Lecture 17: Deep Generative Models VI: DDPM

Lan Xu
SIST, ShanghaiTech
Fall, 2024

Outline

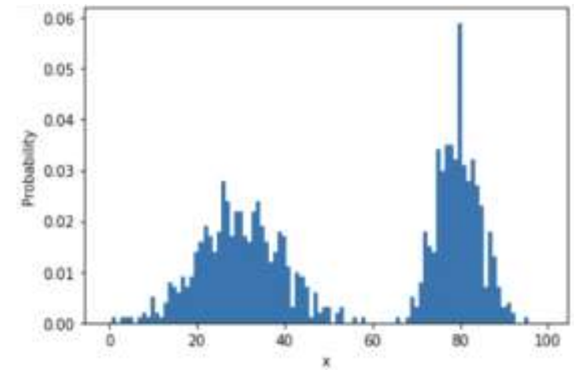
- Diffusion Theory, mostly DDPM
- Diffusion for image generation
- Tricks to improve image synthesis models
- Latent Diffusion Models
- Examples of recent diffusion models

The Landscape of Generative Models

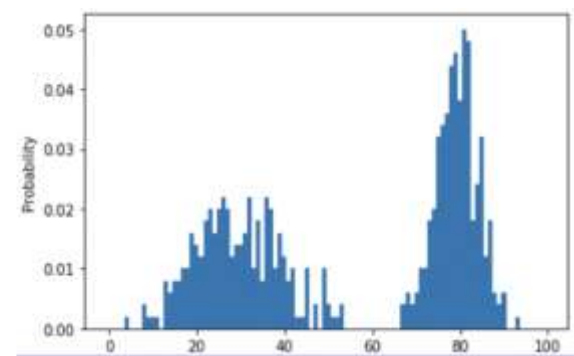
Recap of generative modeling so far

- Assume that all data comes from a distribution $p_{data}(x)$:
- The goal of generative machine learning models is to *learn* this distribution to the best of their ability — the distribution approximated by the model is denoted as $p_\theta(x)$
- We generate new data by *sampling* from the learned distribution
- In practice, train models to maximize the expected log likelihood of $p_\theta(x)$ (or minimizing negative log likelihood)/minimize divergence between $p_\theta(x)$ and $p_{data}(x)$

Training distribution



Samples from learned distribution



Prior methods

- VAE

$$\begin{aligned} L_{\text{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\ &= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\ \theta^*, \phi^* &= \arg\min_{\theta,\phi} L_{\text{VAE}} \end{aligned}$$

$$\boxed{-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}$$

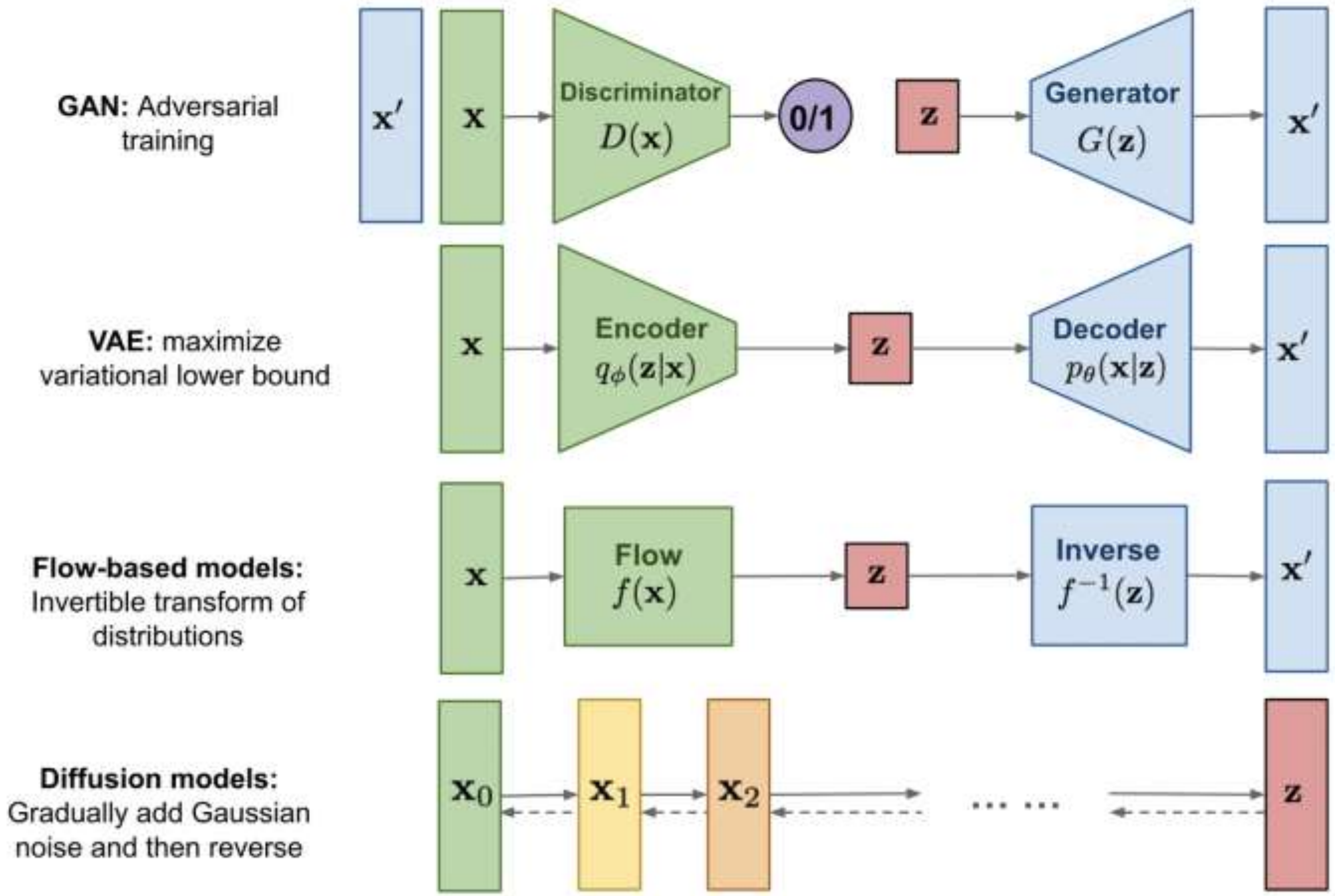
- GAN

$$\begin{aligned} \min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)] \end{aligned}$$

$$\boxed{L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2\log 2}$$

Sampling from noise

- A quick paradigm ……



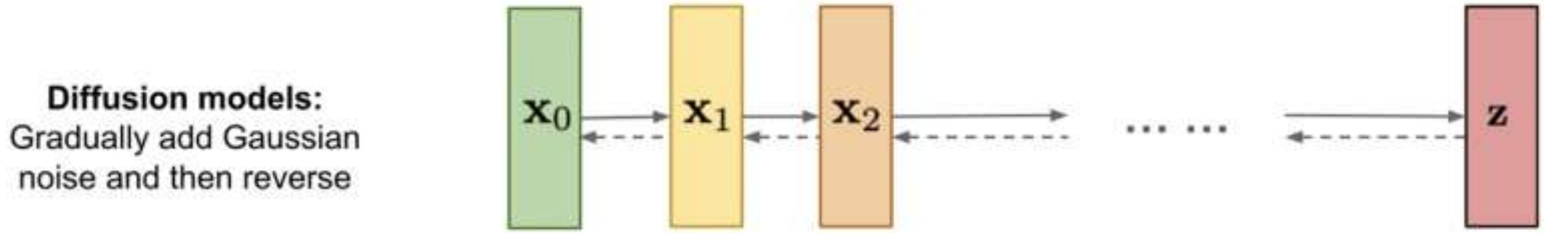
Diffusion Model

- Another kind of generative modeling technique that takes inspiration from physics (**non-equilibrium statistical physics** and **stochastic differential equations** to be more exact)!
- Main idea: convert a well-known and simple base distribution (like a Gaussian) to the target (data) distribution **iteratively**, with small step sizes, via a **Markov chain**:
  - Treat the the output of the Markov Chain as the model's approximation for the learned distribution
  - Inspiration? Estimating and analyzing small step sizes is more tractable/easier than describing a single non-normalizable step from random noise to the learned distribution

    (which is what VAEs/GANs are doing)

Anatomy of a Diffusion Model

- Forward Process
- Reverse Process



DDPM

- Denoising Diffusion Probabilistic Models
- Target: understand the training and sampling phases!

