Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

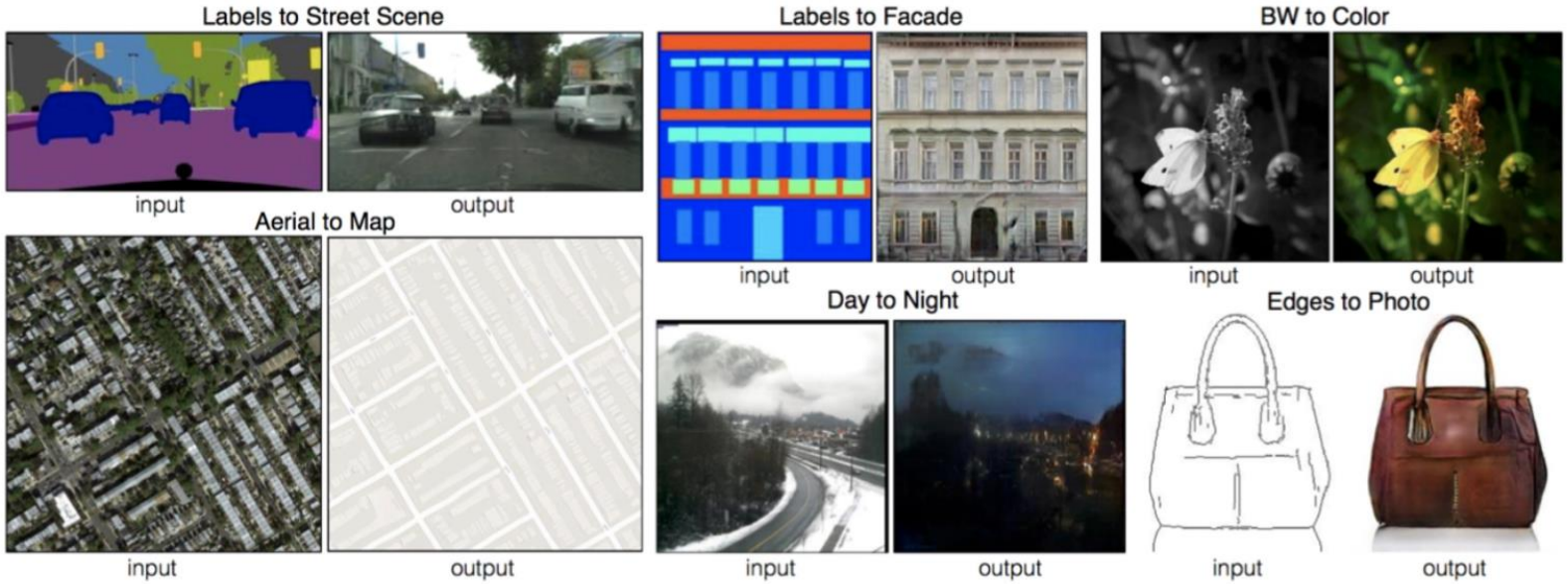
Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros

https://arxiv.org/abs/1703.10593

Pix2Pix

Image-to-Image Translation with Conditional Adversarial Networks

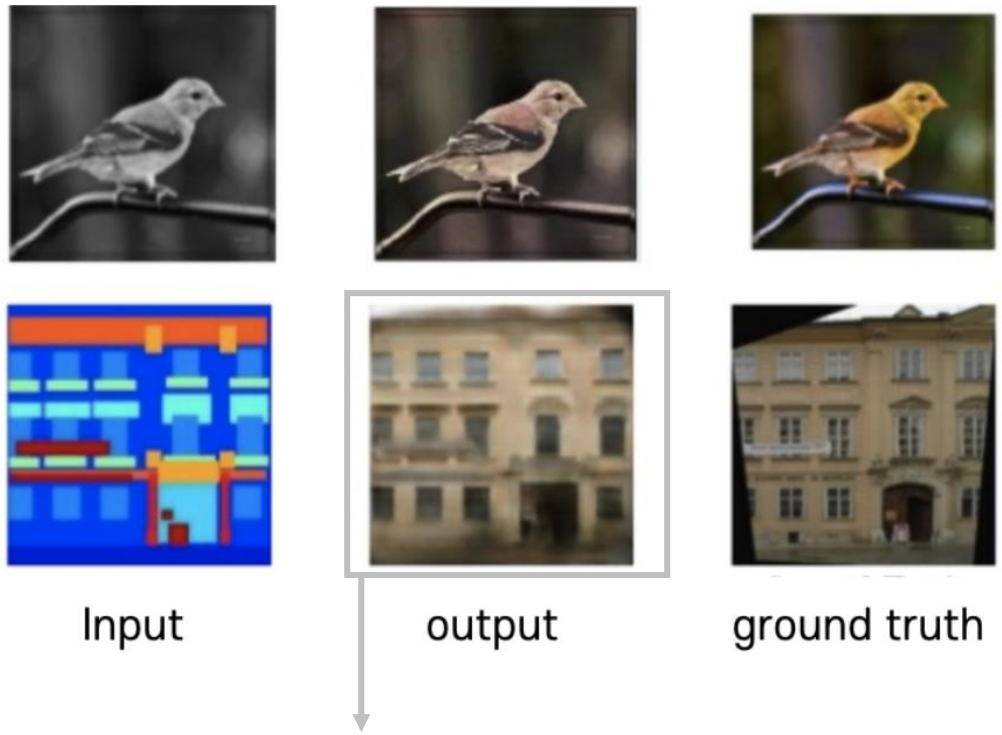
- Pix2Pix(Pixel to Pixel, =Image Translation) 알고리즘은 이미지의 Style을 변형시키는 알고리즘.
- "Pixel을 다른 Pixel로 바꿔준다."
- 흑백사진 → 컬러사진 변환, Edge image → 원본 복원 등.
- Pix2pix는 Cycle GAN을 연구한 연구실에서 Cycle GAN 이전에 만든 것.
- 지난 회 차에 다룬 Domain transfer랑은 다름.



Pix2Pix

Image-to-Image Translation with Conditional Adversarial Networks

- Loss Function



Pix2pix 결과가 뿌옇게 나오는 것(한계점)은
pix2pix를 위한 네트워크에서
pixel를 어떤 색으로 칠해야 할 지 모를 경우
중간 값을 선택하지 때문

