Diffusion Rejection Sampling

Byeonghu Na [1] **Yeongmin Kim** [1] **Minsang Park** [1] **Donghyeok Shin** [1] **Wanmo Kang** [1] **Il-Chul Moon** [1 2]

Abstract

Recent advances in powerful pre-trained diffusion models encourage the development of methods to improve the sampling performance under well-trained diffusion models. This paper introduces Diffusion Rejection Sampling (DiffRS), which uses a rejection sampling scheme that aligns the sampling transition kernels with the true ones at each timestep. The proposed method can be viewed as a mechanism that evaluates the quality of samples at each intermediate timestep and refines them with varying effort depending on the sample. Theoretical analysis shows that DiffRS can achieve a tighter bound on sampling error compared to pre-trained models. Empirical results demonstrate the state-of-the-art performance of DiffRS on the benchmark datasets and the effectiveness of DiffRS for fast diffusion samplers and large-scale text-to-image diffusion models. Our code is available at https://github.com/aailabkaist/DiffRS.

1. Introduction

Diffusion models have attracted considerable interest in various domains, such as image (Dhariwal & Nichol, 2021; Rombach et al., 2022) and video generation (Ho et al., 2022b; Voleti et al., 2022), due to their remarkable ability to generate high-quality samples. The powerful generative capabilities of diffusion models have spurred extensive efforts to further improve the sampling quality. A common strategy is to reduce the sampling interval, thereby increasing the iterative sampling count (Karras et al., 2022). However, this comes at the cost of a higher number of network evaluations, which slows down the sampling speed. An alternative approach is to improve the training of the reverse diffusion process to accurately model the reverse transition (Kim et al., 2022b; Rombach et al., 2022; Lai et al., 2023; Zheng et al., 2023a). Nonetheless, these methods require time-consuming training of the diffusion model.

In contrast to these approaches, recent advances in powerful pre-trained models (Rombach et al., 2022; Karras et al., 2022) have led to a growing body of research focused on leveraging them (Kim et al., 2023; Xu et al., 2023a; Ning et al., 2024). In line with these efforts, our goal is to effectively and efficiently leverage a well-trained diffusion model to improve the sampling quality. We introduce a mechanism that assesses the quality of a sample at each intermediate timestep, allowing us to keep good samples as well as to refine poor samples by injecting appropriate noise and by going back to prior timesteps.

Specifically, we propose Diffusion Rejection Sampling (DiffRS), which is based on the ratio of the true transition kernel to the transition kernel of the pre-trained model for each timestep, see Figure 1. The ratio can be estimated by a time-dependent discriminator that distinguishes between data and generated samples at each timestep. In cases where samples are rejected, we adjust the noise intensity depending on the rejected samples. We theoretically prove that discriminator training leads to a tighter upper bound on the sampling error of DiffRS compared to a pre-trained diffusion model. In the experiments, DiffRS achieves new state-of-the-art (SOTA) performance on CIFAR-10, and near-SOTA performance on ImageNet 64×64 with fewer NFEs. Moreover, we demonstrate the effective application of DiffRS to the fast diffusion samplers, such as DPM-Solver++ (Lu et al., 2022b) and Consistency Model (Song et al., 2023), and large-scale text-to-image generation models, including Stable Diffusion (Rombach et al., 2022).

2. Preliminary

Diffusion Model Diffusion-based generative models (Ho et al., 2020; Song et al., 2021b; Dhariwal & Nichol, 2021) are one of the most prominent deep generative models that aim to approximate the data distribution to the model distribution. This model includes a forward diffusion process that iteratively perturbs the data instances toward the prior distribution, and a corresponding reverse process that inverts the forward process to sample from the modeled distribution.

[1]Department of Industrial & Systems Engineering, KAIST, Daejeon, Republic of Korea [2]summary.ai, Daejeon, Republic of Korea. Correspondence to: Il-Chul Moon <icmoon@kaist.ac.kr>, Byeonghu Na <byeonghu.na@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

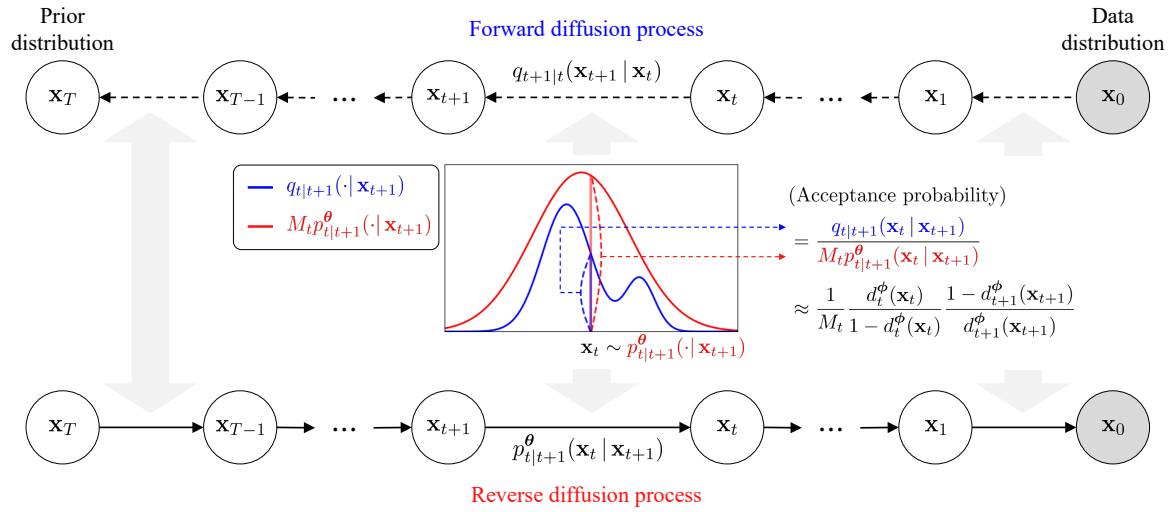


Figure 1. Overview of DiffRS. We sequentially apply the rejection sampling on the pre-trained transition kernel $p^{\boldsymbol{\theta}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})$ (red) to align the true transition kernel $q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})$ (blue). The acceptance probability is estimated by the time-dependent discriminator $d^{\boldsymbol{\phi}}_t$.

The forward process is formulated by a fixed Markov chain that constructs a set of latent variables $\mathbf{x}_{1:T}$ by adding Gaussian noises from data distribution $q_0(\mathbf{x}_0)$ (Ho et al., 2020):

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q_{t|t-1}(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{1}$$

where $q_{t|t-1}(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$ and β_t is a variance schedule parameter at time t. Most diffusion models define the reverse process by a Markov chain with a Gaussian transition kernel $p_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})$:

$$p(\mathbf{x}_{0:T}) := p_T(\mathbf{x}_T) \prod_{t=0}^{T-1} p_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}), \tag{2}$$

where $p_T(\mathbf{x}_T)$ is the prior distribution. Then, the goal of the diffusion model is to approximate the transition kernel $p_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})$ by a Gaussian with parameterized mean $\boldsymbol{\mu}^{\boldsymbol{\theta}}$ and time-dependent variance σ^2_{t+1},

$$p^{\boldsymbol{\theta}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}) := \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}^{\boldsymbol{\theta}}(\mathbf{x}_{t+1}, t+1), \sigma^2_{t+1}\mathbf{I}), \tag{3}$$

using the objective of variational bound on the log likelihood. When the parameterized transition kernel $p^{\boldsymbol{\theta}}_{t|t+1}$ is obtained, we proceed with iterative sampling from T to 0 using Eq. (2), replacing a transition kernel with $p^{\boldsymbol{\theta}}_{t|t+1}$:

$$\mathbf{x}_t = \boldsymbol{\mu}^{\boldsymbol{\theta}}(\mathbf{x}_{t+1}, t+1) + \sigma^2_{t+1}\mathbf{z} \text{ where } \mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}). \tag{4}$$

Refining Sampling Process from Pre-trained Models While most previous methods require training of diffusion models to reduce the sampling error, some recent work has explored refining the sampling process from pre-trained diffusion models. DG (Kim et al., 2023) corrects the transition kernel by adding an auxiliary term from the discriminator $d^{\boldsymbol{\phi}}_t$ that distinguishes between real and generated samples:

$$\boldsymbol{\mu}^{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_t, t) := \boldsymbol{\mu}^{\boldsymbol{\theta}}(\mathbf{x}_t, t) + \alpha_t \nabla_{\mathbf{x}_t} \log \frac{d^{\boldsymbol{\phi}}_t(\mathbf{x}_t)}{1-d^{\boldsymbol{\phi}}_t(\mathbf{x}_t)}, \tag{5}$$

where α_t is a time-dependent constant. After that, sampling proceeds with the adjusted transition kernel $\boldsymbol{\mu}^{\boldsymbol{\theta},\boldsymbol{\phi}}$ to reduce the network estimation error. We also use a fixed pre-trained diffusion and utilize the discriminator, but our distinctive method is the application of a rejection sampling scheme.

In addition, Restart (Xu et al., 2023a) introduces a strategy of repeating the backward and forward steps at fixed time interval $[t_{\text{min}}, t_{\text{max}}]$. Specifically, Restart iteratively samples with a deterministic sampler, such as an ODE sampler, from T to t_{min}. Then, it imposes stochasticity by adding large noise and simulates a reverse process from t_{max} to t_{min}:

$$\text{(Restart forward) } \mathbf{x}^{i+1}_{t_{\text{max}}} = \mathbf{x}^{i}_{t_{\text{min}}} + \epsilon_{t_{\text{min}} \to t_{\text{max}}}, \tag{6}$$

$$\text{(Restart reverse) } \mathbf{x}^{i+1}_{t_{\text{min}}} = \text{ODE}_{\boldsymbol{\theta}}(\mathbf{x}^{i+1}_{t_{\text{max}}}, t_{\text{max}} \to t_{\text{min}}), \tag{7}$$

where $\epsilon_{t_{\text{min}} \to t_{\text{max}}}$ denotes the injected noise of the forward process from t_{min} to t_{max} and $\text{ODE}_{\boldsymbol{\theta}}$ represents the reverse process using a deterministic sampler from t_{max} to t_{min}. These processes are repeated, demonstrating an increased contraction effect on accumulated errors. Our rejection sampling differs in that the timesteps for applying the forward process are determined probabilistically for each sample.

Rejection Sampling Rejection sampling is a numerical sampling method to be used when a target distribution $q(\mathbf{x})$ can be evaluated whereas its direct sampling is difficult (Ripley, 2009). For this, we need a proposal distribution $p(\mathbf{x})$ that can be evaluated and from which we can draw samples. We also need to find a constant M satisfying $q(\mathbf{x}) \leq Mp(\mathbf{x})$ for all $\mathbf{x}$. Then, we accept a sample $\mathbf{x}$ drawn from $p(\mathbf{x})$ with probability of $q(\mathbf{x})/Mp(\mathbf{x})$, and otherwise reject it.

Some work on generative models takes advantage of this rejection sampling scheme. Grover et al. (2018) use it to

Algorithm 1 `OneStepDiffRS` $(t, \mathbf{x}_{t+1}, L_{t+1})$

Input: $p^{\boldsymbol{\theta}}_{t|t+1}$, $q_t/p^{\boldsymbol{\theta}}_t$ (or $d^{\boldsymbol{\phi}}_t/[1-d^{\boldsymbol{\phi}}_t]$), M_t
Output: $\mathbf{x}_t$, L_t
1: $\mathbf{x}_t \leftarrow$ None
2: **while** $\mathbf{x}_t$ is None **do**
3: Sample $\tilde{\mathbf{x}}_t$ from the transition kernel $p^{\boldsymbol{\theta}}_{t|t+1}(\cdot|\mathbf{x}_{t+1})$
4: Compute $L_t \leftarrow \frac{q_t(\tilde{\mathbf{x}}_t)}{p^{\boldsymbol{\theta}}_t(\tilde{\mathbf{x}}_t)}$ and $A_t \leftarrow \frac{L_t}{M_t L_{t+1}}$
5: Sample $u \sim \text{Uniform}(0,1)$
6: **if** $u < A_t$ **then**
7: $\mathbf{x}_t \leftarrow \tilde{\mathbf{x}}_t$
8: **else**
9: $\mathbf{x}_{t+1}, L_{t+1} \leftarrow$ `Re-initialization`$(t+1, \tilde{\mathbf{x}}_t)$
10: **end if**
11: **end while**

improve samples drawn from the variational posterior of the variational autoencoder. Azadi et al. (2019); Turner et al. (2019) generate data instances from the generative adversarial network by evaluating the acceptance probability using the discriminator. Compared to previous studies, the sampling of diffusion models is iterative, which requires a sequential rejection sampling method over diffusion timesteps.

3. Methods

3.1. Diffusion Rejection Sampling (DiffRS)

We assume the existence of a pre-trained diffusion model that allows the generation of samples using the transition kernel $p^{\boldsymbol{\theta}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})$. The distribution of a sample $\mathbf{x}_0$ obtained through a sequence of transition samples $\mathbf{x}_t|\mathbf{x}_{t+1}$, denoted $p^{\boldsymbol{\theta}}_0(\mathbf{x}_0)$, may deviate from the true data distribution $q_0(\mathbf{x}_0)$ if the pre-trained transition kernel $p^{\boldsymbol{\theta}}_{t|t+1}$ differs from the true transition kernel $q_{t|t+1}$. Consequently, we apply a rejection sampling scheme for each timestep in the transition kernel to mitigate this discrepancy, as described in Figure 1.

