KUBIG 24-WINTER

COMPUTER VISION STUDY

Generative Models

분반장 : 17기 문성빈 임청수

Contents

1.Generative Models

01. Generative Models

■ Generative AI의 시대

- 생성 인공지능(Generative AI)의 발전은 매우 빠르게 진행 중
- 이미지, 영상 데이터 뿐만 아니라 텍스트나 오디오 등 다양한 형태의 데이터를 고품질로 생성하는데 활용되고 있음

ex) 텍스트를 입력하여 원하는 영상이나 오디오 출력, 로봇 및 게임 캐릭터 모션 생성, 텍스트 기반 데이터 편집 등

STABLE DIFFUSION

ChatGPT

Dall-E 3

Dall-E 2

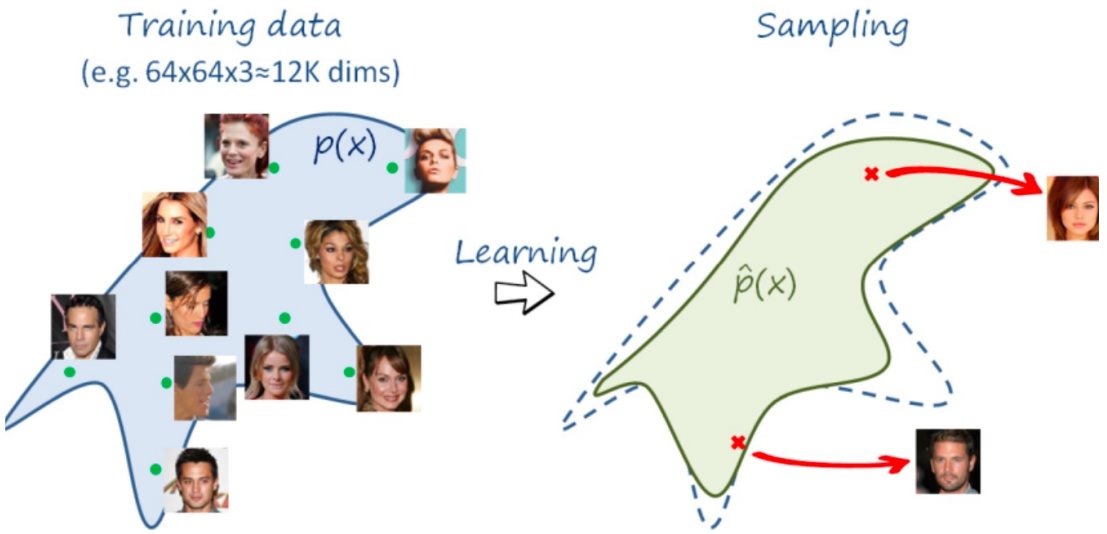
01. Generative Models

Generative Modeling

생성 : 통계학, 머신러닝 분야에서 생성은 학습과정에서 사용되지 않은 데이터를 데이터 공간 상의 확률분포 p_data에서 샘플링하는 것

생성모델의 목적 : 주어진 학습 데이터와 동일한 분포를 학습한 후 새로운 샘플을 생성하는 것

- 생성 모델이 출력한 값들의 분포와 실제 데이터가 인코딩된 분포가 매우 유사하여 통계적으로 구분하기 힘든 상태가 되도록 학습

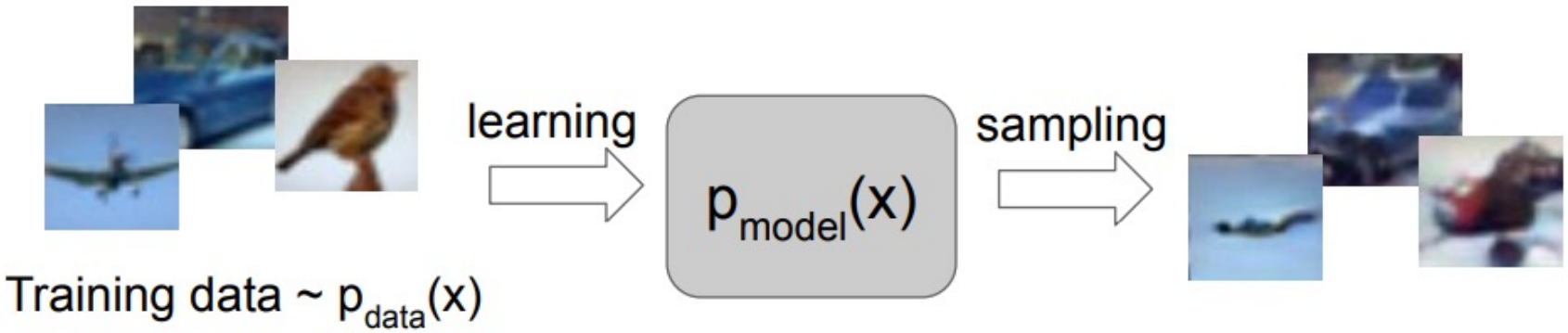


01. Generative Models

Generative Modeling

Explicit density estimation : P_model(x)을 명시적으로 정의함(PixelCNN, VAE 등)

Implicit density estimation : P_model(x) 을 명시적으로 정의하지 않고, sampling만 가능한 모델 학습(GAN 등)



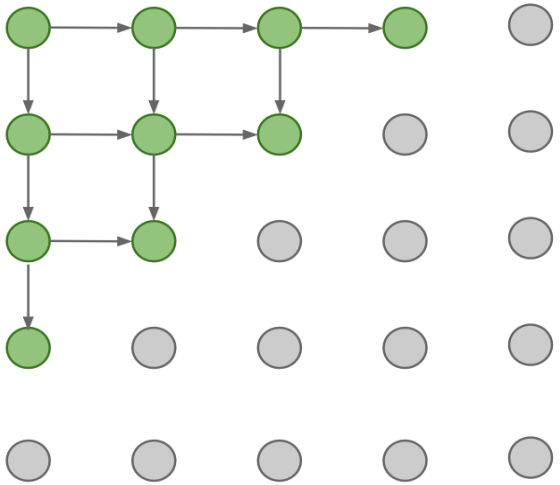
2. PixelRNN and PixelCNN

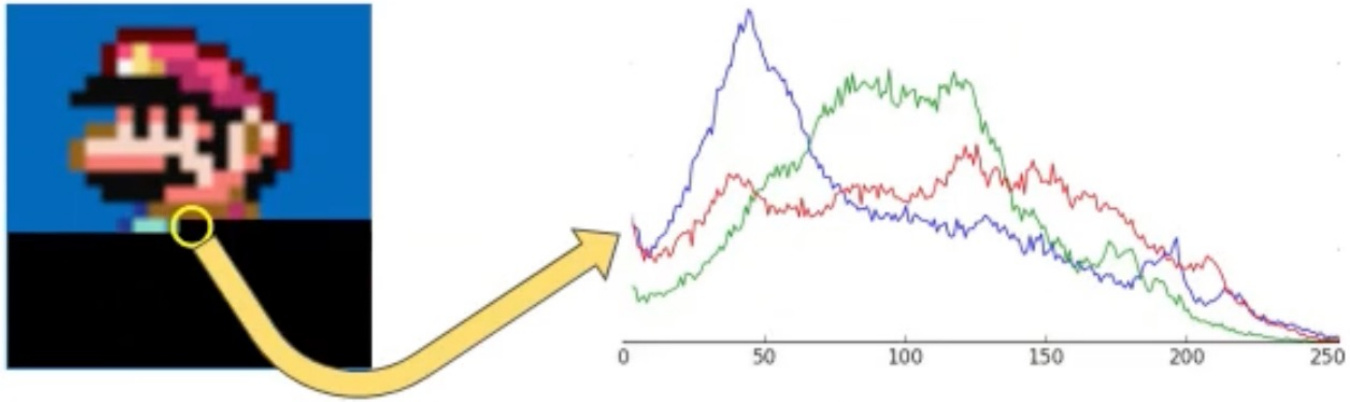
02. PixelRNN and PixelCNN

■ Fully Visible Belief Network

- 이미지가 주어졌을 때 이미지를 포함하는 확률분포 p(x)을 직접 구하는 방법
- 전체 이미지에 대한 확률분포는 매우 복잡하기 때문에 픽셀 단위로 확률분포를 계산하여 이미지 생성

1. 이미지 내의 픽셀들의 순서를 결정
2. 앞에서 생성한 픽셀을 조건부로 하여 다음 픽셀이 무슨 색인지 확률분포를 모델링함(분포 예측)
3. 분포에서 샘플링해서 다음 순서의 픽셀 값 생성



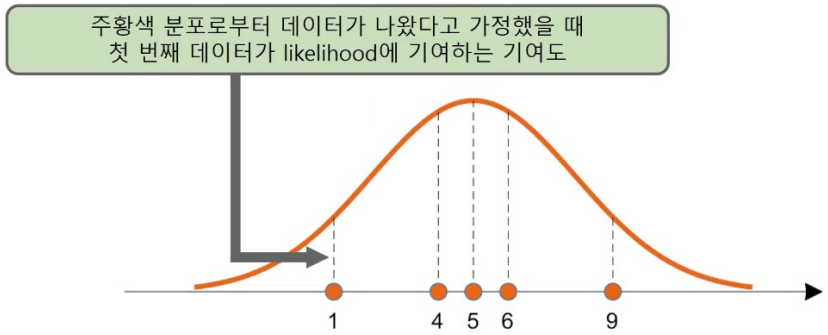


02. PixelRNN and PixelCNN

■ Explicit density estimation

Likelihood : 입력 데이터가 이 확률분포에서 나왔을 가능도.(=y값)

