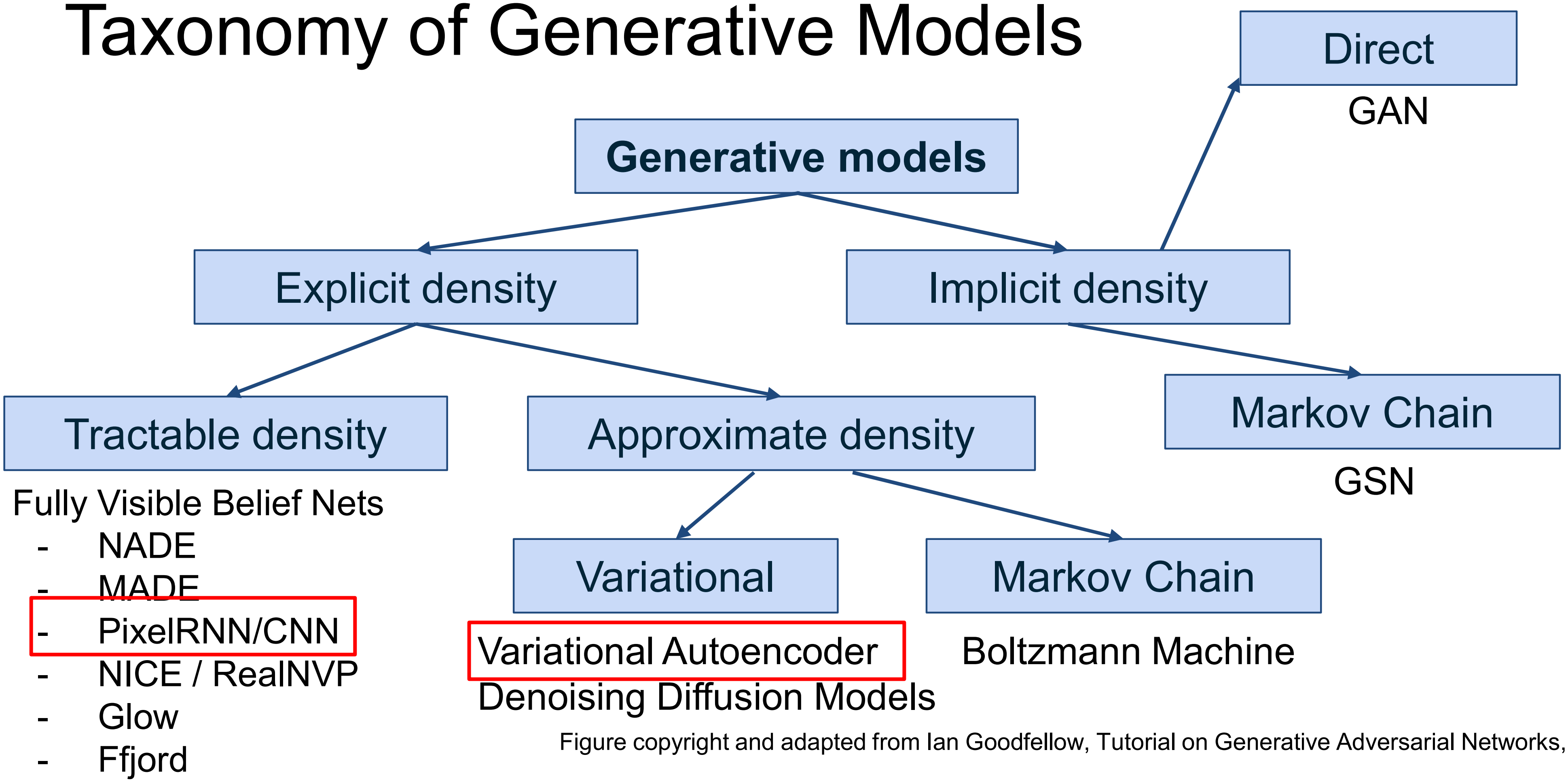
CS 4644-DL / 7643-A
ZSOLT KIRA

Generative Models:
Denoising Diffusion Probabilistic Models (DDPMs)

Slides adapted from those by Danfei Xu

Taxonomy of Generative Models



Generative models

Explicit density

Implicit density

Direct

GAN

Tractable density

Fully Visible Belief Nets
- NADE
- MADE
- PixelRNN/CNN
- NICE / RealNVP
- Glow
- Ffjord

Approximate density

Variational

Variational Autoencoder

Denoising Diffusion Models

Markov Chain

Boltzmann Machine

Markov Chain

GSN

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region (**masked convolution**)

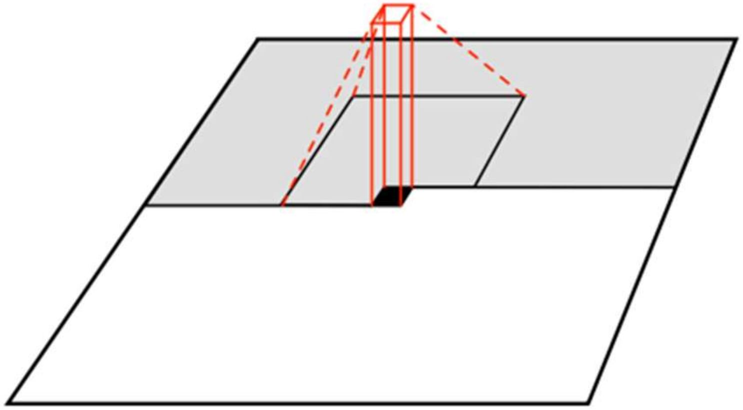


Figure copyright van der Oord et al., 2016. Reproduced with permission.

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

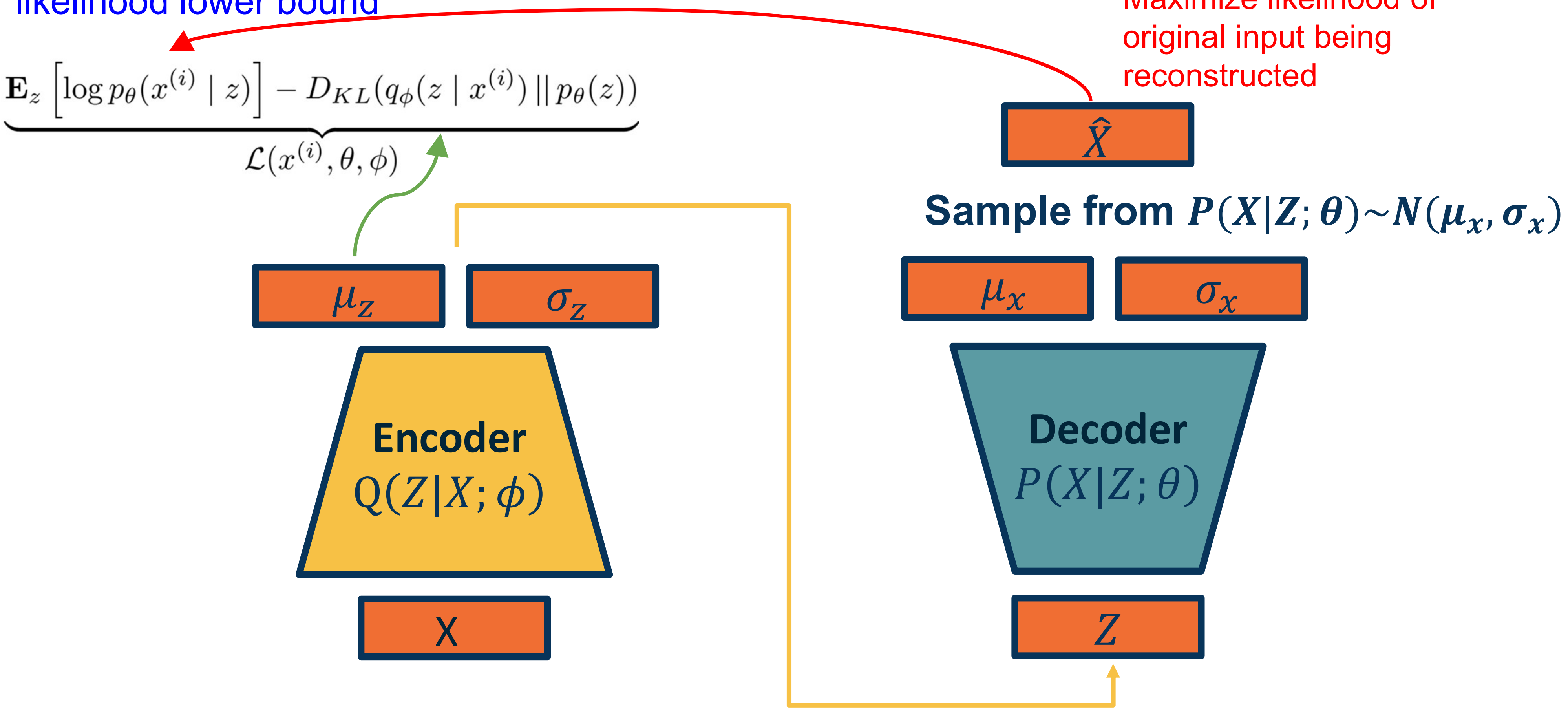
Decoder network gives $p_\theta(x|\mathbf{z})$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and **z** prior) has nice closed-form solution!

