데이터 분석 경진대회 보고서

수학전공
201720014 강민주
201820973 김정태
201820995 박은비

1. 해결하고자 하는 과제

2. 사용한 이미지 데이터

3. 전처리 방안

4. 활용 기법
    1. GAN
    2. CycleGAN

5. 생성된 모형의 실생활 적용 방안
    1. 의료데이터에서의 데이터 부족 해결
    2. 금융 데이터에서의 이상치 탐지
    3. 자율 주행을 위한 가상 데이터 생성

6. 프로그래밍 코드 및 결과
    1. 구현 코드
    2. 구현 결과

1. 해결하고자 하는 과제

우리는 GAN 중 CycleGAN을 이용해서 사과를 오렌지로 바꾸고, 오렌지를 사과로 바꿔보려고 한다. 이를 통해 짝지어진 데이터가 없거나 얻기 힘든 경우에도 사용할 수 있는 이미지 변환에 대해 알아보고 실생활에서는 어떻게 활용될 수 있는지 알아보고자 한다.

2. 사용 이미지 데이터

사용한 이미지는 다음 링크에서 다운받은 CycleGAN의 datasets인 apple2orange.zip을 사용했다. https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/

3. 전처리 방안

너무 큰 화질의 이미지는 학습에 너무 오랜 시간이 소요되므로 이미지들을 128×128차원으로 변환하였다. 이미지 데이터 전부 -1에서 1까지의 값으로 정규화(Normalization)하였다.

4. 활용기법

4-1. GAN

GAN(generative Adversarial Network)이란 적대적 생성 신경망으로, 동시에 두 개의 모델을 훈련하는 머신러닝의 한 종류이다. 여기서의 두 개의 모델은 생성모델과 분류모델을 뜻한다. 생성모델은 실제 데이터를 학습한 것을 바탕으로 진짜 같은 거짓데이터를 만들어낸다. 진짜 같은 가짜를 만드는 것이 목적이며 계속 진짜를 만들어내면서 분류모델을 속이는 것을 목표로 학습을 한다. 반대로 분류모델은 생성모델이 만들어낸 데이터가 진짜 데이터인지 가짜 데이터인지 판별하는 것을 목표로 한다.

Generative(생성적)는 GAN의 목적을 잘 설명하는 용어이다. 즉 새로운 데이터를 생성한다는 의미이다. GAN이 생성하기 위해 학습할 데이터는 훈련 세트에 따라서 결정된다. 예를 들어 GAN을 이용해 고흐가 만든 것 같은 작품을 만들고 싶다면 고흐의 작품들을 데이터셋으로 사용해야 한다. Adversarial(적대적)은 생성자와 판별자의 경쟁 구도를 나타낸다. 생성자의 목표는 판별자가 데이터셋에 있는 실제 데이터와 구분이 안 될 정도로 유사한 샘플을 만드는 것이다. 판별자의 목표는 생성자가 만든 가짜 데이터를 실제 데이터와 구별하는 것이다. 이렇게 생성자와 판별자는 서로를 이기려는 경쟁을 지속한다. 생성자가 더 그럴싸한 데이터를 생성할수록 판별자 역시 가짜 데이터와 진짜 데이터를 구별하는 일을 더 잘할 줄 알아야 한다. 마지막으로 Network(신경망)는 생성자와 판별자를 만드는 데 가장 널리 사용하는 머신러닝 모델 중 한 종류이다.

GAN의 수학적 기반은 복잡하지만, 현실 세계의 비유를 활용하면 더 쉽게 이해할 수 있다. 형사를 판별자, 지폐 위조범을 생성자라고 생각해보자. 위조범이 위조지폐를 더 그럴듯하게 만들수록 형사가 지폐를 판별하는 능력 또한 더 뛰어나야 한다. 반대상황에도 마찬가지로 형사가 위조지폐를 찾는 능력이 뛰어날수록 위조범 역시 잡히지 않기 위해 더 지폐와 구별하기

어려운 위조지폐를 만들어내야 한다.

