

2021 직업계고 AI 전문교육

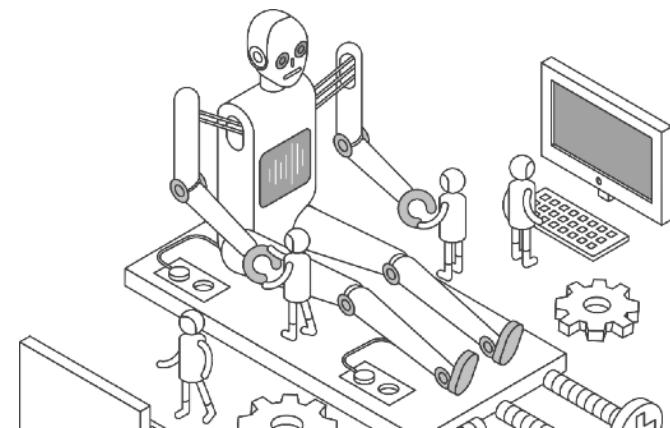
ARTIFICIAL INTELLIGENCE
BIG DATA
SMART FACTORY

AI·빅데이터 심화과정

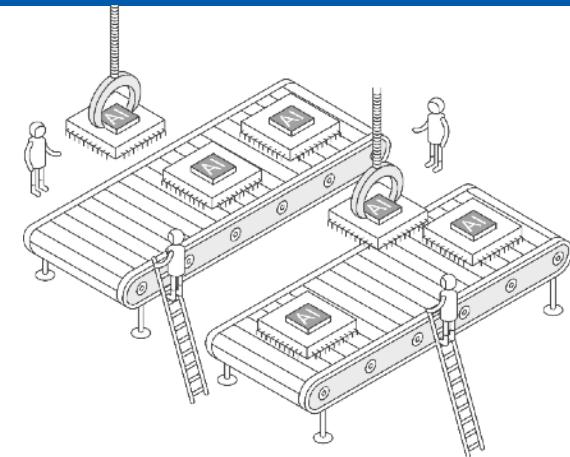
22. 난 스케치를 할 테니 너는 채색을 하거라

박수철

github.com/scspark20
GaudioLab, 모두의연구소



22. 난 스케치를 할 테니 너는 채색을 하거라



후반기 일정

Seq2Seq

9/24 - RNN으로 소설쓰기 (Aiffel 외)

9/29, 10/1 - 26. 뉴스 요약봇 만들기

10/6, 10/8, 10/15 - 27. 트랜스포머로 만드는 대화형 챗봇

CNN/GAN

10/20, 10/22 - 22. 난 스케치를 할테니 너는 채색을 하거라

10/27, 11/3 - 21. 흐린 사진을 선명하게

RNN+CNN

11/5 - RNN으로 음성인식하기 (Aiffel 외)

11/10, 11/12 - 19. 직접 만들어보는 OCR

Ablation study

11/17 - 17. 없다면 어떻게 될까?

UDACITY TALKS

Yann LeCun

Director of AI Research

facebook

A portrait photograph of Yann LeCun, a middle-aged man with short brown hair and glasses, wearing a blue polo shirt under a dark grey zip-up hoodie. The hoodie has the Facebook logo and 'Facebook AI Research' printed on it.

“The most important idea in Machine Learning in the last 20 years”

- GAN은 Generative Adversarial Networks의 약자로 두 개의 네트워크가 서로 독립적으로 경쟁하며 트레이닝이 진행됩니다.
- 이는 눈을 가린채 그림을 그리는 화가와 그림을 판별하는 탐정에 비유할 수 있습니다.
- 화가는 진품과 모방한 모조품을 그리는 역할을 합니다. 하지만 진품을 직접 볼 수는 없고, 오직 탐정이 가려내는 결과만을 들을 수 있습니다. 많은 시행 착오 끝에 결국 탐정이 진품과 모조품을 구별할 수 없도록 만듭니다.
- 탐정은 화가가 그리는 모조품과 진품을 받고서 서로 구별하는 능력을 키웁니다. 그리고 구별한 결과를 화가에게 알려줍니다.

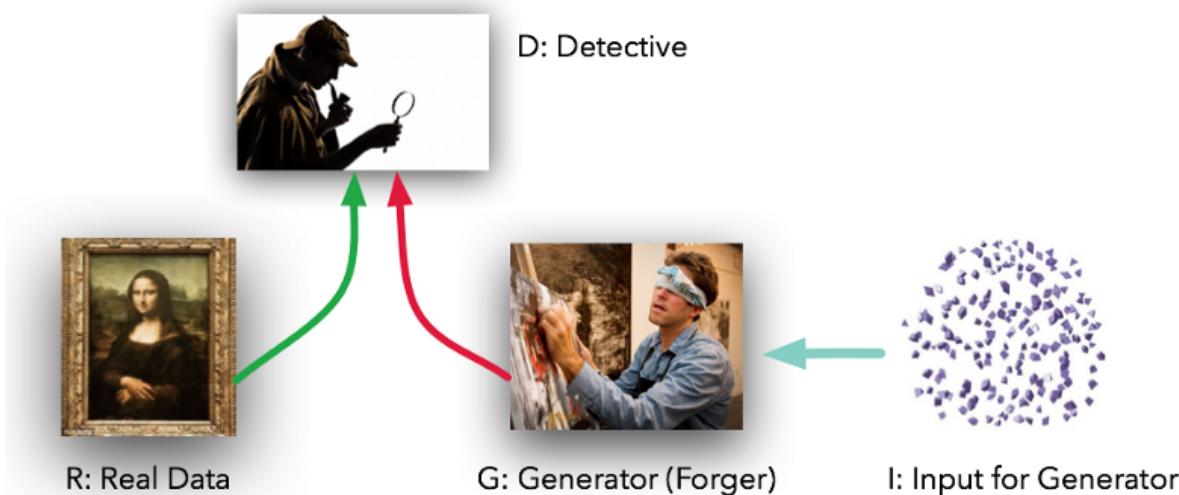
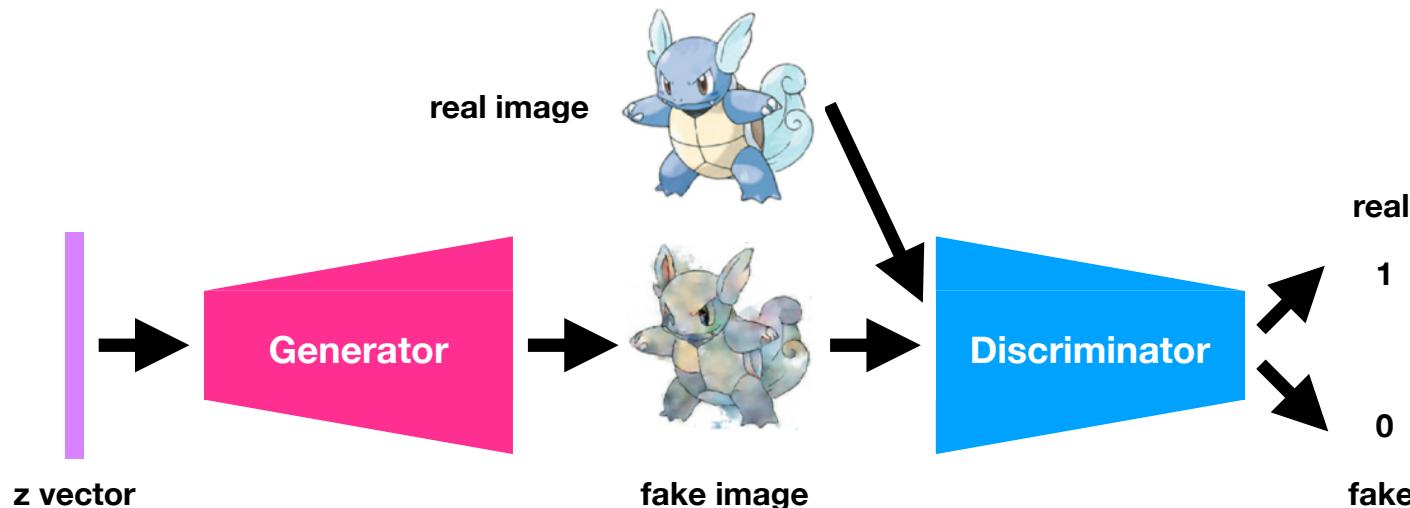


