

Generative Adversarial Nets

Ian J. Goodfellow*, Jean Pouget-Abadie†, Mehdi Mirza, Bing Xu, David Warde-Farley,
Sherjil Ozair‡, Aaron Courville, Yoshua Bengio§
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Communications of the ACM, 2020

Chungang University
DMAIS Lab
Master Coarse Park Sae Joon

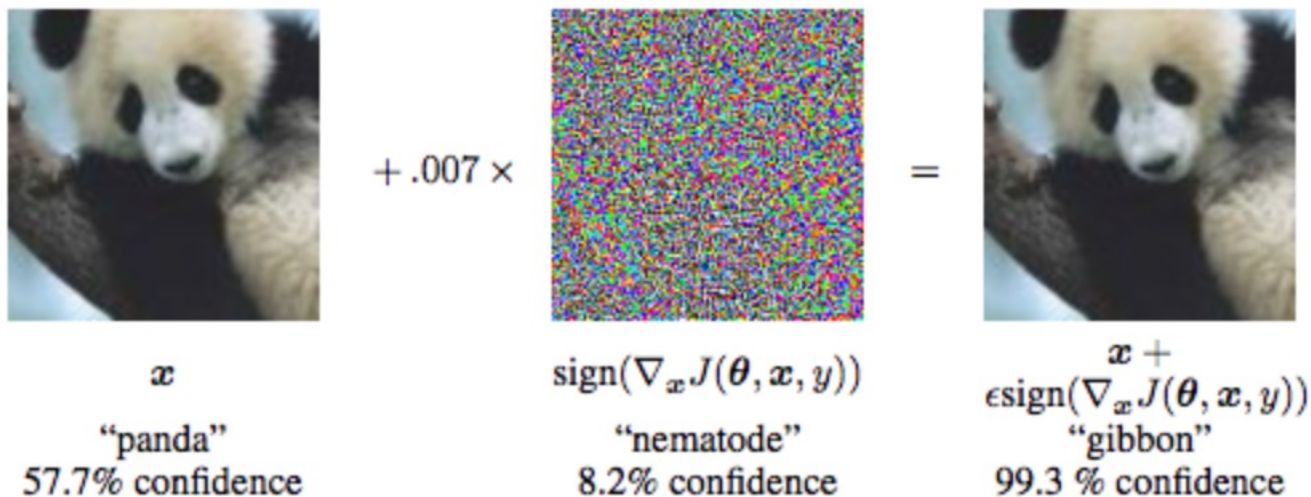
CONTENTS

- Background
- Previous Work
- Gan
- Experiments
- Pros And Cons
- Conclusions

BACKGROUND

● Adversarial Train

- 적대학습이라고 부름
- 데이터의 의도적인 노이즈를 줘 Discriminator가 구분하기 어렵도록 함

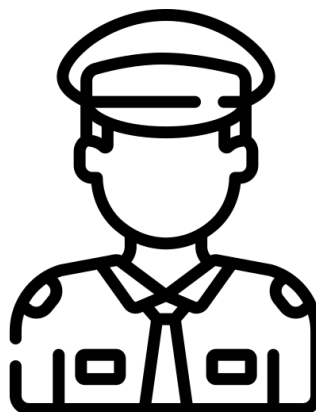


$$\begin{array}{ccc}
 \text{Image of a panda} & + .007 \times \text{Noise Image} & = \text{Adversarial Image} \\
 x & \text{sign}(\nabla_x J(\theta, x, y)) & x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \\
 \text{"panda"} & \text{"nematode"} & \text{"gibbon"} \\
 57.7\% \text{ confidence} & 8.2\% \text{ confidence} & 99.3\% \text{ confidence}
 \end{array}$$

BACKGROUND

- **Discriminator Based Model**

- 입력 데이터를 보고 특정 속성이나 레이블을 예측하는 모델



BACKGROUND

- **Generator Based Model**

- 생성 모델은 데이터 자체의 분포 $P(x)$ 를 학습하여 새로운 데이터를 생성하거나 샘플링하는 모델



BACKGROUND

● Markov Chain

- 마르코프 성질을 가진 이산시간 확률과정
- 현재 상태에서 다음상태를 결정하는 확률적 프로세스

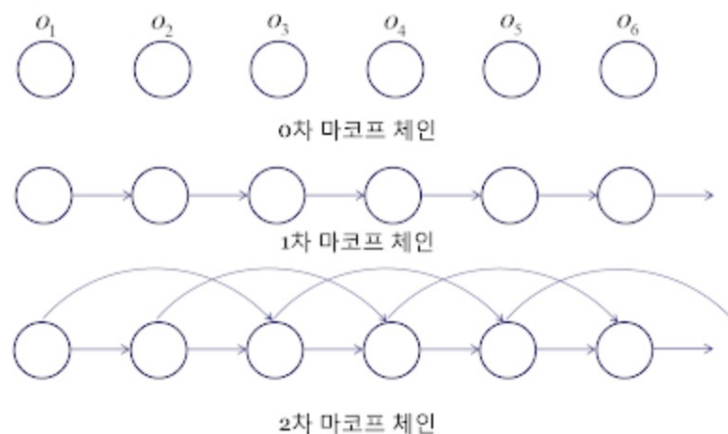
※마르코프 성질: 과거와 현재 상태가 있을 때 미래 상태의 조건부 확률 분포가 과거와는 독립적으로 현재 상태에 의해서만 결정

