# 적대적 생성 신경망 (GAN)

# 적대적 생성 신경망(GAN)이란

# 13.3 적대적 생성 신경망(GAN)이란

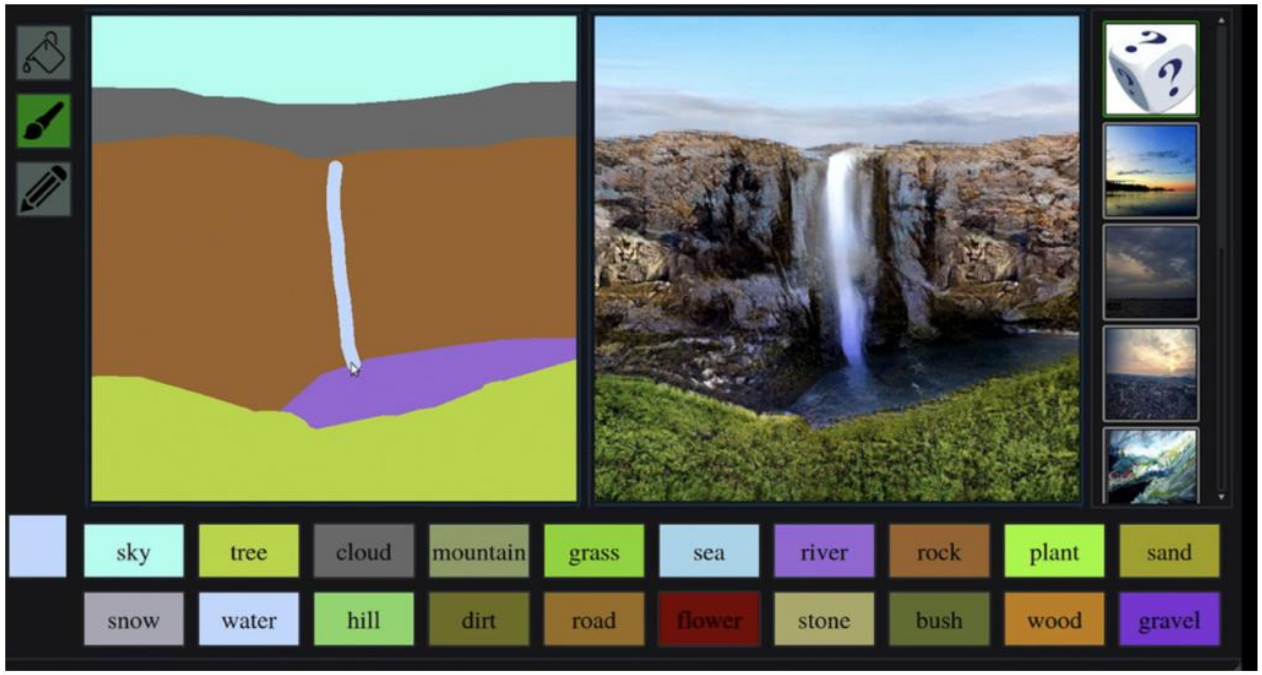
- **적대적 생성 신경망(GAN)이란**
  - 처음 적대적 생성 신경망(Generative Adversarial Network, GAN)을 제안한 이안 굿펠로우(Ian Goodfellow)는 GAN을 경찰과 위조지폐범 사이의 게임에 비유
  - 위조지폐범은 진짜와 같은 위조 화폐를 만들어 경찰을 속이고, 경찰은 진짜 화폐와 위조 화폐를 판별하여 위조지폐범을 검거
  - 위조지폐범과 경찰의 경쟁이 지속되면 어느 순간 위조지폐범은 진짜와 같은 위조지폐를만들 수 있게 되고, 결국 경찰은 위조지폐와 실제 화폐를 구분할 수 없는 상태에 이르게 됨

# 13.3 적대적 생성 신경망(GAN)이란

1) GAN 사진 전환

1) NVIDIA **GauGAN**(3월 27, 2019 by NVIDIA KOREA,)

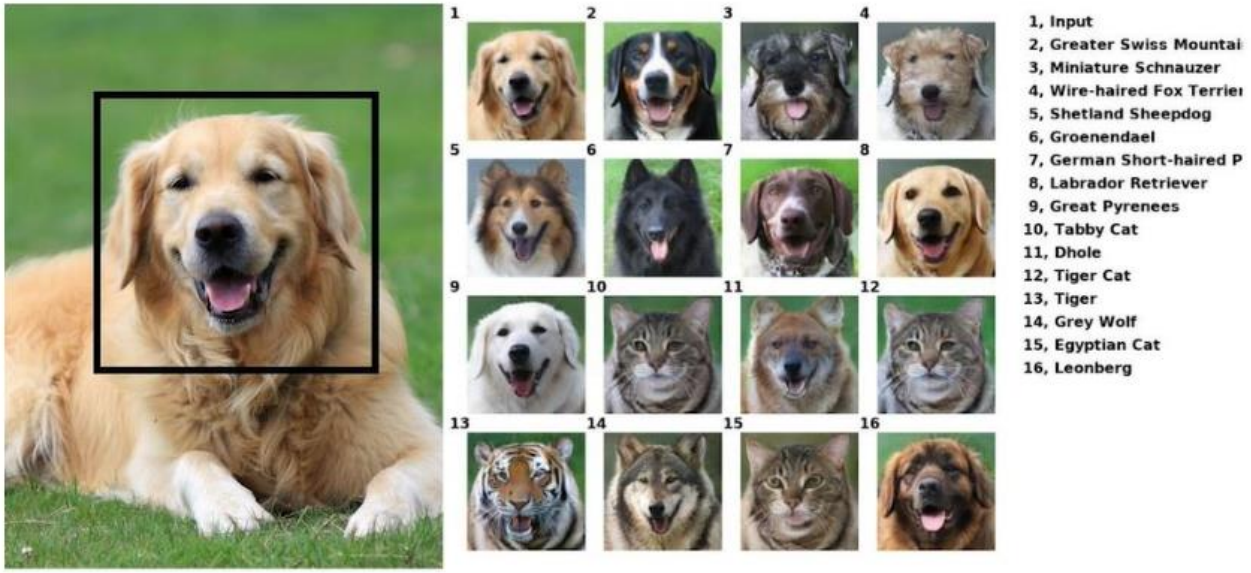
**NVIDIA AI GauGAN으로 몇 초안에 스케치를 사실적 풍경으로 탄생**




