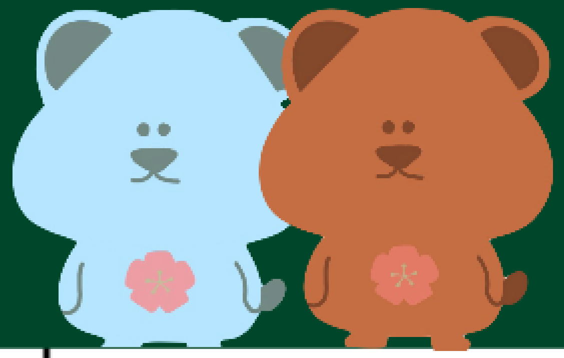


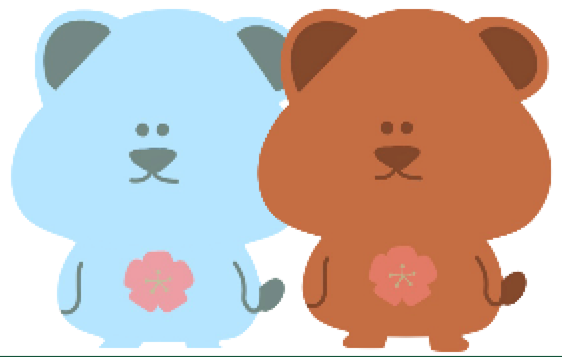
Generative Models

Week 15 구미진, 최지우

EWHA EURON

Index

overview



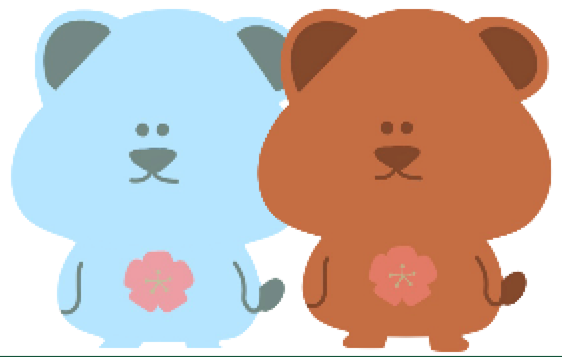
EWHA EURON

Overview

Unsupervised Learning

Generative Models
- PixelRNN and PixelCNN
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)

Supervised vs Unsupervised Learning



EWHA EURON

#01 Supervised vs Unsupervised Learning

Supervised Learning

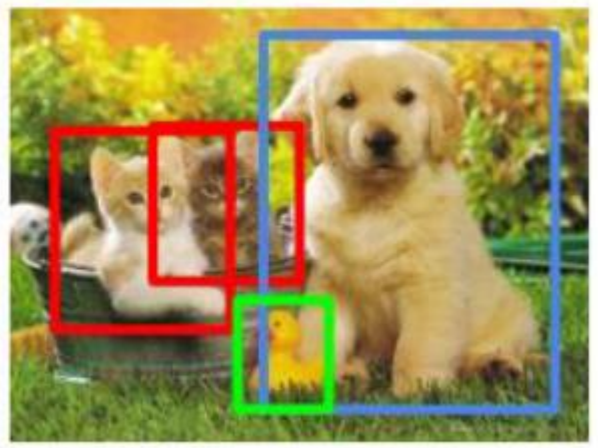
Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Classification

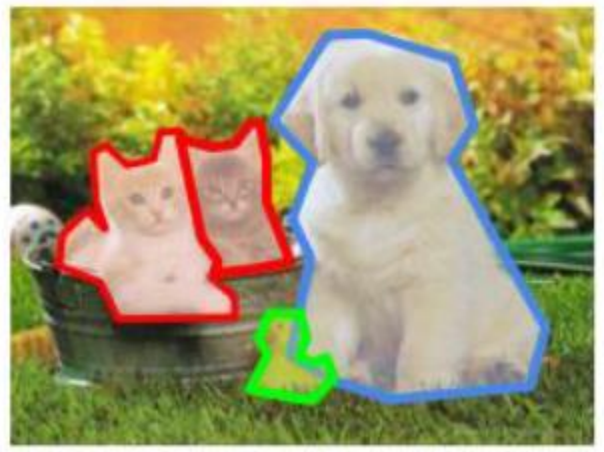
CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



CAT, DOG, DUCK

#01 Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

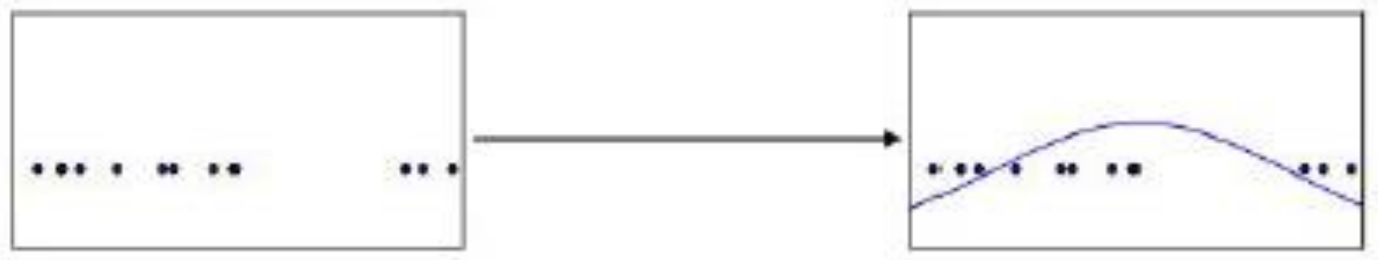
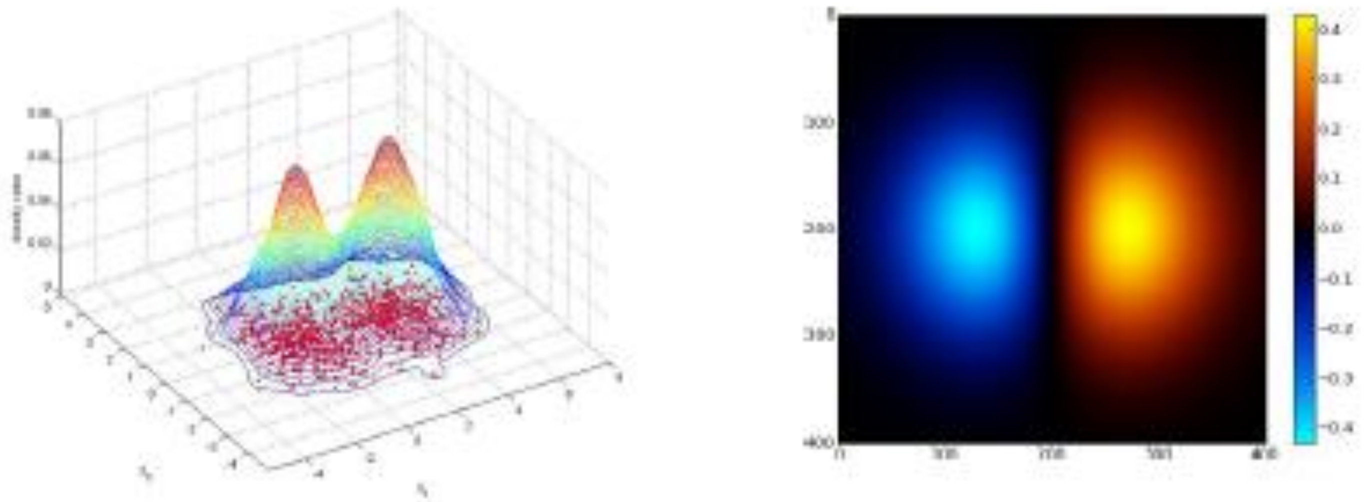


Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation



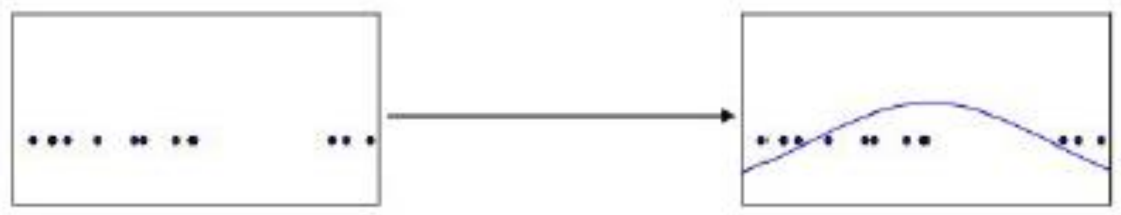
2-d density estimation

#01 Supervised vs Unsupervised Learning

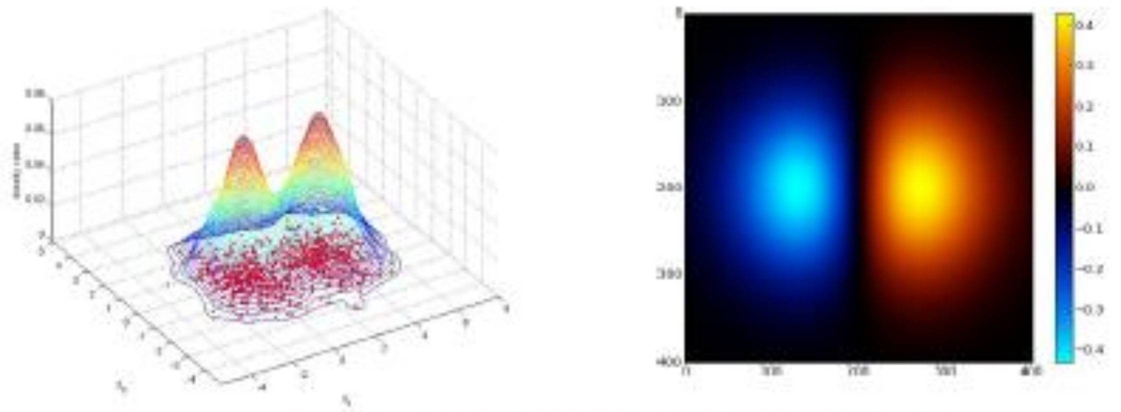
Unsupervised Learning

Data: x
Just data, no labels!

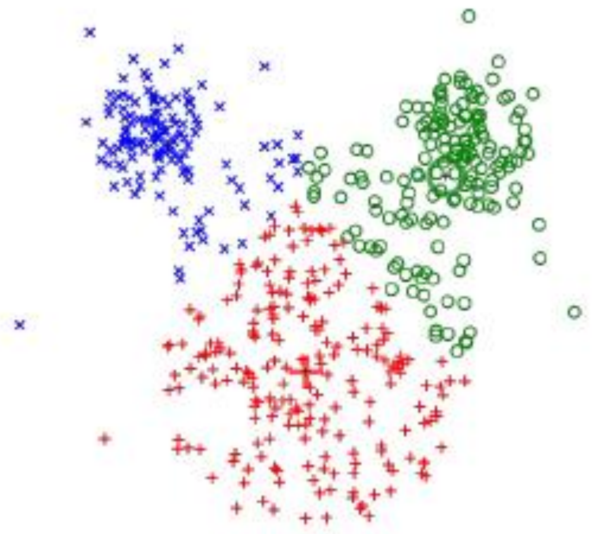
Goal: Learn some underlying hidden *structure* of the data



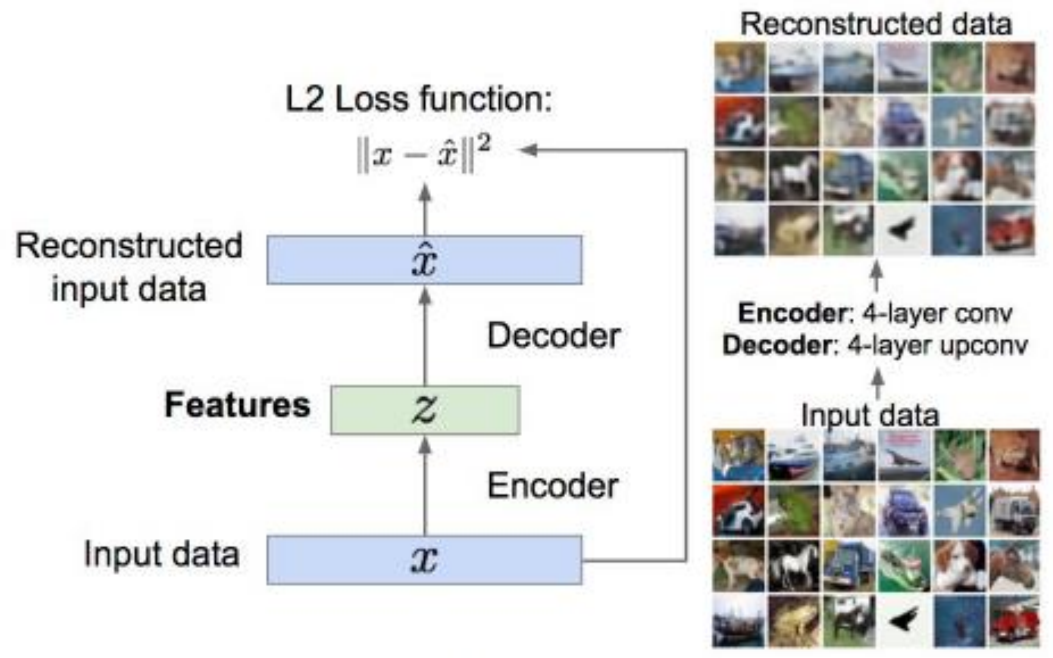
1-d density estimation



2-d density estimation

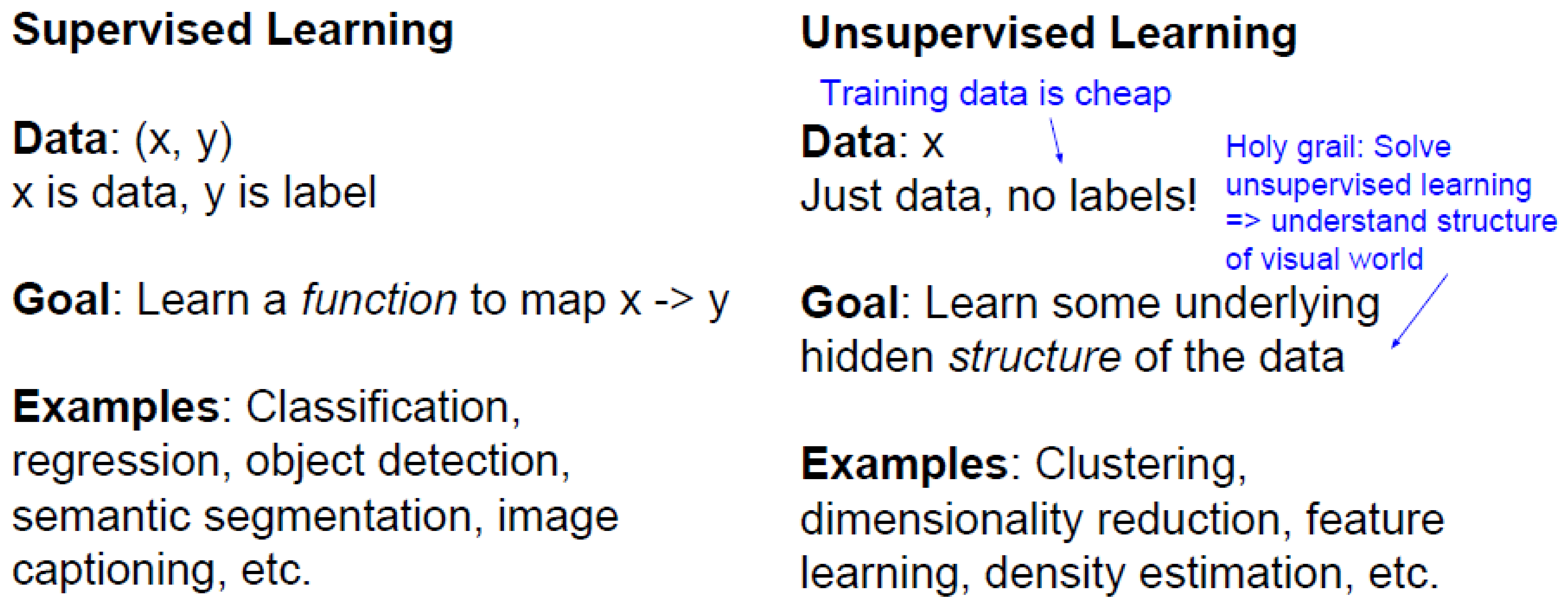


K-means clustering



Autoencoders
(Feature learning)

#01 Supervised vs Unsupervised Learning



Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Training data is cheap

Data: x
Just data, no labels!

