

# Dynamic Dual-Output Diffusion Models

Yaniv Benny

Tel Aviv University

yanivbenny@mail.tau.ac.il

Lior Wolf

Tel Aviv University

wolf@cs.tau.ac.il

## Abstract

*Iterative denoising-based generation, also known as denoising diffusion models, has recently been shown to be comparable in quality to other classes of generative models, and even surpass them. Including, in particular, Generative Adversarial Networks, which are currently the state of the art in many sub-tasks of image generation. However, a major drawback of this method is that it requires hundreds of iterations to produce a competitive result. Recent works have proposed solutions that allow for faster generation with fewer iterations, but the image quality gradually deteriorates with increasingly fewer iterations being applied during generation. In this paper, we reveal some of the causes that affect the generation quality of diffusion models, especially when sampling with few iterations, and come up with a simple, yet effective, solution to mitigate them. We consider two opposite equations for the iterative denoising, the first predicts the applied noise, and the second predicts the image directly. Our solution takes the two options and learns to dynamically alternate between them through the denoising process. Our proposed solution is general and can be applied to any existing diffusion model. As we show, when applied to various SOTA architectures, our solution immediately improves their generation quality, with negligible added complexity and parameters. We experiment on multiple datasets and configurations and run an extensive ablation study to support these findings.*

## 1. Introduction

Over the past few years, deep generative models have reached the ability to generate high-quality samples in various domains, including images [2], speech [25], and natural language [3]. For image generation, generative models can be divided into two main branches: approaches based on generative adversarial networks (GAN) [7] and log-likelihood-based methods, such as variational autoencoders (VAE) [15], autoregressive models [26], and normalizing flows [14, 29]. Log-likelihood models have the advantage of possessing a straightforward objective, which makes

them easier to optimize, while GANs are known to be unstable during training [8, 31]. However, until recently, well optimized GAN models outperformed their log-likelihood counterparts in generation quality [2, 11–13].

This changed when Ho et al. [9] introduced a new type of log-likelihood model called the Denoising Diffusion Probabilistic Model (DDPM). With this model, image quality surpasses GANs [6], while it is also very stable and easy to train. DDPMs follow the concept of iterative denoising: given a noisy image  $x_t$ , it is gradually denoised by predicting a less noisy image  $x_{t-1}$ . This process, when done over hundreds (or thousands) of iterations, is able to generate images with very high quality and diversity, even when starting from random noise. DDPMs have many computer vision applications, such as super-resolution [18, 30] and image translation [33], and are also extremely effective in non-visual domains [4, 21, 28].

DDPM incorporates a probabilistic denoising process that is dependant on the estimation of the mean component  $\mu_{t-1}$ . This is done by a neural network parameterized over  $\theta$  and denoted as  $\mu_\theta(x_t, t)$ . However, it was found that through the forward and backward equations this process is better formalized by predicting either the noise  $\epsilon_\theta(x_t, t)$  or the original image  $x_\theta(x_t, t)$  [9]. Their experiments found the former to be empirically superior, and, as far as we can ascertain, no further comparisons between the two options (noise or original image) have been performed as yet.

In this work, we revisit the original implementation of DDPM, and find that the preference of  $\epsilon_\theta$  over  $x_\theta$  is circumstantial and depends on the hyperparameters and datasets. In addition, in certain timesteps, the denoising process has less error when predicting the noise component  $\epsilon_\theta$ , while in others it predicts the original image  $x_\theta$  better. This realization motivated us to design a model capable of predicting both values and adaptively selecting the more reliable output at each sampling iteration. The modified model has a negligible number of added parameters and complexity. We apply this method to various DDPM models and show a marked improvement in terms of image quality (measured by FID) for many benchmarks. This addition to the framework is orthogonal to existing advancements (that we know

of), and is able to improve sampling quality, especially with the restriction of few iterations.

## 2. Related work

Diffusion probabilistic models were introduced by Sold-Dickstein et al. [34], who proposed a model that can learn to reverse a gradual noising schedule. This framework is part of long research on generative models that are based on Markov chains [1, 32], that has led to the development of Noise Conditional Score Networks (NCSN) [36, 37] and Denoising Diffusion Probabilistic Models (DDPM) [9]. Although very similar, DDPMs try to minimize the log-likelihood, while NCSNs optimize the matching objective [10].

The success of DDPMs has sparked a lot of interest in improving upon the original design. Song et al. proposed an implicit sampling (DDIM) [35] that reduces the number of iterations while maintaining high image quality. Nichol and Dhariwal [23] proposed a cosine noising schedule and a learned denoising variance factor, and in a second work [6] proposed architectural improvements and classifier guidance. Watson et al. [38] proposed a dynamic programming algorithm to find an efficient denoising schedule. Nachmani et al. [22] applied a Gamma distribution instead of Gaussian. Luhman and Luhman [20] applied knowledge distillation with DDPMs.

The solution proposed in this work is orthogonal to the contribution of these methods. It is, therefore, possible to apply our method to the above advancements and increase the performance of all these networks. This is demonstrated in our experiments for some of the existing approaches.

## 3. Setup

Diffusion models operate as the reversal of a gradual noising process. Given a sample  $x_0$ , we consider the samples  $x_t$  for  $t \in [1, T]$  obtained by gradually adding noise, starting from  $x_0$ . Noise is applied in such a way that each instance is noisier than the previous, and the final instance  $x_T$  is completely destroyed and can be seen as a sample from a predefined noise distribution. Ho et al. [9] proposed a Gaussian noise that is applied iteratively as:

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t \mathbf{I}) \quad (1)$$

Here,  $\beta_t \in [0, 1]$  for  $t \in [1, T]$  are a group of scalars selected so that  $x_T \sim \mathcal{N}(x_T; \mathbf{0}, \mathbf{I})$  (a multivariate i.i.d normal distribution). Due to the choice of applying Gaussian noise, a simpler transition can be applied directly from  $x_0$  to any  $x_t$ , which makes training much more efficient. Using  $a_t := 1 - \beta_t$  and  $\bar{\alpha}_t := \prod_{s=1}^t a_s$ , we get:

$$q(x_t|x_0) := \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}) \quad (2)$$

This formulation also reveals a simpler constraint, which is that  $\bar{\alpha}_T = \prod_{s=1}^T (1 - \beta_s) \approx 0$ , and multiple such schedules have been proposed [9, 23] and tested.

Through this equation, any intermediate step  $x_t$  can be sampled, given a noise sample  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ :

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad (3)$$

Notice that  $x_0$  can easily be backtraced through

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1 - \bar{\alpha}_t}\epsilon), \quad (4)$$

which is an important property for the denoising process.

