DL Seminar

Pix2Pix

Image-to-Image Translation with Conditional Adversarial Networks

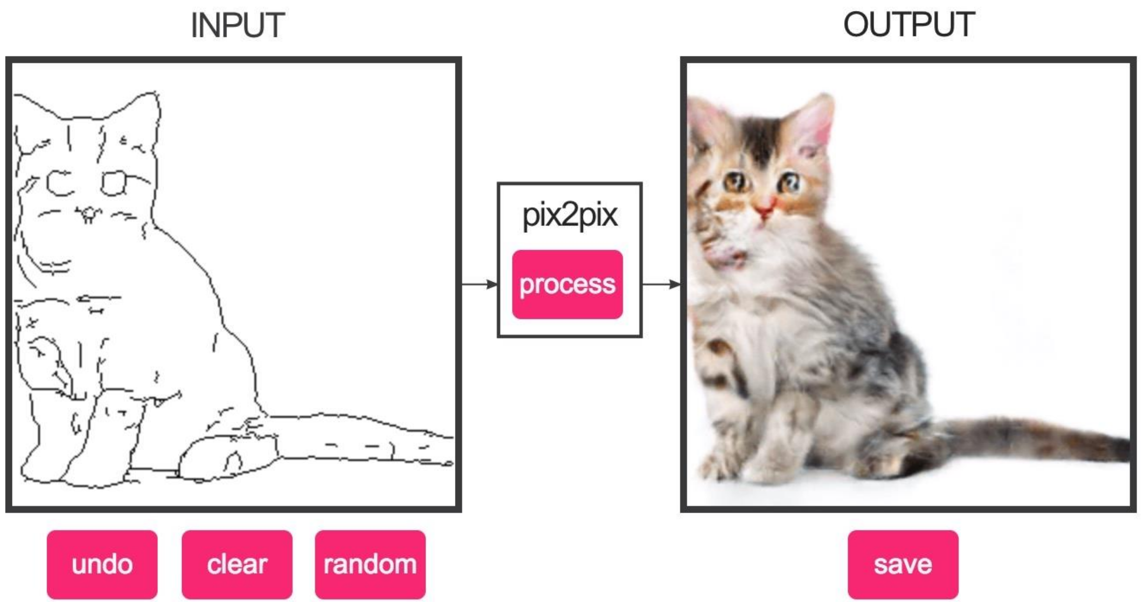
한양대학교
HANYANG UNIVERSITY

인공지능 연구실
김지성

Index

Introduction

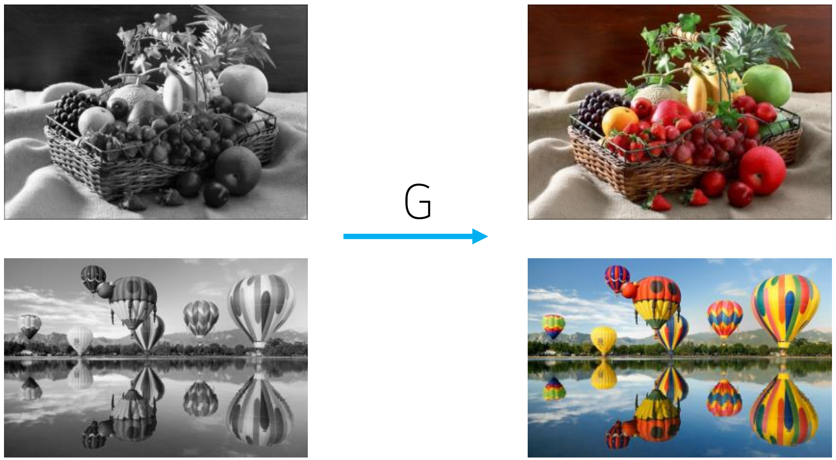
Pix2Pix



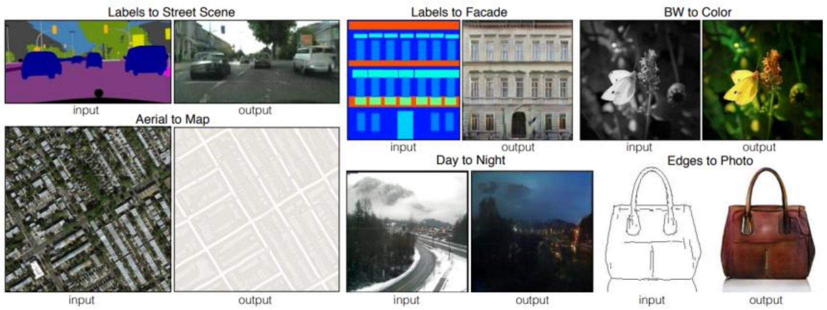
Interactive Demo
https://affinelayer.com/pixsrv/

Introduction

Pix2Pix



- Input, Output : Image
- Self-Supervised
- Loss : Minimize the difference between output G(x) and ground truth y



