DDA4210/AIR6002 Advanced Machine Learning
Lecture 08 Generative Models

Tongxin Li

School of Data Science, CUHK-Shenzhen

Spring 2024

Overview

Generative Models

- Generative models: generate new data instances with similar distribution as the training data
  - Learn a probability distribution $p(\mathbf{x})$ from $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
  - Then sample from $p(\mathbf{x})$ to generate new data instances
- Deep Generative Models (DGMs) are formed through the combination of generative models and deep neural networks.
- DGMs achieved SOTA performances in many real cases (e.g., image generation, text generation, ChatGPT, etc)

Generative Models

- Generative models: generate new data instances with similar distribution as the training data
  - Learn a probability distribution $p(\mathbf{x})$ from $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
  - Then sample from $p(\mathbf{x})$ to generate new data instances
- Deep Generative Models (DGMs) are formed through the combination of generative models and deep neural networks.
- DGMs achieved SOTA performances in many real cases (e.g., image generation, text generation, ChatGPT, etc)
- Types of Deep Generative Models
  - Variational AutoEncoder (VAE)
  - Generative Adversarial Networks (GANs)
  - Diffusion Models
  - $\cdots$

Generation instances via deep generative models

NVAE(2020)

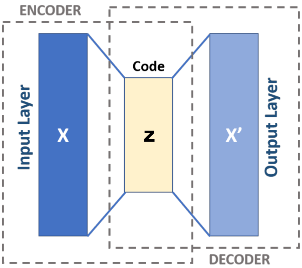
DDPM(2020)

StyleGAN(2018)

- Vahdat Arash and Jan Kautz. "NVAE: A Deep Hierarchical Variational Autoencoder". NeurIPS 2020.
- Ho Jonathan et al. "Denoising Diffusion Probabilistic Models", NeurIPS 2020.
- Karras Tero et al. "A Style-Based Generator Architecture for Generative Adversarial Networks". CVPR 2021.

AutoEncoder (AE)

- AutoEncoder (AE) is a type of neural network designed to learn an approximate identity transformation using an unsupervised way and then to reconstruct high-dimensional data and consists of an encoder network f_ϕ and a decoder network g_θ, parameterized by ϕ, θ respectively.
- The middle layer of AE usually has a narrow bottleneck to compress original high-dimensional data to low-dimensional representations.

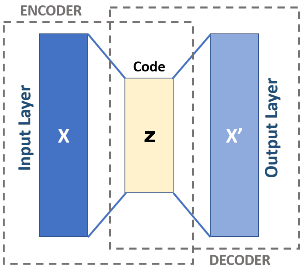


- Encoder network $f_\phi : \mathbf{x} \to \mathbf{z}$
- Decoder network $g_\theta : \mathbf{z} \to \mathbf{x}'$
- Optimization objective:

$$\min_{\phi,\theta} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - g_\theta(f_\phi(\mathbf{x}_i))\|^2 + \mathcal{R}(\phi, \theta)$$

AutoEncoder (AE)

- AutoEncoder (AE) is a type of neural network designed to learn an approximate identity transformation using an unsupervised way and then to reconstruct high-dimensional data and consists of an encoder network f_ϕ and a decoder network g_θ, parameterized by ϕ, θ respectively.
- The middle layer of AE usually has a narrow bottleneck to compress original high-dimensional data to low-dimensional representations.



- Encoder network $f_\phi : \mathbf{x} \to \mathbf{z}$
- Decoder network $g_\theta : \mathbf{z} \to \mathbf{x}'$
- Optimization objective:

$$\min_{\phi,\theta} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - g_\theta(f_\phi(\mathbf{x}_i))\|^2 + \mathcal{R}(\phi, \theta)$$

- Can we generate new data using AE?

Image from Wikipedia.

Variational AutoEncoder (VAE): Motivation

- VAE aims to transform $\mathbf{x}$ into a prior distribution p_z (rather than a fixed vector $\mathbf{z}$) using encoder f_ϕ and then to reconstruct $\mathbf{x}$ using decoder g_θ.
- Given training data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and a prior distribution p_z. Assume we have trained f_ϕ and g_θ of VAE successfully. In order to generate a new sample that looks like a real data point $\mathbf{x}_i$, we need the following steps:
    - First, sample a $\mathbf{z}_i$ from the prior distribution p_z.
    - Then, a new sample can be generated via the decoder, i.e., $g_\theta(\mathbf{z}_i)$.

Variational AutoEncoder (VAE): Motivation

- VAE aims to transform $\mathbf{x}$ into a prior distribution p_z (rather than a fixed vector $\mathbf{z}$) using encoder f_ϕ and then to reconstruct $\mathbf{x}$ using decoder g_θ.
- Given training data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and a prior distribution p_z. Assume we have trained f_ϕ and g_θ of VAE successfully. In order to generate a new sample that looks like a real data point $\mathbf{x}_i$, we need the following steps:
    - First, sample a $\mathbf{z}_i$ from the prior distribution p_z.
    - Then, a new sample can be generated via the decoder, i.e., $g_\theta(\mathbf{z}_i)$.
- How to obtain the decoder g_θ? Maximize the probability of generating real data samples (maximum likelihood):

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p(\mathbf{x}_i|\theta)$$

For simplicity, $p(\mathbf{x}_i|\theta)$ abbreviates as $p_\theta(\mathbf{x}_i)$.