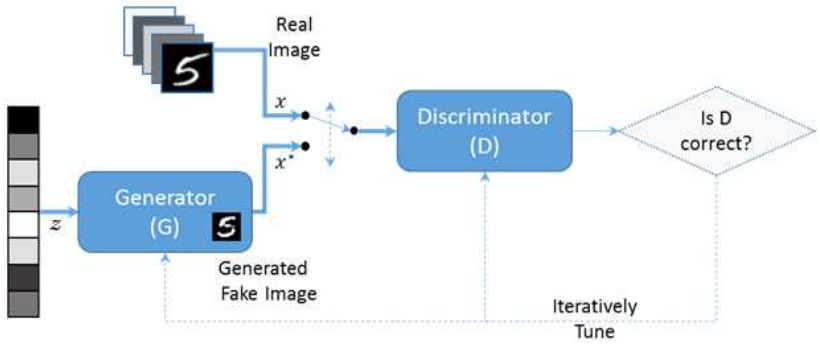

그림. GAN의 구조

실제 이미지(Real Image)는 생성자가 그림을 모방하기 위해서 학습하는 진짜 샘플 데이터셋이다. 위 사진에선 MNIST 데이터셋으로 이루어져 있다. 이 데이터는 판별자의 훈련을 위해서 판별자의 입력(x)으로 제공된다.

z는 생성자 네트워크에 입력으로 이 입력부터 생성자가 가짜 샘플 합성을 시작한다. 생성자는 무작위로 숫자 벡터(z)를 받아서 가짜 샘플(x^*)을 출력한다. 생성자의 목표는 훈련 데이터셋의 진짜 샘플과 구별이 안 되는 가짜 샘플을 생성하는 것이다. 판별자는 훈련 데이터셋의 진짜 샘플(x)과 생성자가 만든 가짜 샘플(x^*)을 입력으로 받고 각 샘플이 진짜일 확률을 계산해 출력한다. 이 모든 과정을 계속 반복하여 생성자와 판별자를 훈련시킨다. 판별자는 진짜 샘플(x)와 가짜 샘플(x^*)을 제대로 구분하도록 훈련하고, 생성자는 판별자가 가짜 샘플(x^*)을 진짜로 잘못 분류할 확률을 최대화하도록 훈련한다.

훈련은 생성자가 실제 데이터와 구별이 안 되는 데이터를 생성하여, 판별자가 두 데이터를 50 대 50의 확률로 랜덤하게 추측하게 됐을 때 훈련을 종료하게 된다. 이 이상으로 훈련을 진행하게 되면 과대적합이 진행이 되어 오히려 나쁜 데이터를 생성하게 된다.

4-2. CycleGAN

우리는 GAN의 한 종류 중 **CycleGAN**을 이용해 문제를 해결하려고 한다. 우선 이미지 대 이미지(image-to-image) 변환이란 짝진 형태의 이미지 훈련 세트를 이용해 입력 이미지와 출력 이미지를 매핑하는 것이 목표인 컴퓨터 비전과 그래픽의 한 분야이다. 그러나 많은 작업의 경우 짝이 지어진 학습 데이터를 얻는 것은 어렵다. 예를 들어 흑백 이미지의 경우 먼저 컬러 사진을 로드하여 흑백 필터를 적용한다. 그다음 원본 이미지를 한 도메인으로 사용하고 흑백 필터를 적용한 이미지를 다른 도메인으로 사용한다. 이렇게 하면 양쪽 도메인에 모두 존재하는 이미지를 준비할 수 있고, 그다음 훈련된 GAN을 다른 곳에 적용할 수 있다. 하지만 완벽한 이런 쌍을 준비하기 쉽지 않다면 모델을 학습하기 어렵다고 할 수 있다.

짝지어진 훈련 데이터가 없는 경우, 한 이미지 컬렉션의 특수 특성을 잡아내고 이러한 특성이 다른 이미지 컬렉션으로 어떻게 바꿀 수 있는지 알아내는 것이 요점이다. 이 문제는 더 광범위하게 이미지를 주어진 장면의 한 표현에서 다른 표현으로 바꾸는 이미지 변환으로 설명될 수 있다. 그러나 짝지어진 훈련 데이터를 얻는 것은 어렵고 비용이 많이 들 수 있다.

따라서 CycleGAN은 두 가지 훈련 데이터가 없는 경우, 짝지어진 입력과 출력 데이터 없이 도메인 간 이미지를 변환하는 방법을 학습하기 위한 접근방식을 제시한다. 우선 도메인들 사이에 몇 가지 근본적인 관계가 있다고 가정한다. 예를 들어 이미지들은 동일한 근본적인 장면의 서로 다른 렌더링이라고 생각하고 그 관계를 학습한다. 여기서 렌더링이란 컴퓨터 프로그램을 사용하여 모델 또는 이들을 모아놓은 장면인 씬 파일로부터 영상을 만들어내는 과정을 말한다. 비록 쌍체 데이터의 형태로 지도는 부족하지만 세트 레벨에서 지도할 수 있다. 우선 도메인 X의 이미지 세트와 도메인 Y의 다른 세트를 받는다. 그리고 출력 $\hat{y} = G(x), x \in X$가 $\hat{y}$과 y를 구분하도록 훈련된 적수에 의해 이미지 $y \in Y$와 구별되지 않도록 매핑 $G : X \rightarrow Y$를 훈련한다. 이론적으로 이 목표는 경험적 분포 $p_{data}(y)$와 일치하는 $\hat{y}$에 대한 출력 분포를 유도할 수 있다. 따라서 최적의 G는 도메인 X를 Y와 동일하게 분산된 $\hat{Y}$로 변환한다. 즉 CycleGAN의 목표는 $G(x)$의 이미지분포가 적대적 손실(Adversarial loss)을 이용해 분포 Y와 구별할 수 없도록 매핑 $G : X \rightarrow Y$를 학습하는 것이다. 이 매핑은 제약이 적기 때문에 역매핑 $F : Y \rightarrow X$과 함께 진행했고, $F(G(X)) \approx X$을 시행하기 위한 주기 일관성 손실(Cycle Consistency loss)을 도입한다.

5. 생성된 모형의 실생활 적용 방안

GAN은 현재 많은 분야에 사용되고 있는 기술이다. 예를 들면 낡은 옛날 사진을 복원해 준다든지, 흑백 사진에 채색해준다든지, 심지어는 눈을 감고 있는 사진을 눈을 뜬 것처럼 바꿀 수도 있다. 또한 현실에 없는 새로운 데이터를 생성하거나 새로운 형태로 데이터를 변환, 데이터 품질을 향상하는 등 새로운 기회 가능성을 제시한다. 최근에는 음성 등 자연어 처리에도 많이 이용되고 있다. 다음의 예시들은 구체적인 GAN의 실생활 적용 방안 사례들이다.

5-1. 의료데이터에서의 데이터 부족 해결

합성 데이터란 “직접 확보하지 못하는 특정 환경에 적용할 수 있는 모든 데이터”를 말한다. 합성 데이터는 ‘존재하지 않으나 그럴듯한 가짜 데이터’로 진료정보 및 유전체, 라이프로그 등 실제 의료데이터를 기반으로 만들어진 것이 합성 의료 데이터이다. 이것의 생성기술의 목적은 ‘데이터의 개인정보 및 기밀성 보호’와 ‘데이터의 양적, 질적 고도화’로 구분된다. 이를 통해 실제와 유사한 가짜 의료데이터를 생성함으로써 의료데이터의 민감 정보 식별 문제를 해결하는 것이 가능하다.

합성 의료 데이터의 장점은 크게 세 가지가 있다. 첫 번째, 소량의 원(기존) 데이터로 필요한 만큼의 데이터를 빠르고 저렴하게 생성할 수 있다. 두 번째로는 데이터 라벨링(labeling) 작업을 위한 시간과 비용 절감 및 정확한 라벨링 된 데이터셋을 확보할 수 있다. 마지막으로 민감

한 정보를 포함하고 있는 실제 의료데이터를 대체할 수 있는 장점이 존재한다. 즉, 인공지능을 개발하는 데 있어서 학습된 데이터가 충분하지 않거나, 데이터 확보 및 수집이 불가능한 경우 GAN 등을 활용한 합성 의료데이터 생성기술이 주목받고 있다.

GAN을 이용한 합성 의료데이터 활용 사례는 다음과 같다.
1) 진행성 전립선암 병변 검출을 위한 MRI 영상 분할
2) 저선량 CT 영상의 노이즈 제거를 통한 일반 선량 CT 영상으로 변환
3) 뇌 MRI를 CT영상으로 변환
4) GAN을 이용한 신약후보 물질 탐색
5) 비장비대 병변 자동검출 및 MRI↔CT 영상 간 이미지 합성
6) PET 이미지 누락 문제 해결을 통한 알츠하이머 진단 AI 모델 성능 개선
7) GAN을 활용한 데이터 부족 및 과 최적화 문제 해결방안 연구
8) GAN을 이용한 고해상도 피부병변 합성
9) 전자건강기록(EHR)을 이용한 합성데이터 생성 및 재식별 가능성 평가 연구
10) 합성 의료데이터 임상적 유효성 검증 연구