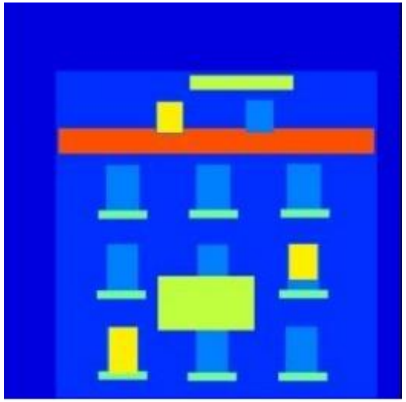
Introduction

L1 loss

Loss: Minimize the difference between output *G(x) and the ground truth y*

$$\sum_{(x,y)} \|y - G(x)\|_1$$

완벽한 답을 찾으려 하기보다, 안전한 값에 도달하려 하기때문에 Blurry한 결과가 나옴



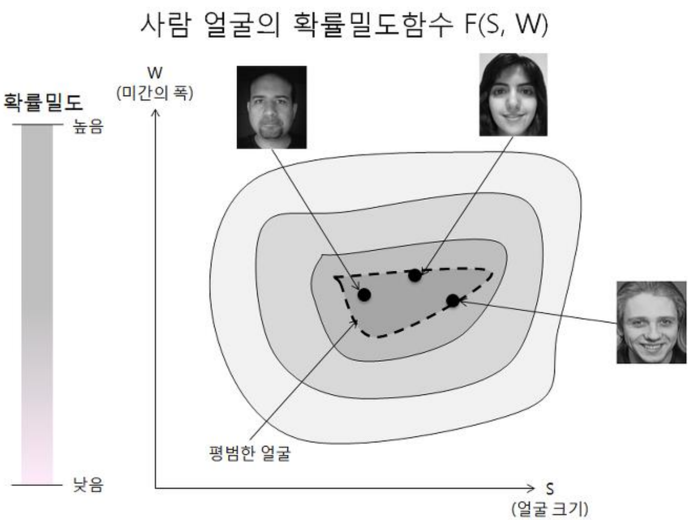
Input

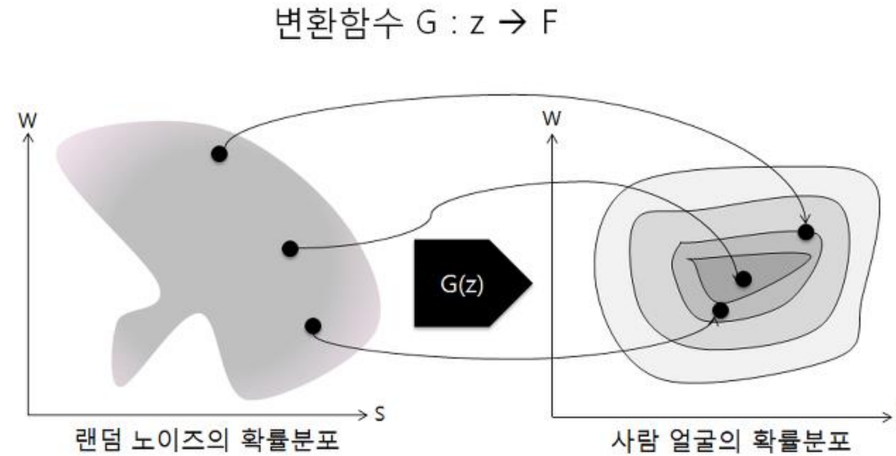
Output

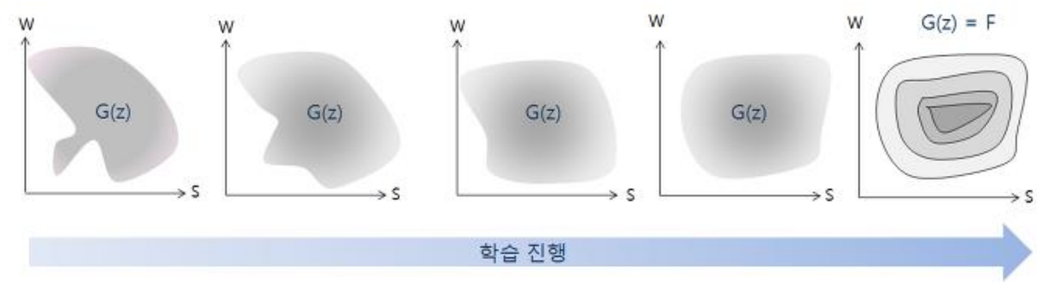
Ground Truth

Method

GAN Training







Method

Object Function – GAN

GAN

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

Method

Object Function – GAN

z가 가우시안 확률분포를 따를
때의 logD(1-D(G(z))의 평균

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

X가 Pdata라는 확률분포를
따를 때의 logD(x)의 평균

Method

Object Function – GAN

