

Diffusion Models

Lecture 19
Subir Varma

Deep Generative Learning: Learning to Generate Data



Samples from a Data Distribution

Train

A thick green arrow pointing from the cloud of cat images towards a head silhouette on the right.



Neural Network



Sample

A thick green arrow pointing from the head silhouette on the left towards a generated cat image on the right.

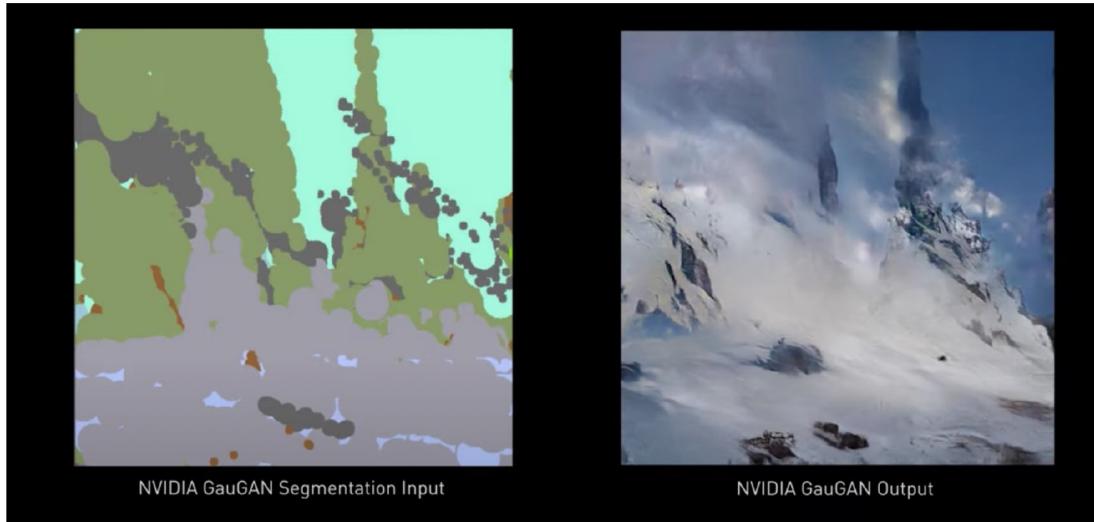


Applications

Content Generation



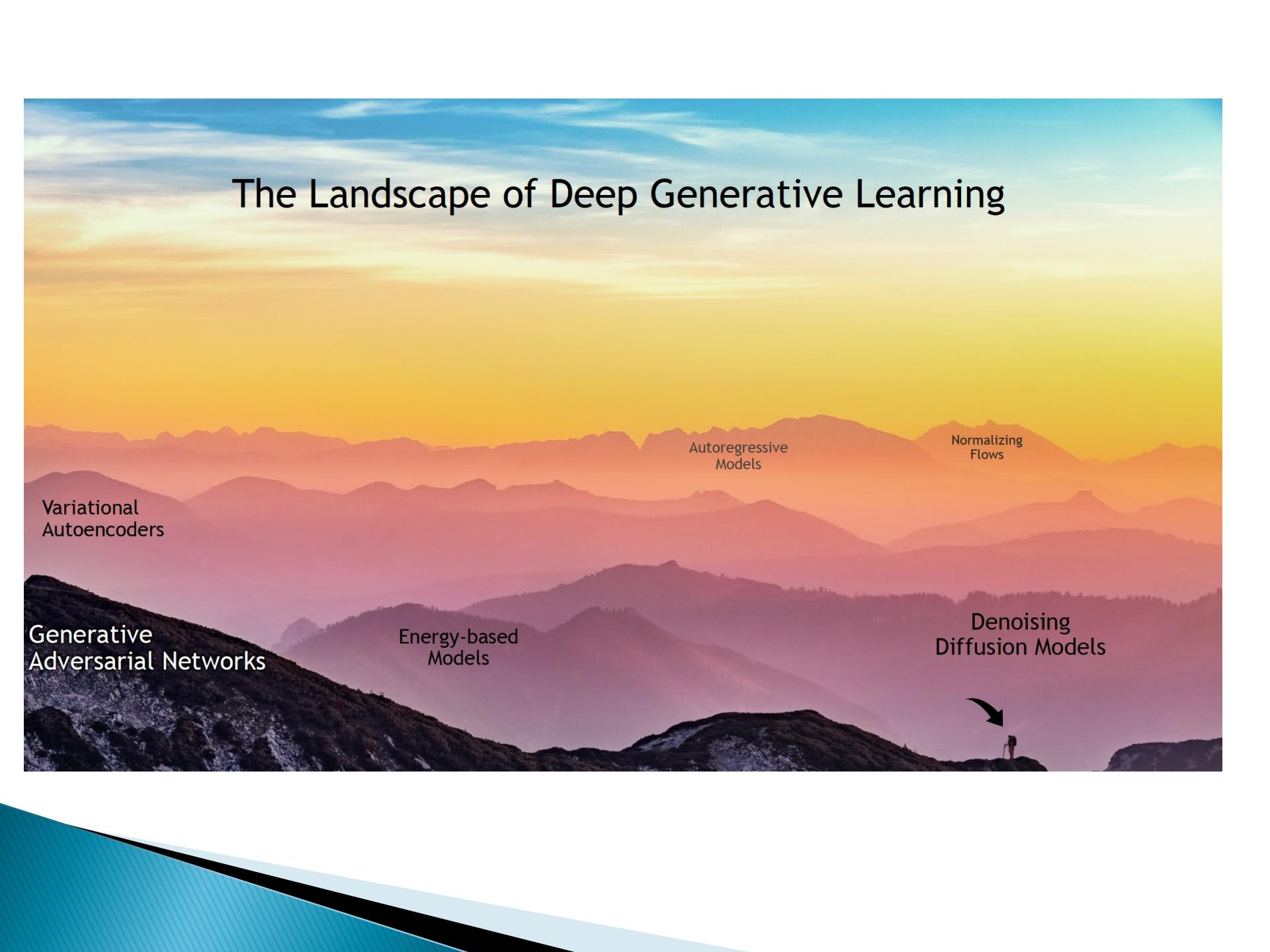
Artistic Tools



NVIDIA GauGAN Segmentation Input

NVIDIA GauGAN Output

The Landscape of Deep Generative Learning

A scenic sunset over a range of mountains. The sky transitions from blue at the top to orange and yellow near the horizon. In the foreground, there are dark silhouettes of mountain peaks. Overlaid on the image are several text labels representing different deep generative learning models, each associated with a specific mountain peak or cluster of peaks.

