BERNOULLI FLOW MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based generative modeling for data with Bernoulli distributions has broad potential applications, but it relies on carefully designed forward processes. Recently, flow matching-based methods have addressed this issue. However, when these methods are naively applied to the Bernoulli distribution, their dependence on predicting the instantaneous velocity field during sampling can introduce invalid Bernoulli parameters, leading to model collapse. To address this challenge, we introduce **Bernoulli Flow Models (BFM)**, a novel generative framework that fuses flow matching with vanilla binary diffusion. BFM ensures valid Bernoulli parameters throughout the sampling process by deriving a one-step forward transition kernel and a closed-form, normalized posterior based on the pre-defined flow-matching probability path in the Bernoulli parameter space. As a result, BFM simplifies the training process of current binary diffusion models and can be easily integrated into existing architectures with minimal modification. We empirically validate the generative performance of BFM on high-dimensional binary manifolds, including Ising model simulations, both unconditional and conditional image generation. Experiments show that our model achieves comparable performance to both continuous and discrete space generative models.

1 INTRODUCTION

The emergence of diffusion-based Sohl-Dickstein et al. (2015); Ho et al. (2020); Song et al. (2020) and flow-matching Lipman et al. (2022); Liu et al. (2022) generative models represents a significant advance in modeling high-dimensional data manifolds of real-valued states. However, Bernoulli distributions, which constitute the fundamental discrete state space, have not undergone the same extent of advancement as their continuous or multi-category discrete counterparts.

In the study of discrete generative models, Continuous-Time Markov Chains (CTMC) serves as the underlying framework for many works, such as Lou et al. (2023), Gat et al. (2024), and Campbell et al. (2024), however, these CTMC-based methods must overcome the challenge of managing the explosion in predictive dimensionality caused by transition matrices. Binary Latent Diffusion Wang et al. (2023) circumvent CTMC by Bernoulli probability transition and keeping Bernoulli parameter valid throughout sampling time by a normalized posterior, however it necessitates a carefully designed forward binarized diffusion process. In contrast, flow matching methods Lipman et al. (2022) Liu et al. (2022) offer a simpler alternative by constructing probability paths through straightforward interpolation between the data and prior distributions, thereby significantly streamlining the training procedure. In flow matching, the generative process is modeled as a deterministic Ordinary Differential Equation (ODE): $\frac{d}{dt}x_t = v_\theta(x_t, t)$, where $v_\theta(x_t, t)$ is the velocity field predicted by the neural network. This ODE describes how a sample x_t evolves over time from the prior distribution p_1 to the data distribution p_0. To sample from the model, one integrates this ODE over time using numerical methods such as the Euler method. However, when applied to Bernoulli distributions, the predicted velocity field can cause parameters to fall outside the valid range [0, 1], leading to invalid values and potential model collapse, as the ODE integration does not constrain the parameters within this range.

To address the above challenges, in this work, we introduce **Bernoulli Flow Models (BFM)**, a novel generative framework that leverages the simplicity of discrete flow models, combined with the Markov posterior Bayesian framework inherent in diffusion processes, to develop a generative model applicable to arbitrary binary distributions. BFM constructs a pre-defined, meaningful probability path within the Bernoulli parameter space. From this path, we derive two critical components: 1)

a *one-step forward transition kernel* that defines how data are corrupted at any given time, and 2) a *closed-form, normalized posterior distribution*. The existence of this analytical posterior is crucial, as it allows for training using a simplified, denoising-based objective reminiscent of standard diffusion models. More importantly, during sampling, the model navigates a *deterministic* trajectory defined by the flow, entirely avoiding the pitfalls of regressing an unconstrained velocity field. This guaranties that all intermediate Bernoulli parameters are valid, preventing model collapse.

Our contributions can be summarized as follows.

- **A New Generative Framework:** We propose Bernoulli Flow Models (BFM), a new novel generative framework that unifies the training simplicity of flow matching with the sampling stability of binary diffusion. BFM ensures the validity of all parameters throughout the generative process.
- **Theoretical Formulation:** We derive a closed-form one-step transition kernel and the corresponding posterior distribution based on a pre-defined flow-based probability path. This derivation enables a straightforward training procedure that simplifies existing approaches to binary diffusion.
- **Sampling Efficiency and Consistency:** Experiments on LSUN Churches 256×256 indicate that BFM outperforms existing baselines methods in terms of inference efficiency. Unlike previous methods that suffer from quality degradation when the Number of Function Evaluations (NFE) is reduced, BFM exhibits high stability and **self-consistency cross-step sampling**. This allows for high-quality generation with fewer sampling steps, significantly reducing computational costs without sacrificing visual fidelity.
- **Ease of Integration and Competitive Performance:** We demonstrate that BFM can be easily integrated into existing architectures for binary data with minimal modification. Through extensive experiments on high-dimensional binary manifolds—including simulations of the Ising model, unconditional image generation, and conditional generation tasks—we show that BFM achieves generative performance comparable to both state-of-the-art continuous and discrete space generative models.

2 RELATED WORK

Discrete Diffusion and Flow Matching Models. Diffusion models for discrete data modeling, introduced for generative modeling in Sohl-Dickstein et al. (2015), have evolved significantly over time. Early works such as Argmax flows Hoogeboom et al. (2021) and D3PM Austin et al. (2021) extended these models to categorical data by considering the noising process as a discrete Markov chain. However, diffusion models still faced challenges when applied to certain discrete data types, like text. To address this underperformance, Lou et al. (2023) extended the score-based diffusion framework, introducing a new Score Entropy loss to estimate the ratios of the data distribution, resulting in the highly competitive SEDD models. Concurrently, Varma et al. (2024) proposed the Glauber Generative Model (GGM), a discrete diffusion model that utilizes the Glauber dynamics (a type of heat bath dynamics) to denoise sequences of tokens. Building on these developments, Campbell et al. (2024) and Gat et al. (2024) leveraged the Continuous-Time Markov Chains (CTMC) framework to create powerful discrete flow models. These models have achieved state-of-the-art results in domains such as protein co-design and code generation, respectively. Following the success of these CTMC-based models, a distinct line of research has introduced a more geometric perspective. Fisher-Flow Davis et al. (2024), for instance, departs from the CTMC framework by viewing categorical distributions as points on a statistical manifold, equipped with the Fisher-Rao metric. Similarly, Statistical Flow Matching (SFM) Cheng et al. (2024) also explores the geometric structure of statistical manifolds, proposing a new framework for discrete flow matching. To address the critical gap in guidance techniques for discrete generative models, Nisonoff et al. (2024) recently introduced the *Discrete Guidance* method, providing a general and principled framework for applying guidance to diffusion and flow-matching-based discrete state-space models.

Binary Diffusion Models. Despite the pioneering work of Sohl-Dickstein et al. (2015) in applying diffusion models to binary data, their adoption for binary tasks has remained limited due to suboptimal generative performance, particularly in high-dimensional binary manifolds. Recently, leveraging advancements in vector quantized encoder-decoder techniques Van Den Oord et al. (2017) Esser

et al. (2021), Wang et al. (2023) introduced a vector quantized encoder-decoder with a binary latent space. This approach enables the application of diffusion models in the binary latent space, achieving performance competitive with continuous latent diffusion models. Wolleb et al. (2024) further extends this approach by incorporating a multi-scale Bernoulli diffusion model, which allows for the capture of more complex patterns in the latent space, thereby enhancing the ability to detect subtle anomalies. More recently, Kinakh & Voloshynovskiy (2025) proposed the Binary Diffusion Probabilistic Model (BDPM), which is tailored for binary data by leveraging XOR-based noise and binary cross-entropy loss, enabling highly efficient inference with fewer steps despite a lack of detailed theoretical proof.

3 BACKGROUND

3.1 FLOW MATCHING

Given a target data distribution $X_0 \sim \pi_0$ and a prior distribution $X_1 \sim \pi_1$ that is easy to sample from (typically Guassian $\mathcal{N}(0, I)$ or uniform noise). Flow Matching (FM) is a framework for generative modeling that constructs a probability path $(p_t)_{0\leq t\leq 1}$ between a target data distribution $p_0 = \pi_0$ a known prior distribution $p_1 = \pi_1$ during the forward process. The core idea is to learn a *velocity field* $u_t^\theta : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ implemented via a neural network with parameters θ, whose flow ψ_t transforms samples from π_1 to π_0 through an ODE:

$$\frac{d}{dt}\psi_t(x) = u_t^\theta(\psi_t(x)), \quad \psi_0(x) = x. \tag{1}$$

After training, new samples $X_0 \sim \pi_0$ are generated by solving the ODE starting from the prior distribution samples $X_1 \sim \pi_1$ with the learnd velocity field $u_t^{\theta^*}$.

