EstimDiff: Accelerating Diffusion Models with Estimation

Aida Ebrahimi
University of Toronto
aidaa@cs.toronto.edu

Abstract

Diffusion models have gained popularity for their ability to produce high-quality and diverse samples without relying on adversarial training. However, their computational cost remains a significant challenge for real-world applications. In this paper, we propose a novel approach to accelerate diffusion models while preserving sample quality. Our method capitalizes on the observation that the updates made by the neural network in consecutive time steps of Denoising Diffusion Models (DDMs) often follow a pattern and direction for a certain number of steps. First, by approximately predicting six upcoming steps based on the previous two steps and denoising based on this estimation, we achieve a 4x speedup with almost no visible loss in image quality. Furthermore, we propose an algorithm that dynamically determines the number of steps to approximate at each period of time based on multiple factors. Our experiments show that using this algorithm can result in more than a 20x speedup and surpass the sample quality of previous works with the same amount of computation.

1 Introduction

Deep generative models have shown great promise in producing high-quality samples in various domains [26]. Generative Adversarial Networks (GANs) [6], Variational Autoencoders (VAEs) [12], and Denoising Diffusion Models (DDMs) [7] are among the most well-known and widely used generative models. An ideal generative model is one that can generate samples that are both diverse and of high quality and also exhibit a fast sampling rate [35]. However, in reality, most current generative learning frameworks compromise between high-quality sampling, mode coverage (diversity), and fast and computationally inexpensive sampling [35]. For example, DDMs are capable of generating high-quality samples, and they benefit from good mode coverage, which translates to good diversity in the generated samples. In fact, recent work shows that DDMs can outperform GANs in generation quality thanks to likelihood computation and much more stable training dynamics [19]. However, the main problem with DDMs is their slow sampling rate [30]. The slow sampling comes from the nature of DDMs, which require hundreds to thousands of steps for generating a single sample, and since each of these steps requires one inference through a neural network, they are extremely slower than generative models like GANs, which only require one inference for generating a sample.

In this paper, we tackle the problem of slow sampling in diffusion models. To overcome this issue, we introduce a new approach that divides the sampling steps into two categories: real steps and estimated steps.

During the real steps, a neural network is employed to perform denoising on the data. In contrast, the estimated steps rely on the previous outputs of the neural network to estimate the current output. To be more specific, our approach involves estimating the output at each timestep by considering the changes between the output of the last real step and the output of the real step immediately preceding it. We build our method on top of the two solvers, first DDIM, proposed in [30] and second PNDM,

proposed in [15], which is an extension of DDIM and claims to produce higher quality samples compared to DDIM.

We make our estimation based on two observations: first, the input data is highly similar between consecutive steps, and second, the neural network is expected to move closer to the data with each step. This enables us to move forward with denoising in the right direction without the need for expensive computations. By using the estimated values, our method compensates for any quality loss that would occur if no estimation were used. Our experiments demonstrate that our approach produces images of comparable quality to those generated by the original DDIM and PNDM methods, but four times faster.

Additionally, to achieve even greater speedup, we have developed a novel algorithm called the "Guided Skipping Algorithm". This innovative approach dynamically determines the number of steps to skip at each point in time and identifies the optimal cutoff point to transition from approximation to normal skipping. What sets this algorithm apart is its ability to adapt the step number according to individual samples, resulting in varying step sizes across samples.

With the implementation of the Guided Skipping Algorithm, we have demonstrated remarkable results, achieving over 20 times faster sample generation while maintaining a high level of fidelity to the samples generated with the standard 1000 steps.

In summary, our approach leverages the inherent similarity of input data and the expected behavior of the neural network to accelerate the sampling process while ensuring high-quality samples.

2 Background

Generative models, enabled by deep learning, are powerful tools capable of generating new data samples based on the dataset they were trained on [21]. These models find applications in various fields including image and video generation [9], text-based image editing [11], data augmentation [**?**], and medicine [17]. By learning and capturing the underlying probability distribution of the training data, generative models generate new data that resembles the training data [21].

GANs are one of the most popular classes of generative models. They framed the process of learning the underlying data distribution as a two-player adversarial game between a generator network, responsible for generating new data, and a discriminator network, responsible for distinguishing fake and real data. Through this adversarial training process, GANs learn to generate highly realistic and diverse samples. The generator network learns to transform random noise into meaningful data samples, while the discriminator network learns to differentiate between real and generated samples [6].

GANs are capable of generating high-fidelity samples with high levels of detail in a relatively short amount of time. However, as their biggest drawback, training GANs can be challenging, and they may face instability during training. The optimization process involves finding a delicate balance between the generator and discriminator networks. This often requires careful tuning and extensive computational resources. Moreover, they may suffer from mode collapse, where the generator produces a limited variety of samples that the discriminator accepts as real samples [32].

VAEs are another popular class of generative models. They consist of an encoder network and a decoder network. Together, these two networks form an autoencoder architecture. The encoder network maps the input data to a latent space, and the decoder network reconstructs the original input data from this latent space representation [12].

By imposing a probabilistic constraint on the latent space, VAEs enable the generation of new data samples by sampling from the known distribution, which allows for the generation of diverse and novel samples. Also, by manipulating the latent space, they can be used for interpolation and to control the generated outputs[14]. However, training VAEs can also be challenging. And more importantly, VAEs can suffer from producing blurry reconstructions and limited sample quality.

Denoising diffusion models are a new class of generative models that promise diverse and high-quality samples without the need for adversarial training. DDMs are based on the idea of using a Markov chain to gradually transform one distribution into another. These models are inspired by non-equilibrium statistical physics [29]. In DDMs, a Markov chain converts a simple and known distribution, such as a Gaussian distribution, into the desired target distribution, which represents the

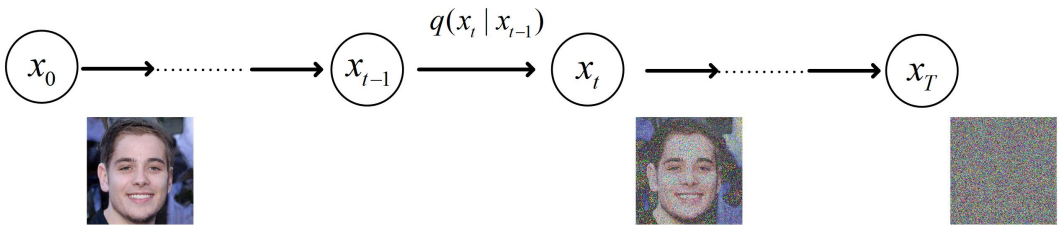

Figure 1: Forward diffusion process

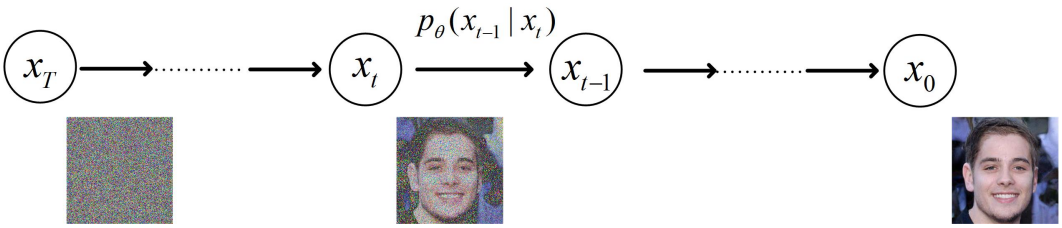

Figure 2: Reverse diffusion process

data we want to model. This transformation happens through a diffusion process. Where each step in the chain gradually alters the distribution and makes it more similar to the desired distribution[29] [7].

In diffusion models, the forward diffusion process gradually destroys desired data with noise addition (Figure 1), and a reverse diffusion process, using a neural network, reconstructs the image, i.e., removes the noise that it estimates was added at each step (Figure 2). Doing so enough times and with good data, the neural network eventually learns to estimate the underlying (original) data distribution. Then, we can simply start with noise and use the trained neural network to generate a new image representative of the original training dataset.

2.1 Forward Diffusion Process

Given a data point, x_0, sampled from a real data distribution $x_0 \sim q(x)$, the forward diffusion process is defined as the process of adding small amounts of Gaussian noise to the sample and producing a sequence of noisy samples $x_1, x_2, ...x_T$ in T timesteps, as shown in figure 1. Assuming the mean of the Gaussian noise distribution added to the data at timestep t is denoted by $\sqrt{\alpha_t}$ and the Variance denoted by $\beta_t = 1 - \alpha_t$, we can obtain the perturbed data at timestep t given the data at timestep $t-1$ using the following formula:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, 1 - \alpha_t\mathbf{I}) \tag{1}$$

and then, using this equation, we can write equation 2 for obtaining x_t from x_0:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2}$$

Thus, we can express x_t as a linear combination of x_0 and a noise variable ϵ:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \tag{3}$$

where $\bar{\alpha}_t$ is equal to $\prod_{i=1}^{t} \alpha_i$. Equation 3 gives us a well-defined form for obtaining any noisy sample, given x_0 Furthermore, from equation 3 we can see that when $\bar{\alpha}_t$ is equal to or close to 0, x_t will be equal to ϵ which means that it will resemble Gaussian distribution.

