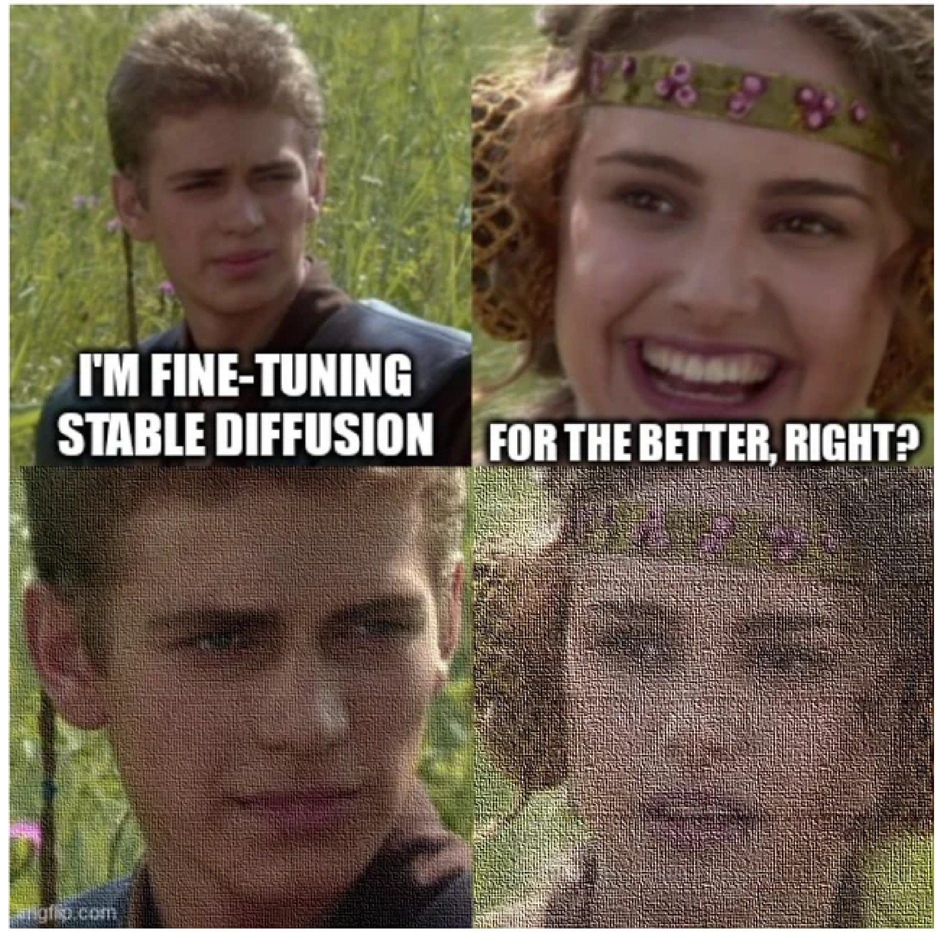
# Diffusion Models


Slides adapted from

Zsolt Kira (who adapted them from

Danfei Xu and

Roni Sengupta (who partially adapted them from

Sander Dieleman

)

)

Also see:

CVPR 2022 Tutorial on Diffusion Models: https://cvpr2022-tutorial-diffusion-models.github.io/

# Taxonomy of Generative Models

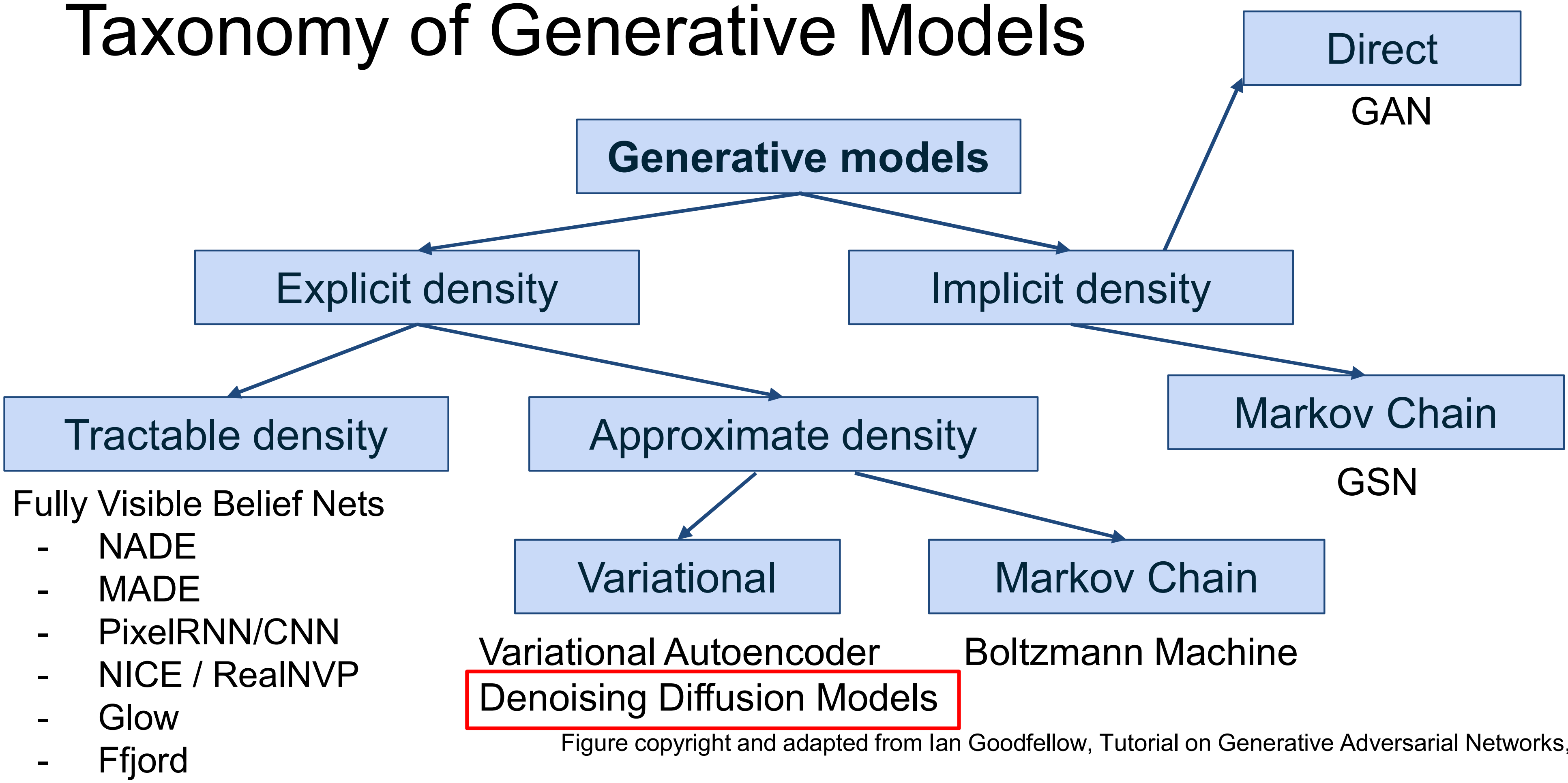



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# 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

