DATA
SCIENCE LAB

Generative Model (1)
(Autoregressive, VAE, GAN)

23.03.16 / 7기 전재현

CONTENTS

01. Generative Model

Discriminative Model

Data: x

- 지금까지 해온 일반적인 classification task
- Data x 주어졌을 때 Class y 속할 확률 구하자

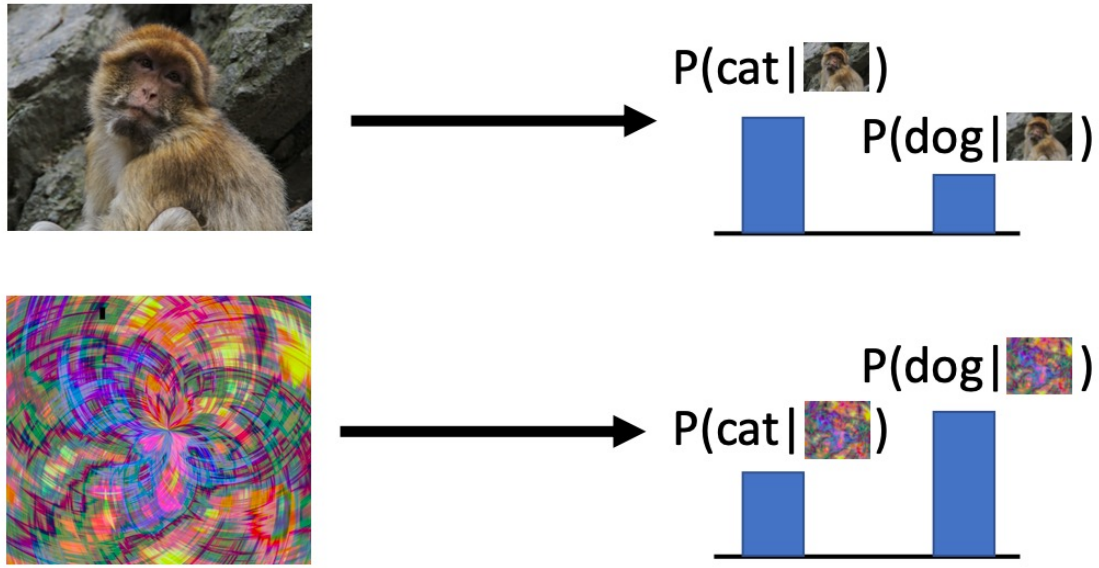
Label: y

Cat

Discriminative Model:
Learn a probability distribution $p(y|x)$

01. Generative Model

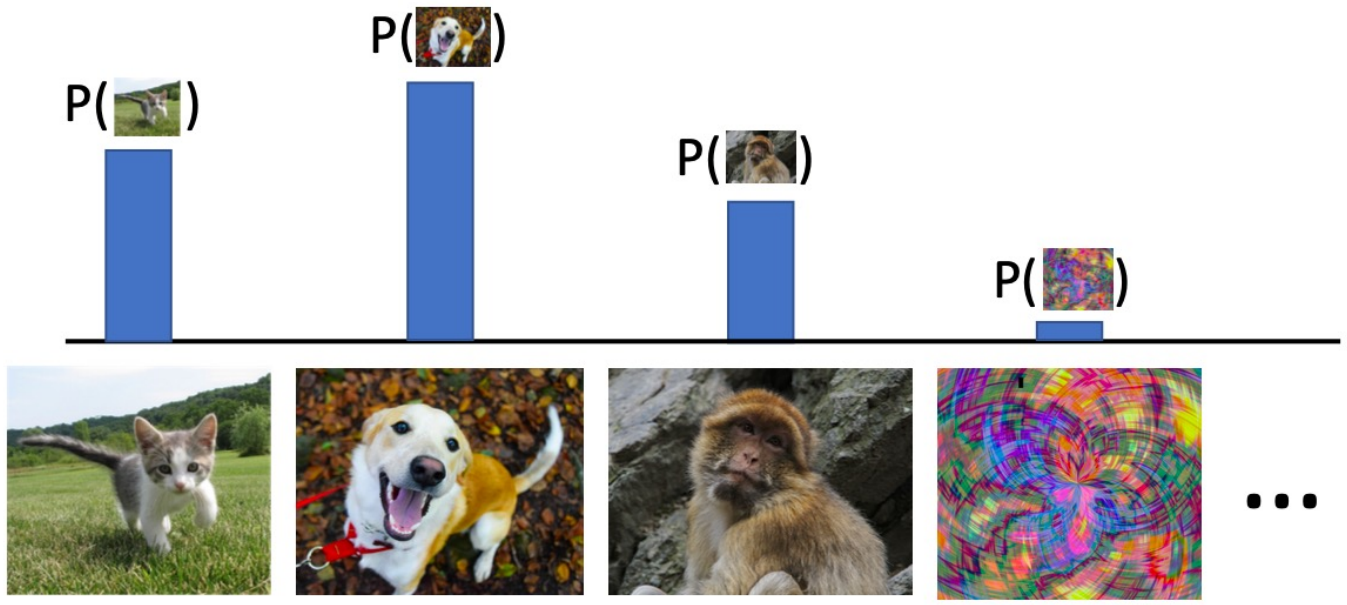
Discriminative Model



- but different input (아예 다른 클래스의 img), unreasonable input (존재하기부터 어려운 img) 처리할 수 없음..
- 어떻게든 확률을 낼 수 밖에 없기에

01. Generative Model

(Unconditional) Generative Model

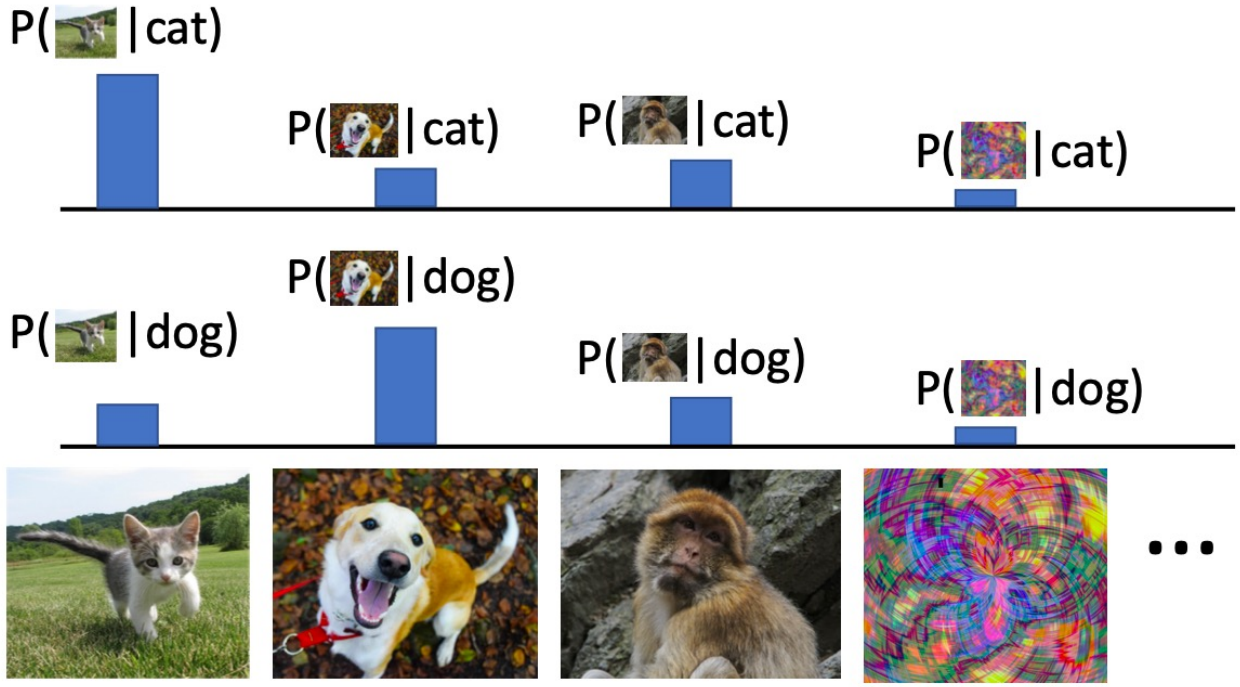


- img가 실제로 존재할 만한 img인지 그 확률을 나타내는 p(x)를 알아내자!
- x : image
- then model can 'reject' unreasonable input (outliers)
- 이렇게 p(x)를 알아냈다면 여기서 sample해서 new data를 generate할 수 있는것!

Generative Model:
Learn a probability distribution p(x)

01. Generative Model

Conditional Generative Model



- Class y 대해 Data (img) x가 나올 확률인 p(x|y)를 구할 수도 있음
- 원하는 y 넣고 p(x|y)에서 sample하면 class y의 new data generate 가능

Conditional Generative Model: Learn p(x|y)

