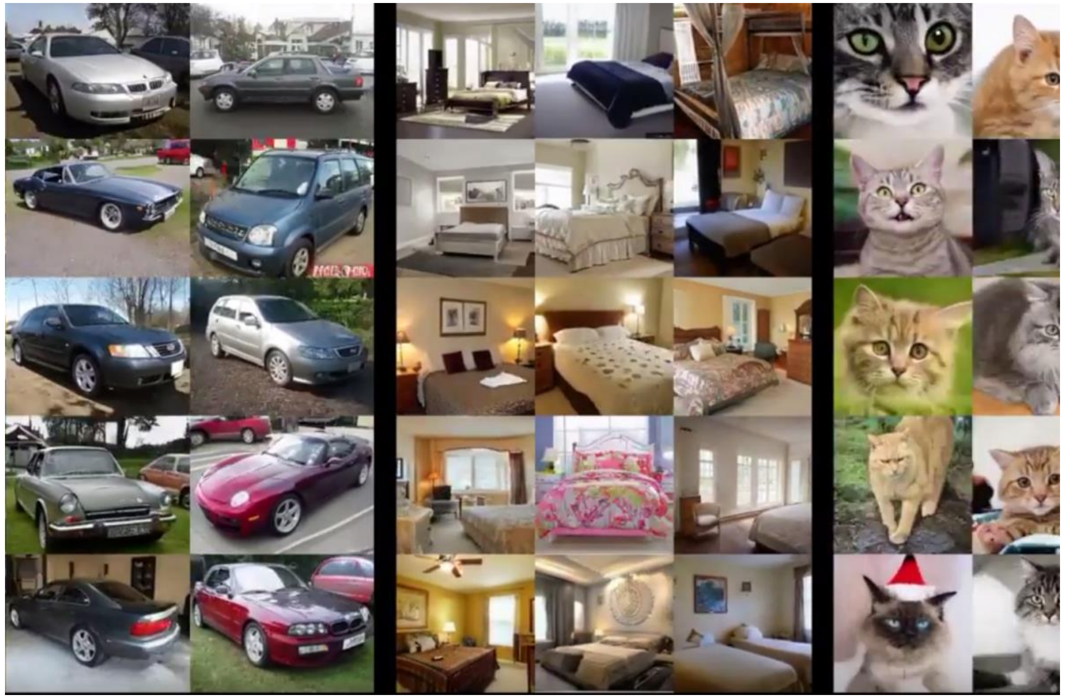
# GAN

nonezero@kumoh.ac.kr
고재필

# Generative-Adversarial Networks

- 대립(Adversarial) 방식을 통해 (학습하는) **생성적(Generative) 신경망**

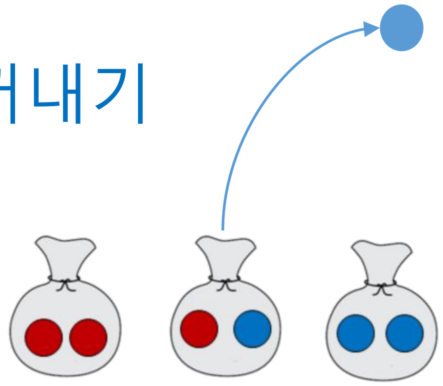
Google (2017)



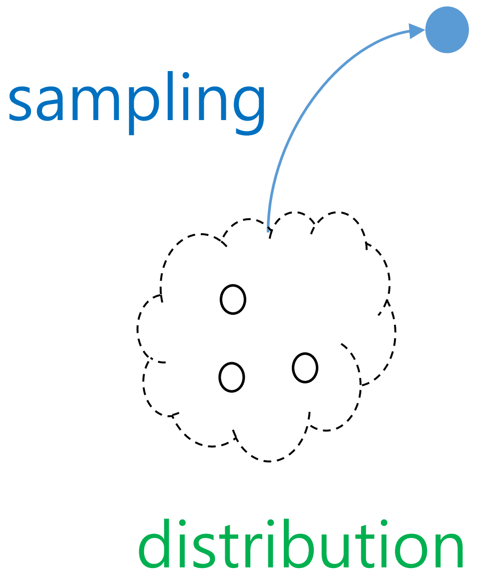
NVidia (2019)

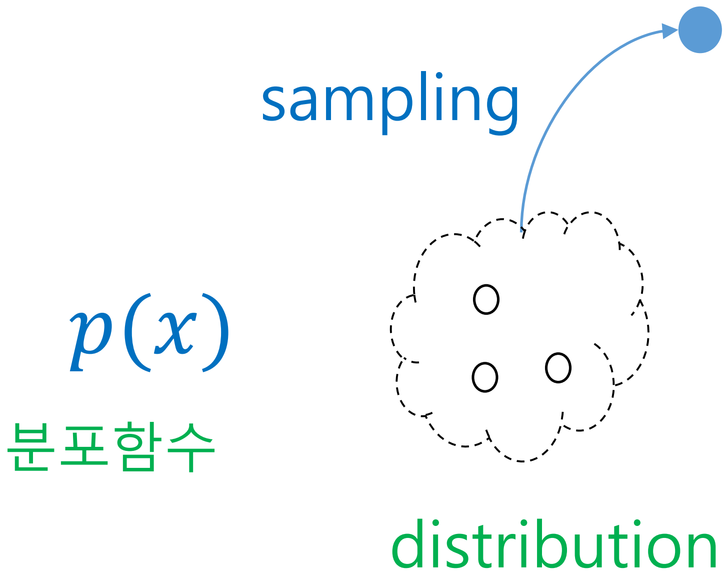
# 들어가기...

꺼내기



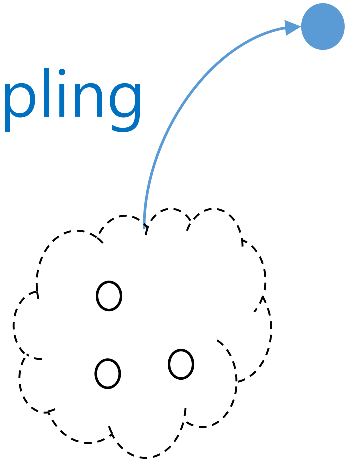
'어느 주머니'인지에 따라
꺼내 온 구슬이 달라진다.


sampling
distribution




$$p(x) \leftrightarrow p_X(x) \leftrightarrow p(X = x)$$

$$x \sim p\,(x)$$

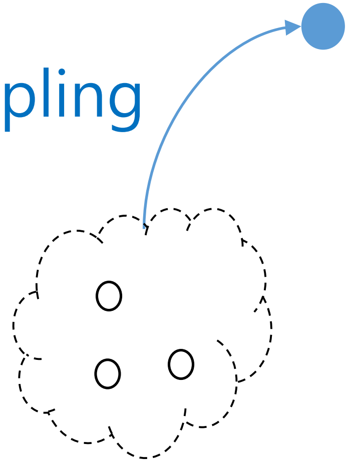



$p(x)$

분포함수

$$p(x) \leftrightarrow p_X(x) \leftrightarrow p(X = x)$$

sampling

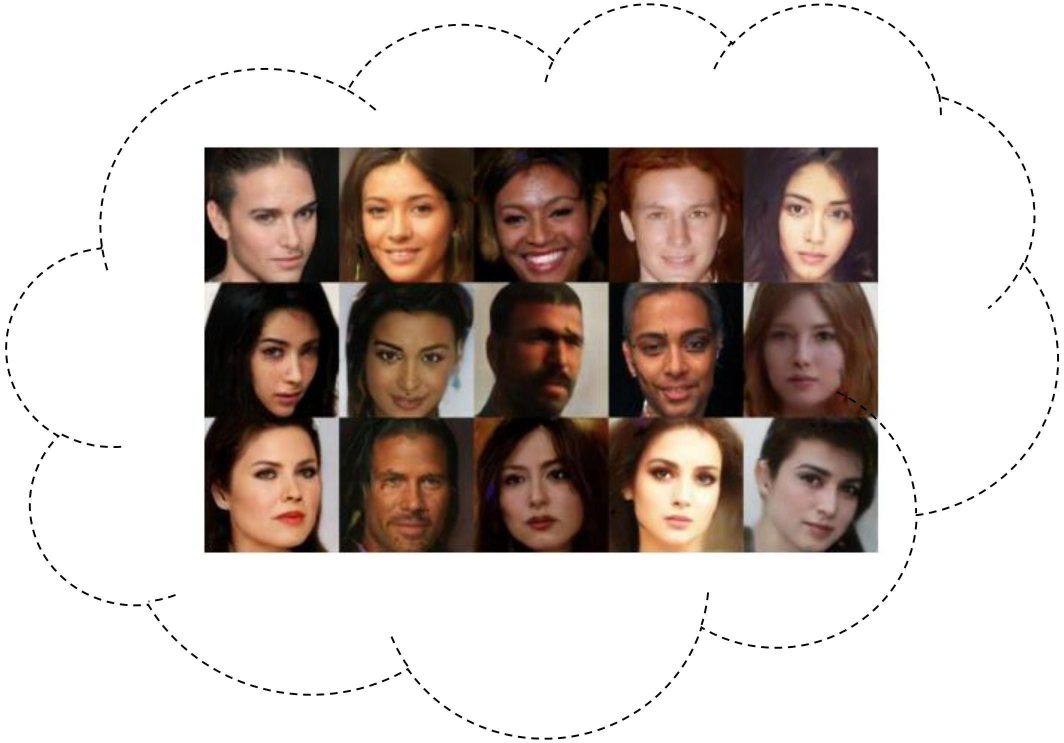


$x \sim p\,(x)$ generation

$p(x)$

분포함수

distribution



$$p(x) \leftrightarrow p_X(x) \leftrightarrow p(X = x)$$

**information** $-log^{p(x)}$

**expectation** $E[X] = \sum_x xp(x)$

$$E[f(X)] = \sum_x f(x)p(x)$$

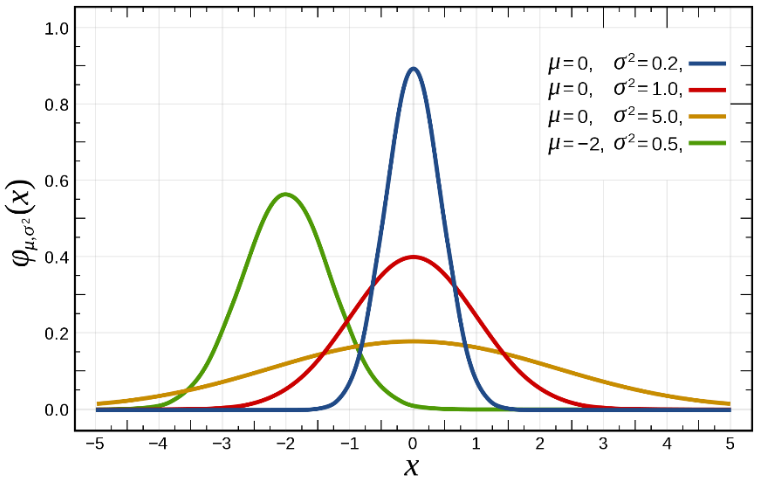
**entropy** $E\left[-log^{p(X)}\right] = \sum_x -log^{p(x)}p(x)$