Figure credit: Blog. Generative Adversarial Networks (GANs) in 50 lines of code (PyTorch)



- 전체 네트워크는 그림을 생성하는 네트워크인 generator와 그림이 진짜인지 가짜인지 판별하는 discriminator로 이루어져 있습니다.
- Generator는 임의의 random distribution으로부터 샘플링한 z vector를 사용해 그림을 생성하며, 이를 fake image라 칭합니다.
- Discriminator는 generator가 생성한 fake image와 trainset으로 준비된 real image를 입력받아 각각 0과 1을 출력하도록 학습합니다.

Cost Functions

$$\mathcal{L}^D = -\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$\begin{aligned} & \mathbb{E}_{x \sim p_{data}(x)}[H[Ber(y; \mu = 1), Ber(y; \mu = D(x))]] \\ &= \mathbb{E}_{x \sim p_{data}(x)}[-Ber(1; \mu = 1)\log Ber(1; \mu = D(x)) - Ber(0; \mu = 1)\log Ber(0; \mu = D(x))] \\ &= \mathbb{E}_{x \sim p_{data}(x)}[-1 \log D(x) - 0 \log(1 - D(x))] \\ &= -\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] \end{aligned}$$

$$\begin{aligned} & \mathbb{E}_{z \sim p_z(z)}[H[Ber(y; \mu = 0), Ber(y; \mu = D(G(z)))]]] \\ &= \mathbb{E}_{z \sim p_z(z)}[-Ber(1; \mu = 0)\log Ber(1; \mu = D(G(z))) - Ber(0; \mu = 0)\log Ber(0; \mu = D(G(z)))] \\ &= \mathbb{E}_{z \sim p_z(z)}[-0 \log D(G(z)) - 1 \log(1 - D(G(z)))] \\ &= -\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \end{aligned}$$

$$\mathcal{L}^G = -\mathbb{E}_{x \sim p_{data}(x)}[\log D(G(z))]$$

Cross-Entropy

$$\begin{aligned} H(p, q) &= -\mathbb{E}_p[\log q] \\ &= -\sum_{x \in X} p(x) \log q(x), \text{ for discrete} \end{aligned}$$