Maximizing Likelihood is Minimizing Cross-Entropy

$$Likelihood = \prod_{n=1}^{N} P(t_n|x_n, w) = \prod_{n=1}^{N} \begin{cases} P(t_n = 1|x_n) \ \mathbf{when}(t_n = 1) \\ 1 - P(t_n = 1|x_n) \ \mathbf{when}(t_n = 0) \end{cases}$$

$\downarrow$ Bernoulli 분포(t = 0 or 1)로 표현

$$Likelihood = \prod_{n=1}^{N} {y_n}^{t_n} (1 - y_n)^{1-t_n}$$

$\downarrow$ Log

$$LogLikelihood = \sum_{n=1}^{N} t_n log(y_n) + (1 - t_n) log(1 - y_n)$$

Method

Object Function – GAN

D가 1인 경우의 우도 D가 0인 경우의 우도

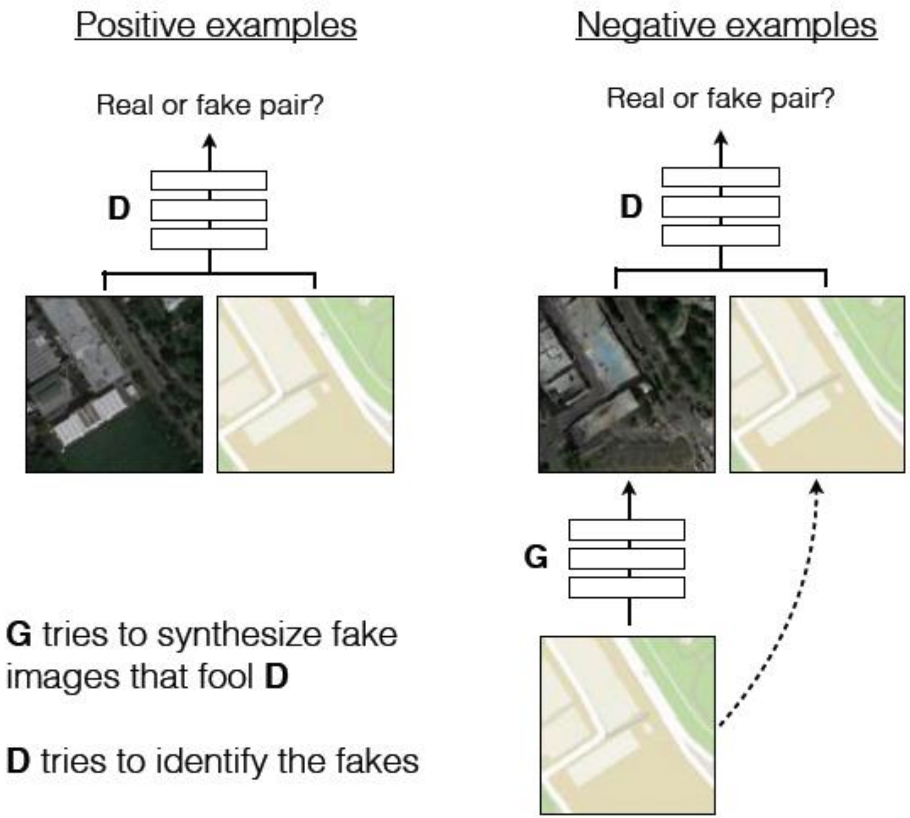
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\underbrace{[logD(x)]}_{tn = 1} + \mathbb{E}_{z\sim p_z(z)}\underbrace{[log(1 - D(G(z)))]}_{tn = 0}$$