이 모델을 사용한 앱은 인상파 대표 화가인 폴 고갱의 이름을 따 고갱(GauGAN)으로 불립니다.

# 13.3 적대적 생성 신경망(GAN)이란

1) **Edge정보에 채색을 하는 GAN**
2) **특정 화가의 화풍을 학습, 적용하는 '딥드림'**
3) **GAN을 기반으로 이미지 합성과 활용(비슷한 종의 개체를 합성 'GANimal'**
   - 동물의 사진을 보고 그 동물의 표정이나 포즈를 다른 동물에 똑같이 재현하는 AI 기술을 개발(10월 31, 2019 by NVIDIA KOREA)

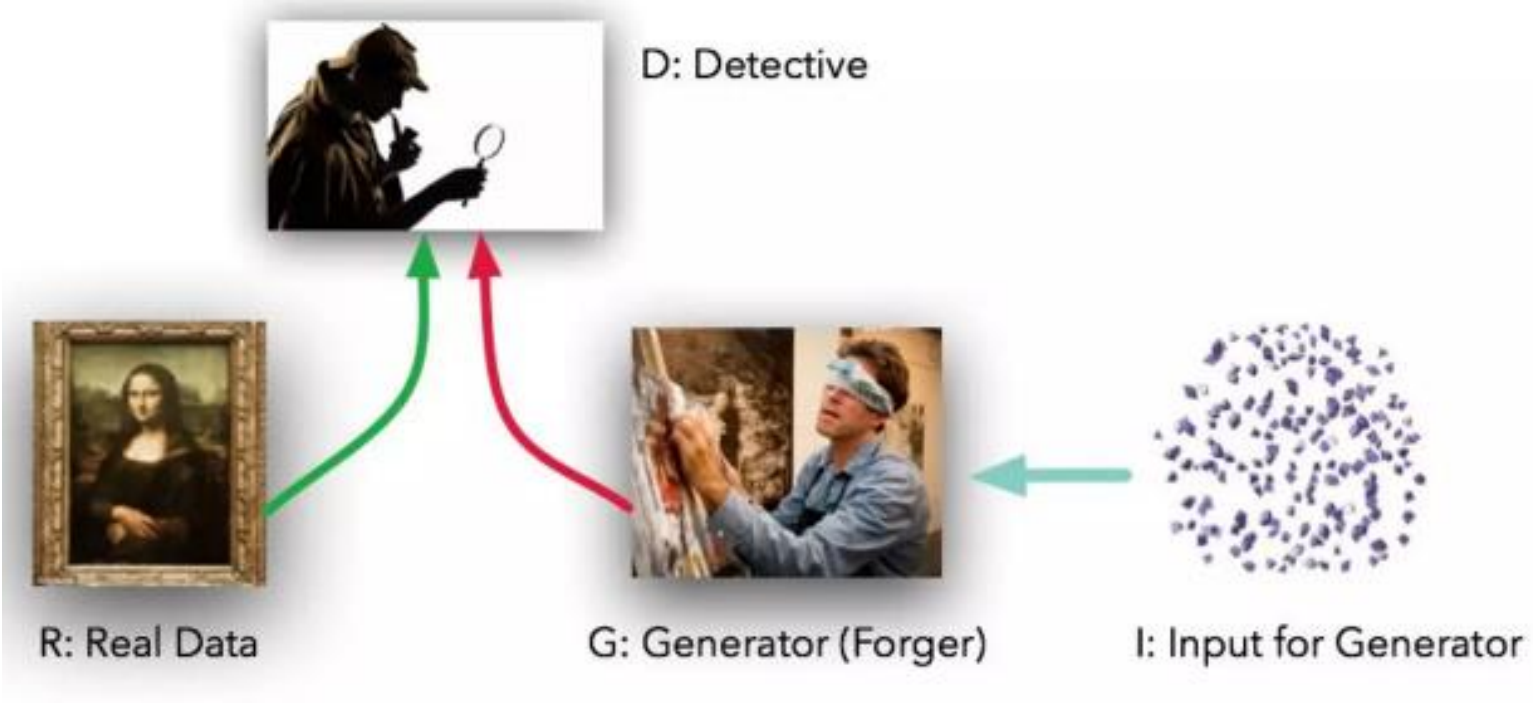



- **의료 분야**
- **가상인물 'Fake Face**

# 13.3 적대적 생성 신경망(GAN)이란

- StackGan++(2017)
  - 주어진 문장을 통해 사진을 창작
- 사진 복구 기술(Gan) – 상처난 이미지 복구
- MI-GAN –부족한 의료 이미지 데이터 보충
- DeblurGAN 이미지복구 – 흔들린 영상 복구, 의료용
- CT 스캔을 MRI로 전환 (MRI의 경우 일반적으로 **40만원** 전후로 가격이 형성)
- MRI를 CT영상으로 변환
  - CT 영상은 방사선 치료를 위한 방사선량 계산이나 양전자 단층촬영(PET) 감쇠 보정 등 다양한 임상 응용 분야에서 중요하나, 촬영 중 신체세포 손상 등 방사선 노출의 부작용 문제 발생
  - MRI 영상에서 CT 영상을 생성하기 위해 3D FCN 구조의 생성기(generator)로 3D 공간정보를 모델링하고 다음으로 추가 판별자 네트워크가 생성한 데이터에서 CT 이미지를 구분하는 판별기(discriminator)의 GAN 학습방법 재현

(참고: 2020. 11. 13 발행처_ 한국보건산업진흥원,p11)




위의 이미지는 GAN 간의 기능을 설명하는 훌륭한 비유입니다. Generator는 사기성 문서를 만드는 위조자로, Discriminator는 이를 탐지하려는 형사로 볼 수 있습니다.

