2019 DIYA COMPUTER VISION
Second Meet-up

GAN

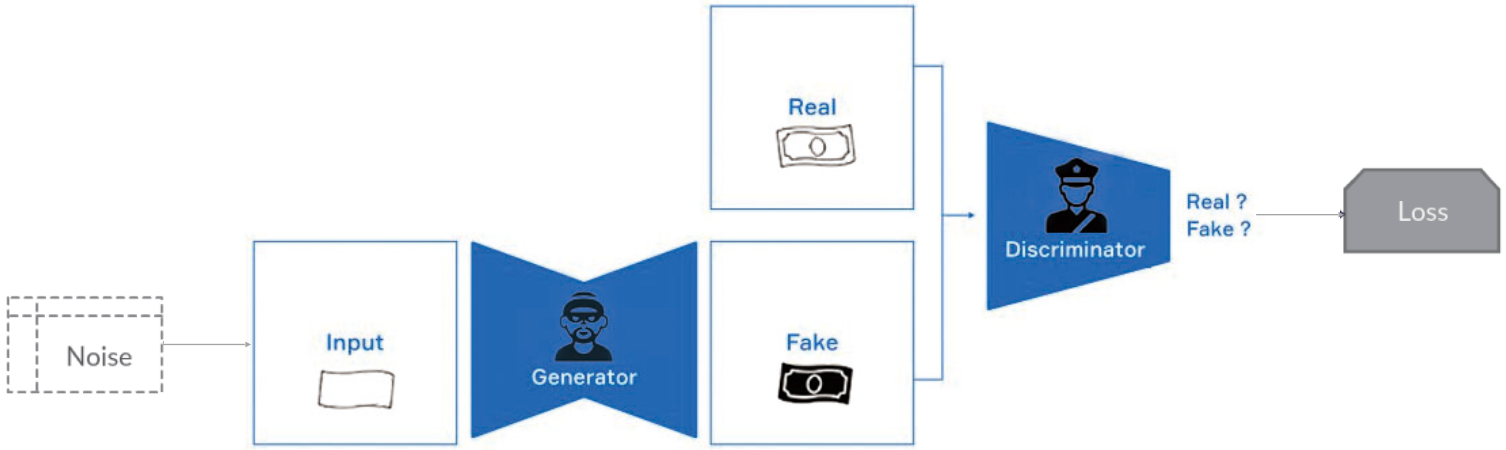
CONTENTS

CONTENTS

GAN?

Generative Adversarial Nets

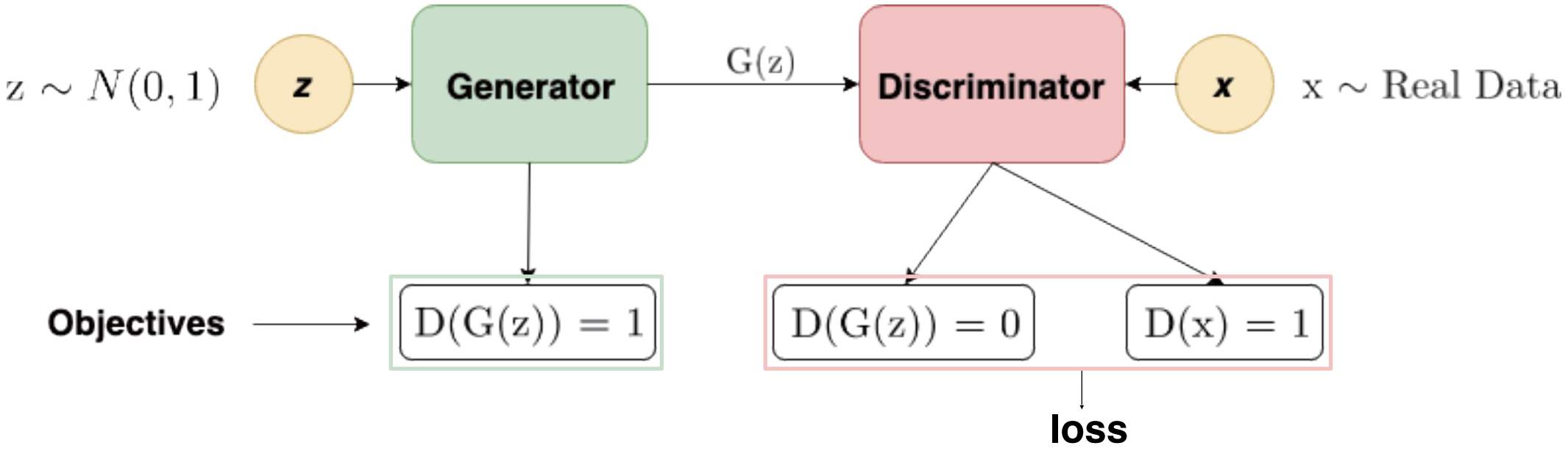
Ian J. Goodfellow, Jean Pouget-Abadie,* Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,† Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7



GAN?

How GAN works?

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



- D는 real과 fake를 구별하려고 함→ D는 Pg와 Pdata를 구분하는 확률을 maximize하는 방향으로 학습
- G는 (fake를 real로) D를 속이려고 함 → G는 log(1-D(G(z)))값을 minimize하는 방향으로 학습

→ Algorithm !

GAN?

Algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Sample minibatch of m examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$$

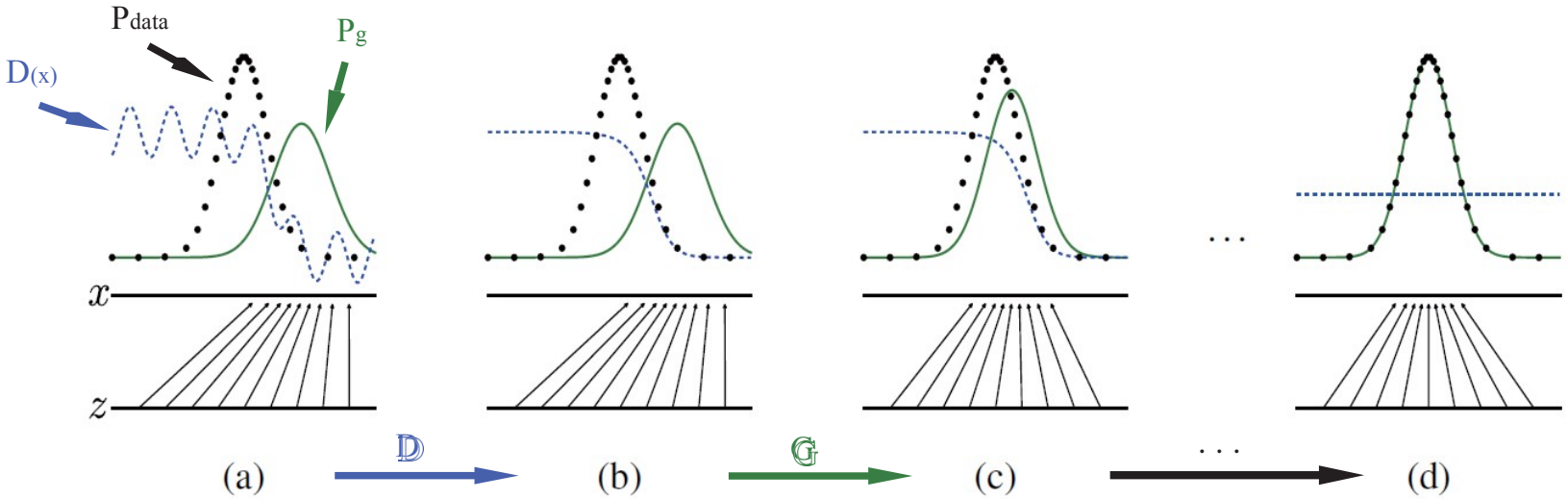
  end for

  - Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

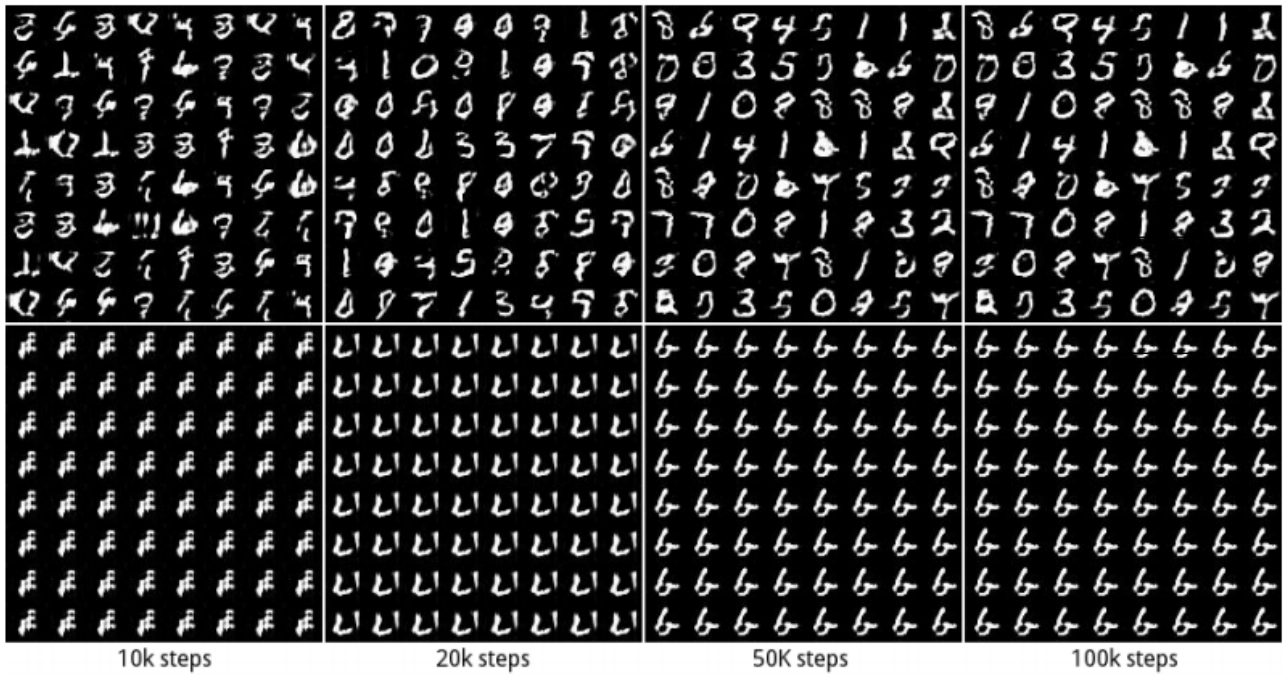


GAN?

Disadvantages

- Non-convergence : the model parameters oscillate → 모델을 안정화 하는 것이 어려움
- Mode collapse : 같은 내용의 데이터만 생성하게 됨(다양성 감소)
- Diminished gradient : Discriminator가 너무 빨리 학습하면 generator는 거의 학습을 못하게 됨
- 하이퍼 파라미터 선택에 보다 민감함
- 평가할 수 있는 객관적인 기준을 정하는 것이 매우 어려움

using MNIST

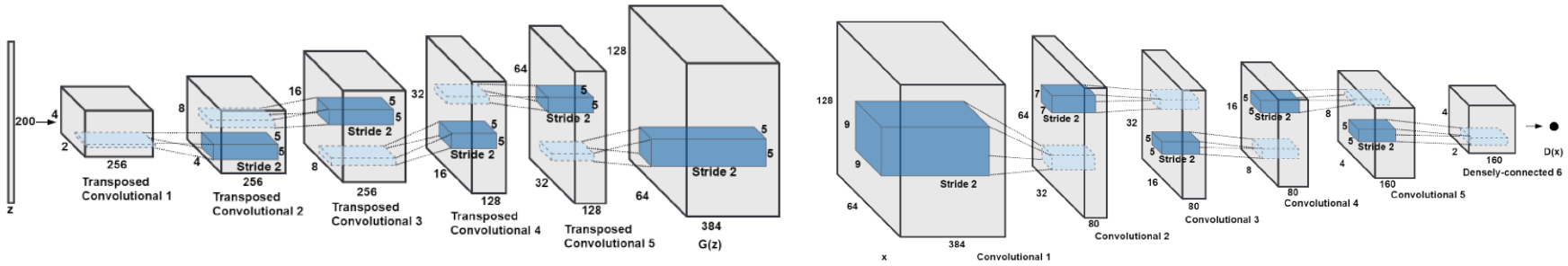


DCGAN?

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

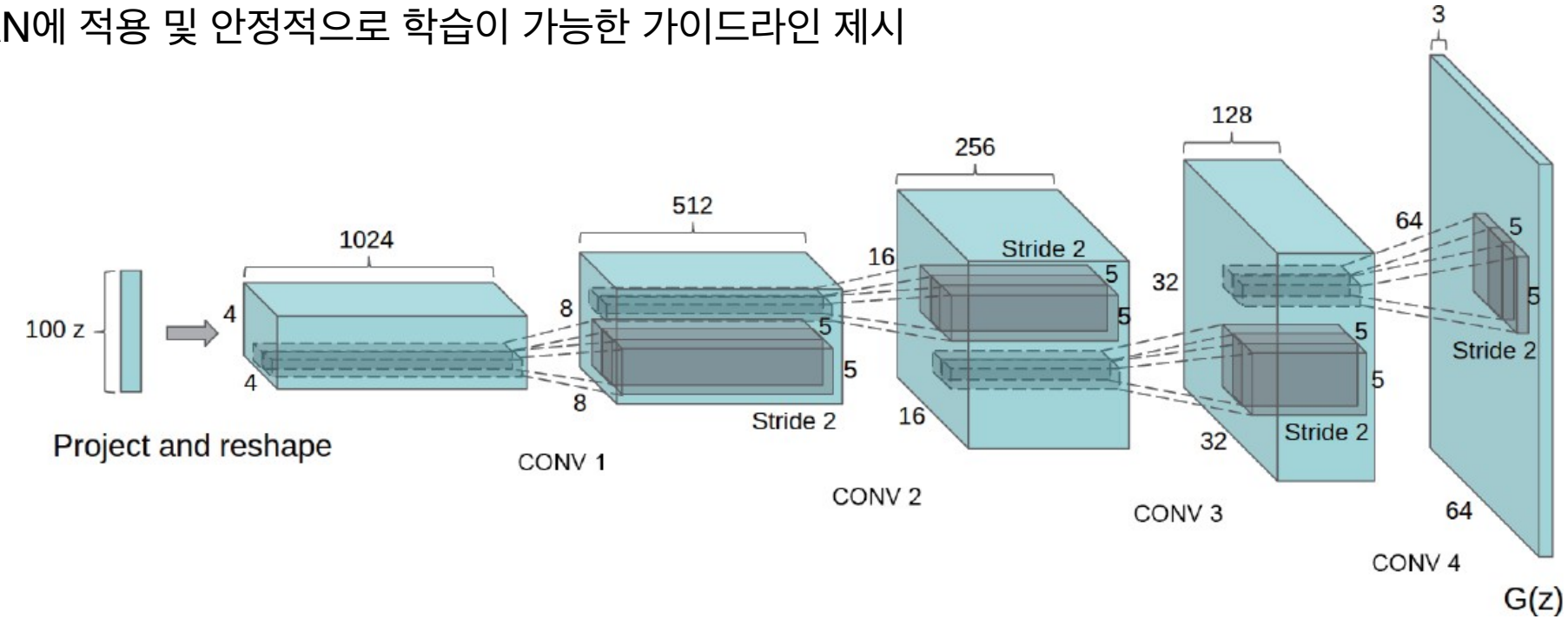
Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com



DCGAN?

CNN을 GAN에 적용 및 안정적으로 학습이 가능한 가이드라인 제시



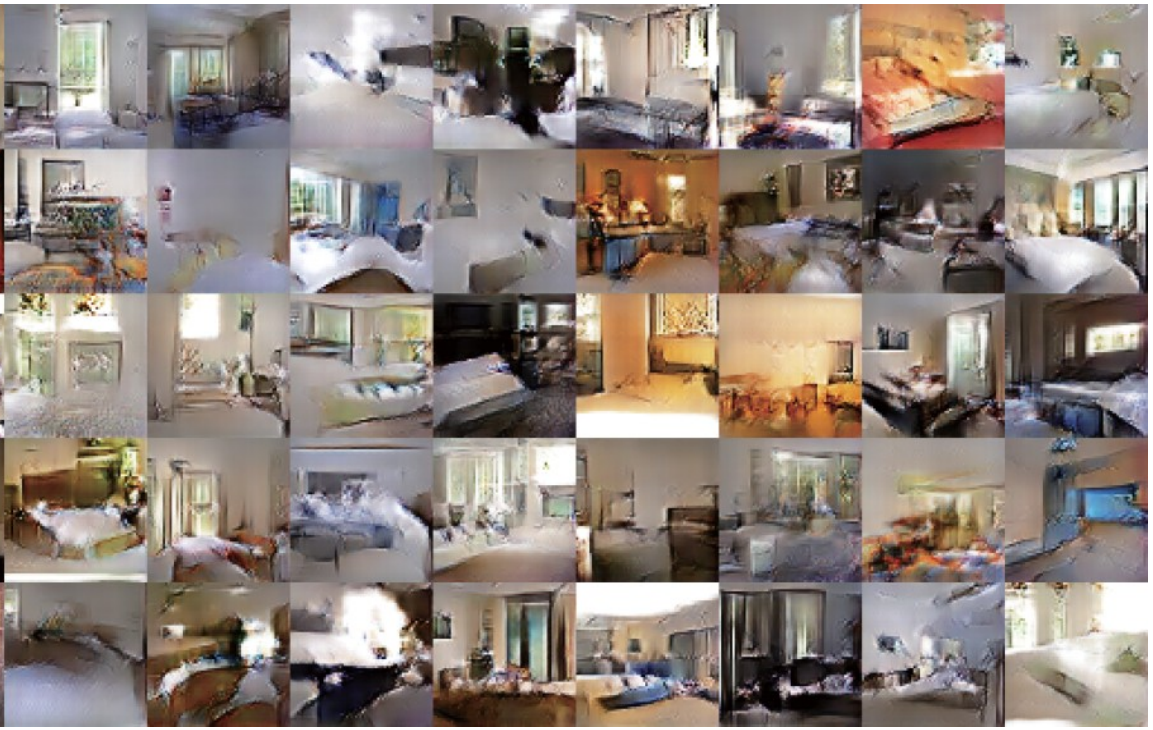
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

DCGAN?

DCGAN 생성 이미지(LSUN 학습)

1epoch 학습



5epoch 학습

DCGAN?

Not memorization, walking in the latent space

CycleGAN

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

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

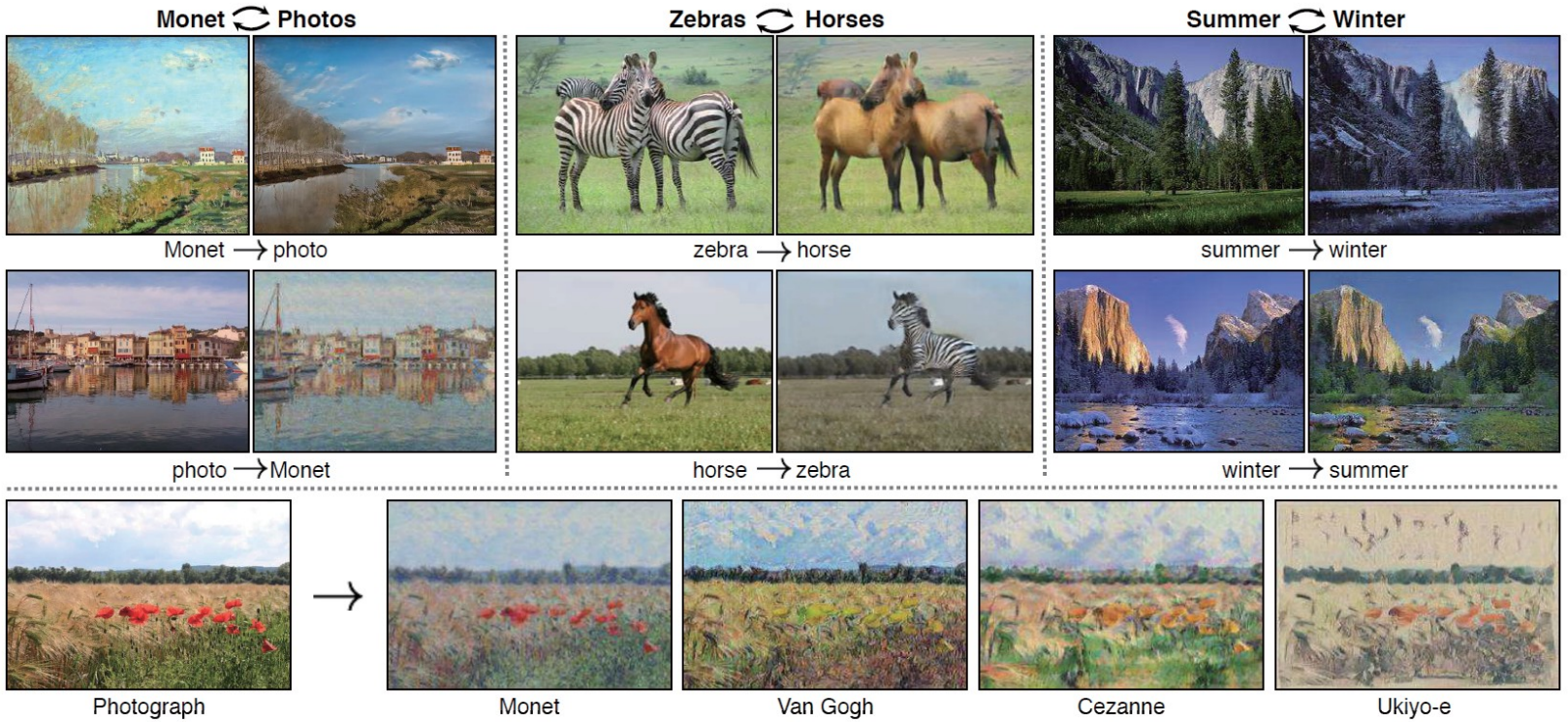


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa. Example application (*bottom*): using a collection of paintings of a famous artist, learn to render a user's photograph into their style.

CycleGAN

cf . Pix2Pix

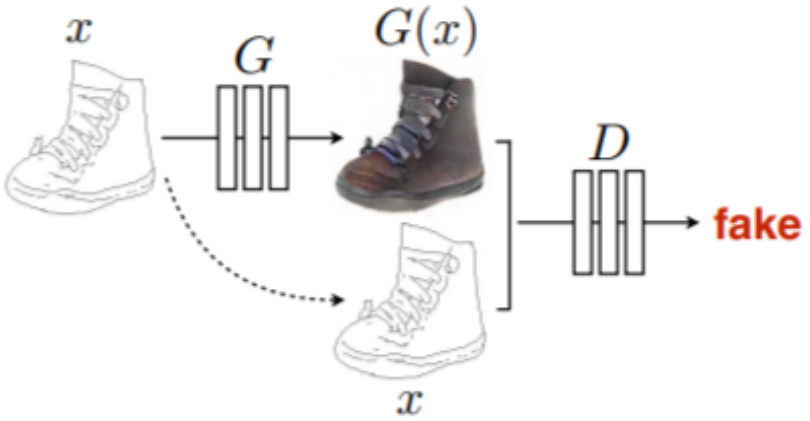
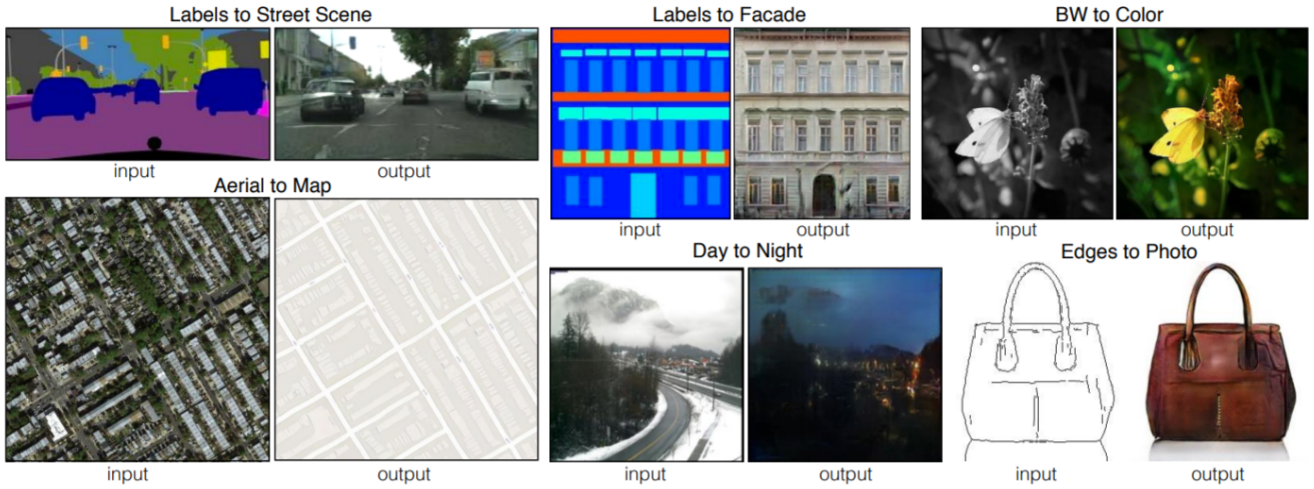


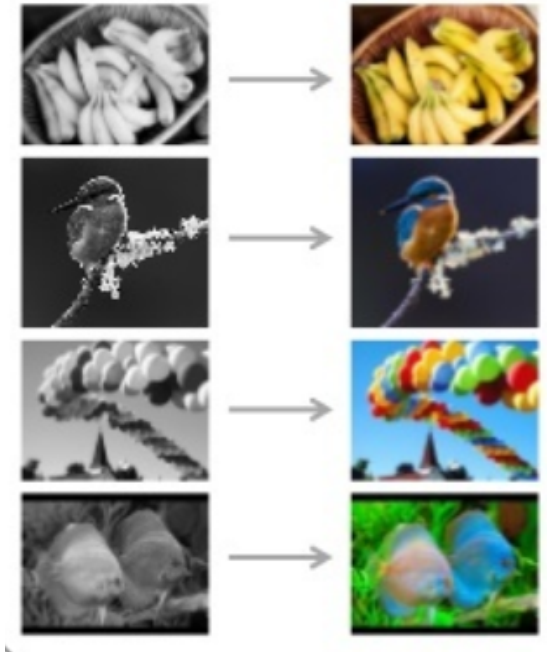
Image-to-Image Translation with Conditional Adversarial Networks

Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory, UC Berkeley

{isola,junyanz,tinghuiz,efros}@eecs.berkeley.edu





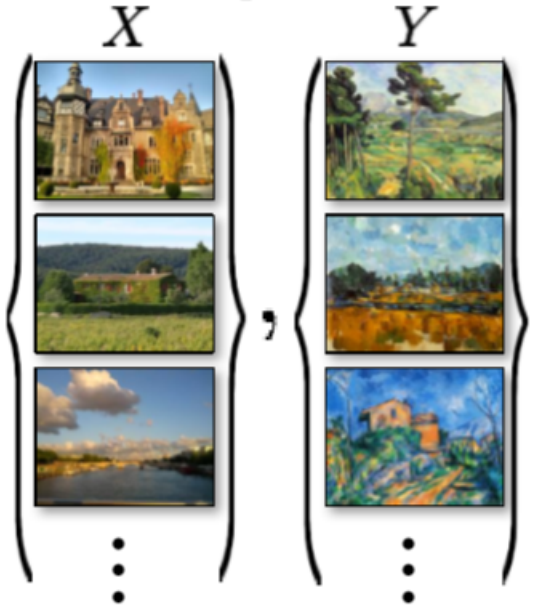
CycleGAN

Cycle consistency Adversarial Network

Paired

$\{x_i, y_i\}_{i=1}^{N}$

Unpaired

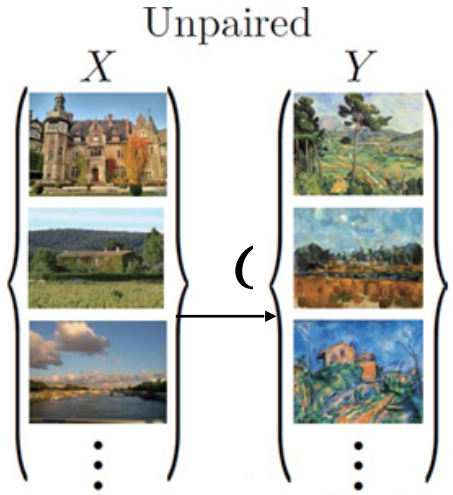


$\{x_i\}_{i=1}^{N}\ (x_i \in X)$

$\{y_j\}_{j=1}\ (y_j \in Y)$

CycleGAN

Cycle consistency Adversarial Network

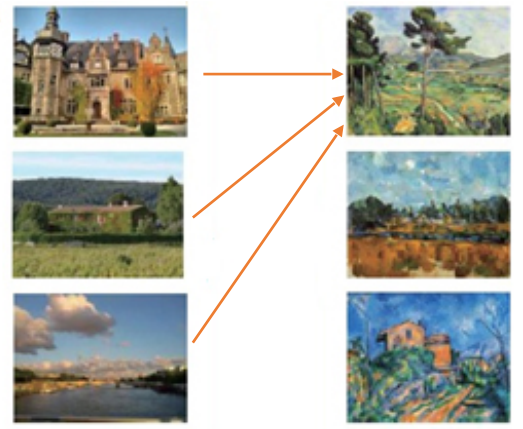


도메인X, Y 사이의 관계가 있다고 가정

Train mapping G

$$G : X \rightarrow Y$$

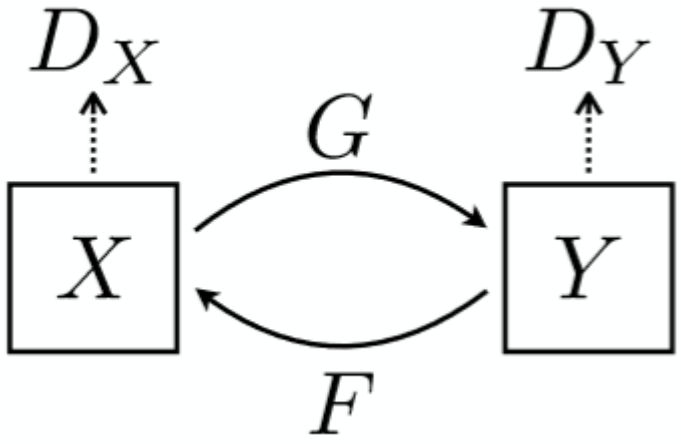
$$\hat{y} = G(x).$$



Mode collapse

CycleGAN

Cycle consistency Adversarial Network



Adversarial Loss

Cycle Consistency Loss

$$G : X \rightarrow Y \text{ and } F : Y \rightarrow X$$

Zebras ⇄ Horses

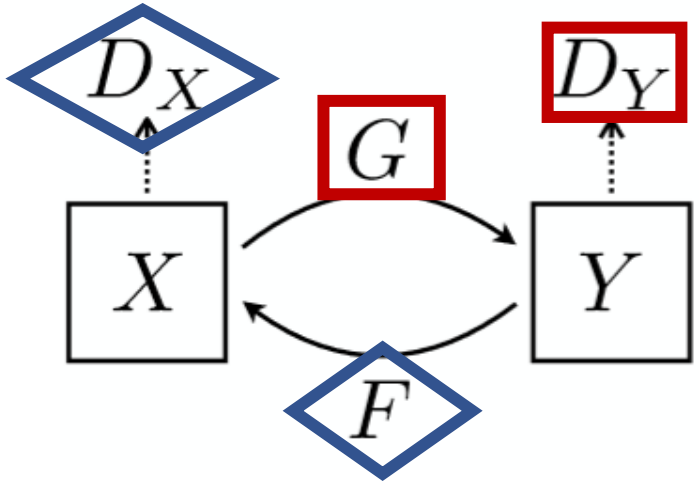
zebra → horse

horse → zebra

CycleGAN

Cycle consistency Adversarial Network

Adversarial loss



(1) $\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$

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

(2) $\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$.

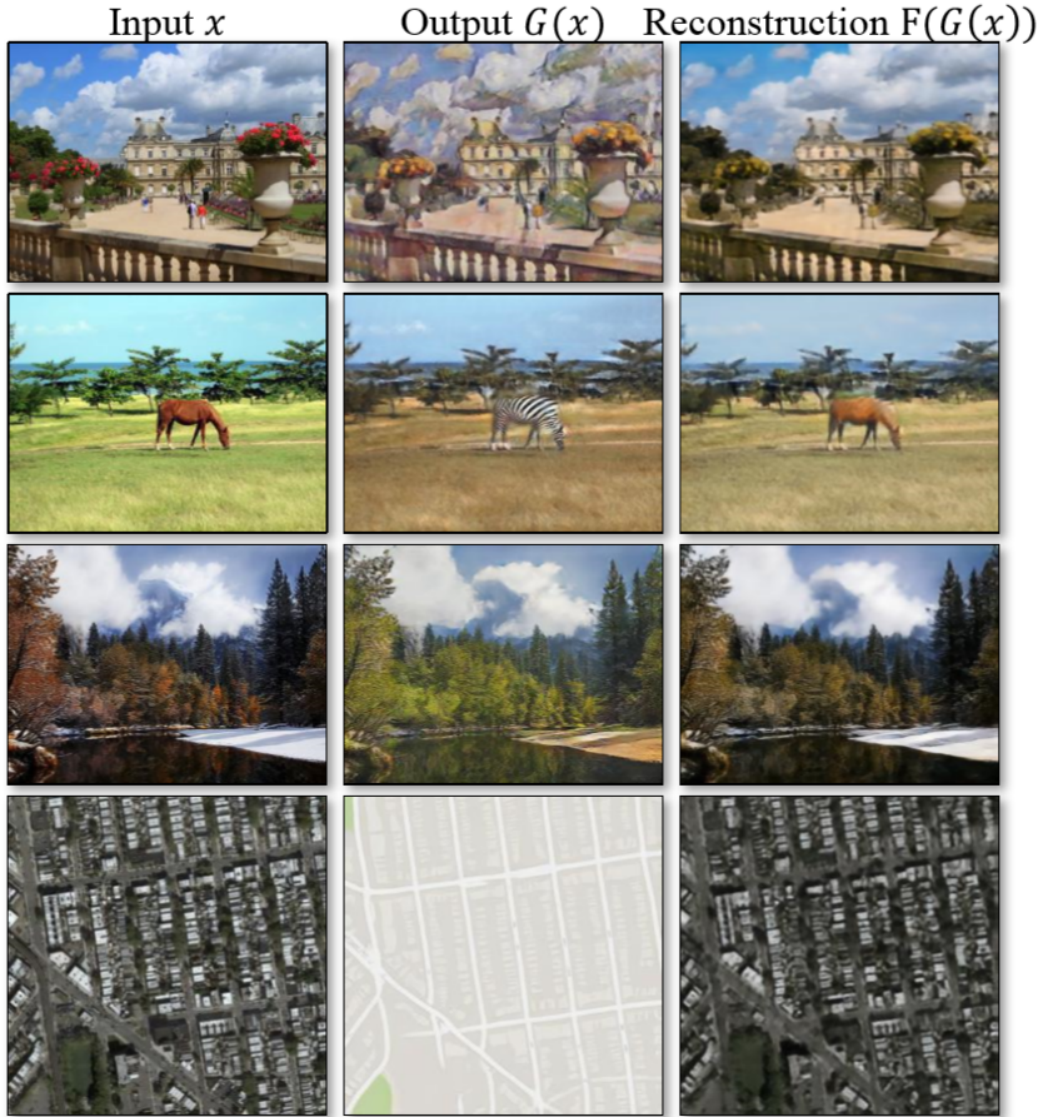
CycleGAN

Cycle consistency Adversarial Network

Cycle-consistency loss

$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$$
$$\shortparallel$$
$$\hat{x}$$

Forward cycle-consistency loss

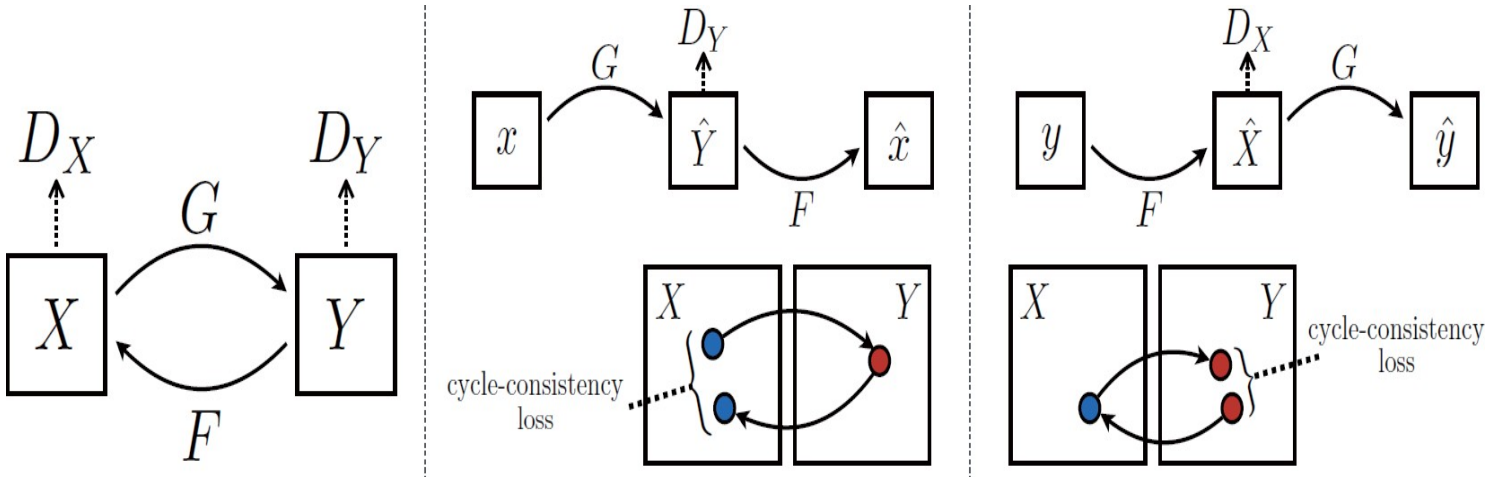


CycleGAN

Cycle consistency Adversarial Network

Cycle-consistency loss

$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x, \qquad y \rightarrow F(y) \rightarrow G(F(y)) \approx y$$

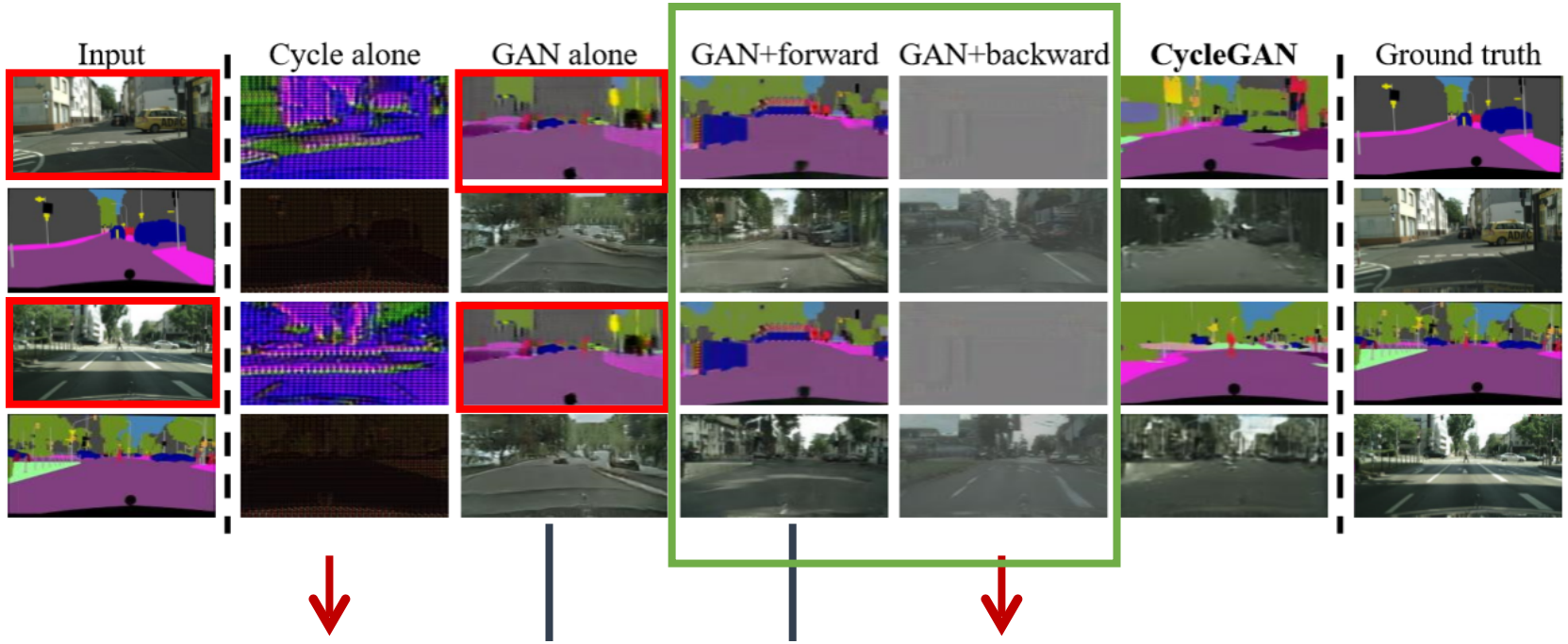


Forward Cycle Consistency Loss Backward Cycle Consistency Loss

Two Cycle Consistency Loss = Forward + Backward →

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

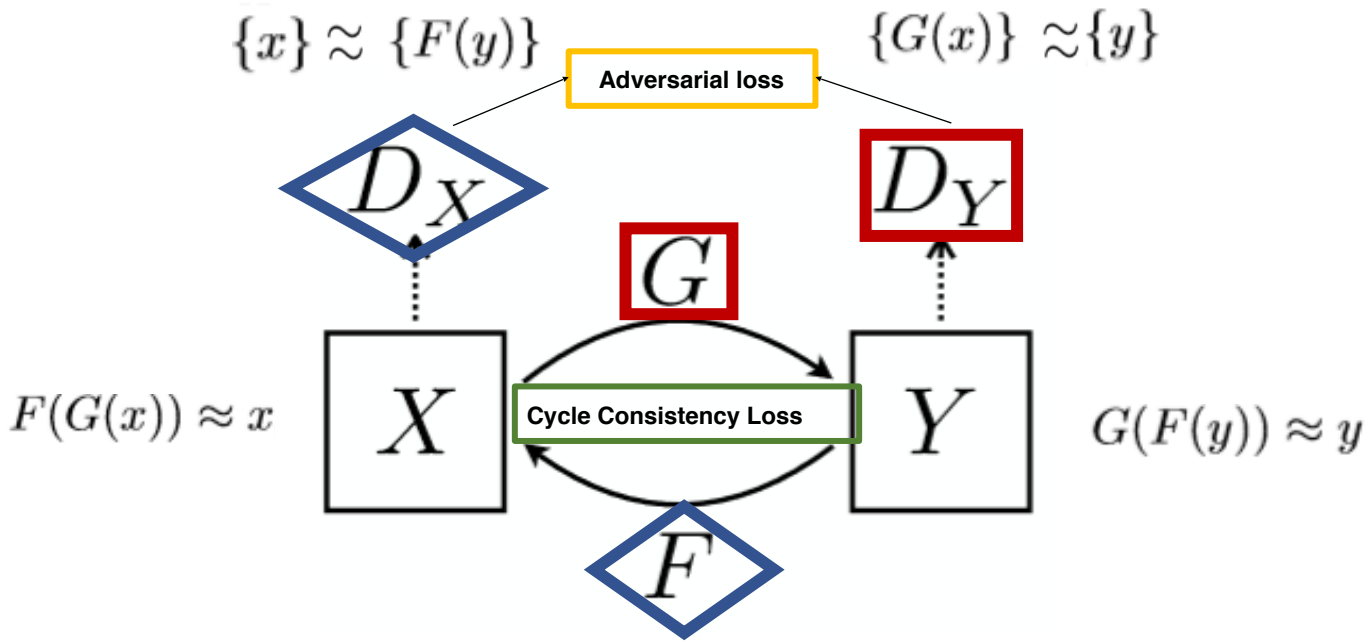
CycleGAN



target domain과 유사한 이미지 생성에 실패

Mode collapse 발생 : 동일한 output 생성

CycleGAN



$G : X \rightarrow Y$ and $F : Y \rightarrow X$

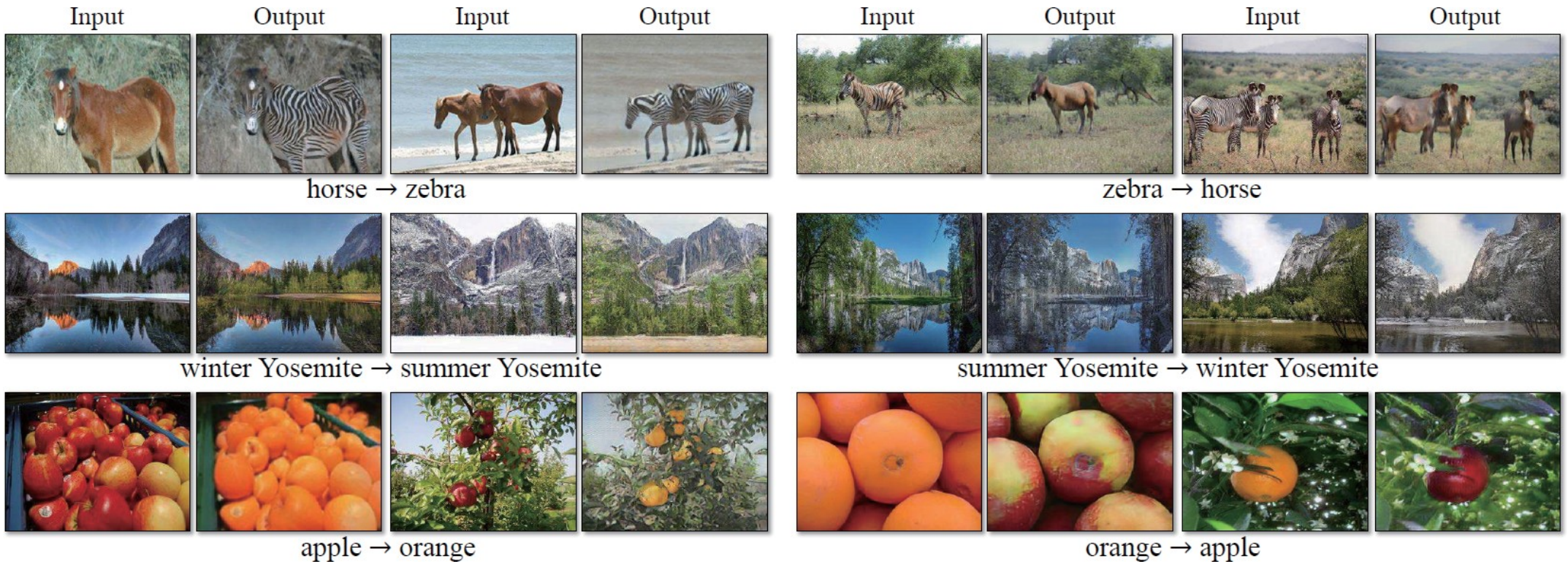
Total loss = Adversarial Loss + Cycle Consistency Loss

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

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

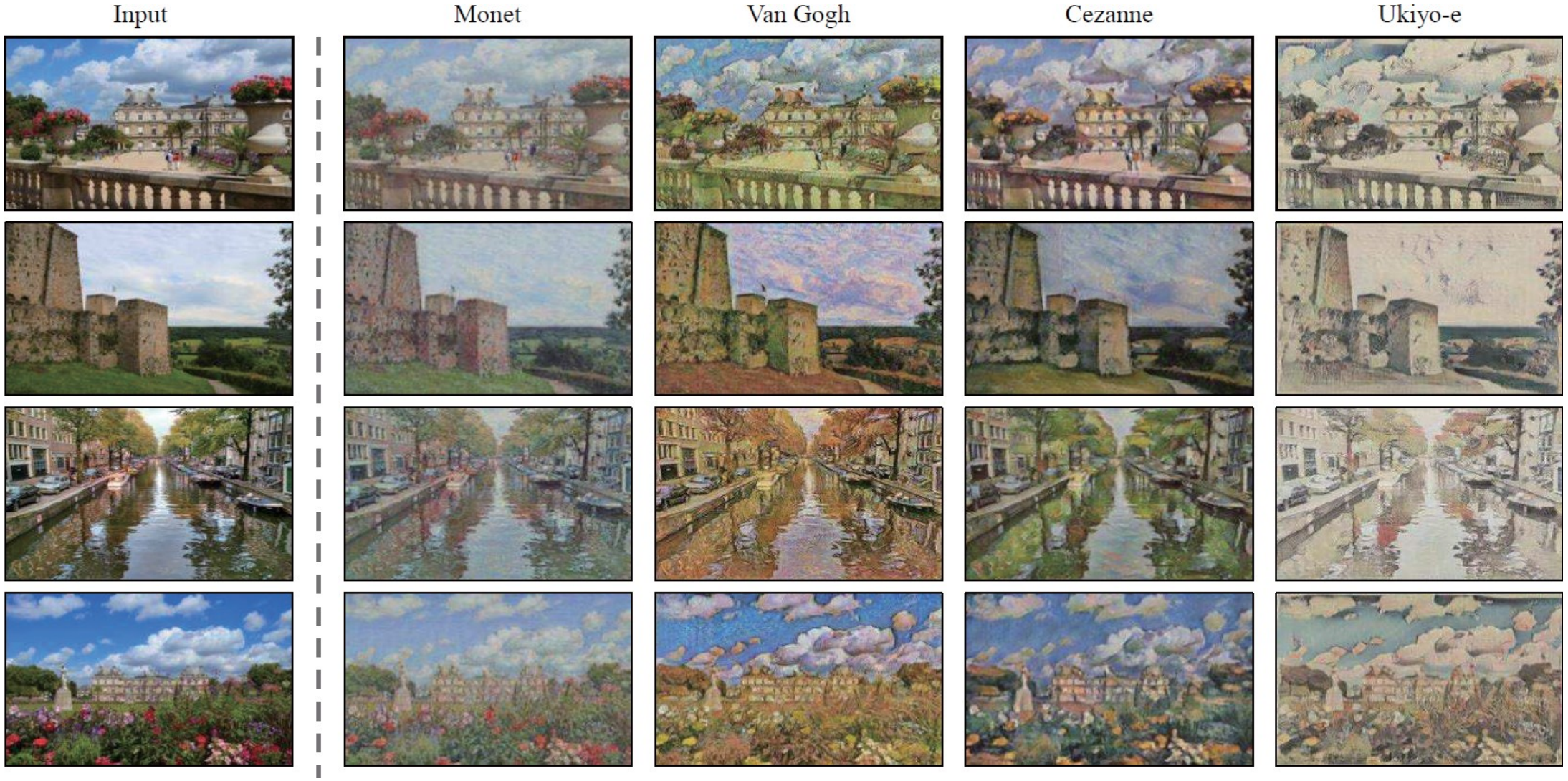
CycleGAN

Object transfiguration / Season transfer



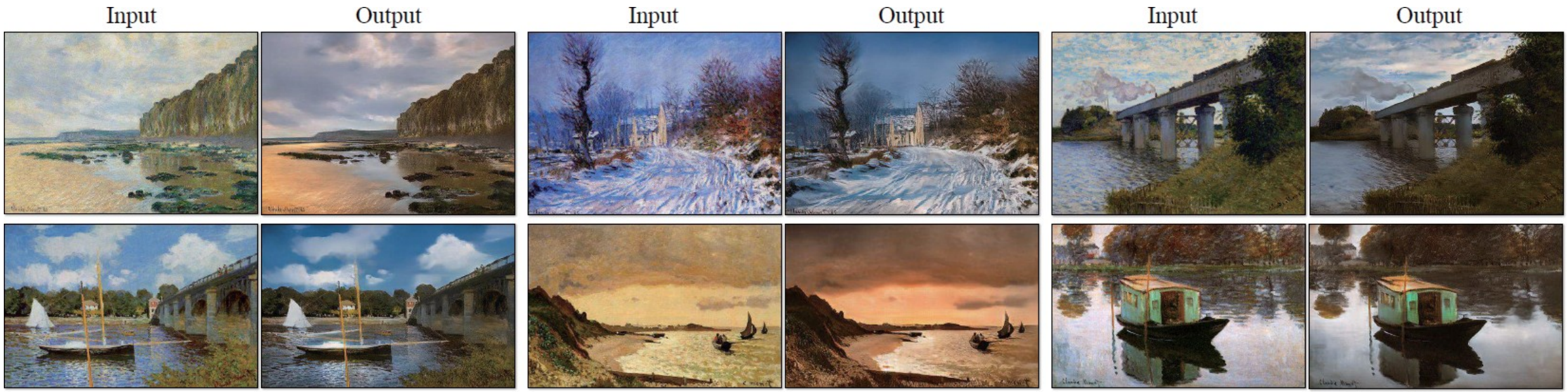
CycleGAN

Collection style transfer



CycleGAN

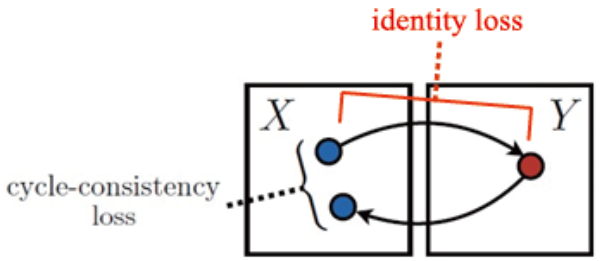
Monet paintings to photographs

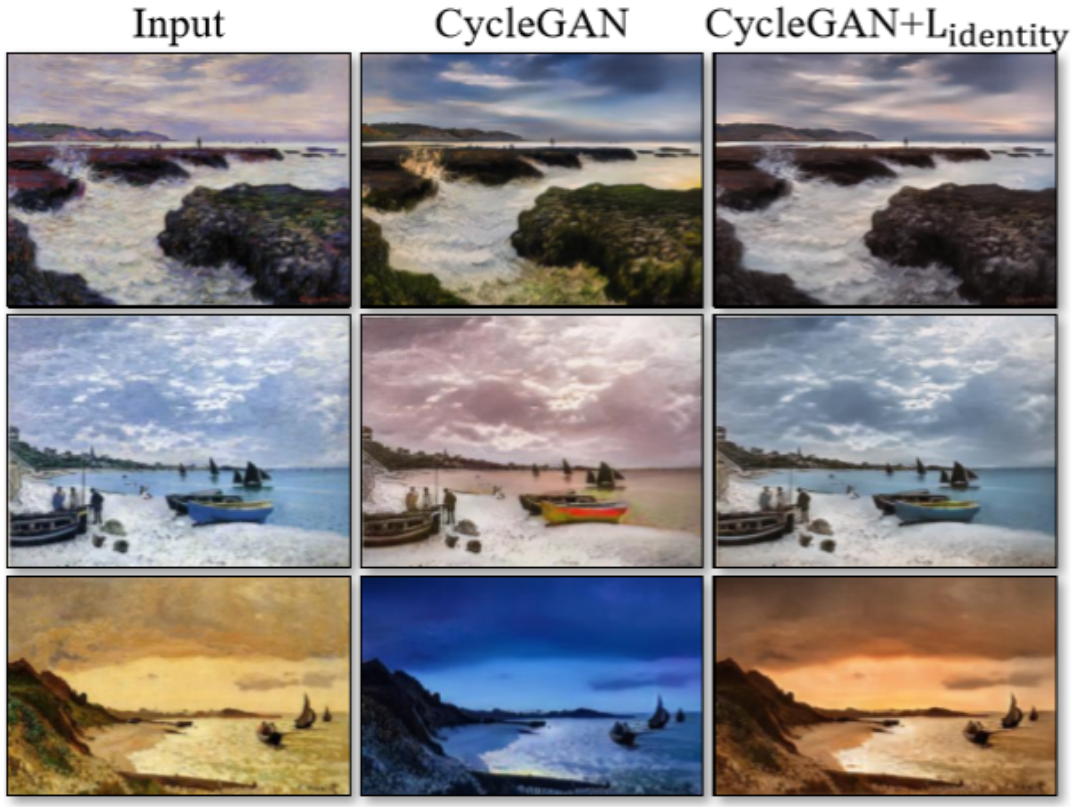


CycleGAN

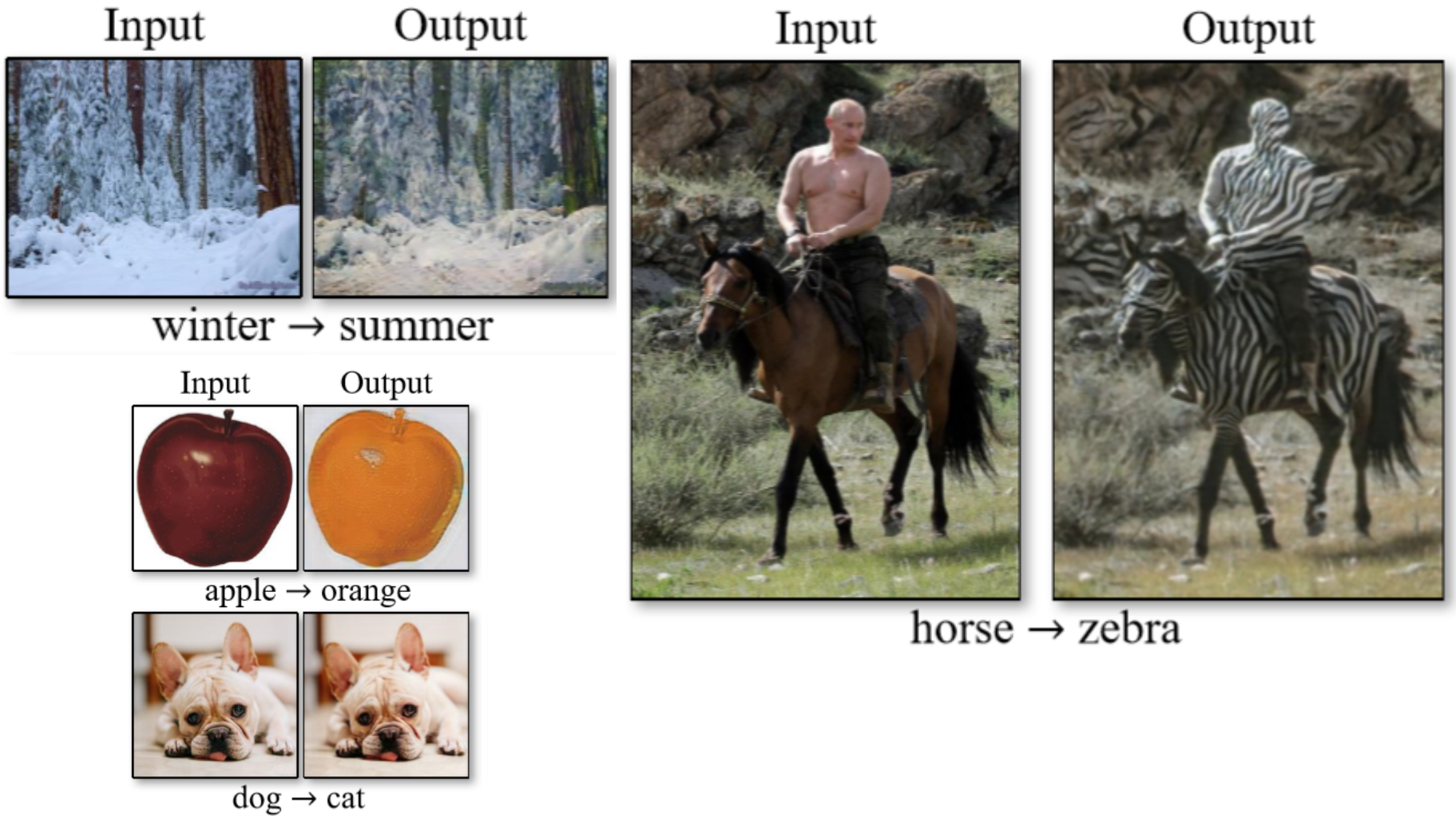
추가적인 Loss - identity

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





CycleGAN


Input
Output
winter → summer
Input
Output
apple → orange
dog → cat
Input
Output
horse → zebra

CONTENTS

CartoonGAN?

CartoonGAN: Generative Adversarial Networks for Photo Cartoonization

Yang Chen
Tsinghua University, China
chenyang15@mails.tsinghua.edu.cn

Yu-Kun Lai
Cardiff University, UK
Yukun.Lai@cs.cf.ac.uk

Yong-Jin Liu*
Tsinghua University, China
liuyongjin@tsinghua.edu.cn

CartoonGAN?

CartoonGAN?

1. 카툰 이미지의 높은 단순성과 모호함

2. 카툰 이미지의 분명한 선과 부드러운 음영

CartoonGAN?

1. Loss

(1) Edge-promoting adversarial loss

(2) Sematic content loss

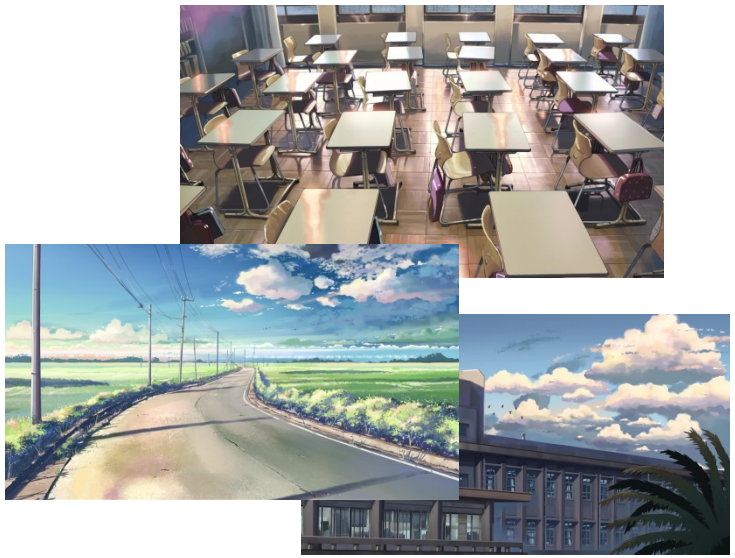
CartoonGAN?

2. Unpaired training data set

- 모든 훈련 이미지는 사이즈 256 X 256로 조정됨

Real world photo

Cartoon Image



With initialization phase

& VGG network

2. Unpaired training data set

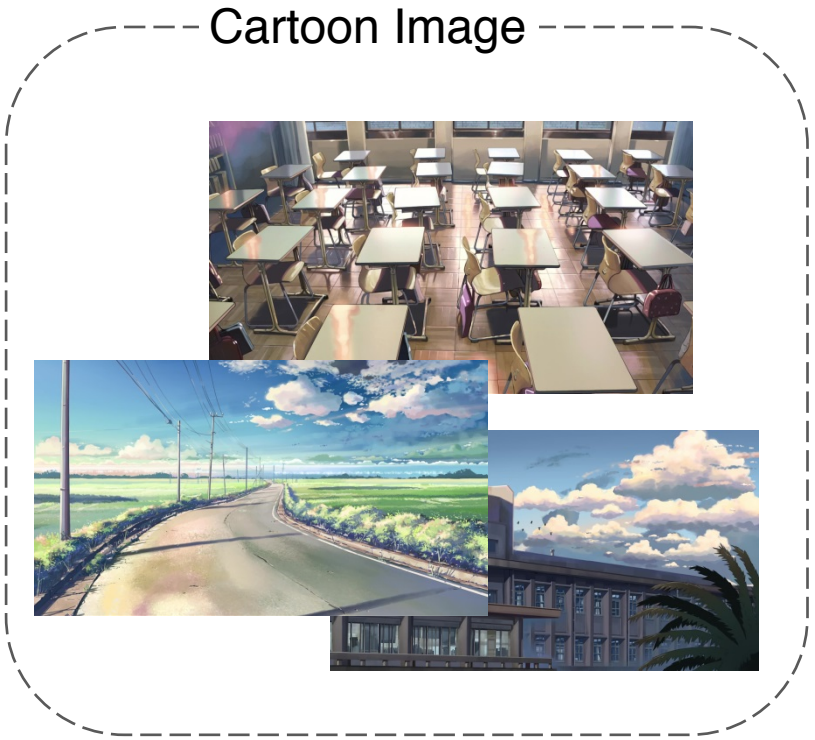
Real world photo

- 6,153장 의 사진이미지(Flickr)
- 5,401 장은 Training
- 752 장은 testing

2. Unpaired training data set

- 4,573 장의 카툰이미지
  (Makoto Shinkai 스타일화)

- 3,617 장의 카툰이미지
  (Miyazaki Hayao 스타일화)



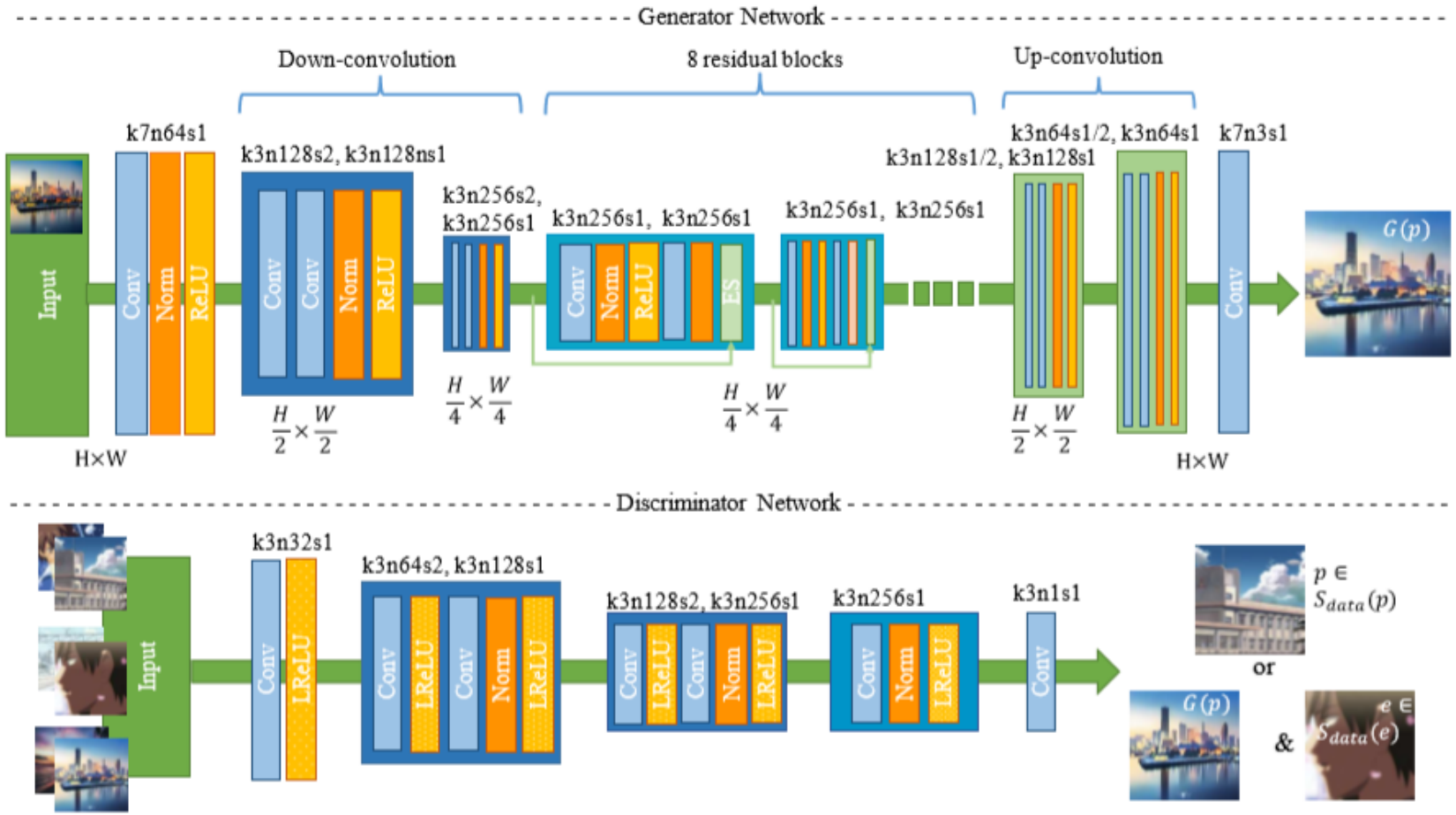
Contributions

1. 언페어 이미지 셋을 사용한 사진의 카툰화를 효율적으로 이루는 GAN 베이스의 접근

2. 두 개의 독특한 losses

3. 네트워크의 수렴을 향상시키는 Initialization phase의 도입

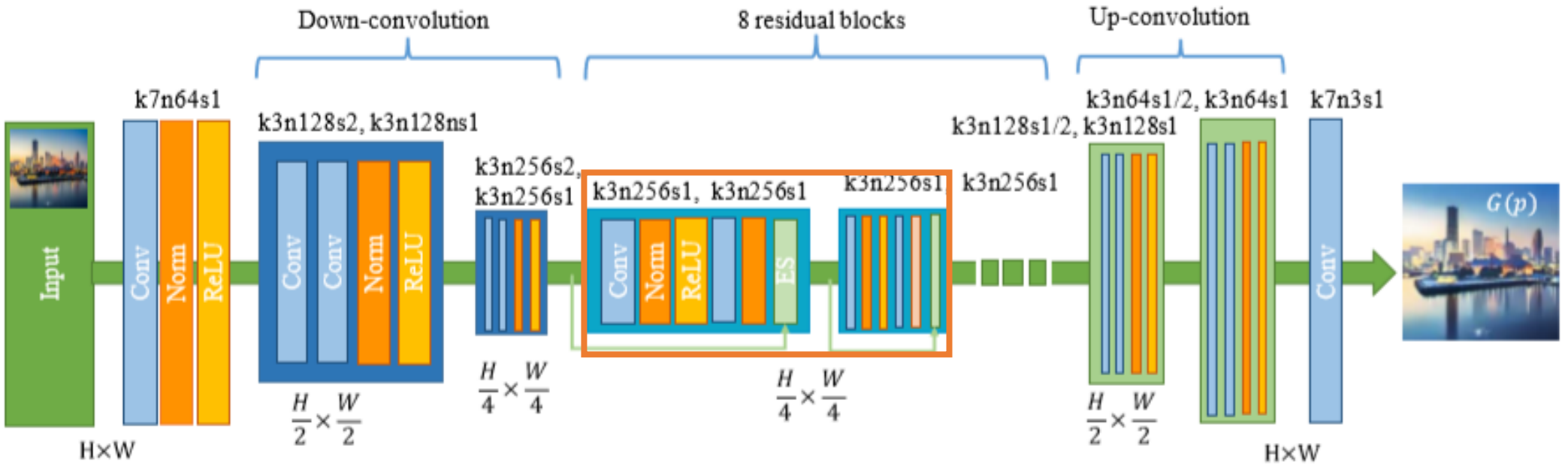
CartoonGAN?



$$(G^*, D^*) = \arg\min_G \max_D \mathcal{L}(G, D)$$

CartoonGAN?

1. Generator

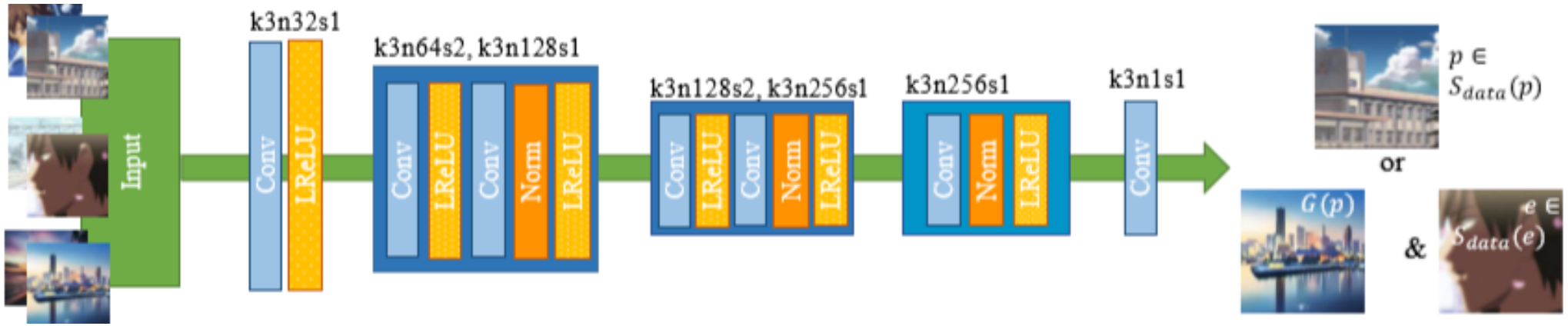


1) Residual block

2) Batch normalization

CartoonGAN?

2. Discriminator



1) patch-level discriminator 2) shallow 3) Leaky Relu

CartoonGAN?

Loss

$$\mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \omega \mathcal{L}_{con}(G, D)$$

1. **Adversarial loss** $\mathcal{L}_{adv}(G, D)$

2. **Content loss** $\mathcal{L}_{con}(G, D)$

$$\omega = 10$$

Loss

1. Adversarial loss $\mathcal{L}_{adv}(G, D)$

- Generator network 가 이미지를 변환 시키도록 하는 Loss
- networks G and D 에 적용됨
- G의 output이 얼마나 카툰 이미지 같은가를 나타냄

Real photo

G

Generated cartoon image

적절한 음영 (O)

분명한 선 (X)

D

CartoonGAN?

Loss

1. Edge-promoting adversarial lo$\mathcal{L}_{adv}(G, D)$

$$\mathcal{L}_{adv}(G, D) = \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)]$$
$$+ \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))]$$
$$+ \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))].$$

Cartoon image ->1

Edge smoothed ->0

Generated Cartoon image ->0

CartoonGAN?

Loss

$$S_{data}(e) = \{e_i \,|i = 1 \ldots M\} \subset \mathcal{E}$$

(Removing clear edges in $c_i \in S_{data}(c)$)

(a) A cartoon image c_i (b) The edge-smoothed version e_i

(1) 선 요소를 찾음

(2) 선 요소를 삭제함

(3) Gaussian smoothing을 적용함

CartoonGAN?

Loss

1. Edge-promoting adversarial lo$\mathcal{L}_{adv}(G, D)$

$$\mathcal{L}_{adv}(G, D) = \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)]$$
$$+ \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))]$$
$$+ \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))].$$

Cartoon image ->1

Edge smoothed ->0

Generated Cartoon image ->0

CartoonGAN?

Loss

1. Edge-promoting adversarial lo$\mathcal{L}_{adv}(G, D)$

(a) Input Photo

(b) Without edge loss **VS** (c) CartoonGAN

Loss

2. Content loss $\mathcal{L}_{con}(G, D)$

- input photos의 내용을 유지함
- VGG_l: feature maps of a specific VGG layer (‘conv4_4’)

$$\mathcal{L}_{con}(G, D) = \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]$$

Features of generated image

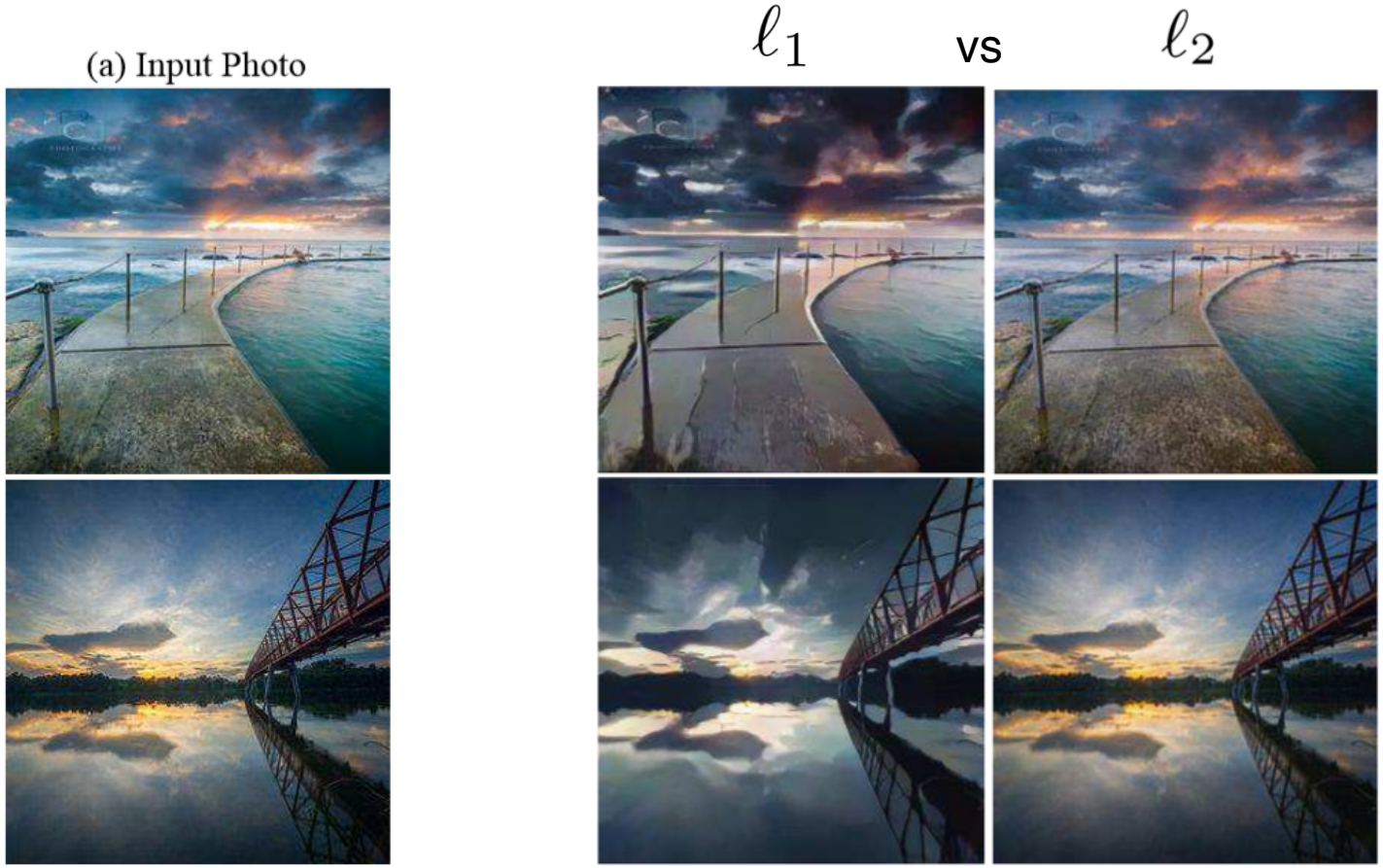
Features of photo

- ℓ_1 : 지역적으로 스타일 차이가 큰 부분들의 영향을 적게 받음

CartoonGAN?

Loss

2. Content loss $\mathcal{L}_{con}(G, D)$



CartoonGAN?

Initialization phase

(a) Original photo

(b) Image after initialization

- GAN model 은 매우 비선형적임
- 잠재적 local minimum을 방지함
- 네트워크의 수렴을 도움
- Framework의 학습을 시작할 때, G가 input images의 내용만을 재생성 하도록 선행 훈련 시킴

Method

- the semantic content loss $\mathcal{L}_{con}(G, D)$ 만을 이용하여 G를 선행훈련 시킴

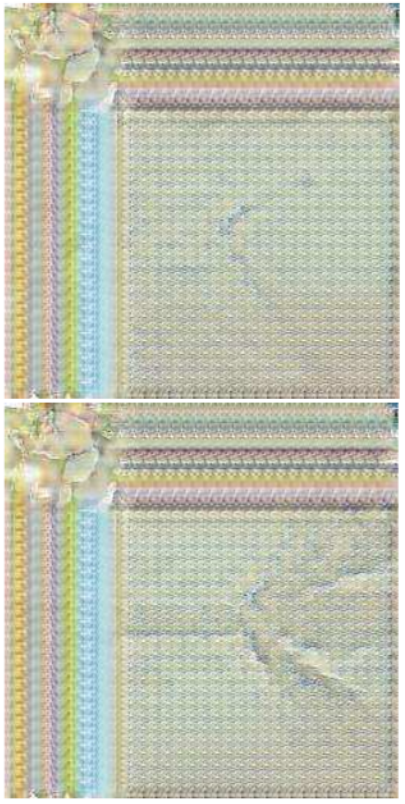
CartoonGAN?

Initialization phase

initialization 10 epochs or none

(a) Input Photo

(b) Without Initialization



(C) CartoonGAN(ours)

Comparison with CycleGAN

CycleGAN

- 두 종류의 버전으로 테스트

1. CycleGAN (without identity loss $L_{identity}$)
2. with identity loss $L_{identity}$

- $L_{identity}$: 내용(색상 등)을 잘 유지시킴

- CycleGAN와 CartoonGAN 모두 200 epochs 훈련시킴

CartoonGAN?