Chain rule : joint distribution을 product rule로 표현



Product rule (chain rule)

$$P(X_1, \ldots, X_n) = P(X_1)P(X_2 \mid X_1) \ldots P(X_n \mid X_1, \ldots, X_{n-1})$$

$$p(x) = p(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

Likelihood of image x

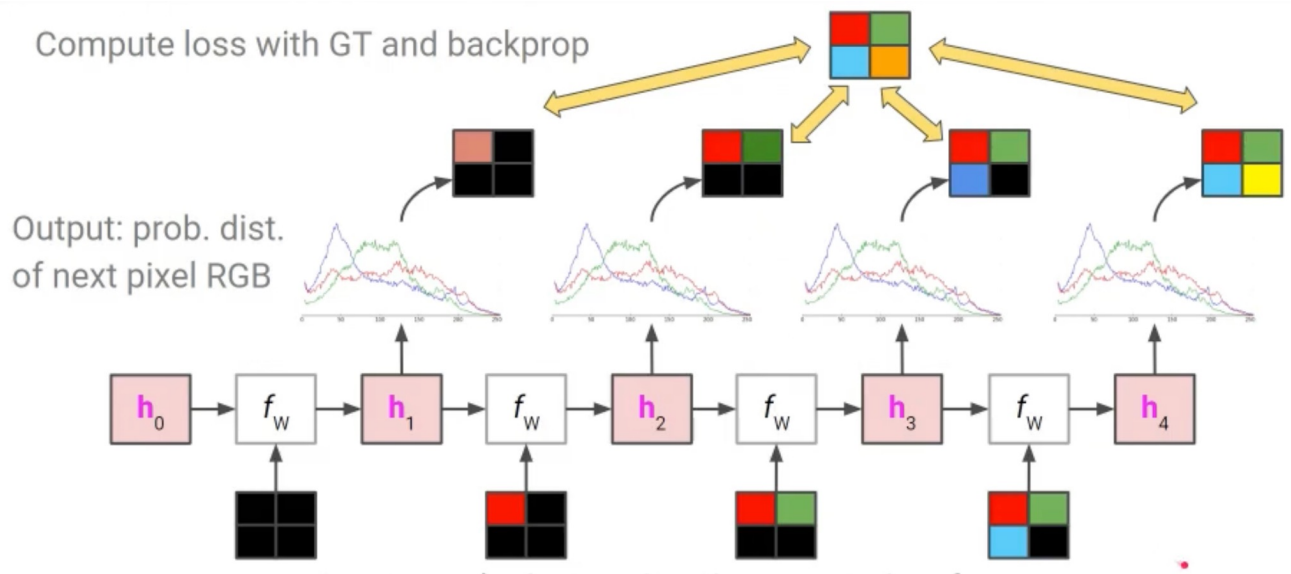
Joint likelihood of each pixel in the image

Probability of i'th pixel value given all previous pixels

02. PixelRNN and PixelCNN

PixelRNN

1. hidden_0 랜덤 초기화
2. 학습 데이터와 hidden_0로 hidden_1 추출
3. Hidden_1에서 다음 픽셀에 대한 확률분포 모델링 후 샘플 추출
4. 생성한 픽셀 값과 실제 데이터와의 Loss 계산
5. 실제 값을 입력하고 다음 hidden_state 계산



02. PixelRNN and PixelCNN

■ Row LSTM vs Diagonal BiLSTM

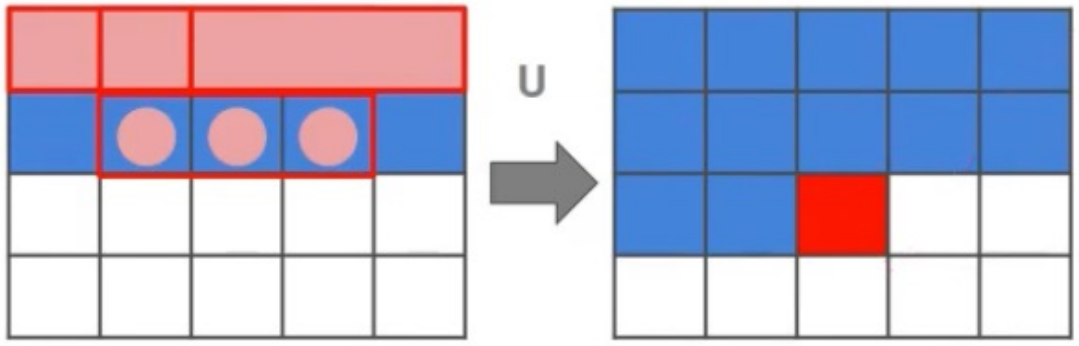
: 생성 순서에 따른 모델 구분

Row LSTM : 좌에서 우 방향으로 픽셀 생성.

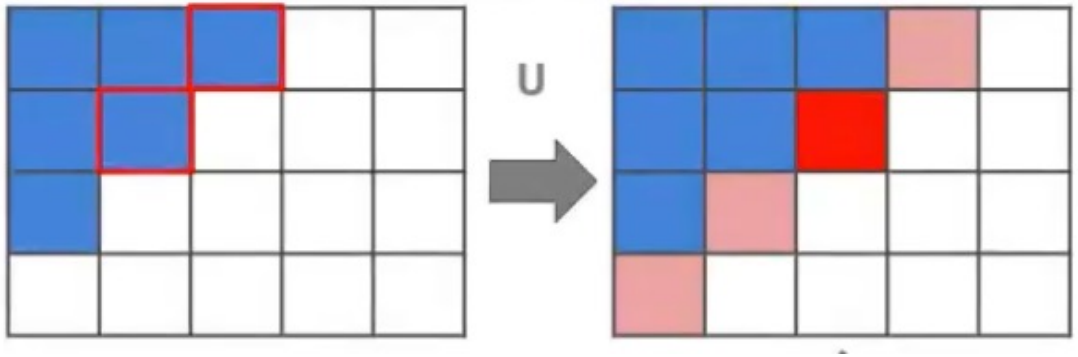
- 수용영역(Receptive field)이 삼각형 모양이 되므로 전에 생성된 픽셀들을 모두 반영하기 어려움

Diagonal BiLSTM : 좌상단에서 대각선 방향으로 픽셀 생성. 완료되면 우상단에서 동일하게 대각선 방향으로 픽셀 생성.

- 대각선은 직전 대각선 정보를 모두 포함하고 있음.



Row LSTM

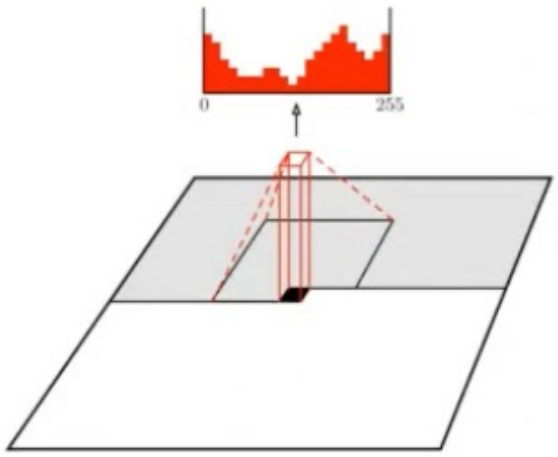


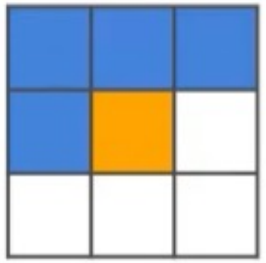
Diagonal BiLSTM

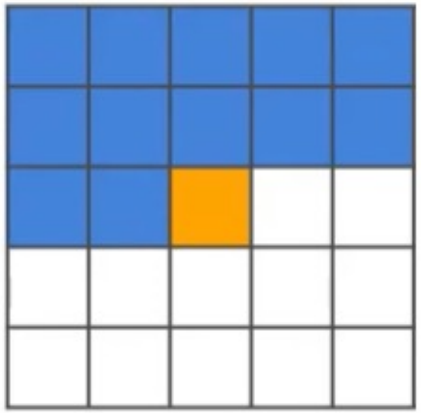
02. PixelRNN and PixelCNN

■ PixelCNN

- PixelRNN 모델은 sequential하게 학습되기 때문에 시간이 많이 소요됨
- PixelCNN은 근처에 픽셀만 고려하여 새로운 픽셀을 생성하는 모델
- 학습 시 모든 데이터 값을 알고 있기 때문에 병렬적으로 학습가능하다는 장점이 있음
- Mask를 통해 아직 생성되지 않은 픽셀을 못보도록 가리고 학습 진행

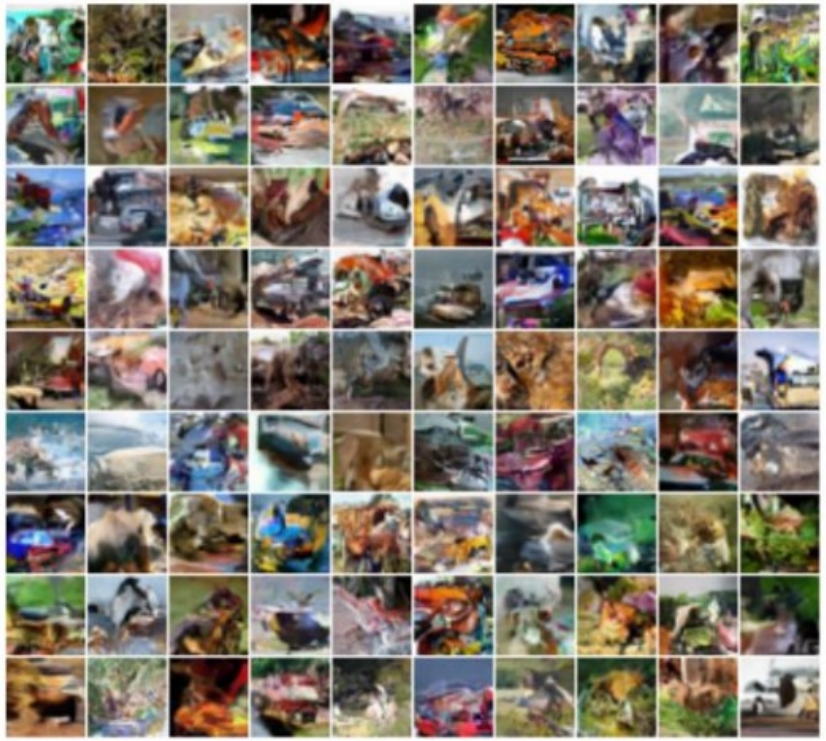




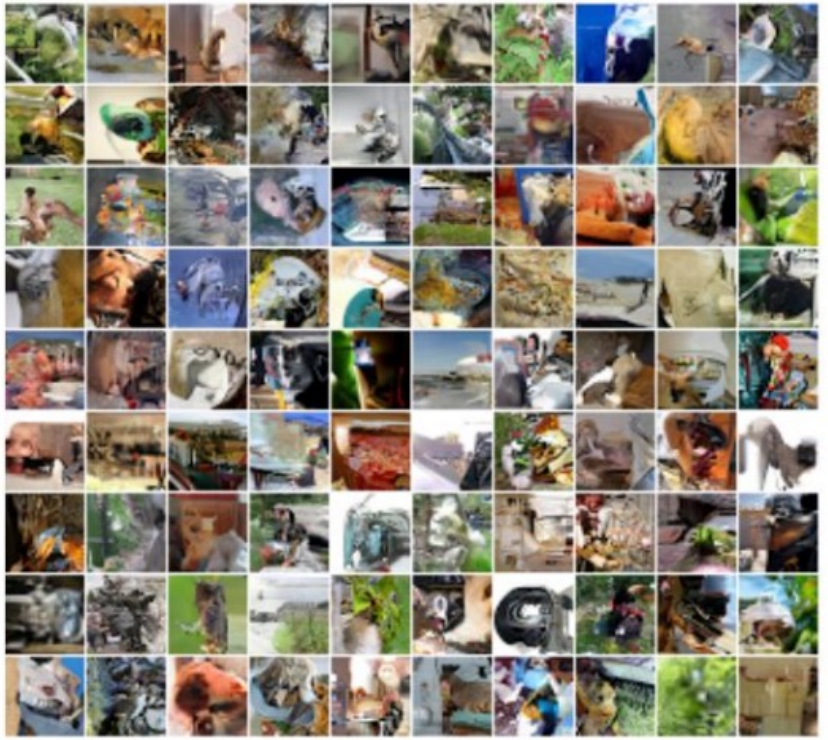


02. PixelRNN and PixelCNN

PixelCNN



32x32 CIFAR-10



32x32 ImageNet

02. PixelRNN and PixelCNN

■ PixelRNN and PixelCNN 정리

요약 : 픽셀 단위로 확률분포를 계산하여 chain rule을 통해 전체 확률분포를 명시적으로 구하는 모델

RowLSTM : 수직 순서로 픽셀을 생성하고 생성 시 수용영역이 삼각형으로 제한됨

Diagonal BiLSTM : 대각선 순서로 픽셀을 생성하고 직전에 생성한 모든 픽셀을 수용할 수 있음

PixelCNN : sequential하게 생성하지 않고 주변 픽셀로만 생성하면서 병렬적 학습이 가능하여 속도가 빨라지고 성능도 개선됨

- 장점

1. 명시적인 이미지의 확률분포인 p(x) 계산가능
2. 상대적으로 최적화하기 쉬움

- 단점

1. Inference 단계에서 이미지 생성 시 픽셀들을 순차적으로 생성해야 해서 매우 느림
2. 초기 픽셀이 이후 생성되는 모든 픽셀에 영향을 주기 때문에 초기 픽셀 값 선정에 매우 민감해짐
3. 생성된 이미지가 의미론적 관점에서 정보를 반영하지 못함(vs VAE, GAN)

3. Variational Autoencoders

03. Variational Autoencoders

■ What is VAE?