하지만 GAN을 통해 구분하기 어려운 합성 의료데이터 생성 및 위변조가 될 수 있기 때문에 부작용에 대한 보안 강화 및 법, 제도적 규제 완화 등 다양한 대비책이 필요하다. 데이터 결과를 의료행위의 임상적 근거로 사용하기 때문에 잘못된 의료정보는 예상치 못한 결과로 이어질 수 있고, 딥페이크(Deep fake) 등 가짜 의료데이터를 생성으로 심각한 부작용이 초래하므로 의료 인공지능 기술이 실제로 임상에 활용되고 환자들에 대한 진단 및 치료에 의미 있는 결과를 가져오는지 임상적 유효성을 엄밀하게 검증하는 노력이 필요할 것이다.

5-2. 금융 데이터에서의 이상치 탐지

일반적으로 이상치 탐지(Anomaly detection)라 불리는 이 시스템은 천문학, 의학, 결함 검출과 같은 수많은 분야에 쓰이는 기본적인 기계학습 기법이다. 전통적인 알고리즘은 저차원(low dimension)에 초점이 맞춰 있기 때문에, 이미지 등과 같은 고차원에 적용할 때 어려움을 겪는다. 따라서 GAN을 사용하여 딥러닝에 기반한 이상치 탐지를 이용한다.

금융거래 중 부정하게 사용되는 거래를 부정 거래라고 하는데, 그중에서도 신용카드 부정 거래에 대한 비율은 해마다 증가하고 있다. 현재는 주요 은행 및 보험사에서 FDS(Fraud Detection System)를 사용하는데, 이것은 주로 규칙(Rule)기반으로 사람에 의해 발생하기 때문에 실시간으로 탐지하기가 쉽지 않다고 한다. 그러나 GAN은 정규분포를 가정하는 잠재 변수를 입력 값으로 받아 정상을 모사하는 위조데이터를 생성하고 실제 데이터와 위조 데이터 간의 맨해튼 거리를 이용하여 이상 점수를 도출한다. 즉, 금융 분야에서는 신용카드 거래내역을 실제 데이터로 하고, 새로 생성된 위조데이터간의 맨해튼 거리를 이용한 이상 점수로 이상치를 탐지하고, 거래를 중단시킨다. 따라서 GAN을 이용한 금융 데이터의 이상치 탐지가 점점 발전하고 있다.

5-3. 자율 주행을 위한 가상 데이터 생성

자율 주행 기술이 활발하게 개발됨에 따라 자율주행 기술의 평가도 중요해지고 있다. 매우 다양한 상황에 대한 자율주행 기술 평가는 시간과 비용의 문제로 인해 실제 자동차가 아닌 가상 환경에서 이루어지고 있다. 이때 가상 환경에서 자율주행 기술 평가의 신뢰성을 높이기 위해서는 실제 사람 운전자와 같은 운전자 모델이 자율주행 자동차의 주변 차로써 배치되어야 한다. 실제 다양한 상황에 대처하는 사람을 모사한 운전자 모델은 규칙기반학습(Rule-Based Learning) 기반보다는 인공지능 기반 모델이 더 좋은 연구 결과를 나타내고 있다. 하지만 인공지능 운전자 모델을 학습시키기 위해서는 엄청난 양의 실제 운전자 주행 데이터가 필요하며, 이를 수집하기 위한 시간과 비용은 매우 커서 실제 연구를 진행하기에 문제가 된다. 따라서 비교적 손쉽게 취득 가능한 가상 환경의 주행 데이터가 적은 양의 실제 주행 데이터를 보충하여 인공지능 운전자 모델의 성능을 높여준다면, 적은 비용과 노력으로 높은 성능의 인공지능 운전자 모사 모델을 개발할 수 있을 것이다. 여기서 운전자 모사 모델 개발을 위한 인공지능은 많은 연구에서 모사 모델로 활용되는 GAN을 기반으로 개발되고 있다.

