UDPM: Upsampling Diffusion Probabilistic Models

Shady Abu-Hussein
Department of Electrical Engineering
Tel Aviv University
shady.abh@gmail.com

Raja Giryes
Department of Electrical Engineering
Tel Aviv University
raja@tauex.tau.ac.il

Abstract

Denoising Diffusion Probabilistic Models (DDPM) have recently gained significant attention. DDPMs compose a Markovian process that begins in the data domain and gradually adds noise until reaching pure white noise. DDPMs generate high-quality samples from complex data distributions by defining an inverse process and training a deep neural network to learn this mapping. However, these models are inefficient because they require many diffusion steps to produce aesthetically pleasing samples. Additionally, unlike generative adversarial networks (GANs), the latent space of diffusion models is less interpretable. In this work, we propose to generalize the denoising diffusion process into an Upsampling Diffusion Probabilistic Model (UDPM). In the forward process, we reduce the latent variable dimension through downsampling, followed by the traditional noise perturbation. As a result, the reverse process gradually denoises and upsamples the latent variable to produce a sample from the data distribution. We formalize the Markovian diffusion processes of UDPM and demonstrate its generation capabilities on the popular FFHQ, AFHQv2, and CIFAR10 datasets. UDPM generates images with as few as three network evaluations, whose overall computational cost is less than a single DDPM or EDM step while achieving an FID score of 6.86. This surpasses current state-of-the-art efficient diffusion models that use a single denoising step for sampling. Additionally, UDPM offers an interpretable and interpolable latent space, which gives it an advantage over traditional DDPMs. Our code is available online: https://github.com/shadyabh/UDPM/

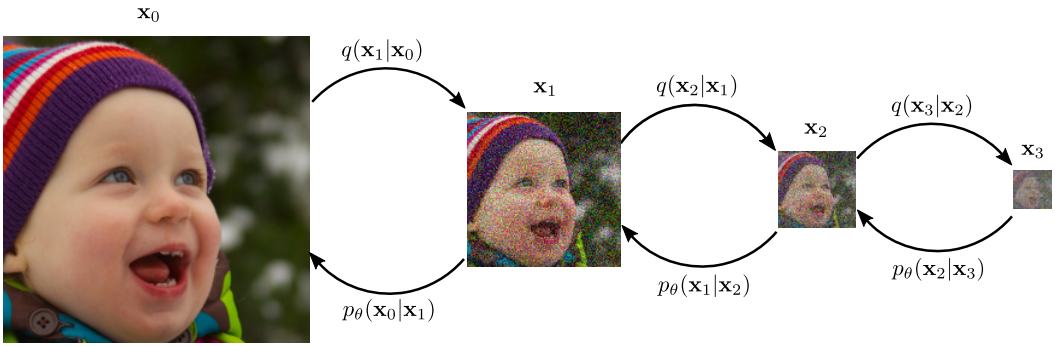


Figure 1: The Upsampling Diffusion Probabilistic Model (UDPM) scheme for 3 diffusion steps ($L = 3$). In addition to the gradual noise perturbation in traditional DDPMs, UDPM also downsamples the latent variables. Accordingly, in the reverse process, UDPM denoises and upsamples the latent variables to generate images from the data distribution.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0868

1 Introduction

In recent years, Denoising Diffusion Probabilistic Models (DDPMs) have become popular for image generation due to their ability to learn complex data distributions and generate high-fidelity images. These models work by starting with data samples and gradually adding noise through a Markovian process until pure white noise is reached. This process is known as the forward diffusion process, defined by the joint distribution $q(\mathbf{x}_{0:L})$. The reverse diffusion process, used for generating new samples, is defined by the learned reverse process $p_\theta(\mathbf{x}_{0:L})$ using a deep neural network. This methodology allows them to achieve impressive performance in learning data distributions and sampling from them.

Although DDPMs have shown impressive results in image generation, they possess some limitations. One major limitation is that they require a large number of denoising diffusion steps to produce aesthetically pleasing samples. This can be quite intensive computationally, which makes the sampling process slow and resource-intensive.

Additionally, the latent space of these models is not interpretable, which limits their utility for certain types of image generation tasks, such as video generation or animation, especially when used in an unconditional setting. The vast majority of works using diffusion models for editing rely on manipulating the CLIP [28] embeddings used with these models and not the latent space itself.

In this work, we propose a generalized scheme of DDPMs called the Upsampling Diffusion Probabilistic Model (UDPM). In addition to the gradual noise addition in the forward diffusion process, we downsample the latent diffusion variables to "dissolve" the data information spatially, as demonstrated in Figure 1. We thoroughly formulate the generalized model and derive the assumptions required for obtaining a viable scheme.

Using our approach, we can sample images from CIFAR10 [22], AFHQv2 [6], and FFHQ [19] datasets, with as few as 3 UDPM steps, where the cost of all three steps together is $\sim 30\%$ of a single regular diffusion step. This is less than two orders of magnitude compared to standard DDPMs: guided diffusion [9] typically requires 1000 iterations, denoising diffusion implicit models [35] require 250 iterations, stable diffusion [29] requires at least 50 iterations with an additional decoder, EDM [18] can reduce the number of steps to 39, and other recent works [24, 25, 10] can sample with 10-20 network evaluations, which is still considerably more than UDPM. Indeed, in [41], an integration of discriminative loss with diffusion models has shown great generation performance while requiring only 2 diffusion steps. Recent works [43, 26, 31] have demonstrated that sampling can be performed with as few as a single denoising diffusion step while maintaining competitive generation performance. Yet, UDPM achieves better generation quality than these works on the CIFAR10 dataset [22] while requiring a smaller computational cost ($1/3$ of a typical diffusion step).

Because UDPM gradually reduces the dimensions of the latent variables in each step, the size of the random noise added is considerably smaller. Specifically, all the dimensions of the latent variables together are smaller than the dimensions of the original images. As a result, UDPM is much more interpretable compared to conventional DDPMs. In our experiments, we show how one may manipulate the generated images by changing the latent variables, which is similar to what has been done in Generate Adversarial Networks (GANs) [19].

Our contributions may be summarized as follows: (i) A novel efficient diffusion model for image generation that achieves a significant improvement over current state-of-the-art methods by reducing the number and cost of diffusion steps required to generate high-quality images; (ii) achieving good interpretability and interpolability of the latent space.

2 Related Work

Diffusion models are latent variable models defined through a diffusion probabilistic model. On one side we have the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and on the other side, we have pure noise $\mathbf{x}_L \sim q(\mathbf{x}_L)$. Both are related to each other using a Markovian diffusion process, where the forward process is defined by the joint distribution $q(\mathbf{x}_{1:L}|\mathbf{x}_0)$ and the reverse process by $p(\mathbf{x}_{0:L-1}|\mathbf{x}_L)$. Using the Markov chain

https://doi.org/10.52202/079017-0868

Figure 2: Generated 64×64 images of AFHQv2 [6] with **FID=7.10142**, produced using unconditional UDPM with only 3 steps, which are equivalent to 0.3 of a single typical 64×64 diffusion step.

property, $q(\mathbf{x}_{1:L}|\mathbf{x}_0)$ and $p(\mathbf{x}_{0:L-1}|\mathbf{x}_L)$ can be expressed by

$$p(\mathbf{x}_{0:L-1}|\mathbf{x}_L) = \prod_{l=1}^{L} p(\mathbf{x}_{l-1}|\mathbf{x}_l), \tag{1}$$

and

$$q(\mathbf{x}_{1:L}|\mathbf{x}_0) = \prod_{l=1}^{L} q(\mathbf{x}_l|\mathbf{x}_{l-1}). \tag{2}$$

As stated in previous literature [14, 9, 34], the common approach is to assume that the Markov chain is constructed using normal distributions defined by

$$q(\mathbf{x}_l|\mathbf{x}_{l-1}) := \mathcal{N}(\sqrt{1-\beta_l}\mathbf{x}_{l-1}, \beta_l \mathbf{I}), \tag{3}$$

and

$$p(\mathbf{x}_{l-1}|\mathbf{x}_l) := \mathcal{N}(\mu_l(\mathbf{x}_l), \Sigma_l(\mathbf{x}_l)), \tag{4}$$

where $\beta_1, \beta_2, \ldots \beta_L$ are hyperparameters that control the noise levels of the diffusion process, while μ_l, Σ_l denote the mean and variance of the reverse process, respectively.

By learning the reverse process $p(\mathbf{x}_{l-1}|\mathbf{x}_l)$ using a deep neural network, one can generate samples from the data distribution by running the reverse process: starting from pure noise $\mathbf{x}_L \sim \mathcal{N}(0, \mathbf{I})$, then progressively predicting the next step of the reverse process using a network trained to predict $\mathbf{x}_{l-1}$ from $\mathbf{x}_l$, until reaching $\mathbf{x}_0$.

Over the past couple of years, this principle has shown marvelous performance in generating realistic-looking images [9, 29, 14, 34, 27, 20, 11, 32, 1]. However, as noted by [14], the number of diffusion steps L is required to be large for the model to produce pleasing-looking images.

Recently, many studies have utilized diffusion models for image manipulation and reconstruction tasks [40, 30], where a denoising network is trained to learn the prior distribution of the data. At test time, some conditioning mechanism is combined with the learned prior for solving highly challenging imaging tasks [3, 2, 7]. Note that our novel adaptive diffusion ingredient can be incorporated into any conditional sampling scheme that is based on diffusion models.

The works in [40, 30] addressed the problems of deblurring and super-resolution using diffusion models. These works try to deblur [40] or increase the resolution [30] of a blurry or low-res input image. Unlike our work, their goal is not image generation, but rather image reconstruction from a given degraded image. Therefore, their trained model is significantly different from ours.

Cold diffusion [4] performs diffusion steps by replacing the steps of noise addition with steps of blending with another image or other general steps. This approach differs significantly from ours, as their goal is to demonstrate that denoising can be replaced with other operations without developing the corresponding diffusion equations. In our case, we formally show what operations can be used without losing the Markovian property of the forward and reverse diffusion processes, and without omitting the noise component.

https://doi.org/10.52202/079017-0868

Soft diffusion [8] proposes to blur the signal before adding noise to it in the forward process. Then, for solving the reverse process they train a network to deblur and denoise the signal. However, the addition of the blur operation to the forward process prohibits explicit access to the reverse process and therefore relies solely on the network to predict the clean sharp sample. In contrast, in our method, the reverse process is explicitly accessible and can be sampled easily.

