Generative Adversarial Nets
(GANs)

Goodfellow et al., 2014
NIPS 9,000 citation

Dongjin Yoon

Data Science Lab.
Dept. Computer Engineering
Inha university

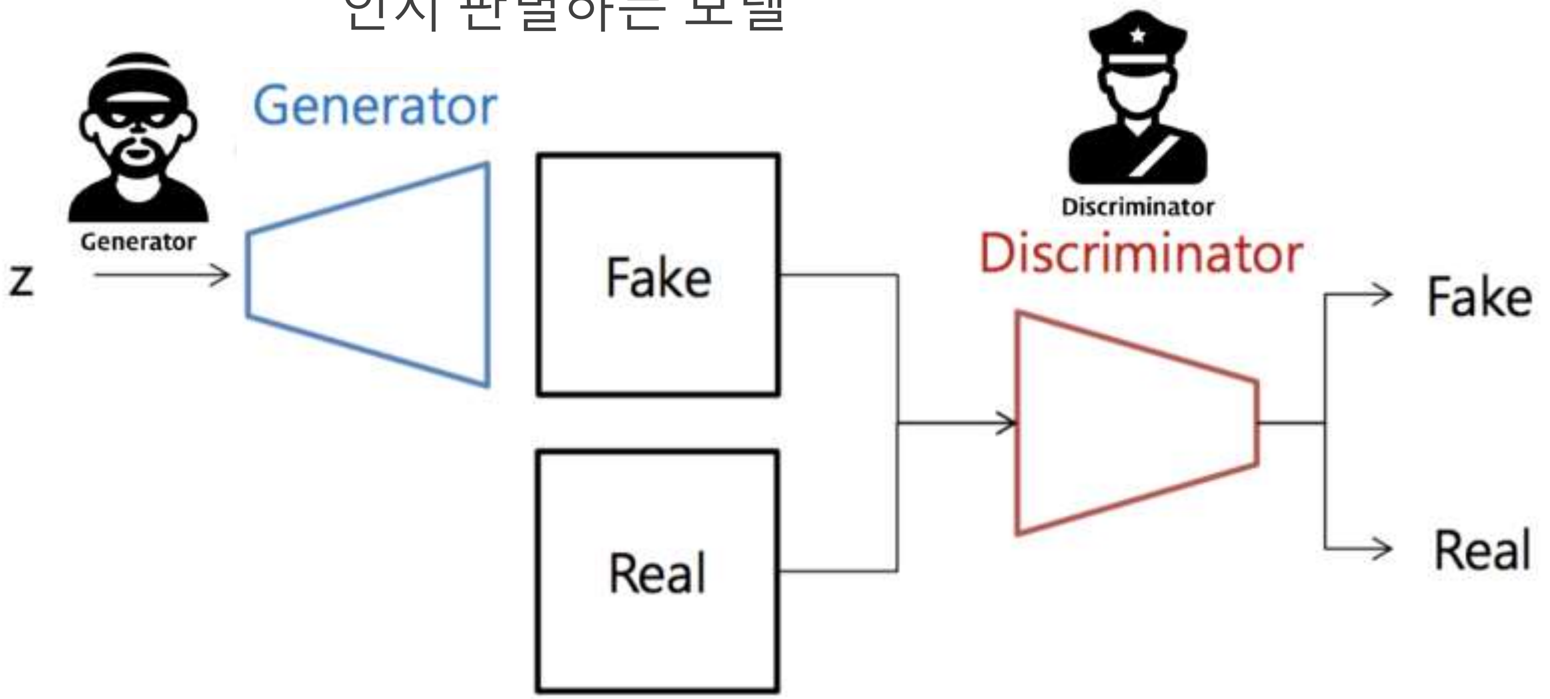
Table of Contents

1. Introduction

1. Introduction

- **생성모델(Generator)**   : 학습 데이터의 분포를 추정하여 가짜 데이터를 생성하는 모델

  판별모델(Discriminator): 주어진 데이터가 생성된 데이터(Fake)인지 학습 데이터(Real)인지 판별하는 모델

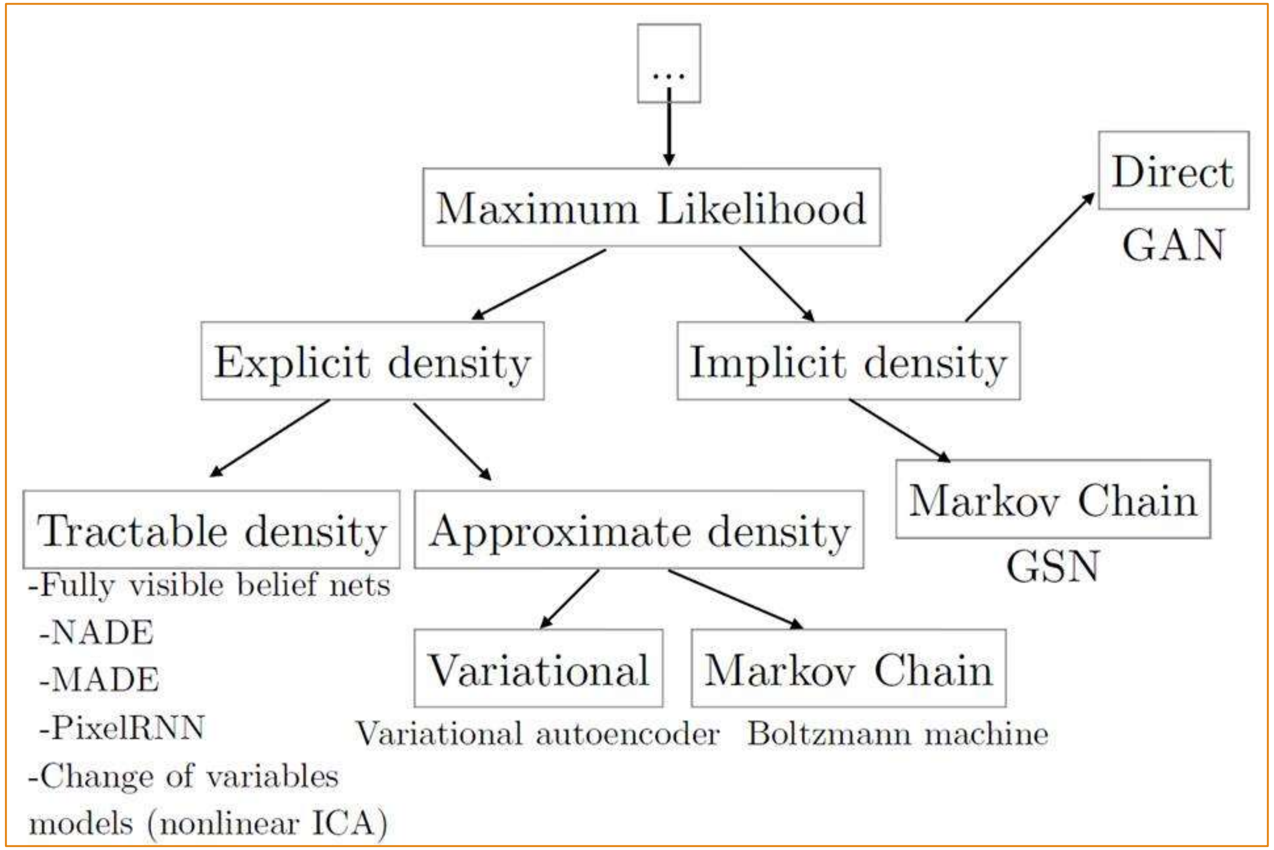


2. Related works & Motivations

2. Related works & Motivations

- **GANs**는 Maximum Likelihood Estimation(MLE)을 이용한 방법론의 하나이다.

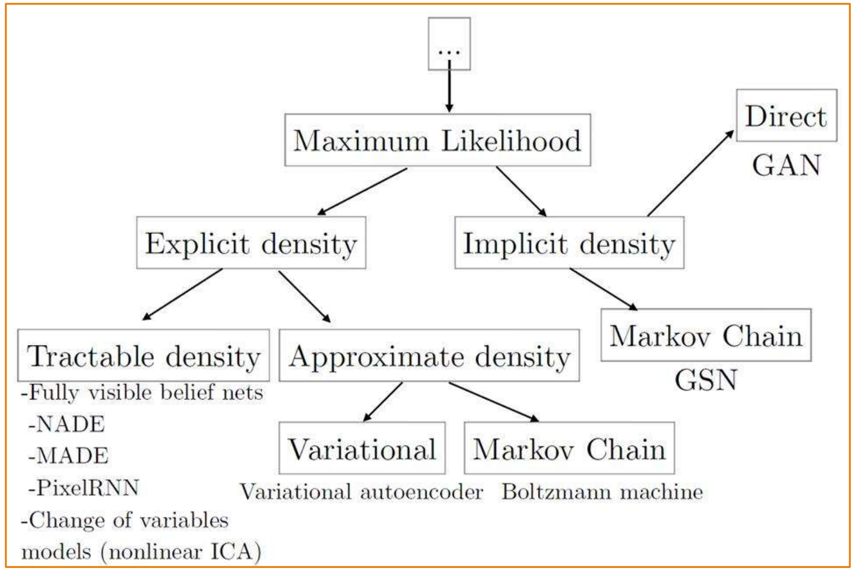
- 기존에 주로 사용되었던 **Fully Visible Belief Nets, Variational autoencoder, GSN** 등에서 나타난 문제점들을 해결할 수 있도록 설계되었으며 큰 성능향상을 보여 생성 모델의 발전에 큰 영향을 끼치게 되었다.



