https://arxiv.org/abs/2112.07804

Tackling the Generative Learning Trilemma with DENOISING DIFFUSION GANS

Vikram Voleti
Christopher Beckham
Jae Hyun Lim
PhD students, Mila

voletiv.github.io, beckham.nz, jaehyunlim.com

Feb15, 2022 @ (virtual)

Introduction

TACKLING THE GENERATIVE LEARNING TRILEMMA WITH DENOISING DIFFUSION GANS

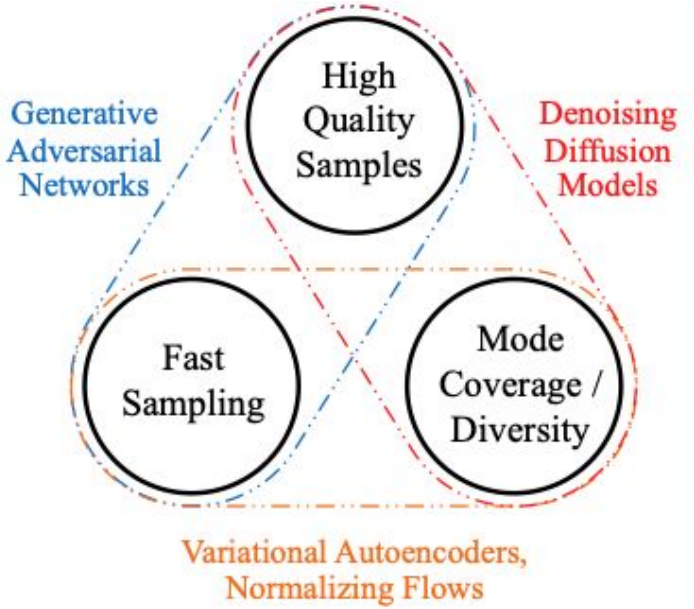
Zhisheng Xiao*
The University of Chicago
zxiao@uchicago.edu

Karsten Kreis
NVIDIA
kkreis@nvidia.com

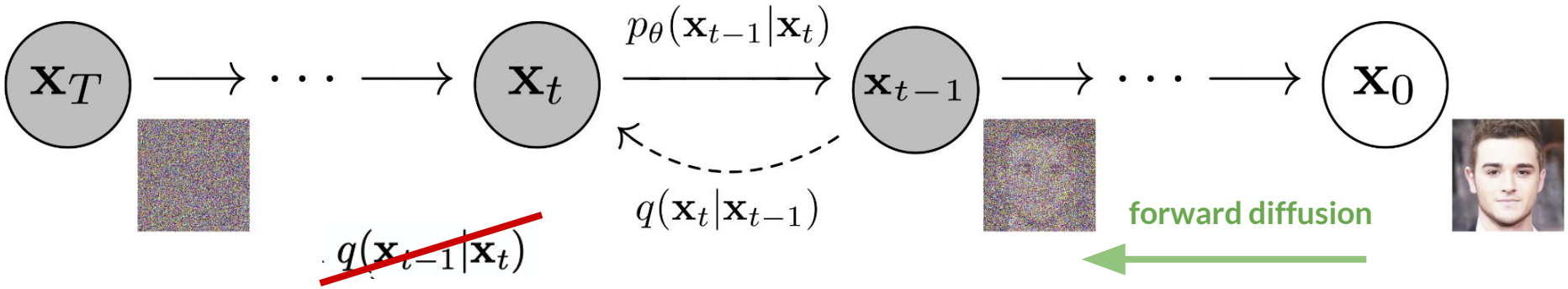
Arash Vahdat
NVIDIA
avahdat@nvidia.com

ABSTRACT

A wide variety of deep generative models has been developed in the past decade. Yet, these models often struggle with simultaneously addressing three key requirements including: high sample quality, mode coverage, and fast sampling. We call the challenge imposed by these requirements *the generative learning trilemma*, as the existing models often trade some of them for others. Particularly, denoising diffusion models have shown impressive sample quality and diversity, but their expensive sampling does not yet allow them to be applied in many real-world applications. In this paper, we argue that slow sampling in these models is fundamentally attributed to the Gaussian assumption in the denoising step which is justified only for small step sizes. To enable denoising with large steps, and hence, to reduce the total number of denoising steps, we propose to model the denoising distribution using a complex multimodal distribution. We introduce *denoising diffusion generative adversarial networks (denoising diffusion GANs)* that model each denoising step using a multimodal conditional GAN. Through extensive evaluations, we show that denoising diffusion GANs obtain sample quality and diversity competitive with original diffusion models while being 2000× faster on the CIFAR-10 dataset. Compared to traditional GANs, our model exhibits better mode coverage and sample diversity. To the best of our knowledge, denoising diffusion GAN is the first model that reduces sampling cost in diffusion models to an extent that allows them to be applied to real-world applications inexpensively. Project page and code: https://nvlabs.github.io/denoising-diffusion-gan.