Another effort [12, 17] suggests replacing the denoising steps by learning the wavelet coefficients of the high frequencies. They show that this can reduce the number of diffusion steps. Our work differs from theirs in the fact that we rely on upsampling with an additive noise step. We also employ advanced loss functions and present state-of-the-art generation results. This is in addition to our interpretable latent space, a component missing from many of the recent diffusion models.

Many works tried to accelerate the sampling procedure of the denoising diffusion model [35, 18, 24, 25, 10]. However, they only focus on reducing the number of sampling steps, while ignoring the diffusion structure itself. By contrast, in this work we propose to degrade the signal not only over the noise domain but also in the spatial domain, thereby "dissolving" the signal much faster.

3 Method

Traditional denoising diffusion models assume that the probabilistic Markov process is defined by (3) and (4). These equations construct forward and backward processes that progress by adding and removing noise, respectively. In this work, we generalize this scheme by adding a degradation element to the forward process. Specifically, we downsample the spatial dimension of the latent variable and upsample it when reversing the process.

3.1 Upsampling Diffusion Probabilistic Model (UDPM)

We begin by redefining the marginal distributions $q(\mathbf{x}_l|\mathbf{x}_{l-1})$ and $p(\mathbf{x}_{l-1}|\mathbf{x}_l)$ of the forward and reverse processes:

$$q(\mathbf{x}_l|\mathbf{x}_{l-1}) := \mathcal{N}(\alpha_l \mathcal{H}\mathbf{x}_{l-1}, \sigma_l^2 \mathbf{I}), \tag{5}$$

and

$$p(\mathbf{x}_{l-1}|\mathbf{x}_l) := \mathcal{N}(\mu(\mathbf{x}_l; l), \Sigma_l), \tag{6}$$

where in contrast to previous diffusion models that used $\mathcal{H} = I$, in this work we define the operator $\mathcal{H}$ as a downsampling operator, defined by applying a blur filter $\mathcal{W}$ followed by subsampling with stride γ. As a result, the forward diffusion process decreases the variables' dimensions in addition to the increased noise levels.

In diffusion models, the goal is to match the joint distributions $p_\theta(\mathbf{x}_{1:L}|\mathbf{x}_0)$ (learned) and $q(\mathbf{x}_{1:L}|\mathbf{x}_0)$ under some statistical distance. One particular choice is the Kullback-Leibler (KL) divergence, which we adopt here. Formally,

$$D_{\mathrm{KL}}(q(\mathbf{x}_{1:L}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:L}|\mathbf{x}_0)) := \mathbf{E}_q\left[\log \frac{q(\mathbf{x}_{1:L}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{1:L}|\mathbf{x}_0)}\right] = \log p_\theta(\mathbf{x}_0) \underbrace{-\mathbf{E}_q\left[\frac{p_\theta(\mathbf{x}_{0:L})}{q(\mathbf{x}_{1:L}|\mathbf{x}_0)}\right]}_{\text{ELBO}}. \tag{7}$$

Thus, one can minimize the KL-divergence between $p_\theta(\mathbf{x}_{1:L}|\mathbf{x}_0)$ and $q(\mathbf{x}_{1:L}|\mathbf{x}_0)$ by minimizing the Evidence Lower Bound (ELBO). As we show in Appendix B.1, this is equivalent to

$$\begin{aligned}\mathbf{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:L})}{q(\mathbf{x}_{1:L}|\mathbf{x}_0)}\right] &= \mathbf{E}_q[D_{\mathrm{KL}}(p(\mathbf{x}_L)||q(\mathbf{x}_L|\mathbf{x}_0)) \\ &+ \sum_{l=2}^{L} D_{\mathrm{KL}}(p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l)||q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0) - \log p_\theta(\mathbf{x}_1|\mathbf{x}_0)].\end{aligned} \tag{8}$$

The right-hand side of (8) can be then minimized stochastically w.r.t. θ using gradient descent, where at each step a random l is chosen and a single term of (8) is optimized.

In order to be able to use (8) for training $p_\theta(\cdot)$, one needs explicit access to $q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)$, for which we need to obtain $q(\mathbf{x}_l|\mathbf{x}_0)$ first. Then, using Bayes' theorem, we can derive $q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)$. To do so, we first present Lemma 1 (the proof is in Appendix B.2)

Figure 3: Generated 64×64 images of FFHQ with **FID=7.41065**, produced using unconditional UDPM with only 3 steps, which are equivalent to 0.3 of a single typical 64×64 diffusion step.

Lemma 1. *Let* $\mathbf{e} \overset{iid}{\sim} \mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^N$ *and* $\mathcal{H} = \mathcal{S}_\gamma \mathcal{W}$*, where* $\mathcal{S}_\gamma$ *is a subsampling operator with stride* γ *and* $\mathcal{W}$ *is a blur operator with blur kernel* $\mathbf{w}$*. Then, if the support of* $\mathbf{w}$ *is at most* γ*, we have* $\mathcal{H}\mathbf{e} \overset{iid}{\sim} \mathcal{N}(0, \|\mathbf{w}\|_2^2 \mathbf{I})$.

If Lemma 1 holds, then by assuming that $\|\mathbf{w}\|_2^2 = 1$, we get the following result (see Appendix B.3)

$$q(\mathbf{x}_l|\mathbf{x}_0) = \mathcal{N}(\bar{\alpha}_l \mathcal{H}^l \mathbf{x}_0, \tilde{\sigma}_l^2 \mathbf{I}) \text{ where } \bar{\alpha}_l = \prod_{k=0}^{l} \alpha_k, \text{ and } \tilde{\sigma}_l = \bar{\alpha}_l^2 \sum_{k=1}^{l} \frac{\sigma_k^2}{\bar{\alpha}_l^2}. \tag{9}$$

Using (9), we can obtain $\mathbf{x}_t$ from $\mathbf{x}_0$ simply by applying $\mathcal{H}$ l-times on $\mathbf{x}_0$, followed by the addition of white Gaussian noise with standard deviation $\tilde{\sigma}_l$.

Given $q(\mathbf{x}_l|\mathbf{x}_0)$, we can use Bayes' theorem and utilize the Markov chain property to get

$$q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0) = \frac{q(\mathbf{x}_l|\mathbf{x}_{l-1}) q(\mathbf{x}_{l-1}|\mathbf{x}_0)}{q(\mathbf{x}_l|\mathbf{x}_0)},$$

which, as shown in Appendix B.4, is of the following form

$$q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0) = \mathcal{N}(\mu(\mathbf{x}_l, \mathbf{x}_0, l), \Sigma_l), \tag{10}$$

where

$$\Sigma_l = \left(\frac{\alpha_l^2}{\sigma_l^2} \mathcal{H}^T \mathcal{H} + \frac{1}{\tilde{\sigma}_{l-1}} \mathbf{I} \right)^{-1}, \tag{11}$$

and

$$\mu(\mathbf{x}_l, \mathbf{x}_0, l) = \Sigma_l \left(\frac{\alpha_l}{\sigma_l^2} \mathcal{H}^T \mathbf{x}_l + \frac{\bar{\alpha}_{l-1}}{\tilde{\sigma}_{l-1}^2} \mathcal{H}^{l-1} \mathbf{x}_0 \right). \tag{12}$$

Although expression (11) seems to be implacable, in practice it can be implemented efficiently using the Discrete Fourier Transform and the poly-phase filtering identity used in [15, 5], where $\mathcal{H}^T \mathcal{H}$ is equivalent to a convolution between $\mathbf{w}$ and its flipped version, followed by subsampling with stride γ. More details are provided in Appendix B.5.

Following (10), the true posterior $q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)$ in this new setup is a Gaussian distribution with parameters $(\mu(\mathbf{x}_l, \mathbf{x}_0, l), \Sigma_l)$. Therefore, one may assume that $p_\theta(\cdot)$ is also Gaussian with parameters (μ_θ, Σ_l), where μ_θ is parameterized by a deep neural network with learned parameters θ.

For normal distributions, a single term of (8) is equivalent to

$$\begin{aligned} \ell^{(l)} &= D_{\text{KL}}(p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l) || q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0) \\ &= C_l + \frac{1}{2} (\mu_\theta - \mu_l)^T \Sigma_l^{-1} (\mu_\theta - \mu_l), \end{aligned} \tag{13}$$

where C_l is a constant value independent of θ.

https://doi.org/10.52202/079017-0868

Algorithm 1 UDPM training algorithm

Require: $f_\theta(\cdot), L, q(\mathbf{x}), D_\phi(\cdot)$
while Not converged **do**
  $\mathbf{x}_0 \sim q(\mathbf{x})$
  $l \in \{1, 2, \dots, L\}$
  $\mathbf{e} \sim \mathcal{N}(0, I)$
  $\mathbf{x}_l = \bar{\alpha}_l \mathcal{H}^l \mathbf{x}_0 + \tilde{\sigma}_l \mathbf{e}$
  $\ell = \lambda_{\text{fid}}^{(l)} \ell_{\text{simple}} + \lambda_{\text{per}}^{(l)} \ell_{\text{per}} + \lambda_{\text{adv}}^{(l)} \ell_{\text{adv}}$
  ADAM step on θ
  Adversarial ADAM step on ϕ
end while
return $f_\theta(\cdot)$

Algorithm 2 UDPM sampling algorithm

Require: $f_\theta(\cdot), L$
$\mathbf{x}_L \sim \mathcal{N}(0, I)$
for all $l = L, \dots, 1$ **do**
  $\Sigma = \left(\frac{\alpha_l^2}{\sigma_l^2} \mathcal{H}^T \mathcal{H} + \frac{1}{\tilde{\sigma}_{l-1}^2} \mathbf{I} \right)^{-1}$
  $\mu_\theta = \Sigma \left[\frac{\alpha_l}{\sigma_l^2} \mathcal{H}^T \mathbf{x}_l + \frac{\bar{\alpha}_{l-1}}{\tilde{\sigma}_{l-1}^2} f_\theta^{(l)}(\mathbf{x}_l) \right]$
  $\mathbf{x}_{l-1} \sim \mathcal{N}(\mu_\theta, \Sigma)$
end for
return $\mathbf{x}_0$

