Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Image-to-Image Translation

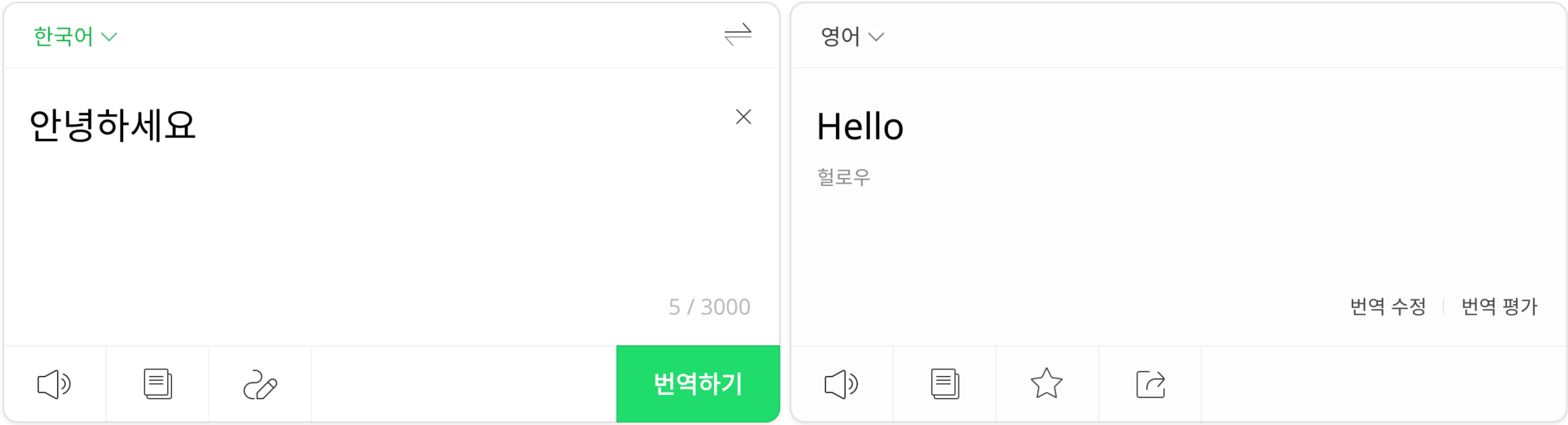


Image-to-Image Translation

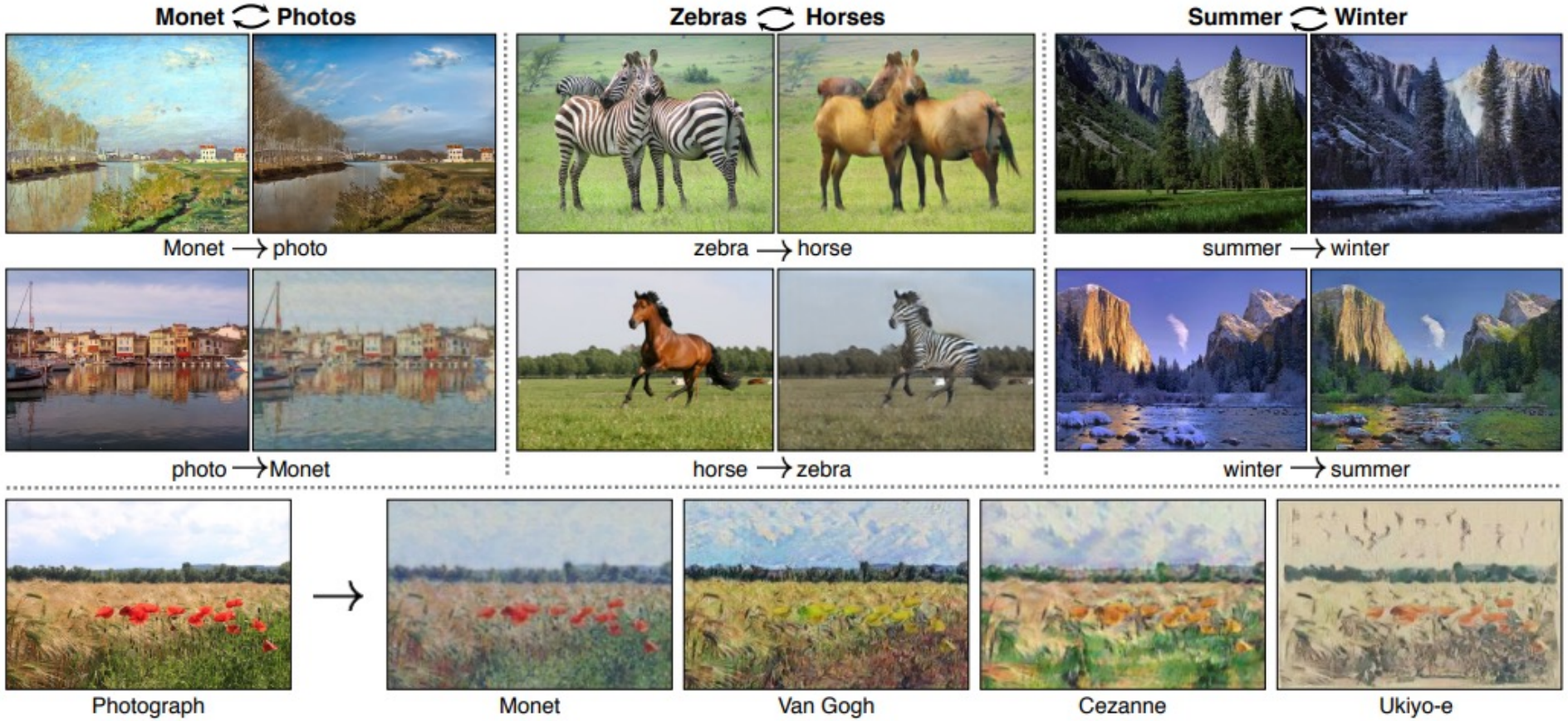
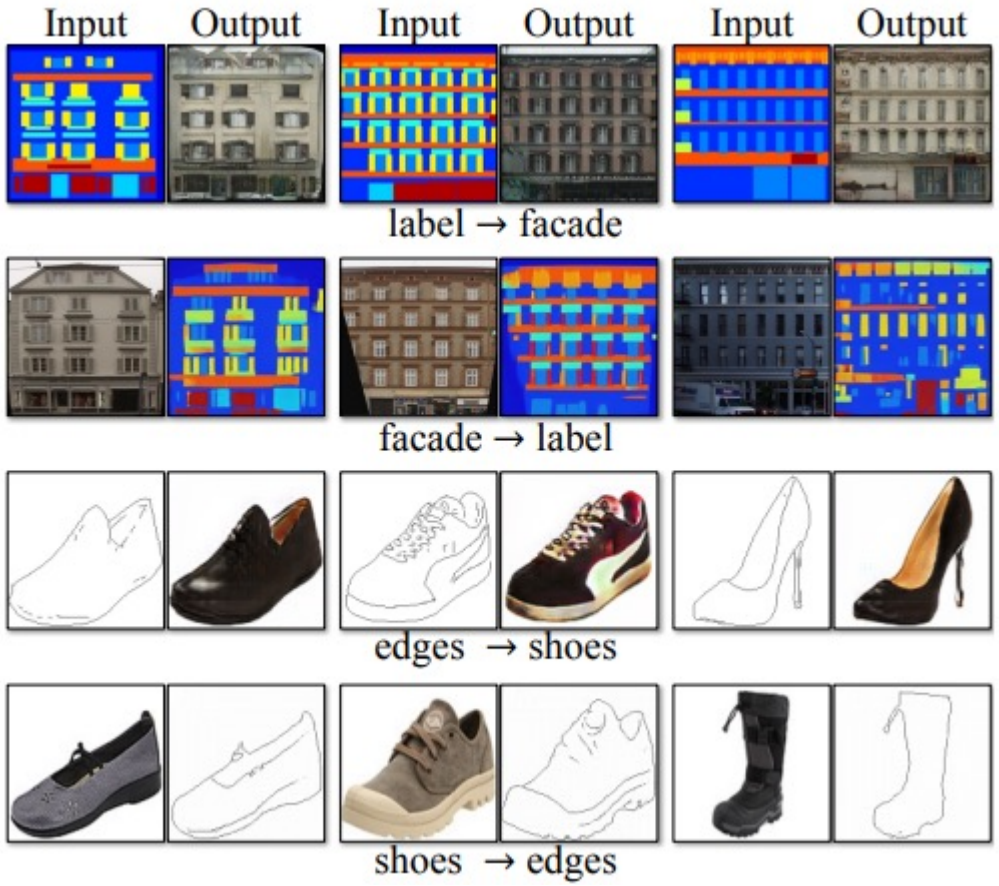
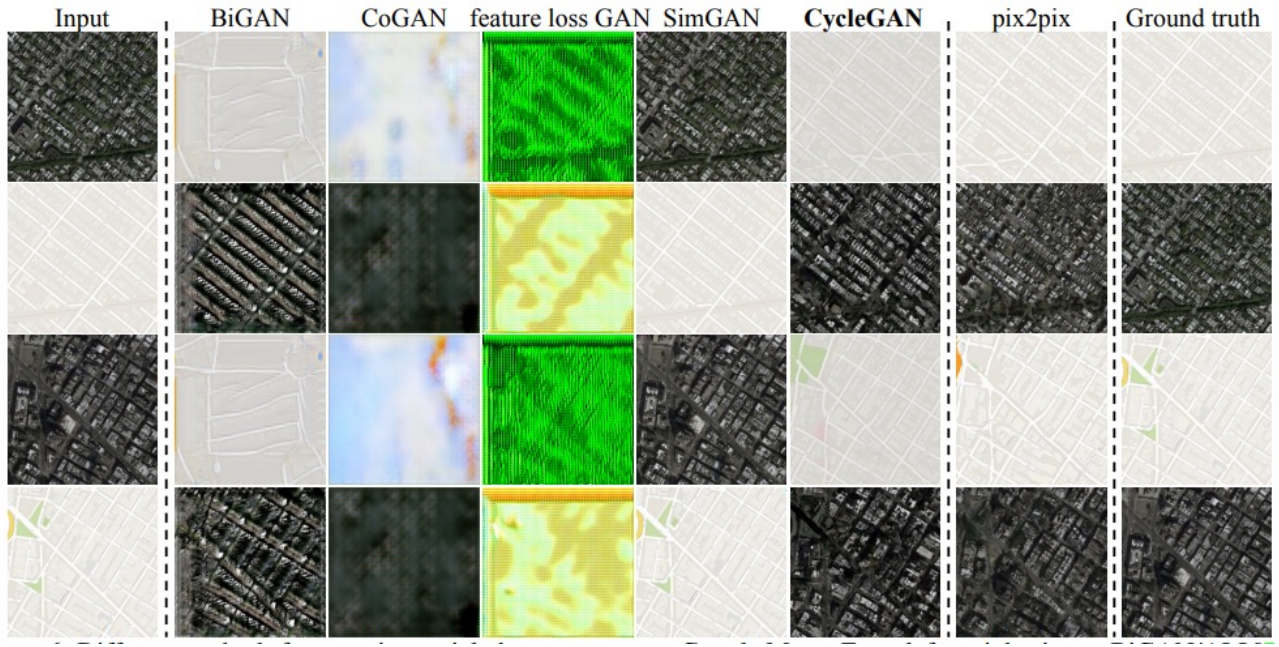
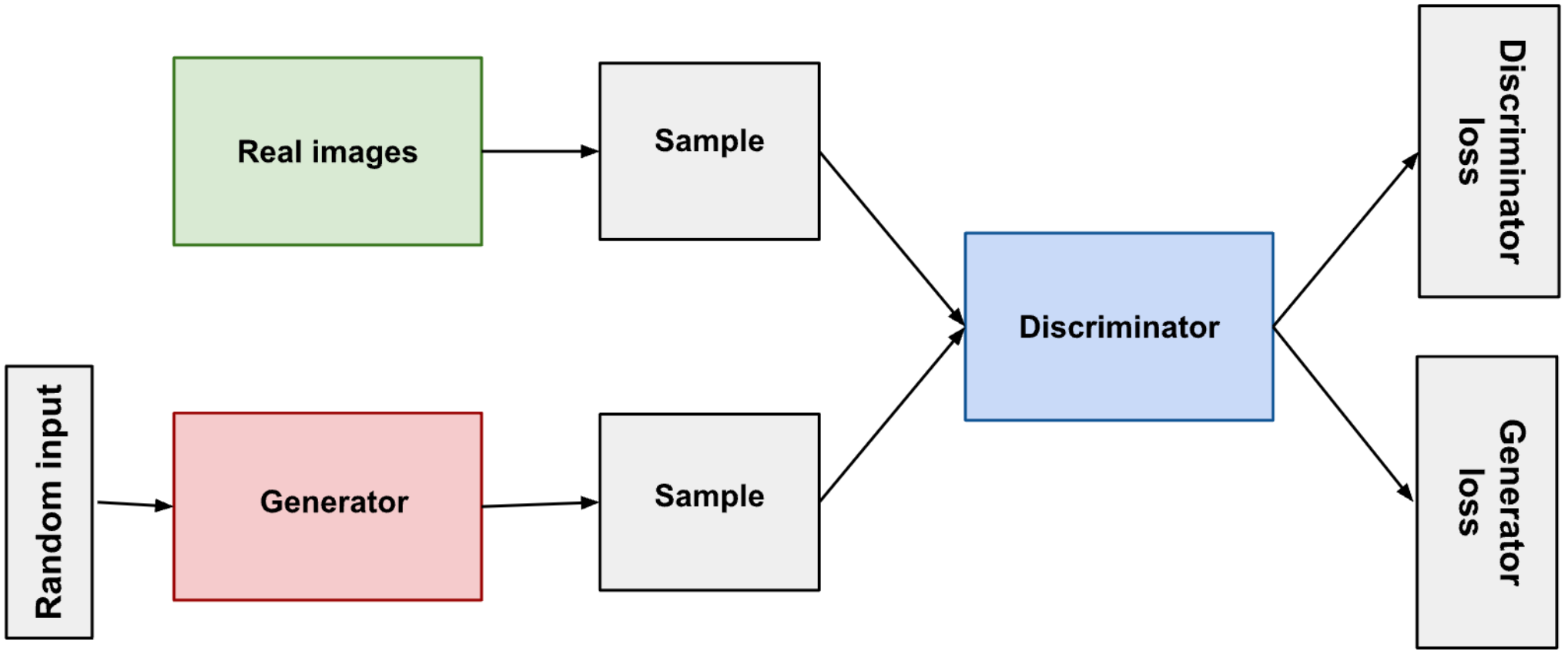


