 0,\ D_{\mathrm{KL}}(P^* \,\|\, \hat{M}_k) - D_{\mathrm{KL}}(P^* \,\|\, M_k) \Big\}, \quad (16)$$

where $\hat{M}_k \in \mathcal{M}$ denotes the path measure induced by the approximate drift $\hat{\mathbf{s}}_t^k$.

Proposition 3. *Suppose $\mathbf{s}_t^k$ is approximated by a neural network $\hat{\mathbf{s}}_t^k$, and let δ_k satisfy Eq (16) with $\sum_{k=1}^\infty \delta_k < \infty$. For $\rho, \Gamma \in (0, 1)$ and $\gamma_0, v_0 > 0$, choose the step size at iteration k as*

$$\gamma_k = \min\Big(\tfrac{\gamma_0}{\sqrt{v_k}}, \Gamma \Big), \quad v_k = \rho v_{k-1} + (1 - \rho)\delta_k^2. \quad (17)$$

Then $p_0^k \to p_{data}$ as $k \to \infty$.

Prop 3 basically suggests using larger step sizes γ_k when the exponential moving average (EMA) of the error bound decreases. Intuitively, this means if the network $\hat{\mathbf{s}}^k$ provides a good approximation of the target, a larger fraction of denoised samples can be substituted with those generated by $\hat{\mathbf{s}}^k$ without hindering convergence. We provide a practical way to construct δ_k in Sec 5.

The parameter Γ acts as an upper bound on the fraction of samples in $\mathcal{E}$ that may be refreshed at each iteration. Early in training, δ_k is typically large, which yields a large v_k and therefore a very small effective update ratio $\frac{\gamma_0}{\sqrt{v_k}}$. In this stage, Γ is essentially inactive. As training progresses and the model better aligns with the distribution represented by $\mathcal{E}$, $\frac{\gamma_0}{\sqrt{v_k}}$ may grow large. At this point, Γ becomes the active constraint, limiting the maximum update ratio and preventing overly aggressive updates.

5 ONLINE SFBD OPTIMIZATION

As discussed in Sec 4, when γ is small, Prop 2 shows that the sequence $\mathbf{s}^k$ closely tracks its continuous limit $\mathbf{s}^\kappa$. Since $\mathbf{s}^k$ is parameterized by neural networks, this continuity motivates replacing iterative fine-tuning in SFBD with a single network $\mathbf{s}^{\boldsymbol{\phi}}$ that continuously approximates the evolving $\mathbf{s}^k$. The optimization of $\mathbf{s}^{\boldsymbol{\phi}}$

Algorithm 1 Online SFBD

Input: clean data: $\mathcal{E}_{\text{clean}} = \{\mathbf{x}^{(i)}\}_{i=1}^{M}$, noisy data: $\mathcal{E}_{\text{noisy}} = \{\mathbf{y}_\tau^{(i)}\}_{i=1}^{N}$, num of gradient steps: m, base γ_0
// Initialize Denoiser
$\boldsymbol{\phi} \leftarrow$ Pretrain $D_{\boldsymbol{\phi}}$ using Eq (4) with $p_0 = p_{\mathcal{E}_{\text{clean}}}$
$\mathcal{E} \leftarrow \{\mathbf{y}_0^{(i)}$: solve Eq (3) from $t = \tau$ to 0 with $\mathbf{s}_t(\mathbf{x}_t) = \hat{\mathbf{s}}_t(\mathbf{x}_t) := \frac{D_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{x}_t}{t}$, $\mathbf{x}_\tau = \mathbf{y}_\tau^{(i)} \in \mathcal{E}_{\text{noisy}}\}$
$v \leftarrow$ compute δ for $\hat{\mathbf{s}}_t(\mathbf{x}_t)$ **if** adapt. γ, **else** -1
repeat
  Update $\boldsymbol{\phi}$ with m gradient steps on Eq (4) with $p_0 = p_{\mathcal{E}}$. // M-Proj
  $\delta \leftarrow$ compute δ for $\hat{\mathbf{s}}_t(\mathbf{x}_t)$ **if** adapt. γ, **else** -1
  $(\gamma, v) \leftarrow \mathsf{getStepSize}(v, \delta)$ **if** adapt. γ, **else** $(\gamma_0, -1)$.
  $\mathcal{E} \leftarrow \{$Replace ratio γ of denoised samples in $\mathcal{E}$ with the new ones by solving Eq (3) from $t = \tau$ to 0 with $\mathbf{s}_t(\mathbf{x}_t) = \hat{\mathbf{s}}_t(\mathbf{x}_t) := \frac{D_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{x}_t}{t}$, $\mathbf{x}_\tau = \mathbf{y}_\tau^{(i)}$ randomly picked from $\mathcal{E}_{\text{noisy}}\}$ // γ-D-Proj
until *reach the maximum number of iterations*
Output: Final denoiser $D_{\boldsymbol{\phi}}$

follows M-Proj Eq (7), implemented by minimizing the loss of matching score Eq (4) with $p_0 = p_0^{k,\gamma}$. Unlike standard SFBD, γ-SFBD refreshes only a fraction γ of denoised samples in each γ-D-Proj step, inducing small changes to $p_0^{k,\gamma}$ - so a few gradient steps suffice for $\mathbf{s}^{\boldsymbol{\phi}}$ to track the new minimizer. Building on this insight, we propose *Online SFBD* in Alg 1, which eliminates the need to fine-tune a sequence of networks.

Adaptive step size γ. In Prop 3, we show that the imperfect approximation of network $\hat{\mathbf{s}}_t$ can be accommodated by adjusting the step size, which requires us to build valid δ_k satisfying Eq (16). In our implementation, we construct $\mathcal{E}_{\boldsymbol{\phi}}$ by sampling 256 images $\mathbf{y}$ from $\mathcal{E}_{\text{noisy}}$ and solving Eq (3) from $t = \tau$ to 0 with drift $\hat{\mathbf{s}}_t(\mathbf{x}_t)$ and initial condition $\mathbf{x}_\tau = \mathbf{y}$. We then heuristically set $\delta^2 \propto \max\big(0, \mathsf{KID}(\mathcal{E}_{\text{clean}}, \mathcal{E}_{\boldsymbol{\phi}}) - \mathsf{KID}(\mathcal{E}_{\text{clean}}, \mathcal{E})\big)$, with the scaling absorbed into γ_0. Here KID denotes Kernel Inception Distance (Bińkowski et al., 2018), an unbiased alternative to Fréchet Inception Distance (Heusel et al., 2017) that does not require a minimum sample size. Given δ^2, Alg 2 defines the update of γ. If $\delta^2 \geq \eta v$, we set $\gamma = 0$ and skip the partial update of $\mathcal{E}$, allowing $\hat{\mathbf{s}}$ more steps to converge before the target shifts. This design ensures that the effective δ decays faster than a geometric sequence with some ratio $\alpha < 1$, guaranteeing $\sum_k \delta_k < \infty$ as required by Prop 3 (see Sec A.5 for details and justification of the chosen δ^2 formulation).

Combining Denoised and Clean Samples. Since the copyright-free clean samples are drawn from the true data distribution, we follow the original SFBD framework (Lu et al., 2025) and set $p_0 = p_{\mathcal{E} \cup \mathcal{E}_{\text{clean}}}$ in the M-Proj step. This choice helps accelerate optimization by aligning the target distribution for updating $\boldsymbol{\phi}$ more closely with the true data distribution. As detailed in

Algorithm 2 getStepSize – Adaptive step size update

Input: EMA v, error bound δ, threshold $\eta \in (0, 1)$, decay $\rho \in (0, 1)$, base $\gamma_0 > 0$, cap $\Gamma \in (0, 1]$.
if $\delta^2 < \eta\, v$ **then**
  $v \leftarrow \rho\, v + (1 - \rho)\, \delta^2$ // Update EMA
  $\gamma \leftarrow \min\big(\frac{\gamma_0}{\sqrt{v}}, \Gamma\big)$ // Compute step size
else
  $\gamma \leftarrow 0$ // Skip $\mathcal{E}$ update for large error
Output: (γ, v)

Sec A.6, this corresponds to a variant of γ-Diff Proj, and we provide additional justification there for the observed performance gains.

Denoising and Sampling. While Alg 1 uses a naive backward sampler by solving Eq (3), the algorithm allows any backward SDE and solver that yield the same marginals. We adopt the 2nd-order Heun method from EDM (Karras et al., 2022) for better error control and efficiency. To improve sample quality (Nichol and Dhariwal, 2021; Karras et al., 2022), we maintain an EMA version of the model for denoising and use it to update $\mathcal{E}$; all reported results in Sec 6 are based on this EMA model. In practice, γ is typically small (e.g., $\gamma < 0.02$), so the mild asynchrony between γ-D-Proj and M-Proj has negligible effect, as suggested by preliminary exploration during framework implementation. This motivates a practical strategy we call *asynchronous denoising*: denoising runs independently on a separate, low-performance GPU, updating $\mathcal{E}$ in the background, while the main training loop minimizes Eq (4) on high-performance hardware using the latest $p_0 = p_{\mathcal{E}}$. We adopt this strategy throughout our study in Sec 6.

Relationship to Consistency Constraint-based Methods. Consistency constraint-based (CC-based) methods such as TweedieDiff (Daras et al., 2024) and TweedieDiff+ (Daras et al., 2025), which enforce consistency only between time zero and positive time steps, can be seen as special cases of Online SFBD with a single gradient step ($m = 1$). (See Sec A.7 for details and an extension to arbitrary time pairs.) These methods approximate $p^{k,\gamma}$ using m_0^k rather than the EMA over $\{m_0^j\}_{j \leq k}$ as defined in Eq (14), which is not exact unless $\gamma = 1$. Since $\mathbf{s}$ is updated just once per iteration, m_0^j for j close to k tends to be similar, making m_0^k a reasonable proxy of $p^{k,\gamma}$ when γ is not too small.

In Sec 6, we show that avoiding this approximation enables Online SFBD to consistently outperform CC-based methods. Remarkably, it also achieves significantly lower computational cost. This is because Online SFBD reuses cached denoised samples throughout training, whereas CC-based methods generate them on demand – requiring more samples per step for stability

and making asynchronous denoising impractical. Moreover, CC-based methods typically enforce consistency between arbitrary time pairs, requiring multiple neural network forward passes per update. In contrast, Online SFBD matches the compute cost of a standard diffusion model, apart from denoised sample updates – which can be performed asynchronously on separate GPUs.

6 EMPIRICAL STUDY

In this section, we demonstrate the effectiveness of Online SFBD. We first study its behaviour under different configurations to identify practical settings, with ablation results supporting the theory and offering guidance for use. Building on these insights, we benchmark Online SFBD and show that it consistently outperforms models trained directly on noisy data. Compared to standard SFBD, the online variant achieves better results while eliminating costly iterative finetuning and denoising.

Datasets and Evaluation Metrics. We conduct experiments on CIFAR-10 (Krizhevsky and Hinton, 2009) and CelebA (Liu et al., 2022), using image resolutions of 32×32 and 64×64, respectively. CIFAR-10 contains 50,000 training and 10,000 test images across 10 classes. CelebA includes 162,770 training, 19,867 validation, and 19,962 test images; we use the preprocessed version from the official DDIM repository (Song et al., 2021a). Corrupted images are generated by adding independent Gaussian noise with standard deviation σ to each pixel after rescaling to $[-1, 1]$. Notably, only one noisy counterpart is generated per clean image.

Image quality is evaluated using Fréchet Inception Distance (FID), computed between the reference dataset and 50,000 model-generated samples. Generated image samples are shown in Sec B.