→

DALL·E 2

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

TEXT DESCRIPTION

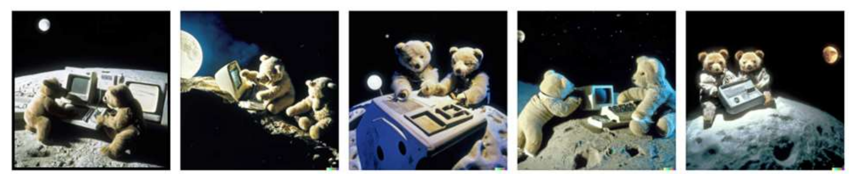
An astronaut Teddy bears A bowl of soup

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

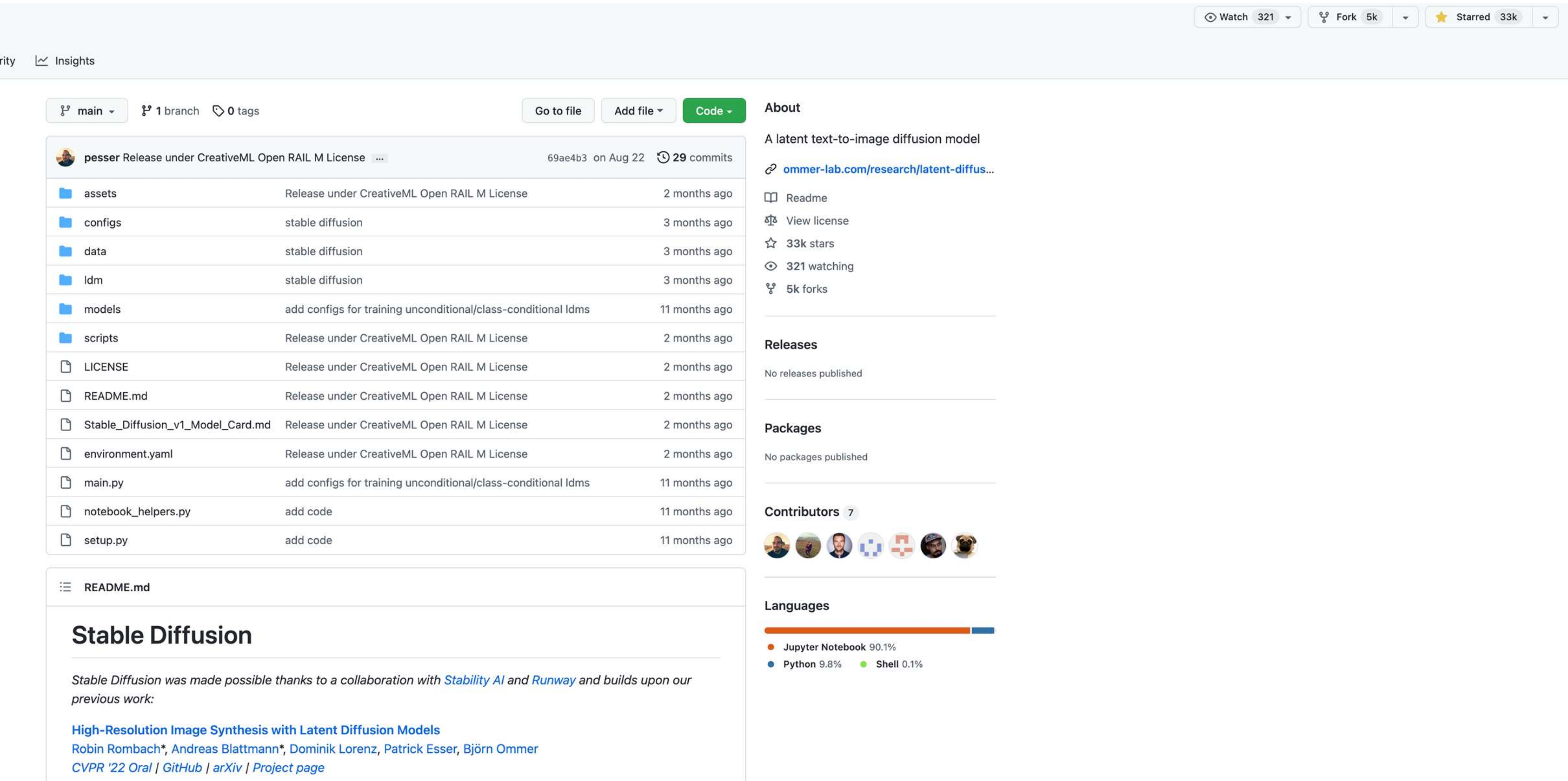
→

DALL·E 2



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

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




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

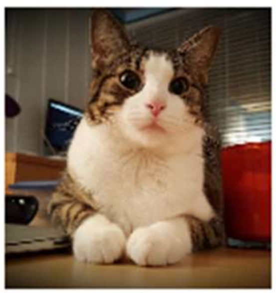
$x_0$

# The Denoising Diffusion Process

image from
dataset

The "forward diffusion" process:
add Gaussian noise each step

$x_0 \longrightarrow x_1 \longrightarrow$



$\cdots$

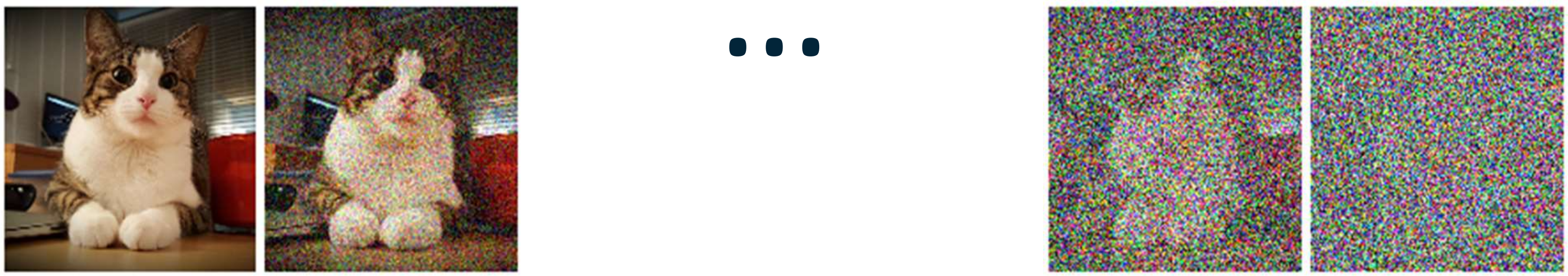
# The Denoising Diffusion Process

image from dataset

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

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

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

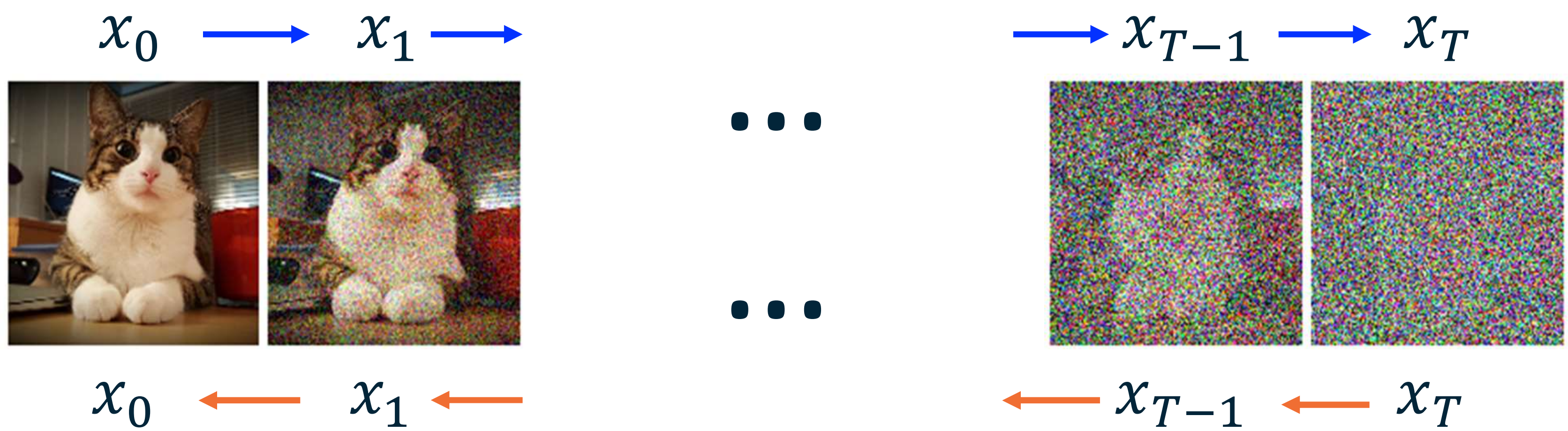


# The Denoising Diffusion Process

image from dataset

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

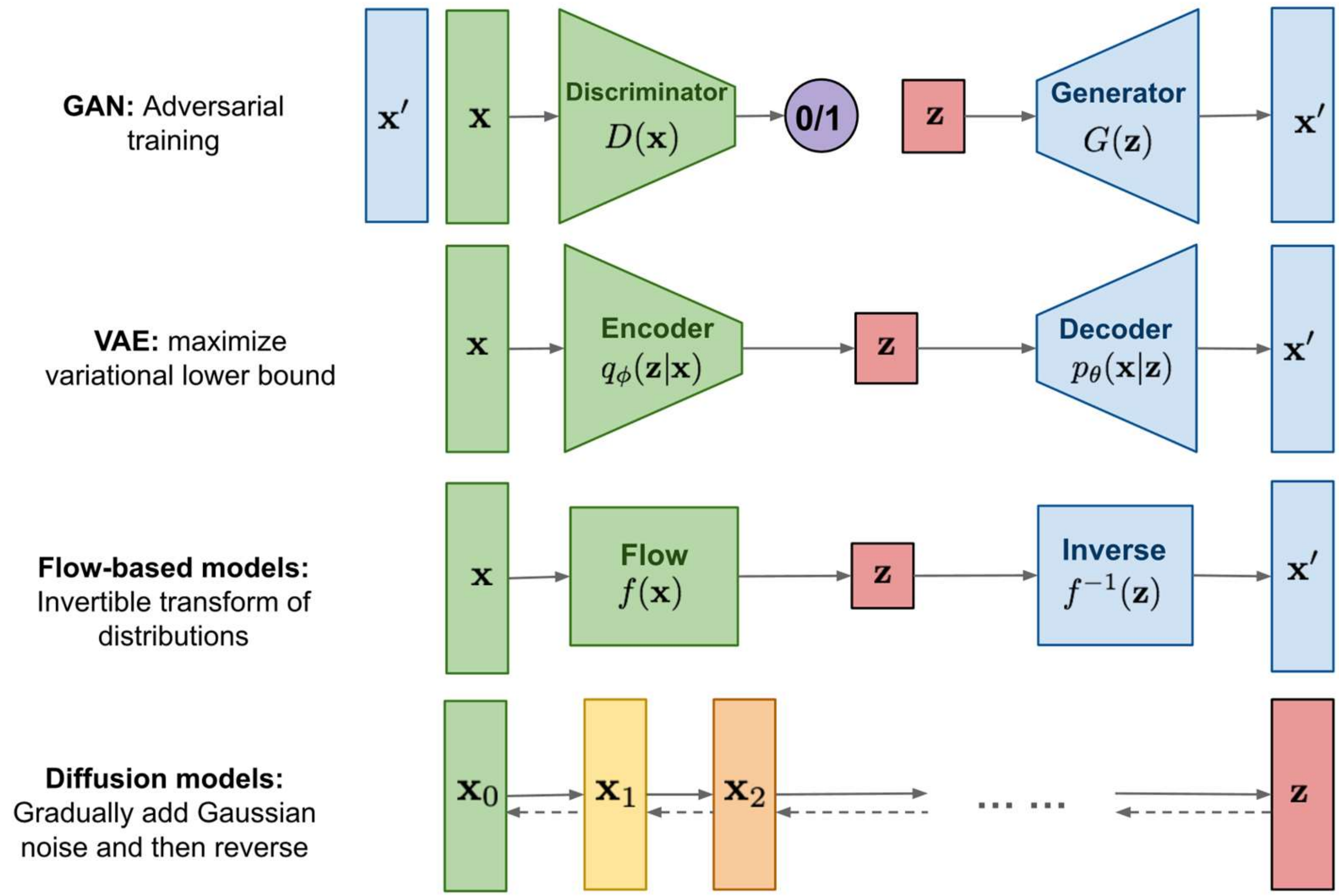
noise $\mathcal{N}(0, I)$




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

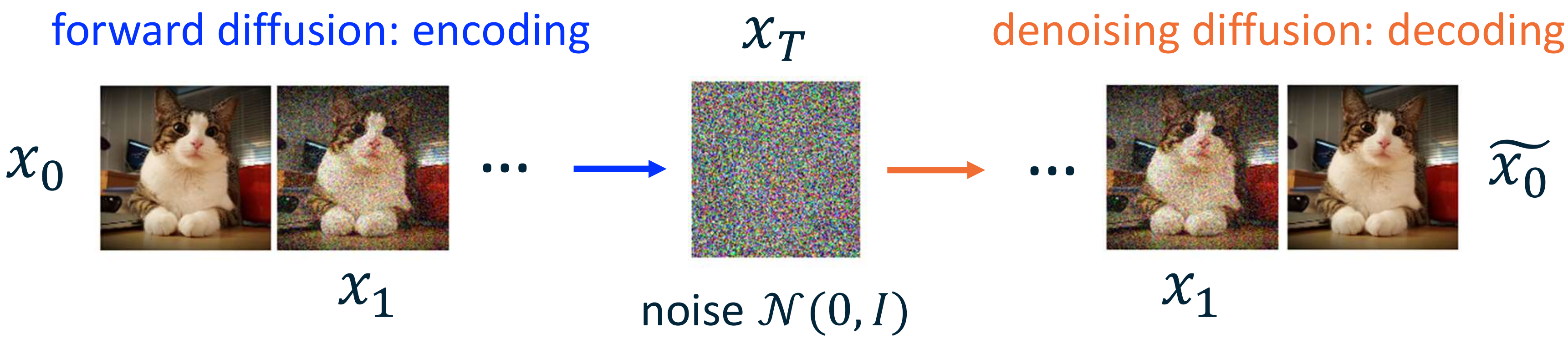
Ties/inspiration form Annealed Imporantce Sampling in physics

# Comparison



GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based models: Invertible transform of distributions
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
