B935464 최승희

SQ-AGGAN

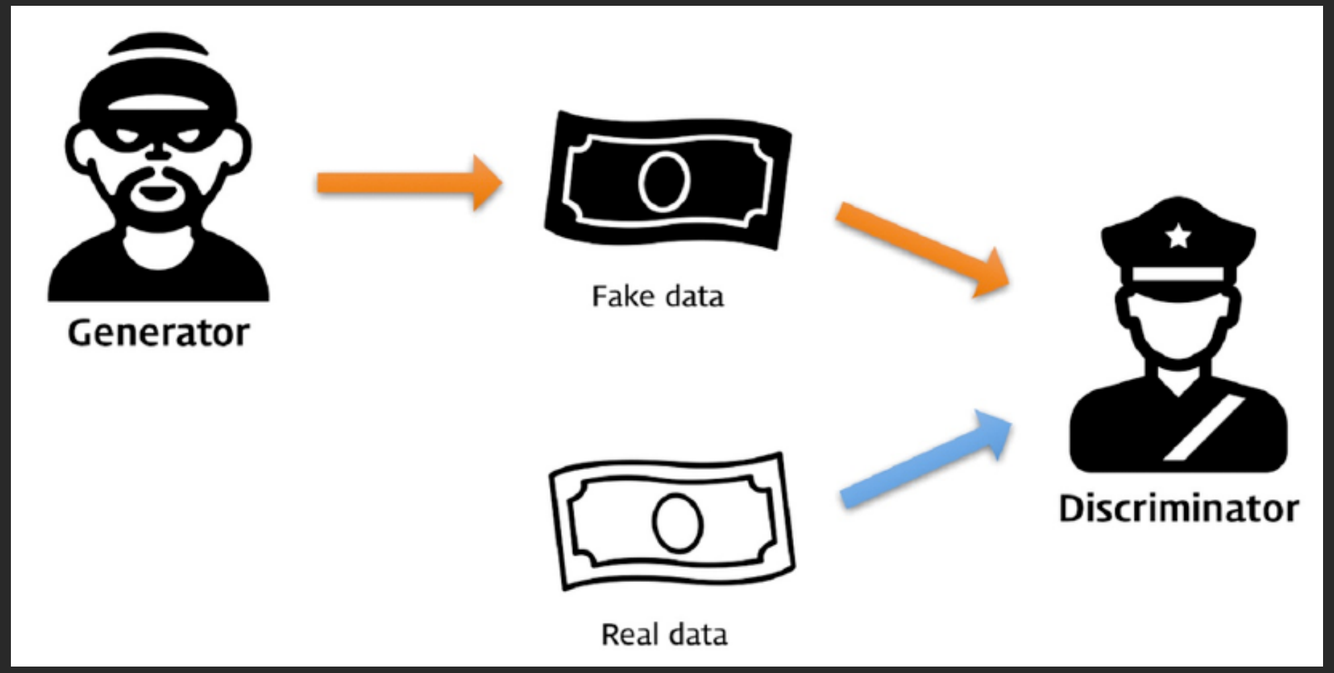
GAN을 이용한 폐 결절 합성 모델

Index

About GAN

PART 1

GAN



생성자(Generator): 위조지폐범

판별자(Discriminator): 경찰

Generator가 위조지폐를 만들고

Discriminator가 위조지폐를 찾아낸다

Generator는 그 과정에서 점점 더

그럴싸한 위조지폐를 만들어낸다

Discriminator가 가짜라고 판별 할 수 없는

위조지폐를 만들어내는 것이 목표

(실제와 가짜의 구분이 불가능 할 때)