Variational AutoEncoder (VAE): Maximum Likelihood

- Compute the marginal likelihood

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

Variational AutoEncoder (VAE): Maximum Likelihood

- Compute the marginal likelihood

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

  - Prior $p_\theta(\mathbf{z})$, e.g. $\mathcal{N}(\mathbf{0}, \mathbf{I})$
  - Likelihood $p_\theta(\mathbf{x}|\mathbf{z})$
  - But it is impossible to integrate over all $\mathbf{z}$.

- How about using Bayes' theorem?

$$p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}$$

  - $p_\theta(\mathbf{z}|\mathbf{x})$ cannot be computed.

Variational AutoEncoder (VAE): Maximum Likelihood

- Compute the marginal likelihood

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

  - Prior $p_\theta(\mathbf{z})$, e.g. $\mathcal{N}(\mathbf{0}, \mathbf{I})$
  - Likelihood $p_\theta(\mathbf{x}|\mathbf{z})$
  - But it is impossible to integrate over all $\mathbf{z}$.
- How about using Bayes' theorem?

$$p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}$$

  - $p_\theta(\mathbf{z}|\mathbf{x})$ cannot be computed.
- Solution: Train another neural network (encoder) f_ϕ that learns

$$q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$$

Variational AutoEncoder (VAE): Maximum Likelihood

- Decompose the log-likelihood:

$$\log p_\theta(\mathbf{x}) = \log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} = \log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})q_\phi(\mathbf{z}|\mathbf{x})}$$

$$= \log p_\theta(\mathbf{x}|\mathbf{z}) - \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} + \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}$$

- Take expectation:

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x})] = \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) d\mathbf{z}$$

$$= \int q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) d\mathbf{z} - \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} d\mathbf{z}$$

$$+ \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

$$= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

$$+ D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}))$$

Variational AutoEncoder (VAE): Evidence Lower Bound

- We have got

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$

- Because KL-divergence is always non-negative, we obtain

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) \triangleq \mathcal{L}_{\phi,\theta}(\mathbf{x})$$

Variational AutoEncoder (VAE): Evidence Lower Bound

- We have got

  $\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$

- Because KL-divergence is always non-negative, we obtain

  $$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) \triangleq \mathcal{L}_{\phi,\theta}(\mathbf{x})$$

- $\mathcal{L}_{\phi,\theta}(\mathbf{x})$ is a lower bound (called evidence lower bound, ELBO) of $\log p_\theta(\mathbf{x})$ and

  $$\log p_\theta(\mathbf{x}) = \mathcal{L}_{\phi,\theta}(\mathbf{x}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$

  ELBO is also known as the variational lower bound.

Variational AutoEncoder (VAE): Evidence Lower Bound

- We have got

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$

- Because KL-divergence is always non-negative, we obtain

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) \triangleq \mathcal{L}_{\phi,\theta}(\mathbf{x})$$

- $\mathcal{L}_{\phi,\theta}(\mathbf{x})$ is a lower bound (called evidence lower bound, ELBO) of $\log p_\theta(\mathbf{x})$ and

$$\log p_\theta(\mathbf{x}) = \mathcal{L}_{\phi,\theta}(\mathbf{x}) + D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))$$

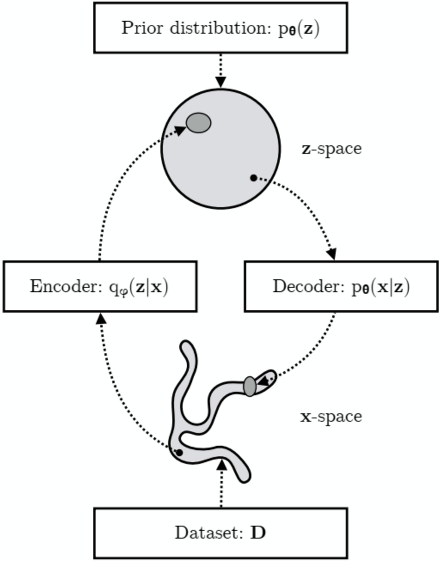
ELBO is also known as the variational lower bound.

- VAE maximizes ELBO, i.e.,

$$\phi^*, \theta^* = \operatorname*{argmax}_{\phi,\theta} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

When $\mathcal{L}_{\phi,\theta}(\mathbf{x}) = \log p_\theta(\mathbf{x})$, it holds that $q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$.