The method operates in two stages: 1)*Probability Path Design.* A conditional optimal transport path is constructed by aggregating per-datapoint Gaussian paths:

$$p_t(x) = \int p_{t|0}(x|x_0)q(x_0)dx_0, \quad p_{t|0}(x|x_0) = \mathcal{N}\left(x \mid tx_0, (1-t)^2 I\right).$$

This corresponds to the linear interpolation:

$$X_t = tX_0 + (1-t)X_1, \quad X_0 \sim \pi_0, X_1 \sim \pi_1. \tag{2}$$

2)*Velocity Field Regression.* The velocity field u_t^θ is trained to match the *conditional velocity field*:

$$u_t(x|x_0) = \frac{x_0 - x}{1-t}, \tag{3}$$

using the Conditional Flow Matching (CFM) loss:

$$\begin{aligned} \mathcal{L}_{\text{CFM}}(\theta) &= \mathbb{E}_{t,X_0,X_1}\left\|u_t^\theta(X_t) - u_t(X_t|X_0)\right\|^2 \\ &= \mathbb{E}_{t\sim\mathcal{U}[0,1],X_0\sim\pi_0,X_1\sim\pi_1}\left\|u_t^\theta(X_t) - (X_0 - X_1)\right\|^2. \end{aligned} \tag{4}$$

Crucially, this loss provides identical gradients to the intractable Flow Matching objective $\mathcal{L}_{\text{FM}}$ while being computationally efficient.

The simplicity of the CFM objective enables straightforward implementation, as the target velocity reduces to the constant vector $X_0 - X_1$ independent of time t and current position X_t. During inference, samples are generated by solving the learned ODE from $X_1 \sim \mathcal{N}(0, I)$ to $t = 0$.

3.2 DIFFUSION MODELS

A typical diffusion model Ho et al. (2020); Sohl-Dickstein et al. (2015); Song et al. (2020) usually involves a forward process and a backward process. In the forward process, we construct a forward diffusion transition kernel $q(x_t|x_{t-1})$ to transfer our data distribution into an easy-to-sample prior distribution, usually the standard Gaussian distribution. In continuous diffusion models, the transition kernel is usually Gaussian. With the reparameterization property, we can easily obtain the marginal distribution $q(x_t|x_0)$. Then with Bayes' theorem, we can obtain the ground-truth posterior distribution $q(x_{t-1}|x_t, x_0)$.

Since x_0 is usually available during the training phase, we typically use a parametric network $f_\theta(x_t, t)$ to predict x_0 (x_0-prediction) or predict noise (ϵ-prediction) to minimize the KL divergence between our parametric posterior $p_\theta(\cdot)$ and the ground-truth posterior distribution $q(x_{t-1}|x_t, x_0)$.

When the training phase is done, we successfully learn the backward transition. We first sample from the prior distribution and sample back to our target distribution with the learned posterior $p_\theta(x_{t-1}|x_t, t)$.

Song et al. (2020) unified these diffusion models into a forward and backward framework. The training target is to learn the score (i.e., the log-gradient of the data distribution). It's interesting to mention that the *score* actually has a quantitative equality to $-\epsilon/\sigma^2$ Vincent (2011); Song & Ermon (2019), which leads to the gradient equality to the epsilon prediction in vanilla DDPM diffusion.

4 Bernoulli Flow Models

Bernoulli Flow Models are a **binary** generative model family that learn to sample from a target Binary distribution by transforming samples from a simple base distribution (e.g., pure noise with Bernoulli parameter 0.5) via a learned continuous-time Bernoulli probability flow. We first give the high-level intuitive overview of the core differences between heuristic Bernoulli diffusion use and BFM in below Fig.1, then we present the detailed formulation of BFM.

Early Bernoulli diffusion models, including the Bernoulli diffusion in Sohl-Dickstein et al. (2015) and Binary Latent Diffusion (BLD) Wang et al. (2023), follow a common design pattern: one first specifies a one-step forward noise schedule $q(x_t \mid x_{t-1})$ and then, by analytic derivation, obtains a closed-form multi-step transition $q(x_t \mid x_0)$ whose probability path mimics that of a continuous-space diffusion process. Since their probability paths are constructed in this heuristic manner from continuous-space diffusion, we refer to this family of methods as *Heuristic Bernoulli Diffusion Models (HBDM)*.

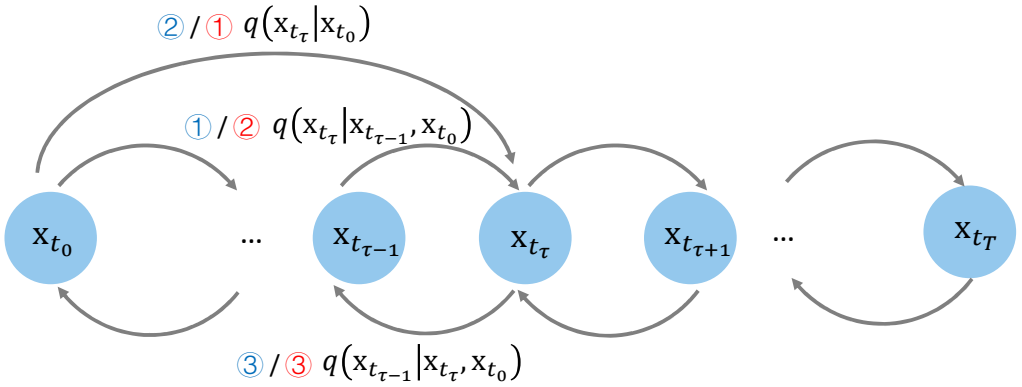


Figure 1: HBDM vs BFM. We illustrates the different design order choice in the forward process between HBDM and BFM.

As shown in Fig.1, HBDM first designs a forward diffusion process with a carefully designed transition kernel, then they obtain the marginal distribution and posterior distribution. While BFM first construct the optimal transport path between the target distribution and the prior distribution based on (2) to construct the marginal Bernoulli probability distribution, then we derive the one-step transition kernel and posterior distribution.

4.1 Notation and Problem Formulation

Consider an arbitrary binary distribution $p_{\mathbf{X}}(\mathbf{x})$ over the sample space $\mathcal{X} = \{0, 1\}^d$, where:

- $d \in \mathbb{N}^+$ denotes the dimensionality,
- $\mathbf{X} = (X^1, \ldots, X^d)$ is a discrete random vector taking values in $\mathcal{X}$,

- $\mathbf{x} = (x^1, \ldots, x^d) \in \{0, 1\}^d$ represents a specific realization of the random vector.
- $\mathcal{B}$ represents the Bernoulli distribution.

Given a target binary target dataset observations of $X_0 \sim \pi_0$ and a prior $X_1 \sim \pi_1$ where $\pi_1 = \mathcal{B}(0.5)$, the Optimal Transport path of Bernoulli probability flow can be constructed as follows:

$$X_{t_\tau} \sim \mathcal{B}(t_\tau X_0 + (1 - t_\tau)X_1), \quad t_\tau \in [1, 0], \tau = 0, 1, 2, ..., T \tag{5}$$

Which means $X_{t_0} \equiv X_0$, $X_{t_T} \equiv X_1$ and X_{t_τ} is the Bernoulli probability flow path from X_0 to X_1.

4.2 Bernoulli Probability Flow Dynamics

With the Bernoulli probability flow path defined above and the prior distribution π_1 being a Bernoulli distribution with a parameter of 0.5 (binary white noise), the marginal probability transition from target observations X_0 (or X_{t_0}) to X_{t_τ} can be expressed as:

$$\begin{aligned} q(X_{t_\tau}|X_{t_0}) &= \mathcal{B}(X_{t_\tau}; \alpha_{t_\tau}), \\ \alpha_{t_\tau} &= g(X_{t_0}, \frac{1 - t_\tau}{2}), \\ g(a, b) &= a(1 - b) + b(1 - a), \end{aligned} \tag{6}$$

And we can further obtain the one-step transition probability:

$$\begin{aligned} q(X_{t_T}|X_{t_{\tau-1}}) &= \mathcal{B}(X_{t_\tau}; g(X_{t_{\tau-1}}, \gamma_{t_\tau})), \\ \gamma_{t_\tau} &= \frac{0.5(t_{\tau-1} - t_\tau)}{t_{\tau-1}}, \end{aligned} \tag{7}$$

where γ_{t_τ} is the flip probability from $X_{t_{\tau-1}}$ to X_{t_τ}. We can further obtain the ground truth posterior distribution of X_{t_τ} given $X_{t_{\tau-1}}, X_{t_0}$:

$$q(X_{t_{\tau-1}}|X_{t_\tau}, X_{t_0}) = \mathcal{B}(X_{t_{\tau-1}}; \frac{g(X_{t_\tau}, \gamma_{t_\tau})\alpha_{t_{\tau-1}}}{g(g(X_{t_\tau}, \gamma_{t_\tau}), 1 - \alpha_{t_{\tau-1}})}) \tag{8}$$

with

$$q(X_{t_{\tau-1}}|X_{t_\tau}, X_{t_0}) = \frac{q(X_{t_\tau}|X_{t_{\tau-1}}, X_{t_0})q(X_{t_{\tau-1}}|X_{t_0})}{q(X_{t_\tau}|X_{t_{\tau-1}})q(X_{t_{\tau-1}}|X_{t_0}) + q(X_{t_\tau}|\neg X_{t_{\tau-1}})q(\neg X_{t_{\tau-1}}|X_{t_0})}. \tag{9}$$