- 생성자는 감별자를 속일 목적으로 훈련
- 감별자(탐정)

https://www.slideshare.net/ShamaneSiriwardhana/generative-adversarial-networks-slides-auckland-ai-ml-meetup

# 생성자 학습방식/ 감별자의 학습 방식

- 한번의 학습을 진행
- 생성자가 가짜 이미지 하나 생성해서 감별자에게 전달
- 감별자가 가짜라고 감별하면 실패
- 그러면, 생성자는 가중치 보정
- 감별자가 진짜라고 속으면 '성공'

- 두 번의 학습 진행
- 진품 창고에서 진짜 이미지 하나 추출
- 진짜 이미지 감별자에게 전달해서 진짜로 판별하도록 학습
- 생성자가 가짜 이미지 하나 생성
- 가짜 이미지를 감별자에게 전달
- 가짜로 판별하도록 학습

# 13.3 적대적 생성 신경망(GAN)이란

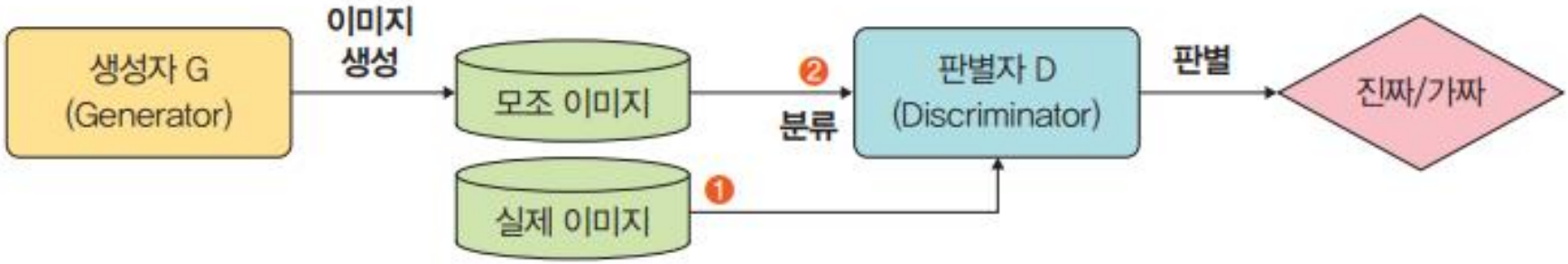
- **적대적 생성 신경망(GAN)이란**
  - 딥러닝 용어로 설명하자면, 경찰은 진짜 지폐와 위조지폐를 구분하는 판별자가 되며 위조지폐범은 위조지폐를 생성하는 생성자가 됨
  - 생성 모델은 최대한 진짜와 비슷한 데이터를 생성하려는 **생성자**와 진짜와 가짜를 구별하는 **판별자**가 각각 존재하여 서로 적대적으로 학습

# 13.3 적대적 생성 신경망(GAN)이란

- **적대적 생성 신경망(GAN)이란**
  - 적대적 학습에서는 판별자를 먼저 학습시킨 후 생성자를 학습시키는 과정을 반복
  - 판별자 학습은 크게 두 단계로 진행
  - 먼저 실제 이미지를 입력해서 네트워크(신경망)가 해당 이미지를 진짜로 분류하도록 학습
  - 그런 다음 생성자가 생성한 모조 이미지를 입력해서 해당 이미지를 가짜로 분류하도록 학습
  - 이 과정을 거쳐 판별자는 실제 이미지를 진짜로 분류하고, 모조 이미지를 가짜로 분류

