On Analyzing Generative and Denoising Capabilities of Diffusion-based Deep Generative Models

Kamil Deja[*†]
Warsaw University of Technology
Warsaw, Poland
kamil.deja@pw.edu.pl

Anna Kuzina[*]
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
a.kuzina@vu.nl

Tomasz Trzciński
Warsaw University of Technology
Jagiellonian University of Cracow
Tooploox, IDEAS NCBR
tomasz.trzcinski@pw.edu.pl

Jakub M. Tomczak
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
j.m.tomczak@vu.nl

Abstract

Diffusion-based Deep Generative Models (DDGMs) offer state-of-the-art performance in generative modeling. Their main strength comes from their unique setup in which a model (the backward diffusion process) is trained to reverse the forward diffusion process, which gradually adds noise to the input signal. Although DDGMs are well studied, it is still unclear how the small amount of noise is transformed during the backward diffusion process. Here, we focus on analyzing this problem to gain more insight into the behavior of DDGMs and their denoising and generative capabilities. We observe a fluid transition point that changes the functionality of the backward diffusion process from generating a (corrupted) image from noise to denoising the corrupted image to the final sample. Based on this observation, we postulate to divide a DDGM into two parts: a denoiser and a generator. The denoiser could be parameterized by a denoising auto-encoder, while the generator is a diffusion-based model with its own set of parameters. We experimentally validate our proposition, showing its pros and cons.

1 Introduction

Diffusion-based Deep Generative Models [22] (DDGM) have recently attracted increasing attention, due to the unprecedented quality of generated samples [5, 9, 11]. The general idea behind this set of methods is to generate samples using diffusion processes [8, 10, 11, 23, 24]. In the *forward diffusion process*, an image is passed through a number of steps that consecutively add a small portion of noise to it. The *backward diffusion process* is a direct reverse of the forward process, where a generative model is trained to gradually denoise the image. With a sufficient number of the forward diffusion steps, noisy images approach isotropic Gaussian noise. Then, generating new examples is possible by applying the backward diffusion to the noise sampled from the standard Gaussian distribution.

While the performance of DDGMs is impressive, not all of their aspects are fully understood. Intuitively, a DDGM is trained to *remove* small amounts of noise from many intermediary corrupted images. Although this perspective is reasonable and complies with the interpretation of DDGMs using stochastic differential equations [10, 24], it is still unclear how the small amount of noise

[*]Equal Contribution
[†]Work done while visiting Vrije Universiteit Amsterdam

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

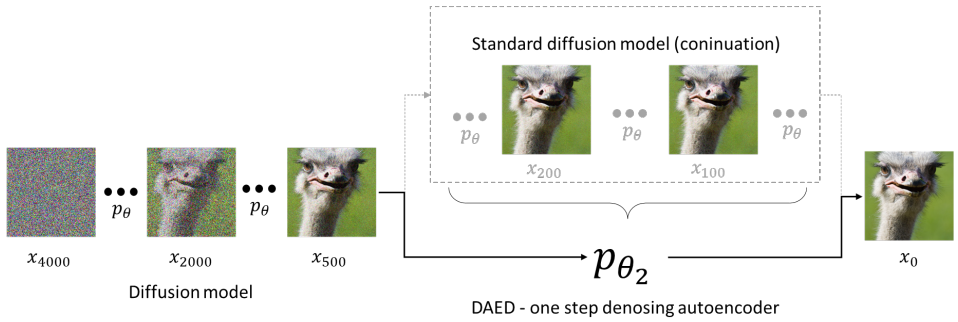


Figure 1: Overview of the proposed Denoising Auto-Encoder with Diffusion (DAED). To validate our hypothesis that DDGMs can be understood as a composition of a *generator* and *denoiser*, we propose to explicitly model the denoising part with a separate denoising autoencoder.

is *removed* during the backward diffusion process where images are composed of almost entirely random values. The more adequate intuition might be that in its initial steps, a diffusion model does not only remove noise but also introduces a new signal according to the distribution learned from the data. In this work, we further investigate this observation to understand the balance between the generative and denoising capabilities of DDGMs.

In particular, we aim to answer the following three questions in this paper: **(i)** Is there a transition in the functionality of the backward diffusion process that switches from generating to denoising? **(ii)** How does this split of functionality affect the performance? **(iii)** Does the denoising part in DDGMs generalize to other data distributions? As a result, the contribution of the paper is threefold:

- First, we analyze the noise distribution in the forward diffusion process and how steps of the diffusion process are correlated with the reconstruction error.
- Second, based on our analysis, we postulate that DDGMs are composed of two parts: a *denoiser* and a *generator*. As a result, we propose a new class of models that consist of a Denoising Auto-Encoder and a Diffusion-based generator shortened as DAED. DAED could be considered as a variation of DDGMs with an explicit split into the denoising part and the generating part.
- Third, we empirically assess the performance of DDGMs and DAED on three datasets (FashionMNIST, CIFAR10, CelebA) in terms of data generation and transferability (i.e., how DDGMs behave on different data distribution).

2 Background

2.1 Diffusion-Based Deep Generative Models (DDGMs)

Model formulation We follow the formulation of the Diffusion-based Deep Generative Models (DDGMs) as presented in [8, 22]. DDGMs could be seen as infinitely deep hierarchical VAEs with a specific family of variational posteriors [10, 11, 25, 26], namely, Gaussian diffusion processes [22]. Given a data point $\mathbf{x}_0$ and latent variables $\mathbf{x}_t, \ldots, \mathbf{x}_T$, we want to optimize the marginal likelihood $p_\theta(\mathbf{x}_0) = \int p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_T) \mathrm{d}\mathbf{x}_1, \ldots, \mathbf{x}_T$. We define the *backward* (or *reverse*) *process* as a Markov chain with Gaussian transitions starting with $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, that is:

$$p_\theta(\mathbf{x}_0, \ldots, \mathbf{x}_T) = p(\mathbf{x}_T) \prod_{t=0}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{1}$$

where $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$. Additionally, we define the *forward diffusion process* as a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, ..., \beta_T$, namely, $q(\mathbf{x}_1, \ldots, \mathbf{x}_T|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$, where $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$. Let us further define $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=0}^{t} \alpha_i$. Since the conditionals in the forward diffusion can be seen as Gaussian linear models, we can analytically calculate the following distributions:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\mathbf{x}_0, (1 - \overline{\alpha}_t)\mathbf{I}), \tag{2}$$

and

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}), \tag{3}$$

where $\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\overline{\alpha}_{t-1}}\beta_t}{1-\overline{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{x}_t$, and $\tilde{\beta}_t = \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t}\beta_t$. We can use (2) and (3) to define the variational lower bound as follows:

$$\begin{aligned}\ln p_\theta(\mathbf{x}_0) \geq L_{vlb}(\theta) := & \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\ln p_\theta(\mathbf{x}_0|\mathbf{x}_1)]}_{-L_0} - \underbrace{D_{\mathrm{KL}}\left[q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T)\right]}_{L_T} \\ & - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} D_{\mathrm{KL}}\left[q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right]}_{L_{t-1}}.\end{aligned} \tag{4}$$

that we further optimize with respect to the parameters of the backward diffusion.

