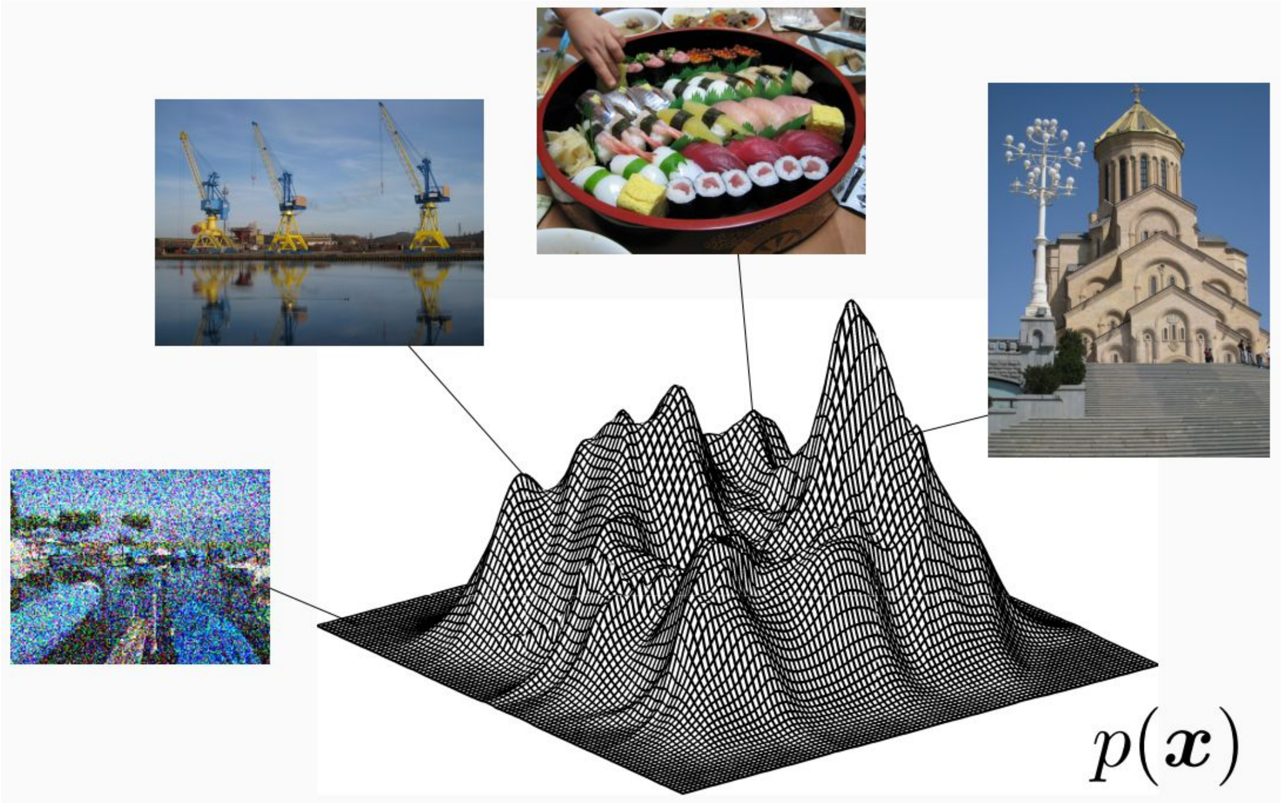
Generative Modeling

Denoising Diffusion Models: Theory, Implementation and Control

Baptiste Engel, baptiste.engel@cea.fr
November 2023

“A view of a jungly-mountain landscape, with a small adventurer in the middle, facing the enormous forest. 4k, hd quality, in the style of a 19th century painter” - SDXL

Generative Modeling: what and why?



How to learn this distribution so that we can sample new elements from it?

From *UCSD ECE285 Machine learning for image processing*, C. Deledalle

Generative Modeling: what and why?

Text Generation (eg LLM): ChatGPT, Llama

Text-to-Image

Image-to-Image

Image-to-Video

Video-to-video

Image-to-3D

Text-to-video

Text-to-3D

Text-to-audio

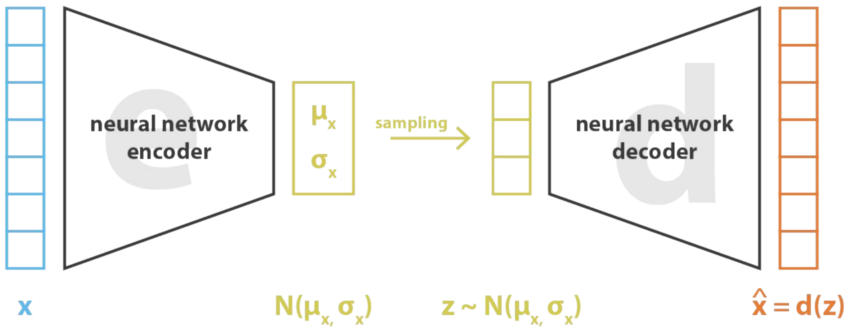
Audio-to-image

Audio-to-video

And many more…

Previously…

Variational Auto-Encoders :