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
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** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
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** $\mathbf{x}_0$

Diffusion models

- What if we add a bunch of Gaussian noise to an image?

Diffusion models

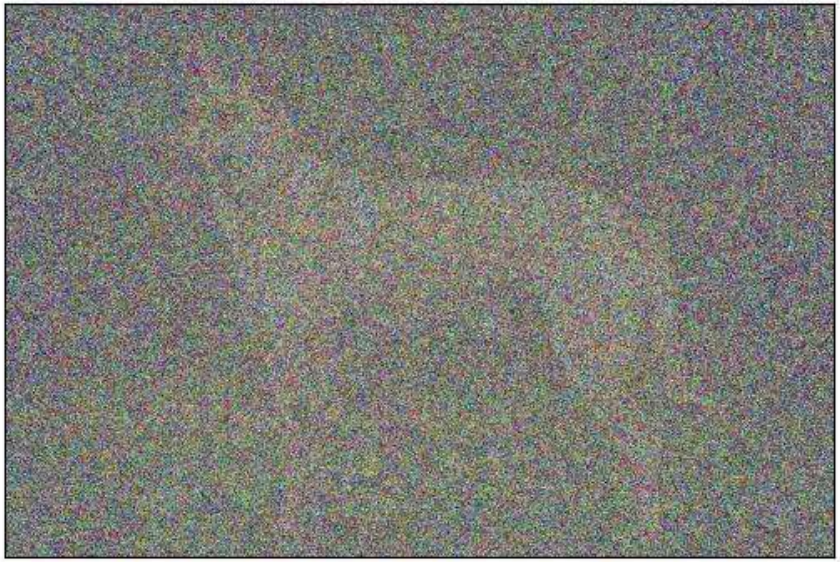
- and again…

Diffusion models

- and again…

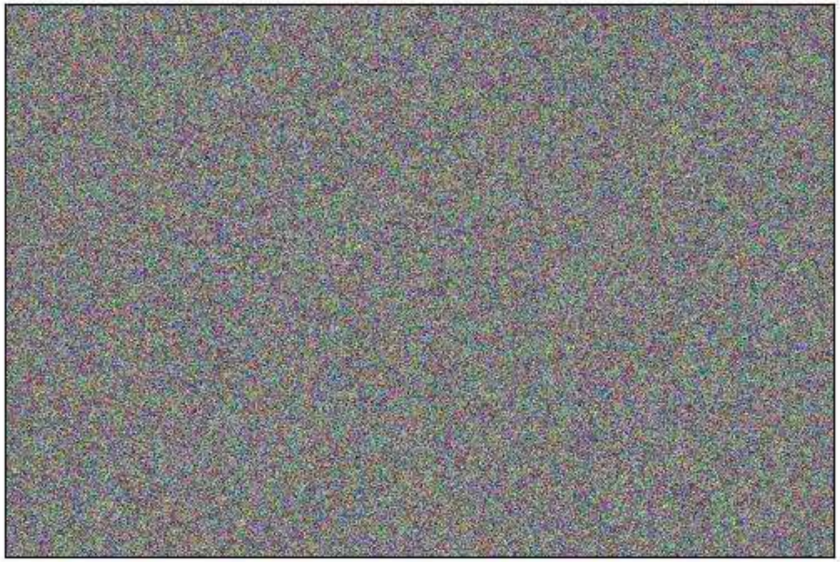
Diffusion models

- and again…



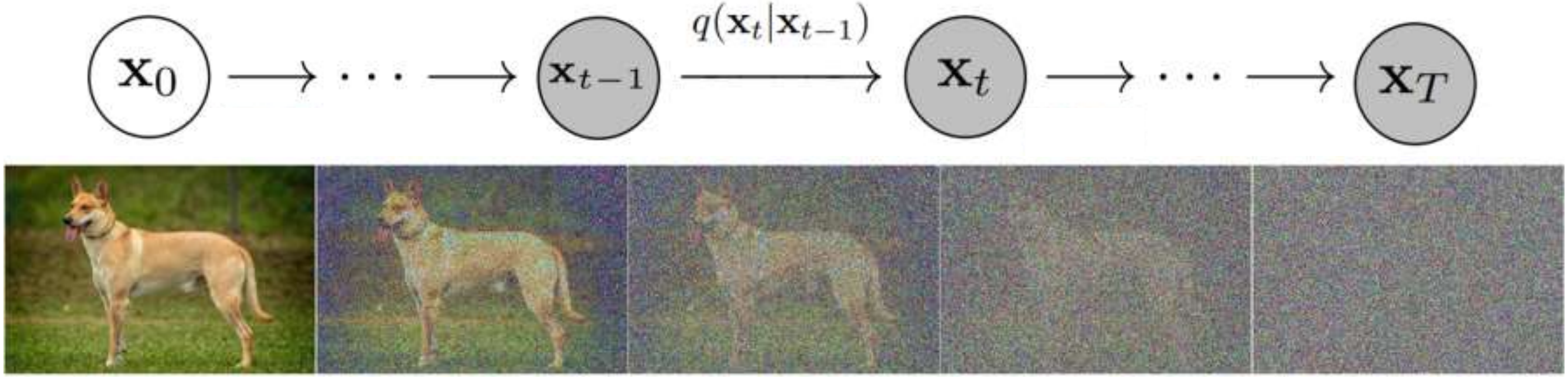
Diffusion models

- … until it resembles pure noise



Diffusion models

- Forward Process



- Take a datapoint **x_0** and gradually add very small amounts of Gaussian noise to it
- Let **x_t** be the datapoint after t iterations
- This is called the **forward diffusion process**
- Repeat this process for **T** steps — over time, more and more features of the original input are destroyed until you get something resembling **pure noise**

Diffusion models

- Forward Process

$$\mathbf{x}_0 \longrightarrow \cdots \longrightarrow \mathbf{x}_{t-1} \xrightarrow{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \mathbf{x}_t \longrightarrow \cdots \longrightarrow \mathbf{x}_T$$

- More formally, we update each image over time as

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0,\, I)$$

where $\{\beta_t \in (0,1)\}_{t=1}^{T}$

is called the **noise schedule** (basically a hyperparameter describing how much noise to add at a given timestep).
The update above can equivalently be written as a sampling process from the following Gaussian distribution:

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

Diffusion models

- A neat (reparametrization) trick! $\{\beta_t \in (0,1)\}_{t=1}^{T}$

$$\beta_1 < \beta_2 < \ldots < \beta_T$$

- Define:

$$\alpha_t = 1 - \beta_t \qquad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

- Then:

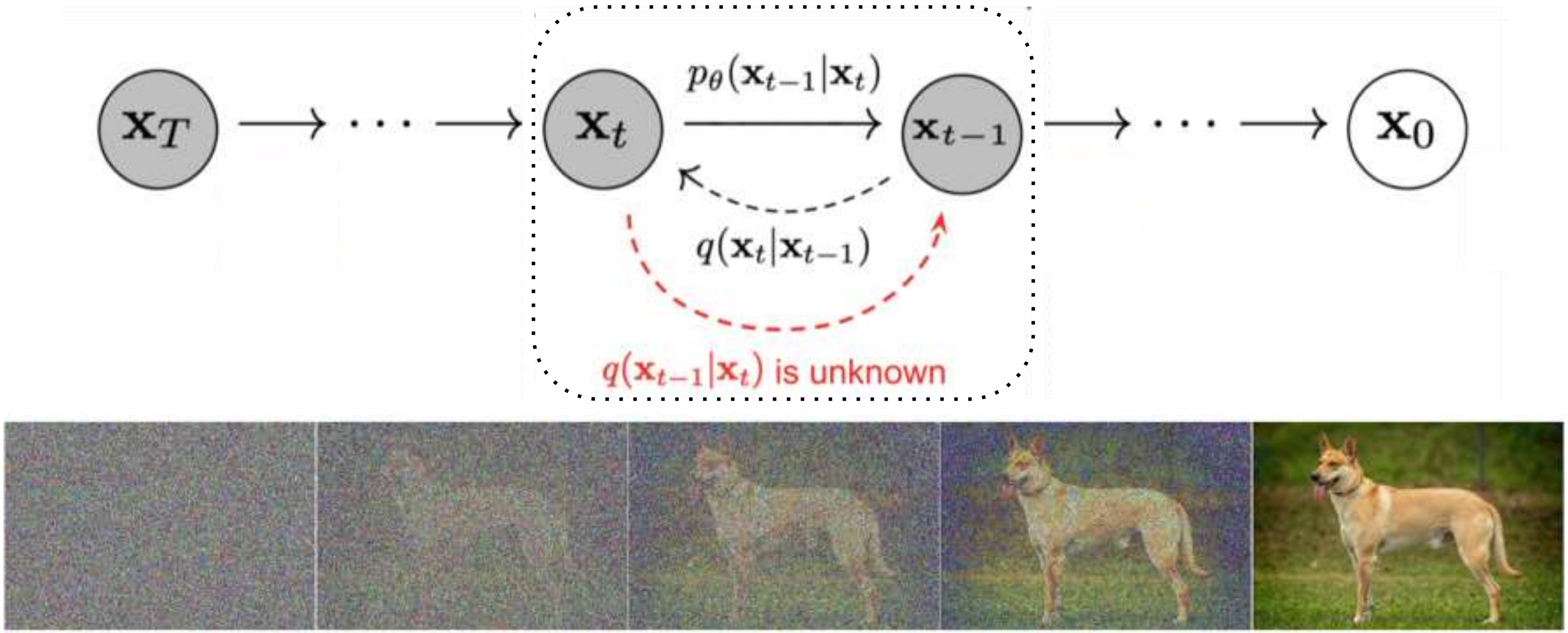
$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right)$$

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \\
&= \ldots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}$$

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

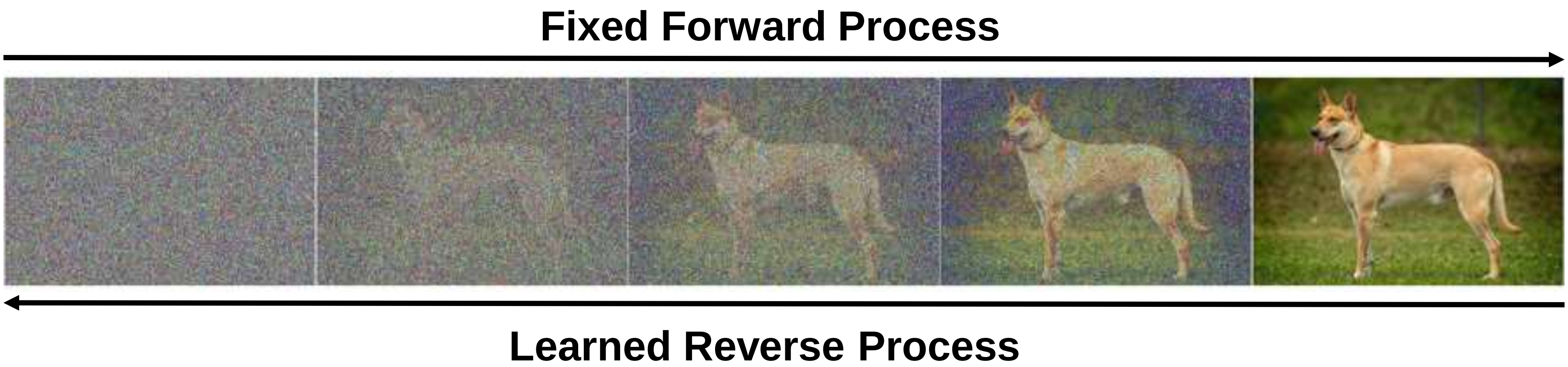
Diffusion models

- Can we go in the other direction?



Diffusion models

- Reverse Process

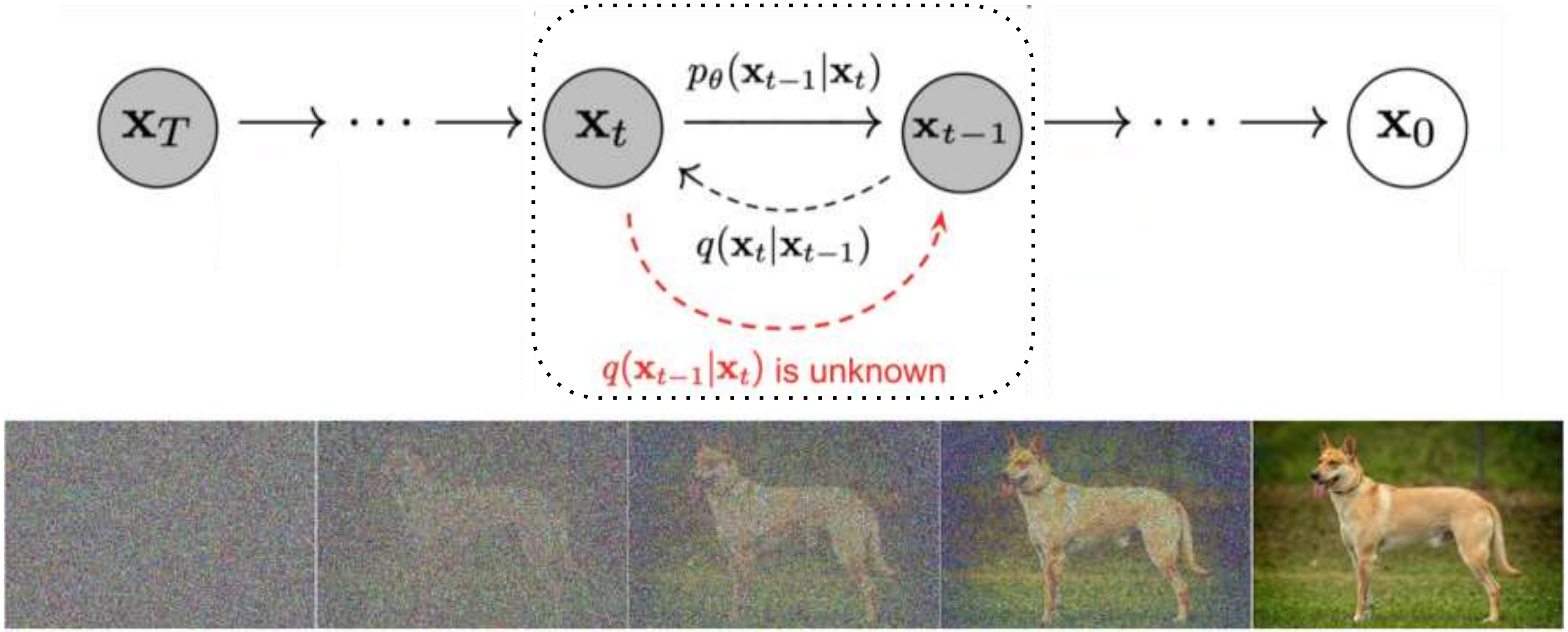


- The goal of a diffusion model is to **learn the reverse denoising process** to iteratively **undo** the forward process
- In this way, the reverse process appears as if it is generating new data from random noise!

