Difference GAN

의료영상시스템 및 인공지능응용

Sunghyun Ahn

skdyonsei.ac.kr

<2023/11/27>

1 DFGAN

Denoising Image

- **f_nd:** 일반 선량 -> 높은 선량의 방사선을 사용하여 얻은 이미지, 이미지의 대비가 높고 상세하게 나타남 (normal)
- **f_qd:** 1/4 선량 -> 낮은 선량의 방사선을 사용하여 얻은 이미지, 이미지의 대비가 낮고, 세부 정보가 덜 표현됨 (noise)

→ Denoising이 필요함

Dataset

```
train_file_path = 'data/trainset.npy' # 2100장
val_file_path = 'data/validset.npy' # 300장
test_file_path = 'data/testset.npy' # 600장
```

[2] ✓ 0.0s

Load dataset

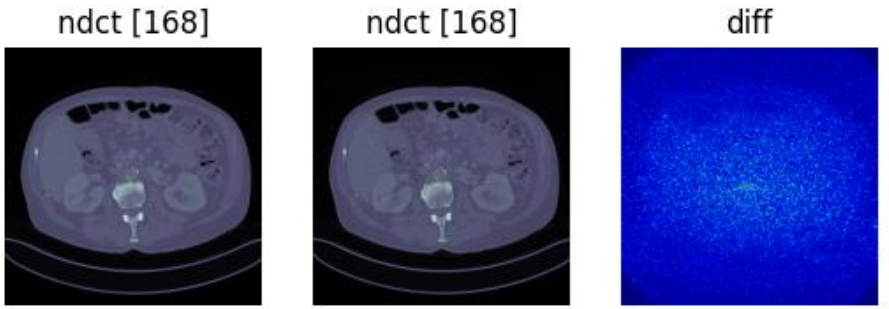
```
loaded = np.load(test_file_path, allow_pickle=True)
```

[3] ✓ 0.3s

```
data = loaded.tolist()
d = dict(data)
print(d.keys())
```

[4] ✓ 0.0s

```
dict_keys(['f_nd', 'f_qd'])
```



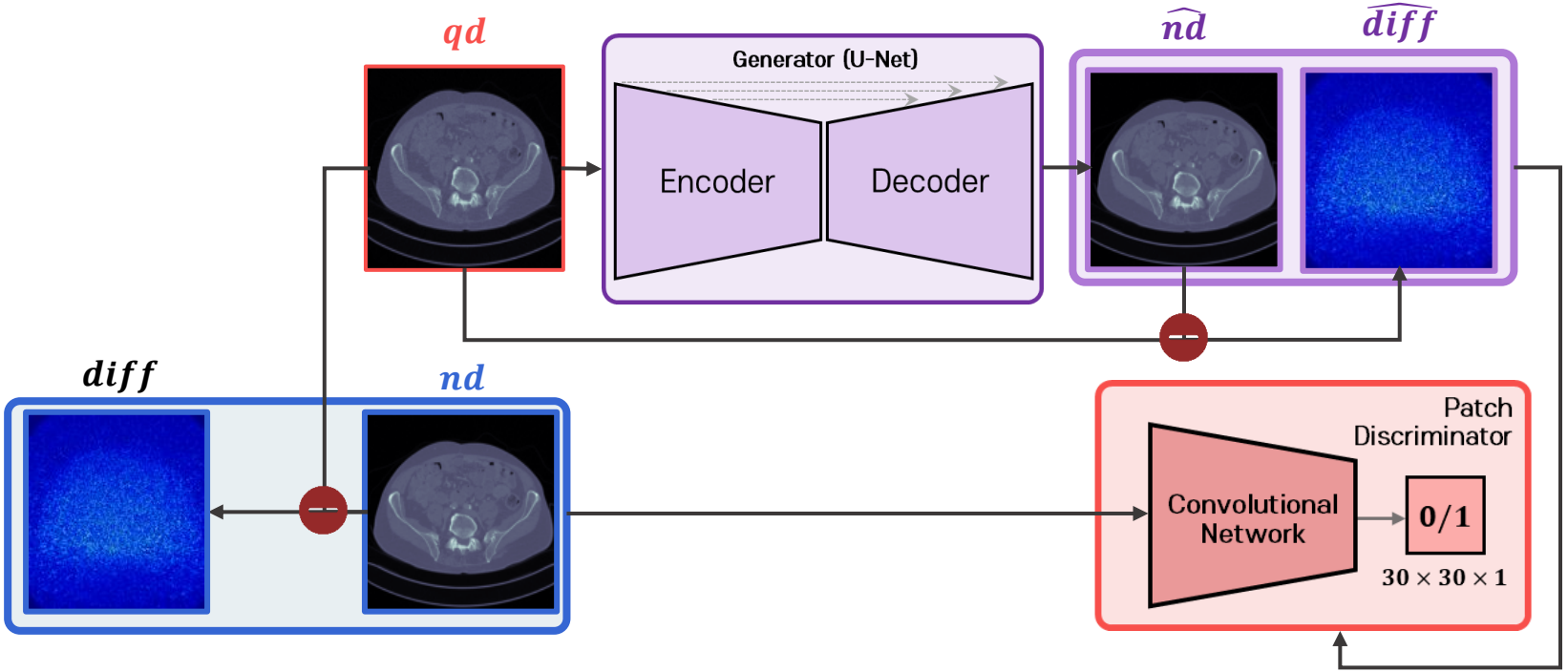

```
psnr: 31.599647723805813
```

```
ssim: 0.7459428800844302
```

