Generative Models

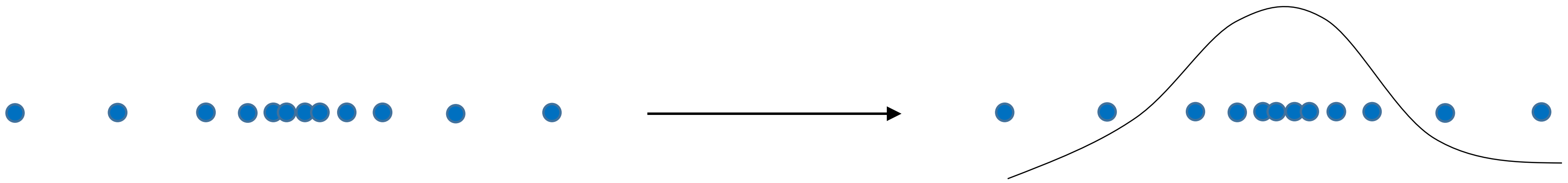
Tianwen Jiang

Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology

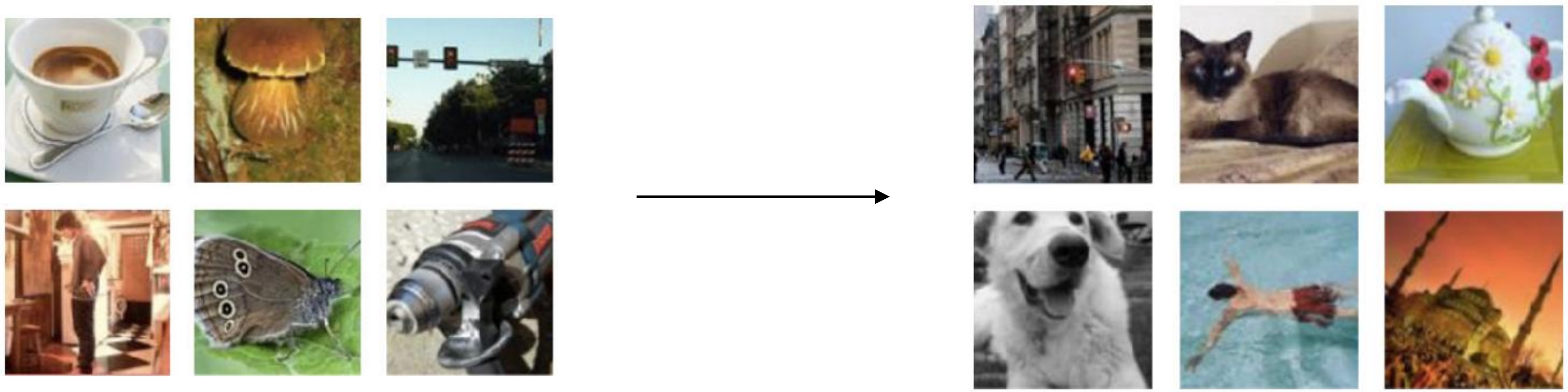
twjiang@ir.hit.edu.cn

Generative Modeling

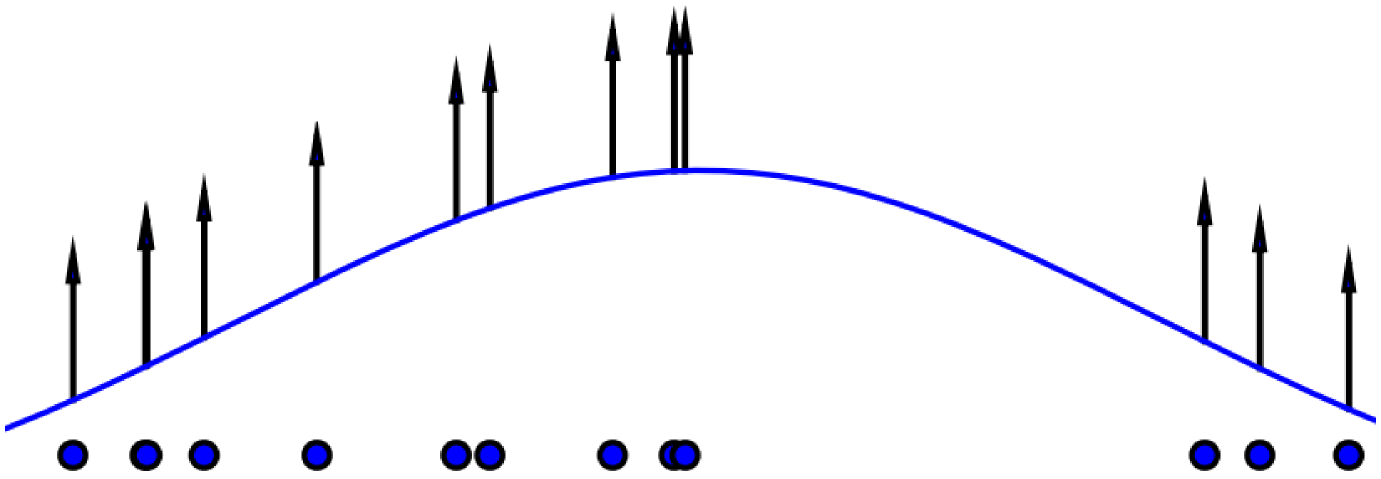
- Density estimation



- Sample generation

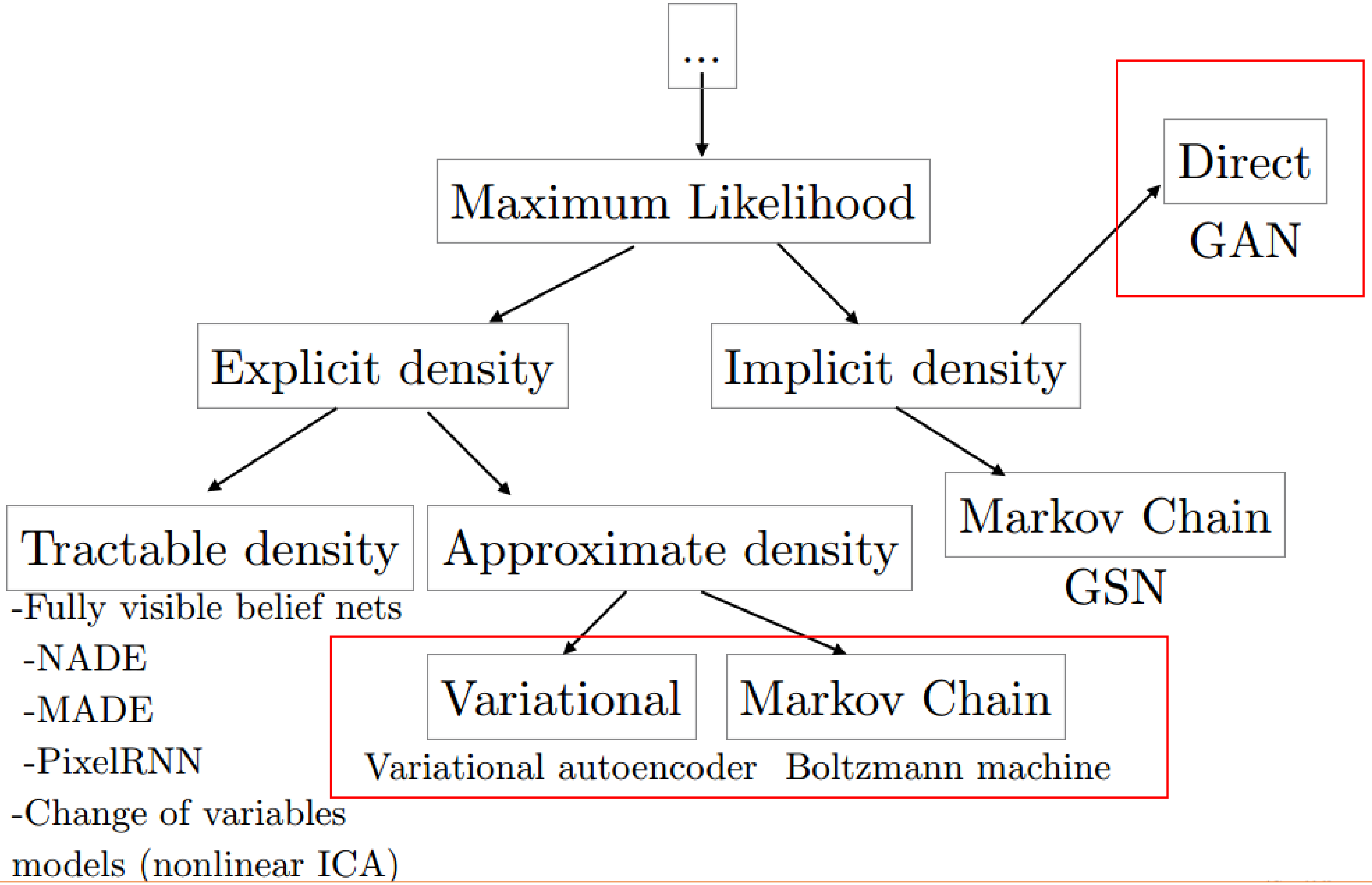


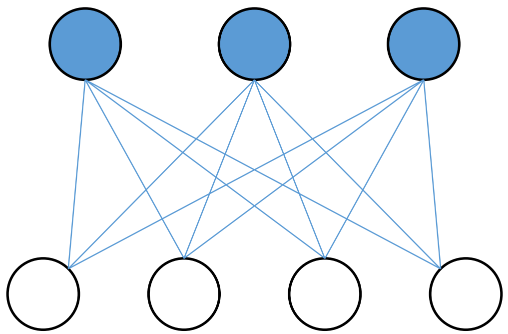
Maximum Likelihood



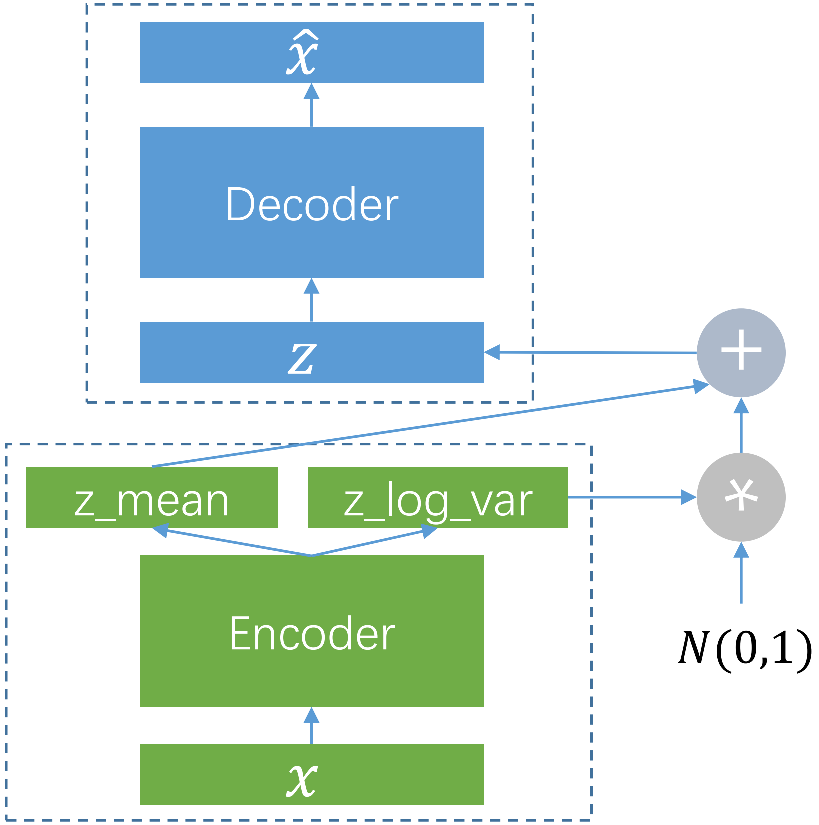
$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{data}} \log p(\boldsymbol{x}; \boldsymbol{\theta})$$