2. Related works & Motivations

- **Fully Visible Belief Nets(FVBNs)**
  - GAN과 Variational autoencoder와 함께 가장 유명한 생성모델
  - Data를 생성하는 분포를 직접적으로 나타내고 사용하는 방법론 (Explicit density)

$$p_{\text{model}}(\boldsymbol{x}) = \prod_{i=1}^{n} p_{\text{model}}\left(x_i \mid x_1, \ldots, x_{i-1}\right)$$



- **단점**
  - 한 번에 하나의 dimension만을 생성하여 생성속도가 느림
  - Data가 Sequential하게 생성되기 때문에 병렬처리가 불가능

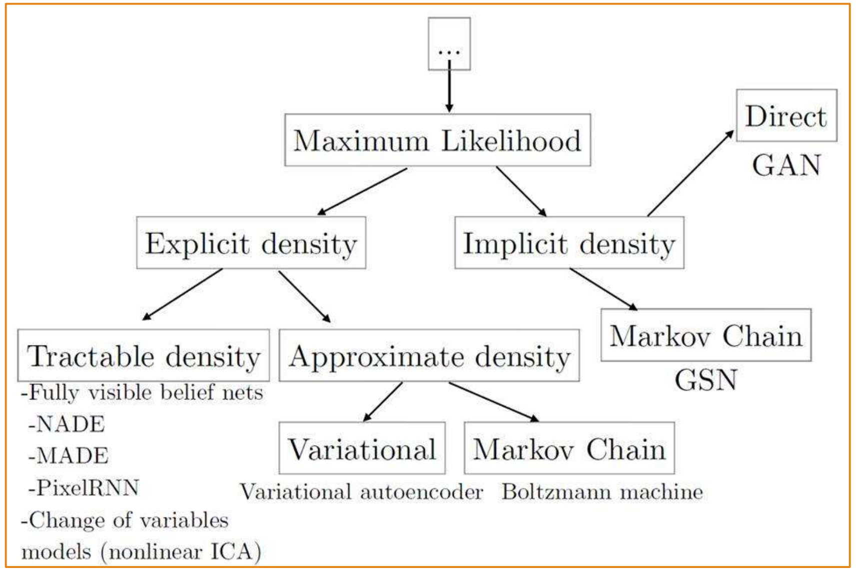
- **GANs:** 한 번에 하나의 data를 생성하며 병렬처리가 가능

2. Related works & Motivations

- **Variational autoencoder (VAE)**
  - 정확한 log-likelihood 대신 구하기 쉬운 lower bound를 구하고 그 값을 maximize하는 생성모델

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta}) \le \log p_{\text{model}}(\boldsymbol{x};\boldsymbol{\theta}).$$

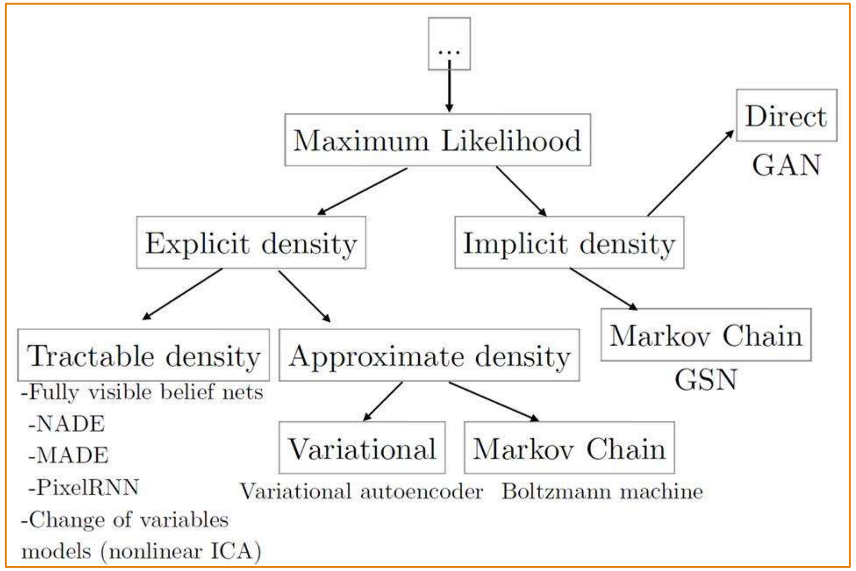
- **단점**
  - Lower bound와 log-likelihood의 차이로 인해 실제 data의 분포와는 다르게 학습될 수 있음
  - GANs과 비교하여 흐릿한 상을 생성



- **GANs:** lower bound를 필요로 하지 않고, VAE에 비해 좀 더 안정적이며 더 좋은 상을 생성

2. Related works & Motivations

- **Generative Stochastic Network (GSN)**
  - Data를 생성하는 분포를 직접적으로 학습시키는 것이 아니라, 생성 모델을 학습시키는 방법론 (Implicit density)
  - Markov chain transition operator를 정의하여 이를 반복 실행함 으로써 data를 생성

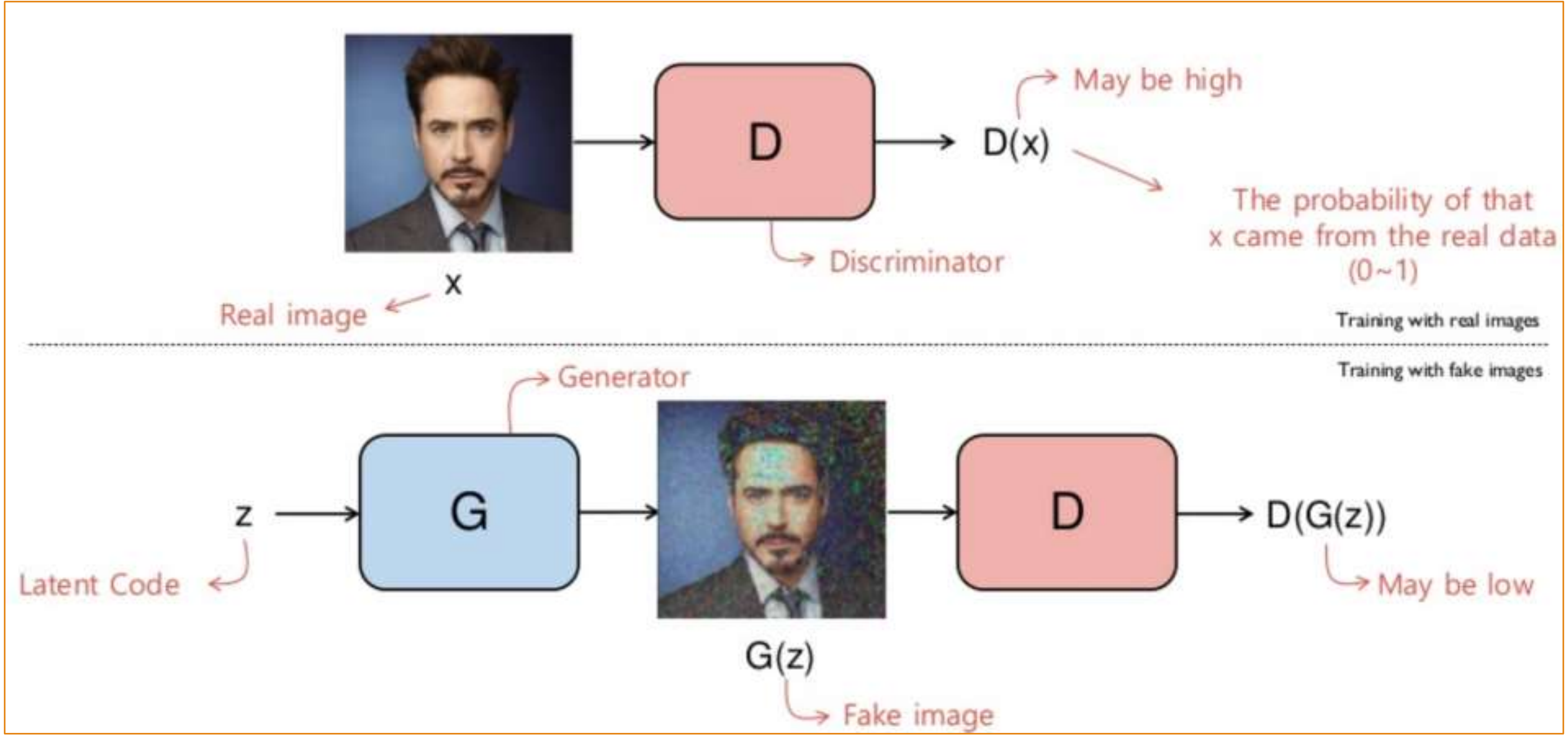


- **단점**
  - Markov chain은 high dimension space에서 제대로 작동하지 않는 경우가 많고, 계산비용도 크기 때문에 생성모델로 사용하기에는 비효율적