01. Generative Model

Bayes' Theorem

$$P(x \mid y) = \frac{P(y \mid x)}{P(y)} P(x)$$

Conditional Generative Model

Discriminative Model

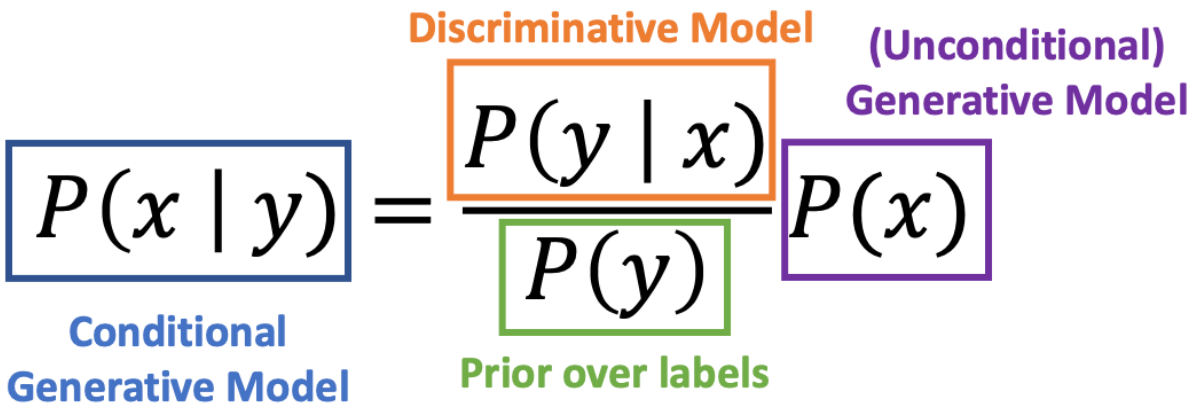
Prior over labels

(Unconditional) Generative Model

- easy to derive..
- 앞서 살펴본 모델 간의 관계 볼 수 있음

01. Generative Model

Bayes' Theorem



- p(y|x)에 대한 식으로 바꿔 쓰고

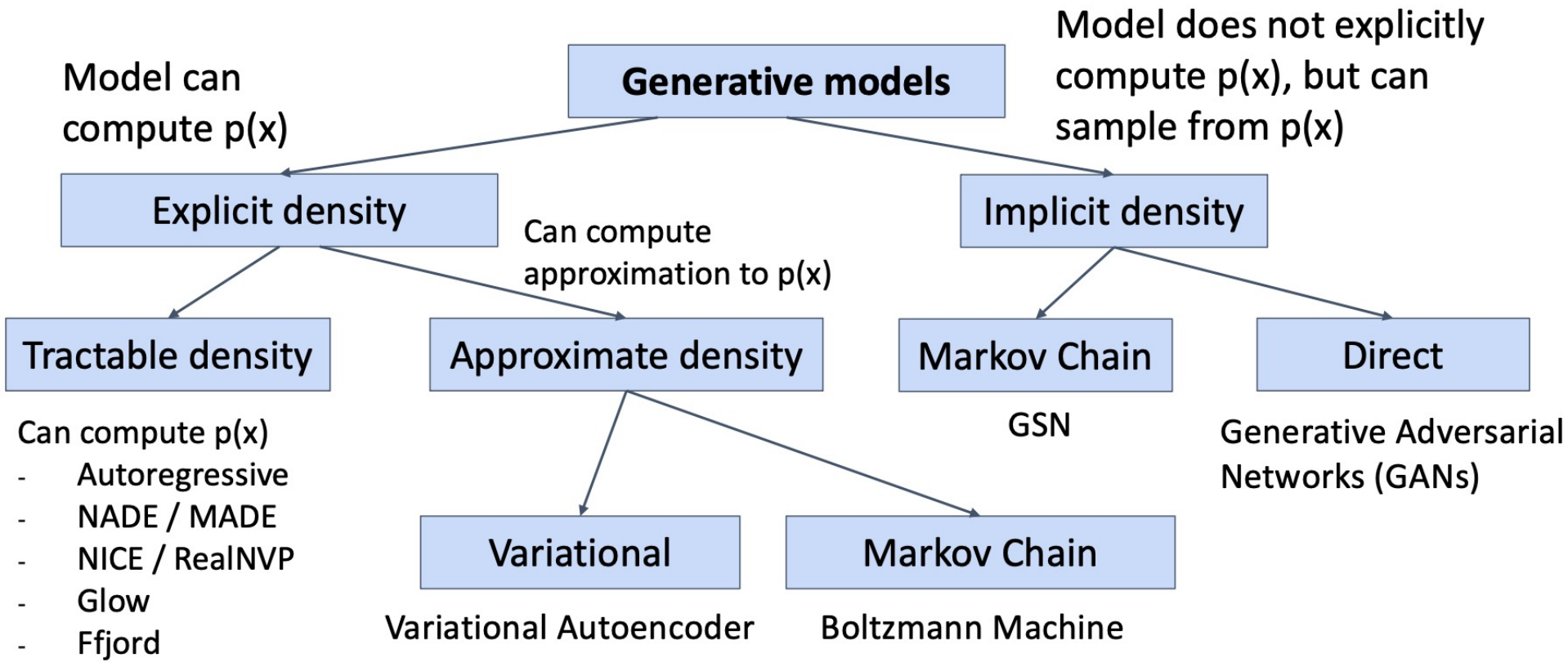
- 현재 구하고자 하는 대상 = y
- 주어진 관측값 = x

- posterior : 관측값이 주어졌을때 구하고자 하는 대상 나올 확률 = p(y|x)
- likelihood : 구하고자 하는 대상 정해졌을때 관측값 나올 확률 = p(x|y)
- prior : 구하고자 하는 대상에 대한 확률 = p(y) (dist over labels)
- more on MLE, MAP..

01. Generative Model

Taxonomy

- we need to generate new data x
- explicit : p(x) computed / implicit : don't have to compute p(x)
- 3가지 방식 살펴보고 그림 이따가 다시 보자



02. Autoregressive Model

Overview

- Taxonomy에서 봤듯이 prob dist p(x)를 정확히 구하는 것이 목표! (then sample)
- how?

02. Autoregressive Model

Overview

Goal: Write down an explicit function for $p(x) = f(x, W)$

Given dataset $x^{(1)}, x^{(2)}, \ldots x^{(N)}$, train the model by solving:

$$W^* = \arg\max_W \prod_i p(x^{(i)})$$

Maximize probability of training data
(Maximum likelihood estimation)

$$= \arg\max_W \sum_i \log p(x^{(i)})$$

Log trick to exchange product for sum

$$= \arg\max_W \sum_i \log f(x^{(i)}, W)$$

This will be our loss function!
Train with gradient descent

- p(x) model을 parameterize
- training data x(i)들은 이미 가지고 있는 data -> p(x) model로부터 x(i) 들이 나올 확률이 높아야 함! (MLE)
- 편의상 p(x) maximize하는거 likelihood maximize라고 함 (맥락은 같으니)
- 아래 식이 max되는 w 찾아서 p(x) modeling 하는 것

02. Autoregressive Model

Overview

- p(x) model 구성하는 trick : x를 쪼개자
- then 아래처럼 구성 가능
- again, train할때 training data 대해 p(x) maximize (이미 training set에 나온거니까 나중에 p(x)에서 sample해도 그거 나올 확률 높아야됨)
- test할때는 이런 p(x) 기반으로 generate x

Assume x consists of multiple subparts:

$$x = (x_1, x_2, x_3, \ldots, x_T)$$

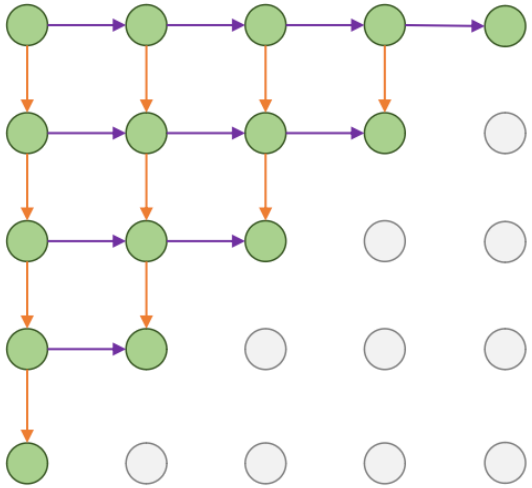
Break down probability using the chain rule:

$$\begin{aligned} p(x) &= p(x_1, x_2, x_3, \ldots, x_T) \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots \\ &= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}) \end{aligned}$$

Probability of the next subpart given all the previous subparts

02. Autoregressive Model

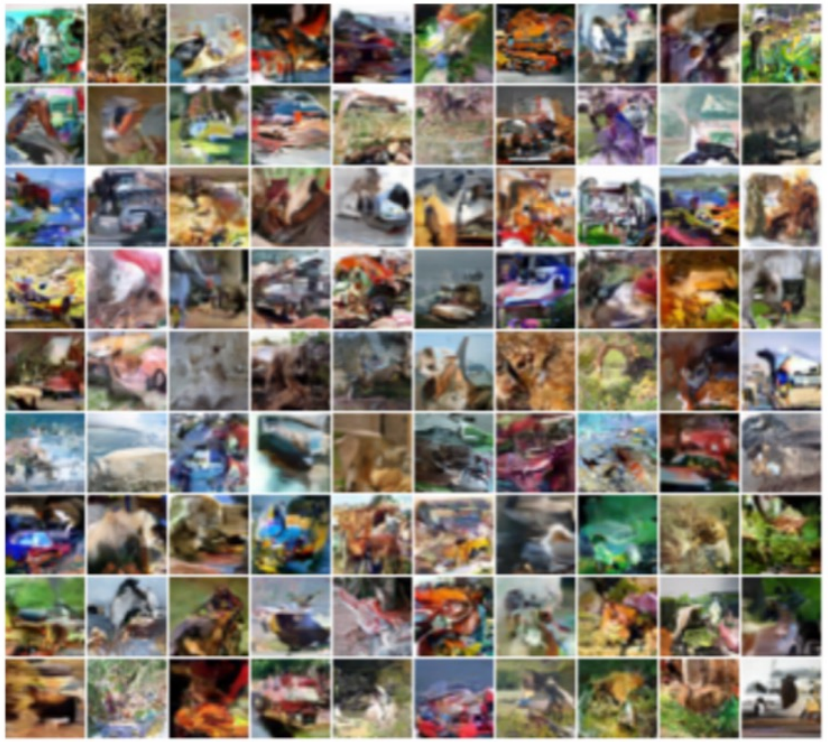
PixelRNN



- For image,
- x를 pixel x0, x1,… 로 쪼개고
- train할때 위쪽 왼쪽 pixel이 a,b라면 현재 pixel c 나올 확률 높아야돼를 학습
- test할때는 pixel 단위로 차례대로 generate (sample from p(x)) -> 다 거치면 하나의 img 생성됨

02. Autoregressive Model

PixelRNN



32x32 CIFAR-10

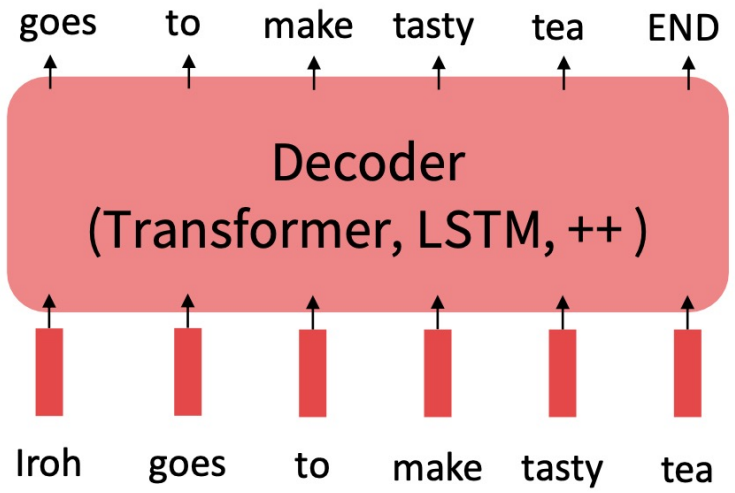
- generated img not bad
- but 가까이서 보면..

02. Autoregressive Model

Language Modeling (GPT)

Step 1: Pretrain (on language modeling)

Lots of text; learn general things!

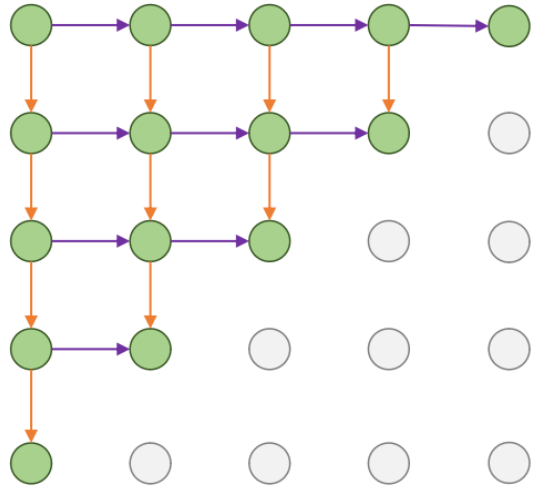


- For text,
- 아까 본 p(x) 형태가 language model의 방식과 정확하게 일치!
- 그 중에서 GPT : Decoder of Transformer alone Pretrained
- pretrained how? by language modeling on lots of lots of text data (확률적으로 다음 단어(token)은 이게 나오겠구나..)
- after, may be fine-tuned / other decoding methods (why?)

02. Autoregressive Model

Pros and Cons

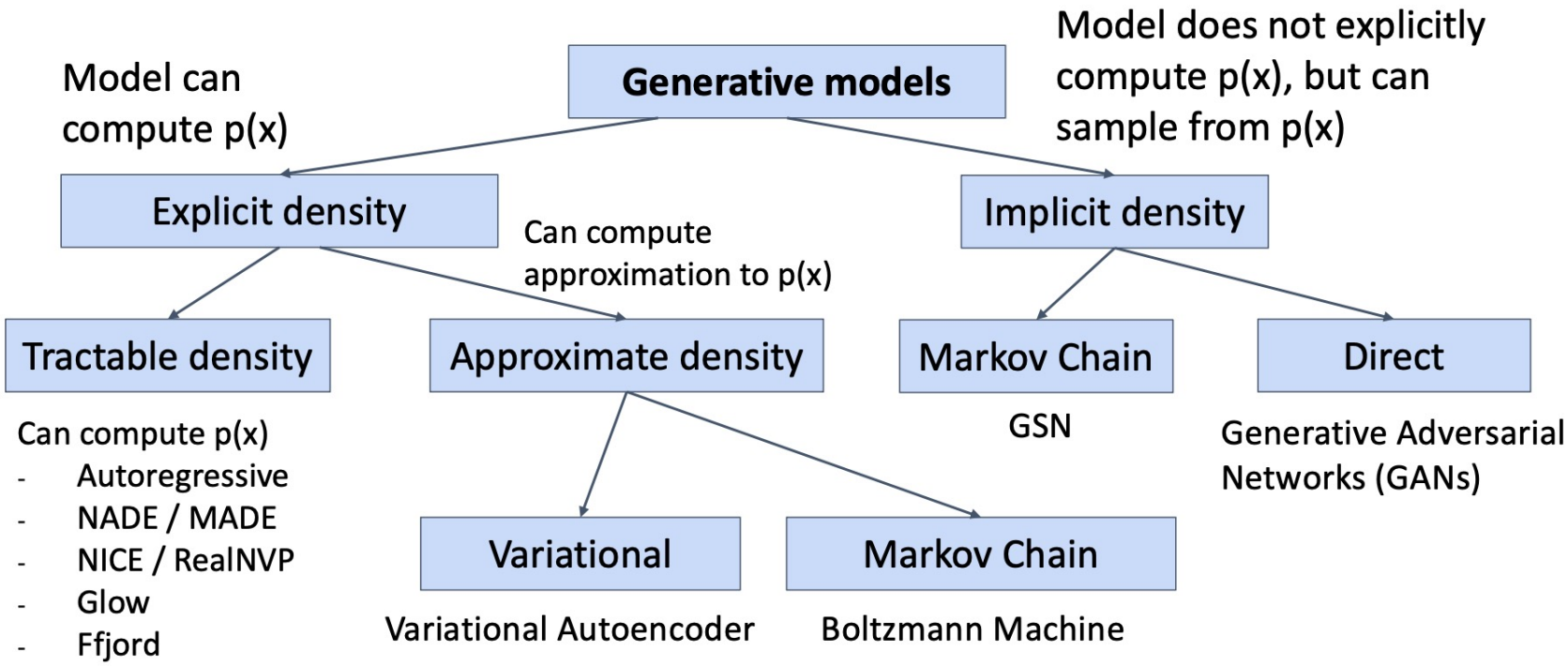
- 이렇듯 p(x) 식을 정확히 놓고 구해내려는 방식이 Autoregressive
- Pros
  - explicitly compute p(x)
- Cons
  - sequential generation -> very slow (one by one)