1 DFGAN

Difference GAN *[DFGAN]*

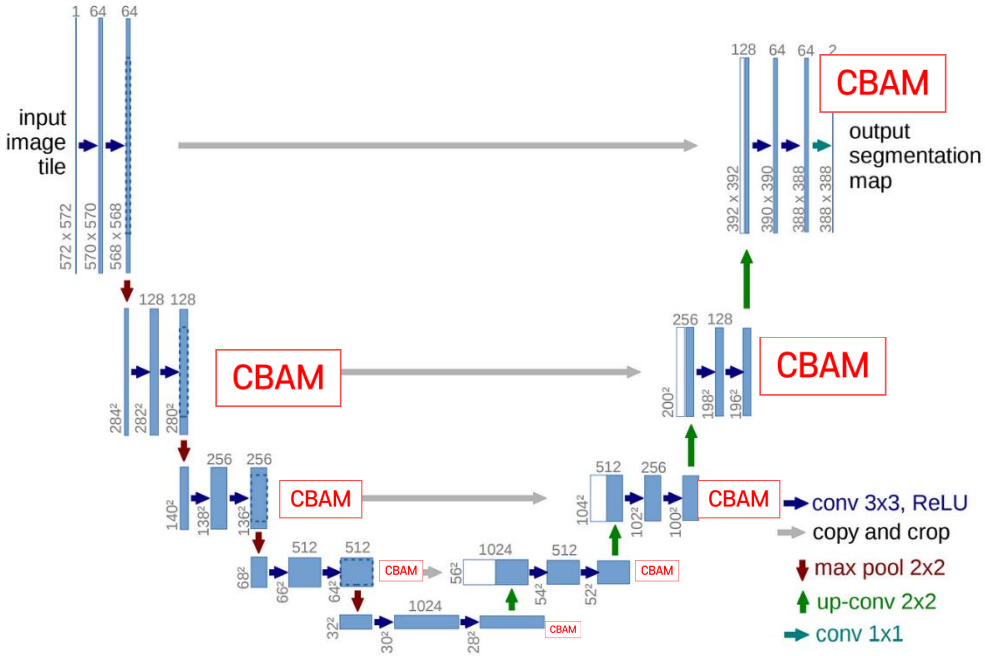
- Target처럼 Denoising되기 위해 Difference map을 같이 적대적 학습하는 GAN
- Input과 Target이 유사하므로 output도 Input처럼 나올 수 있지만, diff까지 고려하면 Target에 더 가깝게 만들어 지는 효과



