딥러닝 기반
이미지 생성 및 변환.

주 재 걸 KAIST 김재철AI대학원

Fast campus

the Red.

주 재 걸 KAIST 김재철AI대학원 부교수

약력

- 학사, 2001 서울대학교 전기공학부
- 석사, 2009 Georgia Tech, Electrical and Computer Eng.
- 박사, 2013 Georgia Tech, Computational Science and Eng.
- 조교수, 2015-2019, 고려대학교 컴퓨터학과
- 부교수, 2019-2020, 고려대학교 인공지능학과

데이터 시각화 연구실

- 박사과정 12명, 석박사통합과정 20명, 석사과정 21명

Publication Highlights

- **2021** : 3 CVPR (1 Oral), 3 ICCV (2 Oral) 3 ACL (1 Long, 1 Short, 1 Findings), 1 NAACL, 2 EMNLP (1 Short, 1 Findings), 1 NeurIPS (Oral), 1 AAAI, 1 CSCW, 1 ICASSP, 3 BMVC (1 Oral), 1 ACML
- **2020** : 3 CVPR, 1 ICLR, 1 ICML, 1 CIKM, 1 BMVC (Oral), 1 IEEE VIS
- **2019** : 2 CVPR (1 Oral), 1 AAAI, 1 BMVC (Spotlight), 1 EMNLP, 1 CIKM, 1 CHI, 1 IEEE VIS Short, 1 EuroVIS
- **2018** : 1 CVPR (Oral), 1 ECCV, 1 EMNLP, 1 WWW, 1 IJCAI, 1 CHI, 1 IEEE VIS, 1 EuroVIS
- **2017** : 1 AAAI, 2 IJCAI, 1 ICDM, 2 IEEE VIS

KAIST AI
Graduate School of AI

DAVIAN
Data and Visual Analytics Lab

Fast campus

Selected Publications by Areas - 최근 5년간 피인용수 4,800회 이상

영상 인식 및 합성

- VITON-HD: High-Resolution Virtual Try-On via Misalignment-Aware Normalization, **CVPR'21**
- ① Reference-based Sketch Image Colorization using Augmented-Self Reference, **CVPR'20**
- Image-to-Image Translation via Group-wise Deep Whitening and Coloring, **CVPR'19**, Oral paper (Top 5.5%)
- StarGAN: Unified GANs for Multi-Domain Image Translation, **CVPR'18**, Oral paper (Top 2%)

자연어 이해 및 생성

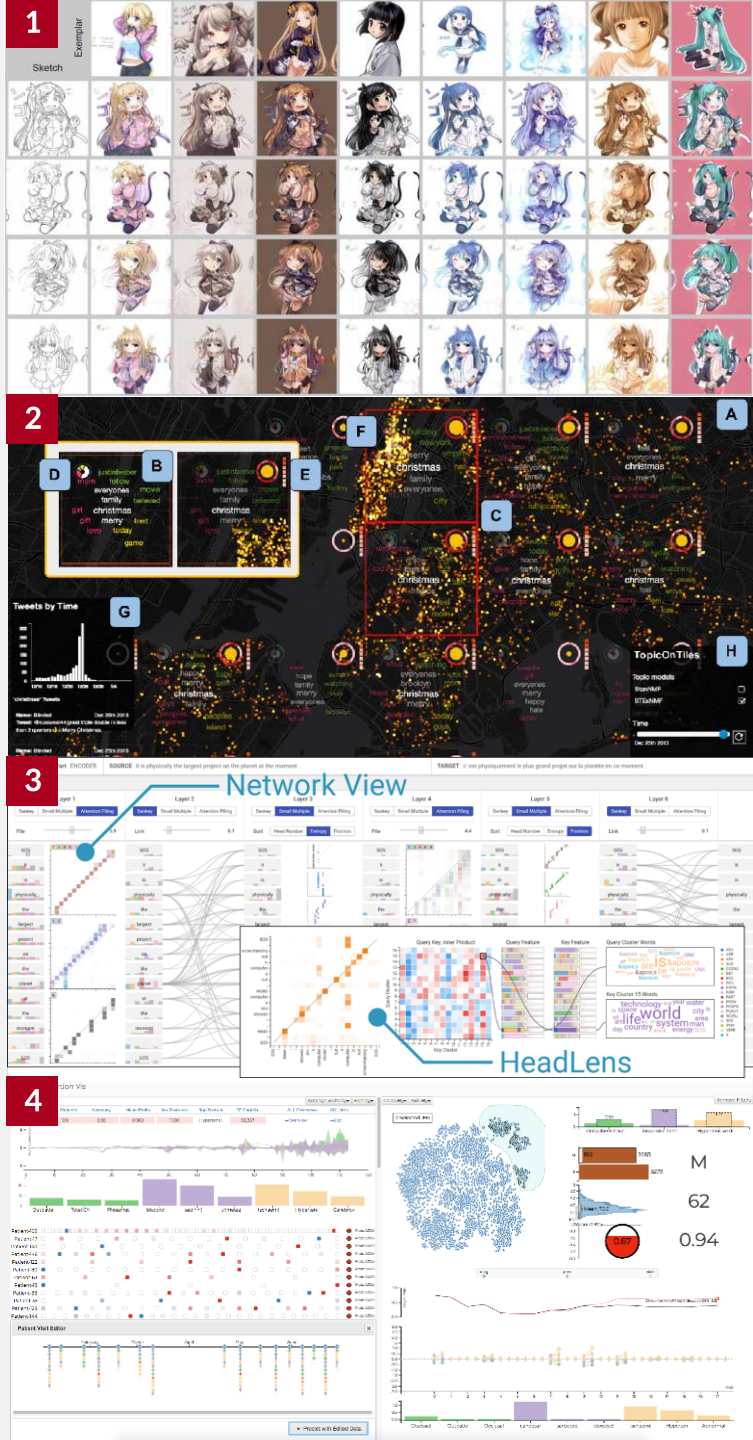
- Restoring and Mining the Records of the Joseon Dynasty via Neural Language Modeling and Machine Translation, **NAACL'21**
- NeurQuRI: Neural Question Requirement Inspector for Answerability Prediction in Machine Reading Comprehension, **ICLR'20**
- Paraphrase Diversification using Counterfactual Debiasing, **AAAI'19**

텍스트 및 소셜미디어 빅데이터 분석

- ST-GRAT: Spatio-Temporal Graph Attention Network for Traffic Forecasting, , **CIKM'20**
- Recommender System via Sequential and Global Preference via Attention Mechanism and Topic Modeling, **CIKM'19**
- ② TopicOnTiles: Tile-Based Spatio-Temporal Event Analytics via Exclusive Topic Modeling on Social Media, **CHI'18**

인공지능 모델 해석 및 상호작용 시스템

- HyperTendril: Visual Analytics for User-Driven Hyperparameter Optimization of Deep Neural Networks, **IEEE VIS'20**
- ③ SANVis: Visual Analytics for Understanding Self-Attention Networks, **IEEE VIS'19 Short**
- AILA: Attentive Interactive Labeling Assistant for Document Classification through Deep Neural Networks, **CHI'19**
- ④ RetainVis: Visual Analytics with Interpretable and Interactive RNNs on Electronic Medical Records, **IEEE VIS'18**



본 강의의 목차

1. 컴퓨터 비전 분야 및 생성 모델의 소개
2. 적대적 생성 신경망 (generative adversarial networks)
3. 다양한 생성 모델
4. 다양한 이미지 변환 모델
5. 이미지 생성 및 변환 모델의 최신 연구 동향 및 실제 활용 사례

01
생성 모델의 기초.

Ch 1. 생성 모델 소개

Clip 1. 인식 vs. 생성 및 변환 모델

컴퓨터 비전 분야의 기본 태스크

두 가지 태스크로 나뉨

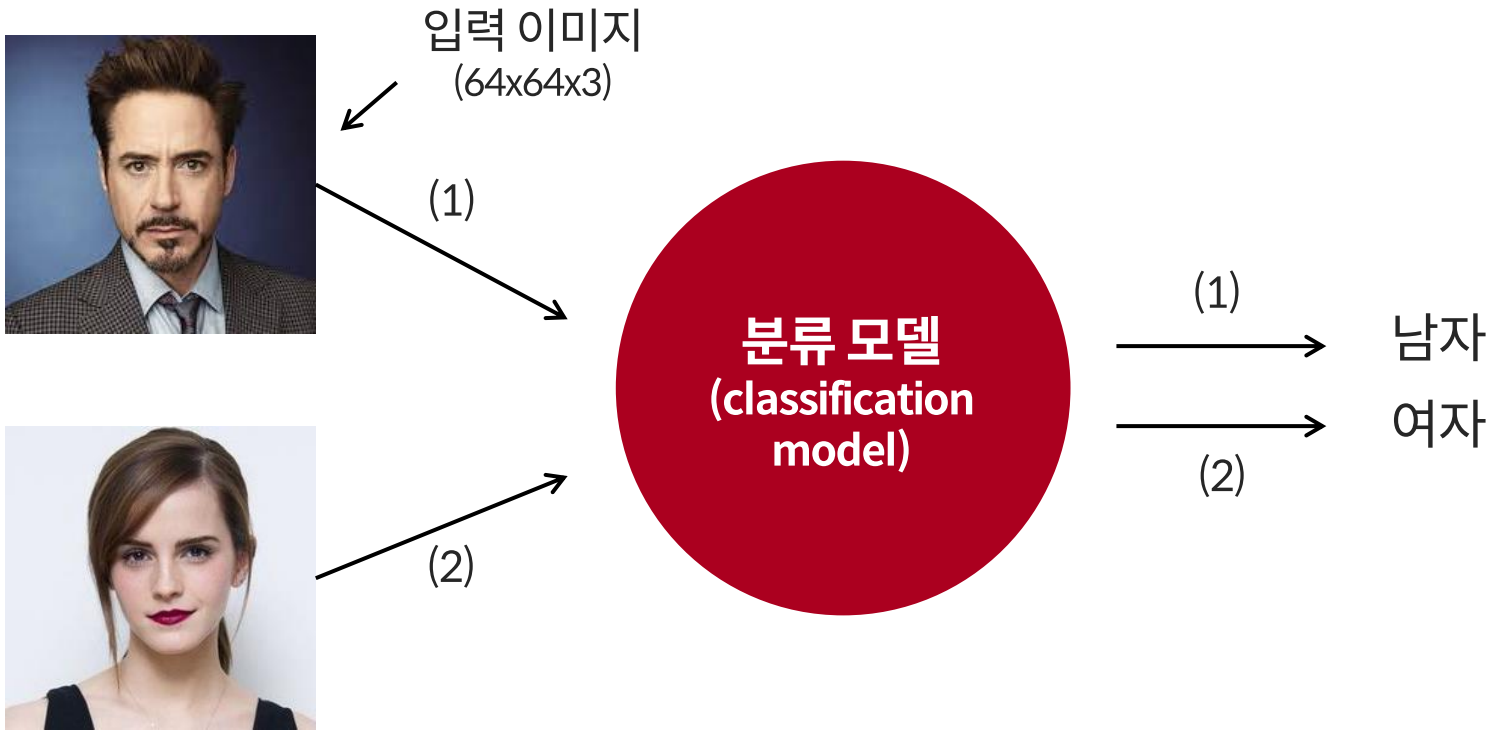
- **인식 태스크** 예측할 변수의 형태 (범주형 변수 vs. 연속형 변수)에 따라, 분류 (classification) 모델과 회귀 (regression) 모델로 나뉨

- **생성 태스크** 순수 생성 (별도의 조건을 입력으로 주지 않고 이미지를 랜덤하게 생성) 및 조건부 생성 (특정 조건을 입력으로 주고, 이를 만족하는 이미지를 랜덤하게 생성) 태스크로 나뉨. 조건부 생성의 경우, 이미지를 특정 조건으로서의 입력으로 주면, 해당 이미지를 원하는 형태로 변환하는 태스크가 됨

과거에는 인식 태스크가 많이 연구되어 왔으나,
현재는 생성 및 변환 태스크가 활발히 연구되고 있고, 후자가 본 강의의 주된 주제임

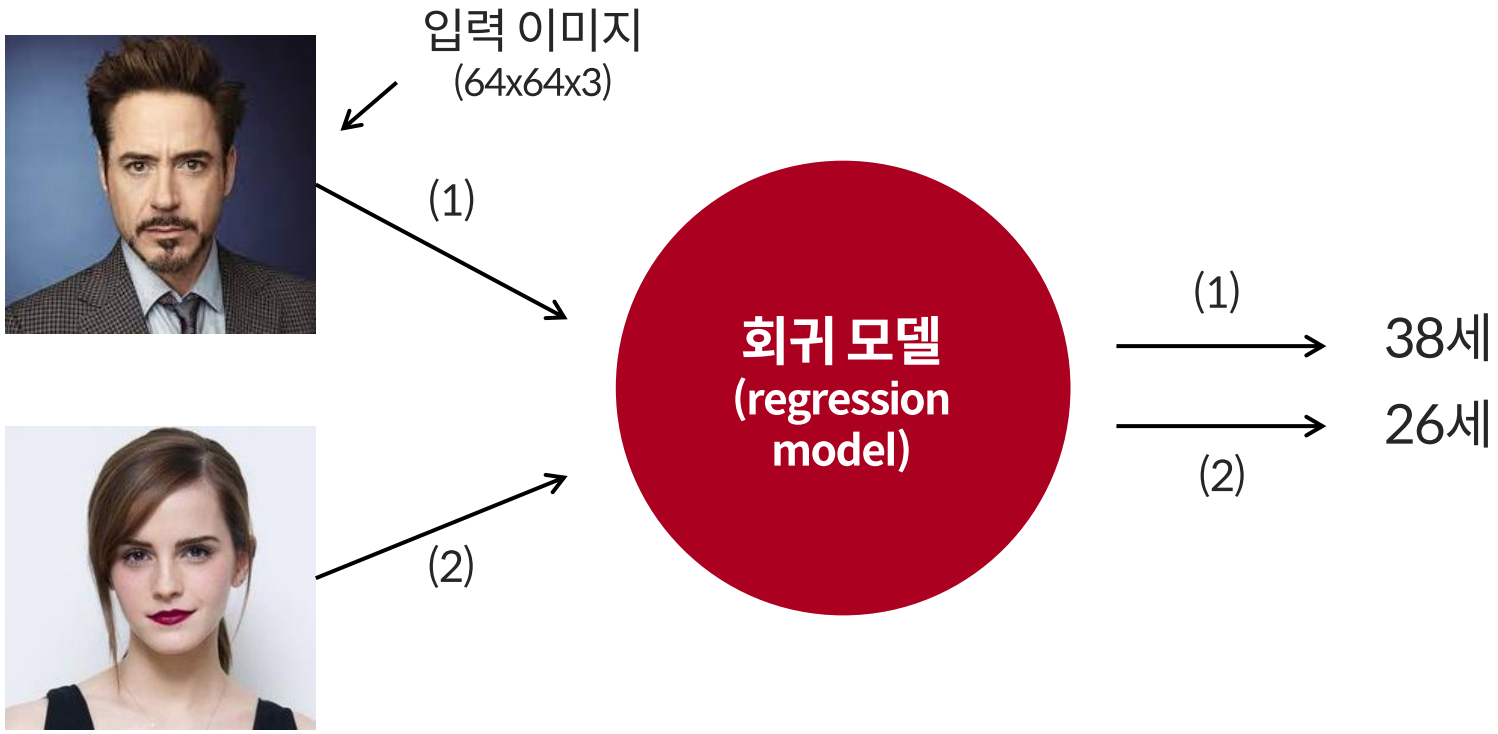
인식 태스크 (Recognition Tasks): 분류 모델

분류 모델은 주어진 입력 이미지를 그에 맞는 카테고리로 분류하는 법을 학습함



인식 태스크 (Recognition Tasks): 회귀 모델

회귀 모델은 주어진 입력 이미지에 대한 연속형 변수값(예: 나이)을 추정하는 법을 학습함



생성 태스크 (Generation Tasks)

생성 모델은 주어진 학습데이터의 확률 분포를 학습한 후,
해당 확률분포로부터 랜덤하게 샘플 데이터를 생성함

변환 태스크 (Translation Tasks): 조건부 생성 모델의 한 형태

변환 모델은 주어진 입력 이미지에 원하는 속성을 반영한 이미지를 합성함

출력이미지
(64x64x3)

변환 모델 (Translation model)

출력이미지
(64x64x3)

인식 태스크

인식 태스크 (recognition task)

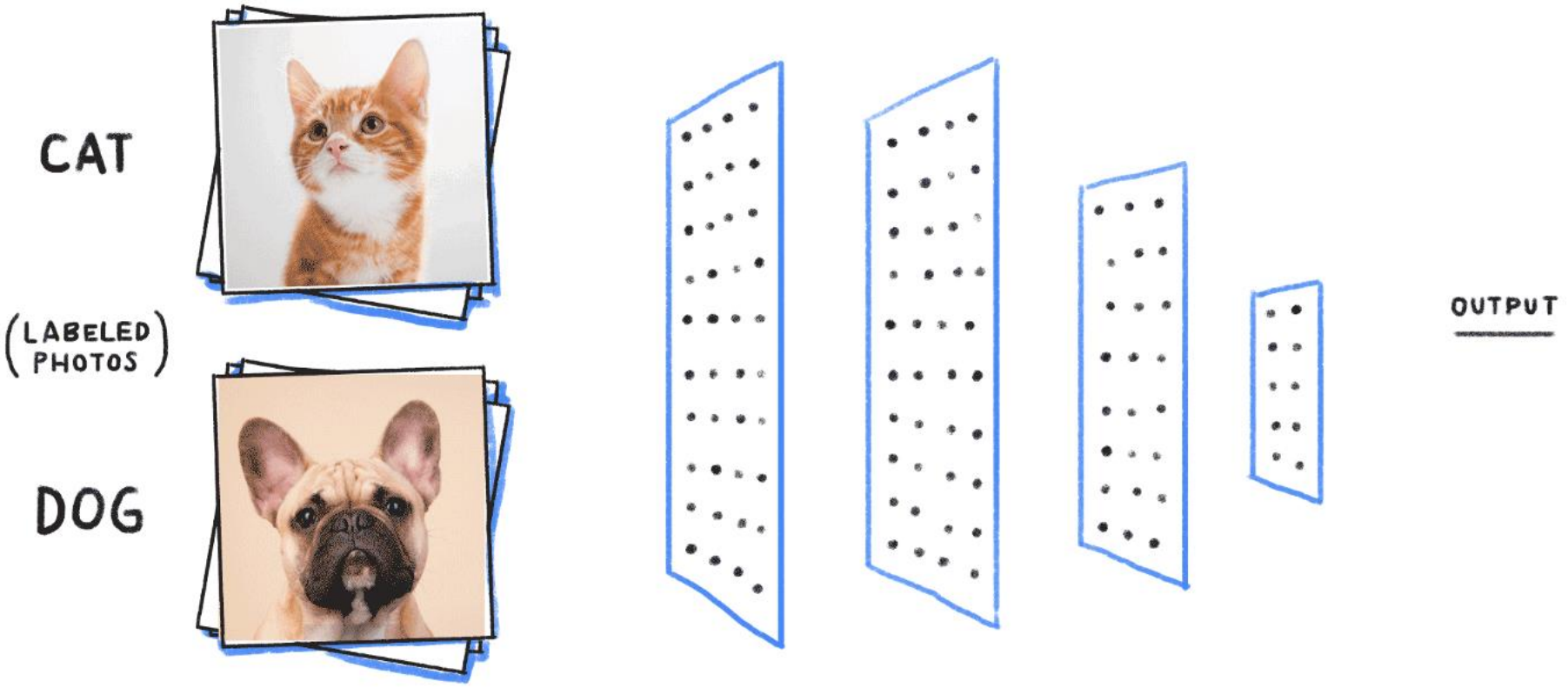
Image level

- Image classification
- Object detection and localization
- Semantic segmentation, instance segmentation, panoptic segmentation
- Face detection, recognition, identification
- Landmark detection, keypoint estimation
- Data augmentation techniques
- Knowledge distillation, lightweight low-power models

Video level

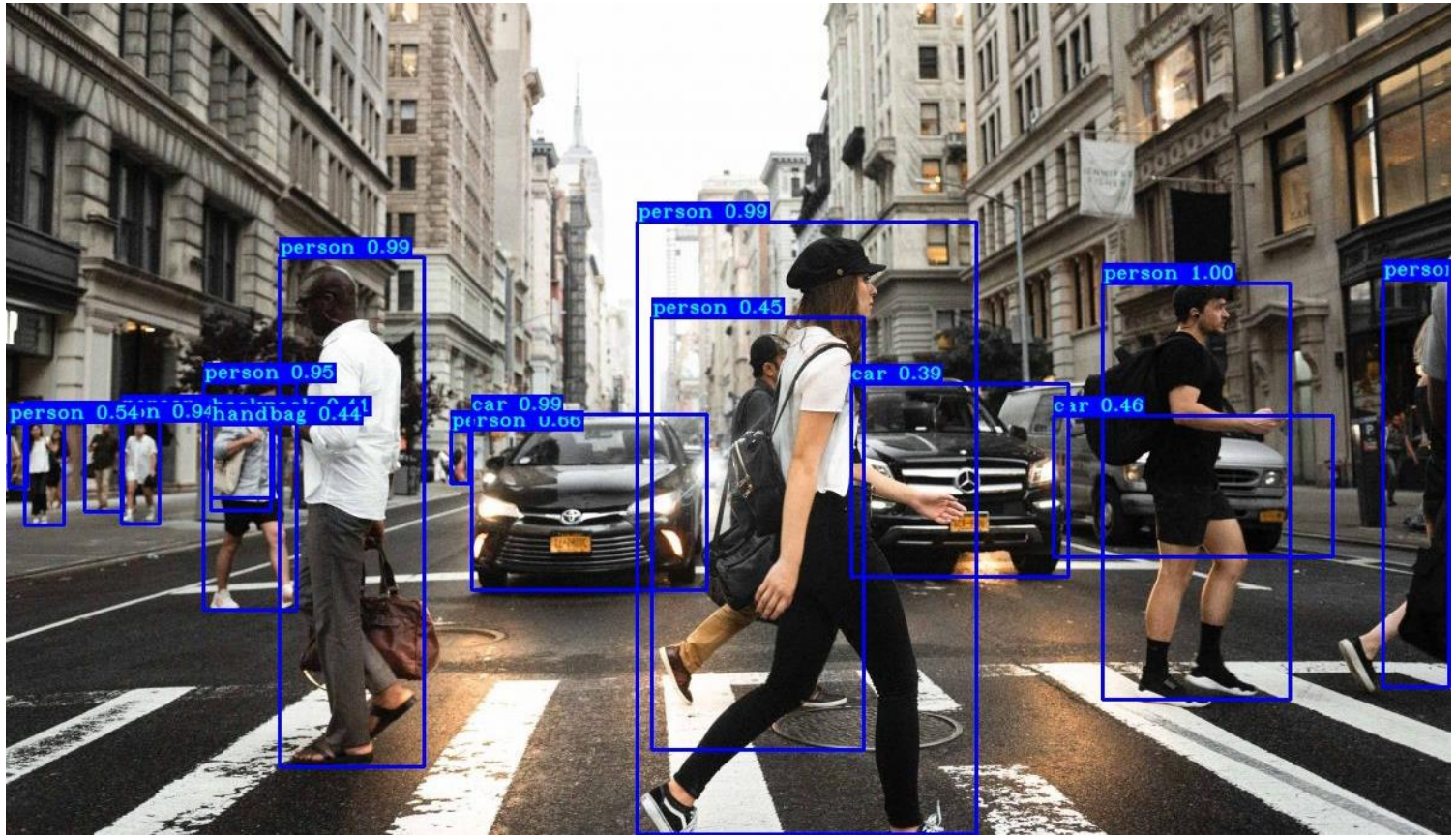
- Video object tracking
- Action recognition

Image Classification

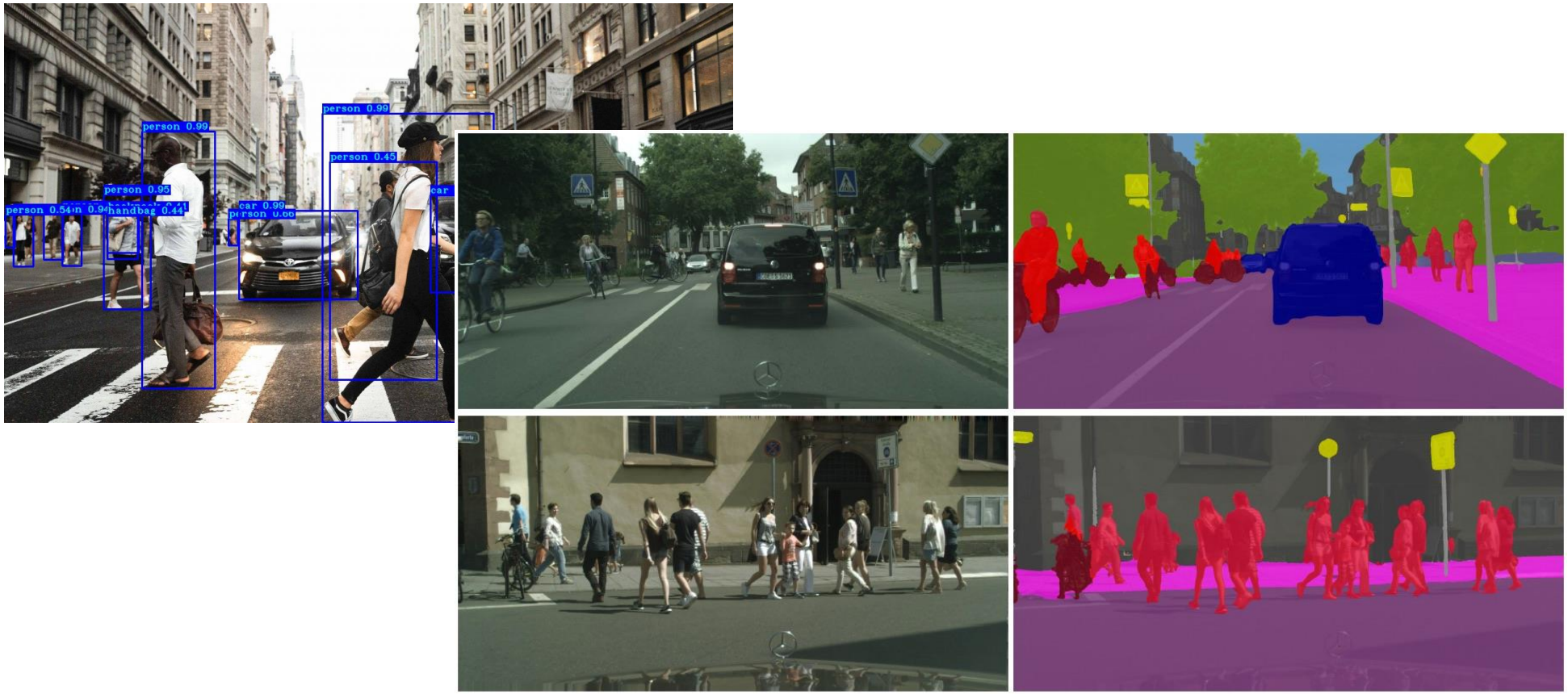


https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8

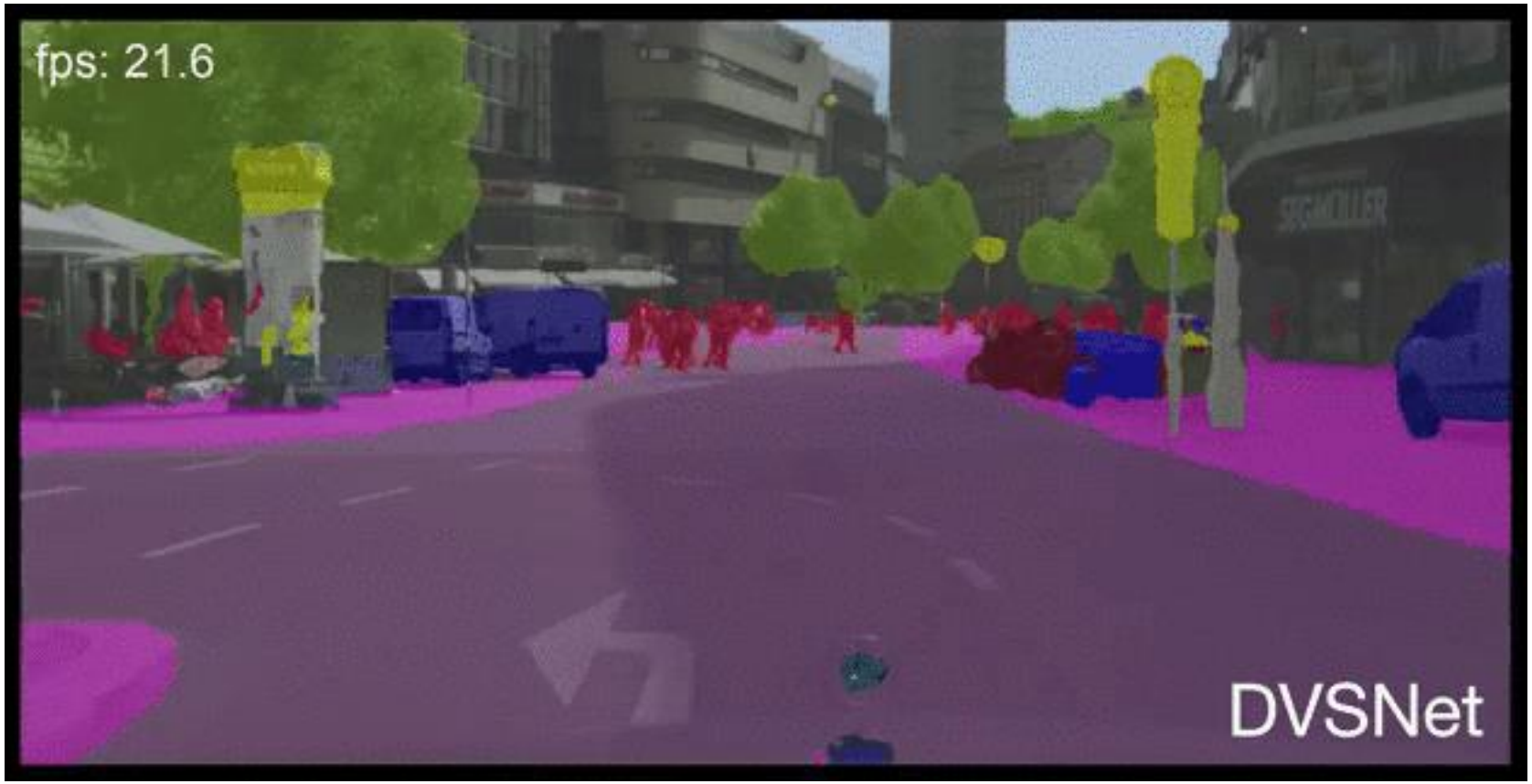
Object Detection/Localization



Semantic Segmentation

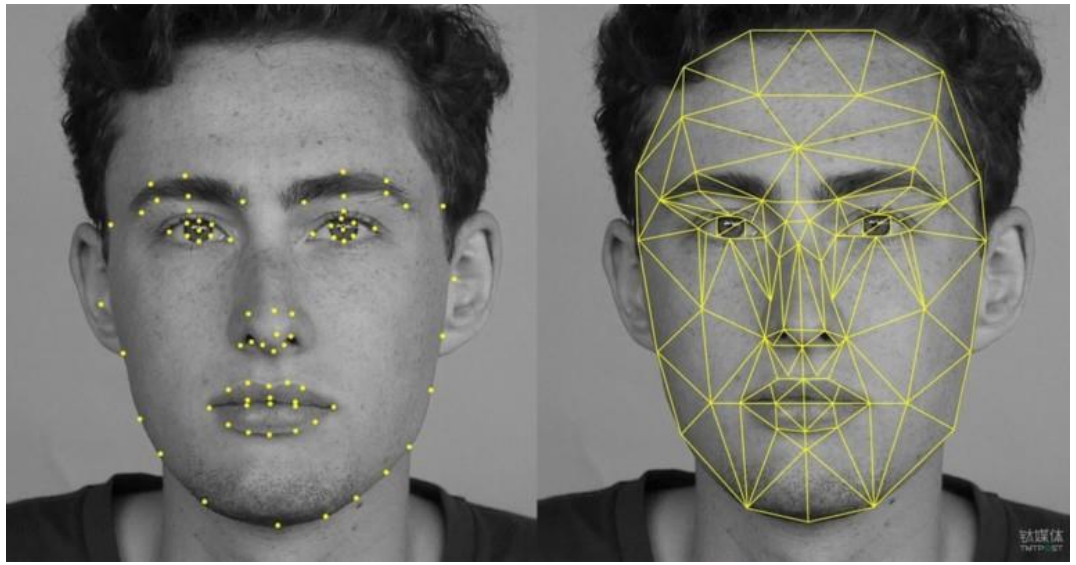


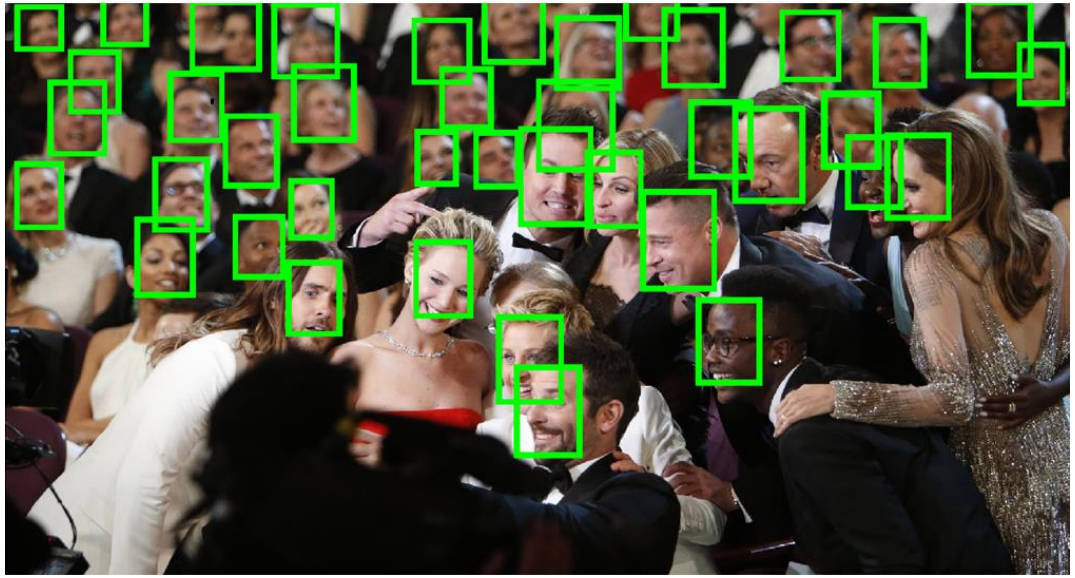
Semantic Segmentation for Video



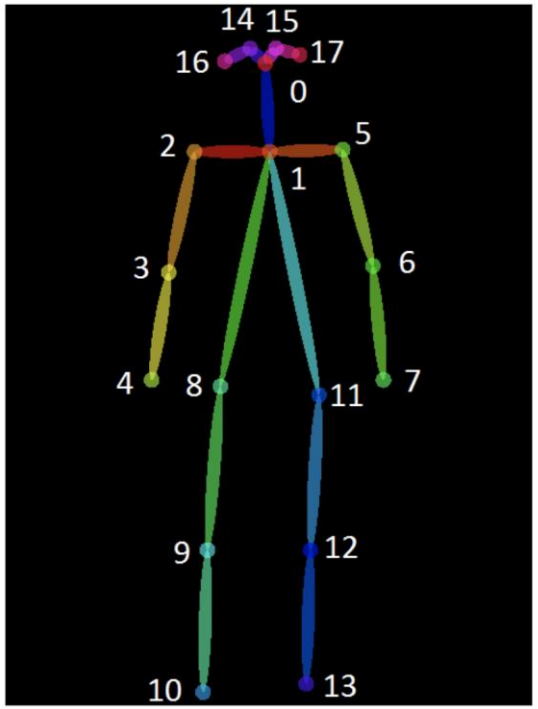
Dynamic Video Segmentation Network (CVPR 2018): https://arxiv.org/abs/1804.00931

Facial Landmark Detection / Face Detection
(Localization)





Pose Estimation



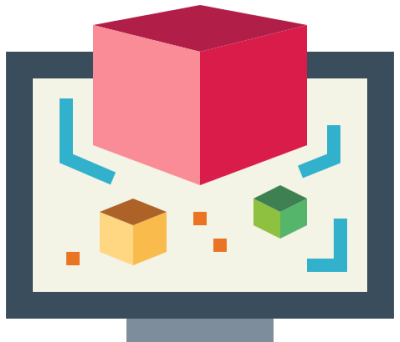
데이터의 생성 및 변환

가짜 얼굴 영상, 음성, 텍스트 등을 자유로이 생성

웹툰, 애니메이션, 영화, 미술, 음악 등의 컨텐츠 제작 분야에서 활용 가치가 큼

태스크 예시

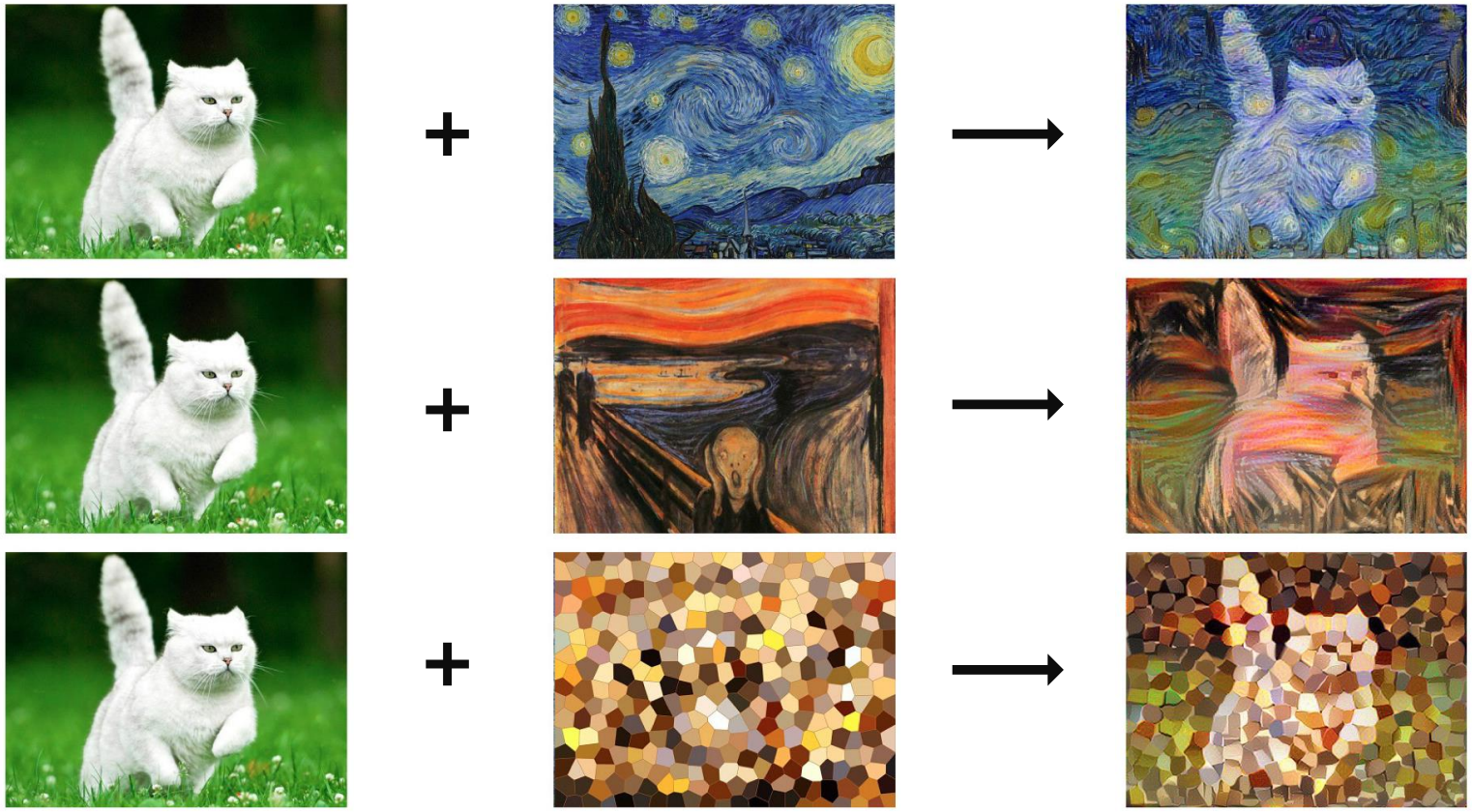
- Style transfer
- Image generation
- Image inpainting
- Translating into animation characters
- Image manipulation and editing
- 3D reconstruction
- DeepFake



Style Transfer

Style Transfer

Content image + Style image → Output image



고해상도 영상 합성 사례

Demo:
https://thispersondoesnotexist.com

영상 변환 사례 (I)