Diffusion models

- Reverse Process

We are given **q(x_t | x_{t-1})**. How do we find **q(x_{t-1} | x_t)**?



Diffusion models

- Finding the exact distribution is hard
- Remember Bayes rule?

$$f(\theta \mid x) = \frac{f(\theta, x)}{f(x)} = \frac{f(\theta)\, f(x \mid \theta)}{f(x)} \quad \Longrightarrow \quad q(x_{t-1} \mid x_t) = q(x_t \mid x_{t-1}) \frac{q(x_{t-1})}{q(x_t)}$$

$$q(x_t) = \int q(x_t \mid x_{t-1}) q(x_{t-1}) \mathrm{d}x$$

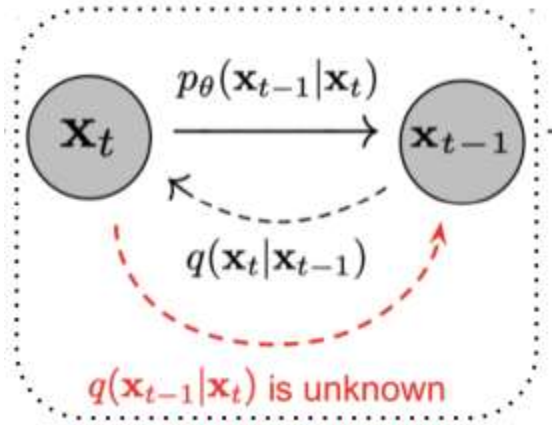
- The distribution of each timestep and q(x_t | x_{t-1}) depends on the entire data distribution:
- This is computationally intractable
  - Need to integrate over the whole data distribution to find **q(x_t)** and **q(x_{t-1})**
  - Where else have we seen this dilemma?
- We still need the posterior distribution to carry out the reverse process. Can we approximate this somehow?

Diffusion models

- Can we go in the other direction?
- A naïve solution, don't work:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon$$

$$x_{t-1} = (x_t - \sqrt{\beta_t}\epsilon_{t-1})/\sqrt{1-\beta_t}$$



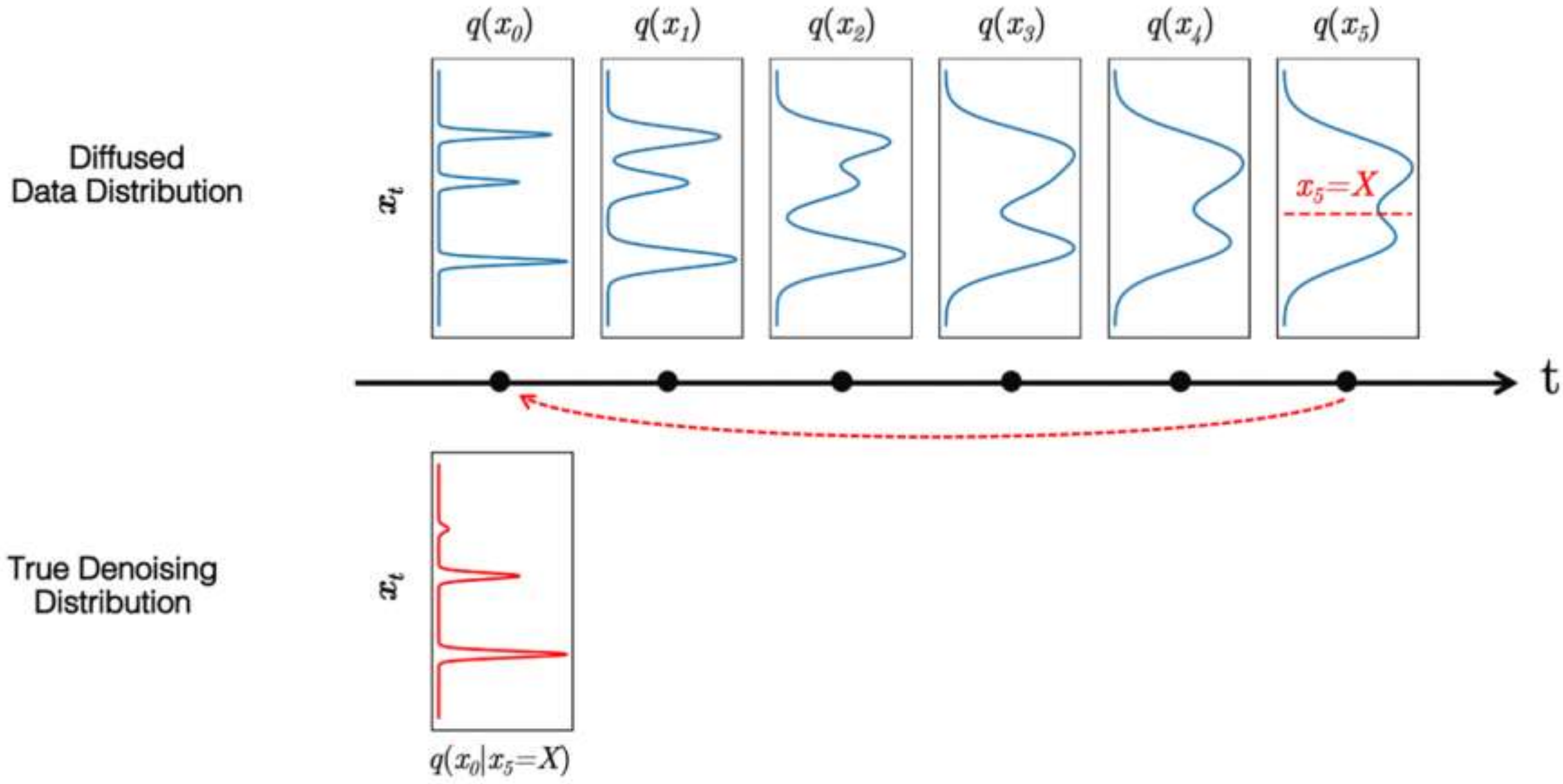
- Then, let a NN estimate ϵ_{t-1}

$$x_{t-1} = (x_t - \sqrt{\beta_t}\epsilon_\theta(x_t, t))/\sqrt{1-\beta_t}$$

- Problem: interactive training, super non-efficient

- **Solution in DDPM**: use the reparametrization trick, from $q(x_{t-1}|x_t)$ to $q(x_{t-1}|x_t, x_0)$

Denoising Diffusion Probabilistic Models



Denoising Diffusion Probabilistic Models

- What does the final reverse process look like?

- In practice, we choose **our noise schedule** such that the forward process steps are **very small**.

$$\{\beta_t \in (0,1)\}_{t=1}^T$$

- Thus, we approximate the reverse posterior distributions

$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ as Gaussians and **learn** their parameters (i.e., the mean and variance) via neural networks

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

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

- Cool, we now have an idea of what the model looks like. How do we train it?

