CS 4644-DL / 7643-A: Lecture 18 Danfei xu

Generative Models:

Denoising Diffusion Probabilistic Models (DDPMs)

Taxonomy of Generative Models

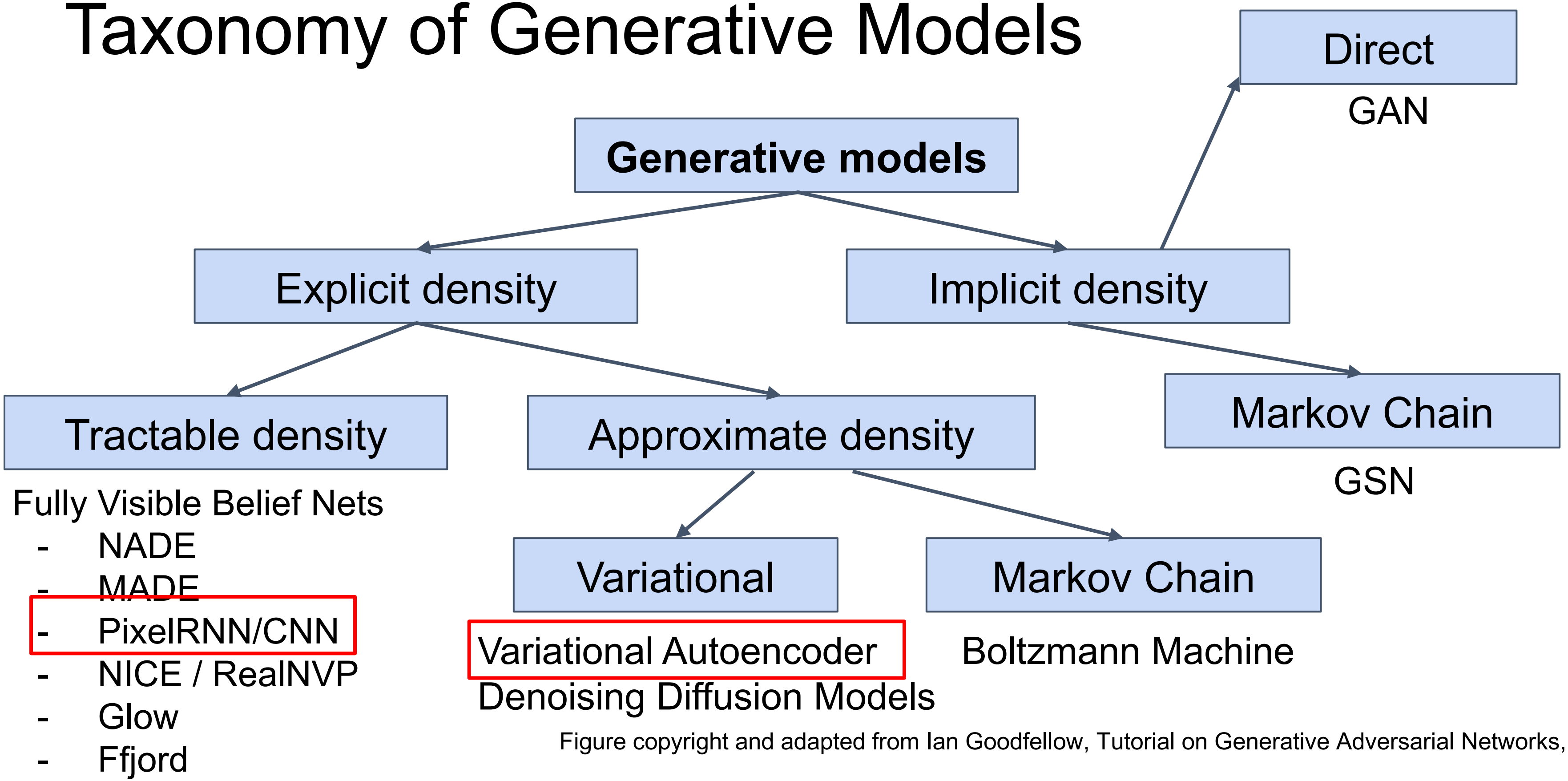
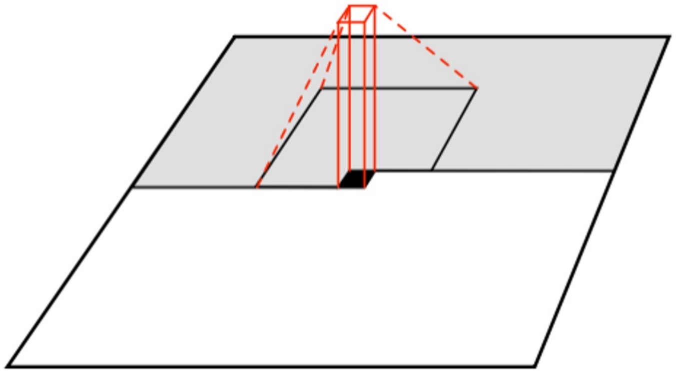


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

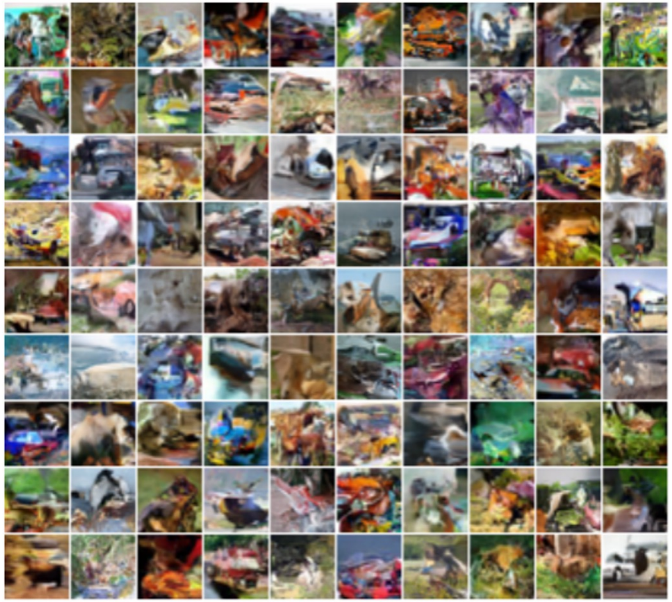
PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

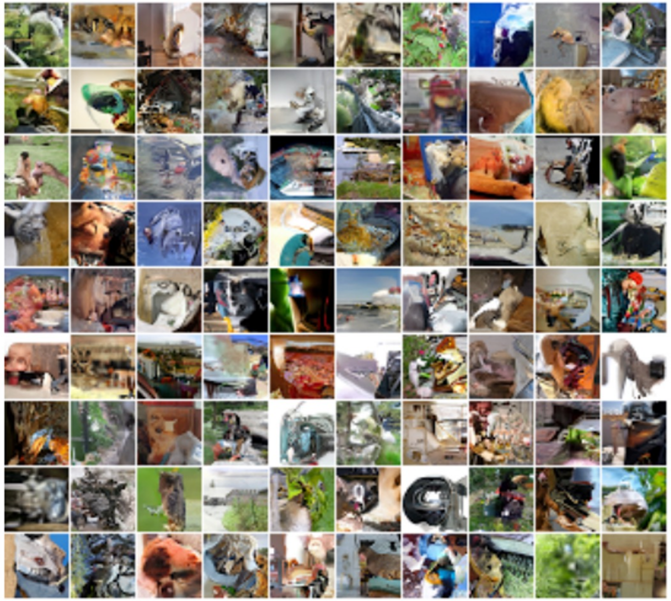
Dependency on previous pixels now modeled using a CNN over context region (**masked convolution**)



Generation Samples



32x32 CIFAR-10



32x32 ImageNet

Variational Autoencoders

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

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

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

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

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

Decoder: reconstruct the input data

Encoder: make approximate posterior distribution close to prior

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Variational Autoencoders

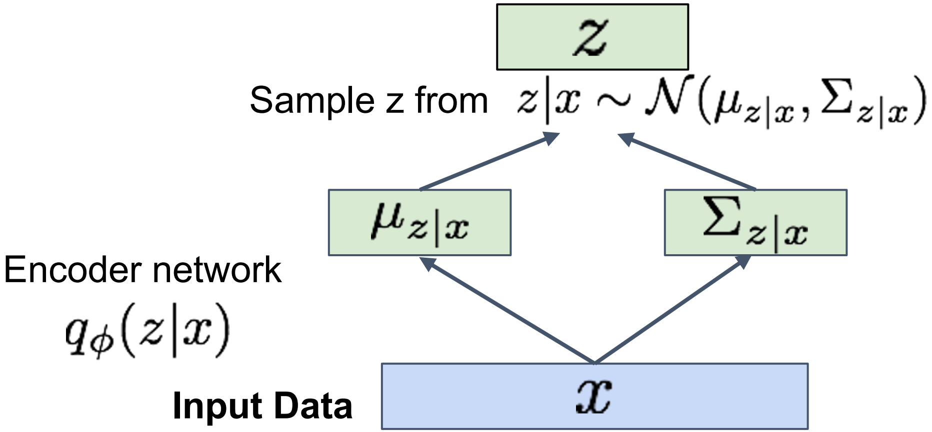
Putting it all together: maximizing the likelihood lower bound

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

Reparameterization trick to make sampling differentiable:

