이미지 딥러닝 개인 프로젝트

훈련 안정성을 위한 병렬 구조의 생성자 기반 GAN

GAN based on Parallel Structured Generator for Stability Training

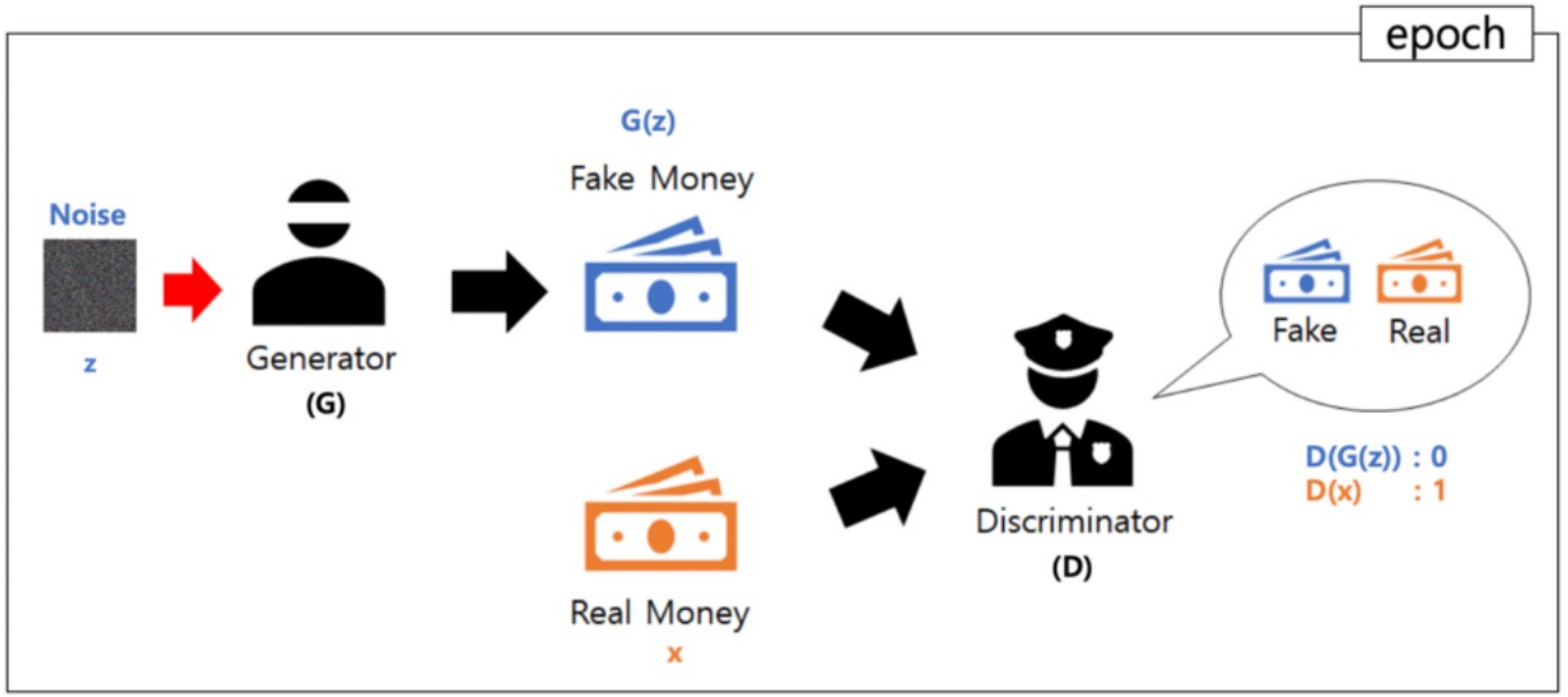
2023.12.13 (수)

한국성서대학교 컴퓨터소프트웨어학과 3학년

박범찬(Bumchan Park)

Index

GAN (Generative Adversarial Network)



목적

-생성자 : D(G(z))를 1로 만들기 위함

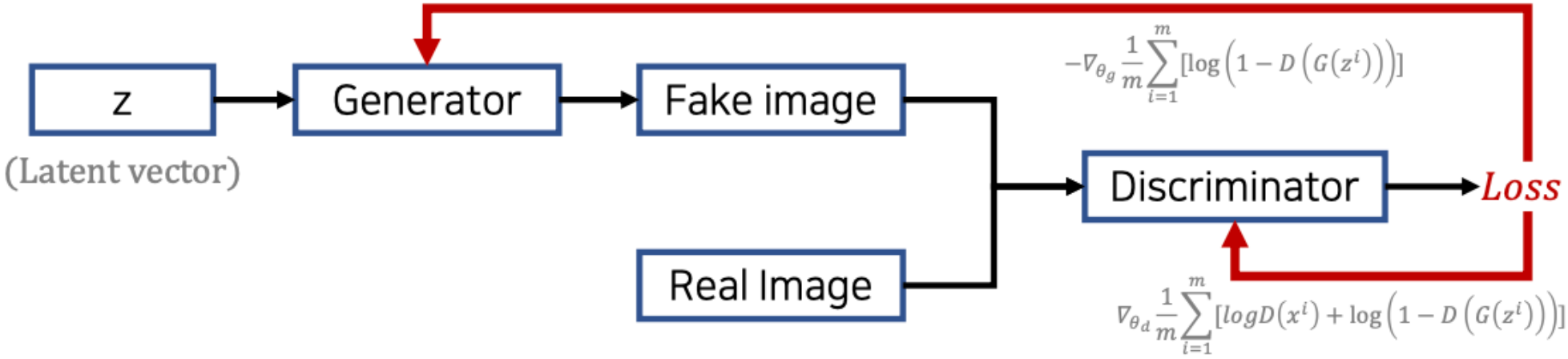
-판별자 : D(G(z))과 D(x)를 잘 구분하기 위함

GAN (Generative Adversarial Network)

