머신러닝응용 제15강

Deep Learning 2.

첨단공학부 김동하교수

제15강 Deep Learning 2.

1	심층생성모형의 개념에 대해 학습한다.
2	GAN 방법론에 대해 학습한다.
3	Python에서 GAN 방법론을 구현한다.

핵심 단어

- 심층생성모형
- GAN
- Tensorflow 모듈

15강. Deep Learning 2.

01. 심층비지도학습

1) 비지도 학습

- 사람 없이 컴퓨터가 스스로 레이블이 없는 데이터에 대해 학습하는 것.
- y값 없이 x값만을 이용하여 학습하는 것을 의미.
- 데이터가 어떤 특징을 가지고 있는지를 알아내는 것이 목표.

1) 비지도 학습

- 데이터 특징 요약, 밀도 추정, 군집 분석, 독립성 분석 등.

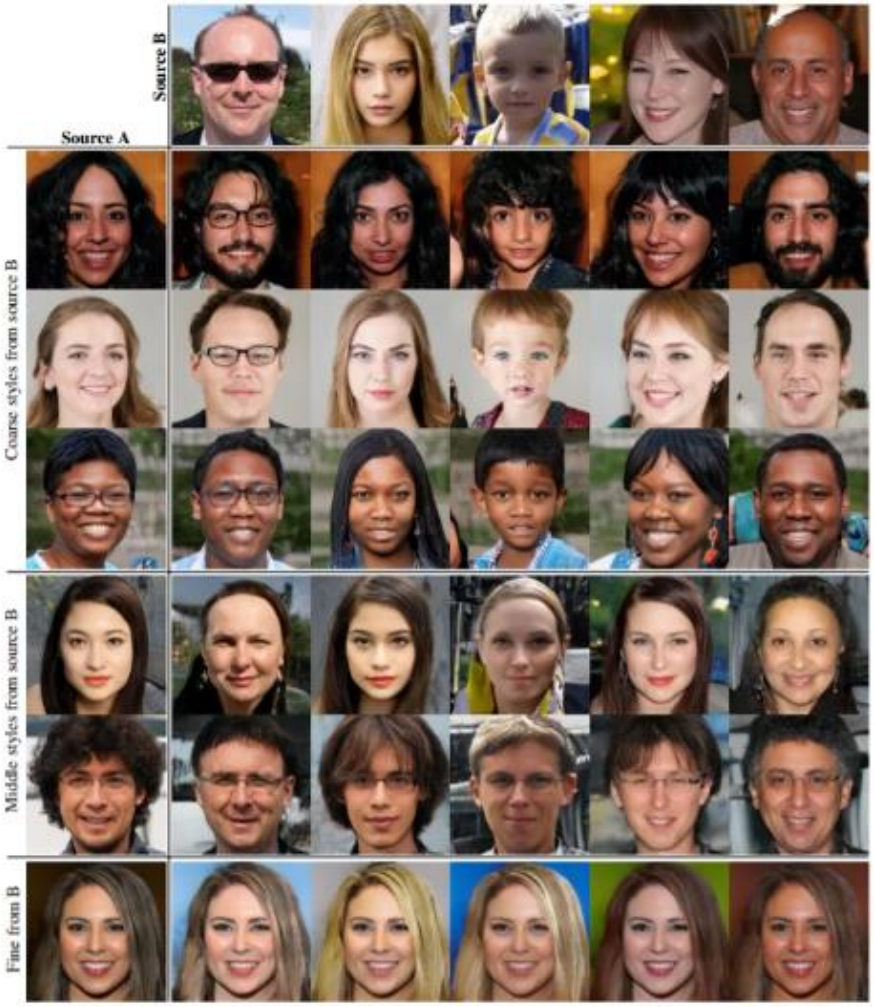
- 본 강의에서는 실제같은 데이터를 생성하는 심층 생성 모형 (deep generative model) 에 대해서 다룰 것.

2) 심층 비지도 학습

- 딥러닝 (deep learning)의 일종.
- 심층 신경망 모형에 기초한 비지도 학습 방법론을 총칭.