From our experiments and following [9, 14], training p_θ to predict μ directly leads to worse results. Therefore, we train the network to predict the second term in (12), i.e. to estimate $\mathcal{H}^{l-1}\mathbf{x}_0$ from $\mathbf{x}_l$. As a result, minimizing (13) can be simplified to minimizing the following term

$$\tilde{\ell}_{\text{simple}}^{(l)} = (f_\theta(\mathbf{x}_l) - \mathcal{H}^{l-1}\mathbf{x}_0)^T \Sigma_l^{-1} (f_\theta(\mathbf{x}_l) - \mathcal{H}^{l-1}\mathbf{x}_0), \tag{14}$$

where $f_\theta(\cdot)$ is a deep neural network that upsamples its input by a scale factor of γ. By the definition of Σ_l, it is easy to show that it is a diagonal positive matrix, and therefore it can be dropped. As a result, one may simplify the objective to the following term

$$\ell_{\text{simple}}^{(l)} = \| f_\theta(\mathbf{x}_l) - \mathcal{H}^{l-1}\mathbf{x}_0 \|_2^2. \tag{15}$$

Unlike denoising diffusion models, UDPM tackles a super-resolution task at each reverse step. Consequently, relying solely on ℓ_{simple} for training $f_\theta(\cdot)$ produces softer images, as shown in Figure 7 and discussed in the ablation studies section 5. To address this, we propose to incorporate two additional regularization terms, following [38]. The first term is a perceptual loss [42], denoted by ℓ_{per}, which aligns the VGG features of $\mathcal{H}^{l-1}\mathbf{x}_0$ and $f_\theta(\mathbf{x}_l)$. The second term is an adversarial loss denoted as ℓ_{adv}, which is similar to the one used in [38, 23]. Both are used to ensure sharp detailed results. Overall, $f_\theta(\cdot)$ is trained using the following objective function

$$\ell = \lambda_{\text{fid}}^{(l)} \ell_{\text{simple}} + \lambda_{\text{per}}^{(l)} \ell_{\text{per}} + \lambda_{\text{adv}}^{(l)} \ell_{\text{adv}}. \tag{16}$$

Algorithm 1 and Figure 4 provide an overview of the UDPM training scheme.

3.2 Image Generation using UDPM

The UDPM scheme presented in the previous section introduces a new approach for capturing the true implicit data distribution $q(\mathbf{x}_0)$ of a given dataset. Given that we trained UDPM, the remaining question is how it can be utilized for sampling from $q_(\mathbf{x}_0)$.

Given a deep neural network f_θ trained to predict $\mathcal{H}^{l-1}\mathbf{x}_0$ from $\mathbf{x}_l$, we start with a pure Gaussian noise sample $\mathbf{x}_L \sim \mathcal{N}(0, I)$. Subsequently, by substituting $f_\theta(\mathbf{x}_l)$ into $\mathcal{H}^{l-1}\mathbf{x}_0$ in (12), we get an estimate of μ_L. Then, the next reverse diffusion step $\mathbf{x}_{L-1}$ can be obtained by sampling from $\mathcal{N}(\mu_L, \Sigma_L)$. By iteratively repeating these steps L times, we can acquire a sample $\mathbf{x}_0 \sim p_\theta(\mathbf{x}_0)$, as outlined in Algorithm 2.

Note that sampling the posterior requires parameterizing $\mathcal{N}(\mu_l, \Sigma_l)$ in the form of

$$\mathbf{x}_{l-1} = \mu_l + \Sigma_l^{\frac{1}{2}} \mathbf{e},$$

where $\mathbf{e} \sim \mathcal{N}(0, I)$ has the same dimensions as $\mathbf{x}_{l-1}$. This requires having access to $\Sigma_l^{\frac{1}{2}}$. Due to the structure of Σ_l, it is possible to apply it on $\mathbf{e}$ efficiently, as shown in Appendix B.6.

4 Experiments

In this section, we present the evaluation of UDPM under multiple scenarios. We tested our method on CIFAR10 [22], FFHQ [19], and AFHQv2 [6] datasets. Here we focus on the qualitative performance

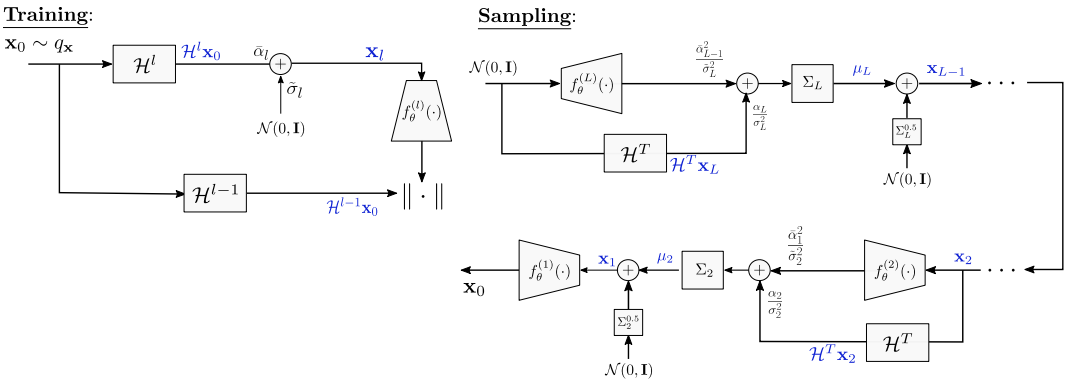


Figure 4: The training and sampling procedures of UDPM. During the *training* phase, an image $\mathbf{x}_0$ is randomly selected from the dataset. It is then degraded using (9) to obtain a downsampled noisy version $\mathbf{x}_l$, which is then plugged into $f_\theta^{(l)}(\cdot)$, that is trained to predict $\mathcal{H}^{l-1}\mathbf{x}_0$. In the *sampling* phase, we start from pure noise $\mathbf{x}_L \sim \mathcal{N}(0, \mathbf{I})$. This noise is passed through the network $f_\theta^{(L)}(\cdot)$ to estimate $\mathcal{H}^{L-1}\mathbf{x}_0$, used to compute μ_L through (12), with Σ_L obtained from (11). Afterwards, $\mathbf{x}_{L-1}$ is drawn from $\mathcal{N}(\mu_L, \Sigma_L)$ using the technique described in Appendix B.6. By repeating this procedure for L iterations, the final sample $\mathbf{x}_0$ is obtained.

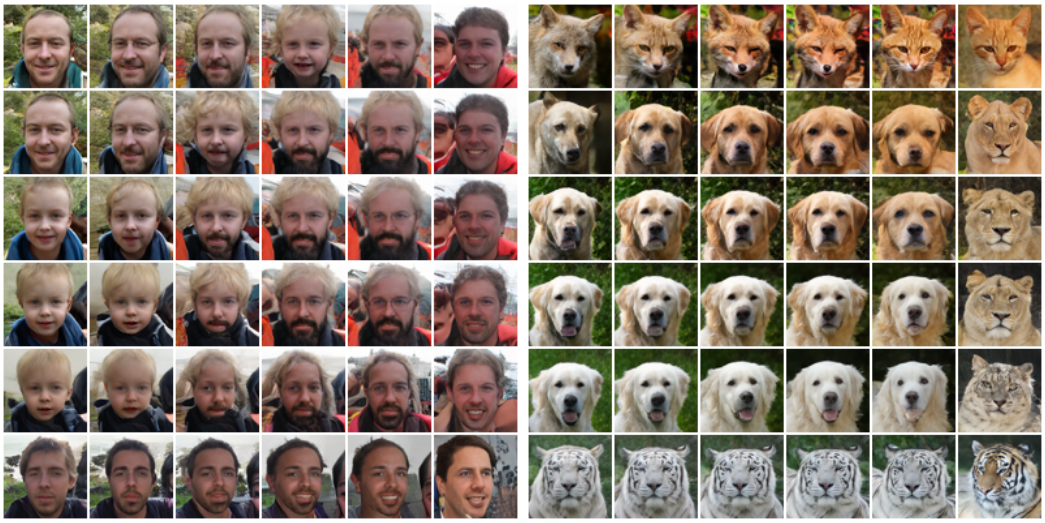


Figure 5: Latent space interpolation for 64×64 generated images. The four corner images are interpolated by a weighted mixture of their latent noises, such that the other images are "in-between" images from the latent perspective, similar to what has been done in GANs [19].

of UDPM and demonstrate its interpolatable latent space. In the appendix, we provide additional quantitative and qualitative results.

We set $L = 3$ and fix $\gamma = 2$ for all datasets. We also use a uniform box filter of size 2×2 as the downsampling kernel $\mathbf{w}$ as it satisfies the condition in Lemma 1. We then normalize it w.r.t. its norm $\|\mathbf{w}\|$ and use it to construct the downsampling operator $\mathcal{H}$. Unlike previous diffusion approaches that are limited to analytically defined noise schedulers due to the large diffusion steps number, UDPM allows the noise scheduler to be fine-tuned manually by setting only 6 values ($\{\alpha_l\}_{l=1}^L$ and $\{\sigma_l\}_{l=1}^L$). In our tests we set $\{\alpha_l\}_{l=1}^3 = \{0.5, 0.2, 10^{-3}\}$ and $\{\sigma_l\}_{l=0}^3 = \{0.1, 0.2, 0.3\}$ for all datasets.

https://doi.org/10.52202/079017-0868

Figure 6: Latent space perturbation for 64×64 generated images. The original image is on the left. To its right we present images that were generated by adding a small noise to the latent noise from diffusion step l. As can be seen, the initial diffusion step ($l = 1$) controls the fine details of the image, while the final diffusion step ($l = 3$) changes the semantics of the image.

