Understanding Galaxy Morphology Evolution Through Cosmic Time via Redshift Conditioned Diffusion Models

Author

Tianyue Yang[1]

Supervised by

Dr Sandro Tacchella[2]

[1] *Department of Physics, University of Cambridge*
[2] *Institute of Astronomy, University of Cambridge*

Word Count: 6945 / 7000

Abstract

Understanding galaxy morphology evolution across cosmic time requires models that can generate realistic galaxy populations conditioned on redshift. In this work, we study efficient redshift-conditioned generative modeling for astrophysical image synthesis using diffusion models and pixel-MeanFlow. We first review the connections between score-based diffusion models, Flow Matching, one-step generative models, and modern diffusion samplers. We then evaluate DDPM, DDIM, DEIS-AB2, DPM++2M, and one-step pixel-MeanFlow on the GalaxiesML-64 dataset using morphology-based metrics, including ellipticity, semi-major axis, Sérsic index, and isophotal area. Our results show a clear accuracy-efficiency trade-off: standard DDPM sampling achieves the best distributional fidelity but requires high computational cost, while second-order samplers substantially improve efficiency over DDIM. Pixel-MeanFlow enables single-step generation and achieves competitive performance on several morphology statistics, though it remains weaker than many-step DDPM for fine-grained structure. These findings suggest that few-step and one-step generative models are promising tools for fast astrophysical image generation.

Contents

Introduction

Motivation. Understanding the formation and evolution of astrophysical systems has been an important frontier problem in modern astronomy. Galaxies are dynamical systems that evolve over cosmic time. Directly modeling this evolutionary process is challenging, since we cannot continuously observe a single galaxy over the extremely long timescales on which astrophysical systems evolve. As a result, studies of galaxy evolution are often restricted to learning from discrete snapshots of different systems observed at different cosmic time. Unlike large photometric surveys, such as the Sloan Digital Sky Survey (SDSS) [1], the Dark Energy Survey (DES) [2], the Kilo-Degree Survey (KiDS) [3], and the Hyper Suprime-Cam (HSC) Survey [4], which contain millions or even billions of galaxies, redshift-matched datasets such as GalaxiesML [5] often contain only a few hundred thousand samples because obtaining reliable redshift measurements requires substantial computational time, cost, and effort. In this study, we investigate how generative models can be used to synthesize galaxy images conditioned on cosmic redshift. By generating high-fidelity galaxy populations across cosmic time, this approach helps bridge the observational gap imposed by sparse temporal sampling and provides a novel computational lens for simulating and understanding galaxy evolutionary trajectories [6, 7].

Generative Modeling. Generative modeling techniques have long been used for image-based synthesis tasks. Prominent paradigms include Variational Autoencoders (VAEs) [8], Generative Adversarial Networks (GANs) [9], and Normalizing Flows (NFs) [10]. Since the introduction of Denoising Diffusion Probabilistic Models (DDPMs) [11], diffusion-based models have demonstrated superior performance on a wide range of generative vision tasks [12], and have been adapted for scientific tasks in various fields, including weather forecasting [13, 14], fluid mechanics [15, 16, 17, 18, 19], and material science [20]. Flow Matching [21, 22] (FM) has emerged as a promising alternative formulation to diffusion models, enabling more efficient sampling and more concise training objectives. However, these methods are intrinsically multi-step, and generation quality typically degrades significantly when the number of sampling steps is small. Various approaches have been proposed to address this limitation. For standard diffusion models, the number of sampling steps can be reduced by leveraging more efficient samplers, including Denoising Diffusion Implicit Models (DDIMs) [23], DPM-Solver [24], and DPM-Solver++ [25]. Another line of work focuses on few-step generative models. Continuous-time approaches, such as Consistency Models (CMs) [26], simplified Consistency Models (sCMs) [27], and Shortcut Diffusion Models [28], leverage self-supervised signals from many-step trajectories to reduce the number of required sampling steps [29]. In contrast, discrete-time approaches typically rely on distribution-matching techniques, with Inductive Moment Matching (IMM) [30] serving as a representative example. Recent studies suggest that continuous-time models generally achieve better generation quality [29].

Our Methods. In the one-step and few-step generation regime, MeanFlow models [31] (MF) have emerged as a leading framework. Built upon the Flow Matching formulation, these models learn the average velocity over a time interval. In this study, we explore the capability of the latent-free variant of MF, namely the pixel MeanFlow model (p-MF) [32], for continuous redshift-conditioned galaxy generation in astrophysics. Our work builds on prior studies of diffusion-based modeling for understanding galaxy morphology evolution [33]. Our study provides the following contributions to address these limitations and advance the field:

1. We adapt pixel-MeanFlow to the continuously conditioned setting and incorporate modern diffusion training techniques, such as classifier-free guidance, to improve generative performance.

2. We investigate the efficiency–accuracy trade-off of different few-step diffusion solvers and compare them against pixel-MeanFlow in the astrophysical image-generation setting.

3. We provide a systematic and extensive review of existing diffusion models, few-step diffusion models, and sampling techniques, with the goal of inspiring new applications of diffusion-based generative modeling in astrophysics.

To the best of the authors' knowledge, this is the first application of few-step diffusion models, together with a detailed investigation of solver efficiency, in the context of astrophysics. Therefore, our study represents a meaningful advancement in efficient generative modeling for astrophysical applications.

Methodology

2.1 Diffusion Models

2.1.1 Score-based Perspective: Noise-Conditional Score Networks

Score Functions and Energy-Based Models. Long before the recent breakthroughs in deep learning, physics-inspired complex networks were used for data storage and generation [34, 35]. A representative class of such models is the Boltzmann machine [34], which models data using the Boltzmann distribution:

$$p(\mathbf{x}) = \frac{e^{-E_\phi(\mathbf{x})}}{\int e^{-E_\phi(\mathbf{x})}\,\mathrm{d}\mathbf{x}} = \frac{1}{Z} e^{-E_\phi(\mathbf{x})}, \tag{2.1}$$

where Z is referred to as the *partition function* and ϕ represents the parameterization. For complex, high-dimensional real-world distributions, the integral defining the partition function is generally difficult to evaluate. This prevents the direct use of the ideal maximum-likelihood estimation (MLE) objective,

$$\mathcal{L}_{\text{MLE}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{e^{-E_\phi(\mathbf{x})}}{Z}\right] = -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[E_\phi(\mathbf{x})\right] - \log Z, \tag{2.2}$$

where the second term is intractable.

Definition 2.1: Score Function

For a density $p(\mathbf{x})$ on $\mathbb{R}^D$, the score function is defined as the gradient of the logarithmic probability density:

$$\mathbf{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x}), \qquad \mathbf{s} : \mathbb{R}^D \mapsto \mathbb{R}^D. \tag{2.3}$$

For the Boltzmann distribution, the corresponding score function is

$$\mathbf{s}_{\text{Boltzmann}}(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\phi(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z}_{=0}, \tag{2.4}$$

which allows us to bypass the partition function entirely. Assuming that the true score function is available, the original probability density can be recovered up to a reference density value via

$$\log p(\mathbf{x}) = \log p(\mathbf{x}_0) + \int_0^1 \mathbf{s}(\mathbf{x}_0 + t(\mathbf{x}-\mathbf{x}_0))^\top (\mathbf{x}-\mathbf{x}_0)\,\mathrm{d}t, \tag{2.5}$$

where $\mathbf{x}_0$ is a sampled reference point. Thus, using the score function as a representation of the probability landscape does not discard information about the underlying density, up to an additive normalization constant.

Data Generation by Score Matching. It is therefore natural to train a neural network for data generation using the *score matching objective*:

$$\mathcal{L}_{\text{SM}} = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\left\|\mathbf{s}_\phi(\mathbf{x}) - \mathbf{s}(\mathbf{x})\right\|_2^2\right]. \tag{2.6}$$

In practice, however, the ground-truth score function $\mathbf{s}(\mathbf{x})$ is not available. To address this issue, Hyvärinen and Dayan [36] derived an equivalent objective that is tractable to compute during training.

Theorem 2.1: Tractable Score Matching

It can be shown that the loss

$$\tilde{\mathcal{L}}_{\mathrm{SM}} := \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}}\left[\mathrm{Tr}\big(\nabla_{\mathbf{x}}\mathbf{s}_\phi(\mathbf{x})\big) + \frac{1}{2}\left\|\mathbf{s}_\phi(\mathbf{x})\right\|_2^2\right] \tag{2.7}$$

differs from the ideal score matching loss only by an additive constant:

$$\mathcal{L}_{\mathrm{SM}} = \tilde{\mathcal{L}}_{\mathrm{SM}} + C, \tag{2.8}$$

where C is independent of the model parameters.

Sampling by Langevin Dynamics. Suppose that the true score function $\mathbf{s}$ is available. A naive deterministic sampling procedure of the form

$$\mathrm{d}\mathbf{x}(t) = \mathbf{s}(\mathbf{x}(t))\,\mathrm{d}t \tag{2.9}$$

can easily converge to local modes due to its deterministic nature and lack of exploration. It is therefore desirable to introduce a stochastic noise term:

$$\mathrm{d}\mathbf{x}(t) = \mathbf{s}(\mathbf{x}(t))\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\mathbf{W}_t, \tag{2.10}$$

where $\mathbf{W}_t$ denotes a standard Brownian motion. This yields the Langevin stochastic differential equation:

$$\mathrm{d}\mathbf{x}(t) = \nabla_{\mathbf{x}} \log p(\mathbf{x}(t))\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}\mathbf{W}_t. \tag{2.11}$$

The factor $\sqrt{2}$ ensures that $p(\mathbf{x})$ is the stationary distribution of the corresponding dynamics. Discretizing this equation using the Euler–Maruyama scheme gives the sampling update

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \eta\,\mathbf{s}(\mathbf{x}_i) + \sqrt{2\eta}\,\boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2.12}$$

where η is the step size.

Denoising Score Matching. Returning to Theorem 2.1, although the tractable score matching objective avoids explicit access to the ground-truth score, computing the Jacobian trace term, $\mathrm{Tr}(\nabla_{\mathbf{x}}\mathbf{s}_\phi(\mathbf{x}))$, can still incur substantial computational cost during training. To address this issue, Song et al. [37] proposed *sliced score matching*, which uses Hutchinson's estimator to obtain a stochastic estimate of the Jacobian trace [37]. Although this makes score matching more computationally feasible, the method can still suffer as the dimensionality of $\mathbf{x}$ increases.

Vincent [38] proposed *denoising score matching* (DSM), which mitigates this issue by perturbing samples $\mathbf{x} \sim p_{\mathrm{data}}$ with noise drawn from a known conditional distribution $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$, where σ denotes the noise scale. The corresponding objective is

$$\mathcal{L}_{\mathrm{DSM}} = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\,\tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left[\left\|\mathbf{s}_\phi(\tilde{\mathbf{x}};\sigma) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\right\|_2^2\right]. \tag{2.13}$$

Suppose the injected noise is Gaussian:

$$\tilde{\mathbf{x}} = \mathbf{x} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.14}$$

Then the perturbed data follow

$$\tilde{\mathbf{x}}|\mathbf{x} \sim \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I}), \tag{2.15}$$

and the corresponding conditional score admits the closed-form expression

$$\nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \nabla_{\tilde{\mathbf{x}}} \left[-\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|_2^2}{2\sigma^2} \right] = \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma^2}. \tag{2.16}$$

Therefore, the denoising score matching objective becomes

$$\begin{aligned} \mathcal{L}_{\text{DSM}} &= \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}},\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \mathbf{s}_\phi(\tilde{\mathbf{x}}; \sigma) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma^2} \right\|_2^2 \right] \\ &= \frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}},\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \mathbf{s}_\phi(\mathbf{x} + \sigma \boldsymbol{\epsilon}; \sigma) + \frac{\boldsymbol{\epsilon}}{\sigma} \right\|_2^2 \right]. \end{aligned} \tag{2.17}$$

This loss is theoretically equivalent to the SM loss by an additive constant. Sampling then follows the standard Langevin dynamics:

$$\tilde{\mathbf{x}}_{i+1} = \tilde{\mathbf{x}}_i + \eta \underbrace{\mathbf{s}_\phi(\tilde{\mathbf{x}}_i; \sigma)}_{\approx \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}_i)} + \sqrt{2\eta}\, \boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.18}$$

Intuitively, the injected noise smooths the landscape of the original score function $\mathbf{s}(\mathbf{x})$, thereby improving sampling stability and convergence. This perspective can be further illustrated through the process of denoising. *Tweedie's formula* [39] provides a principled basis for denoising noisy observations using the score of the perturbed data distribution.

Theorem 2.2: Tweedie's Formula

Assume $\mathbf{x} \sim p_{\text{data}}$ and define a noisy observation

$$\tilde{\mathbf{x}} \sim \mathcal{N}(\alpha \mathbf{x}, \sigma^2 \mathbf{I}), \qquad \alpha \neq 0. \tag{2.19}$$

Tweedie's formula states that

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\tilde{\mathbf{x}})}[\mathbf{x}|\tilde{\mathbf{x}}] = \frac{1}{\alpha} \left[\tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}) \right], \tag{2.20}$$

where the expectation is taken with respect to the posterior distribution $p(\mathbf{x}|\tilde{\mathbf{x}})$, and $p_\sigma(\tilde{\mathbf{x}})$ denotes the marginal distribution of the noisy observation.

Suppose we have a score network trained using the DSM objective. Tweedie's formula can then be written as

$$\underbrace{\mathbb{E}[\mathbf{x}|\tilde{\mathbf{x}}]}_{\text{Denoiser}} = \frac{1}{\alpha} \left[\tilde{\mathbf{x}} + \sigma^2 \mathbf{s}_\phi(\tilde{\mathbf{x}}) \right], \tag{2.21}$$

where $\mathbf{s}_\phi(\tilde{\mathbf{x}})$ approximates the score of the perturbed data distribution. Thus, a single gradient-ascent step on the noisy log-likelihood, with step size σ^2, yields the denoised estimate, namely the conditional mean of the clean signal. This establishes a direct connection between DSM training and denoising.

Multi-Level DSM: Noise-Conditional Score Networks. In practice, the performance of DSM is often limited during sampling. At low noise levels, DSM can preserve fine-grained details of the score landscape, but sampling may become unstable and inefficient because the dynamics can be disrupted in low-density regions of the data distribution, where the score may be poorly estimated or have vanishing gradients [40, 41]. At high noise levels, sampling can more easily move across different modes of the distribution, but the injected noise also smooths out fine-scale details. To address this trade-off, Song and Ermon [40] proposed Noise-Conditional Score Networks (NCSNs), which allow a single score network to operate across multiple noise levels.

The NCSN objective is a weighted average of DSM losses evaluated at different noise scales:

$$\mathcal{L}_{\text{NCSN}} = \sum_{i=1}^{L} \lambda(\sigma_i)\mathcal{L}_{\text{DSM}}(\phi;\sigma_i). \tag{2.22}$$

The trained NCSN model is expected to approximate the score field of the perturbed data distribution at each noise level:

$$\mathbf{s}_\phi(\cdot;\sigma_i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\cdot), \qquad i = 1,\dots,L. \tag{2.23}$$

Sampling from NCSNs is then performed using *annealed Langevin dynamics*, which proceeds progressively from high noise levels to low noise levels. At each noise level σ_l, the update is given by

$$\tilde{\mathbf{x}}_{i+1} = \tilde{\mathbf{x}}_i + \eta_l\, \mathbf{s}_\phi(\tilde{\mathbf{x}}_i;\sigma_l) + \sqrt{2\eta_l}\,\boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0},\mathbf{I}). \tag{2.24}$$

The step size is typically scaled according to the noise level, with larger noise levels using larger step sizes:

$$\eta_l \propto \frac{\sigma_l^2}{\sigma_1^2}, \tag{2.25}$$

where σ_1 denotes the smallest noise level, at which the final sample $\hat{\mathbf{x}}$ is obtained. In practice, NCSNs typically require many MCMC steps to converge at each noise level σ_l. This *slow mixing* phenomenon leads to slow inference, since the learned score is reliable only for small perturbations and therefore requires small step sizes and many iterations per noise level to avoid bias or instability. Moreover, the updates are strictly sequential, as each iteration depends on the previous one, and each step requires an expensive neural network evaluation. As a result, sampling requires $O(LK)$ sequential network evaluations, where L is the number of noise levels and K is the number of MCMC steps per level, making inference computationally expensive.

2.1.2 Variational Perspective: Diffusion Denoising Probabilistic Model

Variational Autoencoders and Evidence Lower Bounds. In generative modeling, Variational Autoencoders (VAEs) [8] provide a foundational framework for variational probabilistic modeling. A modern VAE consists of two components: an *encoder*, or approximate posterior, $q_\phi(\mathbf{z}|\mathbf{x})$, and a *decoder*, or likelihood model, $p_\theta(\mathbf{x}|\mathbf{z})$, where ϕ and θ denote the corresponding model parameters and $\mathbf{z}$ is a latent variable. We begin by deriving the evidence lower bound (ELBO).

Lemma 2.1: ELBO for VAEs

For any data point $\mathbf{x}$, the log-likelihood satisfies

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}_{\text{ELBO}}(\mathbf{x};\theta,\phi), \tag{2.26}$$

where the ELBO is given by

$$\mathcal{L}_{\text{ELBO}}(\mathbf{x};\theta,\phi) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathcal{D}_{\text{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z})\right). \tag{2.27}$$

Proof. Using Jensen's inequality, we obtain

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z})\, \mathrm{d}\mathbf{z} = \log \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, \mathrm{d}\mathbf{z} \\
&= \log \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\, \mathrm{d}\mathbf{z} = \log \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\
&\geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \mathcal{D}_{\mathrm{KL}} \left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z}) \right).
\end{aligned} \tag{2.28}$$

□

The first term in the ELBO is the expected log-likelihood, often interpreted as the reconstruction term. When the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ is modeled as a Gaussian distribution with fixed variance, maximizing this term is equivalent, up to an additive and multiplicative constant, to minimizing the mean squared error (MSE). If a Laplace likelihood is assumed instead, the corresponding reconstruction loss reduces to an ℓ_1 loss.