The conditional likelihood In this paper, we focus on images, thus, data is represented by integers from 0 to 255. Following [8], we scale them linearly to $[-1, 1]$. As a result, to obtain discrete log-likelihoods, we consider the discretized (binned) Gaussian conditional likelihood [8]:

$$p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right) = \prod_{i=1}^{D} \int_{\delta_-\left(x_0^i\right)}^{\delta_+\left(x_0^i\right)} \mathcal{N}\left(x; \mu_\theta^i\left(\mathbf{x}_1, 1\right), \sigma_1^2\right) \mathrm{d}x, \tag{5}$$

where D is the data dimensionality of $\mathbf{x}_0$, and i denotes one coordinate of $\mathbf{x}_0$, and:

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \quad \delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}. \tag{6}$$

Noise scheduling Originally, [8] propose to linearly scale the noise parameters β_t (*linear scheduling*), e.g., scaling linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. In [17], authors suggest to increase the number of less noisy steps through *cosine scheduling*: $\bar{\alpha}_t = \frac{f(t)}{f(0)}$, $f(t) = \cos\left(\frac{t/T+c}{1+c} \cdot \frac{\pi}{2}\right)^2, c > 0$ with clipping the values of β_t to 0.999 to prevent potential instabilities at the end of the diffusion.

Training details In [8], authors notice that a single part of the variational lower bound is equal to:

$$L_t(\theta) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t\left(1-\overline{\alpha}_t\right)} \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right], \tag{7}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\boldsymbol{\epsilon}_\theta$ is a neural network predicting the noise $\boldsymbol{\epsilon}$ from $\mathbf{x}_t$. Since we use (3) in the variational lower bound objective (4), and $\mathbf{x}_t$ could be sampled from the forward diffusion for a given data, see (2), we can optimize one layer at a time. In other words, we can randomly pick a specific component of the objective, L_t, and update the parameters by optimizing L_t without running the whole forward process from $\mathbf{x}_0$ to $\mathbf{x}_T$. As a result, the training becomes very efficient and learning very deep models (with hundreds or even thousands of steps) is possible.

In [8], it is also proposed to train a simplified objective that is a version of (7) without scaling, namely:

$$L_{t,\text{simple}}(\theta) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right], \tag{8}$$

where t is uniformly sampled between 1 and T. To further reduce computational and memory costs, typically, a single, shared neural network is used for modeling $\boldsymbol{\epsilon}_\theta$ [8, 11, 17] that is parameterized by an architecture based on U-Net type neural net [18]. The U-Net could be seen as a specific auto-encoder that passes all codes from the encoder to the decoder.

2.2 Denoising Auto-Encoders

Another class of models, Denoising Auto-Encoders (DAEs), is similar to DDGMs in the sense that they also revert a known corruption process. However, DAEs are trained to remove the noise in a single pass, and unlike DDGMs, they cannot generate new objects. Specifically, DAEs are auto-encoders that reconstruct a data point $\mathbf{x}_0$ from its corrupted (noisy) version [1, 2, 4, 28]. Let us

denote the auto-encoder by $f_\varphi(\cdot)$. Using the same notation as for DDGMs, the Gaussian corruption distribution is $q(\mathbf{x}_1|\mathbf{x}_0)$. Then, a DAE maximizes the following objective function:

$$\ell(\mathbf{x}_0;\varphi) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\ln p\left(\mathbf{x}_0|f_\varphi\left(\mathbf{x}_1\right)\right)\right]. \tag{9}$$

and, in particular, for the Gaussian distribution with the identity covariance matrix, we get the original objective for DAEs [28]: $\ln p\left(\mathbf{x}_0|f_\varphi\left(\mathbf{x}_1\right)\right) = -\left\|\mathbf{x}_0 - f_\varphi\left(\mathbf{x}_1\right)\right\|^2 + const$.

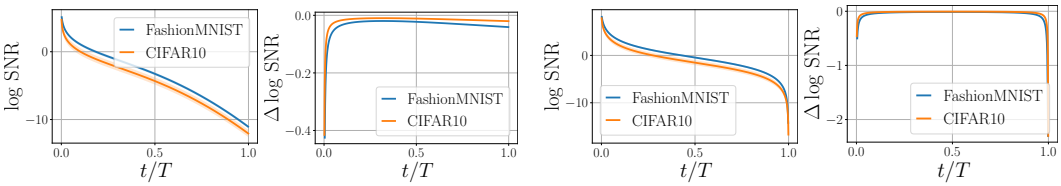
3 An analysis of DDGMs

The core idea behind DDGMs is the gradual noise injection to images as we go forward in time such that the final object is a sample from the standard Gaussian distribution. Then, in the backward diffusion process model reverts this procedure and, as a result, generates new objects. Therefore, understanding the success of DDGMs relies heavily on understanding how the injected noise influences the behavior of both training and the model itself.

The noise distribution in the forward diffusion process The first question we ask is how much corrupted an image gets after applying a specific noise schedule. Following [8, 12, 17], we can utilize the signal-to-noise ratio (SNR), expressed as the squared mean of a signal (here: image) divided by the variance of a signal, to quantify the amount of noise in $\mathbf{x}_t$. For this purpose, the quantity of interest is the forward diffusion for a given $\mathbf{x}_0$, namely, $q(\mathbf{x}_t|\mathbf{x}_0)$, that results in the following SNR:

$$SNR(\mathbf{x}_0, t) = \frac{\overline{\alpha}_t \mathbf{x}_0^2}{1 - \overline{\alpha}_t}. \tag{10}$$

Similarly to [11], we formulate the forward diffusion in such a way that the SNR is strictly monotonically decreasing in time, namely, $SNR(\mathbf{x}_0, t) < SNR(\mathbf{x}_0, s)$ for $t > s$. This means that an image becomes more noisy as we go forward in time.



(a) Linear noise schedule (b) Cosine noise schedule

Figure 2: Logarithm of the signal-to-noise ratio averaged over the dataset (solid line) and its standard deviation, and the difference of the log SNR within two consecutive time steps.

In Figure 2 (left) we plot the logarithm of the SNR for both linear (Figure 2.a) and cosine (Figure 2.b) noise schedules for two datasets (FashionMNIST and CIFAR10). We average SNR over the $\mathbf{x}_0$'s (from the corresponding dataset). The right column depicts the change of the log SNR, i.e., its discrete derivative $\Delta \log \text{SNR}(t) = \log \text{SNR}(x_0, t) - \log \text{SNR}(x_0, t-1)$. First of all, we can notice a point at which the log-SNR drops below 0. This corresponds to the situation of the noise overshadowing the signal. In the case of the linear noise schedule, this happens after about 20% of steps, while for the cosine noise schedule, it appears after about $25 - 50\%$ of steps. However, the transition occurs in both cases. The biggest changes in the log-SNR are noticeable within the first 10% of steps. This may suggest that the signal is the strongest within the first $10 - 20\%$ of the forward diffusion process steps, and then it starts being overshadowed by the noise.