Introduction



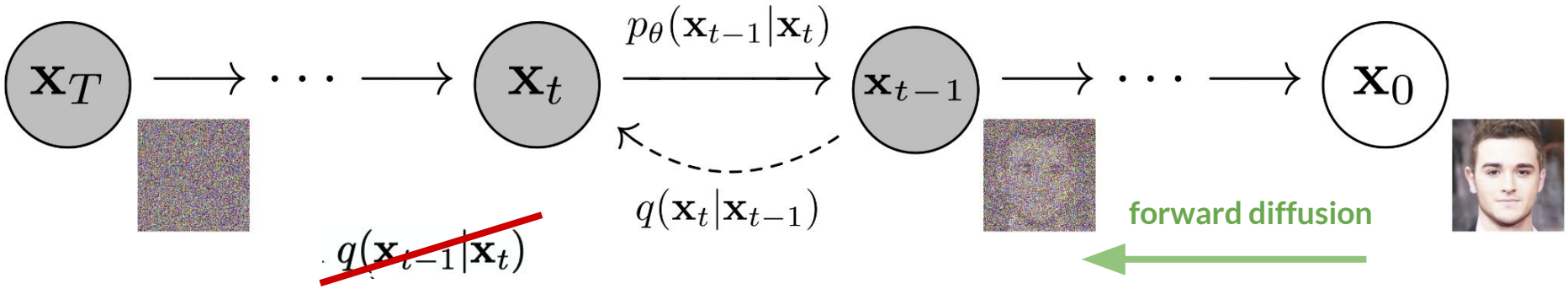
In diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020), there is a forward process that gradually adds noise to the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ in T steps with pre-defined variance schedule β_t:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t\geq 1} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{1}$$

where $q(\mathbf{x}_0)$ is a data-generating distribution. The reverse denoising process is defined by:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t\geq 1} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}), \tag{2}$$

Introduction



$$\mathcal{L} = -\sum_{t\geq 1} \mathbb{E}_{q(\mathbf{x}_t)} \left[D_{\mathrm{KL}} \left(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \right) \right] + C,$$

Denoising distribution p(.) is assumed to be Gaussian, and with infinitesimal step sizes, it can also be shown that the reversal of q (the denoising form of q) is as well

Introduction

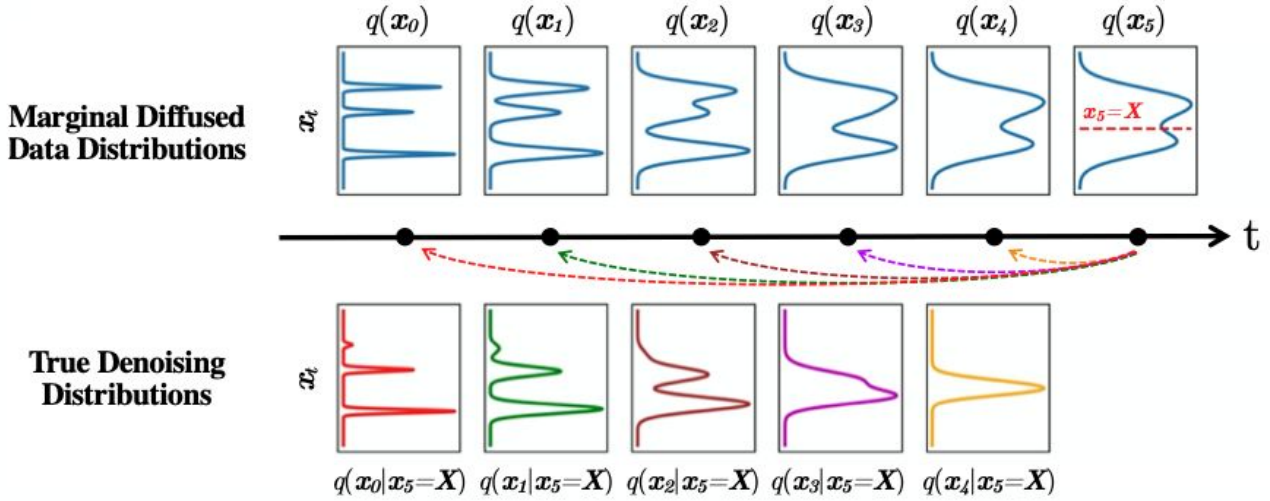


Figure 2: **Top**: The evolution of 1D data distribution $q(\mathbf{x}_0)$ through the diffusion process. **Bottom**:, The visualization of the true denoising distribution for varying step sizes conditioned on a fixed $\mathbf{x}_5$. The true denoising distribution for a small step size (i.e., $q(\mathbf{x}_4|\mathbf{x}_5 = X)$) is close to a Gaussian distribution. However, it becomes more complex and multimodal as the step size increases.