Although this framework is promising, simply increasing the depth of a VAE is often insufficient. One limitation is that many VAE variants, including β-VAE [42], typically rely on Gaussian latent distributions. This assumption can restrict the expressiveness of the latent representation, particularly when the data distribution is multi-modal. Another limitation arises from a common failure mode known as posterior collapse. To see this, we further decompose the expected ELBO:

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}} \left[\mathcal{L}_{\mathrm{ELBO}} \right] &= \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}} \left[\mathcal{D}_{\mathrm{KL}} \left(q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, p(\mathbf{z}) \right) \right] \\
&= \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) \right] - \mathcal{I}_q(\mathbf{x}; \mathbf{z}) - \mathcal{D}_{\mathrm{KL}} \left(q_\phi(\mathbf{z}) \,\|\, p(\mathbf{z}) \right),
\end{aligned} \tag{2.29}$$

where $q_\phi(\mathbf{z}) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}}} \left[q_\phi(\mathbf{z}|\mathbf{x}) \right]$ denotes the aggregated posterior, and the mutual information is defined as

$$\mathcal{I}_q(\mathbf{x}; \mathbf{z}) = \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{data}},\, \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z})} \right]. \tag{2.30}$$

A clear failure mode occurs when the decoder can model $p(\mathbf{x})$ without using the latent variable $\mathbf{z}$. In this case, a degenerate optimum is obtained when

$$q_\phi(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}), \tag{2.31}$$

which yields $\mathcal{I}_q(\mathbf{x}; \mathbf{z}) = 0$ and $q_\phi(\mathbf{z}) = p(\mathbf{z})$. This corresponds to posterior collapse, where the latent representation carries no information about the input. Such behavior is especially common when the decoder is overly expressive [43, 44].

Hierarchical Variational Autoencoders. Hierarchical VAEs (HVAEs) [44] mitigate these limitations by introducing multiple layers of stochastic latent variables. The encoder takes the form of a Markovian factorization:

$$q_\theta(\mathbf{z}_{1:L}|\mathbf{x}) = q_\theta(\mathbf{z}_1|\mathbf{x}) \prod_{i=2}^{L} q_\theta(\mathbf{z}_i|\mathbf{z}_{i-1}). \tag{2.32}$$

The corresponding generative model is given by

$$p_\phi(\mathbf{x}, \mathbf{z}_{1:L}) = p_\phi(\mathbf{x}|\mathbf{z}_1) \prod_{i=2}^{L} p_\phi(\mathbf{z}_{i-1}|\mathbf{z}_i) p(\mathbf{z}_L). \tag{2.33}$$

The ELBO for an HVAE has the same general form as in the standard VAE:

$$\mathcal{L}_{\mathrm{ELBO}} = \mathbb{E}_{q_\theta(\mathbf{z}_{1:L}|\mathbf{x})} \left[\log \frac{p(\mathbf{z}_L) p_\phi(\mathbf{x}|\mathbf{z}_1) \prod_{i=2}^{L} p_\phi(\mathbf{z}_{i-1}|\mathbf{z}_i)}{q_\theta(\mathbf{z}_1|\mathbf{x}) \prod_{i=2}^{L} q_\theta(\mathbf{z}_i|\mathbf{z}_{i-1})} \right], \tag{2.34}$$

where the derivation follows directly from Lemma 2.1. This objective can be decomposed as

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\theta}\left[\log p_\phi(\mathbf{x}|\mathbf{z}_1)\right] - \mathbb{E}_{q_\theta}\left[\mathcal{D}_{\text{KL}}\left(q_\theta(\mathbf{z}_1|\mathbf{x}) \,\|\, p_\phi(\mathbf{z}_1|\mathbf{z}_2)\right)\right] \\ - \sum_{i=2}^{L-1} \mathbb{E}_{q_\theta}\left[\mathcal{D}_{\text{KL}}\left(q_\theta(\mathbf{z}_i|\mathbf{z}_{i-1}) \,\|\, p_\phi(\mathbf{z}_i|\mathbf{z}_{i+1})\right)\right] \\ - \mathbb{E}_{q_\theta}\left[\mathcal{D}_{\text{KL}}\left(q_\theta(\mathbf{z}_L|\mathbf{z}_{L-1}) \,\|\, p(\mathbf{z}_L)\right)\right].\end{aligned} \tag{2.35}$$

The first term can be interpreted as the reconstruction term, as in a standard single-latent VAE, while the remaining terms encourage consistency between the encoder and decoder latent hierarchies. This hierarchical structure substantially improves the expressiveness of the model [44]. However, although HVAEs extend the VAE framework by introducing multiple latent layers, their training presents additional challenges. Since the encoder and decoder must be optimized jointly, learning can become unstable: lower-level latent variables and the decoder may already be sufficient to reconstruct $\mathbf{x}$, leaving higher-level latent variables with little effective learning signal. Moreover, gradient information reaching deeper latent variables is often indirect and weak, making it difficult for those variables to contribute meaningfully to the generative process.

Denoising Diffusion Probabilistic Models. Diffusion models, as introduced in *Denoising Diffusion Probabilistic Models* (DDPMs) [11], can be viewed as an improved variant of hierarchical VAEs with a fixed closed-form encoder. In this formulation, the forward process is fixed, and learning focuses on the reverse, or denoising, process. DDPMs define a Markovian forward diffusion process as

$$q(\mathbf{z}_i|\mathbf{z}_{i-1}) := \mathcal{N}\left(\mathbf{z}_i; \sqrt{1-\beta_i}\,\mathbf{z}_{i-1}, \beta_i \mathbf{I}\right), \tag{2.36}$$

where $\{\beta_i\}_{i=1}^L$ is a pre-determined noise schedule. Letting $\alpha_i = 1 - \beta_i$ and $\bar{\alpha}_i = \prod_{j=1}^i \alpha_j$, this construction yields the closed-form forward diffusion kernel

$$q(\mathbf{z}_i|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_i; \sqrt{\bar{\alpha}_i}\,\mathbf{x}, (1-\bar{\alpha}_i)\mathbf{I}\right). \tag{2.37}$$

DDPMs then train a neural network to approximate the reverse process $p_\phi(\mathbf{z}_{i-1}|\mathbf{z}_i)$, which acts as a denoising model. This motivates the derivation of the ELBO for DDPMs.

Theorem 2.3: ELBO for DDPMs

For any data point $\mathbf{x}$, the log-likelihood satisfies

$$\log p_\phi(\mathbf{x}) \geq \mathcal{L}_{\text{ELBO}}(\mathbf{x}; \phi), \tag{2.38}$$

where the ELBO is given by

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}(\mathbf{x};\phi) = \mathbb{E}_{\mathbf{z}_1\sim q(\mathbf{z}_1|\mathbf{x})}\left[\log p_\phi(\mathbf{x}|\mathbf{z}_1)\right] - \mathcal{D}_{\text{KL}}\left(q(\mathbf{z}_L|\mathbf{x}) \,\|\, p(\mathbf{z}_L)\right) \\ - \sum_{i=2}^{L} \mathbb{E}_{\mathbf{z}_i\sim q(\mathbf{z}_i|\mathbf{x})}\left[\mathcal{D}_{\text{KL}}\left(q(\mathbf{z}_{i-1}|\mathbf{z}_i,\mathbf{x}) \,\|\, p_\phi(\mathbf{z}_{i-1}|\mathbf{z}_i)\right)\right].\end{aligned} \tag{2.39}$$

Equation 2.39 corresponds to the variational lower-bound formulation underlying Equation (3) in Ho, Jain, and Abbeel [11]. Comparing Equation 2.39, Equation 2.35, and Equation 2.27, we summarize the relationships among VAEs, HVAEs, and DDPMs in Table 2.1.

Connection between DDPMs and NCSNs. Finally, we show that DDPMs and NCSNs can be connected through Tweedie's formula. Consider the forward diffusion kernel $q(\mathbf{z}_i|\mathbf{x})$. Since

$$q(\mathbf{z}_i|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}_i; \sqrt{\bar{\alpha}_i}\mathbf{x}, \sigma_i^2\mathbf{I}\right), \qquad \sigma_i^2 = 1 - \bar{\alpha}_i, \tag{2.40}$$

Table 2.1: Comparison of the ELBO terms in VAEs, HVAEs, and DDPMs.

Term Name	Model	Expression
Prior	VAE	$\mathcal{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{z}\|\mathbf{x}) \,\|\, p(\mathbf{z})\right)$
	HVAE	$\mathbb{E}_{q_\theta}\left[\mathcal{D}_{\mathrm{KL}}\left(q_\theta(\mathbf{z}_L\|\mathbf{z}_{L-1}) \,\|\, p(\mathbf{z}_L)\right)\right]$
	DDPM	$\mathcal{D}_{\mathrm{KL}}\left(q(\mathbf{z}_L\|\mathbf{x}) \,\|\, p(\mathbf{z}_L)\right)$
Reconstruction	VAE	$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}\|\mathbf{x})}\left[\log p_\theta(\mathbf{x}\|\mathbf{z})\right]$
	HVAE	$\mathbb{E}_{q_\theta}\left[\log p_\phi(\mathbf{x}\|\mathbf{z}_1)\right]$
	DDPM	$\mathbb{E}_{\mathbf{z}_1\sim q(\mathbf{z}_1\|\mathbf{x})}\left[\log p_\phi(\mathbf{x}\|\mathbf{z}_1)\right]$
Denoising	VAE	No explicit denoising process.
	HVAE	$\mathbb{E}_{q_\theta}\left[\mathcal{D}_{\mathrm{KL}}\left(q_\theta(\mathbf{z}_1\|\mathbf{x}) \,\|\, p_\phi(\mathbf{z}_1\|\mathbf{z}_2)\right)\right] + \sum_{i=2}^{L-1}\mathbb{E}_{q_\theta}\left[\mathcal{D}_{\mathrm{KL}}\left(q_\theta(\mathbf{z}_i\|\mathbf{z}_{i-1}) \,\|\, p_\phi(\mathbf{z}_i\|\mathbf{z}_{i+1})\right)\right]$
	DDPM	$\sum_{i=2}^{L}\mathbb{E}_{\mathbf{z}_i\sim q(\mathbf{z}_i\|\mathbf{x})}\left[\mathcal{D}_{\mathrm{KL}}\left(q(\mathbf{z}_{i-1}\|\mathbf{z}_i,\mathbf{x}) \,\|\, p_\phi(\mathbf{z}_{i-1}\|\mathbf{z}_i)\right)\right]$

the conditional score is given by

$$\nabla_{\mathbf{z}_i}\log q(\mathbf{z}_i|\mathbf{x}) = -\frac{\mathbf{z}_i - \sqrt{\bar{\alpha}_i}\mathbf{x}}{\sigma_i^2}. \tag{2.41}$$

Taking the posterior expectation over $\mathbf{x}|\mathbf{z}_i$, we obtain the marginal score

$$\nabla_{\mathbf{z}_i}\log q(\mathbf{z}_i) = -\frac{\mathbf{z}_i - \sqrt{\bar{\alpha}_i}\mathbb{E}[\mathbf{x}|\mathbf{z}_i]}{\sigma_i^2}. \tag{2.42}$$

Rearranging this expression yields Tweedie's formula for DDPMs:

$$\mathbb{E}[\mathbf{x}|\mathbf{z}_i] = \frac{1}{\sqrt{\bar{\alpha}_i}}\left(\mathbf{z}_i + \sigma_i^2\nabla_{\mathbf{z}_i}\log q(\mathbf{z}_i)\right), \tag{2.43}$$

which has the same form as Theorem 2.2.

In practice, DDPMs are commonly implemented using the *ϵ-prediction* parameterization, where the optimal network predicts

$$\epsilon_\phi(\mathbf{z}_i, i) = \mathbb{E}[\boldsymbol{\epsilon}|\mathbf{z}_i], \tag{2.44}$$

with

$$\boldsymbol{\epsilon} = \frac{\mathbf{z}_i - \sqrt{\bar{\alpha}_i}\mathbf{x}}{\sigma_i}. \tag{2.45}$$

Therefore,

$$\mathbb{E}[\boldsymbol{\epsilon}|\mathbf{z}_i] = \frac{\mathbf{z}_i - \sqrt{\bar{\alpha}_i}\mathbb{E}[\mathbf{x}|\mathbf{z}_i]}{\sigma_i}. \tag{2.46}$$

Substituting Tweedie's formula into this expression gives

$$\mathbb{E}[\boldsymbol{\epsilon}|\mathbf{z}_i] = -\sigma_i\nabla_{\mathbf{z}_i}\log q(\mathbf{z}_i). \tag{2.47}$$

This is the Tweedie-form interpretation of ϵ-prediction in DDPMs.

From the NCSN perspective, the optimal noise-conditional score network estimates the score of the perturbed data distribution at each noise level:

$$\mathbf{s}_\phi(\mathbf{z}_i; \sigma_i) \approx \nabla_{\mathbf{z}_i} \log q(\mathbf{z}_i). \tag{2.48}$$

Combining this relation with the DDPM ϵ-prediction objective yields

$$\mathbf{s}_\phi(\mathbf{z}_i; \sigma_i) \approx -\frac{1}{\sigma_i} \epsilon_\phi(\mathbf{z}_i, i). \tag{2.49}$$

Thus, DDPMs and NCSNs can be viewed as two parameterizations of the same underlying score-learning principle under Gaussian perturbations. Up to this point, we have unified the two views. However, the discrete-time formulations of both NCSNs and DDPMs limit the flexibility of this framework, motivating continuous-time extensions.

2.1.3 Continuous Formulation: From Score-SDE to EDM

Song et al. [41] (commonly referred to as *Score-SDE* model) first proposed a systematic framework that unifies diffusion-based models through stochastic differential equations (SDEs). We begin by stating the general form:

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}(t), \tag{2.50}$$

where $\mathbf{w}(t)$ denotes a Wiener process, or Brownian motion. We first discuss several useful properties of this general SDE and then show that both DDPMs and NCSNs arise as special cases of this formulation. The SDE can be interpreted as consisting of two components:

$$\mathrm{d}\mathbf{x}(t) = \underbrace{\mathbf{f}(\mathbf{x}(t), t)\,\mathrm{d}t}_{\text{mean evolution}} + \underbrace{g(t)\,\mathrm{d}\mathbf{w}(t)}_{\text{variance evolution}}. \tag{2.51}$$

This formulation allows us to derive closed-form conditional distributions at different times by solving the corresponding mean and variance ordinary differential equations (ODEs). Although, in principle, $\mathbf{f}(\mathbf{x}(t), t)$ can take a general form, a common choice is the affine drift

$$\mathbf{f}(\mathbf{x}(t), t) = f(t)\mathbf{x}(t), \tag{2.52}$$

where $f(t)$ is a scalar function of time. Both DDPMs and NCSNs can be expressed using this type of drift. Solving the resulting ODEs gives

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \mathbf{m}(t), P(t)\mathbf{I}\right), \tag{2.53}$$

where

$$\mathbf{m}(t) = e^{\int_0^t f(u)\,\mathrm{d}u}\mathbf{x}_0, \qquad P(t) = \int_0^t e^{2\int_s^t f(u)\,\mathrm{d}u} g^2(s)\,\mathrm{d}s. \tag{2.54}$$

Here, $\mathbf{m}(0) = \mathbf{x}_0$ and $P(0) = 0$. By choosing $f(t)$ and $g(t)$ appropriately, the distribution gradually loses information about the original data as t increases and approaches a simple prior distribution:

$$p(\mathbf{x}_T|\mathbf{x}_0) \approx p(\mathbf{x}_T) \approx p_{\text{prior}}(\mathbf{x}), \qquad T \gg 0. \tag{2.55}$$

The evolution from 0 to T describes the data corruption process; for data generation, we need to reverse this SDE. Reversing a stochastic forward process requires the corresponding *marginal densities* to remain aligned. This relationship is characterized by the *Fokker–Planck equation* [45, 41].

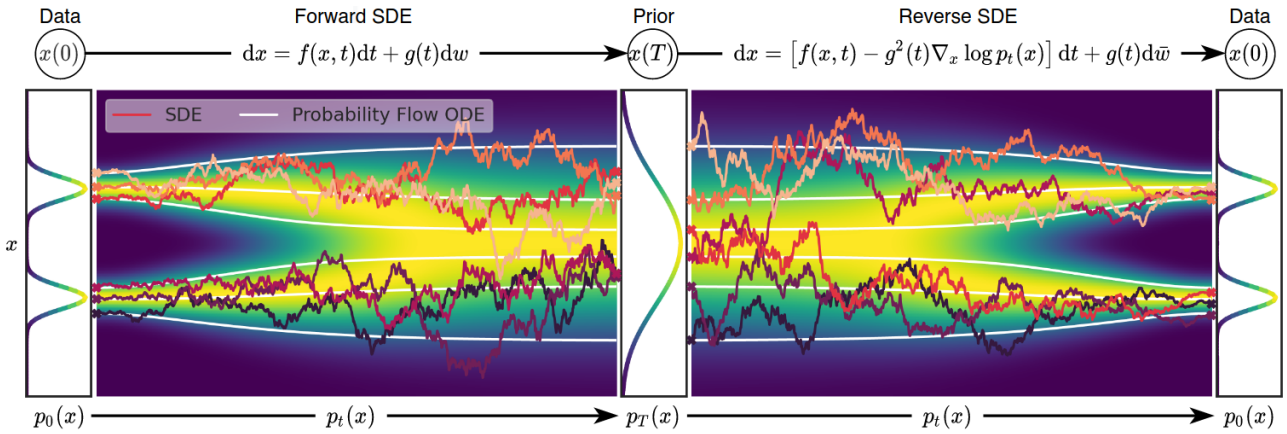

Figure 2.1: Plot adapted from Figure 2 of Song et al. [41]. A forward-time SDE transforms the data distribution $\mathbf{x}(0)$ into a prior distribution $\mathbf{x}(T)$, such as a Gaussian prior in DDPMs. Data generation is then performed by sampling from the prior distribution and evolving the sample backward through the reverse-time SDE. The corresponding PF-ODE provides an alternative deterministic generation process, although it is not shown here.

Theorem 2.4: Fokker–Planck Equation

Let $\{\mathbf{x}(t)\}_{t\in[0,T]}$ evolve according to the forward SDE

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}(t), \tag{2.56}$$

with initial condition $\mathbf{x}(0) \sim p_{\text{data}}$. Then its marginal densities p_t satisfy the Fokker–Planck equation

$$\begin{aligned}\partial_t p_t(\mathbf{x}) &= -\nabla_{\mathbf{x}} \cdot [\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}g^2(t)\nabla^2_{\mathbf{x}} p_t(\mathbf{x}) \\ &= -\nabla_{\mathbf{x}} \cdot [\mathbf{v}(\mathbf{x}, t)p_t(\mathbf{x})]\,,\end{aligned} \tag{2.57}$$

where

$$\mathbf{v}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}). \tag{2.58}$$

Then both the probability flow ODE (PF-ODE) and the reverse-time SDE yield the same family of marginal densities $\{p_t\}_{t\in[0,T]}$, with the latter evolving in reverse time:

1. The PF-ODE $\{\tilde{\mathbf{x}}(t)\}_{t\in[0,T]}$,
$$\frac{\mathrm{d}\tilde{\mathbf{x}}(t)}{\mathrm{d}t} = \mathbf{v}(\tilde{\mathbf{x}}(t), t), \tag{2.59}$$
when initialized with $\tilde{\mathbf{x}}(0) \sim p_0$ and run forward in t, or equivalently initialized with $\tilde{\mathbf{x}}(T) \sim p_T$ and run backward in t, has marginals $\tilde{\mathbf{x}}(t) \sim p_t$ for all $t \in [0, T]$.