1 DFGAN

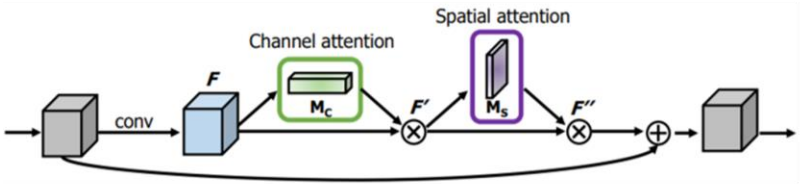
Difference GAN + CBAM *[AttnDFGAN]*

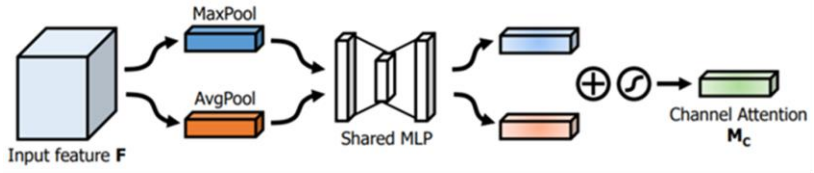
Generator에서 Target을 생성하기 위해 필요한 feature를 집중함 (Resolution view, Channel view)

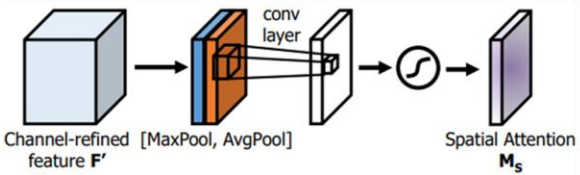


1 DFGAN

Convolutional Block Attention Module







CBAM 모듈 (ECCV 2018)

$$\mathbf{F}' = \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F},$$
$$\mathbf{F}'' = \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}',$$

채널 어텐션 (Channel Attention)

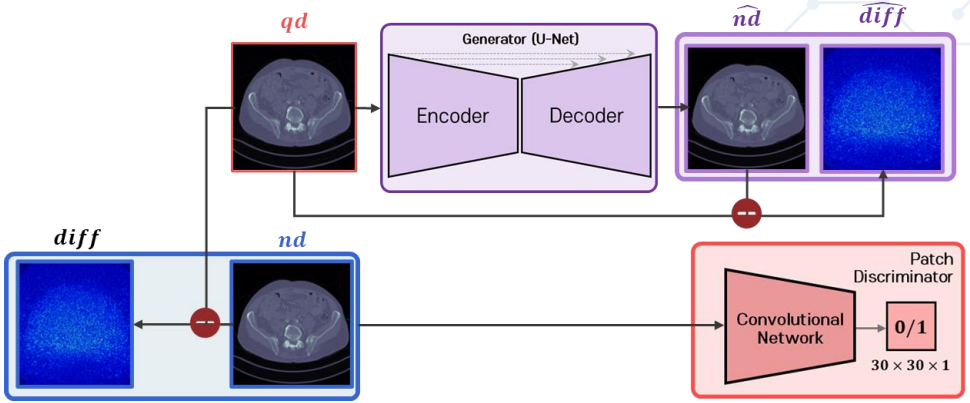
$$\begin{aligned}\mathbf{M_c}(\mathbf{F}) &= \sigma(MLP(AvgPool(\mathbf{F})) + MLP(MaxPool(\mathbf{F}))) \\ &= \sigma(\mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{avg}})) + \mathbf{W_1}(\mathbf{W_0}(\mathbf{F^c_{max}}))),\end{aligned}$$

공간 어텐션 (Spatial Attention)

$$\begin{aligned}\mathbf{M_s}(\mathbf{F}) &= \sigma(f^{7\times7}([AvgPool(\mathbf{F}); MaxPool(\mathbf{F})])) \\ &= \sigma(f^{7\times7}([\mathbf{F^s_{avg}}; \mathbf{F^s_{max}}])),\end{aligned}$$