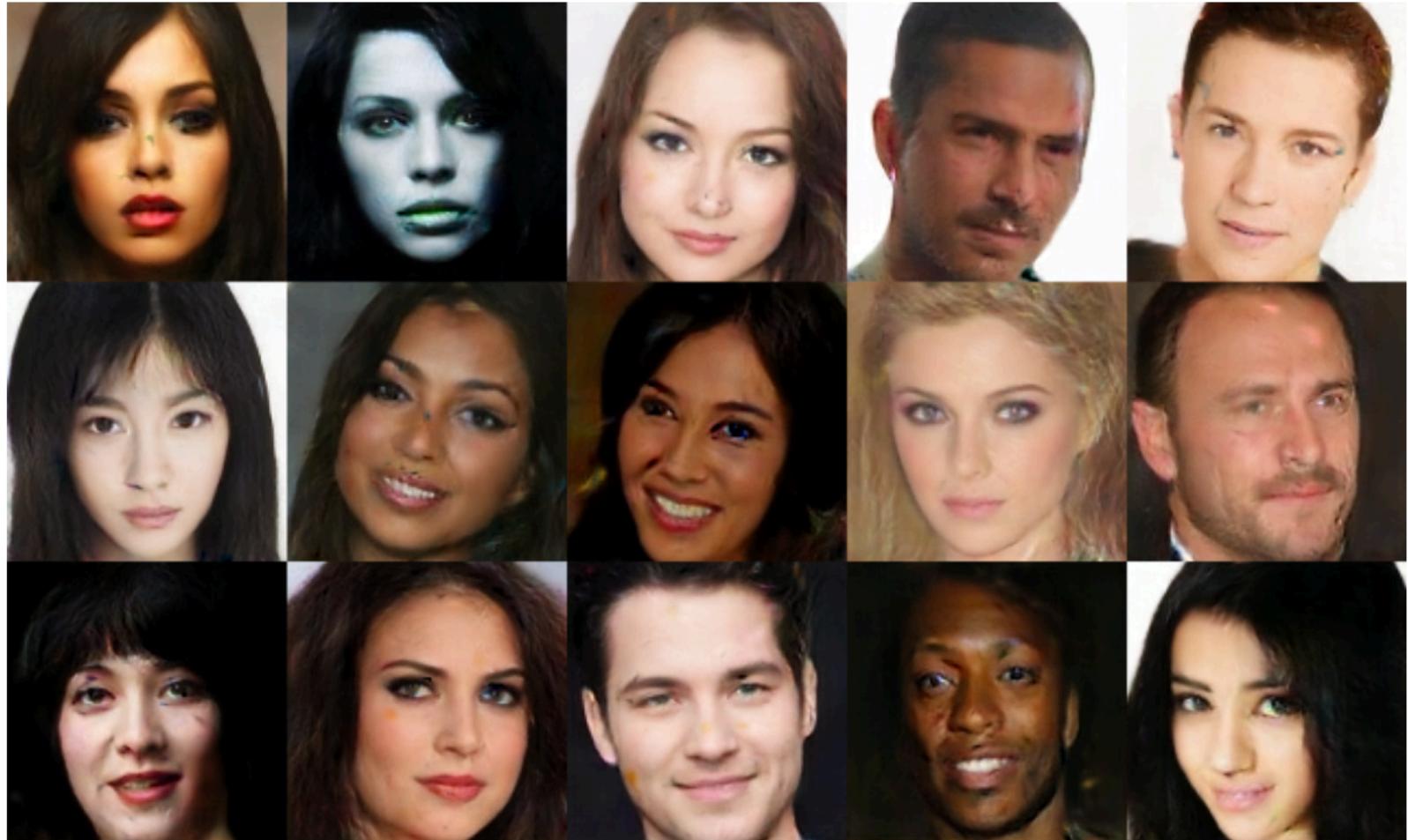
GAN 2014.06



DCGAN 2015.11



BEGAN 2017.03



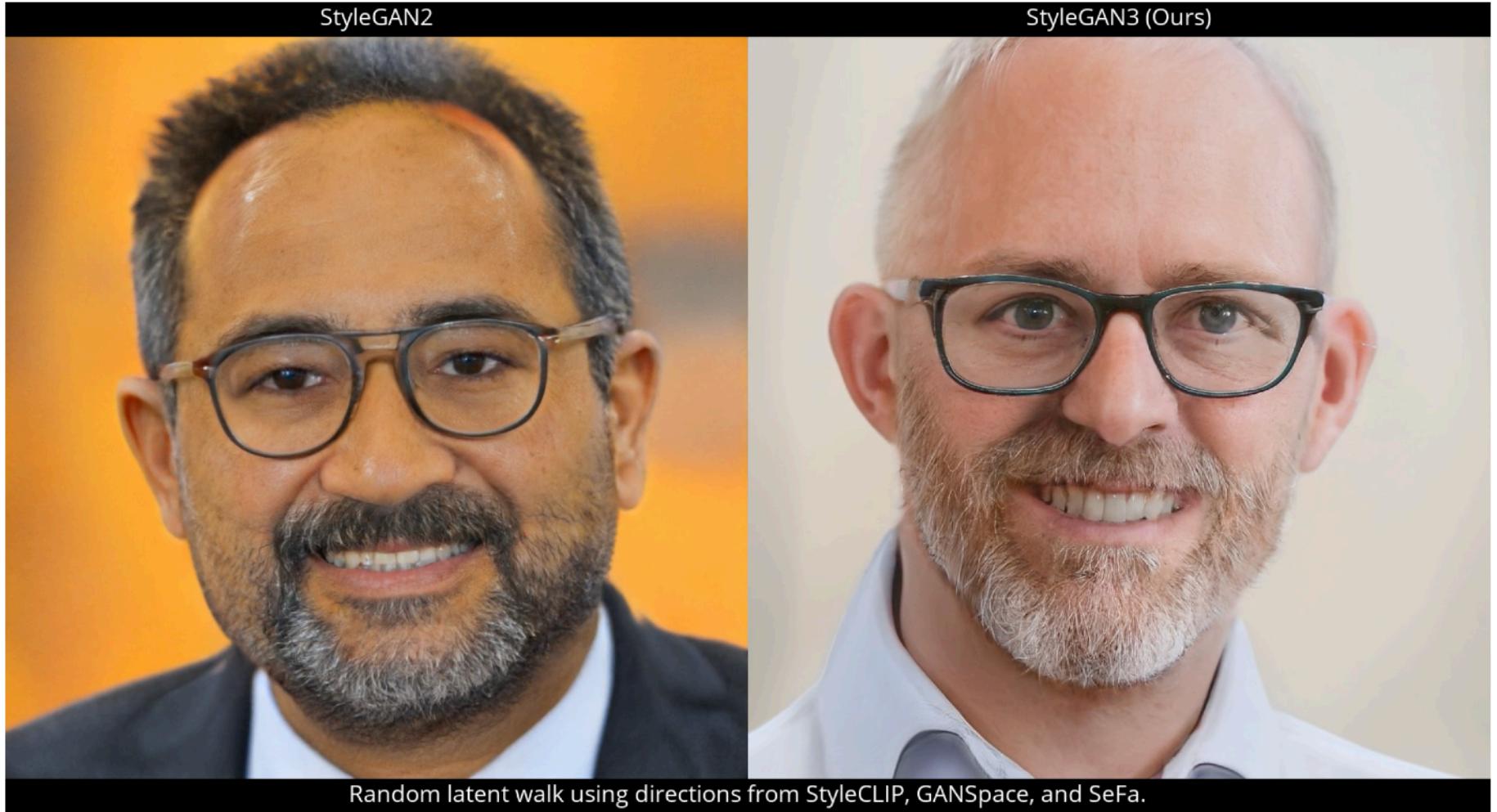
ProgressiveGAN 2017.10



StyleGAN 2018.12



StyleGAN3 2021.06



nvlabs.github.io/stylegan3

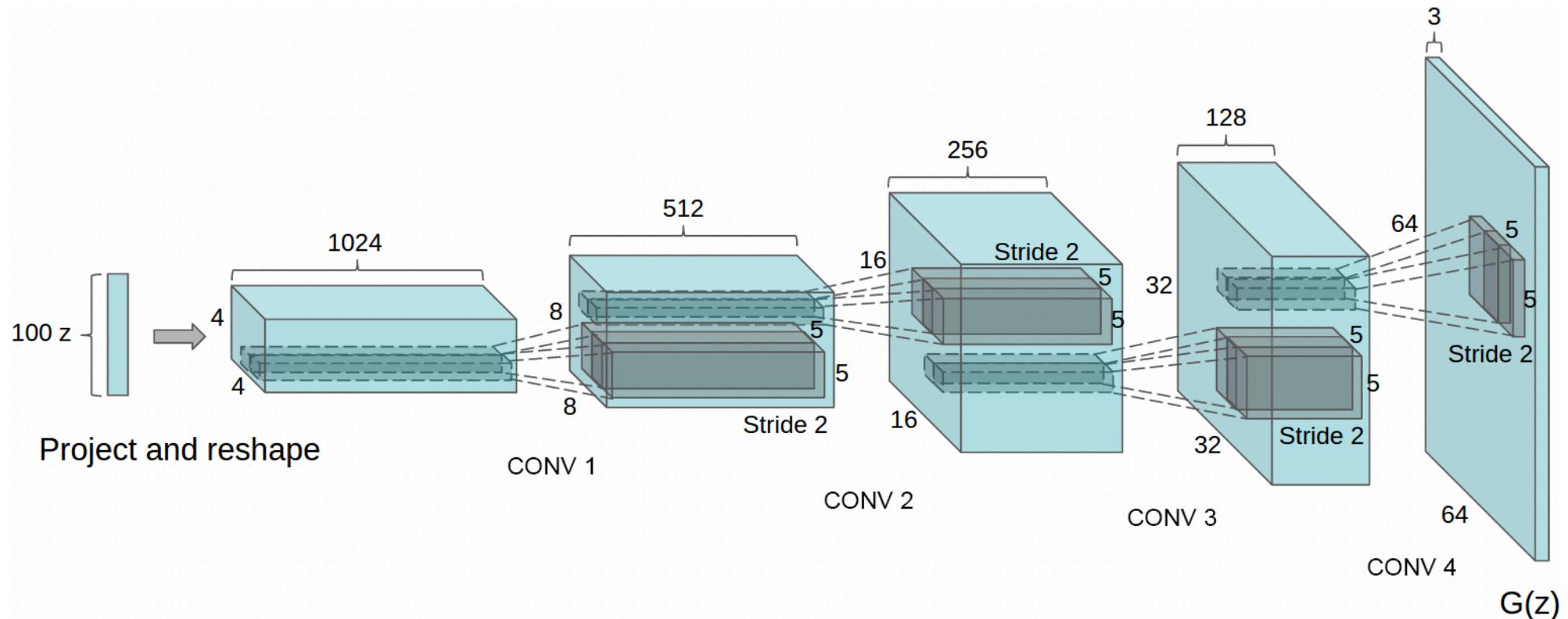
- DCGAN은 GAN에 deep convolutional networks를 적용한 모델로써 GAN이 활발히 연구되게 한 시작점입니다.
- GAN의 트레이닝은 매우 불안정하여 많은 데이터를 오래 트레이닝한다고 해서 성공적인 결과를 낳는 것이 아닙니다. 