Generative Adversarial Networks - 2

2021
Multicampus

Il Gu Yi
ModuLabs, Research Scientist
Soochul Park
GaudioLab, Research Scientist

v.1.2

Taxonomy of Generative Models

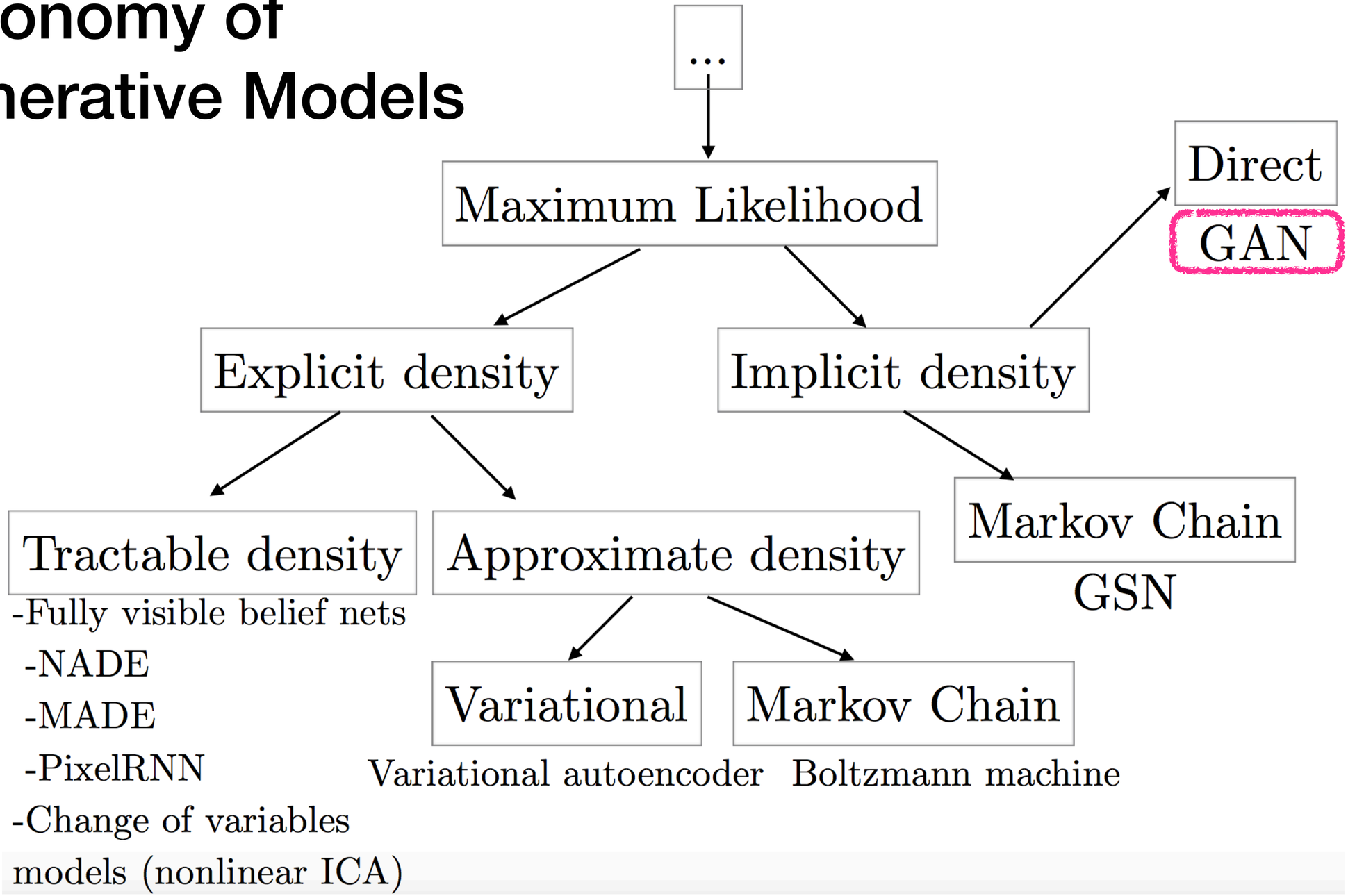


Figure credit: Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks

Wasserstein GAN (WGAN)

WGAN

- WGAN은 Wasserstein GAN 논문을 통해 제안되었습니다.
- 기존은 GAN은 Jensson-shanon divergence를 통해서 generator가 만들어내는 distribution p_g이 data distribution p_{data}에 가까이 가게 됩니다.
- 이러한 방식은 트레이닝 초기, 즉 p_g와 p_{data}가 둘 다에 포함되는 support vector 들이 없을 때, 트레이닝이 잘 이루어지지 않는다는 단점이 있습니다.
- -이를 해결하기 위해 earth mover's distance라는 개념을 도입하고, 이와 동치인 Wasserstein distance을 GAN을 트레이닝하는데 loss function으로 사용합니다.
- -현재 WGAN은 많은 모델들에서 표준으로 사용되고 있으며, 이에 gradient penality등 여러 regularization을 덧붙여 더욱 안정되게 트레이닝을 시킬 수 있습니다.

WGAN [Recap of GANs]

$$\min_G \max_D V(D,G) = \min_G V(D^*,G)$$ for fixed optimal $\mathcal{D}$

$$V(D^*,G) = -\log 4 + 2 \cdot JSD(p_{\text{data}} || p_g)$$

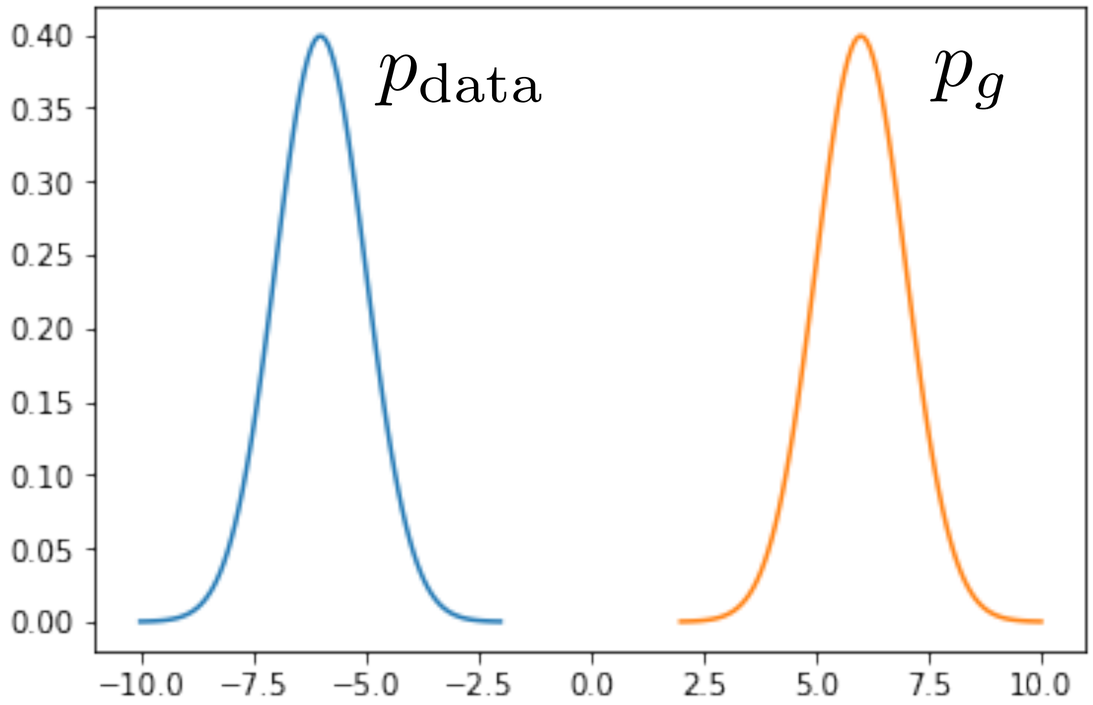
What is JSD between p_{data} and p_g ?

$$JSD(p_{data} \| p_g)$$

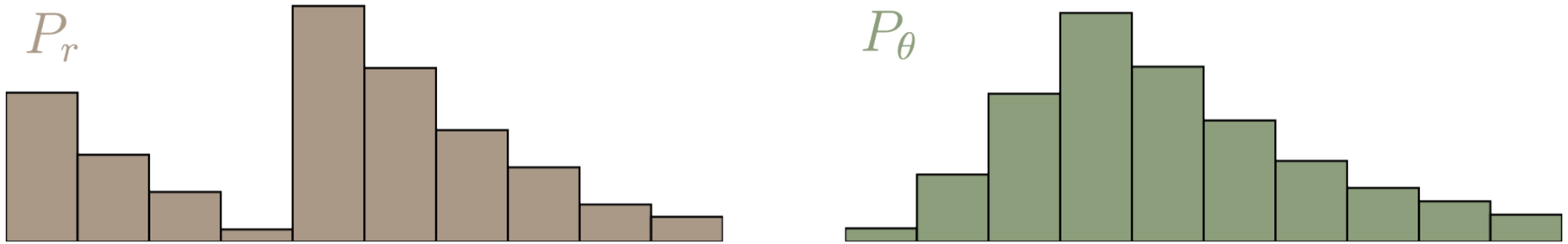
$$= \frac{1}{2} D_{KL}(p_{data} \| \frac{p_{data} + p_g}{2}) + \frac{1}{2} D_{KL}(p_g \| \frac{p_{data} + p_g}{2})$$

$$= \frac{1}{2} \int p_{data}(x) \log \frac{p_{data}(x)}{\frac{p_{data}(x) + p_g(x)}{2}} dx + \frac{1}{2} \int p_g(x) \log \frac{p_g(x)}{\frac{p_{data}(x) + p_g(x)}{2}} dx$$

$$= \frac{1}{2} \log 2 + \frac{1}{2} \log 2 = 2$$



Earth Mover's Distance



Consider the distribution as if it were a block and
move the blocks to make the two distributions the same
"Minimal effort"

Figure credit: https://vincentherrmann.github.io/blog/wasserstein/

Earth Mover's Distance

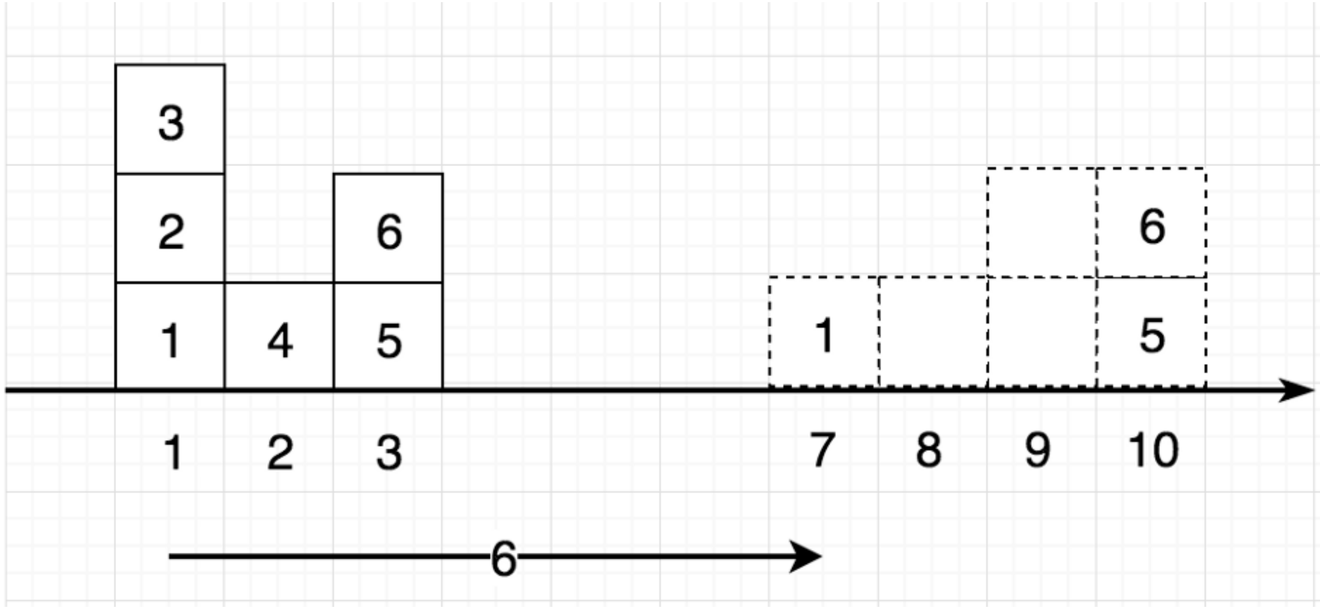
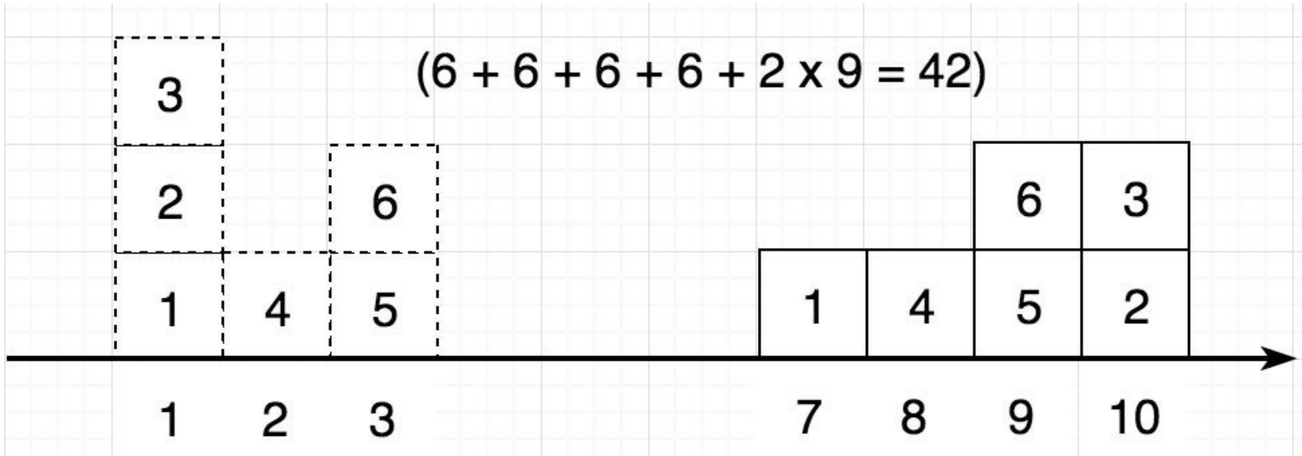


