CS 4644-DL / 7643-A: Lecture 17
Danfei Xu

Generative Models:

Denoising Diffusion Probabilistic Models (DDPMs)

Taxonomy of Generative Models

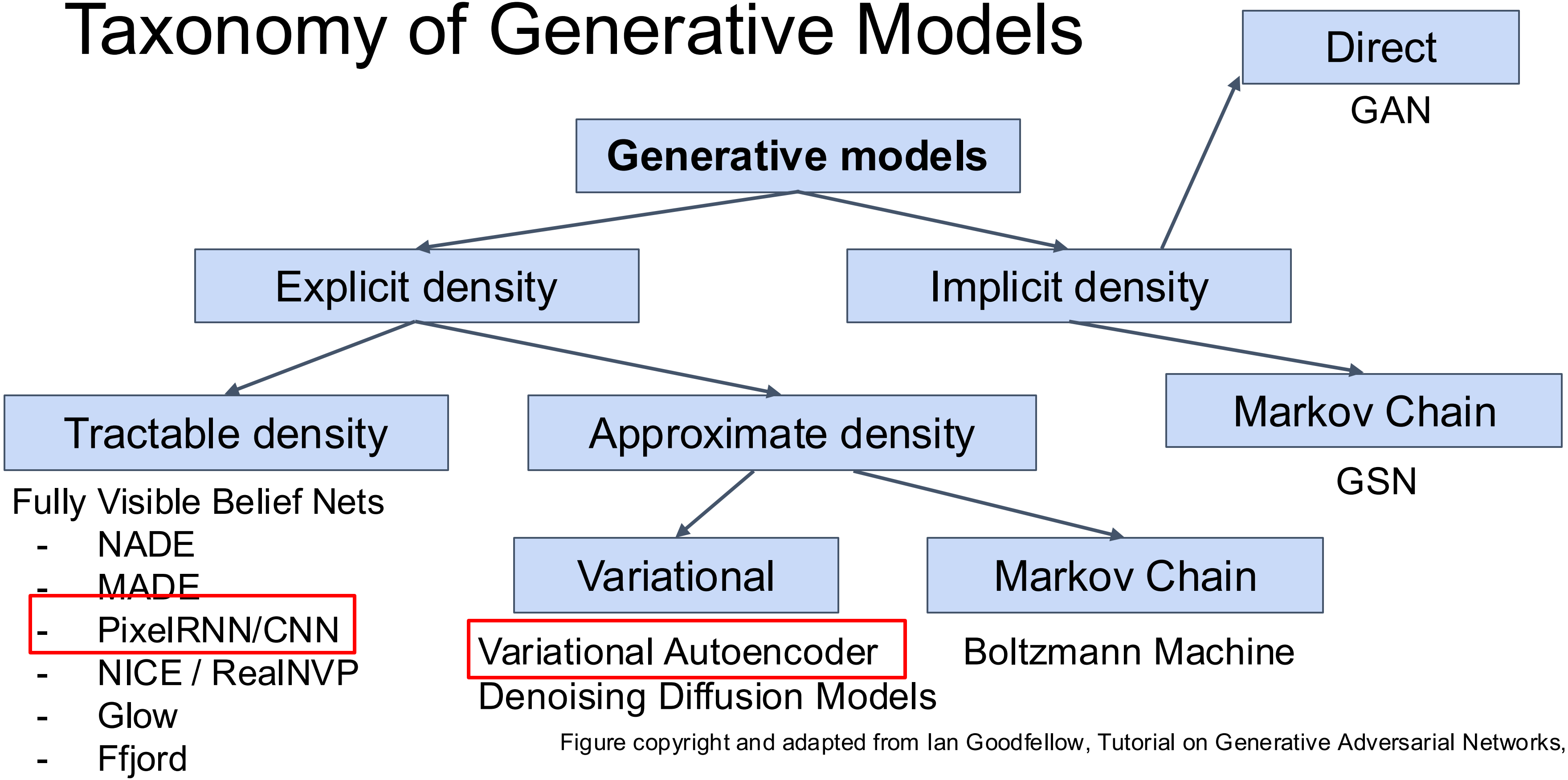


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Fully visible belief network (FVBN)

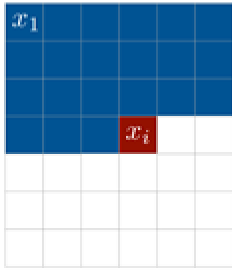
Explicit density model

Use probability chain rule to decompose likelihood of an image x into product of 1-d distributions:

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

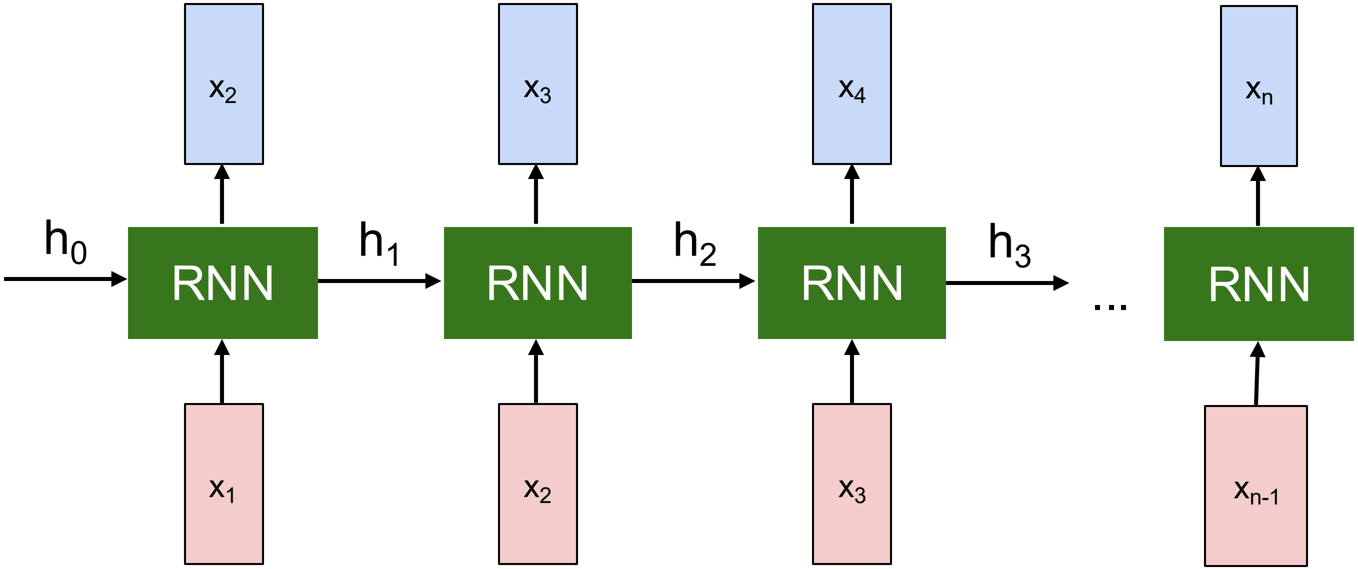
Likelihood of image x

Probability of i'th pixel value given all previous pixels



Then maximize likelihood of training data

Recurrent Neural Network



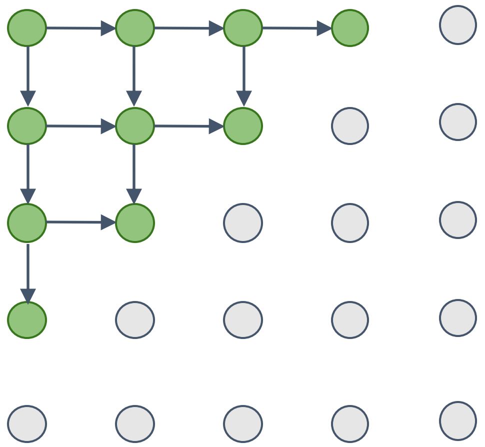
$$p(x_i|x_1, ..., x_{i-1})$$

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

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

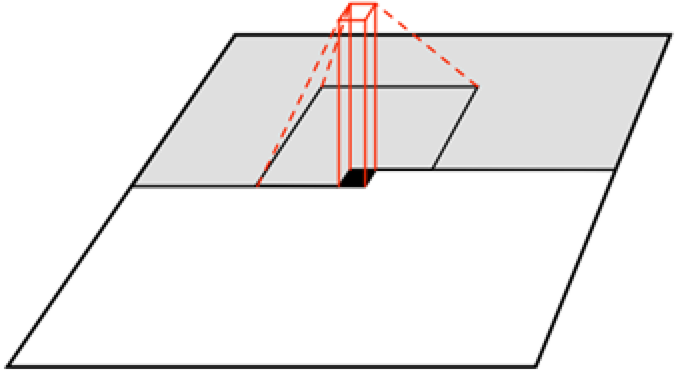
Drawback: sequential generation is slow in both training and inference!