Holy grail: Solve unsupervised learning => understand structure of visual world

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

#02 Generative model

Given training data, generate new samples from same distribution

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Generation : 이미지를 sampling하는 모델

Explicit model : 어떠한 입력이 주어질 때 이것에 대한 확률값을 얻어낼 수 있는 모델

Implicit model : 단순히 generation만 할 수 있는 모델

#02 Generative model

Taxonomy of Generative Models

Today: discuss 3 most popular types of generative models today

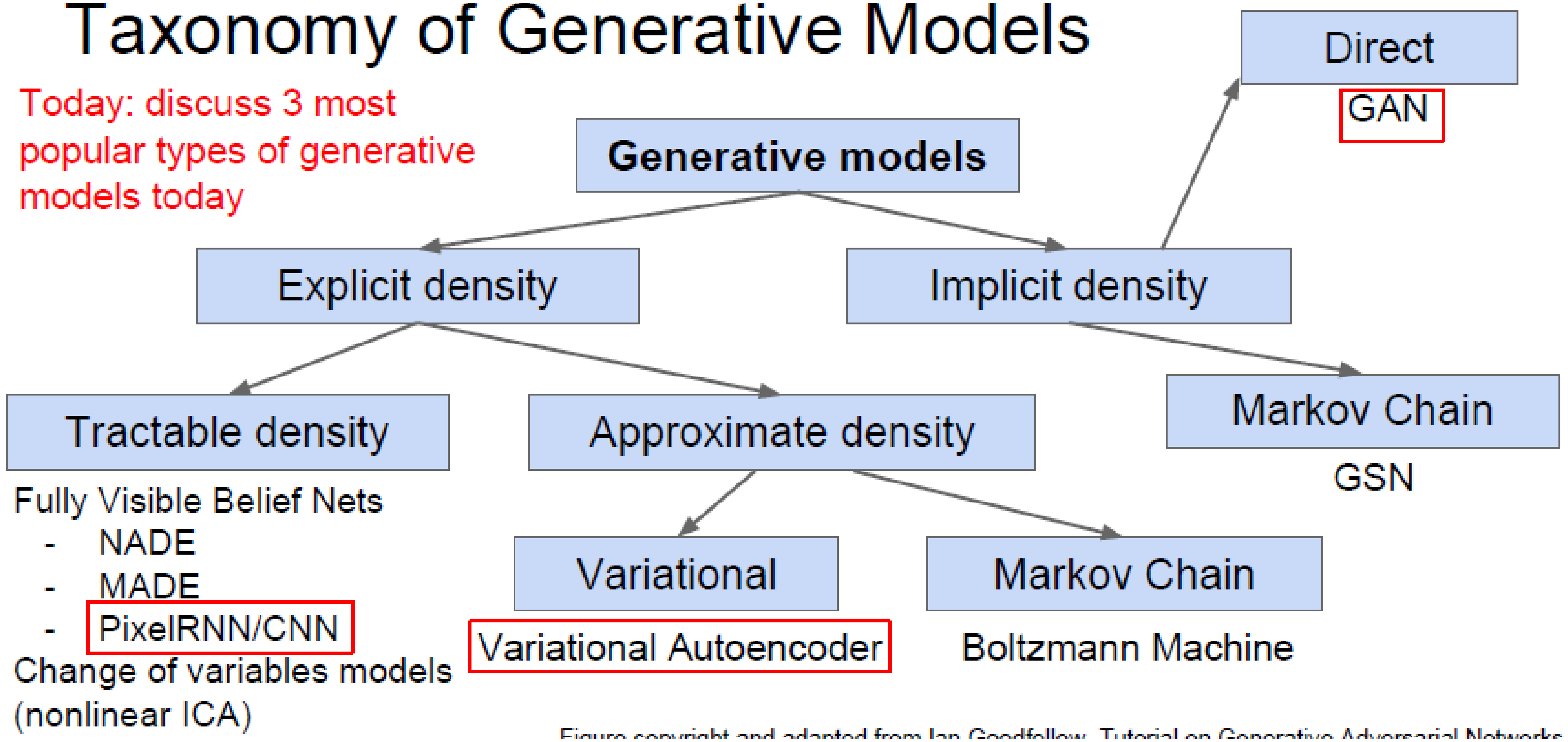
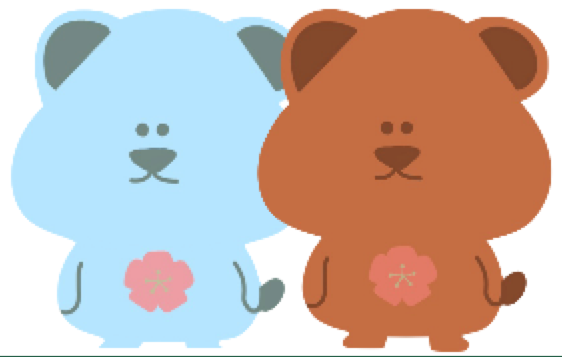


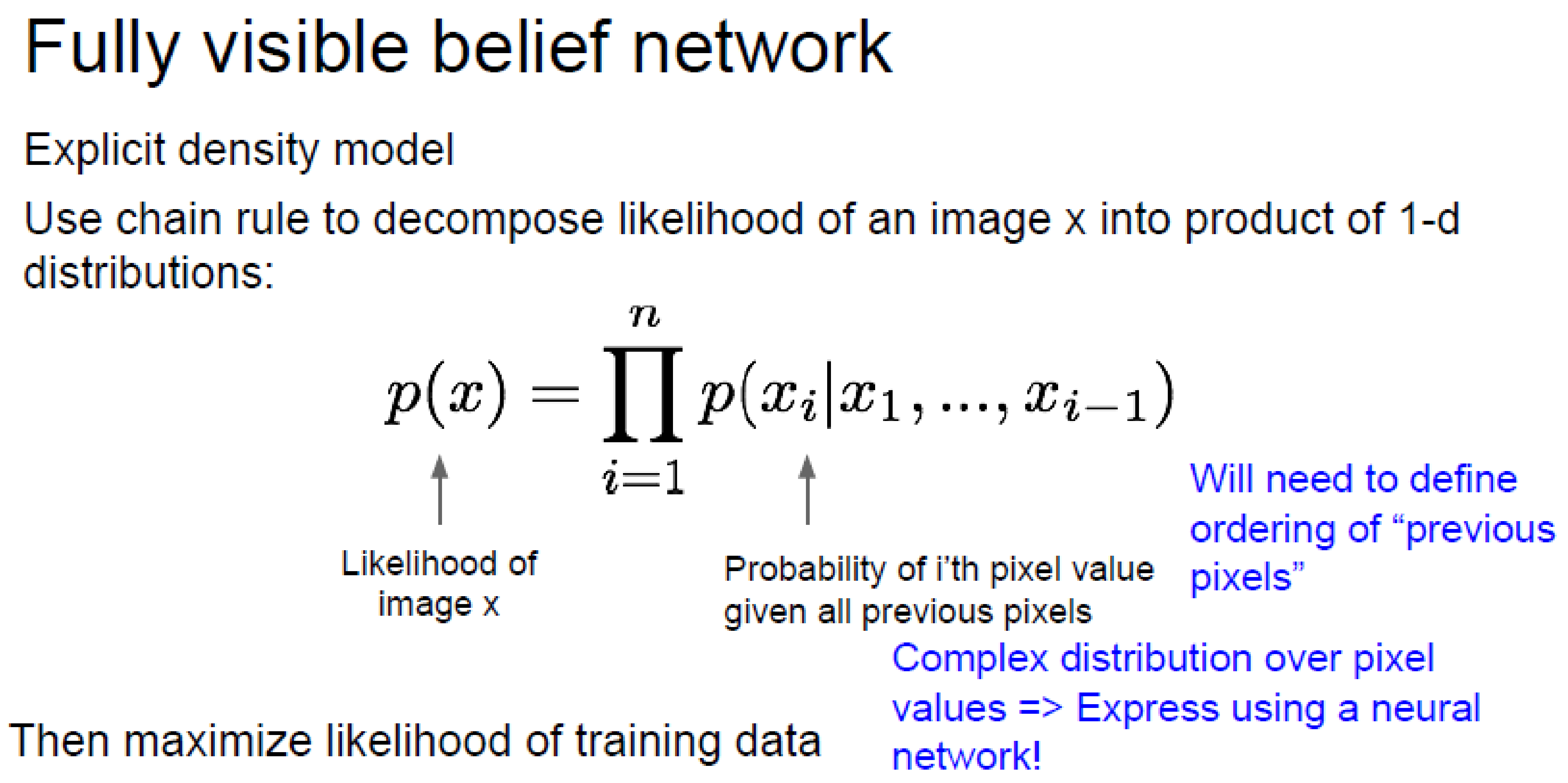
Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017

PixelRNN and PixelCNN



EWHA EURON

#03 PixelRNN and PixelCNN



Fully visible belief network

Explicit density model

Use 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

Will need to define ordering of "previous pixels"

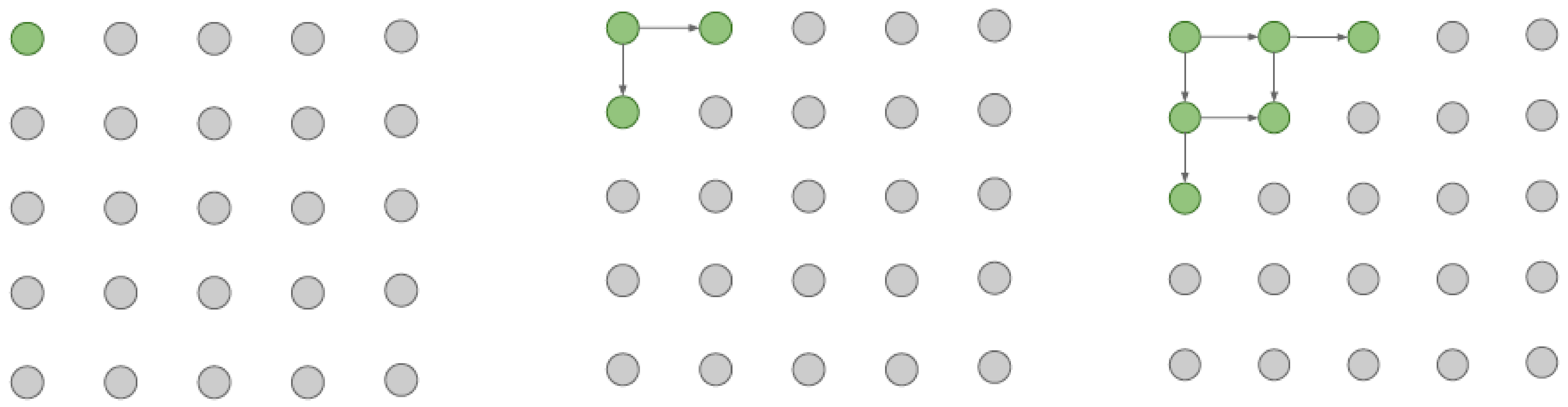
Complex distribution over pixel values => Express using a neural network!

Then maximize likelihood of training data

Explicit model은 모델이 어떠한 분포를 띄는지 정의하고 찾는데 초점을 둔다.
Training data의 likehood를 높이는 방향으로 학습! 해당 pixel들로 구성된 이미지가 나타날 확률은 각 pixel들의 확률곱을 의미한다.
모델을 정의하는 것이 한계가 있고 계산이 어려워지기 때문에 implicit model를 많이 선택.

#03 PixelRNN and PixelCNN

PixelRNN



코너에서 시작해서 image pixel을 생성. RNN(또는 LSTM)을 사용하여 이전 pixels의 종속성을 사용!

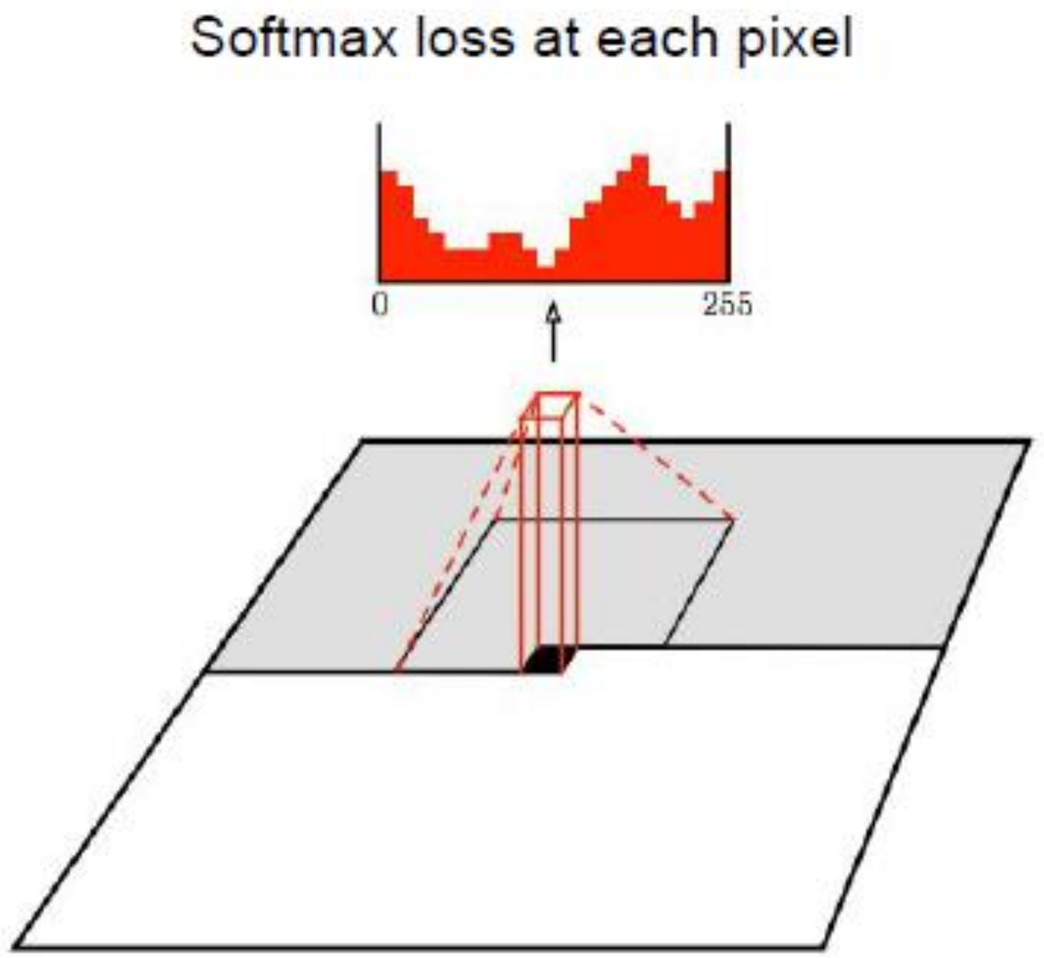
→ 순차적 생성이 매우 느리다는 결점을 가짐

#03 PixelRNN and PixelCNN

PixelCNN

Training: maximize likelihood of training images

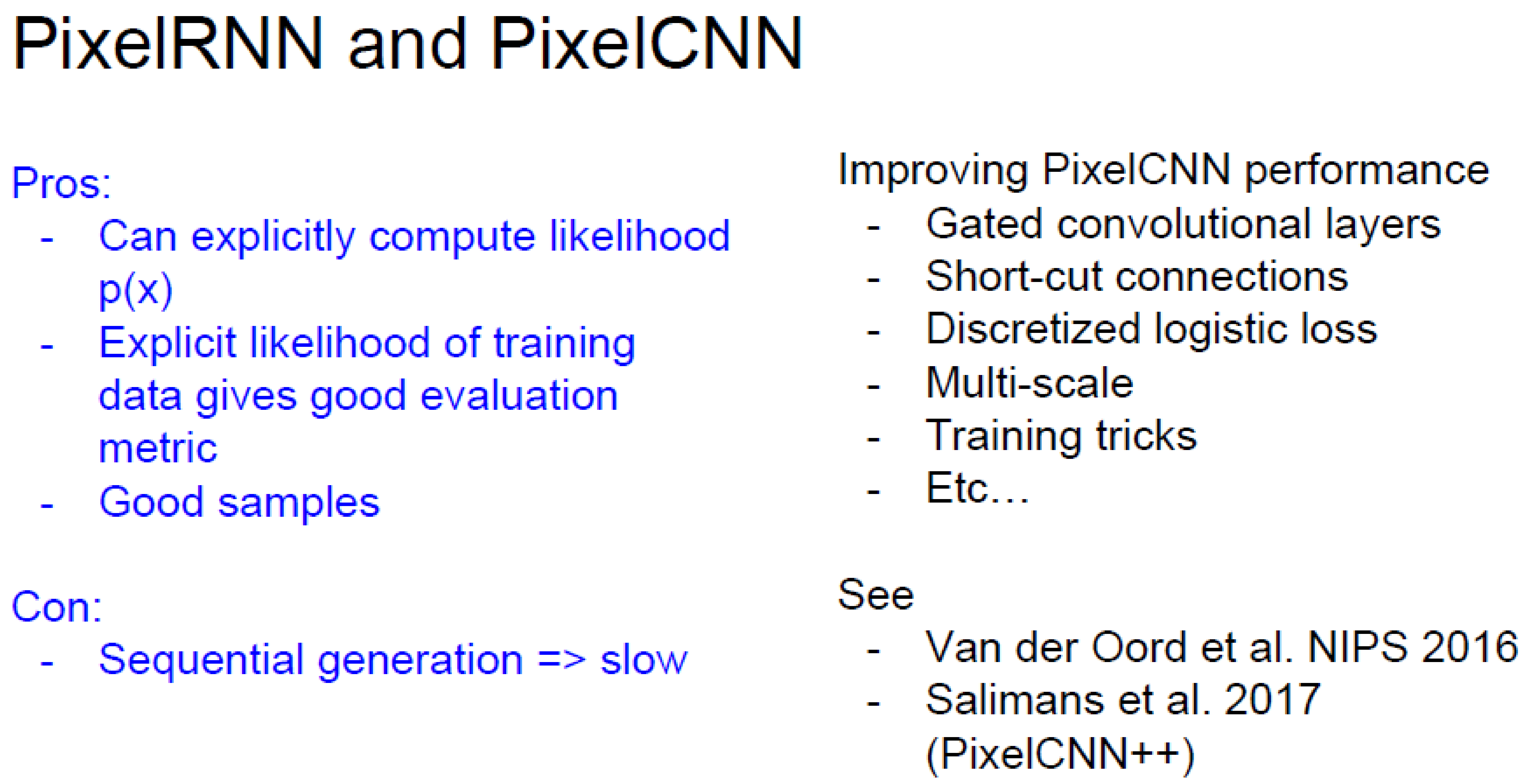
$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$



코너에서 시작해서 image pixel을 생성, CNN을 사용하여 이전 pixels의 종속성을 사용!