3) 심층 비지도학습의 응용분야

◆ 데이터 생성



출처: https://cv-tricks.com/how-to/understanding-stylegan-for-image-generation-using-deep-learning/

3) 심층 비지도학습의 응용분야

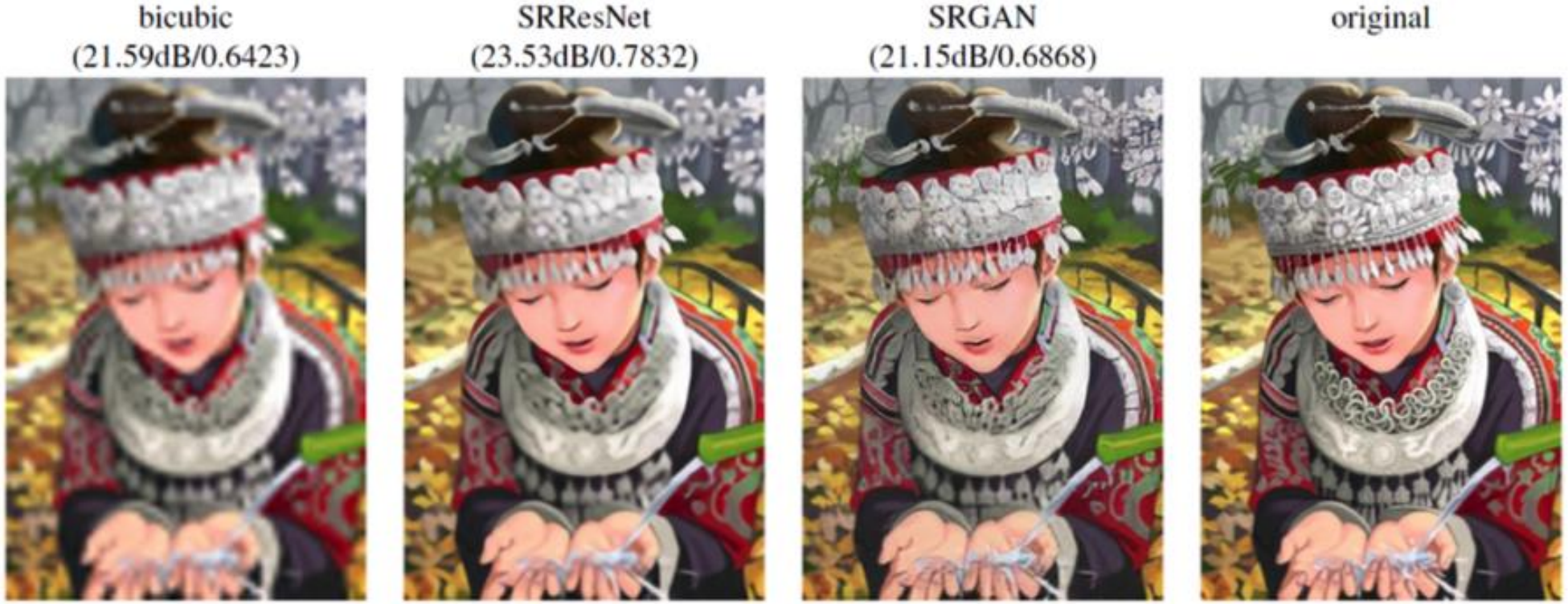
◆데이터 복원

original enhanced

출처:
https://en.wikipedia.org/wiki/Inpainting

3) 심층 비지도학습의 응용분야

◆데이터 화질 개선



출처: https://medium.com/ai%C2%B3-theory-practice-business/super-resolution-using-deep-learning-2cc391f92a4d

15강. Deep Learning 2.

02. 심층생성모형

1) 심층생성모형이란?

- 주어진 자료: $x_1, \ldots, x_n$

- 심층인공신경망 기반의 심층모형을 기반.
  - 이미지 자료, 오디오 자료 등 고차원 자료 분석에 많이 사용

- 실제 자료와 비슷한 자료를 생성하는 모형을 학습.

1) 심층생성모형이란?

- Generative model에는 크게 두 가지의 모형이 존재함.

- 잠재 변수를 가지고 있지 않은 모형
  - Autoregressive generative models

- 잠재 변수를 가지고 있는 모형
  - Latent variable generative models

2) Deep latent generative model

- 저차원 잠재 변수의 변환으로부터 자료가 생성된다고 가정.