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**)



Taxonomy of Generative Models

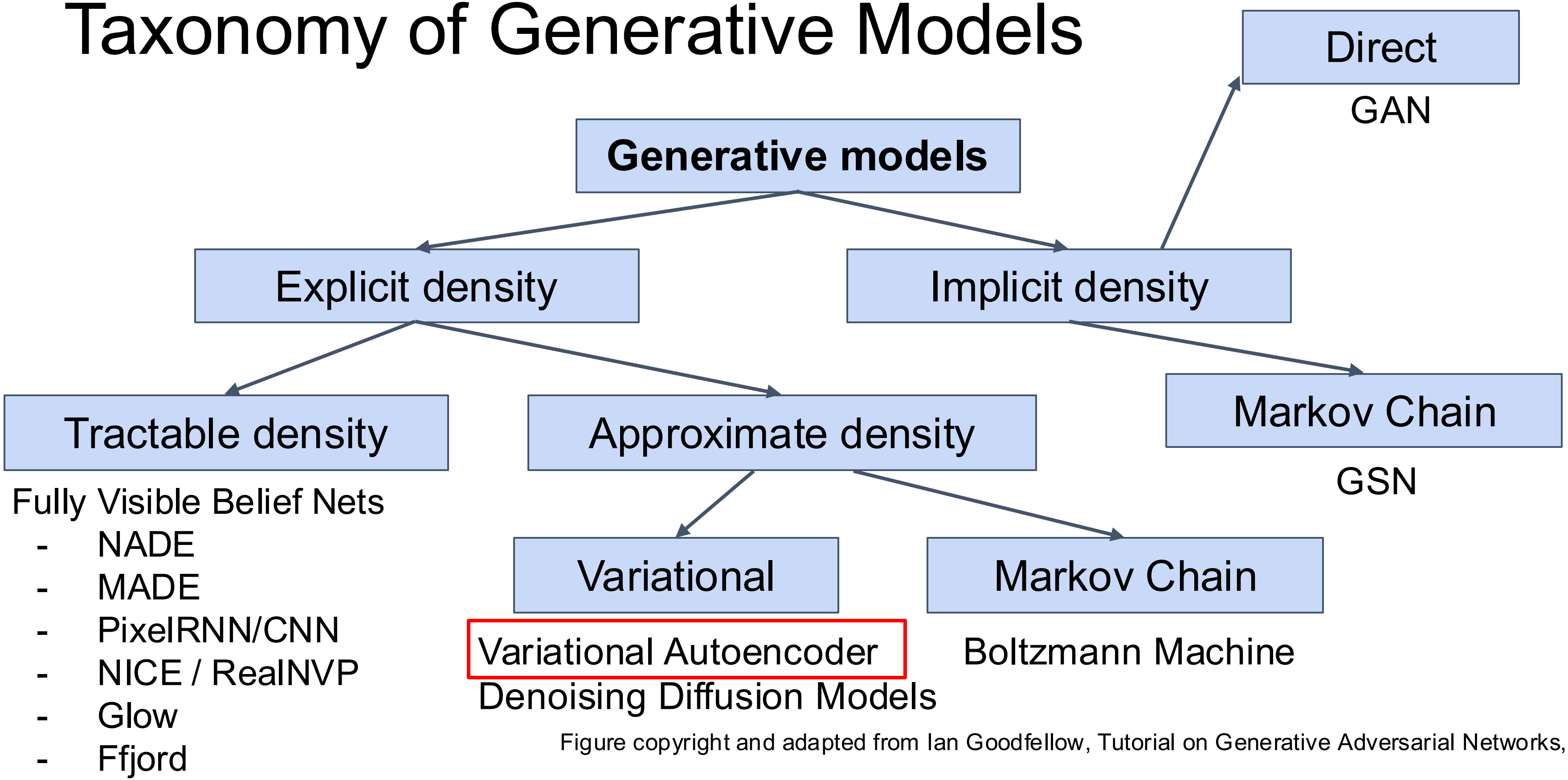


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

So far...

PixelR/CNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

Variational Autoencoders (VAEs) define intractable density function with latent **z**:

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

No dependencies among pixels, can generate all pixels at the same time!

Latent variable z that captures important *factors of variations* in dataset

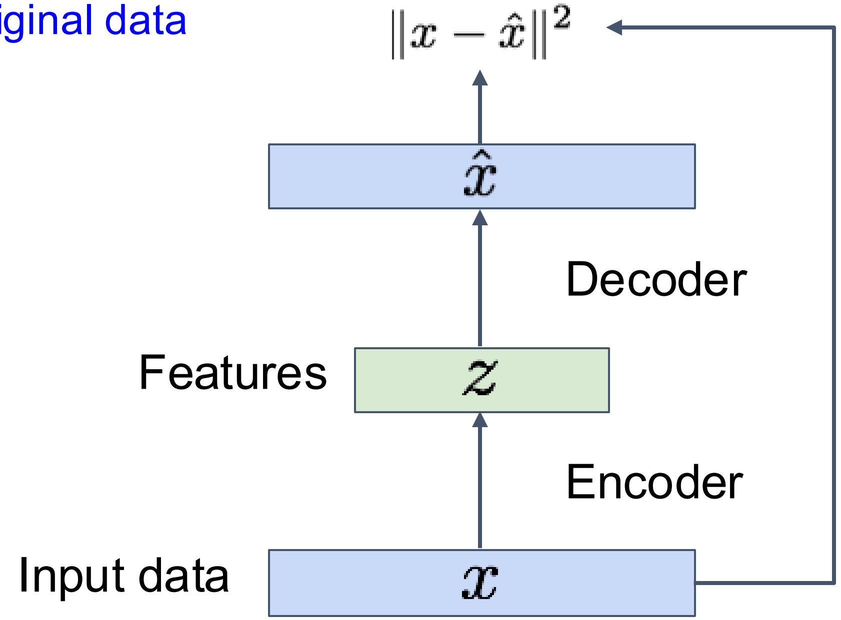
Cannot optimize (maximum likelihood estimation) directly, derive and optimize lower bound on likelihood instead

Some background first: Autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!

L2 Loss function:



Reconstructed data

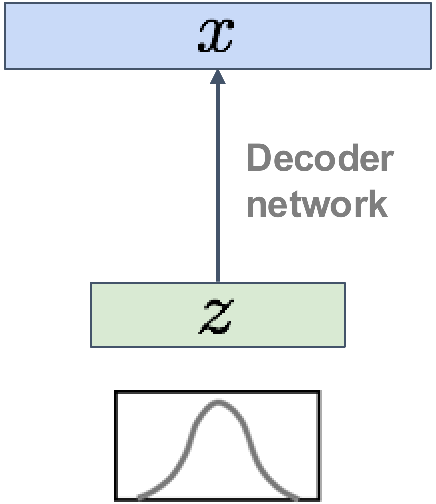
Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data

Variational Autoencoders

Sample from
true conditional
$p_{\theta^*}(x \mid z^{(i)})$

Sample from
true prior
$z^{(i)} \sim p_{\theta^*}(z)$



We want to estimate the true parameters θ^* of this generative model given training data x.

How should we represent this model?

Assume p(z) is *known* and ***simple***, e.g. isotropic Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Conditional p(x|z) is ***complex*** (generates image) => represent with neural network

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

Variational Autoencoders: Intractability

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Can we do Monte Carlo sampling?

$$\log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{where } z^{(i)} \sim p(z)$$

Can we estimate posterior density? Not quite, but …

$$p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$$

VAE: We can use an approximate posterior $q_\theta(z|x)$ (variational distribution) to form a *tractable lower bound* of the data likelihood p(x).

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

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

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

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

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

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

Decoder:
reconstruct
the input data

Encoder:
make approximate
posterior distribution
close to prior

Sample z from the learned posterior (encoder) to train the decoder to reconstruct!

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

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

$$D_{KL}(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) || \mathcal{N}(0, I))$$

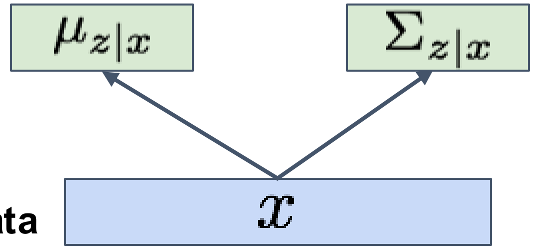
Have analytical solution

Make approximate posterior distribution close to prior

Encoder network

$q_\phi(z|x)$

Input Data



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

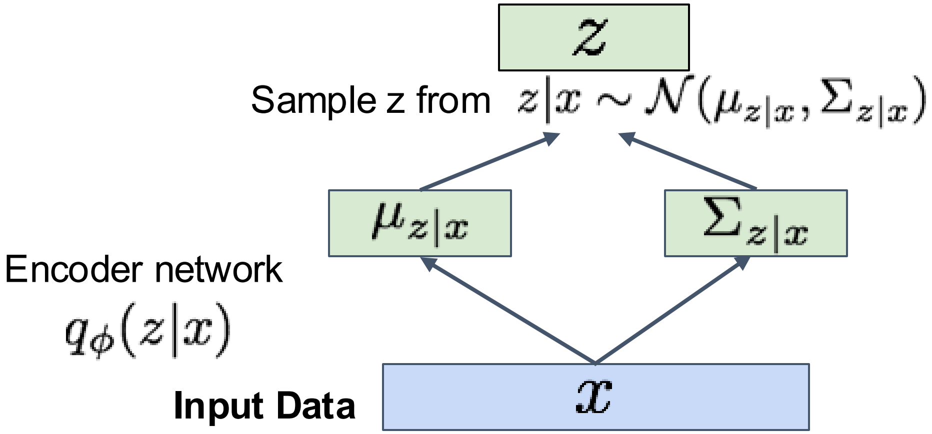
Reparameterization trick to make sampling differentiable:

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

Input to the graph

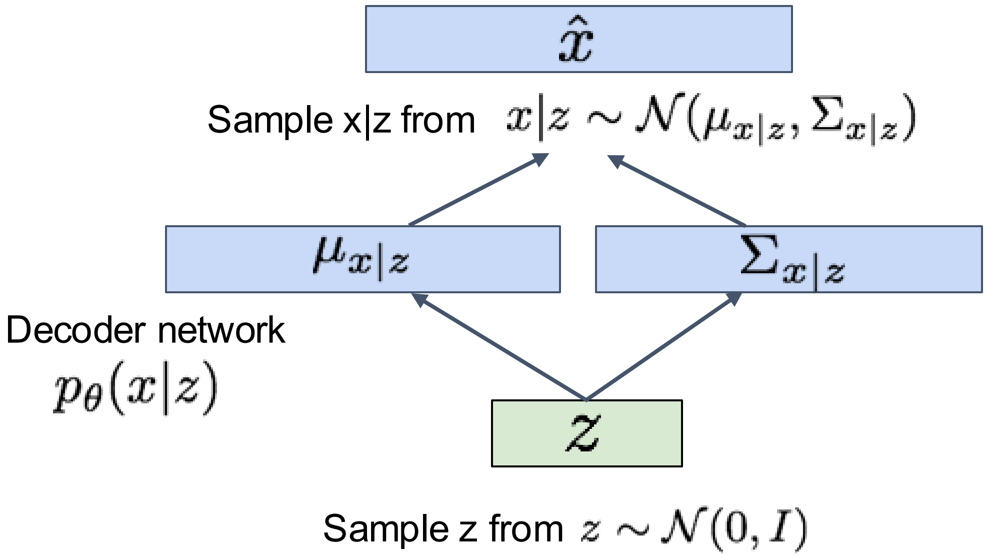
$$z = \boxed{\mu_{z|x}} + \boxed{\epsilon}\boxed{\sigma_{z|x}}$$

Part of computation graph



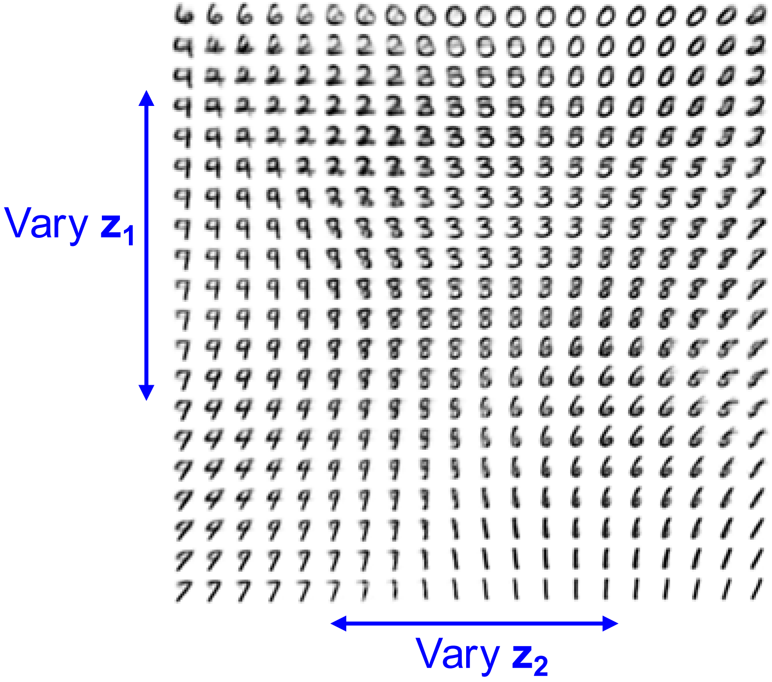
Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Data manifold for 2-d **z**



Variational Autoencoders: Generating Data!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

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

Variational Autoencoders: Generating Data!

32x32 CIFAR-10

Labeled Faces in the Wild

Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data
Defines an intractable density => derive and optimize a (variational) lower bound

Pros:

- Principled approach to generative models
- Latent space z is interpretable and may be useful for other downstream tasks.

Cons:

- Samples are blurry
- KL weights are hard to tune
- Latent distributions are aggressive representation bottlenecks that may limit the expressiveness of the model.

Can be made more powerful by making VAE hierarchical (multiple layers of latents).
Diffusion model (denoising diffusion) can be thought of a type of hierarchical VAE!

Taxonomy of Generative Models

Generative models

- Explicit density
  - Tractable density
    - Fully Visible Belief Nets
      - NADE
      - MADE
      - PixelRNN/CNN
      - NICE / RealNVP
      - Glow
      - Ffjord
  - Approximate density
    - Variational
      - Variational Autoencoder
      - Denoising Diffusion Models
    - Markov Chain
      - Boltzmann Machine
- Implicit density
  - Direct
    - GAN
  - Markov Chain
    - GSN

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/

Seminal Papers

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)

Seminal Papers

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)

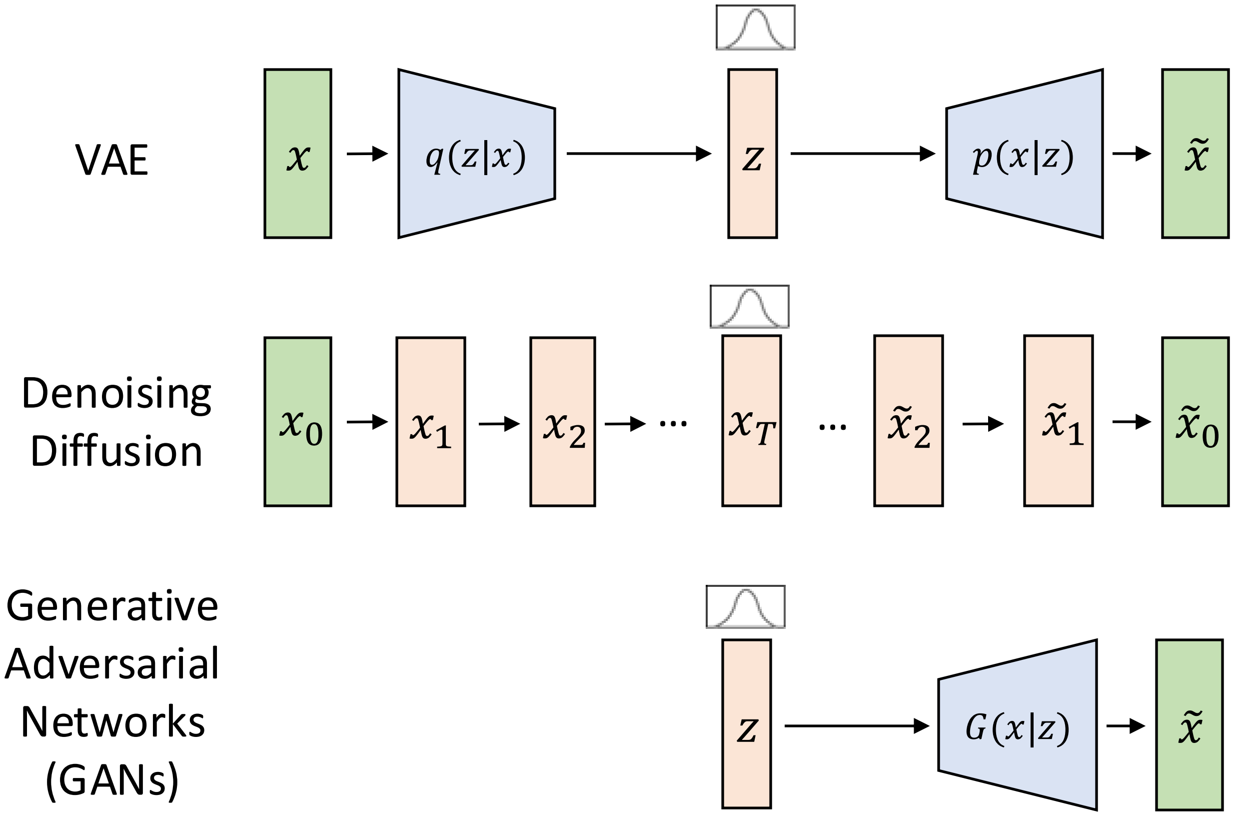
conditional & high-res image generation

- *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

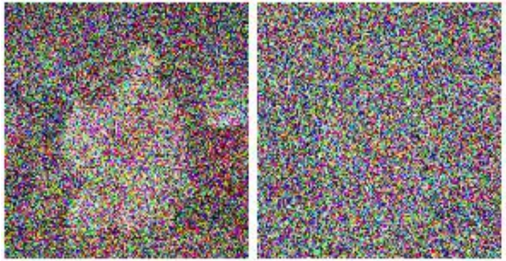
image from dataset

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

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

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

…



The Denoising Diffusion Process

image from dataset

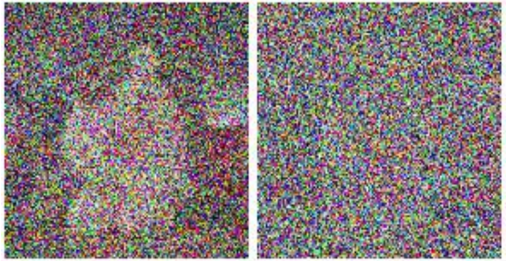
The "forward diffusion" process: add Gaussian noise each step