Models and Other Configurations. We implement Online SFBD using the EDM architecture and hyperparameters (Karras et al., 2022) in an unconditional setting, with non-leaky augmentation to mitigate overfitting. Backward sampling is performed with the 2^{nd}-order Heun method (Karras et al., 2022); details are in Sec C. As discussed in Sec 5, Online SFBD minimizes the denoising score-matching loss Eq (4) with $p_0 = p_0^{k,\gamma}$ updated via sample denoising. Effective training requires sufficiently minimizing Eq (4), before each p_0 update. For fixed γ, this balance can be controlled either by adjusting the update ratio γ or the number of gradient steps m. Since batch denoising is more efficient with larger updates, we fix γ (updating 640 samples per iteration) and vary m, using $m = 20$ by default unless otherwise noted. For adaptive γ, we fix $m = 20$ and compute γ using Alg 2 with $\eta = 0.99$ and $\rho = 0.9$. The number of updated samples per γ-D-Proj step is capped at 2048, corresponding to $\Gamma = 0.04$ for CIFAR-10 and 0.01 for CelebA.

Role and Practical Setting of Γ. As discussed in Sec 4, Γ becomes active only in the later stages of training, where it caps the maximum update ratio and prevents overly aggressive updates. Empirically, this cap slows optimization but improves stability.

As we will show in Fig 2c, with a fixed γ, a small m leads to a large effective update rate, yielding a rapid initial FID drop, whereas a smaller update rate slows progress yet stabilizes convergence. Considering that the adaptive-γ scheme automatically adjusts the effective update ratio to prevent training collapse, in practice, we simply set Γ to the value corresponding to the largest single sampling batch size that fits within GPU memory, which yields the fastest convergence. Following this principle, low-resolution datasets such as CIFAR-10 allow for a larger Γ and therefore potentially faster convergence, whereas high-resolution datasets like CelebA require smaller Γ values due to reduced batch capacity.

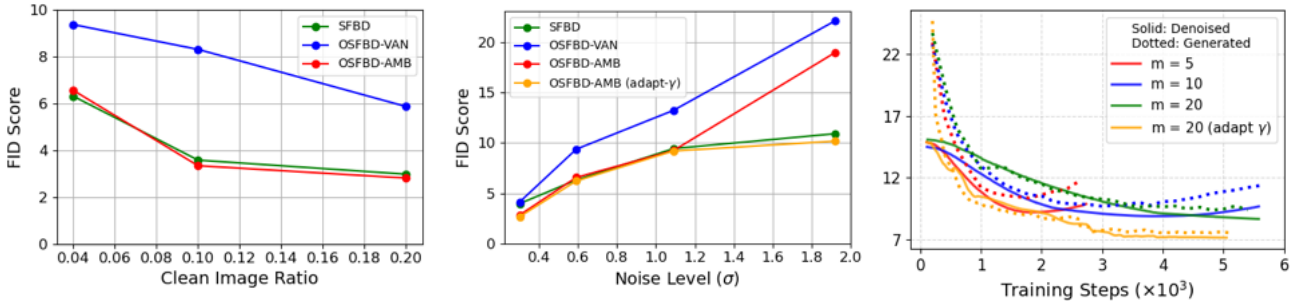
6.1 Ablation Study

We examine the behaviour of Online SFBD under various settings on CIFAR-10. Informed configurations lead to state-of-the-art results on all benchmarks, as shown in Sec 6.2.

Methods of Pretraining. While the score function for $t < \tau$ must be estimated from limited clean, copyright-free data, ambient score matching (ASM) can guide score estimation for $t > \tau$ using noisy samples (Daras et al., 2025). In Fig 2, we compare models pretrained only on clean data (**OSFBD-VAN**ILLA) with those jointly pretrained using ASM and noisy samples (**OSFBD-AMB**IENT). As shown in (a) and (b), OSFBD-AMB consistently outperforms OSFBD-VAN across all clean image ratios and noise levels. This improvement is expected, as OSFBD-AMB better leverages the noisy data to refine score estimation for $t > \tau$ and boost overall performance. We therefore use OSFBD-AMB for the benchmarks in Sec 6.2.

Number of Gradient Steps and Adaptive γ. Fig 2c shows the FID trajectories of generated and denoised samples during training for different gradient step counts m for fixed and adaptive γ. The models are pretrained using OSFBD-VAN.

The FID of denoised samples reflects the distance between $p^{k,\gamma}$ and p_{data}. Since a model that fully learns $p^{k,\gamma}$ would generate samples with FIDs matching the denoised ones, the FID gap indicates how well Eq (4)



(a) Clean Image Ratio (fixed γ) (b) Noise Level (fixed γ) (c) Num of Grad. Steps per Iteration

Figure 2: FID scores of Online SFBD (OSFBD) on CIFAR-10 under different settings. Unless specified, the clean ratio is 0.04, noise level $\sigma = 0.59$, and gradient steps $m = 20$. (c) reports results with OSFBD-VAN pretraining.

Table 1: Comparison. For $\sigma > 0$, models are trained on images corrupted with Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.

Method	CIFAR-10 (32×32)			CelebA (64×64)		
	σ	Pretrain (50 imgs)	FID	σ	Pretrain (50 imgs)	FID
DDPM (Ho et al., 2020)	0.0	No	4.04	0.0	No	**3.26**
DDIM (Song et al., 2021a)	0.0	No	4.16	0.0	No	6.53
EDM (Karras et al., 2022)	0.0	No	**1.97**	–	–	–
EMDiff (Bai et al., 2024)	0.2	Yes	86.47	–	–	–
TweedieDiff (Daras et al., 2024)	0.2	Yes	65.21	0.2	Yes	58.52
TweedieDiff+ (Daras et al., 2025)	0.2	Yes	8.05	0.2	Yes	6.81
SFBD (Lu et al., 2025)	0.2	Yes	13.53	0.2	Yes	6.49
OSFBD-AMB (fixed γ)	0.2	Yes	3.22	0.2	Yes	3.23
OSFBD-AMB (adaptive γ)	0.2	Yes	**3.12**	0.2	Yes	**3.19**

Table 2: Additional results for competitive models under various settings. (All models are pretrained.)

Method	CIFAR-10 (32×32)			CelebA (64×64)		
	σ	clean samples	FID	σ	clean samples	FID
TweedieDiff+ (Daras et al., 2025)	0.2	10%	2.81	1.38	50	35.65
SFBD (Lu et al., 2025)	0.2	10%	2.58	1.38	50	23.63
OSFBD-AMB (fixed γ)	0.2	10%	2.73	1.38	50	27.09
OSFBD-AMB (adaptive γ)	0.2	10%	**2.49**	1.38	50	**20.21**
TweedieDiff+ (Daras et al., 2025)	0.59	4%	6.75	1.38	1,500	6.81
SFBD (Lu et al., 2025)	0.59	4%	6.31	1.38	1,500	5.91
OSFBD-AMB (fixed γ)	0.59	4%	6.56	1.38	1,500	5.72
OSFBD-AMB (adaptive γ)	0.59	4%	**6.21**	1.38	1,500	**5.40**

has been minimized. A large gap suggests incomplete minimization at the current step. As shown in Fig 2c, smaller m values lead to more frequent updates of $p^{k,\gamma}$, causing a faster initial FID drop for denoised samples (e.g., $m = 5$). However, this rapid updating prevents the model from fully learning $p^{k,\gamma}$ before it shifts, as indicated by the widening FID gap after 1.5k steps. Consequently, the denoising process degrades, and FIDs for both denoised and generated samples begin to rise. With larger m, the model has more time to minimize the loss before $p^{k,\gamma}$ changes, delaying such degradation and achieving lower FIDs overall. Similar trends are observed with OSFBD-AMB, though to a milder degree. Importantly, although a bigger m can improve the training performance, this does not imply that an arbitrarily large m should be chosen, as it will significantly slow down the training process. In practice, we find $m = 20$ strikes a good balance.

Adaptive γ adjusts the update speed according to training progress, enabling faster FID reduction without the divergence seen at small m. The orange curve in Fig 2c appears less smooth than with fixed γ due to these adaptive adjustments, but this reflects the method's responsiveness and results in stable, accelerated convergence. The close overlap of the orange solid and dotted lines further indicates that adaptive γ provides enough iterations for the model to converge to each evolving target distribution.

6.2 Performance Comparison

We compare OSFBD-AMB with representative models for training on noisy images, as summarized in Table 1. EMDiff (Bai et al., 2024) uses a diffusion-based EM algorithm for inverse problems. TweedieDiff (Daras et al., 2024) applies the original consistency loss from Eq (6) and is pretrained on clean data. TweedieDiff+ (Daras et al., 2025) adopts the same pretraining as OSFBD-AMB, followed by joint training with a simplified consistency objective. SFBD (Lu et al., 2025) is the original algorithm requring iteratively finetuning.

Benchmark. Following the setup of Bai et al. (2024); Lu et al. (2025), we use 50 clean samples along with data corrupted Gaussian noise ($\sigma = 0.2$), with the same clean set across all experiments. For reference, we also report results for models trained on fully clean data ($\sigma = 0$). As shown in Table 1, OSFBD-AMB with fixed and adaptive γ consistently outperforms all baselines, producing significantly higher-quality images. Notably, it even surpasses DDPM and DDIM trained exclusively on clean samples on both datasets.

To further evaluate OSFBD-AMB, we test it under additional dataset configurations (Table 2) against the two strongest baselines, TweedieDiff+ and SFBD. With fixed γ, OSFBD-AMB consistently outperforms Tweed-

Table 3: Training time comparison between SFBD and OSFBD-AMB (adaptive γ) on CIFAR-10 and CelebA.

Method	CIFAR-10 ($\sigma = 0.59$, 4% clean)			CelebA ($\sigma = 1.38$, 1500 clean)		
	Pretrain	Optimization	Total	Pretrain	Optimization	Total
SFBD	24 h	18 h $\times 4$	96 h	36 h	24 h $\times 5$	156 h
OSFBD-AMB	24 h	36 h	60 h	36 h	72 h	108 h

ieDiff+ and matches SFBD in most cases, except on a challenging CelebA setting with scarce clean data and high noise (σ). The adaptive γ variant delivers consistently superior performance.

OSFBD-AMB vs SFBD. Results in Tables 1 and 2 show that SFBD can outperform fixed-γ OSFBD-AMB in cases with very limited clean data and high noise. To probe this further, we compare SFBD and OSFBD-AMB on CIFAR-10 across the same clean-data ratios and noise levels used in the original SFBD setup (Lu et al., 2025) (see Fig 2a,b). Specifically, when a moderate amount of clean data is available, the two methods perform comparably. Under low noise and scarce clean data, OSFBD-AMB outperforms SFBD, likely due to ambient-based pretraining and smoother updates of $p^{k,\gamma}$, which mitigate the overfitting issues noted in SFBD on small datasets (Lu et al., 2025). At high noise levels, however, denoising requires more backward SDE steps, compounding errors from imperfect training. In this regime, accurate score estimation becomes critical and demands many more gradient steps m for fixed-γ OSFBD-AMB, making SFBD more stable and effective. By contrast, adaptive-γ OSFBD-AMB tracks estimation quality and dynamically adjusts step size, alleviating imperfect training and yielding the superior performance seen in Tables 1 and 2.

To demonstrate the superior training efficiency of the proposed flow-based method, Table 3 presents a comparison of training times for SFBD and OSFBD-AMB (with adaptive γ) on CIFAR-10 and CelebA. As shown, SFBD requires approximately 50% more training time than OSFBD-AMB. Notably, this difference does not yet account for the additional human intervention and iterative tuning required across SFBD runs.

OSFBD-AMB vs TweedieDiff+. We observe that OSFBD-AMB with adpative and fixed γ consistently outperforms TweedieDiff+ across all settings in Tables 1 and 2, consistent with our discussion on their relationship in Sec 5. Both methods share the same pretraining procedure and differ only in how they learn the score function for $t < \tau$. By updating the denoised sample set in an EMA-like manner, OSFBD-AMB presents a significantly more accurate target distribution $p^{k,\gamma}$, leading to improved performance. Notably, this improvement also reduces computational cost – though at the expense of additional memory to cache denoised samples.