※이산시간 확률과정 : 이산적인 시간의 변화에 따라 확률이 변화하는 과정

$r = 0$ 이면, 0차 마코프 체인 $P(o_t | o_{t-1} o_{t-2} \dots o_1) = P(o_t)$

$r = 1$ 이면, 1차 마코프체인 $P(o_t | o_{t-1} o_{t-2} \dots o_1) = P(o_t | o_{t-1})$

$r = 2$ 이면, 2차 마코프체인 $P(o_t | o_{t-1} o_{t-2} \dots o_1) = P(o_t | o_{t-1}, o_{t-2})$



BACKGROUND

- **Propagation**

- 입력층부터 출력층까지 순서대로 변수들을 계산하고 갱신

- **Backpropagation**

- 입력층부터 출력층까지 역순으로 전파해 가면서 변수들을 갱신

PREVIOUS WORK

- **Explicit Density Model**

- 데이터의 확률 분포를 명시적으로 정의하고 학습하는 방식
 - Variational Autoencoders (VAEs)

- **Markov Chains Model**

- 데이터 생성 과정에서 마르코프 체인을 사용해 분포를 학습하는 방식
 - Deep Boltzmann Machines
 - Deep Belief Networks

PREVIOUS WORK

- **Noise-Contrastive Estimation (NCE)**

- 확률 분포 $P(x)$ 를 근사하기 위해 진짜 데이터와 가짜 데이터 간의 구별 작업 수행

- **Generative Stochastic Networks (GSN)**

- 데이터를 생성하기 위해 propagation 메커니즘 사용

- **Deep Auto Regressive Models**

- 데이터의 각 요소를 순차적으로 생성, 이전 생성 결과를 다음 데이터의 조건으로 사용

PREVIOUS WORK - LIMITS

- **Low Quality Of Generated Data**

- 생성된 데이터의 품질이 떨어짐

- **Need Markov Chain**

- Markov chain은 여러번의 샘플링 거쳐서 데이터를 생성함
- 이과정의 backpropagation을 통해 진행하는 것 보다 느림

PREVIOUS WORK - LIMITS

- **Need Strong Assumption**

- 데이터 분포를 추정하기 위해 강한 가정이 필요하다

- **Complex Network Structure And Train Process**

- Markov chain과 같이 여러번의 샘플링을 거치는 등 복잡한 구조와 학습 과정이 필요

GAN

- **Generate Adversarial Nets**

- Generative model과 Discriminative model로 이루어짐
- 두 모델이 adversarial training을 통해 학습을 진행