The “reversed” denoising process is then a Markovian process, parameterized with a neural network over  $\theta$  as:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (5)$$

Training this model is done by sampling a random  $t \in [1, T]$  and minimizing the loss  $L_t$  with stochastic gradient descent.  $L_t$  is the KL-divergence:

$$L_t = D_{\text{KL}}(q(x_{t-1}|x_t) \| p_\theta(x_{t-1}|x_t)) \quad (6)$$

Some critical modifications are responsible for the stabilization of this objective. The high variance distribution  $q(x_{t-1}|x_t)$  is replaced with the more stable  $q(x_{t-1}|x_t, x_0)$ , which is practically the combination of the posterior  $q(x_{t-1}|x_t)$  and the “forward” process  $q(x_{t-1}|x_0)$ .

$$q(x_{t-1}|x_t, x_0) := \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0, t), \tilde{\beta}_t \mathbf{I}) \quad (7)$$

$$\tilde{\mu}_t(x_t, x_0, t) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 + \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}x_t \quad (8)$$

$$\tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t \quad (9)$$

As for  $p_\theta(x_{t-1}|x_t)$ , Ho et al. [9] found that fixing  $\Sigma_t$  to a constant  $\sigma_t^2$  makes it easier to optimize an objective that is reduced to predicting the mean vector  $\tilde{\mu}_t$  only:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 \mathbf{I}) \quad (10)$$

As a corollary, the loss function becomes:

$$L_t := \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(x_t, x_0, t) - \mu_\theta(x_t, t)\|^2 \quad (11)$$

The constant  $\sigma_t$  was selected to be  $\beta_t$ , even though there is also a theoretical explanation for choosing  $\beta_t$  instead.

In addition, the prediction of  $\mu_\theta$  directly was replaced by either ① the prediction of  $x_0$ , denoted as  $x_\theta$ , and computing  $\mu_\theta$  (denoted as  $\mu_x(x_\theta)$ ) through Eq. 8, or ② predicting  $\epsilon$ , denoted as  $\epsilon_\theta$ , and using Eq. 4, 8 ( $\mu_\epsilon(\epsilon_\theta)$ ).

$$\textcircled{1} \quad \mu_x(x_\theta) := \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_\theta \quad (12)$$

$$\textcircled{2} \quad \mu_\epsilon(\epsilon_\theta) := \frac{1}{\sqrt{\bar{\alpha}_t}}x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}\sqrt{\bar{\alpha}_t}}\epsilon_\theta \quad (13)$$

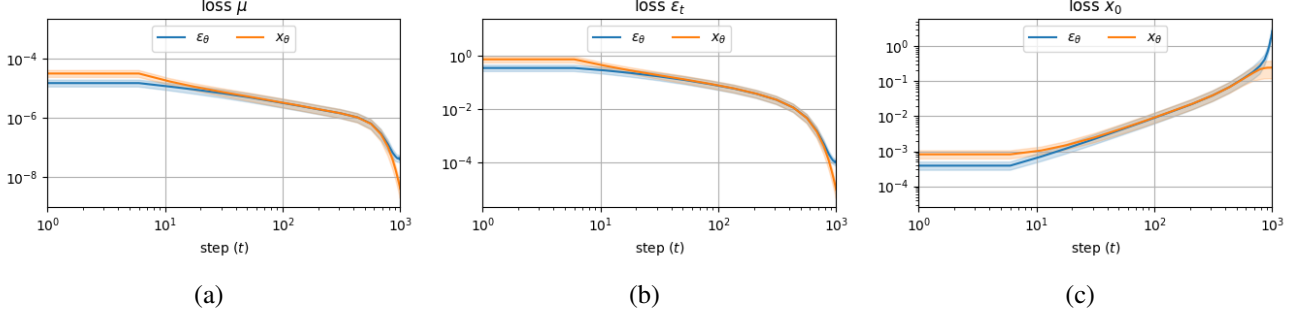


Figure 1. **Loss comparison between  $\epsilon_\theta$  and  $x_\theta$ .** (a) Loss on predicting  $\tilde{\mu}_t$ , (b) loss on predicting  $\epsilon_t$ , (c) loss on predicting  $x_0$ .

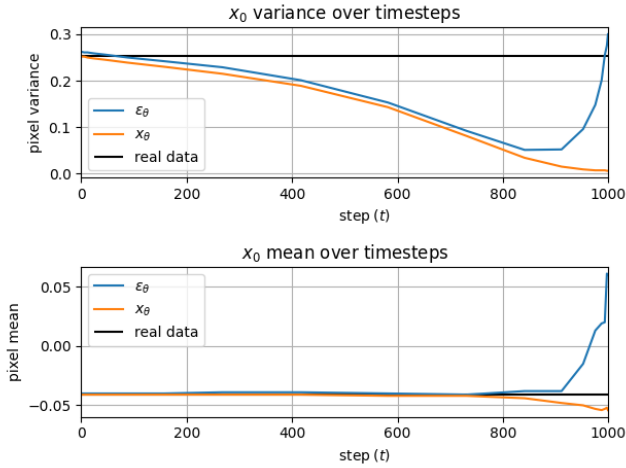


Figure 2. Pixel mean and variance of predicted  $x_0$  over timesteps, for both the subtractive ( $\epsilon_\theta$ ) and the additive ( $x_\theta$ ) paths. For a model trained on CIFAR10. Real data statistics are in black.

The latter was chosen based on empirical evidence, and by further developing the equations this choice resulted in a new formalization of  $L_t$  as:

$$L_t = M_t \|\epsilon - \underbrace{\epsilon_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon, t)}_{x_t}\|^2, \quad (14)$$

where  $M_t$  is a weight that should be equal to  $\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\alpha_t)}$  for consistency with Eq. 11, but was set to 1 for simplicity.

### 3.1. Pros and cons for predicting $\epsilon$

There are multiple justifications for why the backward process should be driven by predicting the noise  $\epsilon$ . The first is that the noise  $\epsilon$  has always zero mean and unit variance, and the model can learn these statistics quite easily. A second is that it gives a residual-like equation, where image  $x_0$  is predicted by subtracting the output of the model from the input (Eq. 4). This provides the model with the option to preserve the information in the input, by predicting zero

noise or multiply it by a small  $\sqrt{1-\alpha_t}$ . This approach becomes increasingly beneficial towards the end of the denoising process, where the amount of noise becomes small, and only minor modifications are needed.

The main disadvantage of this approach is that after the subtraction of noise from  $x_t$ , the result is scaled with  $\sqrt{\alpha_t}$  (Eq. 4), which can be a very small value for some steps (large  $t$ ). This can lead to a very large error even for a small error in  $\epsilon_\theta$ . This error propagates, since the model is limited to modifying the intermediate states with something that resembles noise, and if previous iterations produced a state  $x_{t-1}$  that is not viable, it becomes difficult for the model to correct this path. In such cases, multiple iterations may be required just to revert a previous bad prediction.

This problem is demonstrated in Fig. 1,2. Fig. 1 shows that loss when using  $\epsilon_\theta$  is significantly larger at high  $t$  than using a direct  $x_\theta$  approach. Note that Fig. 1 is in log-scale, and the error of  $\epsilon_\theta$  is larger by orders of magnitude for hundreds of steps. Fig. 2 shows the mean and variance of the predicted  $x_0$  through the denoising process. As can be seen, the prediction of  $x_0$  using  $\epsilon_\theta$  starts with very high bias and variance, and it takes multiple steps to correct this. In contrast, the direct prediction using  $x_\theta$  immediately starts with very low bias, and its variance increases monotonically with the real data variance.

