쉽게 이해하는 (?)

Generative Adversarial Networks

최종현

Machine Learning의 종류

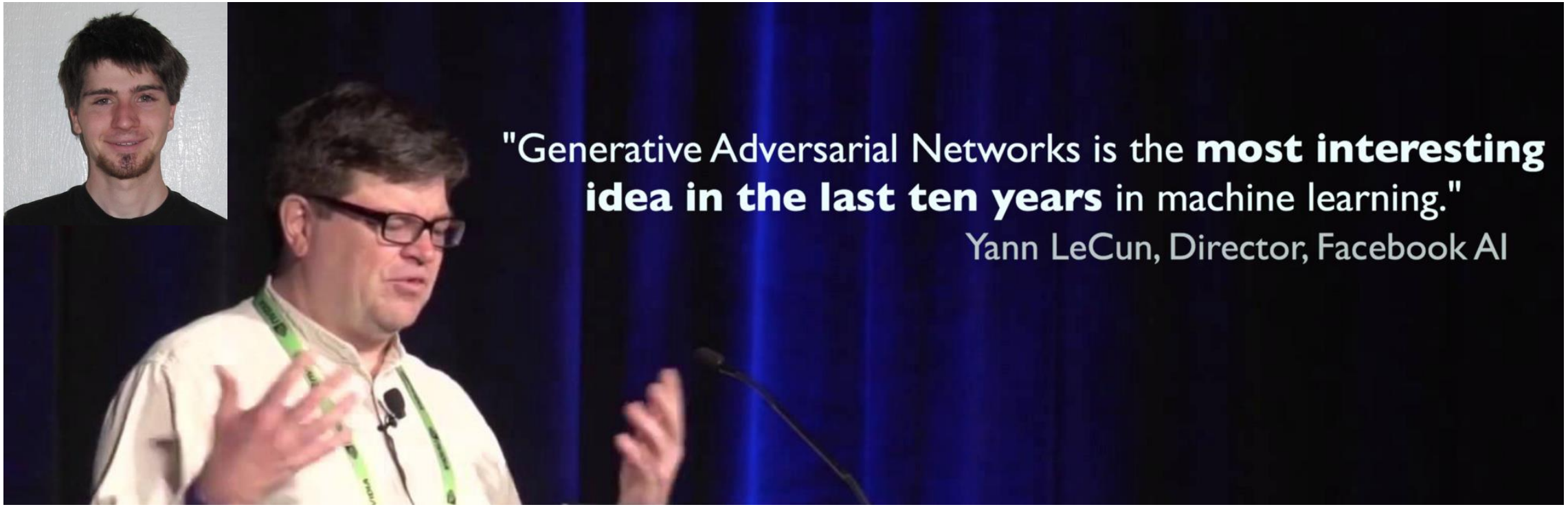
- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

- **(Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)**

From **Yann Lecun**, (NIPS 2016)

What are GANs?

- ✓ Ian Goodfellow(2014)가 제안한 Neural Network Model
- ✓ **Unsupervised Learning**(비지도학습) 알고리즘
- ✓ Yann Lecun 교수가 극찬한 바로 그 알고리즘!



GANs 예시

DCGAN

BEGAN

GANs 구조

Generative Adversarial Networks

생성의 VS 적대적인

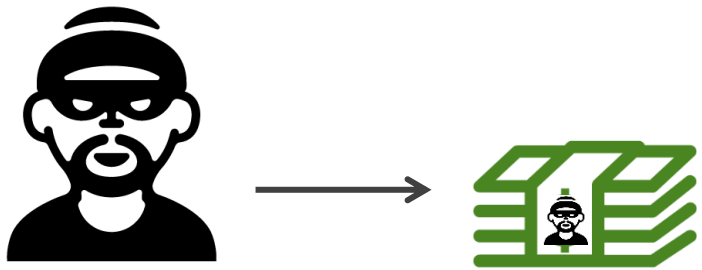
- The discriminator learns using traditional supervised learning techniques, dividing inputs into two classes (real or fake).
- The generator is trained to fool the discriminator.
- We can think of the generator as being like a counterfeiter, trying to make fake money, and the discriminator as being like police, trying to allow legitimate money and catch counterfeit money.
- To succeed in this game, the counterfeiter must learn to make money that is indistinguishable from genuine money, and the generator network must learn to create samples that are drawn from the same distribution as the training data.