Monet ⇄ Photos

Monet → photos

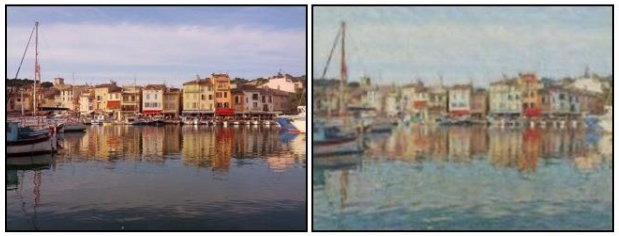


photo → Monet

Zebras ⇄ Horses

zebras → horses

horse → zebra

Summer ⇄ Winter

summer → winter

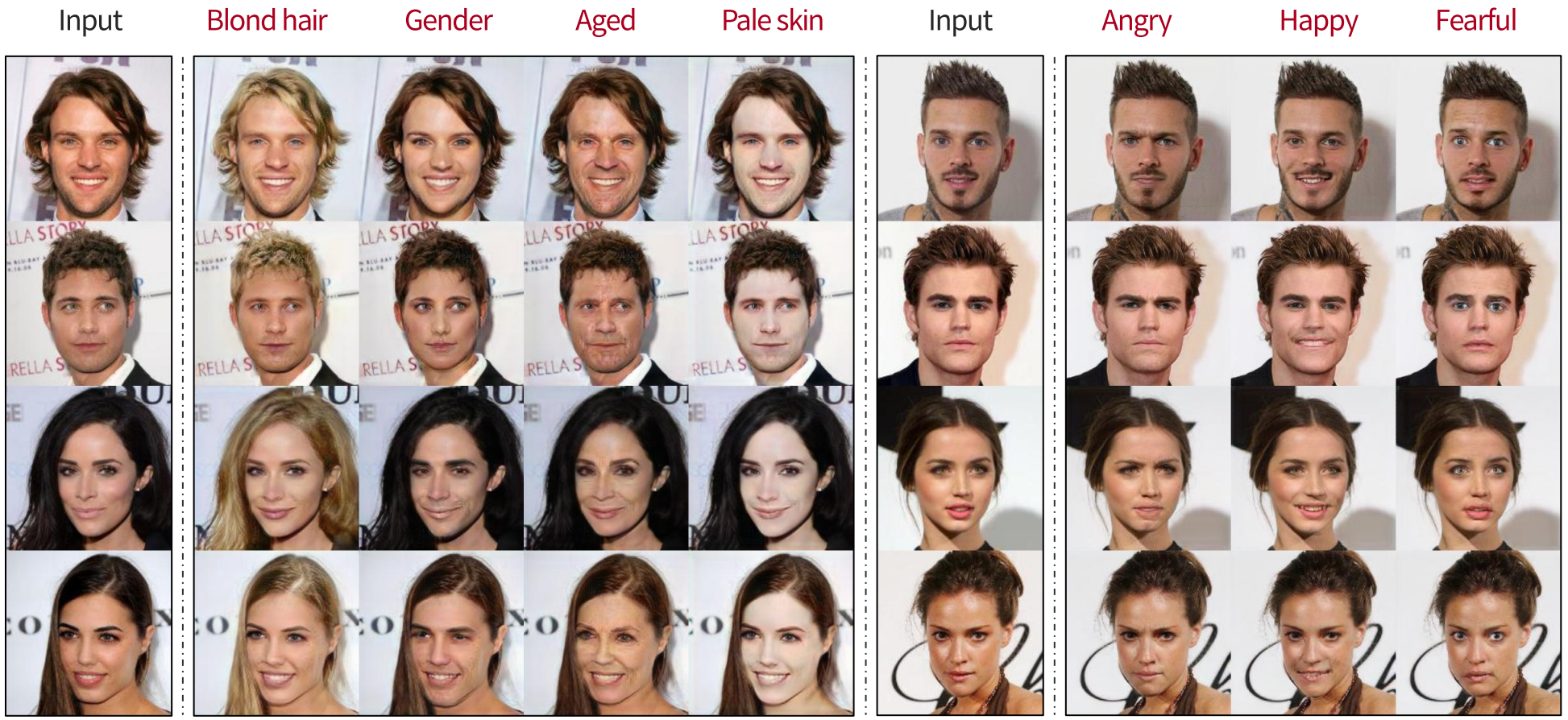
winter → summer

Photograph →

Monet | Van Gogh | Cezanne | Ukiyo-e

https://junyanz.github.io/CycleGAN/

영상 변환 사례 (II)



https://github.com/yunjey/stargan

딥페이크 예시

Living portraits

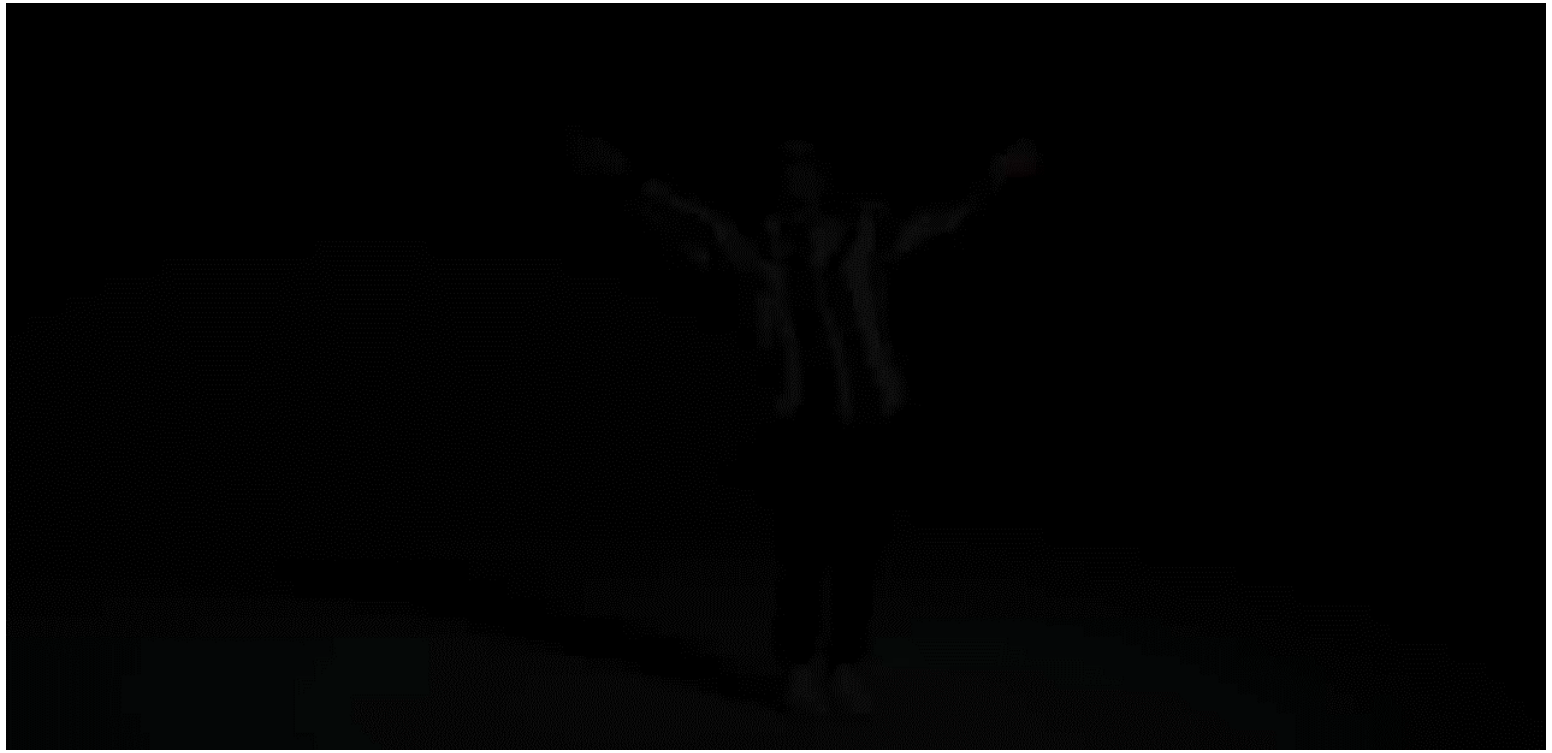
Demo: https://youtu.be/p1b5aiTrGzY

Zakharov1 et al., “Few-Shot Adversarial Learning of Realistic Neural Talking Head Models,”, ICCV’19

모션 딥페이크 예시

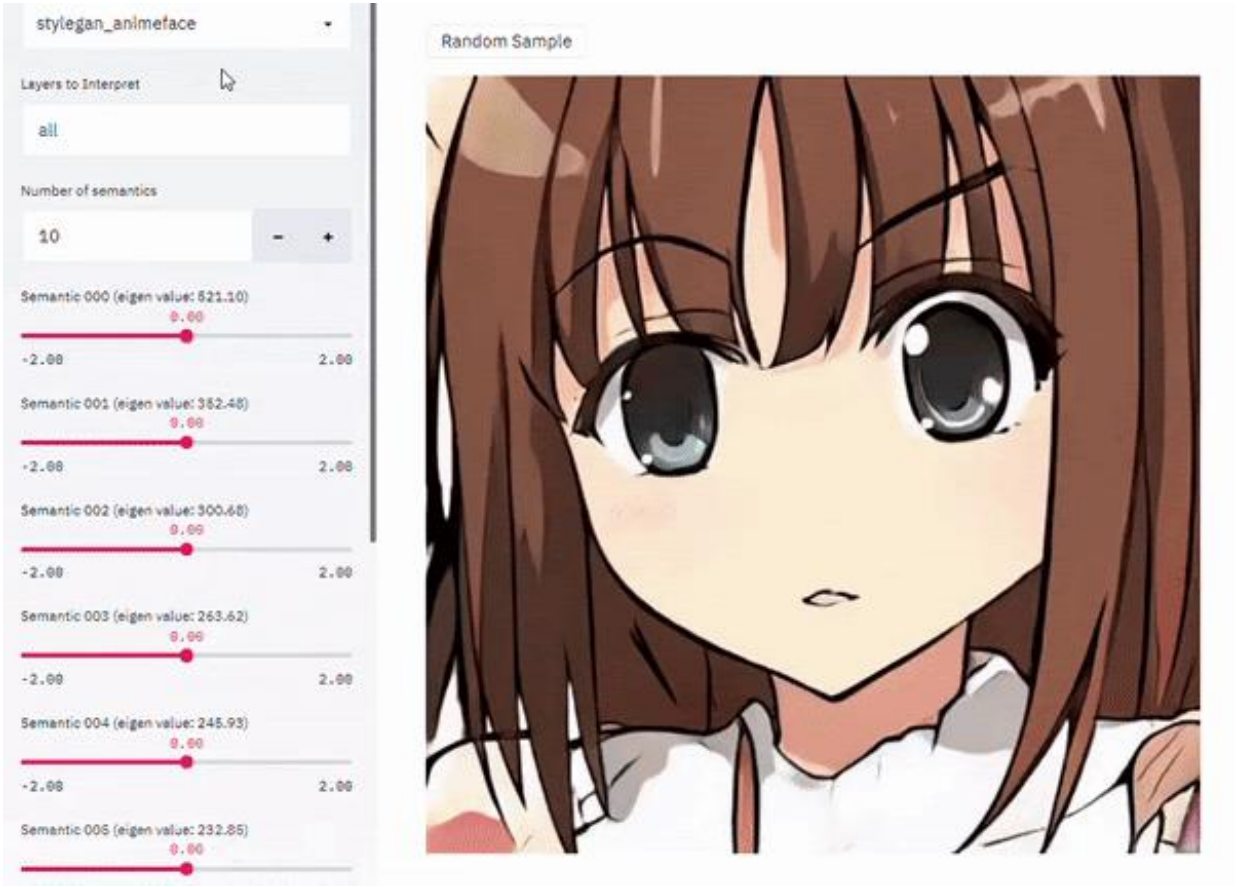
Chan et al, Everybody Dance Now, ICCV'19

Demo: https://youtu.be/PCBTZh41Ris



Zakharov1 et al., "Few-Shot Adversarial Learning of Realistic Neural Talking Head Models,", ICCV'19

Image manipulation 예시



Shen et al., Closed-Form Factorization of Latent Semantics in GANs, CVPR'21

Ch 1. 생성 모델 소개

Clip 2. 확률 밀도 함수 추정 및 샘플링

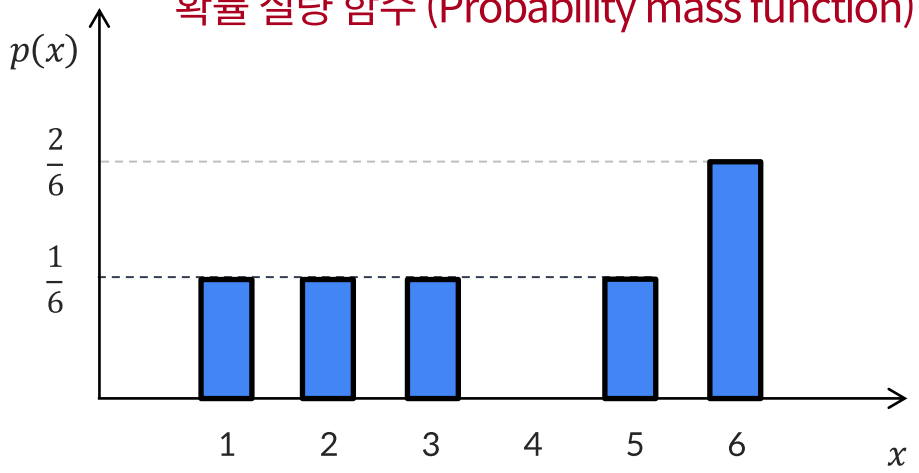
확률 분포 (Probability distribution)의 추정 (estimation)

확률 기초 (복습)

확률 변수 (Random variable)

X	1	2	3	4	5	6
$P(X)$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{0}{6}$	$\frac{1}{6}$	$\frac{2}{6}$

확률 질량 함수 (Probability mass function)



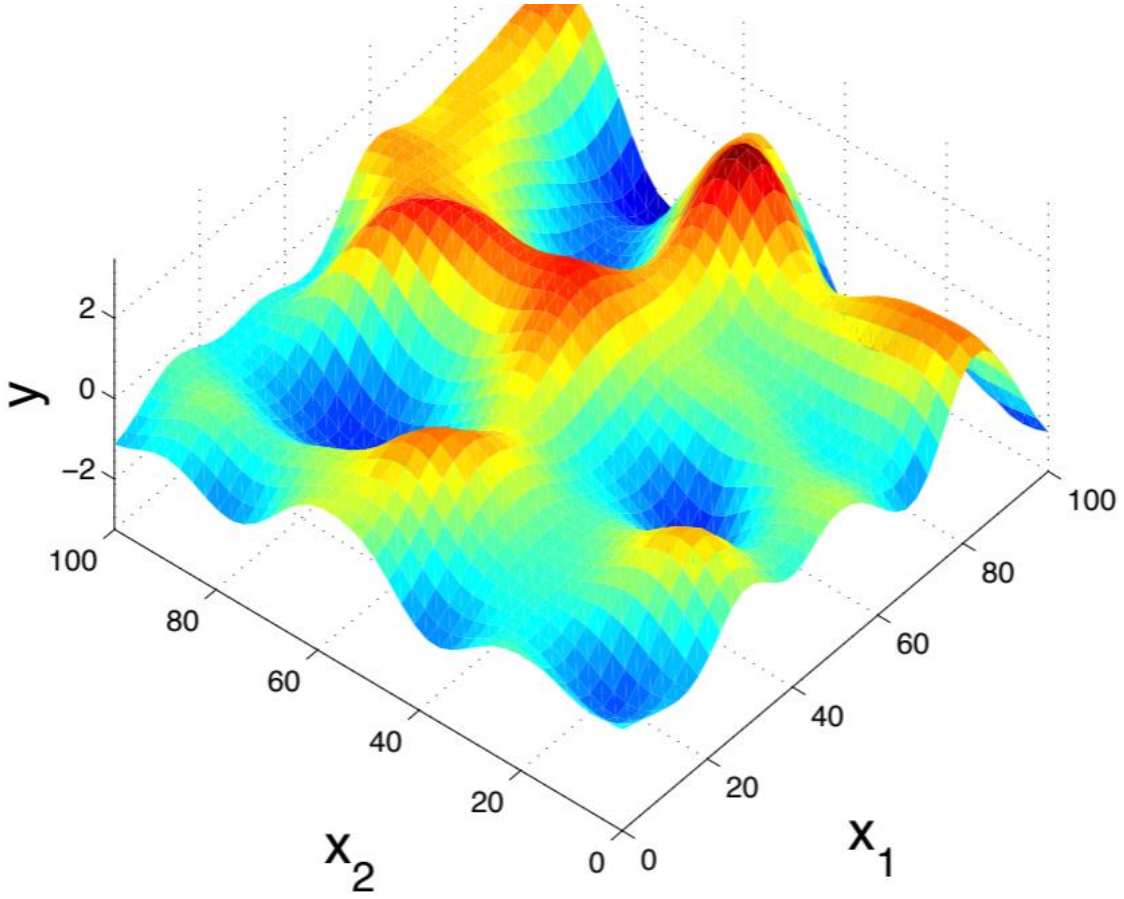
확률 분포 (Probability distribution)의 추정 (estimation)

만약 여기서 x 가 학습 데이터 내의 실제 이미지라면?

이 때, 이미지 x 는 64×64×3와 같은 차원을 갖는 고차원의 벡터 (vector) 로 표현될 수 있음

확률 분포 (Probability distribution)의 추정 (estimation)

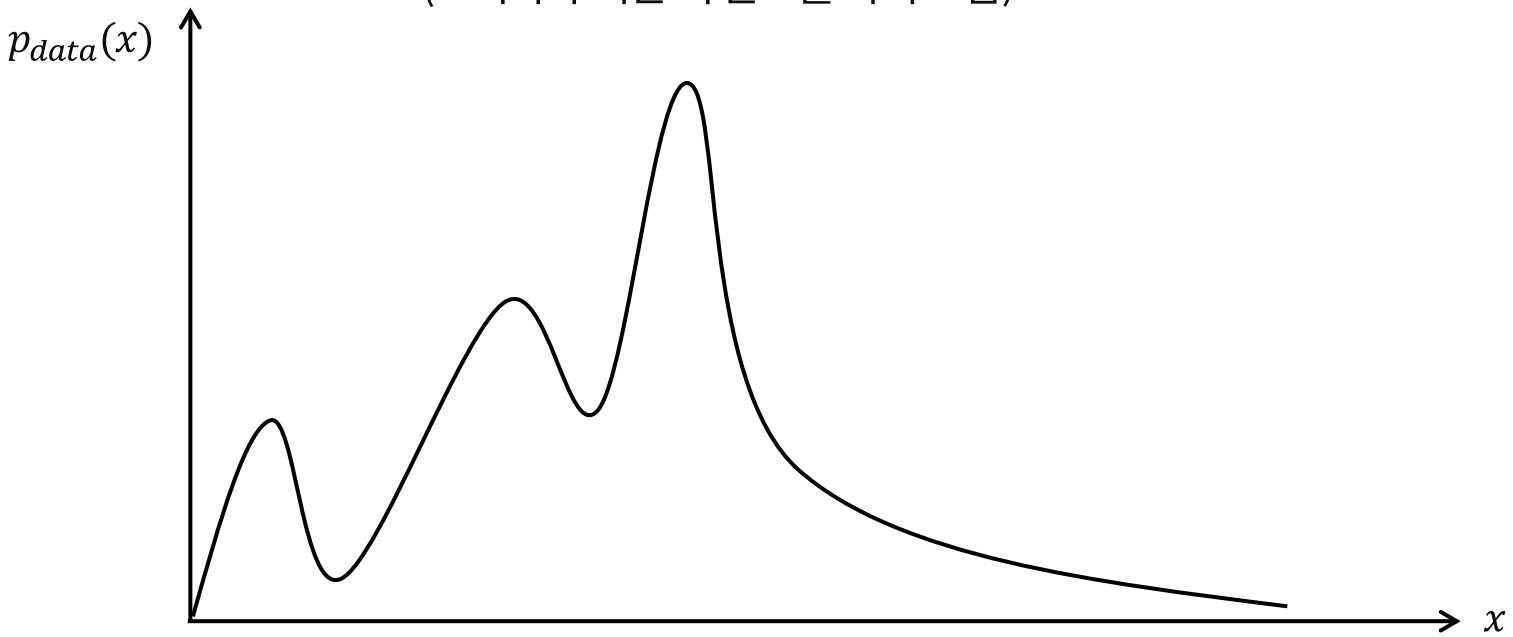
고차원 벡터에 대한 확률 분포도 생각할 수 있음



확률 분포 (Probability distribution)의 추정 (estimation)

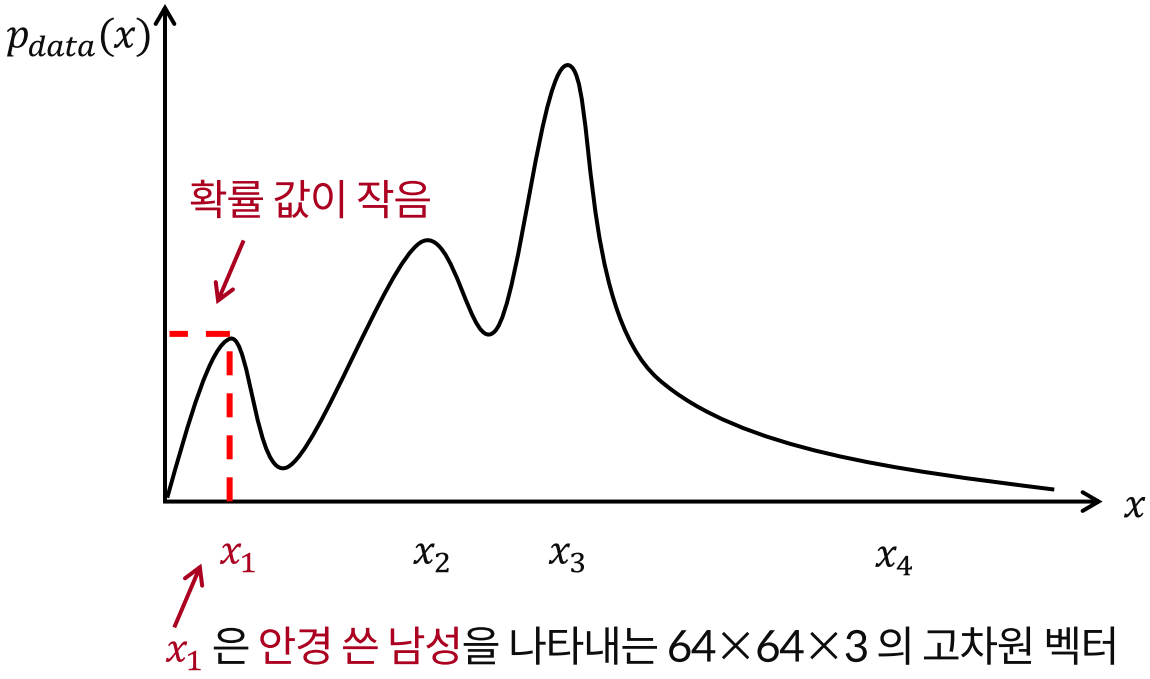
확률 밀도 함수 (Probability density function)

$p_{data}(x)$ 는 실제 이미지들의 정확한 분포를 나타냄
(그러나 우리는 이 분포를 아직 모름)



확률 분포 (Probability distribution)의 추정 (estimation)

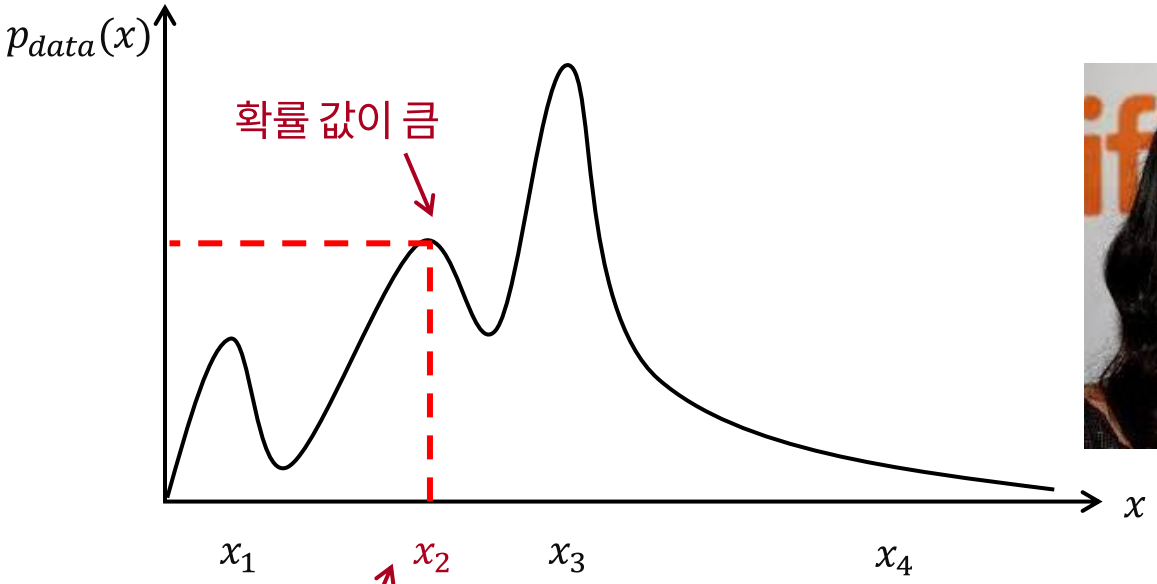
사람 얼굴 이미지 데이터셋의 예시로 살펴보면,
이 데이터셋은 안경을 쓴 남성의 이미지들을 몇몇 포함하고 있을 수 있음



x_1 은 안경 쓴 남성을 나타내는 64×64×3 의 고차원 벡터

확률 분포 (Probability distribution)의 추정 (estimation)

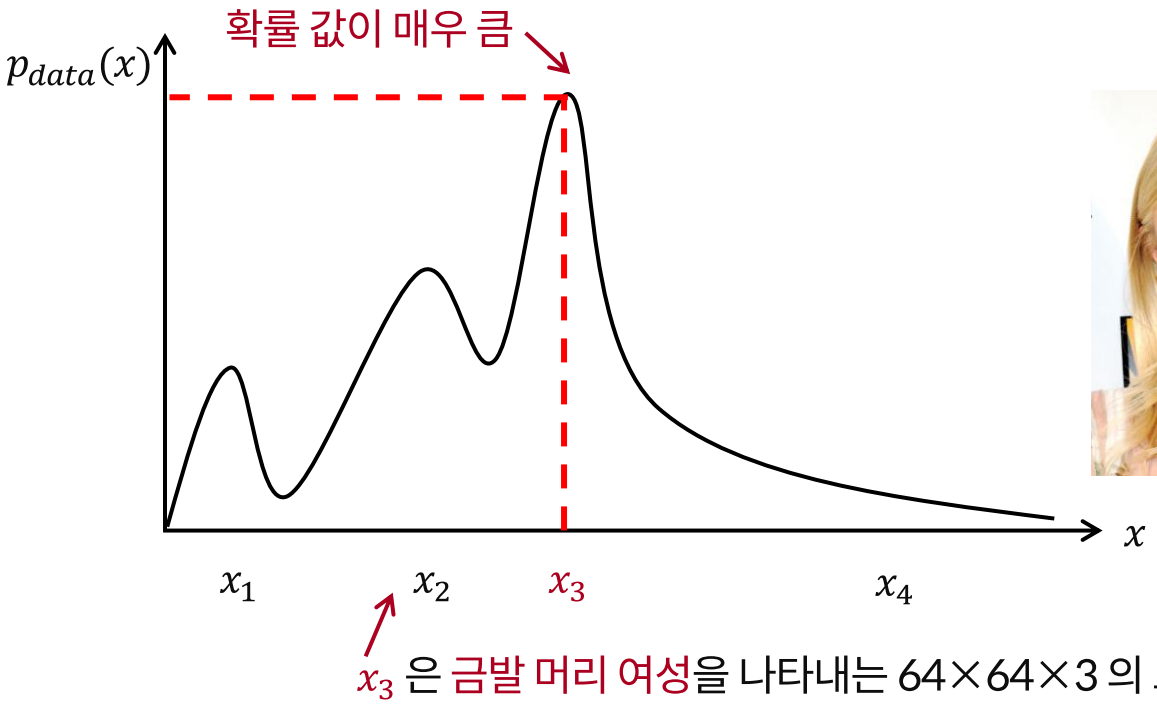
이 데이터셋은 검은 머리 여성의 이미지들을 포함하고 있을 수 있음



x_2 은 검은 머리 여성을 나타내는 64×64×3 의 고차원 벡터

확률 분포 (Probability distribution)의 추정 (estimation)

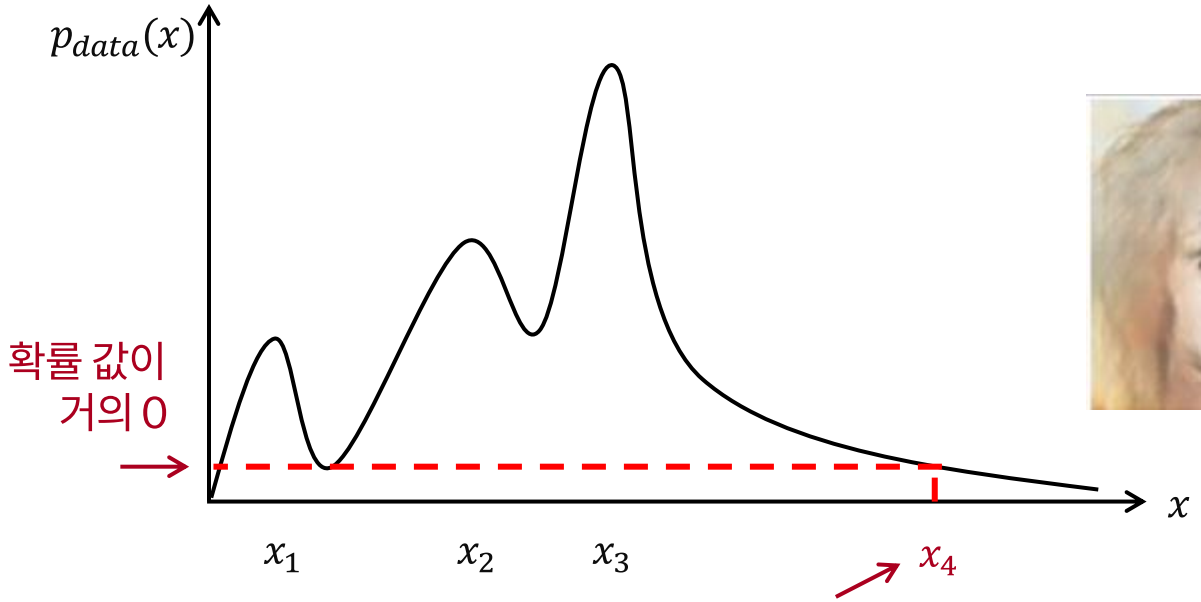
이 데이터셋은 금발 머리 여성의 이미지들을 다수 포함하고 있을 수 있음



x_3 은 금발 머리 여성을 나타내는 64×64×3 의 고차원 벡터

확률 분포 (Probability distribution)의 추정 (estimation)

이 데이터셋은 오른쪽과 같은 이상한 이미지의 이미지들을 포함하고 있을 수 있음



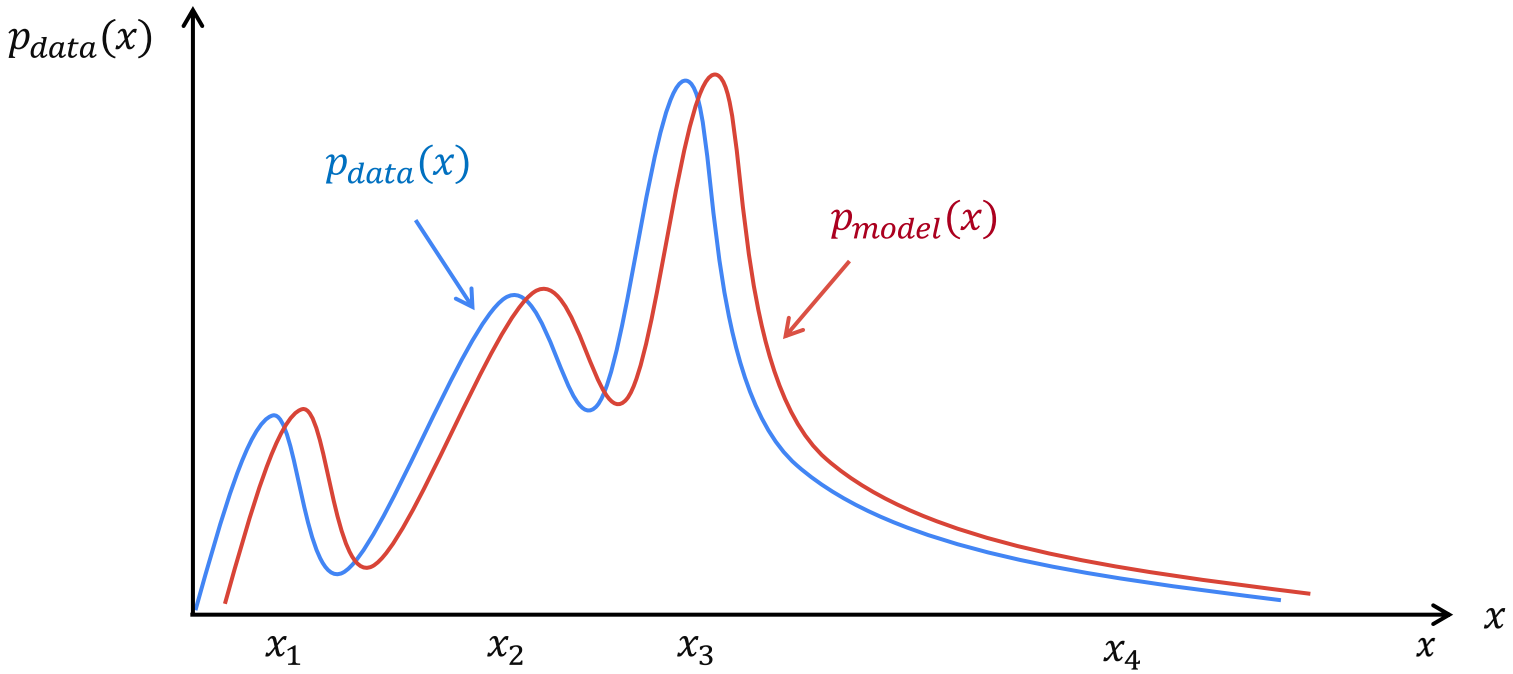
x_4 는 매우 이상한 이미지를 나타내는 64×64×3 의 고차원 벡터

확률 분포 (Probability distribution)의 추정 (estimation)

모델이 생성한 이미지들의 분포

생성 모델은 $p_{data}(x)$ 를 잘 근사하는 $p_{model}(x)$ 를 찾는 것이 목표

실제 이미지들의 분포



Ch 1. 생성 모델 소개

Clip 3. Variational AutoEncoder (VAE)의 이해

생성 모델의 분류 체계

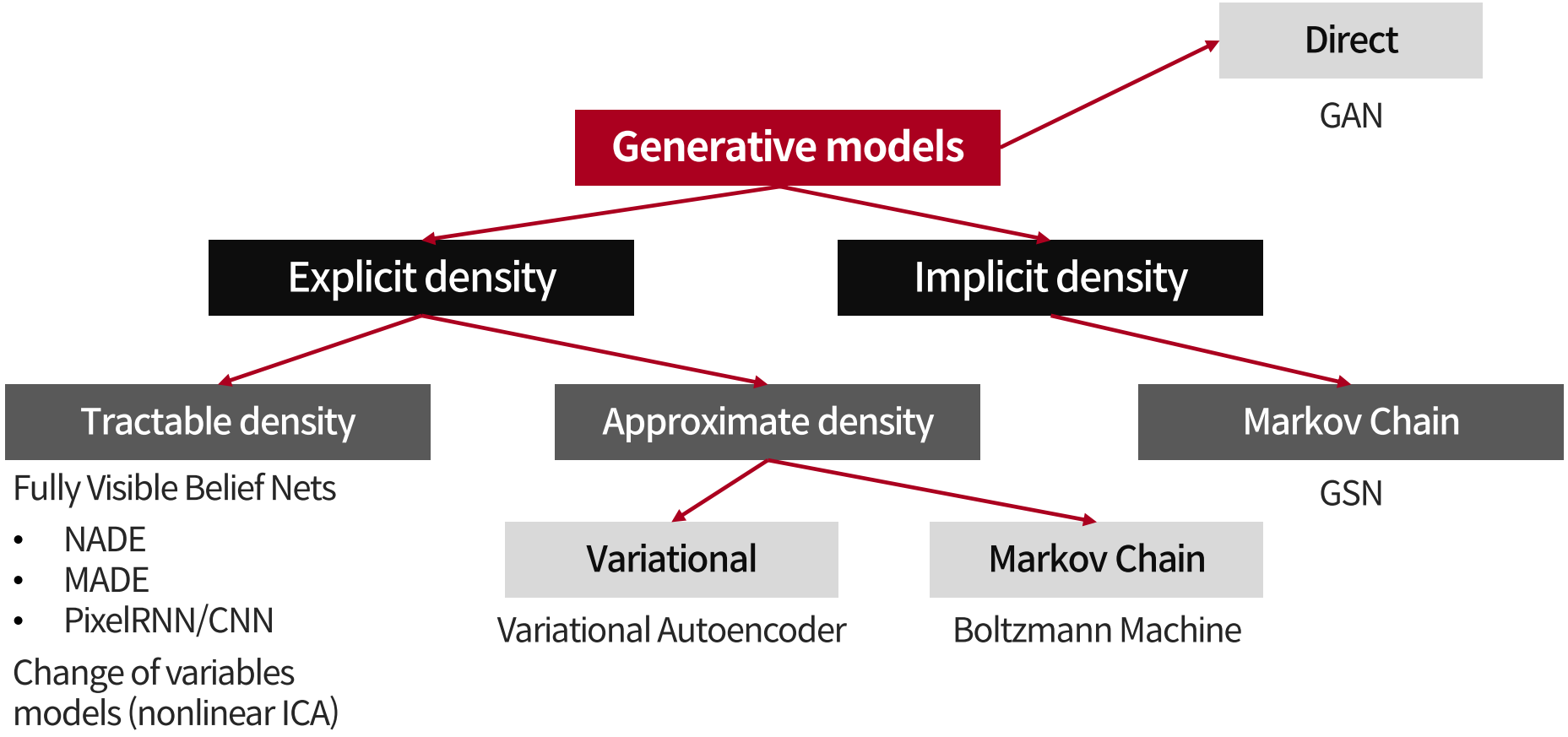
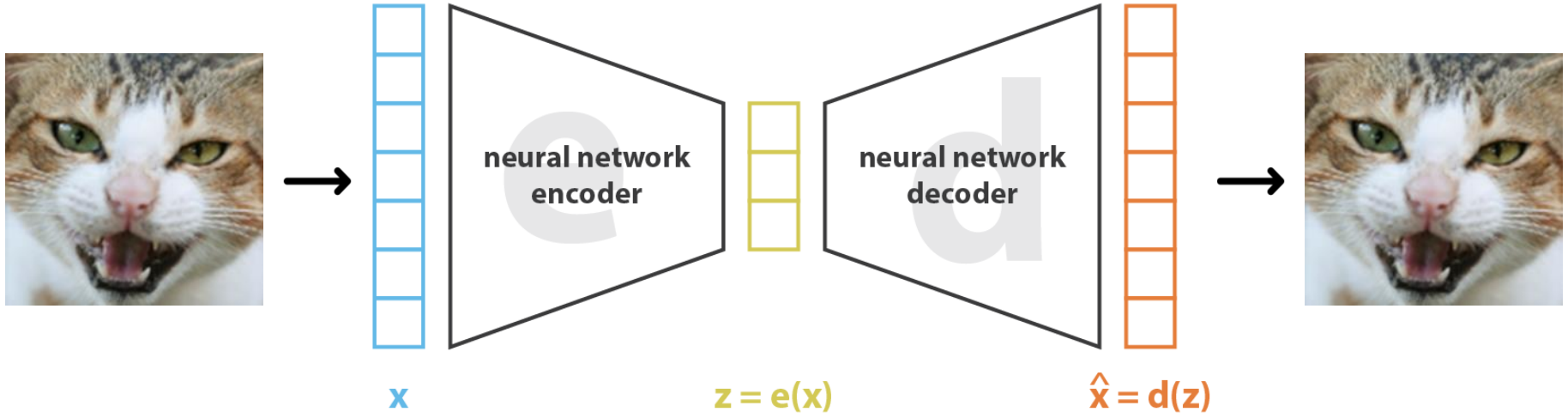


Figure from Ian Goodfellow,s Tutorial on Generative Adversarial Networks, 2017.