GAN의 손실함수

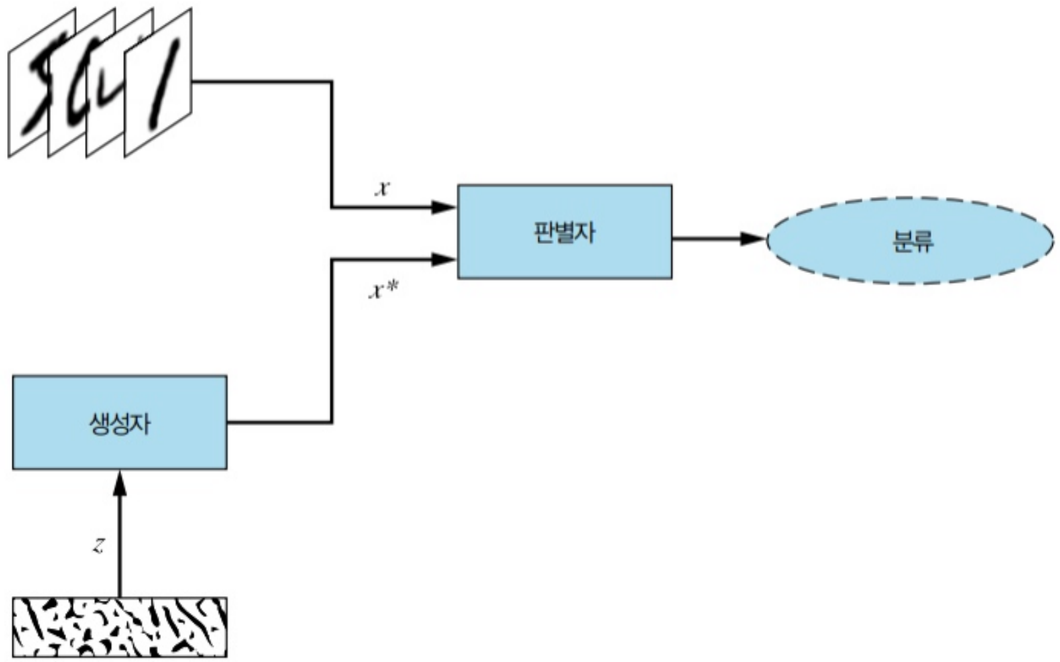
$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim Pdata(x)}[logD(x)] + \mathbb{E}_{z\sim Pz(z)}[\log(1 - D(G(z)))]$$

x: 진짜 이미지
z: 노이즈
G(z): 생성자가 노이즈 벡터를 이용해 이미지를 생성
D(x): 판별자 함수 -> 진짜 이미지면 1 가짜 이미지면 0

값을 G는 낮추고자 하고 D는 높이고자 함.

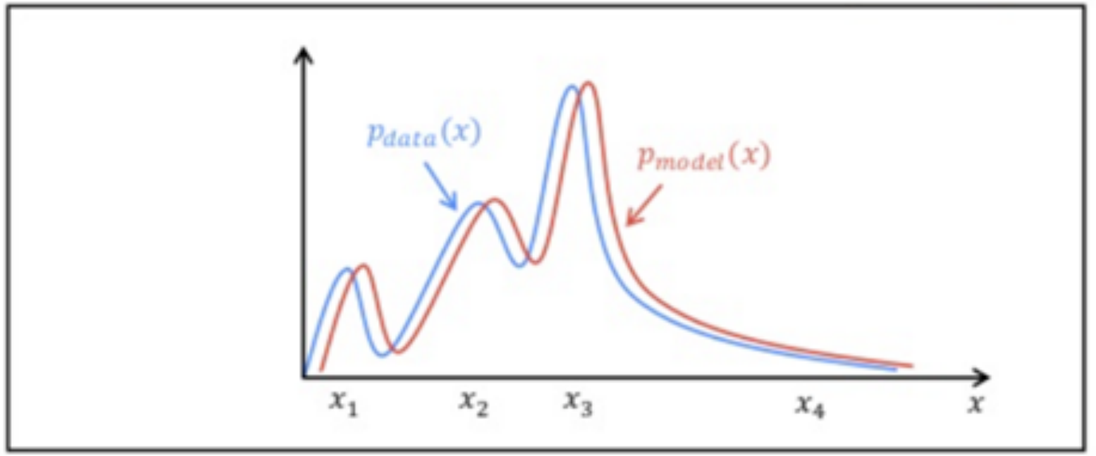
=> 생성자가 실제 데이터와 구별이 안 되는 데이터를 생성 할 때/
판별자가 할 수 있는 최선이 진짜인지 가짜인지 랜덤하게 추측할때(확률이 50:50) 내시균형에 도달했다고 표현

GAN의 구조와 학습 진행 방향



출처: GAN 인 액션

파란색: actual image
빨간색: generated by model



출처: https://roytravel.tistory.com/109

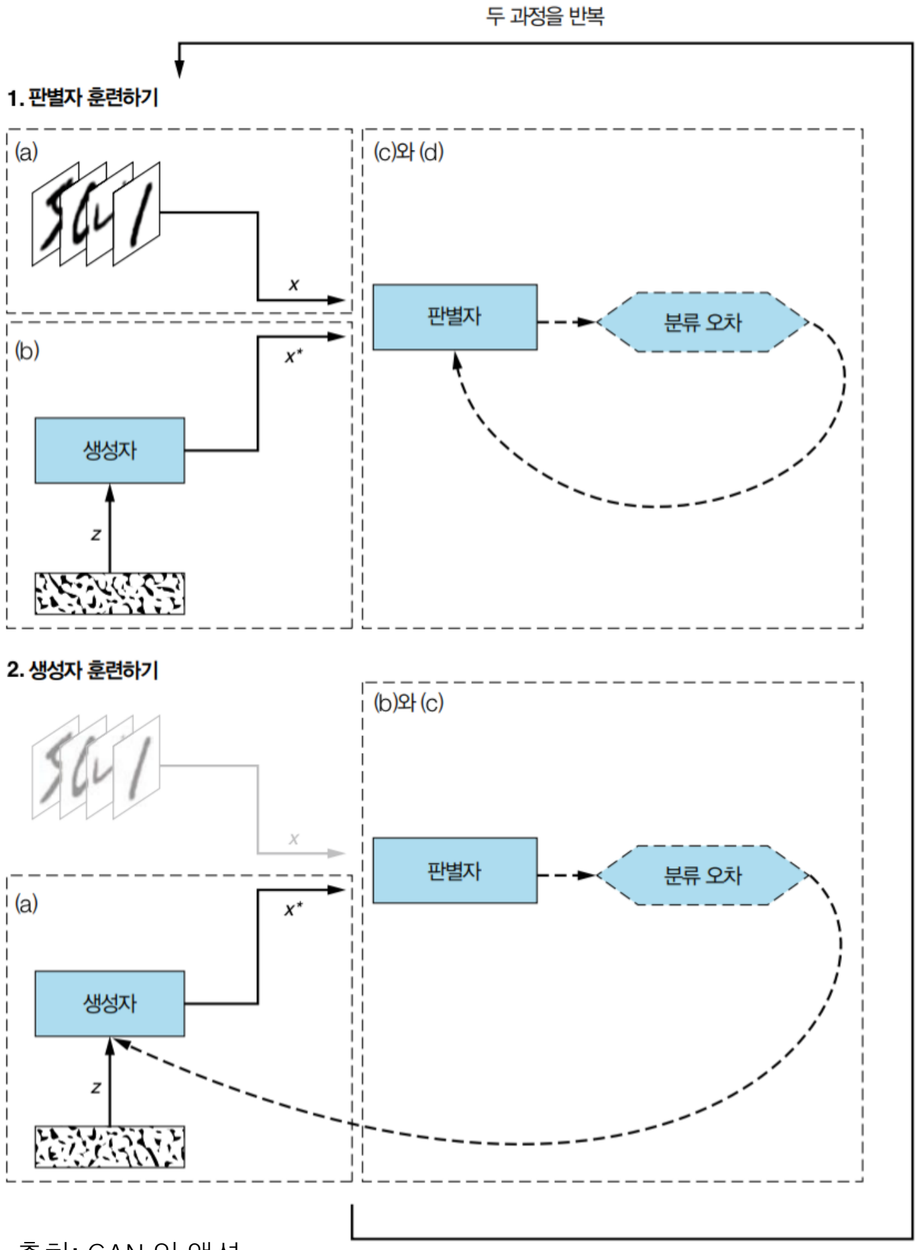
생성자는 진짜 이미지를 통해 학습된 확률분포를 따라 이미지를 생성.
생성자가 만들어낸 이미지와 실제 이미지의 확률분포의 차이가 줄어드는 방향으로 학습이 진행된다.

GAN훈련 알고리즘

1. 판별자 훈련하기
   a) 훈련 데이터셋에서 랜덤하게 진짜 샘플 x선택
   b) 새로운 랜덤한 잡음 벡터 z를 얻어 생성자 네트워크를 이용해 가짜 샘플x*을 합성
   c) 판별자를 이용해 x와 x*을 분류
   d) 분류 오차를 계산하고 back propagation을 통해 파라미터 업데이트하여 분류 오차를 최소화

2. 생성자 훈련하기
   a) 생성자를 통해 새로운 랜덤한 잡음 벡터 z에서 가짜 샘플 x*을 합성
   b) 판별자를 이용해 x*을 분류
   c) 분류 오차를 계산하고 back propagation을 통해 파라미터 업데이트하여 판별자의 오차를 최대화
   => 생성자는 판별자를 속이는 것이 목표이기 때문에 판별자의 오차를 최대화

=> 판별자와 생성자는 번갈아가며 학습을 진행



출처: GAN 인 액션

Contrubutions of SQ-AGGAN

PART 2

Contributions of SQ-AGGAN

1.FEATURE (MALIGNANCY) CONTROL
WITH LOWER COMPUTATION

2. SYNTHESIS QUALITY
+ MINIMIZING THE BACKGROUND CHANGES

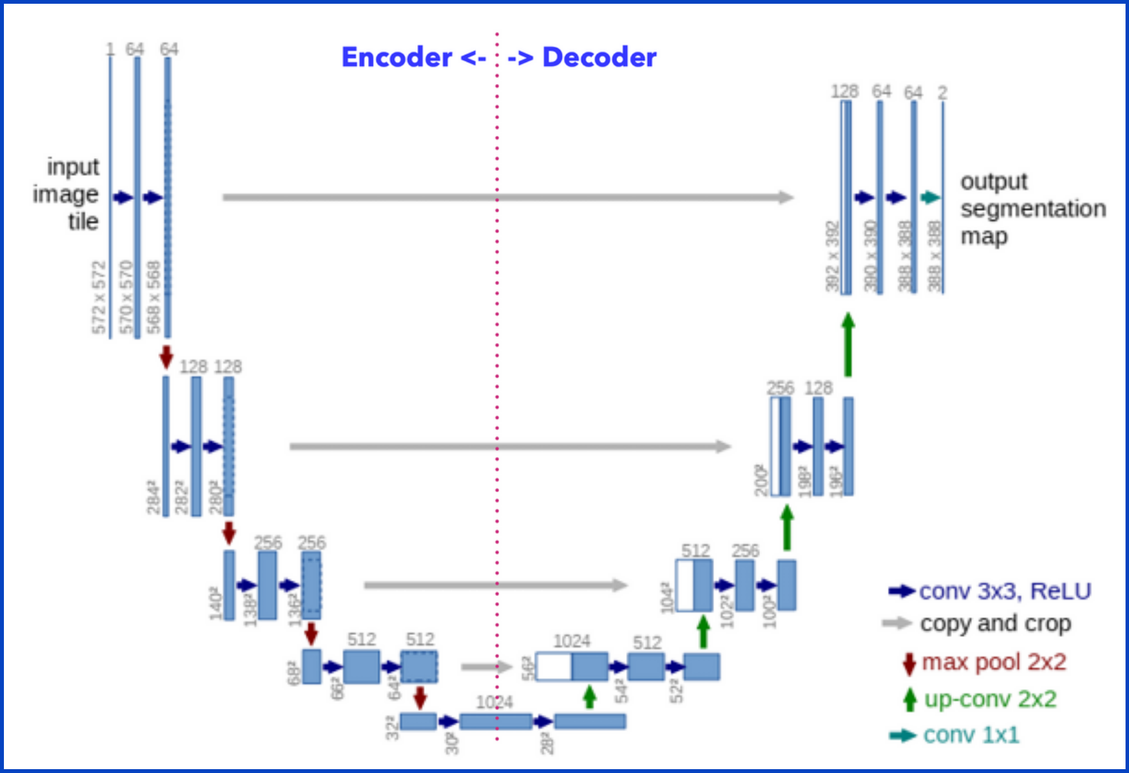
3. RECONSTRUCTION & INJECTION

SQ-AGGAN model architecture

PART 3

About UNET

인코더-디코더 기반 모델



ENCODER

: 입력 이미지의 특징을 포착할 수 있도록 채널의 수를 늘리면서 차원을 축소

DECODER

: 저차원으로 인코딩 된 정보만 이용하여 채널의 수를 줄이고 차원을 늘려서 고차원의 이미지를 복원
-> 이때 차원축소를 거치면서 위치정보를 잃게 되고 이를 해결하기 위해 Unet의 디코더 에서는 같은 크기의 인코딩 단계에서 얻은 고차원의 특징을 디코딩 각 레이어에 합치는 방법을 사용 = 스킵 연결