DDPM models

- A preliminary objective

We want to maximize the log-likelihood of the data generated by a reverse process.

Remember that VAEs tried to do something similar but they maximized a lower bound on the likelihood instead because the actual likelihood is computationally intractable

$$\boxed{-L_{\mathrm{VAE}} = \log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}$$

We can apply the same trick to diffusion!

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

DDPM models

- A preliminary objective

The VAE (ELBO) loss is a bound on the true log likelihood (also called the *variational lower bound*)

$$\boxed{-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}$$

Apply the same trick to diffusion:

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

In details:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\text{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big]
\end{aligned}$$

$$\text{Let } L_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)$$

DDPM models

- A preliminary objective

The VAE (ELBO) loss is a bound on the true log likelihood (also called the *variational lower bound*)

$$\boxed{-L_{\mathrm{VAE}} = \log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}$$

Apply the same trick to diffusion:

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

Expanding out,

$$\begin{aligned}
L_{\mathrm{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where } L_T &= D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)) \\
L_t &= D_{\mathrm{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \text{ for } 1 \leq t \leq T-1 \\
L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}$$

DDPM models

- A more thorough derivation

$$
\begin{aligned}
L &= \mathbb{E}_q\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t\geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_q\left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= \mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,\|\,p(\mathbf{x}_T))}_{L\_T} + \underbrace{\sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,\|\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L\_t} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L\_0}\Big]
\end{aligned}
$$

DDPM models

- A simplified objective: use the **reparametrization trick**, from $q(x_{t-1}|x_t)$ to $q(x_{t-1}|x_t, x_0)$

- The reverse step conditioned on x_0 is a **Gaussian**:

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

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

- After **doing some algebra**, each loss term can be approximated by:

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I)$$

$$\alpha_t = 1 - \beta_t \quad \text{and} \quad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_t$$

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t,, t)\|_2^2\right]$$

$$= \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\left\|\boxed{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)} - \mu_\theta(\mathbf{x}_t,, t)\right\|_2^2\right]$$

DDPM models

- Some algebra here ……
- The reverse step conditioned on x_0 is a **Gaussian**

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_{t-1}, x_t, x_0)}{q(x_t, x_0)} = \frac{q(x_t|x_{t-1}, x_0) \cdot q(x_{t-1}, x_0)}{q(x_t, x_0)} = \frac{q(x_t|x_{t-1}, x_0) \cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$$
$$= \frac{q(x_t|x_{t-1}) \cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

- Note that: $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \boldsymbol{I})$
- Let's handle: $q(x_t|x_0)$ using the **reparametrization trick**

$$\begin{aligned}
q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) &= \mathcal{N}\left(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \\
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \\
q(\mathbf{x}_t \mid \mathbf{x}_0) &= \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right)
\end{aligned}$$

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

DDPM models

- Some algebra here …… $\frac{q(x_t|x_{t-1})\cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$
- All are Gaussians now：
  → If $x \sim \mathcal{N}(\mu, \sigma^2)$, then $q(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2})$
- Thus, $\boxed{q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \boldsymbol{I})}$

  where $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
- Recall $x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{1-\overline{\alpha}_t}\,\overline{\epsilon}_t$ and $x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\,\overline{\epsilon}_t)$
- Thus, $\boxed{\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\overline{\epsilon}_t)}$, use NN to estimate it !!
- Only rely on $\overline{\epsilon}_t$, from x_0 to x_t, with only one sampling!

$$L = \mathbb{E}_{\mathbf{x}_0,\,\epsilon}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t,\, t)\|_2^2\right] = \mathbb{E}_{\mathbf{x}_0,\,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\, t\right)\right\|_2^2\right]$$

DDPM models

- A simplified objective: use the **reparametrization trick**
- Instead of predicting the **mu**, Ho et al. say that we should predict **epsilon** instead!

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

- Thus, our loss becomes:

$$\begin{aligned}
L_{t-1} &= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) - \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)\right\|_2^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|_2^2\right]
\end{aligned}$$

DDPM models

- The authors of DDPM say that it's fine to drop all that baggage in the front and instead just use

$$
\begin{aligned}
L &= \mathbb{E}_{\mathbf{x}_0,\,\epsilon}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t,\, t)\|_2^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0,\,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\, t\right)\right\|_2^2\right]
\end{aligned}
$$

- Note that this is not a variational lower bound on the log-likelihood anymore: in fact, you can view it as a **reweighted version of ELBO** that emphasizes reconstruction quality!

DDPM models

- Training

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\boxed{\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)} \right\|^2$$

6: **until** converged

DDPM models

- Sampling

Algorithm 2 Sampling

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

DDPM models

- Some algebra here …… $\frac{q(x_t|x_{t-1})\cdot q(x_{t-1}|x_0)}{q(x_t|x_0)}$
- All are Gaussians now:
  → If $x \sim \mathcal{N}(\mu, \sigma^2)$, then $q(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2})$
- Thus, $\boxed{q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \boldsymbol{I})}$

  where $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
- Recall $x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\overline{\epsilon}_t$ and $x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}(x_t - \sqrt{1-\overline{\alpha}_t}\overline{\epsilon}_t)$
- Thus, $\boxed{\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\overline{\epsilon}_t)}$, use NN to estimate it !!
- Only rely on $\overline{\epsilon}_t$, from x_0 to x_t, with only one sampling!

$$L = \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|_2^2\right] = \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)\right\|_2^2\right]$$

DDPM models

- If we have the noise, sampling by using Gaussians:

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

- 1) sampling z_t
- 2) sampling x_t-1, using the estimated noise

$$x_{t-1} = \tilde{\mu}_t + \tilde{\beta}_t \cdot z_t = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\overline{\epsilon}_t) + \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t} \cdot \beta_t \cdot z_t$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t)) + \frac{1-\overline{\alpha}_{t-1}}{1-\overline{\alpha}_t} \cdot \beta_t \cdot z_t$$

DDPM models

- Rethinking the Training and Sampling processes…..

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

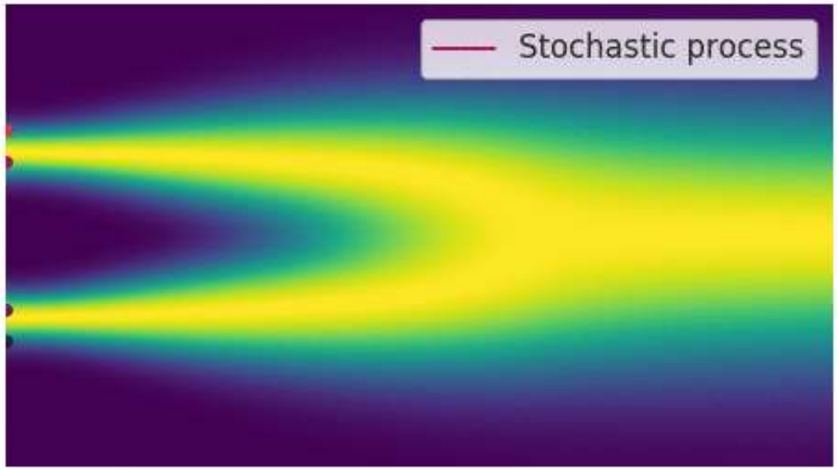
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
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** $\mathbf{x}_0$

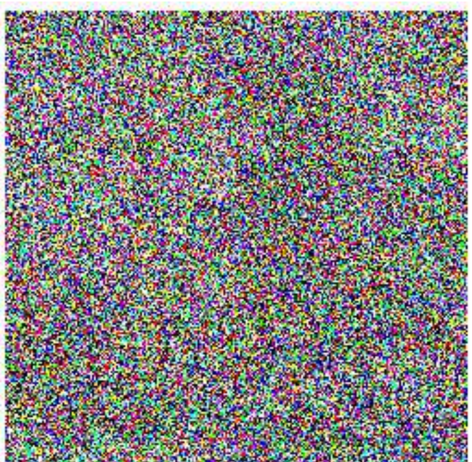
- During training, add noise from 0 to t, then estimate it
- During sampling, note that $\sigma_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$
- As t increases, $\bar{\alpha}_t$ decreases, $\sqrt{1-\bar{\alpha}_t}$ increases
- Thus, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ works as denoise auto-encoder for various noise levels!