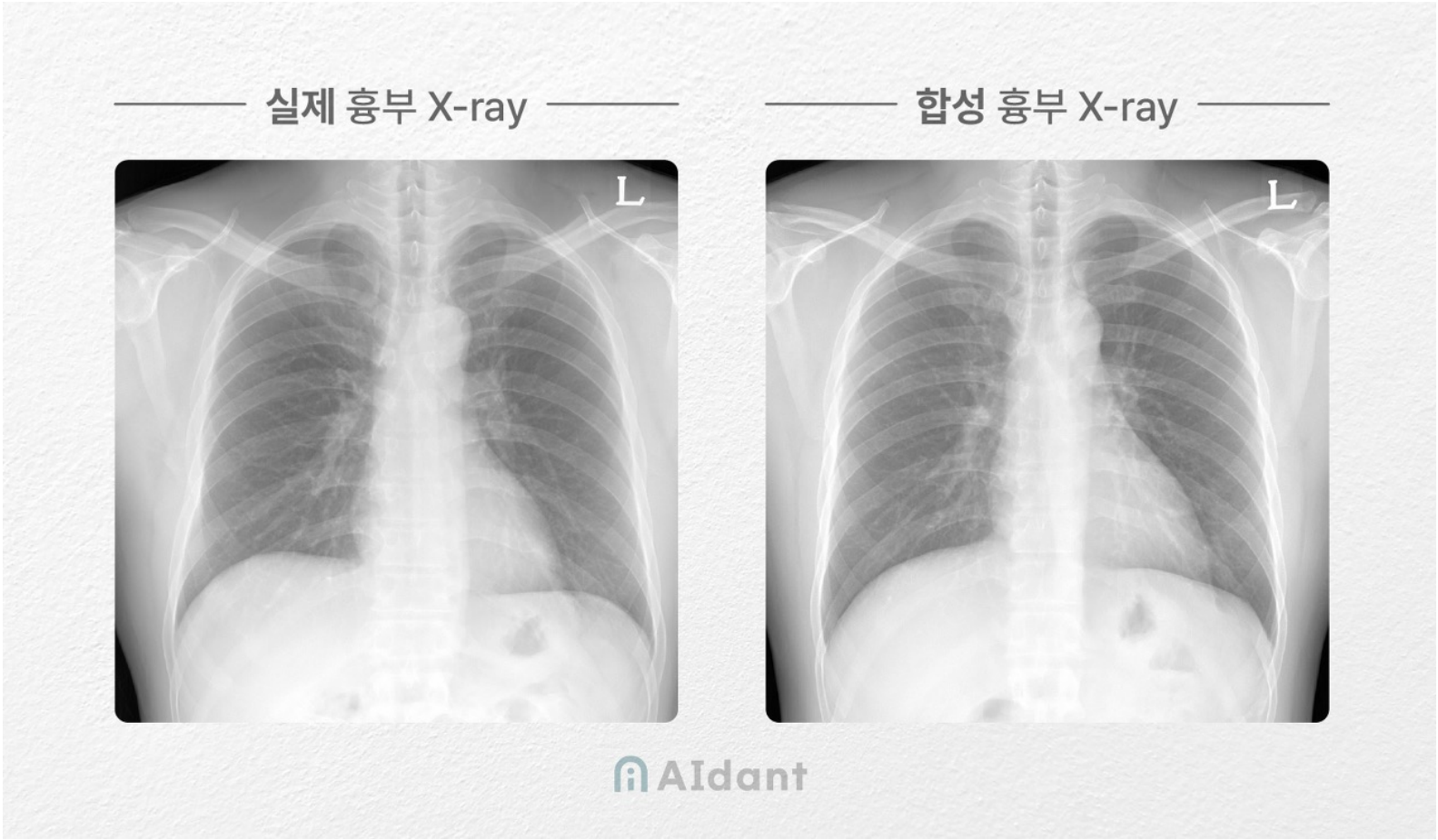
$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Generator — G(z): *new data instance*

Discriminator — $D(x) = Probability$: *a sample came from the real distribution* (*Real*: 1 ~ *Fake*: 0)