* Text from NIPS 2016 Tutorial: Generative Adversarial Networks, Ian Goodfellow, 2016

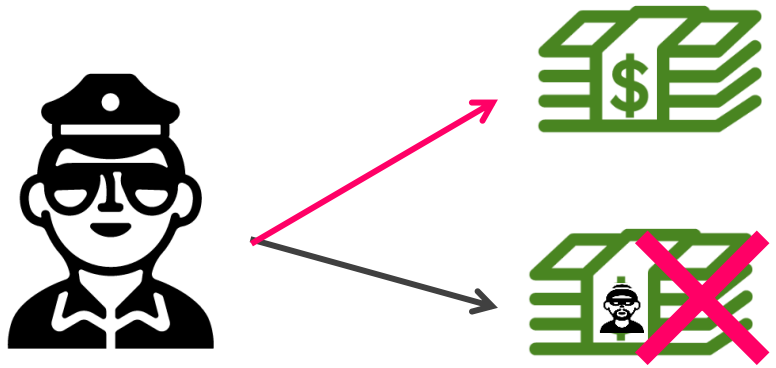
GANs 구조 (cont'd)

Generative Adversarial Networks

생성의 vs 적대적인



Goal: 진짜 지폐와 최대한 비슷한 위조지폐를 만들자!

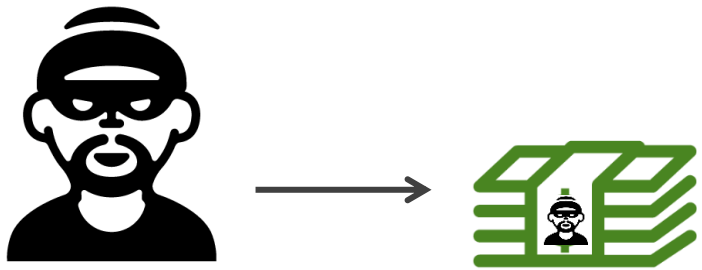


Goal: 위조지폐와 진짜 지폐를 잘 구별해내자!

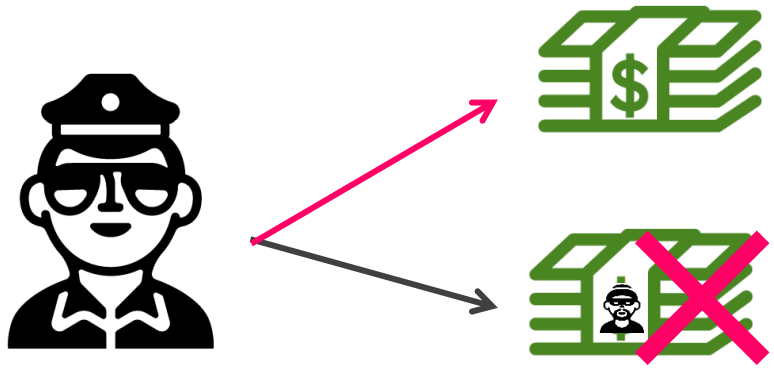
GANs 구조 (cont'd)