$\mathbf{z}$

# Connection to VAEs



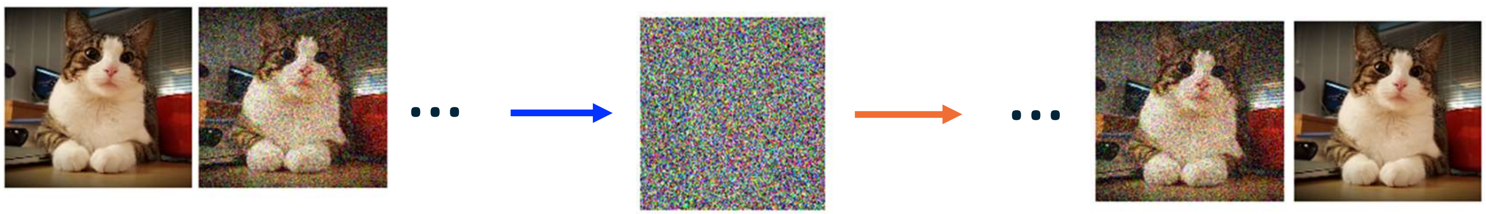
# Connection to VAEs

forward diffusion: encoding

$x_T$

denoising diffusion: decoding

$x_0$



$\widetilde{x_0}$

$x_1$

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

$x_1$

Known / predefined:

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

Unknown / learned:

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

$x_t$


$x_{t-1}$

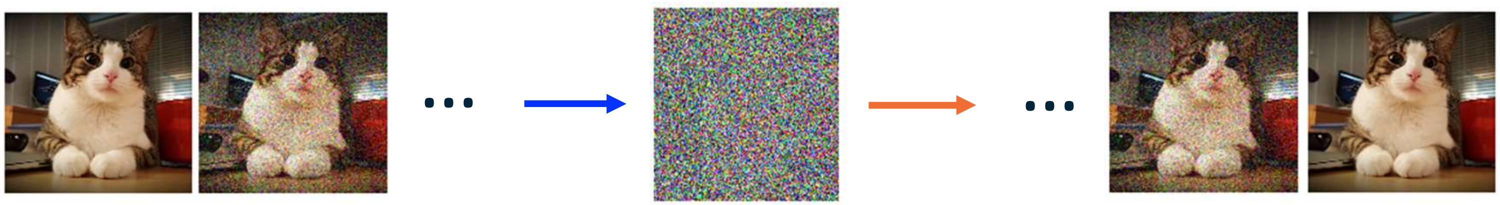
# Connection to VAEs

forward diffusion: encoding

$x_T$

denoising diffusion: decoding

$x_0$



$\widetilde{x_0}$

$x_1$

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

$x_1$

Known / predefined:

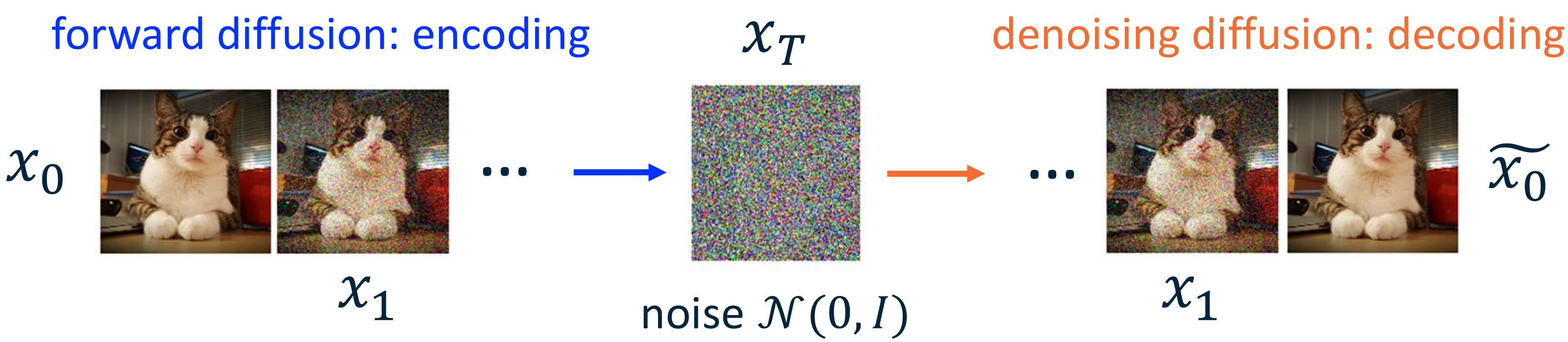
$q(x_{1:T}|x_0)$

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




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

$\beta_t$ is the *variance schedule* at the diffusion step $t$

# The Diffusion (Encoding) Process

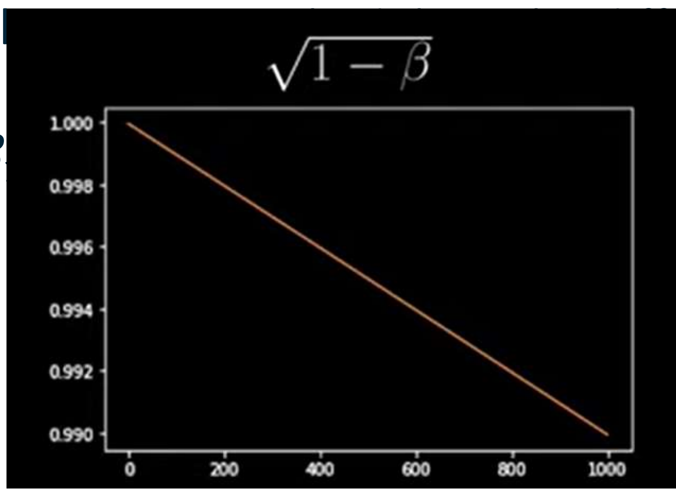
The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

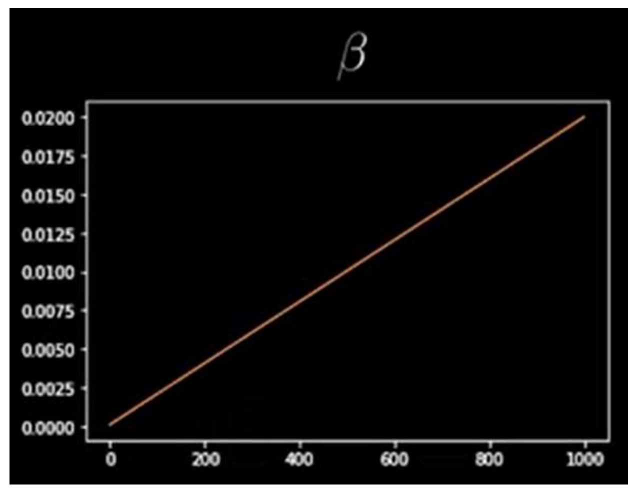
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

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

$\beta_t$ 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

$\beta_t$ 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)

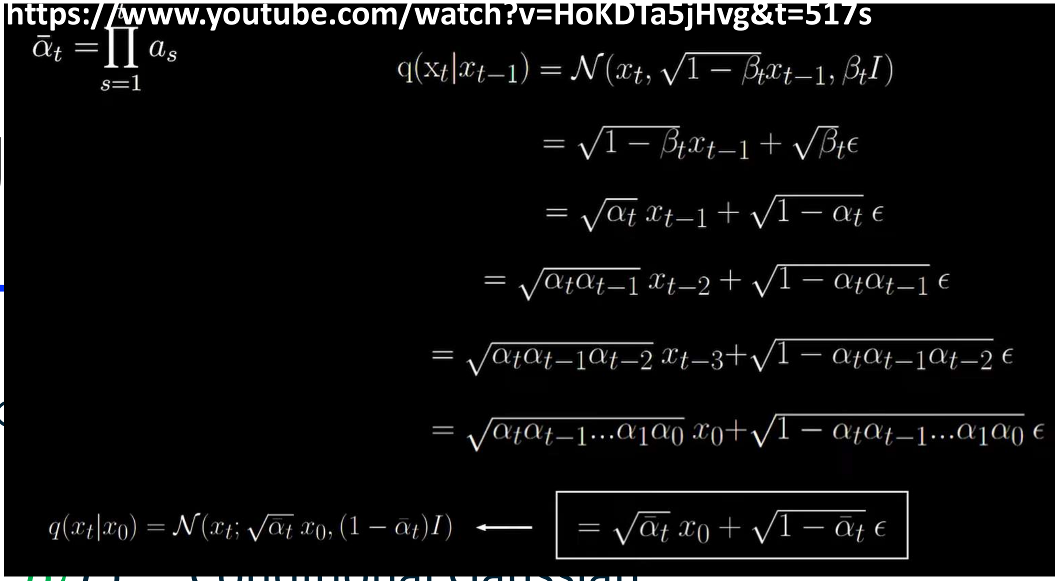
$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (\sqrt{1-\beta_t})x_{t-1}, \beta_t I)$$ Conditional Gaussian