$p_\theta(\mathbf{z}|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Putting it all together: maximizing the likelihood lower bound



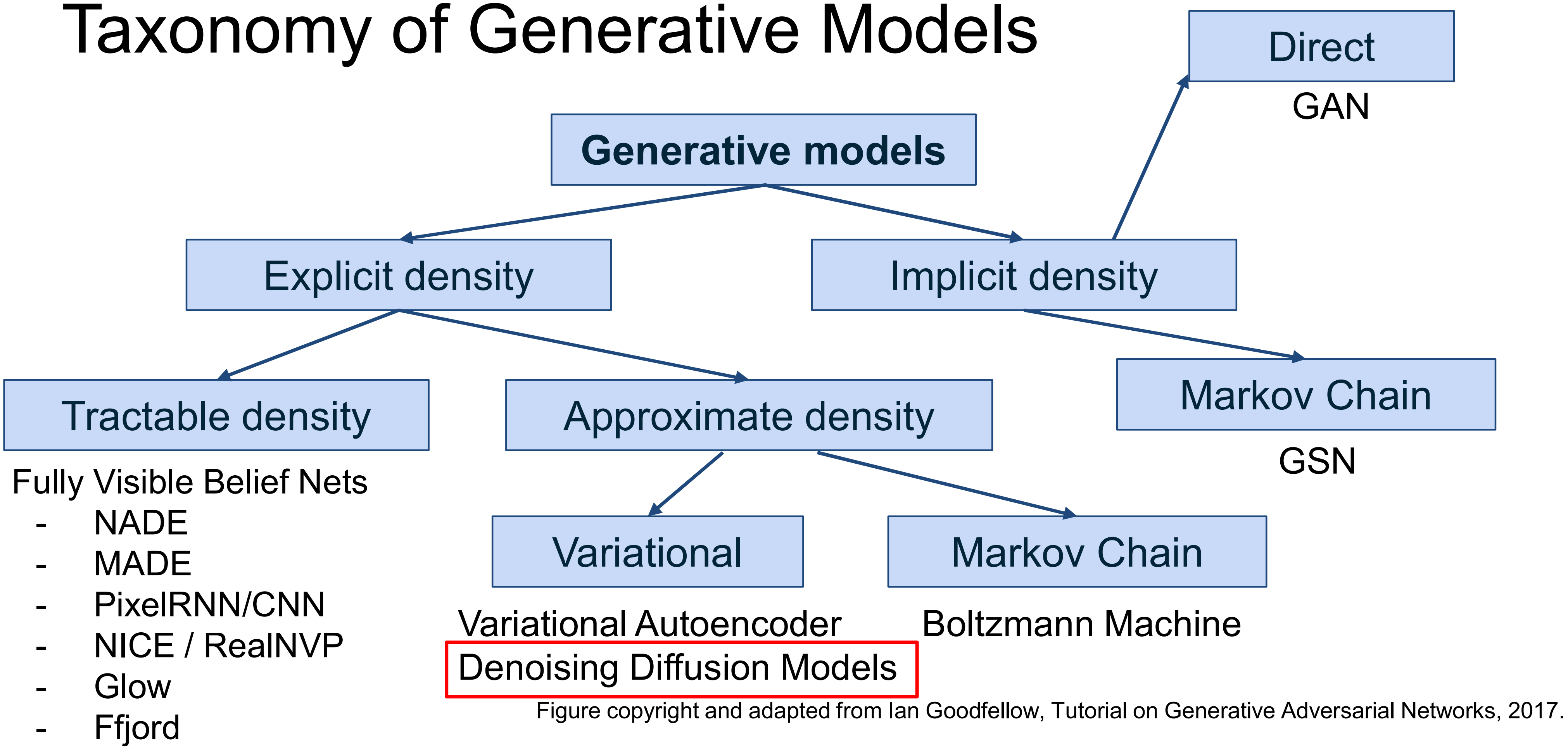
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Maximize likelihood of original input being reconstructed

Sample from $\boldsymbol{P(X|Z;\theta) \sim N(\mu_x, \sigma_x)}$

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Taxonomy of Generative Models



Generative models

Explicit density

Implicit density

Direct

GAN

Markov Chain

GSN

Tractable density

Fully Visible Belief Nets
- NADE
- MADE
- PixelRNN/CNN
- NICE / RealNVP
- Glow
- Ffjord

Approximate density

Variational

Variational Autoencoder

Denoising Diffusion Models

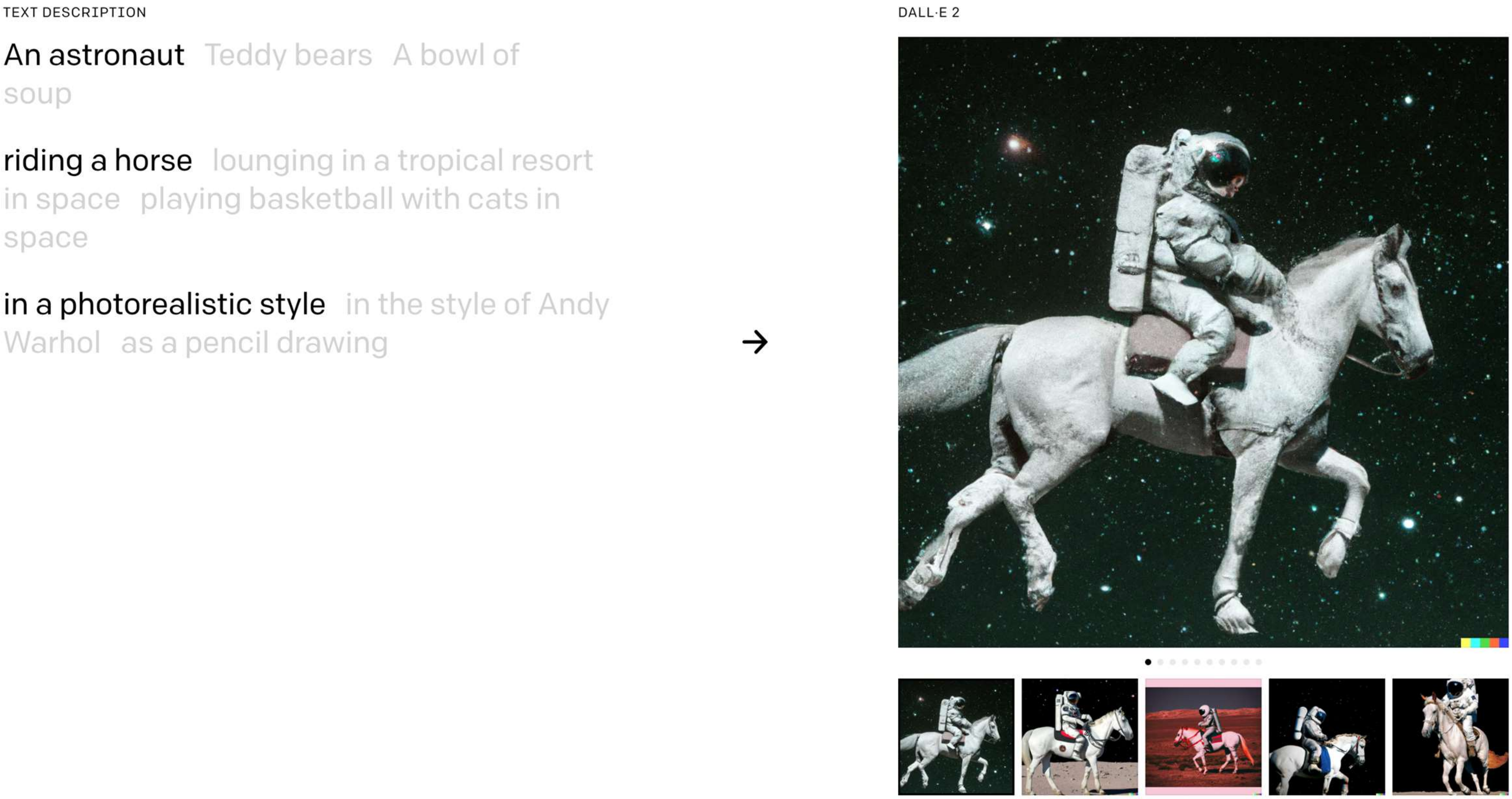
Markov Chain

Boltzmann Machine

Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Denoising Diffusion Probabilistic Models (DDPMs)

And Conditional Diffusion Models



TEXT DESCRIPTION

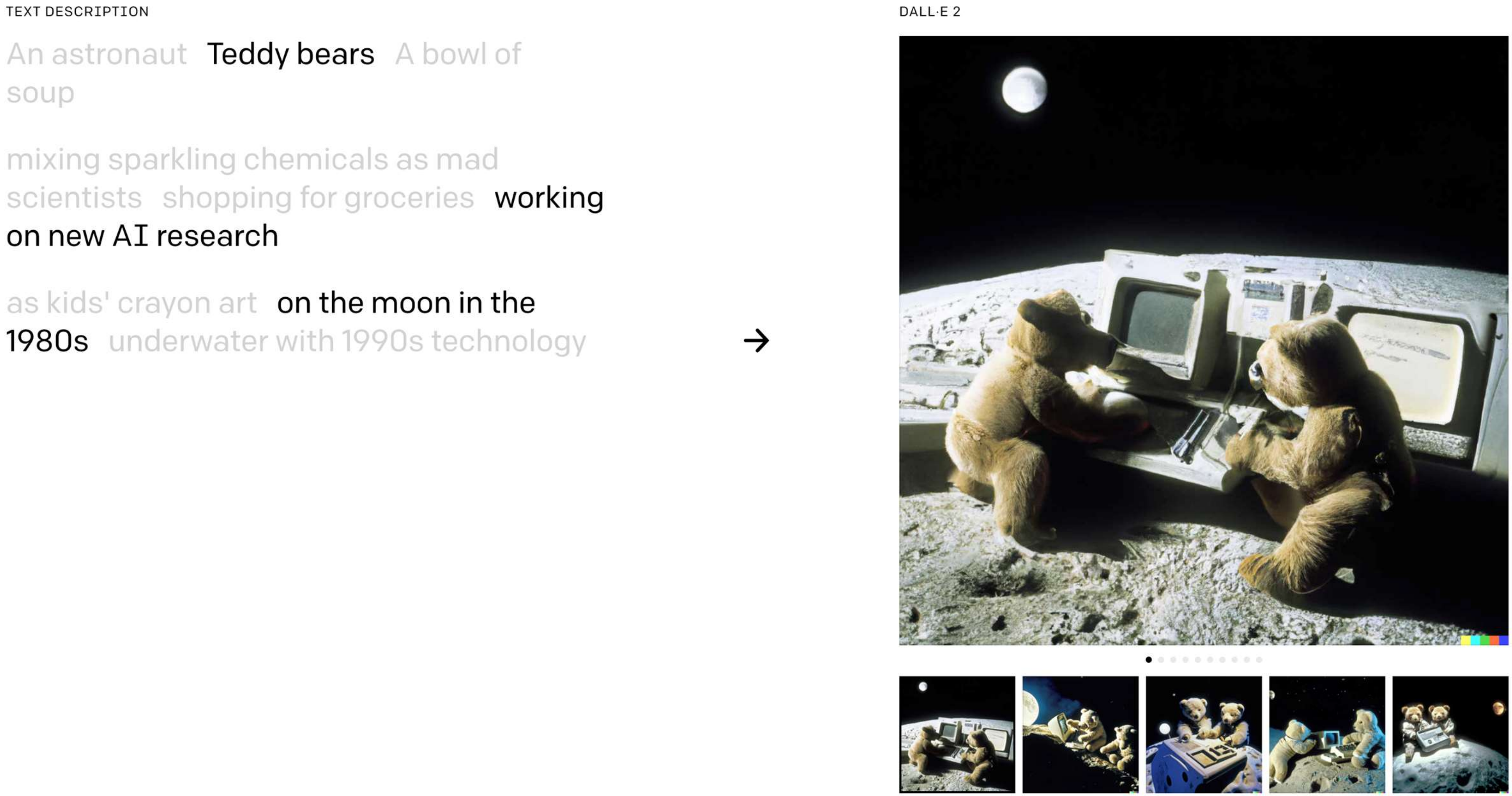
An astronaut Teddy bears A bowl of soup

riding a horse lounging in a tropical resort in space playing basketball with cats in space