A scenic sunset over a range of mountains with text labels for various deep generative learning models.

Variational
Autoencoders

Generative
Adversarial Networks

Energy-based
Models

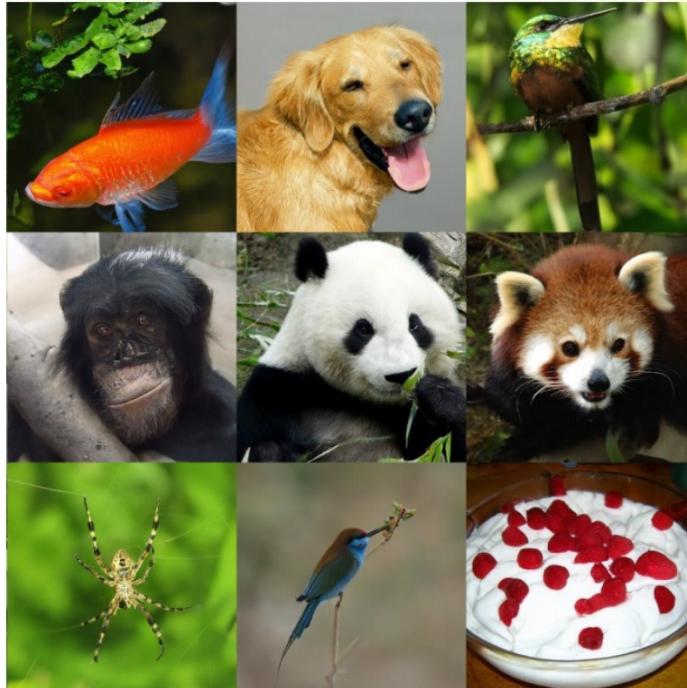
Autoregressive
Models

Normalizing
Flows

Denoising
Diffusion Models

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](#)

Text to Image Generation

DALL·E 2

"a teddy bear on a skateboard in times square"



["Hierarchical Text-Conditional Image Generation with CLIP Latents"](#)
[Ramesh et al., 2022](#)

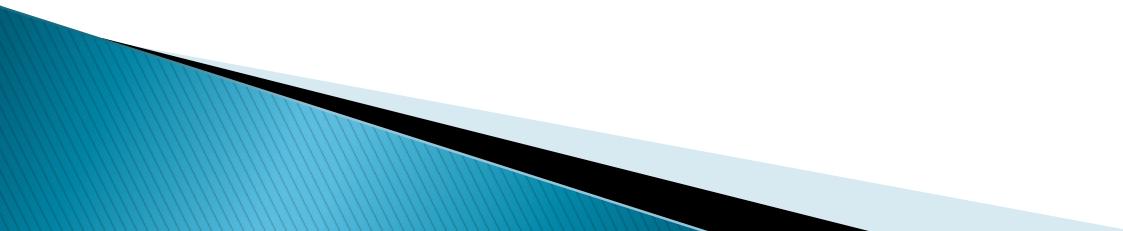
Imagen

A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk.



["Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", Saharia et al., 2022](#)