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

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

…

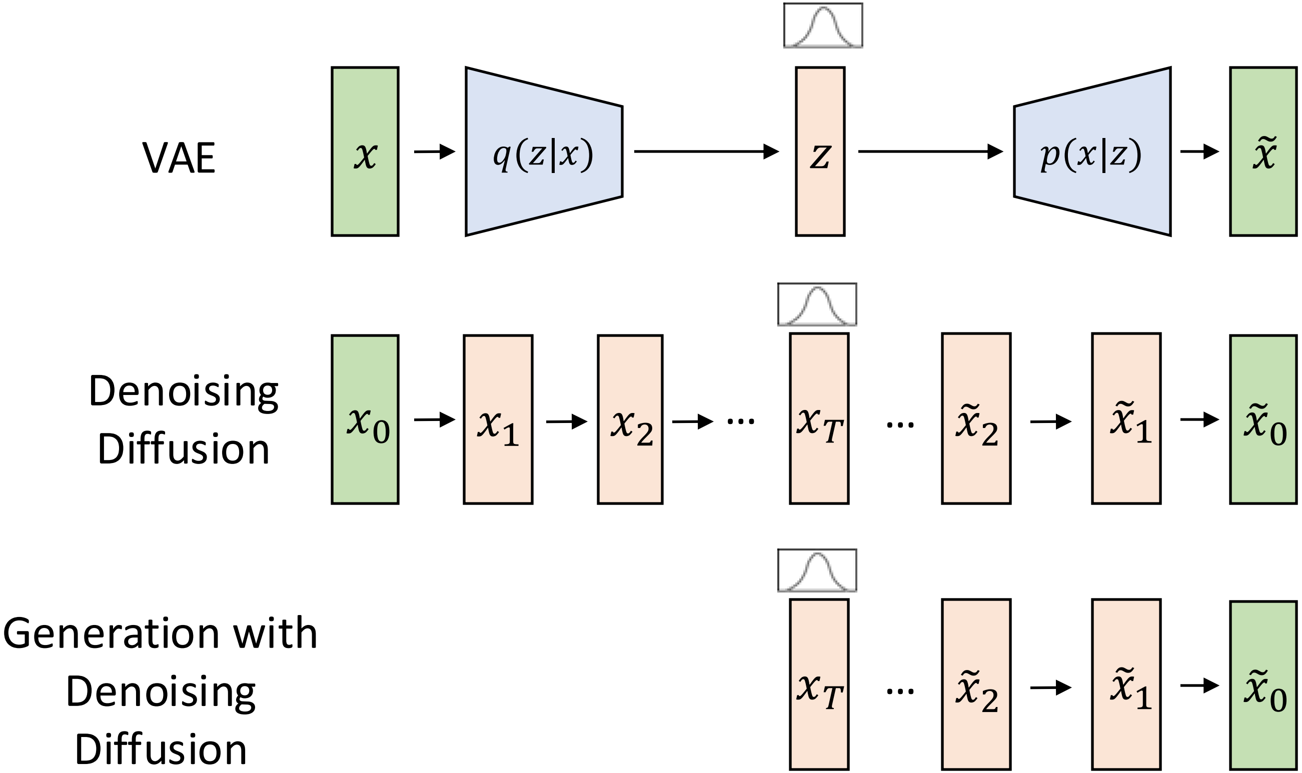
…



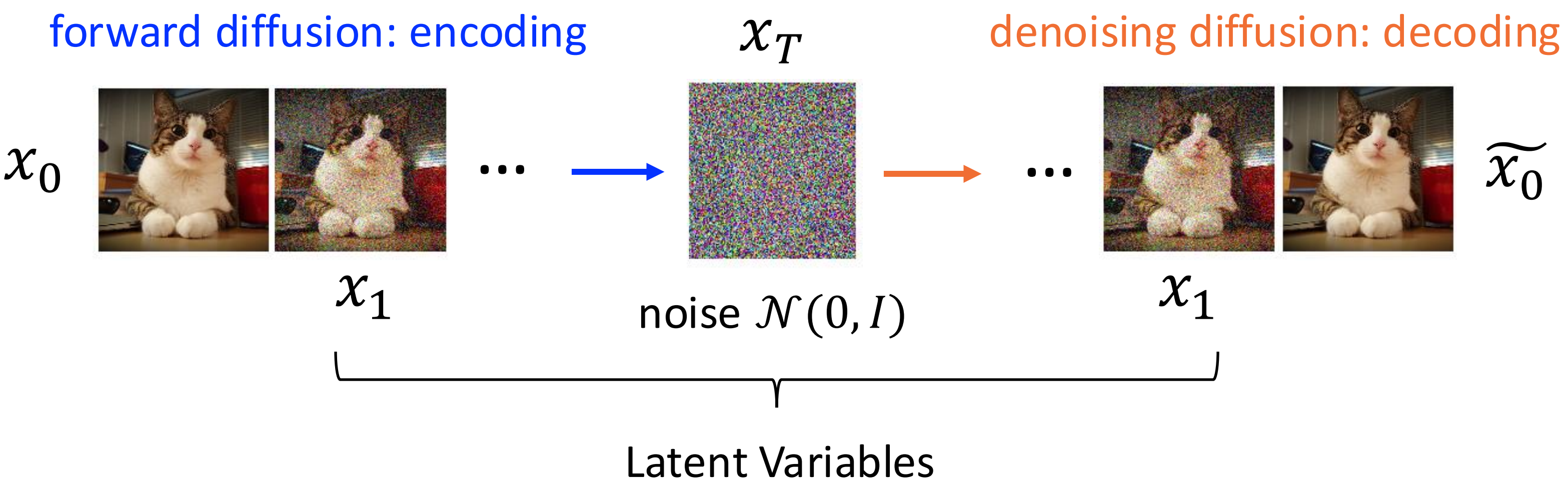
$$x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_{T-1} \longleftarrow x_T$$

The "denoising diffusion" process: generate an image from noise by removing the gaussian noises

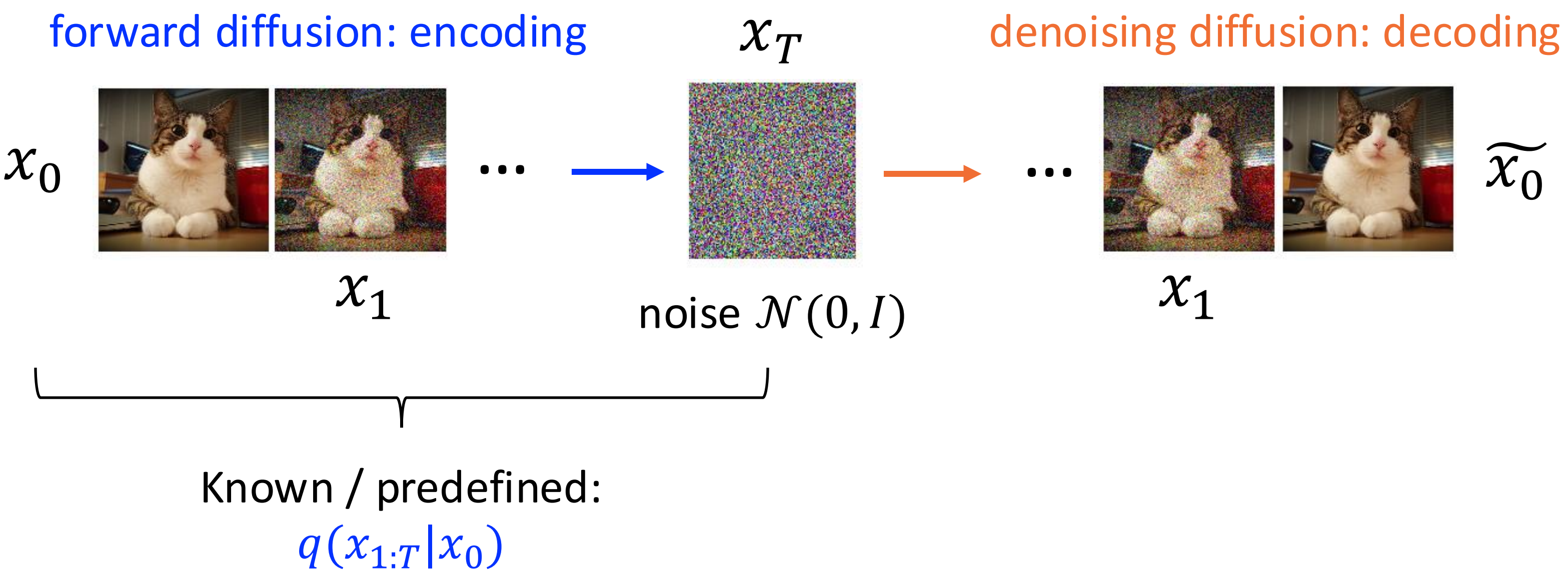
Connection to VAEs



Connection to VAEs

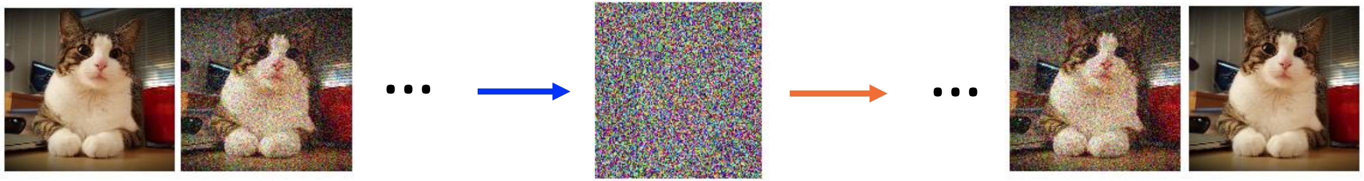


Connection to VAEs



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:

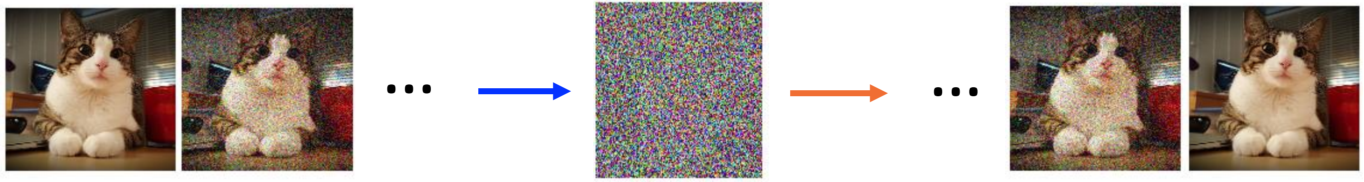
$q(x_{1:T}|x_0)$

Unknown / learned:

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

Connection to VAEs

forward diffusion: encoding $\quad x_T \quad$ denoising diffusion: decoding

x_0  $\tilde{x}_0$

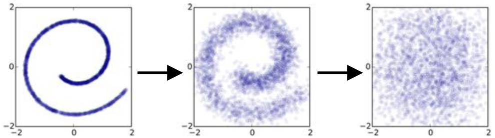
$x_1 \qquad$ noise $\mathcal{N}(0, I) \qquad x_1$

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

Unknown / learned:

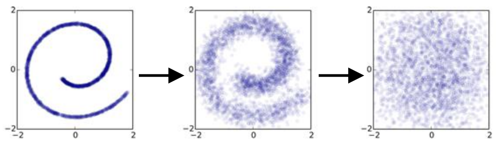
$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

The Diffusion (Encoding) Process



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

The Diffusion (Encoding) Process

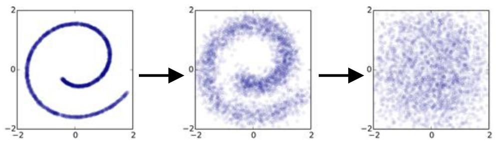


The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

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

Probability Chain Rule (Markov Chain)

The Diffusion (Encoding) Process

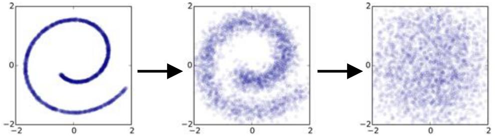


The **known** forward process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ 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 $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

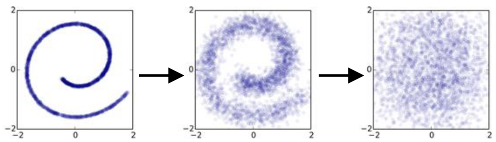
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

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



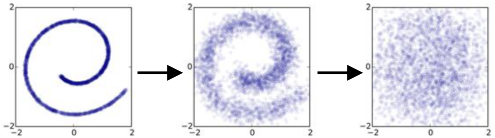
The **known** forward process $\quad x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ 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 $\quad x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$

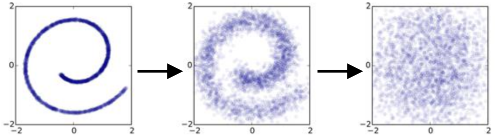
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

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

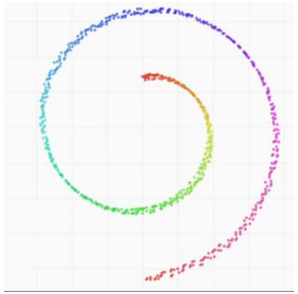
$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$ Probability Chain Rule (Markov Chain)

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

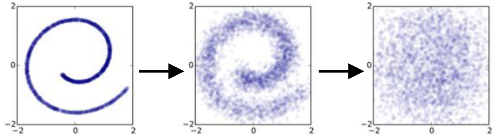
β_t is the *variance schedule* at the diffusion step t

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

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



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

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

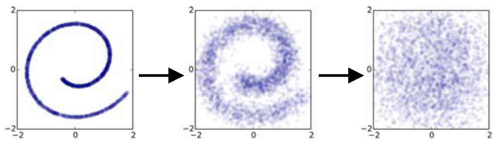
$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

, 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 (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!

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

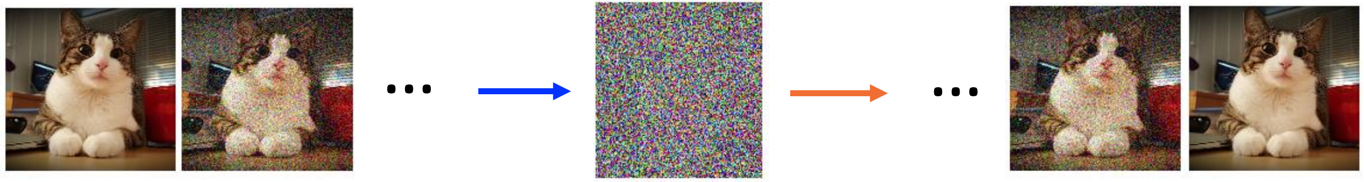
Gaussian reparameterization trick (recall from VAEs!):

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

Intuition: We can directly computed noised sample at arbitrary step *t* without going through the Markov chain

The Diffusion and Denoising Process

forward diffusion: encoding x_T denoising diffusion: decoding

x_0  $\widetilde{x_0}$

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

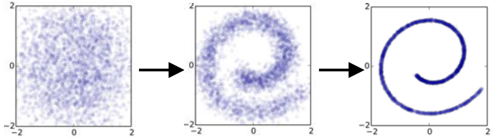
Known / predefined:

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

Unknown / learned:

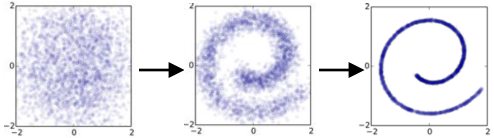
$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

The Denoising (Decoding) Process



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

The Denoising (Decoding) Process

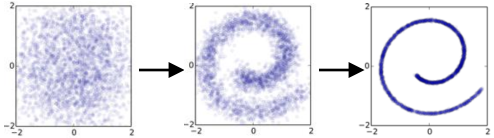


The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

Probability Chain Rule (now reversed)

The Denoising (Decoding) Process

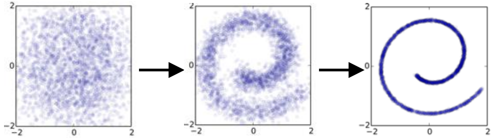


The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \dots \longleftarrow x_T$

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

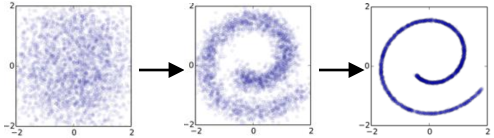
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

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)

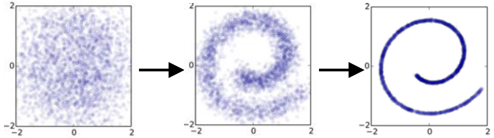
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$ Conditional Gaussian

Want to learn time-dependent mean

Assume fixed / known variance (simplification)

How do we form a learning objective?

The Denoising (Decoding) Process

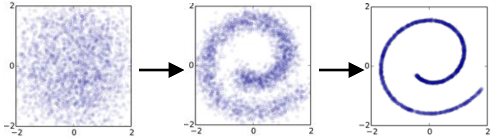


The **learned** denoising process $x_0 \longleftarrow x_1 \longleftarrow \cdots \longleftarrow x_T$

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

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



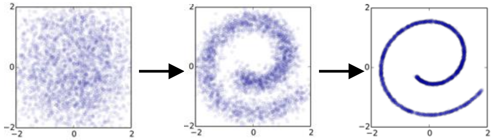
The **learned** denoising process $x_0 \leftarrow x_1 \leftarrow \cdots \leftarrow x_T$

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

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

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

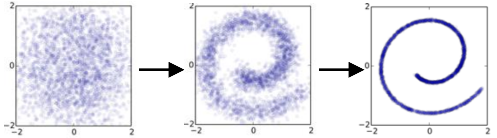
High-level intuition: derive a *ground truth denoising distribution* $q(x_{t-1}|x_t, x_0)$ and train a neural net $p_\theta(x_{t-1}|x_t)$ to match the distribution.

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

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

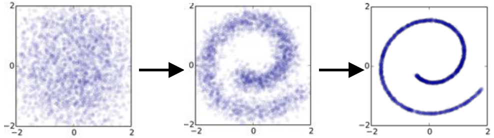
$$\mu_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon\right), \qquad \epsilon \sim \mathcal{N}(0, I) \leftarrow$$

Recall: Gaussian reparameterization trick

Recall:
β_t is the *variance schedule* at the forward diffusion step t
$\alpha_t = 1 - \beta_t, \overline{\alpha_t} = \prod_{s=1}^{t} \alpha_s$

The "ground truth" noise that brought x_{t-1} to x_t .
We know this during training because we took this sample during the forward process!

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

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

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

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

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

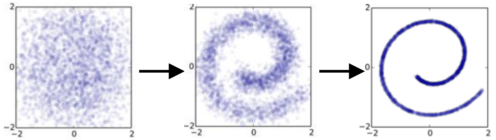
Assuming identical variance $\Sigma_q(t)$, we have:

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

Intuition: learn to estimate the added noise and remove it!