2.2 Reverse Diffusion Process

Reversing the above process, and sampling x_{t-1} from x_t, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, allows for reversing the diffusion process and recreating the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Calculating the exact value of $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is not tractable, and thus, diffusion models use a neural network with parameters θ, ϵ_θ, to estimate this value. As shown in figure 2, at each timestep t, given the sample x_t, x_{t-1} is calculated, with calculating $p_\theta(x_{t-1}|x_t)$, which is obtained using the output of the neural network at timestep t, $\epsilon_\theta(x_t, t)$. The model's parameters are optimized using variational inference. Variational inference is a powerful technique used in probabilistic modeling to approximate complex probability distributions when direct calculations are infeasible. This approximation is achieved by minimizing the dissimilarity between the variational distribution and the true posterior distribution. In DDMs, this translates to minimizing the difference between $p_\theta(x_{t-1}|x_t)$ and $q(x_{t-1}|x_t)$, where $p_\theta(x_{t-1}|x_t)$ is the approximated previous distribution. In turn, minimizing this difference can be translated into minimizing the variational lower bound:

$$\max_\theta \mathbb{E}_{q(x_0)} \left[\log p_\theta(x_0)\right] \leq \max_\theta \mathbb{E}_{q(x_0,x_1,\ldots,x_T)} \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T}|x_0)\right] \tag{4}$$

In Denoising Diffusion Probabilistic Model (DDPM) [7], they fixed all the conditionals as Gaussians with trainable mean functions and fixed variances, and thus the objective in equation 4 was simplified to:

$$L(\epsilon_\theta) := \sum_{t=1}^{T} \mathbb{E}_{x_0\sim q(x_0),\epsilon_t\sim N(0,I)} \left[\| \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t) - \epsilon_t \|_2^2\right] \tag{5}$$

From a trained model, x_0 is sampled by first sampling x_T from the Gaussian distribution, and then sampling x_{t-1} from the generative processes iteratively, with the following formula for sampling:

$$x_{t-1} = \sqrt{\bar{\alpha_{t-1}}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1 - \bar{\alpha_{t-1}} - \sigma_t^2}\epsilon_\theta(x_t, t) + \sigma_t\epsilon_t \tag{6}$$

where σ represents an arbitrary real vector $\sigma \in \mathbb{R}^T_{\geq 0}$, defining the level of stochasticity in the sampling process. Varying choices of σ lead to distinct generative processes, all of which utilize the same model ϵ_θ, thereby obviating the need for re-training the model. When $\sigma_t = \sqrt{\frac{1-\alpha_{t-1}}{1-\alpha_t}}\sqrt{\frac{1-\alpha_t}{\alpha_{t-1}}}$ for all t, the forward process becomes Markovian, and the generative process becomes a DDPM.

DDPMs, introduced in 2020, have shown promising results[7] and are the foundation for many future works on generating samples with diffusion models. The length T of the forward process is an important hyperparameter in DDPMs. From a variational perspective, a large T allows the reverse process to be close to a Gaussian[29], so that the generative process modeled with Gaussian conditional distributions becomes a good approximation; this motivates the choice of large T values, such as $T = 1000$ in DDPMs. The biggest drawback of DDPMs is that all T iterations have to be performed sequentially, and thus, sampling from DDPMs is much slower than sampling from other deep generative models. This makes them impractical for tasks where compute resources are limited and latency is critical, and it has inspired many future works on accelerating diffusion models.

3 Related Work

DDPMs were introduced in 2020 and have shown promising results[7] for generating samples of high quality and high diversity. However, generating a single sample with DDPMs requires hundreds to thousands of iterative steps. It is a well-known fact that there is a trade-off between the sample quality and the number of iterative steps in DDMs [36]. This has prompted various proposals to reduce the number of steps while minimizing the impact on quality [30] [8] [20]. One broad class of approaches that accelerate diffusion models involves using some form of learning process. While effective in reducing the number of steps, these approaches introduce a computational overhead due to the requirement of additional training. One approach that has garnered attention lately is to start with a non-Gaussian sample, or, in other words, provide the non-Gaussian sample as the input to the model [4] [38] [16].

One of the methods in this regard is called "Come Closer, Diffuse Faster" (CCDF), which starts the diffusion from timestep t_0, a value smaller than the original number of timesteps, T [4]. CCDF claims that it can mitigate the error resulting from not starting from the Gaussian distribution due to the contraction property of the reverse diffusion path [22]. The contraction property refers to the

characteristic of the equation that reduces errors rapidly. Thus, the reverse conditional diffusion path rapidly diminishes the error exponentially, meaning that estimation errors decrease at a fast rate.

Another approach of the same nature is the Early-Stopped DDPM (ES-DDPM). Similarly, ES-DDPM starts with a non-Gaussian noise distribution q. Their work is based on the observation that q can be approximated by diffusing samples from the real data distribution. Many existing generative models provide a good approximation of the real data distribution. Therefore, ES-DDPM incorporates a pre-trained generative model like GAN [10] or VAE [13] to generate the replacement for the Gaussian noise input, enabling sampling from q to correspond to diffusing samples obtained from the pre-trained generative model. Additionally, a similar approach involves training a GAN concurrently for generating samples instead of using pre-trained GANs [38]. However, a limitation of the last two aforementioned approaches is that the GAN will be responsible for generating the coarse outline of the data, which could potentially bottleneck the variety of the data, resulting in reduced diversity compared to what is expected from diffusion models. Another drawback of these approaches is that they don't work with pre-trained models.

Another class of works attempts to accelerate diffusion models through the use of knowledge distillation [27] [18]. Distillation refers to the process of transferring knowledge from one model to another. One noteworthy approach in [27] involves the authors proposing a scheme to reduce the number of steps to only two. They achieved this by starting with a pre-trained model and then using knowledge distillation to combine every two consecutive steps. This results in halving the number of steps. Then, by progressively repeating this operation, halving the number of steps each time, they reduced the number of steps to only two. However, this will require extensive computation, and the number of parameters in the student model must be extensive so that it can mimic the model with 1000 steps.

Another approach, known as GENIE [5], utilizes the truncated Taylor method and trains an additional model on top of a first-order score network to create a second-order solver. The Taylor method is a mathematical technique for approximating functions using polynomials. The score network refers to the neural network that estimates the gradient of a given data distribution with respect to the input. By using higher-order gradients (Jacobian vector products) derived from the perturbed data distribution, GENIE achieves improved sample quality with fewer steps.

Dynamic Programming techniques have also been proposed to accelerate the sampling rate of diffusion models [34]. In [34], the authors suggested treating the selection of inference time schedules as an optimization problem. This resulted in the design of an exact dynamic programming algorithm that can identify a set of good discrete time schedules for any pre-trained DDPM. Their approach utilizes the decomposition of Evidence Lower Bound (ELBO) into separate Kullback-Leibler (KL) terms. ELBO is a measure used in variational inference to approximate the likelihood of data given a probabilistic model. By decomposing ELBO into KL terms, they can analyze and optimize each term individually. Kullback-Leibler (KL) divergence is a mathematical concept that quantifies the difference between two probability distributions. In this context, it measures the dissimilarity between the true data distribution and the distribution generated by the diffusion model. Through the process of maximizing the training Evidence Lower Bound (ELBO), it becomes feasible to identify an optimal set of timesteps for each model. However, a significant challenge arises as the variational lower bound does not strongly correlate with sample quality, thereby imposing limitations on the overall performance of the method [1].

In a similar approach, the authors of [34] pursued the optimization of the sampling procedure directly using Kernel Inception Distance (KID) [2]. KID serves as a perceptual evaluation metric, assessing the similarity between generated and real samples based on feature extraction by a pre-trained Inception model. One major drawback of their work was the necessity for extensive and time-consuming training.

In an effort to address this limitation, the authors in [1] proposed a different strategy by optimizing the model directly for a specified budget of timesteps instead of modifying the sampling procedure. However, a new challenge arises as this approach requires training for each model and for each number of timesteps. This additional requirement poses a potential hurdle in terms of computational resources and time constraints.

Leveraging assistance from other neural network models is another way to accelerate both the training and inference of diffusion models. A highly successful method in this regard is reducing

the computation cost of diffusion models by decreasing their input dimension [24][8][25][3][23][33]. For example, [24] proposed performing the diffusion process on the latent space of an autoencoder model. These approaches are particularly useful because they reduce the number of parameters in the model by decreasing the input size. This results in a more compact model, greatly facilitating the training process. This is crucial because when it comes to training diffusion models, memory limitations can become a bottleneck [24], preventing learning from occurring, especially for models with a large number of parameters.