03. VAE Recap

Overview

- 꼭 p(x)에 대한 식을 정확히 구해야 할까? just approximate



03. VAE Recap

Overview

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation **z**

Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

- VAE(라는 이름의 모델)이 이미 unobserved latent z로부터 온 training data로 train된 상황 가정 (z 모르는 상황)

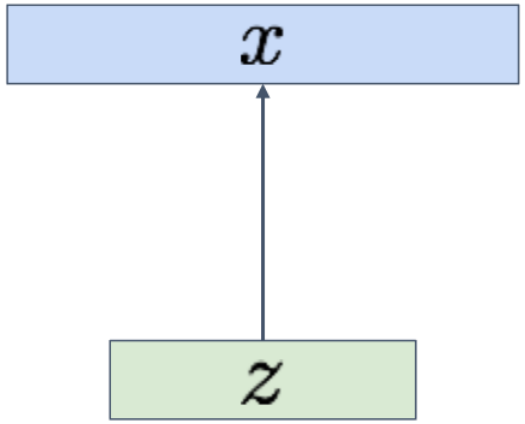
03. VAE Recap

Overview

After training, sample new data like this:

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

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



- while test time (sample new data할 시간)
- Decoder 파트가 이처럼 구성

[Decoder 단계]

- p(z)에서 z 하나 sample
- 그 z를 NN 같은거에 넣어서 p(x|z)의 distribution이 output됨
- 그렇게 나온 p(x|z)에서 x 하나 sample하면 generate new data 된것

1. but what distribution is p(z)..?
2. and how to output prob dist with NN..?

03. VAE Recap

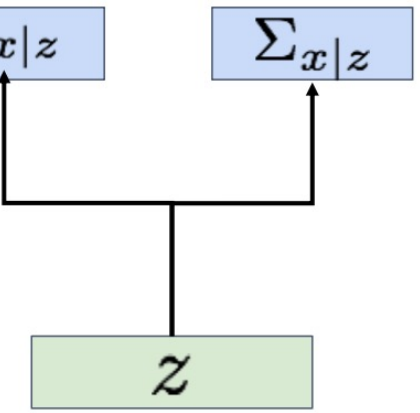
Overview

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

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



1. Assume simple prior p(z) – into Gaussian
2. img의 prob dist를 parameteric form으로 보고 그 param이 NN 거쳐서 나오는 값이도록!
   - p(x|z) 역시 Gaussian 가정 & mean, covariance를 output
   - per pixel mean, cov를 생각하면됨
   - 이때 편의상 pixel 간 cov=0 (indep 가정) -> diagonal Gaussian

03. VAE Recap

How to train

- train으로 돌아와서 then how to train?
- easy idea) training data x들 나올 likelihood maximize 하는 방식으로 (maximize p(x))

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation **z**

Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

03. VAE Recap

How to train

- training data들이 어떤 z로부터 generate된건지 알수 있다면?
- 그냥 p(x|z)들을 곧바로 maximize해서 최종적으로 p(x) maximize하면됨
- but we don't know z (model should learn z by itself)

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation **z**

Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

03. VAE Recap

How to train

We don't observe z, so need to marginalize:

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

- idea1) marginalize z (cuz we don't know)
- both terms computable
  - p(x|z) : what decoder network is computing
    - 앞에서 모르는 값이라고 했는데?
    - 아까는 각 x가 어떤 z로부터 온건지 정확히 알아야했던거
    - 여기서는 integral로 모든 가능성 있는 z에 대해서 계산
  - p(z) : assumed Gaussian
- but impossible to integrate over all z

03. VAE Recap

How to train

Another idea: Try Bayes' Rule:

$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{\boxed{p_\theta(z \mid x)}}$$

Problem: No way to compute this!

Another idea: Try Bayes' Rule:

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

Solution: Train another network **(encoder)** that learns $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

- idea2) Bayes' Thm
- Trick comes in : auxillary network 'Encoder' 추가! (variational inference)

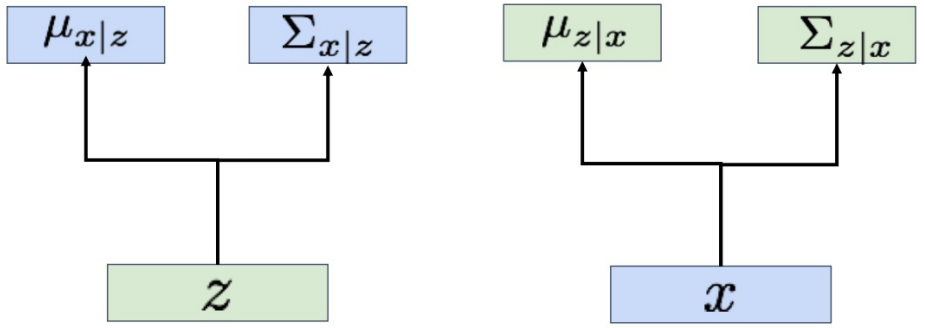
03. VAE Recap

How to train

Decoder network inputs latent code z, gives distribution over data x

Encoder network inputs data x, gives distribution over latent codes z

$p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$ $q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$



If we can ensure that $q_\phi(z \mid x) \approx p_\theta(z \mid x)$,

then we can approximate

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

Idea: Jointly train both encoder and decoder

- Decoder와 같은 형식으로 Encoder 구성
- we are 'approximating' p(x)! & again, maximize such (paramterized) p(x) on training data

03. VAE Recap

How to train

- maximize하고 싶은 p(x)에 대한 식으로

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

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

Split up using rules for logarithms

03. VAE Recap

How to train

- 양변에 expectation over z from q(z|x)

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

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$\log p_\theta(x) = E_{z\sim q_\phi(z|x)}[\log p_\theta(x)]$$

We can wrap in an expectation since it doesn't depend on z

03. VAE Recap

How to train

- into KL-divergence -> 우선 확률분포의 차이 정도로만 이해하자
- we don't know p(z|x) (recall. encoder 추가한 이유)

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

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

KL is >= 0, so dropping this term gives a **lower bound** on the data likelihood:

03. VAE Recap

How to train

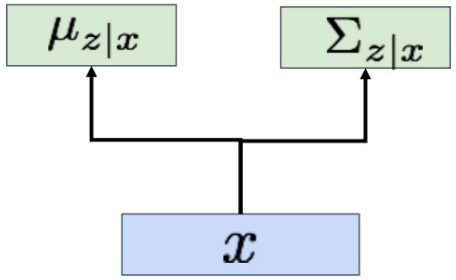
- p(x) maximize 대신 lower bound maximize하자! <-> autoregressive
  - learn phi of q, theta of p

Jointly train **encoder** q and **decoder** p to maximize
the **variational lower bound** on the data likelihood

$$\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

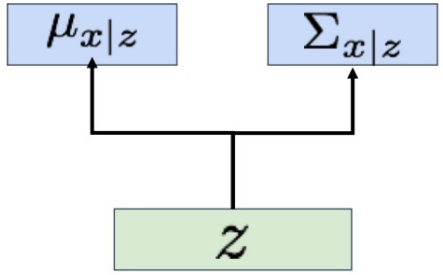
Encoder Network

$$q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$$



03. VAE Recap

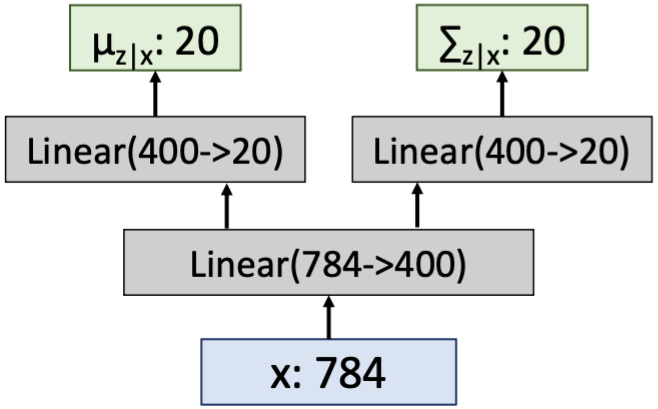
Case example

Example: Fully-Connected VAE

x: 28x28 image, flattened to 784-dim vector
z: 20-dim vector

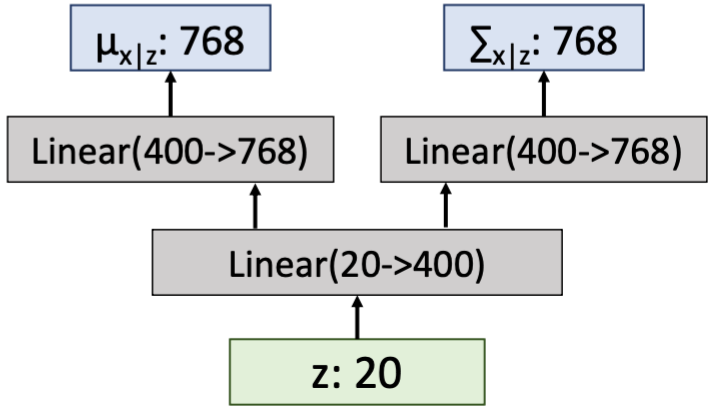
Encoder Network

$$q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x \mid z) = N(\mu_{x|z}, \Sigma_{x|z})$$



03. VAE Recap

Case example

- Train 과정 살펴보면서 아까 구한 objective의 intuition을 가져보자

- minimize KL divergence btw q(z|x), p(z)

$$-D_{KL}\left(q_\phi(z|x), p(z)\right)$$

  - 방금 encoder가 output한 dist q(z|x)가 Gaussian으로 가정한 p(z)와 유사해지도록!
  - 둘 사이 분포 가깝게 학습
- maximize prob of data reconstruction
  - 원래 x 기반으로 만든 p(z|x)에서 나오는 z sample들 하에서 expectation
  - 그런 z들 하에서 원래 img x가 나올 likelihood 확률을 maximize하는 셈!