**Bayes' rule** $p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(y)p(x|y)}{\sum_y p(y)p(x|y)}$

$$p(x,y) = p(x)p(y|x) = p(y)p(x|y)$$

$$p(x,y,z) = p(x)p(y|x)p(z|x,y)$$

$$p(x) = \sum_y p(x,y) = \sum_y p(y)p(x|y)$$




$$p(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}((x-\mu)/\sigma)^2}$$

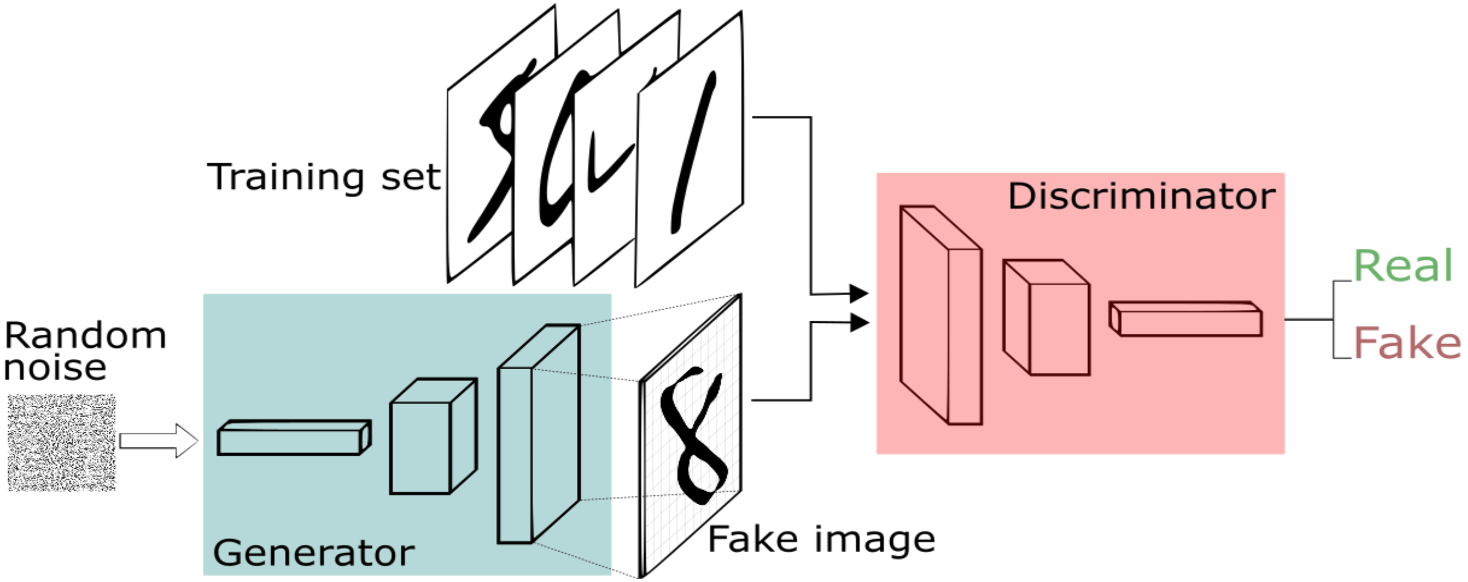
$$p(x; u, \sigma)$$

$p_\theta(x)$ $\theta = (u, \sigma)$

# 신경망 구조

- 대립(Adversarial) 방식을 통해 학습하는 생성적(Generative) 신경망
- 생성자(**G**enerator)와 구분자(**D**iscriminator)가 대립하는 네트워크 구조

adversarial



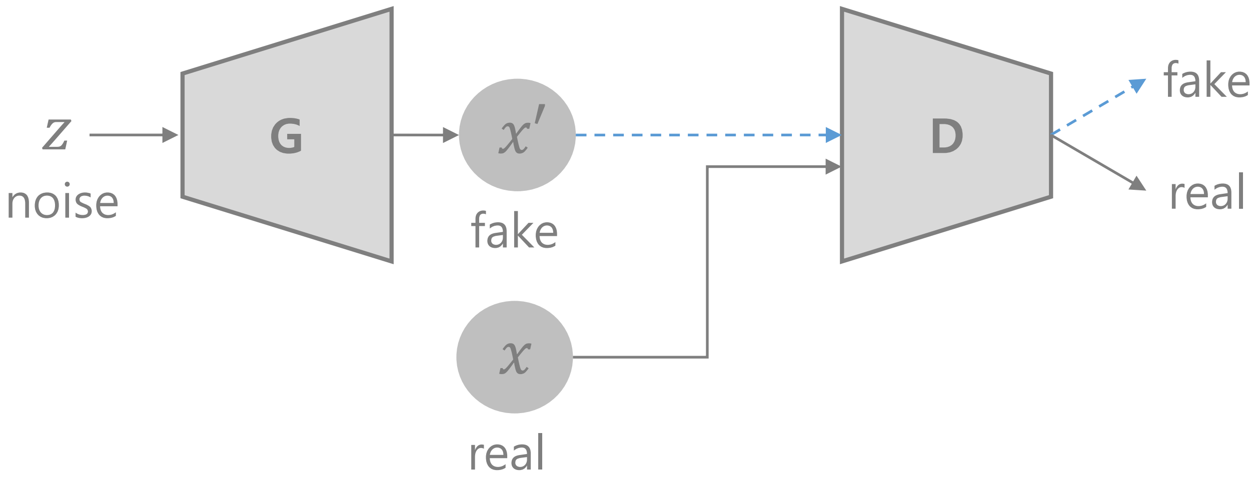

생성자의
손실함수 역할

Credit: Rohith Gandhi, Medium

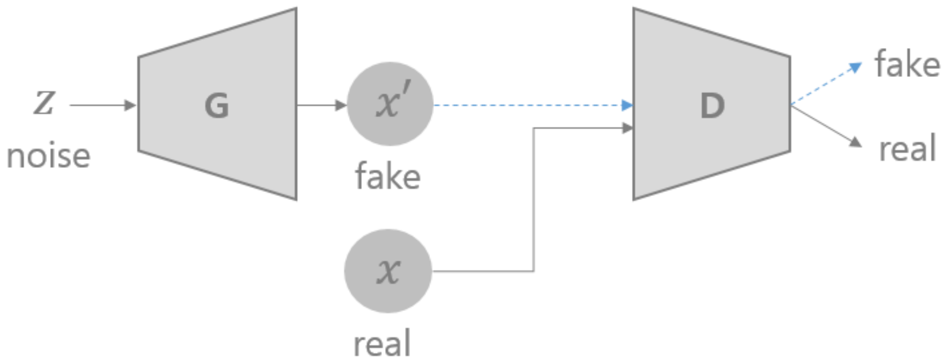
# 손실함수

- 목적함수
  - 생성자(G)는 진짜 같은 데이터를 생성하고,
  - 구분자(D)는 진짜 데이터와 생성된 데이터를 잘 구분하도록,
  - 학습하는 것을 목적으로 해야 한다.

네트워크 구조




목적함수 ?




$0 \leq D(x) \leq 1$

$D(x)$를 $x$**가 진짜일 확률(**분포)라고 간주

① 우리의 목표는 진짜인 $x$에 대해서 $D(x)$를 커지게 하고
합성된 녀석인 $x' = G(z)$에 대해서 $D(G(z))$가 작아지도록 하는 것이다.

$\max D(x)$

$\max \left(1 - D(G(z))\right)$

$D(G(z))$를 최소화하는 것을
최대화 하는 식으로 바꾼 것

$\max D(x) + \left(1 - D(G(z))\right)$

이 식을 최대화 하자

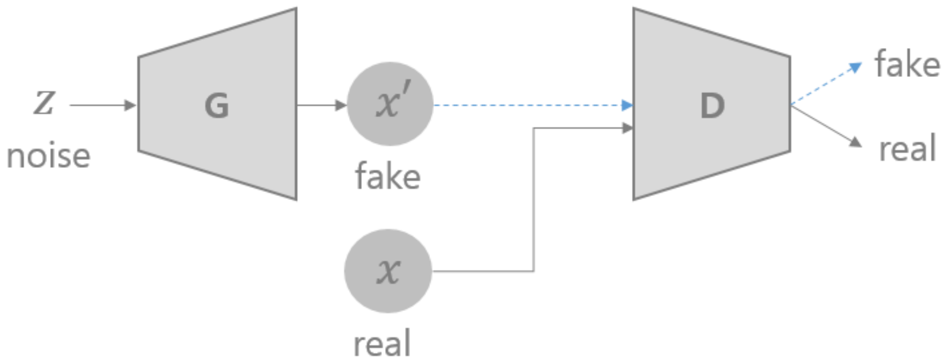
결론적으로 다음과 같이 표현할 수 있다.

$$\min_G \max_D D(x) + \left(1 - D(G(z))\right)$$

② 그리고,
$D(G(z))$가 1이 되도록 한다.

$max\, D(G(z))$

$min\, \left(1 - D(G(z))\right)$




$0 \leq D(x) \leq 1$

$D(x)$를 $x$**가 진짜일 확률(**분포함수)라고 간주

① 우리의 목표는 진짜인 $x$에 대해서 $D(x)$를 커지게 하고
합성된 녀석인 $x' = G(z)$에 대해서 $D(G(z))$가 작아지도록 하는 것이다.