**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

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

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

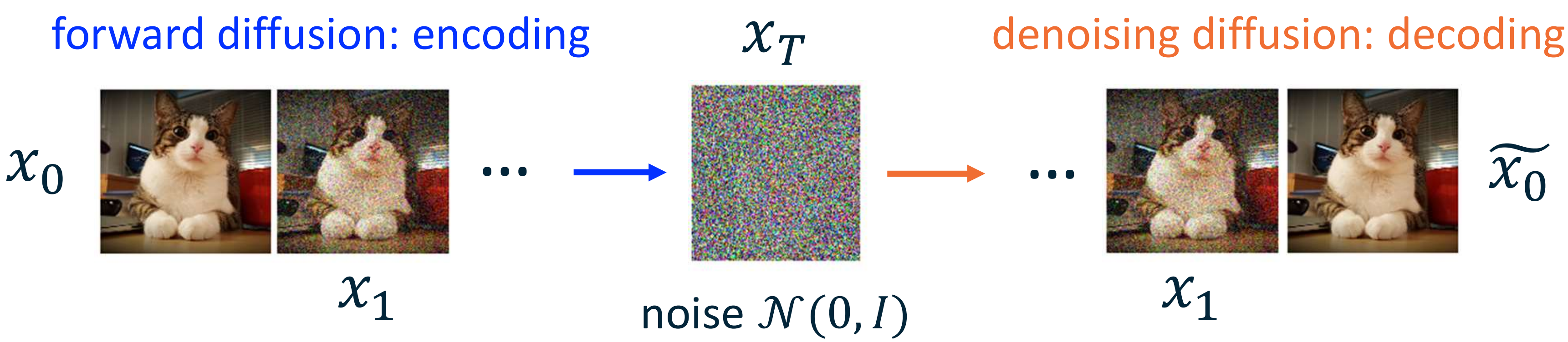
**Gaussian reparameterization trick** (recall from VAEs!):

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

(square root appears because reparameterization trick has just $\sigma$)

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

# The Diffusion and Denoising Process




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 $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

# 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 $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$ 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 $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

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

# The Denoising (Decoding) Process

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

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

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

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

# The Denoising (Decoding) Process

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

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

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

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

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:

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

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

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:

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

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

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)

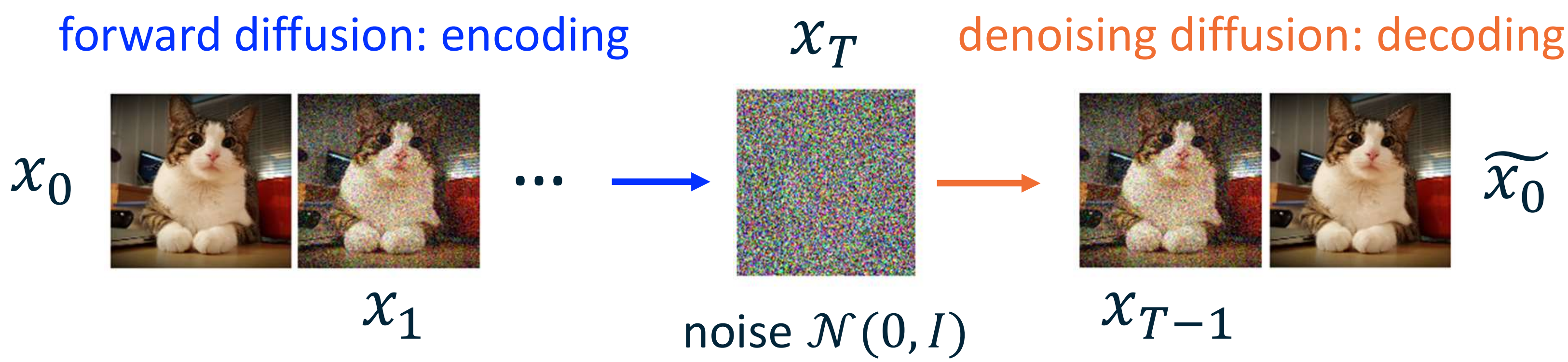
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$ Conditional Gaussian

Assume fixed / known variance

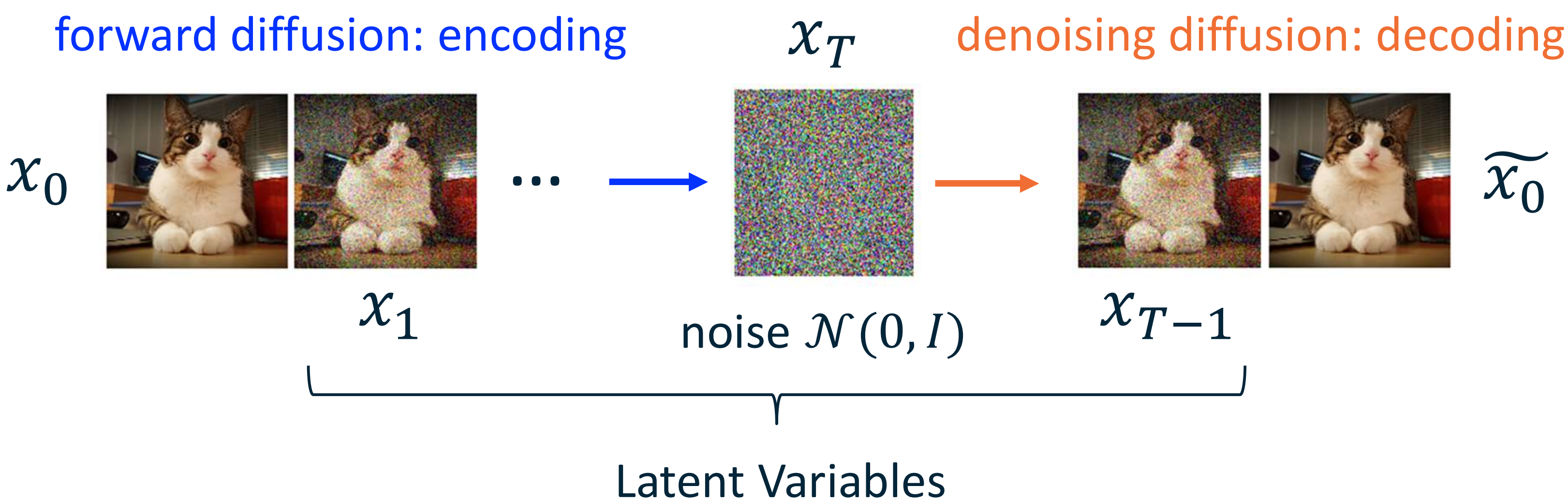
How did we arrive at the learning objective?
Let's go back to the basics of variational models …

# (Quick) Derivation!

# Connection to VAEs



# Connection to VAEs



**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!

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

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

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)

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

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

… (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!

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

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

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)

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

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)

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

... (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!

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

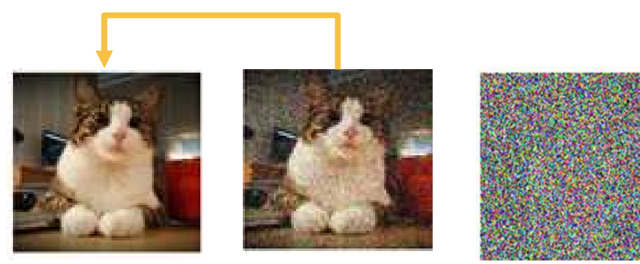
$$\geq \mathrm{E}_q\left[\log \frac{p(x|z)p(z)}{q(z|x)}\right]$$ Evidence Lower Bound (ELBO)

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

… (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 $p_\theta$ 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)

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

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1}; \mu_q(t), \Sigma_q(t)\right)$$

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

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)

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

… (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

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

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

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

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

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

$$\mu_q(t) = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\right.}_{\text{known during inference}}\underbrace{\epsilon}_{\text{Unknown during inference}}\Bigg), \qquad \epsilon \sim \mathcal{N}(0, I)$$

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

Idea: just learn $\epsilon$ 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: $\text{argmin}_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2$

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

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: $\text{argmin}_\theta ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)||^2$

# 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

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 $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

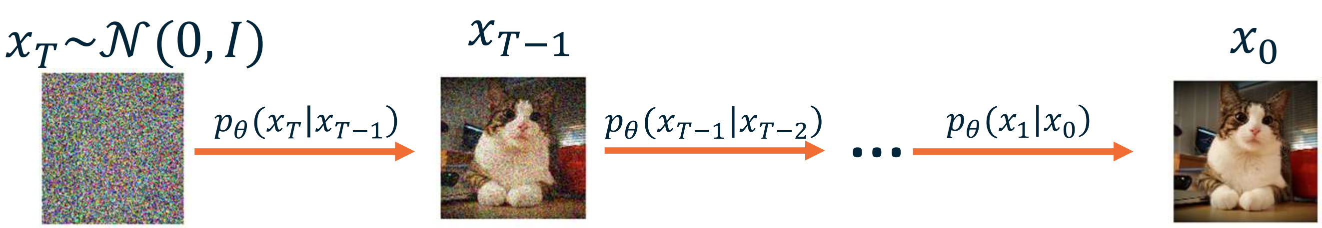
Probability Chain Rule (Markov Chain)

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

Conditional Gaussian

We know how to learn

Assume fixed / known variance




Generate new images!