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$$

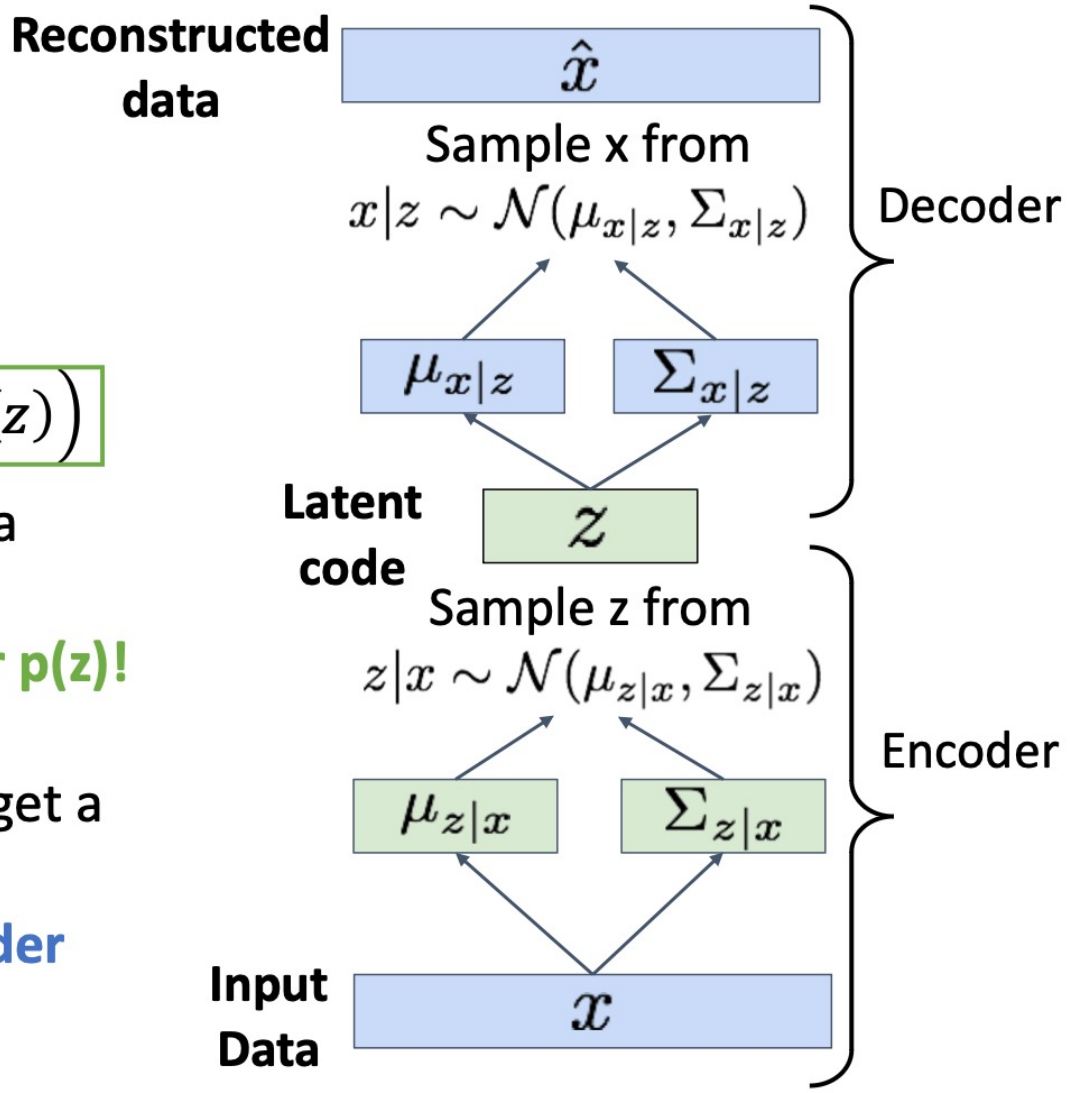
03. VAE Recap

Variational Autoencoders

Train by maximizing the **variational lower bound**

$$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**
3. Sample code z from encoder output
4. Run sampled code through **decoder** to get a distribution over data samples
5. **Original input data should be likely under the distribution output from (4)!**
6. Can sample a reconstruction from (4)



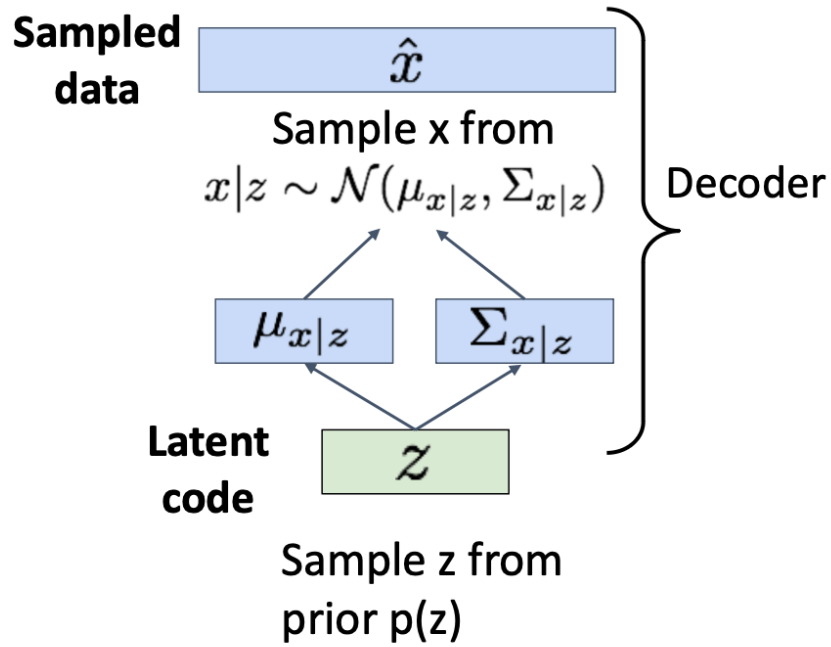
03. VAE Recap

Case example

- after training, we want to generate new data
- latent variable z 잘 넣어서 (sample해서) decoder만 사용하면 됨

After training we can generate new data!

1. Sample z from prior p(z)
2. Run sampled z through decoder to get distribution over data x
3. Sample from distribution in (2) to generate data



03. VAE Recap

Case example

- Result

32x32 CIFAR-10

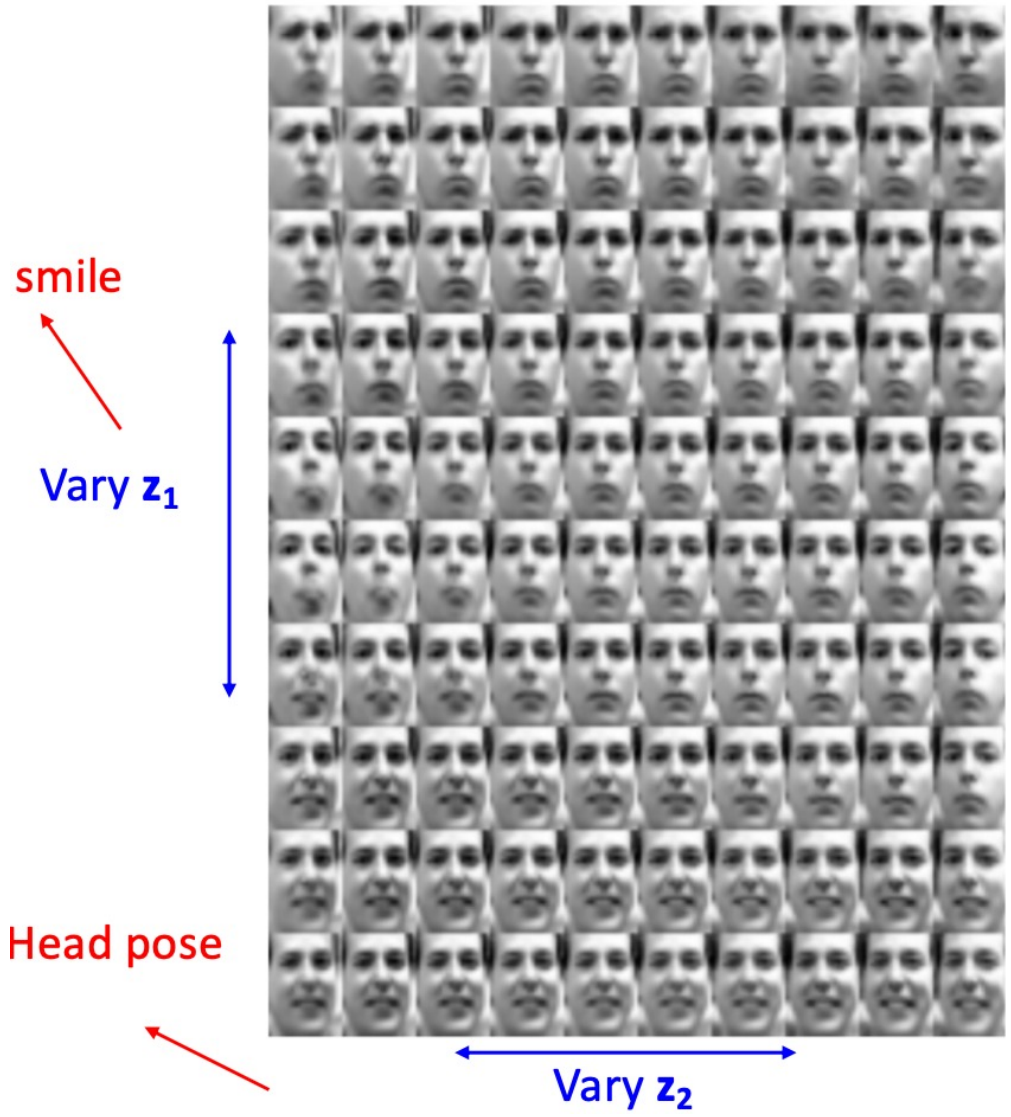
Labeled Faces in the Wild

Figures from (L) Dirk Kingma et al. 2016; (R) Anders Larsen

03. VAE Recap

Latent Variable

- VAE 가장 큰 특징은 latent variable z가 있다는 것
  - autoregressive는 없음
- 예시처럼 latent가 data의 internal representation 을 학습함을 알 수 있음!!
  - z를 2차원으로 차원축소 & z 기반 생성 결과
  - 어떤걸 represent할지는 model이 결정
  - latent 바꾸면 생성 결과도 바뀌게 할 수 있음



03. VAE Recap

Pros and Cons

Autoregressive models

- Directly maximize p(data)
- High-quality generated images
- Slow to generate images
- No explicit latent codes

Variational models

- Maximize lower-bound on p(data)
- Generated images often blurry
- Very fast to generate images
- Learn rich latent codes

*04. Information Theory

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

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

KL is >= 0, so dropping this term gives a **lower bound** on the data likelihood:

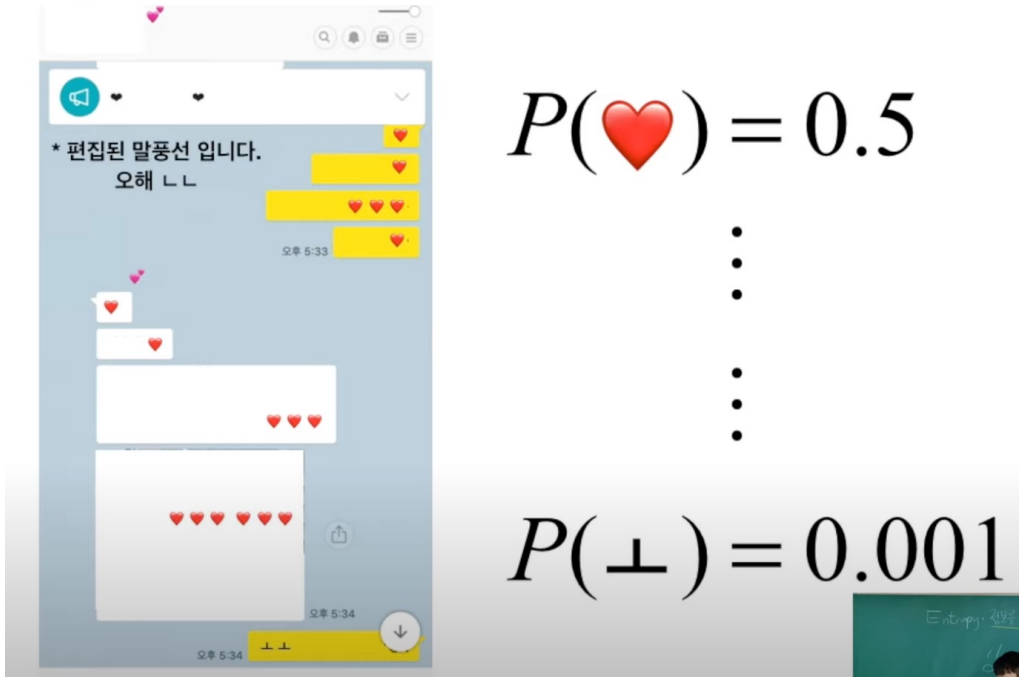
- KL = 확률 분포의 차이? how to measure that?
- Information Theory
- 상당히 자주 나오는 개념!