Comparison with CycleGAN

Result (for Miyazaki Hayao style)

Input Photo | (a) CycleGAN | (b) CycleGAN(+*Lidentity*) | (c)CartoonGAN

(a) input photo의 내용을 잘 유지하지 못함

(b) 내용을 유지하나, 스타일화의 결과가 좋지 못함

(c) 사용된 카툰 작가의 개인의 스타일을 높은 수준으로 카툰화하여 이미지를 재생성함

Comparison with CycleGAN

Result (for Miyazaki Hayao style)

(b) CycleGAN(+*Lidentity*)

For each epoch,
3020.31 s

Train two
GAN models

내용의 유지를 위해
cycle architecture
를 사용

For each epoch,
1517.69s

Train one
GAN model

내용의 유지를 위해
VGG feature maps
을 사용

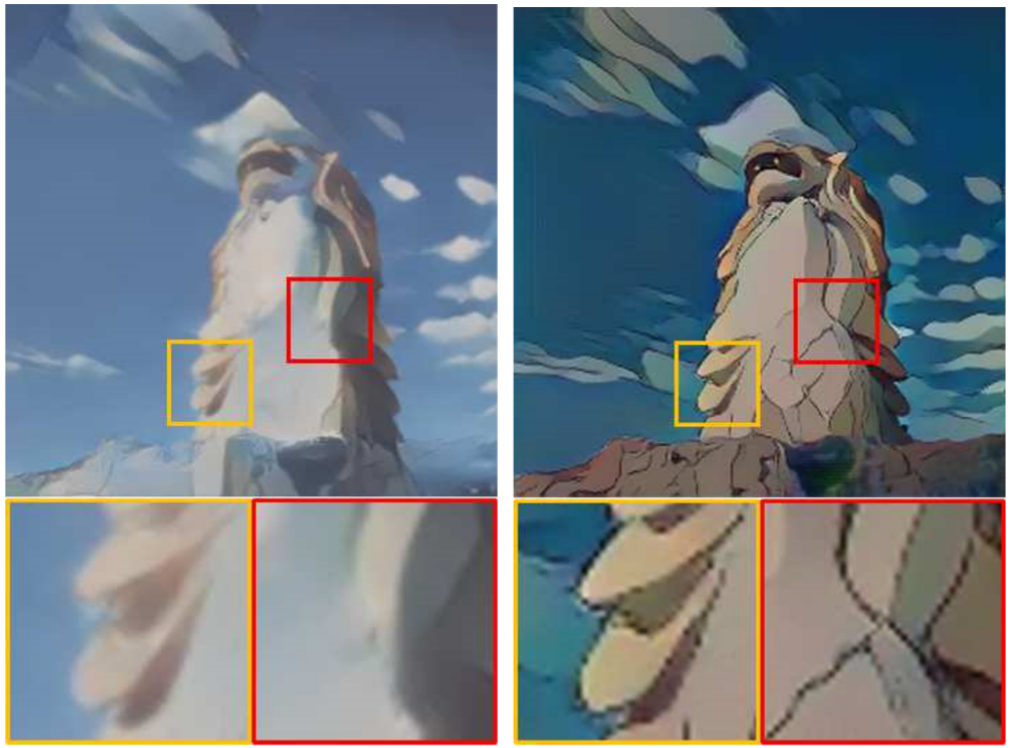
(c)CartoonGAN

CartoonGAN은 보다 효율적인 cartoon stylization 학습이 이루어지도록 함

CartoonGAN?

Comparison with CycleGAN

Result (for Miyazaki Hayao style)



(a) CycleGAN(+*Lidentity*)

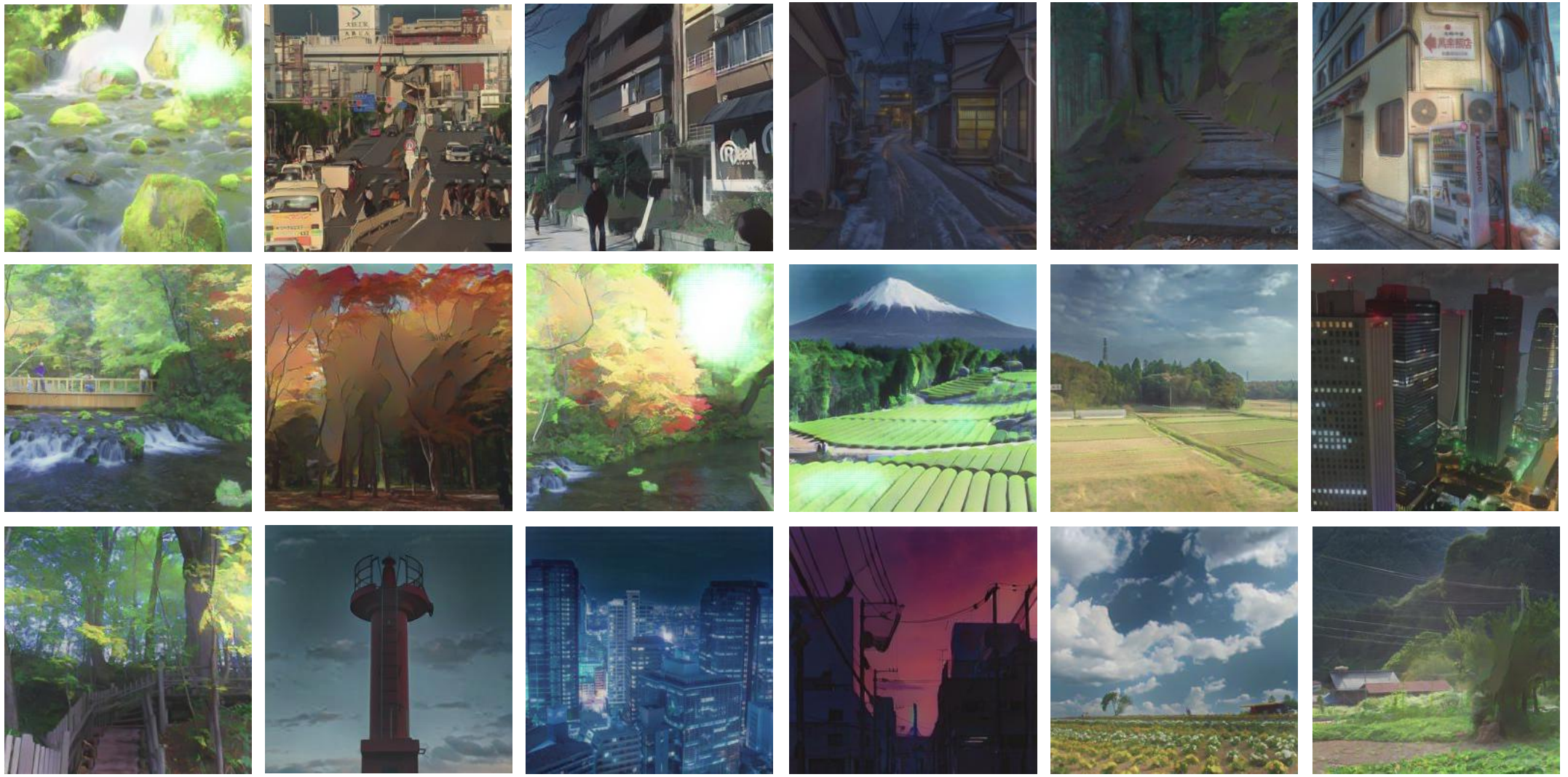
(b) CartoonGAN

(b) 카툰 스타일에 매우 중요한 특징인 분명한 가장자리 선이 잘 생성됨

CONTENTS

CartoonGAN 성능 검증

CartoonGAN 생성 이미지



CartoonGAN 성능 검증

GAN의 성능을 어떻게 평가할 것인가?

1. 생성된 이미지들을 확인(객관성이 떨어짐)

   → 설문을 활용하여 객관성을 어느정도 보장할 수 있음

2. Inception Score

3. FID(Frechet Inception Distance)

CartoonGAN 성능 검증(평가 방법)

1. 설문조사

Amazon에서 제공하는 데이터 수집 플랫폼 AMT를 이용하려 했으나 동일한 방식에 추가적 비용이 없는 구글 설문을 이용

→ GAN으로 생성된 이미지의 퀄리티를 다수의 사람들이 평가하게 함

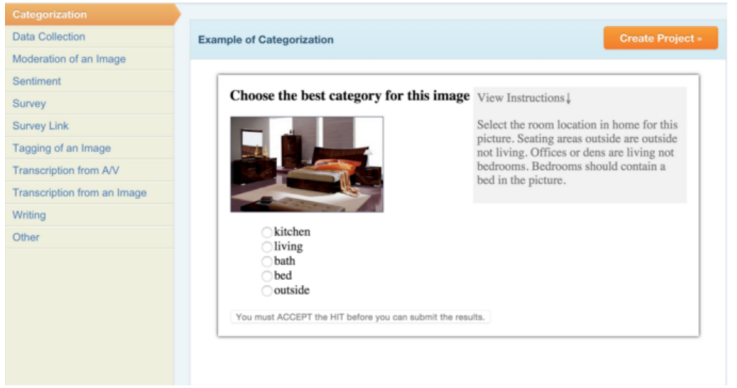
1. 실제 애니메이션 이미지 vs CartoonGAN vs CycleGAN(13문항)
2. CartoonGAN vs CycleGAN(11문항)

✓ 장점:

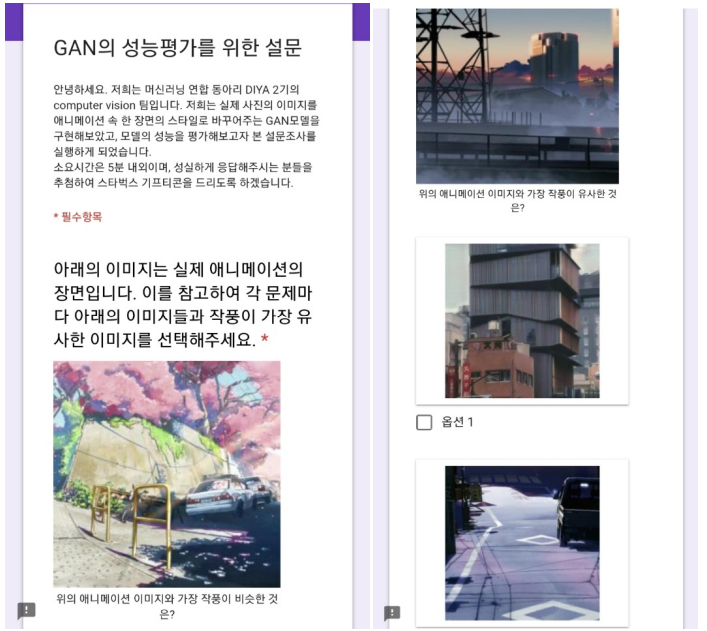
다수의 평가를 통해 어느 정도의 객관성을 보장할 수 있음

✓ 단점:

생성된 이미지의 일부만을 평가할 수 있고, mode collapse가 발생한 모델의 문제점을 파악하기 어려움



AMT설문 예시



구글 설문 예시

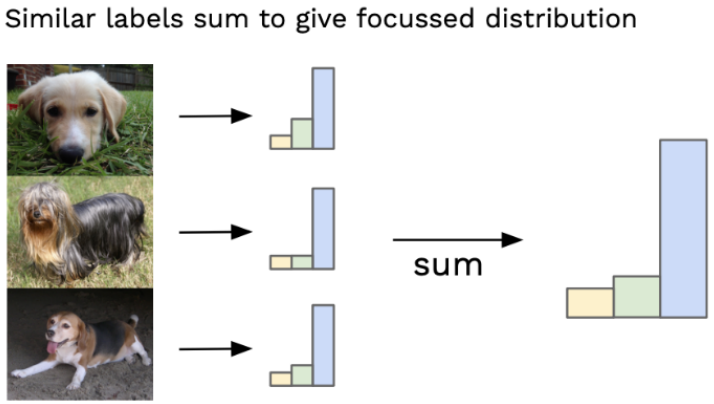
CartoonGAN 성능 검증(평가 방법)

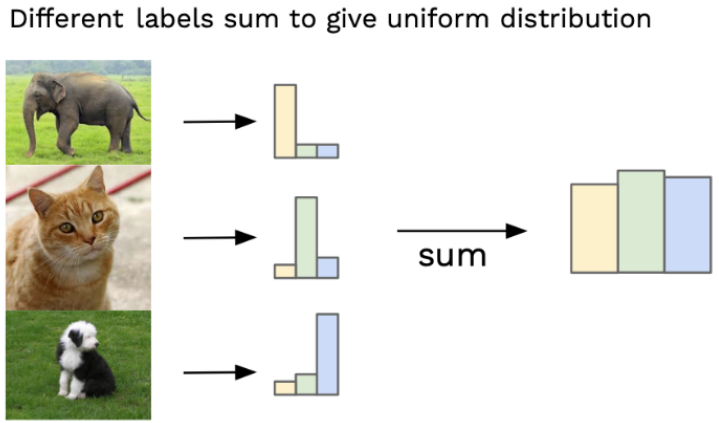
2. Inception Score

사전에 학습된 구글의 인셉션 이미지 분류 모델에 GAN으로 생성된 이미지를 넣어서 나오는 값을 바탕으로 모델을 평가

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right)$$

* Kullback-Leibler Divergence 로 두 분포가 얼마나 다른 지 나타냄





가 커질 수록 높은 값을 갖게 됨

But,
1. Real sample과의 비교가 아님

2. GAN이 레이블 마다 하나의 이미지만을 생성한다면(mode collapse)?

CartoonGAN 성능 검증(평가 방법)

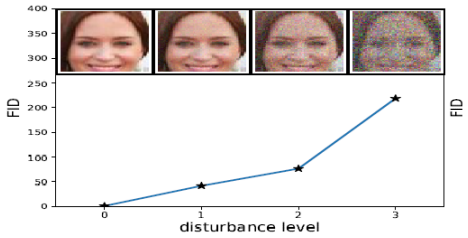
3. FID(Frechet Inception Distance)

feature extraction을 하기 위해서 inception network를 사용

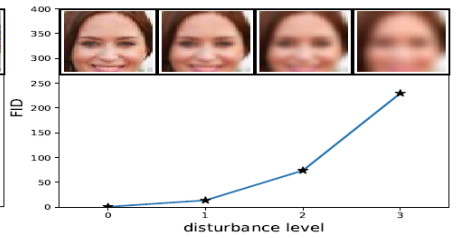
위에서 얻은 feature들을 이용하여 real data()와 generated data()의 multivariate gaussian distribution을 평균()과 공분산행렬()로 각각 추정. 다음과 같이 값을 계산함(두 m.g.d.이 얼마나 다른지 측정-거리)

작은 값일 수록 좋은 이미지 퀄리티와 다양성을 보장 함(예시)

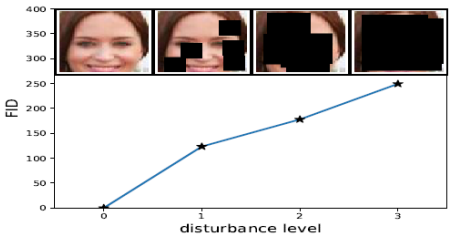
$$\mathrm{FID}(x, g) = ||\mu_x - \mu_g||_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x\Sigma_g)^{\frac{1}{2}})$$



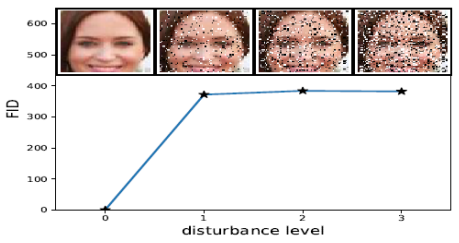
Gaussian Noise



Gaussian blur



Black rectangle



Salt & pepper noise

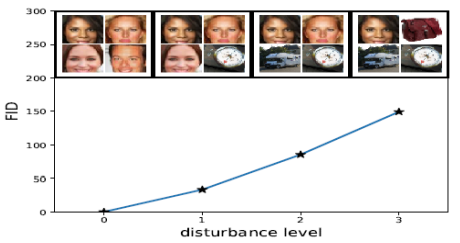


Image net contamination

CartoonGAN 성능 검증

CartoonGAN v.s. CycleGAN

Stylization이 가능한 CycleGAN을 활용하여, CartoonGAN의 성능 검증

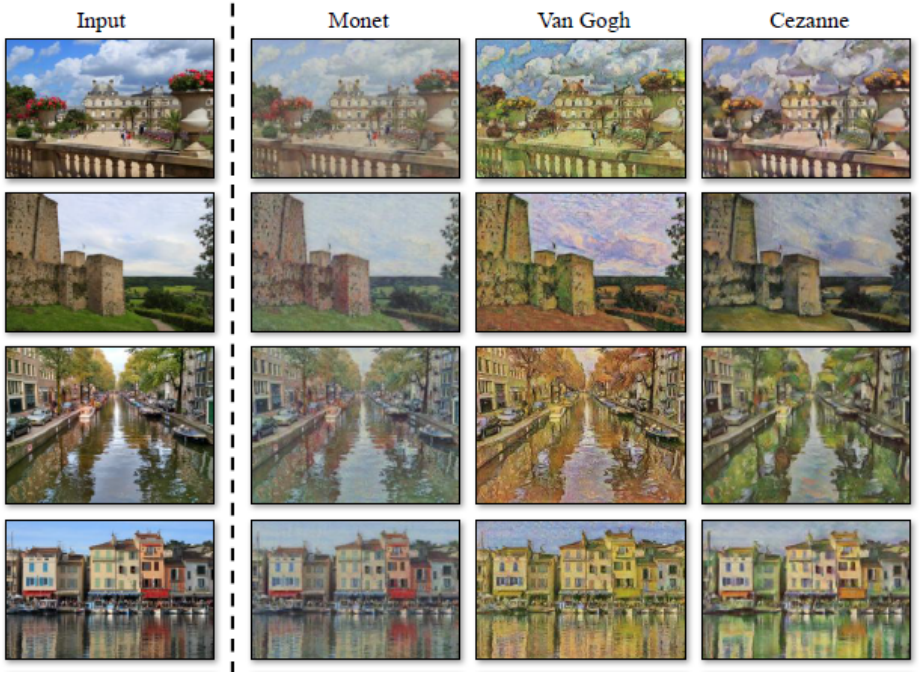
CartoonGAN이 더 나은 성능을 보인다면 무엇 때문인지?

1. CartoonGAN v.s. CycleGAN — 설문 + FID 측정
2. CartoonGAN v.s. CycleGAN + edge smoothed animation data — FID 측정
3. CartoonGAN v.s. CycleGAN + initialization phase — FID 측정

CartoonGAN 성능 검증

왜 CycleGAN인가?

1. Unpaired 데이터 셋으로 Stylization이 가능한 대표적인 GAN



2. 유사한 구조

Content loss(CartoonGAN)와 CycleConsistency loss(CycleGAN)의 기능이 유사하여 전체 목적함수가 유사한 구조(GAN loss + 이미지의 형태를 보존하기 위한 loss)를 취함

CartoonGAN의 technique들을 CycleGAN에 적용하면서 각각의 영향을 평가할 수 있음

$$\mathcal{L}_{con}(G,D) = \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]$$

← content loss

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

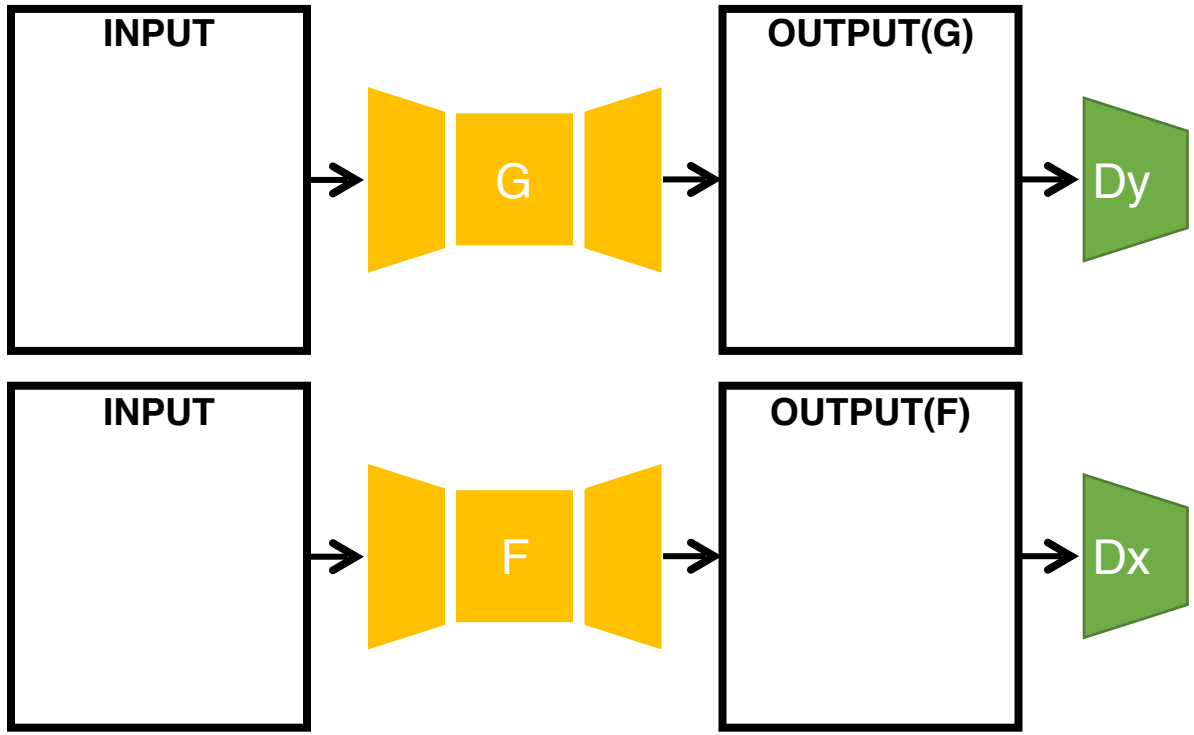
← cycle

CartoonGAN 성능 검증

CycleGAN의 cartoon stylization

1. 두 개의 Generator G와 F에 각각 인풋 이미지를 넣어 아웃풋 이미지를 생성(50개의 이미지 풀)

