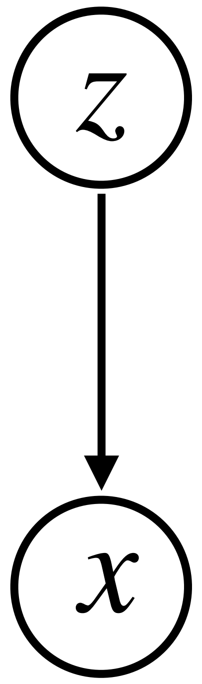
6.7900 Machine Learning (Fall 2024)

Lecture 22: generative models — diffusion

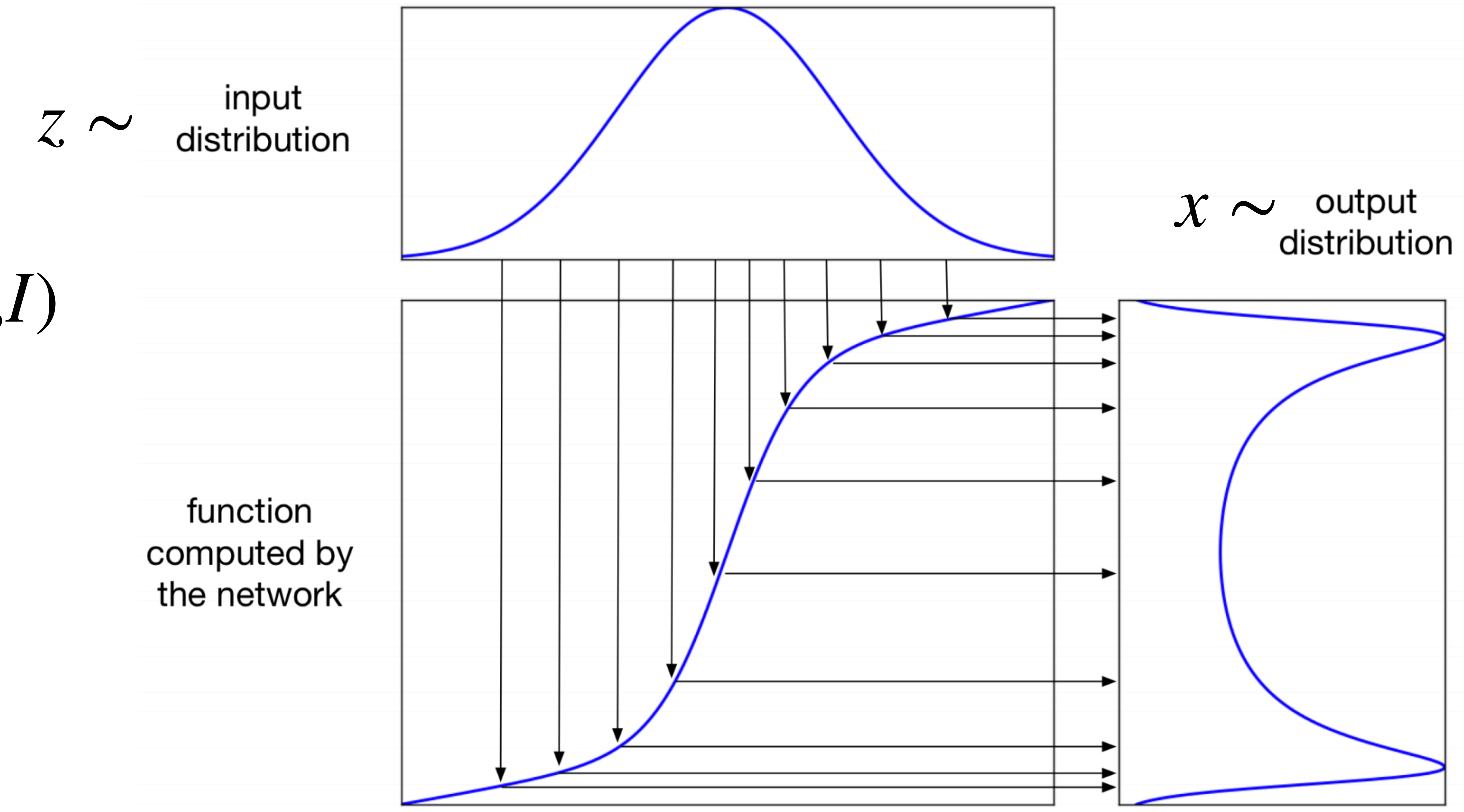
Preface: from simple samples to complex objects

- We can realize complex distributions over (e.g.) images by drawing samples from a fixed simple distribution and then mapping such samples through a complicated function (a neural network)



$$P(z) = N(z \mid 0, I)$$

$$P(x \mid z, \theta)$$



Preface: latents and how they are used

- The latent "degrees of freedom" z are useful if they are strongly coupled with x values; specifically, they should help us realize x's more easily conditionally on z
- In VAEs, we have an encoder and a decoder and both models represent this coupling between x and z.

- The ELBO criterion adjusts these models so they agree as distributions

Preface: latents and how they are used

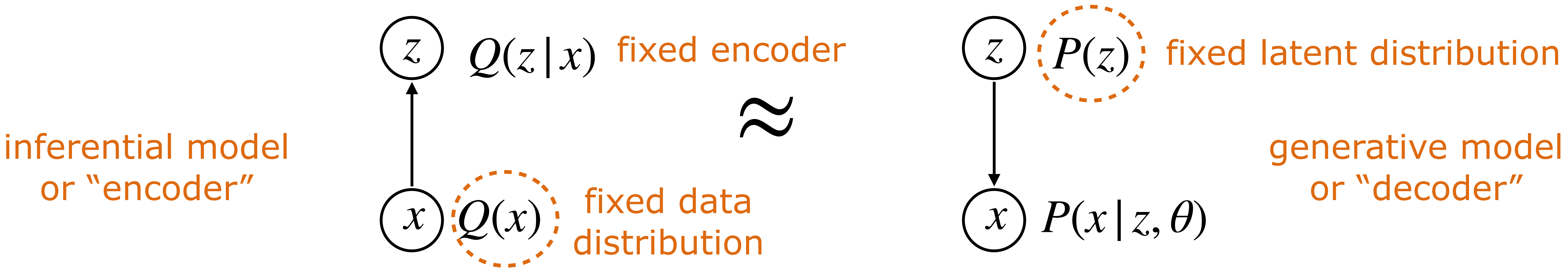
- The latent “degrees of freedom” z are useful if they are strongly coupled with x values; specifically, they should help us realize x’s more easily conditionally on z
- In VAEs, we have an encoder and a decoder and both models represent this coupling between x and z.

- The ELBO criterion adjusts these models so they agree as distributions, including marginals

$$\int Q_D(x)Q(z|x,\phi)dx \approx P(z) \qquad \int P(z)P(x|z,\theta)dz \approx Q_D(x)$$

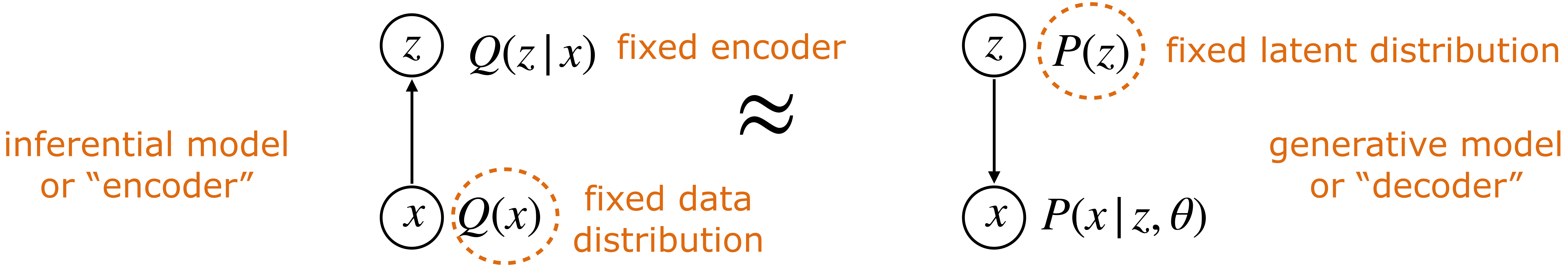
Preface: latents and how they are used

- We can actually always just **fix the encoder** and only learn the decoder



Preface: latents and how they are used

- We can actually always just **fix the encoder** and only learn the decoder

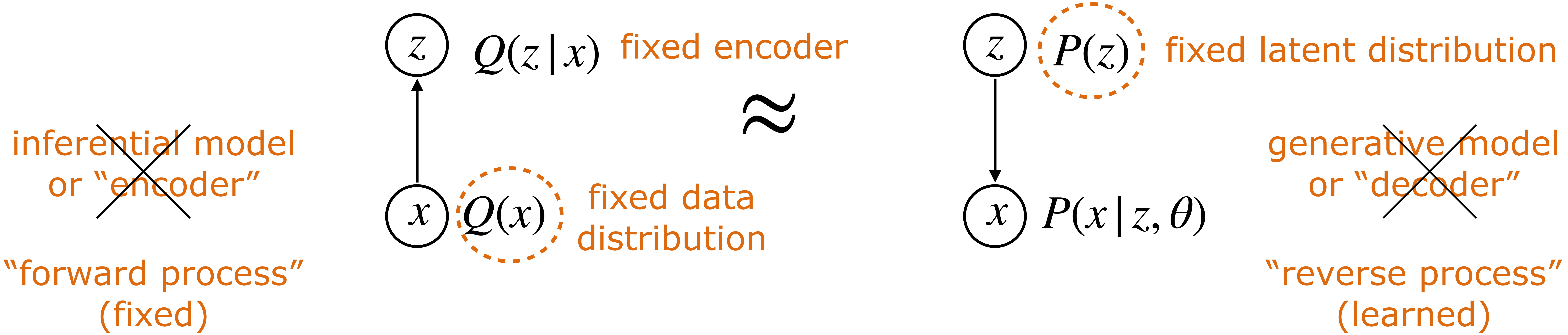


- The fixed encoder should still agree with the decoder’s fixed latent marginal (otherwise they’d be permanently inconsistent)

$$\int Q_D(x)Q(z|x)dx \approx P(z)$$

Preface: latents and how they are used

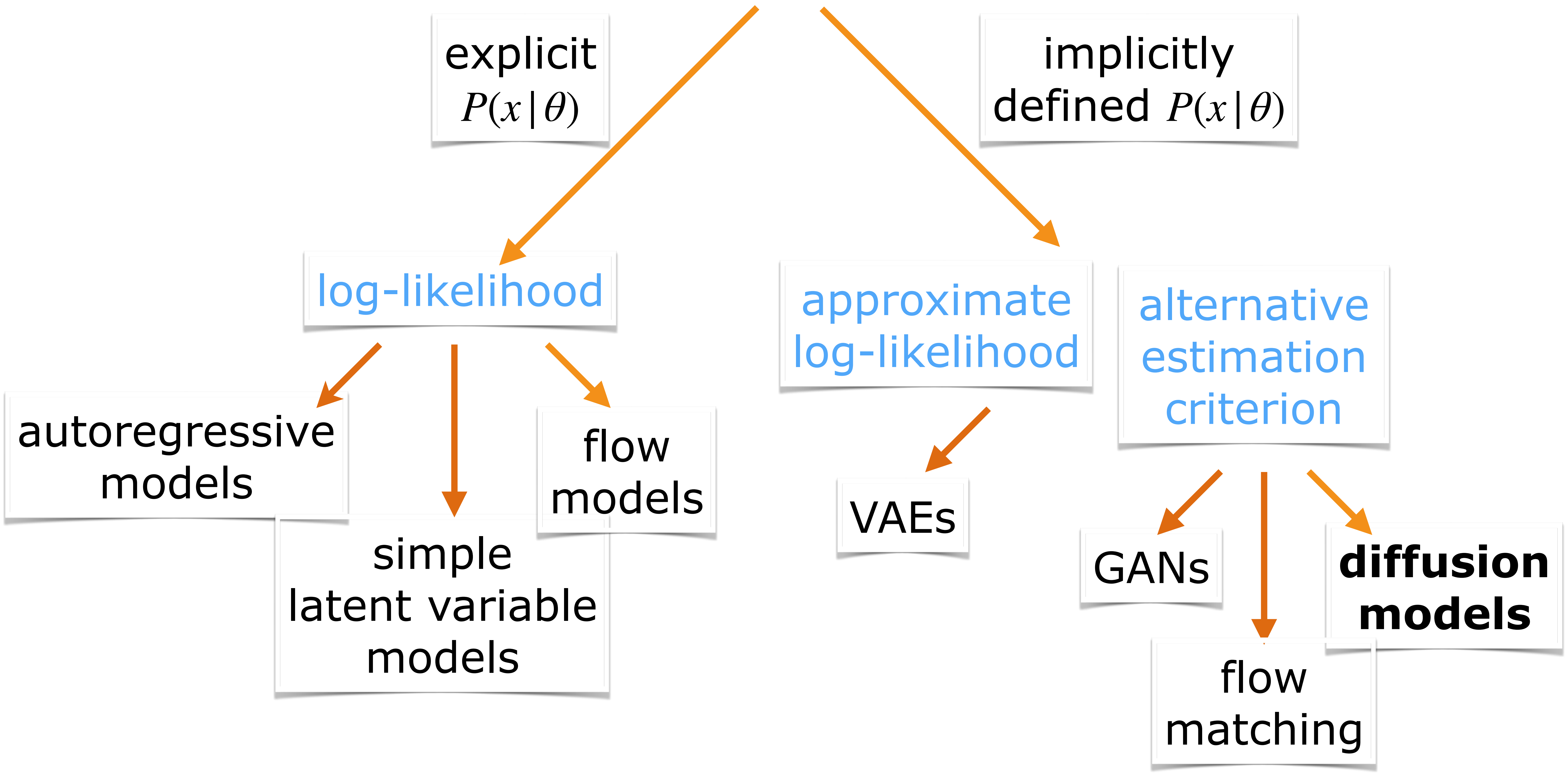
- We can actually always just **fix the encoder** and only learn the decoder