In an attempt to enhance the quality of generated samples, the authors of [37] introduced a technique called Spectral Diffusion (SD). SD is suitable for lightweight image synthesis and it incorporates wavelet gating within its architecture to facilitate dynamic feature extraction at each reverse step. This is fruitful because latent diffusion models tend to exhibit a significant drop in fidelity when the number of parameters is limited [37]. Additionally, traditional DDMs have an inherent bias against generating high-frequency components and learn to recover different frequency components at different time steps. As a result, compact networks struggle to accurately represent frequency dynamics, particularly in high-frequency estimation. To address this limitation, SD employs spectrum-aware distillation, which promotes high-frequency recovery by inversely weighting the objective based on spectrum magnitude. These approaches have demonstrated their effectiveness in preserving sample quality. However, it is worth noting that they still rely on iterative sampling, which can be time-consuming, which means that they will still not support fast sampling.

Another approach proposed in [35] combines steps by incorporating a GAN as the neural network instead of the U-net architecture used by DDPMs. By utilizing GANs, they aim to condense many steps in the reverse process into one. The main issue with this approach is that it alters the structure of the diffusion models and cannot be applied to pre-trained models. Additionally, it would also necessitate adversarial training.

Another class of approaches that accelerate diffusion models are those that do not require a learning process. These approaches can also work with pre-trained models. Furthermore, they are less computationally expensive. The main drawback of these approaches is that they may affect the quality of the generated photos.

One of the most famous approaches in this category is denoising diffusion implicit models (DDIMs) [30]. DDIMs aim to accelerate the sampling process and produce high-quality samples. They achieve this by altering the sampling process in the original DDPMs to a non-Markovian process. This non-Markovian process corresponds to a deterministic generative process, resulting in implicit models that generate high-quality samples at a much faster rate. Although DDIMs demonstrate improved sample quality compared to DDPMs, they still suffer from quality degradation when the number of steps is small.

"In [31], the authors leveraged the Stochastic Differential Equation (SDE) formulation of diffusion models to develop a faster sampling method. More precisely, they proposed the utilization of SDE solvers with adaptive timestep sizes. This was a response to the observation that existing approaches heavily rely on the Euler-Maruyama (EM) solver with a fixed step size, and straightforwardly replacing it with other SDE solvers results in a degradation in sample quality or slower performance than EM. To overcome this challenge, they designed an SDE solver with adaptive step sizes tailored specifically for score-based generative models.

In an attempt to enhance generated sample quality, the authors of [15] propose to treat DDPMs as solvers of differential equations on manifolds. Within this framework, they introduce pseudo-numerical methods for diffusion models (PNDMs), which offer a means to solve differential equations on manifolds. In other words, they changed the scheduler of the diffusion model. Furthermore, they demonstrate that DDIMs can be viewed as simple instances of pseudo-numerical methods.

4 Step Estimation

Several approaches have been proposed to skip certain steps in diffusion models to improve their computational efficiency. Some of these methods focus on changing the sampling equation and employ inexpensive algorithms like linear skipping [30] to decide which steps to take and which steps to skip. Others suggest more sophisticated methods for determining the optimal steps to take and skip, attempting to learn the most efficient approach [28]. However, none of these approaches

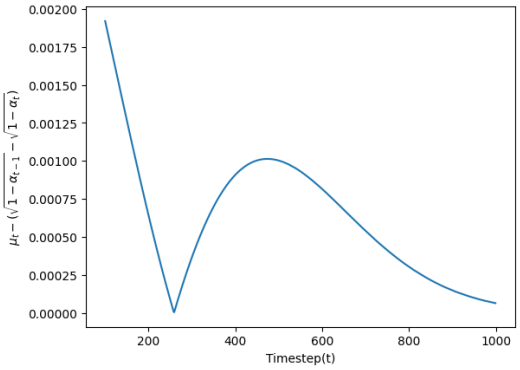

Figure 3: Difference between μ_t and $\sqrt{1-\alpha_{t-1}} - \sqrt{1-\alpha_t}$ for timesteps 0 to 1000

attempt to denoise the information without assistance from the neural network. In the rest of this section, we establish a mathematical foundation for the expected behavior of DDIMs [30]. Based on this understanding, we then develop a cost-effective formula to estimate the neural network's output at a given timestep using its output from previous timesteps.

4.1 Input Similarity

In this section, we will approximate the difference between the means of the absolute values of two consecutive inputs of the diffusion model. It is important to note that the aim of this section is to demonstrate that the inputs change gradually, allowing us to make these approximations.

In DDIMs, we use the following equation for calculating the input of the model at timestep $t-1$:

$$x_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}}\epsilon_\theta(x_t, t) \tag{7}$$

This equation is very similar to equation 6, with one key difference: it assumes $\sigma = 0$ for all timesteps, t which will result in the sampling process being deterministic. Moreover, for simplicity in the equations, we substituted $\bar{\alpha}_t$, with α_t, this results in defining the mean of the Gaussian noise added to the data at timestep t as $\frac{\sqrt{\alpha_t}}{\sqrt{\alpha_{t-1}}}$ instead of $\sqrt{\alpha_t}$. As a quick reminder, in equation 7, x_t is the sample at timestep t, $\frac{\sqrt{\alpha_t}}{\sqrt{\alpha_{t-1}}}$ is the mean value of the noise added to the input in the forward process at timestep t, and ϵ_θ is the neural network with parameters θ which means that $\epsilon_\theta(x_t, t)$ is the output of the neural network given the input x_t and at timestep t. We want to approximate x_{t-1} based on x_t, thus, we can re-write equation 7 in the following form:

$$x_{t-1} = \frac{\sqrt{\alpha_{t-1}}}{\sqrt{\alpha_t}}(x_t) - \frac{\sqrt{\alpha_{t-1}}}{\sqrt{\alpha_t}}(\sqrt{1-\alpha_t}\epsilon_\theta(x_t, t)) + \sqrt{1-\alpha_{t-1}}\epsilon_\theta(x_t, t) \tag{8}$$

Considering that α_t exhibits absolute decreasing behavior in relation to t, we introduce μ_t, defined for $t \in [2, 1000]$, which fulfills the following condition:

$$\sqrt{\mu_t} = \sqrt{\alpha_{t-1}} - \sqrt{\alpha_t} \tag{9}$$

Leveraging the fact that without trained α values the difference between values of α_t and α_{t-1}, i.e., μ_t is very small, and also since $\sqrt{\alpha_t}$ is a small value by the nature of DDMs, we can make the following approximation:

$$\sqrt{1-\alpha_t} - \sqrt{1-\alpha_{t-1}} \approx \sqrt{\mu_t} \tag{10}$$

In other words, we are assuming that the following equation is true.

$$\sqrt{1-\alpha_{t-1}} - \sqrt{1-\alpha_t} = \sqrt{\alpha_t} - \sqrt{\alpha_{t-1}} = \sqrt{\mu_t} \tag{11}$$

It is worth noting that in reality, the difference between the value of $\sqrt{1-\alpha_{t-1}} - \sqrt{1-\alpha_t}$ and the value of $\sqrt{1-\alpha_{t-1}} - \sqrt{1-\alpha_t}$, for most values of t is less than 0.001. For a better demonstration, the value of their absolute difference in timesteps, $t \in (100, 1000)$ is shown in figure 3. Based on the approximation made on equation 11, we can rewrite equation 8 in the following form:

$$x_{t-1} \approx \frac{\sqrt{\alpha_t} + \sqrt{\mu_t}}{\sqrt{\alpha_t}}(x_t - \sqrt{1-\alpha_t}\epsilon_\theta(x_t, t)) + (\sqrt{1-\alpha_t} - \sqrt{\mu_t})\epsilon_\theta(x_t, t) \tag{12}$$

By comparing equation 12 and equation 7, we anticipate their discrepancy to be on the order of 0.0001. This expectation arises from the similarity of the initial segments of both equations, with the only source of error being $(\sqrt{1-\alpha_t} - \sqrt{\mu_t})\epsilon_\theta(x_t, t)$. More precisely, the error can be estimated as $|(\sqrt{1-\alpha_t} - \sqrt{\mu_t}) - \sqrt{1-\alpha_{t-1}}|\epsilon_\theta(x_t, t)$. It is worth noting that $\epsilon_\theta(x_t, t)$ emulates Gaussian noise with a mean value of $\sqrt{\frac{\alpha_t}{\alpha_{t-1}}}$, which is inherently small.