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

The Denoising (Decoding) Process



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

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

Probability Chain Rule (Markov Chain)

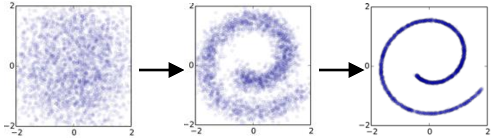
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

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)

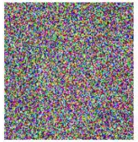
$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_q(t))$$

Conditional Gaussian

We know how to learn

Assume fixed / known variance

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



$\xrightarrow{p_\theta(x_T|x_{T-1})}$

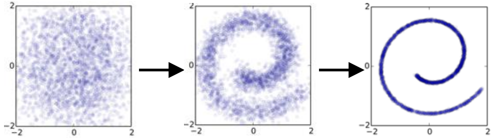
x_{T-1}

$\xrightarrow{p_\theta(x_{T-1}|x_{T-2})} \cdots \xrightarrow{p_\theta(x_1|x_0)}$

x_0

Generate new images!

The Denoising (Decoding) Process



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

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

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

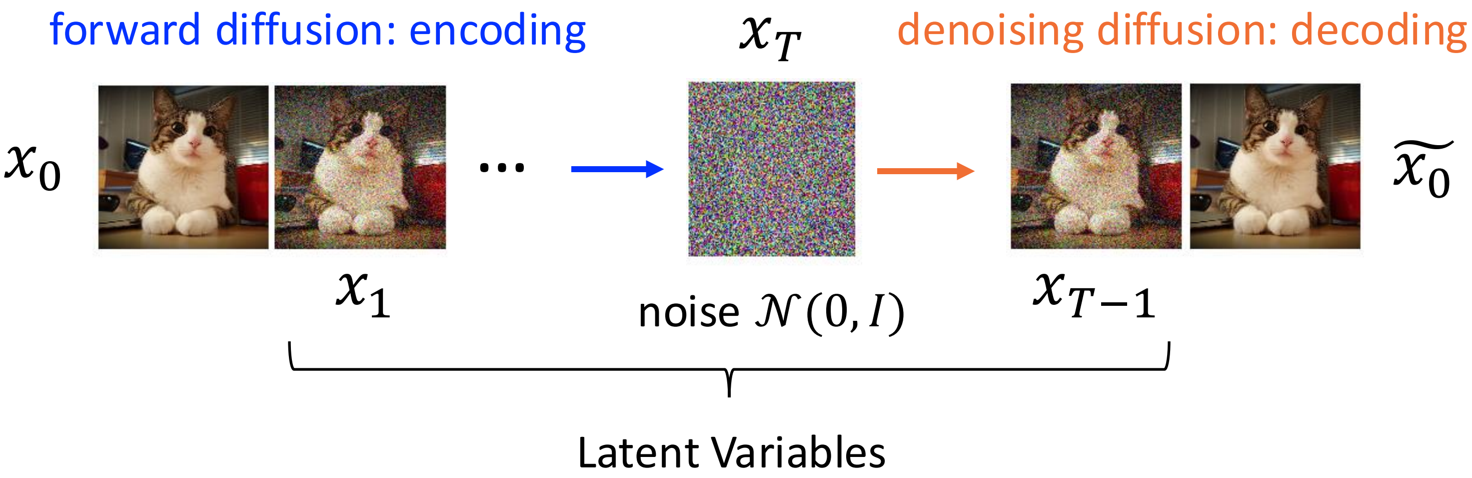
We know how to learn

Assume fixed / known variance

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!

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

Known forward noise (posterior)
$q(x_{1:T}|x_0)$

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)

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

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

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

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

… (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)

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

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

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

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

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)

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

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

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

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

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

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)

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

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

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

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

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

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

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

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

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

Proof using bayes rule and gaussian reparameterization trick

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

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

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

Proof using bayes rule and gaussian reparameterization trick

The “ground truth” noise that brought x_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)

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

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

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

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

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

Minimize the difference of distribution means (assuming identical variance)

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

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

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

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

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

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

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

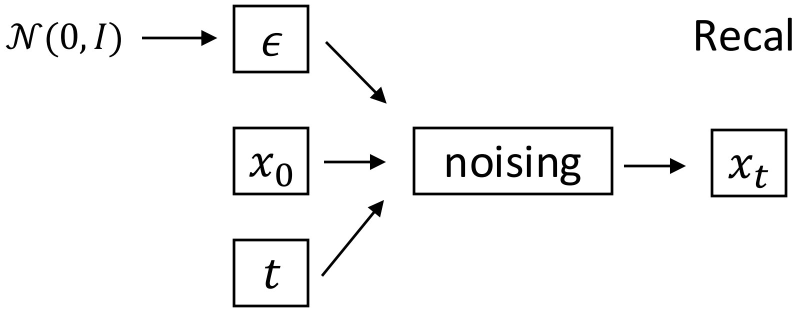
Inference time: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{(1-\bar{\alpha}_t)}}\epsilon_\theta(x_t, t)\right)$

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:

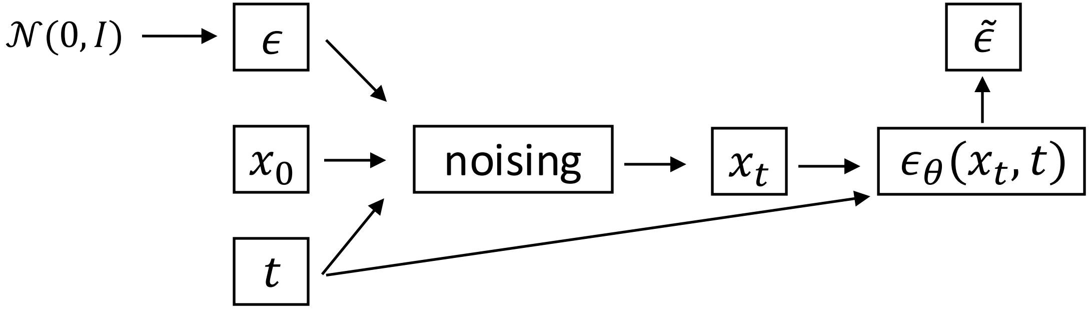
$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$$

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

The Denoising Diffusion Algorithm

Algorithm 1 Training

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



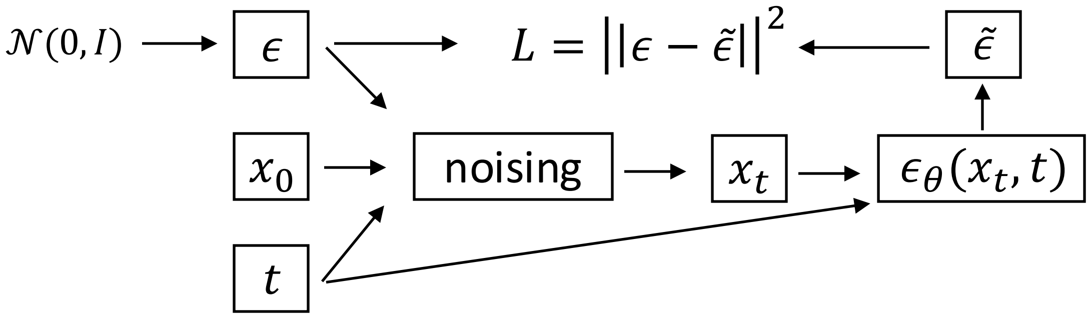
Predict noise with a denoiser network ϵ_θ

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

The Denoising Diffusion Algorithm

Algorithm 1 Training

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



Compute regression loss

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

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

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

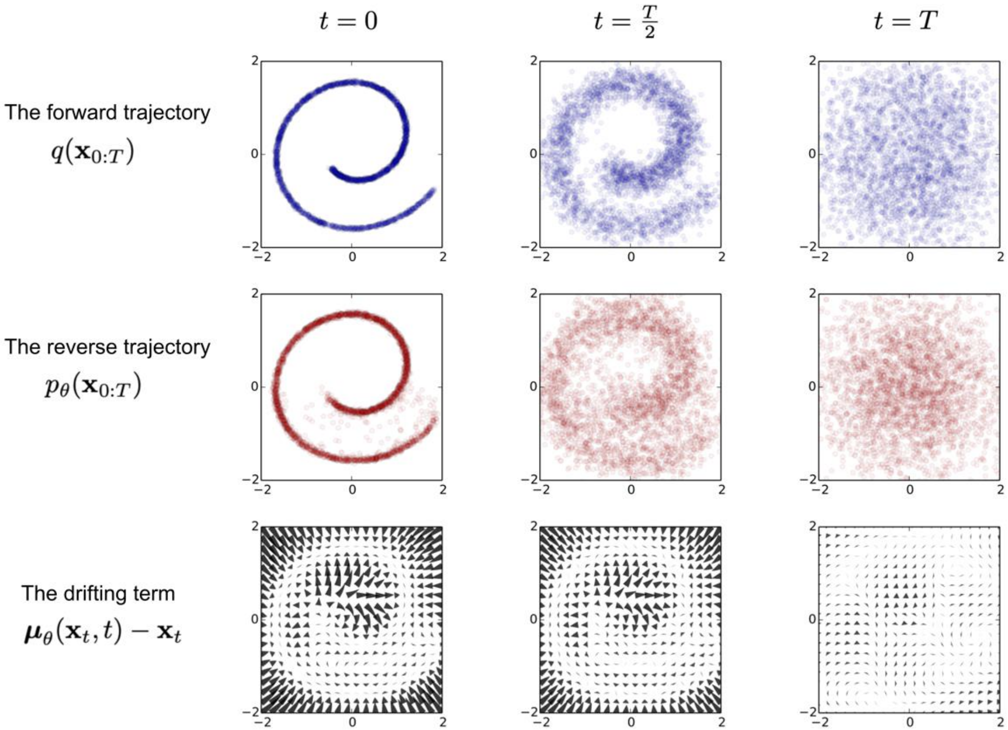
Algorithm 2 Sampling

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

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

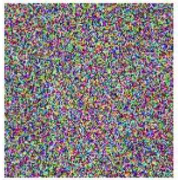
$$\begin{tikzcd} \boxed{x_t} \arrow[r] & \boxed{\epsilon_\theta(x_t, t)} \arrow[r] & \boxed{\mu_t} \\ \boxed{t} \arrow[ur] & & \end{tikzcd}$$