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

$$z = \mu_{z|x} + \epsilon\sigma_{z|x}$$

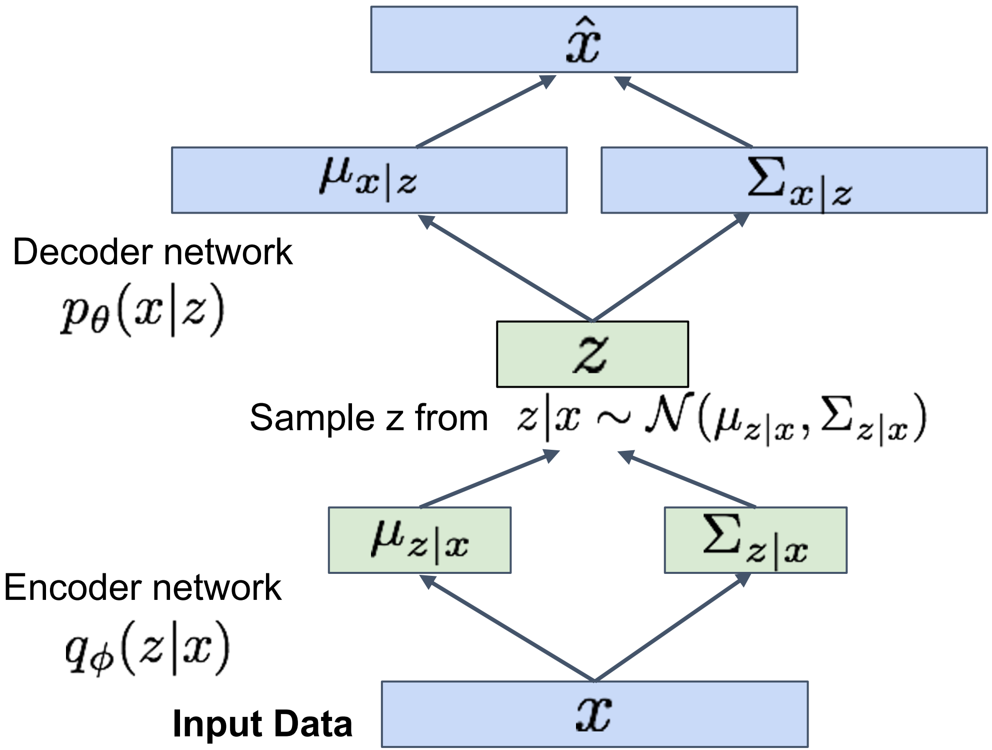


Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

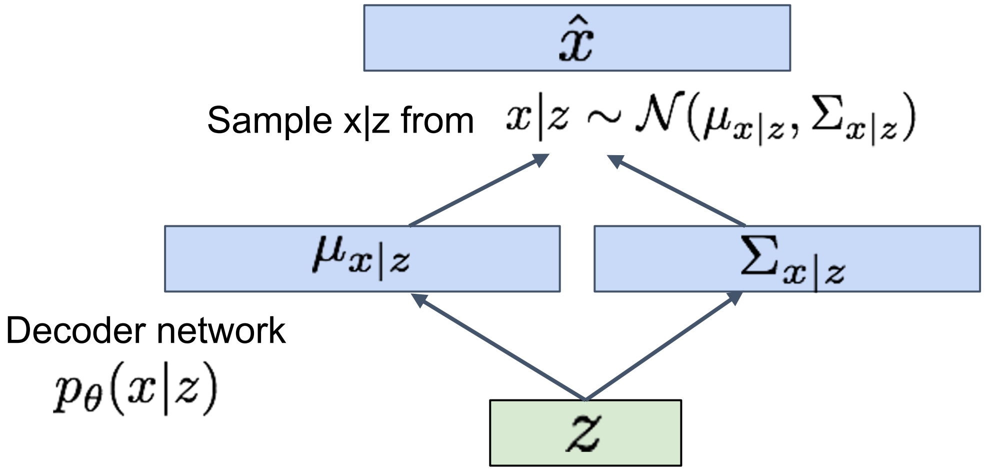
$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} \mid z) \right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

For every minibatch of input data: compute this forward pass, and then backprop!



Variational Autoencoders: Generating Data!

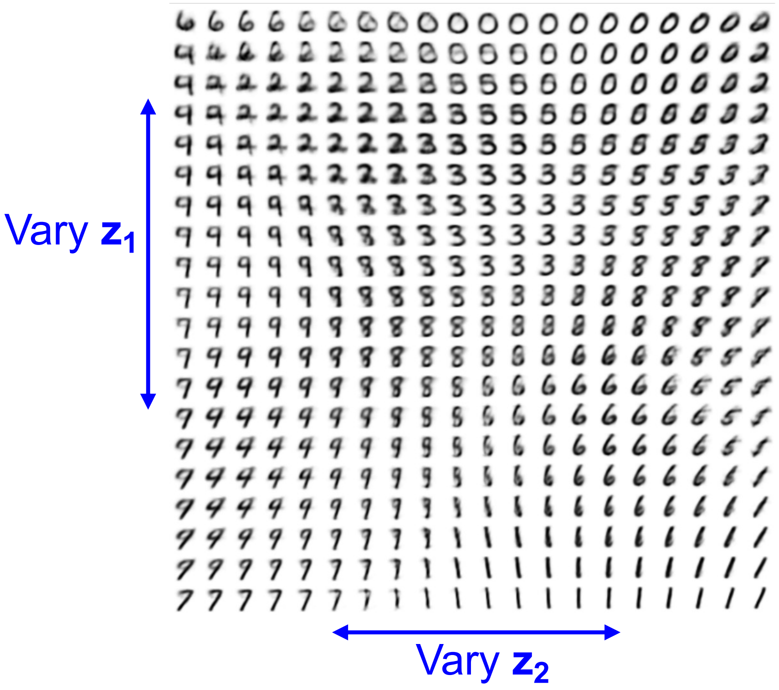
Use decoder network. Now sample z from prior!



Sample z from $z \sim \mathcal{N}(0, I)$

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Data manifold for 2-d **z**



Taxonomy of Generative Models

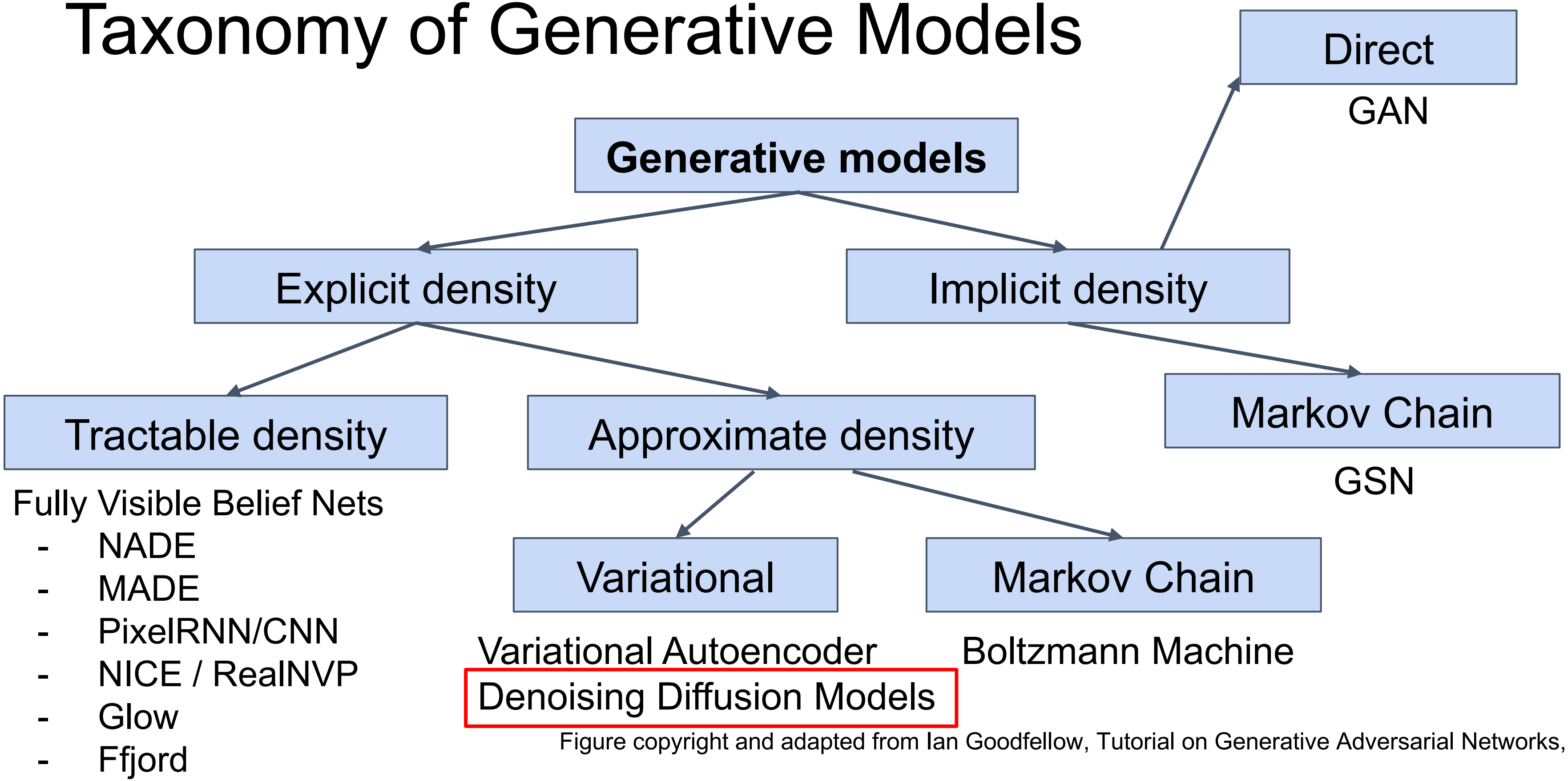


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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/