Conceptually, DiffRS performs the rejection sampling of the transition probability in reverse diffusion, $p^{\boldsymbol{\theta}}_{t|t+1}$.[1] During the generation procedure, the sampling means selecting an instance from $p^{\boldsymbol{\theta}}_{t|t+1}$, which follows a Gaussian of Eq. (3), so it can perform as a proposal distribution of the rejection sampling. Meanwhile, the ordinary forward diffusion, $q_{t+1|t}$, follows a Gaussian distribution; but its reverse-time version, $q_{t|t+1}$, does not follow a Gaussian distribution, which becomes the target distribution of the rejection sampling.

To formulate DiffRS, let $q_t(\mathbf{x}_t)$ and $p^{\boldsymbol{\theta}}_t(\mathbf{x}_t)$ represent the marginal distributions of the forward diffusion process starting from $q_0(\mathbf{x}_0)$ and $p^{\boldsymbol{\theta}}_0(\mathbf{x}_0)$, respectively. We introduce a one-step DiffRS procedure from $t+1$ to t to obtain a sample $\mathbf{x}_t$ from q_t, given a sample $\mathbf{x}_{t+1}$ from q_{t+1}. This procedure can be applied sequentially from $T-1$ to 0, yielding a sample from the data distribution q_0.

Proposal Distribution At time $t+1$, we assume that we have a sample $\mathbf{x}_{t+1}$ drawn from the perturbed data distribution $q_{t+1}(\mathbf{x}_{t+1})$ through the sampling iterations from T to $t+1$. Then, a sample $\mathbf{x}_t$ at time t can be drawn using the pre-trained transition kernel $p^{\boldsymbol{\theta}}_{t|t+1}$ following the generative reverse process by Eq. (4). Our goal is to ensure that the sampling closely follows the true transition kernel $q_{t|t+1}$ (blue in Figure 1). This is achieved by applying the rejection sampling, where the proposal distribution is set by the pre-trained transition kernel $p^{\boldsymbol{\theta}}_{t|t+1}$ (red in Figure 1).

Acceptance Probability To implement the rejection sampling scheme on the transition kernel, we need to compute the acceptance probability $A_t(\mathbf{x}_t, \mathbf{x}_{t+1})$, which is expressed as the ratio of the true and pre-trained transition kernel:

$$A_t(\mathbf{x}_t, \mathbf{x}_{t+1}) := \frac{1}{M_t} \frac{q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})}{p^{\boldsymbol{\theta}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})}, \tag{8}$$

where M_t is a constant that satisfies $q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1}) \leq M_t p^{\boldsymbol{\theta}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})$ for all $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$. The density ratio can be further derived as follows:

$$\begin{aligned}\frac{q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})}{p^{\boldsymbol{\theta}}_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})} &= \frac{q_{t+1|t}(\mathbf{x}_{t+1}|\mathbf{x}_t)}{p_{t+1|t}(\mathbf{x}_{t+1}|\mathbf{x}_t)} \frac{q_t(\mathbf{x}_t)}{p^{\boldsymbol{\theta}}_t(\mathbf{x}_t)} \frac{p^{\boldsymbol{\theta}}_{t+1}(\mathbf{x}_{t+1})}{q_{t+1}(\mathbf{x}_{t+1})} \\ &= \frac{q_t(\mathbf{x}_t)}{p^{\boldsymbol{\theta}}_t(\mathbf{x}_t)} \frac{p^{\boldsymbol{\theta}}_{t+1}(\mathbf{x}_{t+1})}{q_{t+1}(\mathbf{x}_{t+1})} = \frac{L_t(\mathbf{x}_t)}{L_{t+1}(\mathbf{x}_{t+1})}, \end{aligned} \tag{9}$$

where $L_t(\mathbf{x}_t) := \frac{q_t(\mathbf{x}_t)}{p^{\boldsymbol{\theta}}_t(\mathbf{x}_t)}$. The first equality holds by Bayes' rule, and we use the fact that the perturbed kernels, $q_{t+1|t}$ and $p_{t+1|t}$, are the same for the second equality. Therefore, the acceptance probability $A_t(\mathbf{x}_t, \mathbf{x}_{t+1})$ of the one-step DiffRS at time t can be expressed as follows:

$$A_t(\mathbf{x}_t, \mathbf{x}_{t+1}) = \frac{L_t(\mathbf{x}_t)}{M_t L_{t+1}(\mathbf{x}_{t+1})}. \tag{10}$$

$L_t(\mathbf{x}_t)$ is estimated by the density ratio estimation via a discriminator $d^{\boldsymbol{\phi}}_t$, which will be discussed in Section 3.2.

Algorithm of One-step DiffRS We formulate a one-step DiffRS procedure in Algorithm 1. Note that Re-initialization (line 9 in Algorithm 1) refers to the process of drawing a new sample at timestep $t+1$ after a rejection, which we will explain further in Section 3.3.

3.2. Estimation of the Acceptance Probability

As indicated in Eq. (10), the acceptance probability is expressed as the ratio of the likelihood ratios at time t and $t+1$. Therefore, if we can estimate the likelihood ratio $L_t(\mathbf{x}_t)$ at each timestep, we can compute the acceptance probability. Following the approach of DG (Kim et al., 2023), we

[1] It should be noted that the rejection sampling is imposed on the transition probability, $p^{\boldsymbol{\theta}}_{t|t+1}$; not its marginal probability, $p^{\boldsymbol{\theta}}_t$.

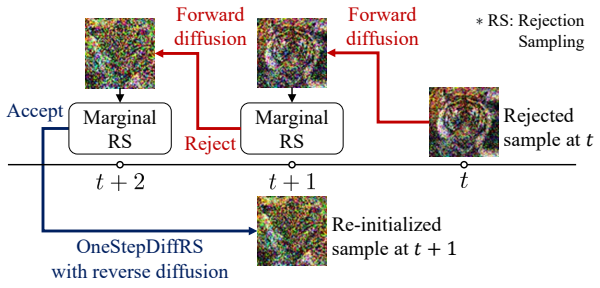


Figure 2. Overview of the proposed re-initialization.

estimate this ratio using a time-dependent discriminator, denoted by $d_t^{\boldsymbol{\phi}}$. This discriminator is designed to distinguish between samples of q_t and $p_t^{\boldsymbol{\theta}}$ at all timesteps.

To train the discriminator, we generate the samples of $p_0^{\boldsymbol{\theta}}$ using Eq. (4) with the pre-trained diffusion model. The training objective is the time-weighted binary cross-entropy loss using the real and generated samples:

$$\mathcal{L}_{\text{BCE}}(\boldsymbol{\phi}) := \mathbb{E}_t \Big[\lambda(t) \mathbb{E}_{q_0(\mathbf{x}_0) q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)} \big[-\log d_t^{\boldsymbol{\phi}}(\mathbf{x}_t) \big] + \mathbb{E}_{p_0^{\boldsymbol{\theta}}(\mathbf{x}_0) q_{t|0}(\mathbf{x}_t|\mathbf{x}_0)} \big[-\log(1 - d_t^{\boldsymbol{\phi}}(\mathbf{x}_t)) \big] \Big], \quad (11)$$

where $\lambda(t)$ is the temporal weighting function. Then, the optimal discriminator $d_t^{\boldsymbol{\phi}^*}$ satisfies the following equations:

$$d_t^{\boldsymbol{\phi}^*}(\mathbf{x}_t) = \frac{q_t(\mathbf{x}_t)}{q_t(\mathbf{x}_t) + p_t^{\boldsymbol{\theta}}(\mathbf{x}_t)}; L_t(\mathbf{x}_t) = \frac{q_t(\mathbf{x}_t)}{p_t^{\boldsymbol{\theta}}(\mathbf{x}_t)} = \frac{d_t^{\boldsymbol{\phi}^*}(\mathbf{x}_t)}{1 - d_t^{\boldsymbol{\phi}^*}(\mathbf{x}_t)}. \quad (12)$$

Therefore, using the time-dependent discriminator $d_t^{\boldsymbol{\phi}}$, we derive the estimators $\hat{L}_t^{\boldsymbol{\phi}}$ and $\hat{A}_t^{\boldsymbol{\phi}}$ for the ratio L_t and the acceptance probability A_t, respectively:

$$L_t(\mathbf{x}_t) \approx \hat{L}_t^{\boldsymbol{\phi}}(\mathbf{x}_t) := \frac{d_t^{\boldsymbol{\phi}}(\mathbf{x}_t)}{1 - d_t^{\boldsymbol{\phi}}(\mathbf{x}_t)}, \quad (13)$$

$$A_t(\mathbf{x}_t, \mathbf{x}_{t+1}) \approx \hat{A}_t^{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{x}_{t+1}) := \frac{1}{M_t} \frac{\hat{L}_t^{\boldsymbol{\phi}}(\mathbf{x}_t)}{\hat{L}_{t+1}^{\boldsymbol{\phi}}(\mathbf{x}_{t+1})}. \quad (14)$$

3.3. Re-initialization

The primary challenge associated with the rejection sampling is the increased number of sampling iterations caused by rejections. This problem is particularly exacerbated in diffusion models that use iterative sampling, since rejections require resampling starting from the timestep T. To mitigate this challenge, we introduce a re-initialization method tailored for diffusion models, utilizing rejected samples.

Motivated by the observation from Restart (Xu et al., 2023a) that incorporating the forward process into the sampling process reduces the accumulated error, we add noise to the rejected samples $\mathbf{x}_t$. Unlike Restart, we inject different amounts of noise for each sample based on the likelihood ratio information we already have, as illustrated in Figure 2.

Specifically, we first apply a one-step forward transition $q_{t+1|t}$ to the rejected sample $\mathbf{x}_t$ at time t to obtain the candidate sample $\mathbf{x}_{t+1}$ at time $t+1$. Then, we apply an additional rejection sampling procedure to the candidate sample $\mathbf{x}_{t+1}$ based on the marginal distributions q_{t+1} and $p_{t+1}^{\boldsymbol{\theta}}$. If the sample is rejected again, we iterate through the one-step forward transition and the marginal rejection sampling. Consequently, the intensity of the noise is adjusted based on the probability that a rejected sample is drawn from the true distribution. We present this re-initialization procedure in Algorithm 2. Empirically, we find that this re-initialization procedure leads to effective and efficient sample generation.

Algorithm 2 `Re-initialization`$(t+1, \tilde{\mathbf{x}}_t)$

Input: $q_{t+1|t}$, $q_{t+1}/p_{t+1}^{\boldsymbol{\theta}}$ (or $d_{t+1}^{\boldsymbol{\phi}}/[1 - d_{t+1}^{\boldsymbol{\phi}}]$), $\tilde{M}_{t+1}$
Output: $\mathbf{x}_{t+1}$, L_{t+1}

Sample $\tilde{\mathbf{x}}_{t+1}$ from the forward process $q_{t+1|t}(\cdot|\mathbf{x}_t)$
Compute $L_{t+1} \leftarrow \frac{q_{t+1}(\tilde{\mathbf{x}}_{t+1})}{p_{t+1}^{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t+1})}$ and $\tilde{A}_{t+1} \leftarrow \frac{L_{t+1}}{\tilde{M}_{t+1}}$
Sample $u \sim \text{Uniform}(0, 1)$
if $(u < \tilde{A}_{t+1})$ or $(t+1 == T)$ **then**
  $\mathbf{x}_{t+1} \leftarrow \tilde{\mathbf{x}}_{t+1}$
else
  $\mathbf{x}_{t+2}, L_{t+2} \leftarrow$ `Re-initialization`$(t+2, \tilde{\mathbf{x}}_{t+1})$
  $\mathbf{x}_{t+1}, L_{t+1} \leftarrow$ `OneStepDiffRS`$(t+1, \mathbf{x}_{t+2}, L_{t+2})$
end if

Algorithm 3 Diffusion Rejection Sampling (DiffRS)

$\mathbf{x}_T \leftarrow$ None
while $\mathbf{x}_T$ is None **do**
  Sample $\tilde{\mathbf{x}}_T$ from the prior distribution $p_T(\mathbf{x}_T)$
  Compute $L_T \leftarrow \frac{q_T(\mathbf{x}_T)}{p_T(\mathbf{x}_T)}$ and $\tilde{A}_T \leftarrow \frac{L_T}{\tilde{M}_T}$
  Sample $u \sim \text{Uniform}(0, 1)$
  if $u < \tilde{A}_T$ **then**
    $\mathbf{x}_T \leftarrow \tilde{\mathbf{x}}_T$
  end if
end while
for $t = T - 1$ **to** 0 **do**
  $\mathbf{x}_t, L_t \leftarrow$ `OneStepDiffRS`$(t, \mathbf{x}_{t+1}, L_{t+1})$
end for