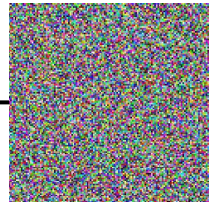
GAN

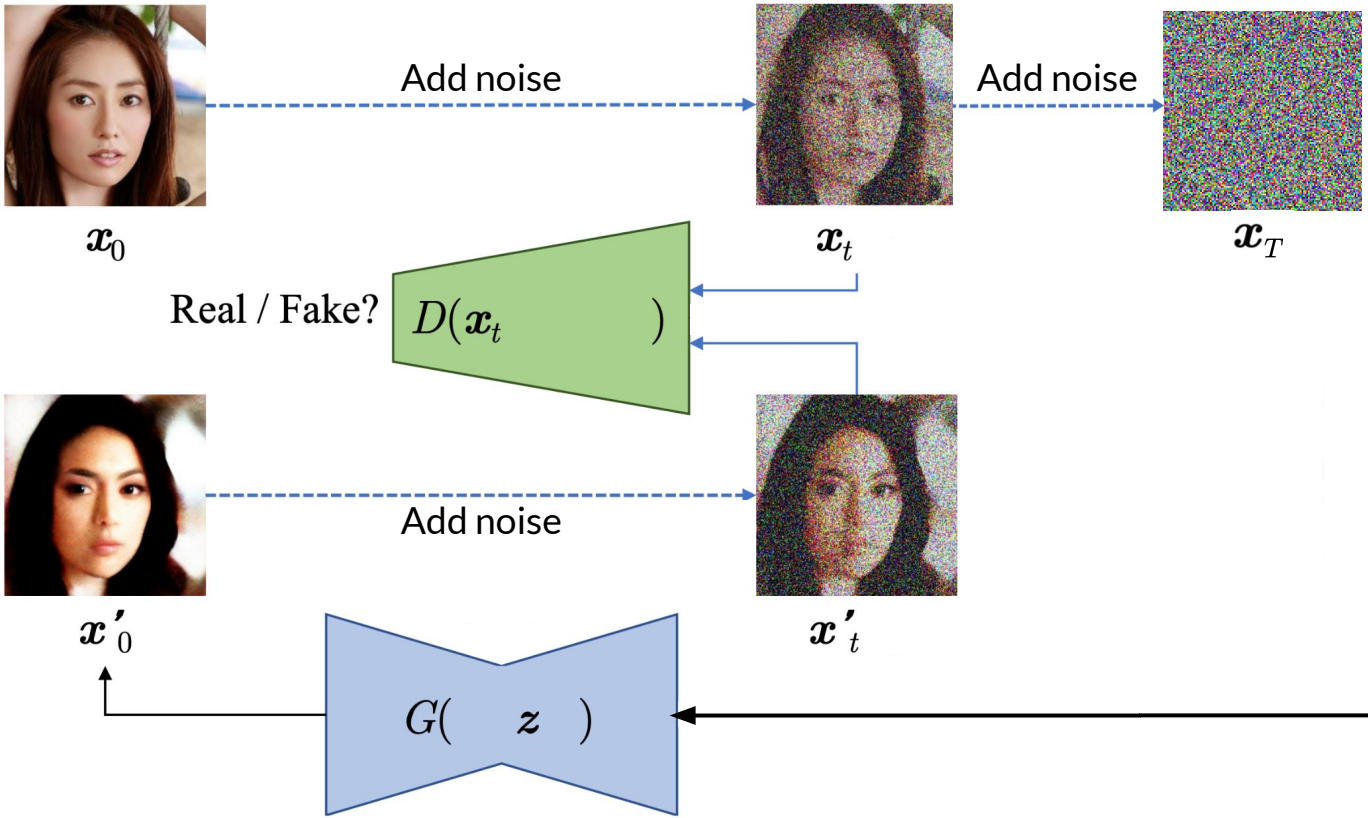
x_0

Real / Fake? $D(x_0)$

x'_0

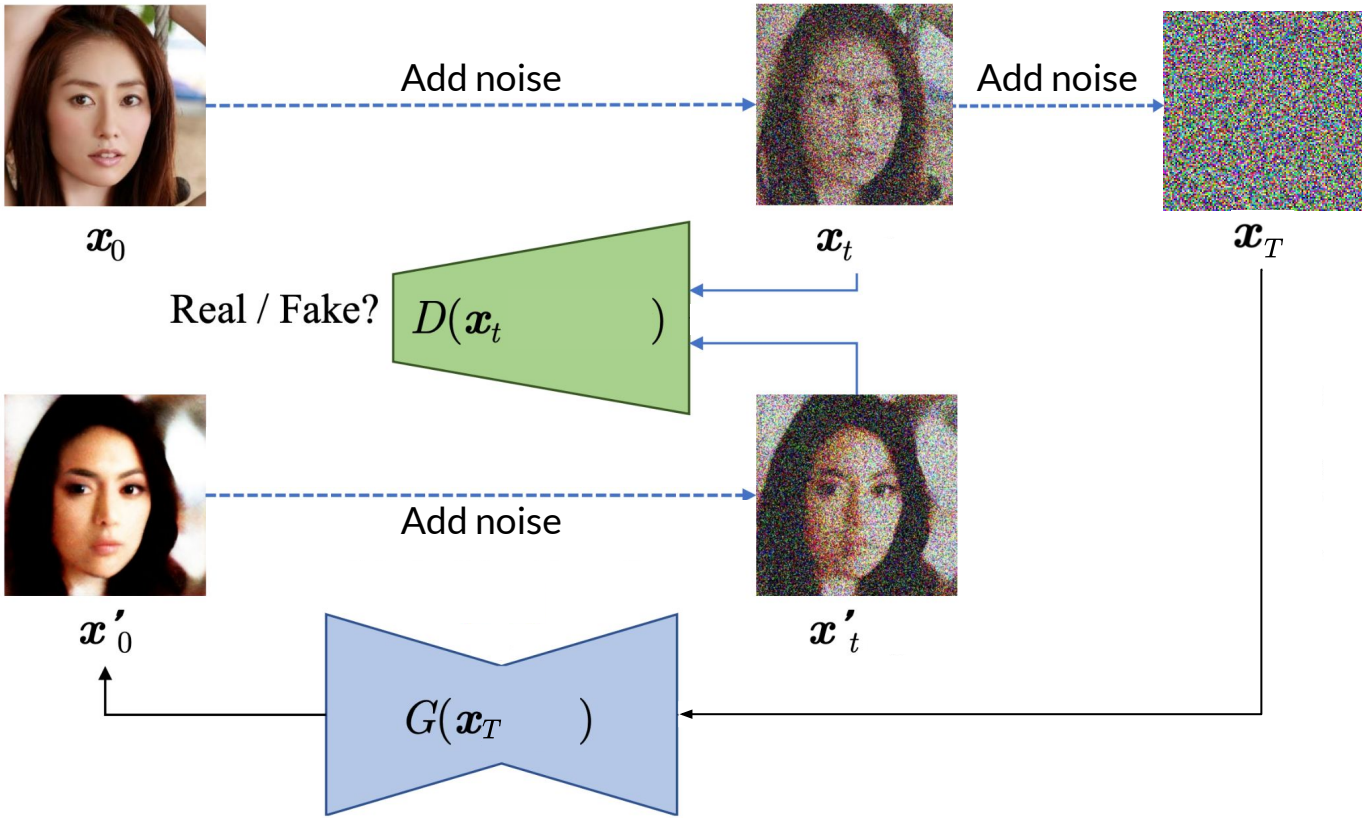
$G(z)$





GAN, but

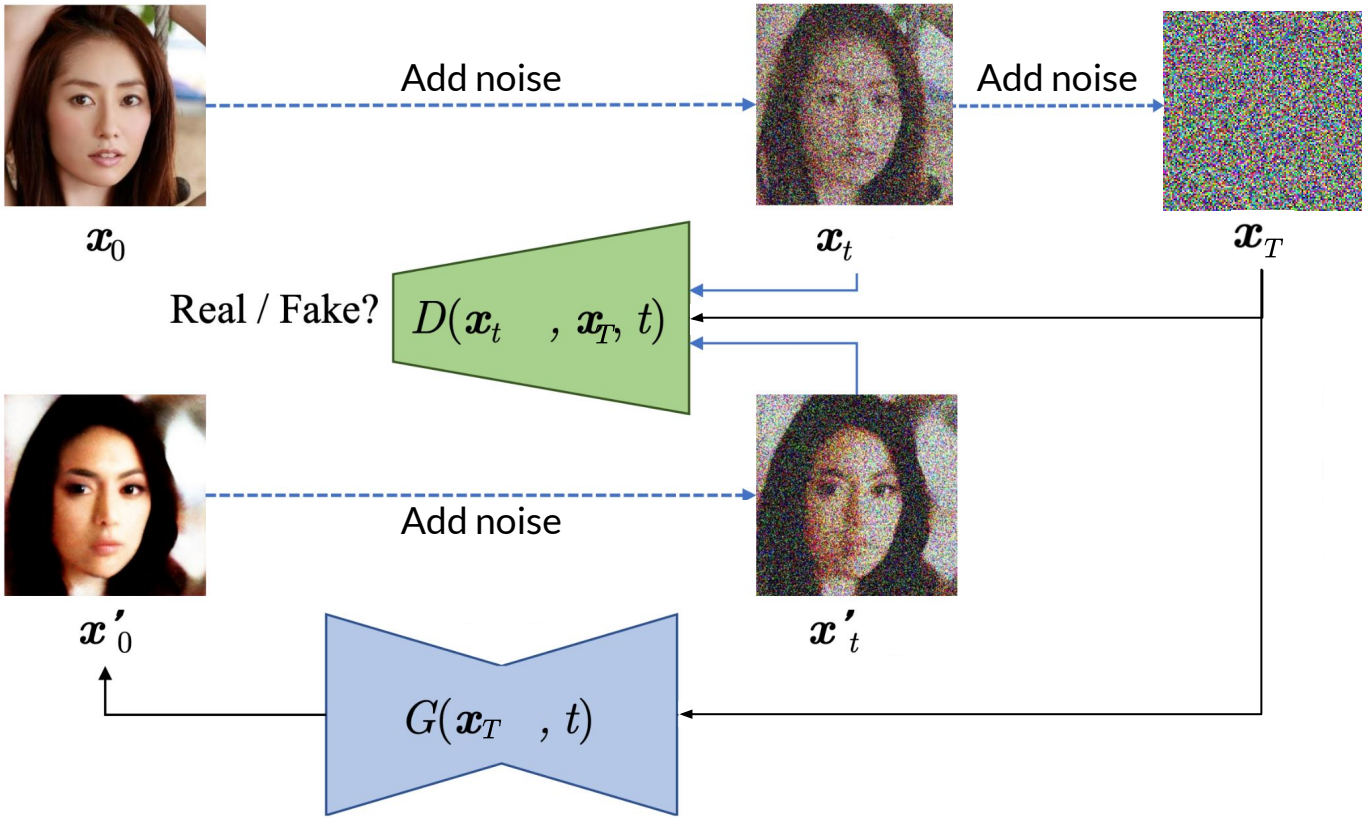
D discriminates image+noise



GAN, but

D discriminates image+noise,

input to G is real image + noise

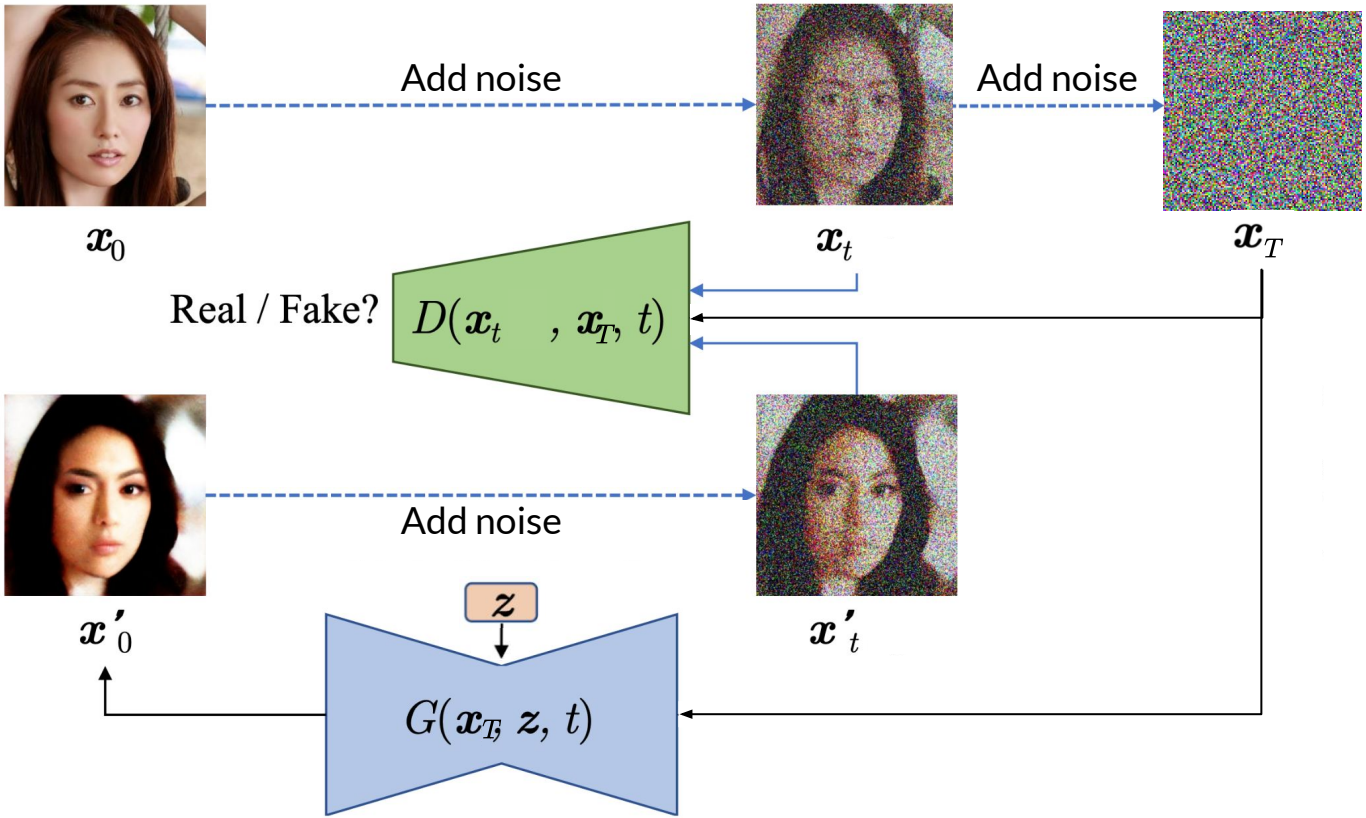


GAN, but

D discriminates image+noise,

input to G is real image + noise,

D is conditioned on real image + noise