PixelCNN의 p(x)는 n개의 계산 가능한 픽셀 확률분포에 대하여 다음과 같이 정의된다.

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

Variational Autoencoders (VAEs)는 latent variable z에 대하여 다음과 같이 정의된다.

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

위 식은 모든 z에 대해 적분할 수 없기 때문에 계산이 불가능하다.

따라서 p(x)를 직접 최적화할 수 없으며 그 대신 p(x)의 lower bound를 구해서 최적화해야 한다.

1) z가 무엇인지 2) lower bound 최적화는 어떻게 진행하는지 알아보자

03. Variational Autoencoders

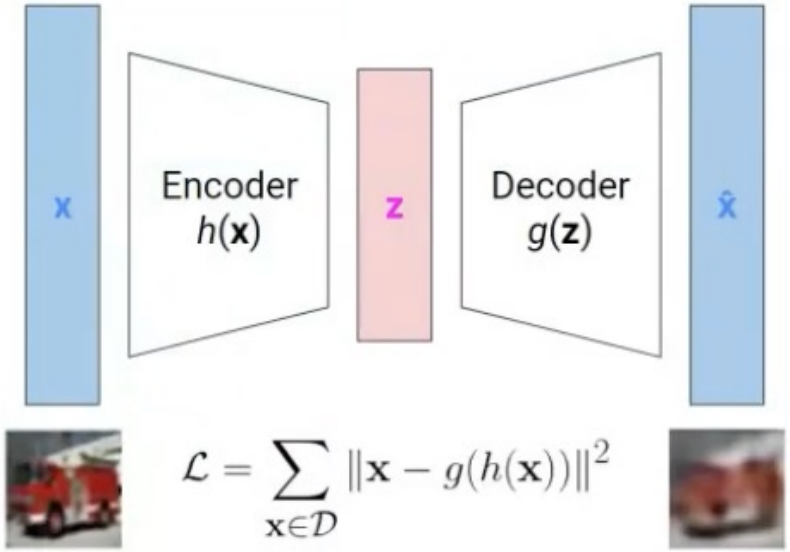
■ Background : Autoencoders

AE : 비지도학습으로 feature representation을 학습할 수 있는 모델이며 생성모델은 아님

AE는 z를 잘 추출하는 인코더를 학습하기 위해 디코더를 사용하므로 학습이 끝나면 디코더는 사용하지 않음

Loss는 입력 데이터와 출력 데이터를 비교하여 계산

1) 입력 데이터 x를 인코더에 통과시켜 차원이 축소된 잠재 벡터인 z 출력
2) z를 다시 원래 데이터 x로 복원시키는 디코더에 통과시켜 x_hat 출력



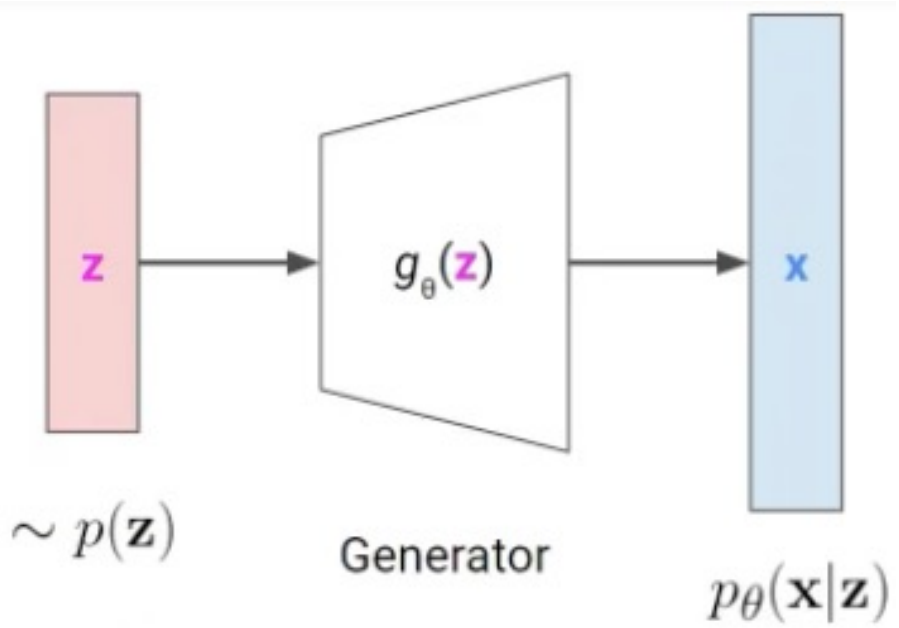
03. Variational Autoencoders

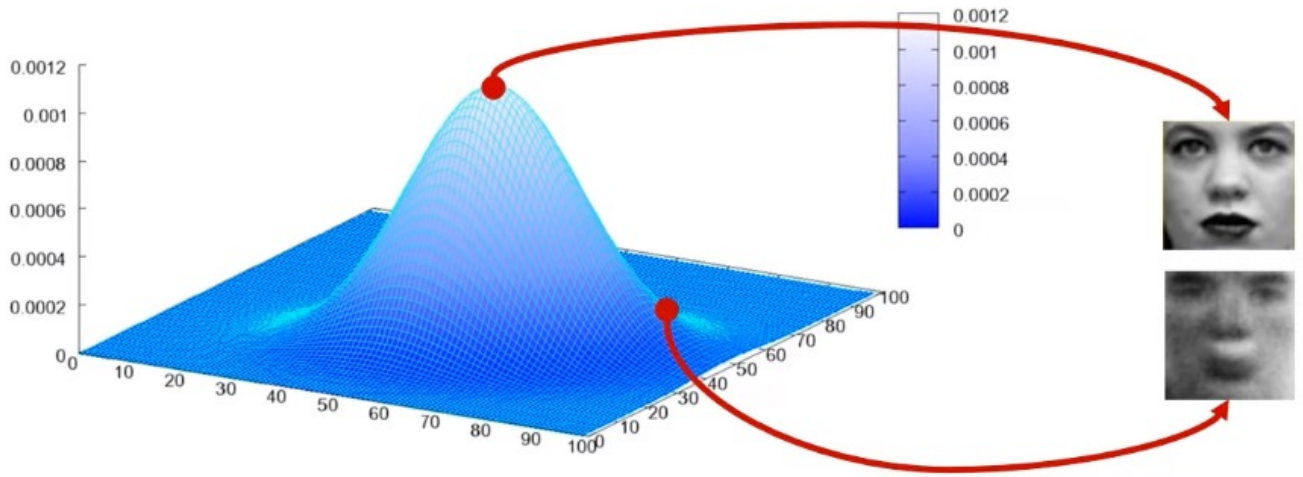
■ VAEs

AE에서 디코더만 사용하여 random z값을 입력하면 생성이 잘 되지 않음

➔ 생성이 잘 되기 위해서는 z가 원본 이미지를 잘 설명할 수 있는 확률분포로 가정해야 한다.

따라서 VAE는 z가 학습 이미지를 샘플링할 수 있는 확률분포를 모사하도록 하는 것이 목표





03. Variational Autoencoders

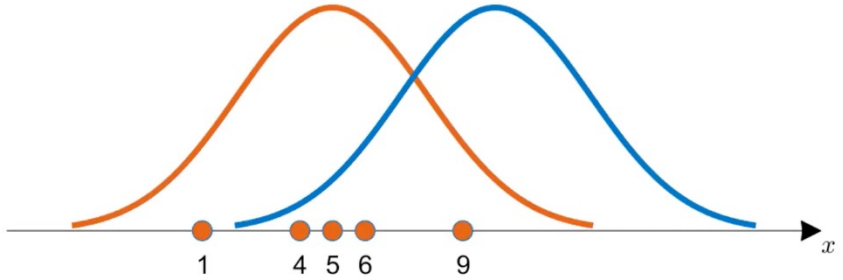
■ VAE : Intractability

학습 데이터의 likelihood를 maximize하는 확률분포를 찾자

즉, 학습 데이터 x가 나올 확률이 가장 커지는 확률분포를 찾고 싶다

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

Joint probability : $p(x,z) = p(x|z) * p(z)$



maximize likelihood estimation

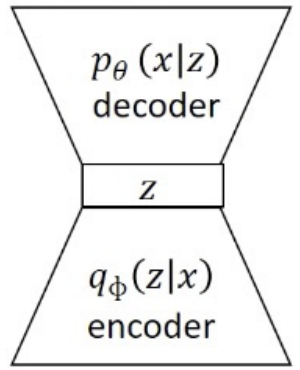
Intractible to compute $p(x|z)$ for every z!

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

Simple Gaussian prior

Decoder neural network

Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$



p(z|x)를 q(z|x)의 네트워크로 근사하여 계산하는 것으로 가정
P(x)는 계산 불가능하므로 lower bound를 통해 maximize 할 수 있는지 확인

03. Variational Autoencoders

■ VAE loss function

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

Taking expectation wrt. z (using encoder network) will come in handy later

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)}|z)p_\theta(z)}{p_\theta(z|x^{(i)})}\right]$$

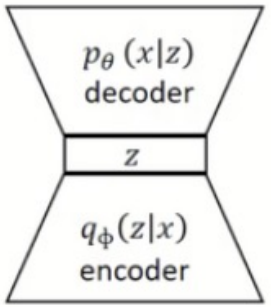
(Bayes' Rule)

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

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

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

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



참고: $E_{z\sim q_\phi(z|x^{(i)})}\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)}\right] = \int_z \log \frac{q_\phi(z|x^{(i)})}{p_\theta(z)} q_\phi(z|x^{(i)})dz$

$$KL(P||Q) = \sum_x P(x)\log\frac{P(x)}{Q(x)}$$

03. Variational Autoencoders

■ VAE loss function

첫번째 항 : 모든 학습 데이터에서 출력된 z에 대하여 디코더의 likelihood를 최대화

두번째 항 : 실제 z의 분포와 인코더 q가 예측한 분포가 유사하도록 KL Divergence를 최소화

실제 z의 분포는 가우시안 분포로 가정

세번째 항 : 인코더를 근사한 q와 true distribution인 p의 분포를 비교하는데 p는 계산 불가

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

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling.