서론

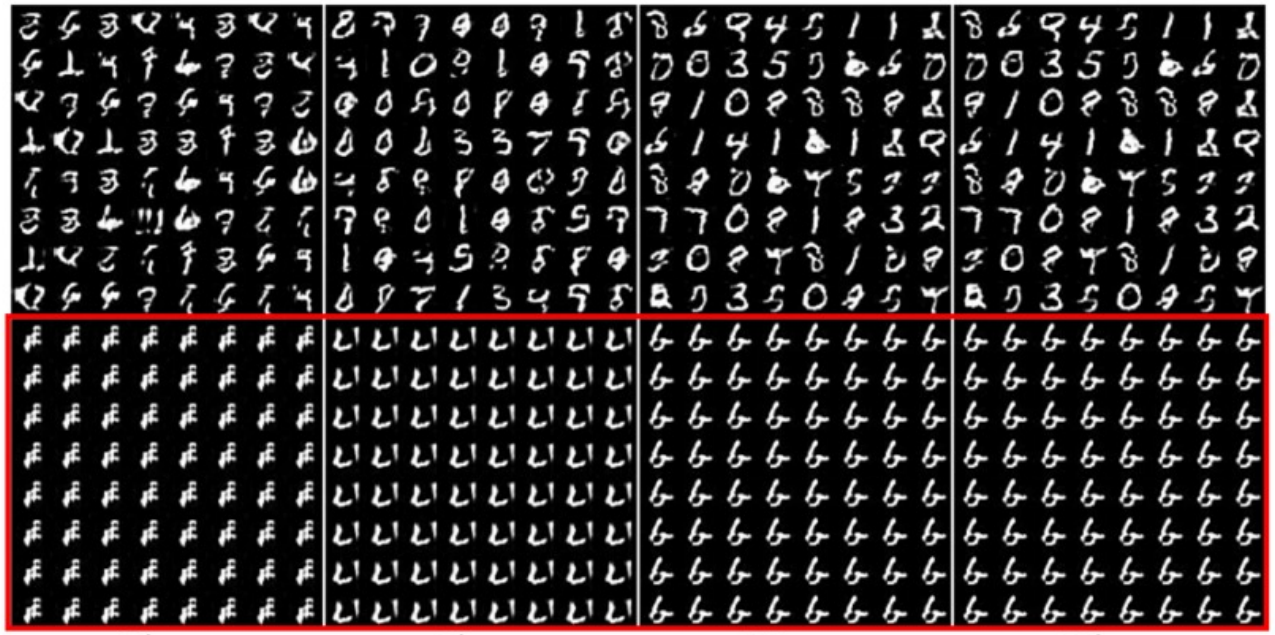


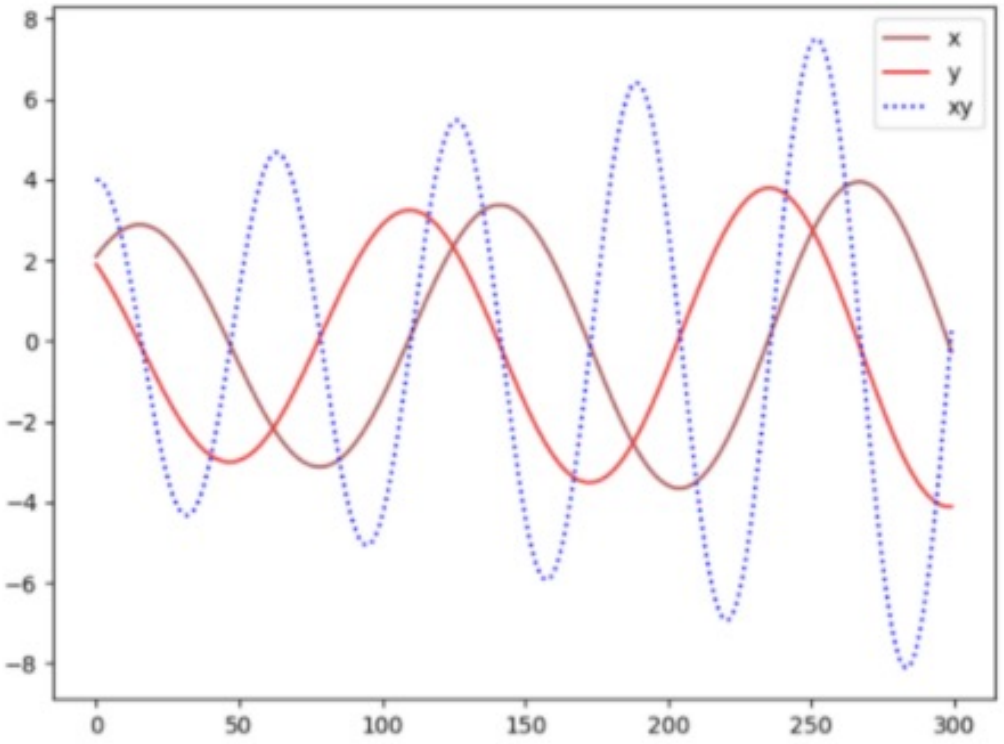
민감한 정보를 규정하고 있는 만큼, 많은 학습 데이터를 필요로 하는 의료 인공지능 분야

정보 주체의 프라이버시를 침해하지 않을 수 있는, 규제에서 비교적 자유롭다고 여겨지는 합성데이터

서론

모드 붕괴 (Mode Collapse) :
생성자가 입력 값을 하나의 Mode에 치우쳐 변환시키는 현상





수렴 (Convergence) :
판별자와 생성자 간의 학습이 비슷하게 이뤄지는 지점

서론

제안 메커니즘 :