Denoising Diffusion Probabilistic Models

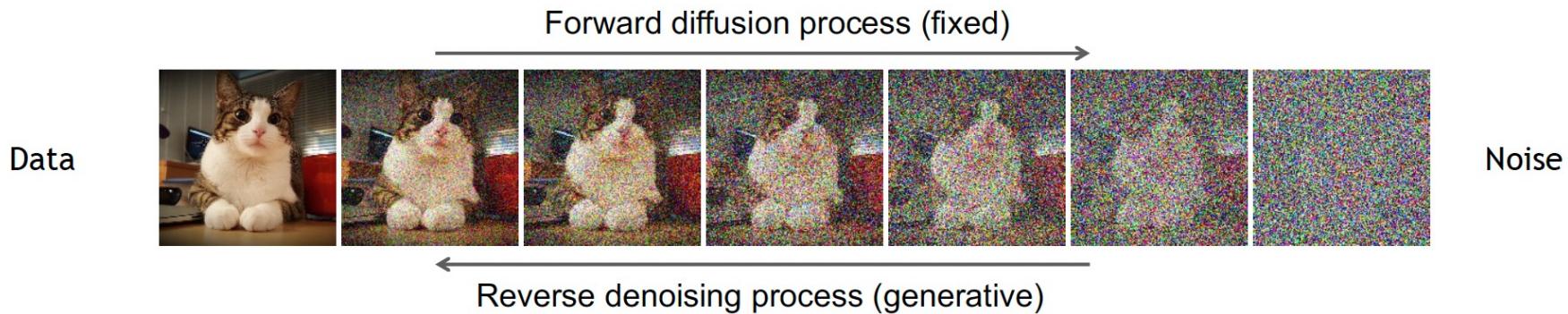


Denoising Diffusion Models

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



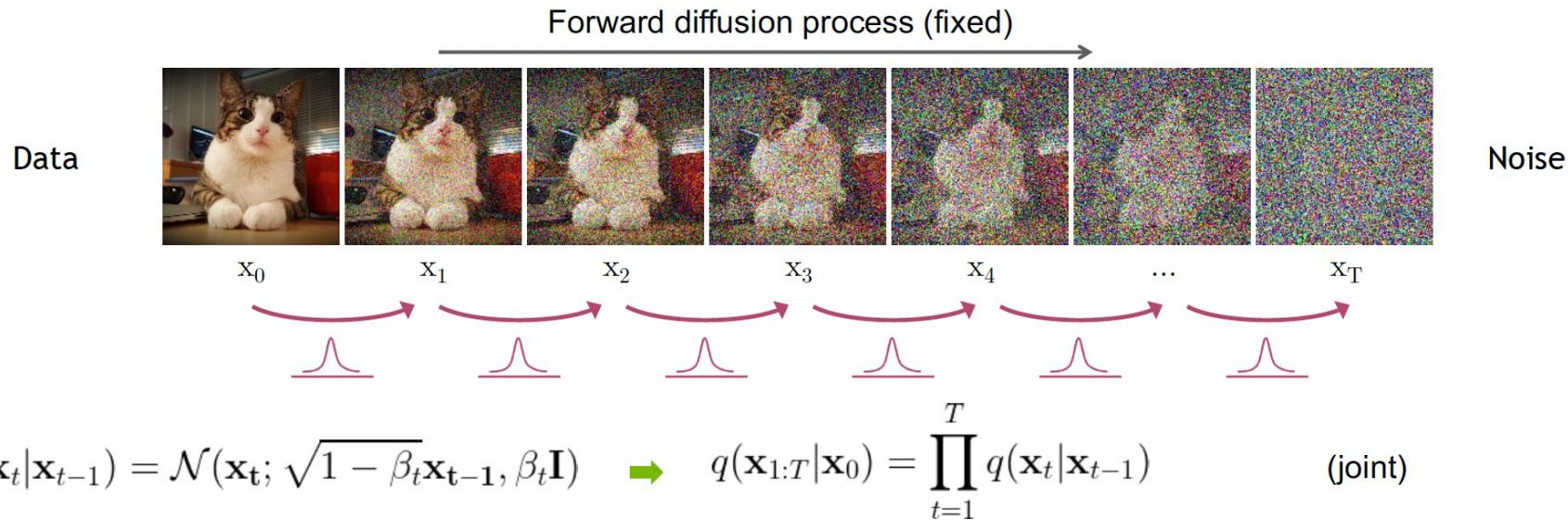
[Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015](#)

[Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020](#)

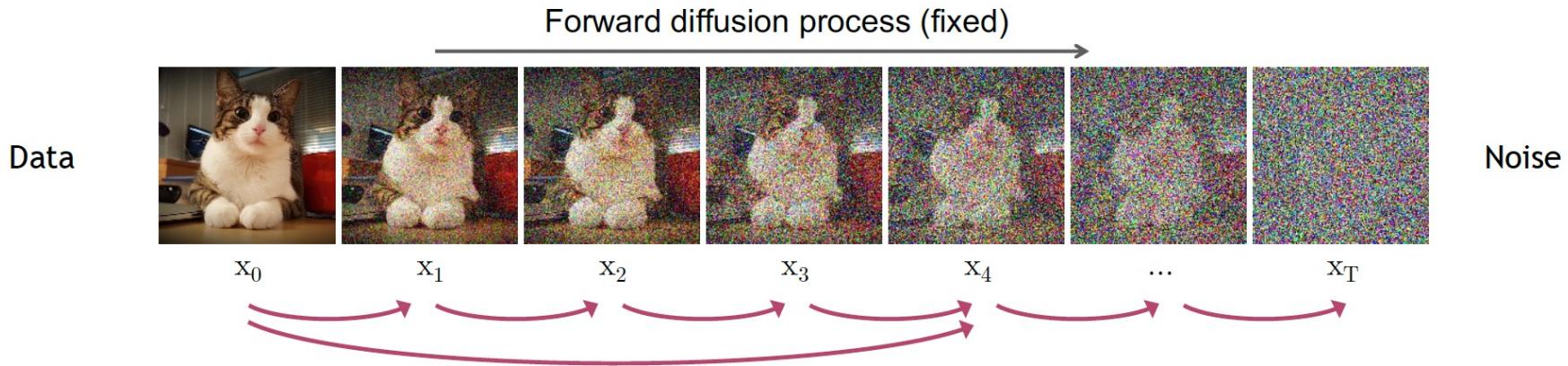
[Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021](#)

