Catholic 2 Gogh

(b y c y c l e G A N)

201720014강민주
201820973김정태
201820995박은비

목차

01 연구동기

02 배경이론

03 CycleGAN

04 우리의 목표

05 구현코드 및 결과

06 결론

01

연구동기

01 연구동기

연구동기

02

배경이론

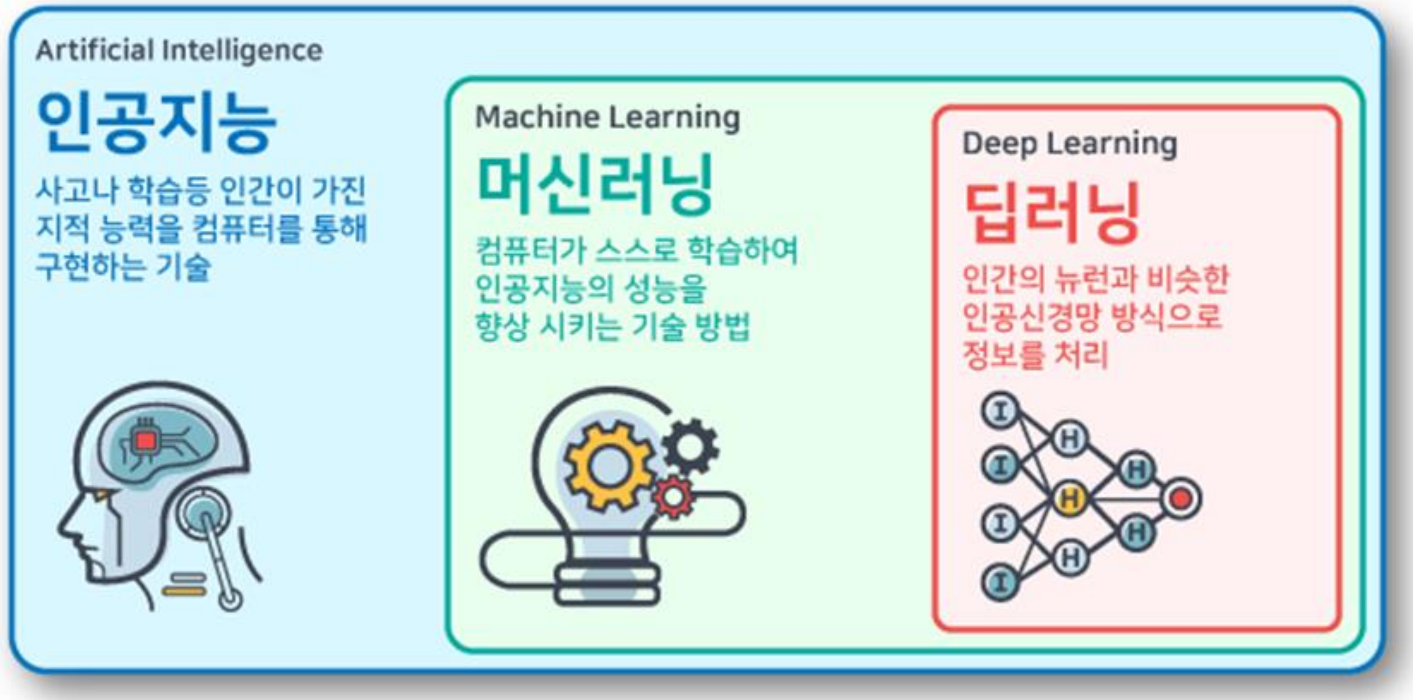
02 배경이론

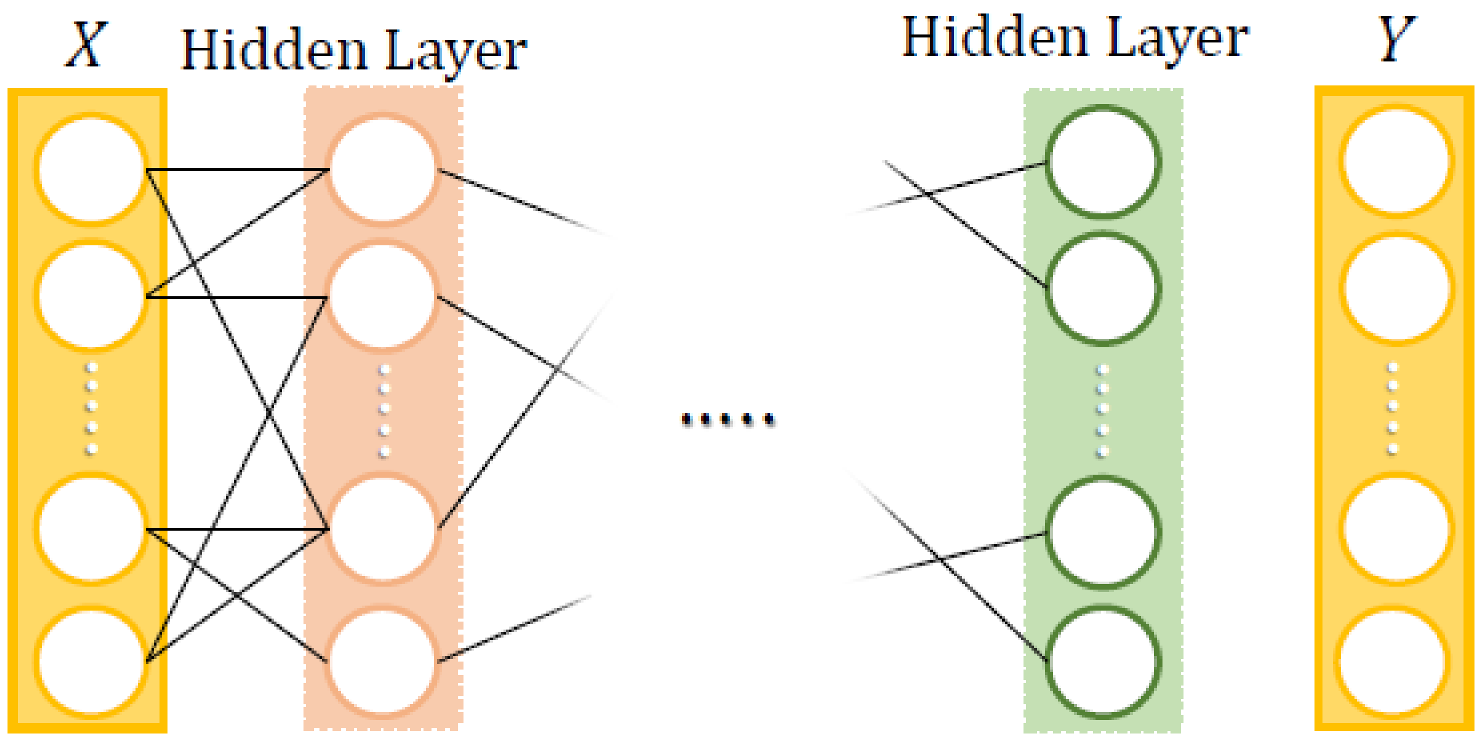
01. 딥러닝

02. 오토인코더

03. GAN

딥러닝





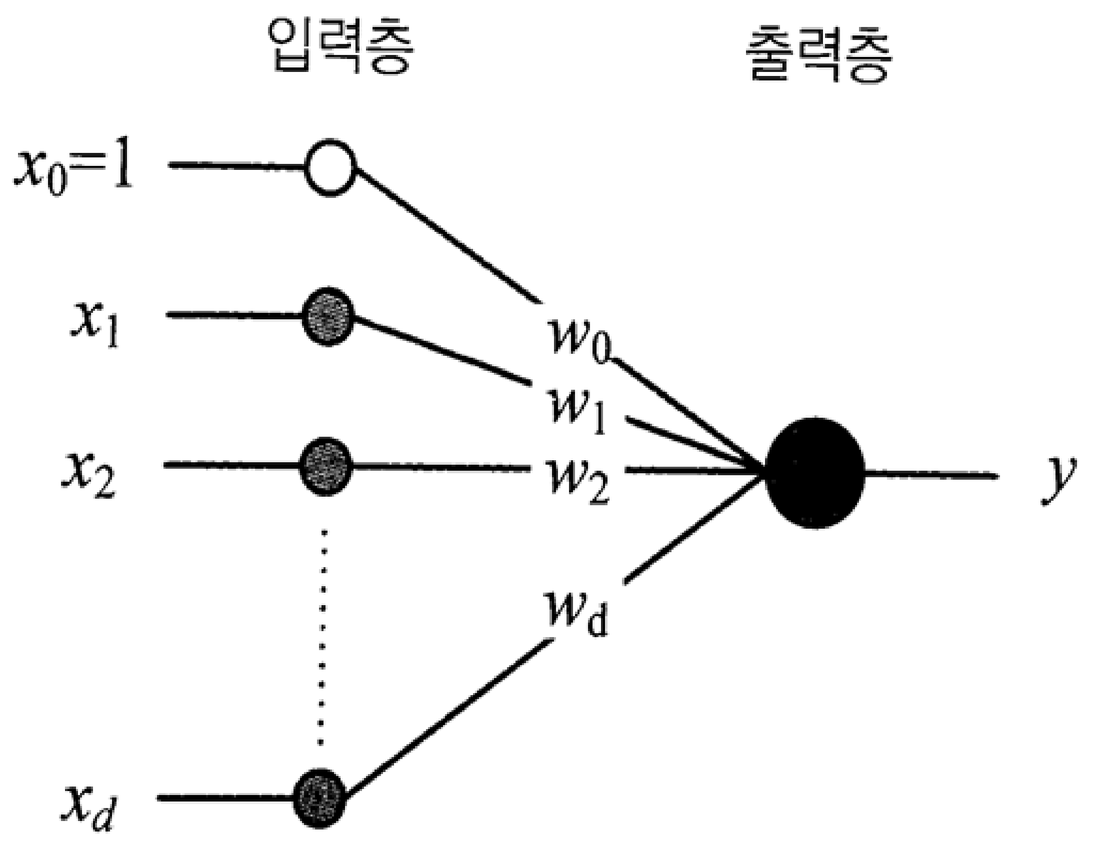
02 배경이론

01. 딥러닝

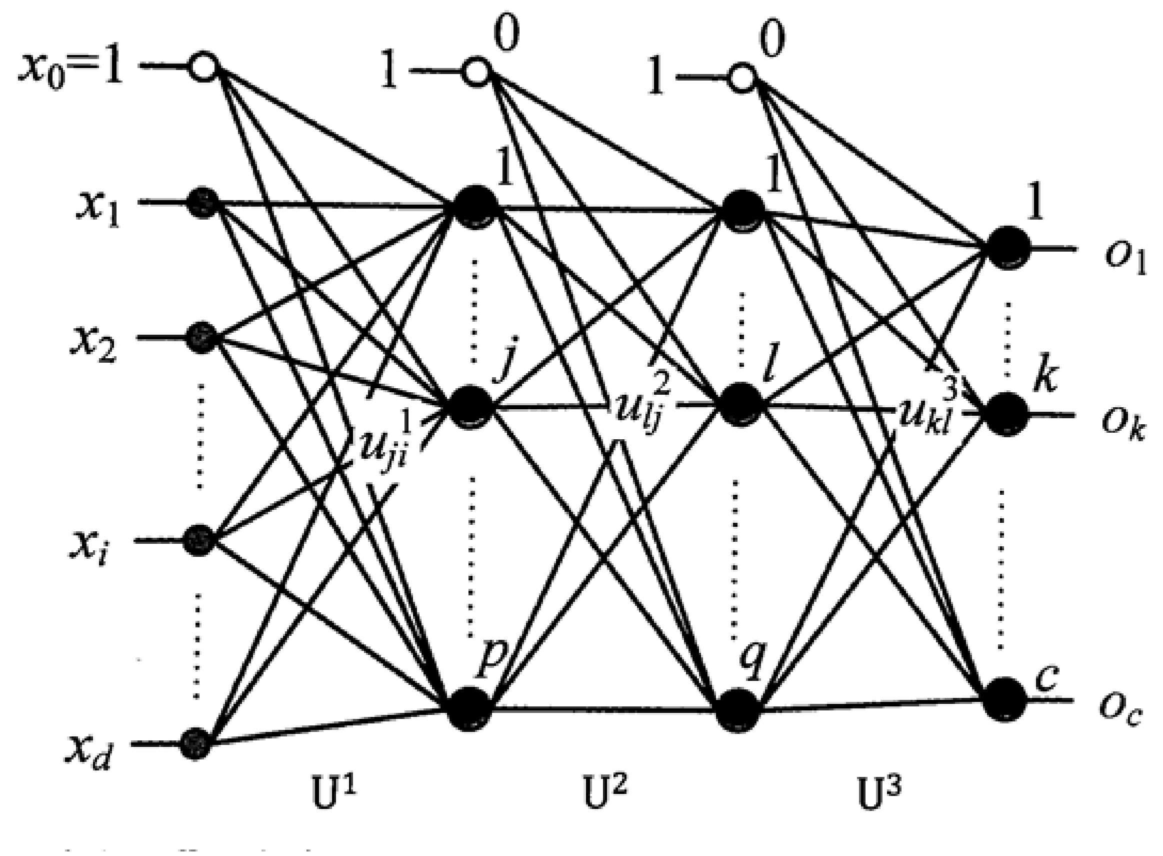
02. 오토인코더

03. GAN

신경망



단층 퍼셉트론의 예시

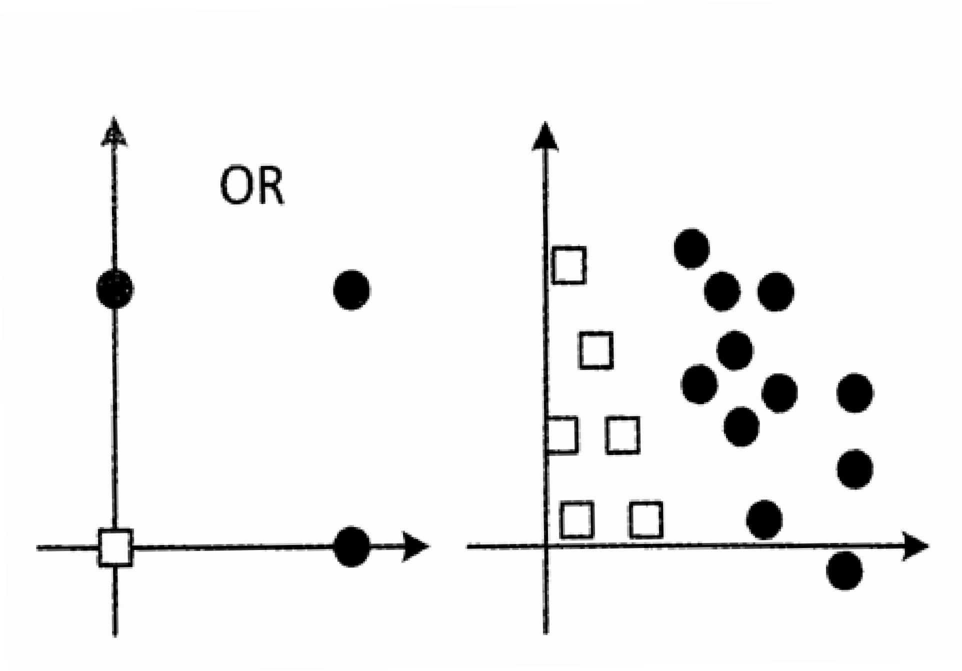


다층 퍼셉트론의 예시

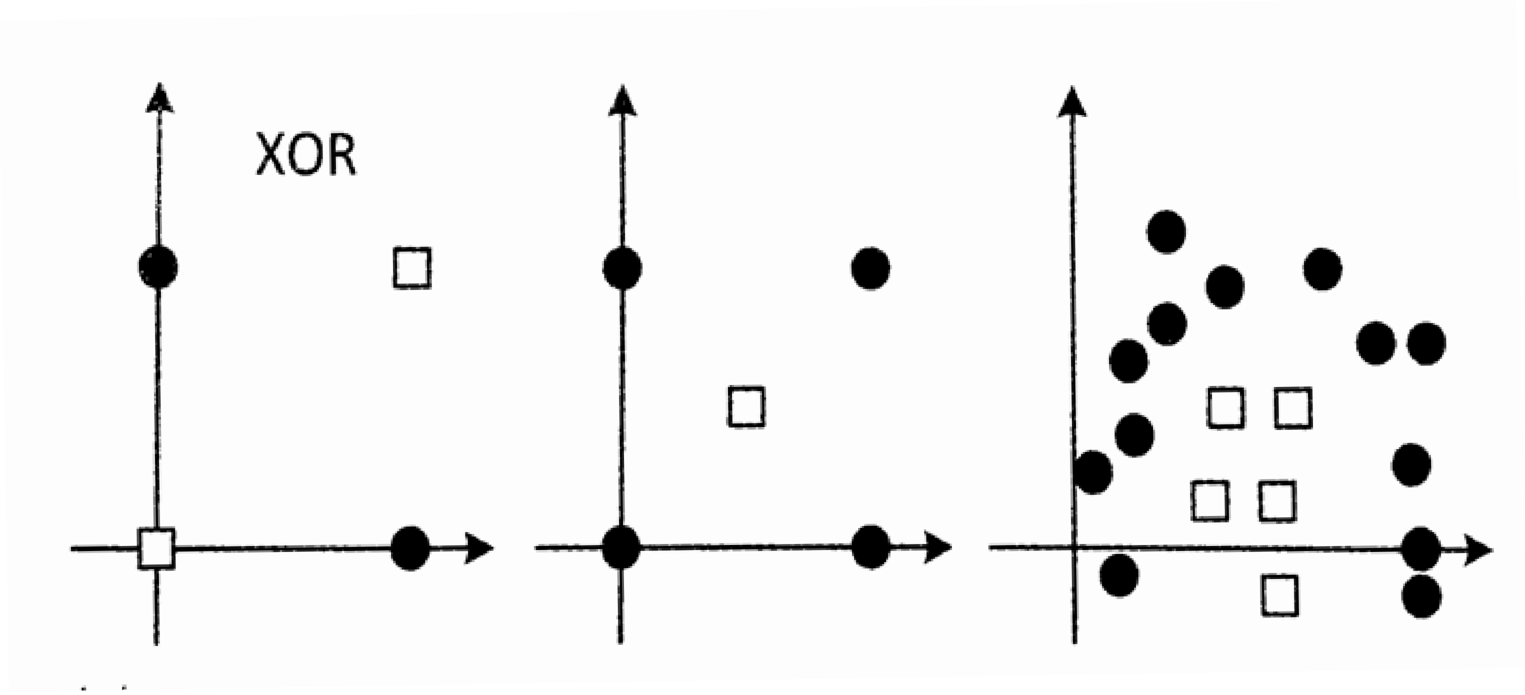
02 배경이론