To verify the accuracy of this approximation and to make sure the value of error caused by this is small compared to the data, we examine the impact of the approximation made in equation 11 on equation 7, as depicted in figure 4. In this figure, the mean of the absolute values obtained from equation 7 is presented for various timesteps, along with the corresponding error caused by equation 11 for 20 images generated using models trained on the LSUN Church, LSUN Bedroom, and CelebA datasets. To better demonstrate the magnitude of the error relative to the actual values, we calculate the ratio of the mean of the absolute data values to the mean of the absolute error values for different timesteps. These ratios are illustrated in figure 5 As depicted in the figure, these ratios fall within the range of 16000 to 300, confirming that the value of the errors is small in relation to the actual values. Moreover, to make sure and demonstrate that these values do not vary from one dataset to the other, we have shown three datasets in figure 5, all of which show the same behavior.

For calculating the difference between x_{t-1} and x_t, we can rewrite equation 12 in the following form:

$$x_{t-1} \approx x_t + \sqrt{\mu_t}\left(\frac{1}{\sqrt{\alpha_t}}(x_t - \epsilon_\theta(x_t, t)) - \epsilon_\theta(x_t, t)\right) \tag{13}$$

Based on the above equation, the difference between the value of x_t and x_{t-1} comes from the term $\sqrt{\mu_t}(\frac{1}{\sqrt{\alpha_t}}(x_t - \epsilon_\theta(x_t, t)) - \epsilon_\theta(x_t, t))$. $\frac{\sqrt{\mu}}{\sqrt{\alpha_t}}$ is depicted in figure 6 and is between 0.00008 to 0.01. It is important to note that, as our input information becomes more valuable and meaningful, i.e., as we go forward with the denoising process, the value of $\frac{\sqrt{\mu_t}}{\sqrt{\alpha_t}}$ becomes smaller. For example, for $t = 200$, this value is equal to 0.002, and for $t = 800$ this value is close to 0.007. The term $x_t - \epsilon_\theta(x_t, t)$ is also small. This is because diffusion models mimic subtracting Gaussian noise, and thus we do not expect the difference between the input and output of the neural network, i.e., x_t, $\epsilon_\theta(x_t, t)$ to be in the order of a great two or three-digit value. While this is a behavior that we logically expect from the model, to confirm that this is actually true in practice, we generated 20 images with models trained on LSUNA Bedroom, LSUNA Church, and CelebA datasets and depicted the absolute mean value of the difference between the input and the output of the neural network, $|x_t - \epsilon_\theta(x_t, t)|$, across timesteps 1 to 1000, and the result, as shown in figure 7 was confirming.

From all the above points, we can conclude that the second part of equation 13 is very small, and thus, the difference between consecutive input values of our network is very small, meaning the term $\sqrt{\mu_t}(\frac{1}{\sqrt{\alpha_t}}(x_t - \epsilon_\theta^{(t)}(x_t)) - \epsilon_\theta(x_t, t))$ is very small and at most is equal to 0.005. Furthermore, again to make sure that the model behaves as expected, we plotted the absolute mean value of the second term of equation 13 which is approximately equal to $x_{t-1} - x_t$, for 20 images generated with datasets LSUN Bedroom, LSUN Church, and CelebA. As shown in figure 8 this value is in the order 10^{-6}. Thus, we have established that the difference between x_{t-1} and x_t, is very small. In fact, in the worst-case scenario, where $\frac{\sqrt{\mu_t}}{\sqrt{\alpha_t}}$ is equal to 0.01, and $x_t - \epsilon_\theta(x_t, t)$ is equal to 0.02, since μ_t is very small, the value of the difference between the input and output would be in the order of 0.0002. Interestingly, this worse case is not something that we expect to happen as the model is mimicking the Gaussian noise that was added at each timestep with a mean value of $\frac{\sqrt{\alpha_t}}{\sqrt{\alpha_{t-1}}}$. Thus, in reality, when the difference between the input and output of the network gets bigger, $\frac{\sqrt{\mu_t}}{\sqrt{\alpha_t}}$, gets smaller.

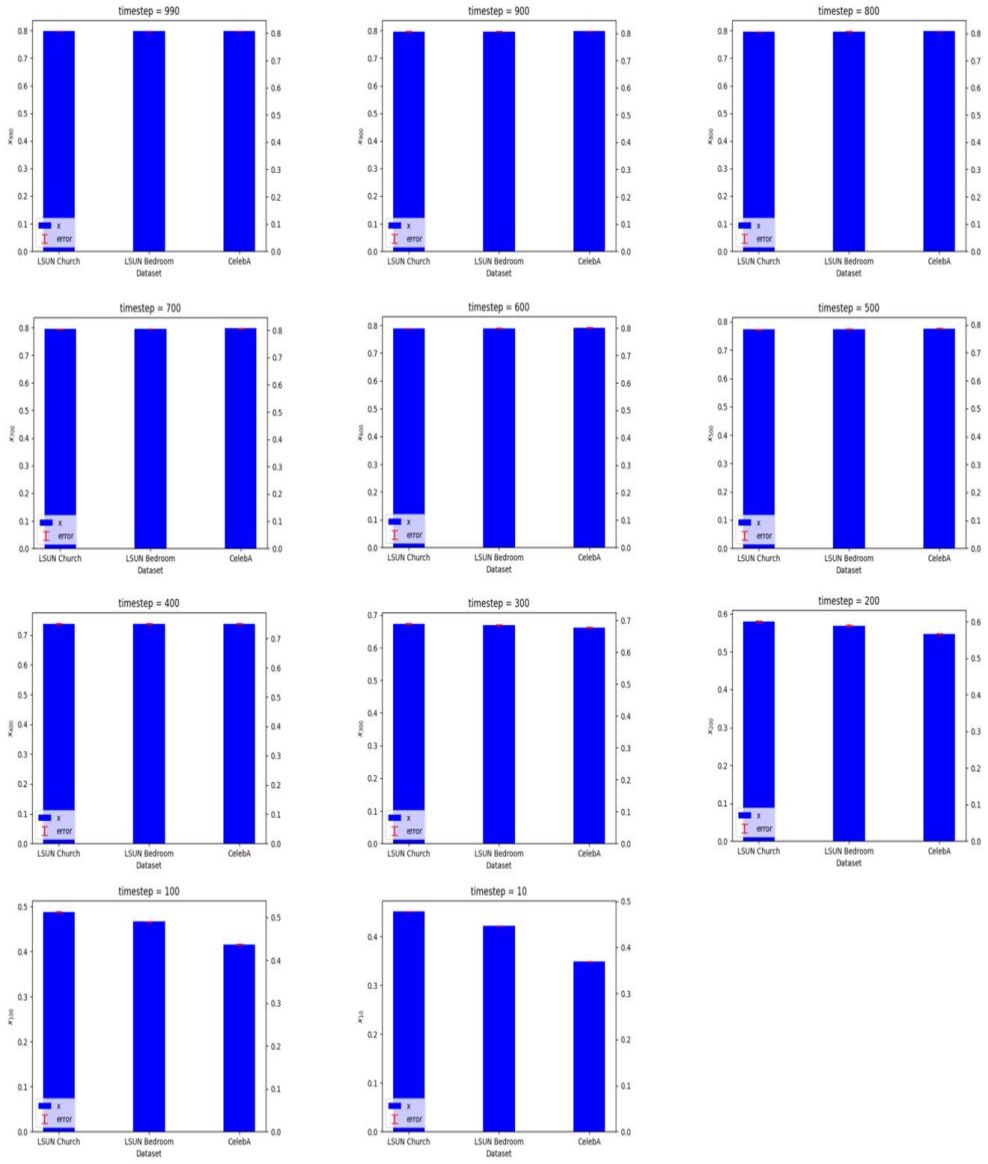

Figure 4: Error value introduced to the values of the equation 7 as a result of approximation made on equation 11 for different timesteps

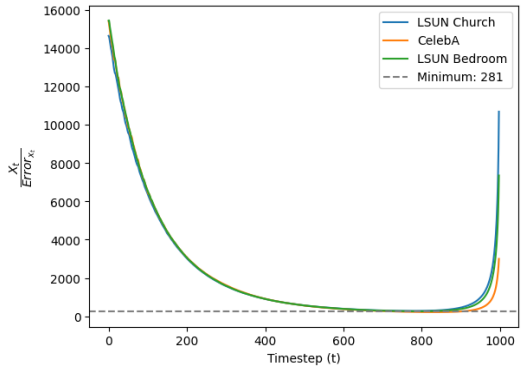

Figure 5: Ratio of the data to error introduced by equation 11 on equation 7 for different timesteps and different datasets