정답 생성한 이미지

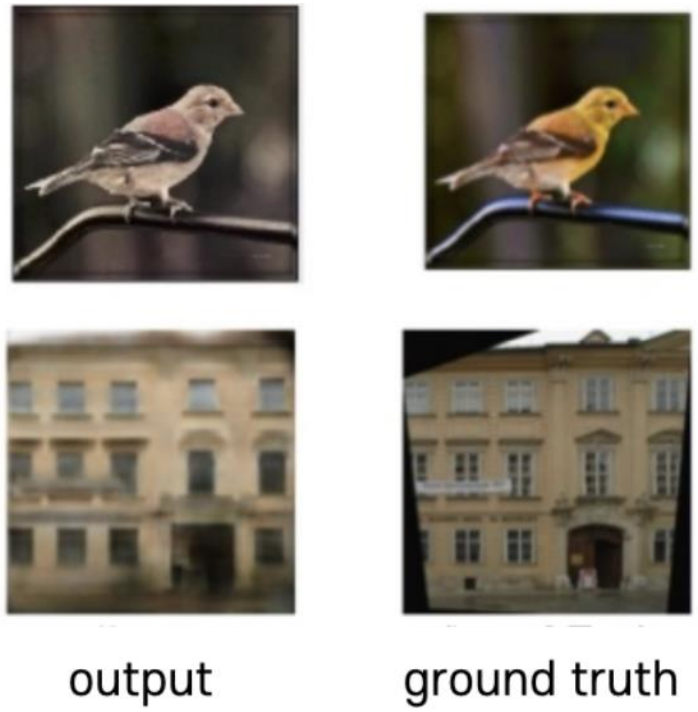
$$\sum_{(x,y)} \|y - G(x)\|_1$$

▶**Pixel level의 차이를 loss로 두고 최소화 하도록 학습!**

Pix2Pix

Image-to-Image Translation with Conditional Adversarial Networks

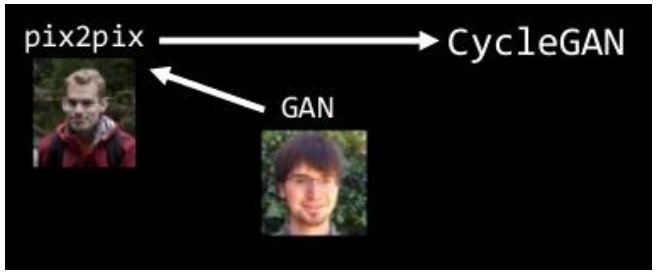
- Loss Function



사람이 보기에는 output과 Ground Truth의 차이가 명확!
즉 사람은 결과를 구별(discrimination)할 수 있다.

사람이 구별할 수 있으면 딥 러닝으로도 구별 할 수 있지 않을까?

Neural Network가 이 역할을 하게 만들어보자.



Abstract

- **(목표)** Adversarial loss를 이용해,
  G(x)-생성 이미지로부터의 이미지 데이터의 분포와 Y-정답으로부터의 이미지 데이터의 분포가 구분 불가능하도록
  G:X→Y를 학습시키는 것.

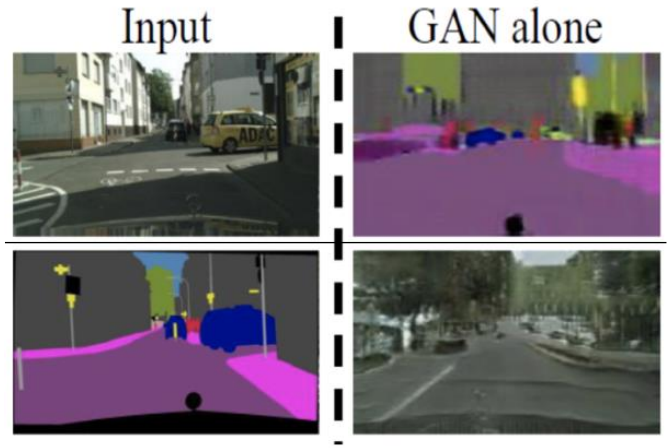
- F:Y→X 와 같은 **역방향 매핑**을 함께 진행

- F(G(x))가 X와 유사해지도록 강제하는 **cycle consistency loss**를 도입.

Introduction

- (Cycle GAN's key object)

  input을 주었을 때 해당 input을 의도한 방향으로 바꾸는데, 이걸 다시 원래의 input으로 되돌릴 때 변환 가능 할 수 있도록 이를 바꾸라는 것.



의도한대로 segmentation 을 했지만, input image랑 다른 image에다가 segmentation을 진행.

의도한대로 일반 image를 출력했지만, input image랑 다른 image가 나옴.

▶ GAN 입장에서는 input을 그냥 image segmentation 한 결과 처럼 나타내면 Discriminator net은 이름 real image 라고 생각하기 때문이다.

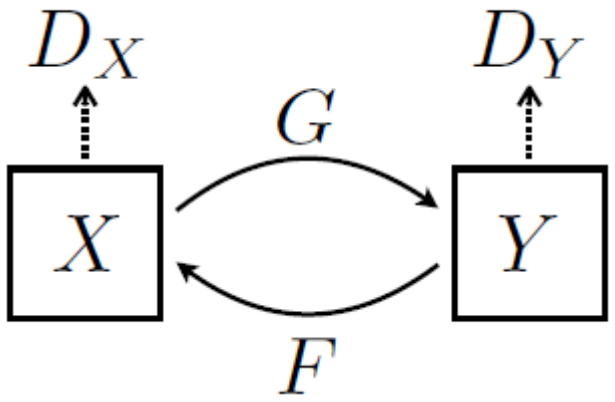
즉, Domain Y의 이미지 처럼만 보이면 되므로 Input의 특성을 무시하게 되는 것!

→ input의 성질을 유지하면서 Y의 Style로만 바꾸는 것을 못하는 결과가 발생함.

* **mode-collapse** : 어떤 인풋 이미지든 모두 같은 아웃풋 이미지로 매핑하면서 최적화에 실패하는 것

Formulation

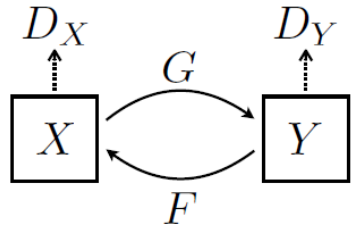
- x는 X 에 속하는, y 는 Y 에 속하는 샘플.



- 두 개의 매핑 함수 $G: X \rightarrow Y$와 $F: Y \rightarrow X$ 를 포함.
- 두 개의 적대적인(adversarial) discriminator D_x와 D_y 를 도입.
- 목적함수(objective)는 adversarial losses와 cycle consistency losses, 두 종류의 항으로 구성.

Formulation

- adversarial losses



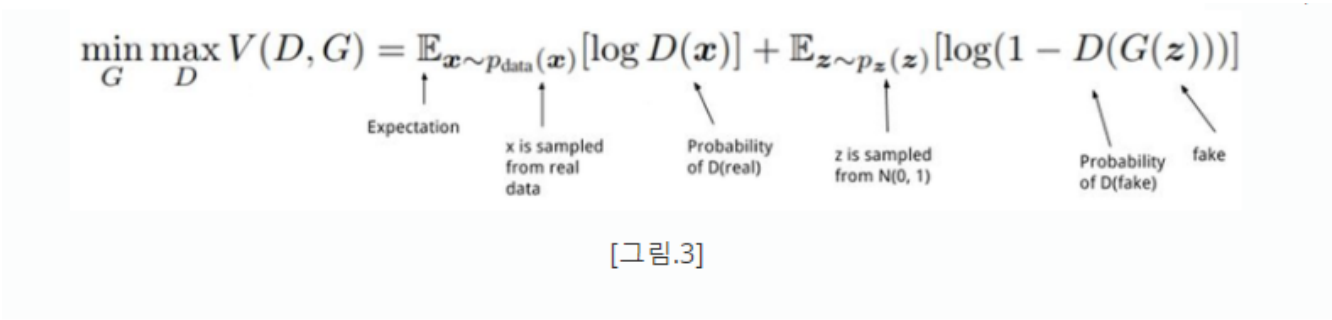
- 두 매핑 함수 모두 adversarial losses 적용.
- 함수 $G: X \rightarrow Y$와 D_y에 대해서 아래와 같은 목적함수를 적용.