$$LogLikelihood = \sum_{n=1}^{N} t_n log(y_n) + (1 - t_n) log(1 - y_n)$$

우도 : 특정 사건이 일어날 가능성을 비교하기위한 척도

Method

Object Function – Adversarial



Method

Object Function – Adversarial

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))], \quad (1)$$

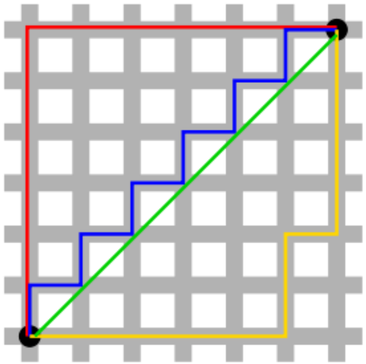
$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{y}[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x, z))]. \quad (2)$$

x : 관찰한 이미지
y : 출력된 이미지
z : 랜덤 노이즈 벡터

GAN : 랜덤 노이즈 벡터 z로부터 이미지 y를 출력하는 (G : z -> y) 를 학습
cGAN : 관찰한 이미지 x와 랜덤 노이즈 벡터 z로부터 이미지 y를 출력하는 (G : { x, z } -> y) 를 학습

Method

Object Function - Regularization



$\|x\|_1 = \sum_{i=1}^{n} |x_i| = l_1 norm$ = 맨하튼 노름

$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} = l_2 norm$ = 유클리드 노름

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1]. \quad (3)$$

이전 연구에서 GAN Object Function에 L2 distance loss를 추가하면 D를 잘 속이면서도 ground truth에 가까운 이미지를 만들어낸다는 실험결과가 있으나, 본 논문의 Translating Model에서는 L1 Distance가 덜 Blurry 한 이미지를 생성해 냄

Method

Object Function

$$Object\ Function = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G). \quad (4)$$