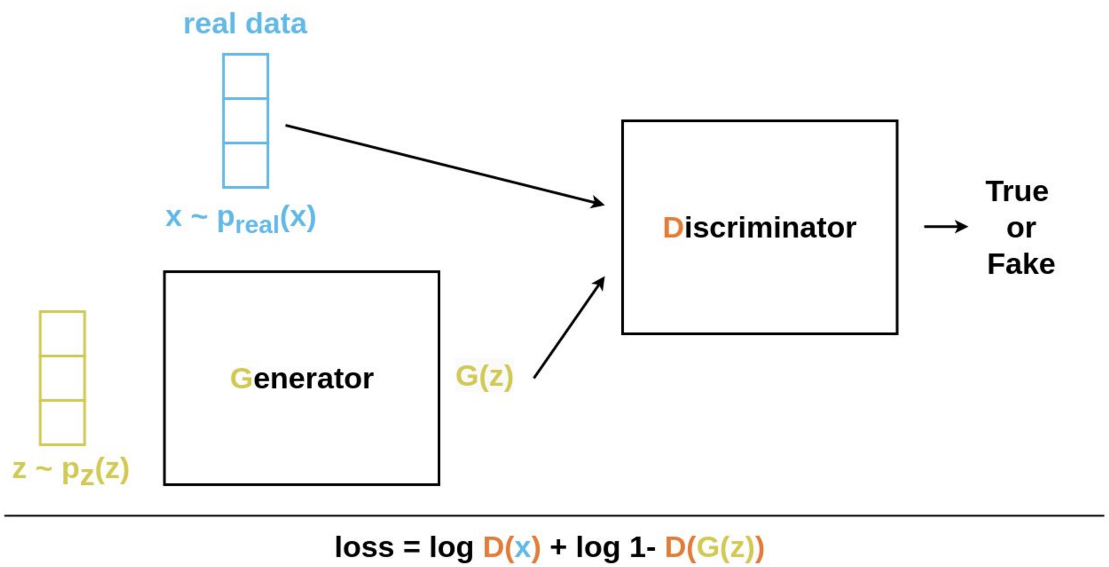
$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

Understanding Variational Autoencoders (VAEs), Joseph Rocca
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

- Encoding/Decoding process (AE)
- Fit a distribution over the latent space
- Blurry Images : “mean” term in loss

Previously…

Generative Adversarial Networks :



Original figure.

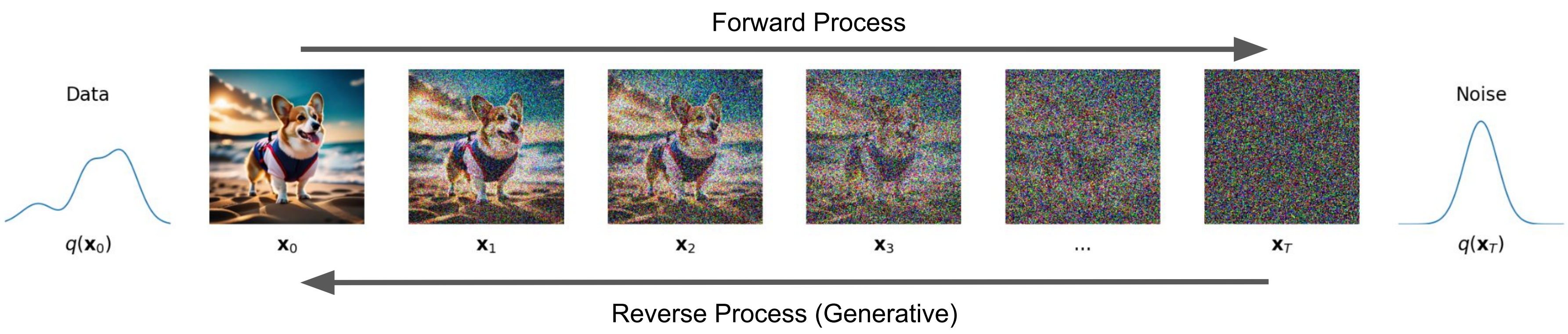
- Minimax 2-players game
- Generator tries to fool the discriminator
- Realistic images
- Difficult control (No encoder), difficult to train, mode collapse

Denoising Diffusion Models

“A picture of a corgi wearing a bathsuit. HD, high quality, 4K.” - SDXL

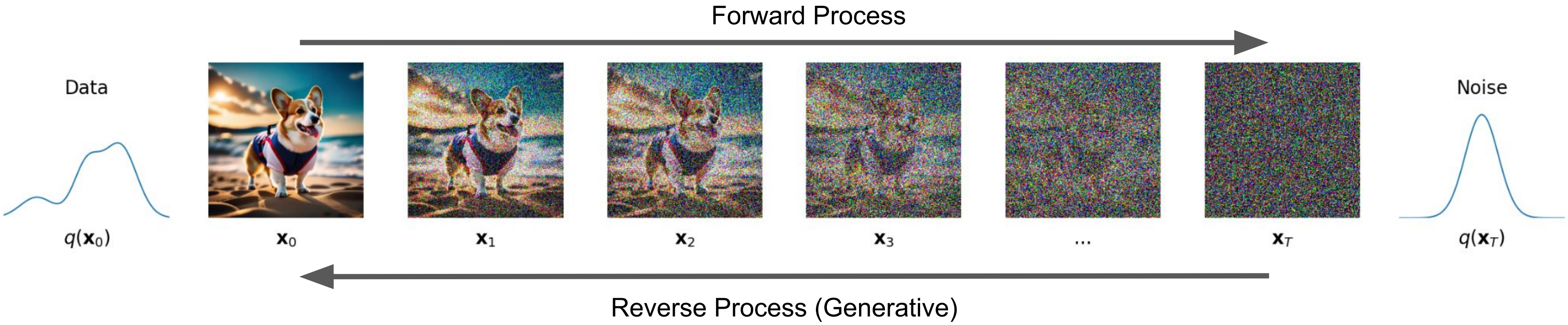
- How do they work?
- How to train them?
- How to use them?
- How to control them?