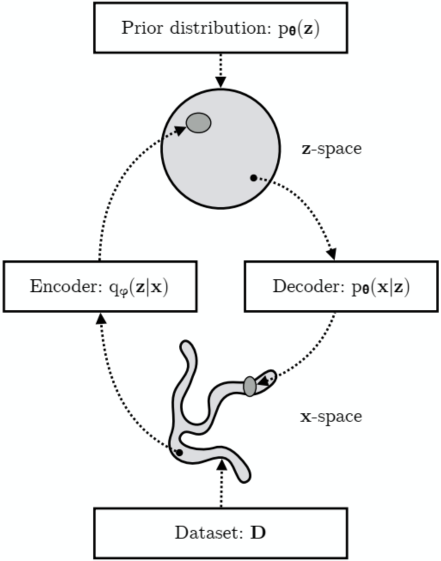
Variational Auto-Encoder (VAE): Optimization



$$\max_{\phi,\theta} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

- maximize $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$: reconstruct $\mathbf{x}$
- minimize $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$: approximate prior

Variational Auto-Encoder (VAE): Optimization

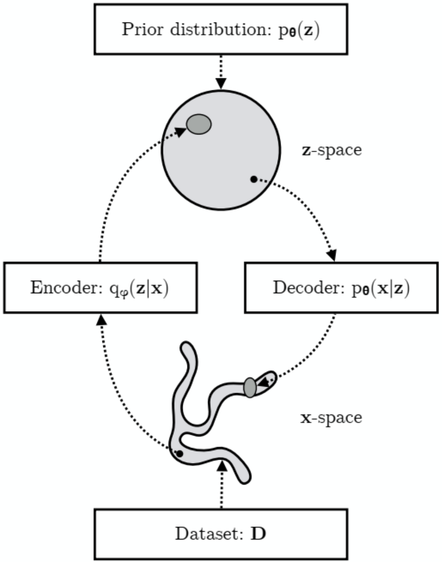


$$\max_{\phi,\theta} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

- maximize $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$: reconstruct $\mathbf{x}$
- minimize $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$: approximate prior

Suppose $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Then let $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_\phi(\mathbf{x}), \text{diag}(\sigma^2_\phi(\mathbf{x})))$.

Variational Auto-Encoder (VAE): Optimization



$$\max_{\phi,\theta} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

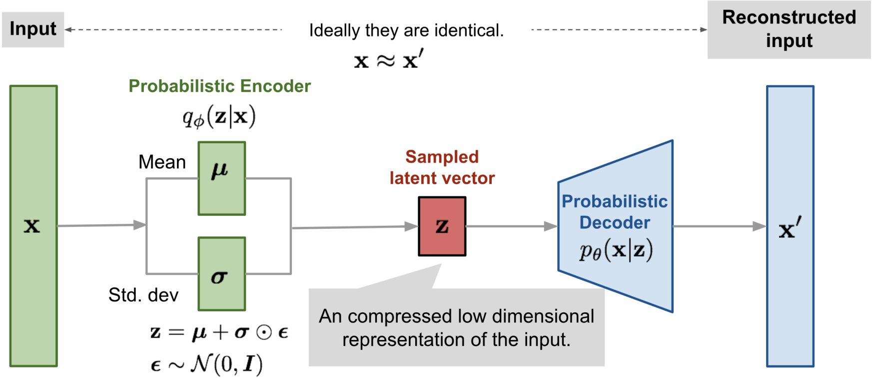
- maximize $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$: reconstruct $\mathbf{x}$
- minimize $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$: approximate prior

Suppose $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. Then let $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu_\phi(\mathbf{x}), \text{diag}(\sigma^2_\phi(\mathbf{x})))$.

The expectation term in the loss function requires sampling $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$, which is a stochastic process. Therefore we cannot backpropagate the gradient.

Image from Kingma and Welling. An Introduction to Variational Autoencoders.2019.

VAE: Reparameterization Trick



$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu_\phi(\mathbf{x}), \text{diag}(\sigma^2_\phi(\mathbf{x})))$$

$$\mathbf{z} = \mu_\phi(\mathbf{x}) + \sigma_\phi(\mathbf{x}) \odot \epsilon, \text{ where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$ *Reparameterization Trick*

Image from https://lilianweng.github.io/posts/2018-08-12-vae/

Variational AutoEncoder (VAE): Details about Loss

- $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$: consider one element of $\mathbf{z}$

$$
\begin{aligned}
& -D_{KL}\left(q_\phi\left(z|x\right)\|p(z)\right) \\
= & \int \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(z-\mu_q)^2}{2\sigma_q^2}\right) \log\left(\frac{\frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(-\frac{\left(z-\mu_p\right)^2}{2\sigma_p^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{\left(z-\mu_q\right)^2}{2\sigma_q^2}\right)}\right) dz \\
= & \log\left(\frac{\sigma_q}{\sigma_p}\right) - \frac{\sigma_q^2 + (\mu_q-\mu_p)^2}{2\sigma_p^2} + \frac{1}{2} \\
= & \frac{1}{2}\left[1 + \log\left(\sigma_q^2\right) - \sigma_q^2 - \mu_q^2\right]
\end{aligned}
$$