in a photorealistic style in the style of Andy Warhol as a pencil drawing

→

https://openai.com/dall-e-2/



TEXT DESCRIPTION

An astronaut Teddy bears A bowl of soup

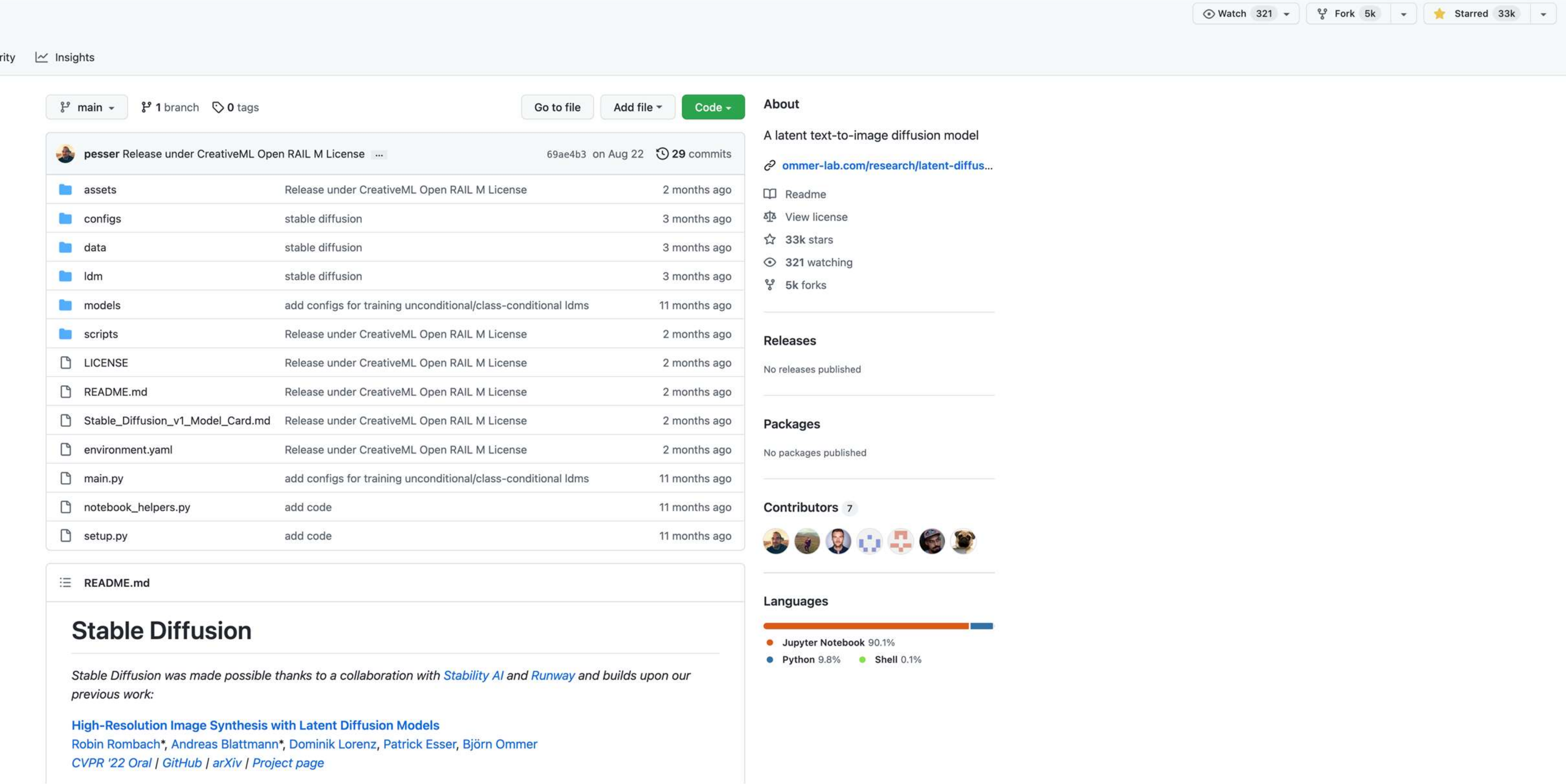
mixing sparkling chemicals as mad scientists shopping for groceries working on new AI research

as kids' crayon art on the moon in the 1980s underwater with 1990s technology

→

https://openai.com/dall-e-2/

https://openai.com/dall-e-2/



pesser Release under CreativeML Open RAIL M License … 69ae4b3 on Aug 22 29 commits

assets	Release under CreativeML Open RAIL M License	2 months ago
configs	stable diffusion	3 months ago
data	stable diffusion	3 months ago
ldm	stable diffusion	3 months ago
models	add configs for training unconditional/class-conditional ldms	11 months ago
scripts	Release under CreativeML Open RAIL M License	2 months ago
LICENSE	Release under CreativeML Open RAIL M License	2 months ago
README.md	Release under CreativeML Open RAIL M License	2 months ago
Stable_Diffusion_v1_Model_Card.md	Release under CreativeML Open RAIL M License	2 months ago
environment.yaml	Release under CreativeML Open RAIL M License	2 months ago
main.py	add configs for training unconditional/class-conditional ldms	11 months ago
notebook_helpers.py	add code	11 months ago
setup.py	add code	11 months ago

About

A latent text-to-image diffusion model

ommer-lab.com/research/latent-diffus...

Readme
View license
33k stars
321 watching
5k forks

Releases

No releases published

Packages

No packages published

Contributors 7

Languages

- **Jupyter Notebook** 90.1%
- **Python** 9.8%
- **Shell** 0.1%

README.md

Stable Diffusion

Stable Diffusion was made possible thanks to a collaboration with Stability AI and Runway and builds upon our previous work:

High-Resolution Image Synthesis with Latent Diffusion Models
Robin Rombach*, Andreas Blattmann*, Dominik Lorenz, Patrick Esser, Björn Ommer
CVPR '22 Oral | *GitHub* | *arXiv* | *Project page*

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

Landscape Highlights of Diffusion Models (Nov 2022)

basic principles

- *Diffusion probabilistic models* (Sohl-Dickstein et al., 2015)
- *Noise-conditioned score network* (**NCSN**; Yang & Ermon, 2019)
- *Denoising diffusion probabilistic models* (**DDPM**; Ho et al. 2020)

conditional & high-res image generation

- *Classifier-guided conditional generation* (Dhariwal and Nichole, 2021)
- *Classifier-free Diffusion Guidance* (Ho and Salimans, 2022)
- *Latent-space Diffusion* (**StableDiffusion**; Rombach and Blattmann et al., 2022)

new applications

- *Planning with Diffusion for Flexible Behavior Synthesis* (**Diffuser**; Janner et al., 2022)
- *DreamFusion: Text-to-3D using 2D Diffusion* (Poole and Jain et al., 2022)
- *Make-A-Video: Text-to-Video Generation without Text-Video Data* (Singer et al., 2022)

Landscape Highlights of Diffusion Models (Nov 2022)

basic principles

- *Diffusion probabilistic models* (Sohl-Dickstein et al., 2015)
- *Noise-conditioned score network* (**NCSN**; Yang & Ermon, 2019)
- *Denoising diffusion probabilistic models* (**DDPM**; Ho et al. 2020)

conditional & high-res image generation

- *Classifier-guided conditional generation* (Dhariwal and Nichole, 2021)
- *Classifier-free Diffusion Guidance* (Ho and Salimans, 2022)
- *Latent-space Diffusion* (**StableDiffusion**; Rombach and Blattmann et al., 2022)

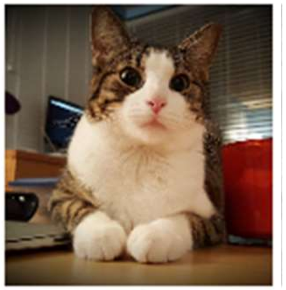
new applications

- *Planning with Diffusion for Flexible Behavior Synthesis* (**Diffuser**; Janner et al., 2022)
- *DreamFusion: Text-to-3D using 2D Diffusion* (Poole and Jain et al., 2022)
- *Make-A-Video: Text-to-Video Generation without Text-Video Data* (Singer et al., 2022)