The reconstruction error of DDGMs Since we know that the signal is not lost within the first $10 - 20\%$ of steps, the next question is about the reconstruction capabilities of DDGMs, namely, what is the reconstruction error of $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$. To be clear, we are not interested in how much each step of a DDGM contributes to the final objective (e.g., see Figure 2 in [17]) but rather how well a DDGM reconstructs a noisy image $\mathbf{x}_t$. In Figure 3 we plot the *Mean Absolute Error* (MAE) and the *Multi-Scale Structural Similarity* (MS-SSIM) [29] that both measure the difference between an original image $\mathbf{x}_0$ and a corrupted image at the tth step $\mathbf{x}_t$ reversed by the backward diffusion.

We present the values on two datasets (FashionMNIST and CIFAR10) for the first 20% of steps. Apparently, after around 10% of the steps, the reconstruction error starts growing, and the MAE increases linearly above 0.1 (i.e., about 6% of error per pixel). At the same time, the MS-SSIM drops below $0.9 - 0.95$ (i.e., the discrepancy between original images and reconstructions becomes perceptually evident). This observation might suggest that DDGMs could be roughly divided into two parts: a fraction of steps of a DDGM (e.g., first 10% of the steps) constitute a *denoiser* that turns a corrupted image into a clear image, and the remaining steps of the DDGM are responsible for turning noise into a noisy structure (a corrupted image), i.e., a *generator* that generates meaningful patterns. In other words, we claim that DDGM can be interpreted as a composition of a denoiser and a generator, but the boundary between those two parts is fluid. Moreover, the denoiser gradually removes the noise in a generative manner (i.e., by sampling $\mathbf{x}_{t-1} \sim p(\mathbf{x}_{t-1}|\mathbf{x}_t)$).

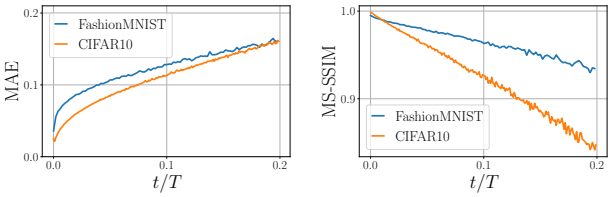


Figure 3: The averaged reconstruction error calculated using (*left*) the MAE, and (*right*) the MS-SSIM at different steps of a DDGM.

DDGMs as hierarchical VAEs In this paper, we postulate that DDGMs could be seen as a composition of parts that serve different purposes. We can get additional insight into our claim by noticing a close connection between DDGMs and hierarchical VAEs. As presented in [10, 11, 25], if we treat all $\mathbf{x}_t$'s with $t > 0$ as latents, and see the forward diffusion process as a composition of (non-trainable) variational posteriors, DGGMs become a specific formulation of hierarchical VAEs. On the other hand, we can start with a VAE with a single latent variable, $\mathbf{x}_1$, for which the variational lower bound is equal to:

$$\ln p(\mathbf{x}_0) \geq \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)} \left[\ln p(\mathbf{x}_0|\mathbf{x}_1)\right] - D_{\mathrm{KL}}[q(\mathbf{x}_1|\mathbf{x}_0)||p(\mathbf{x}_1)]. \tag{11}$$

Then, similarly to [27, 30], the marginal $p(\mathbf{x}_1)$ could be further modeled by a DDGM. By keeping the dimensionality of $\mathbf{x}_1$ the same as $\mathbf{x}_0$, and taking the variational posterior $q(\mathbf{x}_1|\mathbf{x}_0)$ to be fixed and part of the forward diffusion, we get the DDGM model. This perspective of combining a VAE with a DDGM opens new possibilities for developing hybrid models.

4 DEAD: Denoising Auto-Encoder with Diffusion

In this work, we propose a specific combination that distinctly splits the DDGM into generative and denoising parts. As noted in the previous section, the signal in the forward diffusion process is the strongest within the first $10 - 20\%$ of steps, and, thus, we postulate to perceive this first part of a DDGM as a denoiser. Together with the observation about the combination of a VAE with a DDGM-based prior, we consider turning a denoising auto-encoder into a generative model as presented in Figure 1. We bring a DDGM-based part into DAE for generating corrupted images. The resulting objective is the following:

$$\overline{\ell}(\mathbf{x}_0; \varphi, \theta) = \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)} \left[\ln p\left(\mathbf{x}_0|f_\varphi\left(\mathbf{x}_1\right)\right) + \ln p_\theta(\mathbf{x}_1)\right] \tag{12}$$

$$\geq \underbrace{\mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1|\mathbf{x}_0)} \left[\ln p\left(\mathbf{x}_0|f_\varphi\left(\mathbf{x}_1\right)\right)\right]}_{\ell_{\mathrm{DAE}}(\mathbf{x}_0;\varphi)} + \underbrace{\mathbb{E}_{q(\mathbf{x}_2,\ldots,\mathbf{x}_T|\mathbf{x}_1)} \left[\frac{\ln p_\theta(\mathbf{x}_1,\ldots,\mathbf{x}_T)}{q(\mathbf{x}_1,\ldots,\mathbf{x}_T|\mathbf{x}_0)}\right]}_{\ell_{\mathrm{D}}(\mathbf{x}_0;\theta)}, \tag{13}$$

where in (13) we introduce additional latent variables and the variational posterior over them, that yields the variational lower bound. We call the resulting model *DAE with a Diffusion*, or DEAD for short. In a sense, DAED is a DDGM with distinct parameterizations of the part between $\mathbf{x}_0$ and $\mathbf{x}_1$, and the part for the remaining $\mathbf{x}$'s. Thus, DEAD is almost identical to a DDGM, but there are the following differences: **(i)** We can control the amount of noise in $q(\mathbf{x}_1|\mathbf{x}_0)$. It can correspond to the first step of the forward diffusion model, or we can introduce more noise at once that would correspond to several steps in the DDGM. **(ii)** We use two different parameterizations, namely, an auto-encoder (e.g., a U-Net architecture) for $f_\varphi(\cdot)$ and a separate, shared U-Net for modeling the DDGM from $\mathbf{x}_1$ to $\mathbf{x}_T$. Since there are two neural networks, the lower bound to the objective $\overline{\ell}$ is in fact a composition of two objectives with disjunctive parameters, namely, the objective for the

denoiser, ℓ_{DAE}, and the objective for the *generator* (i.e., the diffusion-based generative model), ℓ_{D}. **(iii)** In the DAED, we introduce the *denoiser* explicitly and make a clear distinction between the denoising and the generating parts while, as discussed earlier, this boundary is rather fluid in DDGMs. By introducing DAED, we can analyze what happens if we distinctly divide those two aspects with two separate parametrizations.