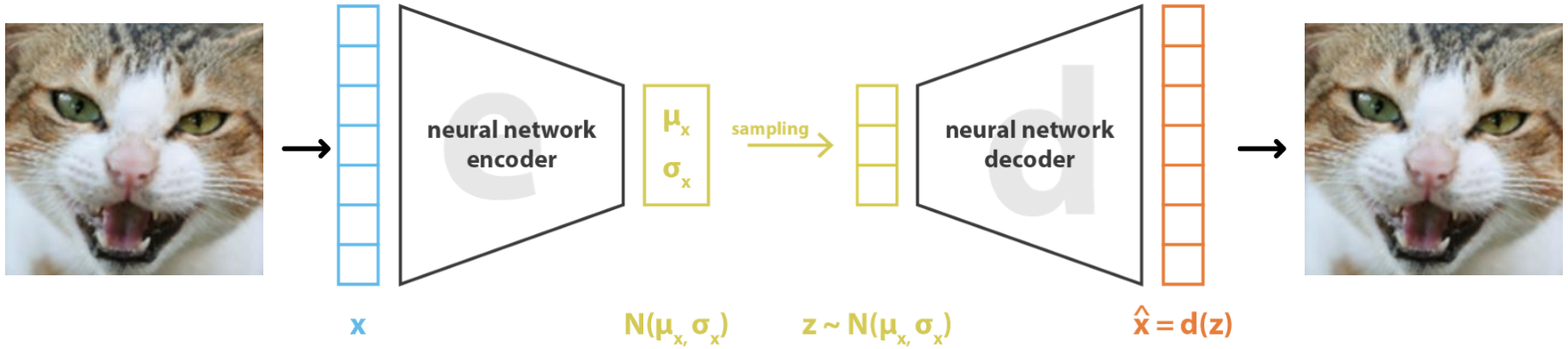
Variational Auto-Encoder

Traditional auto-encoder



http://kvfrans.com/variational-autoencoders-explained/, https://github.com/GunhoChoi/PyTorch-FastCampus/blob/master/08_Autoencoder/%5B2%EA%B8%B0%5DAutoEncoder.pdf

Variational Auto-Encoder



$$\text{loss} = \|\text{x} - \hat{\text{x}}\|^2 + KL\big(N(\mu_\text{x}, \sigma_\text{x})||N(0, I)\big) = \|\text{x} - \text{d}(\text{z})\|^2 + KL\big(N(\mu_\text{x}, \sigma_\text{x})||N(0, I)\big)$$

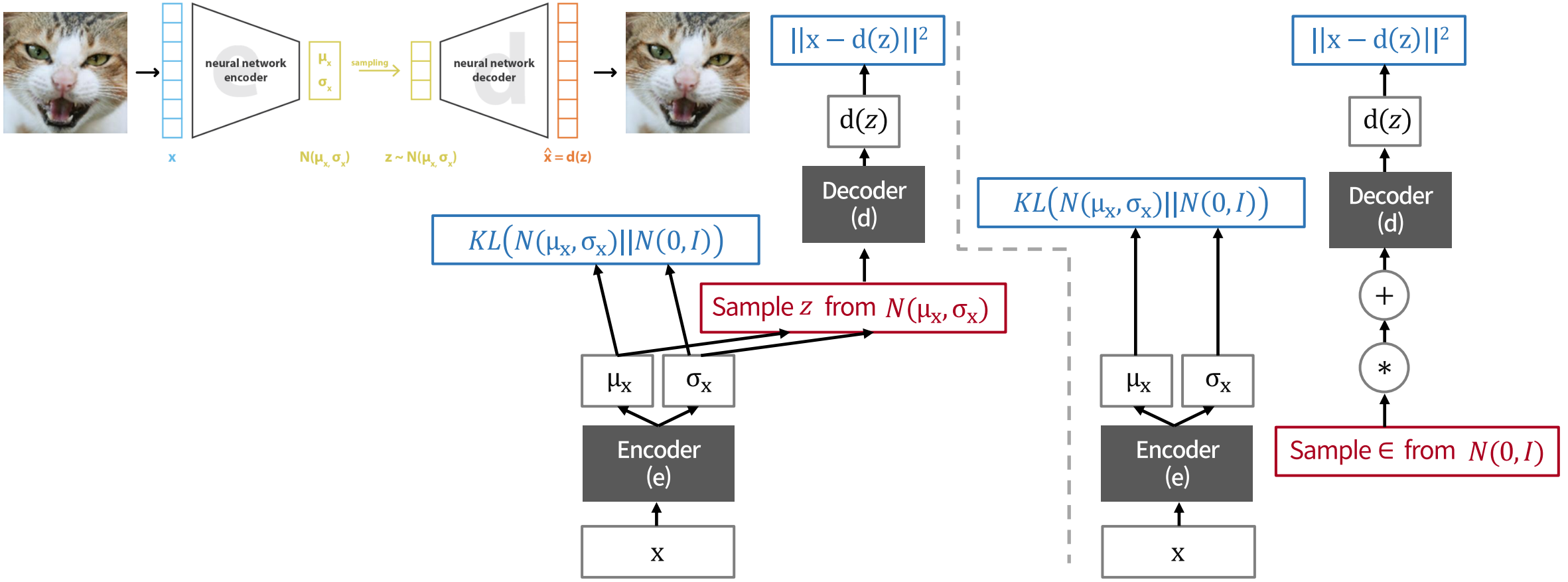
http://kvfrans.com/variational-autoencoders-explained/, https://github.com/GunhoChoi/PyTorch-FastCampus/blob/master/08_Autoencoder/%5B2%EA%B8%B0%5DAutoEncoder.pdf

Ch 1. 생성 모델 소개

Clip 4. Variational AutoEncoder (VAE)의 학습 과정 및 생성 이미지 결과

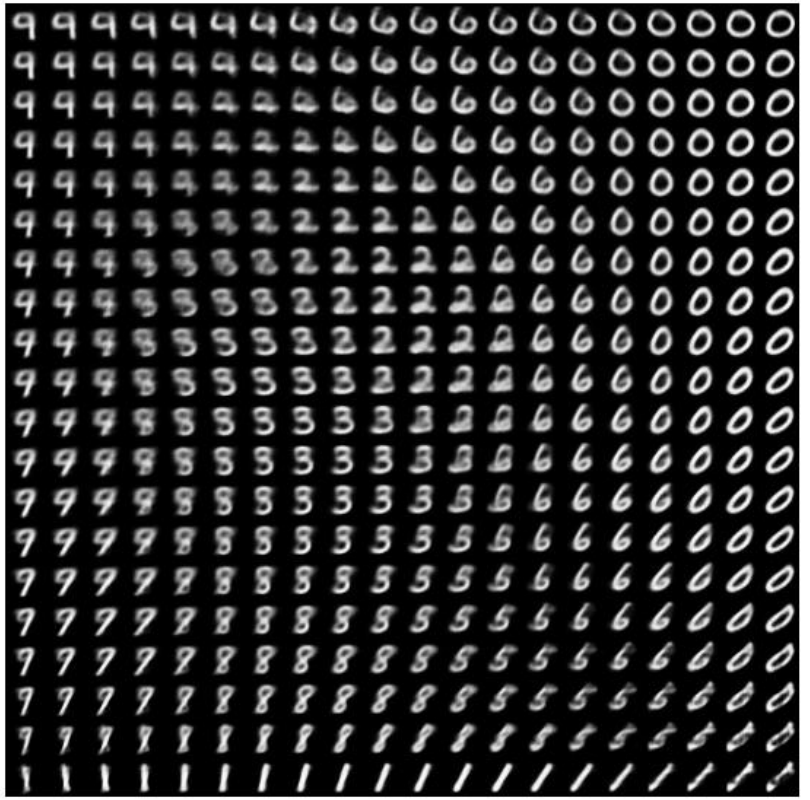
Variational Auto-Encoder

Reparametrization trick



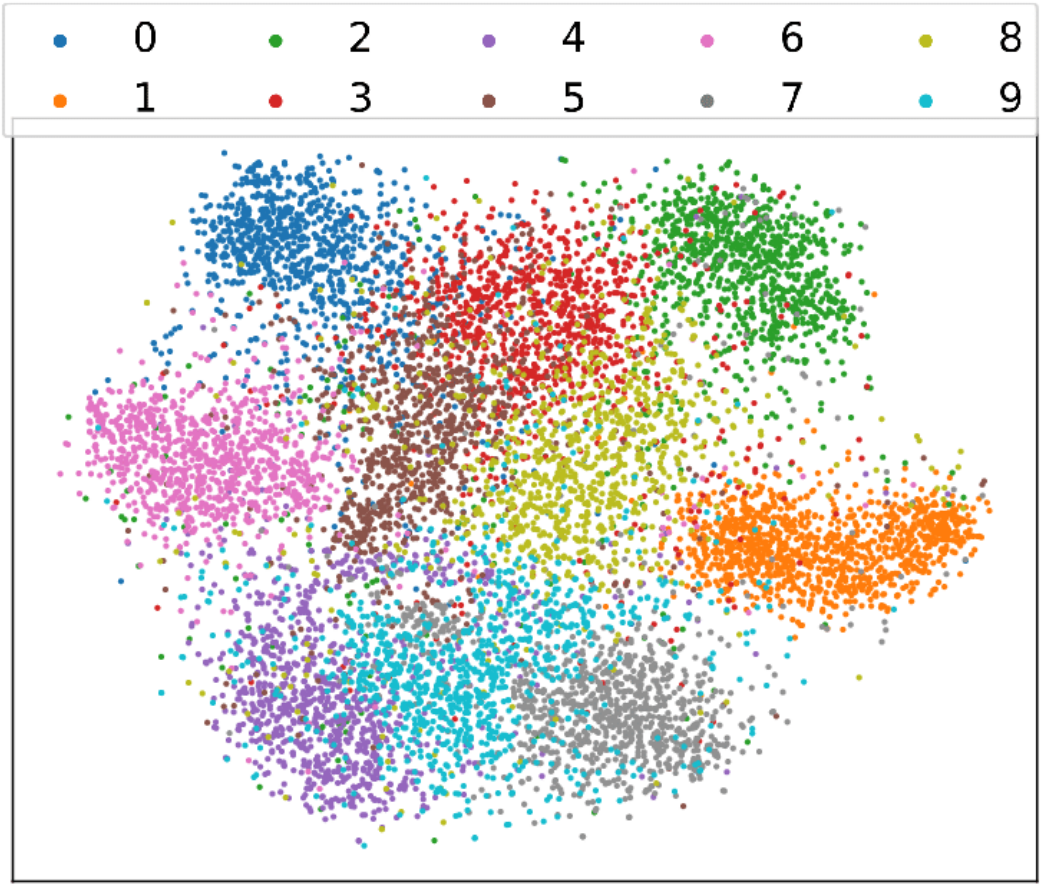
Variational Auto-Encoder

생성된 결과 이미지들



Variational Auto-Encoder

벡터 z 의 2D 임베딩 시각화 결과



https://www.researchgate.net/publication/320727335_Deep_Generative_Dual_Memory_Network_for_Continual_Learning

Variational Auto-Encoder

simple and stable

학습 프로세스가
간단하면서 안정적임

pure generation

순수 생성을 위한
목적으로
사용될 수 있음

blurry

종종 흐릿한 이미지를
생성함

Ch 2. Generative Adversarial Networks (GANs)

Clip 1. GAN이란 무엇인가?

Generative Adversarial Networks (GANs)

- **문제** 복잡하고 고차원인 학습 분포로부터 데이터를 샘플링을 하고자 하나, 이를 직접적으로 하는 것이 불가능

- **해결책** 쉽게 데이터를 샘플링하는 것이 가능한 간단한 분포를 이용
(예를 들면, random noise와 같은 것)
이 간단한 분포를 학습 분포로 변형 (transformation) 하는 법을 학습

Q: 이러한 복잡한 변형을 표현하려면
어떤 것을 사용해야 할까?

Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks (GANs)

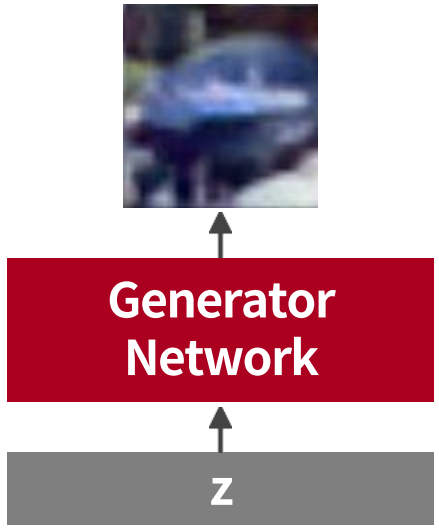
- **문제** 복잡하고 고차원인 학습 분포로부터 데이터를 샘플링을 하고자 하나, 이를 직접적으로 하는 것이 불가능

- **해결책** 쉽게 데이터를 샘플링하는 것이 가능한 간단한 분포를 이용 (예를 들면, random noise와 같은 것)
  이 간단한 분포를 학습 분포로 변형 (transformation) 하는 법을 학습

Q : 이러한 복잡한 변형을 표현하려면 어떤 것을 사용해야 할까?

A : 정답은 neural network!

Output : Sample from training distribution



Input : Random noise

Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks (GANs)

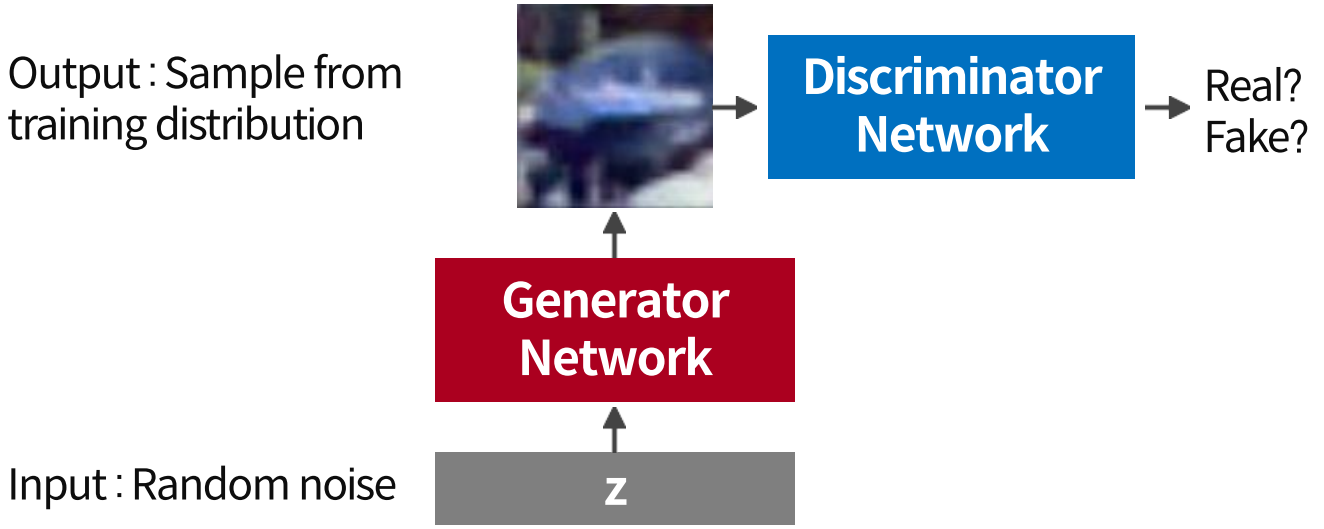
- **문제** 복잡하고 고차원인 학습 분포로부터 데이터를 샘플링을 하고자 하나, 이를 직접적으로 하는 것이 불가능

- **해결책** 쉽게 데이터를 샘플링하는 것이 가능한 간단한 분포를 이용 (예를 들면, random noise와 같은 것)
이 간단한 분포를 학습 분포로 변형 (transformation) 하는 법을 학습

하지만,
각 sample z가 어떤 학습 이미지로
매핑이 되는지 알 수 없음

→ 학습 이미지를 복원하는 것으로는 학습 불가능

해결책!
discriminator network를 사용하여
생성된 이미지가 데이터 분포 내에
속하는지 판단하도록 함



Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

GAN 학습 : Two-player game

Discriminator network

진짜 이미지와
가짜 이미지를 구분

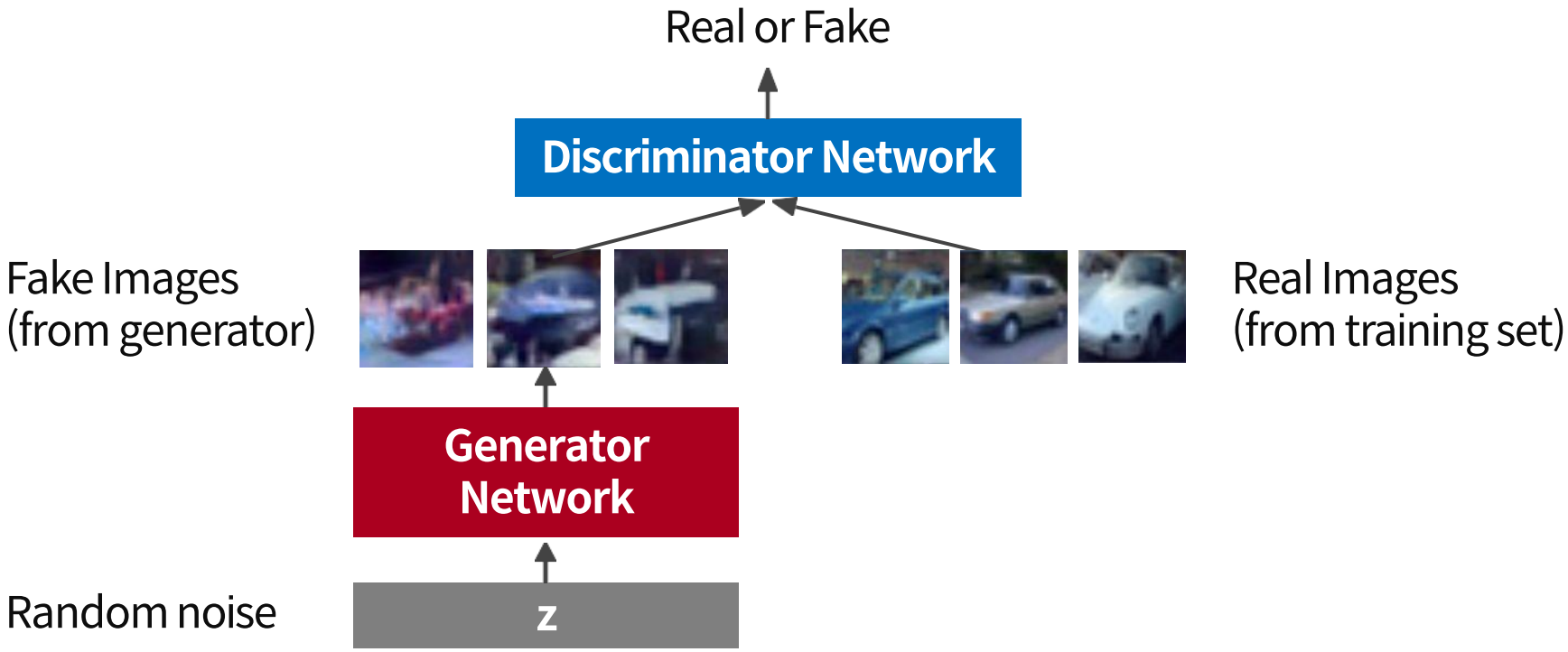
Generator network

진짜처럼 보이는 이미지를 생성해서
Discriminator를 속임

Ian Goodfellow et al., “Generative Adversarial Nets”, NeurIPS 2014

GAN 학습 : Two-player game

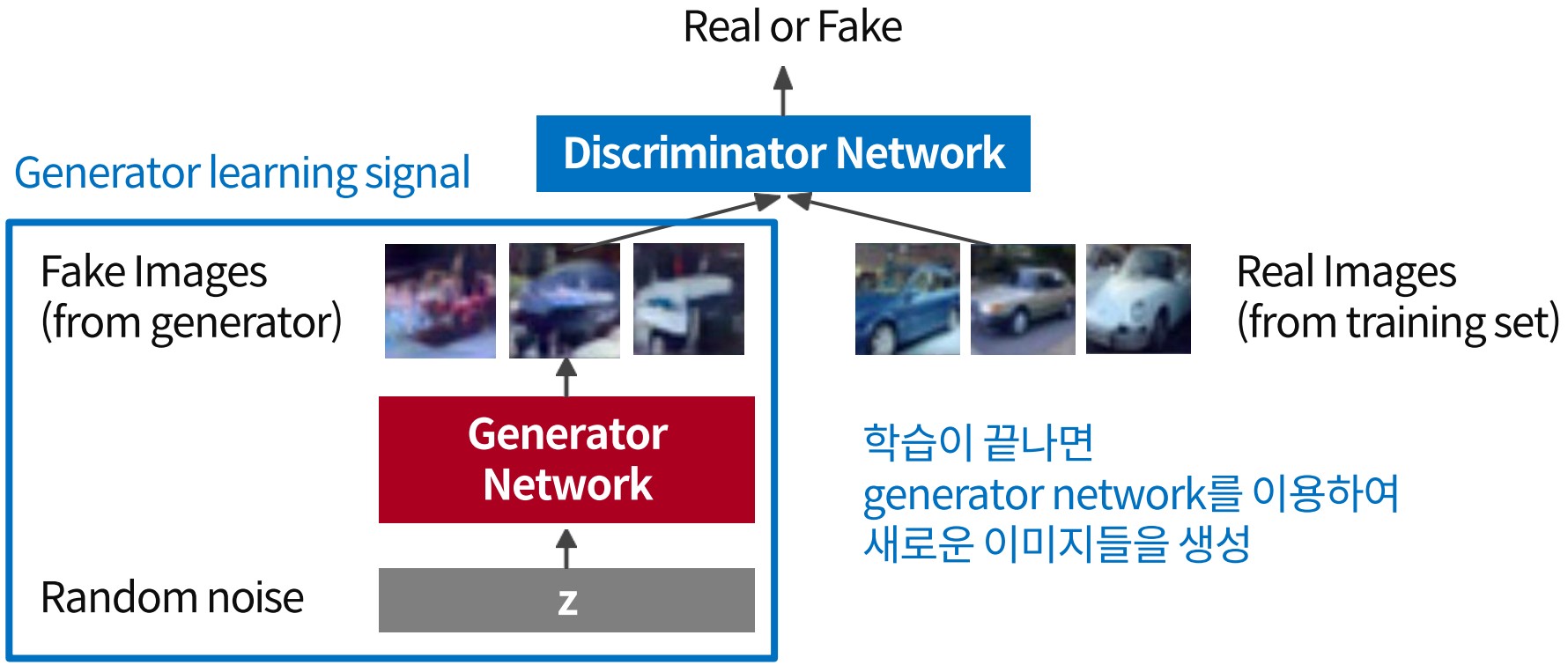
- **Discriminator network** 진짜 이미지와 가짜 이미지를 구분
- **Generator network** 진짜처럼 보이는 이미지를 생성해서 Discriminator를 속임



Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

GAN 학습 : Two-player game

- **Discriminator network** 진짜 이미지와 가짜 이미지를 구분
- **Generator network** 진짜처럼 보이는 이미지를 생성해서 Discriminator를 속임



Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

GAN의 직관적 이해

x

진짜 이미지

D

Discriminator

D(x)

아마도 큰 값

x 가 실제 이미지일 확률
(0~1)

진짜 이미지로 학습

가짜 이미지로 학습

Generator

z

Latent Code

G

G(z)

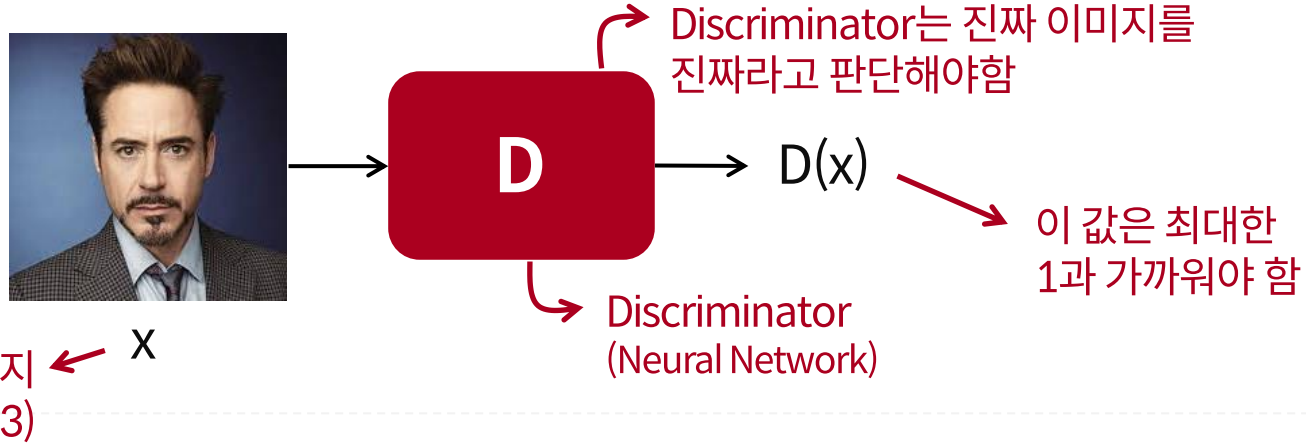
가짜 이미지

D

D(G(z))

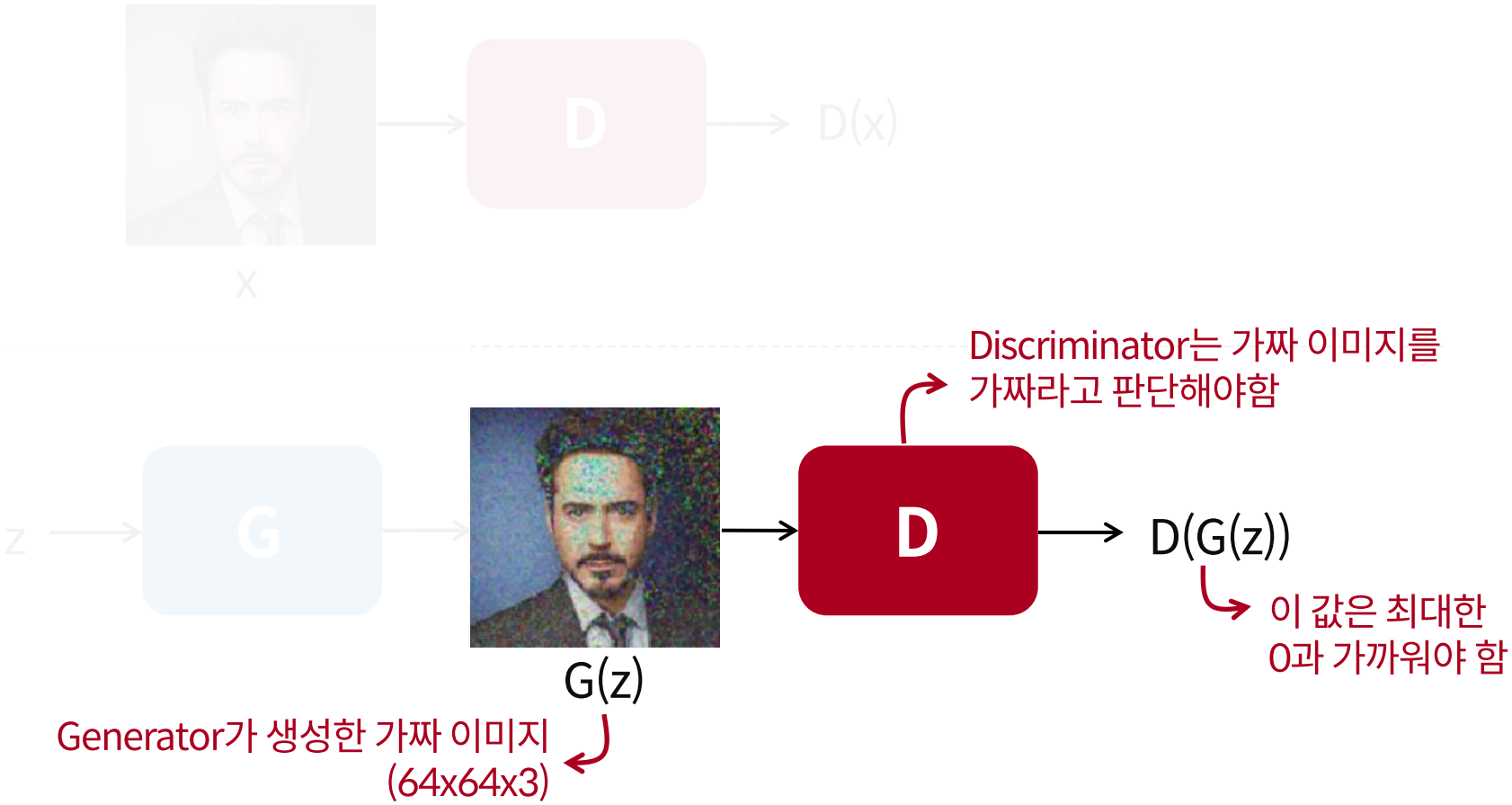
아마도
작은 값

GAN의 직관적 이해

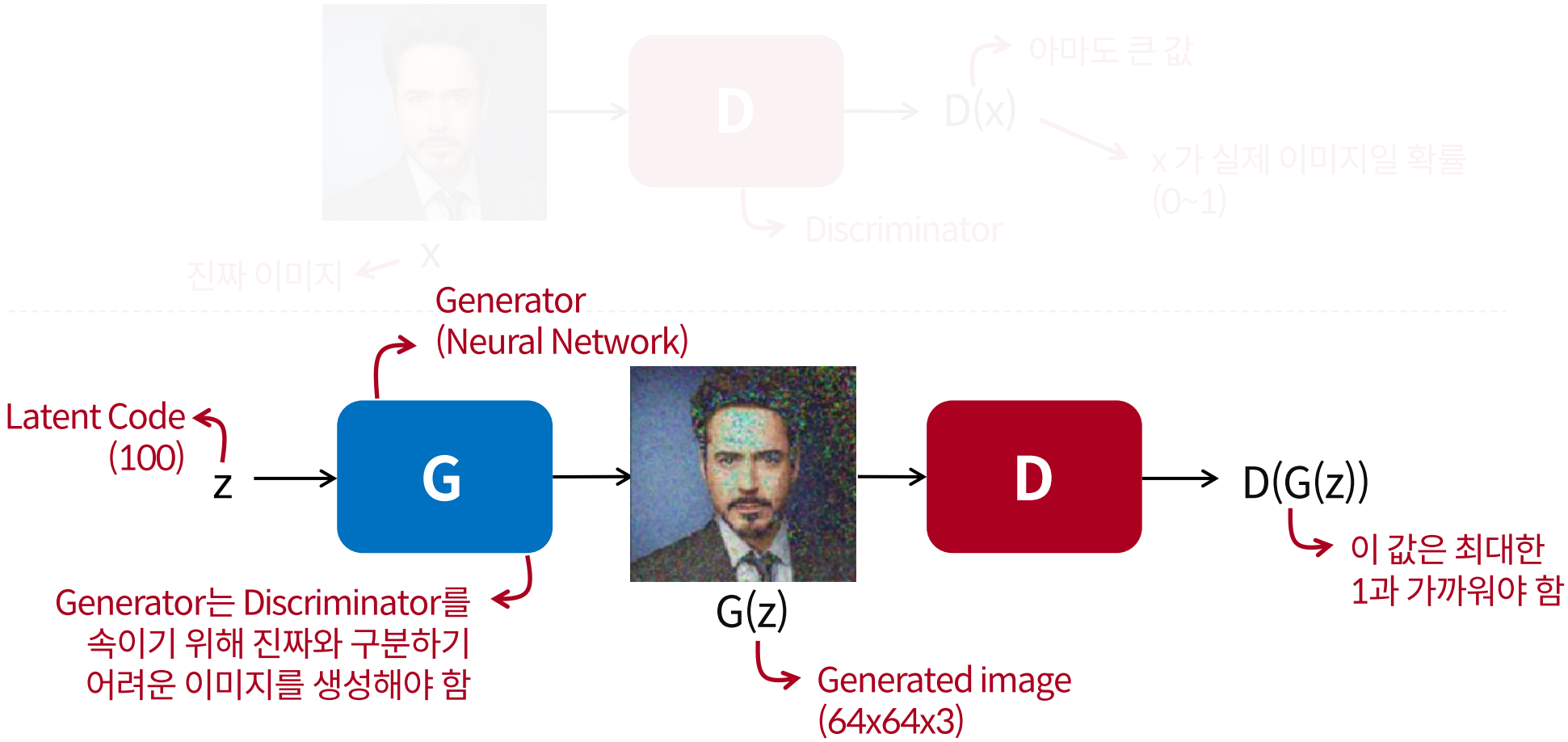

Discriminator는 진짜 이미지를
진짜라고 판단해야함
D
D(x)
이 값은 최대한
1과 가까워야 함
Discriminator
(Neural Network)
x
진짜 이미지
(64x64x3)

z
G
D
D(G(z))
G(z)

GAN의 직관적 이해


D
D(x)
x
Discriminator는 가짜 이미지를
가짜라고 판단해야함
z
G
D
D(G(z))
이 값은 최대한
0과 가까워야 함
G(z)
Generator가 생성한 가짜 이미지
(64x64x3)

GAN의 직관적 이해


D
아마도 큰 값
D(x)
x 가 실제 이미지일 확률
(0~1)
Discriminator
진짜 이미지
x
Generator
(Neural Network)
Latent Code
(100)
z
G
G(z)
D
D(G(z))
이 값은 최대한
1과 가까워야 함
Generator는 Discriminator를
속이기 위해 진짜와 구분하기
어려운 이미지를 생성해야 함
Generated image
(64x64x3)

Ch 2. Generative Adversarial Networks (GANs)

Clip 2. GAN의 목적 함수

GAN의 목적 함수

실제 데이터 분포로부터 x를 샘플링

정규 분포로부터 latent code z를 샘플링

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

D 는 $L(D,G)$를 최소화해야 함

$D(x) = 1$일 때 최소가 됨

$D(G(z)) = 0$일 때 최소가 됨

목적 함수

x

D

D(x)

D가 진짜 이미지를 진짜로 분류하도록 학습

진짜 이미지로 학습

가짜 이미지로 학습

z

G

G(z)

D

D(G(z))

D가 가짜 이미지를
가짜로 분류하도록 학습

GAN의 목적 함수

G 는 이 부분과는 관련이 없음

$$\max_G \min_D L(D,G) = \cancel{E_{x\sim p_{data}(x)}[-\log D(x)]} + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

G 는 $L(D,G)$를 최대화해야 함

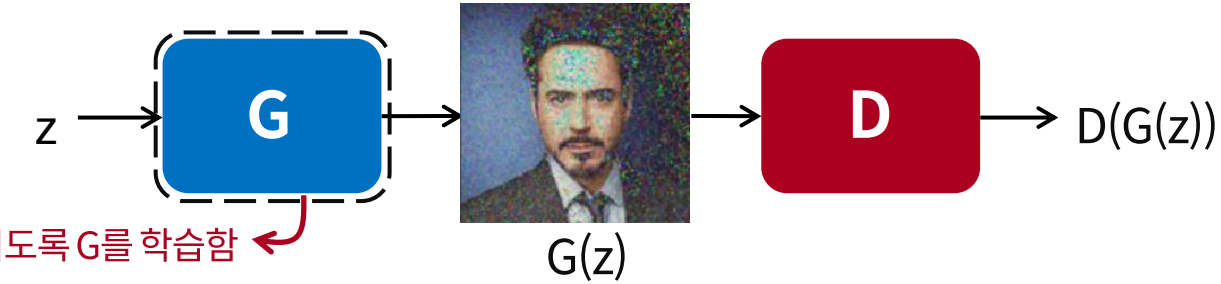
$D(G(z)) = 1$ 일 때 최대가 됨

목적 함수

x

진짜 이미지로 학습

가짜 이미지로 학습



G(z)

GAN 학습 : Two-player game

- **Discriminator network** 진짜 이미지와 가짜 이미지를 구분
- **Generator network** 진짜처럼 보이는 이미지를 생성해서 Discriminator를 속임

minimax game 으로 두 network를 같이 학습

Minimax 목적 함수:

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

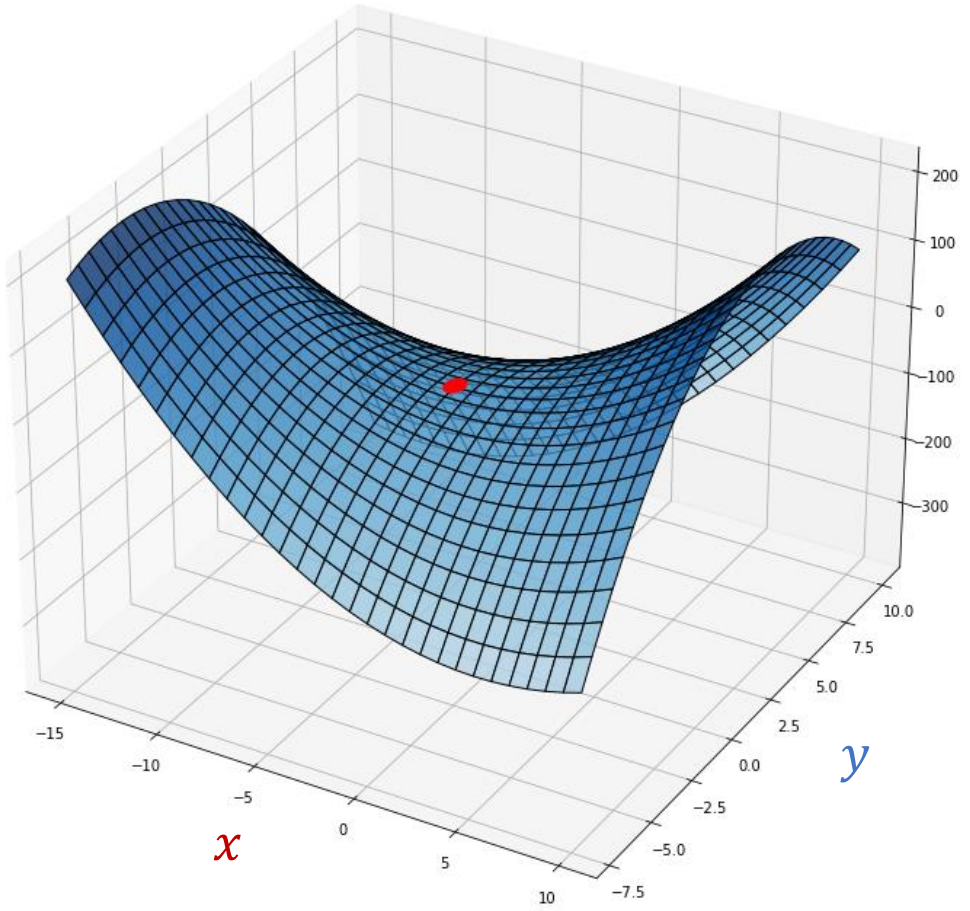
Generator objective

Discriminator objective

Mini-max optimization 문제

예를 들어, 다음의 mini−max optimization 문제를 생각해보자:

$$\min_{x} \max_{y} f(x, y) = x^2 + 2xy - 3y^2 + 4x + 5y + 6$$



$f(x, y)$

Mini-max optimization 문제

$$\min_{x}\max_{y} f(x,y) = x^2 + 2xy - 3y^2 + 4x + 5y + 6$$

(랜덤) 초기값 설정: $x = -1, y = -0.5$

$y = -0.5$ 로 고정할 때, x만의 함수:

$$f(x, y = -0.5) = x^2 - x - 0.75 + 4x - 2.5 + 6 = x^2 + 3x + 2.75$$

$x = -1$ 로 고정할 때, y만의 함수:

$$f(x = -1, y) = 1 - 2y - 3y^2 - 4 + 5y + 6 = -3y^2 + 3y + 3$$

x 만의 함수에 대해, x 에 대한 편미분을 구하면,

$$\frac{\partial f(x, y = -0.5)}{\partial x} = 2x + 3$$

x 만의 함수에 대해, x 에 대한 편미분을 구하면,

$$\frac{\partial f(x = -1, y)}{\partial y} = -6y + 3$$

Mini-max optimization 문제

$$\frac{\partial f(x,y=-0.5)}{\partial x} = 2x + 3$$
$$\frac{\partial f(x=-1,y)}{\partial y} = -6y + 3$$

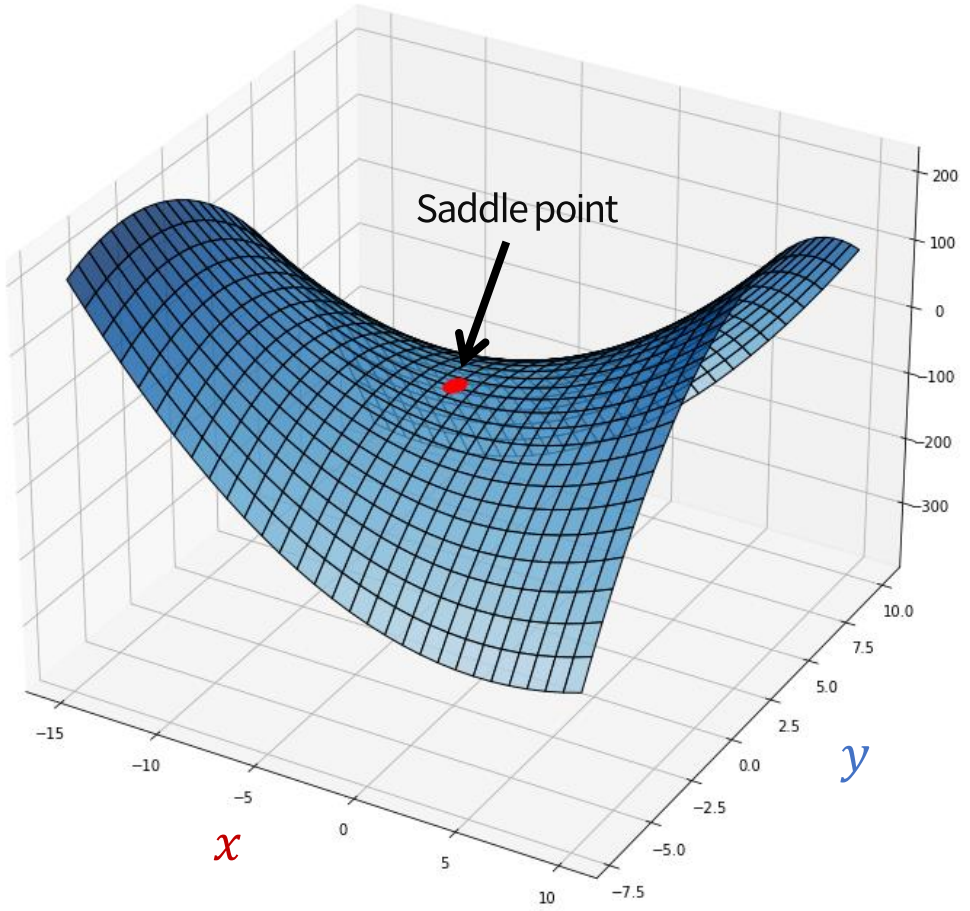
x는 $f(x, y)$를 최소화하는 방향으로 gradient descent를 수행:

$$x = -1 - \underbrace{0.1}_{\text{learning rate}} \times \left.\frac{\partial f(x, y=-0.5)}{\partial x}\right|_{x=-1} = -1 - 0.1 \times 1 = -1.1$$

y는 $f(x, y)$를 최대화하는 방향으로 gradient ascent를 수행:

$$y = -0.5 + \underbrace{0.1}_{\text{learning rate}} \times \left.\frac{\partial f(x=-1, y)}{\partial y}\right|_{y=-0.5} = -0.5 + 0.1 \times 6 = 0.1$$

Saddle Point: Mini-max optimization이 수렴하는 최종 솔루션



$f(x, y)$

GAN 학습 : Two-player game

Minimax 목적 함수:

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

아래의 두 과정을 번갈아 진행:

1. **Gradient descent** on discriminator

$$\min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

2. **Gradient ascent** on generator

$$\max_G L(D,G) = E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

GAN 학습 : Two-player game

Minimax 목적 함수:

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

아래의 두 과정을 번갈아 진행:

1. **Gradient descent** on discriminator

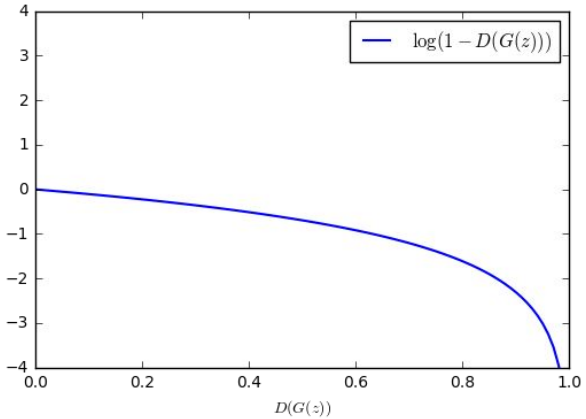
$$\min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

2. **Gradient ascent** on generator

$$\max_G L(D,G) = E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

하지만 실제 상황에서는 이러한 Generator 목적 함수가
잘 학습되지 않음

샘플이 가짜처럼 보일 때,
이 샘플을 통해 Generator를
학습하려고 함
(X 축의 오른쪽으로 옮기려 함)



Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

GAN 학습 : Two-player game

Minimax 목적 함수:

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

아래의 두 과정을 번갈아 진행:

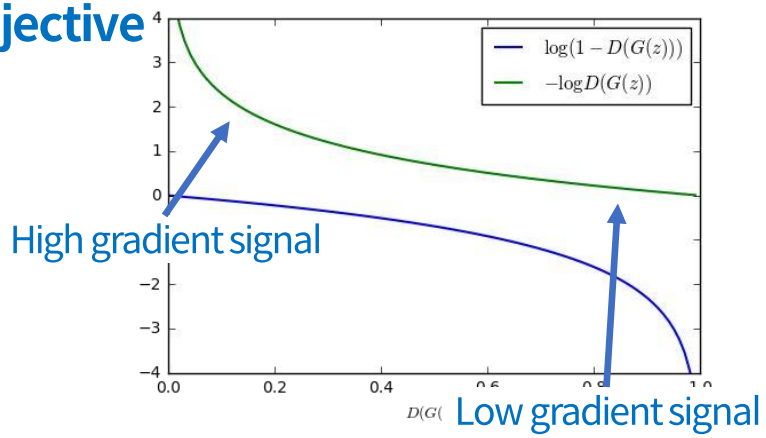
1. **Gradient descent** on discriminator

$$\min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

2. **Instead: Gradient descent** on generator using **different objective**

$$\max_G L(D,G) = E_{z\sim p_z(z)}[\log(D(G(z)))]$$

$$\leftrightarrow \min_G L(D,G) = E_{z\sim p_z(z)}[-\log(D(G(z)))]$$



기존과 동일하게 Discriminator를 속이기 위한 목적 함수이지만 가짜 같이 보
이는 샘플들에 대한 그라디언트가 커짐.
그 결과, 실제 학습에서 잘 작동함.

Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

GAN 학습 : Two-player game

앞의 내용을 합친 GAN 학습 알고리즘

For number of training iterations **do**
 for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \cdots, z^{(m)}\}$ from noise prior $P_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \cdots, x^{(m)}\}$ from the training set of real data.
- Update the discriminator by descending 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 $\{z^{(1)}, \cdots, z^{(m)}\}$ from noise prior $P_g(z)$.
- Update the generator by descending its stochastic gradient (improved objective):

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

k=1일 때 더 안정적으로
학습된다는 주장도 있고,
k>1 을 사용하는 경우도 존재함.
최적이라 판명된 방법론은 없음.

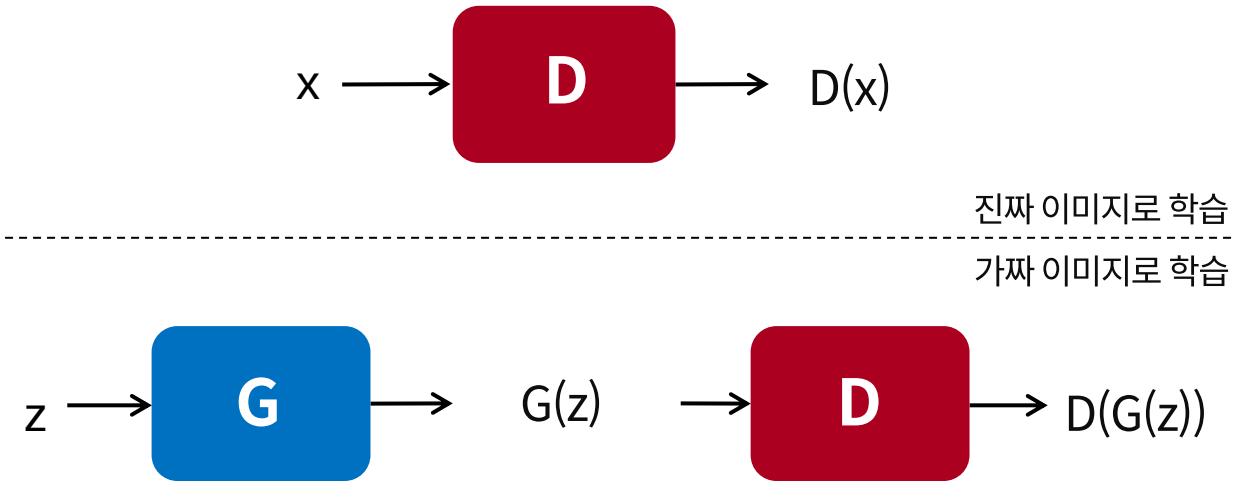
이후에 배울 LSGAN,
WGAN-GP 등의 방법론들이
이러한 문제를 완화하여
더 안정적인 학습을 도와줌.

Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014

Ch 2. Generative Adversarial Networks (GANs)

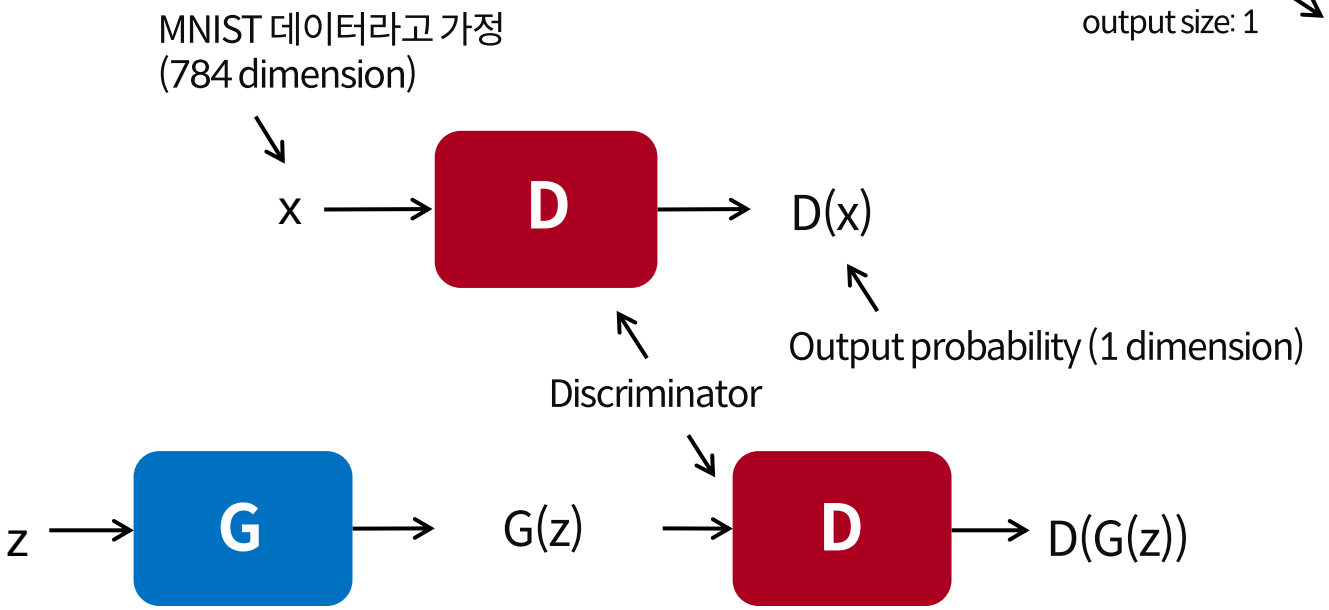
Clip 4. 코드로 본 GAN의 학습과정

PyTorch Implementation




```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

PyTorch Implementation

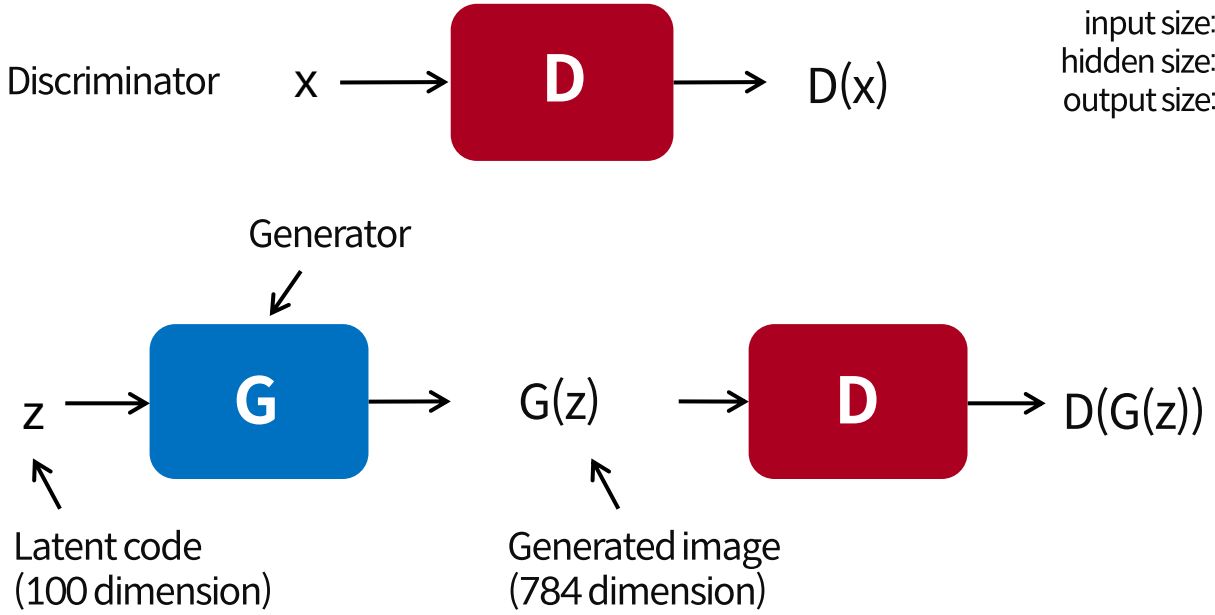


Discriminator 정의

input size: 784
hidden size: 128
output size: 1

```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

PyTorch Implementation

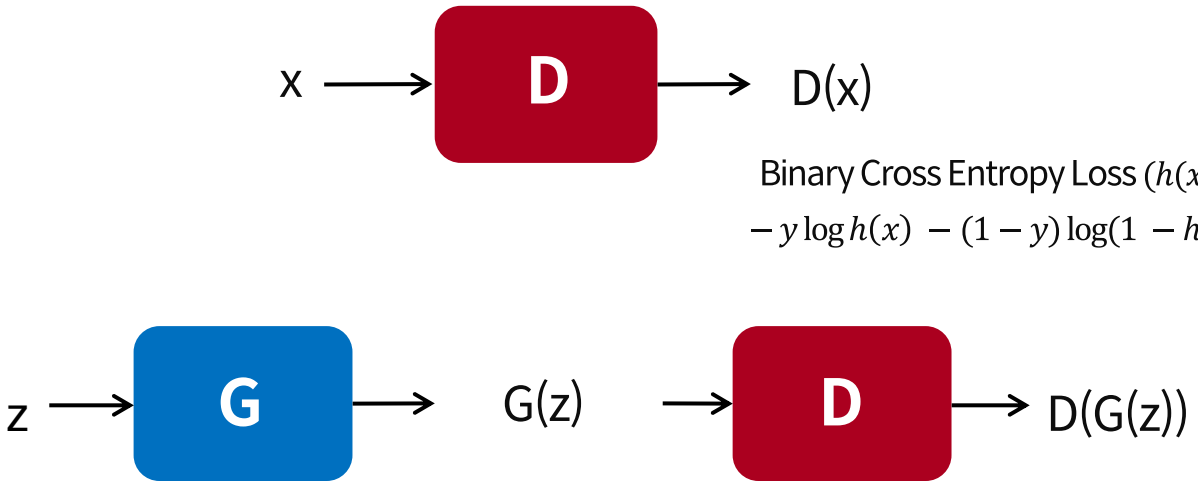


Generator 정의

input size: 100
hidden size: 128
output size: 784

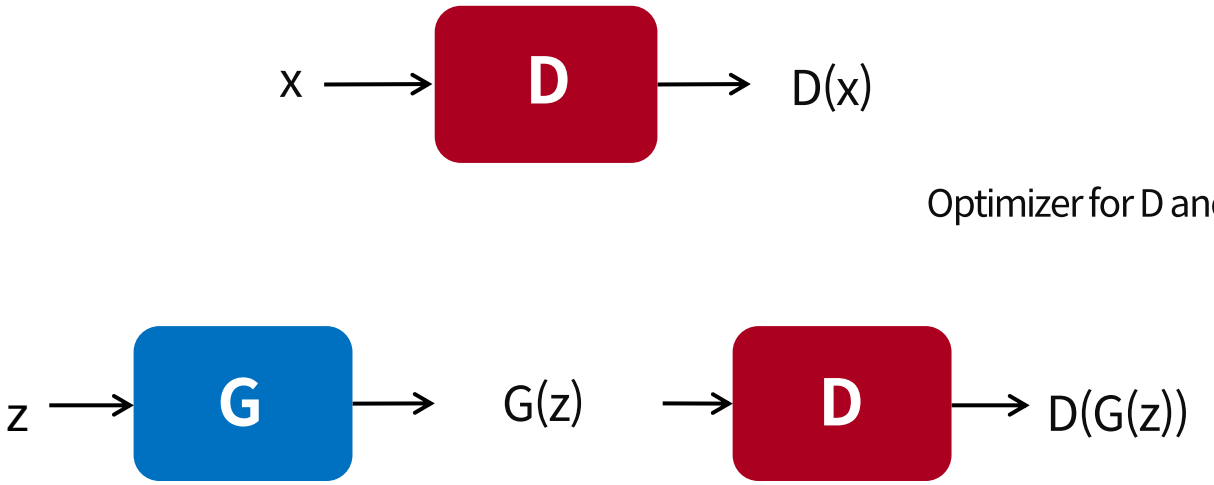
```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

PyTorch Implementation




```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

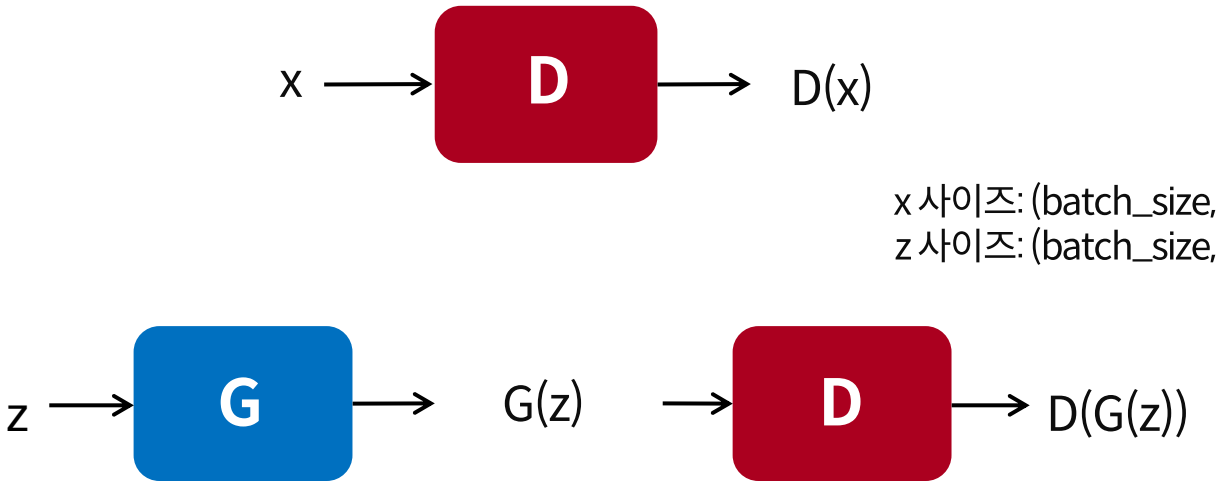
PyTorch Implementation



Optimizer for D and G

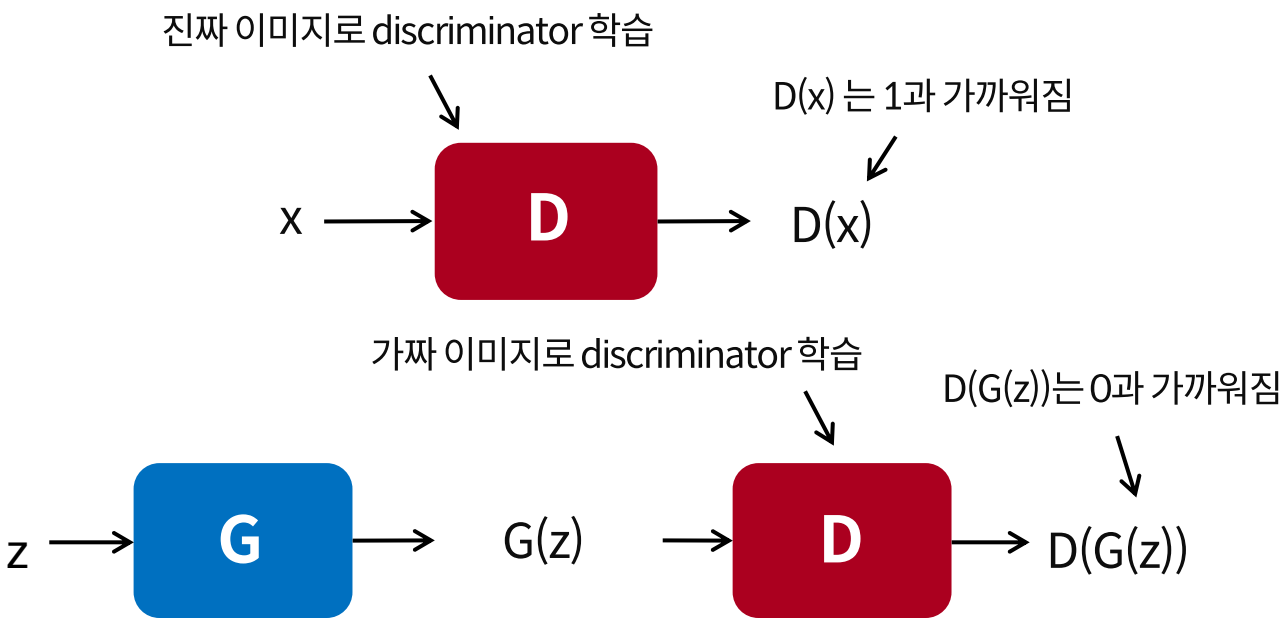
```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

PyTorch Implementation




```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

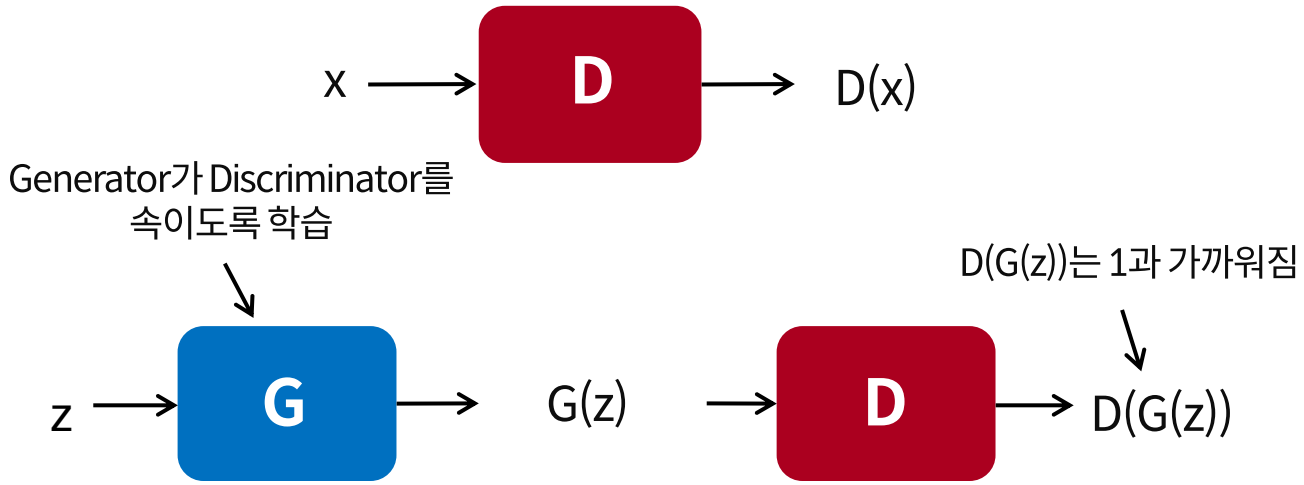
PyTorch Implementation



Forward, Backward and Gradient Descent

```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

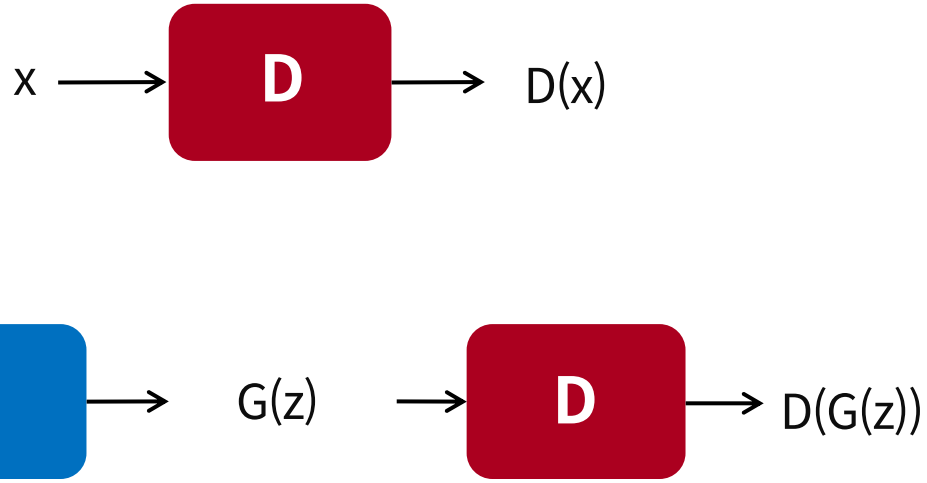
PyTorch Implementation



Forward, Backward and Gradient Descent →

```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

PyTorch Implementation

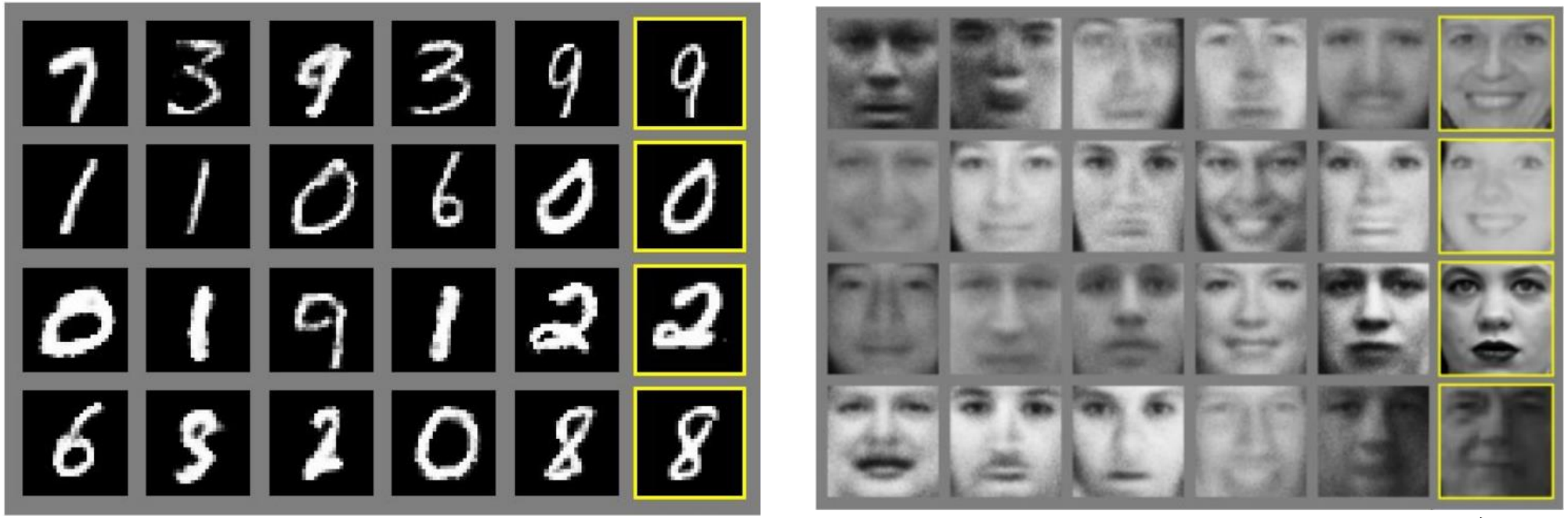


최종 코드는 아래 링크에서 확인 가능합니다.
https://github.com/yunjey/pytorch-tutorial

```python
import torch
import torch.nn as nn


D = nn.Sequential(
    nn.Linear(784, 128),
    nn.ReLU(),
    nn.Linear(128, 1),
    nn.Sigmoid())

G = nn.Sequential(
    nn.Linear(100, 128),
    nn.ReLU(),
    nn.Linear(128, 784),
    nn.Tanh())

criterion = nn.BCELoss()

d_optimizer = torch.optim.Adam(D.parameters(), lr=0.01)
g_optimizer = torch.optim.Adam(G.parameters(), lr=0.01)

# Assume x be real images of shape (batch_size, 784)
# Assume z be random noise of shape (batch_size, 100)

while True:
    # train D
    loss = criterion(D(x), 1) + criterion(D(G(z)), 0)
    loss.backward()
    d_optimizer.step()

    # train G
    loss = criterion(D(G(z)), 1)
    loss.backward()
    g_optimizer.step()
```

Generative Adversarial Nets

생성 예제

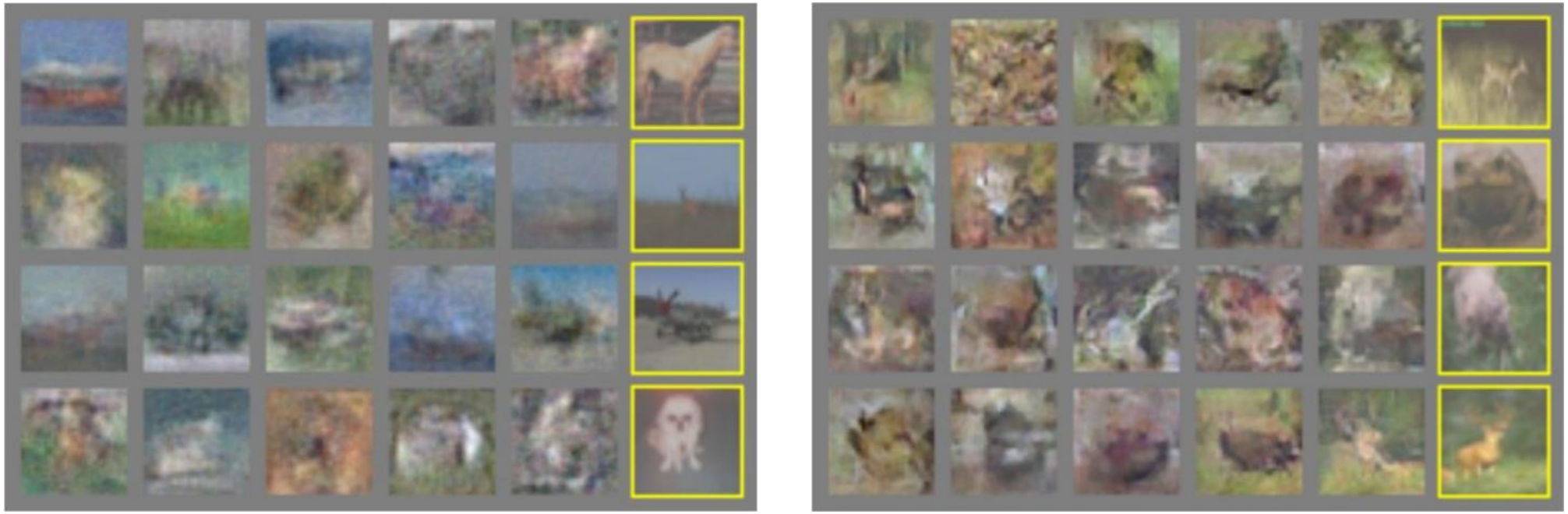


Nearest neighbor from training set

Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014
Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission

Generative Adversarial Nets

생성 예제 (CIFAR-10)



Nearest neighbor from training set

Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014
Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission

GAN 요약

“

explicit density function이 아닌, 게임 이론을 이용한 방법론을 쓰세요.
2-player game을 통해 학습 데이터의 분포에서 생성하는 방법을 배우도록 하세요.

장점
- 아주 뛰어난 품질의 데이터 샘플들을 보여줍니다!

단점
- 더 까다롭고 안정적이지 못한 학습 과정
- p(x), p(z|x) 같은 inference query들을 해결할 수 없음

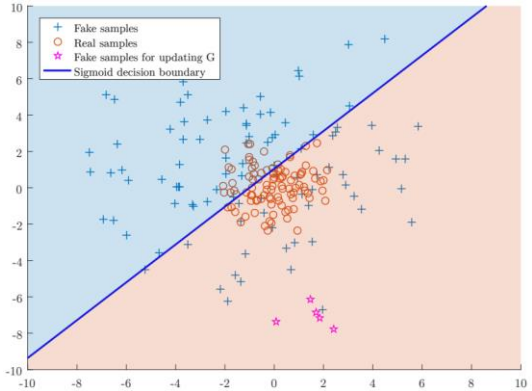
활발히 연구되고 있는 분야들
- 더 안정적인 학습, 더 좋은 목적 함수 (LSGAN, Wasserstein GAN 등)
- 다양한 조건을 수용할 수 있는 Conditional GANs 기법 및 이를 활용한 다양한 응용 사례

Ch 3. GAN 학습 안정화

Clip 1. GAN 모델 안정화를 위한 기법

LSGAN

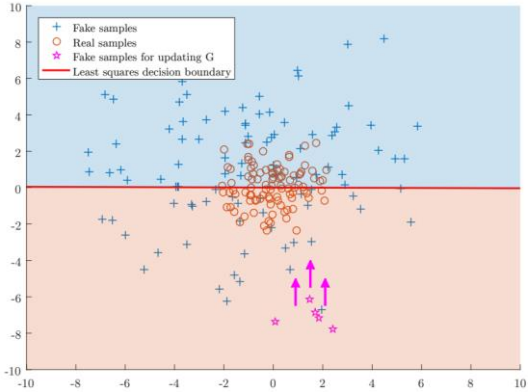
- Least Squares Generative Adversarial Networks (LSGANs)
- Discriminator 학습에 binary cross entropy loss 대신 least squares loss를 사용



binary cross entropy loss

$$\max_G \min_D L_{\text{GAN}}(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[-\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[-\log(1-D(G(z)))]$$

Vanishing gradients 문제:
여전히 진짜 이미지 같지 않아 보여도 그래디언트가 작음.



least squares loss

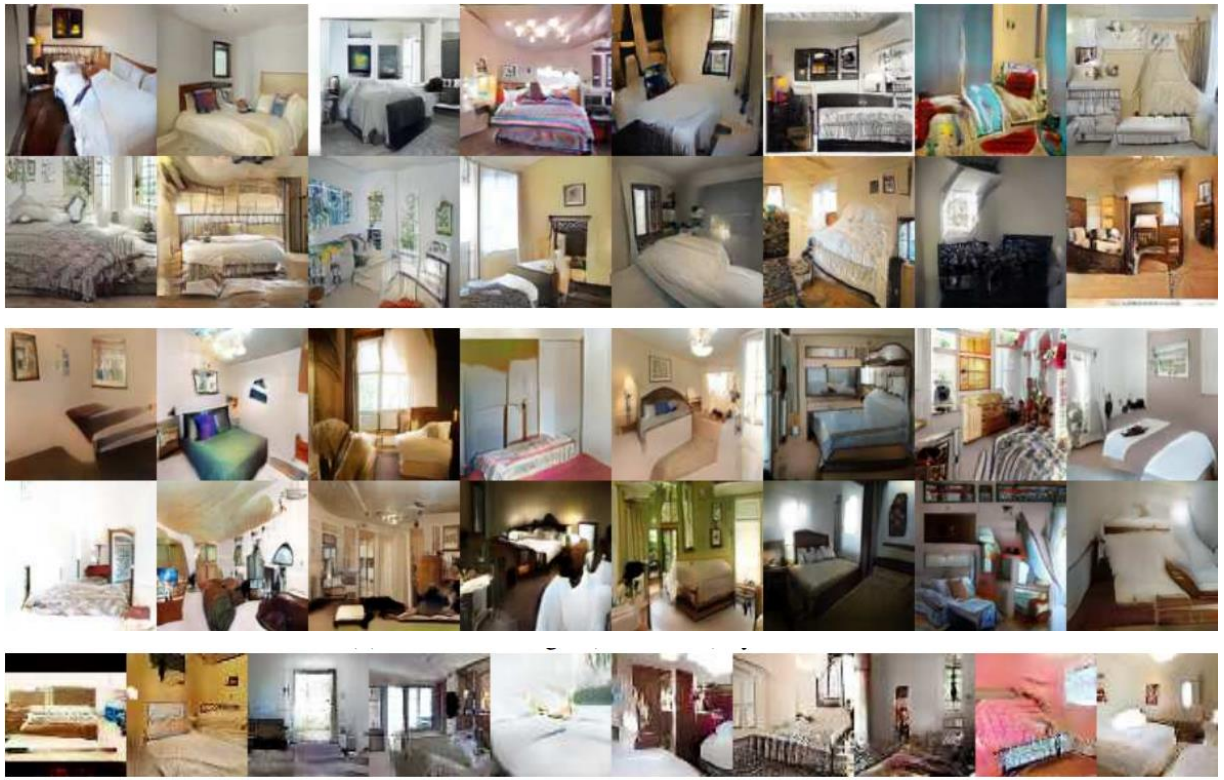
- $\min_D L_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}[(D(x)-b)^2] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}[(D(G(z))-a)^2]$
- $\min_G L_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[\left(D(G(z))\right)-c)^2\right]$ (예: $a=0, b=1, c=1$)

- 진짜 이미지라고 분류된 생성 데이터일지라도, 진짜 이미지들과 거리가 멀면 진짜 이미지들과 비슷한 예측값을 가지도록 학습됨.
- 생성 이미지들의 품질이 향상됨.
- Vanishing gradient 문제를 완화해서 더 안정적으로 학습 가능.

Mao et al. "Least squares generative adversarial networks.", ICCV 2017

LSGAN

LSGAN 모델은 기존의 GAN 보다 더 고품질의 이미지를 생성함



(a) Generated images (112×112) by LSGANs.

(b) Generated images (112×112) by DCGANs.

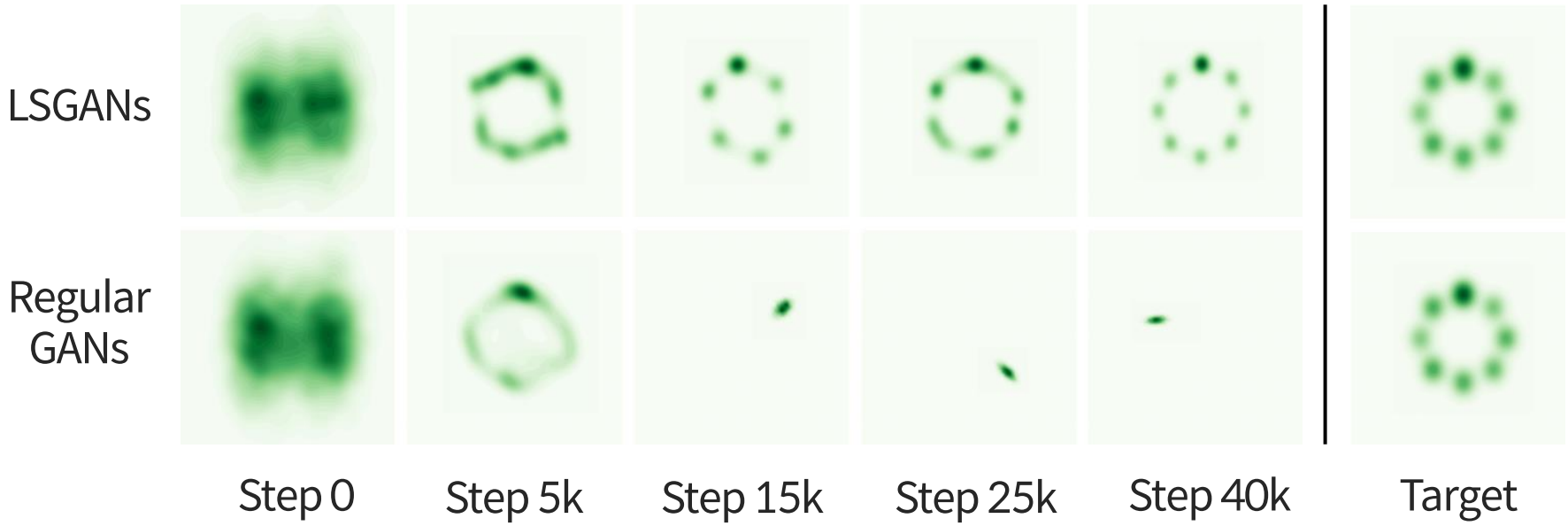
(b) Generated images(64×64) by DCGANs(reported in [25]).

Figure4. Generated images on LSUN-bedroom

Mao et al. "Least squares generative adversarial networks.“, ICCV 2017

LSGAN

LSGAN은 기존의 GAN보다 mode-collapsing이 없이 더 안정적인 학습이 가능함



Mao et al. "Least squares generative adversarial networks.“, ICCV 2017

Improved Wasserstein GAN (WGAN-GP)

Result

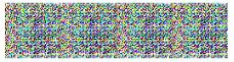
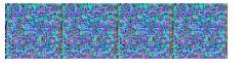
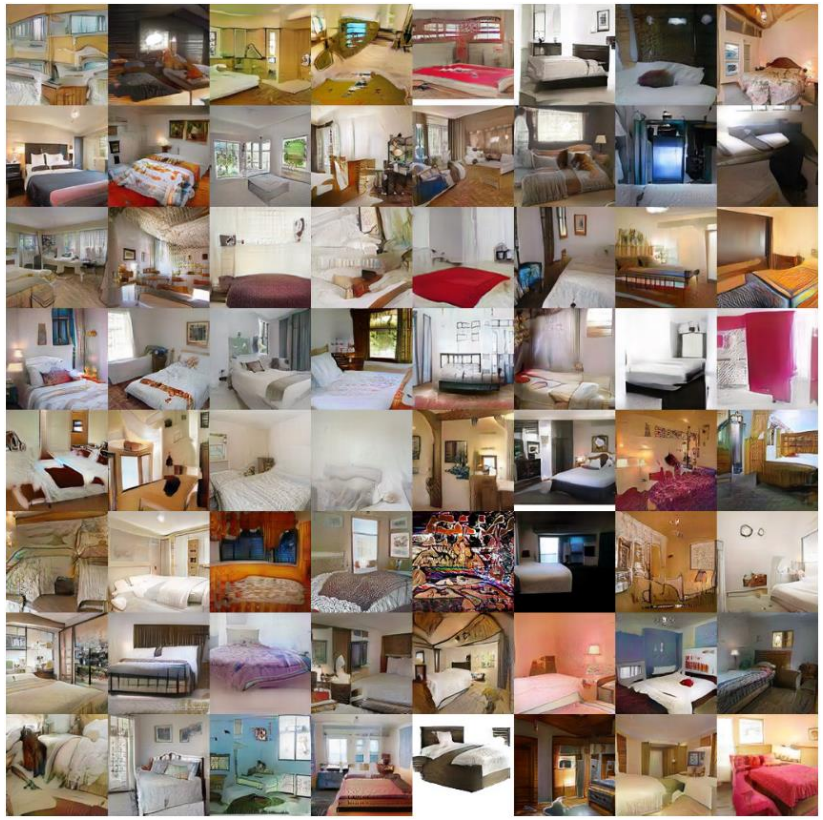
DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)				
G: No BN and a constant number of filters, D: DCGAN				
G: 4-layer 512-dim ReLU MLP, D: DCGAN				
No normalization in either G or D				
Gated multiplicative nonlinearities everywhere in G and D				
tanh nonlinearities everywhere in G and D				
101-layer ResNet G and D				

Figure 2: Different GAN architectures trained with different methods. We only succeeded in training every architecture with a shared set of hyperparameters using WGAN-GP.