3.4. Overall Algorithm

Algorithm 3 presents the overall algorithm of DiffRS. First, we sample $\mathbf{x}_T$ from the prior distribution p_T and then perform the marginal rejection sampling with the acceptance probability $A_T(\mathbf{x}_T) = \frac{q_T(\mathbf{x}_T)}{M_T p_T(\mathbf{x}_T)}$ (lines 1-9). This process aims to bring the prior distribution closer to q_T, thereby

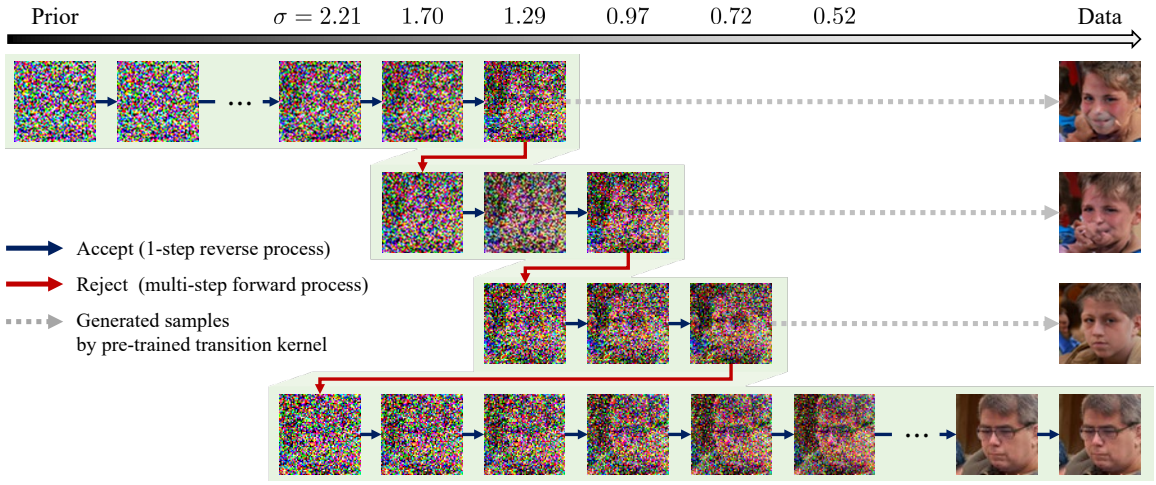


Figure 3. Illustration of the sampling process for DiffRS. The path with the green background represents the DiffRS sampling process, and the rightmost images are generated from the intermediate images using a base sampler without rejection. Timesteps are expressed as the noise level σ from the EDM scheme (Karras et al., 2022).

reducing the prior mismatch error. Subsequently, we iteratively apply the one-step DiffRS from $T-1$ to 0 (lines 10-12), ultimately obtaining a sample $\mathbf{x}_0$ on the data space.

Figure 3 visually illustrates the DiffRS process, highlighted with a green background. The rightmost images show the generated samples when continuing to sample from the intermediate images without rejection. The sample is refined by finding new sampling paths through rejection.

It is important to note that DiffRS can enhance sample quality for most diffusion samplers. A necessary condition is that the sampler aims to sample from the true perturbed data distribution $q_t(\mathbf{x}_t)$ at time t. This condition holds true for most samplers, including diffusion distillation methods.

Practical Consideration The implementation of DiffRS requires the determination of the rejection constant M_t. It should be noted that M_t exists for all t, since the diffusion process is based on a Gaussian distribution, so the support of the transition kernels becomes the entire space. However, finding an exact value for M_t is nearly impossible, and even if it were possible, it would be computationally intractable in practice. In accordance with previous research (Azadi et al., 2019), we determine M_t as follows: we store the ratio $\frac{\hat{L}_t^{\boldsymbol{\phi}}(\mathbf{x}_t)}{\hat{L}_{t+1}^{\boldsymbol{\phi}}(\mathbf{x}_{t+1})}$ of samples from the base sampler and select the γ^{th} percentile of these stored values as M_t. We apply this method similarly to the marginal rejection sampling.

3.5. Theoretical Analysis

We provide a theoretical analysis of the DiffRS algorithm based on distribution divergence. Ho et al. (2020) derived the upper bound of the Kullback-Leibler (KL) divergence between the data distribution $q_0(\mathbf{x}_0)$ and the pre-trained distribution $p_0^{\boldsymbol{\theta}}(\mathbf{x}_0)$ in diffusion models:

$$D_{\text{KL}}(q_0||p_0^{\boldsymbol{\theta}}) \leq D_{\text{KL}}(q_T||p_T^{\boldsymbol{\theta}}) + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t+1}}\left[D_{\text{KL}}(q_{t|t+1}||p_{t|t+1}^{\boldsymbol{\theta}})\right] =: J(\boldsymbol{\theta}). \tag{15}$$

Therefore, to minimize the KL divergence on the data space, we need to match prior distributions, q_T and $p_T^{\boldsymbol{\theta}}$; and transition kernels, $q_{t|t+1}$ and $p_{t|t+1}^{\boldsymbol{\theta}}$; which is the purpose of DiffRS.

For further theoretical analysis, let $p_*^{\boldsymbol{\theta},\boldsymbol{\phi}}$ be the distribution refined by DiffRS. We also define the unnormalized acceptance probability $\bar{A}_t^{\boldsymbol{\phi}} := M_t \hat{A}_t^{\boldsymbol{\phi}} = \frac{\hat{L}_t^{\boldsymbol{\phi}}(\mathbf{x}_t)}{\hat{L}_{t+1}^{\boldsymbol{\phi}}(\mathbf{x}_{t+1})}$. Then, the refined prior distribution and the refined transition kernels of DiffRS are expressed by the pre-trained distribution and the acceptance probability:

$$p_T^{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_T) = p_T^{\boldsymbol{\theta}}(\mathbf{x}_T)\bar{A}_T^{\boldsymbol{\phi}}(\mathbf{x}_T), \tag{16}$$

$$p_{t|t+1}^{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_t|\mathbf{x}_{t+1}) = p_{t|t+1}^{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t+1})\bar{A}_t^{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{x}_{t+1}). \tag{17}$$

Theorem 3.1 formulates the upper bound of the KL divergence between the data and refined distribution.

Theorem 3.1. *The KL divergence between data distribution q_0 and refined distribution $p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}$ is bounded by:*

$$D_{KL}(q_0||p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}) \leq J(\boldsymbol{\theta}) + R(\boldsymbol{\phi}) =: J(\boldsymbol{\theta}, \boldsymbol{\phi}), \tag{18}$$

where $R(\boldsymbol{\phi}) := \mathbb{E}_{q_T}[-\log \bar{A}_T^{\boldsymbol{\phi}}] + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t,t+1}}[-\log \bar{A}_t^{\boldsymbol{\phi}}]$. Moreover, this bound attains equality for the optimal $\boldsymbol{\phi}^*$, and in such cases the value becomes 0.

Table 1. Performance comparison on CIFAR-10. The values in the first block are taken from the original paper.

Model	Unconditional		Conditional	
	FID↓	NFE↓	FID↓	NFE↓
DDPM (Ho et al., 2020)	3.17	1000	-	-
DDIM (Song et al., 2021a)	4.16	100	-	-
ScoreSDE (Song et al., 2021b)	2.20	2000	-	-
iDDPM (Nichol & Dhariwal, 2021)	2.90	1000	-	-
LSGM (Vahdat et al., 2021)	2.10	138	-	-
CLD-SGM (Dockhorn et al., 2022b)	2.25	312	-	-
STF (Xu et al., 2022b)	1.90	35	-	-
ST (Kim et al., 2022b)	2.33	2000	-	-
PFGM (Xu et al., 2022a)	2.35	110	-	-
INDM (Kim et al., 2022a)	2.28	2000	-	-
PFGM++ (Xu et al., 2023b)	1.93	35	-	-
PSLD (Pandey & Mandt, 2023)	2.10	246	-	-
ES (Ning et al., 2024)	1.95	35	1.80	35
EDM (Heun) (Karras et al., 2022)	2.01	35	1.83	35
	2.03	65	1.90	89
EDM+DG (Kim et al., 2023)	1.78	35	1.66	35
	1.90	65	1.72	89
EDM+Restart (Xu et al., 2023a)	1.95	43	1.85	43
	1.93	65	1.90	89
EDM+DiffRS (ours)	**1.59**	64.06	**1.52**	88.22

The proof is in Appendix A. If the discriminator is completely indistinguishable, i.e., $d_t^{\phi} \equiv 0.5$ for all t, then $R(\phi) = 0$ because $\bar{A}_t^{\phi} \equiv 1$ for all t, indicating that all instances are accepted in the rejection sampling process. Therefore, the refined distribution from DiffRS is same as the distribution from the pre-trained diffusion model. As the discriminator is trained, $R(\phi)$ converges to $-J(\boldsymbol{\theta})(\leq 0)$ according to Theorem 3.1, making the upper bound for DiffRS tighter than that for the pre-trained diffusion model.

4. Experiments

In this section, we empirically validate the proposed method, DiffRS. First, we conduct experiments on standard benchmark datasets for image generation tasks, such as CIFAR-10 (Krizhevsky, 2009), and ImageNet 64×64 and 256×256 (Deng et al., 2009). Next, we present the analysis of DiffRS and its applicability to fast diffusion samplers. Finally, we perform experiments on large-scale text-conditional image generation using Stable Diffusion (Rombach et al., 2022) with a resolution of 512×512.

Experimental Setting We primarily use the pre-trained networks on CIFAR-10 and ImageNet 64×64 from EDM (Karras et al., 2022), which is known for the superior performance of the pre-trained models. For ImageNet 256×256, we use the checkpoint from DiT (Peebles & Xie, 2023). Additional results on other datasets (e.g., FFHQ (Karras et al., 2019), AFHQv2 (Choi et al., 2020)) and networks (e.g., DDPM++ cont. (Song et al., 2021b)) are provided in Appendix D. All settings related to the discriminator are identical to Kim et al. (2023), which is provided in Appendix C.2. Note that the process of sampling from a pre-trained model and training a discriminator requires significantly less time and memory than training a diffusion model. Further experimental details are specified in Appendix C. We mainly evaluate the generation performance using the Fréchet Inception Distance (FID) (Heusel et al., 2017) on 50K samples, and we report the number of function evaluations (NFE) on the diffusion network. In the case of DiffRS, the NFE varies for each sample, so we take the average NFE of the samples.

Table 2. Performance comparison on class-conditional ImageNet 64×64. The values in the first block are from the original paper.

Model	FID↓	NFE↓
DDPM (Ho et al., 2020)	11.0	250
iDDPM (Nichol & Dhariwal, 2021)	2.92	250
ADM (Dhariwal & Nichol, 2021)	2.07	250
CFG (Ho & Salimans, 2021)	1.55	250
CDM (Ho et al., 2022a)	1.48	8000
RIN (Jabri et al., 2023)	**1.23**	1000
VDM++ (Kingma & Gao, 2023)	1.43	511
EDM (Heun) (Karras et al., 2022)	2.18	511
EDM (SDE) (Karras et al., 2022)	1.38	511
EDM+DG (Kim et al., 2023)	1.38	511
EDM+Restart (Xu et al., 2023a)	1.37	623
EDM+DiffRS (ours)	1.26	273.93

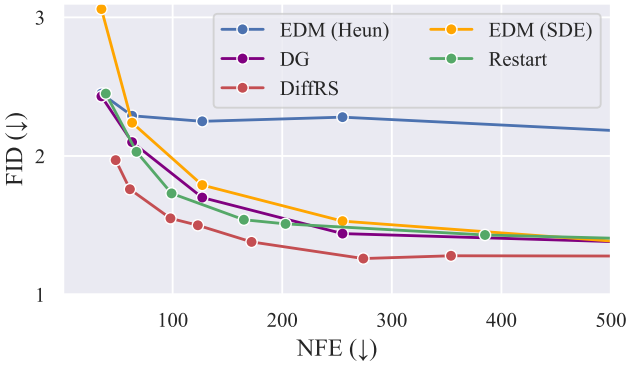


Figure 4. FID vs. NFE on ImageNet 64×64 with EDM.

4.1. Analysis on Benchmark Datasets

CIFAR-10 Table 1 presents the performance of previous diffusion models and our proposed method on CIFAR-10. The proposed method achieves new SOTA with FID scores of 1.59 for the unconditional case and 1.52 for the class-conditional case.

Table 3. Performance comparison on class-conditional ImageNet 256×256 with DiT-XL/2-G (Peebles & Xie, 2023). 'Time' is the average sampling time to generate 100 samples in minutes.

Sampler	NFE↓	Time↓	FID↓	sFID↓	IS↑	Prec↑	Rec↑	F1↑
DDPM (Ho et al., 2020)	250	3.71	2.30	4.72	277.2	0.826	0.579	0.681
	300	4.38	2.33	4.69	280.8	0.830	0.582	0.684
	415	5.91	2.30	**4.68**	279.8	0.831	0.572	0.678
DG (Kim et al., 2023)	250	4.02	1.88	5.15	284.1	0.786	0.633	0.701
	300	4.76	1.98	5.35	**287.9**	0.793	0.621	0.696
	375	5.87	1.83	4.99	**287.9**	0.791	0.624	0.698
DG+DiffRS (ours)	306.88	5.87	**1.76**	**4.68**	279.1	0.796	0.629	**0.703**

(a) $\sigma = 28.4$ (b) $\sigma = 1.92$ (c) $\sigma = 0.002$

Figure 5. Generated images with the highest (top) and lowest (bottom) acceptance probability at each timestep, obtained using the EDM (Heun) sampler on CIFAR-10. $\sigma = \{28.4, 1.92, 0.002\}$ corresponds to the $t = \{15, 9, 1\}$, respectively, with $T = 18$.