The Denoising Diffusion Process

image from
dataset

x_0



The Denoising Diffusion Process

image from
dataset

The “forward diffusion” process:
add Gaussian noise each step

$$x_0 \longrightarrow x_1 \longrightarrow$$

…

The Denoising Diffusion Process

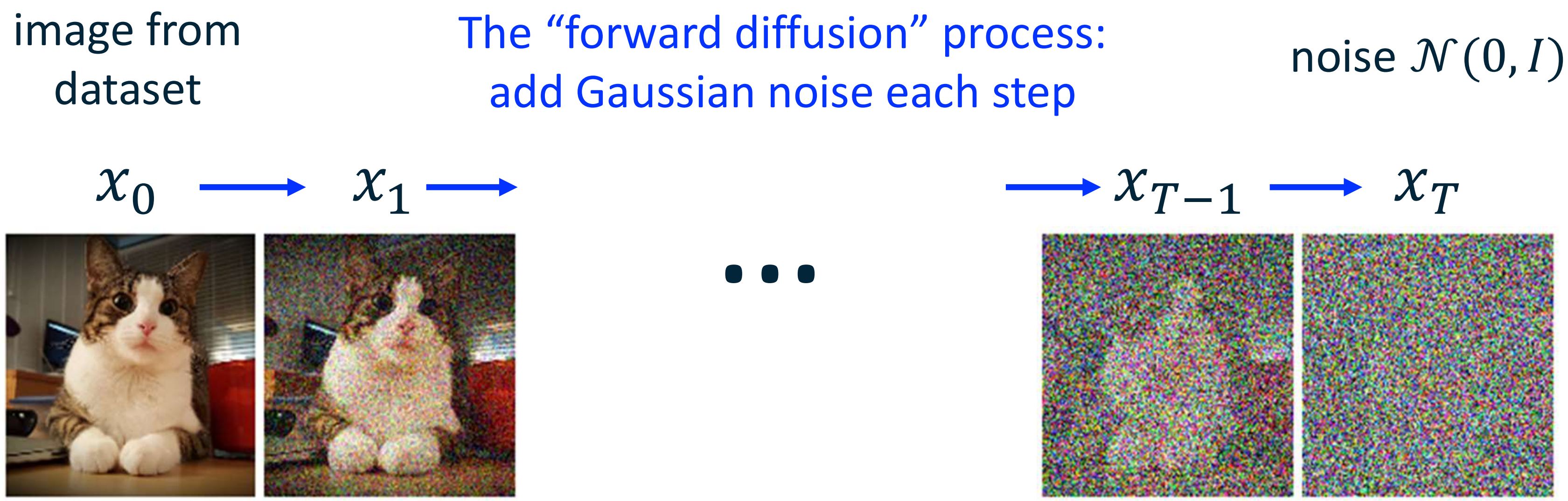


image from
dataset

The “forward diffusion” process:
add Gaussian noise each step

noise $\mathcal{N}(0, I)$

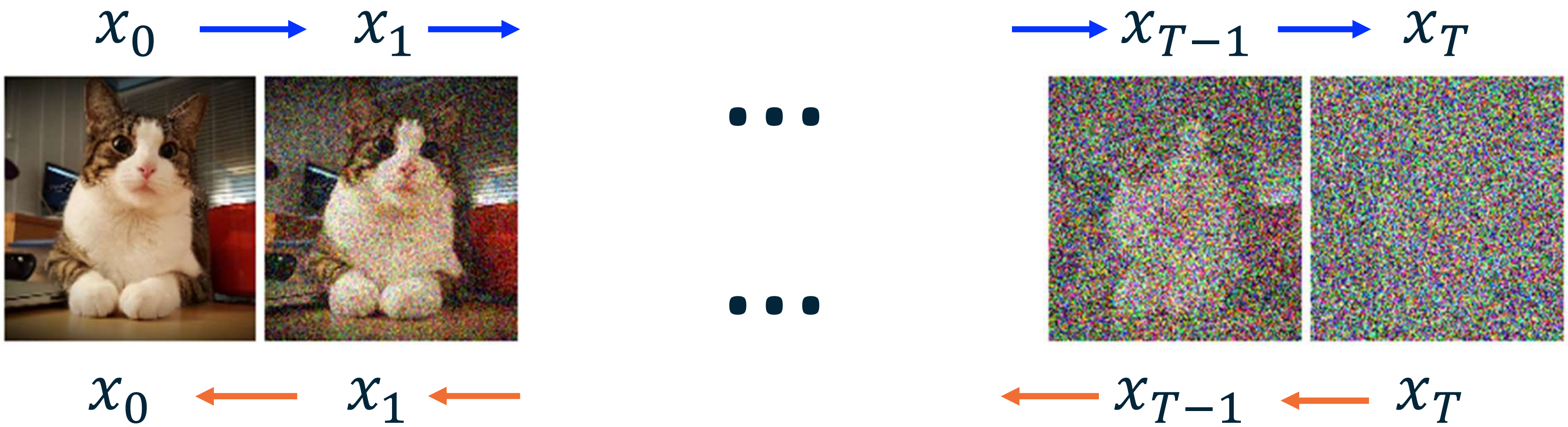
$$x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_{T-1} \longrightarrow x_T$$

The Denoising Diffusion Process

image from
dataset

The “forward diffusion” process:
add Gaussian noise each step

noise $\mathcal{N}(0, I)$



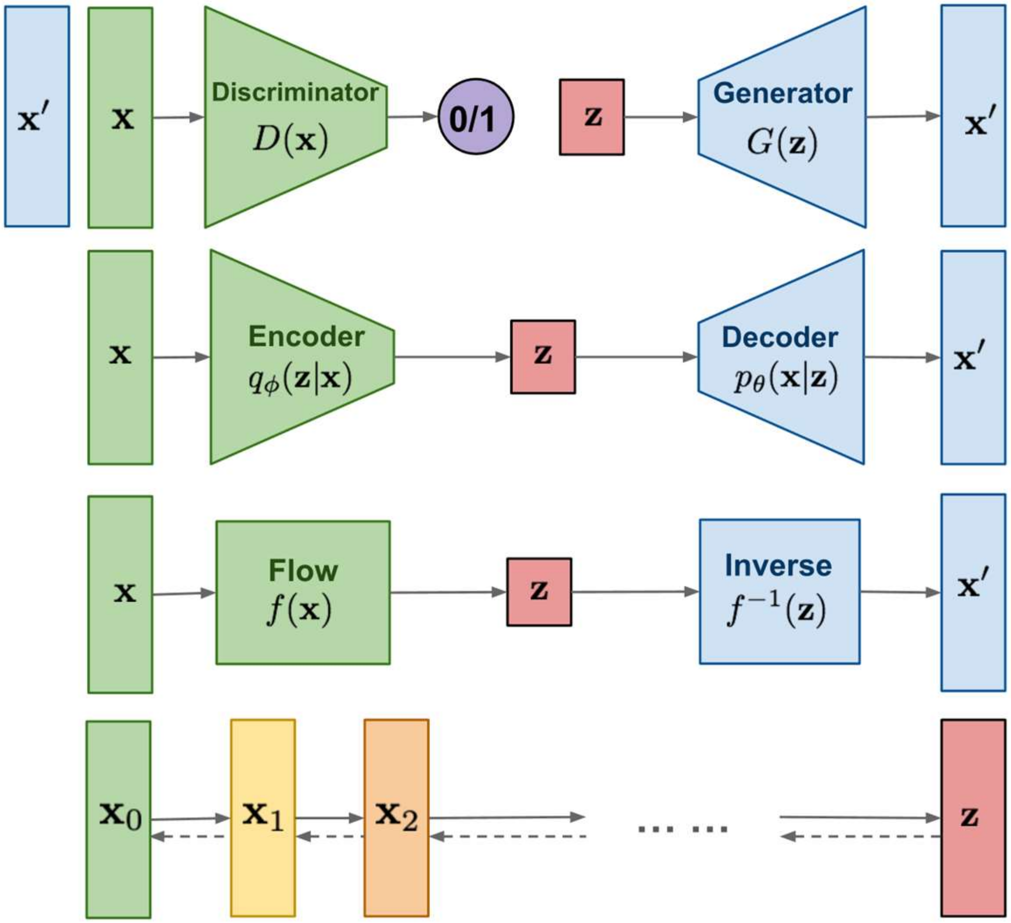
$x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{T-1} \rightarrow x_T$

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

The “denoising diffusion” process:
generate an image from noise by
denoising the gaussian noises

Ties/inspiration form Annealed
Imporantce Sampling in physics

Comparison



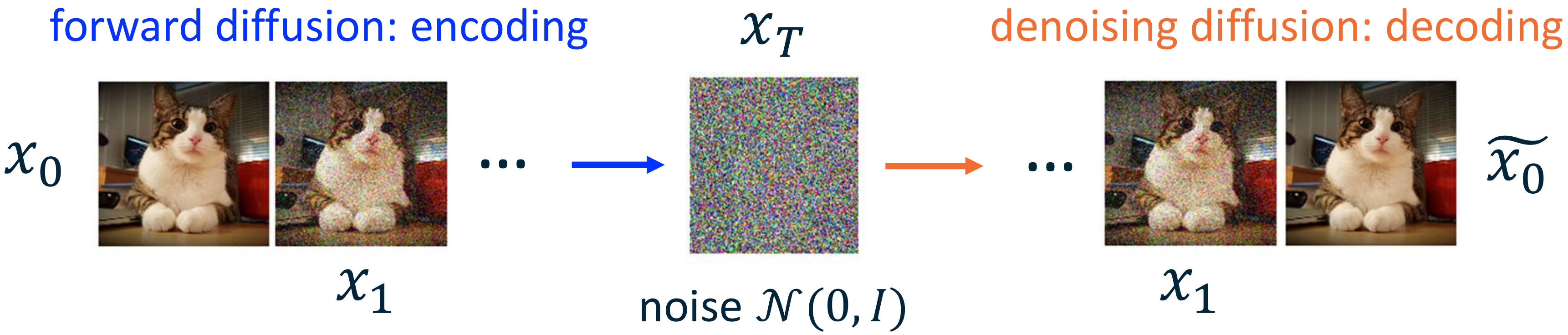
GAN: Adversarial training

VAE: maximize variational lower bound

Flow-based models: Invertible transform of distributions

Diffusion models: Gradually add Gaussian noise and then reverse

Connection to VAEs



forward diffusion: encoding

x_T

denoising diffusion: decoding

x_0 … $\rightarrow$ x_1

noise $\mathcal{N}(0, I)$

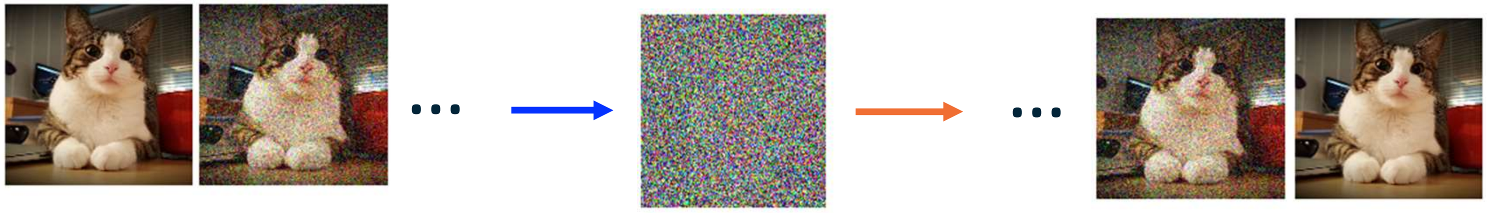
$\rightarrow$ … x_1 $\widetilde{x_0}$

Connection to VAEs

forward diffusion: encoding

x_T

denoising diffusion: decoding



x_0

x_1

noise $\mathcal{N}(0, I)$

x_1

$\widetilde{x_0}$

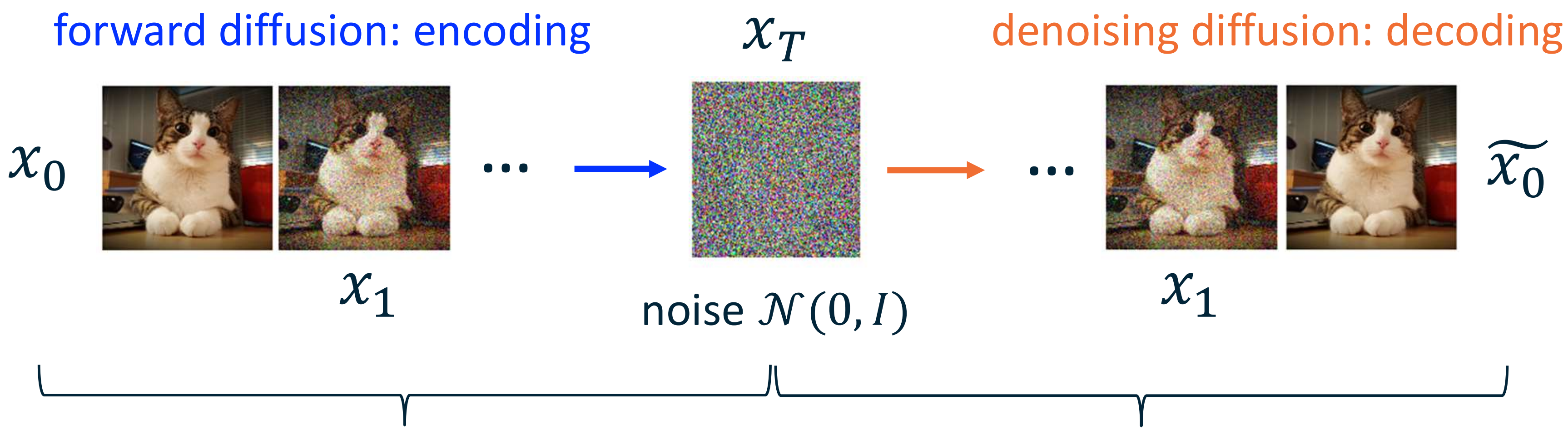
Known / predefined:

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

Unknown / learned:

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

Connection to VAEs



forward diffusion: encoding

x_T

denoising diffusion: decoding

x_0 x_1 … noise $\mathcal{N}(0, I)$ … x_1 $\widetilde{x_0}$

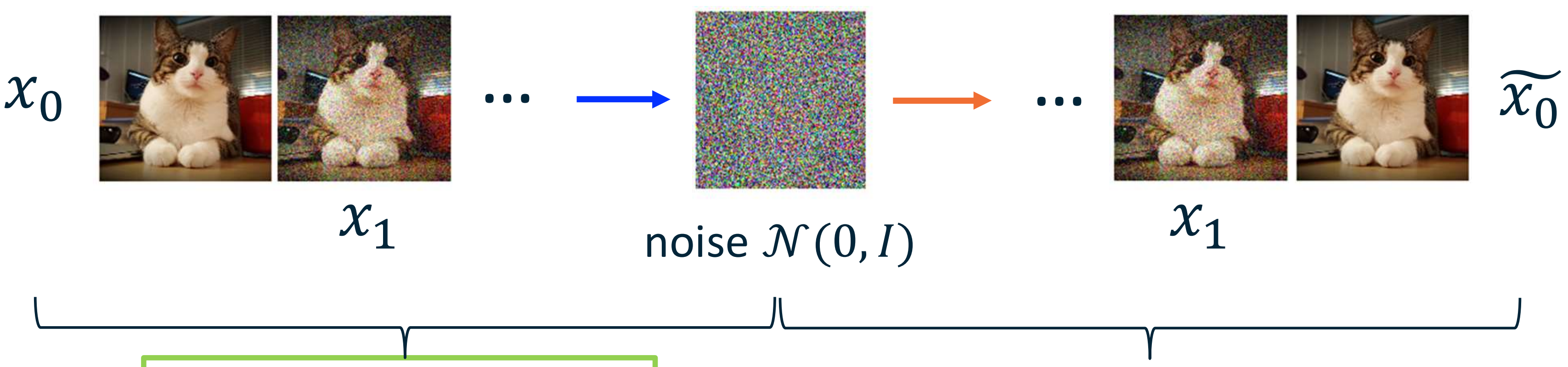
Known / predefined:
$q(x_{1:T}|x_0)$

Unknown / learned:

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

Similar to VAEs, use the denoising decoding process to generate new images.

Connection to VAEs



forward diffusion: encoding

x_T

denoising diffusion: decoding

x_0

x_1

noise $\mathcal{N}(0, I)$

x_1

$\widetilde{x_0}$

Known / predefined:
$q(x_{1:T}|x_0)$

Unknown / learned:

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

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

Probability Chain Rule (Markov Chain)

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t})x_{t-1}, \beta_t I)$$

Conditional Gaussian

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

Notation: A Gaussian distribution “for” x_t

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

β_t is the *variance schedule* at the diffusion step t

The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

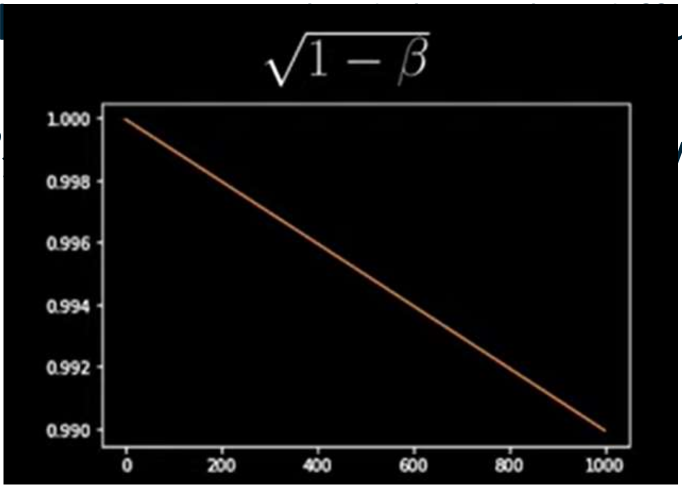
β_t is t

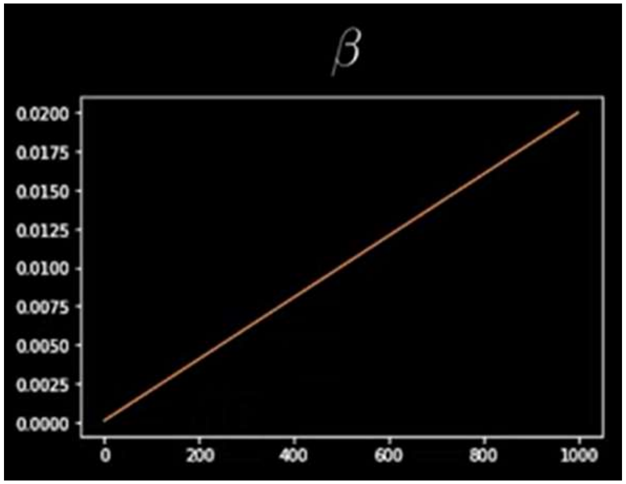
usion

$0 < \beta$

value

$= 1000$





https://www.youtube.com/watch?v=HoKDTa5jHvg&t=517s

The Diffusion (Encoding) Process

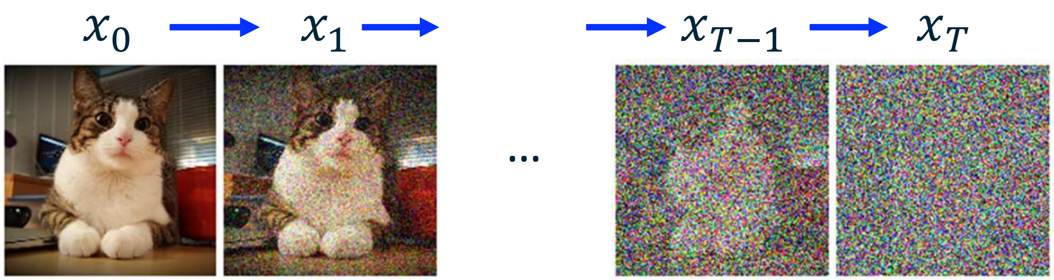
The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t})x_{t-1}, \beta_t I)$$ Conditional Gaussian

β_t is the *variance schedule* at the diffusion step t

$0 < \beta_1 < \beta_2 < \cdots < \beta_T < 1$, typical value range $[0.0001, 0.02]$, with $T = 1000$



The Diffusion (Encoding) Process

The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

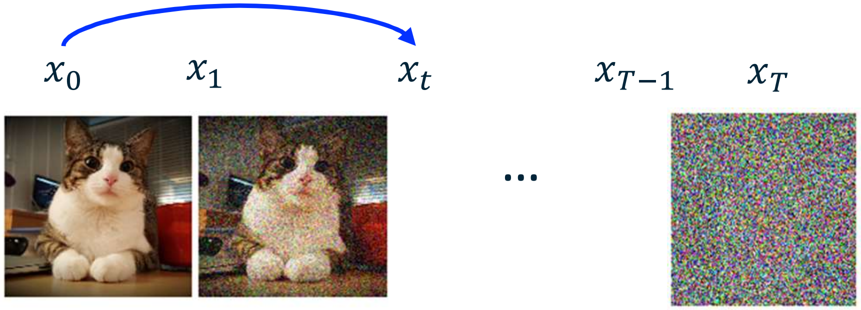
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian

Nice property: samples from an *arbitrary forward step* are also Gaussian-distributed!

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) I)$$

, where $\alpha_t = (1-\beta_t)$, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

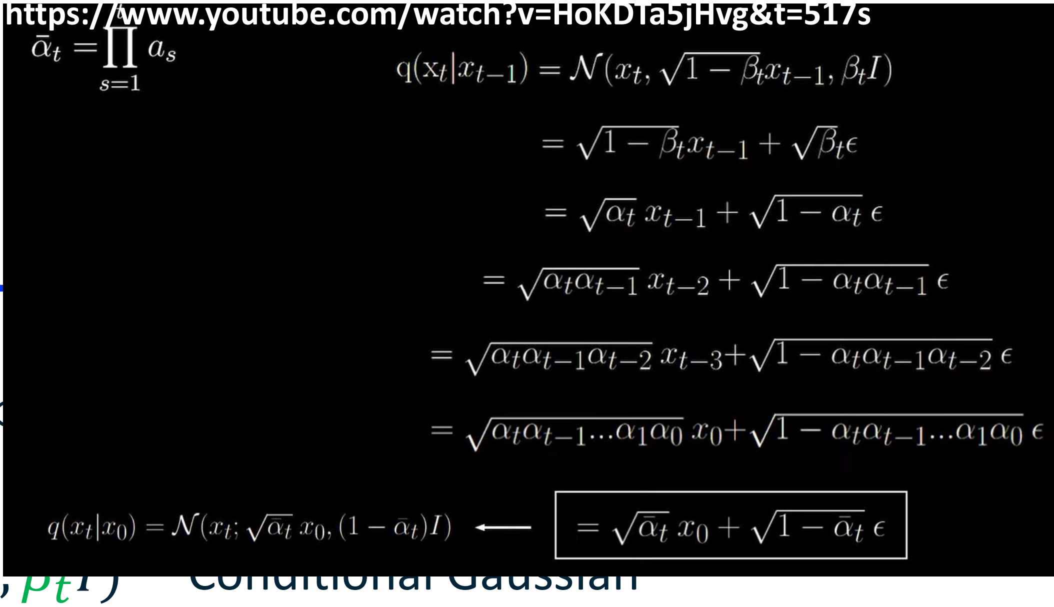


The Diffusion (Encoding

The **known** forward process x_0

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probab

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t}\ )x_{t-1}, \beta_t I)$$ Conditional Gaussian



https://www.youtube.com/watch?v=HoKDTa5jHvg&t=517s

$$\bar{\alpha}_t = \prod_{s=1} a_s$$

$$\begin{aligned} q(x_t|x_{t-1}) &= \mathcal{N}(x_t, \sqrt{1-\beta_t} x_{t-1}, \beta_t I) \\ &= \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t}\epsilon \\ &= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\,\epsilon \\ &= \sqrt{\alpha_t\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\,\epsilon \\ &= \sqrt{\alpha_t\alpha_{t-1}\alpha_{t-2}}\, x_{t-3} + \sqrt{1-\alpha_t\alpha_{t-1}\alpha_{t-2}}\,\epsilon \\ &= \sqrt{\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}\, x_0 + \sqrt{1-\alpha_t\alpha_{t-1}...\alpha_1\alpha_0}\,\epsilon \end{aligned}$$

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t)I) \longleftarrow \boxed{= \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}$$

Nice property: samples from an *arbitrary forward step* are also Gaussian-distributed!