Generative Adversarial Networks

생성의 vs 적대적인



VS



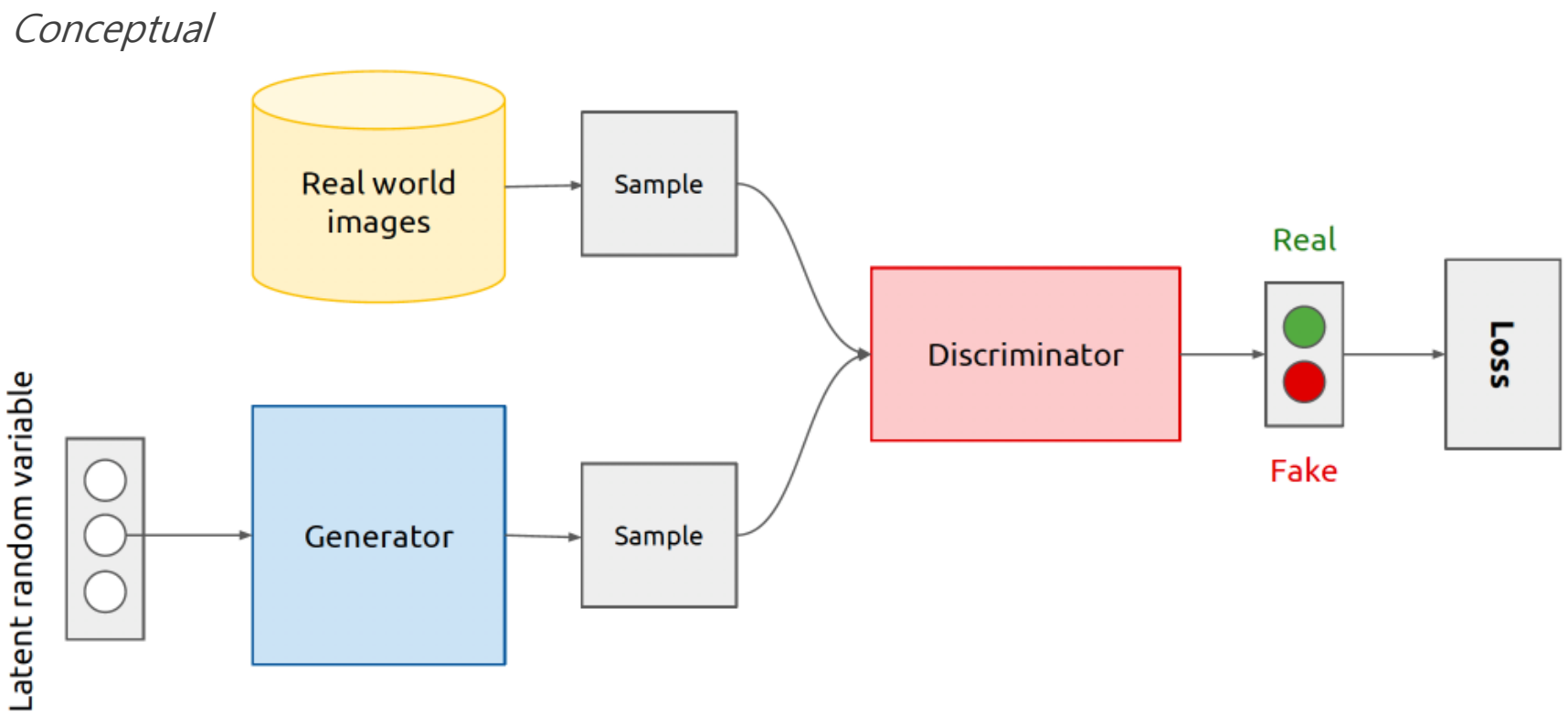
Goal: 진짜 지폐와 최대한 비슷한 위조지폐를 만들자!

Goal: 위조지폐와 진짜 지폐를 잘 구별해내자!