Moreover, we hypothesize that the resulting model may better generalize across various data distributions due to decoupling the parameterization of the denoiser and the generator. The training dataset may bias a single, shared parameterization in a DDGM, and while denoising an image from a different domain, it may add some artifacts from the source. While with two distinct parameterizations, there might be a lower chance for that. We evaluate this hypothesis in the experiments.

5 Related work

DDGM for image generation Various modifications of DDGMs were recently proposed to improve their sampling quality. This includes simplifying the learning objective and proposing new noise schedulers, which allow DDGMs to achieve state-of-the-art results. In this work, we show that splitting the decoder into two parts, namely, a denoiser and a generator, can benefit the performance, especially when training with the variational lower bound.

Properties of DDGMs In [8] authors notice that DDGMs can be beneficial for lossy compression, observing (Figure 5 in [8]) that most of the bits are allocated to the region of the smallest distortion that corresponds to the first steps of a DDGM. We draw a similar conclusion when discussing the denoising ability of the diffusion model in Section 3. However, we base our analysis on the signal-to-noise ratio rather than compression. On the other hand [21] focus on the computational complexity of DDGM and propose a progressive distillation that iteratively reduces the number of diffusion steps. The work shows that it is possible to considerably reduce the number of sampling steps without losing performance. We believe that their results support our intuition that it is reasonable to combine several initial steps into a single denoiser model. In [3], authors evaluate how the diffusion process changes in time when model is trained with different objectives (Eq. 7 or Eq. 8). They observe that the image generation process differs significantly and that it is more beneficial to switch between those two approaches at different stages of the diffusion. In this work, we also investigate changes in the diffusion process, but we focus on the generative and denoising capabilities of the model instead.

Connection to hierarchical Variational Autoencoders Several works have noted the connection of DDGM to VAEs. In [10] authors focus on the continuous diffusion models and draw the connection to the infinitely deep hierarchical VAEs. In [11] authors further explore this connection, formulate a VLB objective in terms of the signal-to-noise ratio and propose to learn noise schedule, which brings the forward diffusion process even closer to the encoder of a VAE. Recently a latent score-based generative model (LSGM) was proposed [27], which can be seen as a VAE with the score-based prior. We follow a similar direction and propose to see a DDGM as a combination of a denoising auto-encoder with an additional diffusion-based generator of corrupted images.

6 Experiments

Experimental setup In all the experiments, we use a U-Net-based architecture with timestep embeddings as proposed in [8, 17]. We train all the models with a linear β scheduler and uniform steps sampler to simplify the comparison. All implementation details and hyperparameters are included in the Appendix (A.4) and code repository [3]. For DAED, we use the same architecture for both the diffusion part and the denoising autoencoder. We run experiments on three standard benchmarks with different complexity: FashionMNIST [31] of gray-scale 28×28 images, CIFAR-10 [13] of 32×32 natural images, and CelebA [14] of 64×64 photographs of faces. We do not use any augmentations during training for any dataset. We report results for both variational lower bound loss (VLB) [22] and simplified objective [8]. Following [17] we evaluate the quality of generations with Fréchet Inception Distance (FID) [6] and distributions Precision (Prec) and Recall (Rec) metrics [19] that disentangle FID score into two aspects: the quality of generated results (Precision) and their diversity (Recall).

[3] `https://github.com/KamilDeja/analysing_ddgm`

6.1 Is there a transition in functionality of the backward diffusion process that switches from generating to denoising?

In section 3, we investigate how the signal-to-noise ratio and the reconstruction error of a DDGM change with the increasing number of diffusion steps (see Figure 4). Based on this analysis, we postulate that DDGMs can be divided into two parts: a *denoiser* and a *generator*. To determine the switching point, we propose an experiment that answers the following question:

Is there a denoising part of a DDGM that is agnostic to the signal from the data?

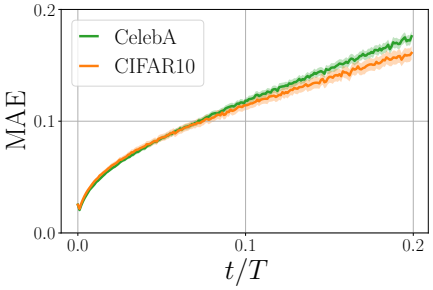


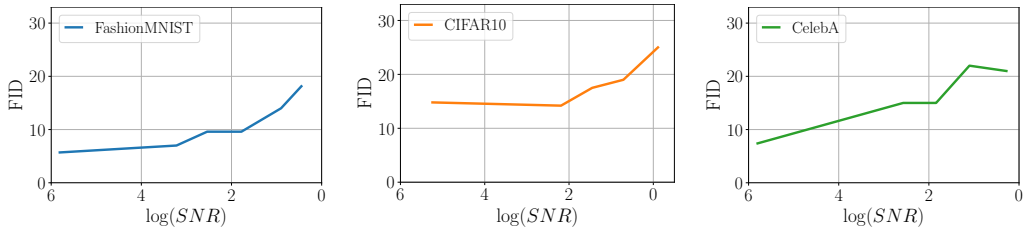
Figure 4: The MAE for a DDGM trained on CIFAR10 and evaluated on CIFAR10 & CelebA, with a 0.95 confidence interval.

To that end, we refer once more to the analysis of the reconstruction error (e.g., MAE) from different diffusion steps. This time, however, we compare the quality of reconstructions with a single DDGM model trained on the CIFAR10 dataset and then evaluated on CIFAR10 and CelebA. The result of this experiment is presented in Figure 4. Interestingly, we notice that for approximately 10% of the initial steps of the DDGM, there is a negligible difference in the reconstruction error between these two datasets. This fact may suggest that, indeed, the model does not require any information about the background data signal in the first steps, and it is capable of denoising corrupted images. However, after this point (about 10% of steps), the reconstruction error starts growing faster for the dataset the model was not trained on. This indicates that information about the domain becomes important and affects performance.

6.2 How does splitting DDGMs into generative and denoising parts affect the performance?

The results so far confirm our claims that DDGMs could be divided into denoising and generative parts. Independently of a dataset, there appears to be a transition point at which a DDGM stops generating a corrupted image from noise and starts denoising it in a generative manner. Here, we aim to verify whether it is possible to do a clear split into a denoising part and a generating part. For this purpose, we use the introduced DAED approach that consists of a DAE part (the denoiser) and a DDGM (the generator) parameterized by two distinct U-Nets.