01. 딥러닝
02. 오토인코더
03. GAN

신경망



선형분류 가능한 예시



선형분류가 불가능한 예시

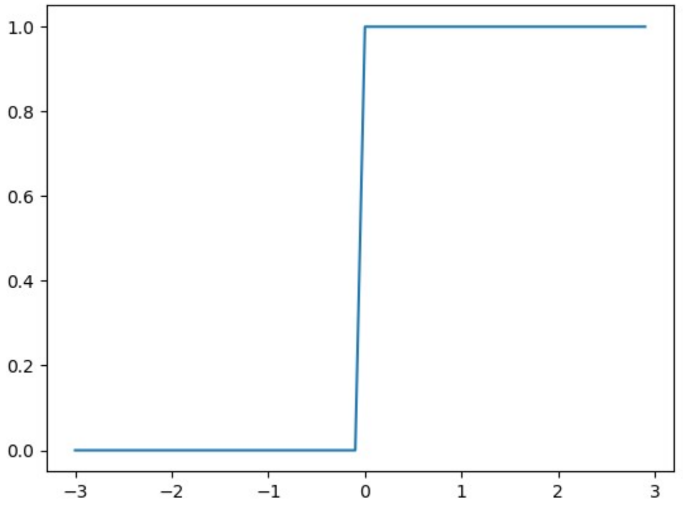
02 배경이론

활성화 함수

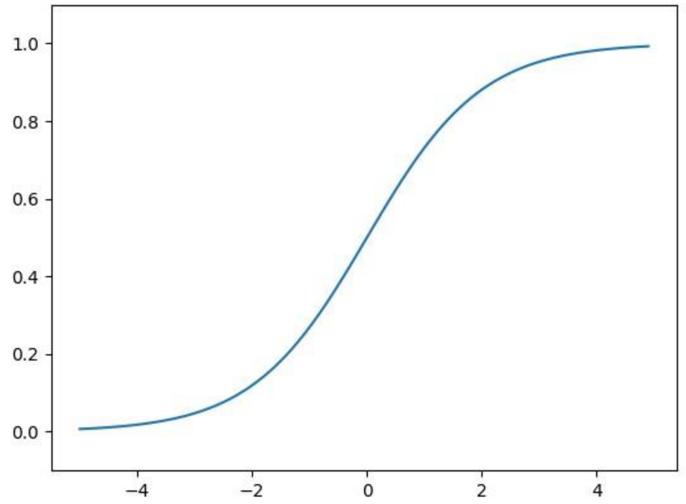
01. 딥러닝

02. 오토인코더

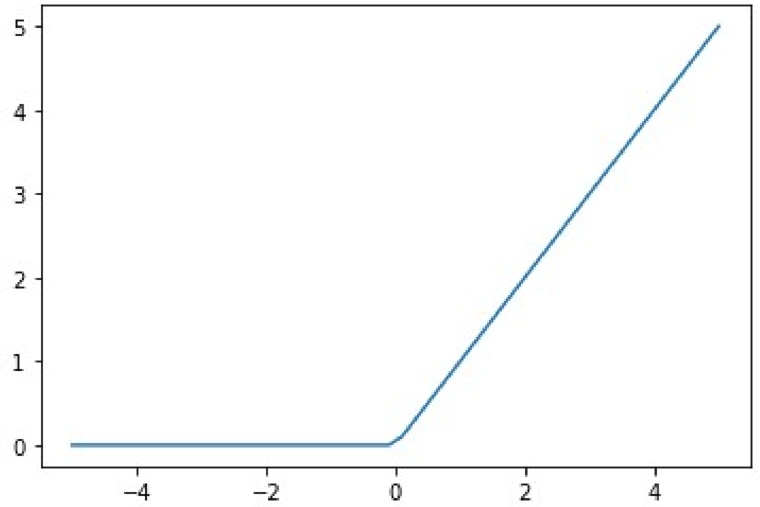
03. GAN



계단함수



시그모이드



ReLU

결과함수

01. 딥러닝
02. 오토인코더
03. GAN

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

소프트맥스함수

$$y = x$$

항등함수

01. 딥러닝
02. 오토인코더
03. GAN

손실 함수

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

평균제곱오차

$$E = -\sum_{i} y_i \log \tilde{y_i}$$

교차엔트로피 오차

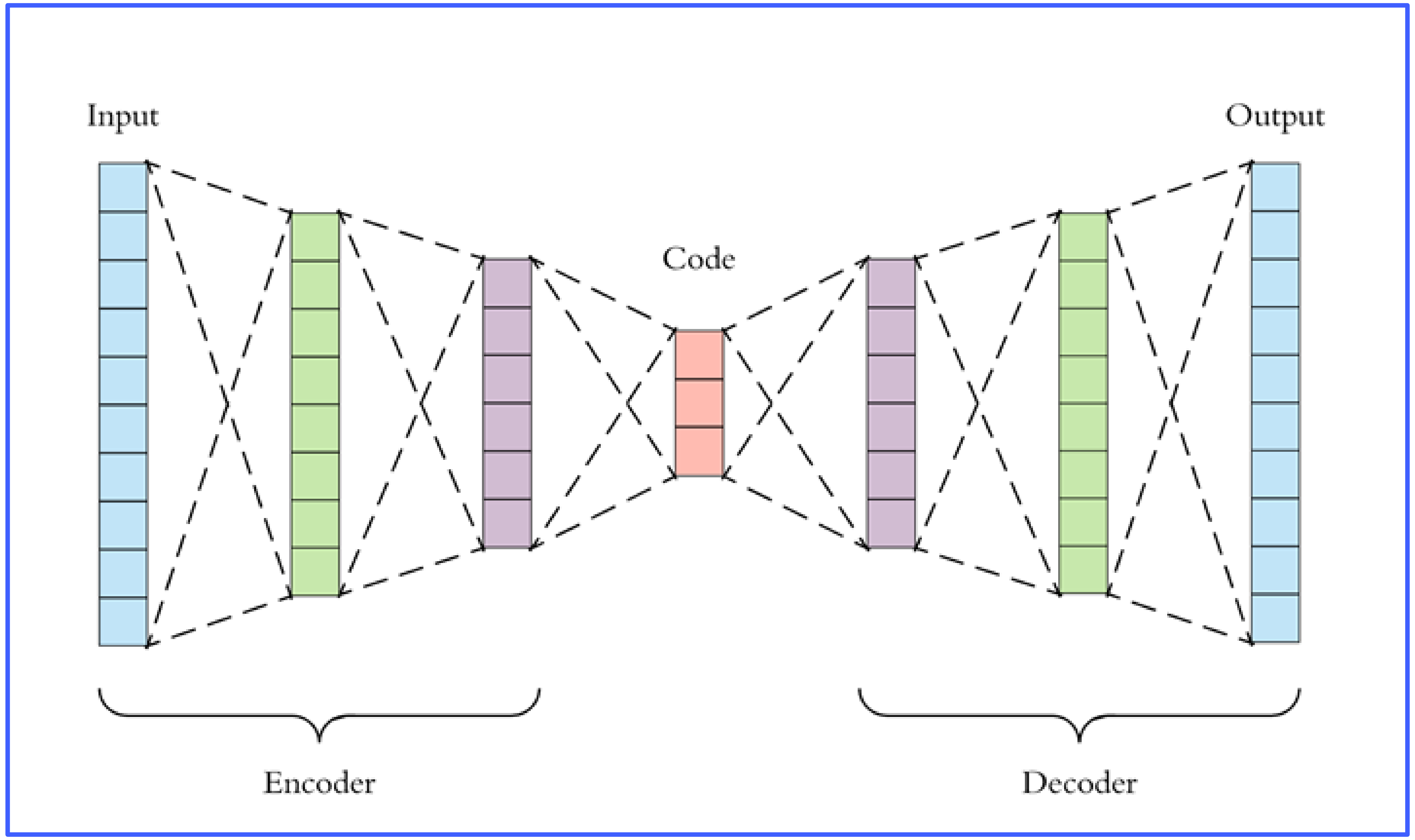