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

- G는 위 함수를 최소화, D는 위 함수를 최대화 시키려고 함. → $min_G min_{D_y} L_{GAN}(F, D_y, Y, X)$ 로 나타낼 수 있음.
- $\mathbf{F}: Y \rightarrow X$ 와 D_x 대해서도 유사함. → $min_F min_{D_x} L_{GAN}(F, D_x, Y, X)$

Formulation

- (참고) 주연이의 저번 GAN 세미나 자료

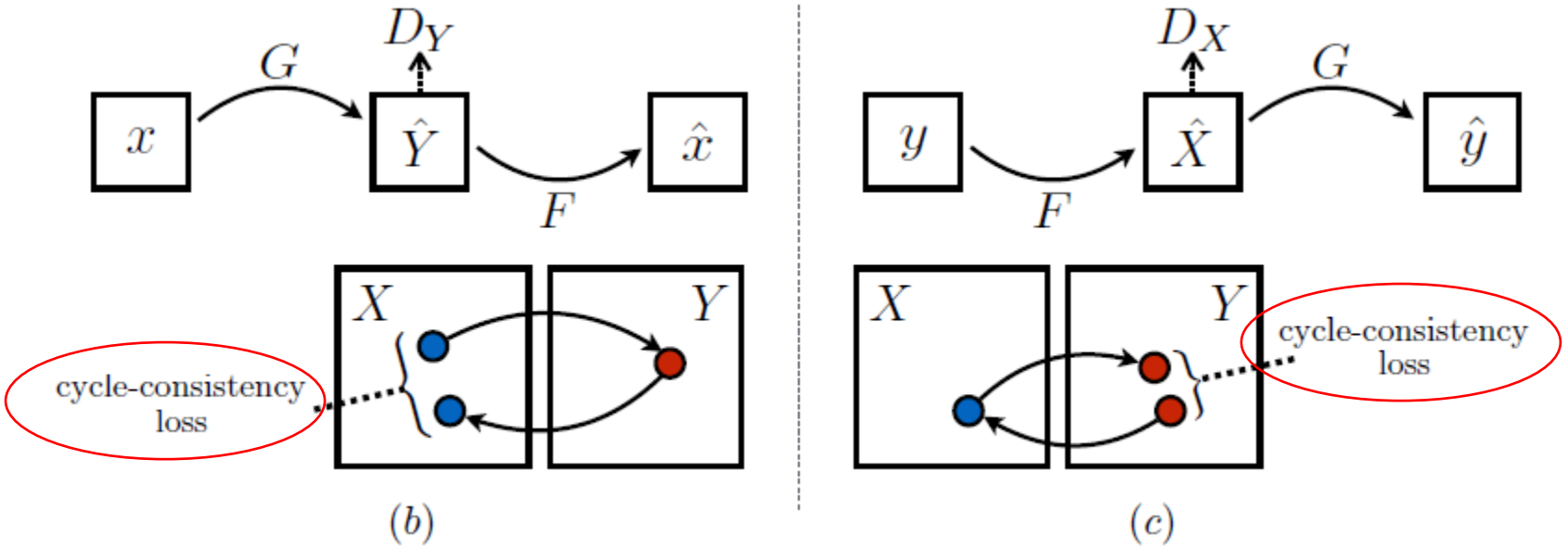


[그림.3]

- 첫번째 항 : real data x를 discriminator 에 넣었을 때 나오는 결과를 log취했을 때 얻는 기댓값.
- 두번째 항 : fake data z를 generator에 넣었을 때 나오는 결과를 discriminator에 넣었을 때 그 결과를 log(1-결과)했을 때 얻는 기댓값.
- 이 방정식을 D의 입장, G의 입장에서 각각 이해해보면,
  먼저 D의 입장에서 이 value function V(D,G)의 이상적인 결과를 생각해보면, D가 매우 뛰어난 성능으로 판별을 잘 해낸다고 했을 때, D가 판별하려는 데이터가 실제 데이터에서 온 샘플일 경우에는 D(x)가 1이 되어 첫번째 항은 0이 되어 사라지고 G(z)가 생성해낸 가짜 이미지를 구별해낼 수 있으므로 D(G(z))는 0이 되어 두번째 항은 log(1-0)=log1=0이 되어 전체 식 V(D,G) = 0이 된다. 즉 D의 입장에서 얻을 수 있는 이상적인 결과, '최댓값'은 0임을 확인 할 수 있다.
- G의 입장에서 이 value function V(D,G)의 이상적인 결과를 생각해보면, G가 D가 구별못할만큼 진짜와 같은 데이터를 잘 생성해낸다고 했을 때, 첫번째 항은 D가 구별해내는 것에 대한 항으로 G의 성능에 의해 결정될 수 있는 항이 아니므로 패스하고 두번째 항을 살펴보면 G가 생성해낸 데이터는 D를 속일 수 있는 성능이라 가정했기 때문에 D가 G가 생성해낸 이미지를 가짜라고 인식하지 못하고 진짜라고 결정내버린다. 그러므로 D(G(z)) =1이 되고 log(1-1)=log0=마이너스무한대가 된다. 즉, G의 입장에서 얻을 수 있는 이상적인 결과, '최솟값'은 '마이너스무한대'임을 확인할 수 있다.

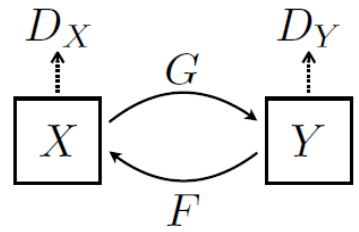
Formulation

- cycle consistency losses
  - Introduction에서 언급한 것과 같이 mode collapse 문제가 생길 수 있기 때문에, Adversarial losses 단독으로는 매핑 함수를 제대로 된 학습을 보장하기 어려움.
  - 가능한 매핑 함수의 공간을 줄이기 위해, 아래의 그림(b), (c)와 같이 매핑 함수는 cycle-consistent 해야 함.



Formulation

- cycle consistency losses



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Formulation

- cycle consistency losses

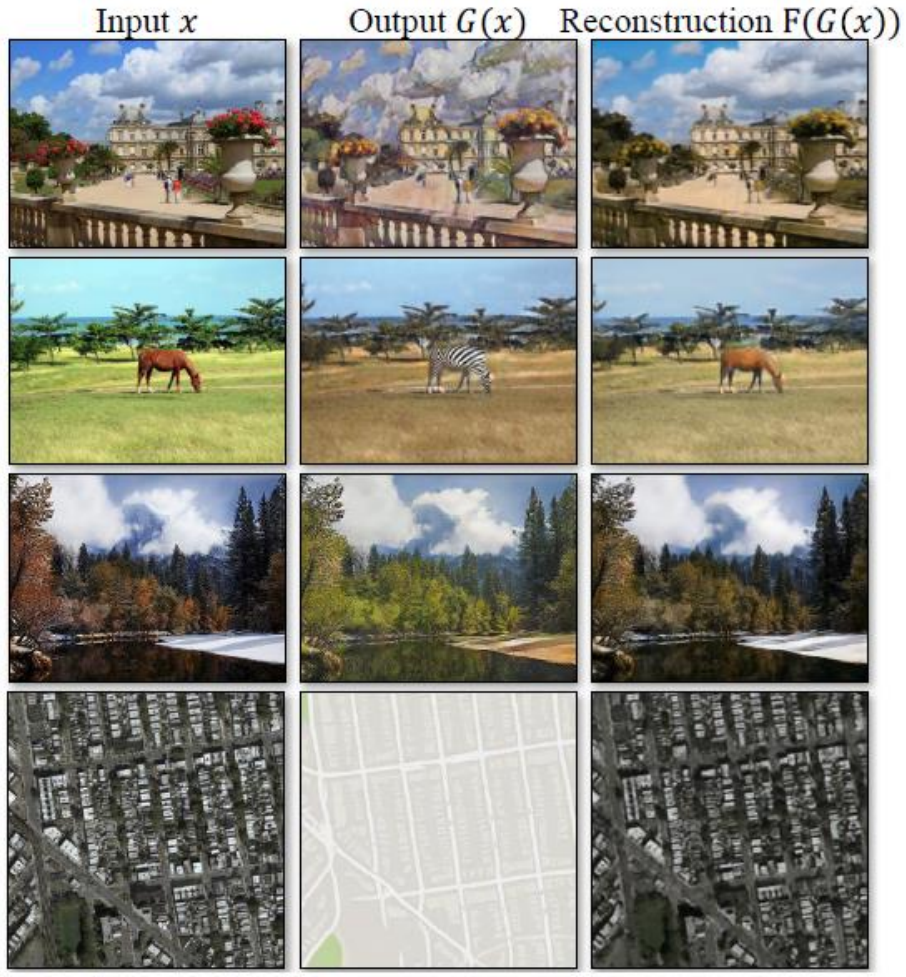


Figure 4: The input images x, output images $G(x)$ and the reconstructed images $F(G(x))$ from various experiments. From top to bottom: photo↔Cezanne, horses↔zebras, winter→summer Yosemite, aerial photos↔Google maps.