Image-to-Image Translation





GAN



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

GAN

for number of training iterations **do**

for k steps **do**

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

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

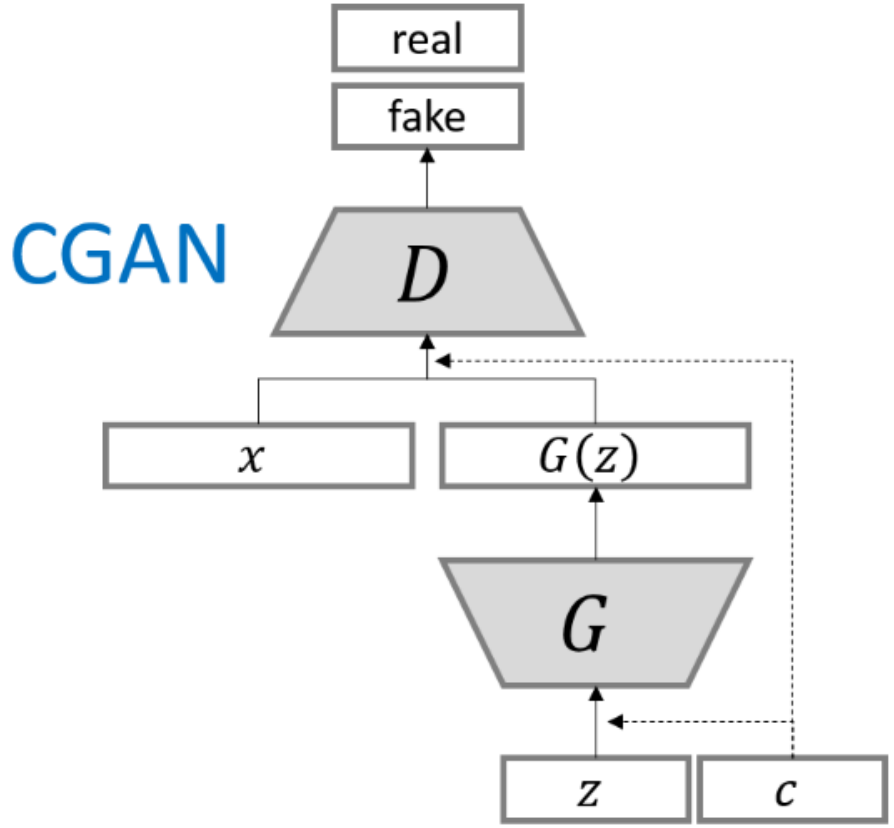
end for

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

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

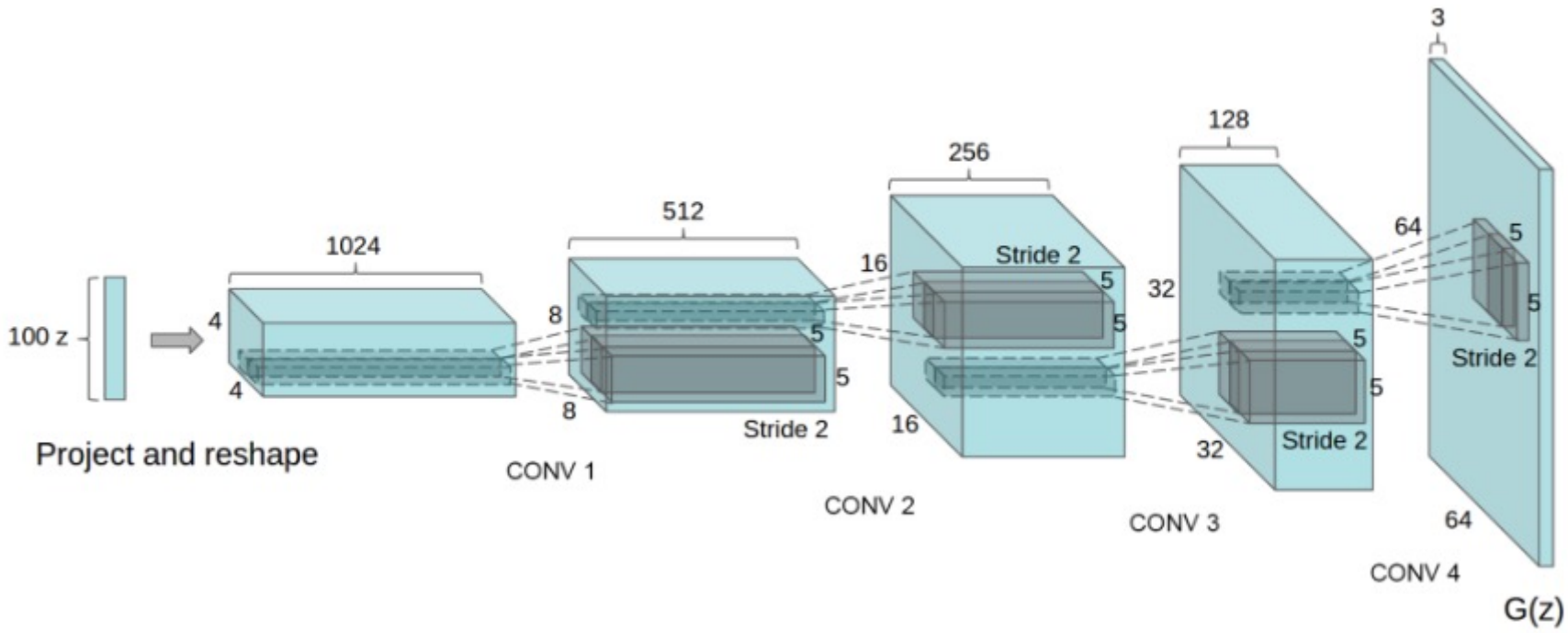
GAN - Mode Collapse

Conditional GAN

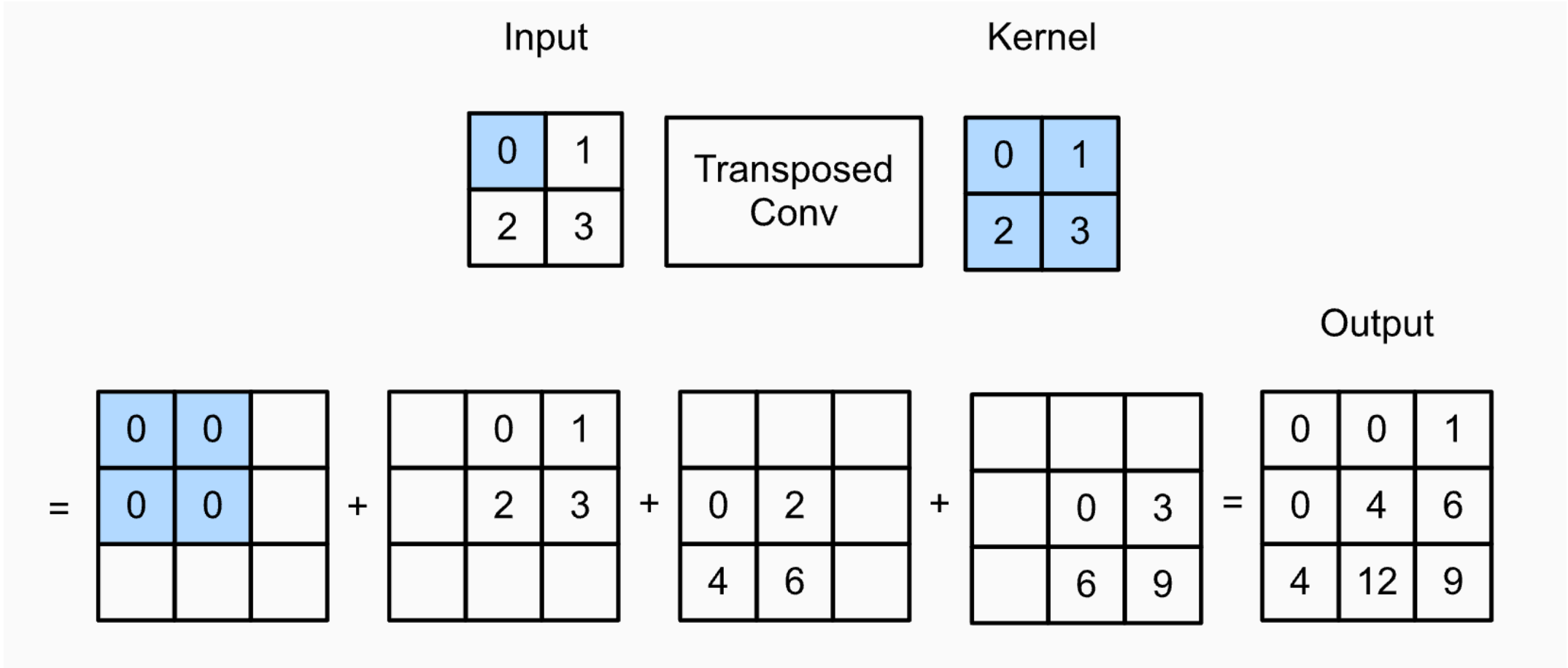


기본 GAN에 Condition Vector가 추가

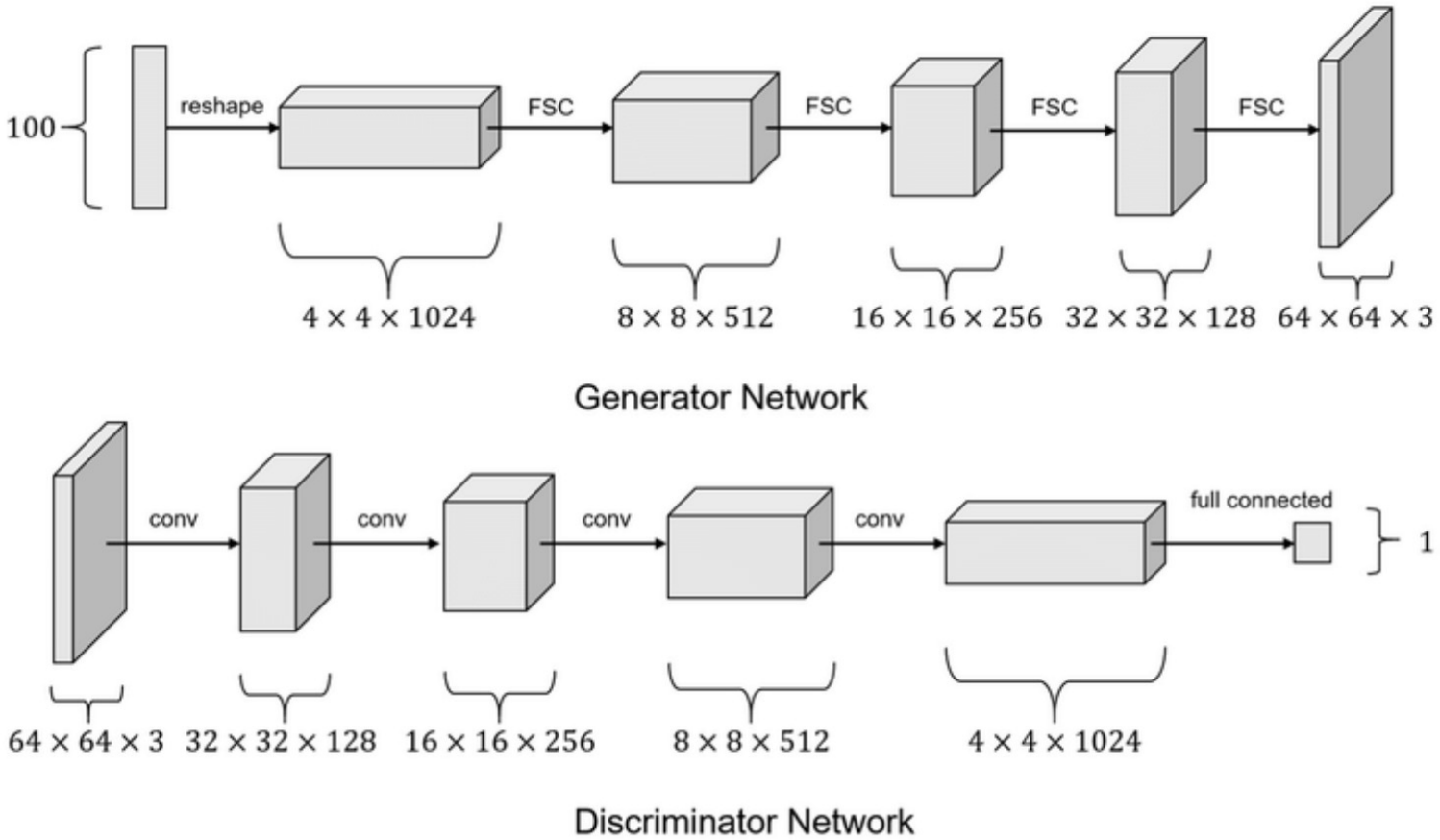
DCGAN


100 z
Project and reshape
4
4
1024
8
8
512
CONV 1
Stride 2
5
5
16
16
256
CONV 2
Stride 2
5
5
32
32
128
CONV 3
Stride 2
5
5
64
64
3
CONV 4
Stride 2
5
5
G(z)