# The Denoising Diffusion Algorithm

**Algorithm 1** Training

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

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

# The Denoising Diffusion Algorithm

**Algorithm 1** Training

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

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

# The Denoising Diffusion Algorithm

**Algorithm 1** Training

**repeat**

$\mathbf{x}_0 \sim q(\mathbf{x}_0)$

$t \sim \mathrm{Uniform}(\{1, \ldots, T\})$

$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

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 $\mathbf{z} = \mathbf{0}$

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

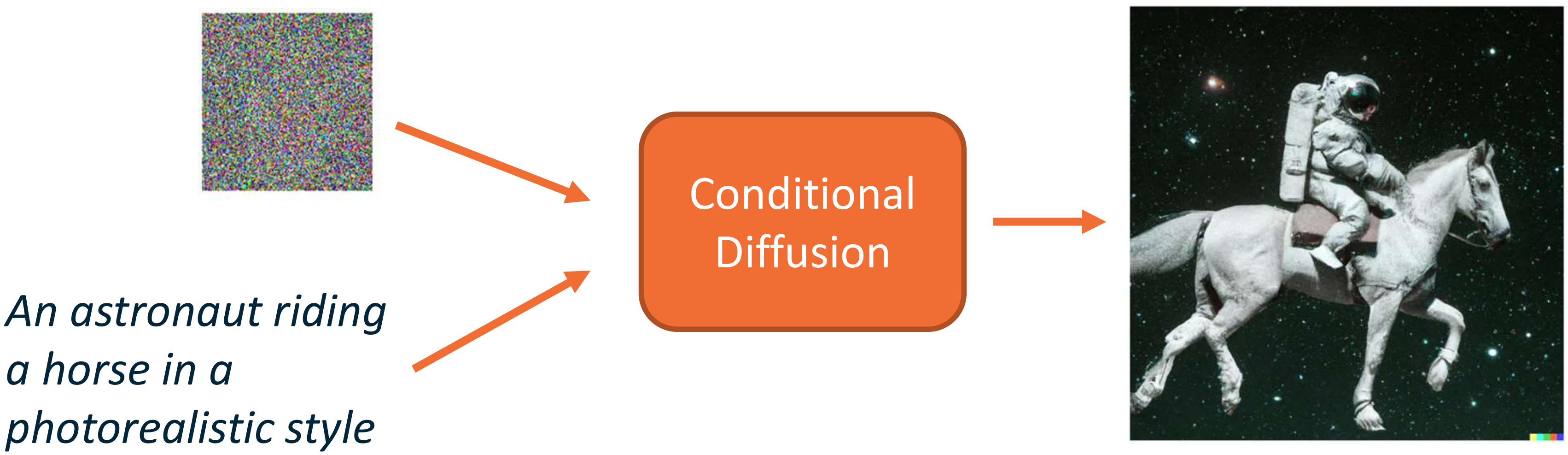
**end for**

**return** $\mathbf{x}_0$

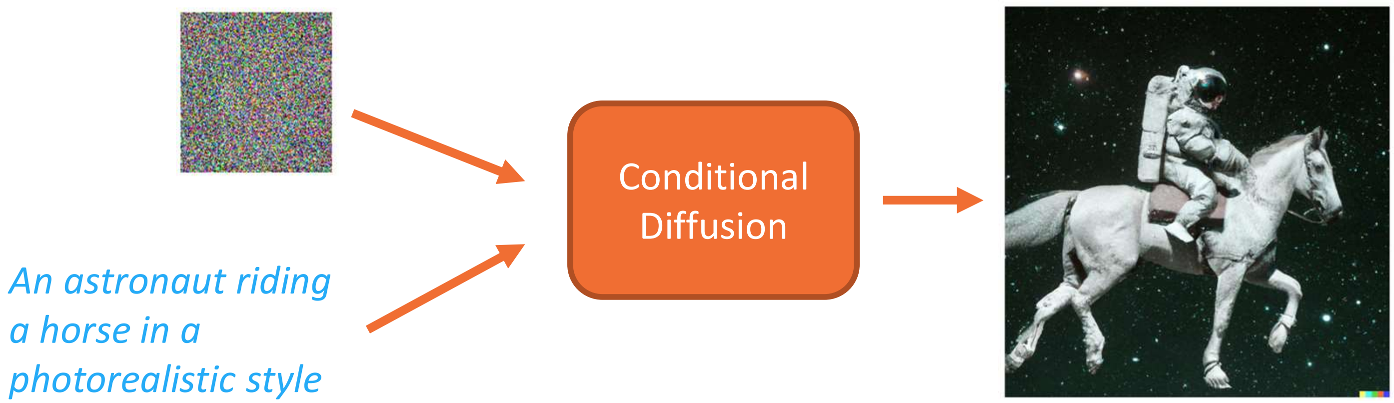
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma(t))$$

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

# Conditional Diffusion Models



# Conditional Diffusion Models




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

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

# Conditional diffusion models

Include condition as input to reverse process

Reverse process: $p_\theta(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1}; \boxed{\boldsymbol{\mu}_\theta(\mathbf{x}_t, t, \mathbf{c}), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t, \mathbf{c}))}$

Variational upper bound: $L_\theta(\mathbf{x}_0|\mathbf{c}) = \mathbb{E}_q\left[L_T(\mathbf{x}_0) + \sum_{t>1} D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\ \|\ \boxed{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})) - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{c})}\right].$

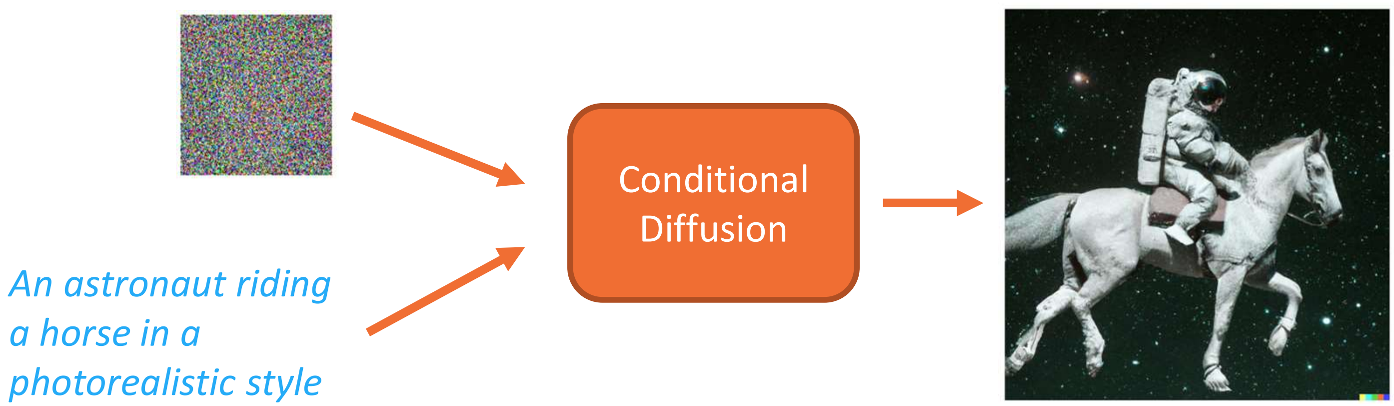
Incorporate conditions into U-Net

- Scalar conditioning: encode scalar as a vector embedding, simple spatial addition or adaptive group normalization layers.
- Image conditioning: channel-wise concatenation of the conditional image.
- Text conditioning: single vector embedding - spatial addition or adaptive group norm / a seq of vector embeddings - cross-attention.



Slide by Soumyadip (Roni) Sengupta

# Conditional Diffusion Models

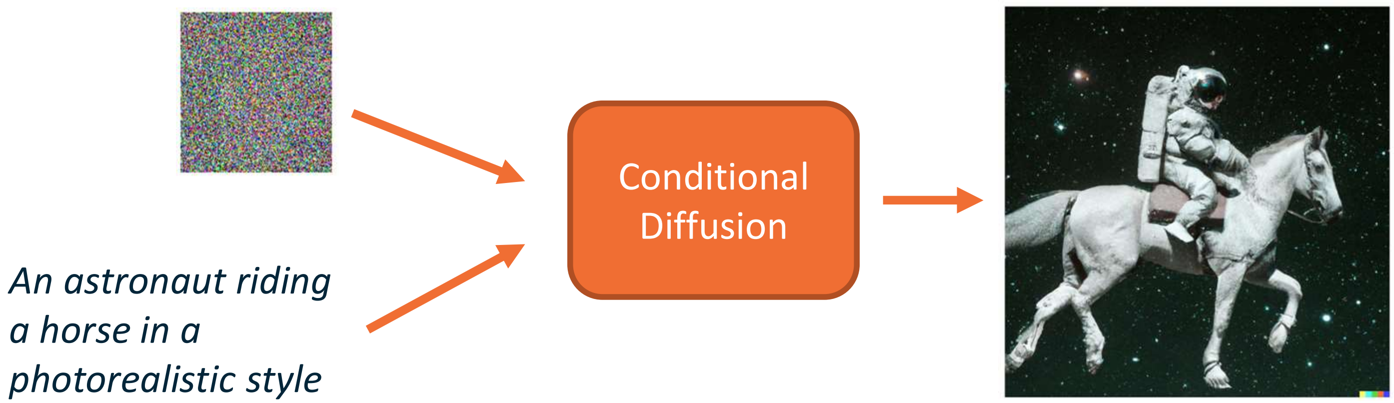



Simple idea: just condition the model on some text labels $y$!