   단, G : 실제 사진 → 애니메이션, F : 애니메이션 → 실제 사진

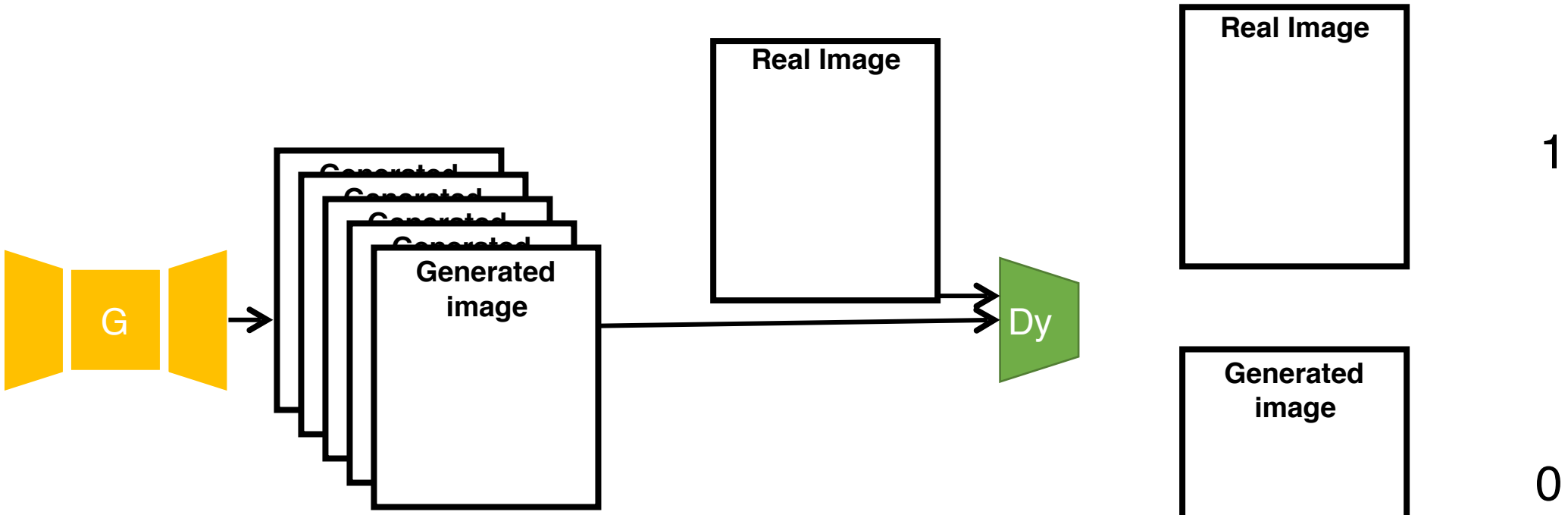


CartoonGAN 성능 검증

CycleGAN의 cartoon stylization

2. Discriminator Dy, Dx 는 생성된 이미지들은 0, 실제 이미지들은 1을 출력하도록 학습.

Generator는 D(G(x))가 1이 되도록 학습

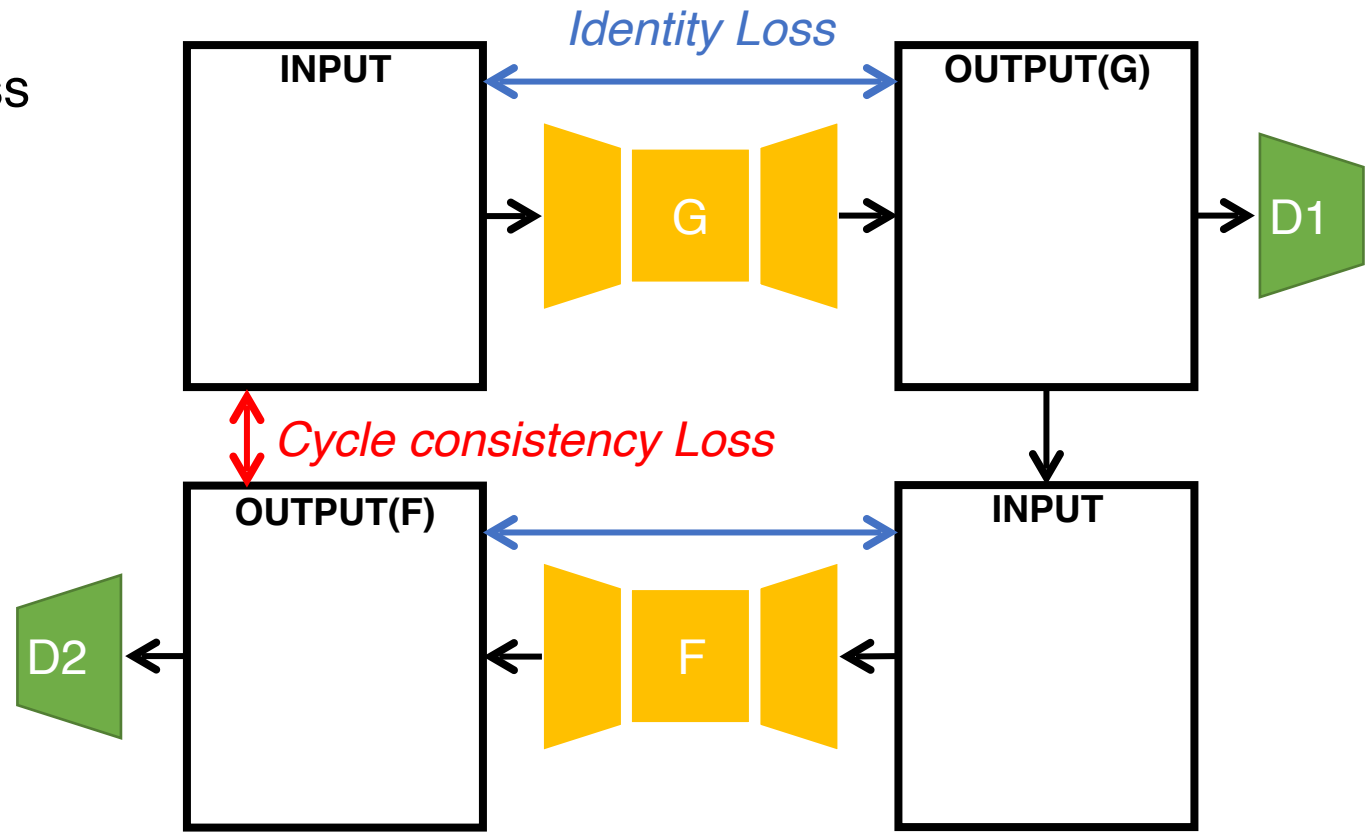


$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

CartoonGAN 성능 검증

CycleGAN의 cartoon stylization

3. Cycle consistency loss
와 identity loss를 계산



Cycle consistency Loss

Identity Loss

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

+

CartoonGAN 성능 검증

CycleGAN 생성 이미지

CartoonGAN 성능 검증

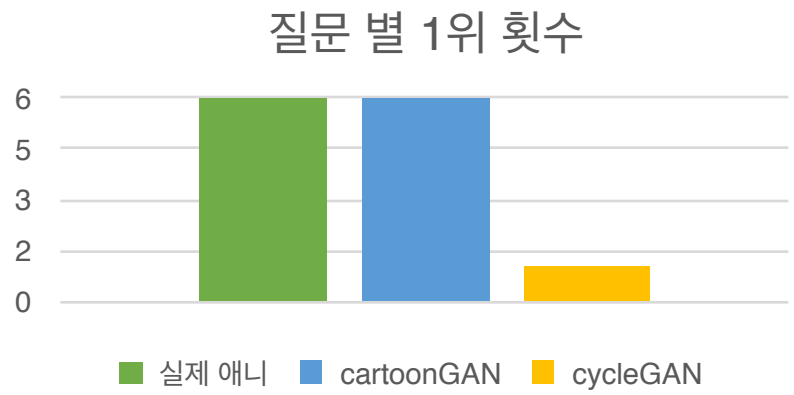
CycleGAN 생성 이미지

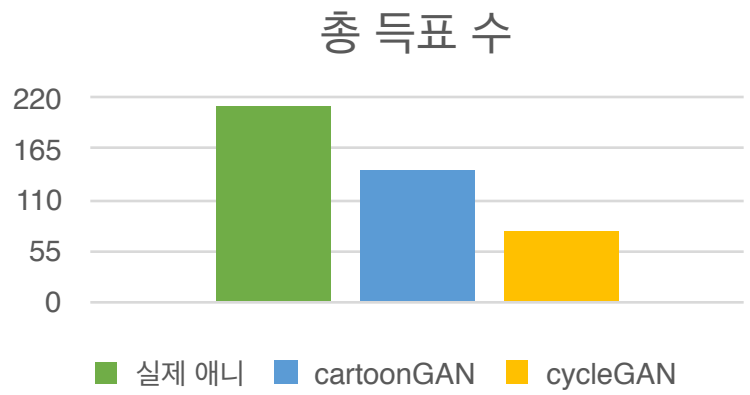
CartoonGAN 성능 검증

CycleGAN 생성 이미지

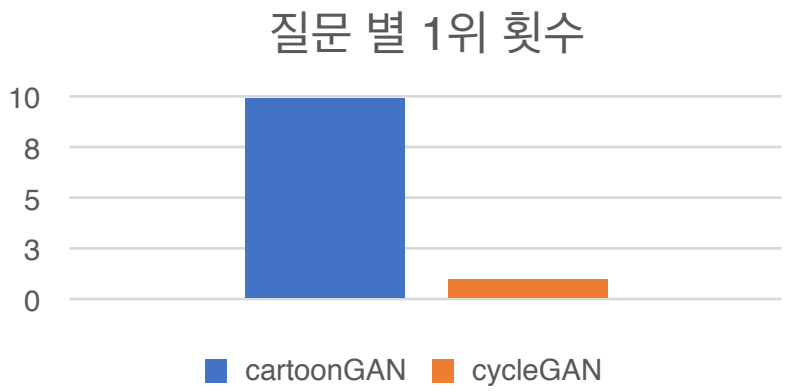
CartoonGAN 성능 검증 – 설문 조사(총 38명 참여)

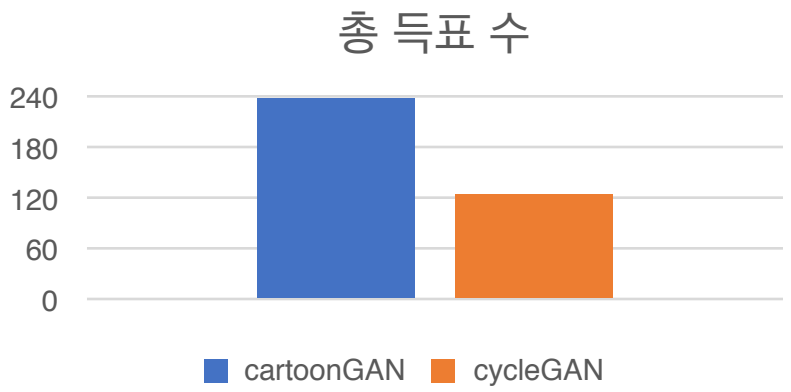
설문조사 결과 1. 실제 애니메이션 이미지 vs CartoonGAN vs CycleGAN





설문조사 결과 2. CartoonGAN vs CycleGAN





CartoonGAN 성능 검증

FID 측정 결과

CartoonGAN과 CycleGAN이 생성한 애니메이션 이미지에 대해서 FID score 값을 계산한 결과

FID with ani : 실제 애니메이션 도메인의 m.g.d.과 생성된 이미지의 m.g.d.의 거리(작을수록 우수)

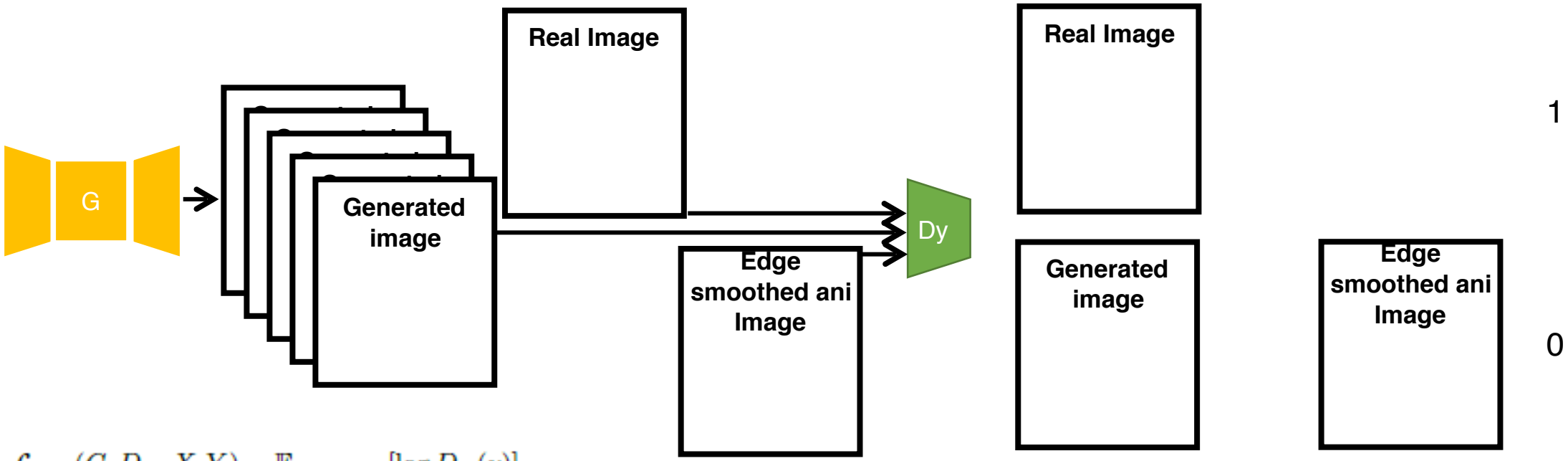
FID with photo : 실제 사진 도메인의 m.g.d.과 생성된 이미지의 m.g.d.의 거리(클수록 우수)

	CartoonGAN	CycleGAN
FID with ani	**100.2994251**	108.6875588
FID with photo	**80.95636448**	76.12667691

CartoonGAN 성능 검증

CycleGAN의 cartoon stylization + edge smoothed animation data

기존 CycleGAN 을 학습하는 과정에서 추가적으로 edge smoothed animation data에 대해서
Discriminator가 0을 출력하도록 학습

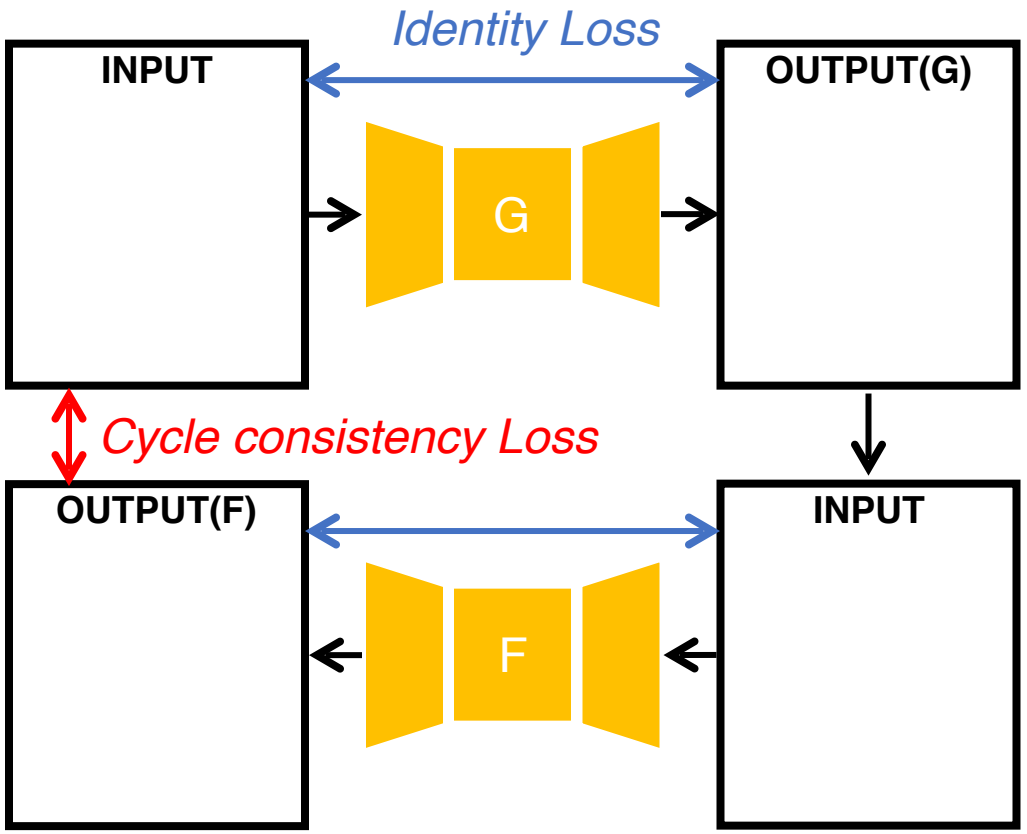


$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))] \text{ “} + \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))] \text{”}$$

CartoonGAN 성능 검증

CycleGAN의 cartoon stylization + initialization phase

기존 CycleGAN 을 학습하기 이전에 cycle consistency loss와 identity loss에 대해서 15epoch씩 학습함



Cycle consistency Loss

Identity Loss

+

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

CartoonGAN 성능 검증

CartoonGAN과 CycleGAN training details.(정리)

CartoonGAN 및 CycleGAN의 4가지 모델을 아래와 같이 학습시킴

CartoonGAN	CycleGAN	CycleGAN with edge smoothed	CycleGAN with initialization
initialization epochs : 15 training epochs : 100 batch size: 8 Adam optimizer with lr=0.0002, beta1=0.5 content_loss_weight=10	training epochs : 100 batch size : 4 lambda cycle=10 lambda identity=0.5 Adam optimizer with lr=0.0002, beta1=0.5	training epochs : 100 batch size : 4 lambda cycle=10 lambda identity=0.5 Adam optimizer with lr=0.0002, beta1=0.5	initialization epochs : 15 training epochs : 100 batch size : 4 lambda cycle=10 lambda identity=0.5 Adam optimizer with lr=0.0002, beta1=0.5 content loss weight=10

CartoonGAN 성능 검증

FID 측정 결과

CartoonGAN과 CycleGAN이 생성한 애니메이션 이미지에 대해서 FID score 값을 계산한 결과

FID with ani 는 실제 애니메이션 도메인의 m.g.d.과 생성된 이미지의 m.g.d.의 거리(작을수록 우수)

FID with photo 는 실제 사진 도메인의 m.g.d.과 생성된 이미지의 m.g.d.의 거리(클수록 우수)

	CartoonGAN	CycleGAN	CycleGAN with edge smoothed	CycleGAN with initialization
FID with ani	**100.2994251**	108.6875588	108.6880403	**108.1116794**
FID with photo	**80.95636448**	76.12667691	77.17832337	**77.9440937**

*CycleGAN에 initialization phase를 넣은 것과 edge smoothed 애니메이션 이미지의 loss를 추가한 것의 효과는 미미함

CartoonGAN 성능 검증

CycleGAN with edge-smoothed animation image

CycleGAN은 CartoonGAN보다 loss함수가 더 복잡한 구조를 가짐 → 추가한 loss의 영향이 비중이 있는지?

CartoonGAN*의 *Loss

$$\mathcal{L}(G,D) = \mathcal{L}_{adv}(G,D) + \omega \mathcal{L}_{con}(G,D)$$

$$\mathcal{L}_{adv}(G,D) = \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)] + \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))] + \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))]$$

$$\mathcal{L}_{con}(G,D) = \mathbb{E}_{p_i \sim S_{data}(p)}[||VGG_l(G(p_i)) - VGG_l(p_i)||_1]$$

CycleGAN*의 *Loss

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

$$\mathcal{L}_{adv}(G,D) = \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)] + \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))] + \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))]$$

2(대해서)

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

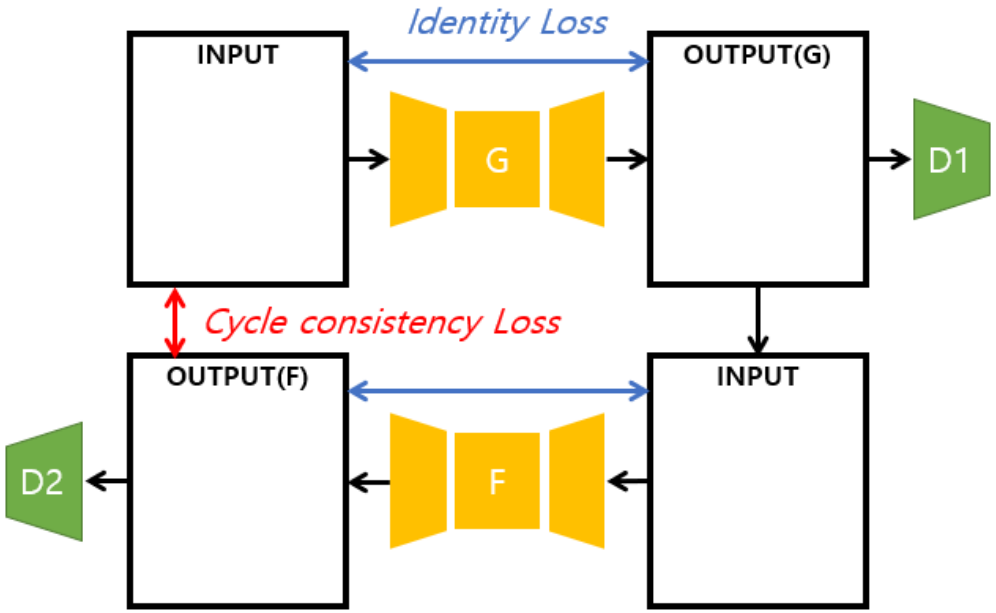
+

CartoonGAN 성능 검증

CycleGAN with initialization phase

CartoonGAN에서 initialization phase가 필요한 이유 → 주어진 이미지의 형태를 복원하기 위함

CycleGAN에서는 cycle-consistency loss와 identity loss를 통해서 initialization 없이도 주어진 이미지의 형태를 복원할 수 있음



CONTENTS

CartoonGAN 성능 향상

CartoonGAN 더 개선할 수 없을까?