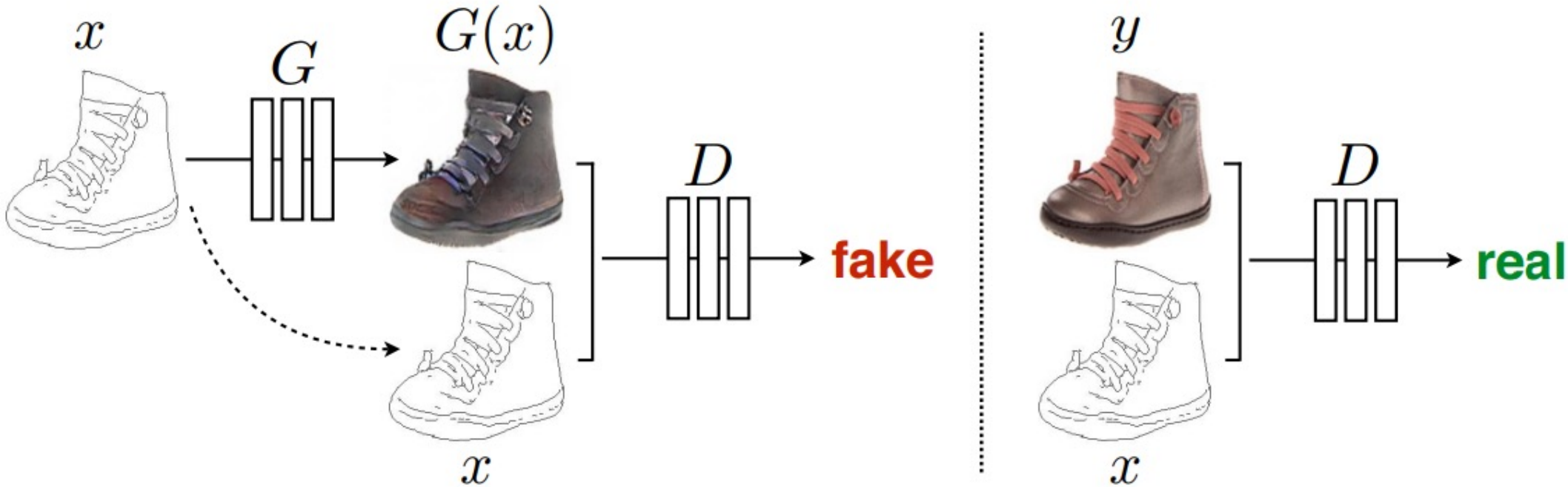
Tranpose Convolution



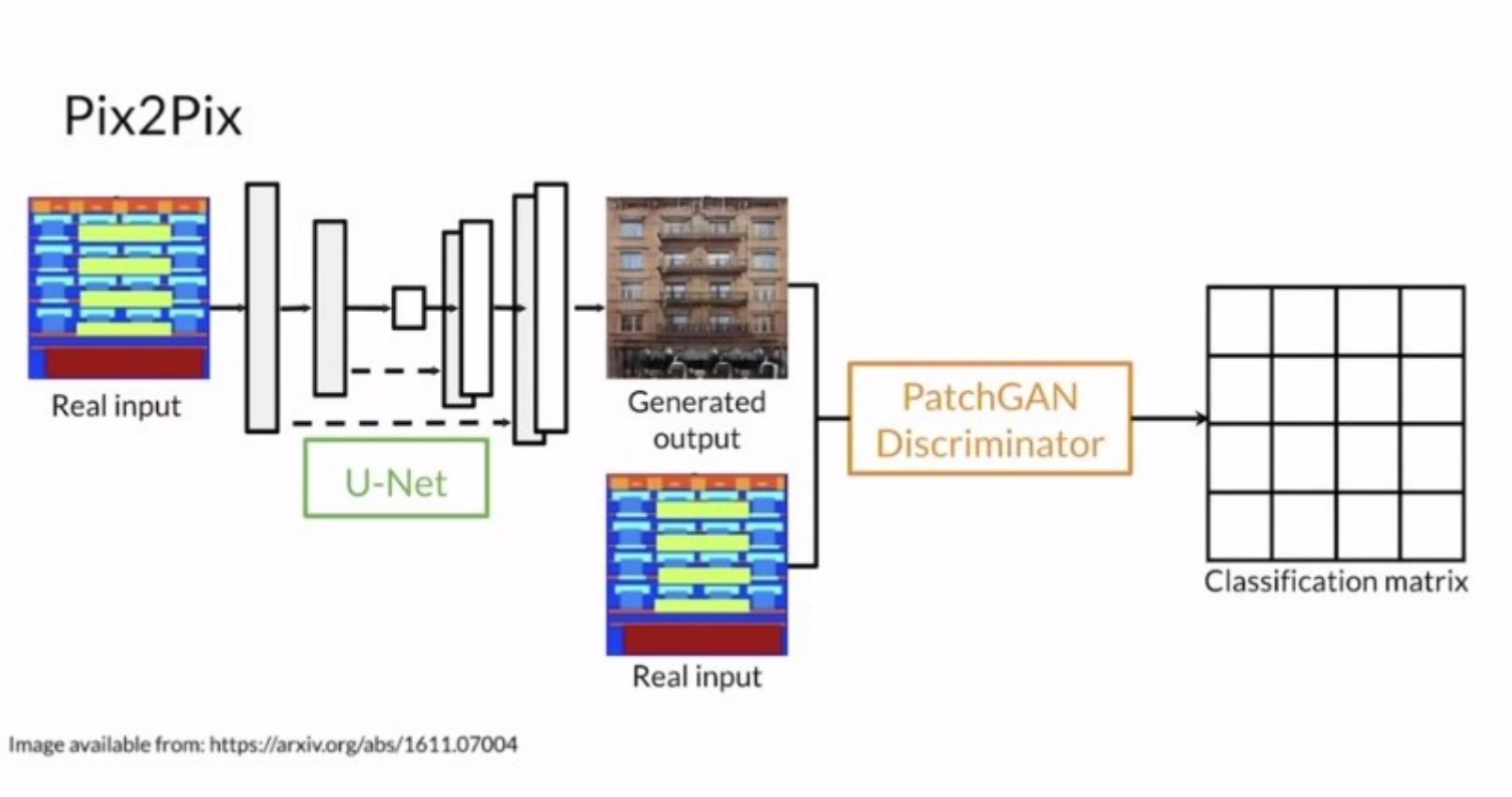
DCGAN


100
reshape
FSC
FSC
FSC
FSC
4 × 4 × 1024
8 × 8 × 512
16 × 16 × 256
32 × 32 × 128
64 × 64 × 3
Generator Network
conv
conv
conv
conv
full connected
1
64 × 64 × 3
32 × 32 × 128
16 × 16 × 256
8 × 8 × 512
4 × 4 × 1024
Discriminator Network

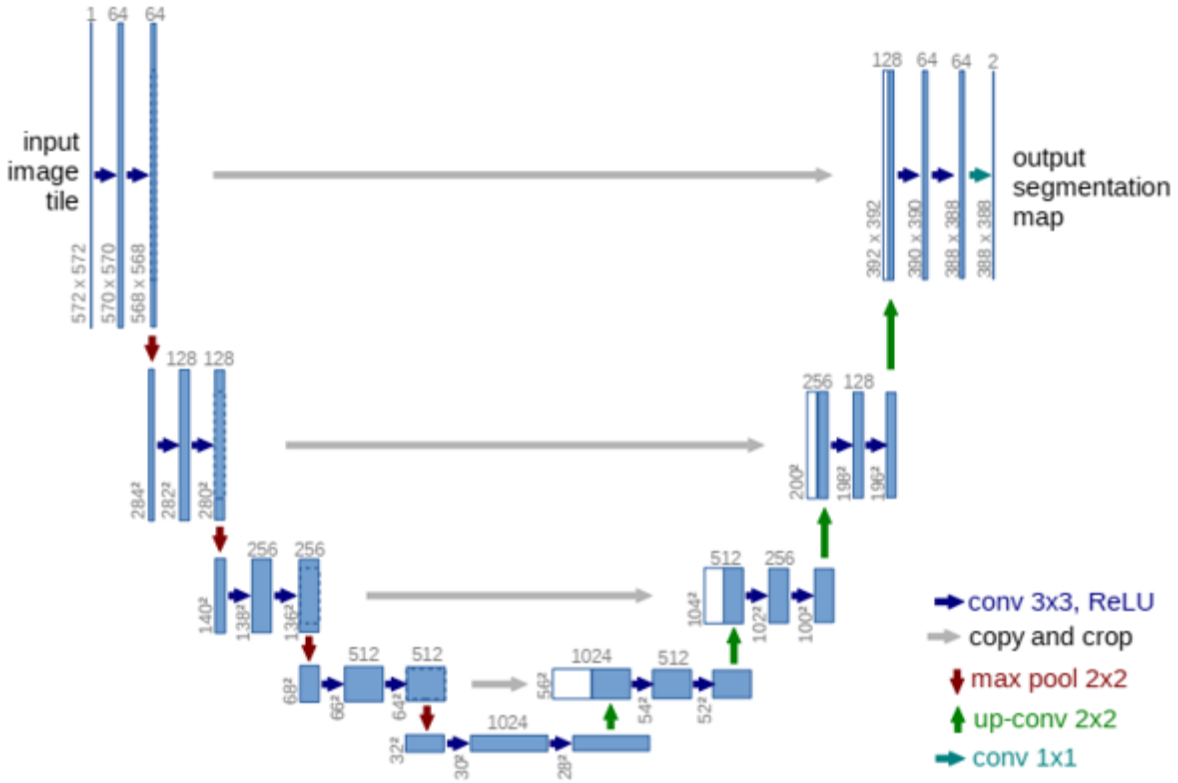
Pix2Pix



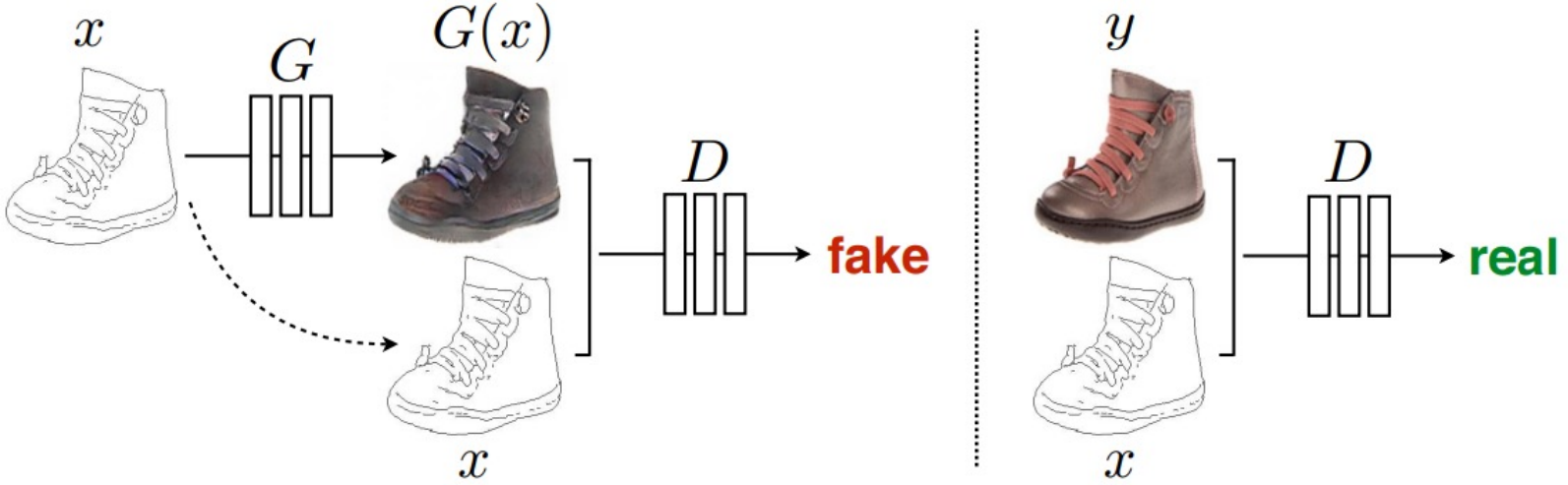
Pix2Pix


Pix2Pix
Real input
U-Net
Generated output
Real input
PatchGAN Discriminator
Classification matrix
Image available from: https://arxiv.org/abs/1611.07004