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

03. Variational Autoencoders

VAE loss function

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

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

← Evidence lower bound (ELBO)

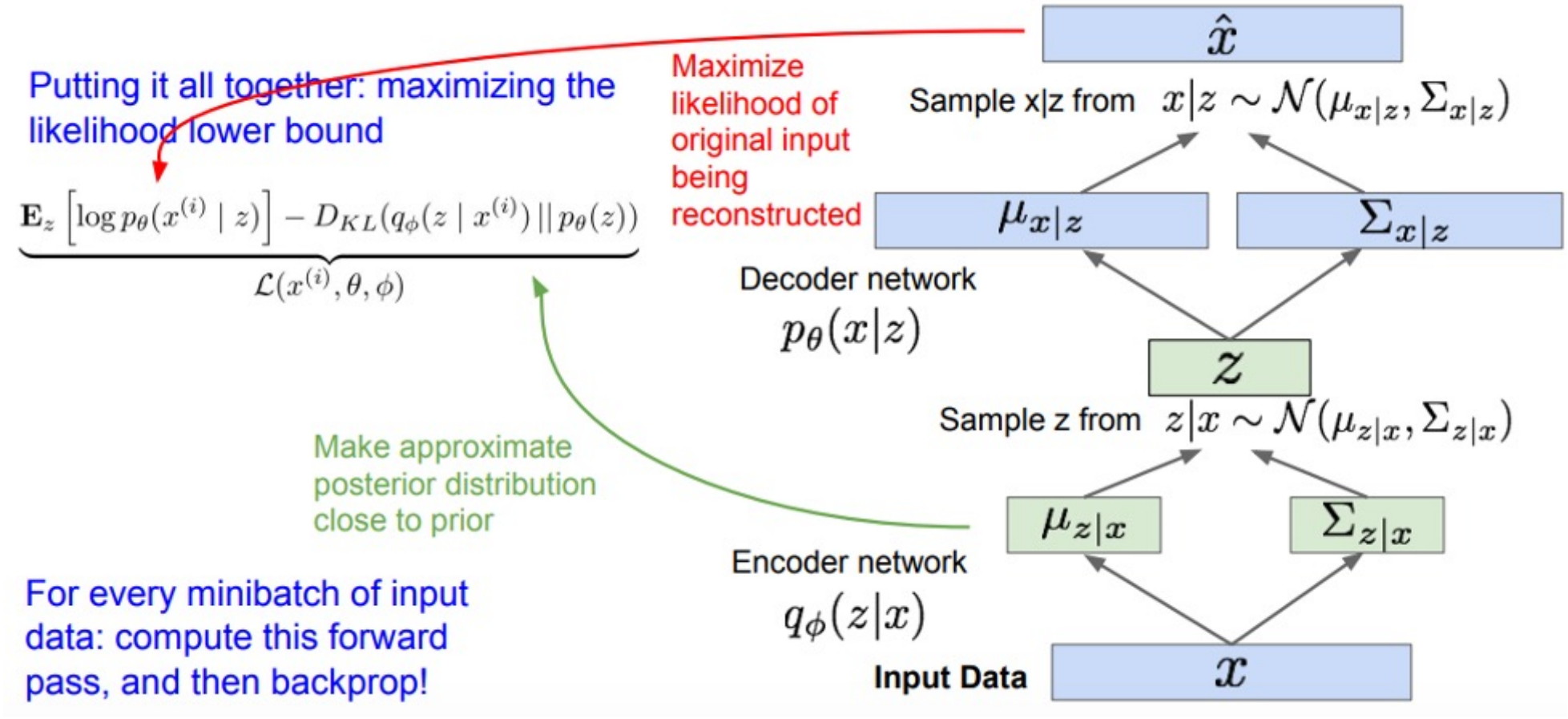
Reconstruction error　　Regularization error

$$arg \min_{\theta,\phi} \sum_i -\mathbb{E}_{q_\phi(z|x_i)}\left[log\left(p(x_i|g_\theta(z))\right)\right] + KL(q_\phi(z|x_i)||p(z))$$