Try yourself! https://huggingface.co/spaces/google/sdxl

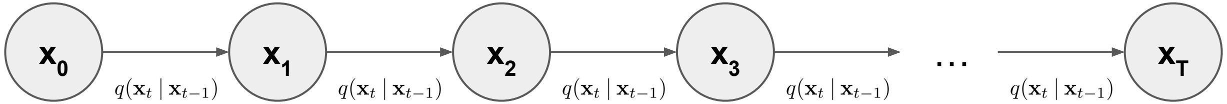


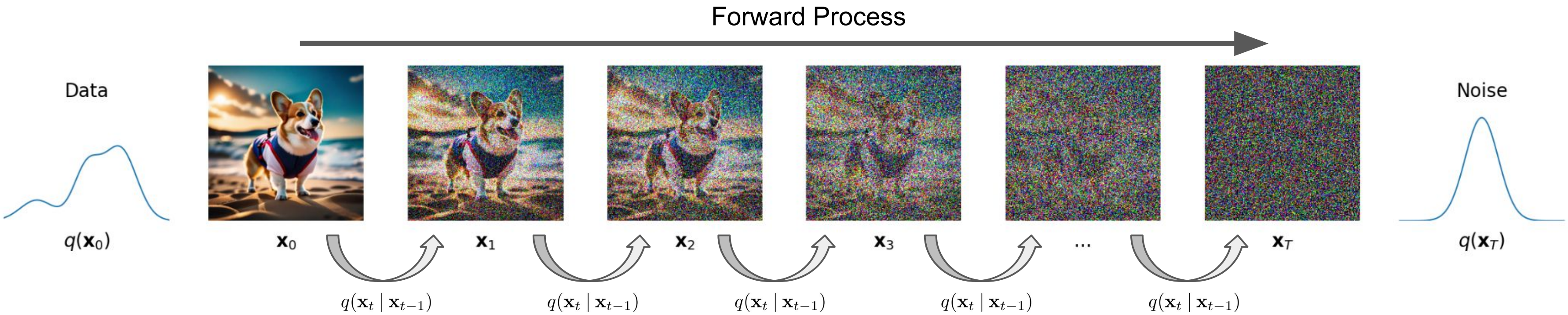
Two processes:

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



As a Markov chain:





Markov Kernel:

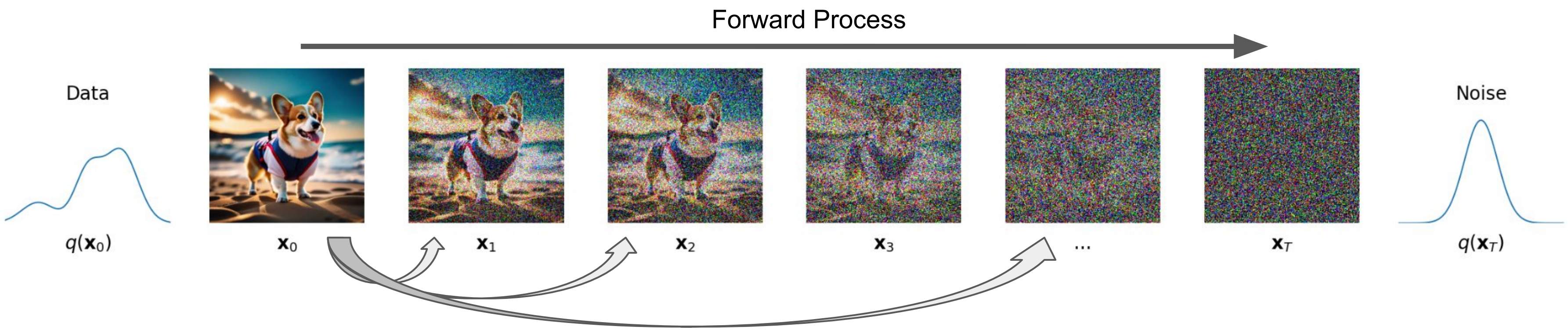
$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

(Or any Markov Diffusion Kernel actually [1, 2])

Forward trajectory:

$$q(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

To go to step t from t-1 in the Markov chain, sample from the kernel distribution.