# 13.3 적대적 생성 신경망(GAN)이란

▼ 그림 13-21 적대적 생성 신경망 학습 과정



# 13.3 적대적 생성 신경망(GAN)이란

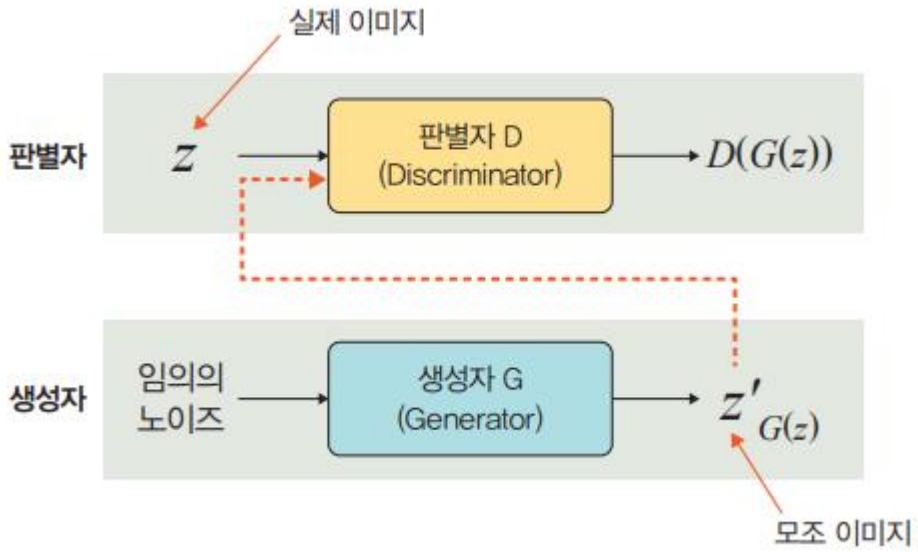
- **적대적 생성 신경망(GAN)이란**
  - 이와 같은 학습 과정을 반복하면 판별자와 생성자가 서로를 적대적인 경쟁자로 인식하여 모두 발전하게 됨
  - 결과적으로 생성자는 진짜 이미지에 완벽히 가까울 정도의 유사한 모조 이미지를 만들고, 이에 따라 판별자는 실제 이미지와 모조 이미지를 구분할 수 없게 됨
  - 즉, 생성자는 분류에 성공할 확률을 낮추고 판별자는 분류에 성공할 확률을 높이면서 서로 경쟁적으로 발전시키는 구조

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 동작 원리**
  - 적대적 생성 신경망(GAN)은 생성자(Generator)와 판별자(Discriminator) 네트워크 두 개로 구성
  - 이름에서 알 수 있듯이 두 네트워크는 서로 적대적으로 경쟁하여 학습을 진행
  - 생성자 G는 판별자 D를 속이려고 원래 이미지와 최대한 비슷한 이미지를 만들도록 학습
  - 반대로 판별자 D는 원래 이미지와 생성자 G가 만든 이미지를 잘 구분하도록 학습을 진행

# 13.3 적대적 생성 신경망(GAN)이란

▼ 그림 13-23 생성자와 판별자



# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 동작 원리**
  - 실제 데이터를 판단하려고 판별자 D를 학습시킬 때는 생성자 G를 고정시킨 채 실제 이미지는 높은 확률을 반환하는 방향으로, 모조 이미지는 낮은 확률을 반환하는 방향으로 가중치를 업데이트

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 동작 원리**
  - GAN 구조를 살펴보았으니, 이제 GAN의 손실 함수를 살펴보자
  - 먼저 GAN의 손실 함수는 다음과 같음

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

- **$x\sim P_{data}(x)$**: 실제 데이터에 대한 확률 분포에서 샘플링한 데이터
- **$z\sim P_z(z)$**: 가우시안 분포를 사용하는 임의의 노이즈에서 샘플링한 데이터
- **D(x)**: 판별자 D(x)가 1에 가까우면 진짜 데이터로 0에 가까우면 가짜 데이터로 판단,
- 0이면 가짜를 의미

  **D(G(z))**: 생성자 G가 생성한 이미지인 G(z)가 1에 가까우면 진짜 데이터로, 0에

  가까우면 가짜 데이터로 판단

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 동작 원리**
  - 이때 판별자 입장에서는 D(x)=1, D(G(z))=0이 최상의 결과(진짜 이미지는 1, 가짜 이미지는 0을 출력할 경우)가 될 것이기 때문에 이 식의 최댓값으로 업데이트
  - 또한, 판별자 입장에서는 log(D(x))와 log(1-D(G(z))) 모두 최대가 되어야 함
  - 즉, D(x)는 1이 되어야 실제 이미지를 진짜라고 분류하며, 1-D(G(z))는 1이 되어야 생성자가 만든 모조 이미지를 가짜라고 분류

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - GAN 예제 역시 MNIST 데이터셋을 사용하여 파이토치로 구현하는 방법을 알아보자 먼저 필요한 라이브러리를 호출

**코드 13-22** 라이브러리 호출

```
import imageio ------ 이미지 데이터를 읽고 쓸 수 있는 쉬운 인터페이스를 제공하는 라이브러리
from tqdm import tqdm

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import matplotlib.pylab as plt

from torchvision.utils import make_grid, save_image
import torchvision.datasets as datasets
import torchvision.transforms as transforms
matplotlib.style.use('ggplot') ------①
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - ① 맷플롯립(matplotlib) 라이브러리는 시각화에 사용
  - 맷플롯립은 폰트, 색상 등을 변경하여 사용할 수 있는데 예제에서는 스타일시트(stylesheet)를 바꾸어서 사용
  - 스타일시트로 사용할 수 있는 것들은 다음 코드와 같이 확인할 수 있음

```
import matplotlib.pyplot as plt

plt.style.available
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 다음은 사용 가능한 스타일시트 목록에 대한 출력 결과

```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 학습을 위해 사용 가능한 스타일시트를 하나씩 적용해 보는 것도 도움이 됨
  - 필요한 변수에 대한 값을 지정

**코드 13-23** 변수 값 설정

```
batch_size = 512
epochs = 200
sample_size = 64 ------①
nz = 128 ------②
k = 1 ------③
```

- **GAN 구현**
  - ① 노이즈 벡터를 사용하여 가짜 이미지를 생성
  - sample_size는 생성자에 제공할 고정 크기의 노이즈 벡터에 대한 크기
  - ② 잠재 벡터의 크기를 의미
  - 이때 잠재 벡터의 크기는 생성자의 입력 크기와 동일
  - ③ 판별자에 적용할 스텝 수를 의미
  - 스텝 수를 1로 지정한 이유는 훈련 비용을 최소화하기 위함

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 예제에서 진행할 MNIST를 내려받아 정규화를 적용
  - 이후에는 데이터로더에 데이터를 전달하여 모델의 학습에 사용할 수 있도록 함

**코드 13-24** MNIST를 내려받은 후 정규화

```
transform = transforms.Compose([
                                transforms.ToTensor(), ------ 이미지를 텐서로 변환
                                transforms.Normalize((0.5,),(0.5,)), -------
])                                    이미지를 평균이 0.5, 표준편차가 0.5가 되도록 정규화

train_dataset = datasets.MNIST(
    root="../chap13/data", train=True, transform=transform, download=True)

train_loader = DataLoader(
    train_dataset, batch_size=batch_size, shuffle=True, num_workers=4)
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 데이터셋이 준비되었기 때문에 네트워크를 생성할 텐데, 먼저 생성자 네트워크를 만들어 보자
  - 간단한 예제를 위해 네 개의 선형 계층과 세 개의 리키렐루(LeakyReLU) 활성화 함수를 사용

**코드 13-25** 생성자 네트워크 생성

```
class Generator(nn.Module):
    def __init__(self, nz):
        super(Generator, self).__init__()
        self.nz = nz
        self.main = nn.Sequential(
            nn.Linear(self.nz, 256),
            nn.LeakyReLU(0.2),
            nn.Linear(256, 512),
            nn.LeakyReLU(0.2),
            nn.Linear(512, 1024),
            nn.LeakyReLU(0.2),
            nn.Linear(1024, 784),
            nn.Tanh(),
        )
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**

```
def forward(self, x):
    return self.main(x).view(-1, 1, 28, 28)
```

------ 생성자 네트워크의 반환값은 '배치 크기×1×28×28'이 됩니다.

- **Leaky ReLU 활성화 함수**
  - ReLU 함수에서 파생
  - ReLU의 단점인 죽은 뉴런을 방지하는 효과
  - 기울기 유실 문제 부분적 해결
  - 음수쪽의 기울기 조정 가능

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 생성자 네트워크가 완료되었고, 이제 판별자 네트워크를 생성해 보자
  - 판(감)별자는 이진 분류자라는 것을 고려하여 신경망을 구축
  - 목적 : 이미지 감별(진짜 v 가짜 )
  - 입력: 이미지 출력:[0,1]확률(0 가짜, 1 진짜)

**코드 13-26** 판별자 네트워크 생성

```
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.n_input = 784 ------ 판별자의 입력 크기
        self.main = nn.Sequential( ------ 판별자 역시 선형 계층과 리키렐루 활성화 함수로 구성
            nn.Linear(self.n_input, 1024),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.3),
            nn.Linear(1024, 512),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.3),
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**

```
            nn.Linear(512, 256),
            nn.LeakyReLU(0.2),
            nn.Dropout(0.3),
            nn.Linear(256, 1),
            nn.Sigmoid(),
        )
    def forward(self, x):
        x = x.view(-1, 784)
        return self.main(x) ------ 이미지가 진짜인지 가짜인지를 분류하는 값을 반환
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 앞에서 생성한 생성자와 판별자 네트워크를 초기화
  - 이때 생성자는 잠재 벡터 nz라는 변수를 파라미터로 전달

**코드 13-27** 생성자와 판별자 네트워크 초기화

```
generator = Generator(nz).to(device)
discriminator = Discriminator().to(device)
print(generator)
print(discriminator)
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 다음은 앞에서 생성한 생성자와 판별자 네트워크를 보여 줌

```
Generator(
  (main): Sequential(
    (0): Linear(in_features=128, out_features=256, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=256, out_features=512, bias=True)
    (3): LeakyReLU(negative_slope=0.2)
    (4): Linear(in_features=512, out_features=1024, bias=True)
    (5): LeakyReLU(negative_slope=0.2)
    (6): Linear(in_features=1024, out_features=784, bias=True)
    (7): Tanh()
  )
)
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**