→ RNN보다 빠르지만 여전히 느리다는 단점을 가짐

#03 PixelRNN and PixelCNN



PixelRNN and PixelCNN

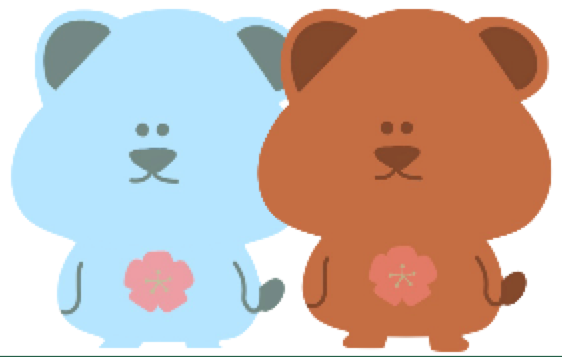
Pros:
- Can explicitly compute likelihood p(x)
- Explicit likelihood of training data gives good evaluation metric
- Good samples

Con:
- Sequential generation => slow

Improving PixelCNN performance
- Gated convolutional layers
- Short-cut connections
- Discretized logistic loss
- Multi-scale
- Training tricks
- Etc…

See
- Van der Oord et al. NIPS 2016
- Salimans et al. 2017 (PixelCNN++)

Variational Autoencoders(VAE)



#04 Variational Autoencoders(VAE)

PixelCNNs define tractable density function, optimize likelihood of training data:

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

VAEs define intractable density function with latent **z**:

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

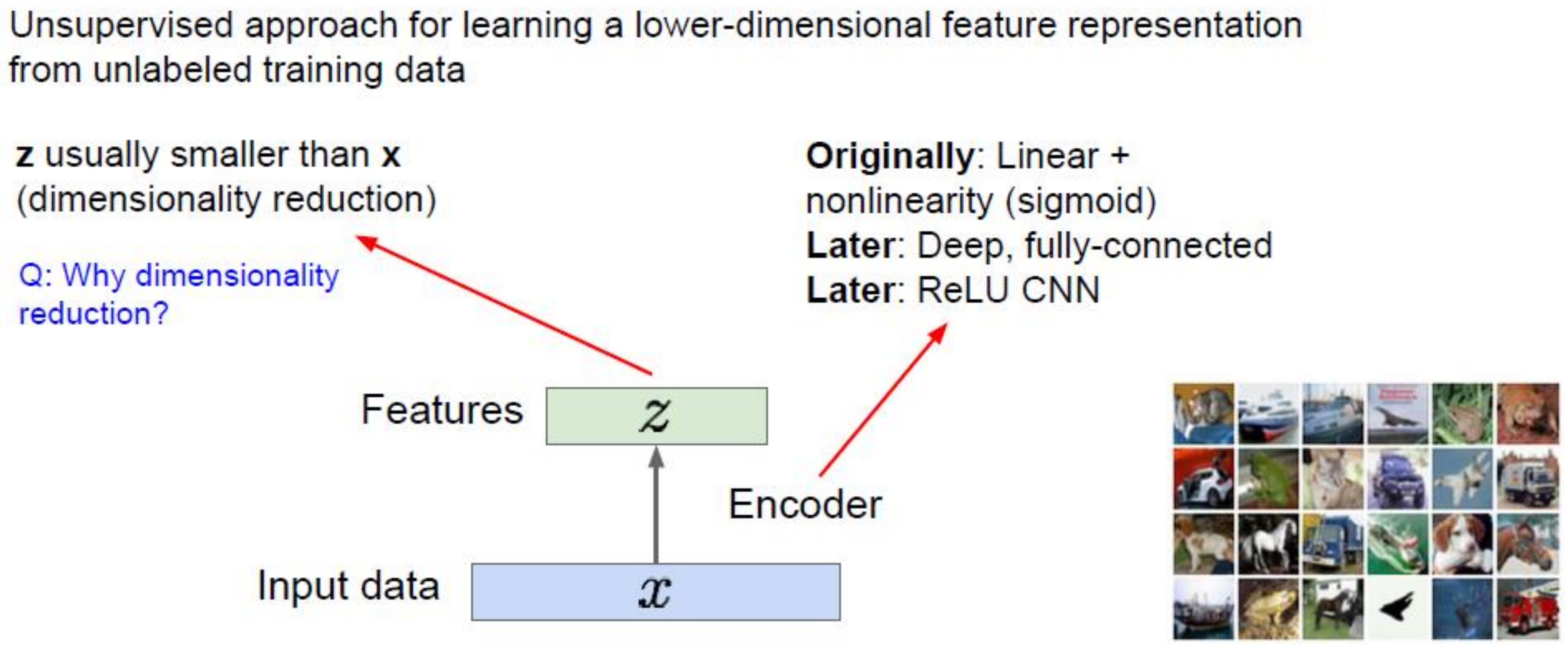
VAE는 intractable density function 이다. 즉 복잡해서 계산할 수 없는 함수를 갖는다.

계산을 못하기 때문에 function의 하한선을 찾아서 그 하한선을 maximize하는 방법을 채택!

그 방법으로 최적화를 대신한다

#04 Variational Autoencoders(VAE)

AE는 Auto Encoder로 VAE의 Background가 된다.



Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

레이블이 없는 데이터에서 feature representation을 뽑는 비지도 학습법!

#04 Variational Autoencoders(VAE)

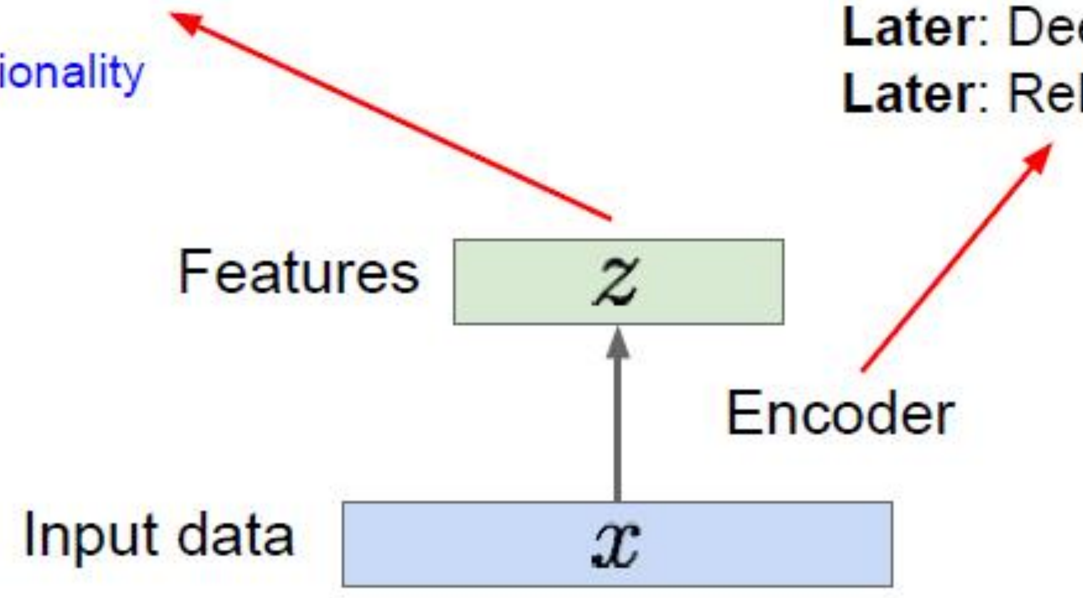
AE.

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x**
(dimensionality reduction)

Q: Why dimensionality reduction?

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

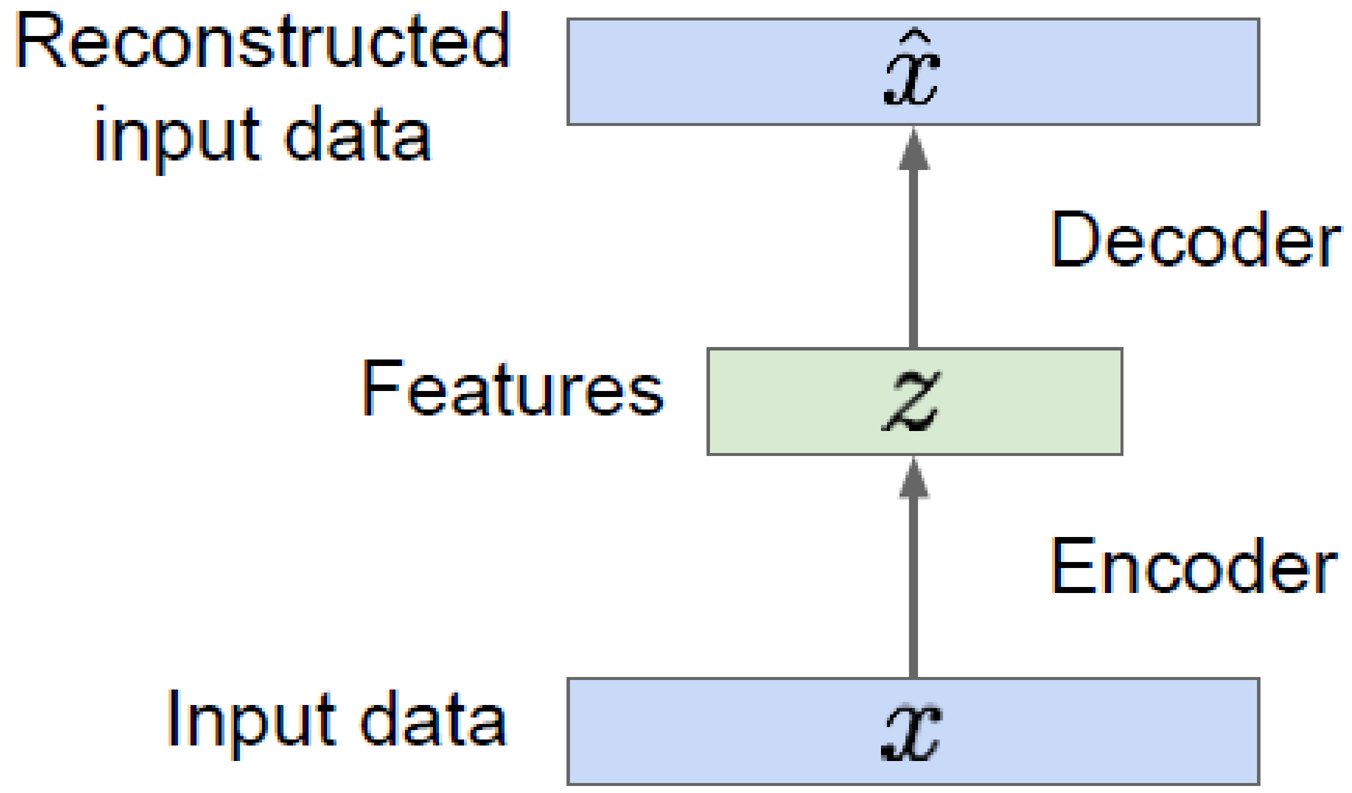


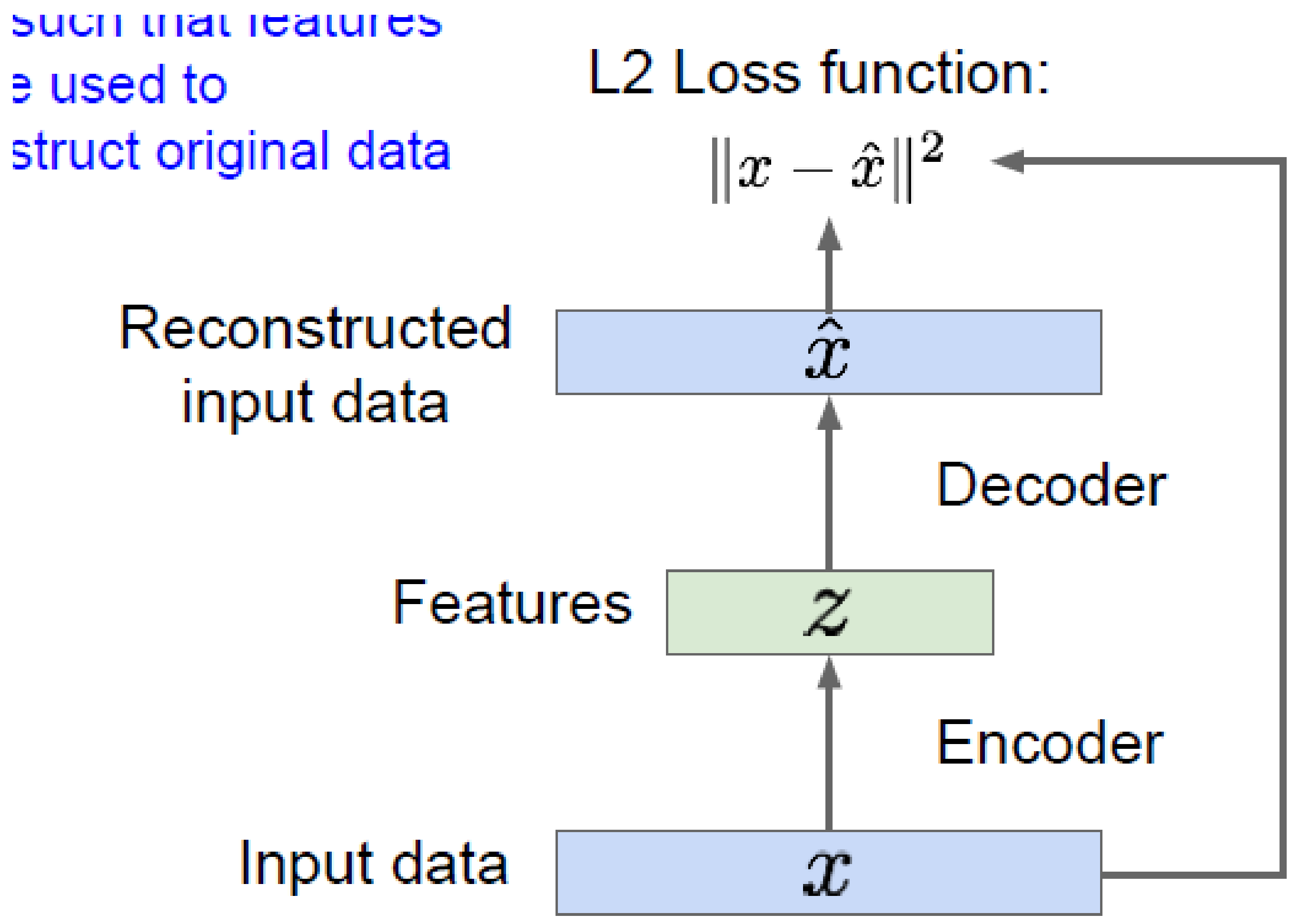
Q. Why dimensionality reduction?

A. Input data x를 특징 z에 매핑, z가 x보다 작게 지정되기 때문에 x를 차원 축소를 해야한다. 이때 z가 작게 지정되는 이유는 데이터에서 meaningful factors of variation만을 다루고 싶기 때문이다.

#04 Variational Autoencoders(VAE)

AE.



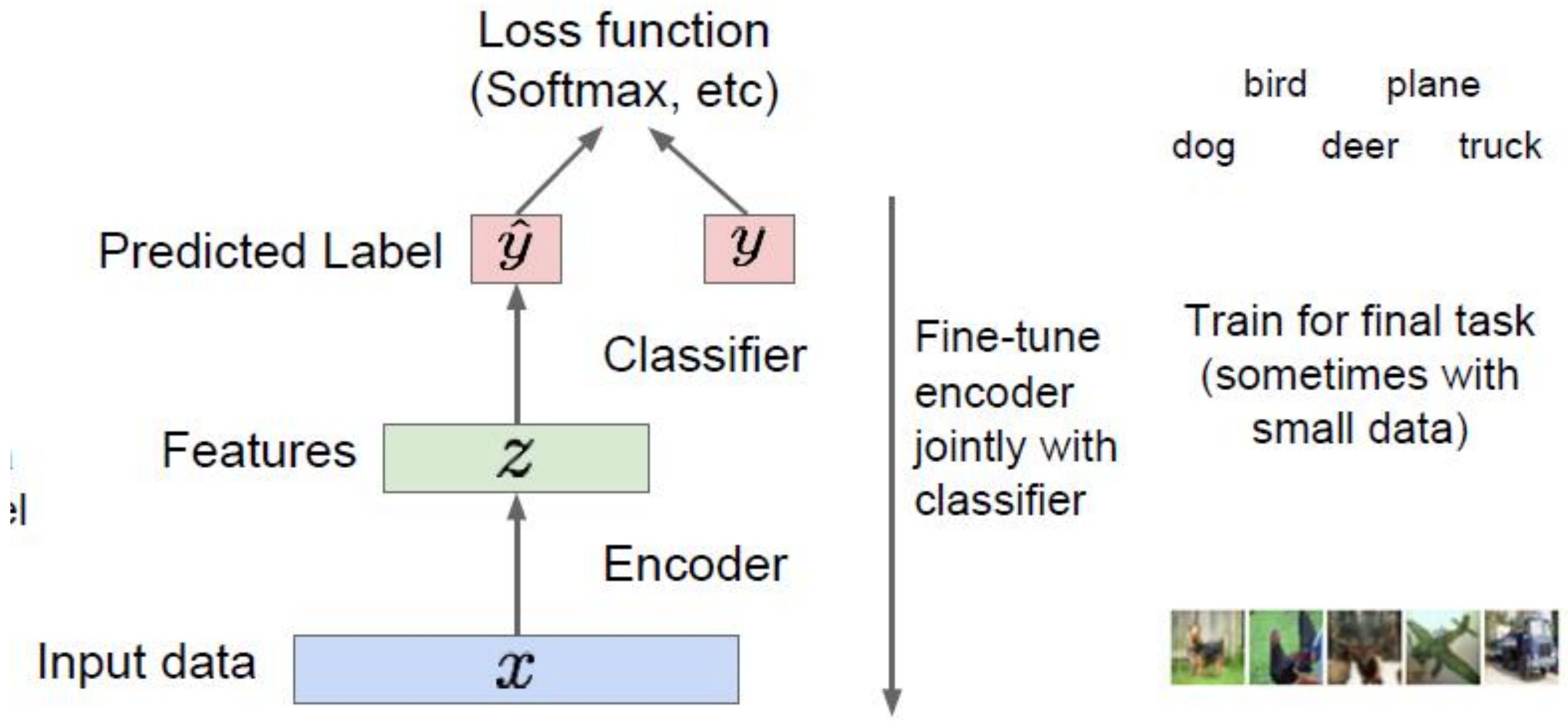


1. Input data x 에서 feature vector z를 추출한다(Encoder)
2. Feature z에서 reconstruct input data x^를 다시 만들어낸다 (Decoder)

→ 원본 x 와 재구성된 x^의 차이를 최대한 줄이도록 feature z 를 학습하는 것이 목표!
→ No labels, input data만 필요

#04 Variational Autoencoders(VAE)

AE.