First, we consider a situation in which we train a DDGM using the simplified objective (8) and then replace the first steps with a DAE. In other words, we train a DAED in two steps: first the DDGM and then the DAE. This experiment aims to check how the decoupling of the DDGM into two parts influences the model performance. In Figure 5 we present the dependency between the log-SNR at the splitting point and the FID score. In all cases, the performance of DAED is comparable to the DDGM if we replace the DAE with up to the 10% of the steps that correspond to $log(SNR)$ is equal to around 4. For more complicated datasets like CIFAR10 and CelebA, fewer steps could be replaced. This effect could be explained by the fact that images in these datasets have three channels (RGB), and removing noise is more problematic. That outcome reconfirms our presumptions that it is reasonable to split the DDGM since the final performance is not significantly affected by the division for an adequately chosen splitting point.



(a) FashionMNIST (b) CIFAR10 (c) CelebA

Figure 5: The performance (FID) of DAED with different switching points with respect to the logarithm of the signal to noise ratio (10) on three different datasets.

Table 1: FID Precision (Prec) and Recall (Rec) scores. For each row, we indicate the length of the diffusion process (T) and the training objective (Loss). **Best results in bold**.

Model		Fashion Mnist				CIFAR10				CelebA			
	Loss	T	FID ↓	Prec ↑	Rec ↑	T	FID ↓	Prec ↑	Rec ↑	T	FID ↓	Prec ↑	Rec ↑
DDGM	VLB	500	8.9	68	53	1000	26	53	54	1000	23	51	21
DAED $\beta_1 = 0.1$	VLB	468	9.1	**71**	60	900	20	59	46	900	18	63	**30**
DAED $\beta_1 = 0.001$	VLB	499	**7.5**	**71**	**64**	999	**15**	**60**	**60**	999	**16**	**70**	27
DDGM	Simple	500	7.8	72	**65**	1000	**7.2**	**65**	**61**	1000	**4.9**	66	**57**
DAED $\beta_1 = 0.1$	Simple	468	9.6	**73**	58	900	19	62	50	900	22	**67**	27
DAED $\beta_1 = 0.001$	Simple	499	**5.7**	69	64	999	14.8	**65**	53	999	7.4	**67**	54

To get further insight into the qualitative performance, in Figure 6 we demonstrate how the selection of the splitting point with respect to the Signal to Noise Ratio (SNR) affects the quality of final generations[4]. We present non-cherry-picked samples from DAED trained in the same manner as described in the previous paragraph. As expected, the more noise the DAE part (the denoiser) must deal with (see the values of β_1 in Figure 6), the fewer details in the generations there are. These samples again indicate that by replacing some steps with a denoiser, we get a trade-off between "cleaning" the corrupted image or, in fact, further generating details. It seems that there is a sweet spot for perceptually appealing images that contain details and are "smooth" at the same time, see $\beta_1 = 0.025$ in Figure 6. However, as it is typically difficult to provide convincing arguments by *staring* at samples, we further propose to analyze quantitative measures.

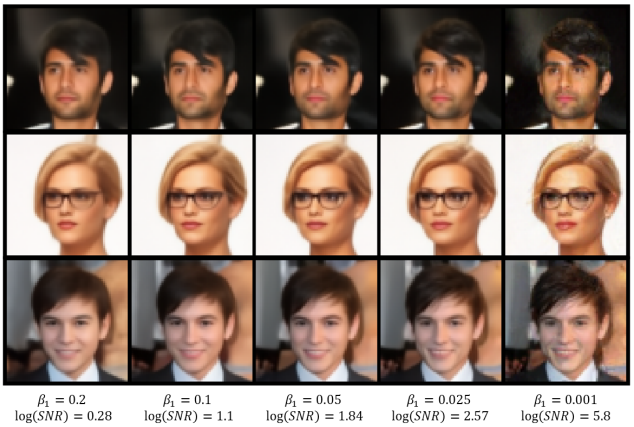


Figure 6: Examples of generations from DAED with the same noise value and different switching points.

In Table 1, we compare the performance of DAED against the DDGM on FashionMNIST, CIFAR10, and CelebA in terms of FID, Precision and Recall scores. We want to highlight that our goal is not to achieve SOTA results on the before-mentioned datasets but to verify whether we can gain some further understanding and, potentially, some improvement by splitting the denoising and generative parts. We consider two scenarios, namely, learning a DDGM and DAEDs using either the variational lower bound (VBL) or the simplified objective (Simple) with various lengths of the diffusion. Interestingly, DAED outperforms the DDGM when these models are trained using the VBL loss. For the simplified objective, DAED trained with the same number of diffusion steps yields slightly lower performance than standard DDGMs. As indicated by the Precision/Recall, generations from DAED are as precise as those from DDGM. However, they lack certain diversity, probably due to the smoothing effect of the DAE part. Detailed results for other setups are presented in Appendix A.[5]

6.3 Does the noise removal in DDGMs generalize to other data distributions?

The last question we are interested in is the generalizability of DDGMs to other data distributions. We refer to this concept as *transferability* for short. In other words, the goal of this experiment is to determine whether we can reuse a model or its part on new data with as good performance as possible. In this experiment, we rely on the results presented in Section 6.1 where roughly the first 10% of steps could be seen as the denoising part. To further strengthen this perspective, we also utilize DAED with an explicit division into the denoising and generating parts.

First, we consider the case in which we compare the reconstruction errors measured by the MAE and the MS-SSIM. In this scenario, we train a DDGM on a source dataset and then assess it on

[4]Generations for all datasets are presented in Appendix A.2

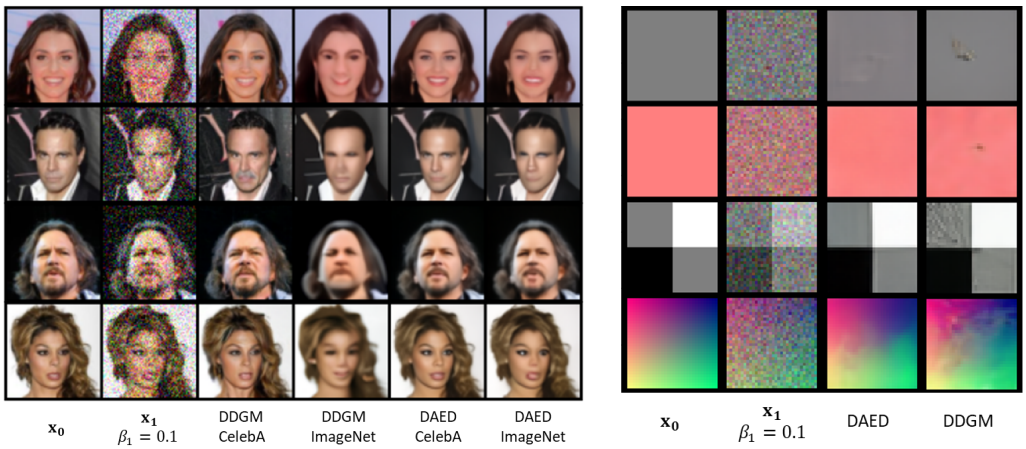
[5]In Appendix A.6 we show that increasing the number of parameters of DDGMs to be comparable to DAED does not lead to significant performance improvements.