Similar to DDPM Ho et al. (2020) X_0-predction, we parameterize our flow model f_θ to predict X_{t_0} directly to match the Bernoulli posterior probability path, this would allow us to sample back to our target distribution π_0 from the prior distribution π_1. The backward Bernoulli flow transition step can be expressed as:

$$p_\theta(X_{t_{\tau-1}}|X_{t_\tau}, X_{t_0}) = \mathcal{B}(X_{t_{\tau-1}}; \frac{g(X_{t_\tau}, \gamma_{t_\tau})g(f_\theta(X_{t_\tau}, t_\tau), \frac{1-t_{\tau-1}}{2})}{g(g(X_{t_\tau}, \gamma_{t_\tau}), 1 - g(f_\theta(X_{t_\tau}, t_\tau), \frac{1-t_{\tau-1}}{2}))}) \tag{10}$$

In order to predict X_{t_0} accurately, we use binary cross entropy to calculate the entropy loss and the loss function can be written as:

$$\begin{aligned} \mathcal{L}_{\text{BCE}}(\theta) := \mathbb{E}_{t_\tau, X_{t_0}, X_{t_\tau}} \big[& X_{t_0} \log \left(f_\theta(X_{t_\tau}, t_\tau) \right) \\ & + (1 - X_{t_0}) \log \left(1 - f_\theta(X_{t_\tau}, t_\tau) \right) \big] \end{aligned} \tag{11}$$

Proofs of this section can be found in Appendix A.

4.3 Model Training and Sampling

We now describe how BFM can be trained and sampled by minimal modification of existing binary diffusion Wang et al. (2023) training and sampling architectures, as summarized in Algorithms 1 and 2, where we highlight the differences between BFM and existing binary diffusion models in blue. The prediction of X_{t_0} is implemented via a neural network $f_\theta(X_{t_\tau}, t_\tau) = \sigma(\mathcal{T}_\theta(X_{t_\tau}, t_\tau)/\kappa)$, where $\sigma(\cdot)$ is the sigmoid function, $\mathcal{T}_\theta$ is a neural network with parameters θ, and κ is a temperature hyperparameter. The temperature is used to control the diversity of the generated samples. A smaller temperature leads to less diverse samples, while a larger temperature increases diversity.

Algorithm 1 Training procedure.

Given: Binary diffusion model f_θ parametrized by $\mathcal{T}_\theta$; An dataset $\mathbf{X}$.
Given: Diffusion steps T; Probability path defined by (5); Training steps I.
Initializing $\mathcal{T}_\theta$.
for Step $i = 1 : I$ **do**
    Sampling data $\mathbf{x}_{t_0} \sim \mathbf{X}$, and time step $\tau \sim \{1, \dots, T\}$.
    Obtaining $\mathbf{x}_{t_\tau}$ using $\mathbf{x}_{t_0}$, t_τ , and probability path with (5).
    Predicting the probability that the state is $\mathbf{x}_{t_0}$ using $f_\theta(\mathbf{x}_{t_\tau}, t_\tau) = \sigma(\mathcal{T}_\theta(\mathbf{x}_{t_\tau}, t_\tau)/\kappa)$.
    Calculating loss $\mathcal{L}$ using (11).
    Backpropagating $\mathcal{L}$ and updating θ.
end for
Return Binary diffusion model f_θ.

Algorithm 2 Sampling procedure.

Given: Trained binary diffusion model f_θ;The sample shape dimension $shape$.
Given: Diffusion steps T; Probability path defined by (5); Temperature κ.
Sampling $\mathbf{x}_{t_T} = \text{Bernoulli}(\mathbf{x}^{\text{init}})$, where $\mathbf{x}^{\text{init}} \in \mathbb{R}^{shape}$ and contains 0.5 only.
for Step $\tau = T : 1$ **do**
    Predicting $p_\theta(\mathbf{x}_{t_{\tau-1}})$ with $f_\theta(\mathbf{x}_{t_\tau}, t_\tau) = \sigma(\mathcal{T}_\theta(\mathbf{x}_{t_\tau}, t_\tau)/\kappa)$ and (10).
    Sampling $\mathbf{x}_{t_{\tau-1}} = \text{Bernoulli}(p_\theta(\mathbf{x}_{t_{\tau-1}}))$
end for
$\hat{\mathbf{x}}_{t_0} = \mathbf{x}_{t_{\tau-1}}$
Return the final sample $\hat{\mathbf{x}}_{t_0}$.

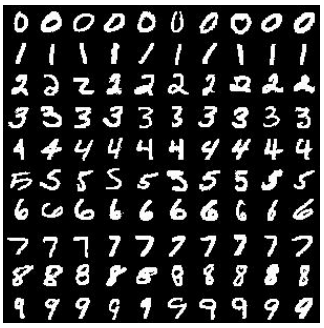
5 Experiments

We empirically validate the proposed BFM on diverse binary generative modeling tasks. In order to demonstrate the binary generative capability of BFM, we first conduct experiments on binarized MNIST, CIFAR10, and FFHQ 64x64 datasets. Next, we connect BFM to statistical physics by applying it to the Ising model, a canonical probabilistic model defined over a set of interacting spin variables, and we visualize the magnetization of the generated samples and compare it with the standard Wolff cluster algorithm Wolff (1989) samples. Finally, we demonstrate that our BFM can also supportted as a binary-representation latent space generative model for high-dimensional image generation tasks, and achieve competitive performance compared to current state-of-the-art continuous and discrete generative models, we report the Frechet Inception Distance (FID) Heusel et al. (2017) scores and Precision and Recall (Prec.&Rrec.) Sajjadi et al. (2018) metrics on the LSUN Bedroom and Church datasets and FFHQ dataset. We conduct comparisions by generating 50K samples and comparing them with the corresponding training datasets in every experiment.

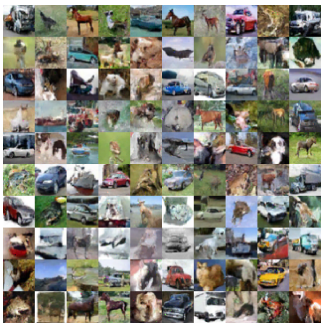
5.1 Model and Training Setup

To ensure a rigorous and fair comparison with BLD Wang et al. (2023), we implement BFM directly on top of the official BLD codebase. For the Ising model, binarized MNIST, and the 256×256 datasets (LSUN Bedrooms, LSUN Churches, and FFHQ), we fully adhere to the experimental settings of BLD, employing the identical full transformer backbone and hyperparameters. The only exceptions are the CIFAR-10 and FFHQ 64×64 datasets, where we employ the U-Net architecture from guided-diffusion Dhariwal & Nichol (2021). For these two datasets, apart from the specific U-Net backbone, all other hyperparameters remain identical to the BLD baseline.

All experiments are conducted on two NVIDIA GeForce RTX 4090 GPUs. We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 1 \times 10^{-8}$, coupled with a warm-up scheduler (peak learning rate of 1×10^{-4} after 10K iterations). The total training iterations are set to 800K for the 256×256 datasets, 60K for CIFAR-10 and FFHQ 64×64, and 100K for binarized MNIST.



(a) Conditional MNIST



(b) CIFAR10

(c) FFHQ 64x64

Figure 2: **Samples from BFM.** Fig. 2a shows conditional binarized MNIST samples with conditional BFM. Figs. 2b and 2c shows unconditional samples from binarized CIFAR10 and FFHQ datasets, respectively.