Visualizing the Diffusion Process on 2D data



Sohl-Dickstein *et al.*, 2015

Conditional Diffusion Models

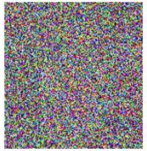


An astronaut riding a horse in a photorealistic style

Conditional Diffusion

$p(x|y)$

Conditional Diffusion Models



An astronaut riding a horse in a photorealistic style

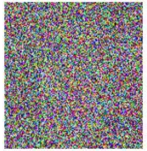
Conditional Diffusion

$p(x|y)$

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

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

Conditional Diffusion Models



An astronaut riding a horse in a photorealistic style

Conditional Diffusion

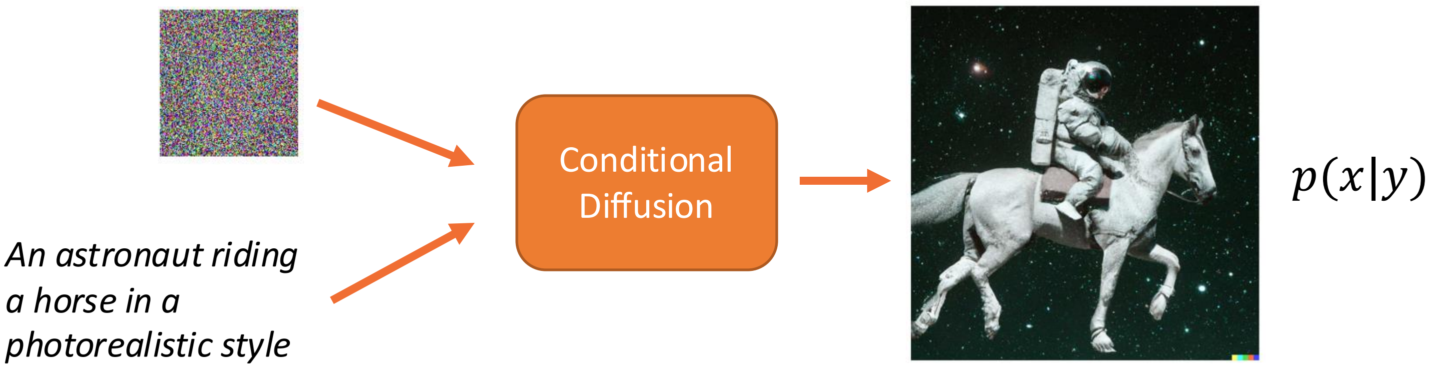
$p(x|y)$

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

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

Problem: Very blurry generation

Classifier-guided Diffusion

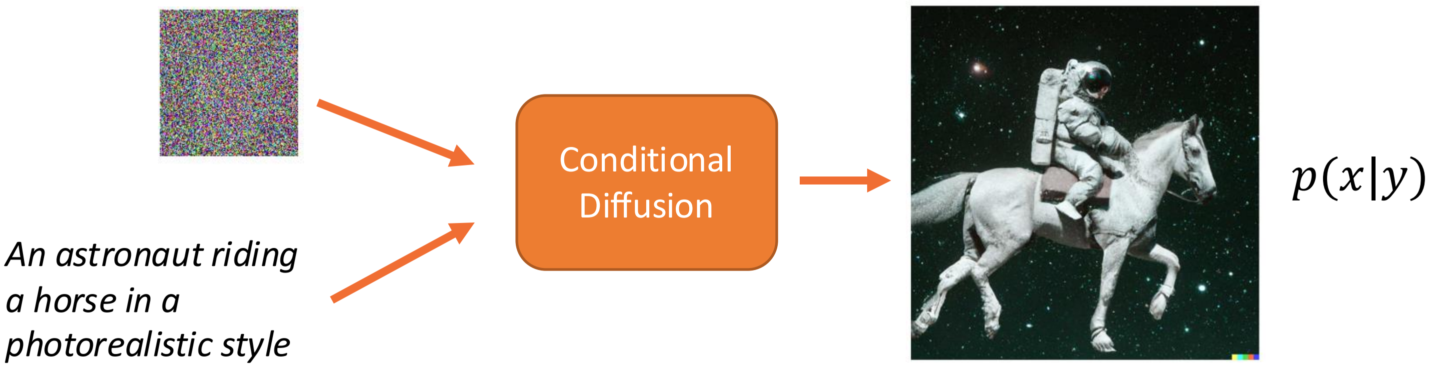


Better idea: use the *gradients* from an image captioning model $f_\varphi(y|x_t)$ to guide the diffusion process!

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

Dhariwal & Nichol, 2021

Classifier-guided Diffusion



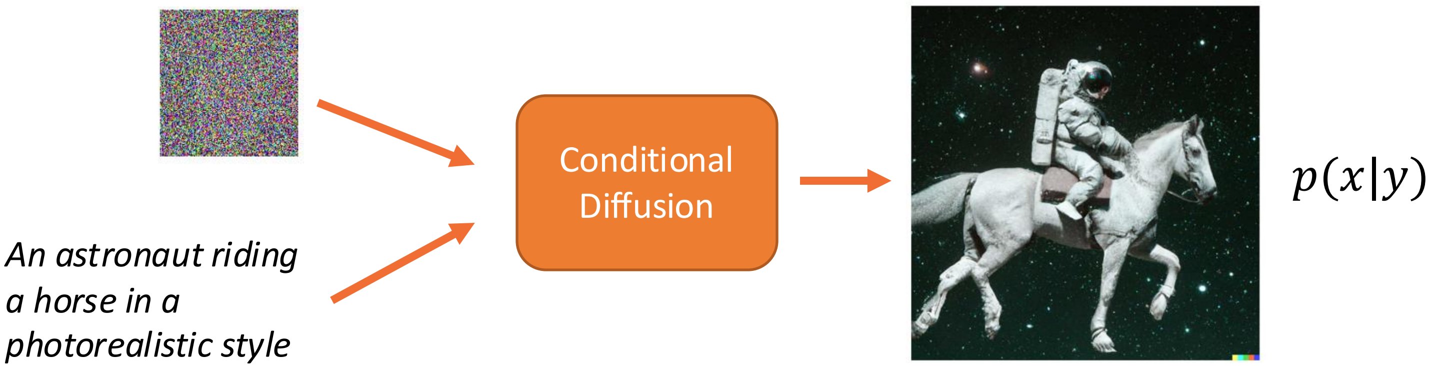
Better idea: use the *gradients* from an image captioning model $f_\varphi(y|x_t)$ to guide the diffusion process!

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

Problem: need a classifier

Dhariwal & Nichol, 2021

Classifier-free Guided Diffusion

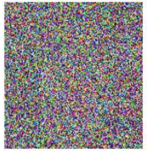


Classifier-free Guided Diffusion: estimate the gradient of the classifier model with conditional diffusion models!

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

Ho and Salimans, 2022

Classifier-free Guided Diffusion



An astronaut riding a horse in a photorealistic style

Conditional Diffusion

$p(x|y)$

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

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

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

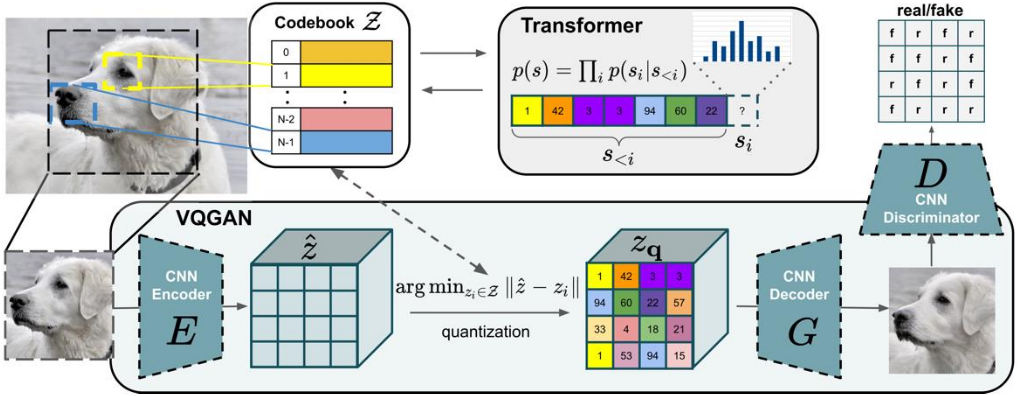
Linearly combine denoisers from an unconditional distribution and a conditional distribution