2. The reverse-time SDE $\{\bar{\mathbf{x}}(t)\}_{t\in[0,T]}$,
$$\mathrm{d}\bar{\mathbf{x}}(t) = \left[\mathbf{f}(\bar{\mathbf{x}}(t), t) - g^2(t)\nabla_{\bar{\mathbf{x}}} \log p_t(\bar{\mathbf{x}}(t))\right]\mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}(t), \tag{2.60}$$
where $\bar{\mathbf{w}}(t)$ is a standard Wiener process in reverse time, has marginals $\bar{\mathbf{x}}(t) \sim p_{T-t}$ when initialized with $\bar{\mathbf{x}}(0) \sim p_T$.

The appearance of the score function,

$$\mathbf{s}_\phi(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x}), \tag{2.61}$$

is therefore unsurprising: it is precisely the quantity required to construct both the PF-ODE and the reverse-time SDE for generating samples from the prior distribution. This process is illustrated in

Figure 2.1. We now discuss how this general framework applies to DDPMs and NCSNs [41][1].

SDE for NCSNs: Variance Exploding. Recall the NCSN perturbation parameterization:

$$\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.62}$$

This can be written as a Markov chain:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}\, \boldsymbol{\epsilon}. \tag{2.63}$$

Introducing a continuous time index, we obtain

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}(t - \Delta t) + \sqrt{\frac{\sigma^2(t) - \sigma^2(t - \Delta t)}{\Delta t}}\, \sqrt{\Delta t}\, \boldsymbol{\epsilon} \\ \implies \mathrm{d}\mathbf{x}(t) &= 0\, \mathrm{d}t + \sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}\, \mathrm{d}\mathbf{w}(t). \end{aligned} \tag{2.64}$$

Here, $\mathrm{d}\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}\, \mathrm{d}t)$, so that $\sqrt{\Delta t}\boldsymbol{\epsilon} \to \mathrm{d}\mathbf{w}(t)$ in the continuous-time limit. Therefore, the SDE governing the NCSN perturbation process is

$$\mathrm{d}\mathbf{x}(t) = \sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}\, \mathrm{d}\mathbf{w}(t). \tag{2.65}$$

Taking the variance of both sides gives

$$\mathrm{Var}[\mathrm{d}\mathbf{x}(t)] = \mathrm{Var}\left[\sqrt{\frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}}\, \mathrm{d}\mathbf{w}(t)\right] = \frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}\, \mathrm{d}t. \tag{2.66}$$

Thus, the variance $P(t)$ satisfies the ODE

$$\frac{\mathrm{d}P(t)}{\mathrm{d}t} = \frac{\mathrm{d}[\sigma^2(t)]}{\mathrm{d}t}. \tag{2.67}$$

Consequently, $P(t) \approx P(0) + \sigma^2(t)$, implying that the variance increases monotonically with time. This motivates the name *variance exploding* (VE), and we refer to this formulation as the VE-SDE.

SDE for DDPMs: Variance Preserving. Similarly, recall the Markovian DDPM parameterization:

$$q(\mathbf{x}_i|\mathbf{x}_{i-1}) = \mathcal{N}\left(\mathbf{x}_i; \sqrt{1 - \beta_i}\, \mathbf{x}_{i-1}, \beta_i \mathbf{I}\right), \tag{2.68}$$

or equivalently,

$$\mathbf{x}_i = \sqrt{1 - \beta_i}\, \mathbf{x}_{i-1} + \sqrt{\beta_i}\, \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.69}$$

To obtain the continuous-time limit, let $\beta_i = \beta(t)\Delta t$. Then, for small Δt,

$$\sqrt{1 - \beta(t)\Delta t} = 1 - \frac{1}{2}\beta(t)\Delta t + o(\Delta t). \tag{2.70}$$

Thus,

$$\begin{aligned} \mathbf{x}(t + \Delta t) &= \left(1 - \frac{1}{2}\beta(t)\Delta t\right)\mathbf{x}(t) + \sqrt{\beta(t)}\sqrt{\Delta t}\, \boldsymbol{\epsilon} \\ \implies \mathrm{d}\mathbf{x}(t) &= -\frac{1}{2}\beta(t)\mathbf{x}(t)\, \mathrm{d}t + \sqrt{\beta(t)}\, \mathrm{d}\mathbf{w}(t), \end{aligned} \tag{2.71}$$

[1] NCSNs [40] are referred to as score matching with Langevin dynamics (SMLD) models in Song et al. [41].

where $\mathrm{d}\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}\,\mathrm{d}t)$. Therefore, the continuous-time SDE corresponding to DDPMs is

$$\mathrm{d}\mathbf{x}(t) = -\frac{1}{2}\beta(t)\mathbf{x}(t)\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{w}(t). \tag{2.72}$$

This is known as the *variance-preserving* SDE, or VP-SDE.

To see why this name is appropriate, consider the conditional distribution of $\mathbf{x}(t)$ given $\mathbf{x}(0) = \mathbf{x}_0$. Solving the mean and variance ODEs gives

$$p(\mathbf{x}(t)|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}(t); \alpha(t)\mathbf{x}_0, \left(1 - \alpha^2(t)\right)\mathbf{I}\right), \tag{2.73}$$

where

$$\alpha(t) = \exp\left(-\frac{1}{2}\int_0^t \beta(s)\,\mathrm{d}s\right). \tag{2.74}$$

Equivalently, the forward process can be written as

$$\mathbf{x}(t) = \alpha(t)\mathbf{x}_0 + \sqrt{1 - \alpha^2(t)}\,\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.75}$$

The conditional variance therefore satisfies

$$P(t) = 1 - \alpha^2(t) = 1 - \exp\left(-\int_0^t \beta(s)\,\mathrm{d}s\right). \tag{2.76}$$

More generally, the variance evolves according to

$$\frac{\mathrm{d}P(t)}{\mathrm{d}t} = -\beta(t)P(t) + \beta(t). \tag{2.77}$$

If the data are standardized so that the marginal variance of $\mathbf{x}(0)$ is $\mathbf{I}$, then the marginal variance of $\mathbf{x}(t)$ remains approximately $\mathbf{I}$ throughout the forward process. Hence, unlike the VE-SDE, whose variance grows monotonically with time, the DDPM forward process preserves the overall variance scale while gradually destroying the signal-to-noise ratio. This motivates the name *variance preserving*.

Elucidated Diffusion Models. Elucidated Diffusion Models (EDMs) [46] provide a unified and practical design framework for diffusion-based generative models. Rather than introducing a fundamentally new diffusion process, EDMs identify and refine several important design choices, including the noise parameterization, network preconditioning, loss weighting, noise-level sampling strategy, and numerical solver. This makes EDMs particularly useful as a bridge between score-based diffusion models and efficient sampling algorithms.

EDMs are most naturally connected to the VE-SDE formulation. Recall the Gaussian perturbation process

$$\mathbf{x}_\sigma = \mathbf{x}_0 + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2.78}$$

where σ directly parameterizes the noise level. Instead of learning the score function explicitly, EDMs train a denoising network $D_\theta(\mathbf{x}_\sigma; \sigma)$ to predict the clean sample $\mathbf{x}_0$ from the noisy observation $\mathbf{x}_\sigma$. The training objective takes the form

$$\mathcal{L}_{\mathrm{EDM}} = \mathbb{E}_{\mathbf{x}_0 \sim p_{\mathrm{data}},\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\, \sigma \sim p_{\mathrm{train}}(\sigma)}\left[\lambda(\sigma)\,\|D_\theta(\mathbf{x}_0 + \sigma\boldsymbol{\epsilon}; \sigma) - \mathbf{x}_0\|_2^2\right], \tag{2.79}$$

where $p_{\mathrm{train}}(\sigma)$ is the training distribution over noise levels and $\lambda(\sigma)$ is a noise-dependent loss weighting function.

The connection to score matching follows directly from Tweedie's formula. The optimal denoiser satisfies

$$D^*(\mathbf{x}_\sigma; \sigma) = \mathbb{E}[\mathbf{x}_0|\mathbf{x}_\sigma] = \mathbf{x}_\sigma + \sigma^2 \nabla_{\mathbf{x}_\sigma} \log p_\sigma(\mathbf{x}_\sigma). \tag{2.80}$$

Therefore, the score of the perturbed data distribution can be recovered from the denoiser as

$$\nabla_{\mathbf{x}_\sigma} \log p_\sigma(\mathbf{x}_\sigma) = \frac{D^*(\mathbf{x}_\sigma;\sigma) - \mathbf{x}_\sigma}{\sigma^2}. \tag{2.81}$$

In practice, replacing D^* with the learned denoiser gives

$$\mathbf{s}_\theta(\mathbf{x}_\sigma;\sigma) \approx \frac{D_\theta(\mathbf{x}_\sigma;\sigma) - \mathbf{x}_\sigma}{\sigma^2}. \tag{2.82}$$

Thus, although EDMs are trained as denoisers, they implicitly learn the score field required for diffusion-based sampling.

Using the VE probability flow ODE and taking σ itself as the time variable, the deterministic sampling dynamics become

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\sigma} = -\sigma \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}). \tag{2.83}$$

Substituting the denoiser-score relation yields the EDM sampling ODE:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\sigma} = \frac{\mathbf{x} - D_\theta(\mathbf{x};\sigma)}{\sigma}. \tag{2.84}$$

Sampling is then performed by initializing

$$\mathbf{x}_{\sigma_{\max}} \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I}), \tag{2.85}$$

and numerically solving the ODE from $\sigma_{\max}$ to $\sigma_{\min}$, followed by a final denoising step. With an Euler discretization, the update is

$$\mathbf{x}_{i+1} = \mathbf{x}_i + (\sigma_{i+1} - \sigma_i) \frac{\mathbf{x}_i - D_\theta(\mathbf{x}_i;\sigma_i)}{\sigma_i}. \tag{2.86}$$

Higher-order solvers, such as Heun's method, can further improve sample quality for the same number of function evaluations.

A key contribution of EDM is the use of network *preconditioning*. Instead of directly predicting the clean sample with an unconstrained network, the denoiser is parameterized as

$$D_\theta(\mathbf{x};\sigma) = c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma) F_\theta\left(c_{\text{in}}(\sigma)\mathbf{x}; c_{\text{noise}}(\sigma)\right), \tag{2.87}$$

where F_θ is the neural network and the coefficients are chosen as

$$c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2}{\sigma^2 + \sigma_{\text{data}}^2}, \qquad c_{\text{out}}(\sigma) = \frac{\sigma\sigma_{\text{data}}}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}, \tag{2.88}$$

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}, \qquad c_{\text{noise}}(\sigma) = \frac{1}{4}\log\sigma. \tag{2.89}$$

Here, σ_{data} denotes the standard deviation of the training data. These preconditioning coefficients stabilize training by ensuring that the input and output scales of the network remain well behaved across different noise levels.

EDM also proposes a noise discretization schedule of the form

$$\sigma_i = \left(\sigma_{\max}^{1/\rho} + \frac{i}{N-1}\left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^\rho, \qquad i = 0, \ldots, N-1, \tag{2.90}$$

where ρ controls how densely the solver steps are allocated across noise levels. Larger values of ρ allocate more steps to the low-noise regime, where fine image details are formed and the ODE dynamics are typically more sensitive.

Overall, EDMs can be understood as a carefully optimized realization of the VE diffusion framework. They retain the denoising-score connection of score-based models while introducing practical design choices that substantially improve training stability and sampling efficiency. This makes EDMs an important baseline for studying few-step and high-quality diffusion sampling, especially in image-generation tasks where both fidelity and computational cost are critical.

2.2 Flow Matching

The training of DDPMs and NCSNs is based on SDE models. As stated in Theorem 2.4, the PF-ODE can also be used to connect a base distribution and a target distribution. It is therefore natural to ask whether we can directly construct a training scheme based on the PF-ODE. Lipman et al. [21] proposed the *Flow Matching* (FM) framework to address this question. Historically, FM has a close connection to Normalizing Flows [10, 47, 48, 49] and their continuous-time counterparts, namely Neural ODEs [50]. Here, we do not discuss these connections in detail, but instead follow directly from the SDE view of diffusion-based generative models.

Recall the PF-ODE formulation:

$$\frac{\partial}{\partial t} p_t(\mathbf{x}) = -\nabla \cdot (\mathbf{v}(\mathbf{x}, t) p_t(\mathbf{x})), \qquad \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{v}(\mathbf{x}(t), t). \tag{2.91}$$

Notice that the first equation takes the form of a continuity equation. This structure is widely observed in physical systems that satisfy flux conservation. For example,

$$\text{Navier–Stokes:} \quad \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \qquad \text{Charge Conservation:} \quad \frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{J} = 0. \tag{2.92}$$

Therefore, the FM framework naturally corresponds to a transport process between two distributions. Importantly, the solution to this transport problem is not unique [21, 51, 22, 52, 53][2]. Based on this observation, FM proposes to train a neural network $\mathbf{v}_\theta(\mathbf{x}, t)$ by matching the velocity field that generates the desired probability path $\{p_t\}_{t\in[0,1]}$:

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, \mathbf{x}\sim p_t} \left[\|\mathbf{v}_\theta(\mathbf{x}, t) - \mathbf{u}_t(\mathbf{x})\|_2^2 \right], \tag{2.93}$$

where $\mathbf{u}_t(\mathbf{x})$ denotes the true marginal velocity field satisfying

$$\frac{\partial}{\partial t} p_t(\mathbf{x}) = -\nabla \cdot (\mathbf{u}_t(\mathbf{x}) p_t(\mathbf{x})). \tag{2.94}$$

However, directly evaluating $\mathbf{u}_t(\mathbf{x})$ is generally intractable because it depends on the marginal density p_t. Conditional Flow Matching (CFM) avoids this difficulty by defining a conditional probability path between samples from the base and target distributions.

Let $\mathbf{x}_0 \sim p_0$ denote a sample from the base distribution and $\mathbf{x}_1 \sim p_1$ denote a sample from the target data distribution. A simple conditional path is given by the linear interpolation (this is referred to as the *optimal-transport* (OT) path [21])

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \qquad t \in [0, 1]. \tag{2.95}$$

The corresponding conditional velocity is

$$\mathbf{u}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_t = \mathbf{x}_1 - \mathbf{x}_0. \tag{2.96}$$

Thus, instead of learning the intractable marginal velocity field directly, CFM trains the model using the conditional velocity:

$$\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, \mathbf{x}_0\sim p_0,\, \mathbf{x}_1\sim p_1} \left[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\|_2^2 \right]. \tag{2.97}$$

This objective is tractable because both $\mathbf{x}_t$ and its conditional velocity can be computed explicitly from sampled pairs $(\mathbf{x}_0, \mathbf{x}_1)$. The CFM loss then takes the general form

$$\mathcal{L}_{\mathrm{CFM}} = \mathbb{E} \left[\|\mathbf{v}_\theta(\mathbf{x}_t, t) - \mathbf{u}_t(\mathbf{x}_t|\mathbf{x}_1)\|_2^2 \right]. \tag{2.98}$$

[2] Interestingly, non-uniqueness is also a central problem in natural dynamical systems, with the Navier–Stokes equations being one of the most famous examples [54].

Here we formally show that the CFM loss is equivalent to the FM loss by a constant independent of model parametrization.

Theorem 2.5: Equivalence of CFM and FM

The following relation holds

$$\mathcal{L}_{\text{FM}} = \mathcal{L}_{\text{CFM}} + C, \tag{2.99}$$

where C is constant independent of the parametrization θ. The optimal network prediction for both CFM and FM loss satisfies

$$\mathbf{v}^*(\mathbf{x}_t, t) = \mathbf{v}(\mathbf{x}(t), t), \quad \forall \mathbf{x}_t \sim p_t. \tag{2.100}$$

Proof. It suffices to compare the terms that depend on θ. Expanding the squared norms gives

$$\begin{aligned} \|\mathbf{v}_\theta(\mathbf{x}, t) - \mathbf{u}_t(\mathbf{x})\|_2^2 &= \|\mathbf{v}_\theta(\mathbf{x}, t)\|_2^2 - 2\langle \mathbf{v}_\theta(\mathbf{x}, t), \mathbf{u}_t(\mathbf{x})\rangle + \|\mathbf{u}_t(\mathbf{x})\|_2^2, \\ \|\mathbf{v}_\theta(\mathbf{x}, t) - \mathbf{u}_t(\mathbf{x}|\mathbf{x}_1)\|_2^2 &= \|\mathbf{v}_\theta(\mathbf{x}, t)\|_2^2 - 2\langle \mathbf{v}_\theta(\mathbf{x}, t), \mathbf{u}_t(\mathbf{x}|\mathbf{x}_1)\rangle + \|\mathbf{u}_t(\mathbf{x}|\mathbf{x}_1)\|_2^2. \end{aligned} \tag{2.101}$$

The final terms in both expressions are independent of θ, so they can be ignored when optimizing over θ. Let

$$q_t(\mathbf{x}, \mathbf{x}_1) := p_t(\mathbf{x}|\mathbf{x}_1)p_1(\mathbf{x}_1),$$

whose marginal over $\mathbf{x}_1$ is $p_t(\mathbf{x})$. For the quadratic term, we have

$$\begin{aligned} \mathbb{E}_{\mathbf{x}\sim p_t}\left[\|\mathbf{v}_\theta(\mathbf{x}, t)\|_2^2\right] &= \int \|\mathbf{v}_\theta(\mathbf{x}, t)\|_2^2 \, p_t(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &= \iint \|\mathbf{v}_\theta(\mathbf{x}, t)\|_2^2 \, p_t(\mathbf{x}|\mathbf{x}_1)p_1(\mathbf{x}_1) \, \mathrm{d}\mathbf{x}_1 \, \mathrm{d}\mathbf{x} \\ &= \mathbb{E}_{(\mathbf{x},\mathbf{x}_1)\sim q_t}\left[\|\mathbf{v}_\theta(\mathbf{x}, t)\|_2^2\right]. \end{aligned} \tag{2.102}$$

For the cross term, using

$$\mathbf{u}_t(\mathbf{x}) = \mathbb{E}_{\mathbf{x}_1\sim p_t(\mathbf{x}_1|\mathbf{x})}\left[\mathbf{u}_t(\mathbf{x}|\mathbf{x}_1)\right], \qquad p_t(\mathbf{x}_1|\mathbf{x}) = \frac{p_t(\mathbf{x}|\mathbf{x}_1)p_1(\mathbf{x}_1)}{p_t(\mathbf{x})},$$

we obtain

$$\begin{aligned} \mathbb{E}_{\mathbf{x}\sim p_t}\left[\langle \mathbf{v}_\theta(\mathbf{x}, t), \mathbf{u}_t(\mathbf{x})\rangle\right] &= \int \left\langle \mathbf{v}_\theta(\mathbf{x}, t), \int \mathbf{u}_t(\mathbf{x}|\mathbf{x}_1)p_t(\mathbf{x}_1|\mathbf{x}) \, \mathrm{d}\mathbf{x}_1 \right\rangle p_t(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &= \iint \langle \mathbf{v}_\theta(\mathbf{x}, t), \mathbf{u}_t(\mathbf{x}|\mathbf{x}_1)\rangle \, p_t(\mathbf{x}|\mathbf{x}_1)p_1(\mathbf{x}_1) \, \mathrm{d}\mathbf{x}_1 \, \mathrm{d}\mathbf{x} \\ &= \mathbb{E}_{(\mathbf{x},\mathbf{x}_1)\sim q_t}\left[\langle \mathbf{v}_\theta(\mathbf{x}, t), \mathbf{u}_t(\mathbf{x}|\mathbf{x}_1)\rangle\right]. \end{aligned} \tag{2.103}$$

Therefore, the two objectives have identical θ-dependent quadratic and cross terms, and differ only by additive terms independent of θ. Hence they have the same minimizers with respect to θ. □

Therefore, CFM provides a practical way to train continuous-time generative models without explicitly simulating an SDE or estimating a score function.