- **GANs:** Markov chain이 전혀 사용되지 않음

3. Adversarial Nets

3. Adversarial Nets



3. Adversarial Nets

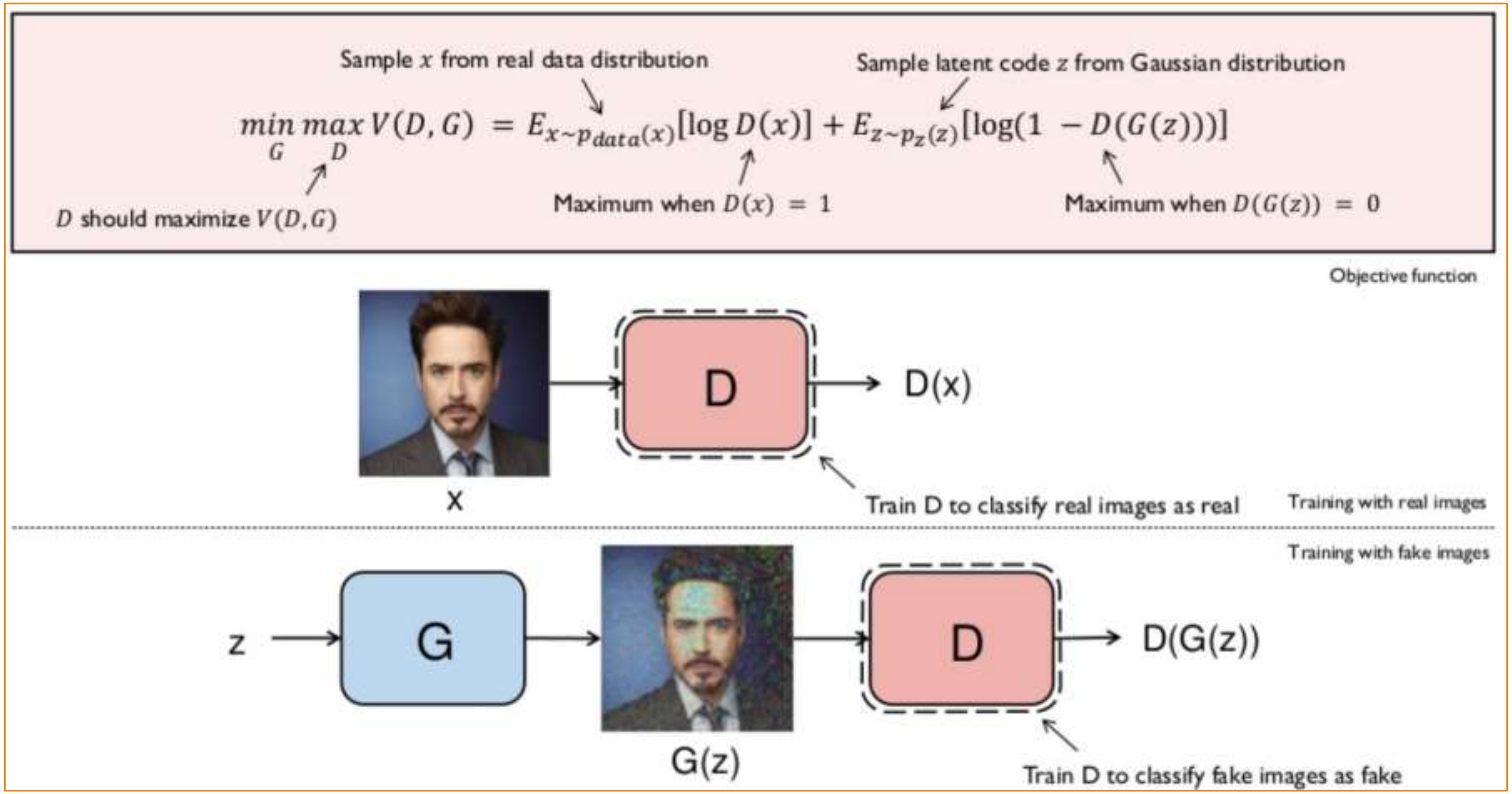
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Discriminator D
D가 진짜 data를 진짜라고 판별해야 하고 (max log D(x)),
D가 가짜 data를 가짜라고 판별해야 한다 (max log (1 – D(G(z))))

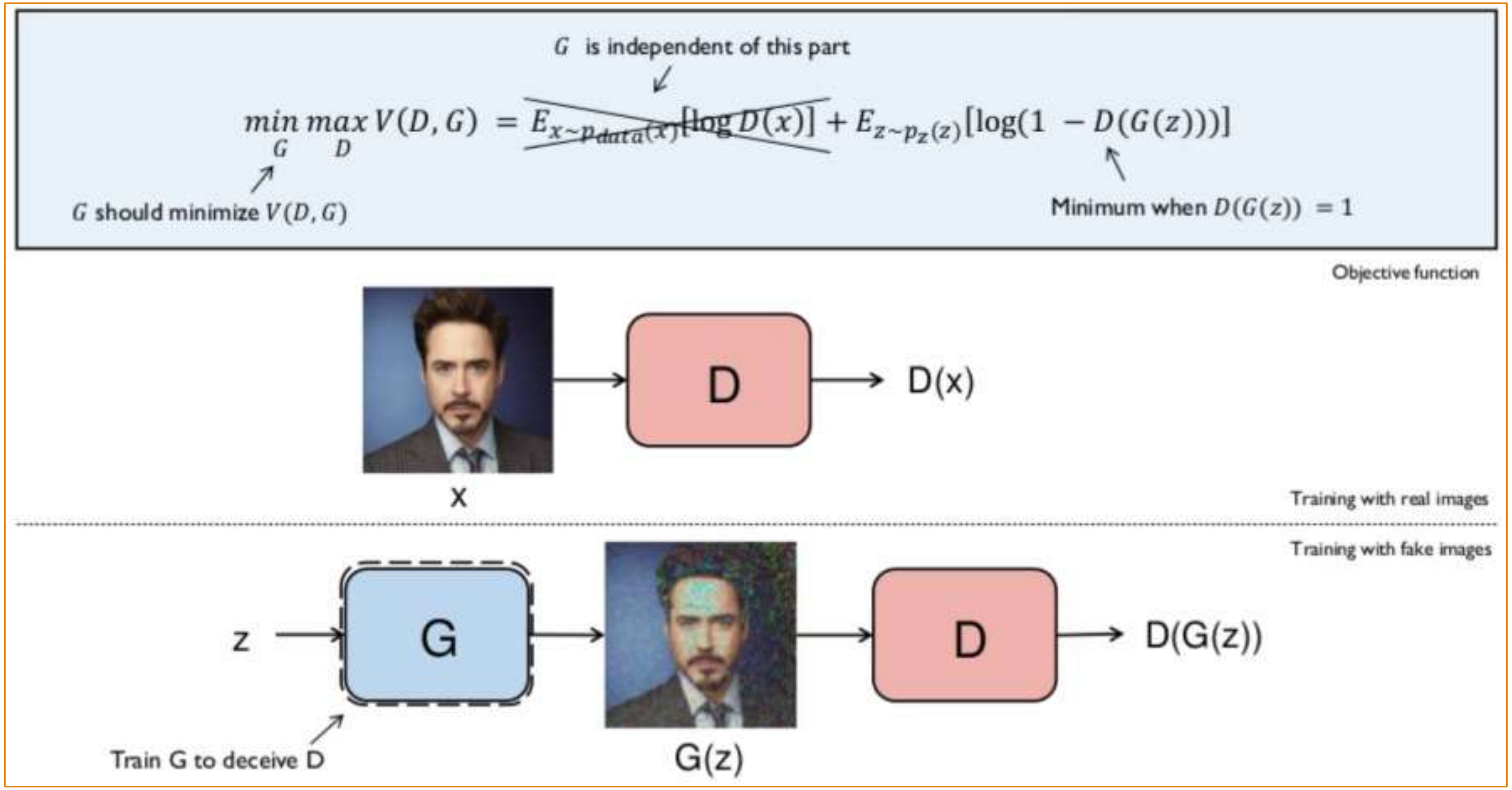
Generator G
D가 가짜 data를 진짜라고 판별해야 한다 (min log (1 – D(G(z))))