Table 2: Reconstruction errors measured by MAE (↓), MS-SSIM (↑) for images noised with $\beta_1 = 0.1$. *To evaluate models trained on CIFAR10, we downscale CelebA to 32×32. **Best results in bold.**

	Target dataset	CIFAR10		CIFAR00		CelebA*	
Source Dataset	Model	MAE	MS-SSIM	MAE	MS-SSIM	MAE	MS-SSIM
CIFAR10	DDGM VLB	0.091	0.94	0.097	0.94	0.093	0.95
	DDGM Simple	0.085	0.95	0.097	0.94	0.096	0.95
	DAED	**0.065**	**0.97**	**0.074**	**0.97**	**0.068**	**0.97**
ImageNet	DDGM VLB	0.113	0.93	0.110	0.93	0.077	0.96
	DDGM Simple	0.113	0.94	0.111	0.93	0.068	0.96
	DAED	**0.071**	**0.97**	**0.071**	**0.97**	**0.050**	**0.98**

a target dataset. We use CIFAR10 or ImageNet (32x32 or 64x64) as source data and CIFAR10, CIFAR100, or CelebA as target data. For each image from the target dataset, we apply the DAE part of DAED to obtain the reconstruction or 793 steps of the forward and backward diffusion in the case of the DDGM, which corresponds to the same level of added noise. For this experiment, we use the pre-trained DDGM from [17] that consists of 4000 steps and uses the cosine noise scheduler. The results are outlined in Table 2. First of all, there is no significant difference in the performance of DDGMs trained with either the VBL objective or the simplified objective. They achieve a quite satisfactory MAE and MS-SSIM scores. However, DAED outperforms the DDGMs, obtaining much better transferability. We explain it by the fact that probably, with each step in the denoising part DDGM adds details that are typical for source data while DAED focuses on removing noise and produces a smoother output. This outcome may further suggest that splitting DDGMs into two parts with two separate parameterizations is reasonable and even beneficial.

To get further insight into the transferability behavior, we present a few (non-cherry-picked) examples from CelebA in Figure 7.a and four toy examples in Figure 7.b. We use the same setup as explained in the previous paragraph (i.e., the pre-trained DDGM provided in [17]), and the images are noised with $\beta_1 = 0.1$. In columns 3–6 in Figure 7.a, we present reconstructions for the DDGM trained on CelebA, the DDGM trained on ImageNet, DAED trained on CelebA, and DAED trained on ImageNet, respectively. It becomes apparent that the DDGM trained on CelebA denoises the image by generating new details while DAED denoises by smoothing. Interestingly, DAED performs better than the DDGM when we use ImageNet-trained models to denoise CelebA. In Figure 7.b, we depict several toy examples that were denoised with the DDGM and DAED trained on CIFAR10. We see that the DDGM adds many details that are artifacts from the source data. It seems that DAED does not suffer from that behavior.



(a) Reconstructions on CelebA (b) Toy examples

Figure 7: (a) Denoising of image with 0.1 noise using either DAED or the corresponding number of the DDGM steps. (b) Four noisy toy examples denoised by DAED and the DDGM.

7 Conclusion

In this work, we investigate the generative and denoising capabilities of the Diffusion-based Deep Generative Models. We observe and experimentally validate that it is reasonable to understand DDGMs as a combination of two parts. The first one generates noisy samples from the pure noise by inputting more signal from a learned data distribution, while the second one removes the remaining noise from the signal. Although for standard DDGMs, the exact switching point between those two parts is fluid, we propose a new approach dubbed DAED that is explicitly built as a combination of a generative component (a DDGM) and a denoising one (a DAE). In the experiments, we observe that DAED simplifies training with a standard VLB loss function that leads to improved performance. On the other hand, with increasing noise processed by DAE, DAED smoothens the generations resulting in lower performance when training with the simplified objective. We further show that DDGMs, and DAED especially, generalize well to unseen data, what opens new possibilities for further research in terms of transfer or continual learning of DDGMs.

8 Acknowledgements

This research was funded by National Science Centre, Poland (grant no 2018/31/N/ST6/02374 and 2020/39/B/ST6/01511), Foundation for Polish Science (grant no POIR.04.04.00-00-14DE/18-00 carried out within the Team-Net program co-financed by the European Union under the European Regional Development Fund), and the Hybrid Intelligence Center, a 10-year programme funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research. Computation was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

References

[1] G. Alain and Y. Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

[2] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. *Advances in neural information processing systems*, 26, 2013.

[3] Y. Benny and L. Wolf. Dynamic dual-output diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11482–11491, 2022.

[4] M. Chen, K. Weinberger, F. Sha, and Y. Bengio. Marginalized denoising auto-encoders for nonlinear representations. In *International conference on machine learning*, pages 1476–1484. PMLR, 2014.

[5] P. Dhariwal and A. Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

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

[7] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

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

[9] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

[10] C.-W. Huang, J. H. Lim, and A. C. Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34, 2021.

[11] D. P. Kingma, T. Salimans, B. Poole, and J. Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

[12] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

[13] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. In *Citeseer*, 2009.

[14] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[15] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[16] C. Nash, J. Menick, S. Dieleman, and P. W. Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021.

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

[18] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[19] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. *arXiv preprint arXiv:1806.00035*, 2018.

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

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

[22] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[23] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

[25] J. M. Tomczak. *Deep Generative Modeling*. Springer Cham, 2022.

[26] B. Tzen and M. Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

[27] A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34, 2021.

[28] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[29] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. IEEE, 2003.

[30] A. Wehenkel and G. Louppe. Diffusion priors in variational autoencoders. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, 2021.

[31] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017. arXiv:1708.07747.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary material
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplementary material
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use pretrained model for the transferability experiment and cite the authors correspondingly
   (b) Did you mention the license of the assets? [Yes] See supplementary material
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional experiments

In this section, we present extended evaluation of all models introduced in the main work. Following [17], we show the assessment of generations quality in terms of additional metrics namely Inception Score [20] and spatial Fréchet Inception Distance [16] – a version of standard FID score but based on spatial image features.

Table 3: Extended evaluation results for CIFAR10 dataset.

Model			CIFAR-10				
	Loss	T	IS ↑	FID ↓	sFID ↓	Prec ↑	Rec ↑
DDGM	VLB	1000	7.6	26.1	10.5	54	55
DAED $\beta_1 = 0.1$	VLB	900	8.2	20.4	16.1	59	46
DAED $\beta_1 = 0.025$	VLB	979	7.7	22.4	15.8	57	53
DAED linear	VLB	999	8.1	14.5	9.8	60	59
DDGM	Simple	1000	9.5	7.2	8.6	65	61
DAED $\beta_1 = 0.2$	Simple	891	7.8	29.4	24.7	53	40
DAED $\beta_1 = 0.1$	Simple	900	8.0	19.0	14.9	62	50
DAED $\beta_1 = 0.025$	Simple	979	8.6	14.2	14.6	60	53
DAED $\beta_1 = 0.001$	Simple	999	9.1	14.9	10.1	66	54

Table 4: Extended evaluation results for CelebA dataset. Additionally to standard models, we also include evaluation for DAED setup where DAE model is trained only on ImageNet dataset.