After training, generation is performed by sampling $\mathbf{x}(0) \sim p_0$ and solving the learned ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = \mathbf{v}_\theta(\mathbf{x}(t), t), \qquad t : 0 \to 1. \tag{2.104}$$

The final sample $\mathbf{x}(1)$ is then expected to follow the target data distribution p_1. In this sense, FM replaces score estimation and reverse-time stochastic sampling with direct velocity-field learning and deterministic transport from noise to data.

2.3 One-step Generative Models

2.3.1 Consistency Models & Shortcut Diffusion

In Sections 2.1 and 2.2, we have discussed in detail the development and theoretical foundations of multi-step generative methods. In practical applications, such as image generation [55] and video generation [56], the backbone neural networks are typically very large, often containing several billion parameters. Consequently, repeatedly evaluating these networks during sampling can incur substantial computational costs, limiting their applicability in production settings. This motivates the question of whether diffusion models can be reformulated to reduce the number of required function evaluations.[3] In this section, we introduce two pioneering works in this regime: *Consistency Models* (CMs) and *Shortcut Models*.

Consistency Models. Song et al. [26] is generally considered the first work to study how to train a one-step generative model within the diffusion framework. The proposed method is known as the *Consistency Model*.

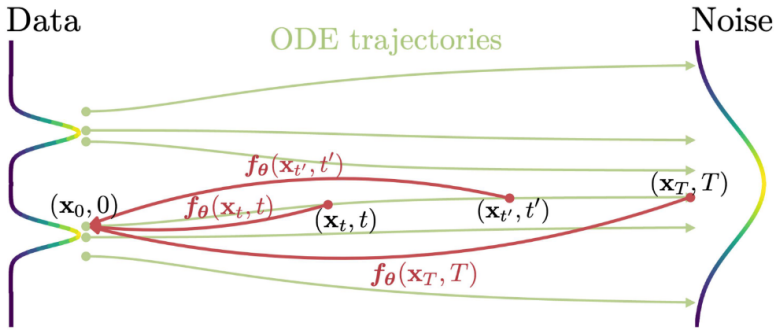

Figure 2.2: Plot adapted from Figure 2 of Song et al. [26]. The consistency function $f_\theta(\mathbf{x}_t, t)$ can be interpreted as an observable that remains invariant along a given PF-ODE trajectory connecting two distributions; see Theorem 2.4.

The central idea of consistency models is simple. Let $F_\theta(\mathbf{x}, t)$ denote the raw network output. We define a *consistency function* $\mathbf{f}(\mathbf{x}_t, t; \theta)$ along the trajectory $t \in [0, T]$ as

$$\mathbf{f}(\mathbf{x}_t, t; \theta) = c_{\text{skip}}(t)\mathbf{x}_t + c_{\text{out}}(t)F_\theta(\mathbf{x}_t, t), \tag{2.105}$$

where $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions satisfying

$$c_{\text{skip}}(0) = 1, \qquad c_{\text{out}}(0) = 0. \tag{2.106}$$

These boundary conditions ensure that the consistency function recovers the clean data at the endpoint $t = 0$. This parameterization is closely related to the EDM formulation discussed earlier, and practical implementations of CMs can therefore benefit from insights developed in prior work on EDMs.

Consistency models require the consistency function to remain invariant along a PF-ODE trajectory. In practice, this means that evaluating the function at two different points on the same trajectory should yield the same value:

$$\mathbf{f}(\mathbf{x}_t, t; \theta) = \mathbf{f}(\mathbf{x}_{t'}, t'; \theta). \tag{2.107}$$

This property is illustrated in Figure 2.2. Once trained, a consistency model can generate the desired noise-free sample in a single step by evaluating $\mathbf{f}(\mathbf{x}_T, T; \theta)$. The corresponding training objective can be written as

$$\mathcal{L}_{\text{CM}} = \mathbb{E}\left[\lambda(t_n) d\left(\mathbf{f}(\mathbf{x}_{t_n}, t_n; \theta), \mathbf{f}(\mathbf{x}_{t_{n+1}}, t_{n+1}; \theta^-)\right)\right], \tag{2.108}$$

[3] Historically, this question is closely related to model distillation, including paradigms such as Distribution Matching Distillation (DMD) [57] and Variational Score Distillation (VSD) [58]. These formulations also have counterparts in the one-step diffusion regime, but we do not discuss them here in order to keep the exposition simple.

where $d(\cdot,\cdot)$ is a distance function, $\lambda(t_n)$ is a weighting function, and θ^- denotes the exponential moving average of the model parameters. Simplified Consistency Models (sCMs) [27] further extend CMs by adopting a trigonometric parameterization of the probability path.

Shortcut Models. Consistency models constrain the model trajectory through a pointwise invariance condition imposed on the consistency function. *Shortcut Models*, proposed by Frans et al. [28], make this idea more explicit at the level of the transition operator. In general, let $\Psi_{t_0\to t_1}$ denote the push-forward operator that transports samples from diffusion time t_0 to t_1. Few-step generative models aim to satisfy the following operator-level equivalence:

$$\Psi_{t_0\to t_2} = \Psi_{t_1\to t_2} \circ \Psi_{t_0\to t_1}. \tag{2.109}$$

Rather than enforcing a pointwise consistency condition, shortcut models explicitly impose this equivalence over a finite interval. The training objective of shortcut models can be written as

$$\mathcal{L}_{\text{shortcut}} = \underbrace{\|\mathbf{v}_\theta(\mathbf{x}_t, t, 0) - \mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1)\|_2^2}_{\text{flow matching}} + \underbrace{\left\|\mathbf{v}_\theta(\mathbf{x}_t, t, 2d) - \mathbf{v}_{\text{target}}\right\|_2^2}_{\text{consistency}}, \tag{2.110}$$

where

$$\mathbf{v}_{\text{target}} = \frac{1}{2}\left[\mathbf{v}_\theta(\mathbf{x}_t, t, d) + \mathbf{v}_\theta(\mathbf{x}_{t+d}, t+d, d)\right], \qquad \mathbf{x}_{t+d} = \mathbf{x}_t + d\,\mathbf{v}_\theta(\mathbf{x}_t, t, d). \tag{2.111}$$

Here, the third argument of the network denotes the shortcut step size. Ideally, a well-trained shortcut model should be able to perform one-step sampling via

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{v}_\theta(\mathbf{x}_0, 0, 1)(1-0). \tag{2.112}$$

Thus, shortcut models impose a consistency constraint over a finite *line segment*, rather than only at individual points along the trajectory. This perspective will be useful for understanding the subsequent development of one-step and few-step generative models.

2.3.2 MeanFlow Models

MeanFlow (MF) [31] was introduced as a one-step alternative to standard optimal-transport flow matching. Rather than learning only the instantaneous velocity $\mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1)$, MeanFlow models the time-averaged velocity between diffusion times r and t:

$$\bar{\mathbf{u}}_{r,t}(\mathbf{x}_t \mid \mathbf{x}_1) = \frac{1}{t-r}\int_r^t \mathbf{u}_\tau(\mathbf{x}_\tau \mid \mathbf{x}_1)\,\mathrm{d}\tau. \tag{2.113}$$

This parameterization enables single-step transport from $\mathbf{x}_t$ to $\mathbf{x}_r$ via

$$\mathbf{x}_r = \mathbf{x}_t - (t-r)\bar{\mathbf{u}}_{r,t}(\mathbf{x}_t \mid \mathbf{x}_1). \tag{2.114}$$

Differentiating Eq. (2.113) gives the **MeanFlow identity**

$$\bar{\mathbf{u}}_{r,t}(\mathbf{x}_t \mid \mathbf{x}_1) = \mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1) - (t-r)\frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{u}}_{r,t}(\mathbf{x}_t \mid \mathbf{x}_1). \tag{2.115}$$

In practice, MeanFlow parameterizes the average velocity by a neural network $\bar{\mathbf{v}}_\theta(\mathbf{x}_t, r, t)$. The original MeanFlow objective can then be written as

$$\mathcal{L}_{\text{MF}} = \mathbb{E}_{t,r,\mathbf{x}_0,\mathbf{x}_1}\left[\left\|\bar{\mathbf{v}}_\theta(\mathbf{x}_t, r, t) - \text{sg}\big(\bar{\mathbf{u}}_{r,t}(\mathbf{x}_t \mid \mathbf{x}_1)\big)\right\|_2^2\right], \tag{2.116}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator.

Despite its conceptual simplicity, MeanFlow has been reported to exhibit unstable training dynamics. In particular, the training loss may increase even as sample quality improves [59, 60]. This phenomenon is largely attributed to samples near the flow-matching limit $t = r$, where MeanFlow degenerates to standard OT-FM. α-Flow [60] addresses this issue by introducing an annealing schedule that smoothly recovers the flow-matching objective. A more direct remedy was proposed by Geng et al. [59], who introduced improved MeanFlow (i-MF). Instead of directly supervising the interval-averaged velocity, i-MF uses Eq. (2.115) to construct an estimate of the instantaneous velocity:

$$\widehat{\mathbf{u}}_\theta(\mathbf{x}_t, r, t) \equiv \bar{\mathbf{v}}_\theta(\mathbf{x}_t, r, t) + (t - r)\ \mathrm{sg}\big(\partial_t \bar{\mathbf{v}}_\theta(\mathbf{x}_t, r, t) + \nabla_{\mathbf{x}_t} \bar{\mathbf{v}}_\theta(\mathbf{x}_t, r, t)\, \mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1)\big)\,. \tag{2.117}$$

The model is then trained with the objective

$$\mathcal{L}_{\text{i-MF}} = \mathbb{E}_{t,r,\mathbf{x}_0,\mathbf{x}_1}\left[\left\|\widehat{\mathbf{u}}_\theta(\mathbf{x}_t, r, t) - \mathbf{u}_t(\mathbf{x}_t \mid \mathbf{x}_1)\right\|_2^2\right]. \tag{2.118}$$

Both MF and i-MF were originally developed in the VAE latent space for 256×256 image generation. To remove the VAE bottleneck, Li and He [61] proposed the Just-image Transformer (JiT), enabling pixel-space image generation at resolutions up to 1024×1024. Building on this direction, Lu et al. [62] introduced pixel MeanFlow (p-MF), which reparameterizes the prediction target directly in pixel space. In the common case where the reference time is $r = 0$, this can be written as

$$\mathbf{X}_\theta(\mathbf{x}_t, r, t) = \mathbf{x}_t - t\,\bar{\mathbf{v}}_\theta(\mathbf{x}_t, r, t), \qquad \bar{\mathbf{v}}_\theta(\mathbf{x}_t, r, t) = \frac{1}{t}\left[\mathbf{x}_t - \mathbf{X}_\theta(\mathbf{x}_t, r, t)\right]. \tag{2.119}$$

The induced average velocity $\bar{\mathbf{v}}_\theta$ is then used within the i-MF objective. Overall, MeanFlow models extend shortcut diffusion models by imposing the consistency condition along an *entire trajectory* and by directly modeling the corresponding average velocity.

2.4 Diffusion Sampling Techniques

2.4.1 Solvers with Semilinear ODE

Following our discussion of the Score-SDE framework in Section 2.1.3, sampling from a diffusion model can be viewed as solving the inverse problem associated with the forward SDE. In this section, we focus on diffusion models, and in particular on the VP-SDE. The original DDPM sampling algorithm is directly related to the reverse-time SDE induced by the Fokker–Planck theorem (Theorem 2.4). However, DDPM sampling is extremely slow, since it typically requires a large number of discretization steps, often comparable to the number of forward diffusion steps, in order to reach the data distribution.

Since the probability-flow ODE (PF-ODE) has the same marginal distributions as the corresponding reverse-time SDE, it is natural to use well-developed ODE solvers to simulate the PF-ODE. Recall the forward diffusion kernel of DDPM:

$$\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{x} + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, \tag{2.120}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{x} \sim p_{\text{data}}$. For clarity, we define the *signal intensity* s_t and the *noise intensity* σ_t as

$$s_t := \sqrt{\bar{\alpha}_t}, \qquad \sigma_t := \sqrt{1 - \bar{\alpha}_t}. \tag{2.121}$$

Then the forward noising process can be written compactly as

$$\mathbf{z}_t = s_t\mathbf{x} + \sigma_t\boldsymbol{\epsilon}. \tag{2.122}$$

Taking the time derivative gives

$$\begin{aligned}\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} &= \dot{s}_t\mathbf{x} + \dot{\sigma}_t\boldsymbol{\epsilon} \\ &= \dot{s}_t\left(\frac{\mathbf{z}_t - \sigma_t\boldsymbol{\epsilon}}{s_t}\right) + \dot{\sigma}_t\boldsymbol{\epsilon}.\end{aligned} \tag{2.123}$$

Therefore, after replacing the true noise $\boldsymbol{\epsilon}$ by the denoiser prediction $\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t)$, we obtain

$$\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} = \underbrace{\frac{\dot{s}_t}{s_t}\mathbf{z}_t}_{\text{linear term}} + \underbrace{\left(\dot{\sigma}_t - \frac{\dot{s}_t}{s_t}\sigma_t\right)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t)}_{\text{nonlinear denoising term}}. \tag{2.124}$$

This equation is known as the *semilinear PF-ODE*, and it can be shown to be equivalent to the probability-flow ODE [63]. It forms the foundation of many ODE-based samplers, which we discuss below. Dividing both sides by s_t yields the simpler normalized form

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathbf{z}_t}{s_t}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\sigma_t}{s_t}\right)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t), \tag{2.125}$$

which will be useful for the derivations that follow.

2.4.2 Denoising Diffusion Implicit Models

Denoising Diffusion Implicit Models (DDIMs), proposed by Song, Meng, and Ermon [23], were among the first methods to interpret diffusion sampling through the lens of ODE-based solvers. Under the formulation above, DDIM can be viewed as a first-order method for discretizing the semilinear PF-ODE in Eq. (2.125). Specifically, DDIM adopts the approximation

$$\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t) \approx \hat{\boldsymbol{\epsilon}}_t, \tag{2.126}$$

that is, the noise prediction is held fixed over the interval. The normalized PF-ODE then gives

$$\begin{aligned}&\frac{\mathbf{z}_t}{s_t} - \frac{\mathbf{z}_{t-1}}{s_{t-1}} = \left(\frac{\sigma_t}{s_t} - \frac{\sigma_{t-1}}{s_{t-1}}\right)\hat{\boldsymbol{\epsilon}}_t, \\ \Rightarrow \quad &\mathbf{z}_{t-1} = \frac{s_{t-1}}{s_t}\mathbf{z}_t + \left(\sigma_{t-1} - \frac{s_{t-1}}{s_t}\sigma_t\right)\hat{\boldsymbol{\epsilon}}_t.\end{aligned} \tag{2.127}$$

Substituting $s_t = \sqrt{\bar{\alpha}_t}$ and $\sigma_t = \sqrt{1-\bar{\alpha}_t}$ yields

$$\mathbf{z}_{t-1} = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}\mathbf{z}_t + \left(\sqrt{1-\bar{\alpha}_{t-1}} - \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}\sqrt{1-\bar{\alpha}_t}\right)\hat{\boldsymbol{\epsilon}}_t, \tag{2.128}$$

which is precisely the deterministic DDIM update [23]. PF-ODE sampling can sometimes reduce sample diversity, since it produces deterministic trajectories once the initial noise is fixed. DDIM can be augmented with additional stochasticity to address this issue. Specifically, fresh Gaussian noise can be injected by decomposing the next-step noise component. The resulting update is

$$\mathbf{z}_{t-1} = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}\mathbf{z}_t + \left(\sqrt{1-\bar{\alpha}_{t-1}-\tilde{\sigma}_t^2} - \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}}\sqrt{1-\bar{\alpha}_t}\right)\hat{\boldsymbol{\epsilon}}_t + \tilde{\sigma}_t\boldsymbol{\xi}_t, \tag{2.129}$$

where $\boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and

$$\tilde{\sigma}_t = \eta\sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}}\sqrt{1-\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}. \tag{2.130}$$

Here, η controls the amount of stochasticity. The case $\eta = 0$ recovers the deterministic DDIM update, while larger values of η interpolate toward more stochastic sampling dynamics.

