Denoising Diffusion Models

vi-lab
권민기

Contents

- DDPM
- DDIM
- Score-based models
- CDM
- Others…
- 제 연구 소개..?

참고자료 : CVPR2022 diffusion tutorials.

Before start..

Q. 수식이 많은데 다 알아야 하나요?
A.목적에 따라 다를 것 같습니다.

1. 새로운 학습 방법을 만들고 싶다 -> 자세히 알아야 합니다.
2. 관련 연구를 하고싶고, Diffusion 논문을 읽고 싶다. -> 식 전개까지는 모르셔도 될 듯 합니다만 흐름은 알아야 합니다.
3. 모델을 돌리고 싶다. / 단순한 활용만 하고 싶다. -> 결론만 알아도 됩니다. ; 실제로 코드상에선 제일 마지막 결론만 사용됩니다.

Denoising Diffusion Models

Emerging as powerful generative models, outperforming GANs

"Diffusion Models Beat GANs on Image Synthesis"
Dhariwal & Nichol, OpenAI, 2021

"Cascaded Diffusion Models for High Fidelity Image Generation"
Ho et al., Google, 2021

Physical intuition

Diffusion destroys structure!

연기의 밀도를 알아내는 것은 어렵다.

하지만 시간이 지나면 결국 고르게 분포되어 Uniform 해 질 것이다.

Physical intuition

Diffusion destroys structure!

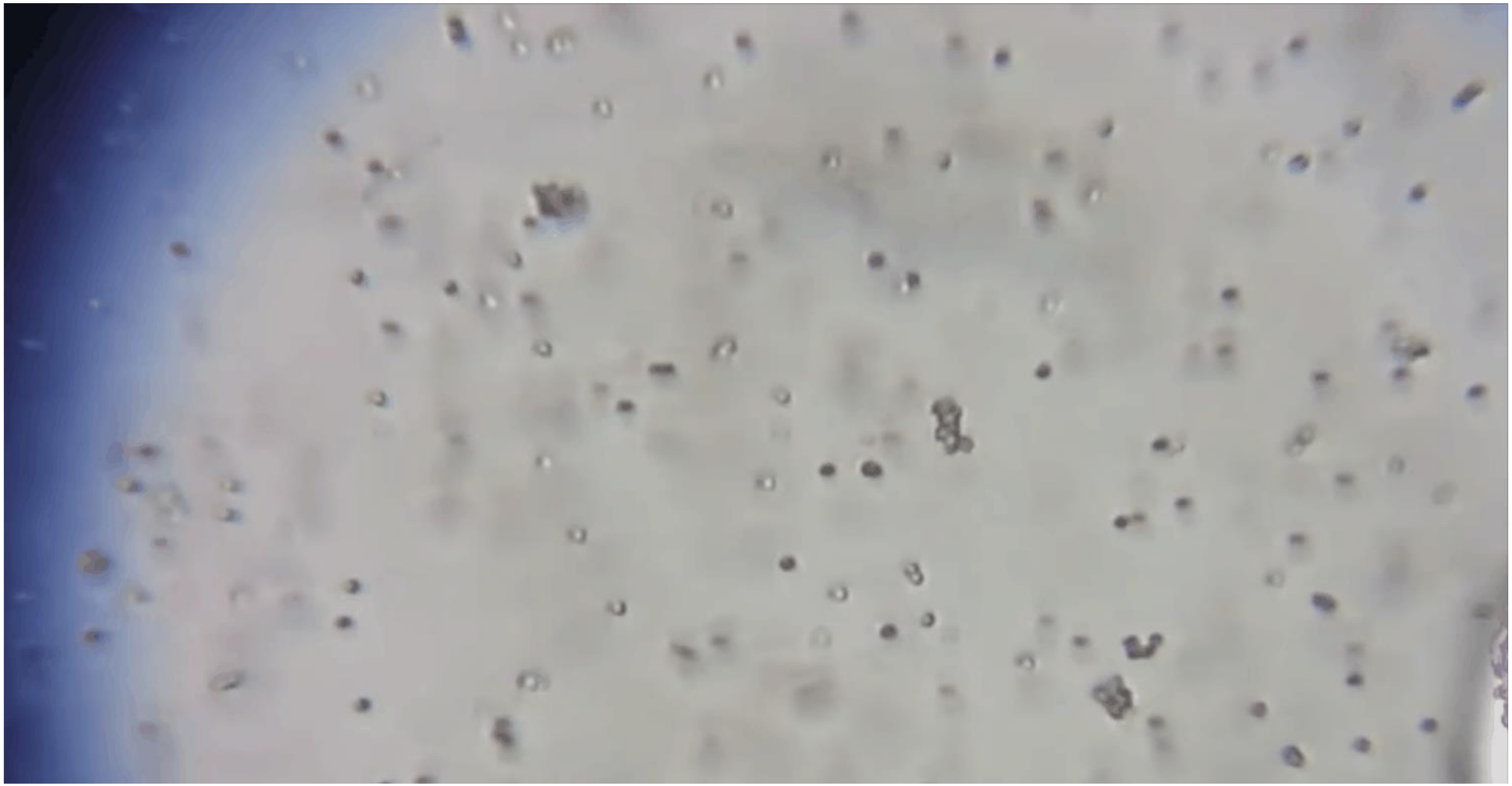
연기의 밀도를 알아내는 것은 어렵다.

하지만 시간이 지나면 결국 고르게 분포되어 Uniform 해 질 것이다.

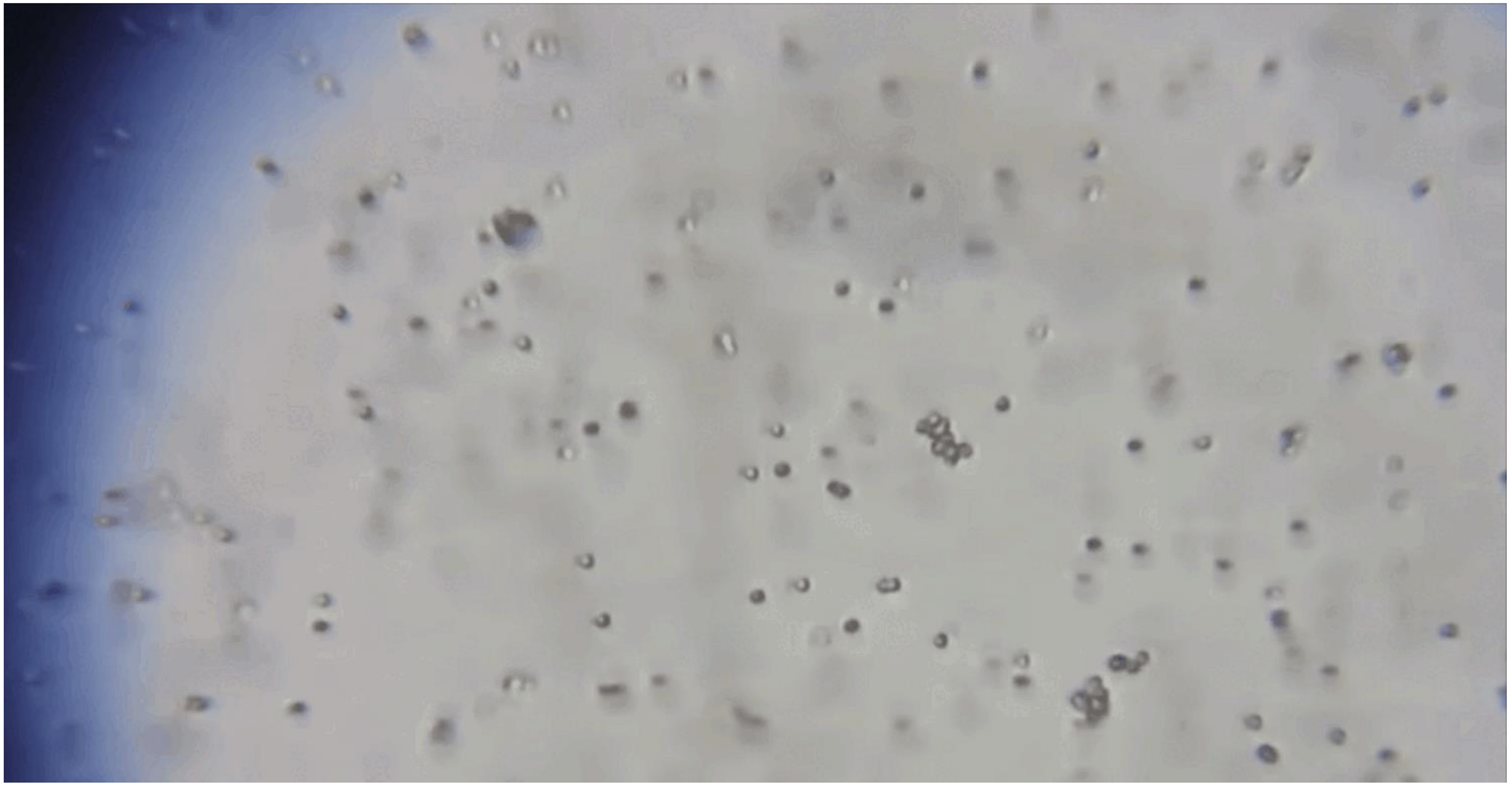
그렇다면 Uniform에서 시작해서 다시 처음 상태로 되돌릴 수 있다면?

딥러닝을 이용하여 이를 해결해 볼 수 있다.

Physical intuition

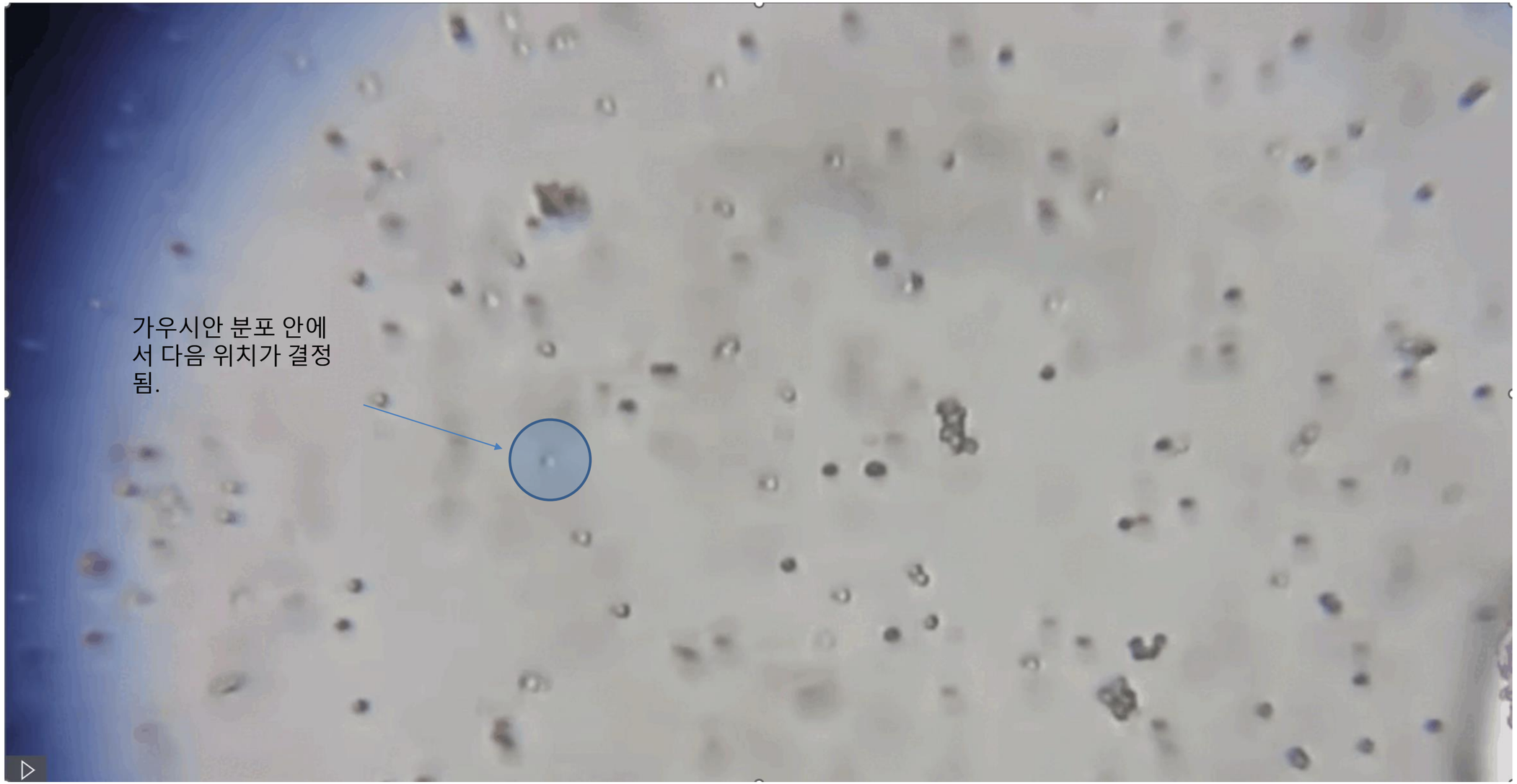


Physical intuition



Physical intuition

작은 sequence에서의 확산은 forward와 reverse 모두 가우시안일 수 있다. (물리적으로)

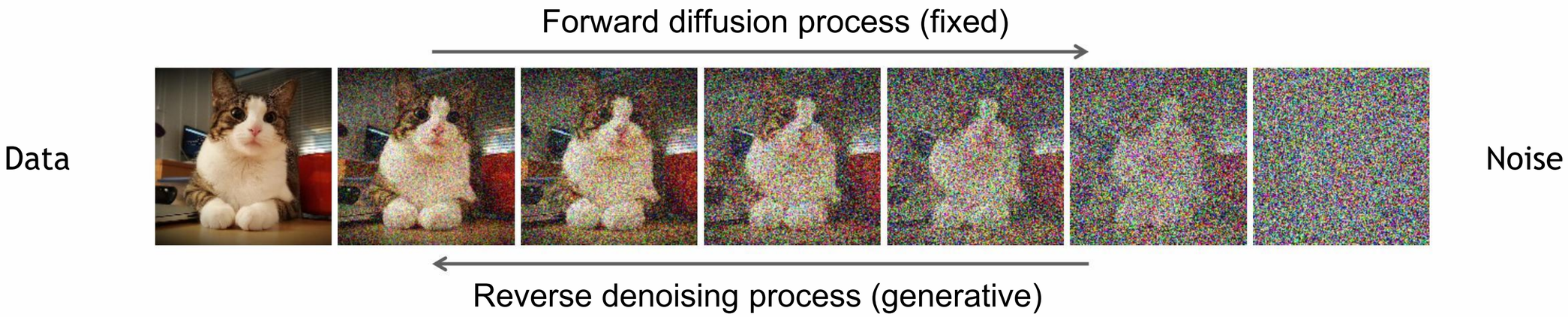


Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising

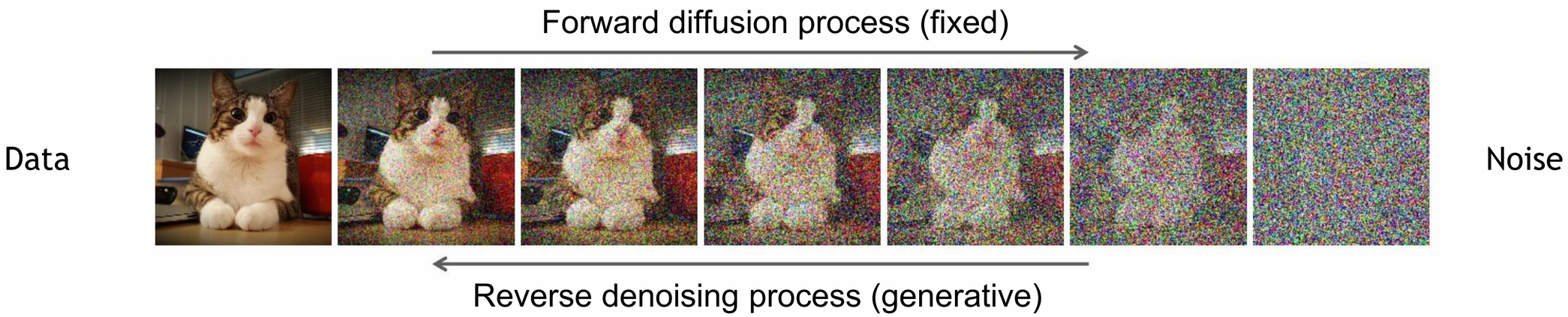


Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Denoising Diffusion Models

Learning to generate by denoising

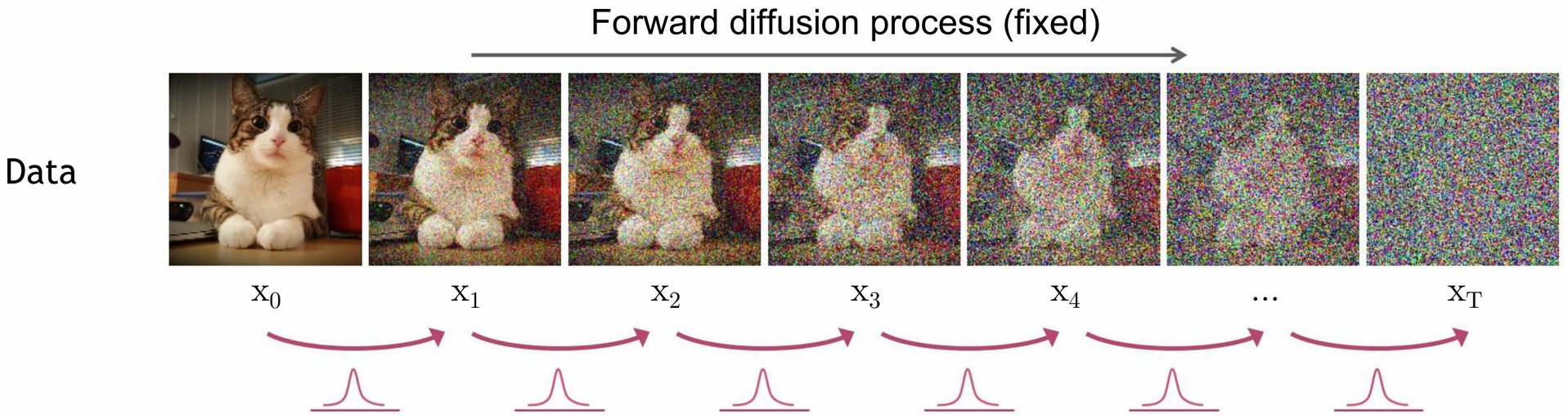
Image to noise : EASY!!



Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Forward Diffusion Process

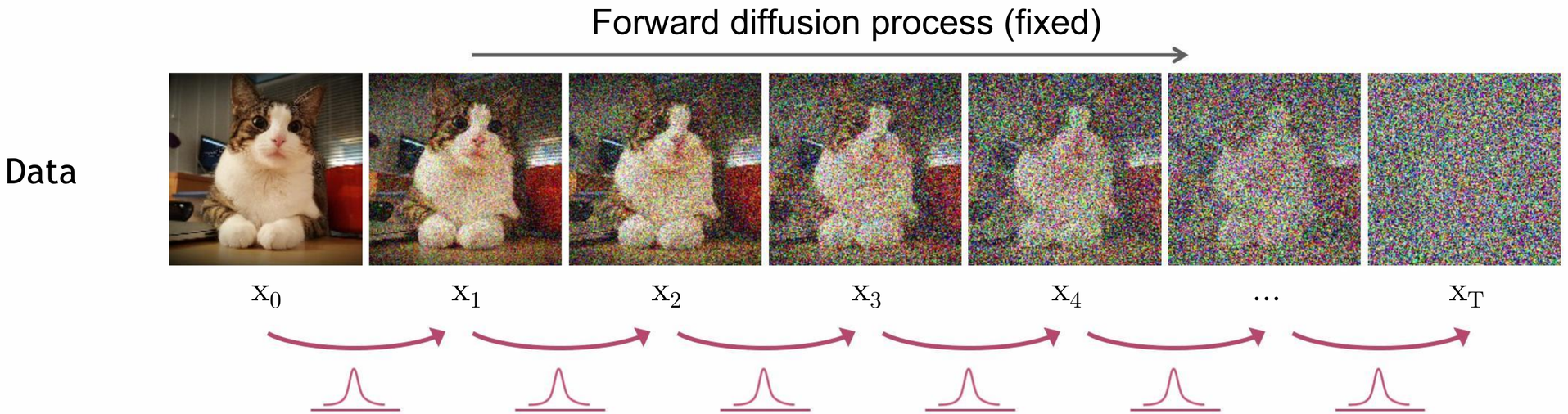
The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

Forward Diffusion Process

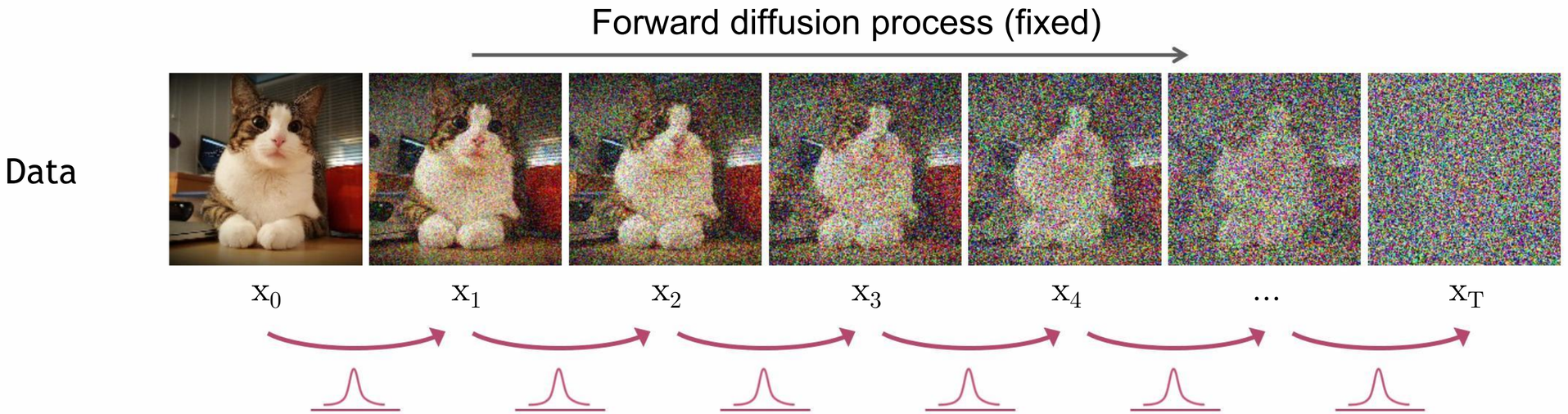
The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \underbrace{\sqrt{1-\beta_t}\mathbf{x_{t-1}}}_{\text{약간 감소시킴.}}, \underbrace{\beta_t\mathbf{I}}_{\substack{\text{노이즈를} \\ \text{조금 더함.}}}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

Forward Diffusion Process

The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \underline{\sqrt{1-\beta_t}\mathbf{x_{t-1}}}, \underline{\beta_t}\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

약간 감소시킴.