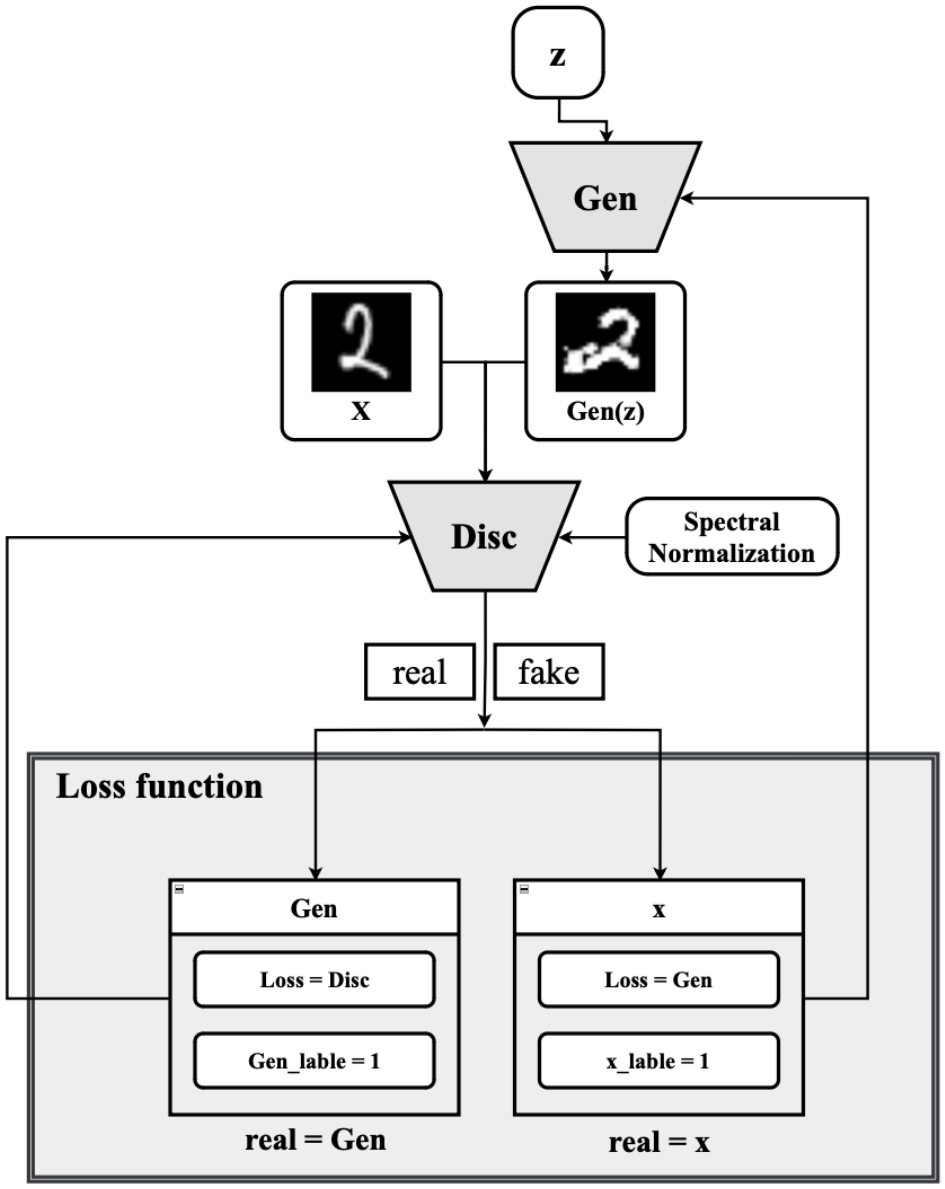
1. GAN의 생성자를 두 개로 변형하는 Parallel Structured Generator(병렬 생성자) 형식을 제안

2. Parallel Generator를 통해 얻은 Loss들을 평균을 내어 생성자에 적용하는 방법을 제안

=>GAN에서 발생하는 학습의 불안정성 문제를 해결

=>생성 모델이 생성한 데이터의 품질이나 성능을 더 발전

관련 연구 (SNGAN)



z = 노이즈 벡터

Gen = 생성자
Disc = 판별자

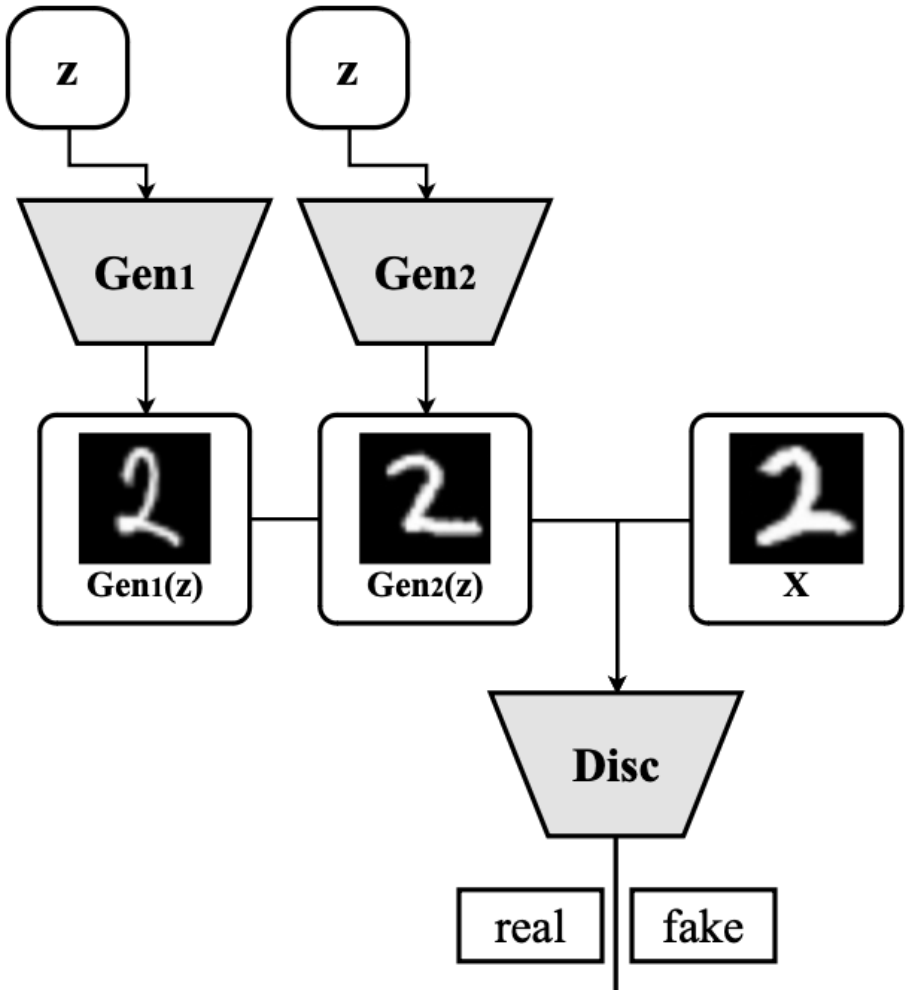
Gen(x) = 거짓 데이터
x = 실제 데이터

GAN이 수렴하기 어려운 이유 :
첫 번째, 판별자에 제약이 없음
두 번째, 계산이 불가능함

So,
Disc에 Spectral Normalization을 통해 제약을 두어
Disc가 Gen에 비해 강해지는 것을 막는 것이 목적!