For a detailed analysis, the second block of Table 1 compares samplers that improve the sampling process using the same fixed pre-trained diffusion model on CIFAR-10. DiffRS exhibits the best performance under the same diffusion checkpoint. DiffRS is based on Heun's 2nd order sampler (Heun) with 35 NFEs, and the NFE is increased due to rejection. For a fair comparison, we evaluate the baseline samplers with the same NFEs as DiffRS, and we observe that DiffRS still outperforms other baseline samplers under the same NFEs.

ImageNet 64×64 Table 2 shows the performance for class-conditional image generation on ImageNet 64×64. We report the best FID performances over NFE for each method. DiffRS achieves competitive performance on class-conditional ImageNet 64×64, approaching SOTA with an FID score of 1.26 while requiring fewer NFEs compared to the current SOTA model (1.23 with 1000 NFEs).

In Figure 4, we evaluate the FID values of various NFEs for each method with the fixed pre-trained diffusion checkpoint on ImageNet 64×64. We compare with the deterministic sampler (Heun) and the stochastic sampler (SDE) proposed by EDM (Karras et al., 2022), DG (Kim et al., 2023), and Restart (Xu et al., 2023a). DG and DiffRS utilize Heun as the base sampler for small NFE regime and switch to the SDE sampler for large NFE regime. Restart employs Heun as the base sampler because the method is inherently based on the ODE sampler. DiffRS adjusted the backbone sampler and the value of γ to measure performance on different NFEs, as detailed in Appendix C.3. Notably, DiffRS consistently outperforms on all NFE regimes. We include the uncurated generated images in Appendix D.7. These results highlight the effective and efficient sampling capabilities of DiffRS from the provided pre-trained network information.

Table 4. Ablation studies on unconditional CIFAR-10.

	Methods	FID↓	NFE↓
	No rejection sampling	2.01	35
(a)	No sequential rejection sampling	3.73	295.34
(b)	Marginal sequential rejection sampling	1.66	63.57
(c)	Re-init. to $t+1$ by one-step forward only	1.84	47.69
(d)	Re-init. to T by prior distribution	1.72	138.07
	DiffRS	**1.59**	64.06

ImageNet 256×256 We perform the experiment on high-resolution class-conditional image generation using ImageNet 256×256 with DiT-XL/2-G (Peebles & Xie, 2023). We apply DiffRS to DG sampler, and we also measure the performances of DDPM and DG on comparable NFEs and sampling time. As shown in Table 3, DiffRS performs better than DDPM and DG on the FID metric. Additionally, DiffRS achieves performance on par with the best results for the sFID and F1 metrics, while DDPM and DG have lower performance on one of these metrics. Therefore, DiffRS can be effectively used for sample refinement in high-resolution image generation.

Acceptance Probability Figure 5 visualizes the top 10 and bottom 10 images for each timestep, determined by calculating the acceptance probability for 50,000 generated CIFAR-10 images sampled by the EDM (Heun) sampler. We observe that the top images have better visual quality. Conversely, for the bottom images, the images at large timesteps often have an overall unclear appearance, and the images at small timesteps have distortions in finer details. DiffRS effectively eliminates these problematic images, resulting in new high-quality images.

4.2. Ablation Studies

Sequential Rejection Sampling We investigate the effect on the sequential rejection sampling based on the transition kernel, considering two scenarios: (a) marginal rejection sampling only at $t = 0$ using L_0, and (b) sequential rejection sampling based on the marginal probability using L_t. As seen in (a) of Table 4, performance deteriorates without sequential rejection sampling, attributed to the challenges of density ratio estimation in high-dimensional data space (Rhodes et al., 2020). Additionally, rejections require iterative sampling from the prior distribution, significantly increasing the NFE. In contrast, DiffRS performs sequential rejection sampling utilizing the time-dependent density ratios. As t increases, the two distributions in the ratio become

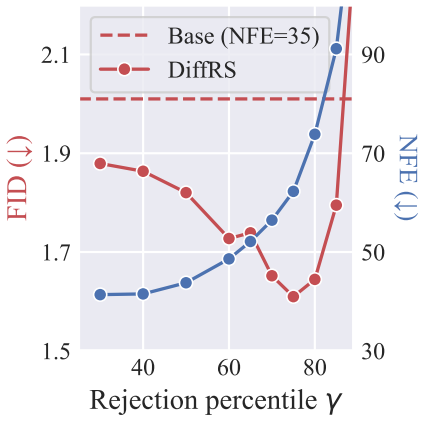


Figure 6. Sensitivity analysis of γ on unconditional CIFAR-10.

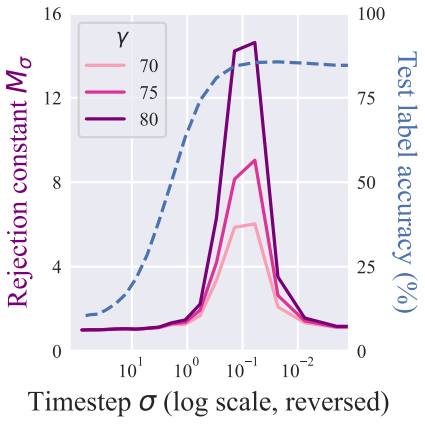


Figure 7. Rejection constant M_σ over timesteps on unconditional CIFAR-10.

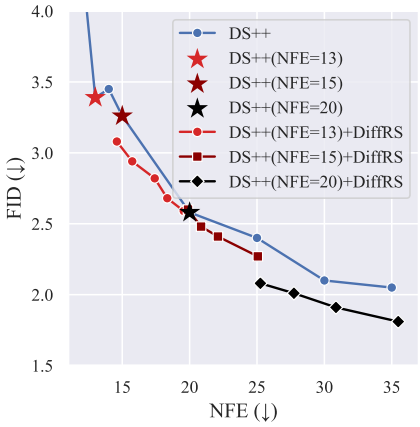


Figure 8. FID vs. NFE on unconditional CIFAR-10 with DPM-Solver++ (DS++).

closer, leading to relatively accurate ratio estimation (Kim et al., 2024). Moreover, rejections at intermediate timesteps contribute to a relative reduction in NFE. On the other hand, in case of (b), using the marginal probability for sequential rejection sampling leads to performance degradation due to the mismatch between sampling and proposal distributions.

Re-initialization The third block in Table 4 presents the variants of the re-initialization methods. In the case of rejection at timestep t, the first method, denoted (c), performs only one-step forward to $t+1$ and continues DiffRS from $t+1$; and the second method, denoted (d), transitions to timestep T and restarts DiffRS from the prior distribution. The results show that both variants outperform the backbone sampler, but fall short of the performance of the proposed re-initialization method. In the case of (c), the re-initialized samples could deviate from the true distribution q_{t+1}, leading to a drop in performance. In the case of (d), there is a significant increase in NFE because sampling is restarted from timestep T. In contrast, our re-initialization method performs additional rejection sampling on the samples obtained through the forward step, attempting to initialize similar to the true distribution. Furthermore, by conducting the adequate number of forward steps for each sample, our method achieves superior performance at suitable NFEs.

Rejection Constant Figure 6 shows the effect of the hyperparameter γ on FID and NFE on CIFAR-10, where the rejection percentile γ determines the rejection constants M_t in the experiment. We observe that the NFE increases exponentially with increasing γ. While DiffRS generally has a better FID than the base sampler, there is an increase beyond an extreme threshold of γ. We empirically observe that the FID tends to increase when the NFE exceeds 2-3 times that of the base sampler. Therefore, we set γ to keep the NFE at this level, typically in the range of $[75, 85]$.

Figure 7 visualizes the rejection constant M_σ over timesteps on unconditional CIFAR-10 under various rejection percentile γ. As the rejection constant is inversely proportional to the acceptance probability (Ripley, 2009), a higher rejection constant implies a higher proportion of rejected samples. The distribution of the rejection constant over timesteps is bell-shaped, with a peak around $\sigma = 0.1$. Interestingly, Restart (Xu et al., 2023a) also adds noise around this timestep, which was chosen heuristically.

To further analyze this interval, we include the test label accuracy of a time-dependent classifier trained by CIFAR-10 (blue dotted line). This result indicates the level of semantic information in the images at each timestep. We observe that the sample quality becomes distinguishable once a certain level of semantic information is reached. Also, in regions very close to the data space, the rejection rate decreases as the sample quality is almost determined.

4.3. Application to Fast Sampler

Diffusion models inherently suffer from problems of sampling speed due to the need for iterative sampling. To address this, various methods for fast sampling, such as the use of efficient ODE and SDE solvers, have been proposed (Jolicoeur-Martineau et al., 2021; Lu et al., 2022a; Dockhorn et al., 2022a; Zhang & Chen, 2023). Most of these methods aim to follow the perturbed data distribution $q_t(\mathbf{x}_t)$ at time t, making it possible to apply DiffRS to these fast samplers.

We verify this experimentally on unconditional CIFAR-10 with DPM-Solver++, one of the few-step accelerated sampling methods (Lu et al., 2022a;b). As shown in Figure 8, when comparing stars and line segments of the same color, we observe that although additional NFEs are incurred, the performance is improved compared to the base sampler. Additionally, we find that the performance is improved compared to the same NFEs of DPM-Solver++.

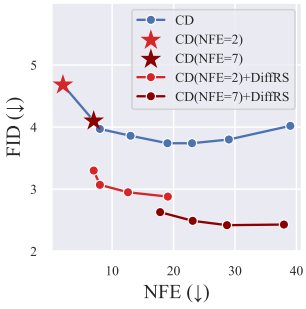


Figure 9. FID vs. NFE on ImageNet 64×64 with CD.

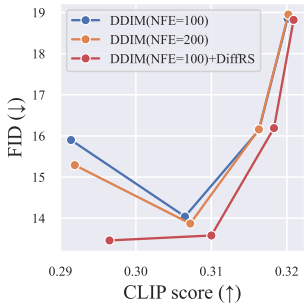
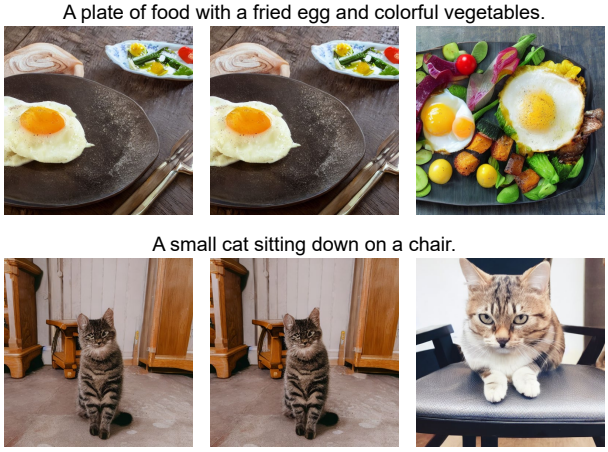


Figure 10. FID vs. CLIP score with Stable Diffusion v1.5.



(a) DDIM (NFE=100) (b) DDIM (NFE=200) (c) DDIM (NFE=100) + DiffRS (ours)

Figure 11. Example of generated images with Stable Diffusion v1.5. We use a classifier-free guidance weight of 2, and images on the same row are generated from the same noise from the prior distribution and the text prompt located above.

4.4. Application to Distillation Methods

Diffusion distillation methods are an alternative approach to accelerating the sampling process. They aim to obtain a distilled generative model with fewer NFEs from the information of the existing diffusion model process (Salimans & Ho, 2022; Song et al., 2023; Meng et al., 2023). As discussed in Section 3.4, DiffRS can be applied to diffusion distillation methods where an intermediate sample $\mathbf{x}_t$ is required to follow a perturbed data distribution $q_t(\mathbf{x}_t)$.

To investigate the effectiveness of DiffRS in distillation methods, we apply it to the Consistency Distillation (CD) (Song et al., 2023). We use CD with 2 and 7 NFEs as base samplers. For DiffRS, we adjust the hyperparameter γ to observe the changes in FID over NFE. Figure 9 shows that the combination of CD and DiffRS can generate images with an FID of less than 3.0 at an NFE nearly 10. This result suggests that DiffRS can also be effectively applied to diffusion distillation models.

4.5. Application to Large-scale Text-conditional Model

We further show that DiffRS can be applied to large-scale text-conditional diffusion models such as Stable Diffusion (Rombach et al., 2022). We use the publicly available Stable Diffusion v1.5 pre-trained on LAION-5B (Schuhmann et al., 2022) with a resolution of 512×512. We apply DiffRS to DDIM (Song et al., 2021a) with 100 NFEs. Following the evaluation protocol of previous studies (Nichol et al., 2022; Xu et al., 2023a), we generate 5,000 images from captions randomly sampled from the COCO (Lin et al., 2014) validation set using the classifier-free guidance method (Ho & Salimans, 2021). We evaluate the sample quality using the FID metric and measure the image-text alignment through the CLIP score (Hessel et al., 2021).

Figure 10 plots the trade-off between FID and CLIP scores, varying the classifier-free guidance weights. DiffRS exhibits a superior FID for the same CLIP score, with an average of 166 NFEs. In contrast, the performance of DDIM did not significantly improve even with an increased number of NFEs. Figure 11 visualizes the example of images generated by DDIM and ours. These results demonstrate the scalability of our model to effectively improve the sampling performance of a well-trained diffusion model even in text-to-image generation scenarios.

5. Conclusion

We present Diffusion Rejection Sampling (DiffRS), a new diffusion sampling approach that ensures alignment between the reverse transition and the true transition at each timestep. The acceptance probability is estimated by training a time-dependent discriminator. We also propose the re-initialization method for DiffRS to effectively and efficiently refine the rejected samples. Theoretical analysis shows that discriminator training tightens the upper bound on the divergence between the data distribution and the refined distribution by DiffRS. Empirically, DiffRS achieves the state-of-the-art performances on the benchmark datasets, and DiffRS demonstrates its effectiveness on few-step accelerated samplers, diffusion distillation models, and large-scale text-to-image generation models.

Potential future work includes applying advanced sampling methods, such as Metropolis-Hastings sampling (Turner et al., 2019), to diffusion models. Additionally, developing methods to deal with discrepancies between the data distribution learned by a pre-trained diffusion model and the target data distribution, such as focusing on minority samples or the presence of label noise (Um et al., 2024; Na et al., 2024), will be promising applications of DiffRS.

Acknowledgements

This research was supported by AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data (IITP) funded by the Ministry of Science and ICT (2022-0-00077).

Impact Statement

This paper primarily focuses on improving sample quality and efficiency in the diffusion generation process. The application of our method is promising in various fields such as art, design, and entertainment. However, ethical considerations, including the responsible use of AI-generated content and the prevention of harmful information creation, require careful attention. Implementing features such as a safety checker module and invisible watermarking can address some of these concerns.

References

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. Discriminator rejection sampling. In *International Conference on Learning Representations*, 2019.

Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8188–8197, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Dockhorn, T., Vahdat, A., and Kreis, K. Genie: Higher-order denoising diffusion solvers. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30150–30166. Curran Associates, Inc., 2022a.

Dockhorn, T., Vahdat, A., and Kreis, K. Score-based generative modeling with critically-damped langevin diffusion. In *International Conference on Learning Representations*, 2022b.

Grover, A., Gummadi, R., Lazaro-Gredilla, M., Schuurmans, D., and Ermon, S. Variational rejection sampling. In *International Conference on Artificial Intelligence and Statistics*, pp. 823–832. PMLR, 2018.

Hessel, J., Holtzman, A., Forbes, M., Le Bras, R., and Choi, Y. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528, 2021.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. Cascaded diffusion models for high fidelity image generation. *The Journal of Machine Learning Research*, 23(1):2249–2281, 2022a.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 8633–8646. Curran Associates, Inc., 2022b.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

Jabri, A., Fleet, D. J., and Chen, T. Scalable adaptive computation for iterative generation. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 14569–14589. PMLR, 23–29 Jul 2023.

Jolicoeur-Martineau, A., Li, K., Piché-Taillefer, R., Kachman, T., and Mitliagkas, I. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

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

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K.

(eds.), *Advances in Neural Information Processing Systems*, 2022.

Kim, D., Na, B., Kwon, S. J., Lee, D., Kang, W., and Moon, I.-c. Maximum likelihood training of implicit nonlinear diffusion model. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 32270–32284. Curran Associates, Inc., 2022a.

Kim, D., Shin, S., Song, K., Kang, W., and Moon, I.-C. Soft truncation: A universal training technique of score-based diffusion model for high precision score estimation. In *International Conference on Machine Learning*, pp. 11201–11228. PMLR, 2022b.

Kim, D., Kim, Y., Kwon, S. J., Kang, W., and Moon, I.-C. Refining generative process with discriminator guidance in score-based diffusion models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 16567–16598. PMLR, 23–29 Jul 2023.

Kim, Y., Na, B., Park, M., Jang, J., Kim, D., Kang, W., and chul Moon, I. Training unbiased diffusion models from biased dataset. In *The Twelfth International Conference on Learning Representations*, 2024.

Kingma, D. P. and Gao, R. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., and Aila, T. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

Lai, C.-H., Takida, Y., Murata, N., Uesaka, T., Mitsufuji, Y., and Ermon, S. Fp-diffusion: Improving score-based diffusion models by enforcing the underlying score fokker-planck equation. In *International Conference on Machine Learning*, pp. 18365–18398. PMLR, 2023.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022a.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.

Meng, C., Rombach, R., Gao, R., Kingma, D., Ermon, S., Ho, J., and Salimans, T. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

Na, B., Kim, Y., Bae, H., Lee, J. H., Kwon, S. J., Kang, W., and chul Moon, I. Label-noise robust diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

Nash, C., Menick, J., Dieleman, S., and Battaglia, P. Generating images with sparse representations. In *International Conference on Machine Learning*, pp. 7958–7968. PMLR, 2021.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Nichol, A. Q., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., Mcgrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pp. 16784–16804. PMLR, 2022.

Ning, M., Li, M., Su, J., Salah, A. A., and Ertugrul, I. O. Elucidating the exposure bias in diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

Pandey, K. and Mandt, S. A complete recipe for diffusion generative models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4261–4272, 2023.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Rhodes, B., Xu, K., and Gutmann, M. U. Telescoping density-ratio estimation. *Advances in neural information processing systems*, 33:4905–4916, 2020.

Ripley, B. D. *Stochastic simulation*. John Wiley & Sons, 2009.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 23–29 Jul 2023.

Turner, R., Hung, J., Frank, E., Saatchi, Y., and Yosinski, J. Metropolis-hastings generative adversarial networks. In *International Conference on Machine Learning*, pp. 6345–6353. PMLR, 2019.

Um, S., Lee, S., and Ye, J. C. Don't play favorites: Minority guidance for diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024.

Vahdat, A., Kreis, K., and Kautz, J. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

Voleti, V., Jolicoeur-Martineau, A., and Pal, C. MCVD - masked conditional video diffusion for prediction, generation, and interpolation. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Xu, Y., Liu, Z., Tegmark, M., and Jaakkola, T. Poisson flow generative models. *Advances in Neural Information Processing Systems*, 35:16782–16795, 2022a.

Xu, Y., Tong, S., and Jaakkola, T. S. Stable target field for reduced variance score estimation in diffusion models. In *The Eleventh International Conference on Learning Representations*, 2022b.

Xu, Y., Deng, M., Cheng, X., Tian, Y., Liu, Z., and Jaakkola, T. S. Restart sampling for improving generative processes. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Xu, Y., Liu, Z., Tian, Y., Tong, S., Tegmark, M., and Jaakkola, T. PFGM++: Unlocking the potential of physics-inspired generative models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38566–38591. PMLR, 23–29 Jul 2023b.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023.

Zheng, H., He, P., Chen, W., and Zhou, M. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. In *The Eleventh International Conference on Learning Representations*, 2023a.

Zheng, K., Lu, C., Chen, J., and Zhu, J. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

A. Proof of Theoretical Analysis

In this section we provide a proof of Theorem 3.1.

Theorem 3.1. *The KL divergence between data distribution q_0 and refined distribution $p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}$ is bounded by:*

$$D_{KL}(q_0||p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}) \leq J(\boldsymbol{\theta}) + R(\boldsymbol{\phi}) =: J(\boldsymbol{\theta},\boldsymbol{\phi}), \tag{18}$$

where $R(\boldsymbol{\phi}) := \mathbb{E}_{q_T}[-\log \bar{A}_T^{\boldsymbol{\phi}}] + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t,t+1}}[-\log \bar{A}_t^{\boldsymbol{\phi}}]$. Moreover, this bound attains equality for the optimal $\boldsymbol{\phi}^*$, and in such cases the value becomes 0*.*

Proof. First, we provide the derivation of Eq. (15), the upper bound of KL divergence between the data distribution and the model distribution, comes from (Ho et al., 2020).

$$\begin{aligned}
D_{\mathrm{KL}}(q_0||p_0^{\boldsymbol{\theta}}) &= \mathbb{E}_{q_0}[-\log p_0^{\boldsymbol{\theta}}(\mathbf{x}_0)] - H(q_0) && (19)\\
&\leq \mathbb{E}_{q_{0:T}}\Big[-\log \frac{p_{0:T}^{\boldsymbol{\theta}}(\mathbf{x}_{0:T})}{q_{1:T|0}(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big] - H(q_0) && (20)\\
&= \mathbb{E}_{q_{0:T}}\Big[-\log p_T^{\boldsymbol{\theta}}(\mathbf{x}_T) - \sum_{t=0}^{T-1} \log \frac{p_{t|t+1}^{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_{t+1|t}(\mathbf{x}_{t+1}|\mathbf{x}_t)}\Big] - H(q_0) && (21)\\
&= \mathbb{E}_{q_{0:T}}\Big[-\log p_T^{\boldsymbol{\theta}}(\mathbf{x}_T) - \sum_{t=0}^{T-1} \log \frac{p_{t|t+1}^{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})}\frac{q_t(\mathbf{x}_t)}{q_{t+1}(\mathbf{x}_{t+1})}\Big] - H(q_0) && (22)\\
&= \mathbb{E}_{q_{0:T}}\Big[-\log \frac{p_T^{\boldsymbol{\theta}}(\mathbf{x}_T)}{q_T(\mathbf{x}_T)} - \sum_{t=0}^{T-1} \log \frac{p_{t|t+1}^{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t+1})}{q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})} - \log q_0(\mathbf{x}_0)\Big] - H(q_0) && (23)\\
&= D_{\mathrm{KL}}(q_T||p_T^{\boldsymbol{\theta}}) + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t+1}}\Big[D_{\mathrm{KL}}(q_{t|t+1}||p_{t|t+1}^{\boldsymbol{\theta}})\Big] =: J(\boldsymbol{\theta}). && (24)
\end{aligned}$$

If we substitute $p_0^{\boldsymbol{\theta}}$ with $p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}$ in the above, the following equation for the refined distribution $p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}$ by DiffRS holds:

$$D_{\mathrm{KL}}(q_0||p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}) \leq D_{\mathrm{KL}}(q_T||p_T^{\boldsymbol{\theta},\boldsymbol{\phi}}) + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t+1}}\Big[D_{\mathrm{KL}}(q_{t|t+1}||p_{t|t+1}^{\boldsymbol{\theta},\boldsymbol{\phi}})\Big]. \tag{25}$$

By the relationship between $p^{\boldsymbol{\theta}}$ and $p^{\boldsymbol{\theta},\boldsymbol{\phi}}$, as described in Eqs. (16) and (17), each term in the upper bound is further derived as follows:

$$\begin{aligned}
D_{\mathrm{KL}}(q_T||p_T^{\boldsymbol{\theta},\boldsymbol{\phi}}) &= \mathbb{E}_{q_T}[-\log p_T^{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_T) + \log q_T(\mathbf{x}_T)] && (26)\\
&= \mathbb{E}_{q_T}[-\log p_T^{\boldsymbol{\theta}}(\mathbf{x}_T) - \log \bar{A}_T^{\boldsymbol{\phi}}(\mathbf{x}_T) + \log q_T(\mathbf{x}_T)] && (27)\\
&= D_{\mathrm{KL}}(q_T||p_T^{\boldsymbol{\theta}}) + \mathbb{E}_{q_T}[-\log \bar{A}_T^{\boldsymbol{\phi}}(\mathbf{x}_T)], && (28)
\end{aligned}$$

$$\begin{aligned}
D_{\mathrm{KL}}(q_{t|t+1}||p_{t|t+1}^{\boldsymbol{\theta},\boldsymbol{\phi}}) &= \mathbb{E}_{q_{t|t+1}}[-\log p_{t|t+1}^{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_t|\mathbf{x}_{t+1}) + \log q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})] && (29)\\
&= \mathbb{E}_{q_{t|t+1}}[-\log p_{t|t+1}^{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t+1}) - \log \bar{A}_t^{\boldsymbol{\phi}}(\mathbf{x}_t,\mathbf{x}_{t+1}) + \log q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})] && (30)\\
&= D_{\mathrm{KL}}(q_{t|t+1}||p_{t|t+1}^{\boldsymbol{\theta}}) + \mathbb{E}_{q_{t|t+1}}[-\log \bar{A}_t^{\boldsymbol{\phi}}(\mathbf{x}_t,\mathbf{x}_{t+1})] && (31)
\end{aligned}$$

Therefore, we can derive the upper bound of the KL divergence as follows:

$$D_{\mathrm{KL}}(q_0||p_0^{\boldsymbol{\theta},\boldsymbol{\phi}}) \leq D_{\mathrm{KL}}(q_T||p_T^{\boldsymbol{\theta}}) + \mathbb{E}_{q_T}[-\log \bar{A}_T^{\boldsymbol{\phi}}(\mathbf{x}_T)] + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t+1}}\Big[D_{\mathrm{KL}}(q_{t|t+1}||p_{t|t+1}^{\boldsymbol{\theta}}) + \mathbb{E}_{q_{t|t+1}}[-\log \bar{A}_t^{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{x}_{t+1})]\Big] \tag{32}$$

$$= J(\boldsymbol{\theta}) + \mathbb{E}_{q_T}[-\log \bar{A}_T^{\boldsymbol{\phi}}(\mathbf{x}_T)] + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t,t+1}}[-\log \bar{A}_t^{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{x}_{t+1})] \tag{33}$$

$$= J(\boldsymbol{\theta}) + R(\boldsymbol{\phi}) =: J(\boldsymbol{\theta}, \boldsymbol{\phi}). \tag{34}$$

where $R(\boldsymbol{\phi}) := \mathbb{E}_{q_T}[-\log \bar{A}_T^{\boldsymbol{\phi}}] + \sum_{t=0}^{T-1} \mathbb{E}_{q_{t,t+1}}[-\log \bar{A}_t^{\boldsymbol{\phi}}]$.