*04. Information Theory

Entropy

- Entropy = 정보를 표현하는데 필요한 최소 평균 자원량
- 정보를 어떻게 표현?
  - by bit (0 or 1)
  - then '최소' : 최대한 적은 bit 수로 정보를 표현하고 싶은 것

- 확률이 낮은 애를 길게 코딩해야 됨! (vice versa)
  - if ❤️ = 111 / ㅗ = 0 -> too long..
  - 길이가 -log_2 p_i 함수로 표현됨
  - then bit 길이 기댓값 =
  - 이와 같이 '최적의 전략' 하에서 자원량 기댓값의 최소가 나옴

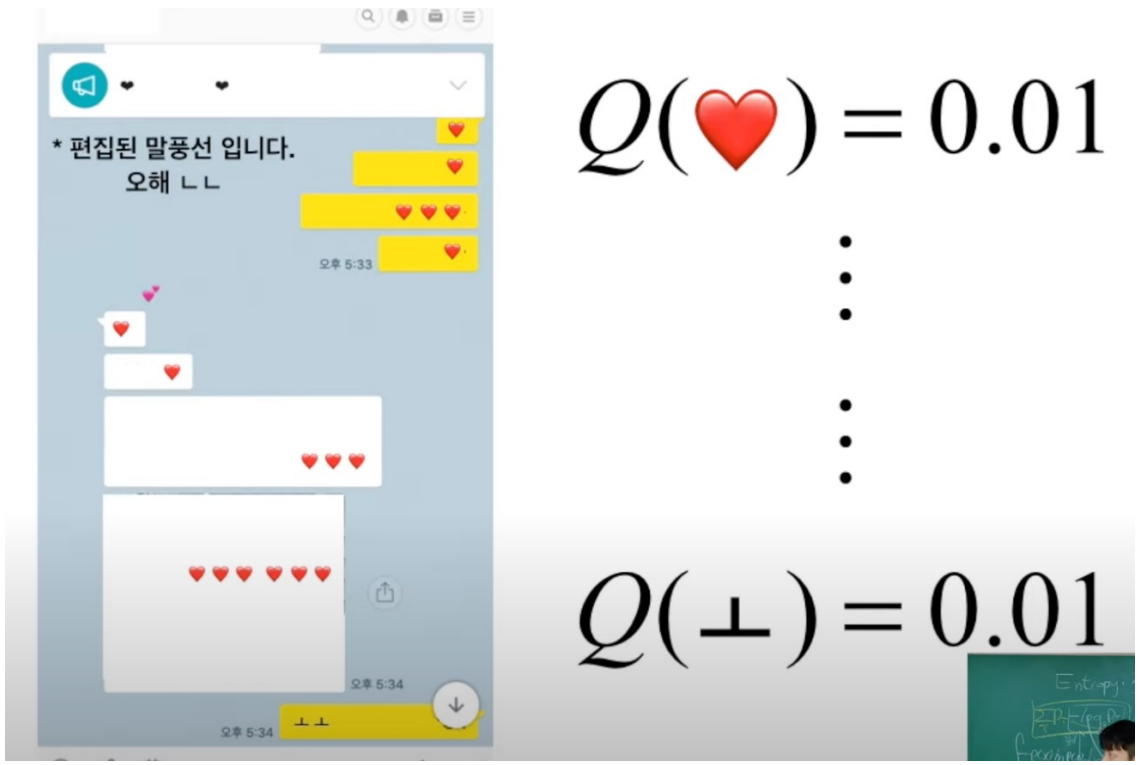


*04. Information Theory

Cross Entropy

- 내가 생각한 확률 q가 실제 확률 p와 다르다면?
  - 길이가 -log_2 q_i
  - then bit 길이 기댓값 =
  - 이게 아까 Entropy 값보다 클 수밖에

- recall BCE (binary cross entropy)
  - 실제 class값 : y
  - our prediction : y hat

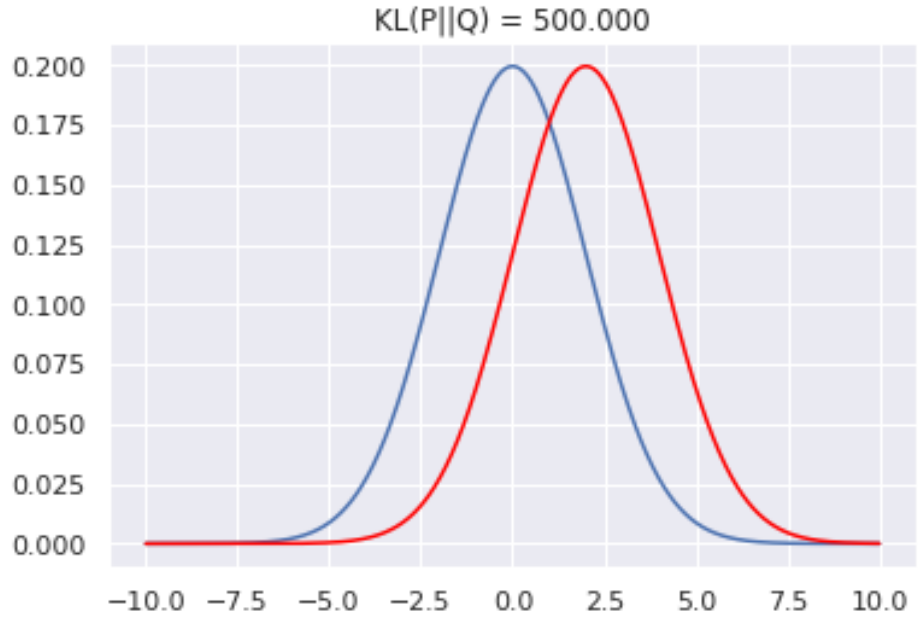


$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

*04. Information Theory

KL divergence

- q일때가 얼마나 더 비효율적인지
  - (Cross Entropy) – (Entropy) =
  - 이걸로 두 확률분포 p, q의 차이를 나타내는 것
- 주의) 거리 개념이 아님!



*04. Information Theory

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

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

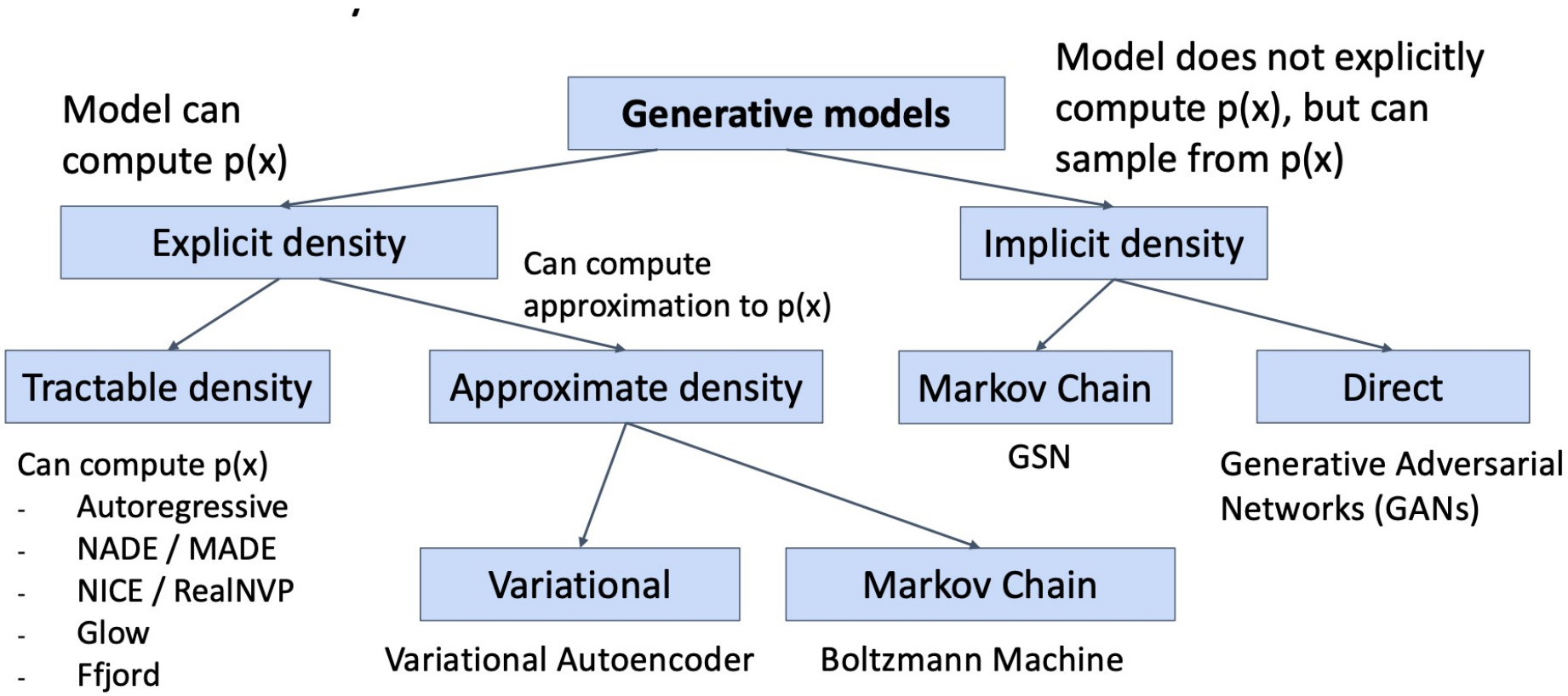
$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right) + D_{KL}(q_\phi(z|x), p_\theta(z|x))$$

KL is >= 0, so dropping this term gives a **lower bound** on the data likelihood:

- 식 다시 써보면
- okay..

05. GAN

Overview



05. GAN

Overview

Autoregressive Models directly maximize likelihood of training data:

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

Variational Autoencoders introduce a latent z, and maximize a lower bound:

$$p_\theta(x) = \int_Z p_\theta(x|z)p(z)dz \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

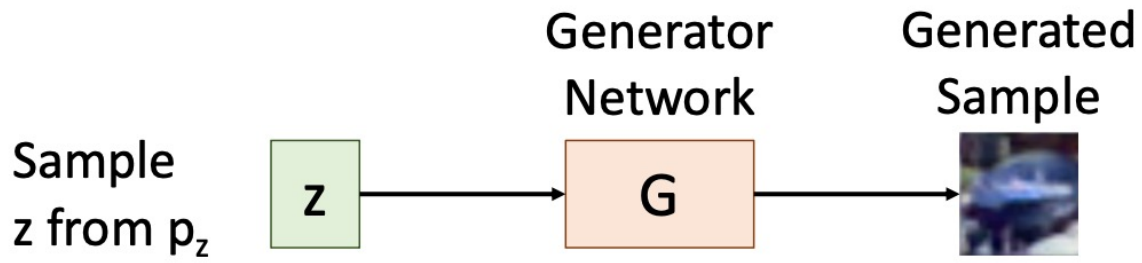
Generative Adversarial Networks give up on modeling p(x), but allow us to draw samples from p(x)

- p(x) modeling 자체를 안하고
- then how can we sample?

05. GAN

Overview

Idea: Introduce a latent variable z with simple prior p(z).
Sample $z \sim p(z)$ and pass to a **Generator Network** x = G(z)
Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!