  - 변환 함수: 심층신경망모형

2) Deep latent generative model

생성 모형 수식

$$Z \sim N(0, I)$$
$$X|(Z = z) = G(z; \theta)$$

- $Z \in \mathbb{R}^d, X \in \mathbb{R}^D$,
  - 보통 $d \ll D$를 가정

- $G(z; \theta)$: 심층신경망모형

2) Deep latent generative model

◆ 대표적인 추정 방법

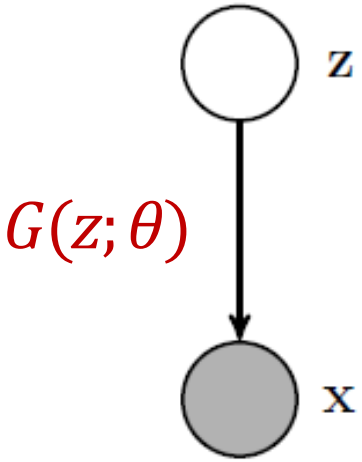
- Generative Adversarial Networks (GANs)

15강. Deep Learning 2.

03. Generative Adversarial Networks

1) Generative adversarial networks

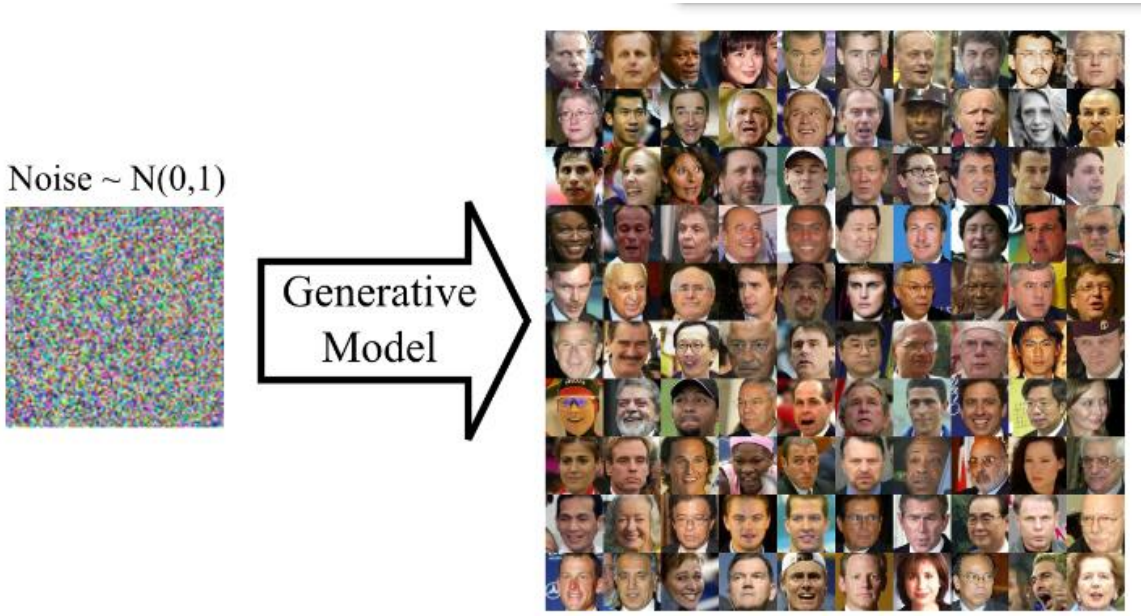
- GANs (Goodfellow et al. (2014))
- 대표적인 심층생성모형
- 독특한 추정 방식



$$z \sim \mathcal{N}(0, \sigma^2 I),$$
$$x = G(z;\theta)$$

1) Generative adversarial networks

- Generative
  - 어떤 목적을 가진 방법론인지.
  - 실제 데이터의 분포를 최대한 따라하는 생성 모형을 학습.



출처:
https://medium.com/@myouis/introduction-to-generative-adversarial-networks-a8f7dcb02cab

1) Generative adversarial networks

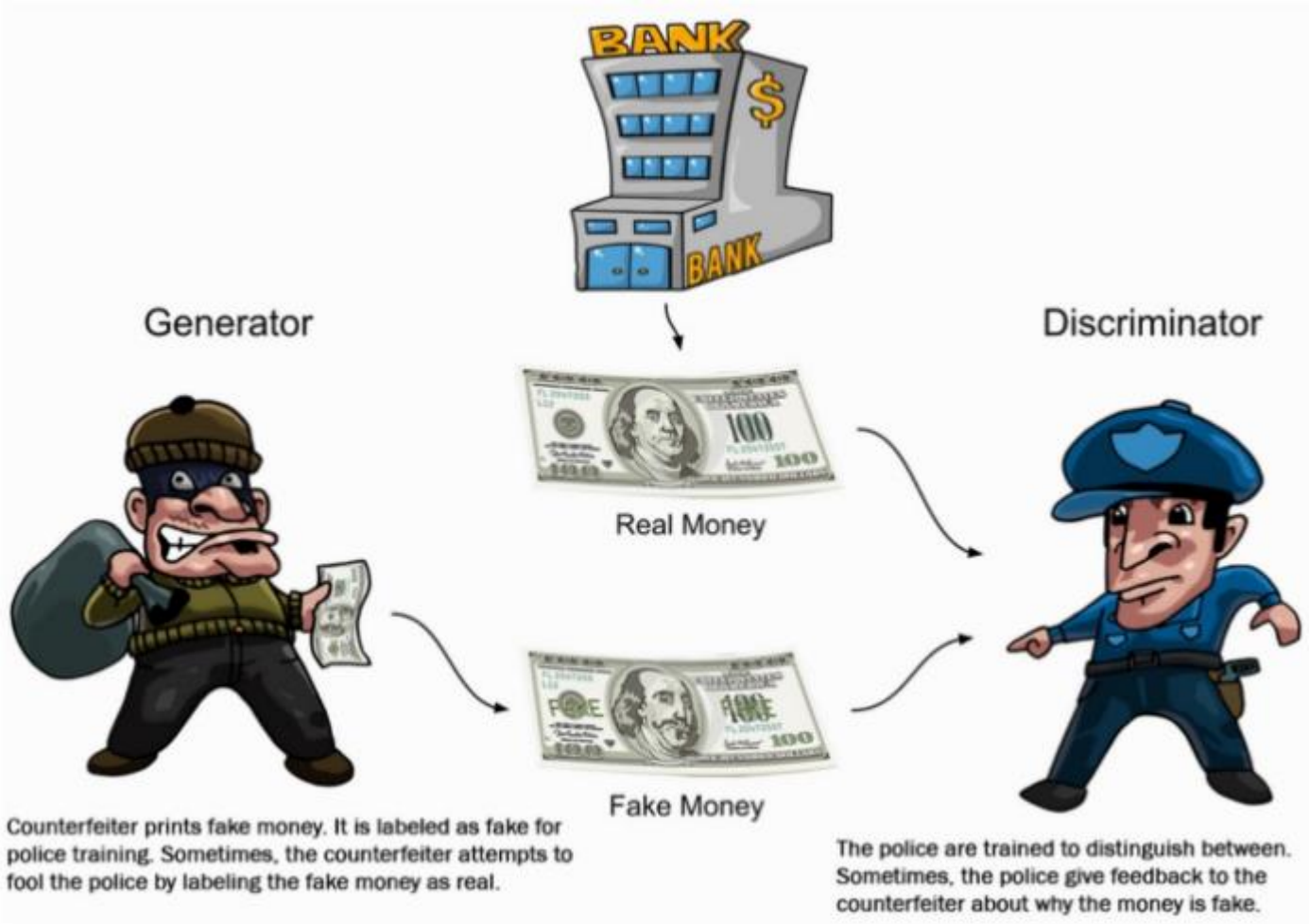
- Adversarial
  - 생성 모형을 어떻게 학습할 것인지.
  - 생성자와 구분자 두 개의 모형을 적대적 (adversarial)으로 경쟁시키며 학습함.

1) Generative adversarial networks

- Adversarial
  - 생성자 : 위조 지폐범, 구분자 : 경찰
    - 생성자의 목적 : 구분자가 구분할 수 없게끔 그럴듯한 가짜 데이터를 생성.
    - 구분자의 목적 : 생성자가 만든 가짜 데이터와 진짜 데이터를 구분.
  - 이 둘을 함께 학습 -> 진짜와 구분할 수 없는 가짜를 만들어내는 생성자를 얻을 수 있음.

1) Generative adversarial networks

Adversarial



출처: https://towardsdatascience.com/the-math-behind-gans-generative-adversarial-networks-3828f3469d9c

1) Generative adversarial networks

Networks

- 어떤 형태의 모형을 사용할 것인지.
- 심층 인공 신경망 함수를 사용.

2) GAN을 구성하는 두 모형

- 생성 모형 (Generator)
  - 랜덤 노이즈에서 자료를 생성하는 심층 신경망 모형.
  - Z : 랜덤 노이즈. 균일 분포 혹은 표준 정규 분포를 사용.
  - $G(z;\theta)$: 표본 z 를 이용하여 자료를 생성하는 심층 신경망 모형.
    - θ 는 함수에 필요한 모수.

2) GAN을 구성하는 두 모형

- **판별 모형 (Discriminator)**
  - 자료가 입력값으로 들어올 때, 주어진 자료가 학습자료에서 얻어진 것인지, 혹은 생성 모형을 이용해 생성되었는지를 판별하는 심층 신경망 모형.

2) GAN을 구성하는 두 모형

- 판별 모형 (Discriminator)
  - $D(x;\eta)$: 주어진 x가 실제 자료인지 인공 자료인지를 판단하는 심층 신경망 모형.
    - 0과 1 사이의 값을 가짐
    - η 는 함수에 필요한 모수
    - $D(x;\eta) \geq 0.5$: 실제 자료로 판단
    - $D(x;\eta) < 0.5$: 생성된 인공 자료로 판단

3) GAN의 학습

- Two-player minimax game
  - 두 함수가 서로 경쟁하는 게임의 형태로 학습.

- 생성 함수 G는 판별 함수 D가 학습 자료와 생성 자료를 구분하지 못하도록 θ를 학습.

- 판별 함수 D는 학습 자료와 생성 자료를 잘 구분하도록 η를 학습.

3) GAN의 학습

◆목적 함수

$$\min_{\theta}\max_{\eta}\mathbb{E}_{x\sim p_{data}}[\log D(x;\eta)]+\mathbb{E}_{z\sim p(z)}[\log(1-D(G(z;\theta);\eta))]$$

◆주어진 목적 함수를 η에 대해서 최대화하고, θ에 대해서 최소화하는 작업을 번갈아 진행.

3) GAN의 학습

- 주어진 생성모형에 대해서 구분모형은 실제 데이터와 만들어진 데이터의 분류를 최대한 잘 하는 방향으로 학습.

- 주어진 구분모형에 대해서 생성모형은 구분모형이 최대한 헷갈리도록 실제 데이터와 비슷한 데이터를 생성하는 방향으로 학습.

4) GAN의 장단점

장점

- 다른 심층생성모형 기법에 비해 학습이 빠르고 생성 과정도 훨씬 빠르다.
- 뚜렷한 이미지를 생성할 수 있다.