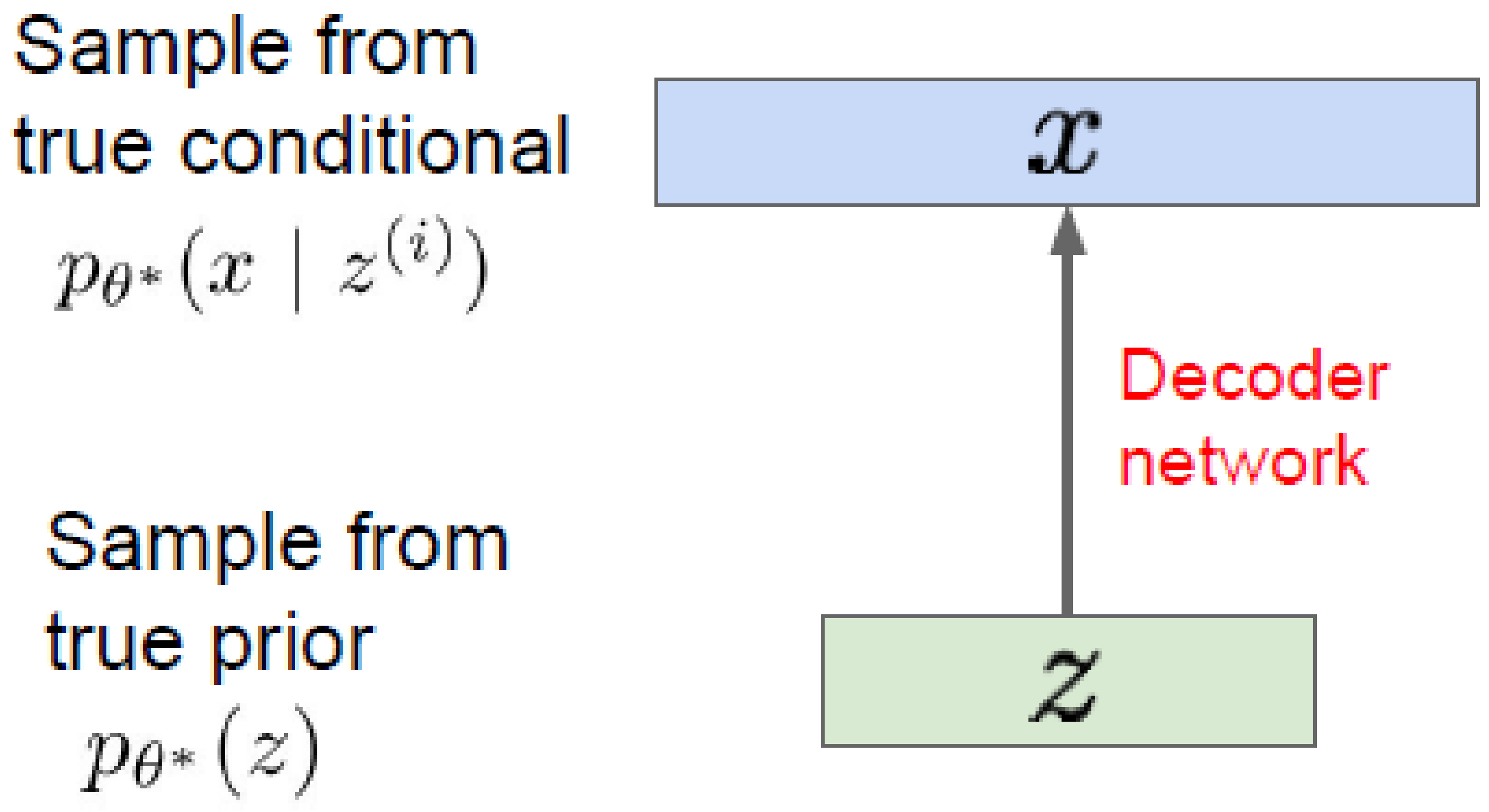


3. Decoder를 제거하고 feature를 적절한 학습을 통해 input data의 중요한 feature를 추출
4. Features 다시 supervised model input으로 넣어서 classification 하는데 사용

→ Feature를 data의 특성에 맞게 잘 반영하고 추출하는 게 목표!
→ Data가 많지 않을 때, overfit/underfit 되는 걸 최소화하는 방법

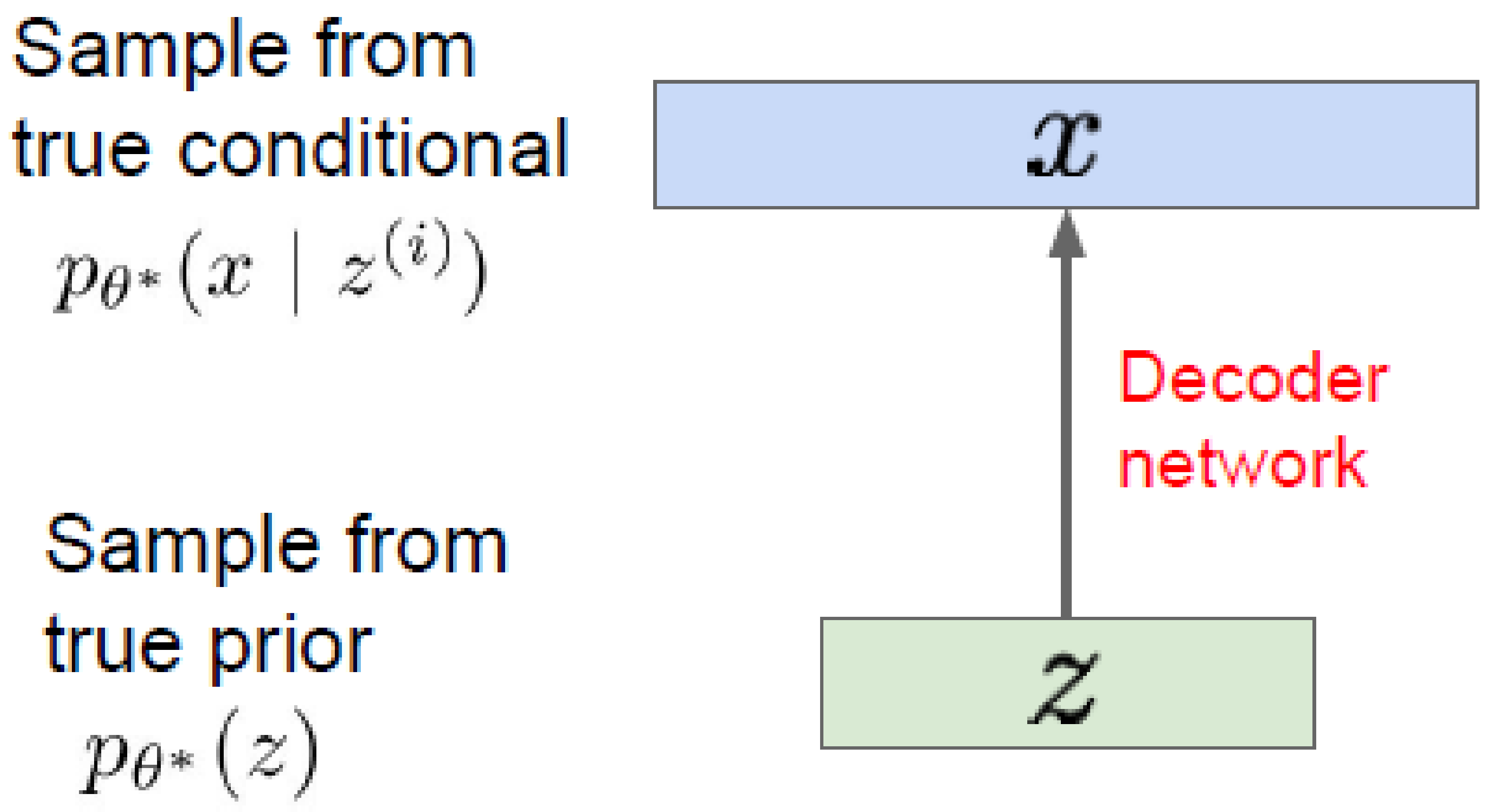
#04 Variational Autoencoders(VAE)

VAE.
(AE가 추출된 feature를 이용해 이미지를 분류했다면, VAE는 feature를 이용해서 새로운 이미지를 생성하는 것에 착안)



#04 Variational Autoencoders(VAE)

VAE.



Feature와 likelihood 분포를 사전에 sampling하고 그것은 z로 input 한다.
Function을 통해 x값을 생성하는 방식.

#04 Variational Autoencoders(VAE)

VAE.

How to train the model?

Remember strategy for training generative models from FVBNs. Learn model parameters to maximize likelihood of training data

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

Q: What is the problem with this?

Intractable!

Data likelihood: $p_\theta(x) = \overset{\text{☹}}{\int} \overset{\checkmark}{p_\theta(z)}\overset{\checkmark}{p_\theta(x|z)}dz$

Posterior density also intractable: $p_\theta(z|x) = \overset{\checkmark}{p_\theta(x|z)}\overset{\checkmark}{p_\theta(z)}/\overset{\text{☹}}{p_\theta(x)}$

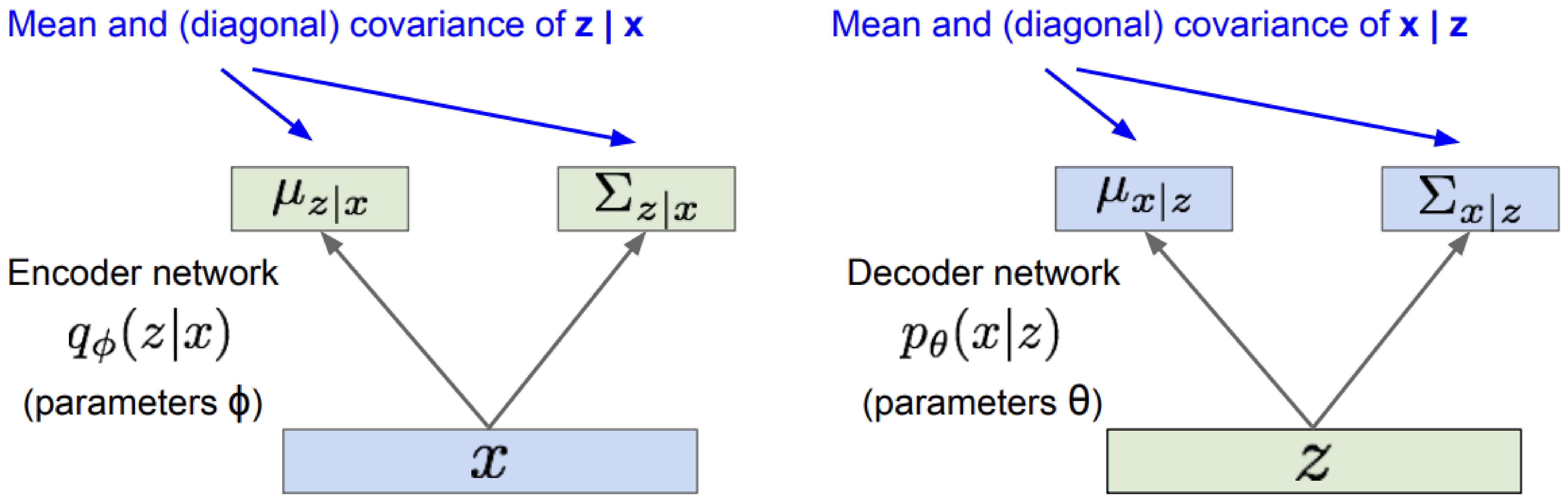
Intractable data likelihood

모든 z에 대해 계산이 매우 힘들기 때문에 다른 방법이 필요하다.
→ Encoder network q를 추가하여 p식을 근사(또는 하한) 을 이용하는 방식을 채택할 수 있다.

#04 Variational Autoencoders(VAE)

VAE

- encoder/decoder 구조
- + 확률론적인 의미 추가
- -> probabilistic generation 모델을 만드는 것!
- X와 z의 평균과 분산에 대한 분포를 생성하고, 이로부터 샘플링



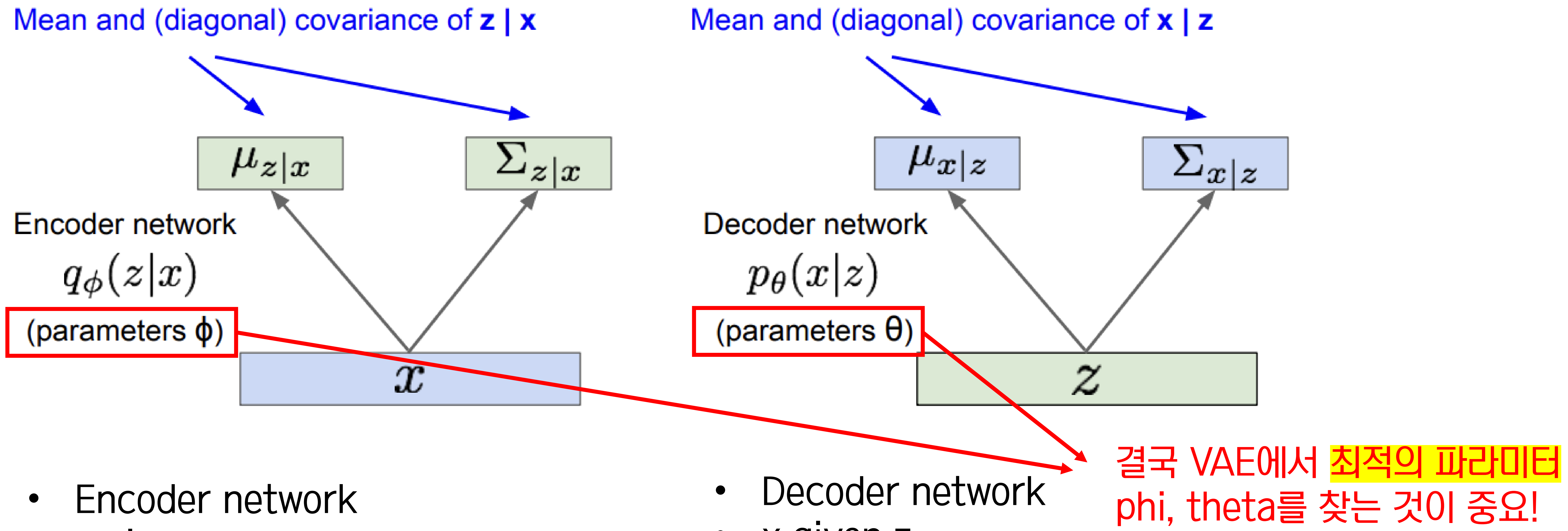
- Encoder network
- z given x
- 잠재 변수 z를 추론하는 네트워크
- “Recognition/inference network”

- Decoder network
- x given z
- z로부터 다시 x를 복원해내는 네트워크
- “Generation network”

#04 Variational Autoencoders(VAE)

VAE

- encoder/decoder 구조
- + 확률론적인 의미 추가
- -> probabilistic generation 모델을 만드는 것!
- X와 z의 평균과 분산에 대한 분포를 생성하고, 이로부터 샘플링



결국 VAE에서 최적의 파라미터 phi, theta를 찾는 것이 중요!