모델의 구성요소들 즉, layer의 종류, 층 수, 채널 수, 커널의 크기, activation, normalization의 종류와 유무, optimizer의 종류와 learning rate 모든 것이 잘 맞아떨어져야 제대로 트레이닝됩니다.
- 어떤 구성요소가 왜 성공적으로 트레이닝을 이끄는지 그 이유에 대해서는 명확하게 밝혀내지 못합니다. 단지 많은 실험에 의해 어떤 구성요소들이 좋은 결과를 낳는지 알아내고 있습니다.

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.

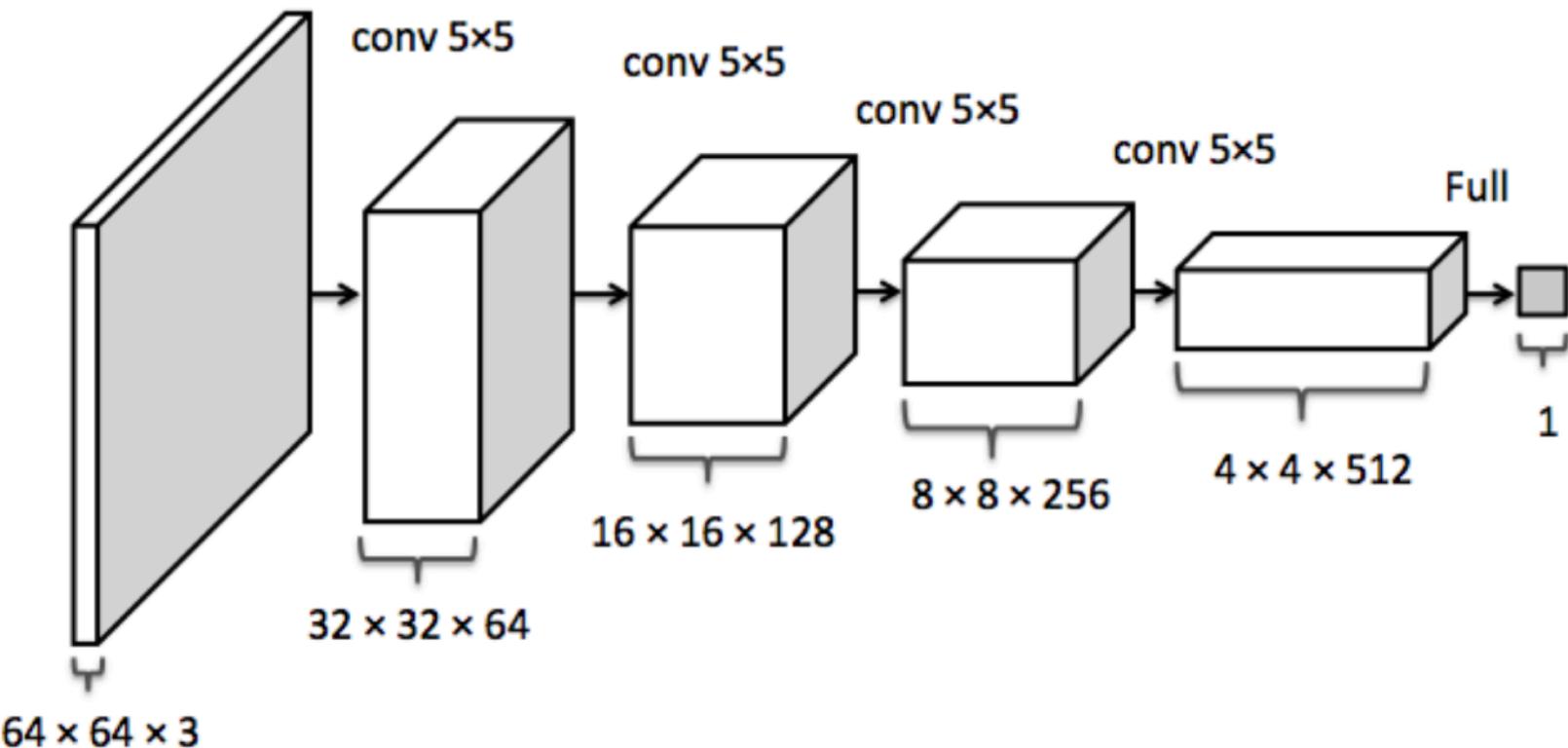
Figure credit: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN



- DCGAN의 generator는 conv2d transpose layer로 구성되어 있으며, 점차 크기를 늘여가고 channel을 줄여가 마지막에 이미지와 같은 shape의 결과를 내보냅니다.