$\max D(x)$

$\max \left(1 - D(G(z))\right)$

$D(G(z))$를 최소화하는 것을
최대화 하는 식으로 바꾼 것

$\max D(x) + \left(1 - D(G(z))\right)$

이 식을 최대화 하자

이 값은 최소화

**서로 상충**

② 그리고,
$D(G(z))$가 1이 되도록 한다.

$max\, D(G(z))$

$min\left(1 - D(G(z))\right)$

결론적으로 다음과 같이 표현할 수 있다.

$$\min_G \max_D D(x) + \left(1 - D(G(z))\right)$$

동시에 만족시킬 수 없다.
번갈아 가며 최대화/최소화를 해야 한다.

# 목적함수

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

D에 log 취해도 무방-대소관계유지, 그리고 값 하나보단 평균이 낫다.

$$max D(x) + \left(1 - D\left(G(z)\right)\right)$$

$max\, D\left(G(z)\right)$ 최대화 식

$min\left(1 - D\left(G(z)\right)\right)$ 최소화 식

# 손실함수

$$LossD = -\left(E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}\left[log\left(1 - D\left(G(z)\right)\right)\right]\right)$$

$$LossG = E_{z\sim p_z(z)}\left[log\left(1 - D\left(G(z)\right)\right)\right]$$

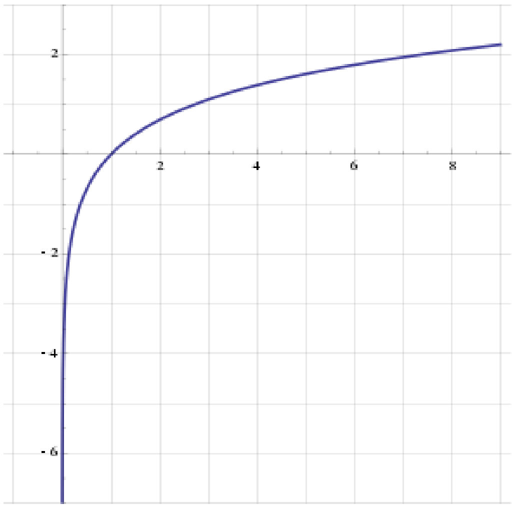
$$LossG = -E_{z\sim p_z(z)}\left[logD\left(G(z)\right)\right]$$ 선호

$$LossG = E_{z\sim p_z(z)}\left[log\left(1 - D\left(G(z)\right)\right)\right]$$

$$LossG = -E_{z\sim p_z(z)}\left[logD\left(G(z)\right)\right]$$

$$LossG = 1 - E_{z\sim p_z(z)}\left[logD\left(G(z)\right)\right]$$

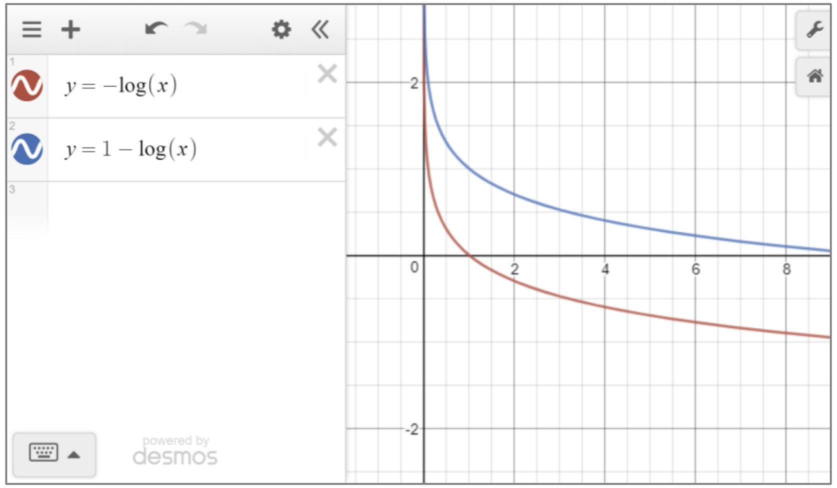
+1을 해서 최소가 0이 되도록 하여 사용
(그림에서 파란색 그래프)




학습 초기에는
G가 만들어내는 결과가 형편 없음
→ D가 거의 0일 가능성이 높음.

log(1-D),
log(1) **근방**에서는 기울기가 작다.

log(D),
log(0) **근방**에서는 기울기가 크다.




https://www.desmos.com/calculator

**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**

update D
for a fixed G

**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:

최대값 구하기이므로,
그레디언트의 + 방향으로 업데이트

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

**end for**

- 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:

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

최소값 구하기이므로,
그레디언트의 - 방향으로 업데이트

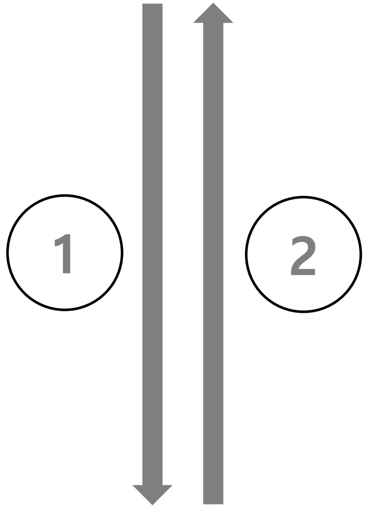
**end for**

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

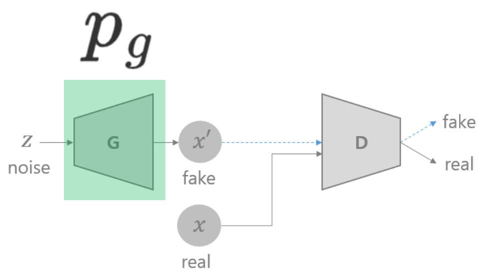
https://arxiv.org/pdf/1406.2661.pdf

# 최적화 목적을 달성하면...

$$p_g = p_{\text{data}}$$



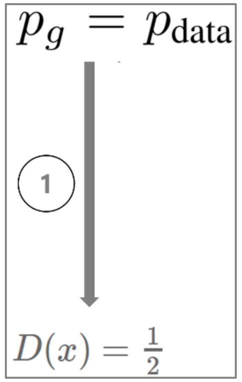

$$D(x) = \frac{1}{2}$$




G가 실재 data 분포를 완벽하게 나타내어, D가 구분 못하는 상황

# ½ 이 나오는지 보자

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




일단 G가 구해졌다고 가정하고, 목적함수를 최대화하는 D를 먼저 구해보자.

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

$x = G(z)$

$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(\ x\ )))dz$$

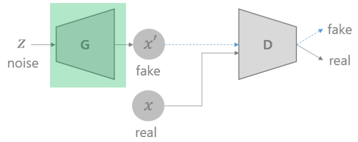
z를 없애자

$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D( \; x \; ))dz$$

$$x = G(z)$$
$$\Rightarrow z = G^{-1}(x)$$
$$\Rightarrow \frac{d}{dx} z = \frac{d}{dx} G^{-1}(x)$$
$$\Rightarrow \frac{d}{dx} z = \left(G^{-1}\right)'(x)$$
$$\Rightarrow dz = \left(G^{-1}\right)'(x)dx$$

$$= \int_x p_{data}(x) log\big(D(x)\big)dx + \int_x p_z\big(G^{-1}(x)\big) log\big(1 - D(x)\big)\left(G^{-1}\right)'(x)dx$$

$$p_g(x) \equiv p_z\big(G^{-1}(x)\big)\left(G^{-1}\right)'(x)$$




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

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

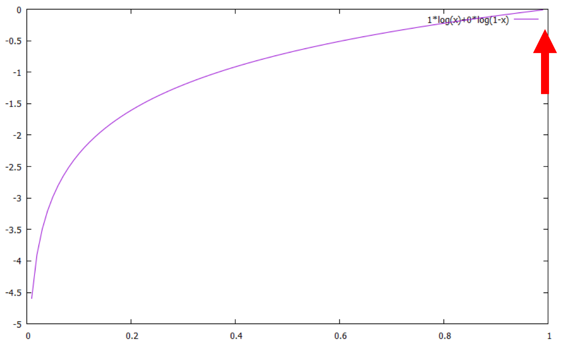
$$\int_x p_{data}(x)log\big(D(x)\big) + p_g(x)log\big(1 - D(x)\big)dx$$

이 식을 최대화하는 D를 구하고 있었다.

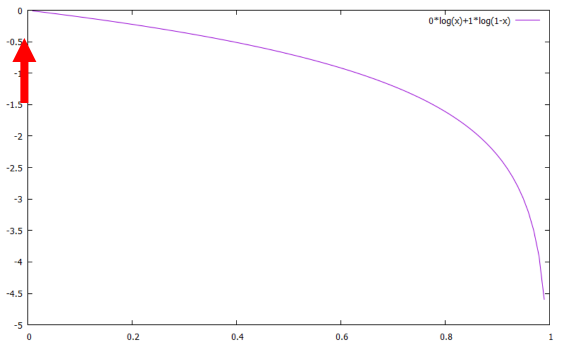
위 식의 특성을 살펴보기 좋게 대치

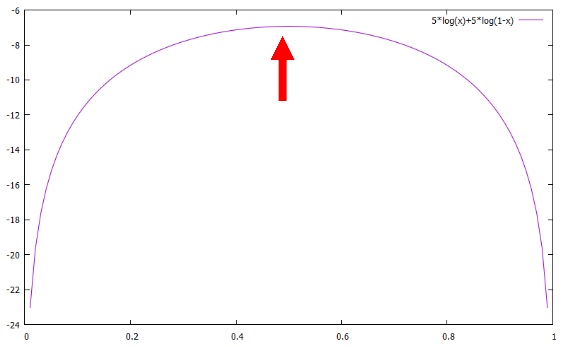
$$= \int a \cdot log(y) + b \cdot log(1-y)dx$$