노이즈를 조금 더함.

$Var(ax) = a^2 Var(x)$

$Var(x+y) = Var(x) + Var(y)$

$Var(\sqrt{1-\beta_t}x_t + \beta_t) = Var(\sqrt{1-\beta_t}^2 + \beta_t)$

Diffusion Kernel

Forward diffusion process (fixed)

Data

Noise

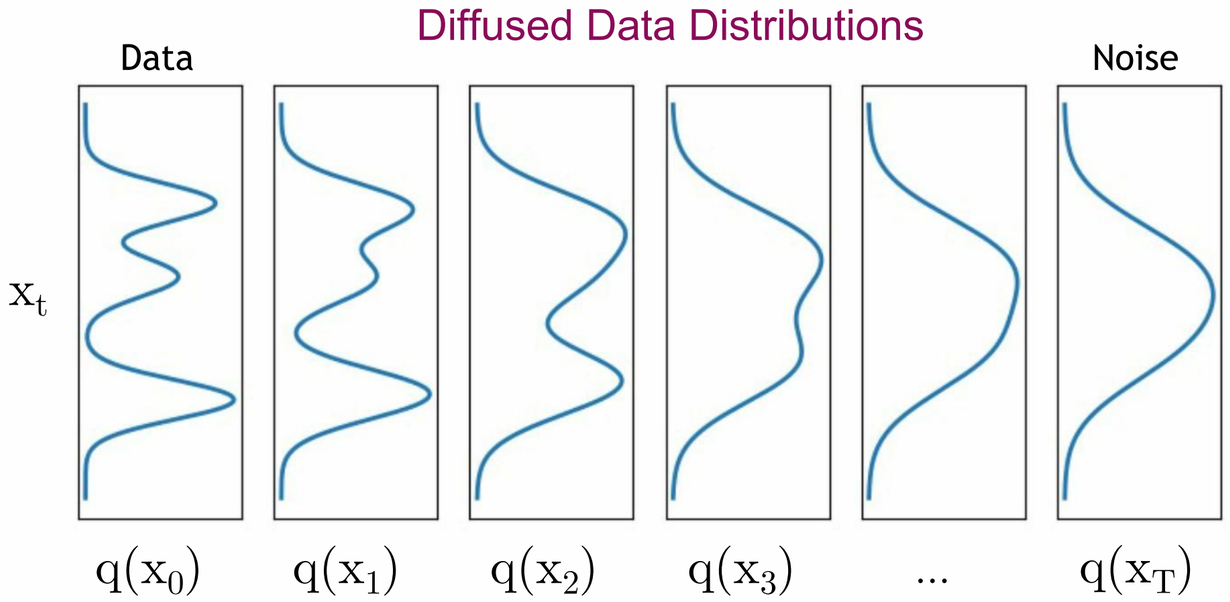
Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

What happens to a distribution in the forward diffusion?

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} d\mathbf{x}_0$$



We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

Generative Learning by Denoising

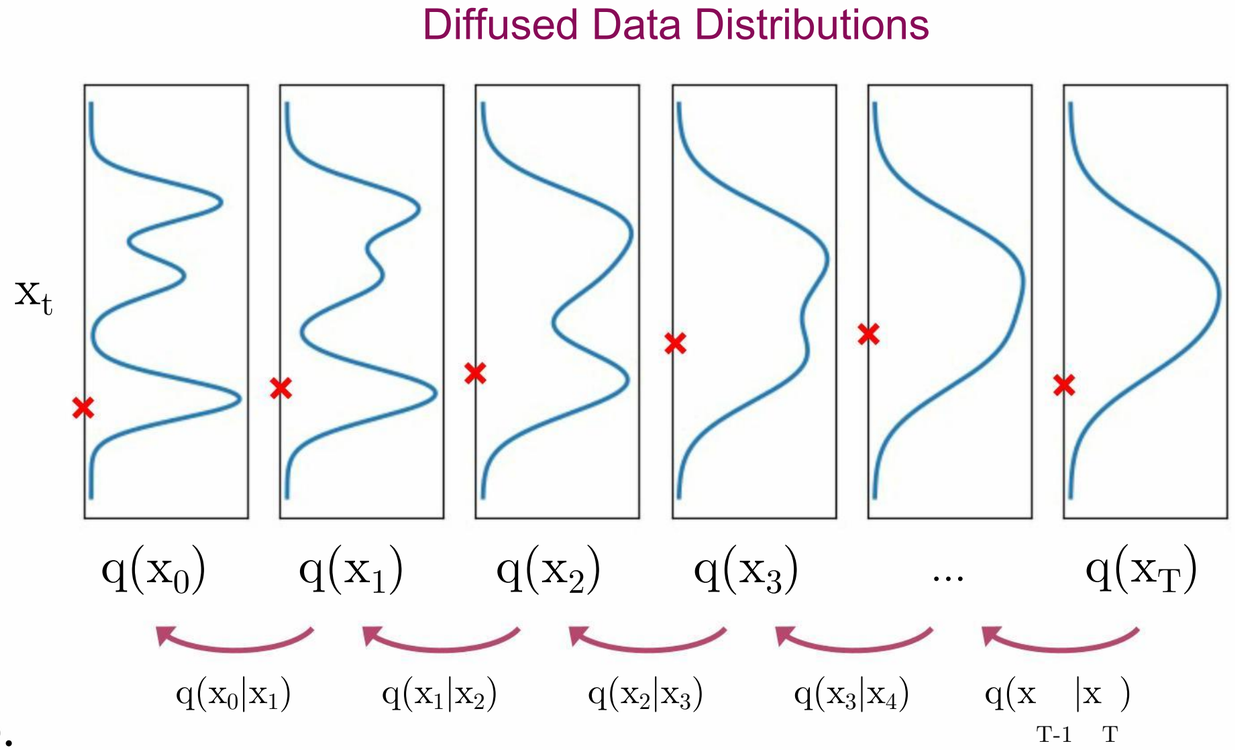
Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Generation:

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}$

True Denoising Dist.

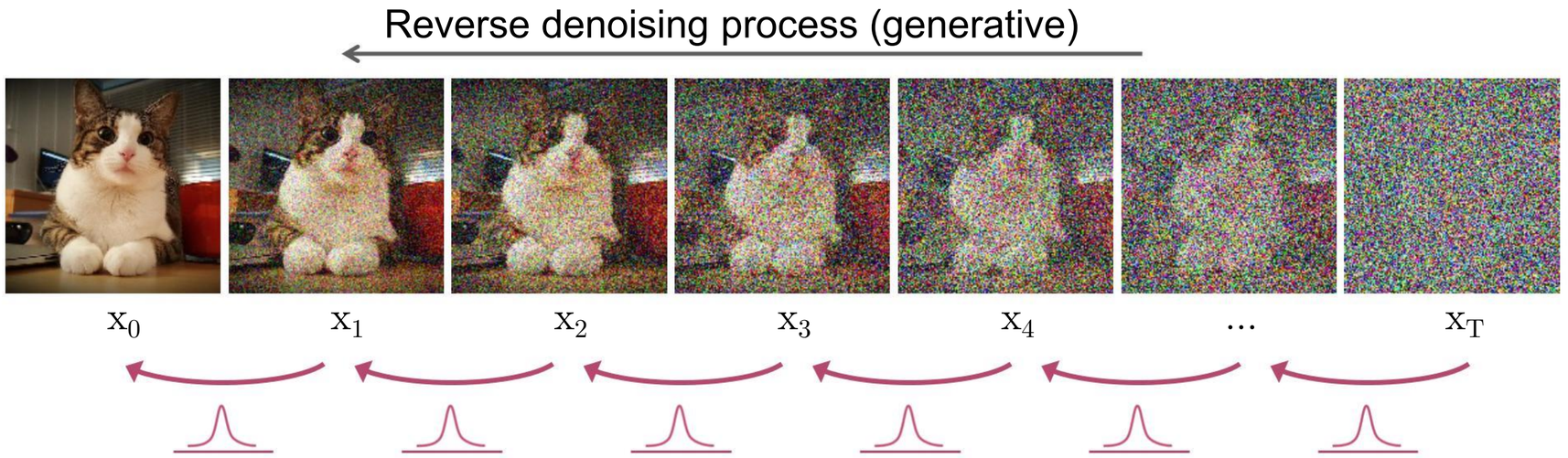


In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$? Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020 show that:

$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1))}_{L_0}\right]$$

Regression

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is the tractable posterior distribution:

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

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Parameterizing the Denoising Model

Since both $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are Normal distributions, the KL divergence has a simple form:

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} ||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2 \right] + C$$

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\, \epsilon$. Ho et al. NeurIPS 2020 observe that:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right)$$

They propose to represent the mean of the denoising model using a *noise-prediction* network:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\, \epsilon_\theta(\mathbf{x}_t, t) \right)$$

With this parameterization

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{\beta_t^2}{2\sigma_t^2 (1-\beta_t)(1-\bar{\alpha}_t)} ||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\mathbf{x}_t}, t)||^2 \right] + C$$

Training Objective Weighting

Trading likelihood for perceptual quality

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \Big[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon, t)||^2 \Big]$$

The time dependent λ_t ensures that the training objective is weighted properly for the maximum data likelihood training.

However, this weight is often very large for small t's.

Ho et al. NeurIPS 2020 observe that simply setting $\lambda_t = 1$ improves sample quality. So, they propose to use:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \Big[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon}_{\mathbf{x}_t}, t)||^2 \Big]$$

advanced weighting - Choi et al., Perception Prioritized Training of Diffusion Models, CVPR 2022.
- Karras et al., Elucidating the Design Space of Diffusion-Based Generative Models, arxiv preprint 2022

Summary

Training and Sample Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

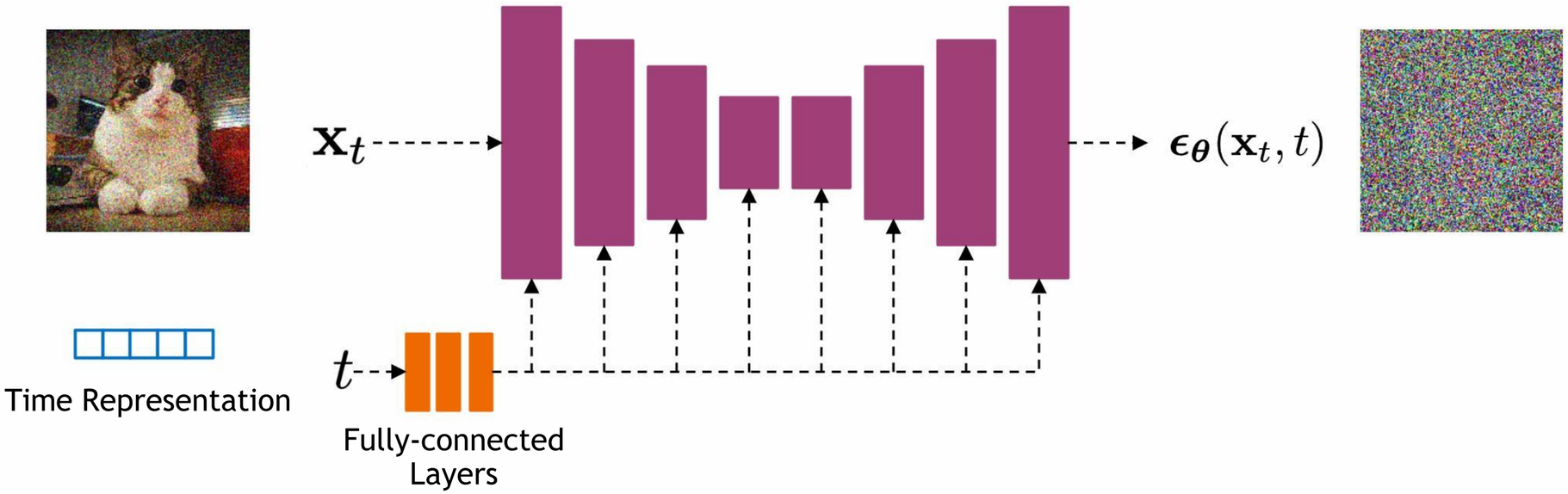
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Implementation Considerations

Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dhariwal and Nichol NeurIPS 2021)

Diffusion Parameters

Noise Schedule

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

Data

Noise

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

Above, β_t and σ_t^2 control the variance of the forward diffusion and reverse denoising processes respectively.

Often a linear schedule is used for β_t, and σ_t^2 is set equal to β_t.

Kingma et al. NeurIPS 2022 introduce a new parameterization of diffusion models using signal-to-noise ratio (SNR), and show how to learn the noise schedule by minimizing the variance of the training objective.

We can also train σ_t^2 while training the diffusion model by minimizing the variational bound (Improved DPM by Nichol and Dhariwal ICML 2021) or after training the diffusion model (Analytic-DPM by Bao et al. ICLR 2022).

Content-Detail Tradeoff

Reverse denoising process (generative)

Data

Noise

x_0 x_1 x_2 x_3 x_4 ... x_T

The denoising model is specialized for generating the high-frequency content (i.e., low-level details)

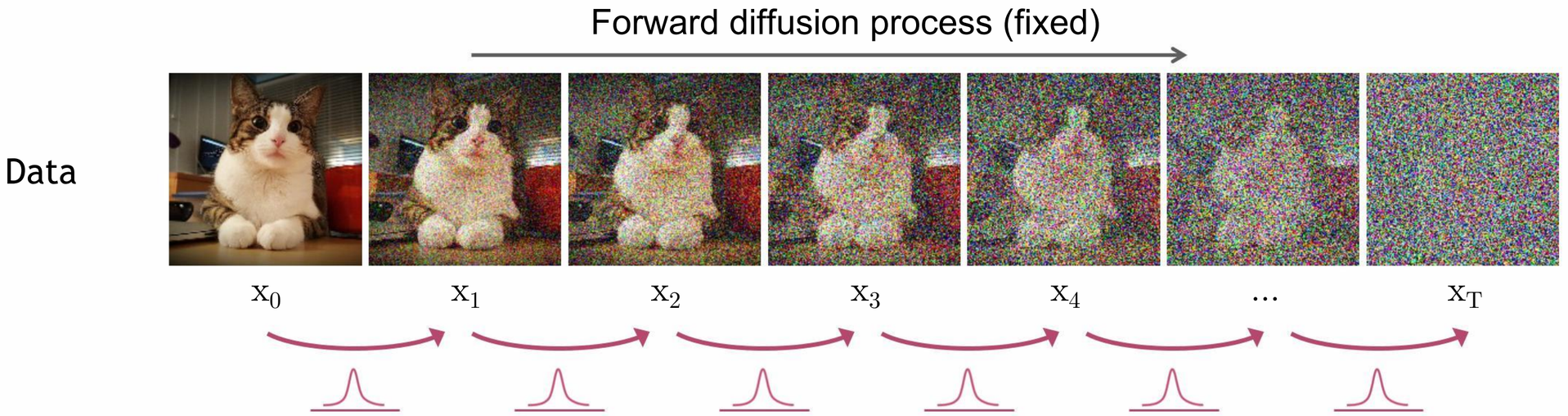
The denoising model is specialized for generating the low-frequency content (i.e., coarse content)

The weighting of the training objective for different timesteps is important!

OK, Go Back!

Forward Diffusion Process

The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

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

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

$q(x_{1:T}|x_0) = \frac{q(x_0,x_1,x_2,x_3,\ldots,x_T)}{q(x_0)}$ =?? $\prod_{t=1}^{T} q(x_t|x_{t-1}) = q(x_1|x_0)q(x_2|x_1)q(x_3|x_2)\ldots q(x_T|x_{T-1})$

Marcov chain!!

$q(x_T|x_{T-1}) = q(x_T|x_{T-1}, x_{T-2}, \ldots, x_1, x_0)$

$$q(x_1|x_0)q(x_2|x_1)q(x_3|x_2)\ldots q(x_T|x_{T-1}) = \frac{q(x_1,x_0)}{x_0}\frac{q(x_2,x_1)}{x_1}\ldots\frac{q(x_T,x_{T-1})}{x_{T-1}}$$

$$= \frac{q(x_1,x_0)}{x_0}\frac{q(x_2,x_1,x_0)}{x_1,x_0}\ldots\frac{q(x_T,x_{T-1},\ldots,x_1,x_0)}{x_{T-1},\ldots,x_0}$$

$$= \frac{q(x_0,x_1,x_2,x_3,\ldots,x_T)}{q(x_0)}$$

Diffusion Kernel

Forward diffusion process (fixed)

Data

Noise

Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x_t} ; \sqrt{1-\beta_t} \mathbf{x_{t-1}}, \beta_t \mathbf{I}\right)$$

$$q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{\bar{\alpha}_t} \mathbf{x}_0,\left(1-\bar{\alpha}_t\right) \mathbf{I}\right))$$

$$Var(ax) = a^2 Var(x)$$

$$Var(x+y) = Var(x) + Var(y)$$

$$Var(\sqrt{1-\beta_t}x_t + \beta_t) = Var(\sqrt{1-\beta_t}^2 + \beta_t)$$

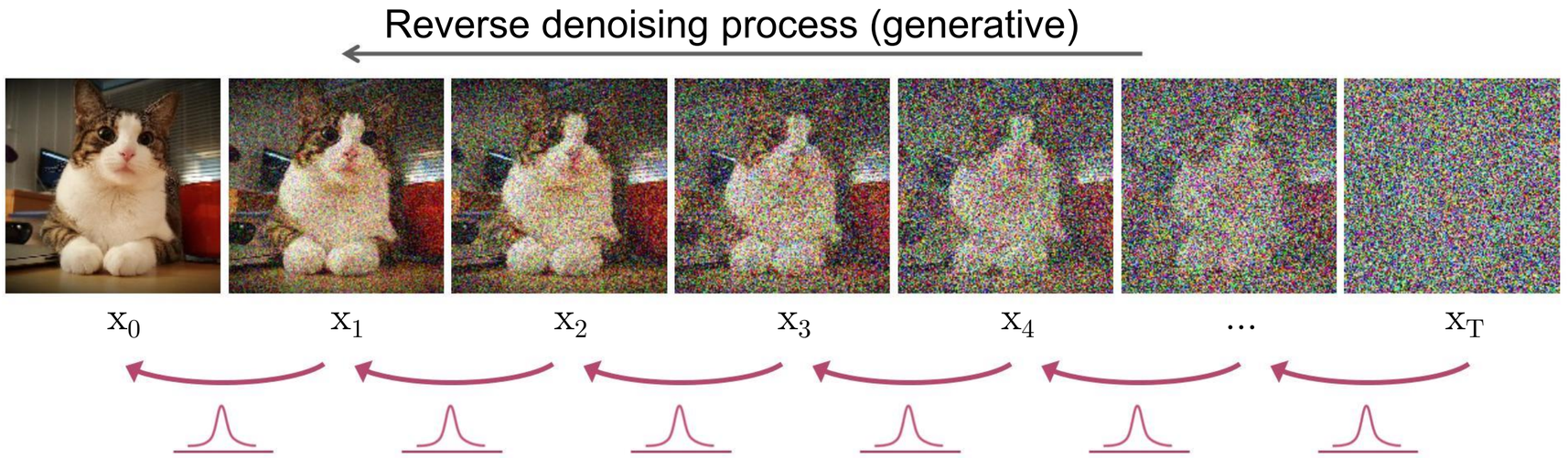
Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network}}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Marcov chain!!

$$p_\theta(x_0|x_1)p_\theta(x_1|x_2)p_\theta(x_2|x_3)...p_\theta(x_{T-1}|x_T) = \frac{p_\theta(x_0,x_1,...,x_{T-1},x_T)}{p_\theta(x_1,...,x_{T-1},x_T)} \frac{p_\theta(x_1,...,x_{T-1},x_T)}{p_\theta(x_2,...,x_{T-1},x_T)} ... \frac{p_\theta(x_{T-1},x_T)}{p_\theta(x_T)}$$

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020 show that:

$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1))}_{L_0}\right]$$

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is the tractable posterior distribution:

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

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020 show that:

$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1))}_{L_0}\right]$$

VAE to DDPM

$$\mathbb{E}_{x_T\sim q(x_T|x_0)}[-\log p_\theta(x_0)]$$

① $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_0,x_1,x_2,\dots,x_T)}{p_\theta(x_1,x_2,x_3,\dots,x_T|x_0)}\right]$ $\because$ bayes rule, $p_\theta(x_T|x_0)=\frac{p_\theta(x_T,x_0)}{p_\theta(x_0)}$

② $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_0,x_1,x_2,\dots,x_T)}{p_\theta(x_1,x_2,x_3,\dots,x_T|x_0)}\cdot\frac{q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)}\right]$

③ $\leq \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_0,x_1,x_2,\dots,x_T)}{q(x_{1:T}|x_0)}\right]$ $\because KL\ divergence > 0$, "**ELBO**"

④ $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$ $\because Notation$

⑤ $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(\boldsymbol{x_T})\prod_{t=1}^{T}p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})}{\prod_{t=1}^{T}q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})}\right]$ $\because Below\ Markov\ chain\ property$

⑥ $= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T})-\sum_{t=1}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$ $\because separating\ to\ summation\ in\ logarithm$

$$p_\theta(\boldsymbol{x_{0:T}}) \coloneqq p_\theta(\boldsymbol{x_T})\prod_{t=1}^{T}p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}) \qquad q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}) \coloneqq \prod_{t=1}^{T}q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})$$

DDPM loss

$$\mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}[-\log p_\theta(x_0)]$$

⑦ $$\leq \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T}) - \sum_{t=1}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$$

⑧ $$= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T}) - \sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$$

⑨ $$= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T}) - \sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\cdot\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$$ ∵*

⑩ $$= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T}) - \sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)} - \sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$$

⑪ $$= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T}) - \sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)} - \log\frac{q(x_1|x_0)}{q(x_T|x_0)} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$$

⑫ $$= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log\frac{p_\theta(\boldsymbol{x_T})}{q(x_T|x_0)} - \sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)} - \log p_\theta(x_0|x_1)\right]$$

* $q(x_t|x_{t-1})$
  $= q(x_t|x_{t-1},x_0)$ ∵ Markov chain property
  $= \frac{q(x_t,x_{t-1},x_0)}{q(x_{t-1},x_0)}$ ∵ bayes rule
  $= \frac{q(x_{t-1},x_t,x_0)}{q(x_{t-1},x_0)}\cdot\frac{q(x_t,x_0)}{q(x_t,x_0)}$
  $= q(x_{t-1}|x_t,x_0)\cdot\frac{q(x_t,x_0)}{q(x_{t-1},x_0)}$

$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1))}_{L_0}\right]$$

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020 show that:

$$L = \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1}\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1))}_{L_0}\right]$$

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ is the tractable posterior distribution:

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

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is the tractable posterior distribution:

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

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

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

Using Bayes' rule, we have:

$$\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\
&\propto \exp\Big(-\frac{1}{2}\big(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big) \\
&= \exp\Big(-\frac{1}{2}\big(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t \mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big) \\
&= \exp\Big(-\frac{1}{2}\big((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})\mathbf{x}_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)\mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\big)\Big)
\end{aligned}$$

where $C(\mathbf{x}_t, \mathbf{x}_0)$ is some function not involving $\mathbf{x}_{t-1}$ and details are omitted. Following the standard Gaussian density function, the mean and variance can be parameterized as follows (recall that $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$):

$$\begin{aligned}
\tilde{\beta}_t &= 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = 1/(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\
\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) \\
&= (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\
&= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0
\end{aligned}$$

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is the tractable posterior distribution:

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

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Parameterizing the Denoising Model

$$KL(p,q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

Since both $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are Normal distributions, the KL divergence has a simple form:

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} ||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2 \right] + C$$

Parameterizing the Denoising Model

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$. Ho et al. NeurIPS 2020 observe that:

$$\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t)$$

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

They propose to represent the mean of the denoising model using a *noise-prediction* network:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$$

With this parameterization

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)||^2\right] + C$$

Training Objective Weighting

Trading likelihood for perceptual quality

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \Big[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon, t)||^2 \Big]$$

The time dependent λ_t ensures that the training objective is weighted properly for the maximum data likelihood training.