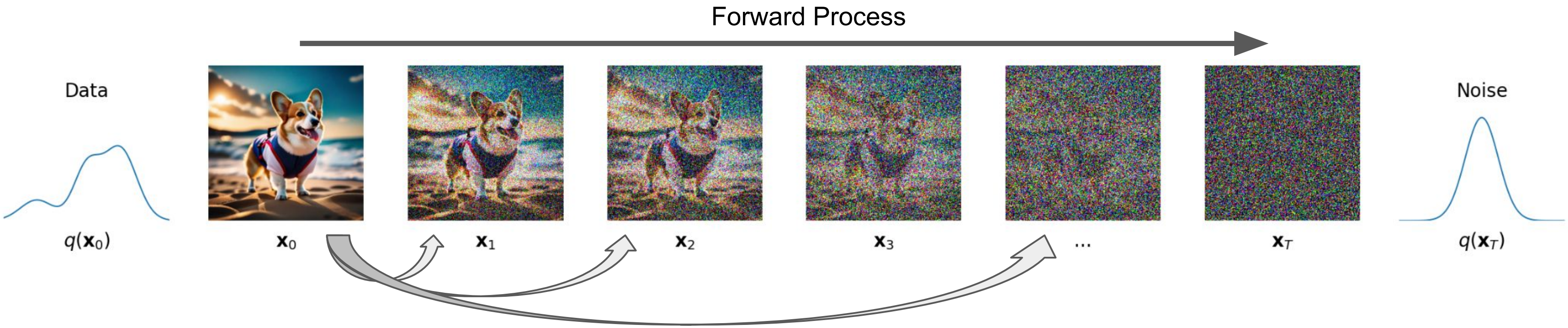


$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

$$\begin{cases} \alpha_t := 1 - \beta_t \\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s \end{cases} \Longrightarrow \underbrace{q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})}_{\text{Diffusion Kernel}}$$

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

→ Reach any node of the chain in one operation.

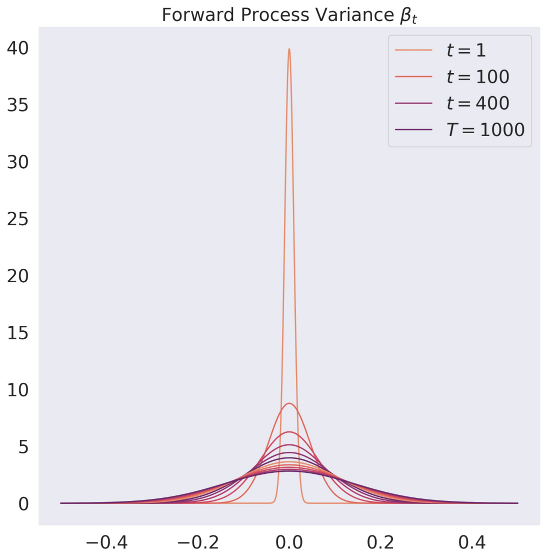


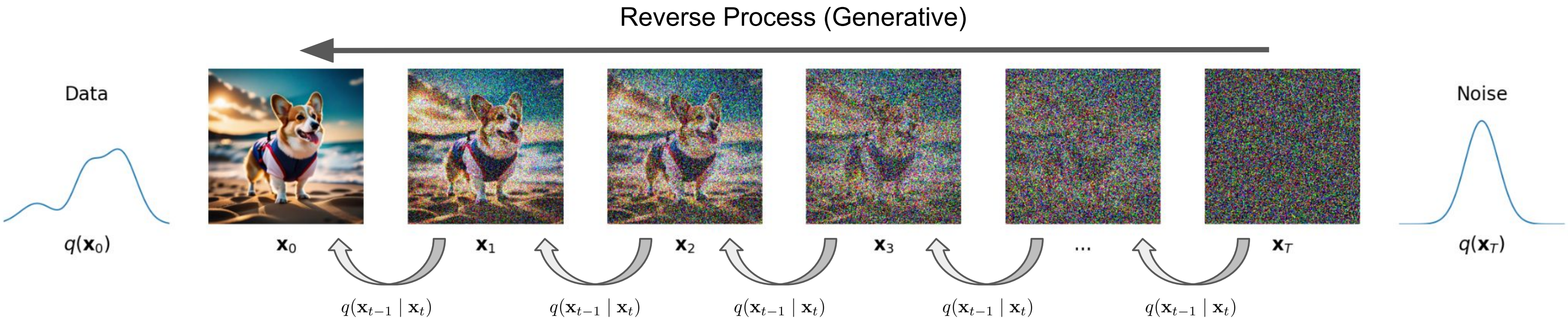
$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

- β_t values schedule = noise scheduler
- Such that:

$\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T \mid \mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