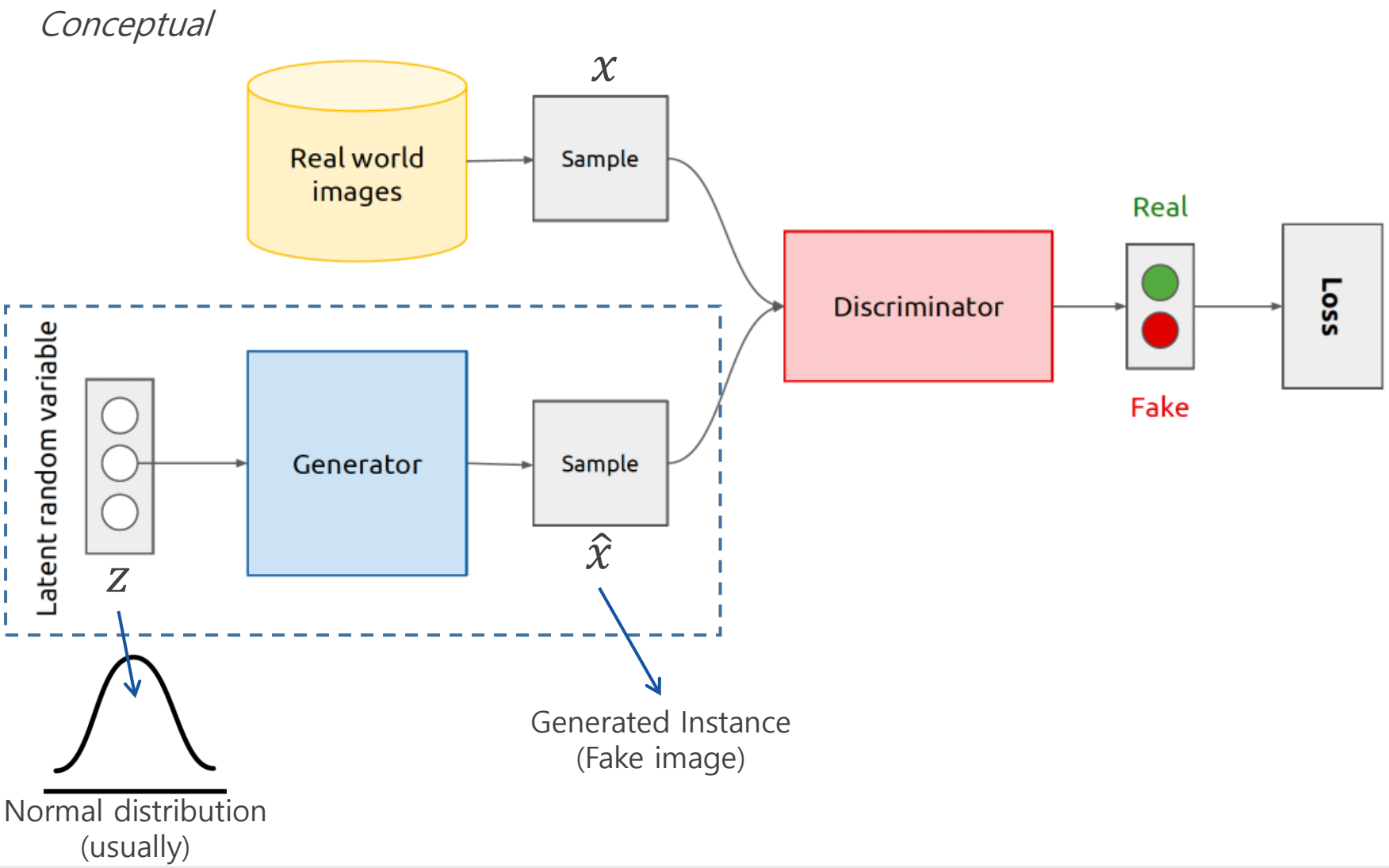
Generator

Discriminator

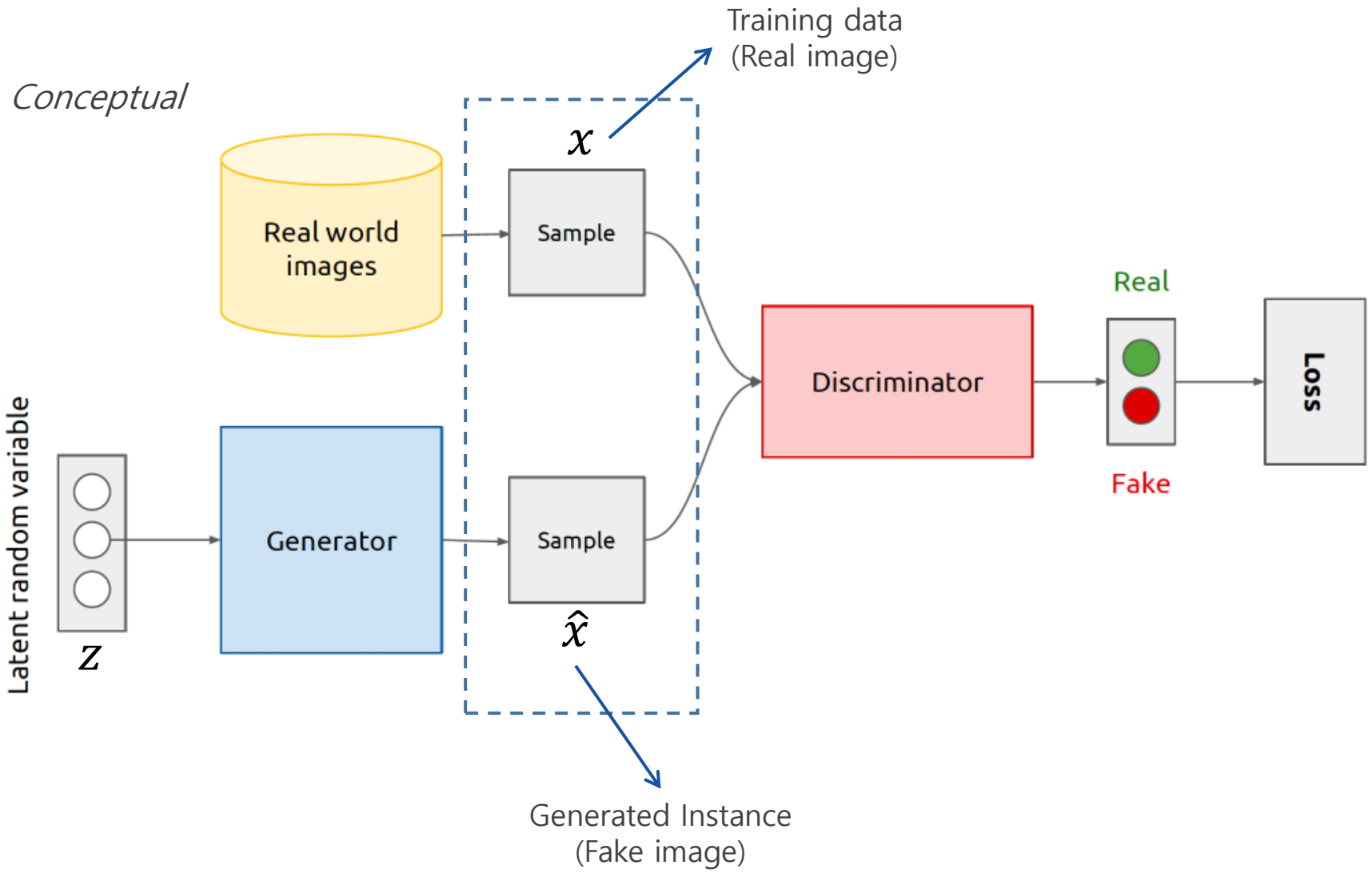
GANs 구조 (cont'd)