128x128 LSUN samples



Gulrajani et al, Improved Training of Wasserstein GANs, NeurIPS 2017

Improved Wasserstein GAN (WGAN-GP)

Training algorithm

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

- 구현 예제 (공식 코드): https://github.com/igul222/improved_wgan_training
- 구현 예제 (PyTorch): https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/wgan_gp/wgan_gp.py

Gulrajani et al, Improved Training of Wasserstein GANs, NeurIPS 2017

GAN 학습안정화 기법 요약

“

2-player game을 통해 학습하는 GAN 모델은, 두 모델 중 어느 한 쪽이 과도하게 학습되고
다른 쪽은 전혀 학습이 되지 않는 등, 학습이 불안정해질 수 있습니다.

LSGAN, WGAN-GP 처럼 목적함수를 새로이 정의하거나 추가적인 regularization term을
원래 목적함수에 추가하여 학습함으로써 안정화를 꾀할 수 있습니다.

이 외에도, 다양한 이론적인 유도 과정 및 경험적인 실험 과정을 통해 제안된 많은 방법들이
존재합니다 (예시: Spectral Normalization)

02
Advanced GAN Models.

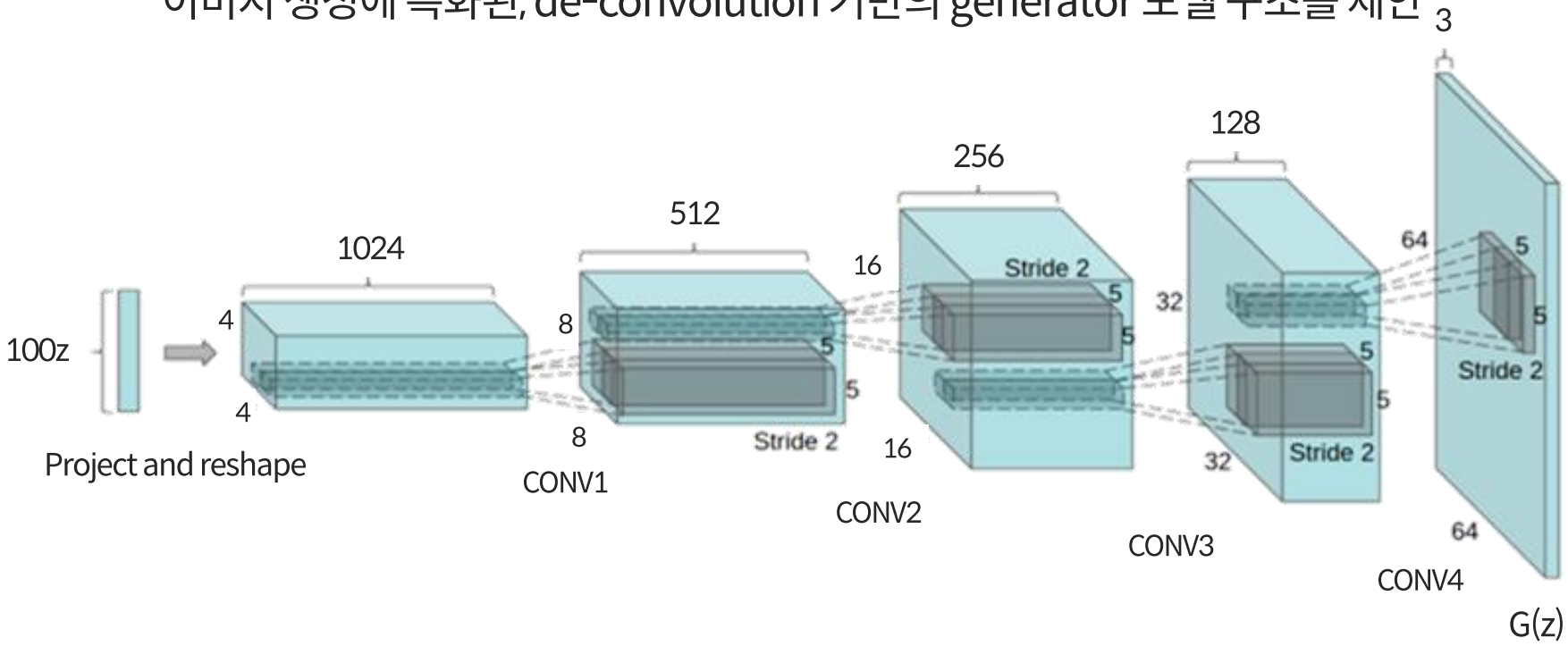
Ch 1. Advanced GAN Models

Clip 1. Deep Convolutional Generative Adversarial Networks (DCGAN)

DCGAN

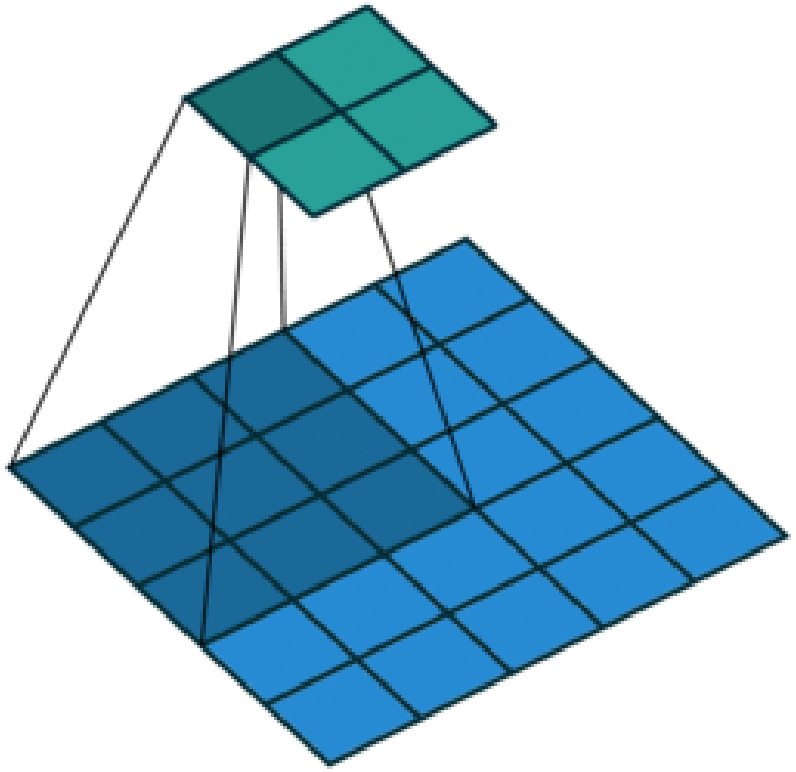
Deep Convolutional GAN (DCGAN), 2016

이미지 생성에 특화된, de-convolution 기반의 generator 모델 구조를 제안



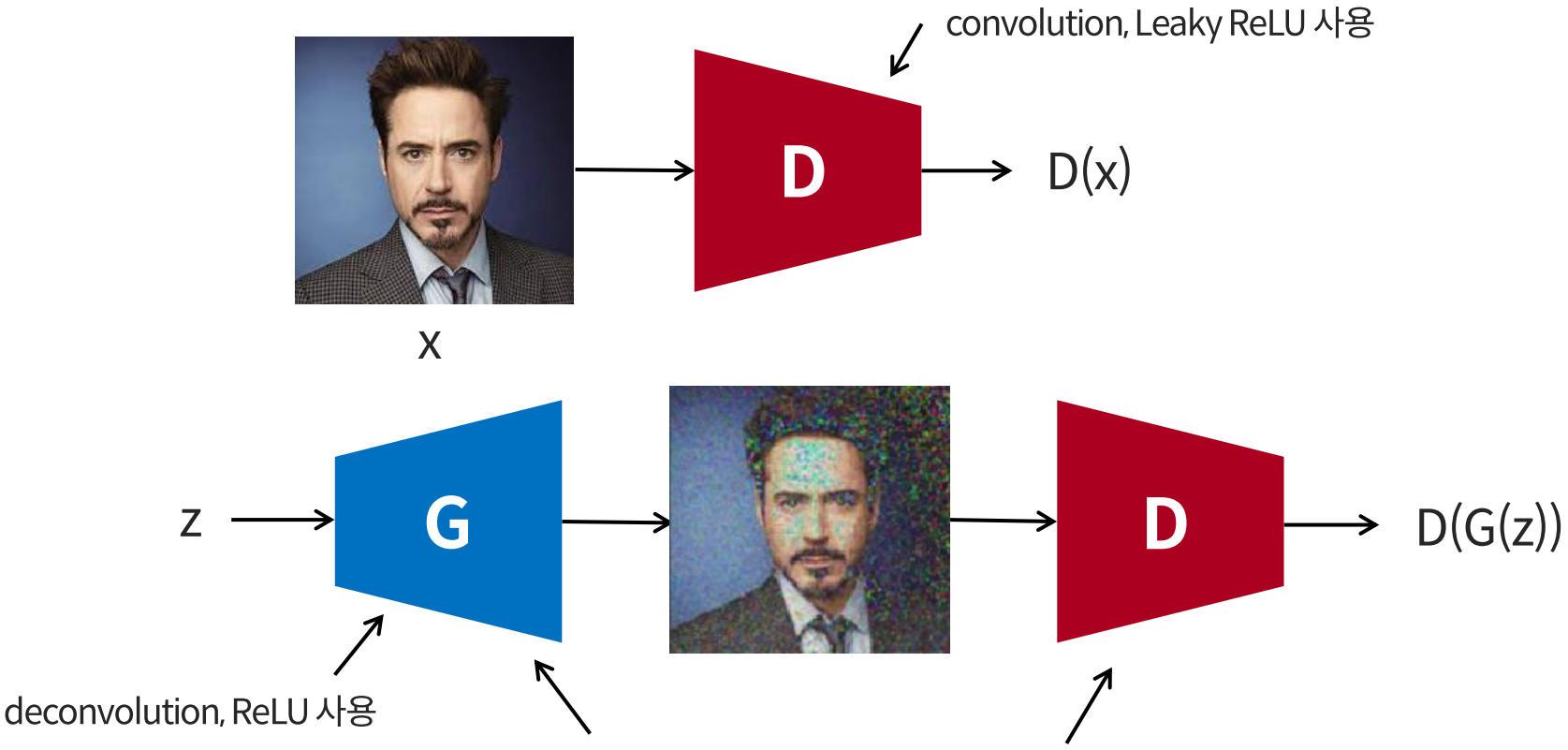
De-convolution (up-convolution, fractionally-strided convolution)

Convolution의 반대 과정을 수행하는 layer



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

DCGAN



- pooling layer 없음 (대신 strided convolution 사용)
- batch normalization 사용
- Adam optimizer(lr=0.0002, beta1=0.5, beta2=0.999)

Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

DCGAN

Generator는 upsampling network with fractionally-strided convolutions,
Discriminator는 convolutional network.

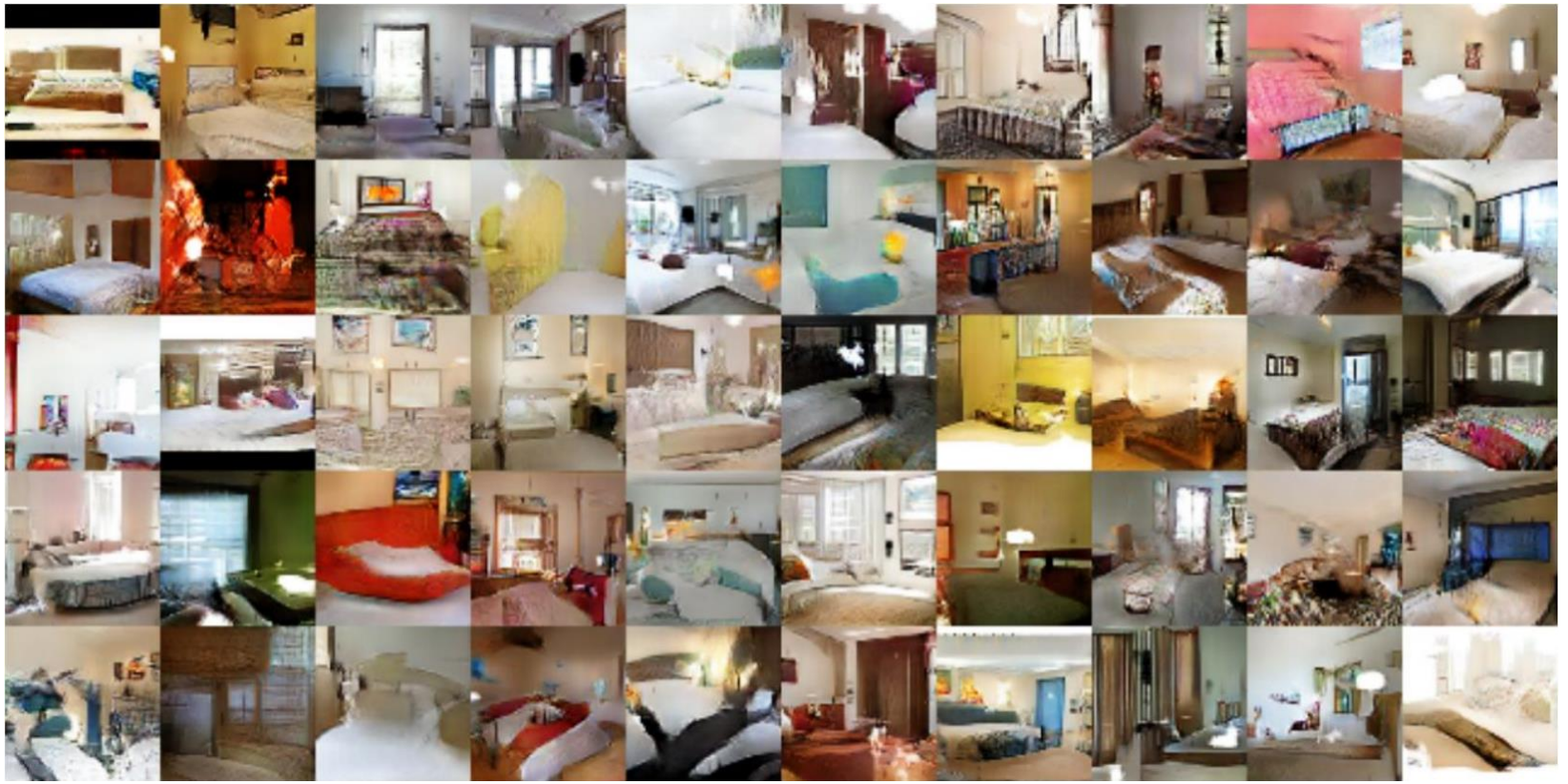
안정적인 Deep Convolutional GAN 학습을 위한
아키텍처 가이드라인

- 모든 pooling layers은 strided convolutions (Discriminator)와 fractional-strided convolutions (Generator)로 대체
- Generator와 Discriminator 모두 batch-norm 사용
- fully connected hidden layer를 제거
- Generator의 모든 layer에서 ReLU activation를 사용 (Tanh를 사용하는 출력단 제외)
- Discriminator의 모든 layer에서 Leaky ReLU activation 사용

Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

DCGAN

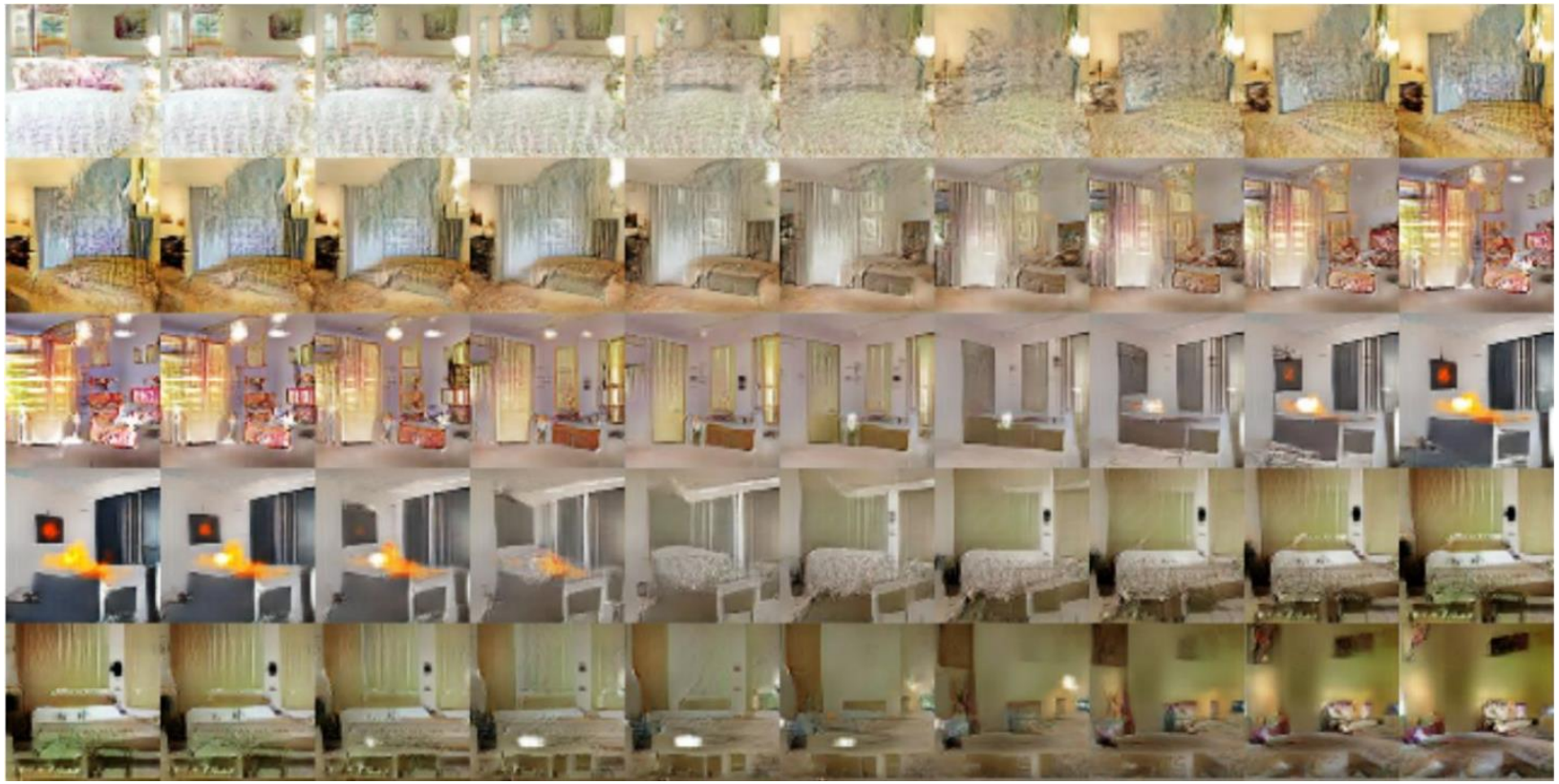
더 높은 품질의 생성 이미지를 결과로 냄



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

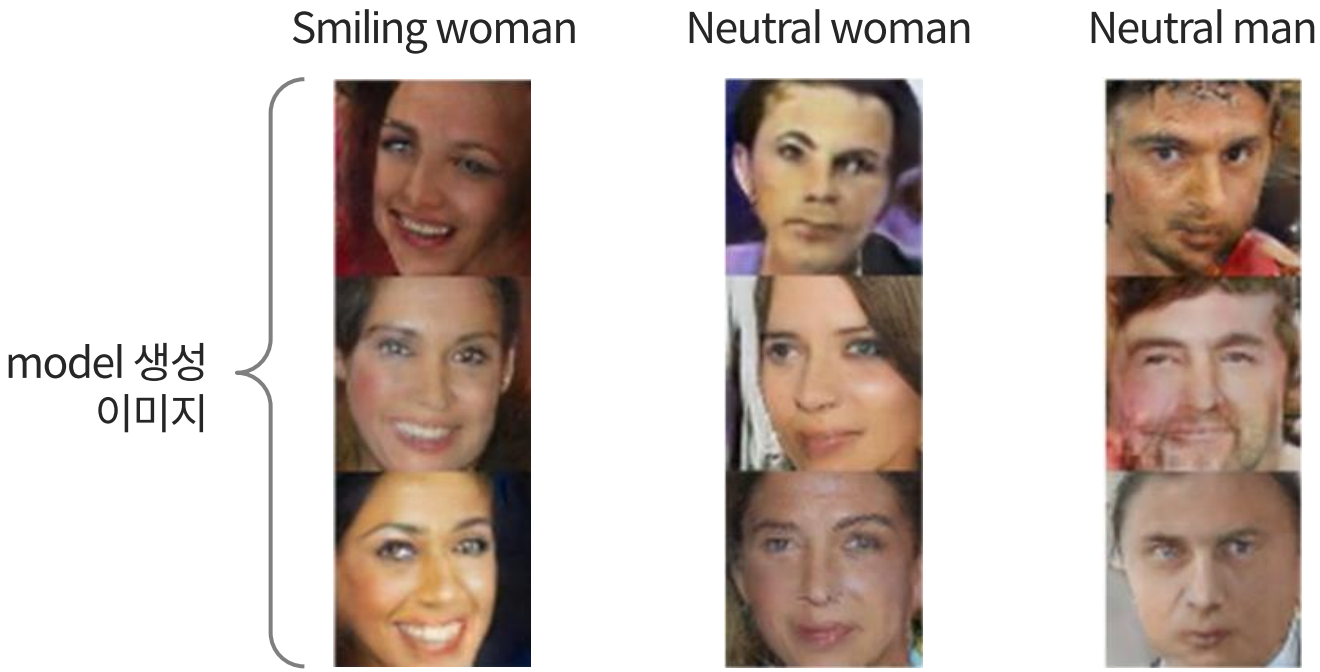
DCGAN

latent space 에서 random point들 간에 interpolation을 한 결과



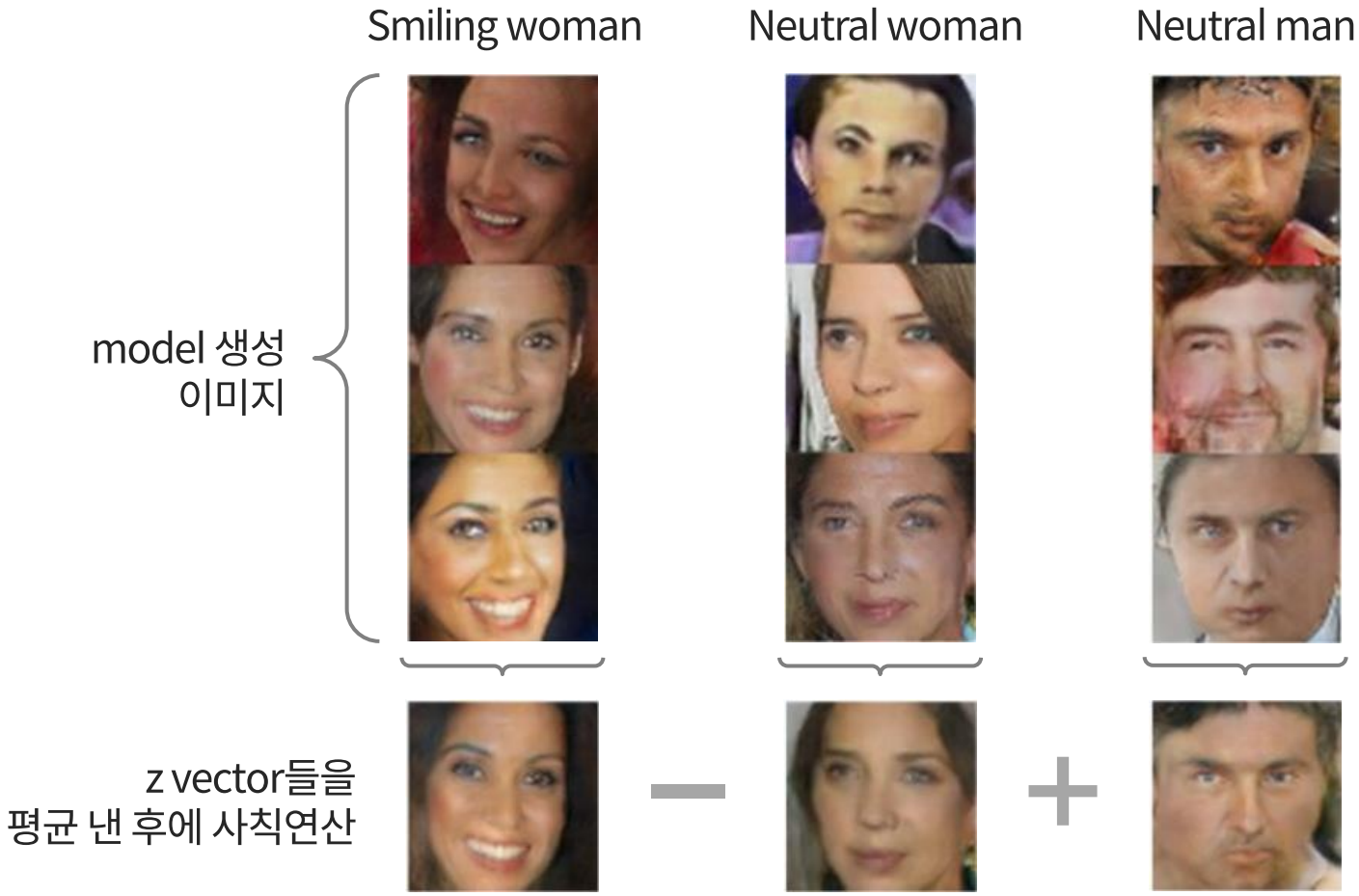
Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

DCGAN: Interpretable Vector Math



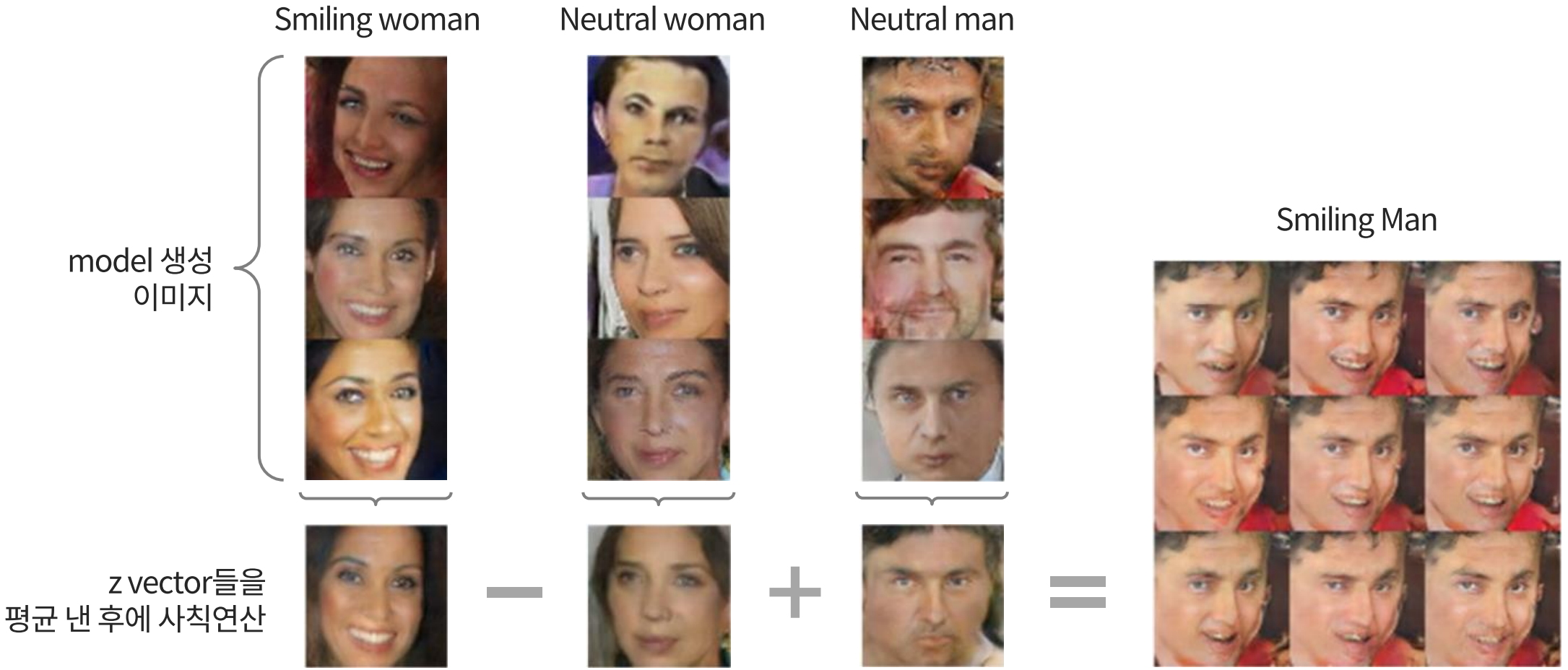
Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

DCGAN: Interpretable Vector Math



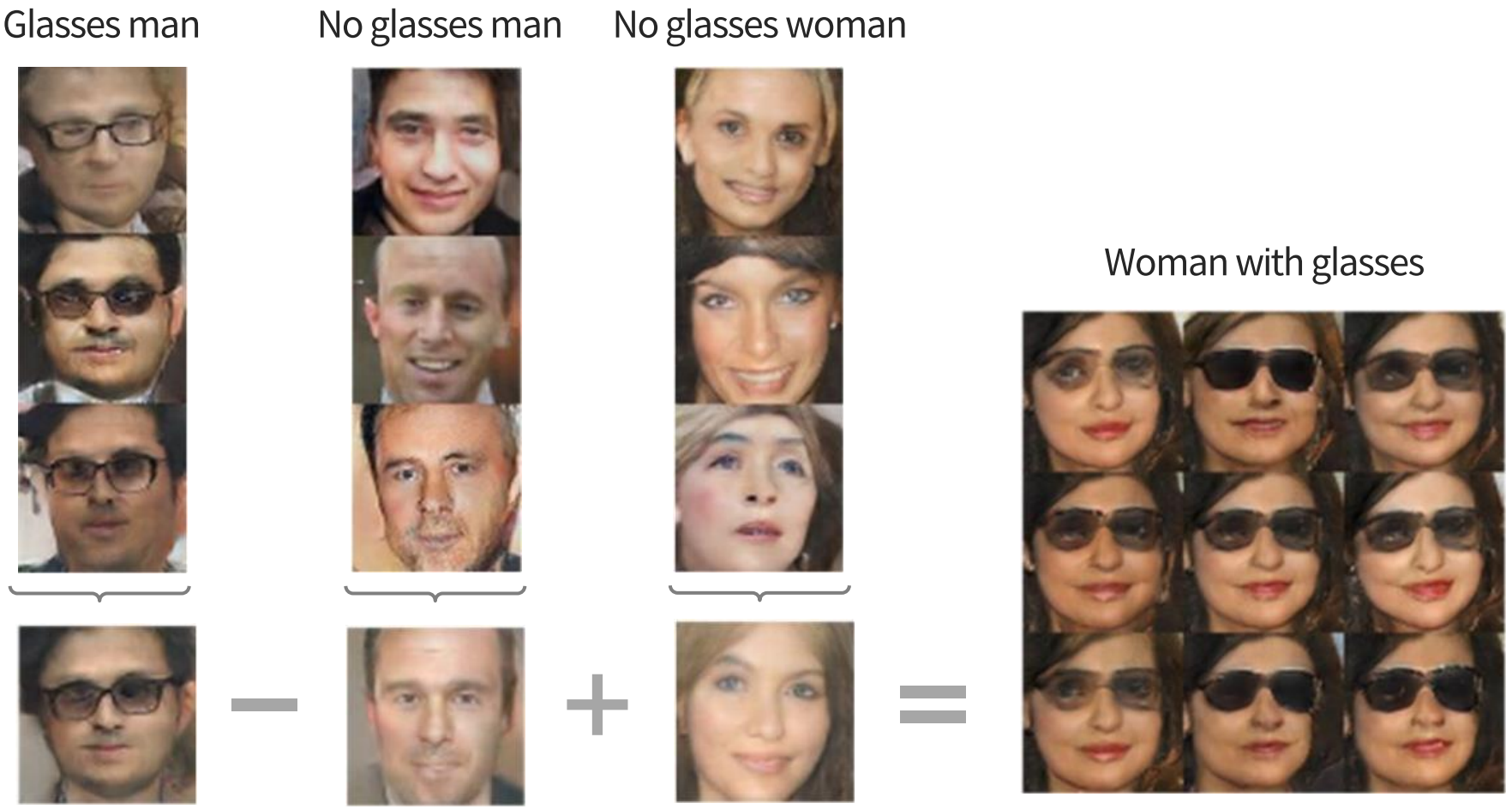
Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

DCGAN: Interpretable Vector Math



Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

DCGAN: Interpretable Vector Math



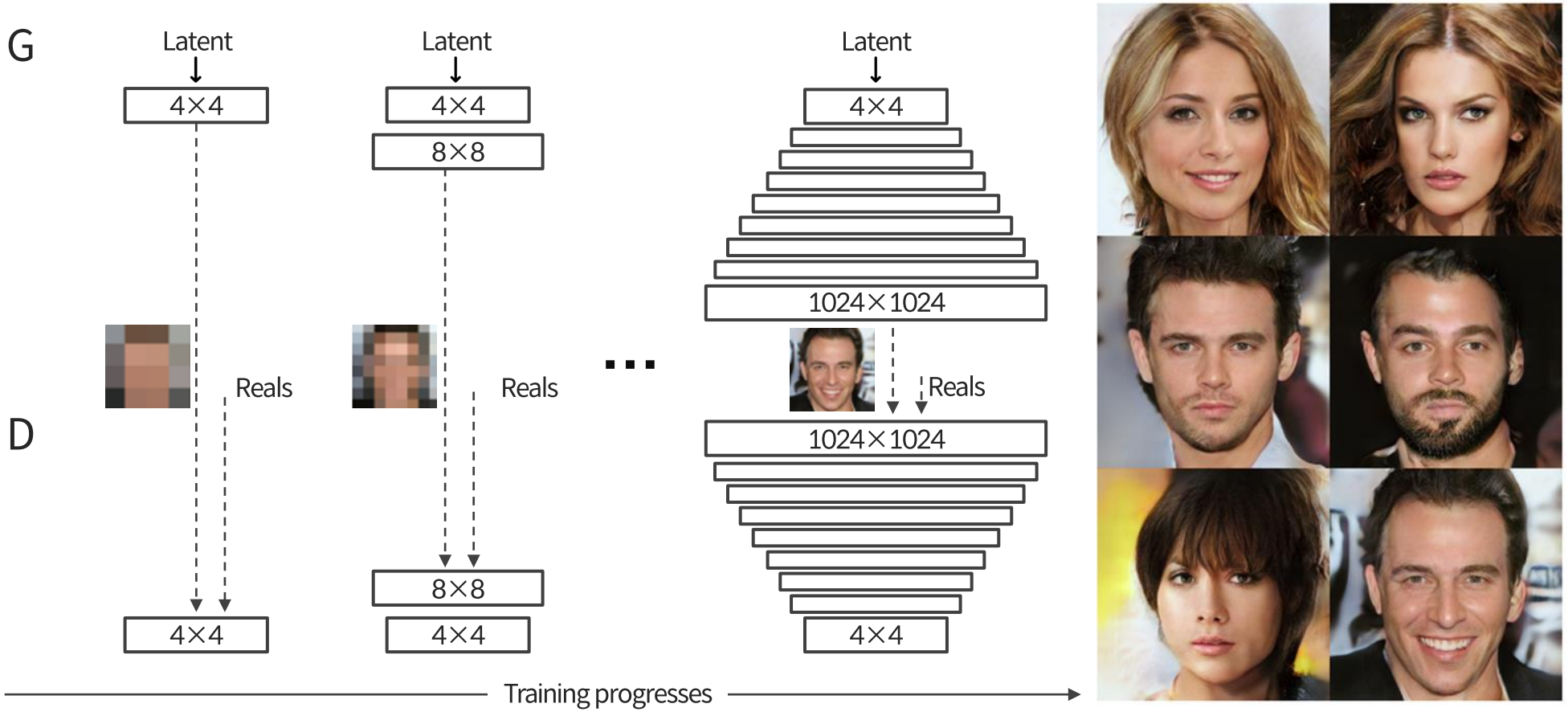
Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

Ch 1. Advanced GAN Models

Clip 2. Progressive Generative Adversarial Networks (PGGAN)

Progressive GAN

1024x1024 의 고해상도 이미지 생성을 가능하게 한 GAN 모델



Karras et al, Progressive Growing of GANs for Improved Quality, Stability, and Variation, ICLR 2018

Progressive GAN

생성된 고해상도 (1024x1024) 이미지 샘플

Karras et al, Progressive Growing of GANs for Improved Quality, Stability, and Variation, ICLR 2018