Variational AutoEncoder (VAE): Details about Loss

- $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$: consider one element of $\mathbf{z}$

$$
\begin{aligned}
&- D_{KL}\left(q_\phi\left(z|x\right)\|p(z)\right) \\
&= \int \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(z-\mu_q)^2}{2\sigma_q^2}\right) \log\left(\frac{\frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(-\frac{(z-\mu_p)^2}{2\sigma_p^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(z-\mu_q)^2}{2\sigma_q^2}\right)}\right) dz \\
&= \log\left(\frac{\sigma_q}{\sigma_p}\right) - \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{2\sigma_p^2} + \frac{1}{2} \\
&= \frac{1}{2}\left[1 + \log\left(\sigma_q^2\right) - \sigma_q^2 - \mu_q^2\right]
\end{aligned}
$$

- $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$: $p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; g_\theta(\mathbf{z}), \Sigma_{\mathbf{x}})$

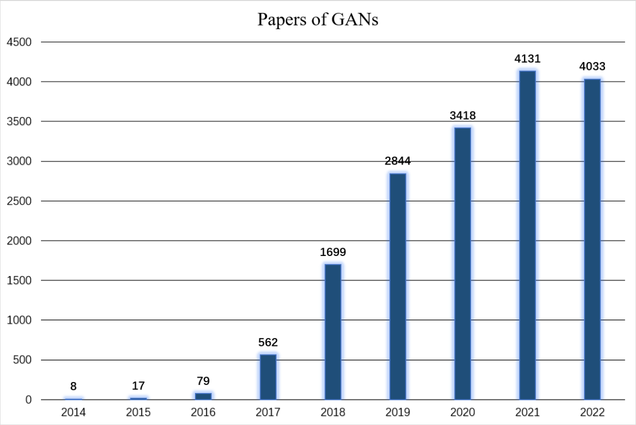
$$
\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \propto \|\mathbf{x} - g_\theta(\mathbf{z})\|^2
$$

References

- [Kingma and Welling, 2013] Auto-Encoding Variational Bayes
- [Higgins et al., 2020] beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework
- [Oord et al., 2017] VQ-VAE: Neural Discrete Representation Learning
- [Razavi et al., 2019] Generating Diverse High-Fidelity Images with VQ-VAE-2
- https://lilianweng.github.io/posts/2018-08-12-vae/
- https://en.wikipedia.org/wiki/Variational_autoencoder
- https://en.wikipedia.org/wiki/Autoencoder

Generative Adversarial Networks (GANs)

- Generative Adversarial Networks is a kind of well-known and popular generative model designed by Ian J. Goodfellow and his colleagues in June 2014.



https://www.aminer.cn/search/pub?q=generative%20adversarial%20networks&t=b

Generative Adversarial Networks

Inspired by game theory, GAN estimates generator via an adversarial process, in which we simultaneously train two neural networks

- A generator G that is trained to capture the real data distribution so that the generated samples can be as real as possible.
- A discriminator D that estimates the probability that a sample came from the training data rather than the generator G.
- Adversarial process: training D to maximize the probability of assigning the correct label to both training examples and samples from G and simultaneously training G to maximize the probability of D making a mistake.

Generative Adversarial Networks

Training steps:

1. Fix parameters of generator G, train discriminator D
2. Fix parameters of discriminator D, train generator G
3. Repeat step 1,2

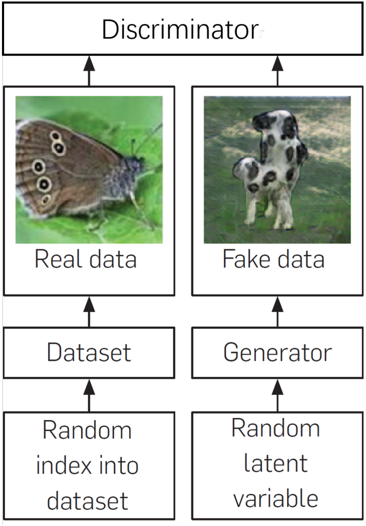


Image from Generative Adversarial Networks (https://dl.acm.org/doi/10.1145/3422622)

Model architecture of GAN

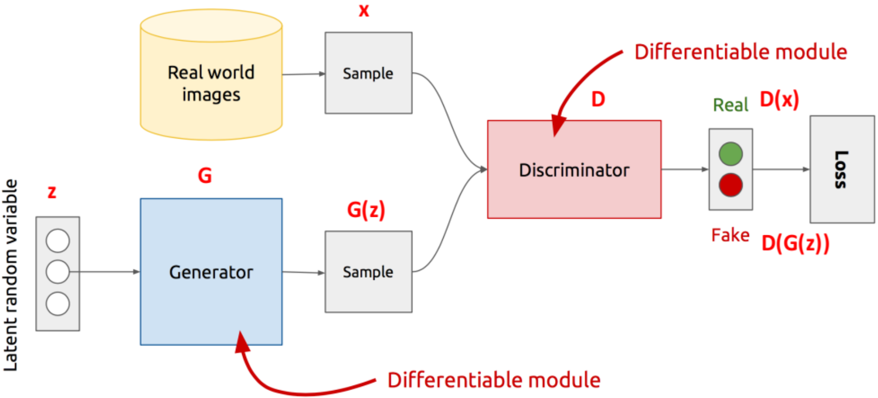


Image from https://www.cs.toronto.edu/ rgrosse/courses/csc321_2018/slides/lec19.pdf