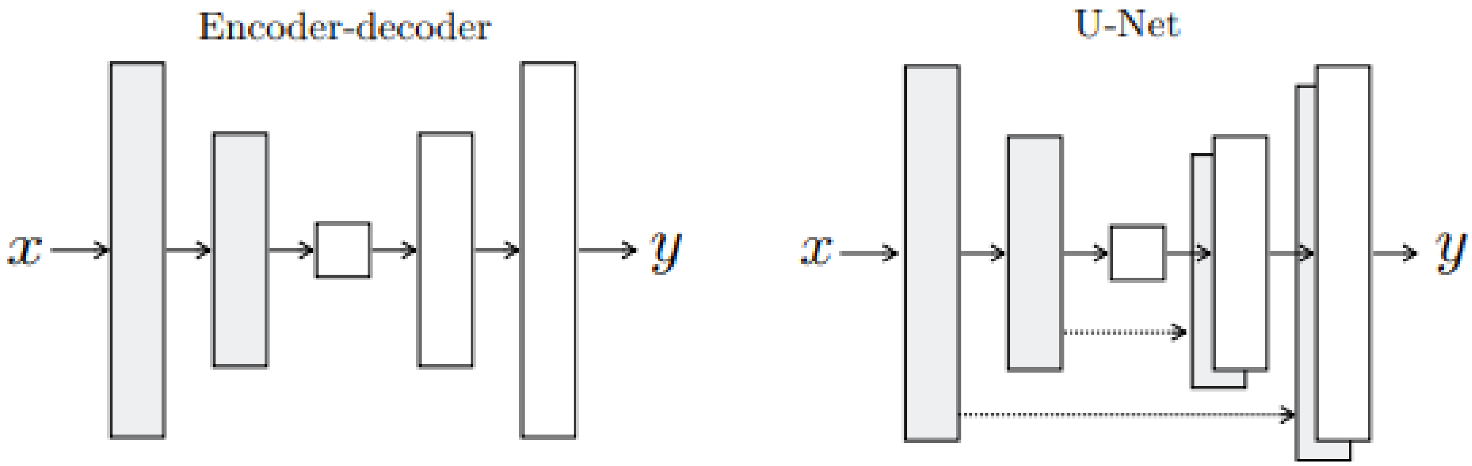
$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))], \quad (1)$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]. \quad (3)$$

G와 D의 Object가 다름
G의 입장 : D를 잘 속이고 싶다!
D의 입장 : G를 잘 잡고 싶다!

Method

Generator with Skips



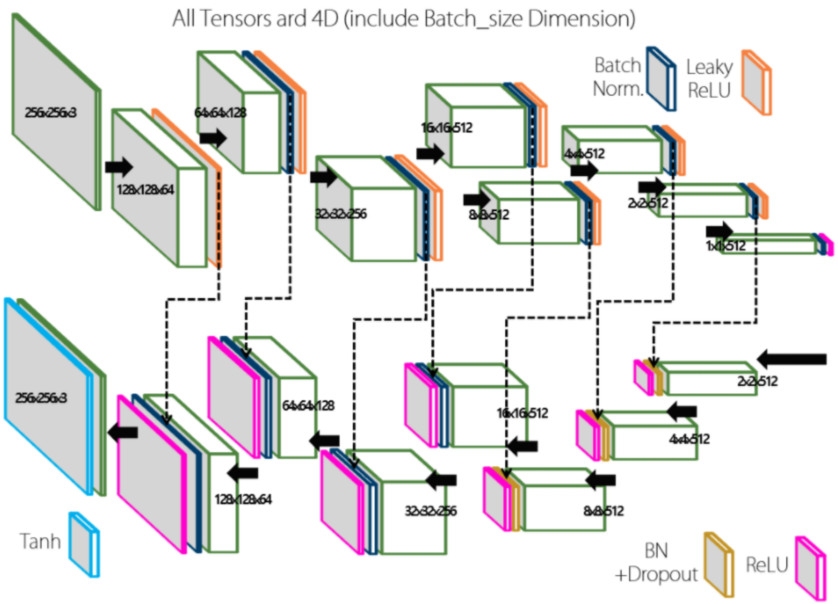
U-Net은 Encoder-Decoder 구조에 Skip-Connection을 연결한 구조

- 처음 디테일이 마지막 레이어까지 고속도로처럼 전달되어 디테일이 많이 간직된다
- 두 데이터셋이 어느정도 비슷한 경우에 Skip-Connection을 과도하게 사용하는 경향을 보임
- Output으로 나온 결과의 이미지가 원본과 비교했을 때 큰 변화를 기대하기 힘듬