Ch 1. Advanced GAN Models

Clip 3. Style Transfer

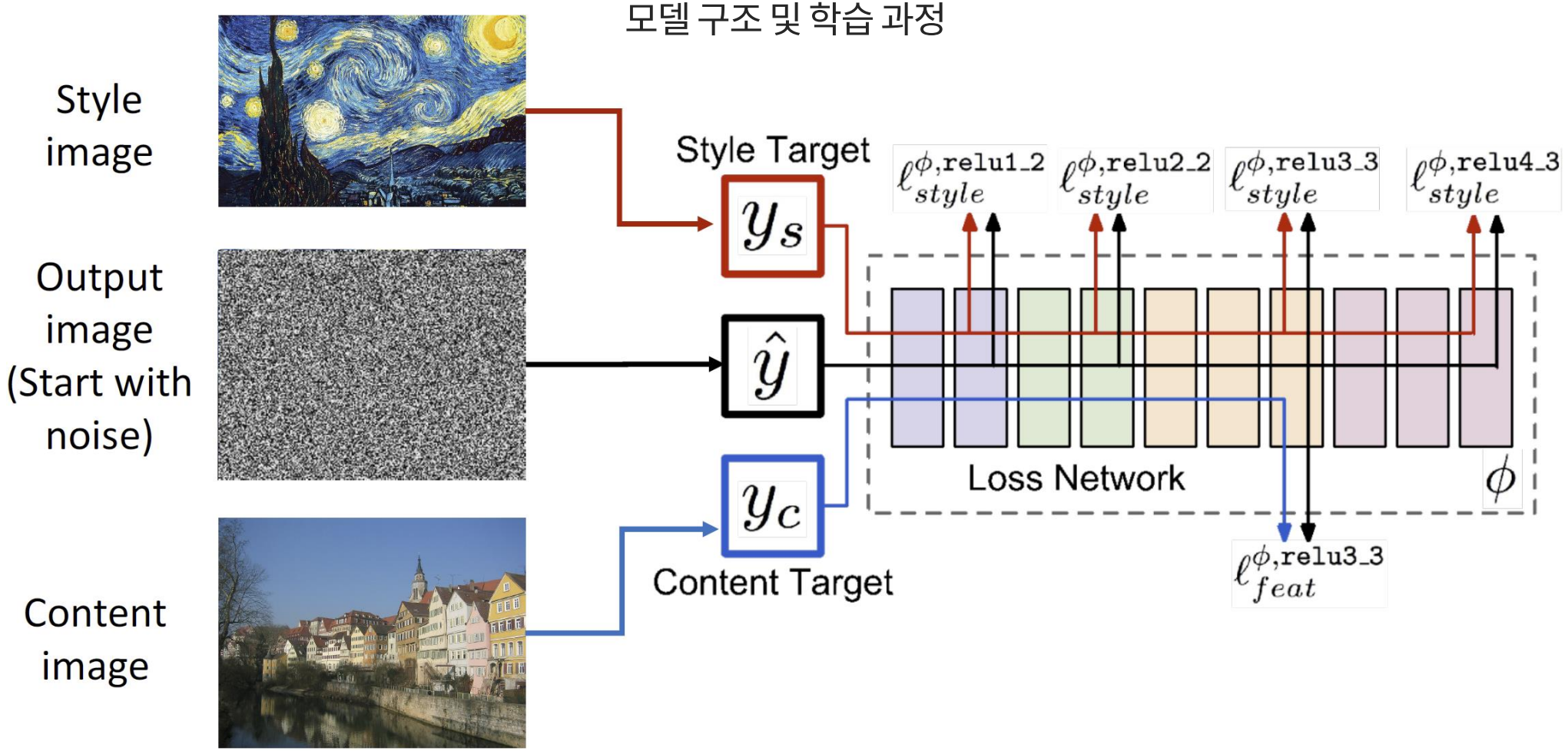
Style Transfer

주어진 이미지에 다른 이미지의 스타일을 주입하여 스타일 변환된 이미지를 합성하는 기술

Gatys et al, Image style transfer using convolutional neural networks, CVPR 2016

Style Transfer

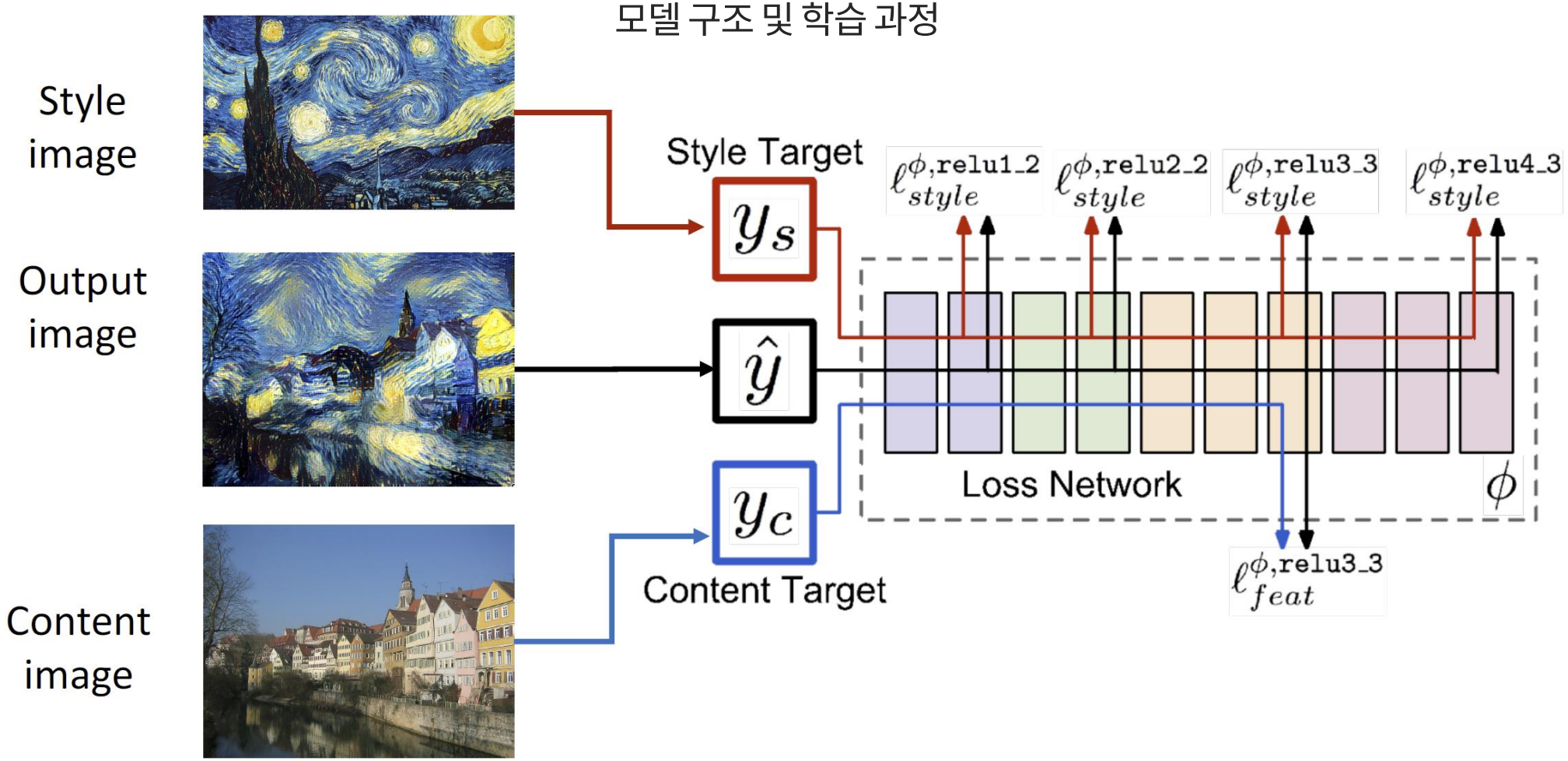
모델 구조 및 학습 과정



Gatys et al, Image style transfer using convolutional neural networks, CVPR 2016

Style Transfer

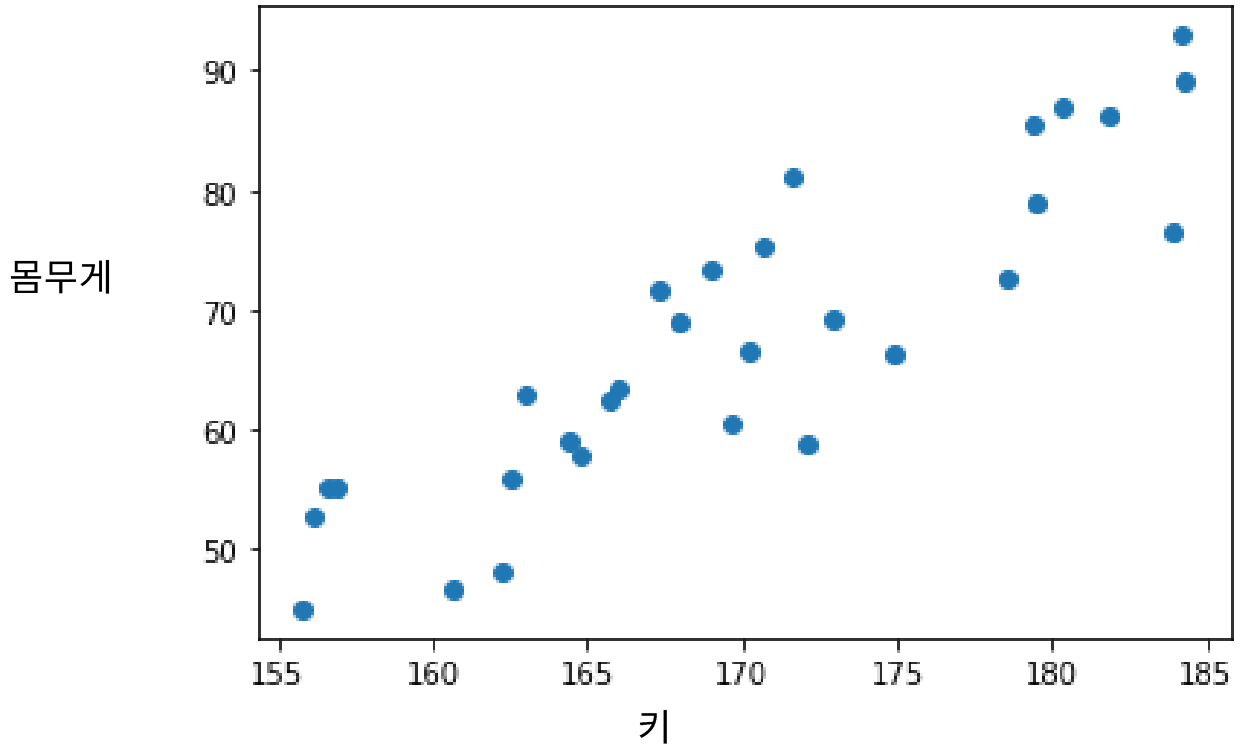
모델 구조 및 학습 과정

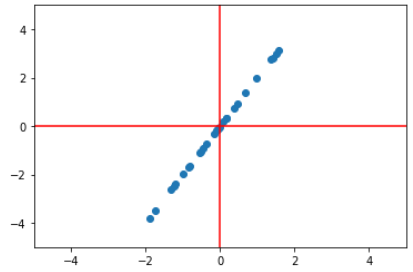


Gatys et al, Image style transfer using convolutional neural networks, CVPR 2016

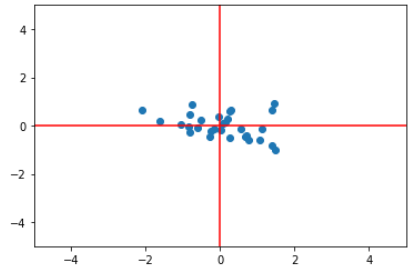
Feature Correlation

(키, 몸무게) 두개의 feature로 이루어진 사람들의 데이터를 총 30개 수집했다고 가정하자.
키와 몸무게는 일반적으로 관계성이 있는 것으로 알려져 있음.
이러한 두 개의 피처 간 관계성을 수치로 나타낸 것이 feature correlation 임.

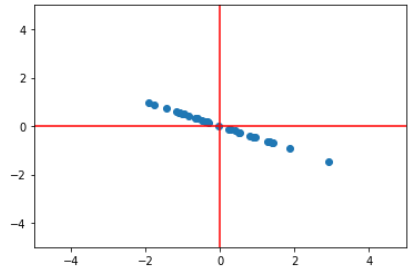




Correlation = 1



Correlation = 0

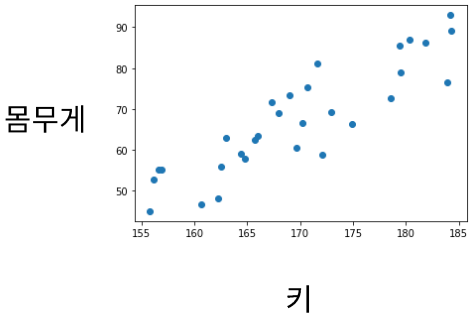


Correlation = -1

Feature Correlation

Correlation 을 구하는 과정

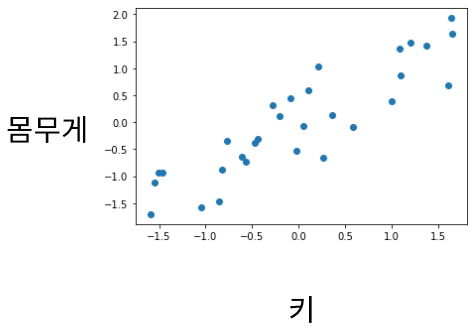
실제 데이터



키	몸무게
59	172
69	173
86	182
72	167
73	169
79	179

⋮

평균이 0, 분산이 1이 되도록 각 feature를 정규화



키	몸무게
-0.66	0.27
0.14	0.36
1.42	1.37
0.32	-0.28
0.44	-0.09
0.87	1.10

⋮

Correlation 계산

$$\text{Correlation} = \frac{\sum_{i=1}^{N} x_i y_i}{N}$$

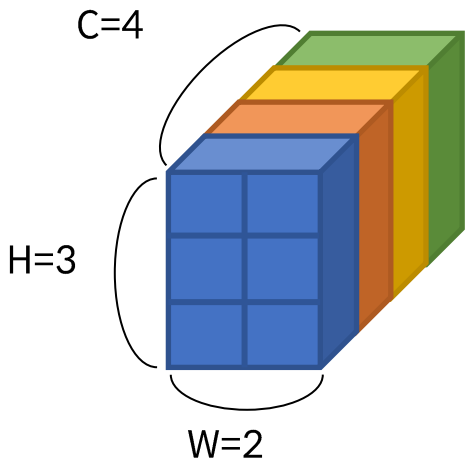
$$\frac{0.27 \times (-0.66) + 0.36 \times 0.14 + \cdots}{N} = 0.85$$

Gram matrix: Correlation의 변형된 형태

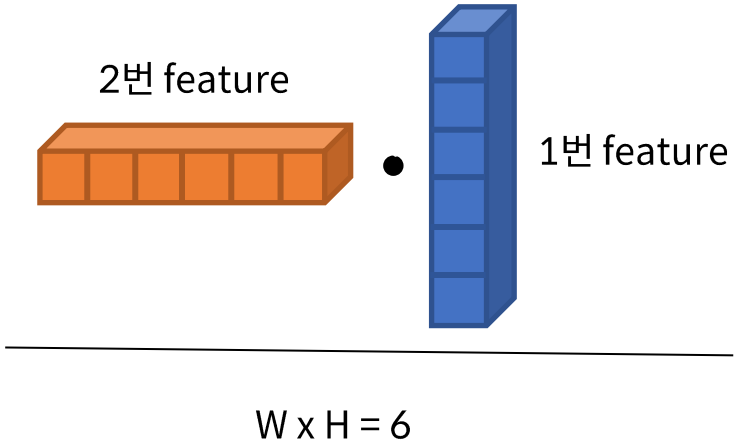
한 이미지의 스타일은 채널 간의 correlation 으로 나타낼 수 있고, 이 정보가 바로 Gram matrix를 통해 표현됨

계산 방법:
각 feature 간의 정규화를 생략하고, feature 쌍 간의 내적을 통해 바로 그 상관도를 계산함
모든 feature 상 간의 상관도를 모두 계산하여 이를 행렬로 표현함

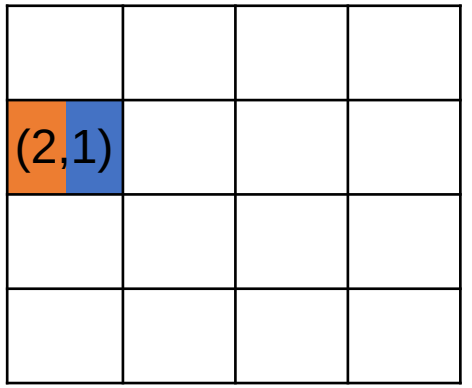
Image feature



Correlation 계산

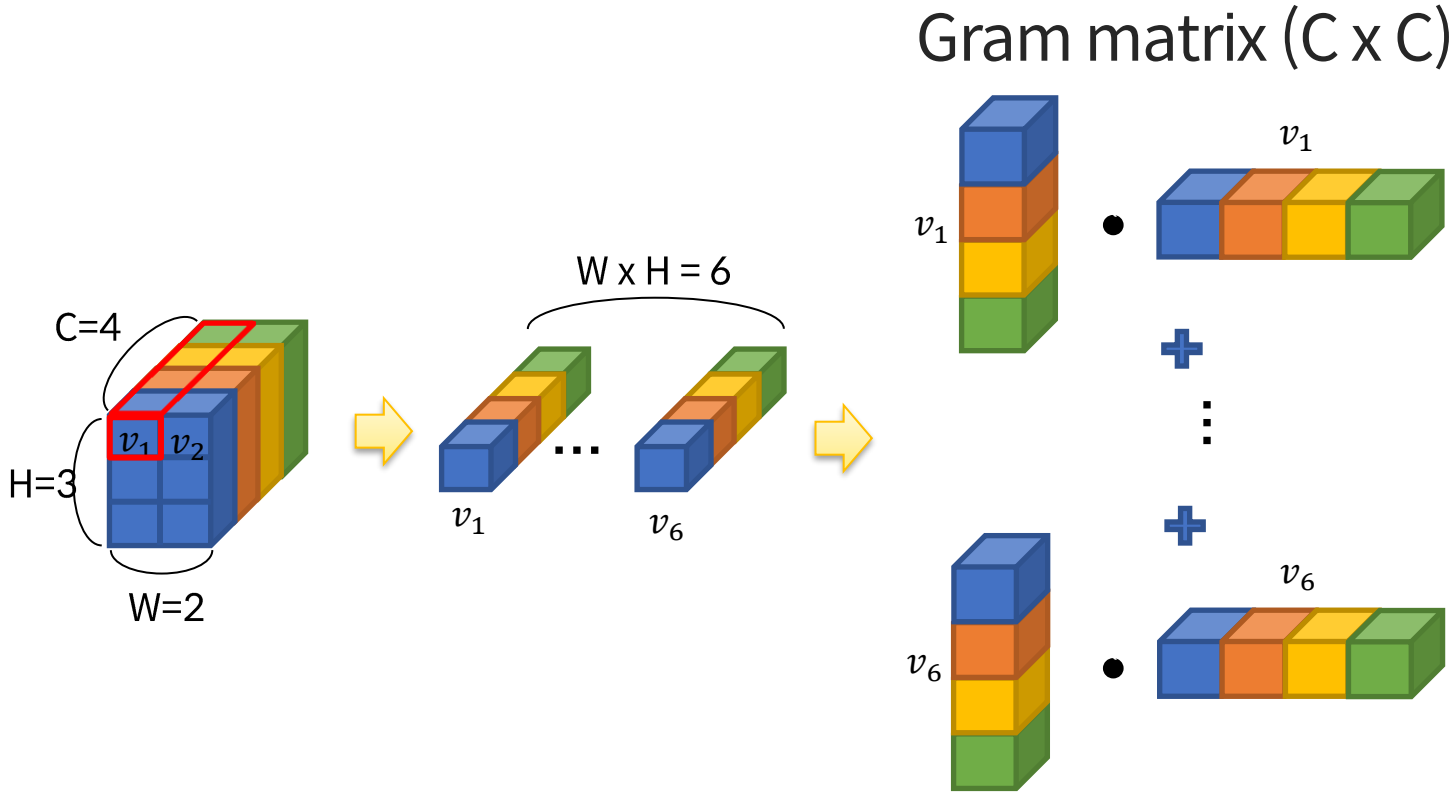


Gram matrix (C x C)



벡터 외적을 이용한 Gram matrix 계산

벡터 외적의 합으로 Gram matrix를 계산할 수 있음



벡터의 외적 및 외적의 합을 통한 행렬곱

벡터의 외적

$$\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ -1 & -2 & -3 \\ 1 & 2 & 3 \end{bmatrix}$$

임의의 행렬곱은 벡터의 외적의 합을 통해 나타낼 수 있음

$$\begin{bmatrix} 1 & 1 \\ -1 & 7 \\ 1 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 7 \\ 8 \end{bmatrix} \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

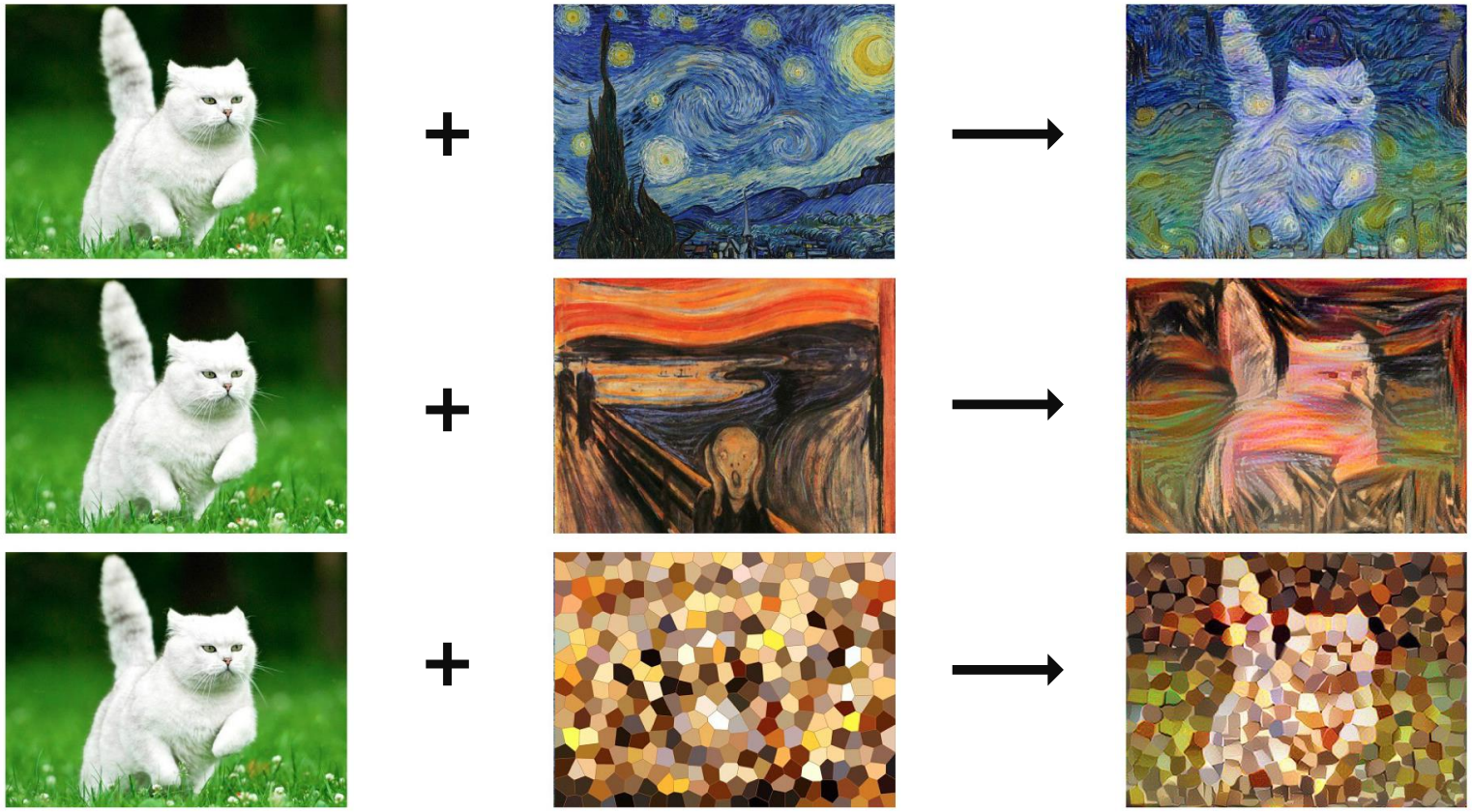
$$= \begin{bmatrix} 1 & 2 & 3 \\ -1 & -2 & -3 \\ 1 & 2 & 3 \end{bmatrix} + \begin{bmatrix} 4 & 5 & 6 \\ 28 & 35 & 42 \\ 32 & 40 & 48 \end{bmatrix}$$

Style Transfer

Content image

Style image

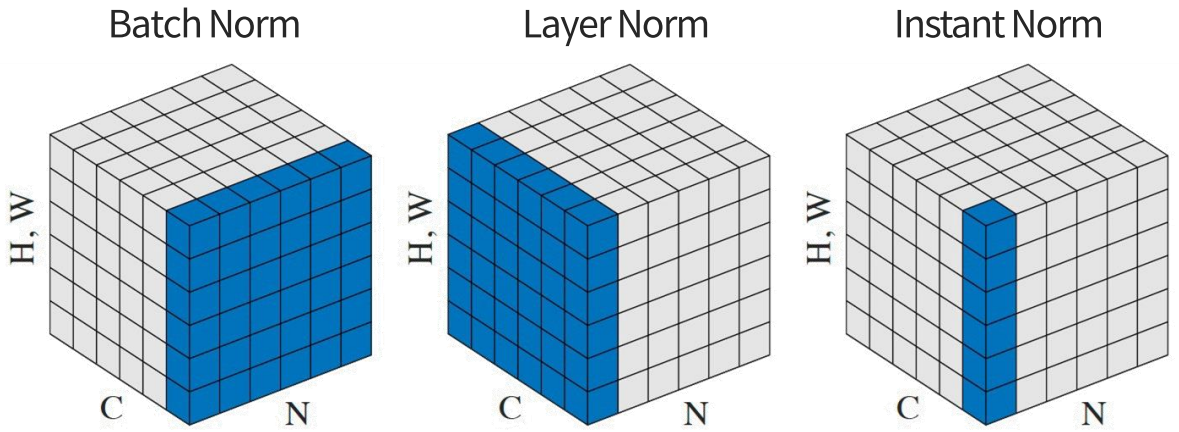
Output image



Ch 1. Advanced GAN Models

Clip 4. Adaptive Instance Normalization (AdaIN)

AdaIN



$$BN(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta$$

$$\mu_c(x) = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw}$$

$$\sigma_c(x) = \sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}\left(x_{nchw}-\mu_c(x)\right)^2}$$

$$IN(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta$$

$$\mu_{nc}(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw}$$

$$\sigma_{nc}(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}\left(x_{nchw}-\mu_{nc}(x)\right)^2}$$

Huang et al., Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization , ICCV'17

AdaIN

$$\text{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

- Adaptive IN (AdaIN) 연산을 style transfer 모델에 적용
- AdaIN은 content image의 각 채널 별 feature map의 평균과 분산을 style image의 각 채널 별 feature map의 평균과 분산으로 변환해 줌

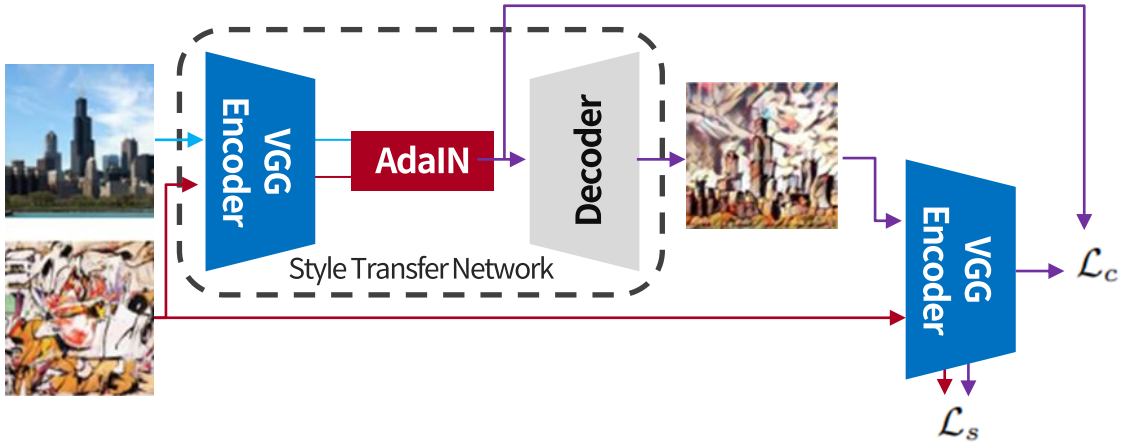


Figure 2. An overview of our style transfer algorithm. We use the first layers of a fixed VGG-19 network to encode the content and style images. An AdaIN later is used to perform style transfer in the feature space. A decoder is learned to invert the AdaIN output to the image spaces. We use the same VGG encoder to compute a content loss $\mathcal{L}_c$ (Equ. 12) and a style loss $\mathcal{L}_s$ (Equ. 13).

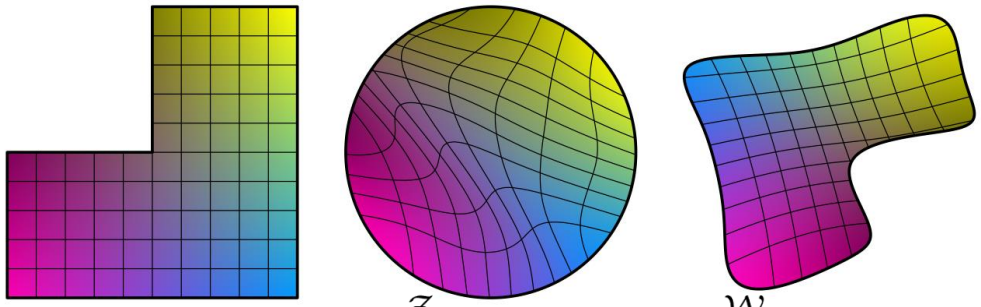
Huang et al., Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization , ICCV'17

Ch 1. Advanced GAN Models

Clip 5. Style GAN

StyleGAN

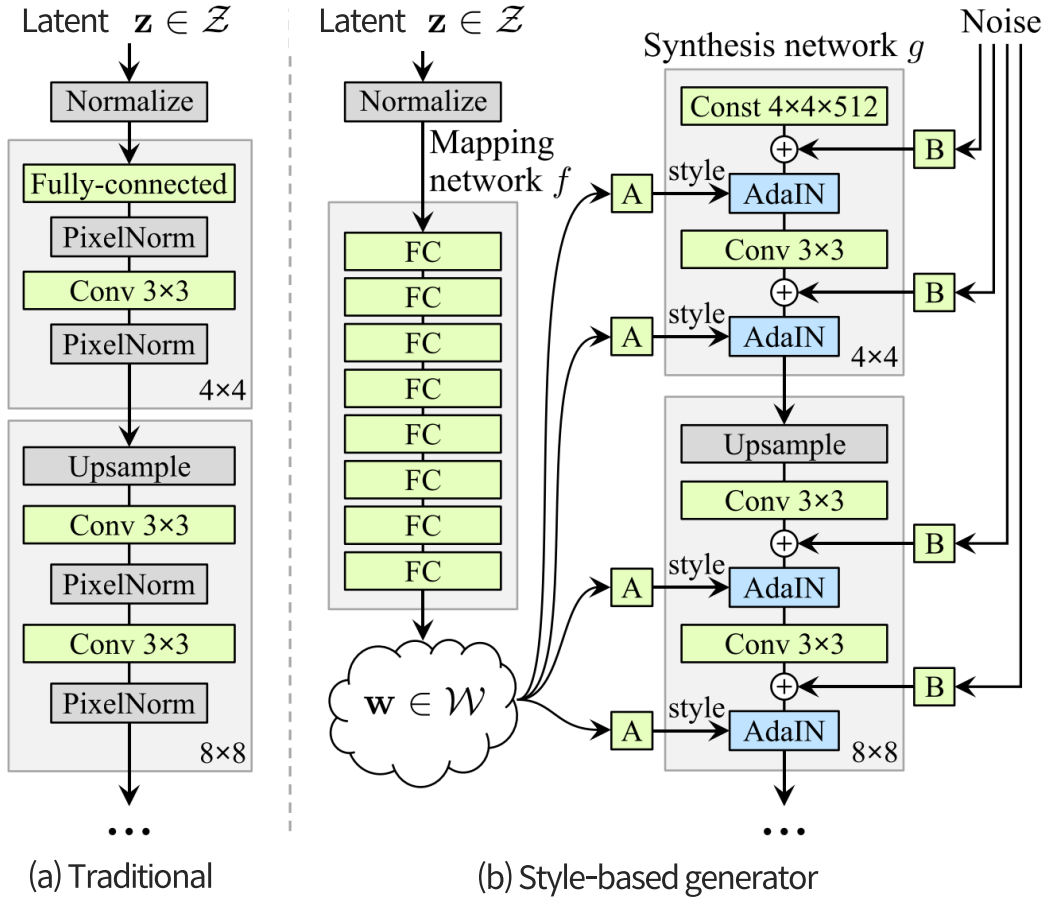
- Adaptive IN 연산을 generator 모델에 적용
- 표준 정규분포에서 구한 z 벡터를 mapping network을 통해 변환하고 (이를 w 벡터라 부름), 이를 generator 모델의 AdapIN 모듈의 입력으로 줌으로써, 보다 유연하게 학습된 w를 통해 실제 데이터의 분보를 보다 잘 모사하여 이미지 생성 품질을 고도화함



(a) Distribution of Features in training set

(b) Mapping from $\mathcal{Z}$ to features

(b) Mapping from $\mathcal{W}$ to features

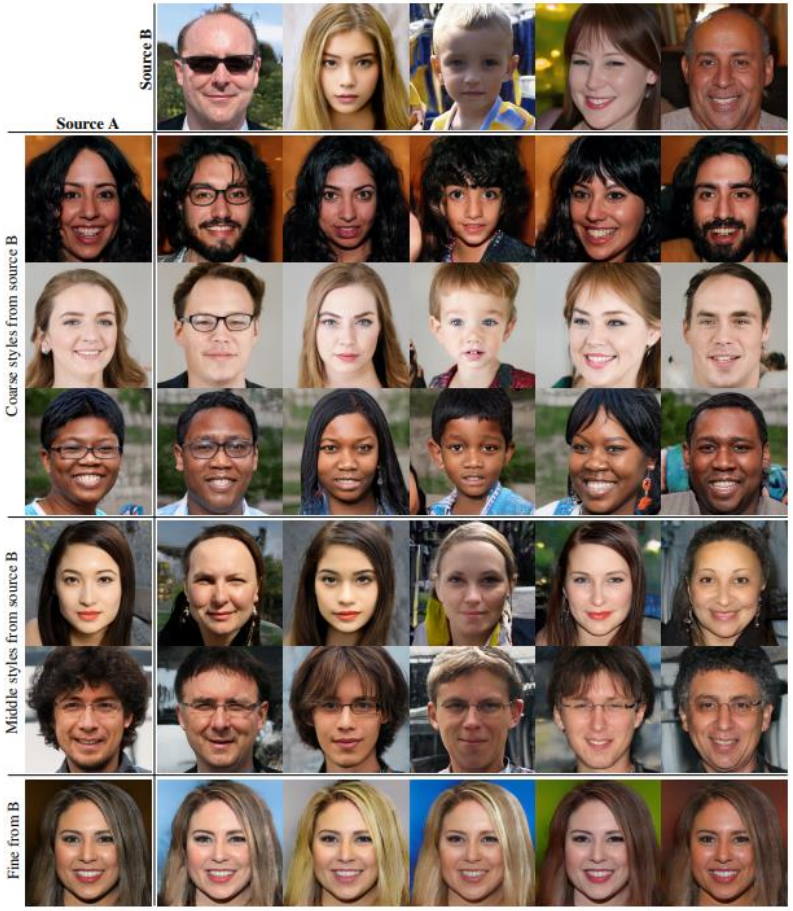


(a) Traditional

(b) Style-based generator

Karras et al, A Style-Based Generator Architecture for Generative Adversarial Networks, CVPR'19

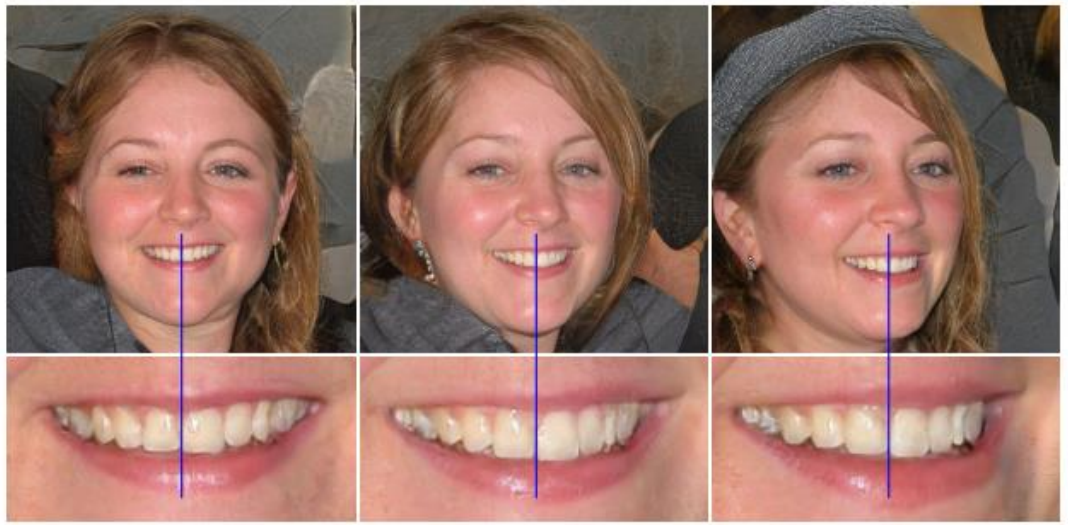
StyleGAN



Karras et al, A Style-Based Generator Architecture for Generative Adversarial Networks, CVPR'19

Problem of StyleGAN

Droplet-like Artifacts



Phase Artifacts

Karras et al, Analyzing and Improving the Image Quality of StyleGAN. CVPR 2020

StyleGAN2

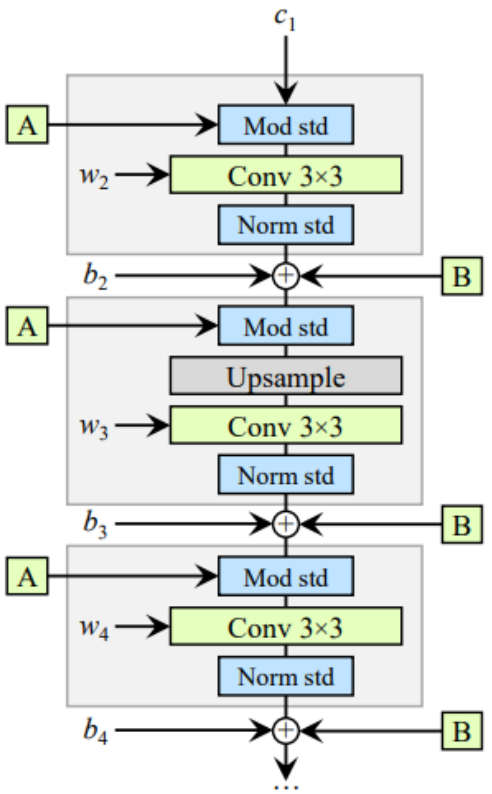
- Convolution filter 가중치 w를 re-scaling하는 것으로 AdaIN을 대체함.
- w'는 modulated된 weights를 나타냄.
- i 는 input channel, j는 output channel, k는 convolution filter의 spatial index를 나타냄

[Modulation]

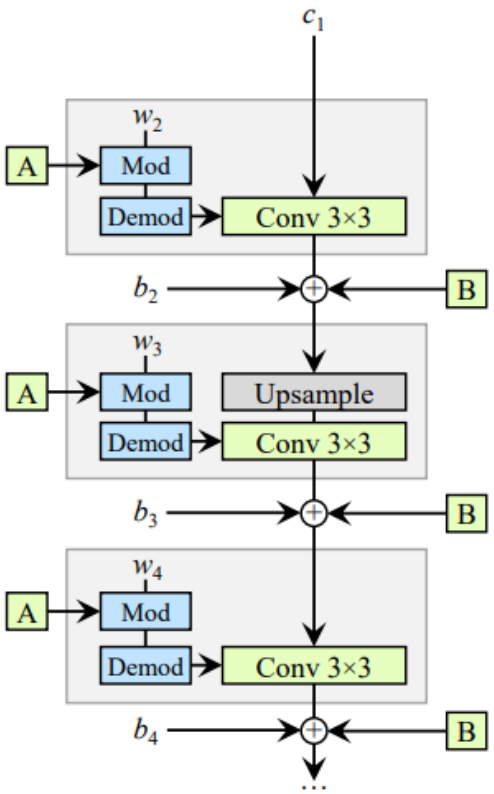
$$w'_{ijk} = s_i \cdot w_{ijk}$$

[Demodulation]

$$w''_{ijk} = w'_{ijk} / \sqrt{\sum_{i,k} w'^{2}_{ijk} + \epsilon}$$



(c) Revised architecture



(d) Weight demodulation

Karras et al, Analyzing and Improving the Image Quality of StyleGAN. CVPR 2020

StyleGAN2

Demo:
https://thispersondoesnotexist.com

Advanced GAN 모델 요약

“

고해상도, 고품질의 이미지를 얻기 위한 다양한 GAN 모델이 나오고 있습니다.

고품질의 이미지의 생성을 위해, 이미지의 스타일 변환을 담당하는 AdaIN 기법이
핵심 모듈로 사용되어 StyleGAN, StyleGAN2 등의 모델들이 활발히 사용되고 있습니다.