Generative Adversarial Networks

Train the discriminator

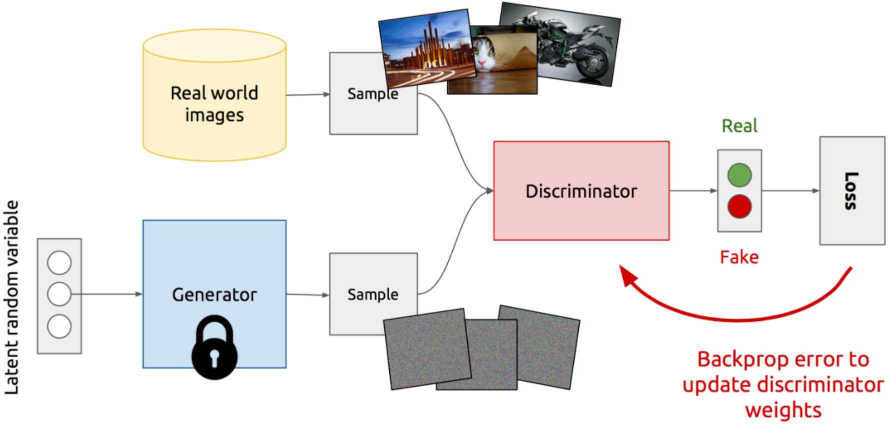


Image from https://www.cs.toronto.edu/ rgrosse/courses/csc321_2018/slides/lec19.pdf

Generative Adversarial Networks

Train the generator

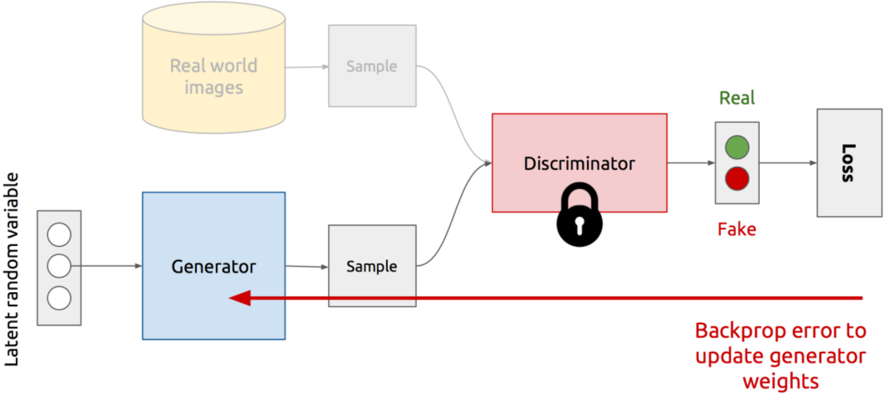


Image from https://www.cs.toronto.edu/ rgrosse/courses/csc321_2018/slides/lec19.pdf

Generative Adversarial Networks

- Notations
  - p_r: data distribution over real samples $\mathbf{x}$
  - p_g: the generator's distribution over data $\mathbf{x}$
  - p_z: a prior on input noise variable $\mathbf{z}$

Generative Adversarial Networks

- Notations
  - p_r: data distribution over real samples $\mathbf{x}$
  - p_g: the generator's distribution over data $\mathbf{x}$
  - p_z: a prior on input noise variable $\mathbf{z}$
- Ensure the discriminator $D's$ decisions over real data are accurate by

$$\text{maximize}_D \; \mathbb{E}_{\mathbf{x} \sim p_r(\mathbf{x})}[\log D(\mathbf{x})]$$

Generative Adversarial Networks

- Notations
  - p_r: data distribution over real samples $\mathbf{x}$
  - p_g: the generator's distribution over data $\mathbf{x}$
  - p_z: a prior on input noise variable $\mathbf{z}$
- Ensure the discriminator $D's$ decisions over real data are accurate by

$$\text{maximize}_D \ \mathbb{E}_{\mathbf{x}\sim p_r(\mathbf{x})}[\log D(\mathbf{x})]$$

- Given a fake sample $G(\mathbf{z}), \mathbf{z} \sim p_z(\mathbf{z})$, the discriminator is expected to output a probability, $D(G(\mathbf{z}))$, close to zero by

$$\text{maximize}_D \ \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

- On the other hand, the generator is trained to increase the chances of D producing a high probability for generated samples, thus

$$\text{minimize}_G \ \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

- Therefore, D and G play the following two-player minimax game with loss function $\mathcal{L}(G, D)$:

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{\mathbf{x} \sim p_r(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Pseudo Code of GAN

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

Discriminator updates

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

Generator updates

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Image from https://www.cs.toronto.edu/ rgrosse/courses/csc321_2018/slides/lec19.pdf

Advantages and Disadvantages of GAN

- Advantages
  - Sampling (or generation) is intuitive and straightforward.
  - Compared to VAE, the training of GAN doesn't involve MLE.
  - Compared to VAE, the generated samples of GAN are more realistic.