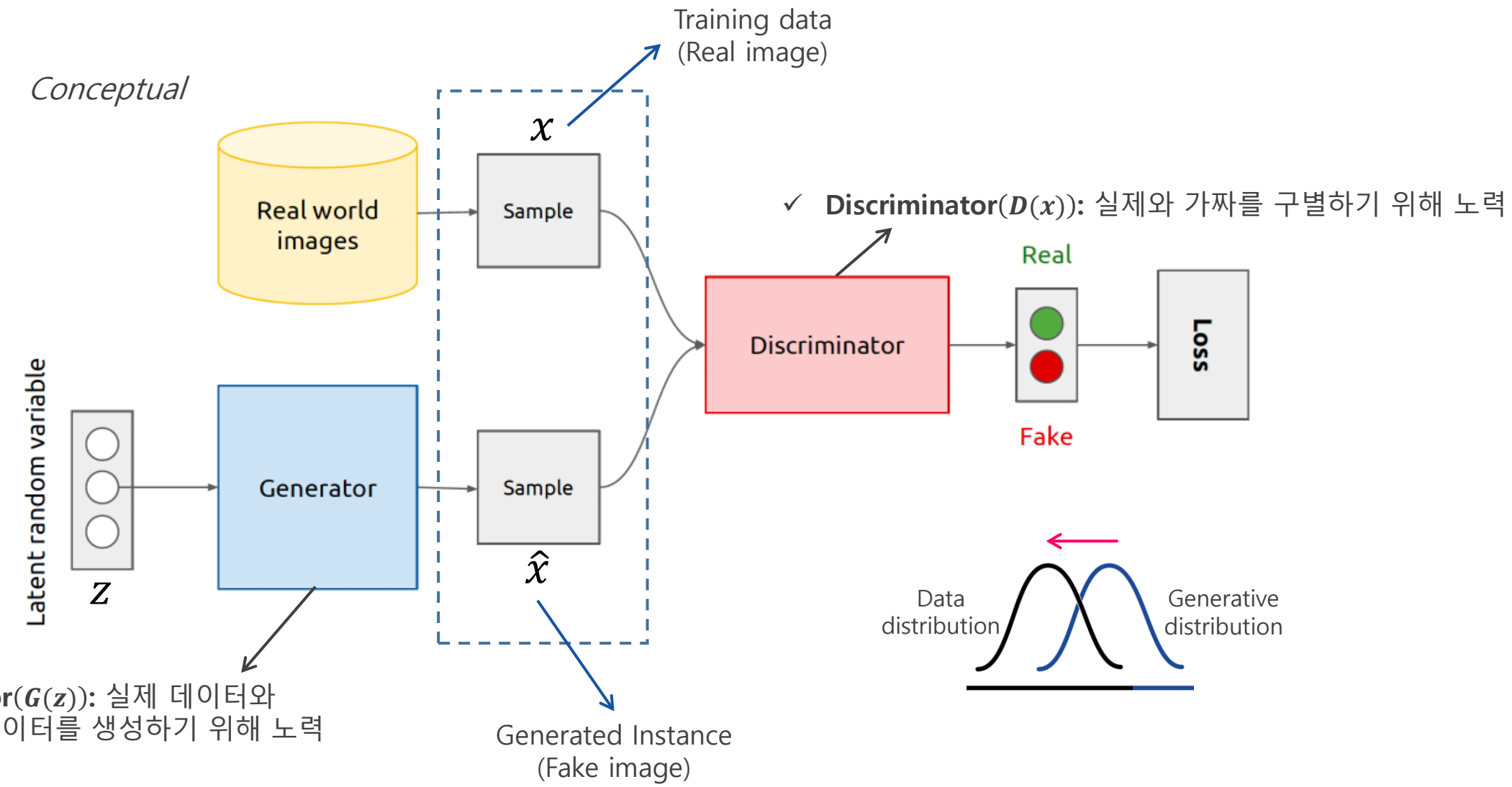
GANs 구조 (cont'd)



