

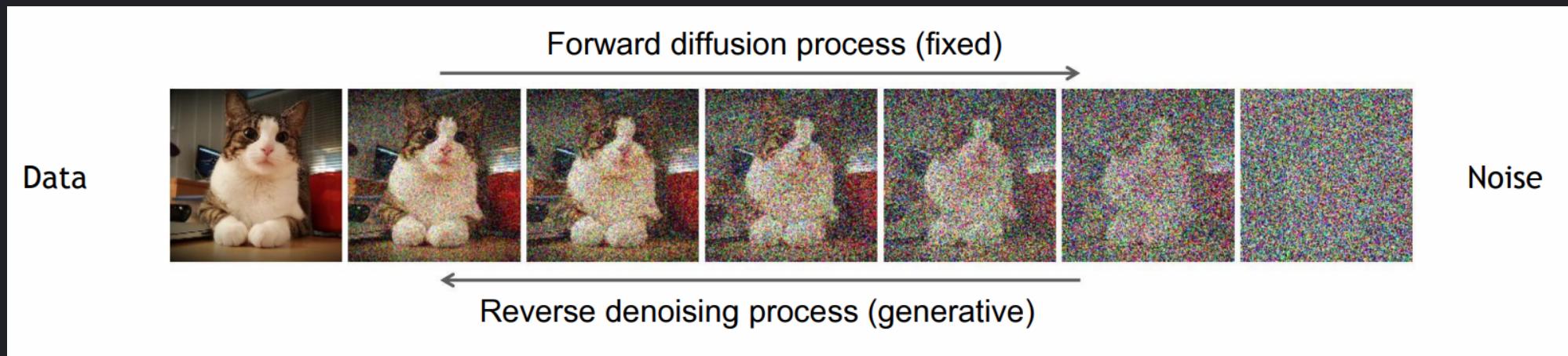
Diffusion Models

An Introduction

Thodoris Kouzelis

Learning to Generate by Denoising

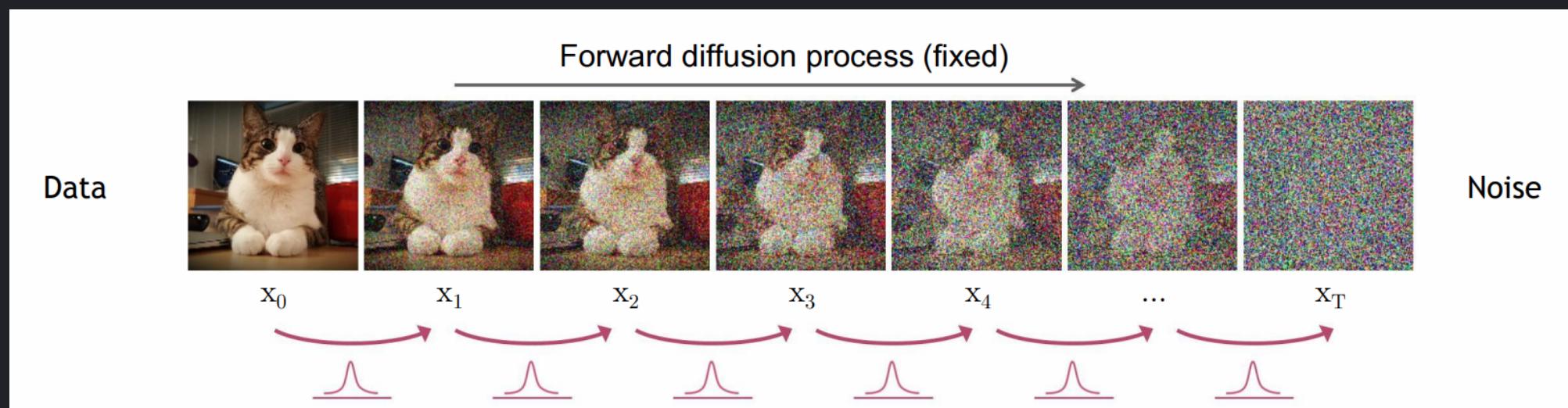
- Diffusion models consist of two processes:
 - Forward process that gradually adds noise to input.
 - Reverse process that learns to generate data by denoising.



Forward Process

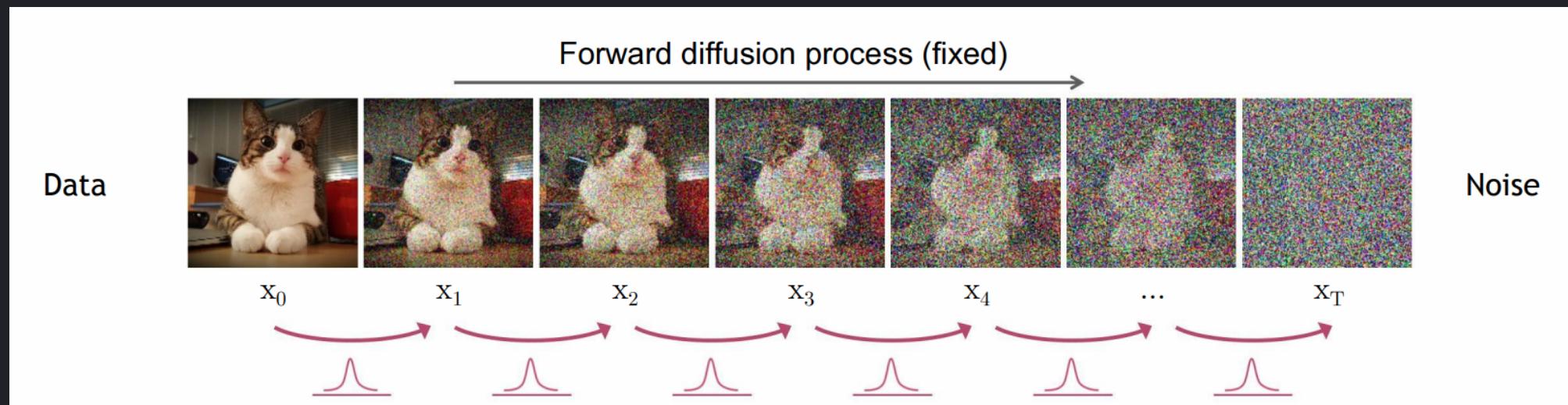
- The definition of the forward process in T steps:

- Posterior: $q(x_{1:T}|x_0) = \prod_{t=1}^T q(x_t|x_{t-1})$,
 - where $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I)$
 - We will often denote $a_t = 1 - \beta_t$