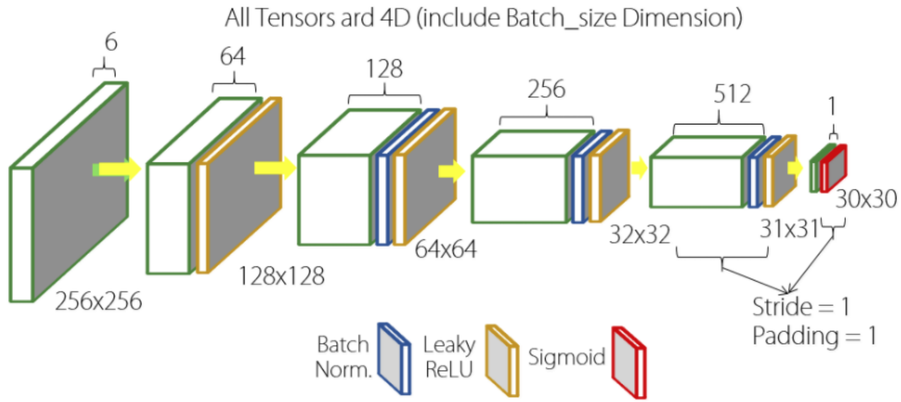
Experiments

CNN Model

3.2 Generator Networks (network.py)



3.3 Discriminator Networks (network.py)



Experiments

CNN Model

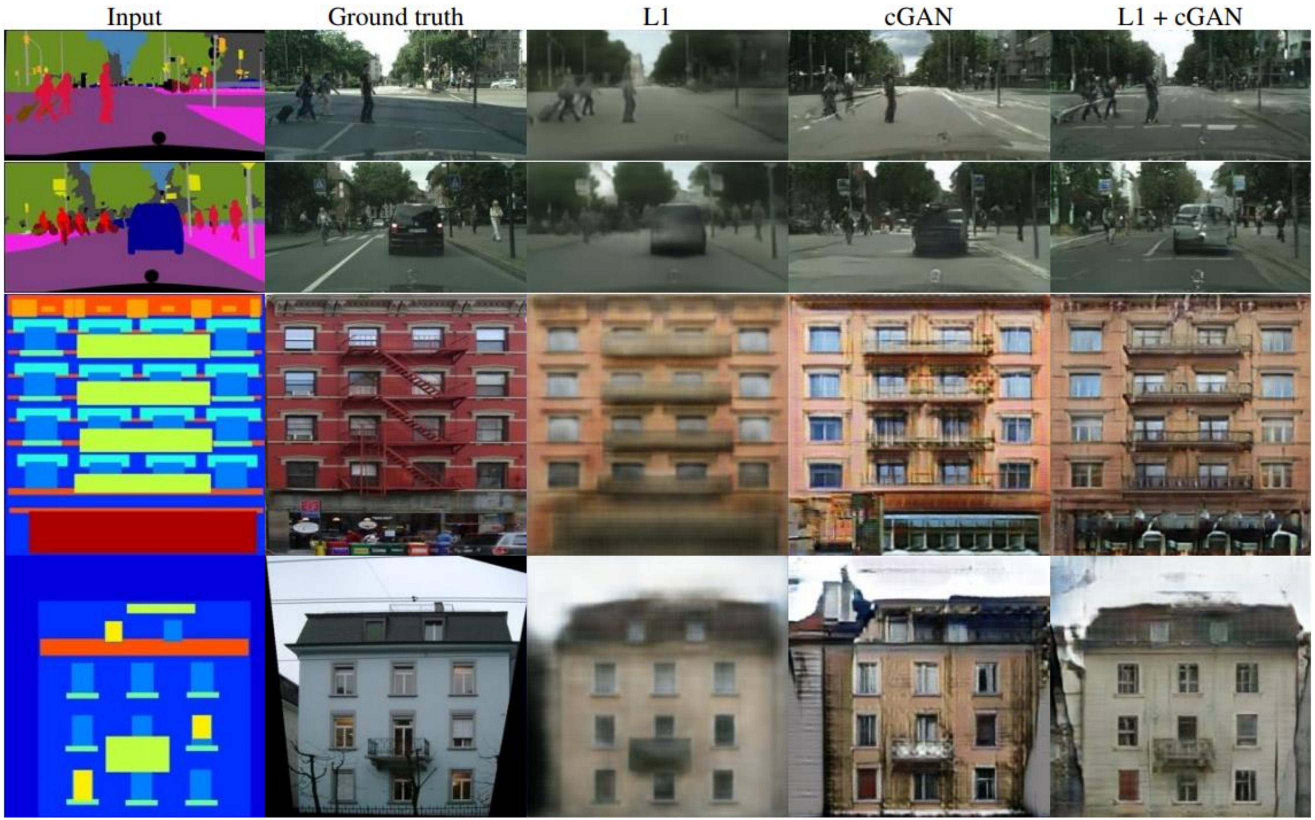
Pix2pix.py – Train D

```
...
discriminator.zero_grad()

# Forward
real_A = to_variable(input_A)
fake_B = generator(real_A)
real_B = to_variable(input_B)

pred_fake = discriminator(real_A, fake_B)
pred_real = discriminator(real_A, real_B)

# Loss (CriterionGAN: Cross Entropy)
#   Fake-Fake Loss
loss_D_fake = GAN_Loss(pred_fake, False, criterionGAN)

#   Real-Real Loss
loss_D_real = GAN_Loss(pred_real, True, criterionGAN)

loss_D = (loss_D_fake + loss_D_real) * 0.5

# Optimize
loss_D.backward(retain_graph=True)
d_optimizer.step()
```