D와 G는 value function V(G, D)에 대해 two-player minimax game을 한다

3. Adversarial Nets – Train Discriminator

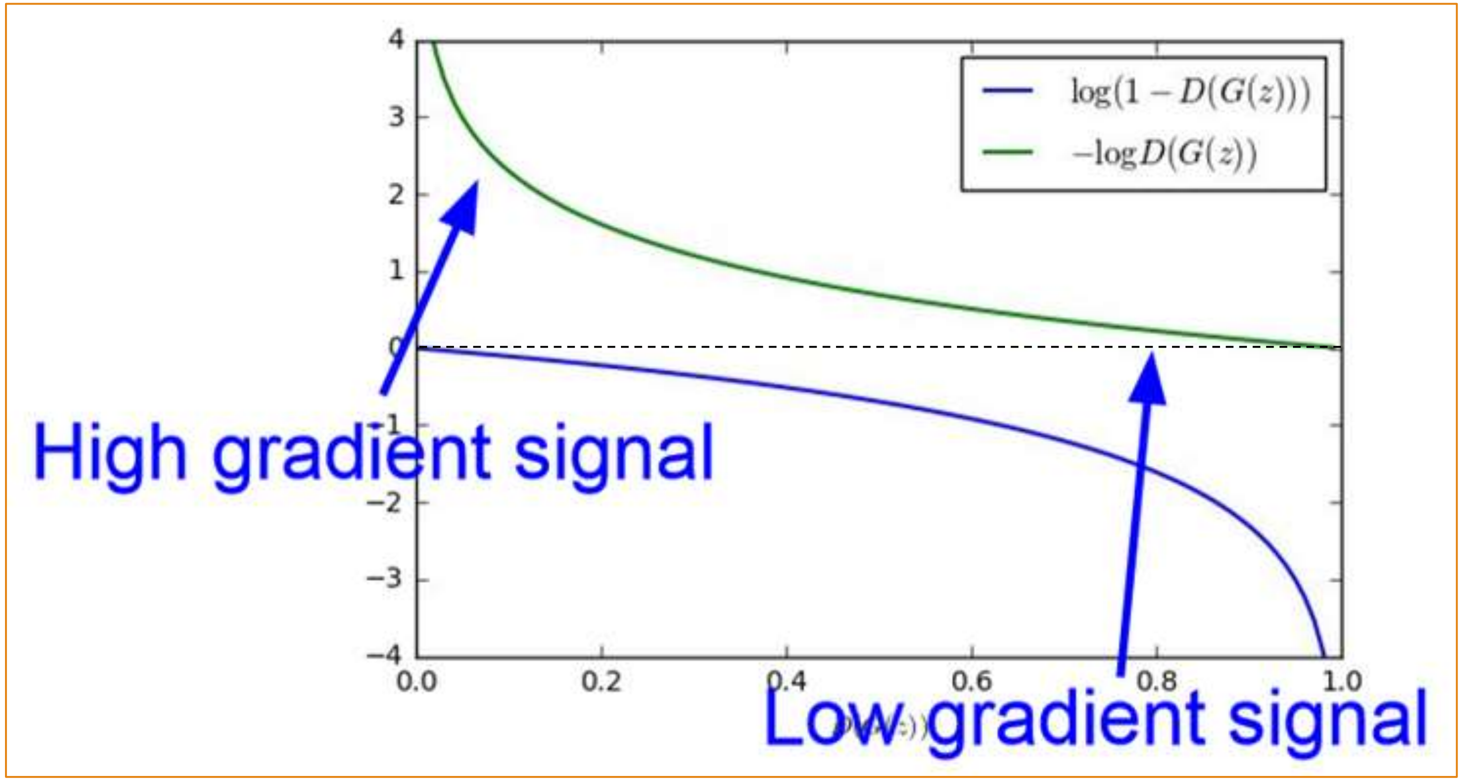


3. Adversarial Nets – Train Generator



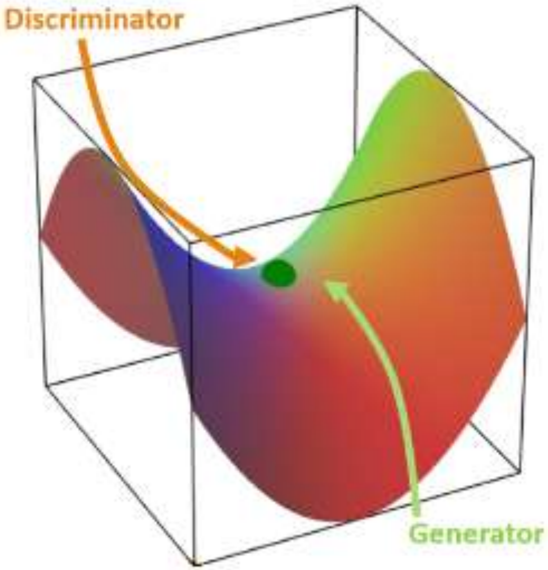
3. Adversarial Nets – Train Generator

- 학습 초기에는 $D(G(z))$가 많이 작기 때문에 $\log\left(1 - D(G(z))\right)$ 대신 $-\log D(G(z))$를 사용



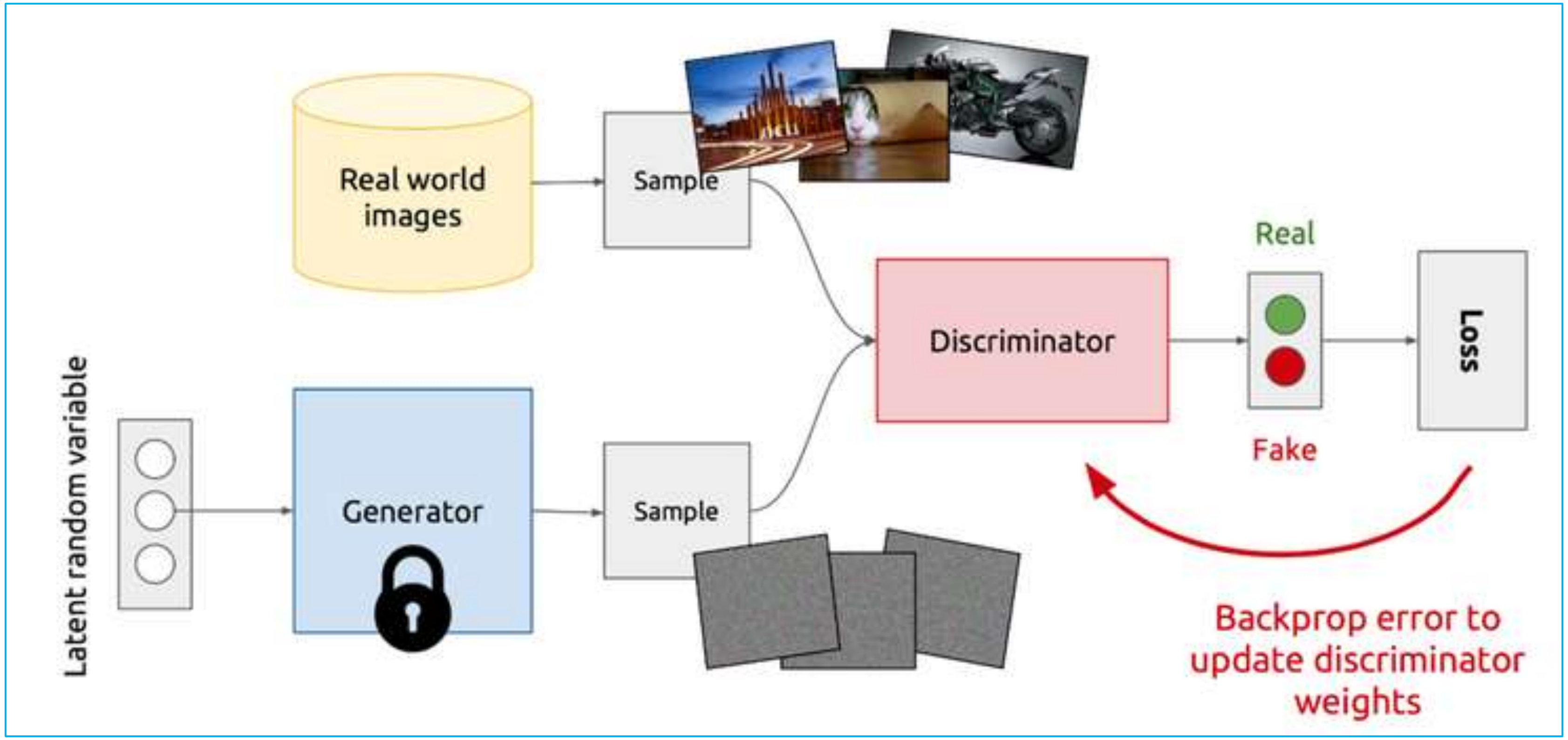
3. Adversarial Nets – Algorithm1

- 학습 방향이 정해지더라도 2개의 결합된 network를 학습시키기 위해서는 기존의 방법과는 다른 방법이 학습 방법이 필요
- 본 논문은 2개의 network를 번갈아 가며 학습시키는 방법을 사용



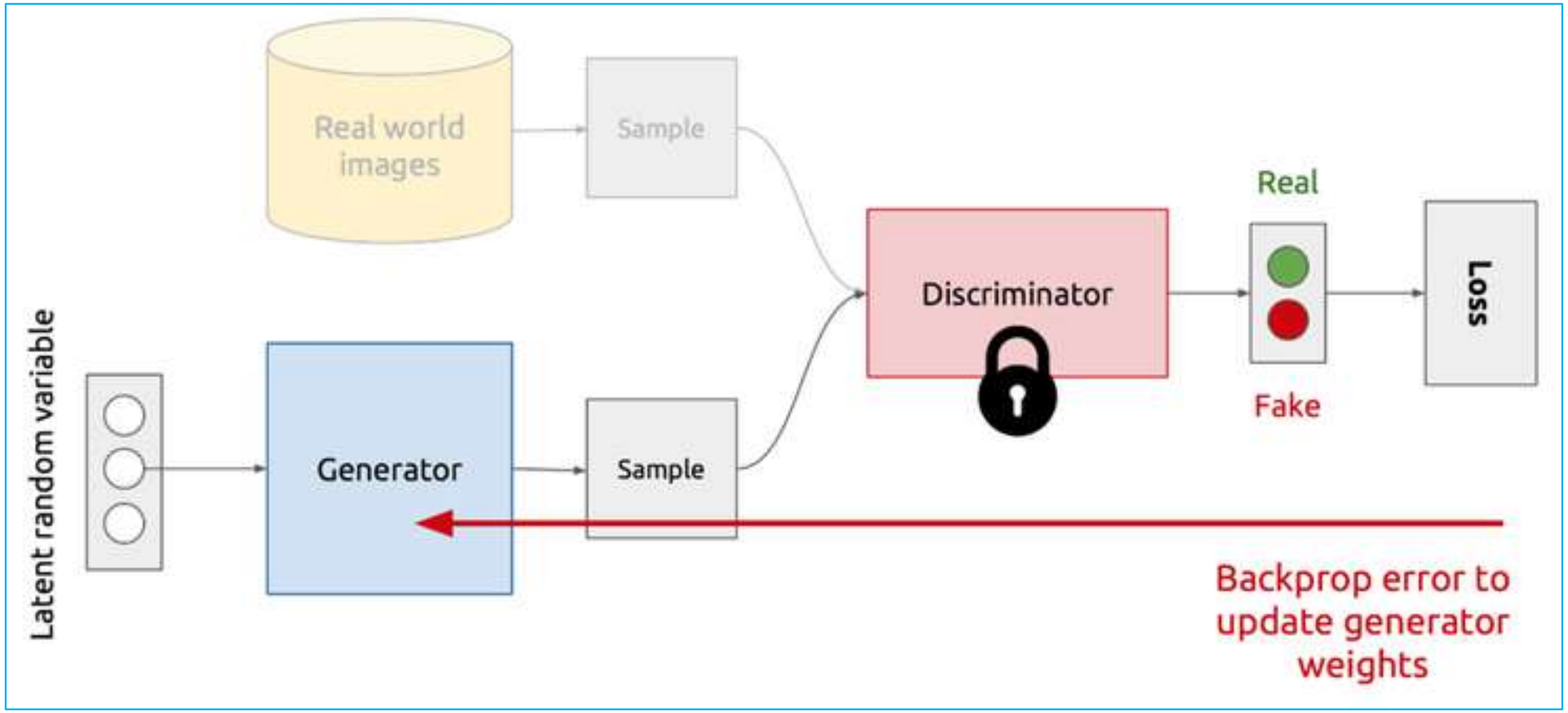
3. Adversarial Nets – Algorithm1

첫 번째, Generator를 고정시키고 Discriminator를 학습



3. Adversarial Nets – Algorithm1

두 번째, Discriminator를 고정시키고 Generator를 학습



3. Adversarial Nets – Algorithm1

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

for number of training iterations **do**

Train Discriminator by Gradient Ascent

for k steps **do**

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

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

end for

Train Generator by Gradient Descent

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

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

end for

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

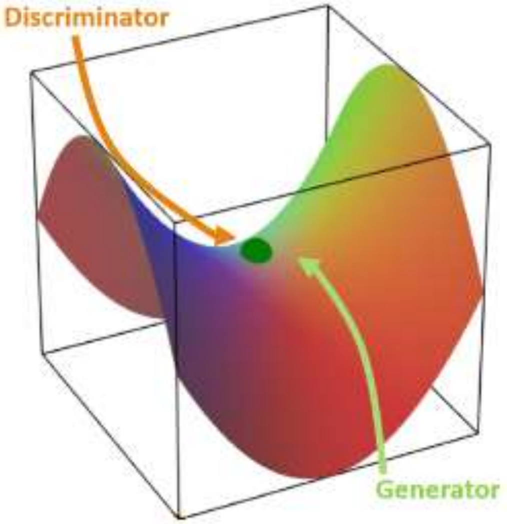
4. Theoretical Results

4. Theoretical Results

- 2가지 증명이 필요

- **1) Theorem 1**
  $p_g = p_{data}$ 인 경우에서, $\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$ 의 optimal값이 존재하고 그 값은 –log4 이다.