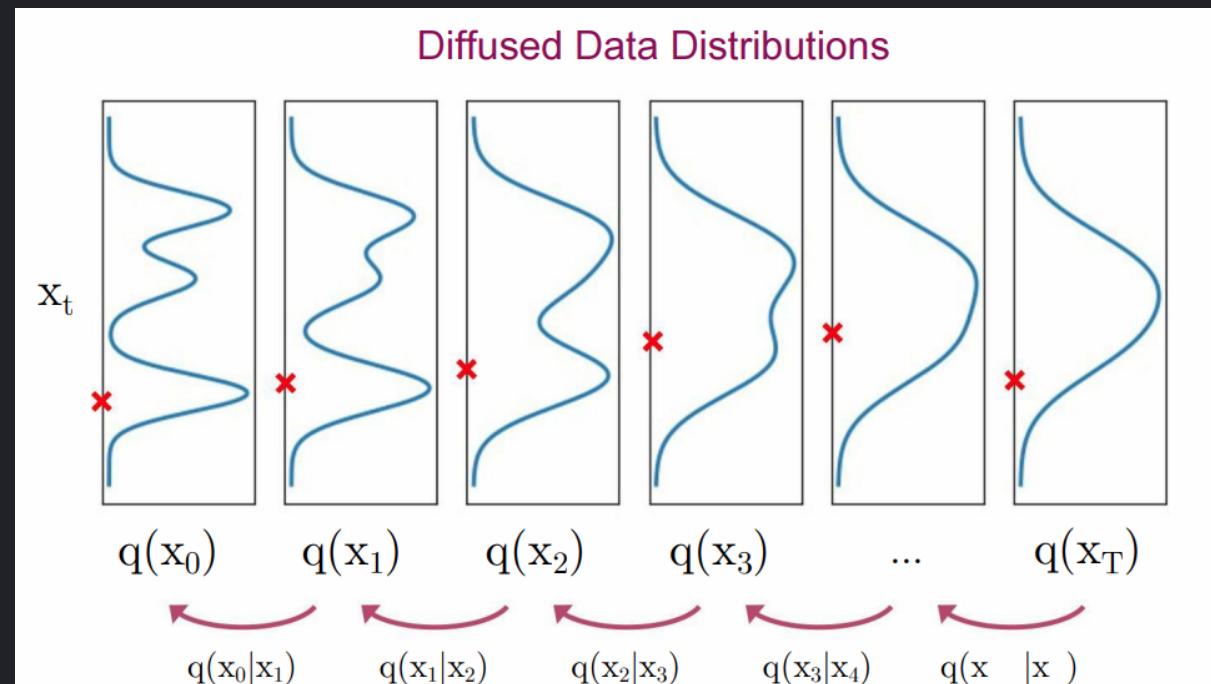
Characteristics of the forward process

- Markov Chain: x_t only depends on x_{t-1} .
- Generally $\beta_t \in (0, 1)$ follows a fixed linearly increasing schedule.
- As $T \rightarrow \infty$, $q(x_T|x_0) \approx \mathcal{N}(0, I)$.
- If we set a large enough T we can set $\beta_t \ll 1$.

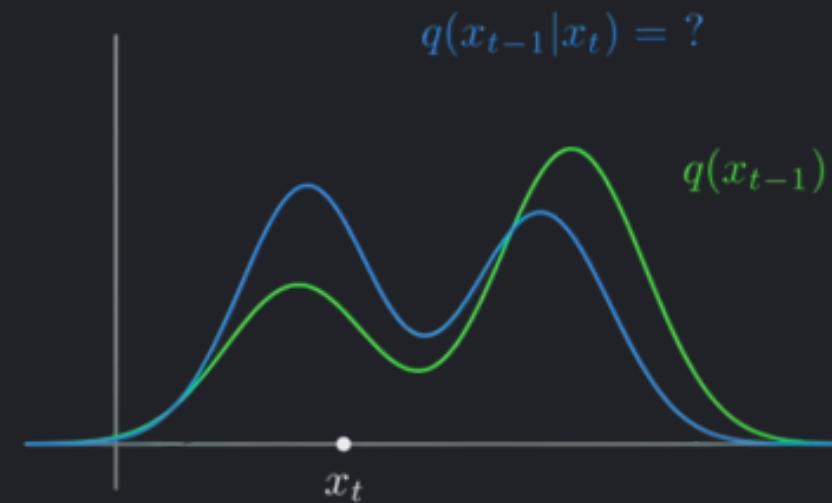
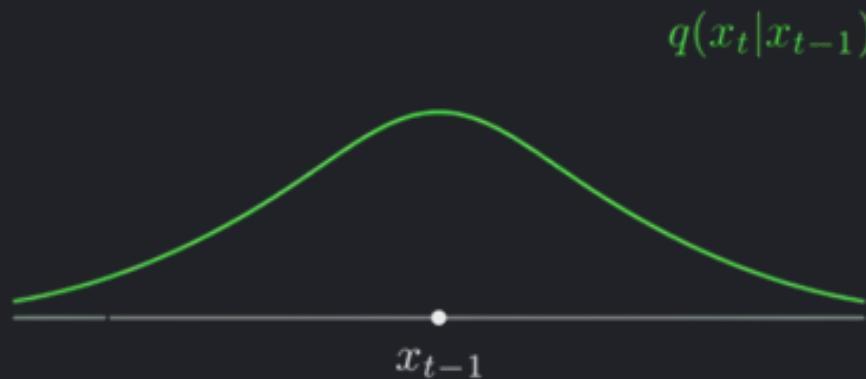


Backward Process

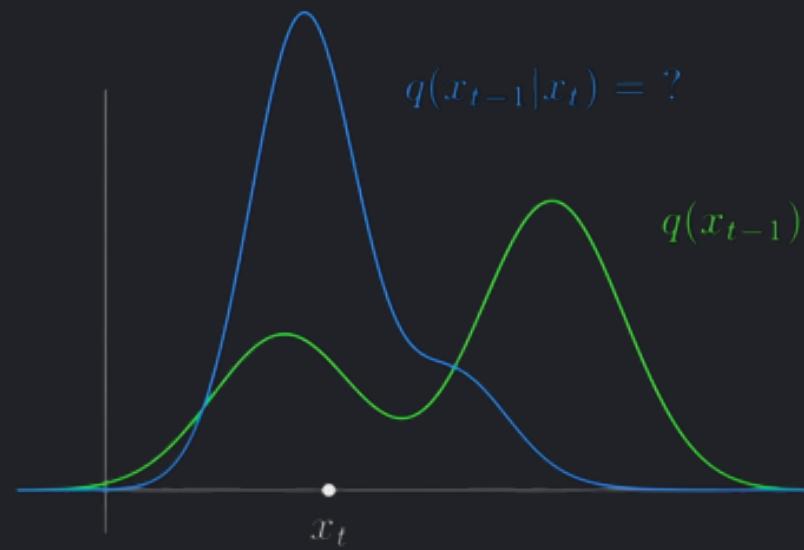
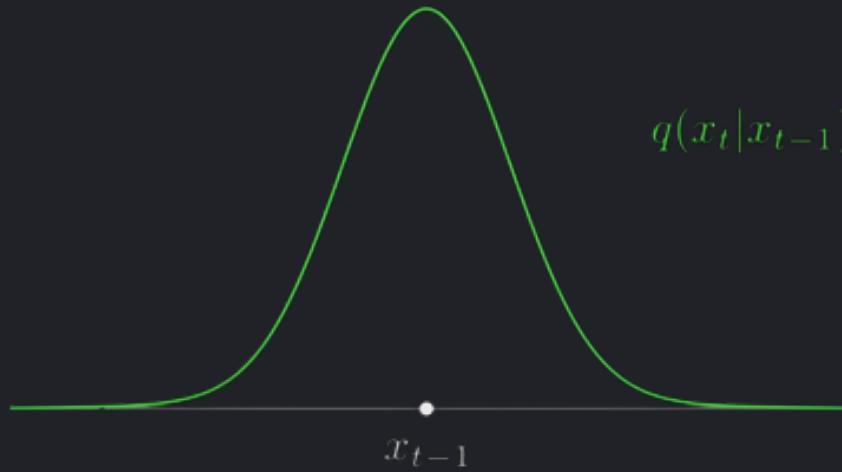
- We have to approximate $q(x_{t-1}|x_t)$.
- We can model $q(x_{t-1}|x_t)$ as a Gaussian Distribution if β_t is small.
- Why?