Forward Diffusion Process

The formal definition of the forward process in T steps:



Diffusion Kernel



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

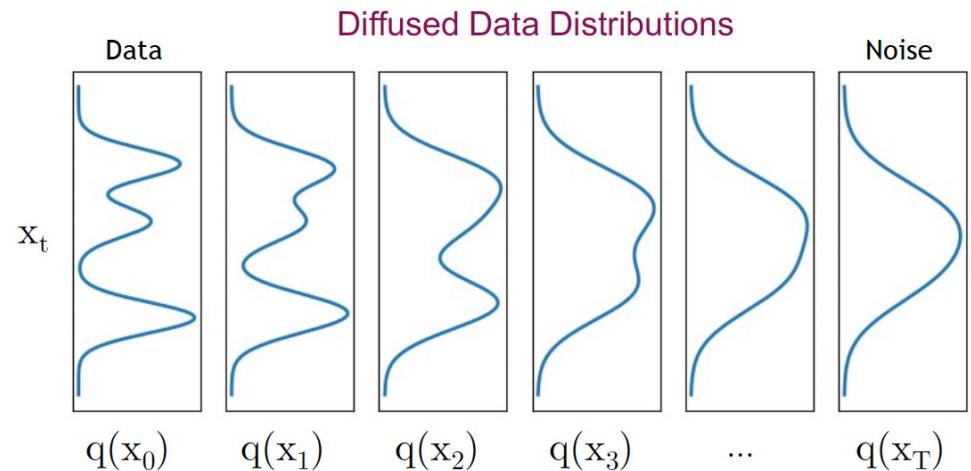
β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \rightarrow 0$ and $q(\mathbf{x}_T | \mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

What Happens to a Distribution in the Forward Diffusion?

So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$?

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

The diffusion kernel is Gaussian convolution.



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 $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

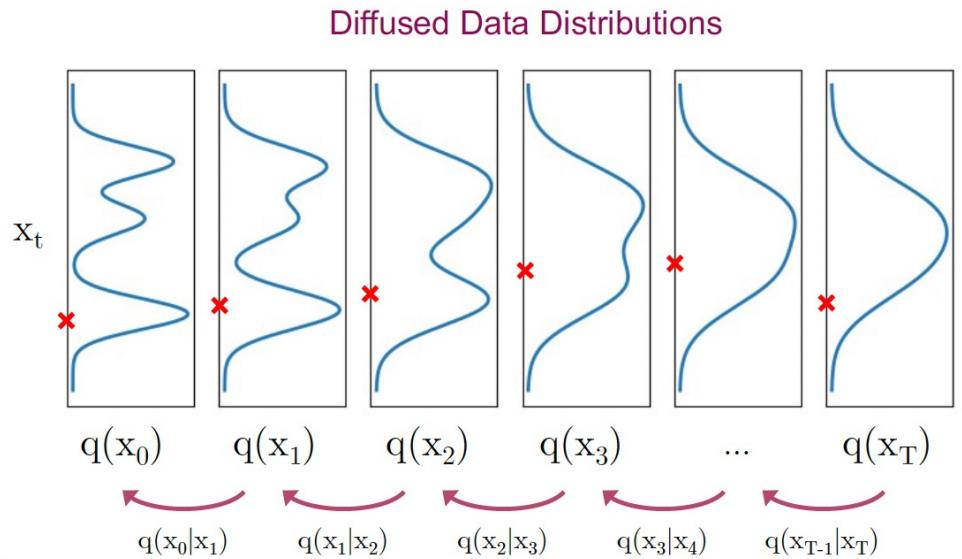
Generation:

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

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1} | \mathbf{x}_t)}_{\text{True Denoising Dist.}}$