But,
1. Spectral Normalization의 연산량이 큼
2. 모드 붕괴여부를 정확히 알기 어려움

제안한 방법 (Parallel Structured Generator)



Gen1 = 생성자1
Gen2 = 생성자2

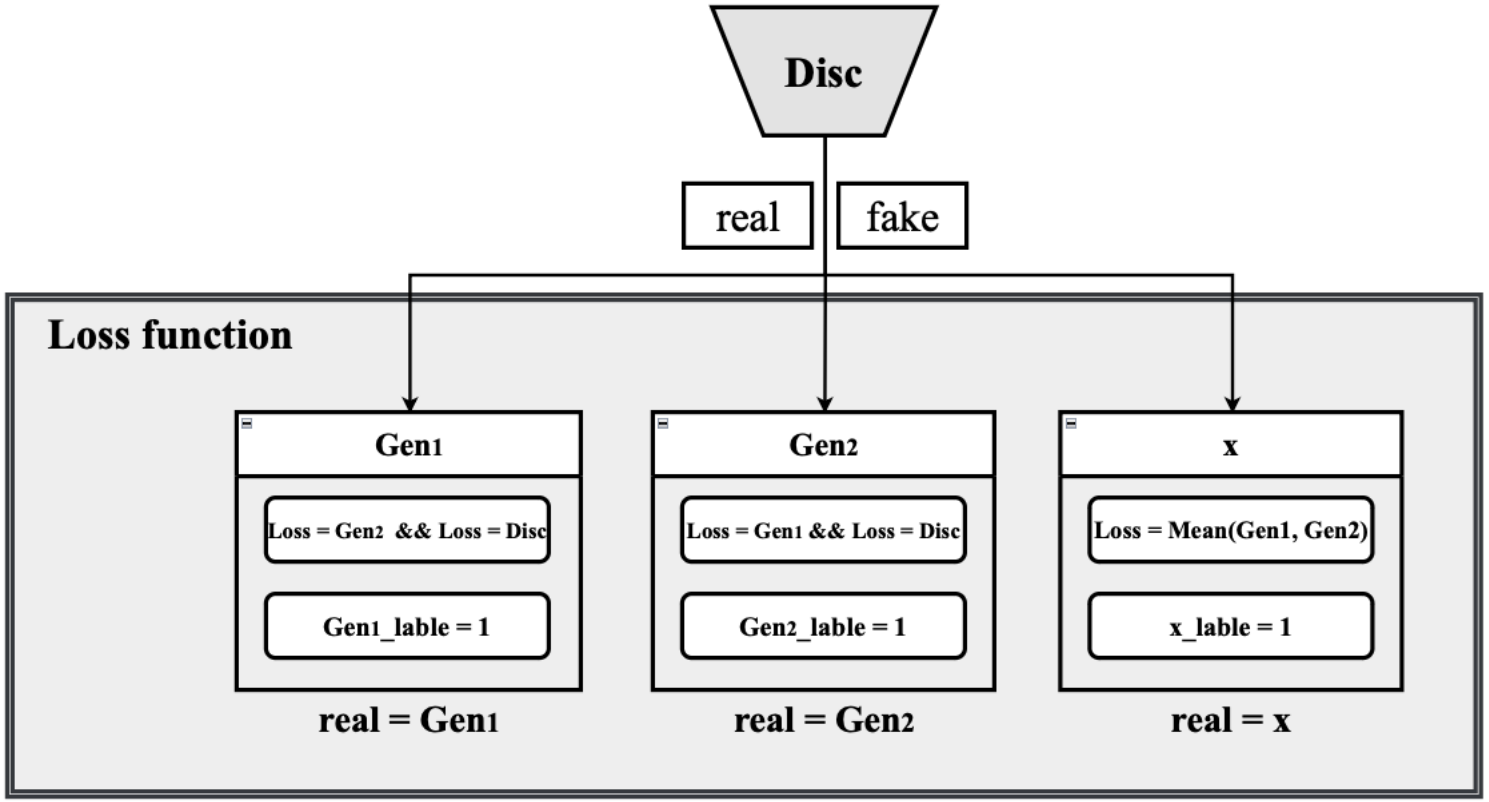
Gen1(z) = 거짓 데이터1
Gen2(z) = 거짓 데이터2

기존 GAN의 방법론과는 다르게(2개 데이터)
판별자에 총 세 개의 데이터가 분류기에 진입함

판별자의 분류 :
real – 1개 / fake – 2개

Disc에 들어가는 데이터의 개수를 늘려
Disc와 Gen1,2간 균형을 이루는 것이 목적!

제안한 방법 (Average Loss)