U-net


input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

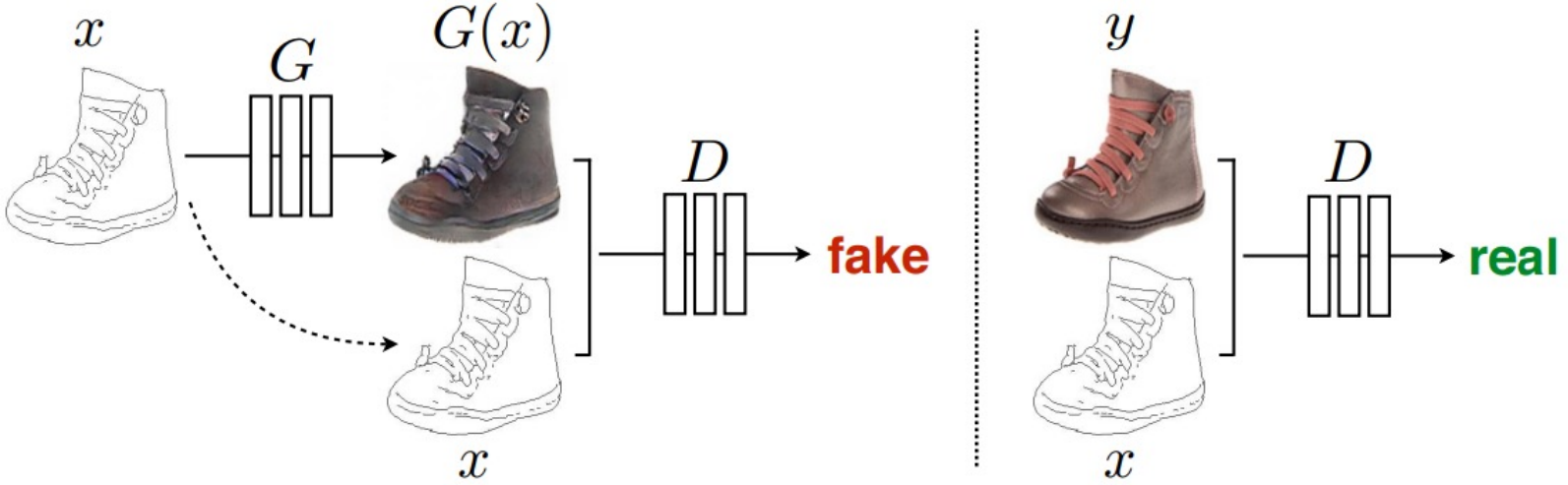
Pix2Pix Loss



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

-> Adversarial Loss

Pix2Pix Loss



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

-> Adversarial Loss

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

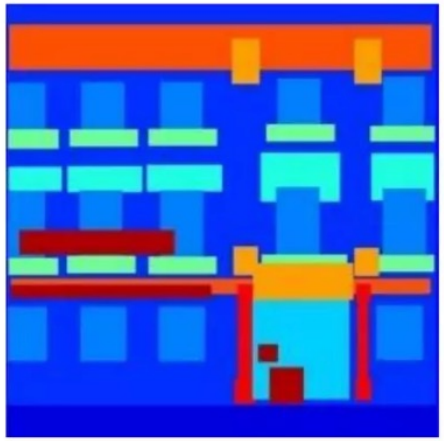
-> L1 Loss (GT 근처에 있게)

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

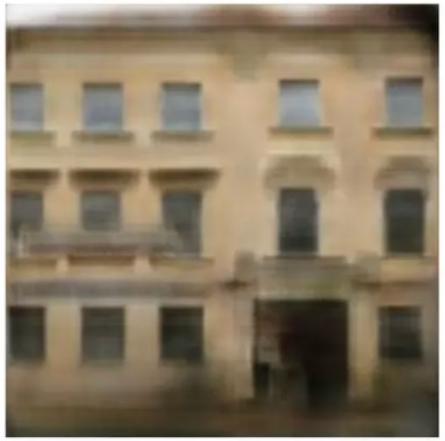
왜 L1, L2 Loss 안쓰지?

Loss: Minimize the difference between output *G(x) and the ground truth y*

$$\sum_{(x,y)} \|y - G(x)\|_1$$



Input



Output

Ground Truth

왜 L1, L2 Loss 안쓰지?

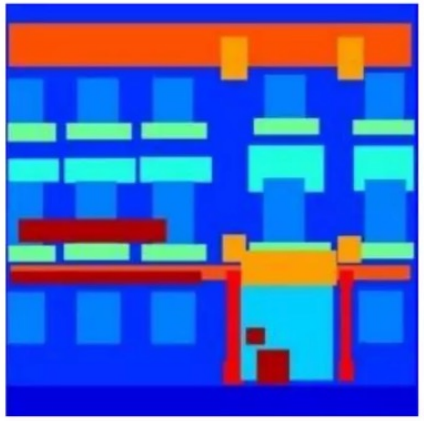
$$\sum_{(x,y)} \|y - G(x)\|_1$$

Input

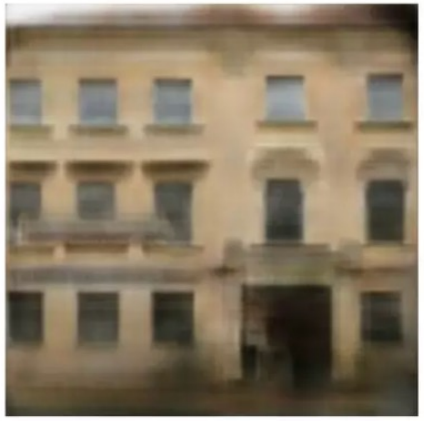
Output

Ground Truth

왜 L1, L2 Loss 안쓰지?



Input



Output

Ground Truth

모델은 Ground Truth의 색상을 모른다.

따라서 항상 Loss를 최소화 시키는

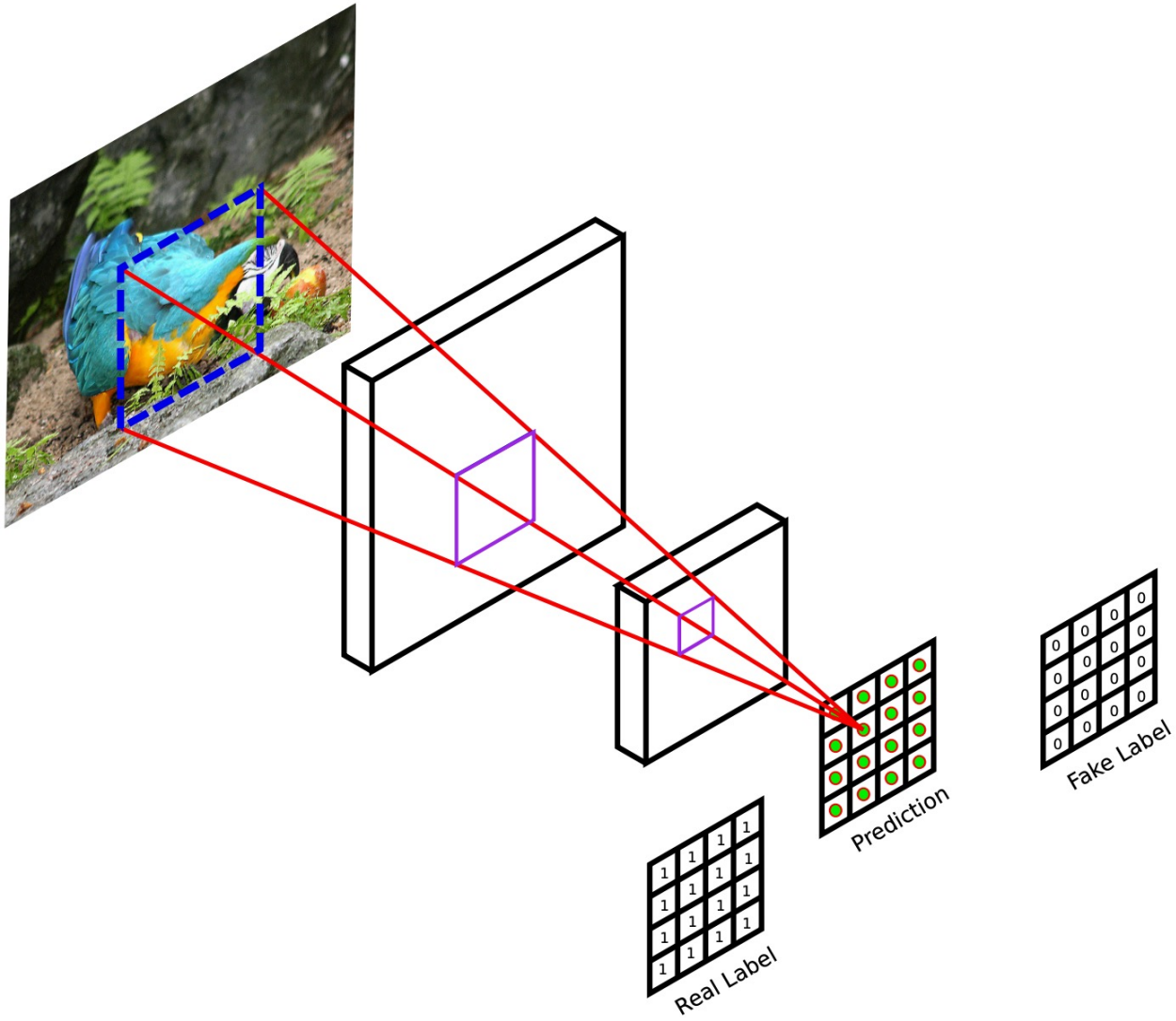
애매한 값을 취한다.

Input

Output

Ground Truth

Pix2Pix



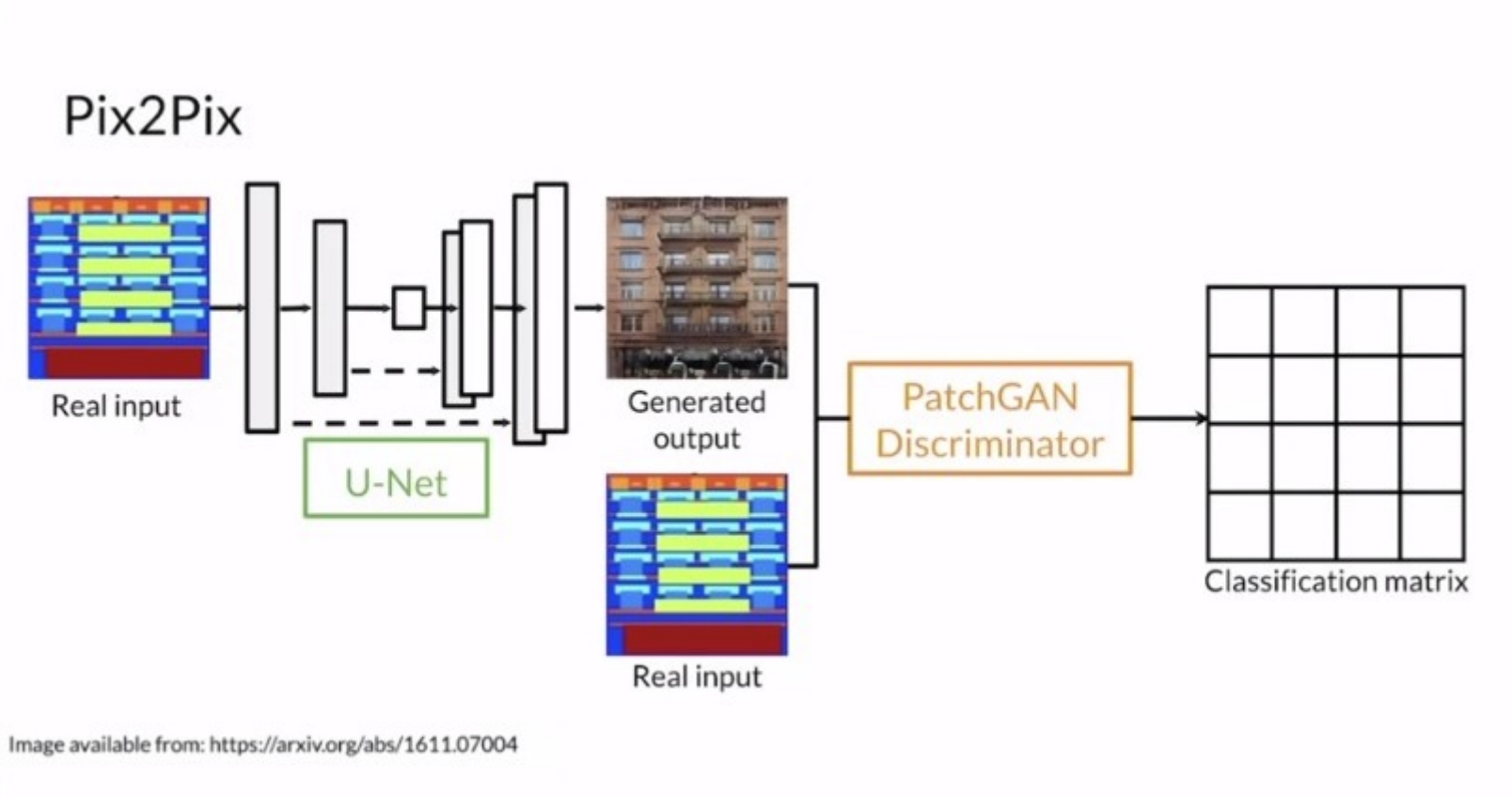
L1 Loss가 이미 low-frequency를 잘 잡는다.

따라서 discriminator는 high-frequency를 잘 잡도록 만들면 된다.