GANs 구조 (cont'd)



GANs 구조 (cont'd)

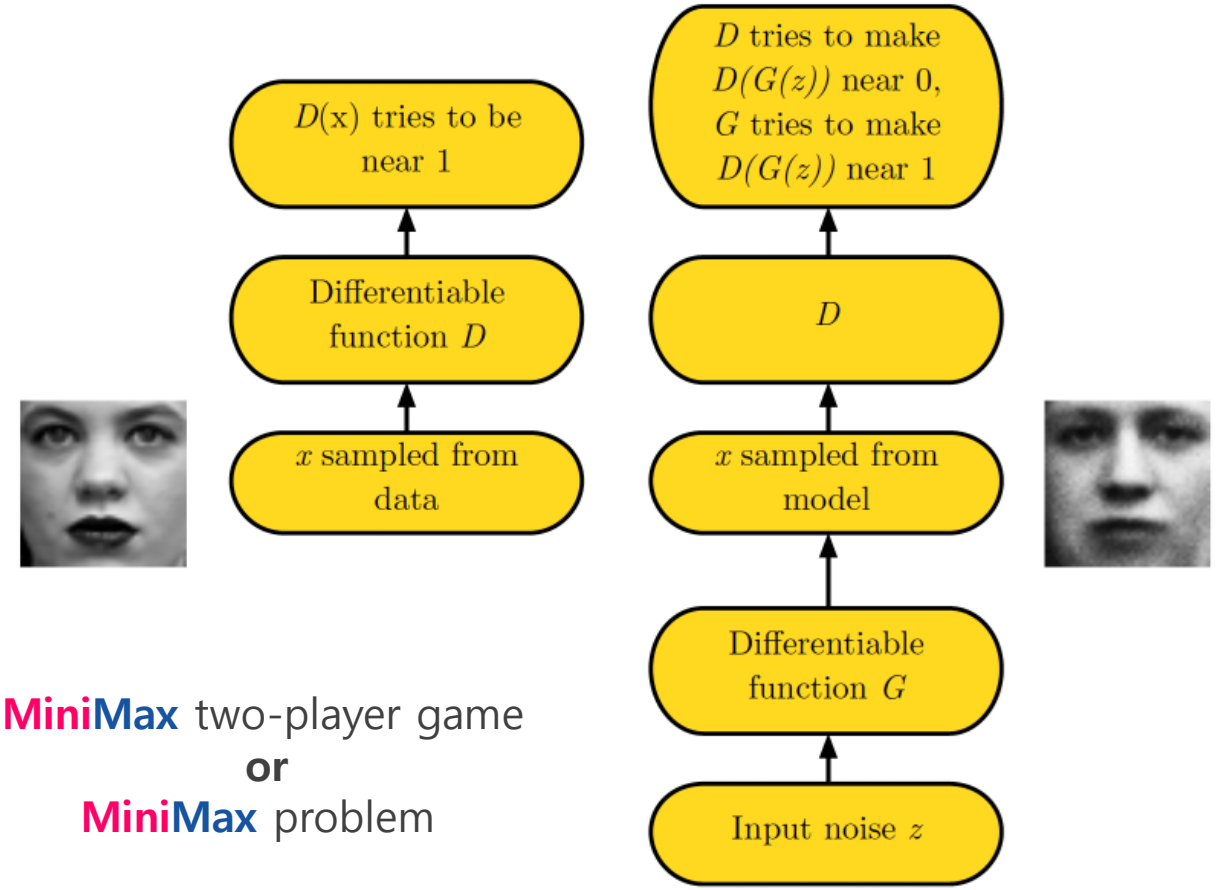


✓ **Discriminator**$(D(x))$**:** 실제와 가짜를 구별하기 위해 노력

✓ **Generator**$(G(z))$**:** 실제 데이터와 비슷한 데이터를 생성하기 위해 노력

GANs 식

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



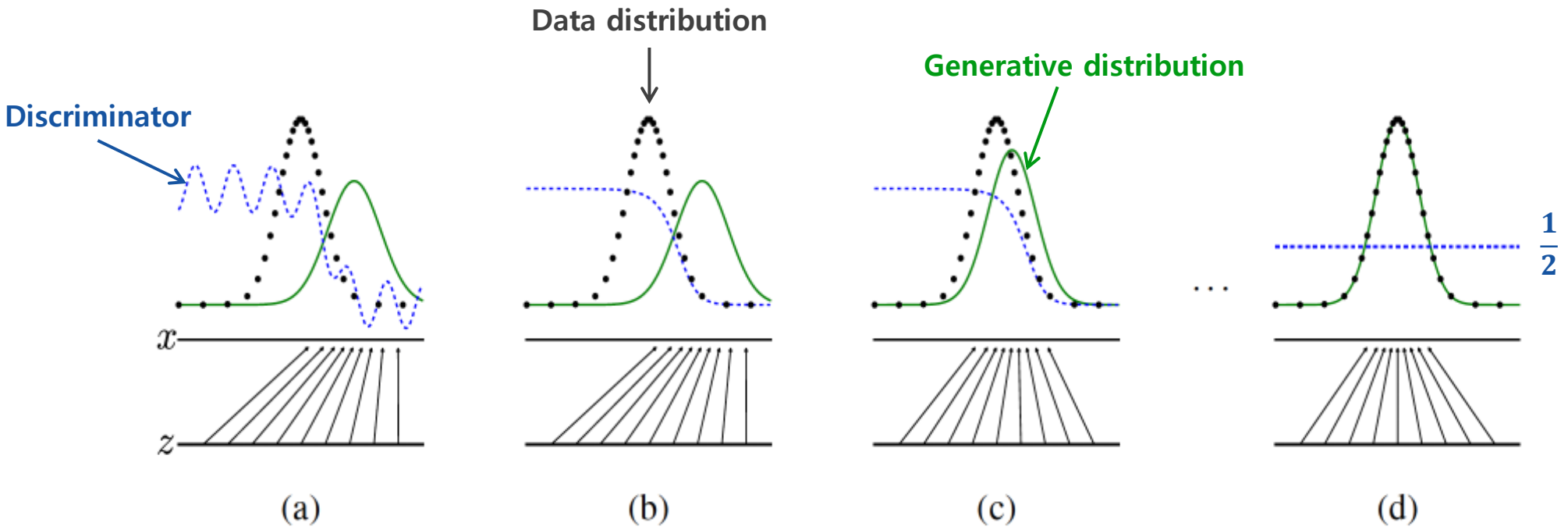
MiniMax two-player game
or
MiniMax problem

* Figure adopted from NIPS 2016 Tutorial: Generative Adversarial Networks, Ian Goodfellow, 2016

GANs 식 (cont'd)

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

- ✓ **Discriminator**($\boldsymbol{D(x)}$): $\boldsymbol{D(x) = 1, D(G(z)) = 0}$ → $V(D,G) = 0$ 로 만드는 것이 **maximize** 한 값
- ✓ **Generator**($\boldsymbol{G(z)}$): $\log\left(1 - D(G(z))\right)$의 minimize → $D(G(z))$의 **maximize**



* Figure adopted from Generative Adversarial Nets, Ian Goodfellow et al. 2014

GANs 식 (cont'd)

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

- ✓ The minimax problem of GAN has a global optimum at $\boldsymbol{p_g = p_{data}}$
- ✓ 즉, $\boldsymbol{p_g = p_{data}}$ 일 때, 최적의 값을 가짐

For G fixed, the optimal discriminator D is

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

$proof.$

$$V(G,D) = \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log\big(1-D(G(z))\big)dz$$
$$= \int_x p_{data}(x)log(D(x)) + p_g(x)log(1-D(x))dx$$

For any $(a,b) \in \mathbb{R}^2 \setminus \{0,0\}$, the function $y \to a\log(y) + b\log(1-y)$ achieves its maximum in $[0,1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. □

* Figure adopted from Generative Adversarial Nets, Ian Goodfellow et al. 2014

GANs 식 (cont'd)

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

- ✓ The minimax problem of GAN has a global optimum at $\boldsymbol{p_g = p_{data}}$
- ✓ 즉, $\boldsymbol{p_g = p_{data}}$ 일 때, 최적의 값을 가짐

Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D_G^*(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

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

and that by subtracting this expression from $C(G) = V(D_G^*, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data generating process. □