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t)I)$$

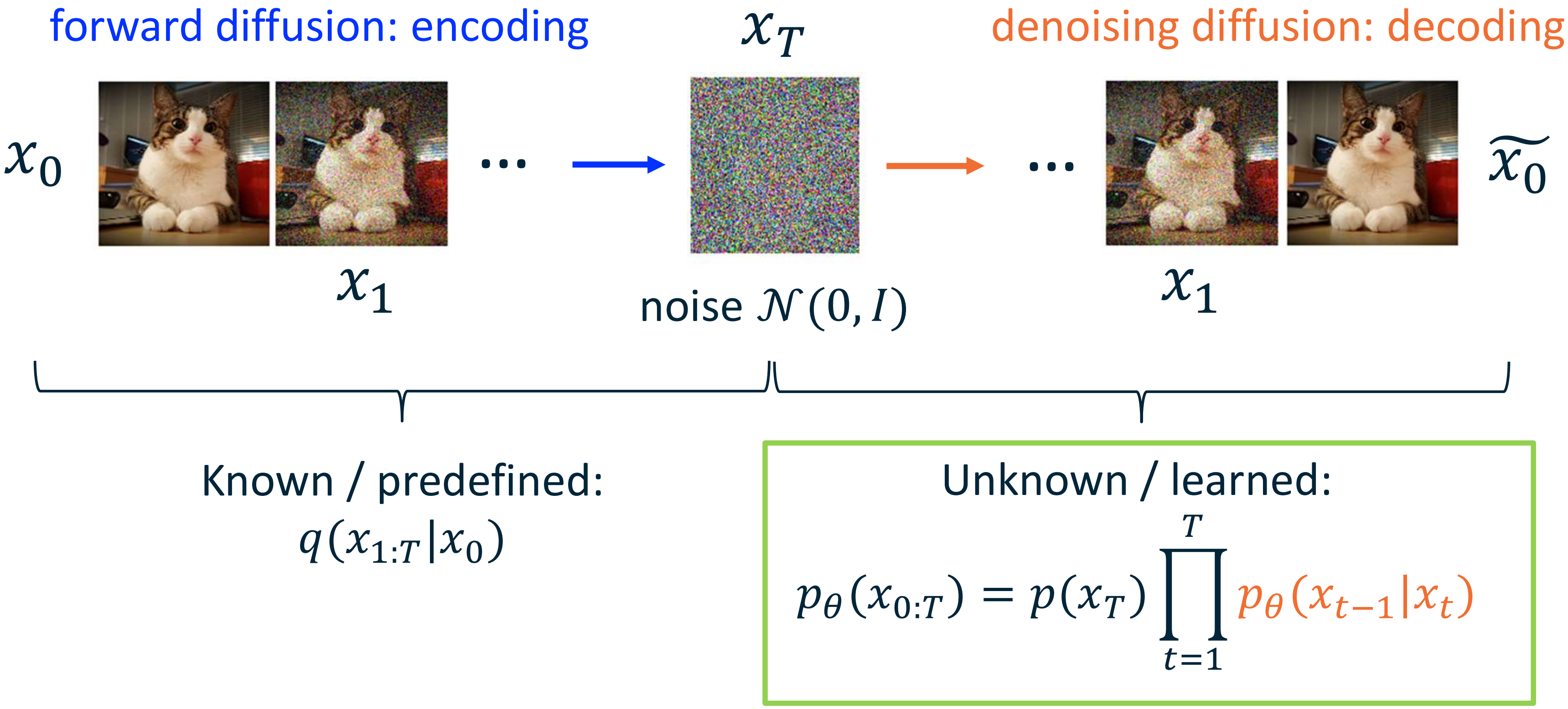
Gaussian reparameterization trick (recall from VAEs!):

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

(square root appears because reparameterization trick has just σ)

$$z = \mu + \epsilon * \sigma, \epsilon \sim N(0,1)$$

The Diffusion and Denoising Process



forward diffusion: encoding

x_T

denoising diffusion: decoding

x_0 x_1 … noise $\mathcal{N}(0, I)$ … x_1 $\widetilde{x_0}$

Known / predefined:

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

Unknown / learned:

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

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$ Probability Chain Rule (Markov Chain)

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$ Probability Chain Rule (Markov Chain)

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$ Conditional Gaussian

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

Probability Chain Rule (Markov Chain)

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

Conditional Gaussian

Want to learn time-dependent mean

Assume fixed / known variance (simplification)

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

Probability Chain Rule (Markov Chain)

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

Conditional Gaussian

Want to learn time-dependent mean

Assume fixed / known variance (simplification)

How do we form a learning objective?

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\text{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\text{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \mu_q(t), \Sigma_q(t)\right)$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I)$$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\text{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \mu_q(t), \Sigma_q(t)\right)$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I) \longleftarrow$$

Recall: Gaussian reparameterization trick

The "ground truth" noise that brought x_{t-1} to x_t

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\text{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \mu_q(t), \Sigma_q(t)\right)$

Assuming identical variance $\Sigma_q(t)$, we have:

$$\text{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0)\big|\big|p_\theta(x_{t-1}|x_t)\big) = \text{argmin}_\theta w||\mu_q(t) - \mu_\theta(x_t, t)||^2$$

Should be variance-dependent, but constant works better in practice

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

The learning objective: $\mathrm{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$

What does it look like? $q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \mu_q(t), \Sigma_q(t)\right)$

Assuming identical variance $\Sigma_q(t)$, we have:

$$\mathrm{argmin}_\theta D_{KL}(q(x_{t-1}|x_t, x_0)\big|\big|p_\theta(x_{t-1}|x_t)\big) = \mathrm{argmin}_\theta w||\mu_q(t) - \mu_\theta(x_t, t)||^2$$

Simplified learning objective: $\boldsymbol{\mathrm{argmin}_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2}$

Predict the one-step noise that was added (and remove it)!

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

Probability Chain Rule (Markov Chain)

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

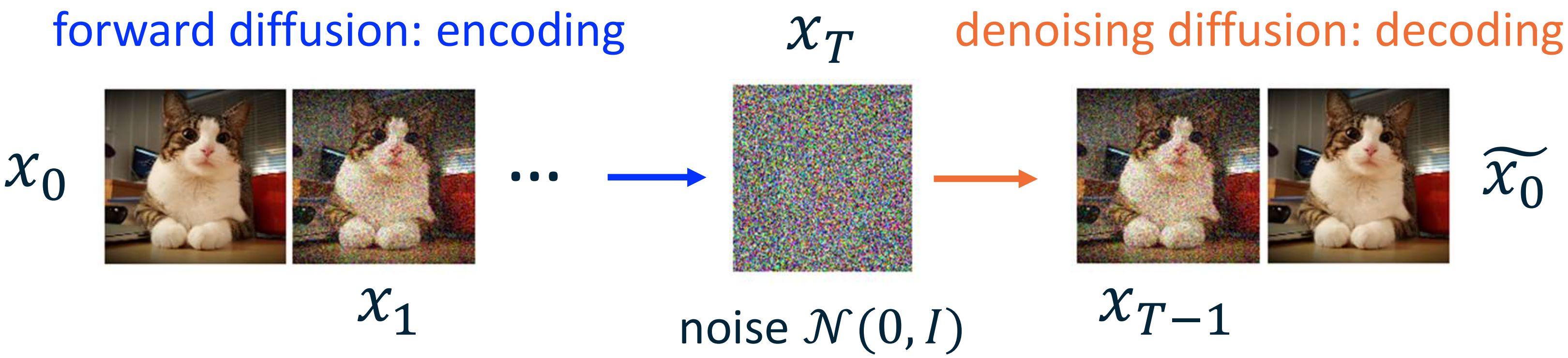
Conditional Gaussian

Assume fixed / known variance

How did we arrive at the learning objective?
Let's go back to the basics of variational models …

(Quick) Derivation!

Connection to VAEs



forward diffusion: encoding

x_T

denoising diffusion: decoding

x_0 …

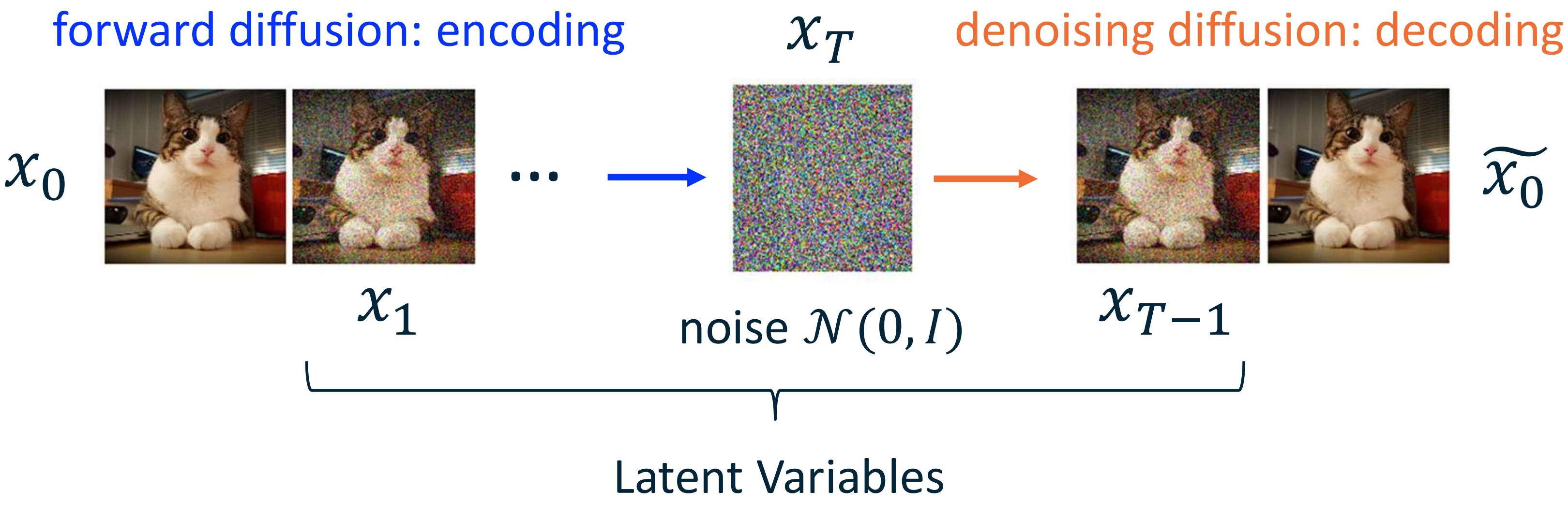
x_1

noise $\mathcal{N}(0, I)$

x_{T-1}

$\widetilde{x_0}$

Connection to VAEs



forward diffusion: encoding

x_T

denoising diffusion: decoding

x_0

…

$\widetilde{x_0}$

x_1

noise $\mathcal{N}(0, I)$

x_{T-1}

Latent Variables

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization** → **Remove (variance-dependent) constant** → **Predict the noise!!!**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}\big(q(z|x)\big|\big|p(z|x)\big)$$

$$\geq \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right]$$

Evidence Lower Bound (ELBO) – **From last lecture on VAEs**

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right]$$ Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$ $x = x_0,\ z = x_{1:T}$