1. BatchNorm → InstanceNorm

2. ReLU → LeakyReLU

3. VGG → ResNet

4. GAN → LSGAN

CartoonGAN 성능 향상

CartoonGAN 더 개선할 수 없을까?

1. BatchNorm → InstanceNorm

2. ReLU → LeakyReLU

3. VGG → ResNet

4. GAN → LSGAN

CartoonGAN 성능 향상

BatchNorm : problem and solution

- Unintended dependency of a content image contrast
- Solution : instance normalization

(a) Content image. (b) Stylized image.

(c) Low contrast content image. (d) Stylized low contrast image.

CartoonGAN 성능 향상

What is InstanceNorm?

- Remove instance-specific contrast information from the content image, which simplifies generation
- Key difference : applies the normalization to a single instance not whole

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \mu_i = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_i)^2. \tag{2}$$

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2. \tag{3}$$

K, j : spatial dimensions
i : feature channel (color channel if the input is an RGB image)
t : the index of the image in the batch

CartoonGAN 성능 향상

InstanceNorm : result

CartoonGAN 성능 향상

InstanceNorm : result

BatchNorm

InstanceNorm

CartoonGAN 성능 향상

CartoonGAN 더 개선할 수 없을까?

1. BatchNorm → InstanceNorm

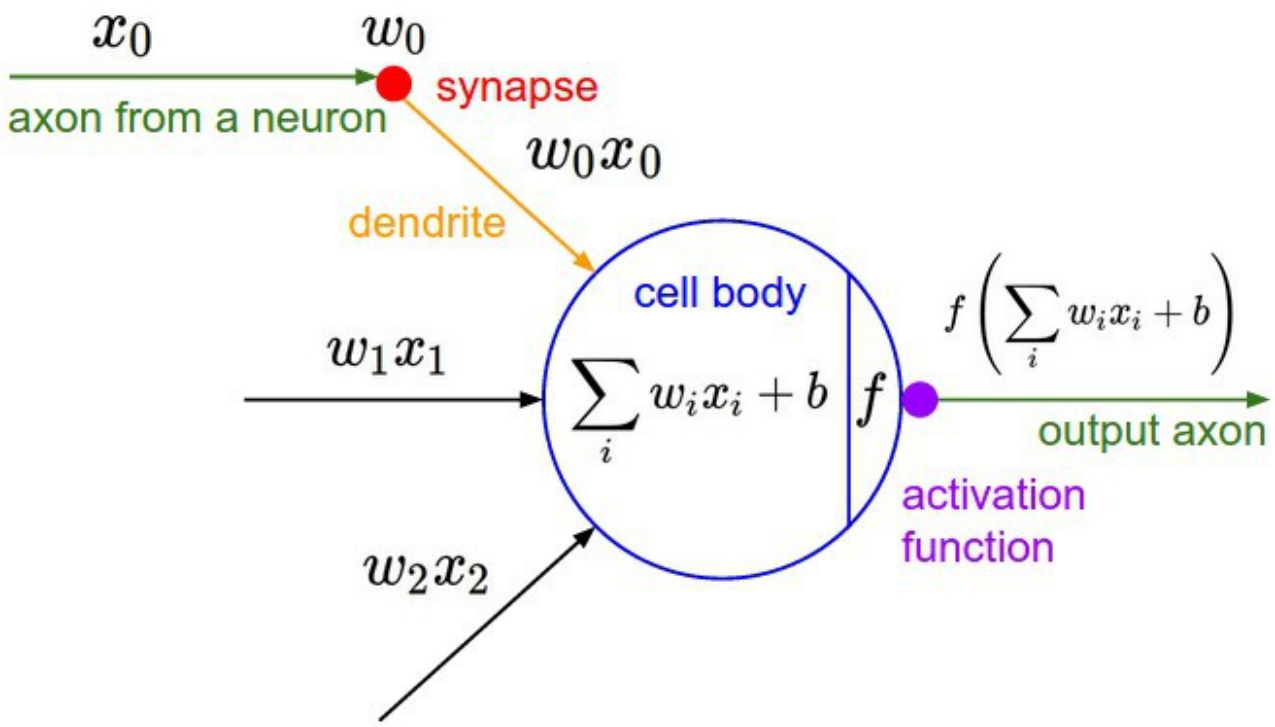
2. ReLU → LeakyReLU

3. VGG → ResNet

4. GAN → LSGAN

CartoonGAN 성능 향상

- 다양한 종류의 활성화 함수
  - Step function
  - Sigmoid function
  - ReLU function
  - Leaky ReLU function etc.



CartoonGAN 성능 향상

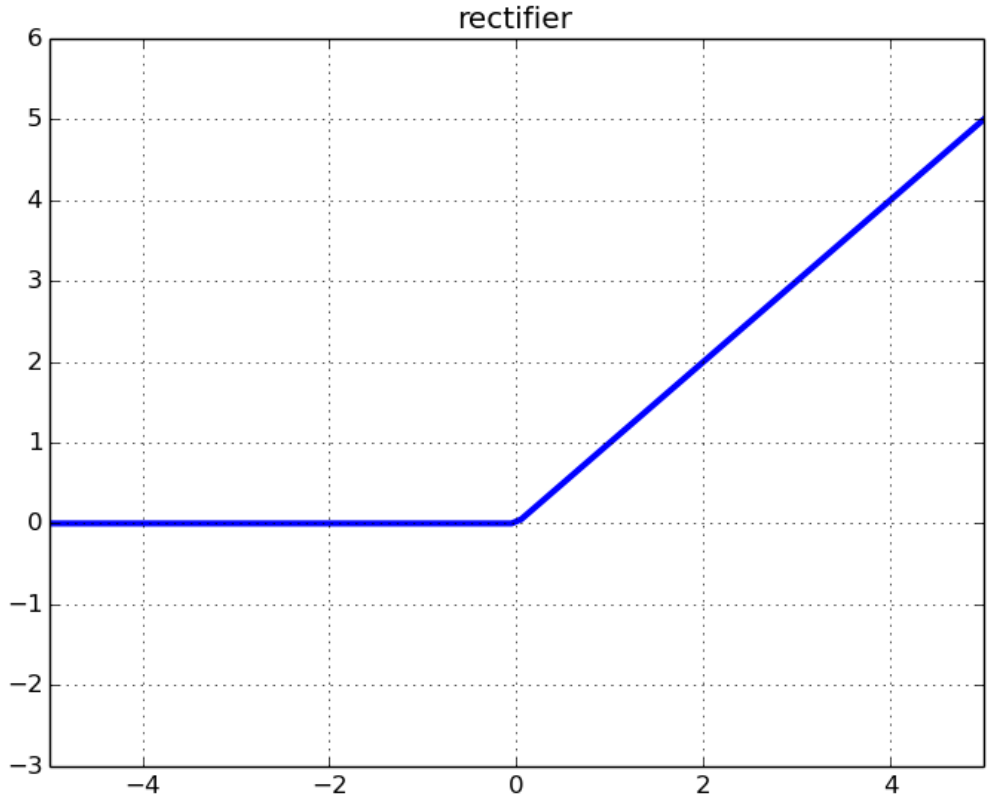
The ‘ReLU function’ : definition, problem and solution

- Definition : $$RELU(x) = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x >= 0 \end{cases} \qquad RELU(x) = \max(0, x)$$

- Problem : when the input to it is negative, they “die” (output zero)

  → block back propagation

- Solution : Leaky ReLU function

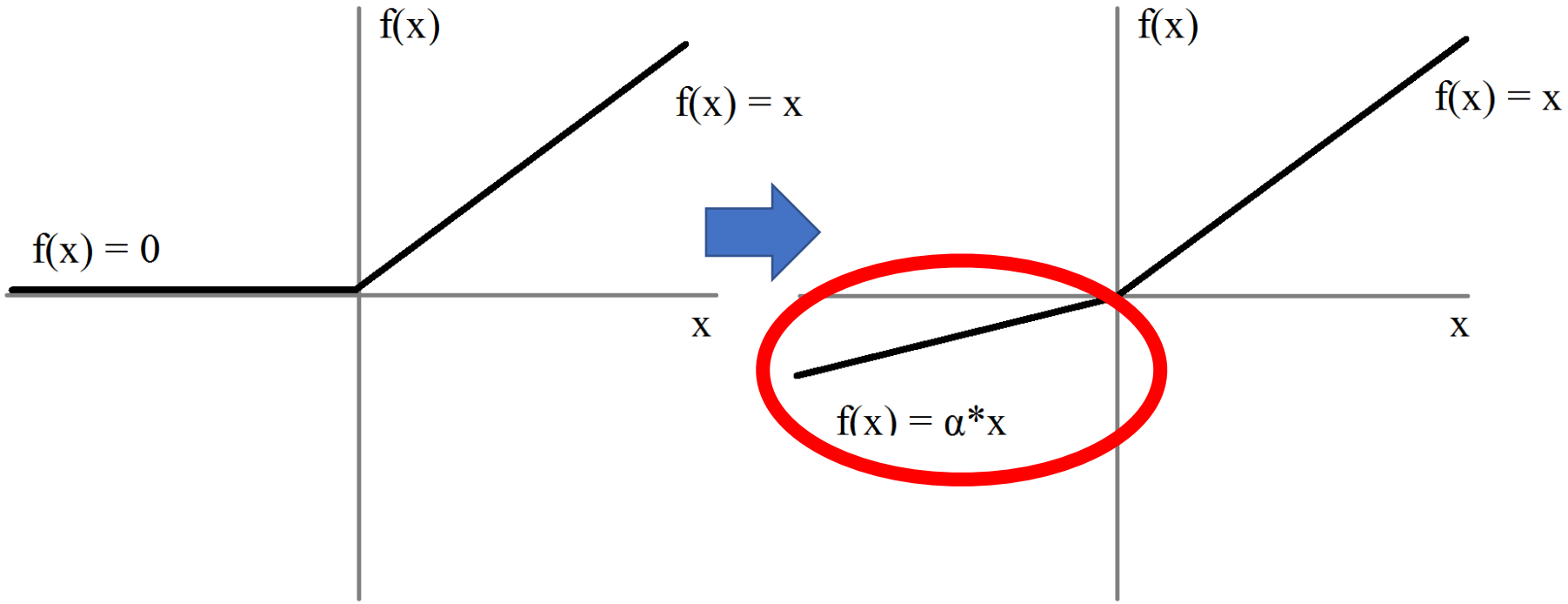


CartoonGAN 성능 향상

The LeakyRELU function

- Definition : $Leaky\,RELU(x) = \begin{cases} 0 \ if \ x < 0 \\ x \ if \ x >= 0 \end{cases}$ $\varepsilon = 0.01$

$$RELU(x) = \max(\varepsilon x, x)$$



CartoonGAN 성능 향상

CartoonGAN 더 개선할 수 없을까?

1. BatchNorm → InstanceNorm

2. ReLU → LeakyReLU

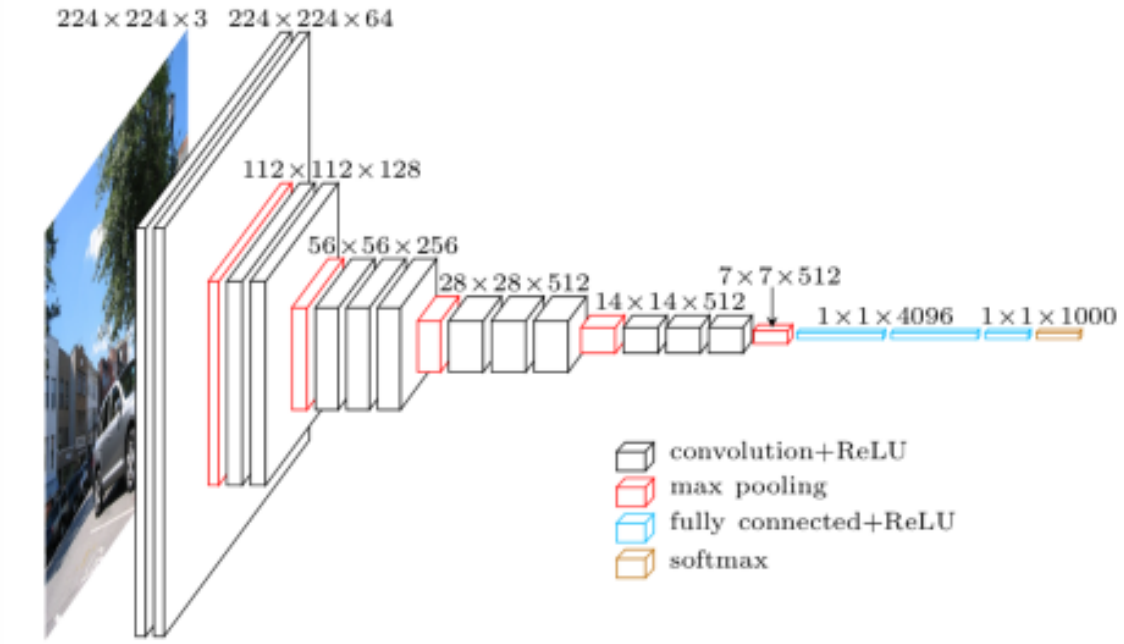
3. VGG → ResNet

4. GAN → LSGAN

CartoonGAN 성능 향상

What is the VGG?

- Definition
  - ImageNet database의 1백만 개가 넘는 이미지에 대해 훈련된 신경망
  - 이미지를 키보드, 마우스, 연필, 각종 동물 등 1,000가지 사물 범주로 분류

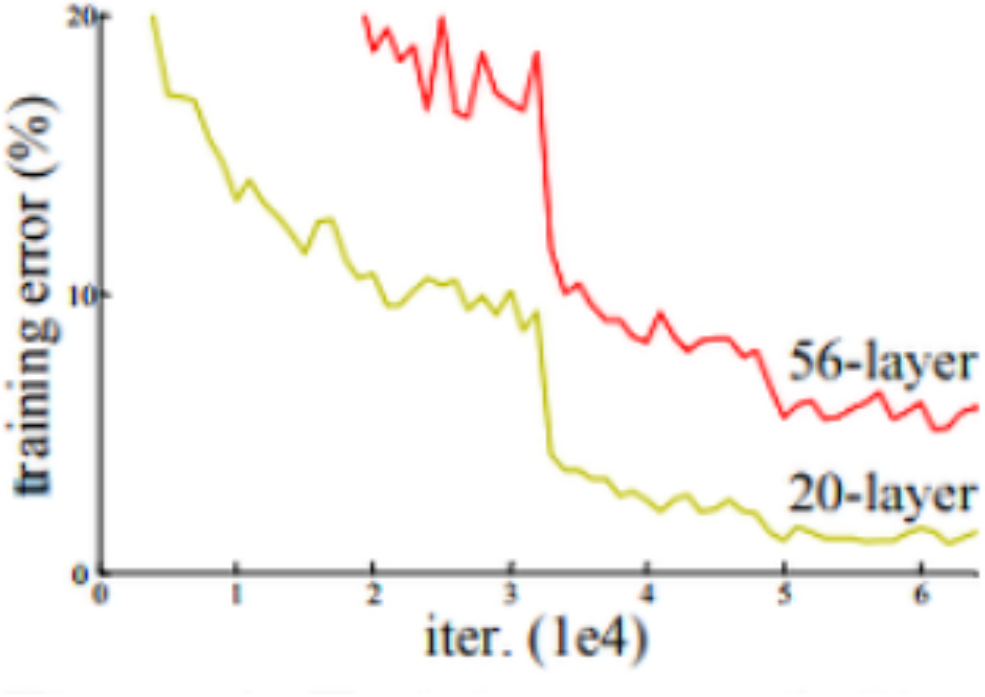


CartoonGAN 성능 향상

VGG : problem and solution

- Exploding / Degradation problem : 망이 깊어질 시 vanishing gradient 문제로 인해 학습 불가 → 오히려 정확도 저하

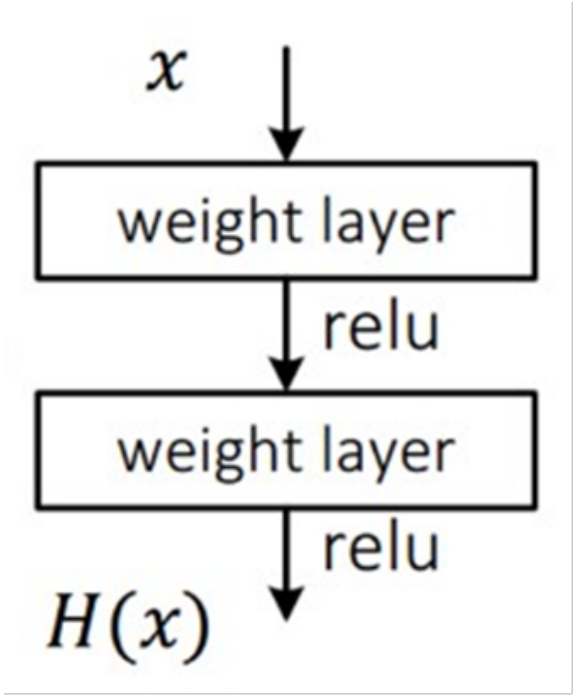
- Solution : ResNet

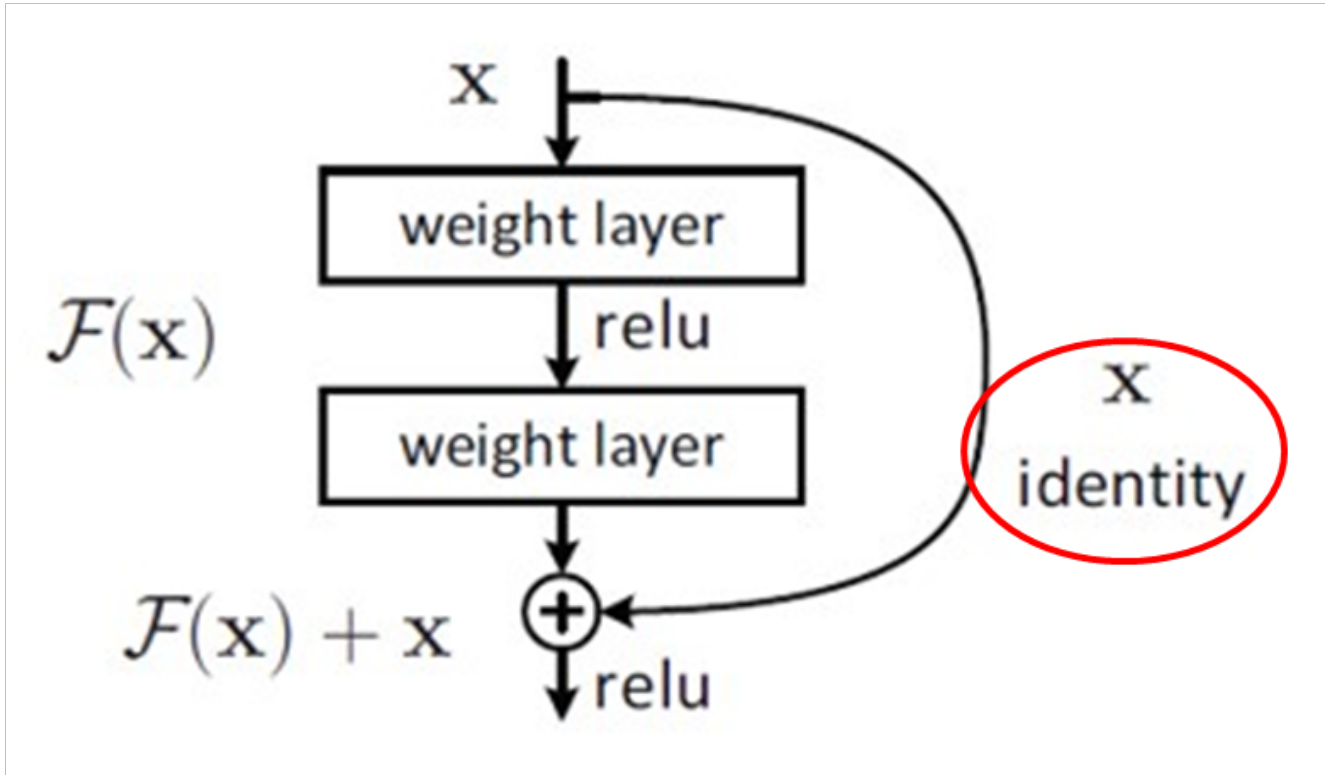


CartoonGAN 성능 향상

What is the ResNet?