### 3.2. Pros and cons for predicting $x_0$

An advantage and also disadvantage of predicting  $x_0$  directly is that the model needs to produce the entire image, not just subtract some noise from its input. This can be an advantage during the first stages when the input image is very noisy. As can be seen in Fig. 2, it is easier to predict an unbiased estimate of the image directly than to do so by subtracting noise, and the loss during these steps is substantially lower than  $\epsilon_\theta$  (Fig. 1). It becomes a disadvantage during later steps, by when a substantial structure has been formed in  $x_t$ , and the direct prediction of  $x_0$  needs to rebuild it in each step, instead of simply subtracting small noise artifacts. Fig 1 shows that all predictions are less ac-

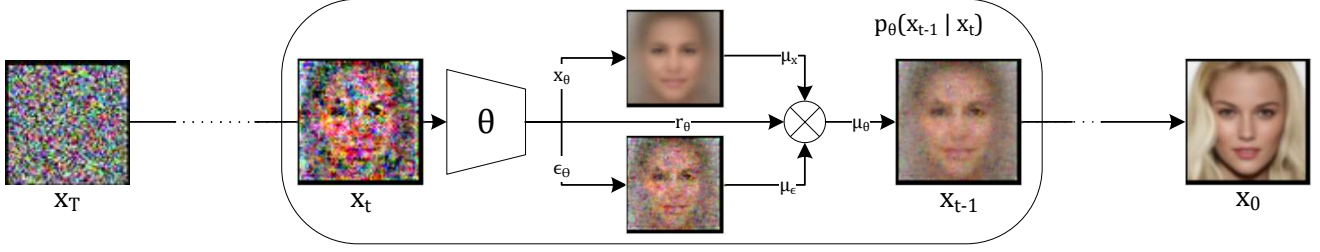


Figure 3. **The dual output diffusion model.** A noisy image  $x_T$  is gradually denoised into  $x_0$ . In each iteration, an intermediate state  $x_t$  is inserted into a model  $f_\theta(x_t, t)$  that predicts  $x_\theta, \epsilon_\theta, r_\theta$ . These outputs are combined into a mean vector  $\mu_\theta$ , that is subsequently used to sample the next state  $x_{t-1}$ .

curate for small  $t$  when using  $x_\theta$ .

At first glance, it appears that the backward process using  $x_\theta$  loses the residual-like property of  $\epsilon_\theta$ , since the image  $x_0$  is estimated directly and not subtracted from  $x_t$ . However, the objective of step  $t$  is not to predict  $x_0$ , but to obtain  $\tilde{\mu}_t$ . According to Eq. 8, the residual property is still present in this alternative backward process, which relies on  $x_\theta$ .

Note that while  $\epsilon_\theta$  is subtracted from  $x_t$  (Eq. 13),  $x_\theta$  is added to it (Eq. 12). Therefore, we distinguish between the processes by calling the  $\epsilon_\theta$  process the “subtractive” backward process, and calling the  $x_\theta$  process the “additive” backward process.

## 4. Method

To leverage the advantages of both flows, we can consider two models. The first,  $f_\phi(x_t, t)$ , predicts  $\epsilon_\phi$  (and  $x_\phi$  through Eq. 4), and the second,  $f_\psi(x_t, t)$ , predicts  $x_\psi$ . Each of them can estimate its own  $\tilde{\mu}_t$  (Eq. 12, 13), but in order to control how much we want to rely on each model’s output, we can interpolate between their estimates with an additional parameter  $r_t$ , as  $r_t \cdot \mu_\psi + (1 - r_t) \cdot \mu_\phi$ . By selecting a different value for  $r_t$  for each step  $t$ , we can control how much influence we want each path to have on each step.

To simplify and generalize this solution, we fuse  $f_\phi, f_\psi$  into one model  $f_\theta$ , and make the interpolation parameter  $r_t$  learned as well ( $r_\theta$ ). The generalized model  $f_\theta$  computes:

$$\epsilon_\theta, x_\theta, r_\theta = f_\theta(x_t, t) \quad (15)$$

$$\mu_\theta = r_\theta \cdot \mu_x(x_\theta) + (1 - r_\theta) \cdot \mu_\epsilon(\epsilon_\theta) \quad (16)$$

An illustration can be seen in Fig. 3. The modifications required to go from a model that only predicts  $\epsilon_\theta$  to our new model might seem complex, but they are in fact very simple. The only change to the model is in the number of output channels in the last layer. For example, for  $x_0, \epsilon \in \mathbb{R}^{H,W,C}$  and  $r \in \mathbb{R}^{H,W,1}$ , the output of  $f_\theta$  changes from  $\mathbb{R}^{H,W,C}$  to  $\mathbb{R}^{H,W,2 \cdot C+1}$ . This means that the number of added parameters should be negligible. Complexity and runtime are also unaffected since the computation of  $\mu_\theta$  is negligible.

This new model requires a new loss function  $L_t$ , which optimizes  $\epsilon_\theta, x_\theta$ , and  $r_\theta$ . We separate this into three components:

$$L_t^\epsilon = \|\epsilon - \epsilon_\theta\|^2 \quad (17)$$

$$L_t^x = \|x_0 - x_\theta\|^2 \quad (18)$$

$$L_t^\mu = \|\tilde{\mu}_t - (r_\theta[\mu_x(x_\theta)]_{\text{sg}} + (1 - r_\theta)[\mu_\epsilon(\epsilon_\theta)]_{\text{sg}})\|^2 \quad (19)$$

$$L_t = \lambda_t^\epsilon L_t^\epsilon + \lambda_t^x L_t^x + \lambda_t^\mu L_t^\mu, \quad (20)$$

where  $[\cdot]_{\text{sg}}$  denotes “stop-grad”, which means that inner values are detached and no gradient propagated back from them. The  $\lambda$ ’s are weights that can be applied to each loss, which we kept as 1 throughout our experiments. We found that optimizing with these stop-grads results in a much more stable training regime than the alternative of allowing gradients to propagate through  $\mu_x, \mu_\epsilon$ , because the gradients of  $\frac{\partial \mu_x}{\partial x_\theta}, \frac{\partial \mu_\epsilon}{\partial \epsilon_\theta}$  are subject to intense rescaling (see Eq. 12, 13).

### 4.1. Implicit sampling

Song et al. [35] proposed an implicit sampling method (DDIM) that is deterministic after generating the first seed  $x_T$ . Since our method only changes how  $\mu_\theta$  is estimated, it does not affect the ability to perform implicit sampling. Their generalized formula is as follows:

$$q(x_t | x_{t-1}, x_0) := \mathcal{N}(\mu_t, \sigma_t^2 \mathbf{I}) \quad (21)$$

$$\mu_t := \sqrt{\bar{\alpha}_{t-1}} x_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \epsilon_t \quad (22)$$

In [35],  $x_0, \epsilon_t$  were estimated with  $\hat{x}_0 = x_0(\epsilon_\theta)$  (Eq. 4),  $\hat{\epsilon}_t = \epsilon_\theta$ . Our method uses the interpolated estimation of:

$$\mu_{tx} = \sqrt{\bar{\alpha}_{t-1}} x_\theta + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \underbrace{\frac{x_t - \sqrt{\bar{\alpha}_t} x_\theta}{\sqrt{1 - \bar{\alpha}_t}}}_{\hat{\epsilon}_t} \quad (23)$$

$$\mu_{t\epsilon} = \underbrace{\frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta}{\sqrt{\bar{\alpha}_t}}}_{\sqrt{\bar{\alpha}_{t-1}} \hat{x}_0} + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \epsilon_\theta \quad (24)$$



When  $\sigma_t = 0$ , deterministic behavior is obtained, and the model is not sensitive to the value of  $\sigma_t^2$  of the probabilistic approach that would have been selected in Eq. 10.