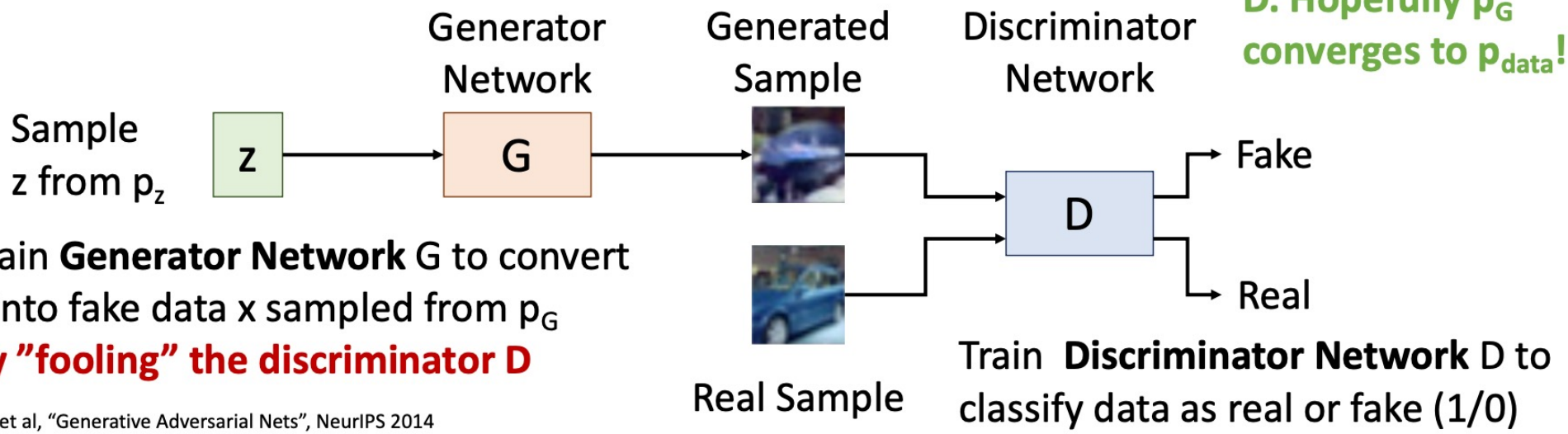
- [Goal] we want to sample x_i from true (Img) distribution p_data
- p(z) : let Gaussian (simple prior) 놓고 sample z
- then sample한 z들을 Generator Network G에 넣고 나오는 x=G(z) 값들의 분포가 존재할 것 -> 이를 p_G
  - x가 p_G로부터 sample된 것이라고 생각하자
  - explicit하게 이 p_G가 어떤 분포인지 식으로는 알 수 없음 but 이렇게 sample은 가능! <-> autoregressive, VAE
- we want p_G to be p_data -> how?

05. GAN

Overview

Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!

Jointly train G and D. Hopefully p_G converges to p_{data}!

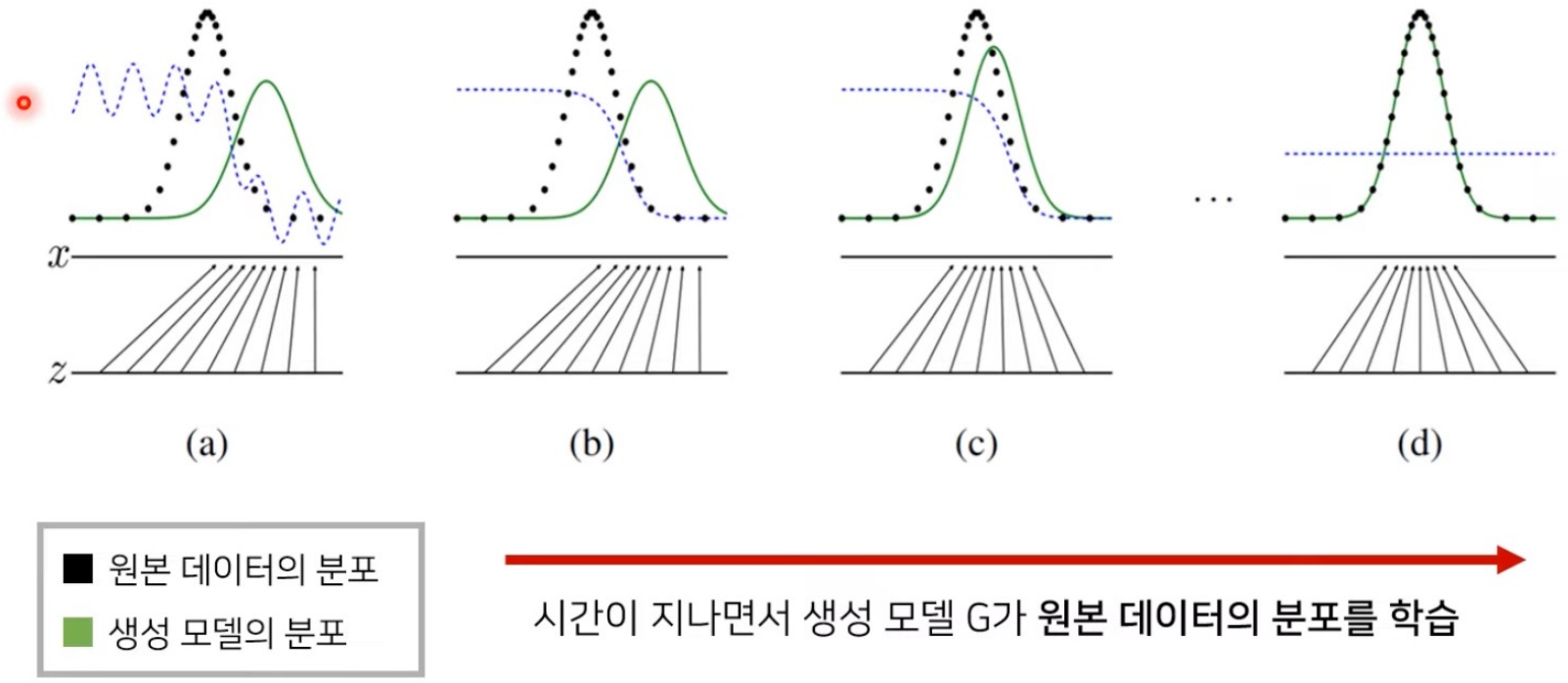


odfellow et al, "Generative Adversarial Nets", NeurIPS 2014

- idea of discriminator D : real, fake img를 구분함 (그러도록 train)
- then if G 통해 나오는 x=G(z) 들이 D에서 real로도 판별된다면, 이는 생성된 x가 그럴 법한 img라는 것. 즉, p_G와 p_data의 확률분포가 유사해지는 것!
- so, G는 real로 판별될 만한 img를 생성하도록, D는 real, fake img를 잘 구별하도록 train하자!
  - jointly train G & D

05. GAN

Overview



05. GAN

Objective: minimax game

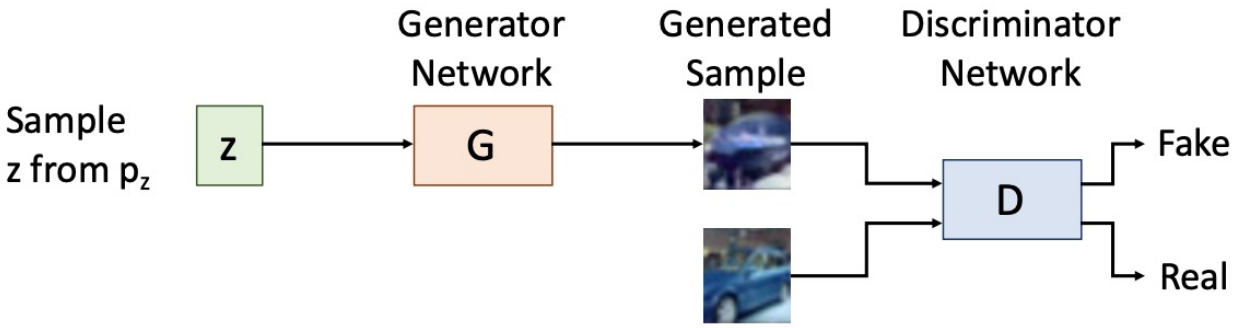
- jointly train generator G, discriminator D with minimax game

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

Generator wants D(x) = 1 for fake data



05. GAN

Objective: minimax game

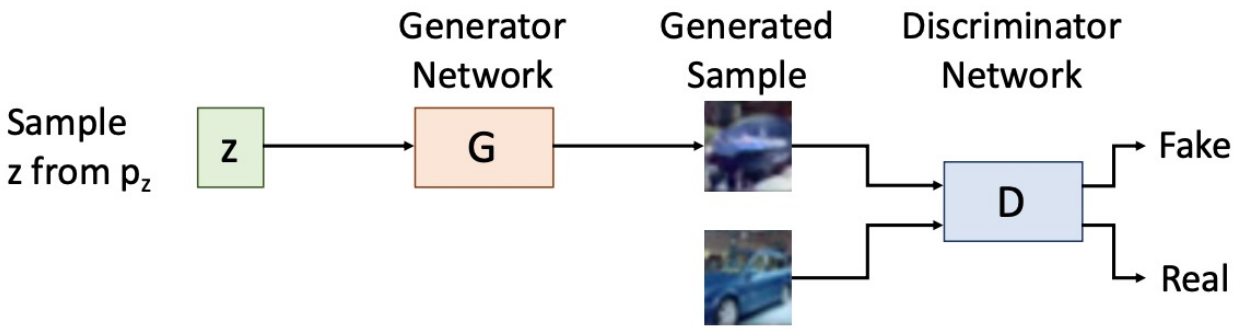
- for real data, discriminator D wants to maximize (label 1 = real)
  - D 입장에서 real data를 real로 분류하도록
  - Generator G는 관여 못함

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

Generator wants D(x) = 1 for fake data



05. GAN

Objective: minimax game

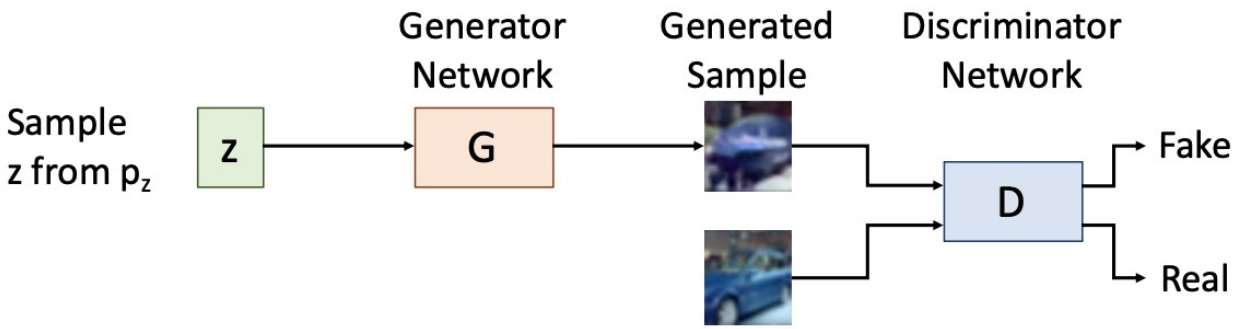
- for fake data, D wants to maximize & G wants to minimize term
  - D 입장에서 D(G(z)) 작았으면 (fake로 분류했으면) -> maximize
  - G 입장에서 D(G(z)) 컸으면 (real로 분류했으면) -> minimize

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

Generator wants D(x) = 1 for fake data



05. GAN

Objective: minimax game

- in practice, G, D가 번갈아가면서 gradient update

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)]+E_{z\sim p(z)}\left[\log\left(1-D(G(z))\right)\right]\right)$$

$$=\min_{G}\max_{D}V(G,D)$$

For t in 1, … T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

05. GAN

Objective: minimax game

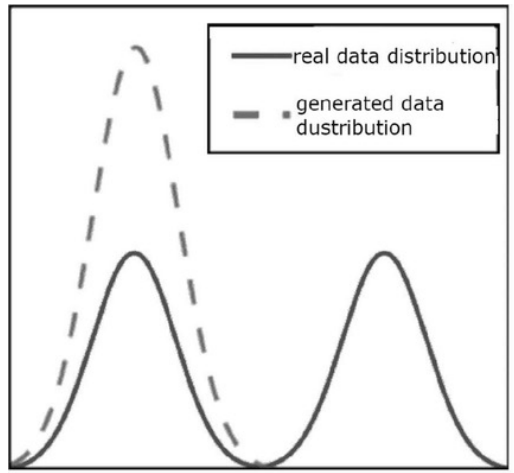
- 근데 이런 objective로 train하면 p_G = p_data 되는거 맞음??
- yes by calculating when global minimum happens
  - then global minimum에 (만약에) 도달한다면 p_G = p_data가 될것
  - but 실제로 잘 converge할지는 미지수
    - ex) mode collapse : G가 p_data에 가까운 p_G를 만들어내는게 니라 D 속일 수 있는 일부 sample만 만들어낸다면?