Formulation

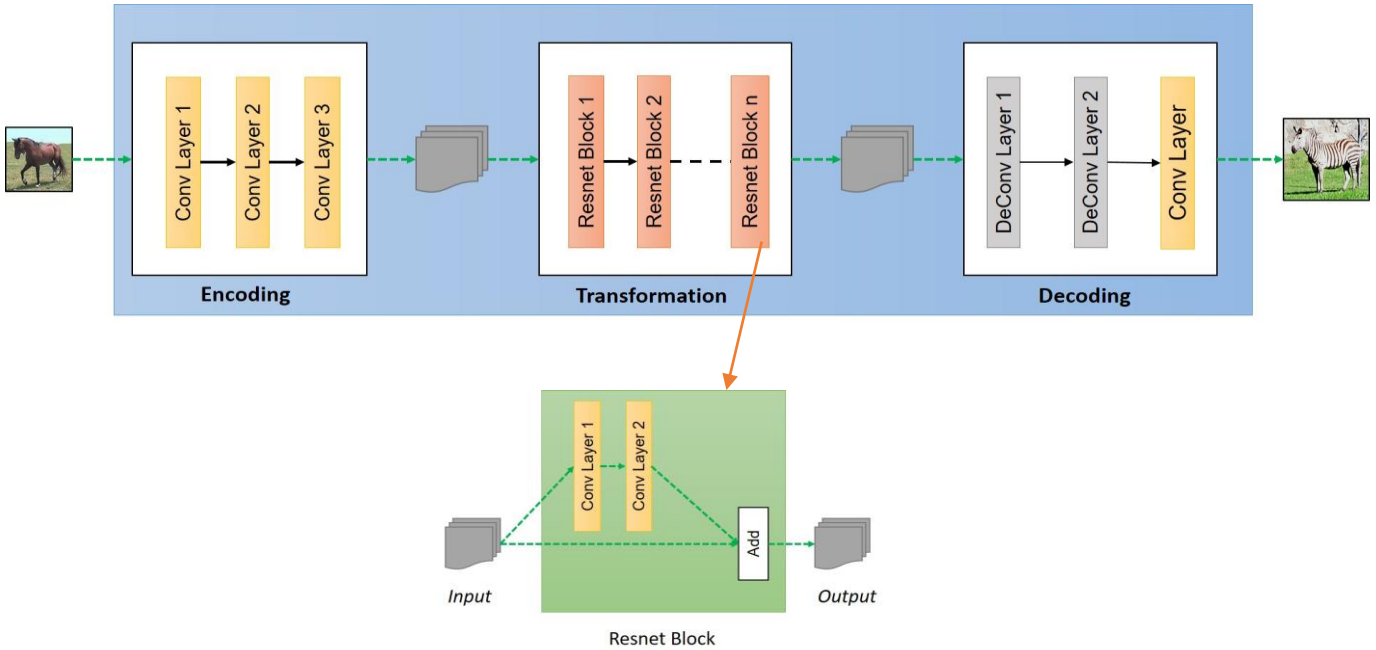
- **Full objective Function**
  - λ는 두 함수(= 위 식에서의 첫 번째 항과 두 번째 항)의 상대적인 중요도에 따라 결정.

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

Implementation

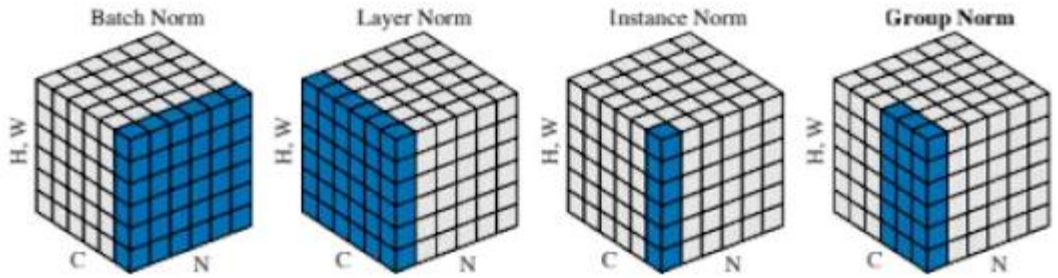
- Network Architecture
  - 128x128 이미지에는 6-block, 256x256 혹은 그 이상의 해상도의 이미지에는 9-block.
  - instance normalization 적용.
  - 70x70 PatchGANs 사용.



Implementation

- **Network Architecture**

  · instance normalization.



※ 이미지의 Height, Width를 1 차원으로 합치고, Channel C와 N개의 Batch로 도식화

- 파란색 부분은 평균과 분산을 구해 normalization 하는 영역.
- Batch Norm.에서는 batch 샘플 별 평균과 분산을 구하는 것을 확인할 수 있다.
- Layer Norm.에서는 모든 Channel의 평균과 분산을 구하기 때문에 1개 batch에서도 동작할 수 있다. (Feature 차원의 normalization)
- **Instance Norm.은 1개 batch내의 각 Channel 마다 평균과 분산으로 normalization하는 방법이다.**
  (real-time generation에 효과적이라고 한다. style transfer을 위해 고안된 방법.)