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

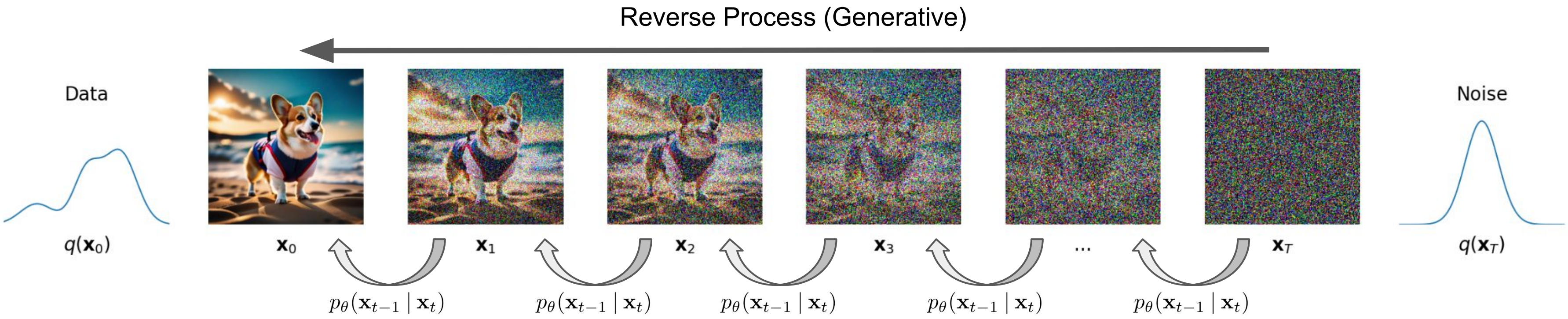




- Parameters are such that $q(\mathbf{x}_T)$ is a standard Gaussian
- If βs are small enough, the reverse transitions **are also Gaussian**
- $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ is intractable

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \propto q(\mathbf{x}_{t-1}) q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

Let’s use a model to approximate the reverse process!



Our parametric reverse model:

- Also a Markov chain
- Starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$ because $q(\mathbf{x}_T) \approx \mathcal{N}(0, \mathbf{I})$
- Learned transition

Reverse transitions are gaussian, so ours too:

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

Reverse trajectory:

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

- Find a model that fit the true data distribution q($\mathbf{x}_0$).
- How to find the right parameters? Maximizing the log-likelihood (Minimizing the NLL).

$$\underbrace{-\log p_\theta(\mathbf{x}_0)}_{\text{Our model parameters}} = \underbrace{-\log \int p_\theta(\mathbf{x}_{0:T})d\mathbf{x}_{1:T}}_{\text{Marginalisation of the trajectory over the latents}}$$

- Find a model that fit the true data distribution q($\mathbf{x}_0$).
- How to find the right parameters? Maximizing the log-likelihood (Minimizing the NLL).

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &= -\log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\
&= -\log \int p_\theta(\mathbf{x}_{0:T}) \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} d\mathbf{x}_{1:T}
\end{aligned}
$$

- Find a model that fit the true data distribution q($\mathbf{x}_0$).
- How to find the right parameters? Maximizing the log-likelihood (Minimizing the NLL).

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &= -\log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\
&= -\log \int p_\theta(\mathbf{x}_{0:T}) \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} d\mathbf{x}_{1:T} \\
&= -\log \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \right]
\end{aligned}
$$

- Find a model that fit the true data distribution q($\mathbf{x}_0$).
- How to find the right parameters? Maximizing the log-likelihood (Minimizing the NLL).

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &= -\log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\
&= -\log \int p_\theta(\mathbf{x}_{0:T}) \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} d\mathbf{x}_{1:T} \\
&= -\log \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \Big[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \Big] \\
&\leq \quad \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \Big[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)} \Big] := L_{\text{VLB}}
\end{aligned}$$

Jensen's Inequality

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

Kullback-Leibler Divergence: distance between two distributions (/!\ not symmetric)

After some derivations…

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

No learnable parameters,
we can remove it.
(also = 0)

= 0 when both distribution
are equal

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

- Minimizing the NLL = Minimizing the KLD terms
- $D_{KL}(P||Q) \geq 0$. Zero when P=Q
- When conditioned on x_t and x_0, the reverse process is tractable:

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

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

- We thus want $p_\theta(x_{t-1}|x_t)$ to be as close a possible to $q(x_{t-1}|x_t,x_0)$:
  - Normal law
  - Approximate $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)$ in our approximated reverse process

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

- We have:

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

- KL divergence of 2 gaussian: $L_{t-1}=\mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\right]+C$

- $\mathbf{x}_t=\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ $\quad\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0):=\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0+\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$

Thus,

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

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

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

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right)$$

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

Thus,

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

Noise estimator (eg, a neural net!)

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

Noise we added

x_t obtained with reparameterization

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

- Discrete decoder
- Clip values in {0,1,...255} for images

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

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

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

- We now can derive L_{t-1} and L_0 wrt θ
- But easier AND better sample quality with:

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

Diffusion models = noise estimator

Sampling:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}) \quad \text{with} \quad \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