Taxonomy of Generative Models

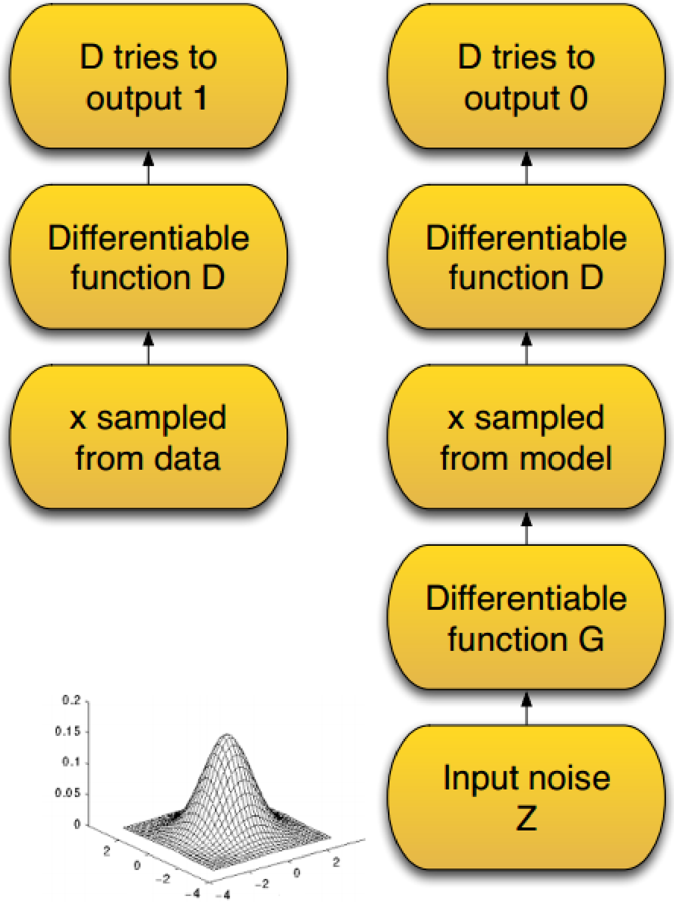




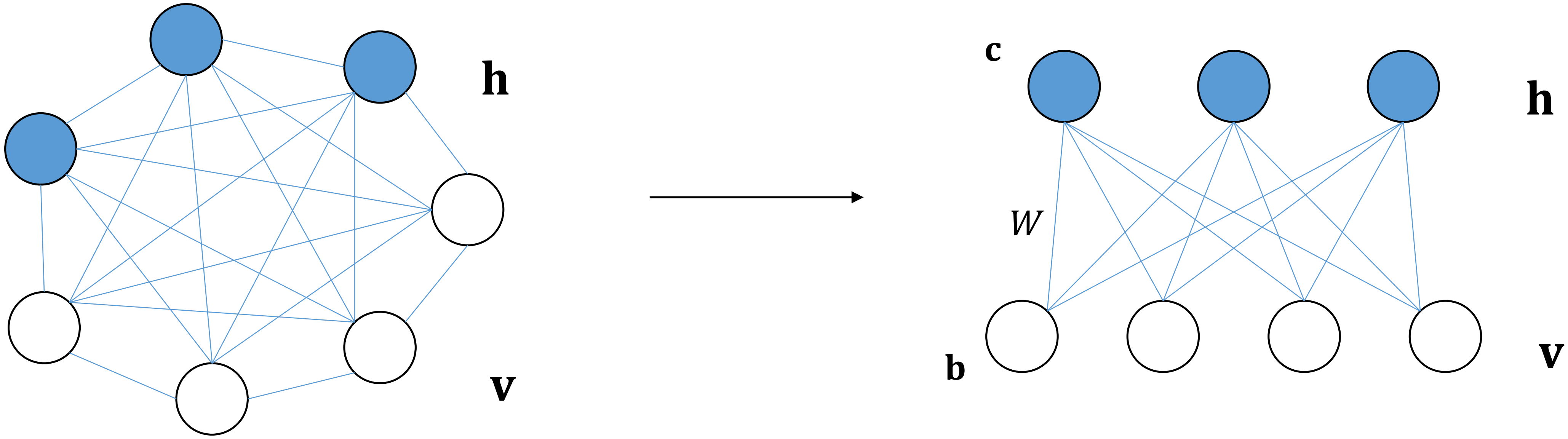
Restricted Boltzmann Machines



Variational Autoencoder



Boltzmann Machines



Boltzmann Machines

Restricted Boltzmann Machines

$$\theta = \{\mathbf{b}, \mathbf{c}, W\}$$

Restricted Boltzmann Machines

Energy Based Model (EBM)

Energy function: $E_\theta(\mathbf{v},\mathbf{h}) = -\mathbf{b}^{\mathrm{T}}\mathbf{v} - \mathbf{c}^{\mathrm{T}}\mathbf{h} - \mathbf{v}^{\mathrm{T}}W\mathbf{h}$

Probability: $P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{Z_\theta} e^{-E_\theta(\mathbf{v},\mathbf{h})}$

Partition function: $Z_\theta = \sum_{\mathbf{v},\mathbf{h}} e^{-E_\theta(\mathbf{v},\mathbf{h})}$

Restricted Boltzmann Machines

$$\ell(\theta) = \prod_{t=1}^{n} P(\mathbf{v}^t) \Rightarrow \ell(\theta) = \log \prod_{t=1}^{n} P(\mathbf{v}^t)$$

$$\Rightarrow \ell(\theta) = \sum_{t=1}^{n} \log P(\mathbf{v}^t) = \sum_{t=1}^{n} \log \sum_{\mathbf{h}} P(\mathbf{v}^t, \mathbf{h})$$

$$\ell(\theta) = \sum_{t=1}^{n} \log \sum_{\mathbf{h}} e^{-E_\theta(\mathbf{v}^t, \mathbf{h})} - n \log \sum_{\mathbf{v},\mathbf{h}} e^{-E_\theta(\mathbf{v}, \mathbf{h})}$$

Restricted Boltzmann Machines

$$\ell(\theta) = \sum_{t=1}^{n} \log \sum_{\mathbf{h}} \boldsymbol{e}^{-E_\theta(\mathbf{v}^t,\mathbf{h})} - n\log \sum_{\mathbf{v},\mathbf{h}} e^{-E_\theta(\mathbf{v},\mathbf{h})}$$

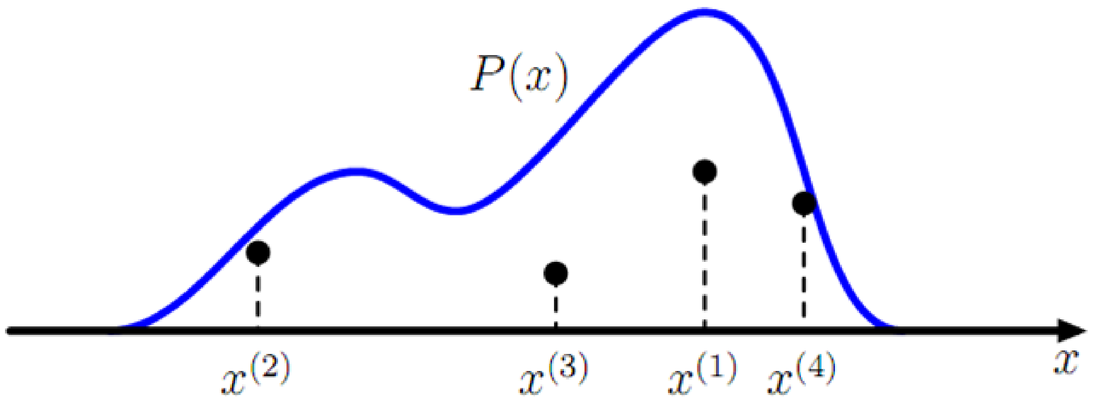
Restricted Boltzmann Machines

$$\nabla_\theta \ell(\theta) = \sum_{t=1}^{n} \nabla_\theta \log \sum_{\mathbf{h}} \boldsymbol{e}^{-E_\theta(\mathbf{v}^t,\mathbf{h})} - n\nabla_\theta \log \sum_{\mathbf{v},\mathbf{h}} e^{-E_\theta(\mathbf{v},\mathbf{h})}$$

$$= \sum_{t=1}^{n} \mathbb{E}_{P(\mathbf{h}|\mathbf{v}^t)}[\nabla_\theta - E_\theta(\mathbf{v}^t, \mathbf{h})] - n\mathbb{E}_{P(\mathbf{h},\mathrm{v})}[\nabla_\theta - E_\theta(\mathbf{v}, \mathbf{h})]$$

it is impractical to compute the exact log-likelihood gradient

Markov Chain Monte Carlo



$Z \sim N(0,1)$

Box–Muller transform

$U_1, U_2, \ldots U_n \sim Uniform(0,1)$

linear congruential generator

Gibbs Sampling

Metropolis-Hastings Sampling

detailed balance condition

Markov Chains - Stationary Distributions

Restricted Boltzmann Machines

$$\nabla_\theta \ell(\theta) = \sum_{t=1}^{n} \mathbb{E}_{P(\mathbf{h}|\mathbf{v}^t)}[\nabla_\theta - E_\theta(\mathbf{v}^t, \mathbf{h})] \; - n\mathbb{E}_{P(\mathbf{h},\mathbf{v})}[\nabla_\theta - E_\theta(\mathbf{v}, \mathbf{h})]$$