Implementation

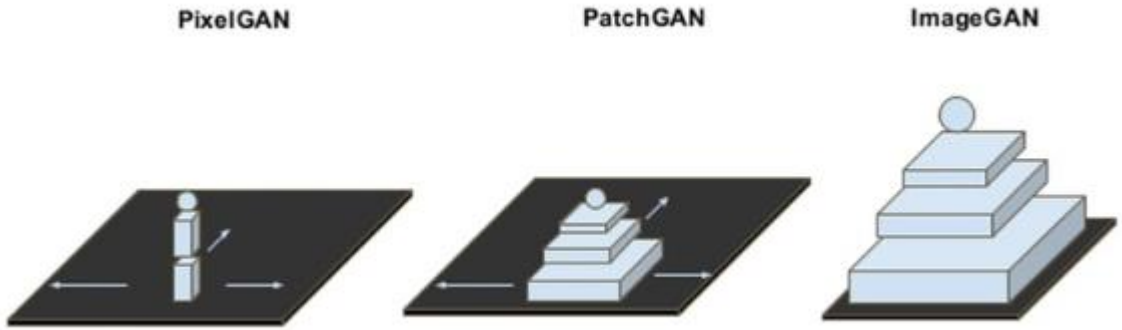
- **Network Architecture**

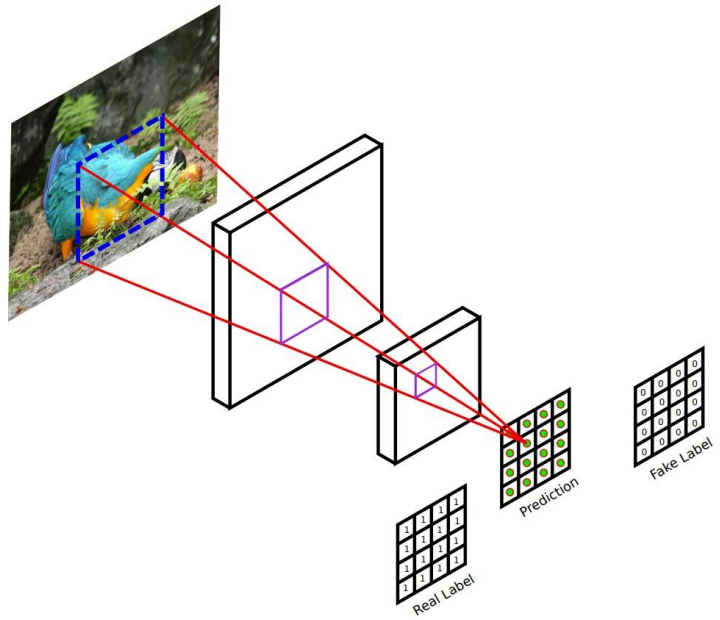
  · PatchGANs

  ➢ 특정 크기의 patch 단위로 진짜/가짜를 판별하고, 그 결과에 평균을 취하는 방식 → PatchGAN

  ➢ 픽셀들 간의 연관성은 거리에 비례하여 작아지는 경향이 있으며, 일정한 거리를 넘어서게 되면 상호간에 별 의미가 없다.

  ➢ 따라서, 특정 크기의 patch에 대하여 진짜 같은 이미지를 생성할 수 있고, 그런 patch의 수가 많아지는 방향으로 학습하게 된다면, Generator의 성능은 향상 될 것.



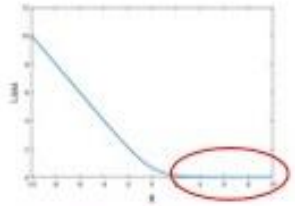


Implementation

- Training details
  - GAN이 발산을 잘 하기로 악명이 높음,,,
  - GAN의 Loss function이 양의 방향으로 갈수록 Vanishing gradient가 발생하여 학습이 잘 안됨.
  - 이를 LSGAN(Least square GANs[Mao et al. 2016])을 사용하여 학습에 더 용이하게 함.

• GANs with cross-entropy loss

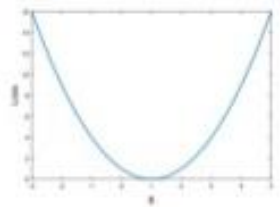
$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$



Vanishing gradients

• Least square GANs [Mao et al. 2016]
Stable training + better results

$$\mathcal{L}_{\text{LSGAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[(D_Y(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D_Y(G(x))^2],$$



Implementation

- Training details

· 모델의 진동(oscillation)을 줄이기 위하여 Discriminator는 Generator가 생성한 최근 하나의 이미지를 이용하지 않고,

지금까지 생성된 이미지들을 사용하여 판단.

· 이미지 50개를 저장할 수 있는 버퍼를 이용.

· λ는 10.

· 배치 사이즈는 1, Adam solver를 이용했습니다.

· 처음 100 epoch에 대해서는 learning rate 0.0002를 유지하였고, 이후 100 epoch에서는 선형적으로 0에 가까워지게 lr를 줄여가며 학습.

Result

- Evaluation metrics

① **AMT perceptual studies**

· 사람을 대상으로 한 실험.

· 참가자들에게는 실제 사진 혹은 지도와 (우리의 알고리즘 혹은 baseline 모델을 통해 생성된) 가짜 이미지를 보여준 후 그들이 진짜라고 생각하는 것을 선택하게 함.

② **FCN score**

· pre-trained 된 semantic classifier를 이용해서 생성한 이미지로부터 정확한 object를 구별해 낼 수 있는지를 나타냄.

③ **Semantic segmentation metrics**

· 사진을 라벨링 하는 성능을 평가하기 위해 우리는 per-pixel 정확도(accuracy)와 IOU를 포함하는 기본적인 평가지표를 이용.

Result

- Comparison against baselines

Loss	Map → Photo % Turkers labeled *real*	Photo → Map % Turkers labeled *real*
CoGAN [32]	0.6% ± 0.5%	0.9% ± 0.5%
BiGAN/ALI [9, 7]	2.1% ± 1.0%	1.9% ± 0.9%
SimGAN [46]	0.7% ± 0.5%	2.6% ± 1.1%
Feature loss + GAN	1.2% ± 0.6%	0.3% ± 0.2%
CycleGAN (ours)	**26.8% ± 2.8%**	**23.2% ± 3.4%**

Table 1: AMT "real vs fake" test on maps↔aerial photos at 256 × 256 resolution.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.40	0.10	0.06
BiGAN/ALI [9, 7]	0.19	0.06	0.02
SimGAN [46]	0.20	0.10	0.04
Feature loss + GAN	0.06	0.04	0.01
CycleGAN (ours)	**0.52**	**0.17**	**0.11**
pix2pix [22]	0.71	0.25	0.18

Table 2: FCN-scores for different methods, evaluated on Cityscapes labels→photo.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
CoGAN [32]	0.45	0.11	0.08
BiGAN/ALI [9, 7]	0.41	0.13	0.07
SimGAN [46]	0.47	0.11	0.07
Feature loss + GAN	0.50	0.10	0.06
CycleGAN (ours)	**0.58**	**0.22**	**0.16**
pix2pix [22]	0.85	0.40	0.32

Table 3: Classification performance of photo→labels for different methods on cityscapes.

Result

- Analysis of the loss function

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Cycle alone	0.22	0.07	0.02
GAN alone	0.51	0.11	0.08
GAN + forward cycle	**0.55**	**0.18**	**0.12**
GAN + backward cycle	0.39	0.14	0.06
CycleGAN (ours)	0.52	0.17	0.11

Table 4: Ablation study: FCN-scores for different variants of our method, evaluated on Cityscapes labels→photo.

Loss	Per-pixel acc.	Per-class acc.	Class IOU
Cycle alone	0.10	0.05	0.02
GAN alone	0.53	0.11	0.07
GAN + forward cycle	0.49	0.11	0.07
GAN + backward cycle	0.01	0.06	0.01
CycleGAN (ours)	**0.58**	**0.22**	**0.16**

Table 5: Ablation study: classification performance of photo→labels for different losses, evaluated on Cityscapes.