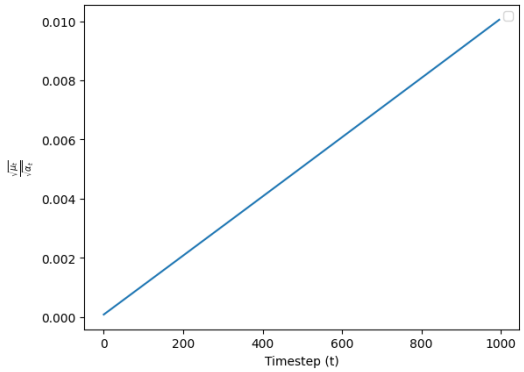

Figure 6: value of $\frac{\sqrt{\mu_t}}{\sqrt{\alpha_t}}$ over time

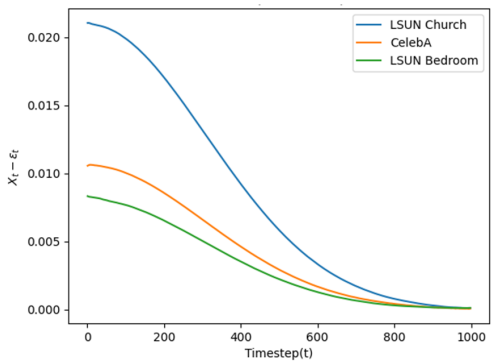

Figure 7: The absoulte value of the mean of the difference between the input and the output of the neural network over time

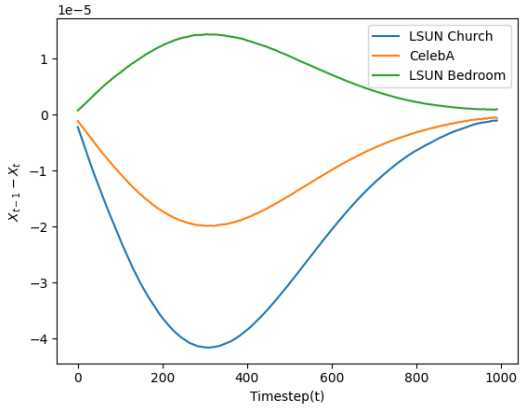

Figure 8: The absoulte value of the mean of the difference between two consecutive inputs of the neural network for different datasets

4.2 Output Approximation

In this section, we are going to prove that the difference between two consecutive outputs of the neural network is very similar over a short period of time. We are then going to use similarity to update the model's output without the neural network and just using the previous updates. For that, we first calculate the output of the model at timesteps t and $t-1$ as below:

$$\epsilon_\theta(x_t, t) \approx \frac{x_{t-1} - (\sqrt{\frac{\mu_t}{\alpha_t}} + 1)x_t}{\sqrt{\frac{\mu_t}{\alpha_t}} - 1} \tag{14}$$

$$\epsilon_\theta(x_{t-1}, t-1) = \frac{x_{t-2} - (\sqrt{\frac{\mu_t}{\alpha_t + \mu_t}} + 1)x_{t-1}}{\sqrt{\frac{\mu_t}{\alpha_t + \mu_t}} - 1} \tag{15}$$

Since we are approximating for the sake of simplicity, we put $\sqrt{\frac{\mu_t}{\alpha_t \pm \mu_t}} \approx \sqrt{\frac{\mu_t}{\alpha_t}}$ as their difference is in the order of 10^{-5}. It is important to note again that we are just trying to find a simple and approximate connection between $\epsilon_\theta(x_{t-1}, t-1)$ and its successors, which are known samples to us, and thus we can afford to make these kinds of approximations for the sake of simplicity. With that, we can write:

$$\epsilon_\theta(x_{t-1}, t-1) - \epsilon_\theta(x_t, t) = \frac{x_{t-2} - (\sqrt{\frac{\mu_t}{\alpha_t}} + 1)x_{t-1} - x_{t-1} - (\sqrt{\frac{\mu_t}{\alpha_t}} + 1)x_t}{\sqrt{\frac{\mu_t}{\alpha_t}} - 1} \tag{16}$$

Similarly, we can write the difference between the network's output at timesteps t and $t+1$ as below:

$$\epsilon_\theta(x_t, t) - \epsilon_\theta(x_{t+1,t+1}) = \frac{x_{t-1} - (\sqrt{\frac{\mu_t}{\alpha_t}} + 1)x_t - x_t - (\sqrt{\frac{\mu_t}{\alpha_t}} + 1)x_{t+1}}{\sqrt{\frac{\mu_t}{\alpha_t}} - 1} \tag{17}$$

Using the same logic that μ_t and $x_t - x_{t+1}$ are very small, we can establish that for the purpose of our estimation, the difference between $\epsilon_\theta(x_t, t) - \epsilon_\theta(x_{t+1}, t+1)$ and $\epsilon_\theta(x_t, t) - \epsilon_\theta(x_{t+1}, t+1)$ is very small. Thus, the changes that the neural network makes to its inputs are very similar. Consequently, it makes sense to instead of acquiring the sample at the next time step, using $\epsilon_\theta(x_{t-1}, t-1)$ estimate them by adding the difference that we expect the model to make on its input, to the last acquired output of the neural network. In other words, we can write:

$$\epsilon_\theta(x_{t-1}, t-1) \approx \epsilon_\theta(x_t, t) + (\epsilon_\theta(x_t, t) - \epsilon_\theta(x_{t+1}, t+1) \tag{18}$$

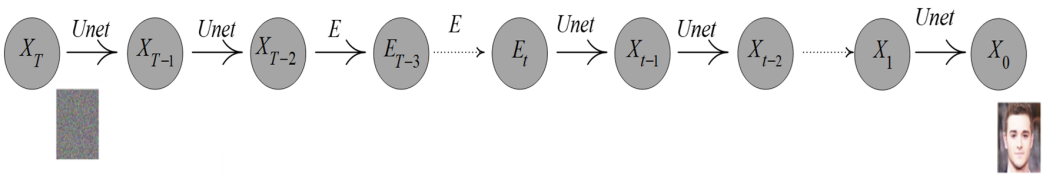

Figure 9: Uniform skipping scheduler.

and use equation 18, instead of the neural network to calculate $\epsilon_\theta^{(t-1)}(x_{t-1})$. This is fast to calculate, and its memory overhead is minimal as we only need to save the output of two previous timesteps in memory.

To sum up, in the previous section, we showed that the nature of solvers in diffusion models results in the difference between two consecutive inputs being small, which is also on par with the behavior that we expect from the model to gradually change the noise into data. Then we have shown that the difference between the consecutive outputs of the neural network is bounded by the input differences. Thus, the difference in the outputs is very small, meaning that we can use the difference between previous known outputs in order to obtain the unknown neural network's output that comes immediately after them.

In addition to the mathematical reasoning, we can support our estimation scheme by considering the practical implications of each step the network takes. Specifically, in DDMs, each step is designed to move the input closer to the desired data, which means that we can reasonably expect the value of a particular pixel to increase from timestep $t-1$ to $t-2$ if the neural network is increasing the value of that same pixel from timestep t to $t-1$. Therefore, this approximation aligns with our expectation that the network's outputs should gradually move away from the initial noise and toward the true data distribution over time. Finally, it is important to note that our scheme works as long as the difference between two consecutive inputs is small, which translates to every solver in which the difference between α values is small and also in which σ is set to 0.

5 Skip Scheduling

In this section, we present two skip scheduling algorithms tailored for optimal use with our estimation scheme. The first algorithm, known as the uniform skipping algorithm, offers a simple and straightforward approach to skipping. The second algorithm, the guided skipping algorithm, leverages guidance acquired from the estimation process. It efficiently determines the number of steps to skip and the number of steps to estimate for a given time and specific sample.

5.1 Uniform Skipping

The Uniform Skipping Algorithm is a straightforward technique that leverages the estimation approach described earlier. Our estimation method requires the difference between two consecutive outputs, which means that at any point in time, we need to take two consecutive steps with the neural network in order to make use of our estimation. Thus, in the uniform skipping algorithm, for some particular points in time, t, we take the steps t and $t+1$ with the neural network and then estimate the output of the neural network for the next n steps forward, $t-n$. This approach results in a speed-up of $\frac{n+2}{2}$. For instance, if we first take two steps using the neural network and then use equation 18 to estimate the output of the neural network for the next six steps, we will be able to accelerate sample generation by a factor of four. Figure 9 provides a visual representation of this approach.

A very important hyperparameter in the Uniform Skipping Algorithm is the number of steps to estimate, n. It is important to note that this approximation method may lead to an accumulative error over time. Consequently, selecting an appropriate value for n is essential to avoid distortions in the final sample.