An additional advantage of DDIM is that it was shown to be able to generate exceptionally well with far fewer iterations than the probabilistic sampling of [9]. For these reasons, our experiments will follow mostly this approach.

## 5. Experiments

We start by showcasing our method’s generation results from each independent output  $\epsilon_\theta$ ,  $x_\theta$ , and show how the interpolation parameter  $r_\theta$  affects the generation process. We then evaluate our model against existing state-of-the-art methods on multiple datasets, including CIFAR10 [16], CelebA [19], and ImageNet [5], and perform ablation evaluations with each experiment.

### 5.1. Dual-Output Denoising

As presented in Sec. 4, our proposed solution consists of a dual output model. One head predicts the noise  $\epsilon_\theta$  while the second predicts the image  $x_\theta$ , and a third head,  $r_\theta$ , efficiently balances the two options. To understand how each method affects the iterative process, we visualize their intermediate results during each denoising process. For  $x_\theta$ , the intermediate result is simply the predicted image. For  $\epsilon_\theta$ , it is following Eq. 4.

These results can be seen in Fig. 4. Evidently, the two outputs produce two very different iterative processes, which to some extent act as opposites. The denoising process that uses  $\epsilon_\theta$ , produces a very noisy start and gradually removes noise from its previous estimates. In contrast, the process that relies on  $x_\theta$  starts with a very blurry image, which resembles an average of many images, and iteratively adds content to it. These two different sequences inevitably result in two different final images, but it appears that the initial seed  $x_T$  is a strong enough condition to guide them both in a similar direction in the image space.

In the bottom row of each grid in Fig. 4, we visualize the dual-output denoising process, driven by the interpolation parameter  $r_\theta$ .

Evidently, the interpolation magnitude in each step is dataset dependent. For example, the dual-output process in CIFAR10 starts very similarly to  $x_\theta$ , while that of CelebA is mixed. It can also be observed that the dual-output result is different from either of the two options. In CIFAR10, it can be observed that the dual-output produces less noisy images than  $\epsilon_\theta$ , and sharper than  $x_\theta$ . For CelebA, we noticed that the dual-output image quality is higher. For example, pieces of hair are more refined and glasses are noticeable.

To better understand the denoising process with each output, we perform an additional experiment, where the method is switched between subtractive and additive at

some point in the middle of the process. Fig. 5 depicts multiple sequences, where the model starts with  $x_\theta$  and at some point continues the task with  $\epsilon_\theta$  (this order is more natural than starting with  $\epsilon_\theta$  and switching to  $x_\theta$ , see Sec. 3.1). In this figure, the steps surrounded by red boxes are the result of progressing using  $\epsilon_\theta$ , while steps marked in blue are intermediate results of using  $x_\theta$ . The top row uses only  $\epsilon_\theta$  and the bottom row only  $x_\theta$ . We again add the sequence produced by the interpolated results. In this example, generation using  $x_\theta$  failed to produce a pleasing image, and the other option was superior. However, it seems that some mixture of the two yields the best result. This is the motivation behind our adaptive interpolation, which allows the model to choose dynamically how to proceed.

A valid question would be “how much better is a dynamic interpolation parameter than learning a constant  $r_t$  for each step  $t \in [1, T]$ ?”. To answer this, we measure  $r_\theta$  at each step for multiple denoising processes. In Fig. 6, we show the average value of  $r_\theta$  at each step  $t$ . The two plots show the average value in black, with the grey region marking the dynamic range of the parameter. We also show the interpolation value for 16 different trajectories in blue.

This visualization shows that there is a large variability in the trajectory that the interpolation values take. Moreover, when it comes to a particular generation process, our method usually prefers a different value than the overall average. We also evaluate image quality using a fixed  $r_t$  in Sec. 5.2, and compare it to the dynamic  $r_\theta$ .

Interestingly,  $r_\theta$  seems to behave differently for each dataset. In CIFAR10, interpolation is more clear-cut. It starts with a very high preference towards  $x_\theta$ , and somewhere around the middle of the process starts to drop fast towards  $\epsilon_\theta$ . On CelebA,  $r_\theta$  begins at around 0.5 and maintains a relatively narrow dynamic range. Nevertheless, it also drops fast towards  $\epsilon_\theta$  in the second half of the process. In both datasets, the model finished with a very high preference for the subtractive process.

### 5.2. Image Quality

We conduct image quality evaluations on multiple datasets and baselines. In all evaluations, we use the implicit sampling formula proposed by Song et al. [35], in order to maintain high quality with low iteration count. In each evaluation, we specify the number of denoising iterations. Timesteps were respaced uniformly, following [35].

We evaluate by generating 50K images for each dataset, which are then compared with the full training set for CIFAR10 and CelebA and the validation set for ImageNet. We evaluate the models on the basis of image quality measured with FID [8]. FID is known to be sensitive for even slightly different preprocessing and methodology [27]. For reproducibility and reliability, we use the torch-fidelity [24] library. For ImageNet, we also measure “improved preci-

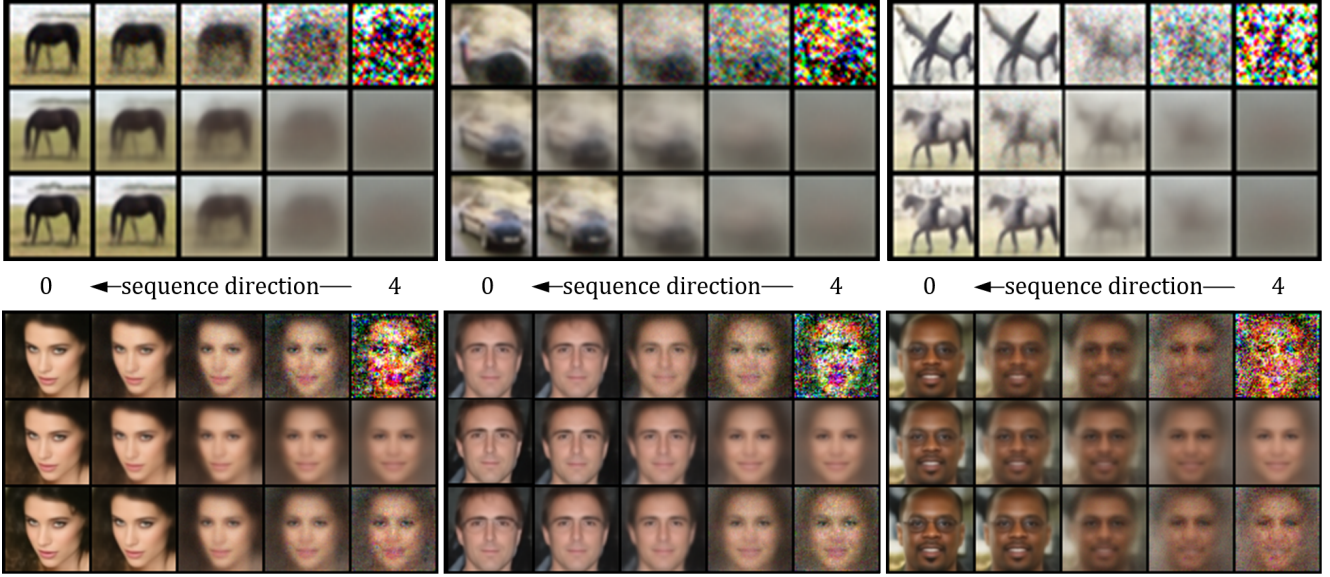


Figure 4. **Progressive generation in 5 steps.** From top to bottom: ① Prediction using  $\epsilon_\theta$  (subtractive), ② prediction using  $x_\theta$  (additive), ③ our dual-output. The images generated with the dual-output method are overall cleaner and sharper.