Figure credit: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

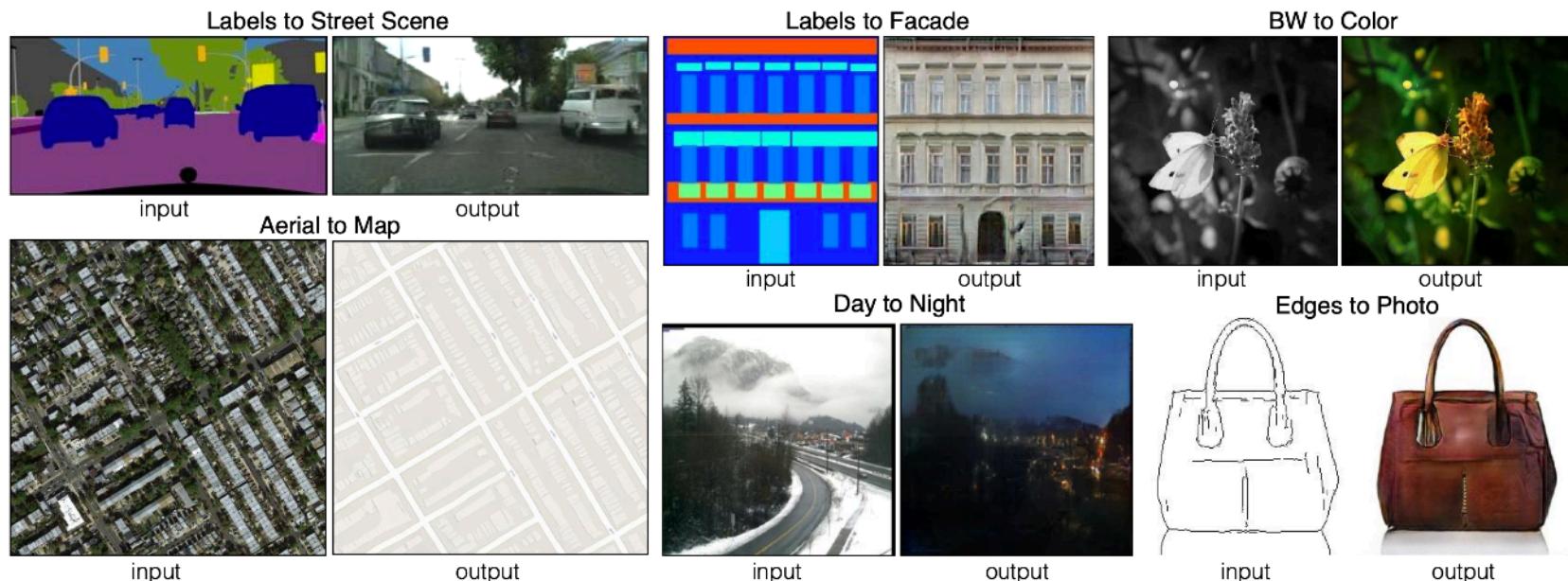


- DCGAN의 discriminator는 conv2d layer로 구성되어 있으며, 점차 크기를 줄여가고 channel을 늘여가다가 마지막에 하나의 스칼라 값을 낼고 이를 real/fake에 대한 확률로 예깁니다.

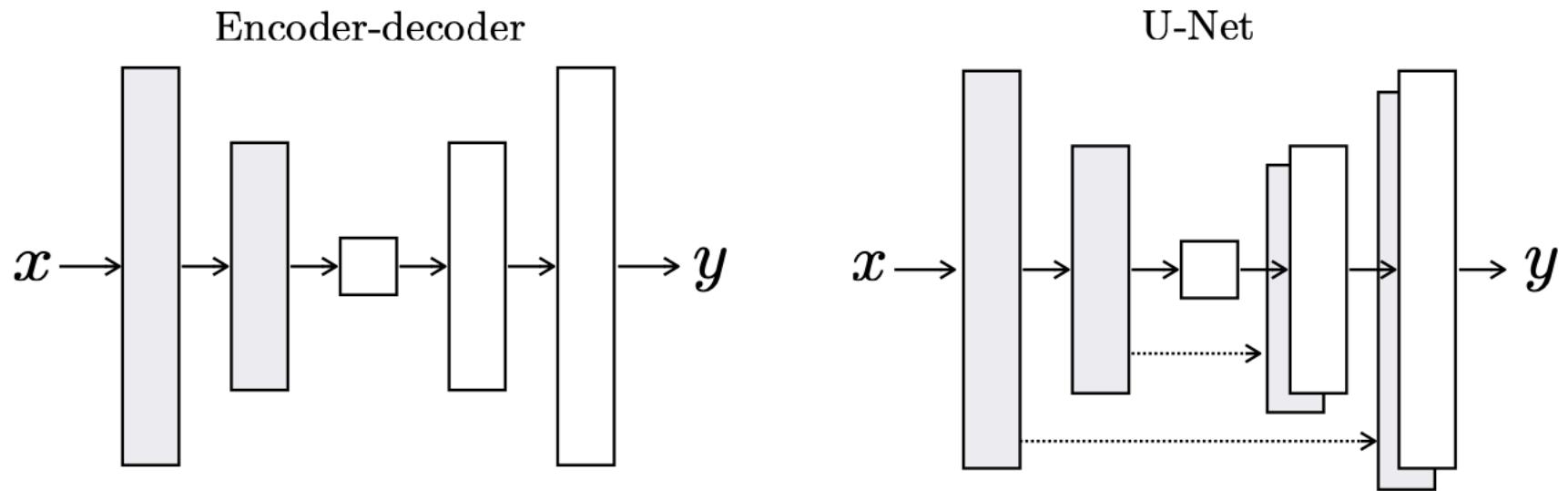
Figure credit: R. Yeh, et al., Semantic Image Inpainting with Perceptual and Contextual Losses

Pix2Pix

- Pix2Pix는 주어지는 조건이 없는 GAN이나, 카테고리로 조건이 주어지는 cGAN과 다르게 그림이 입력으로 주어지고 이와 구조적으로 동일한 그림을 출력하도록 합니다. 하지만 구조안에서 세부적인 사항은 차이가 납니다.
- 예를들면 건물 스케치를 입력하면 건물 사진을 출력하고, 흑백사진을 입력하면 컬러사진을 출력하고, 낮에 찍은 사진을 입력하면 밤에 찍은 사진을 출력하는 식입니다.
- Trainset으로 paired data(입력/출력쌍)이 요구되므로 호랑이 사진을 고양이로 바꾼다거나, 과거사진을 현재사진으로 바꾸는 등의 작업은 힘들다는 한계점이 있습니다.



- Pix2Pix의 구조로 encoder-decoder나 U-net을 사용합니다.
- encoder-decoder 구조는 입력을 압축하는 encoder와 압축된 데이터를 다시 복원하는 decoder로 이루어져 있습니다. 압축되는 과정에서 정보 손실이 일어나는 단점이 있습니다.
- U-net 구조는 encoder-decoder의 단점을 극복하기 위해 제안 되었습니다. Encoder의 매 layer의 정보가 decoder의 매 layer에 전달되어 정보 손실을 피할 수 있습니다.

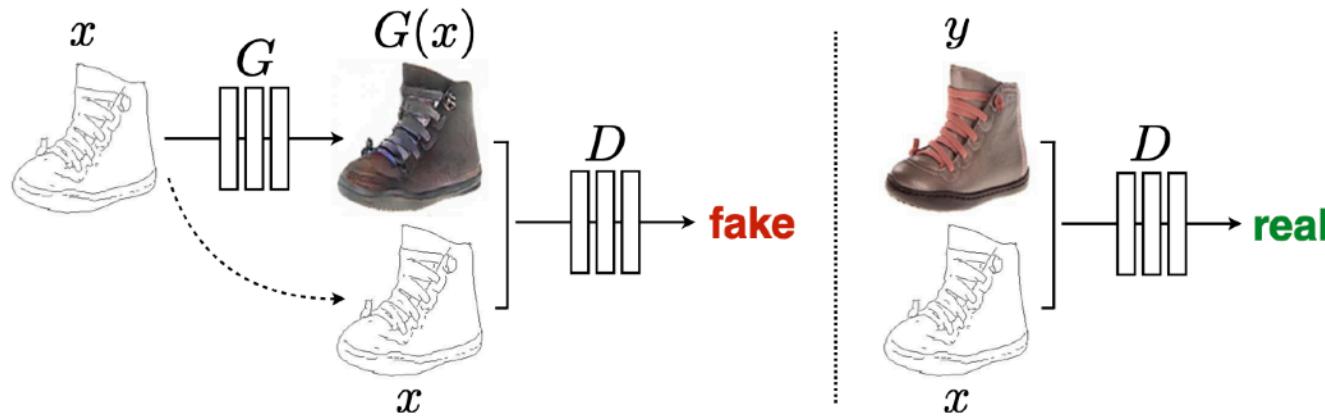


- Pix2Pix에서는 GAN loss와 L1 loss가 동시에 사용됩니다.
- Target이 존재하므로 L1 loss를 사용하여 prediction과 target을 비교하고 최대한 같아지도록 만들 수 있습니다. 하지만 Pix2Pix 작업은 one-to-many 변환이므로 하나의 target에만 같아지도록 만들기 힘듭니다. 따라서 L1 loss만 쓰는 경우 결과물이 blurry한 형태로 나타납니다.
- GAN loss는 blurry한 현상을 완화해주는 데 기여합니다. GAN loss는 prediction이 주어진 하나의 target에 가깝지는 않더라도 다른 real data들과 비슷해지도록 만들기 때문에 blurry함을 없애줄 수 있습니다.

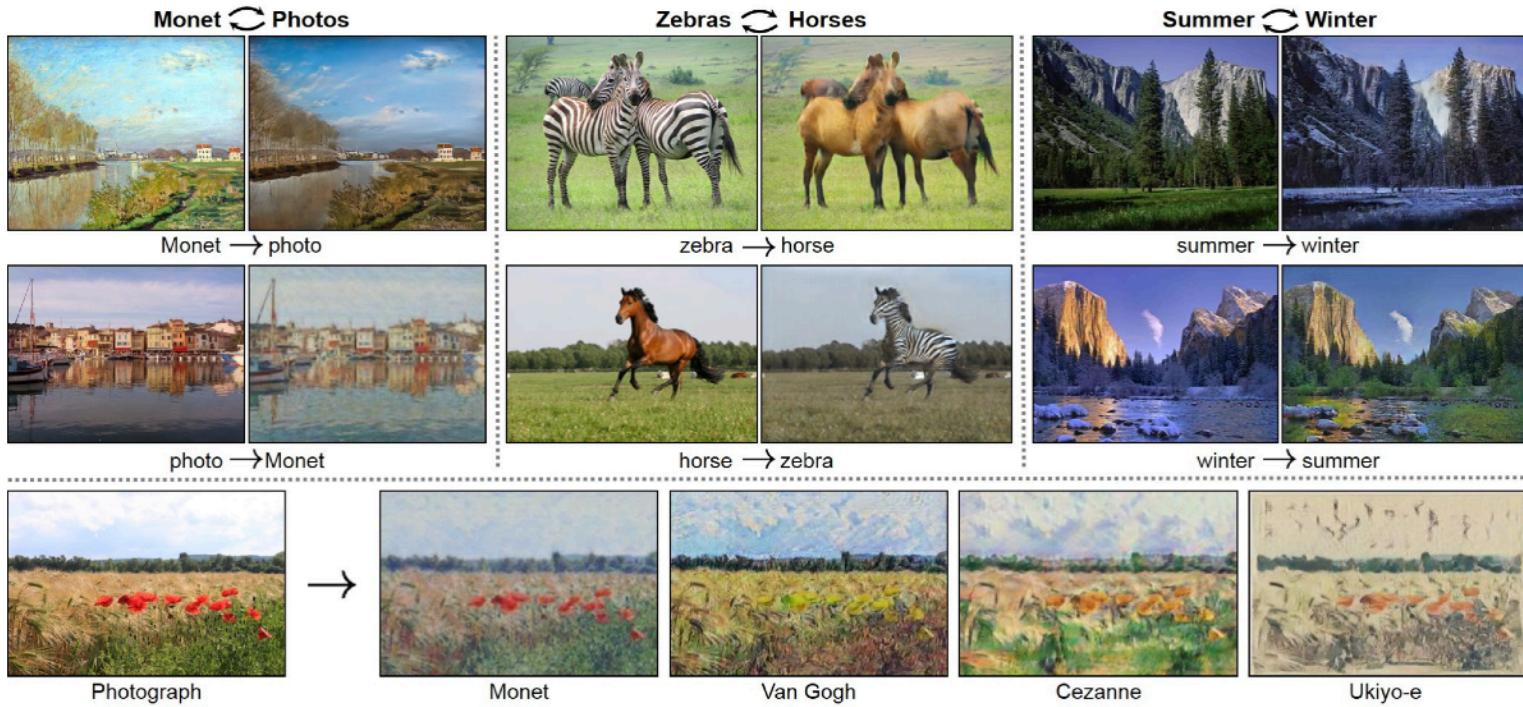
$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

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

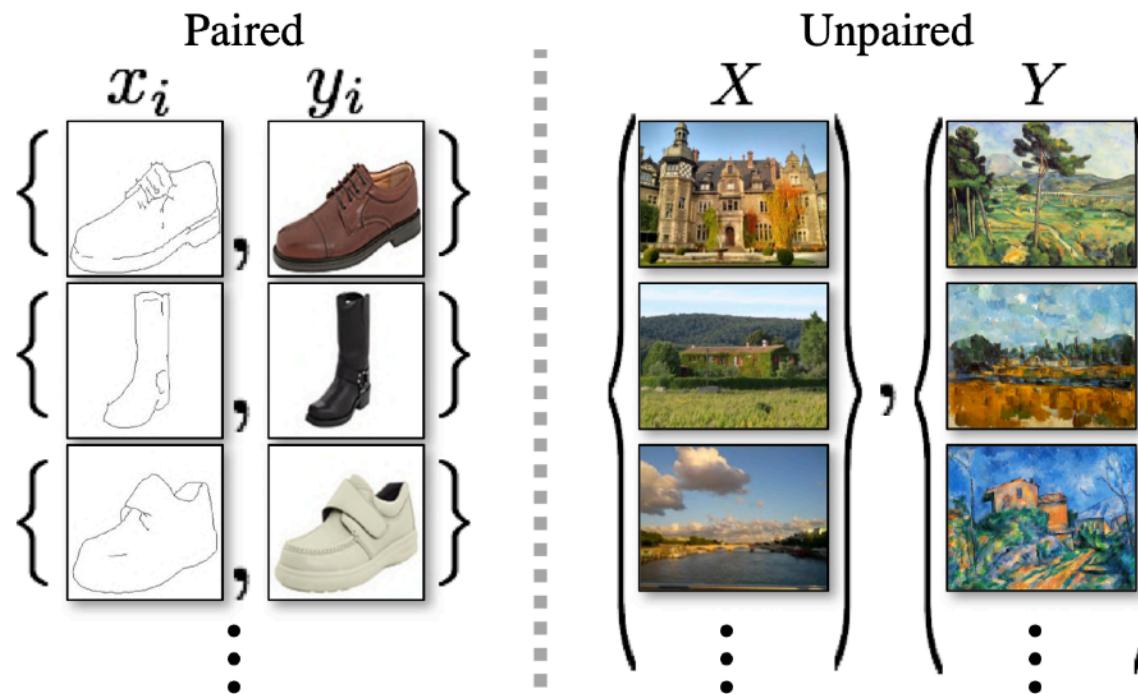
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z} [\|y - G(x, z)\|_1]$$



CycleGAN



- CycleGAN은 Pix2Pix가 paired data를 요구한다는 한계를 극복하기 위해 제안되었습니다.
- CycleGAN 역시 입력과 출력에 해당하는 데이터들이 필요하지만 구조적으로 꼭 맞는 paired data가 필요하지는 않습니다. 단지, 입력 도메인에 속하는 데이터들과 출력 도메인에 속하는 데이터들이 요구됩니다.
- CycleGAN은 cycle consistency loss라는 방법을 통해서 학습됩니다.

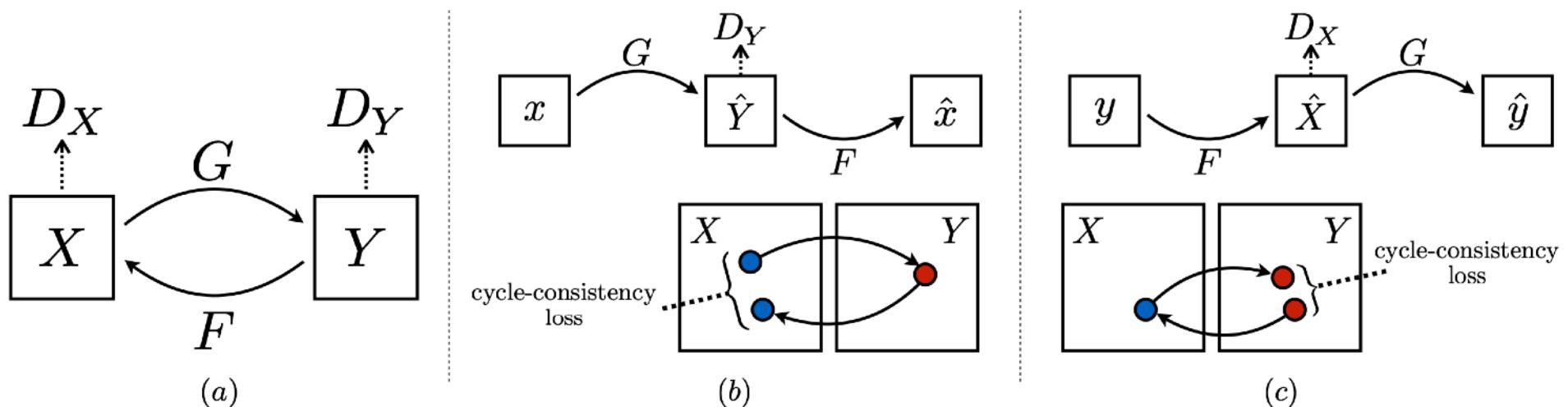


- Pix2Pix는 구조적으로 일치하는 paired data가 필요했지만, CycleGAN은 구조적으로 일치하지 않는 unpaired data 집합들로도 트레이닝이 가능합니다.