Watch 321 | Fork 5k | Starred 33k

rity | Insights

main | 1 branch | 0 tags | Go to file | Add file | Code

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

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*

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%

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)

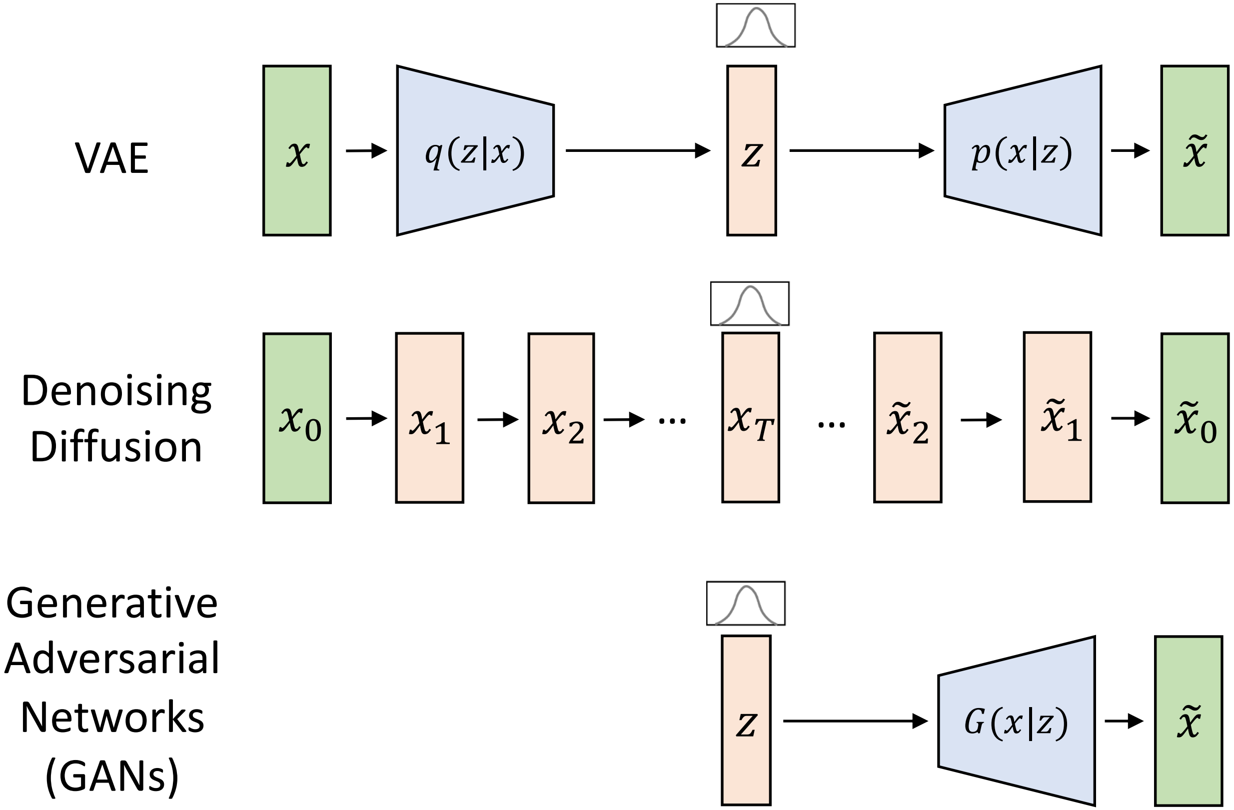
conditional & high-res image generation

- *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)

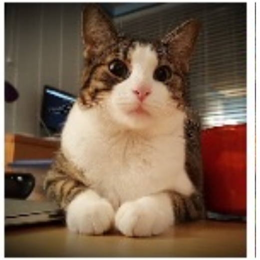
Denoising Diffusion: Image to Noise and Back



The Denoising Diffusion Process

image from
dataset

x_0

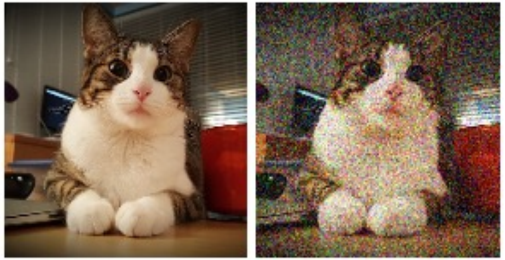


The Denoising Diffusion Process

image from dataset

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

$x_0 \longrightarrow x_1 \longrightarrow$



...

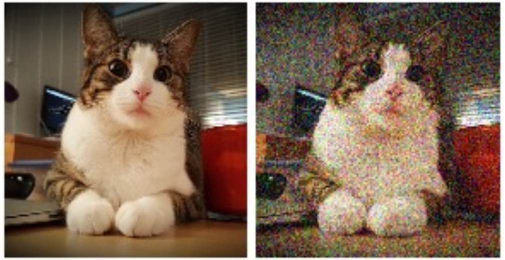
The Denoising Diffusion Process

image from dataset

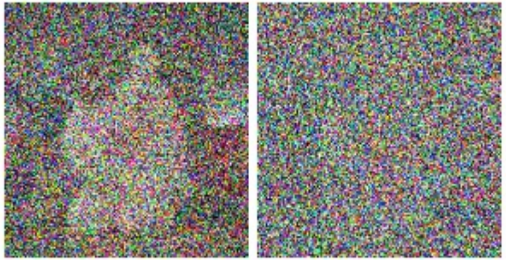
The “forward diffusion” process: add Gaussian noise each step

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

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



...



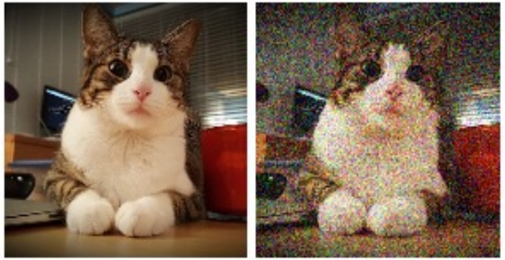
The Denoising Diffusion Process

image from
dataset

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

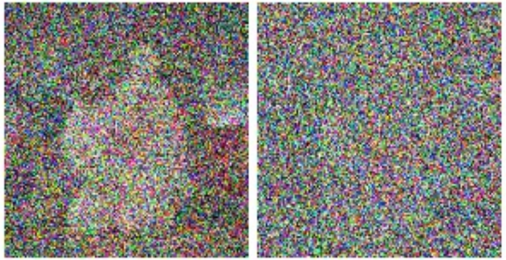
noise $\mathcal{N}(0, I)$

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



...

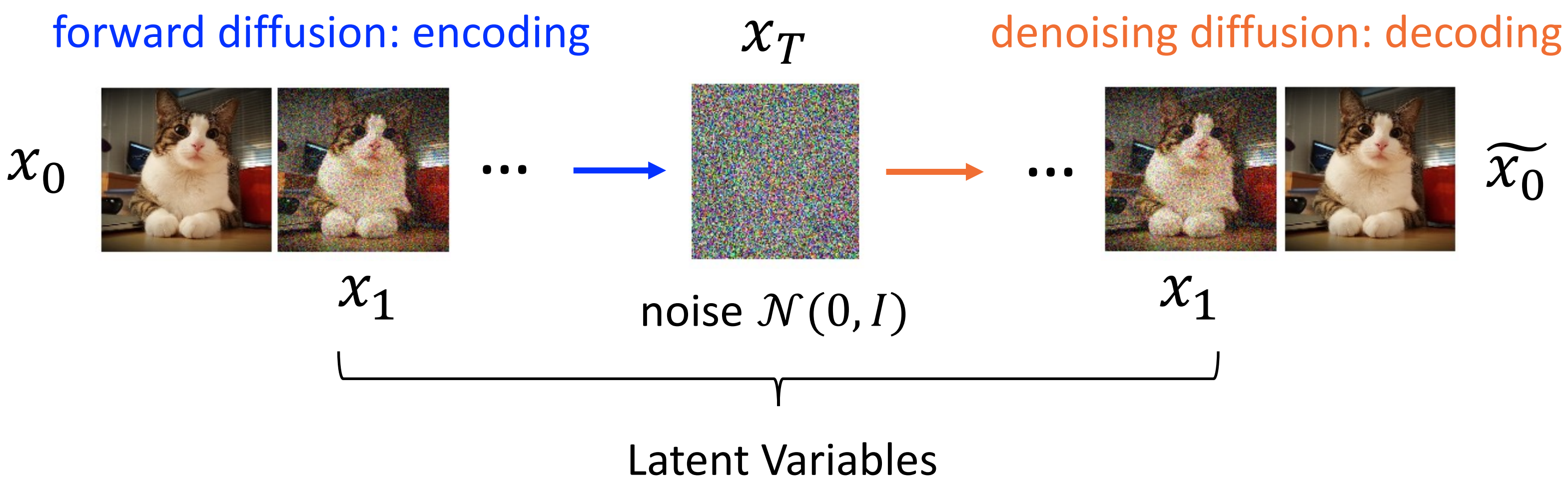
...