판별자의 분류 :
real – 1개 / fake – 2개

Gen1이 real로 판별된 경우 :
Loss : Gen2, x

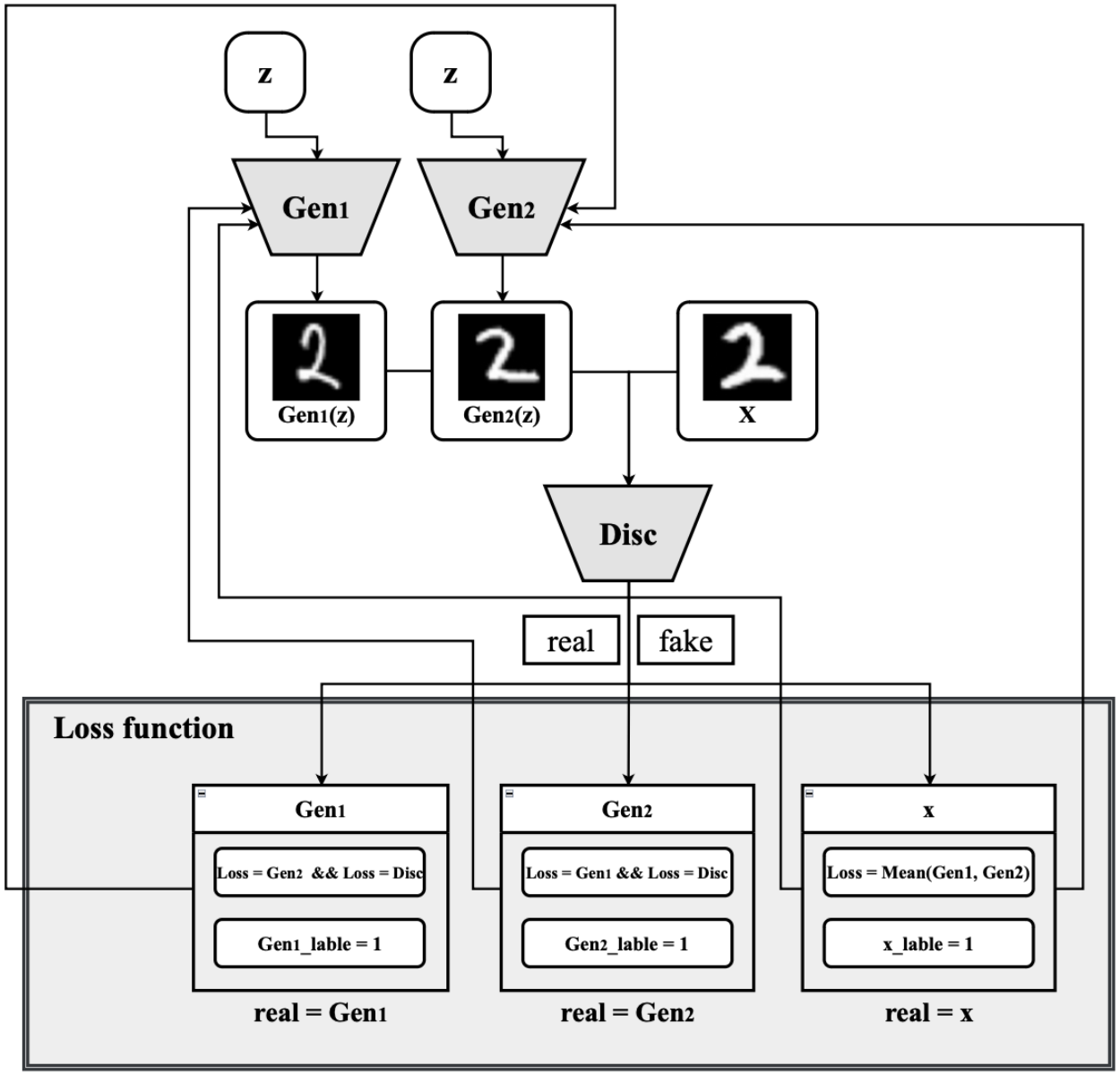
Gen2가 real로 판별된 경우 :
Loss : Gen1, x

x가 real로 판별된 경우 :
Loss : Gen1, Gen2의 평균

생성자와 판별자 간의 균형 뿐만 아니라,

두 생성자 간의 균형도 매우 중요함!

제안한 방법 (Average Loss)



코드 (Code)

```python
# Generator1
class Generator1(nn.Module):
    def __init__(self):
        super(Generator1, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(100, 256),
            nn.ReLU(),
            nn.Linear(256, 512),
            nn.ReLU(),
            nn.Linear(512, 784),
            nn.Tanh()
        )

    def forward(self, x):
        return self.model(x)

# Generator2
class Generator2(nn.Module):
    def __init__(self):
        super(Generator2, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(100, 256),
            nn.ReLU(),
            nn.Linear(256, 512),
            nn.ReLU(),
            nn.Linear(512, 784),
            nn.Tanh()
        )
```