Moreover, the optimal discriminator ϕ^* satisfies that:

$$\bar{A}_T^{\boldsymbol{\phi}^*}(\mathbf{x}_T) = \frac{q_T(\mathbf{x}_T)}{p_T^{\boldsymbol{\theta}}(\mathbf{x}_T)}, \text{ and } \bar{A}_t^{\boldsymbol{\phi}^*}(\mathbf{x}_t, \mathbf{x}_{t+1}) = \frac{q_{t|t+1}(\mathbf{x}_t|\mathbf{x}_{t+1})}{p_{t|t+1}^{\boldsymbol{\theta}}(\mathbf{x}_t|\mathbf{x}_{t+1})}. \tag{35}$$

Substituting $\bar{A}_T^{\boldsymbol{\phi}^*}(\mathbf{x}_T)$ into Eq. (27) and $\bar{A}_t^{\boldsymbol{\phi}^*}(\mathbf{x}_t, \mathbf{x}_{t+1})$ into Eq. (30) respectively, we observe that each KL term becomes zero. Consequently, the upper bound $J(\boldsymbol{\theta}, \boldsymbol{\phi}) = 0$, which leads to the KL divergence on the data space, $D_{\mathrm{KL}}(q_0||p_0^{\boldsymbol{\theta},\boldsymbol{\phi}})$, to be zero. □

B. Related Works

B.1. Reducing Sampling Error of Diffusion Models

The sampling error can be measured by the distribution discrepancy between the data distribution and the generated distribution. This error is decomposed into three factors: the network approximation error, the prior mismatch error, and the temporal-discretization error (Kim et al., 2022a). To reduce the temporal-discretization error, reducing the sampling interval, which increases the iterative sampling count, is a common strategy; but it comes at the cost of a higher number of network evaluations, which slows down the sampling speed.

A significant amount of research has focused on improving the expressiveness of diffusion models through advances in network architecture or objective structure. For example, some studies proposed loss weights for timesteps or regularization methods for the diffusion objectives (Kim et al., 2022b; Kingma & Gao, 2023; Lai et al., 2023). Additionally, alternative approaches involve the investigation of the effective latent space (Vahdat et al., 2021; Rombach et al., 2022; Kim et al., 2022a). Other efforts aim at learning an implicit prior distribution to minimize the prior mismatch error and reduce the sampling length (Zheng et al., 2023a). However, these methods require time-consuming training of the diffusion model.

B.2. Rejection Sampling

Several researches utilize rejection sampling to discard poor samples for better generation quality in generative models. Grover et al. (2018) propose the rejection sampling on the approximated variational posterior of variational autoencoder. Azadi et al. (2019) introduce the rejection sampling by utilizing the discriminator of the generative adversarial network (GAN) to adjust the implicit distribution of the GAN generator. Similarly, Turner et al. (2019) combine the Metropolis-Hastings algorithm and GAN. However, there is no previous attempt to improve the sampling quality of the diffusion model via rejection sampling. It should be noted that it is difficult to naively apply the rejection sampling to the diffusion model due to the nature of its iterative sampling process.

C. Additional Experimental Settings

C.1. Configurations of Baseline Samplers

We use the baseline samplers as follows: Heun's 2$^{\text{nd}}$ ODE sampler (Heun) (Karras et al., 2022), Improved SDE sampler (SDE) (Karras et al., 2022), DG (Kim et al., 2023), and Restart (Xu et al., 2023a) for the standard benchmark datasets; and DDIM (Song et al., 2021a) for the text-to-image generation task. We adopt the sampling hyperparameter settings from the

experiments of the original papers. In cases where the experiment was not performed in the original paper, we used settings as similar as possible. In the CIFAR-10, FFHQ, and AFHQv2 experiments, we use the Heun sampler serves as the backbone sampler for DG and Restart. In the ImageNet 64×64 experiments, we use the better sampler between Heun and SDE at each NFE as the backbone sampler for DG, while we use Heun for Restart. In the ImageNet 256×256 experiments, we use the DDPM sampler as the backbone sampler for DG. For DPM-Solver++ (Lu et al., 2022b), we apply the singlestep DPM-Solver++. For the diffusion distillation method, we apply the multi-step consistency sampling for the consistency distillation model (Song et al., 2023).

C.2. Settings of Discriminator Training

We follow DG (Kim et al., 2023) to train a time-dependent discriminator by utilizing the code and some checkpoints from the DG repositories.[2][3] We use the provided checkpoints for CIFAR-10 and FFHQ generation, and we train our own discriminator for other datasets. Our discriminator is trained on a single NVIDIA GeForce RTX 4090 GPU using CUDA 11.8 and PyTorch 1.12 versions. The discriminator structure consists of two stacked U-net encoders. The pre-trained U-net encoder is from ADM (Dhariwal & Nichol, 2021) utilized as a feature extractor.[4] We utilize a randomly initialized feature extractor for the COCO dataset and pre-trained extractor with ImageNet classification for the remaining dataset. The shallow U-net encoders are only the trainable parameters for discriminating, which maps from feature to logits. For the conditional diffusion backbones, the shallow U-net encoders are also designed as a conditional model. The specific configurations are described in Table 5.

Table 5. Configurations of the discriminator.

	CIFAR-10		ImageNet64		ImageNet256	FFHQ	AFHQv2	COCO
Diffusion Backbone								
Model	EDM	EDM	EDM	CD	DiT-XL/2	EDM	EDM	Stable Diffusion
Conditional model	✗	✔	✔	✔	✔	✗	✗	✔
Feature Extractor								
Model	ADM	ADM	ADM	ADM	ADM	ADM	ADM	ADM
Architecture	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder
Pre-trained	✔	✔	✔	✔	✔	✔	✔	✗
Depth	4	4	4	4	4	4	4	4
Width	128	128	128	128	128	128	128	128
Attention Resolutions	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8
Input shape (data)	(B,32,32,3)	(B,32,32,3)	(B,64,64,3)	(B,64,64,3)	(B,32,32,4)	(B,64,64,3)	(B,64,64,3)	(B,64,64,4)
Output shape (feature)	(B,8,8,512)	(B,8,8,512)	(B,8,8,512)	(B,8,8,512)	(B,8,8,384)	(B,8,8,512)	(B,8,8,512)	(B,8,8,512)
Discriminator								
Model	ADM	ADM	ADM	ADM	ADM	ADM	ADM	ADM
Architecture	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder	U-Net encoder
Pre-trained	✔	✔	✗	✗	✗	✔	✗	✗
Depth	2	2	2	2	2	4	2	2
Width	128	128	128	128	128	128	128	128
Attention Resolutions	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8	32,16,8
Input shape (feature)	(B,8,8,512)	(B,8,8,512)	(B,8,8,512)	(B,8,8,512)	(B,8,8,384)	(B,8,8,512)	(B,8,8,512)	(B,8,8,512)
Output shape (logit)	(B,1)	(B,1)	(B,1)	(B,1)	(B,1)	(B,1)	(B,1)	(B,1)
Discriminator Training								
Time scheduling	VP	VP	Cosine VP	Cosine VP	VP	Cosine VP	Cosine VP	Cosine VP
Time sampling	Importance	Importance	Importance	Importance	Importance	Importance	Importance	Importance
Time weighting	$\frac{g^2}{\sigma^2}$	$\frac{g^2}{\sigma^2}$	$\frac{g^2}{\sigma^2}$	$\frac{g^2}{\sigma^2}$	$\frac{g^2}{\sigma^2}$	$\frac{g^2}{\sigma^2}$	$\frac{g^2}{\sigma^2}$	$\frac{g^2}{\sigma^2}$
Batch size	128	128	128	128	512	128	128	128
# data samples	50,000	50,000	50,000	50,000	50,000	60,000	15,803	5,000
# generated samples	25,000	50,000	50,000	50,000	50,000	60,000	15,803	5,000
# Epoch	60	250	20	50	20	250	20	10

C.3. Configurations of DiffRS

We integrate DiffRS into the code implementation of each base sampler: DG codebase[2] for EDM-based samplers; DG-ImageNet codebase[3] for ImageNet 256×256; DPM-Solver-v3 (Zheng et al., 2023b) codebase[5] for DPM-Solver++; Con-

[2] https://github.com/alsdudrla10/DG
[3] https://github.com/alsdudrla10/DG_imagenet
[4] https://github.com/openai/guided-diffusion
[5] https://github.com/thu-ml/DPM-Solver-v3

Table 6. Configuration details for each experimental result.

Pre-trained diffusion			Performance		Configuration		
Dataset	Task	Model	FID↓	NFE↓	Base sampler	Rejection percentile γ	Max. iteration K
CIFAR-10	Unconditional	DDPM++ cont.	1.91	151.86	EDM (SDE) (NFE=63)	85	∞
		EDM	1.59	64.06	EDM (Heun) (NFE=35)	75	105
			1.88	41.37	EDM (Heun) (NFE=35)	30	∞
			1.86	41.52	EDM (Heun) (NFE=35)	40	∞
			1.82	43.78	EDM (Heun) (NFE=35)	50	∞
			1.73	48.61	EDM (Heun) (NFE=35)	60	∞
			1.74	52.13	EDM (Heun) (NFE=35)	65	∞
			1.65	56.45	EDM (Heun) (NFE=35)	70	∞
			1.60	62.28	EDM (Heun) (NFE=35)	75	∞
			1.64	73.84	EDM (Heun) (NFE=35)	80	∞
			1.79	91.17	EDM (Heun) (NFE=35)	85	∞
		EDM	3.08	14.60	DPM-Solver++ (NFE=13)	20	∞
			2.94	15.74	DPM-Solver++ (NFE=13)	30	∞
			2.82	17.41	DPM-Solver++ (NFE=13)	40	∞
			2.68	18.34	DPM-Solver++ (NFE=13)	50	∞
			2.59	19.56	DPM-Solver++ (NFE=13)	60	∞
			2.60	19.88	DPM-Solver++ (NFE=15)	40	∞
			2.48	20.85	DPM-Solver++ (NFE=15)	45	∞
			2.41	22.10	DPM-Solver++ (NFE=15)	50	∞
			2.27	25.06	DPM-Solver++ (NFE=15)	60	∞
			2.08	25.25	DPM-Solver++ (NFE=20)	40	∞
			2.01	27.74	DPM-Solver++ (NFE=20)	50	∞
			1.91	30.86	DPM-Solver++ (NFE=20)	60	∞
			1.81	35.48	DPM-Solver++ (NFE=20)	70	∞
CIFAR-10	Class-conditional	EDM	1.52	88.22	EDM (Heun) (NFE=35)	80	105
FFHQ	Unconditional	EDM	1.60	198.65	EDM (Heun) (NFE=71)	90	213
AFHQv2	Unconditional	EDM	1.80	144.92	EDM (Heun) (NFE=71)	85	213
ImageNet 64×64	Class-conditional	EDM	1.97	48.23	EDM (Heun) (NFE=27)	60	∞
			1.76	60.95	EDM (Heun) (NFE=27)	70	∞
			1.55	98.60	EDM (Heun) (NFE=27)	80	∞
			1.50	123.00	EDM (SDE) (NFE=63)	60	∞
			1.38	171.95	EDM (SDE) (NFE=63)	70	∞
			1.26	273.93	EDM (SDE) (NFE=127)	70	∞
			1.27	353.60	EDM (SDE) (NFE=127)	75	∞
			1.27	1169.67	EDM (SDE) (NFE=511)	70	∞
		CD	3.30	7.00	CD (NFE=2)	70	∞
			3.07	8.01	CD (NFE=2)	75	∞
			2.95	12.58	CD (NFE=2)	85	∞
			2.88	19.07	CD (NFE=2)	90	∞
			2.63	17.81	CD (NFE=7)	60	∞
			2.49	23.11	CD (NFE=7)	80	∞
			2.42	28.69	CD (NFE=7)	85	∞
			2.43	37.96	CD (NFE=7)	90	∞
ImageNet 256×256	Class-conditional	DiT-XL/2	1.76	306.88	DDPM+DG (NFE=250)	65	∞
COCO	Text-to-image	Stable Diffusion (weight=2)	13.46	166.95	DDIM (NFE=100)	80	∞
		Stable Diffusion (weight=3)	13.58	166.36	DDIM (NFE=100)	80	∞
		Stable Diffusion (weight=5)	16.19	217.13	DDIM (NFE=100)	80	∞
		Stable Diffusion (weight=8)	18.82	115.24	DDIM (NFE=100)	80	∞

sistency Models codebase[6] for CD; and Restart codebase[7], built on Diffusers[8], for Stable Diffusion. For the benchmark datasets, we utilize a single NVIDIA GeForce RTX 4090 GPU, CUDA 11.8, and PyTorch 1.12. For the text-to-image generation, we use a single NVIDIA L40S GPU with CUDA 11.8 and PyTorch 2.1. Our implementation is available at: https://github.com/aailabkaist/DiffRS.