7 DISCUSSION

This paper extends the original SFBD algorithm to a family of variants, γ-SFBD. When $\gamma = 1$, it recovers SFBD; as $\gamma \to 0$, it yields SFBD flow and its practical counterpart – Online SFBD – which eliminates the need for alternating between denoising and fine-tuning. We also highlight its close connection to CC-based methods, another class of leading diffusion-based deconvolution techniques. Empirical results corroborate our analysis, showing that Online SFBD consistently outperforms strong baselines across some benchmarks.

Acknowledgement

We gratefully acknowledge funding support from NSERC, the Canada CIFAR AI Chairs program and the Ontario Early Researcher program. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

References

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

Weimin Bai, Yifei Wang, Wenzheng Chen, and He Sun. An expectation-maximization algorithm for training clean diffusion models from corrupted observations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=jURBh4V9N4.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. OpenAI, 2023. URL https://cdn.openai.com/papers/dall-e-3.pdf.

Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1801.01401.

Ashish Bora, Eric Price, and Alexandros G. Dimakis. AmbientGAN: Generative models from lossy measurements. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hy7fDog0b.

Valentin De Bortoli, Iryna Korshunova, Andriy Mnih, and Arnaud Doucet. Schrodinger bridge flow for unpaired data translation. In *The Thirty-eighth Annual*

Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=1F32iCJFfa.

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. In *32nd USENIX Security Symposium*, pages 5253–5270, 2023. URL https://www.usenix.org/system/files/usenixsecurity23-carlini.pdf.

R. Courant and D. Hilbert. *Methods of Mathematical Physics*. WILEY-VCH Verlag GmbH & Co. KGaA, 1989. ISBN 9783527414475. doi: 10.1002/9783527617210.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023. URL https://doi.org/10.1109/TPAMI.2023.3261988.

Giannis Daras and Alex Dimakis. Solving inverse problems with ambient diffusion. In *NeurIPS 2023 Workshop on Deep Learning and Inverse Problems*, 2023. URL https://openreview.net/forum?id=mGwg10bgHk.

Giannis Daras, Yuval Dagan, Alex Dimakis, and Constantinos Daskalakis. Consistent diffusion models: Mitigating sampling drift by learning to be consistent. In *Advances in Neural Information Processing Systems*, pages 42038–42063, 2023a. URL https://openreview.net/forum?id=GfZGdJHj27.

Giannis Daras, Kulin Shah, Yuval Dagan, Aravind Gollakota, Alex Dimakis, and Adam Klivans. Ambient diffusion: Learning clean distributions from corrupted data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=wBJBLy9kBY.

Giannis Daras, Alex Dimakis, and Constantinos Costis Daskalakis. Consistent diffusion meets tweedie: Training exact ambient diffusion models with noisy data. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=PlVjIGaFdH.

Giannis Daras, Yeshwanth Cherapanamjeri, and Constantinos Costis Daskalakis. How much is a noisy image worth? data scaling laws for ambient diffusion. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=qZwtPEw2qN.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. URL https://doi.org/10.1145/3422622.

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. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=f3zNgKga_ep.

Thomas Kailath. The Structure of Radon-Nikodym Derivatives with Respect to Wiener and Related Measures. *The Annals of Mathematical Statistics*, 42(3):1054–1067, 1971. URL https://doi.org/10.1214/aoms/1177693332.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=k7FuTOWMOc7.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=a-xFK8Ymz5J.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto,

2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*, April 2020.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=XVjTT1nw5z.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. URL http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

Haoye Lu, Qifan Wu, and Yaoliang Yu. Stochastic forward-backward deconvolution: Training diffusion models with finite noisy datasets, 2025. arXiv:2502.05446.

Alexander Meister. *Deconvolution Problems in Nonparametric Statistics*. Springer, 2009. URL https://doi.org/10.1007/978-3-540-87557-4.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-NEXDKk8gZ.

Bernt Oksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer, 6th edition, 2003. URL https://doi.org/10.1007/978-3-642-14394-6.

Michele Pavon and Anton Wakolbinger. On free energy, stochastic control, and Schrödinger processes. In *Modeling, Estimation and Control of Systems with Uncertainty: Proceedings of a Conference held in Sopron, Hungary, September 1990*, pages 334–348. Birkhäuser Boston, 1991. URL https://doi.org/10.1007/978-1-4612-0443-5_22.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=di52zR8xgf.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. URL https://doi.org/10.1109/CVPR52688.2022.01042.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, Cambridge, 2019.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265, 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6048–6058, 2023. URL https://doi.org/10.1109/CVPR52729.2023.00586.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

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*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 32211–32252, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Francisco Vargas, Pierre Thodoroff, Austen Lamacraft, and Neil Lawrence. Solving schrodinger bridges via maximum likelihood. *Entropy*, 23(9), 2021. URL https://www.mdpi.com/1099-4300/23/9/1134.

Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1720–1733, 2023. URL https://doi.org/10.1109/TASLP.2023.3268730.

Checklist

1. For all models and algorithms presented, check if you include:

(a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] All assumptions for the given theorem are clearly stated if needed, with more detailed derivations provided in the appendix. We provide clear descriptions of the algorithms in Alg 1 and Alg 2.

(b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No] An analysis of algorithmic properties and complexity (time, space, sample size) is not applicable here, as this work does not focus on complexity. Nonetheless, we provide a general discussion of the memory requirements of our proposed algorithms relative to existing methods.

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No] We will provide anonymized source code if requested but not otherwise.

2. For any theoretical claim, check if you include:

(a) Statements of the full set of assumptions of all theoretical results. [Yes] We provide the full set of assumptions for each theoretical result.

(b) Complete proofs of all theoretical results. [Yes] We provide a complete proof of theoretical results in the appendix.

(c) Clear explanations of any assumptions. [Yes] We clearly explain all assumptions.

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] The code is not currently provided but will be released upon acceptance. While the supplementary materials provides sufficient details, additional usage instructions will be provided with code release.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] Refer to Sec C.2

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [No] Training diffusion models from scratch is computationally expensive, and our access to high-performance hardware is limited in our school. Consequently, each configuration is run once, which aligns with common practice in diffusion model research. Nonetheless, the results are consistent across datasets and configurations, supporting the reliability of our conclusions.

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] Refer to Sec C.1

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes] We reference the licenses of existing datasets and cite the original papers corresponding to all code used.

(b) The license information of the assets, if applicable. [Yes]

(c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable] No new assets were released.

(d) Information about consent from data providers/curators. [Not Applicable] No consent from data providers was required.

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable] No sensible content was used.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Justification: The project does not use crowdsourcing nor did it conduct research with human subjects.

A THEORETICAL RESULTS

A.1 The Equivalence of the Forward Process Among the Existing Diffusion Models

With some simple extension, our work can handle a more general forward diffusion process of the form

$$\mathrm{d}\mathbf{x}_t = f(t)\mathbf{x}_t \ \mathrm{d}t + g(t) \ \mathrm{d}\mathbf{w}_t. \tag{18}$$

It can be shown that its transition kernel is a Gaussian distribution such that

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(m_t\mathbf{x}_0, \sigma_t^2 I) \tag{19}$$

See, for example, (Karras et al., 2022, Appendix B.1) or (Särkkä and Solin, 2019, Chapter 6). To the best of our knowledge, almost all popular diffusion-based methods adopt a forward process that can be written of this general form. Based on this fact, starting from $\mathbf{x}_0$, $\mathbf{x}_t$ can be seen as a sample of form

$$\mathbf{x}_t = m_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon} \tag{20}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$. As a result, we obtain

$$\mathbf{x}'_t := \frac{\mathbf{x}_t}{m_t} = \mathbf{x}_0 + \frac{\sigma_t}{m_t}\boldsymbol{\epsilon} \tag{21}$$

which shows that $\mathbf{x}'_t$ is equivalent to a noisy observation of $\mathbf{x}_0$ corrupted by Gaussian noise with noise level $\frac{\sigma_t}{m_t}$. This implies that the more general formulation considered here naturally reduces to the setting studied in the main paper. Consequently, our proposed method can also be implemented using the forward process in Eq (18), while the backward denoising step Eq (3) becomes

$$\mathrm{d}\mathbf{x}_t = f(t)\mathbf{x}_t \ \mathrm{d}t - g(t)^2 \nabla \log p_t(\mathbf{x}_t) + g(t) \ \mathrm{d}\mathbf{w}_t. \tag{22}$$

A.2 Minimizing KL Divergence is Equivalent to Conditional Drift Matching

In Sec 3, we claimed that minimizing $\mathcal{L}^\dagger$ defined in Eq (10) is equivalent to minimizing

$$\mathcal{L}(\mathbf{s}, M^k) = \int_0^\tau \mathcal{L}_t \, \mathrm{d}t = \int_0^\tau \mathbb{E}_{D(m_0^k)} \, \tfrac{1}{2} \left\| \frac{\mathbf{x}_0 - \mathbf{x}_t}{t} - \mathbf{s}_t(\mathbf{x}_t) \right\|^2 \mathrm{d}t. \tag{23}$$

To see this, note that according to Eq (9), D-Proj sets $P^{k+1} = \mathrm{proj}_{\mathcal{D}} \, M^k = D(m_0^k)$. As a result,

$$\mathcal{L}^\dagger(\mathbf{s}, M^k) = D_{\mathrm{KL}}(\mathrm{proj}_{\mathcal{D}} \, M^k \,\|\, M(\mathbf{s})) = D_{\mathrm{KL}}(D(m_0^k) \,\|\, M(\mathbf{s})).$$

By Lem 1, the KL divergence

$$D_{\mathrm{KL}}(D(m_0^k) \,\|\, M(\mathbf{s})) = \underbrace{D_{\mathrm{KL}}(m_0^k * \mathcal{N}(\mathbf{0}, \tau\mathbf{I}) \,\|\, p_\tau^*)}_{\text{const.}} + \mathop{\mathbb{E}}_{D(m_0^k)} \int_0^\tau \frac{1}{2} \left\| \mathbf{b}(\mathbf{x}_t, t) - \mathbf{s}_t(\mathbf{x}_t) \right\|^2 \mathrm{d}t,$$

where $\mathbf{b}^k(\mathbf{x}_t, t)$ is the drift of the backward SDE starting from τ with the initial distribution $m_0^k * \mathcal{N}(\mathbf{0}, \tau\mathbf{I})$. Anderson (1982) showed that $\mathbf{b}^k(\mathbf{x}_t, t) = \nabla \log m_t^k(\mathbf{x}_t)$, where $m_t^k(\mathbf{x}_t)$ denotes the density of the marginal distribution of M^k. It can be shown that (e.g., see (Song et al., 2023, Lemma 1)):

$$\nabla \log m_t^k(\mathbf{x}_t) = \mathbb{E}_{m_{0|t}^k}[\nabla_{\mathbf{x}_t} \log m_t^k(\mathbf{x}_t|\mathbf{x}_0)|\mathbf{x}_t] = \mathbb{E}_{m_{0|t}^k} \left[\frac{1}{t}(\mathbf{x}_0 - \mathbf{x}_t) \middle| \mathbf{x}_t \right]. \tag{24}$$

As a result,

$$\mathop{\mathbb{E}}_{D(m_0^k)} \int_0^\tau \frac{1}{2} \left\| \mathbf{b}(\mathbf{x}_t, t) - \mathbf{s}_t(\mathbf{x}_t) \right\|^2 \mathrm{d}t = \mathop{\mathbb{E}}_{D(m_0^k)} \int_0^\tau \frac{1}{2} \left\| \mathbb{E}_{m_{0|t}^k} \left[\frac{1}{t}(\mathbf{x}_0 - \mathbf{x}_t) \middle| \mathbf{x}_t \right] - \mathbf{s}_t(\mathbf{x}_t) \right\|^2 \mathrm{d}t.$$