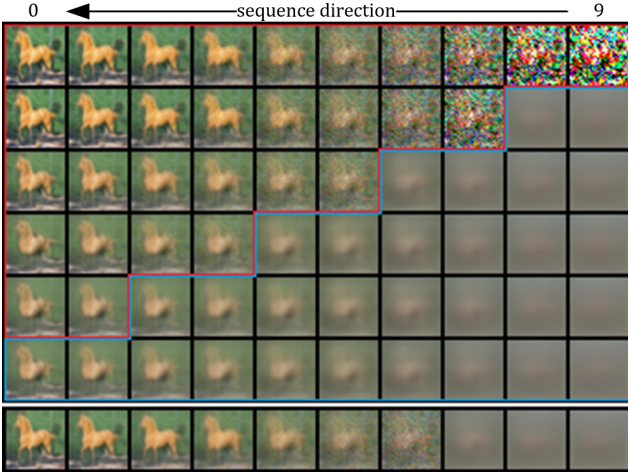


Figure 5. **Progressive generation in 10 steps.** Each row is a different generation sequence from the same initial noise, with the intermediate results visualized. Steps marked by red are produced with the  $\epsilon_\theta$  output and blue are produced with  $x_\theta$ . Thus, the sequences in the middle rows start with  $\epsilon_\theta$  and switch to  $x_\theta$  at some point. The sequence at the bottom represents our dynamic dual-output technique.

sion and recall” [17] over VGG feature manifolds between 10K real images and 50K generated images with  $k=3$ .

We compare our model to official pretrained models of DDPM [9], DDIM [35], IDDP [23], and ADM [6]. We also included IDDP with implicit sampling (IDDIM), which uses the same official pretrained model, but applies the implicit backward process. Since DDPM and IDDP

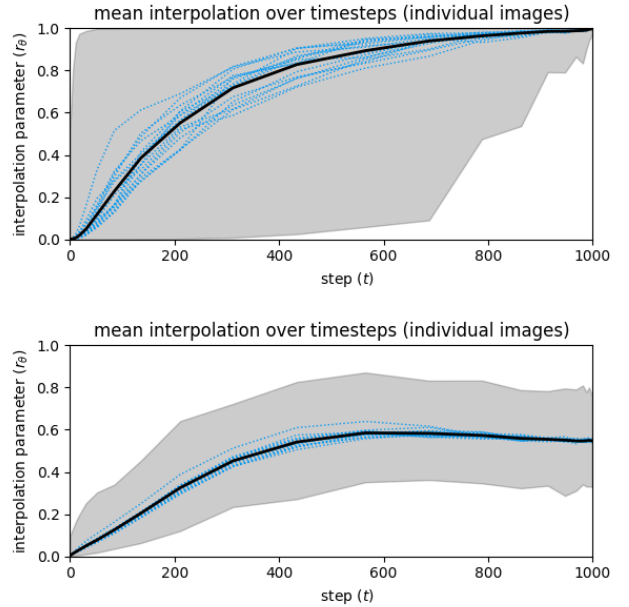


Figure 6. Mean value for the interpolation parameter  $r_\theta$  over the generation steps. For our model trained on CIFAR10 (top) and CelebA (bottom).

enforce a different noising schedule (linear and cosine, respectively), we separate them and compare the models that were trained under equal conditions. DDIM used the pre-trained model of DDPM for CIFAR10, but trained a new model for CelebA.



Figure 7. **Generation on CIFAR10.** Comparison between similar images. (a) 5, (b) 10, and (c) 20 steps. **Top: DDIM, Bottom: Ours.**

Each comparison to a baseline involves modifying the architecture and loss function to suit our method, and then train the model using the same hyperparameters. Training of each model was performed on 4 NVIDIA RTX 2080 TI GPUS in a distributed fashion. CIFAR10 and CelebA were trained from scratch. On ImageNet, this was not feasible, since the baseline (ADM) took 4.36 million iterations on extremely high-end devices. Instead, we loaded the encoded with the pretrained weights of their model and trained the rest of the model (decoder and residual block) for 80 thousand iterations.

### CIFAR10 and CelebA

In Tab. 1, we show the results of evaluation on CIFAR10 and CelebA. We perform the evaluations with 5, 10, 20, 50, and 100 iterations. As can be seen, our method outperforms the baselines on all metrics and under all resampling conditions, except for IDDPM with 100 iterations. The image quality inevitably declines with the reduction of denoising iterations, but our method maintains a significantly lower FID than the equivalent baselines.

When comparing to the ablation experiments, it can be observed that using a fixed  $r_t$ , which was taken to be the mean value as in Fig. 6, worse performance is obtained. The results for the additive and the subtractive paths reveal that no single path is always better than the other. CIFAR10 with linear schedule measured a lower FID with  $\epsilon_\theta$ , but the cosine schedule and CelebA did better with  $x_\theta$ .

For a visual comparison, we show generated images of our method alongside DDIM, for 5, 10, and 20 steps, see Fig. 7. The images were not cherry-picked, but we did manually select samples in DDIM and our method, that looked relatively similar. For each image in our method, we show the most similar image from 100 generated images in DDIM. It can be seen that our method generated better-looking, more detailed and sharper images. It is also evident that more steps produce higher quality results.

		# iterations				
		Method	5	10	20	50 100
CIFAR10 (32×32)	Linear	DDPM <sup>†</sup> [9]	196.54	160.18	145.45	65.43 32.65
		DDIM <sup>†</sup> [35]	49.70	18.57	10.87	7.03 5.57
		<b>ours</b>	<b>35.12</b>	<b>11.68</b>	<b>8.62</b>	<b>6.68</b> <b>5.54</b>
		- fixed $r_t$	38.50	12.08	8.71	6.89 5.57
		- only $\epsilon_\theta$	41.99	12.30	8.74	7.11 6.01
		- only $x_\theta$	45.53	24.27	16.93	12.47 7.39
	Cosine	IDDPM <sup>‡</sup> [23]	<b>X</b>	29.10	13.33	5.73 <b>4.58</b>
		IDDIM <sup>‡</sup> [23]	<b>X</b>	38.14	19.68	8.98 6.29
		<b>ours</b>	<b>X</b>	<b>18.25</b>	<b>12.54</b>	<b>5.59</b> 5.10
		- fixed $r_t$	<b>X</b>	19.60	13.93	7.24 6.17
		- only $\epsilon_\theta$	<b>X</b>	36.78	17.08	8.85 6.72
		- only $x_\theta$	<b>45.75</b>	19.75	13.21	7.13 5.93
CelebA (64×64)	Linear	DDPM <sup>†</sup> [9]	304.89	278.31	160.67	88.74 43.90
		DDIM <sup>†</sup> [35]	56.16	16.90	13.38	8.80 6.15
		<b>ours</b>	<b>26.22</b>	<b>14.96</b>	<b>8.74</b>	<b>5.54</b> <b>4.07</b>
		- fixed $r_t$	32.64	16.19	8.85	6.20 4.44
		- only $\epsilon_\theta$	64.82	27.53	12.64	9.03 8.68
		- only $x_\theta$	29.79	16.03	9.18	6.57 4.23

Official pretrained models by [9]<sup>†</sup>, [23]<sup>‡</sup>, and [35]<sup>†</sup>.