Reconstruction Error　　Regularization

03. Variational Autoencoders

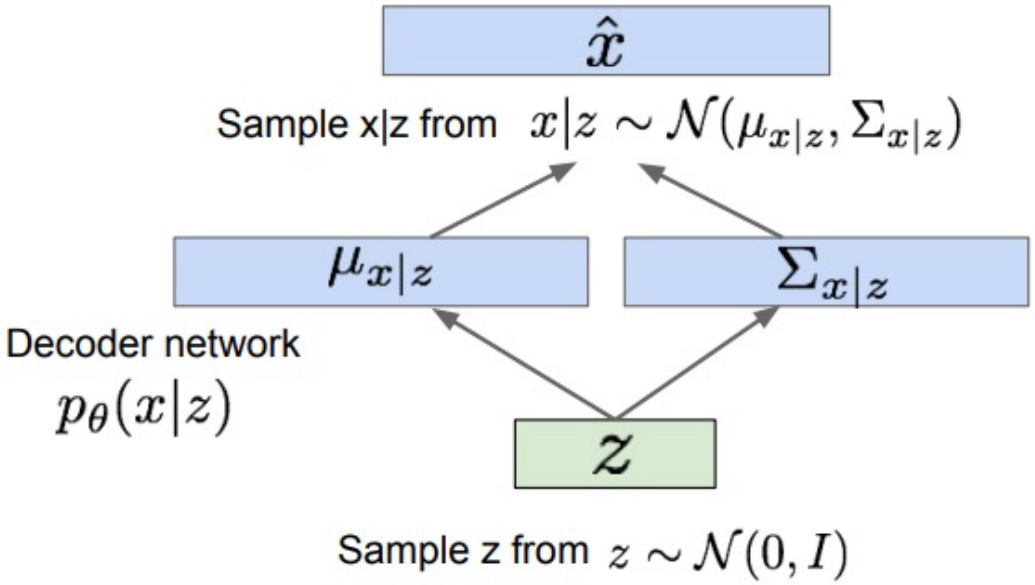
VAE loss function

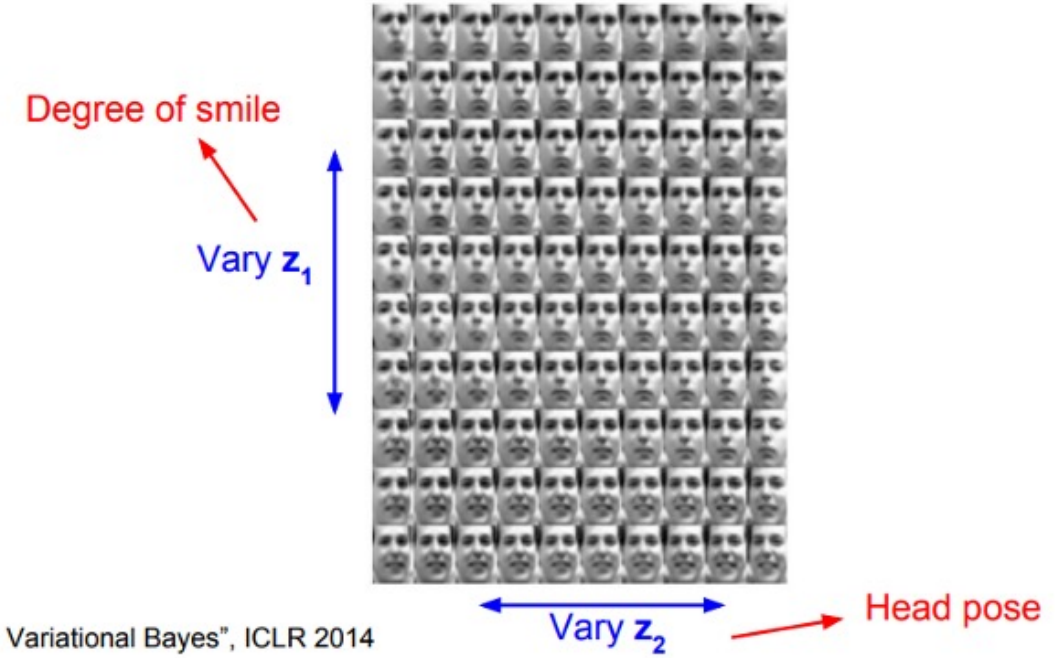


03. Variational Autoencoders

Generating data

학습 단계에서는 z를 p(z|x) 분포에서 샘플링했는데 생성 단계에서는 z를 가우시안 분포에서 샘플링 진행





03. Variational Autoencoders

■ VAE 요약

데이터를 생성하기 위하여 AE에 분포와 샘플링 개념을 추가하고 확률분포의 근사적 최적화 진행

장점

1. 수학적으로 잘 설명 가능한 모델
2. 의미론적 해석 가능

단점

1. GAN에 비해 블러가 많고 퀄리티가 떨어짐(가우시안 분포가 아닐 수도 있음)
2. PixelRNN/CNN 같이 직접적으로 density를 구한 모델보다는 성능이 떨어짐

4. Generative Adversarial Network (GAN)

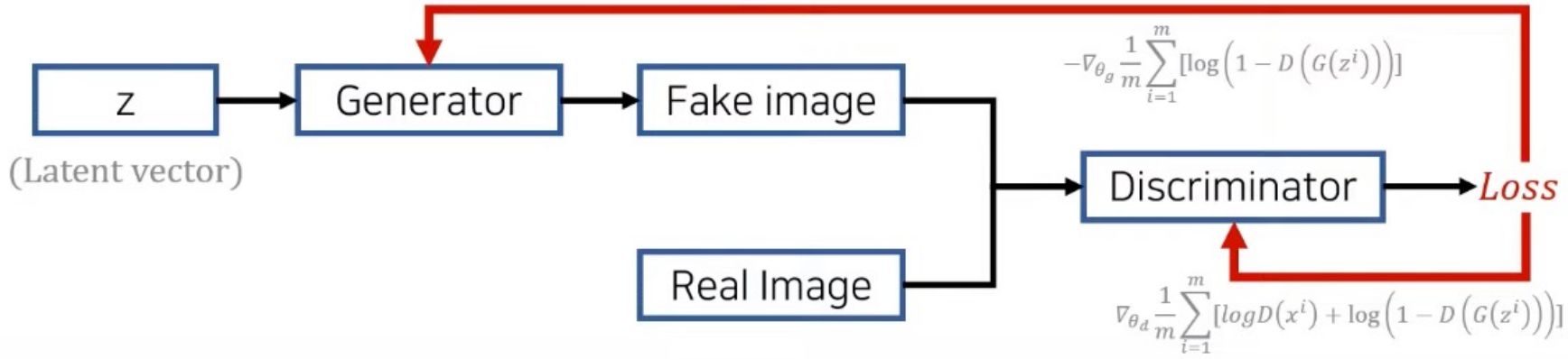
04. Generative Adversarial Network

GAN

: 생성자(Generator)와 판별자(Discriminator) 두 개의 네트워크를 활용한 생성모델

PixelCNN과 VAE는 확률분포를 직접 모델링하지만 GAN은 직접 모델링하지 않음

두 네트워크의 게임이론을 통해 복잡한 확률분포를 학습한 후 샘플링을 잘하고자 함

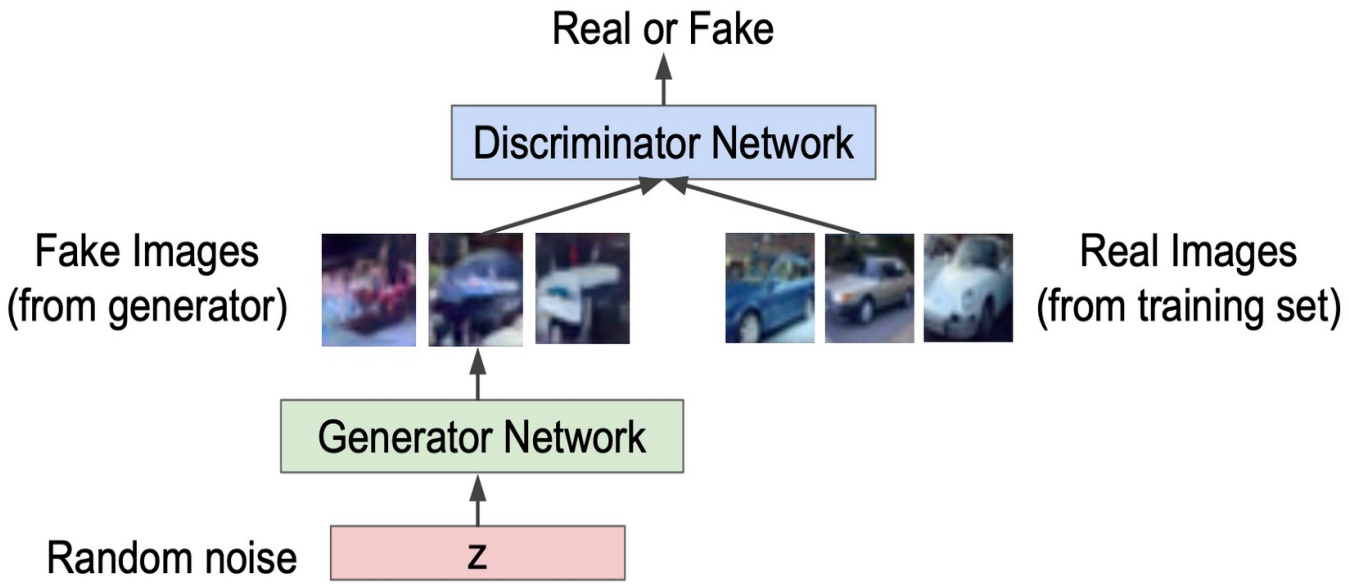


04. Generative Adversarial Network

■ Training GAN : Two-player game

Generator Network : real-looking images를 생성하여 discriminator 속이기

Discriminator Network : real images와 fake images 구분



Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

04. Generative Adversarial Network

■ Training GAN : Two-player game

- minimax objective function 정의

Theta_g는 목적함수를 최소화하는 것이 목표이고 theta_d는 목적함수를 최대화하는 것이 목표

Discriminator output : input data(fake or real)가 real image distribution P_data 에 속할 likelihood (0~1)

Generator output : new data instance

Goal of Discriminator : Maximize objective = D(x) is close to 1 (real) + D(G(z)) is close to 0 (fake)

Goal of Generator : Minimize objective = D(G(z)) is close to 1 (real) <- fooling discriminator

Minimax objective function:

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

04. Generative Adversarial Network

■ Training GAN : Two-player game

GAN은 Generator와 Discriminator를 번갈아가면서 학습

D는 maximize를 위하여 gradient ascent를 이용

G는 minimize를 위하여 gradient descent를 이용

Minimax objective function:

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

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

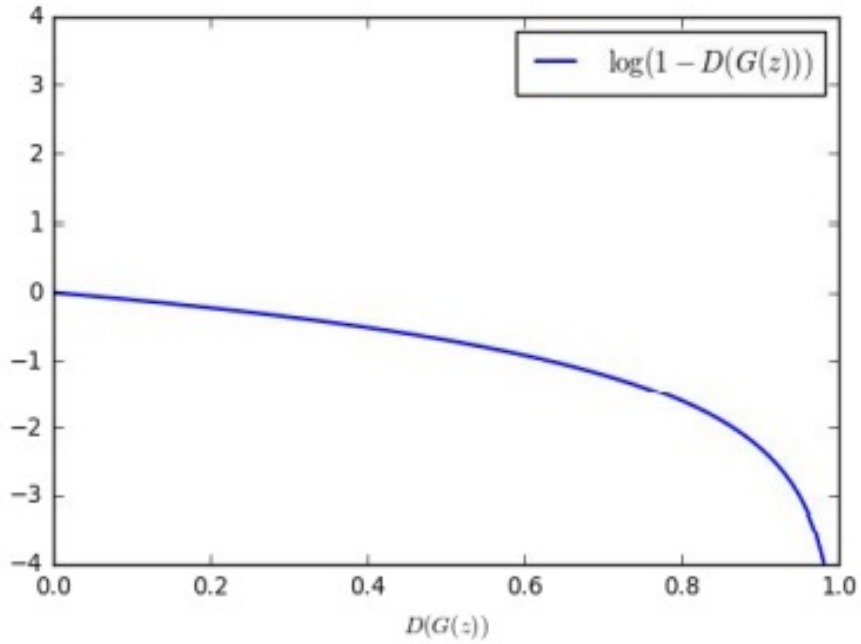
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

04. Generative Adversarial Network

■ Training GAN : Problem – optimizing G does not work well

Log (1-x) 그래프 특성상, x가 1에 가까울수록 기울기 급격히 증가, 반면 0에서는 flat에 가까움

Generated sample이 이미 좋은 상태일때 학습이 많이 이루어지고, 초기에는 학습이 잘 되지 않음



Loss landscape

2. **Gradient descent** on generator

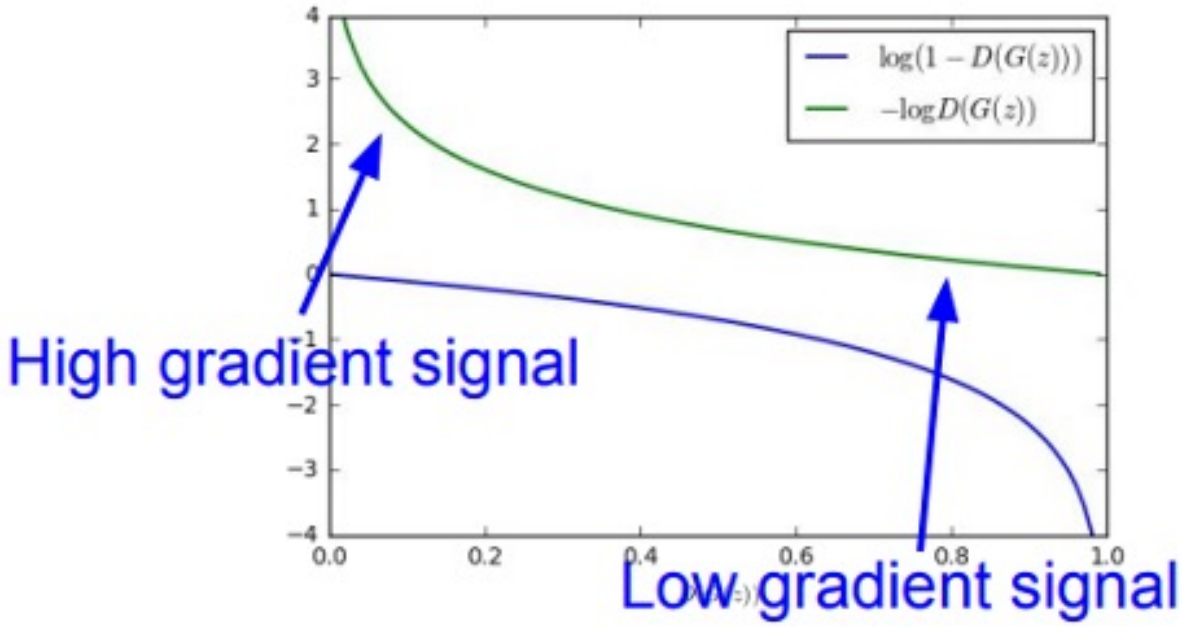
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

04. Generative Adversarial Network

■ Training GAN : Problem – optimizing G does not work well

Log graph를 뒤집는 trick 사용

초기 bad sample에서 higher gradient 얻을 수 있음!