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

Problem: Very blurry generation

# Classifier-guided Diffusion




Better idea: use the *gradients* from a image captioning model $f_\varphi(y|x_t)$ to guide the diffusion process!

$$\bar{\epsilon}_\theta(x_t, t) = \epsilon_\theta(x_t, t) - \sqrt{1 - \bar{\alpha}_t} \nabla_{x_t} \log f_\varphi(y|x_t)$$

Dhariwal & Nichol, 2021

# Classifier guidance

Using the gradient of a trained classifier as guidance

Applying Bayes rule to obtain conditional score function $\nabla_{x_t} log\ q_t(x_t/y)$

$$p(x \mid y) = \frac{p(y \mid x) \cdot p(x)}{p(y)}$$

$$\implies \log p(x \mid y) = \log p(y \mid x) + \log p(x) - \log p(y)$$

$$\implies \nabla_x \log p(x \mid y) = \nabla_x \log p(y \mid x) + \nabla_x \log p(x),$$

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x).$$ ⟵ Classifier

Guidance scale: value >1 amplifies the influence of classifier signal.

$$p_\gamma(x \mid y) \propto p(x) \cdot p(y \mid x)^\gamma.$$

Slide Credits of guidance: https://benanne.github.io/2022/05/26/guidance.html

**Slide by Soumyadip (Roni) Sengupta**

# Classifier guidance

Using the gradient of a trained classifier as guidance

**Algorithm 1** Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale $s$.

Input: class label $y$, gradient scale $s$ Score model Classifier gradient

$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$

**for all** $t$ from $T$ to 1 **do**

  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$

  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$

**end for**

**return** $x_0$

- Train unconditional Diffusion model

- Take your favorite classifier, depending on the conditioning type

- During inference / sampling mix the gradients of the classifier with the predicted score function of the unconditional diffusion model.

**Slide by Soumyadip (Roni) Sengupta**

# Classifier guidance

Using the gradient of a trained classifier as guidance

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x).$$

Samples from an unconditional diffusion model with classifier guidance, for guidance scales 1.0 (left) and 10.0 (right), taken from Dhariwal & Nichol (2021).

**Slide by Soumyadip (Roni) Sengupta**

# Classifier guidance

## Problems of classifier guidance

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x).$$

Classifier

Guidance scale: value >1 amplifies the influence of classifier signal.

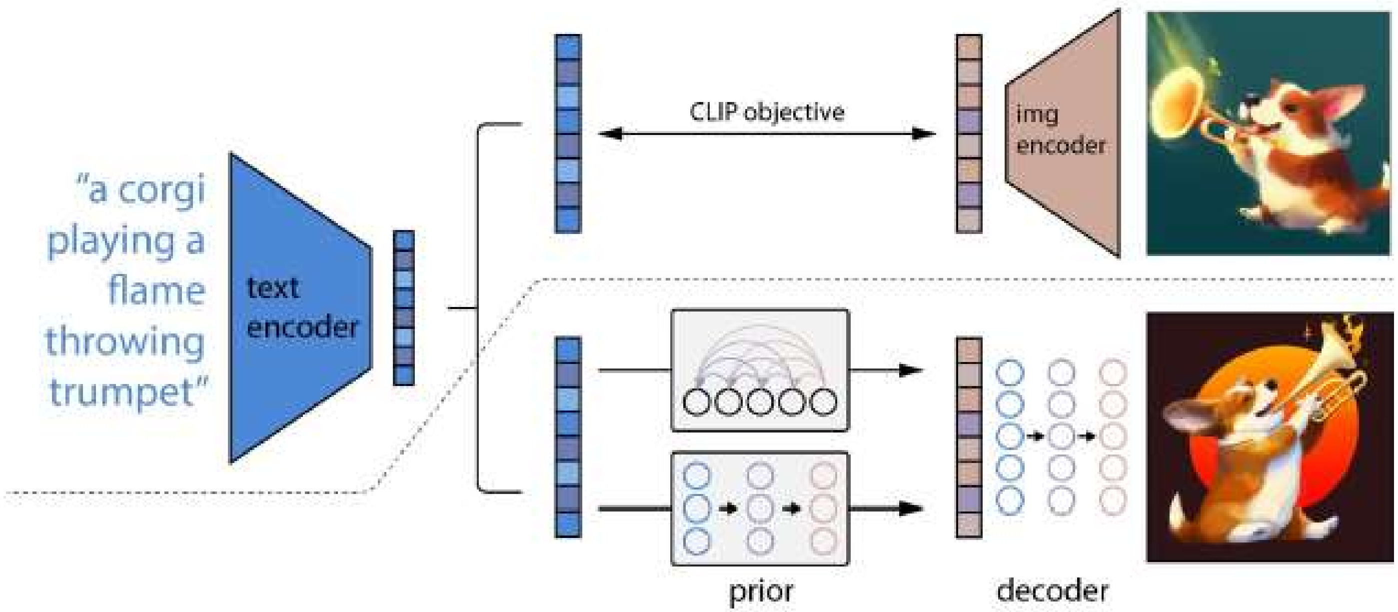
- At each step of denoising the input to the classifier is a noisy image $x_t$. Classifier is never trained on noisy image. So one needs to re-train classifier on noisy images! Can't use existing pre-trained classifiers.

- Most of the information in the input x is not relevant to predicting y, and as a result, taking the gradient of the classifier w.r.t. its input can yield arbitrary (and even adversarial) directions in input space.

Solution 1 (DALL-E 2): Use CLIP Model

# DALL·E 2

## Model components

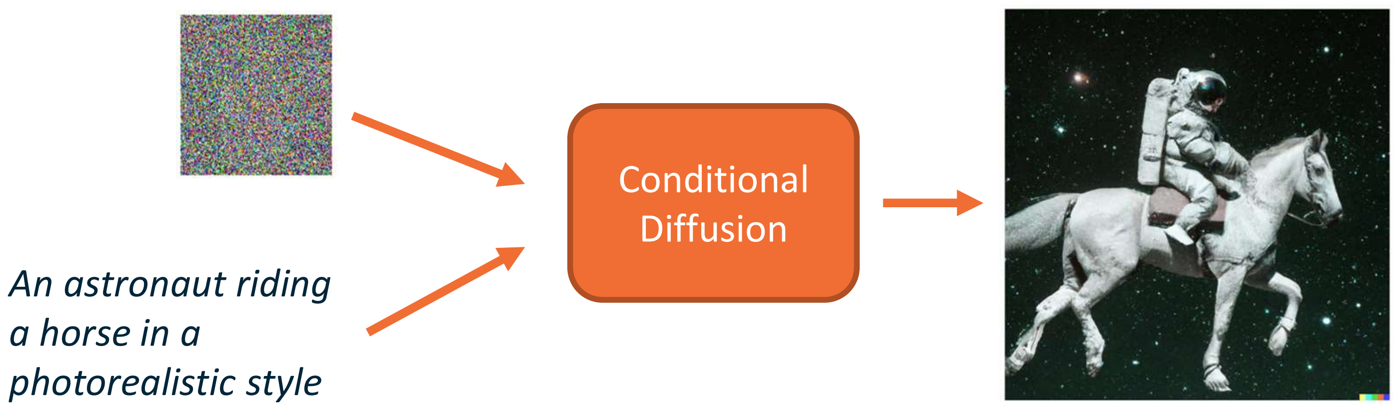



Why conditional on CLIP image embeddings?

CLIP image embeddings capture high-level semantic meaning.

Slide by Soumyadip (Roni) Sengupta

# Classifier-free Guided Diffusion




**Classifier-free Guided Diffusion**: estimate the gradient of the classifier model with conditional diffusion models!

$$\nabla_{x_t} \log f_\varphi(y|x_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\left(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t)\right)$$

Ho and Salimans, 2022

# Classifier-free guidance

Get guidance by Bayes' rule on conditional diffusion models

$$p(y \mid x) = \frac{p(x \mid y) \cdot p(y)}{p(x)}$$

$$\implies \log p(y \mid x) = \log p(x \mid y) + \log p(y) - \log p(x)$$

$$\implies \boxed{\nabla_x \log p(y \mid x) = \nabla_x \log p(x \mid y) - \nabla_x \log p(x).}$$

We proved this in classifier guidance.

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y \mid x).$$

$$\nabla_x \log p_\gamma(x \mid y) = \nabla_x \log p(x) + \gamma \left(\nabla_x \log p(x \mid y) - \nabla_x \log p(x)\right),$$

$$\nabla_x \log p_\gamma(x \mid y) = (1-\gamma)\nabla_x \log p(x) + \gamma \nabla_x \log p(x \mid y).$$

Score function for unconditional diffusion model

Score function for conditional diffusion model

Slide by Soumyadip (Roni) Sengupta

# Classifier-free guidance

Get guidance by Bayes' rule on conditional diffusion models

$$\hat{\epsilon} = (1+\omega)\epsilon_\theta(x_t, y) - \omega\epsilon_\theta(x_t)$$

$$\nabla_x \log p_\gamma(x \mid y) = (1-\gamma)\nabla_x \log p(x) + \gamma\nabla_x \log p(x \mid y).$$

Score function for unconditional diffusion model

Score function for conditional diffusio model

In practice:

- Train a conditional diffusion model p(x|y), with *conditioning dropout*: some percentage of the time, the conditioning information y is removed (10-20% tends to work well).
- The conditioning is often replaced with a special input value representing the absence of conditioning information.
- The resulting model is now able to function both as a conditional model p(x|y), and as an unconditional model p(x), depending on whether the conditioning signal is provided.
- During inference / sampling simply mix the score function of conditional and unconditional diffusion model based on guidance scale.