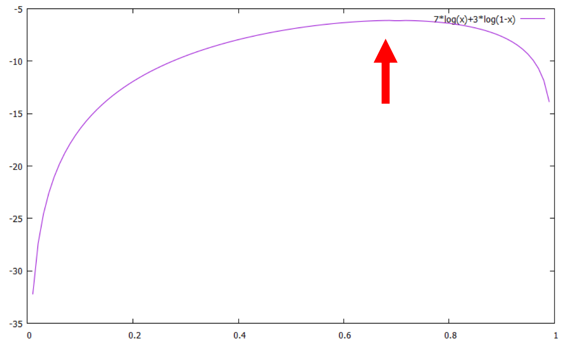
이 식의 그래프 모양은 아래와 같음




0 y 1




미분해서 0되는 지점이 최대 값

$$V\left(G^{fixed}, D\right) = a \cdot log(y) + b \cdot log(1-y)$$

D; y에 대해 미분해서 0되는 지점이 최대 값

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

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

*any given G*

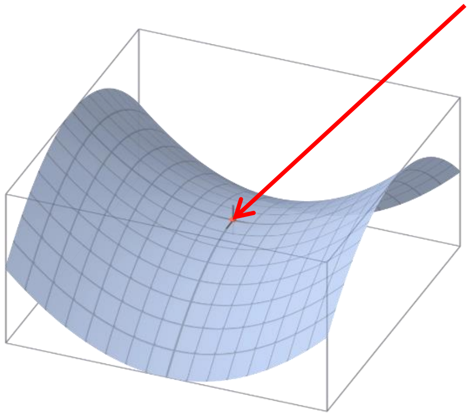
*Optimal Discriminator*

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

$$
D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} = \frac{1}{2}
$$

$p_g = p_{data}$

$$
C(G) = \log \tfrac{1}{2} + \log \tfrac{1}{2} = -\log 4
$$



- **K**ullback-**L**eibler divergence

$p(x)$에 대한 $q(x)$의 엔트로피

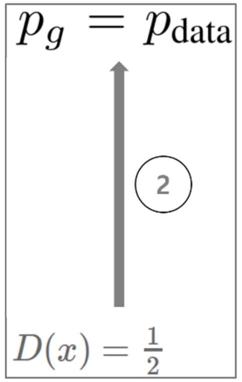
$p(x)$에 대한 $p(x)$의 엔트로피

$$KL(p||q) = -\int p(x)\ln q(x)dx - \left(-\int p(x)\ln p(x)dx\right) = -\int p(x)\ln\left(\frac{q(x)}{p(x)}\right)dx$$

$KL(p||q) \geq 0$

$KL(p||q) \neq KL(q||p)$

# $p_g = p_{\text{data}}$ 인지 보자




$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] - \log(4) + \log(4)
\end{aligned}
$$

$$
= -\log(4) + \int p_{data} \cdot log \frac{p_{data}}{\frac{p_{data} + p_g}{2}} dx + \int p_g \cdot log \frac{p_g}{\frac{p_{data} + p_g}{2}} dx
$$

$$= -\log(4) + \int p_{data} \cdot \log \frac{p_{data}}{\frac{p_{data} + p_g}{2}} dx + \int p_g \cdot \log \frac{p_g}{\frac{p_{data} + p_g}{2}} dx$$

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

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

정의

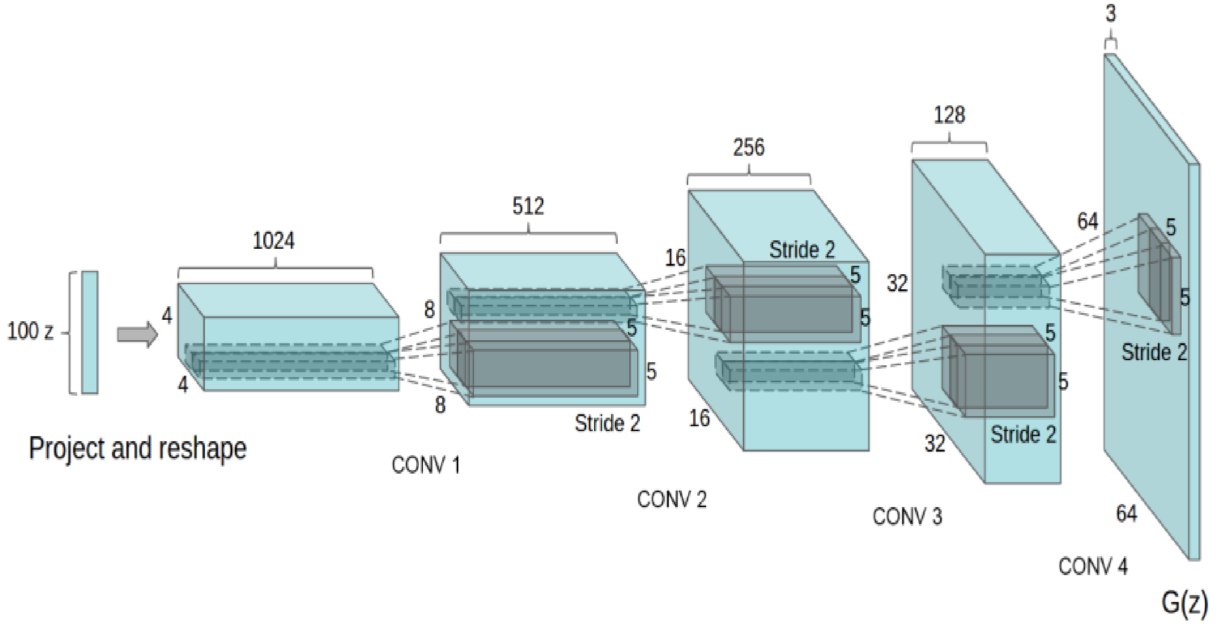
$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

**Jensen–Shannon divergence** (symmetric KL)

**non-negative** and **zero** only when they are equal

목적함수가 최소가 되려면 $\Rightarrow \quad p_g = p_{\text{data}}$

# DCGAN Deep Convolutional GANs

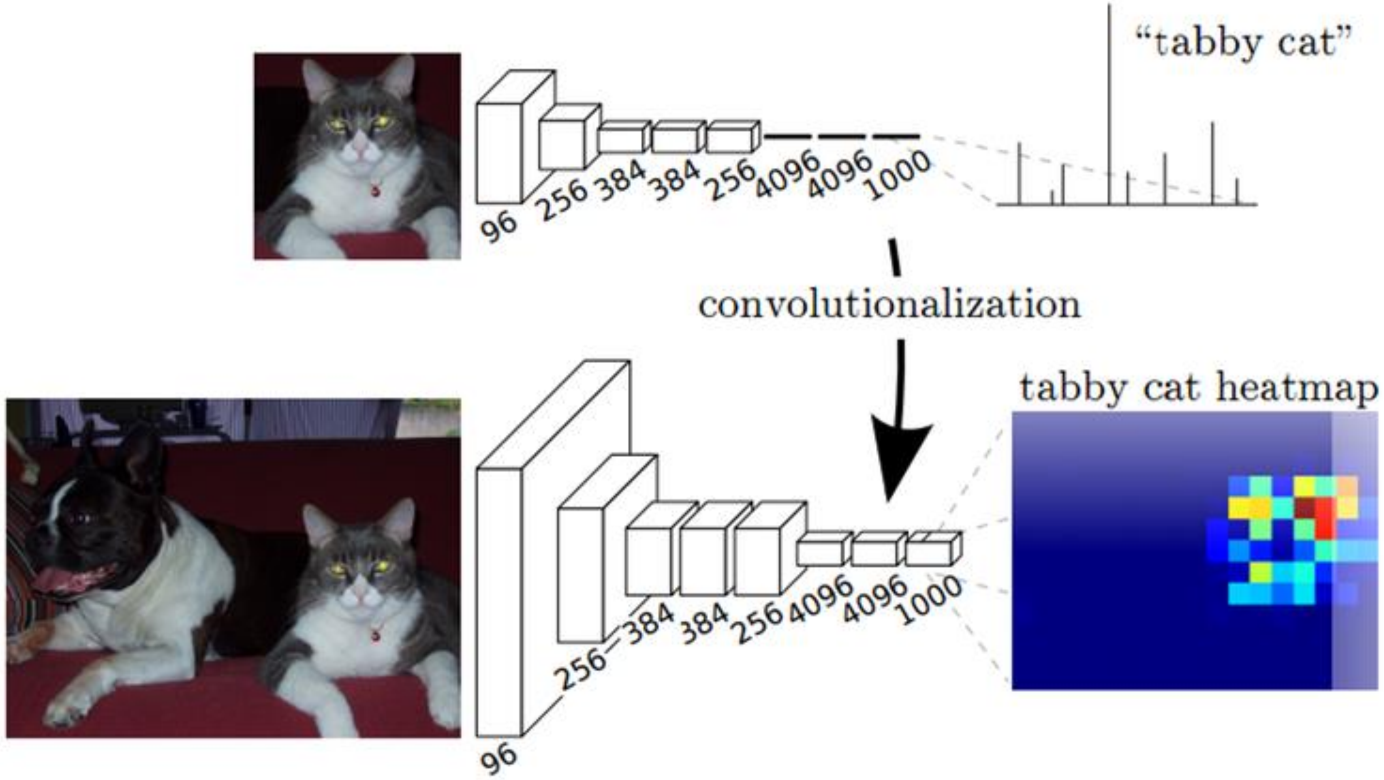



Architecture guidelines for stable Deep Convolutional GANs

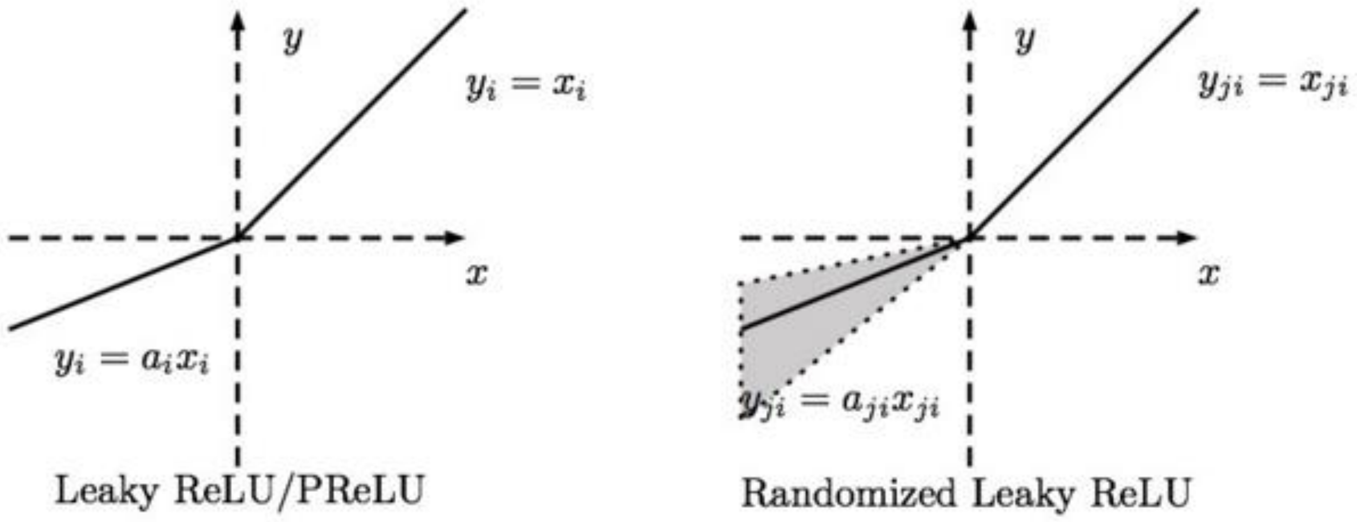
- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except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

https://arxiv.org/abs/1511.06434 (2015, 2016)