Loss landscape

2. **Gradient descent** on generator

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

2. **Instead: Gradient ascent** on generator, different objective

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

04. Generative Adversarial Network

■ Training GAN : Algorithm

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

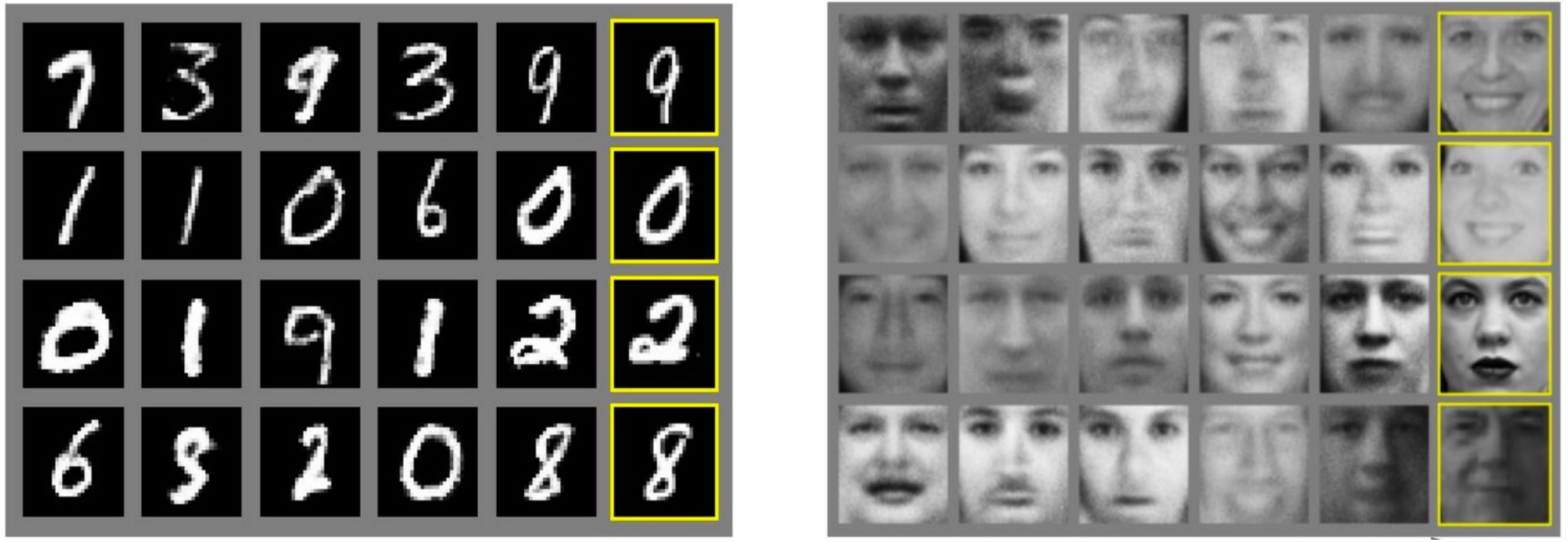
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