  2) Convergence of Algorithm1
  제시한 Algorithm1이 global optimum인 $p_g = p_{data}$ 로 수렴한다.



4.1 Global Optimality of $p_g = p_{data}$

high dimensional vector (e.g. 64 × 64)

low dimensional vector (e.g. 100)

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

Objective function of GANs

real data distribution

Gaussian distribution

4.1 Global Optimality of $p_g = p_{data}$

$$\cancel{\min_G} \max_D V(D, \cancel{G}) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Fix G to make it to a function of D

$$D^*(x) = \arg\max_D V(D) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Optimal D *Get D when V(D) is maximum*

4.1 Global Optimality of $p_g = p_{data}$

$$\cancel{\min_{G}} \max_{D} V(D,\cancel{G}) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Fix G to make it to a function of D

$$D^*(x) = \arg\max_{D} V(D) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

vector of 64 × 64 dimension

sampling x from p_g
instead of sampling z from p_z

vector of 100 dimension

$$= E_{x\sim p_{data}(x)}[\log D(x)] + E_{x\sim p_g(x)}[\log(1-D(x))]$$

model G distribution for high dimensional vector (e.g. 64 × 64)

4.1 Global Optimality of $p_g = p_{data}$

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

Fix G to make it to a function of D

$$D^*(x) = \arg\max_D V(D) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

sampling x from p_g
instead of sampling z from p_z

$$= E_{x\sim p_{data}(x)}[\log D(x)] + E_{x\sim p_g(x)}[\log(1 - D(x))]$$

Definition of Expectation

$$E_{x\sim p(x)}[f(x)] = \int_x p(x)f(x)dx$$

$$= \int_x p_{data}(x)\log D(x)\,dx + \int_x p_g(x)\log(1 - D(x))\,dx$$

Integrate for all possible x

4.1 Global Optimality of $p_g = p_{data}$

$$\cancel{\min_G} \max_D V(D, \cancel{G}) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Fix G to make it to a function of D

$$D^*(x) = \arg\max_D V(D) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

sampling x from p_g
instead of sampling z from p_z

$$= E_{x \sim p_{data}(x)}[\log D(x)] + E_{x \sim p_g(x)}[\log(1 - D(x))]$$

Definition of Expectation

$$E_{x \sim p(x)}[f(x)] = \int_x p(x) f(x) dx$$

$$= \int_x p_{data}(x) \log D(x)\, dx + \int_x p_g(x) \log\big(1 - D(x)\big)\, dx$$

Basic property of Integral

$$= \int_x p_{data}(x) \log D(x) + p_g(x) \log\big(1 - D(x)\big)\, dx$$

4.1 Global Optimality of $p_g = p_{data}$

$$\cancel{\min_G} \max_D V(D, \cancel{G}) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Fix G to make it to a function of D

$$D^*(x) = arg \max_D V(D) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

sampling x from p_g instead of sampling z from p_z

$$= E_{x \sim p_{data}(x)}[\log D(x)] + E_{x \sim p_g(x)}[\log(1 - D(x))]$$

Definition of Expectation

$$E_{x \sim p(x)}[f(x)] = \int_x p(x) f(x) dx$$

$$= \int_x p_{data}(x) \log D(x)\, dx + \int_x p_g(x) \log(1 - D(x))\, dx$$

Basic property of Integral

$$= \int_x \boxed{p_{data}(x) \log D(x) + p_g(x) \log(1 - D(x))}\, dx$$

Now we need to find D(x) which makes the funtion inside integral maximum.

4.1 Global Optimality of $p_g = p_{data}$

$$D^*(x) = \arg\max_D V(D)$$

$$= \arg\max_D \; p_{data}(x)\log D(x) + p_g(x)\log\big(1 - D(x)\big)$$

The funtion inside integral

4.1 Global Optimality of $p_g = p_{data}$

$$D^*(x) = \arg\max_D V(D)$$

$$= \arg\max_D \; p_{data}(x) \log D(x) + p_g(x) \log(1 - D(x))$$

Substitute $a = p_{data}(x)$, $y = D(x)$, $b = p_g(x)$

$$a \log y + b \log(1 - y)$$

4.1 Global Optimality of $p_g = p_{data}$

$$D^*(x) = \arg\max_D V(D)$$

$$= \arg\max_D \; p_{data}(x)\log D(x) + p_g(x)\log(1 - D(x))$$

Substitute $a = p_{data}(x),\ y = D(x),\ b = p_g(x)$

$$a\log y + b\log(1 - y)$$

Differentiate with respect to $D(x)$ *using* $\frac{d}{dx}\log f(x) = \frac{f'(x)}{f(x)}$

Note that $D(x)$ *can not affect to* $p_{data}(x)$ *and* $p_g(x)$.

$$\frac{a}{y} + \frac{-b}{1-y} = \frac{a-(a+b)y}{y(1-y)}$$

4.1 Global Optimality of $p_g = p_{data}$

$$D^*(x) = \arg\max_D V(D)$$

$$= \arg\max_D p_{data}(x) \log D(x) + p_g(x) \log(1 - D(x))$$

Substitute $a = p_{data}(x),\ y = D(x),\ b = p_g(x)$

$$a \log y + b \log(1 - y)$$

Differentiate with respect to $D(x)$ *using* $\frac{d}{dx} \log f(x) = \frac{f'(x)}{f(x)}$

Note that $D(x)$ *can not affect to* $p_{data}(x)$ *and* $p_g(x)$.

$$\frac{a}{y} + \frac{-b}{1-y} = \frac{a - (a+b)y}{y(1-y)}$$

Find the point where the derivative value is 0 (local extreme).

$$\frac{a - (a+b)y}{y(1-y)} = 0$$

It has a maximum value when $y = \frac{a}{a+b}$

Note that the local maximum is the global maximum when there are no other local extremes.

4.1 Global Optimality of $p_g = p_{data}$

$$D^*(x) = \arg\max_D V(D)$$

$$= \arg\max_D p_{data}(x) \log D(x) + p_g(x) \log(1 - D(x))$$

Substitute $a = p_{data}(x)$, $y = D(x)$, $b = p_g(x)$

$$a \log y + b \log(1 - y)$$

Differentiate with respect to $D(x)$ using $\frac{d}{dx} \log f(x) = \frac{f'(x)}{f(x)}$

Note that $D(x)$ can not affect to $p_{data}(x)$ and $p_g(x)$.

$$\frac{a}{y} + \frac{-b}{1-y} = \frac{a - (a+b)y}{y(1-y)}$$

Find the point where the derivative value is 0 (local extreme).

$$\frac{a - (a+b)y}{y(1-y)} = 0$$

It has a maximum value when $y = \frac{a}{a+b}$

Substitute $a = p_{data}(x)$, $y = D(x)$, $b = p_g(x)$

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

G가 고정되어 있을 때, $D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$

4.1 Global Optimality of $p_g = p_{data}$

$$\min_G \max_D V(D,G) = \min_G V(D^*,G)$$

$$\begin{aligned}
V(D^*,G) &= E_{x\sim p_{data}}[\log D^*(x)] + E_{x\sim p_g}[\log(1-D^*(x))] \\
&= \int_x p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}dx + \int_x p_g(x)\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}dx \\
&= -log4 + log4 + \int_x p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}dx + \int_x p_g(x)\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}dx \\
&= -log4 + \int_x p_{data}(x)\log\frac{2\cdot p_{data}(x)}{p_{data}(x)+p_g(x)}dx + \int_x p_g(x)\log\frac{2\cdot p_g(x)}{p_{data}(x)+p_g(x)}dx \\
&= -log4 + KL(p_{data}||\frac{p_{data}+p_g}{2}) + KL(p_g||\frac{p_{data}+p_g}{2}) \\
&= -log4 + 2\cdot JSD(p_{data}||p_g)
\end{aligned}$$

두 분포의 Jensen-Shannon divergence (JSD)는 항상 0 또는 양수이고,
두 분포가 같을 때 0으로 최솟값을 가진다.

4.1 Global Optimality of $p_g = p_{data}$

$$\min_G \max_D V(D,G) = \min_G V(D^*,G)$$

Optimal D

$$V(D^*,G) = E_{x\sim p_{data}}[\log D^*(x)] + E_{x\sim p_g}[\log(1-D^*(x))]$$

G should minimize

$$= \int_x p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}dx + \int_x p_g(x)\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}dx$$

$$= -log4 + log4 + \int_x p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}dx + \int_x p_g(x)\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}dx$$

$$= -log4 + \int_x p_{data}(x)\log\frac{2\cdot p_{data}(x)}{p_{data}(x)+p_g(x)}dx + \int_x p_g(x)\log\frac{2\cdot p_g(x)}{p_{data}(x)+p_g(x)}dx$$

$$= -log4 + KL(p_{data}||\frac{p_{data}+p_g}{2}) + KL(p_g||\frac{p_{data}+p_g}{2})$$

$$= -log4 + 2\cdot JSD(p_{data}||p_g)$$

Optimizing $V(D,G)$ is same as minimizing $JSD(p_{data}||p_g)$

G should minimize

결론: $p_g = p_{data}$인 경우에서 최적의 Generator와 Discriminator가 구해진다.