- Encoder network
- z given x
- 잠재 변수 z를 추론하는 네트워크
- “Recognition/inference network”

- Decoder network
- x given z
- z로부터 다시 x를 복원해내는 네트워크
- “Generation network”

#04 Variational Autoencoders(VAE)

최적의 파라미터란?
-> data likelihood p_theta(x)를 최대화시키는 파라미터!

z는 encoder network로 모델링한 q(z|x) 분포로부터 샘플링

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

학습데이터의 likelihood

p(x) -> log(p(x)) -> E(log(p(x)))
log 기댓값 … p(x)가 z에 독립적이므로

#04 Variational Autoencoders(VAE)

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

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

베이즈 룰 적용

사전 확률
(prior)

$$\boxed{P(H|E)} = \frac{P(E|H)\boxed{P(H)}}{P(E)}$$

사후 확률
(posterior)

#04 Variational Autoencoders(VAE)

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})
\end{aligned}$$

#04 Variational Autoencoders(VAE)

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})
\end{aligned}$$

#04 Variational Autoencoders(VAE)

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad &&(p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad &&(\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad &&(\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad &&(\text{Logarithms}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}$$

#04 Variational Autoencoders(VAE)

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

- Decoder network
- 샘플링을 이용해서 계산 가능

Kullback Leibler Divergence (KL)

- 특정 사건이 일어날 확률

$$P(X)$$

- 사건의 정보량

$$I(X) = -log_2 P(X)$$

- 엔트로피

$$H(X) = -\Sigma P(x) log_2 P(X) = E(I(X))$$

- 크로스 엔트로피

$$H(P,Q) = -\Sigma P(x) log_2 Q(X) = E_P(I_Q(X))$$

- KL

$$KL(P||Q) = H(P,Q) - H(P)$$

- P와 Q라는 두 분포의 차이를 나타내는 지표
- 항상 0보다 크거나 같다

#04 Variational Autoencoders(VAE)

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

- Encoder와 prior z 두 분포의 차이를 나타내는 지표
- q(z | x)는 encoder에서 발생하는 분포로 평균/공분산을 가지는 가우시안 분포
- p(z) 또한 가우시안 분포
- KL divergence에서 두 개의 분포가 모두 가우시안이면 closed from solution으로 풀 수 있다

- p(z | x)는 계산할 수 없는 항(untractable)

#04 Variational Autoencoders(VAE)

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

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

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

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

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

Tractable lower bound

>= 0

VAE를 학습시키기 위해
-> Lower bound가 최대가 되도록 최적화시킨다.
-> Lower bound를 최대로 만드는 파라미터 theta와 phi를 구한다.

#04 Variational Autoencoders(VAE)

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

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

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

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

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

데이터를 잘 복원해냄 $\uparrow$

Tractable lower bound

$\downarrow$ 잠재 변수 z의 분포를 prior 분포와 유사하게

$>= 0$

VAE를 학습시키기 위해
-> Lower bound가 최대가 되도록 최적화시킨다.
-> Lower bound를 최대로 만드는 파라미터 theta와 phi를 구한다.

#04 Variational Autoencoders(VAE)

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms}) \\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}
\end{aligned}$$

Reconstruct the input data

Make approximate posterior distribution close to prior

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

#04 Variational Autoencoders(VAE)

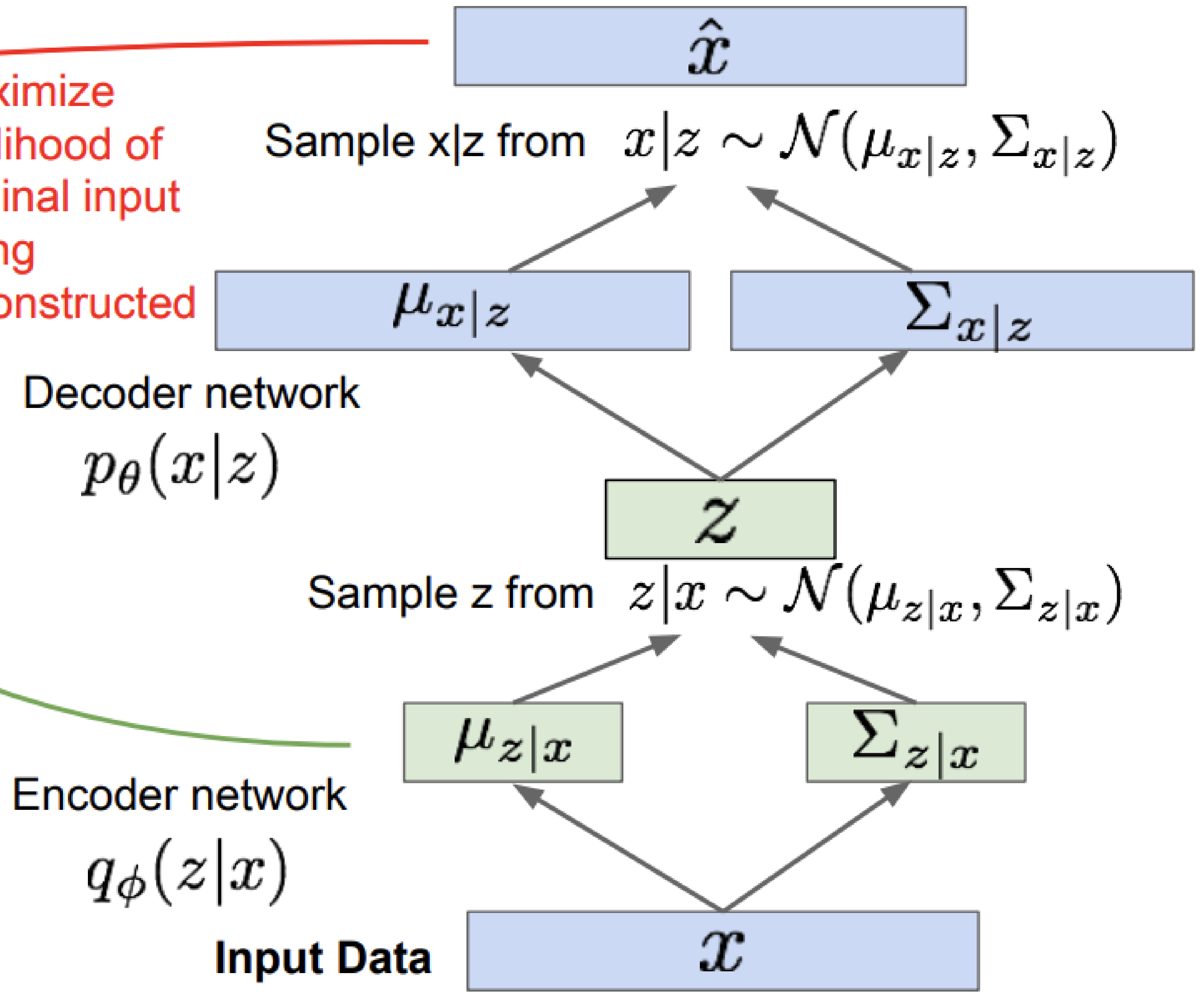
VAE를 학습시키는 과정

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

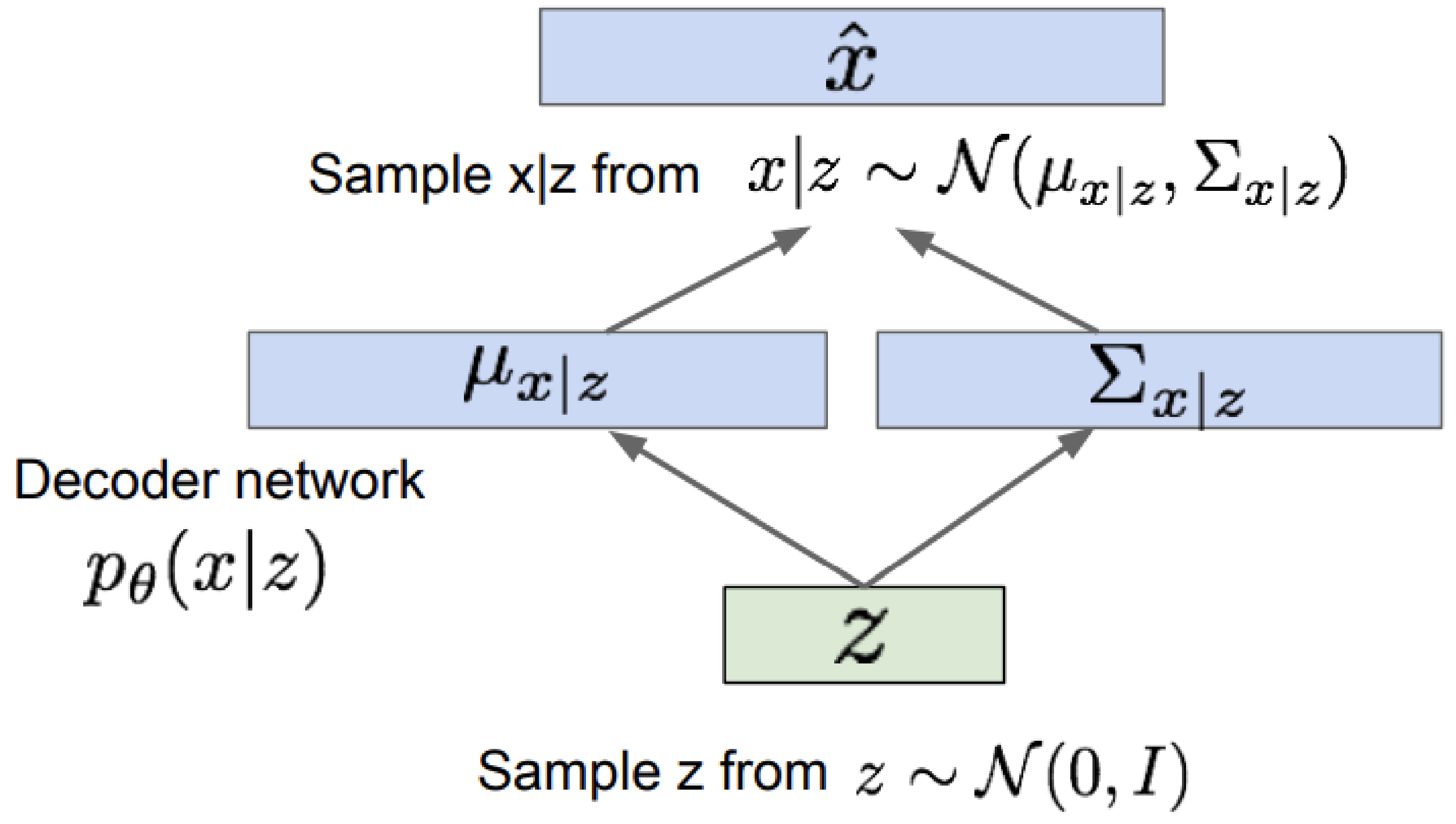
Make approximate posterior distribution close to prior

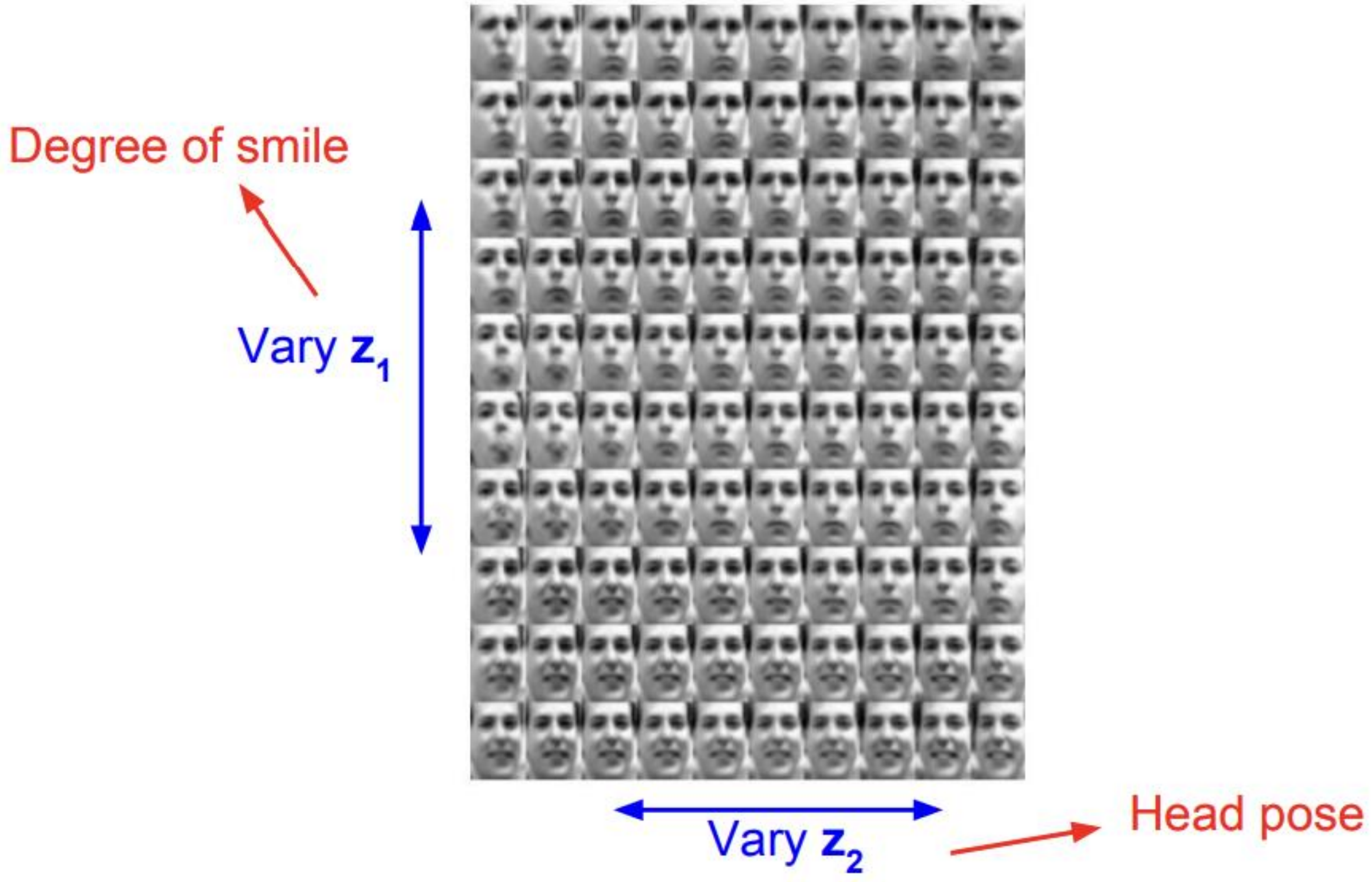


#04 Variational Autoencoders(VAE)