Therefore,

$$\begin{aligned}\operatorname{argmin}_{\mathbf{s}}\tilde{\mathcal{L}}(\mathbf{s},M^k) &= \operatorname{argmin}_{\mathbf{s}}\ \mathop{\mathbb{E}}_{D(m_0^k)}\int_0^\tau \frac{1}{2}\left\|\mathbf{b}(\mathbf{x}_t,t)-\mathbf{s}_t(\mathbf{x}_t)\right\|^2 \mathrm{d}t \\ &= \operatorname{argmin}_{\mathbf{s}}\mathop{\mathbb{E}}_{D(m_0^k)}\int_0^\tau \left\|\mathbb{E}_{m_{0|t}^k}\left[\frac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\|^2 \mathrm{d}t.\end{aligned}$$

In addition, for any $t\in[0,\tau]$,

$$\begin{aligned}&\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbf{s}_t(\mathbf{x}_t)\right\|^2 \\ =&\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]+\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\|^2 \\ =&\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]\right\|^2+\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\|^2 \\ &\qquad+\mathop{\mathbb{E}}_{D(m_0^k)}\left\langle\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right],\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\rangle.\end{aligned}$$

For the last term,

$$\begin{aligned}&\mathop{\mathbb{E}}_{D(m_0^k)}\left\langle\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right],\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\rangle \\ =&\mathop{\mathbb{E}}_{m_t^k}\left\langle\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right],\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\rangle \\ =&\mathop{\mathbb{E}}_{m_t^k}\left\langle\mathbf{0},\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\rangle=0.\end{aligned}$$

As a result,

$$\begin{aligned}&\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbf{s}_t(\mathbf{x}_t)\right\|^2 \\ =&\underbrace{\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]\right\|^2}_{\text{Independent of }\mathbf{s}\Rightarrow\text{Const.}}+\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\mathbb{E}_{m_{0|t}^k}\left[\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\|^2.\end{aligned}$$

Thus,

$$\begin{aligned}\operatorname{argmin}_{\mathbf{s}}\mathcal{L}^\dagger(\mathbf{s},M^k) &= \operatorname{argmin}_{\mathbf{s}}\mathop{\mathbb{E}}_{D(m_0^k)}\int_0^\tau\left\|\mathbb{E}_{m_{0|t}^k}\left[\frac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)\middle|\mathbf{x}_t\right]-\mathbf{s}_t(\mathbf{x}_t)\right\|^2\mathrm{d}t \\ &= \operatorname{argmin}_{\mathbf{s}}\ \int_0^\tau\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbf{s}_t(\mathbf{x}_t)\right\|^2\mathrm{d}t+\text{Const.} \\ &= \operatorname{argmin}_{\mathbf{s}}\ \int_0^\tau\mathop{\mathbb{E}}_{D(m_0^k)}\left\|\tfrac{1}{t}(\mathbf{x}_0-\mathbf{x}_t)-\mathbf{s}_t(\mathbf{x}_t)\right\|^2\mathrm{d}t.\end{aligned}$$

A.3 Optimal Solution to Eq (13)

Note that, by the disintegration theorem (e.g., see Vargas et al. 2021, Appx B),

$$\begin{aligned}
&\operatorname*{argmin}_{P\in\mathcal{D}} (1-\gamma)D_{\mathrm{KL}}(P^{k,\gamma}\,\|\,P) + \gamma D_{\mathrm{KL}}(M^k\,\|\,P)\\
=\;& \operatorname*{argmin}_{P\in\mathcal{D}} (1-\gamma)\left[D_{\mathrm{KL}}(p_0^{k,\gamma}\,\|\,p_0) + \underbrace{\mathop{\mathbb{E}}_{P^{k,\gamma}}\Big[\log\frac{\mathrm{d}P^{k,\gamma}(\cdot|\mathbf{x}_0)}{\mathrm{d}P(\cdot|\mathbf{x}_0)}\Big]}_{\text{Const.}}\right]\\
&\qquad\qquad + \gamma\left[D_{\mathrm{KL}}(m_0^k\,\|\,p_0) + \underbrace{\mathop{\mathbb{E}}_{M^k}\Big[\log\frac{\mathrm{d}M^k(\cdot|\mathbf{x}_0)}{\mathrm{d}P(\cdot|\mathbf{x}_0)}\Big]}_{\text{Const.}}\right]\\
=\;& \operatorname*{argmin}_{P\in\mathcal{D}} (1-\gamma)D_{\mathrm{KL}}(p_0^{k,\gamma}\,\|\,p_0) + \gamma D_{\mathrm{KL}}(m_0^k\,\|\,p_0)\\
=\;& \operatorname*{argmin}_{P\in\mathcal{D}} -\int_{\mathbb{R}^d}\Big[(1-\gamma)\,p_0^{k,\gamma}(\mathbf{x}_0) + \gamma\,m_0^k(\mathbf{x}_0)\Big]\log p_0(\mathbf{x}_0)\;\mathrm{d}\mathbf{x}_0 + \text{Const.}\\
=\;& \operatorname*{argmin}_{P\in\mathcal{D}} D_{\mathrm{KL}}((1-\gamma)p_0^{k,\gamma} + \gamma m_0^k\,\|\,p_0).
\end{aligned}$$

As a result,

$$p_0^{k+1,\gamma} = (1-\gamma)\,p_0^{k,\gamma} + \gamma\,m_0^k. \tag{25}$$

A.4 Results Related to SFBD Flow

Proposition 1. *For $k\geq 0$, $D_{\mathrm{KL}}(p_{\mathrm{data}}\,\|\,p_0^{k+1,\gamma}) - D_{\mathrm{KL}}(p_{\mathrm{data}}\,\|\,p_0^{k,\gamma}) \leq -\gamma D_{\mathrm{KL}}(p_\tau^*\,\|\,p_\tau^{k,\gamma})$. In addition,*

$$\min_{k=1,\dots,K}\Big|\Phi_{p_{\mathrm{data}}}(\mathbf{u}) - \Phi_{p_0^{k,\gamma}}(\mathbf{u})\Big| \leq \exp\Big(\tfrac{\tau}{2}\|\mathbf{u}\|^2\Big)\Big(\tfrac{2M}{\gamma K}\Big)^{1/2}$$

for $K\geq 1$, $\mathbf{u}\in\mathbb{R}^d$, and $M = D_{\mathrm{KL}}(p_{\mathrm{data}}\,\|\,p_{\mathcal{E}_{\mathrm{clean}}})$.

Proof. Let P^* denote the path measure induced by the forward process Eq (1) with $p_0 = p_{\mathrm{data}}$. In addition, let $\mathcal{F}(q) = D_{\mathrm{KL}}(p_{\mathrm{data}}\,\|\,q)$. For brevity, we drop the γ in $P^{k,\gamma}$ and its marginal distributions $p_0^{k,\gamma}$ and $p_\tau^{k,\gamma}$.

Note that,

$$D_{\mathrm{KL}}(P^*\,\|\,M^k) = \mathcal{F}(m_0^k) + \underbrace{\mathbb{E}_{P^*}\left[\frac{1}{2}\int_0^\tau \|\mathbf{b}^k(\mathbf{x}_t,t)\|^2\,\mathrm{d}t\right]}_{:=\mathcal{B}_k}, \tag{26}$$

where $\mathbf{b}^k(\mathbf{x}_t,t)$ is the drift of the forward process inducing M^k with $\mathbf{x}_0\sim m_0^k$.

In addition, through the convexity of the KL divergence,

$$\mathcal{F}\big(p_0^{k+1}\big) = \mathcal{F}\big((1-\gamma)p_0^k + \gamma m_0^k\big) \leq (1-\gamma)\mathcal{F}(p_0^k) + \gamma\mathcal{F}(m_0^k),$$

which implies,

$$\mathcal{F}(m_0^k) \geq \mathcal{F}(p_0^k) + \tfrac{1}{\gamma}\big(\mathcal{F}(p_0^{k+1}) - \mathcal{F}(p_0^k)\big). \tag{27}$$

As a result,

$$\begin{aligned}\mathcal{F}(p_0^k) = D_{\mathrm{KL}}(P^* \,\|\, P^k) &= D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + \mathbb{E}_{p^*}\left[\int_0^\tau \frac{1}{2}\left\|\nabla \log p_t(\mathbf{x}_t) - \mathbf{s}_t^k(\mathbf{x}_t)\right\|^2\right] \\ &= D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + D_{\mathrm{KL}}(P^* \,\|\, M^k) \overset{(26)}{=} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + \mathcal{F}(m_0^k) + \mathcal{B}_k \\ &\overset{(27)}{\geq} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + \mathcal{B}_k + \tfrac{1}{\gamma}\Big(\mathcal{F}(p_0^{k+1}) - \mathcal{F}(p_0^k)\Big) + \mathcal{F}(p_0^k) \\ &\geq D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + \tfrac{1}{\gamma}\Big(\mathcal{F}(p_0^{k+1}) - \mathcal{F}(p_0^k)\Big) + \mathcal{F}(p_0^k). \end{aligned} \tag{28}$$

Rearrangement yields

$$D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^{k+1,\gamma}) - D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^{k,\gamma}) \leq -\gamma D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^{k,\gamma}), \tag{29}$$

the monotonicity of $p_0^{k,\gamma}$ in k in the proposition. Equivalently,

$$\mathcal{F}(p_0^{k+1,\gamma}) - \mathcal{F}(p_0^{k,\gamma}) \leq -\gamma D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^{k,\gamma}). \tag{30}$$

Telescoping it yields:

$$\mathcal{F}(p_0^{0,\gamma}) = \sum_{k=0}^{K} \mathcal{F}(p_0^{k,\gamma}) - \mathcal{F}(p_0^{k+1,\gamma}) \geq \gamma \sum_{k=1}^{K} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^{k,\gamma}). \tag{31}$$

Thus,

$$\min_{k\in\{1,2,\ldots,K\}} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^{k,\gamma}) \leq \frac{\mathcal{F}(p_0^{0,\gamma})}{\gamma K} = \frac{\mathcal{F}(p_{\mathcal{E}_{\mathrm{clean}}})}{\gamma K}. \tag{32}$$

Applying Prop 4, we get

$$\min_{k\in\{1,2,\ldots,K\}} \left|\Phi_{p_{\mathrm{data}}}(\mathbf{u}) - \Phi_{p_0^{k,\gamma}}(\mathbf{u})\right| \leq \exp\left(\frac{\tau}{2}\|\mathbf{u}\|^2\right)\left(\frac{2D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_{\mathcal{E}_{\mathrm{clean}}})}{\gamma K}\right)^{1/2}. \tag{33}$$

□

Proposition 2. *Let $\gamma \in (0,1]$ and $k \in \mathbb{N}$. Let $P^{k,\gamma}$ and M^k denote the stochastic process sequences generated by γ-SFBD via the update rules in Eq (7) and Eq (13). Then the update of $M(\mathbf{s}^k) = M^k$ satisfies*

$$\mathbf{s}_t^{k+1}(\mathbf{x}) = \mathbf{s}_t^k(\mathbf{x}) - \gamma \nabla_{P_t^{k+1,\gamma}} \mathcal{L}_t(\mathbf{s}_t^k, m_0(\mathbf{s}^k))(\mathbf{x}) \tag{15}$$

for all $\mathbf{x} \in \mathbb{R}^d$ and $t \in [0,\tau]$.