Advantages and Disadvantages of GAN

- Advantages
  - Sampling (or generation) is intuitive and straightforward.
  - Compared to VAE, the training of GAN doesn't involve MLE.
  - Compared to VAE, the generated samples of GAN are more realistic.
- Disadvantages
  - Probability distribution is implicit
    - Not straightforward to compute $p(\mathbf{x})$
    - Thus only good for generating new samples
  - The training is hard
    - No convergence guarantee
    - May encounter mode collapse

A brief history of GANs

- [Goodfellow et al., 2014]: Generative Adversarial Networks (GAN)
- [Mirza et al. 2014]: Conditional GAN
- [Radford et al. 2015]: Deep Convolutional GAN
- [Ming-Yu Liu et al., 2016]: Coupled GAN
- [Karras et al. 2017]: Progressive Growing of GANs
- [Arjovsky et al. 2017]: Wasserstein GAN
- [Zhu et al. 2017]: CycleGAN
- [Han Zhang et al. 2018]: Self-Attention GAN
- [Brock et al. 2018]: Large-scale GAN training (BigGAN)
- [Karras et al. 2018]: A style-based generator architecture for GAN (StyleGAN)

Progress of GANs on image generation

- human face

2014 (GAN) 2015 (DCGAN) 2016 (CoGAN) 2017 (ProGAN) 2018 (StyleGAN)

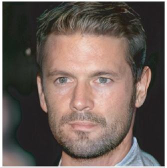
Progress of GANs on image generation

- human face


2014 (GAN)

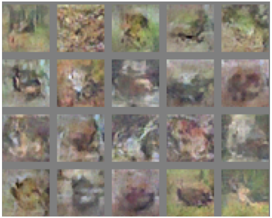

2015 (DCGAN)


2016 (CoGAN)


2017 (ProGAN)


2018 (StyleGAN)

- other objects


2014 (GAN)


2015 (DCGAN)


2018 (BigGAN)

Diffusion Models: Overview

- Diffusion models, also known as diffusion probabilistic models, are a class of latent variable models introduced in 2015 with inspiration from non-equilibrium thermodynamics.
- Overview of different types of generative models

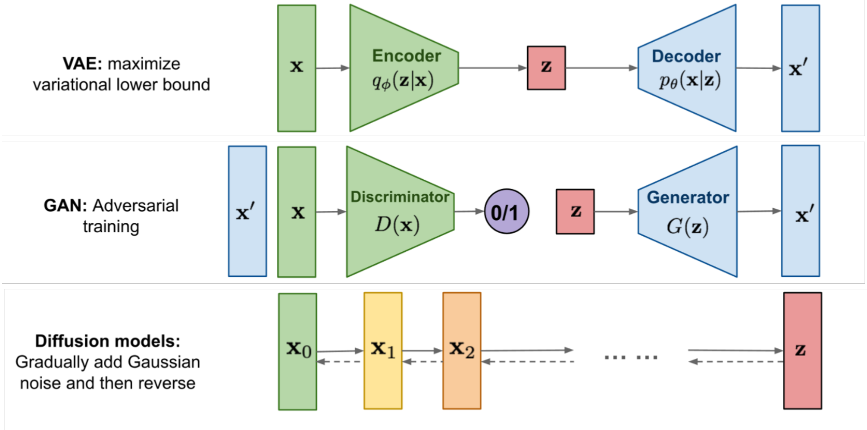


Image from https://lilianweng.github.io/posts/2021-07-11-diffusion-models

Diffusion Models: Forward Diffusion Process

- Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, a forward diffusion process adds small noise (e.g. Gaussian noise) to the sample in T steps slowly, which produces a sequence of noisy samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x_T}$:

$$\mathbf{x}_t = \sqrt{1 - \beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1}$$

  The step sizes are controlled by a variance schedule $\{\beta_t \in (0, 1)\}_{t=1}^{T}$.
- When $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Diffusion Models: Forward Diffusion Process

- Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, a forward diffusion process adds small noise (e.g. Gaussian noise) to the sample in T steps slowly, which produces a sequence of noisy samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x_T}$:

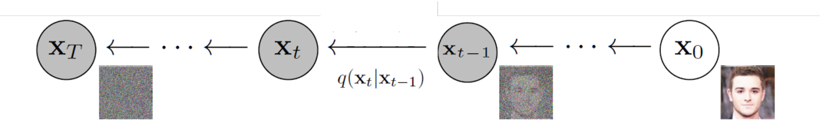
$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1}$$

  The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^T$.