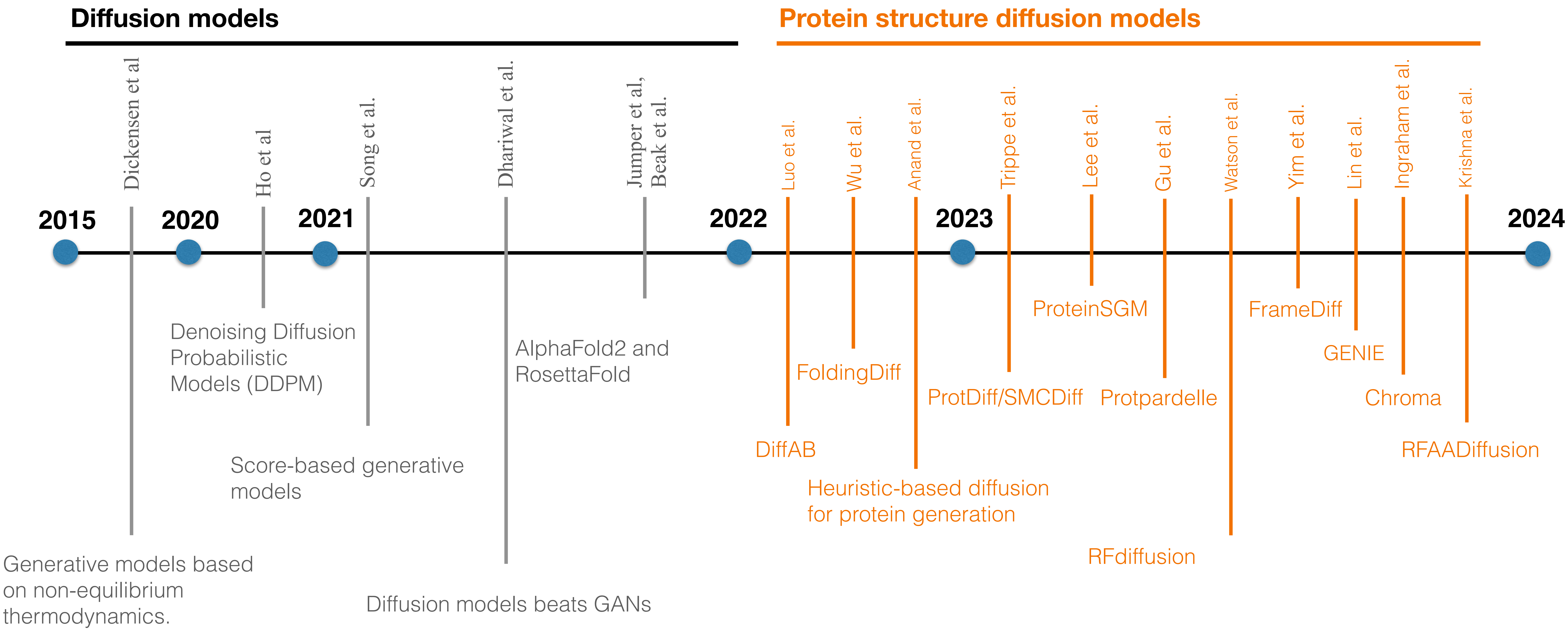
- The fixed encoder should still agree with the decoder's fixed latent marginal (otherwise they'd be permanently inconsistent)

$$\int Q_D(x)Q(z|x)dx \approx P(z)$$

A slice of the generative “landscape”



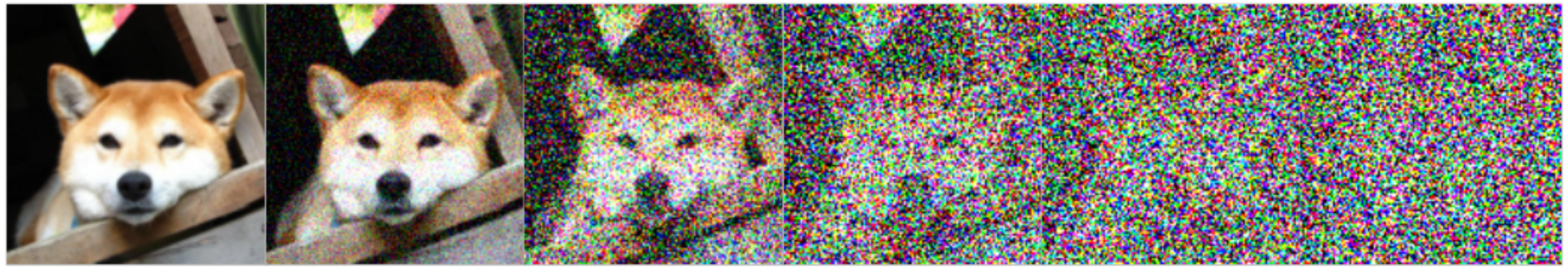
Ex: rapid adoption of diffusion models



[Yim et al 2024, "Diffusion models in protein structure and docking"]

Diffusion models

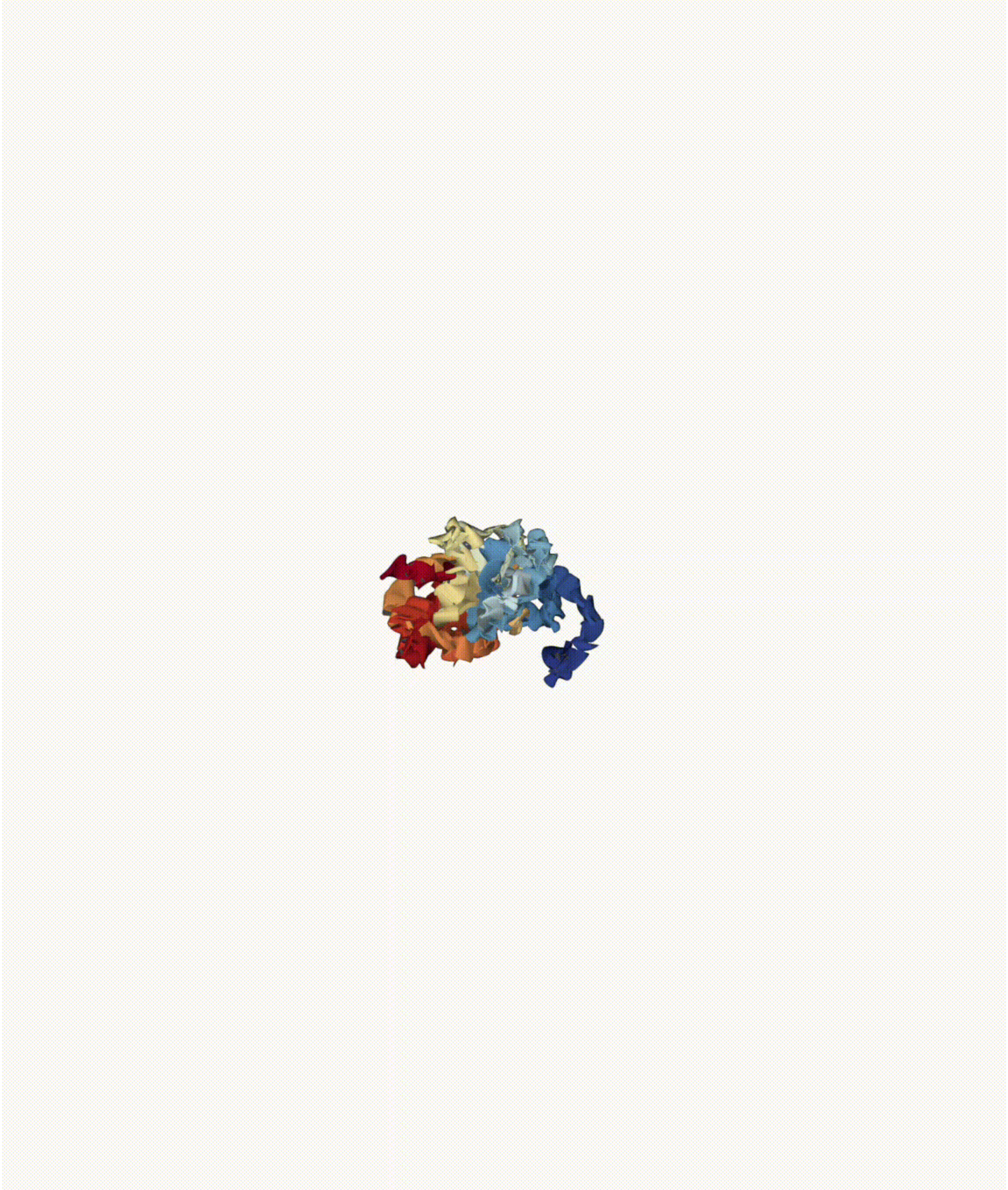
- De-noising diffusion models over images (e.g., Ho et al., Song et al.)



[image from
Rissanen et al 2022]

$x \longleftarrow z \sim N(0,I)$

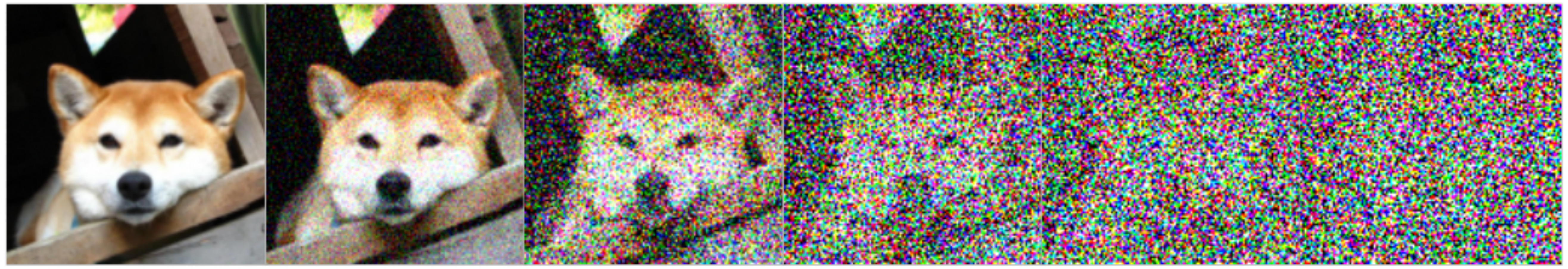
- 3D structures of molecules (e.g., proteins)



- Discrete diffusion for language, sequence modeling, etc.

Diffusion motivation

- De-noising diffusion models over images (e.g., Ho et al., Song et al.)



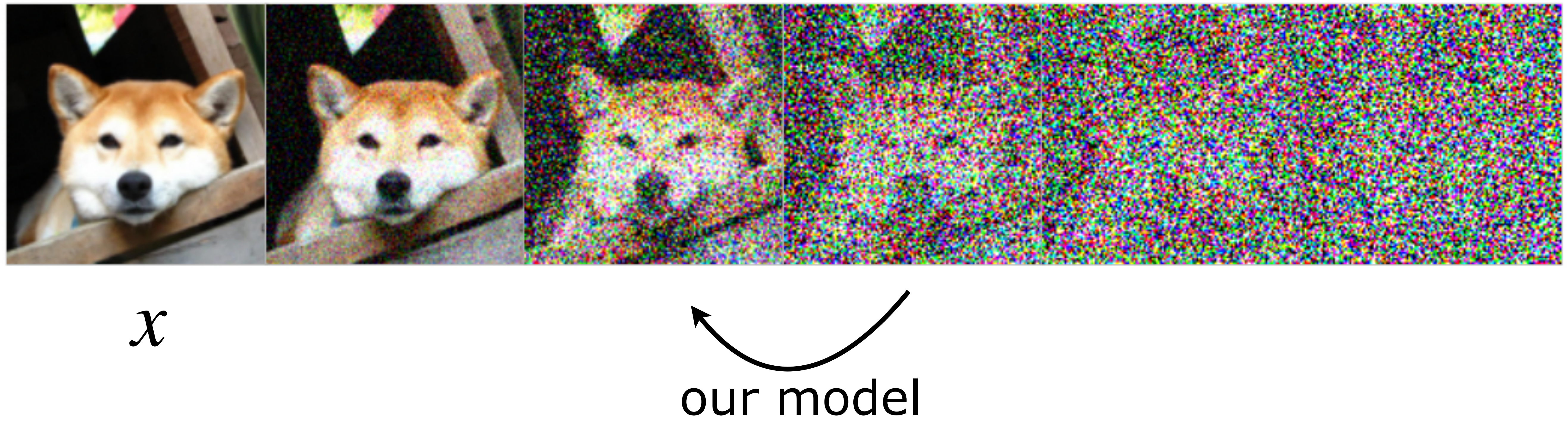
x

[image from Rissanen et al 2022]

- It's really helpful to give a generative model a stack of noisy versions of the same image… different features are present at different noise levels

Diffusion motivation

- De-noising diffusion models over images (e.g., Ho et al., Song et al.)