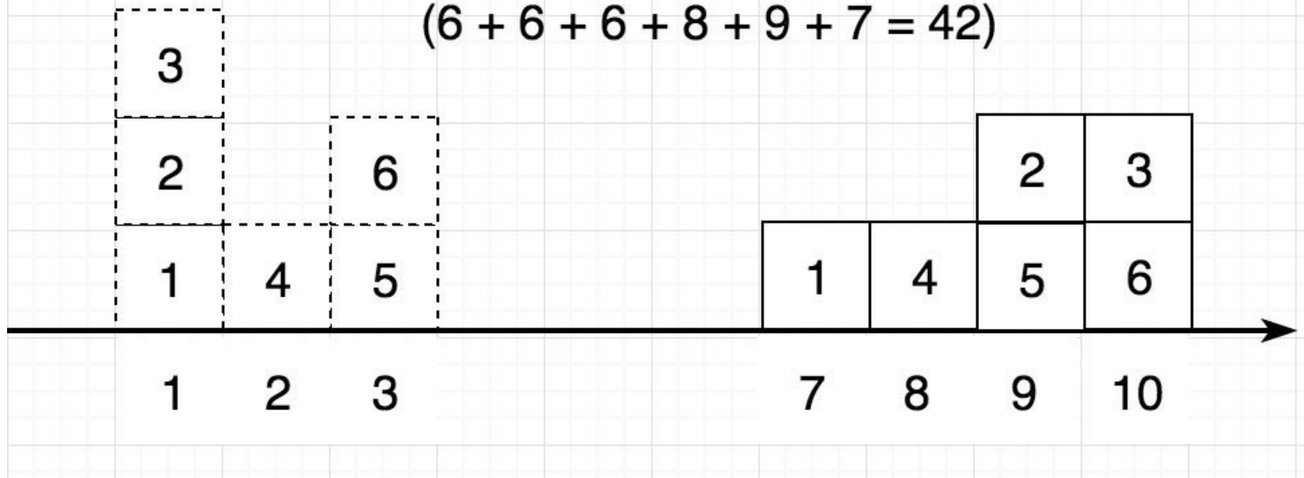
Figure credit: https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Earth Mover's Distance



γ_1

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0



γ_2

	7	8	9	10
1	1	0	1	1
2	0	1	0	0
3	0	0	1	1

Figure credit: https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Earth Mover's Distance

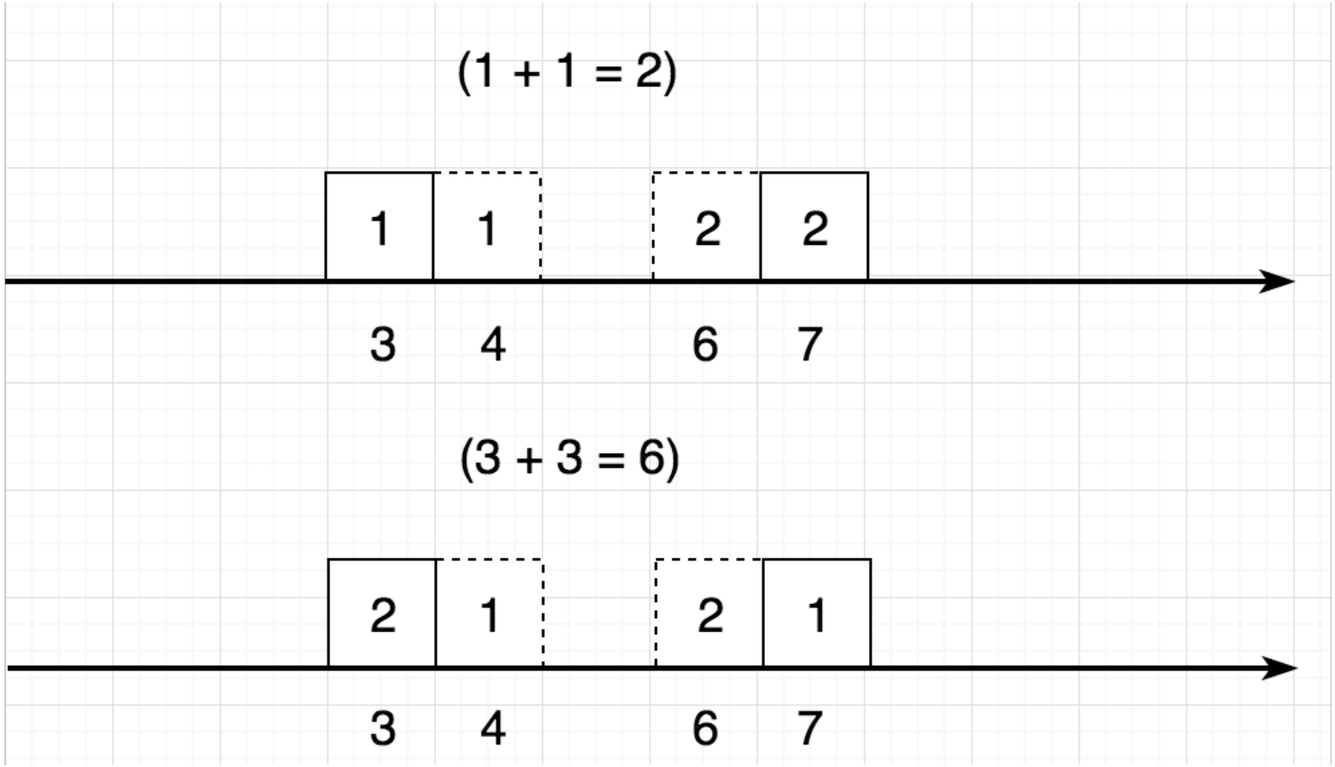
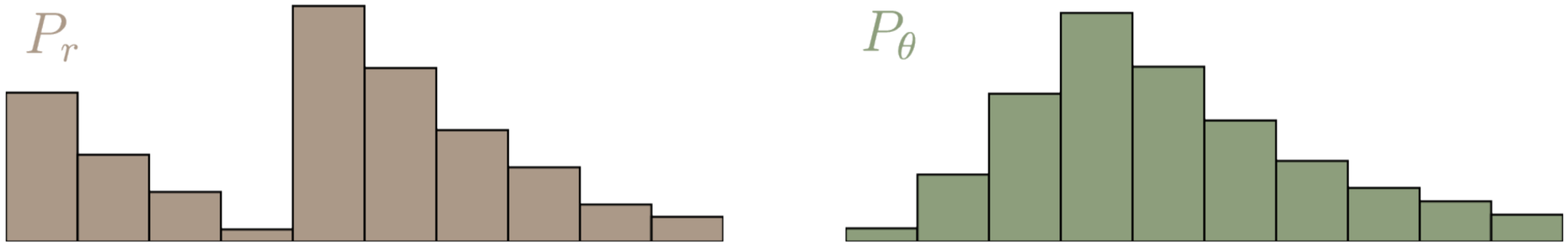


Figure credit: https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Earth Mover's Distance



Consider the distribution as if it were a block and move the blocks to make the two distributions the same "Minimal effort"

$$\text{EMD}(p_{\text{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[||x - y||\right]$$

EM distance: **cost** of the optimal transport plan

Figure credit: https://vincentherrmann.github.io/blog/wasserstein/

Earth Mover's Distance

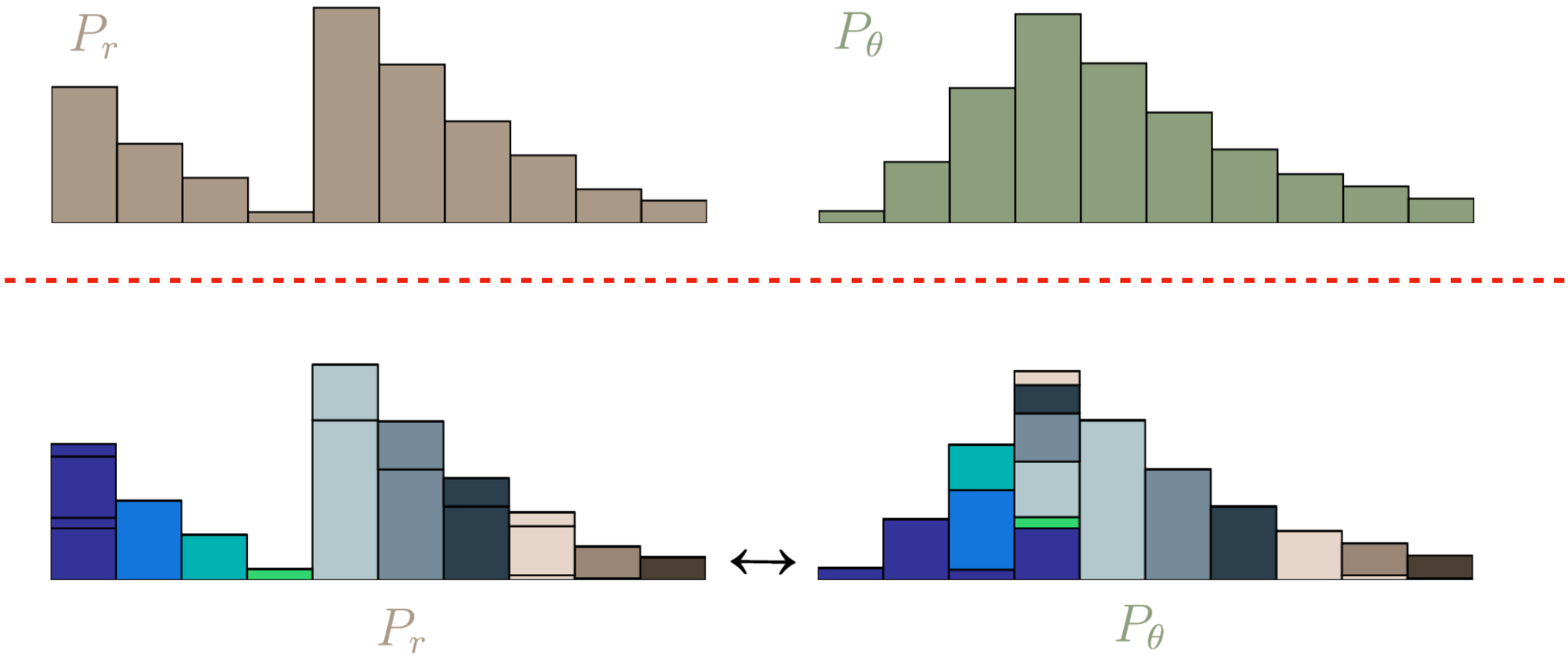


Figure credit: https://vincentherrmann.github.io/blog/wasserstein/

Kantorovich-Rubinstein Duality

$$\mathrm{EMD}(p_{\mathrm{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\mathrm{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[||x - y||\right]$$

$$W(p_{\mathrm{data}}, p_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim p_{\mathrm{data}}}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]$$

1-Lipschitz Functions

$$\mathrm{EMD}(p_{\mathrm{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\mathrm{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[||x - y||\right]$$

$$W(p_{\mathrm{data}}, p_g) = \sup_{||f||_L \leq 1} \mathbb{E}_{x\sim p_{\mathrm{data}}}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]$$

1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$

$$\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim p_{\mathrm{data}}}[f_w(x)] - \mathbb{E}_{z\sim p_z}[f_w(G(z))]$$

How to implement f as 1-Lipschitz function?