02 배경이론

01. 딥러닝

02. 오토인코더

03. GAN

오토인코더



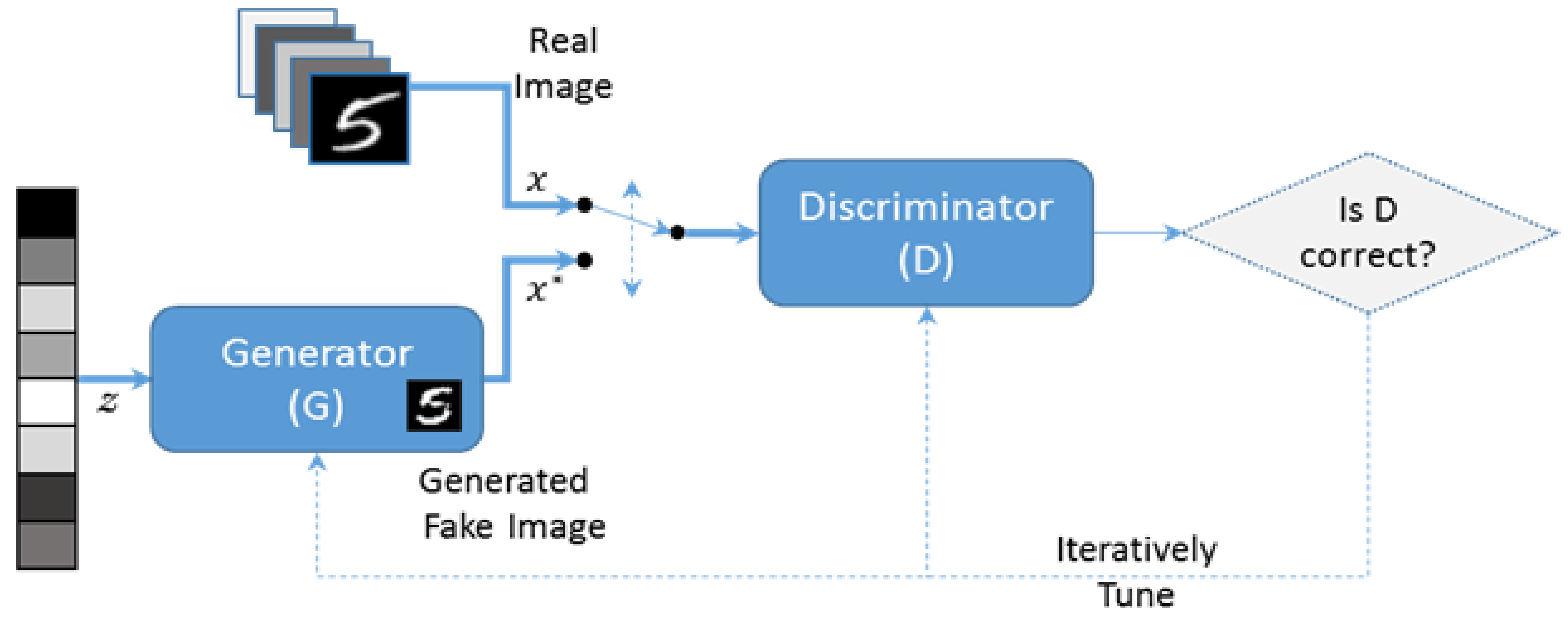
02 배경이론

01. 딥러닝

02. 오토인코더

03. GAN

GAN



02 배경이론

01. 딥러닝

02. 오토인코더

03. GAN

GAN의 손실함수

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

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

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

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

01. 딥러닝
02. 오토인코더
03. GAN

GAN의 손실함수

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

$$a(1-y) - by = -(a+b)y + a = 0$$