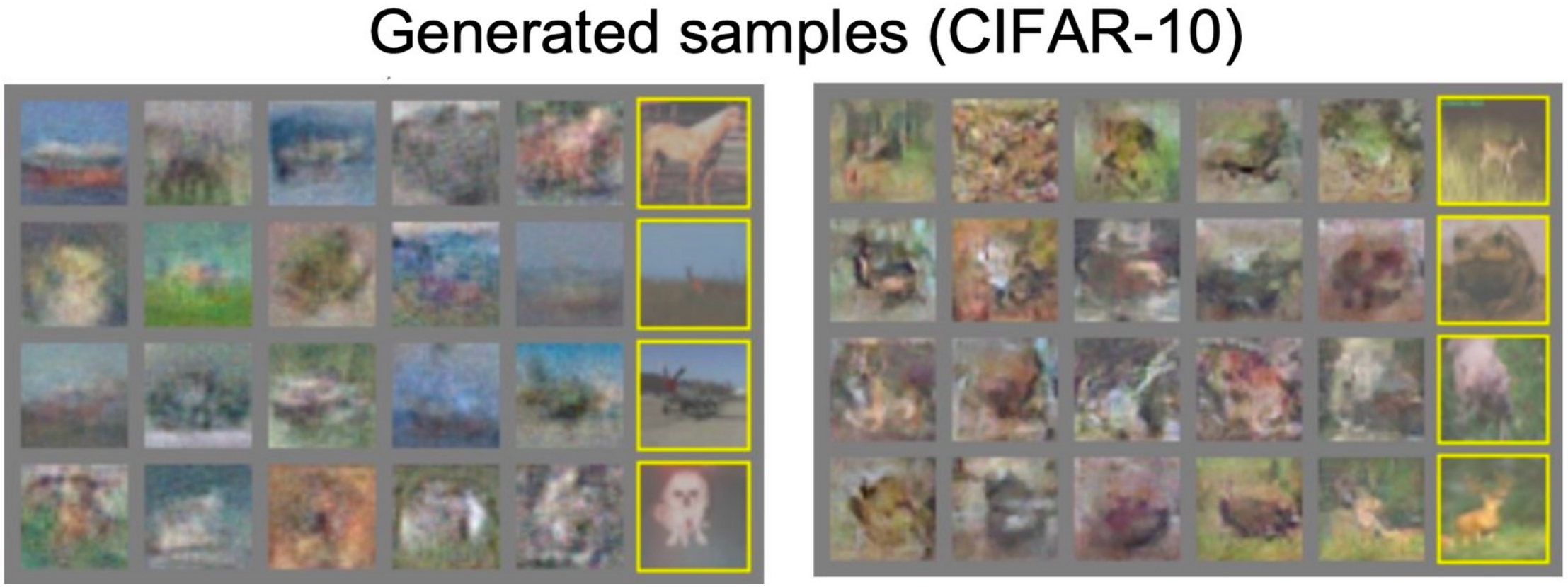
04. Generative Adversarial Network

Generated Samples



04. Generative Adversarial Network

Generated Samples

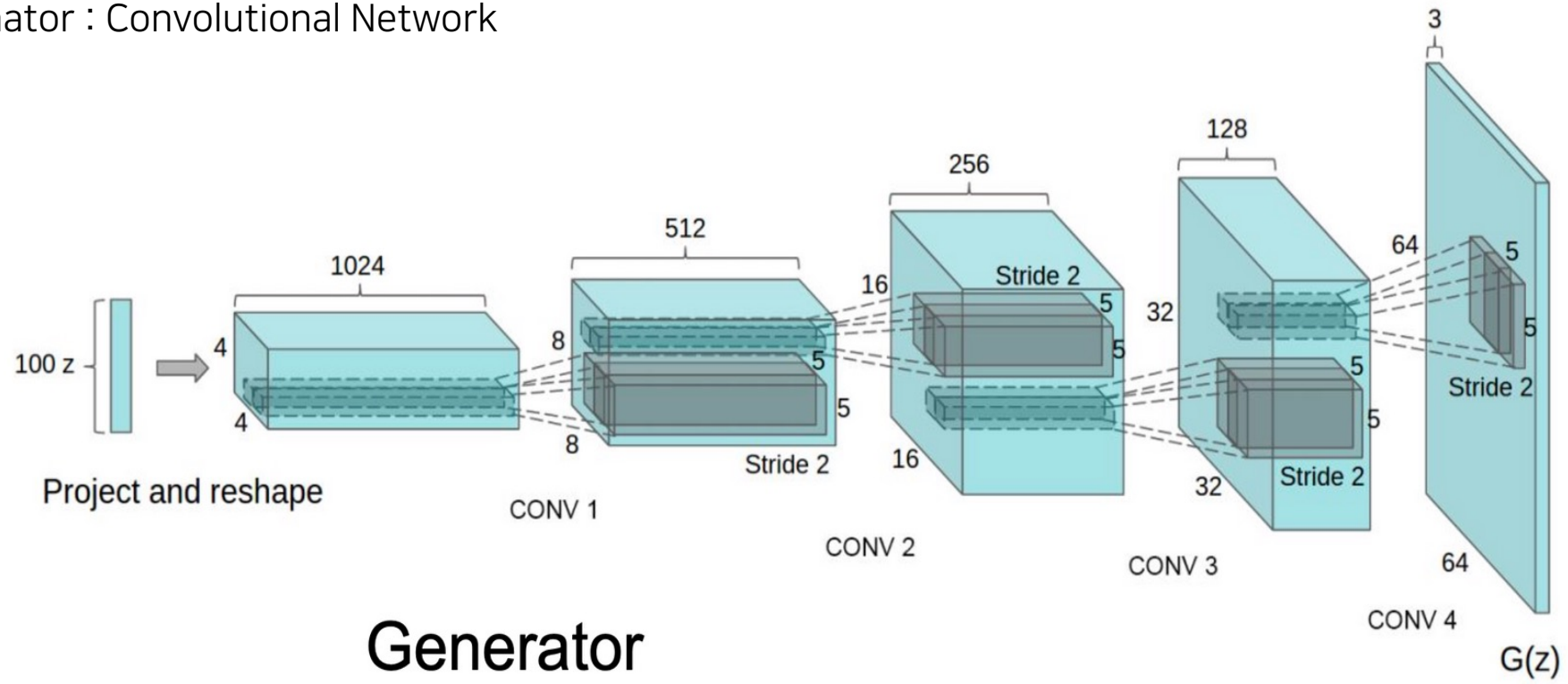


04. Generative Adversarial Network

■ Convolutional Architectures

Generator : Upsampling Network with Transposed convolutions

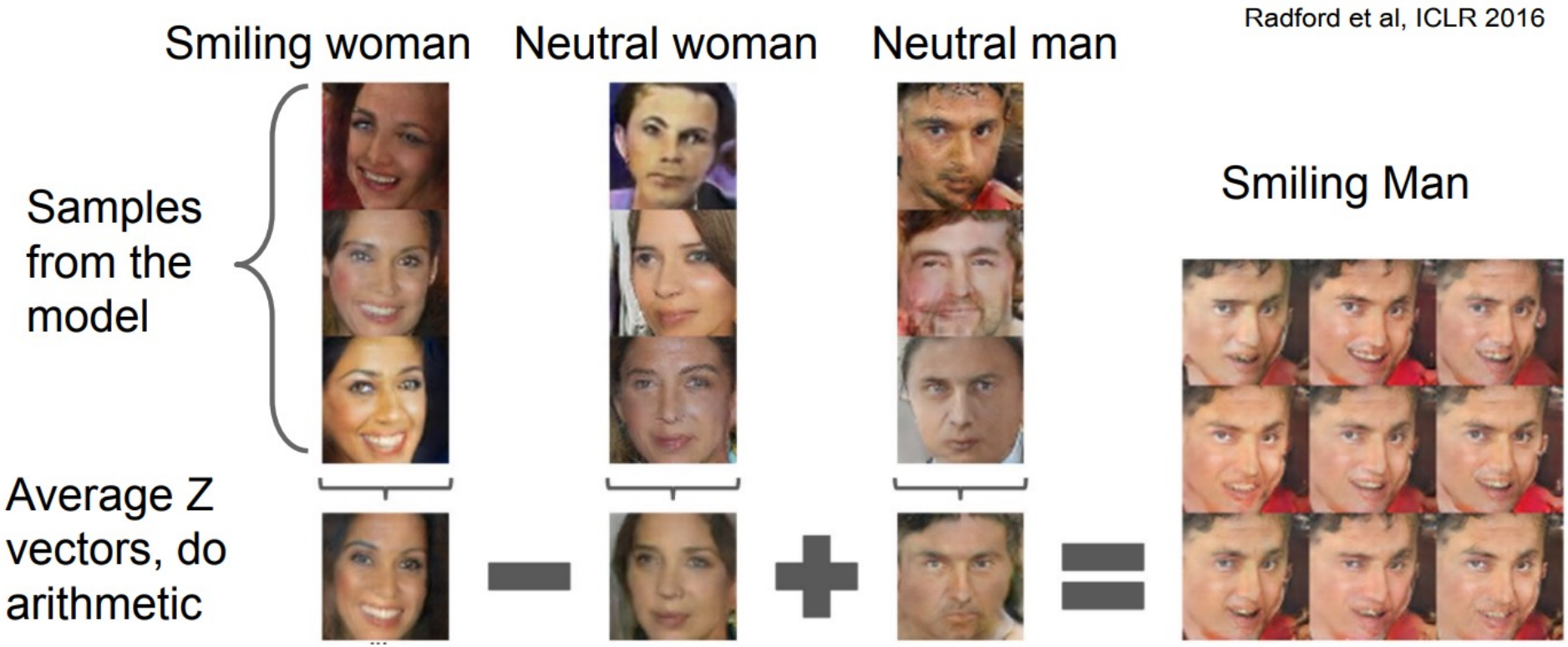
Discriminator : Convolutional Network



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

04. Generative Adversarial Network

Generative Adversarial Networks: Interpretable Vector Math



04. Generative Adversarial Network

■ GAN 요약

학습 데이터에 대한 특정 확률 분포를 정의하지 않고 샘플링을 통해 간접적으로 생성

Two player game이론을 통해 학습 데이터의 분포로부터 생성 모델을 학습

장점

1. 앞선 PixelCNN, VAE보다 좋은 성능
2. 정확한 density function을 몰라도 이미지 생성 가능

단점

1. 학습이 까다롭고 불안정
2. Likelihood에 대한 Objective function을 직접적으로 최적화하지 못함

5. Diffusion

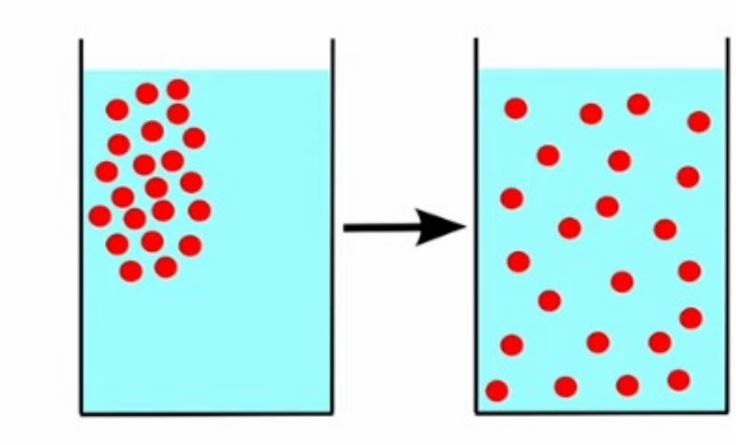
05. Diffusion

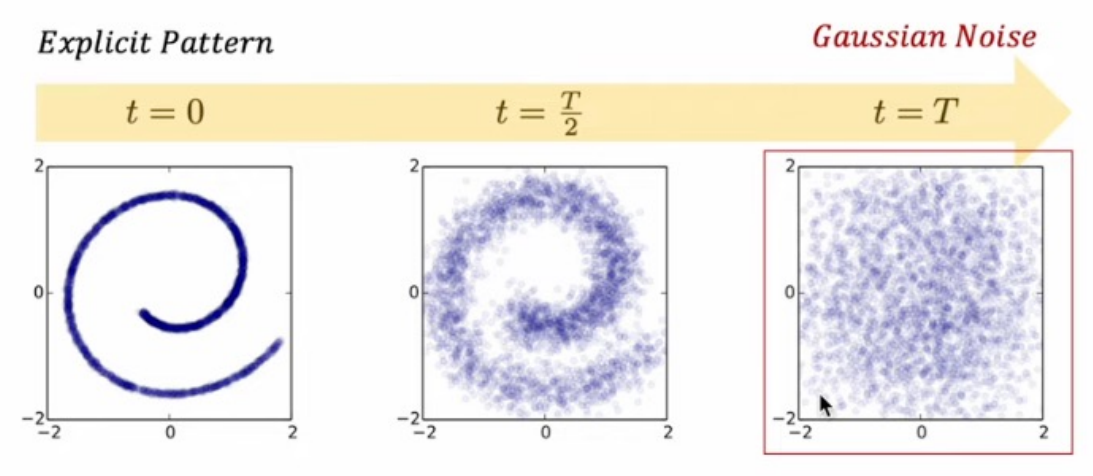
What is Diffusion?

Diffusion = 확산

물리학에서 'Diffusion'은특정한 물질(기체, 액체 등)이 조금씩 번지면서 농도로 바뀌는 현상

생성모델에서 'Diffusion'은 원래 데이터의 패턴이 diffusion 과정을 거쳐서 Gaussian Noise로 와해되는 현상을 의미함





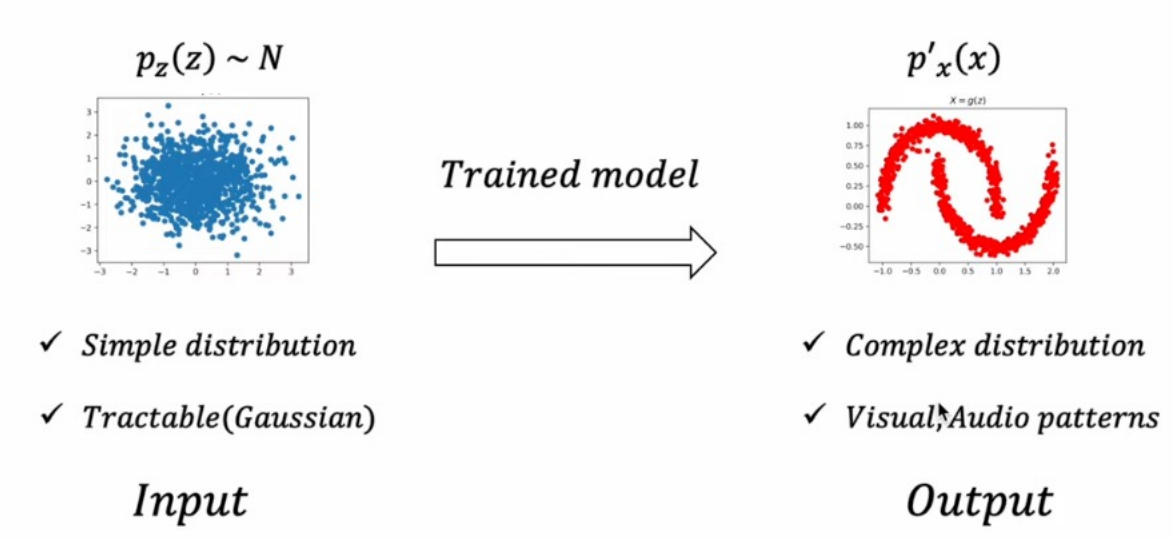
05. Diffusion

■ Generative model : Latent variable model

매우 간단한 분포에서 특정한 패턴을 갖는 분포로 변환(mapping, transformation, sampling)하는 것

VAE : 학습된 디코더를 통해 latent variable을 특정한 패턴의 분포로 mapping

GAN : 학습된 Generator를 통해 latent variable을 특정한 패턴의 분포로 mapping



05. Diffusion

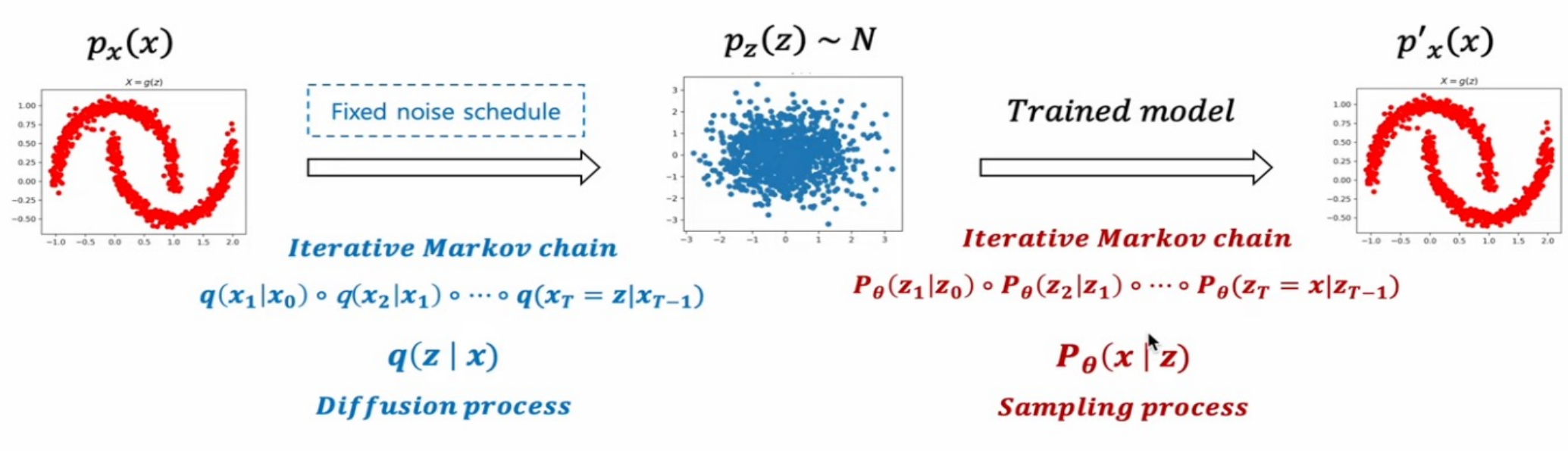
■ Generative model : Latent variable model

Diffusion : 학습된 Diffusion model의 조건부 확률분포 p(x|z)를 통해 특정한 패턴의 분포 획득

이러한 forward 과정에서 다른 모델과 달리 네트워크를 학습하지 않음.

왼쪽 과정을 학습과정으로 정의하지 않고 사전에 정의된 스케쥴에 따라 노이즈를 주입하는 과정으로 정의

두 과정 모두 조건부 확률 분포의 마코브 체인으로 구성

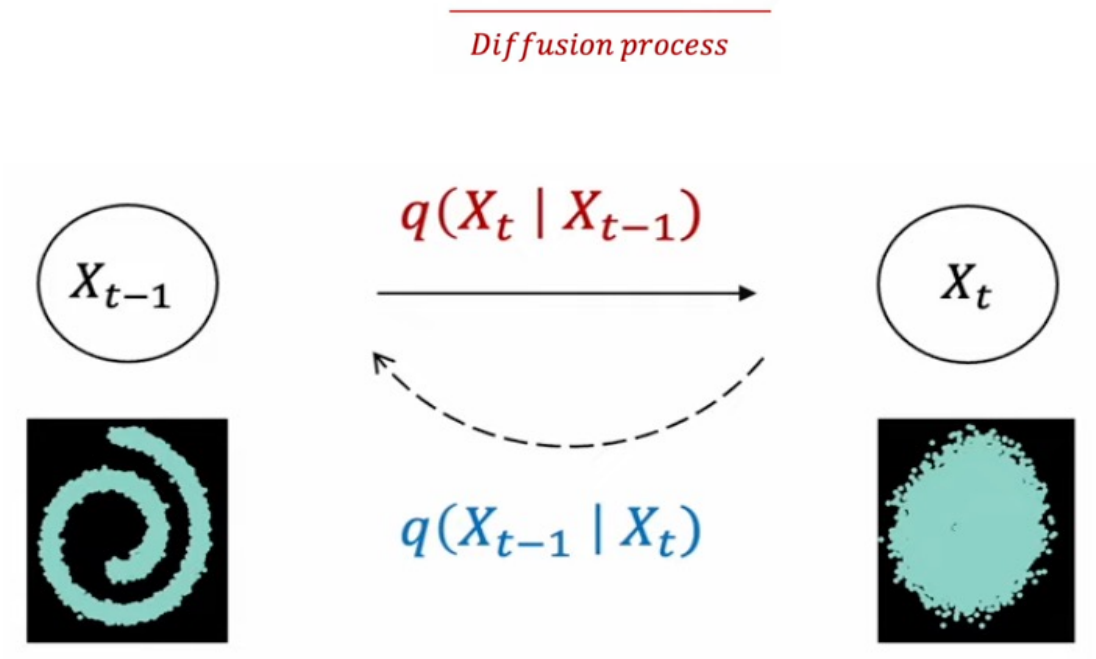


05. Diffusion

Diffusion model

Diffusion model은 Generative model에서 학습된 데이터의 패턴을 생성해내는 역할을 함

패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너뜨리고(Noising), 이를 다시 복원하는 조건부 PDF를 학습(Denoising)