Why can we model $q(x_{t-1}|x_t)$ as a Gaussian?



Why can we model $q(x_{t-1}|x_t)$ as a Gaussian?

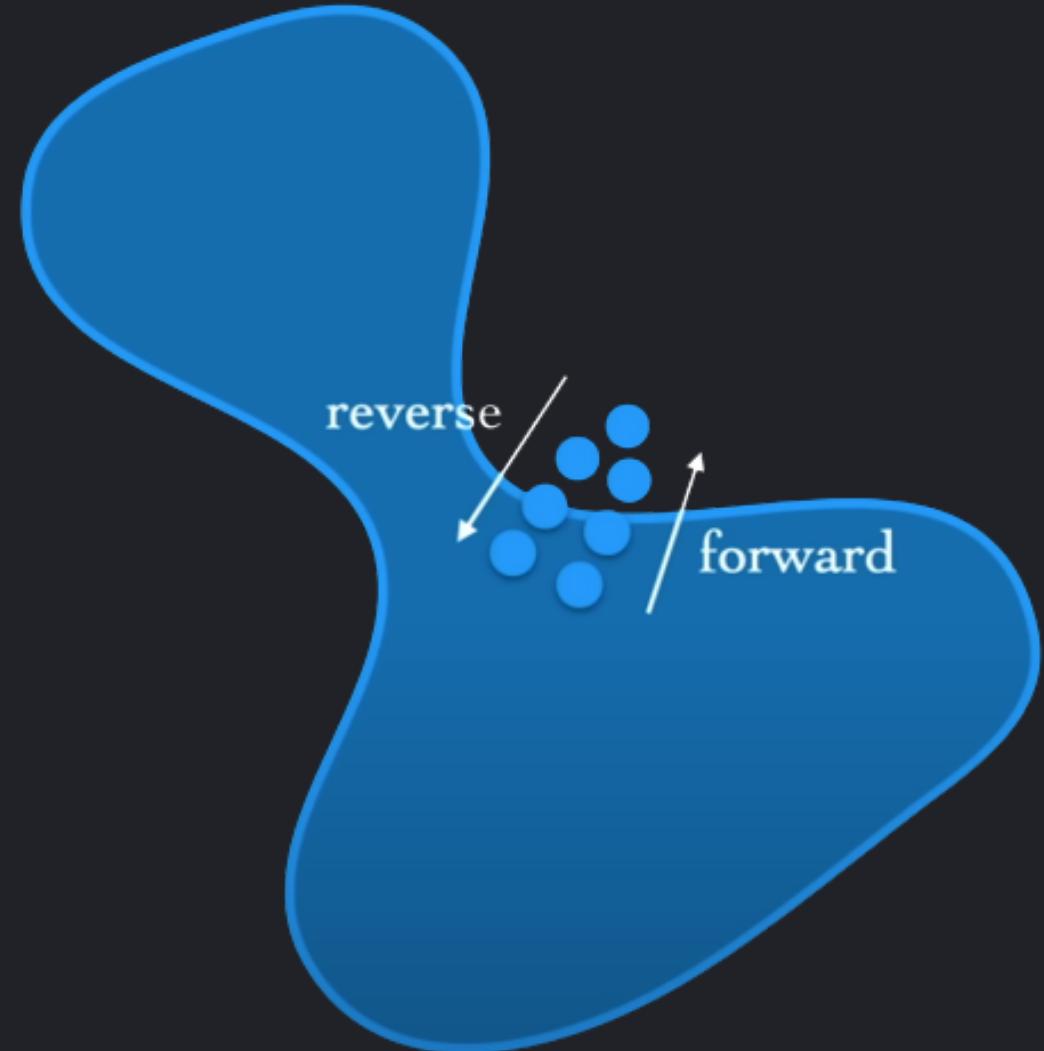


Definition of the Backward Process

- We will model the true reverse process $q(x_{t-1}|x_t)$ with $p_\theta(x_{t-1}|x_t)$.
- $p_\theta(x_T) = \mathcal{N}(x_T; 0, I)$.
- $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$
- $p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)$

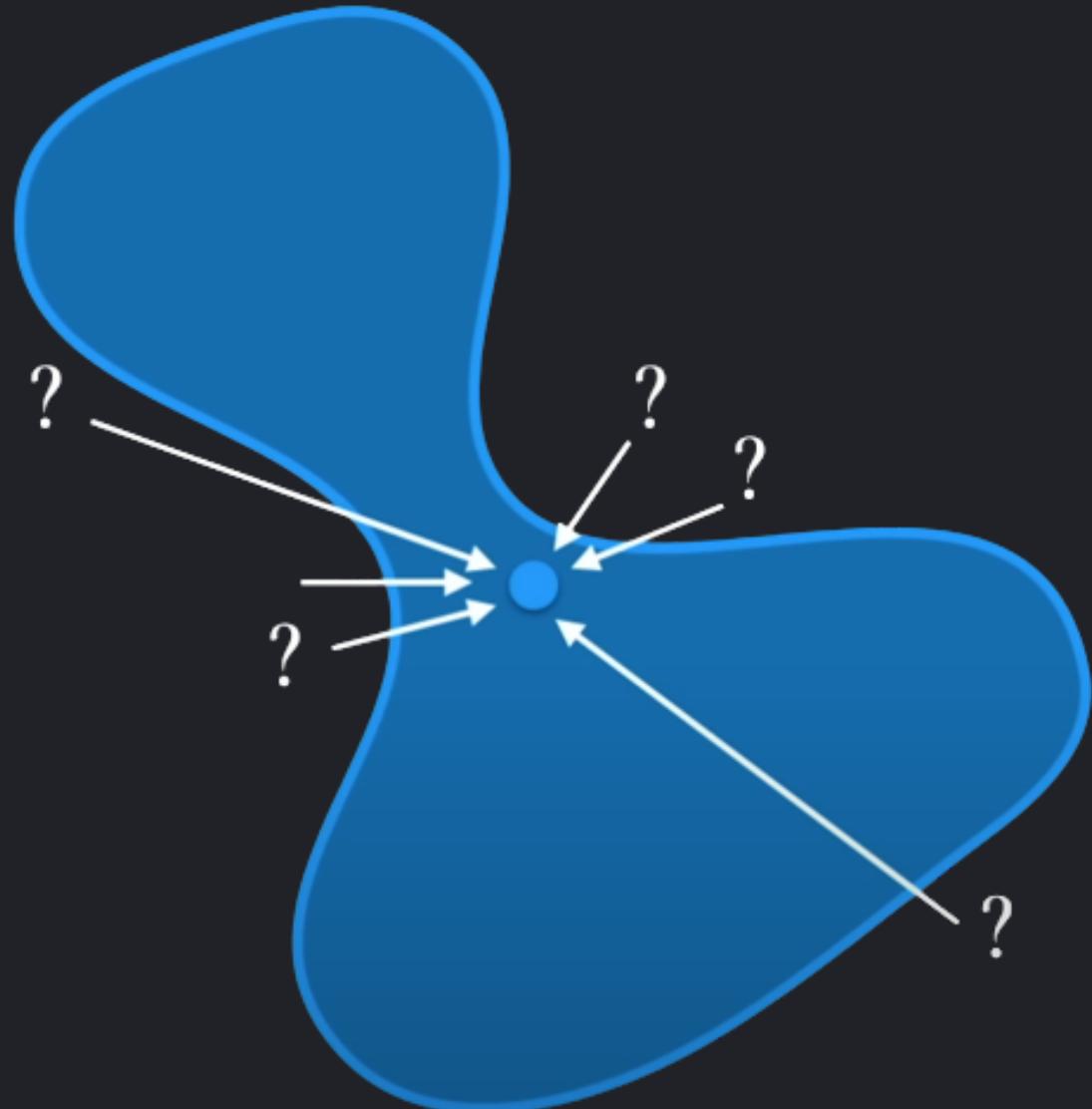
What objective will we optimize?