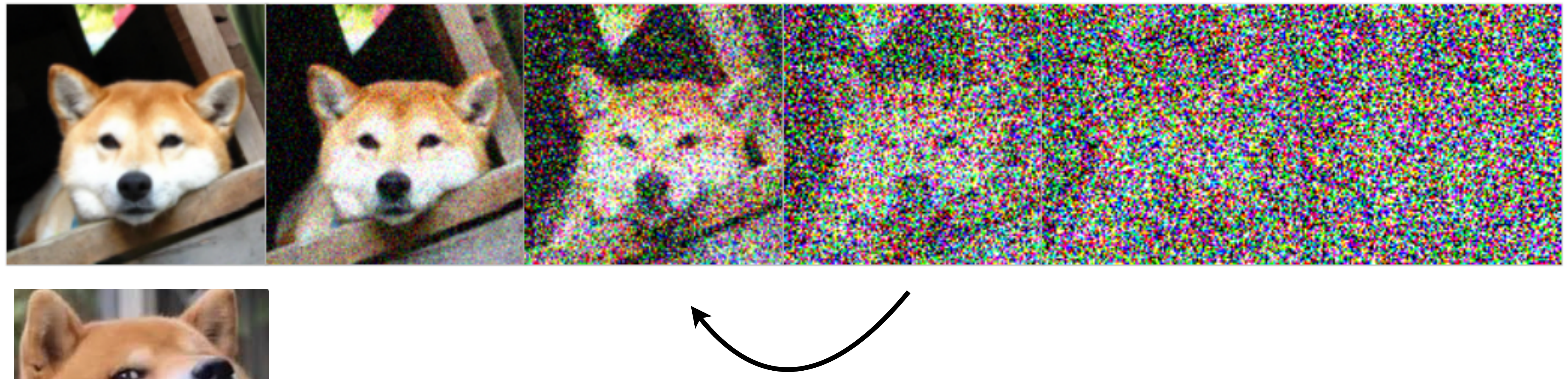


[image from Rissanen et al 2022]

- It's really helpful to give a generative model a stack of noisy versions of the same image… different features are present at different noise levels
- The goal is then to learn to successively uncover these lost features (de-noise images back into cleaner instances)

…Diffusion motivation

- De-noising diffusi els over images (e.g., Ho et al., Song et al.)

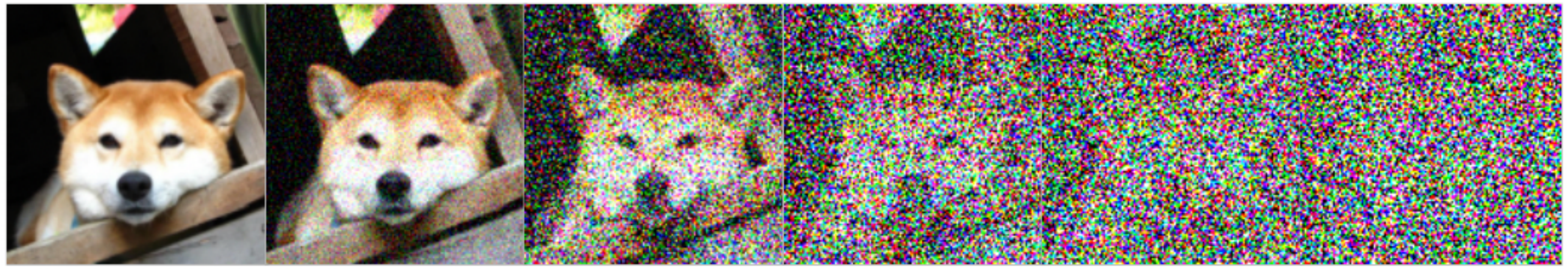


[image from Rissanen et al 2022]

- It's really helpful t a generative model a stack of noisy versions of the same image… differ … t features are present at different noise levels
- The goal is then to learn to successively uncover these features (de-noise images back into cleaner instances)
- This is hard since there are potentially many images that could have similar noisy versions

Diffusion motivation

- De-noising diffusion models over images (e.g., Ho et al., Song et al.)



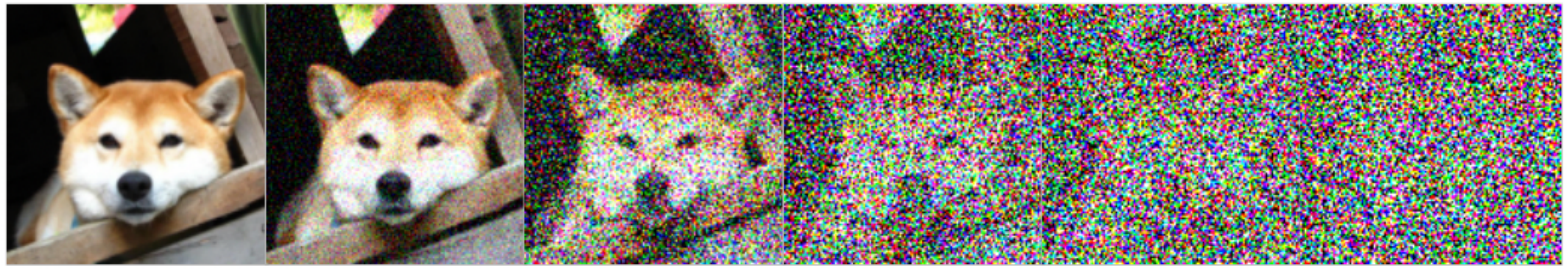
[image from Rissanen et al 2022]

$$x \longleftarrow z \sim N(0,I)$$

- It's really helpful to give a generative model a stack of noisy versions of the same image… different features are present at different noise levels
- The goal is then to learn to successively uncover these features (de-noise images back into cleaner instances)
- This is hard since there are potentially many images that could have similar noisy versions
- Once we have our de-noising model, we can draw a noisy sample, and iterate to generate a new clean instance (works the same with other types of objects, e.g., molecule structures)

Mechanisms of generation: diffusion

- De-noising diffusion models over images (e.g., Ho et al., Song et al.)



[image from Rissanen et al 2022]

$$x_0 \longrightarrow z \sim N(0,I)$$

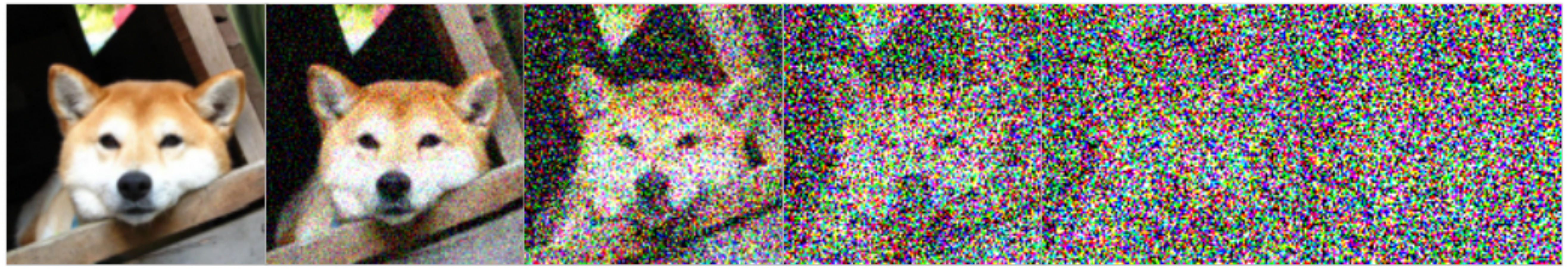
- A <u>forward process</u> “simplifies” objects (images) by adding more and more noise (each noise addition removes high frequency features from the image)

$$x_0 \xrightarrow{q(x_1|x_0)} x_1 \xrightarrow{q(x_2|x_1)} x_2 \longrightarrow \cdots \quad x_{T-1} \xrightarrow{q(x_T|x_{T-1})} x_T$$

simple noise

Mechanisms of generation: diffusion

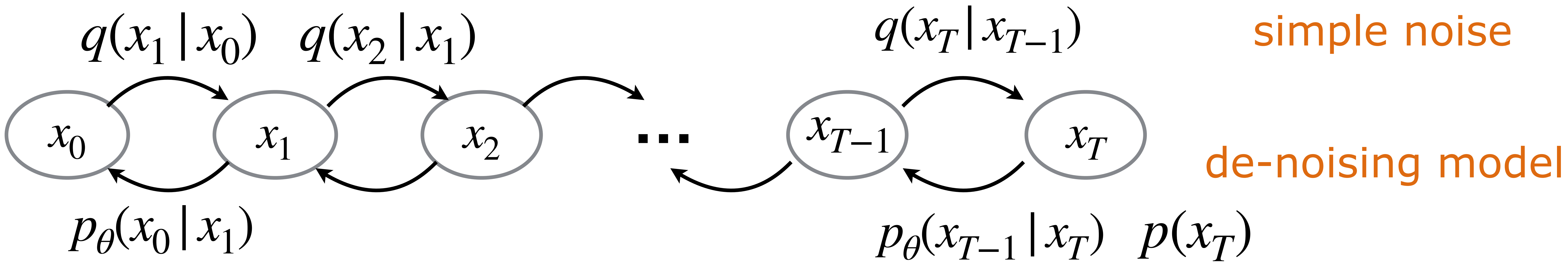
- De-noising diffusion models over images (e.g., Ho et al., Song et al.)



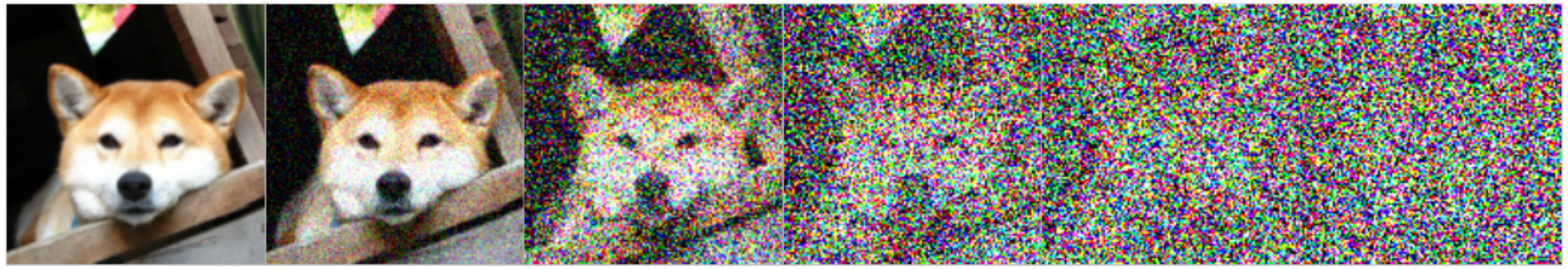
[image from Rissanen et al 2022]

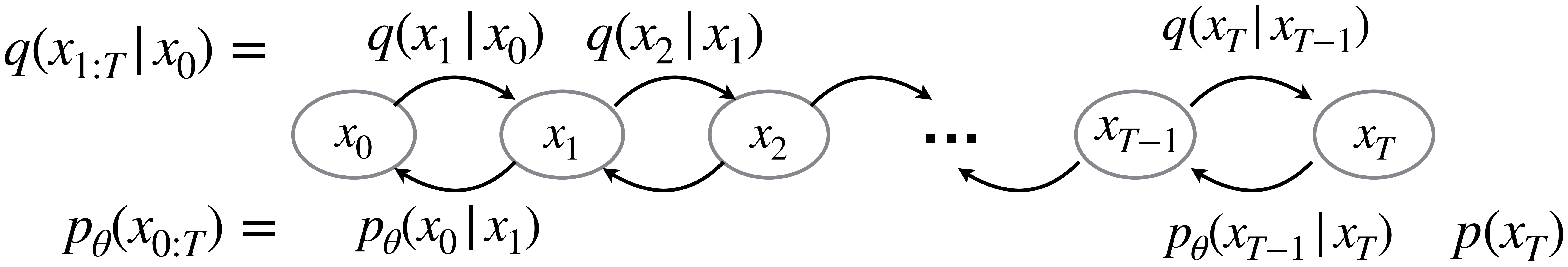
$$x_0 \longleftarrow z \sim N(0,I)$$

- A <u>forward process</u> “simplifies” objects (images) by adding more and more noise (each noise addition removes high frequency features from the image)
- The <u>reverse process</u> tries to reverse each step, i.e., it learns to predict how to de-noise images back into high quality versions