G

VS



D

GAN

- **Generative Model**

- D를 속여 데이터의 출처를 구별하지 못하도록 학습
- 데이터를 D가 구별 불가능할 때 까지 학습이 이루어짐

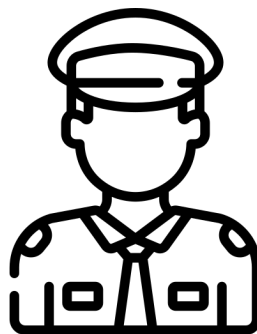


G

GAN

- **Discriminative Model**

- 데이터가 G에서 나왔는지 실제 데이터 분포에서 나왔는지 구별하도록 학습

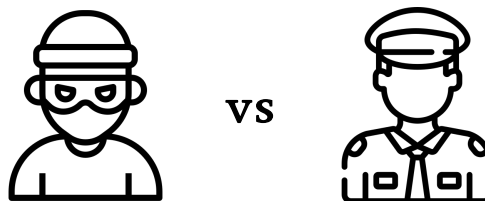


D

GAN

- **Generate Adversarial Nets**

- 경쟁 게임(min max 게임)을 진행 하듯이 학습을 진행함



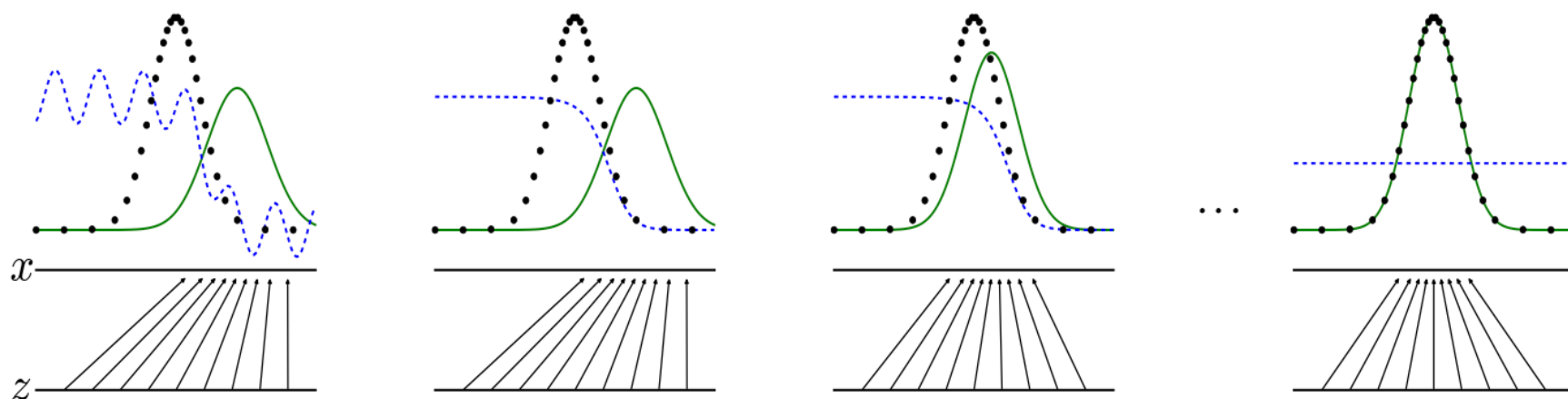
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

GAN

Blue, Dashed Line : D

Green, Solid Line : G

Black, Dotted Line : real data



GAN

- **Backpropagation**

- Gan에선 G를 학습하려면 gradient descent를 사용해야함
- 일부 직접적인 미분이 어려울 수 있어 stochastic gradient descent를 이용

$$\lim_{\sigma \rightarrow 0} \nabla_{\mathbf{x}} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})} f(\mathbf{x} + \epsilon) = \nabla_{\mathbf{x}} f(\mathbf{x})$$

• Algorithm

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**

for k steps **do**

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

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

end for

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

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

end for

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

GAN

- **Global Optimality of $p_g = p_{data}$**
 - G와 D가 충분한 능력을 가질 때 p_g 는 p_{data} 를 정확히 복원할 수 있음
- **Proposition**
 - 고정된 G에서 최적의 D

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

GAN

- **Proof**

- D는 $V(G, D)$ 를 최대화 하는 것이 목적

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) dx + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(g(\mathbf{z}))) dz \\ &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) dx \end{aligned}$$

GAN

- $y \rightarrow a \log(y) + b \log(1 - y)$
 - 위 식의 극댓값을 찾기 위해 미분을 진행함
 - $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$ 이고 $[0, 1]$ 이 조건
1. 도함수 계산

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

2. 극값 찾기

$$\frac{a}{y} = \frac{b}{1 - y}$$

$$a(1 - y) = by$$

$$a = ay + by$$

$$y = \frac{a}{a + b}$$

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



GAN

• Theorem

- $\max V(G, D)$ 를 최소화 하는 G 를 위해 아래 식의 극솟값을 찾아야 함
- \minmax 에서 \min 부분에 해당

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

GAN

• Proof

- $p_g = p_{data}$ 일 때 $D_G^*(x) = 1/2$ 이기에 $C(G) = \log\left(\frac{1}{2}\right) + \log\left(\frac{1}{2}\right) = -\log 4$ 가 됨

$$\mathbb{E}_{\mathbf{x} \sim p_{data}} [-\log 2] + \mathbb{E}_{\mathbf{x} \sim p_g} [-\log 2] = -\log 4$$

- $C(G) = V(D_G^*, G)$ 로부터 나타내면 아래와 같은 식을 도출

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

- JSD는 항상 음수가 아닌 값을 가지며 $p_g = p_{data}$ 일 때 0을 가짐

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

- 극솟값은 $p_g = p_{data}$ 일 때 $-\log 4$ 를 갖게 됨

EXPERIMENTS

- Parzen Window Log – Likelihood

- 전반적으로 GAN이 높은 품질의 샘플을 생성

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	2110 ± 50
Deep GSN [5]	214 ± 1.1	1890 ± 29
Adversarial nets	225 ± 2	2057 ± 26