5.2 Toy Binarized Image Generation

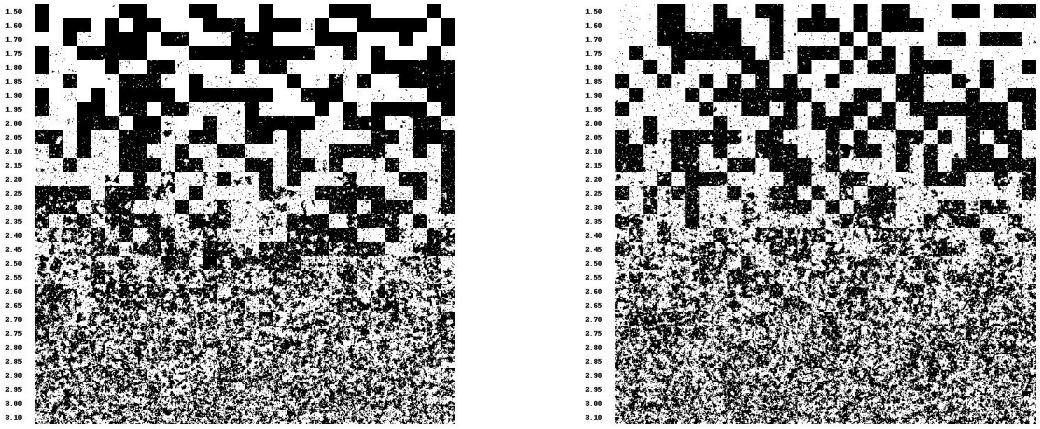
To validate the generative performance of BFM on high-dimensional binary manifolds, we first conduct experiments on binarized MNIST, CIFAR10, and FFHQ 64x64 datasets. For the binarized MNIST dataset, we apply a thresholding scheme where pixels greater than 0 are set to 1, and others are set to 0. For CIFAR-10 and FFHQ 64×64, we treat each pixel's color channel intensity (an 8-bit unsigned integer) as a separate scalar value. We then binarize it by converting the value into its fixed-length 8-bit binary expansion. This process effectively unpacks the bit depth of each channel, replacing a single intensity value with 8 binary bits. Thus, each pixel is transformed from a 3-channel tuple into a flattened binary vector of length 24 (3 channels × 8 bits). We visualize the generated samples in Fig. 2. We evaluate the conditional generative performance of BFM on the binarized MNIST dataset and report the FID scores in Table 1, comparing it with other state-of-the-art methods. As shown, BFM achieves the second-best FID score of 0.42. The FID metric used to evaluate our generated binarized MNIST samples follows the same codebase as the one used in the SFM Cheng et al. (2024) GitHub repository. For binarized CIFAR10 and FFHQ 64x64 datasets, we achieve FID scores of 72.15 and 35.72, respectively.

Model	Base method	FID
Blackout Diffusion Santos et al. (2023)	Diffusion	**0.02**
SFM Cheng et al. (2024)	Flow Matching	4.62
α-Flow Cheng et al. (2025)	Flow Matching	5.02
DMPM Pham et al. (2025)	Score-based Diffusion	2.89
CoVAE Silvestri & Ambrogioni (2025)	VAE	0.58
BFM (ours)	Flow Matching & Diffusion	<u>0.42</u>

Table 1: Comparison of FID scores for various methods on binarized MNIST. The best and second-best results are highlighted in **bold** and <u>underline</u>, respectively.

5.3 Connecting to Statistical Physics: Ising Model Generation

We evaluate our BFM algorithm on the Ising model, a canonical probabilistic model in statistical physics defined over a set of D interacting spin variables $\mathbf{s} \in \{+1, -1\}^D$ (Ising, 1925). In order to obtain the simulate Ising samples at different temperatures, we use the Wolff cluster algorithm Wolff (1989) to obtain the Ising samples at different temperatures. We conisder a 20x20 2D lattice Ising model with periodic boundary conditions, where each spin interacts with its four nearest neighbors. The Markov chain is run for 1000 steps to reach thermal equilibrium, we sample 500 samples for as training set and 100 samples for validation set at each temperature from 1.50 to 3.10. We train conditional BFM for 100K iterations with a batch size of 64 using above simulated training set. During inference, we use 64 sampling steps to generate Ising samples from pure noise. Fig. 3 compares the samples generated by BFM and the validate samples from above simulate results at



(a) BFM Ising Samples (b) Wolff cluster Ising Samples

Figure 3: Comparison of Ising samples generated by BFM and ground truth across temperatures (1.50 to 3.10).

different temperatures. We can see that BFM can generate high-quality Ising samples that are visually indistinguishable from the ground truth samples across a range of temperatures. We show the magnetization of the generated samples compared to the samples from the Wolff cluster algorithm in different temperatures in Appendix B.1.

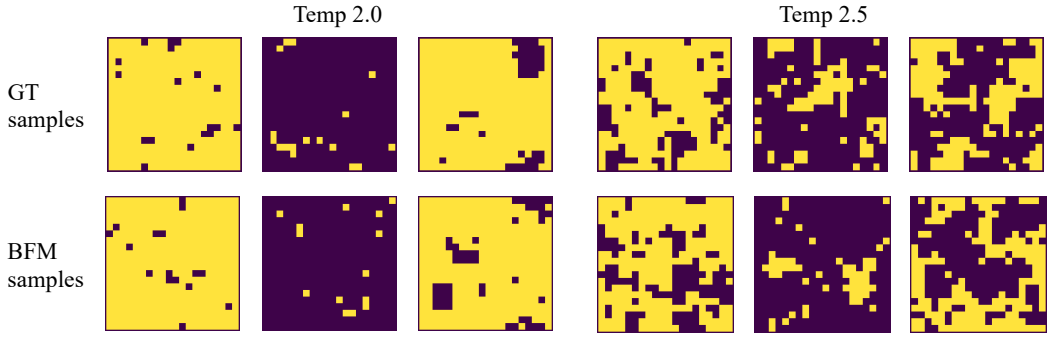


Figure 4: Comparison of Ising 20x20 lattice samples generated by BFM and ground truth at temperatures 2.0 and 2.5.

5.4 High-Dimensional Image Generation With Latent BFM

To further validate the generative performance of BFM, we employ it as a latent space generator for high-dimensional image synthesis. We leverage the pre-trained encoder and decoder from BLD Wang et al. (2023) to map images from the LSUN (Bedroom and Churches) and FFHQ datasets into a binary latent space of $16 \times 16 \times 64$. Detailed implementation specifics of the autoencoder can be found in Wang et al. (2023).

Ensuring a fair comparison with the BLD baseline presented a challenge due to the absence of original training scripts and hardware differences. To address this, we provide results from both the original paper and our own reproduction. **Crucially, both BFM and the reproduced BLD strictly follow the training configurations of the original BLD**, sharing the identical backbone, learning rate, loss function, and iterations. As shown in Table 2, BFM demonstrates competitive performance against established discrete baselines (e.g., D3PM, VQ-Diffusion) while requiring significantly fewer sampling steps (64 vs. 200). On the challenging FFHQ dataset, BFM achieves a Recall of 0.49, rivaling state-of-the-art methods in diversity. While a numerical gap exists compared to the original BLD report due to hardware constraints, **BFM consistently outperforms the**

Figure 5: Samples from BFM on LSUN-Bedrooms, LSUN-Churches and FFHq datasets. All samples resolution are 256x256.

Table 2: Comparison of various methods for generating images of LSUN Bedrooms and Churches and FFHQ datasets. All images are of resolution 256x256.

Methods	Steps	LSUN-Bedrooms 256x256			LSUN-Churches 256x256			FFHQ 256x256		
		FID ↓	Prec. ↑	Recall ↑	FID ↓	Prec. ↑	Recall ↑	FID ↓	Prec. ↑	Recall ↑
Continuous Generative Models										
PGGAN Karras et al. (2017)	-	8.34	0.48	0.40	6.42	0.65	0.39	-	-	-
StyleGAN Karras et al. (2019; 2020)	-	**2.35**	0.59	**0.48**	**3.86**	0.60	0.43	**4.16**	0.71	0.46
LDM-4/8/4 Rombach et al. (2022)	200	2.95	0.66	**0.48**	4.02	0.64	0.52	4.98	**0.73**	**0.50**
Patch-DM Ding et al. (2024)	50	6.04	0.56	0.44	5.49	0.62	**0.53**	10.02	0.68	0.44
VQ-LCMD Nguyen et al. (2024)	200	4.16	**0.72**	0.40	4.99	**0.75**	0.42	7.25	0.72	0.46
Discrete Generative Models										
D3PM Austin et al. (2021)	200	6.60	0.60	0.35	6.02	0.68	0.39	9.49	0.71	0.41
VQ-Diffusion Gu et al. (2022)	200	7.19	0.54	0.37	6.88	**0.72**	0.37	8.79	0.70	0.43
MaskGIT Chang et al. (2022)	200	8.39	**0.67**	0.38	**4.07**	0.71	0.45	11.45	**0.75**	0.44
BLD Wang et al. (2023)	64	**3.85**	0.65	**0.44**	4.36	0.68	**0.50**	**5.85**	0.73	**0.50**
BLD (Reproduced)	64	6.22	**0.69**	0.36	5.48	**0.66**	**0.41**	11.46	**0.68**	0.48
BFM (ours)	64	**5.86**	**0.69**	**0.37**	**5.32**	**0.66**	**0.41**	**10.87**	**0.68**	**0.49**

reproduced BLD baseline under the strictly unified setting across all datasets (e.g., FID 5.86 vs. 6.22 on Bedrooms, 10.87 vs. 11.46 on FFHQ). This confirms that under identical computational budgets, BFM provides a superior trade-off between generation quality and efficiency. The visualized samples shown in Fig. 5 further validate BFM's ability to generate diverse and realistic high-dimensional images.