Our extensive experimentation reveals that the optimal value for n is six. Figure 10 illustrates how the value of a single random pixel in a sample changes over time for the case where all the one thousand

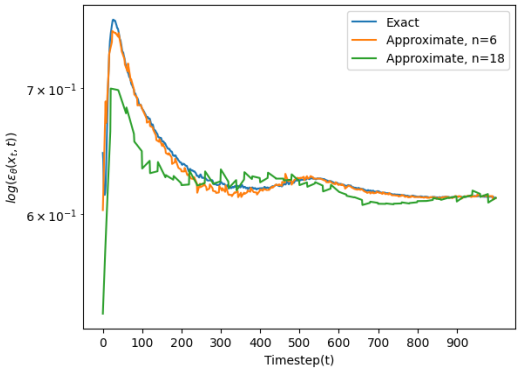

Figure 10: Changes in a value of a single pixel through time, without skipping and with skipping and estimation where $n = 6$ and $n = 18$

Figure 11: Samples generated without skipping, and skipping and estimation with n=6 and n=18

steps are taken and for the cases with estimation and n values of six and eighteen. With n set to six and using the estimation, the behavior of the pixel closely resembles that of the model when taking all steps. However, occasional sparks and sharp changes are observed at certain timesteps. We found that when n is small enough, the model can handle these sharp changes. However, as n increases, such as in the case of eighteen, the model struggles to recover from these sparks, leading to divergent behavior from the exact model.

It is crucial to emphasize that this behavior propagates throughout the model, as depicted in figure 11. When $n = 6$, the output images remain relatively similar, but with $n = 18$, the output images exhibit significant differences from the exact model.

In conclusion, the choice of n in the Uniform Skipping Algorithm is critical, and through our experiments, we determined that a value of six yields the best results, striking a balance between acceleration and fidelity. Selecting larger values of n may lead to notable discrepancies from the true model behavior and compromise the accuracy of the generated samples.

5.2 Guided Skipping

While the Uniform Skipping Algorithm proves effective in reducing the number of steps, it encounters limitations when attempting to compress the timestep count to extremely small values, such as 50. To address these limitations and achieve even better performance, we propose the "Guided Skipping Algorithm."

The Guided Skipping Algorithm is a straightforward yet powerful approach designed to accelerate DDMs. The primary objective of this algorithm is to dynamically adjust the number of steps based on the complexity of the sample being generated. By doing so, it aims to strike a balance between delivering high-quality results and minimizing computational overhead.

The Guided Skipping algorithm offers a more adaptive and refined solution compared to the Uniform Skipping algorithm, allowing it to handle a wider range of timesteps effectively. This adaptability makes it a promising choice for scenarios where high-quality output is critical, even with significantly reduced computational resources.

The algorithm starts the denoising process by taking two steps using the neural network. then it estimates a pre-determined number of steps, n, using equation 18 to estimate the output of the neural network. After executing these estimated steps, the algorithm then employs the neural network to calculate the actual output for the current step. A crucial step follows, where the actual output is compared to the estimated value that would have been used in its place. If the difference between these values is significant, it indicates that the step size was too large, and the algorithm responds by reducing the step size. Conversely, if the difference is not substantial, it suggests that the step size can be increased, allowing for a potentially larger number of steps to be estimated. This adaptive adjustment of the step size enables the algorithm to effectively optimize the denoising process and achieve a balance between accuracy and computational efficiency.

The determination of whether the actual neural network output and the estimated output are significantly different is a critical aspect of the algorithm. To address this, we devised a method that offers more informative insights than the standard difference metric. Instead, we introduced a percentage-based approach, focusing on the number of elements in the output that were updated in the wrong direction.

Specifically, we count the number of elements of the output, that would have increased if we used the estimated value but instead decreased with the neural network and those that would have decreased with the estimated value but instead increased with the neural network. This is fruitful because of the simple fact that if the estimated value updates the pixels in the same direction as the neural network, it will still be better than nothing, and it does not matter if we are not reaching the exact same output. However, going in the opposite direction for a huge fraction of the elements can result in samples with bad quality.

Determining an acceptable percentage of the elements that were updated in the wrong direction presents a challenge. This difficulty arises because even in the base case, where we take all the steps using the neural network, changes in the values of many elements of the neural network's output flip from one timestep to the next. In other words, the neural network may increase some values in one step and decrease most of them in the next. Additionally, the percentage of these elements that flip can vary from one sample to another and even from one model to another, making it impractical to find a universally suitable threshold.

Moreover, we strive to keep our scheduler as simple as possible and avoid imposing excessive computational overhead on the model. To address this challenge efficiently, we leveraged the observation of temporal locality between the number of elements in the output of the neural network that experience changes in their update direction across consecutive timesteps. This temporal locality is a result of input similarity in the neural network.

To establish a suitable threshold at each timestep, we set the threshold as the percentage of elements whose update direction flipped in the previous timestep (the output calculated at the preceding step), along with a small constant number. This approach incorporates both the model's behavior and the underlying sample characteristics in the decision-making process, allowing the algorithm to adapt its step size effectively. Importantly, this method is computationally inexpensive, ensuring efficient operation of the scheduler while maintaining a balance between sample fidelity and computational cost.

In determining how much to increase or decrease the skip size, we considered two critical factors. Firstly, the early steps in the denoising process are more susceptible to potential missteps. This is because, at the initial stages, the distance between the input and the final desired data is considerably larger compared to the later stages, which are primarily focused on refining the final results. As such, an erroneous step taken early on can lead the model astray, making it crucial to be cautious during these early steps.

Secondly, we need to ensure that the increment applied to the step size is greater than the decrement. This precaution is necessary to avoid the possibility of the step size oscillating between two values in a loop, which could impede the denoising process. Based on the aforementioned two considerations, we used equations 19 and 20 for updating the number of steps for the model to skip:

$$skip\_num = skip\_num + (4 + \left\lfloor \frac{1300 - t}{300} \right\rfloor) \tag{19}$$

$$skip\_num = skip\_num - (1 + \left\lfloor \frac{1600 - t}{600} \right\rfloor) \tag{20}$$

In the above equations, $skip_num$ represents the number of steps that the model skips during the denoising process. As mentioned earlier, we aim to ensure that the increment in the number of steps to skip is greater than the decrement, promoting cautious exploration in the early steps and more rapid progress towards the end. To achieve this, we incorporated a function with a smaller growth rate into the decreasing equation.

In equation 19, we update the number of steps to skip by adding $(4 + \lfloor \frac{1300-t}{300} \rfloor)$ to it. This equation increases the number of steps more rapidly as the denoising process progresses. The expression $\lfloor (\frac{1300-t}{300}) \rfloor$ starts from 1 at $t = 1000$ and approaches 4 as t goes to 0. This ensures that the rate of increase in the number of skipped steps grows as the algorithm approaches the final steps of denoising.

In equation 20, we update the number of steps to skip by subtracting $(1 + \lfloor \frac{1600-t}{600} \rfloor)$. The expression $\lfloor (\frac{1600-t}{600}) \rfloor$ starts from 1 at $t = 1000$ and approaches 2 as t goes to 0. This equation decreases the number of steps to skip more rapidly towards the end of the denoising process, ensuring that the algorithm becomes more cautious in the final steps.

Furthermore, to prevent an excessive decrease in the number of skipped steps, we only apply the decrement if the current number of steps to skip is greater than 8. This choice is based on the observation that up to 8 steps, our estimation is mostly accurate.

One critical issue with the guided skipping algorithm is the potential for some updates to lead in the wrong direction. While a model taking advantage of all 1000 training steps can tolerate occasional missteps without significant repercussions, our approach not only reduces the number of steps taken but also amplifies their impact on the output. Consequently, this can lead to certain samples being severely distorted, as illustrated in figure 11.

We observed that this problem was less pronounced when the number of steps to approximate was limited to six or fewer. However, this restriction also reduced the speed-up achieved, making it less desirable. To overcome this challenge and strike a better balance between accuracy and efficiency, we developed a hybrid algorithm. In this approach, we intelligently combined the benefits of guided skipping with estimation and traditional skipping. The key was to determine which steps to use estimation on and which steps to solely skip.

We found that when the updates were relatively small, even if they were in the wrong direction, their impact was minimal. Thus, we decided to set a limit on how much the approximation could change the mean value of the neural network output. Before reaching this limit, our algorithm used the approximated value. Afterward, until the next actual update with the neural network, we opted to skip the steps entirely. This approach improved stability and reduced the risk of severe distortions in the generated samples.

Furthermore, we observed that as the denoising process progressed and the model approached the final sample, the likelihood of significant missteps by the neural network diminished. To account for this behavior, we introduced an adaptive limit using equation 21. This allowed the algorithm to become less cautious as it neared the final denoised output.

By incorporating this hybrid approximation scheme and utilizing both guided skipping and traditional skipping strategically, we successfully resolved the problem of distorted samples. The algorithm achieved a more favorable balance between computational speed-up and high-fidelity denoising, resulting in improved sample quality and overall performance.

$$Limit = \left(\frac{1600 - t}{600} \right) * 0.001 \tag{21}$$

Our guided skipping approach offers two significant advantages over other methods. Firstly, it intelligently adapts the number of steps based on the complexity of the image being denoised. This

adaptability allows for more efficient computation without compromising the quality of the final output. By dynamically adjusting the skipping strategy, we strike a balance between accuracy and computational speed-up, making the denoising process more efficient.

Secondly, our guided skipping algorithm is computationally efficient and does not necessitate any pre-computation or additional overhead. Unlike some other approaches that may require complex pre-processing steps or computationally expensive evaluations, our method seamlessly integrates with the denoising process and operates in real-time, making it practical and easy to implement.

6 Experiments

This section presents the results of our study, demonstrating the capability of our approach to generate high-quality images at significantly faster speeds compared to DDPM. We begin by showing that our method can produce images almost indistinguishable from those generated by DDIM and PNDM with 1000 steps while achieving a four-fold increase in speed using our uniform scheduler. The evaluation of our estimation scheme involved experiments on four datasets: CelebA, LSUN Church, LSUN Bedroom, and CIFAR10.

Next, we assessed the performance of our guided skipping approach on three datasets: CelebA, LSUN Church, and LSUN Bedroom. For each dataset, we generated 25,000 images using our guided skipping algorithm. For our guided skipping algorithm, the number of steps required for generating a sample is different for each sample. Thus, to ensure a fair comparison, we calculated the number of steps taken by the neural network for each sample and utilized this value to generate samples using DDIM and PNDM schedulers. Subsequently, we measured the Frechet Inception Distance (FID) score, which quantifies the similarity between the generated samples and real images, for each of the three datasets.

Our results demonstrate that our scheme consistently produces images of good quality that are comparable to those generated by DDIM and PNDM with 1000 steps.

6.1 Quality

In this section, we will assess the quality of the photos generated using the Uniform Skipping and Guided Skipping algorithms. For the photos generated with the Uniform Skipping algorithm, we will evaluate their similarity to the photos generated with 1000 steps as a reference. As for the Guided Skipping algorithm, we will analyze two aspects of the generated samples' quality. First, we will assess their similarity to samples generated with 1000 steps, similar to the Uniform Skipping evaluation. Second, we will also evaluate their similarity to the real dataset, providing a more comprehensive measure of their performance.

6.1.1 Uniform skipping

To evaluate the quality of the generated images using the uniform skipping algorithm, we conducted experiments on the LSUN Church, LSUN Bedroom, CelebA, and CIFAR10 datasets and generated 500 photos under six different settings. In the first three settings, we built our estimation on top of the DDIM scheduler. With the first setting, we used the DDIM approach with 1000 steps as our baseline and compared the quality of the images generated in the other two settings against this baseline. In the second setting, we used the DDIM approach with 250 steps, and in the third setting, we employed our proposed uniform skipping approach with 250 steps using the neural network and 750 steps with the estimated values. The FID score for each setting is reported in Table 2 with the results showing a significantly lower FID score for the proposed uniform skipping approach compared to the DDIM approach. Moreover, to show the estimation overhead, we also reported the time for each of the cases. We then repeated the same experiment, but this time with the PNDM scheduler.

As shown in table 2 samples generated with 250 steps benefiting from estimation exhibit much more faithfulness to the original sample generated with 1000 steps than the samples generated without estimation.

Furthermore, as shown in figure 12 the difference between the image generated using the DDIM approach with 1000 steps and the image generated using our proposed approach is indistinguishable to the naked eye, indicating that our approach can achieve high-quality samples with four times

Figure 12: Compartion of the details with respect to the original image, between our approach and DDIM approach with the same number of steps.

fewer steps. However, the image generated using the DDIM approach with 250 steps exhibits visible differences compared to the baseline. Notably, as shown in figure 12 our approach outperforms DDIM in texture-rich areas and fine details of images when the number of steps is the same.

Dataset	Model	FID	PSNR	time
LSUN Church	DDIM 1000	-	-	658.13
	DDIM 250	10.49	35.02	164.54
	Proposed	2.74	45.86	165.23
LSUN Bedroom	DDIM 1000	-	-	672.25
	DDIM 250	13.44	35.02	168.10
	Proposed	4.35	41.91	168.8
CelebA	DDIM 1000	-	-	671.31
	DDIM 250	19.38	29.42	168.05
	Proposed	3.24	38.00	168.8
CIFAR10	DDIM 1000	-	-	77.28
	DDIM 250	25.45	27.91	19.17
	Proposed	4.67	50.29	19.84

Table 1: Comparison between the quality vs. time for 500 images generated from a model trained on LSUN church, LSUN Bedroom, CelebA, and CIFAR10 datasets with DDIM solver

6.1.2 guided skipping

To evaluate the guided skipping algorithm, also test it on top of three datasets: LSUN Church, LSUN Bedroom, and CelebA. For each of these datasets, we generated 25000 images once using our proposed algorithm and once using the same number that our algorithm took but without benefiting from our schemes. We then calculated the FID scores of the generated samples with respect to their corresponding real datasets. Results for implementing our algorithm on top of DDIm and PNDM schedulers are reported in table 3. As clear from table 3, our algorithm on top of the DDIM scheduler results in a better FID score in comparison with the real dataset and gains the best FID score for LSUN church and LSUN Bedroom datasets. However, for the CelebA dataset, it will report a higher FID, which means photos are less like the real dataset. This is because based on our observation, a higher step number does not always translate to better photo quality. This phenomenon is demonstrated in figure 16. As our algorithm is only capable of making the samples more similar to the photo samples generated with 1000 steps, we are unable to increase the output quality in those cases. Moreover, as illustrated in figures 13 and 14 photos generated by our approach in comparison to the ones generated by the DDIM and PNDM scheduler with approximately 50 steps, have better contrast and are more faithful in terms of content and colors to the photos generated by the DDIM and PNDM schedulers with 1000 steps.

Dataset	Model	FID	PSNR	time
LSUN Church	PNDM 1000	-	-	664.08
	PNDM 250	18.21	28.14	164.58
	Proposed	3.06	39.19	165.32
LSUN Bedroom	PNDM 1000	-	-	678.28
	DDIM 250	21.4	27.93	168.15
	Proposed	4.44	41.28	168.93
CelebA	PNDM 1000	-	-	678.56
	PNDM 250	36.21	29.57	168.00
	Proposed	13.58	35.11	168.74
CIFAR10	PNDM 1000	-	-	76.67
	PNDM 250	90.76	28.06	19.17
	Proposed	26.81	32.78	19.9

Table 2: Comparison between the quality vs time for 500 images generated from a model trained on LSUN church, LSUN Bedroom, CelebA and CIFAR10 datasets with PNDM solver

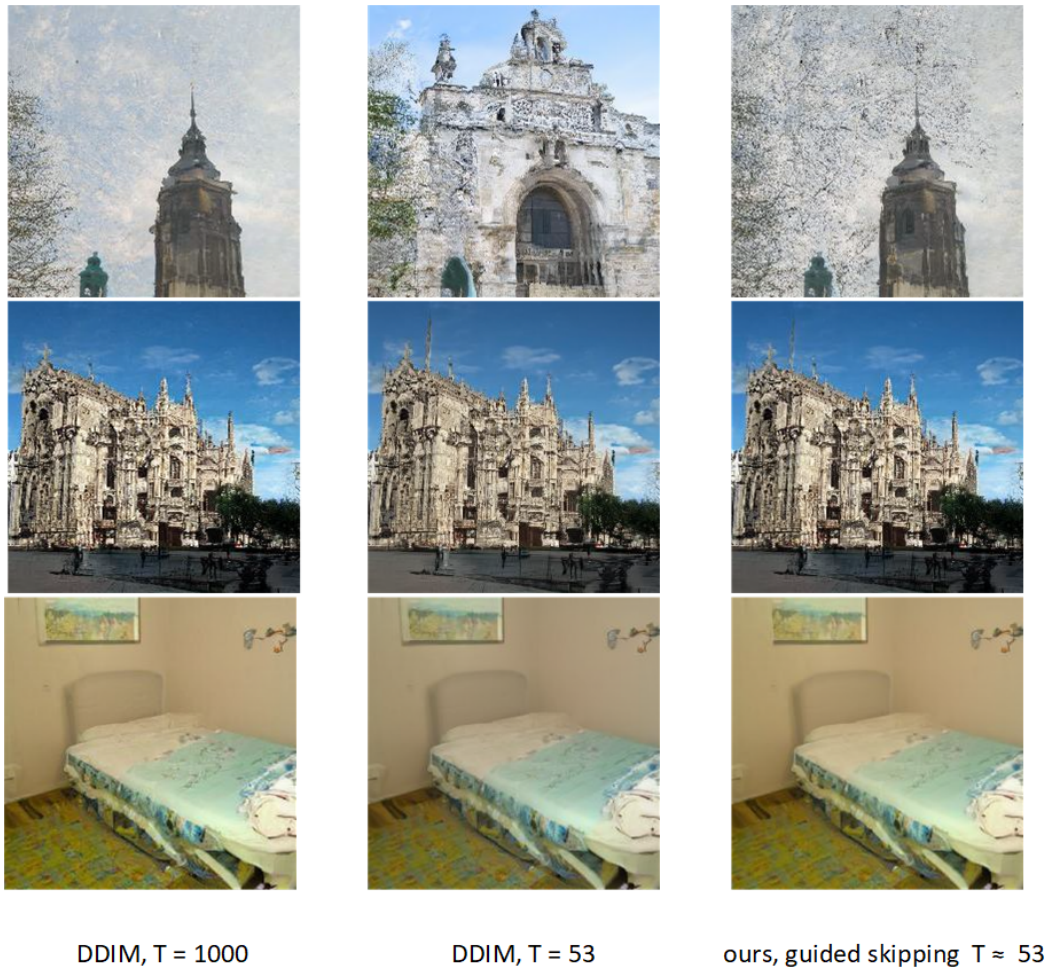

Figure 13: Samples generated with 1000 and 53 steps with DDIM scheduler and 53 steps using Estimdiff on top of the DDIM

Figure 14: Images generated with DDIM with 1000 steps, DDIM with 50 steps, and EstiDDIM with 50 steps, and images generated with PNDM with 1000 steps, PNDM with 50 steps, and EstiPNDM 50 steps.

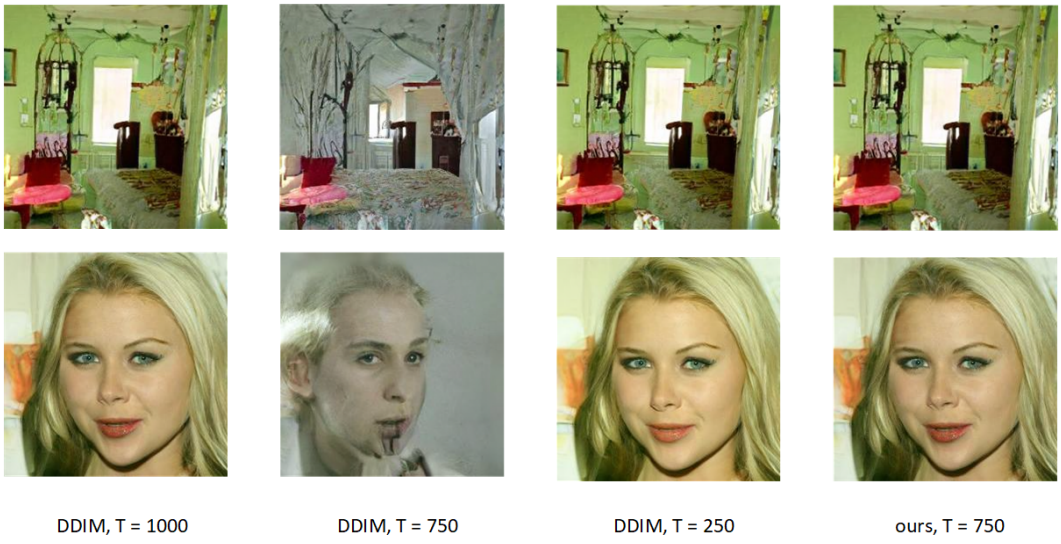

Figure 15: Samples generated with 1000 steps, 750 steps and 250 steps with and without estimation

Figure 16: Samples generated by 1000 steps and 50 steps with a model trained on CelebA dataset

Furthermore, to prove our claim that the photos generated with our approach exhibit more faithfulness to the photos generated with 1000 steps, we repeated the same experiment we did for the uniform skipping algorithm and reported the results in table 4.

Dataset	Model	FID	step number
LSUN Church	DDIM	10.95	43.3
	EstiDDIM	**10.31**	
	PNDM	11.24	42.8
	EstiPNDM	10.72	
LSUN Bedroom	DDIM	18.92	45.9
	EstiDDIM	**15.79**	
	PNDM	20.57	45.3
	EstiPNDM	25.93	
CelebA	DDIM	20.2	47.2
	EstiDDIM	22.70	
	PNDM	**19.4**	45.07
	EstiPNDM	25.25	

Table 3: Comparison between the FID score for 25000 images generated from a model trained on LSUN church, LSUN Bedroom, and CelebA datasets with DDIM and PNDM solvers

6.2 Consistency

Our experimental results demonstrate that the DDIM approach exhibits good performance for certain numbers of steps, but its behavior for other values remains unclear. For example, as shown in figure 15 DDIM is capable of generating high-quality samples when run for 250 steps, but when run for 750 steps, the resulting images lack coherence and fail to convey meaningful information. Nonetheless, there may be scenarios where users place a high priority on image quality and are willing to tolerate a moderate increase in computational cost to obtain a high-quality sample. Regrettably, the DDIM approach does not provide users with the flexibility to adjust sample quality levels. Conversely, our approach permits users to specify any desired number of steps and achieve high-quality samples. However, it should be noted that our experiments demonstrate that the perceptual discriminability between images generated with our approach and with 250 and 1000 steps is limited.

Dataset	Model	FID	PSNR
LSUN Church	DDIM	21.15	30.20
	EstiDDIM	12.44	33.35
	PNDM	19.70	30.09
	EstiPNDM	11.78	33.98
LSUN Bedroom	DDIM	23.7	31.59
	EstiDDIM	20.22	34.02
	PNDM	25.9	29.54
	EstiPNDM	17.24	35.21
CelebA	DDIM	53.49	29.12
	EstiDDIM	35.55	32.73
	PNDM	23.02	29.48
	EstiPNDM	20.69	33.10

Table 4: Comparison between the quality for 500 images generated from models trained on LSUN church, LSUN Bedroom, and CelebA datasets with DDIM and PNDM solvers

7 Conclusion

In conclusion, our paper presents two approaches for accelerating DDIMs while maintaining output quality. We have demonstrated that our uniform skipping approach can generate photos that are visually identical to those generated by DDIM and PNDM scheduler but four times faster. Moreover, our guided skipping approach can produce images that are very similar to the ones generated by the baseline DDIM and PNDM with 1000 steps, with fewer than 50 steps. Our results outperform the DDIM and PNDM approaches with the same amount of computation, showing the effectiveness of our proposed methods. The most noteworthy advantage of our approach is that it is effectively cost-free. It does not require retraining the neural network and avoiding the need for computationally expensive auxiliary models. Additionally, our method can seamlessly integrate with any scheduler for DDMs. Overall, we hope our approach for estimation and skipping can pave the way for future research in the acceleration of the generative models.

References

[1] Emanuele Aiello, Diego Valsesia, and Enrico Magli. Fast inference in denoising diffusion models via mmd finetuning, 2023.

[2] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans, 2021.

[3] Adam Block, Youssef Mroueh, Alexander Rakhlin, and Jerret Ross. Fast mixing of multi-scale langevin dynamics under the manifold hypothesis. *arXiv preprint arXiv:2006.11166*, 2020.

[4] 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*, pages 12413–12422, 2022.

[5] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. *arXiv preprint arXiv:2210.05475*, 2022.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[8] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23(47):1–33, 2022.

[9] Yuxin Hou, Hongxun Yao, Xiaoshuai Sun, and Haoran Li. Soul dancer: emotion-based human action generation. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(3s):1–19, 2020.

[10] 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*, pages 4401–4410, 2019.

[11] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[14] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[15] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[16] Zhaoyang Lyu, Xudong Xu, Ceyuan Yang, Dahua Lin, and Bo Dai. Accelerating diffusion models via early stop of the diffusion process. *arXiv preprint arXiv:2205.12524*, 2022.

[17] Neal Mangaokar, Jiameng Pu, Parantapa Bhattacharya, Chandan K. Reddy, and Bimal Viswanath. Jekyll: Attacking medical image diagnostics using deep generative models. In *2020 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 139–157, 2020.

[18] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models, 2023.

[19] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[20] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[21] Achraf Oussidi and Azeddine Elhassouny. Deep generative models: Survey. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8, 2018.

[22] Sung Woo Park, Dong Wook Shu, and Junseok Kwon. Generative adversarial networks for markovian temporal dynamics: Stochastic continuous data generation. In *International Conference on Machine Learning*, pages 8413–8421. PMLR, 2021.

[23] Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10619–10629, 2022.

[24] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[25] 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*, 2022.

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[27] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[28] Robin San-Roman, Eliya Nachmani, and Lior Wolf. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021.

[29] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[30] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

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

[32] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2020.

[33] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

[34] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models, 2021.

[35] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.

[36] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022.

[37] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22552–22562, 2023.

[38] Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders. *arXiv preprint arXiv:2202.09671*, 2022.