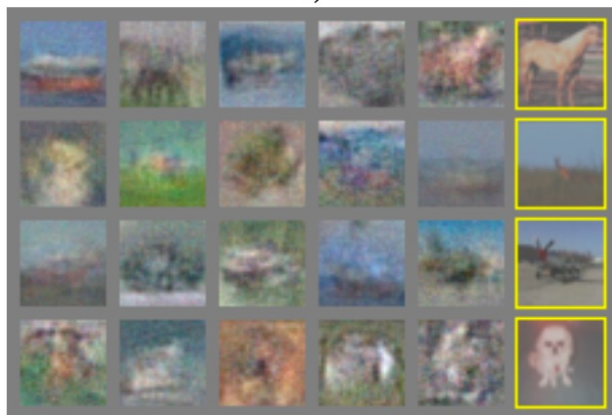
EXPERIMENTS



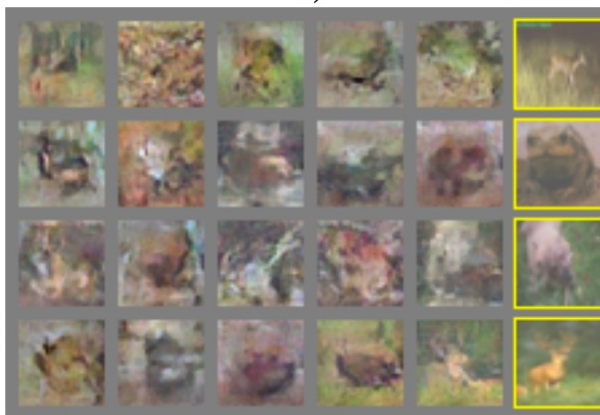
a)



b)



c)



d)

- a) MNIST
- b) TFD
- c) CIFAR-10 (fully connected model)
- d) CIFAR-10

EXPERIMENTS

- Digits obtained by linearly interpolating between coordinates in z space



EXPERIMENTS

	Deep directed graphical models	Deep undirected graphical models	Generative autoencoders	Adversarial models
Training	Inference needed during training.	Inference needed during training. MCMC needed to approximate partition function gradient.	Enforced tradeoff between mixing and power of reconstruction generation	Synchronizing the discriminator with the generator. Helvetica.
Inference	Learned approximate inference	Variational inference	MCMC-based inference	Learned approximate inference
Sampling	No difficulties	Requires Markov chain	Requires Markov chain	No difficulties
Evaluating $p(x)$	Intractable, may be approximated with AIS	Intractable, may be approximated with AIS	Not explicitly represented, may be approximated with Parzen density estimation	Not explicitly represented, may be approximated with Parzen density estimation
Model design	Models need to be designed to work with the desired inference scheme — some inference schemes support similar model families as GANs	Careful design needed to ensure multiple properties	Any differentiable function is theoretically permitted	Any differentiable function is theoretically permitted

PROS

- **No Need Markov Chain**

- 단일 단계로 샘플을 생성하기 때문에 훨씬 빠르고 효율적

- **Inference Free Generation**

- 노이즈 z 를 입력하면 G 가 즉시 샘플을 생성할 수 있음
- 추가적인 inference과정이 필요 없어 간단하고 효율적

- **More Sharp Image**

- G 와 D 가 adversarial training 덕분에 생성하는 데이터의 품질이 점진적으로 개선

- **Flexible Modeling**

- 특정한 확률 모델을 가정하지 않아도 되며, 다양한 구조로 확장 가능

CONS

- **Training Instability**

- Minmax를 진행하다 보니, 일반적인 신경망 학습보다 최적화가 어려움
- 학습초기 D가 너무 강하면, G가 의미 있는 학습신호를 받지 못할 수 있음
- G가 너무 강하면, D는 학습할 수 없는 상태가 됨

- **No Explicit Likelihood**

- VAE와 같은 명시적 확률 분포 모델들은 평가가 상대적으로 쉬움
- GAN은 직접적인 평가 방법이 없고, 눈으로 직접 확인 하거나, Parzen window 방법등을 사용

- **Model Collapse**

- G가 일부 데이터 패턴만 학습하여 다양성이 부족한 데이터를 생성하는 현상
- GAN이 충분한 다양성을 확보하지 못하고 특정 패턴만 학습하는 경우 발생

CONCLUSIONS

- **Gan**

- Generative model과 Discriminative model로 이루어져 Adversarial train을 진행
- Backpropagation을 활용함

- **Pros**

- Markov chain과 inference과정을 필요로 하지 않아 속도가 빠름
- 더 정확하고 명확한 image를 생성함

- **Cons**

- GAN은 직접적인 평가 방법이 없고, 눈으로 직접 확인 하거나, Parzen window 방법등을 사용
- G가 일부 데이터 패턴만 학습하여 다양성이 부족한 데이터를 생성하는 현상 발생하기도 함

- **Result**

- 유연한 모델을 가지고 있기 때문에 후에 단점을 극복한 모델들 연구가 필요함