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right]$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right] \qquad x = x_0,\ z = x_{1:T}$$

… (derivation omitted, see Sohl-Dickstein *et al.,* 2015 Appendix B)

$$= \mathrm{E}_q\left[\log \frac{p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T} q(x_t|x_{t-1})}\right]$$

← reverse denoising

← forward diffusion

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL**

$$p(x) = \int p(x|z)p(z)dz$$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}\big(q(z|x)\big|\big|p(z|x)\big)$$

$$\geq \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right]$$ Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$ $x = x_0,\ z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}\big(q(z|x)||p(z|x)\big)$$

$$\geq \mathrm{E}_q\left[\log \frac{p(x|Z)p(z)}{q(Z|x)}\right]$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$x = x_0, \; z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}\big(q(x_T|x_0)||p(x_T)\big)] - \sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log \frac{p(X|Z)p(z)}{q(Z|X)}\right]$$

Evidence Lower Bound (ELBO)

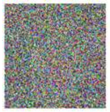
$$\log p(x_0) \geq \mathrm{E}_q\left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$x = x_0,\ z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))] - \sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

fixed



Easy to optimize / sometimes omitted

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right]$$

Evidence Lower Bound (ELBO)

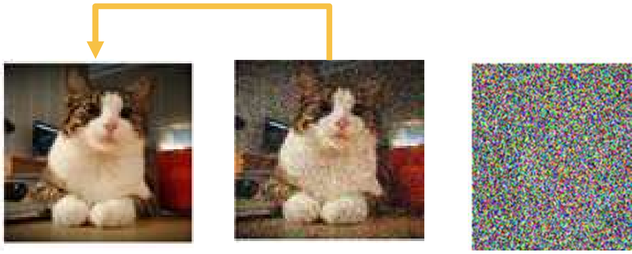
$$\log p(x_0) \geq \mathrm{E}_q\left[\log\frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$x = x_0,\ z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.,* 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))]\boxed{-\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))} + \log p_\theta(x_0|x_1)$$

Maximize the agreement between the predicted reverse diffusion distribution p_θ and the “ground truth” reverse diffusion distribution q



Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL** → **Reverse Process => Normal**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right]$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log\frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$x = x_0,\ z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))] - \sum_{t=2}^{T} D_{KL}(\boxed{q(x_{t-1}|x_t, x_0)}||p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL** → **Reverse Process => Normal**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log\frac{p(X|Z)p(z)}{q(Z|X)}\right]$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log\frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$x = x_0,\ z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))] - \sum_{t=2}^{T} D_{KL}(\boxed{q(x_{t-1}|x_t, x_0)}||p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

$$q(x_{t-1}|x_t) = \boldsymbol{q(x_{t-1}|x_t, x_0)} \quad \text{(markov assumption)}$$

$$= \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_{t|}x_0)} \quad \text{(Bayes rule)}$$

$$= \frac{\mathcal{N}(x_t;\sqrt{\alpha_t}x_{t-1},\beta_t I)\mathcal{N}(x_{t-1};\sqrt{\bar{\alpha}_{t-1}}x_{t-1},(1-\bar{\alpha}_{t-1})I)}{\mathcal{N}(x_t;\sqrt{\bar{\alpha}_t}x_0,(1-\bar{\alpha}_{t-1})I)}$$

$$\propto \mathcal{N}\left(x_{t-1};\ \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t+\sqrt{\bar{\alpha}_{t-1}}(1-a_t)x_0}{1-\sqrt{\bar{\alpha}_t}}, \Sigma_q(t)\right) \quad \text{(Property of Gaussian)}$$

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right]$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log \frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$x = x_0,\ z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))] - \sum_{t=2}^{T} D_{KL}(\boxed{q(x_{t-1}|x_t, x_0)}||p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

$$\boxed{\begin{gathered} q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \mu_q(t), \Sigma_q(t)\right) \\ \mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I) \end{gathered}}$$

Proof using bayes rule and gaussian reparameterization trick

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization**

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right]$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log\frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$x = x_0,\ z = x_{1:T}$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))] - \sum_{t=2}^{T} D_{KL}(\boxed{q(x_{t-1}|x_t,x_0)}||p_\theta(x_{t-1}|x_t)) + \log p_\theta(x_0|x_1)$$

$$\boxed{\begin{aligned} q(x_{t-1}|x_t,x_0) &= \mathcal{N}\left(x_{t-1};\mu_q(t),\Sigma_q(t)\right) \\ \mu_q(t) &= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon\sim\mathcal{N}(0,I) \end{aligned}}$$

Proof using bayes rule and gaussian reparameterization trick

The “ground truth” noise that brought x_{t-1} to x_t

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → Simplify to KL → Reverse Process => Normal → Bayes + Reparameterization → Remove (variance-dependent) constant

$p(x) = \int p(x|z)p(z)dz$ Intractable to estimate!

$$\log p(x) = \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right] + D_{KL}(q(z|x)||p(z|x))$$

$$\geq \mathrm{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right]$$

Evidence Lower Bound (ELBO)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log\frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$

$$x = x_0,\ z = x_{1:T}$$

… (derivation omitted, see Sohl-Dickstein *et al.*, 2015 Appendix B)

$$= -\mathrm{E}_q[D_{KL}(q(x_T|x_0)||p(x_T))] - \boxed{\sum_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))} + \log p_\theta(x_0|x_1)$$

Minimize the difference of distribution means (assuming identical variance)

$$\mathrm{argmin}_\theta w||\mu_q(t) - \mu_\theta(x_t, t)||^2$$

Deep Unsupervised Learning using Nonequilibrium Thermodynamics, Sohl-Dickstein *et al.*, 2015

Variational Inference → Simplify to KL → Reverse Process => Normal → Bayes + Reparameterization → Remove (variance-dependent) constant → Predict the noise!!!

Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$

Conditional Gaussian

Learning objective: $\text{argmin}_\theta ||\mu_q(t) - \mu_\theta(x_t, t)||^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I)$$

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization** → **Remove (variance-dependent) constant** → **Predict the noise!!!**

Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$ Conditional Gaussian

Learning objective: $\text{argmin}_\theta ||\mu_q(t) - \mu_\theta(x_t, t)||^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I)$$

Do we actually need to learn the entire $\mu_\theta(x_t, t)$?

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization** → **Remove (variance-dependent) constant** → **Predict the noise!!!**

Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$ Conditional Gaussian

Learning objective: $\text{argmin}_\theta ||\mu_q(t) - \mu_\theta(x_t, t)||^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I)$$

known during inference

Unknown during inference

Recall: this is the “ground truth” noise that brought x_{t-1} to x_t

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization** → **Remove (variance-dependent) constant** → **Predict the noise!!!**

Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$

Conditional Gaussian

Learning objective: $\mathrm{argmin}_\theta ||\mu_q(t) - \mu_\theta(x_t, t)||^2$

$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I)$$

known during inference

Unknown during inference

Recall: this is the "ground truth" noise that brought x_{t-1} to x_t

Idea: just learn ϵ with $\epsilon_\theta(x_t, t)$!

Variational Inference → Simplify to KL → Reverse Process => Normal → Bayes + Reparameterization → Remove (variance-dependent) constant → Predict the noise!!!

Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$ Conditional Gaussian

Simplified learning objective: $\mathrm{argmin}_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2$

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization** → **Remove (variance-dependent) constant** → **Predict the noise!!!**

Learning the Denoising Process

The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$

Conditional Gaussian

Simplified learning objective: $\text{argmin}_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2$

Recall: the simplified t-step forward sample:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$$

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization** → **Remove (variance-dependent) constant** → **Predict the noise!!!**

Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$

Conditional Gaussian

Simplified learning objective: $\mathrm{argmin}_\theta ||\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)||^2$

Recall: the simplified t-step forward sample:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

Variational Inference → **Simplify to KL** → **Reverse Process => Normal** → **Bayes + Reparameterization** → **Remove (variance-dependent) constant** → **Predict the noise!!!**

Learning the Denoising Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$ Conditional Gaussian

Simplified learning objective: $\mathrm{argmin}_\theta ||\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t\right)||^2$

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \dots \leftarrow x_T$

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$ Probability Chain Rule (Markov Chain)

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$ Conditional Gaussian

We know how to learn

Assume fixed / known variance

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

The Denoising (Decoding) Process

The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

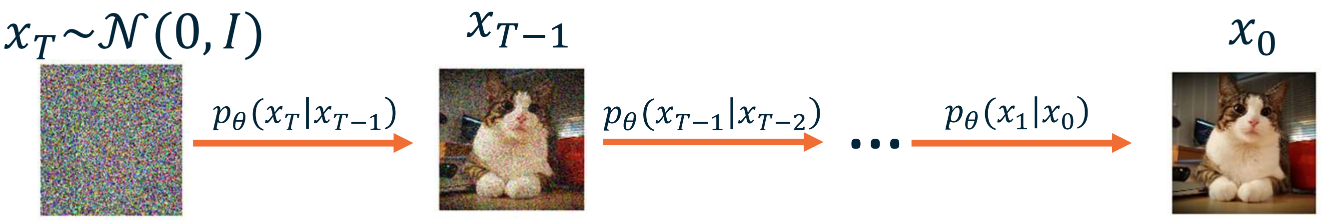
Probability Chain Rule (Markov Chain)

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

Conditional Gaussian

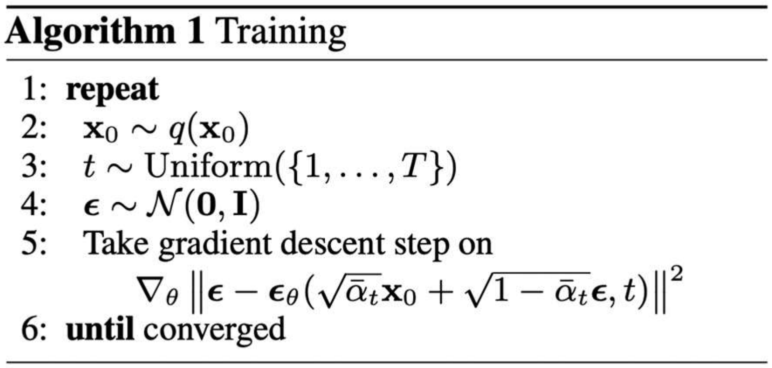
We know how to learn

Assume fixed / known variance



Generate new images!