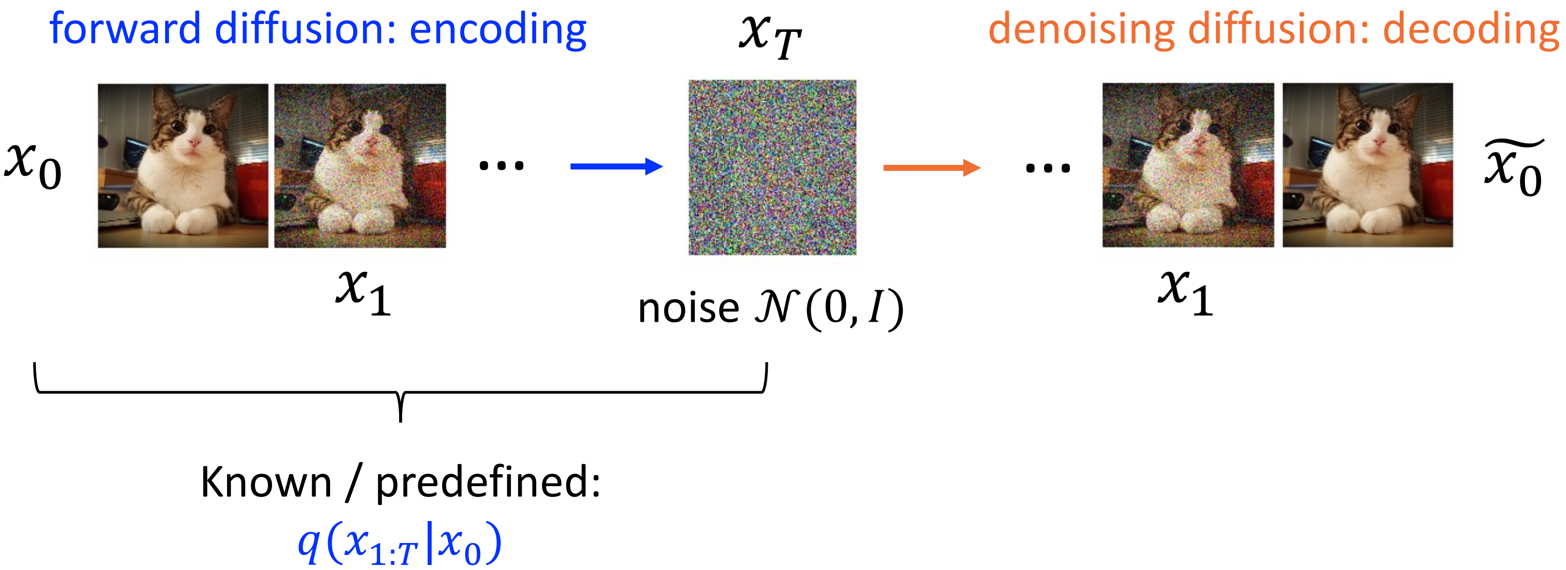
$x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_{T-1} \leftarrow x_T$

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

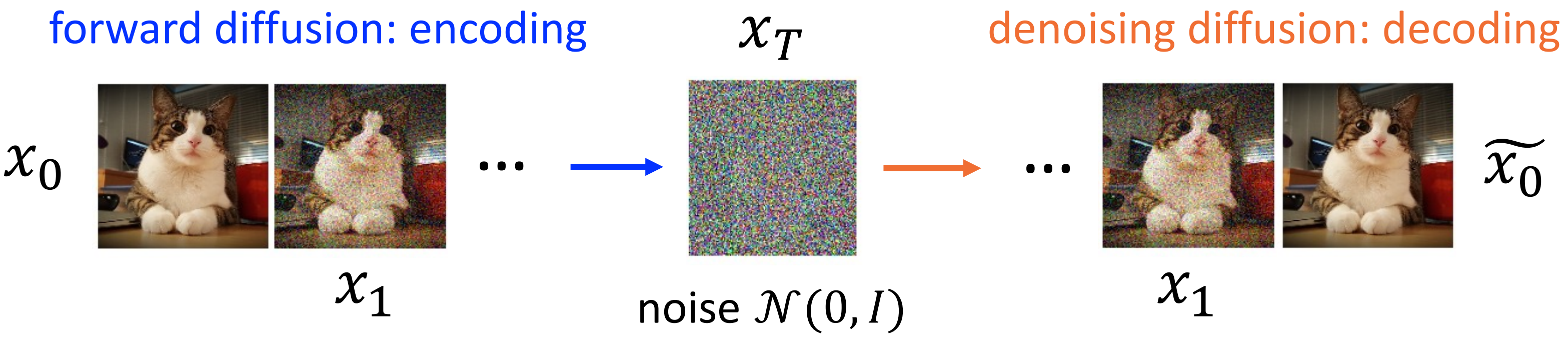
Connection to VAEs



Connection to VAEs



Connection to VAEs



Known / predefined:

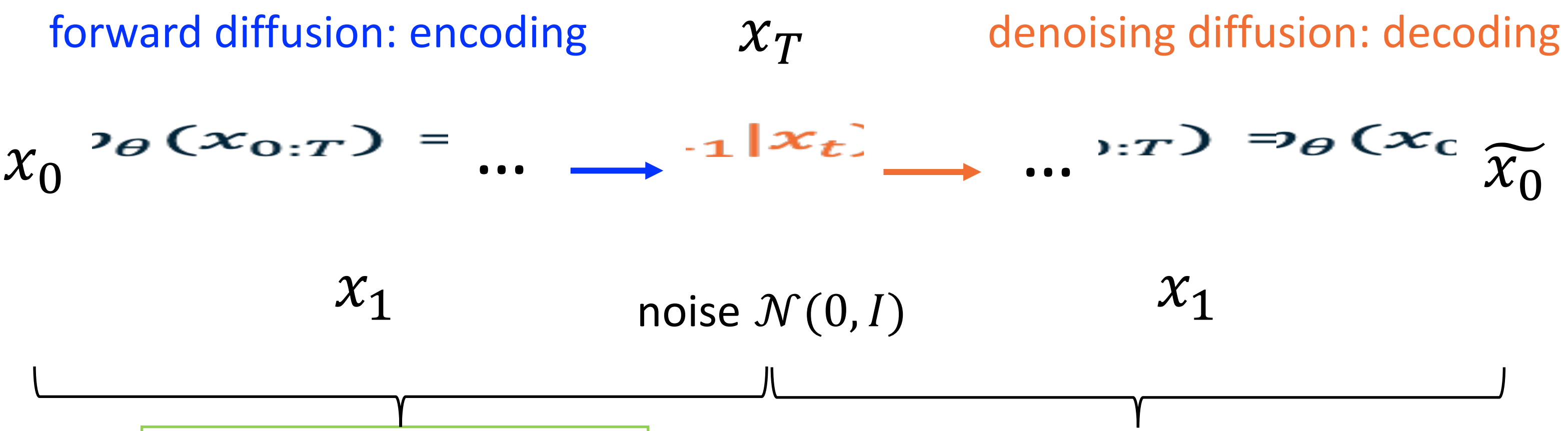
$q(x_{1:T}|x_0)$

Unknown / learned:

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

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)

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

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

Notation: A Gaussian distribution "for" x_t

Plain English: the distribution for x_t is a Gaussian with mean of $(1-\beta_t)x_{t-1}$, where x_{t-1} is a sample from the previous step, and variance of $\beta_t I$

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)

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

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (1-\beta_t)x_{t-1}, \beta_t I)$$ 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 $x_0 \rightarrow x_1 \rightarrow \dots \rightarrow x_T$

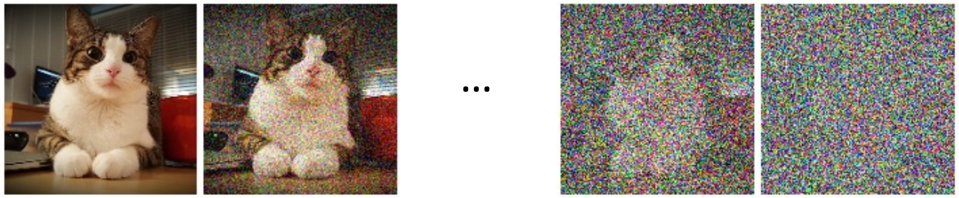
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (1-\beta_t)x_{t-1}, \beta_t I)$$ 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 $T = 1000$

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



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)

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; (1-\beta_t)x_{t-1}, \beta_t I)$$ 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 $a_t = (1-\beta_t)$, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

x_0 x_1 x_t x_{T-1} 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)