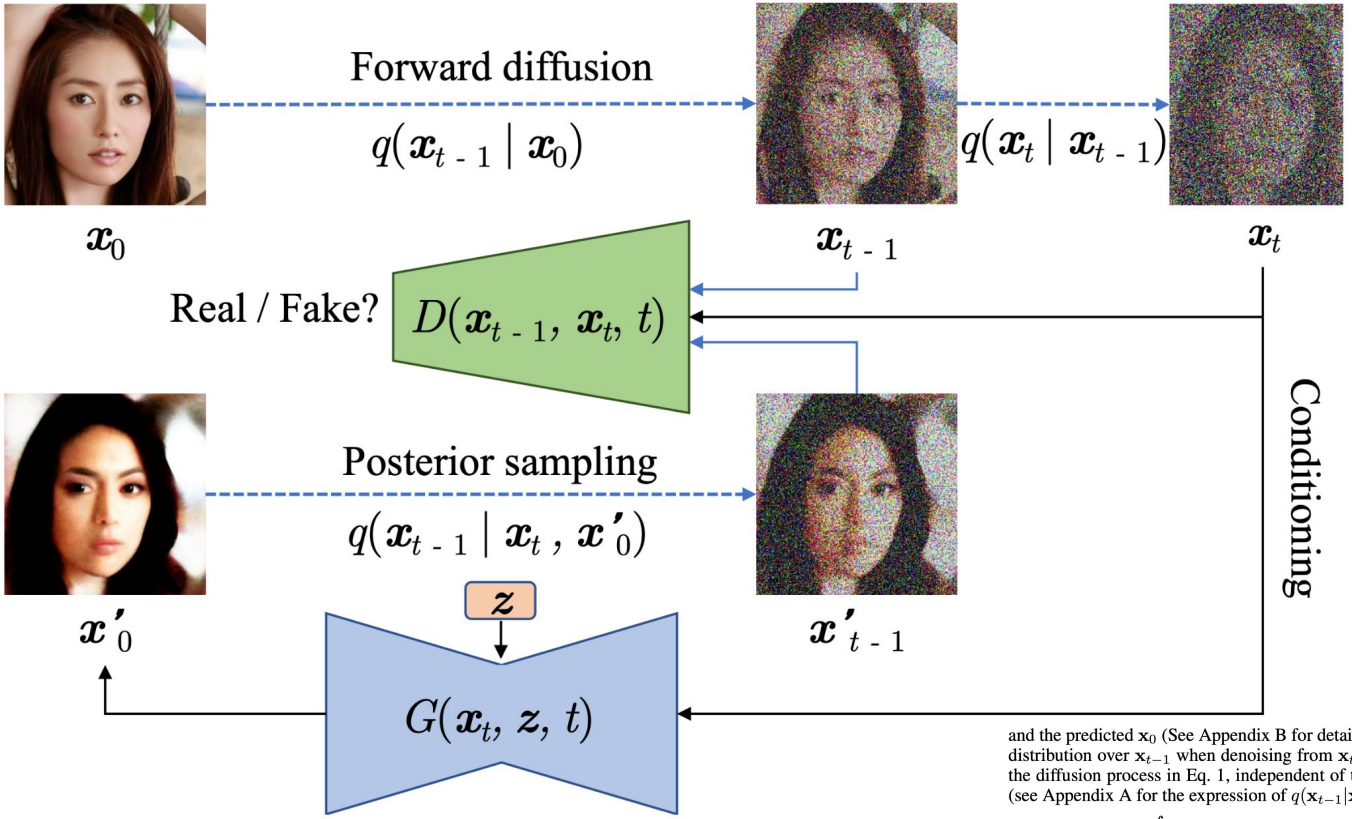
GAN, but

D discriminates image+noise,

input to G is real image + noise,

D is conditioned on real image + noise,

G is conditioned on random noise



Iterate, et voila!

Denoising Diffusion GAN!

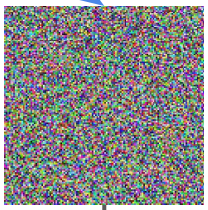
and the predicted $\mathbf{x}_0$ (See Appendix B for details). The distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$ is intuitively the distribution over $\mathbf{x}_{t-1}$ when denoising from $\mathbf{x}_t$ towards $\mathbf{x}_0$, and it always has a Gaussian form for the diffusion process in Eq. 1, independent of the step size and complexity of the data distribution (see Appendix A for the expression of $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$). Similarly, we define $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \int p_\theta(\mathbf{x}_0|\mathbf{x}_t) q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) d\mathbf{x}_0 = \int p(\mathbf{z}) q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0 = G_\theta(\mathbf{x}_t, \mathbf{z}, t)) d\mathbf{z}, \quad (6)$$

where $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$ is the implicit distribution imposed by the GAN generator $G_\theta(\mathbf{x}_t, \mathbf{z}, t) : \mathbb{R}^N \times \mathbb{R}^L \times \mathbb{R} \to \mathbb{R}^N$ that outputs $\mathbf{x}_0$ given $\mathbf{x}_t$ and an L-dimensional latent variable $\mathbf{z} \sim p(\mathbf{z}) := \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$.