```
Discriminator(
  (main): Sequential(
    (0): Linear(in_features=784, out_features=1024, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Dropout(p=0.3, inplace=False)
    (3): Linear(in_features=1024, out_features=512, bias=True)
    (4): LeakyReLU(negative_slope=0.2)
    (5): Dropout(p=0.3, inplace=False)
    (6): Linear(in_features=512, out_features=256, bias=True)
    (7): LeakyReLU(negative_slope=0.2)
    (8): Dropout(p=0.3, inplace=False)
    (9): Linear(in_features=256, out_features=1, bias=True)
    (10): Sigmoid()
  )
)
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 이제 모델의 네트워크에서 사용할 옵티마이저와 손실 함수를 정의
  - 중요한 것은 생성자와 판별자에서 사용할 옵티마이저를 따로 정의해야 한다는 것

**코드 13-28** 옵티마이저와 손실 함수 정의

```
optim_g = optim.Adam(generator.parameters(), lr=0.0002)
optim_d = optim.Adam(discriminator.parameters(), lr=0.0002)

criterion = nn.BCELoss()

losses_g = [] ------ 매 에포크마다 발생하는 생성자 오차를 저장하기 위한 리스트형 변수
losses_d = [] ------ 매 에포크마다 발생하는 판별자 오차를 저장하기 위한 리스트형 변수
images = [] ------ 생성자에 의해 생성되는 이미지를 저장하기 위한 리스트형 변수
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 생성자에 의해 만들어지는 새로운 이미지(텐서)를 저장하기 위한 함수를 저장
  - 이 함수는 모델 학습에 반드시 필요한 것은 아니지만 이미지가 생성되는 과정을 이해할 수 있도록 시각화하여 보여 줌

**코드 13-29** 생성된 이미지 저장 함수 정의

```
def save_generator_image(image, path):
    save_image(image, path)
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 판별자를 학습시키기 위한 함수를 정의
  - 판별자의 학습은 진짜 데이터의 레이블과 가짜 데이터의 레이블을 모두 이용하여 학습

**코드 13-30** 판별자 학습을 위한 함수

```
def train_discriminator(optimizer, data_real, data_fake):
    b_size = data_real.size(0) ------ 배치 크기 정보 얻기
    real_label = torch.ones(b_size, 1).to(device) ------ ①
    fake_label = torch.zeros(b_size, 1).to(device) ------ ②
    optimizer.zero_grad()
    output_real = discriminator(data_real)
    loss_real = criterion(output_real, real_label) ------ 진짜 데이터를 판별자에 제공하여 학습한 결과와
                                                          진짜 데이터의 레이블을 이용하여 오차를 계산
    output_fake = discriminator(data_fake)
    loss_fake = criterion(output_fake, fake_label) ------ 가짜 데이터를 판별자에 제공하여 학습한 결과와
                                                          가짜 데이터의 레이블을 이용하여 오차를 계산
    loss_real.backward()
    loss_fake.backward()
    optimizer.step()
    return loss_real + loss_fake ------ 진짜 데이터와 가짜 데이터의 오차가 합쳐진 최종 오차를 반환
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 이제 생성자 학습을 위한 함수를 정의할 텐데, 상대적으로 판별자의 네트워크보다는 간단함

**코드 13-31** 생성자 학습을 위한 함수

```
def train_generator(optimizer, data_fake):
    b_size = data_fake.size(0)
    real_label = torch.ones(b_size, 1).to(device) ------①
    optimizer.zero_grad()
    output = discriminator(data_fake)
    loss = criterion(output, real_label)
    loss.backward()
    optimizer.step()
    return loss
```

  - ① 생성자 네트워크에서는 가짜 데이터만 사용하고 있는데, 생성자 입장에서는 가짜 데이터가 실제로 진짜라는 것에 주의할 필요가 있음

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 이제 for 구문을 이용하여 200회 에포크만큼 모델을 학습

**코드 13-32** 모델 학습

```
generator.train() ------ 생성자를 학습 모드로 설정
discriminator.train() ------ 판별자를 학습 모드로 설정

for epoch in range(epochs):
    loss_g = 0.0 ------ 생성자 오차를 추적(저장)하기 위한 변수
    loss_d = 0.0 ------ 판별자 오차를 추적(저장)하기 위한 변수
    for idx, data in tqdm(enumerate(train_loader), total=int(len(train_dataset)/train_
                                                        loader.batch_size)):

        image, _ = data ------ 학습을 위한 이미지 데이터를 가져옵니다.
        image = image.to(device) ------ 데이터셋이 CPU/GPU 장치를 사용하도록 지정
        b_size = len(image)
        for step in range(k): ------ k(1) 스텝 수에 따라 판별자를 실행, 이때 k 수를 증가시킬 수 있지만
                                    학습 시간이 길어질 수 있으므로 주의하세요.
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**

```
            data_fake = generator(torch.randn(b_size, nz).to(device)).detach() ------ ①
            data_real = image
            loss_d += train_discriminator(optim_d, data_real, data_fake) ------ ①'
        data_fake = generator(torch.randn(b_size, nz).to(device))
        loss_g += train_generator(optim_g, data_fake) ------ 생성자 학습
    generated_img = generator(torch.randn(b_size, nz).to(device)).cpu().detach()
    generated_img = make_grid(generated_img) ------ 이미지를 그리드 형태로 표현
    save_generator_image(generated_img, f"../chap13/img/gen_img{epoch}.png")
    images.append(generated_img)                          생성된 이미지(텐서)를 디스크에 저장
    epoch_loss_g = loss_g / idx ------ 에포크에 대한 총 생성자 오차 계산
    epoch_loss_d = loss_d / idxH ------ 에포크에 대한 총 판별자 오차 계산
    losses_g.append(epoch_loss_g)                  생성자를 이용하여 새로운 이미지를 생성하고
    losses_d.append(epoch_loss_d)                        CPU 장치를 이용하여 디스크에 저장

    print(f"Epoch {epoch} of {epochs}")
    print(f"Generator loss: {epoch_loss_g:.8f}, Discriminator loss: {epoch_loss_d:.8f}")
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - ① 생성자에서 새로운 이미지 데이터(텐서)를 생성하기 위해서는 노이즈 데이터가 필요함
  - 노이즈 데이터는 torch.randn()을 이용해서 생성할 수 있으며, 여기에서 사용되는 파라미터는 다음과 같음