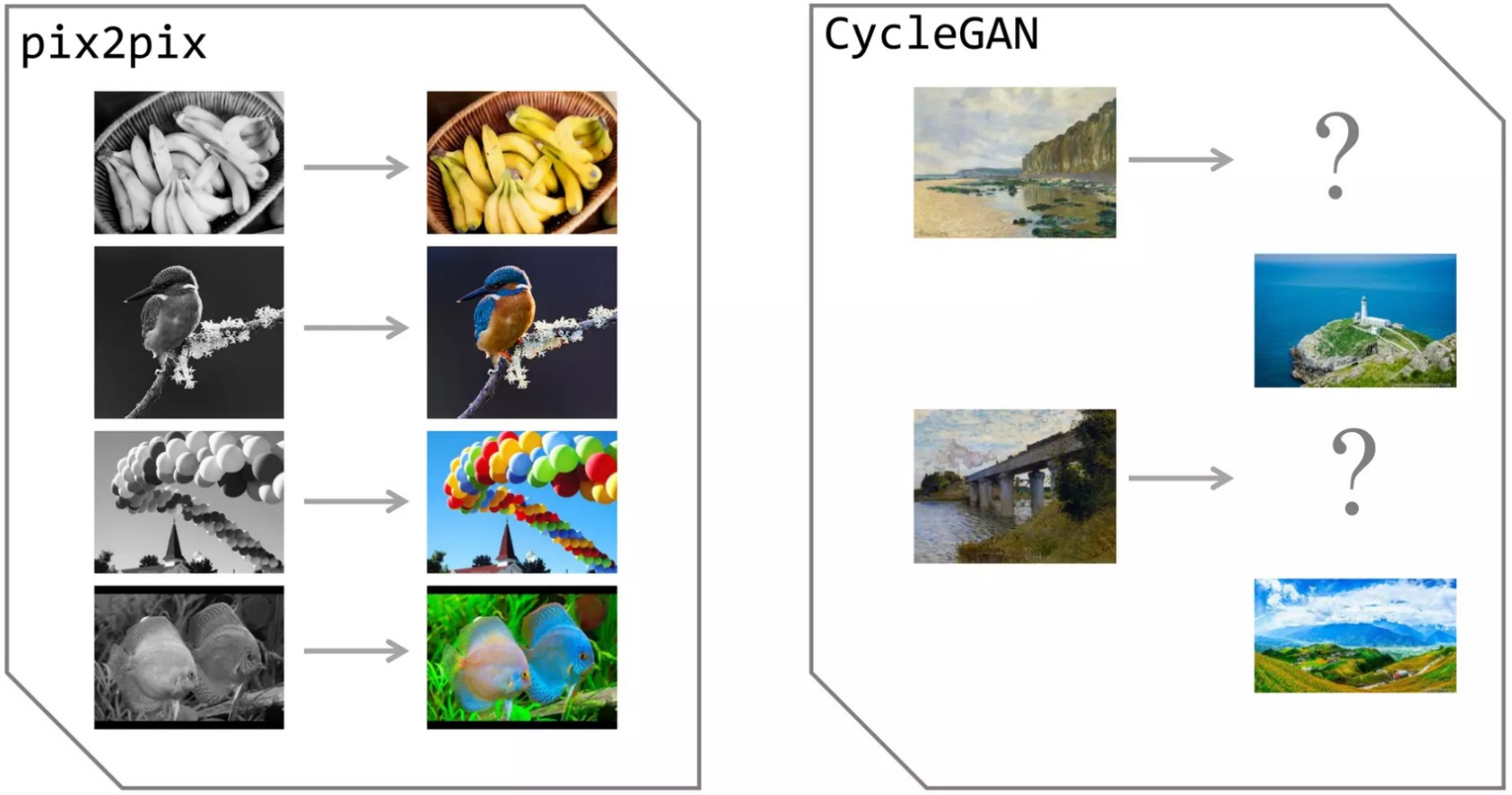
따라서 70x70의 receptive field만 가지는 Patch 단위에서 판별을 하자

- 파라미터 수가 줄어든다
- 전체 이미지 크기에 영향을 받지 않는다

Pix2Pix

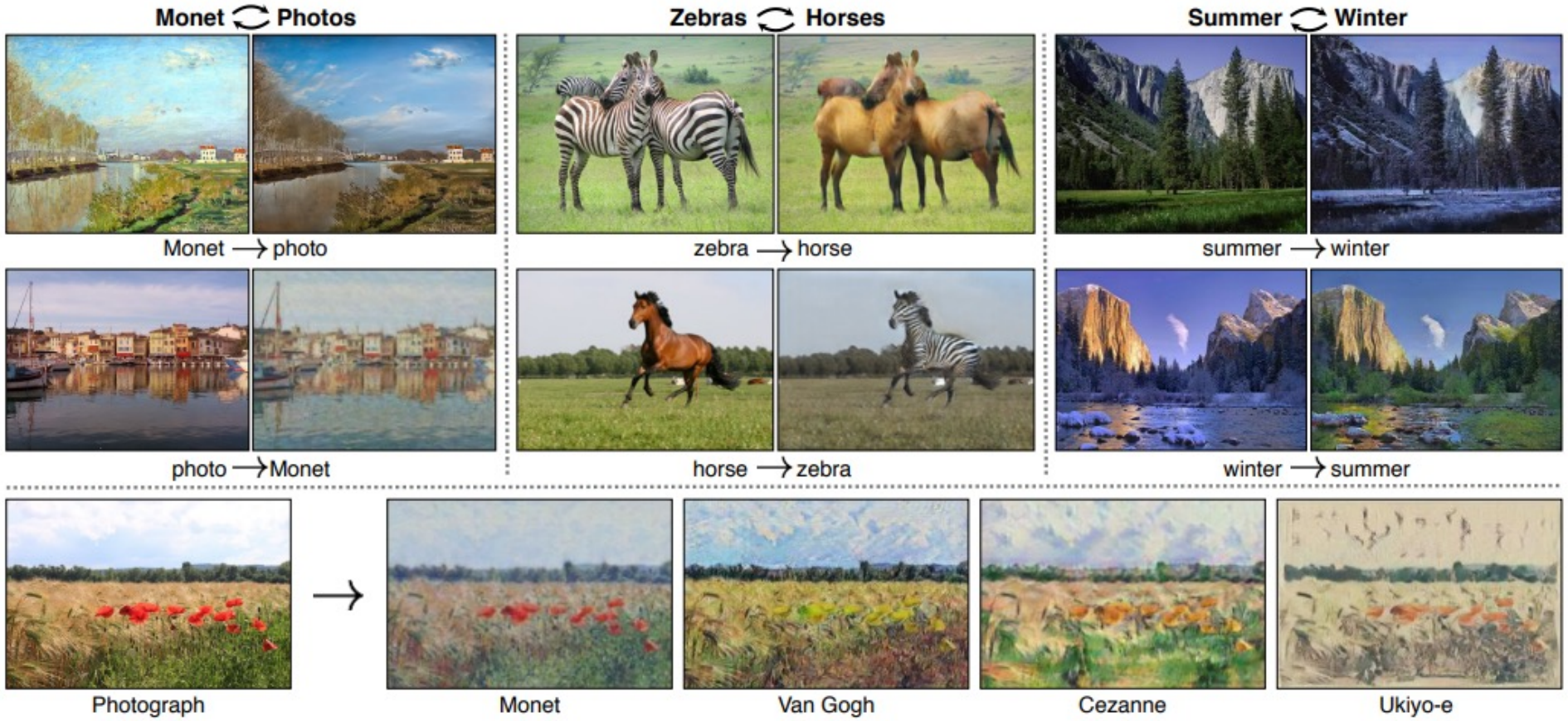

Pix2Pix
Real input
U-Net
Generated output
Real input
PatchGAN Discriminator
Classification matrix
Image available from: https://arxiv.org/abs/1611.07004

Pix2Pix to CycleGAN



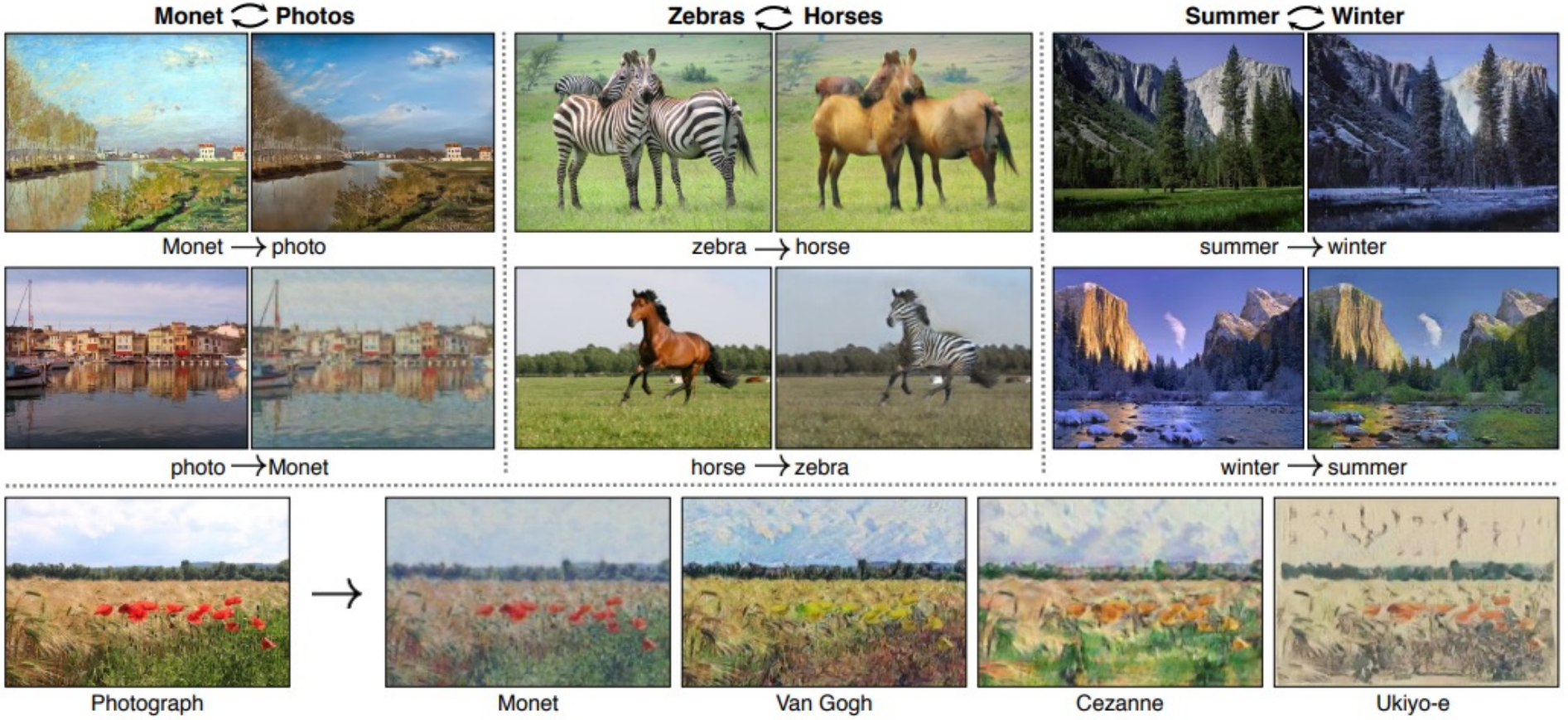
Pix2Pix의 문제점 : Ground Truth Image가 존재해야한다!

CycleGAN



사람은 모네의 그림을 보고 실제 풍경을 상상할 수 있다.
말을 보고 얼룩말을 상상할 수 있다.

CycleGAN



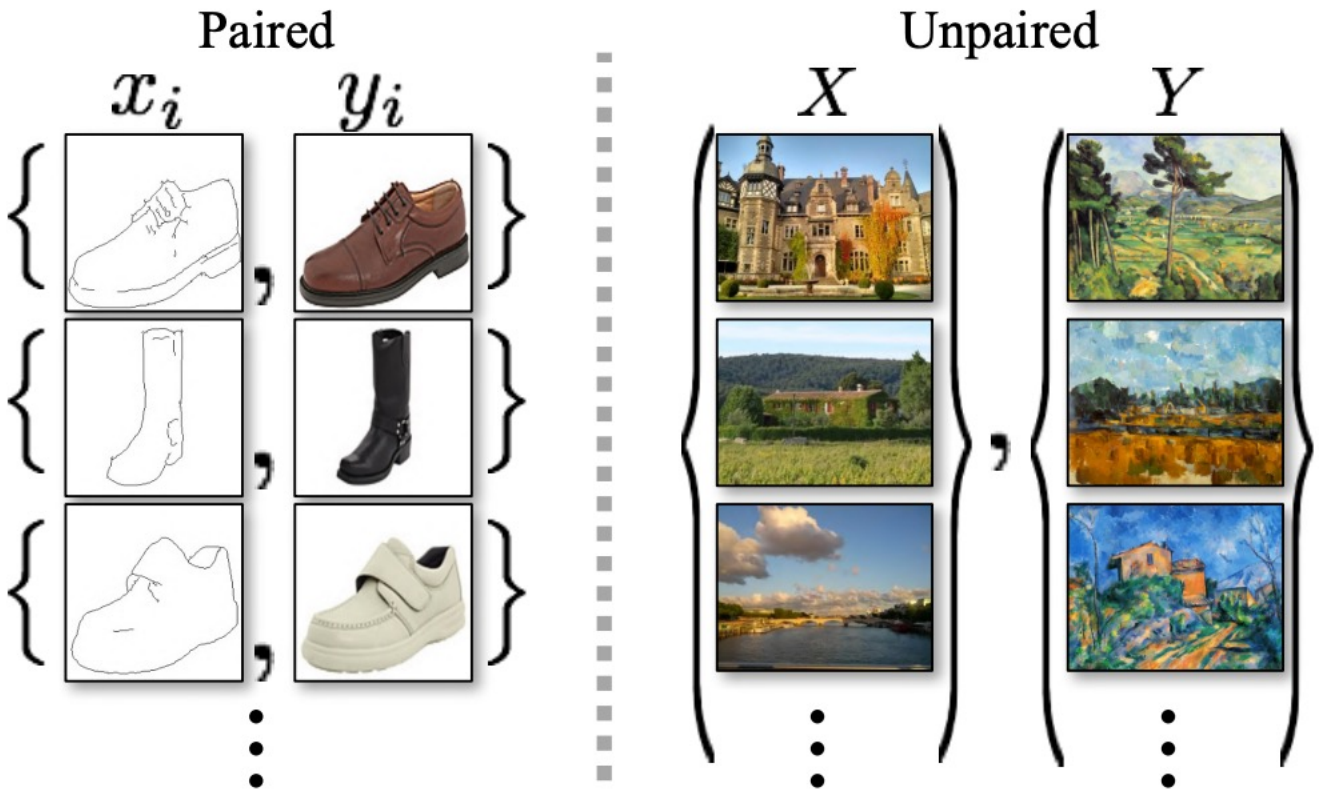
인간은 Paired data를 가지고 있지 않다!
해당 Domain의 이미지 집합을 알고 있고 그것을 바탕으로 상상 할 수 있다.

CycleGAN

- 두 도메인간의 underlying relationship이 있다고 가정

- Relationship을 배우는 방법을 찾는다.