Recap: Denoising Diffusion Models

1) Training algo

Going from the data to the easy-to-sample distribution

Algorithm 1 Training

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

2) Generation algo

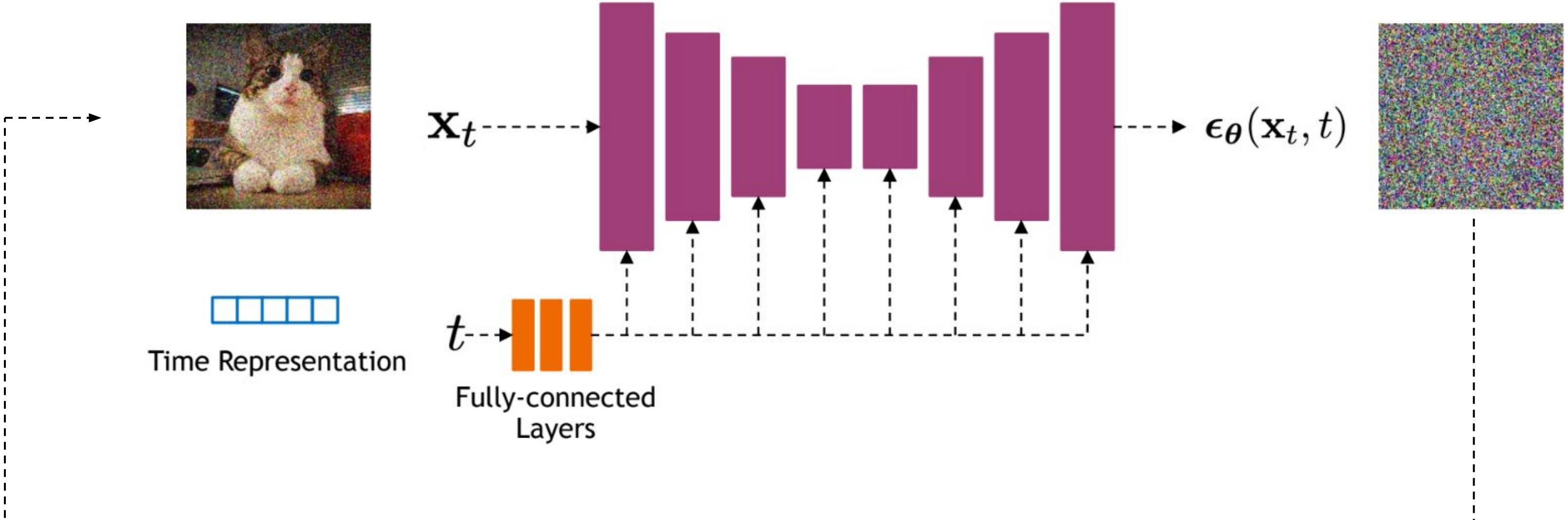
Going from the easy-to-sample to the data distribution

Algorithm 2 Sampling

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

[3] Denoising Diffusion Probabilistic Models, Ho et al.

The actual denoising model!

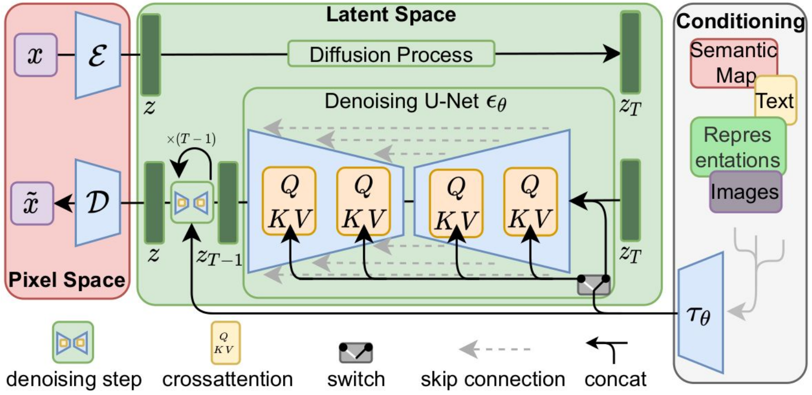


$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$$

From *Denoising Diffusion-based Generative Modeling: Foundations and Applications Tutorial*, CVPR 2022

Implementation Consideration

- Diffusion in a smaller latent space = dimension reduction
- Neural Networks are good function estimators
- In practice:
  - First proposition [1] (2015) : Radial Basis Networks
  - UNet with skip connections (DDIM [5], Stable Diffusion [6]), VAE to encode the latent
  - Transformers [DiT]



[6] High-Resolution Image Synthesis with Latent Diffusion Models

- One network for every timestep: t is a parameter of the network
- Very large datasets: billions of image = cost + carbon footprint (11 tons CO2eq for SD 1.5)

Example : “a photograph of an astronaut riding a horse”

Stable Diffusion 1.5 (Oct. 2022)

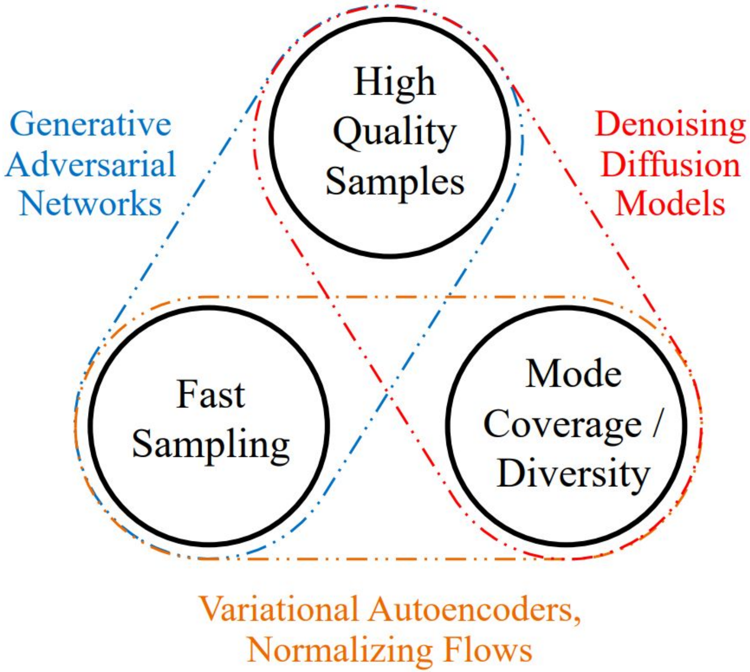
Stable Diffusion XL (Jul. 2023)