However, this weight is often very large for small t's.

Ho et al. NeurIPS 2020 observe that simply setting $\lambda_t = 1$ improves sample quality. So, they propose to use:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)} \Big[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\ \epsilon}_{\mathbf{x}_t}, t)||^2 \Big]$$

advanced weighting - Choi et al., Perception Prioritized Training of Diffusion Models, CVPR 2022.
- Karras et al., Elucidating the Design Space of Diffusion-Based Generative Models, arxiv preprint 2022

Summary

Training and Sample Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Summary

Denoising Diffusion Probabilistic Models

we reviewed denoising diffusion probabilistic models.

The model is trained by sampling from the forward diffusion process and training a denoising model to predict the noise.

We discussed how the forward process perturbs the data distribution or data samples.

The devil is in the details:

- Network architectures
- Objective weighting
- Diffusion parameters (i.e., noise schedule)

See “Elucidating the Design Space of Diffusion-Based Generative Models” by Karras et al. for important design decisions.

Summary

Denoising Diffusion Probabilistic Models

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

Forward diffusion process (fixed)

Data

Noise

Reverse denoising process (generative)

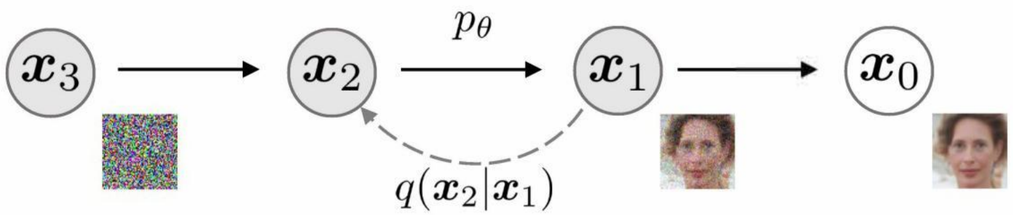
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I})$$

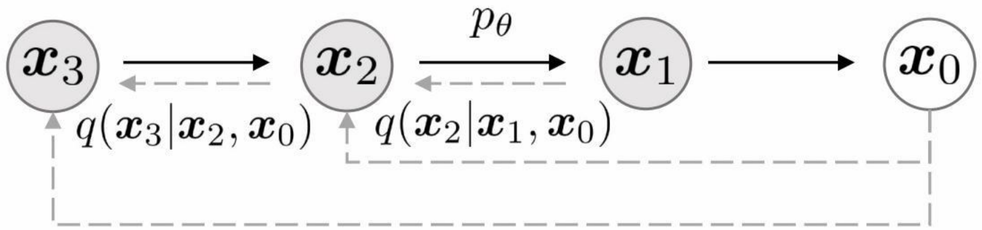
$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\mathbf{x}_t, t)\right)$$

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}),t\sim\mathcal{U}(1,T)}\left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)||^2\right]$$

Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process





Main Idea

Design a family of non-Markovian diffusion processes and corresponding reverse processes.

The process is designed such that the model can be optimized by the same surrogate objective as the original diffusion model.

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

Therefore, can take a pretrained diffusion model but with more choices of sampling procedure.

Song et al., “Denoising Diffusion Implicit Models”, ICLR 2021.

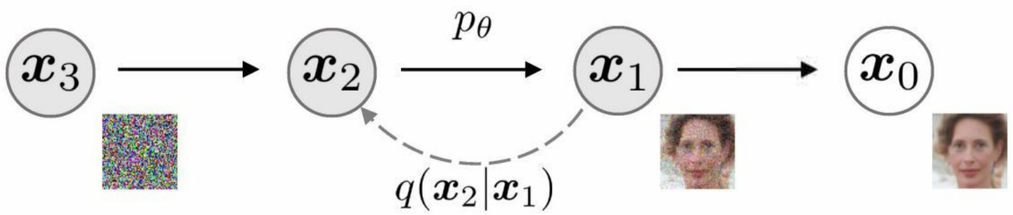
where $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is the tractable posterior distribution:

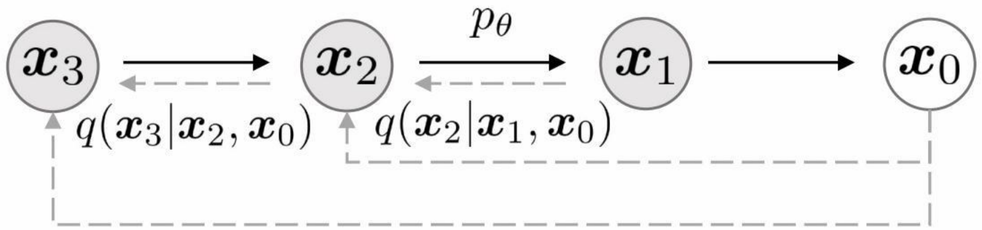
$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process





$$q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0) \prod_{t=2}^{T} q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$$

$$q(x_1|x_0)q(x_2|x_1, x_0)q(x_3|x_2, x_0)...q(x_T|x_{T-1}, x_0)$$

$$q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}$$

Song et al., “Denoising Diffusion Implicit Models”, ICLR 2021.

Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process

$$q_\sigma(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) := q_\sigma(\boldsymbol{x}_T|\boldsymbol{x}_0)\prod_{t=2}^{T} q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$$

$$q(x_1|x_0)q(x_2|x_1,x_0)q(x_3|x_2,x_0)...q(x_T|x_{T-1},x_0)$$

$$q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_0) = \frac{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)q_\sigma(\boldsymbol{x}_t|\boldsymbol{x}_0)}{q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}$$

For another approach, let's rewrite $q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ to be parameterized by a desired standard deviation σ_t according to the nice property:

$$\begin{aligned}
\mathbf{x}_{t-1} &= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\mathbf{z}_{t-1} \\
&= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\mathbf{z}_t + \sigma_t\mathbf{z} \\
&= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\mathbf{z}
\end{aligned}$$

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I})$$

reparameterization trick

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\,\epsilon$$

Recall that in $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$, therefore we have:

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

Song et al., "Denoising Diffusion Implicit Models", ICLR 2021.

Denoising diffusion implicit models (DDIM)

Non-Markovian diffusion process

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)=\mathcal{N}(\mathbf{x}_{t-1};\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}},\sigma_t^2\mathbf{I})$$

$$\mathbf{x}_t=\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0+\sqrt{(1-\bar{\alpha}_t)}\,\epsilon$$

$$x_0=\frac{x_t-\sqrt{1-\alpha_t}\epsilon}{\sqrt{\alpha_t}}$$

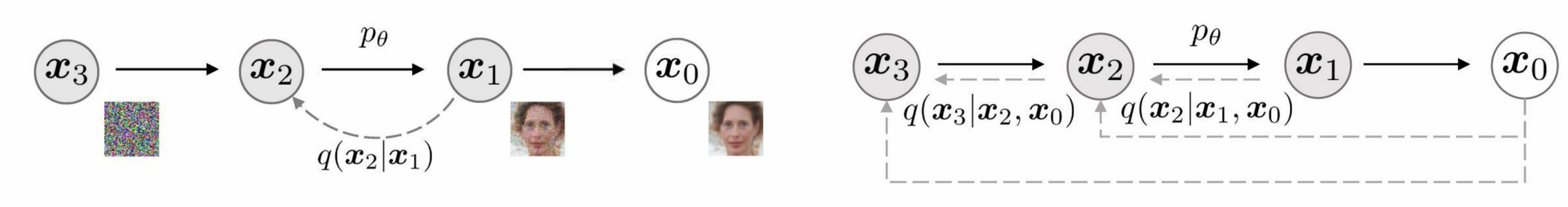
Notation changed. alpha_bar = alpha

$$\boldsymbol{x}_{t-1}=\sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\boldsymbol{x}_t-\sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{“ predicted } \boldsymbol{x}_0\text{”}}+\underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}}+\underbrace{\sigma_t\epsilon_t}_{\text{random noise}}$$

Song et al., “Denoising Diffusion Implicit Models”, ICLR 2021.

DDIM sampler

Deterministic generative process



$$p(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\hat{\mathbf{x}}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\sigma}_t^2}\cdot\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\hat{\mathbf{x}}_0}{\sqrt{1-\bar{\alpha}_t}}, \tilde{\sigma}_t^2\mathbf{I}\right)$$

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{“ predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}} + \underbrace{\sigma_t\epsilon_t}_{\text{random noise}}$$

- DDIM sampler - $\tilde{\sigma}_t^2 = 0, \forall t$

  - a deterministic generative process, with randomness from only t=T.

Song et al., “Denoising Diffusion Implicit Models”, ICLR 2021.

Denoising diffusion implicit models (DDIM)

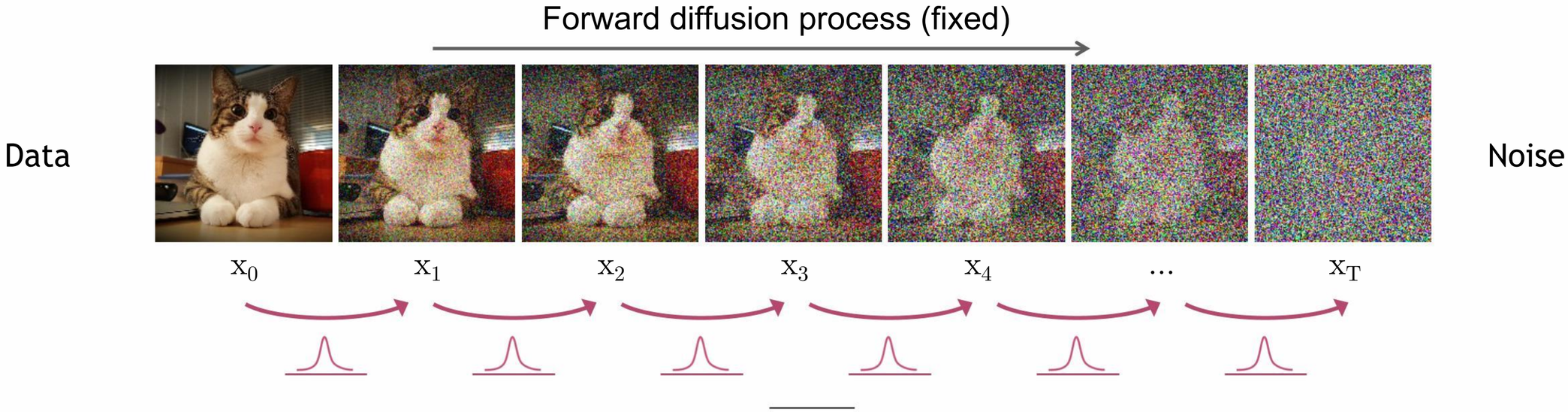
Non-Markovian diffusion process

- Can be deterministic.
- Fast sampling

Part (2):
Score-based Generative Modeling with Differential Equations

Forward Diffusion Process

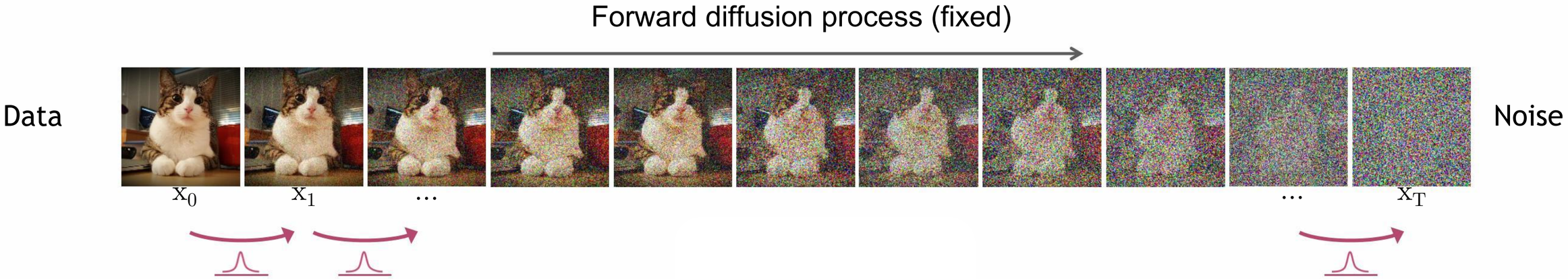
Consider the forward diffusion process again:



$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{1-\beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$$

Forward Diffusion Process

Consider the limit of **many small steps:**

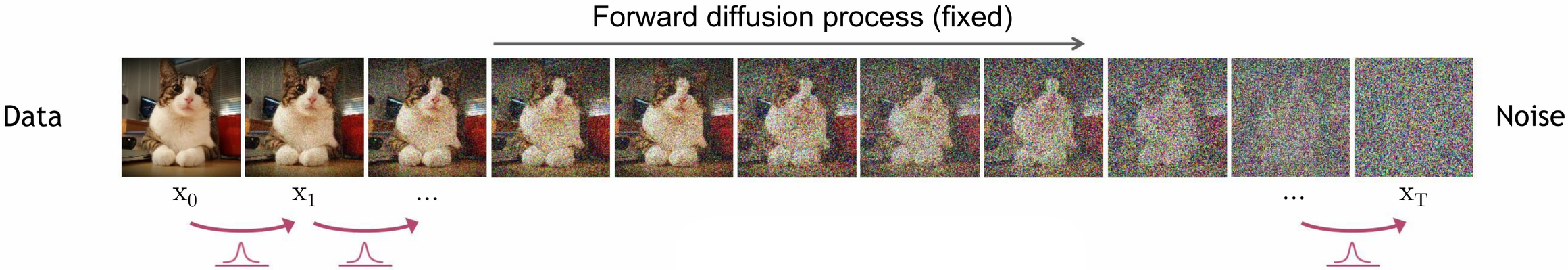


$$q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)=\mathcal{N}\left(\mathbf{x}_t ; \sqrt{1-\beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \Rightarrow \mathbf{x}_t=\sqrt{1-\beta_t} \mathbf{x}_{t-1}+\sqrt{\beta_t} \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, *ICLR*, 2021

Forward Diffusion Process

Consider the limit of **many small steps:**



$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{1-\beta(t)\Delta t}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I})
\end{aligned}$$

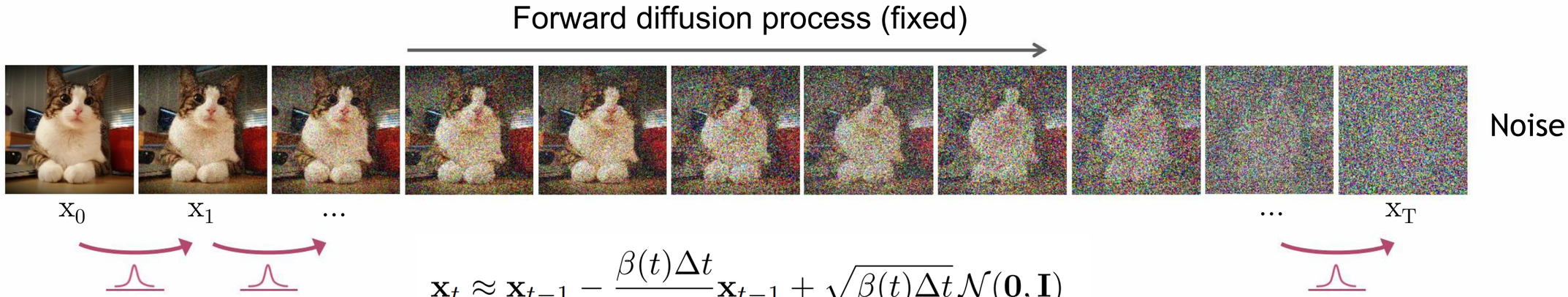
$$(\beta_t := \beta(t)\Delta t)$$

(Taylor expansion)

Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, *ICLR*, 2021

Forward Diffusion Process as Stochastic Differential Equation

Consider the limit of **many small steps:**



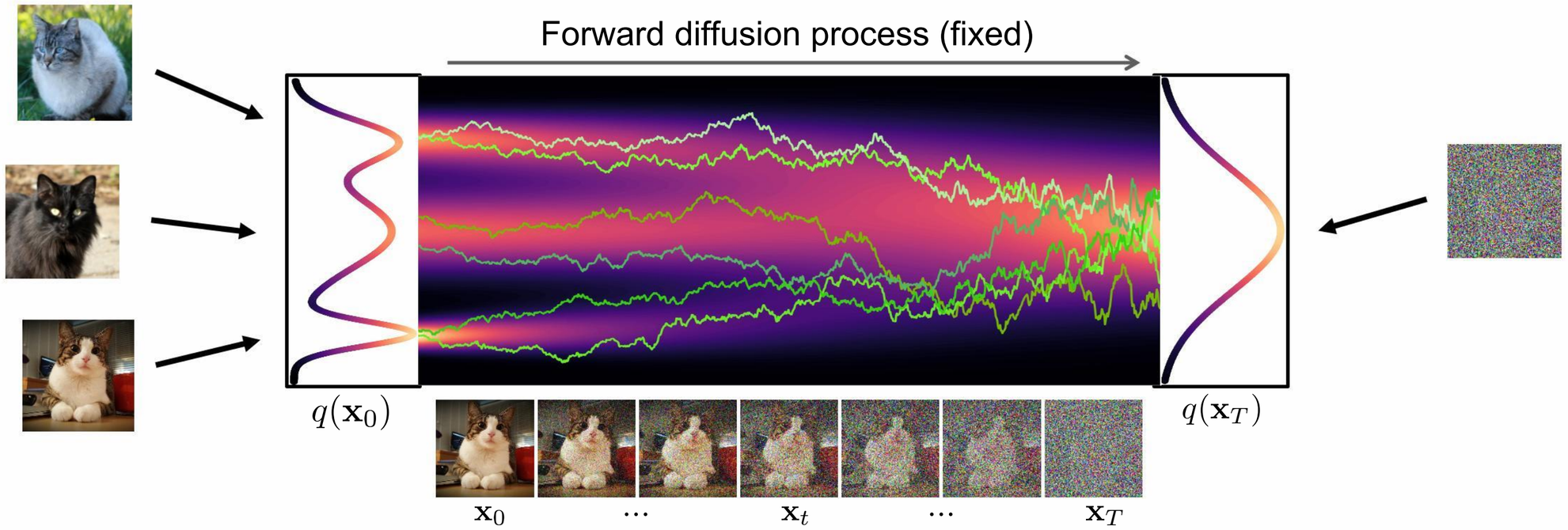
$$\mathbf{x}_t \approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Stochastic Differential Equation (SDE)
describing the diffusion in infinitesimal limit

Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, *ICLR*, 2021

Forward Diffusion Process as Stochastic Differential Equation

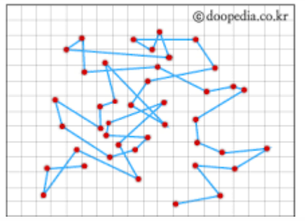


Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{diffusion term}}$$

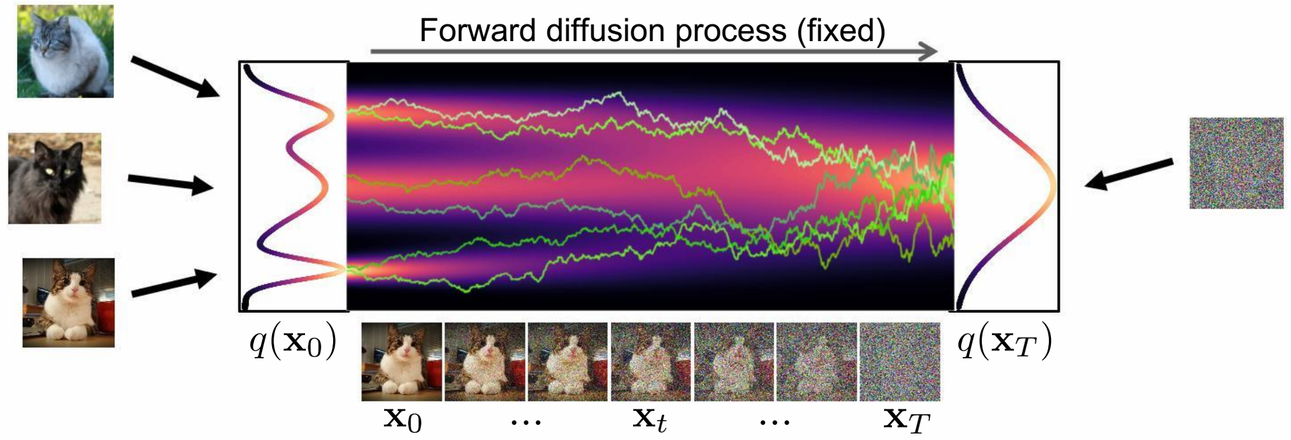
drift term
(pulls towards mode)

diffusion term
(injects noise)



Song et al., *ICLR*, 2021

Forward Diffusion Process as Stochastic Differential Equation



Forward Diffusion SDE:

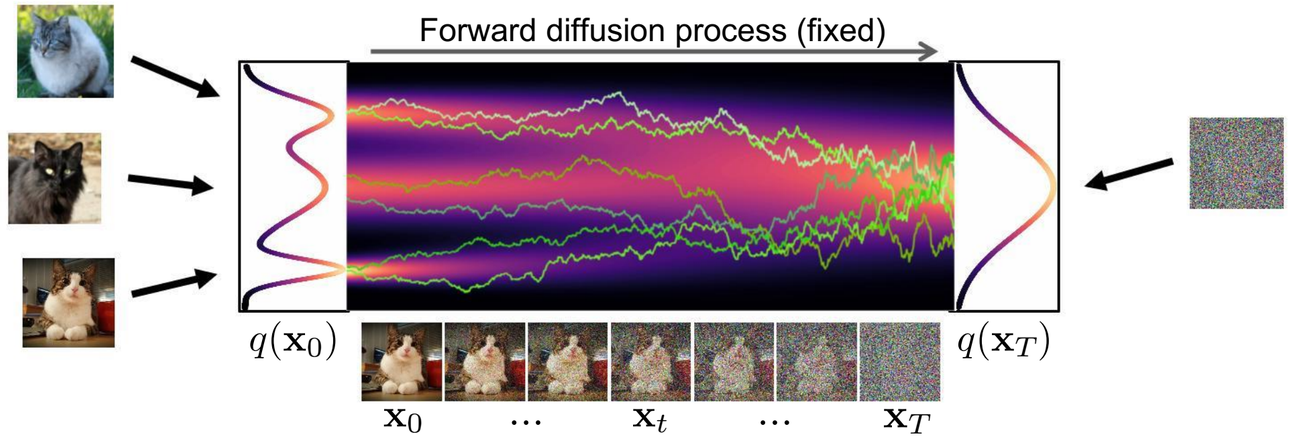
$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\text{drift term (pulls towards mode)}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\text{diffusion term (injects noise)}}$$

Special case of more general SDEs used in generative diffusion models:

$$\mathrm{d}\mathbf{x}_t = f(t)\mathbf{x}_t\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\omega}_t$$

Song et al., *ICLR*, 2021

The Generative Reverse Stochastic Differential Equation



Forward Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

Anderson 1982

Reverse Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}}\mathrm{d}t + \overbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

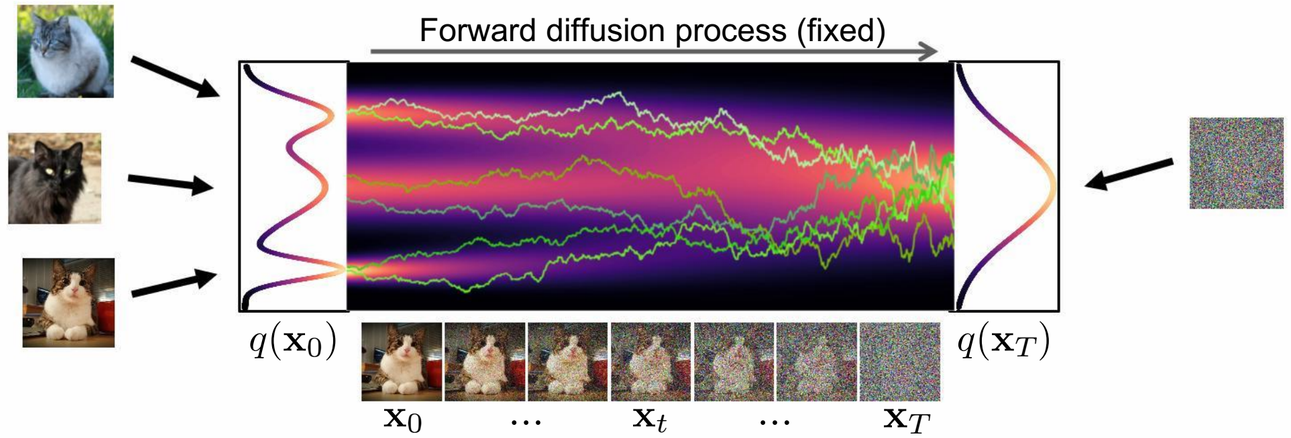
➡ **Simulate reverse diffusion process: Data generation from random noise!**

Song et al., *ICLR,* 2021
Anderson, in *Stochastic Processes and their Applications,* 1982

The Generative Reverse Stochastic Differential Equation

But how to get the score function $\nabla^{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$**?**

Score Matching



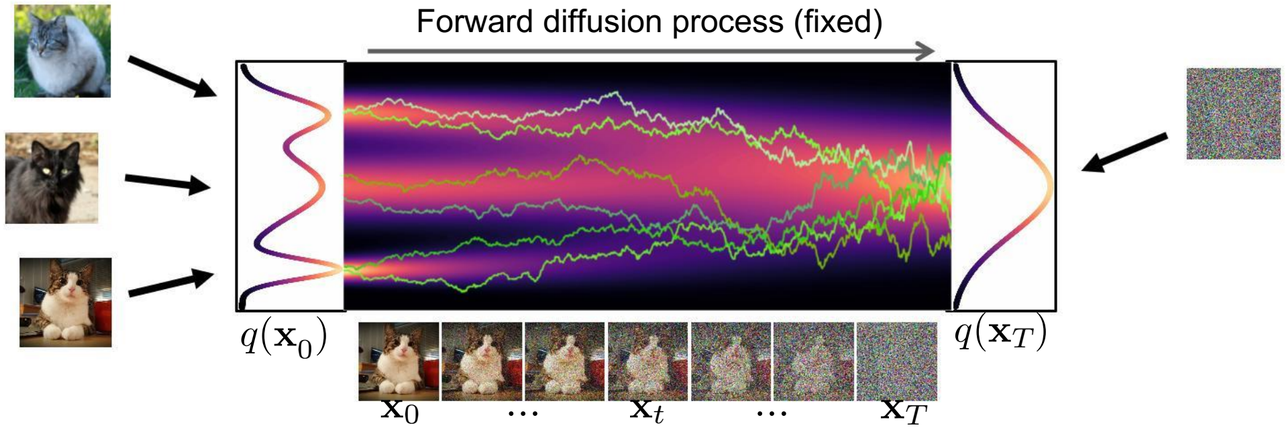
- Naïve idea, learn model for the score function by direct regression?

$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t)}}_{\text{diffused data } \mathbf{x}_t} \big|\big| \underbrace{\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)}_{\text{score of diffused data (marginal)}} \big|\big|_2^2$$

➡ **But $\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$ (score of the *marginal diffused density* $q_t(\mathbf{x}_t)$) is not tractable!**

Vincent, "A Connection Between Score Matching and Denoising Autoencoders", *Neural Computation,* 2011
Song and Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution", *NeurIPS,* 2019

Denoising Score Matching



- Instead, diffuse individual data points $\mathbf{x}^0$. Diffused $q^t(\mathbf{x}^t|\mathbf{x}^0)$ **is** tractable!
- **Denoising Score Matching:**

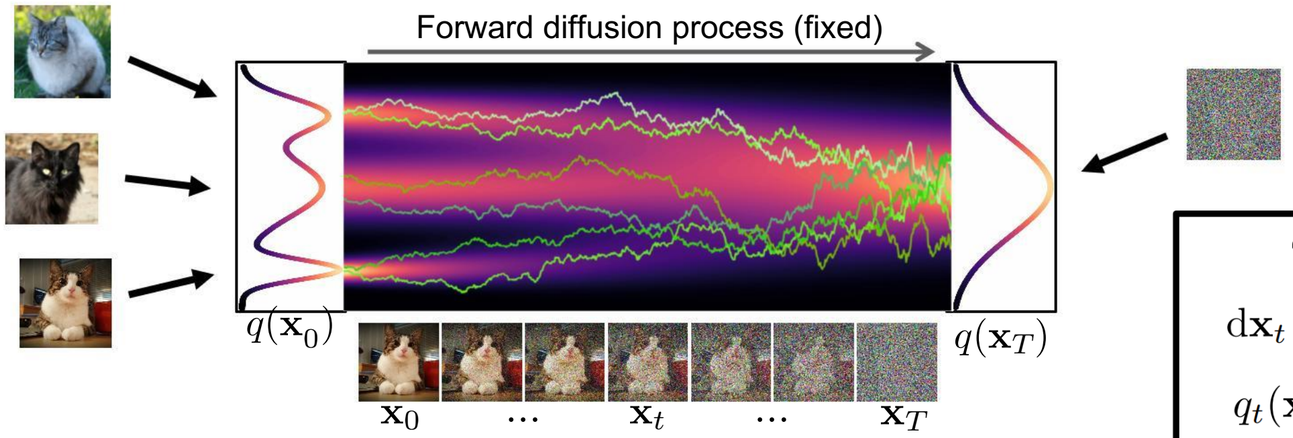
$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{t\sim\mathcal{U}(0,T)}}_{\text{diffusion time } t} \underbrace{\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}}_{\text{data sample } \mathbf{x}_0} \underbrace{\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}}_{\text{diffused data sample } \mathbf{x}_t} \Big|\Big| \underbrace{\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t|\mathbf{x}_0)}_{\text{score of diffused data sample}} \Big|\Big|_2^2$$

Vincent, in *Neural Computation,* 2011
Song and Ermon, *NeurIPS,* 2019
Song et al. *ICLR*, 2021

➡ **After expectations,** $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q_t(\mathbf{x}_t)$**!**

Denoising Score Matching

Implementation 1: Noise Prediction



"Variance Preserving" SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t$$

$$q_t(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \gamma_t\mathbf{x}_0, \sigma_t^2\mathbf{I})$$

$$\gamma_t = e^{-\frac{1}{2}\int_0^t \beta(s)ds}$$

$$\sigma_t^2 = 1 - e^{-\int_0^t \beta(s)ds}$$

- **Denoising Score Matching:**

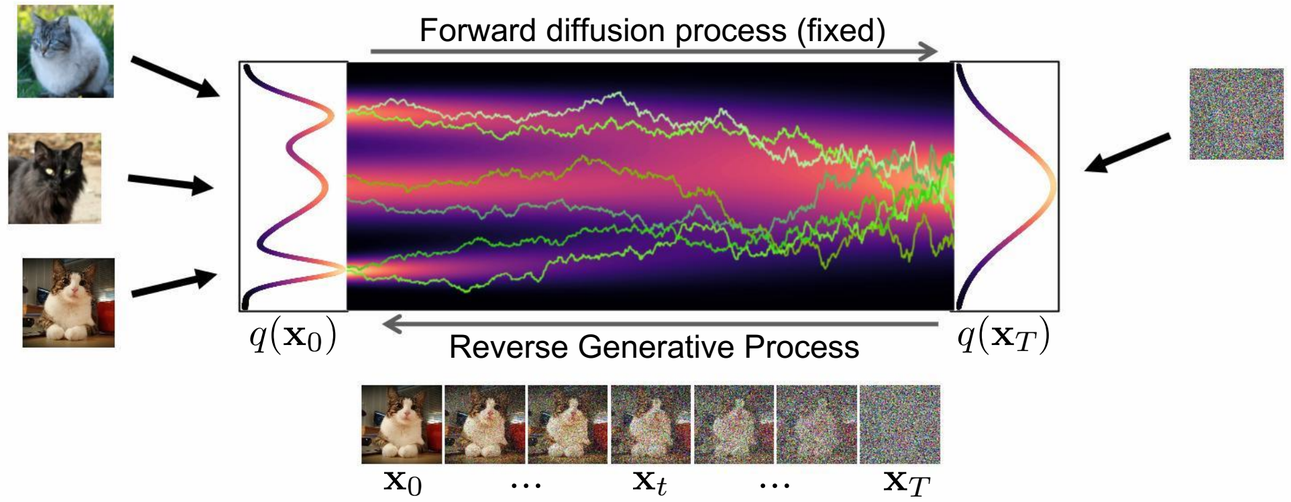
$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\mathbf{x}_t\sim q_t(\mathbf{x}_t|\mathbf{x}_0)}||\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0)||_2^2$$

  - Re-parametrized sampling: $\mathbf{x}_t = \gamma_t\mathbf{x}_0 + \sigma_t\boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$
    - Score function: $\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t|\mathbf{x}_0) = -\nabla_{\mathbf{x}_t}\frac{(\mathbf{x}_t-\gamma_t\mathbf{x}_0)^2}{2\sigma_t^2} = -\frac{\mathbf{x}_t-\gamma_t\mathbf{x}_0}{\sigma_t^2} = -\frac{\gamma_t\mathbf{x}_0+\sigma_t\boldsymbol{\epsilon}-\gamma_t\mathbf{x}_0}{\sigma_t^2} = -\frac{\boldsymbol{\epsilon}}{\sigma_t}$
  - Neural network model: $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) := -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)}{\sigma_t}$

Vincent, in *Neural Computation*, 2011
Song and Ermon, *NeurIPS*, 2019
Song et al. *ICLR*, 2021

➡ $$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\frac{1}{\sigma_t^2}||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)||_2^2$$

Probability Flow ODE



- Consider reverse generative diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

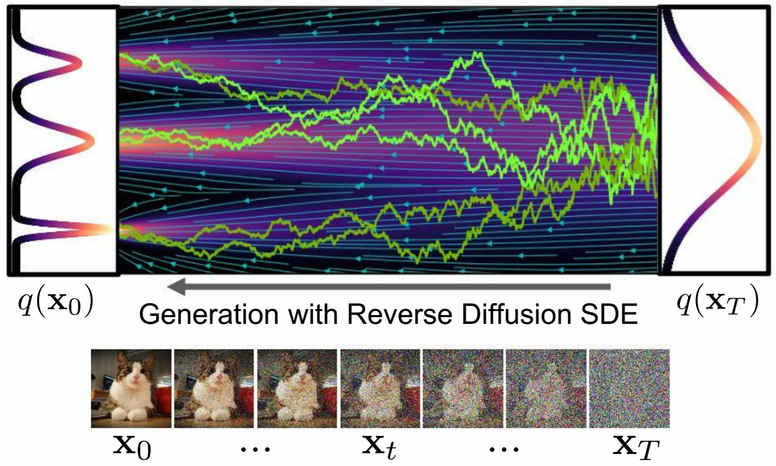
- In distribution equivalent to **"Probability Flow ODE"**: (solving this ODE results in the same $q_t(\mathbf{x}_t)$ when initializing $q^T(\mathbf{x}^T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$)

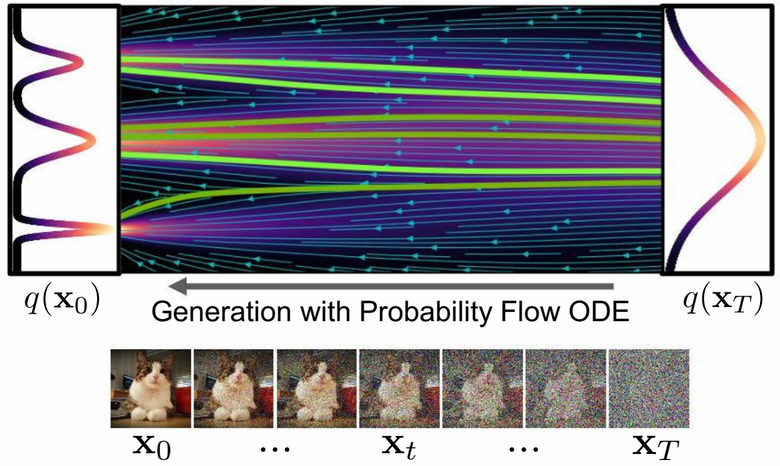
$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\right]\mathrm{d}t$$

Song et al., *ICLR*, 2021

Sampling from “Continuous-Time” Diffusion Models

How to solve the generative SDE or ODE in practice?





Generative Diffusion SDE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t$$

- *Euler-Maruyama*:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta(t)\left[\mathbf{x}_t + 2\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\Delta t + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

- *Ancestral Sampling* (Part 1) is also a generative SDE sampler!

Probability Flow ODE:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\mathrm{d}t$$

- *Euler’s Method*:

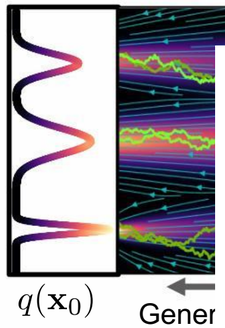
$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta(t)\left[\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\right]\Delta t$$

- In practice: Higher-Order ODE solvers (*Runge-Kutta, linear multistep methods, exponential integrators, …*)

Song et al., *ICLR*, 2021

Sampling from "Continuous-Time" Diffusion Models

How to solve the generative SDE or ODE in practice?



- Approximate the reverse SDE with our score-based model.

$$\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$$

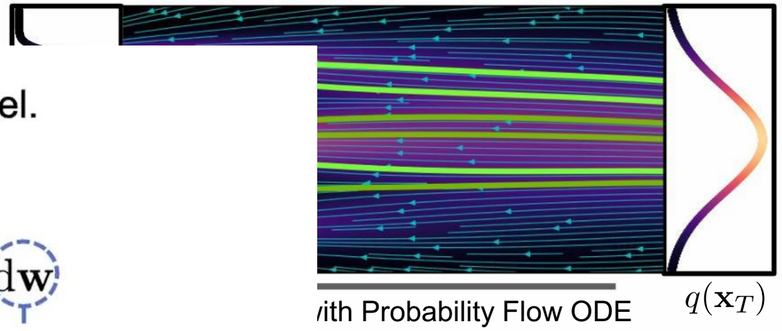
$$\mathrm{d}\mathbf{x} = [\boldsymbol{f}(\mathbf{x}, t) - g^2(t)\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)]\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}$$

- Numerical SDE solvers.
  - E.g., Euler-Maruyama solver

Initialize $\quad t \leftarrow T, \quad \mathbf{x} \sim p_T(\mathbf{x})$

$$\Delta\mathbf{x} \leftarrow [\boldsymbol{f}(\mathbf{x}, t) - g^2(t)\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)]\Delta t + g(t)\mathbf{z}$$

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\boldsymbol{I})$$



Generative Dif … **E:**

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t + \beta(t)\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)]\,\mathrm{d}t$$

➡ *Euler-Maru*

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta_t\mathbf{x}_t + \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)]\,\Delta t$$

Repea t until $t = 0$

$$\begin{cases} \Delta\mathbf{x} \leftarrow [\boldsymbol{f}(\mathbf{x}, t) - g^2(t)\boldsymbol{s}_{\boldsymbol{\theta}}(\mathbf{x}, t)]\Delta t + g(t)\mathbf{z} \\ \mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x} \\ t \leftarrow t + \Delta t \end{cases}$$

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, |\Delta t|\boldsymbol{I})$$

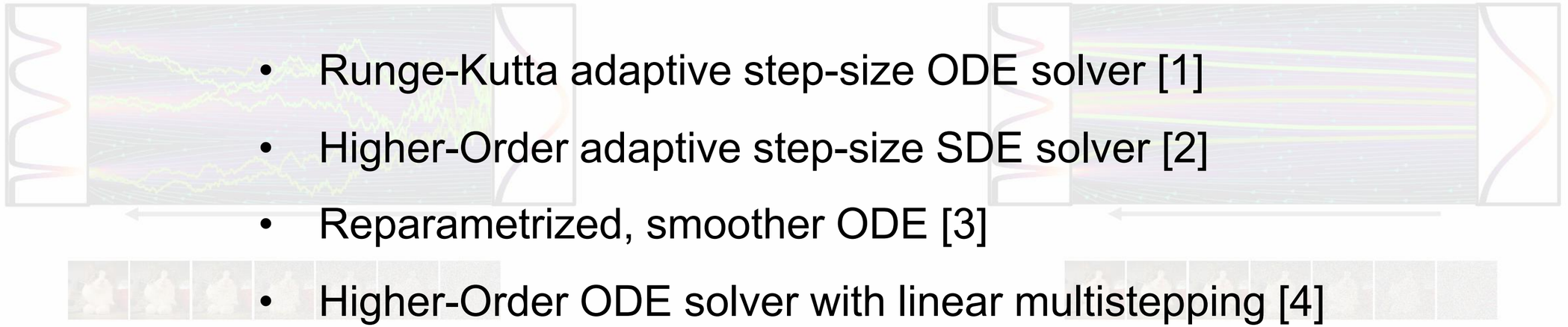
➡ *Ancestral Sampling* (Part 1) is also a generative SDE sampler!

➡ In practice: Higher-Order ODE solvers (*Runge-Kutta, linear multistep methods, exponential integrators, …*)

Song et al., *ICLR*, 2021

Sampling from “Continuous-Time” Diffusion Models

How to solve the generative SDE or ODE in practice?



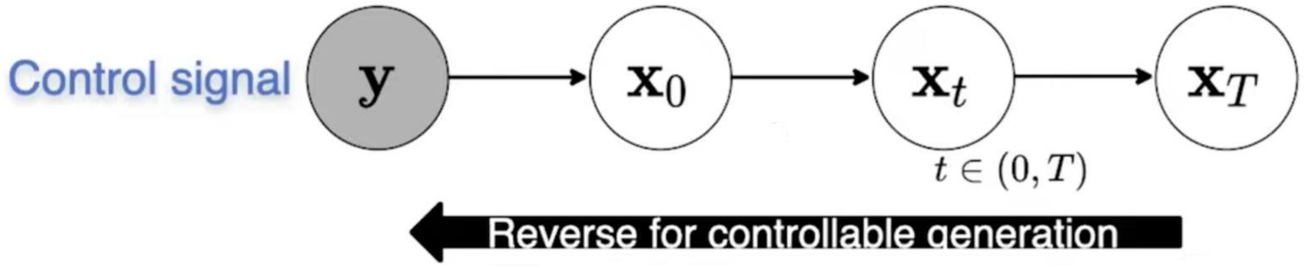
- Runge-Kutta adaptive step-size ODE solver [1]
- Higher-Order adaptive step-size SDE solver [2]
- Reparametrized, smoother ODE [3]
- Higher-Order ODE solver with linear multistepping [4]
- Exponential ODE Integrators [5,6]
- Higher-Order ODE solver with Heun’s Method [7]

[1] Song et al., “Score-Based Generative Modeling through Stochastic Differential Equations”, *ICLR*, 2021
[2] Jolicoeur-Martineau et al., “Gotta Go Fast When Generating Data with Score-Based Models”, *arXiv*, 2021
[3] Song et al., “Denoising Diffusion Implicit Models”, *ICLR*, 2021
[4] Liu et al., “Pseudo Numerical Methods for Diffusion Models on Manifolds”, *ICLR*, 2022
[5] Zhang and Chen, “Fast Sampling of Diffusion Models with Exponential Integrator”, *arXiv*, 2022
[6] Lu et al., “DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps”, *arXiv*, 2022
[7] Karras et al., “Elucidating the Design Space of Diffusion-Based Generative Models”, *arXiv*, 2022

Part 3:
Q: How to do conditional generation?