Proof. For $t \in [0,\tau]$, let $\boldsymbol{\phi}$ be a function of the same function space as $\mathbf{s}_t^k$ and p_0 the density of a distribution defined on $\mathbb{R}^d$. Then for $\epsilon \in (0,1]$, we have

$$\begin{aligned}\mathcal{L}_t(\mathbf{s}_t + \epsilon\boldsymbol{\phi}, p_0) =& \mathop{\mathbb{E}}_{D(p_0)}\left[\frac{1}{2}\left\|\frac{\mathbf{x}_0 - \mathbf{x}_t}{t} - (\mathbf{s}_t + \epsilon\boldsymbol{\phi})(\mathbf{x}_t)\right\|^2\right] \\ =& \mathcal{L}_t(\mathbf{x}_t, p_0) + \epsilon \mathop{\mathbb{E}}_{D(p_0)}\left[\left\langle \mathbf{s}_t(\mathbf{x}_t) - \frac{\mathbf{x}_0 - \mathbf{x}_t}{t}, \boldsymbol{\phi}(\mathbf{x}_t)\right\rangle\right] + o(\epsilon) \\ =& \mathcal{L}_t(\mathbf{x}_t, p_0) + \epsilon\left\langle \boldsymbol{\phi}(\mathbf{x}_t), \left(\mathbf{s}_t(\mathbf{x}_t) - \frac{\mathbf{x}_0 - \mathbf{x}_t}{t}\right) \mathrm{d}P_{0t}(\mathbf{x}_0, \mathbf{x}_t)\right\rangle \\ =& \mathcal{L}_t(\mathbf{x}_t, p_0) + \epsilon\left\langle \boldsymbol{\phi}(\mathbf{x}_t), \left(\mathbf{s}_t(\mathbf{x}_t) - \frac{\mathbf{x}_0 - \mathbf{x}_t}{t}\right) \mathrm{d}P_{0t}(\mathbf{x}_0, \mathbf{x}_t)\right\rangle \\ =& \mathcal{L}_t(\mathbf{x}_t, p_0) + \epsilon\left\langle \boldsymbol{\phi}(\mathbf{x}_t), \left(\mathbf{s}_t(\mathbf{x}_t) - \frac{\mathbb{E}_{P_{0|t}}[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t}{t}\right) dP_t(\mathbf{x}_t)\right\rangle. \end{aligned}$$

As a result,

$$\nabla_\mu \mathcal{L}(\mathbf{s}_t, p_0)(\mathbf{x}_t) = \Big(\mathbf{s}_t(\mathbf{x}_t) - \frac{\mathbb{E}_{P_{0|t}}[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t}{t}\Big) \frac{\mathrm{d}P_t}{\mathrm{d}\mu}(\mathbf{x}_t). \tag{34}$$

We note that when $k = 0$, $\mathbf{s}_t^0(\mathbf{x}_t)$ is pretrained on $P_{\mathcal{E}_{\text{clean}}}$. As a result, by e.g., (Song et al., 2023, Lemma 1),

$$\mathbf{s}_t^0(\mathbf{x}_t) = \frac{\mathbb{E}_{(P_{\mathcal{E}_{\text{clean}}})_{0|t}}(\mathbf{x}_0|\mathbf{x}_t) - \mathbf{x}_t}{t} = \frac{\mathbb{E}_{P^{0,\gamma}_{0|t}}(\mathbf{x}_0|\mathbf{x}_t) - \mathbf{x}_t}{t} \tag{35}$$

for any $t \in [0, \tau]$ and $\mathbf{x}_t \in \mathbb{R}^d$, which is the negative backward drift of M^0 in γ-SFBD.

Then assume that for $k \in \mathbb{N}$, for any $t \in [0, \tau]$ and $\mathbf{x}_t \in \mathbb{R}^d$, we have

$$\mathbf{s}_t^k(\mathbf{x}_t) = \frac{\mathbb{E}_{P^{k,\gamma}_{0|t}}(\mathbf{x}_0|\mathbf{x}_t) - \mathbf{x}_t}{t}, \tag{36}$$

correpsonding to the negative backward drift of M^k in γ-SFBD.

Then for $k+1$, Eq (15) gives

$$\begin{aligned}
\mathbf{s}_t^{k+1}(\mathbf{x}_t) &= \mathbf{s}_t^k(\mathbf{x}_t) - \gamma \nabla_{P_t^{k+1,\gamma}} \mathcal{L}_t(\mathbf{s}_t^k, m_0(\mathbf{s}^k))(\mathbf{x}_t) \\
&\overset{(34)}{=} \mathbf{s}_t^k(\mathbf{x}_t) - \gamma\Big(\mathbf{s}_t^k(\mathbf{x}_t) - \frac{\mathbb{E}_{D(m_0^k)_{0|t}}[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t}{t}\Big) \frac{\mathrm{d}D(m_0^k)_t}{\mathrm{d}P_t^{k+1,\gamma}}(\mathbf{x}_t) \\
&= \big(1 - \delta(\mathbf{x}_t)\big)\, \mathbf{s}_t^k(\mathbf{x}_t) + \delta(\mathbf{x}_t)\, \frac{\mathbb{E}_{D(m_0^k)_{0|t}}[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t}{t},
\end{aligned} \tag{37}$$

where $\delta(\mathbf{x}_t) = \gamma \frac{\mathrm{d}D(m_0^k)_t}{\mathrm{d}P_t^{k+1,\gamma}}(\mathbf{x}_t)$. We note that, by Eq (14),

$$p_0^{k+1,\gamma} = (1-\gamma)\, p_0^{k,\gamma} + \gamma\, m_0^k. \tag{38}$$

As a result,

$$P_t^{k+1,\gamma} = (1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t \tag{39}$$

and

$$\delta(\mathbf{x}_t) = \frac{\gamma\, \mathrm{d}D(m_0^k)_t}{\mathrm{d}(1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t}(\mathbf{x}_t), \;\; 1 - \delta(\mathbf{x}_t) = \frac{(1-\gamma)\, \mathrm{d}P_t^{k,\gamma}}{\mathrm{d}(1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t}(\mathbf{x}_t). \tag{40}$$

Thus,

$$\begin{aligned}
\mathbf{s}_t^{k+1}(\mathbf{x}_t) &\overset{(37)}{=} \mathbf{s}_t^k(\mathbf{x}_t)\, \frac{(1-\gamma)\, \mathrm{d}P_t^{k,\gamma}}{\mathrm{d}(1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t}(\mathbf{x}_t) \\
&\qquad + \frac{\mathbb{E}_{D(m_0^k)_{0|t}}[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t}{t}\, \frac{\gamma\, \mathrm{d}D(m_0^k)_t}{\mathrm{d}(1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t}(\mathbf{x}_t) \\
&\overset{(36)}{=} \frac{\mathbb{E}_{P^{k,\gamma}_{0|t}}(\mathbf{x}_0|\mathbf{x}_t) - \mathbf{x}_t}{t}\, \frac{(1-\gamma)\, \mathrm{d}P_t^{k,\gamma}}{\mathrm{d}(1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t}(\mathbf{x}_t) \\
&\qquad + \frac{\mathbb{E}_{D(m_0^k)_{0|t}}[\mathbf{x}_0|\mathbf{x}_t] - \mathbf{x}_t}{t}\, \frac{\gamma\, \mathrm{d}D(m_0^k)_t}{\mathrm{d}(1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t}(\mathbf{x}_t) \\
&= -\frac{1}{t}\mathbf{x}_t + \frac{1}{t}\int_{\mathbf{x}_0' \in \mathbb{R}^d} \mathbf{x}_0'\, \frac{\mathrm{d}\,(1-\gamma)P_{0t}^{k,\gamma} + \gamma D(m_0^k)_{0t}}{\mathrm{d}\,(1-\gamma)\, P_t^{k,\gamma} + \gamma\, D(m_0^k)_t}(\mathbf{x}_0', \mathbf{x}_t) \\
&= -\frac{1}{t}\mathbf{x}_t + \frac{1}{t}\int_{\mathbf{x}_0' \in \mathbb{R}^d} \mathbf{x}_0'\, \frac{\mathrm{d}\, P_{0t}^{k+1,\gamma}}{\mathrm{d}\, P_t^{k+1,\gamma}}(\mathbf{x}_0', \mathbf{x}_t) \\
&= \frac{\mathbb{E}_{P^{k+1,\gamma}_{0|t}}(\mathbf{x}_0|\mathbf{x}_t) - \mathbf{x}_t}{t},
\end{aligned}$$

which is the negative backward drift of M^{k+1}. □

Corollary 1. *For $\kappa > 0$, we have $\frac{\mathrm{d}}{\mathrm{d}\kappa} D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^\kappa) \le -D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^\kappa)$. Additionally,*

$$\inf_{\kappa \in [0,\mathcal{K}]} \left|\Phi_{p_{data}}(\mathbf{u}) - \Phi_{p_0^\kappa}(\mathbf{u})\right| \le \exp\left(\tfrac{\tau}{2}\|\mathbf{u}\|^2\right) \left(\tfrac{2M}{\mathcal{K}}\right)^{1/2}$$

for $\mathcal{K} > 0$, $\mathbf{u} \in \mathbb{R}^d$ and $M = D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_{\mathcal{E}_{\mathrm{clean}}})$.

Proof. According to Eq (29), we have

$$\frac{1}{\gamma}\left(D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^{k+1,\gamma}) - D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^{k,\gamma})\right) \le -D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^{k,\gamma}), \tag{41}$$

for all $\gamma > 0$ and $k \in \mathbb{N}$.

Fix $\kappa > 0$ and let $\{\gamma_i\} \to 0$ with $k_i = \kappa/\gamma_i \in \mathbb{N}$. Then $p_0^{k_i,\gamma_i} \to p_0^\kappa$ via Euler approximation. Taking the limit yields:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\kappa} D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^\kappa) &= \lim_{i\to\infty} \frac{1}{\gamma_i}\left(D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^{k_i+1,\gamma_i}) - D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^{k_i,\gamma_i})\right) \\ &\overset{(41)}{\le} \lim_{i\to\infty} -D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^{k_i,\gamma_i}) = -D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^\kappa),\end{aligned} \tag{42}$$

establishing the monotonicity claim.

In addition, integrating both sides of Eq (42) over $[0, \mathcal{K}]$ gives:

$$D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^{\mathcal{K}}) - D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^0) \le -\int_0^{\mathcal{K}} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^\kappa)\,\mathrm{d}\kappa. \tag{43}$$

As a result,

$$\inf_{\kappa\in[0,\mathcal{K}]} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^\kappa) \le \frac{1}{\mathcal{K}} D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_0^0) = \frac{1}{\mathcal{K}} D_{\mathrm{KL}}(p_{\mathrm{data}} \,\|\, p_{\mathcal{E}_{\mathrm{clean}}}).$$

Applying Prop 4 concludes the convergence argument in the corollary. □

Proposition 3. *Suppose $\mathbf{s}_t^k$ is approximated by a neural network $\hat{\mathbf{s}}_t^k$, and let δ_k satisfy Eq (16) with $\sum_{k=1}^\infty \delta_k < \infty$. For $\rho, \Gamma \in (0,1)$ and $\gamma_0, v_0 > 0$, choose the step size at iteration k as*

$$\gamma_k = \min\Big(\tfrac{\gamma_0}{\sqrt{v_k}}, \Gamma\Big), \quad v_k = \rho v_{k-1} + (1-\rho)\delta_k^2. \tag{17}$$

Then $p_0^k \to p_{data}$ as $k \to \infty$.