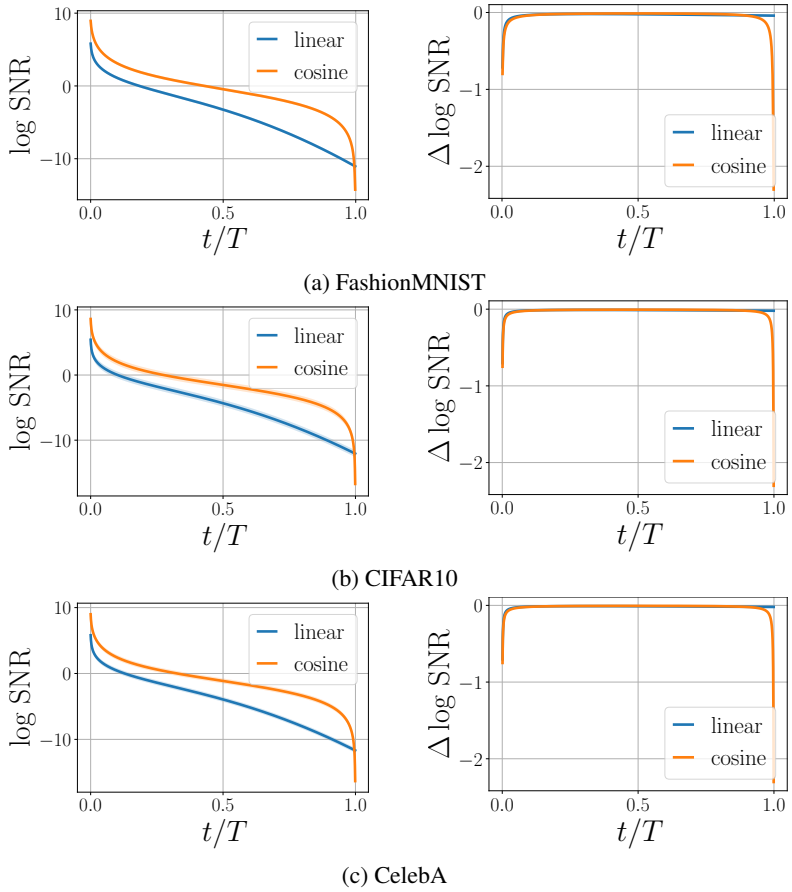
Model			CelebA				
	Loss	T	IS ↑	FID ↓	sFID ↓	Prec ↑	Rec ↑
DDGM	VLB	1000	2.4	23.1	37.3	51	21
DAED $\beta_1 = 0.1$	VLB	900	2.9	18.2	23.9	63	31
DAED $\beta_1 = 0.025$	VLB	979	2.7	25.4	35.8	64	17
DAED linear	VLB	1000	2.6	16.8	23.6	70	27
DDGM	Simple	1000	3.0	6.1	14.7	66	56
DAED $\beta_1 = 0.2$	Simple	890	2.7	21.0	31.2	63	22
DAED $\beta_1 = 0.1$	Simple	900	3.0	17.0	23.3	66	31
DAED $\beta_1 = 0.025$	Simple	979	2.7	15.1	17.6	64	38
DAED $\beta_1 = 0.001$	Simple	999	2.8	6.2	11.0	69	55
DAED (IN) $\beta_1 = 0.1$	Simple	900	2.9	25.6	30.5	44	29

Table 5: Extended evaluation results for Fashion MNIST dataset.

			Fashion Mnist				
	Loss	T	IS ↑	FID ↓	sFID ↓	Prec ↑	Rec ↑
DDGM	vlb	500	4.1	8.9	11	68	53
DAED $\beta_1 = 0.1$	vlb	468	4.06	9.1	13	71	60
DAED $\beta_1 = 0.025$	vlb	489	4.02	9.7	11	70	62
DAED linear	vlb	499	4.1	7.5	11.3	70.5	64
DDGM	Simple	500	4.3	7.8	9.03	71.5	65.3
DAED $\beta_1 = 0.3$	Simple	426	3.78	18	24	73.8	41
DAED $\beta_1 = 0.2$	Simple	445	3.87	14	20	74.8	47
DAED $\beta_1 = 0.1$	Simple	468	3.95	9.6	11.2	73.2	58.4
DAED $\beta_1 = 0.025$	Simple	489	4.05	7.36	13	73	61
DAED $\beta_1 = 0.001$	Simple	499	4.3	5.7	11.3	69.3	64.2

A.1 Signal-to-noise ratio detailed plots

In this section we present detailed signal-to-noise ratio (SNR) plots that are used for analysis in Sec. 3 for all evaluated datasets. Independently on the original dataset, SNR changes in the similar manner – with the most drastic loss in the first 10% steps.



(a) FashionMNIST

(b) CIFAR10

(c) CelebA

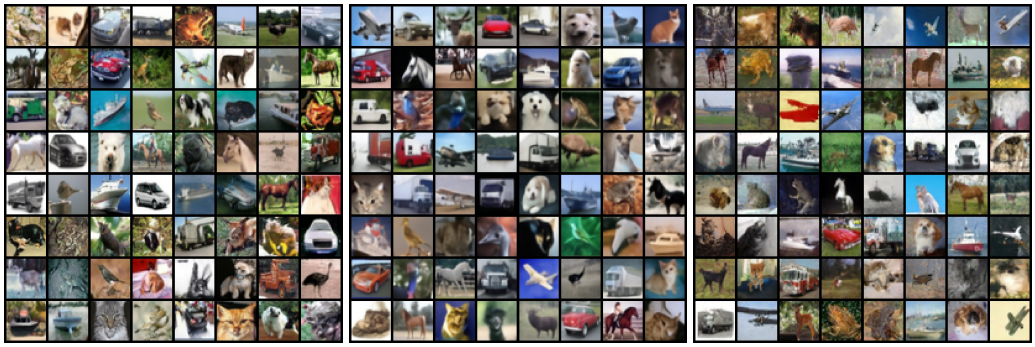
Figure 8: Signal-to-noise ratio and its discrete derivative for each of the three datasets: (a) Fashion-MNIST, (b) CIFAR10 and (c) CelebA).

A.2 Examples of generations

In this section we present generations for all datasets with different models we compare in this work.

(a) DDGM (b) DAED $\beta_1 = 0.1$ (c) DAED $\beta_1 = 0.001$

Figure 9: Generations from different models trained on FashionMNIST dataset. All models were trained with Simple loss function.