Example : “a photograph of an astronaut riding a horse”

DALL·E 2 (Apr. 2022)

DALL·E 3 (Aug. 2023)

Generative models trilemma



➡ How to sample faster?

[9] Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. Xiao et al., 2021

Fast sampling

Evaluation of the neural network is a slow process!

- Usually, T=1000: performing 1000 denoising steps means evaluating 1000 times our network.

Fast sampling

Denoising Diffusion Implicit Models [5]:

- *Non-markovian* inference process: shortcuts!

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \underbrace{\boldsymbol{x}_0}_{\text{We know where we're heading!}}) = \mathcal{N}\left(\underbrace{\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}}_{\text{(check the paper to understand this mean)}}, \underbrace{\sigma_t^2}_{\text{Add some stochasticity to the process}}\boldsymbol{I}\right)$$

Fast sampling

Denoising Diffusion Implicit Models [5]:

- *Non-markovian* inference process: shortcuts!

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\alpha_t}\boldsymbol{x}_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 \boldsymbol{I}\right)$$

(check the paper to understand this mean)

We don’t know x_0!

- *Denoised Observation*: $\hat{\mathbf{x}}_0^t = \mathbf{f}_\theta^t(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \sqrt{1-\alpha_t} \cdot \boldsymbol{\epsilon}_\theta^t(\mathbf{x}_t))$

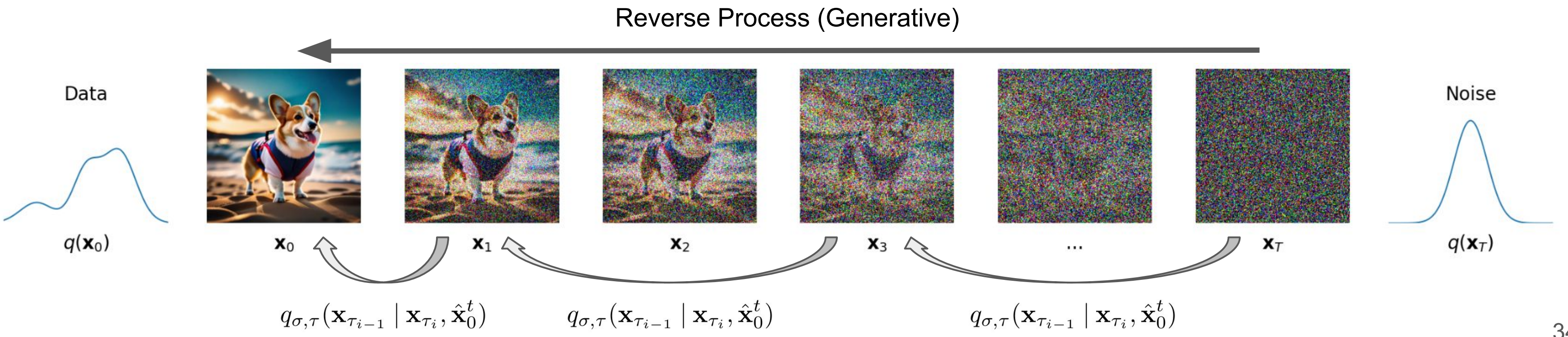
Thus, $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \begin{cases} \mathcal{N}(\hat{\mathbf{x}}_0^1, \sigma_1^2 \mathbf{I}) & \text{if t=1} \\ q_\sigma(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \hat{\mathbf{x}}_0^t) & \text{otherwise.} \end{cases}$

Fast sampling

Denoising Diffusion Implicit Models [5]:

- Only denoise for a subsample of the node

$$\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots \tau_K\} \subset \mathbf{T} = t_1, t_2, \ldots, t_T$$



Fast sampling

Denoising Diffusion Implicit Models (Song et al., 2021):

- The reverse process is no longer Markovian
- No retrain needed!
- Produces good results with 50 steps (instead of 1000): ~x20 speed-up!