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

01. 딥러닝
02. 오토인코더
03. GAN

GAN의 손실함수

$$\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$$

$$p_g = p_{data}$$

$$D_G^*(x) = \frac{1}{2}$$

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

02 배경이론

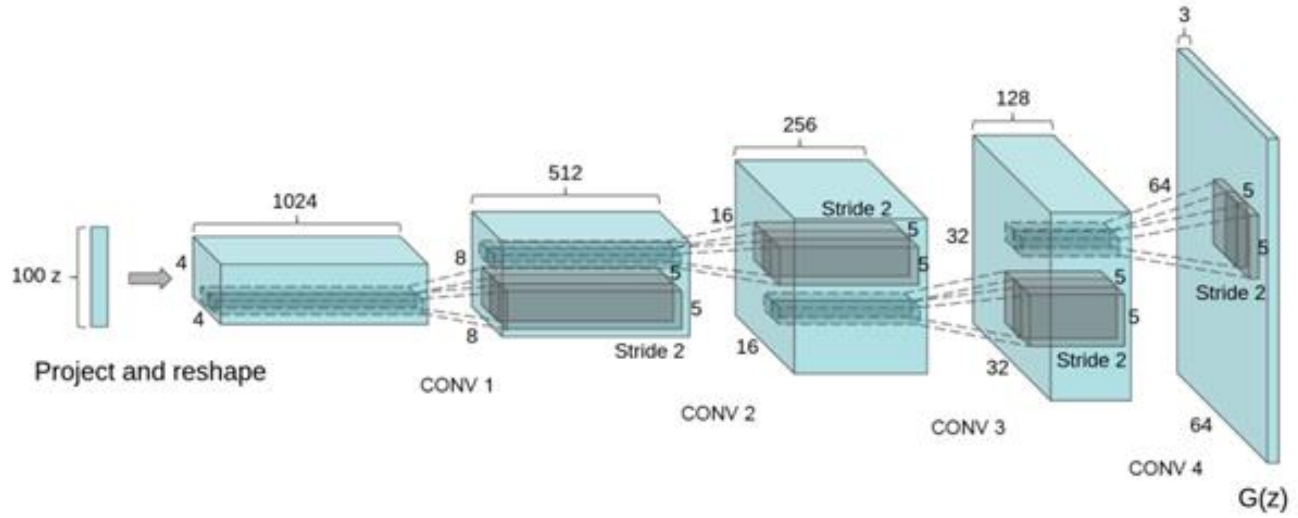
01. 딥러닝

02. 오토인코더

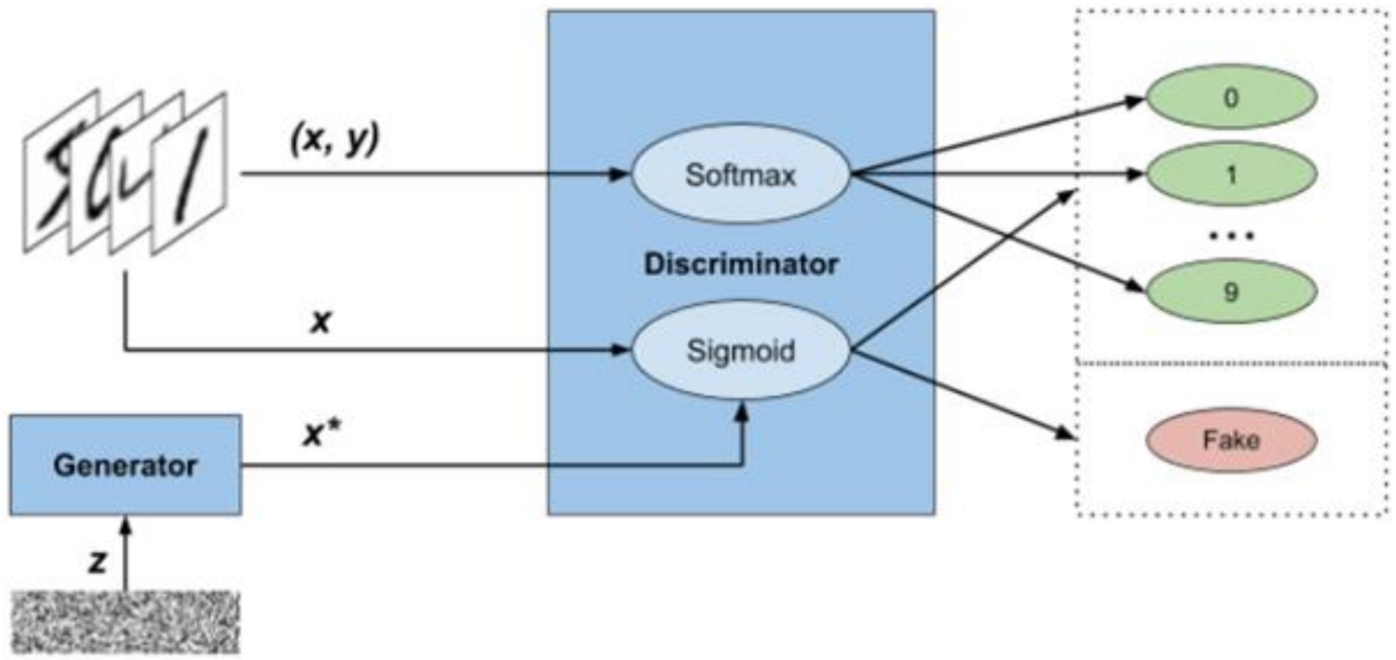
03. GAN

다른 종류의 GAN

DCGAN



SGAN



02 배경이론

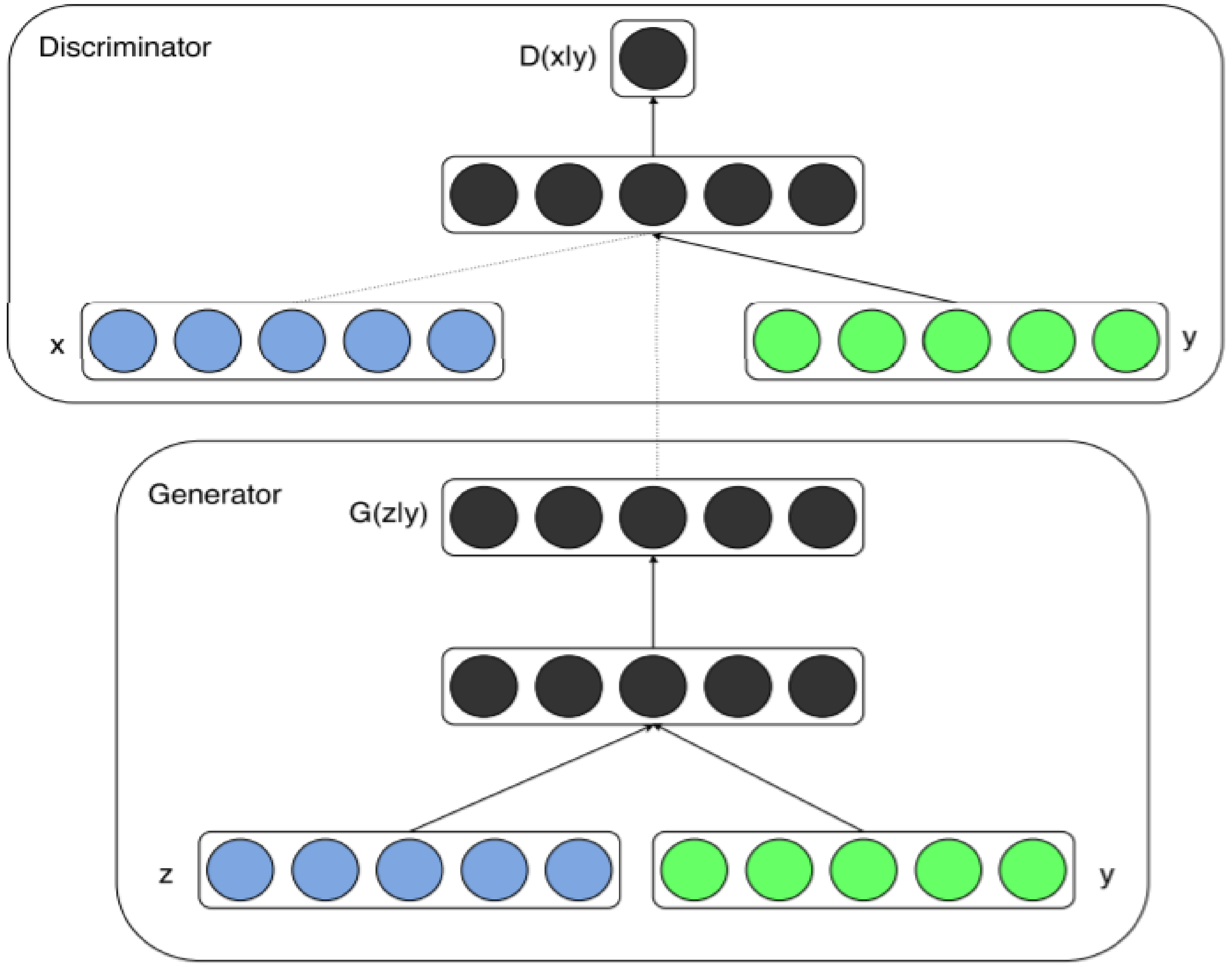
01. 딥러닝

02. 오토인코더

03. GAN

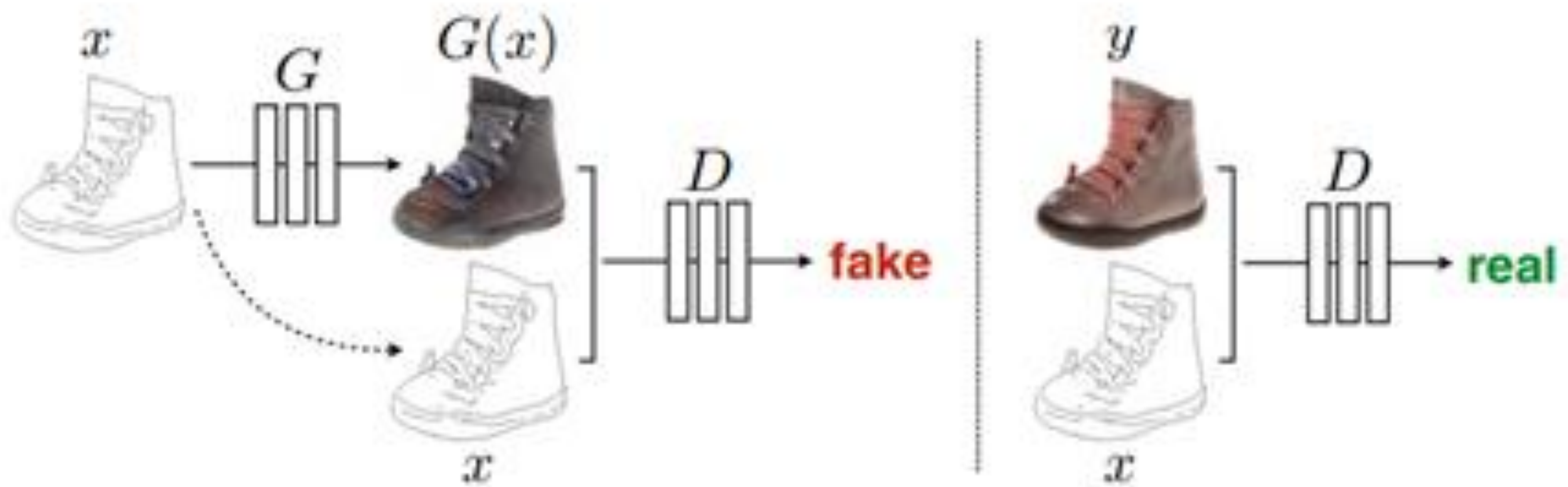
다른 종류의 GAN

CGAN



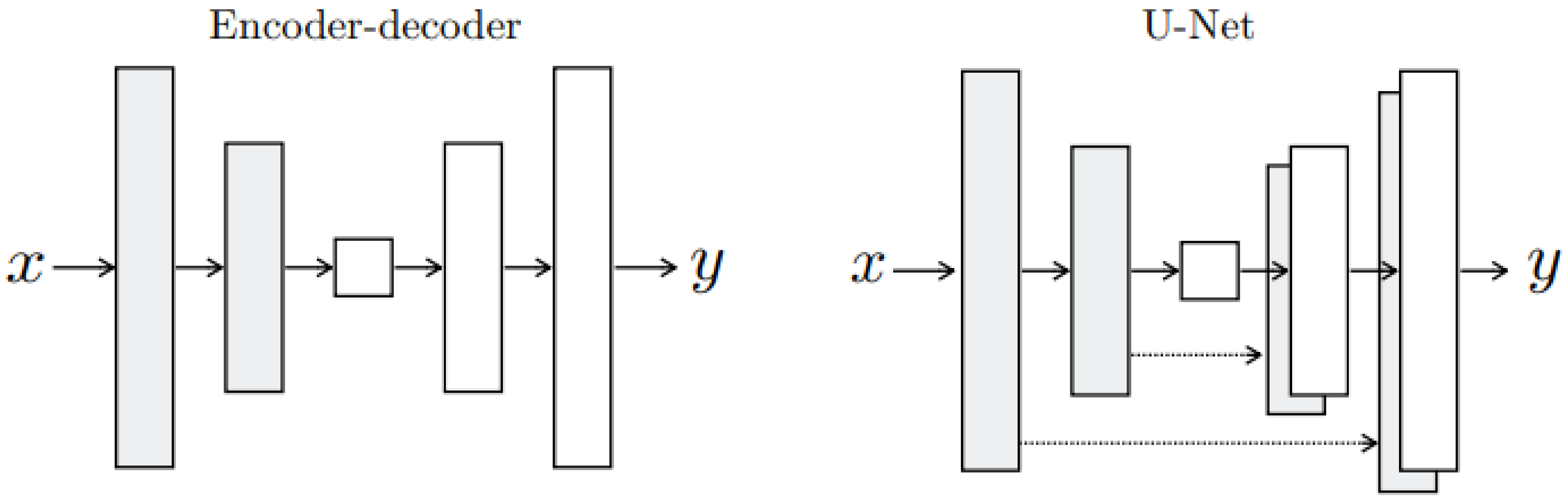
01. 딥러닝
02. 오토인코더
03. GAN

Pix2pix 학습절차



01. 딥러닝
02. 오토인코더
03. GAN

U-Net 구조

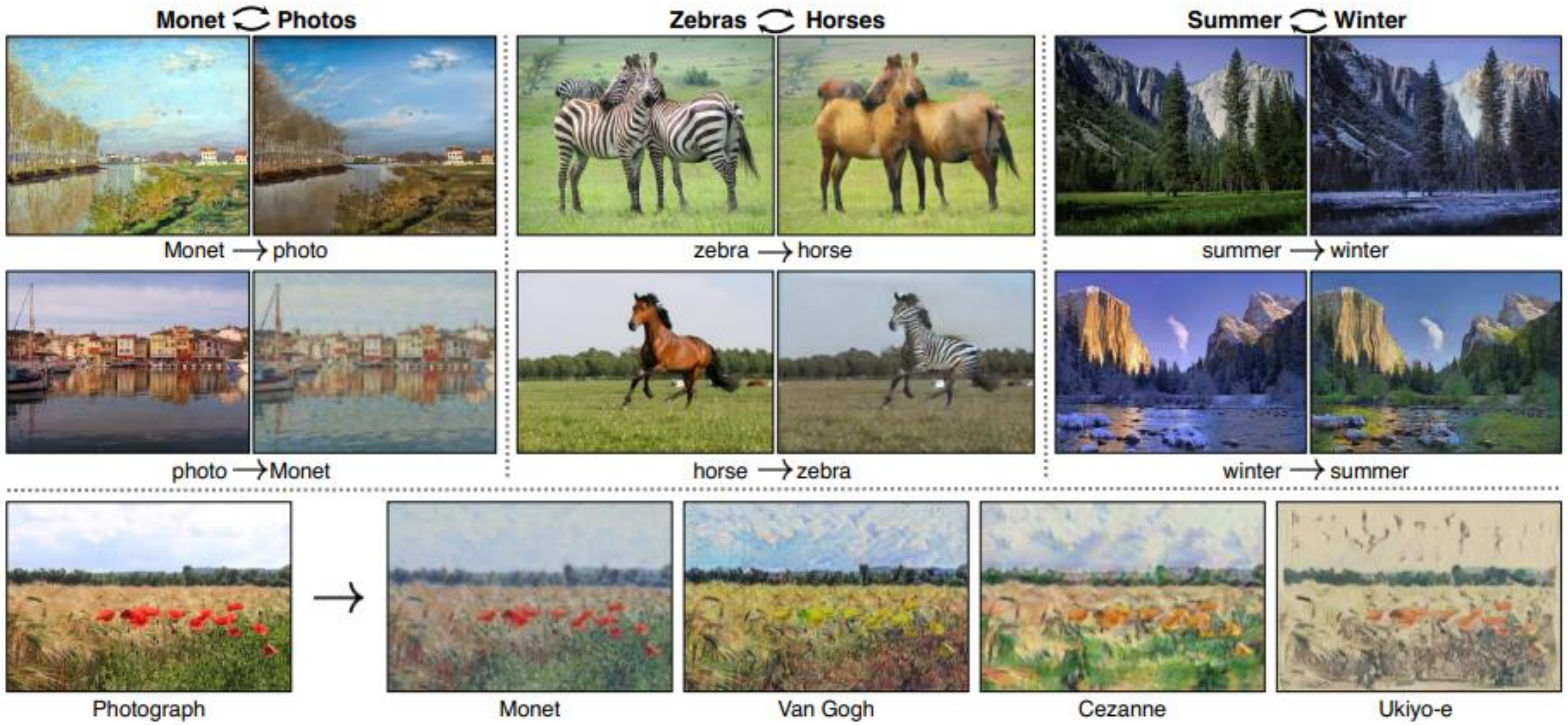


03

CycleGAN

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

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

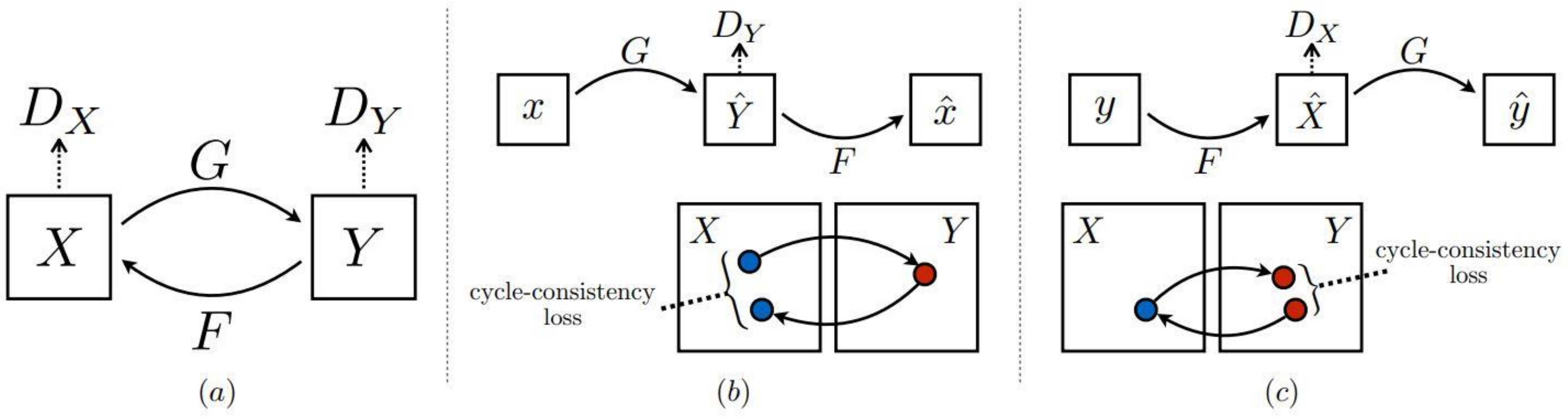


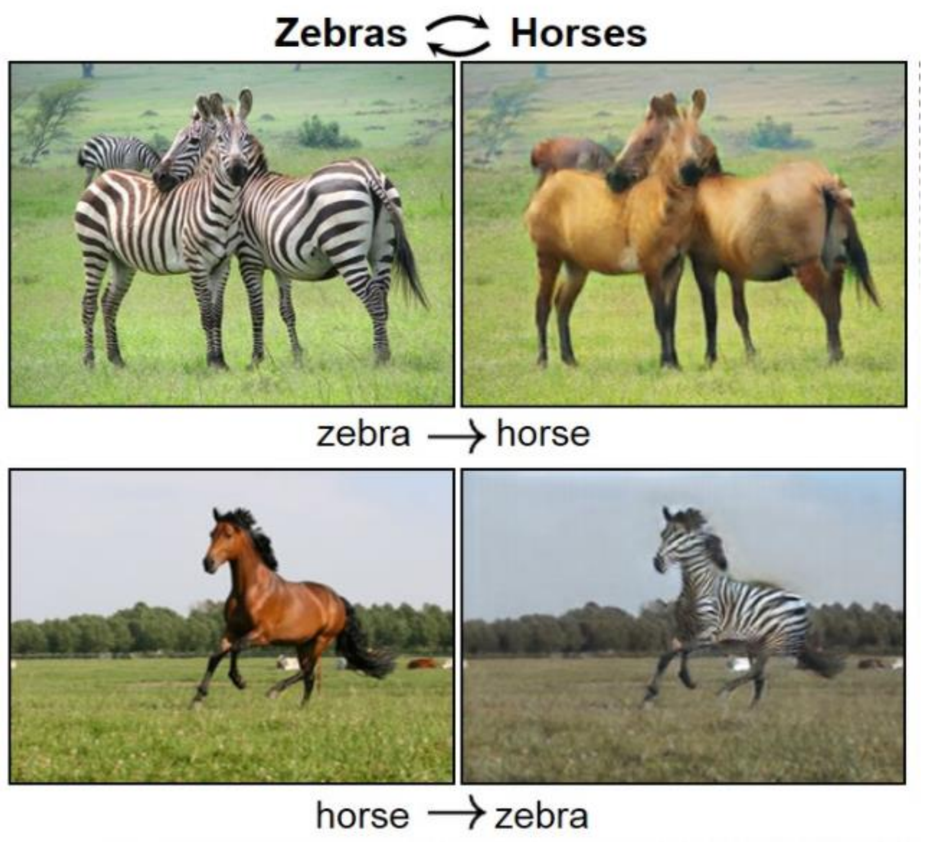
03 CycleGAN

01. CycleGAN 구조

02. 손실함수

CycleGAN 구조





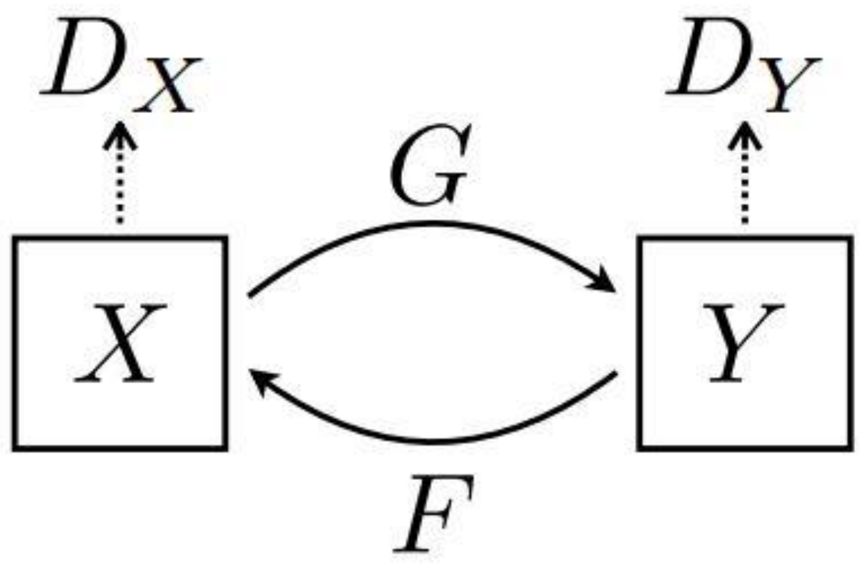
03 CycleGAN

01. CycleGAN 구조

02. 손실함수

적대적 손실

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



(a)

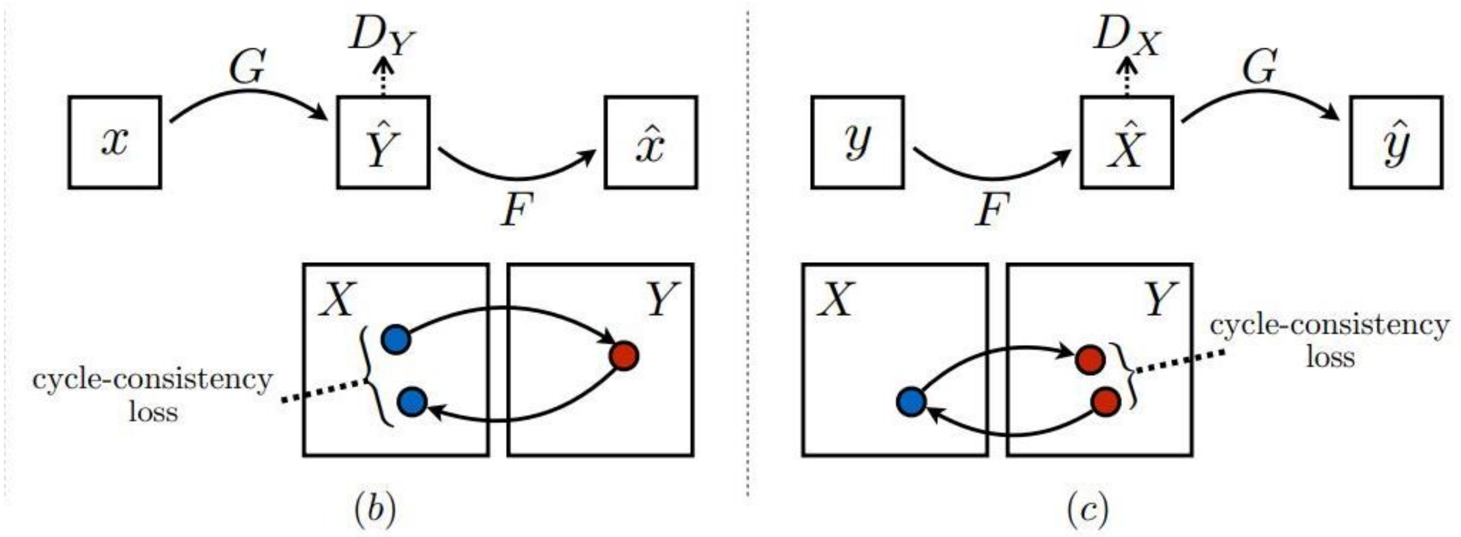
03 CycleGAN

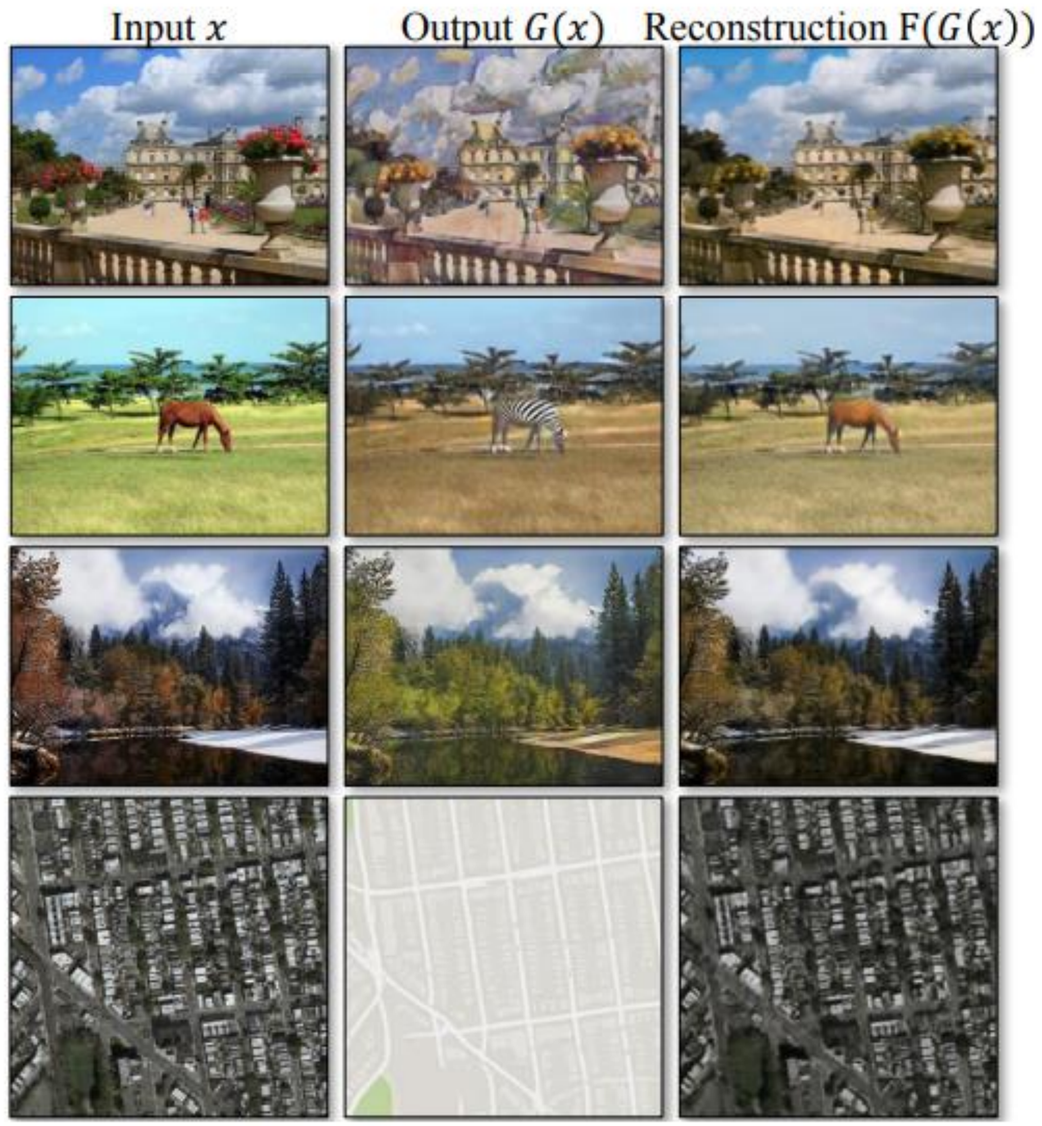
01. CycleGAN 구조

02. 손실함수

주기 일관성 손실

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



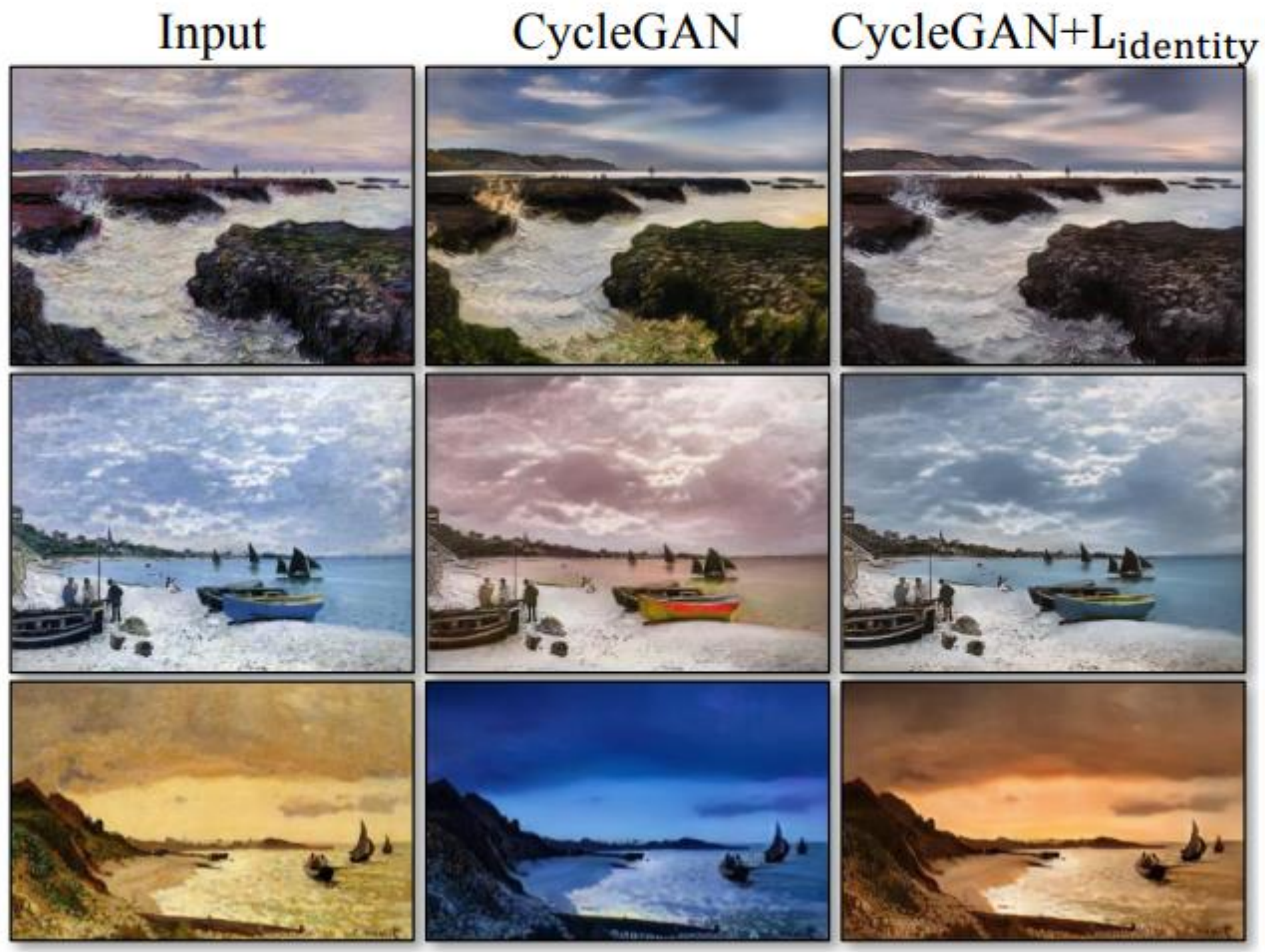


01. CycleGAN 구조

02. 손실함수

동일성 손실

$$L_{identity}(G, F) = E_{y \sim p_{data}(y)}[\|G(y) - y\|_1] + E_{x \sim p_{data}(x)}[\|F(x) - x\|_1]$$



03 CycleGAN

01. CycleGAN 구조

02. 손실함수

전체 목적함수

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

04 우리의 목표

04 우리의 목표

우리의 목표

- 학교 사진에 고흐를 포함한 여러 화가들의 화풍을 입히는 것에 도전
- 손실함수 최적화를 통한 성능향상 시도
- 파라미터 튜닝을 통한 성능향상 시도

05 구현코드 및 결과

05 구현코드 및 결과

01. 네트워크 구조
02. 구현 상세 코드
03. 구현결과

네트워크 구조 설명

01. 네트워크 구조
02. 구현 상세 코드
03. 구현결과

구현 상세 코드

```
class CycleGAN(CycleGAN):
    def build_generator(self):
        """U-Net 생성자"""
        d0 = Input(shape=self.img_shape)

        d1 = self.conv2d(d0, self.gf)
        d2 = self.conv2d(d1, self.gf * 2)
        d3 = self.conv2d(d2, self.gf * 4)
        d4 = self.conv2d(d3, self.gf * 8)

        d5 = self.conv2d(d4, self.gf * 16)
        d6 = self.conv2d(d5, self.gf * 32)

        u1 = self.deconv2d(d6, d5, self.gf * 16)
        u2 = self.deconv2d(u1, d4, self.gf * 8)
        u3 = self.deconv2d(u2, d3, self.gf * 4)
        u4 = self.deconv2d(u3, d2, self.gf * 1)
        u5 = self.deconv2d(u4, d1, self.gf)

        u6 = UpSampling2D(size=2)(u5)
        output_img = Conv2D(self.channels, kernel_size=4,
                            strides=1, padding='same', activation='tanh')(u6)

        return Model(d0, output_img)
```

05 구현코드 및 결과

01. 네트워크 구조

02. 구현 상세 코드

03. 구현결과

구현 상세 코드

```
class CycleGAN(CycleGAN):
    def build_discriminator(self):
        img = Input(shape=self.img_shape)

        d1 = self.conv2d(img, self.df, normalization=False)
        d2 = self.conv2d(d1, self.df * 2)
        d3 = self.conv2d(d2, self.df * 4)
        d4 = self.conv2d(d3, self.df * 8)
        d5 = self.conv2d(d4, self.df * 16)
        d6 = self.conv2d(d5, self.df * 32)

        validity = Conv2D(1, kernel_size=4, strides=1, padding='same')(d6)

        return Model(img, validity)
```

구현 상세 코드

01. 네트워크 구조
02. 구현 상세 코드
03. 구현결과

```
class CycleGAN(CycleGAN):
    def train(self, epochs, batch_size=1, sample_interval=50):

        valid = np.ones((batch_size,) + self.disc_patch)
        fake = np.zeros((batch_size,) + self.disc_patch)


        for epoch in range(epochs):
            for batch_i, (imgs_A, imgs_B) in enumerate(self.data_loader.load_batch(batch_size)):

                fake_B = self.g_AB.predict(imgs_A)
                fake_A = self.g_BA.predict(imgs_B)

                dA_loss_real = self.d_A.train_on_batch(imgs_A, valid)
                dA_loss_fake = self.d_A.train_on_batch(fake_A, fake)
                dA_loss = 0.5 * np.add(dA_loss_real, dA_loss_fake)

                dB_loss_real = self.d_B.train_on_batch(imgs_B, valid)
                dB_loss_fake = self.d_B.train_on_batch(fake_B, fake)
                dB_loss = 0.5 * np.add(dB_loss_real, dB_loss_fake)

                d_loss = 0.5 * np.add(dA_loss, dB_loss)

                g_loss = self.combined.train_on_batch([imgs_A, imgs_B],
                                                      [valid, valid,
                                                       imgs_A, imgs_B,
                                                       imgs_A, imgs_B])

                if batch_i % sample_interval == 0:
                    self.sample_images(epoch, batch_i)
```

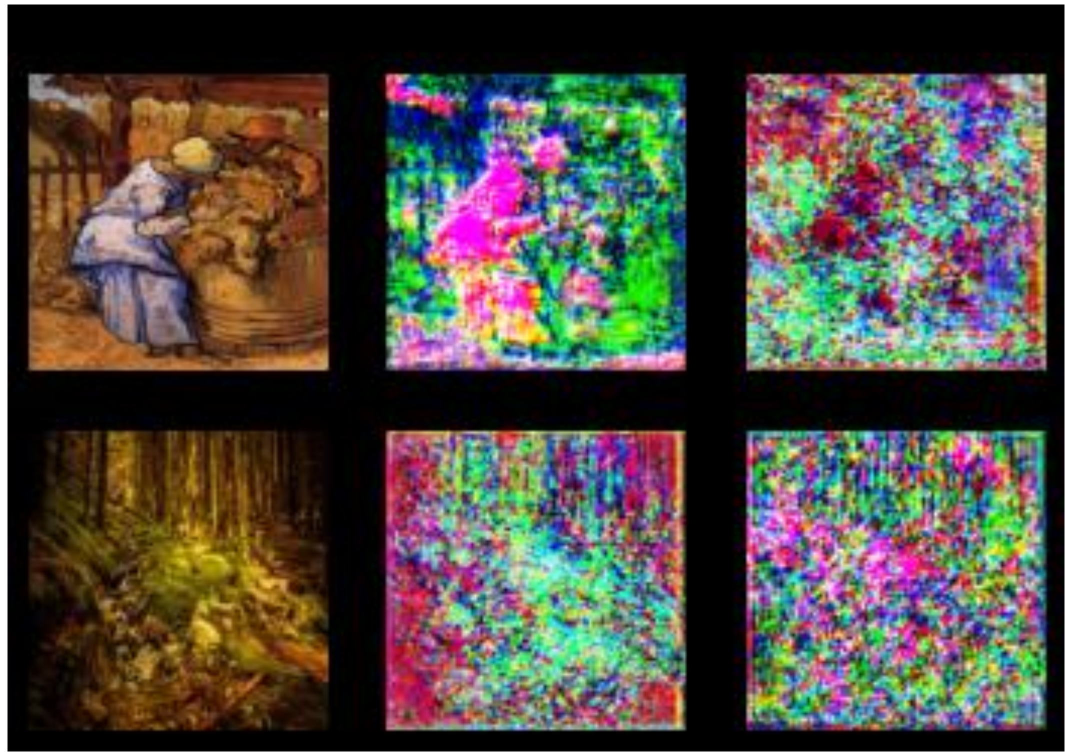
05 구현코드 및 결과

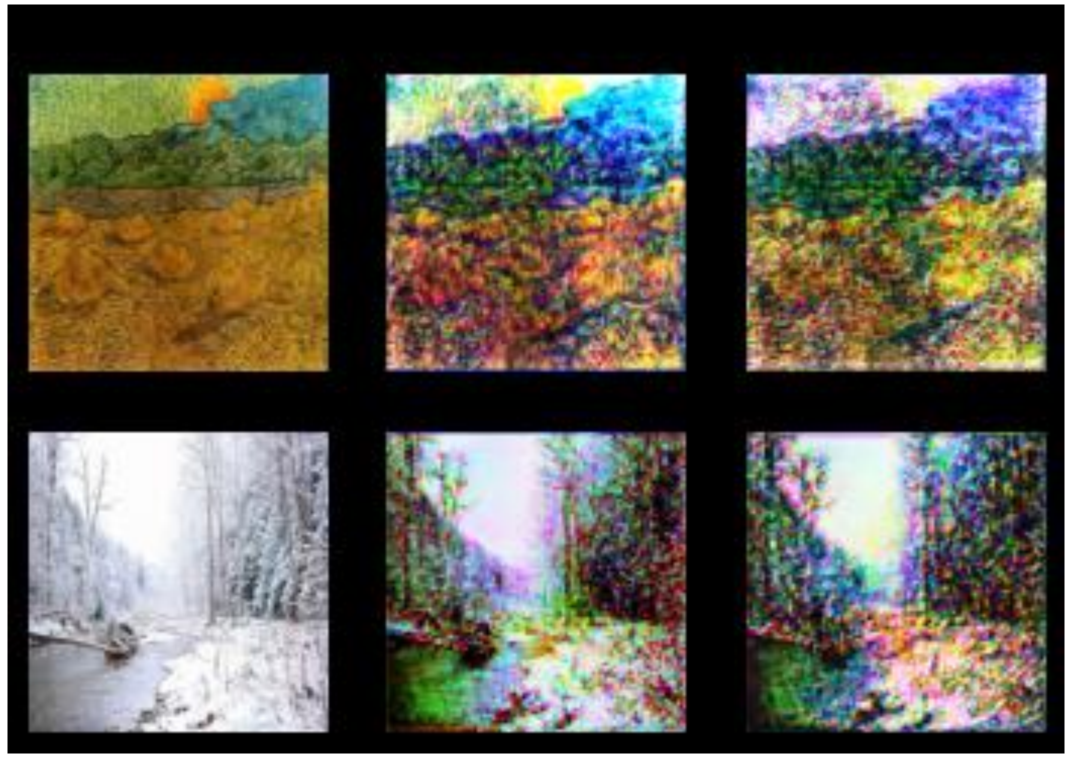
01. 네트워크 구조

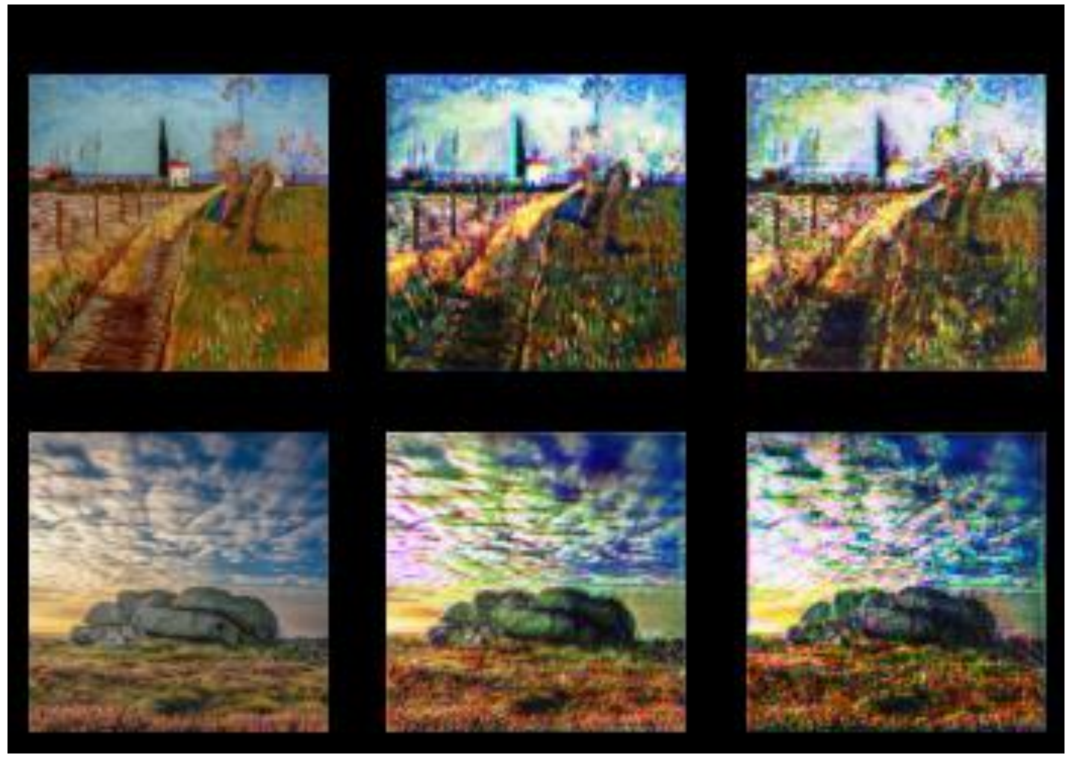
02. 구현 상세 코드

03. 구현결과

구현결과







05 구현코드 및 결과

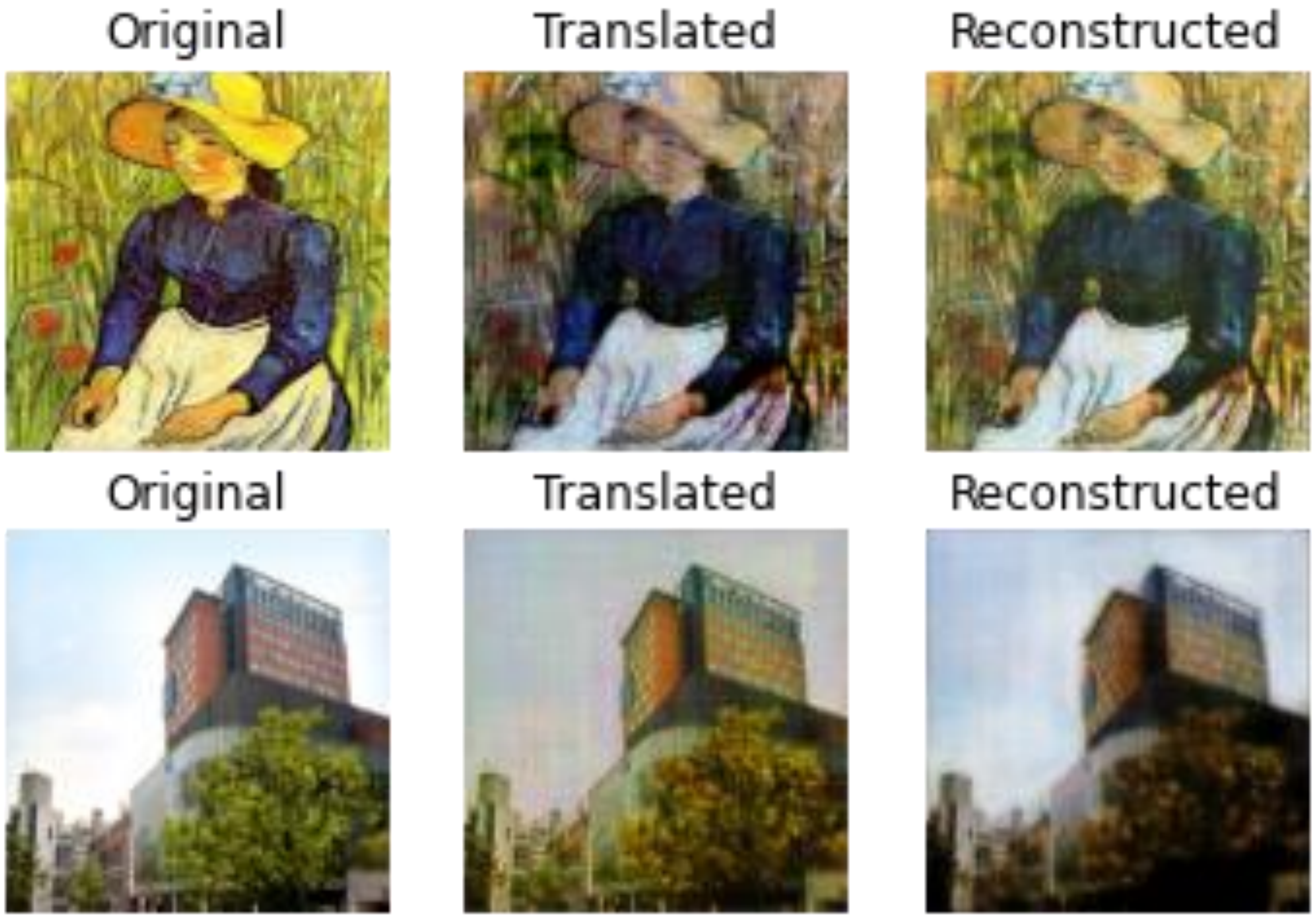
01. 네트워크 구조

02. 구현 상세 코드

03. 구현결과

구현결과

구현결과


Original
Translated
Reconstructed
Original
Translated
Reconstructed

05 구현코드 및 결과

