GAN

유튜브: https://www.youtube.com/watch?v=sVjHNIJlm-o

GAN 개념과 원리

GAN 구조

GAN 종류

GAN 구현

GAN(Generative Adversarial Network)

GAN: 생성적 적대 신경망

Generative: 생성모델
Adversarial: 두 모델을 적대적으로 경쟁
Network:인공신경망

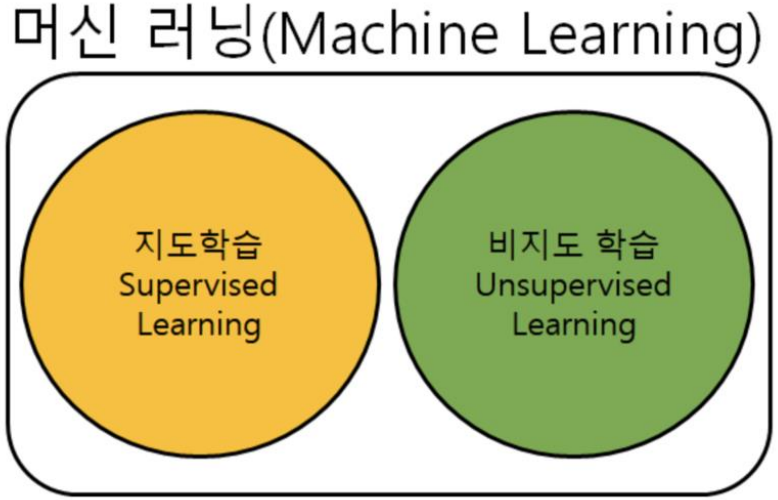
딥페이크로 만든 영상 및 사진

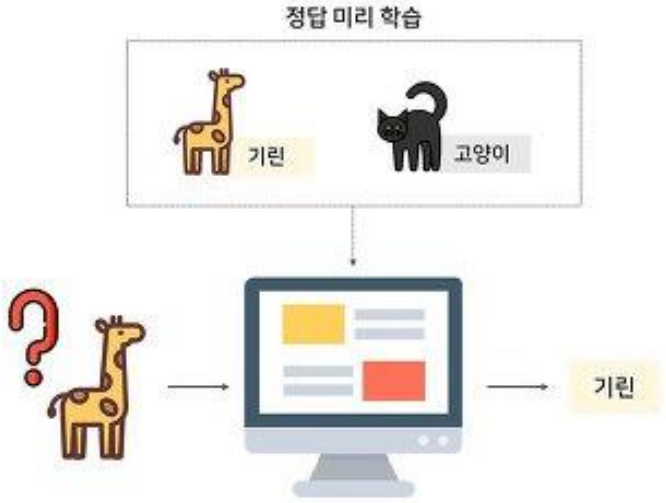
→ GAN을 이용

GAN 개념과 원리

지도학습(Supervised Machine Learning):
- 정답을 전달하며 학습시킴
- Input과 Output 확실

비지도 학습(Unsupervised Machine Learning):
- 정답이 주어지지 않은 상태에서 학습하는 알고리즘
- 데이터의 특성을 학습하여 스스로 패턴 확인