CycleGAN

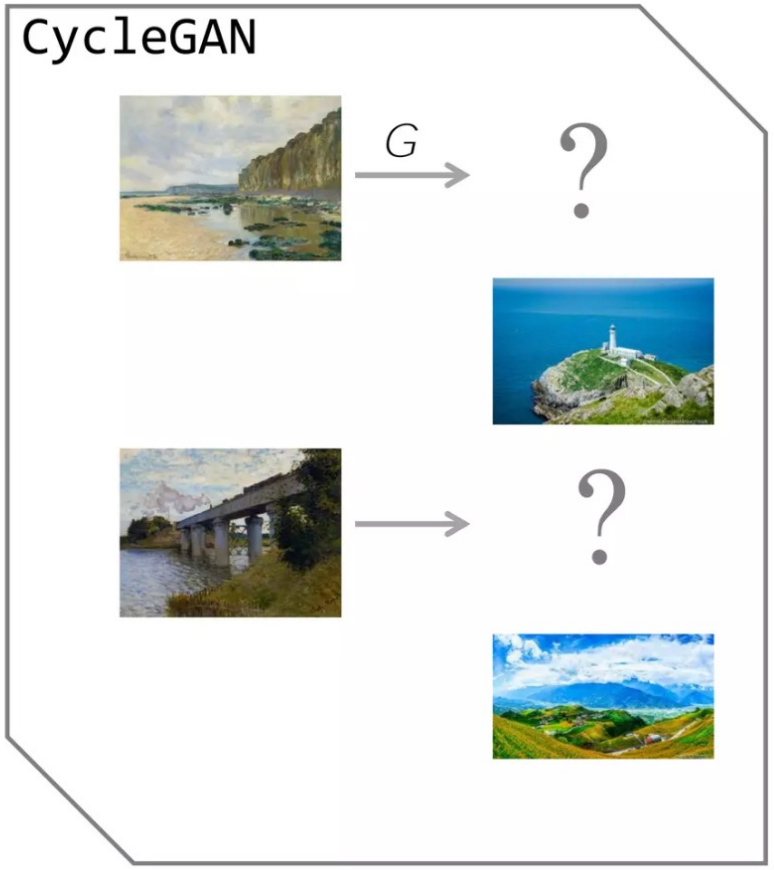


CycleGAN – Task의 문제점

Loss: $L_{GAN}(G(x), y)$

G(x) should just look photorealistic

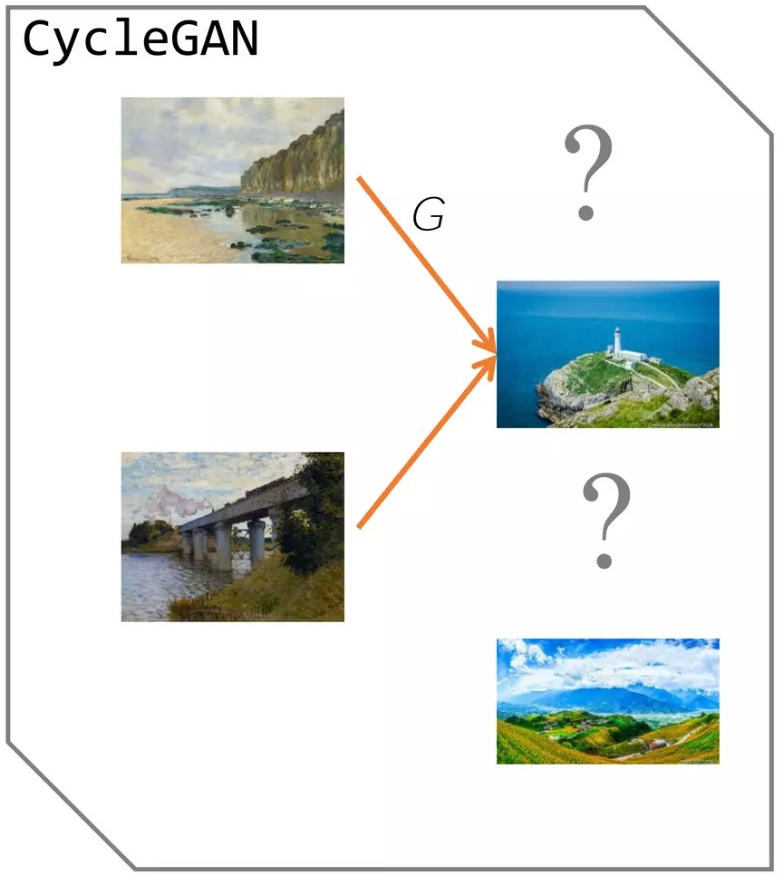
$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_y[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z))].$$



CycleGAN – Task의 문제점

Loss: $L_{GAN}(G(x), y)$

G(x) should just look photorealistic



- Input을 무시하고 하나의 Image만 생성 할 수 있다.

Image-to-Image Translation

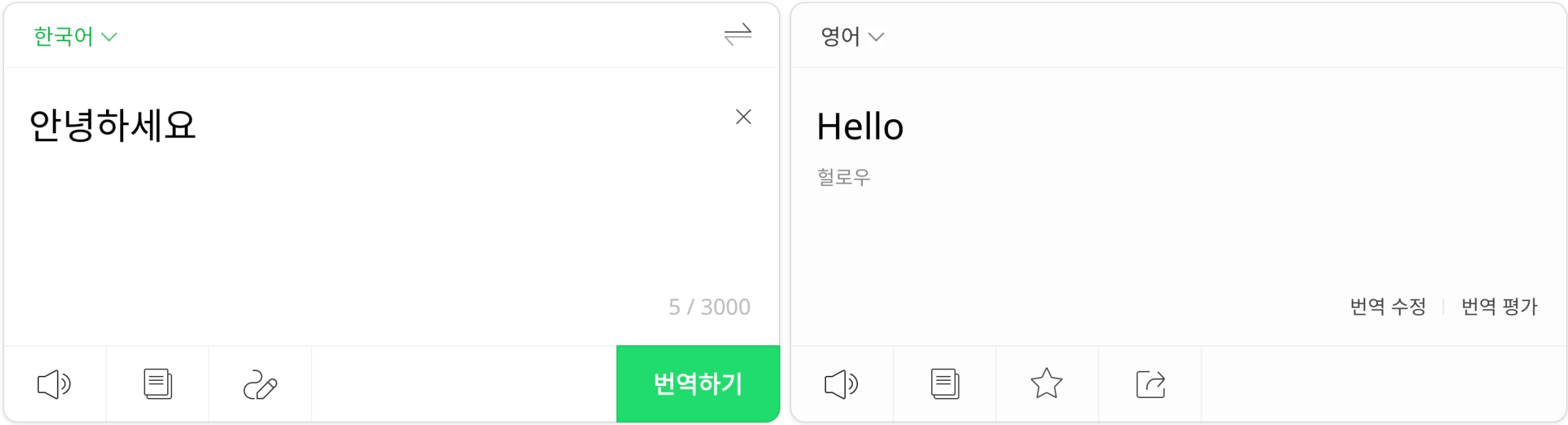
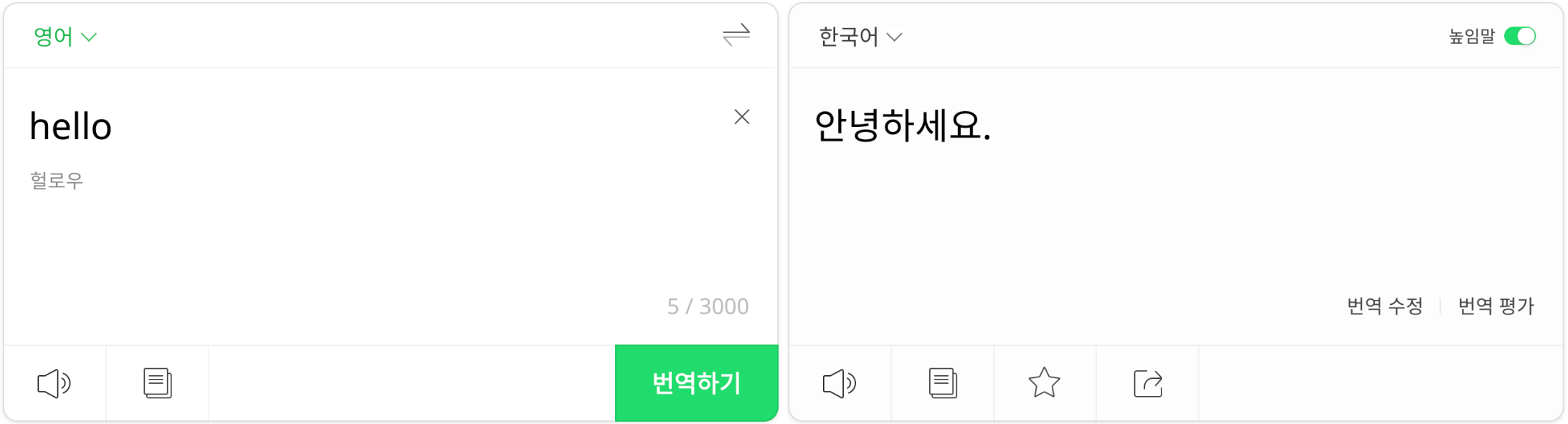


Image-to-Image Translation

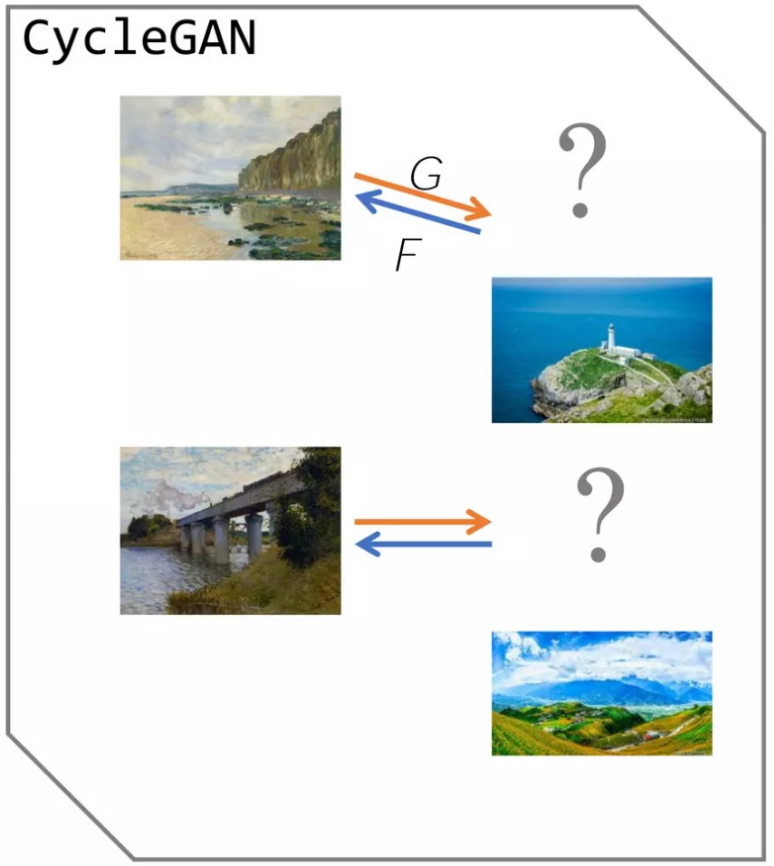


CycleGAN – add Cycle Consistency

Loss

$$L_{GAN}(G(x), y) + \left\| F(G(x)) - x \right\|_1$$

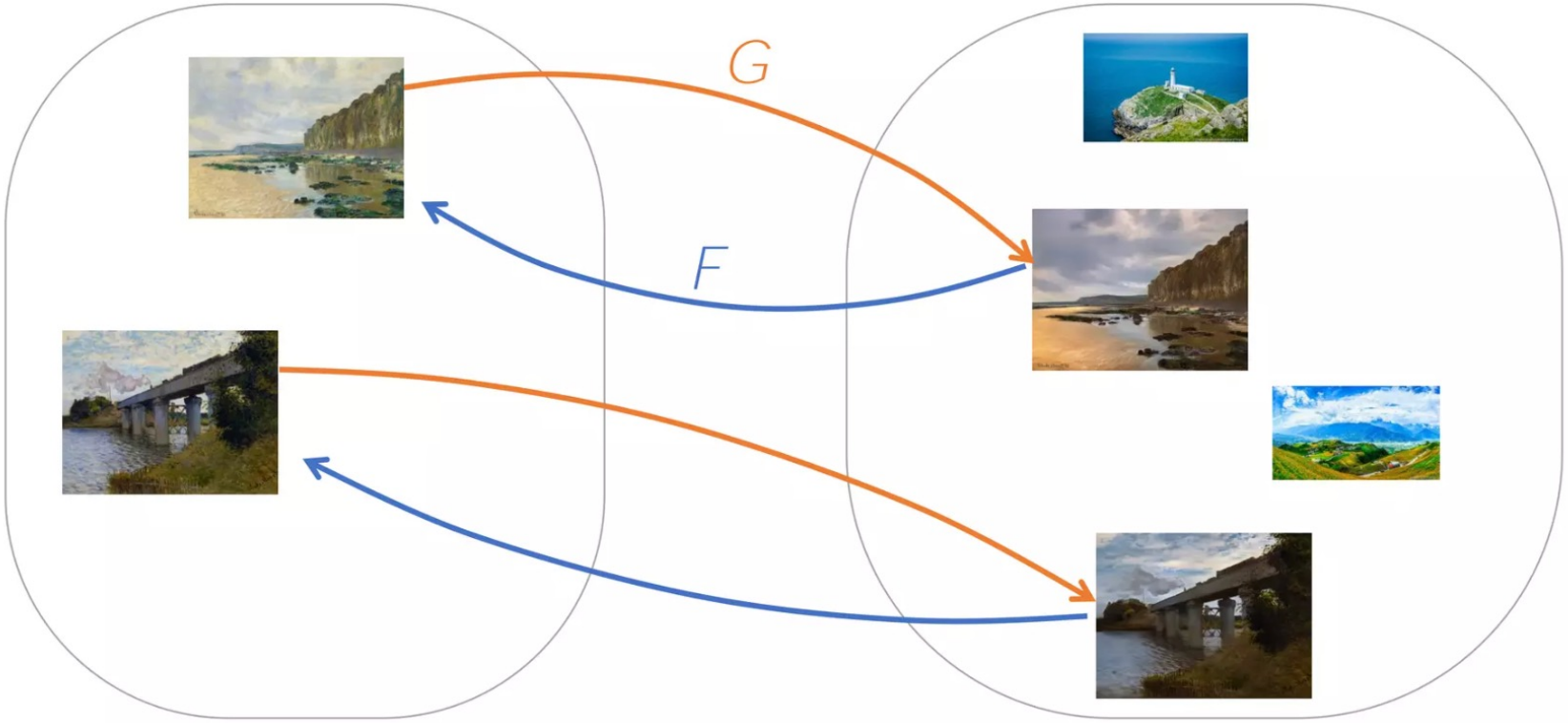
G(x) should just look photorealistic
and *F(G(x))* should be *F(G(x)) = x*,
where *F* is the inverse deep network