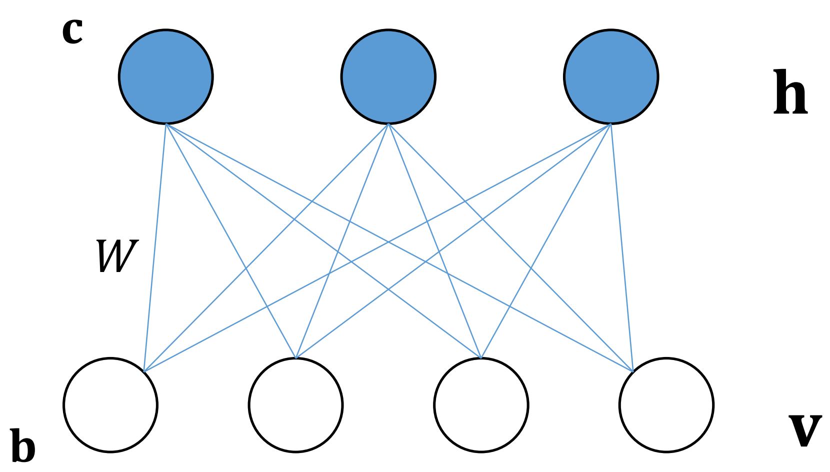
Gibbs Sampling

Contrastive Divergence

$$\mathbf{h}^t \sim P(\mathbf{h}|\mathbf{v}^t)$$
$$\mathbf{v}^{t+1} \sim P(\mathbf{v}|\mathbf{h}^t)$$
$$\ldots$$

$$\mathbb{E}_{P(\mathbf{h},\mathbf{v})}[\nabla_\theta - E_\theta(\mathbf{v}, \mathbf{h})]$$
$$\approx \nabla_\theta - E_\theta(\mathbf{v}, \mathbf{h})|_{\mathbf{v}=\tilde{\mathbf{v}}\ ,\mathbf{h}=\tilde{\mathbf{h}}}$$

Boltzmann Machines



Restricted Boltzmann Machines

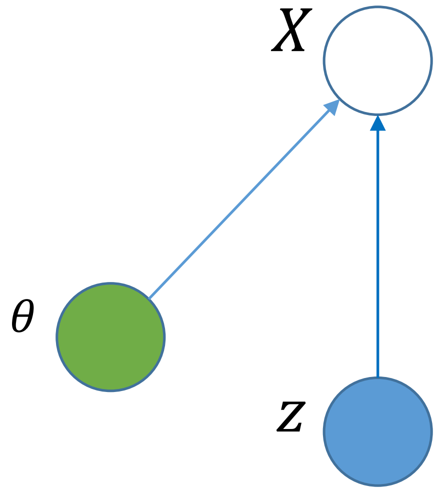
✓ *Dimensionality reduction*

✓ *Classification*

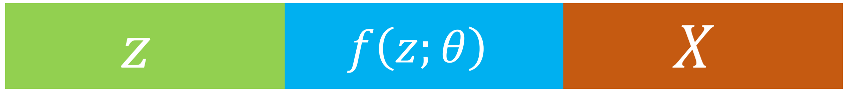
✓ *Feature learning*

✓ *Deep learning*

Variational Autoencoder



$$P(X) = \int P(X|z;\theta)P(z)dz$$



$$P(X|z;\ \theta) = \mathcal{N}(X|\, f(z;\theta), \sigma^2 * I)$$

$$f(z;\theta)\colon \mathcal{Z} \times \Theta \rightarrow \mathcal{X}$$

Variational Autoencoder

Where is z from? $z \sim \mathcal{N}(0, I)$, $Q(z|X)$

Goal: $Q(z|X) \approx P(z|X)$

Kullback-Leibler divergence: $\mathcal{D}[Q(z|X)||P(z|X)]$

$$\mathcal{D}[Q(z|X)||P(z|X)] = \mathbb{E}_{z\sim Q}\left[\log Q(z|X) - \log P(z|X)\right]$$

$$\Downarrow$$

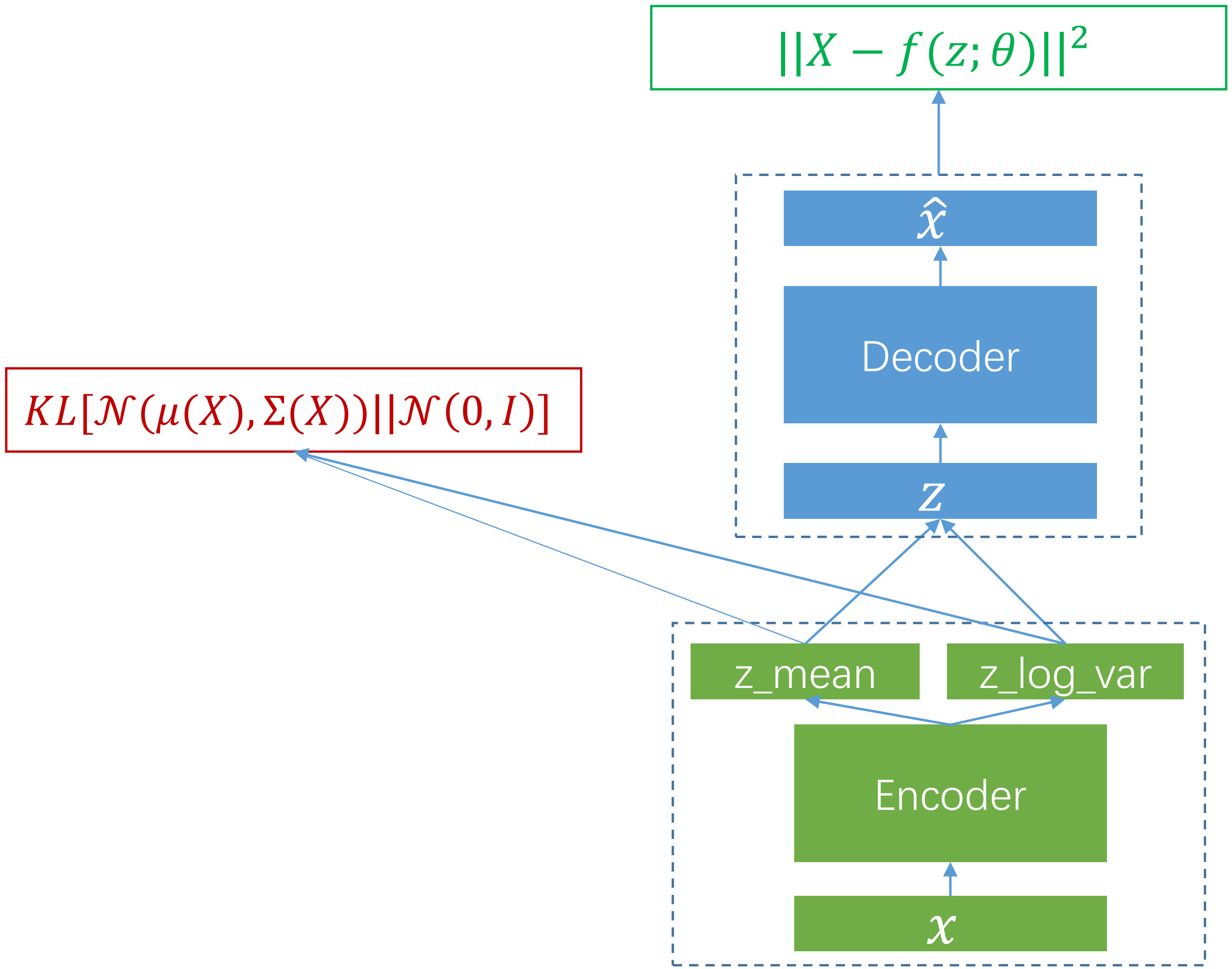
$$\log P(X) - \mathcal{D}[Q(z|X)||P(z|X)] = \mathbb{E}_{z\sim Q}\left[-\log Q(z|X) + \log P(z) + \log P(X|z)\right]$$