Table 1. **FID on CIFAR10 and CelebA.** Results are separated by the applied noising schedule “linear/cosine”. **X** marks unstable conditions that produced NaNs; due to dividing by a very low  $\bar{\alpha}$ .

### ImageNet

Fig. 8 shows generated results with our model on ImageNet, for different images generated with the same initial noise  $x_T$ , but different denoising paths ( $\epsilon_\theta$ , “dual”, and  $x_\theta$  respectively). To qualitatively compare the images, we performed a user study, where the subjects were asked to select the most visually convincing image from the three options. Among 25 participants, our images were selected 78% of





Figure 8. **Image generation on ImageNet.** Comparison of generated results for different paths on the same initial noise  $x_T$ .

the time, followed by 17%  $x_\theta$ , and 5%  $\epsilon_\theta$ .

Tab. 2 shows our evaluation on conditional ImageNet with  $128 \times 128$  resolution. Since we did not train our model for nearly as long as the baseline, we do not compare our results to theirs, but only add them as reference. Our evaluation is focused on comparing the subtractive ( $\epsilon_\theta$ ) and the additive ( $x_\theta$ ) paths to the dual-output solution. In here,  $\epsilon_\theta$  can act as a representative for the baseline, as that is the baseline’s method of choice.

The evaluation was performed on 25 and 50 denoising iterations, with and without the classifier guidance proposed by the baseline [6]. All evaluations were performed by generating 50 images per class (50K images in total), and comparing them to 50K validation images for FID and 10K images for precision and recall. The results show that the dual-output outperforms each of the alternative paths on all three metrics. Also, again we observed that the results of  $x_\theta$  were superior to  $\epsilon_\theta$ , which shows that the advantage of the subtractive path is circumstantial. Finally, while there is a considerable gap between our results and the ADM baseline, this evaluation solidifies our speculation about the dual-output process, and suggests that with enough training resources, could surpass the baseline.

## 6. Discussion and limitations

While we are able to select an effective value for  $r_\theta$  by considering the next-step measure derived from the loss in Eq. 19, this does not necessarily lead to optimal image quality at the end of the generation process. While one can intuitively expect such a greedy approach to be close to optimal, this requires validation. If the greedy approach turns out to be significantly suboptimal, a beam search approach may be able to improve image quality further.

From the societal perspective, the study of diffusion models has two immediate negative outcomes: environmental and harmful use. The environmental footprint of training high-resource neural networks is becoming a major concern. Our work enables the reduction of the number of iterations required to achieve a certain level of visual quality,

		# iterations					
		25			50		
Method (+train steps)		FID	PR	RC	FID	PR	RC
— no classifier guidance —							
ADM [6]	(4.36M)	11.7	0.92	0.14	7.6	0.92	0.21
<b>dual</b>	(*80K)	<b>27.7</b>	<b>0.90</b>	<b>0.11</b>	<b>25.3</b>	<b>0.89</b>	<b>0.15</b>
- only $\epsilon_\theta$	(*80K)	51.3	0.89	0.08	49.1	0.86	0.09
- only $x_\theta$	(*80K)	29.5	0.90	0.08	27.4	0.88	0.12
— classifier scale 1.0 —							
ADM [6]	(4.36M)	10.2	0.95	0.09	7.1	0.96	0.16
<b>dual</b>	(*80K)	<b>24.5</b>	<b>0.94</b>	<b>0.08</b>	<b>22.1</b>	<b>0.92</b>	<b>0.12</b>
- only $\epsilon_\theta$	(*80K)	44.1	0.93	0.07	36.8	0.89	0.07
- only $x_\theta$	(*80K)	26.0	0.93	0.07	24.7	0.90	0.10

Table 2. **Generation evaluation on ImageNet  $128 \times 128$ .** With and without classifier guidance. Measuring FID, precision, and recall. \* Using pretrained encoder from ADM.

thus lowering their computation cost. In addition, our experiments are done at a relatively modest energy cost, especially since we opted to train the ImageNet models only in part. The second concern is the ability to generate realistic fake media with generative methods. Our hope is that open academic study of generative models will raise public awareness of the associated risks and enable the development of methods for identifying fake images and audio.

## 7. Conclusions

When applying diffusion models, one can choose to transition to the next step by estimating either the slightly improved image after applying the current step or by estimating the target image. As we show, the accuracy of each of the two, depends on the exact stage of the inference process. Moreover, the ideal trajectory varies depending on the specific sample, and for most of the process, mixing the two estimates for the next step provides a better results.

## Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant ERC CoG 725974). The contribution of the first author is part of a PhD thesis research conducted at Tel Aviv University.

## References

- [1] Yoshua Bengio, Eric Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, pages 226–234. PMLR, 2014. 2
- [2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018. 1
- [3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1
- [4] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020. 1
- [5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5
- [6] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. *arXiv preprint arXiv:2105.05233*, 2021. 1, 2, 6, 8, 11
- [7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 1
- [8] 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. 1, 5
- [9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020. 1, 2, 5, 6, 7, 11
- [10] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 2
- [11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 1
- [12] 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. 1
- [13] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 1
- [14] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018. 1
- [15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1
- [16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. URL: <https://www.cs.toronto.edu/kriz/cifar.html>, 6(1):1, 2009. 5
- [17] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Conference on Neural Information Processing Systems*. Neural Information Processing Systems Foundation, 2019. 6
- [18] Haoying Li, Yifan Yang, Meng Chang, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *arXiv preprint arXiv:2104.14951*, 2021. 1
- [19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 5
- [20] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 2
- [21] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021. 1
- [22] Eliya Nachmani, Robin San Roman, and Lior Wolf. Denoising diffusion gamma models. *arXiv preprint arXiv:2110.05948*, 2021. 2
- [23] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021. 2, 6, 7, 11
- [24] Anton Obukhov, Maximilian Seitzer, Po-Wei Wu, Semen Zhydenko, Jonathan Kyl, and Elvis Yu-Jing Lin. High-fidelity performance metrics for generative models in pytorch, 2020. Version: 0.3.0, DOI: 10.5281/zenodo.4957738. 5
- [25] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 1
- [26] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016. 1
- [27] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On buggy resizing libraries and surprising subtleties in fid calculation. *arXiv preprint arXiv:2104.11222*, 2021. 5



- [28] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. *arXiv preprint arXiv:2101.12072*, 2021. 1
- [29] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 1
- [30] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *arXiv preprint arXiv:2104.07636*, 2021. 1
- [31] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016. 1
- [32] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226. PMLR, 2015. 2
- [33] Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models. *arXiv preprint arXiv:2104.05358*, 2021. 1
- [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. 2
- [35] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2, 4, 5, 6, 7
- [36] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *arXiv preprint arXiv:1907.05600*, 2019. 2
- [37] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *arXiv preprint arXiv:2006.09011*, 2020. 2
- [38] Daniel Watson, Jonathan Ho, Mohammad Norouzi, and William Chan. Learning to efficiently sample from diffusion probabilistic models. *arXiv preprint arXiv:2106.03802*, 2021. 2

## A. Model specifications and hyperparameters

The hyperparameters used across our experiments are the same as the compared baselines, this is in order to perform a concise evaluation. Still, for clarity and completeness of this work, we indicate the hyperparameters used for each model version.