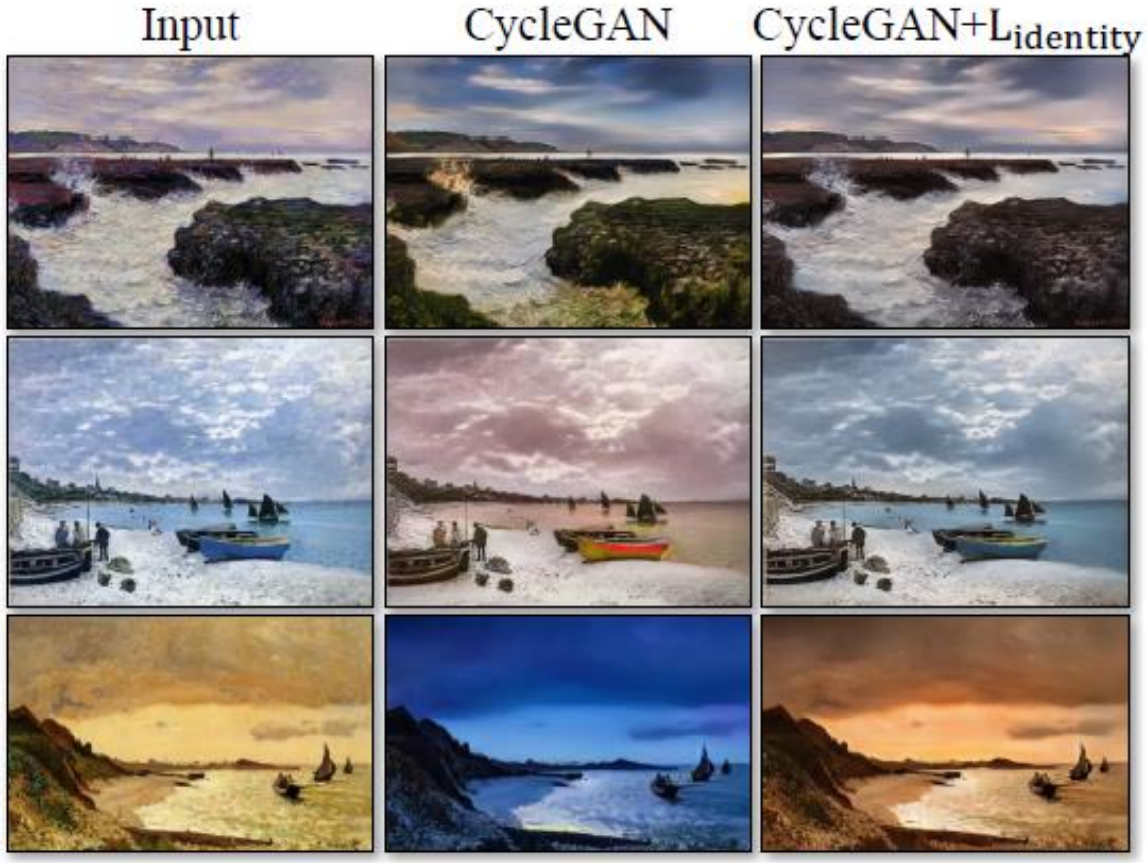
Result

- Analysis of the loss function

Figure 7: Different variants of our method for mapping labels↔photos trained on cityscapes. From left to right: input, cycle-consistency loss alone, adversarial loss alone, GAN + forward cycle-consistency loss ($F(G(x)) \approx x$), GAN + backward cycle-consistency loss ($G(F(y)) \approx y$), CycleGAN (our full method), and ground truth. Both *Cycle alone* and *GAN + backward* fail to produce images similar to the target domain. *GAN alone* and *GAN + forward* suffer from mode collapse, producing identical label maps regardless of the input photo.

Application

- 짝지어진 학습 데이터가 없는 Task에 대해 모델을 적용한 결과

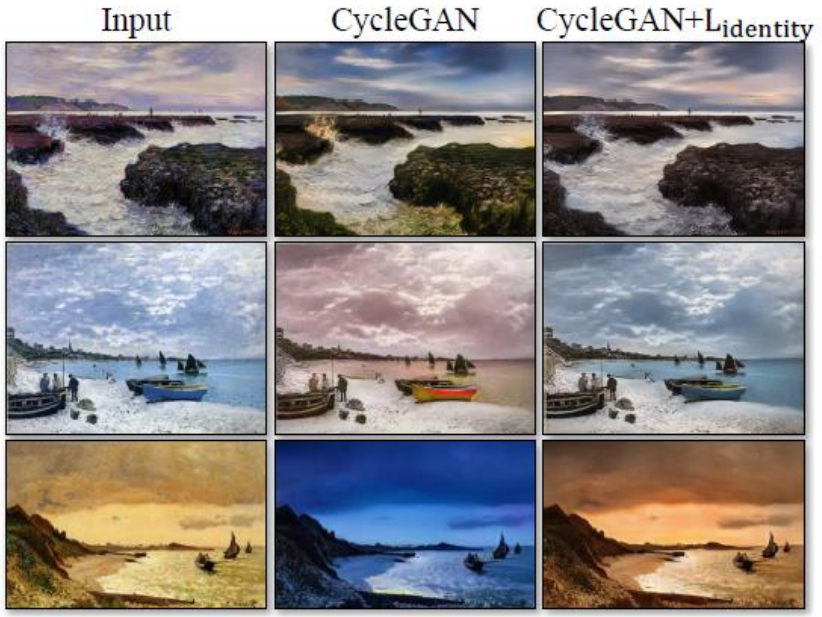


* PIX2PIX는 pair된 데이터셋이 있어야 학습이 가능한데 CycleGAN은 완벽히 pair한 데이터가 아니어도 학습이 가능하다. 예를 들어 설명하자면 PIX2PIX는 한강의 흑백사진을 갖고 있고 한강의 컬러사진도 있어야 학습이 가능하다.

Figure 9: The effect of the *identity mapping loss* on Monet's painting→ photos. From left to right: input paintings, CycleGAN without identity mapping loss, CycleGAN with identity mapping loss. The identity mapping loss helps preserve the color of the input paintings.

Application

- 짝지어진 학습 데이터가 없는 Task에 대해 모델을 적용한 결과



  - $L_{identity}$(=L_1 Loss) 적용.

  - 연구진은 그림으로부터 사진을 생성하는 태스크를 실행하는 중,

    generator가 종종 낮의 그림을 해 질 녘의 사진으로 바꾸는 것과 같은 문제 확인.

  - 인풋과 아웃풋의 색 구성을 보존하기 위해 추가적인 loss인 $L_{identity}$ 도입.

  - 이 loss는 타깃 도메인의 실제 샘플,

    즉, 그림이 아닌 사진이 인풋으로 들어왔을 때는 사진 자기 자신을 아웃풋으로 산출하도록 generator를 regularize함.

  - 저자의 설명으로는 큰 방향으로는 L1 loss를 통해 학습하고 작은 부분을 GAN을 이용하여 학습을 진행하는 것이라고 이해하면 된다고 한다.

$$L_{identity}(G, F) = E_{y \sim p_{data}(y)}[\| G(y) - y \|_1] + E_{x \sim p_{data}(x)}[\| F(x)) - x \|_1]$$