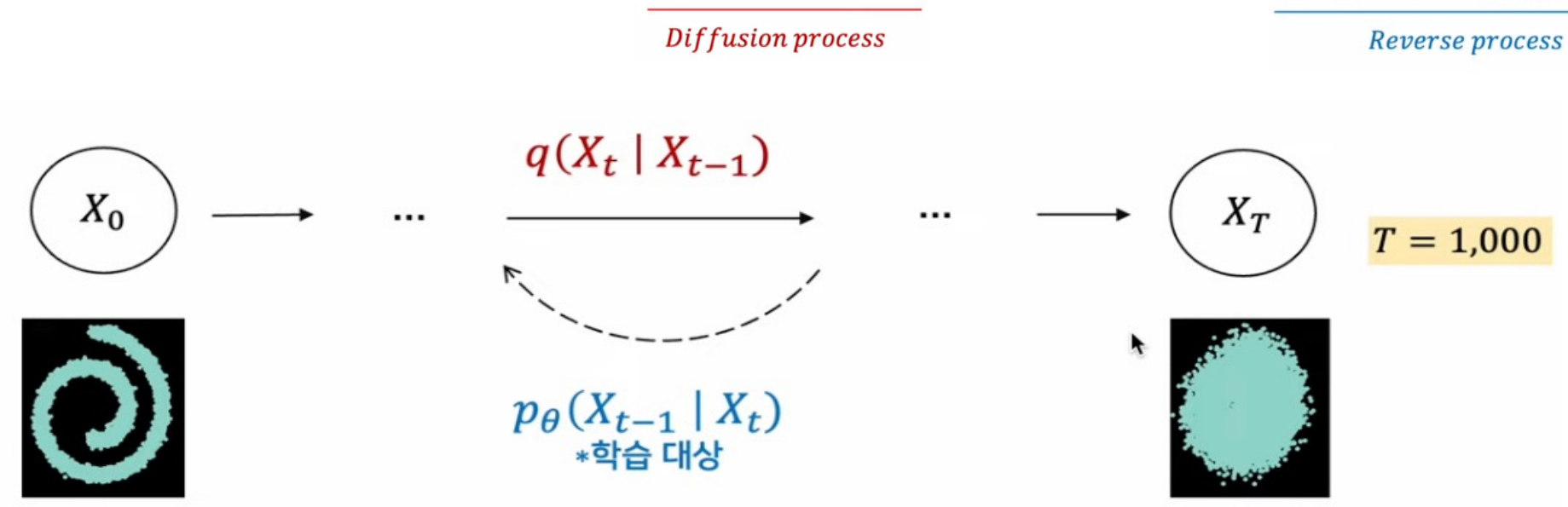


05. Diffusion

Diffusion model

Diffusion model은 Generative model에서 학습된 데이터의 패턴을 생성해내는 역할을 함

패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너뜨리고(Noising), 이를 다시 복원하는 조건부 PDF를 학습(Denoising)



한번에 변환하기 어렵기 때문에 2개의 변화 과정은 Markov Chain을 통해 매우 많은 단계로 구성됨

05. Diffusion

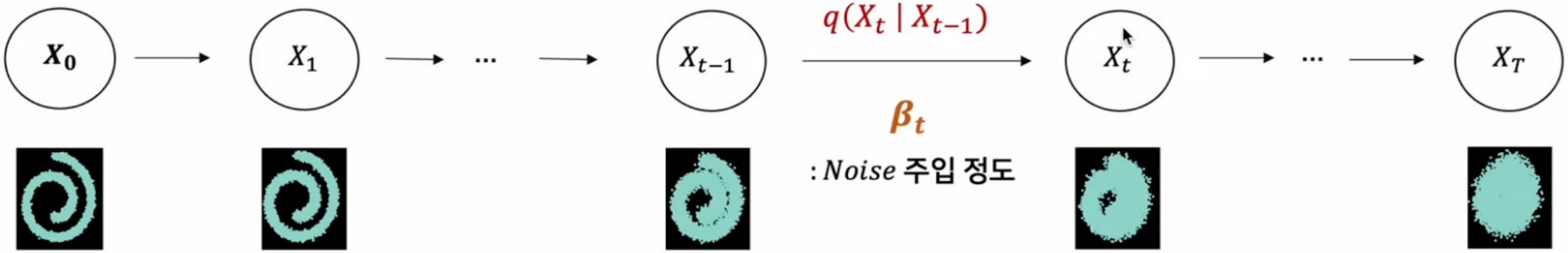
■ Diffusion model : Forward process

Gaussian Noise를 주입하는 과정

한 step마다 주입하는 noise는 사전에 정의됨 (β)

점진적으로 커지는 Noise (Linear, Sigmoid, Quadratic)

$$q(X_{1:T} \mid \boldsymbol{X_0}) \coloneqq \prod_{t=1}^{T} q(X_t \mid X_{t-1}), \qquad q(X_t \mid X_{t-1}) \coloneqq N(X_t\,;\,\mu_{X_{t-1}}, \Sigma_{X_{t-1}}) \coloneqq N\left(X_t\,;\sqrt{1-\beta_t}\,X_{t-1},\ \beta_t \cdot I\right)$$



05. Diffusion

■ Diffusion model : Reverse process

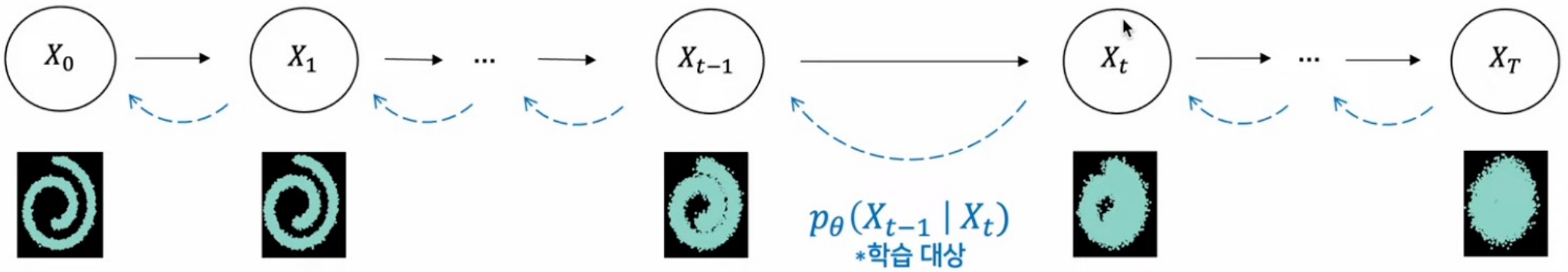
패턴을 복원해내는 과정으로 학습의 대상

조건부 가우시안 분포의 parameter : 평균과 분포를 학습

$$p_\theta(X_{0:T}) := p(X_T) \prod_{t=1}^{T} q(X_{t-1} \mid X_t), \qquad p_\theta(X_{t-1} \mid X_t) := N(X_{t-1}; \mu_\theta(X_t, t), \Sigma_\theta(X_t, t))$$

학습 대상

(mean & variance function)



05. Diffusion

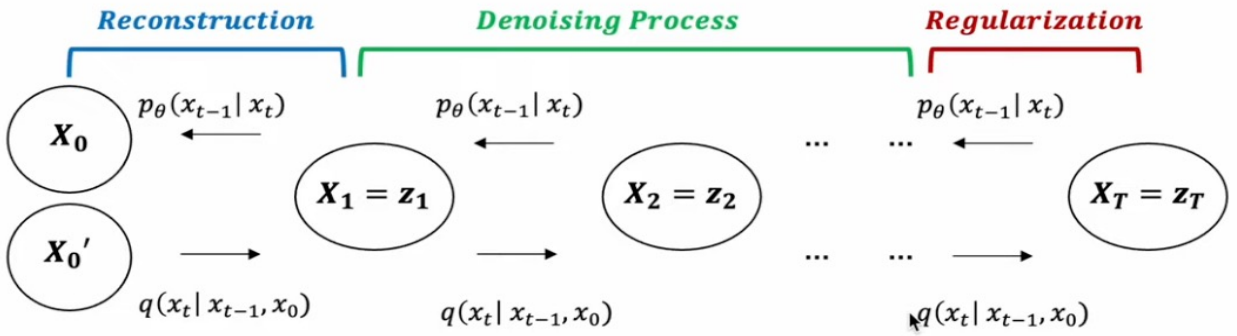
■ Diffusion model : Reverse process Loss

VAE와 동일하게 reconstruction loss와 regularization을 가짐

Denoising process 항을 추가로 가지고 있다는 것이 특징

모든 t 시점의 z를 학습하기 위해 t시점의 forward process 분포와 reverse process 분포가 KL Divergence로 같아지도록 학습

✓ Diffusion



$$Loss_{Diffusion} = D_{KL}(q(z \mid x_0) \| P_\theta(x_0 \mid z)) - E_{z\sim q(z|x)}[\log P_\theta(z)]$$

$$= D_{KL}(q(z \mid x_0) \| P_\theta(z)) + \sum_{t=2} D_{kL}(q(x_{t-1} \mid x_t, x_0) \| P_\theta(x_{t-1} \mid x_t)) - E_q[\log P_\theta(x_0 \mid x_1)]$$

Regularization　　Denoising Process　　Reconstruction

과제 소개

homework

■ 코드과제

주어진 코드 과제 문제 풀이 후 kubig 공식 github에 업로드

마감 :과제가 제시된 후 2월 5(월) 자정까지 제출

Github는 과제제출 - 해당주차내에 파일명 : 00기_이름_0주차 로 통일

homework

■ 공지

개별 스터디(3주간 진행)

- 큰 주제는 설문 관련 주제로 선정하되 디테일한 방향성은 스터디원과 논의

프로젝트(4주 후 2월 29일 KUBIG Contest 진행)

- 다음주까지 주제 선정 권장

Thank you for Listening