Proof. The proof follows a similar idea used in the proof of Prop 1. Recall Eq (28):

$$\begin{aligned}\mathcal{F}(p_0^k) = D_{\mathrm{KL}}(P^* \,\|\, P^k) &= D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + \mathbb{E}_{p^*}\left[\int_0^\tau \frac{1}{2}\left\|\nabla \log p_t(\mathbf{x}_t) - \mathbf{s}_t^k(\mathbf{x}_t)\right\|^2\right] \\ &= D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + D_{\mathrm{KL}}(P^* \,\|\, M^k) \ge D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + \tfrac{1}{\gamma}\Big(\mathcal{F}(p_0^{k+1}) - \mathcal{F}(p_0^k)\Big) + \mathcal{F}(p_0^k).\end{aligned}$$

When $\mathbf{s}^k$ is approximated by $\hat{\mathbf{s}}^k$, and $\gamma = \gamma_k$, we instead have

$$\begin{aligned}\mathcal{F}(p_0^k) = D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + D_{\mathrm{KL}}(P^* \,\|\, M^k) &\ge D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + D_{\mathrm{KL}}(P^* \,\|\, \hat{M}^k) - \delta_k \\ &\ge D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) + \tfrac{1}{\gamma_k}\Big(\mathcal{F}(p_0^{k+1}) - \mathcal{F}(p_0^k)\Big) + \mathcal{F}(p_0^k) - \delta_k.\end{aligned}$$

Cancelling out $\mathcal{F}(p_0^k)$ on both sides followed by rearrangement yields:

$$\mathcal{F}(p_0^k) - \mathcal{F}(p_0^{k+1}) \ge \gamma_k\Big(D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) - \delta_k\Big). \tag{44}$$

Apply telescoping to obtain

$$\mathcal{F}(p_0) \geq \sum_{k=0}^{K-1} \gamma_k D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) - \sum_{k=0}^{K-1} \gamma_k \delta_k. \tag{45}$$

As a result,

$$\mathcal{F}(p_0) + \sum_{k=0}^{K-1} \gamma_k \delta_k \geq \sum_{k=0}^{K-1} \gamma_k D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) \geq \left(\sum_{k=1}^{K-1} \gamma_k\right) \left(\frac{1}{K} \sum_{k=1}^{K-1} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k)\right), \tag{46}$$

where the last inequality is by Chebyshev's sum inequality. As a result,

$$\frac{1}{K} \sum_{k=0}^{K-1} D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) \leq \frac{\mathcal{F}(p_0)}{\sum_{k=0}^{K-1} \gamma_k} + \frac{\sum_{k=0}^{K-1} \gamma_k \delta_k}{\sum_{k=0}^{K-1} \gamma_k}. \tag{47}$$

Assume we can show $\sum_k \gamma_k \to \infty$ and $\sum_k \gamma_k \delta_k < \infty$ (also known as the Robbins-Monro requirements). Then the left-hand side goes to zero as $K \to \infty$. Therefore, $D_{\mathrm{KL}}(p_\tau^* \,\|\, p_\tau^k) \to 0$. Then, applying the same argument as the one in the proof of Prop 1, we conclude $p_0^k \to p_{\mathrm{data}}$.

We complete the proof by showing that the Robbins-Monro requirements are indeed satisfied.

Recall that the step size at iteration k is defined by

$$\gamma_k = \min\left(\frac{\gamma_0}{\sqrt{v_k}}, \Gamma\right), \qquad\qquad v_k = \rho v_{k-1} + (1-\rho)\delta_k^2, \tag{48}$$

with $\rho, \Gamma \in (0,1)$, $\gamma_0, v_0 > 0$, and $\sum_{k=1}^{\infty} \delta_k < \infty$.

Decay of v_k. Unrolling the recursion gives

$$v_k = \rho^k v_0 + (1-\rho) \sum_{j=1}^{k} \rho^{\,k-j}\, \delta_j^2. \tag{49}$$

Since $\rho \in (0,1)$, the first term vanishes as $k \to \infty$. Moreover, $\sum_j \delta_j^2 < \infty$ (because $\delta_j^2 \leq \delta_j$ eventually), so the convolution with the geometric kernel also vanishes. Therefore, $v_k \to 0 \quad$ as $k \to \infty$.

Divergence of $\sum_k \gamma_k$. Because $v_k \to 0$, for large k we have $\gamma_0/\sqrt{v_k} \geq \Gamma$, hence

$$\gamma_k = \Gamma > 0 \quad \text{eventually.}$$

Therefore

$$\sum_{k=1}^{\infty} \gamma_k \;\geq\; \sum_{k=K}^{\infty} \Gamma \;=\; \infty,$$

so the first Robbins–Monro condition holds.

Convergence of $\sum_k \gamma_k \delta_k$. For all k,

$$\gamma_k \leq \Gamma \quad \Longrightarrow \quad \gamma_k \delta_k \leq \Gamma \delta_k.$$

Hence

$$\sum_{k=1}^{\infty} \gamma_k \delta_k \;\leq\; \Gamma \sum_{k=1}^{\infty} \delta_k \;<\; \infty,$$

by assumption.

Thus both Robbins–Monro requirements are satisfied:

$$\sum_{k=1}^{\infty} \gamma_k = \infty, \qquad \sum_{k=1}^{\infty} \gamma_k \delta_k < \infty.$$

□

A.5 Construction of δ_k for the γ-Adaptive Online SFBD Algorithm

In this section, we provide additional implementation details on how we construct δ_k for the γ-adaptive Online SFBD algorithm.

As indicated in Prop 3, we need to design a sequence $\{\delta_k\}$ satisfying Eq (16). Since the scaling factor can be absorbed into γ_0, and Prop 3 imposes no constraint on γ_0 other than positivity, our goal reduces to constructing δ_k such that

$$\delta_k \geq K \max\Big\{0,\ D_{\mathrm{KL}}(P^* \,\|\, \hat{M}_k) - D_{\mathrm{KL}}(P^* \,\|\, M_k)\Big\} \tag{50}$$

for some $K > 0$. Hence, we focus on constructing a sequence that reflects the relative variation in the difference between $D_{\mathrm{KL}}(P^* \,\|\, \hat{M}_k)$ and $D_{\mathrm{KL}}(P^* \,\|\, M_k)$.

Recall that, by Lem 1, we have

$$D_{\mathrm{KL}}(P^* \,\|\, \hat{M}_k) = D_{\mathrm{KL}}(p^*_\tau \,\|\, p^*_\tau) + \mathbb{E}_{p^*}\left[\int_0^\tau \frac{1}{2}\big\|\nabla \log p_t(\mathbf{x}_t) - \hat{\mathbf{s}}^k_t(\mathbf{x}_t)\big\|^2\right] = \mathbb{E}_{p^*}\left[\int_0^\tau \frac{1}{2}\big\|\nabla \log p_t(\mathbf{x}_t) - \hat{\mathbf{s}}^k_t(\mathbf{x}_t)\big\|^2\right].$$

In practice, this quantity is not directly accessible. Conventional diffusion models instead learn to align the estimated score $\hat{\mathbf{s}}^k_t$ with the true score $\nabla \log p_t$ via conditional score matching, which relies on noisy estimates of the score function. However, this approach cannot be applied here, as it introduces excessive variance and requires direct access to the ground-truth clean data distribution p_{data}, which is unavailable in our setting.

Due to these challenges and the limited availability of clean samples, we instead measure the distance between the denoised samples generated using $\hat{\mathbf{s}}^k_t$ and the clean samples $\mathcal{E}_{\mathrm{clean}}$ in a suitable feature space as a proxy for the above quantity. This approximation is justified, as the quality of denoised samples directly depends on the accuracy of $\hat{\mathbf{s}}^k_t$ across all $t \in [0, \tau]$.

While the Fréchet Inception Distance (FID) (Heusel et al., 2017) is the classical choice for quantifying such distances, its standard implementation typically requires at least 2,048 clean samples to ensure a well-conditioned covariance estimate – substantially more than what is available in our setting. Moreover, FID is known to be biased when computed on small sample sets, which further limits its reliability under our data constraints.

To address these issues, we adopt the Kernel Inception Distance (KID) (Bińkowski et al., 2018), which provides an unbiased estimator of the squared Maximum Mean Discrepancy (MMD) between feature embeddings and exhibits greater robustness with limited sample sizes. Empirically, we observe that the estimation remains stable and reliable even when the clean sample set contains as few as 50 images.

Likewise, we measure the KID between the clean sample set and the running denoised sample set $\mathcal{E}$ as a practical proxy for $D_{\mathrm{KL}}(P^* \,\|\, M_k)$. Based on this, we define

$$\delta^2 \propto \max\Big(0,\ \mathsf{KID}(\mathcal{E}_{\mathrm{clean}}, \mathcal{E}_{\boldsymbol{\phi}}) - \mathsf{KID}(\mathcal{E}_{\mathrm{clean}}, \mathcal{E})\Big), \tag{51}$$

where we use δ^2 instead of δ so that v_k admits a clear interpretation as the exponential moving average of the KID difference δ_k^2. This formulation makes v_k a smoother and lower-variance estimator of the KID gap across iterations.

Ensuring the Convergence of $\sum_k \delta_k$ via Skipping Denoised Set Updates. As discussed in Sec 5, by conditionally skipping the denoised sample updates (i.e., setting $\lambda = 0$) as specified in Alg 2, we ensure that the effective sequence δ_k satisfies the convergence condition $\sum_k \delta_k < \infty$.

To see this, note that the alternative projection updates are performed only when the denoised sample set is refreshed. In other words, the deviation term δ_k of an iteration is counted as effective only when $\lambda > 0$ in that iteration. As a result, by Alg 2, we have

$$v_k = \rho v_{k-1} + (1-\rho)\,\delta_k^2, \qquad 0 < \rho < 1,$$

and the skipping mechanism ensures

$$\delta_k^2 \leq \eta\, v_{k-1}$$

for $\eta \in (0,1)$. Then

$$v_k \le \big[\rho + (1-\rho)\eta\big]\, v_{k-1} =: \beta\, v_{k-1}.$$

As $\eta < 1$ (so $\beta = \rho + (1-\rho)\eta < 1$),

$$v_k \le \beta^k v_0 \quad \text{(geometric decay)}.$$

Plugging back into the δ_k^2 bound gives

$$\delta_k^2 \le \eta\, v_{k-1} \le \eta\, \beta^{\,k-1} v_0, \qquad \beta = \rho + (1-\rho)\eta < 1.$$

So δ_k is upper-bounded by the geometric sequence $\sqrt{\eta v_0/\beta}\,\beta^{\frac{k}{2}}$. As a result, $\sum_k \delta_k < \sqrt{\frac{\eta v_0}{\beta}} \sum_k \beta^{\frac{k}{2}} < \infty$.

Finally, we note that, in theory, $\hat{M}^k \to M^k$ given sufficiently many gradient descent updates. Consequently, γ should not be set to zero infinitely often. In practice, however, due to the inherent approximation error of neural networks, the denoised set update naturally stabilizes and ceases near the end of the Online SFBD training.

A.6 A Variant of γ-SFBD

Since the copyright-free clean samples are drawn from the true data distribution, it is practical to mix them with the denoised samples during denoiser updates to enhance overall sample quality. In particular, we generally believe that

$$\mathcal{L}_{\text{vis}}(\alpha\, p_{\text{clean}} + (1-\alpha)\, p_{\text{denoise}}) \le \mathcal{L}_{\text{vis}}(p_{\text{denoise}}), \tag{52}$$

where $\mathcal{L}_{\text{vis}}(p)$ denotes an aggregate loss that measures the visual quality of samples drawn from distribution p, and p_{clean} and p_{denoise} represent the distributions of clean and denoised samples, respectively. α depends on the ratios between the numbers of clean samples and the denoised samples. In practice, we have observed that this is always true when $\mathcal{L}_{\text{vis}}$ is instantiated as the FID.