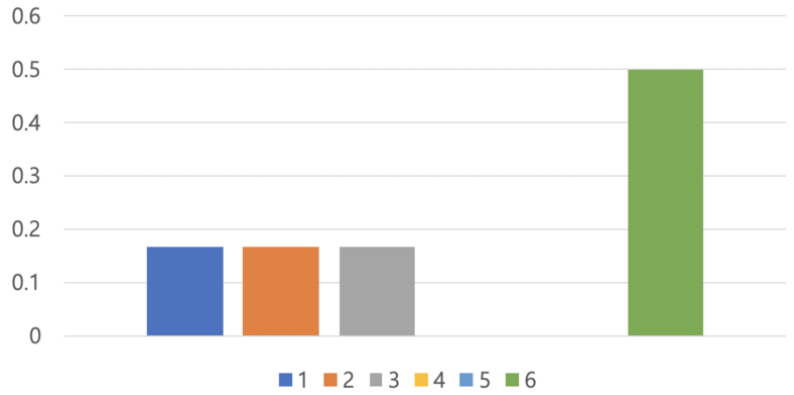


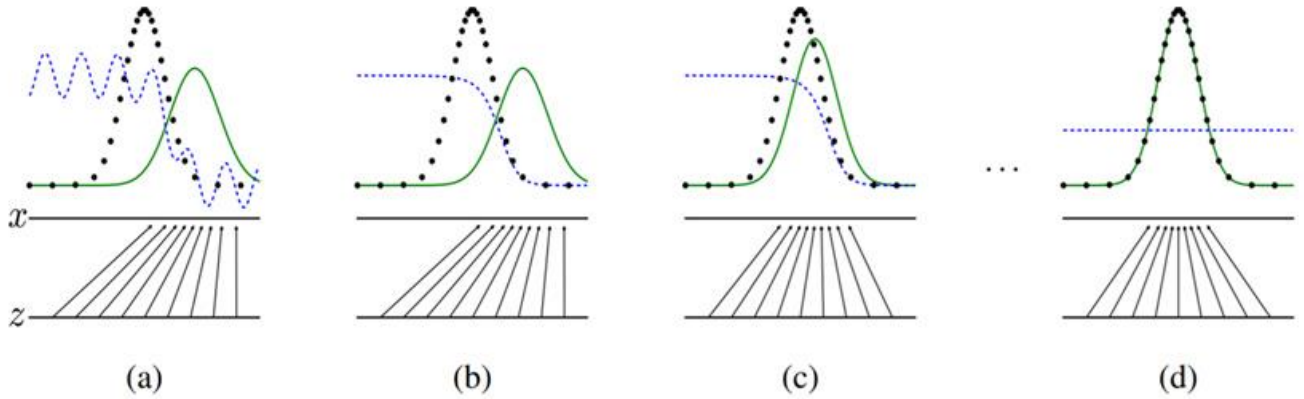


GAN 개념과 원리

- 확률분포: 확률 변수가 특정한 값을 가질 확률

X	1	2	3	4	5	6
P(X)	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{0}{6}$	$\frac{0}{6}$	$\frac{5}{6}$





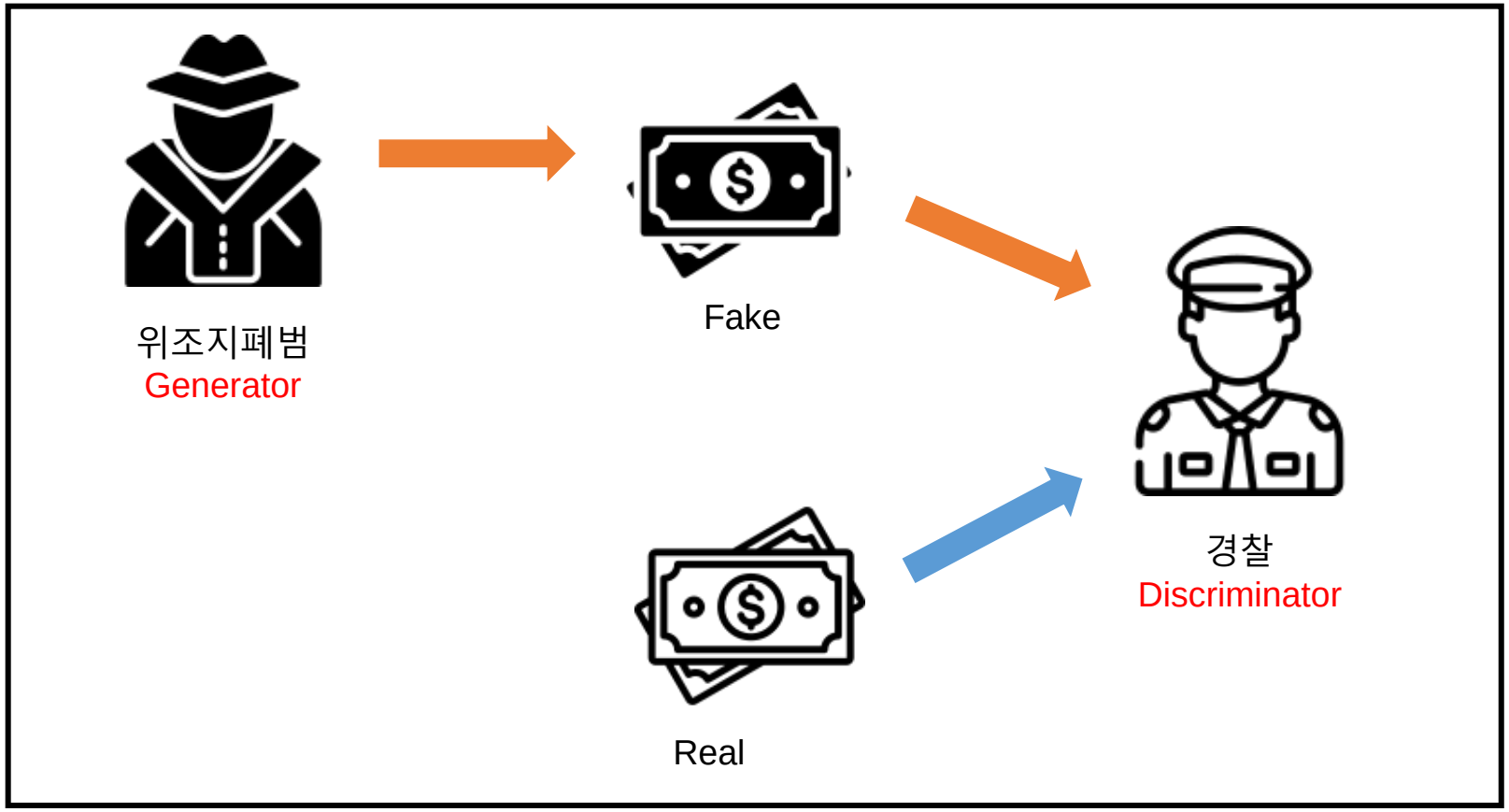
※ 검은 점선: 원 데이터의 확률분포, 녹색 점선: GAN이 만들어 내는 확률분포, 파란 점선: 분류자의 확률분포

GAN 개념과 원리

경찰과 위조지폐범 게임에 기반한 원리

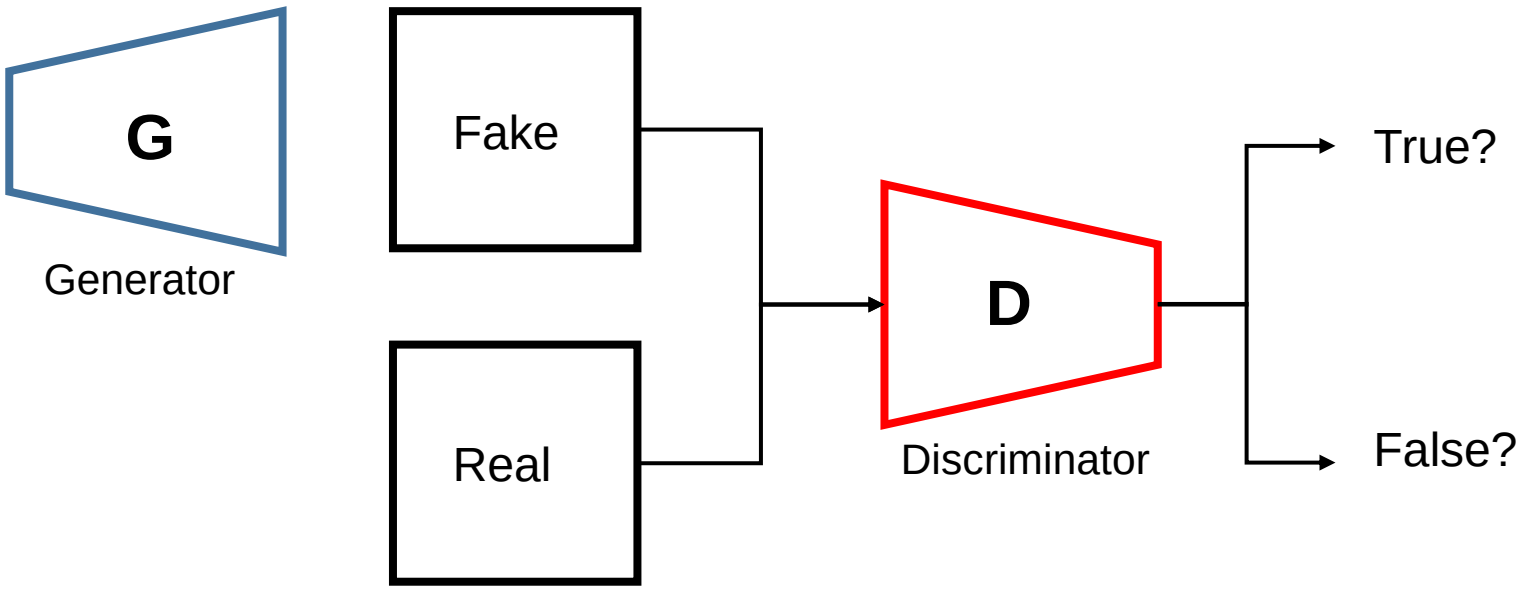
위조지폐범: 진짜 같은 화폐 생성

경찰: 진짜와 가짜 완벽하게 구분



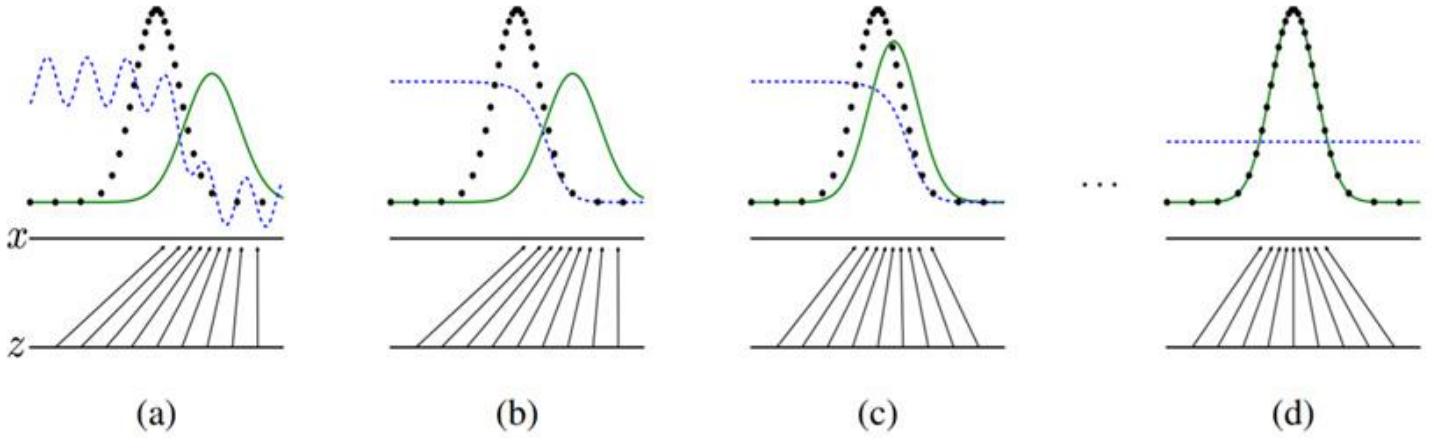
GAN 구조

- 생성 모델(Generator)
  Discriminator를 속이기 위한 가짜 이미지 생성
- 분류 모델(Discriminator)
  주어진 이미지가 진짜인지 가짜인지 판별



GAN 구조

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

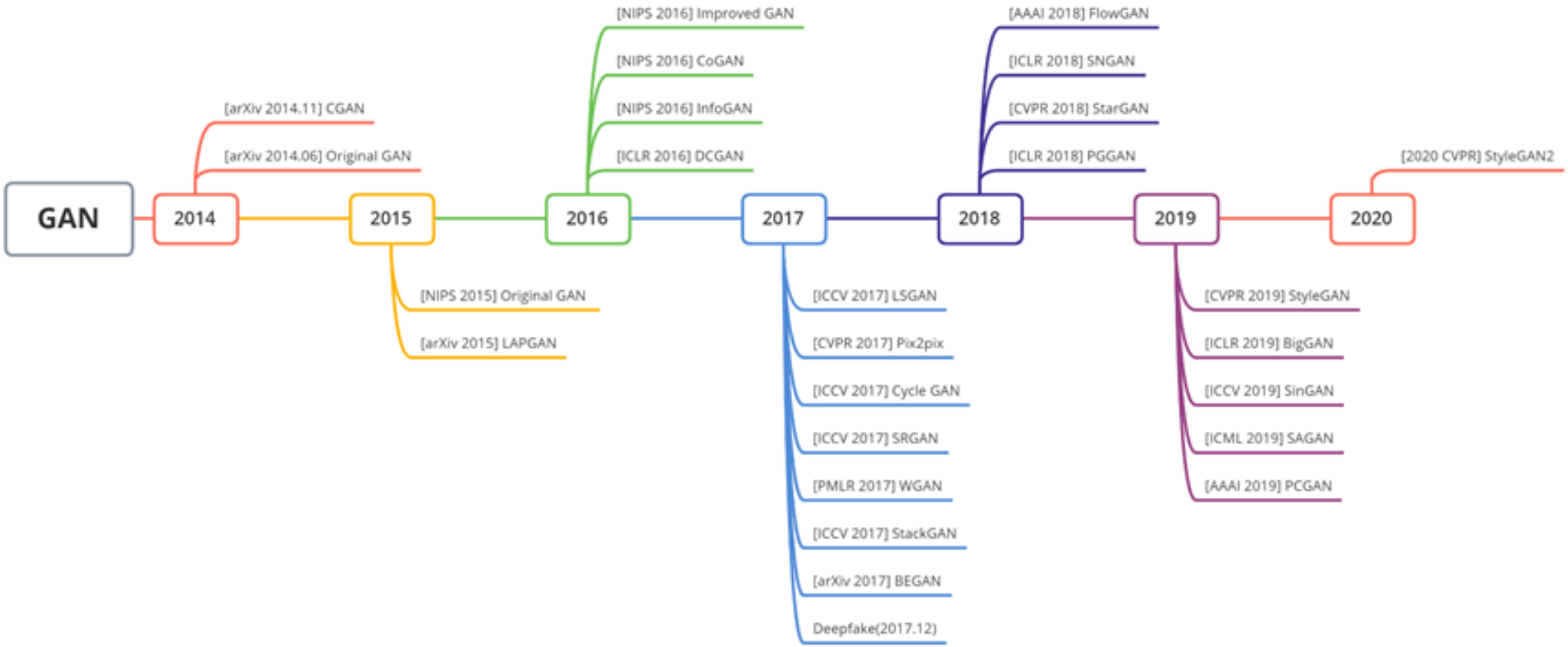


※ 검은 점선: 원 데이터의 확률분포, 녹색 점선: GAN이 만들어 내는 확률분포, 파란 점선: 분류자의 확률분포

D: V가 최대가 되도록.
-> D가 원하는 최적의 상황은 전부 0
G:V가 최소가 되도록
G가 원하는 최적의 상황은 음의 무한대가 되는 방향

D:분류모델,
가짜 데이터=0 진짜 데이터=1
G:생성 모델
D(x)=1 , D(G(z))= 0
X: 실제 데이터
Z: 가짜 데이터

GAN 종류

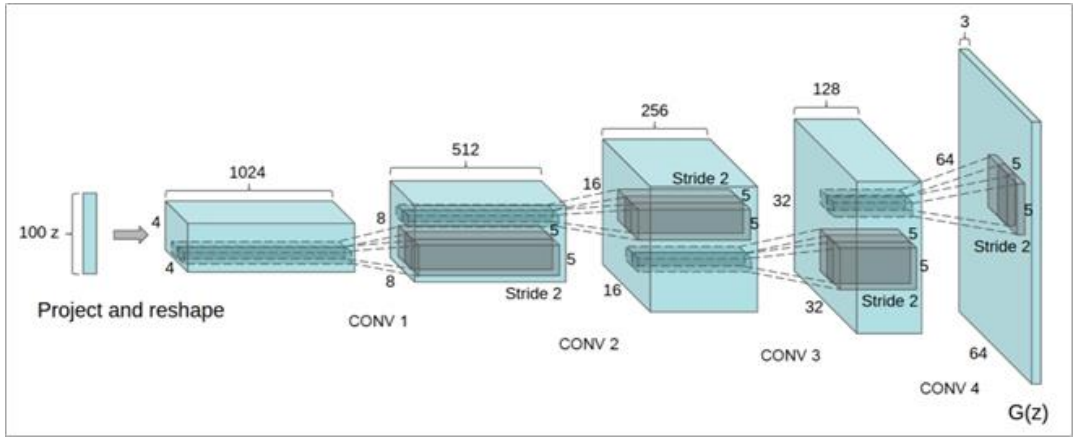

GAN
2014
[arXiv 2014.11] CGAN
[arXiv 2014.06] Original GAN
2015
[NIPS 2015] Original GAN
[arXiv 2015] LAPGAN
2016
[NIPS 2016] Improved GAN
[NIPS 2016] CoGAN
[NIPS 2016] InfoGAN
[ICLR 2016] DCGAN
2017
[ICCV 2017] LSGAN
[CVPR 2017] Pix2pix
[ICCV 2017] Cycle GAN
[ICCV 2017] SRGAN
[PMLR 2017] WGAN
[ICCV 2017] StackGAN
[arXiv 2017] BEGAN
Deepfake(2017.12)
2018
[AAAI 2018] FlowGAN
[ICLR 2018] SNGAN
[CVPR 2018] StarGAN
[ICLR 2018] PGGAN
2019
[CVPR 2019] StyleGAN
[ICLR 2019] BigGAN
[ICCV 2019] SinGAN
[ICML 2019] SAGAN
[AAAI 2019] PCGAN
2020
[2020 CVPR] StyleGAN2

GAN 종류

- DCGAN

facebook 에서 연구.CNN 사용

fully-connected layer 대신 Convolutional layer사용

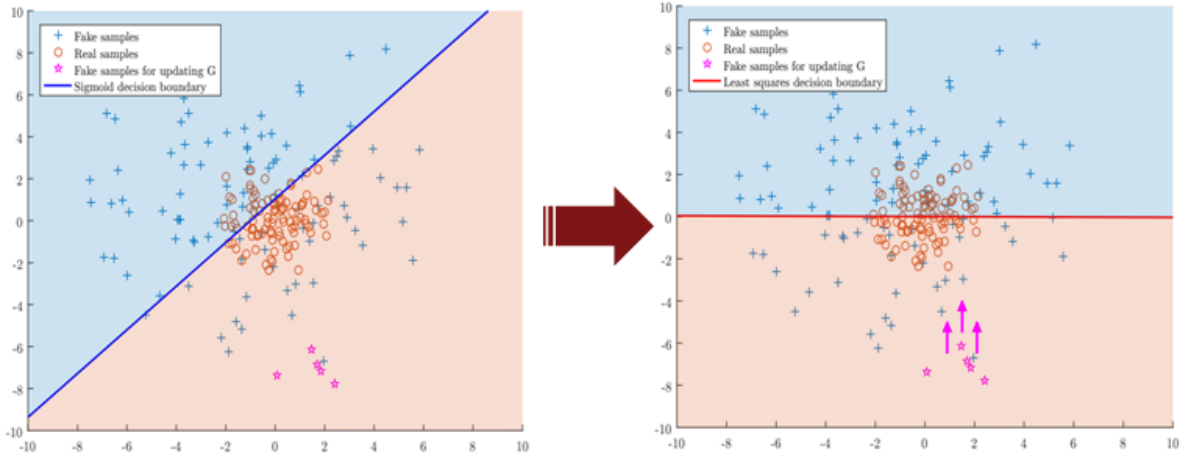


- LSGAN

판별자를 속였기 때문에 더 이상 학습 x(gradient vanishing)

더 정교하게 속이기 위해 real 수준으로 끌어 올림.

Sigmoid cross entropy loss -> Least Square loss

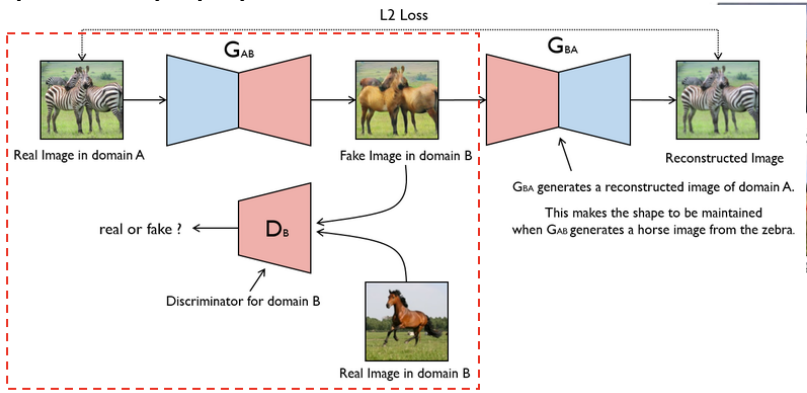


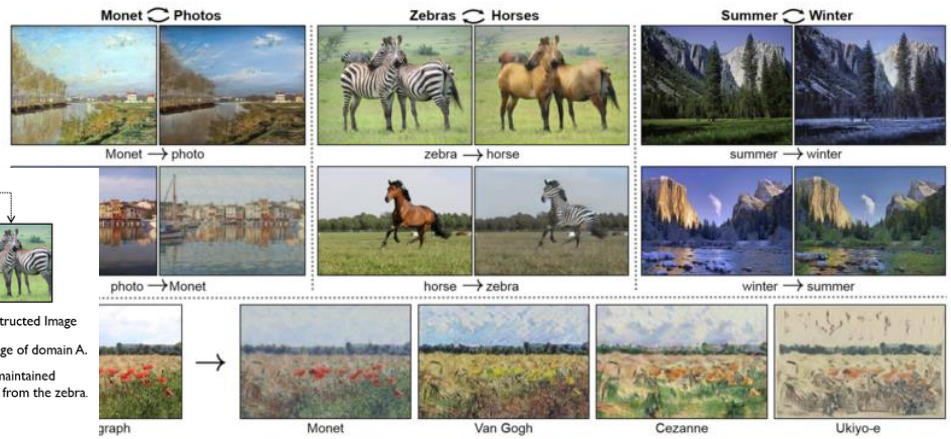
GAN 종류

- Cycle GAN

두개의 unpaired한 이미지를 가지고 새로운 이미지 생성

색상, 질감 만 가능, 모양 x

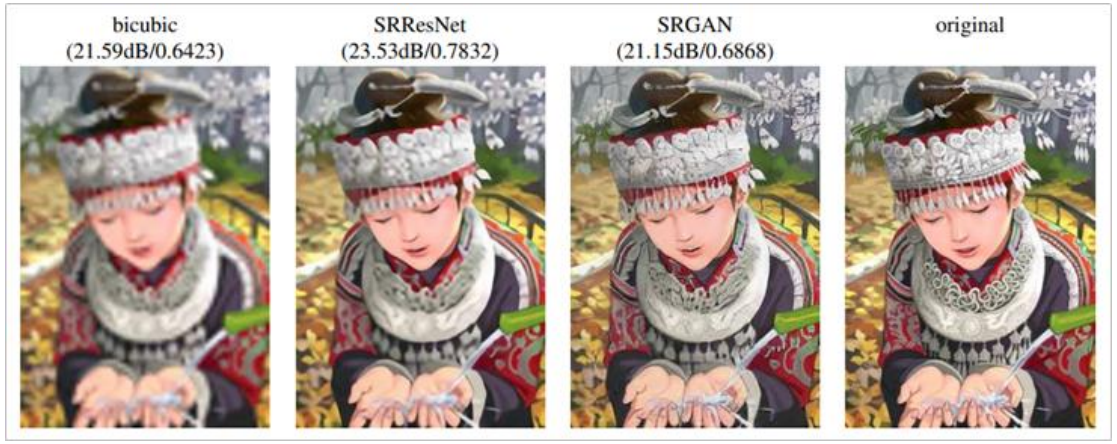




- SRGAN

저해상도 -> 고해상도

기존 = PSNR은 높지만 high frequency detail x



GAN 구현

```
[ ]  from tensorflow.keras.datasets import mnist
```

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

```
Downloading data from https://storage.googleapis.com/tensorflow/
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
x_train = x_train / 127.5 - 1
x_test = x_test / 127.5 - 1
x_train.min(), x_train.max()
```

```
(-1.0, 1.0)
```

```
[ ]  x_train = x_train.reshape(-1, 784)
     x_train.shape
```

```
[ ]  from tensorflow.keras.layers import Dense, LeakyReLU, Dropout, Input
     from tensorflow.keras.models import Sequential, Model
     from tensorflow.keras.optimizers import Adam
     from tensorflow.keras.initializers import RandomNormal
     import numpy as np
     import matplotlib.pyplot as plt
```

```
# gan에 입력되는 noise에 대한 dimension
NOISE_DIM = 10

# adam optimizer 정의, learning_rate = 0.0002, beta_1로
adam = Adam(lr=0.0002, beta_1=0.5)
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argument is deprecated,
  super(Adam, self).__init__(name, **kwargs)
```

```
[ ]  generator = Sequential([
         Dense(256, input_dim=NOISE_DIM),
         LeakyReLU(0.2),
         Dense(512),
         LeakyReLU(0.2),
         Dense(1024),
         LeakyReLU(0.2),