To estimate the rejection constant M_t, we generate 1,000 samples with evaluating the unnormalized acceptance probability $\bar{A}_t^{\phi} = \frac{\hat{L}_t^{\phi}(\mathbf{x}_t)}{\hat{L}_{t+1}^{\phi}(\mathbf{x}_{t+1})}$ using the trained discriminator. Then, we select the γ^{th} percentile values from these values as the rejection

[6] https://github.com/openai/consistency_models
[7] https://github.com/Newbeeer/diffusion_restart_sampling
[8] https://github.com/huggingface/diffusers

Table 7. Performance on FFHQ and AFHQv2 with EDM (Karras et al., 2022).

Sampler	FFHQ		AFHQv2	
	FID	NFE	FID	NFE
EDM (Heun) (Karras et al., 2022)	2.41	71	2.00	71
	2.43	199	2.05	145
DG (Kim et al., 2023)	1.96	71	1.88	71
	1.93	199	1.85	145
DiffRS (ours)	**1.60**	198.65	**1.80**	144.92

Table 8. Performance on CIFAR-10 with DDPM++ cont. (Song et al., 2021b).

Sampler	FID↓	NFE↓
EDM (Heun) (Karras et al., 2022)	2.89	63
EDM (SDE) (Karras et al., 2022)	2.35	1023
Restart (Xu et al., 2023a)	2.11	519
DiffRS (ours)	1.91	151.86

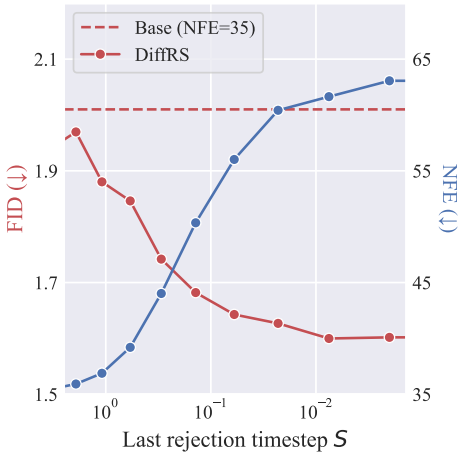


Figure 12. Trade-off between FID and NFE on unconditional CIFAR-10 varying the last rejection timestep.

constant for each timestep, with the minimum value of M_t set to one. Additionally, we set a maximum iteration K to prevent looping within a single path. If this limit is exceeded, we initialize the sampling again from the prior distribution. In most cases, we set K to either ∞ or three times the NFE of the base sampler. The hyperparameters for each experiment, along with their corresponding performance, are provided in Table 6.

C.4. Configurations of Pre-trained Diffusion Models

For CIFAR-10, we employ the pre-trained DDPM++ cont. and EDM models obtained from the EDM repository.[9] For FFHQ and AFHQv2, we use the pre-trained EDM models also available in the EDM repository.[9] In the case of ImageNet 64×64, we use the pre-trained EDM model from the EDM repository[9], and the consistency distillation model from the Consistency Model repository.[6] For ImageNet 256×256, we use the pre-trained DiT-XL/2 from the DiT repository.[10] In the text-to-image generation task, we use Stable Diffusion v1.5 pre-trained on LAION-5B, available from HuggingFace.[11]

C.5. Evaluation Procedure

We evaluate the performance of diffusion models using Fréchet Inception Distance (FID). FID calculations are performed using the DG (Kim et al., 2023) code, and we report the results for the random seeds. For ImageNet 256×256, we also report Inception Score (IS) (Salimans et al., 2016), sFID (Nash et al., 2021), Precision (Prec), Recall (Rec), and F1 of Prec and Rec (Kynkäänniemi et al., 2019), evaluated by ADM (Dhariwal & Nichol, 2021) code. In the stable diffusion experiment, FID and CLIP score calculations are conducted using the Restart code. CLIP scores are evalated using the open-sourced ViT-g/14 (Ilharco et al., 2021).

D. Additional Experiment Results

D.1. Experimental Results on FFHQ and AFHQv2

In Table 7, we present the performance on FFHQ (Karras et al., 2019) and AFHQv2 (Choi et al., 2020). We use the Heun with 71 NFEs as the base sampler for DffRS and compare DiffRS to Heun and DG. Remarkably, DiffRS demonstrates significant improvements in FID over the base sampler on these benchmark datasets (+0.81 for FFHQ and +0.20 for AFHQv2). In addition, our method exhibits superior performance even with similar NFEs.

[9] https://github.com/NVlabs/edm
[10] https://github.com/facebookresearch/DiT
[11] https://huggingface.co/runwayml/stable-diffusion-v1-5

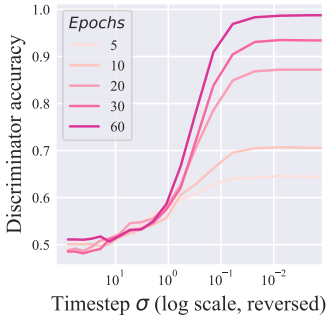


Figure 13. Accuracy of discriminator over each timestep varying discriminator training epochs, on unconditional CIFAR-10.

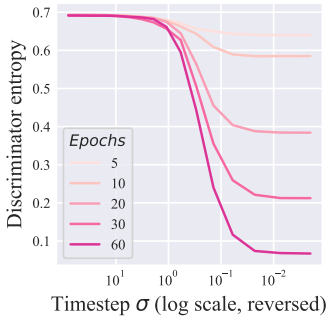


Figure 14. Entropy of discriminator over each timestep varying discriminator training epochs, on unconditional CIFAR-10.

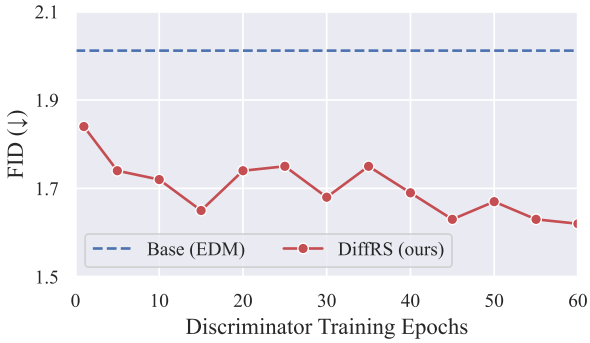


Figure 15. FID performance over discriminator training epochs on unconditional CIFAR-10.

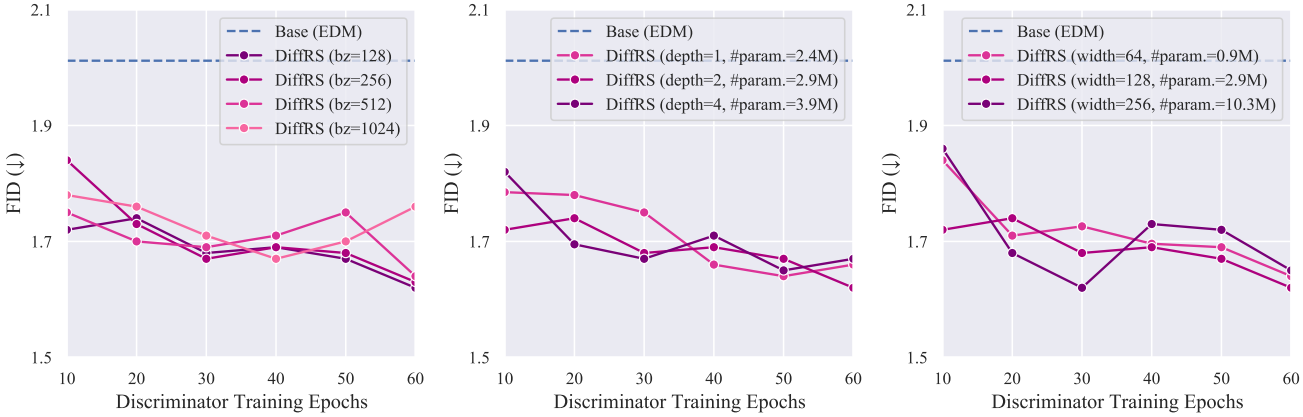


Figure 16. Ablation studies of the discriminator configurations on unconditional CIFAR-10. Each subfigure is for (top) batch size, (middle) depth of U-Net, and (bottom) width of U-Net. `bz` stands for batch size and `#param.` is the number of discriminator parameters.

D.2. Experimental Results on DDPM++ cont.

Table 8 shows the performance with the DDPM++ cont. model (Song et al., 2021b) on the unconditional CIFAR-10 dataset. The baseline results are taken from the reported performance of Restart (Xu et al., 2023a). We find that DiffRS shows the superior performance. Therefore, DiffRS works effectively for other diffusion backbones as well.

D.3. Additional Ablation Studies

Figure 12 shows the changes in FID and NFE when DiffRS is applied only up to S instead of applying it to all timesteps. As S increases, indicating a smaller interval for applying rejection sampling, the FID degrades. Similar to the analysis of the rejection constant in the main manuscript (Figure 7), a drastic change in FID and NFE is observed around $\sigma = 0.1$.

D.4. Ablation Studies of Discriminator

Training Curve As shown in Figures 13 and 14, unlike GAN training, the discriminator training of our method is stable. This is because the score network that serves as the generator is pre-trained and fixed. Therefore, this is a single directional optimization process without min-max game, such as GAN. Experimentally, we plot the sample performance according to the number of discriminator training epochs. As shown in Figure 15, we find that the performance improves and stabilizes already from the early epochs.

Configurations We perform an ablation study to explore the effects of the discriminator configurations. We measured the sample performance across training epochs, varying the discriminator training batch size, and the depth and width of the U-Net. As shown in Figure 16, we observed superior performance compared to the base sampler across all settings.

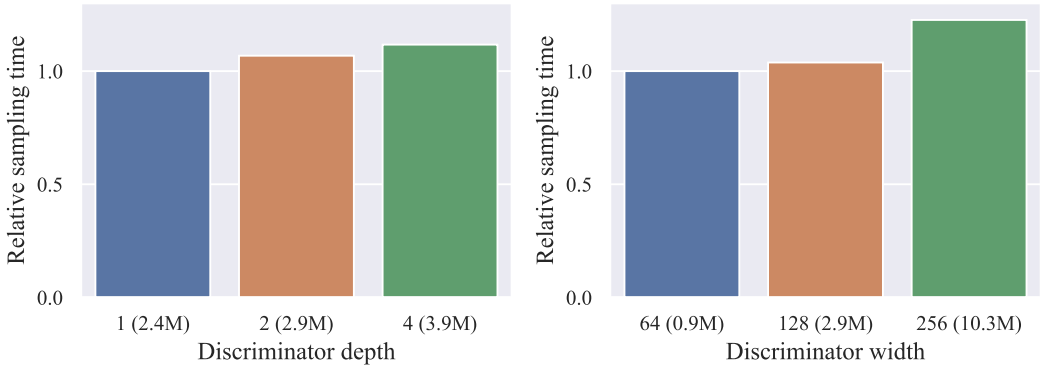


Figure 17. Relative sampling time varying discriminator configurations. The numbers in parentheses indicate the number of parameters in the discriminator.

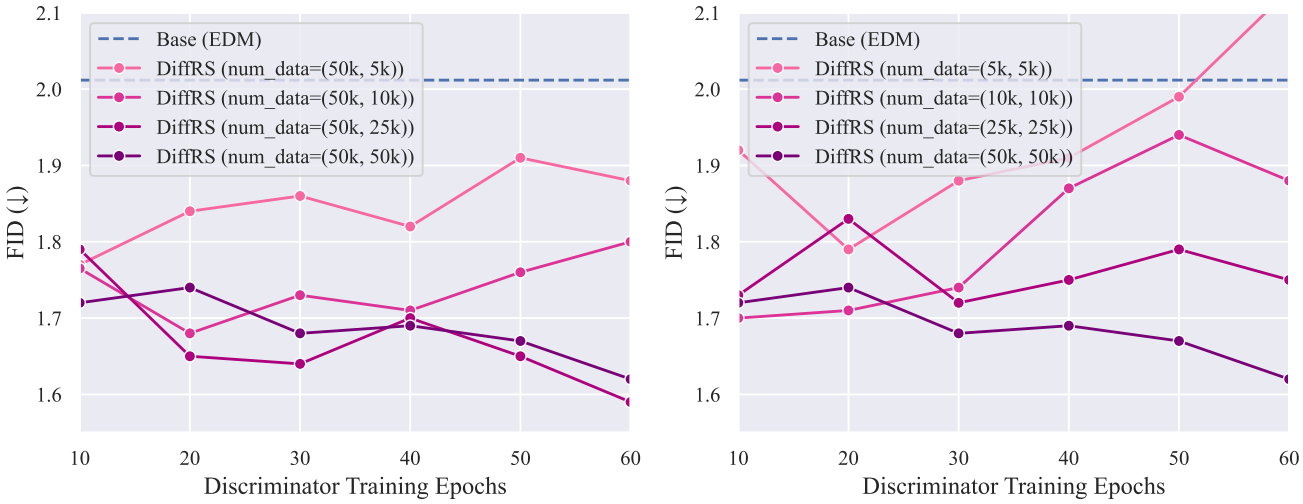


Figure 18. Ablation studies of the number of training samples for the discriminator on unconditional CIFAR-10. The tuple in the legend represents (number of training data, number of generated data).