# Fully-Convolutional Neural Networks



# Leaky ReLU

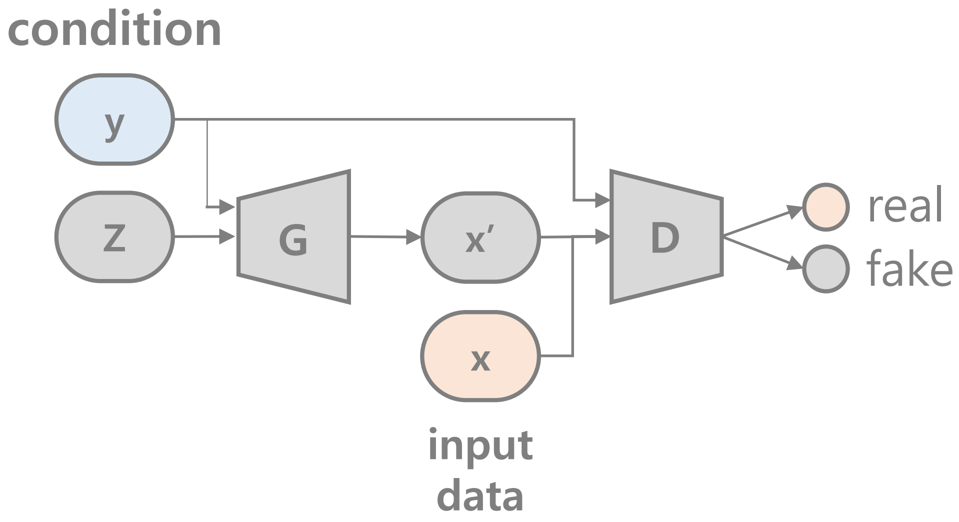



https://blog.csdn.net/cyh_24/article/details/50593400

# Conditional GAN

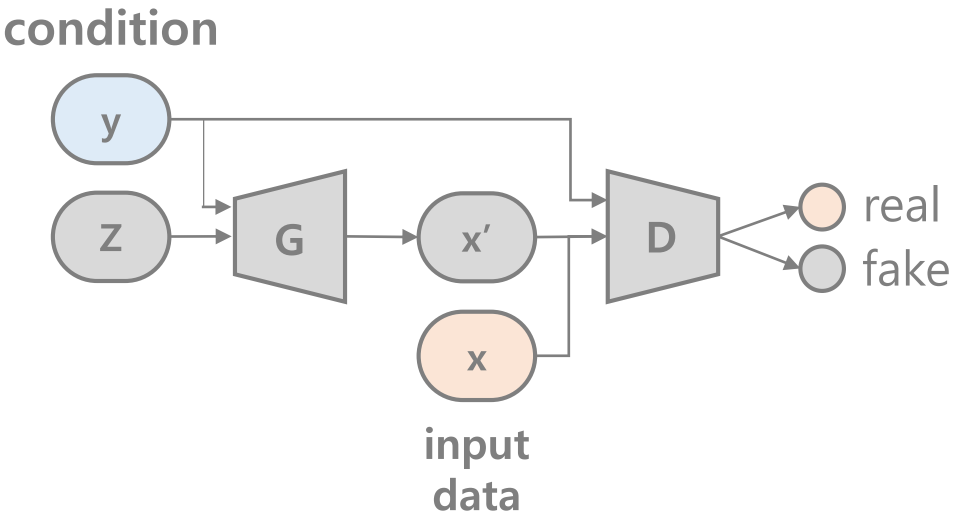
클래스 정보를 주어서 학습하자.
특정 클래스를 생성할 수 있게 된다.

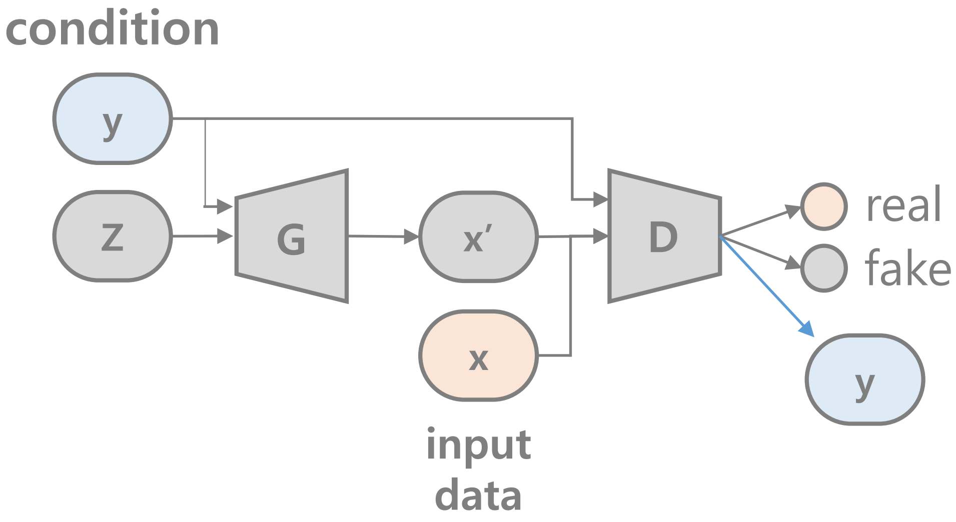
조건 정보를 제공
예를 들면, x는 숫자영상
y는 그 숫자의 값, 그러면 y에 해당되는 x'가 생성된다.
이전 방식은 어떤 숫자의 x'가 생성될지 모름




"사람이 알고 있는 정보를 제공하여,
신경망의 학습부담을 줄여준다."

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

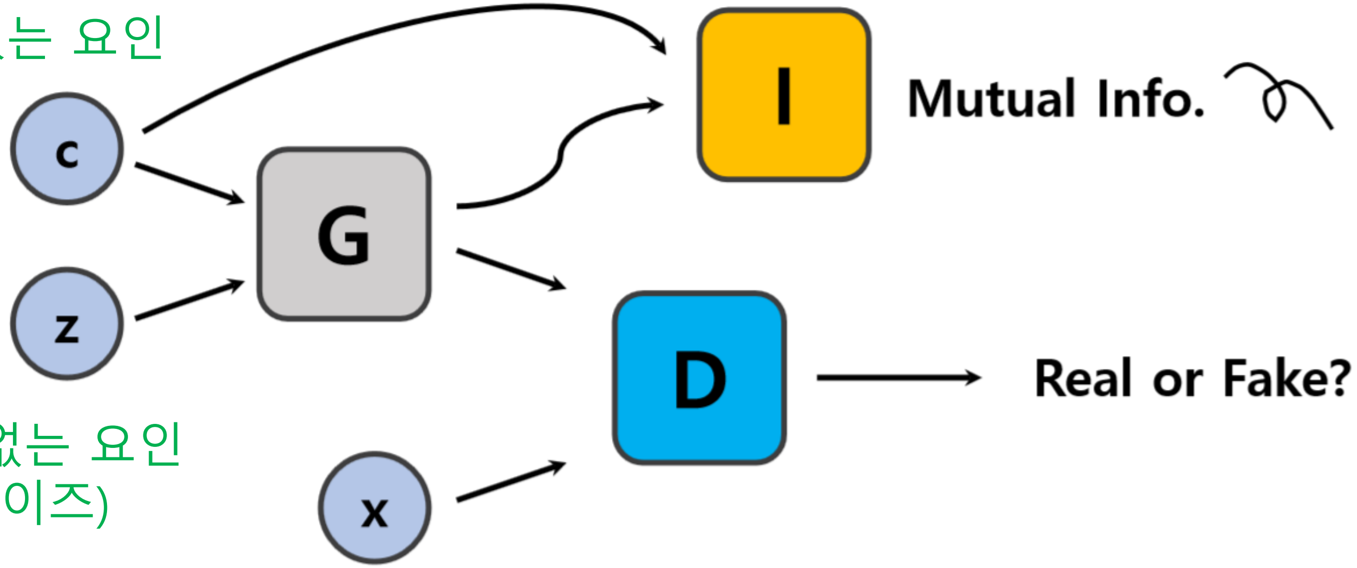

condition
y
Z
G
x′
D
real
fake
x
input
data


condition
y
Z
G
x′
D
real
fake
x
y
input
data

# InfoGAN

- 의미 있는 잠재요인latent variable을 네트워크가 추출하도록 구성

의미 있는 요인




의미 없는 요인
(노이즈)

**infoGAN**
**: maximize I(c,G(z,c))**

mutual information
두 확률변수 간의 연관성
(독립이 아닌 정도)

이를 최대화하면
C가 제공되는 만큼,
G는 배움이 쉬워질 것 같기도...