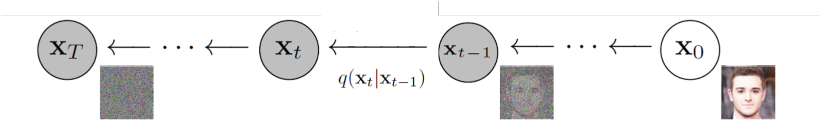
- When $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we have

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

- The sample $\mathbf{x}_0$ gradually loses its distinguishable features as t becomes larger. Eventually when $T \to \infty$, $\mathbf{x}_T$ becomes isotropic Gaussian.



Diffusion Models: Forward Diffusion Process



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

We can sample $\mathbf{x}_t$ at any arbitrary time step t in a closed form. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, then

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t \alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}
\end{aligned}$$

* $\bar{\boldsymbol{\epsilon}}_{t-2}$ merged $\boldsymbol{\epsilon}_{t-1}$ and $\boldsymbol{\epsilon}_{t-2}$. $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It follows that

$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$.

Diffusion Models: Reverse Diffusion Process

- If the diffusion process can be reversed, using $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we can create a true sample from a Gaussian noise input $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
- If β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian.

Diffusion Models: Reverse Diffusion Process

- If the diffusion process can be reversed, using $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we can create a true sample from a Gaussian noise input $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
- If β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian.
- We learn a model p_θ to conduct the reverse diffusion process:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ are the outputs of a neural network parameterized by θ. The inputs are $\mathbf{x}_t$ and t.

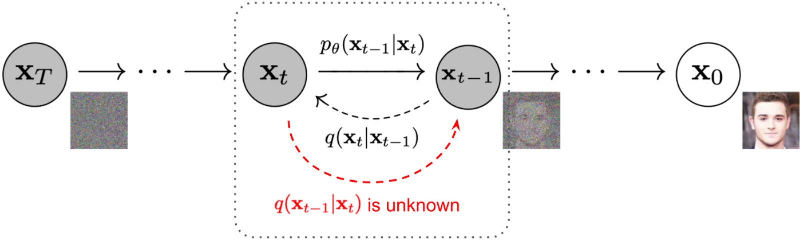
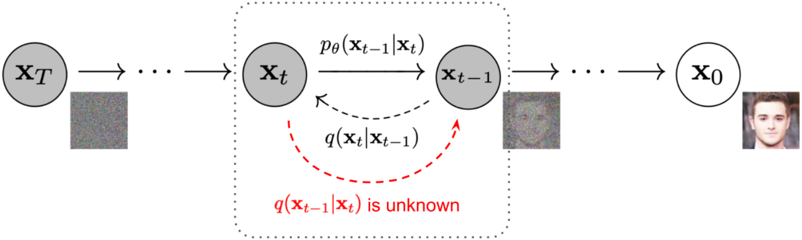


Image from the https://lilianweng.github.io/posts/2021-07-11-diffusion-models

Diffusion Models: Reverse Diffusion Process



The reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t, \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0$$

The derivation is a little complex and hence omitted. See [Sohl-Dickstein et al. 2015].

Diffusion Models: Training (optional)

Minimize the variational bound on negative log-likelihood:

$$\begin{aligned}
\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq & \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] \\
= & \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right)-\sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right] \\
= & \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T}\right. \\
& \left.+\sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}\right]
\end{aligned}$$

Diffusion Models: Training (optional)

Minimize the variational bound on negative log-likelihood:

$$
\begin{aligned}
\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq & \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] \\
= & \mathbb{E}_q\left[-\log p\left(\mathbf{x}_T\right)-\sum_{t \geq 1} \log \frac{p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right)}\right] \\
= & \mathbb{E}_q\left[\underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_T \mid \mathbf{x}_0\right) \| p\left(\mathbf{x}_T\right)\right)}_{L_T}\right. \\
& \left.+\sum_{t>1} \underbrace{D_{\mathrm{KL}}\left(q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)}_{L_0}\right]
\end{aligned}
$$

$$
\begin{aligned}
L_{t-1} = & \mathbb{E}_q\left[\frac{1}{2 \sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right)-\boldsymbol{\mu}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right]+C \\
= & \mathbb{E}_{\mathbf{x}_0, \epsilon}\left[\frac{1}{2 \sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right)-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}\right)-\boldsymbol{\mu}_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0, \boldsymbol{\epsilon}\right), t\right)\right\|^2\right]+C
\end{aligned}
$$

For derivations, refer to [Sohl-Dickstein et al. 2015] and [Ho et al. 2020].