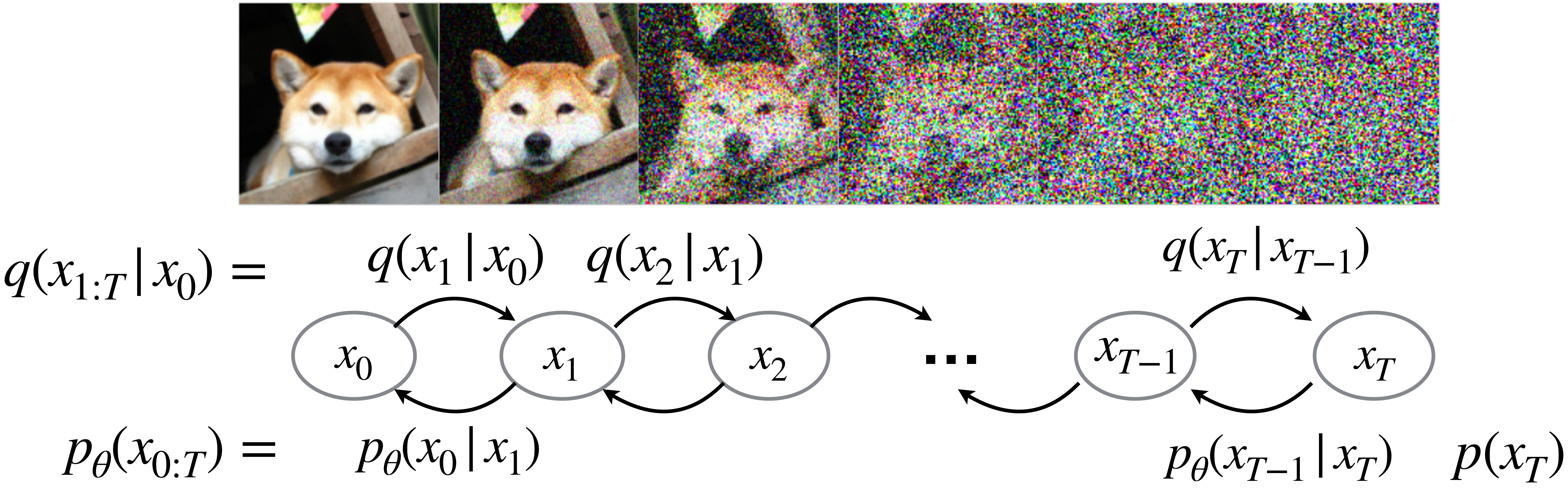
Basic diffusion models





- We have a *fixed* (not learned) "forward process" $q(x_t|x_{t-1})$ which adds Gaussian noise (+ shrinkage)
- We use a reverse de-noising process $p_\theta(x_{t-1}|x_t)$ to (statistically) invert those steps
- This is a latent variable model where z = latent trajectories $(x_1, \ldots, x_T)$

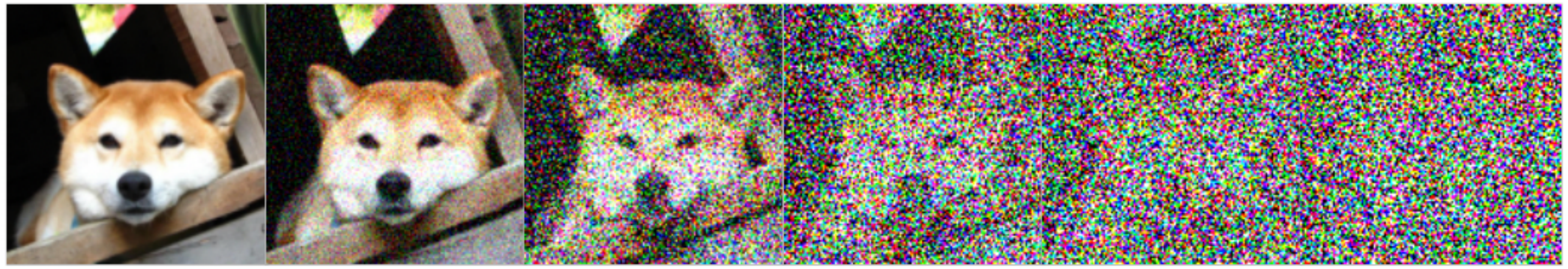
Basic diffusion models



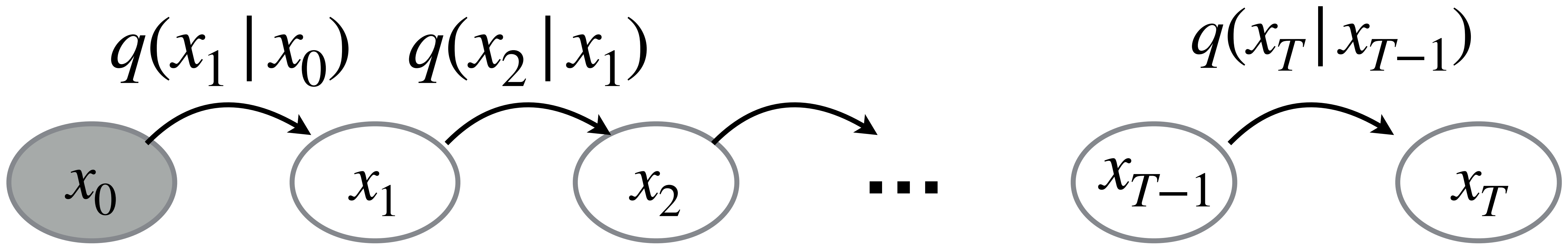
- We have a *fixed* (not learned) “forward process” $q(x_t|x_{t-1})$ which adds Gaussian noise (+ shrinkage)
- We use a reverse de-noising process $p_\theta(x_{t-1}|x_t)$ to (statistically) invert those steps
- This is a latent variable model where z = latent trajectories $(x_1, \ldots, x_T)$
- In principle, we can learn the reverse de-noising process by maximizing the ELBO criterion

$$\log p_\theta(x_0) \geq E_q\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$$

Forward process



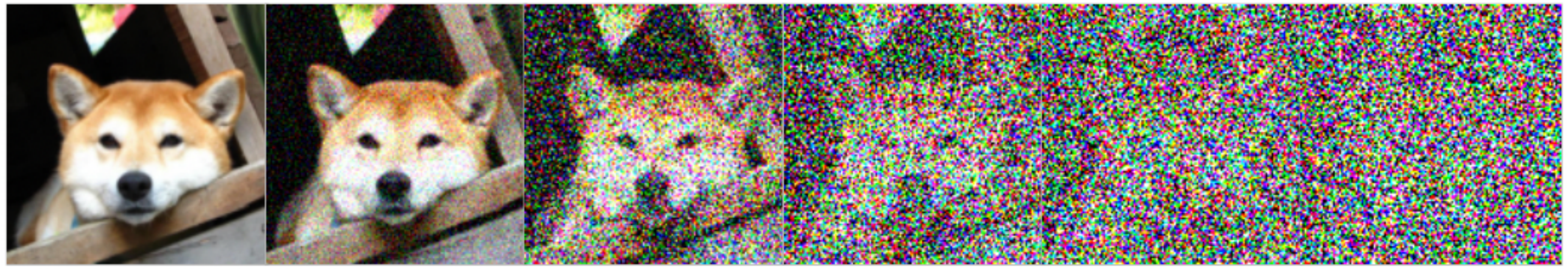
$q(x_{1:T}|x_0) =$



$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} N(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

- We start with x_0 and add a bit of Gaussian noise at each step (with shrinkage). So the distribution of x_t at step t will have to be Gaussian as well

Forward process: diffusion kernel



$$q(x_{1:T}|x_0) = \quad x_0 \xrightarrow{q(x_1|x_0)} x_1 \xrightarrow{q(x_2|x_1)} x_2 \longrightarrow \cdots \quad x_{T-1} \xrightarrow{q(x_T|x_{T-1})} x_T$$

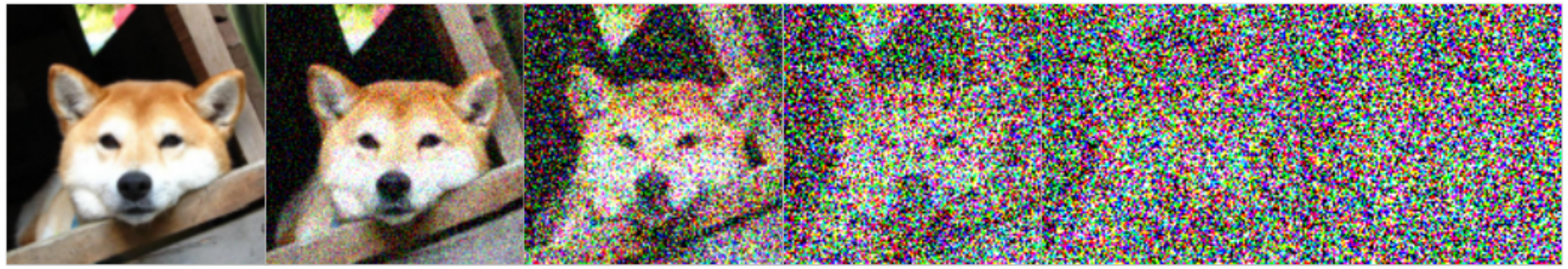
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} N(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

- We start with x_0 and add a bit of Gaussian noise at each step (with shrinkage). So the distribution of x_t at step t will have to be Gaussian as well

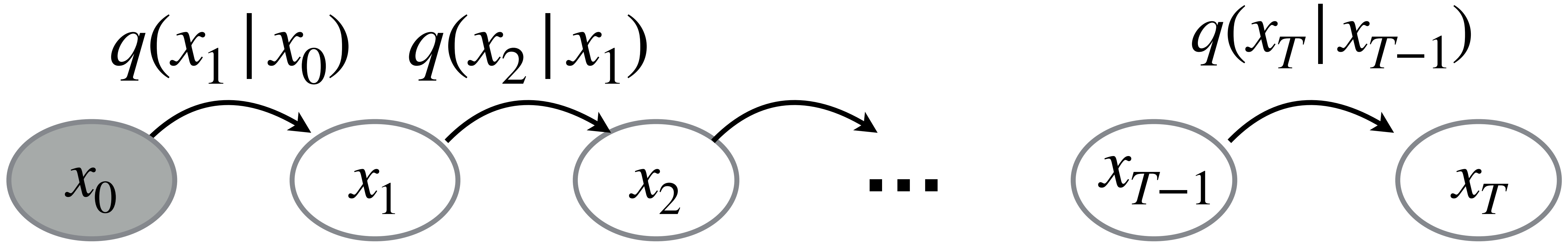
$$q_t(x_t|x_0) = N\left(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)$$ (diffusion kernel) $$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \;\; \epsilon \sim N(0,I)$$

Forward process: diffusion kernel



$q(x_{1:T}|x_0) =$



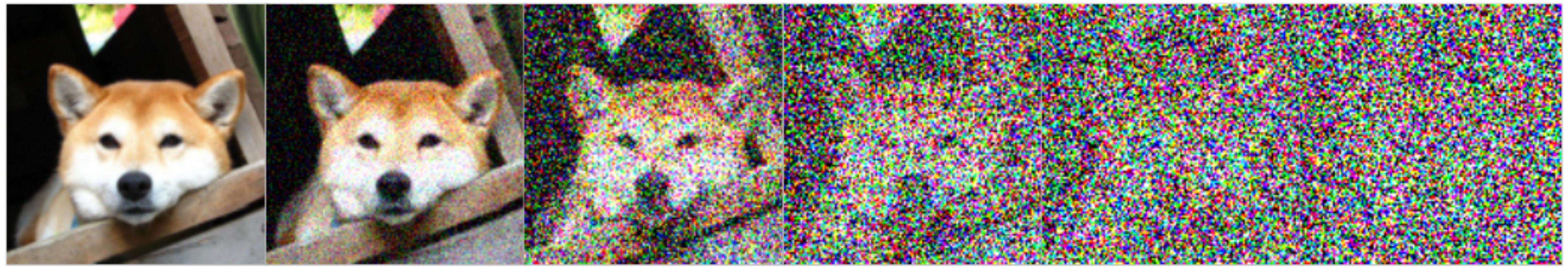
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} N(x_t|\sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

- We start with x_0 and add a bit of Gaussian noise at each step (with shrinkage). So the distribution of x_t at step t will have to be Gaussian as well

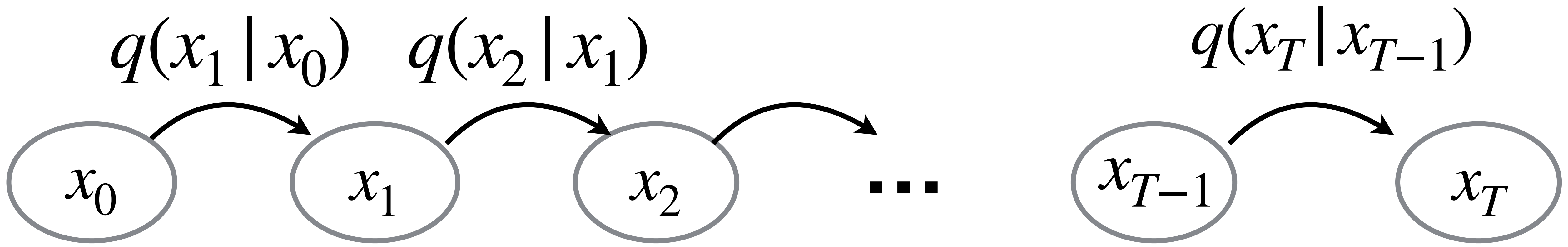
$$q_t(x_t|x_0) = N\left(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)$$ (diffusion kernel) $$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \ \ \epsilon \sim N(0,I)$$ For large T $x_T \approx \epsilon, \ \ \epsilon \sim N(0,I)$