1 DFGAN

Difference GAN Loss



$$L_{int-1}(\hat{I}, I) = \left\| \hat{I} - I \right\|_2^2$$

$$L_{int-2}(\hat{I}, I) = \left\| \hat{I} - I \right\|_2^2$$

$$L_{adv}^{G}(\hat{I}) = \sum_{i,j} \frac{1}{2} L_{MSE}\left(D(\hat{I})_{i,j}, 1\right)$$

$$L_{adv}^{D}(\hat{I}, I) = \sum_{i,j} \frac{1}{2} L_{MSE}\left(D(I)_{i,j}, 1\right) + \sum_{i,j} \frac{1}{2} L_{MSE}\left(D(\hat{I})_{i,j}, 0\right)$$

$$coefs = [1, 1, 0.05]$$

1 DFGAN

$$PSNR(I, \hat{I}) = 10 \log_{10} \frac{[\max_{\hat{I}}]^2}{\frac{1}{N} \sum_{i=0}^{N} (I_i - \hat{I}_i)^2}$$

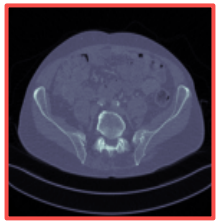
AttnDFGAN Results *[PSNR]*

(1) input(qd) - target(nd)

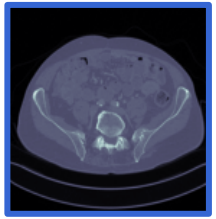
```
psnr_sum = 0

for i, data in enumerate(test_loader, 0):
    target, input = data
    target = target.to(device)
    input = input.to(device)
    psnr_sum += psnr_error(input, target)

psnr_avg = psnr_sum / len(test_loader)
print('psnr([qd, nd]:', psnr_avg)
```

```
psnr([qd, nd]: 33.78583356698258
```

qd



nd

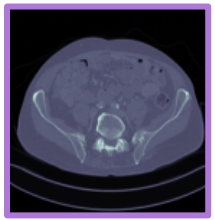


(2) output(pred) - target(nd)

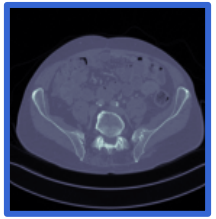
```
psnr_sum2 = 0

for i, data in enumerate(test_loader, 0):
    target, input = data
    target = target.to(device)
    input = input.to(device)

    out = generator(input)
    psnr_sum2 += psnr_error(out.cpu(), target.cpu())

psnr_avg2 = psnr_sum2 / len(test_loader)
print('psnr([PRED, nd]:', psnr_avg2)
```

```
/home/sha/anaconda3/envs/dvaa/lib/python3.8/site-packages/t
  warnings.warn("nn.functional.sigmoid is deprecated. Use t
psnr([PRED, nd]: 36.17765081034578
```

$\widehat{nd}$



nd



1 DFGAN

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

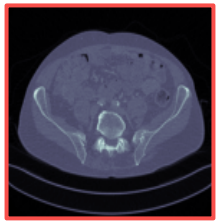
AttnDFGAN Results *[SSIM]*

(1) input(qd) - target(nd)

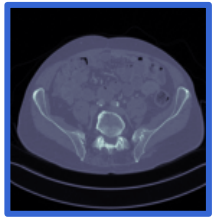
```
ssim_sum = 0

for i, data in enumerate(test_loader, 0):
    target, input = data
    target = target.to(device)
    input = input.to(device)
    ssim_sum += ssim_error(input, target)

ssim_avg = ssim_sum / len(test_loader)
print('ssim([qd, nd]:', ssim_avg)
```

```
ssim([qd, nd]: 0.8532261348331079
```

qd



nd

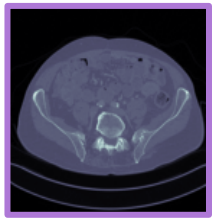


(2) output(pred) - target(nd)

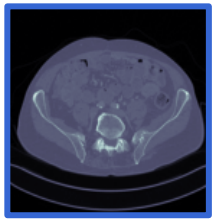
```
ssim_sum2 = 0

for i, data in enumerate(test_loader, 0):
    target, input = data
    target = target.to(device)
    input = input.to(device)

    out = generator(input)
    ssim_sum2 += ssim_error(out, target)

ssim_avg2 = ssim_sum2 / len(test_loader)
print('ssim([PRED, nd]:', ssim_avg2)
```

```
ssim([PRED, nd]: 0.9224878011992047
```