2.4.3 DPM++ Solvers

DPM-Solver [24] and its improved variant DPM++ [25] reformulate the semilinear PF-ODE in the logarithmic signal-to-noise ratio (log-SNR) space. Define the log-SNR as

$$\lambda_t := \log \frac{s_t}{\sigma_t}. \tag{2.131}$$

For reverse-time sampling from t to $t-1$, the signal-to-noise ratio increases, so

$$\lambda_{t-1} > \lambda_t, \qquad h_t := \lambda_{t-1} - \lambda_t > 0. \tag{2.132}$$

To see why this parameterization is useful, we rewrite the semilinear PF-ODE in terms of the **x-prediction** network. Recall that

$$\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t) = \frac{\mathbf{z}_t - s_t \mathbf{x}_\theta(\mathbf{z}_t, t)}{\sigma_t}. \tag{2.133}$$

Substituting this into the semilinear PF-ODE gives

$$\begin{aligned}
\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} &= \frac{\dot{s}_t}{s_t}\mathbf{z}_t + \left(\dot{\sigma}_t - \frac{\dot{s}_t}{s_t}\sigma_t\right)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t) \\
&= \frac{\dot{s}_t}{s_t}\mathbf{z}_t + \left(\dot{\sigma}_t - \frac{\dot{s}_t}{s_t}\sigma_t\right)\frac{\mathbf{z}_t - s_t\mathbf{x}_\theta(\mathbf{z}_t, t)}{\sigma_t} \\
&= \frac{\dot{\sigma}_t}{\sigma_t}\mathbf{z}_t + s_t\left(\frac{\dot{s}_t}{s_t} - \frac{\dot{\sigma}_t}{\sigma_t}\right)\mathbf{x}_\theta(\mathbf{z}_t, t) \\
&= \frac{\dot{\sigma}_t}{\sigma_t}\mathbf{z}_t + s_t\dot{\lambda}_t\mathbf{x}_\theta(\mathbf{z}_t, t).
\end{aligned} \tag{2.134}$$

Dividing both sides by σ_t yields

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathbf{z}_t}{\sigma_t}\right) &= \frac{s_t}{\sigma_t}\dot{\lambda}_t\mathbf{x}_\theta(\mathbf{z}_t, t) \\
&= e^{\lambda_t}\dot{\lambda}_t\mathbf{x}_\theta(\mathbf{z}_t, t).
\end{aligned} \tag{2.135}$$

Using the change of variables $\mathrm{d}\lambda = \dot{\lambda}_t\,\mathrm{d}t$, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}\lambda}\left(\frac{\mathbf{z}_\lambda}{\sigma_\lambda}\right) = e^{\lambda}\mathbf{x}_\theta(\mathbf{z}_\lambda, \lambda). \tag{2.136}$$

Integrating over the interval $[\lambda_t, \lambda_{t-1}]$ gives

$$\mathbf{z}_{t-1} = \frac{\sigma_{t-1}}{\sigma_t}\mathbf{z}_t + \sigma_{t-1}\int_{\lambda_t}^{\lambda_{t-1}} e^{\lambda}\mathbf{x}_\theta(\mathbf{z}_\lambda, \lambda)\,\mathrm{d}\lambda. \tag{2.137}$$

This exponential-integration form of the semilinear PF-ODE is the foundation of DPM-Solver and its variants [63, 24]. Different solvers primarily differ in how they approximate the integral term.

First-order DPM++ recovers DDIM. Lu et al. [24] showed that first-order DPM++ recovers the DDIM update. In this case, we use the same constant-prediction approximation as in DDIM:

$$\mathbf{x}_\theta(\mathbf{z}_\lambda, \lambda) \approx \hat{\mathbf{x}}_t, \qquad \lambda \in [\lambda_t, \lambda_{t-1}]. \tag{2.138}$$

The integral then becomes

$$\int_{\lambda_t}^{\lambda_{t-1}} e^{\lambda}\mathbf{x}_\theta(\mathbf{z}_\lambda, \lambda)\,\mathrm{d}\lambda = \left(e^{\lambda_{t-1}} - e^{\lambda_t}\right)\hat{\mathbf{x}}_t. \tag{2.139}$$

Substituting this into the exponential-integration formula gives

$$\mathbf{z}_{t-1} = \frac{\sigma_{t-1}}{\sigma_t}\mathbf{z}_t + s_{t-1}\left(1 - e^{-h_t}\right)\hat{\mathbf{x}}_t, \tag{2.140}$$

which is equivalent to the deterministic DDIM update in Eq. (2.128).

Second-order single-step DPM++: DPM++2S. The first-order DPM++ update freezes the data prediction $\mathbf{x}_\theta(\mathbf{z}_\lambda, \lambda)$ over the interval $[\lambda_t, \lambda_{t-1}]$. DPM++2S improves this approximation by introducing an intermediate point in log-SNR space and using it to estimate the first-order variation of the data prediction along the PF-ODE trajectory. Let

$$h_t = \lambda_{t-1} - \lambda_t, \qquad \lambda_\tau = \lambda_t + \rho h_t, \qquad \rho \in (0, 1), \tag{2.141}$$

where τ denotes the diffusion time corresponding to λ_τ. In practice, $\rho = \frac{1}{2}$ is commonly used. Starting from $\mathbf{z}_t$, DPM++2S first takes a first-order step to the intermediate point:

$$\mathbf{z}_\tau = \frac{\sigma_\tau}{\sigma_t}\mathbf{z}_t + s_\tau \left(1 - e^{-\rho h_t}\right) \hat{\mathbf{x}}_t, \qquad \hat{\mathbf{x}}_t = \mathbf{x}_\theta(\mathbf{z}_t, t). \tag{2.142}$$

The model is then evaluated again at this intermediate point,

$$\hat{\mathbf{x}}_\tau = \mathbf{x}_\theta(\mathbf{z}_\tau, \tau). \tag{2.143}$$

Using the finite difference

$$\frac{\hat{\mathbf{x}}_\tau - \hat{\mathbf{x}}_t}{\rho h_t} \tag{2.144}$$

as an estimate of the derivative of the data prediction with respect to λ, the second-order DPM++2S update is

$$\mathbf{z}_{t-1} = \frac{\sigma_{t-1}}{\sigma_t}\mathbf{z}_t + s_{t-1} \left(1 - e^{-h_t}\right) \left[\hat{\mathbf{x}}_t + \frac{1}{2\rho} \left(\hat{\mathbf{x}}_\tau - \hat{\mathbf{x}}_t\right)\right]. \tag{2.145}$$

Thus, DPM++2S can be viewed as a second-order Runge–Kutta-type exponential integrator in log-SNR space. Compared with DDIM, it uses one additional network evaluation per step to correct the constant-prediction approximation of the integral term.

Second-order multi-step DPM++: DPM++2M. DPM++2M avoids the extra intermediate network evaluation used by DPM++2S by reusing the model prediction from the previous sampling step. Suppose that, in addition to the current prediction

$$\hat{\mathbf{x}}_t = \mathbf{x}_\theta(\mathbf{z}_t, t), \tag{2.146}$$

we also have the previous prediction $\hat{\mathbf{x}}_{t+1} = \mathbf{x}_\theta(\mathbf{z}_{t+1}, t+1)$ stored from the preceding reverse-time step. Define

$$h_t = \lambda_{t-1} - \lambda_t, \qquad h_{t+1} = \lambda_t - \lambda_{t+1}, \qquad r_t = \frac{h_{t+1}}{h_t}. \tag{2.147}$$

Then a first-order finite-difference estimate of the variation of the data prediction is

$$\Delta_t = \frac{1}{r_t} \left(\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_{t+1}\right). \tag{2.148}$$

The DPM++2M update is then given by

$$\mathbf{z}_{t-1} = \frac{\sigma_{t-1}}{\sigma_t}\mathbf{z}_t + s_{t-1} \left(1 - e^{-h_t}\right) \left[\hat{\mathbf{x}}_t + \frac{1}{2}\Delta_t\right]. \tag{2.149}$$

This is a second-order Adams–Bashforth-type exponential integrator. Its main advantage is efficiency: after the first step, each update requires only one new network evaluation, because the previous data prediction is reused from the solver history. For a fixed number of function evaluations, DPM++2M can therefore take more, smaller steps than DPM++2S, which often makes it more stable and effective in low-NFE sampling regimes.

2.4.4 Diffusion Exponential Integrator Samplers

Diffusion Exponential Integrator Samplers (DEIS) [64] provide another principled family of ODE-based samplers for diffusion models. Similar to DPM-Solver, DEIS starts from the semilinear structure of the PF-ODE and integrates the linear part analytically. The key difference is that DEIS works naturally with the normalized variable

$$\rho_t := \frac{\sigma_t}{s_t} = e^{-\lambda_t}, \tag{2.150}$$

which is the noise-to-signal ratio. Recall from Eq. (2.125) that

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathbf{z}_t}{s_t}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\sigma_t}{s_t}\right)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, t). \tag{2.151}$$

Using the change of variables $\rho = \sigma_t/s_t$, this becomes

$$\frac{\mathrm{d}}{\mathrm{d}\rho}\left(\frac{\mathbf{z}_\rho}{s_\rho}\right) = \boldsymbol{\epsilon}_\theta(\mathbf{z}_\rho, \rho). \tag{2.152}$$

Therefore, the exact reverse-time update from t to $t-1$ can be written as

$$\mathbf{z}_{t-1} = \frac{s_{t-1}}{s_t}\mathbf{z}_t + s_{t-1}\int_{\rho_t}^{\rho_{t-1}} \boldsymbol{\epsilon}_\theta(\mathbf{z}_\rho, \rho)\,\mathrm{d}\rho. \tag{2.153}$$

Since reverse sampling moves toward lower noise levels, we have $\rho_{t-1} < \rho_t$. The remaining problem is therefore to approximate the integral of the neural network prediction along the trajectory.

DEIS approximates this integral by polynomial interpolation of the noise prediction. Suppose we have access to the current and previous model predictions

$$\hat{\boldsymbol{\epsilon}}_{t+j} = \boldsymbol{\epsilon}_\theta(\mathbf{z}_{t+j}, t+j), \qquad j = 0, \ldots, k-1. \tag{2.154}$$

Let $\ell_j(\rho)$ denote the Lagrange basis polynomial associated with the interpolation nodes $\{\rho_t, \rho_{t+1}, \ldots, \rho_{t+k-1}\}$. Then a k-th order DEIS update takes the form

$$\int_{\rho_t}^{\rho_{t-1}} \boldsymbol{\epsilon}_\theta(\mathbf{z}_\rho, \rho)\,\mathrm{d}\rho \approx \sum_{j=0}^{k-1} b_{t,j}\,\hat{\boldsymbol{\epsilon}}_{t+j}, \qquad b_{t,j} = \int_{\rho_t}^{\rho_{t-1}} \ell_j(\rho)\,\mathrm{d}\rho. \tag{2.155}$$

Substituting this approximation gives the general DEIS update

$$\mathbf{z}_{t-1} = \frac{s_{t-1}}{s_t}\mathbf{z}_t + s_{t-1}\sum_{j=0}^{k-1} b_{t,j}\,\hat{\boldsymbol{\epsilon}}_{t+j}. \tag{2.156}$$

The first-order case freezes the noise prediction over the interval:

$$\boldsymbol{\epsilon}_\theta(\mathbf{z}_\rho, \rho) \approx \hat{\boldsymbol{\epsilon}}_t. \tag{2.157}$$

This yields

$$\mathbf{z}_{t-1} = \frac{s_{t-1}}{s_t}\mathbf{z}_t + s_{t-1}\,(\rho_{t-1} - \rho_t)\,\hat{\boldsymbol{\epsilon}}_t. \tag{2.158}$$

Since $\rho_t = \sigma_t/s_t$, this can be rewritten as

$$\mathbf{z}_{t-1} = \frac{s_{t-1}}{s_t}\mathbf{z}_t + \left(\sigma_{t-1} - \frac{s_{t-1}}{s_t}\sigma_t\right)\hat{\boldsymbol{\epsilon}}_t, \tag{2.159}$$

which is exactly the deterministic DDIM update. Higher-order DEIS samplers improve this approximation by using the solver history to capture the variation of $\boldsymbol{\epsilon}_\theta$ along the trajectory. Thus, DEIS can be interpreted as a multistep exponential integrator for the semilinear PF-ODE, with DDIM appearing as its first-order special case.

2.5 Our Method

In this study, we investigate how advanced few-step generative models can be applied to astrophysical image generation. Our approach builds on Lizarraga et al. [33], whose model is based on DDPMs. The comparison against our method is illustrated in Figure 2.3, and the training & inference procedures are summarized in Algorithm 1, 2. In contrast, we adopt pixel-MeanFlow (p-MF), adapted from Lu et al. [32], together with an NCSN-style attention-augmented U-Net architecture adapted from Song and Ermon [40]. We re-implement the training and evaluation framework in `jax` [65] and `flax` [66].

Algorithm 1 pixel-MeanFlow Training

Require: dataset $\mathcal{D}$ (empirical distribution $\hat{p}_{\mathcal{D}}$), denoiser D_θ, iterations K, batch size B, learning rate η

Initialize θ
for $k = 1, \ldots, K$ **do**
  Sample $\{\mathbf{x}^{(i)}, c^{(i)}\}_{i=1}^{B} \sim \hat{p}_{\mathcal{D}}$, $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(0, I)$, $t^{(i)}$, $r^{(i)}$, $w^{(i)}$ ▷ w is CFG weight
  $\mathbf{z}_t^{(i)} \leftarrow (1 - t^{(i)})\mathbf{x}^{(i)} + t^{(i)}\boldsymbol{\epsilon}^{(i)}$ ▷ Flow Matching latent state
  # Sample Velocities
  $\mathbf{v}_{\theta,\,\text{uncond}}^{(i)} \leftarrow \frac{1}{t^{(i)}}\left[\mathbf{z}_t^{(i)} - D_\theta(\mathbf{z}_t^{(i)}, r^{(i)}, t^{(i)}, w^{(i)}, \varnothing)\right]$
  $\mathbf{v}_{\theta,\,\text{cond}}^{(i)} \leftarrow \frac{1}{t^{(i)}}\left[\mathbf{z}_t^{(i)} - D_\theta(\mathbf{z}_t^{(i)}, r^{(i)}, t^{(i)}, w^{(i)}, c^{(i)})\right]$
  $\mathbf{v}_{\theta,\,\text{guidance}}^{(i)} \leftarrow \left(\boldsymbol{\epsilon}^{(i)} - \mathbf{x}^{(i)}\right) + \left(1 - \frac{1}{w^{(i)}}\right)(\mathbf{v}_{\theta,\,\text{cond}}^{(i)} - \mathbf{v}_{\theta,\,\text{uncond}}^{(i)})$ ▷ No guidance at $w^{(i)} = 1$
  # MeanFlow Identity
  $\mathbf{V}_\theta^{(i)} \leftarrow \mathbf{v}_{\theta,\,\text{cond}}^{(i)} + (t^{(i)} - r^{(i)}) \cdot \text{sg}\left(\partial_t \mathbf{v}_{\theta,\,\text{cond}}^{(i)} + \mathbf{v}_{\theta,\,\text{cond}}^{(i)\,\top} \nabla_z \mathbf{v}_{\theta,\,\text{cond}}^{(i)}\right)$
  $\mathcal{L}_{\text{p-MF}} \leftarrow \frac{1}{B}\sum_{i=1}^{B}\left\|\mathbf{V}_\theta^{(i)} - \mathbf{v}_{\theta,\,\text{guidance}}^{(i)}\right\|_2^2$
  $\theta \leftarrow \theta - \eta\nabla_\theta\left(\mathcal{L}_{\text{p-MF}}\right)$
end for

Furthermore, we implement three state-of-the-art efficient sampling algorithms: DDIM with $\eta = 0$, DPM++2M [25], and an AB2-style DEIS sampler [64]. We evaluate all models and solvers on the GalaxiesML-64 dataset [5], following the protocol of Lizarraga et al. [33]. We also ablate the effectiveness of several key design choices proposed by Lizarraga et al. [33] in the context of p-MF.

Performance is assessed by comparing the distributional similarity between generated and observed astrophysical images using domain-specific morphological metrics, including ellipticity, semi-major axis, Sérsic index, and isophotal area. Through these experiments, we aim to provide a comprehensive qualitative and quantitative comparison, thereby helping to bridge the gap between practical applications of AI in astrophysics and recent advances at the frontier of generative modeling.

Algorithm 2 pixel-MeanFlow Inference

Require: trained models D_{θ^*}, condition $\{c^{(i)}\}_{i=1}^{B}$, guidance strength $\{w^{(i)}\}_{i=1}^{B}$

1: Sample $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(0, I)$

2: $\hat{\mathbf{x}}^{(i)} \leftarrow \boldsymbol{\epsilon}^{(i)} - \mathbf{v}_{\theta^*}^{(i)}(\boldsymbol{\epsilon}^{(i)}, 0, 1, w^{(i)}, c^{(i)})$

Return: generated images $\{\hat{\mathbf{x}}^{(i)}\}_{i=1}^{B}$.

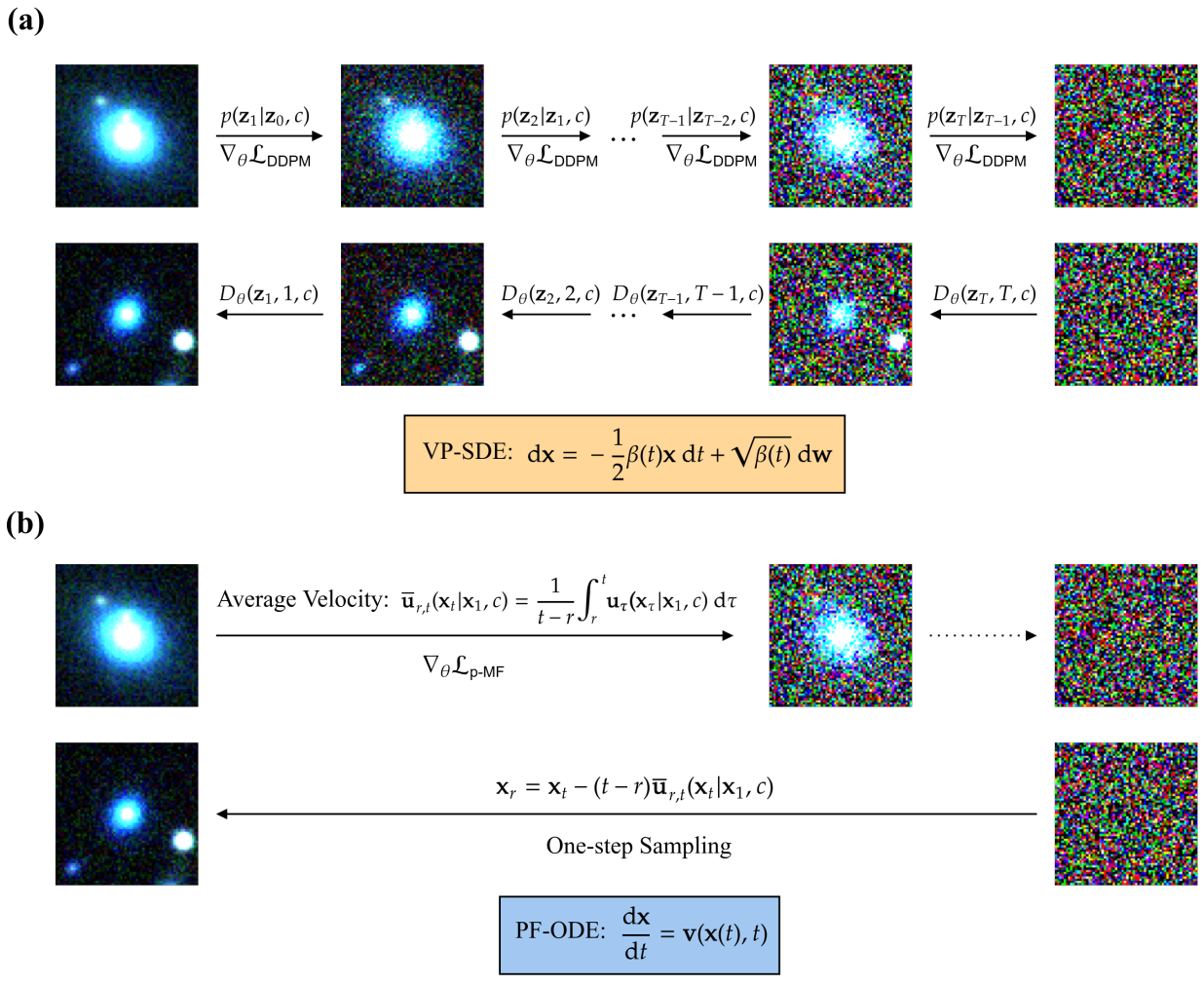

Figure 2.3: Comparison between our approach and that of Lizarraga et al. [33]. Panel (a) illustrates the DDPM training and inference process. The notation follows the main text, and the conditioning variable, which in our case is the continuous redshift, is denoted by c. Both DDPM training and sampling are inherently multi-step, with the sampling dynamics following the VP-SDE discussed above. Panel (b) illustrates pixel MeanFlow, where we directly model the average velocity induced by the PF-ODE, thereby enabling one-step inference.