4) GAN의 장단점

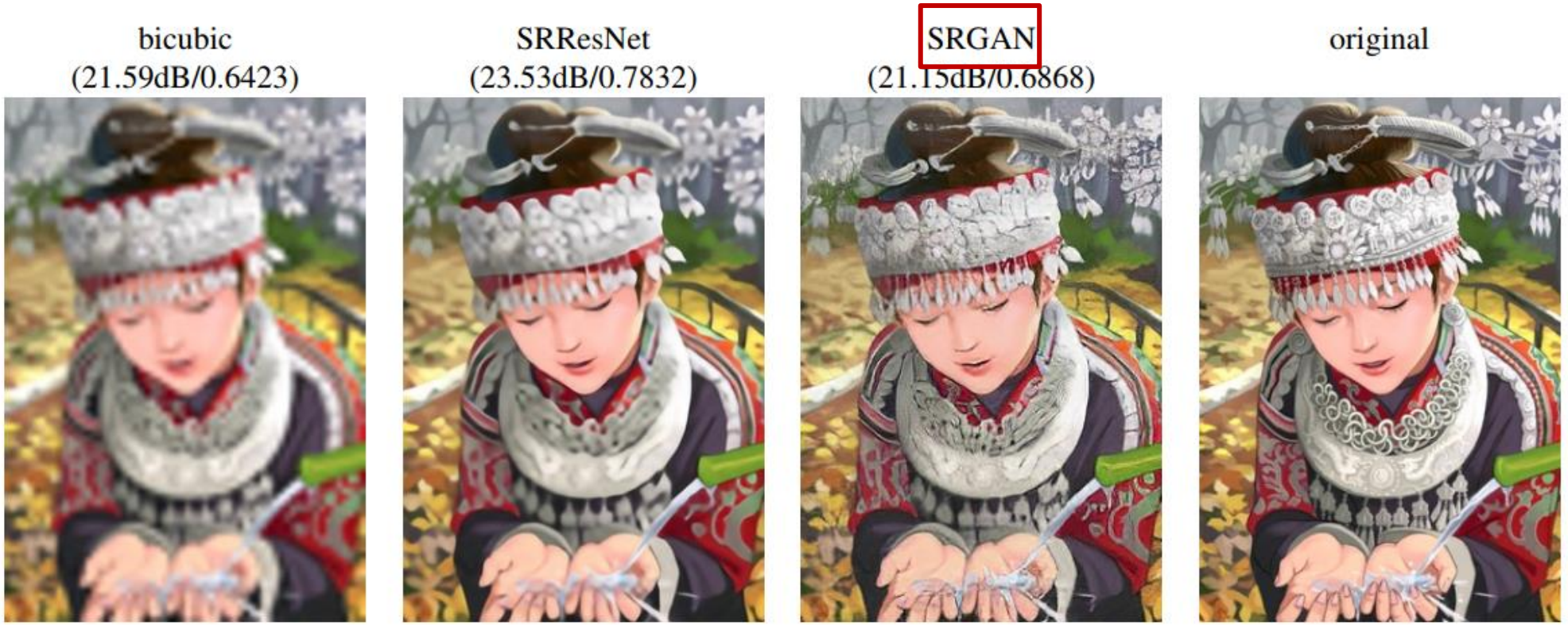
단점

- 학습이 불안정하다. (min, max 를 동시에 하기 때문)

- Mode collapsing 이 생긴다.
  - Ex: Fashion MNIST 로 학습할 경우, 10가지 중 몇 가지 의류 이미지만 생성되기 쉽다.

5) GAN의 응용

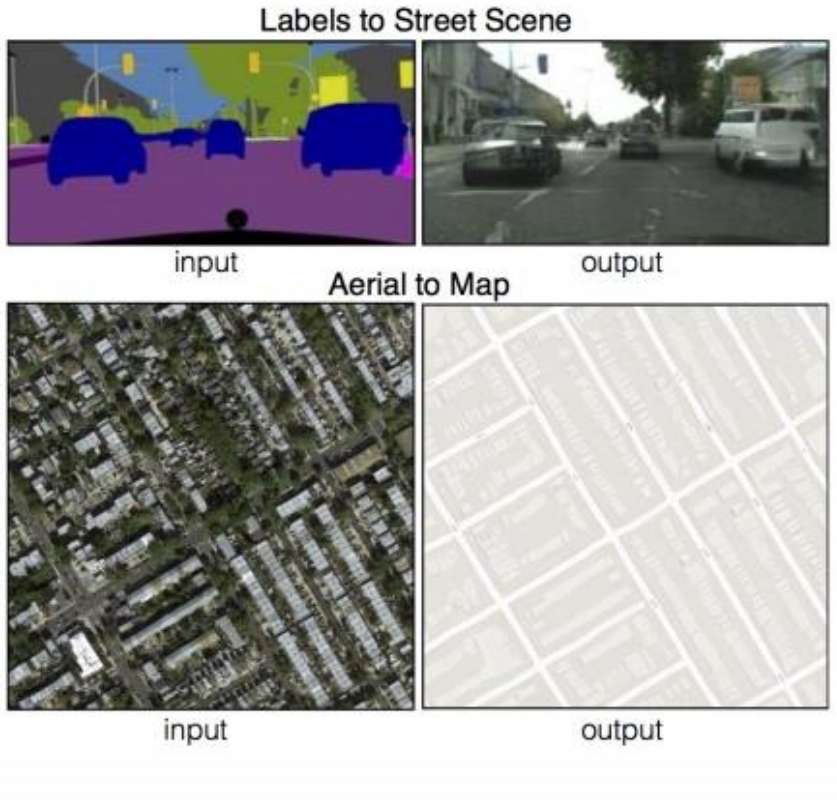
◆ 이미지 해상도 복원 (SRGAN; Ledig et al., 2016)



출처: Ledig et al., 2016

5) GAN의 응용

◆ 이미지 변환 (pix2pix; Isola et al., 2016)



출처: Isola et al., 2016

15강. Deep Learning 2.

04. Python을 이용한 실습

1) 데이터 설명

- MNIST
  - 숫자 손글씨 이미지 데이터
  - 훈련 자료: 60,000개, 시험 자료: 10,000개
  - 각 이미지는 10가지의 숫자 중 하나가 쓰여져 있음.

- MNIST 자료를 이용하여 숫자 그림을 생성하는 GAN 모형을 학습해보자.

2) 환경설정

◆ 필요한 패키지 불러오기

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras import layers

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import time
```

3) 데이터 불러오기

◆데이터 불러오기

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

3) 데이터 불러오기

◆데이터 확인하기

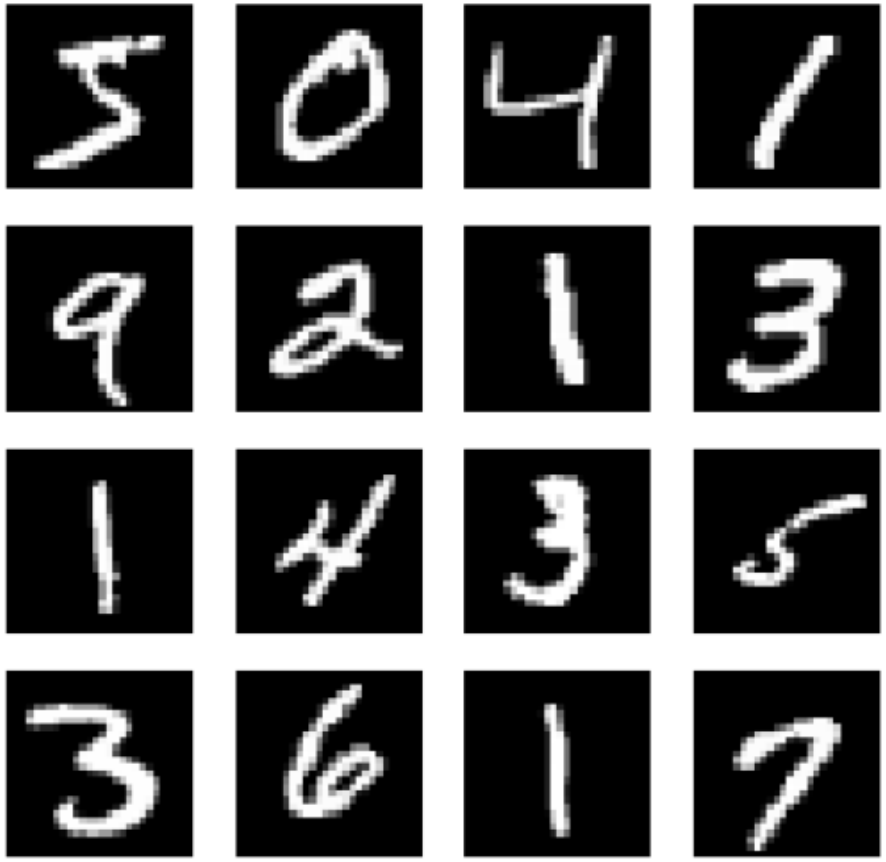
```
# train 이미지 확인
plt.figure(figsize=(8, 8))
for i in range(16):
    plt.subplot(4, 4, i+1)
    plt.suptitle('Train Images', fontsize=20)
    plt.imshow(x_train[i], cmap=plt.cm.gray)
    plt.axis("off")

plt.show()
```

3) 데이터 불러오기

◆데이터 확인하기

Train Images



4) 데이터 전처리

- 입력값을 32 비트 형태로 변환
- 픽셀값을 -1과 1 사이로 재조정