또 다른 GAN Network인 F를 추가해서

다시 Input을 복구 할 정도로만 Image를 변경하자

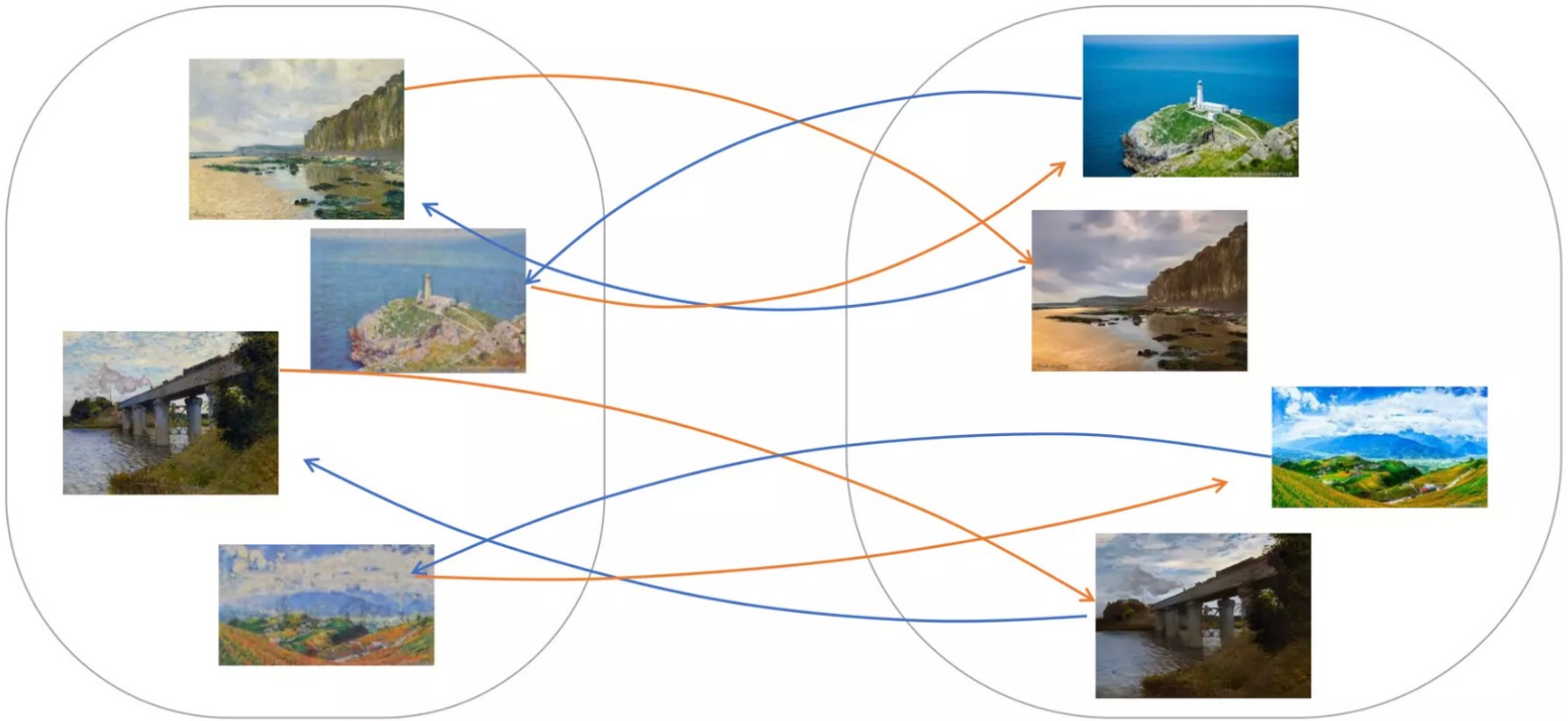
CycleGAN



$$L_{GAN}(G(x), y) + \left\| F(G(x)) - x \right\|_1$$

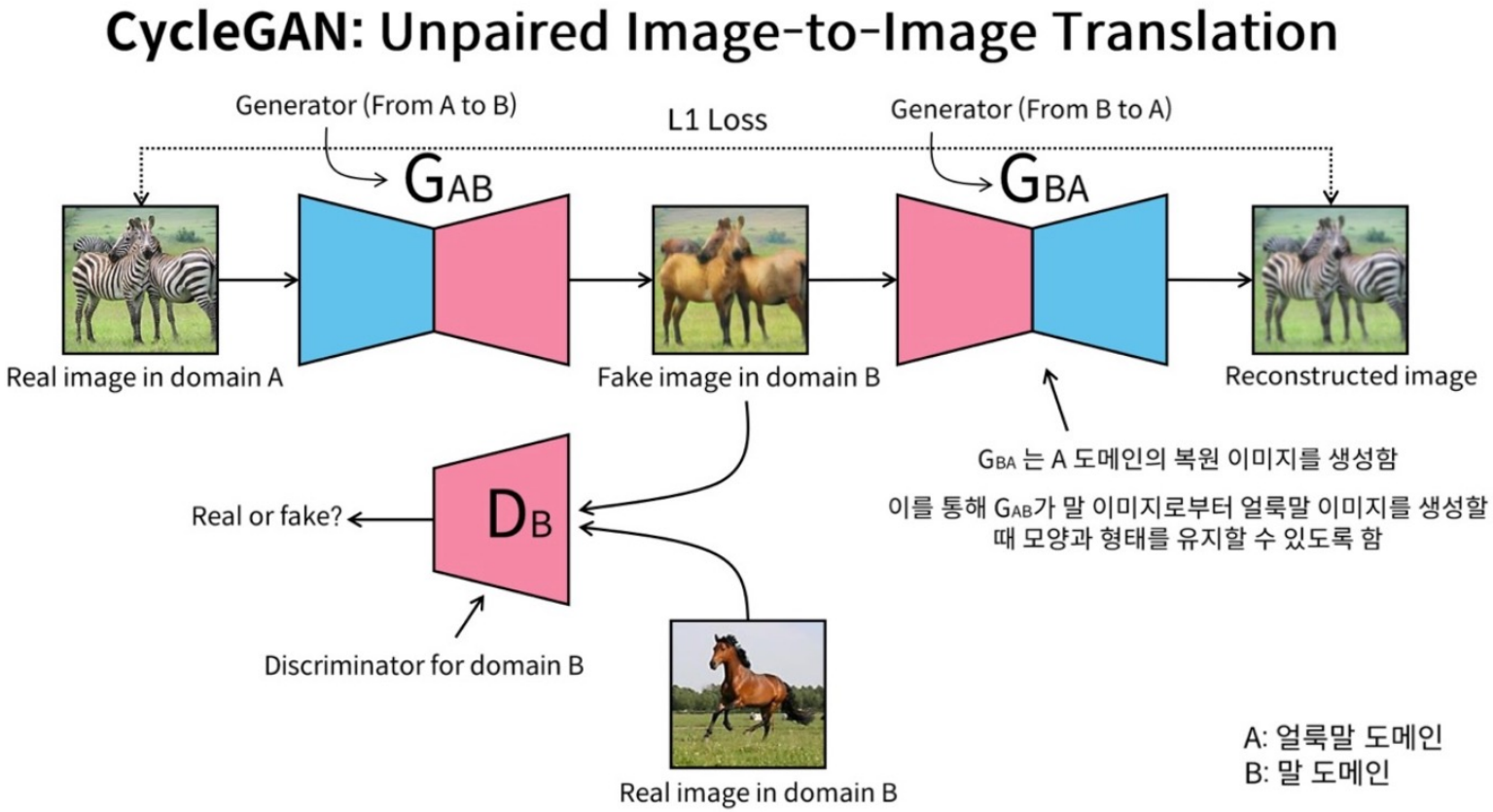
위의 Loss를 사용하면 F는 Translation된 이미지를 원래대로 돌리는 역할
But F 자체가 그럴듯한 Image의 수정을 하는 network가 된다는 보장은 없다.

CycleGAN



$$L_{GAN}(G(x), y) + \left\| F(G(x)) - x \right\|_1 \quad + \quad L_{GAN}(F(y), x) + \left\| G(F(y)) - y \right\|_1$$

CycleGAN



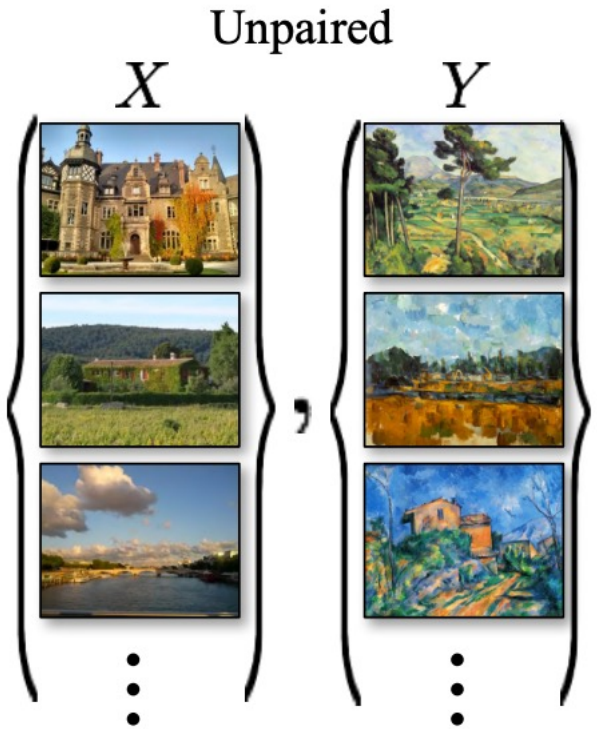
Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV'17

$$L_{GAN}(G(x), y) + \left\| F(G(x)) - x \right\|_1 \quad + \quad L_{GAN}(F(y), x) + \left\| G(F(y)) - y \right\|_1$$

CycleGAN More Detail

$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$$

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = &\mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]\end{aligned}$$

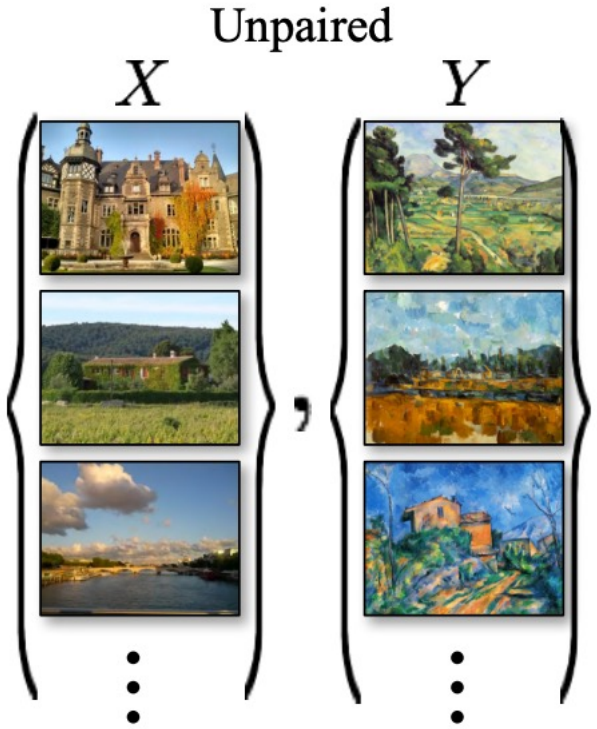


CycleGAN More Detail

$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$$

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$



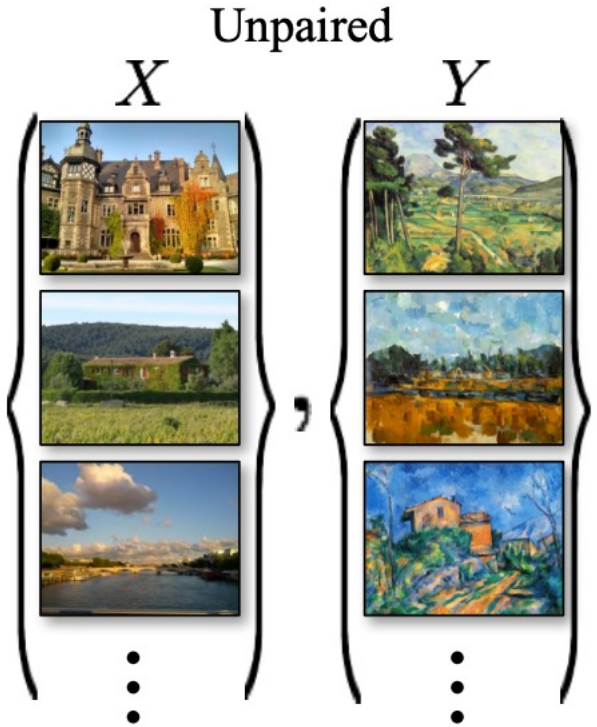
CycleGAN More Detail

$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$$

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = &\mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = &\mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

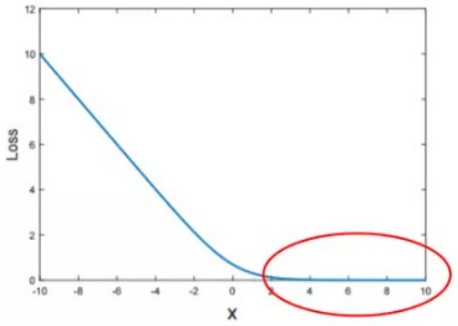
$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$