Experiments

Baselines. In this section, we evaluate p-MF and DDPM models using four samplers: standard DDPM with 1000 sampling steps, DDIM with $\eta = 0$ and 60 sampling steps, DPM++2M with 30 sampling steps, and DEIS-AB2 with 30 sampling steps. The DDPM model with the standard DDPM sampler, implemented following Lizarraga et al. [33], serves as the baseline for comparison with our p-MF models. See Appendix A.1 for details.

Metrics. We assess generation quality using two types of metrics. First, we compute the Jensen–Shannon divergence (JSD) between the unconditional distributions of physical quantities measured from real and generated samples, including ellipticity, semi-major axis, Sérsic index, and isophotal area. Second, we evaluate the normalized Bin Center Differences (BCD) between real and generated samples under redshift-conditioned generation. These physical quantities are computed using the implementation of Lizarraga et al. [33]. All evaluations are performed on the official test split of the GalaxiesML-64 dataset [5]. See Appendix A.2 for details.

Results. The experimental results are summarized in Table 3.1. We report conditional binned distributional discrepancies (BCD) and unconditional Jensen–Shannon divergences (JSD) for four astrophysical morphology statistics: semi-major axis (SMA), Sérsic index (SI), isophotal area (IA), and ellipticity (Ell.). Lower values indicate better agreement between the generated and observed galaxy distributions.

Among diffusion-based methods, the standard DDPM sampler achieves the best overall performance. This is expected, since DDPM uses 1000 reverse steps and therefore most closely follows the original reverse diffusion process. It obtains the lowest unconditional JSD across all four quantities and substantially outperforms the accelerated samplers in conditional BCD, especially for SI, IA, and ellipticity. This indicates that, although the DDPM sampler is computationally expensive, it remains the strongest baseline in terms of distributional fidelity.

Table 3.1: Summary of the main model results. Uncertainty estimates obtained by bootstrapping are reported in parentheses.

Model Specs		**Conditional BCD↓**				**Unconditional JSD↓**			
Model Name	**NFE**	**SMA**	**SI**	**IA**	**Ell.**	**SMA**	**SI**	**IA**	**Ell.**
Diffusion Models									
DDPM	1000	0.249(40)	1.37(50)	0.414(54)	0.460(91)	0.0036(4)	0.0003(1)	0.0045(4)	0.0049(4)
DDIM ($\eta = 0$)	60	2.99(23)	11.3(8)	2.54(14)	6.83(38)	0.0305(16)	0.0062(6)	0.1832(33)	0.0779(23)
DEIS-AB2	30	0.368(75)	5.69(39)	1.94(11)	1.06(21)	0.0142(12)	0.0011(3)	0.0776(25)	0.0311(17)
DPM++2M	30	0.328(110)	5.80(45)	2.53(17)	1.06(29)	0.0497(19)	0.0033(5)	0.0682(22)	0.0227(14)
p-MF Models									
p-MF (25M, L_2)	1	0.462(87)	9.03(71)	2.18(6)	3.33(34)	0.0947(33)	0.0069(9)	0.2126(27)	0.0311(22)

For efficient diffusion sampling, the second-order solvers consistently improve over the first-order DDIM sampler. DDIM with $\eta = 0$ and 60 NFEs exhibits a clear degradation relative to standard DDPM, particularly in the conditional metrics. In contrast, DEIS-AB2 and DPM++2M use only 30 NFEs but achieve better results than DDIM on most quantities. For example, both second-order solvers substantially reduce the conditional BCD for SMA, SI, and ellipticity. DEIS-AB2 gives the best overall performance among the efficient solvers, achieving lower conditional BCD for SI and IA and lower unconditional JSD for SMA and SI, while DPM++2M performs slightly better on unconditional IA

and ellipticity. These results suggest that higher-order exponential-integrator-based solvers are more effective than first-order DDIM in the low-NFE regime.

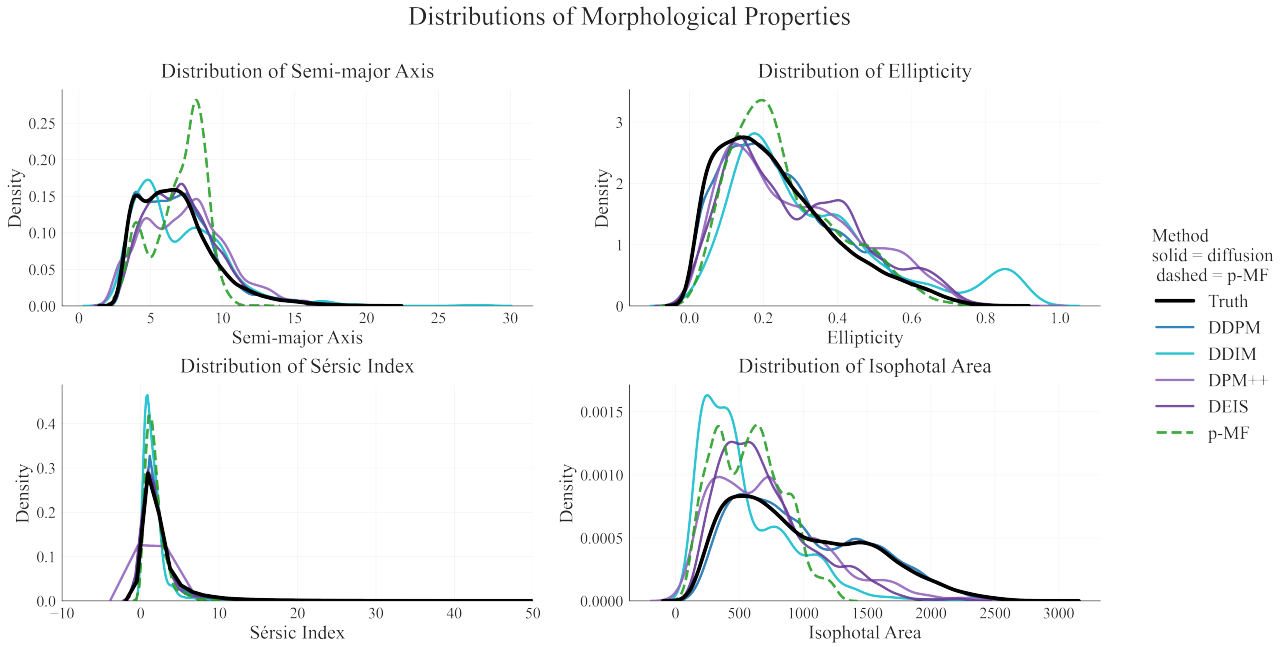

Figure 3.1: Unconditional morphological distributions for all models. The p-MF model is shown with green dashed lines.

We also compare the diffusion baselines against a 25M-parameter p-MF model, which requires only a single neural function evaluation at inference time. As expected, p-MF does not match the full 1000-step DDPM sampler in generation quality. Nevertheless, despite using only one inference step, p-MF achieves metrics that are competitive with accelerated diffusion samplers on several quantities. In particular, its conditional IA is comparable to DEIS-AB2 and better than DDIM and DPM++2M, while its unconditional ellipticity JSD is comparable to DEIS-AB2. The main degradation appears in the Sérsic index and unconditional SMA/IA distributions, suggesting that one-step p-MF captures coarse morphology reasonably well but still struggles with some fine-grained structural statistics.

The computational trade-off is shown in Table 3.2. Standard DDPM requires 33.5 seconds and 7020 GFLOPs for inference, making it substantially more expensive than all accelerated alternatives. DDIM reduces inference time to 2.6 seconds, while DEIS-AB2 and DPM++2M further reduce it to approximately 1.5–1.6 seconds. The 25M p-MF model achieves a comparable inference time of 1.5 seconds with only one network evaluation and a lower total GFLOP count than the 30-step diffusion solvers. The smaller 1M p-MF model further reduces inference time to 0.9 seconds and requires only 23.2 GFLOPs. Overall, these results highlight a clear accuracy–efficiency trade-off: standard DDPM provides the best fidelity, second-order diffusion solvers offer strong performance at much lower cost, and p-MF provides a promising one-step alternative with substantial computational savings. The comparison of unconditional distributions for different morphological parameters is give in Figure 3.1.

Ablation Studies. Lizarraga et al. [33] proposed two practical modifications for astrophysical diffusion models: label smoothing on the redshift conditioning variable and replacing the standard DDPM L_2 objective with the Huber loss. We investigate whether these design choices remain effective in the context of one-step p-MF models. We perform ablations at two model scales, 1M and 25M parameters. In Table 3.3, NLS denotes the variant without label smoothing.

The ablation results are shown in Table 3.3. For the 1M p-MF models, the effect of the Huber loss is mixed. Compared with the L_2 objective, Huber loss improves the conditional BCD for SMA and slightly improves the unconditional JSD for SI and ellipticity. However, it substantially worsens the IA metrics and also degrades the unconditional SMA distribution. The L_2 objective therefore provides

Table 3.2: Summary of inference speed (in seconds), GFLOPs, and model size for all models recorded on a single NVIDIA 4090 GPU. For diffusion models, the GFLOP count is computed by multiplying the single-step FLOP count by the number of solver steps. All samplers are compiled at the loop level using `jax.jit`.

Model Name	NFE	Size	Inference Speed	GFLOPs
Diffusion Models				
DDPM	1000	21.4M	33.5(6)	7020
DDIM ($\eta = 0$)	60	21.4M	2.6(4)	421
DEIS-AB2	30	21.4M	1.6(3)	197
DPM++2M	30	21.4M	1.5(4)	198
p-MF Models				
p-MF (1M)	1	1.26M	0.9(2)	23.2
p-MF (25M)	1	25.2M	1.5(1)	154

the most balanced performance at the 1M scale, achieving the best conditional IA and ellipticity, as well as the best unconditional SMA and IA.

Removing label smoothing also has non-uniform effects. In the 1M setting, comparing the Huber and Huber–NLS variants shows that removing label smoothing improves conditional SI, conditional IA, and unconditional ellipticity, but significantly worsens conditional SMA. This suggests that label smoothing may help stabilize the conditioning signal for certain morphology statistics, while potentially oversmoothing others.

At the larger 25M scale, the trade-off becomes clearer. The 25M p-MF model trained with the L_2 objective achieves the best conditional BCD across all four quantities, indicating that label smoothing is beneficial for redshift-conditioned generation. In contrast, the 25M model without label smoothing obtains better unconditional JSD across all four quantities. This indicates that removing label smoothing improves the marginal realism of generated samples, but at the cost of weaker conditional alignment.

Overall, these ablations suggest that design choices originally proposed for DDPMs do not transfer uniformly to p-MF models. The Huber loss improves some individual morphology statistics but does not consistently outperform the simpler L_2 objective. Label smoothing appears more useful for conditional generation, particularly at larger model scale, whereas removing it can improve unconditional distributional matching. Therefore, in our main comparison, we use the 25M p-MF model trained with the L_2 objective as the default setting, since it provides the strongest conditional performance and the best overall trade-off for redshift-conditioned galaxy generation.

Table 3.3: Ablation study of p-MF models with different loss functions and redshift-conditioning strategies. NLS denotes training without label smoothing. Uncertainty estimates obtained by bootstrapping are reported in parentheses.

Model Specs		Conditional BCD↓				Unconditional JSD↓			
Model Name	**NFE**	**SMA**	**SI**	**IA**	**Ell.**	**SMA**	**SI**	**IA**	**Ell.**
1M p-MF Models									
p-MF (1M, Huber, NLS)	1	1.47(4)	**10.1**(6)	2.17(4)	3.78(11)	0.0749(15)	0.0050(4)	0.2428(26)	**0.0313**(11)
p-MF (1M, L_2)	1	0.864(45)	11.4(6)	**1.98**(5)	**3.55**(14)	**0.0282**(14)	0.0041(5)	**0.1520**(28)	0.0486(19)
p-MF (1M, Huber)	1	**0.597**(24)	10.4(5)	3.16(4)	3.75(12)	0.0806(16)	**0.0039**(4)	0.3140(28)	0.0345(12)
25M p-MF Models									
p-MF (25M, L_2)	1	**0.462**(87)	**9.03**(71)	**2.18**(6)	**3.33**(34)	0.0947(33)	0.0069(9)	0.2126(27)	0.0311(22)
p-MF (25M, L_2, NLS)	1	0.780(49)	9.99(67)	2.41(6)	3.55(18)	**0.0748**(25)	**0.0045**(6)	**0.2022**(28)	**0.0293**(16)

Conclusion

Conclusion. In this work, we studied efficient redshift-conditioned generative modeling for galaxy morphology using diffusion models and pixel-MeanFlow. We reviewed the theoretical connections between score-based diffusion, Flow Matching, one-step generative models, and modern diffusion samplers, and evaluated these methods on the GalaxiesML-64 dataset.

Our results show a clear accuracy–efficiency trade-off. The 1000-step DDPM sampler achieves the best overall morphology fidelity, but at high computational cost. Efficient samplers such as DEIS-AB2 and DPM++2M substantially reduce inference time while outperforming first-order DDIM in the low-NFE regime. The one-step p-MF model does not yet match full DDPM quality, but achieves competitive performance on several morphology statistics with only a single neural function evaluation. These results suggest that one-step and few-step generative models are promising tools for fast astrophysical image generation.

Limitations and Future Work. This study has several limitations. First, our experiments are restricted to low-resolution GalaxiesML-64 images, whereas real survey data include higher resolution, heterogeneous noise, point-spread-function effects, and selection biases. Second, our evaluation uses only a small set of morphology statistics, which may not fully capture image realism or physical consistency. Future work should include additional diagnostics, multi-band information, downstream scientific tasks, and expert visual assessment.

The current p-MF model also remains weaker than many-step DDPM sampling, especially for fine-grained structural statistics such as the Sérsic index. Improving one-step generation may require larger models, longer training, improved MeanFlow objectives, or a small number of refinement steps. Finally, conditioning only on redshift limits the physical control of the model. Extending the framework to include additional galaxy properties such as stellar mass, star-formation rate, and environment would enable more detailed studies of galaxy evolution.

Acknowledgments. The author sincerely thanks Dr. Xue Xiao at the Center for Computational Science, University College London, for providing computational resources and valuable advice. Large language models were used only to improve the text and logical flow of this article.

Bibliography

[1] Christopher P. Ahn et al. "The Ninth Data Release of the Sloan Digital Sky Survey: First Spectroscopic Data from the SDSS-III Baryon Oscillation Spectroscopic Survey". In: *The Astrophysical Journal Supplement Series* 203.2, 21 (Dec. 2012), p. 21. DOI: 10.1088/0067-0049/203/2/21. arXiv: 1207.7137 [astro-ph.IM].

[2] T. M. C. Abbott et al. "Dark Energy Survey Year 3 results: Cosmological constraints from galaxy clustering and weak lensing". In: *Phys. Rev. D* 105 (2 2022), p. 023520. DOI: 10.1103/PhysRevD.105.023520. URL: https://link.aps.org/doi/10.1103/PhysRevD.105.023520.

[3] K. Kuijken et al. "The fourth data release of the Kilo-Degree Survey:ugriimaging and nine-band optical-IR photometry over 1000 square degrees". In: *Astronomy & Astrophysics* 625 (Apr. 2019), A2. ISSN: 1432-0746. DOI: 10.1051/0004-6361/201834918. URL: http://dx.doi.org/10.1051/0004-6361/201834918.

[4] Hiroaki Aihara et al. "Second data release of the Hyper Suprime-Cam Subaru Strategic Program". In: *Publications of the Astronomical Society of Japan* 71.6, 114 (Dec. 2019), p. 114. DOI: 10.1093/pasj/psz103. arXiv: 1905.12221 [astro-ph.IM].

[5] Tuan Do et al. *GalaxiesML: a dataset of galaxy images, photometry, redshifts, and structural parameters for machine learning*. 2024. arXiv: 2410.00271 [astro-ph.CO]. URL: https://arxiv.org/abs/2410.00271.

[6] Tri Nguyen et al. *How DREAMS are made: Emulating Satellite Galaxy and Subhalo Populations with Diffusion Models and Point Clouds*. 2024. arXiv: 2409.02980 [astro-ph.GA]. URL: https://arxiv.org/abs/2409.02980.

[7] E. Lastufka et al. "Examining vision foundation models for classification and detection in optical and radio astronomy". In: *Astronomy &; Astrophysics* 703 (Nov. 2025), A217. ISSN: 1432-0746. DOI: 10.1051/0004-6361/202553691. URL: http://dx.doi.org/10.1051/0004-6361/202553691.

[8] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[9] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[10] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[12] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[13] Zhihan Gao et al. "Prediff: Precipitation nowcasting with latent diffusion models". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 78621–78656.

[14] Bi Kaifeng et al. “Pangu-Weather: A 3D High-Resolution System for Fast and Accurate Global Weather Forecast”. In: *Nature, https://doi. org/10.1038/s41586-023-06185-3* (2023).

[15] Tianyue Yang and Xiao Xue. “MENO: MeanFlow-Enhanced Neural Operators for Dynamical Systems”. In: *arXiv preprint arXiv:2604.06881* (2026).

[16] Xiao Xue et al. “Uni-Flow: a unified autoregressive-diffusion model for complex multiscale flows”. In: *arXiv preprint arXiv:2602.15592* (2026).

[17] Salva Rühling Cachay et al. “Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting”. In: *Advances in neural information processing systems* 36 (2023), pp. 45259–45287.

[18] David Ruhe et al. “Rolling diffusion models”. In: *arXiv preprint arXiv:2402.09470* (2024).

[19] Salva Rühling Cachay et al. “Elucidated Rolling Diffusion Models for Probabilistic Forecasting of Complex Dynamics”. In: *arXiv preprint arXiv:2506.20024* (2025).

[20] Xiaoshan Luo et al. “CrystalFlow: a flow-based generative model for crystalline materials”. In: *Nature Communications* 16.1 (2025), p. 9267.

[21] Yaron Lipman et al. “Flow matching for generative modeling”. In: *arXiv preprint arXiv:2210.02747* (2022).