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

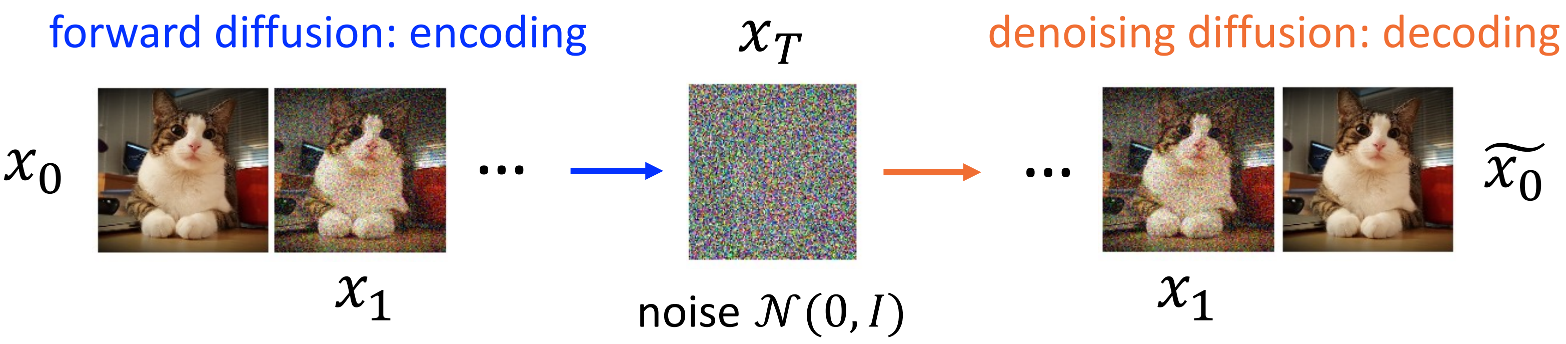
Nice property: samples from an *arbitrary forward step* are also Gaussian-distributed!

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

Gaussian reparameterization trick (recall from VAEs!):

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

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 $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

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)

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 (now reversed)

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

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

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

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

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:

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

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

The Denoising (Decoding) Process

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

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

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

The Denoising (Decoding) Process

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

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

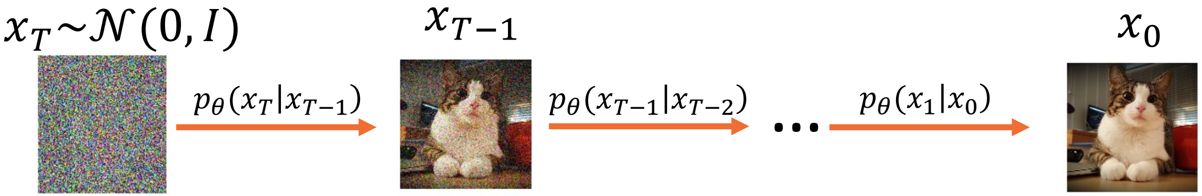
Probability Chain Rule (Markov Chain)

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

Conditional Gaussian

We know how to learn

Assume fixed / known variance



Generate new images!

The Denoising (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)

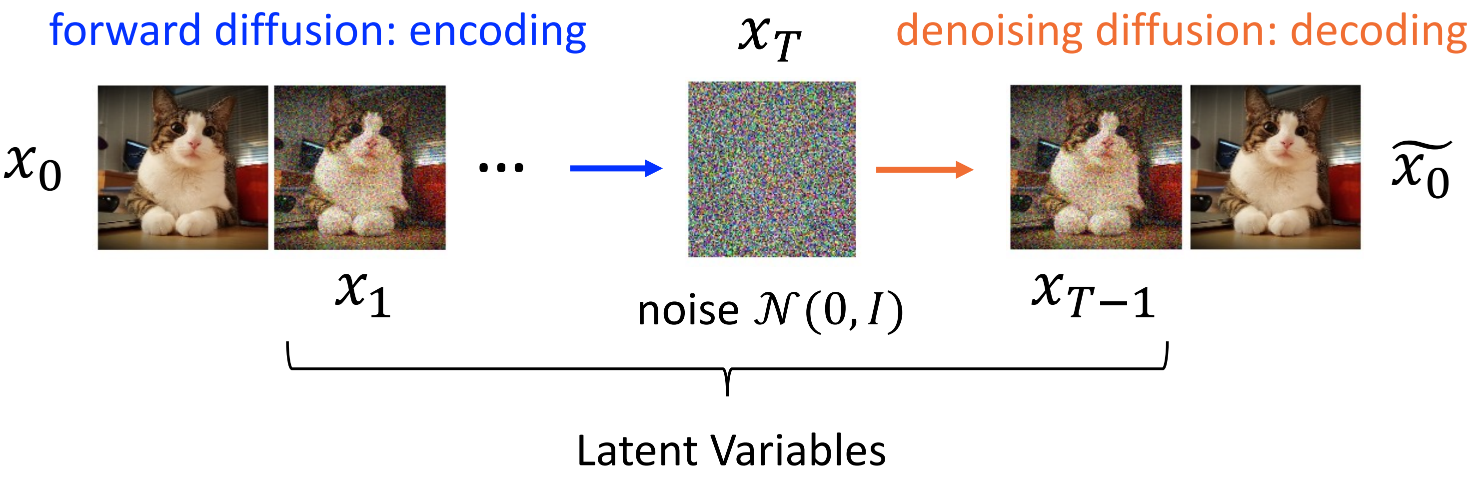
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$ Conditional Gaussian

We know how to learn ($\mu_\theta(x_t, t)$) — Assume fixed / known variance ($\Sigma_q(t)$)

How did we arrive at the learning objective? Why is this mathematically correct?

Let's go back to the basics of variational models …

Connection to VAEs



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

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

$p(x) = \int p(x|z)p(z)dz$ 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)

Known forward noise (posterior)

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

$p(x) = \int p(x|z)p(z)dz$ 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

$$p(x) = \int p(x|z)p(z)dz$$ 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)

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

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

← reverse denoising

← forward diffusion

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

$p(x) = \int p(x|z)p(z)dz$ 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)

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

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

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

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

$p(x) = \int p(x|z)p(z)dz$ 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)

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

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

… (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

$p(x) = \int p(x|z)p(z)dz$ 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)

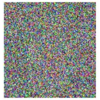
$$\log p(x_0) \geq \mathrm{E}_q\left[\log\frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right]$$ $x = x_0,\ z = x_{1:T}$

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

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

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

known



Easy to optimize / sometimes omitted

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

$p(x) = \int p(x|z)p(z)dz$ 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)

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

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

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

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

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

$p(x) = \int p(x|z)p(z)dz$ 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)

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

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

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

… (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

$$p(x) = \int p(x|z)p(z)dz$$ 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)

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

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

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

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

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

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

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

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

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

$p(x) = \int p(x|z)p(z)dz$ 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)

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

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

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

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

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

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

Proof using bayes rule and gaussian reparameterization trick

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

$p(x) = \int p(x|z)p(z)dz$ 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)

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

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

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

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

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

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

Proof using bayes rule and gaussian reparameterization trick

The "ground truth" noise that brought x_0 to x_t

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

$p(x) = \int p(x|z)p(z)dz$ 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)

$$\log p(x_0) \geq \mathrm{E}_q\left[\log\frac{p(x_0|x_{1:T})p(x_{1:T})}{q(x_{1:T}|x_0)}\right] \qquad x = x_0,\ z = x_{1:T}$$
$$= \mathrm{E}_q\left[\log\frac{p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}{\prod_{t=1}^{T} q(x_t|x_{t-1})}\right]$$

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

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

Minimize the difference of distribution means (assuming identical variance)

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

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

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

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

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

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

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

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

known during inference

Unknown during inference

Recall: this is the "ground truth" noise that brought x_0 to x_t

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

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

known during inference

Unknown during inference

Recall: this is the "ground truth" noise that brought x_0 to x_t

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

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

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

Recall: the simplified t-step forward sample:

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

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

Recall: the simplified t-step forward sample:

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

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

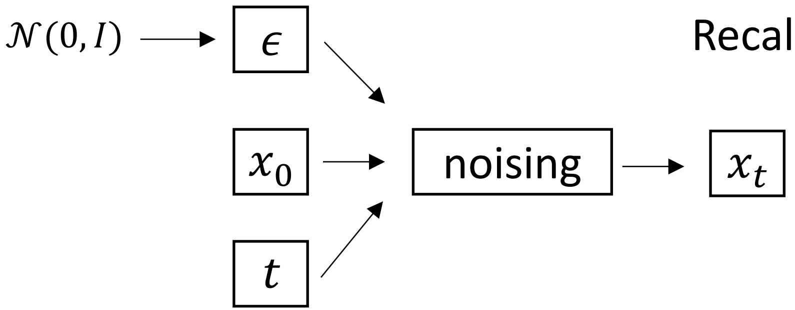
Inference time: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon_\theta(x_t, t)\right)$

Predicted "denoising noise"

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



Recall: the simplified t-step forward sample:

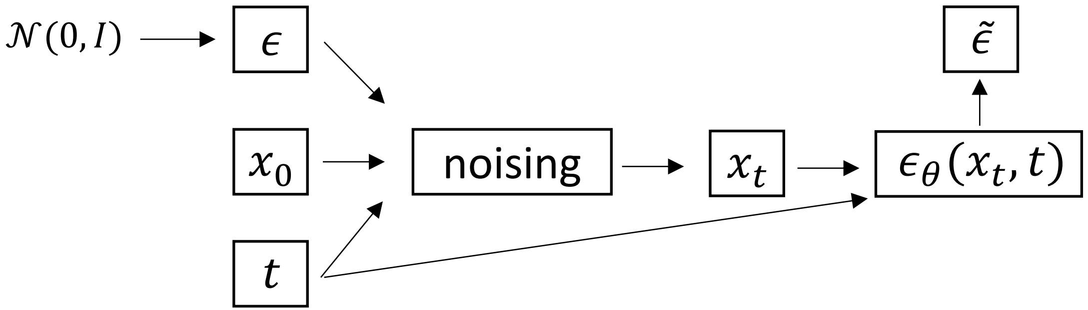
$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