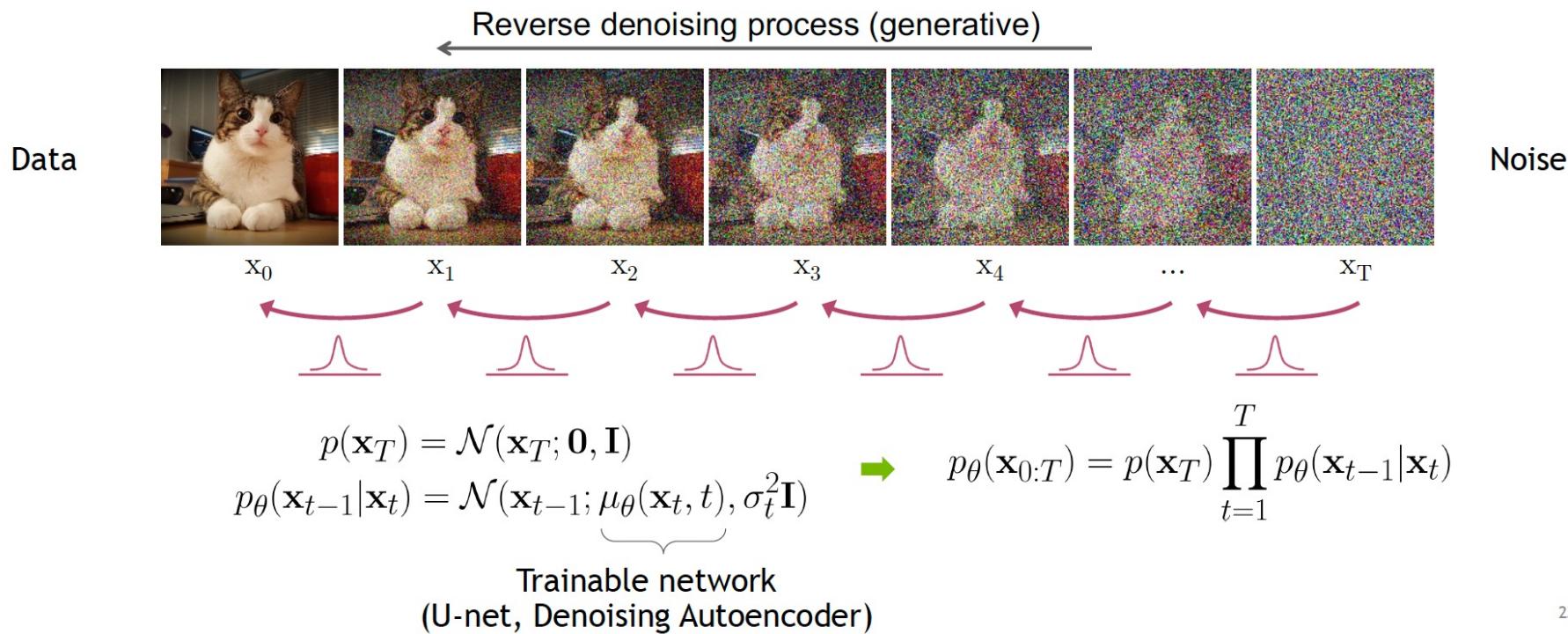
In general, $q(\mathbf{x}_{t-1} | \mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t | \mathbf{x}_{t-1})$ is intractable.

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:



Learning Denoising Model

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

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

[Sohl-Dickstein et al. ICML 2015](#) and [Ho et al. NeurIPS 2020](#) show that:

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

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

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

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

Parametrizing 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:

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

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 Weighing

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

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:

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

DDPM Algorithm

Algorithm 1 Training

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

6: until converged
```

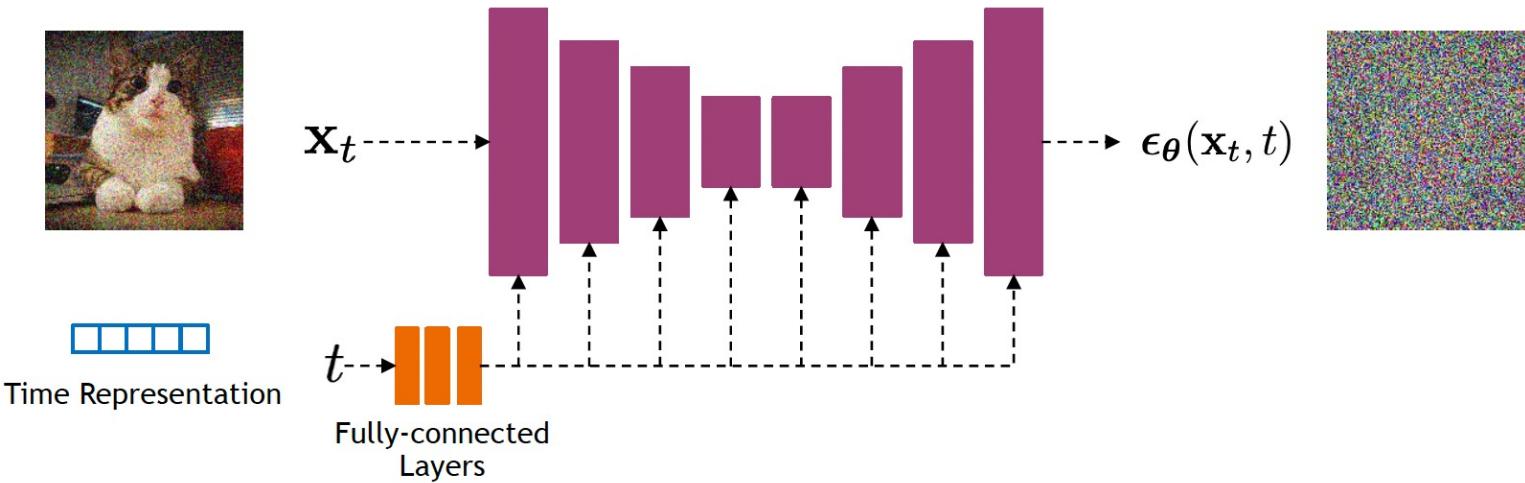
Algorithm 2 Sampling

```
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:   
$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$$

5: end for
6: return  $\mathbf{x}_0$ 
```

Implementation Considerations

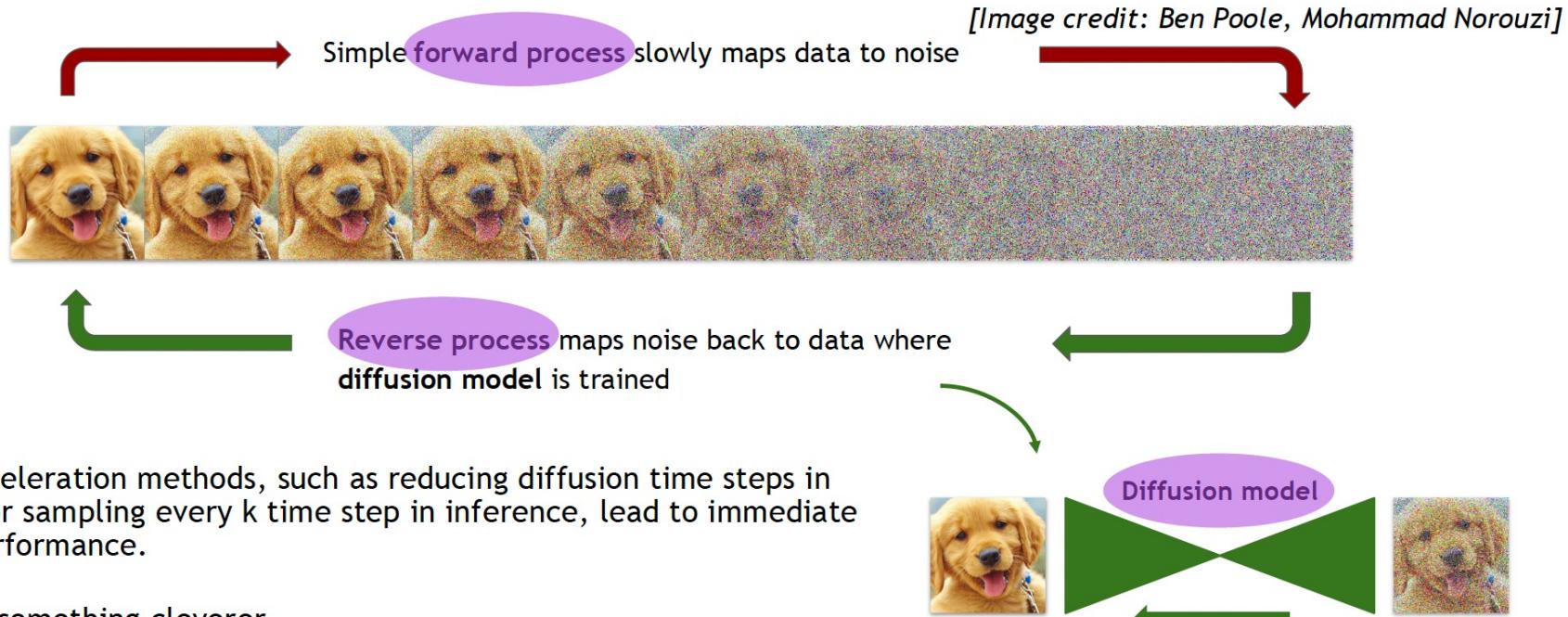
Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



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](#))

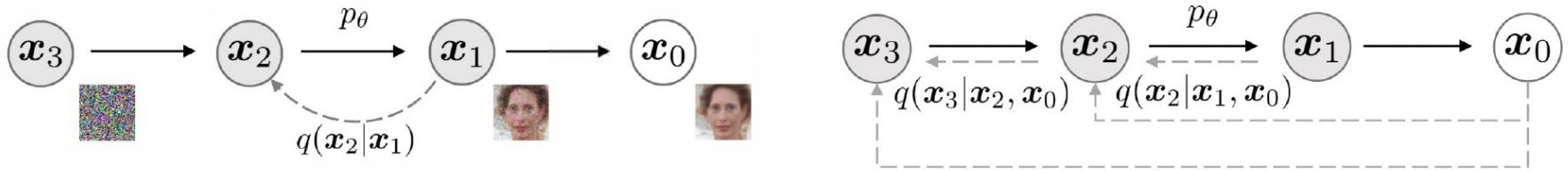
How to Accelerate Diffusion Models



- Naïve acceleration methods, such as reducing diffusion time steps in training or sampling every k time step in inference, lead to immediate worse performance.
- We need something cleverer.
- Given a limited number of functional calls, usually much less than 1000s, how to improve performance?

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.

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

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

Denoising Diffusion Implicit Models (DDIM)

How to define the non-Markovian forward process?

Recall that the KL divergence in the variational upper bound can be written as:

$$\begin{aligned} L_{t-1} = D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) &= \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] + C \\ &= \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\lambda_t \|\underbrace{\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \epsilon, t)}_{\mathbf{x}_t}\|^2 \right] + C \end{aligned}$$

$$\begin{aligned} \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon \right) \\ \mu_\theta(\mathbf{x}_t, t) &= \frac{1}{\sqrt{1-\beta_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \end{aligned}$$

If we assume loss weighting λ_t can be arbitrary values, the above formulation holds as long as

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1-\bar{\alpha}_t) \mathbf{I}) \quad (\text{make sure } \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)} \epsilon)$$

Forward process: $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_t^2 \mathbf{I}), \quad \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}$

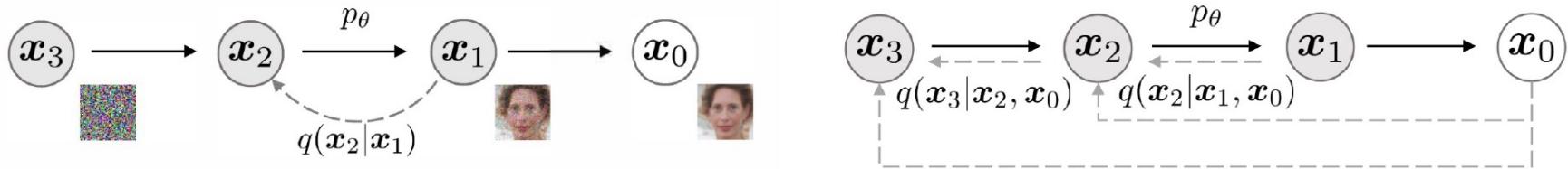
Reverse process: $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \tilde{\sigma}_t^2 \mathbf{I}), \quad \mu_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b\epsilon_\theta(\mathbf{x}_t, t) = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \hat{\mathbf{x}}_0}{\sqrt{1-\bar{\alpha}_t}}$

No need to specify $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ to be a Markovian process!

(assume $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \hat{\mathbf{x}}_0 + \sqrt{(1-\bar{\alpha}_t)} \epsilon_\theta(\mathbf{x}_t, t)$)

Denoising Diffusion Implicit Models (DDIM)

Non-Markovian diffusion process



For the forward process $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\sigma}_t^2 \mathbf{I})$, $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = a\mathbf{x}_t + b\epsilon = a\mathbf{x}_t + b\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}}$, need to choose a, b such that $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$

Define a family of forward processes that meets the above requirement:

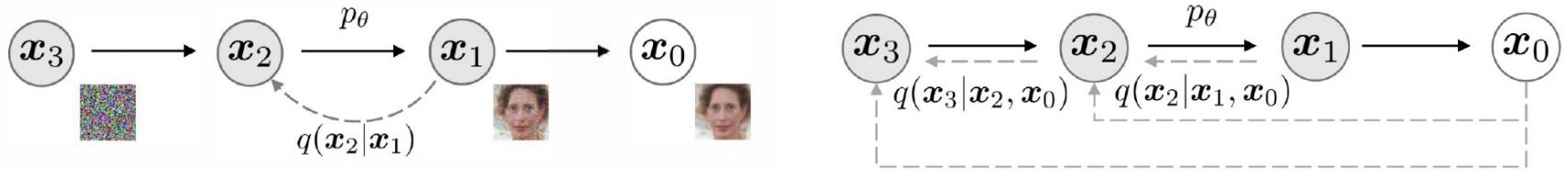
$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \tilde{\sigma}_t^2} \cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}}, \tilde{\sigma}_t^2 \mathbf{I}\right)$$

The corresponding reverse process is

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

DDIM Sampler

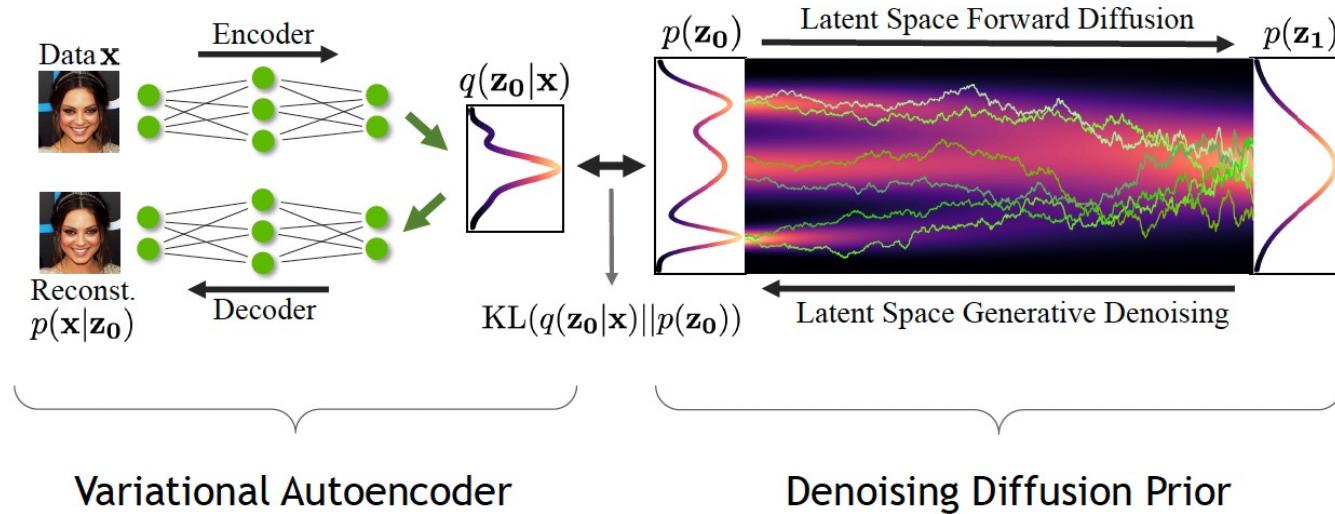
Deterministic generative process



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

- DDIM sampler - $\tilde{\sigma}_t^2 = 0, \forall t$