```
data_fake = generator(torch.randn(b_size, nz).to(device)).detach()
                      ⓐ                    ⓑ          ⓒ
```

ⓐ 생성자에서 가짜 이미지를 생성하기 위해서는 생성자에 노이즈 데이터(벡터)를 제공
노이즈 데이터는 잠재 벡터(nz)의 크기와 동일해야 하며, 평균이 0이고 표준편차가 1인 가우시안 정규분포를 이용하여 (b_size×nz) 크기를 갖도록 함

ⓑ 모델이 데이터를 처리하려면 모델과 데이터 모두 동일한 장치(CPU 혹은 GPU)를 사용
여기에서는 to(device)를 이용하여 모델이 사용할 장치를 지정

ⓒ detach()는 + 와 같은 기능을 함
즉, detach()를 통해 떼어 낸 데이터를 이용하여 ①'처럼 판별자를 학습시키고 그 결과를 loss_d에 붙여 넣음

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 다음은 모델을 학습시킨 결과

```
Epoch 0 of 200
Generator loss: 1.28965569, Discriminator loss: 0.94043517
Epoch 1 of 200
Generator loss: 2.55902362, Discriminator loss: 1.09625375
Epoch 2 of 200
Generator loss: 6.74595404, Discriminator loss: 0.20054729
Epoch 3 of 200
Generator loss: 1.58426058, Discriminator loss: 1.02565455
Epoch 4 of 200
Generator loss: 2.40962863, Discriminator loss: 1.11043680
Epoch 5 of 200
Generator loss: 4.19420624, Discriminator loss: 0.73168665
... 중간 생략 ...
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**

```
Epoch 194 of 200
Generator loss: 1.26986551, Discriminator loss: 1.09756744
Epoch 195 of 200
Generator loss: 1.28353238, Discriminator loss: 1.06755567
Epoch 196 of 200
Generator loss: 1.26950192, Discriminator loss: 1.08437514
Epoch 197 of 200
Generator loss: 1.23257947, Discriminator loss: 1.10518491
Epoch 198 of 200
Generator loss: 1.28899515, Discriminator loss: 1.08550680
Epoch 199 of 200
Generator loss: 1.31970513, Discriminator loss: 1.07432187
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 다음 그림은 모델 학습 결과 새로 생성된 이미지에 대한 결과

▼ **그림 13-25 에포크 0번째에서 생성된 이미지**

# 13.3 적대적 생성 신경망(GAN)이란

▼ 그림 13-27 에포크 100번째에서 생성된 이미지

# 13.3 적대적 생성 신경망(GAN)이란

▼ **그림 13-30 에포크 199번째에서 생성된 이미지**

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 0·1번째 에포크에서 생성된 이미지는 노이즈 그 이상의 의미를 갖지 않음
  - 100번째 에포크에서는 이미지의 형태가 보이지만 여전히 이미지 주위에 노이즈가 많은 것을 볼 수 있음
  - 마지막 199번째 에포크에서는 노이즈가 많이 없어지면서 좀 더 선명한 이미지를 보여 줌

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 이번에는 생성자와 판별자의 오차에 대한 변화를 그래프로 살펴보자

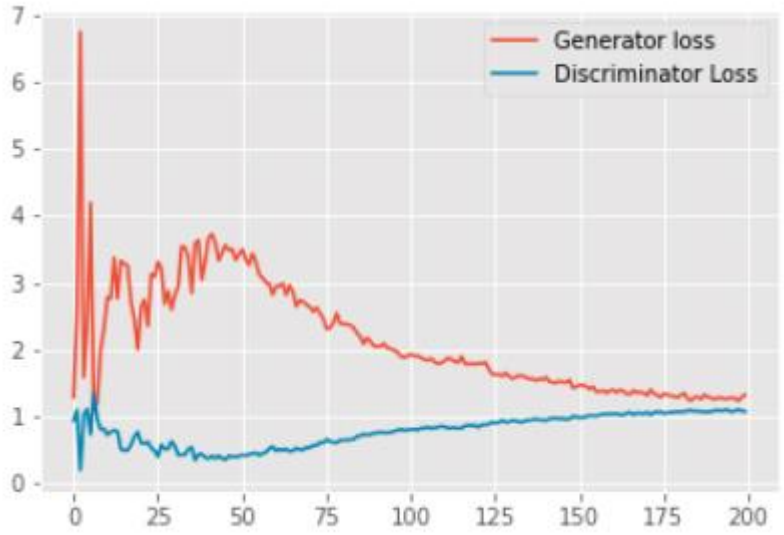
**코드 13-33** 생성자와 판별자의 오차 확인