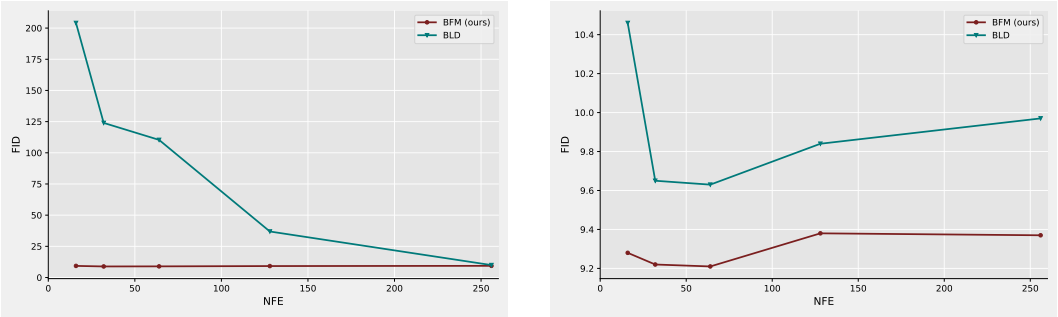
5.5 Sampling Efficiency Analysis

In order to further demonstrate the potential advantage of our proposed BFM in terms of inference speed and quality. We conduct experiments on LSUN churches 256×256 dataset with different NFE. We compare our BFM with BLD in terms of FID versus the NFE from two perspectives: **Fixed training sampling steps.** We use checkpoints trained with 256 sampling steps and vary the NFE at inference, reporting the corresponding FID. **Varying training sampling steps.** We train separate models with 16, 32, 64, 128, and 256 sampling steps and, for each model, evaluate FID across its own range of NFE.

We use the same training settings as described in Sec. 5.1, except that we set the number of training iterations to 50K for all models to ensure a fair comparison.

In Fig. 6a, notice that when we sampling with different NFE in the same 256 checkpoint, BFM consistently outperforms BLD by a large margin. BLD performance poor at low NFE, whereas BFM demonstrates strong self-consistency and stable performance across different NFE. Upon investigating the underlying causes, we identified a **theoretical discrepancy** in the **cross-step** posterior sampling process of the official BLD implementation. This issue hinders the model from accurately sampling the correct posterior transition. Here we give a detail discussion in Appendix C.1.

In Fig. 6b, we can see that when we train separate models with different sampling steps, BFM consistently outperforms BLD across all NFE. This further demonstrates the advantage of BFM in



(a) Fixed training sampling steps (256).

(b) Varying training sampling steps.

Figure 6: FID versus NFE on LSUN Churches 256×256 for BFM and BLD under two evaluation settings: (a) using checkpoints trained with 256 sampling steps and varying the NFE at inference; (b) using checkpoints trained with 16, 32, 64, 128, and 256 sampling steps and, for each, reporting FID across its own NFE schedule.

terms of inference speed and quality. The discussion on probability path design for this comparision can be found in Appendix C.2.

6 Discussion

The proposed BFM presents a novel and versatile framework for generative modeling of binary data, with significant potential for expansion into temporal and cross-disciplinary applications. By constructing a continuous probability flow between arbitrary Bernoulli distributions and a simple noise prior, BFM eliminates the need for problem-specific codecs while maintaining high expressivity. Notably, BFM exhibits high stability and self-consistency cross-step sampling; experiments confirm its ability to generate high-quality samples even with significantly low NFE. Above advantages positions BFM as a promising candidate for modeling complex time-dependent binary phenomena in social thermodynamics—such as election outcomes, financial market movements, and migration patterns—where traditional Ising-like models have been applied but often lack efficient sampling and training mechanisms. Furthermore, the model's ability to capture binary state dynamics suggests immediate applicability in biological systems, including neural spiking activity and gene expression switching, where binary states are inherent. Future work will focus on integrating temporal dependencies through recurrent or attention-based mechanisms and exploring large-scale real-world applications in biological systems. Interdisciplinary collaboration will be essential to address domain-specific challenges such as non-stationarity, sparse data, and interpretability requirements.

7 Reproducibility Statement

We provide detailed descriptions of our model architecture, training procedures, and hyperparameter settings in the main text and Appendix. To ensure the reproducibility of our results, we will release the complete codebase soon.

REFERENCES

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint*, 2024. URL https://arxiv.org/abs/2402.04997.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. *Advances in Neural Information Processing Systems*, 37:54787–54819, 2024.

Chaoran Cheng, Jiahan Li, Jiajun Fan, and Ge Liu. α-Flow: A Unified Framework for Continuous-State Discrete Flow Matching Models. *arXiv preprint arXiv:2504.10283*, 2025.

Oscar Davis, Samuel Kessler, Mircea Petrache, Ismail Ceylan, Michael Bronstein, and Joey Bose. Fisher flow matching for generative modeling over discrete data. *Advances in Neural Information Processing Systems*, 37:139054–139084, 2024.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Zheng Ding, Mengqi Zhang, Jiajun Wu Wu, and Zhuowen Tu. Patched denoising diffusion models for high-resolution image synthesis. 2024.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10696–10706, 2022.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

Ernst Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258, 1925.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

Volodymyr Kinakh and Slavi Voloshynovskiy. Binary diffusion probabilistic model. *arXiv preprint*, 2025. URL https://arxiv.org/abs/2501.13915. Under review.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

Bac Nguyen, Chieh-Hsin Lai, Yuhta Takida, Naoki Murata, Toshimitsu Uesaka, Stefano Ermon, and Yuki Mitsufuji. Improving vector-quantized image modeling with latent consistency-matching diffusion. *arXiv preprint arXiv:2410.14758*, 2024.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. *arXiv preprint arXiv:2406.01572*, 2024.

Le-Tuyet-Nhi Pham, Dario Shariatian, Antonio Ocello, Giovanni Conforti, and Alain Durmus. Discrete markov probabilistic models. *arXiv preprint arXiv:2502.07939*, 2025.

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*, pp. 10684–10695, 2022.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in neural information processing systems*, 31, 2018.

Javier E Santos, Zachary R Fox, Nicholas Lubbers, and Yen Ting Lin. Blackout diffusion: generative diffusion models in discrete-state spaces. In *International Conference on Machine Learning*, pp. 9034–9059. PMLR, 2023.

Gianluigi Silvestri and Luca Ambrogioni. Covae: Consistency training of variational autoencoders. *arXiv preprint arXiv:2507.09103*, 2025.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Harshit Varma, Dheeraj Nagaraj, and Karthikeyan Shanmugam. Glauber generative model: Discrete diffusion models via binary classification. *arXiv preprint arXiv:2405.17035*, 2024.

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

Ze Wang, Jiang Wang, Zicheng Liu, and Qiang Qiu. Binary latent diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22576–22585, 2023.

Ulli Wolff. Collective monte carlo updating for spin systems. *Physical Review Letters*, 62(4):361, 1989.

Julia Wolleb, Florentin Bieder, Paul Friedrich, Peter Zhang, Alicia Durrer, and Philippe C Cattin. Binary noise for binary tasks: Masked bernoulli diffusion for unsupervised anomaly detection. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 135–145. Springer, 2024.

A PROOFS OF BERNOULLI PROBABILITY FLOW DYNAMICS

We give the detailed proofs of Bernoulli Probability Flow Dynamics in this section.

Proof of Marginal Distribution from X_{t_0} to X_{t_τ}. Let's first prove the marginal distribution from X_{t_0} to X_{t_τ} in equation 6:

Let $X_0 \in \{0, 1\}$ be a random variable from some data distribution and $X_1 \sim \text{Bernoulli}(0.5)$ be a random variable sampled from a Bernoulli distribution with parameter 0.5, where all elements of X_0 and X_1 are either 0 or 1. For a sequence of parameters $t_\tau \in [1, 0]$ with $\tau = 0, 1, 2, \ldots, T$ (note that t_τ decreases from 1 to 0 as τ increases), we define the random variable X_{t_τ} conditionally on X_0 and X_1 as:

$$X_{t_\tau} \sim \text{Bernoulli}\left(t_\tau X_0 + (1 - t_\tau)X_1\right).$$

This definition implies that given X_0 and X_1, the value of X_{t_τ} is sampled from a Bernoulli distribution with parameter $p = t_\tau X_0 + (1 - t_\tau)X_1$. Since X_0 and X_1 are binary, p is always a valid probability between 0 and 1.