Variational Autoencoder

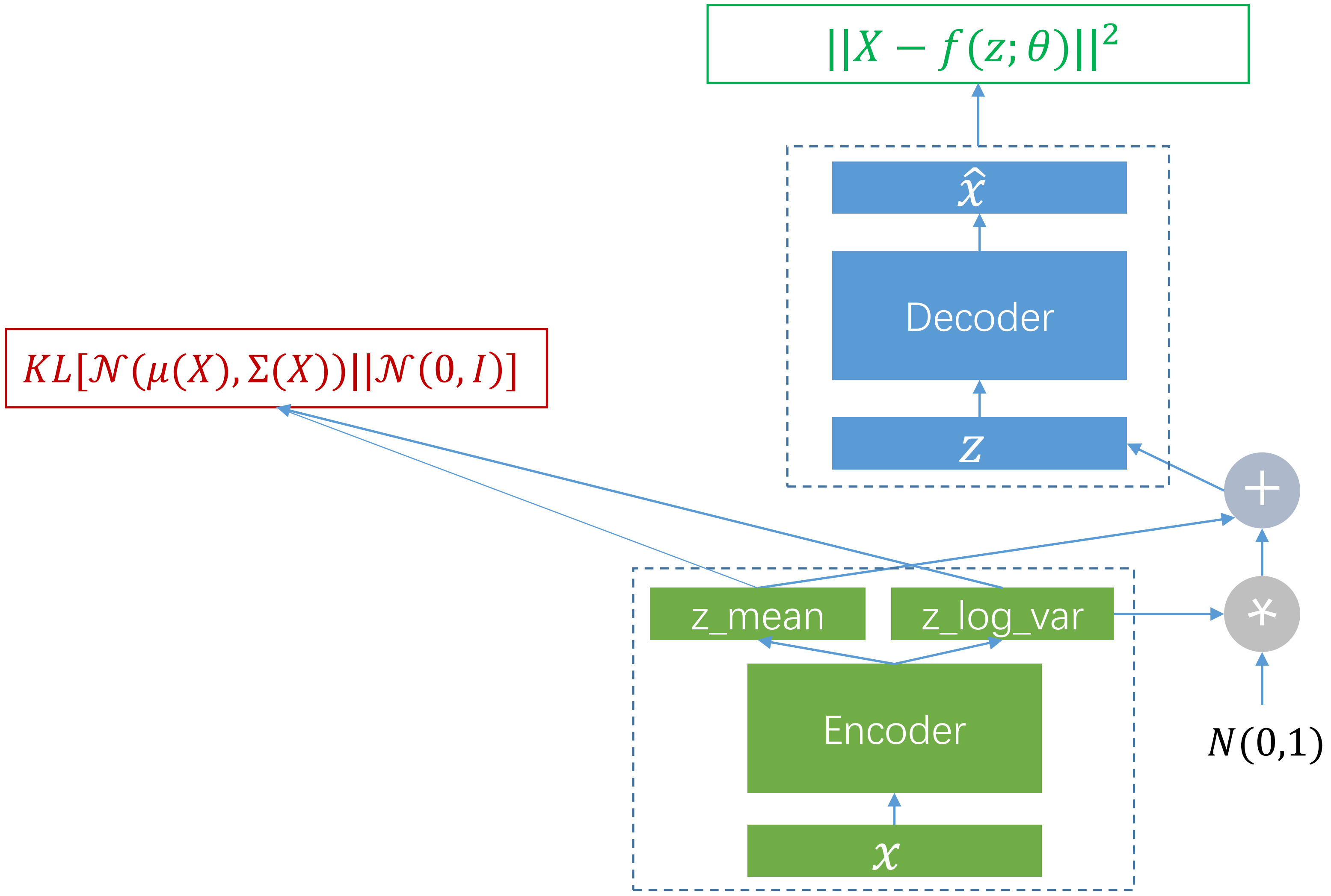
$$\log P(X) - \mathcal{D}[Q(z|X)||P(z|X)] = \mathbb{E}_{z\sim Q}\,[\log P(X|z)] - \mathcal{D}[Q(z|X)||P(z)]$$

$$Q(z|X) \sim \mathcal{N}(z|\mu(X;\vartheta), \Sigma(X;\vartheta))$$

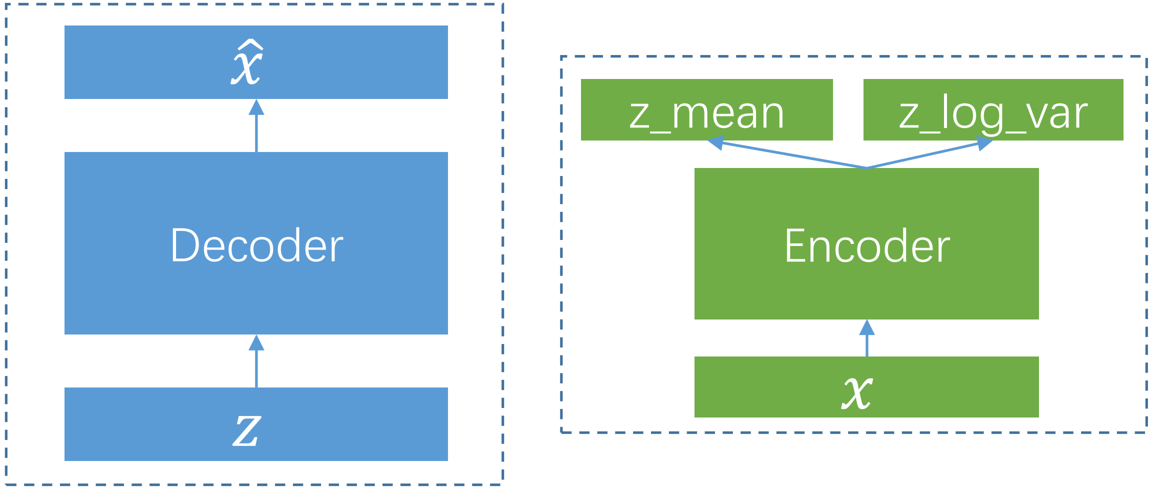
$$\log P(X) - \mathcal{D}[Q(z|X)||P(z|X)] = \mathbb{E}_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}[Q(z|X)||P(z)]$$



$$\log P(X) - \mathcal{D}[Q(z|X)||P(z|X)] = \mathbb{E}_{z\sim Q}[\log P(X|z)] - \mathcal{D}[Q(z|X)||P(z)]$$