(a) DDGM (b) DAED $\beta_1 = 0.1$ (c) DAED $\beta_1 = 0.001$

Figure 10: Generations from different models trained on CIFAR10 dataset. All models were trained with Simple loss function.

(a) DDGM (b) DAED $\beta_1 = 0.1$ (c) DAED $\beta_1 = 0.001$

Figure 11: Generations from different models trained on CelebA dataset. All models were trained with Simple loss function.

(a) DDGM (b) DAED $\beta_1 = 0.1$ (c) DAED $\beta_1 = 0.001$

Figure 12: Generations from different models trained on CelebA dataset with original VLB loss function.

(a) DAED $\beta_1 = 0.1$

Figure 13: Generations from DAED model where DDGM part was trained on CelebA dataset while DAE on ImageNet.

A.3 Training Dynamics

How does the objective of a diffusion model change in time? In the standard DDGM setup, a single model is optimized with a joint loss from all of the diffusion steps. However, as depicted in Fig 15a, different parts of the diffusion contribute to the sum differently. In fact, the first step of the diffusion is already responsible for 75% of the whole training loss, while first 1% of steps contributes to over the 90% of the training objective. This observation implies that a single neural network applied to all diffusion steps is mostly optimized to denoise the initial steps. In Fig. 14 we present how this loss contribution changes over time. Surprisingly, only 2% of the training time is needed to align latter 90% of training steps to the loss value below 0.01. These observations led to the emergence of cosine scheduler [17] where authors change the noise scheduler to increase the number of steps with higher loss values.

In this work, we propose to tackle this problem from a different perspective and to analyze what happens if we detach the loss from initial diffusion steps from the total sum. In Figure 15b, we compare how such a detachment of the first step of 1000-stepped DDGM with DAED influence the loss value on the remaining 999 steps. As depicted in DAED, the loss converges to lower values that explains the improvement of the performance of DAED when training with the VLB loss.

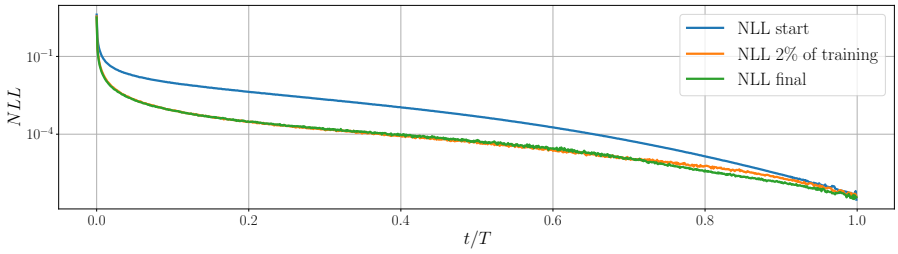
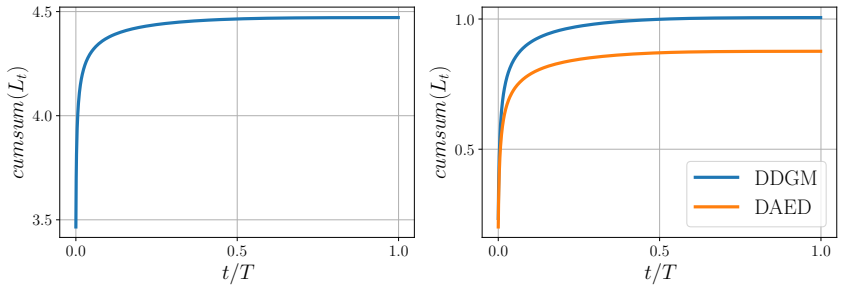


Figure 14: Dynamics of the negative log likelihood for different steps of standard DDGM trained on CIFAR10 with VLB objective. Already after 2% of training time, p_θ converges to very low loss values (below 0.001) for all of the training steps above 0.1T.



(a) NLL Cumsum

(b) NLL Cumsum without t_1

Figure 15: The cumulative sum of the negative log likelihood for different steps of a standard DDGM trained on CIFAR10 with the VLB objective (left), and the same cumulative sum without the first diffusion step in comparison to DAED with exactly the same β scheduler.

A.4 Training Hyperparameters

In all of our experiments, we follow [17]. We train all models with U-Net architecture, with three or four depth levels (depending on a dataset), with three residual blocks each, with a given number of filters depending on the dataset – as presented in 6. In all of our models, we use time embeddings and attention-based layers with three attention heads in each model.

We optimize our models on the basis of randomly selected diffusion steps. For the standard DDGM, for simplicity, we use a uniform sampler, while for DAED, we propose a weighted uniform sampler, where the probability of sampling from a given step t is proportional to the given β_t. This also applies to the Denoising Autoencoder as a part of DAED that is updated accordingly to the new sampler. We update models parameters with AdamW [15] optimizer for a given number of batches as presented in 6. To prevent our model from overfitting, we use dropout [7] with probability $p = 0.3$. Detailed implementation choices, examples of training runs and models can be found in the attached code repository.

Table 6: DDGM and DAED hyperparameters for different datasets

Dataset	train-steps	depth	channels
FashionMNIST	100k	3	64, 128, 128
CIFAR10	500k	3	128, 256, 256, 256
CelebA	200k	4	128, 256, 384, 512

A.5 Computational details

Diffusion-based deep generative models are known for being computationally expensive. For our training, we used Nvidia Titan RTX GPUs for complex datasets (CIFAR, CelebA, ImageNet) and Nvidia GeForce 1080Ti for FashionMNIST. Full training of our model on FashionMNIST for 100k steps on a single GPU took approximately 35 hours. For CIFAR and CelebA we used parallel computation based with four GPUs. Full training with this setup took approximately 48 hours. Those estimates are valid for training of both DDGM and DAED.

A.6 A comparison between DAED and DDGMs with more parameters

The DAED model uses two separate UNet models for the generative and denoising parts. As a result, it has twice as many parameters as a DDGM. In Table 7 we compare DEAD with DDGMs that have a comparable number of parameters. We double the size of the UNet model for vanilla DDGM in two setups. In the first one we increase the number of convolution channels, while in the second one, we double the number of residual blocks.

Table 7: A comparison of DAED with DDGMs of different sizes on the FashionMNIST dataset.

	Total Params (mln.)	Inference Time (sec. per sample)	FID ↓	Prec ↑	Rec ↑
DDGM	8.8	0.65	7.8	72	65
DDGM $1.5\times$ channels	19.8	0.84	8	74	65
DDGM $2\times$ blocks	15.1	1.19	7.5	66	66
DAED	17.6	0.66	5.7	69	64

The results in Table 7 suggest that the performance of DAED over DDGMs cannot be attributed purely to the larger number of parameters. As we increase the number of layers of the UNet used by the DDGM, we see only a slight improvement of the performance. Furthermore, a larger UNet leads to a significant increase in the inference time compared to the smaller DDGM and DAED.