### A.1. CIFAR10

In CIFAR10, we used the baselines DDPM [9]. The model is a UNet architecture, with the following hyperparameters. The UNet had a depth of 4 downsampling (and upsampling) blocks, with a base number channel size of 128 and channel multiplier of [1,2,2,2]. Each block contained a residual block with 2 residual layers, and an attention block at the 16x16 resolution. The model was subject to dropout of 0.1. Following the baseline, the linear noise schedule was from 1e-4 to 2e-2 in 1000 steps. Training was done on 4 GPUs, with a batch size of 128x4, for 1M iterations. The Adam optimizer was used with learning rate of 2e-4, and EMA decay of 0.9999.

For the cosine noise schedule, we used IDDPM [23]. The improved UNet model included a scale-shift GroupNorm instead of the standard GroupNorm, three residual layers in each block, attention on both the 16x16 and 8x8 resolutions, 4 attention heads instead of 1, and a cosine noise schedule.

### A.2. CelebA

DDPM was also selected for CelebA evaluation of 64x64 image resolution. The difference from its CIFAR10 counterpart, is the addition of a fifth downsampling layer, with the same base channel size of 64x4, and a channel multiplier of 4. Model was trained for 500K iterations, using batch size of 32, and Adam optimizer with learning rate 1e-5.

### A.3. ImageNet

The evaluated model on ImageNet is based on ADM [6]. This architecture has some major differences from the previous ones. The model had classifier condition, which were being added to the time condition. Instead of pooled downsampling and interpolated upsampling, a learned up/downsampling was applied through the residual block. In addition, each block has a base channel size of 256, with channel multipliers of [1,1,2,3,4]. Attention of 4 heads was applied on the 32x32, 16x16, and 8x8 resolutions. A dropout of 0.1 and scale-shift GroupNorm. Noise schedule was the default linear schedule. Finally, we trained only the decoder weights, while using the pretrained weights of the baseline for the rest of the model. Model was trained for 80K iterations, with batch size of 32x4 (with 8 mini-batches of 4). Adam optimizer with learning rate of 1e-5.

## B. Generated images

In addition to the images in the paper, we provide additional generated images for the various datasets, for further inspection.

## C. Progressive generation

Fig. 9 shows progressive generation results for CIFAR10 and CelebA. All grids show the intermediate results of the three paths  $\epsilon_\theta$ , *dual*, and  $x_\theta$ , from top to bottom. It can be seen how in all cases,  $\epsilon_\theta$  starts very noisy, while  $x_\theta$  is blurry. All paths end with a similar image, but the dual method provides a sharper and less noisy result. In CelebA we noted more difference between the images. The additive path often produced darker images, and the noise in the final result of  $\epsilon_\theta$  is very noticeable.

## D. Effect of iteration count

Fig. 10 shows generation for both CIFAR10 and CelebA with a different number of denoising iterations. Iteration are monotonically increasing from left to right (5, 10, 20, 50, 100). The effect of the number of iterations is very clear as the image becomes more detailed and sharp when more denoising iterations are applied. Sometimes there is also a change in appearance, but an improvement in quality is always present. However, it can be seen that the change in quality is relatively low, and an already good image is achieved with few iterations.

## E. High quality ImageNet result (128×128)

Fig. 11 shows generated images from ImageNet, using 50 denoising iterations. There is a high variety in the images, and the class condition successfully represents the chosen category. Considering that the model was only fine-tuned for 80K steps, and the denoising is done with only 50 iterations, the image quality is quite good.

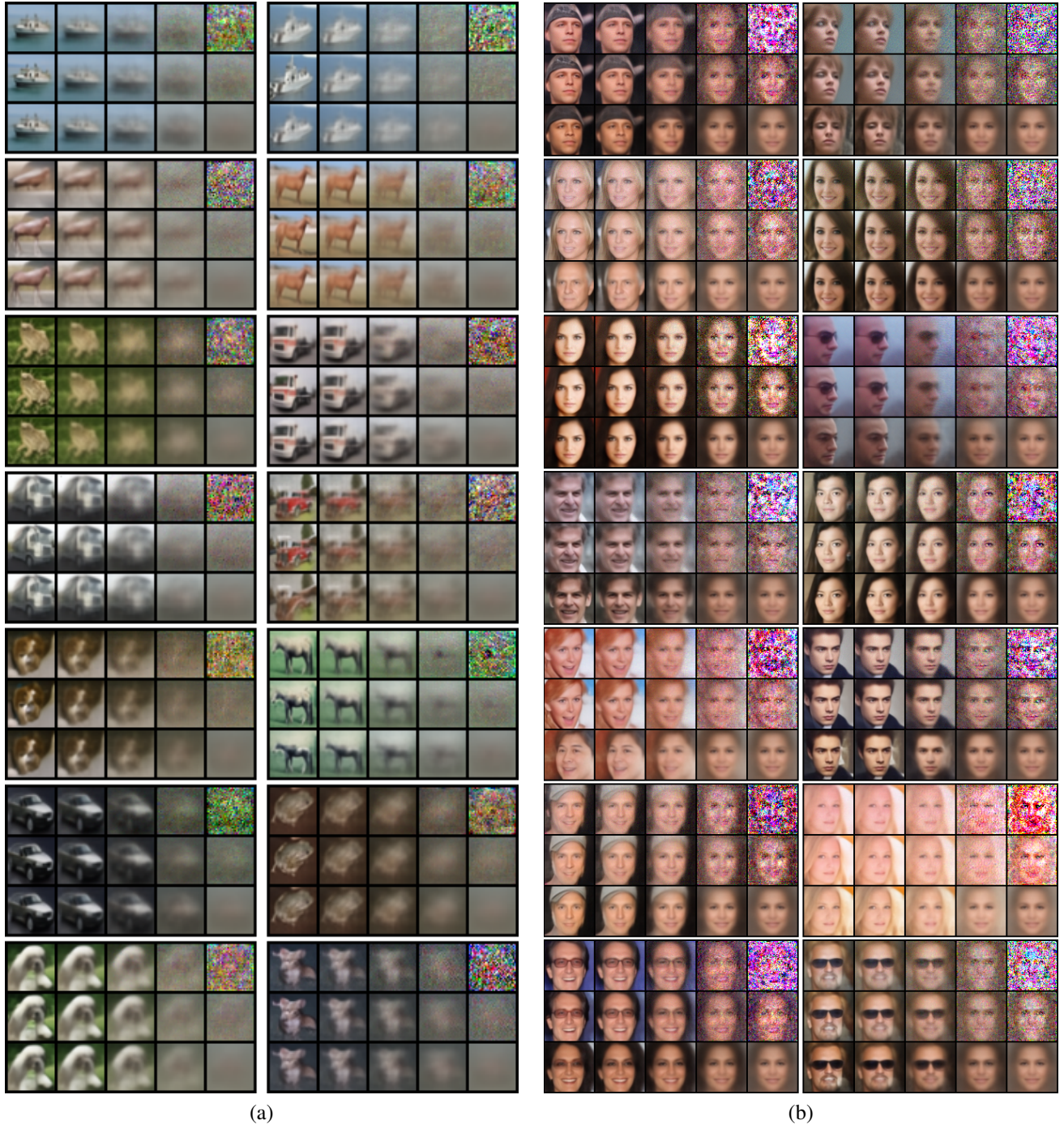
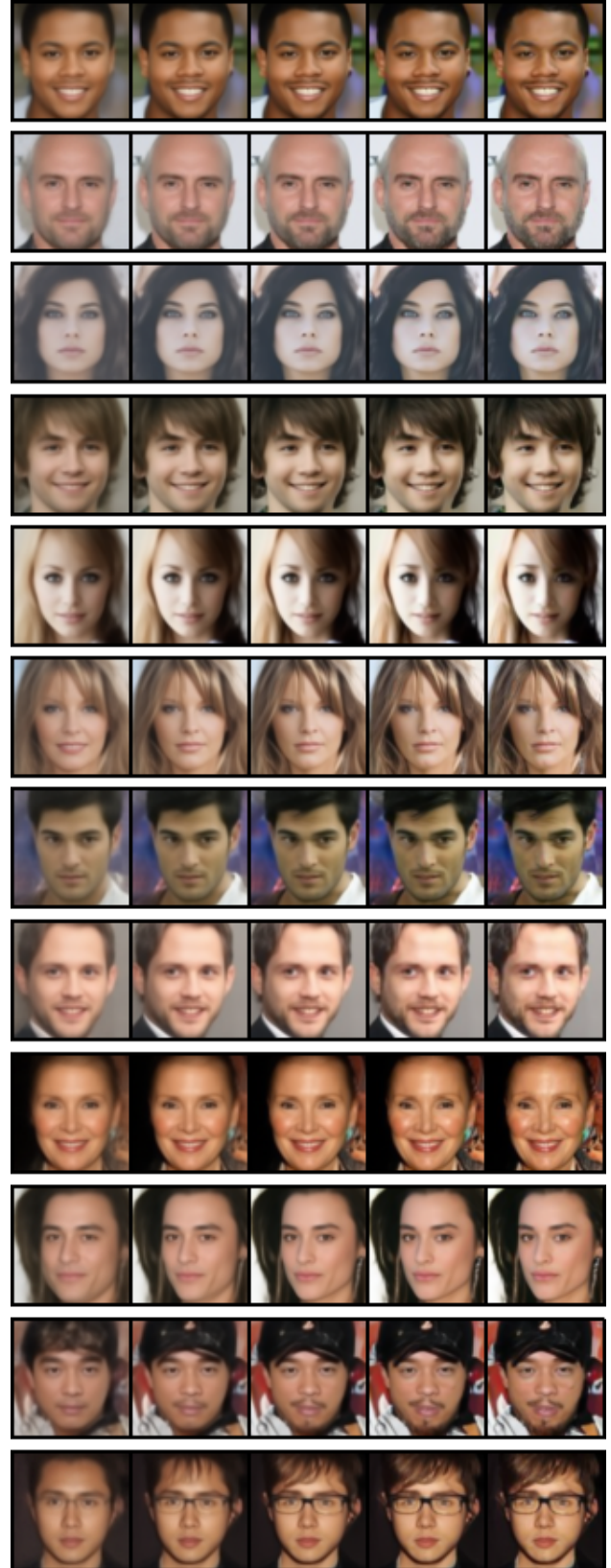