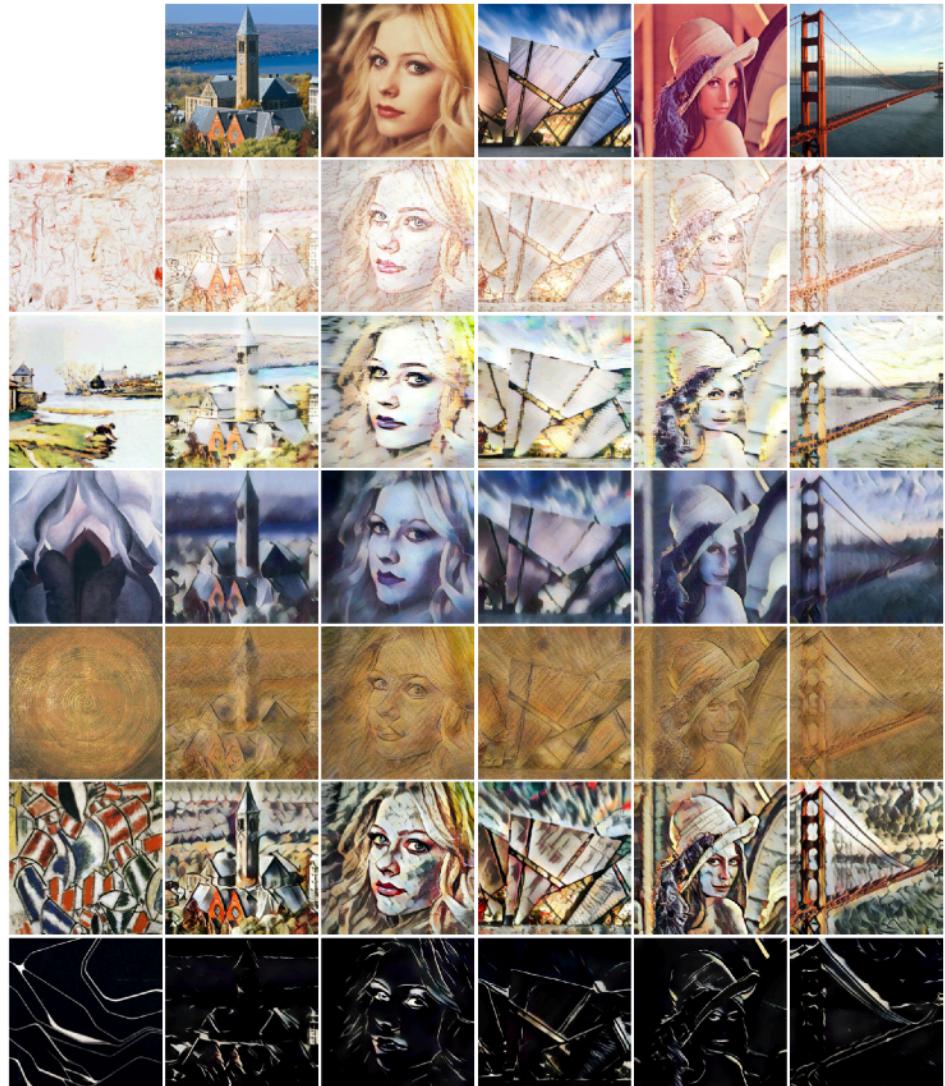
CycleGAN

- CycleGAN은 4개의 네트워크로 구성되어 있습니다.
 Generator G : 도메인 X 의 이미지를 도메인 Y 의 이미지로 변환
 Generator F : 도메인 Y 의 이미지를 도메인 X 의 이미지로 변환
 Discriminator D_X : 도메인 X 의 이미지인지 아닌지를 판별
 Discriminator D_Y : 도메인 Y 의 이미지인지 아닌지를 판별

- cycle-consistency loss는 generator에 의해서 다른 도메인으로 변환된 이미지가 구조적으로 원본과 같게하기 위해 사용합니다.
- 도메인 X 의 이미지 x 를 generator G 에 의해 도메인 Y 의 이미지 \hat{Y} 로 변환합니다. 이를 다시 generator F 에 의해 도메인 X 의 이미지 \hat{x} 로 변환하고 원본 이미지 x 와 L1 loss를 취해 같아지도록 만듭니다.
- 도메인 Y 로부터 변환되는 경우도 마찬가지의 작업을 수행합니다.



Style Transfer

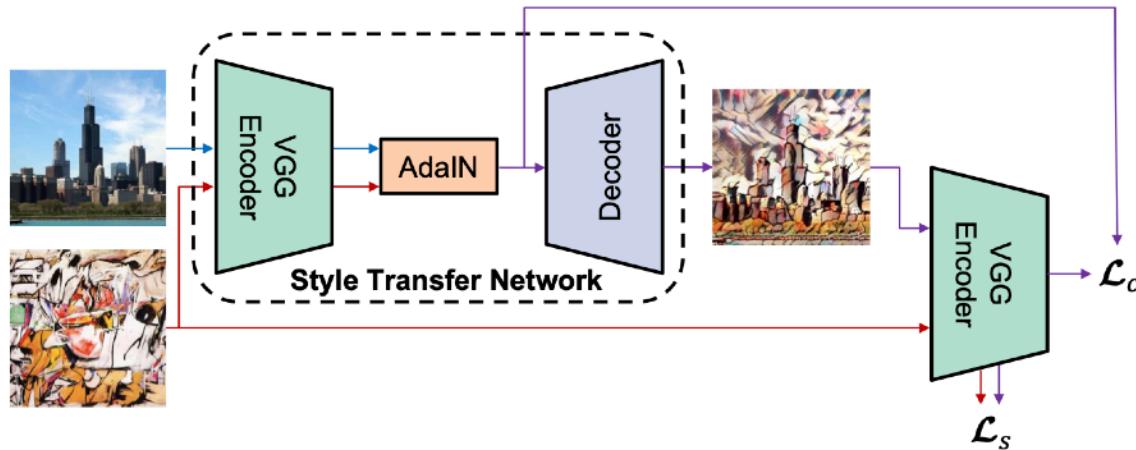


- Style Transfer는 두 장의 이미지를 입력으로 받아 하나는 content, 하나는 style을 관여하는데 사용하여 이미지를 변환하는 작업입니다.
- 출력으로 나온 이미지는 content 이미지와 구조적으로 동일하며, style 이미지와는 텍스쳐, 색깔이 비슷합니다.
- 초기에는 (Gatys 외, 2016) 이미지를 변환하는데 오랜시간이 걸리거나 트레이닝 때 주어졌었던 style로만 변환이 되었습니다.
- 이후에 이러한 단점을 보완한 모델들이 제안되었고, 그 중 하나인 adaptive instance normalization은 real-time으로, 주어진 어떠한 그림의 style로도 변환을 할 수 있는 모델입니다.

Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization
<https://arxiv.org/abs/1703.06868>

Style Transfer

- Adaptive normalization을 이용한 style transfer는 아래 그림과 같이 pretrained VGG encoder와 AdaIN, trainable decoder로 구성됩니다.
- Loss는 content loss와 style loss로 이루어져 있습니다.
- Content loss는 content 이미지의 구조를 유지하는 역할을 하며, style loss는 style 이미지의 텍스쳐를 가져오는 역할을 합니다.



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

Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization
<https://arxiv.org/abs/1703.06868>

모두 모두 파이팅!!