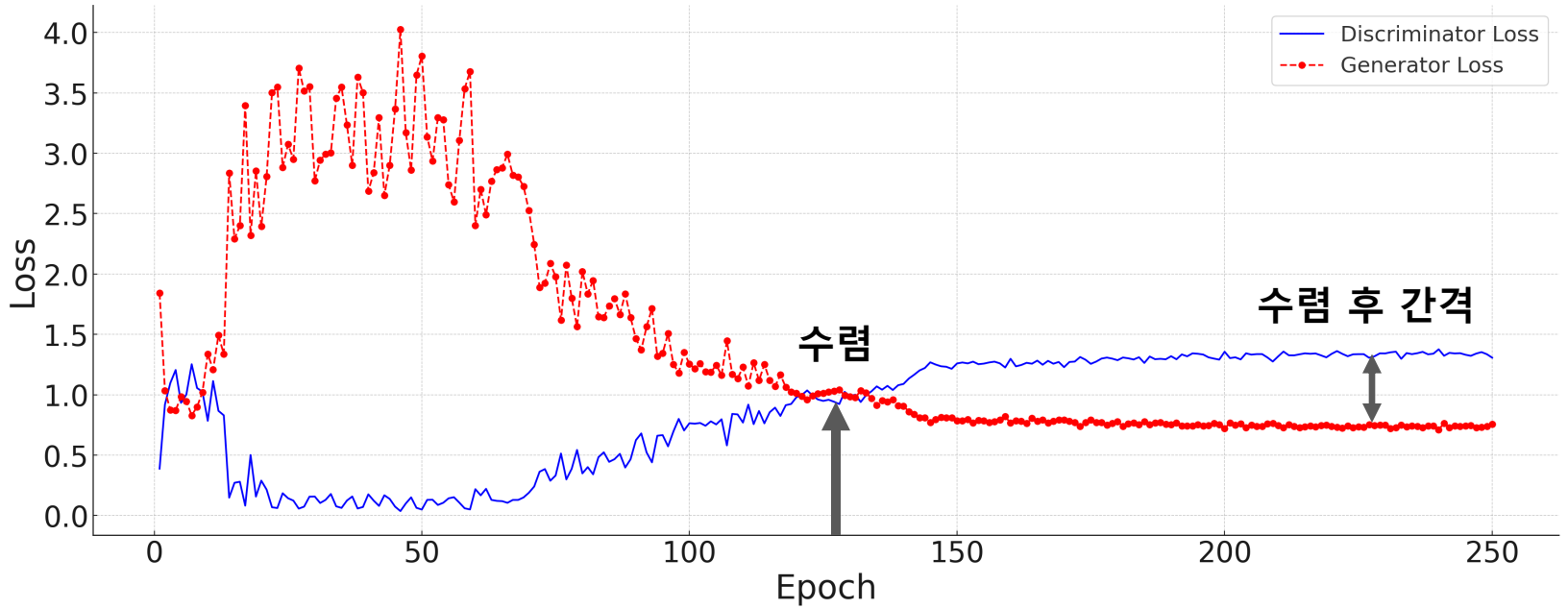
← 동일한 생성자를 두 개 생성한다

손실의 평균을 계산 후, Loss에 적용한다 →

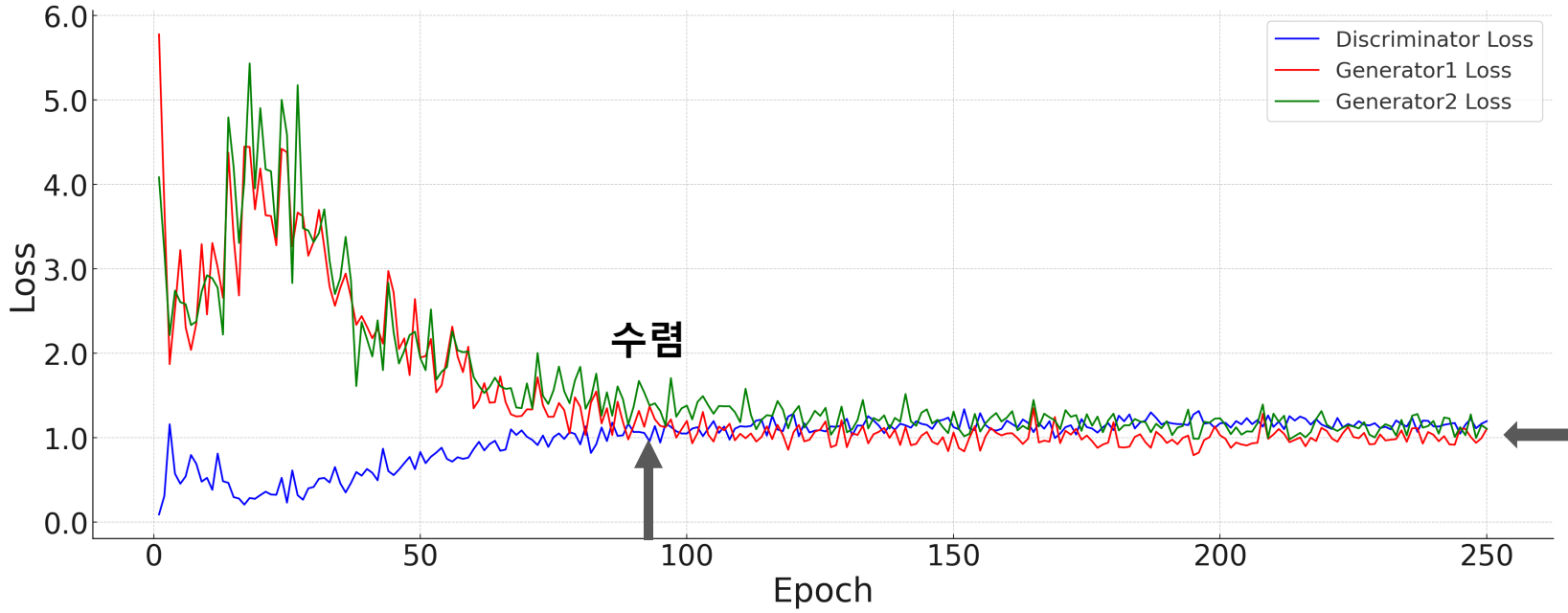
```python
# Train Discriminator
optimizer_D.zero_grad()
output_real = discriminator(real_images)
loss_real = criterion(output_real, real_labels)

noise1 = torch.randn(batch_size, 100)
fake_images1 = generator1(noise1)
output_fake1 = discriminator(fake_images1.detach())
loss_fake1 = criterion(output_fake1, fake_labels)

noise2 = torch.randn(batch_size, 100)
fake_images2 = generator2(noise2)
output_fake2 = discriminator(fake_images2.detach())
loss_fake2 = criterion(output_fake2, fake_labels)

loss_D = loss_real + (loss_fake1 + loss_fake2) / 2
loss_D.backward()
optimizer_D.step()
```