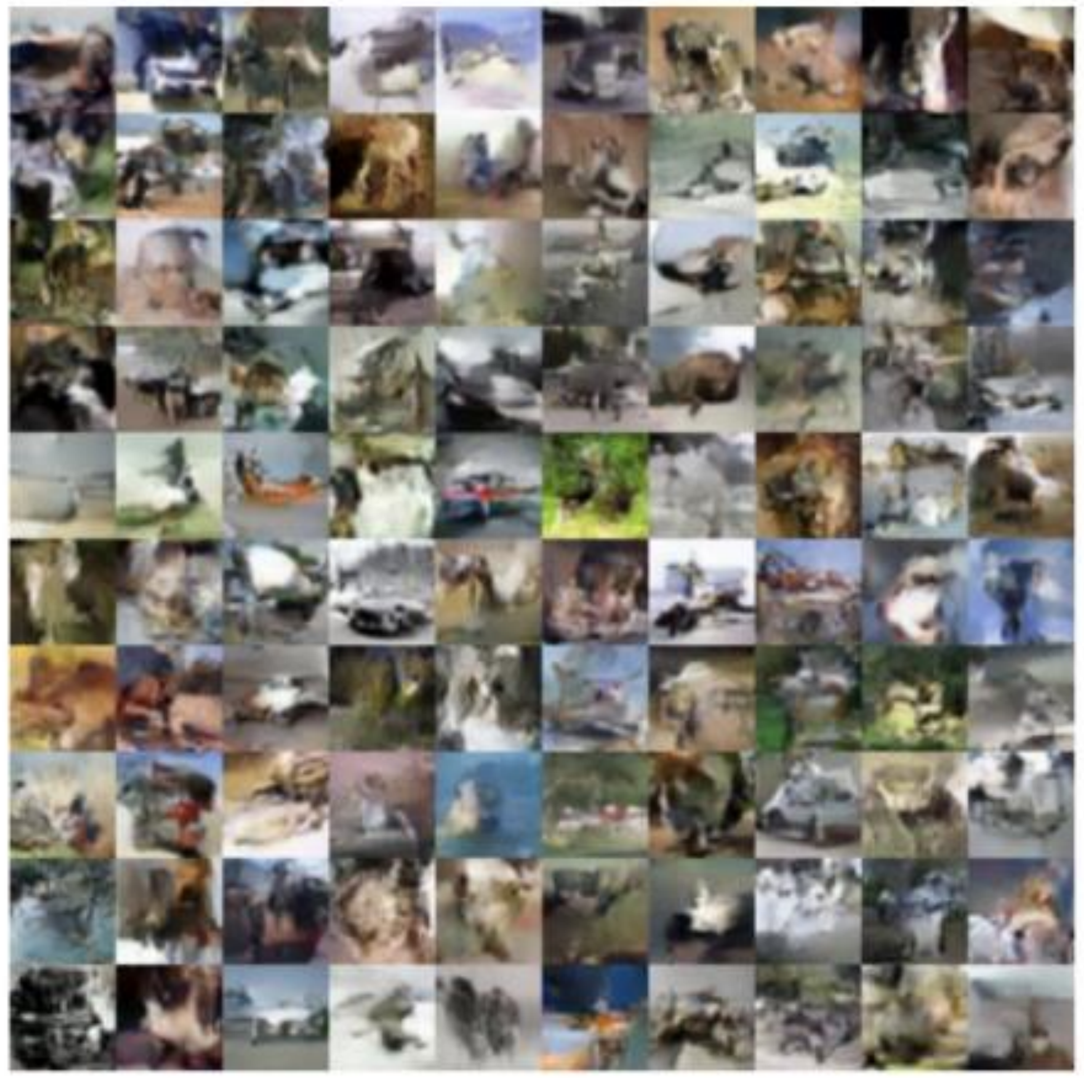
VAE가 데이터를 생성하는 과정

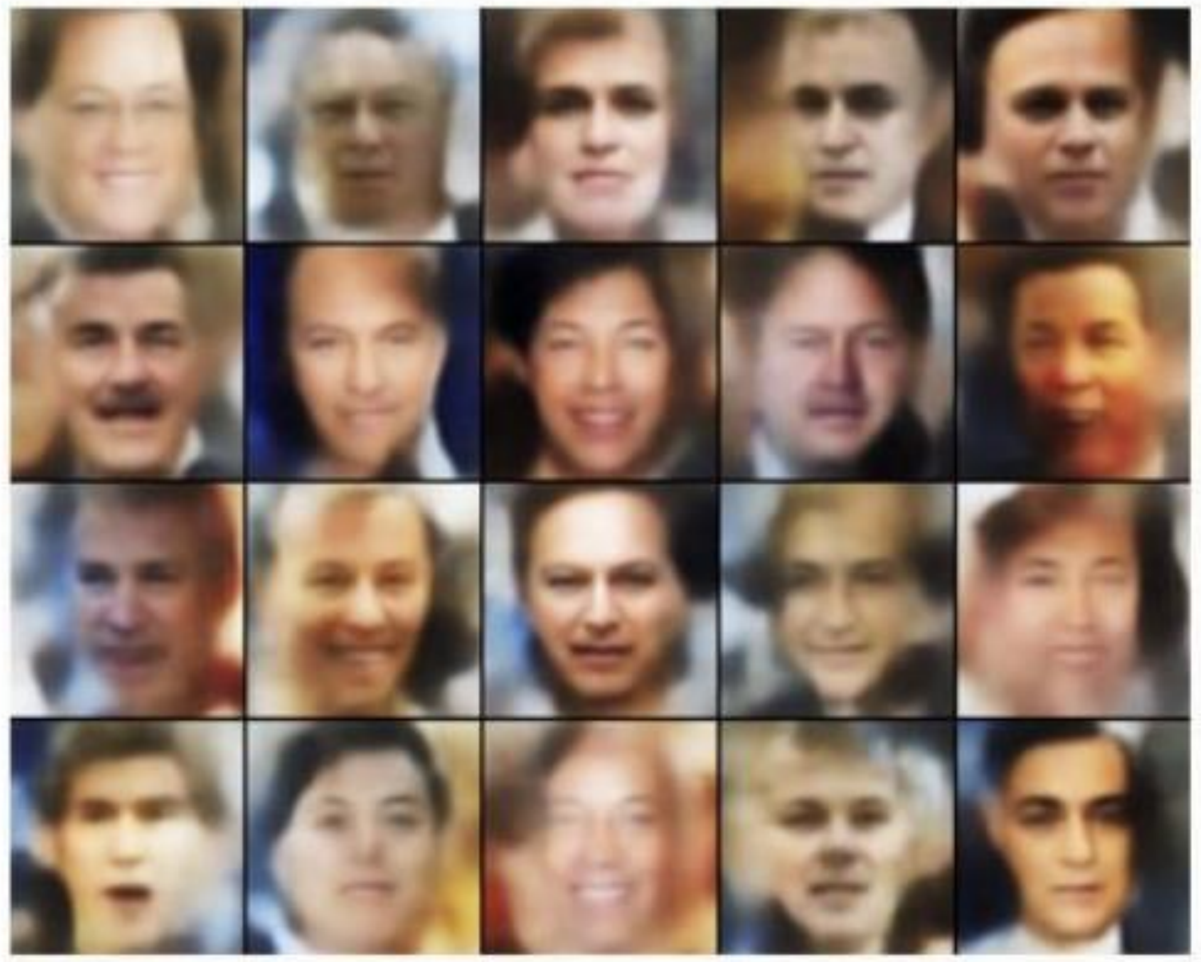
- Decoder network
- Prior z로부터 데이터 x를 샘플링
- Z의 각 차원이 독립적이라고 가정


$\hat{x}$
Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$
$\mu_{x|z}$
$\Sigma_{x|z}$
Decoder network
$p_\theta(x|z)$
z
Sample z from $z \sim \mathcal{N}(0, I)$



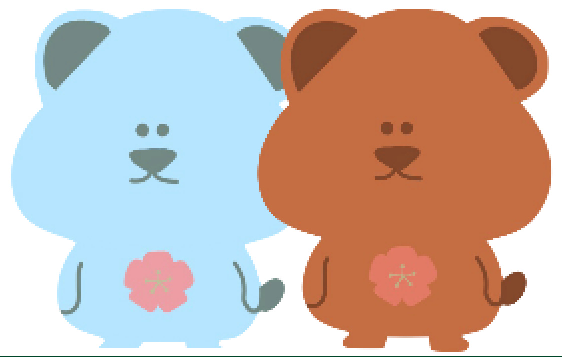
#04 Variational Autoencoders(VAE)


32x32 CIFAR-10


Labeled Faces in the Wild

VAE의 단점
-> 생성한 이미지들이 원본에 비해 blurry 함

Generative Adversarial Networks (GAN)



EWHA EURON

#05 GAN

Pixel CNN/RNN

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

- 계산 가능한 확률분포를 가정
- 학습데이터의 likelihood를 최적화

VAE

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

- 잠재변수 z를 이용
- 계산할 수 없는 확률분포를 가정
- 학습데이터의 likelihood를 직접 최적화
- 대신 lower bound를 정의해서 최적화

#05 GAN

Pixel CNN/RNN

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

- 계산 가능한 확률분포를 가정
- 학습데이터의 likelihood를 최적화

VAE

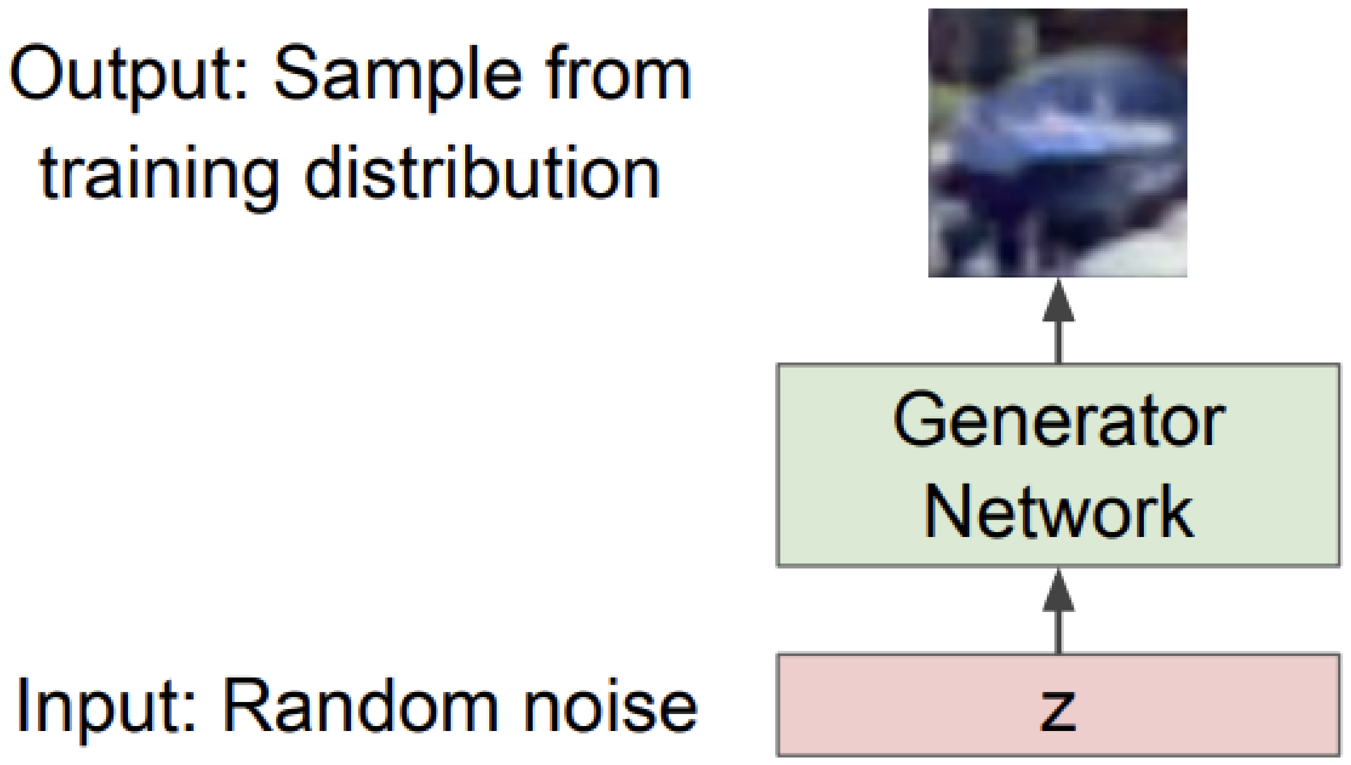
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

- 잠재변수 z를 이용
- 계산할 수 없는 확률분포를 가정
- 학습데이터의 likelihood를 직접 최적화
- 대신 lower bound를 정의해서 최적화

확률 분포를 직접 모델링하는 게 아니라,
생성모델이 확률 분포로부터 알아서 샘플링을 하는 방법은 없을까?

GAN!

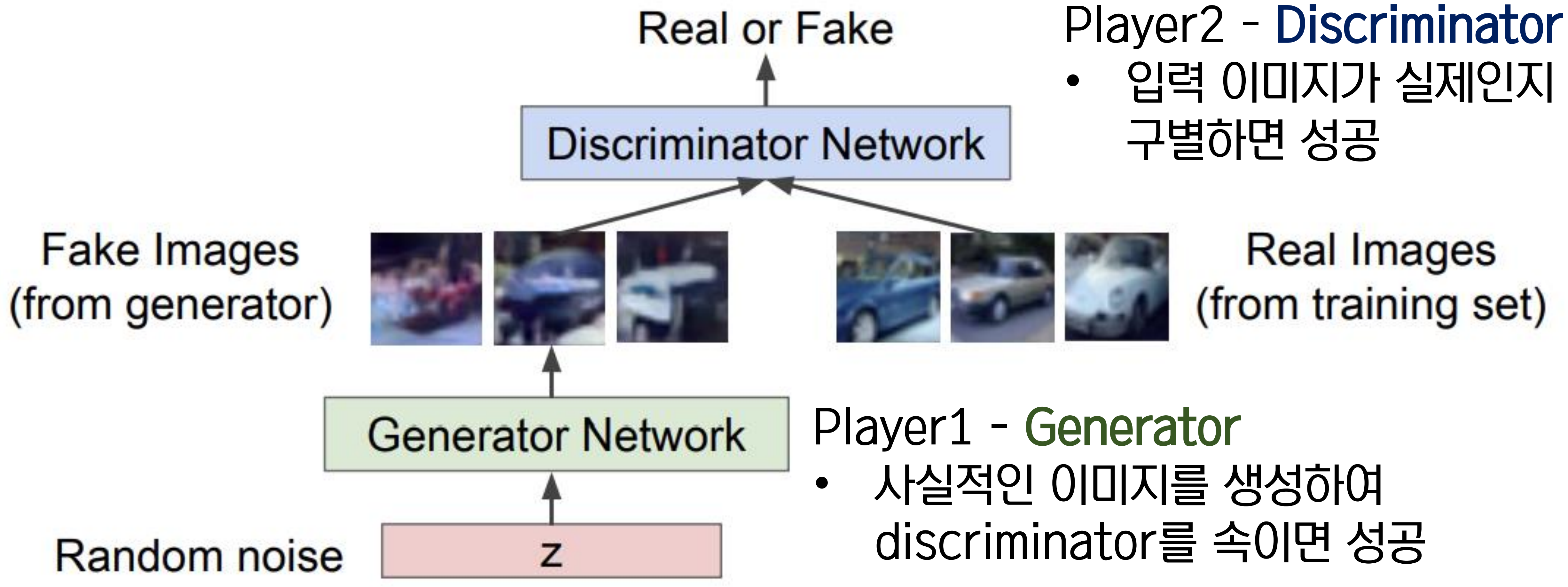
#05 GAN



Goal	복잡한 고차원 학습 분포로부터 샘플링
Problem	인풋으로 받아들인 분포가 아주 복잡하기 때문에 직접 샘플링하는 것이 불가능
Solution	random noise와 같은 단순한 분포를 이용해서 샘플링을 하자
	-> 이때 단순한 분포를 우리가 원하는 학습 분포로 변환하기 위해 neural network 이용!

#05 GAN

GAN을 학습시키는 방법: Two-player game



Player2 - Discriminator
- 입력 이미지가 실제인지 거짓인지를 구별하면 성공

Player1 - Generator
- 사실적인 이미지를 생성하여 discriminator를 속이면 성공

GAN의 아이디어
- Discriminator가 잘 학습돼서 실제/거짓 이미지를 잘 구별해냄
- Generator가 잘 학습돼서 더 실제같은 이미지를 만들어내어 discriminator를 잘 속임
  -> 성능이 좋은 generative model!

#05 GAN

GAN을 학습시키는 방법: Two-player game

- Generator에서 샘플링
- 생성된 fake 이미지에 대한 discriminator의 출력

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \boxed{\log D_{\theta_d}(x)} + \mathbb{E}_{z \sim p(z)} \log(1 - \boxed{D_{\theta_d}(G_{\theta_g}(z))}) \right]$$

- Real 데이터 x에 대한 discriminator의 출력
- X가 데이터 분포 p_data에 속할 likelihood

Discriminator는
- D(x)는 1에 가까울수록
- D(G(z))는 0에 가까울수록 좋음

Generator는
- D(G(z))가 1에 가까울수록 좋음

  -> discriminator가 fake 이미지를 real 이미지로 잘못 분류

  -> 즉, generator가 진짜같은 이미지를 잘 만들어 내고 있다!

#05 GAN

GAN을 학습시키는 과정

- Discriminator와 generator를 다음과 같이 번갈아 가면서 학습시킨다

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

#05 GAN

GAN을 학습시키는 과정

- Discriminator와 generator를 다음과 같이 번갈아 가면서 학습시킨다

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

하지만 실제로 generator의 objective function은 학습이 잘 일어나지 않는다

#05 GAN

GAN을 학습시키는 방법: Two-player game

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

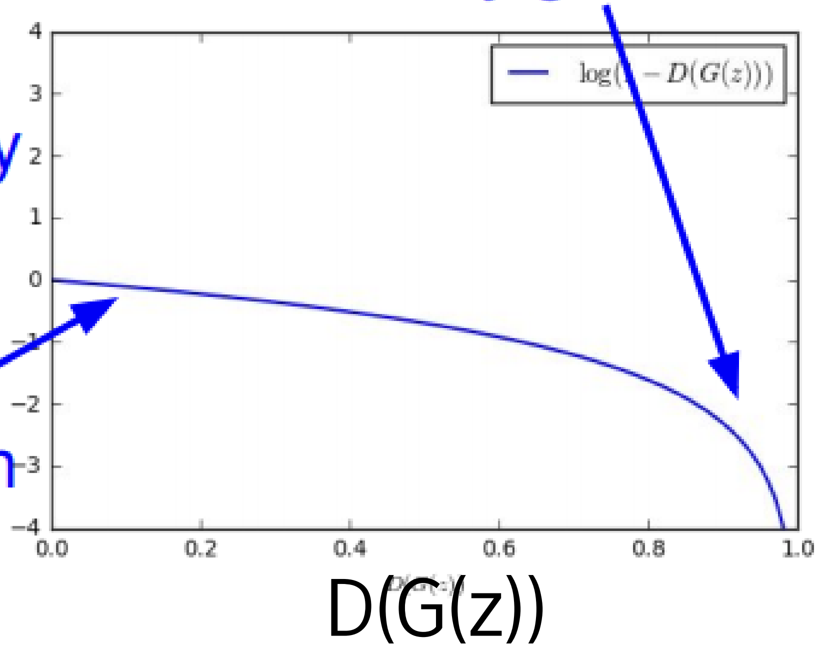
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Generator가 discriminator를 잘 속일 때

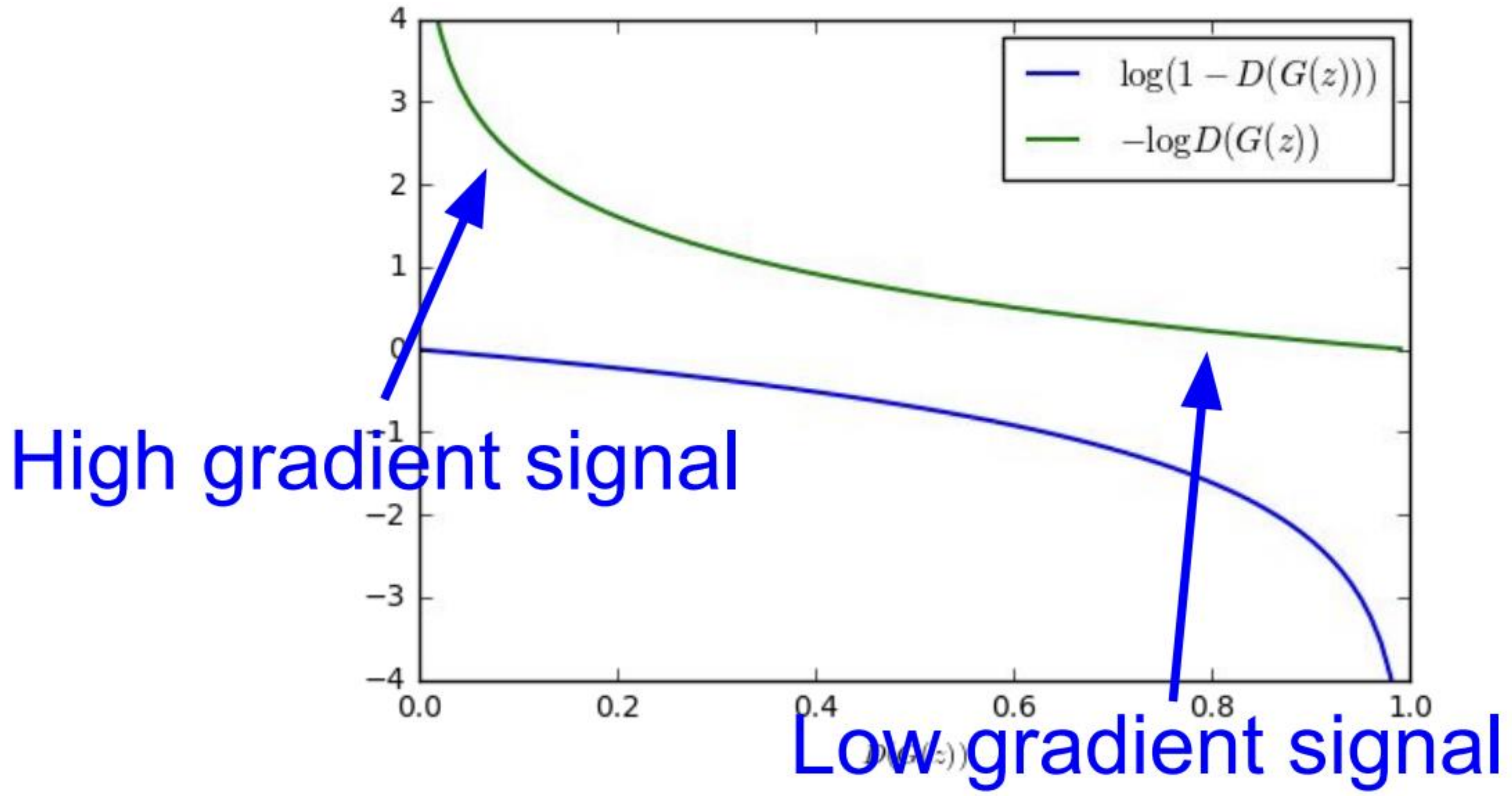
Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