- Maximize $p_\theta(x_0)$?
- $p_\theta(x_0) = \int p_\theta(x_{0:T})dx_{1:T}$



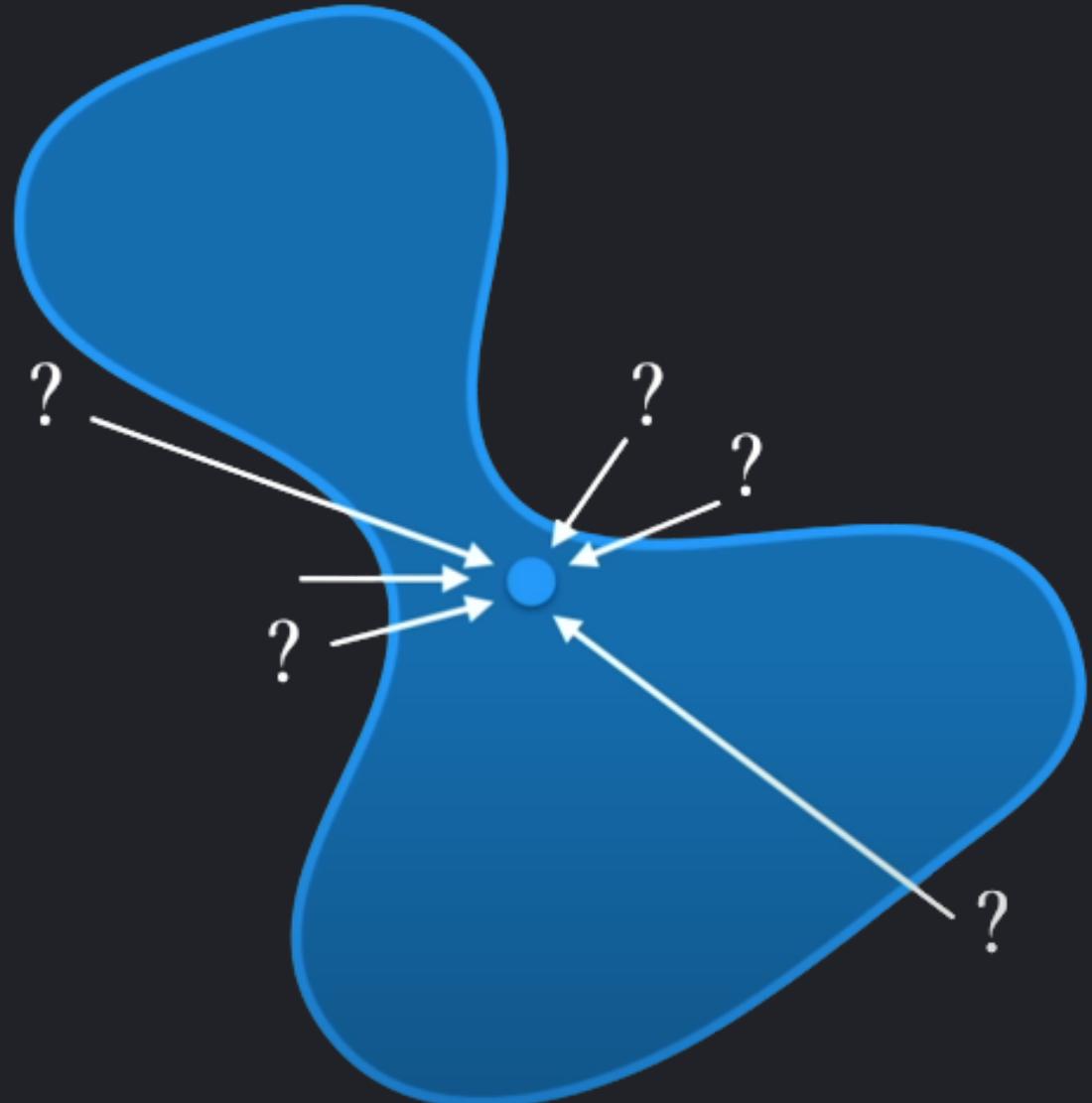
What objective will we optimize?

- Marginalizing over all possible trajectories is intractable.
- $p_\theta(x_0) = \int p_\theta(x_{0:T}) \underline{dx_{1:T}}$

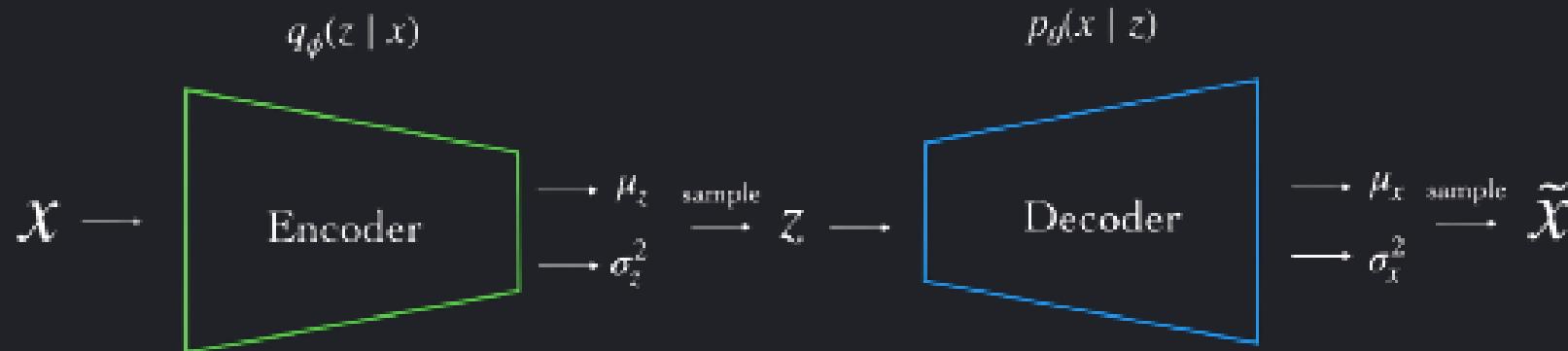


What objective will we optimize?