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_G (2 * JSD(p_{data}, p_G) - \log 4)$$

Summary: The global minimum of the minimax game happens when:

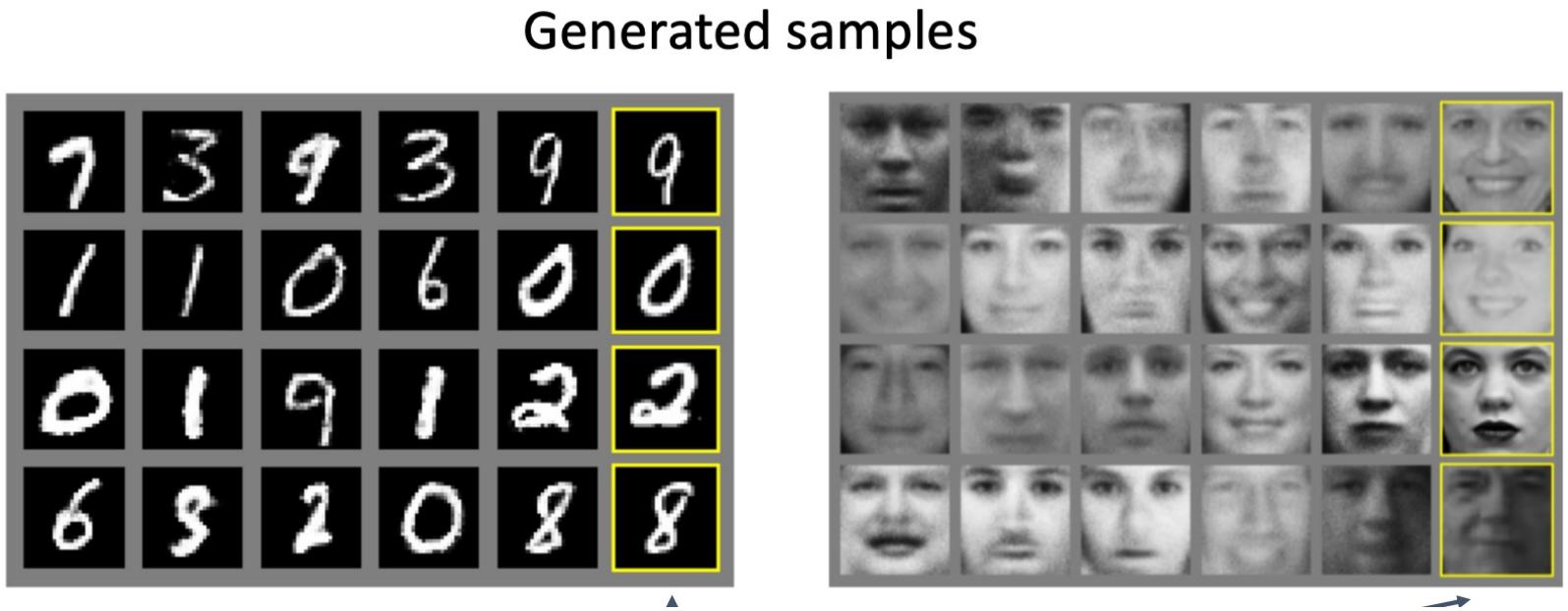
1. $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$ (Optimal discriminator for any G)
2. $p_G(x) = p_{data}(x)$ (Optimal generator for optimal D)



(a)

05. GAN

Result & Latent Space



05. GAN

Result & Latent Space

- VAE처럼 GAN에도 img의 underlying representation (structure)을 배우는 latent variable z가 존재 -> latent space 구성
  - 이걸 조절하면 – interpolate

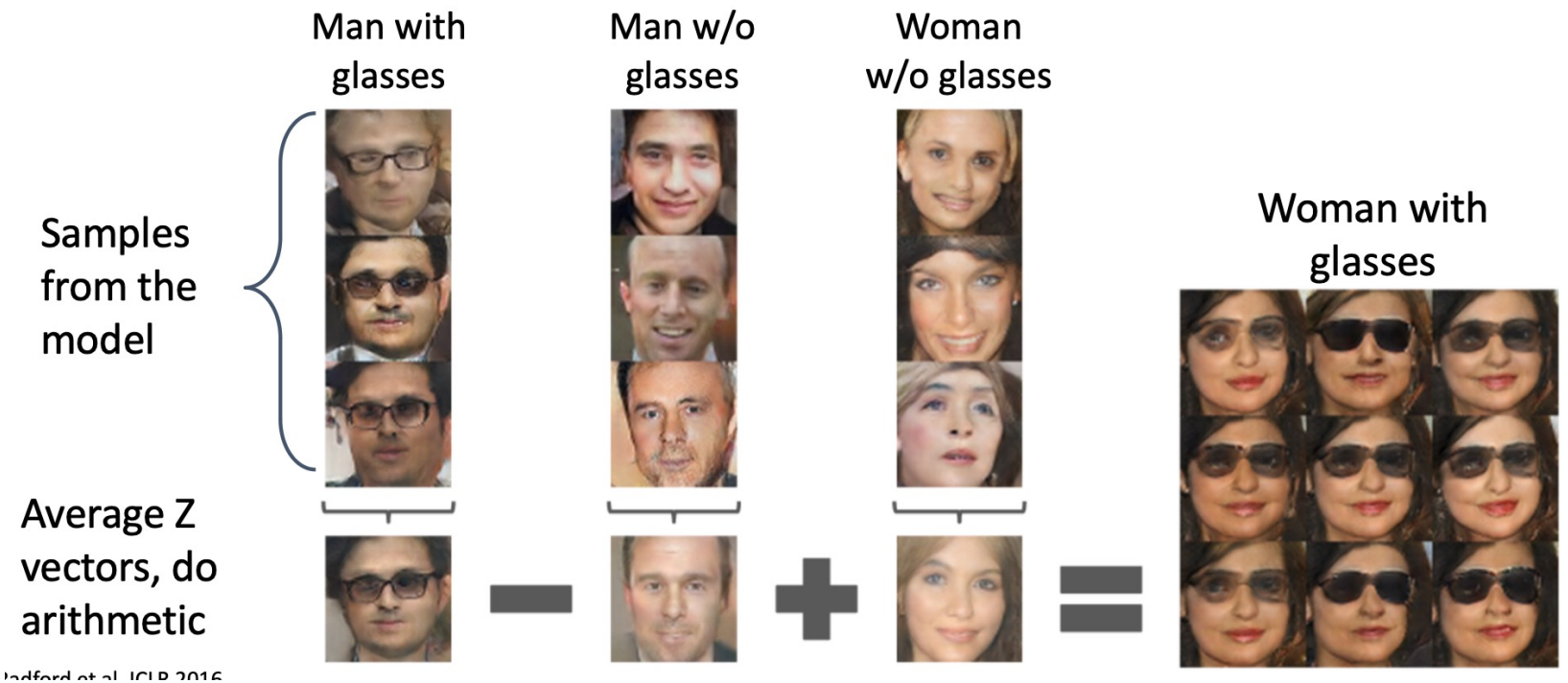
Interpolating between points in latent z space

Radford et al, ICLR 2016

05. GAN

Result & Latent Space

- VAE처럼 GAN에도 img의 underlying representation (structure)을 배우는 latent variable z가 존재 -> latent space 구성
  - 이걸 조절하면 – vector math



Radford et al. ICLR 2016

05. GAN

Result & Latent Space

VAE

Blurry

Tend to remember input images

Smooth

GAN

Sharp

Generate new unseen images

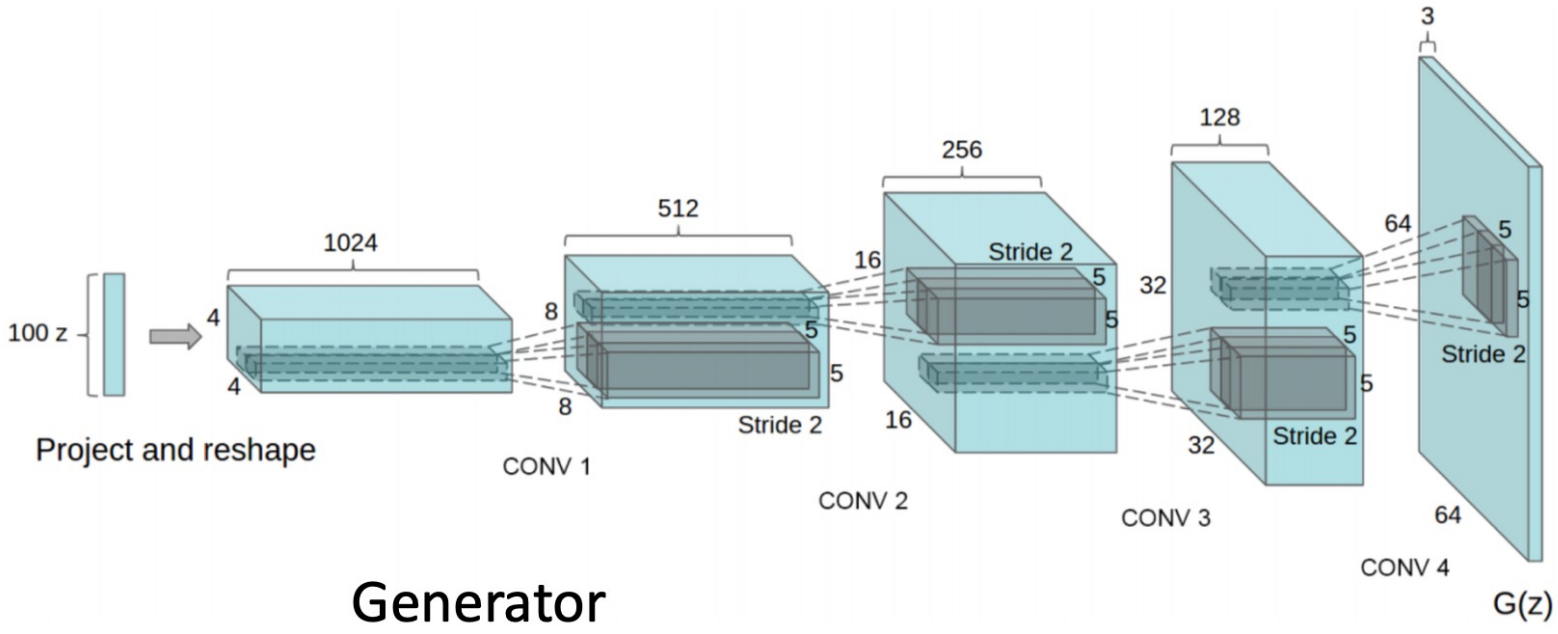
Mode collapse

Unstable convergence

05. GAN

Further Work

- GAN 유형, 구조, objective 등 다양한 variation 그 후로 계속 나옴
- 구조를 바꾼 것 중에 DCGAN : Generator를 CNN으로 구성