[22] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. “Stochastic interpolants: A unifying framework for flows and diffusions”. In: *arXiv preprint arXiv:2303.08797* (2023).

[23] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising diffusion implicit models”. In: *arXiv preprint arXiv:2010.02502* (2020).

[24] Cheng Lu et al. *DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps*. 2022. arXiv: 2206.00927 [cs.LG]. URL: https://arxiv.org/abs/2206.00927.

[25] Cheng Lu et al. “Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models”. In: *Machine Intelligence Research* 22.4 (2025), pp. 730–751.

[26] Yang Song et al. “Consistency models”. In: (2023).

[27] Cheng Lu and Yang Song. “Simplifying, stabilizing and scaling continuous-time consistency models”. In: *arXiv preprint arXiv:2410.11081* (2024).

[28] Kevin Frans et al. “One step diffusion via shortcut models”. In: *arXiv preprint arXiv:2410.12557* (2024).

[29] Haitao Lin et al. *On the Design of One-step Diffusion via Shortcutting Flow Paths*. 2026. arXiv: 2512.11831 [cs.LG]. URL: https://arxiv.org/abs/2512.11831.

[30] Linqi Zhou, Stefano Ermon, and Jiaming Song. “Inductive moment matching”. In: *arXiv preprint arXiv:2503.07565* (2025).

[31] Zhengyang Geng et al. “Mean flows for one-step generative modeling”. In: *arXiv preprint arXiv:2505.13447* (2025).

[32] Yiyang Lu et al. *One-step Latent-free Image Generation with Pixel Mean Flows*. 2026. arXiv: 2601.22158 [cs.CV]. URL: https://arxiv.org/abs/2601.22158.

[33] Andrew Lizarraga et al. *Understanding Galaxy Morphology Evolution Through Cosmic Time via Redshift Conditioned Diffusion Models*. 2025. arXiv: 2411.18440 [astro-ph.GA]. URL: https://arxiv.org/abs/2411.18440.

[34] G. E. Hinton and R. R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647. eprint: https://www.science.org/doi/pdf/10.1126/science.1127647. URL: https://www.science.org/doi/abs/10.1126/science.1127647.

[35] Jehoshua Bruck. “On the Convergence Properties of the Hopfield Model”. In: *Proceedings of the IEEE* 78.10 (Oct. 1990). DOI: 10.1109/5.58341. URL: https://doi.org/10.1109/5.58341.

[36] Aapo Hyvärinen and Peter Dayan. “Estimation of non-normalized statistical models by score matching.” In: *Journal of Machine Learning Research* 6.4 (2005).

[37] Yang Song et al. “Sliced score matching: A scalable approach to density and score estimation”. In: *Uncertainty in artificial intelligence*. PMLR. 2020, pp. 574–584.

[38] Pascal Vincent. “A connection between score matching and denoising autoencoders”. In: *Neural computation* 23.7 (2011), pp. 1661–1674.

[39] Bradley Efron. “Tweedie’s formula and selection bias”. In: *Journal of the American Statistical Association* 106.496 (2011), pp. 1602–1614.

[40] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019).

[41] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *arXiv preprint arXiv:2011.13456* (2020).

[42] Irina Higgins et al. “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *International Conference on Learning Representations*. 2016. URL: https://api.semanticscholar.org/CorpusID:46798026.

[43] Chris Cremer, Xuechen Li, and David Duvenaud. “Inference suboptimality in variational autoencoders”. In: *International conference on machine learning*. PMLR. 2018, pp. 1078–1086.

[44] Arash Vahdat and Jan Kautz. “NVAE: A deep hierarchical variational autoencoder”. In: *Advances in neural information processing systems* 33 (2020), pp. 19667–19679.

[45] Dar’ya Spivakovs’ka. “Reverse-time diffusion in environmental models”. In: *Delft University of Technology* (2007).

[46] Tero Karras et al. “Elucidating the design space of diffusion-based generative models”. In: *Advances in neural information processing systems* 35 (2022), pp. 26565–26577.

[47] Seunghun Lee et al. “Latent bayesian optimization via autoregressive normalizing flows”. In: *arXiv preprint arXiv:2504.14889* (2025).

[48] Clemens Arndt and Judith Nickel. “Invertible ResNets for Inverse Imaging Problems: Competitive Performance with Provable Regularization Properties”. In: *SIAM Journal on Imaging Sciences* 19.1 (2026), pp. 266–301.

[49] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. “i-revnet: Deep invertible networks”. In: *arXiv preprint arXiv:1802.07088* (2018).

[50] Ricky TQ Chen et al. “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31 (2018).

[51] Xingchao Liu, Chengyue Gong, and Qiang Liu. “Flow straight and fast: Learning to generate and transfer data with rectified flow”. In: *arXiv preprint arXiv:2209.03003* (2022).

[52] Alexander Tong et al. “Improving and generalizing flow-based generative models with minibatch optimal transport”. In: *arXiv preprint arXiv:2302.00482* (2023).

[53] Valentin De Bortoli et al. “Diffusion schrödinger bridge with applications to score-based generative modeling”. In: *Advances in neural information processing systems* 34 (2021), pp. 17695–17709.

[54] Matei P Coiculescu and Stan Palasek. “Non-uniqueness of smooth solutions of the Navier–Stokes equations from critical data”. In: *Inventiones mathematicae* 244.1 (2026), pp. 165–219.

[55] Patrick Esser et al. “Scaling rectified flow transformers for high-resolution image synthesis”. In: *Forty-first international conference on machine learning*. 2024.

[56] Andreas Blattmann et al. “Stable video diffusion: Scaling latent video diffusion models to large datasets”. In: *arXiv preprint arXiv:2311.15127* (2023).

[57] Tianwei Yin et al. “One-step diffusion with distribution matching distillation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 6613–6623.

[58] Zhengyi Wang et al. “Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation”. In: *Advances in neural information processing systems* 36 (2023), pp. 8406–8441.

[59] Zhengyang Geng et al. “Improved Mean Flows: On the Challenges of Fastforward Generative Models”. In: *arXiv preprint arXiv:2512.02012* (2025).

[60] Huijie Zhang et al. “Alphaflow: Understanding and improving meanflow models”. In: *arXiv preprint arXiv:2510.20771* (2025).

[61] Tianhong Li and Kaiming He. “Back to basics: Let denoising generative models denoise”. In: *arXiv preprint arXiv:2511.13720* (2025).

[62] Yiyang Lu et al. “One-step Latent-free Image Generation with Pixel Mean Flows”. In: *arXiv preprint arXiv:2601.22158* (2026).

[63] Chieh-Hsin Lai et al. “The principles of diffusion models”. In: *arXiv preprint arXiv:2510.21890* (2025).

[64] Qinsheng Zhang and Yongxin Chen. “Fast sampling of diffusion models with exponential integrator”. In: *arXiv preprint arXiv:2204.13902* (2022).

[65] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/jax-ml/jax.

[66] Jonathan Heek et al. *Flax: A neural network library and ecosystem for JAX*. Version 0.12.7. 2024. URL: http://github.com/google/flax.

[67] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[68] Priya Goyal et al. “Accurate, large minibatch sgd: Training imagenet in 1 hour”. In: *arXiv preprint arXiv:1706.02677* (2017).

[69] Yang Song and Stefano Ermon. “Improved techniques for training score-based generative models”. In: *Advances in neural information processing systems* 33 (2020), pp. 12438–12448.

Appendix

A.1 Experimental Setting

Optimization Setting. The training setup for the diffusion model follows Lizarraga et al. [33]. We use the Adam optimizer [67, 68] with a learning rate of 5×10^{-5}. We also use the same exponential moving average (EMA) configuration as Lizarraga et al. [33]. The diffusion model is trained on a single NVIDIA A100 80GB GPU for approximately 12 hours. For the p-MF models, we train the 1M-parameter version for 170 epochs on a single A100 GPU. For cost-matching purposes, we train the 25M-parameter version for 45 epochs on 8 NVIDIA GH200 GPUs for 8 hours. All other optimization settings are kept the same as those used for the diffusion models. The hyperparameters are summarized in Table A.1.

Table A.1: Optimization setting, same as Lizarraga et al. [33].

Name	Learning Rate	EMA weight	Gradient Clipping	Learning Rate Schedule
Values	5×10^{-5}	0.995	No	constant

Diffusion & MeanFlow Model Setting. The diffusion model setup follows Lizarraga et al. [33], which is based on Ho, Jain, and Abbeel [11]. The configuration is summarized in Table A.2. The

Table A.2: Diffusion model setting, same as Lizarraga et al. [33].

Name	β_{start}	β_{end}	Diffusion Steps	Loss	Classifier Free Guidance
Values	0.0001	0.02	1000	Huber	No

p-MF model settings follow the ImageNet-256 experimental configuration of Lu et al. [32]. The configuration is summarized in Table A.3.

Table A.3: pixel-MeanFlow model setting, same as Lu et al. [32].

Name	t, r Sampler	w_{max}	$t \neq r$ Percentage	Label Dropout Rate
Values	logit-normal(0.8, 0.8)	7	50%	0.1

Model Architecture. The diffusion model backbone is a `jax`-reimplementation of the UNet used by Lizarraga et al. [33]. It consists of three layers in each downsampling, bottleneck, and upsampling block, and does not include attention or residual blocks. The p-MF backbone is adapted from the NCSN [40]/NCSN++ [69] style UNet. It includes residual blocks at each resolution level and incorporates a self-attention block near the bottleneck.

A.2 Metrics

A.2.1 Redshift-binned Morphology Difference

To evaluate whether generated samples reproduce the redshift dependence of key morphological summary statistics, we compute a **Bin Center Difference** (BCD) between generated and reference measurements. The metric is evaluated independently for each morphology statistic

$$q \in \{\texttt{semi\_major\_axis}, \texttt{ellipticity}, \texttt{isophotal\_area}, \texttt{sersic\_index}\}.$$

Let

$$\{(r_i, q_i, \hat{q}_i)\}_{i=1}^{N}$$

denote the redshift, reference morphology value, and generated morphology value for the ith matched sample, respectively.

Redshift binning. We partition the redshift interval $[0, 4]$ into eight equally spaced bins,

$$0 = e_0 < e_1 < \cdots < e_8 = 4, \tag{A.1}$$

with bin edges given by

$$e_k = \frac{k}{2}, \qquad k = 0, \ldots, 8. \tag{A.2}$$

For bin k, the corresponding sample set is

$$\mathcal{I}_k = \{i : e_k \le r_i < e_{k+1}\}, \tag{A.3}$$

with the final bin closed on the right, so that samples with $r_i = e_8$ are included.

Validity filtering. Before computing the metric, we discard samples with non-finite redshift, reference value, or generated value. We also discard samples for which either the reference or generated morphology value is negative. Thus the valid index set is

$$\mathcal{V} = \{i : r_i, q_i, \hat{q}_i \text{ are finite}, \; q_i \ge 0, \; \hat{q}_i \ge 0\}. \tag{A.4}$$

All bin averages and normalization factors are computed only over samples in $\mathcal{V}$.

Binned morphology means. For each redshift bin, we compute the reference and generated bin centers as the empirical means

$$\mu_k^{\text{ref}} = \frac{1}{|\mathcal{I}_k \cap \mathcal{V}|} \sum_{i \in \mathcal{I}_k \cap \mathcal{V}} q_i, \tag{A.5}$$

and

$$\mu_k^{\text{gen}} = \frac{1}{|\mathcal{I}_k \cap \mathcal{V}|} \sum_{i \in \mathcal{I}_k \cap \mathcal{V}} \hat{q}_i. \tag{A.6}$$

Bins containing no valid samples are ignored when computing the final discrepancy.

Summed Bin Center Difference. The Bin Center Difference is defined as the sum of absolute differences between generated and reference bin-center means:

$$\mathcal{E}_{\text{BCD}}(q) = \sum_{k \in \mathcal{K}} \left| \mu_k^{\text{gen}} - \mu_k^{\text{ref}} \right|, \tag{A.7}$$

where $\mathcal{K}$ denotes the set of redshift bins for which both bin means are finite. Lower values indicate better agreement between the generated and reference redshift-conditioned morphology trends.

Normalized Bin Center Difference. To make the metric comparable across morphology statistics with different physical scales, we report a normalized Bin Center Difference,

$$\mathcal{E}_{\mathrm{nBCD}}(q) = \frac{\mathcal{E}_{\mathrm{BCD}}(q)}{\bar{q}_{\mathrm{ref}}}, \tag{A.8}$$

where

$$\bar{q}_{\mathrm{ref}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} q_i \tag{A.9}$$

is the global mean of the reference morphology statistic after validity filtering. If $\bar{q}_{\mathrm{ref}}$ is zero or non-finite, the normalized metric is undefined.

Bootstrap uncertainty. We estimate uncertainty in the normalized BCD using nonparametric bootstrapping. Given N_{boot} bootstrap replicates, each replicate is formed by sampling $|\mathcal{V}|$ matched triples $(r_i, q_i, \hat{q}_i)$ with replacement from the valid sample set. For bootstrap replicate s, we compute

$$\mathcal{E}^{(s)}_{\mathrm{nBCD}}(q), \qquad s = 1, \ldots, N_{\mathrm{boot}}. \tag{A.10}$$

In our implementation, we use $N_{\mathrm{boot}} = 1000$ bootstrap samples with a fixed random seed for reproducibility. We report the observed normalized BCD together with the bootstrap mean and standard deviation.

A.2.2 Jensen-Shannon Divergence

To compare one-dimensional distributions of generated and reference summary statistics, we compute the **Jensen-Shannon Divergence** (JSD). Unlike the Kullback-Leibler divergence, the Jensen-Shannon divergence is symmetric and remains well defined when comparing two normalized discrete probability distributions with shared support.

Let

$$p, q \in \mathbb{R}^D$$

denote two nonnegative vectors, such as histograms or discretized empirical distributions of a reference and generated statistic. Before computing the divergence, both vectors are normalized to sum to one:

$$\tilde{p}_i = \frac{p_i + \varepsilon}{\sum_{j=1}^{D} (p_j + \varepsilon)}, \qquad \tilde{q}_i = \frac{q_i + \varepsilon}{\sum_{j=1}^{D} (q_j + \varepsilon)}, \tag{A.11}$$

where $\varepsilon \geq 0$ is an optional small constant added before normalization.

Mixture distribution. The midpoint distribution is defined as

$$m_i = \frac{1}{2} (\tilde{p}_i + \tilde{q}_i) . \tag{A.12}$$

Kullback-Leibler divergence. For two normalized discrete distributions a and b, the Kullback-Leibler divergence is

$$D_{\mathrm{KL}}(a \,\|\, b) = \sum_{i : a_i > 0} a_i \log_b \left(\frac{a_i}{b_i} \right), \tag{A.13}$$

where the sum is taken only over entries with $a_i > 0$. In our implementation, the logarithm base is set to $b = 2$, so the divergence is measured in bits.

Jensen-Shannon divergence. The Jensen-Shannon divergence is then computed as

$$D_{\mathrm{JS}}(\tilde{p}, \tilde{q}) = \frac{1}{2} D_{\mathrm{KL}}(\tilde{p} \,\|\, m) + \frac{1}{2} D_{\mathrm{KL}}(\tilde{q} \,\|\, m). \tag{A.14}$$

Lower values indicate closer agreement between the generated and reference distributions. A value of zero indicates identical normalized distributions.

Validity conditions. Both input vectors must be one-dimensional, have the same length, contain only nonnegative entries, and have strictly positive sums after optional ε smoothing. If either vector has zero total mass, the divergence is undefined.

A.3 More Results

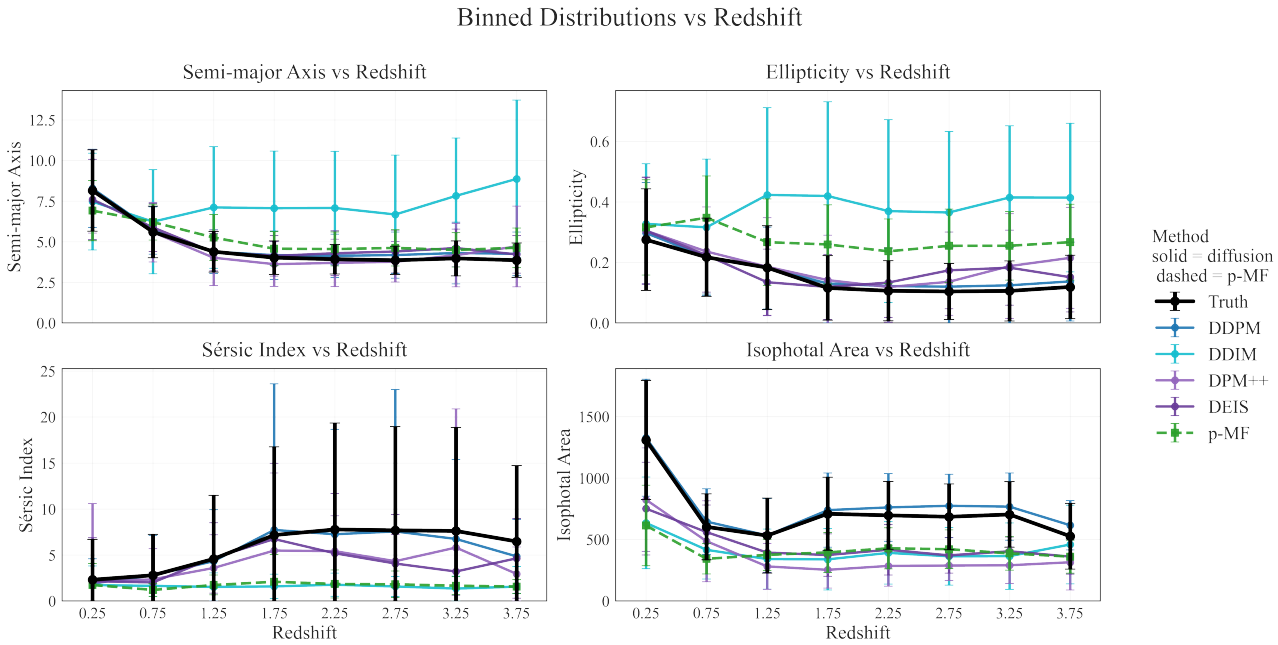

Figure A.1: The redshift-conditional binned morphological distributions from all models, including p-MF model (in green dashed lines.)