Ho and Salimans, 2022

Latent-space Diffusion

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

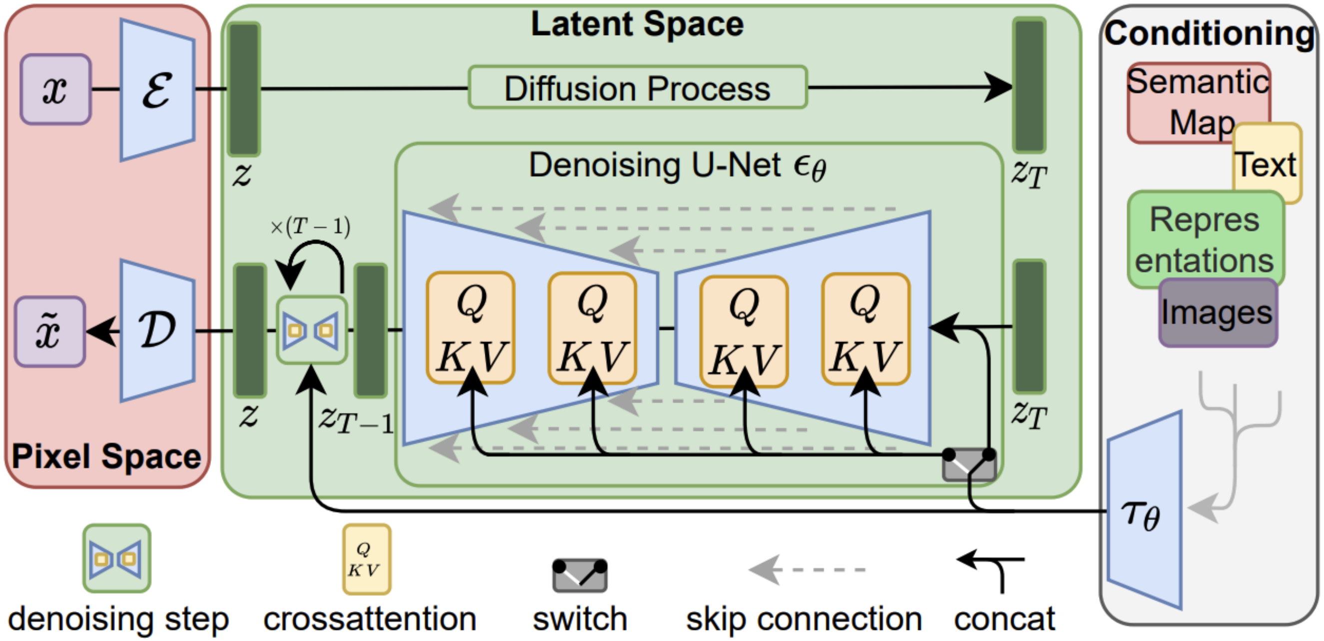
Solution: learn a low-dimensional latent space using a ViT-based autoencoder and *do diffusion on the latent space*!



The latent space autoencoder

Esser and Rombach *et al.*, 2021

“StableDiffusion”



Rombach and Blattmann *et al.*, 2022

“StableDiffusion”

Layout-Conditional Generation

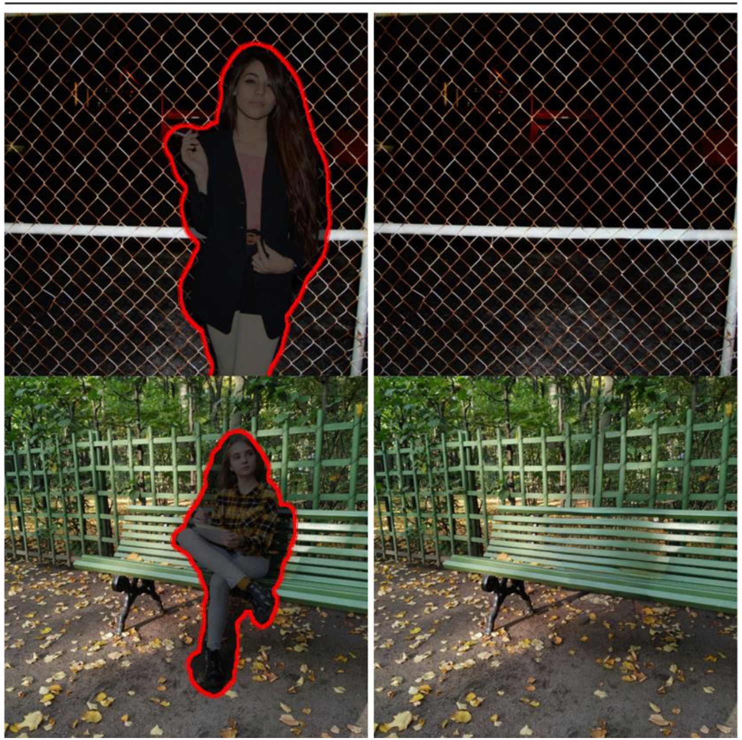
Rombach and Blattmann *et al.*, 2022

“StableDiffusion”

Segmentation-Conditional Generation

Rombach and Blattmann *et al.*, 2022

“StableDiffusion”



Inpainting

Rombach and Blattmann *et al.*, 2022

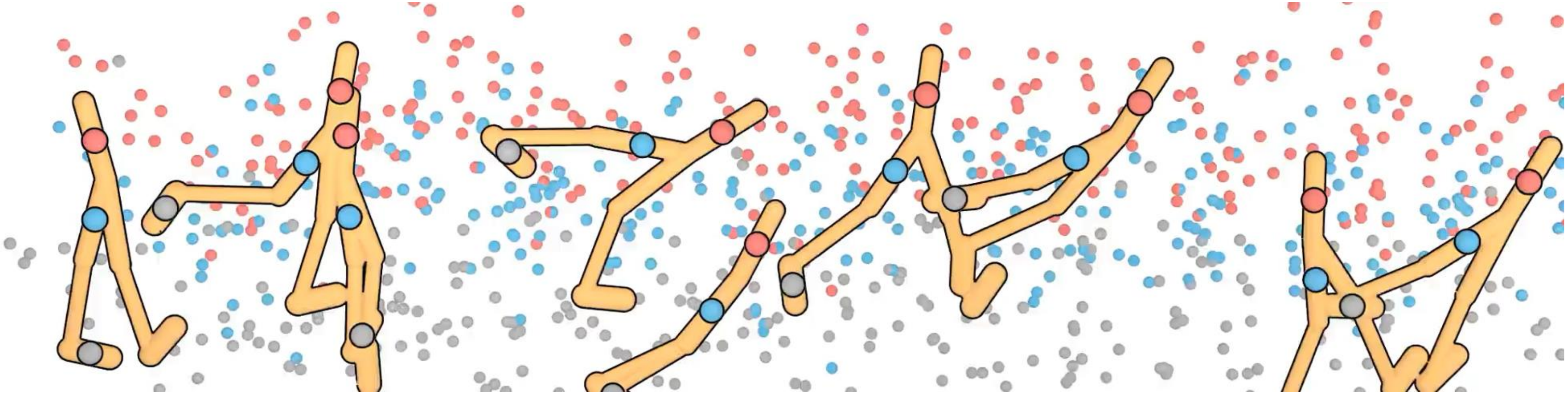
Beyond Image Generation

https://dreamfusion3d.github.io/

Beyond Image Generation

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

Beyond Image Generation

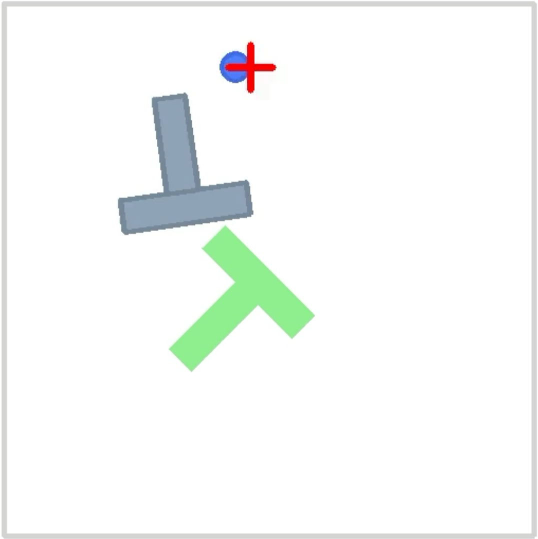


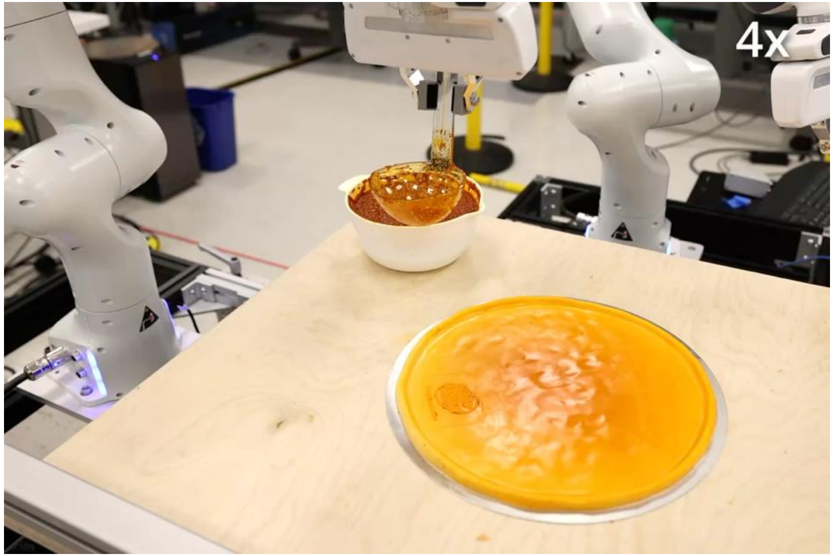
DecisionDiffuser (Ajay, Gupta, Du et al., 2023)
Model future state and reward distributions

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

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!