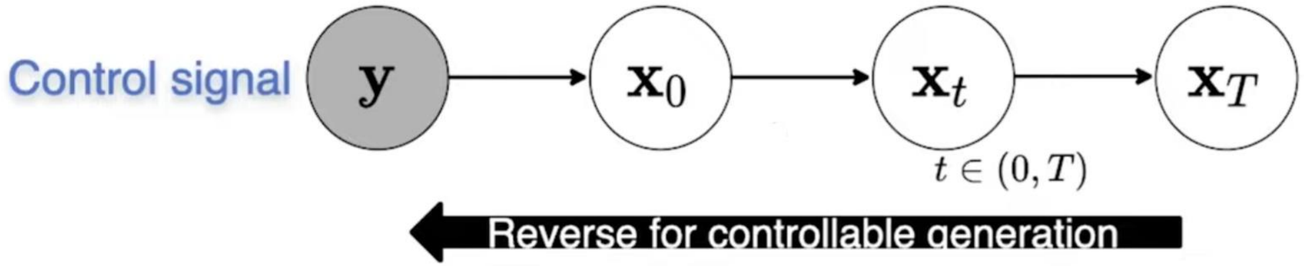
Conditional diffusion models

Include condition as input to reverse process



Conditional diffusion models

Include condition as input to reverse process

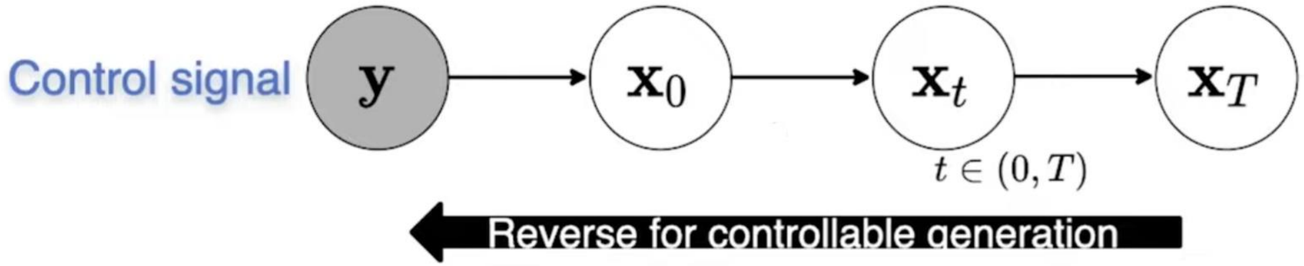


Conditional reverse-time SDE via unconditional scores

$$\mathrm{d}\mathbf{x} = [\boldsymbol{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y})]\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}$$

Conditional diffusion models

Include condition as input to reverse process



$$\mathrm{d}\mathbf{x} = [\boldsymbol{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x} \mid \mathbf{y})]\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}$$

$$\mathrm{d}\mathbf{x} = [\boldsymbol{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})]\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}$$

Unconditional score function

Score function과 별개로 학습할 수 있다.

Conditional diffusion models

Include condition as input to reverse process

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{y} \mid \mathbf{x})$$

Y is label

Pt(y|x) is time-dependent classifier

Classifier guidance

Using the gradient of a trained classifier as guidance

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

Score model

Classifier gradient

Main Idea

For class-conditional modeling of $p(\mathbf{x}_t|\mathbf{c})$, train an extra classifier $p(\mathbf{c}|\mathbf{x}_t)$

Mix its gradient with the diffusion/score model during sampling

Dhariwal and Nichol, “Diffusion models beat GANs on image synthesis”, NeurIPS 2021.

Classifier guidance

Using the gradient of a trained classifier as guidance

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

Score model

Classifier gradient

Main Idea

Sample with a modified score: $\nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega \log p(\mathbf{c}|\mathbf{x}_t)]$

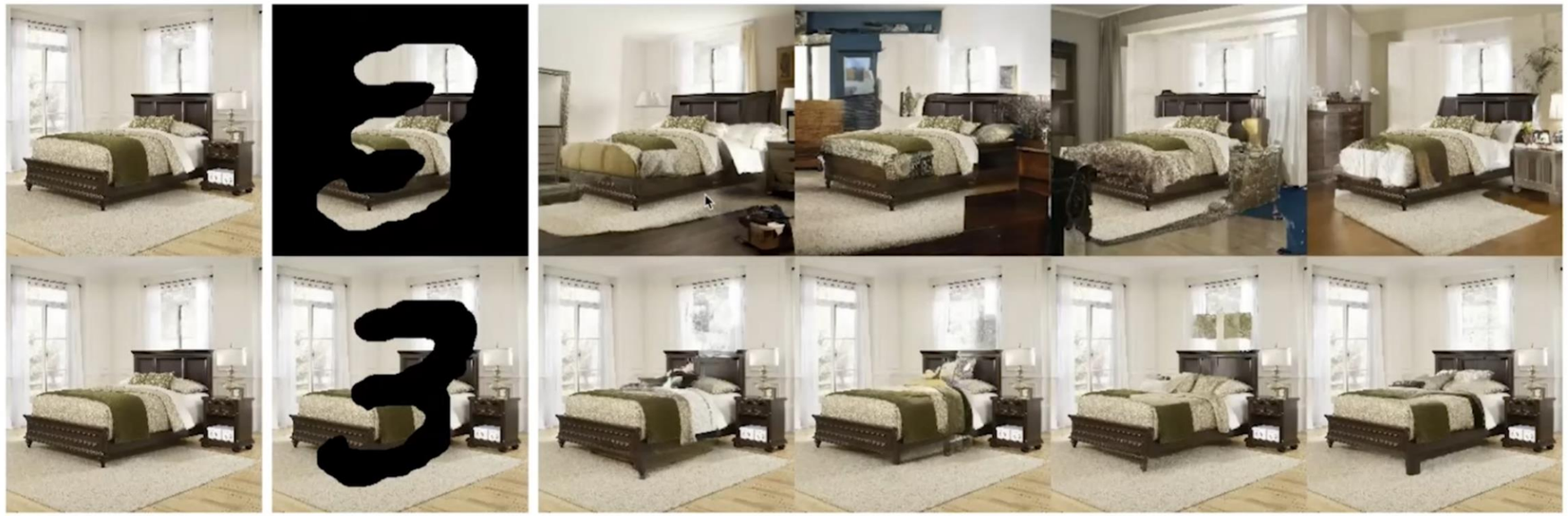
Approximate samples from the distribution $\tilde{p}(\mathbf{x}_t|\mathbf{c}) \propto p(\mathbf{x}_t|\mathbf{c})p(\mathbf{c}|\mathbf{x}_t)^\omega$

Dhariwal and Nichol, “Diffusion models beat GANs on image synthesis”, NeurIPS 2021.

Classifier guidance

Using the gradient of a trained classifier as guidance

- $\mathbf{y}$ is the **masked image**
- $p_t(\mathbf{y} \mid \mathbf{x})$ can be approximated without training



Classifier-free guidance

Get guidance by Bayes' rule on conditional diffusion models

- Instead of training an additional classifier, get an "implicit classifier" by jointly training a conditional and unconditional diffusion model:

$$p(\mathbf{c}|\mathbf{x}_t) \propto p(\mathbf{x}_t|\mathbf{c})/p(\mathbf{x}_t)$$

Conditional diffusion model

Unconditional diffusion model

- In practice, $p(\mathbf{x}_t|\mathbf{c})$ and $p(\mathbf{x}_t)$ by randomly dropping the condition of the diffusion model at certain chance.

- The modified score with this implicit classifier included is:

$$\begin{aligned}\nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega \log p(\mathbf{c}|\mathbf{x}_t)] &= \nabla_{\mathbf{x}_t}[\log p(\mathbf{x}_t|\mathbf{c}) + \omega(\log p(\mathbf{x}_t|\mathbf{c}) - \log p(\mathbf{x}_t))] \\ &= \nabla_{\mathbf{x}_t}[(1+\omega)\log p(\mathbf{x}_t|\mathbf{c}) - \omega \log p(\mathbf{x}_t)]\end{aligned}$$

Ho & Salimans, "Classifier-Free Diffusion Guidance", 2021.

Impressive conditional diffusion models

Text-to-image generation

DALL·E 2

"a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese"

IMAGEN

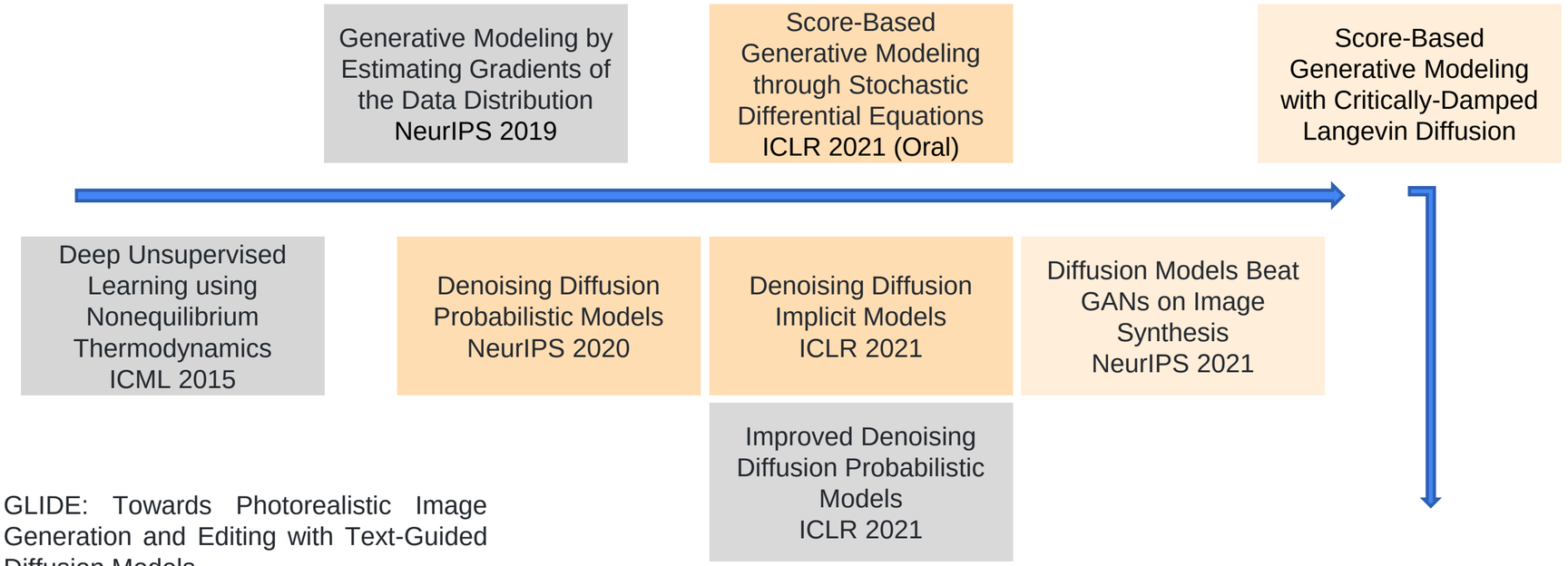
"A photo of a raccoon wearing an astronaut helmet, looking out of the window at night."

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", arXiv 2022.

Diffusion models are…

- **Better or comparable sample quality to GANs**
  - State-of-the-art performance on CIFAR-10
  - Scalable to resolution of 1024x1024 for image generation
- **Equivalence to flow models**
  - Exact likelihood computation
  - Sample editing via manipulating latent codes
- **Uniquely identifiable encoding**
- **Controllable generation**
  - Class-conditional generation, inpainting, colorization, etc.
  - No need for re-training score-based models on new tasks

Tree



GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models
arXiv 2021. [Paper]
20 Dec 2021

Palette: Image-to-Image Diffusion Models
NeurIPS 2022. [Paper]
10 Nov 2021

Hierarchical Text-Conditional Image Generation with CLIP Latents
arXiv 2022. [Paper]
13 Apr 2022

ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models
ICCV 2021 (Oral). [Paper] [Github]
6 Aug 2021

Denoising Diffusion Restoration Models
ICLR 2022(Oral). [Paper]
27 Jan 2022

SDEdit: Image Synthesis and Editing with Stochastic Differential Equations
ICLR 2022. [Paper] [Project] [Github]
2 Aug 2021

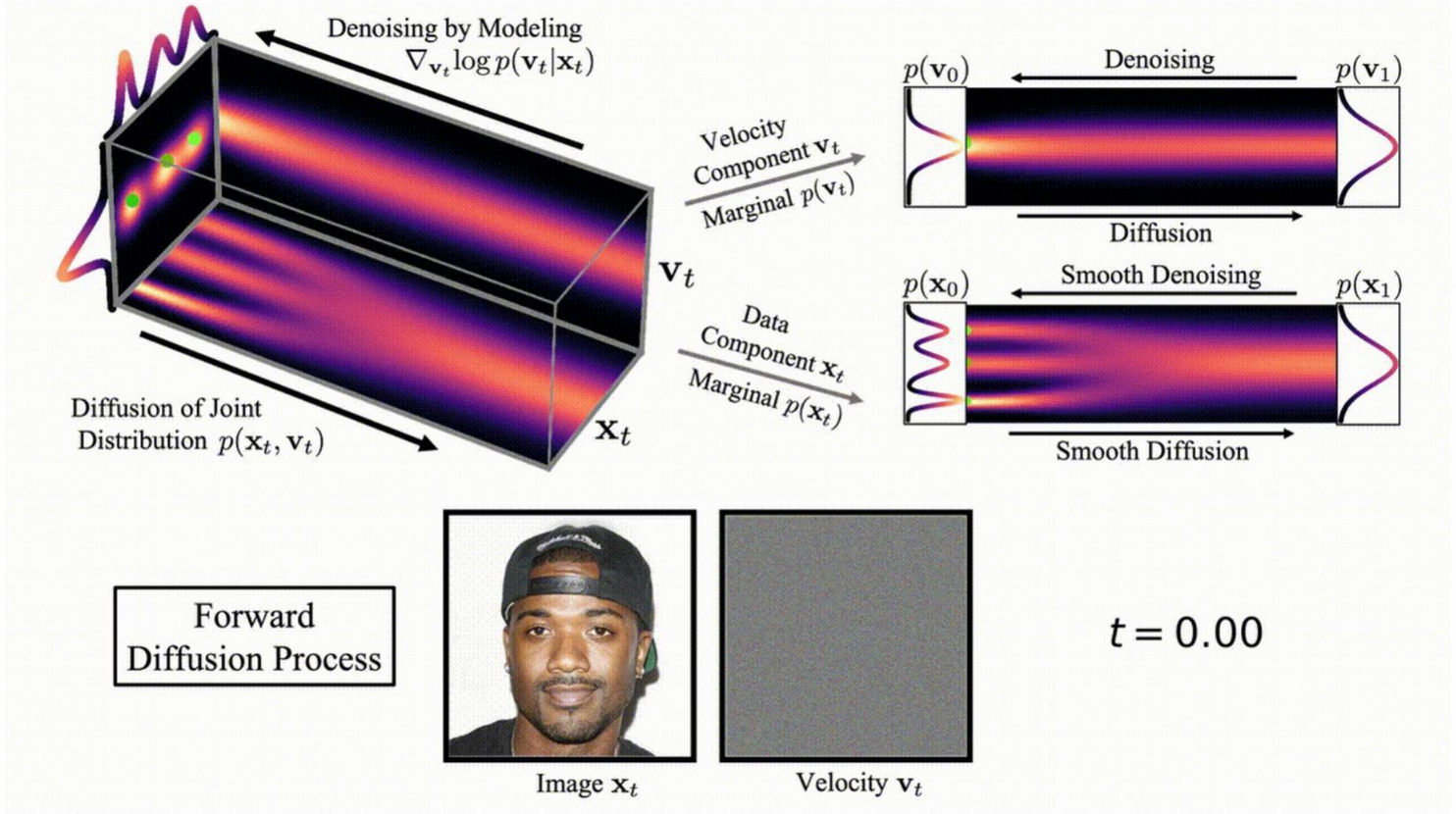
DiffuseVAE: Efficient, Controllable and High-Fidelity Generation from Low-Dimensional Latents
arXiv 2022. [Paper] [Github]
2 Jan 2022

Tackling the Generative Learning Trilemma with Denoising Diffusion GANs
arXiv 2021. [Paper] [Project]
15 Dec 2021

Diffusion Autoencoders: Toward a Meaningful and Decodable Representation
CVPR 2022 (Oral). [Paper] [Project]
30 Dec 2021

“Momentum-based” diffusion

Introduce a velocity variable and run diffusion in extended space



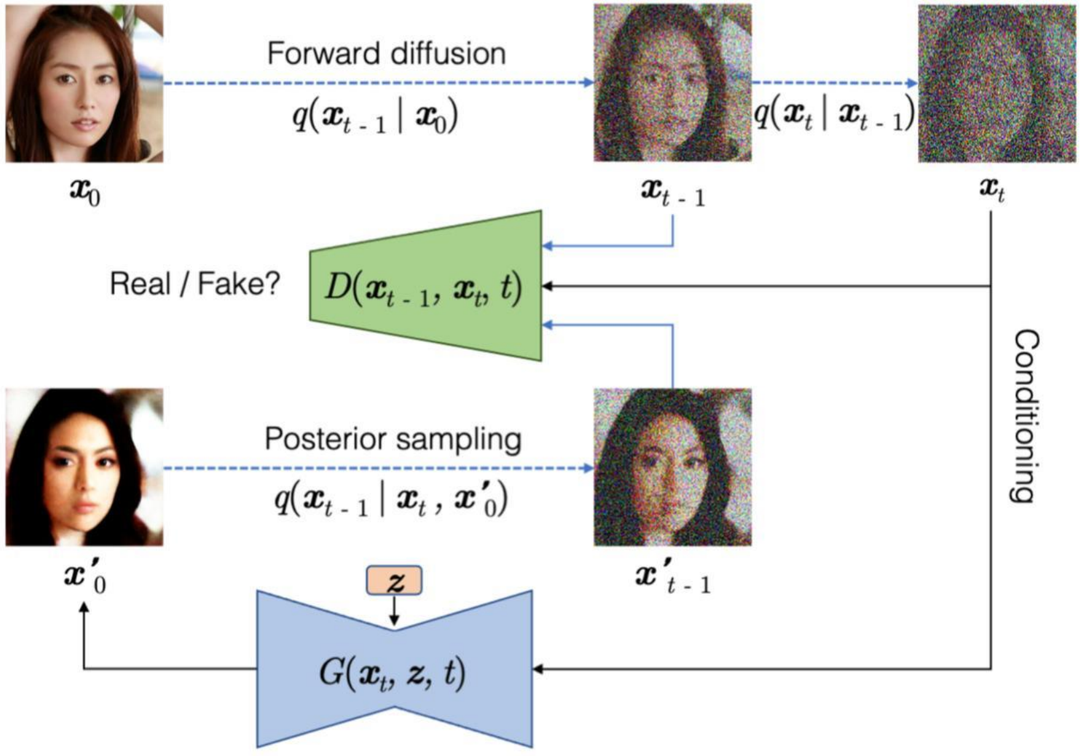
Main idea: Inject noise only into $\mathbf{v}_t$, faster mixing through Hamiltonian component!

Dockhorn et al., “Score-Based Generative Modeling with Critically-Damped Langevin Diffusion”, ICLR 2022.

Denoising diffusion GANs

Approximating reverse process by conditional GANs

$$\min_{\theta} \sum_{t \geq 1} \mathbb{E}_{q(\mathbf{x}_t)} \left[D_{\text{adv}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right]$$



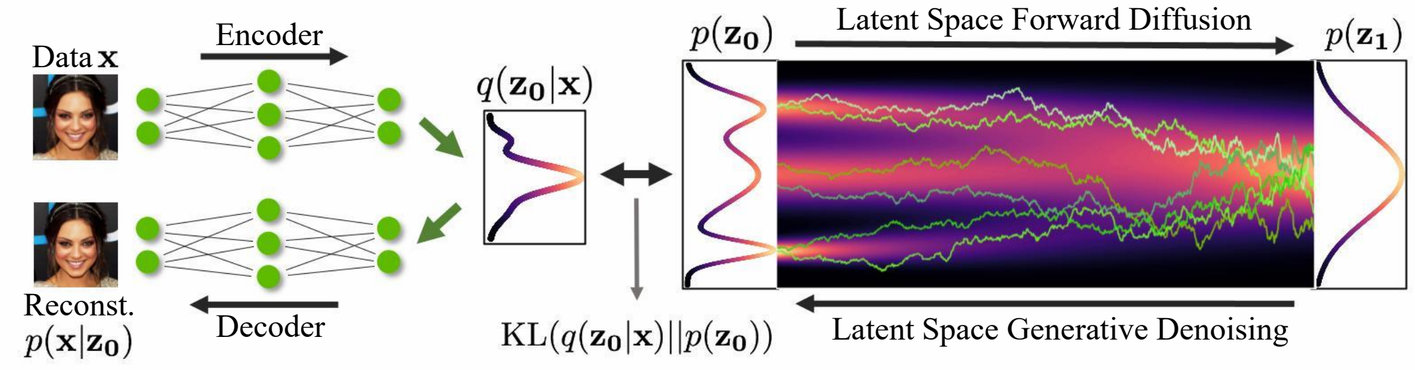
Compared to a one-shot GAN generator:

- Both generator and discriminator are solving a much simpler problem.
- Stronger mode coverage
- Better training stability

Xiao et al., “Tackling the Generative Learning Trilemma with Denoising Diffusion GANs”, ICLR 2022

Latent-space diffusion models

Variational autoencoder + score-based prior



Main Idea

Encoder maps the input data to an embedding space

Denoising diffusion models are applied in the latent space

Vahdat et al., “Score-based generative modeling in latent space”, NeurIPS 2021.
Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, CVPR 2022.

Classifier-free guidance

Trade-off for sample quality and sample diversity

Non-guidance

$\omega = 1$

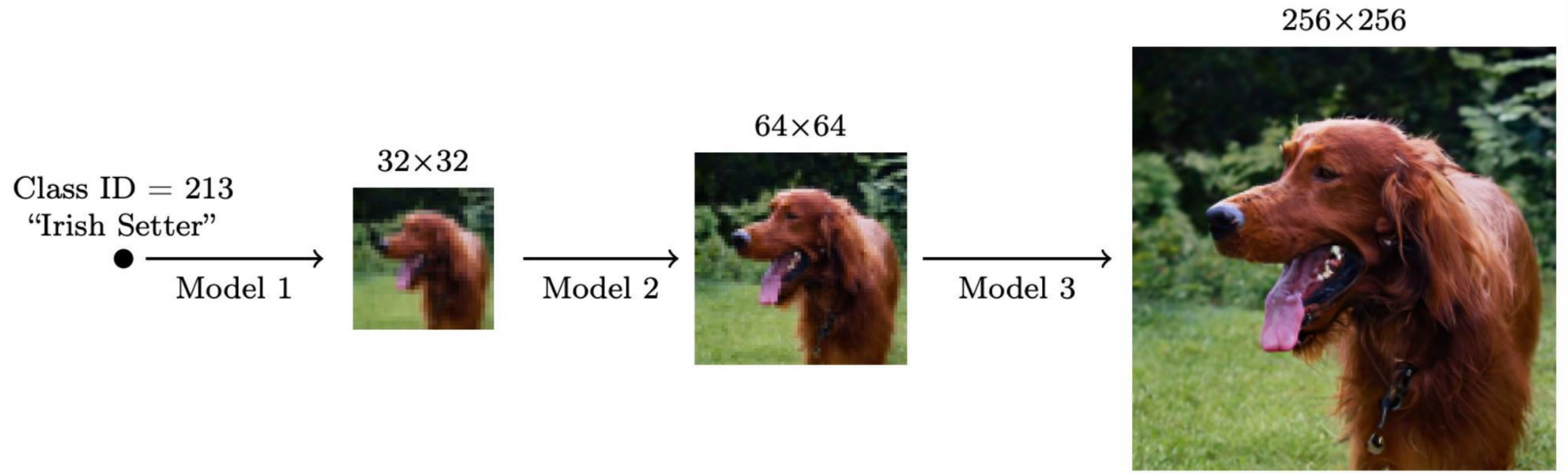
$\omega = 3$

Large guidance weight (ω) usually leads to better individual sample quality but less sample diversity.