실험 결과

SNGAN

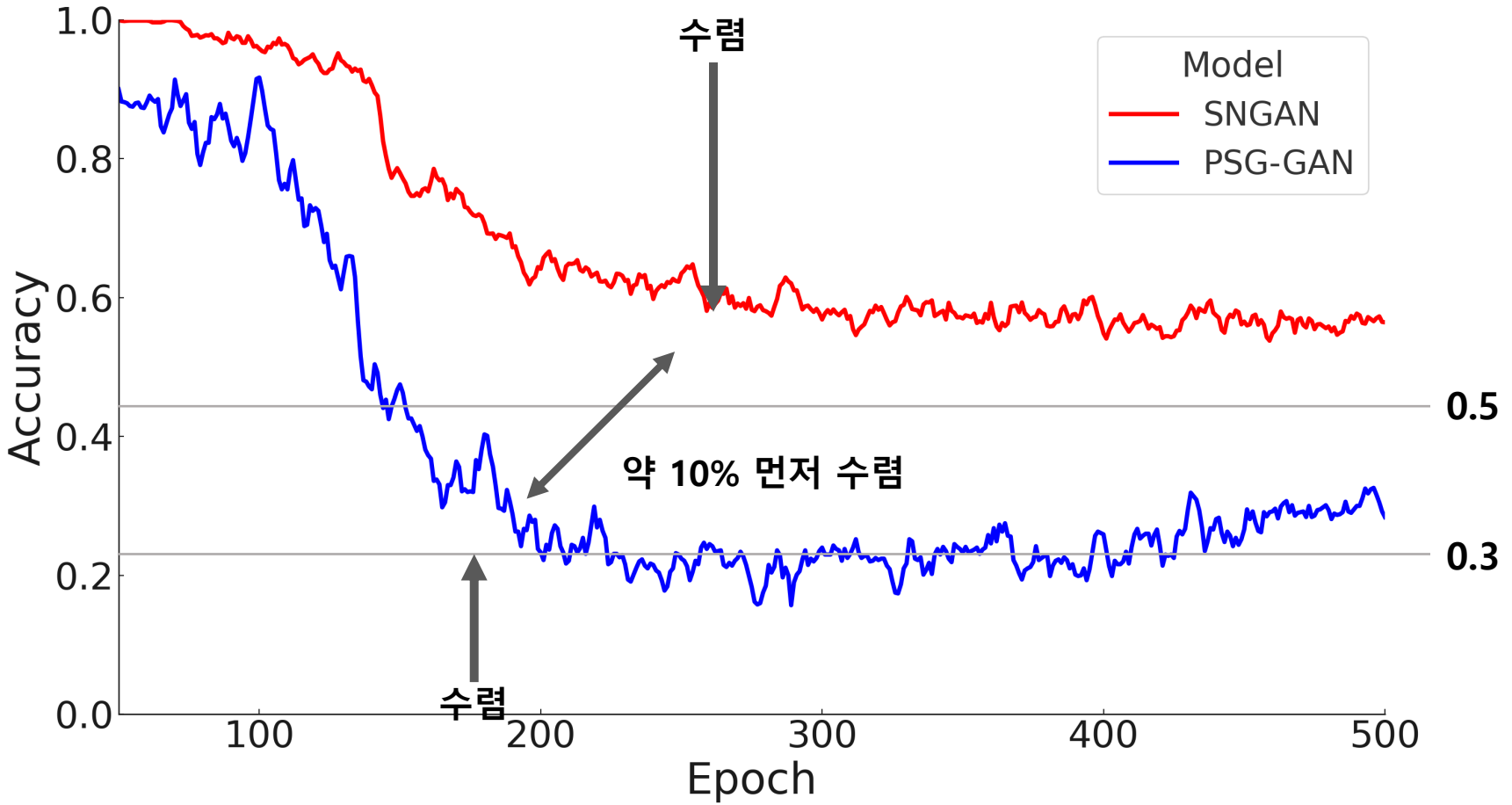


PSG-GAN



간격이 거의 없음

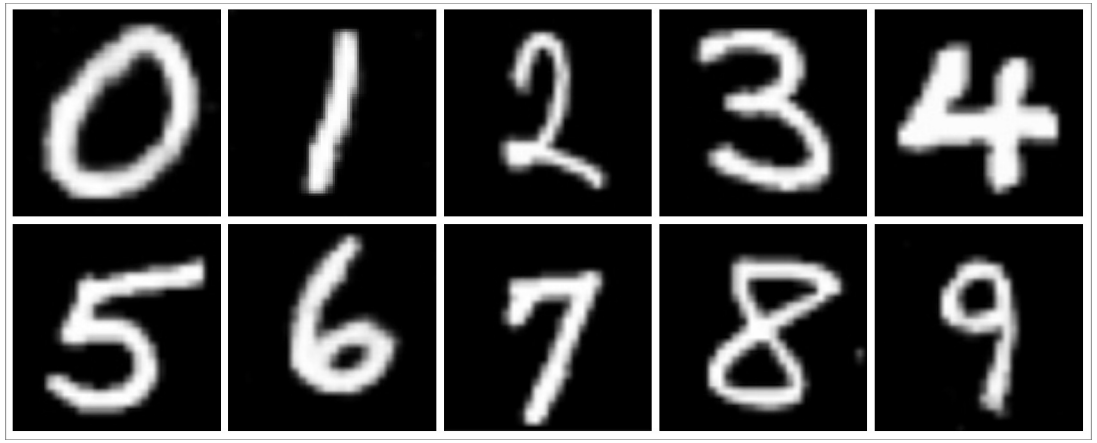
실험 결과


1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
100
200
300
400
500
Epoch
수렴
수렴
약 10% 먼저 수렴
0.5
0.3
Model
SNGAN
PSG-GAN

실험 결과

SNGAN

PSG-GAN



결론

1. PSG-GAN 알고리즘은 학습 과정에서의 불안정성과 과적합 방지에 탁월한 효과가 있음

2. PSG-GAN 알고리즘은 생성 데이터의 우수한 성능이 보임을 입증함

3. 정확하고 세밀한 작업을 필요로 하는 의료분야, 빠른 객체의 변화에 대한 인식을 필요로 하는 자율주행 차량 등에서 적절한 방향성을 제시할 수 있을 것

4. 향후에는 학습의 수렴속도에 따른 데이터의 품질과 수렴의 균형을 적절히 고려하여 더욱 안정적인 시스템을 구축하고자 함

Thank you