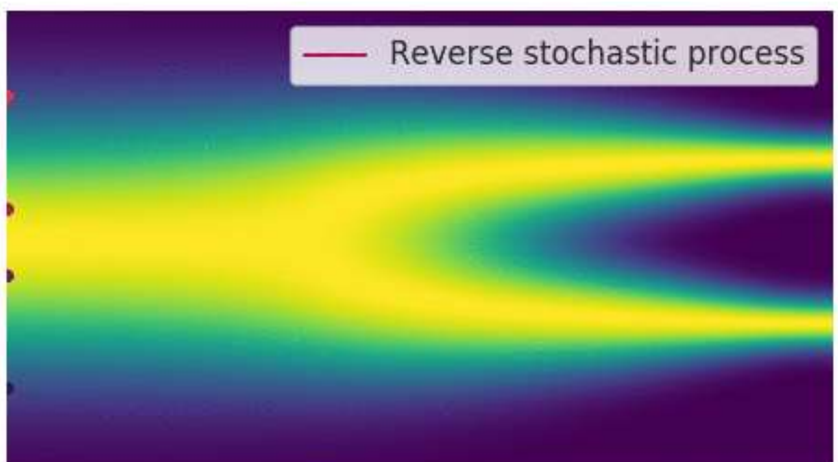
DDPM models

- Forward/Reverse process for Image Generation



Forward process: converting the image distribution to pure noise

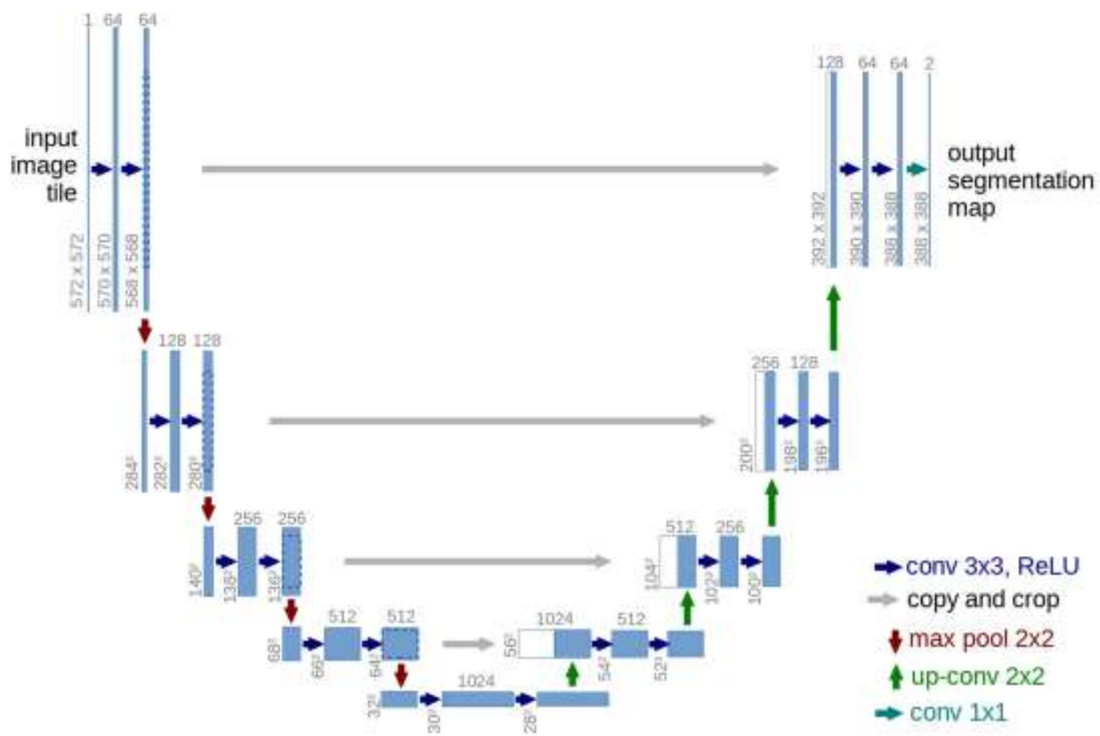




Reverse process: sampling from the image distribution, starting with pure noise

DDPM models

- UNet + Other Stuff



Diffusion models typically use a U-Net on steroids as the noise predictive model — you take the good ol' model that you are already familiar with and add:

- Positional Embeddings
- ResNet Blocks
- ConvNext Blocks
- Attention Modules
- Group Normalization
- Swish and GeLU

It's a massive kitchen sink of modern CV tricks

Tricks for Improving Generation

- Linear vs Cosine **Schedule**
- A linear noise schedule converts initial data to noise really quickly, making the reverse process harder for the model to learn.
- Researchers hypothesized that a cosine-like function that is changing relatively gradually near the endpoints might work better
- Note: It did end up working better but this choice of cosine was completely arbitrary

Tricks for Improving Generation

- Learning a Covariance matrix

- DDPM authors said that it's better to use a fixed covariance matrix $\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2\mathbf{I}$ where $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$.
  - The intuition is that covariance does not contribute as significantly as the mean does to the learned conditional distributions during the reverse process
  - However, it can still help us improve log-likelihood!
- So, Nichol and Dhariwal propose

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t)$$

This modification leads to better likelihood estimates while maintaining image quality!

Tricks for Improving Generation

- Architecture Improvements
- Nichol and Dhariwal proposed several architectural changes that seem to help diffusion training:
  - Increasing model depth vs width (not both): both help but increasing width is computationally cheaper while providing similar gains as increased depth
  - Increasing number of attention heads and applying it to multiple resolutions
  - Stealing BigGAN residual blocks for upsampling and downsampling
  - Adaptive Group Normalization — hopes to better incorporate timestep (and potentially class) information during the training/reverse process

Tricks for Improving Generation

- Classifier Guidance

- Recall conditional GANs from the previous lectures: they can be conditioned on class labels to synthesize specific kinds of images. We can apply the same idea to diffusion!
- The main idea is this:
  - Take a pre-trained unconditional diffusion model
  - During sampling, inject the gradients of a classifier model (that is trained from scratch on noisy images) into the unconditional reverse process
  - Classifier guidance trades off image diversity for model fidelity, allowing it to push the performance of a diffusion model past that of a GAN

Classifier 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

Classifier Guidance

At a high level:

- FID and sFID captures image quality
- Precision measures image fidelity (“resemblance to training images”)
- Recall measures image diversity/distribution coverage

Lower FID/sFID is better

Higher Precision and Recall is better

Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
DCTransformer† [42]	6.40	6.66	0.44	**0.56**
DDPM [25]	4.89	9.07	0.60	0.45
IDDPM [43]	4.24	8.21	0.62	0.46
StyleGAN [27]	2.35	6.62	0.59	0.48
ADM (dropout)	**1.90**	**5.59**	**0.66**	0.51
LSUN Horses 256×256				
StyleGAN2 [28]	3.84	6.46	0.63	0.48
ADM	2.95	**5.94**	0.69	**0.55**
ADM (dropout)	**2.57**	6.81	**0.71**	**0.55**
LSUN Cats 256×256				
DDPM [25]	17.1	12.4	0.53	0.48
StyleGAN2 [28]	7.25	**6.33**	0.58	0.43
ADM (dropout)	**5.57**	6.69	**0.63**	**0.52**
ImageNet 64×64				
BigGAN-deep* [5]	4.06	3.96	**0.79**	0.48
IDDPM [43]	2.92	**3.79**	0.74	0.62
ADM	2.61	**3.77**	0.73	0.63
ADM (dropout)	**2.07**	4.29	0.74	**0.63**