Forward process: marginal at t



$q(x_{1:T}|x_0) =$

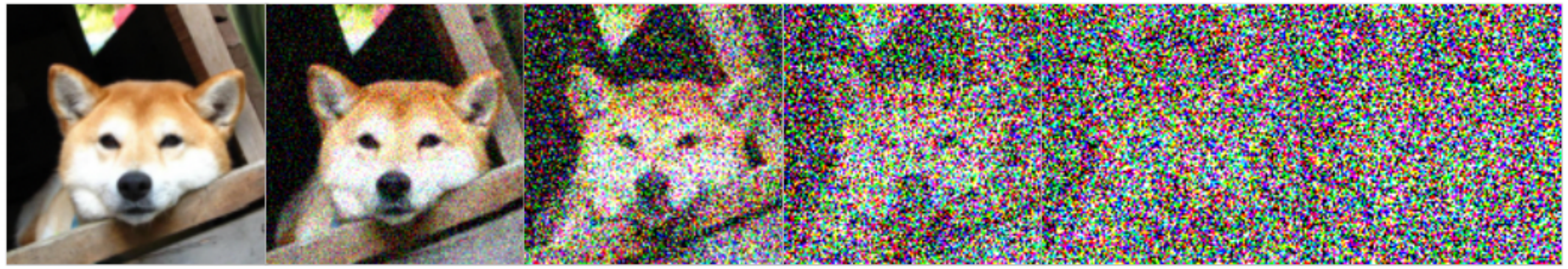


- The diffusion kernel tells us how x_t is distributed conditioned on x_0

$$q_t(x_t|x_0) = N\left(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)$$

- If we sample data $x_0 \sim q(x_0)$ (e.g., uniform at random), the noisy image x_t at step t has a complex marginal distribution

Forward process: marginal at t



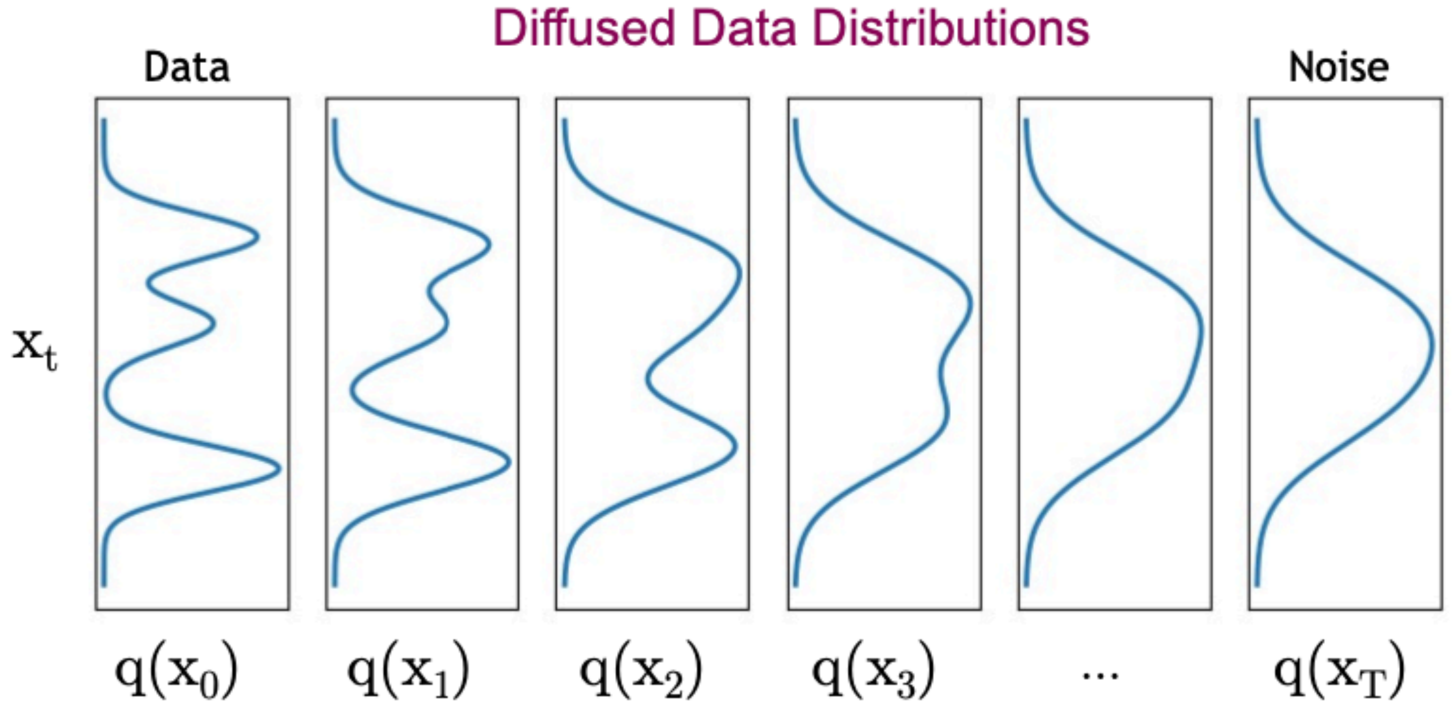
$$q(x_{1:T}|x_0) = \begin{tikzcd} x_0 \arrow[r, bend left, "q(x_1|x_0)"] & x_1 \arrow[r, bend left, "q(x_2|x_1)"] & x_2 \arrow[r, bend left] & \cdots & x_{T-1} \arrow[r, bend left, "q(x_T|x_{T-1})"] & x_T \end{tikzcd}$$

- The diffusion kernel tells us how x_t is distributed conditioned on x_0

$$q_t(x_t|x_0) = N\left(x_t|\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I\right)$$

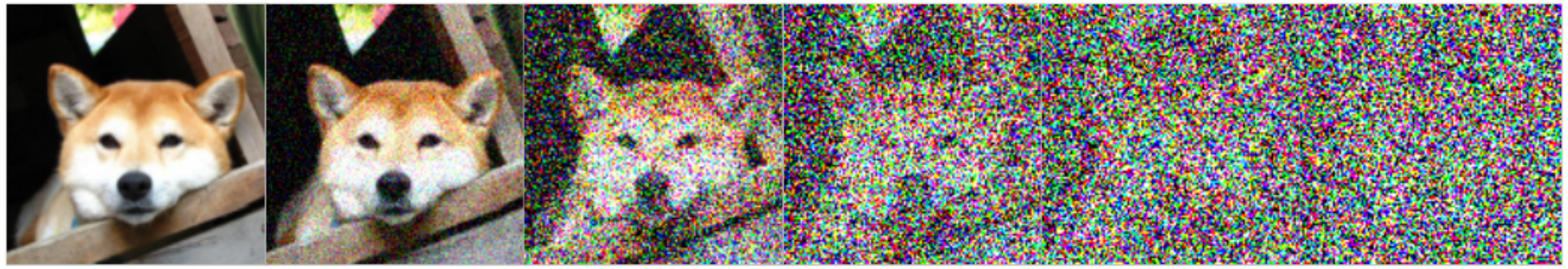
- If we sample data $x_0 \sim q(x_0)$ (e.g., uniform at random), the noisy image x_t at step t has a complex marginal distribution

$$q_t(x_t) = \int q_t(x_t|x_0)q(x_0)dx_0 \approx \frac{1}{n}\sum_{i=1}^{n} q_t(x_t|x_0^i)$$



[Vahdat et al 2022]

Reverse process — our generative model

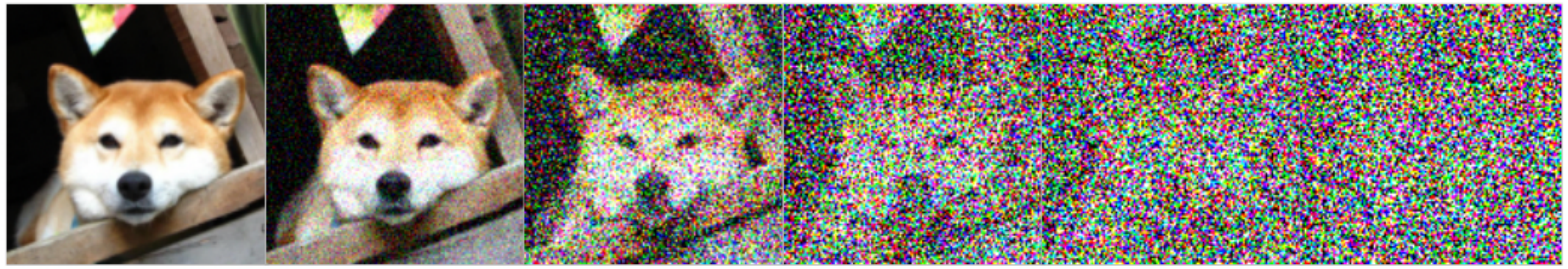


$p_\theta(x_{0:T}) = \quad p_\theta(x_0 \mid x_1) \quad \cdots \quad p_\theta(x_{T-1} \mid x_T) \quad p(x_T)$

- We use a reverse generative process to try to de-noise images back into clean versions, starting with x_T

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} N(x_{t-1} \mid \mu_\theta(x_t, t), \beta_t I)$$

Reverse process — our generative model



$$x_0 \leftarrow x_1 \leftarrow x_2 \quad \cdots \quad \leftarrow x_{T-1} \leftarrow x_T$$

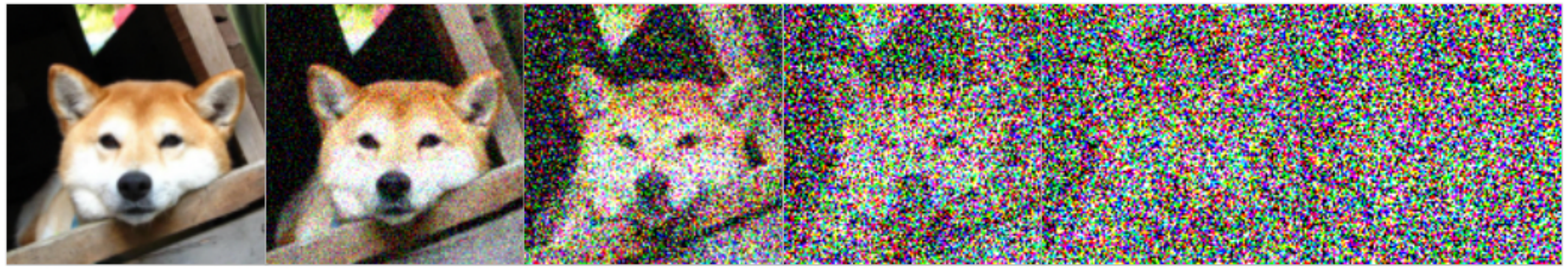
$$p_\theta(x_{0:T}) = \quad p_\theta(x_0 \mid x_1) \qquad\qquad p_\theta(x_{T-1} \mid x_T) \quad p(x_T)$$

- We use a reverse generative process to try to de-noise images back into clean versions, starting with x_T

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} N(x_{t-1} \mid \mu_\theta(x_t, t), \beta_t I)$$

- We can set $p(x_T) = N(x_T \mid 0, I)$ since the forward process always ends with $x_T \sim N(0, I)$

Reverse process — our generative model



$$x_0 \leftarrow x_1 \leftarrow x_2 \quad \cdots \quad \leftarrow x_{T-1} \leftarrow x_T$$

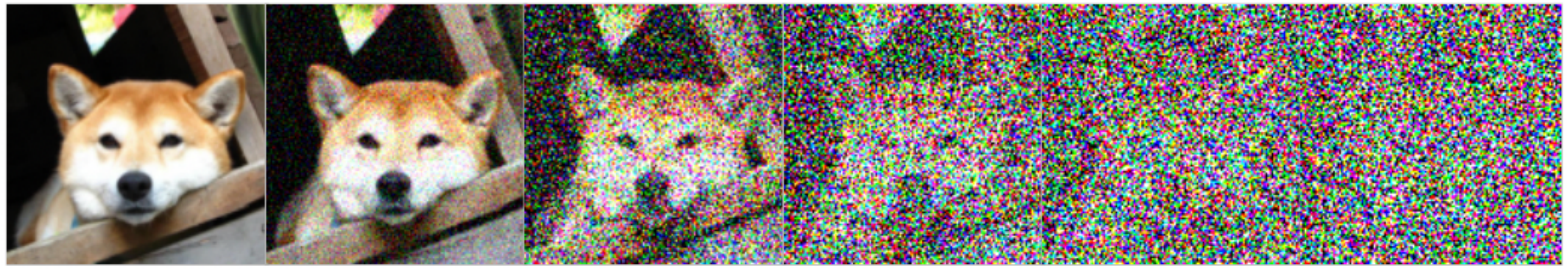
$$p_\theta(x_{0:T}) = \quad p_\theta(x_0 \,|\, x_1) \qquad\qquad p_\theta(x_{T-1} \,|\, x_T) \quad p(x_T)$$

- We use a reverse generative process to try to de-noise images back into clean versions, starting with x_T

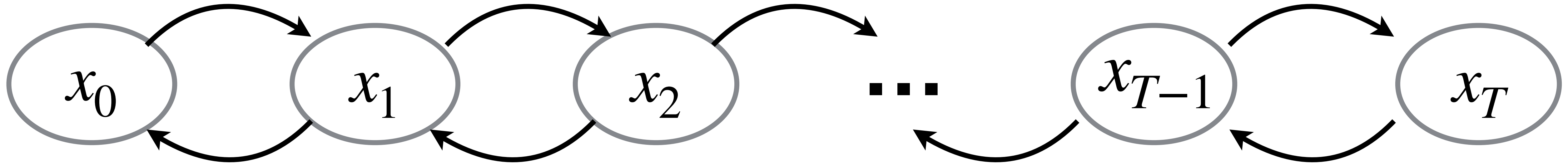
$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} N(x_{t-1} \,|\, \mu_\theta(x_t, t), \beta_t I)$$

- We can set $p(x_T) = N(x_T \,|\, 0, I)$ since the forward process always ends with $x_T \sim N(0, I)$
- Learning the de-noising “vector field” $\mu_\theta(x_t, t)$ is the key part!