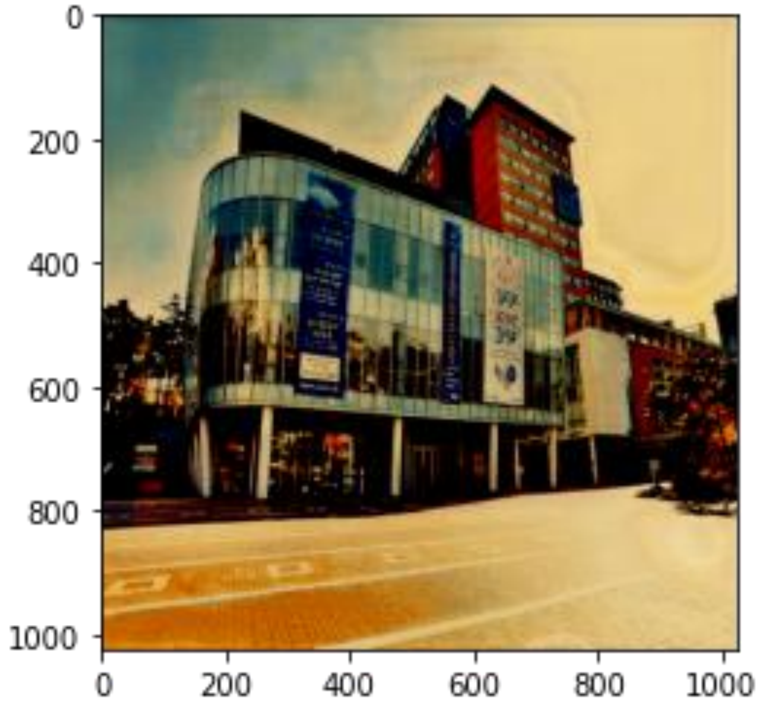
01. 네트워크 구조

02. 구현 상세 코드

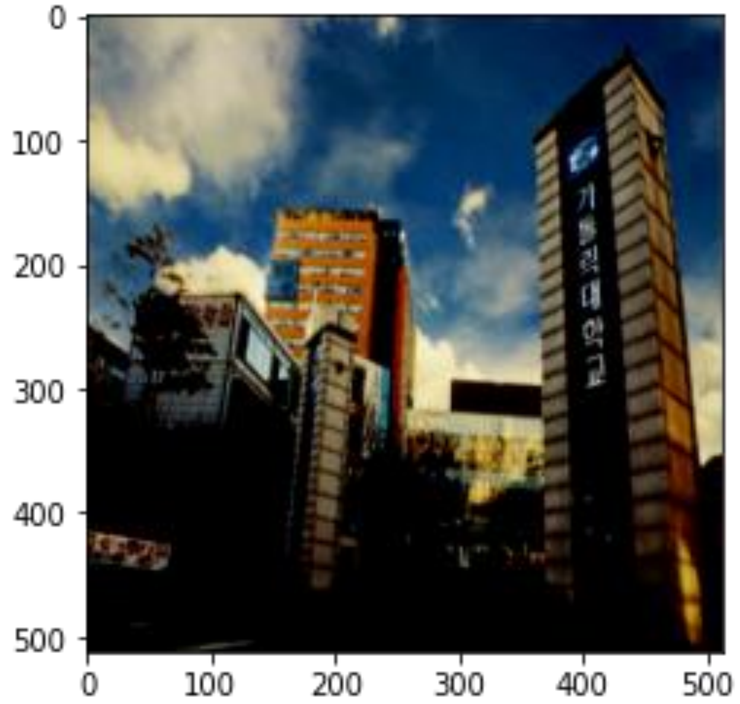
03. 구현결과

구현결과

Cézanne



monet



05 구현코드 및 결과

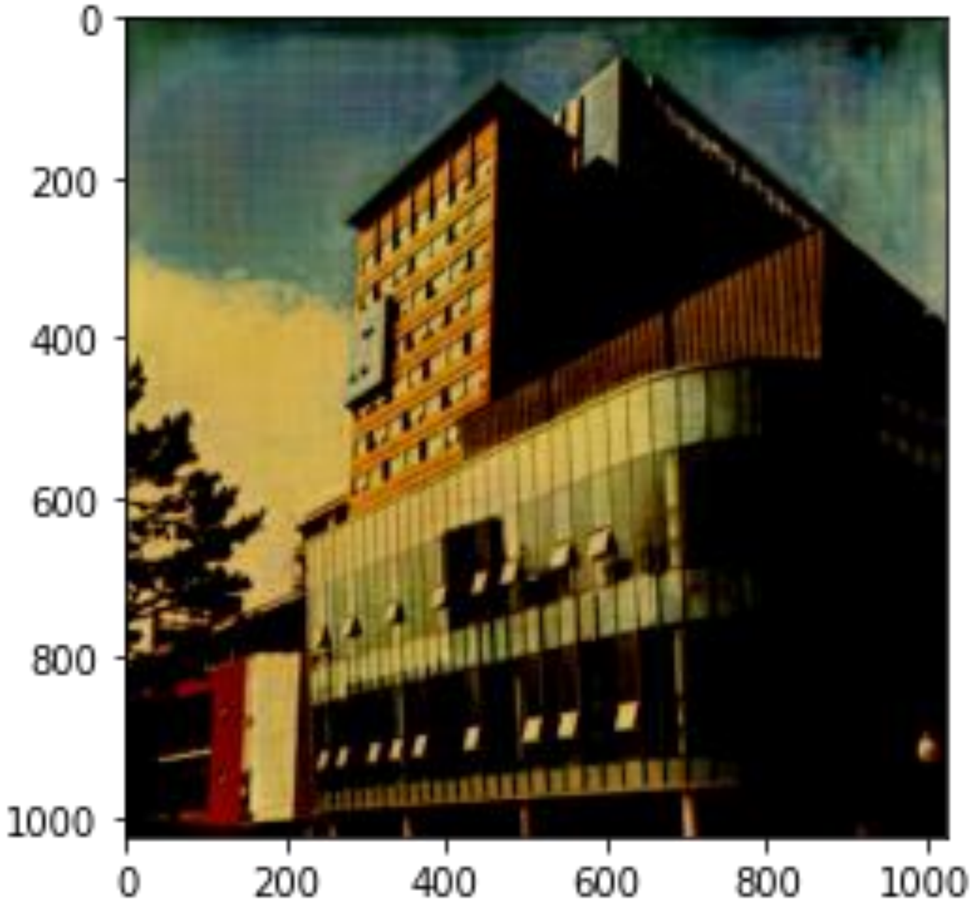
01. 네트워크 구조

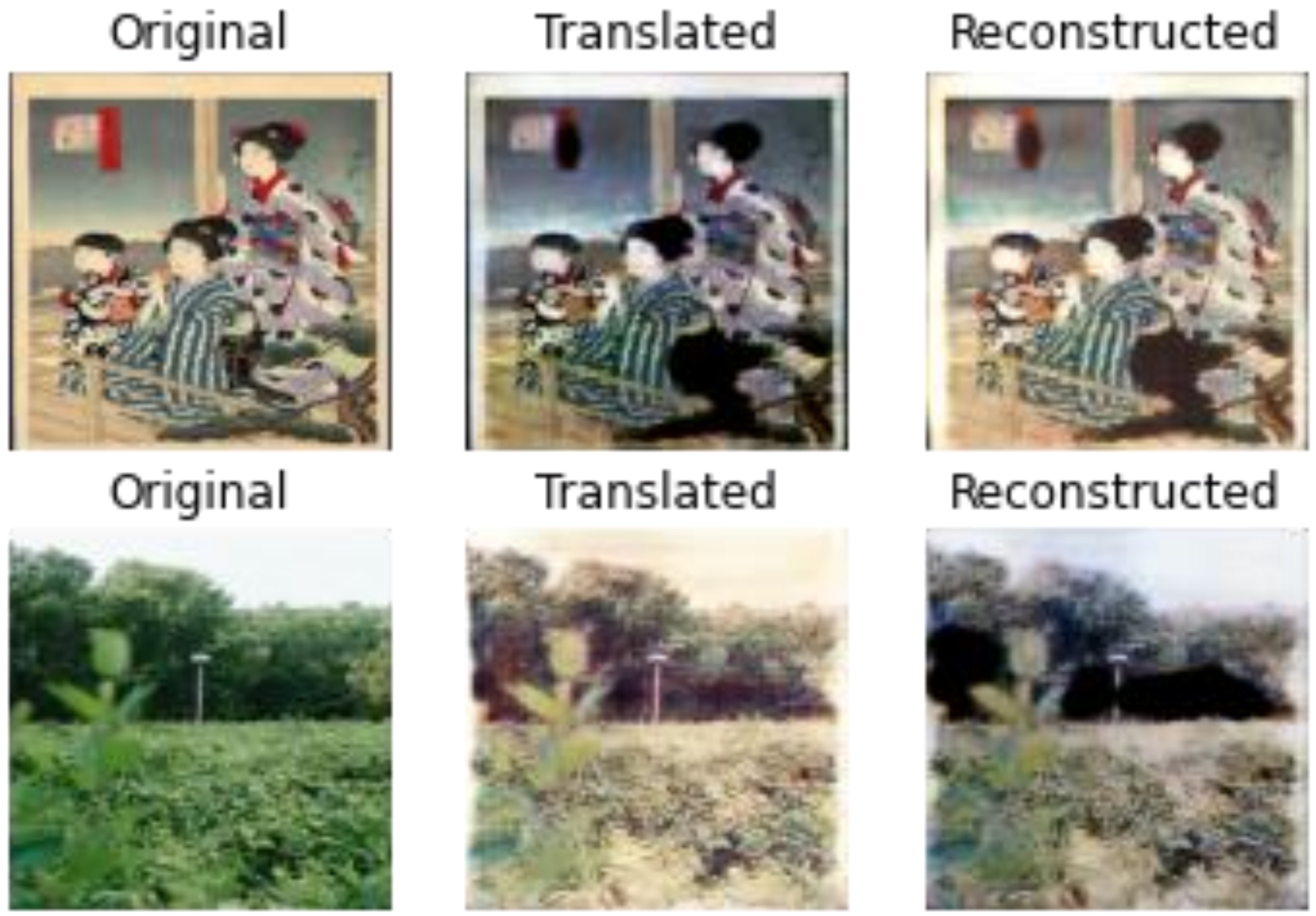
02. 구현 상세 코드

03. 구현결과

구현결과

ukiyoe





06

결론

고찰

유망사용분야: 딥페이크

아심웠던 점: 파라미터 튜닝,
GPU의 한계, 데이터 부족

후속연구

감사합니다^◡^