In the paper, f is called a critic, not a discriminator

1-Lipschitz Function Implementation

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

clipping appropriately

Ref.: Martin Arjovsky, et al., Wasserstein GAN

Optimal Discriminator

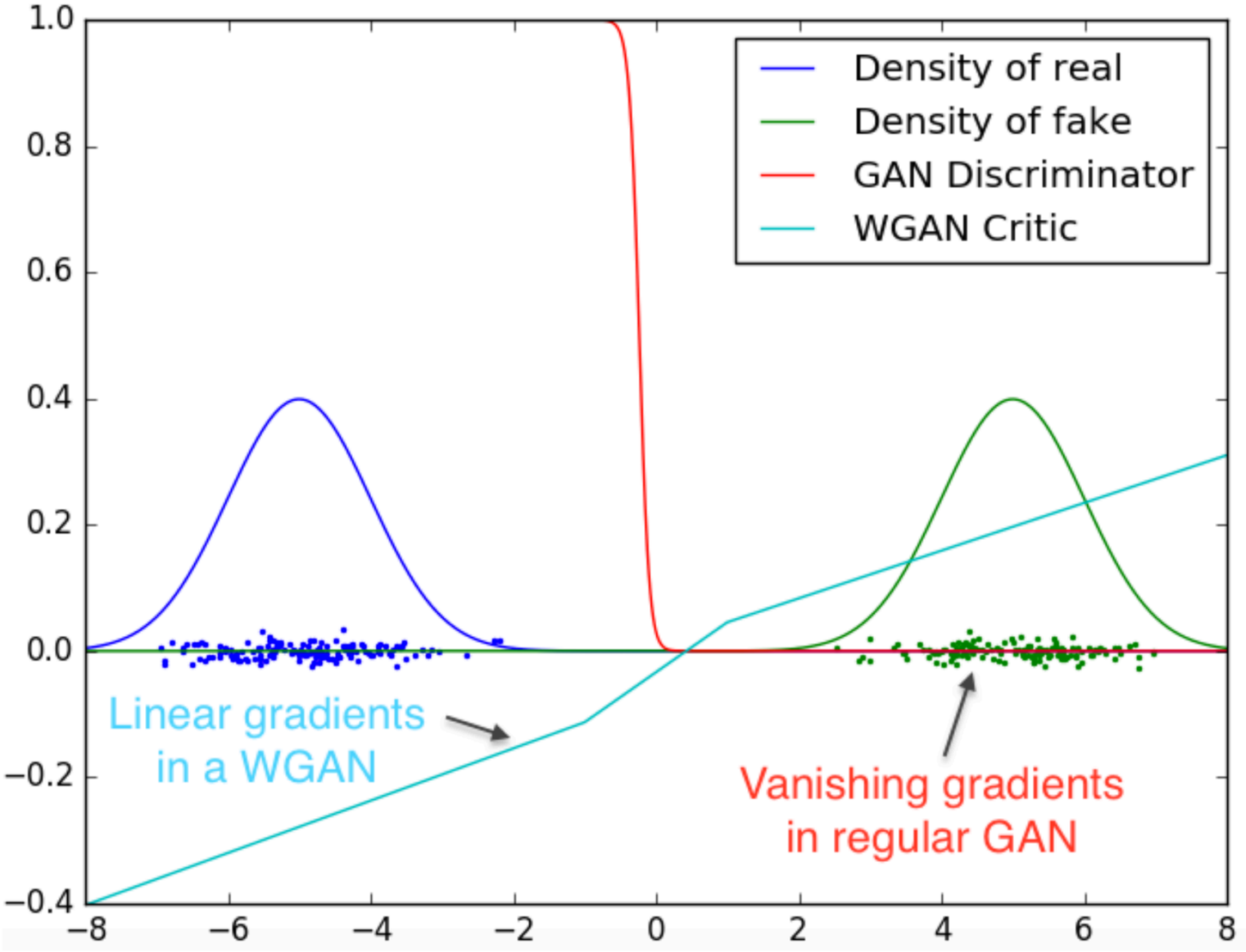
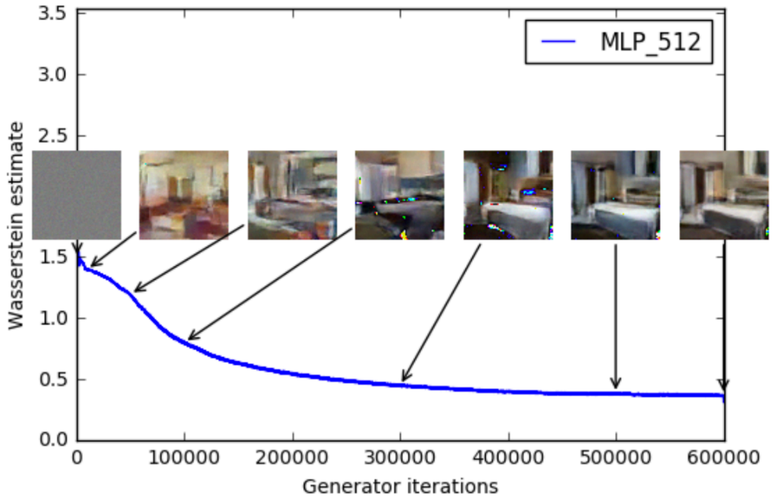


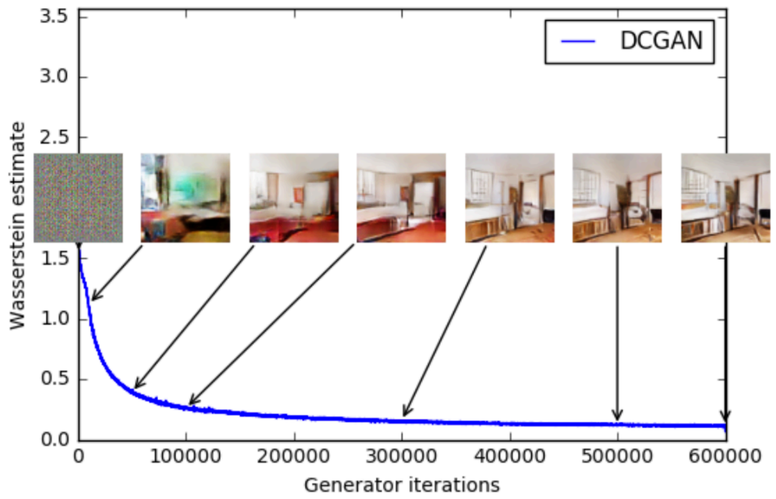
Figure credit: Martin Arjovsky, et al., Wasserstein GAN

Learning Curve for EM Distance

Wasserstein estimate
≈ loss D

$$\mathbb{E}_{x\sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{z\sim p_z}[f_w(G(z))]$$





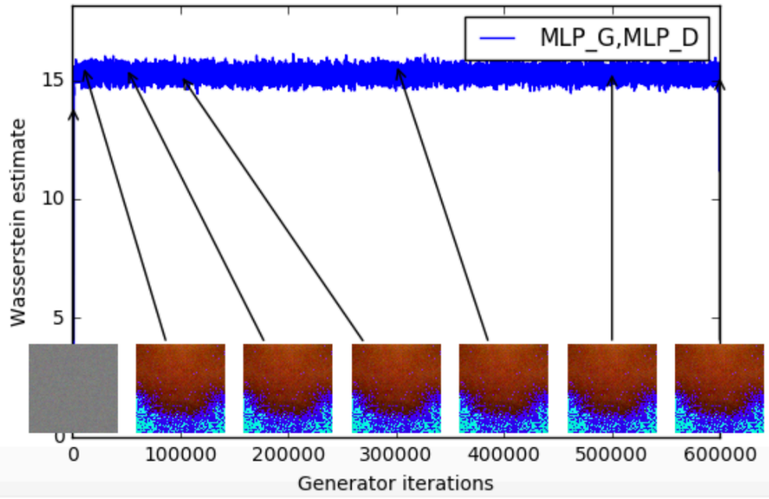
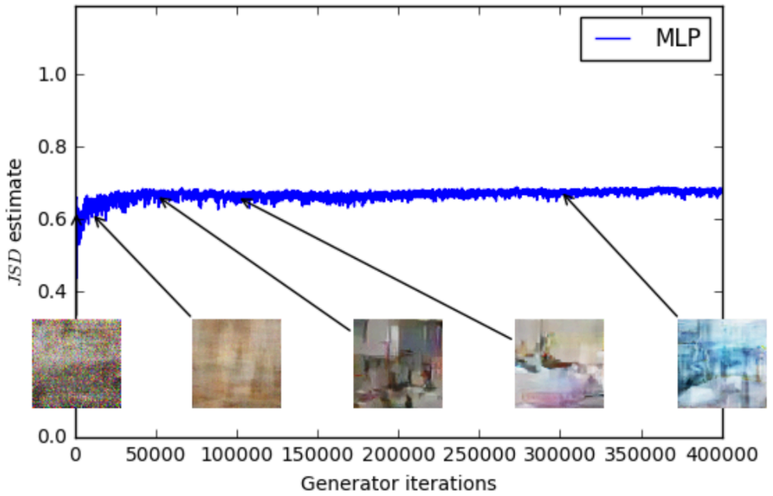


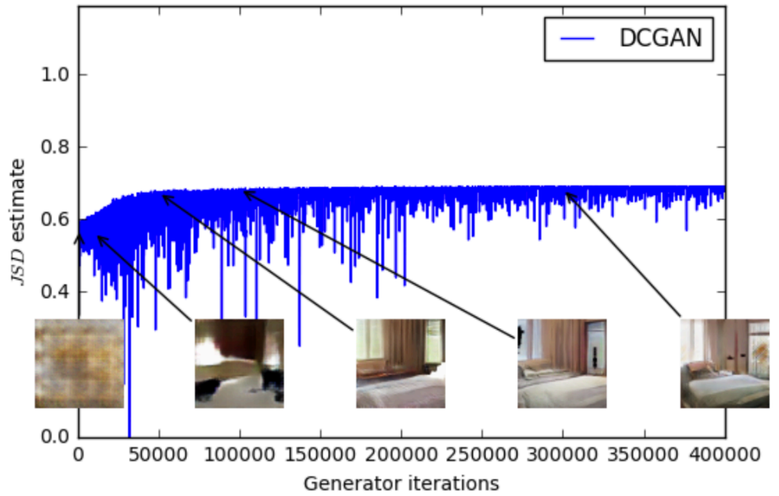
Figure credit: Martin Arjovsky, et al., Wasserstein GAN

Learning Curve for JSD

JSD estimate
≈ loss of Discriminator

$$JSD(p_{\text{data}}, p_g) = \frac{1}{2} J^{(D)} + \log 2$$





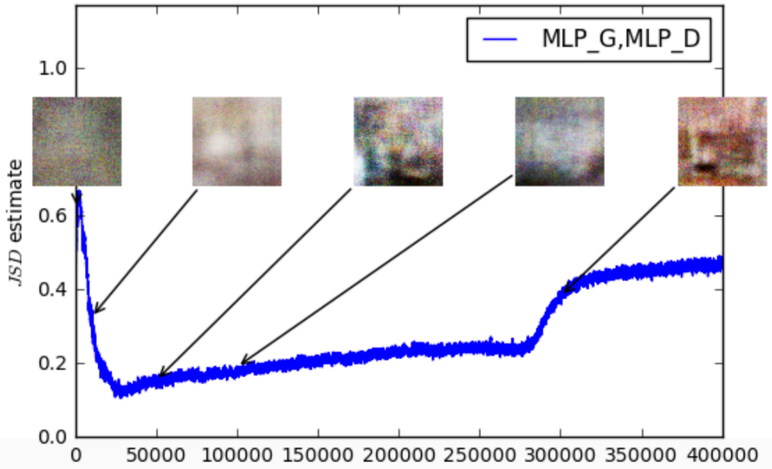


Figure credit: Martin Arjovsky, et al., Wasserstein GAN

WGAN Results

Figure 5: Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

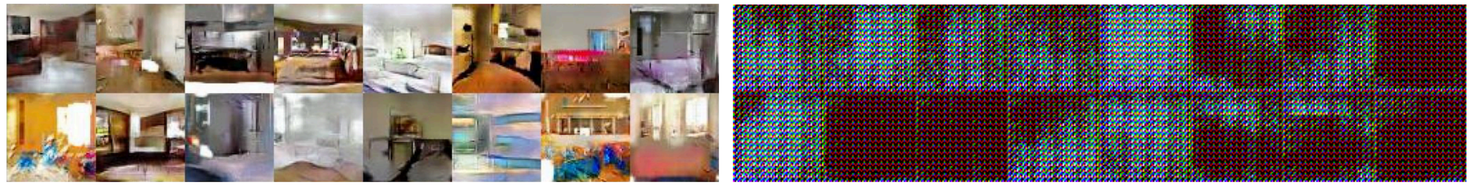


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]).