Reparameterization of the reverse process



$$q(x_{1:T}|x_0) = \quad q(x_1|x_0) \quad q(x_2|x_1) \quad q(x_T|x_{T-1})$$



$$p_\theta(x_{0:T}) = \quad p_\theta(x_0|x_1) \quad p_\theta(x_{T-1}|x_T) \quad p(x_T)$$

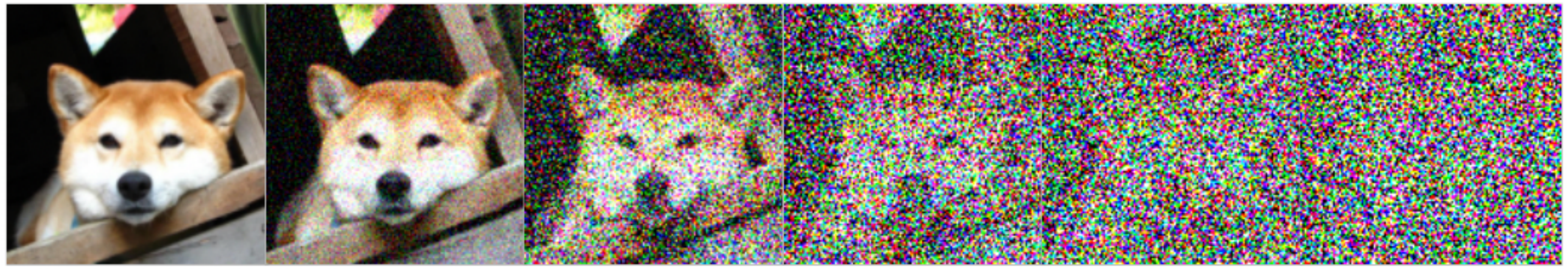
$$\alpha_t = (1-\beta_t)$$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} N(x_{t-1}|\mu_\theta(x_t, t), \beta_t I)$$

- Recall that based on the forward process: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \ \ \epsilon \sim N(0,I)$

Reparameterization of the reverse process



$$q(x_{1:T}|x_0) = \quad q(x_1|x_0) \quad q(x_2|x_1) \quad \cdots \quad q(x_T|x_{T-1})$$

$$\begin{tikzcd} x_0 \arrow[r, bend left, "q(x_1|x_0)"] & x_1 \arrow[r, bend left, "q(x_2|x_1)"] \arrow[l, bend left, "p_\theta(x_0|x_1)"] & x_2 \arrow[r, bend left] \arrow[l, bend left] & \cdots & x_{T-1} \arrow[l, bend left] \arrow[r, bend left, "q(x_T|x_{T-1})"] & x_T \arrow[l, bend left, "p_\theta(x_{T-1}|x_T)"] \end{tikzcd}$$

$$p_\theta(x_{0:T}) = \quad p_\theta(x_0|x_1) \quad \cdots \quad p_\theta(x_{T-1}|x_T) \quad p(x_T)$$

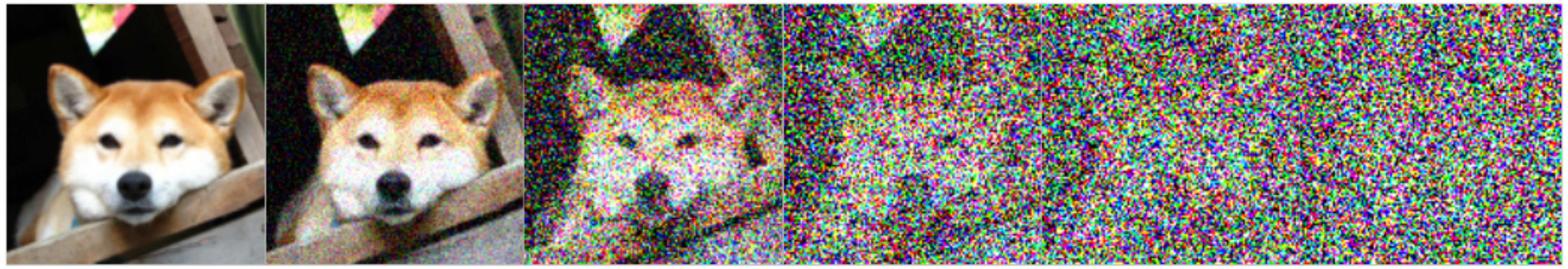
$$\alpha_t = (1 - \beta_t)$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} (1 - \beta_s)$$

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} N(x_{t-1} | \mu_\theta(x_t, t), \beta_t I)$$

- Recall that based on the forward process: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \ \ \epsilon \sim N(0,I)$
- We can predict the added noise ϵ instead (at the same scale for all t).

Reparameterization of the reverse process



$$q(x_{1:T}|x_0) = \quad q(x_1|x_0) \quad q(x_2|x_1) \quad \cdots \quad q(x_T|x_{T-1})$$

$$x_0 \quad x_1 \quad x_2 \quad \cdots \quad x_{T-1} \quad x_T$$

$$p_\theta(x_{0:T}) = \quad p_\theta(x_0|x_1) \quad \cdots \quad p_\theta(x_{T-1}|x_T) \quad p(x_T)$$

$$\alpha_t = (1 - \beta_t)$$

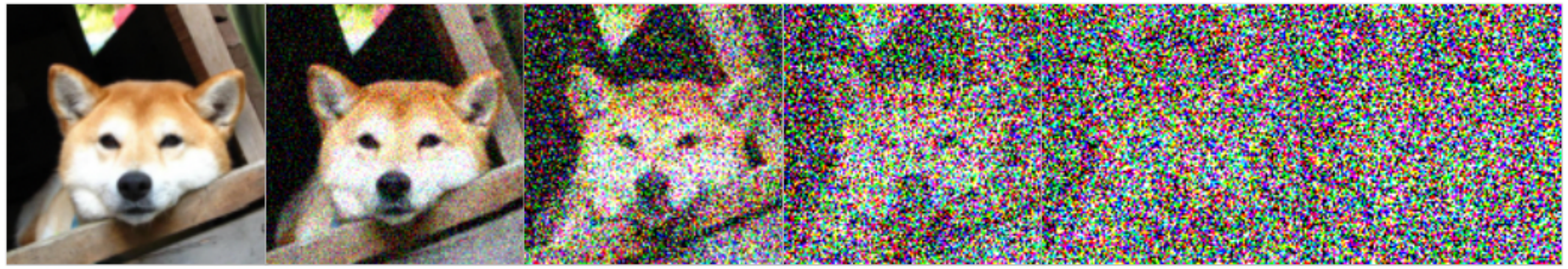
$$\bar{\alpha}_t = \prod_{s=1}^{t}(1 - \beta_s)$$

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} N(x_{t-1}|\mu_\theta(x_t, t), \beta_t I)$$

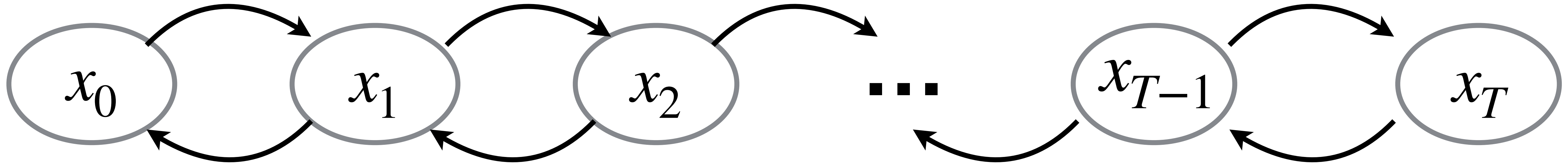
- Recall that based on the forward process: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, \;\; \epsilon \sim N(0,I)$
- We can predict the added noise ϵ instead (at the same scale for all t). It allows us to calculate an estimate of x_0

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}}}(x_t - \sqrt{1 - \bar{\alpha}_t}\,\epsilon)$$

Reparameterization of the reverse process



$q(x_{1:T}|x_0) =$ $q(x_1|x_0)$ $q(x_2|x_1)$ $q(x_T|x_{T-1})$



$p_\theta(x_{0:T}) =$ $p_\theta(x_0|x_1)$ $p_\theta(x_{T-1}|x_T)$ $p(x_T)$

$$\alpha_t = (1-\beta_t)$$

$$\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

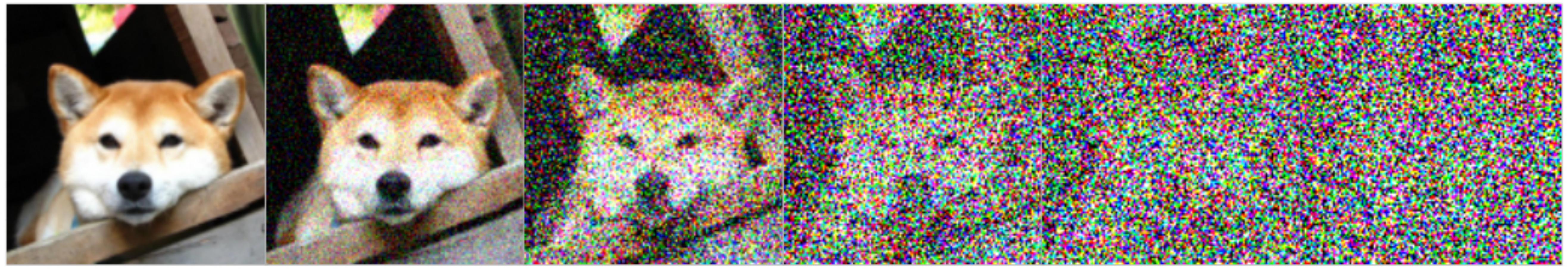
$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} N(x_{t-1}|\mu_\theta(x_t, t), \beta_t I)$$

- Recall that based on the forward process: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \;\; \epsilon \sim N(0,I)$
- We can predict the added noise ϵ instead (at the same scale for all t). It allows us to calculate an estimate of x_0 and what the de-noising step should be

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}}}(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon) \qquad q(x_{t-1}|x_t, \hat{x}_0) = \frac{q(x_t|x_{t-1})q(x_{t-1}|\hat{x}_0)}{q(x_t|\hat{x}_0)}$$

all the terms are Gaussian

Reparameterization of the reverse process



$$q(x_{1:T}|x_0) =$$

$$\begin{tikzcd} x_0 \arrow[r, bend left, "q(x_1|x_0)"] & x_1 \arrow[l, bend left, "p_\theta(x_0|x_1)"] \arrow[r, bend left, "q(x_2|x_1)"] & x_2 \arrow[l, bend left] \arrow[r, bend left] & \cdots & x_{T-1} \arrow[l, bend left] \arrow[r, bend left, "q(x_T|x_{T-1})"] & x_T \arrow[l, bend left, "p_\theta(x_{T-1}|x_T)"] \end{tikzcd}$$

$$p_\theta(x_{0:T}) = \qquad p(x_T)$$

$$\alpha_t = (1 - \beta_t)$$

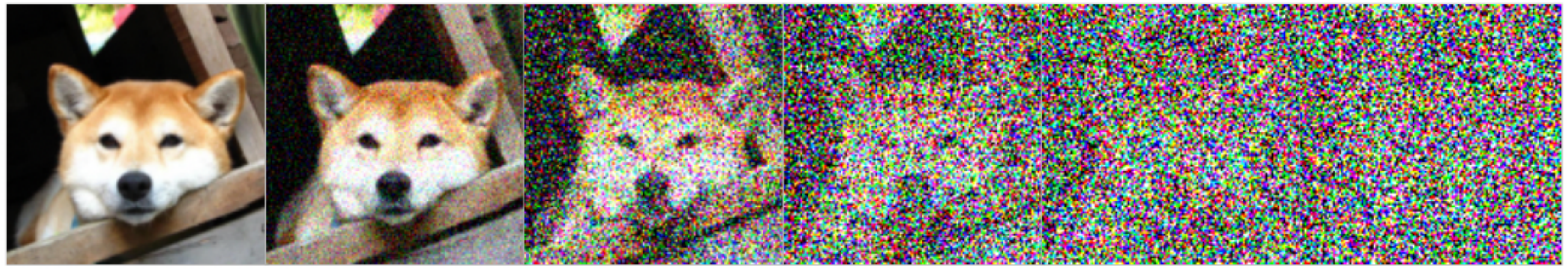
$$\bar{\alpha}_t = \prod_{s=1}^{t} (1 - \beta_s)$$