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

# Mutual Information

정의

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

독립이 아닌 정도에 대한 평균

$$= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)} - \sum_{x,y} p(x,y) \log p(y)$$

$$= \sum_{x,y} p(x)p(y|x) \log p(y|x) - \sum_{x,y} \log p(y) p(x,y)$$

$$= \sum_{x} p(x) \left( \sum_{y} p(y|x) \log p(y|x) \right) - \sum_{y} \log p(y) \left( \sum_{x} p(x,y) \right)$$

$$= -\sum_{x} p(x) \mathrm{H}(Y|X=x) - \sum_{y} p(y) \log p(y)$$

$$= -\mathrm{H}(Y|X) + \mathrm{H}(Y)$$

엔트로피

$$= \mathrm{H}(Y) - \mathrm{H}(Y|X).$$

다른 변수로부터 얻을 수 있는 엔트로피 (평균 정보량)

엔트로피가 크다는 얘기는 가능성이 낮다 (불확실성이 크다)
Y불확실성 – X가 주어졌을 때의 Y불확실성은 X가 주어져서 사라지는 불확실성

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

# 상호정보 최대화

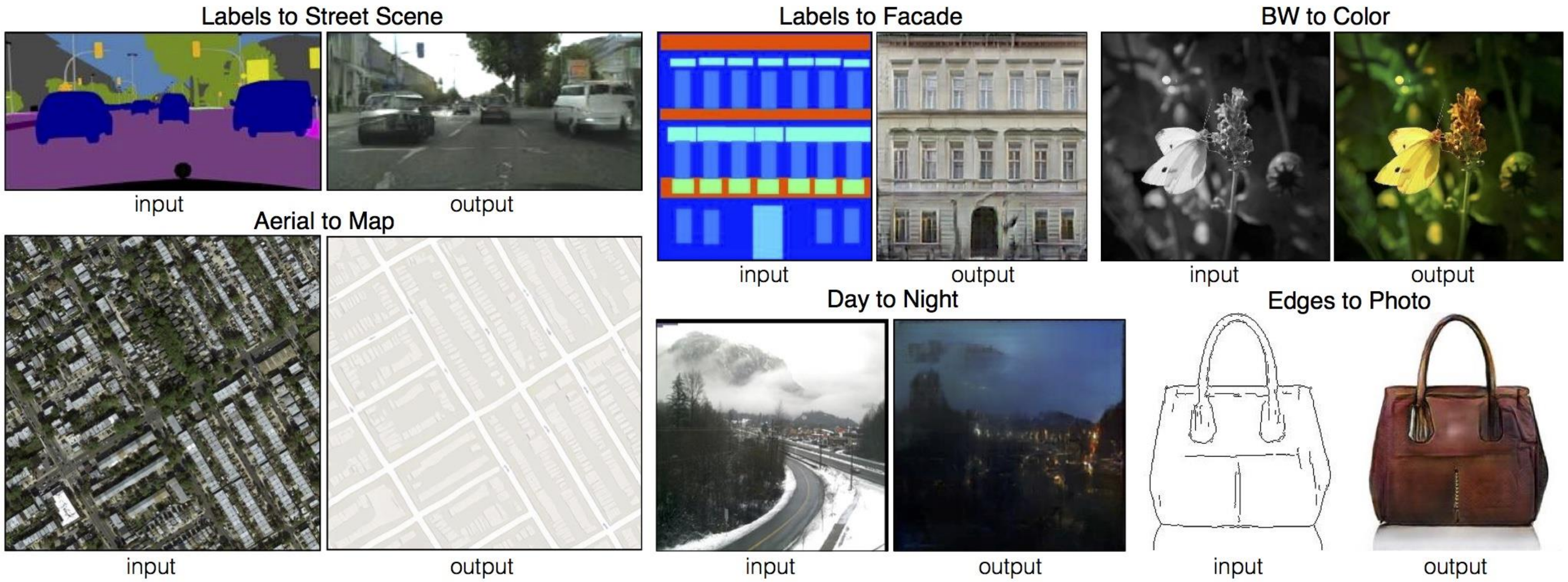
$$
\begin{aligned}
I[X,Y] &= H[Y] - \mathbb{E}_x H[Y|X=x] \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log p(y|x) \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log \frac{p(y|x)q(y|x)}{q(y|x)} \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log q(y|x) + \mathbb{E}_x \mathbb{E}_{y|x} \log \frac{p(y|x)}{q(y|x)} \\
&= H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log q(y|x) + \mathbb{E}_x KL[p(y|x) \| q(y|x)] \\
&\geq \underline{H[Y] + \mathbb{E}_x \mathbb{E}_{y|x} \log q(y|x)}
\end{aligned}
$$

Lower Bound

$q(y|x;\psi)$ is a parametric probability distribution

$p$대신 우리가 다룰 수 있는 분포함수로 대치했음

# Pix2Pix

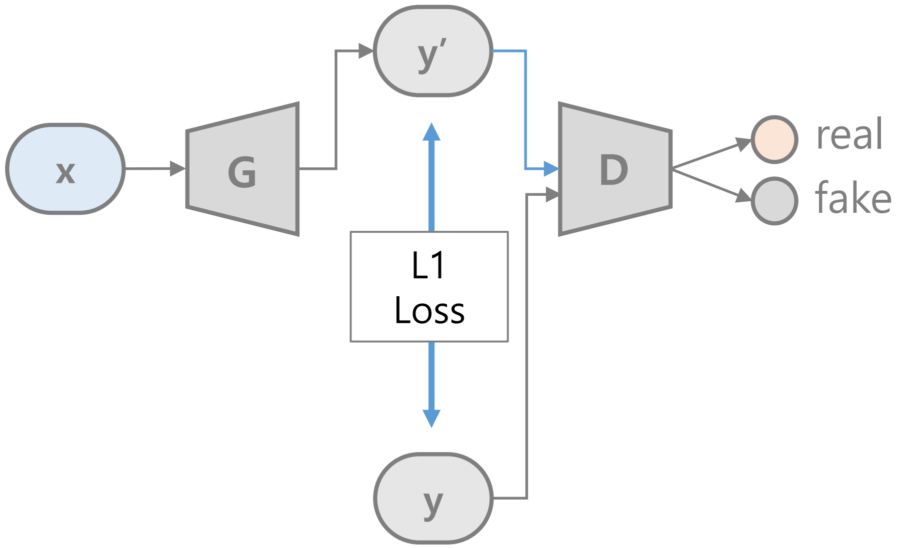


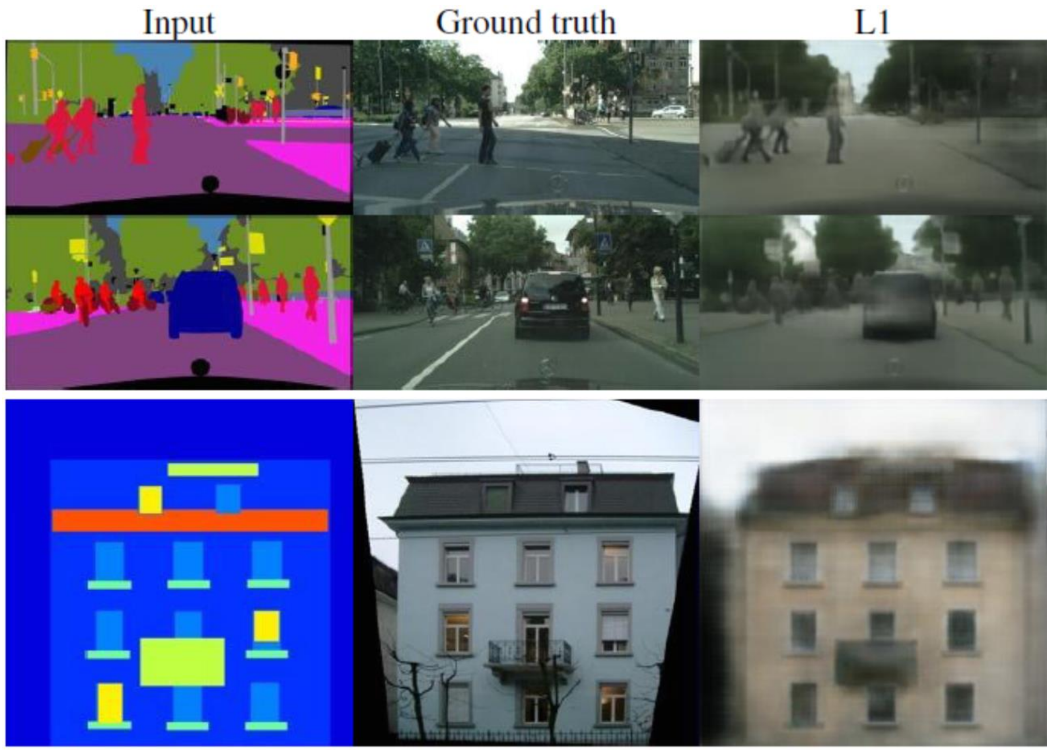

Colorful Image Colorization(ECCV2016)
형태유지(색깔만 변경)
데이터 취득 쉽다.



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

# GAN + L1



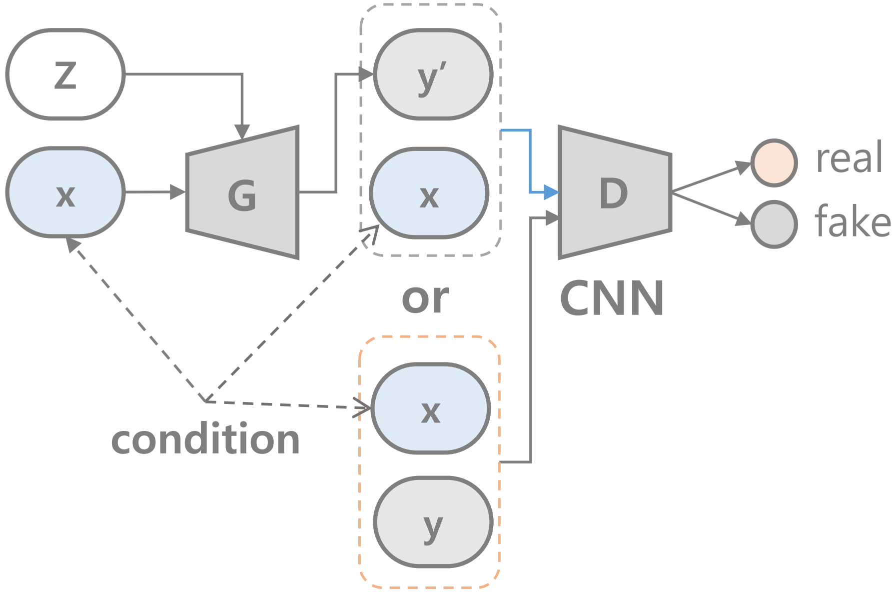



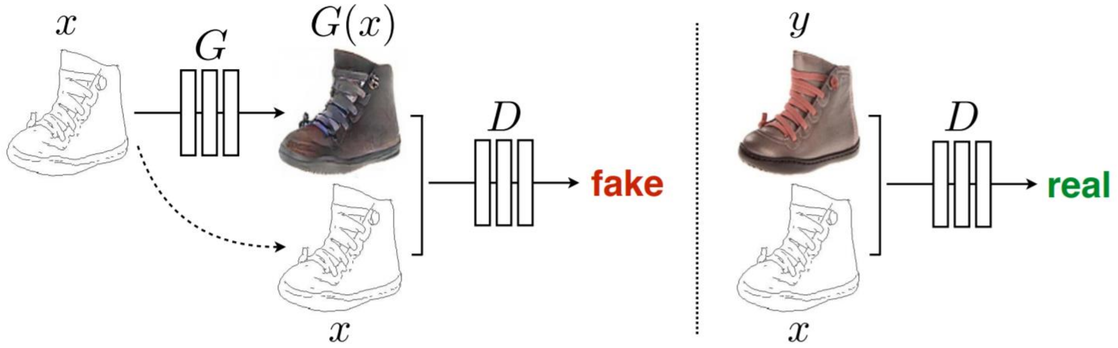

$$Loss = \sum_{x \in EveryPixel} \|GT(x) - Pred(x)\|$$

L1 loss만으로는 형태유지가 어렵다.
합(평균)관점 학습

$$\min_G \max_D \mathbb{E}_y[log(D(y)] + \mathbb{E}_x[log(1 - D(G(x))] + \mathbb{E}_{x,y}[\|y - G(x)\|_1]$$