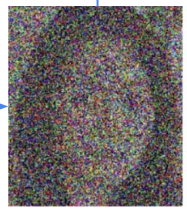
Iteration

Add noise



Real / Fake? $D(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t, t)$

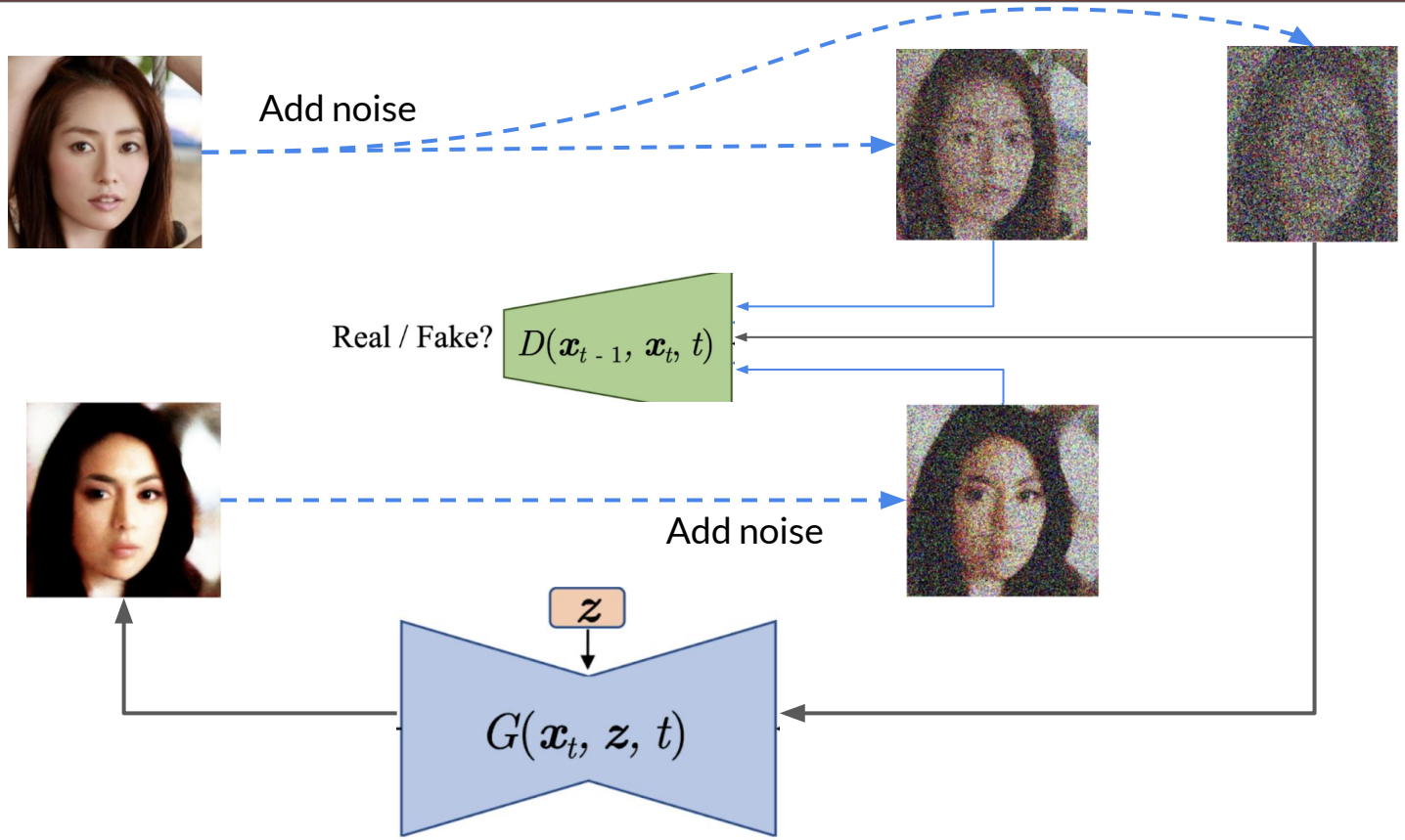
Add noise



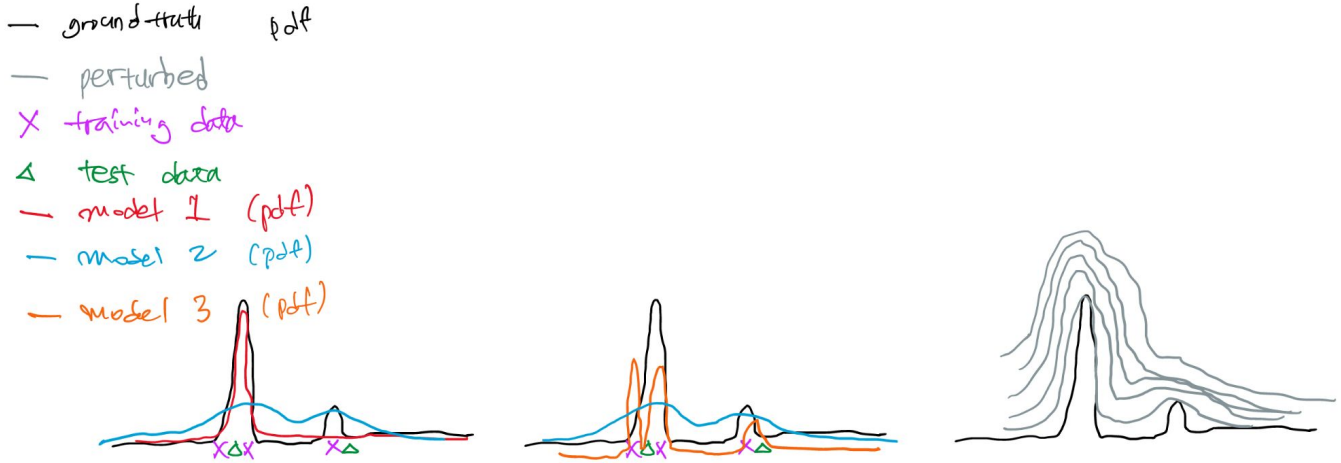
$\boldsymbol{z}$

$G(\boldsymbol{x}_t, \boldsymbol{z}, t)$

Iteration



Jae's explanation



ex) GAN w/o reg
(RKL) $D_{KL}(\text{model 1} \| gt) < D_{KL}(\text{model 2} \| gt)$

ex) VAE $D_{KL}(gt \| \text{model 1}) > D_{KL}(gt \| \text{model 2})$

ex) SM $SM(gt \| \text{model 3}) > SM(gt \| \text{model 2})$ (or model 1)
= matching the gradient.

Thank you!

voletiv.github.io, beckham.nz, jaehyunlim.com