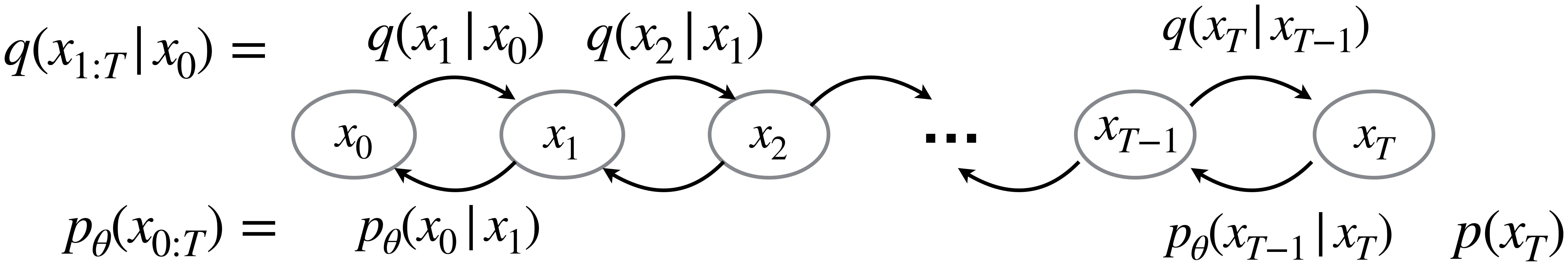
$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} N(x_{t-1} | \mu_\theta(x_t, t), \beta_t I)$$

- Recall that based on the forward process: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \;\; \epsilon \sim N(0,I)$
- We can predict the added noise ϵ instead (at the same scale for all t). It allows us to calculate an estimate of x_0 and what the de-noising step should be

$$\Rightarrow \qquad \mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

Basic training, sampling





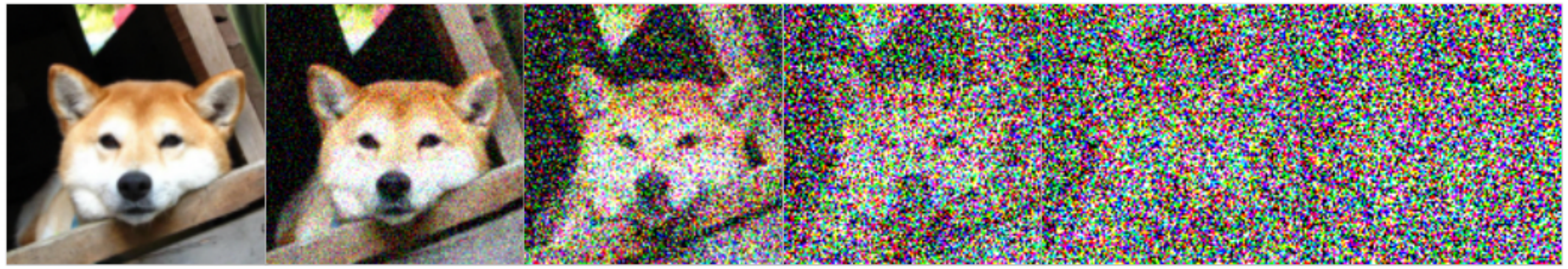
Algorithm 1 Training

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

Basic training, sampling



$$q(x_{1:T}|x_0) = \quad q(x_1|x_0) \quad q(x_2|x_1) \quad \cdots \quad q(x_T|x_{T-1})$$

$x_0 \quad x_1 \quad x_2 \quad \cdots \quad x_{T-1} \quad x_T$

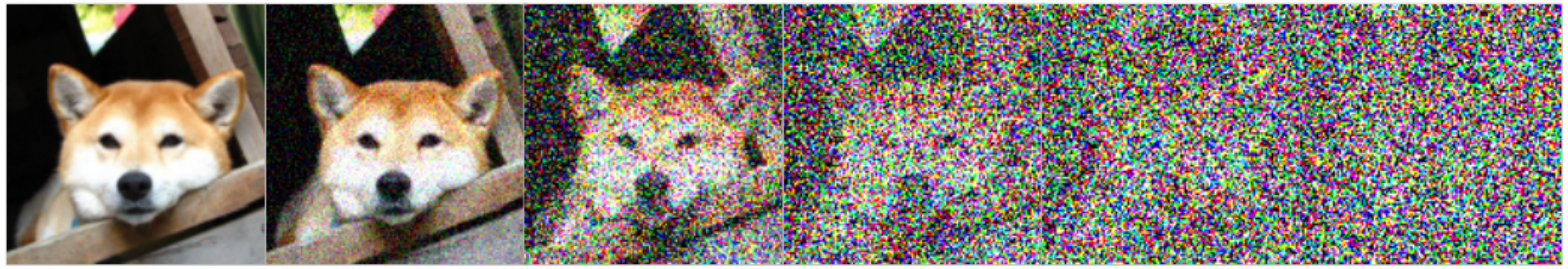
$$p_\theta(x_{0:T}) = \quad p_\theta(x_0|x_1) \quad \cdots \quad p_\theta(x_{T-1}|x_T) \quad p(x_T)$$

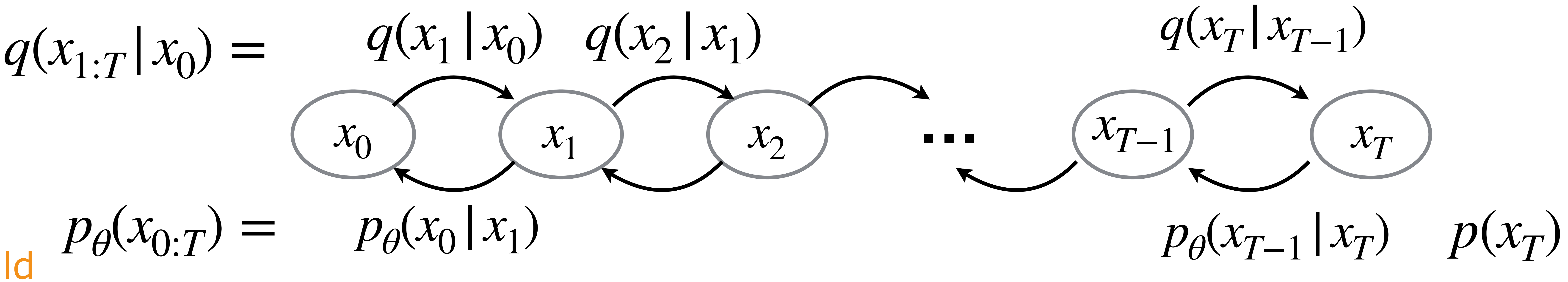
strictly, ELBO would imply a different weighting

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
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

Basic training, sampling





strictly, ELBO would imply a different weighting

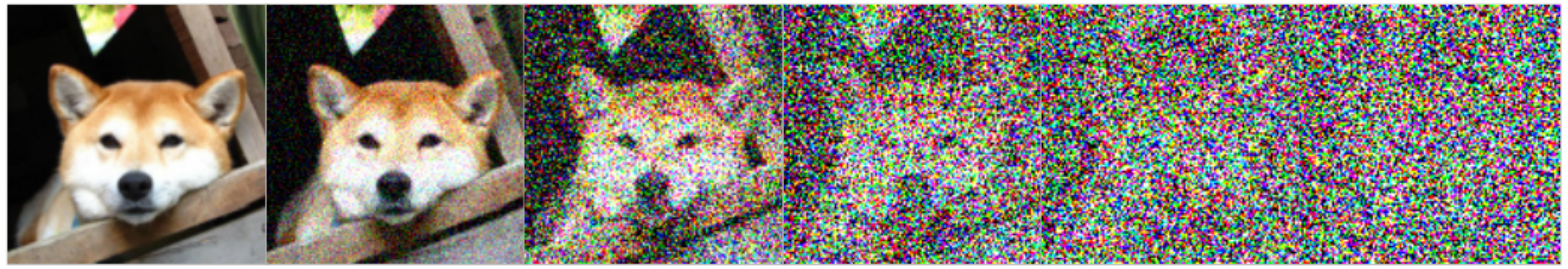
Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  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

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **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 $\mathbf{x}_0$

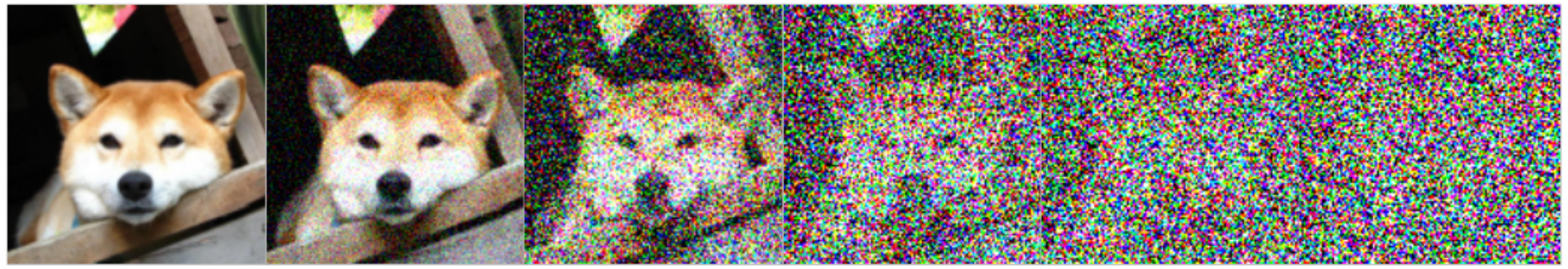
Score based (continuous) diffusion models



x_0 x_1 x_{T-1} x_T

- What if we used more steps in between [0,T], adding less noise at each?

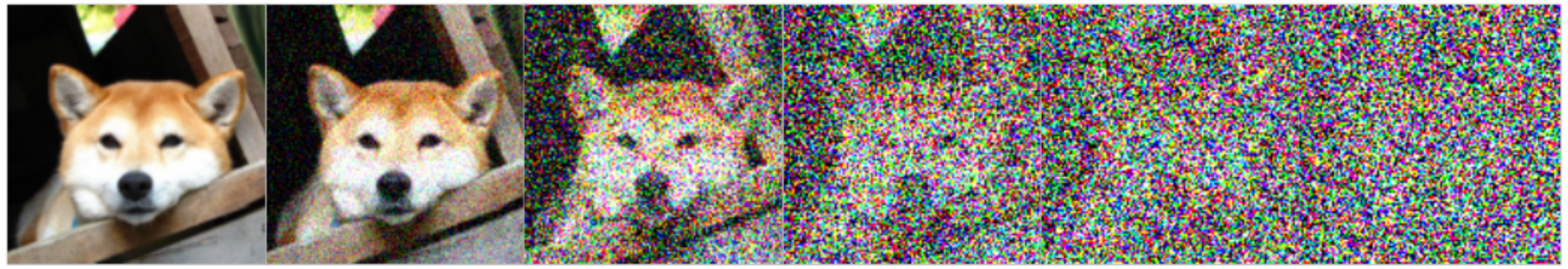
Score based (continuous) diffusion models



- What if we used more steps in between [0,T], adding less noise at each? We can write the noise variance now as $\beta(t)\Delta t$ (approaching zero)

$$x_t = \sqrt{1 - \beta(t)\Delta t}\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \quad \epsilon \sim N(0,I)$$

Score based (continuous) diffusion models

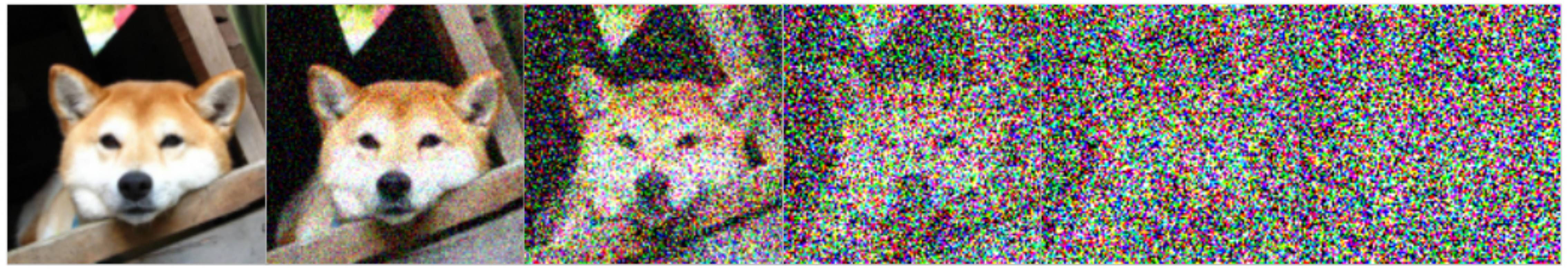


- What if we used more steps in between [0,T], adding less noise at each? We can write the noise variance now as $\beta(t)\Delta t$ (approaching zero)

$$x_t = \sqrt{1 - \beta(t)\Delta t}\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \;\; \epsilon \sim N(0,I)$$

$$x_t - x_{t-1} = (\sqrt{1 - \beta(t)\Delta t} - 1)\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \;\; \epsilon \sim N(0,I)$$