 - a deterministic generative process, with randomness from only $t=T$.

Latent Space Diffusion Models



Variational Autoencoder

Denoising Diffusion Prior

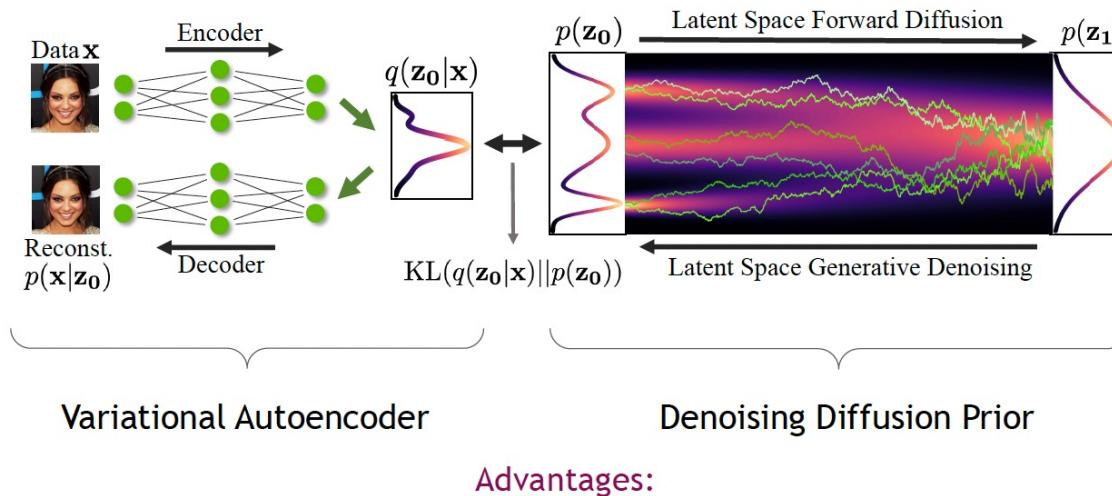
Main Idea

Encoder maps the input data to an embedding space

Denoising diffusion models are applied in the latent space



Latent Space Diffusion Models



- (1) The distribution of latent embeddings close to Normal distribution → *Simpler denoising, Faster Synthesis!*
- (2) Augmented latent space → *More expressivity!*
- (3) Tailored Autoencoders → *More expressivity, Application to any data type (graphs, text, 3D data, etc.) !*

Conditional Diffusion Models

Text-to-image generation

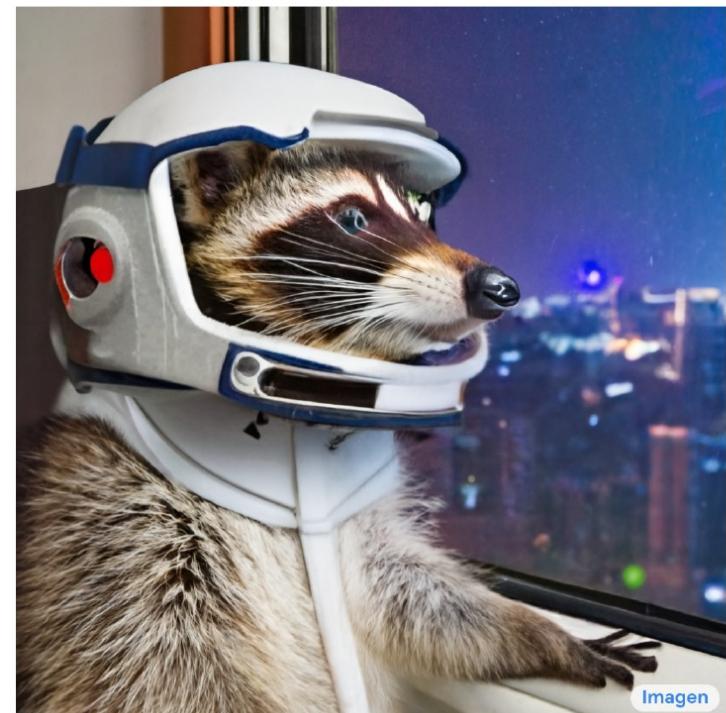
DALL·E 2

"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."



Conditional Diffusion Models

Include condition as input to reverse process

Reverse process: $p_{\theta}(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}), \quad p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t, \mathbf{c}), \boldsymbol{\Sigma}_{\theta}(\mathbf{x}_t, t, \mathbf{c}))$

Variational upper bound: $L_{\theta}(\mathbf{x}_0|\mathbf{c}) = \mathbb{E}_q \left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})) - \log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1, \mathbf{c}) \right].$

Incorporate conditions into U-Net

- Scalar conditioning: encode scalar as a vector embedding, simple spatial addition or adaptive group normalization layers.
- Image conditioning: channel-wise concatenation of the conditional image.
- Text conditioning: single vector embedding - spatial addition or adaptive group norm / a seq of vector embeddings - cross-attention.

Further Reading

- ▶ Das and Varma: Chapter Image Generation Using Diffusion Models, Chapter GenerativeModels