Application

- 짝지어진 학습 데이터가 없는 Task에 대해 모델을 적용한 결과

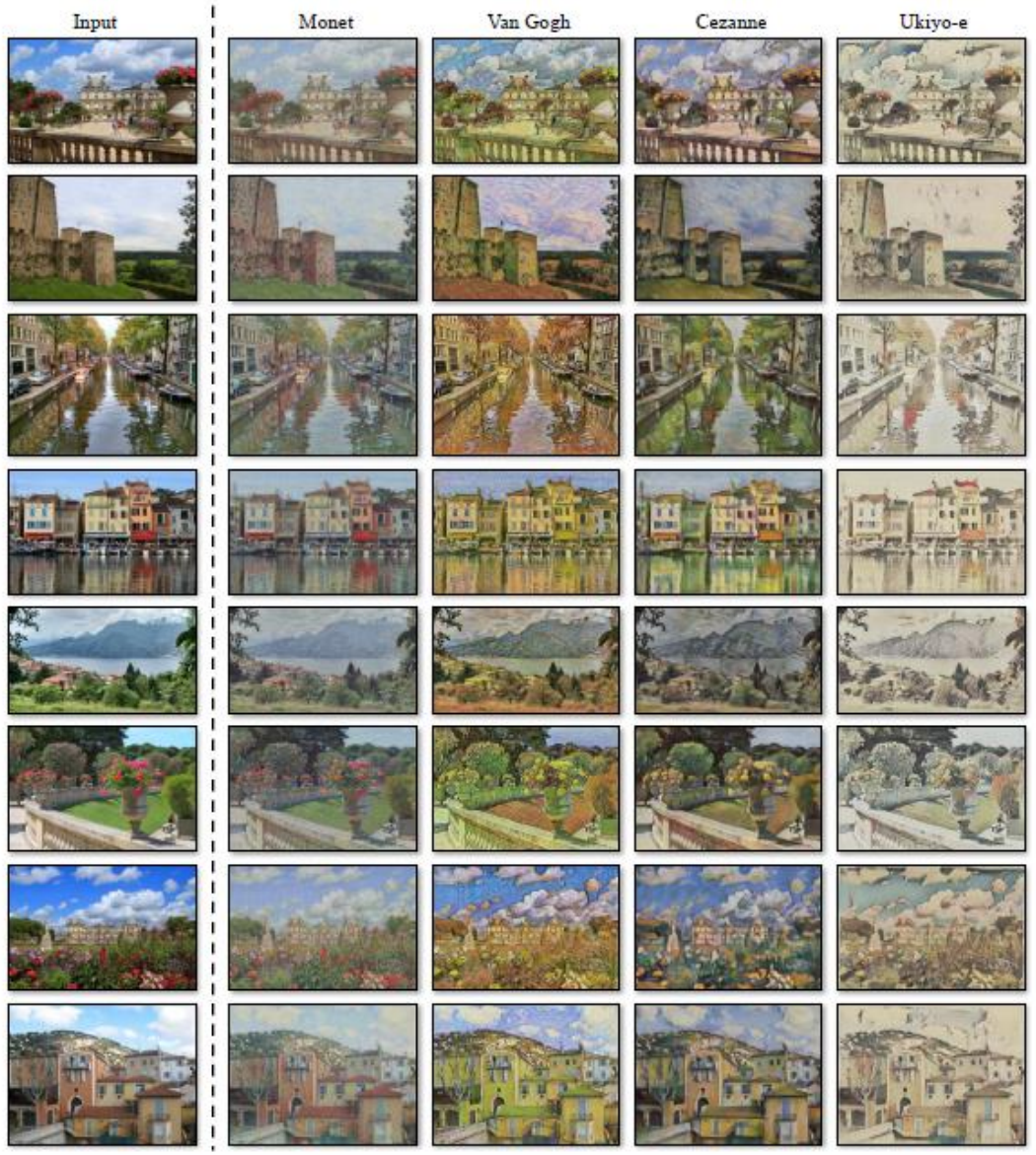
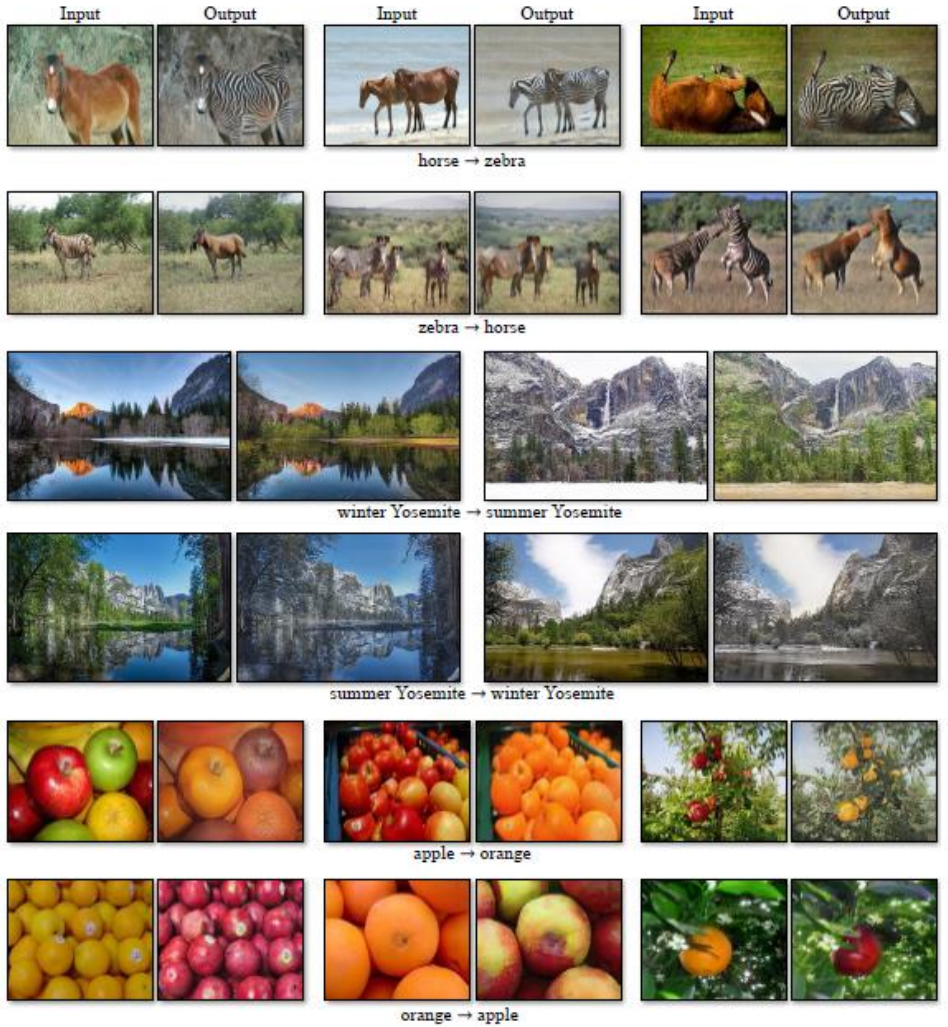


Figure 10: Collection style transfer I: we transfer input images into the artistic styles of Monet, Van Gogh, Cezanne, and Ukiyo-e. Please see our website for additional examples.

Application

- 짝지어진 학습 데이터가 없는 Task에 대해 모델을 적용한 결과



Limitations and Discussion

- 모양을 바꾸기가 어렵다. → 가장 큰 한계점.