Variational Autoencoder

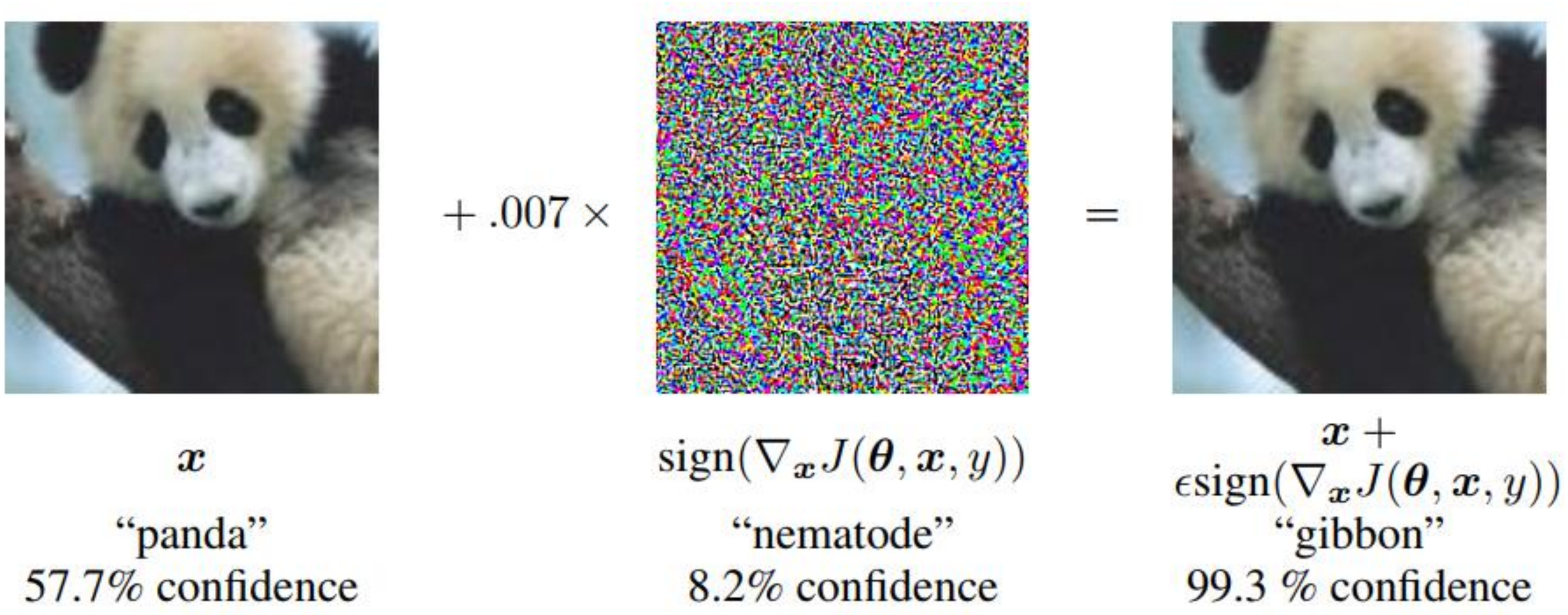


Variational Autoencoder

- ✓ *Data generation*
- ✓ *Dimensionality reduction*
- ✓ *Hole filling*
- ✓ *Semi-supervised learning*