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

The Denoising Diffusion Algorithm

Algorithm 1 Training

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



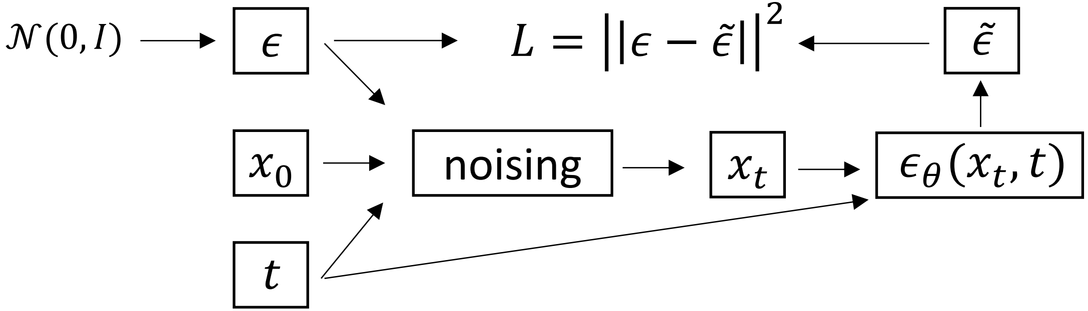
Predict noise with a denoiser network ϵ_θ

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

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



Compute regression loss

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

The Denoising Diffusion Algorithm

Algorithm 1 Training

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

Algorithm 2 Sampling

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

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

The Denoising Diffusion Algorithm

Algorithm 1 Training

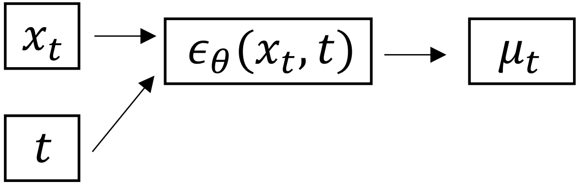
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

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

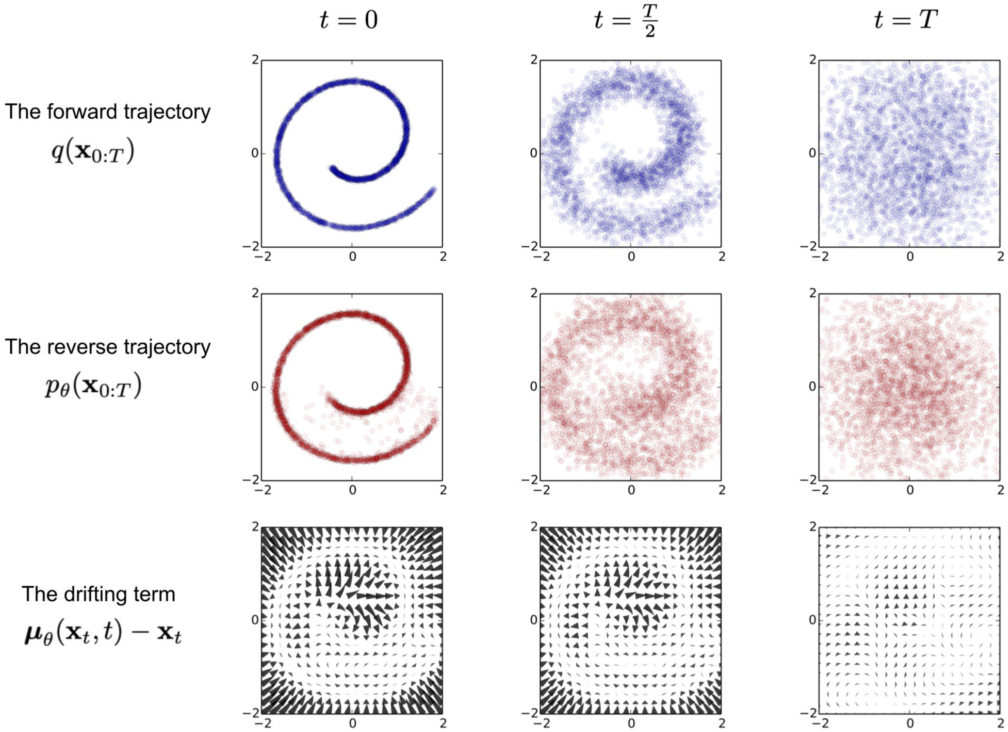
$$\sigma_t = \sqrt{\beta}_t$$

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



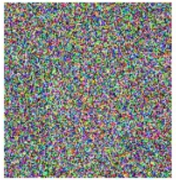
The Denoising Diffusion Probabilistic Models, Ho *et al.*, 2020

Visualizing the Diffusion Process on 2D data



Sohl-Dickstein *et al.*, 2015

Conditional Diffusion Models

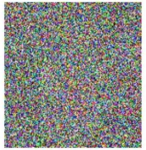


An astronaut riding a horse in a photorealistic style

Conditional Diffusion

$p(x|y)$

Conditional Diffusion Models



An astronaut riding a horse in a photorealistic style

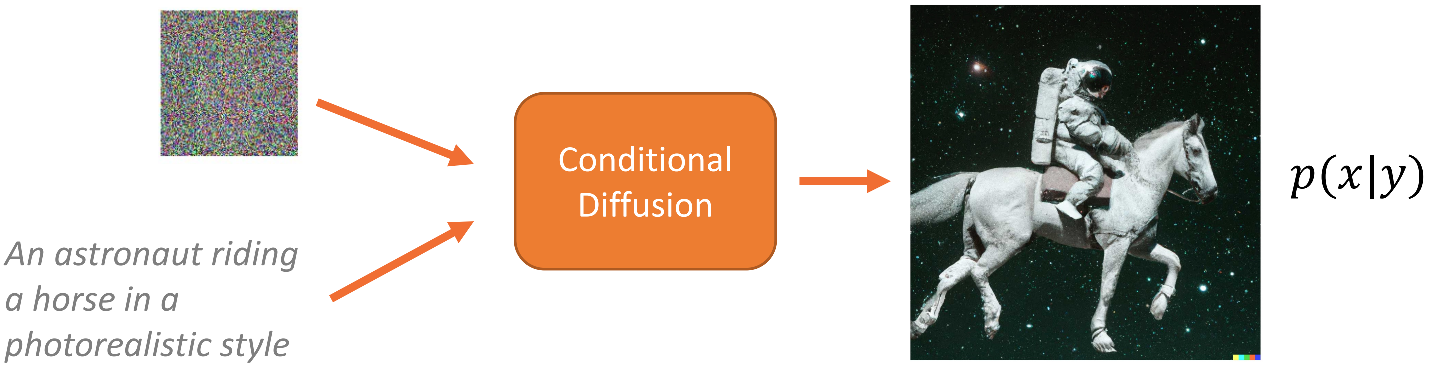
Conditional Diffusion

$p(x|y)$

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

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

Conditional Diffusion Models

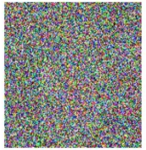


Simple idea: just condition the model on some text labels y!

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

Problem: Very blurry generation

Classifier-guided Diffusion



An astronaut riding a horse in a photorealistic style

Conditional Diffusion

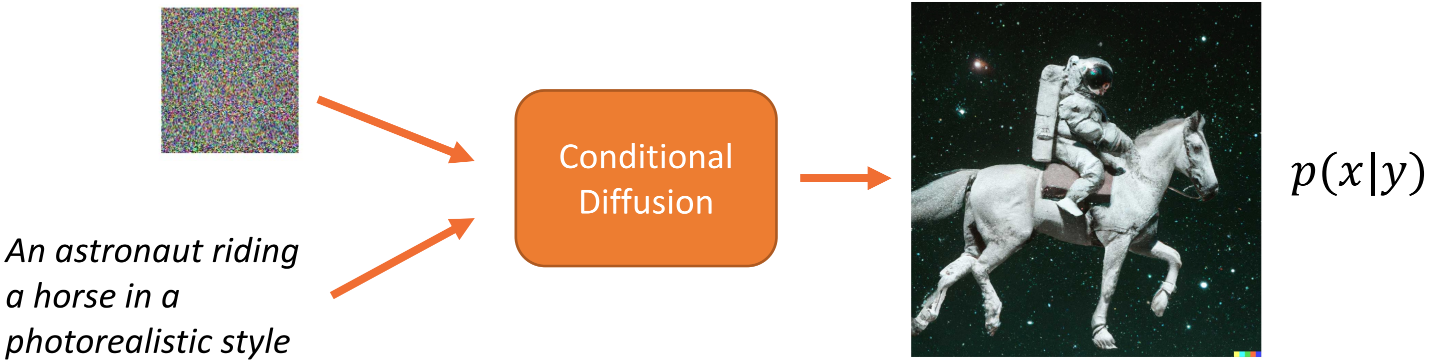
$p(x|y)$

Better idea: use the *gradients* from an 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-guided Diffusion