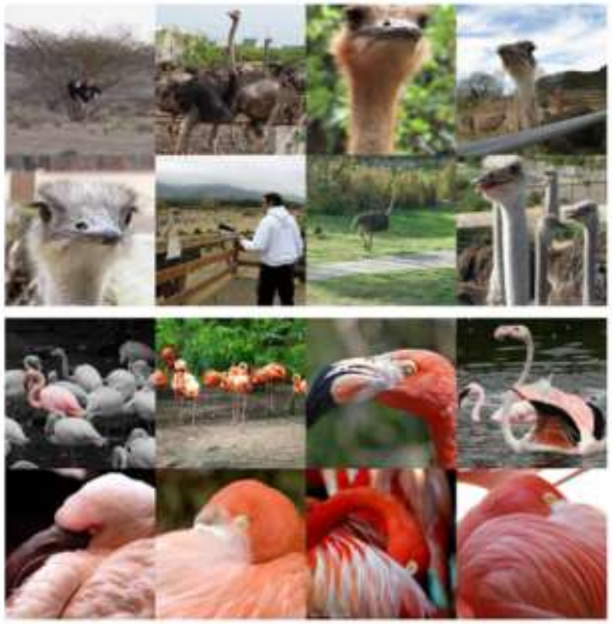
Model	FID	sFID	Prec	Rec
ImageNet 128×128				
BigGAN-deep [5]	6.02	7.18	**0.86**	0.35
LOGAN† [68]	3.36			
ADM	5.91	**5.09**	0.70	**0.65**
ADM-G (25 steps)	5.98	7.04	0.78	0.51
ADM-G	**2.97**	**5.09**	0.78	0.59
ImageNet 256×256				
DCTransformer† [42]	36.51	8.24	0.36	**0.67**
VQ-VAE-2†‡ [51]	31.11	17.38	0.36	0.57
IDDPM‡ [43]	12.26	5.42	0.70	0.62
SR3†‡ [53]	11.30			
BigGAN-deep [5]	6.95	7.36	**0.87**	0.28
ADM	10.94	6.02	0.69	0.63
ADM-G (25 steps)	5.44	5.32	0.81	0.49
ADM-G	**4.59**	**5.25**	0.82	0.52
ImageNet 512×512				
BigGAN-deep [5]	8.43	8.13	**0.88**	0.29
ADM	23.24	10.19	0.73	**0.60**
ADM-G (25 steps)	8.41	9.67	0.83	0.47
ADM-G	**7.72**	**6.57**	0.87	0.42

Diffusion Models Beats GANs

BigGAN

Diffusion

Training Set



Diffusion Models Beats GANs

BigGAN

Diffusion

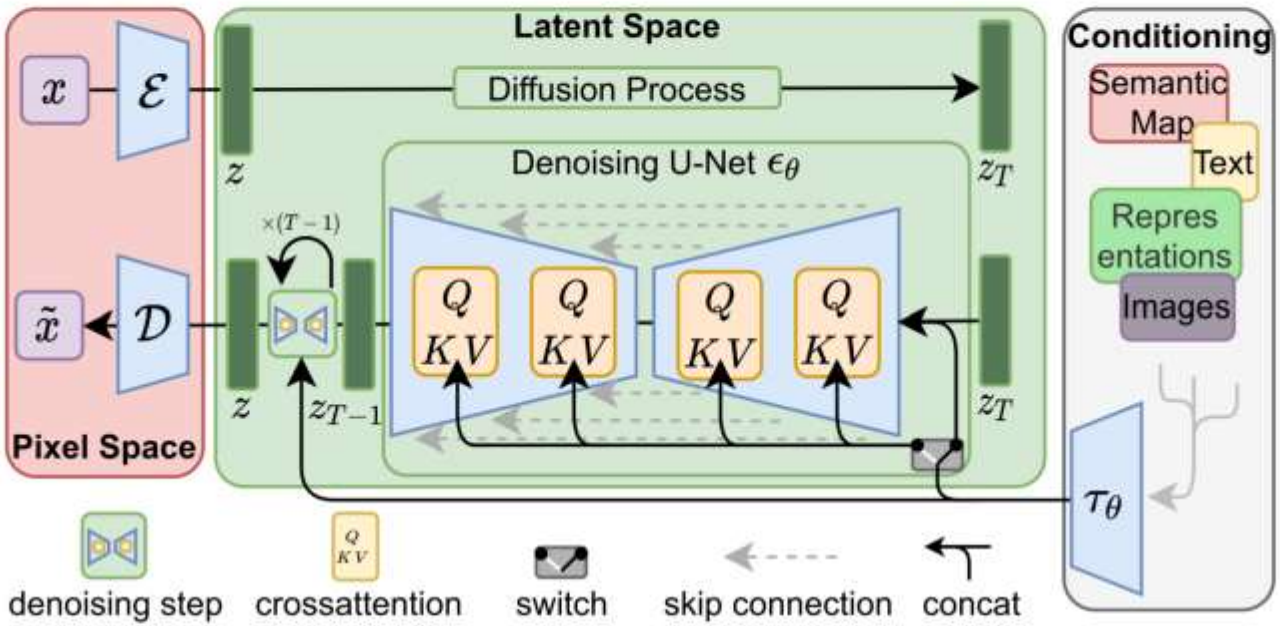
Training Set

Latent Diffusion Models

- Training models in the pixel space is excessively computationally expensive (can easily multiple days on a V100 GPU)
  - Even image synthesis is very slow compared to GANs
  - Images are high dimensional → more things to model
- Researchers observed that most “bits” of an image contribute to its perceptual characteristics since aggressively compressing it usually maintains its semantic and conceptual composition
  - In layman’s terms, there are more bits for describing pixel-level details while less bits for describing “the meaning” within an image
  - Generative models should learn the latter
- Can we separate these two components?

Huge success of text-2-img!

- Stable Diffusion model (CVPR2022)



High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach[1] * Andreas Blattmann[1] * Dominik Lorenz[1] Patrick Esser[R] Björn Ommer[1]

[1]Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany [R]Runway ML

https://github.com/CompVis/latent-diffusion

Latent Diffusion Models

Latent Diffusion Models can be divided **into two stages**:

1）Training perceptual compression models that strip away irrelevant high-level details and learn a latent space that is semantically equivalent to the high level image pixel-space

a. The loss is a combination of a reconstruction loss, an adversarial loss (remember GANs?) that promotes high quality decoder reconstruction, and regularization terms

$$L_{\text{Autoencoder}} = \min_{\mathcal{E},\mathcal{D}} \max_{\psi} \Big(L_{rec}(x, \mathcal{D}(\mathcal{E}(x))) - L_{adv}(\mathcal{D}(\mathcal{E}(x))) + \log D_{\psi}(x) + L_{reg}(x; \mathcal{E}, \mathcal{D}) \Big)$$

2）Performing a diffusion process *in this latent space*. There are several benefits to this:

a. The diffusion process is only focusing on the relevant semantic bits of the data
b. Performing diffusion in a low dimensional space is significantly more efficient

Huge success of text-2-img!