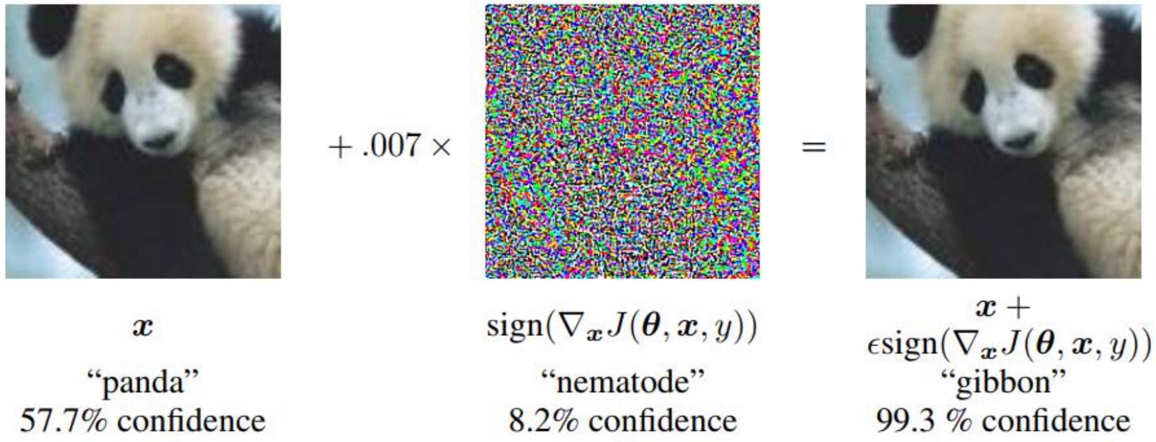
Generative Adversarial Nets

adversarial examples



Generative Adversarial Nets

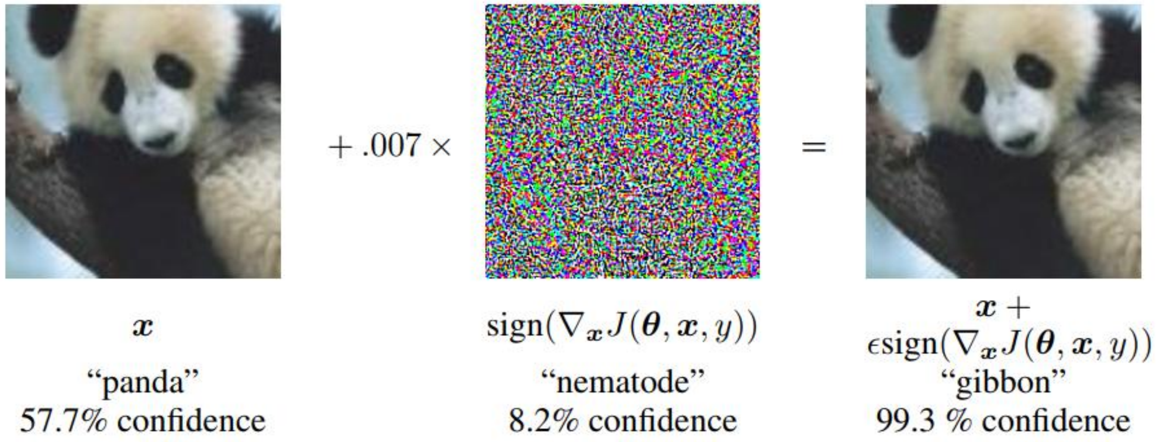
adversarial examples



- **Nonlinearity?**
- **Overfitting?**

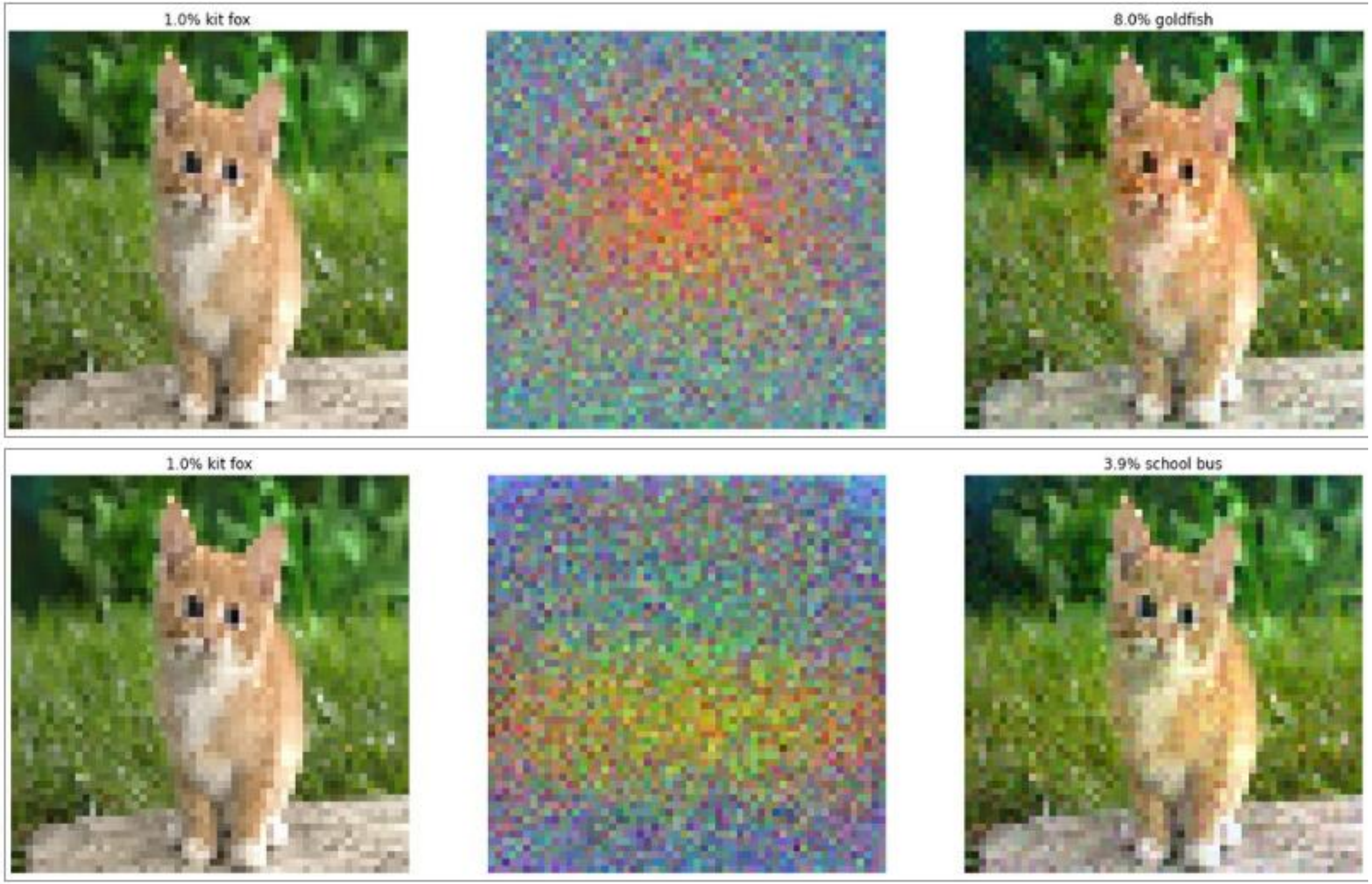
Generative Adversarial Nets

adversarial examples



$$\boldsymbol{w}^{\mathrm{T}}\tilde{\boldsymbol{x}} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\eta}$$

$$\boldsymbol{\eta} = \epsilon \mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$


1.0% kit fox
8.0% goldfish
1.0% kit fox
3.9% school bus

Generative Adversarial Nets

adversarial examples (new objective function)