Ho & Salimans, “Classifier-Free Diffusion Guidance”, 2021.

Cascaded generation

Pipeline



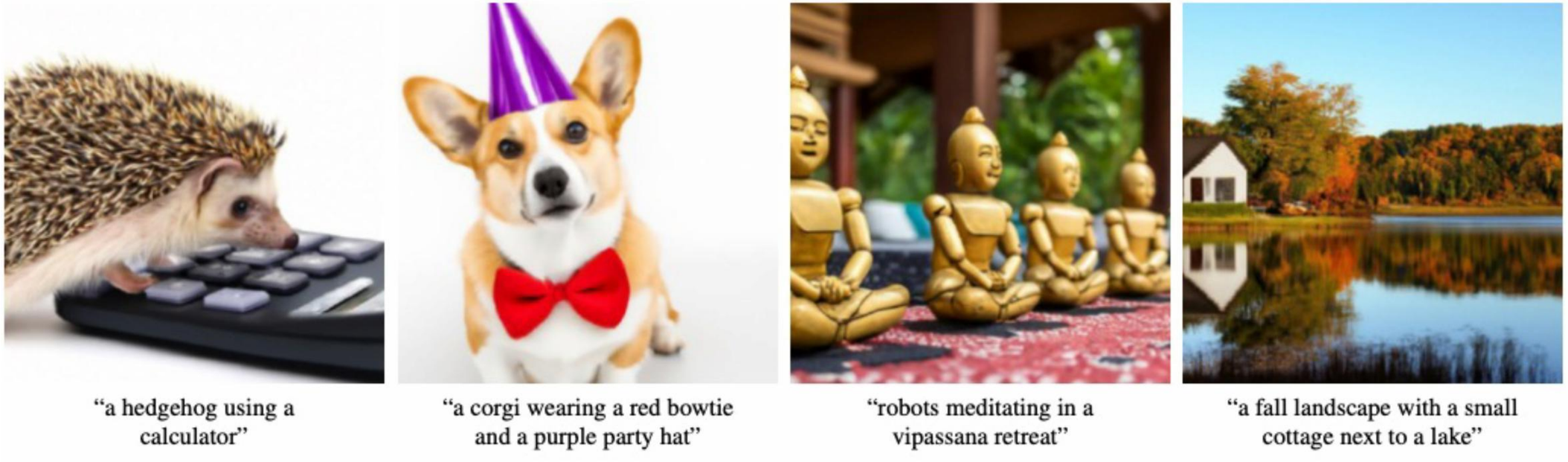
Cascaded Diffusion Models outperform Big-GAN in FID and IS and VQ-VAE2 in Classification Accuracy Score.

Ho et al., "Cascaded Diffusion Models for High Fidelity Image Generation", 2021.

GLIDE

OpenAI

- A 64x64 base model + a 64x64 →256x256 super-resolution model.
- Tried classifier-free and CLIP guidance. Classifier-free guidance works better than CLIP guidance.

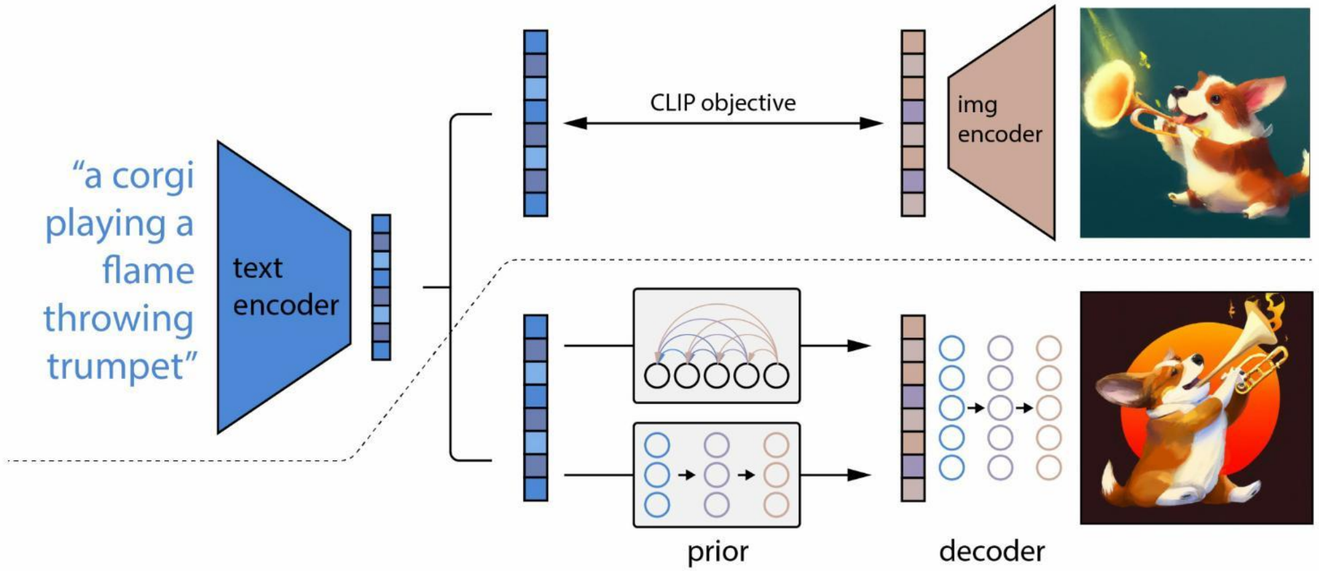


Samples generated with classifier-free guidance (256x256)

Nichol et al., "GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models", 2021

DALL·E 2

Model components



Why conditional on CLIP image embeddings?

CLIP image embeddings capture high-level semantic meaning; latents in the decoder model take care of the rest.

The bipartite latent representation` enables several text-guided image manipulation tasks.

Ramesh et al., "Hierarchical Text-Conditional Image Generation with CLIP Latents", arXiv 2022.

Imagen

Google Research, Brain team

Input: text; Output: 1kx1k images

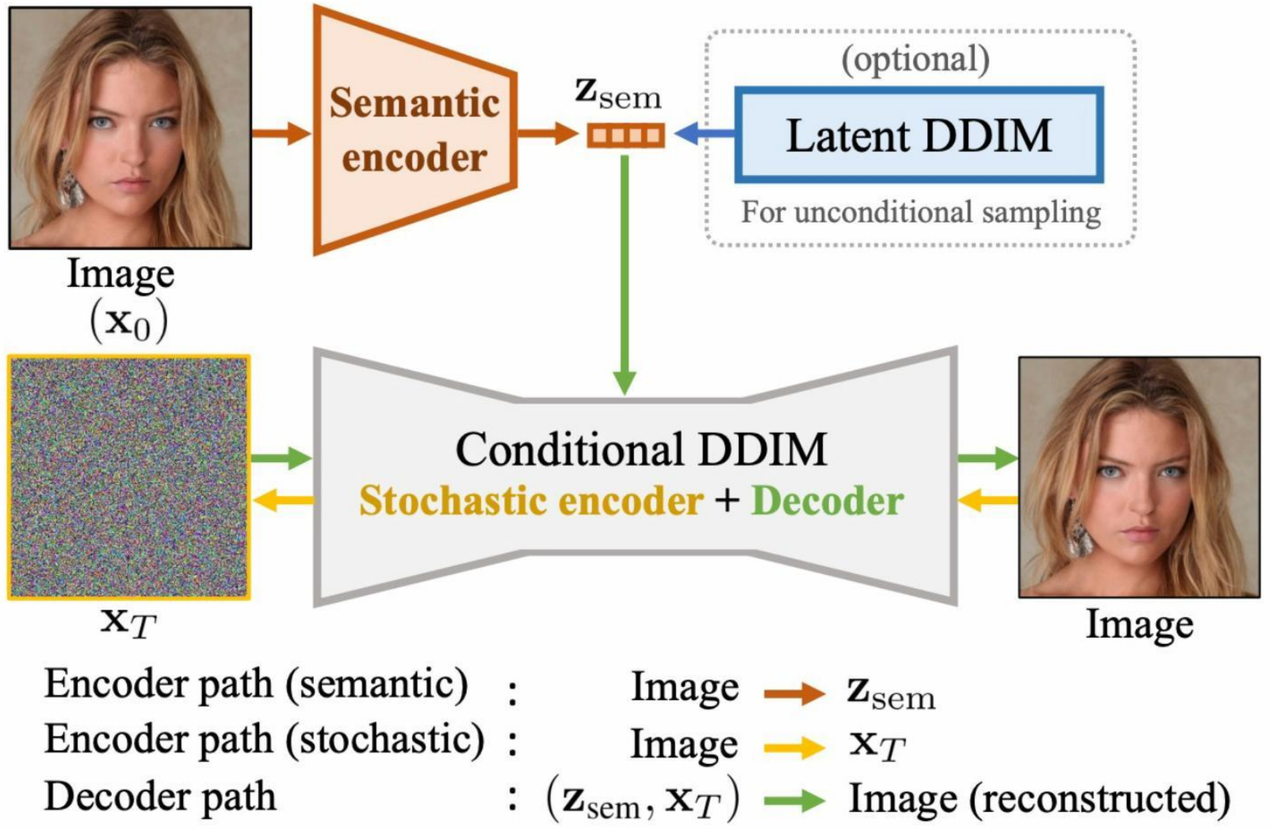
- An unprecedented degree of photorealism
  - SOTA automatic scores & human ratings
- A deep level of language understanding
- Extremely simple
  - no latent space, no quantization

A brain riding a rocketship heading towards the moon.

Saharia et al., “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”, arXiv 2022.

Diffusion Autoencoders

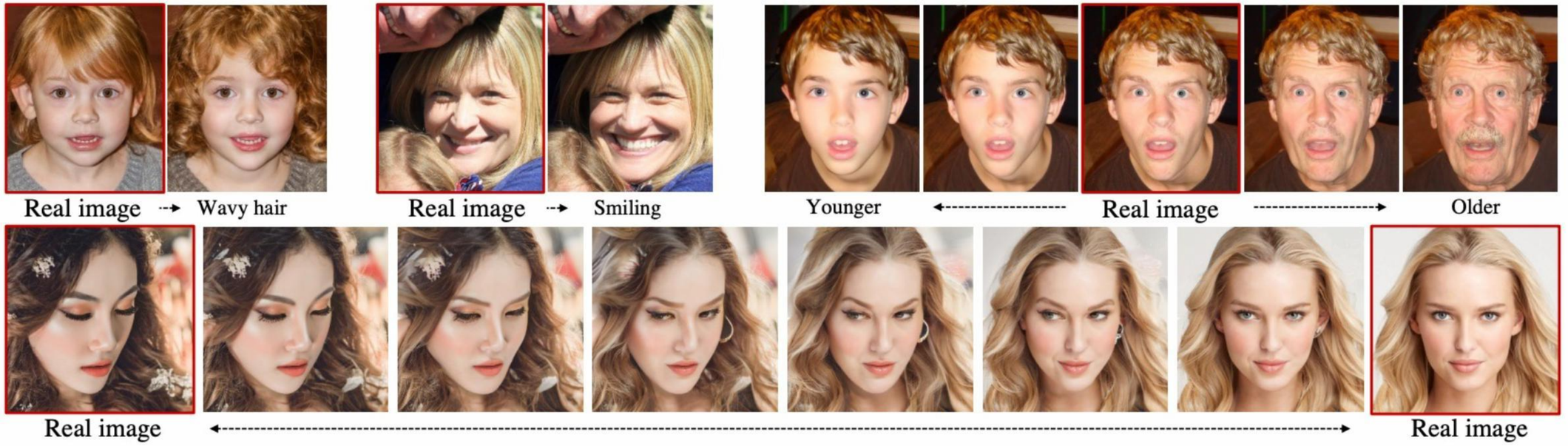
Learning semantic meaningful latent representations in diffusion models



Preechakul et al., “Diffusion Autoencoders: Toward a Meaningful and Decodable Representation”, CVPR 2022

Diffusion Autoencoders

Learning semantic meaningful latent representations in diffusion models

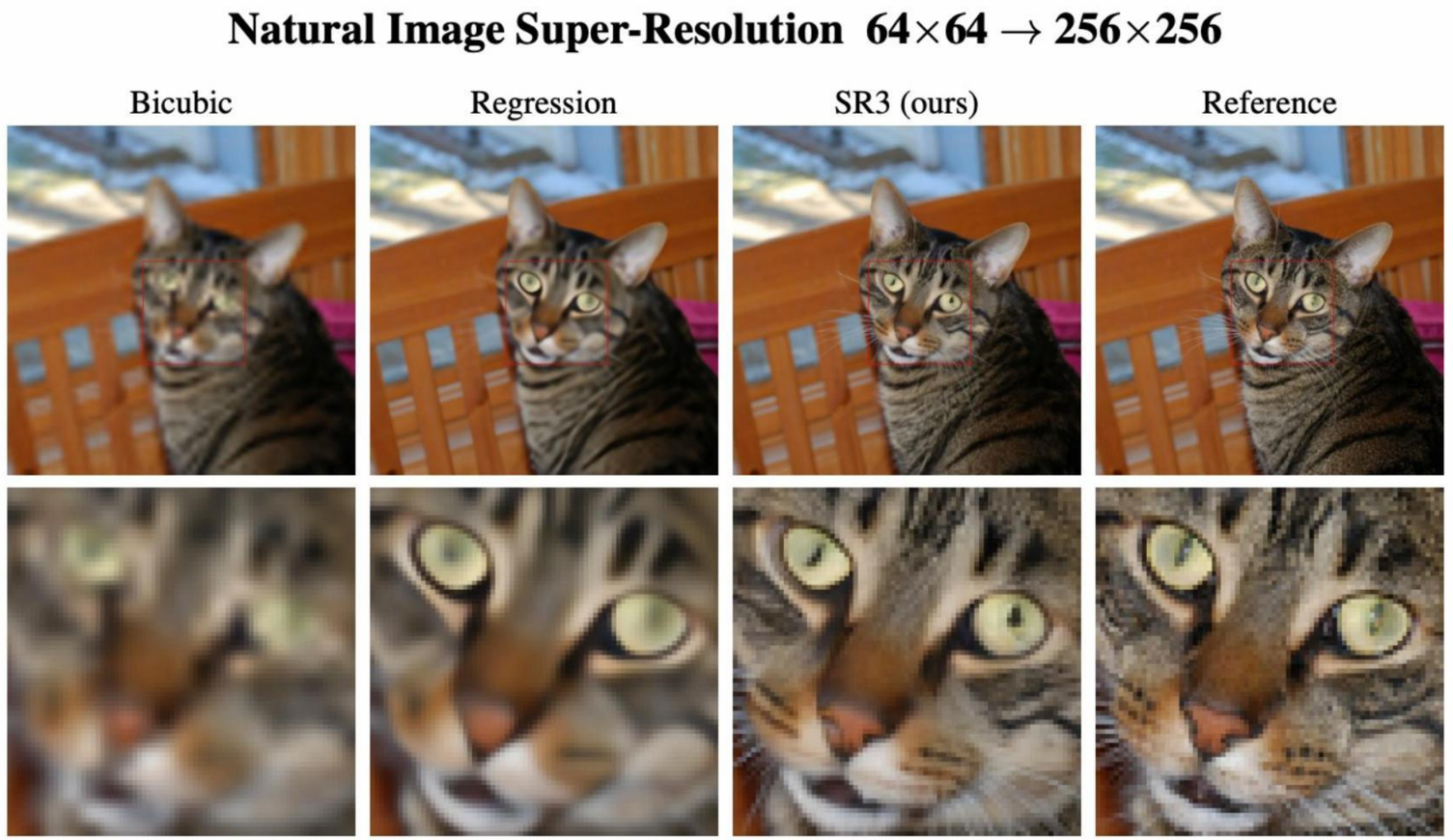


Changing the semantic latent $\mathbf{z}_{\text{sem}}$

Preechakul et al., “Diffusion Autoencoders: Toward a Meaningful and Decodable Representation”, CVPR 2022

Super-Resolution

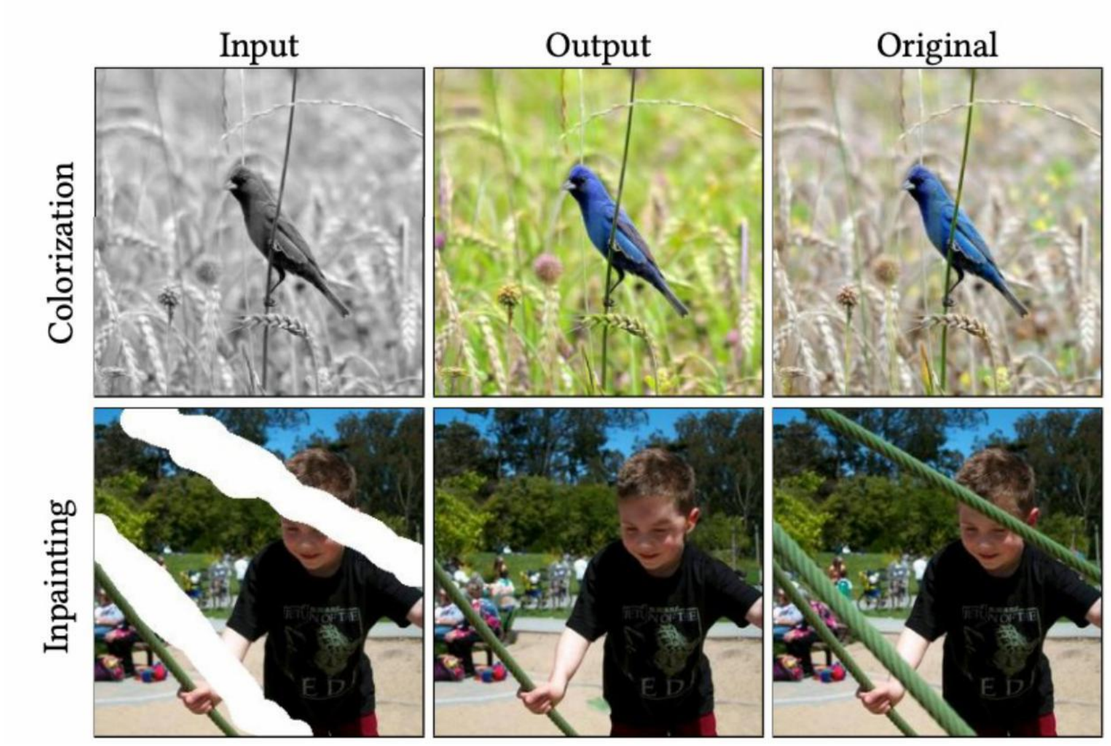
Super-Resolution via Repeated Refinement (SR3)