샘플이 안 좋은 경우에 학습을 많이 시켜야 한다
-> Generator에서 gradient ascent를 이용한다
-> Discriminator가 틀릴 likelihood를 최대화 시키는 쪽으로 학습시킨다

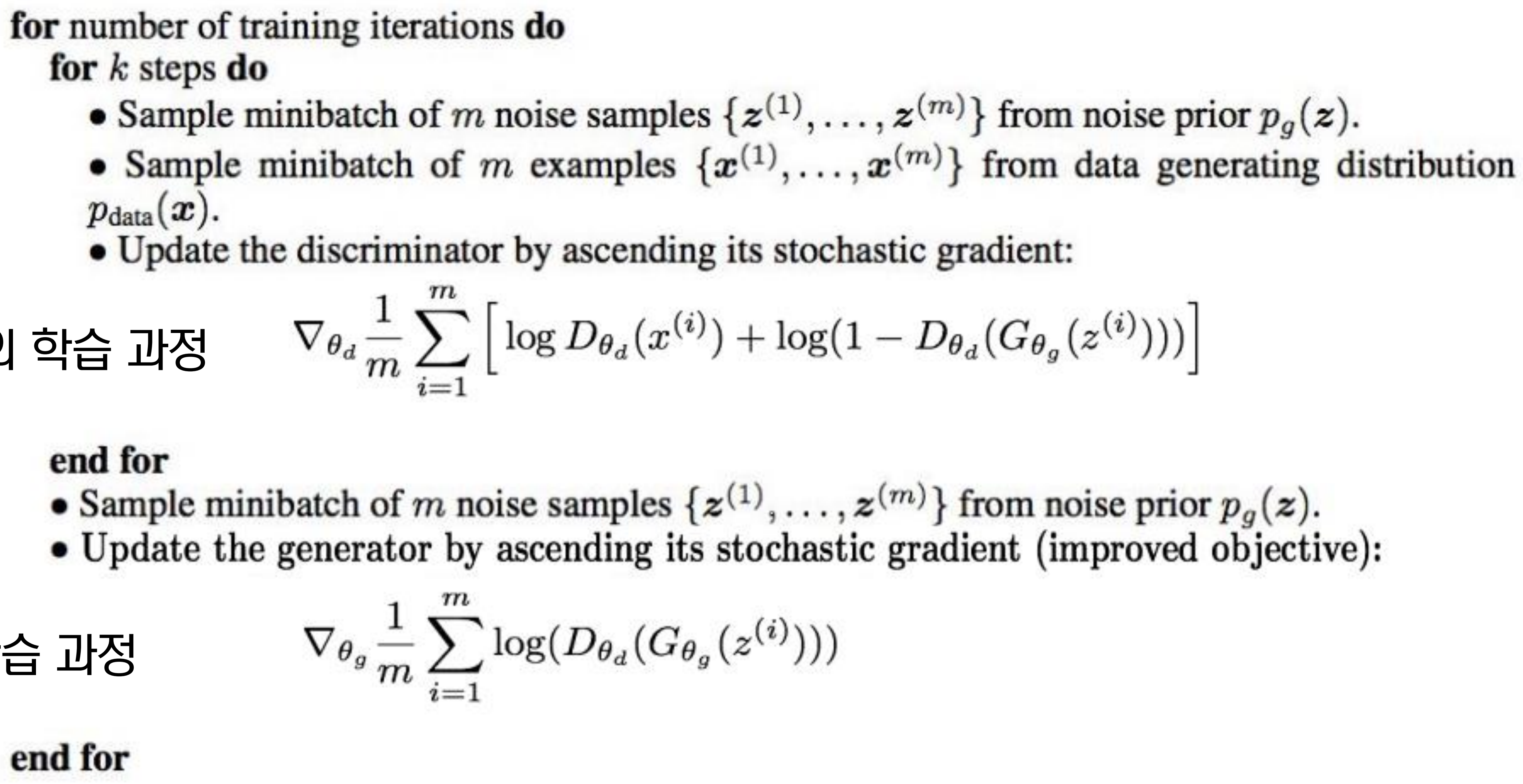
#05 GAN



$\log(1 - D(G(z)))$
$-\log D(G(z))$
High gradient signal
Low gradient signal
$D(G(z))$
4
3
2
1
0
−1
−2
−3
−4
0.0
0.2
0.4
0.6
0.8
1.0

#05 GAN

GAN의 학습 알고리즘



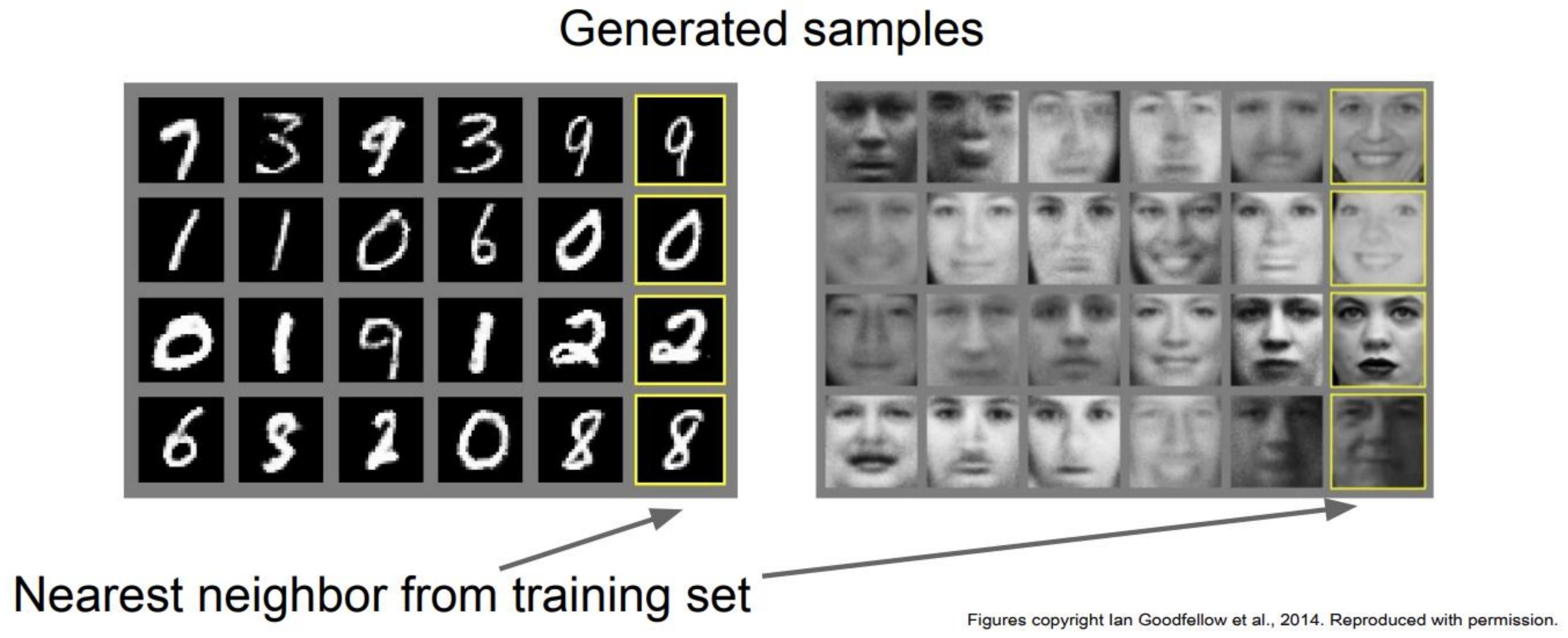
```
for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:
        $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$
  end for
  • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by ascending its stochastic gradient (improved objective):
      $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$
end for
```

Discriminator의 학습 과정

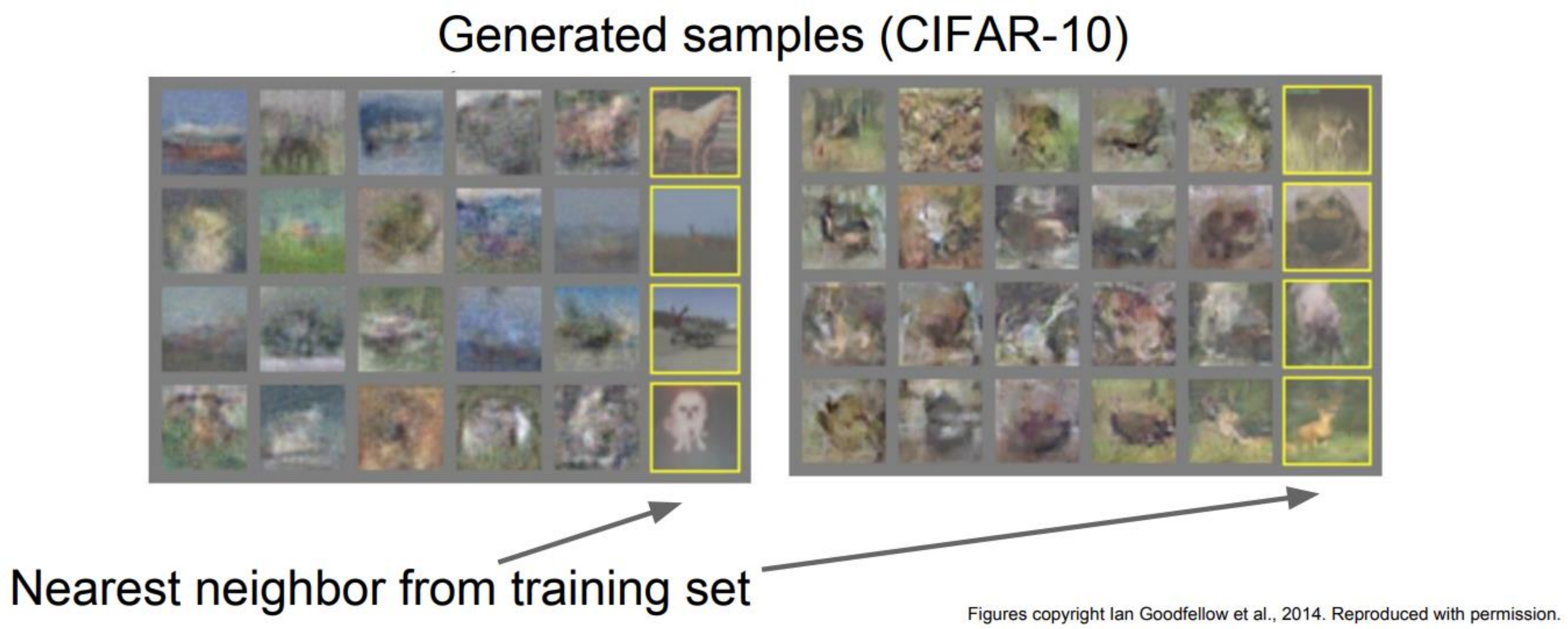
Generator의 학습 과정

#05 GAN

original GAN, 2014



Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.


Generated samples (CIFAR-10)
Nearest neighbor from training set

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.

#05 GAN

GAN + CNN (DCGAN)

Generative Adversarial Nets: Convolutional Architectures

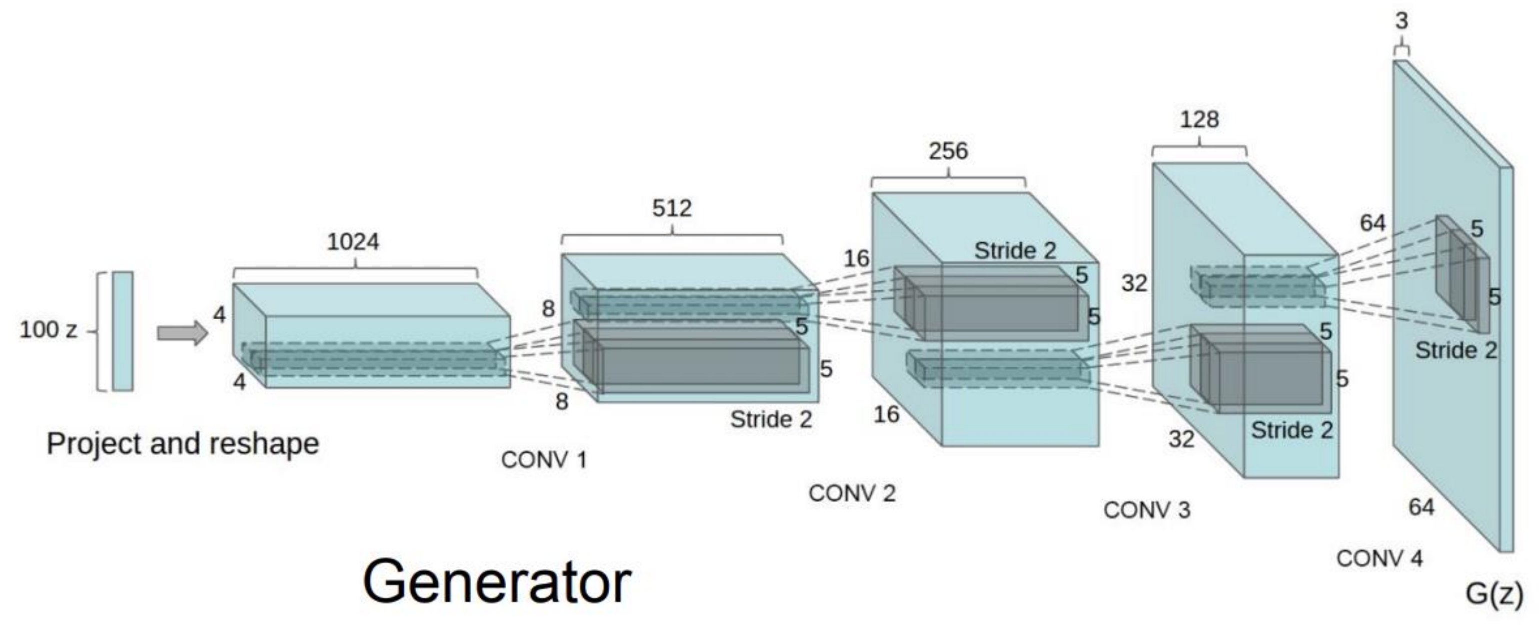
Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

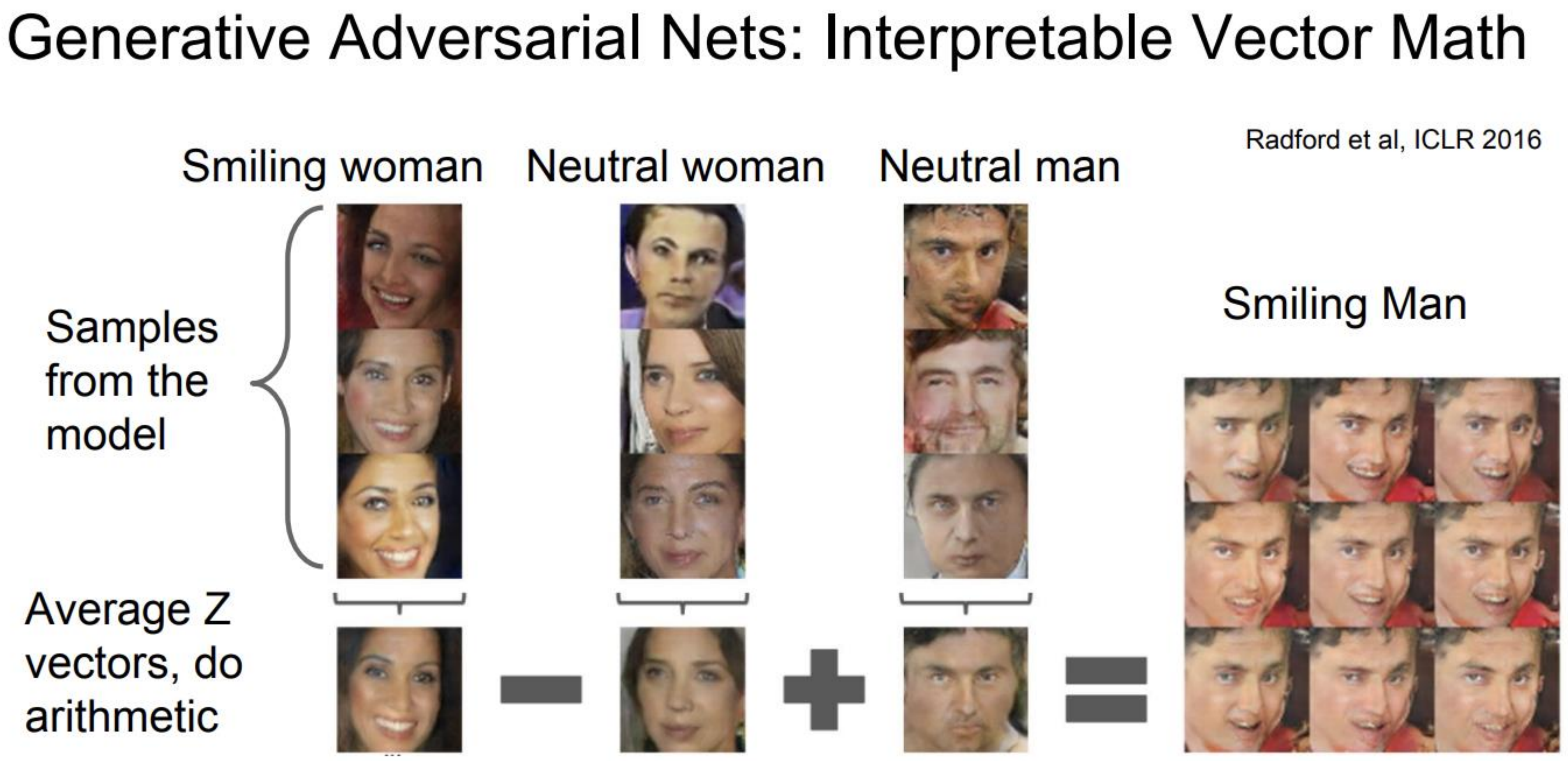
Generative Adversarial Nets: Convolutional Architectures