**Slide by Soumyadip (Roni) Sengupta**

# Classifier-free guidance

Trade-off for sample quality and sample diversity

Non-guidance

Guidance scale = 1

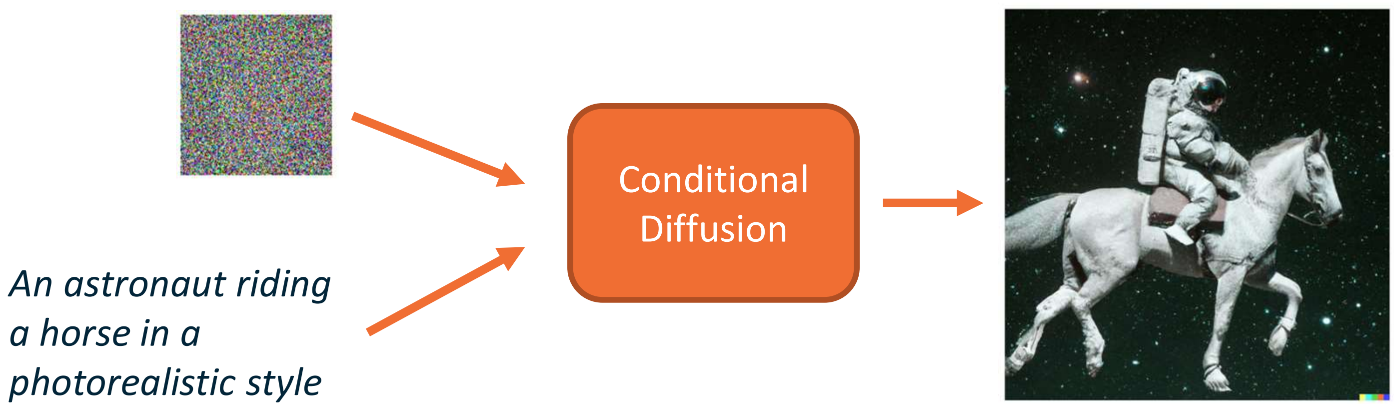
Guidance scale = 3

Large guidance weight ($\omega$) usually leads to better individual sample quality but less sample diversity.

Ho & Salimans, "Classifier-Free Diffusion Guidance", 2021.

Slide by Soumyadip (Roni) Sengupta

# Classifier-free Guided Diffusion




**Classifier-free Guided Diffusion**: estimate the gradient of the classifier model with conditional diffusion models!

$$\nabla_{x_t} \log f_\varphi(y|x_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\left(\epsilon_\theta(x_t, t, y) - \epsilon_\theta(x_t, t)\right)$$

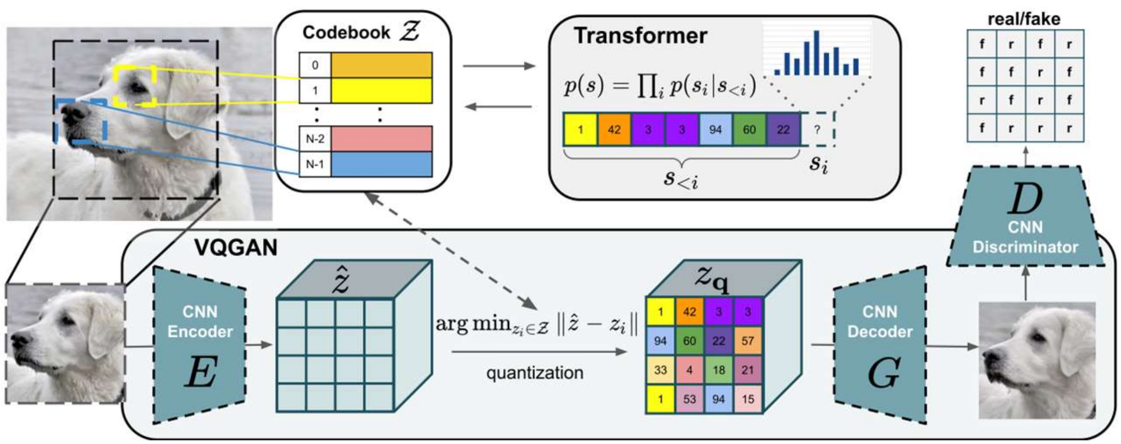
$$\bar{\epsilon}_\theta(x_t, t, y) = (w+1)\epsilon_\theta(x_t, t, y) - w\epsilon_\theta(x_t, t)$$

Ho and Salimans, 2022

# Latent-space Diffusion

Problem: Hard to learn diffusion process on high-resolution images

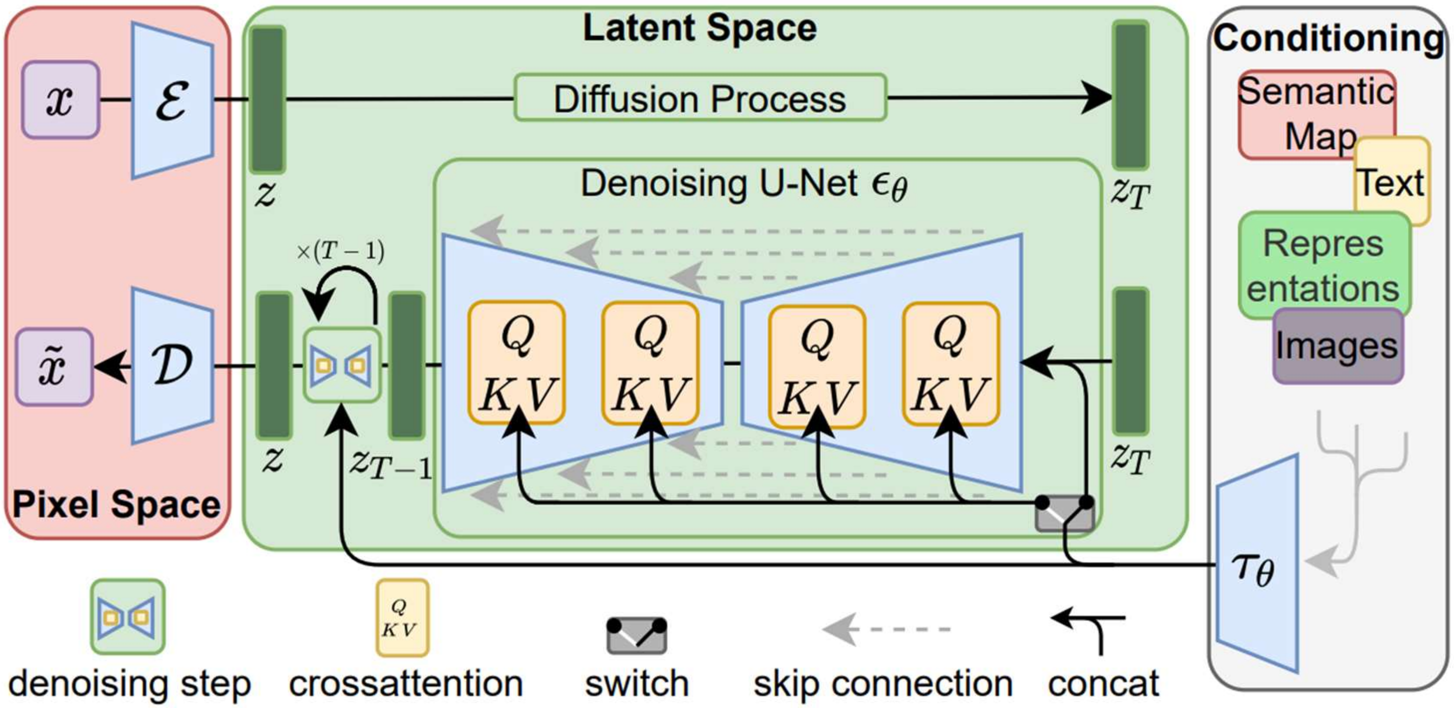
Solution: learn a low-dimensional latent space using a transformer-based autoencoder and *do diffusion on the latent space*!