4.2 Convergence of Algorithm 1

- **Proposition**

  G와 D가 충분한 표현력(capacity)이 있다면, Algorithm1의 매 step마다 주어진 G에 대하여 D는 최적해에 가까워진다.

  또한 p_g는 아래의 기준값을 개선시키는 방향으로 update되므로 p_g는 p_{data}로 수렴한다.

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$$

- **즉, 제시한 알고리즘이 global optimum인 $p_g = p_{data}$ 로 수렴하는지를 확인**

3.2 Convergence of Algorithm 1

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

Proof. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha\in\mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta =$ a
This is equivalent to computing a gradient descent update for p_g at the optimal
responding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as
therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the

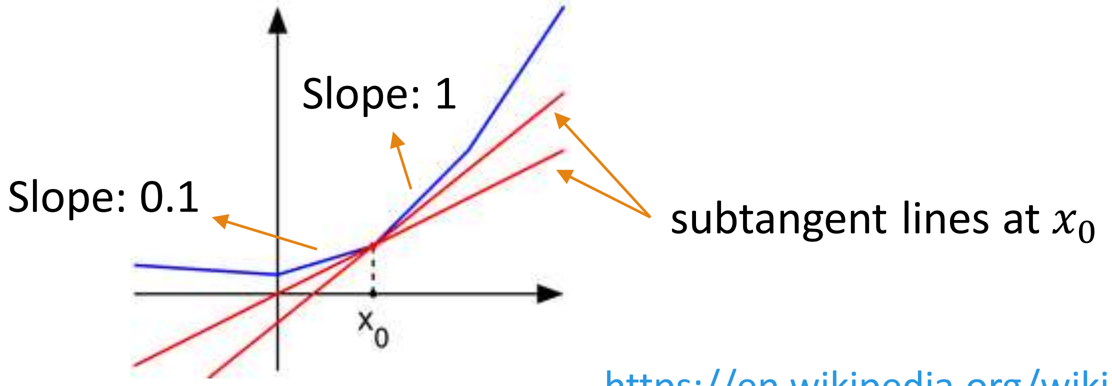
3.2 Convergence of Algorithm 1

- **Subderivatives(subgradient), Subdifferential**

  반드시 미분이 가능할 필요가 없는 convex function에 대한 derivative를 일반화한 개념이다.

  함수의 정의역에 포함된 임의의 x_0에 대하여, 점 $(x_0, f(x_0))$를 지나가고 그래프에 닿거나 아래로만 (sub) 지나가는 line (subtangent line)을 그릴 수 있는데, 해당 line의 기울기를 **subderivative(subgradient)**라 한다.

  또한, x_0에서 left, right derivative를 각각 a, b라 하면, x_0에서 모든 subderivative들의 집합인 [a, b]를 x_0에서 함수의 **subdifferential**이라 한다.



https://en.wikipedia.org/wiki/Subderivative

3.2 Convergence of Algorithm 1

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))]$$

then p_g converges to p_{data}

1

Proof. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha\in\mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha\in\mathcal{A}} f_\alpha(x)$. This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. $\square$

1. $U(p_g, D) = \int_x p_{data}(x)log(D(x)) + p_g(x)log(1 - D(x))dx$

variable *constant*

3.2 Convergence of Algorithm 1

Proposition 2. *If G and D have enough capacity, and at each step of Algorithm 1, the discriminator is allowed to reach its optimum given G, and p_g is updated so as to improve the criterion*

$$\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D^*_G(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D^*_G(\boldsymbol{x}))]$$

then p_g converges to p_{data}

2 *Proof.* Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $\boxed{U(p_g, D) \text{ is convex in } p_g.}$ The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha\in\mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha\in\mathcal{A}} f_\alpha(x)$. This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. □

2. $\boldsymbol{U(p_g, D) = \int_x \underline{p_{data}(x)\log(D(x)) + p_g(x)\log(1 - D(x))}dx}$

variable *constant*

$$\frac{\partial}{\partial p_g} Inside = \log(1 - D(x)) \text{ is nonpositive contant, so Inside is convex}$$

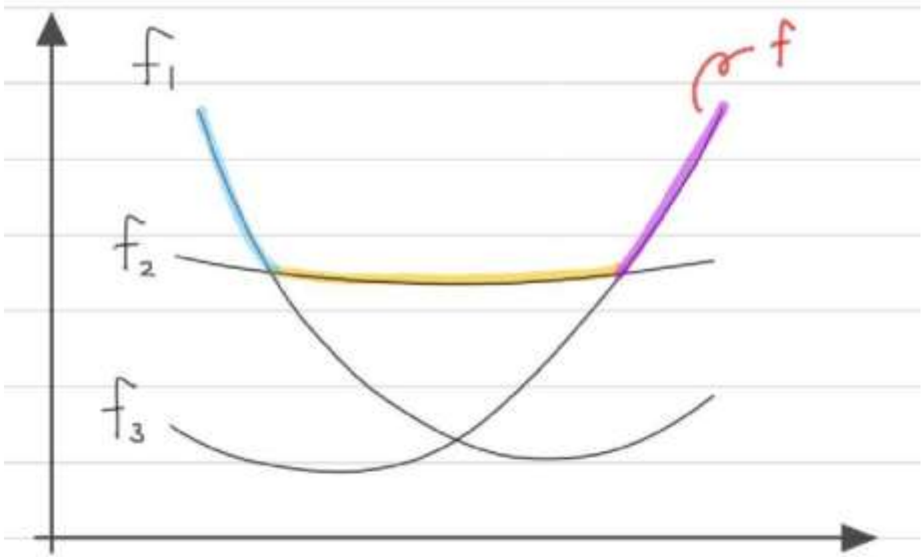
Thus, summation(∫) of convex is also convex

3.2 Convergence of Algorithm 1

Proof. Consider $V(G, D) = U(p_g, D)$ as a function of p_g as done in the above criterion. Note that $U(p_g, D)$ is convex in p_g. 3 The subderivatives of a supremum of convex functions include the derivative of the function at the point where the maximum is attained. In other words, if $f(x) = \sup_{\alpha \in \mathcal{A}} f_\alpha(x)$ and $f_\alpha(x)$ is convex in x for every α, then $\partial f_\beta(x) \in \partial f$ if $\beta = \arg\sup_{\alpha \in \mathcal{A}} f_\alpha(x)$.

3.

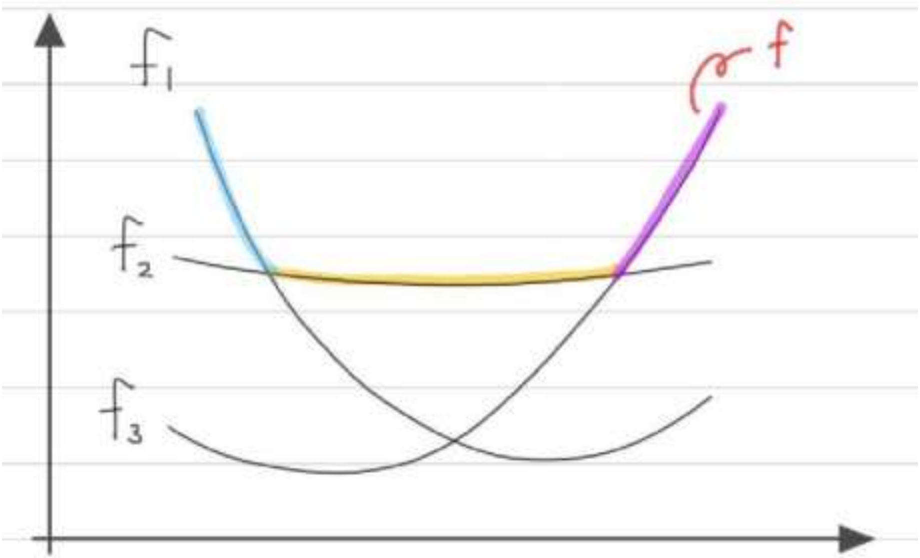
1) $f(x) \triangleq sup_\alpha f_\alpha(x)$ $s.t.\ f_\alpha$ *are convex on some convex domain.* $f(x)$ *is also convex on* D.
2) $f(x) = f_\beta(x)$ $s.t.\ \beta = argsup_\alpha f_\alpha(x)$
3) *Let* g *be any subgradient of* $f_\beta(x)$ $i.e.\ g \in \partial f_\beta$
4) *Thus,* g *is also a sugradient of* $f(x)$ $i.e.\ g \in \partial f$

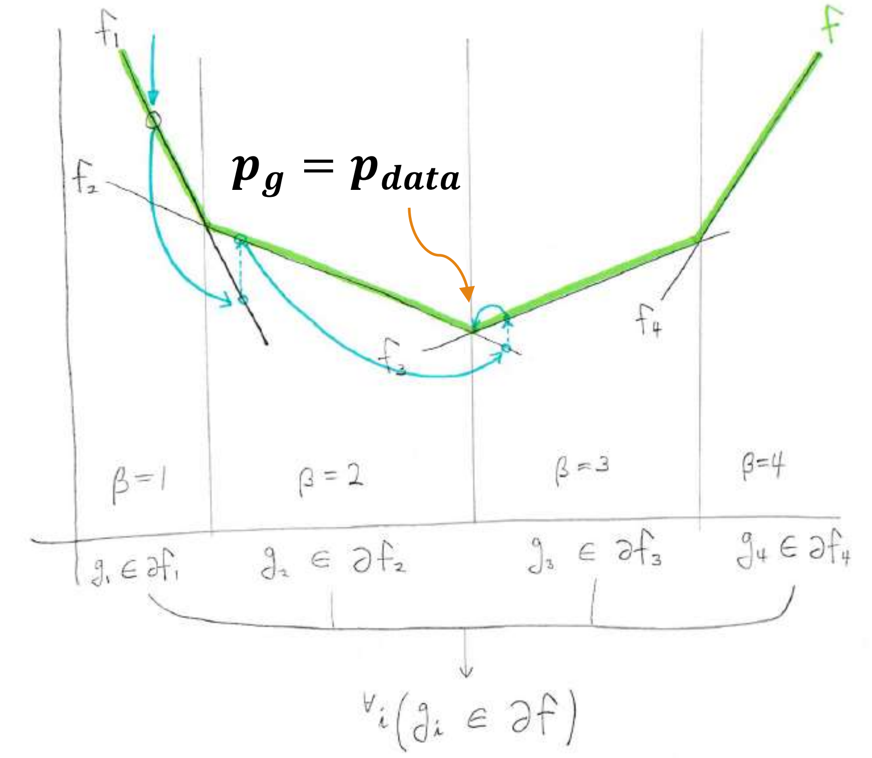


3.2 Convergence of Algorithm 1

4

This is equivalent to computing a gradient descent update for p_g at the optimal D given the corresponding G. $\sup_D U(p_g, D)$ is convex in p_g with a unique global optima as proven in Thm 1, therefore with sufficiently small updates of p_g, p_g converges to p_x, concluding the proof. $\square$





3.2 Convergence of Algorithm 1

- **결론**

  Algorithm1을 사용하여 Generator와 Discriminator를 optimal solution으로 수렴시킬 수 있다.