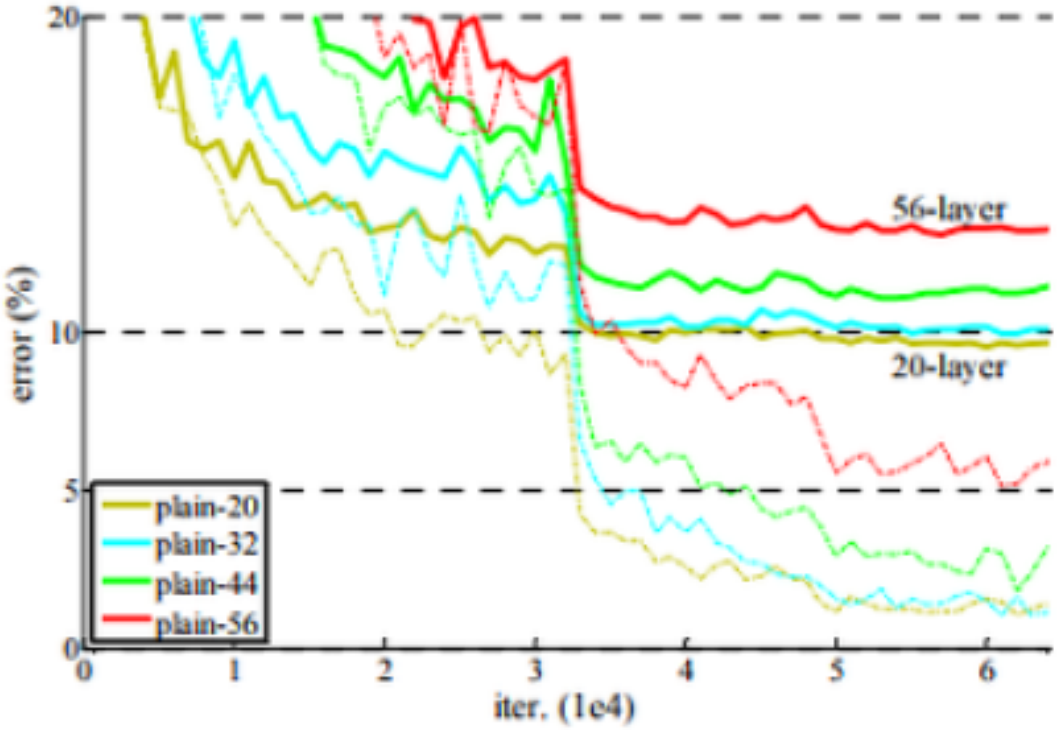
- layer의 입력을 출력에 바로 연결시키는 “skip connection”을 사용 → 사전 학습된 VGG보다 특징추출에 있어서 더 나은 성능을 보일 것이라 예상



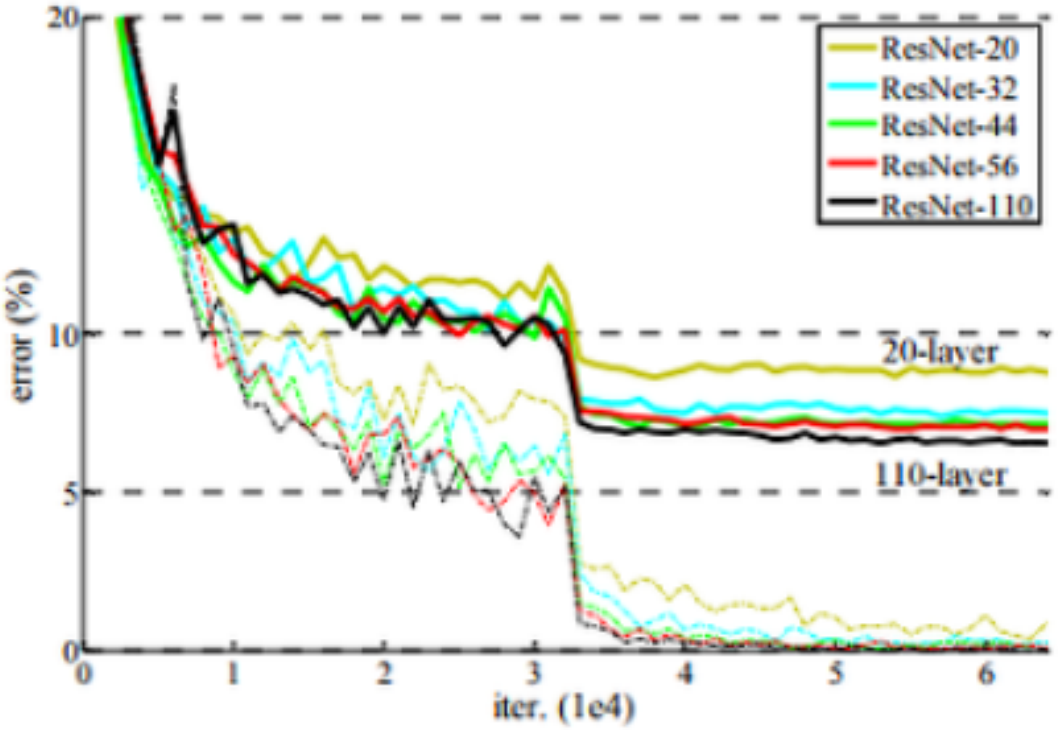


CartoonGAN 성능 향상

ResNet Experiment : Result



Plain network

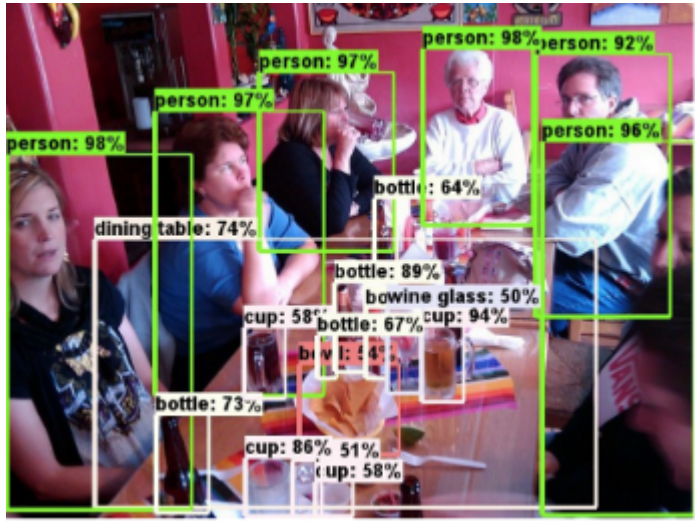


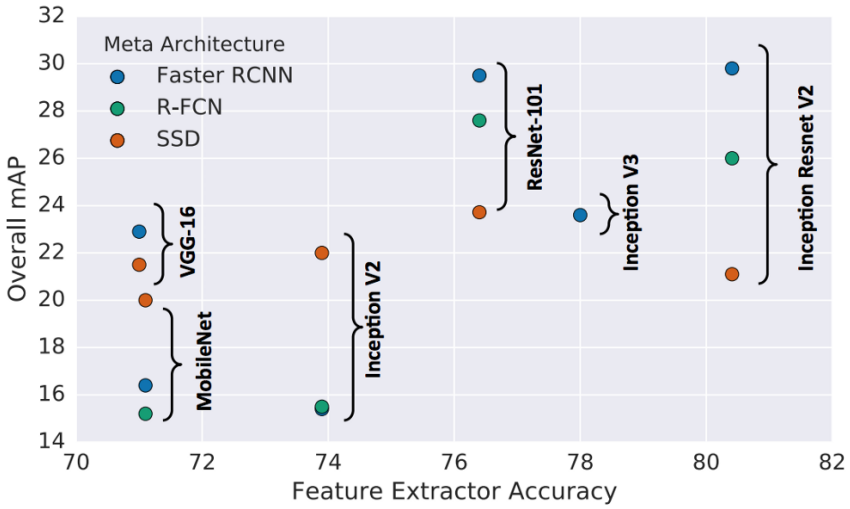
ResNet

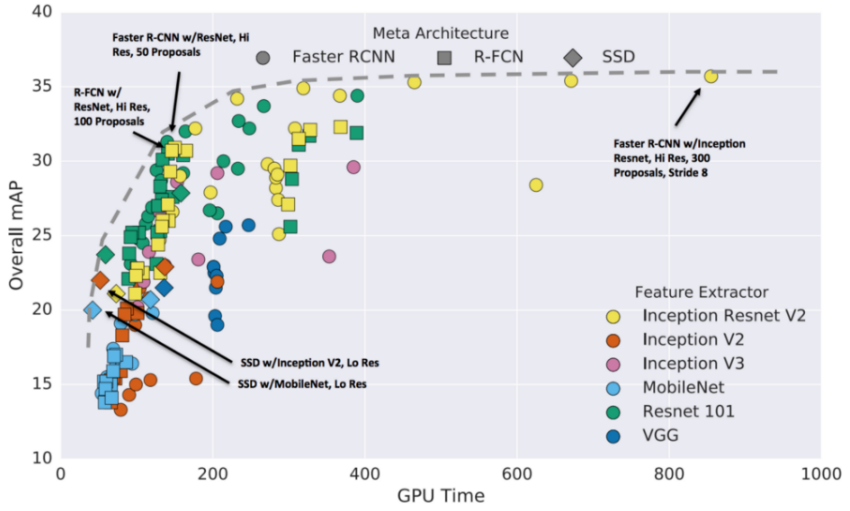
CartoonGAN 성능 향상

ResNet > VGG

- ResNet-50 is faster than VGG-16
- More accurate than VGG-19
- Good effect on object detection







CartoonGAN 성능 향상

CartoonGAN 더 개선할 수 없을까?

1. BatchNorm → InstanceNorm

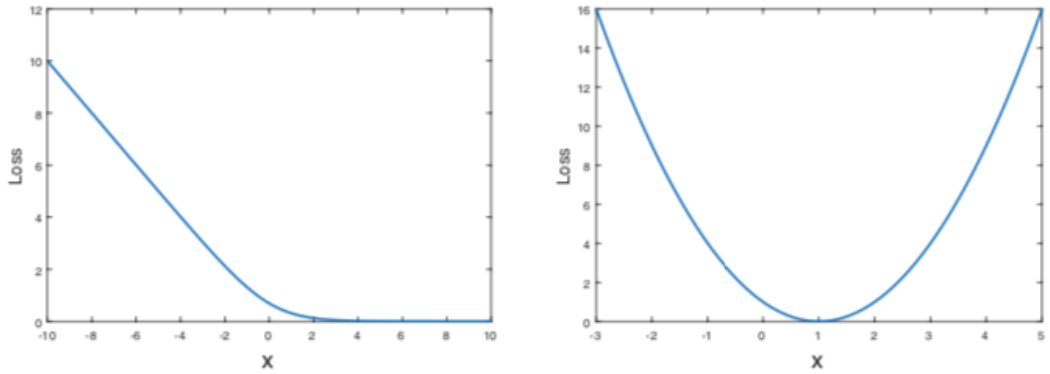
2. ReLU → LeakyReLU

3. VGG → ResNet

4. GAN → LSGAN

CartoonGAN 성능 향상

- GAN : adopt the sigmoid cross entropy loss function for the discriminator
  - Problem : vanishing gradients



(a) (b)

Figure 2: (a): The sigmoid cross entropy loss function. (b): The least squares loss function.

CartoonGAN 성능 향상

What is the “LSGAN”

- Solution : Least Squares Generative Adversarial Networks (LSGANs)
  - Generates more gradients to update the generator

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],$$

CartoonGAN 성능 향상 (결과 1)

FID score : original CartoonGAN vs modified CartoonGAN

- FID with ani : 실제 애니메이션 도메인의 m.g.d.과 생성된 이미지의 m.g.d.의 거리(작을수록 우수)
- FID with photo : 실제 사진 도메인의 m.g.d.과 생성된 이미지의 m.g.d.의 거리(클수록 우수)

	CartoonGAN	CartoonGAN-modified
FID with ani	100.2994251	95.49735281
FID with photo	80.95636448	81.90999061

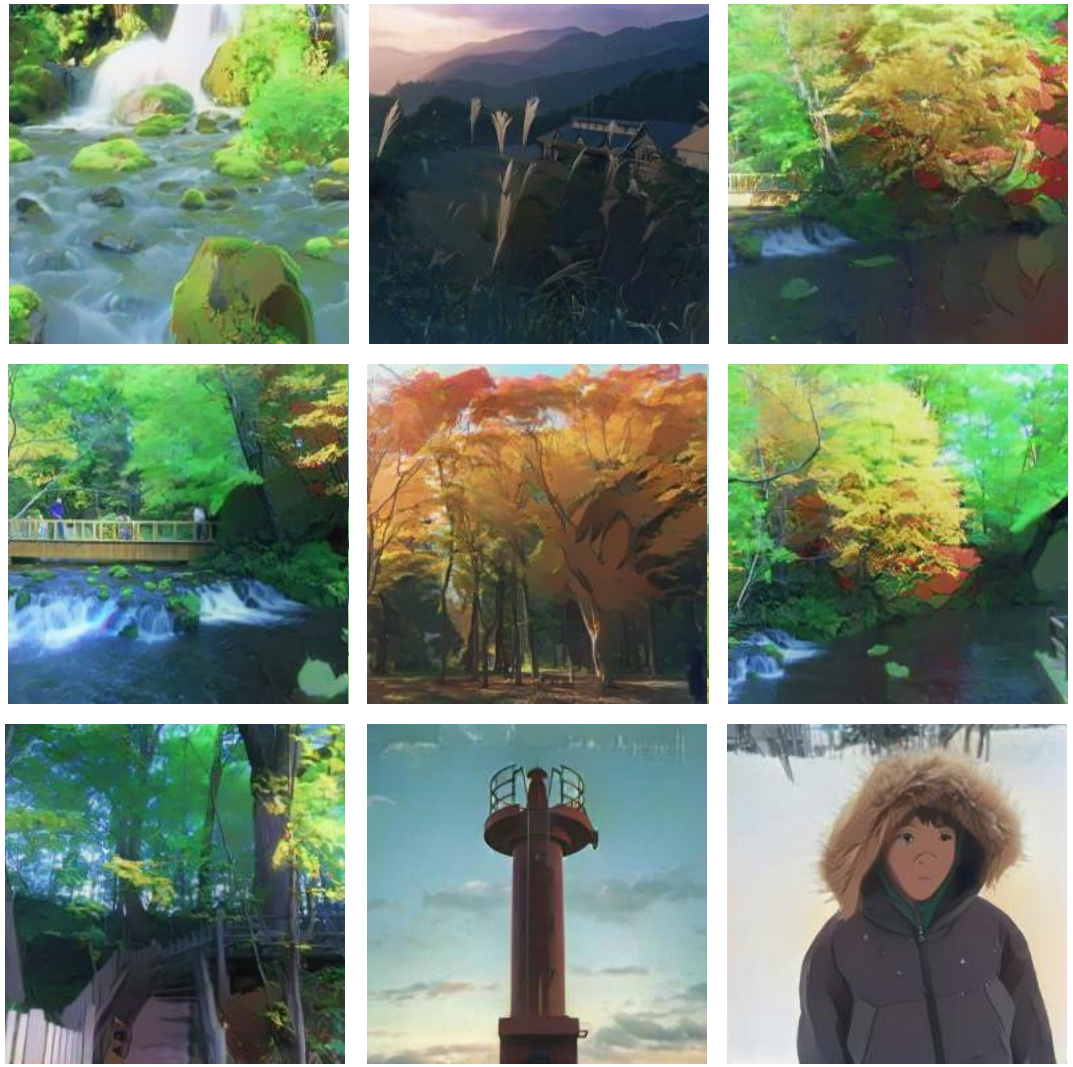
CartoonGAN 성능 향상 (결과 2)

Modified-CartoonGAN

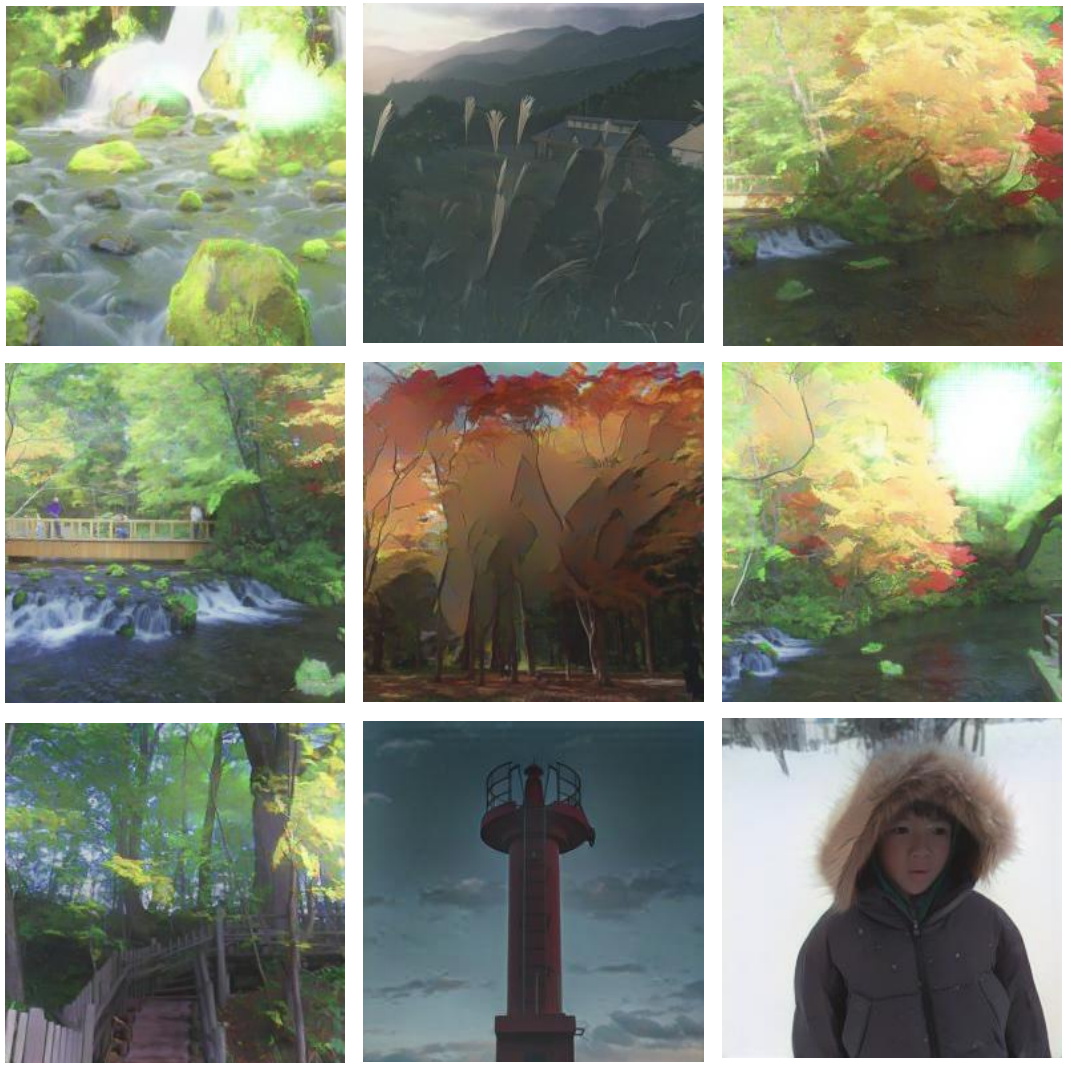
Original - CartoonGAN

CartoonGAN 성능 향상 (결과 3)

Modified-CartoonGAN



Original - CartoonGAN



CartoonGAN 성능 향상 (결과 2)

Modified-CartoonGAN

Original - CartoonGAN

CartoonGAN 성능 향상 (결과 3)

Modified-CartoonGAN

Original - CartoonGAN

CartoonGAN 성능 향상 (결과 3)

Modified-CartoonGAN

Original - CartoonGAN

Q & A

Thank you