Figure 9. Progressive generation (a) CIFAR10 and (b) CelebA. From top to bottom:  $\epsilon_\theta$ ,  $dual$ , and  $x_\theta$ .





(a)



(b)

Figure 10. Generation on (a) CIFAR10 and (b) CelebA. From left to right: 5, 10, 20, 50, 100 iterations.

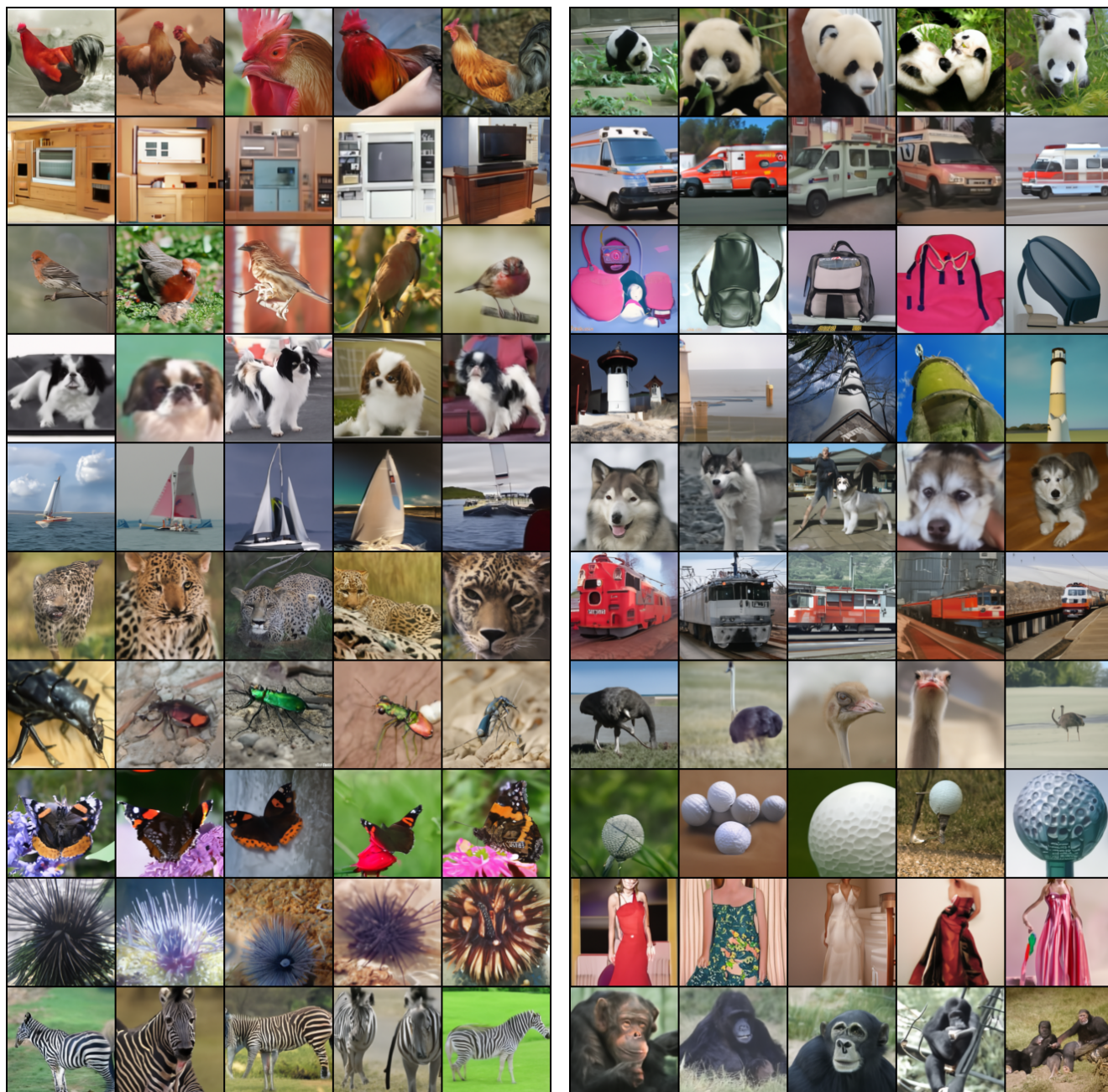


Figure 11. Generation on ImageNet with 50 iterations.