We note that this heuristic technique can be naturally covered in our framework with little work. In particular, we can replace the original γ-Diff Proj with

(γ-Diff Proj-v2) $\quad P^{k+1,\gamma} = \operatorname{argmin}_{P\in\mathcal{D}}\, \alpha\, D_{\text{KL}}(P_{\text{clean}} \,\|\, P) + (1-\alpha)\,\big[(1-\gamma)\, D_{\text{KL}}(P^{k,\gamma} \,\|\, P) + \gamma\, D_{\text{KL}}(M^k \,\|\, P)\big]$

where $P_{\text{clean}} = D(p_{\text{clean}})$ is a fixed diffusion process in $\mathcal{D}$ with the initial distribution having density p_{clean}.

Applying a derivation similar to the one in Sec A.3, again through the disintegration theorem, we have

$$\begin{aligned}
&\arg\min_{P\in\mathcal{D}}\ \alpha\, D_{\text{KL}}(P_{\text{clean}} \,\|\, P) + (1-\alpha)\Big[(1-\gamma)\, D_{\text{KL}}(P^{k,\gamma} \,\|\, P) + \gamma\, D_{\text{KL}}(M^k \,\|\, P)\Big] \\
=&\arg\min_{P\in\mathcal{D}}\ \alpha\, D_{\text{KL}}(p_{\text{clean}} \,\|\, p_0) + (1-\alpha)\Big[(1-\gamma)\, D_{\text{KL}}(p_0^{k,\gamma} \,\|\, p_0) + \gamma\, D_{\text{KL}}(m_0^k \,\|\, p_0)\Big] + \text{Const.} \\
=&\operatorname*{argmin}_{P\in\mathcal{D}}\ -\int_{\mathbb{R}^d} \alpha p_{\text{clean}}(\mathbf{x}_0) + (1-\alpha)\,\Big[(1-\gamma)\, p_0^{k,\gamma}(\mathbf{x}_0) + \gamma\, m_0^k(\mathbf{x}_0)\Big] \log p_0(\mathbf{x}_0)\ \mathrm{d}\mathbf{x}_0 + \text{Const.} \\
=&\arg\min_{P\in\mathcal{D}}\ D_{\text{KL}}(\alpha p_{\text{clean}} + (1-\alpha)\big[(1-\gamma)\, p_0^{k,\gamma} + \gamma\, m_0^k\big] \,\|\, p_0).
\end{aligned}$$

As a result,

$$\begin{aligned}
\tilde{p}_0^{k+1,\gamma} &= \alpha\, p_{\text{clean}} + (1-\alpha)\,\Big[(1-\gamma)\, p_0^{k,\gamma} + \gamma\, m_0^k\Big] \\
&= \alpha\, p_{\text{clean}} + (1-\alpha)\, p_0^{k+1,\gamma},
\end{aligned}$$

where $p_0^{k+1,\gamma}$ is obtained via the standard γ-D-Proj defined in Eq (13), and corresponds to p_{denoise} in Eq (52).

This variant of γ-D-Proj therefore recovers the exact update rule underlying the heuristic practice of mixing clean and denoised samples prior to fine-tuning the diffusion model.

Notably, when $\gamma = 1$, this variant reduces to a form of the original SFBD algorithm, which was heuristically employed in the initial SFBD paper (Lu et al., 2025) to boost model performance—despite lacking theoretical justification at the time.

A.7 Relationship to Consistency Constraint-based Methods

In Sec 5, we argued that consistency-constraint-based (CC-based) methods enforcing consistency only between $r = 0$ and a positive time s can be viewed as a special case of Online SFBD with $m = 1$ with $p^{k,\gamma}$ approximated by m_0^k. In this section, we elaborate on this connection and extend the discussion to more general CC-based methods that enforce consistency between arbitrary time steps $r < s$.

Enforcing Consistency Between $r = 0$ and $s > 0$. We assume the denoiser network satisfies $D_{\boldsymbol{\phi}}(\cdot, 0) = \mathrm{Id}(\cdot)$, a condition explicitly enforced in EDM-based implementations. This design is both natural and intuitive, as $D_{\boldsymbol{\phi}}(\mathbf{x}_0, 0)$ approximates $\mathbb{E}_{p_{0|t}}[\mathbf{x}_0 \mid \mathbf{x}_0]$ at $t = 0$, which is $\mathbf{x}_0$ for any p_{0t} induced by some distribution p_0 argumented by the forward transition kernel $p_{t|0}$ in Eq (2). It has been adopted in all CC-based methods (Daras et al., 2024, 2025), the SFBD framework (Lu et al., 2025), and our work.

Lu et al. (2025) showed that, under this assumption, the denoising loss in Eq (4) is equivalent to the consistency loss Eq (6). For completeness, we include their derivation as follows:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{p_0}\mathop{\mathbb{E}}_{p_{s|0}}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbf{x}_0\|^2\right] = \mathbb{E}_{p_s}\mathbb{E}_{p_{0|s}}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbf{x}_0\|^2\right] \\
=& \mathbb{E}_{p_s}\mathbb{E}_{p_{0|s}}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s] + \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s] - \mathbf{x}_0\|^2\right] \\
=& \mathbb{E}_{p_s}\mathbb{E}_{p_{0|s}}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s]\|^2\right] + \underbrace{\mathbb{E}_{p_s}\mathbb{E}_{p_{0|s}}\left[\|\mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s] - \mathbf{x}_0\|^2\right]}_{\text{Const.}} \\
&\qquad + 2\underbrace{\mathbb{E}_{p_s}\mathbb{E}_{p_{0|s}}\left[\left\langle D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s], \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s] - \mathbf{x}_0\right\rangle\right]}_{=0} \\
=& \mathbb{E}_{p_s}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s]\|^2\right] + \text{Const.} \\
\overset{\text{(arch ass)}}{=}& \mathbb{E}_{p_s}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[D_{\boldsymbol{\phi}}(\mathbf{x}_0, 0)]\|^2\right] + \text{Const.},
\end{aligned}
\tag{53}
$$

which is the consistency loss in Eq (6) when $r = 0$. Therefore,

$$
\mathop{\mathbb{E}}_{p_0}\mathop{\mathbb{E}}_{p_{s|0}}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbf{x}_0\|^2\right] \equiv \mathbb{E}_{p_s}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[D_{\boldsymbol{\phi}}(\mathbf{x}_0, 0)]\|^2\right] \tag{54}
$$

up to a constant, establishing the equivalence between the denoising loss used in Alg 1 (M-proj) and the consistency loss in CC-based methods.

For Online SFBD, at the k-th iteration, we have

$$
p_0 = p_0^{k+1,\gamma} = (1-\gamma)\, p_0^{k,\gamma} + \gamma\, m_0^k, \tag{55}
$$

as presented in Eq (14), and p_s is

$$
p_s = p_0 * \mathcal{N}(0, s\mathbf{I}) = p_0^{k,\gamma} * \mathcal{N}(0, s\mathbf{I}). \tag{56}
$$

Instead, in CC-based methods,

$$
p_0 \approx m_0^k. \tag{57}
$$

To see this, note that practical implementations of CC-based methods typically rely on two approximations:

(a) p_s is approximated using samples generated by adding Gaussian noise to corrupted data, where s is chosen no less than the corruption level τ (Daras et al., 2025);

(b) $p_{0|s}$ is estimated via the backward SDE Eq (3), with the drift term approximated by the current network (i.e., $\mathbf{s}^k$).

For simplicity, we restrict the discussion to the case $s = \tau$. For the cases $s > \tau$, they reduce to the case $s = \tau$ under the assumption that the score function above τ is accurately estimated, which can be achieved by training the model through the ASM loss combined with the noisy samples (Daras et al., 2024, 2025). These approximations

essentially define the backward SDE process M^k, whose marginal at $t=0$ is m_0^k, serving as the effective p_0 in CC-based training.

Note that CC-based methods form $(\mathbf{x}_0, \mathbf{x}_s)$ pairs using the backward SDE, whereas Online SFBD uses the forward process. As CC-based methods assume that corrupted samples can be approximated as drawn from p_s, the two pairing schemes are equivalent: both the forward and backward SDE yield the same joint distribution $p_{0s}(\mathbf{x}_0, \mathbf{x}_s)$, as discussed following Eq (3).

This approximation is reasonable when m_0^k evolves slowly and γ is not too small, as discussed in the main text. This condition typically holds in practice, as CC-based methods only take one gradient step per iteration. Moreover, CC-based methods often adopt the same pretraining strategy as OSFBD, allowing the network to learn global structure early on. As a result, drift updates during subsequent training are small, and m_0^k changes slightly across iterations.

Enforcing Consistency Between $r < s$. For any pair $r < s$, we note that

$$\begin{aligned}
&\mathbb{E}_{p_s}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[D_{\boldsymbol{\phi}}(\mathbf{x}_0, 0)]\|^2\right] \\
\overset{\text{(arch ass)}}{=}\ &\mathbb{E}_{p_s}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s]\|^2\right] \\
=\ &\mathbb{E}_{p_s}\left[\left\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{r|s}}\left[\mathbb{E}_{p_{0|r}}[\mathbf{x}_0|\mathbf{x}_r]\middle|\mathbf{x}_s\right]\right\|^2\right],
\end{aligned} \tag{58}$$

where the final equality uses the fact that the backward process is Markovian. In more detail, since the forward process is Brownian and thus Markovian, its time reversal (the backward process described by Eq (3)) is also Markovian. Consequently, we can justify:

$$\begin{aligned}
\mathbb{E}_{p_{r|s}}\left[\mathbb{E}_{p_{0|r}}[\mathbf{x}_0|\mathbf{x}_r]\middle|\mathbf{x}_s\right] &= \int \mathbf{x}_0 \left(\int p_{0|r}(\mathbf{x}_0|\mathbf{x}_r)\, p_{r|s}(\mathbf{x}_r|\mathbf{x}_s)\ \mathrm{d}\mathbf{x}_r\right) \mathrm{d}\mathbf{x}_0 \\
&= \int \mathbf{x}_0 \left(\int p_{0r|s}(\mathbf{x}_0, \mathbf{x}_r|\mathbf{x}_s)\ \mathrm{d}\mathbf{x}_r\right) \mathrm{d}\mathbf{x}_0 \\
&= \int \mathbf{x}_0\, p_{0|s}(\mathbf{x}_0|\mathbf{x}_s)\ \mathrm{d}\mathbf{x}_0 \\
&= \mathbb{E}_{p_{0|s}}[\mathbf{x}_0|\mathbf{x}_s].
\end{aligned}$$

As a result, by Eq (58), we have

$$\begin{aligned}
\mathcal{L}_{\text{con}}(\boldsymbol{\phi}, 0, s) &= \mathbb{E}_{p_s}\left[\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{0|s}}[D_{\boldsymbol{\phi}}(\mathbf{x}_0, 0)]\|^2\right] \\
&= \mathbb{E}_{p_s}\left[\left\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{r|s}}\left[\mathbb{E}_{p_{0|r}}[\mathbf{x}_0|\mathbf{x}_r]\middle|\mathbf{x}_s\right]\right\|^2\right] \\
&\approx \mathbb{E}_{p_s}\left[\left\|D_{\boldsymbol{\phi}}(\mathbf{x}_s, s) - \mathbb{E}_{p_{r|s}}\left[D_{\mathsf{stopgrad}(\boldsymbol{\phi})}(\mathbf{x}_r, r)\middle|\mathbf{x}_s\right]\right\|^2\right] \\
&= \mathcal{L}_{\text{con}}(\boldsymbol{\phi}, r, s)
\end{aligned}$$

where $\mathbb{E}_{p_{0|r}}[\mathbf{x}_0|\mathbf{x}_r]$ is approximated using the current network, and stopgrad indicates a stop-gradient operation.

This suggests that enforcing consistency between arbitrary time pairs $r < s$ is effectively equivalent to enforcing it between 0 and s, so the same argument for $r = 0$ applies.