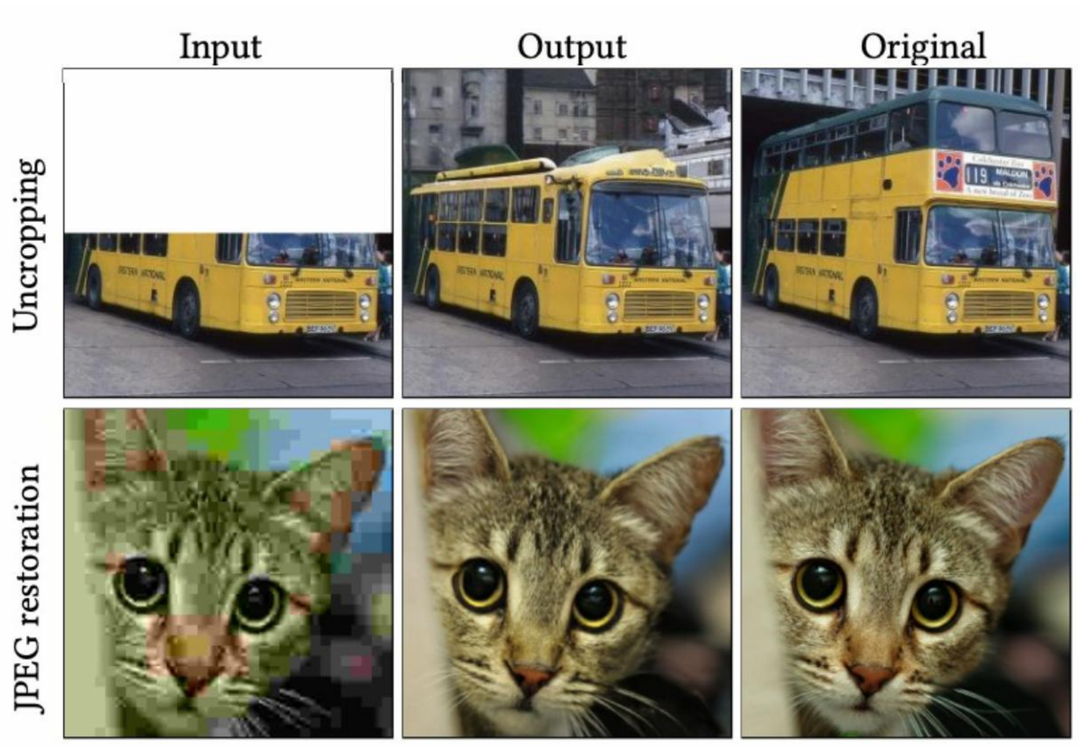


Saharia et al., Image Super-Resolution via Iterative Refinement, 2021

Image-to-Image Translation

Palette: Image-to-Image Diffusion Models

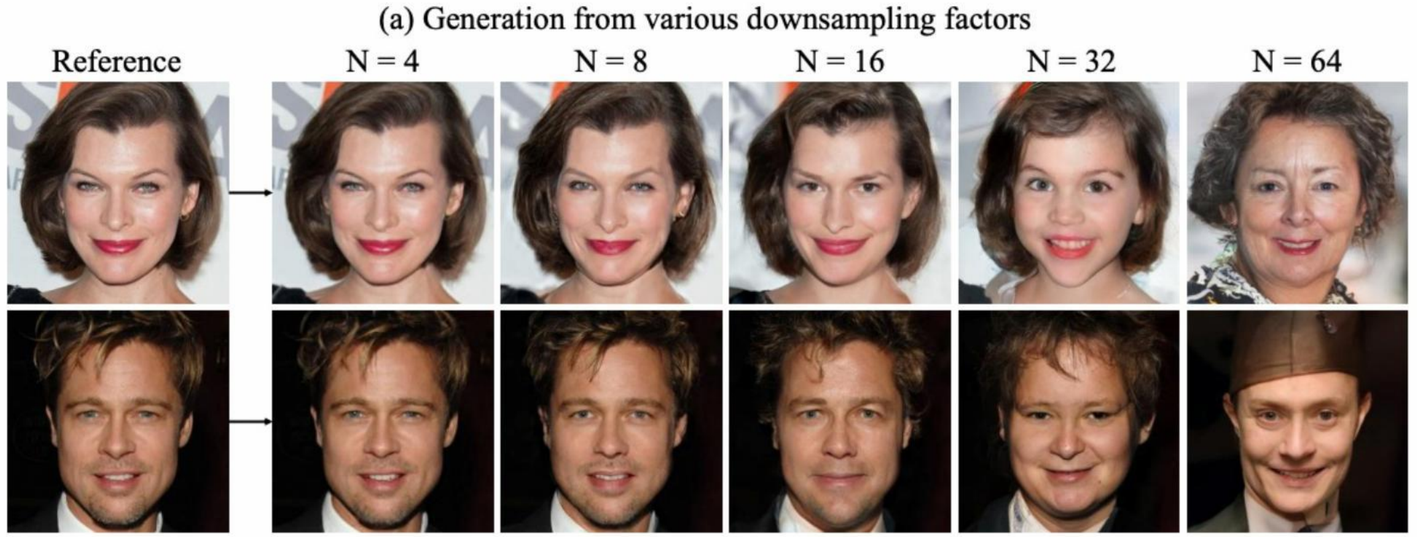


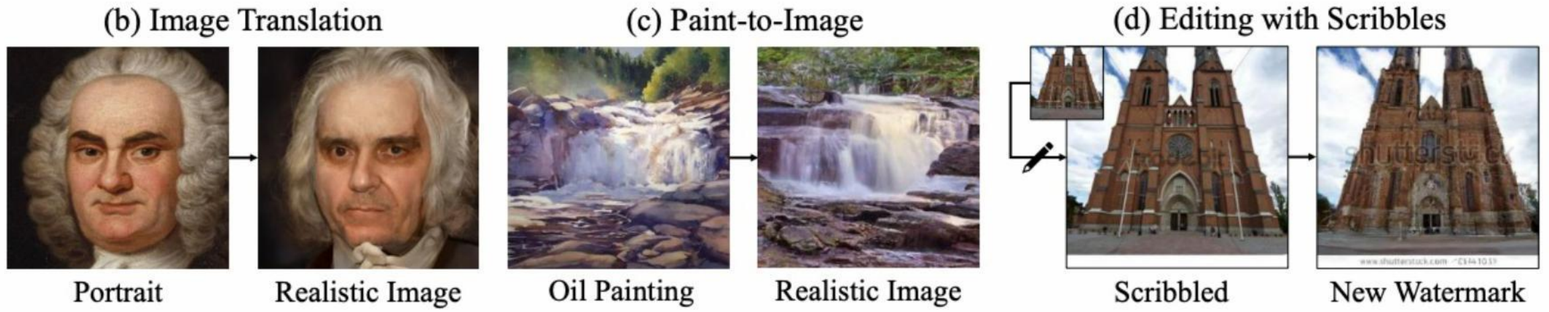


Saharia et al., Palette: Image-to-Image Diffusion Models, 2022

Conditional Generation

Iterative Latent Variable Refinement (ILVR)



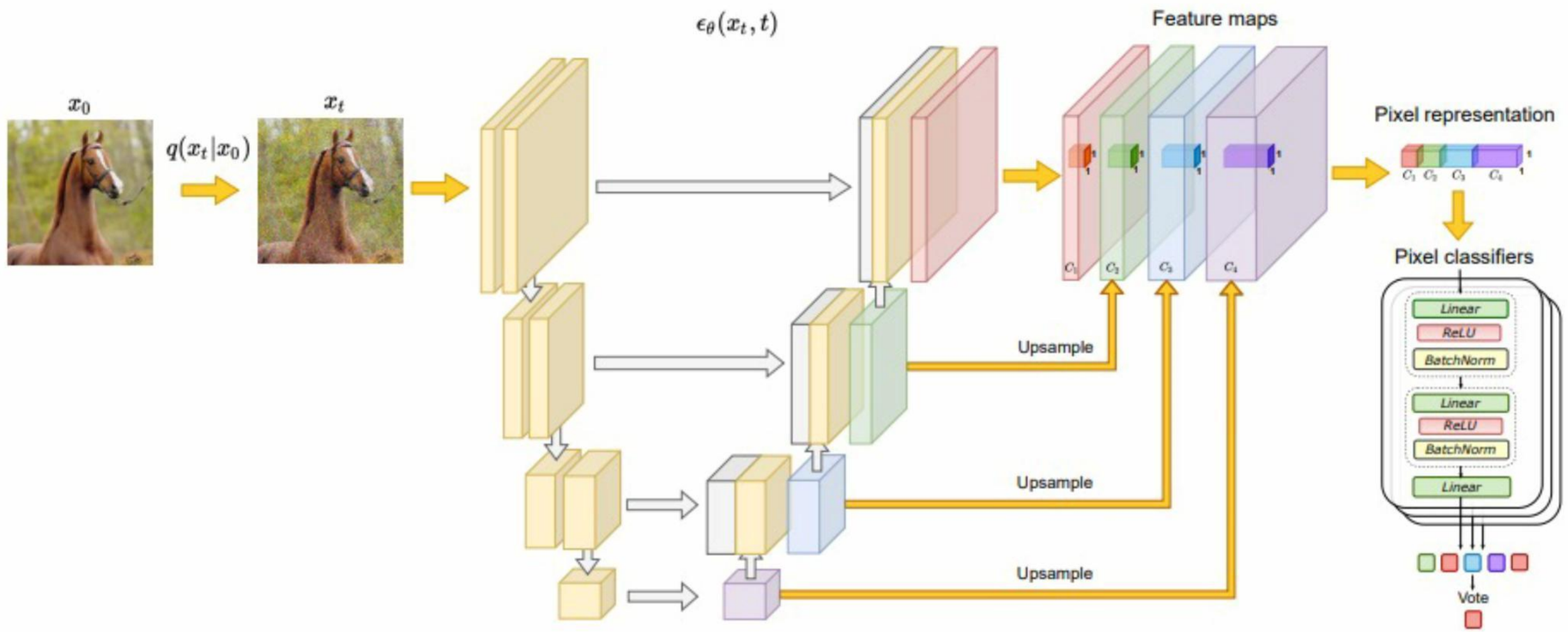


Choi et al., ILVR: Conditioning Method for Denoising Diffusion Probabilistic Models, ICCV 2021

Semantic Segmentation

Label-efficient semantic segmentation with diffusion models

Can we use representation learned from diffusion models for downstream applications such as semantic segmentation?

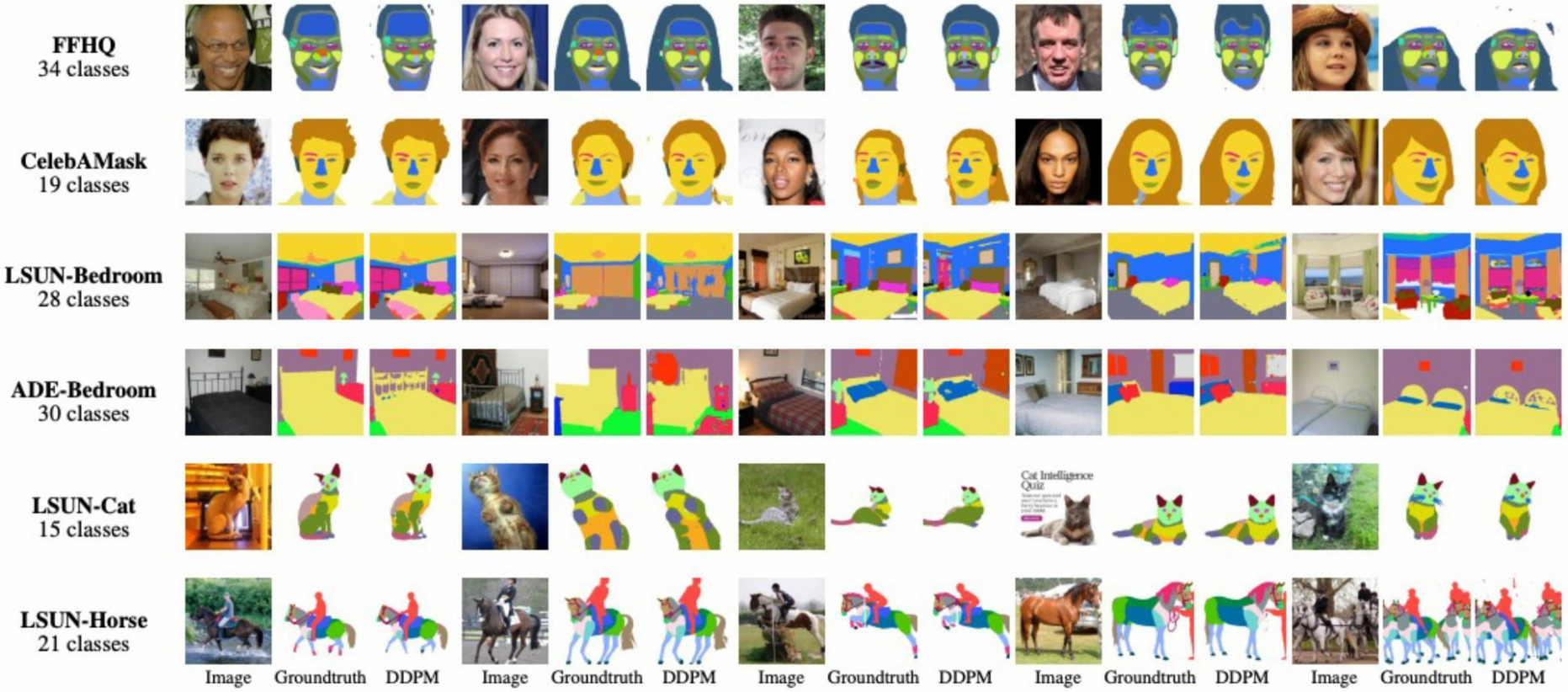


Baranchuk et al., Label-Efficient Semantic Segmentation with Diffusion Models, ICLR 2022

Semantic Segmentation

Label-efficient semantic segmentation with diffusion models

The experimental results show that the proposed method outperforms Masked Autoencoders, GAN and VAE-based models.

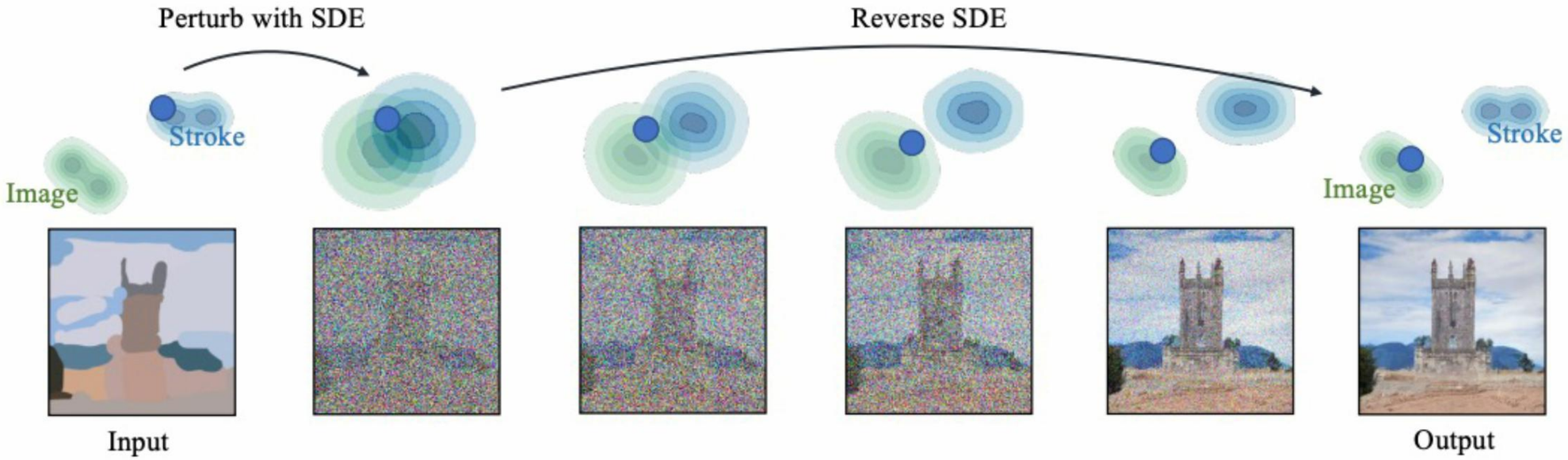


Baranchuk et al., Label-Efficient Semantic Segmentation with Diffusion Models, ICLR 2022

Image Editing

SDEdit

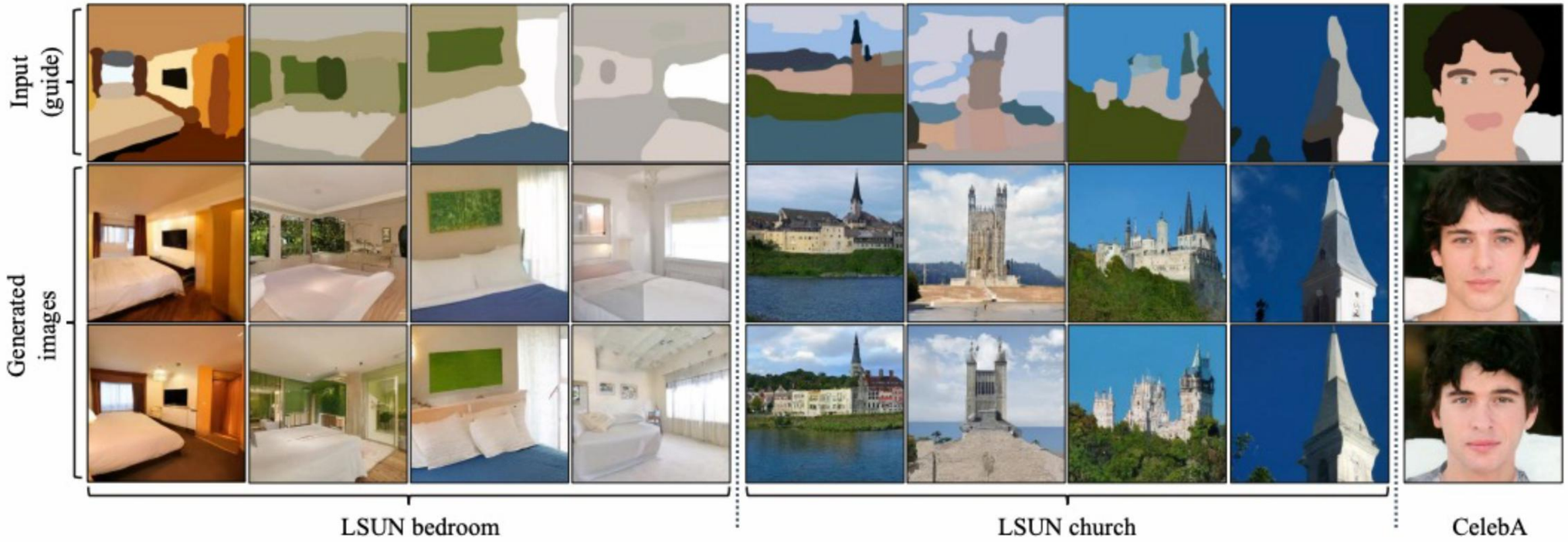
Forward diffusion brings two distributions close to each other



Meng et al., SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations, ICLR 2022

Image Editing

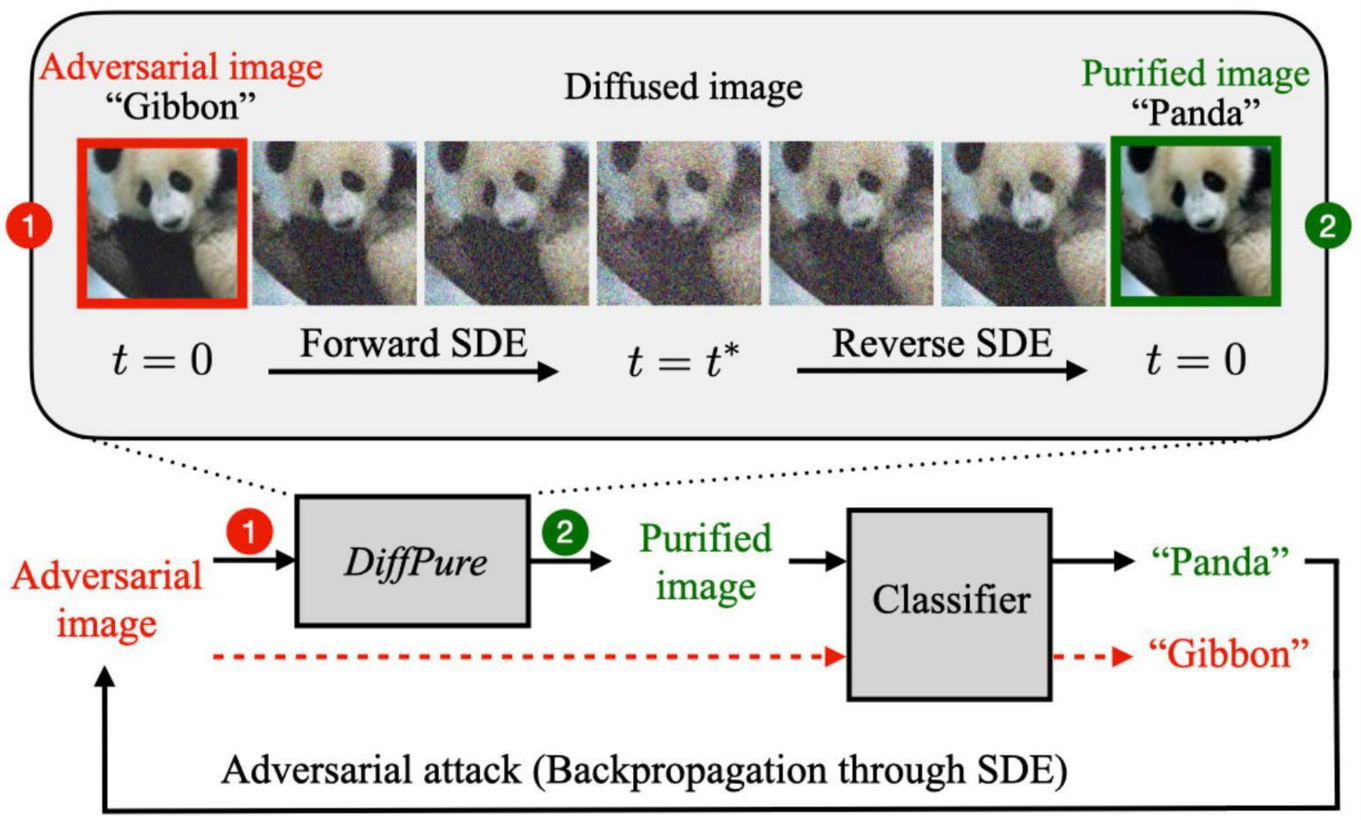
SDEdit



Meng et al., SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations, ICLR 2022

Adversarial Robustness

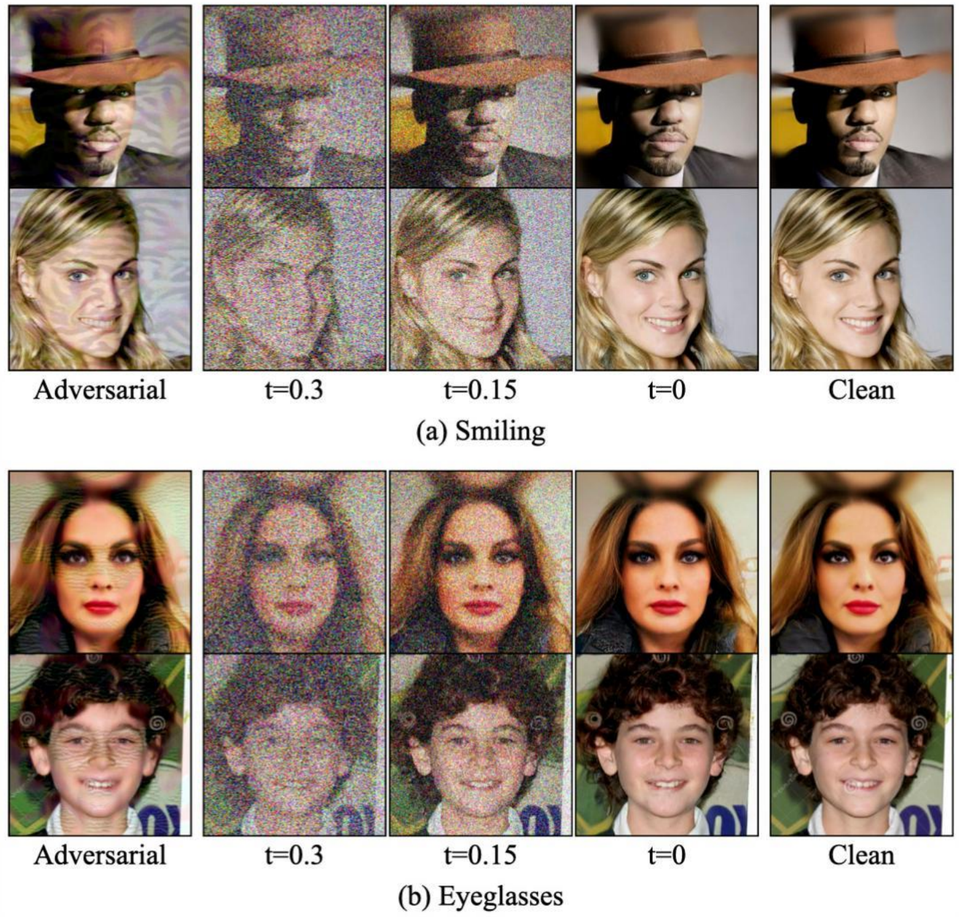
Diffusion Models for Adversarial Purification

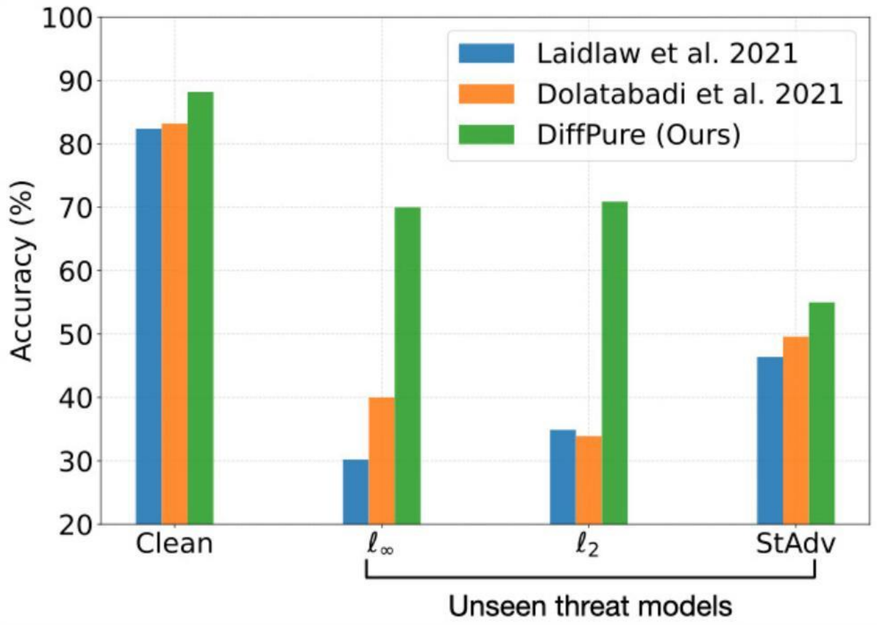


Nie et al., Diffusion Models for Adversarial Purification, ICML 2022

Adversarial Robustness

Diffusion Models for Adversarial Purification





Comparison with state-of-the-art defense methods against unseen threat models (including AutoAttack ℓ_∞, AutoAttack ℓ_2 and StAdv) on ResNet-50 for CIFAR-10.

Nie et al., Diffusion Models for Adversarial Purification, ICML 2022

Video Generation

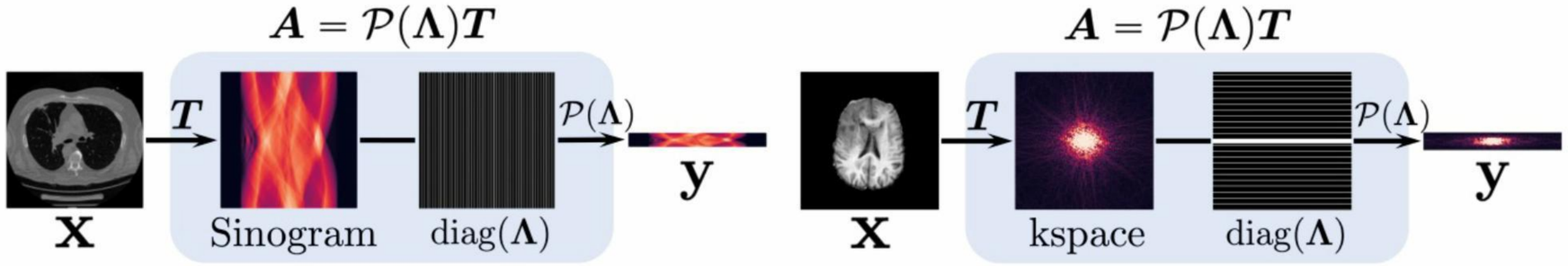
Samples from a text-conditioned video diffusion model, conditioned on the string *fireworks*.

(video from: Ho et al., “Video Diffusion Models”, *arXiv*, 2022,
https://video-diffusion.github.io/)

Ho et al., “Video Diffusion Models”, *arXiv*, 2022
Harvey et al., “Flexible Diffusion Modeling of Long Videos”, arXiv, 2022
Yang et al., “Diffusion Probabilistic Modeling for Video Generation”, *arXiv*, 2022
Höppe et al., “Diffusion Models for Video Prediction and Infilling”, arXiv, 2022
Voleti et al., “MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation”, *arXiv*, 2022

Solving Inverse Problems in Medical Imaging

Forward CT or MRI imaging process (simplified):

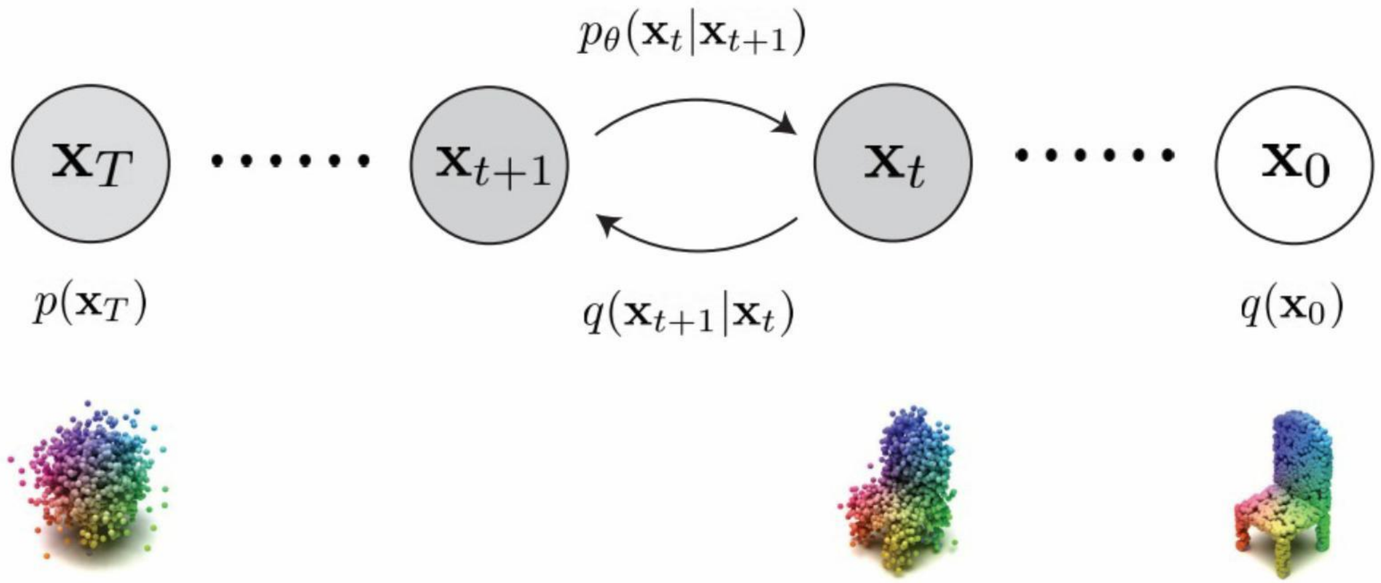


(image from: Song et al., “Solving Inverse Problems in Medical Imaging with Score-Based Generative Models”, *ICLR*, 2022)

➡ *Inverse Problem*:
Reconstruct original image from sparse measurements.

3D Shape Generation

- Point clouds as 3D shape representation can be diffused easily and intuitively
- Denoiser implemented based on modern point cloud-processing networks (PointNets & Point-VoxelCNNs)

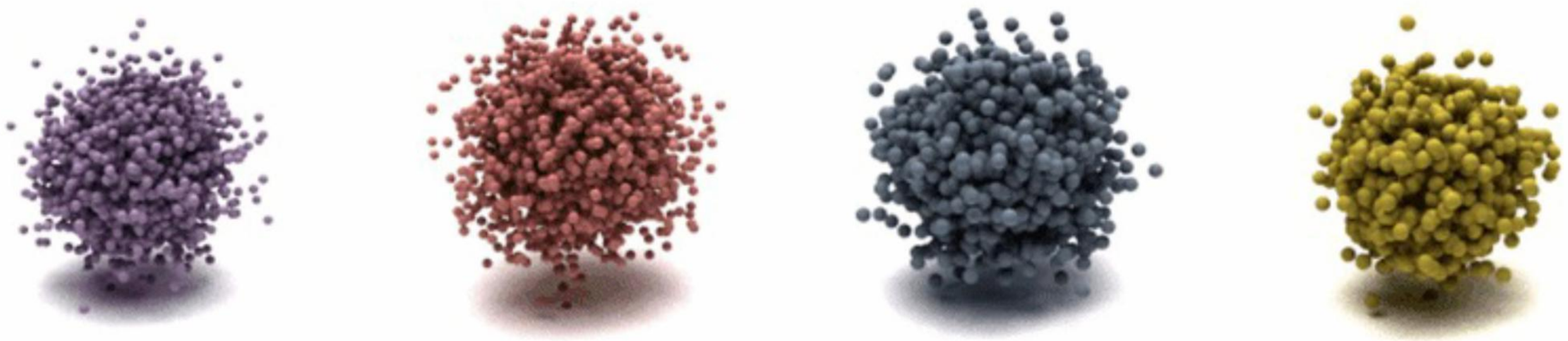


(image from: Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021)

Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021
Luo and Hu, “Diffusion Probabilistic Models for 3D Point Cloud Generation”, *CVPR*, 2021

3D Shape Generation

- Point clouds as 3D shape representation can be diffused easily and intuitively
- Denoiser implemented based on modern point cloud-processing networks (PointNets & Point-VoxelCNNs)



(video from: Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021, https://alexzhou907.github.io/pvd)

Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021

3D Shape Generation

Shape Completion

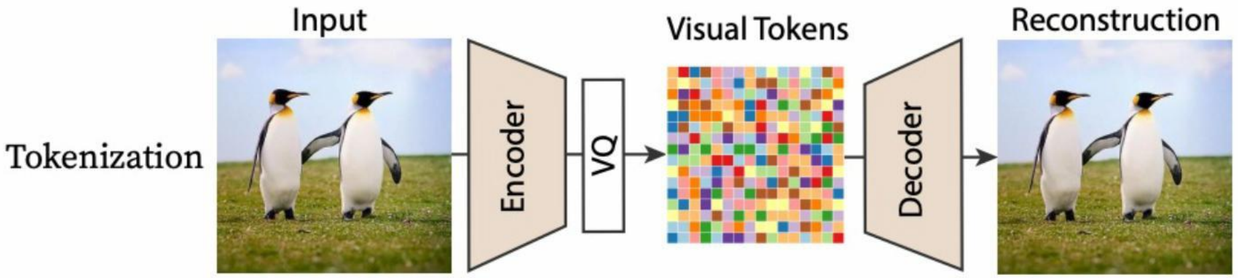
- Can train conditional shape completion diffusion model (subset of points fixed to given conditioning points):

(video from: Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021, https://alexzhou907.github.io/pvd)

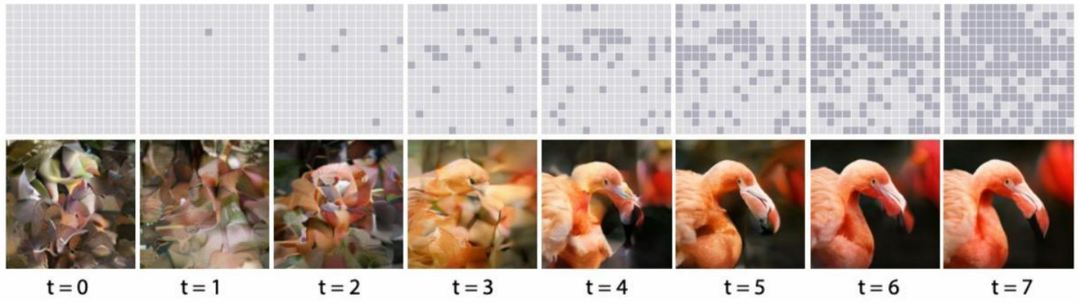
Zhou et al., “3D Shape Generation and Completion through Point-Voxel Diffusion”, *ICCV*, 2021

Discrete State Diffusion Models

Modeling Discrete Image Encodings



Encoding images into latent space with discrete tokens, and modeling discrete token distribution



Iterative generation

Class-conditional model samples

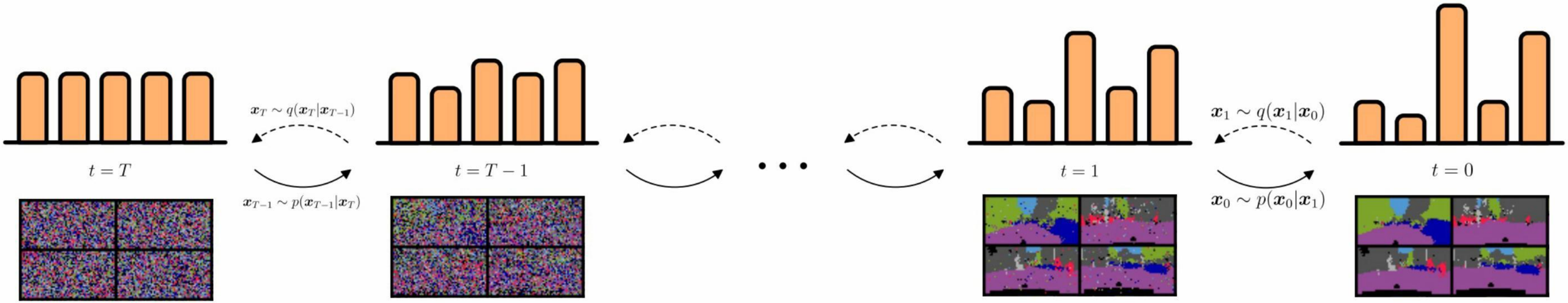
(images from: Chang et al., “MaskGIT: Masked Generative Image Transformer”, *CVPR*, 2022)

Chang et al., “MaskGIT: Masked Generative Image Transformer”, *CVPR*, 2022

Esser et al., “ImageBART: Bidirectional Context with Multinomial Diffusion for Autoregressive Image Synthesis”, *NeurIPS*, 2021

Discrete State Diffusion Models

Modeling Pixel-wise Segmentations



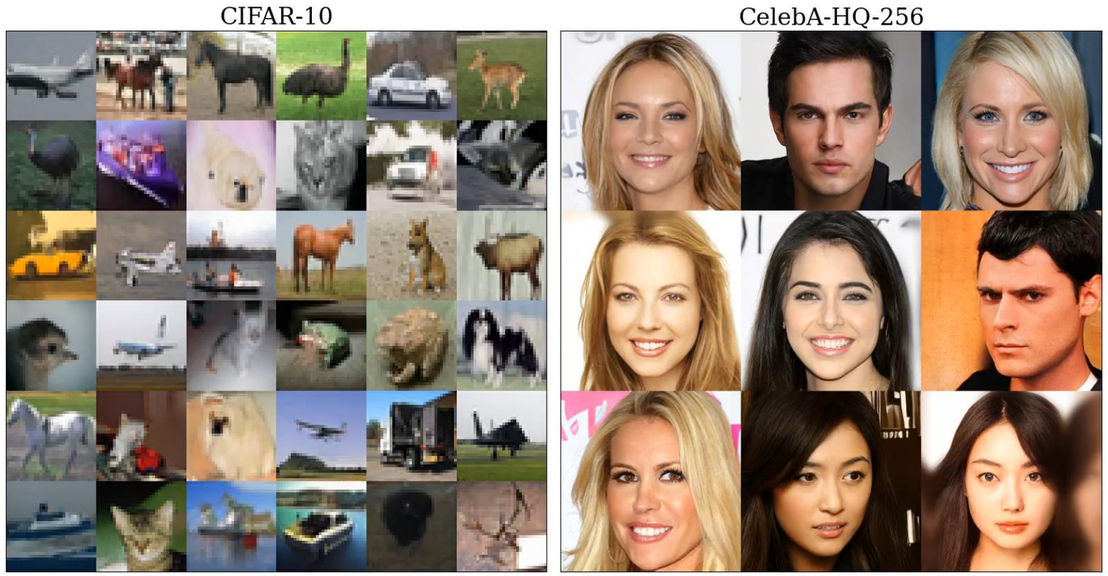
(image from: Hoogeboom et al., “Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions”, *NeurIPS*, 2022)

Introduction..

Open Question @ CVPR2022 Diffusion tutorials

Diffusion models can be considered as latent variable models, but their latent space lacks semantics

- How can we do **latent-space semantic manipulations** in diffusion models



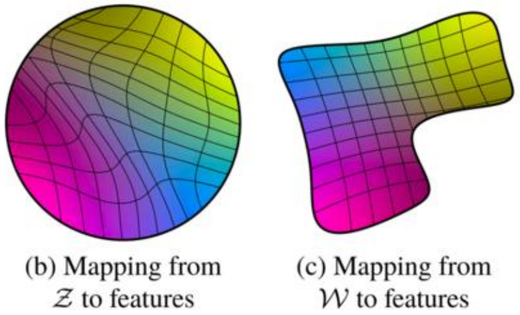
randomly traversing in the latent space of LSGM

Open Question @ CVPR2022 Diffusion tutorials

Diffusion models can be considered as latent variable models, but their latent space lacks semantics

- How can we do **latent-space semantic manipulations** in diffusion models

Interpolation in latent space of DDIM

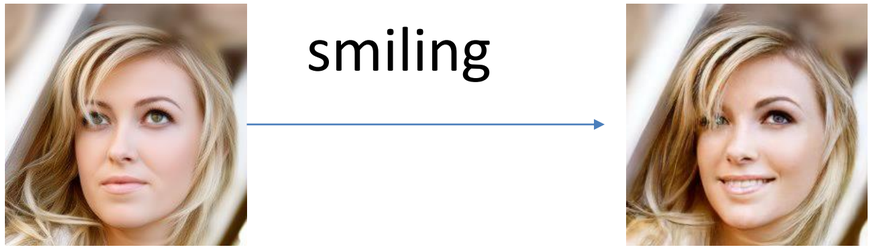


(b) Mapping from $\mathcal{Z}$ to features

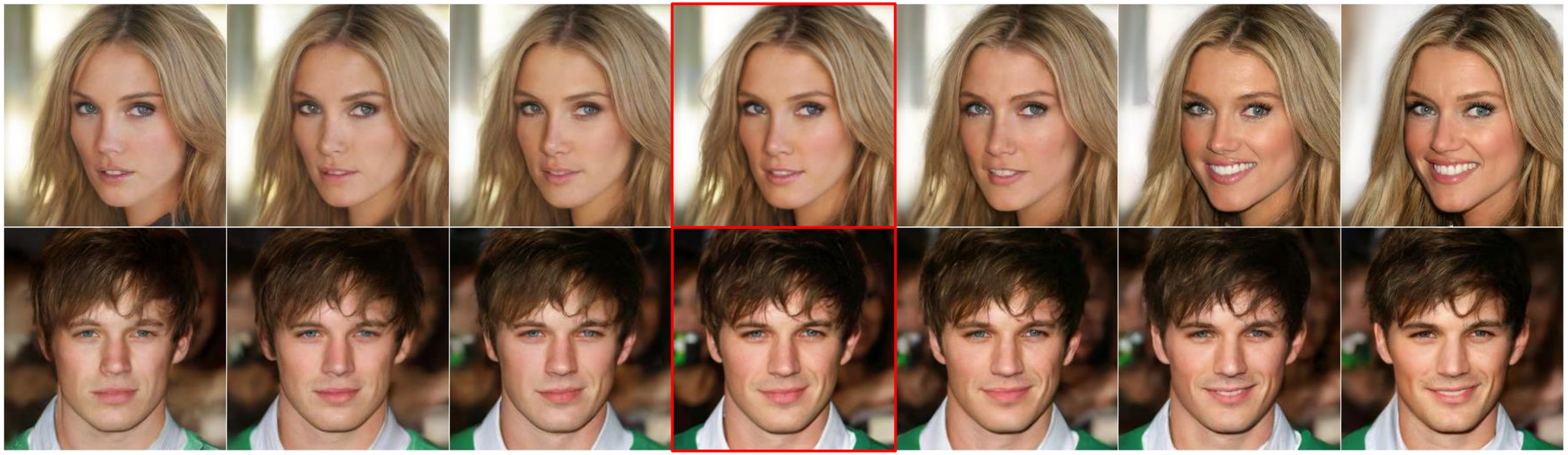
(c) Mapping from $\mathcal{W}$ to features

Our goal

- Finding **semantic latent-space** of Diffusion models
- **Editing real image** by semantic latent-space manipulation



1) Interpolation & Extrapolation



real image

- smiling (untrained)

+ smiling
(trained)

Out of domain.

Modigliani

Frida

VP-SDE (DDPM++)

iDDPM

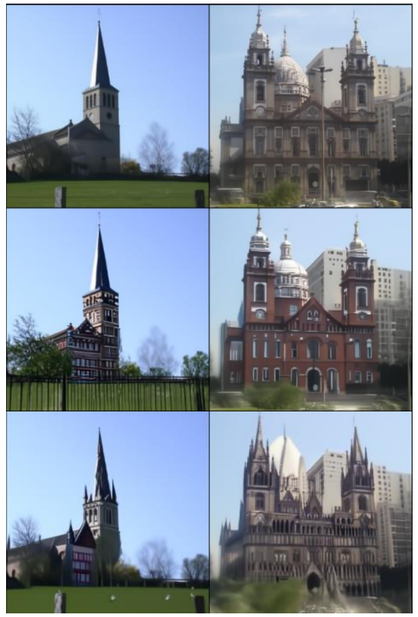
Guided Diffusion

LSUN-church

original

Red brick

Gothic



original

Palace

Hotel

LSUN-bedroom

original

sleepy

happy

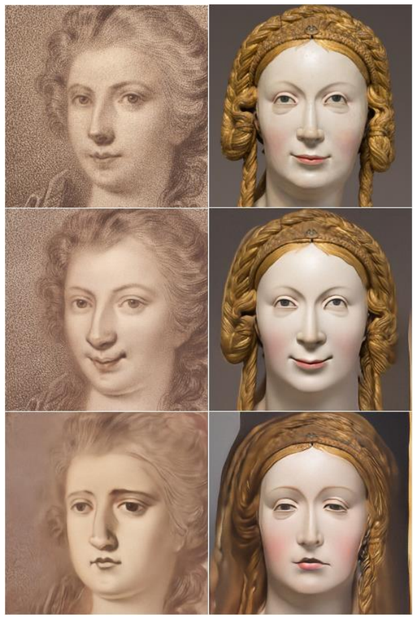
AFHQ-v2

original

smiling

sad

METFACE



mixing

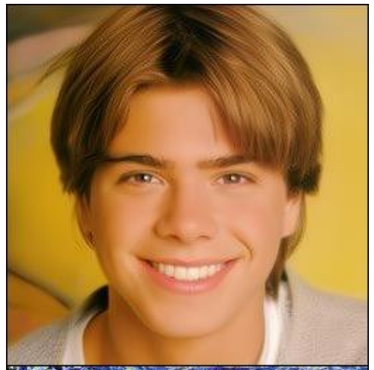
content

style

mixing

mixing

content



style

mixing

Thank you