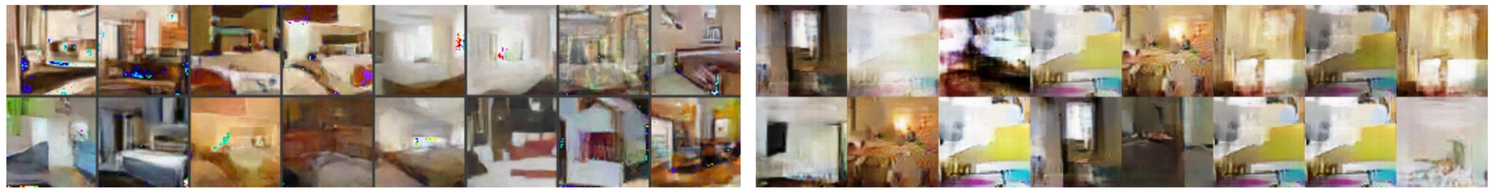


Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a

Figure credit: Martin Arjovsky, et al., Wasserstein GAN

Wasserstein GAN Gradient Penalty (WGAN-GP)

WGAN-GP

- WGAN-GP는 Improved Training of Wasserstein GANs 논문을 통해 제안되었습니다.
- WGAN은 discriminator를 1-lipschitz function으로 만들기 위해서 weight의 크기를 제한했습니다. 이러한 방식은 weight가 모두 boundary에 해당하는 값을 가져 좋지 않습니다.
- WGAN-GP에서는 discriminator를 1-lipschitz function으로 만들기 위해 discriminator의 gradient에 제약을 가합니다.
- Real 이미지와 fake 이미지 간에 linear interpolation을 통해 중간 이미지를 얻고 discriminator에 넣어 gradient를 얻습니다.
- Gradient의 크기가 1이 되도록 penalty term을 loss에 포함시켜, discriminator가 1-lipschitz function이 되도록 만듭니다.

Gulrajani, Ishaan, et al. "Improved training of wasserstein gans." arXiv preprint arXiv:1704.00028 (2017).

Is Right Clipping Method for Implementing 1-Lipschitz Functions?

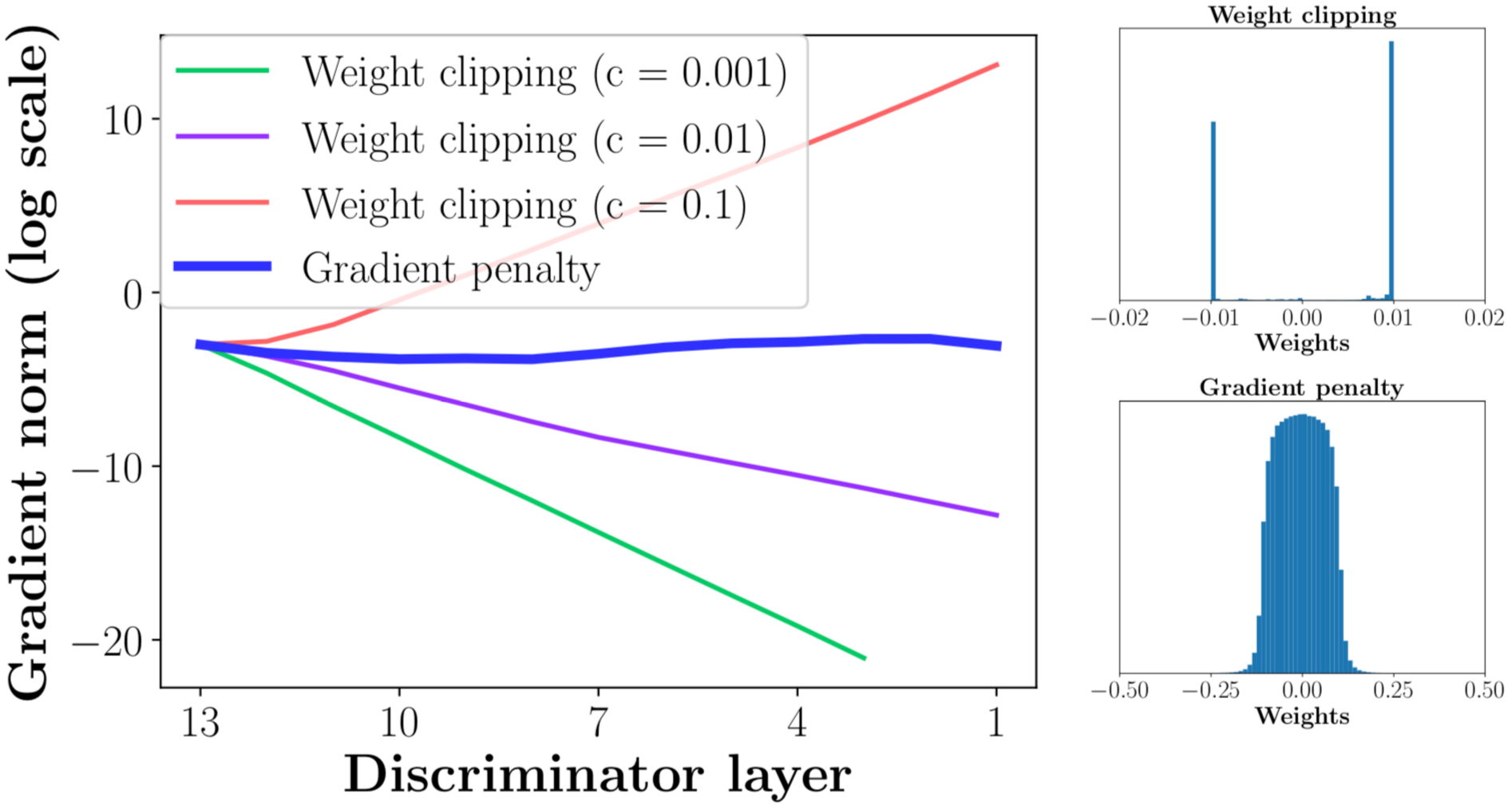


Figure credit: Ishaan Gulrajani, et al., Improved Training of Wasserstein GANs

Proposition & Corollary

Proposition 1. *Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distributions in $\mathcal{X}$, a compact metric space. Then, there is a 1-Lipschitz function f^* which is the optimal solution of* $\max_{\|f\|_L \leq 1} \mathbb{E}_{y \sim \mathbb{P}_r}[f(y)] - \mathbb{E}_{x \sim \mathbb{P}_g}[f(x)]$. *Let π be the optimal coupling between $\mathbb{P}_r$ and $\mathbb{P}_g$, defined as the minimizer of:* $W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\pi \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\pi}[\|x - y\|]$ *where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of joint distributions $\pi(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively. Then, if f^* is differentiable[‡], $\pi(x = y) = 0$[§], and $x_t = tx + (1-t)y$ with $0 \leq t \leq 1$, it holds that* $\mathbb{P}_{(x,y)\sim\pi}\left[\nabla f^*(x_t) = \frac{y - x_t}{\|y - x_t\|}\right] = 1$.

Corollary 1. *f^* has gradient norm 1 almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$.*

Gradient Penalty

$$L = \mathbb{E}_{\tilde{\boldsymbol{x}} \sim P_g}[D(\tilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x} \sim P_{\text{data}}}[D(\boldsymbol{x})] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim P_{\hat{\boldsymbol{x}}}} \left[(||\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})||_2 - 1)^2 \right]$$

where $\hat{\boldsymbol{x}} = \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$

WGAN-GP Results

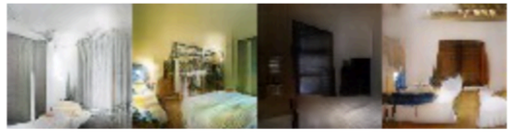
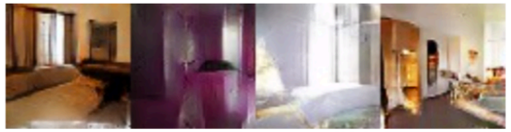
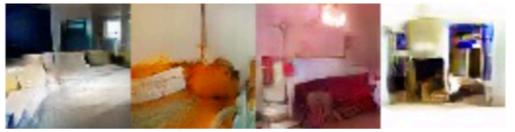
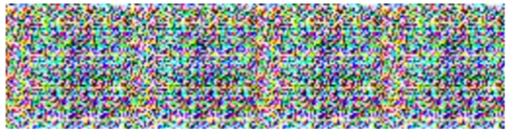
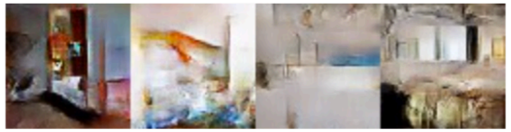
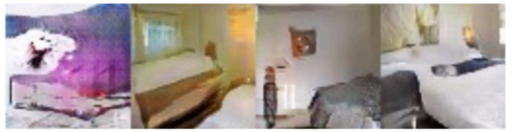
DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)			
			
G: No BN and a constant number of filters, D: DCGAN			
			
G: 4-layer 512-dim ReLU MLP, D: DCGAN			
			
No normalization in either G or D			
			

Figure credit: Ishaan Gulrajani, et al., Improved Training of Wasserstein GANs

Spectral Normalization

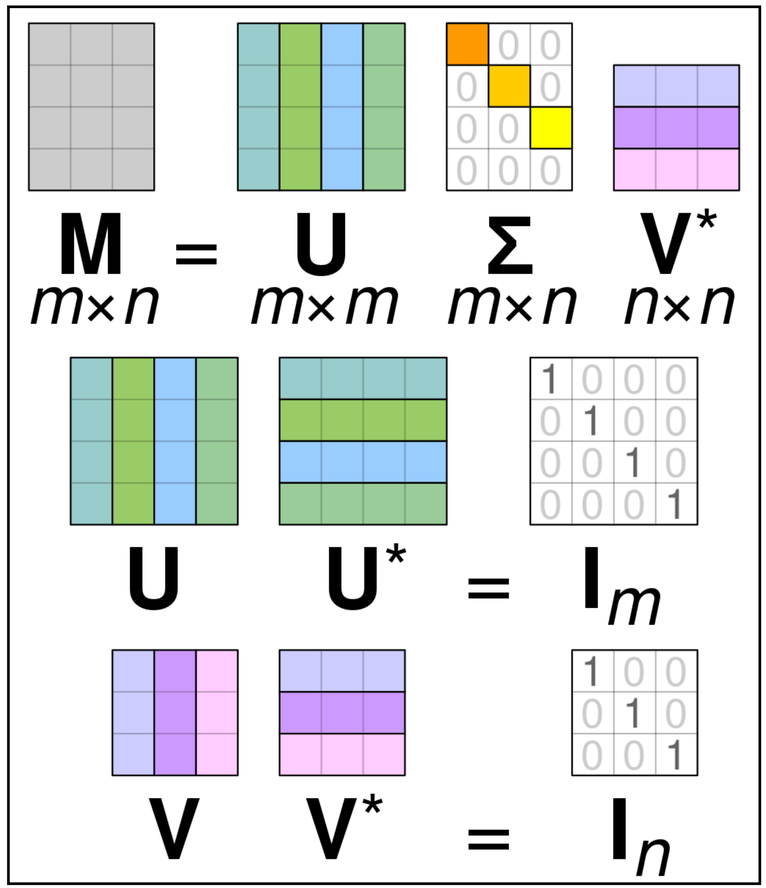
Spectral Normalization

- Spectral normalization 기법은 Spectral normalization for generative adversarial networks 논문에서 제안되었습니다.
- Spectral normalization도 WGAN-GP와 마찬가지로 discriminator를 1-Lipschitz function로 만들기 위한 방법입니다.
- Spectral normalization에서는 discriminator에 사용되는 convolution이나, linear (dense) 레이어의 weight의 크기를 제한하는 방식으로 1-Lipschitz function이 되게 만듭니다.
- Linear function의 Lipschitz norm은 linear function의 matrix의 가장 큰 singular value와 같습니다. 이에 착안해 discriminator를 이루는 각 layer의 weight들을 가장 큰 singular value로 normalization시켜줍니다.
- -Activation인 ReLU, Tanh, sigmoid 등은 1-Lipschitz function을 만족하므로 그대로 사용할 수 있습니다.

Miyato, Takeru, et al. "Spectral normalization for generative adversarial networks." arXiv preprint arXiv:1802.05957 (2018).

Spectral Norm.

Singular Value Decomposition



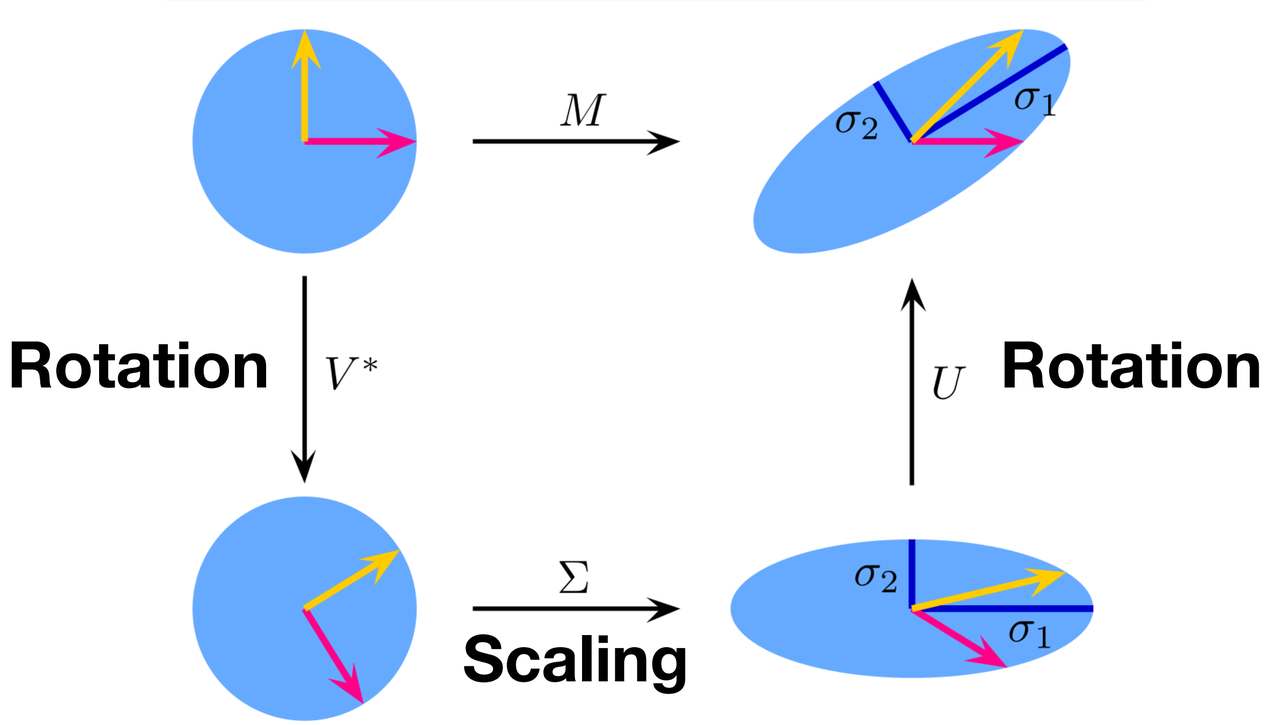
M: an arbitrary $m \times n$ matrix

U: $m \times m$ unitary(orthogonal) matrix
Σ: $m \times n$ diagonal matrix with non-negative real numbers on the diagonal
V: $n \times n$ unitary(orthogonal) matrix

https://en.wikipedia.org/wiki/Singular_value_decomposition

Spectral Norm.

Singular Value Decomposition



$$M = U \cdot \Sigma \cdot V^*$$

https://en.wikipedia.org/wiki/Singular_value_decomposition

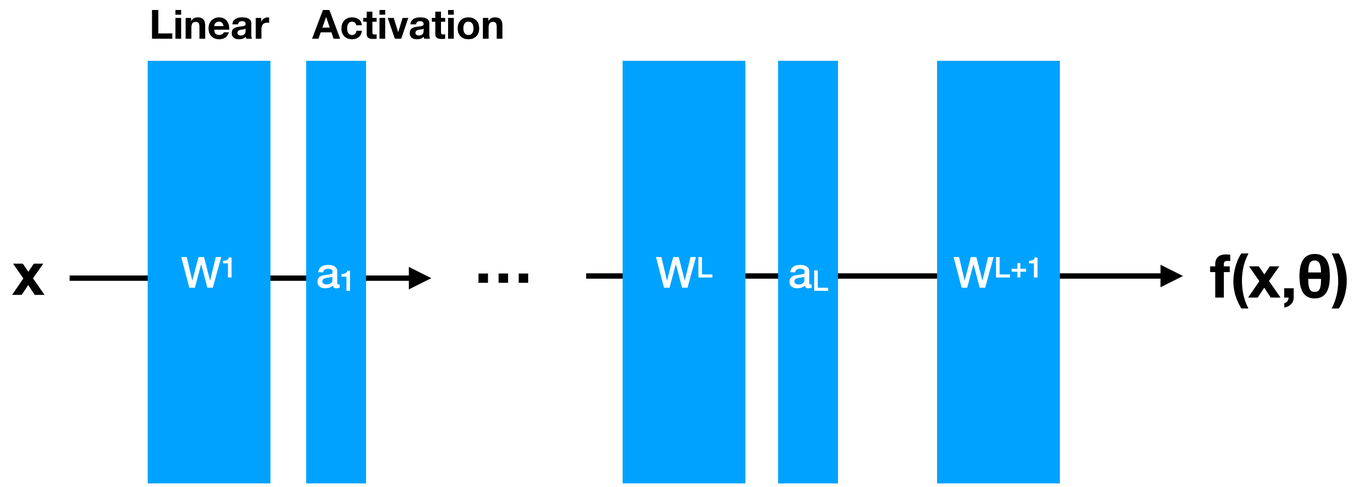
Spectral Norm.

Consider a simple discriminator made of a neural network of the following form, with the input x:

$$f(\boldsymbol{x}, \theta) = W^{L+1} a_L \left(W^L \left(a_{L-1} \left(W^{L-1} \left(\dots a_1 \left(W^1 \boldsymbol{x} \right) \dots \right) \right) \right) \right)$$

where $\theta := \left\{ W^1, \dots, W^L, W^{L+1} \right\}$ is the learning parameters set

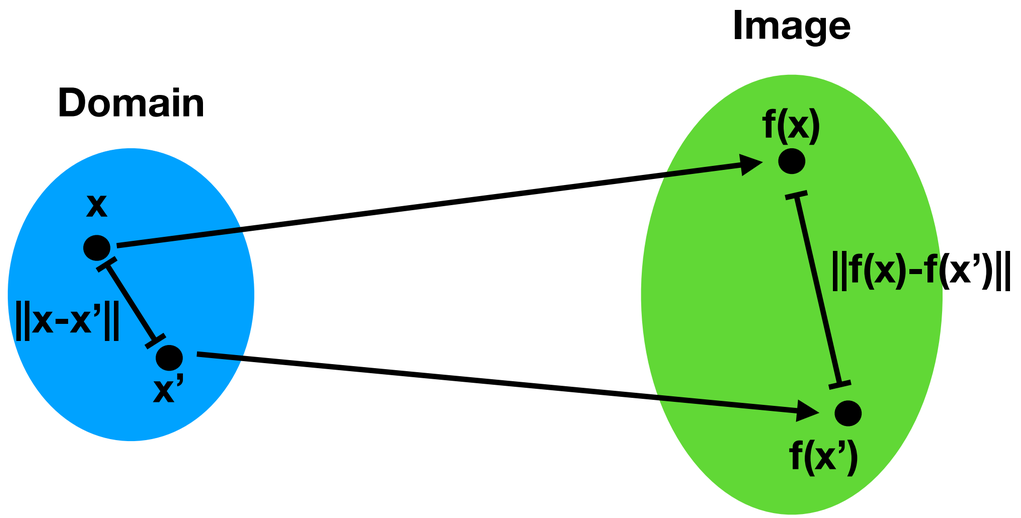
$W^l \in \mathbb{R}^{d_l \times d_{l-1}}, W^{L+1} \in \mathbb{R}^{1 \times d_L}$, and a_l is an element-wise non-linear activation function.



Spectral Norm.

Lipshitz Constant, Lipshitz Norm

$\|f\|_{\text{Lip}}$ the smallest value M such that $\|f(\boldsymbol{x}) - f(\boldsymbol{x}')\|/\|\boldsymbol{x} - \boldsymbol{x}'\| \leq M$ for any $\boldsymbol{x}, \boldsymbol{x}'$, with the norm being the ℓ_2 norm.

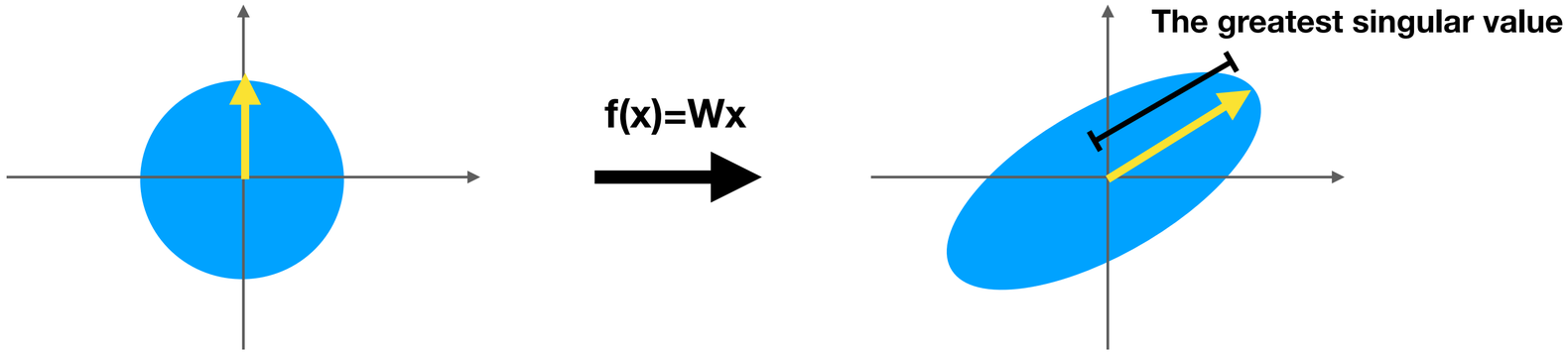


Spectral Norm.

The Lipshitz norm is equal to the greatest singular value if the transform is linear.

$$\sup_{x,x':x-x'\neq 0} \frac{\|f(x)-f(x')\|_2}{\|x-x\|_2} = \max_{h:h\neq 0} \frac{\|Wh\|_2}{\|h\|_2} = \max_{\|h\|_2\leq 1} \|Wh\|_2 = \sigma(W)$$

$$= \frac{\|Wx-Wx'\|_2}{\|x-x\|_2} = \frac{\|W(x-x')\|_2}{\|x-x\|_2} = \frac{\|Wh\|_2}{\|h\|_2}$$



Spectral Norm.