The latent space autoencoder

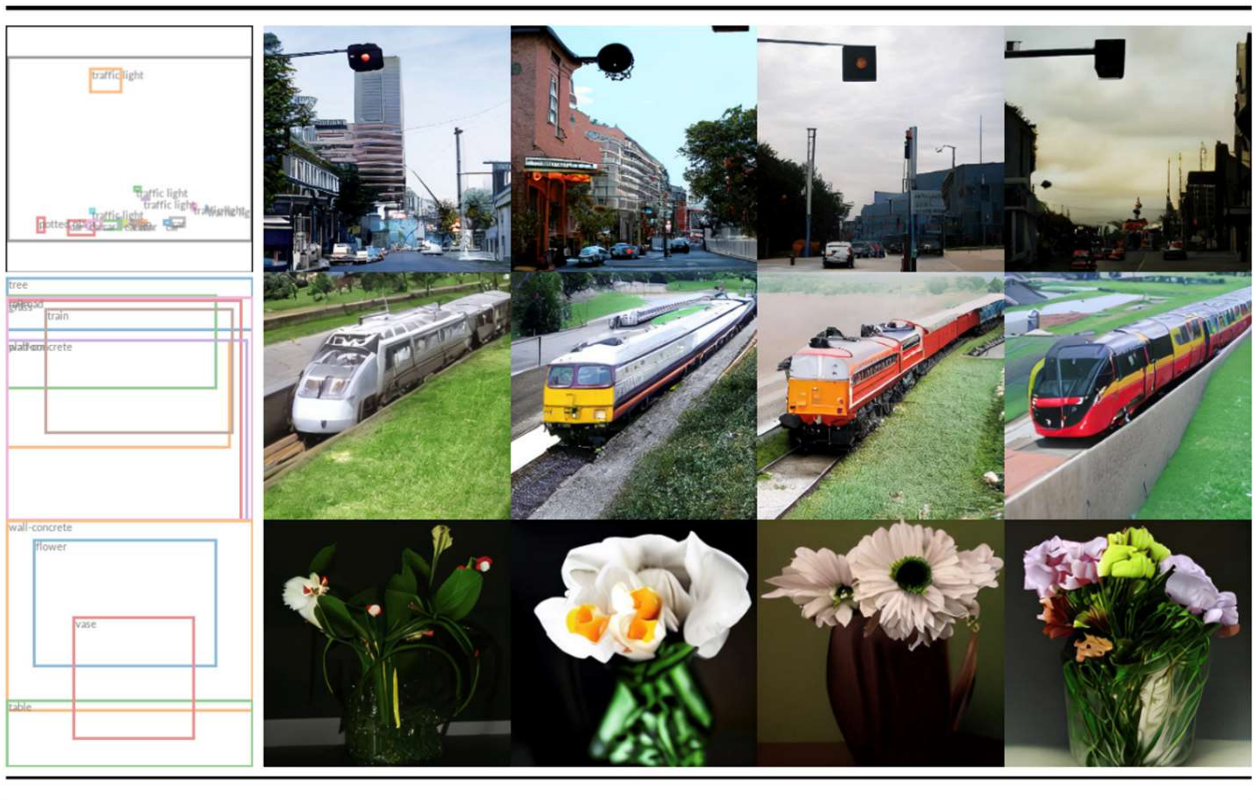
Esser and Rombach *et al.*, 2021

# “StableDiffusion”




Rombach and Blattmann *et al.*, 2022

# “StableDiffusion”




Layout-Conditional Generation

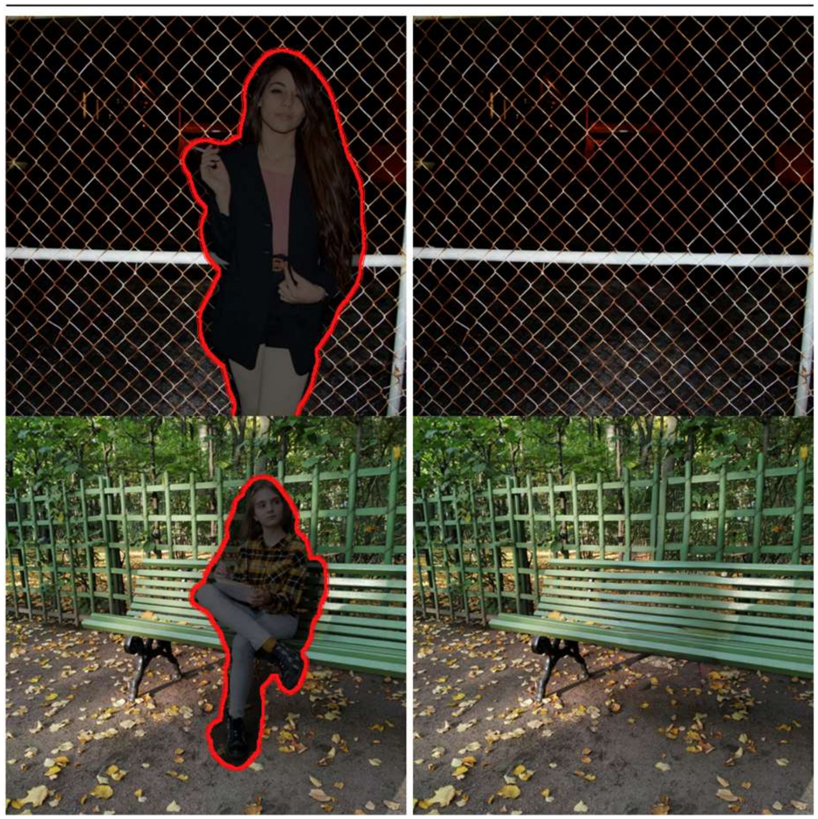
Rombach and Blattmann *et al.*, 2022

# “StableDiffusion”

Segmentation-Conditional Generation

Rombach and Blattmann *et al.*, 2022

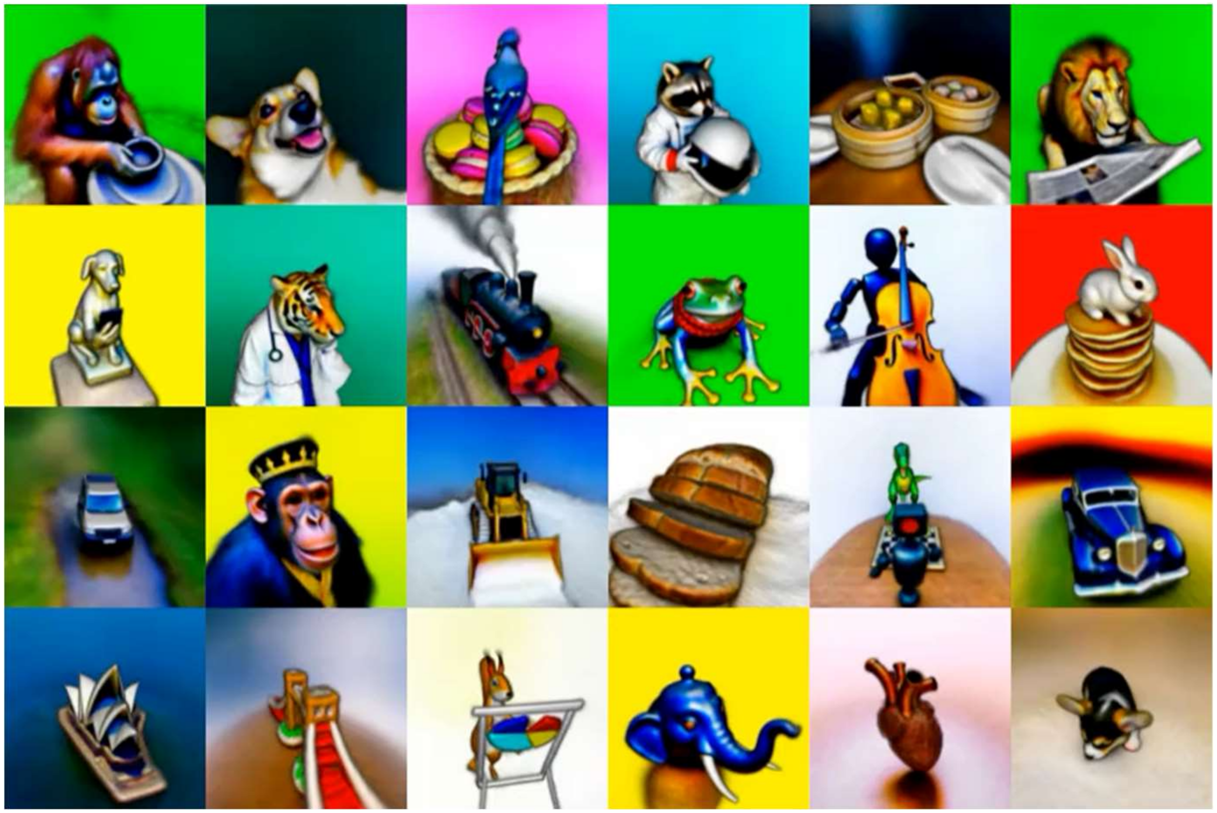
# “StableDiffusion”



Inpainting

Rombach and Blattmann *et al.*, 2022

# Beyond Image Generation



https://dreamfusion3d.github.io/

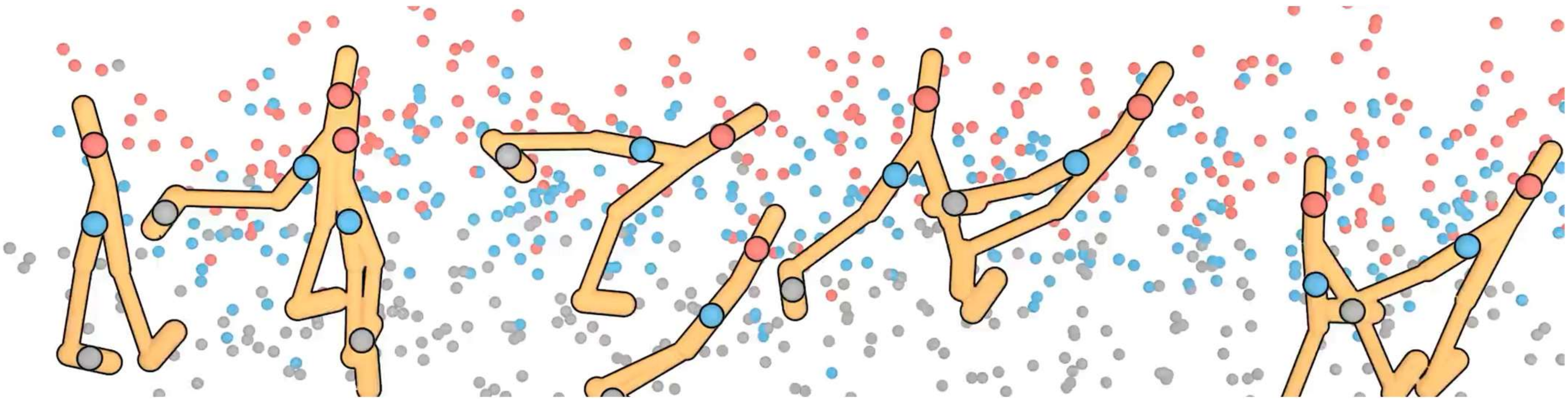
# Beyond Image Generation


https://ai.facebook.com/blog/generative-ai-text-to-video/

# Video LDM

https://research.nvidia.com/labs/toronto-ai/VideoLDM/o-video/

# Beyond Image Generation



https://ai.facebook.com/blog/generative-ai-text-to-video/

# Additional resources / tutorials

- Overview of the research landscape: What are Diffusion Models?
- More math! Understanding Diffusion Models: A Unified Perspective
- Tutorial with hands-on example: The Annotated Diffusion Model
- Nice introduction videos:
  - What are Diffusion Models?
  - Diffusion Models | Math Explained
- CVPR Tutorial: Denoising Diffusion-based Generative Modeling: Foundations and Applications
- Score functions:
  - In general
  - For Diffusion models

# Summary

- Denoising Diffusion model is a type of generative model that learns the process of “denoising” a known noise source (Gaussian).
- We can construct a learning problem by deriving the evidence lower bound (ELBO) of the denoising process.
- The learning objective is to minimize the KL divergence between the “ground truth” and the learned denoising distribution.
- A simplified learning objective is to estimate the noise of the forward diffusion process.
- The diffusion process can be guided to generate targeted samples.
- Can be applied to many different domains. Same underlying principle.
- Very hot topic!