6. 프로그래밍 코드 및 결과

6-1. 구현 코드

```
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
```

다운샘플링하는 동안 사용되는 층(conv2d)과 업샘플링하는 동안 사용되는 층(deconv2d) 정의

```
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)
```

생성모델 build_generator 정의

```
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)
```

분류모델 build_discriminator 정의

```
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)
```

훈련 과정 정의

6-2. 구현 결과

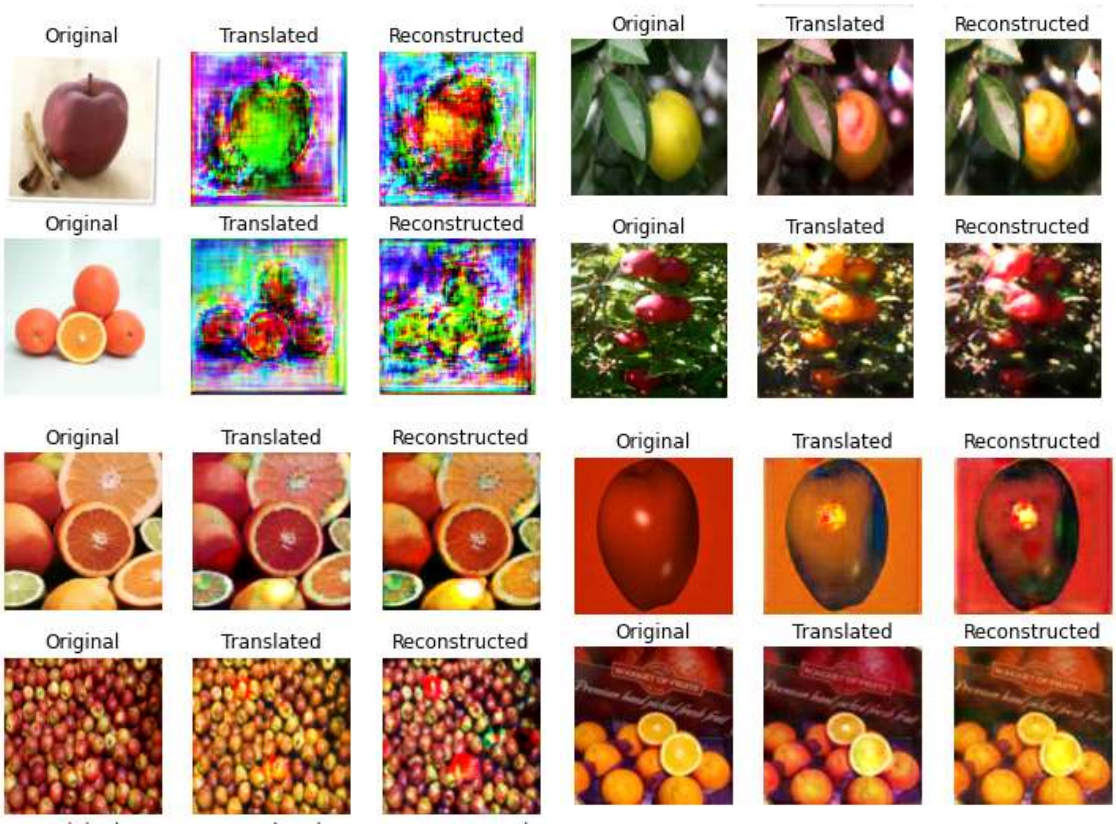

그림. CycleGAN 학습과정

왼쪽 순서대로 원본 이미지(Original), 변환 이미지(Translated), 재구성된 이미지(Reconstructed)이다. 학습이 진행되면서 이미지 변환 성능이 점점 좋아지는 것을 확인할 수 있다.

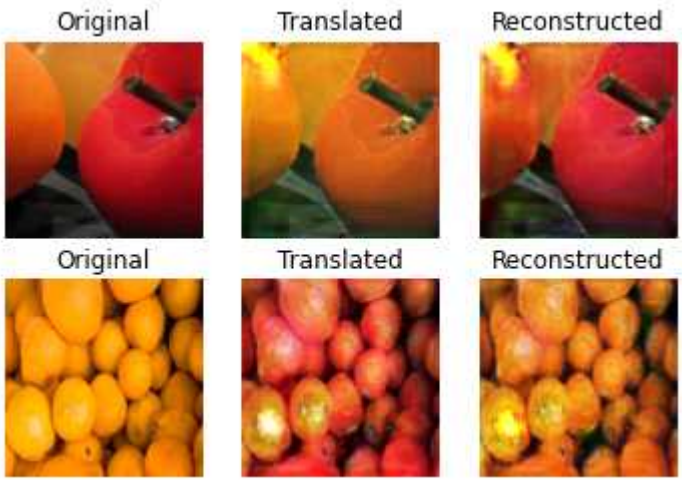

그림. CycleGAN 결과