We aim to derive the conditional distribution $q(X_{t_\tau} \mid X_0)$, which requires marginalizing over X_1. Given that $X_1 \sim \text{Bernoulli}(0.5)$, we compute $\mathbb{P}(X_{t_\tau} = 1 \mid X_0)$ for each value of X_0.

Case 1: $X_0 = 0$

$$\begin{aligned}
\mathbb{P}(X_{t_\tau} = 1 \mid X_0 = 0) &= \mathbb{E}_{X_1}\left[\mathbb{P}(X_{t_\tau} = 1 \mid X_0 = 0, X_1)\right] \\
&= \mathbb{E}_{X_1}\left[t_\tau \cdot 0 + (1 - t_\tau)X_1\right] \\
&= (1 - t_\tau) \cdot \mathbb{E}[X_1] \\
&= (1 - t_\tau) \cdot 0.5 = 0.5(1 - t_\tau).
\end{aligned}$$

Thus, $q(X_{t_\tau} \mid X_0 = 0) = \text{Bernoulli}(0.5(1 - t_\tau))$.

Case 2: $X_0 = 1$

$$\begin{aligned}
\mathbb{P}(X_{t_\tau} = 1 \mid X_0 = 1) &= \mathbb{E}_{X_1}\left[\mathbb{P}(X_{t_\tau} = 1 \mid X_0 = 1, X_1)\right] \\
&= \mathbb{E}_{X_1}\left[t_\tau \cdot 1 + (1 - t_\tau)X_1\right] \\
&= t_\tau + (1 - t_\tau) \cdot \mathbb{E}[X_1] \\
&= t_\tau + (1 - t_\tau) \cdot 0.5 = 0.5(1 + t_\tau).
\end{aligned}$$

Thus, $q(X_{t_\tau} \mid X_0 = 1) = \text{Bernoulli}(0.5(1 + t_\tau))$.

Combining both cases, we express the probability uniformly:

$$\mathbb{P}(X_{t_\tau} = 1 \mid X_0) = \frac{1 + (2X_0 - 1)t_\tau}{2}.$$

Therefore, the transition probability is:

$$q(X_{t_\tau} \mid X_0) = \text{Bernoulli}\left(\frac{1 + (2X_0 - 1)t_\tau}{2}\right),$$

which can also be expressed as:

$$q(X_{t_\tau} \mid X_0) = \text{Bernoulli}\left(g(X_{t_0}, \frac{1 - t_\tau}{2})\right),$$

where $g(a, b) = a(1 - b) + b(1 - a)$. This completes the proof of the marginal distribution from X_{t_0} to X_{t_τ}.

Proof of probability transition from $X_{t_{\tau-1}}$ to X_{t_τ} in equation 7.

We define the binary state variable $X_{t_\tau} \in \{0, 1\}$ at time τ, and assume a Markovian transition process from $X_{t_{\tau-1}}$ to X_{t_τ} with an error rate γ_{t_τ}. The conditional transition probability is given by a Bernoulli distribution:

$$q(X_{t_\tau} | X_{t_{\tau-1}}, X_{t_0}) = \mathcal{B}\left(X_{t_\tau}; X_{t_{\tau-1}}(1 - \gamma_{t_\tau}) + (1 - X_{t_{\tau-1}})\gamma_{t_\tau}\right),$$

where $\mathcal{B}(x; p)$ denotes the Bernoulli probability mass function with parameter p.

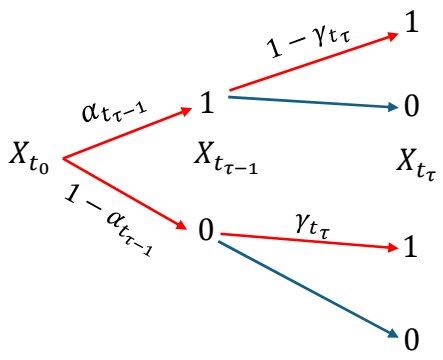


Figure 7: Bernoulli probability transition path illustration.

Let α_{t_τ} be the marginal probability of $X_{t_\tau} = 1$ given X_{t_0}:

$$\alpha_{t_\tau} = \frac{1 + (2X_{t_0} - 1)t_\tau}{2}, \tag{12}$$

where T_{t_τ} is a function defining the schedule of the process. Similarly,

$$\alpha_{t_{\tau-1}} = \frac{1 + (2X_{t_0} - 1)t_{\tau-1}}{2}. \tag{13}$$

We now consider all paths that lead to $X_{t_\tau} = 1$. The marginal probability α_{t_τ} can be expressed via the law of total probability (see Fig. 7 for an illustration):

$$\alpha_{t_\tau} = q(X_{t_\tau} = 1|X_{t_0}) = \sum_{x\in\{0,1\}} q(X_{t_\tau} = 1|X_{t_{\tau-1}} = x, X_{t_0}) \cdot q(X_{t_{\tau-1}} = x|X_{t_0}). \tag{14}$$

Substituting the transition probabilities and marginal probabilities:

$$\alpha_{t_\tau} = \left[\alpha_{t_{\tau-1}}(1 - \gamma_{t_\tau}) + (1 - \alpha_{t_{\tau-1}})\gamma_{t_\tau}\right]. \tag{15}$$

Rearranging terms:

$$\alpha_{t_\tau} = \alpha_{t_{\tau-1}} - 2\gamma_{t_\tau}\alpha_{t_{\tau-1}} + \gamma_{t_\tau}. \tag{16}$$

Solving for γ_{t_τ}:

$$\gamma_{t_\tau} = \frac{\alpha_{t_{\tau-1}} - \alpha_{t_\tau}}{2\alpha_{t_{\tau-1}} - 1}. \tag{17}$$

Thus, the transition probability is fully specified as:

$$q(X_{t_\tau}|X_{t_{\tau-1}}, X_{t_0}) = \mathcal{B}\left(X_{t_\tau}; X_{t_{\tau-1}}(1 - \gamma_{t_\tau}) + (1 - X_{t_{\tau-1}})\gamma_{t_\tau}\right), \tag{18}$$

with

$$\gamma_{t_\tau} = \frac{\alpha_{t_{\tau-1}} - \alpha_{t_\tau}}{2\alpha_{t_{\tau-1}} - 1}.$$

Substituting α_{t_τ} and $\alpha_{t_{\tau-1}}$ in equation 12 and in equation 13, we have:

$$\gamma_{t_\tau} = \frac{0.5(t_{\tau-1} - t_\tau)}{t_{\tau-1}}, \tag{19}$$

Proof of posterior probability in equation 8. We aim to derive the posterior distribution $q(X_{t_{\tau-1}}|X_{t_\tau}, X_{t_0})$ using Bayes' theorem. Starting from the conditional Bayes' rule in binary variables case in equation 9.

Remember that we wanna find the Bernoulli parameter, which means we want to find the probability of $X_{t_{\tau-1}} = 1$ given X_{t_τ} and X_{t_0}:

$$q(X_{t_{\tau-1}} = 1 \mid X_{t_\tau}, X_{t_0}) \tag{20}$$

$$= \frac{q(X_{t_\tau} \mid X_{t_{\tau-1}} = 1, X_{t_0}) \cdot q(X_{t_{\tau-1}} = 1 \mid X_{t_0})}{q(X_{t_\tau} \mid X_{t_{\tau-1}} = 1) \cdot q(X_{t_{\tau-1}} = 1 \mid X_{t_0}) + q(X_{t_\tau} \mid X_{t_{\tau-1}} = 0) \cdot q(X_{t_{\tau-1}} = 0 \mid X_{t_0})}. \tag{21}$$

We now define the following transition probabilities using the noise schedule parameters γ_{t_τ} and α_{t_τ}:

$$q(X_{t_\tau} \mid X_{t_{\tau-1}} = 1) = X_{t_\tau}(1 - \gamma_{t_\tau}) + (1 - X_{t_\tau})\gamma_{t_\tau}, \tag{22}$$

$$q(X_{t_\tau} \mid X_{t_{\tau-1}} = 0) = 1 - (X_{t_\tau}(1 - \gamma_{t_\tau}) + (1 - X_{t_\tau})\gamma_{t_\tau}), \tag{23}$$

$$q(X_{t_{\tau-1}} = 1 \mid X_{t_0}) = \frac{1 + (2X_{t_0} - 1)t_{\tau-1}}{2} = \alpha_{t_\tau}, \tag{24}$$

$$q(X_{t_{\tau-1}} = 0 \mid X_{t_0}) = 1 - \frac{1 + (2X_{t_0} - 1)t_{\tau-1}}{2} = 1 - \alpha_{t_{\tau-1}}. \tag{25}$$

Substituting these into Equation equation 21, we obtain:

$$q(X_{t_{\tau-1}} = 1 \mid X_{t_\tau}, X_{t_0}) \tag{26}$$

$$= \frac{(X_{t_\tau}(1 - \gamma_{t_\tau}) + (1 - X_{t_\tau})\gamma_{t_\tau}) \cdot (\alpha_{t_{\tau-1}})}{(X_{t_\tau}(1 - \gamma_{t_\tau}) + (1 - X_{t_\tau})\gamma_{t_\tau}) \cdot \alpha_{t_{\tau-1}} + (1 - (X_{t_\tau}(1 - \gamma_{t_\tau}) + (1 - X_{t_\tau})\gamma_{t_\tau})) \cdot (1 - \alpha_{t_{\tau-1}})}. \tag{27}$$

Simplifying the denominator and numerator with $g(a, b) = a(1 - b) + b(1 - a)$, we arrive at the final expression:

$$q(X_{t_{\tau-1}} = 1 \mid X_{t_\tau}, X_{t_0}) = \frac{g(X_{t_\tau}, \gamma_{t_\tau})\alpha_{t_{\tau-1}}}{g(g(X_{t_\tau}, \gamma_{t_\tau}), 1 - \alpha_{t_{\tau-1}})} \tag{28}$$

This concludes the derivation of the posterior distribution for the binary state transition process.