Consider a simple discriminator made of a neural network of the following form, with the input x:

$$f(\boldsymbol{x}, \theta) = W^{L+1} a_L(W^L(a_{L-1}(W^{L-1}(\dots a_1(W^1 \boldsymbol{x}) \dots))))$$

By the inequality $\|g_1 \circ g_2\|_{\text{Lip}} \leq \|g_1\|_{\text{Lip}} \cdot \|g_2\|_{\text{Lip}}$

$$\|f\|_{\text{Lip}} \leq \|(\boldsymbol{h}_L \mapsto W^{L+1}\boldsymbol{h}_L)\|_{\text{Lip}} \cdot \|a_L\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_{L-1} \mapsto W^L \boldsymbol{h}_{L-1})\|_{\text{Lip}}$$

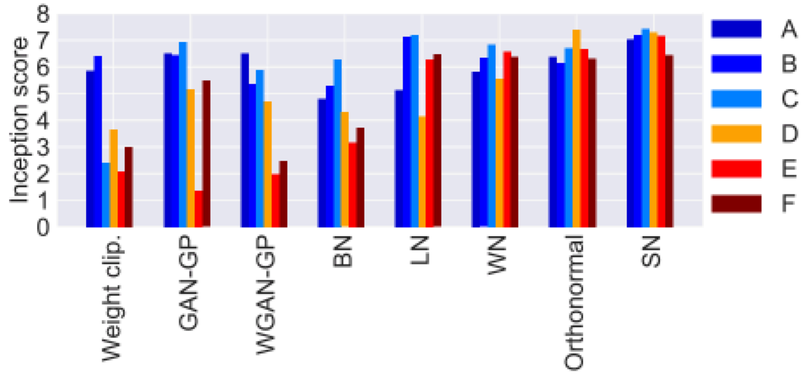
$$\cdots \|a_1\|_{\text{Lip}} \cdot \|(\boldsymbol{h}_0 \mapsto W^1 \boldsymbol{h}_0)\|_{\text{Lip}} = \prod_{l=1}^{L+1} \|(\boldsymbol{h}_{l-1} \mapsto W^l \boldsymbol{h}_{l-1})\|_{\text{Lip}} = \prod_{l=1}^{L+1} \sigma(W^l)$$

Spectral Norm.

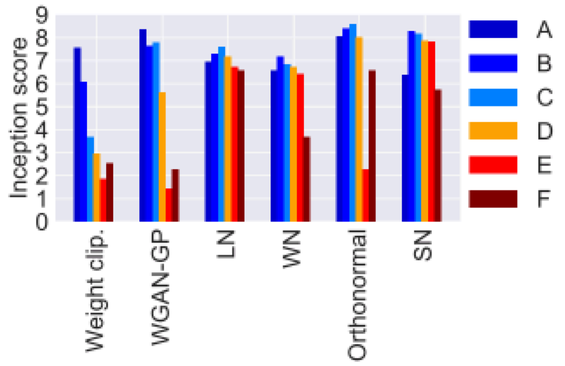
The Lipshitz constraint $\|f\|_{Lip} \leq 1$ will be satisfied
by normalizing the weight matrix W with the greatest singular value $\sigma(W)$.

$$\bar{W}_{SN}(W) = W/\sigma(W)$$

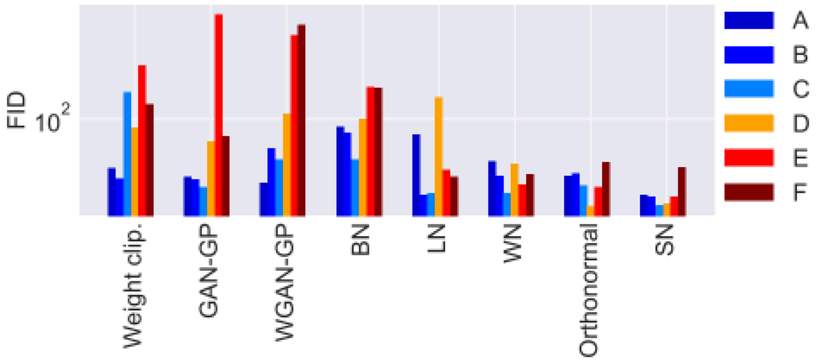
Spectral Norm.



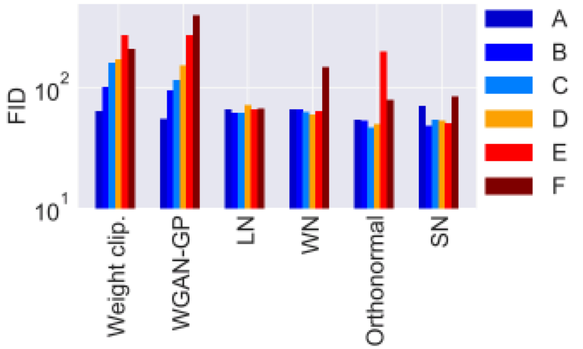
(a) CIFAR-10



(b) STL-10



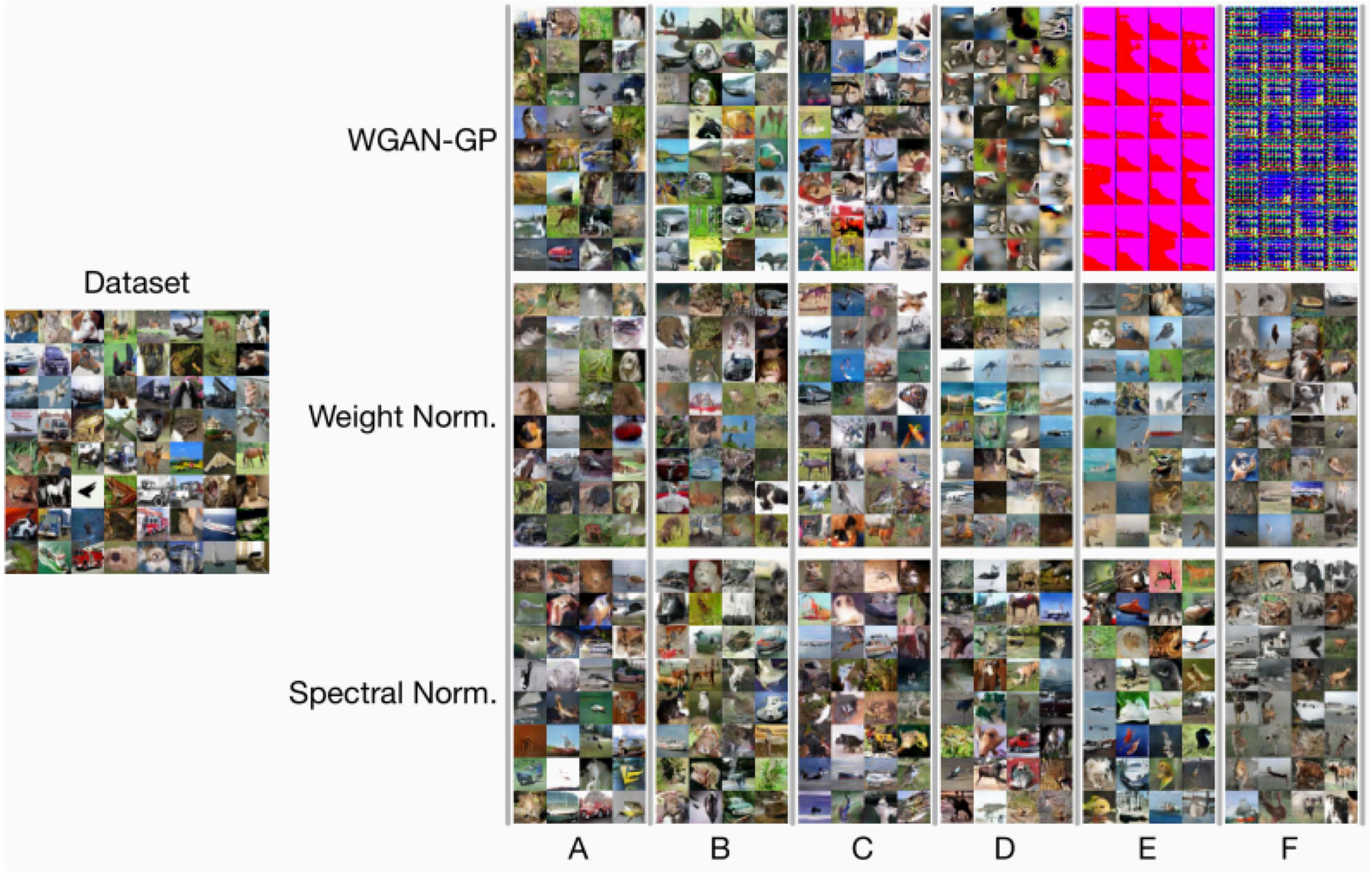
(a) CIFAR-10



(b) STL-10

Figure credit: Takeru Miyato, et al., Spectral Normalization for Generative Adversarial Networks

Spectral Norm.



(a) CIFAR-10

Figure credit: Takeru Miyato, et al., Spectral Normalization for Generative Adversarial Networks

Spectral Norm.

How to compute the greatest singular value efficiently.

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{\boldsymbol{u}}_l \in \mathcal{R}^{d_l}$ for $l = 1, \ldots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l:
  1. Apply power iteration method to a unnormalized weight W^l:

$$\tilde{\boldsymbol{v}}_l \leftarrow (W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l / \|(W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l\|_2 \tag{20}$$

$$\tilde{\boldsymbol{u}}_l \leftarrow W^l \tilde{\boldsymbol{v}}_l / \|W^l \tilde{\boldsymbol{v}}_l\|_2 \tag{21}$$

  2. Calculate $\bar{W}_{\mathrm{SN}}$ with the spectral norm:

$$\bar{W}^l_{\mathrm{SN}}(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{\boldsymbol{u}}_l^{\mathrm{T}} W^l \tilde{\boldsymbol{v}}_l \tag{22}$$

  3. Update W^l with SGD on mini-batch dataset $\mathcal{D}_M$ with a learning rate α:

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}^l_{\mathrm{SN}}(W^l), \mathcal{D}_M) \tag{23}$$

Figure credit: Takeru Miyato, et al., Spectral Normalization for Generative Adversarial Networks

Pix2Pix

Image-to-Image Translation

- Pix2Pix는 주어지는 조건이 없은 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.

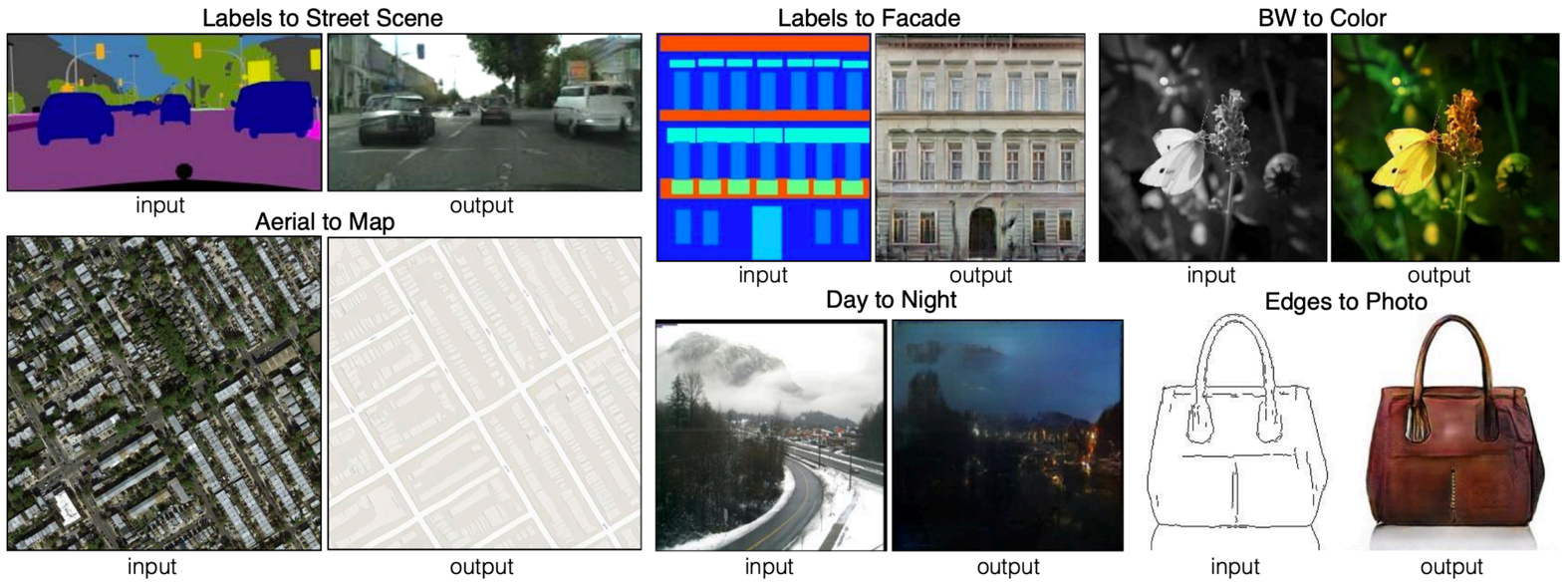


Figure credit: P. Isola, et al., Image-to-Image Translation with Conditional Adversarial Networks

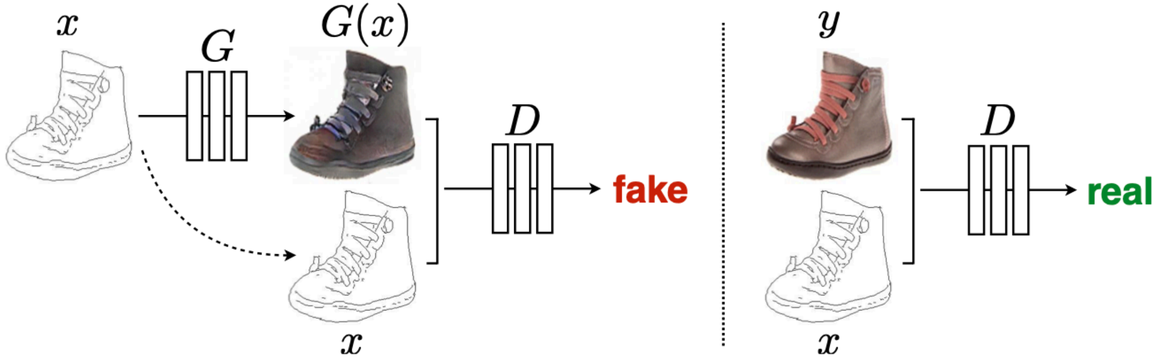
Conditional GAN Loss

- 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 완화해주는데 기여합니다. GAN loss는 prediction이 주어진 하나의 target에 가깝지는 않더라도 다른 real data들과 비슷해지도록 만들기 때문에 blurry함을 없애줄 수 있습니다.