Others:

- Rectified Flow [10], InstaFlow [11] ➡ one step diffusion!
- Variational Diffusion Models [12]

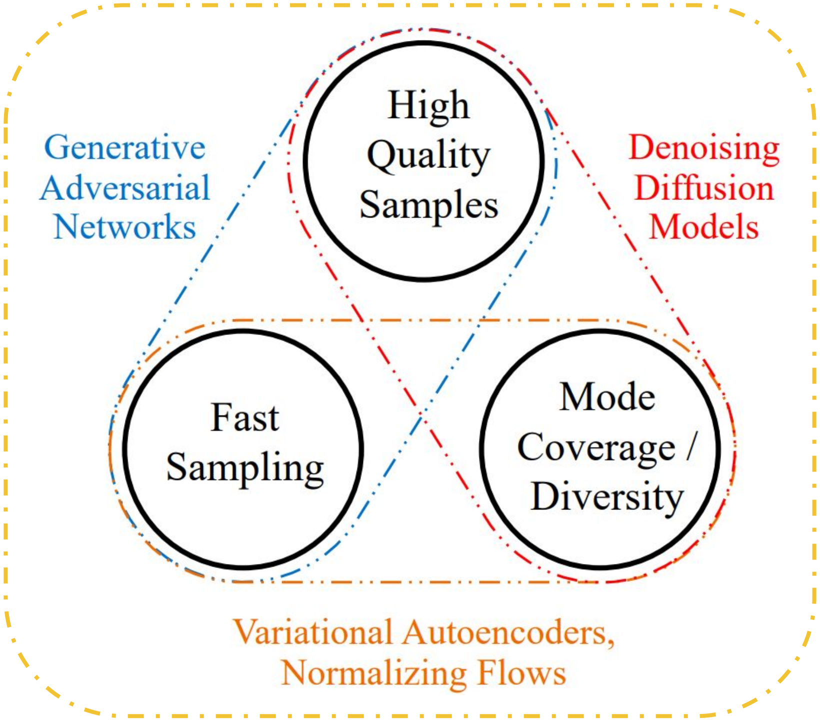
Fast sampling

SD 1.5: “A cat in a sweater, cartoon style, seems nice, eating popcorn”
Generated with different DDIM number of steps

5 steps, 1 sec | 25 steps, 7 sec | 50 steps, 14 sec | 999 steps, 5:05min

Generated on NVidia Titan X (16Gb VRAM) with SD 1.5

Fast sampling



Denoising Diffusion Implicit Models

Controllable Generation - CFG

- “Controllable” generation = approximate $q(\mathbf{x}_0 \mid \mathbf{c})$
- Classifier Free Guidance (CFG) [13]: conditioning as input - $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c})$
- Randomly drop the condition when training: $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \varnothing)$
- Inference: merge both conditional and unconditional prediction:

$$\boxed{\tilde{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t, \mathbf{c}, t) = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \mathbf{c}, t) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, \varnothing, t)}$$

w is the *guidance-scale*.

[13] Classifier-Free Diffusion Guidance, Ho & Salimans, 2022

Controllable Generation - CFG

- In practice: fuse the conditioning vector with *self-attention*

arXiv > cs > arXiv:1706.03762

Computer Science > Computation and Language

[Submitted on 12 Jun 2017 (v1), last revised 2 Aug 2023 (this version, v7)]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

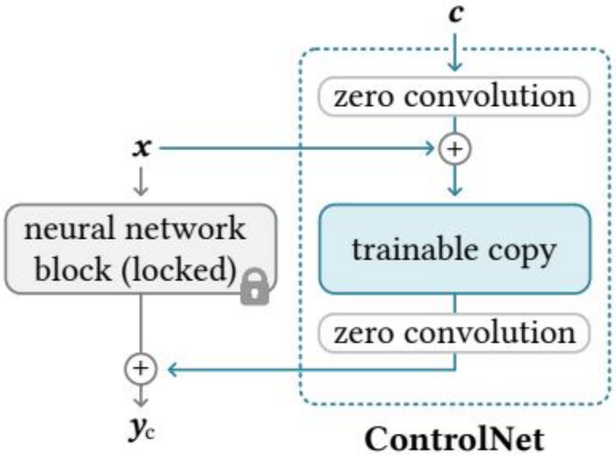
Controllable Generation - CFG

a) Without CFG b) With CFG, w=3.0

[13] Classifier-Free Diffusion Guidance, Ho & Salimans, 2022

Controllable Generation - ControlNet

- Finetune large models with few data without catastrophic forgetting
- Control your model with any modality



[14] Adding Conditional Control to Text-to-Image Diffusion Models

And still more to explore!

[15] Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets

References

[1] Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015
[2] Non Gaussian Denoising Diffusion Models, 2021
[3] Denoising Diffusion Probabilistic Models, Ho et al., 2020
[4] Denoising Diffusion-based Generative Modeling: Foundations and Applications Tutorial, CVPR 2022
[5] Denoising Diffusion Implicit Models, 2020
[6] High-Resolution Image Synthesis with Latent Diffusion Models, 2021: Stable Diffusion,
[7] Scalable Diffusion Models with Transformers, 2022
[8] Hierarchical Text-Conditional Image Generation with CLIP Latents, 2022: DALL-E,
[9] Tackling the Generative Learning Trilemma with Denoising Diffusion GANs,
[10] Learning to Generate and Transfer Data with Rectified Flow, Liu X. and Gong C. and Liu Q., 2022
[11] InstaFlow! One-Step Stable Diffusion with Rectified Flow, Liu X and Zhang X. and Jianzhu M and Peng J. and Liu Q., 2023
[12] Variational Diffusion Models, Kingma D. and Salimans T. and Poole Ben and Ho J., 2021
[13] Classifier-Free Diffusion Guidance,
[14] Adding Conditional Control to Text-to-Image Diffusion Models
[15] Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets