

데이터분석경진대회

(w i t h G A N s)

201720014강민주
201820973김정태
201820995박은비

목차

01 해결하고자 하는 과제

02 사용한 이미지 데이터

03 전처리 방안

04 활용 기법

05 생성된 모형의 실생활 적용 방안

06 프로그래밍 코드 및 결과

01



해결하고자 하는 과제

01

해결하고자 하는 과제

해결하고자 하는 과제



apple → orange



orange → apple



02



사용한 이미지 데이터




















02

사용한 이미지 데이터

사용한 이미지 데이터

[https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/\\$FILE.zip](https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/$FILE.zip)

Index of /~taesung_park/CycleGAN/datasets

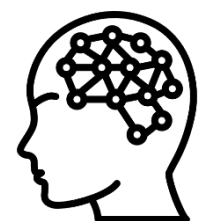
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 NOTICE	2019-08-12 20:45	227	
 ae_photos.zip	2017-04-03 22:06	10M	
 apple2orange.zip	2017-03-28 13:51	75M	
 cezanne2photo.zip	2017-03-28 13:51	267M	
 cityscapes.zip	2019-08-12 20:45	325	
 facades.zip	2017-03-29 23:23	34M	
 grumpifycat.zip	2020-08-03 20:58	19M	
 horse2zebra.zip	2017-03-28 13:51	111M	
 iphone2dslr_flower.zip	2017-03-30 12:05	324M	
 maps.zip	2017-03-26 19:17	1.4G	
 mini.zip	2018-06-07 16:05	1.8M	
 mini_colorization.ta..>	2019-01-01 16:38	303K	
 mini_colorization.zip	2019-01-01 16:44	304K	
 mini_pix2pix.zip	2018-06-07 16:08	1.5M	
 monet2photo.zip	2017-03-26 19:17	291M	
 summer2winter_yosemite..>	2017-03-26 19:17	126M	
 ukiyoe2photo.zip	2017-03-26 19:17	279M	
 vangogh2photo.zip	2017-03-26 19:17	292M	

03

전처리 방안

03 전처리 방안

전처리 방안



이미지의 차원을 256×256 으로 변환



이미지를 -1에서 1 까지의 값으로 정규화

04

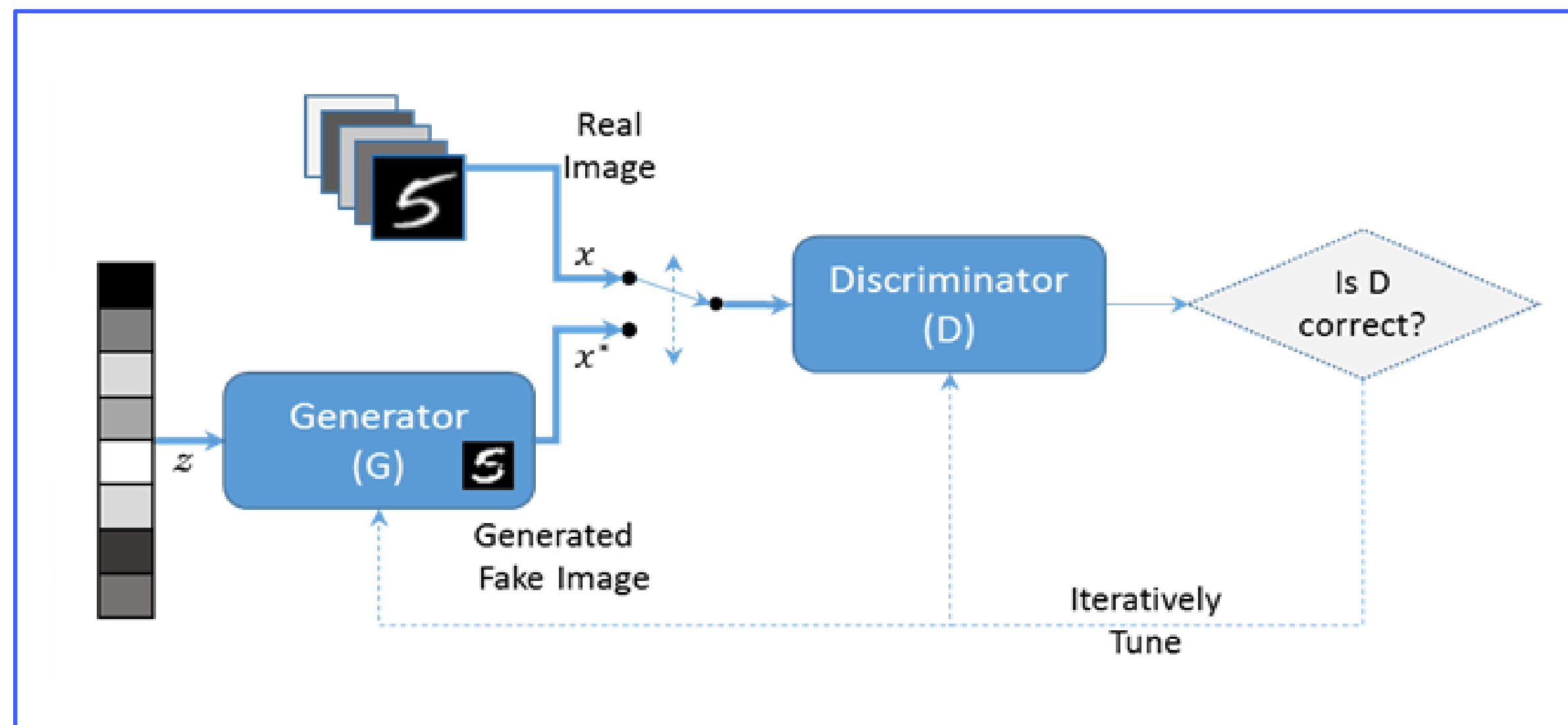
활용 기본

04 활용기법

01. GAN

02. CycleGAN

GAN

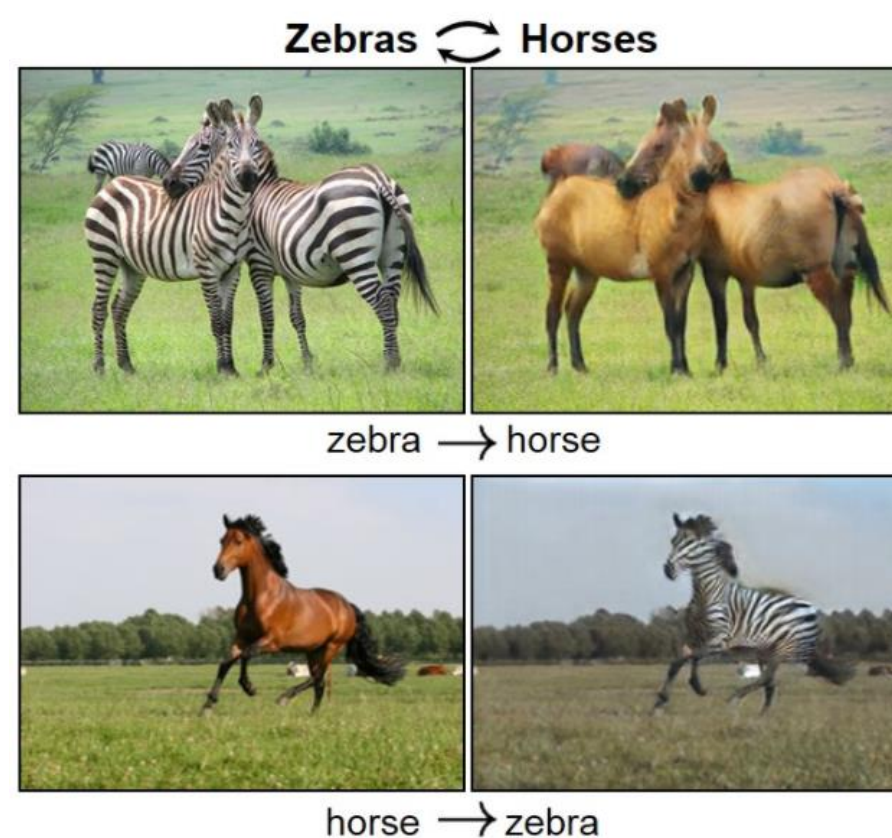
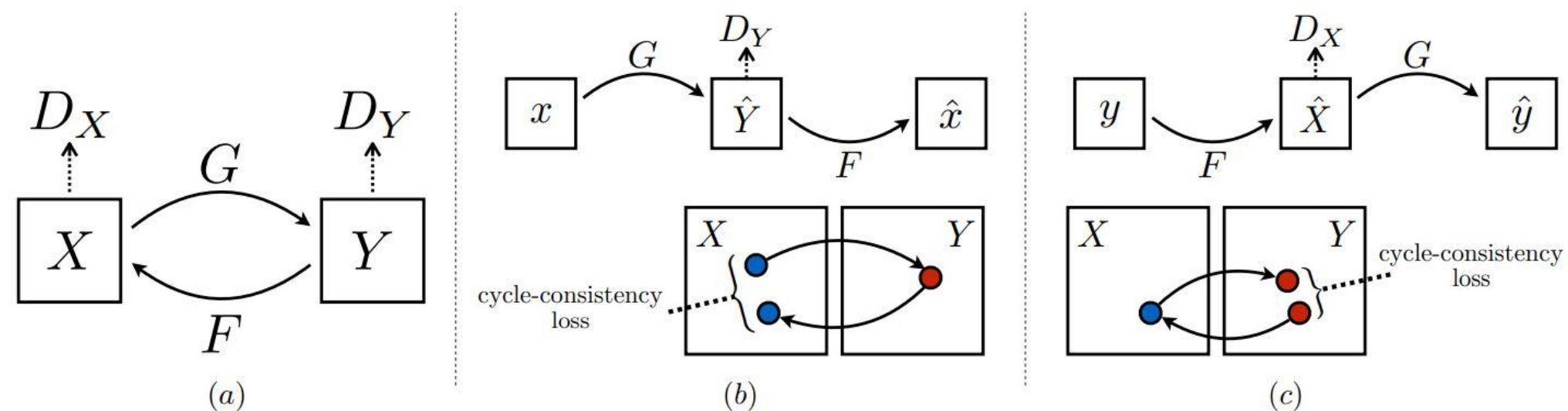


04 활용기법

01. GAN

02. CycleGAN

CycleGAN



05



생성된 모형의
실생활 적용 방안

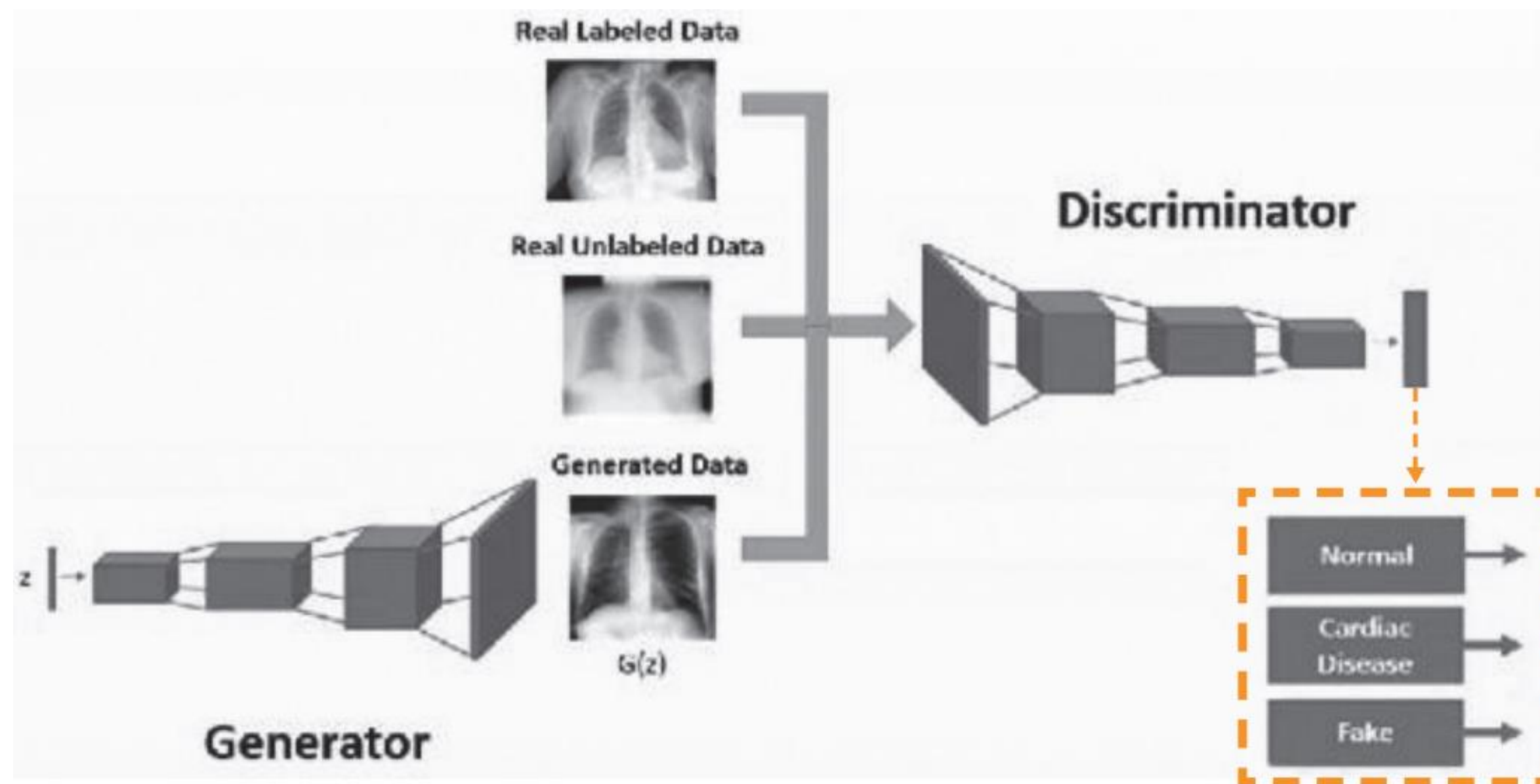
05 생성된 모형의 실생활 적용 방안

01. 의료데이터

02. 금융데이터

03. 자율주행

의료데이터에서의 데이터 부족 해결



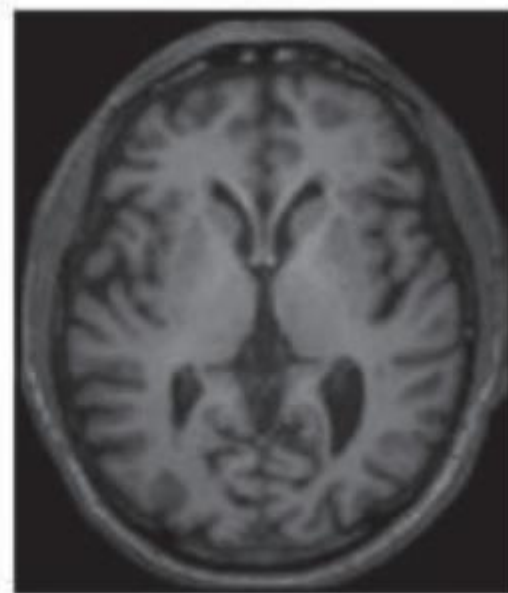
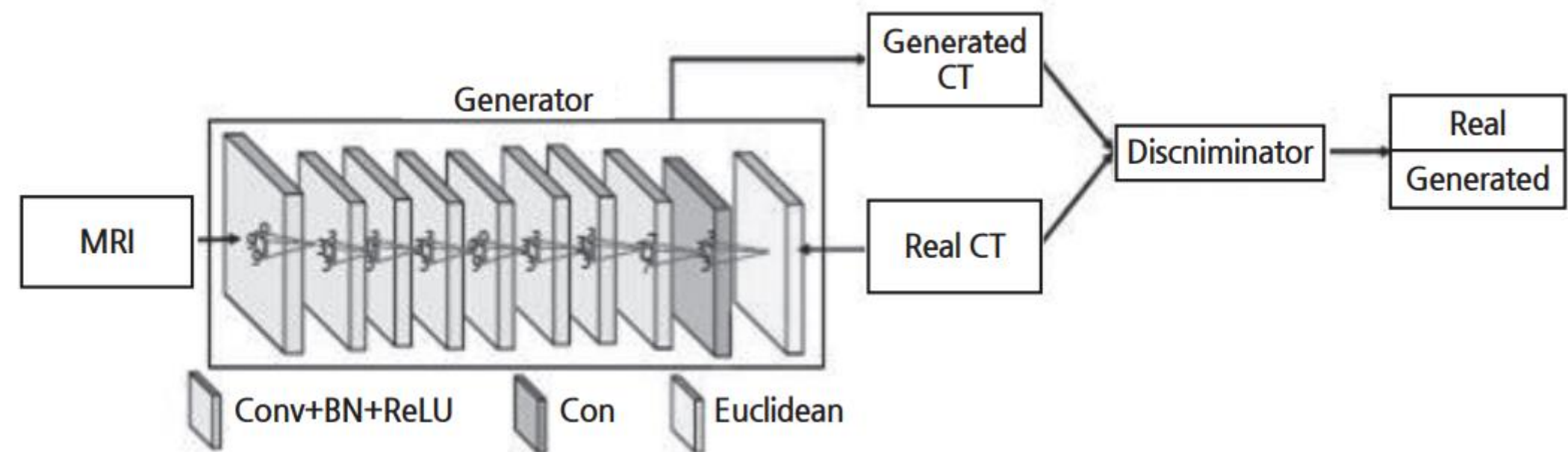
05 생성된 모형의 실생활 적용 방안

01. 의료데이터

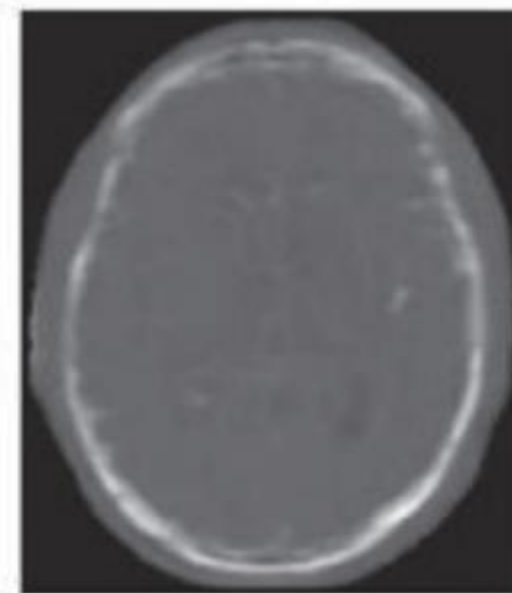
02. 금융데이터

03. 자율주행

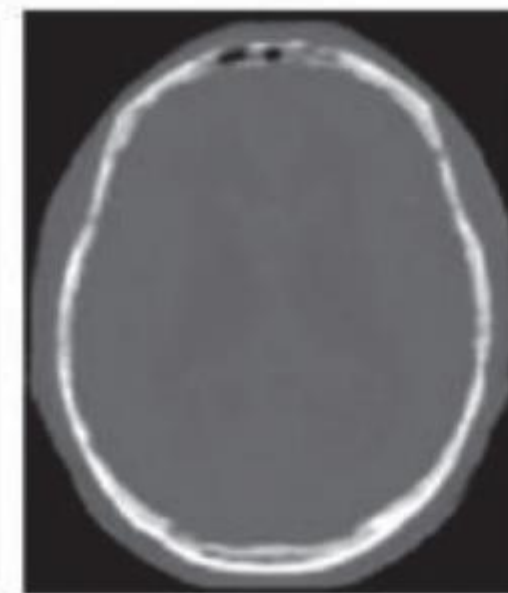
의료데이터에서의 데이터 부족 해결



MRI



FCN



GAN

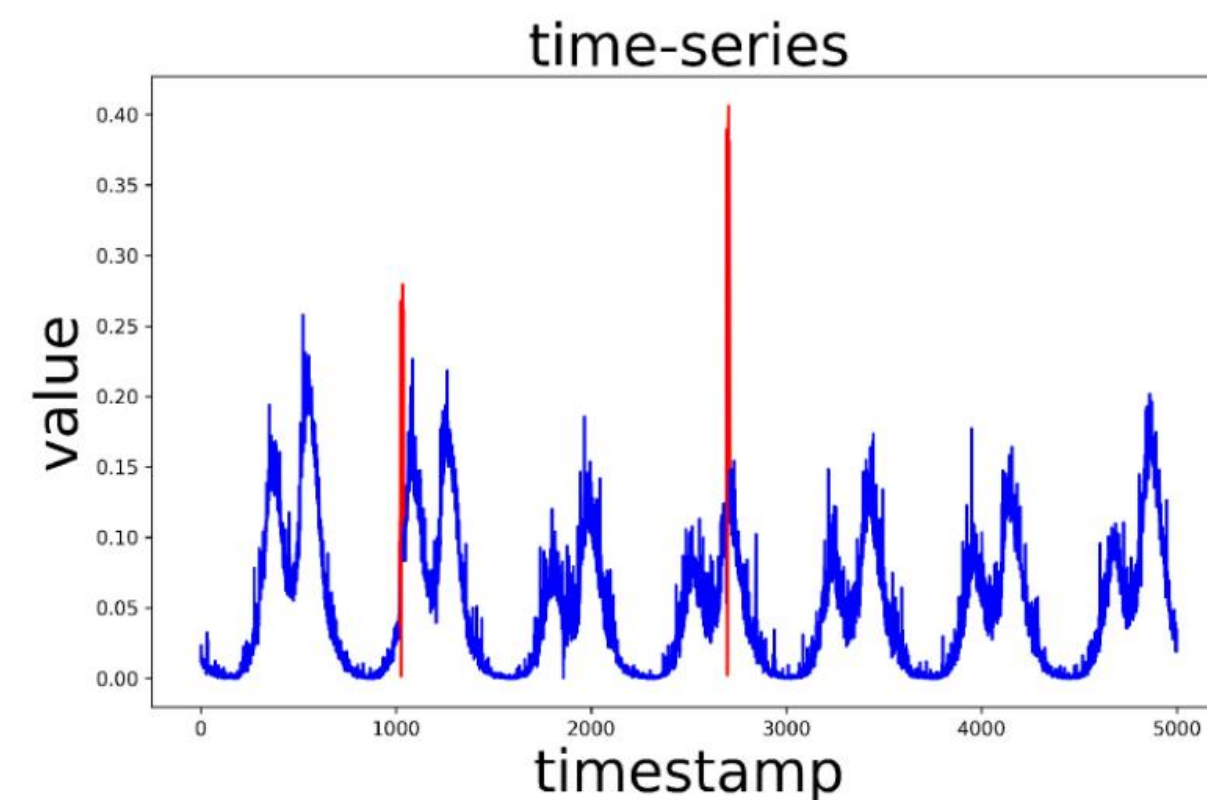
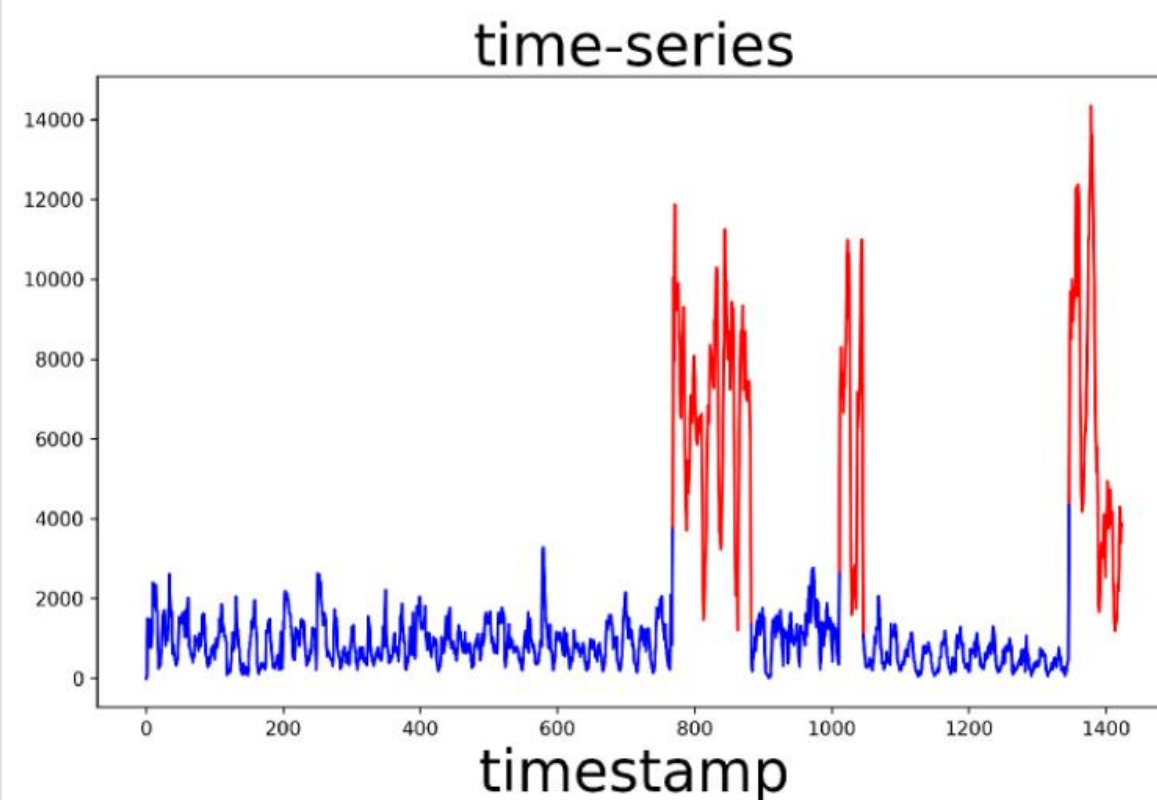
05 생성된 모형의 실생활 적용 방안

01. 의료데이터

02. 금융데이터

03. 자율주행

금융 데이터에서의 이상치 탐지



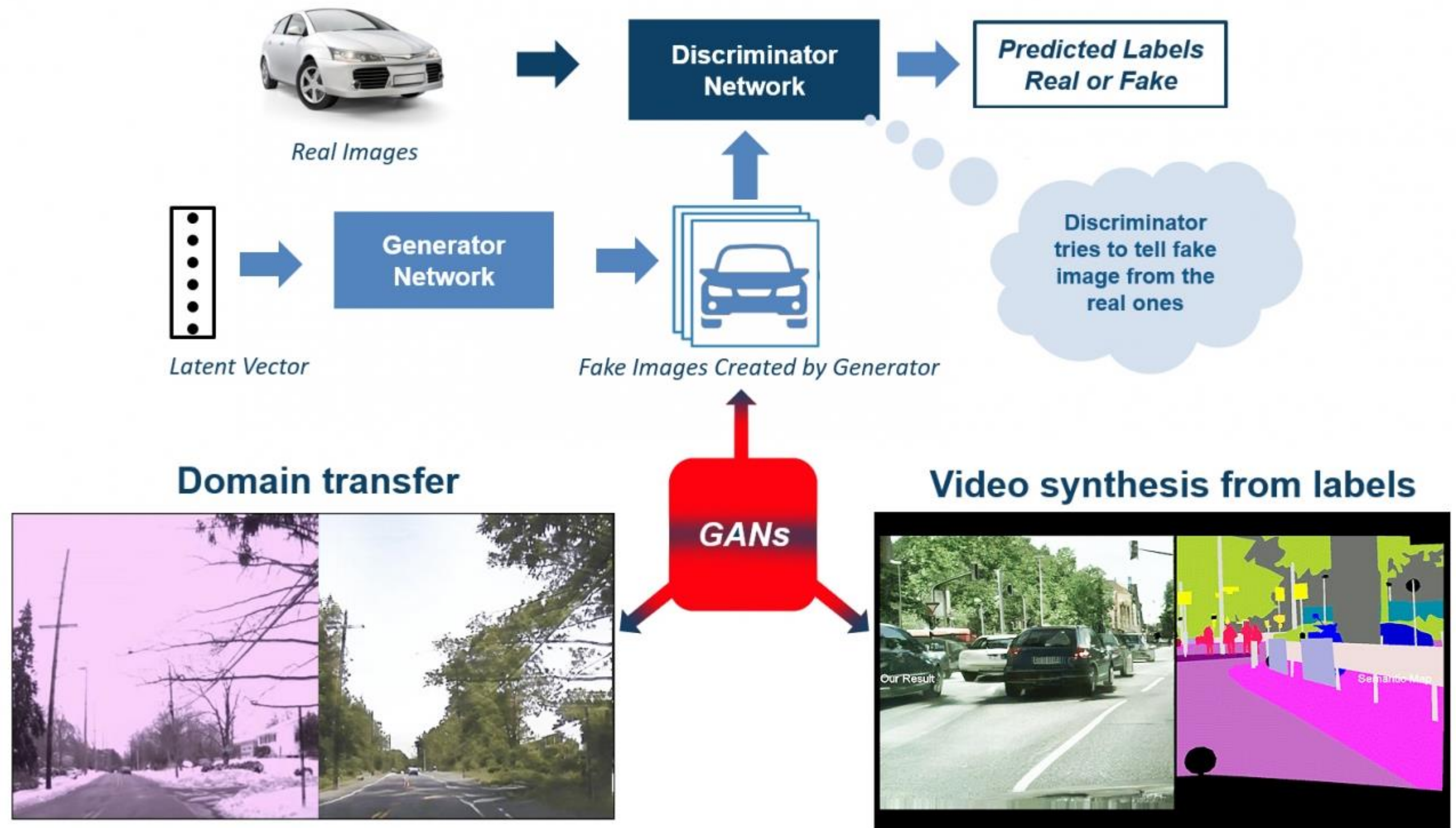
05 생성된 모형의 실생활 적용 방안

01. 의료데이터

02. 금융데이터

03. 자율주행

자율주행을 위한 가상 데이터 생성



06



프로그래밍
코드 및 결과

06

프로그래밍
코드 및 결과

구현 코드

01. 구현 코드

02. 구현 결과

```
class CycleGAN(CycleGAN):
    @staticmethod
    def conv2d(layer_input, filters, f_size=4, normalization=True):
        """다운샘플링하는 동안 사용되는 층"""
        d = Conv2D(filters, kernel_size=f_size,
                    strides=2, padding='same')(layer_input)
        d = LeakyReLU(alpha=0.2)(d)
        if normalization:
            d = InstanceNormalization()(d)
        return d

    @staticmethod
    def deconv2d(layer_input, skip_input, filters, f_size=4, dropout_rate=0):
        """업샘플링하는 동안 사용되는 층"""
        u = UpSampling2D(size=2)(layer_input)
        u = Conv2D(filters, kernel_size=f_size, strides=1,
                    padding='same', activation='relu')(u)
        if dropout_rate:
            u = Dropout(dropout_rate)(u)
        u = InstanceNormalization()(u)
        u = Concatenate()([u, skip_input])
        return u
```

06

프로그래밍
코드 및 결과

구현 코드

01. 구현 코드

02. 구현 결과

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

        # 다운샘플링
        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)

        # 업샘플링
        u1 = self.deconv2d(d4, d3, self.gf * 4)
        u2 = self.deconv2d(u1, d2, self.gf * 2)
        u3 = self.deconv2d(u2, d1, self.gf)

        u4 = UpSampling2D(size=2)(u3)
        output_img = Conv2D(self.channels, kernel_size=4,
                             strides=1, padding='same', activation='tanh')(u4)

        return Model(d0, output_img)
```

06

프로그래밍
코드 및 결과

구현 코드

01. 구현 코드

02. 구현 결과

```
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)  
  
        validity = Conv2D(1, kernel_size=4, strides=1, padding='same')(d4)  
  
        return Model(img, validity)
```

06

프로그래밍
코드 및 결과

01. 구현 코드

02. 구현 결과

구현 코드

```
class CycleGAN(CycleGAN):
    def train(self, epochs, batch_size=1, sample_interval=50):
        # 적대 손실에 대한 정답
        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)):

                # 이미지를 상대 도메인으로 변환합니다.
                fake_B = self.g_AB.predict(imgs_A)
                fake_A = self.g_BA.predict(imgs_B)

                # 판별자를 훈련합니다. (원본 이미지 = real / 변환된 이미지 = fake)
                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)

                # 판별자 전체 손실
                d_loss = 0.5 * np.add(dA_loss, dB_loss)

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

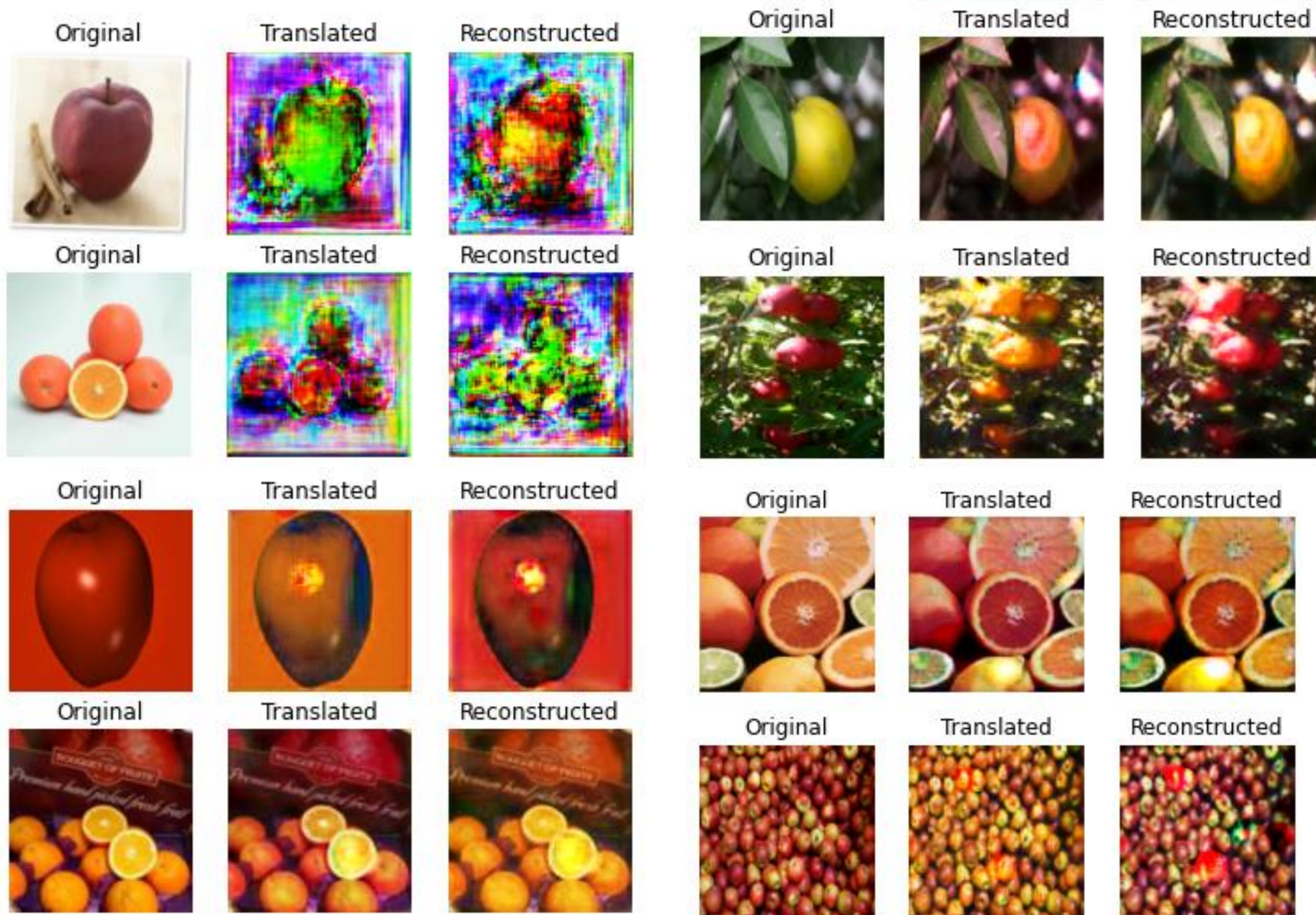
                # save_interval 마다 생성된 이미지 샘플을 저장합니다.
                if batch_i % sample_interval == 0:
                    self.sample_images(epoch, batch_i)
```

06 프로그래밍 코드 및 결과

01. 구현 코드

02. 구현 결과

구현 결과



06

프로그래밍
코드 및 결과

01. 구현 코드

02. 구현 결과

구현 결과

Original



Translated



Reconstructed



Original



Translated



Reconstructed



감사합니다~