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

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

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



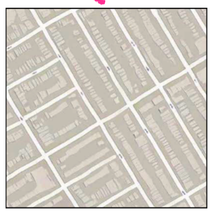
Conditional GAN Loss

Condition

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

Label

Generator



Paired dataset

Conditional GAN Loss

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

Label

Generator

Pixel by pixel L1 loss

Randomness Control

- Pix2Pix don’t need latent codes z vector
  - Learning deterministic mapping from image in domain A to other image in domain B
- In order to add randomness
  - When generator create an image
  - Apply training mode in Dropout and BatchNormalization

Generator Architecture

- 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.

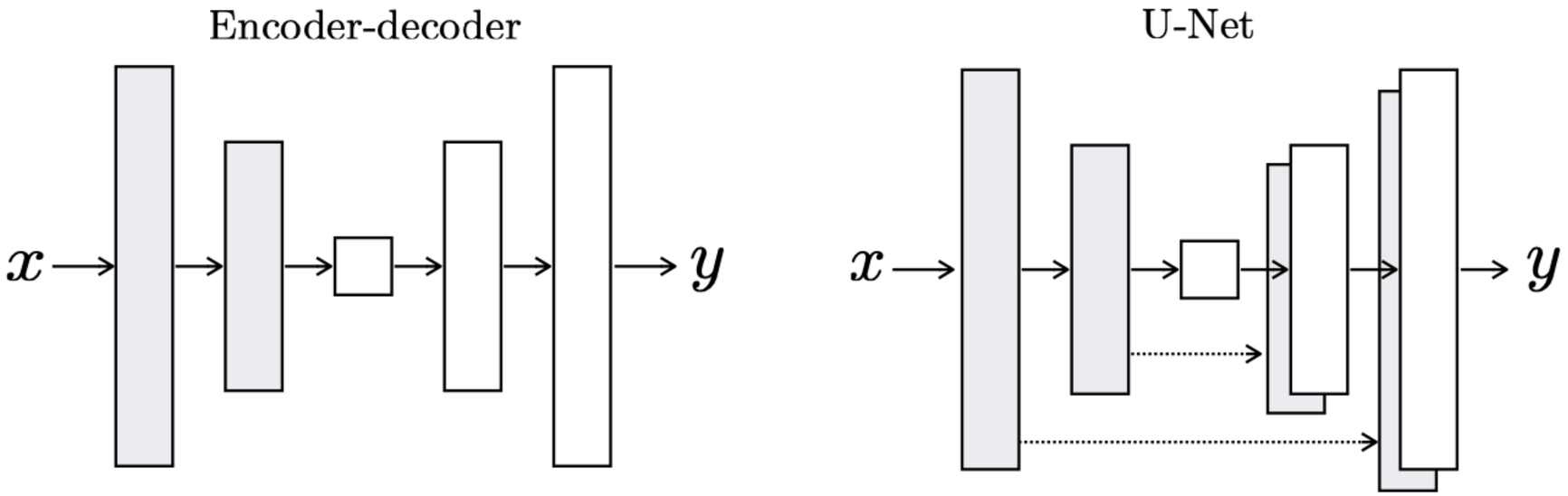


Figure credit: P. Isola, et al., Image-to-Image Translation with Conditional Adversarial Networks

Discriminator Architecture (PatchGAN)

- L2 and L1 loss: blurry images
  - Capture only the low frequencies
- GAN: good for model the high-frequency structures
- PatchGAN
  - D: discriminate only $L \times L$ patches
  - Discriminate based on average all responses

Experiments

- Semantic labels <—> photo (Cityscapes dataset)
- Architectural labels —> photo (CMP Facades dataset)
- Map <—> aerial photo (Google Maps)
- BW —> color photo
- Edges —> photo
- Sketch —> photo
- Day —> night

Pix2Pix Results

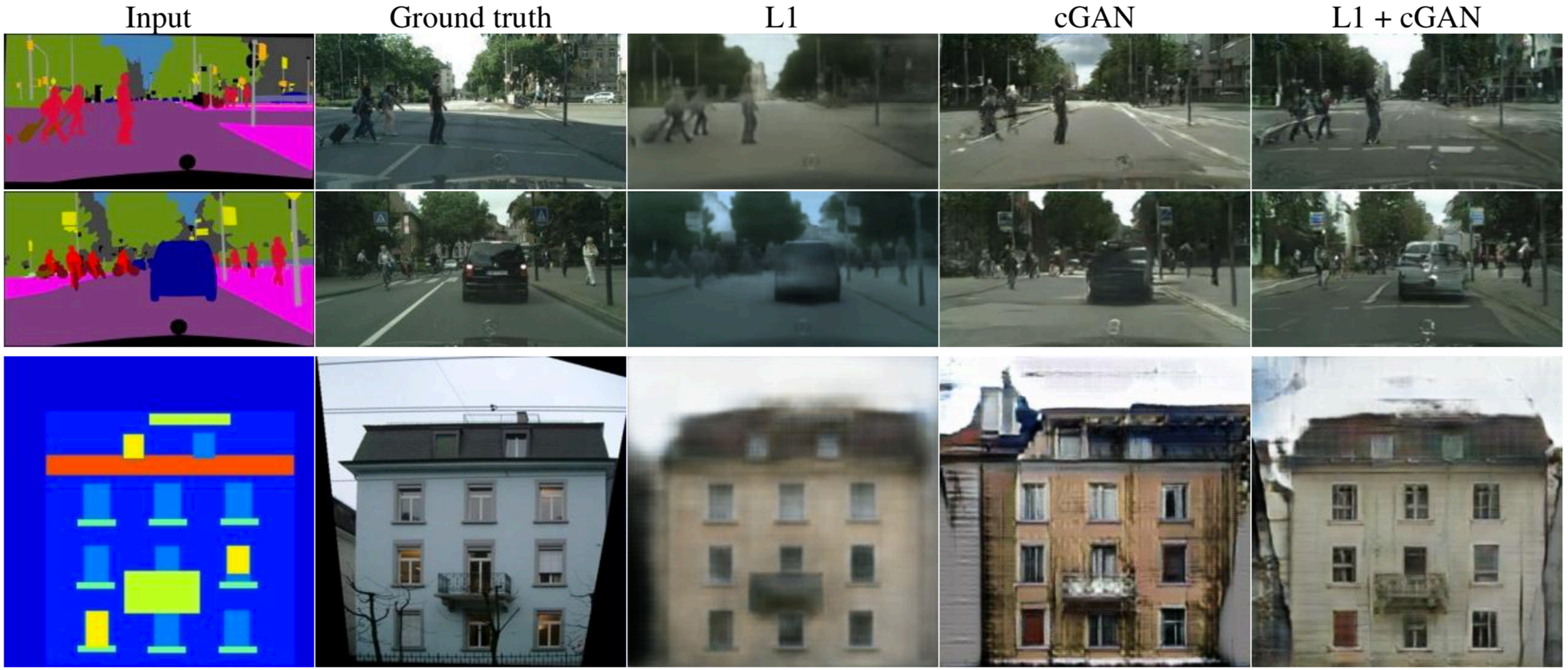


Figure credit: P. Isola, et al., Image-to-Image Translation with Conditional Adversarial Networks

Results (U-net effects)

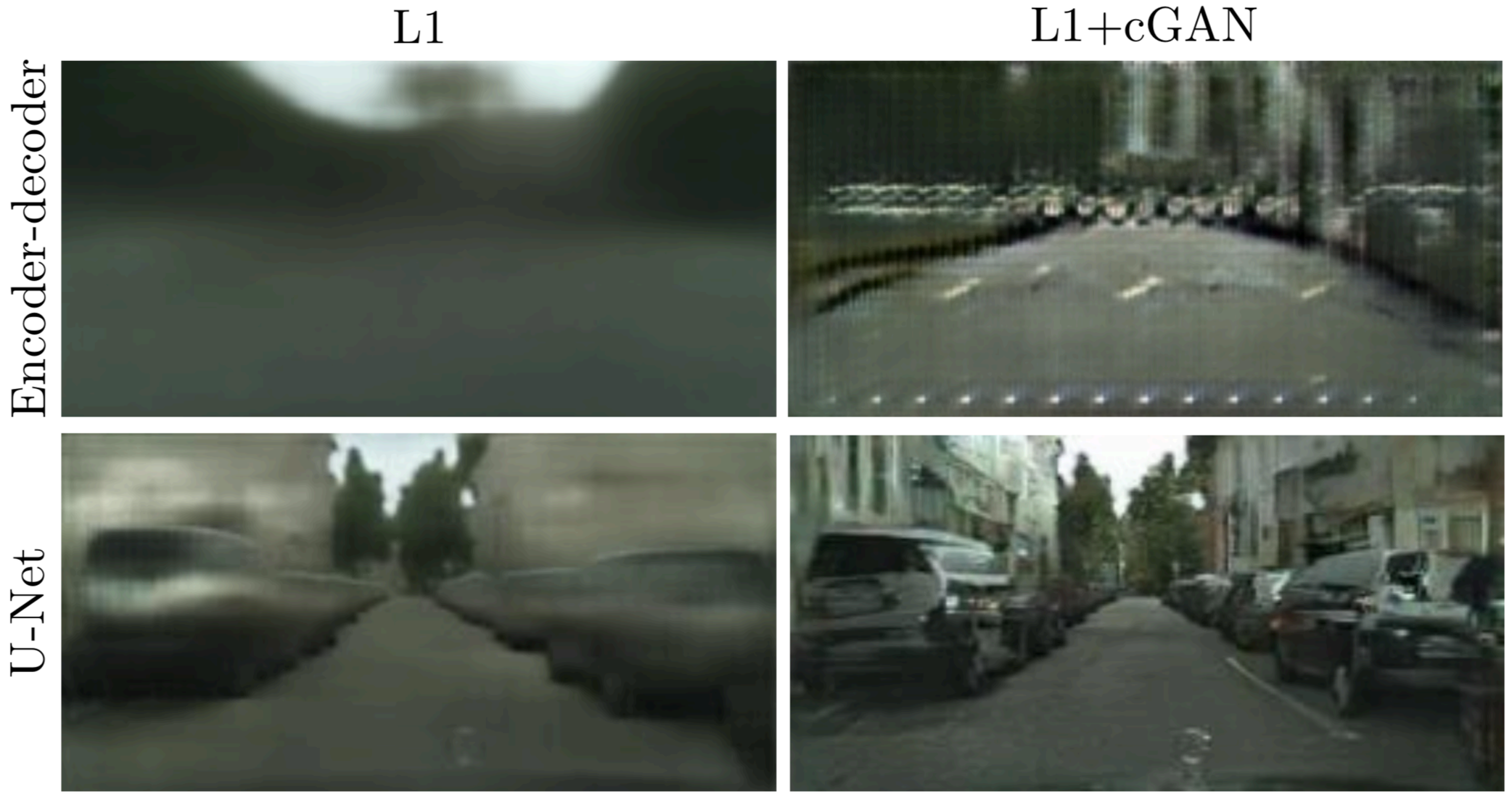


Figure credit: P. Isola, et al., Image-to-Image Translation with Conditional Adversarial Networks