B Additional Qualitative Results

B.1 Ising Models

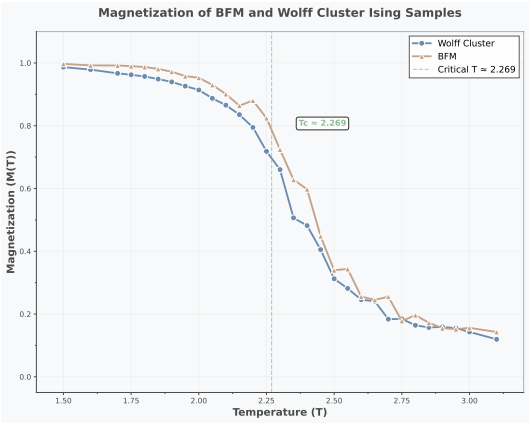


Figure 8: Magnetization curve of BFM generated and Wolff cluster Ising samples.

Fig. 8 illustrates the magnetization curve of the Ising model samples generated by our BFM. We observe that BFM is highly effective at generating physical states conditioned on temperature. The resulting magnetization curve aligns well with the ground truth simulations produced by the traditional Wolff cluster algorithm, capturing the physical properties with high fidelity.

B.2 Visualization of Different Temperatures of BFM Generated Samples

Figure 9: Additional unconditional image generation results and comparisons at 256 × 256 with the LSUN bedrooms dataset. The sampling temperatures are linearly interpolation between 0.5 and 1.0 from left to right.

Figure 10: Additional unconditional image generation results and comparisons at 256 × 256 with the LSUN churches dataset. The sampling temperatures are linearly interpolation between 0.5 and 1.0 from left to right.

Figure 11: Additional unconditional image generation results and comparisons at 256 × 256 with the FFHQ dataset. The sampling temperatures are linearly interpolation between 0.5 and 1.0 from left to right.

B.3 Algorithm for Latent Space BFM

The flipping part in below algorithms is the residual prediction, which means rather than predicting the clean data X_{t_0}, we predict the flipping part from X_{t_τ} to X_{t_0} as $f_\theta(X_{t_\tau}, t_\tau)$. More deetails please refer to Wang et al. (2023).

Algorithm 3 Training procedure.

Given: Trained encoder Ψ; Binary diffusion model f_θ parametrized by $\mathcal{T}_\theta$; An image dataset $\mathbf{Z}$.
Given: Diffusion steps T; Probability path defined by (5); Training steps I.
Initializing $\mathcal{T}_\theta$.
for Step $i = 1 : I$ **do**
  Sampling image $\mathbf{z} \sim \mathbf{Z}$, and time step $\tau \sim \{1, \dots, T\}$.
  Obtaining binary code $\mathbf{x}_{t_0} = \text{Bernoulli}(\sigma(\Psi(\mathbf{z})))$.
  Obtaining $\mathbf{x}_{t_\tau}$ using $\mathbf{x}_{t_0}$, t_τ, and probability path with (5).
  Predicting flipping probability $f_\theta(\mathbf{x}_{t_\tau}, t_\tau)$.
  Obtaining predicted $\mathbf{x}_{t_0}$ as $p_\theta(\mathbf{x}_{t_0}) = (1 - \mathbf{x}_{t_\tau}) \odot f_\theta(\mathbf{x}_{t_\tau}, t) + \mathbf{x}_{t_\tau} \odot (1 - f_\theta(\mathbf{x}_{t_\tau}, t))$.
  Calculating loss $\mathcal{L}$ using (11).
  Backpropagating $\mathcal{L}$ and updating θ.
end for
Return Binary diffusion model f_θ.

Algorithm 4 Sampling procedure.

Given: Trained decoder Φ; Trained binary diffusion model f_θ;Latent dimension specified by h', w', c.
Given: Diffusion steps T; Probability path defined by (5); Temperature κ.
Sampling $\mathbf{x}_{t_T} = \text{Bernoulli}(\mathbf{x}^{\text{init}})$, where $\mathbf{x}^{\text{init}} \in \mathbb{R}^{h' \times w' \times c}$ and contains 0.5 only.
for Step $t = T : 1$ **do**
  Predicting $p_\theta(\mathbf{x}_{t_{\tau-1}})$ with $f_\theta(\mathbf{x}_{t_\tau}, t_\tau) = \sigma(\mathcal{T}_\theta(\mathbf{x}_{t_\tau}, t_\tau)/\kappa)$ and (10).
  Sampling $\mathbf{x}_{t_{\tau-1}} = \text{Bernoulli}(p_\theta(\mathbf{x}_{t_{\tau-1}}))$
end for
Return the sampled image as $\Phi(\mathbf{x}_{t_{\tau-1}})$.

B.4 The Use of Large Language Models

We utilized a large language model to assist in correcting grammatical errors and improving sentence expression in this paper. We acknowledge the large language model for its contribution to this paper.

C Additional Experimental Results

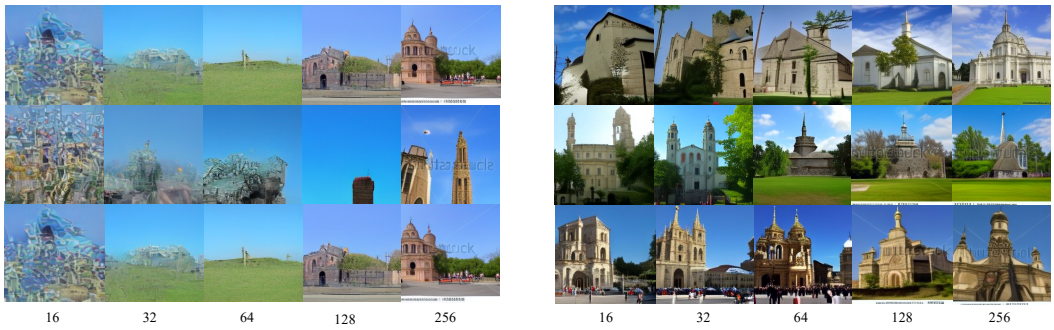
C.1 Discussion on Cross-step Sampling Consistency of BFM

In this section, we discuss the reason why BFM exhibits better sampling quality than BLD when using different NFE from the same trained checkpoint. This property is particularly important in practical applications, as it allows for flexible adjustment of sampling speed and quality trade-offs without the need for retraining the model. We follow the notation used in BLD to explain this issue. In the sampling scenario of Fig. 6a, when we sample with K NFE from a model trained with N steps, we need to perform cross-step sampling with stride $m = N/K$. According to Eq. (9) in BLD, the posterior transition can be written as

$$q(\mathbf{z}^{t-m}|\mathbf{z}^t, \mathbf{z}^0) = \frac{q(\mathbf{z}^t|\mathbf{z}^{t-m}, \mathbf{z}^0)q(\mathbf{z}^{t-m}|\mathbf{z}^0)}{q(\mathbf{z}^t|\mathbf{z}^0)}. \tag{29}$$

In the official BLD code implementation, the evidence transition term is implemented as

$$q(\mathbf{z}^t|\mathbf{z}^{t-m}, \mathbf{z}^0) = \mathcal{B}(\mathbf{z}^t; \mathbf{z}^{t-m}(1 - \beta^{t-m}) + 0.5\beta^{t-m}), \tag{30}$$



(a) BLD (b) BFM

Figure 12: Visual comparison of samples generated with NFE from the respective checkpoints of BLD and BFM, each trained separately with 256 sampling steps on LSUN Churches 256×256. (a) BLD and (b) BFM. For each method, each row corresponds to a different NFE setting, while images in the same column share the same initial latent code.

which effectively uses only the last-step noise parameter β^{t-m} for this cross-step update, however, the corresponding exact evidence transition should be

$$q(\mathbf{z}^t \mid \mathbf{z}^{t-m}) = \mathcal{B}\left(\mathbf{z}^t; \left(\prod_{j=t-m+1}^{t}(1-\beta^j)\right)\mathbf{z}^{t-m} + 0.5\left(1 - \prod_{j=t-m+1}^{t}(1-\beta^j)\right)\right) \tag{31}$$

In contrast, in the framework of BFM, the cross-step posterior sampling is naturally supported by our derived posterior transition in equation 7 and equation 8, which can be directly applied to any arbitrary cross-step posterior transition with slight modification. The only modification is to reparameterize the linear interpolation of t_τ on $[1, 0.5]$ from the original N training steps to a new number of NFE K at sampling time. We refer to this property as the **self-consistency of BFM under cross-step sampling**.