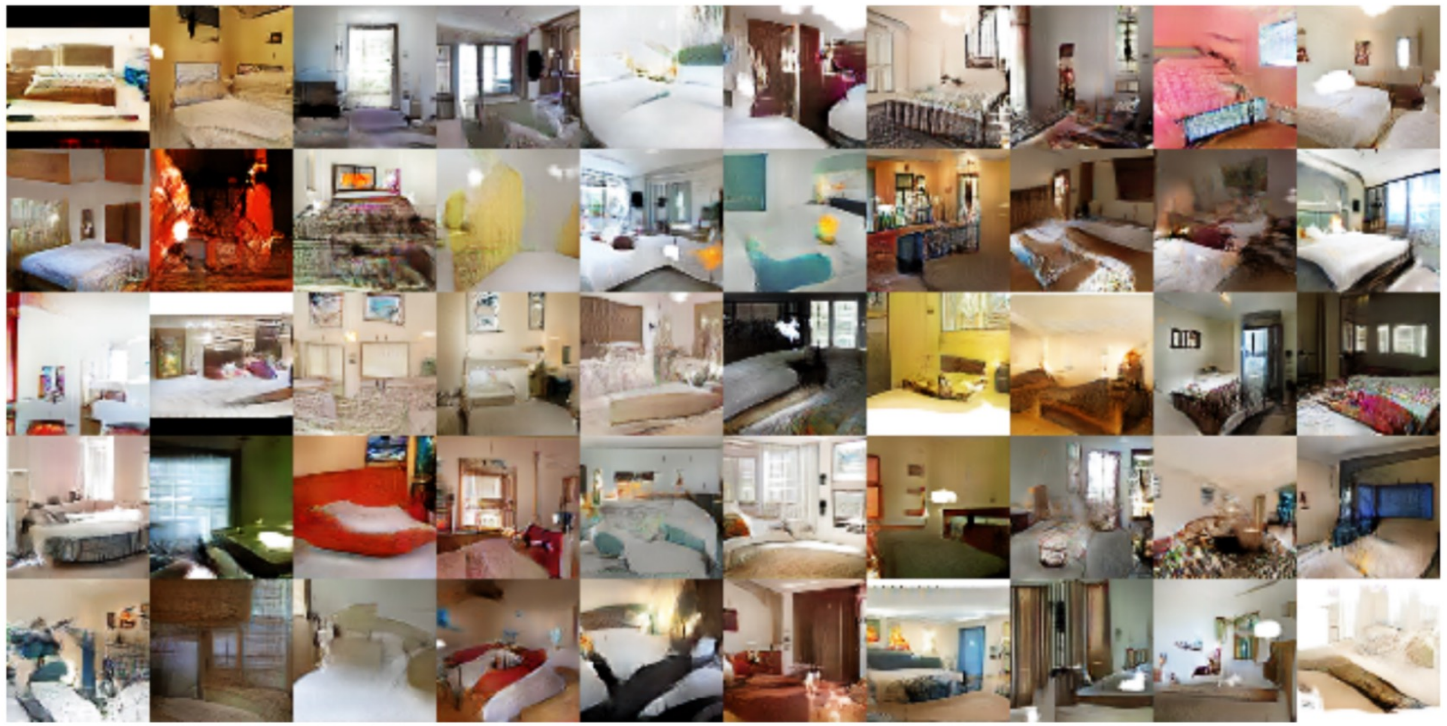


05. GAN

Further Work

- DCGAN : Generator를 CNN으로 구성
- 생성한 bedroom img

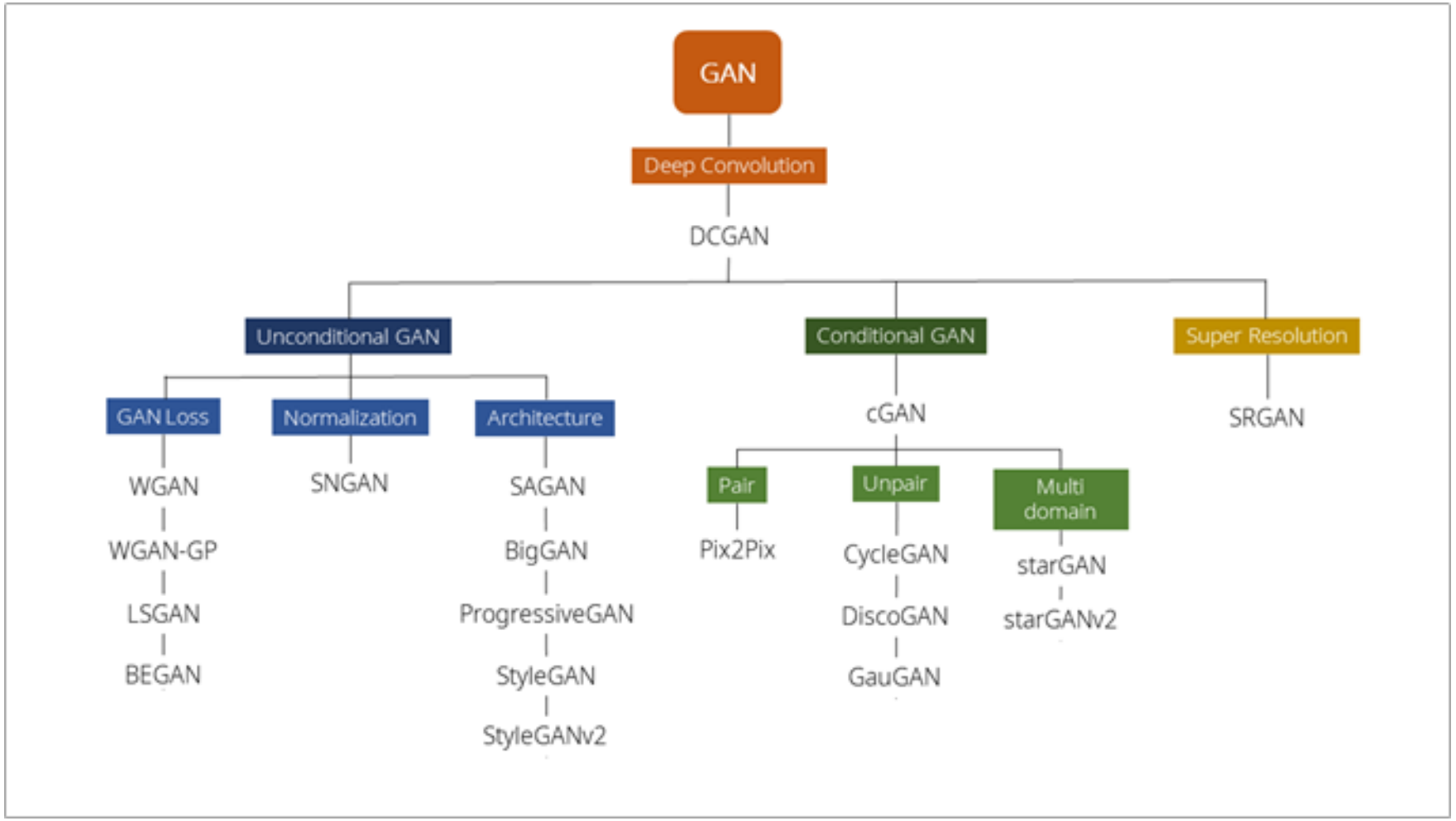
Samples from the model look much better!



Radford et al, ICLR 2016

05. GAN

Further Work

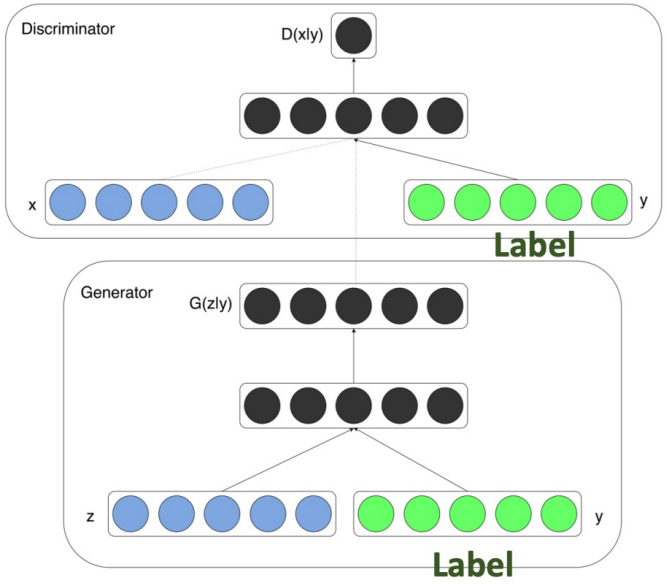


05. GAN

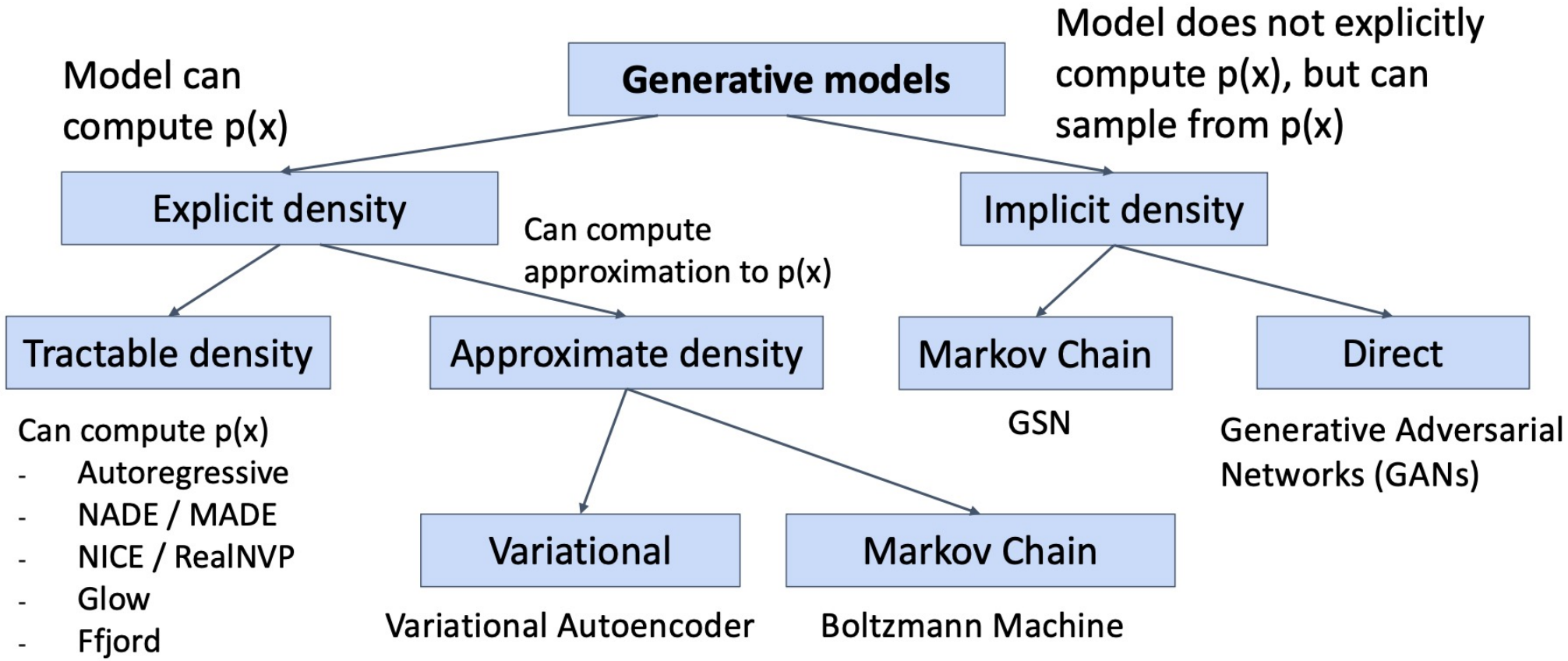
Further Work

- 그 중에서 conditional GAN (cGAN)
  - 지금 본건 unconditional GAN – learn p(x)
  - learn p(x|y)?

- New concept : Diffusion Model..



06. Summary


Generative models
Model can compute p(x)
Model does not explicitly compute p(x), but can sample from p(x)
Explicit density
Implicit density
Can compute approximation to p(x)
Tractable density
Approximate density
Markov Chain
Direct
Can compute p(x)
- Autoregressive
- NADE / MADE
- NICE / RealNVP
- Glow
- Ffjord
Variational
Markov Chain
GSN
Generative Adversarial Networks (GANs)
Variational Autoencoder
Boltzmann Machine

06. Summary

Autoregressive Models directly maximize likelihood of training data:

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

Variational Autoencoders introduce a latent z, and maximize a lower bound:

$$p_\theta(x) = \int_Z p_\theta(x|z)p(z)dz \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

Generative Adversarial Networks give up on modeling p(x), but allow us to draw samples from p(x)

Reference

- eecs 498-007 2019 lecture 19, 20
- https://www.youtube.com/watch?v=z1k8HVU4Mxc
- 7기 김예진, 6기 안민용 2022-2 세션 자료
- cs231n 2021 lecture 10
- https://doooob.tistory.com/173
- https://roytravel.tistory.com/109
- https://hyunw.kim/blog/2017/10/27/KL_divergence.html

DATA SCIENCE LAB

발표자 전재현 010-7211-9910
E-mail:chris990126@yonsei.ac.kr