  그런데.. 이제까지의 증명들은 모두 non-parametric setting을 사용한 것

- 그러나, 실제로 MLP를 사용했을 때 뛰어난 성능을 보여주는 것은 이론적인 보장이 부족함에도 불구하고 사용할 만한 합리적인 모델임을 말해준다.

5. Experiments

5. Experiments

Random noise z: [batch_size, 128]

```
def generator(noise_z): # 128 -> 256 -> 28*28
    hidden = tf.nn.relu(tf.matmul(noise_z, G_W1) + G_b1)
    output = tf.nn.sigmoid(tf.matmul(hidden, G_W2) + G_b2)  # [0, 1] WB image
    return output
```

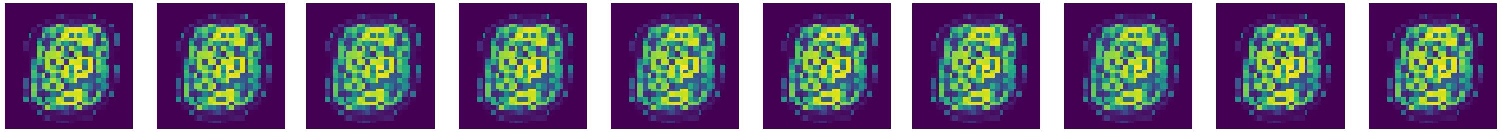
```
def discriminator(inputs): # 28*28 -> 256 -> 1
    hidden = tf.nn.relu(tf.matmul(inputs, D_W1) + D_b1)
    output = tf.nn.sigmoid(tf.matmul(hidden, D_W2) + D_b2)  # [0, 1] prob
    return output
```

```
loss_D = -tf.reduce_mean(tf.log(discriminator(X)) + tf.log(1 - discriminator(G)))
loss_G = -tf.reduce_mean(tf.log(discriminator(G)))
```

5. Experiments

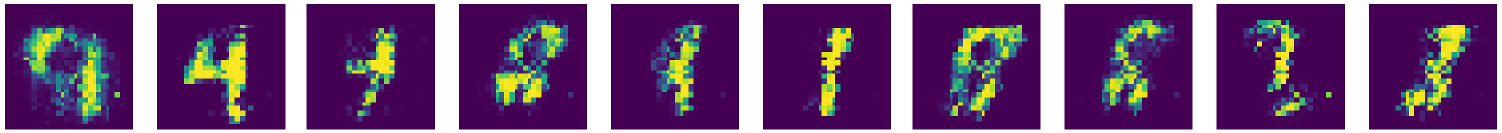
```python
noise_test = np.random.normal(size=(10, 128)) # 10 = Test Sample Size, 128 = Noise Dimension
for epoch in range(200): # 200 = Num. of Epoch
    for i in range(int(mnist.train.num_examples / 100)): # 100 = Batch Size
        batch_xs, _ = mnist.train.next_batch(100)
        noise = np.random.normal(size=(100, 128))

        sess.run(train_D, feed_dict={X: batch_xs, Z: noise})
        sess.run(train_G, feed_dict={Z: noise})

    if epoch == 0 or (epoch + 1) % 10 == 0: # 10 = Saving Period
        samples = sess.run(G, feed_dict={Z: noise_test})
```

5. Experiments



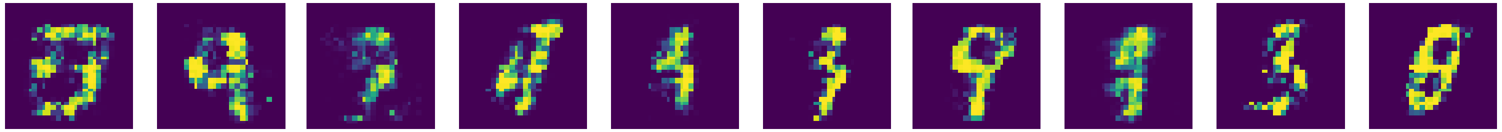
1 Epoch

5. Experiments



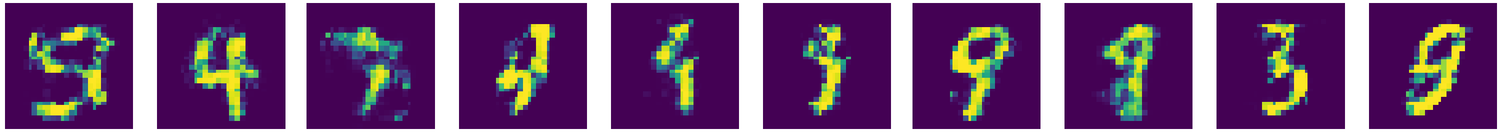
20 Epochs

5. Experiments



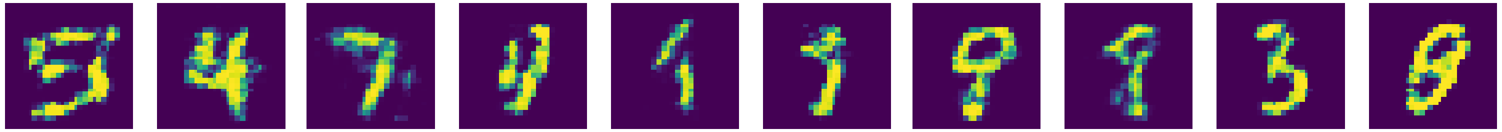
40 Epochs

5. Experiments



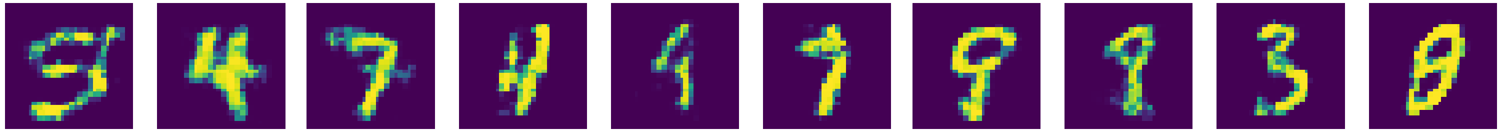
60 Epochs

5. Experiments



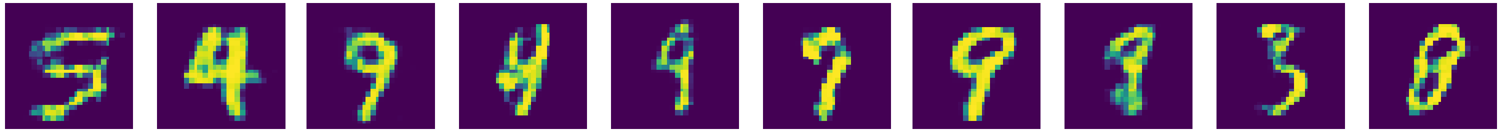
80 Epochs

5. Experiments

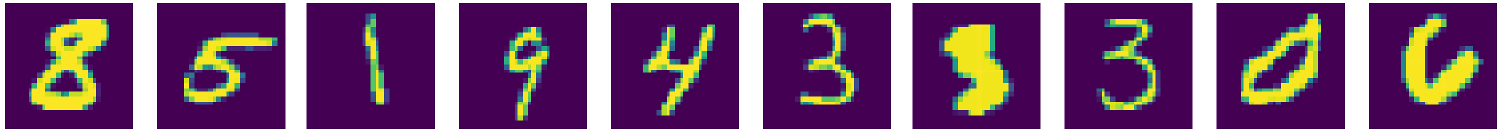


100 Epochs

5. Experiments



190 Epochs



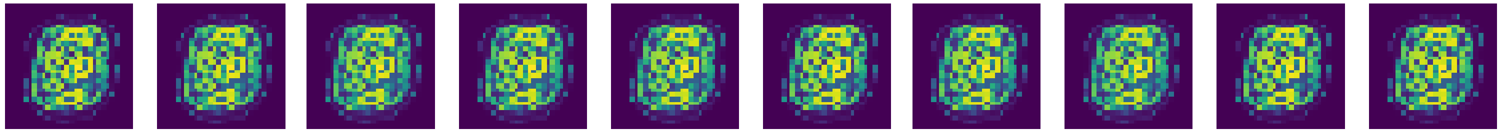
Real Image

5. Experiments

Input random noise

← Z: [28*28]

1 Epochs



← Z: [128]

5. Experiments

10 Epochs

5. Experiments

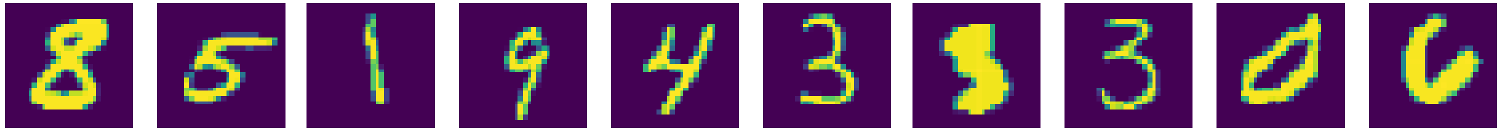
20 Epochs

5. Experiments

30 Epochs

5. Experiments

40 Epochs



Real Image

6. Advantages and Disadvantages

6. Advantages and Disadvantages

- **단점**

1. $p_g(x)$를 명시적으로 표현할 수 없다.