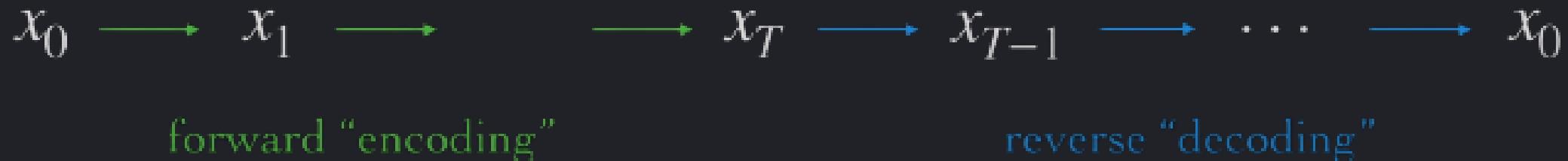
- View x_1, x_2, \dots, x_T as latent variables .
- And x_0 as the observed variable.
- Maximize an Evidence Lower Bound (ELBO)



VAE

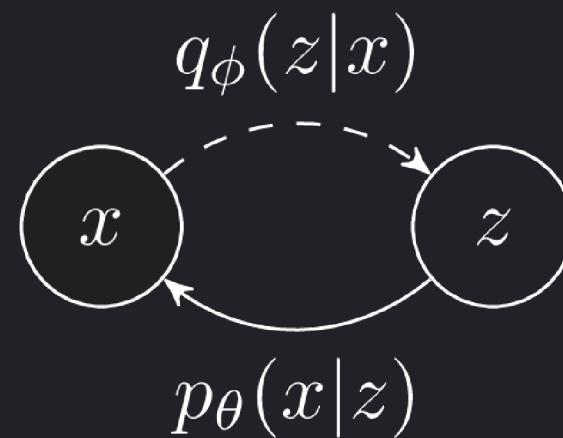


Diffusion model



VAE and ELBO

- $\log p(x) \geq \text{ELBO}$
- Maximizing the ELBO becomes a proxy objective with which to optimize a latent variable model.

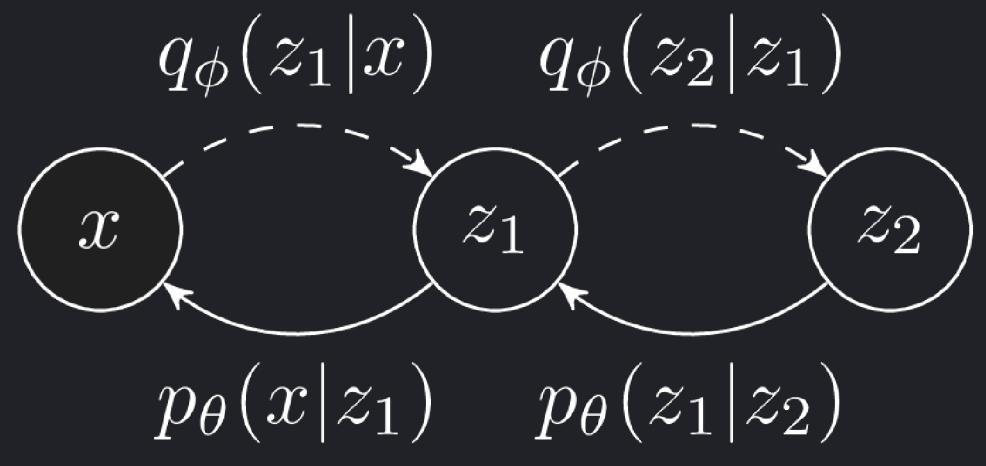


Derivation of ELBO in VAEs

$$\begin{aligned}\log p(x) &= \int \log p(x) q_\phi(z|x) dz && (\int q(z|x) dz = 1) \\&= \mathbb{E}_{q_\phi(z|x)} [\log p(x)] && (\text{By Definition of } \mathbb{E}) \\&= \mathbb{E}_{q_\phi(z|x)} [\log \frac{p(x, z)}{p(z|x)}] && (\text{Chain rule}) \\&= \mathbb{E}_{q_\phi(z|x)} [\log \frac{p(x, z)}{q_\phi(z|x)}] + \mathbb{E}_{q_\phi(z|x)} [\log \frac{q_\phi(z|x)}{p(z|x)}] && (\text{Multiple by } \frac{q_\phi(z|x)}{q_\phi(z|x)} \text{ and split}) \\&= \mathbb{E}_{q_\phi(z|x)} [\log \frac{p(x, z)}{q_\phi(z|x)}] + D_{KL}(q_\phi(z|x) || p(z|x)) && (\text{By definition of } D_{KL}) \\&\geq \underbrace{\mathbb{E}_{q_\phi(z|x)} [\log \frac{p(x, z)}{q_\phi(z|x)}]}_{\text{ELBO}} && (D_{KL} > 0)\end{aligned}$$

Hierarchical VAEs

- VAE with two latent variables
 z_1, z_2
- $\log p(x) \geq \mathbb{E}_{q_\phi(z_1, z_2|x)}[\log \frac{p(x, z_1, z_2)}{q_\phi(z_1, z_2|x)}]$



Diffusion Models as Hierarchical VAEs

Two differences:

- The latent dimension is exactly equal to the data dimension.
- Encoders q are not learned; **pre-defined** as a linear Gaussian model.

An ELBO for the Diffusion Model

$$\begin{aligned}
\log p(x) &\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_{0:T})}{q(x_{1:T}|x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p(x_T) \prod_{t=1}^T p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^T q(x_t|x_{t-1}, \textcolor{red}{x}_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^T \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1}, x_0)} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \prod_{t=2}^T \frac{p_\theta(x_{t-1}|x_t)}{\frac{q(x_{t-1}|x_T, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)}} \right] \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_T)p_\theta(x_0|x_1)}{q(x_1|x_0)} + \log \left(\frac{q(x_1|x_0)}{q(x_T|x_0)} \right) + \log \prod_{t=2}^T \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1}, x_0)} \right] \\
&= \mathbb{E}_{q(x_1|x_0)} [\log p_\theta(x_0|x_1)] + \mathbb{E}_{q(x_T|x_0)} \left[\log \frac{p_\theta(x_T)}{q(x_T|x_0)} \right] + \sum_{t=2}^T \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1}, x_0)}
\end{aligned}$$

(Due to Markovian Property $q(x_t|x_{t-1}) = q(x_t|x_{t-1}, x_0)$)

(1st term out of product and split.)

(Bayes rule)

(Only the first and last term survive.)

(Get log inside product)

An ELBO for the Diffusion Model

$$\begin{aligned} \log p(x) &\geq \mathbb{E}_{q(x_1|x_0)} [\log p_\theta(x_0|x_1)] & (L_0 : \text{Reconstruction term}) \\ &- D_{KL}(q(x_T|x_0)||p(x_T)) & (L_T : \text{Prior matching term}) \\ &- \sum_{t=2}^T \mathbb{E}_{q(x_t|x_0)} [D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))] & (L_{t-1} : \text{Denoising matching term}) \end{aligned}$$

- L_0 : Can be approximated and optimized using a Monte Carlo estimate.
- L_T : Equal to zero under our assumptions since $q(x_T|x_0) \approx p(x_T) = \mathcal{N}(0, I)$
- L_{t-1} : Denoising transition step $p_\theta(x_{t-1}|x_t)$ is learned as an approximation to tractable, ground-truth denoising transition step $q(x_{t-1}|x_t, x_0)$.

What objective will we optimize?