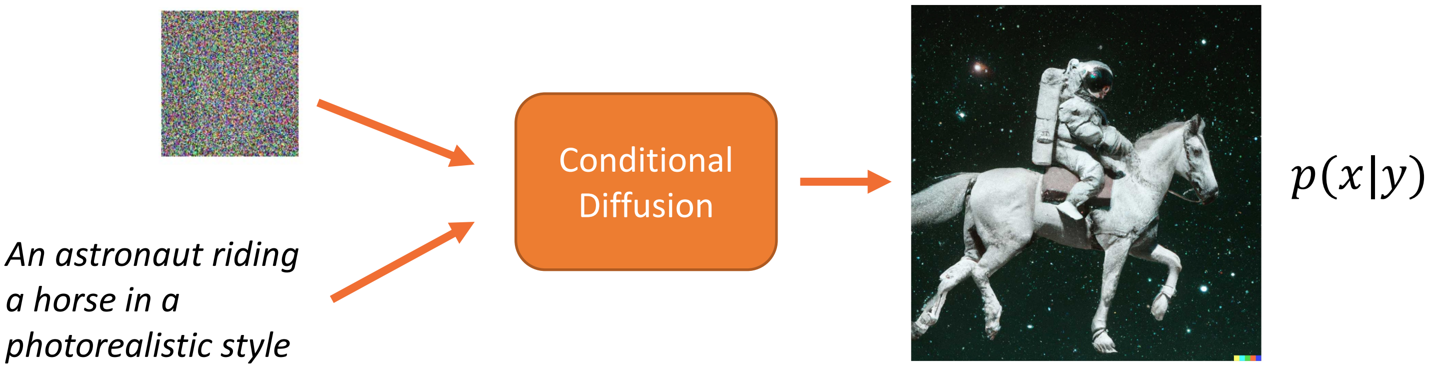
Better idea: use the *gradients* from an image captioning model $f_\varphi(y|x_t)$ to guide the diffusion process!

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

Problem: need a classifier

Dhariwal & Nichol, 2021

Classifier-free Guided Diffusion

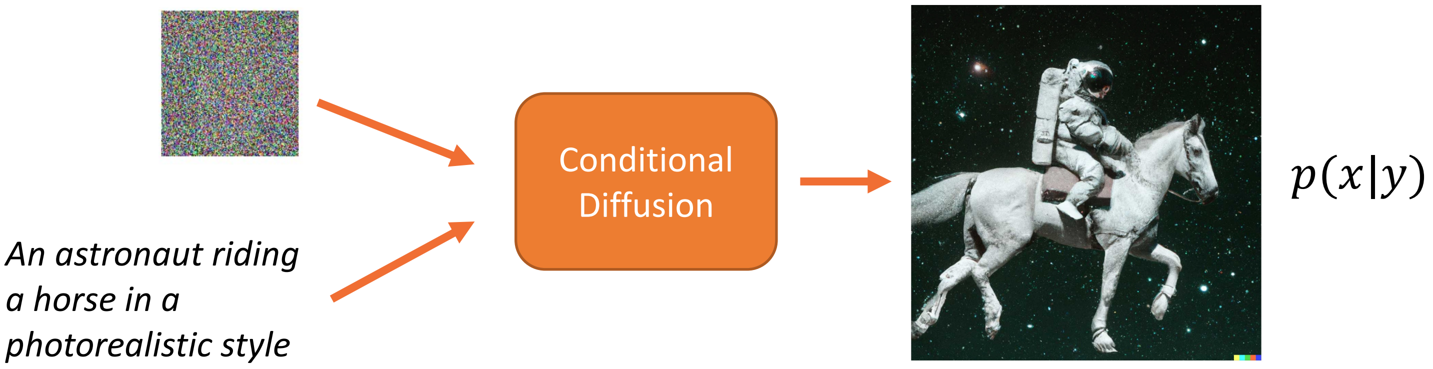


Classifier-free Guided Diffusion: estimate the gradient of the classifier model with conditional diffusion models!

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

Ho and Salimans, 2022

Classifier-free Guided Diffusion



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

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

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

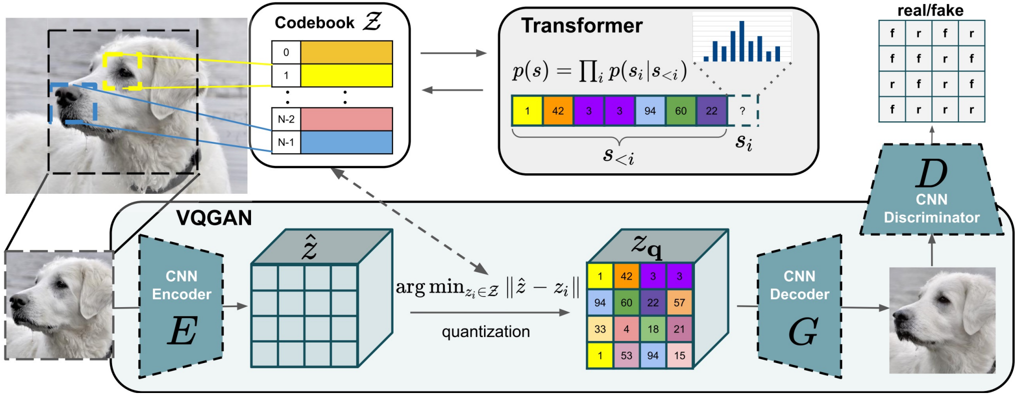
Linearly combine denoisers from an unconditional distribution and a conditional distribution

Ho and Salimans, 2022

Latent-space Diffusion

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

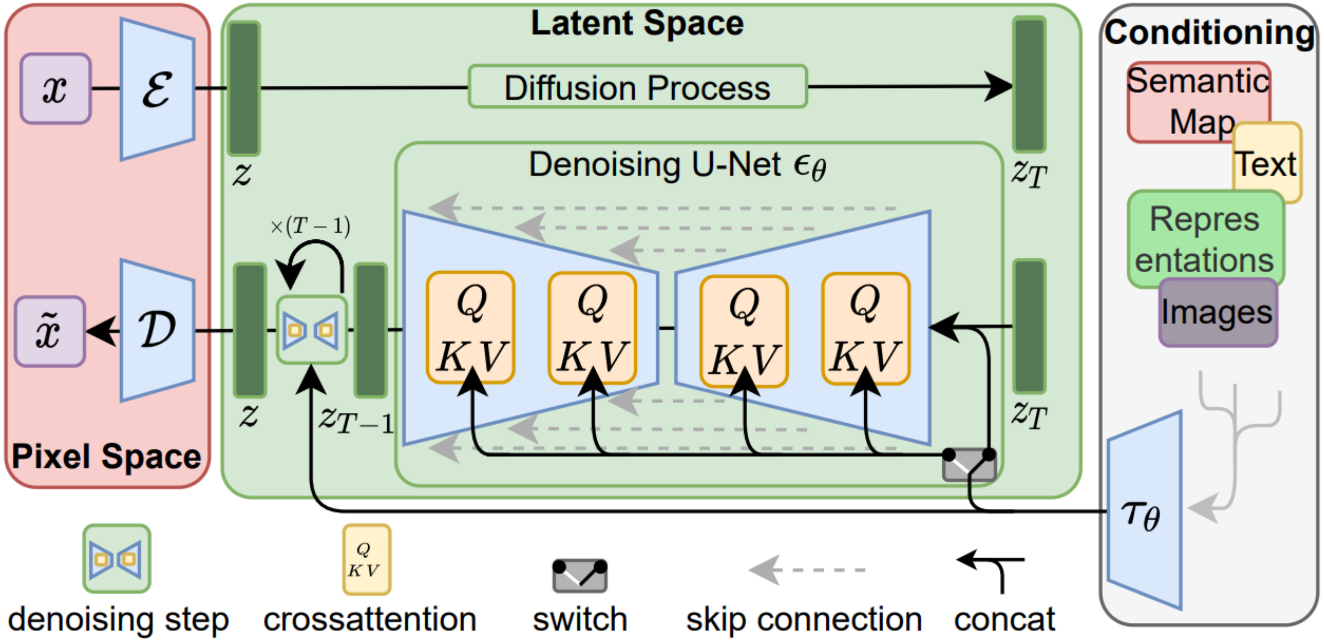
Solution: learn a low-dimensional latent space using a ViT-based autoencoder and *do diffusion on the latent space*!