$$\tilde{\boldsymbol{J}}(\boldsymbol{\theta},\boldsymbol{x},y)=\alpha J(\boldsymbol{\theta},\boldsymbol{x},y)+(1-\alpha)J(\boldsymbol{\theta},\boldsymbol{x}+\epsilon\,\mathrm{sign}(\nabla_{\boldsymbol{x}}J(\boldsymbol{\theta},\boldsymbol{x},y))$$

Generative Adversarial Nets

adversarial examples

modern machine learning techniques

Potemkin village?

Generative Adversarial Nets

Zero-Sum Game

Generative Adversarial Nets

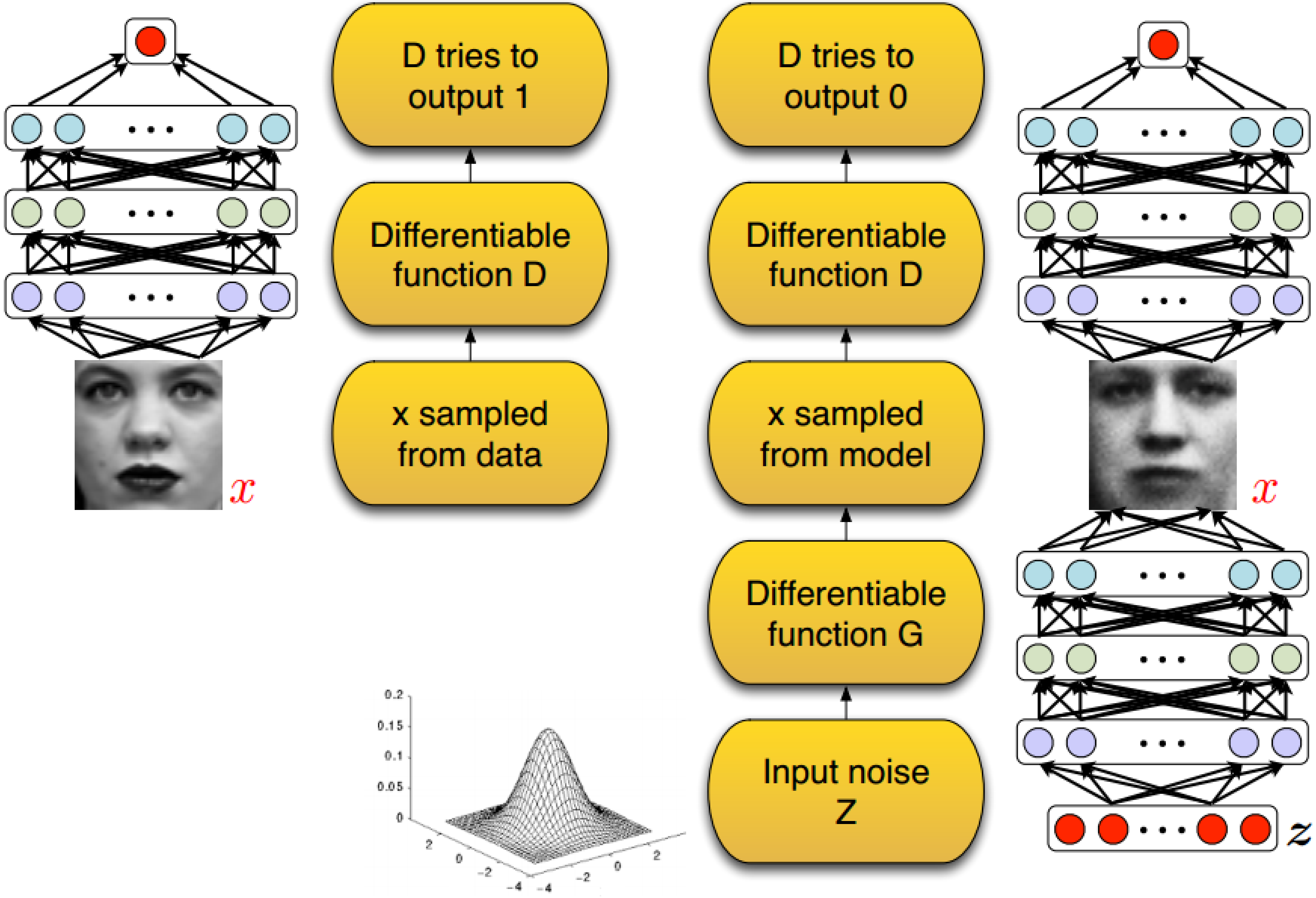
A game between two players:

1. Discriminator D
2. Generator G

D tries to discriminate between:

- A sample from the data distribution.
- And a sample from the generator G.