The Denoising Diffusion Algorithm

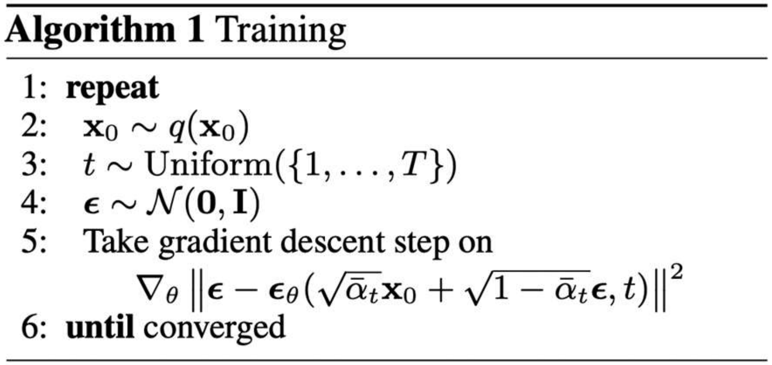


Algorithm 1 Training

```
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
```

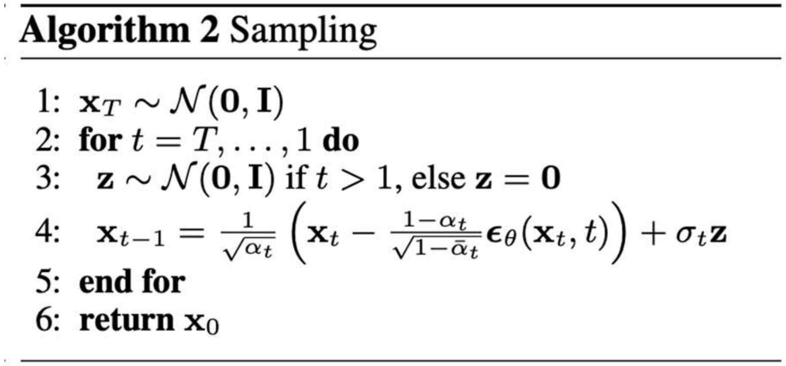
The Denoising Diffusion Probabilistic Models, Ho *et al.*, 2020

The Denoising Diffusion Algorithm



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
```

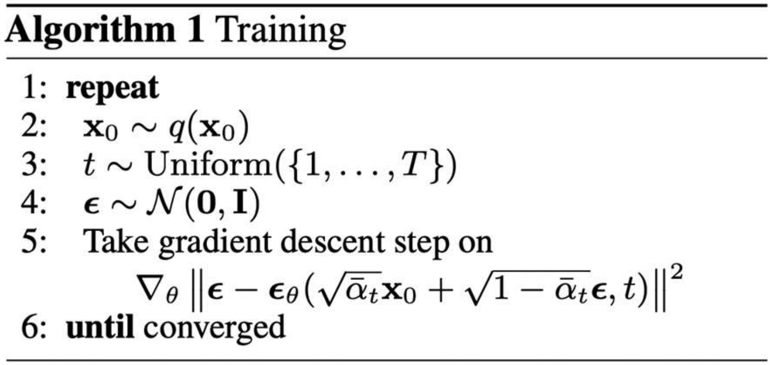


Algorithm 2 Sampling

```
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
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 $\mathbf{x}_0$
```

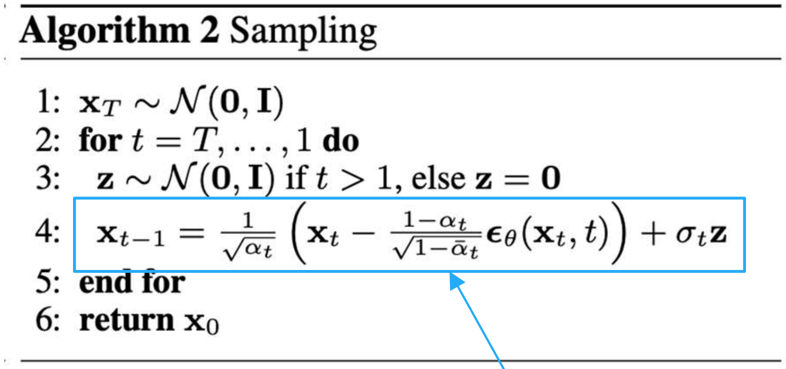
The Denoising Diffusion Probabilistic Models, Ho *et al.*, 2020

The Denoising Diffusion Algorithm



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
```



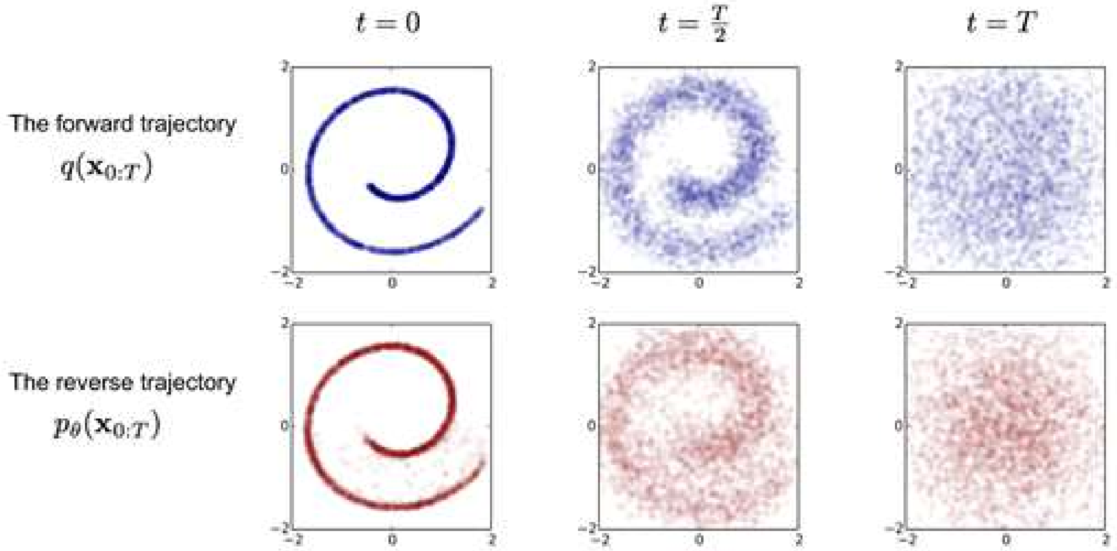
Algorithm 2 Sampling

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$

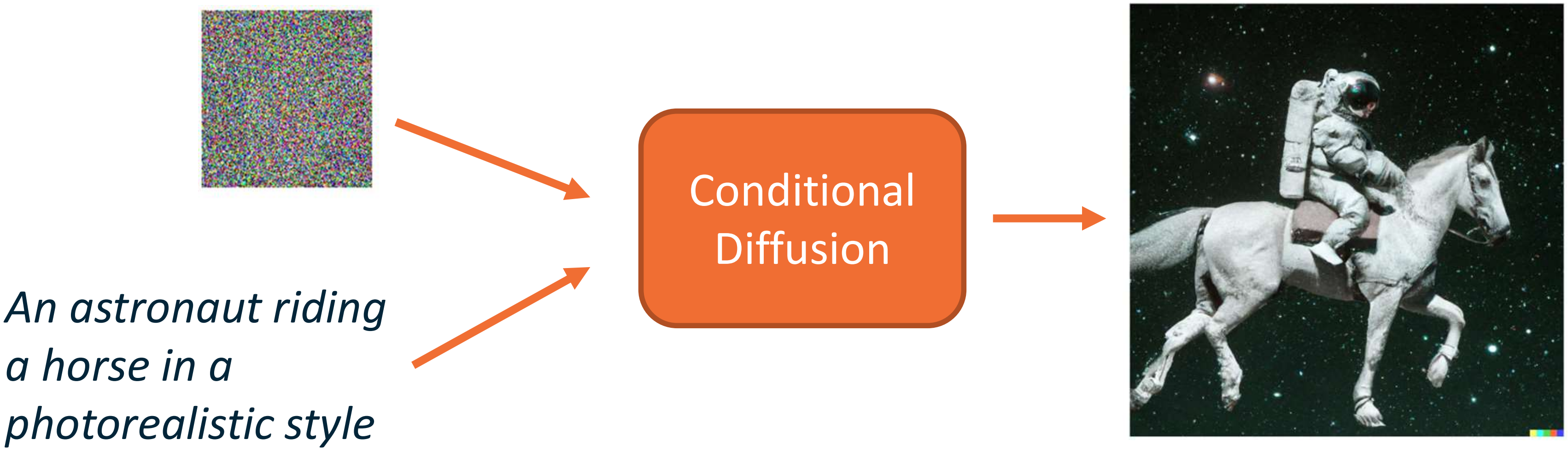
The Denoising Diffusion Probabilistic Models, Ho *et al.*, 2020

Visualizing the Diffusion Process on 2D data



Sohl-Dickstein *et al.*, 2015

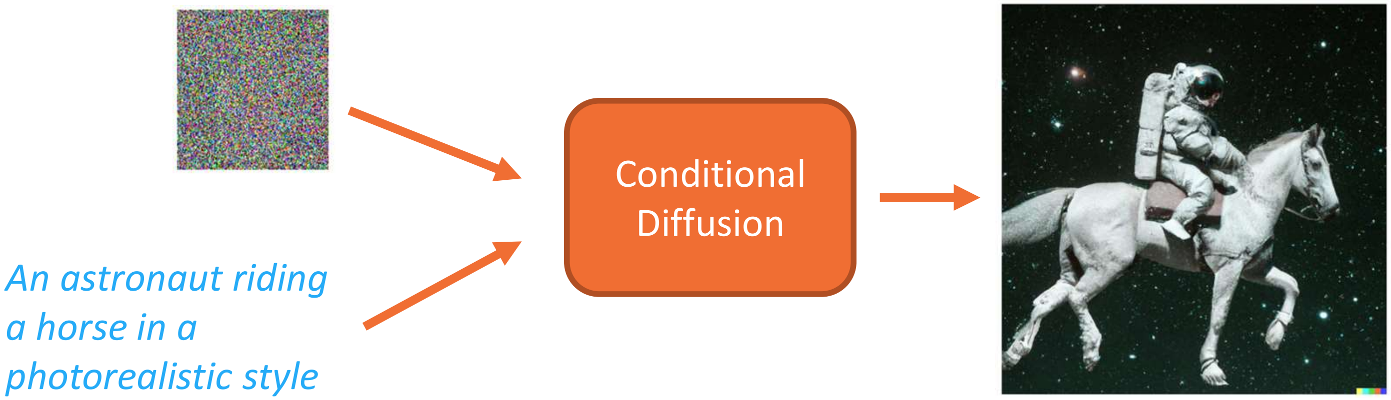
Conditional Diffusion Models



An astronaut riding a horse in a photorealistic style

Conditional Diffusion

Conditional Diffusion Models



An astronaut riding a horse in a photorealistic style

Conditional Diffusion

Simple idea: just condition the model on some text labels y!

$$\epsilon_\theta(x_t, y, t)$$