SQ-AGGAN model Architecture

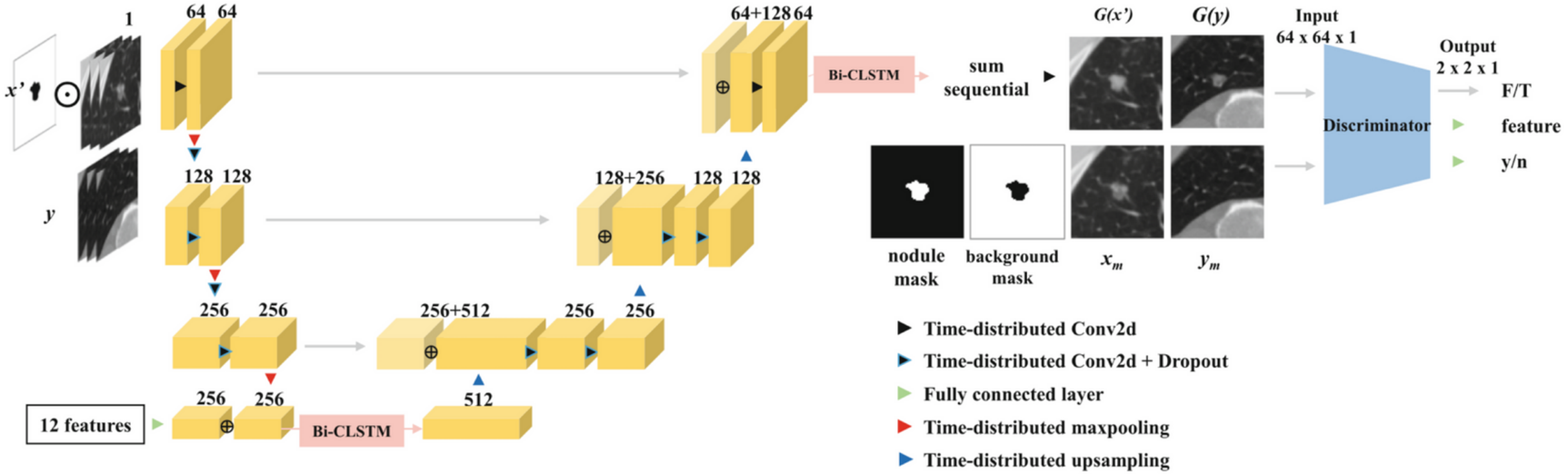


Fig. 1. The structure of the proposed model.

generator relies on U-Net
Encoder: 3 sets of two convolution layers & one max pooling layer
Decoder: 3 sets of One up-sampling & two convolution layers

-> encoder의 끝과 decoder의 끝에 Bi-CLSTM을 배치
: 3D이미지인 CT이미지의 2D slice들의 특징이 유지 됨 (sequential)

x: original nodule image
x': nodule image
Y: nodule free background
G(x'): nodule reconstruction image
G(y): nodule injection image

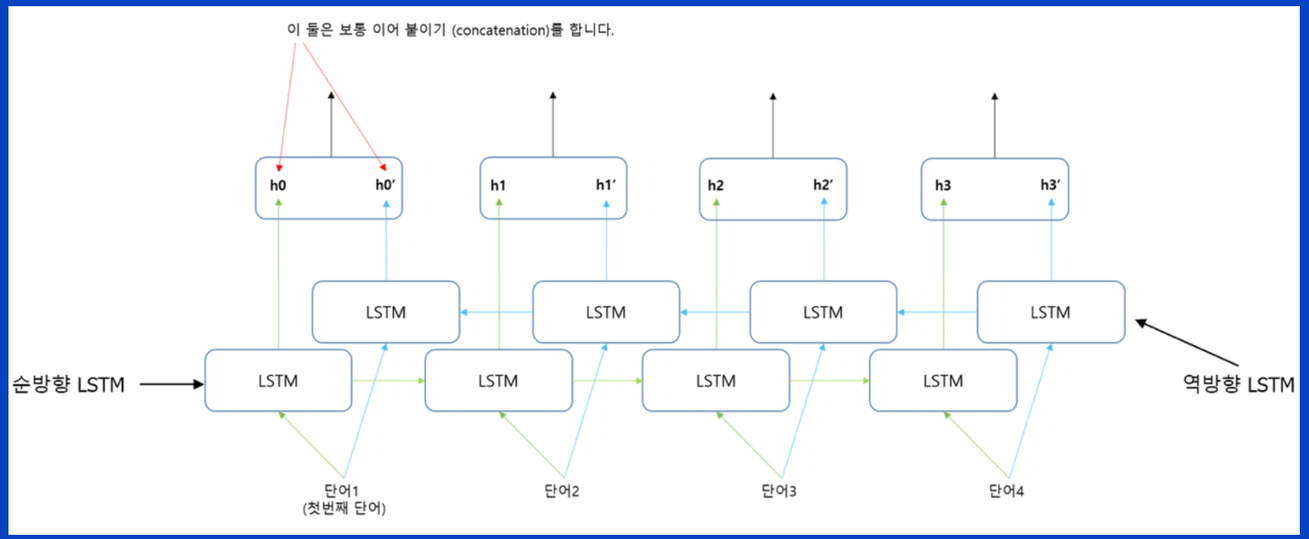
Bi-CLSTM

: Bidirectional Convolution LSTM

LSTM
: 순방향 (왼쪽에서 오른쪽)으로 정보를 추출

Bi_LSTM
: 역방향으로도 정보를 추출
-> 시퀀스 데이터에서 더 많은 정보를 추출할 수 있기 때문에 성능이 더 좋게 나옴

Bi_CLSTM
: Bi_LSTM에 Convolution layer 를 더한 것
-> 사용x시 discontinuity 발생 확인



Generator

[generate]