- In this derivation the bulk of the cost lies in L_{t-1} .
- If we find a closed form for $q(x_{t-1}|x_t, x_0)$ we can compute the KL Divergence and maximize the ELBO.
- By Bayes rule we have $q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)}$.
- We already know that $q(x_t|x_{t-1}, x_0) = q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{a_t}x_t, (1 - a_t)I)$
- What about $q(x_t|x_0)$ and $q(x_{t-1}|x_0)$?

Reparameterization trick

Under the reparameterization trick, samples $x_t \sim q(x_t|x_{t-1})$ can be rewritten as:

$$x_t = \sqrt{a_t}x_{t-1} + \sqrt{1 - a_t}e \quad \text{with } e \sim \mathcal{N}(0, 1)$$

Thus, $q(x_t|x_0)$ can be recursively derived through repeated applications of the reparameterization trick. Assuming access to $2T$ random variables $\{e_t^*\}_{t=0}^T \sim \mathcal{N}(0, 1)$.

$$\begin{aligned} x_t &= \sqrt{a_t}x_{t-1} + \sqrt{1 - a_t}e_{t-1}^* \\ &= \sqrt{a_t} \left(\sqrt{a_{t-1}}x_{t-2} + \sqrt{1 - a_{t-1}}e_{t-2}^* \right) + \sqrt{1 - a_t}e_{t-1}^* \\ &= \sqrt{a_t a_{t-1}}x_{t-2} + \sqrt{a_t - a_t a_{t-1}}e_{t-2}^* + \sqrt{1 - a_t}e_{t-1}^* \\ &= \sqrt{a_t a_{t-1}}x_{t-2} + \sqrt{1 - a_t a_{t-1}}e_{t-2} \quad (\text{Sum of Gaussians}) \\ &= \dots \\ &= \sqrt{\bar{a}_t}x_0 + \sqrt{1 - \bar{a}_t}e_0 \quad (\bar{a}_t = \prod_{i=1}^t a_i) \end{aligned}$$

Now we can calculate the form of $q(x_{t-1}|x_t, x_0) \dots$

$$\begin{aligned} q(x_{t-1}|x_t, x_0) &= \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \\ &= \frac{\mathcal{N}(x_t; \sqrt{a_t}x_t, (1-a_t)I) \quad \mathcal{N}(x_{t-1}; \sqrt{\bar{a}_{t-1}}x_0, (1-\bar{a}_{t-1})I)}{\mathcal{N}(x_t; \sqrt{\bar{a}_t}x_0, (1-\bar{a}_t)I)} \\ &= \dots \\ &= \mathcal{N}(x_{t-1}; \underbrace{\frac{\sqrt{a_t}(1-\bar{a}_{t-1})x_t + \sqrt{\bar{a}_{t-1}}(1-a_t)x_0}{1-\bar{a}_t}}_{\mu_q(x_t, x_0)}, \underbrace{\frac{(1-a_t)(1-\bar{a}_{t-1})}{(1-\bar{a}_t)}I}_{\sum_q(t)}) \end{aligned}$$

... and maximize the ELBO by minimizing the D_{KL}

$$\begin{aligned} & \operatorname{argmin}_{\theta} D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t)) \\ &= \operatorname{argmin}_{\theta} D_{KL}(\mathcal{N}(x_{t-1}; \mu_q(t), \Sigma_q(t)) || \mathcal{N}(x_{t-1}; \mu_{\theta}(t), \Sigma_q(t))) \quad (\text{set denoising transition variance to be } \Sigma_q(t)) \\ &= \dots \quad (\text{KL Divergence Gaussians}) \\ &= \operatorname{argmin}_{\theta} \frac{1}{2\sigma_q^2(t)} [\|\mu_{\theta} - \mu_q\|_2^2] \end{aligned}$$

We want to optimize $\mu_{\theta}(x_t, t)$ to matches $\mu_q(x_t, x_0)$.

... and maximize the ELBO by minimizing the D_{KL}

We can match μ_θ and μ_q as close as possible:

$$\begin{aligned}\mu_q(x_t, x_0) &= \frac{\sqrt{a_t}(1-\bar{a}_{t-1})x_t + \sqrt{\bar{a}_{t-1}}(1-a_t)\textcolor{green}{x}_0}{1-\bar{a}_t}, \\ \mu_\theta(x_t, t) &= \frac{\sqrt{a_t}(1-\bar{a}_{t-1})x_t + \sqrt{\bar{a}_{t-1}}(1-a_t)\textcolor{green}{x}_\theta(\mathbf{x}_t, \mathbf{t})}{1-\bar{a}_t}\end{aligned}$$

$x_\theta(x_t, t)$ is parameterized by a neural network that seeks to predict x_0 from noisy image x_t and time index t . So finally we can write:

$$\operatorname{argmin}_\theta = \frac{\bar{a}_{t-1}(1-a_t)^2}{2\sigma_q^2(t)(1-\bar{a}_t)^2} [||x_\theta(x_t, t) - x_0||_2^2]$$

Simplified Loss

We can choose the parameterization:

$$x_0 = \frac{x_t = \sqrt{1 - \bar{a}_t} \epsilon_0}{\sqrt{\bar{a}_t}}$$

μ_θ and μ_q become:

$$\mu_q(x_t, x_0) = \frac{1}{\sqrt{a_t}} x_t - \frac{1 - a_t}{\sqrt{1 - \bar{a}_t} \sqrt{a_t}} \epsilon_0$$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{a_t}} x_t - \frac{1 - a_t}{\sqrt{1 - \bar{a}_t} \sqrt{a_t}} \epsilon_\theta(x_t, t)$$

Reformulate the loss to:

$$\operatorname{argmin}_\theta = \underbrace{\frac{(1 - a_t)^2}{2\sigma_q^2(t)(1 - \bar{a}_t)a_t}}_{\lambda_t} [||e_0 - e_\theta(x_t, t)||_2^2]$$

- However, λ_t is often very large for small t's.
- Discard λ_t and minimize a weighted version of the ELBO.

Training	Sampling
<p>1: repeat</p> <p>2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$</p> <p>3: $t \sim \text{Uniform}(\{1, \dots, T\})$</p> <p>4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$</p> <p>5: Take gradient descent step on $\nabla_{\theta} \left\ \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta} \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t \right) \right\ ^2$</p> <p>6: until converged</p>	<p>1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$</p> <p>2: for $t = T, \dots, 1$ do</p> <p>3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$</p> <p>4: $\mathbf{x}_{t,1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$</p> <p>5: end for</p> <p>6: return \mathbf{x}_0</p>