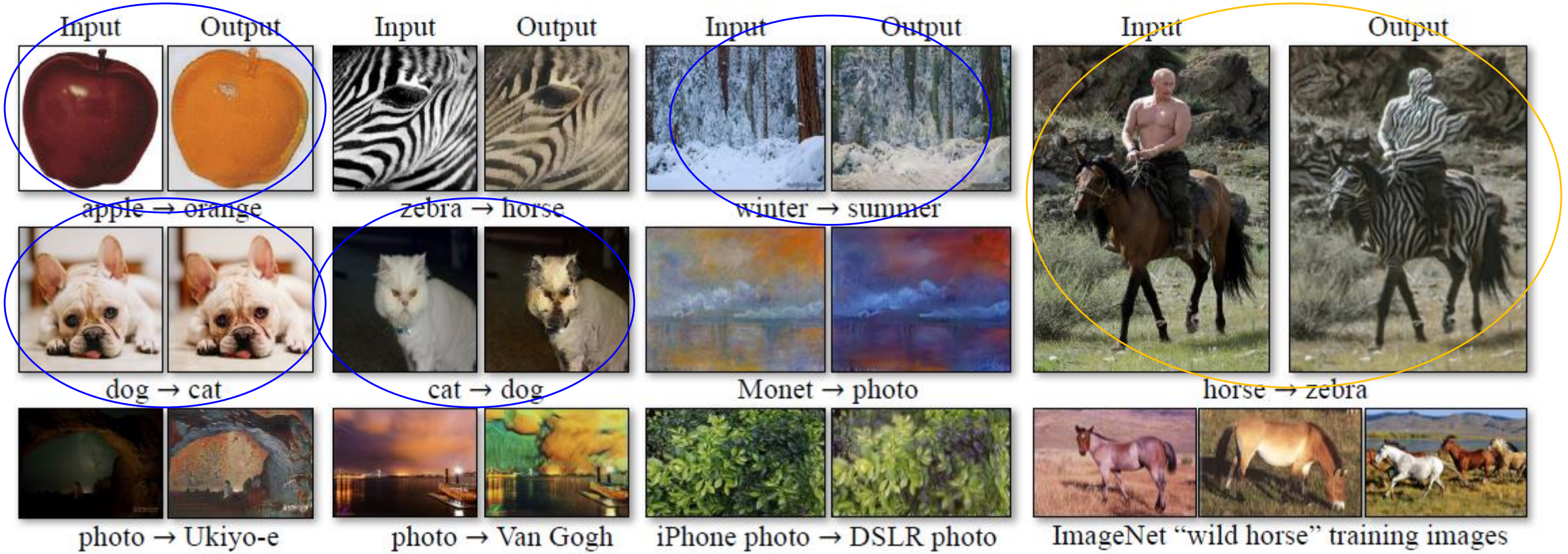
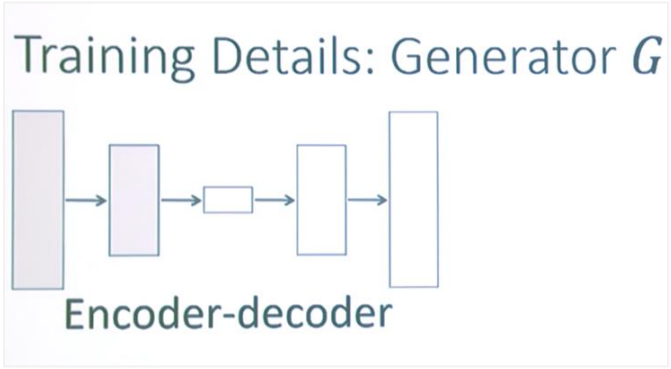


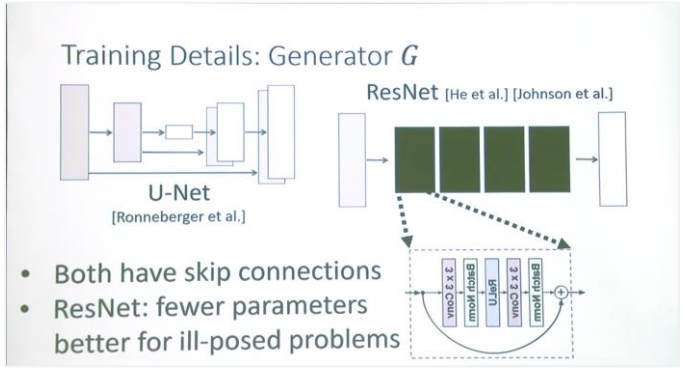
Figure 17: Typical failure cases of our method. Left: in the task of dog→cat transfiguration, CycleGAN can only make minimal changes to the input. Right: CycleGAN also fails in this horse → zebra example as our model has not seen images of horseback riding during training. Please see our website for more comprehensive results.

Limitations and Discussion

- 모양을 바꾸기가 어렵다. → 가장 큰 한계점.



- Generator의 구조를Encoder-decoder를 사용할 경우,
  bottleneck에서 저장할 수 있는 부분이 많이 없다. 극단적으로 말하면 형태도 다 잃어 버리게 된다.
- 따라서 모양의 형태도 잘 바꿀 수 있지만 디테일한 부분을 나타내기 힘들다는 한계점이 있다.

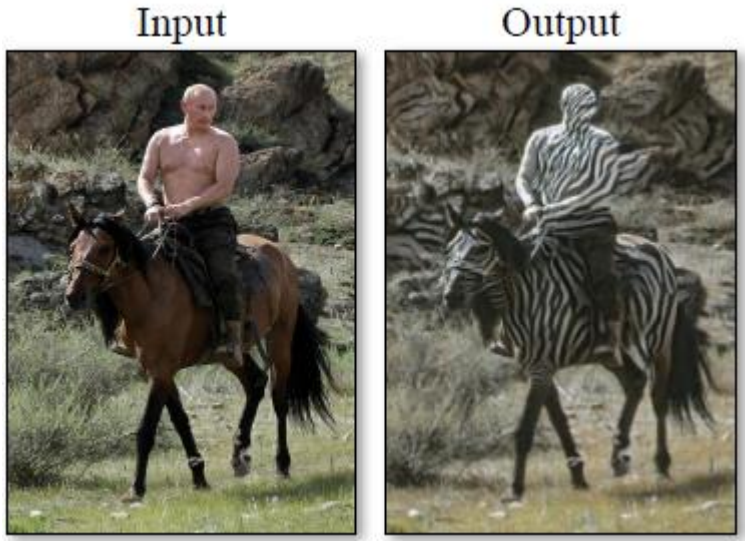


- CycleGAN 같은 경우 처음엔 U-net(encoder - decoder network 에 skip connection 만 한 것)을 사용했다고 한다.
  이 경우 skip connection 때문에 detail이 더 살아 있다고 할 수 있다 하지만 디테일을 살려야 하기 때문에 skip connection에 의존을 많이 한다고 한다.
  하지만 depth가 깊지 않기 때문에 문제가 된다고 한다.

- 따라서 이를 타개하고자 ResNet을 사용하여서 깊이도 깊고 skip-connection도 이용하여 데이터 사진의 디테일한 부분을 다 간직할 수 있게 된다.
  따라서 CycleGAN 같은 경우 결과물을 보면 형태가 잘 간직된것을 볼 수 있다. 하지만 input 의 디테일한 부분을 다 갖고 있기 때문에 모양의 형태 등을 바꾸는 것은 어렵다고 한다.

Limitations and Discussion

- Data set 의 문제



- CycleGAN에서 ImageNet 데이터셋을 가지고 학습을 진행하였는데,
  데이터셋에서 wild Horse의 경우 사람이 타고 있는 경우가 없었다.
- 그래서 Discriminator를 속이기 위해서는 모델내에서 사람도 얼룩형태로 되어야 속일 수 있다고 판단하고 산출한 결과라고 한다.
- 만약 데이터셋에서 말을 사람이 타고 있는 사진이 있었다면 잘 처리했을 것 같다고 한다.

➢ Reference

[논문] https://arxiv.org/abs/1703.10593

[Pix2Pix] https://hyeongminlee.github.io/post/gan004_pix2pix/

[논문] https://algopoolja.tistory.com/49

[논문] http://www.kwangsiklee.com/2018/03/cyclegan%EC%9D%B4-%EB%AC%B4%EC%97%87%EC%9D%B8%EC%A7%80-%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90/

[Pix2pix논문] https://arxiv.org/abs/1611.07004

[논문] https://comlini8-8.tistory.com/9

[Cycle GAN Network Architecture 사진] https://hardikbansal.github.io/CycleGANBlog/

[Instance Normalization] https://sonsnotation.blogspot.com/2020/11/8-normalization.html

[PatchGAN] https://blog.naver.com/laonple/221366130381

[PatchGAN 사진] https://brstar96.github.io/mldlstudy/what-is-patchgan-D/

[Training details에 loss 사진] http://www.kwangsiklee.com/2018/03/cyclegan%EC%9D%B4-%EB%AC%B4%EC%97%87%EC%9D%B8%EC%A7%80-%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90/

[FCN score] https://medium.com/humanscape-tech/paper-review-pix2pix-20418569e0c1