# Conditional GAN + L1




$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

# PatchGAN


Figure 6: Patch size variations. Uncertainty in the output manifests itself differently for different loss functions. Uncertain regions become blurry and desaturated under L1. The 1x1 PixelGAN encourages greater color diversity but has no effect on spatial statistics. The 16x16 PatchGAN creates locally sharp results, but also leads to tiling artifacts beyond the scale it can observe. The 70×70 PatchGAN forces outputs that are sharp, even if incorrect, in both the spatial and spectral (colorfulness) dimensions. The full 286×286 ImageGAN produces results that are visually similar to the 70×70 PatchGAN, but somewhat lower quality according to our FCN-score metric (Table 3). Please see `https://phillipi.github.io/pix2pix/` for additional examples.

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

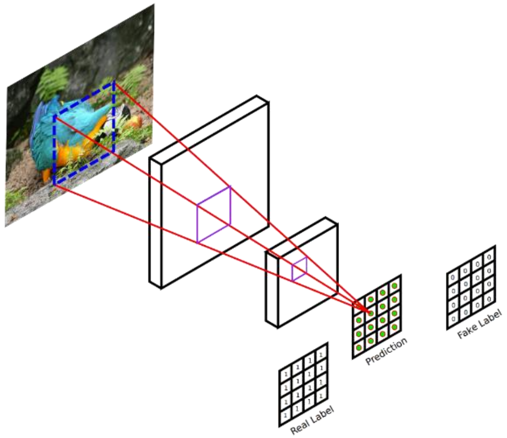
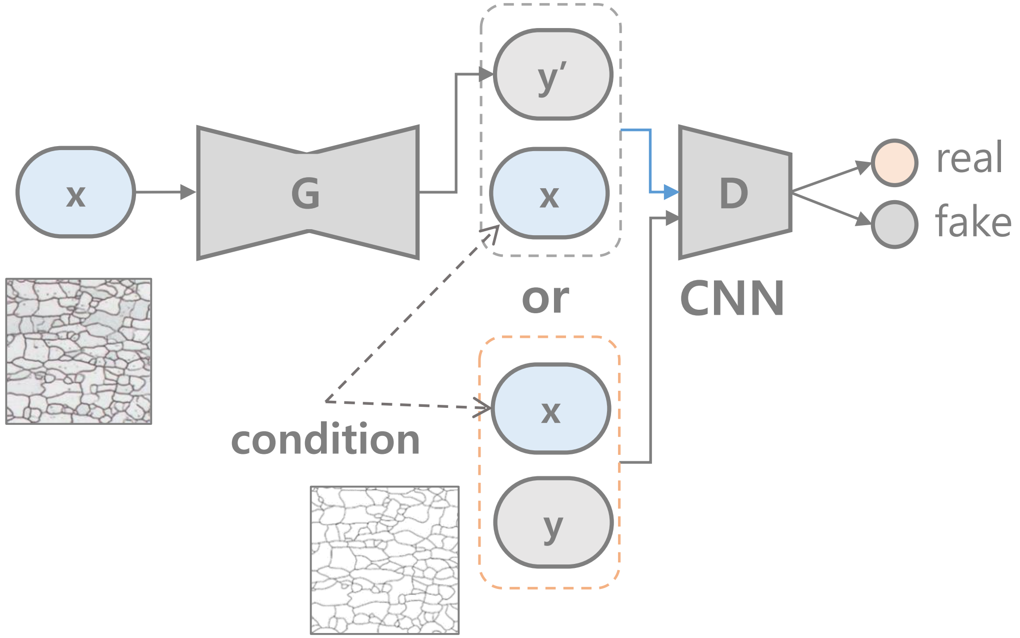
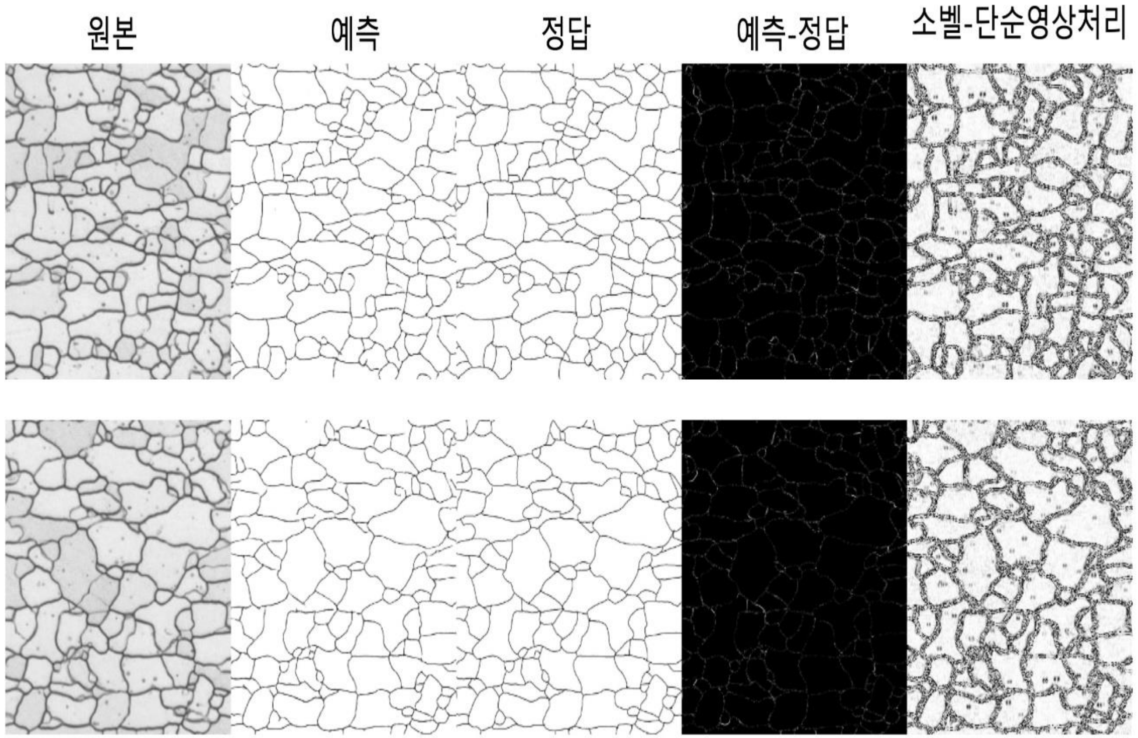



Figure 1: PatchGAN discriminator. Each value of the output matrix represents the probability of whether the corresponding image patch is real or it is artificially generated.

https://arxiv.org/abs/1803.07422

# 윤곽영상생성




생성자로,
RISA 적용 개선 U-Net 사용
SOTA 구조 적용
- Inception Module, Residual Block,
- Xception (depth-wise separable convolution)

# CycleGAN

## Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks (ICCV 2017)

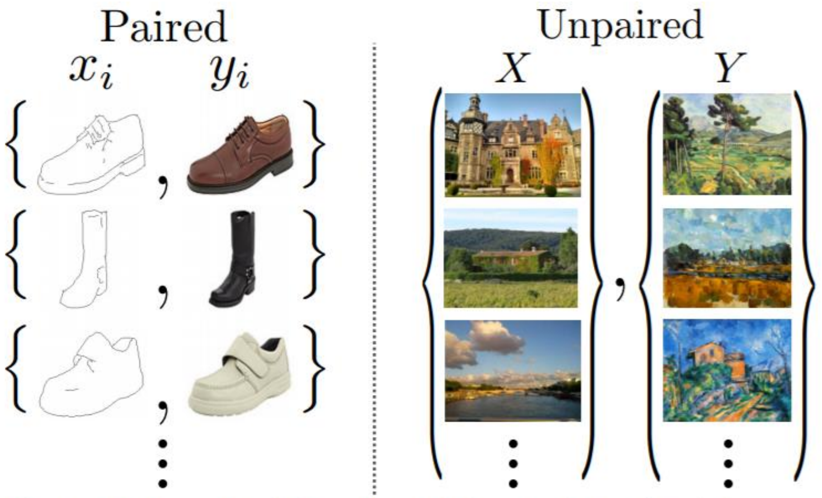
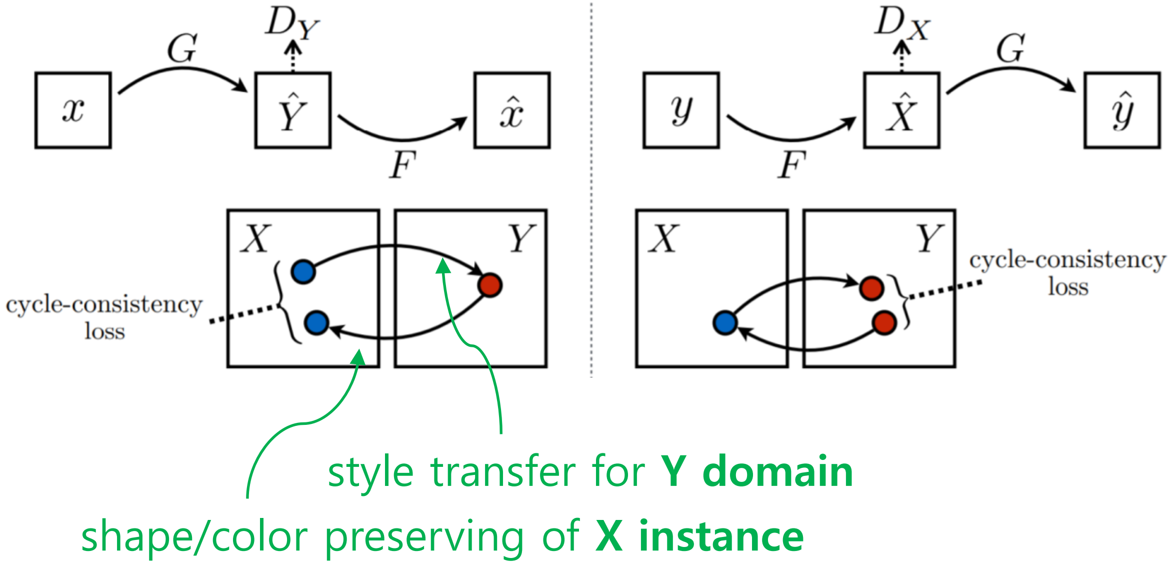



Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between $x_i$ and $y_i$ exists [21]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}$ ($y_j \in Y$), with no information provided as to which $x_i$ matches which $y_j$.




$$Loss_{x \to y} = \mathbb{E}_y[log(D_y(y))] + \mathbb{E}_x[log(1 - D_y(G(x)))] + \mathbb{E}_x[\|F(G(x)) - x\|_1]$$

$$Loss_{y \to x} = \mathbb{E}_x[log(D_x(x))] + \mathbb{E}_y[log(1 - D_x(F(y)))] + \mathbb{E}_y[\|G(F(y)) - y\|_1]$$

$$Loss_{cycleGAN} = Loss_{x \to y} + Loss_{y \to x}$$

NVidia

**ProgressiveGAN(2017)**

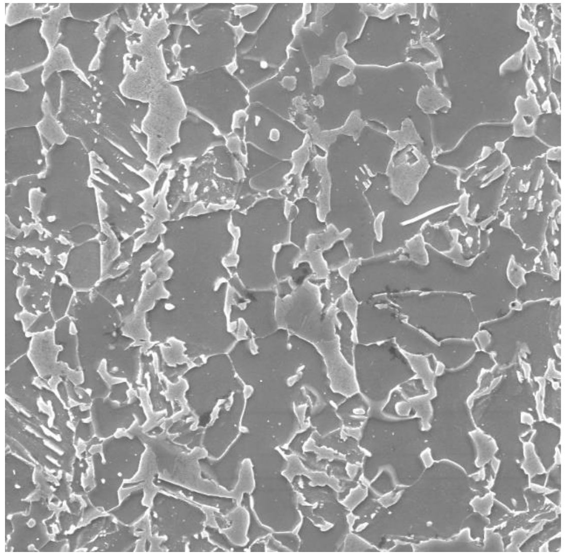
**StyleGAN(2018)**

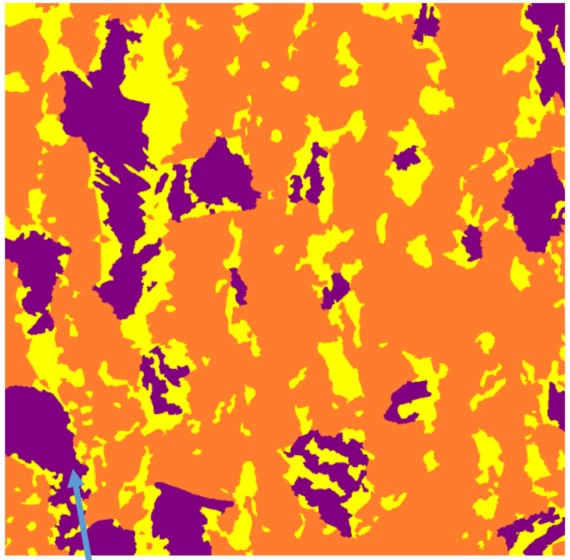
**StyleGAN2(2020)**

https://developer.nvidia.com/blog/synthesizing-high-resolution-images-with-stylegan2/

# Label-Image Pair Generation

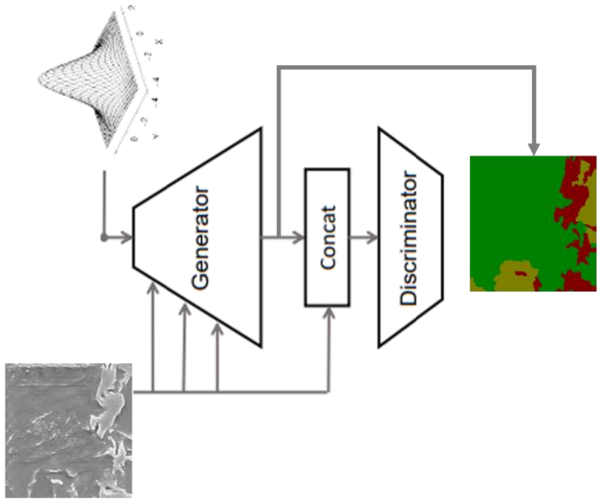
input

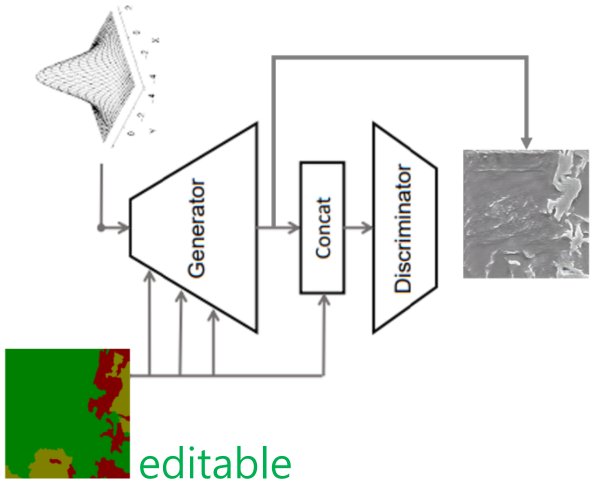




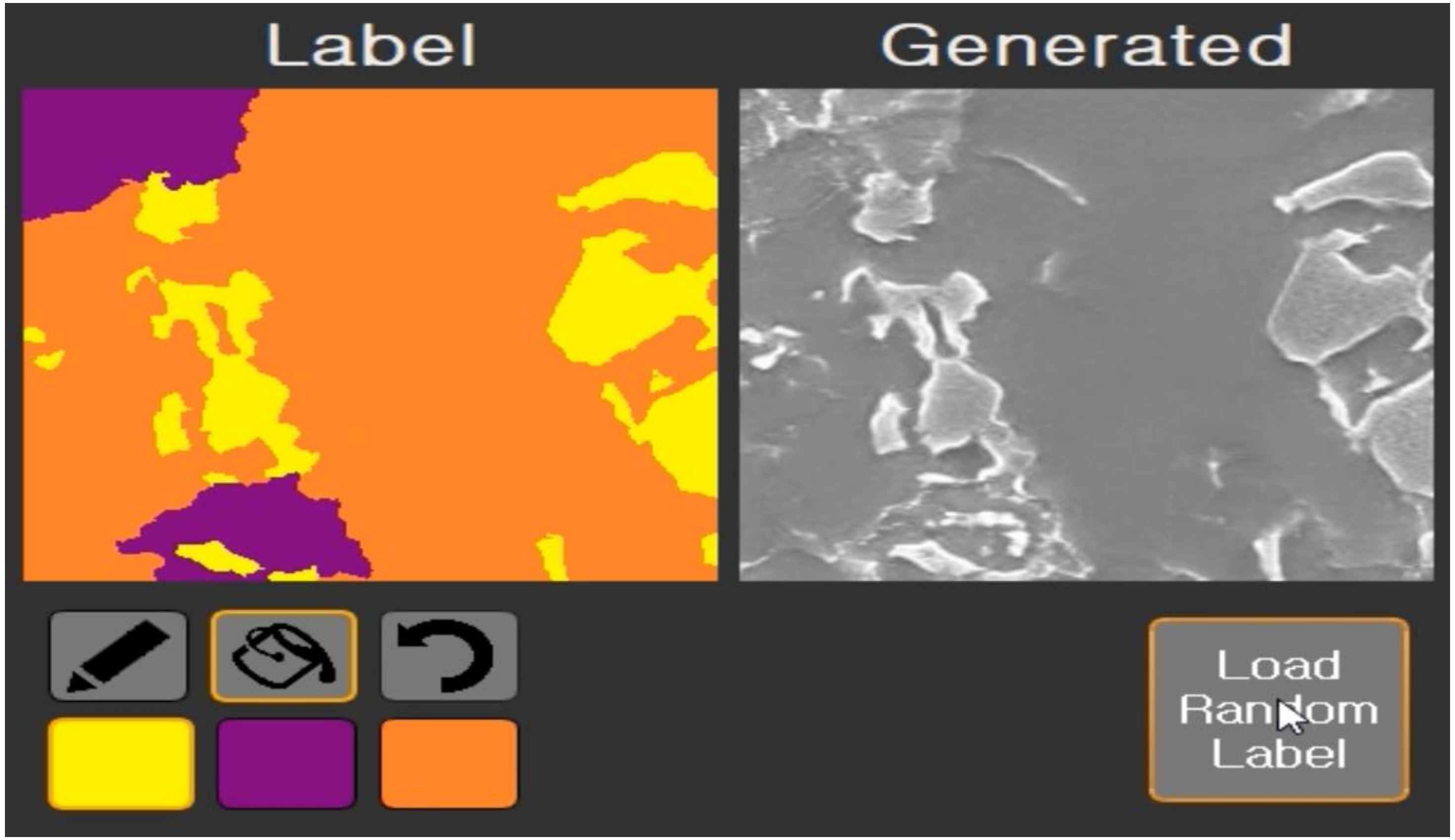
label

구분하기 어려운 레이블




editable


Label
Generated
Load
Random
Label

# 감사합니다.


- nonezero@kumoh.ac.kr