```
plt.figure()
losses_g = [fl.item() for fl in losses_g]
plt.plot(losses_g, label='Generator loss')
losses_d = [f2.item() for f2 in losses_d]
plt.plot(losses_d, label='Discriminator Loss')
plt.legend()
```

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 다음 그림은 생성자와 판별자에 대한 오차를 시각적 그래프로 표현한 결과

▼ **그림 13-31 생성자와 판별자에 대한 오차**



# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 처음 몇 에포크 동안 생성자의 오차는 증가하고 판별자의 오차는 감소하는 것을 볼 수 있음
  - 이러한 증상이 나타나는 이유는 학습 초기 단계에 생성자는 좋은 가짜 이미지를 생성하지 못하기에 판별자가 실제 이미지와 가짜 이미지를 쉽게 구분할 수 있기 때문임
  - 학습이 진행됨에 따라 생성자는 진짜와 같은 가짜 이미지를 만들며 판별자는 가짜 이미지 중 일부를 진짜로 분류
  - 그림과 같이 생성자의 오차가 감소하면 판별자의 오차는 증가

# 13.3 적대적 생성 신경망(GAN)이란

- **GAN 구현**
  - 앞의 이미지를 하나씩 쪼개서 확인하고 싶다면 다음 코드와 같이 실행
  - 생성된 열 개의 이미지를 보여 줌

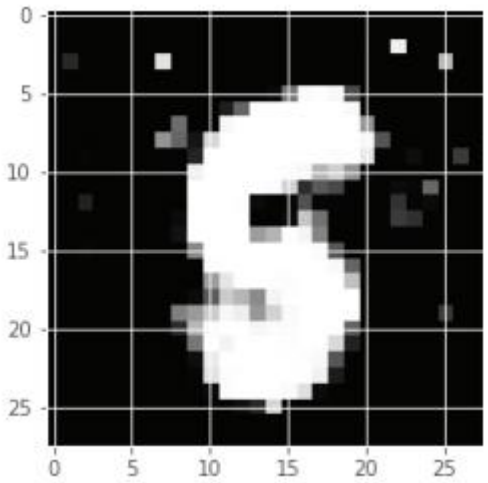
**코드 13-34** 생성된 이미지 출력

```
fake_images = generator(torch.randn(b_size, nz).to(device))
for i in range(10):
    fake_images_img = np.reshape(fake_images.data.cpu().numpy()[i],(28, 28))
    plt.imshow(fake_images_img, cmap='gray')
    plt.savefig('../chap13/img/fake_images_img' + str(i) + '.png')
    plt.show()
```

# 13.3 적대적 생성 신경망(GAN)이란

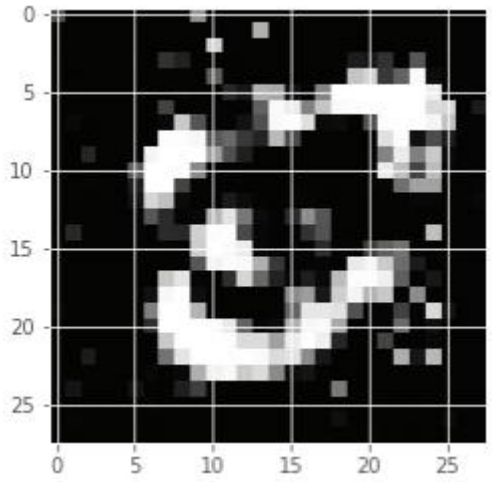
- **GAN 구현**
  - 다음 그림은 생성된 열 개의 이미지가 출력된 결과

▼ **그림 13-32 첫 번째 생성된 이미지**



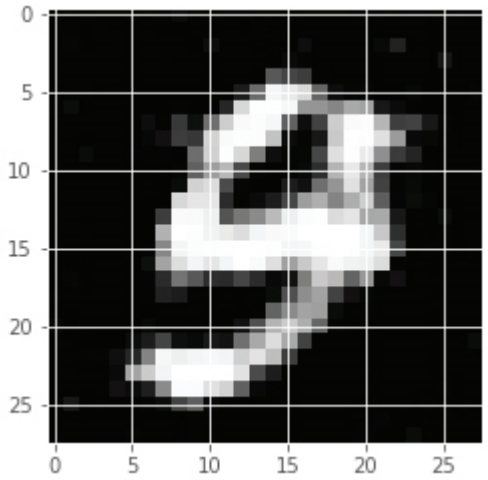
# 13.3 적대적 생성 신경망(GAN)이란

▼ **그림 13-34 아홉 번째 생성된 이미지**



# 13.3 적대적 생성 신경망(GAN)이란

▼ **그림 13-35 열 번째 생성된 이미지**



# 감사합니다

문제 은행

1.생성망과 판별망이 서로 적대적으로 경쟁하며 학습시키는 네트워크를 고르시오.
① Generative Adversarial Network
② Recurrent Neural Network
③ Convolution Neural Network
④ U-Net

2. GAN의 인공신경망 중 하나로, 허구데이터를 생성하는 인공신경망을 고르시오.
① 인코더
② 디코더
③ 생성망
④ 판별망