03
Image-to-Image Translation

Ch 1. Conditional GANs

Clip 1. Architecture of Conditional GANs

Image-to-Image Translation

Translation Tasks

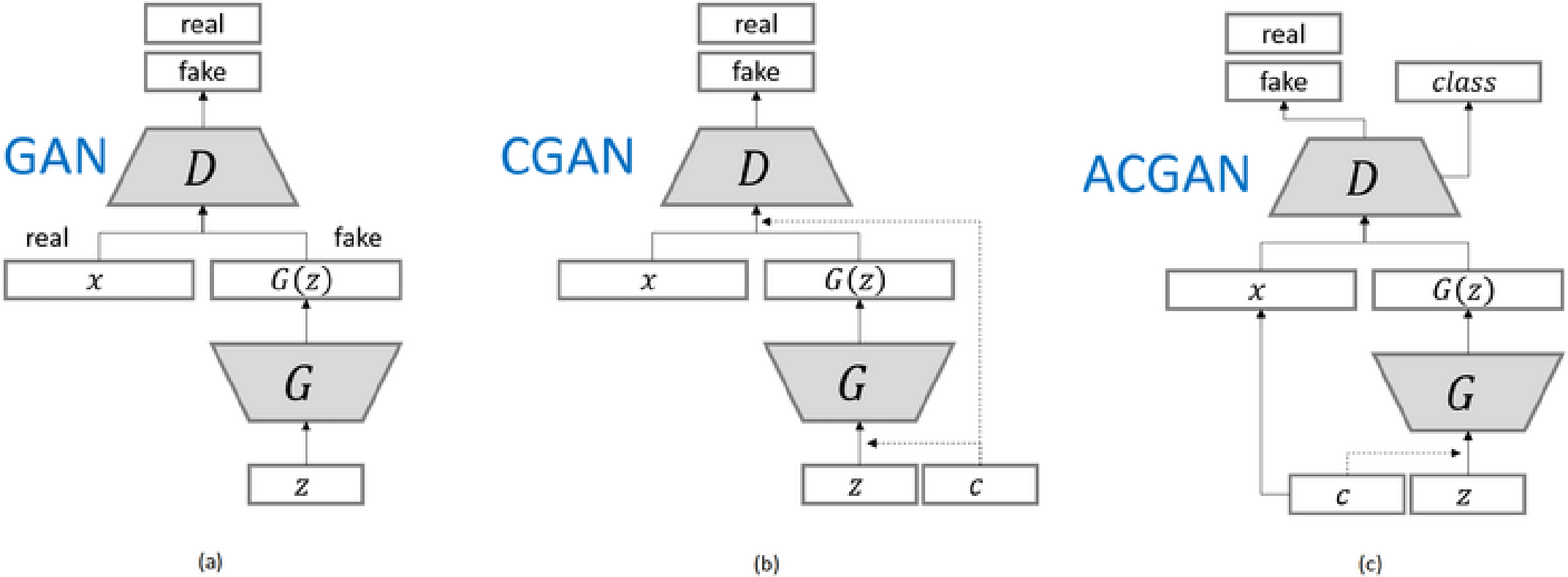
Image
(64x64x3)

Translation
Model

Image
(64x64x3)

조건부 생성 및 변환

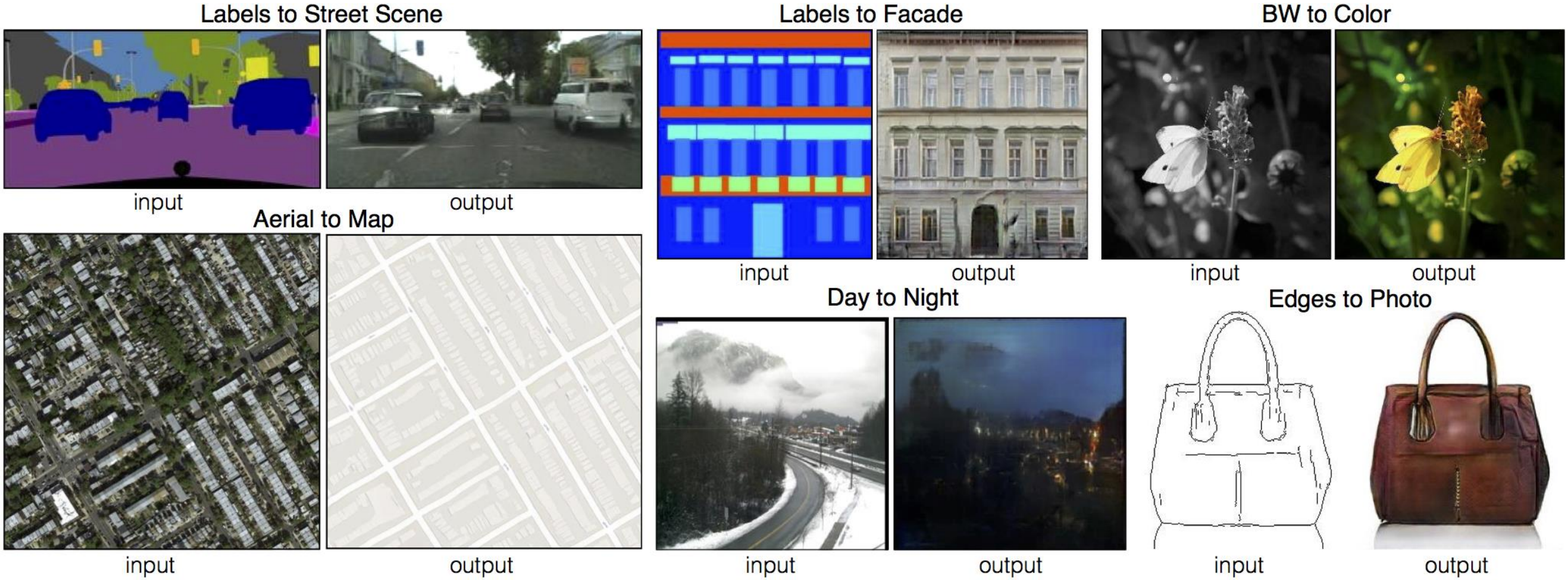
- 사용자가 추가적으로 주는 정보는 이미지 생성 및 변환할 때 해당 조건을 만족해야 하는 조건으로 활용됨
- 조건부 생성 및 변환에는 크게 두가지 방법이 있음: CGAN and ACGAN



Ch 2. Image-to-Image Translation Models

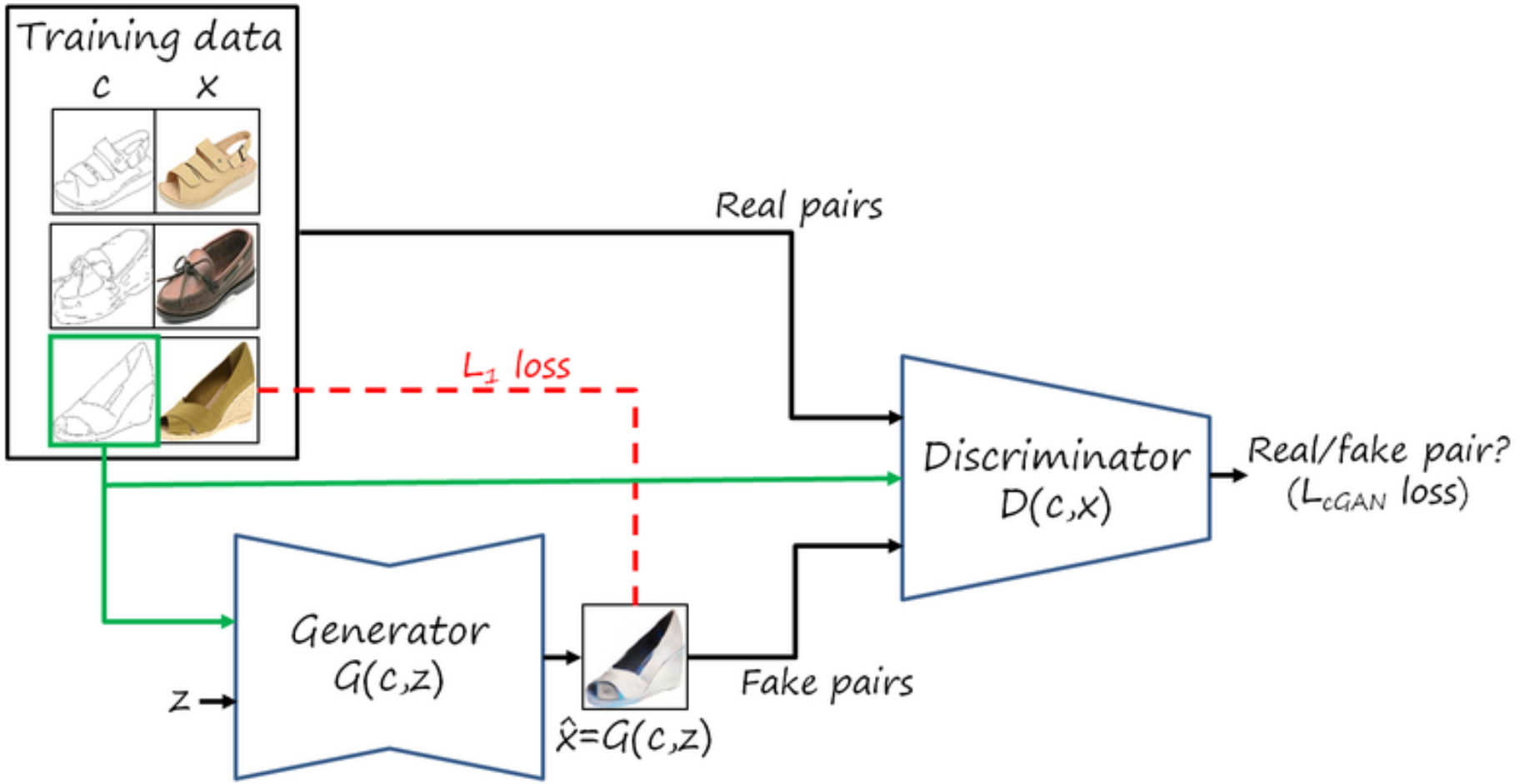
Clip 1. pix2pix

pix2pix: Paired Image-to-Image Translation



Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR'17

pix2pix: CGAN-based Architecture



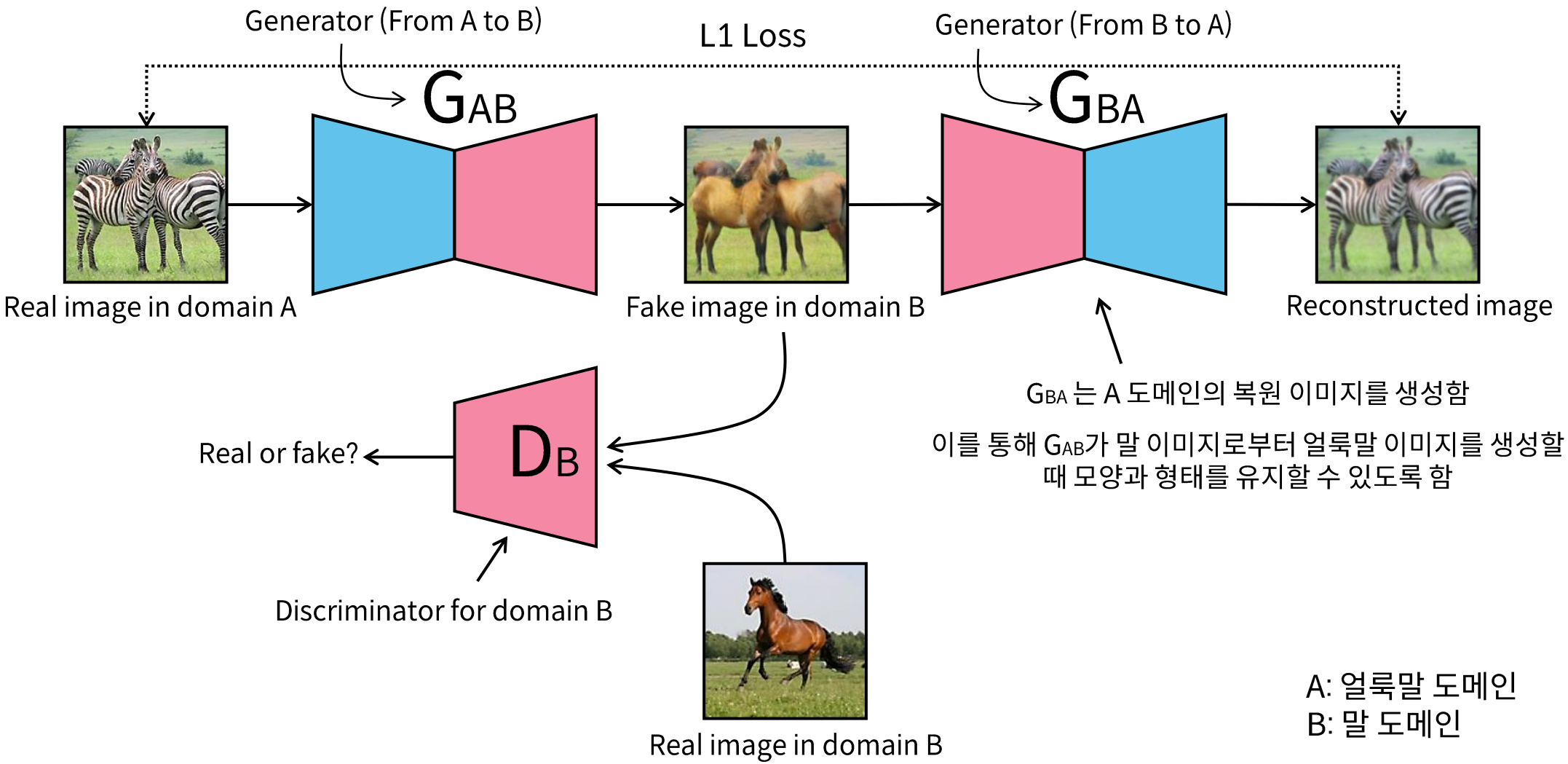
Img source: http://www.lherranz.org/2018/08/07/imagetranslation/

Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR'17

Ch 2. Image-to-Image Translation Models

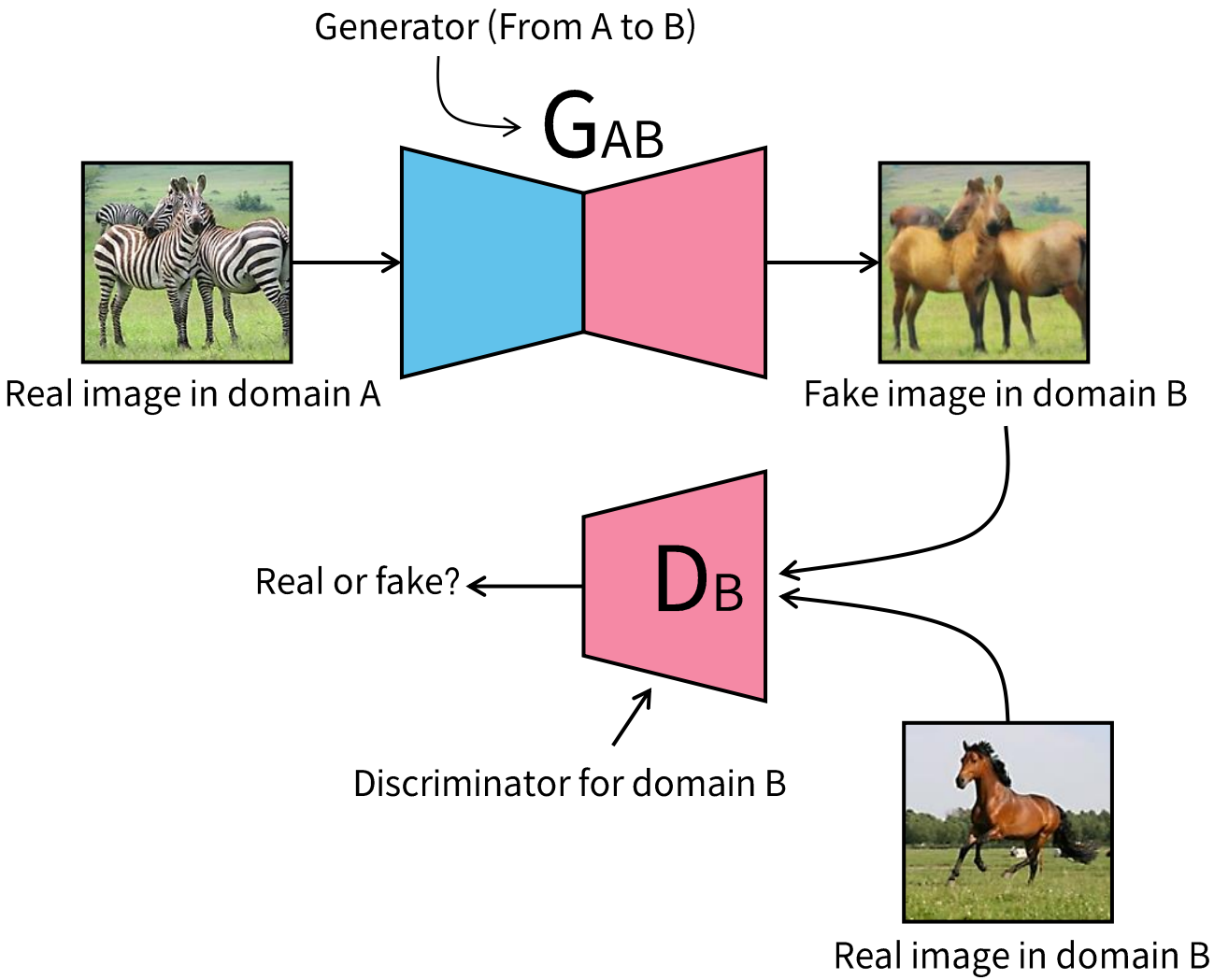
Clip 2. CycleGAN

CycleGAN: Unpaired Image-to-Image Translation



Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV'17

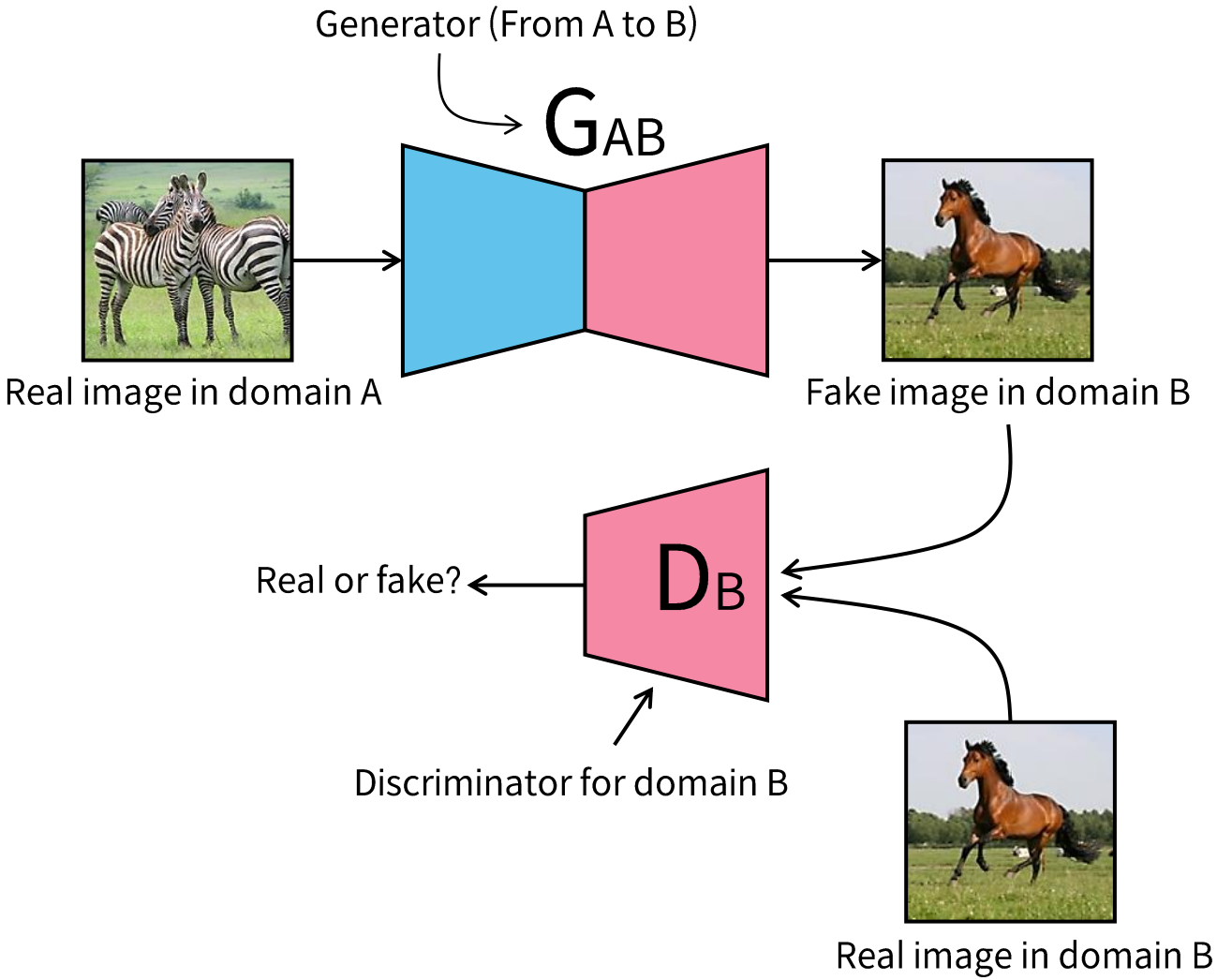
CycleGAN: Unpaired Image-to-Image Translation



A: 얼룩말 도메인
B: 말 도메인

Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV'17

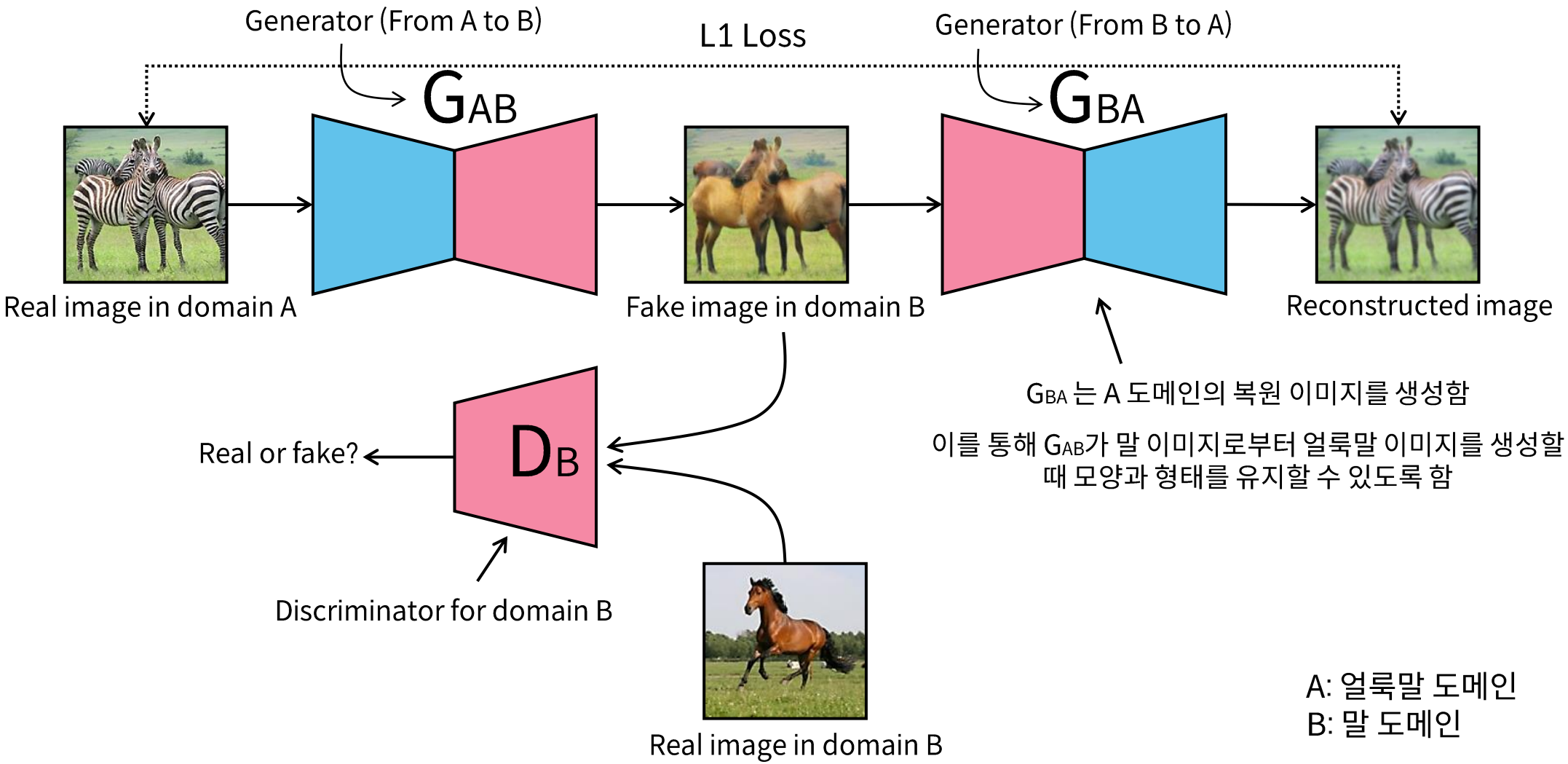
CycleGAN: Unpaired Image-to-Image Translation



A: 얼룩말 도메인
B: 말 도메인

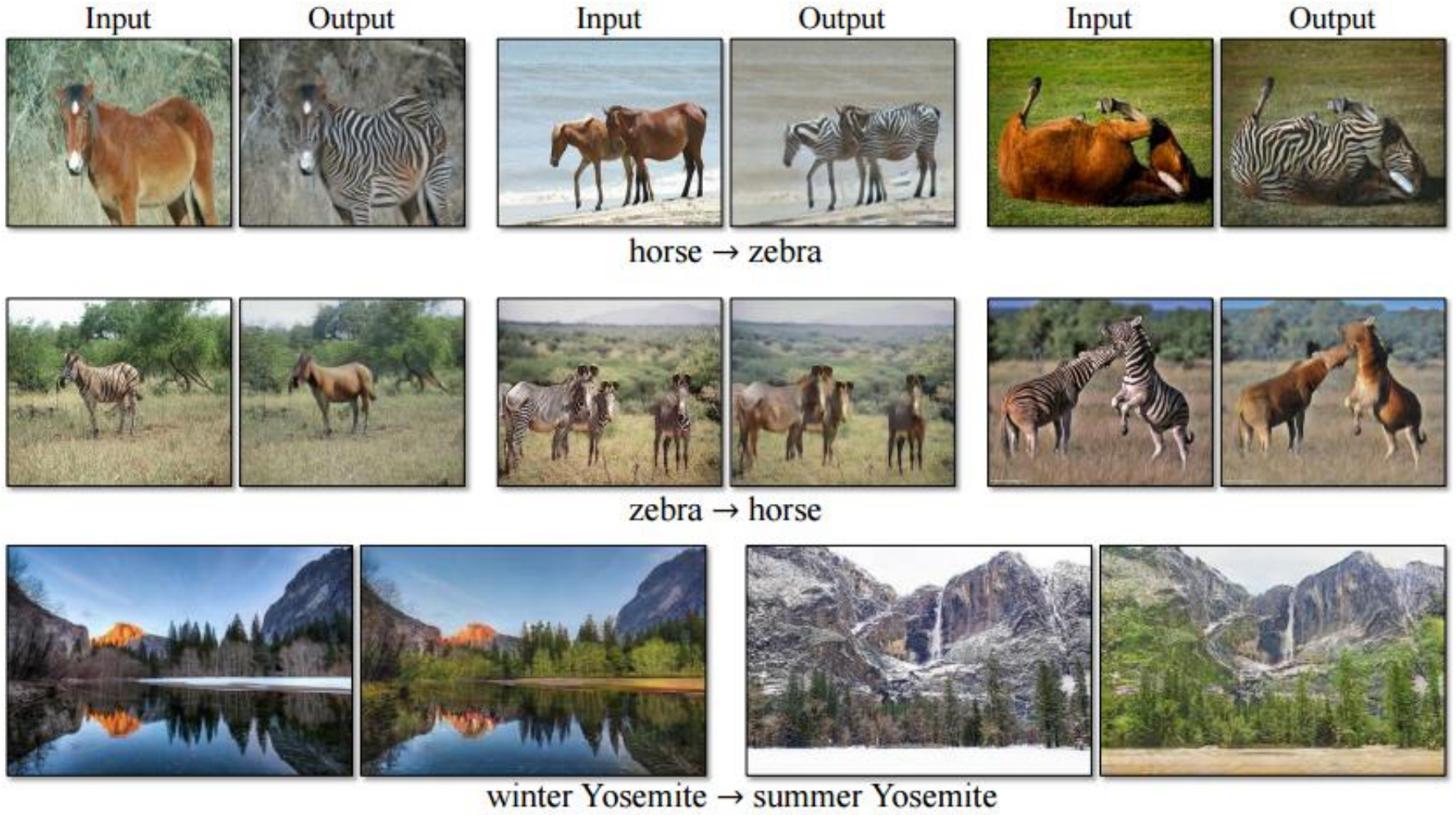
Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV'17

CycleGAN: Unpaired Image-to-Image Translation



Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV'17

CycleGAN 모델의 이미지 변환 결과



Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV'17

CycleGAN 모델의 이미지 변환 결과

홀수 컬럼에는 진짜 이미지, 짝수 컬럼에는 가짜 이미지가 그려져 있음

SVHN-to-MNIST

MNIST-to-SVHN

https://github.com/yunjey/mnist-svhn-transfer

Ch 2. Image-to-Image Translation Models

Clip 3. StarGAN

StarGAN: Multi-Domain Image-to-Image Translation

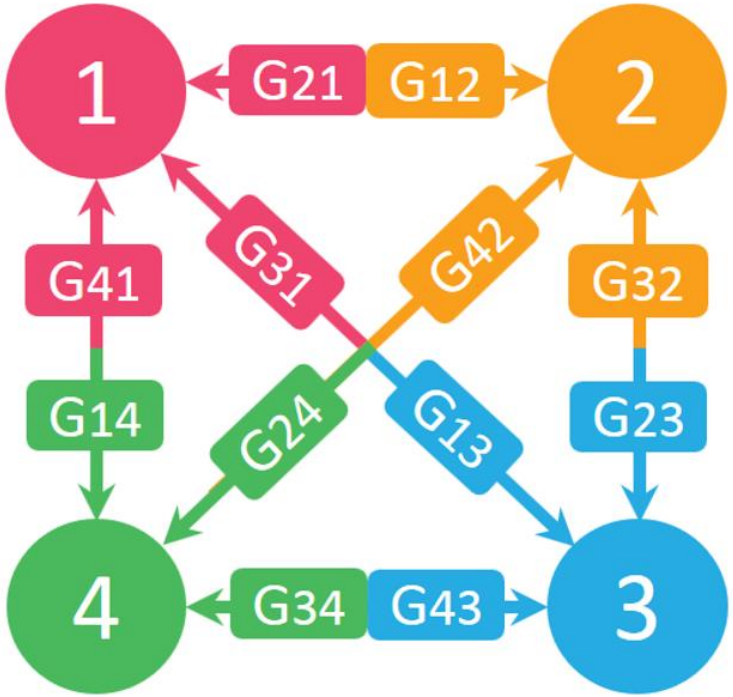
- 사용자가 원하는 조건(condition)에는 여러 도메인 혹은 여러 특성이 포함됨.

Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN: Multi-Domain Image-to-Image Translation

도메인 1
(금발 머리)

도메인 3
(갈색 머리)

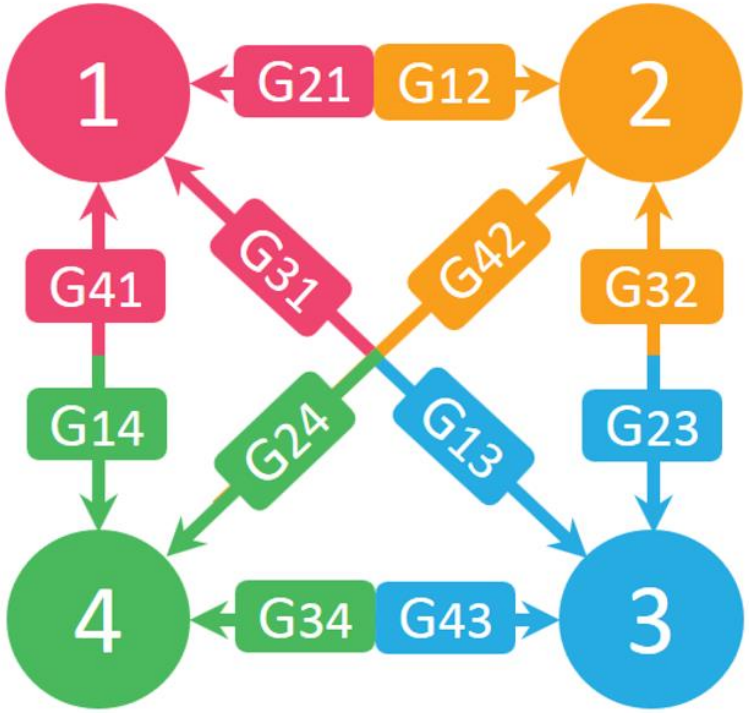


도메인 2
(검은 머리)

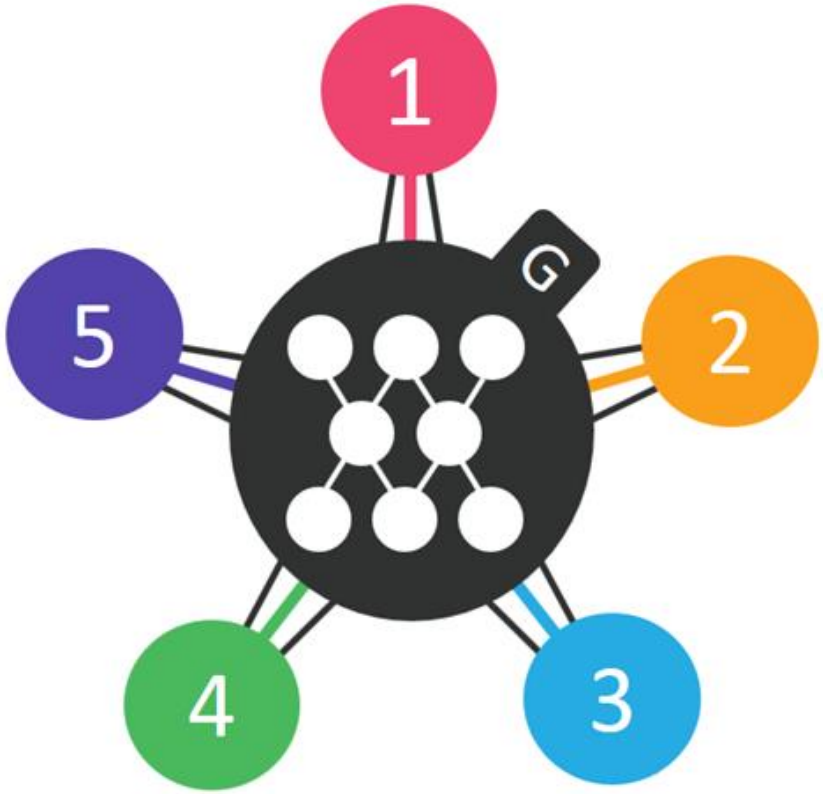
도메인 4
(회색 머리)

Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN: Multi-Domain Image-to-Image Translation



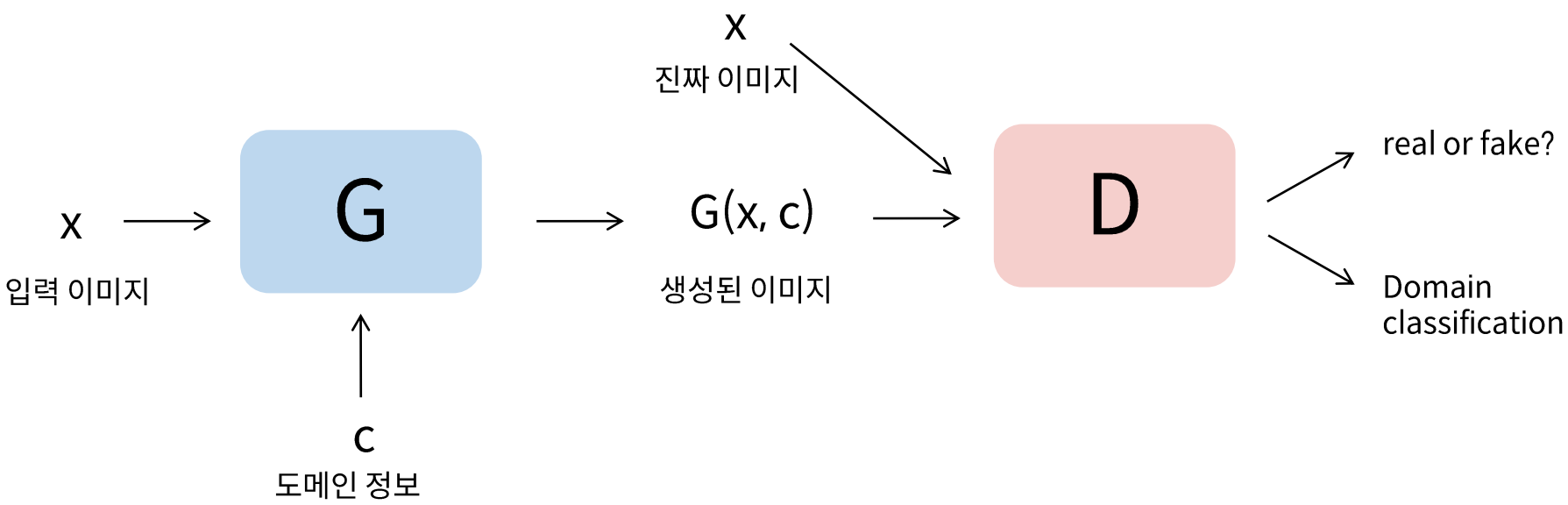
Cross-domain model



StarGAN

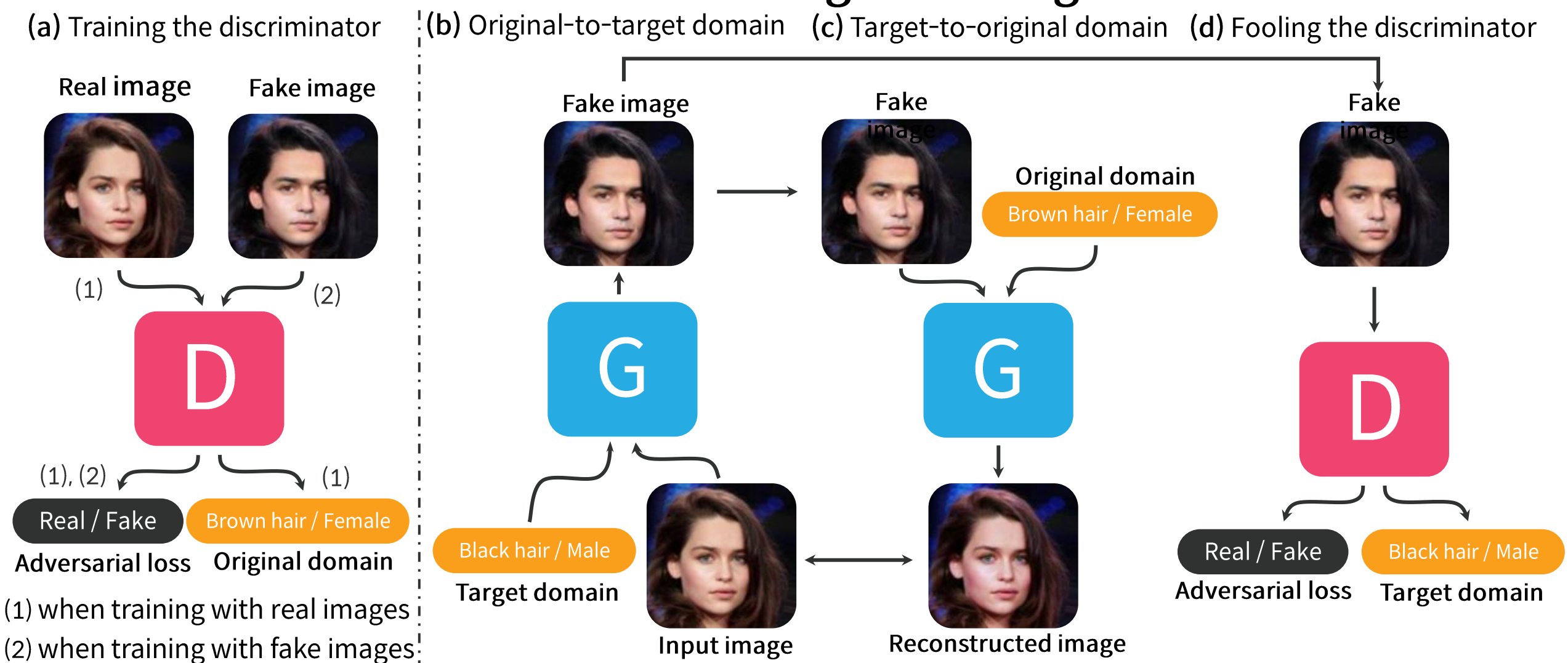
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN의 오버뷰



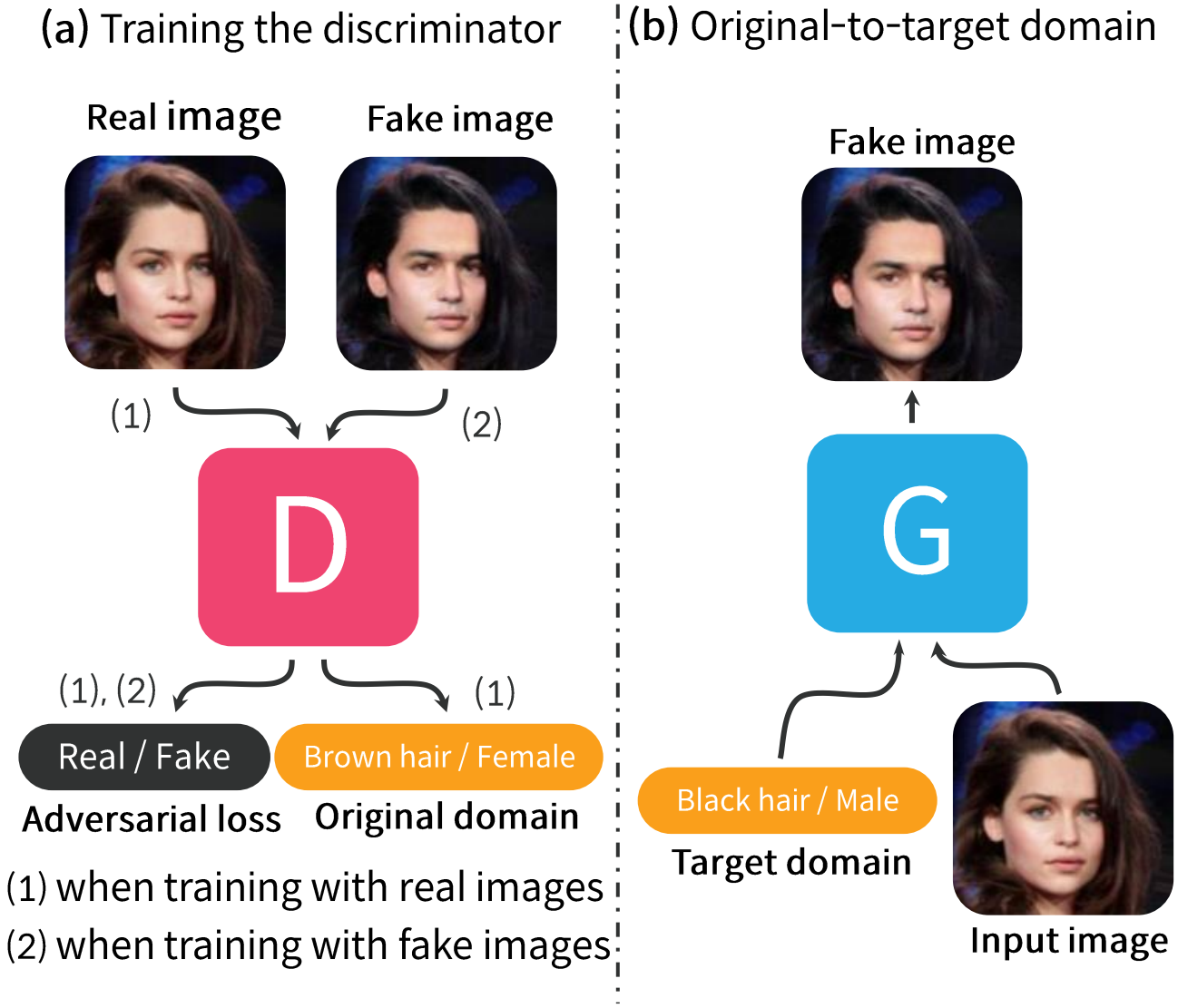
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN: Multi-Domain Image-to-Image Translation