1. Nodule reconstruction image G(x')

-> can control nodule features + without changing the nodule surroundings

2. Nodule injection image G(y)

-> for data augmentation: discriminator에 넣을 데이터의 imbalance 해결

Discriminator

Discriminator = CNN with four convolution layers

[Input]

1. Pair : G(x') & x -> reconstruction image & original image
2. Pair: G(y) & y -> nodule injection image & nodule-free bg
3. Nodule and background masks of the nodule image

-> used in loss function to minimize the synthesis bg change

[classifies]

1. F/T : real & fake
2. Features of nodule -> enhance controllability of the generator
3. y/n: Existence of nodule -> enhances the continuity of nodule synthesis

Loss Function

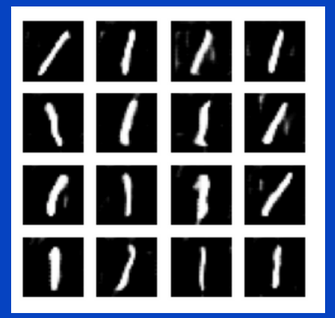
PART 4

BY CAHAYA DEWI

WGAN-GP

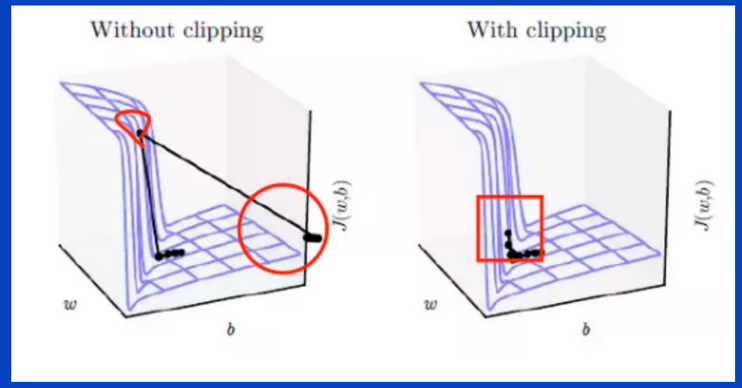
GAN → **WGAN** → **WGAN-GP**

mode of collapse 발생
-> 하나의 condition에 가까운 벡터만 생성

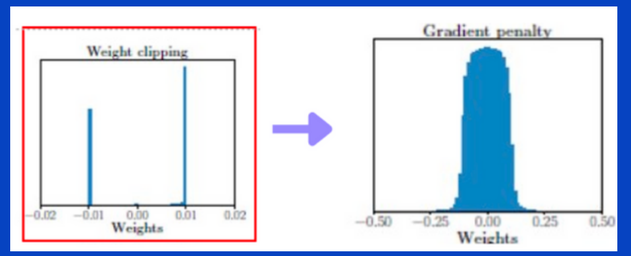


mode collapse를 막기 위함
G와 D의 확률분포 차이를 EMd (= Wasserstein-1) 로 계산
분포가 겹치던 겹치지 않던 간에 일정한 값을 유지
-> 기존의 방법과 달리 vanishing gradient X
-> 학습에 사용하기 쉽다

문제점: weight clipping
-> 말그대로 weight 가 clipping
-> 간격이 너무 크거나 작은 아이들에 대한 가중치 updatae를 최소화 하기 위함
-> 연속성 유지를 가능하게 함



-> WGAN은 weight clipping을 과하게 진행
-> WGAN-GP가 탄생



Loss Function(WGAN-GP)

Discriminator

reconstruct the nodule with feature

inject the nodule with feature

classify the feature

classify the nodule existence
BCE: Binary cross entropy

$$L_{adv,F} = -E_{x_m}[D(x_m)] + E_{x',f}[D(G(x',f))] + \alpha E_{\hat{x}}[\|\nabla D(\hat{x})\|_2 - 1]^2$$
$$L_{adv,B} = -E_{y_m}[D(y_m)] + E_{y,f}[D(G(y,f))] + \alpha E_{\hat{y}}[\|\nabla D(\hat{y})\|_2 - 1]^2$$
$$L_{D,f} = E[\|C_F(x_m) - f\|_2] + E[\|C_F(G(x')) - f\|_2] + E[\|C_F(G(y)) - f\|_2]$$
$$L_{D,e} = L_{BCE}(C_{YN}(x_m), 1) + L_{BCE}(C_{YN}(G(x')), 1) + L_{BCE}(C_{YN}(G(y)), 1) + L_{BCE}(C_{YN}(y_m), 0)$$
$$L_D = L_{adv,F} + L_{adv,B} + \lambda_f L_{D,f} + \lambda_{yn} L_{D,e}$$

$$L = \sum_{i=1}^{n} |y_i - f(x_i)|$$

$$L = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

x_m: sequential 슬라이스 중간에서 추출된 nodule이 있는single slices
y_m: sequential 슬라이스 중간에서 추출된 nodule이 없는single slices
G(x',f): input으로 generated 된 이미지 (reconstruct)
G(y,f): input으로 generated 된 이미지 (injection)
$\hat{x}$: real slice x_m에서 랜덤 샘플링 된 이미지
$\hat{y}$: real slice y_m에서 랜덤 샘플링 된 이미지
α: penalty coefficient
f: nodule's feature vector
C_f: feature classifier
C_{YN}: nodule existence classifier
λ_f, λ_{yn}: control the relative importance of different loss terms
L_f: synthesize the nodule with input feature f using L2 distance