2. Discriminator가 학습 중에 Generator와 동기화가 잘 되어야 한다.

- **장점**

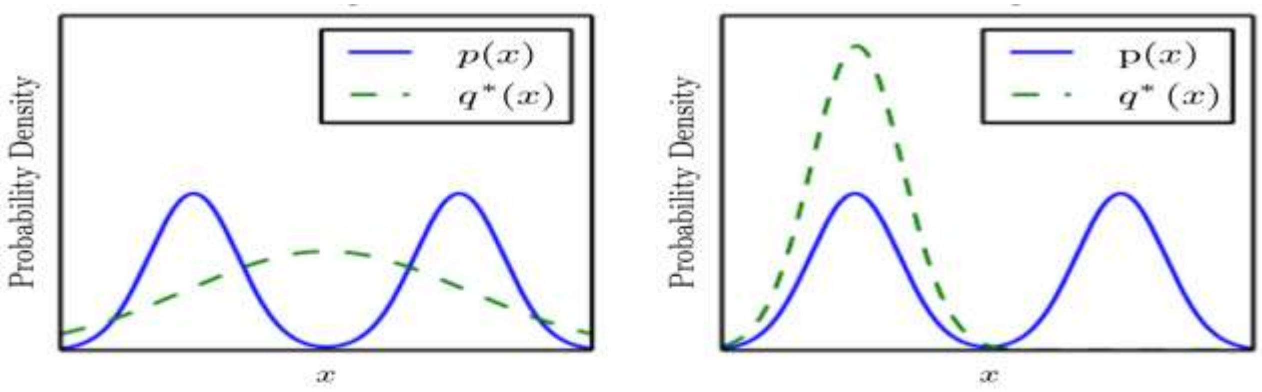
1. 학습 중에 inference가 필요하지 않다.

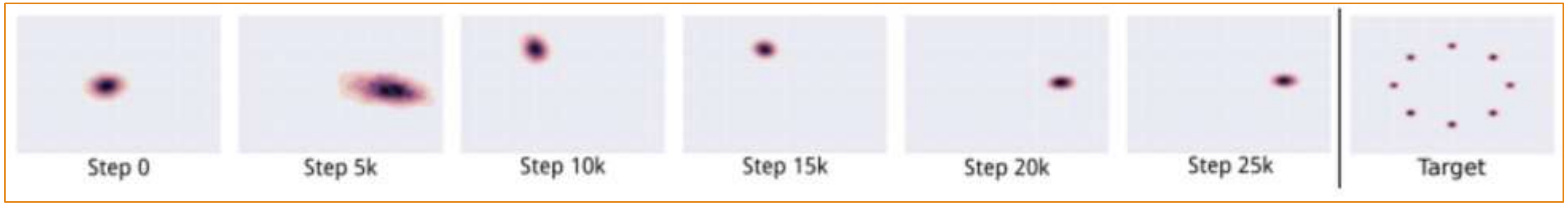
2. Gradient를 구하기 위해서 Markov chain이 필요하지 않다. (Only Backpropagation)

3. 다양한 함수들을 모델에 결합시킬 수 있다.

6. Advantages and Disadvantages

- **Mode collapsing**





7. Future works

7. Future works

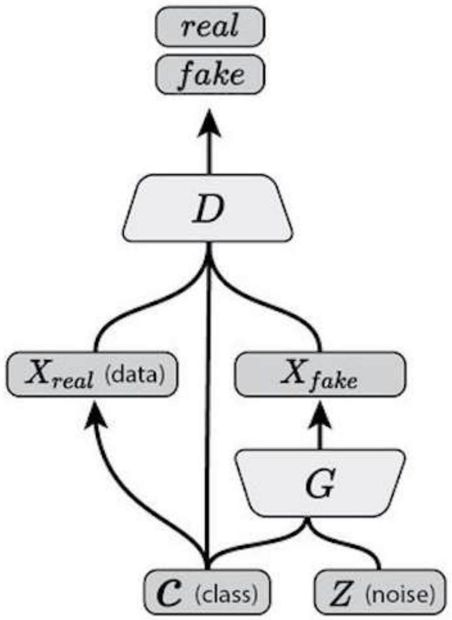
- GAN이 발표된 지 1년 후, 2015년 Deep Convolutional GAN (DCGAN)이 소개되어

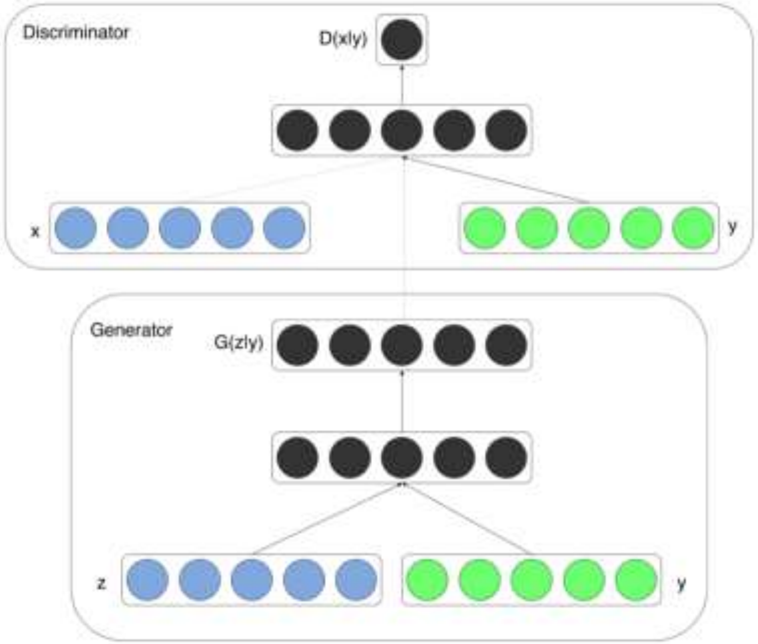
  - All Convolutional Network
  - Fully-Connected Layer를 제거
  - Batch Normalization을 사용
  - 다른 조합의 Activation fn을 사용

  몇 가지 디자인을 통해 GAN에 비해 큰 가시적인 성능향상을 불러 일으켰다.
- 학습의 안정성 문제를 해결하기 위해 최근엔 손실함수를 바꿔서 이 문제를 해결했다. (LSGAN, WGAN, F-GAN, EBGAN)
- 또한, Class까지 구별할 수 있는 CatGAN도 등장했다.

7. Future works

- **Conditional GAN (M Mirza et al., ICLR 2016)**
  Data/Noise 뿐만 아니라 정답 label을 Input으로 추가하여 원하는 label의 data를 생성시킬 수 있는 최초의 GAN

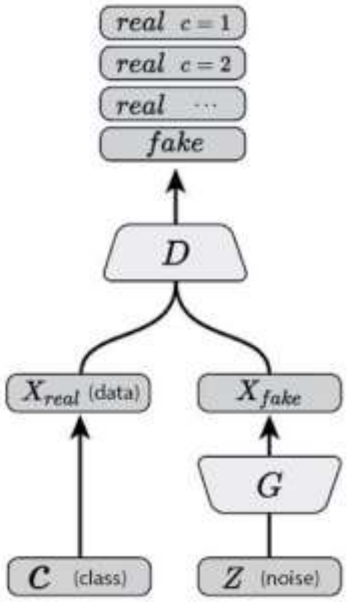




$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x,y)] + E_{z\sim p_z(z)}[\log\{1-D(G(z,y),y)\}]$$

7. Future works

- **Semi-supervised GAN (A Odena et al., ICML 2016)**
  Discriminator를 multinomial classifier로 사용하여 정답이 있는 real data, 정답이 없는 real data, fake data 모두 학습이 가능한 semi-supervised 환경에서도 작동할 수 있는 GAN



$$L_{D_{sup}} = -E_{x,y \sim p_{data}}[log(p_{model}(y = i \mid x, i < k + 1)]$$
$$L_{D_{uns}} = -E_{x \sim p_{data}}[log(1 - p_{model}(y = k + 1 \mid x)] - E_{x \sim p_g}[log(p_{model}(y = k + 1 \mid x)]$$
$$\boldsymbol{L_D = L_{D_{sup}} + L_{D_{uns}}}$$

$$L_{G_{featurematching}} = ||E_{x \sim p_{data}} f(x) - E_{x \sim p_g} f(x)||_2^2$$
$$L_{G_{cross-entropy}} = -E_{x \sim p_g}[log(1 - p_{model}(y = k + 1 \mid x)]$$
$$\boldsymbol{L_G = L_{G_{featurematching}} + L_{G_{cross-entropy}}}$$

Thank you!

and Question?

References

- **Original paper**
  http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
  https://arxiv.org/abs/1701.00160

- **참고자료**
  https://imgur.com/6ZPdsM8
  https://dreamgonfly.github.io/2018/03/17/gan-explained.html
  https://www.slideshare.net/NaverEngineering/1-gangenerative-adversarial-network
  https://laonple.blog.me/221190581073
  https://ratsgo.github.io/generative%20model/2017/12/20/gan/
  http://jaejunyoo.blogspot.com/2017/01/generative-adversarial-nets-1.html
  https://skymind.ai/kr/wiki/generative-adversarial-network-gan
  https://math.stackexchange.com/questions/2226794/convergence-of-gans
  https://kangbk0120.github.io/articles/2017-08/tips-from-goodfellow