         Dense(28*28, activation='tanh'),
     ])
```

```
[ ]  generator.summary()
```

GAN 구현

```
discriminator = Sequential([
    Dense(1024, input_shape=(784,), kernel_initializer=RandomNormal(stddev=0.02)),
    LeakyReLU(0.2),
    Dropout(0.3),
    Dense(512),
    LeakyReLU(0.2),
    Dropout(0.3),
    Dense(256),
    LeakyReLU(0.2),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

```
[ ] discriminator.summary()
```

```
discriminator.compile(loss='binary_crossentropy', optimizer=adam)
```

```
# discriminator는 학습을 하지 않도록 하며, Gan 모델에서는 generator만 학습
discriminator.trainable = False
gan_input = Input(shape=(NOISE_DIM,))
x = generator(inputs=gan_input)
output = discriminator(x)
```

```
[ ] gan = Model(gan_input, output)
```

```
gan.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 10)]              0

sequential (Sequential)      (None, 784)               1463312

sequential_1 (Sequential)    (None, 1)                 1460225

=================================================================
Total params: 2,923,537
Trainable params: 1,463,312
Non-trainable params: 1,460,225
_________________________________________________________________
```

```
[ ] gan.compile(loss='binary_crossentropy', optimizer=adam)
```

GAN 구현

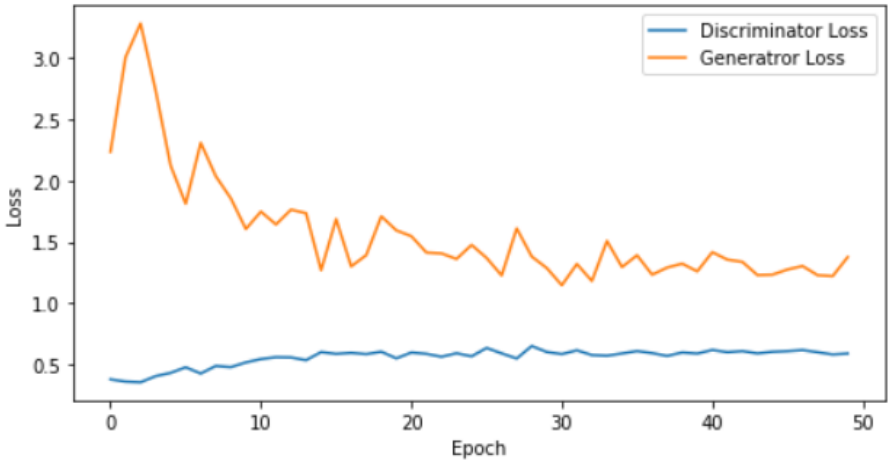
```
[ ]  def get_batches(data, batch_size):
         batches = []
         for i in range(int(data.shape[0] // batch_size)):
             batch = data[i * batch_size: (i + 1) * batch_size]
             batches.append(batch)
         return np.asarray(batches)
```