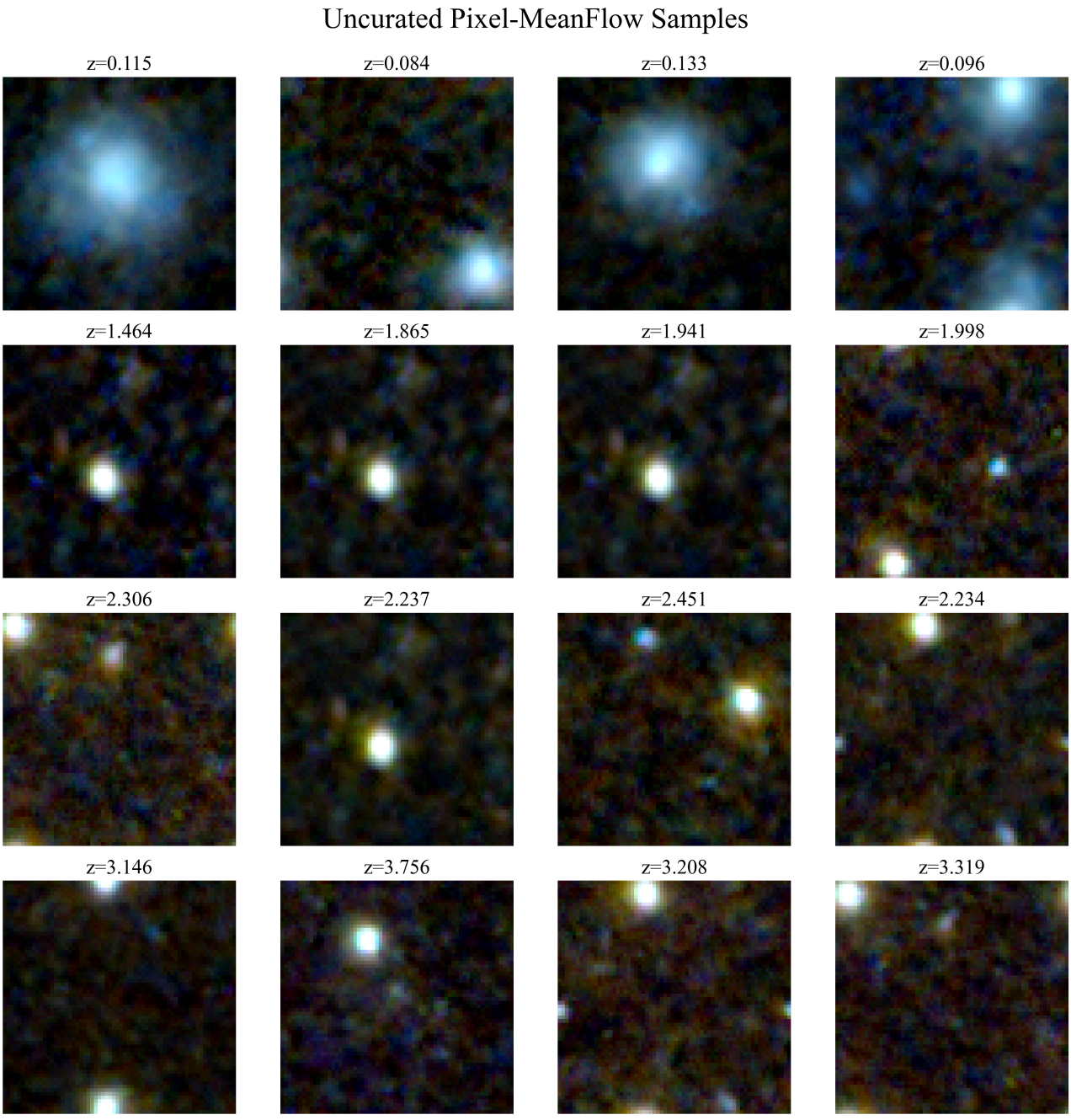

Figure A.2: Uncurated samples from p-MF model in different redshift ranges.

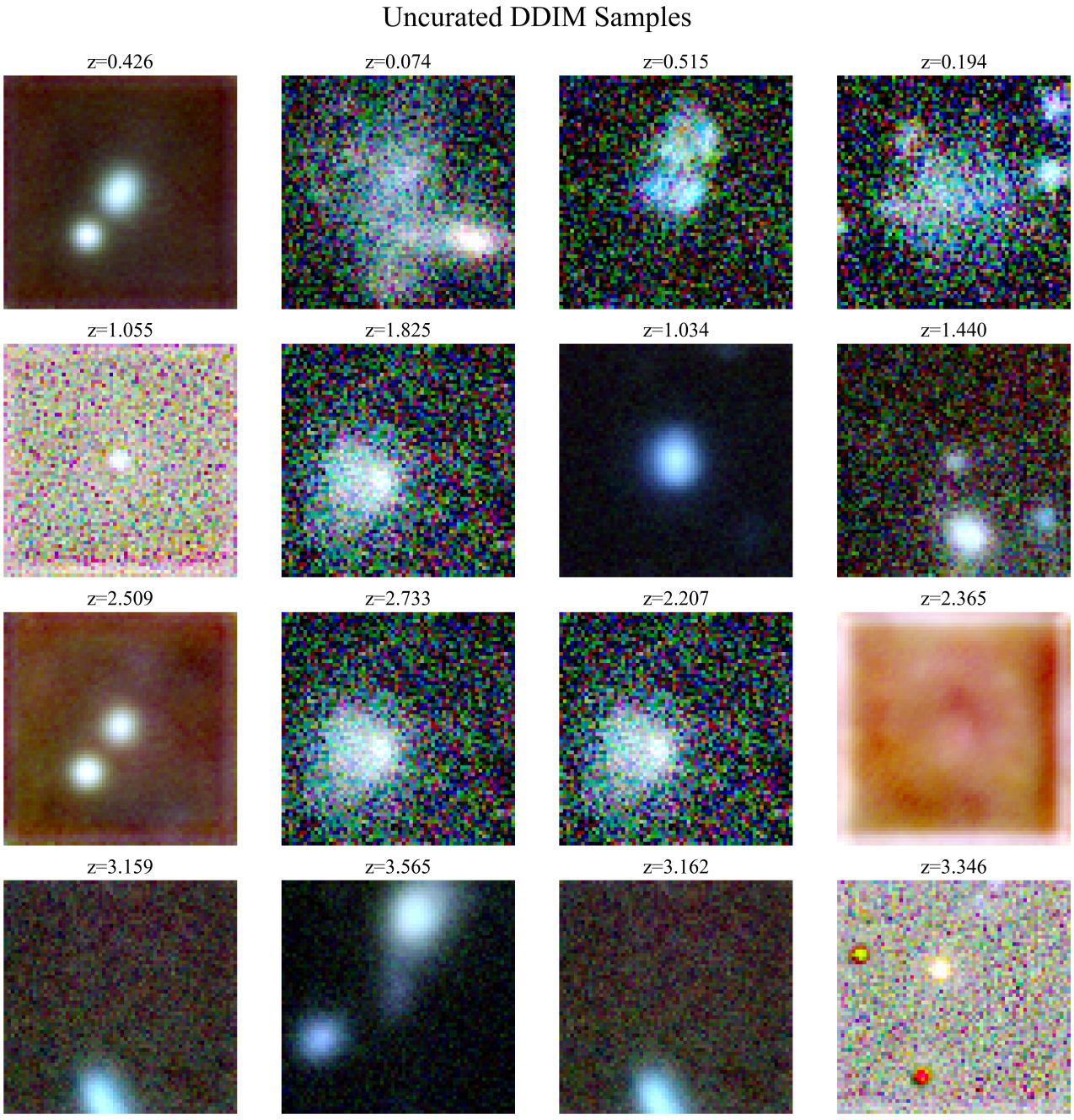

Figure A.3: Uncurated samples from DDIM sampler in different redshift ranges.

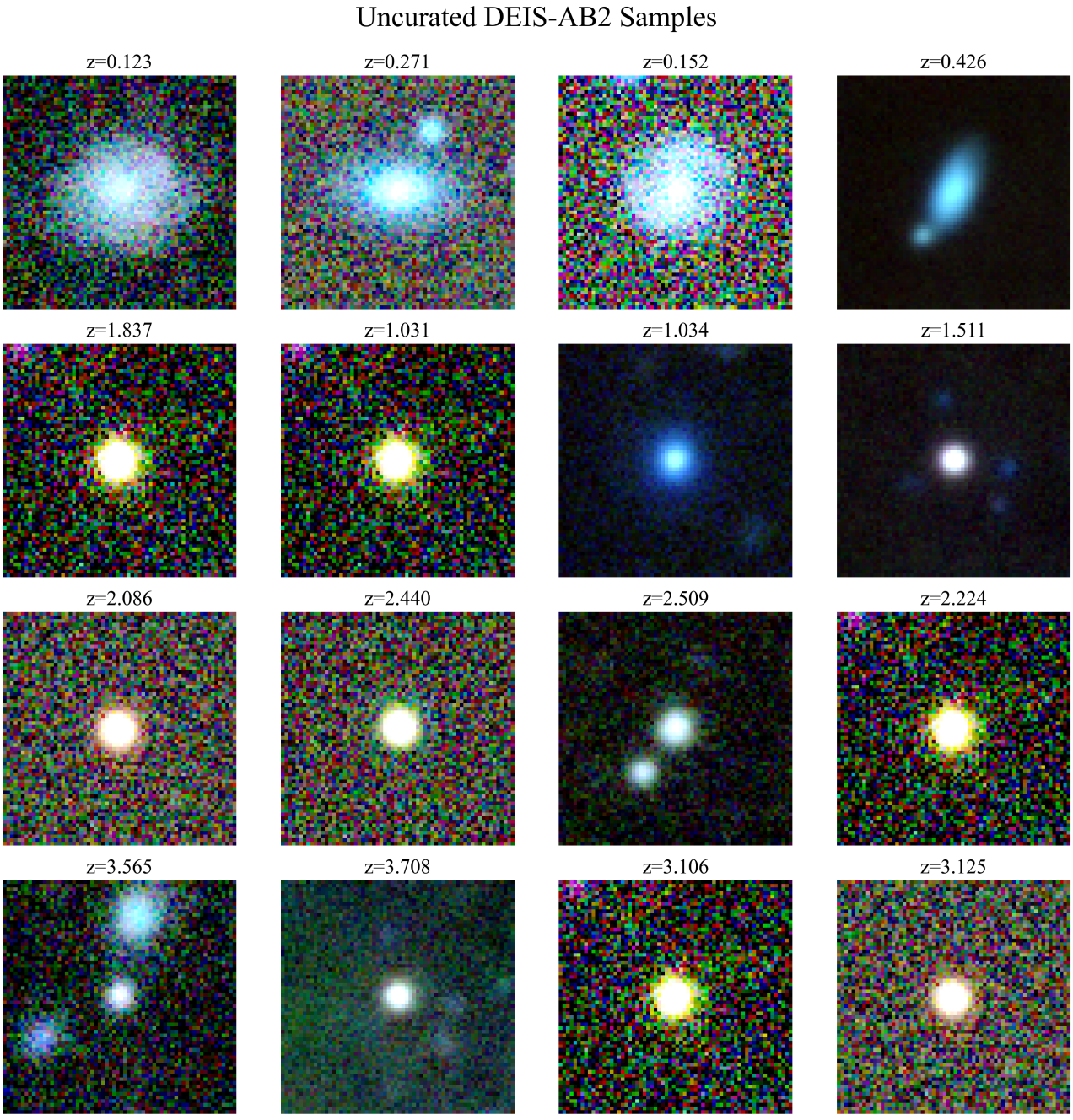

Figure A.4: Uncurated samples from DEIS-AB2 sampler in different redshift ranges.

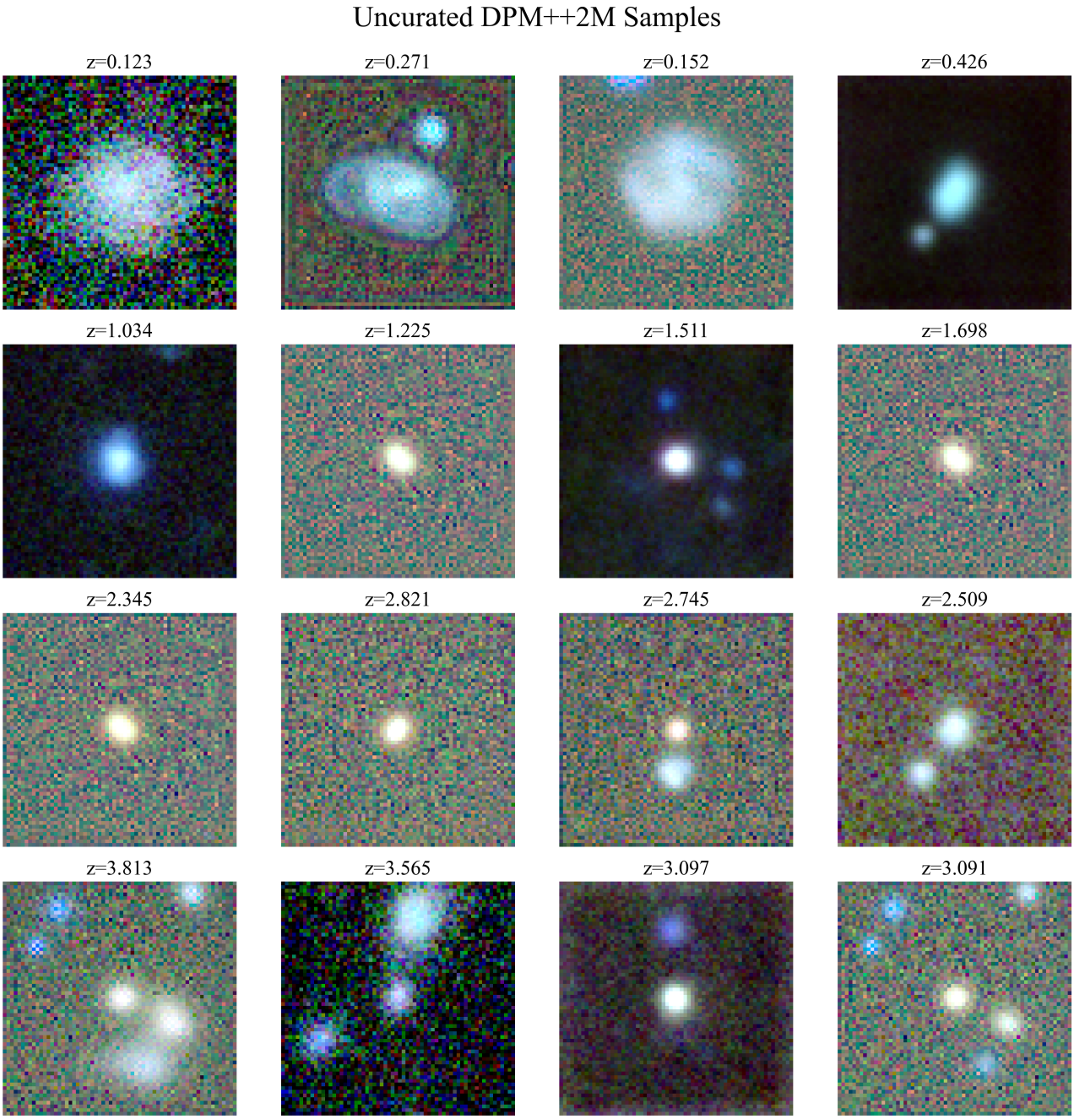

Figure A.5: Uncurated samples from DPM++2M sampler in different redshift ranges.

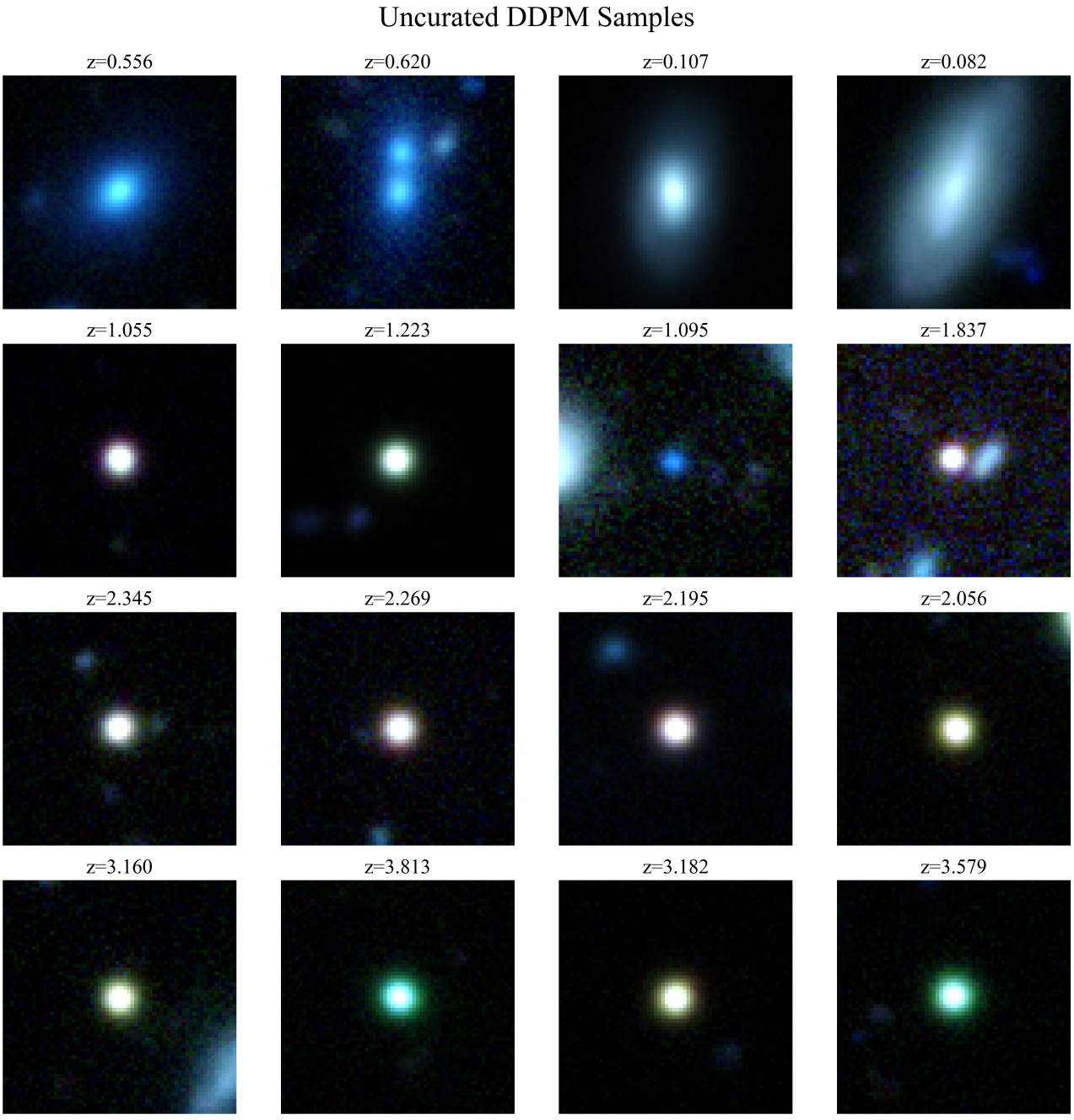

Figure A.6: Uncurated samples from DDPM sampler in different redshift ranges.