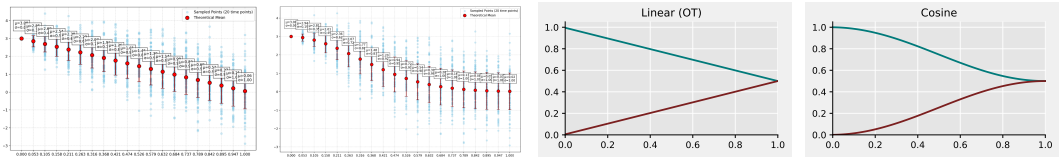
To understand why BFM has this self-consistency property, we refer the reader to Fig. 7. The evidence transition $q(X_{t_\tau}|X_{t_{\tau-1}}, X_{t_0})$ from $X_{t_{\tau-1}}$ to X_{t_τ} is depending on γ_{t_τ}, we use the marginal distribution from X_{t_0} to X_{t_τ} and $X_{t_{\tau-1}}$ to derive γ_{t_τ} and found it only depends on t_τ and $t_{\tau-1}$ as equation 19 shows. Therefore, when we perform cross-step sampling with different NFE, we can simply easily re-calculate γ_{t_τ} based on the new t_τ and $t_{\tau-1}$ without any approximation, which ensures the correctness of the posterior transition. This is the key mechanism of BFM's self-consistency property in cross-step sampling. The self-consistency property makes BFM more flexible and robust in cross-step sampling scenarios.

We further present visual results for different NFE sampled from the respective checkpoints of BLD and BFM, both trained with 256 sampling steps, as shown in Fig. 12. We can observe that, due to the accumulation of errors in the posterior probability transition, the samples generated by BLD become increasingly blurry and noisy as the NFE decreases, whereas the samples generated by BFM remain structurally stable across different NFEs. Although fine-grained content is not always perfectly aligned with the original image, semantic information is largely preserved. These results provide further empirical evidence for the self-consistency property of BFM in cross-step sampling and indicate that BFM exhibits better tolerance to low NFE sampling during inference.

C.2 Dicussion on Probability Path Design for Comparison

To clarify, the probability path used by BFM in these experiments is the optimal transport path defined in (5), which is also called the linear probability path. The probability path of BLD is the cosine path defined in BLD. We now explain why we adopted this setting. Flow Matching Lipman et al. (2022) demonstrated that the optimal transport path achieves better performance in terms of fast training and low-NFE sampling. In their comparison, the baseline of the continuous diffusion model is DDPM Ho et al. (2020), which has a curved probability path. Therefore, we maintain this comparison consistency in terms of the comparison between flow-based models and diffusion-based

models in binary space. Below, we show the continuous probability paths of Flow Matching and DDPM, and we also visualize the probability paths of BFM and BLD for better understanding.

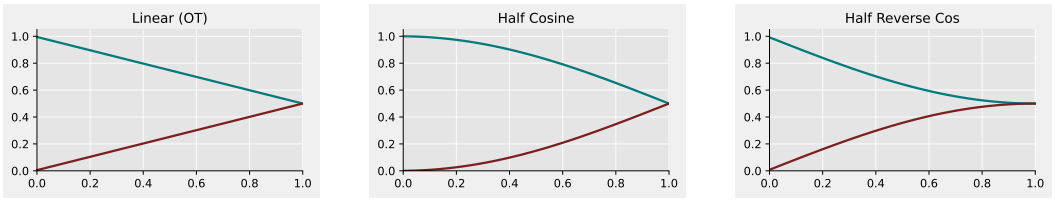


(a) Flow Matching path. (b) DDPM path. (c) BFM Linear path. (d) BLD Cosine path.

Figure 13: Illustration of probability paths in continuous and Bernoulli spaces. Panels (a) and (b) show the evolution of the mean and variance along the probability paths for two representative continuous models, Flow Matching and Diffusion, where the source distribution is a data distribution centered at 3 and the target distribution is the Gaussian $\mathcal{N}(0, 1)$. Panels (c) and (d) depict the Bernoulli probability paths of BFM with an linear (optimal transport) path and BLD with a cosine path, respectively. More disscusion can be found in

C.3 Ablation Study on Bernoulli Probability Path Scheduling Strategies

Fig. 14 visualizes how, under different probability paths, the probabilities associated with 1 (blue curves) and 0 (red curves) are progressively driven towards 0.5 along the diffusion trajectory. To systematically assess the impact of these schedulers on model performance, we select 3 different probability path schedulers and conduct experiments on the LSUN churches 256×256 dataset using the official BLD codebase. We keep the network architecture, learning rate, loss function, and all optimization hyperparameters exactly the same, and train with a batch size of 96 for 50K iterations. The only difference lies in the algorithm at the training and sampling stages.



(a) Linear (b) Half cos (c) Reverse half cos

Figure 14: Bernoulli probability path schedulers.

We summarize the ablation results in Table 3. Under all three probability paths (linear, half-cosine, and reverse half-cosine), BFM consistently achieves lower FID than BLD. Among them, the linear scheduler gives the best overall performance for BFM (9.22). We also note that the FID of BFM varies slightly more across different schedulers than that of BLD, although the overall variation remains small.

Table 3: FID comparison of different Bernoulli probability path schedulers for BFM and BLD on LSUN churches 256×256.

Scheduler	**FID** ↓	
	BFM	**BLD**
Linear	**9.22**	9.55
Half-cosine	9.41	9.53
Reverse half-cos	9.27	9.54

C.4 FASTER TRAINING

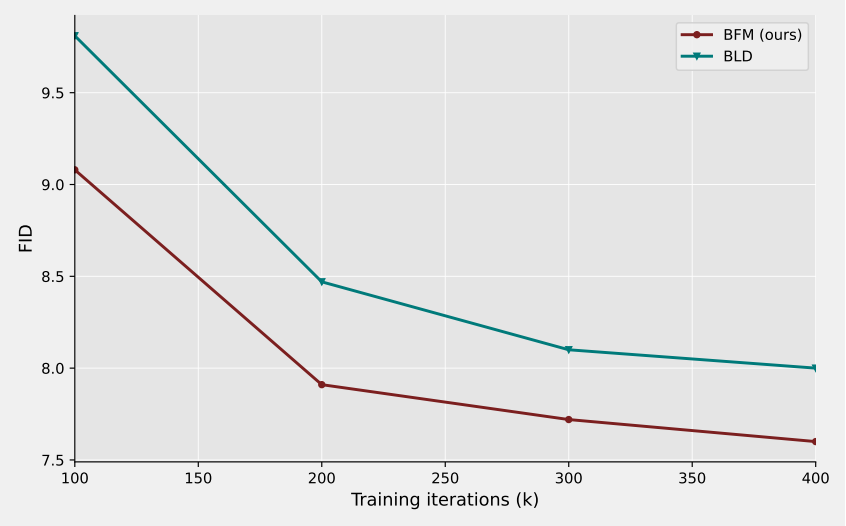


Figure 15: Training convergence comparison on LSUN Bedrooms. We visualize the FID scores over training iterations ($\times 10^3$). Both BFM (ours) and BLD are trained under identical experimental settings using a linear scheduler. Our BFM demonstrates faster convergence, achieving lower FID scores consistently across all training steps compared to BLD.

Faster training convergence. While both BFM and BLD employ the same linear scheduler for the probability path, we observe that BFM demonstrates significantly superior training efficiency. To ensure a fair comparison, we conducted strictly controlled experiments on the LSUN Bedrooms dataset, maintaining identical hyperparameters and hardware environments for both methods (except for the core algorithm).

Fig. 15 illustrates the evolution of FID during training. BFM is able to lower the FID faster and to a greater extent than BLD throughout the training process. In particular, BFM achieves an FID of 7.91 at just 200k iterations, which already exceeds the performance of BLD at 400k iterations (FID 8.0). This implies that BFM requires approximately 50% fewer iterations to reach comparable generation quality, substantially reducing the computational cost required for model training.