```
def visualize_training(epoch, d_losses, g_losses):
    # 오차에 대한 시각화
    plt.figure(figsize=(8, 4))
    plt.plot(d_losses, label='Discriminator Loss')
    plt.plot(g_losses, label='Generatror Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    print('epoch: {}, Discriminator Loss: {}, Generator Loss: {}'.format(epoch, np.asarray(d_losses).mean(), np.asarray(g_losses).mean()))

    #샘플 데이터 생성 후 시각화
    noise = np.random.normal(0, 1, size=(24, NOISE_DIM))
    generated_images = generator.predict(noise)
    generated_images = generated_images.reshape(-1, 28, 28)

    plt.figure(figsize=(8, 4))
    for i in range(generated_images.shape[0]):
        plt.subplot(4, 6, i+1)
        plt.imshow(generated_images[i], interpolation='nearest', cmap='gray')
        plt.axis('off')
    plt.tight_layout()
    plt.show()
```

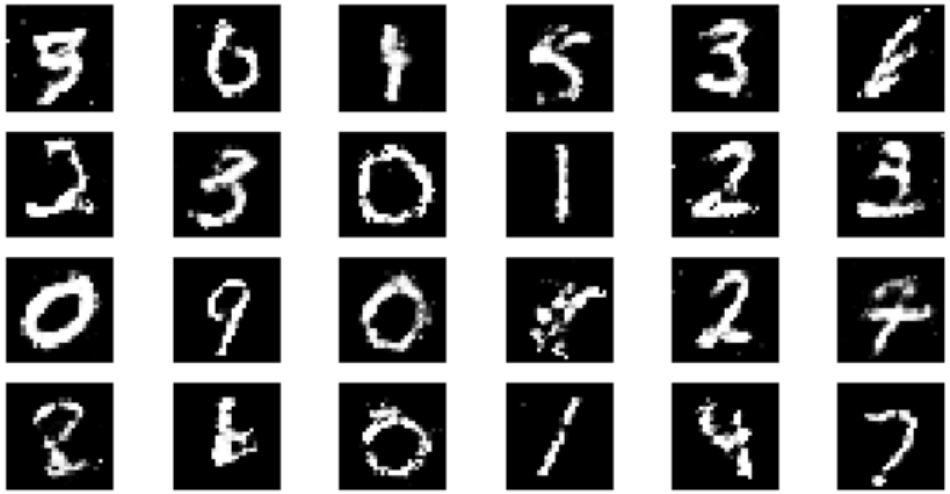
GAN 구현

```
BATCH_SIZE = 128
EPOCHS= 50
d_losses = []
g_losses = []

for epoch in range(1, EPOCHS + 1):
    # 각 배치별 학습
    for real_images in get_batches(x_train, BATCH_SIZE):
        # 랜덤 노이즈 생성
        input_noise = np.random.uniform(-1, 1, size=[BATCH_SIZE, NOISE_DIM])

        # 가짜 이미지 데이터 생성
        generated_images = generator.predict(input_noise)

        # Gan에 학습할 X 데이터 정의
        x_dis = np.concatenate([real_images, generated_images])

        # Gan에 학습할 Y 데이터 정의
        y_dis = np.zeros(2 * BATCH_SIZE)
        y_dis[:BATCH_SIZE] = 0.9

        # Discriminator 훈련
        discriminator.trainable = True
        d_loss = discriminator.train_on_batch(x_dis, y_dis)

        # Gan 훈련
        noise = np.random.uniform(-1, 1, size=[BATCH_SIZE, NOISE_DIM])
        y_gan = np.ones(BATCH_SIZE)

        # Discriminator의 판별 학습을 방지
        discriminator.trainable = False
        g_loss = gan.train_on_batch(noise, y_gan)

    d_losses.append(d_loss)
    g_losses.append(g_loss)

    if epoch == 1 or epoch % 5 == 0:
        visualize_training(epoch, d_losses, g_losses)
```

GAN 구현




```
epoch: 50, Discriminator Loss: 0.5575451850891113, Generator Loss: 1.5767810344696045
```



Q & A