* Figure adopted from Generative Adversarial Nets, Ian Goodfellow et al. 2014

GANs 소스코드 - with MNIST data

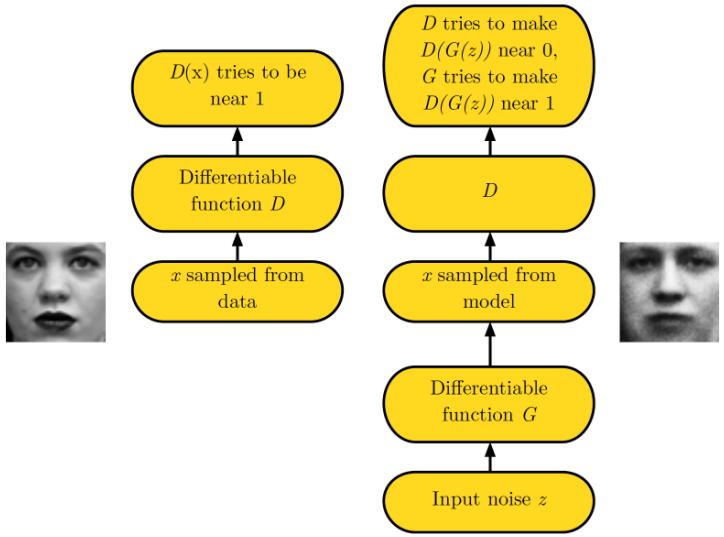
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

```
# 실제 데이터를 입력값으로 사용
X = tf.placeholder(tf.float32, [None, n_input])
# noise Z를 입력값으로 사용
Z = tf.placeholder(tf.float32, [None, n_noise])

# noise를 이용해 fake image 생성
G = generator(Z)
# noise를 이용해 생성한 이미지가 real/fake 인지 확인
D_fake = discriminator(G) # = D(G(Z))
# real image를 이용하여 판별한 값
D_real = discriminator(X) # = D(X)

## cost(loss) function 계산
loss_D = -tf.reduce_mean(tf.log(D_real) + tf.log(1-D_fake))
loss_G = -tf.reduce_mean(tf.log(D_fake))
```

* https://github.com/golbin/TensorFlow-Tutorials

GANs 소스코드 - with MNIST data

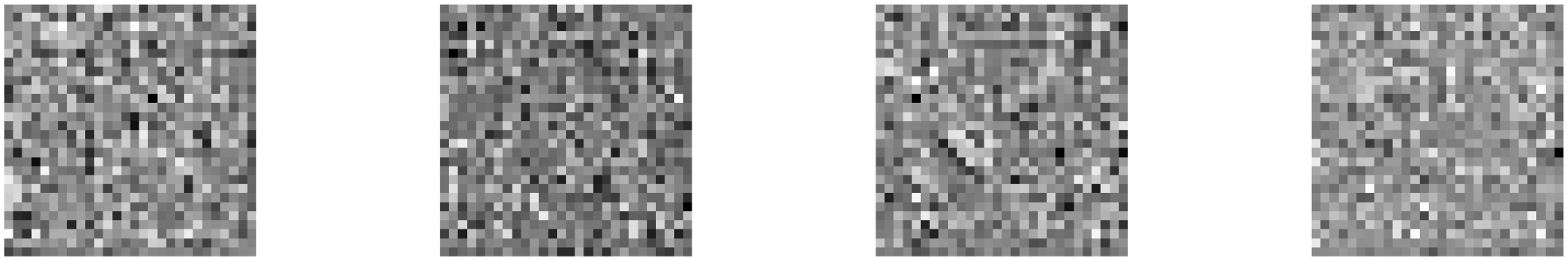
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$




```python
X = tf.placeholder(tf.float32, [None, n_input])
# 노이즈와 실제 이미지에, 그에 해당하는 숫자에 대한 정보를 넣어주기 위해 사용
Y = tf.placeholder(tf.float32, [None, n_class])
Z = tf.placeholder(tf.float32, [None, n_noise])

# Generator와 Discriminator에 Y 즉 labels(class) 정보를 추가하여
# labels 정보에 해장하는 이미지를 생성할 수 있도록 유도
G = generator(Z, Y) # = G(Z)
D_real = discriminator(X,Y) # = D(X)
D_fake = discriminator(G, Y, True) # = D(G(Z))

## loss function 부분
## Real img를 구분하는 D_real(D(X))의 값은 1에 가깝도록 해줌
## Fake img를 구분하는 D_fake(D(G(Z)))의 값은 0에 가깝도록 해줌
loss_D_real = tf.reduce_mean(
                    tf.nn.sigmoid_cross_entropy_with_logits(
                        logits=D_real, labels=tf.ones_like(D_real)))

loss_D_fake = tf.reduce_mean(
                    tf.nn.sigmoid_cross_entropy_with_logits(
                        logits=D_fake, labels=tf.zeros_like(D_fake)))

# loss_D_real 과 loss_D_fake를 더한 뒤 minimize 해줌
loss_D = loss_D_real + loss_D_fake

# Fake img를 Real img에 가깝게 만들어 주기 위해 D_fake (D(G(Z)))를 1에 가깝도록 해줌
loss_G = tf.reduce_mean(
                tf.nn.sigmoid_cross_entropy_with_logits(
                    logits=D_fake, labels=tf.ones_like(D_fake)))
```

* https://github.com/golbin/TensorFlow-Tutorials

GANs 예제 - MNIST data




THANK YOU