DDPM: Implementation details

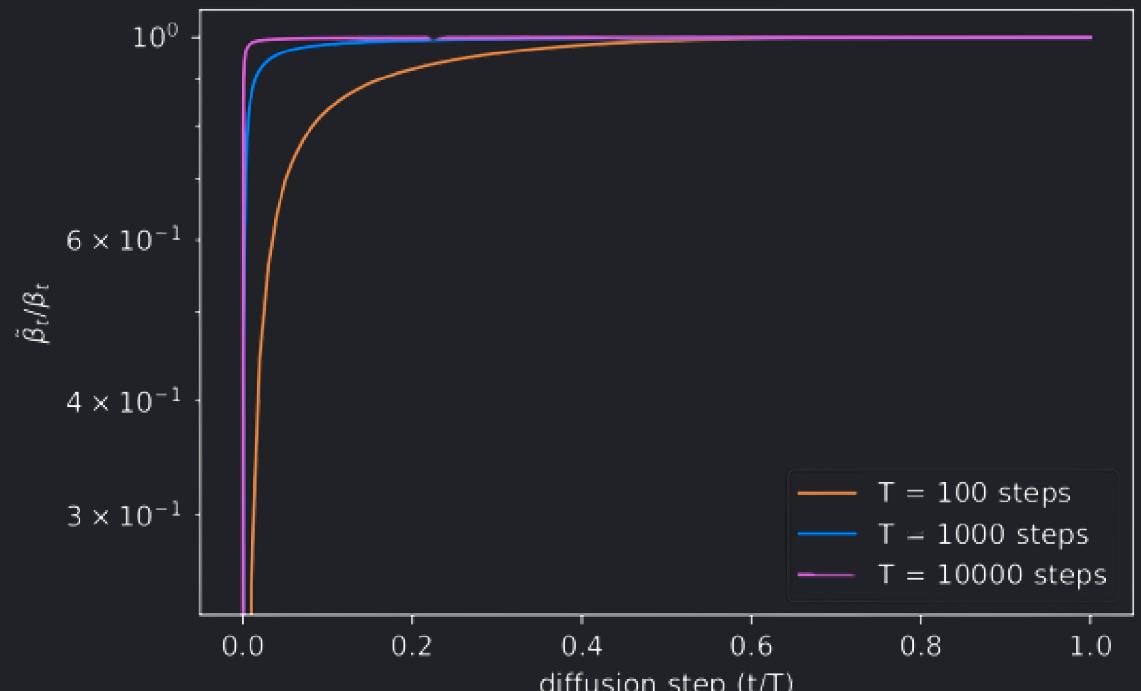
- A U-Net is used to estimate ϵ_θ
- Time information added to U-Net with positional embeddings
- Linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$, with $T = 1000$

DDPM: Predicting μ vs predicting ϵ

Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L , learned diagonal Σ	7.28 ± 0.10	23.69
L , fixed isotropic Σ	8.06 ± 0.09	13.22
$\ \tilde{\mu} - \tilde{\mu}_\theta\ ^2$	—	—
ϵ prediction (ours)		
L , learned diagonal Σ	—	—
L , fixed isotropic Σ	7.67 ± 0.13	13.51
$\ \tilde{\epsilon} - \epsilon_\theta\ ^2$ (L_{simple})	9.46 ± 0.11	3.17

Learning $\Sigma_\theta(x_t, t)$

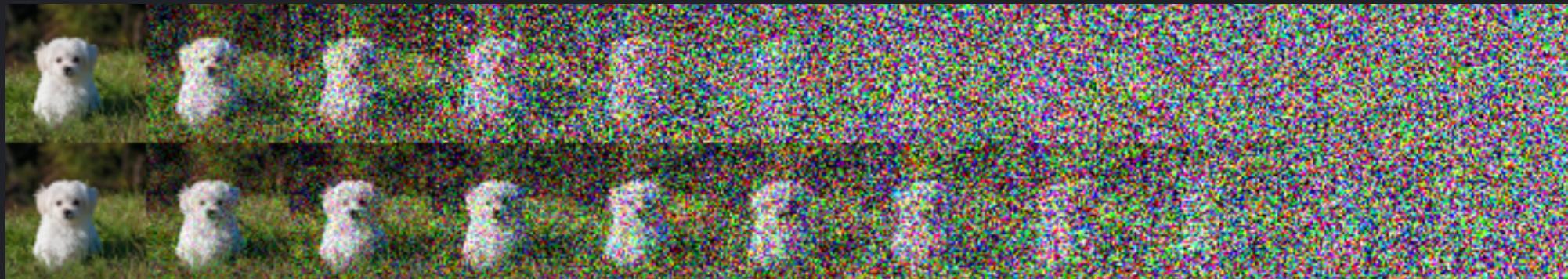
- In DDPM $\Sigma_\theta(x_t, t) = \Sigma(t) = \sigma_t^2 I$
 - with $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \bar{\beta}_t := \frac{1-\bar{a}_{t-1}}{1-\bar{a}_t} \beta_t$
 - β_t and $\bar{\beta}_t$ yield similar results.
- β_t and $\bar{\beta}_t$ almost equal except near $t = 0$
- Learn to interpolate β_t and $\bar{\beta}_t$
 - $\sigma_t^2 = \exp(v \log \beta_t + (1 - v) \log \bar{\beta}_t)$