- Stable Diffusion model (CVPR2022)
- Long story between Stability AI, Runway ML and LAION-5B
- AI paradigm: data + algorithm + computing resource

Computer Vision & Learning Group
Ludwig Maximilian University of Munich
(LMU)

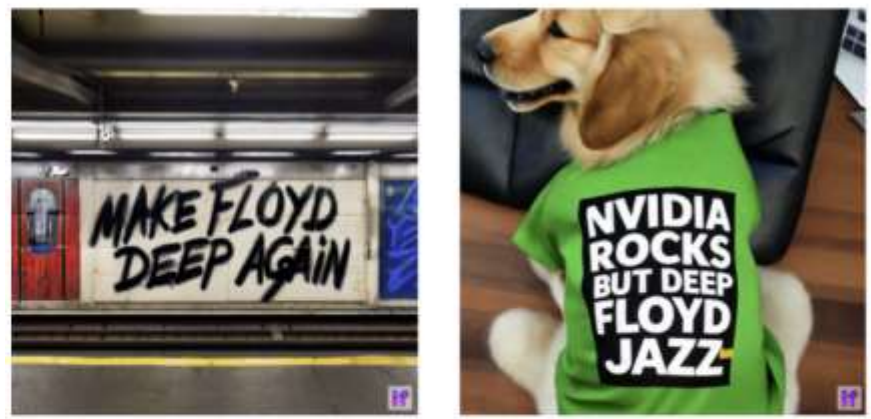
~4000 A100 from
Stability AI

Huge text-image
dataset from
LAION

Huge success of text-2-img!

- The multi-modality framework is important
- The trend continues: big data, big modal ……
- Enjoy better text-encoder and suitable generator

Google: Parti Model, "Scaling Autoregressive Models for Content-Rich Text-to-Image Generation"



Stablity AI: DeepFloyd IF Model

T5-XXL as Text-encoder; pixel-level Diffusion

Huge success of text-2-img!

- Enjoy cross-modality abilities
- Enjoy downstream conditioning abilities

Subject-Driven Generation using **DreamBooth**

Conditioning using **ControlNet**

Editing Instructions using **InstructPix2Pix** (based on GPT-3)

Huge success of text-2-img-3D!

- Use NeRF as inherent representation to bridge 2D-DM with 3D scene
- More explicit disentanglement towards geometry, color, lighting …

3D Editing Instructions
using **InstructNeRF2NeRF**

NVDIA **Magic3D** 18 Nov 2022

OpenAI **Point-E** 21 Dec 2022

DALLE 2 (Text-to-Image)

Imagen (Text-to-Image)

A majestic oil painting of a raccoon Queen wearing red French royal gown.

A robot couple fine-dining with the Eiffel Tower in the background

Video Diffusion (Text-to-Video)

Make-A-Video (Text-to-Video)

Make-A-Video (Text-to-Video)

Imagen Video (Text-to-Video)

DreamFusion (Text-to-3D)

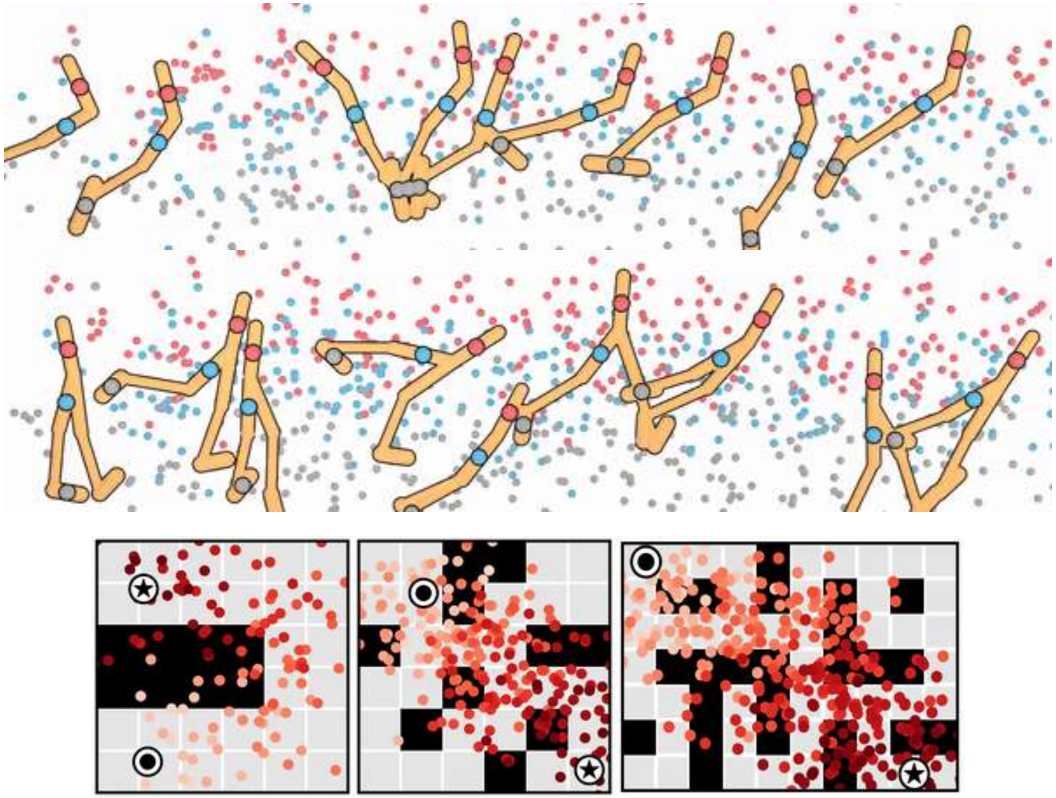
a fox holding a video game controller

a lobster playing the saxophone

a corgi wearing a beret and holding a baguette, standing up on two hind legs

a human skeleton drinking a glass of red wine

Diffuser (Trajectory Planning)



Summary

- A quick tour of generative modeling and how image synthesis can be viewed as sampling from a density
- Preliminary theory of diffusion (don’t worry if this is confusing!)
- Some tricks that modern diffusion models employ for image generation:
  - A U-Net architecture equipped with all kinds of modifications
  - Other architecture improvements
  - Several implementation tricks (different noise schedules, covariance parametrizations)
- Latent diffusion models for improving diffusion quality and efficiency

Summary and Resources

- Deep Unsupervised Learning using Nonequilibrium Thermodynamics: https://arxiv.org/pdf/1503.03585.pdf
- Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2006.11239.pdf
- Improved Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2102.09672.pdf
- Diffusion Models Beat GANs on Image Synthesis: https://arxiv.org/pdf/2105.05233.pdf
- Classifier-free Diffusion Guidance: https://arxiv.org/pdf/2207.12598.pdf
- High Resolution Image Synthesis with Latent Diffusion Models: https://arxiv.org/pdf/2112.10752.pdf
- Denoising Diffusion Implicit Models: https://arxiv.org/pdf/1503.03585.pdf
- Generative Modeling by Estimating Gradients of the Data Distribution: https://yang-song.net/blog/2021/score/
- Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions: https://arxiv.org/pdf/2209.11215.pdf

Summary and Resources

- Lillian Weng's Blog: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- The Annotated Diffusion Model: https://huggingface.co/blog/annotated-diffusion
- The Illustrated Stable Diffusion: https://jalammar.github.io/illustrated-stable-diffusion/
- PyTorch implementation of the DDPM Unet: https://nn.labml.ai/diffusion/ddpm/unet.html
- Guidance: a cheat code for diffusion models: https://benanne.github.io/2022/05/26/guidance.html
- Understanding Diffusion Models: A Unified Perspective: https://arxiv.org/pdf/2208.11970.pdf