We use the same UNet architecture proposed by [37] and utilized in [18] for all diffusion steps (the specific implementation details are presented in Table 3). We increase the number of output channels of the network by a factor of γ^2 and use the depth-to-space layer [33] to rearrange the output pixels to the desired dimension, which is equivalent to a scale-up by a factor of γ. For all datasets, we train the network for 600K training steps using ADAM [21] with learning rate and batch size set to 10^{-4} and 64, respectively. We save the model weights every 10K training steps and pick the model with the best Fréchet inception distance (FID) [13]. For stabilizing the training we use exponential moving averaging (EMA) with the dampening parameter set to 0.9999. We also set $\lambda_{\text{fid}} = (1, 1, 0)$, $\lambda_{\text{per}} = (4, 4, 0)$, and $\lambda_{\text{adv}} = (0.2, 0.5, 1)$. Particularly, we found that setting $\lambda_{\text{fid}} = \lambda_{\text{per}} = 0$ for $l = 3$ achieves the best results with minimal mode-collapse. For the discriminator network, we use the discriminator architecture used in [41], which is a variant of the original discriminator network proposed by [19]. We train all models using a single NVIDIA RTX A6000 GPU.

4.1 Unconditional Generation

In the unconditional scheme, we evaluate UDPM on FFHQ and AFHQv2 datasets, where each dataset contains $50K$ and $14K$ images, respectively. We resize the images to 64×64 and train the UNet using Algorithm 1.

The number of sampling steps required by UDPM is two orders of magnitude smaller than the original denoising diffusion models [14, 34], one order of magnitude less than [35, 18], and up-to 4 times smaller than many recent sophisticated samplers [24, 25, 10] designed for accelerating diffusion generation. Particularly, since the size of latent variables increases progressively when sampling, UDPM requires less total computations than a single denoising step used in traditional denoising diffusion, as detailed in 4.3. While reducing the computational costs, UDPM retains great image generation quality, as can be seen in Figures 3 and 2.

In addition to the decreased number of sampling steps, UDPM has much smaller latent dimensions. This property allows us to smoothly interpolate the latent space, similar to what has been done in GANs [19]. Figures 5, 10, and 11 show the results, where the corner images are randomly generated and the in-between images are generated by averaging the noises used to generate the corner images:

$$\mathbf{e}_l(i,j) = \eta_i(\delta_j \mathbf{e}_1^l + \sqrt{1-\delta_j^2}\mathbf{e}_2^l) + \sqrt{1-\eta_i^2}(\delta_j \mathbf{e}_3^l + \sqrt{1-\delta_j^2}\mathbf{e}_4^l),$$

where $\mathbf{e}_1^l, \mathbf{e}_2^l, \mathbf{e}_3^l, \mathbf{e}_4^l$ are the noises at diffusion step l used to generate the 4 corner images, (i, j) are the indices of the interpolated image in the figure, and (η_i, δ_j) are the interpolation coefficients that lie in the range $[0, 1]$. In the supplementary material, we investigate other interpolation regimes, such as latent variable swapping between two generated images (see Figure 13).

Another benefit of the smaller latent dimensions is that it allows better interpretability of the generative model. While in previous diffusion approaches one needed to add a conditioning mechanism to the network to be able to control the generation, in UDPM, one may control the generation by modifying only the noise maps of the diffusion, as can be seen in Figures 6 and 12.

	steps	FID
DDIM [35]	10/5	13.36/93.51
DPM-Solver [25]	10/5	6.96/288.99
EDM [18]	35/5	1.79/35.54
GENIE [10]	5	11.20
DEIS [16]	5	15.37
GGDM [39]	5	13.77
DDGAN [41]	2	4.08
TDPM [43]	1	8.91
CT [36]	1	8.70
UDPM (ours)	**<1**	**6.86**

Table 1: FID scores on the CIFAR10 dataset [22]. UDPM uses 3 steps, which are equivalent in terms of complexity to 0.3 of a single denoising step used in typical diffusion models like DDPM or EDM.

4.2 Class Conditional Generation

Similarly to previous works, we also propose a conditional generation scheme for UDPM, where we use the label encoding block used in [18] to control the generation class. We evaluate UDPM on the popular CIFAR10 [22] dataset, which has 50K 32×32 images of 10 different classes.

Figure 14 shows the generation results of UDPM on CIFAR10. We also compare the performance of UDPM empirically to many other state-of-the-art non-distillation diffusion methods in Table 1, where we examine the FID results on 50K images generated from the 10 classes equally. For the baseline methods, we show the original results presented in their papers and the results when the number of steps is reduced to 5 diffusion steps. As can be seen, when the number of steps is limited to 5, UDPM outperforms all the methods, while also requiring considerably less computation; specifically, $1/3$ of a single typical diffusion step.

4.3 Runtime

Unlike typical denoising diffusion models, UDPM decreases the dimensions of the latent variables when proceeding with the diffusion process. Thus, the computations required to run the reverse diffusion process for sampling are significantly less extensive. Specifically, sampling a 64×64 image using a single denoising diffusion step requires 40.62 GFLOPS, while UDPM samples images with the same network structure using 13.35 GFLOPs, which is equivalent to 30% of the computations. The reason is that in its 3 steps, the input dimensions of the UDPM network are 8×8, 16×16, and 32×32

When comparing the runtimes, UDPM can sample 64×64 images from FFHQ and AFHQv2 datasets at the rate of 765.21 FPS (frames per second) when benchmarked on NVIDIA RTX A6000 GPU, while a single denoising diffusion step can be run at the rate of 255 FPS using a very similar network.

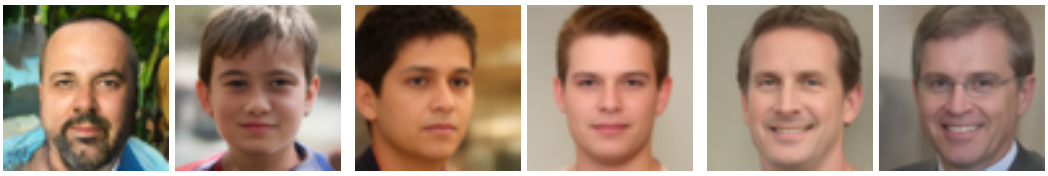
5 Ablation studies

To further understand the diffusion process captured by UDPM, we present two ablation studies in which we examine the effect of each diffusion step on the generation results and demonstrate the effectiveness of the perceptual loss terms used to train the network.

PCA of the generations. Given the noise maps of a generated image, we marginally change the single diffusion step and then perform Principal Component Analysis (PCA) on the generated images, we then examine the top principal components to understand what features are captured in each diffusion step. In Figure 8 we present the top 8 principal components when the l-th noise map is using small noise, as can be seen, the initial diffusion step controls the fine details while the final step controls the semantics and the overall structure of the image. Similarly, in Figure 9 we show the top 8 principal components of the generated images when a single noise map is replaced entirely.

Perceptual loss. From our experiments we saw that using $\|\cdot\|_p$ for training $f_\theta(\cdot)$ leads to very soft results. Hence we use the more sophisticated loss term in (16), which promotes perception over distortion. For evaluating the contribution of each loss term, we examine three different networks:

https://doi.org/10.52202/079017-0868



$\ell_1 + \ell_{\text{per}} + \ell_{\text{adv}}$ | $\ell_1 + \ell_{\text{per}}$ | ℓ_1

FID=7.41 | FID=32.76 | FID=72.91

Figure 7: Visual comparison of the loss terms effect on the FFHQ64 dataset generation results.

The first one is trained using ℓ_{simple}, while the second has ℓ_{per} in addition to ℓ_{simple} in its objective, and finally the network trained using (16). We show qualitative and quantitative comparisons in Figure 7; the advantage of the perceptual terms can be seen clearly, where the FID score and the visual comparison show that the results of using the full loss are significantly better.

6 Conclusion

We have proposed a new diffusion-based generative model called the Upsampling Diffusion Probabilistic Model (UPDM). Our approach reduces the number of diffusion steps required to produce high-quality images, which makes it significantly more efficient than previous solutions.

We have demonstrated the effectiveness of our approach on three different datasets: CIFAR10, FFHQ, and AFHQv2. It is capable of producing high-quality images with only 3 diffusion steps, whose computational cost is less than for one step of the original diffusion models, while significantly outperforming current state-of-the-art methods dedicated to diffusion sampling acceleration.

Furthermore, we have shown that our interpolatable latent space has potential for further exploration, particularly in the realm of image editing. Based on the initial results shown in the paper, it contains semantic directions, such as making people smile or changing their age as shown in Figure 6. Future research may explore using UDPM to perform editing operations similar to those performed in styleGAN while maintaining better generation capabilities and favorable diffusion properties.

7 Limitation

While our approach produces impressive generative performance, it has some limitations. One limitation of our work is the evaluation of relatively small datasets, such as CIFAR10, FFHQ, and AFHQv2, which is a consequence of our limited computational resources. This restricted us from evaluating our approach on larger and more diverse datasets. Future work could overcome this limitation by leveraging a more powerful computational infrastructure to conduct experiments on larger datasets, providing a more comprehensive validation of the model's capabilities. Additionally, while UDPM offers improved interpretability over "standard" denoising diffusion models, it still falls short compared to the interpretability shown for GANs. To address this, further research could explore enhancing the latent space structure of UDPMs to make it more interpretable, perhaps by incorporating techniques from GANs or further analyzing the currently generated latent space.

8 Acknowledgement

This research was partially supported by the Israeli Innovation Authority through the meta consortium. The authors are grateful to the Neubauer family for their kind support.

References

[1] Shady Abu-Hussein, Tom Tirer, and Raja Giryes. Adir: Adaptive diffusion for image reconstruction. 2022.

[2] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *arXiv preprint arXiv:2206.02779*, 2022.

[3] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18208–18218, 2022.

[4] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S. Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. *arXiv:2208.09392*, 2022.

[5] Stanley H Chan, Xiran Wang, and Omar A Elgendy. Plug-and-play admm for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, 2016.

[6] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

[7] Hyungjin Chung, Eun Sun Lee, and Jong Chul Ye. Mr image denoising and super-resolution using regularized reverse diffusion. *arXiv preprint arXiv:2203.12621*, 2022.

[8] Giannis Daras, Mauricio Delbracio, Hossein Talebi, Alexandros G Dimakis, and Peyman Milanfar. Soft diffusion: Score matching for general corruptions. *arXiv preprint arXiv:2209.05442*, 2022.

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

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

[11] Giorgio Giannone, Didrik Nielsen, and Ole Winther. Few-shot diffusion models. *10.48550/ARXIV.2205.15463*, 2022.

[12] Florentin Guth, Simon Coste, Valentin De Bortoli, and Stephane Mallat. Wavelet score-based generative modeling, 2022.

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

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

[15] Shady Abu Hussein, Tom Tirer, and Raja Giryes. Correction filter for single image super-resolution: Robustifying off-the-shelf deep super-resolvers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1428–1437, 2020.

[16] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021.

[17] Zahra Kadkhodaie, Florentin Guth, Stéphane Mallat, and Eero P Simoncelli. Learning multi-scale local conditional probability models of images, 2023.

[18] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

 https://doi.org/10.52202/079017-0868

[19] 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.

[20] 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. *arXiv preprint arXiv:2210.09276*, 2022.

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

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[24] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022.

[25] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

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

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

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[29] 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.

[30] 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.

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

[32] Shelly Sheynin, Oron Ashual, Adam Polyak, Uriel Singer, Oran Gafni, Eliya Nachmani, and Yaniv Taigman. Knn-diffusion: Image generation via large-scale retrieval. *arXiv:2204.02849*, 2022.

[33] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

[34] 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.

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2022.

[36] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[37] 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.

[38] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1905–1914, 2021.

[39] Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2021.

[40] Jay Whang, Mauricio Delbracio, Hossein Talebi, Chitwan Saharia, Alexandros G Dimakis, and Peyman Milanfar. Deblurring via stochastic refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16293–16303, 2022.

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

[42] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[43] Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Truncated diffusion probabilistic models. *arXiv preprint arXiv:2202.09671*, 1(3.1):2, 2022.

 https://doi.org/10.52202/079017-0868

UDPM: Upsampling Diffusion Probabilistic Models – Supplementary Material

Here we provide our social impact statement, extended derivations of our UDPM model, ablation studies on the latent space, and additional generation results.

A Social Impact Statement

The development of Upsampling Diffusion Probabilistic Models (UDPM) represents a significant advancement in the field of image generation, offering the ability to produce high-quality images with fewer computational resources. However, as with any powerful technology, there are potential risks and ethical considerations to address. One major concern is the potential misuse of this technology for creating realistic but deceptive images, such as deepfakes, which can be used to spread misinformation or for malicious purposes. Additionally, the ability to generate high-quality synthetic images raises issues of copyright and ownership, potentially impacting artists and content creators whose work might be replicated or modified without their consent.

To mitigate these risks, it is essential to implement safeguards and establish ethical guidelines for the use of UDPM and similar technologies. This can include developing robust detection methods for synthetic content, ensuring transparency in the creation and distribution of AI-generated media, and promoting responsible use among developers and end-users. Collaborating with policymakers to create regulations that address the misuse of generative models can also help in preventing harm. Furthermore, fostering an open dialogue within the AI research community about the ethical implications and potential societal impacts of such technologies will be crucial in ensuring that advancements in this field are aligned with broader societal values and public interest.

B Extended Derivations

B.1 Evidence Lower Bound

From the Evidence Lower Bound (ELBO) we have $\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_{0:L})}{q(\mathbf{x}_{1:L}|\mathbf{x}_0)}\right]$, hence

$$
\begin{aligned}
\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_{0:L})}{q(\mathbf{x}_{1:L}|\mathbf{x}_0)}\right] &= \mathbb{E}_q\left[-\log p(\mathbf{x}_L) - \sum_{l=1}^{L} \log \frac{p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l)}{q(\mathbf{x}_l|\mathbf{x}_{l-1})}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_L) - \sum_{l=2}^{L} \log \frac{p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l)}{q(\mathbf{x}_l|\mathbf{x}_{l-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&\overset{(\star)}{=} \mathbb{E}_q\left[-\log p(\mathbf{x}_L) - \sum_{l=2}^{L} \log \frac{p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l) q(\mathbf{x}_{l-1}|\mathbf{x}_0)}{q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0) q(\mathbf{x}_l|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_L) - \sum_{l=2}^{L} \log \frac{p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l)}{q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)} - \sum_{l=2}^{L} \log q(\mathbf{x}_{l-1}|\mathbf{x}_0) + \sum_{l=2}^{L} \log q(\mathbf{x}_l|\mathbf{x}_0)\right. \\
&\quad \left. - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_L) - \sum_{l=2}^{L} \log \frac{p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l)}{q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)} - \cancel{\log q(\mathbf{x}_1|\mathbf{x}_0)} + \log q(\mathbf{x}_L|\mathbf{x}_0)\right. \\
&\quad \left. - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{\cancel{q(\mathbf{x}_1|\mathbf{x}_0)}}\right] \\
&= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_L)}{q(\mathbf{x}_L|\mathbf{x}_0)} - \sum_{l=2}^{L} \log \frac{p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l)}{q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= D_{\mathrm{KL}}(p(\mathbf{x}_L)||q(\mathbf{x}_L|\mathbf{x}_0)) + \sum_{l=2}^{L} D_{\mathrm{KL}}(p_\theta(\mathbf{x}_{l-1}|\mathbf{x}_l)||q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)) - \mathbb{E}_q\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right],
\end{aligned}
$$

where in $(\star)$ Bayes was used, particularly

$$q(\mathbf{x}_l|\mathbf{x}_{l-1}) = q(\mathbf{x}_l|\mathbf{x}_{l-1}, \mathbf{x}_0) = \frac{q(\mathbf{x}_{l-1}, \mathbf{x}_l|\mathbf{x}_0)}{q(\mathbf{x}_{l-1}|\mathbf{x}_0)} = \frac{q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0)q(\mathbf{x}_l|\mathbf{x}_0)}{q(\mathbf{x}_{l-1}|\mathbf{x}_0)}.$$

B.2 Lemma 1

Proof. The characteristic function of a Normal vector $\mathbf{e} \sim \mathcal{N}(\mu, \Sigma)$ is in the form

$$\phi_{\mathbf{e}}(\mathbf{t}) = \mathbb{E}[\exp(i\mathbf{t}^T\mathbf{e})] = \exp(i\mathbf{t}^T\mu - \frac{1}{2}\mathbf{t}^T\Sigma\mathbf{t}), \tag{17}$$

when $\mu = 0$ and $\Sigma = \mathbf{I}$ we get

$$\phi_{\mathbf{e}}(\mathbf{t}) = \exp(-\frac{1}{2}\mathbf{t}^T\mathbf{t}).$$

To prove the first claim of the lemma, it is sufficient to prove that $\mathcal{H}\mathbf{e}$ has a characteristic function of the form (17). Denote the transpose operator of $\mathcal{H}$ by $\mathcal{H}^T$, which is defined as zero padding operator followed by applying a flipped version the kernel of $\mathcal{H}$, denoted by $\mathbf{w}$. We have

$$\begin{aligned}
\phi_{\mathcal{H}\mathbf{e}}(\mathbf{t}) &= \mathbb{E}[\exp(i\mathbf{t}^T(\mathcal{H}\mathbf{e}))] = \exp(i(\mathcal{H}^T\mathbf{t})^T\mathbf{e}) = \exp(i(\mathcal{H}^T\mathbf{t})^T\mu - \frac{1}{2}(\mathcal{H}^T\mathbf{t})^T\Sigma(\mathcal{H}^T\mathbf{t})) \\
&= \exp(i\mathbf{t}^T\mathcal{H}\mu - \frac{1}{2}\mathbf{t}^T\mathcal{H}\Sigma\mathcal{H}^T\mathbf{t}) \overset{\{\mu=0,\Sigma=I\}}{=} \exp(-\frac{1}{2}\mathbf{t}^T\mathcal{H}\mathcal{H}^T\mathbf{t}), \\
&\Rightarrow \mathcal{H}\mathbf{e} \sim \mathcal{N}(0, \mathcal{H}\mathcal{H}^T).
\end{aligned}$$

All that remains is to express $\mathcal{H}\mathcal{H}^T$ under the assumption on the support of the downsampling kernel . The operator $\mathcal{H}$ is defined as applying blur kernel $\mathbf{w} = [w_{-\lfloor\gamma/2\rfloor+1}, \ldots, w_0, \ldots w_{\lfloor\gamma/2\rfloor}]$ followed by subsampling with stride γ, which can be represented in matrix form by $\mathcal{H} = \mathcal{S}_\gamma\mathcal{W}$, where $\mathcal{S}_\gamma \in \mathbb{R}^{M\times N}$ and $\mathcal{W} \in \mathbb{R}^{N\times N}$. Specifically

$$\mathcal{S}_\gamma = \begin{pmatrix} 1\,0\,\ldots\,0 & 0\,\ldots\,0 & \ldots & 0 \\ \underbrace{0\,\ldots\,0}_{\gamma} & 1\,0\,\ldots\,0 & \ldots & 0 \\ \underbrace{0\,\ldots\,0}_{\gamma} & \underbrace{0\,\ldots\,0}_{\gamma}\,1 & \ldots & 0 \\ & \vdots & \vdots & \vdots \end{pmatrix},$$

$$\mathcal{W}_\gamma = \begin{pmatrix} w_0 & w_1 & \ldots & w_{\lfloor\gamma/2\rfloor} & 0 & \ldots & w_{-\lfloor\gamma/2\rfloor+1} & w_{-\lfloor\gamma/2\rfloor+2} & \ldots & w_{-1} \\ w_{-1} & w_0 & \ldots & w_{\lfloor\gamma/2\rfloor} & 0 & \ldots & 0 & w_{-\lfloor\gamma/2\rfloor+1} & \ldots & w_{-2} \\ \vdots & & & \vdots & & & \vdots & & & \vdots \end{pmatrix},$$

Therefore we have

$$\mathcal{S}_\gamma\mathcal{W} = \begin{pmatrix} w_0 & \ldots & w_{\lfloor\gamma/2\rfloor} & 0 & \ldots & 0 & \ldots & 0 & w_{-\lfloor\gamma/2\rfloor+1} & \ldots \\ 0 & \ldots & 0 & w_{-\lfloor\gamma/2\rfloor+1} & \ldots & w_{\lfloor\gamma/2\rfloor} & \ldots & 0 & 0 & \ldots \\ \vdots & & & \vdots & & & \vdots & & & \vdots \end{pmatrix}, \tag{18}$$

as can be observed from (18), the rows of $\mathcal{S}_\gamma\mathcal{W}$ do not intersect with each other, as a result we get

$$\mathcal{H}\mathcal{H}^T = \mathcal{S}_\gamma\mathcal{W}(\mathcal{S}_\gamma\mathcal{W})^T = \mathcal{S}_\gamma\mathcal{W}\mathcal{W}^T\mathcal{S}_\gamma^T = \begin{pmatrix} \|\mathbf{w}\|_2^2 & 0 & 0 & \ldots & 0 \\ 0 & \|\mathbf{w}\|_2^2 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & 0 & \ldots & \|\mathbf{w}\|_2^2 \end{pmatrix} = \|\mathbf{w}\|_2^2\mathbf{I}. \tag{19}$$

□

https://doi.org/10.52202/079017-0868

B.3 Efficient representation of the forward process

Similar to earlier works, for efficient training, one would like to sample $\mathbf{x}_l$ directly using $\mathbf{x}_0$. Which is viable when assuming that the support of $\mathbf{w}$ is at most γ and $\|w\|_2^2 = 1$. Formally

$$\begin{aligned}\mathbf{x}_l &= \alpha_l \mathcal{H}\mathbf{x}_{l-1} + \sigma_l \mathbf{e}_1 = \alpha_l \mathcal{H}(\alpha_{l-1}\mathcal{H}\mathbf{x}_{l-2} + \sigma_{l-1}\mathbf{e}_2) + \sigma_l \mathbf{e}_1 \\ &= \alpha_l\alpha_{l-1}\mathcal{H}^2\mathbf{x}_{l-2} + \alpha_l\sigma_{l-1}\mathcal{H}\mathbf{e}_2 + \sigma_l\mathbf{e}_1 \\ &\overset{\text{Lemma (1)}}{=} \alpha_l\alpha_{l-1}\mathcal{H}^2\mathbf{x}_{l-2} + \alpha_l\sigma_{l-1}\tilde{\mathbf{e}}_2 + \sigma_l\mathbf{e}_1,\end{aligned}$$

where $\mathbf{e}_1, \mathbf{e}_2, \tilde{\mathbf{e}}_2 \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{I})$. By definition, $\mathbf{e}_1$ and $\tilde{\mathbf{e}}_2$ are independent identically distributed vectors, therefore one may write

$$\mathbf{x}_l = \alpha_l\alpha_{l-1}\mathcal{H}^2\mathbf{x}_{l-2} + \sqrt{\alpha_l^2\sigma_{l-1}^2 + \sigma_l^2}\mathbf{e},$$

repeating the opening $l-2$ times leads to

$$\begin{aligned}\mathbf{x}_l &= \bar{\alpha}_l\mathcal{H}^l\mathbf{x}_0 + \sqrt{\alpha_l^2\alpha_{l-1}^2...\alpha_2^2\sigma_1^2 + \alpha_l^2\alpha_{l-1}^2...\alpha_3^2\sigma_2^2 + ... + \alpha_l^2\sigma_{l-1}^2 + \sigma_l^2}\mathbf{e} \\ &= \bar{\alpha}_l\mathcal{H}^l\mathbf{x}_0 + \sqrt{\frac{\bar{\alpha}_l^2\sigma_1^2}{\bar{\alpha}_1} + \frac{\bar{\alpha}_l^2\sigma_2^2}{\bar{\alpha}_2} + ... + \frac{\bar{\alpha}_l^2\sigma_1^2}{\bar{\alpha}_l}}\mathbf{e} = \bar{\alpha}_l\mathcal{H}^l\mathbf{x}_0 + \bar{\alpha}_l\sqrt{\frac{\sigma_1^2}{\bar{\alpha}_1} + \frac{\sigma_2^2}{\bar{\alpha}_2} + ... + \frac{\sigma_1^2}{\bar{\alpha}_l}}\mathbf{e} \\ &= \bar{\alpha}_l\mathcal{H}^l\mathbf{x}_0 + \tilde{\sigma}_l\mathbf{e},\end{aligned}$$

where $\bar{\alpha}_l = \prod_{k=1}^{L}\alpha_k$, $\tilde{\sigma}_l^2 = \bar{\alpha}_l^2\sum_{k=1}^{l}\frac{\sigma_k^2}{\bar{\alpha}_k^2}$, and $\mathbf{e} \sim \mathcal{N}(0, \mathbf{I})$.

B.4 Derivation of the posterior

Using Bayes theorem, we have for $l > 1$

$$\begin{aligned}q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0) &= \frac{q(\mathbf{x}_{l-1}, \mathbf{x}_l|\mathbf{x}_0)}{q(\mathbf{x}_l|\mathbf{x}_0)} = \frac{q(\mathbf{x}_l|\mathbf{x}_{l-1}, \cancel{\mathbf{x}_0})q(\mathbf{x}_{l-1}|\mathbf{x}_0)}{q(\mathbf{x}_l|\mathbf{x}_0)} \overset{\text{Markov}}{=} \frac{q(\mathbf{x}_l|\mathbf{x}_{l-1})q(\mathbf{x}_{l-1}|\mathbf{x}_0)}{q(\mathbf{x}_l|\mathbf{x}_0)} \\ &\overset{(\star\star)}{\propto} \exp\left(-\frac{1}{2\sigma_l^2}\|\mathbf{x}_l - \alpha_l\mathcal{H}\mathbf{x}_{l-1}\|_2^2 - \frac{1}{2\tilde{\sigma}_{l-1}^2}\|\mathbf{x}_{l-1} - \bar{\alpha}_{l-1}\mathcal{H}^{l-1}\mathbf{x}_0\|_2^2\right) \\ &= \exp(-\frac{1}{2\sigma_l^2}(\mathbf{x}_l^T\mathbf{x}_l - 2\alpha_l\mathbf{x}_{l-1}^T\mathcal{H}^T\mathbf{x}_l + \alpha_l^2\mathbf{x}_{l-1}^T\mathcal{H}^T\mathcal{H}\mathbf{x}_{l-1}) \\ &- \frac{1}{2\tilde{\sigma}_{l-1}^2}(\mathbf{x}_{l-1}^T\mathbf{x}_{l-1} - 2\bar{\alpha}_{l-1}\mathbf{x}_{l-1}^T\mathcal{H}^{l-1}\mathbf{x}_0 + \bar{\alpha}_{l-1}^2\mathbf{x}_0^T(\mathcal{H}^{l-1})^T\mathcal{H}^{l-1}\mathbf{x}_0)) \\ &\overset{(\star\star)}{\propto} \exp\left\{-\frac{1}{2}\mathbf{x}_{l-1}^T\left(\frac{\alpha_l^2}{\sigma_l^2}\mathcal{H}^T\mathcal{H} + \frac{1}{\tilde{\sigma}_{l-1}^2}\mathbf{I}\right)\mathbf{x}_{l-1} + \mathbf{x}_{l-1}^T\left(\frac{\alpha_l}{\tilde{\sigma}_l^2}\mathcal{H}^T\mathbf{x}_l + \frac{\bar{\alpha}_{l-1}}{\tilde{\sigma}_{l-1}^2}\mathcal{H}^{l-1}\mathbf{x}_0\right)\right\},\end{aligned}$$

which is a Normal distribution form w.r.t. the random vector $\mathbf{x}_{l-1}$. Note that in $(\star\star)$ the proportional equivalence is with relation to $\mathbf{x}_{l-1}$. Let $q(\mathbf{x}_{l-1}|\mathbf{x}_l, \mathbf{x}_0) = \mathcal{N}(\mu, \Sigma)$, then one may write

$$\Sigma^{-1} = \frac{\alpha_l^2}{\sigma_l^2}\mathcal{H}^T\mathcal{H} + \frac{1}{\tilde{\sigma}_{l-1}^2}\mathbf{I}, \tag{20}$$

and

$$\Sigma^{-1}\mu = \frac{\alpha_l}{\sigma_l^2}\mathcal{H}^T\mathbf{x}_l + \frac{\bar{\alpha}_{l-1}}{\tilde{\sigma}_{l-1}^2}\mathcal{H}^{l-1}\mathbf{x}_0. \tag{21}$$

B.5 Mean and variance of the posterior

The terms in (20) and (21) seem hard to evaluate, however, in the following we present an efficient way to compute them. By definition $\mathcal{H}$ is structured from a circular convolution with a blur filter followed by a subsampling with stride α. Therefore, one may use the poly-phase identity used in [15, 5], which states that $\mathcal{H}^T\mathcal{H}$ in this case is a circular convolution between the blur kernel $\mathbf{w}$ and its flipped version $\tilde{\mathbf{w}}$ followed by subsampling with factor α, formally

$$\mathbf{h} = (\mathbf{w} \circledast flip(\mathbf{w})) \downarrow_\alpha,$$

therefore, one may write $\mathcal{H}^T\mathcal{H}$ in the following form

$$\mathcal{H}^T\mathcal{H} = \mathcal{F}^\star\Lambda_{\mathbf{h}}\mathcal{F}, \tag{22}$$

where $\mathcal{F}$ is the Discrete Fourier Transform (DFT), $\mathcal{F}^\star$ is the inverse DFT, and $\Lambda_{\mathbf{h}}$ is a diagonal operator representing the DFT transform of $\mathbf{h}$. By plugging (22) into (20) we get

$$\Sigma^{-1} = \frac{\alpha_l^2}{\sigma_l^2}\mathcal{F}^\star\Lambda_{\mathbf{h}}\mathcal{F} + \frac{1}{\tilde{\sigma}_{l-1}^2}\mathbf{I} = \frac{\alpha_l^2}{\sigma_l^2}\mathcal{F}^\star\Lambda_{\mathbf{h}}\mathcal{F} + \frac{1}{\tilde{\sigma}_{l-1}^2}\mathcal{F}^\star\mathcal{F} = \mathcal{F}^\star\left(\frac{\alpha_l^2}{\sigma_l^2}\Lambda_{\mathbf{h}} + \frac{1}{\tilde{\sigma}_{l-1}^2}\mathbf{I}\right)\mathcal{F},$$

equivalently

$$\Sigma = \mathcal{F}^\star\left(\frac{\alpha_l^2}{\sigma_l^2}\Lambda_{\mathbf{h}} + \frac{1}{\tilde{\sigma}_{l-1}^2}\mathbf{I}\right)^{-1}\mathcal{F}. \tag{23}$$

Therefore, as can be observed from (23), applying Σ is equivalent to applying the inverse of the filter $\left(\frac{\alpha_l^2}{\sigma_l^2}\Lambda_{\mathbf{h}} + \frac{1}{\tilde{\sigma}_{l-1}^2}\mathbf{I}\right)$, which can be performed efficiently using Fast Fourier Transform (FFT). Finally we have

$$\mu = \Sigma\left(\frac{\alpha_l}{\sigma_l^2}\mathcal{H}^T\mathbf{x}_l + \frac{\bar{\alpha}_{l-1}}{\tilde{\sigma}_{l-1}^2}\mathcal{H}^{l-1}\mathbf{x}_0\right). \tag{24}$$

B.6 Sampling the posterior

As discussed in section 3.2, the proposed generation scheme requires to sample from $\mathcal{N}(\mu_l, \Sigma_l)$ at each diffusion step, which due to the dimensions of Σ_l is not naive, since $\Sigma_l^{0.5}$ is needed in order to use the parameterization $\mathbf{x}_{l-1}|\mathbf{x}_l = \mu_l(\mathbf{x}_l) + \Sigma_l^{0.5}\mathbf{e}$ where $\mathbf{e} \sim \mathcal{N}(0, I)$. However, as we saw in section B.5, the operator Σ_l is equivalent to applying a linear filter $\mathbf{h}$, therefore one may seek to find a filter $\mathbf{d}$ such that

$$\Sigma = \mathcal{F}^\star\Lambda_{\mathbf{h}}\mathcal{F} = \mathcal{F}^\star\Lambda_{\mathbf{d}}^2\mathcal{F},$$

equivalently, we can break the DFT of the filter $\mathbf{h}$ to magnitude and phase such that

$$\text{DFT}\{\mathbf{h}\} = |\text{DFT}\{\mathbf{h}\}|e^{i\phi_h} = |\text{DFT}\{\mathbf{d}\}|^2 e^{i2\phi_d},$$

therefore

$$|\text{DFT}\{\mathbf{d}\}| = \sqrt{|\text{DFT}\{\mathbf{h}\}|},\ \measuredangle|\text{DFT}\{\mathbf{d}\}| = \frac{1}{2}\arctan\frac{Im\{\text{DFT}\{\mathbf{h}\}\}}{Re\{\text{DFT}\{\mathbf{h}\}\}}. \tag{25}$$

As a result, applying $\Sigma^{0.5}$ can be performed by employing the filter $\mathbf{d}$ defined in (25).

C Additional results

C.1 Latent space interpolation

As shown previously by [19], generative models have a very interesting property, where one may modify the latent variable of the generative model in order to control the "style" of the generated result, or for instance, given the latent variables of two images, one may interpolate the latent space and get an "in-between" images that in contrast to the pixel-domain interpolation, transition smoothly and naturally, as can be seen in Figures 10 and 11.

C.2 Latent space perturbation

Similar to what is presented in the main paper, we show additional latent variable perturbation in order to obtain better understanding of the latent space, where we add a small noise to a single diffusion step and see how it affects the generated result, as can be seen in Figures 6 and 12.

C.3 Latent variable swapping

In addition to what we presented above, we provide an additional ablation study that shows the benefit of UDPM, that is to swap the latent variables (noise maps) between two generated images, and study the effect on the generated result, as can be seen in Figure 13.

https://doi.org/10.52202/079017-0868

C.4 Generation Results

We provide below additional generation results of UDPM on the CIFAR10 dataset, as can be seen in Figure 14. We also provide an additional quantitative comparison to EDM [18] on the FFHQ [19] and AFHQv2 [6] datasets, as can be seen in Table 2.

	FFHQ		AFHQv2	
	steps	FID	steps	FID
EDM [18]	79/5	2.39/344.763	79/5	1.96/266.024
UDPM (ours)	**<1**	**7.41**	**<1**	**7.10**

Table 2: FID scores comparison between UDPM and EDM [18] on the FFHQ [19] and AFHQv2 [6] datasets. UDPM requires 3 diffusion steps, which is equivalent to 0.3 denoising steps of EDM.

	CIFAR10	AFHQv2	FFHQ
Learning rate	10^{-4}	10^{-4}	10^{-4}
Warmup steps	5000	5000	5000
Batch Size	64	64	64
Dropout	0.1	0.1	0
Optimizer	Adam	Adam	Adam
Number of GPUS	1	1	1

Table 3: Training hyperparameters.

	CIFAR10	AFHQv2	FFHQ
Architecture	NCSN [37]	NCSN [37]	NCSN [37]
Parameters	57.73M	65.41M	65.41M
Base channels	128	128	128
Channels Multiplier	2, 2, 2	1, 2, 2, 2	1, 2, 2, 2
Attention resolution	8, 4, 2	16, 8, 4	16, 8, 4
In channels	3	3	3
Out channels	12	12	12
Blocks per scale	4	4	4

Table 4: Model hyperparameters.

	CIFAR10	AFHQv2	FFHQ
Architecture	DDGAN [41]	DDGAN [41]	DDGAN [41]
Parameters	18.95M	23.86M	23.86M
Channels	192, 384, 512, 512, 512	192, 384, 512, 512, 512, 512	192, 384, 512, 512, 512, 512
In channels	3	3	3
Blocks per scale	1	1	1
Blocks type	NCSN down	NCSN down	NCSN down
Grad Penalty weight [41]	0.025, 0.1, 0.4	0.1, 0.4, 0.8	0.1, 0.4, 0.8
Dropout	0.4	0.4	0.3

Table 5: Discriminator network hyperparameters.

https://doi.org/10.52202/079017-0868

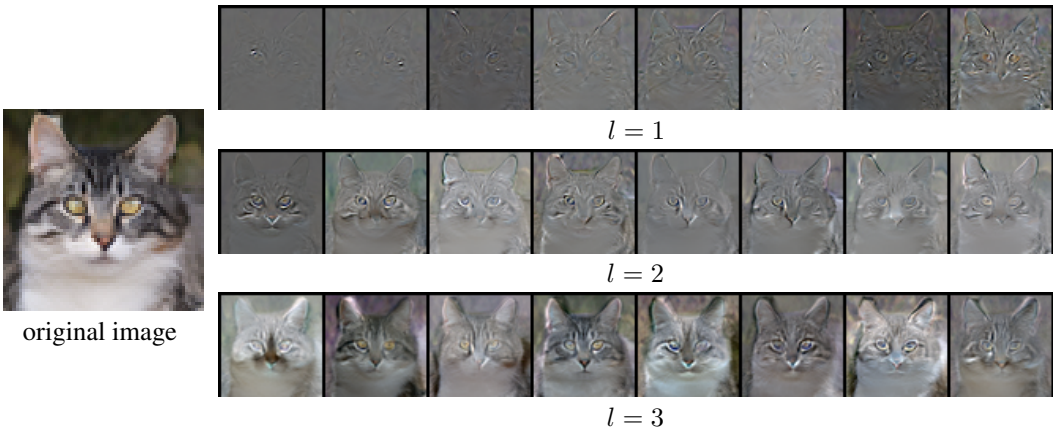


Figure 8: The first 8 principal components of the covariance matrix computed over 128 images generated by fixing two diffusion steps and adding small perturbation noise to the third (indexed above).

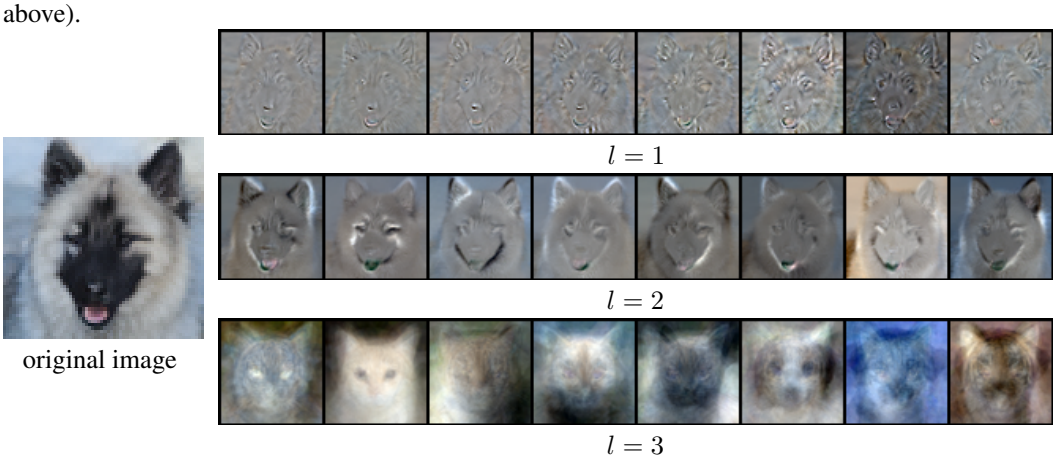


Figure 9: The first 8 principal components of the covariance matrix computed over 128 images generated by fixing two diffusion steps and re-randomizing the third (indexed above).

https://doi.org/10.52202/079017-0868

Figure 10: AFHQv2 [6] latent space interpolation example. The four corner images are interpolated by a weighted mixture of their latent noises, such that the other images are "in-between" images from the latent perspective, similar to what has been done in GANs [19]. All the images are of size 64×64.

https://doi.org/10.52202/079017-0868

Figure 11: FFHQ [19] latent space interpolation example. The four corner images are interpolated by a weighted mixture of their latent noises, such that the other images are "in-between" images from the latent perspective, similar to what has been done in GANs [19]. All the images are of size 64×64.

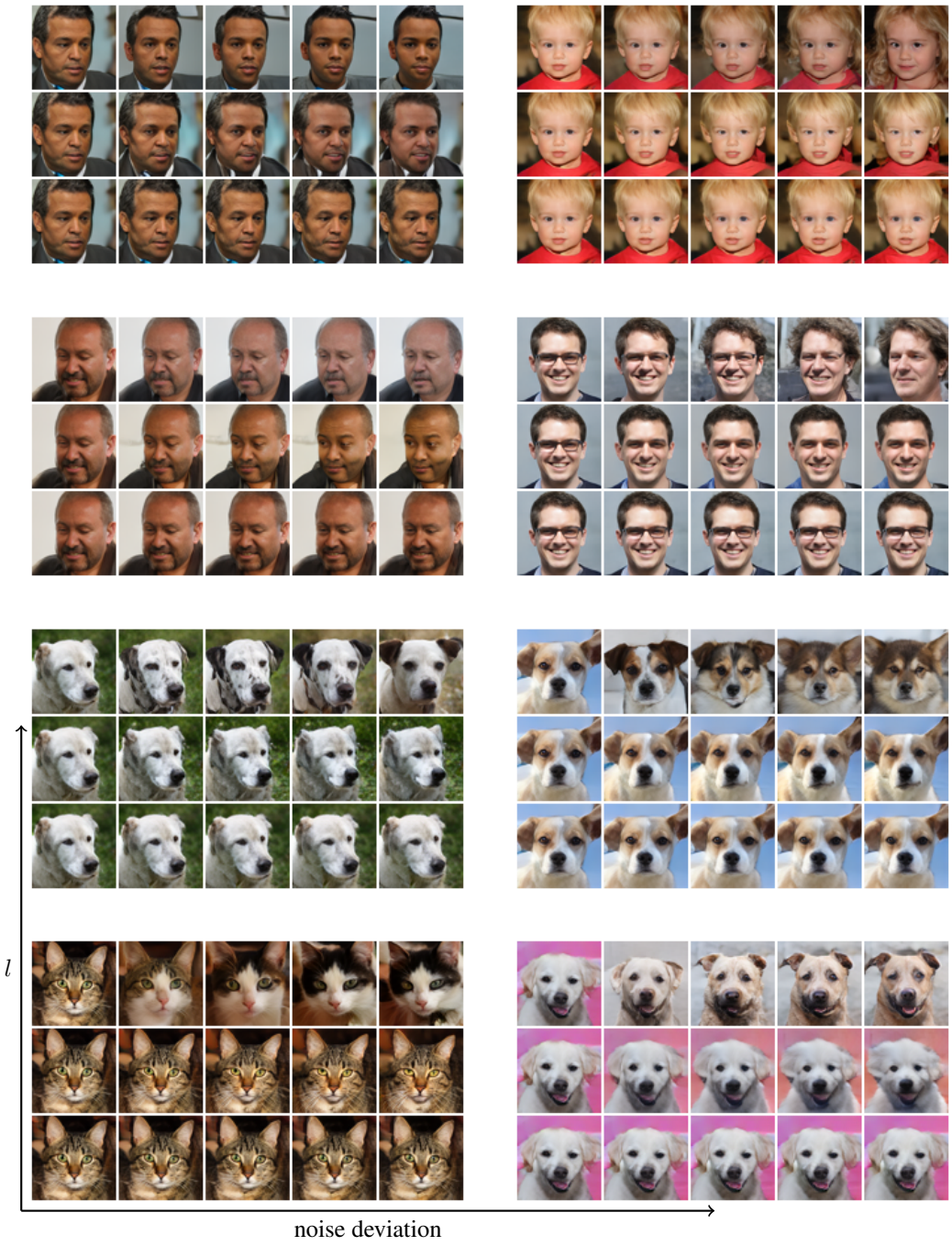


Figure 12: Latent space perturbation for 64×64 generated images. The original image is on the left, then the images generated by adding a small noise to the latent noise from diffusion step l. As can be seen, the initial diffusion steps ($l = 1$) controls the fine details of the image, while the final diffusion step ($l = 3$) changes the semantics of the image.

https://doi.org/10.52202/079017-0868

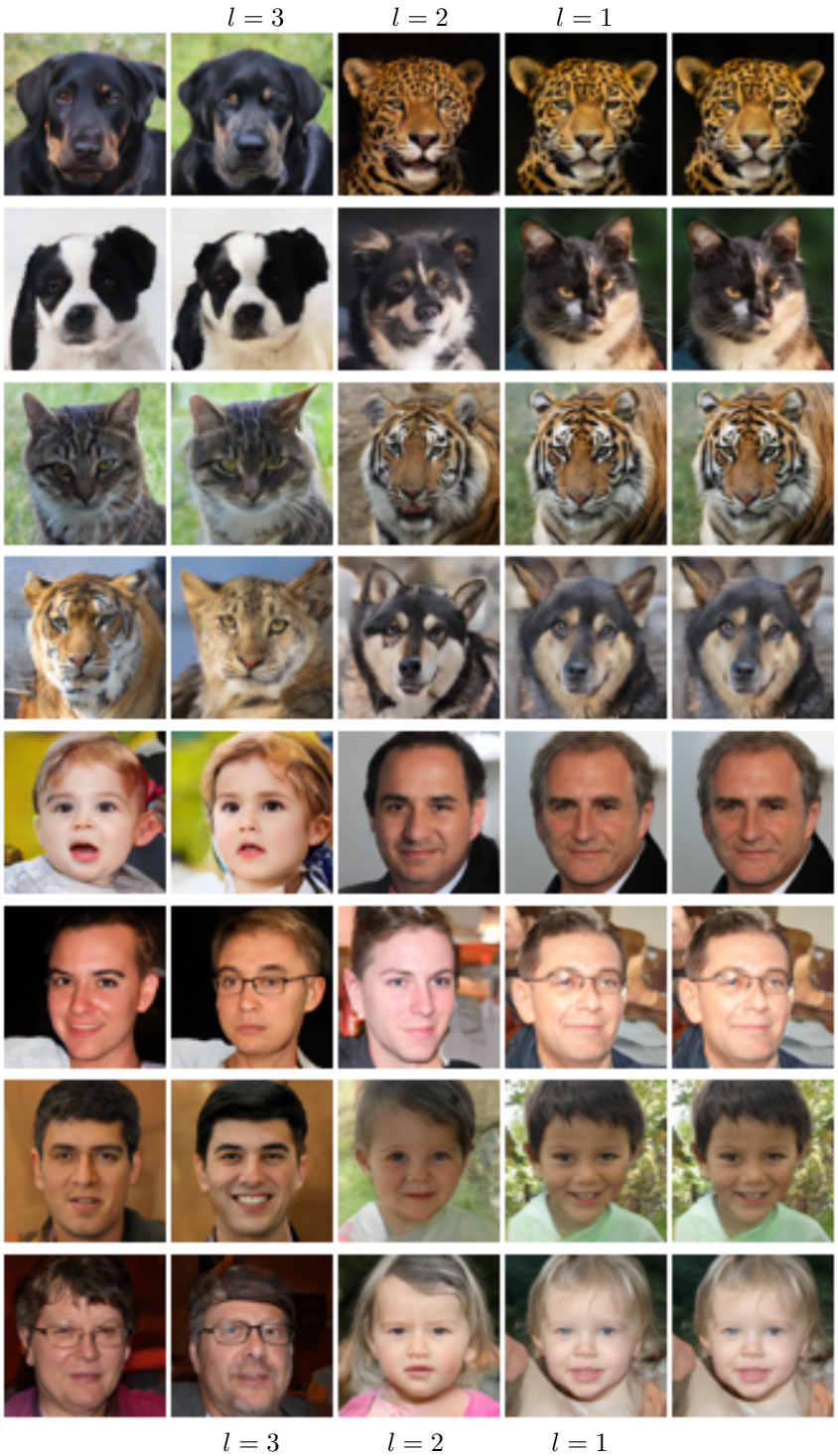
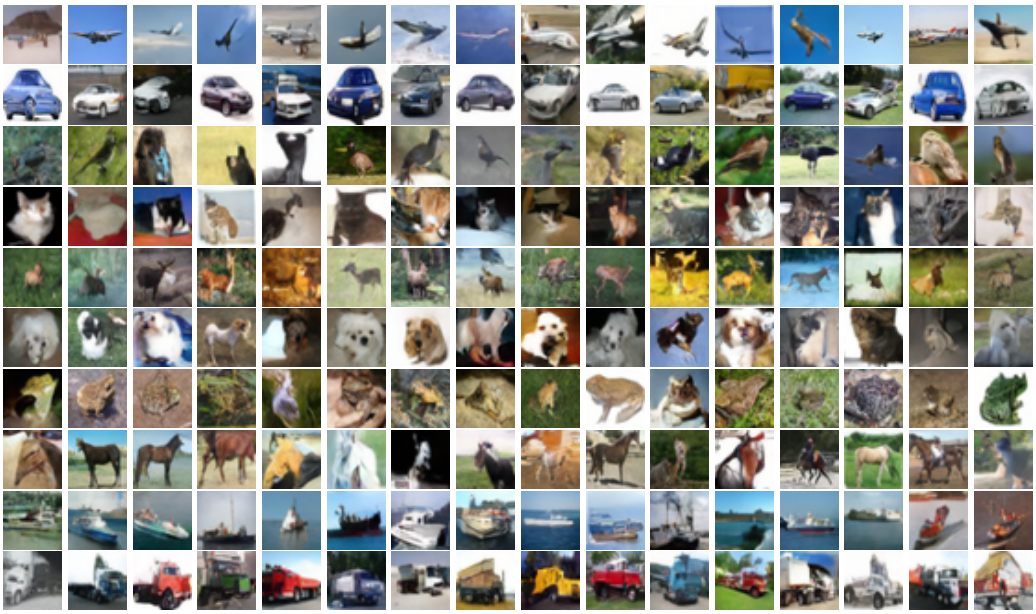


Figure 13: Latent variable swapping: Given the left and right images with the noise maps used for generating them, we replace the l-th noise map of the image on the right with the l-th noise map of the image on the left to see how each diffusion step affect the result (middle columns).



FID=6.86149

Figure 14: Generated 32×32 images of CIFAR10 [22] using conditional UDPM, requiring only 3 diffusion steps; equivalent to 0.3 traditional denoising step.

https://doi.org/10.52202/079017-0868

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See the limitation section in the paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Every assumption was explicitly mentioned in the paper, and the proof of every claim was added to the supplemental material.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In the appendix we present a table of the configurations used in the experiments shown in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

https://doi.org/10.52202/079017-0868

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and the models will be released publicly upon acceptance. The implementation details mentioned in the paper allow the results to be faithfully reproduced.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Tables 3, 4 and 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: we report results using the same metrics used in all previous works.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All the details are reported in the experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: All the points are satisfied.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the social impact statement.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/079017-0868

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: See the social impact statement.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets used in the paper, including code, data, and models, are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will publish our code and models upon acceptance.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-0868