Loss Function(WGAN-GP)

Generator

$$
\begin{aligned}
L_{G,F} &= \frac{N_{total}}{N_n}(E[\|M_n \odot G(x') - M_n \odot x\|_1] + E[\|M_n \odot G(y) - M_n \odot x\|_1]) \\
L_{G,B} &= E[\|M_{bg} \odot G(x') - M_{bg} \odot x\|_1] + E[\|M_{bg} \odot G(y) - M_{bg} \odot y\|_1] \\
L_{G,f} &= E[\|C_F(G(x')) - f\|_2] + E[\|C_F(G(y)) - f\|_2] \\
L_{G,e} &= L_{BCE}(C_{YN}(G(x')), 1) + L_{BCE}(C_{YN}(G(y)), 1) \\
L_G &= -L_{adv,F} - L_{adv,B} + \lambda_f L_{G,f} + \lambda_{yn} L_{G,e} + \lambda_{G,F} L_{G,F} + \lambda_{G,B} L_{G,B}
\end{aligned}
$$

Results

PART 5

Results

Computation Cost

	SQ-AGGAN	MCGAN	T-GAN
Batch size	32	32	15
Parameter amount	62,406,016	55,086,657	408,187,776
GPU training memory (MB)	32,155	18,525	31,483
Training time per epoch (s)	330	257	1,020
Total training time (s)	335,473	1,090,400	1,091,904

Total training time이 가장 짧음 (배치 사이즈가 32임에도 불구하고)
Training time per epoch는 MCGAN에 비해 길다
: SQ-AGGAN은 epoch 1000이 충분하지만 MCGAN은 4700 epoch이 필요

Visual Turing Test

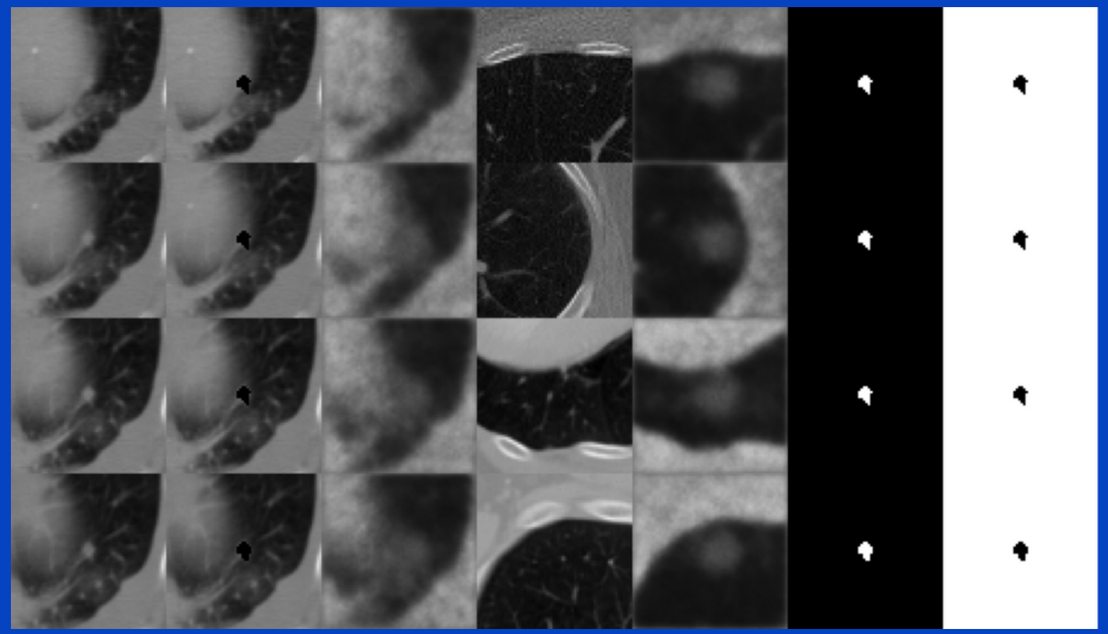
Visual Turing test		LIDC-IRDI	MCGAN	T-GAN	SQ-AGGAN REC	SQ-AGGAN INJ
Radiologist 1	Decision to real	35.00% (TP)	43.33% (FN)	13.33% (FN)	33.33% (FN)	10.00% (FN)
	Decision to synthesis	65.00% (FP)	56.67% (TN)	86.67% (TN)	66.67% (TN)	90.00% (TN)
Radiologist 2	Decision to real	52.63% (TP)	56.67% (FN)	74.07% (FN)	60.00% (FN)	48.28% (FN)
	Decision to synthesis	47.37% (FP)	43.33% (TN)	25.93% (TN)	40.00% (TN)	51.72% (TN)
Mean F1-Score			0.5038	0.5198	0.5086	0.5585

Note: Radiologist 2 did not determine real or synthesis for seven cases.

F1 score가 높다: radiologists are more confuesed when distinguishing between real and synthetic

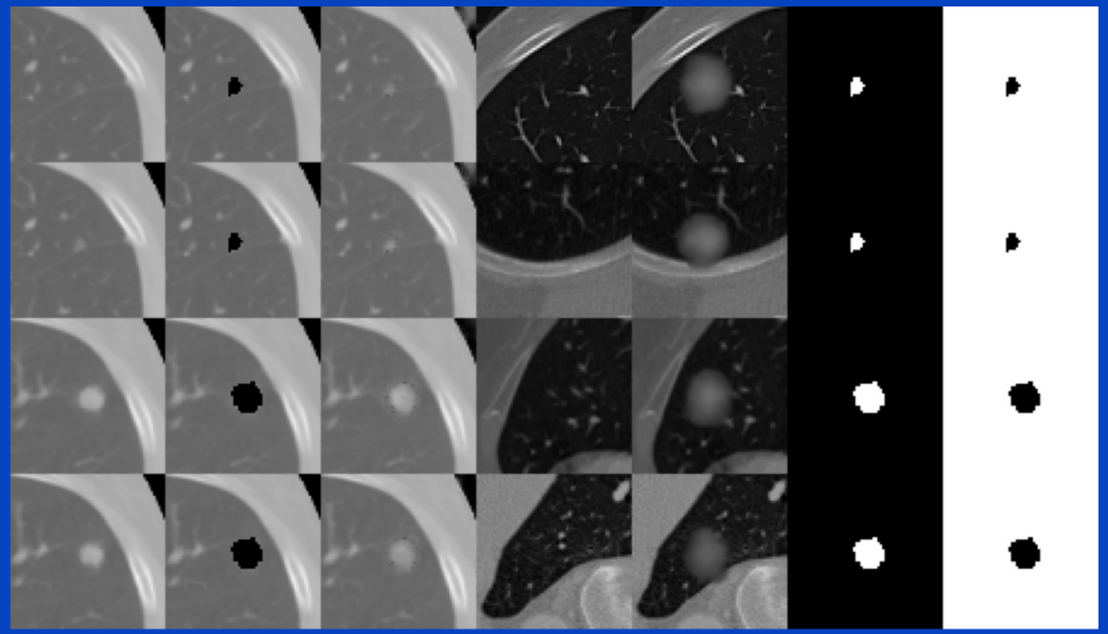
구현 결과

epoch=0



(a) (b) (c) (d) (e) (f) (g)

epoch=99



(a) (b) (c) (d) (e) (f) (g)

(a): original image
(b): Image with nodules removed
(c): reconstructed image
(d): background image
(e): nodule injected image

부딪힌 문제

PART 6

부딪힌 문제 & 해결

메모리 문제 발생

```
RuntimeError: CUDA out of memory. Tried to allocate 32.00 MiB (GPU 0; 23.65 GiB total capacity; 22.08 GiB already all
ocated; 12.62 MiB free; 22.49 GiB reserved in total by PyTorch)
```

1. 배치사이즈 변경
   : 32 -> 8 로 변경
2. Epoch 변경
   : 300에서 100으로 변경
3. 전체에서 일부만 이용
   : 약 1000개의 데이터셋 중 300개만 사용
   (300개만 사용해도 메모리 에러 발생 -> 100개만 이용했으나 에러 발생 + 합성 이미지 품질 저하).
   => 300개의 데이터를 사용해서 에러 발생 전까지 진행

앞으로의 계획

1. Bi-CLSTM 공부
   -> 코드를 조금 더 쉽게 바꿔보기

2. 메모리 부족 문제 해결
   -> 코드 바꿀때 메모리 최소화 하는 방향으로
   https://gldmg.tistory.com/108

3. 다른 데이터셋에도 모델 적용해보기

4. 다른 모델 공부

Thank You