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

- GAN에 적용한 CNN 아키텍처
- 입력 노이즈 벡터 z에 대해 위와 같은 과정으로 샘플링이 일어남

#05 GAN


Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016

#05 GAN

Image-to-Image Translation with Conditional Adversarial Networks (Pix2Pix)
[CVPR 2017]

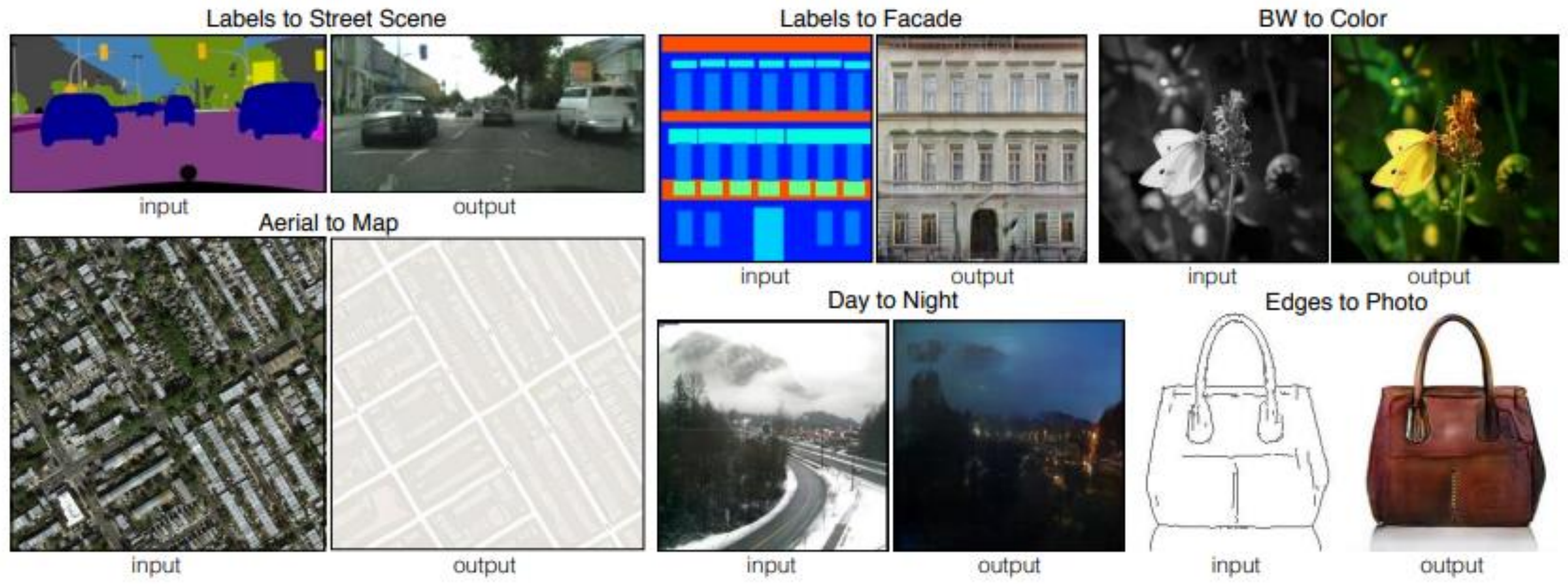


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

#05 GAN

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (CycleGAN) [ICCV 2017]

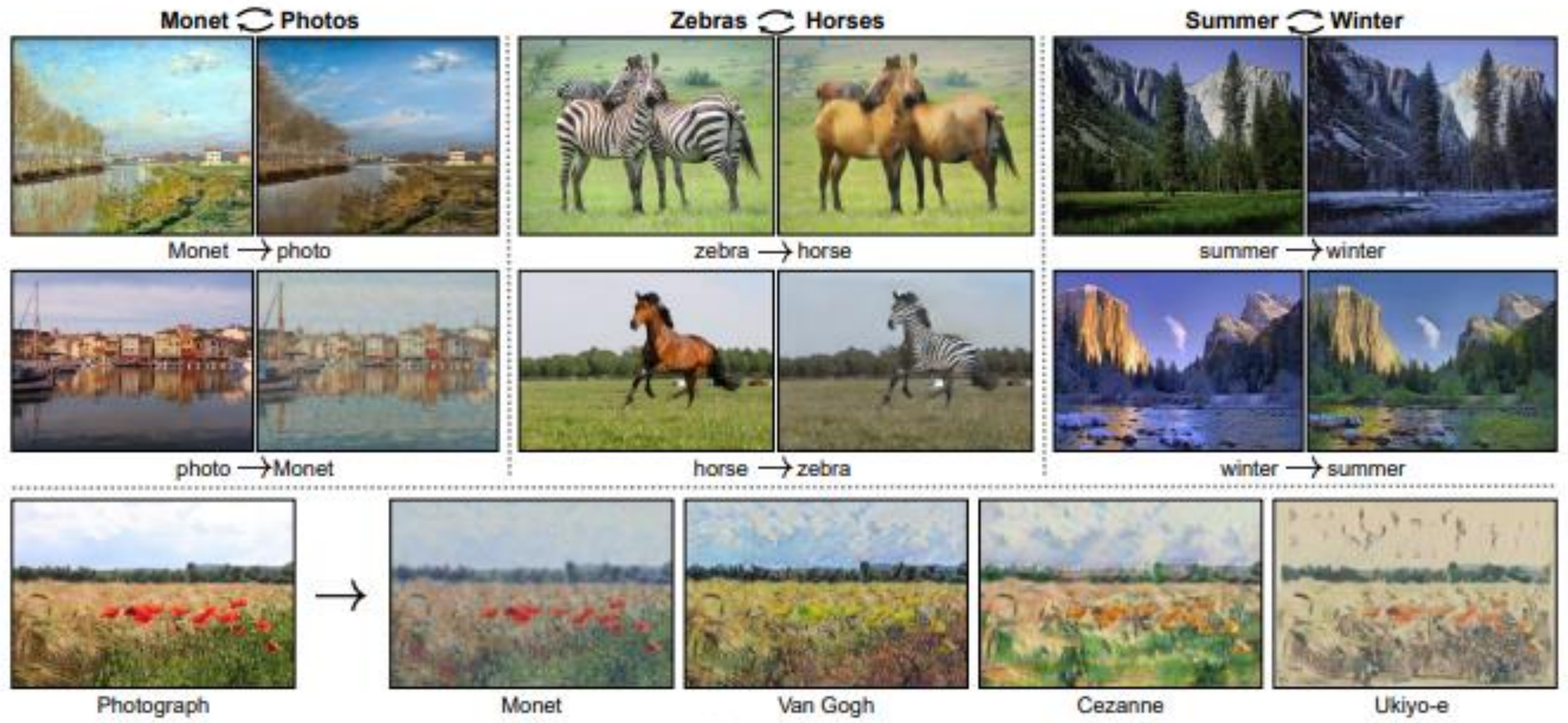


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

#05 GAN

Semantic Image Synthesis with Spatially-Adaptive Normalization (SPADE, GauGAN)
[CVPR 2019]

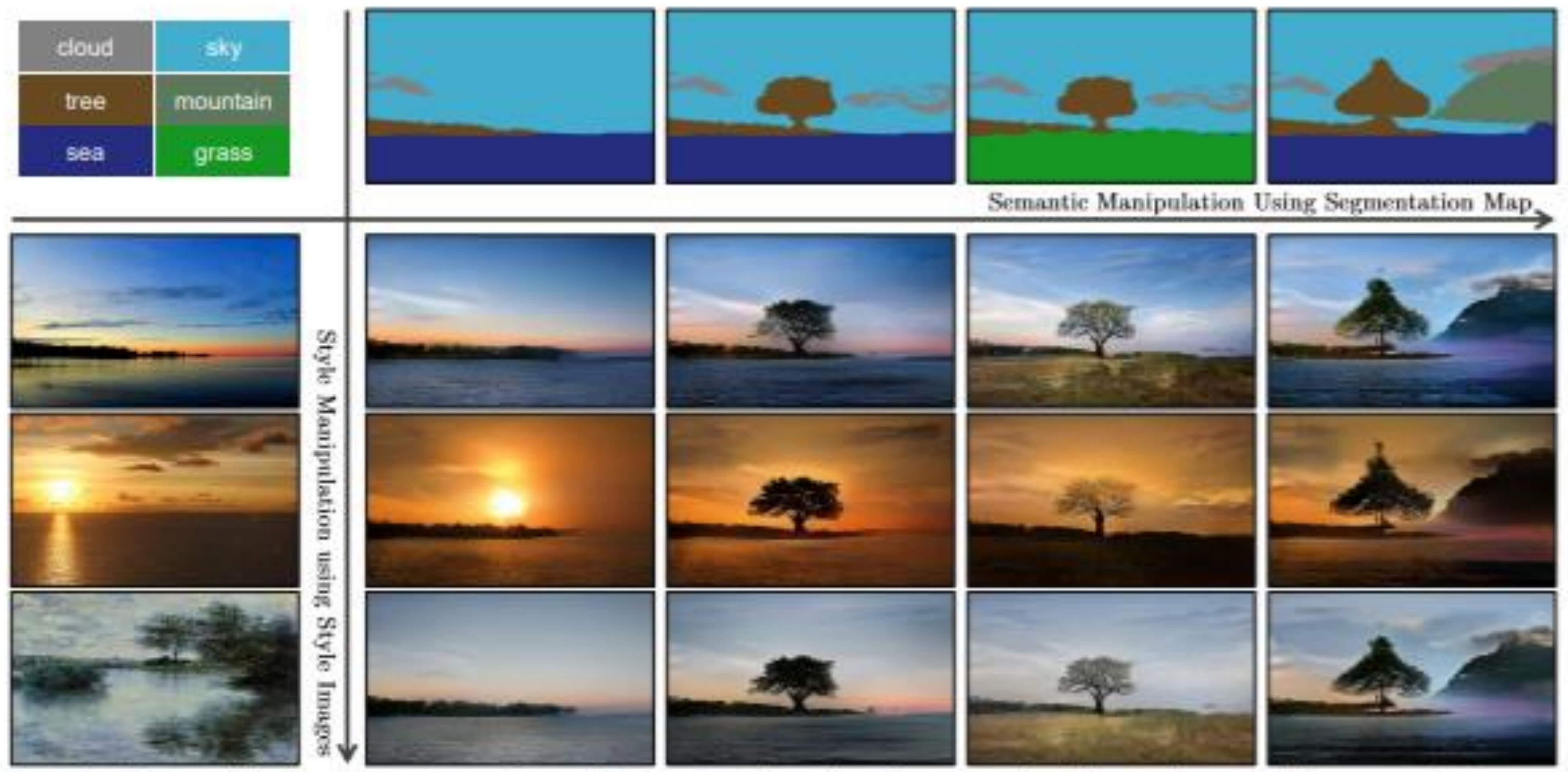

cloud
sky
tree
mountain
sea
grass
Semantic Manipulation Using Segmentation Map
Style Manipulation using Style Images

Figure 1: Our model allows user control over both semantic and style as synthesizing an image. The semantic (e.g., the existence of a tree) is controlled via a label map (the top row), while the style is controlled via the reference style image (the leftmost column). Please visit our website for interactive image synthesis demos.

#05 GAN

StarGAN:
Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation
[CVPR 2018]

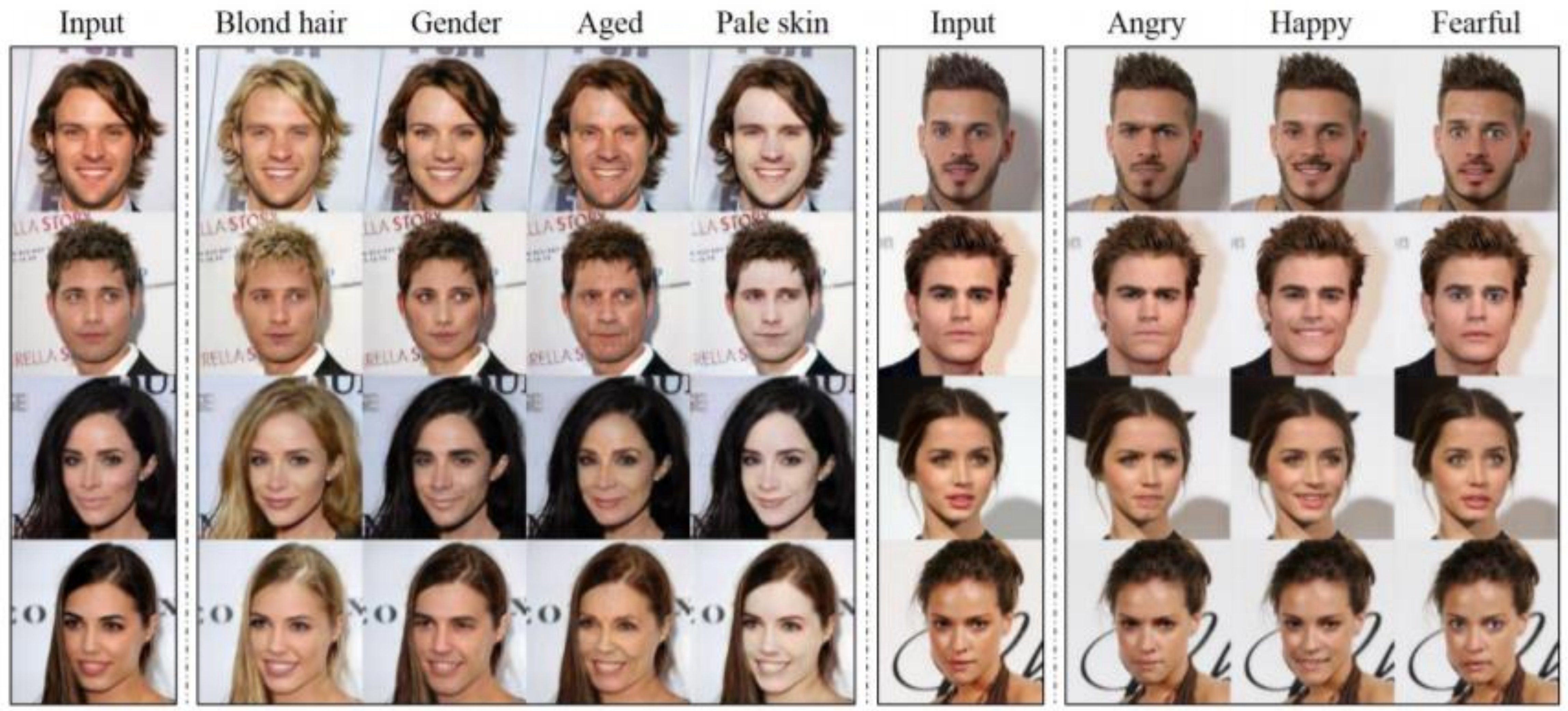
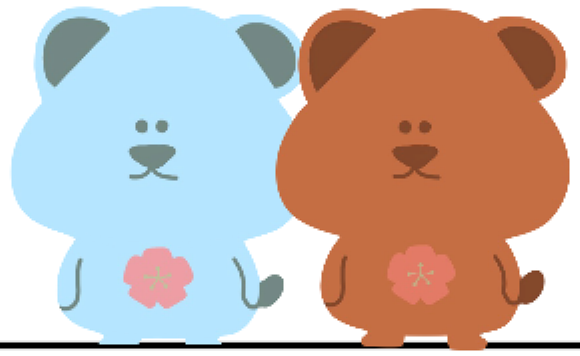


Figure 1. Multi-domain image-to-image translation results on the CelebA dataset via transferring knowledge learned from the RaFD dataset. The first and sixth columns show input images while the remaining columns are images generated by StarGAN. Note that the images are generated by a single generator network, and facial expression labels such as angry, happy, and fearful are from RaFD, not CelebA.

THANK YOU



EWHA EURON