A.8 Auxiliary Results for References

Proposition 4 (Lu et al. 2025, Prop 1)**.** *Let p and q be two distributions defined on $\mathbb{R}^d$. For all $\mathbf{u} \in \mathbb{R}^d$,*

$$|\Phi_p(\mathbf{u}) - \Phi_q(\mathbf{u})| \le \exp\left(\frac{\tau}{2}\|\mathbf{u}\|^2\right)\sqrt{2\, D_{\mathrm{KL}}(p * h \| q * h)},$$

where $h \sim \mathcal{N}(\mathbf{0}, \tau\mathbf{I})$.

Lemma 1 (Pavon and Wakolbinger 1991, Vargas et al. 2021)**.** *Given two SDEs:*

$$\mathrm{d}\mathbf{x}_t = \mathbf{f}_i(\mathbf{x}_t, t)\,\mathrm{d}t + \delta\,\mathrm{d}\mathbf{w}_t, \quad \mathbf{x}_0 \sim p_0^{(i)}(\mathbf{x}) \quad t \in [0, T] \tag{59}$$

for $i = 1, 2$. Let $P^{(i)}_{0:T}$, for $i = 1, 2$, be the path measure induced by them, respectively. Then we have,

$$D_{\mathrm{KL}}(P^{(1)}_{0:T} \,\|\, P^{(2)}_{0:T}) = D_{\mathrm{KL}}(p^{(1)}_0 \,\|\, p^{(2)}_0) \;+\; \mathbb{E}_{P^{(1)}_{0:T}}\Big[\int_0^T \frac{1}{2} \left\|\mathbf{f}_1(\mathbf{x}_t, t) - \mathbf{f}_2(\mathbf{x}_t, t)\right\|^2 dt\Big]. \tag{60}$$

A similar result also applies to a pair of backward SDEs as well, where $p^{(i)}_0$ is replaced with $p^{(i)}_T$.

Proof. By the disintegration theorem (e.g., see Vargas et al. 2021, Appx B), we have

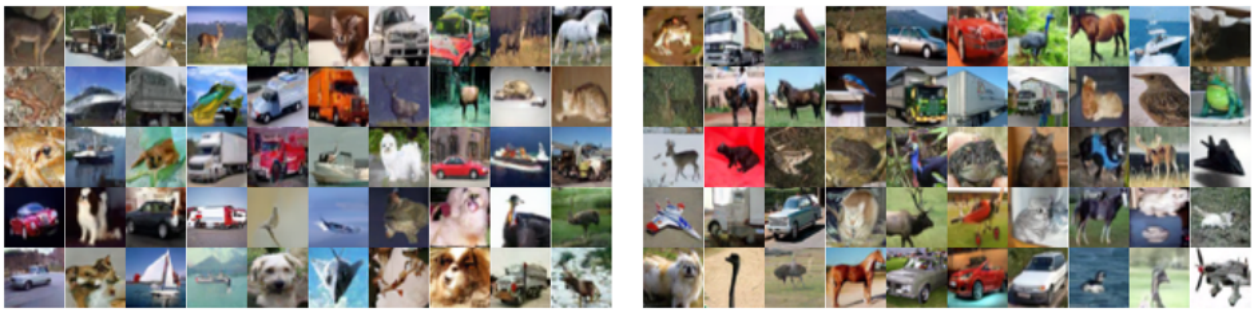
$$D_{\mathrm{KL}}(P_1 \,\|\, P_2) = D_{\mathrm{KL}}(p^{(1)}_0 \,\|\, p^{(2)}_0) \;+\; \mathbb{E}_{P^{(1)}_{0:T}}\left[\log \frac{\mathrm{d}P^{(1)}_{0:T}(\cdot|\mathbf{x}_0))}{\mathrm{d}P^{(2)}_{0:T}(\cdot|\mathbf{x}_0)}\right], \tag{61}$$

where $P^{(i)}_{0:T}(\cdot|\mathbf{x}_0)$ is the conditioned path measure of $P^{(i)}_{0:T}$ given the initial point $\mathbf{x}_0$. Then, applying the Girsanov theorem (Kailath, 1971; Oksendal, 2003) on the second term yields the desired result. □

B SAMPLING RESULTS

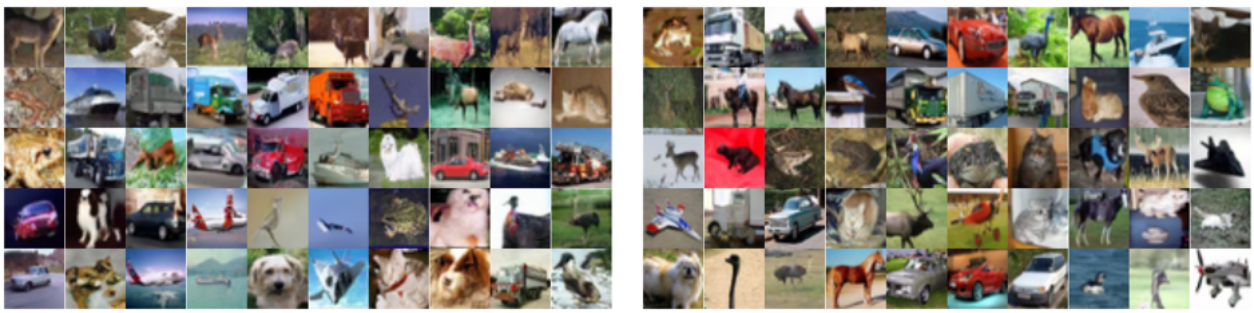
In this section, we present model-generated samples used to compute FID scores in Sec 6, corresponding to the benchmark results in Table 1 and Table 2. We also include denoised samples at the final training step.

B.1 CIFAR-10 (fixed γ)



(a) Generated (FID: 3.22) (b) Denoised (FID: 1.11)

Figure 3: 50 clean samples, noise level $\sigma = 0.2$



(a) Generated (FID: 2.73) (b) Denoised (FID: 1.02)

Figure 4: 5,000 clean samples (10%), noise level $\sigma = 0.2$.

(a) Generated (FID: 6.56) (b) Denoised (FID: 4.84)

Figure 5: 2,000 clean samples (4%), noise level $\sigma = 0.59$.

B.2 CIFAR-10 (adaptive γ)

(a) Generated (FID: 3.12) (b) Denoised (FID: 1.05)

Figure 6: 50 clean samples, noise level $\sigma = 0.2$

(a) Generated (FID: 2.49) (b) Denoised (FID: 1.01)

Figure 7: 5,000 clean samples (10%), noise level $\sigma = 0.2$.

(a) Generated (FID: 6.21) (b) Denoised (FID: 4.80)

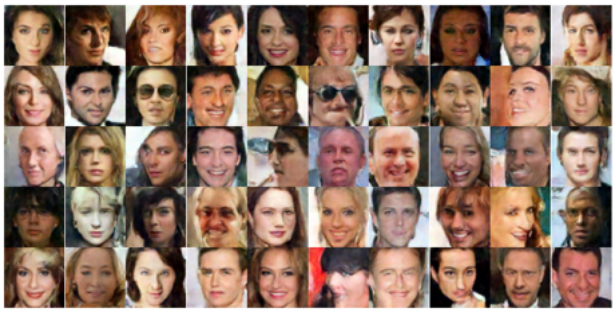
Figure 8: 2,000 clean samples (4%), noise level $\sigma = 0.59$.

B.3 CelebA (fixed γ)

(a) Generated (FID: 3.23)

(b) Denoised (FID: 1.07)

Figure 9: 50 clean samples, noise level $\sigma = 0.2$.



(a) Generated (FID: 27.09)

(b) Denoised (FID: 24.31)

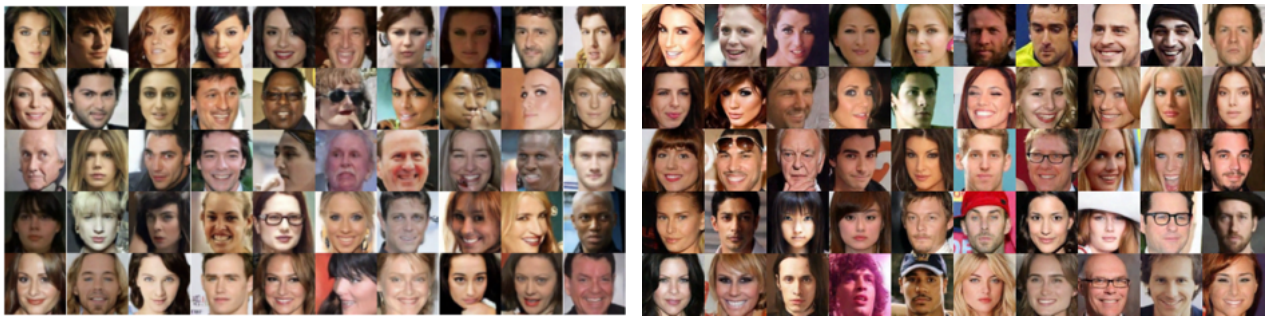
Figure 10: 50 clean samples, noise level $\sigma = 1.38$.

(a) Generated (FID: 5.72)

(b) Denoised (FID: 4.28)

Figure 11: 1,500 clean samples, noise level $\sigma = 1.38$.

B.4 CelebA (adpative γ)



(a) Generated (FID: 3.19) (b) Denoised (FID: 1.06)

Figure 12: 50 clean samples, noise level $\sigma = 0.2$.

(a) Generated (FID: 20.21) (b) Denoised (FID: 19.09)

Figure 13: 50 clean samples, noise level $\sigma = 1.38$.

(a) Generated (FID: 5.40) (b) Denoised (FID: 4.21)

Figure 14: 1,500 clean samples, noise level $\sigma = 1.38$.

C EXPERIMENT CONFIGURATIONS

C.1 Hardware Configurations

All diffusion models were trained on the main process using four NVIDIA A40 or RTX 6000 GPUs, managed by a SLURM scheduling system. The asynchronous denoising process ran concurrently in the background on a

separate RTX 6000 GPU, taking less than 2.5 minutes to update 640 images on CIFAR-10 and under 5 minutes on CelebA.

Training on CIFAR-10 completes in under 5 days, and CelebA experiments in under 8 days.

C.2 Model Architectures

We implement the proposed Online SFBD algorithm using the EDM backbone (Karras et al., 2022), following the configuration described below throughout our empirical studies.

Table 4: Experimental Configuration for CIFAR-10 and CelebA

Parameter	**CIFAR-10**	**CelebA**
General		
Batch Size	512	256
Loss Function	`EDMLoss` (Karras et al., 2022)	`EDMLoss` (Karras et al., 2022)
Denoising Method	2^{nd} order Heun method (EDM) (Karras et al., 2022)	2^{nd} order Heun method (EDM) (Karras et al., 2022)
Sampling Method	2^{nd} order Heun method (EDM) (Karras et al., 2022)	2^{nd} order Heun method (EDM) (Karras et al., 2022)
Sampling steps	18	40
Network Configuration		
Dropout	0.13	0.05
Channel Multipliers	$\{2, 2, 2\}$	$\{1, 2, 2, 2\}$
Model Channels	128	128
Resample Filter	$\{1, 1\}$	$\{1, 3, 3, 1\}$
Channel Mult Noise	1	2
Optimizer Configuration		
Optimizer Class	`RAdam` (Kingma and Ba, 2015; Liu et al., 2020)	`RAdam` (Kingma and Ba, 2015; Liu et al., 2020)
Learning Rate	0.001	0.0002
Epsilon	1×10^{-8}	1×10^{-8}
Betas	(0.9, 0.999)	(0.9, 0.999)

C.3 Datasets

All experiments on CIFAR-10 (Krizhevsky and Hinton, 2009) and CelebA (Liu et al., 2015) are conducted using only the training set. For FID evaluation, the model generates 50,000 samples, and FID is computed against the full training set, which includes both copyright-free and sensitive samples.