Pix2pix.py – Train G

```
generator.zero_grad()

# Forward
pred_fake = discriminator(real_A, fake_B)

# Loss
#   Fake-Real Loss
loss_G_GAN = GAN_Loss(pred_fake, True, criterionGAN)
#   Recon Loss
loss_G_L1 = criterionL1(fake_B, real_B)

loss_G = loss_G_GAN + loss_G_L1 * args.lambda_A

# Optimize
loss_G.backward()
g_optimizer.step()
...
```

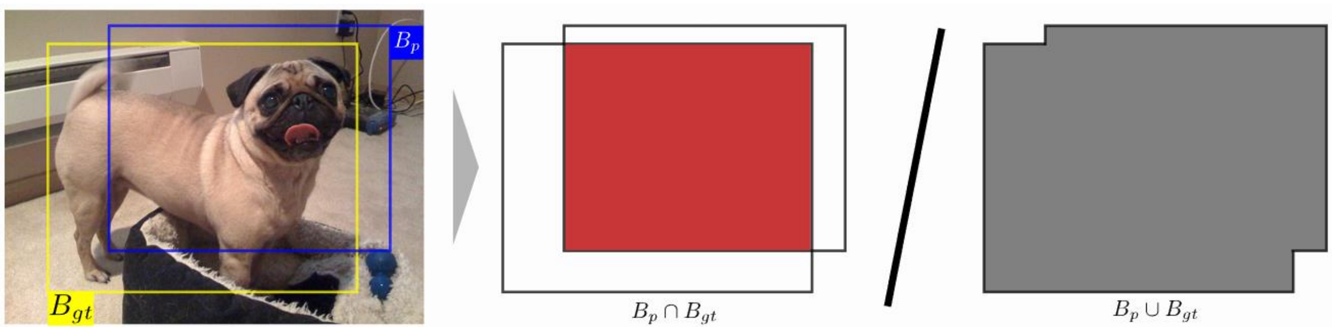
Experiments



Experiments

intersection over union

$$B_p \text{와 } B_{gt} \text{ 의 IOU} = \frac{B_p \cap B_{gt} \text{ 영역 넓이}}{B_p \cup B_{gt} \text{ 영역 넓이}}$$



B_p와 B_{gt} 간의 IOU 계산

Experiments

Loss	Per-pixel acc.	Per-class acc.	Class IOU
L1	0.42	0.15	0.11
GAN	0.22	0.05	0.01
cGAN	0.57	0.22	0.16
L1+GAN	0.64	0.20	0.15
L1+cGAN	**0.66**	**0.23**	**0.17**
Ground truth	0.80	0.26	0.21