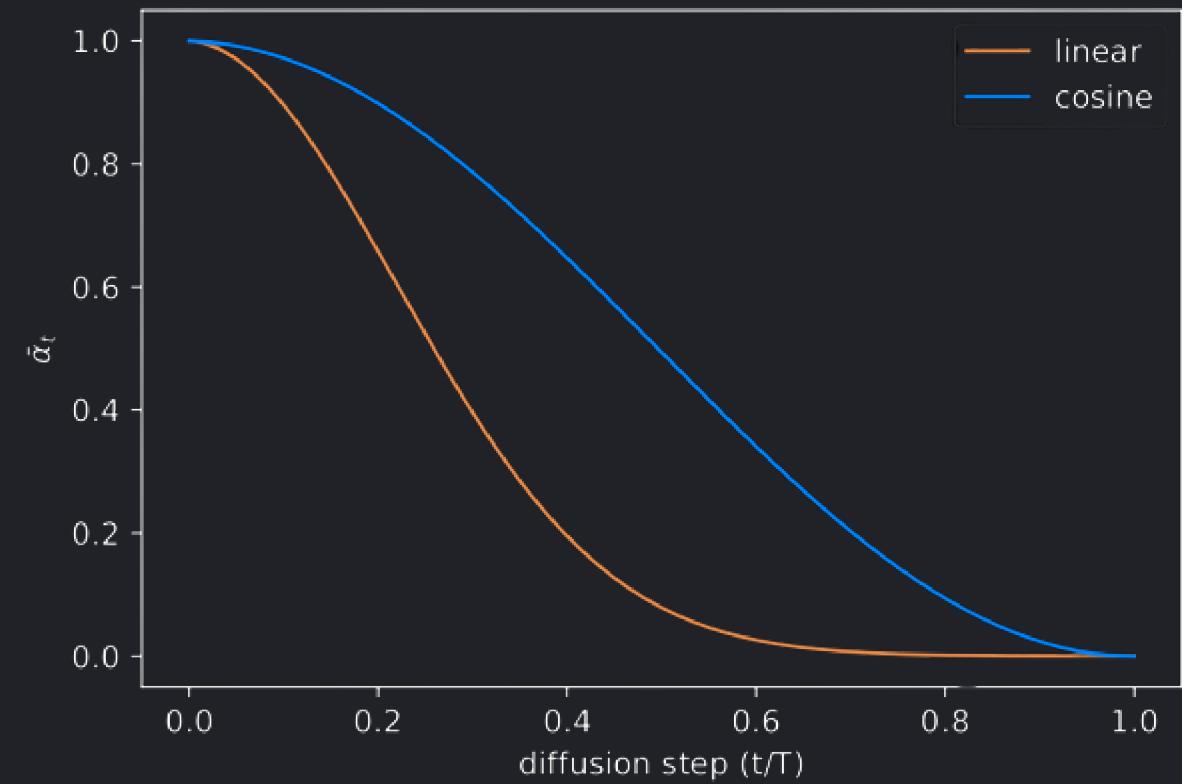
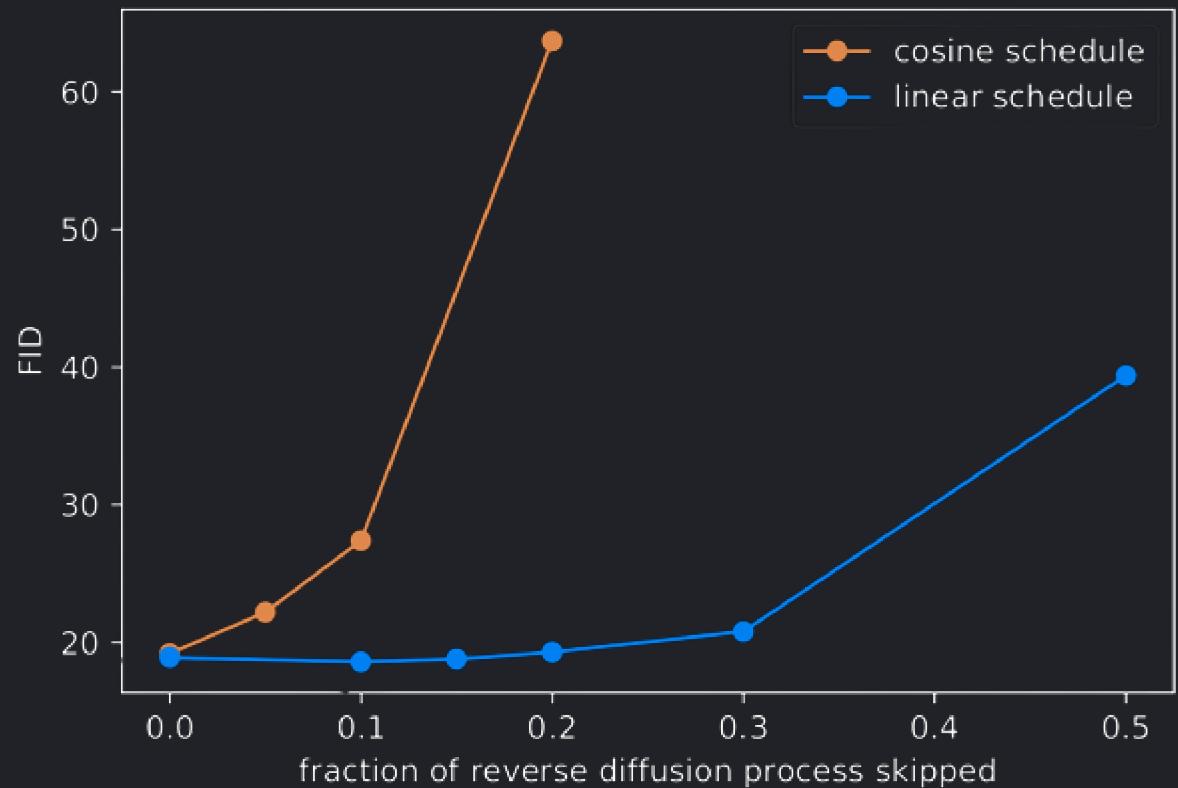
Noise Schedule

- Problems with linear schedule.
 - End of the forward noising process is too noisy.
 - Doesn't contribute very much to sample quality
- Cosine schedule:

- $\bar{a}_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos^2\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)$

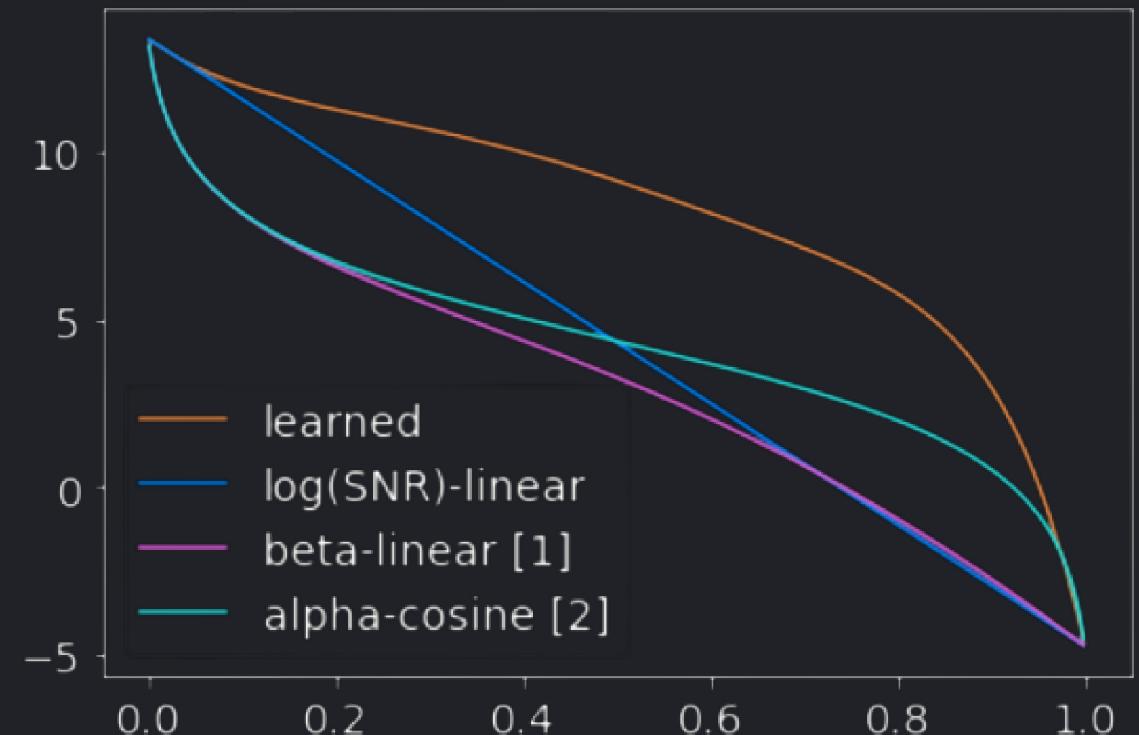


Noise Schedule: Cosine vs Linear



Learning the Noise Schedule

- Recall $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{a}_t}x_0, (1 - \bar{a}_t)I)$
- Define $\text{SNR} = \frac{\mu^2}{\sigma^2} = \frac{\bar{a}_t}{1 - \bar{a}_t}$
- Model SNR with a NN:
 - $\text{SNR}(t) = \exp(-\omega_\phi(t))$
 - $\bar{a}_t = \text{sigmoid}(-\omega_\phi(t))$



(a) log SNR vs time t

Latent Diffusion Models

- Run the diffusion in a latent space.
- Given an image $x \in \mathbb{R}^{H \times W \times 3}$
- Use an encoder \mathcal{E} and extract $z = \mathcal{E}(x)$
- Use a decoder \mathcal{D} and retrieve $\hat{x} = \mathcal{D}(z)$
- $z \in \mathbb{R}^{h \times w \times c}$, with sampling rate $f = H/h = W/w = 2^m$, $m \in \mathbb{N}$.

