RESIDUAL DIFFUSION IMPLICIT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion models have recently achieved state-of-the-art results in image generation and reconstruction, yet their initialization on pure Gaussian noise makes them poorly aligned with inverse problems such as denoising and super-resolution. This mismatch leads to inefficiency, often requiring hundreds of sampling steps, and induces hallucinations that drift away the reconstruction from the ground truth. To overcome these challenges, residual diffusion implicit models (RDIMs) are proposed, constituting a generalized framework that explicitly models the residuals between high-quality (HQ) and low-quality (LQ) images. RDIMs align the forward process with the actual degradation, enabling reconstructions that are faster and more accurate. Inspired by implicit sampling, the reverse process can skip intermediate timesteps, allowing for few-step or even single-step reconstructions while mitigating the hallucinations inherent to long diffusion chains. Furthermore, RDIMs introduce a controllable variance mechanism that interpolates between deterministic and stochastic sampling, balancing fidelity and diversity depending on degradation severity. Experiments on denoising and super-resolution benchmarks demonstrate that RDIMs consistently outperform conventional DDPMs and match or surpass ResShift, while reducing the number of sampling steps by up to $100\times$. The results position RDIMs as an efficient solution for a broad range of image restoration tasks.

1 INTRODUCTION

Image reconstruction is a fundamental problem in computer vision and signal processing, aiming to recover high-quality (HQ) images from corrupted observations. Tasks such as image denoising and super-resolution (SR) are crucial for numerous real-world applications, including medical and biological imaging, satellite imagery, and consumer photo enhancement (Sagheer & George, 2020; Wang et al., 2022; Delbracio et al., 2021).

Denoising diffusion probabilistic models (DDPMs) (Ho et al., 2020) have emerged as a powerful class of models for image reconstruction. Their probabilistic formulation and iterative refinement enable them to handle challenging degradations by progressively improving predictions through small corrective updates (Saharia et al., 2022). Moreover, their stochasticity allows exploration of multiple plausible reconstruction paths, which promotes output diversity and often leads to better solutions (Lugmayr et al., 2022; Whang et al., 2022). These properties make diffusion models well-suited to deal with severe noise and information loss (Chung et al., 2022a).

However, these strengths also introduce practical challenges. Although stochasticity is beneficial for capturing diversity and avoiding poor generalization (Lugmayr et al., 2022; Whang et al., 2022; Dhariwal & Nichol, 2021), excessive and uncontrolled variability can hinder convergence in inverse problems, destabilizing the reconstruction process and leading to inconsistent outputs. Therefore, balancing stochasticity is crucial (Chung et al., 2022b). More critically, the standard DDPM formulation initializes the reverse process from pure noise, which is misaligned with reconstruction tasks where a degraded input already provides valuable information (Chung et al., 2022b; Yue et al., 2023; Wu et al., 2024). Additionally, the recursive formulation of diffusion models leads to an inefficient reverse process requiring to traverse all diffusion timesteps, often hundreds (Shih et al., 2023; Liu et al., 2024), making them computationally expensive and impractical in latency-sensitive settings.

To address these challenges, residual diffusion implicit models (RDIMs) are proposed, constituting a new diffusion framework tailored for inverse problems. Instead of diffusing images into noise, the

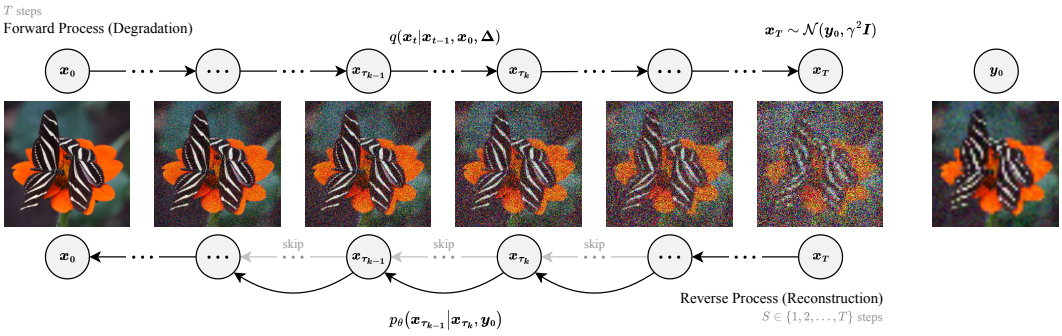


Figure 1: Overview of RDIM, a diffusion framework tailored for inverse problems, such as image reconstruction. The reverse process can accurately reconstruct back the data in $S \leq T$ steps.

RDIM forward process gradually removes the structured residuals between an HQ image and its low-quality (LQ) counterpart (similar to ResShift (Yue et al., 2023)), thereby producing a corrupted version closely aligned with the observed input (see Figure 1). The reverse process then learns to reconstruct the HQ image from the degraded version, effectively undoing the degradation step by step. The framework inherently incorporates variance due to its diffusion nature, which can be controlled via a hyperparameter, thus allowing to interpolate between stochastic and deterministic reconstructions. This leads to improved versatility and greater exploration of solutions. Moreover, the proposed reverse process formulation adopts an implicit sampling in the style of denoising diffusion implicit model (DDIM) (Song et al., 2021), allowing to skip intermediate steps, thus significantly improving the efficiency of the reconstruction process through few-step or even single-step HQ reconstructions. In summary, the main contributions of this paper are:

- A novel diffusion framework for inverse problems that generalizes ResShift and provides an implicit formulation with efficient sampling, enabling reconstructions in a few or even on a single step.
- A controllable variance mechanism that interpolates between deterministic and stochastic reconstructions, balancing fidelity and diversity depending on degradation severity.
- Evidence that reducing the number of reverse steps not only accelerates inference but also yields more faithful reconstructions by mitigating the hallucination effects inherent to long diffusion chains, which otherwise cause outputs to drift away from the degraded input.
- State-of-the-art results on denoising and SR benchmarks, showing that RDIMs outperforms existing methods while reducing the number of inference steps by up to 100×.

The implementation is available at `https://anonymous.4open.science/r/RDIM/`.

2 Methodology

RDIM is a diffusion framework tailored for inverse problems (herein focused on image reconstruction) where the forward process gradually degrades the original data into an informed corrupted version. The reverse process is efficient, allowing for a minimal number of steps (see Figure 1).

2.1 Problem Definition

Inverse problems are concerned with the recovery of a signal, $\boldsymbol{x_0} \in \mathcal{X}$, from a corrupted observation, $\boldsymbol{y_0} \in \mathcal{Y}$. Particularly, the forward model that degrades the original signal can be expressed as:

$$\boldsymbol{y_0} = \mathcal{F}(\boldsymbol{x_0}), \tag{1}$$

where $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$ is a known or unknown forward operator that often entails information loss (e.g., blurring, downsampling, masking, or noise). Accordingly, such problems are often ill-posed.

Meanwhile, deep learning (DL) techniques can be leveraged to learn a parametric reconstruction model $\mathcal{R} : \mathcal{Y} \to \mathcal{X}$, with trainable parameters Θ, that invert the forward model:

$$\boldsymbol{x_0} \approx \mathcal{R}(\boldsymbol{y_0}; \Theta). \tag{2}$$

Traditional diffusion models reconstruct the signal $\boldsymbol{x_0}$ through a parameterized Markov chain with length T, which starts from pure noise and progressively denoises latent variables, $\boldsymbol{x_t}$, at each step $t \in \{1, 2, \dots, T\}$. Hence, they first derive a diffusion process that transforms $\boldsymbol{x_0}$ into pure noise. Subsequently, they learn to reverse this process by training a parametric model, p_θ, which can reconstruct $\boldsymbol{x_0}$ back from pure noise, $\boldsymbol{x_T} \sim \mathcal{N}(0, \boldsymbol{I})$, while conditioning on the corresponding degraded observation, $\boldsymbol{y_0}$. However, this diffusion process is fundamentally misaligned with the degradation model in Equation 1, since it maps $\boldsymbol{x_0}$ to pure noise rather than to the corrupted observation $\boldsymbol{y_0}$. In contrast, the RDIM forward process is explicitly designed to align with the degradation mechanism by progressively removing the residuals between the clean and corrupted signals while optionally injecting a controllable amount of noise. This stochastic component introduces variability that improves generalization, enabling the model to balance fidelity and diversity during reconstruction and better capture the uncertainty inherent in inverse problems.

2.2 Markovian Forward Process

Considering that $\boldsymbol{x_0}$ and $\boldsymbol{y_0}$ denote the original data and its corrupted version[1], respectively, the RDIM forward process (degradation) intends to gradually remove fractions of the residual, $\boldsymbol{\Delta} = \boldsymbol{x_0} - \boldsymbol{y_0}$, from $\boldsymbol{x_0}$ over a series of timesteps $t \in \{1, 2, \dots, T\}$. For that purpose, a forward process fixed to a Markov chain is first defined, which converts the distribution of the original data, $q(\boldsymbol{x_0})$, into the last latent variable distribution, $q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = q(\boldsymbol{x_T}|\boldsymbol{y_0})$ (see Appendix A.1). Subsequently, the Markovian formulation is relaxed to derive a non-Markovian process that preserves the same marginal distributions. Following, the whole Markovian forward process is defined as:

$$q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = \prod_{t=1}^{T} q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{\Delta}), \tag{3}$$

where all latent variables $\boldsymbol{x_1}, \dots, \boldsymbol{x_T}$ have the same dimensionality as the original data, $\boldsymbol{x_0} \sim q(\boldsymbol{x_0})$.

The residual is removed from $\boldsymbol{x_0}$ according to a fixed weighting schedule $\lambda_1, \lambda_2, \dots, \lambda_T$, which is also used to parameterize the variance in each diffusion transition distribution, defined as a Gaussian. Consequently, at each timestep t, the latent variable $\boldsymbol{x_t}$ is expressed in terms of the latent variable at the previous timestep, $\boldsymbol{x_{t-1}}$, and the residual, $\boldsymbol{\Delta}$, as follows:

$$q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{\Delta}) = \mathcal{N}(\boldsymbol{x_t}|\boldsymbol{x_{t-1}} - \lambda_t \boldsymbol{\Delta}, \gamma^2 \lambda_t \boldsymbol{I}), \tag{4}$$

where $\gamma \in [0, \infty)$ is a constant hyperparameter introduced to control the strength of the variance, thus allowing interpolation between a deterministic (when $\gamma = 0$) and a stochastic ($\gamma > 0$) forward process. Moreover, each weight λ_t, used to control the amount of residual to be removed between each diffusion step, is computed in terms of small non-negative constant hyperparameters $\beta_0, \beta_1, \dots, \beta_T$ as $\lambda_t = \beta_t - \beta_{t-1}$ (see Section 2.6 for details on the β-schedule).

Furthermore, to avoid a computationally expensive diffusion process, the cumulative forward transitions are expressed in closed form by relying on the reparameterization trick (see Appendix A.1):

$$q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = \mathcal{N}(\boldsymbol{x_t}|\boldsymbol{x_0} - \beta_t \boldsymbol{\Delta}, \gamma^2 \beta_t \boldsymbol{I}). \tag{5}$$

Although this forward process matches ResShift (Yue et al., 2023), the corresponding recursive formulation yields an inefficient reverse process that must iterate over many timesteps (particularly for HQ inverse problems). Therefore, a DDIM-inspired non-Markovian forward process is derived, which preserves the marginal in Eq. (5) while still allowing a Markovian reverse process.

2.3 Non-Markovian Forward Process

The forward process is implicitly constructed to ensure consistency with the marginal $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ and the reverse process. As a result, each forward transition becomes additionally conditioned on $\boldsymbol{x_0}$ rather than just on the immediate previous timestep, $\boldsymbol{x_{t-1}}$, and the residual, $\boldsymbol{\Delta}$. This introduces explicit dependency on the initial data $\boldsymbol{x_0}$, decoupling the forward process from strict Markovian constraints. Moreover, the forward process is expressed in terms of the forward transition posterior, $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$, further reflecting the non-Markovian behavior and preservation of $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$.

[1]To match dimensionalities, $\boldsymbol{y_0}$ is upsampled for SR tasks and its channels are replicated for colorization.

Therefore, although the RDIM forward process is still a distribution over trajectories that start at $\boldsymbol{x_0}$ and end at $\boldsymbol{x_T}$, it is defined as a joint distribution that is factored in reverse[2]:

$$q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta}) \prod_{t=2}^{T} q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta}). \tag{6}$$

The non-Markovian nature of the forward process enables designing a reverse process that can be deterministic and simulated with a reduced number of transitions due to the conditioning on $\boldsymbol{x_0}$. In addition, since the ResShift training objective only depends on the marginal distribution, $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$, which is preserved, then RDIM optimization (see Section 2.7) will lead to the same training objective as ResShift. Consequently, already trained ResShift models can be leveraged for RDIM sampling without requiring additional retraining.

2.4 Reverse Process

The reverse process (reconstruction) intends to revert the forward process, thus sampling back the data, $\boldsymbol{x_0}$. This is achieved by starting from $\boldsymbol{x_T} \sim \mathcal{N}(\boldsymbol{y_0}, \gamma^2 \boldsymbol{I})$ and iteratively refining the latent variables $\boldsymbol{x_t}$ until $\boldsymbol{x_0}$ is reached. Accordingly, the reverse process involves computing the forward transition posterior $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$ (reverse transition), defined as a Gaussian distribution:

$$q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta}) = \mathcal{N}(\boldsymbol{x_{t-1}}|\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}, \tilde{\sigma}_t^2 \boldsymbol{I}), \tag{7}$$

where $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ is the mean of the Gaussian distribution and $\tilde{\sigma}_t^2 \boldsymbol{I} = \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{t}}$ is the isotropic covariance matrix. Particularly, the reverse transition is designed to preserve the marginal $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ (see Appendix A.2). Considering $\tilde{\sigma}_t^2$ matches the ResShift variance, $\tilde{\lambda}_t = \gamma^2 \frac{\beta_{t-1}}{\beta_t} \lambda_t$, the mean, $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$, is given as:

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}} = \begin{cases} \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta}, & \text{if } \gamma = 0, \\ \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2 \beta_{t-1} - \tilde{\sigma}_t^2} \left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t \boldsymbol{\Delta}}{\sqrt{\gamma^2 \beta_t}} \right), & \text{if } \gamma \neq 0, \end{cases} \tag{8}$$

where, for $\gamma = 0$, the reverse process essentially becomes a linear interpolation between the corrupted and original data, which underscores that the RDIM forward process is aligned with a forward model (degradation process) that converts $\boldsymbol{x_0}$ into $\boldsymbol{y_0}$.

Furthermore, fixing $\tilde{\sigma}_t^2$ to the ResShift variance, $\tilde{\lambda}_t$, also results in $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ matching the mean of the ResShift reverse transition (see Appendix A.3). Hence, RDIM becomes ResShift for this specific variance, revealing that ResShift is a particular case of RDIM. Subsequently, a constant hyperparameter, $\eta \in [0, 1]$, can be introduced to interpolate between a deterministic (η=0) and a stochastic (η>0) reverse process when $\gamma \neq 0$, allowing control over the variability in the RDIM reverse trajectory:

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} = \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2 \beta_{t-1} - \eta^2 \tilde{\lambda}_t} \left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t \boldsymbol{\Delta}}{\sqrt{\gamma^2 \beta_t}} \right), \quad \tilde{\sigma}_{t|\gamma\neq 0}^2 = \eta^2 \tilde{\lambda}_t. \tag{9}$$

where, $\eta = 1$ makes the RDIM reverse process identical to ResShift. Meanwhile, setting $\gamma = 0$ converts RDIM into a strictly deterministic model (γ=0 $\Rightarrow$ $\tilde{\lambda}_t$=0), avoiding sampling random noise.

However, during inference, $\boldsymbol{x_0}$ and $\boldsymbol{\Delta}$ are unknown, thus sampling from the true reverse transition distribution is not possible. Therefore, a learnable parametric model, $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})$, defined as a Gaussian distribution, is introduced to approximate the true reverse transition $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$:

$$p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0}) = \mathcal{N}\left(\boldsymbol{x_{t-1}}|\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right), \sigma_\theta^2\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)\boldsymbol{I}\right), \tag{10}$$

where $\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)$ is the mean of the Gaussian distribution and $\sigma_\theta^2\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)\boldsymbol{I} = \boldsymbol{\Sigma_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)$ is the isotropic covariance matrix. In particular, the variance of the true reverse transition, $\tilde{\sigma}_t^2$, does not have any learnable parameters because it is defined in terms of constant hyperparameters, which are known. Therefore, the variance of $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})$ can be fixed to equal exactly the variance of $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$:

$$\sigma_\theta^2\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) = \tilde{\sigma}_t^2. \tag{11}$$

[2]The forward transition, $q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, \boldsymbol{x_0}, \boldsymbol{\Delta})$, can be derived via Bayes' rule.

Meanwhile, $\boldsymbol{\mu_\theta}(\boldsymbol{x_t}, \boldsymbol{y_0}, t)$ approximates the mean of the true reverse transition, $\tilde{\boldsymbol{\mu}}_\boldsymbol{t}$. Considering that $\boldsymbol{x_0}$ and $\boldsymbol{\Delta}$ are the only unknown terms and $\boldsymbol{\Delta}$ can be estimated from $\boldsymbol{x_0}$ and $\boldsymbol{y_0}$, then the model solely needs to predict $\boldsymbol{x_0}$ (see Appendix A.4). Accordingly, the mean $\boldsymbol{\mu_\theta}(\boldsymbol{x_t}, \boldsymbol{y_0}, t)$ is defined as:

$$\boldsymbol{\mu_\theta}(\boldsymbol{x_t}, \boldsymbol{y_0}, t) = \begin{cases} \hat{\boldsymbol{x}}_\mathbf{0} - \beta_{t-1}\hat{\boldsymbol{\Delta}}, & \text{if } \gamma = 0, \\ \hat{\boldsymbol{x}}_\mathbf{0} - \beta_{t-1}\hat{\boldsymbol{\Delta}} + \sqrt{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}\left(\frac{\boldsymbol{x_t} - \hat{\boldsymbol{x}}_\mathbf{0} + \beta_t\hat{\boldsymbol{\Delta}}}{\sqrt{\gamma^2\beta_t}}\right), & \text{if } \gamma \neq 0, \end{cases} \tag{12}$$

where $\hat{\boldsymbol{x}}_\mathbf{0} = f_\theta(\boldsymbol{x_t}, \boldsymbol{y_0}, t)$ denotes the $\boldsymbol{x_0}$ prediction from a neural network given $\boldsymbol{x_t}$, $\boldsymbol{y_0}$, and timestep t. The neural network is parameterized by weights θ and $\hat{\boldsymbol{\Delta}} = \hat{\boldsymbol{x}}_\mathbf{0} - \boldsymbol{y_0}$ represents the $\boldsymbol{\Delta}$ estimation. Hence, the whole approximate reverse process is expressed by the following joint distribution:

$$p_\theta(\boldsymbol{x_{0:T}}|\boldsymbol{y_0}) = p(\boldsymbol{x_T}|\boldsymbol{y_0}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0}). \tag{13}$$

2.5 Long-Range Reverse Transition

Particularly, the derived reverse transition structurally matches the reparameterized form of the marginal $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ (see Appendix A.2), which models the cumulative transitions from $\boldsymbol{x_0}$ to $\boldsymbol{x_{t-1}}$ in the forward process. Therefore, the reverse transition formulation aligns with the concept of cumulative transitions, allowing the reverse process to efficiently sample any state at an arbitrary timestep $\tau_{k-1} \in \{0, 1, \ldots, T-1\}$ by skipping intermediate latent variables in the reverse trajectory. Accordingly, the reverse process can be simulated with fewer timesteps, thereby accelerating sampling. Using the reparameterization trick, $\boldsymbol{x_{\tau_{k-1}}} \sim p_\theta\left(\boldsymbol{x_{\tau_{k-1}}}|\boldsymbol{x_{\tau_k}}, \boldsymbol{y_0}\right)$ can be sampled as follows:

$$\boldsymbol{x_{\tau_{k-1}}} = \begin{cases} \hat{\boldsymbol{x}}_\mathbf{0} - \beta_{\tau_{k-1}}\hat{\boldsymbol{\Delta}}, & \text{if } \gamma = 0, \\ \hat{\boldsymbol{x}}_\mathbf{0} - \beta_{\tau_{k-1}}\hat{\boldsymbol{\Delta}} + \sqrt{\gamma^2\beta_{\tau_{k-1}} - \eta^2\tilde{\lambda}_{\tau_k}}\hat{\boldsymbol{\epsilon}} + \sqrt{\eta^2\tilde{\lambda}_{\tau_k}}\boldsymbol{z}, & \text{if } \gamma \neq 0, \end{cases} \tag{14}$$

where $(\tau_{k-1}, \tau_k) \in \left\{(t', t) \in \mathbb{N}_0^2 \mid t' + 1 \leq t \leq T\right\}$, $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$, and $\hat{\boldsymbol{\epsilon}}$ is expressed by the following relationship when $\gamma \neq 0$ (see Equation (32) in Appendix A.2):

$$\hat{\boldsymbol{\epsilon}} = \frac{\boldsymbol{x_{\tau_k}} - \hat{\boldsymbol{x}}_\mathbf{0} + \beta_{\tau_k}\hat{\boldsymbol{\Delta}}}{\sqrt{\gamma^2\beta_{\tau_k}}}. \tag{15}$$

Essentially, each iteration of the reverse process involves predicting the original data sample, $\boldsymbol{x_0}$. This estimate is then used to compute the residual $\boldsymbol{\Delta}$ and the noise component $\boldsymbol{\epsilon}$, which together guide the update to the next less-degraded state, $\boldsymbol{x_{\tau_{k-1}}}$. As the reverse process progresses, the model gradually refines its prediction of $\boldsymbol{x_0}$ at each step, leveraging the increasingly accurate intermediate states. This iterative refinement culminates in an accurate prediction of $\boldsymbol{x_0}$. Moreover, the ability of the reverse process to skip intermediate steps not only enables few-step generation but also allows one-step predictions, thus demonstrating the efficiency and flexibility of the RDIM sampling procedure. Here, the number of sampling timesteps along the reverse trajectory, $S \in \{1, 2, \ldots, T\}$, is set arbitrarily. For each case, a uniform schedule is used, as detailed in Appendix C.4.

2.6 Residual $\boldsymbol{\beta}$-Schedule

The residual β-schedule employed is defined by a circular curve (similar to the fourth quadrant p-norm shape), ensuring a smooth and adjustable transition between $\boldsymbol{x_0}$ and $\boldsymbol{x_T}$:

$$\beta_t = \frac{t}{T + (p-1)(T-t)}, \tag{16}$$

where $p \in (0, \infty)$ is a parameter that allows controlling the steepness of the curve. As it increases the β-schedule exhibits a slower initial progression, followed by a rapid increase to larger and more pronounced updates. This design allows for a gentle removal of the residual and injection of noise in the early timesteps of the forward process, which become progressively more aggressive throughout the diffusion trajectory. Figure 2 illustrates the impact of the parameter p on the diffusion process.

Furthermore, this choice for the β-schedule ensures that $\beta_0 = 0$ and $\beta_T = 1$, such that the residual, $\boldsymbol{\Delta}$, is fully removed from $\boldsymbol{x_0}$ after exactly T timesteps. As a result, the last latent variable, $\boldsymbol{x_T}$, converges

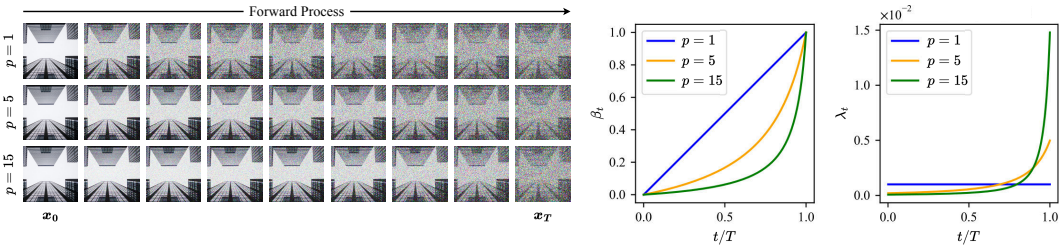


Figure 2: Progression of weights β_t and λ_t across timesteps and impact of p on the diffusion process.

to a noisy sample centered at the corrupted data, $\boldsymbol{y_0}$. Additionally, since $\beta_0 = 0$, it follows that when $\gamma \neq 0$ the variance of any reverse transition distribution from $\boldsymbol{x_{\tau_k}}$ to $\boldsymbol{x_0}$ is $\eta^2 \tilde{\lambda}_{\tau_k} = \eta^2 \gamma^2 \frac{\beta_0}{\beta_{\tau_k}} \lambda_{\tau_k} = 0$. Therefore, $p_\theta(\boldsymbol{x_0}|\boldsymbol{x_{\tau_k}}, \boldsymbol{y_0})_{\gamma \neq 0}$ degenerates into a δ-distribution centered at $\hat{\boldsymbol{x}}_\mathbf{0}$. Logically, under these conditions, η does not have any impact on the last transition of the reverse process.

2.7 Optimization

At each step of the sampling process, the neural network parameterized by weights θ yields an estimate of $\boldsymbol{x_0}$. During training, these parameters are learned to assure that the model marginal $p_\theta(\boldsymbol{x_0}|\boldsymbol{y_0})$ fits the true posterior distribution $q(\boldsymbol{x_0}|\boldsymbol{y_0})$ via:

$$q(\boldsymbol{x_0}|\boldsymbol{y_0}) \approx p_\theta(\boldsymbol{x_0}|\boldsymbol{y_0}) = \int p(\boldsymbol{x_T}|\boldsymbol{y_0}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})\, d\boldsymbol{x_{1:T}}, \tag{17}$$

which ensures, during inference, that the data, $\boldsymbol{x_0}$, can be sampled back accurately given $\boldsymbol{y_0}$. Accordingly, $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})$ is required to closely approximate the true forward transition posterior, $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$. This is achieved by minimizing the Kullback–Leibler (KL) divergence between both distributions, while accounting for all timesteps. In fact, this objective can be reduced for simplicity to (see Appendix A.4):

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{\boldsymbol{x_0}, \boldsymbol{\Delta}, t} \left[\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_\mathbf{0}\|^2 \right]. \tag{18}$$

Notably, ResShift shares the same training objective as RDIM, further highlighting that ResShift is a particular case of RDIM and that its trained models can be used for RDIM sampling without retraining. The RDIM training and sampling procedures are described in Algorithms 1 and 2, respectively.

Algorithm 1 Training

repeat
  $\boldsymbol{x_0}, \boldsymbol{y_0} \sim q(\boldsymbol{x_0}, \boldsymbol{y_0}) = q(\boldsymbol{x_0}) q(\boldsymbol{y_0}|\boldsymbol{x_0})$
  $\boldsymbol{\Delta} = \boldsymbol{x_0} - \boldsymbol{y_0}$
  $t \sim \mathcal{U}(1, T)$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
  $\boldsymbol{x_t} \sim q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$
  $\hat{\boldsymbol{x}}_\mathbf{0} = f_\theta(\boldsymbol{x_t}, \boldsymbol{y_0}, t)$
  $\mathcal{L} = \|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_\mathbf{0}\|^2$
  Take gradient descent step on $\nabla_\theta \mathcal{L}$
until convergence
return f_θ

Algorithm 2 Sampling

$\Upsilon = \{\tau_S = T, \tau_{S-1}, \ldots, \tau_1, \tau_0 = 0\}$
$\boldsymbol{x_T} \sim \mathcal{N}(\boldsymbol{y_0}, \gamma^2 \boldsymbol{I})$
for $k = S, S-1, \ldots, 1$ **do**
  $\hat{\boldsymbol{x}}_\mathbf{0} = f_\theta(\boldsymbol{x_{\tau_k}}, \boldsymbol{y_0}, \tau_k)$
  $\hat{\boldsymbol{\Delta}} = \hat{\boldsymbol{x}}_\mathbf{0} - \boldsymbol{y_0}$
  if $\gamma \neq 0$ **then** $\hat{\boldsymbol{\epsilon}} = \frac{\boldsymbol{x_{\tau_k}} - \hat{\boldsymbol{x}}_\mathbf{0} + \beta_{\tau_k} \hat{\boldsymbol{\Delta}}}{\sqrt{\gamma^2 \beta_{\tau_k}}}$
  $\boldsymbol{x_{\tau_{k-1}}} \sim p_\theta\left(\boldsymbol{x_{\tau_{k-1}}}|\boldsymbol{x_{\tau_k}}, \boldsymbol{y_0}\right)$
end for
return $\boldsymbol{x_0}$

3 Experiments

RDIM is evaluated on image denoising and single image SR using the FMD (Zhang et al., 2019), SIDD (Abdelhamed et al., 2018; 2019), and DIV2K (Agustsson & Timofte, 2017; Timofte et al., 2017) datasets. Two RDIM variants with $\gamma = 3.0$, $\eta = 1.0$, and $p = 5.0$ are considered, differing only in the number of sampling timesteps, S. RDIM-1 corresponds to single-step deterministic inference

($S = 1$), while RDIM-10 denotes sampling with $S = 10$ steps. The deterministic nature of RDIM-1 results from the final reverse transition degenerating into a δ-distribution when $\gamma \neq 0$ and $\beta_0 = 0$ (see Sections 2.5 and 2.6). Moreover, RDIM is compared against DDPM and ResShift with $S = T = 100$. Although ResShift is often employed with $S = T = 10$, there is a significant performance improvement when using longer diffusion chains. This effect is evident in the experiments shown in Appendix C.7, where ResShift improves peak signal-to-noise ratio (PSNR) from 39.363 dB for $T = 10$ to 43.599 dB for $T = 100$. Additional qualitative results on image inpainting, colorization, and deblurring are provided on FFHQ (Karras et al., 2019). Experimental details are in Appendix C, including RDIM assessment when varying S (Figure 11).

Image Denoising. RDIM is compared against BM3D (Dabov et al., 2007), DnCNN (Zhang et al., 2017), DDPM, and ResShift. Diffusion models were trained with the same network architecture (detailed in Appendix C.2) and number of diffusion timesteps ($T = 100$). The only distinction lies in the diffusion framework employed. Results are listed in Table 1a. On FMD-Confocal-BPAE-Raw, RDIM-10 achieves the best results in terms of PSNR and structural similarity index measure (SSIM), followed by RDIM-1. On FMD-Confocal-Zebrafish-Raw, ResShift attains the best PSNR score, but is $10\times$ slower than RDIM-10, which obtains comparable PSNR performance and the best SSIM score. On SIDD-Medium, RDIM-1 yields superior results. Diffusion models, which inherently capture richer structures than DnCNN, have their gains diminished on SIDD-Medium due to a small patch size employed (kept the same across all experiments for consistency). Figure 3 presents a qualitative comparison, further illustrating the enhanced denoising achieved by RDIM.

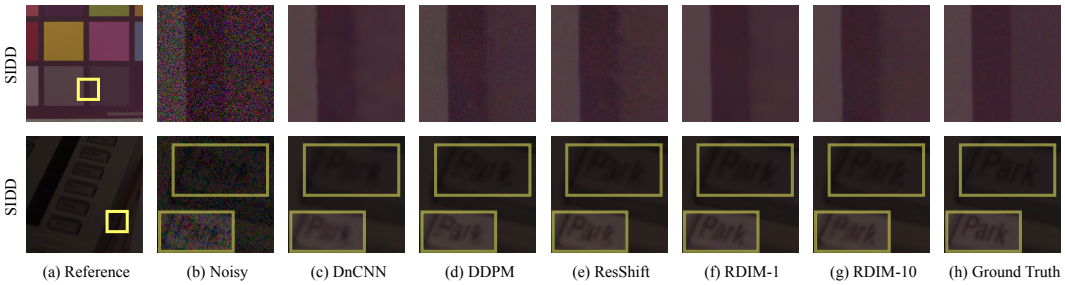


Figure 3: Qualitative denoising analysis on cropped regions from the SIDD dataset. Since SIDD contains noisy images captured under challenging lighting conditions, brightness-adjusted crops of the marked regions are shown in the bottom row for enhanced visualization.

Super-Resolution. A comparative analysis with $\times 2$ and $\times 4$ downsampling factors evaluates RDIM against ESRGAN (Wang et al., 2018), DDPM, and ResShift. Diffusion models were trained under the same constraints and number of diffusion timesteps ($T = 100$), differing only in the diffusion framework employed. Results are shown in Table 1b. On both DIV2K-Unknown-$\times 2$ and DIV2K-Unknown-$\times 4$, RDIM-1 performs the best, followed by RDIM-10, highlighting that RDIM consistently surpasses ResShift and DDPM. Figure 4 showcases qualitative results, demonstrating the high-fidelity reconstruction by RDIM, while other methods often hallucinate details.

Table 1: Comparative analysis of RDIM against relevant state-of-the-art techniques for (a) denoising and (b) SR. Green color highlights the best score overall and Blue color the second best.

(a) Denoising on images from the FMD (BPAE and zebrafish confocal fluorescence microscopy images) and SIDD datasets.

Denoising Method	S↓	FMD-BPAE		FMD-Zebrafish		SIDD-Medium	
		PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Noisy	–	31.596	0.812	26.732	0.603	27.797	0.515
BM3D	–	35.862	0.933	35.289	0.918	35.880	0.906
DnCNN	–	37.609	0.950	37.169	0.941	39.838	0.957
DDPM	100	41.775	0.981	43.214	0.974	39.329	0.945
ResShift	100	43.599	0.984	**45.167**	0.976	39.663	0.949
RDIM-1	**1**	43.987	0.985	44.229	0.976	**40.335**	**0.962**
RDIM-10	10	**44.147**	**0.986**	45.027	**0.978**	39.979	0.958

(b) $\times 2$ and $\times 4$ SR on images from the DIV2K dataset.

SR Method	S↓	DIV2K-$\times 2$		DIV2K-$\times 4$	
		PSNR↑	SSIM↑	PSNR↑	SSIM↑
LR (Bicubic)	–	25.112	0.704	21.742	0.574
ESRGAN	–	30.017	0.857	24.957	0.690
DDPM	100	31.949	0.893	26.446	0.739
ResShift	100	32.368	0.903	26.627	0.750
RDIM-1	**1**	**33.887**	**0.924**	**28.280**	**0.798**
RDIM-10	10	33.019	0.914	27.266	0.770

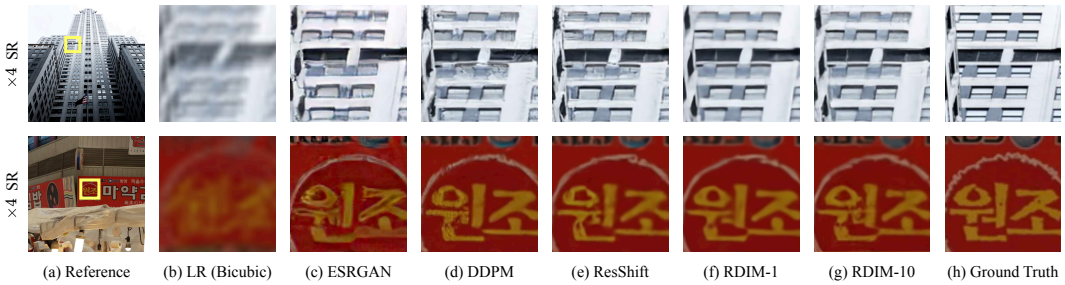


Figure 4: SR qualitative comparison on cropped regions from the DIV2K dataset.

Additional Image Restoration Tasks. Further evaluation on image inpainting, colorization, and deblurring tasks demonstrates the generalization capabilities of RDIM. Figure 5 presents qualitative results obtained with RDIM-10. Additional qualitative results are provided in Appendix C.

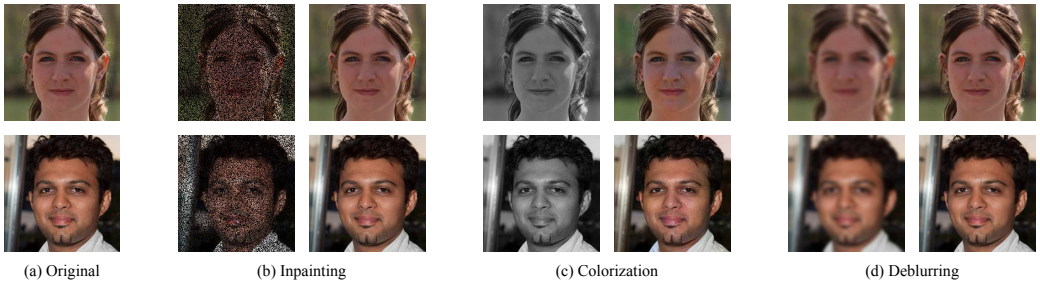


Figure 5: RDIM-10 results in image inpainting, colorization, and deblurring on the FFHQ dataset. In (b), (c) and (d), the left side represents the input image and the right side the output.

Discussion of results. RDIM and ResShift consistently outperform DDPM, emphasizing that their diffusion process is more closely aligned with these inverse problems. Moreover, RDIM demonstrates performance comparable to ResShift, often surpassing it, while requiring significantly fewer sampling timesteps. Since DDPM and ResShift require a reverse process with the same number of timesteps as their forward diffusion process, reducing their diffusion steps to match the RDIM sampling time would result in a degradation in performance (Shih et al., 2023). This effect is evident in the experiments conducted in Appendix C.7, where ResShift with a reduced number of diffusion timesteps underperforms compared to its higher-timestep configurations.

Furthermore, FMD-Confocal-Zebrafish-Raw contains noisier images than FMD-Confocal-BPAE-Raw. As shown in Table 1a, RDIM-1 outperforms ResShift on FMD-Confocal-BPAE-Raw, whereas ResShift performs better on FMD-Confocal-Zebrafish-Raw. This suggests that in the presence of stronger degradations a more stochastic approach is advantageous, as variability promotes output diversity. Conversely, when degradations are mild, a more deterministic method ensures consistent and accurate restoration. Therefore, balancing stochasticity is crucial to adapt the method effectively to varying noise levels and degradation strengths. Notably, RDIM-10 achieves comparable results to ResShift in FMD-Confocal-Zebrafish-Raw while requiring only 10 sampling steps instead of 100, rendering inference $10\times$ faster. Further demonstrating its efficiency, RDIM accelerates sampling up to $100\times$ compared to ResShift and DDPM on FMD-Confocal-BPAE-Raw. Additionally, experiments on SIDD highlight that RDIM effectively supports high-resolution (HR) image reconstruction even when operating on relatively small patches (e.g., 64×64) compared to the full image size, which here reach resolutions of up to $\approx 5300 \times 3000$ pixels. Naturally, increasing the patch size will improve performance and could enable restoration of images at even higher resolutions.

In SR, standard diffusion models and ResShift, often exhibit a tendency to hallucinate details that deviate from the ground truth, particularly when employing long diffusion chains. As illustrated in Figure 4, while iterative refinement encourages the generation of natural-looking textures, it frequently trades off fidelity for perceptual quality, leading to reconstructions that drift away from the original structure (see Appendix C.7 for further evidence). Furthermore, the deterministic RDIM-1

outperforms all methods, suggesting a more deterministic approach to SR is beneficial, as too much stochasticity can introduce unwanted variability in the output and the iterative refinement of long-chain diffusion can become detrimental.

4 Related Work

Recently, several strategies have been proposed to accelerate the sampling process of the DDPMs, broadly divided into *training-based* distillation methods (Salimans & Ho, 2022; Luhman & Luhman, 2021; Song et al., 2023) and *training-free* solver-based approaches. Within the first category, trajectory distillation techniques (Salimans & Ho, 2022; Meng et al., 2023; Li et al., 2023) aim to fine-tune a student model, effectively reducing the number of sampling steps in a multi-stage manner. Similarly, consistency distillation methods, inspired by the consistency model (Song et al., 2023), encourage denoising outputs along the sampling trajectory to remain coherent. Notable examples include latent consistency models for text-to-image generation (Luo et al., 2023) and consistency trajectory models for image generation (Kim et al., 2024). While these approaches demonstrate the ability to produce HQ samples with relatively few steps, they often involve substantial training costs, and some degree of information compression from the original model may occur. On the other hand, training-free samplers provide a complementary direction, typically offering greater flexibility since no additional training is required. Most of these methods build upon the ODE formulation of DDPMs, enabling HQ generation in as few as 20 steps (Song et al., 2020; Popov et al., 2022; Bao et al., 2022; Song et al., 2021; Lu et al., 2025; Zheng et al., 2023). For example, DPM-Solver-v3 (Zheng et al., 2023) introduces additional coefficient refinements, reformulating the original ODE solution in a way that consistently improves sample quality, even within 5–10 steps. Nonetheless, DPM-Solver++ has been primarily tailored for image synthesis and is not explicitly optimized for inverse problems such as denoising or SR, where the input already contains partial but degraded information. Its effectiveness in residual modeling or across different degradation modes is therefore less clearly established. In contrast, our proposed method is designed to extend across a broader spectrum of tasks, including denoising, SR, inpainting, and colorization, while retaining efficiency.

Building on the progress of accelerated sampling strategies, another active research direction has focused on adapting diffusion models specifically for image reconstruction tasks. For instance, SR3 (Saharia et al., 2022) extended DDPMs for SR by conditioning on low-resolution (LR) inputs, while SRDiff (Li et al., 2022) incorporated residual prediction to accelerate convergence. Similarly, Whang et al. (2022) leveraged residual-based refinement on top of deterministic deblurring networks, generating diverse reconstructions. Although effective, these methods rely on iterative denoising trajectories from Gaussian noise to images, which are not optimally suited for reconstruction problems (Chung et al., 2022b). To overcome this limitation, alternative formulations have emerged. Denoising diffusion restoration models (DDRMs) (Kawar et al., 2022) introduced unsupervised posterior sampling with pre-trained diffusion models to address linear inverse problems, and ResShift (Yue et al., 2023) explored residual modeling between HR and LR images. Yet, their inference procedures remain computationally demanding, as they iterate sequentially through all timesteps. In this context, the proposed RDIM offers a new perspective. It provides flexible control over the reverse trajectory through step skipping and interpolation between stochastic and deterministic reconstructions, achieving both efficiency and versatility. This makes RDIM well suited for a wide range of image reconstruction tasks, including denoising, SR, inpainting, and colorization.

5 Conclusion

RDIMs constitute a diffusion framework tailored for inverse problems that explicitly models the residuals between HQ and LQ images. Aligning the forward process with the actual degradation and leveraging implicit sampling enables RDIMs to produce accurate reconstructions with significantly fewer steps than conventional DDPMs. Furthermore, RDIM achieves superior results compared to DDPM, reducing hallucinations while maintaining fidelity, highlighting that starting the reverse process closer to the LQ images offers a more informed and effective initialization. Experiments on denoising and SR demonstrate consistent improvements over DDPMs and performance comparable to or exceeding ResShift, achieving HQ results with single or few step inference. These results establish RDIMs as an efficient and versatile approach for a wide range of image reconstruction tasks.

REFERENCES

Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1692–1700, 2018.

Abdelrahman Abdelhamed, Radu Timofte, and Michael S Brown. NTIRE 2019 challenge on real image denoising: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 126–135, 2017.

Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *International Conference on Learning Representations (ICLR)*, 2022.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022a.

Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Come-Closer-Diffuse-Faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 12413–12422, 2022b.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16 (8):2080–2095, 2007.

Mauricio Delbracio, Damien Kelly, Michael S Brown, and Peyman Milanfar. Mobile computational photography: A tour. *Annual review of vision science*, 7(1):571–604, 2021.

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

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

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

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:23593–23606, 2022.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations (ICLR)*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. SRDiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479: 47–59, 2022.

Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. SnapFusion: Text-to-image diffusion model on mobile devices within two seconds. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:20662–20678, 2023.

Jiawei Liu, Qiang Wang, Huijie Fan, Yinong Wang, Yandong Tang, and Liangqiong Qu. Residual denoising diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2773–2783, 2024.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, pp. 1–22, 2025.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11461–11471, 2022.

Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 14297–14306, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, Mikhail Sergeevich Kudinov, and Jiansheng Wei. Diffusion-based voice conversion with fast maximum likelihood sampling scheme. In *International Conference on Learning Representations (ICLR)*, 2022.

Sameera V Mohd Sagheer and Sudhish N George. A review on medical image denoising algorithms. *Biomedical signal processing and control*, 61:102036, 2020.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:4263–4276, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

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

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 114–125, 2017.

Peijuan Wang, Bulent Bayram, and Elif Sertel. A comprehensive review on deep learning based remote sensing image super-resolution methods. *Earth-Science Reviews*, 232:104110, 2022.

Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018.

Jay Whang, Mauricio Delbracio, Hossein Talebi, Chitwan Saharia, Alexandros G Dimakis, and Peyman Milanfar. Deblurring via stochastic refinement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 16293–16303, 2022.

Rongyuan Wu, Lingchen Sun, Zhiyuan Ma, and Lei Zhang. One-step effective diffusion network for real-world image super-resolution. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:92529–92553, 2024.

Zongsheng Yue, Jianyi Wang, and Chen Change Loy. ResShift: Efficient diffusion model for image super-resolution by residual shifting. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:13294–13307, 2023.

Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26 (7):3142–3155, 2017.

Yide Zhang, Yinhao Zhu, Evan Nichols, Qingfei Wang, Siyuan Zhang, Cody Smith, and Scott Howard. A poisson-gaussian denoising dataset with real fluorescence microscopy images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11710–11718, 2019.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. DPM-Solver-v3: Improved diffusion ODE solver with empirical model statistics. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:55502–55542, 2023.

A Derivations

This section presents detailed mathematical derivations to support this work. All intermediate steps and calculations omitted for brevity in the main text are included here for completeness and reference.

A.1 Forward process cumulative transition distribution $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$

The RDIM forward process is designed to align with a forward model that converts the data, $\boldsymbol{x_0}$, into the corresponding corrupted version, $\boldsymbol{y_0}$. To achieve this, the Gaussian transition distribution in Equation (4) is derived for a Markovian version of the RDIM forward process. However, when generating a latent variable $\boldsymbol{x_t}$ starting from $\boldsymbol{x_0}$, the sequential formulation of the diffusion process can become computationally expensive, particularly as the timestep t increases. To address this problem, the reparameterization trick can be leveraged, allowing the cumulative Gaussian transitions of the forward process to be expressed in closed form. As a result, $\boldsymbol{x_t}$ can be computed at an arbitrary timestep t as a function of $\boldsymbol{x_0}$, the fraction of residual between $\boldsymbol{x_0}$ and $\boldsymbol{y_0}$, $\lambda_t\boldsymbol{\Delta}$ (with λ_t determining the amount of residual to be removed between each diffusion step), and optional forward variance parameter γ:

$$\begin{aligned}
\boldsymbol{x_t} &= \boldsymbol{x_{t-1}} - \lambda_t\boldsymbol{\Delta} + \sqrt{\gamma^2\lambda_t}\boldsymbol{\epsilon_t} \\
&= \boldsymbol{x_{t-2}} - \lambda_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\lambda_{t-1}}\boldsymbol{\epsilon_{t-1}} - \lambda_t\boldsymbol{\Delta} + \sqrt{\gamma^2\lambda_t}\boldsymbol{\epsilon_t} \\
&= \cdots \\
&= \boldsymbol{x_0} - \boldsymbol{\Delta}\underbrace{(\lambda_1 + \lambda_2 + \cdots + \lambda_t)}_{\bar{\lambda}_t} + \sqrt{\gamma^2\lambda_1}\boldsymbol{\epsilon_1} + \sqrt{\gamma^2\lambda_2}\boldsymbol{\epsilon_2} + \cdots + \sqrt{\gamma^2\lambda_t}\boldsymbol{\epsilon_t}
\end{aligned} \tag{19}$$

where $\boldsymbol{\epsilon_1}, \boldsymbol{\epsilon_2}, \ldots, \boldsymbol{\epsilon_t} \sim \mathcal{N}(0, \boldsymbol{I})$. Hence:

$$\begin{aligned}
\boldsymbol{x_t} &\sim \mathcal{N}\left(\boldsymbol{x_0} - \bar{\lambda}_t\boldsymbol{\Delta}, \gamma^2\left(\lambda_1 + \lambda_2 + \cdots + \lambda_t\right)\boldsymbol{I}\right) \\
&\sim \mathcal{N}\left(\boldsymbol{x_0} - \bar{\lambda}_t\boldsymbol{\Delta}, \gamma^2\bar{\lambda}_t\boldsymbol{I}\right),
\end{aligned} \tag{20}$$

Therefore, the cumulative Gaussian transition in the forward process can be defined as in Equation (5) and, when $\gamma = 0$, it collapses into a Dirac delta function.

Cumulative sum of weights λ_t Each weight λ_t, used to control the variance and amount of residual to be removed in each diffusion step, is computed as $\lambda_t = \beta_t - \beta_{t-1}$, with β_t representing the transition at forward step t between original and corrupted data in the Markov chain. Consequently, the cumulative sum of weights λ_t from the initial timestep $t = 1$ up to timestep $t = \tau$ is given as follows:

$$\bar{\lambda}_\tau = \sum_{t=1}^{\tau} \lambda_t = \sum_{t=1}^{\tau} (\beta_t - \beta_{t-1}) = \beta_\tau - \beta_0 \tag{21}$$

Distribution of the last latent variable $q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = q(\boldsymbol{x_T}|\boldsymbol{y_0})$. Given that the RDIM forward process is designed to align with a forward model that converts the data, $\boldsymbol{x_0}$, into the corresponding corrupted version, $\boldsymbol{y_0}$, the residual, $\boldsymbol{\Delta}$, should be fully removed from $\boldsymbol{x_0}$ at the end of the forward process, i.e., after exactly T timesteps. This ensures that the last latent variable, $\boldsymbol{x_T}$, will coincide exactly with the corrupted data, $\boldsymbol{y_0}$, when the forward process is deterministic, and will converge to a noisy sample centered at $\boldsymbol{y_0}$ when the forward process is stochastic. Hence, considering Equation (20), the last latent variable, $\boldsymbol{x_T}$, of the forward process can be sampled as:

$$\begin{aligned} \boldsymbol{x_T} &\sim \mathcal{N}\left(\boldsymbol{x_0} - \bar{\lambda}_T \boldsymbol{\Delta}, \gamma^2 \bar{\lambda}_T \boldsymbol{I}\right) \\ &\sim \mathcal{N}\left(\boldsymbol{x_0} - \bar{\lambda}_T \left(\boldsymbol{x_0} - \boldsymbol{y_0}\right), \gamma^2 \bar{\lambda}_T \boldsymbol{I}\right) \\ &\sim \mathcal{N}\left(\boldsymbol{x_0}\left(1 - \bar{\lambda}_T\right) + \bar{\lambda}_T \boldsymbol{y_0}, \gamma^2 \bar{\lambda}_T \boldsymbol{I}\right). \end{aligned} \tag{22}$$

Logically, to ensure the aforementioned condition of centering the distribution $q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ on the corrupted data, $\boldsymbol{y_0}$, the cumulative sum of weights λ_t over the T timesteps must satisfy $\bar{\lambda}_T = 1$. This imposes that $\beta_0 = 0$ and $\beta_T = 1$, since $\bar{\lambda}_T = \beta_T - \beta_0$, as mentioned above. Accordingly:

$$\boldsymbol{x_T} \sim \mathcal{N}\left(\boldsymbol{y_0}, \gamma^2 \boldsymbol{I}\right). \tag{23}$$

This formulation assures that the residual $\boldsymbol{\Delta}$ is fully removed after exactly T timesteps ($\bar{\lambda}_t = 1$ only when $t = T$) and that the distribution $q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ is centered at the corrupted data, $\boldsymbol{y_0}$. As a result of this deliberate design choice, $q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = q(\boldsymbol{x_T}|\boldsymbol{y_0})$ holds exactly at $t = T$. In addition, when $\gamma = 0$, the Gaussian collapses into a Dirac delta function centered at $\boldsymbol{y_0}$, thereby the final latent variable, $\boldsymbol{x_T}$, coincides exactly with the corrupted data, i.e., $\boldsymbol{x_T} = \boldsymbol{y_0}$.

Additionally, the β-schedule defined in Equation (16) is designed to impose $\beta_0 = 0$ and $\beta_T = 1$, thus satisfying the aforementioned requirements. In particular, the cumulative sum of weights λ_t is $\bar{\lambda}_t = \beta_t$ when $\beta_0 = 0$ (see Equation (21)). Figure 2 showcases the progression of the weights β_t and λ_t across timesteps. If $p = 1.0$, the β-schedule is linear and λ_t is constant, resulting in uniform fractions of Δ removed along the forward process.

Accordingly, under this condition of $\beta_0 = 0$, the cumulative forward transition distribution, $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$, expressed in Equation (20) can be further simplified to:

$$\boldsymbol{x_t} \sim \mathcal{N}\left(\boldsymbol{x_0} - \beta_t \boldsymbol{\Delta}, \gamma^2 \beta_t \boldsymbol{I}\right). \tag{24}$$

A.2 Reverse process transition distribution $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$

The reverse process involves computing the reverse transition, which is defined as the Gaussian distribution in Equation (7) and is designed to preserve the marginal $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ in Equation (5). Considering that Gaussian distributions exhibit the property that their conditional means are linear combinations of the conditioning variables (see Lemma B.2), then the mean $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ of $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$ can be expressed as a linear interpolation between $\boldsymbol{x_t}$, $\boldsymbol{x_0}$, and $\boldsymbol{\Delta}$. Particularly, to match the form of the forward process cumulative transition, $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$, the mean $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ is assumed to be a linear combination between $(\boldsymbol{x_0} - \beta_t \boldsymbol{\Delta})$ and $\boldsymbol{x_t}$:

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}} = a\left(\boldsymbol{x_0} - \beta_t \boldsymbol{\Delta}\right) + b\boldsymbol{x_t}, \tag{25}$$

where a and b are constants.

Following, given $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ and the formulation assumed for $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$, then $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ can be defined by leveraging a property of marginal and conditional Gaussians (see Lemma B.1):

$$\begin{aligned} q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0}, \boldsymbol{\Delta}) &= \mathcal{N}\left(\boldsymbol{x_{t-1}} \middle| b\left(\boldsymbol{x_0} - \beta_t \boldsymbol{\Delta}\right) + a\left(\boldsymbol{x_0} - \beta_t \boldsymbol{\Delta}\right), \tilde{\sigma}_t^2 \boldsymbol{I} + b\gamma^2 \beta_t \boldsymbol{I} b\right) \\ &= \mathcal{N}\left(\boldsymbol{x_{t-1}} \middle| \left(\boldsymbol{x_0} - \beta_t \boldsymbol{\Delta}\right)(a + b), \left(\tilde{\sigma}_t^2 + \gamma^2 \beta_t b^2\right) \boldsymbol{I}\right). \end{aligned} \tag{26}$$

Recalling that $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = \mathcal{N}\left(\boldsymbol{x_t}|\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}, \gamma^2\beta_t\boldsymbol{I}\right)$ is being enforced, the cumulative Gaussian transition to obtain $\boldsymbol{x_{t-1}}$ given $\boldsymbol{x_0}$ and $\boldsymbol{\Delta}$ is also defined as:

$$q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = \mathcal{N}\left(\boldsymbol{x_{t-1}}|\boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta}, \gamma^2\beta_{t-1}\boldsymbol{I}\right). \tag{27}$$

Accordingly, to ensure that the designed reverse transition preserves the marginal $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$, the following equality must be satisfied:

$$\mathcal{N}\left(\boldsymbol{x_{t-1}}\middle|\left(\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}\right)(a+b), \left(\tilde{\sigma}_t^2 + \gamma^2\beta_t b^2\right)\boldsymbol{I}\right) = \mathcal{N}\left(\boldsymbol{x_{t-1}}|\boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta}, \gamma^2\beta_{t-1}\boldsymbol{I}\right), \tag{28}$$

and thus a and b can be computed by solving the following system of equations:

$$\begin{cases} \left(\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}\right)(a+b) = \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} \\ \tilde{\sigma}_t^2 + \gamma^2\beta_t b^2 = \gamma^2\beta_{t-1} \end{cases} \Leftrightarrow \begin{cases} a = 1 + \frac{\lambda_t\boldsymbol{\Delta}}{\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}} - \sqrt{\frac{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}{\gamma^2\beta_t}} \\ b = \sqrt{\frac{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}{\gamma^2\beta_t}} \end{cases}. \tag{29}$$

Consequently, the mean of each reverse transition, $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$, is given as:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}} &= a\left(\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}\right) + b\boldsymbol{x_t} \\ &= \left(1 + \frac{\lambda_t\boldsymbol{\Delta}}{\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}} - \sqrt{\frac{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}{\gamma^2\beta_t}}\right)\left(\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}\right) + \sqrt{\frac{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}{\gamma^2\beta_t}}\boldsymbol{x_t} \\ &= \boldsymbol{x_0} - \beta_t\boldsymbol{\Delta} + \lambda_t\boldsymbol{\Delta} - \sqrt{\frac{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}{\gamma^2\beta_t}}\left(\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}\right) + \sqrt{\frac{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}{\gamma^2\beta_t}}\boldsymbol{x_t} \\ &= \boldsymbol{x_0} - \beta_t\boldsymbol{\Delta} + \left(\beta_t - \beta_{t-1}\right)\boldsymbol{\Delta} + \sqrt{\frac{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}{\gamma^2\beta_t}}\left(\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}\right) \\ &= \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}\left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right), \end{aligned} \tag{30}$$

where, in particular, singularities can occur for $\gamma = 0$. Therefore, for $\gamma \neq 0$, the mean of the reverse process transition distribution that preserves the marginal $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ is given as:

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} = \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}\left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right). \tag{31}$$

Essentially, the mean, $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$, is chosen to ensure that $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta}) = \mathcal{N}\left(\boldsymbol{x_t}|\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}, \gamma^2\beta_t\boldsymbol{I}\right)$ is satisfied for all $t \in \{1, 2, \ldots, T\}$. Meanwhile, the variance $\tilde{\sigma}_t^2$ is set equal to the variance of the ResShift reverse transition (see Appendix A.3), thus $\tilde{\sigma}_t^2 = \gamma^2\frac{\beta_{t-1}}{\beta_t}\lambda_t = \tilde{\lambda}_t$.

Relationship between $\boldsymbol{x_t}$, $\boldsymbol{x_0}$, $\boldsymbol{\Delta}$, and $\boldsymbol{\epsilon}$. Considering the marginal $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ and $\gamma \neq 0$, a relationship between $\boldsymbol{x_t}$, $\boldsymbol{x_0}$, $\boldsymbol{\Delta}$, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ can be derived from the reparameterization trick:

$$\begin{aligned} q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta}) &= \mathcal{N}\left(\boldsymbol{x_t}|\boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}, \gamma^2\beta_t\boldsymbol{I}\right) \\ &\Rightarrow \boldsymbol{x_t} = \boldsymbol{x_0} - \beta_t\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_t}\boldsymbol{\epsilon} \\ &\Leftrightarrow \boldsymbol{\epsilon} = \frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}, \end{aligned} \tag{32}$$

This expression exactly matches the term between parentheses in the mean of the reverse process transition distribution for $\gamma \neq 0$, in Equation (31). Accordingly, the mean can be rewritten as:

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} = \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}\boldsymbol{\epsilon}, \tag{33}$$

which structurally matches the reparameterized form of the marginal $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0}, \boldsymbol{\Delta})$, exhibiting the same functional form and differing only in the variance term. This highlights that, when $\gamma \neq 0$, the reverse transition is aligned with cumulative transitions and can be leveraged to efficiently sample any state at an arbitrary timestep.

Reverse transition with $\gamma = 0$. Particularly, for $\gamma = 0$, the forward process cumulative transition, defined as a Gaussian distribution, degenerates into a Dirac delta function (see also Appendix A.1). Consequently, for $\gamma = 0$, Lemma B.1 is not applicable. In fact, in this case, the forward process effectively becomes a linear interpolation between $\boldsymbol{x_0}$ and $\boldsymbol{y_0}$. Logically, when $\gamma = 0$, it follows that the reverse process simply needs to invert this deterministic process. However, the continuity of the mean, $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$, should be assured at $\gamma = 0$, i.e., $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma=0} = \lim_{\gamma\to 0} \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0}$.

Considering Equation (24) in Appendix A.1, it follows $\lim_{\gamma\to 0} \boldsymbol{x_t} = \boldsymbol{x_0} - \beta_t\boldsymbol{\Delta}$, which implies that $\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta} \to \boldsymbol{0}$ as $\gamma \to 0$. Accordingly, given $\tilde{\sigma}_t^2 = \gamma^2\frac{\beta_{t-1}}{\beta_t}\lambda_t$, then $\lim_{\gamma\to 0} \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} = \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta}$. As a result, to ensure the continuity of the mean $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ at $\gamma = 0$, the Gaussian transition $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$ is assumed to collapse into a Dirac delta function centered at $\boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta}$. Hence, for $\gamma = 0$, the mean of the reverse process transition distribution is defined as:

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma=0} = \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta}. \tag{34}$$

Notably, this formulation of $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma=0}$ matches the mean of the cumulative forward transition, $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_0}, \boldsymbol{\Delta})$ (see Appendix A.1), showing that the reverse process, when $\gamma = 0$, reduces to a linear interpolation between $\boldsymbol{y_0}$ and $\boldsymbol{x_0}$ (inverse of the deterministic forward process). Additionally, it aligns with the concept of cumulative transitions, which is paramount for long-range transitions (see Section 2.5). In essence, the mean, $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$, is expressed as in Equation (8) and is continuous at $\gamma = 0$. Nonetheless, the γ constant hyperparameter is immutable in practice, i.e., set only once for each model instance, thereby no discontinuity issues would ever arise due to γ (see Appendix A.4).

A.3 Reverse transition with $\tilde{\sigma}_t^2 = \gamma^2\frac{\beta_{t-1}}{\beta_t}\lambda_t$ (ResShift variance, $\tilde{\lambda}_t$)

In particular, if the reverse process transition variance, $\tilde{\sigma}_t^2$, is set to be the same as in ResShift, $\tilde{\lambda}_t = \gamma^2\frac{\beta_{t-1}}{\beta_t}\lambda_t$, then the mean, $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0}$, reduces to:

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} &= \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \tilde{\sigma}_t^2}\left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right) \\
&= \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \gamma^2\frac{\beta_{t-1}}{\beta_t}\lambda_t}\left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right) \\
&= \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\frac{\gamma^4\beta_t\beta_{t-1} - \gamma^4\beta_{t-1}\left(\beta_t - \beta_{t-1}\right)}{\gamma^2\beta_t}}\left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right) \\
&= \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \frac{\sqrt{\gamma^4\beta_{t-1}^2}\left(\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}\right)}{\gamma^2\beta_t} \\
&= \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \frac{\beta_{t-1}\boldsymbol{x_t} - \beta_{t-1}\boldsymbol{x_0} + \beta_t\beta_{t-1}\boldsymbol{\Delta}}{\beta_t} \\
&= \frac{\beta_{t-1}}{\beta_t}\boldsymbol{x_t} + \boldsymbol{x_0}\left(1 - \frac{\beta_{t-1}}{\beta_t}\right) \\
&= \frac{\beta_{t-1}}{\beta_t}\boldsymbol{x_t} + \frac{\lambda_t}{\beta_t}\boldsymbol{x_0},
\end{aligned} \tag{35}$$

and thus the distribution $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})_{\gamma\neq 0}$ becomes:

$$q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})_{\gamma\neq 0} = \mathcal{N}\left(\boldsymbol{x_{t-1}}\middle|\frac{\beta_{t-1}}{\beta_t}\boldsymbol{x_t} + \frac{\lambda_t}{\beta_t}\boldsymbol{x_0}, \tilde{\lambda}_t\boldsymbol{I}\right), \tag{36}$$

which is exactly the ResShift reverse transition distribution. In essence, if the RDIM reverse transition variance, $\tilde{\sigma}_t^2$, is set to be the same as in ResShift, then $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ will match the mean of the ResShift reverse transition. Accordingly, RDIM reduces to ResShift for this specific variance, revealing that ResShift is a particular case of RDIM.

Alternatively, for $\gamma \neq 0$, if the variance is set to $\tilde{\sigma}_t^2 = 0$, then there are no stochastic terms involved when traversing the reverse trajectory, as $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})_{\gamma\neq 0}$ degenerates into a δ-distribution

and avoids sampling random noise (given Equations (7) and (31)). Consequently, the reverse process becomes deterministic. Therefore, a constant hyperparameter, $\eta \in [0, 1]$, can be introduced to interpolate between a deterministic and stochastic reverse process when $\gamma \neq 0$, thus allowing control over the variability in the reverse trajectory (see Equation (9)). Specifically, when $\eta = 0$, the Gaussian collapses into a Dirac delta function.

Absence of non-real square roots. From Equation (9), it follows that to avoid a non-real square root, when $\gamma \neq 0$, the condition $\gamma^2 \beta_{t-1} \geq \eta^2 \tilde{\lambda}_t$ must be satisfied. Considering $\tilde{\lambda}_t = \gamma^2 \frac{\beta_{t-1}}{\beta_t} \lambda_t$, then:

$$\gamma^2 \beta_{t-1} \geq \eta^2 \tilde{\lambda}_t \Leftrightarrow 1 \geq \eta^2 \frac{\lambda_t}{\beta_t} \Leftrightarrow 1 \geq \eta^2 \left(1 - \frac{\beta_{t-1}}{\beta_t}\right). \tag{37}$$

Recalling that $\eta \in [0, 1]$, then $0 \leq \eta^2 \leq 1$. Moreover, since $0 \leq \beta_{t-1} \leq \beta_t$, it follows that $0 \leq \frac{\beta_{t-1}}{\beta_t} \leq 1$, which in turn ensures that the term inside parentheses meets the condition $1 - \frac{\beta_{t-1}}{\beta_t} \leq 1$. Consequently, the product of these two terms is always less than or equal to 1, and thus the Inequality (37) is satisfied for all $\eta \in [0, 1]$ and $t \in \{1, 2, \ldots, T\}$.

A.4 Training objective

During inference, $\boldsymbol{x_0}$ and $\boldsymbol{\Delta}$ are unknown, thus sampling from the true reverse transition distribution, $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$, is not possible. Therefore, a learnable parametric model, $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})$, defined as a Gaussian distribution, is introduced to approximate $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})$. Particularly, an accurate estimation is required to ensure precise reconstruction of the data, $\boldsymbol{x_0}$, at inference. This approximation is achieved by minimizing the KL divergence between both distributions, while accounting for all timesteps:

$$\theta^* = \arg\min_\theta D_{\mathrm{KL}}(q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta}) \| p_\theta(\boldsymbol{x_{1:T}}|\boldsymbol{y_0})), \tag{38}$$

where θ^* denotes the optimal parameters. In fact, this objective of minimizing the KL divergence in Equation (38) is equivalent to minimizing the negative variational lower bound (VLB) on the conditional log-likelihood. This is the RDIM objective function and it can expanded further:

$$\begin{aligned}
\mathcal{L}(\theta) &= \mathbb{E}_{q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\log \left(\frac{q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta})}{p_\theta(\boldsymbol{x_{0:T}}|\boldsymbol{y_0})} \right) \right] \\
&= \mathbb{E}_{q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\log \left(\frac{q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta}) \prod_{t=2}^{T} q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})}{p(\boldsymbol{x_T}|\boldsymbol{y_0}) \prod_{t=1}^{T} p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})} \right) \right] \\
&= \mathbb{E}_{q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \Bigg[\log \left(\frac{q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta})}{p(\boldsymbol{x_T}|\boldsymbol{y_0})} \right) \\
&\quad + \log \left(\prod_{t=2}^{T} \frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})}{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})} \right) - \log \left(p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1}, \boldsymbol{y_0}) \right) \Bigg] \\
&= \mathbb{E}_{q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\log \left(\frac{q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta})}{p(\boldsymbol{x_T}|\boldsymbol{y_0})} \right) \right] \\
&\quad + \mathbb{E}_{q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\sum_{t=2}^{T} \log \left(\frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})}{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})} \right) \right] \\
&\quad - \mathbb{E}_{q(\boldsymbol{x_1}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\log \left(p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1}, \boldsymbol{y_0}) \right) \right] \\
&= D_{\mathrm{KL}}(q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta}) \| p(\boldsymbol{x_T}|\boldsymbol{y_0})) \\
&\quad + \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x_{t-1}}, \boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\log \left(\frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})}{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})} \right) \right] \\
&\quad - \mathbb{E}_{q(\boldsymbol{x_1}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\log \left(p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1}, \boldsymbol{y_0}) \right) \right] \\
&= D_{\mathrm{KL}}(q(\boldsymbol{x_T}|\boldsymbol{x_0}, \boldsymbol{\Delta}) \| p(\boldsymbol{x_T}|\boldsymbol{y_0})) \\
&\quad + \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\mathbb{E}_{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})} \left[\log \left(\frac{q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta})}{p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})} \right) \right] \right]
\end{aligned} \tag{39}$$

$$
\begin{aligned}
&\quad - \mathbb{E}_{q(\boldsymbol{x_1}|\boldsymbol{x_0},\boldsymbol{\Delta})}\left[\log\left(p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1},\boldsymbol{y_0})\right)\right] \\
&= \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x_T}|\boldsymbol{x_0},\boldsymbol{\Delta})\|p(\boldsymbol{x_T}|\boldsymbol{y_0}))}_{\mathcal{L}_T} \\
&\quad + \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x_t}|\boldsymbol{x_0},\boldsymbol{\Delta})}\left[D_{\mathrm{KL}}(q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{x_0},\boldsymbol{\Delta})\|p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{y_0}))\right]}_{\mathcal{L}_{t-1}} \\
&\quad \underbrace{-\mathbb{E}_{q(\boldsymbol{x_1}|\boldsymbol{x_0},\boldsymbol{\Delta})}\left[\log\left(p_\theta(\boldsymbol{x_0}|\boldsymbol{x_1},\boldsymbol{y_0})\right)\right]}_{\mathcal{L}_0} = \mathcal{L}_T + \mathcal{L}_{1:T-1} + \mathcal{L}_0.
\end{aligned}
$$

Hence, analogous to DDPMs, the RDIM objective function, $\mathcal{L}(\theta)$, decomposes into $\mathcal{L}_T$ (prior matching term), $\mathcal{L}_{1:T-1}$ (consistency terms), and $\mathcal{L}_0$ (reconstruction term).

Prior matching term $\mathcal{L}_T$. The term $\mathcal{L}_T$ is minimized when the prior, $p(\boldsymbol{x_T}|\boldsymbol{y_0})$, matches the true distribution of the last latent variable, $q(\boldsymbol{x_T}|\boldsymbol{x_0},\boldsymbol{\Delta}) = q(\boldsymbol{x_T}|\boldsymbol{y_0}) = \mathcal{N}(\boldsymbol{y_0},\gamma^2 I)$. Accordingly, $p(\boldsymbol{x_T}|\boldsymbol{y_0})$ is fixed to such a Gaussian distribution, which is parameterized by constants and involves no learnable parameters. Therefore, $\mathcal{L}_T$ is constant with respect to the model parameters, θ, and is minimized, i.e., $\mathcal{L}_T = 0$. Consequently, this term can be excluded from the optimization objective, unlike the terms $\mathcal{L}_{0:T-1}$, which explicitly depend on θ through the parameterized distribution p_θ.

Consistency terms $\mathcal{L}_{1:T-1}$. The terms $\mathcal{L}_{1:T-1}$ enforce that the learnable parametric model, $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{y_0})$, accurately approximates the true reverse transition, $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{x_0},\boldsymbol{\Delta})$. This fundamentally ensures that the model learns to refine the data at intermediate timesteps, leading to consistency in the reconstruction.

The true reverse transition distribution is known in closed form (see Section 2.4 along with Appendices A.2 and A.3), having mean and variance parameterized as:

$$
\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}} = \begin{cases} \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta}, & \text{if } \gamma = 0, \\ \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}\left(\frac{\boldsymbol{x_t}-\boldsymbol{x_0}+\beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right), & \text{if } \gamma \neq 0, \end{cases} \tag{40}
$$

and

$$
\tilde{\sigma}_t^2 = \begin{cases} \tilde{\lambda}_t, & \text{if } \gamma = 0, \\ \eta^2\tilde{\lambda}_t, & \text{if } \gamma \neq 0, \end{cases} \tag{41}
$$

where $\tilde{\lambda}_t = \gamma^2\frac{\beta_{t-1}}{\beta_t}\lambda_t$.

Given that $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{y_0})$ is defined as a Gaussian distribution with mean $\boldsymbol{\mu_\theta}(\boldsymbol{x_t},\boldsymbol{y_0},t)$ and variance $\sigma_\theta^2(\boldsymbol{x_t},\boldsymbol{y_0},t)$, to minimize the KL divergence of each term $\mathcal{L}_{1:T-1}$, the mean and variance of the parametric model should approximate $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ and $\tilde{\sigma}_t^2$, respectively. Particularly, the variance of $q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{x_0},\boldsymbol{\Delta})$ does not have learnable parameters because it is defined in terms of constant hyperparameters, which are known. Therefore, $\sigma_\theta^2(\boldsymbol{x_t},\boldsymbol{y_0},t)$ can be fixed to equal exactly $\tilde{\sigma}_t^2$, as expressed in Equation (11). Following, each term $\mathcal{L}_{1:T-1}$ is computed by applying the closed-form expression for the KL divergence between two d-dimensional multivariate Gaussian distributions, yielding:

$$
\begin{aligned}
&D_{\mathrm{KL}}(q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{x_0},\boldsymbol{\Delta})\|p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t},\boldsymbol{y_0})) \\
&= \frac{1}{2}\Bigg(\log\left(\frac{|\sigma_\theta^2(\boldsymbol{x_t},\boldsymbol{y_0},t)\boldsymbol{I}|}{|\tilde{\sigma}_t^2\boldsymbol{I}|}\right) - d + \mathrm{tr}\left(\left(\sigma_\theta^2(\boldsymbol{x_t},\boldsymbol{y_0},t)\boldsymbol{I}\right)^{-1}\tilde{\sigma}_t^2\boldsymbol{I}\right) \\
&\quad + \left(\boldsymbol{\mu_\theta}(\boldsymbol{x_t},\boldsymbol{y_0},t) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}\right)^\top\left(\sigma_\theta^2(\boldsymbol{x_t},\boldsymbol{y_0},t)\boldsymbol{I}\right)^{-1}\left(\boldsymbol{\mu_\theta}(\boldsymbol{x_t},\boldsymbol{y_0},t) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}\right)\Bigg) \\
&= \frac{1}{2}\Bigg(\log\left(\frac{|\tilde{\sigma}_t^2\boldsymbol{I}|}{|\tilde{\sigma}_t^2\boldsymbol{I}|}\right) - d + \mathrm{tr}\left(\left(\tilde{\sigma}_t^2\boldsymbol{I}\right)^{-1}\tilde{\sigma}_t^2\boldsymbol{I}\right) \\
&\quad + \left(\boldsymbol{\mu_\theta}(\boldsymbol{x_t},\boldsymbol{y_0},t) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}\right)^\top\left(\tilde{\sigma}_t^2\boldsymbol{I}\right)^{-1}\left(\boldsymbol{\mu_\theta}(\boldsymbol{x_t},\boldsymbol{y_0},t) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}\right)\Bigg)
\end{aligned} \tag{42}
$$

$$= \frac{1}{2}\left(\left(\frac{1}{\tilde{\sigma}_t^2}\right) \left(\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}\right)^\top \left(\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}\right) \right)$$
$$= \frac{1}{2\tilde{\sigma}_t^2} \left\| \boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}} \right\|^2,$$

where $|\cdot|$ denotes the determinant of a matrix, and $\mathrm{tr}(\cdot)$ is the trace of a matrix. Notably, minimizing the KL divergence effectively reduces to decreasing the difference between the means $\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)$ and $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$.

However, this is only valid with $\gamma \neq 0$ and $\eta \neq 0$. In contrast, when either $\gamma = 0$ or $\eta = 0$, the true reverse transition Gaussian collapses into a Dirac delta function:

$$q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta}) = \begin{cases} \delta\left(\boldsymbol{x_{t-1}} - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma=0}\right), & \text{if } \gamma = 0, \quad \text{(A)} \\ \delta\left(\boldsymbol{x_{t-1}} - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0}\right), & \text{if } \gamma \neq 0 \text{ and } \eta = 0, \quad \text{(B)} \\ \mathcal{N}(\boldsymbol{x_{t-1}}|\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0}, \tilde{\sigma}^2_{\boldsymbol{t}|\gamma\neq 0}\boldsymbol{I}), & \text{if } \gamma \neq 0 \text{ and } \eta \neq 0, \quad \text{(C)} \end{cases} \tag{43}$$

where A, B, and C correspond to the cases of $\gamma = 0$, ($\gamma \neq 0$ and $\eta = 0$), and ($\gamma \neq 0$ and $\eta \neq 0$), respectively.

The KL divergence between two Dirac delta functions is not defined in the conventional sense due to their singular nature, but it can be analyzed through limiting behavior. Two delta functions centered at different points have infinite divergence, thereby the KL divergence of each term $\mathcal{L}_{1:T-1}$ tends to infinity, when $\gamma = 0$ or $\eta = 0$, unless $\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) = \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$:

$$D_{\mathrm{KL}}(q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta}) \| p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0}))$$
$$= \begin{cases} 0, & \text{if (A and } \boldsymbol{\mu_{\theta|\gamma=0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) = \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma=0}\text{)} \\ & \text{or (B and } \boldsymbol{\mu_{\theta|\gamma\neq 0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) = \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0}\text{)}, \\ \infty, & \text{if (A and } \boldsymbol{\mu_{\theta|\gamma=0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) \neq \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma=0}\text{)} \\ & \text{or (B and } \boldsymbol{\mu_{\theta|\gamma\neq 0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) \neq \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0}\text{)}, \\ \frac{1}{2\tilde{\sigma}^2_{t|\gamma\neq 0}} \left\| \boldsymbol{\mu_{\theta|\gamma\neq 0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} \right\|^2, & \text{if C.} \end{cases} \tag{44}$$

For the Dirac delta cases, where either $\gamma = 0$ or $\eta = 0$, to avoid an infinite loss, the only choice is to force $\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) = \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$. However, directly optimizing under such a hard constraint is infeasible in practice, as it provides no gradient information unless the condition is already satisfied. To circumvent this, a relaxed proxy objective is adopted, mirroring the approach used in the Gaussian case. Specifically, it minimizes half of the squared Euclidean distance between $\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)$ and $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$. This mean-matching proxy loss serves as a differentiable surrogate that naturally encourages the model to align the means and can be interpreted as the limiting case of the KL divergence when the variance tends to zero. Consequently, the reduction of the KL divergence to mean matching holds for all scenarios of γ and η.

Moreover, considering the formulation of $\tilde{\boldsymbol{\mu}}_{\boldsymbol{t}}$ given in Equation (40) and since at every timestep, t, in the reverse process, only the exact values of $\boldsymbol{x_0}$ and $\boldsymbol{\Delta}$ are unknown, then $\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)$ can be defined as in Equation (12). In this definition of $\boldsymbol{\mu_\theta}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right)$, the only components dependent on the parameters θ are $\hat{\boldsymbol{x}}_{\boldsymbol{0}}$ and $\hat{\boldsymbol{\Delta}}$. Since, $\boldsymbol{\Delta}$ can be estimated from $\boldsymbol{x_0}$ and $\boldsymbol{y_0}$, then the model solely needs to predict $\boldsymbol{x_0}$. Hence, $\hat{\boldsymbol{x}}_{\boldsymbol{0}} = f_\theta(\boldsymbol{x_t}, \boldsymbol{y_0}, t)$ denotes the $\boldsymbol{x_0}$ prediction from a neural network given $\boldsymbol{x_t}$, $\boldsymbol{y_0}$, and timestep t. Meanwhile, $\hat{\boldsymbol{\Delta}} = \hat{\boldsymbol{x}}_{\boldsymbol{0}} - \boldsymbol{y_0}$ represents the $\boldsymbol{\Delta}$ estimation, computed from the $\boldsymbol{x_0}$ prediction and the known $\boldsymbol{y_0}$. The remaining components are fixed hyperparameters and $\boldsymbol{x_t}$, which are known for every reverse transition from $\boldsymbol{x_t}$ at any timestep, t. In essence, the approximate reverse transition, $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})$, is modeled as a Gaussian whose mean is computed using a neural network that predicts $\boldsymbol{x_0}$. Accordingly, the KL divergence of each term $\mathcal{L}_{1:T-1}$ can be further expanded as:

$$D_{\mathrm{KL}}(q(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{x_0}, \boldsymbol{\Delta}) \| p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0}))$$
$$= \begin{cases} \frac{1}{2} \left\| \boldsymbol{\mu_{\theta|\gamma=0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma=0} \right\|^2, & \text{if A,} \\ \frac{1}{2} \left\| \boldsymbol{\mu_{\theta|\gamma\neq 0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} \right\|^2, & \text{if B,} \\ \frac{1}{2\tilde{\sigma}^2_{t|\gamma\neq 0}} \left\| \boldsymbol{\mu_{\theta|\gamma\neq 0}}\left(\boldsymbol{x_t}, \boldsymbol{y_0}, t\right) - \tilde{\boldsymbol{\mu}}_{\boldsymbol{t}|\gamma\neq 0} \right\|^2, & \text{if C,} \end{cases}$$

$$
\begin{aligned}
&= \begin{cases} \frac{1}{2}\|\boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} - (\hat{\boldsymbol{x}}_{\boldsymbol{0}} - \beta_{t-1}\hat{\boldsymbol{\Delta}})\|^2, & \text{if A}, \\ \frac{1}{2}\left\| \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}\left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right) \right. & \\ \quad \left. - \left(\hat{\boldsymbol{x}}_{\boldsymbol{0}} - \beta_{t-1}\hat{\boldsymbol{\Delta}} + \sqrt{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}\left(\frac{\boldsymbol{x_t} - \hat{\boldsymbol{x}}_{\boldsymbol{0}} + \beta_t\hat{\boldsymbol{\Delta}}}{\sqrt{\gamma^2\beta_t}}\right)\right)\right\|^2, & \text{if B}, \\ \frac{1}{2\eta^2\tilde{\lambda}_t}\left\| \boldsymbol{x_0} - \beta_{t-1}\boldsymbol{\Delta} + \sqrt{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}\left(\frac{\boldsymbol{x_t} - \boldsymbol{x_0} + \beta_t\boldsymbol{\Delta}}{\sqrt{\gamma^2\beta_t}}\right) \right. & \\ \quad \left. - \left(\hat{\boldsymbol{x}}_{\boldsymbol{0}} - \beta_{t-1}\hat{\boldsymbol{\Delta}} + \sqrt{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}\left(\frac{\boldsymbol{x_t} - \hat{\boldsymbol{x}}_{\boldsymbol{0}} + \beta_t\hat{\boldsymbol{\Delta}}}{\sqrt{\gamma^2\beta_t}}\right)\right)\right\|^2, & \text{if C}, \end{cases} \\
&= \begin{cases} \frac{1}{2}\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}} - \beta_{t-1}(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})\|^2, & \text{if A}, \\ \frac{1}{2}\left\| \boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}} - \beta_{t-1}(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}) \right. & \\ \quad \left. + \sqrt{\frac{\beta_{t-1}}{\beta_t}}(\hat{\boldsymbol{x}}_{\boldsymbol{0}} - \boldsymbol{x_0} + \beta_t(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}))\right\|^2, & \text{if B}, \\ \frac{1}{2\eta^2\tilde{\lambda}_t}\left\| \boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}} - \beta_{t-1}(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}) \right. & \\ \quad \left. + \sqrt{\frac{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}{\gamma^2\beta_t}}(\hat{\boldsymbol{x}}_{\boldsymbol{0}} - \boldsymbol{x_0} + \beta_t(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}))\right\|^2, & \text{if C}, \end{cases} \\
&= \begin{cases} \frac{1}{2}\|(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})(1 - \beta_{t-1})\|^2, & \text{if A}, \\ \frac{1}{2}\left\| (\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})(1 - \beta_{t-1}) \right. & \\ \quad \left. + \sqrt{\frac{\beta_{t-1}}{\beta_t}}(\hat{\boldsymbol{x}}_{\boldsymbol{0}} - \boldsymbol{x_0} + \beta_t(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}))\right\|^2, & \text{if B}, \\ \frac{1}{2\eta^2\tilde{\lambda}_t}\left\| (\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})(1 - \beta_{t-1}) \right. & \\ \quad \left. + \sqrt{\frac{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}{\gamma^2\beta_t}}(\hat{\boldsymbol{x}}_{\boldsymbol{0}} - \boldsymbol{x_0} + \beta_t(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}))\right\|^2, & \text{if C}, \end{cases} \\
&= \begin{cases} \frac{1-\beta_{t-1}}{2}\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}\|^2, & \text{if A}, \\ \frac{1}{2}\left\|(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})(1 - \beta_{t-1}) - \sqrt{\frac{\beta_{t-1}}{\beta_t}}(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})(1 - \beta_t)\right\|^2, & \text{if B}, \\ \frac{1}{2\eta^2\tilde{\lambda}_t}\left\|(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})(1 - \beta_{t-1}) - \sqrt{\frac{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}{\gamma^2\beta_t}}(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})(1 - \beta_t)\right\|^2, & \text{if C}, \end{cases} \\
&= \begin{cases} \frac{1-\beta_{t-1}}{2}\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}\|^2, & \text{if A}, \\ \frac{1}{2}\left\|(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})\left(1 - \beta_{t-1} - \sqrt{\frac{\beta_{t-1}}{\beta_t}}(1 - \beta_t)\right)\right\|^2, & \text{if B}, \\ \frac{1}{2\eta^2\tilde{\lambda}_t}\left\|(\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}})\left(1 - \beta_{t-1} - \sqrt{\frac{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}{\gamma^2\beta_t}}(1 - \beta_t)\right)\right\|^2, & \text{if C}, \end{cases} \\
&= \begin{cases} \frac{1-\beta_{t-1}}{2}\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}\|^2, & \text{if } \gamma = 0, \\ \frac{1-\beta_{t-1} - \sqrt{\frac{\beta_{t-1}}{\beta_t}}(1-\beta_t)}{2}\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}\|^2, & \text{if } \gamma \neq 0 \text{ and } \eta = 0, \\ \frac{1-\beta_{t-1} - \sqrt{\frac{\gamma^2\beta_{t-1} - \eta^2\tilde{\lambda}_t}{\gamma^2\beta_t}}(1-\beta_t)}{2\eta^2\tilde{\lambda}_t}\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}\|^2, & \text{if } \gamma \neq 0 \text{ and } \eta \neq 0, \end{cases} \\
&= \omega_t(\gamma, \eta, t)\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}\|^2.
\end{aligned} \tag{45}
$$

Therefore, irrespective of the specific values of γ and η, each consistency term $\mathcal{L}_{1:T-1}$ ultimately reduces to the expectation of a weighted squared Euclidean distance between the original data $\boldsymbol{x_0}$ and its prediction, where the expectation is taken over $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$:

$$
\mathcal{L}_{t-1} = \mathbb{E}_{q(\boldsymbol{x_t}|\boldsymbol{x_0},\boldsymbol{\Delta})}\left[\omega_t(\gamma, \eta, t)\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_{\boldsymbol{0}}\|^2\right], \tag{46}
$$

with weights $\omega_t(\cdot)$ defined as a function of γ, η, and t. Essentially, approximating $\hat{\boldsymbol{x}}_\mathbf{0}$ to the original data effectively ensures that $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}_\mathbf{0}, t)$ converges to $\tilde{\boldsymbol{\mu}}_t$. As a result, $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y}_\mathbf{0})$ accurately models $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta})$, which is the primary purpose of the consistency terms $\mathcal{L}_{1:T-1}$.

In particular, due to the relationship between $\boldsymbol{x}_\mathbf{0}$ and $\boldsymbol{\epsilon}$ given in Equation (32), the objective derived in Equation (45) could be converted to predicting noise $\boldsymbol{\epsilon}$ similar to DDPMs (Ho et al., 2020). However, this reformulation of the objective would not be possible with a deterministic forward process ($\gamma = 0$), as it works only for cases where noise was added during the forward process ($\gamma \neq 0$). Hence, having the neural network predict $\boldsymbol{x}_\mathbf{0}$ directly is preferred for broader applicability and improved generalizability.

Notably, the mean is continuous at $\gamma = 0$ (see Appendix A.2), thus there are no problems during gradient computation, such as taking gradients where a function is not differentiable. Nonetheless, for each specific value of γ, the mean is continuous and the γ constant hyperparameter is immutable, i.e., set only once for each model instance, thereby no discontinuity issues would ever arise due to γ.

Reconstruction term $\mathcal{L}_0$. The $\mathcal{L}_0$ term is essentially the expectation of the negative log-likelihood (NLL) of the original data, $\boldsymbol{x}_\mathbf{0}$, conditioned on the first latent variable, $\boldsymbol{x}_\mathbf{1}$, and the corrupted version, $\boldsymbol{y}_\mathbf{0}$, where the expectation is taken over $\boldsymbol{x}_\mathbf{1} \sim q(\boldsymbol{x}_\mathbf{1}|\boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta})$. In essence, it quantifies how well the model can reconstruct $\boldsymbol{x}_\mathbf{0}$ given $\boldsymbol{x}_\mathbf{1}$ and $\boldsymbol{y}_\mathbf{0}$. Since minimizing the NLL encourages the model to output high-probability (accurate) reconstructions, it can be interpreted as a reconstruction loss. Conceptually, this term acts as a final quality check, ensuring that after practically all the diffusion degradation is removed[3] iteratively, the model can accurately reconstruct the original clean data, $\boldsymbol{x}_\mathbf{0}$, from the almost degradation-free input, $\boldsymbol{x}_\mathbf{1}$. It assures that the model not only learns to refine the data at intermediate timesteps, but also produces outputs consistent with the underlying real data distribution conditioned on $\boldsymbol{y}_\mathbf{0}$. As a result, it contributes to aligning the model marginal $p_\theta(\boldsymbol{x}_\mathbf{0}|\boldsymbol{y}_\mathbf{0})$ with the true posterior distribution $q(\boldsymbol{x}_\mathbf{0}|\boldsymbol{y}_\mathbf{0})$ as given in Equation (17). Nonetheless, similar to DDPMs, this term is omitted in practice, since it is implicitly included in a simplified training objective.

Simplified objective function. Since the term $\mathcal{L}_T$ can be excluded from the optimization objective, the loss function in Equation (39) becomes:

$$\begin{aligned}\mathcal{L}(\theta) &= \cancel{\mathcal{L}_T} + \mathcal{L}_{1:T-1} + \mathcal{L}_0 = \sum_{t=2}^{T} \mathcal{L}_{t-1} + \mathcal{L}_0 \\ &= \sum_{t=2}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta})} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta}) \| p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y}_\mathbf{0})) \right] + \mathcal{L}_0.\end{aligned} \tag{47}$$

Following, the term $\mathcal{L}_0$ can be omitted, as it is implicitly included by extending the sum to encompass all timesteps, $t \in \{1, 2, \ldots, T\}$, thereby accounting for the transition from $\boldsymbol{x}_\mathbf{1}$ to $\boldsymbol{x}_\mathbf{0}$:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta})} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta}) \| p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y}_\mathbf{0})) \right], \tag{48}$$

and given Equation (45), then:

$$\begin{aligned}\mathcal{L}(\theta) &= \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta})} \left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta}) \| p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y}_\mathbf{0})) \right] \\ &= \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_\mathbf{0}, \boldsymbol{\Delta})} \left[\omega_t(\gamma, \eta, t) \|\boldsymbol{x}_\mathbf{0} - \hat{\boldsymbol{x}}_\mathbf{0}\|^2 \right].\end{aligned} \tag{49}$$

Considering $\tilde{\lambda}_t = \gamma^2 \frac{\beta_{t-1}}{\beta_t} \lambda_t$, the weights ω_t only depend on the predefined γ, η, and β-schedule constant hyperparameters. In many practical implementations, such as DDPMs, this weighting is

[3]The forward process progressively incorporates degradation and removes $\boldsymbol{\Delta}$. The reverse process removes degradation and reintroduces $\boldsymbol{\Delta}$.

often omitted for all timesteps, finding that this still produces excellent results (Ho et al., 2020; Yue et al., 2023). Therefore, the loss function can be further simplified by excluding the scaling:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{q(\boldsymbol{x_t}|\boldsymbol{x_0},\boldsymbol{\Delta})} \left[\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_\mathbf{0}\|^2 \right], \tag{50}$$

and since evaluating the full sum over all time steps is computationally expensive, a single time step can be sampled per training example. This yields an unbiased estimator of the full objective and significantly improves training efficiency:

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{\boldsymbol{x_0},\boldsymbol{\Delta},t} \left[\|\boldsymbol{x_0} - \hat{\boldsymbol{x}}_\mathbf{0}\|^2 \right], \tag{51}$$

where $\boldsymbol{x_t} \sim q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$, $t \sim \mathcal{U}(1, T)$, and the case $t = 1$ corresponds to $\mathcal{L}_0$. Consequently, the objective function of RDIMs simplifies to a squared Euclidean distance between the original data and its prediction. Notably, RDIM and ResShift lead to the same training objective, further highlighting that ResShift is a particular case of RDIM. This follows from the objective depending only on the marginal distribution $q(\boldsymbol{x_t}|\boldsymbol{x_0}, \boldsymbol{\Delta})$, which both models share. It does not strictly require the forward process to be a Markov chain.

B LEMMAS

This section presents lemmas that support this work. These lemmas provide foundational results and properties that support the main arguments and proofs.

Lemma B.1 (Bishop & Nasrabadi (2006)) *Given a marginal Gaussian distribution for random variable $\boldsymbol{x}$ and a conditional Gaussian distribution for random variable $\boldsymbol{y}$ given $\boldsymbol{x}$ in the form:*

$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu_x}, \boldsymbol{\Sigma_x}\right), \\ p(\boldsymbol{y}|\boldsymbol{x}) &= \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{C}\boldsymbol{x} + \boldsymbol{c}, \boldsymbol{\Sigma_{y|x}}\right), \end{aligned} \tag{52}$$

where $\boldsymbol{\mu_x}$, $\boldsymbol{C}$, and $\mathbf{c}$ are parameters governing the means, while $\boldsymbol{\Sigma_x}$ and $\boldsymbol{\Sigma_{y|x}}$ denote covariance matrices. Then the marginal distribution of $\boldsymbol{y}$ and the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$ are in the form:

$$\begin{aligned} p(\boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{C}\boldsymbol{\mu_x} + \boldsymbol{c}, \boldsymbol{\Sigma_{y|x}} + \boldsymbol{C}\boldsymbol{\Sigma_x}\boldsymbol{C}^\top\right), \\ p(\boldsymbol{x}|\boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{x}\middle|\boldsymbol{\Sigma_{x|y}}\left(\boldsymbol{C}^\top\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}}^{-1}(\boldsymbol{y} - \boldsymbol{c}) + \boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1}\boldsymbol{\mu_x}\right), \boldsymbol{\Sigma_{x|y}}\right), \end{aligned} \tag{53}$$

with $\boldsymbol{\Sigma_{x|y}}$ representing the conditional covariance matrix of $\boldsymbol{x}$ given $\boldsymbol{y}$, defined as:

$$\boldsymbol{\Sigma_{x|y}} = \left(\boldsymbol{\Sigma}_{\boldsymbol{x}}^{-1} + \boldsymbol{C}^\top\boldsymbol{\Sigma}_{\boldsymbol{y}|\boldsymbol{x}}^{-1}\boldsymbol{C}\right)^{-1}. \tag{54}$$

Lemma B.2 (Bishop & Nasrabadi (2006)) *Given a joint Gaussian distribution over random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ of the form:*

$$p\left(\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu_x}\\ \boldsymbol{\mu_y}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma_{xx}} & \boldsymbol{\Sigma_{xy}}\\ \boldsymbol{\Sigma_{yx}} & \boldsymbol{\Sigma_{yy}}\end{bmatrix}\right), \tag{55}$$

where $\boldsymbol{\mu_x}$ and $\boldsymbol{\mu_y}$ are the mean vectors of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, while $\boldsymbol{\Sigma_{xx}}$, $\boldsymbol{\Sigma_{xy}}$, $\boldsymbol{\Sigma_{yx}}$, and $\boldsymbol{\Sigma_{yy}}$ denote covariance matrices. Then the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$ is Gaussian:

$$p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{x}\middle|\boldsymbol{\mu_{x|y}}, \boldsymbol{\Sigma_{x|y}}\right), \tag{56}$$

with the conditional mean and covariance given by:

$$\begin{aligned} \boldsymbol{\mu_{x|y}} &= \boldsymbol{\mu_x} + \boldsymbol{\Sigma_{xy}}\boldsymbol{\Sigma}_{\boldsymbol{yy}}^{-1}(\boldsymbol{y} - \boldsymbol{\mu_y}), \\ \boldsymbol{\Sigma_{x|y}} &= \boldsymbol{\Sigma_{xx}} - \boldsymbol{\Sigma_{xy}}\boldsymbol{\Sigma}_{\boldsymbol{yy}}^{-1}\boldsymbol{\Sigma_{yx}}, \end{aligned} \tag{57}$$

where the expressions follow from the Schur complement. This result shows that the conditional mean of $\boldsymbol{x}$ given $\boldsymbol{y}$ is a linear function of $\boldsymbol{y}$.

C Experimental Details and Additional Results

This section presents experimental details and additional results that complement those discussed in the main text.

C.1 Datasets

Experiments were performed across eight subsets, derived from four public data collections, namely (i) Fluorescence Microscopy Denoising (FMD) dataset (Zhang et al., 2019), (ii) DIVerse 2K Resolution High Quality Images (DIV2K) dataset (Agustsson & Timofte, 2017; Timofte et al., 2017), (iii) Smartphone Image Denoising Dataset (SIDD) (Abdelhamed et al., 2018; 2019), and (iv) Flickr–Faces-HQ (FFHQ) dataset (Karras et al., 2019).

The FMD dataset is specifically designed for Poisson-Gaussian denoising tasks and consists of $12{,}000$ real images acquired from representative biological samples, including bovine pulmonary artery endothelial (BPAE) cells, zebrafish embryos, and mouse brain tissues, using confocal, two-photon, and wide-field modalities. The dataset contains images with multiple noise levels, resulting in several subsets, but only the strongest noise level (labeled raw in Zhang et al. (2019)) subsets are considered, thus prioritizing the most challenging conditions. Solely confocal images were used and mouse images are excluded. Accordingly, the two FMD dataset partitions used are Confocal-BPAE-Raw with $1{,}000$ noisy-clean image pairs and Confocal-Zebrafish-Raw with $1{,}000$ pairs. Moreover, each subset was randomly partitioned into training, test and validation splits, corresponding to 80%, 10%, and 10% of the data, respectively.

DIV2K is a publicly available benchmark dataset originally introduced for the NTIRE 2017 Challenge on Single Image Super-Resolution. It is specifically designed for SR tasks and comprises a collection of $1{,}000$ HR images along with their corresponding LR counterparts. Each HR image in the dataset is paired with several downscaled versions, generated through different degradation operations and scaling factors of 2, 3, and 4. Particularly, only two subsets of DIV2K with unknown degradation operators are used, namely DIV2K-Unknown-$\times 2$ with $1{,}000$ LR-HR image pairs and DIV2K-Unknown-$\times 4$ with $1{,}000$ pairs. Each subset is divided into 800 images used for training, 100 for validation, and 100 for testing. The validation split will be employed to evaluate the performance of the models as the testing split is not available.

The SIDD dataset is specifically designed for image denoising tasks, particularly focusing on real-world noisy images captured with smartphone cameras. The dataset consists of $\approx 30{,}000$ noisy images with their corresponding clean ground truth, from 10 scenes under different lighting conditions and using five representative smartphone cameras, hence spanning a wide range of image types and noise levels. Only images from the SIDD-Medium subset are used, comprising 320 noisy-clean image pairs. Ultimately, SIDD-Medium was randomly partitioned into training, test and validation splits, corresponding to 80%, 10%, and 10% of the data, respectively.

The FFHQ dataset consists of $70{,}000$ HQ human face images, originally created as a benchmark for generative adversarial networks (GANs). It contains faces with considerable variation in terms of age, ethnicity, and image background. In this work, it is used for image inpainting, colorization, and deblurring. For computational efficiency, images were downsampled to a quarter of the original resolution using bicubic interpolation. Subsequently, corrupted-original image pairs were generated, resulting in three task-specific subsets, namely FFHQ-Inpainting, FFHQ-Colorization, and FFHQ-Deblurring. For image inpainting, pixels in the original images are randomly masked and set to zero with probability $p_{\text{mask}} = 0.5$. For colorization, grayscale inputs are obtained by converting the original RGB images to luminance. For deblurring, synthetic blurred images are generated from ground truth images by applying a Gaussian blur with kernel size 15×15 and standard deviation $\sigma = 3.0$. Each subset was randomly partitioned into training, validation, and test splits corresponding to 80%, 10%, and 10% of the data, respectively.

C.2 Network Architecture

RDIM employs a U-Net-based architecture to predict $\hat{\boldsymbol{x}}_{\mathbf{0}}$ at each iteration of the reverse process. As illustrated in Figure 6, the network is composed of encoder, bottleneck, and decoder blocks, with skip connections linking encoder and decoder blocks at matching spatial resolutions. For

SR tasks, an upsample block transforms $\boldsymbol{y_0}$ to match the dimensionality (number of channels and resolution) expected by the network. For other tasks, this layer simplifies to a projection layer. At each iteration, the network is conditioned on a timestep embedding, which is computed with sinusoidal positional encoding and transformed through a small multilayer perceptron (MLP) consisting of a fully connected layer, a Swish activation, and a second fully connected layer. This embedding encodes the current diffusion step, providing information about the position within the reverse process.

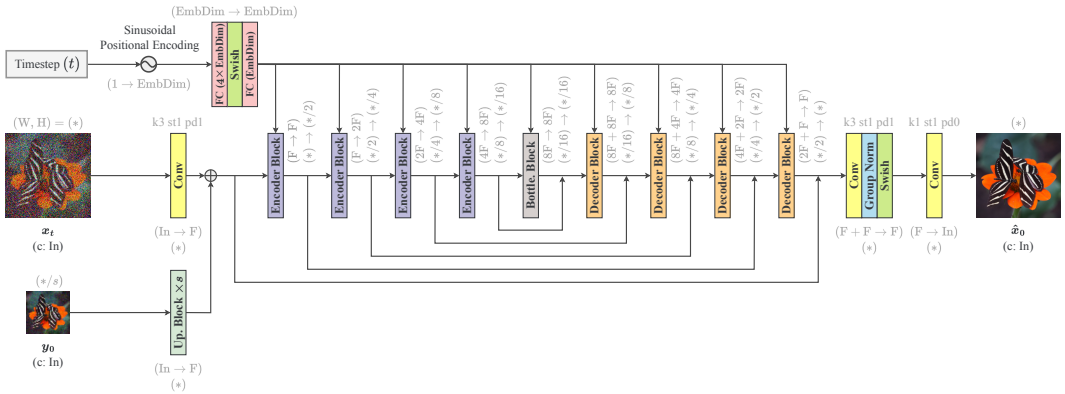


Figure 6: U-Net-based network. In the convolutional layers, the parameters k, st, and pd represent the kernel size, stride, and padding, respectively. Additionally, $(*/s)$ denotes $(W/s, H/s)$, where s is a scale factor ($s > 1$ for SR tasks and $s = 1$ otherwise).

Figure 7 shows the core blocks of the network. Each encoder block consists of multiple residual blocks, each optionally followed by a self-attention block, and concludes with a downsample block to reduce spatial resolution. Bottleneck blocks operate at the lowest spatial resolution and consist of multiple residual blocks interleaved with self-attention blocks. Decoder blocks consist of multiple residual blocks, each optionally followed by a self-attention block, and conclude with an upsample block to increase spatial resolution. Notably, all residual blocks incorporate the timestep embedding. Self-attention blocks are included only at the two lowest spatial resolution levels of the encoder and decoder blocks due to computational constraints at higher resolutions.

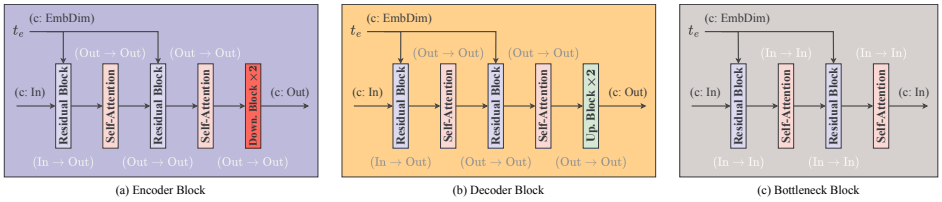


Figure 7: Core blocks of the U-Net-based network. (a) Encoder Block, (b) Decoder Block, and (c) Bottleneck Block.

The building blocks of the network are illustrated in Figure 8. Each residual block applies two convolutional layers with group normalization and Swish activation. They also contain a projection layer for the timestep embedding, composed of a Swish activation followed by a fully connected layer. Moreover, if the number of input channels (In) does not match the number of output channels (Out), an additional convolutional layer is included in the skip connection to project the input to the expected number of channels (Out), ensuring that the element-wise addition is well-defined. Self-attention blocks model long-range dependencies and incorporate group normalization both before and after the attention mechanism, operating over flattened spatial dimensions. Upsample and downsample blocks perform spatial resizing. Upsample blocks first perform bilinear interpolation (trilinear in case of 3D settings) to increase spatial resolution, followed by a convolutional layer, while downsample blocks perform convolution with stride greater than 1 (st > 1) to reduce spatial resolution. In the current implementation, activations are omitted, although the generalized block design can optionally include them.

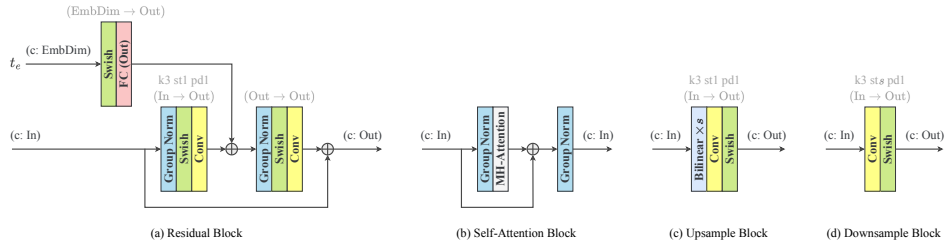


Figure 8: Building blocks. (a) Residual Block, (b) Self-Attention Block, (c) Upsample Block, and (d) Downsample Block.

C.3 Implementation Details

RDIM is implemented in PyTorch 2.5.1 (Paszke et al., 2019) and trained using the Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was initialized at 1.0×10^{-4} and decayed following a cosine annealing schedule with minimum value $\eta_{\min}$. All experiments were conducted with a batch size of 64 and an effective patch resolution of 64×64. For SR, this corresponds to LR patch sizes of 32×32 and 16×16 for $\times 2$ and $\times 4$ scale factors, respectively.

All diffusion models were trained with the same number of diffusion timesteps ($T = 50$ for FFHQ and $T = 100$ for experiments on FMD, SIDD, and DIV2K) and network architecture with 128 base channels (detailed in Appendix C.2). The only difference lies in the diffusion framework employed. ResShift is a specific case of RDIM, thus a single network was trained for both. For DDPM, following SR3 (Saharia et al., 2022), the model learns to approximate a reverse process, starting from pure Gaussian noise and iteratively denoising $\boldsymbol{x_t}$ toward the HQ image, $\boldsymbol{x_0}$, by predicting noise at each step, while conditioned on the LQ input, $\boldsymbol{y_0}$. Training was conducted for 4,000,000 iterations on FMD-Confocal and DIV2K datasets, 2,947,000 iterations on SIDD, and 4,375,000 iterations on FFHQ. For SR tasks in RDIM and ResShift, the LR input, $\boldsymbol{y_0}$, is upsampled to the target HR resolution using bilinear interpolation, ensuring compatibility with the resolution employed in the diffusion framework (i.e., the size of $\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_T}$).

All other techniques used in the comparative analysis of Section 3 strictly followed the reference papers and the official source codes. BM3D was applied with noise standard deviations of 10 for FMD-Confocal-BPAE-Raw, 30 for Confocal-Zebrafish-Raw, and 50 for SIDD-Medium. DnCNN was trained for 2,500,000 iterations on FMD-Confocal and SIDD datasets. ESRGAN was trained for a total of 1,400,000 iterations, with 1,000,000 iterations used to train a PSNR-oriented model that serves as initialization for the adversarial model, which was optimized for the remaining 400,000 iterations.

C.4 Uniform Sampling Timestep Schedule

At inference, RDIM intends to reconstruct the original data, $\boldsymbol{x_0}$, starting from the degraded final latent variable, $\boldsymbol{x_T}$. Unlike DDPMs and ResShift, where the sampling process requires iterating over all diffusion timesteps, T, the RDIM reverse process can be simulated with fewer timesteps. This results from the formulation of the RDIM reverse transition, which allows skipping intermediate timesteps during sampling (see Section 2.5). Accordingly, this flexibility motivates the selection of a subset, Υ, of $S < T$ sampling timesteps to traverse the reverse trajectory.

A simple yet effective approach is to adopt a linear sampling schedule, where the selected timesteps are uniformly spaced. Geometric schedules with denser allocation toward earlier or later stages of the reverse process were empirically evaluated, but they underperformed against a uniform alternative or yielded marginal improvements. As a result, the following uniform scheduler is devised:

$$\Upsilon = \left\{ \tau_k = \left\lfloor \frac{k}{S} \cdot T \right\rfloor \;\middle|\; k \in \{0, 1, \ldots, S\} \right\}, \tag{58}$$

where, during sampling, Υ is iterated from $\tau_S = T$ to τ_1, resulting in the order of sampling points $\tau_S \to \tau_{S-1} \to \cdots \to \tau_1$. Reverse transitions occur exclusively at these selected timesteps, from each τ_t to τ_{t-1}, with all intermediate timesteps being skipped. The exception is the target timestep $\tau_0 = 0$,

which marks the end of the reverse trajectory and does not produce a further transition. Moreover, all adjacent sampling timestep pairs, (τ_{k-1}, τ_k), satisfy the following condition:

$$(\tau_{k-1}, \tau_k) \in \left\{(t', t) \in \mathbb{N}_0^2 \mid t' + 1 \leq t \leq T\right\}. \tag{59}$$

In essence, only the latent variables associated with these timesteps are sampled, enabling a more efficient inference process. Figure 9 illustrates the sampling points (where reverse transitions occur) along the reverse trajectory, contextualized with the corresponding values of the β-schedule.

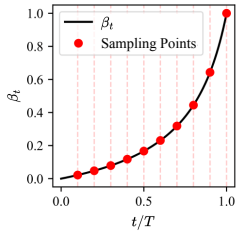


Figure 9: Residual β-schedule ($p = 5.0$) overlaid with red markers indicating the β_t value at each sampling point. The schedule adopted selects timesteps uniformly spaced. In this illustration, the forward process involves $T = 100$ timesteps and the number of sampling steps, where reverse transitions occur, is $S = 10$.

C.5 Impact of β-schedule parameter p

Experiments were conducted to determine an appropriate value for the β-schedule parameter p (see Section 2.6). In these experiments, RDIM employs $T = 10$ diffusion timesteps and a forward variance hyperparameter $\gamma = 9.0$. During inference, the reverse variance hyperparameter is set to $\eta = 1.0$ and multiple reverse trajectory lengths were evaluated. All other implementation details follow those described in Appendix C.3.

Table 2 presents the results for $p = 1$, $p = 5$, and $p = 15$. It follows that on denoising images from the FMD-Confocal-BPAE-Raw dataset, RDIM with $T = 10$ and $\gamma = 9.0$ achieves the best overall performance when $p = 5.0$. Moreover, irrespective of the steepness of the β-schedule, skipping timesteps and using fewer reverse timesteps ($S < T$) consistently yields superior results in terms of PSNR and SSIM compared to iterating through all diffusion steps ($S = T$). Particularly, Figure 2 in Section 2.6 illustrates the β-schedule curves and the effect on the diffusion process corresponding to these parameter values.

Table 2: Impact of parameter p, which controls the steepness of the curve in the β-schedule. All RDIM configurations were trained using a forward process with $T = 10$ timesteps and variance hyperparameter $\gamma = 9.0$. During inference, the reverse process variance hyperparameter is fixed to $\eta = 1.0$. Orange color rows highlight ResShift scenarios, corresponding to particular cases where RDIM reduces to ResShift under the conditions $\eta = 1.0$ and $S = T$.

p	S	FMD-Confocal-BPAE-Raw		
		PSNR↑	SSIM↑	LPIPS↓
1.0	1	40.0836	0.9678	0.0205
	5	40.0772	0.9678	0.0205
	10	40.0565	0.9677	0.0205
5.0	1	**40.1100**	**0.9681**	**0.0202**
	5	40.0436	0.9679	**0.0202**
	10	39.9524	0.9674	**0.0202**
15.0	1	39.4014	0.9639	0.0238
	5	39.1177	0.9624	0.0236
	10	38.8951	0.9609	0.0236

C.6 Impact of variance parameter γ

The diffusion process variance is controlled with a constant hyperparameter $\gamma \in [0, \infty)$, which allows interpolation between a deterministic ($\gamma = 0 \Rightarrow$ Gaussian collapses into a δ-distribution) and a stochastic ($\gamma > 0$) forward process. Figure 10 illustrates the impact of γ on the forward process.

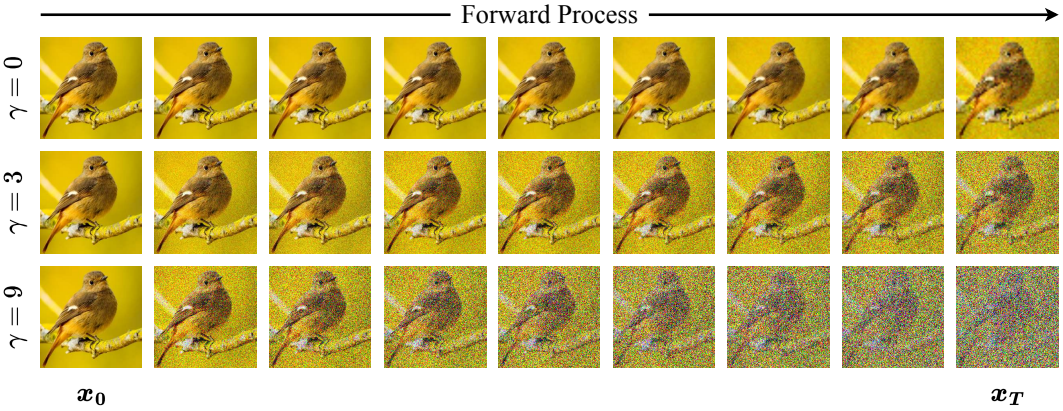


Figure 10: Impact of γ on the diffusion process with β-schedule parameter fixed to $p = 5.0$.

C.7 Comparative analysis of multiple RDIM configurations in image denoising

To identify the best RDIM configuration, several setups were compared on denoising of BPAE confocal images from the benchmark dataset FMD. The experiments explore the impact of the diffusion chain length (T), the number of sampling timesteps (S), and the variance controlled by the constant hyperparameters γ and η. Setups with $T = 100$ followed the implementation details described in Appendix C.3. For configurations with a different number of diffusion timesteps, the number of iterations (and consequently the training time) was adjusted linearly in proportion to the number of diffusion steps, T. This ensures that each timestep undergoes a similar number of weight updates across all configurations, thereby preventing imbalanced training between timesteps in configurations with different chain lengths. All other implementation details follow those described in Appendix C.3.

Table 3 summarizes the results. The RDIM configuration with $\gamma = 3.0$, $T = 100$, and $S = 10$ achieves the best performance in terms of PSNR and SSIM. Overall, the results suggest that increasing the number of diffusion timesteps improves denoising performance. Meanwhile, reducing the number of sampling timesteps ($S < T$) often yields better results. In contrast, learned perceptual image patch similarity (LPIPS) scores show that fewer sampling timesteps lead to worse perceptual quality. This highlights a trade-off between content fidelity (measured by PSNR and SSIM) and perceptual realism (measured by LPIPS). Iterative refinement enhances fine-grained details and promotes the recovery of natural textures. Logically, more sampling timesteps allow greater refinement, producing highly realistic outputs. However, results may diverge slightly from the ground truth in terms of pixel-wise similarity, resulting in lower PSNR and SSIM.

Moreover, results further indicate that controlled stochasticity in the forward process is beneficial. Setting $\gamma = 0.0$ leads to poor results, indicating that some variance is necessary. Conversely, $\gamma = 3.0$ and $\gamma = 9.0$ achieve significantly superior performance. Particularly, $\gamma = 9.0$ outperforms $\gamma = 3.0$ for configurations with few diffusion timesteps, but its relative performance gains diminish as T increases, whereas $\gamma = 3.0$ continues to improve with longer diffusion chains, ultimately surpassing $\gamma = 9.0$ for larger T. These observations underline the importance of carefully balancing variance.

Figure 11 showcases PSNR, SSIM, and LPIPS scores for different numbers of sampling timesteps, using the configuration with $T = 100$ and $\gamma = 3.0$, which obtained the best results in denoising BPAE confocal images from the FMD dataset. The number of sampling timesteps evaluated includes a single-step prediction and then ranges from 10 to 100 in increments of 10. It follows that PSNR and SSIM performances peak around $S = 10$, while LPIPS achieves the best scores between $S = 90$

Table 3: Denoising performance comparison of several RDIM configurations on BPAE confocal images from the benchmark dataset FMD. It exhibits the impact of the constant hyperparameter γ that controls the variance in the forward process and the impact of the number of diffusion timesteps during training (T) and inference (S). The constant hyperparameter that controls the variance in the reverse process is fixed to $\eta = 1.0$. Orange color rows highlight ResShift scenarios, corresponding to cases where RDIM reduces to ResShift ($\eta = 1.0$ and $S = T$).

T	S	FMD-Confocal-BPAE-Raw								
		$\gamma = 0.0$			$\gamma = 3.0$			$\gamma = 9.0$		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
10	1	38.3703	0.9575	0.0296	40.0998	0.9686	0.0196	40.1100	0.9681	0.0202
	10	38.3487	0.9572	0.0299	39.3632	0.9644	0.0187	39.9524	0.9674	0.0202
50	1	38.4181	0.9578	0.0293	42.5354	0.9803	0.0093	43.1161	0.9821	0.0079
	10	38.4162	0.9578	0.0295	42.4566	0.9802	0.0079	43.2029	0.9824	0.0077
	50	38.3198	0.9564	0.0299	42.0566	0.9785	0.0071	43.1970	0.9824	0.0074
100	1	38.3758	0.9575	0.0295	43.9872	0.9851	0.0056	43.3040	0.9828	0.0075
	10	38.3752	0.9575	0.0295	**44.1468**	**0.9855**	0.0047	43.3743	0.9830	0.0073
	100	38.1775	0.9548	0.0298	43.5990	0.9837	**0.0042**	43.2484	0.9826	0.0068

and $S = 100$, showing that more sampling timesteps result in higher perceptual quality but reduced reconstruction fidelity.

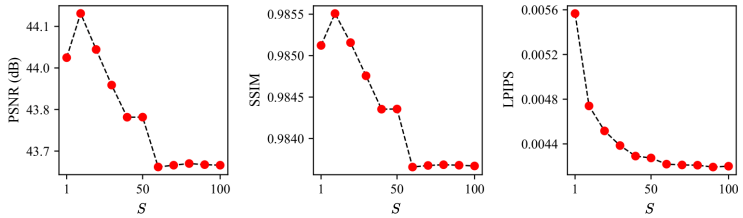


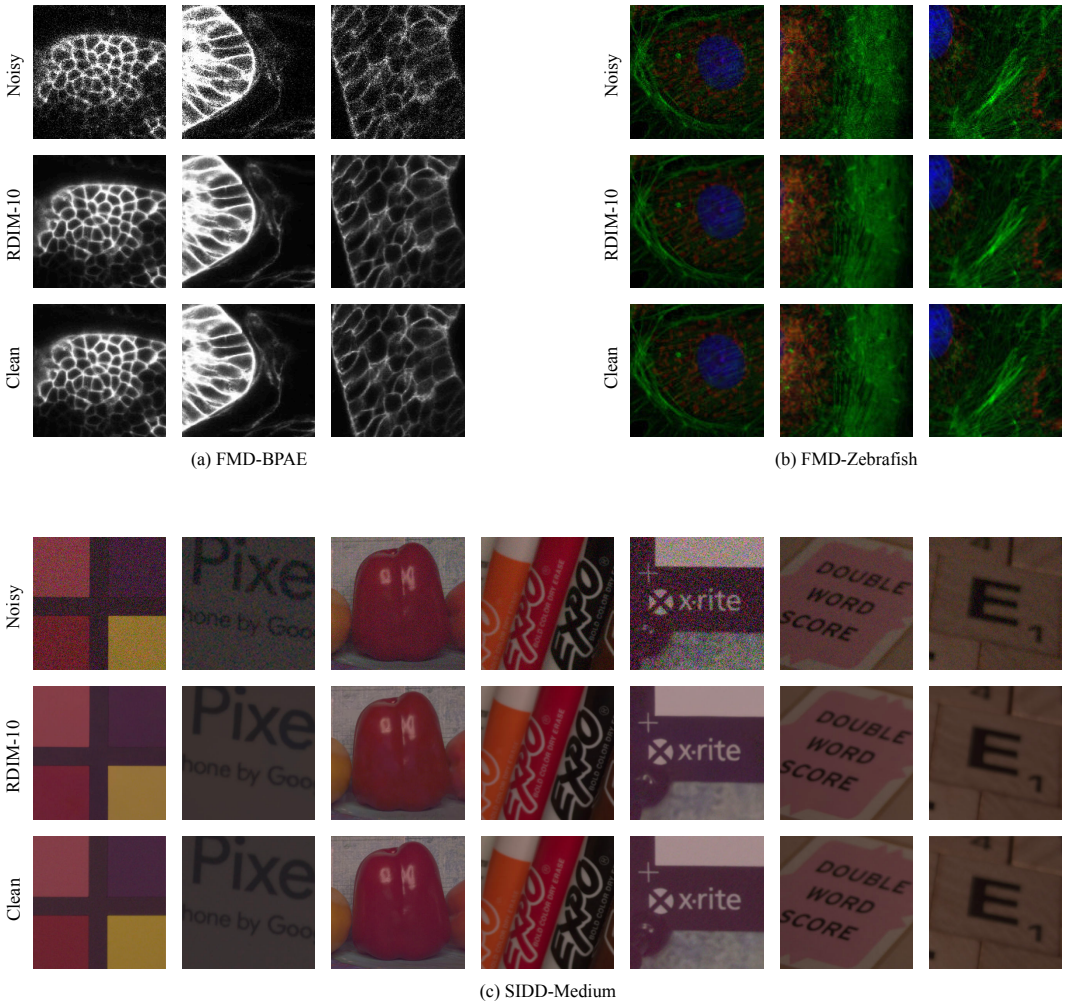
Figure 11: PSNR, SSIM, and LPIPS performance as a function of S (number of sampling timesteps) on denoising of BPAE confocal images from the FMD dataset. RDIM is trained with $T = 100$ and $\gamma = 3.0$. The constant hyperparameter that controls the variance in the reverse process is fixed to $\eta = 1.0$. The number of sampling timesteps evaluated includes a single-step prediction and then ranges from 10 to 100 in increments of 10.

Table 4 demonstrates the effect of varying the constant hyperparameter η, which controls the variance in the reverse process. Looking at Table 4, the parameter η manifests marginal impact on denoising of BPAE confocal images from the FMD dataset. Additionally, when $\gamma = 0$, the parameter η does not affect performance, as η is absent in the reparameterized form of $p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}, \boldsymbol{y_0})_{\gamma=0}$ (see Equations (11) and (12)).

Table 4: Impact of the constant hyperparameter η, which controls the variance in the reverse process, on denoising BPAE confocal images from the benchmark dataset FMD.

T	S	η	FMD-Confocal-BPAE-Raw								
			$\gamma = 0.0$			$\gamma = 3.0$			$\gamma = 9.0$		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
100	10	0.0	38.3752	0.9575	0.0295	44.1429	0.9855	**0.0047**	43.3732	0.9830	0.0073
		0.5	38.3752	0.9575	0.0295	44.1440	0.9855	**0.0047**	43.3738	0.9830	0.0073
		1.0	38.3752	0.9575	0.0295	**44.1468**	**0.9855**	**0.0047**	43.3743	0.9830	0.0073

C.8 Additional Qualitative Results

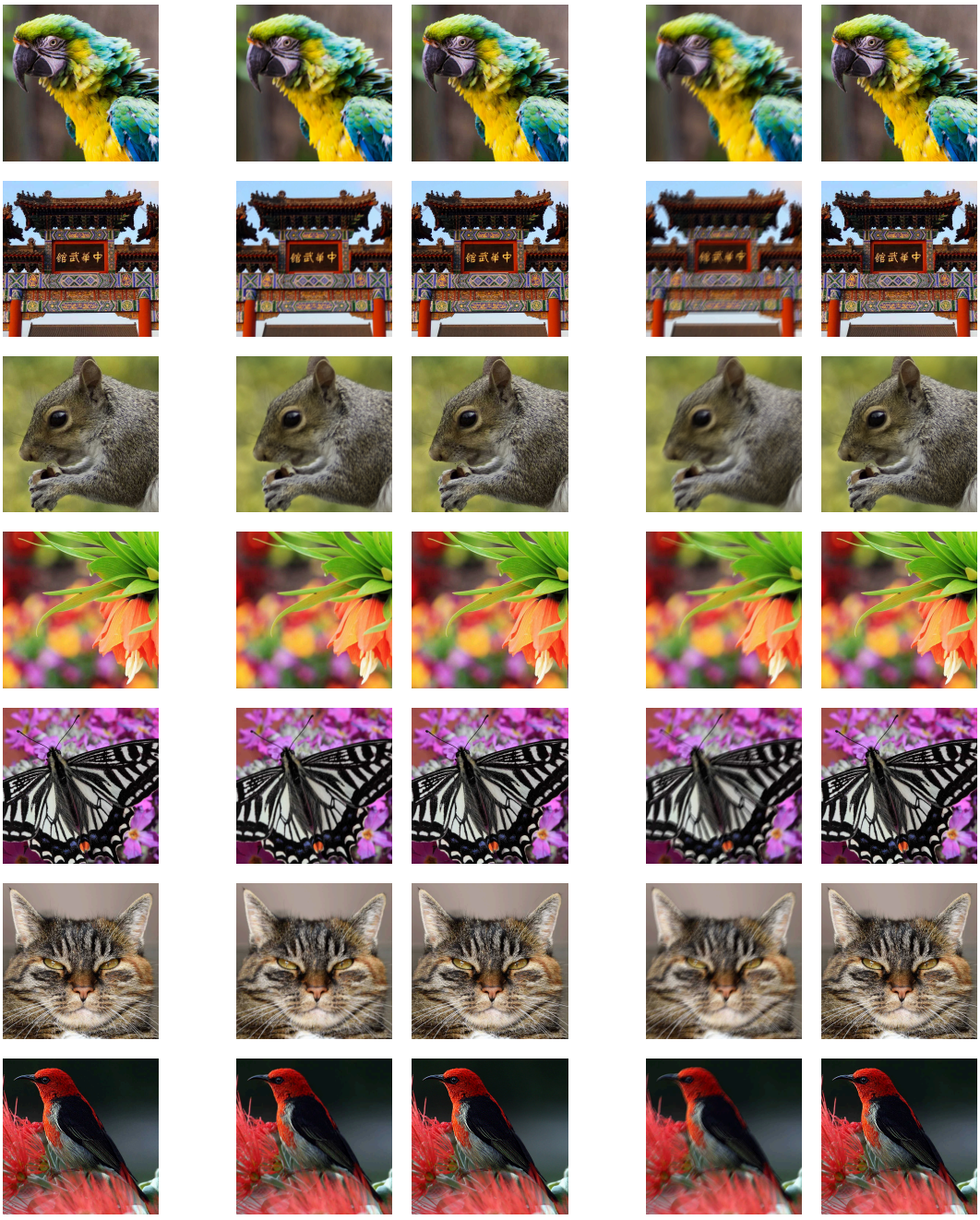


(a) FMD-BPAE

(b) FMD-Zebrafish

(c) SIDD-Medium

Figure 12: Denoising results of RDIM-10 on images from the FMD and SIDD datasets. For improved visualization, only cropped regions are shown. RDIM is trained with $T = 100$ and $\gamma = 3.0$. Inference is conducted with $S = 10$ and $\eta = 1.0$.



(a) Original (b) $\times 2$ SR (c) $\times 4$ SR

Figure 13: $\times 2$ and $\times 4$ SR results of RDIM-10 on images from the DIV2K dataset. RDIM is trained with $T = 100$ and $\gamma = 3.0$. Inference is conducted with $S = 10$ and $\eta = 1.0$. In (b) and (c), the left side represents the input image and the right side the output.

(a) Original (b) Inpainting (c) Colorization (d) Deblurring

Figure 14: Image inpainting, colorization, and deblurring results of RDIM-10 on images from the FFHQ dataset. For inpainting, pixels in the original images are randomly masked and set to zero with probability $p_{\text{mask}} = 0.5$. For colorization, grayscale inputs are obtained by converting the original RGB images to luminance. For deblurring, synthetic blurred images are generated from ground truth images by applying a Gaussian blur with kernel size 15×15 and standard deviation $\sigma = 3.0$. RDIM is trained with $T = 50$ and $\gamma = 3.0$. Inference is conducted with $S = 10$ and $\eta = 1.0$. In (b), (c) and (d), the left side represents the input image and the right side the output.