CycleGAN More Detail

- GANs with cross-entropy loss

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

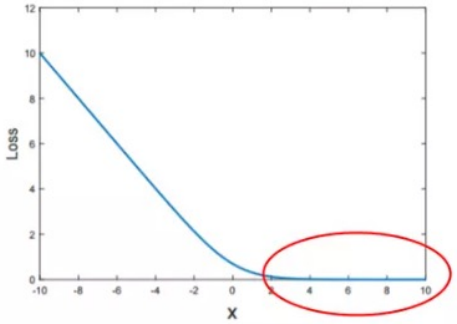


Vanishing gradients

CycleGAN More Detail

- GANs with cross-entropy loss

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$



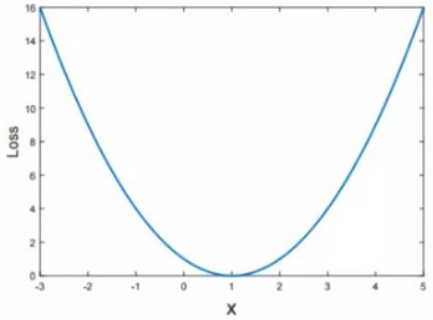
Vanishing gradients

- Least square GANs [Mao et al. 2016]

  Stable training + better results

$$\mathcal{L}_{\text{LSGAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[(D_Y(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D_Y(G(x))^2],$$

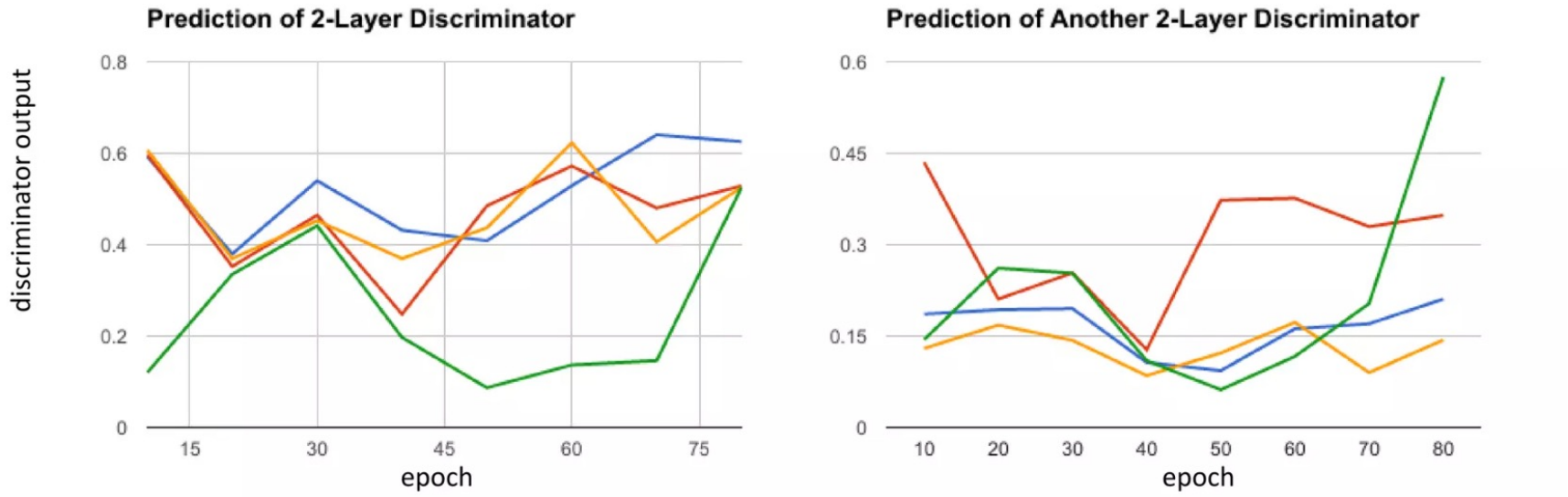
G Minimize : $\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$



CycleGAN More Detail

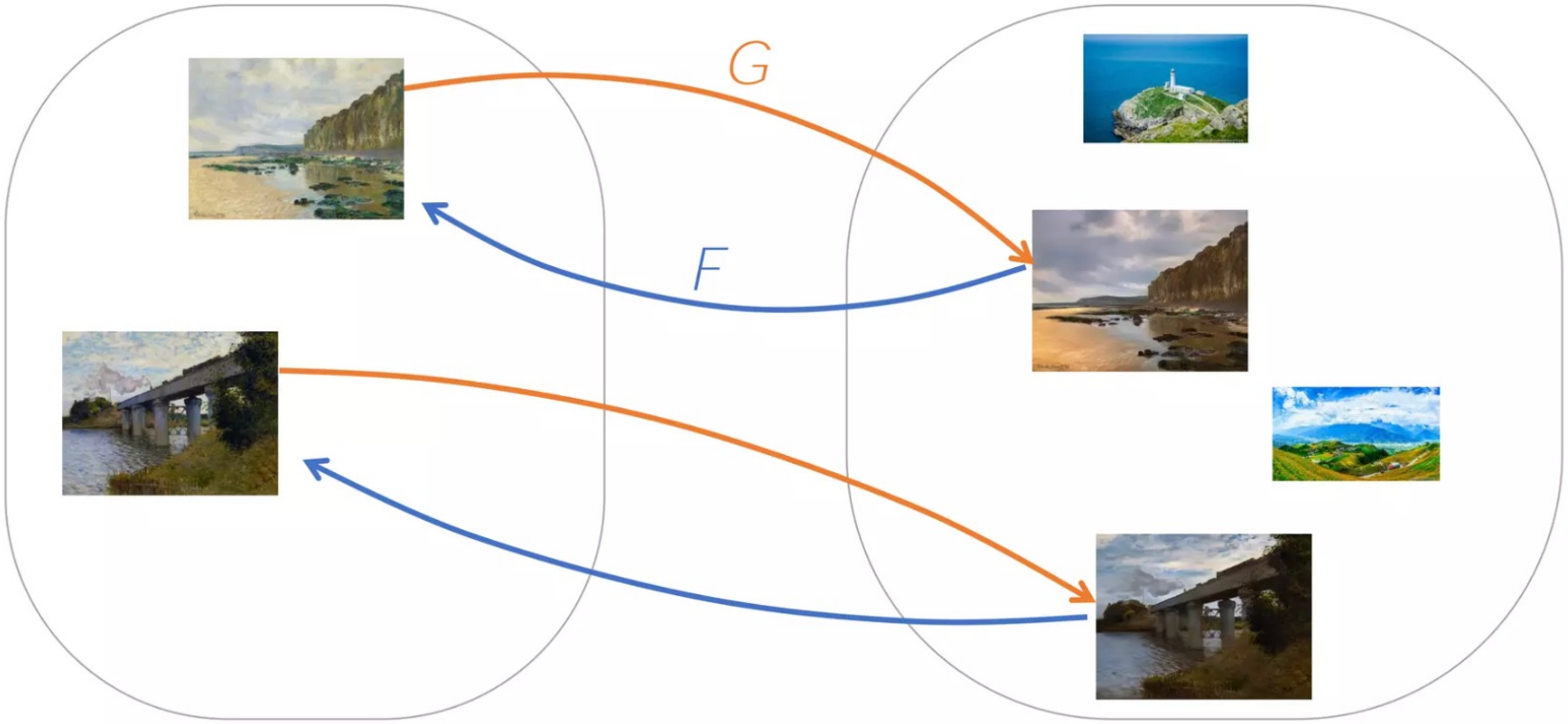
Training Details: replay buffer

- … because the discriminators can take very different trajectories in training



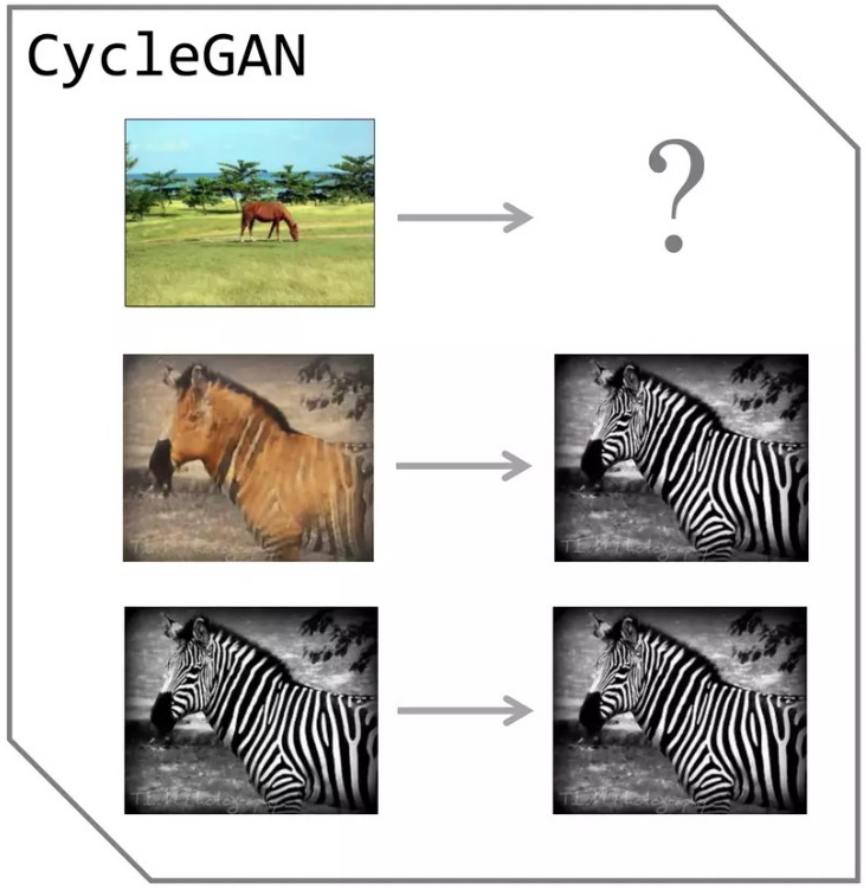
Generator가 생성한 최신 fake image만 사용하는 것이 아니라 과거의 경험을 재사용 하자!

CycleGAN More Detail

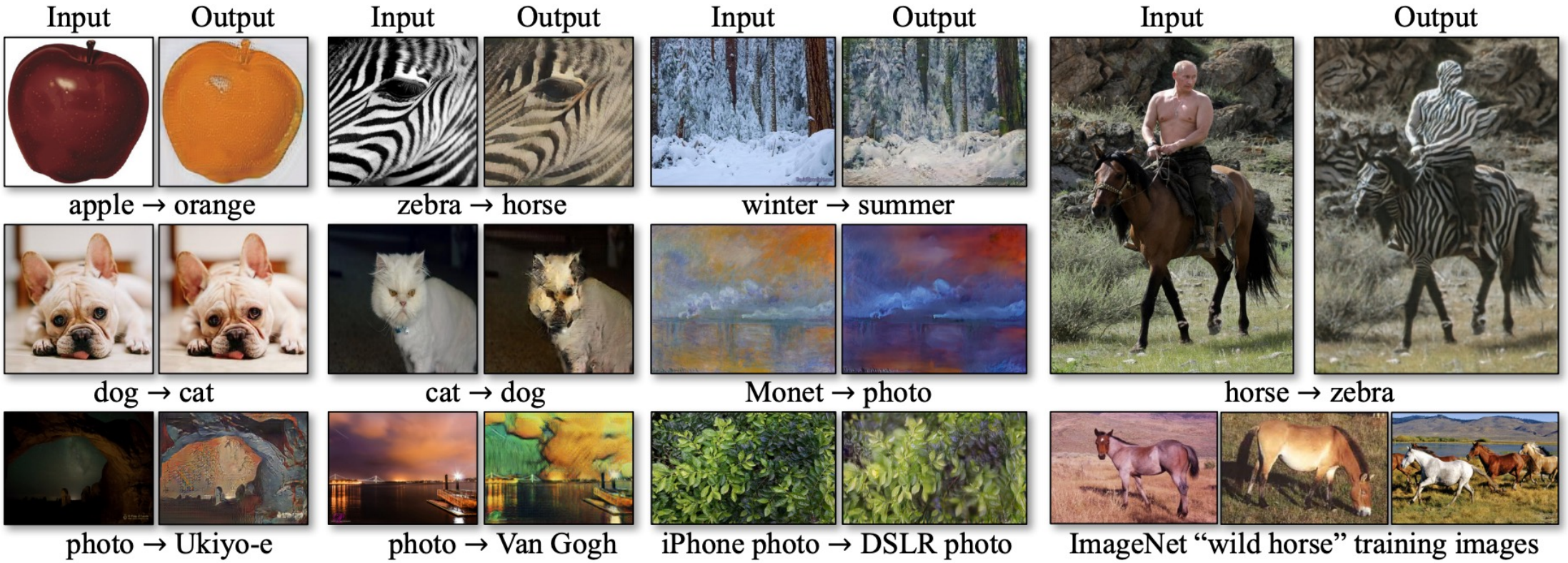


$$L_{GAN}(G(x), y) + \left\| F(G(x)) - x \right\|_1$$

CycleGAN More Detail



CycleGAN Failure



- Shape을 최소한으로 바꾼다.
- Semantic 정보를 완벽하게 잡지 못한다.

출처

- https://d2l.ai/chapter_computer-vision/transposed-conv.html
- https://cedar.buffalo.edu/~srihari/CSE676/22.3-GAN%20Mode%20Collapse.pdf
- https://brstar96.github.io/devlog/mldlstudy/2019-05-13-what-is-patchgan-D/
- https://www.slideshare.net/NaverEngineering/finding-connections-among-images-using-cyclegan
- https://velog.io/@wilko97/%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%B0-Unpaired-Image-to-Image-Translation-using-Cycle-Consistent-Adversarial-Networks-2017-CVPR