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

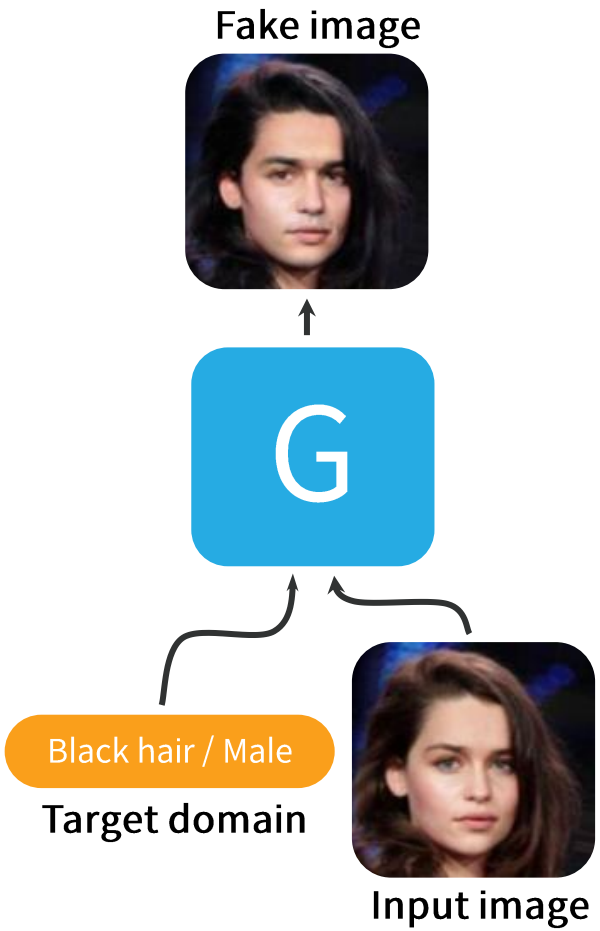
StarGAN 의 학습과정



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN 의 학습과정

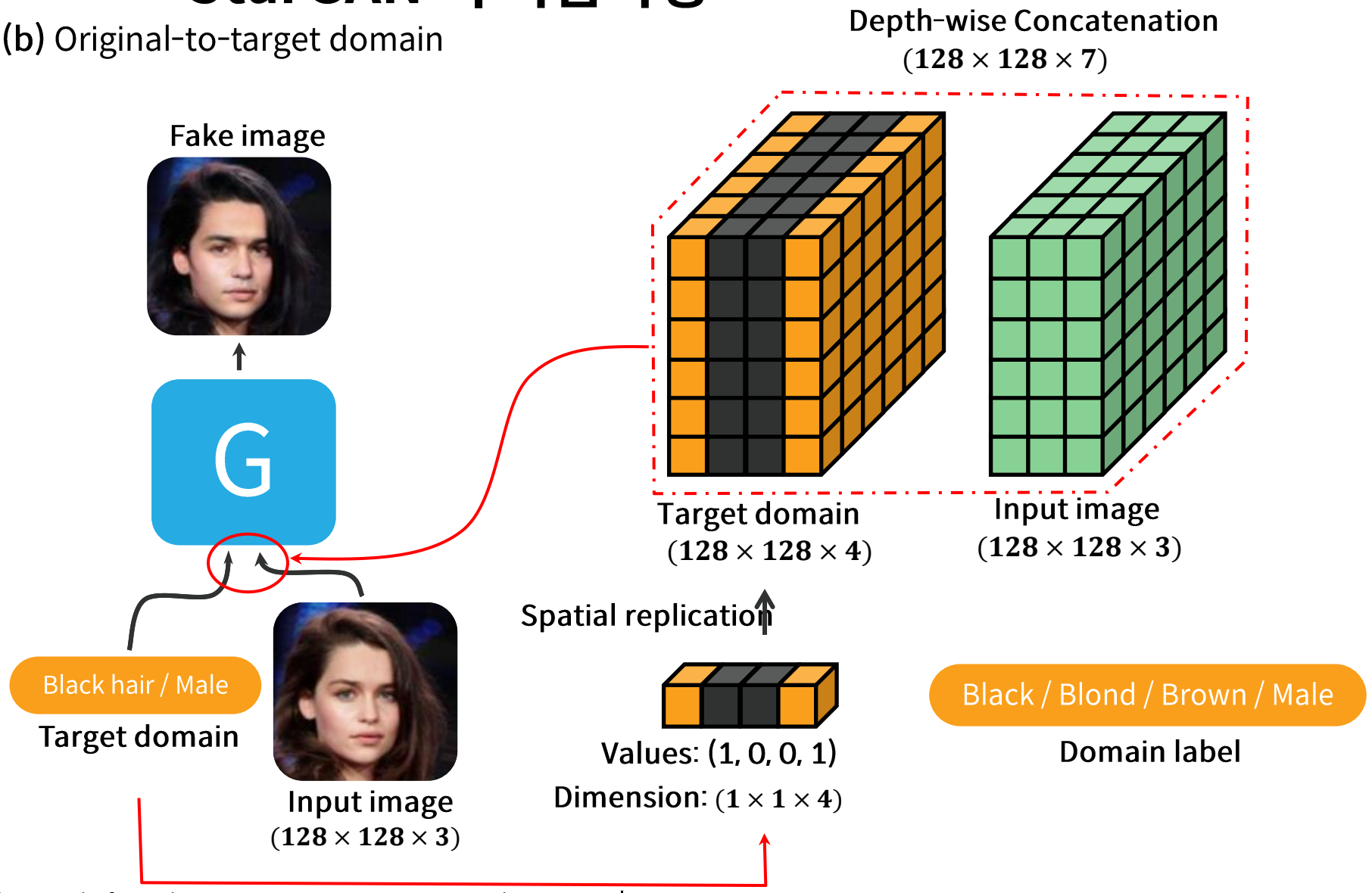
(b) Original-to-target domain



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

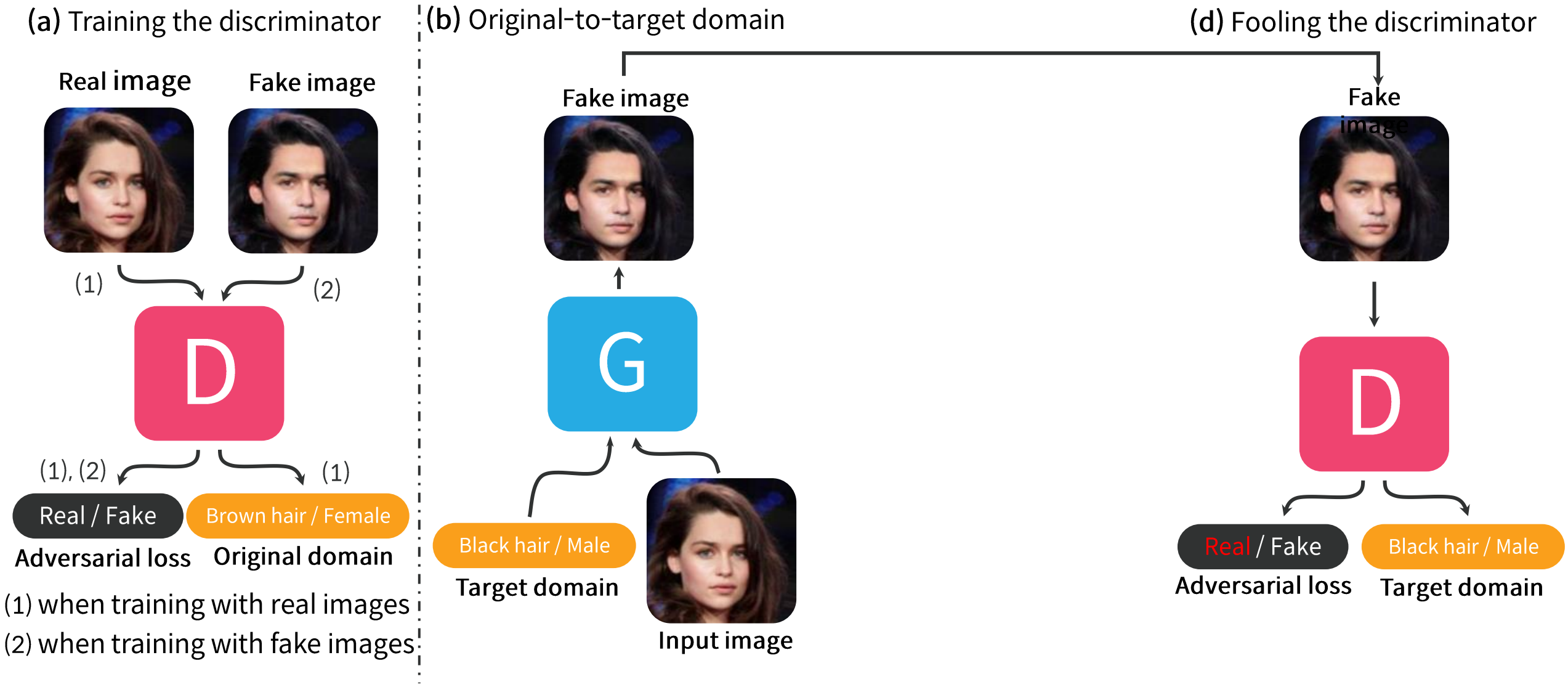
StarGAN 의 학습과정

(b) Original-to-target domain



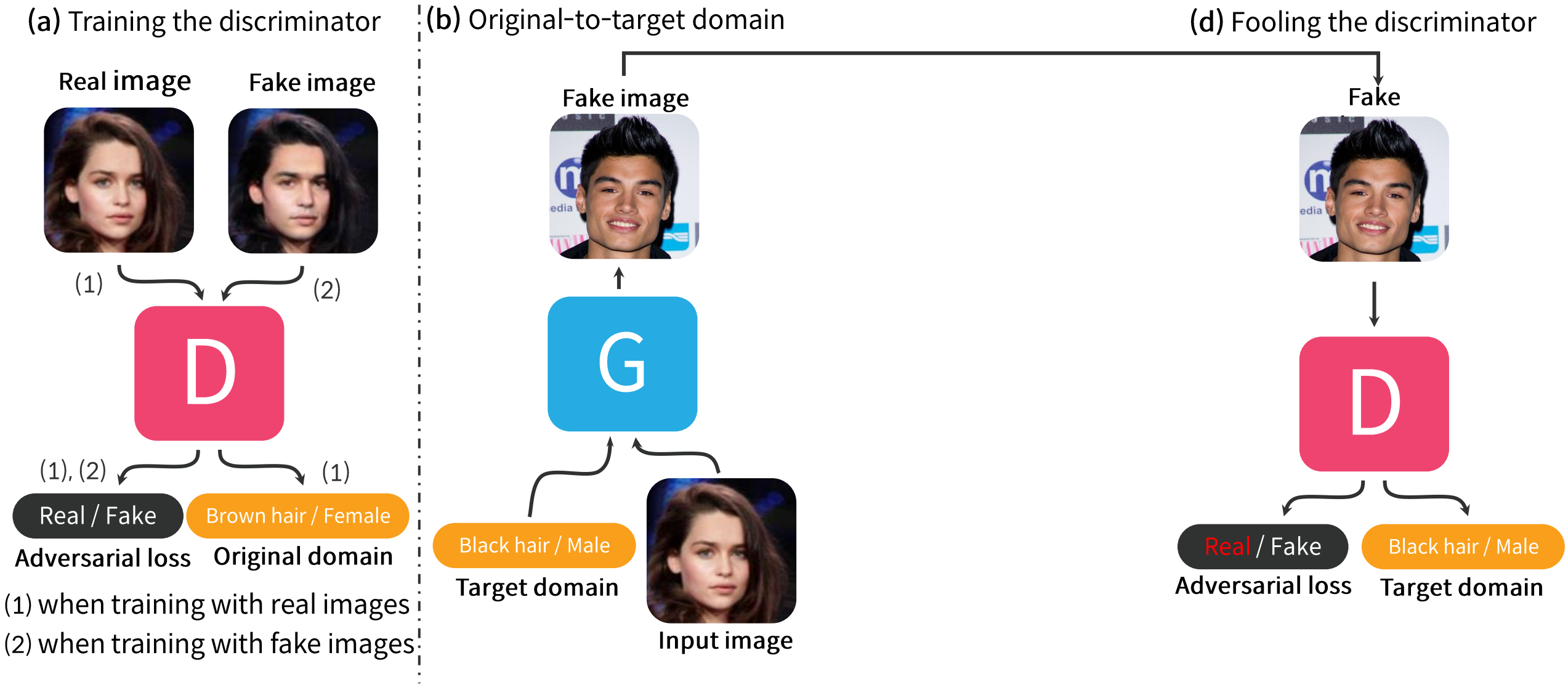
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN 의 학습과정



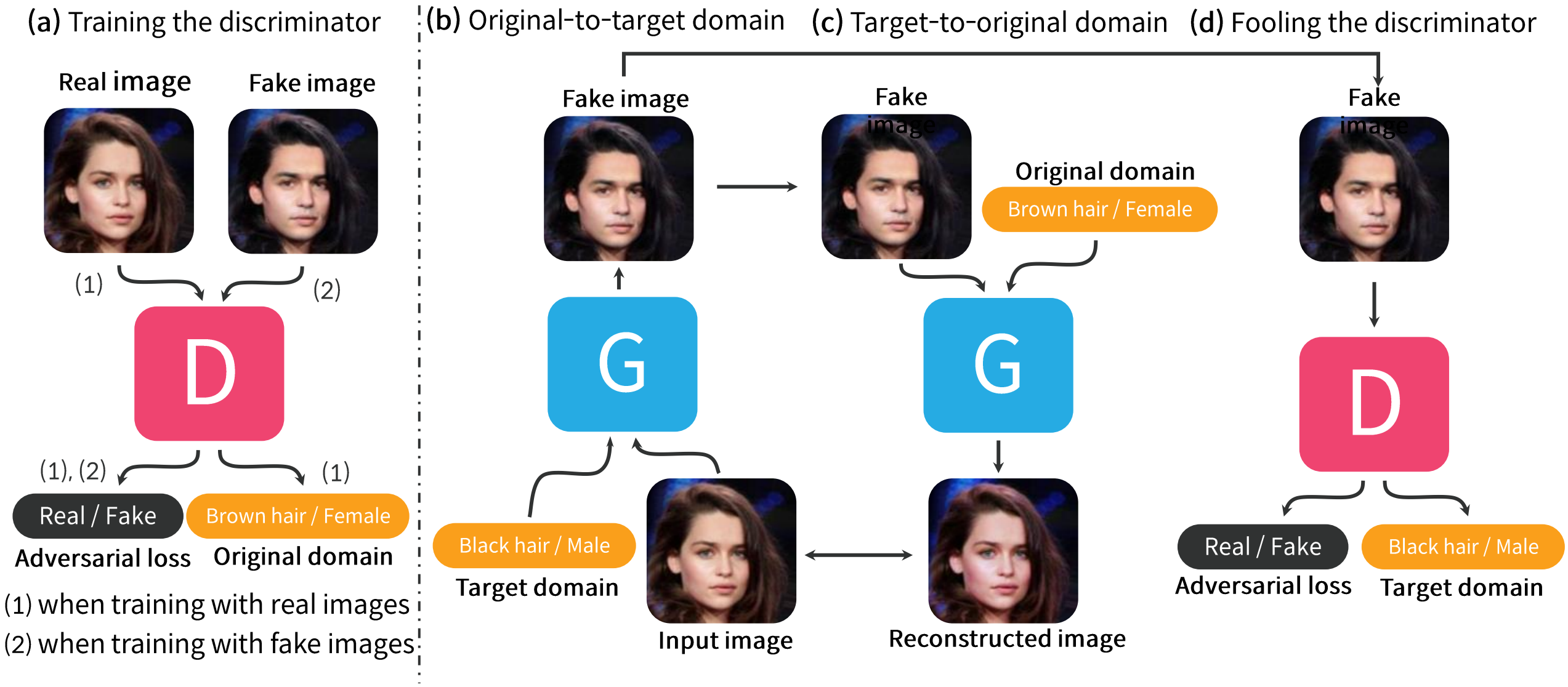
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN 의 학습과정



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN 의 학습과정



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets

CelebA (CelebFaces Attributes)

CelebA 데이터셋은 머리 색과 성별 같은 특성 (attribute) 들로
레이블링된 이미지 20만장으로 구성됨

RaFD (Raboud Faces Database)

RaFD 데이터셋은 기쁨, 화남, 슬픔 같은 얼굴 감정표현 특성
(attribute) 들로 레이블링된 이미지 4000장으로 구성됨

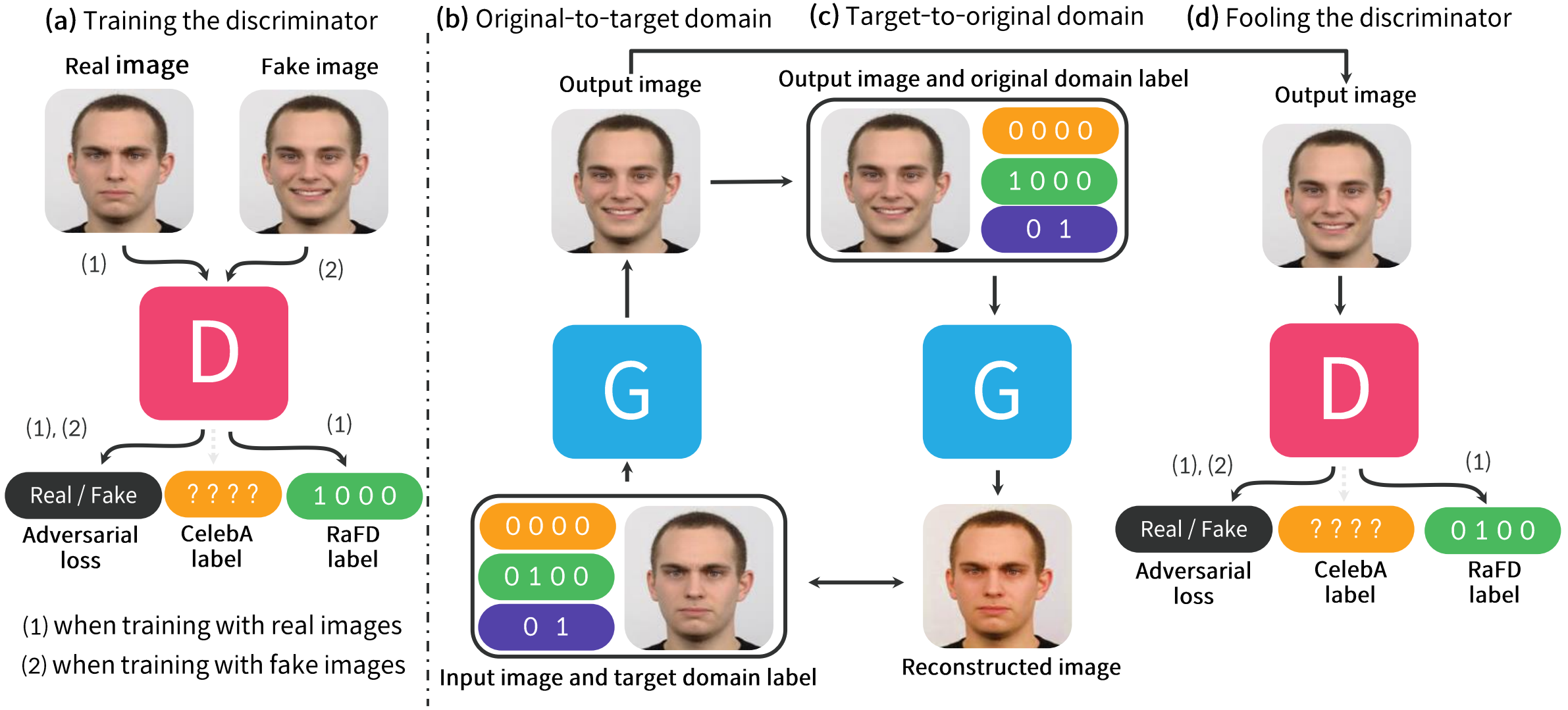
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets

CelebA label: Black / Blond / Brown / Male

RaFD label: Angry / Happy / Sad / Neutral

Mask vector: CelebA / RaFD



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets

CelebA label

Black / Blond / Brown / Male

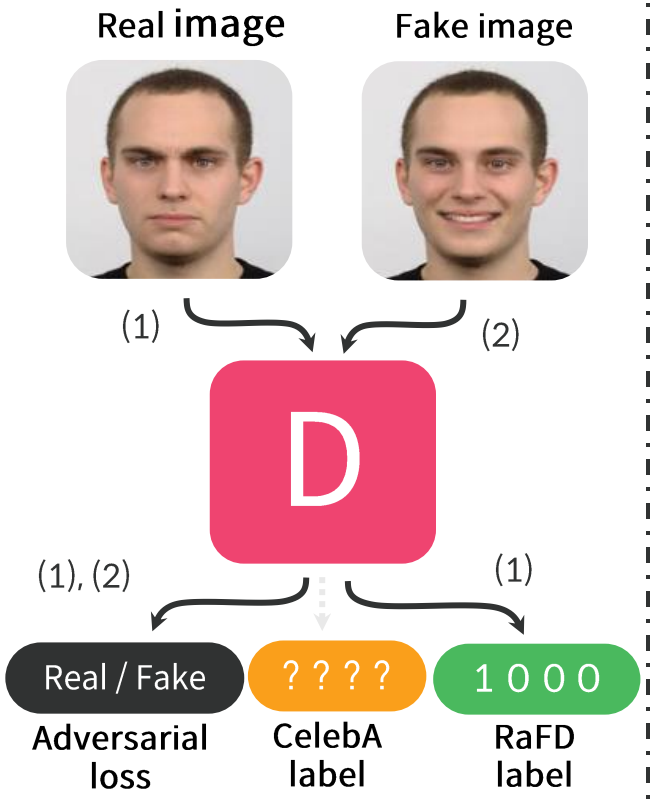
RaFD label

Angry / Happy / Sad / Neutral

Mask vector

CelebA / RaFD

(a) Training the discriminator (RaFD)



(1) when training with real images

(2) when training with fake images

Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets

CelebA label

Black / Blond / Brown / Male

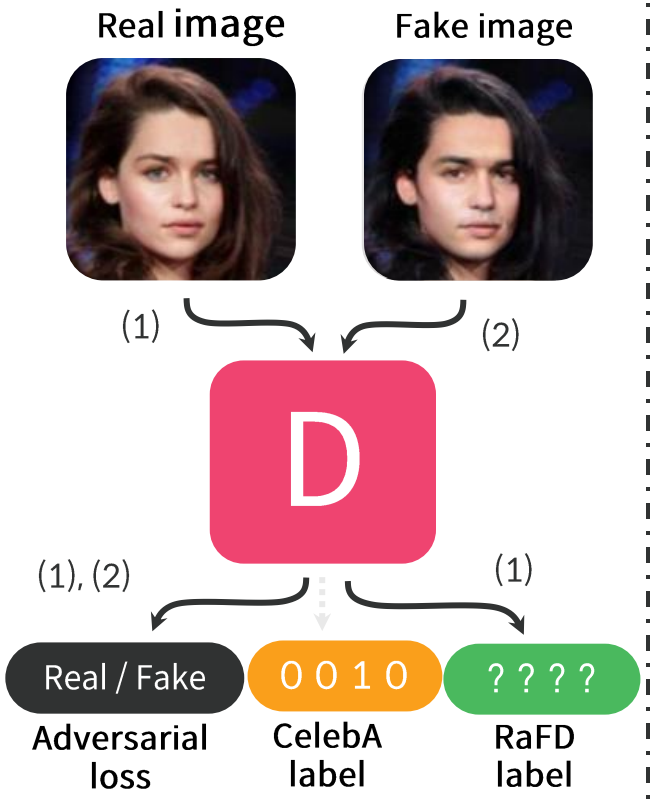
RaFD label

Angry / Happy / Sad / Neutral

Mask vector

CelebA / RaFD

(a) Training the discriminator (CelebA)



(1) when training with real images

(2) when training with fake images

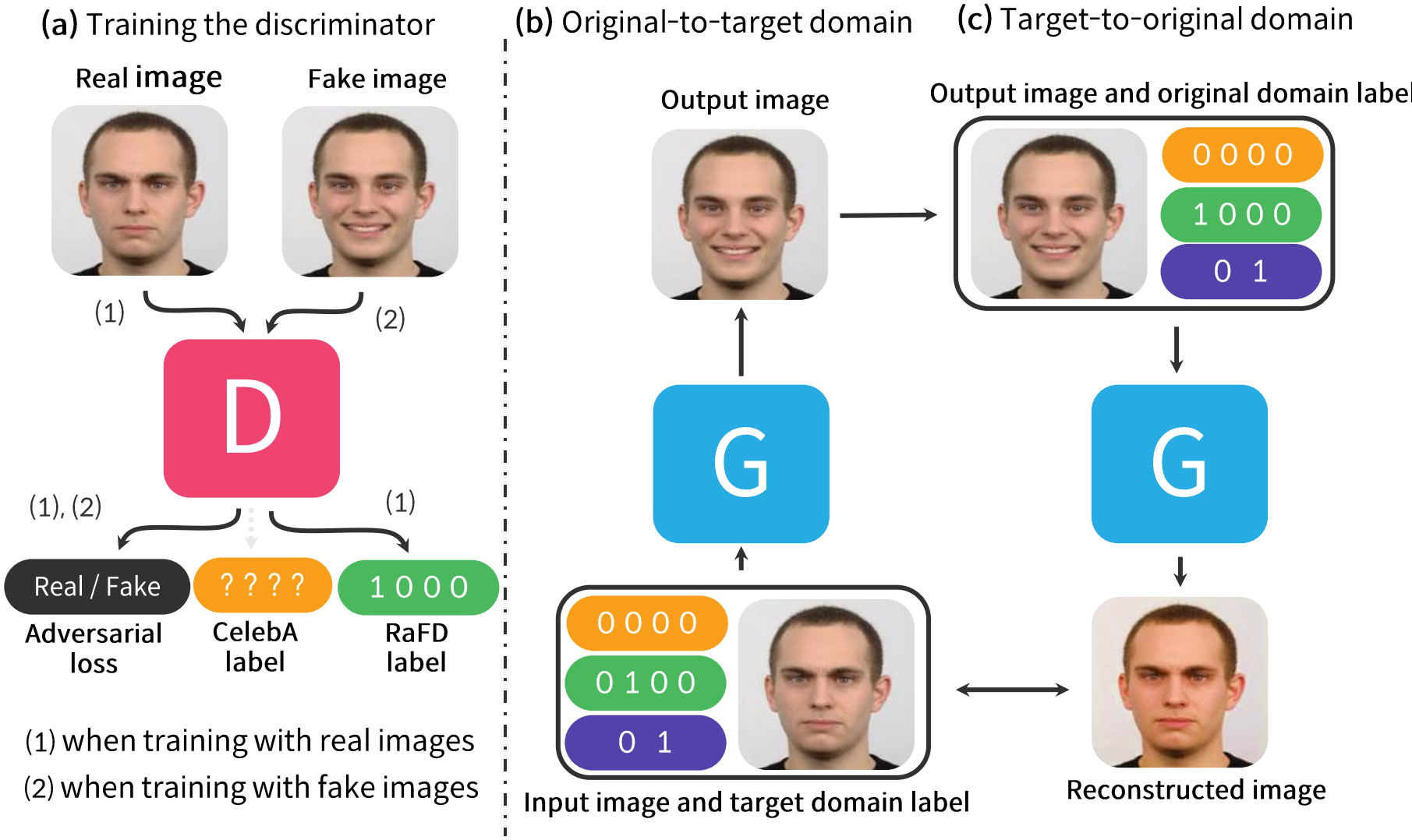
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets

CelebA label: Black / Blond / Brown / Male

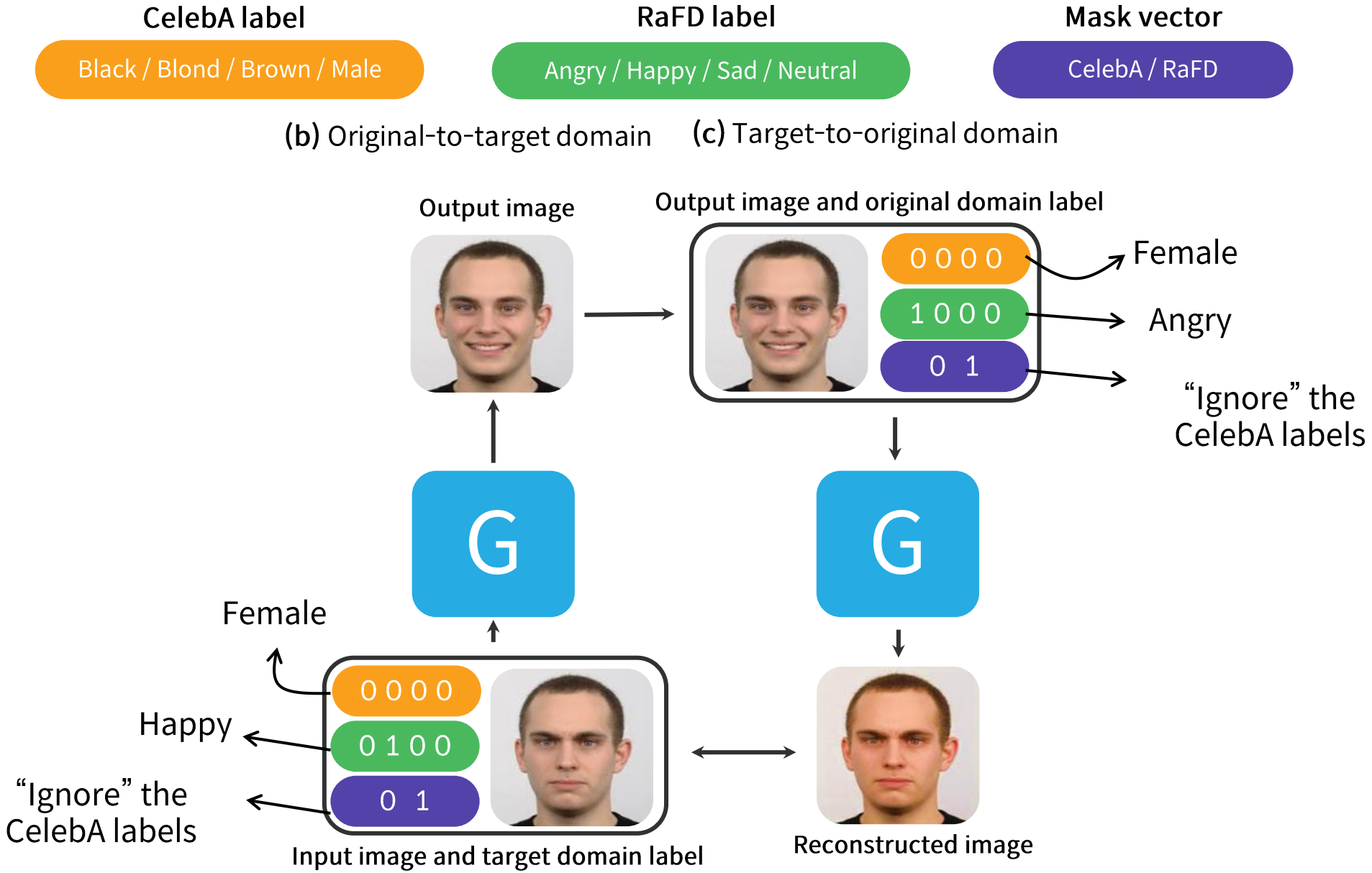
RaFD label: Angry / Happy / Sad / Neutral

Mask vector: CelebA / RaFD



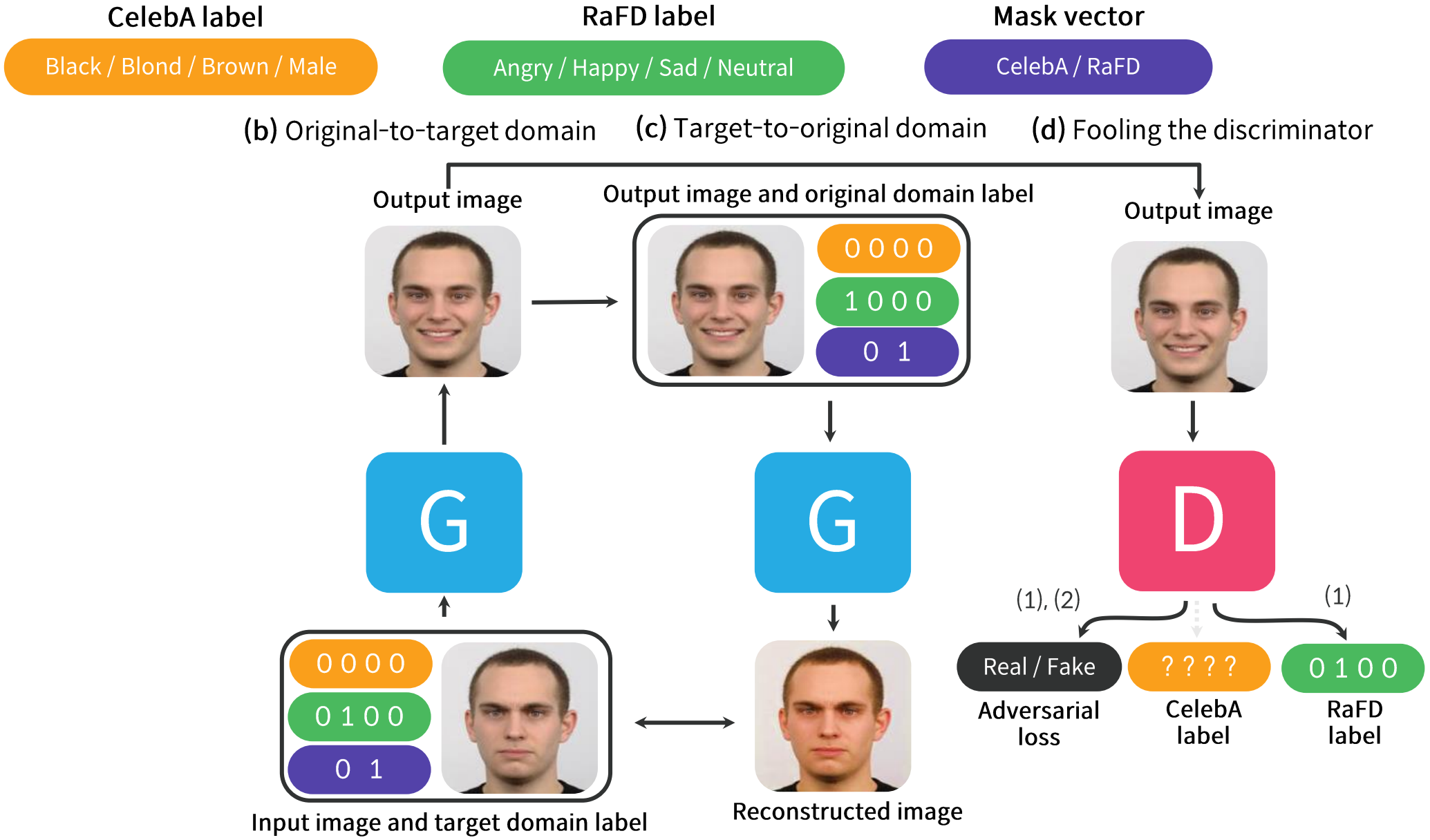
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR’18

StarGAN with multiple datasets



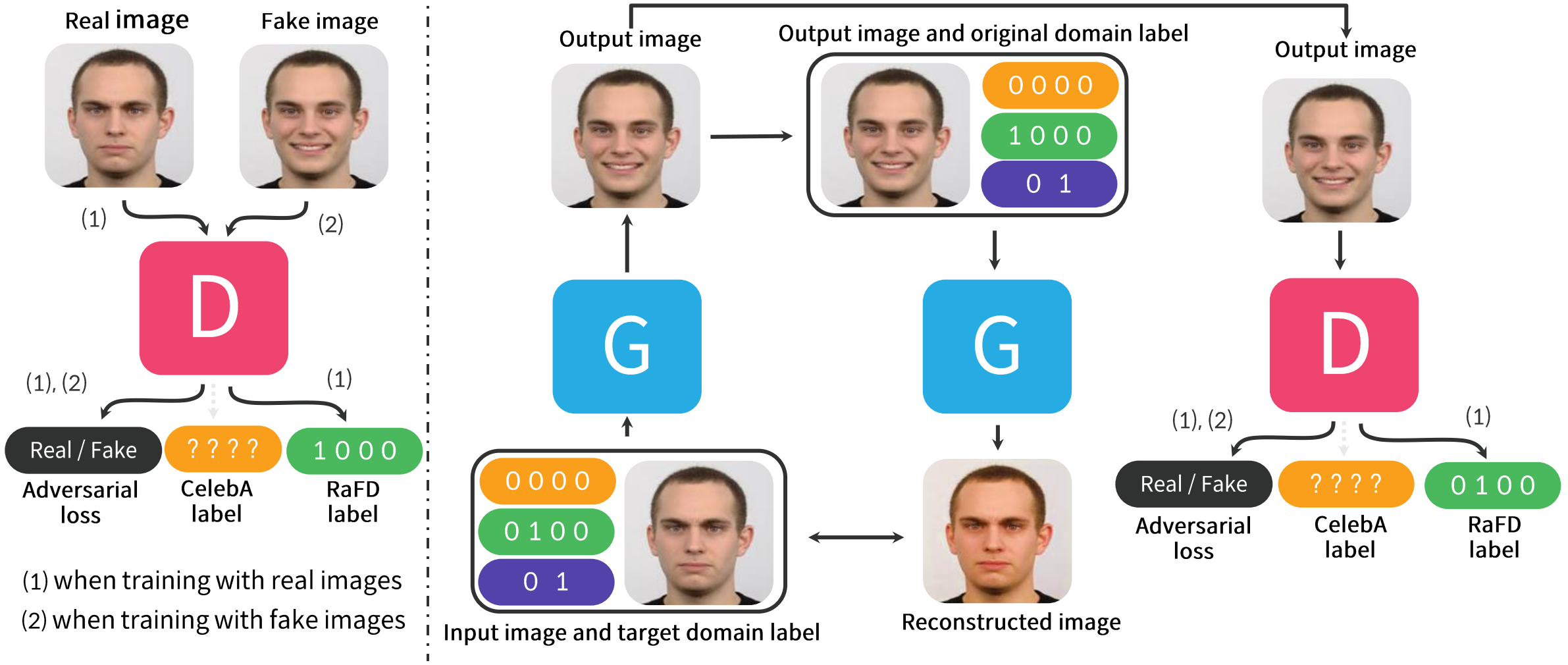
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets

CelebA label: Black / Blond / Brown / Male

RaFD label: Angry / Happy / Sad / Neutral

Mask vector: CelebA / RaFD



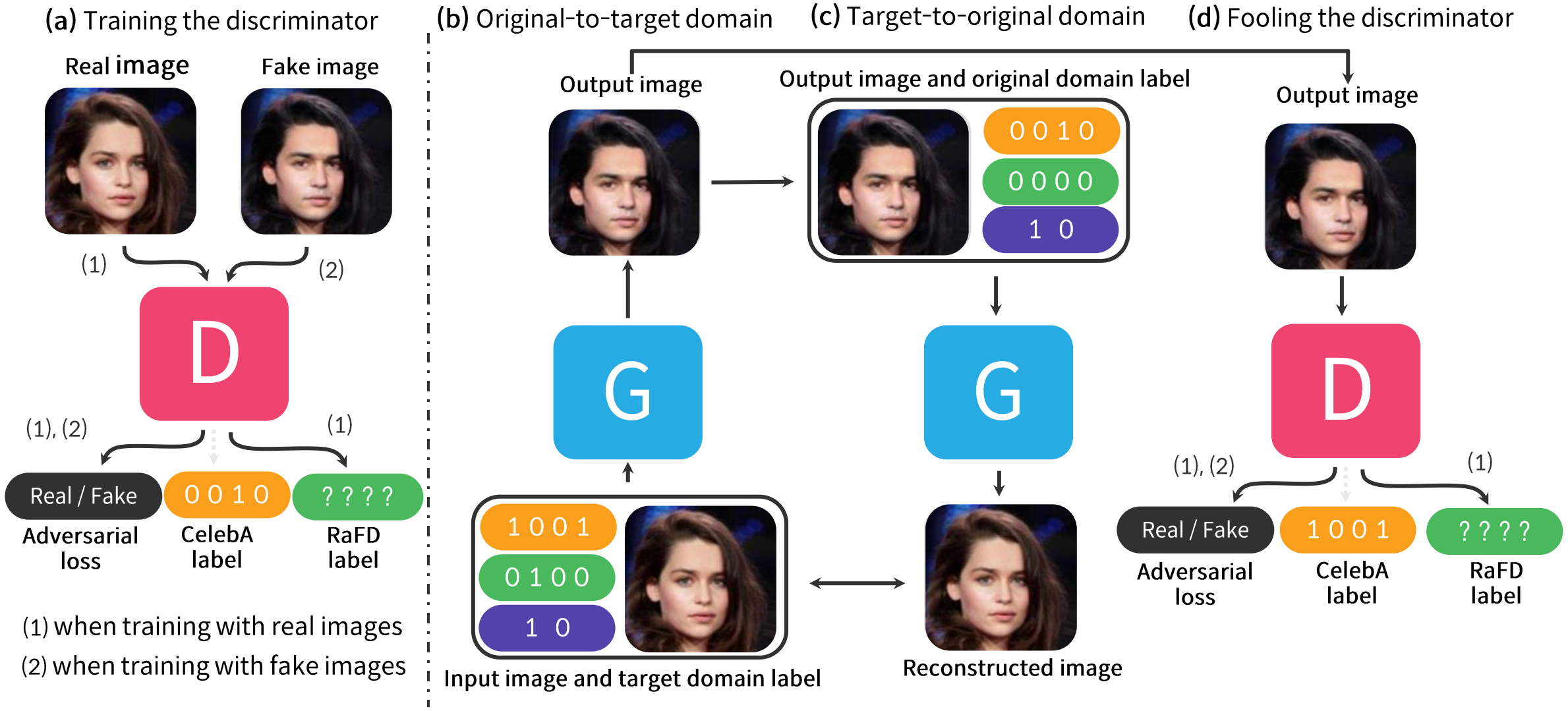
Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN with multiple datasets

CelebA label: Black / Blond / Brown / Male

RaFD label: Angry / Happy / Sad / Neutral

Mask vector: CelebA / RaFD



Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

StarGAN 모델의 이미지 변환 결과

Choi et al., StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation, CVPR'18

Useful Resource

- CS 236: Deep Generative Models (Stanford)
- CS 294-158 Deep Unsupervised Learning (Berkeley)
- CVPR'19 Tutorial on Deep Learning for Content Creation

https://nvlabs.github.io/dl-for-content-creation/

- CVPR'20 Workshop on AI for Content Creation

http://visual.cs.brown.edu/workshops/aicc2020/

- 2019 ICML Workshop on Human In the Loop Learning (HILL)

https://sites.google.com/view/hill2019

Videos: https://icml.cc/Conferences/2019/ScheduleMultitrack?event=3511

- 2020 IUI 2020 Workshop on Human-AI Co-Creation with Generative Models (programs are not yet updated)

https://hai-gen2020.github.io/

Thank you.

Thank you.