$\widehat{nd}$



nd



1 DFGAN

DFGAN Results *[PSNR, SSIM]*

	PSNR	SSIM
ORG	33.78	0.8532
DFGAN	37.43	0.9225
AttnDFGAN	36.17	0.9224

(2) output(pred) - target(nd)

```
psnr_sum2 = 0

for i, data in enumerate(test_loader, 0):
    target, input = data
    target = target.to(device)
    input = input.to(device)

    out = generator(input)
    psnr_sum2 += psnr_error(out.cpu(), target.cpu())

psnr_avg2 = psnr_sum2 / len(test_loader)
print('psnr([PRED, nd]:', psnr_avg2)
```

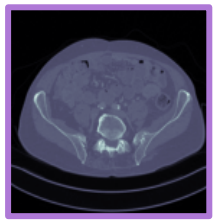
```
psnr([PRED, nd]: 37.43685611713474
```

(2) output(pred) - target(nd)

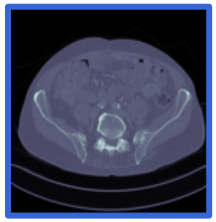
```
ssim_sum2 = 0

for i, data in enumerate(test_loader, 0):
    target, input = data
    target = target.to(device)
    input = input.to(device)

    out = generator(input)
    ssim_sum2 += ssim_error(out, target)

ssim_avg2 = ssim_sum2 / len(test_loader)
print('ssim([PRED, nd]:', ssim_avg2)
```

```
ssim([PRED, nd]: 0.9225143916558117
```

$\widehat{nd}$



nd

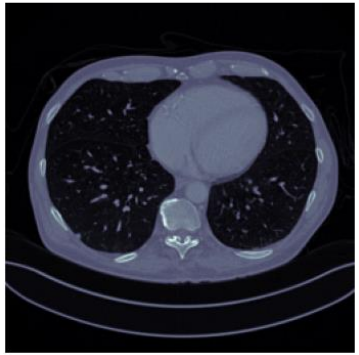


1 DFGAN

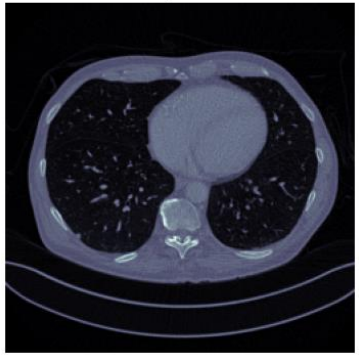
DFGAN Visualization

정량, 정성적으로 유의미하게 이미지의 품질이 좋아진 것을 확인할 수 있음

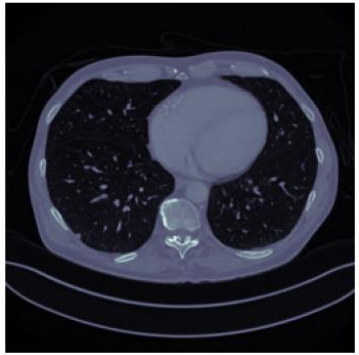
ndct



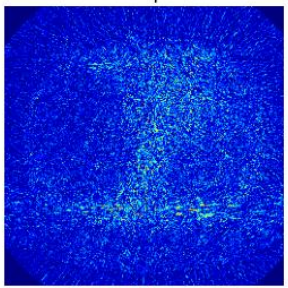
qdct



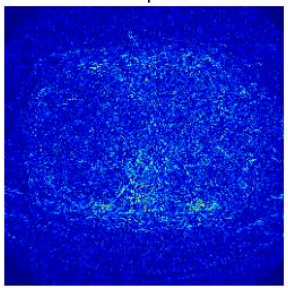
pred



ndct-qdct



ndct-pred



```
psnr_qdct: 35.676940731511124
ssim_qdct: 0.916091536003151

psnr_pred: 38.82638464996408
ssim_ndct: 0.9588744684542974
```

Thank You

의료영상시스템 및 인공지능응용

Sunghyun Ahn

skdyonsei.ac.kr

<2023/11/27>