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Encoder-decoder (L1)	0.35	0.12	0.08
Encoder-decoder (L1+cGAN)	0.29	0.09	0.05
U-net (L1)	0.48	0.18	0.13
U-net (L1+cGAN)	**0.55**	**0.20**	**0.14**

Experiments

Discriminator receptive field	Per-pixel acc.	Per-class acc.	Class IOU
1×1	0.39	0.15	0.10
16×16	0.65	0.21	**0.17**
70×70	**0.66**	**0.23**	**0.17**
286×286	0.42	0.16	0.11

1x1 : pixel
16x16 : patch
70x70 : patch
286x286 : image

- 패치의 크기 N을 1×1의 PixelGAN에서 286×286의 ImageGAN까지 늘려가며 효과를 측정
- PixelGAN은 샤프니스에 효과가 없었으나 colorfullness의 효과를 증가시켰다.
- 16×16 PatchGAN은 샤프니스가 상당히 올랐으며 FCN-score에서도 높은 점수를 얻었으나 여전히 아티팩트가 나타났다.
- 70×70 PatchGAN은 아티팩트가 완화되었고, 좀 더 나은 점수를 보였다.
- 286×286 ImageGAN은 더 이상 품질이 향상되지 않았으며, 오히려 FCN-score가 줄어들었다. ImageGAN은 더 많은 파라미터를 갖고 네트워크가 더 깊기 때문에 학습이 어려웠다.
- 이 논문에서 특별한 언급이 없는 한 70×70 PatchGAN에 L1+cGAN로스를 사용하였다.

Conclusion

결론

- Image to Image Mapping Network에서 Photo-realistic을 추구하고 싶음
- 그래서 GAN의 Adversarial Training을 도입
- U-Net과 PatchGAN등을 통해서 성능 최적화

문제점

- Training Data가 Pair로 존재해야 함
- Input Image를 무시하게 될 수있다(어떤 Input이 들어오던지 똑같은 Output이 나올 수 있다)
- GAN의 학습이 어렵다

Reference

초짜 대학원생 입장에서 이해하는 Generative Adversarial Nets (1)
http://jaejunyoo.blogspot.com/2017/01/generative-adversarial-nets-1.html

Generative Adversarial Network
https://ratsgo.github.io/generative%20model/2017/12/20/gan/

Image-to-Image Translation with Conditional Adversarial Networks, Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros, CVPR 2017
https://arxiv.org/abs/1611.07004

Image-to-Image Translation with Conditional Adversarial Networks
http://www.navisphere.net/5932/image-to-image-translation-with-conditional-adversarial-networks/

(Pix2Pix) Image-to-image Translation with Conditional Adversarial Networks
https://kakalabblog.wordpress.com/2017/08/10/pix2pix-image-to-image-translation-with-conditional-adversarial-networks/

GAN을 이용한 Image to Image Translation: Pix2Pix, CycleGAN, DiscoGAN
https://taeoh-kim.github.io/blog/gan%EC%9D%84-%EC%9D%B4%EC%9A%A9%ED%95%9C-image-to-image-translation-pix2pix-cyclegan-discogan/

Image-to-Image Translation with Conditional Adversarial Networks (arXiv : 1611.07004v1)
https://m.blog.naver.com/audtlr24/220990167303

Finding connections among images using CycleGAN
https://www.youtube.com/watch?v=Fkqf3dS9Cqw&t=2799s

1시간만에 GAN (Generative Adversarial Network) 완전 정복하기
https://www.youtube.com/watch?v=odpjk7_tGY0

discriminative vs generative - data distribution
https://ratsgo.github.io/generative%20model/2017/12/17/compare/

GAN과 확률분포
http://learnai.tistory.com/4#recentComments

이미지인식문제의 개요 : PASCAL VOC Cahallenge를 중심으로 - 평가 척도
http://research.sualab.com/computer-vision/2017/11/29/image-recognition-overview-2.html

감사합니다.