Also, we plot the sampling time according to the discriminator structure in Figure 17. As shown in the figure, although the sampling time increases slightly as the discriminator parameter size increases, the evaluation time of the diffusion models accounts for a larger proportion, resulting in a non-significant difference.

Number of Samples We examine the performance according to the number of discriminator training data. We perform experiments on unconditional CIFAR-10 in two settings: 1) using all training images (50k examples) and varying the number of generated images, 2) using the same number of training images as generated images.

As shown in Figure 18, when using all training images, even generating only 10% of the training images (5k images) outperforms the base sampler. Also, we observe improved performance as the number of generated images increases. However, when matching the number of training images to the number of generated images, we observe a decrease in performance as training progresses when the number of samples is small. This also happens when all training images are used, but the number of generated images is small. We attribute this to the reduced number of training images leading to overfitting problems, resulting in inaccurate density ratio estimation by the discriminator.

Therefore, it is preferable to use all of the given training data, and more generated data generally improves performance. However, even using only 10% of the training data can provide better performance than the base sampler.

Table 9. Sampling time (seconds) to generate 100 samples at 63 NFEs.

EDM (Heun)	EDM (SDE)	DG	DiffRS
45.69	45.97	59.66	54.69

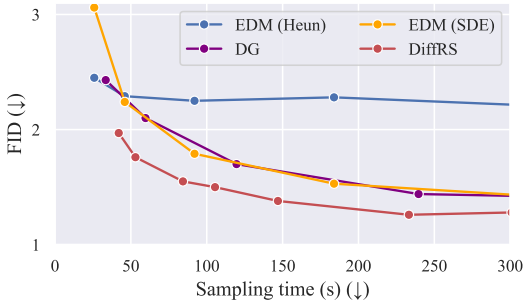


Figure 20. FID vs. Sampling time (seconds per 100 images) on ImageNet 64×64 with EDM.

D.5. Estimation of Rejection Constant M_t

Theoretically, M_t should always be greater than the ratio of the target distribution to the proposal distribution for all instances. Therefore, a proper estimator of M_t would be the maximum value of the density ratio extracted from the samples, i.e., the rejection percentile $\gamma = 100(\%)$. However, in most our experiments, adjusting γ in the range of [75, 85] worked well. Specifically, Figure 6 in the main manuscript illustrates the performance variation with respect to γ, where it can be observed that the FID increases significantly as γ becomes very large.

We believe that such cases are due to problems with the discriminator network used to estimate the density ratio. To investigate this, we measure the entropy and accuracy of the discriminator outputs of the training dataset for each timestep over discriminator training epochs. As shown in Figures 13 and 14, for small epochs, both prediction confidence and accuracy are low, and as the epochs increase, confidence and accuracy increase significantly. This could indicate that overconfidence problems occur as training progresses, possibly leading to an inaccurate density ratio estimate that is skewed toward extreme values, thus degrading performance.

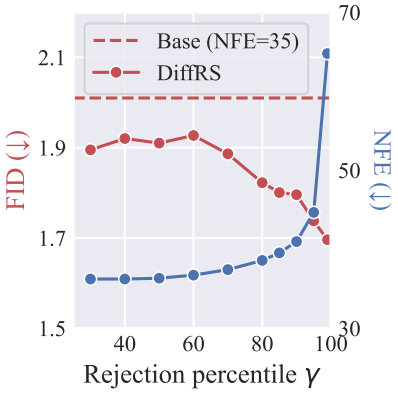


Figure 19. Sensitivity analysis of γ with early-stage discriminator (trained by 5-epochs) on unconditional CIFAR-10.

In this case, we believe that lowering the rejection constant M_t by the rejection percentile γ helped alleviate the problem of overconfidence in the discriminator. To investigate this further, we did a small experiment by limiting the training of discriminator to suppress the overconfidence problem. We examine the performance changes with respect to γ for the early-stage discriminator (i.e., trained by 5-epochs). As shown in Figure 19, the early-stage discriminator continues to perform better as γ increases. While the performance is generally better than the baseline, it did not reach the best performance (FID=1.59) of the final discriminator (trained by 60-epochs). Therefore, while it is necessary to train the discriminator beyond a certain level, the overconfidence problem of neural networks can occur, but this can be mitigated by adjusting γ.

D.6. Sampling Time

DG and DiffRS require additional sampling time due to the use of an auxiliary discriminator network. DG requires discriminator evaluation and gradient computation at each timestep, while DiffRS only requires discriminator evaluation at each timestep. Table 9 shows the sampling time taken to generate 100 samples at the same NFE. DiffRS takes longer than the base samplers because of the discriminator evaluation, and DG takes more time due to the gradient computation. Figure 20 illustrates the FID changes in terms of the sampling time required to generate 100 samples. As seen in the figure, our model demonstrates superior performance for the same sampling time.

D.7. Generated Images

Figures 21 to 24 show the generated images of DiffRS on the benchmark datasets. Figures 25 and 26 provide the uncurated conditional generated images using the base sampler and DiffRS on the ImageNet 64×64 to enable direct comparison of sample quality. For the consistency distillation model, Figure 27 compares the generated images of the base samplers and our method. Figure 28 provides the text-conditional generated images with a resolution of 512×512 from Stable Diffusion.

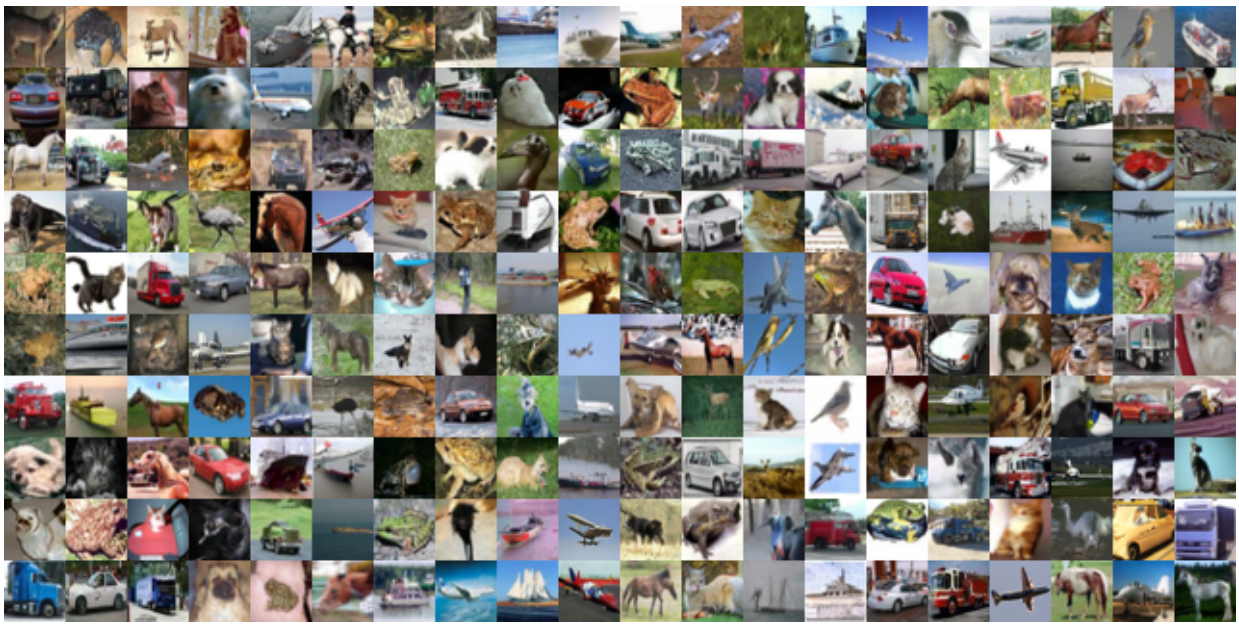


Figure 21. The uncurated generated images of DiffRS on unconditional CIFAR-10 with EDM (NFE=64.06, FID=1.59).

Figure 22. The uncurated generated images of DiffRS on conditional CIFAR-10 with EDM (NFE=88.22, FID=1.52).

Figure 23. The uncurated generated images of DiffRS on unconditional FFHQ with EDM (NFE=198.65, FID=1.60).

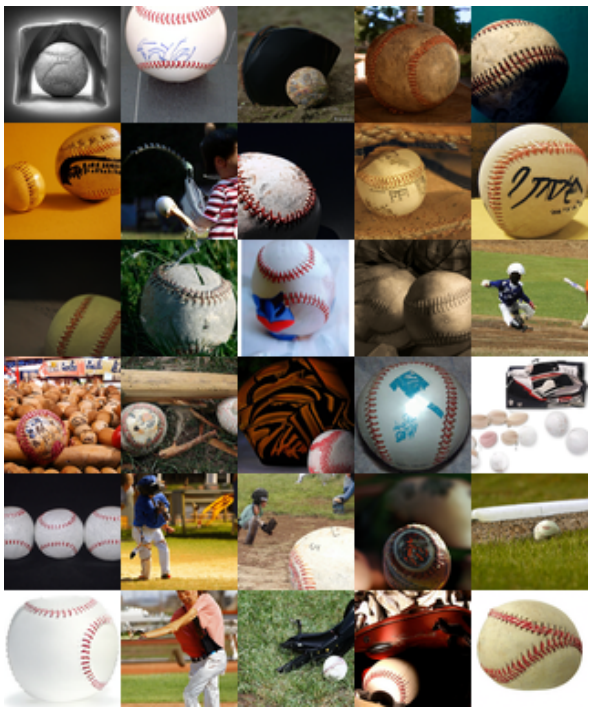
Figure 24. The uncurated generated images of DiffRS on unconditional AFHQv2 with EDM (NFE=144.92, FID=1.80).

(a) EDM (SDE) (NFE=127, FID=1.79)

(b) EDM (SDE) + DiffRS (NFE=273.93, FID=1.26)

Figure 25. The uncurated generated images of `flamingo` class of ImageNet 64×64 with EDM.

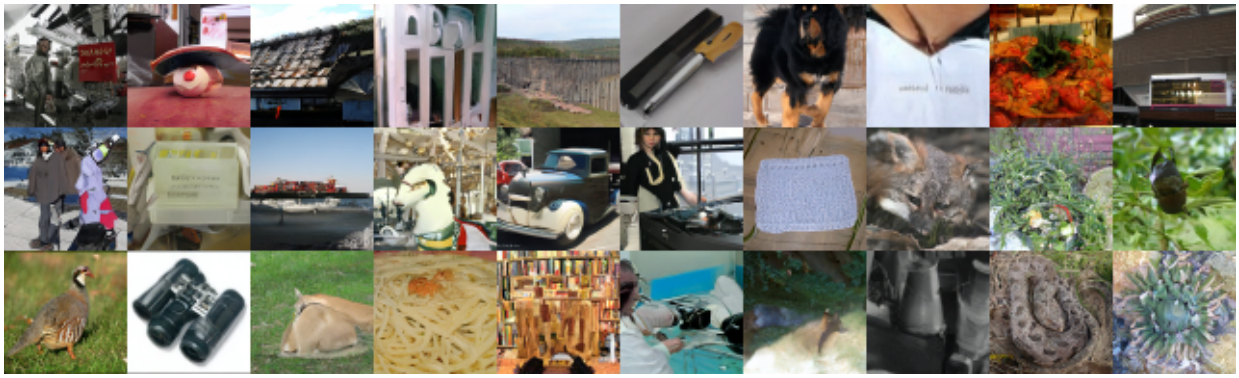
(a) EDM (SDE) (NFE=127, FID=1.79)



(b) EDM (SDE) + DiffRS (NFE=273.93, FID=1.26)

Figure 26. The uncurated generated images of `baseball` class of ImageNet 64×64 with EDM.

(a) CD (NFE=2, FID=4.68)



(b) CD (NFE=8, FID=3.97)

(c) CD (NFE=2) + DiffRS (NFE=8.01, FID=3.07)

Figure 27. The uncurated generated images of (a-b) CD-based sampler and (c) DiffRS on conditional ImageNet 64×64 dataset with CD.

(a) DDIM (NFE=100, FID=15.90)

(b) DDIM (NFE=200, FID=15.29)

(c) DDIM (NFE=100) + DiffRS (ours) (NFE=166.95, FID=13.21)

Figure 28. The uncurated generated images, with a resolution of 512×512, corresponding to the text prompt `A photo of an astronaut riding a horse on mars`, using Stable Diffusion v1.5 with a classifier-free guidance weight of 2.