G tries to "trick" D by generating samples that are hard for D to distinguish from data.


D tries to output 1
Differentiable function D
x sampled from data
D tries to output 0
Differentiable function D
x sampled from model
Differentiable function G
Input noise Z
x
x
z

Generative Adversarial Nets

target function

$$\min_G \max_D V(G, D) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]$$

In practice: $\min_G \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{x}))]$

Generative Adversarial Nets

proof

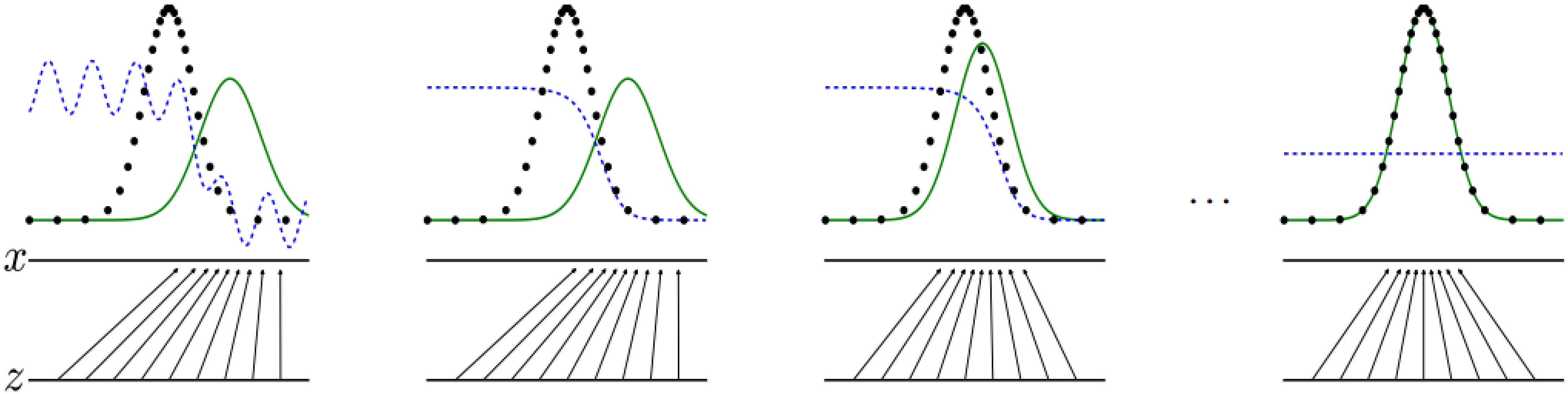
For G fixed, the optimal discriminator D is

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

$$V(G,D) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]$$

$$V(G,D) = \int_x \; p_{data}(\boldsymbol{x}) \log D(\boldsymbol{x}) + p_{\boldsymbol{g}}(\boldsymbol{x}) \log(1 - D(x)) \, dx$$

Generative Adversarial Nets



(a) (b) (c) (d)

$$\min_G \max_D V(G,D) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))]$$

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_{g}(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))] \\ &= \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_{g}(\boldsymbol{x})}\left[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}\right] \end{aligned}$$

$$C(G) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}\left[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{g}}(\boldsymbol{x})}\left[\log \frac{p_g(x)}{p_{data}(x) + p_g(x)}\right]$$

Generative Adversarial Nets

$proof$

$C(G)$ get its opt. at $p_{data}(x)$=$p_g(x)$, and the value is $-\mathbf{log4}$

$$C(G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{g}}(\boldsymbol{x})}\left[\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right]$$

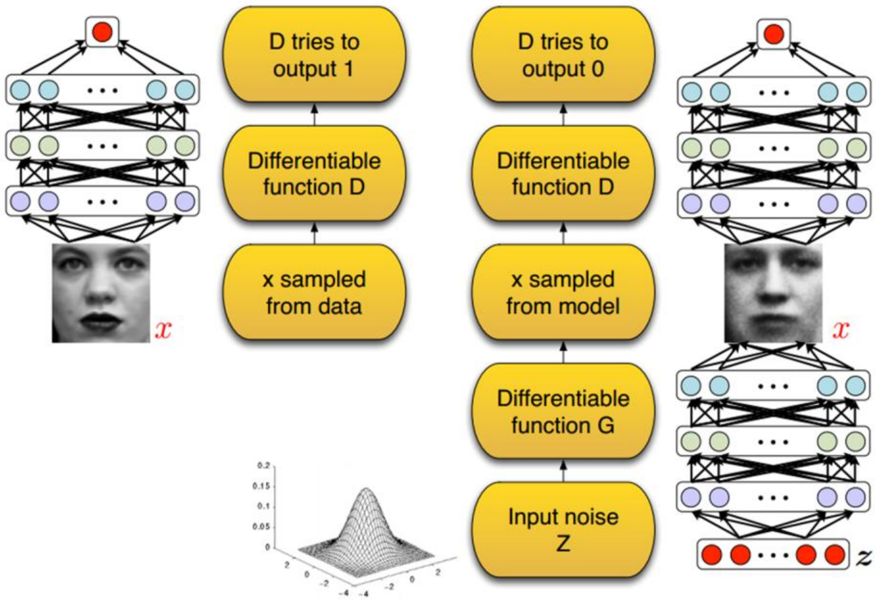
$$C(G) = -\log 4 + KL[p_{data}(x)||\frac{p_{data}(x)+p_g(x)}{2}] + KL[p_g(x)||\frac{p_{data}(x)+p_g(x)}{2}]$$

$$C(G) = -\log 4 + 2\cdot JSD[p_{data}(x)||p_g(x)]$$

Generative Adversarial Nets

$If\ G\ and\ D\ have\ enough\ capacity, p_g\ converges\ to\ p_{data}$

Generative Adversarial Nets



Generative Adversarial Nets

- ✓ *Image generation*
- ✓ *Image Synthesis*
- ✓ *Super Resolution*

Comparison (RBM, VAE, GAN)

- RBM
  - Have intractable likelihood functions and therefore require numerous approximations to the likelihood gradient;
  - Markov chain
- VAE
  - Not asymptotically consistent
- GAN
  - Asymptotically consistent
  - No Markov chains needed

Reference

Fischer A, Igel C. An introduction to restricted Boltzmann machines[J]. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 2012: 14-36.

Hinton G E. Training products of experts by minimizing contrastive divergence[J]. Neural computation, 2002, 14(8): 1771-1800.

Hinton G. A practical guide to training restricted Boltzmann machines[J]. Momentum, 2010, 9(1): 926.

Doersch C. Tutorial on variational autoencoders[J]. arXiv preprint arXiv:1606.05908, 2016.

LeCun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.

Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[J]. arXiv preprint arXiv:1412.6572, 2014.

Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.