Score based (continuous) diffusion models



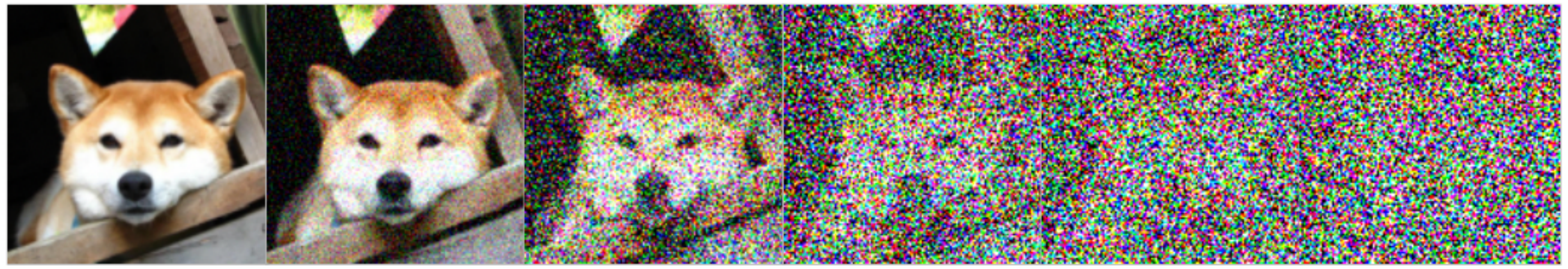
- What if we used more steps in between [0,T], adding less noise at each? We can write the noise variance now as $\beta(t)\Delta t$ (approaching zero)

$$x_t = \sqrt{1 - \beta(t)\Delta t}\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \ \ \epsilon \sim N(0,I)$$

$$x_t - x_{t-1} = (\sqrt{1 - \beta(t)\Delta t} - 1)\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \ \ \epsilon \sim N(0,I)$$

$$\Delta x_t \approx -\frac{1}{2}\beta(t)\Delta t\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \ \ \epsilon \sim N(0,I)$$

Score based (continuous) diffusion models



- What if we used more steps in between [0,T], adding less noise at each? We can write the noise variance now as $\beta(t)\Delta t$ (approaching zero)

$$x_t = \sqrt{1 - \beta(t)\Delta t}\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \;\; \epsilon \sim N(0,I)$$

$$x_t - x_{t-1} = (\sqrt{1 - \beta(t)\Delta t} - 1)\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \;\; \epsilon \sim N(0,I)$$

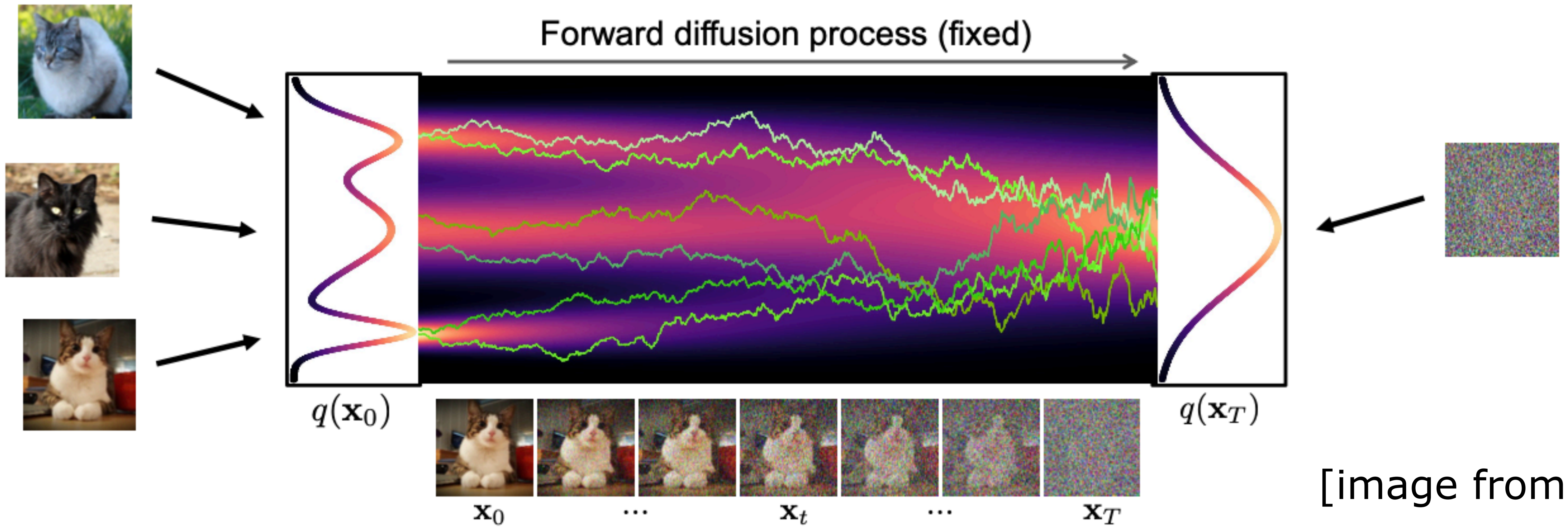
$$\Delta x_t \approx -\frac{1}{2}\beta(t)\Delta t\, x_{t-1} + \sqrt{\beta(t)\Delta t}\, \epsilon, \;\; \epsilon \sim N(0,I)$$

- Taking the limit of Δt we arrive at a stochastic differential equation (SDE)

Brownian motion

$$dx_t = -\frac{1}{2}\beta(t)\, x_t\, dt + \sqrt{\beta(t)}\, dw_t$$

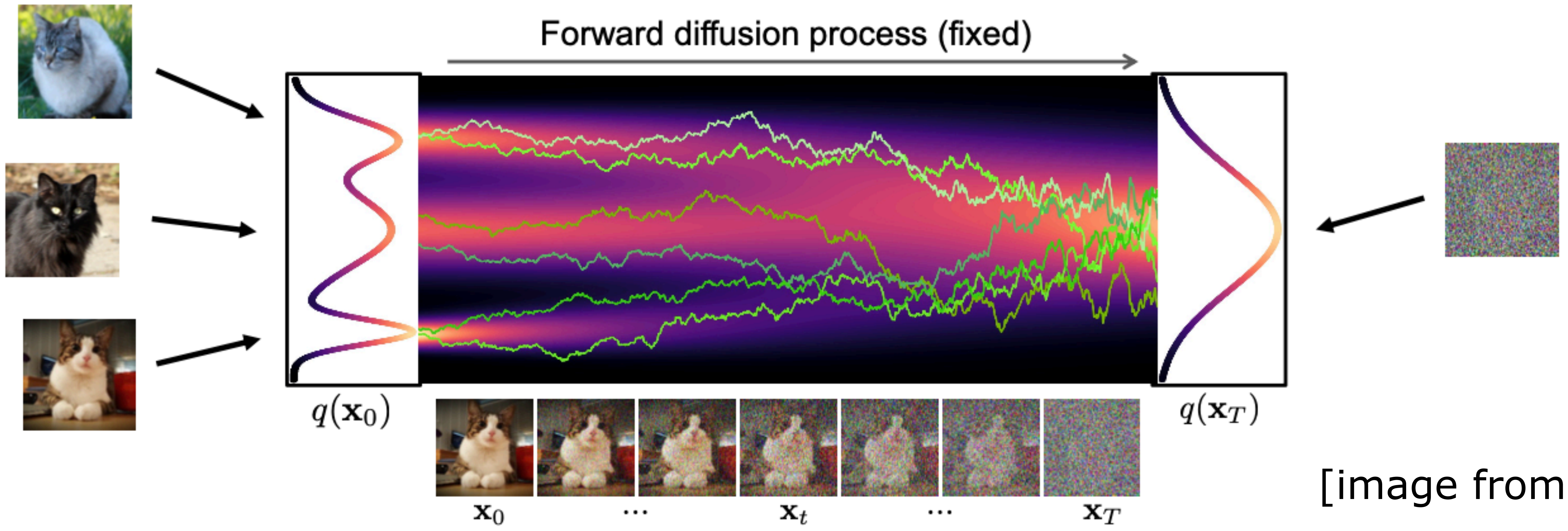
Forward diffusion process



[image from Vahdat et al 2022]

- Forward process: $$dx_t = -\frac{1}{2}\beta(t)\,x_t\,dt + \sqrt{\beta(t)}\,dw_t$$

Reverse diffusion process



[image from Vahdat et al 2022]

- Forward process: $$dx_t = -\frac{1}{2}\beta(t)\, x_t\, dt + \sqrt{\beta(t)}\, dw_t$$

- Key result [Anderson 82]: reverse process is also a diffusion process

also Brownian motion

$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \beta(t)\,\underbrace{\nabla_{x_t}\log q_t(x_t)}_{\text{score function}}\right] dt + \sqrt{\beta(t)}\, d\tilde{w}_t$$

(dt now negative)

Reverse diffusion process

- We would like to use the reverse process to sample new images ($x_T \sim N(0,I)$ as before)

$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \beta(t)\underbrace{\nabla_{x_t}\log q_t(x_t)}_{\text{score function}} \right] dt + \sqrt{\beta(t)}\, d\tilde{w}_t$$

- To do so we should learn a neural model $s_\theta(x,t)$ to approx. the score function
- How to learn $s_\theta(x,t)$?

Reverse diffusion process

- We would like to use the reverse process to sample new images ($x_T \sim N(0,I)$ as before)

$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \beta(t)\,\underbrace{\nabla_{x_t}\log q_t(x_t)}_{\text{score function}} \right] dt + \sqrt{\beta(t)}\,d\tilde{w}_t$$

- To do so we should learn a neural model $s_\theta(x,t)$ to approx. the score function
- How to learn $s_\theta(x,t)$? We could try to (similar to before)

$$x_0 \sim q(x_0),\;\; t \sim U(0,T)$$

$x_t \sim q_t(x_t \mid x_0)$ diffusion kernel (it is still just a Gaussian)

$$\theta \leftarrow \theta - \eta \nabla_\theta \| s_\theta(x_t,t) - \underline{\nabla_{x_t}\log q_t(x_t)} \|^2$$

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

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

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

[Vahdat et al 2022]

- But $q_t(x_t)$ is a complex distribution!

Reverse diffusion process

- We would like to use the reverse process to sample new images ($x_T \sim N(0,I)$ as before)

$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \beta(t)\underbrace{\nabla_{x_t}\log q_t(x_t)}_{\text{score function}} \right] dt + \sqrt{\beta(t)}\, d\tilde{w}_t$$

- To do so we should learn a neural model $s_\theta(x,t)$ to approx. the score function
- How to learn $s_\theta(x,t)$? We can do

$$x_0 \sim q(x_0),\;\; t \sim U(0,T)$$

$$x_t \sim q_t(x_t \,|\, x_0)$$ diffusion kernel (it is still just a Gaussian)

$$\theta \leftarrow \theta - \eta \nabla_\theta \|s_\theta(x_t, t) - \underline{\nabla_{x_t}\log q_t(x_t \,|\, x_0)}\|^2$$

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

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

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

[Vahdat et al 2022]

- $\nabla_{x_t}\log q_t(x_t \,|\, x_0)$ (score of a Gaussian) can be easily calculated and has the same expected value (over x_0 for a given x_t) when $x_0 \sim q(x_0)$!

Score expectation: derivation

- $\nabla_{x_t} \log q_t(x_t | x_0)$ is a score of a Gaussian that can be easily calculated
- It has the same expected value as the complex score function $\nabla_{x_t} \log q_t(x_t)$ we want and thus can be used as a (noisy) learning target

$$
\begin{aligned}
E_{x_0 \sim q(x_0|x_t)}\{\, \nabla_{x_t} \log q_t(x_t | x_0) \,\} &= \int q(x_0 | x_t)\, \nabla_{x_t} \log q_t(x_t | x_0)\, dx_0 \\
&= \int \frac{q(x_0) q_t(x_t | x_0)}{q_t(x_t)} \nabla_{x_t} \log q_t(x_t | x_0)\, dx_0 \\
&= \int \frac{q(x_0)}{q_t(x_t)} \nabla_{x_t} q_t(x_t | x_0)\, dx_0 \\
&= \frac{1}{q_t(x_t)} \nabla_{x_t} \int q_t(x_t | x_0) q(x_0)\, dx_0 \\
&= \frac{1}{q_t(x_t)} \nabla_{x_t} q_t(x_t) = \nabla_{x_t} \log q_t(x_t)
\end{aligned}
$$

Reverse diffusion process

- We would like to use the reverse process to sample new images ($x_T \sim N(0,I)$ as before)

$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \underbrace{\beta(t)\,\nabla_{x_t}\log q_t(x_t)}_{\text{score function}} \right] dt + \sqrt{\beta(t)}\, d\tilde{w}_t$$

- To do so we should learn a neural model $s_\theta(x,t)$ to approx. the score function
- How to learn $s_\theta(x,t)$? We could try to (similar to before)

$$x_0 \sim q(x_0), \;\; t \sim U(0,T)$$

$$x_t \sim q_t(x_t \mid x_0)$$ diffusion kernel (it is still just a Gaussian)

$$\theta \leftarrow \theta - \eta \nabla_\theta \| s_\theta(x_t, t) - \underline{\nabla_{x_t}\log q_t(x_t)} \|^2$$

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

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

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

[Vahdat et al 2022]

Additional reading

- **Some early diffusion model papers:**
- Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics", http://proceedings.mlr.press/v37/sohl-dickstein15.pdf
- Ho et al., "Denoising Diffusion Probabilistic Models", https://arxiv.org/abs/2006.11239
- Song et al., "Generative Modeling by Estimating Gradients of the Data Distribution", https://arxiv.org/abs/1907.05600
- **Tutorials (excerpts used in this lecture)**
- A. Vahdat, K. Kreis, R. Gao, "CVPR 2022 Tutorial — Denoising Diffusion-based Generative Modeling: Foundations and Applications", https://cvpr2022-tutorial-diffusion-models.github.io/