```
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1).astype('float32')
x_train = (x_train - 127.5) / 127.5

BUFFER_SIZE = 60000
BATCH_SIZE = 256
noise_dim = 100

train_dataset = tf.data.Dataset.from_tensor_slices(x_train).shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
```

5) 심층인공신경망 생성

◆ 생성모형 구축

```
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_bias=False, activation='tanh'))

    return model

generator = make_generator_model()
```

5) 심층인공신경망 생성

◆ 구분모형 구축

```
def make_discriminator_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                     input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model

discriminator = make_discriminator_model()
```

6) 모형의 학습

◆ 구분모형 손실함수 정의

```
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss
```

6) 모형의 학습

◆ 생성모형 손실함수 정의

```
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)
```

6) 모형의 학습

◆ 최적화 알고리즘 정의

```
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
```

6) 모형의 학습

◆ 손실함수를 이용한 학습

```
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)

        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)

        gen_loss = generator_loss(fake_output)
        disc_loss = discriminator_loss(real_output, fake_output)

    gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss, discriminator.trainable_variables)

    generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator, discriminator.trainable_variables))
```

6) 모형의 학습

◆ 손실함수를 이용한 학습

```
def train(dataset, epochs):
    for epoch in range(epochs):
        start = time.time()
        for image_batch in dataset:
            train_step(image_batch)
        end = time.time()
        print ('Time for epoch {} is {} sec'.format(epoch + 1, end-start))
```

6) 모형의 학습

◆ 손실함수를 이용한 학습

```
EPOCHS = 30
train(train_dataset, EPOCHS)
```

```
Time for epoch 1 is 35.12040710449219 sec
Time for epoch 2 is 25.38134765625 sec
Time for epoch 3 is 23.504241466522217 sec
Time for epoch 4 is 23.92603087425232 sec
Time for epoch 5 is 23.927464246749878 sec
Time for epoch 6 is 23.741941690444946 sec
Time for epoch 7 is 24.04710602760315 sec
Time for epoch 8 is 23.742584705352783 sec
Time for epoch 9 is 23.755187034606934 sec
Time for epoch 10 is 23.7379891872406 sec
Time for epoch 11 is 24.07156014442444 sec
Time for epoch 12 is 24.076233386993408 sec
Time for epoch 13 is 23.8432941436767 58 sec
```

7) 이미지 생성하기

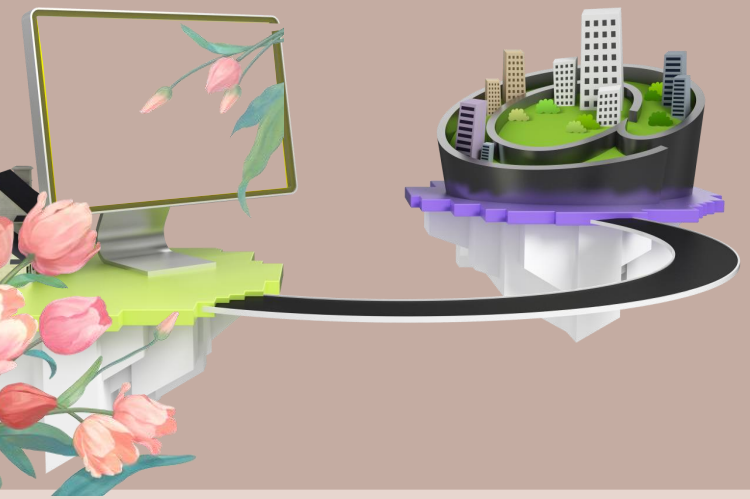
◆ 16개의 이미지 생성하기

```
num_examples_to_generate = 16
seed = tf.random.normal([num_examples_to_generate, noise_dim])
gen_images = generator(seed, training=False)

fig = plt.figure(figsize=(4,4))
for i in range(gen_images.shape[0]):
    plt.subplot(4, 4, i+1)
    plt.imshow(gen_images[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
    plt.axis('off')
```

7) 이미지 생성하기

이미지 생성 결과



그동안
수고하셨습니다.

끝.