The latent space autoencoder

Esser and Rombach *et al.*, 2021

“StableDiffusion”



Rombach and Blattmann *et al.*, 2022

"StableDiffusion"

Layout-Conditional Generation

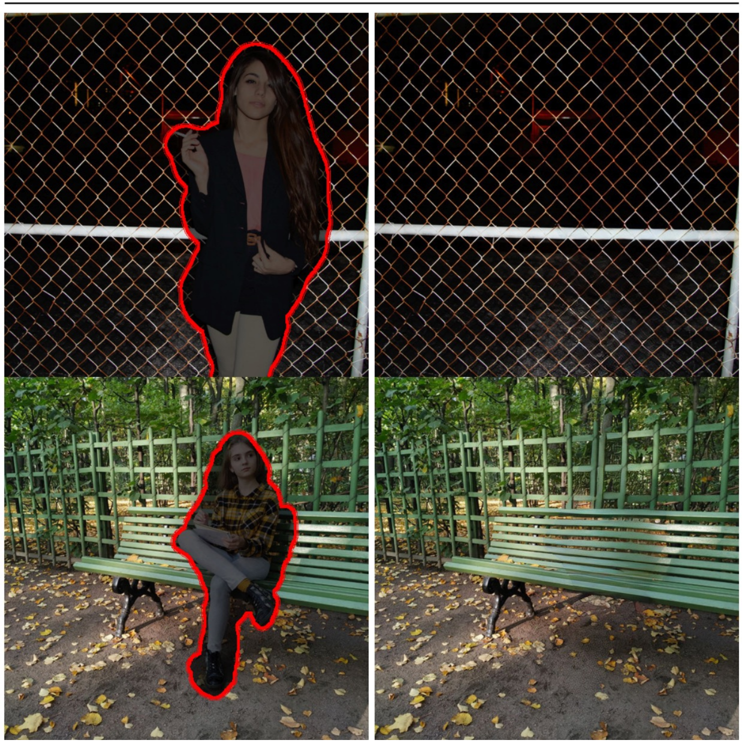
Rombach and Blattmann *et al.*, 2022

“StableDiffusion”

Segmentation-Conditional Generation

Rombach and Blattmann *et al.*, 2022

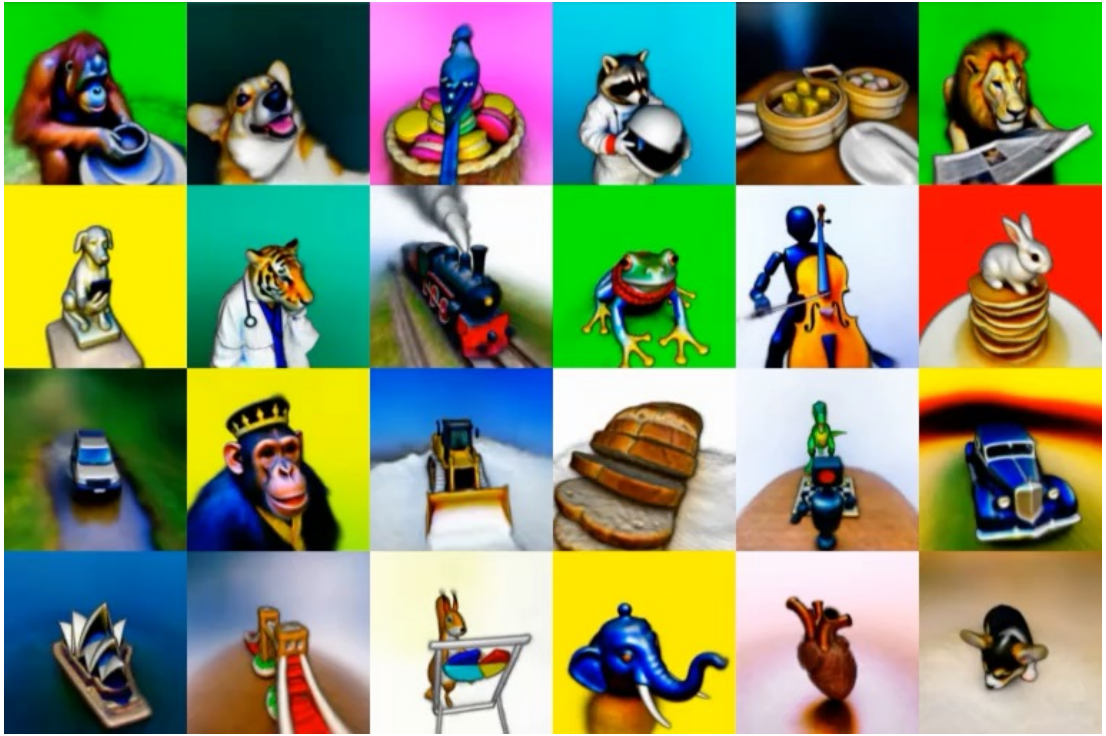
“StableDiffusion”



Inpainting

Rombach and Blattmann *et al.*, 2022

Beyond Image Generation

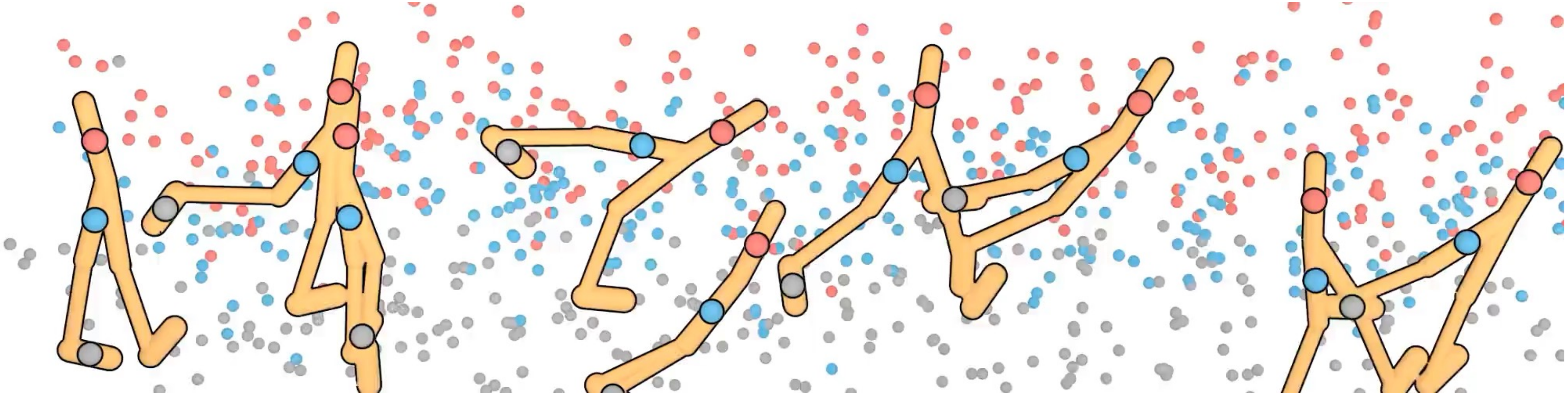


https://dreamfusion3d.github.io/

Beyond Image Generation

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

Beyond Image Generation

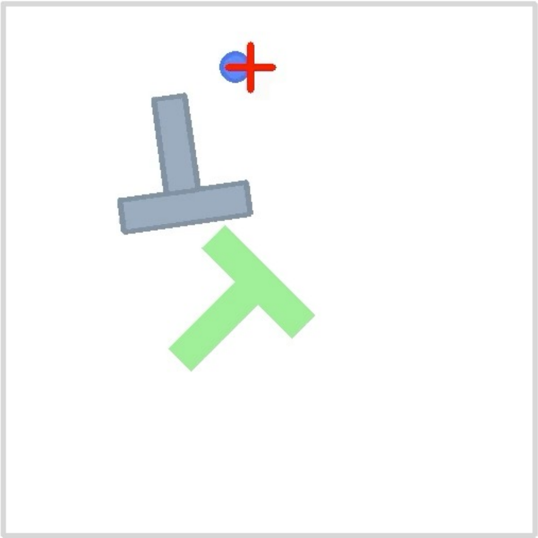


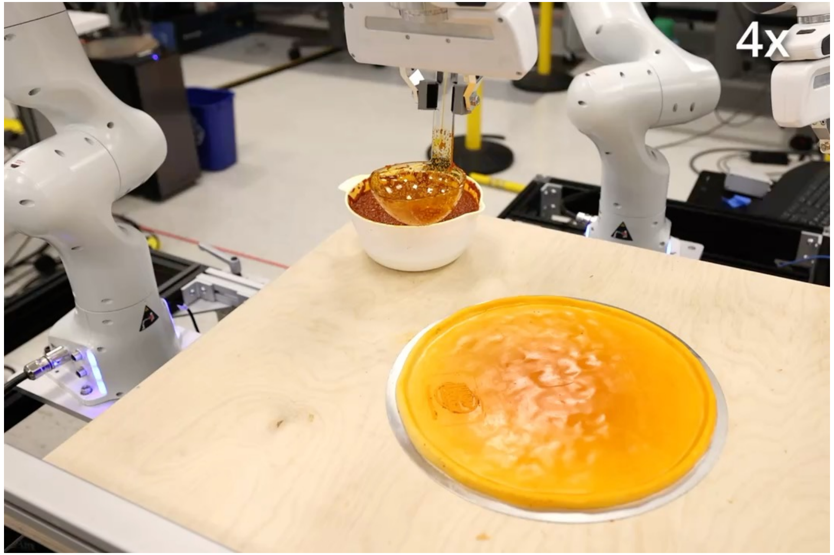
DecisionDiffuser (Ajay, Gupta, Du et al., 2023)
Model future state and reward distributions

$$p(r_{t:t+H}, s_{t:t+H}|s_t)$$

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

Beyond Image Generation





Diffusion Policy (Chi et al., 2023)

Model multimodal action distributions (implement this in your HW4!)

$$p(a_{t:t+H}|s_t)$$

https://diffusion-policy.cs.columbia.edu/

Beyond Image Generation

Generative Skill Chaining (Mishra et al., 2023)

https://diffusion-policy.cs.columbia.edu/

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 video: What are Diffusion Models?
- CVPR Tutorial: Denoising Diffusion-based Generative Modeling: Foundations and Applications

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!