Diffusion Models: Training (optional)

Reparameterization

$$
\begin{aligned}
L_{t-1} =& \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t\left(\mathbf{x}_0,\epsilon\right)-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)-\boldsymbol{\mu}_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0,\epsilon\right),t\right)\right\|^2\right]+C \\
=& \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t\left(1-\bar{\alpha}_t\right)}\left\|\epsilon-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\epsilon,t\right)\right\|^2\right]+C
\end{aligned}
$$

Diffusion Models: Training (optional)

Reparameterization

$$
\begin{aligned}
L_{t-1} =& \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t\left(\mathbf{x}_0,\epsilon\right)-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)-\boldsymbol{\mu}_\theta\left(\mathbf{x}_t\left(\mathbf{x}_0,\epsilon\right),t\right)\right\|^2\right]+C \\
=& \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t\left(1-\bar{\alpha}_t\right)}\left\|\epsilon-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\epsilon,t\right)\right\|^2\right]+C
\end{aligned}
$$

A simplified objective [Ho et al. 2020] that ignores the weighting term and the final optimization objective is:

$$
L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\epsilon}\left[\left\|\epsilon-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\epsilon,t\right)\right\|^2\right] \tag{1}
$$

Optimization: SGD

Diffusion Models: Training and Sampling

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 $\mathbf{x}_0$

Image from Ho et al. 2020.

Diffusion Models: Examples

The forward trajectory $q(\mathbf{x}_{0:T})$

The reverse trajectory $p_\theta(\mathbf{x}_{0:T})$

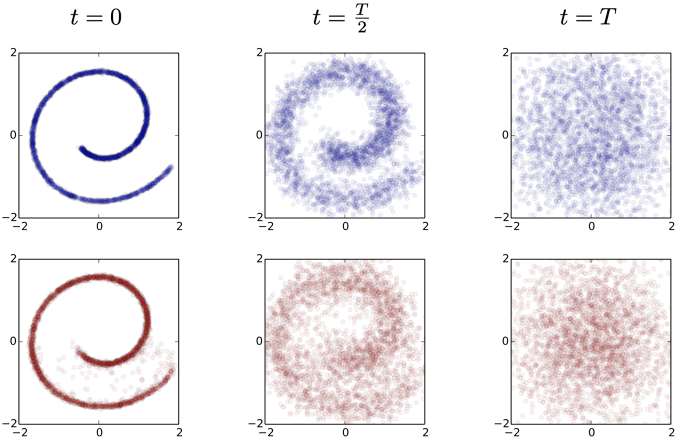


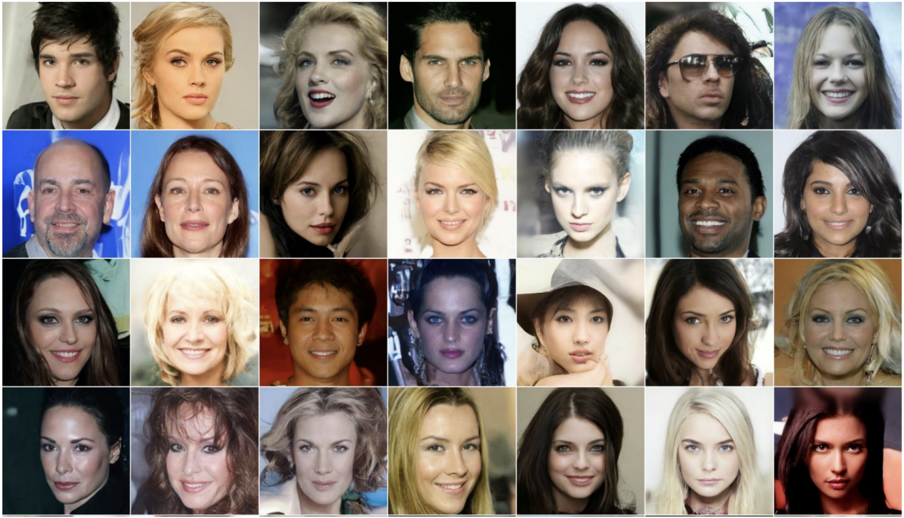
Image from Sohl-Dickstein et al. 2015.

Diffusion Models: Examples

CIFAR10 progressive generation [Ho et al. 2020]

Diffusion Models: Examples

CelebA-HQ 256 $\times$ 256 generated samples [Ho et al. 2020]



Diffusion Models: Advantages and Disadvantages

- Advantages
  - The quality of generated samples are often higher than VAE and GAN.
  - Probability distribution is explicit.

Diffusion Models: Advantages and Disadvantages

- Advantages
  - The quality of generated samples are often higher than VAE and GAN.
  - Probability distribution is explicit.
- Disadvantages
  - The training process is time-consuming.
  - It is very slow to generate a sample from DDPM since T is often very large, i.e. 1000.

Diffusion Models

References

- [Sohi-Dickstein et al. 2015] Deep Unsupervised Learning using Nonequilibrium Thermodynamics
- [Ho et al. 2020] Denoising Diffusion Probabilistic Nodels
- [Nichol et al. 2021] Improved Denoising Diffusion Probabilistic Models
- [Song et al. 2020] Denoising Diffusion Implicit Models
- https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Learning Outcomes

- Understand the main ideas of VAE, GAN, and diffusion model
- Understand the derivation of the objective function of VAE
- Know the advantages and disadvantages of VAE, GAN, and diffusion model
- Be able to use at least one of VAE, GAN, and diffusion model to generate realistic data samples.