Pix2Pix Results

Figure credit: P. Isola, et al., Image-to-Image Translation with Conditional Adversarial Networks

Pix2Pix Results

Figure credit: P. Isola, et al., Image-to-Image Translation with Conditional Adversarial Networks

CycleGAN

Need Paired Dataset for Pix2Pix

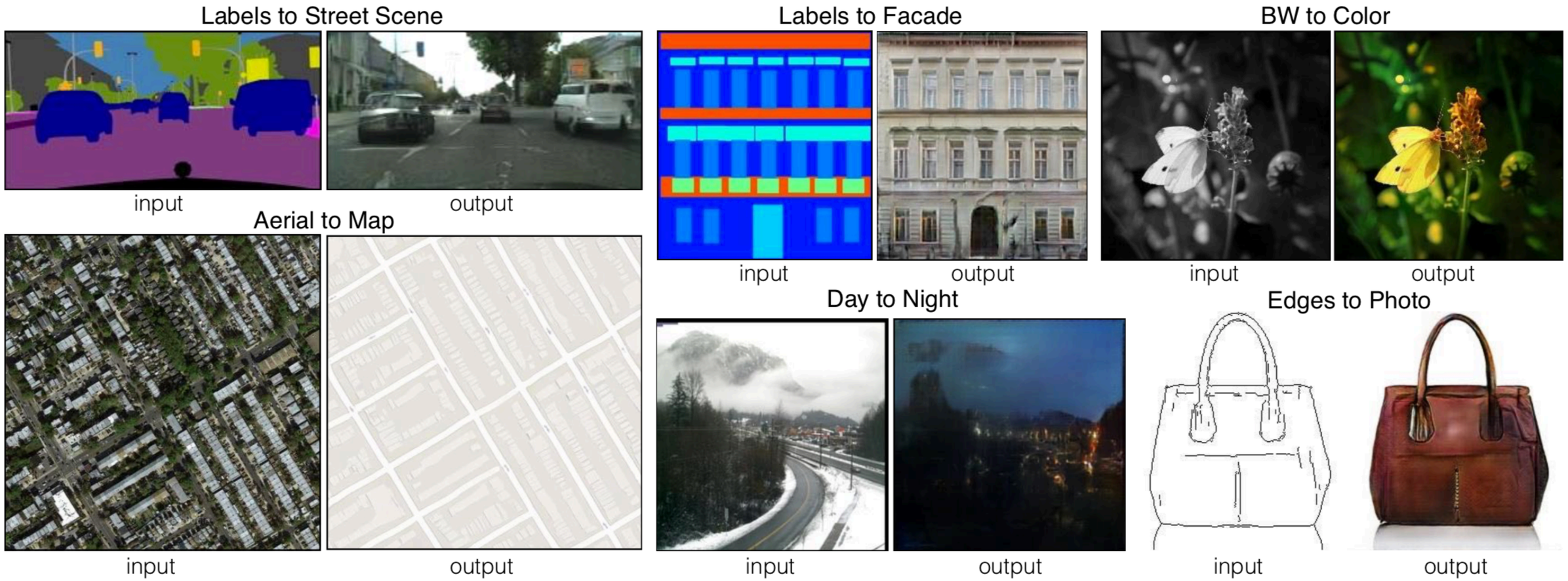


Figure credit: P. Isola, et al., Image-to-Image Translation with Conditional Adversarial Networks

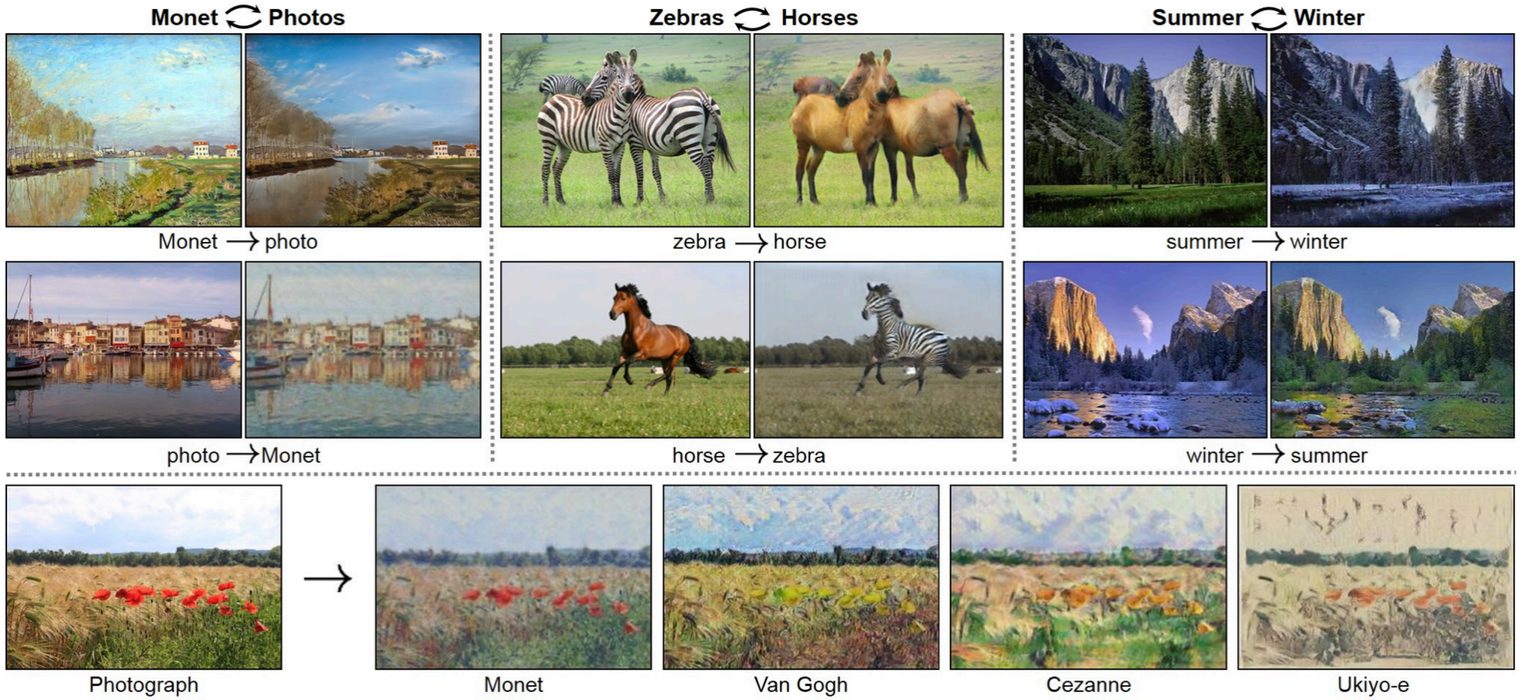
Need Paired Dataset for Pix2Pix

Is there a picture of exactly the same landscape that Monet painted in the past?
Is there a picture of the horse in exactly the same poses as the zebra picture on the left side?

Using Unpaired dataset

Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

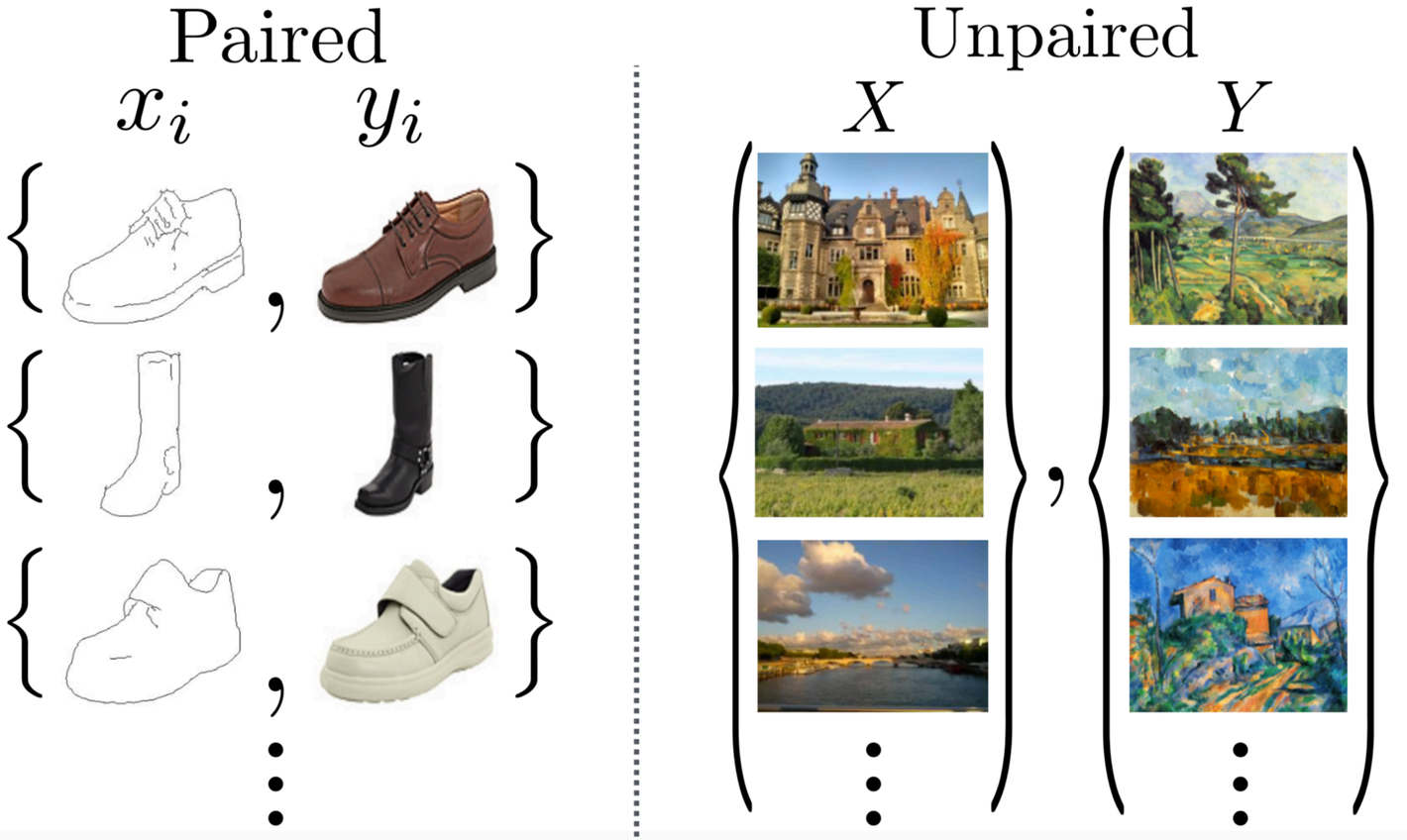
CycleGAN



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

Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

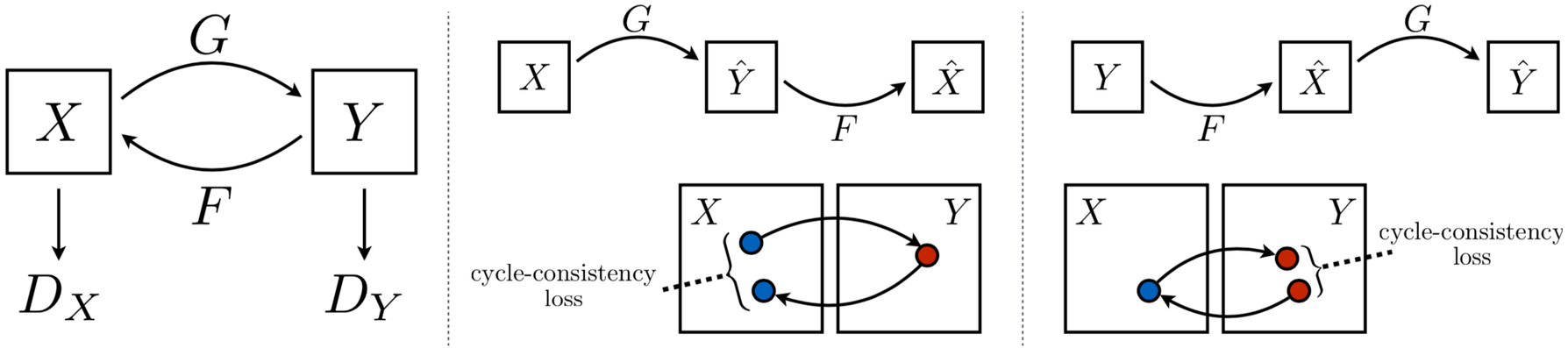
Unpaired Dataset



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

Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN Architecture



- CycleGAN은 4개의 네트워크로 구성되어 있습니다.
  Generator G: 도메인 X의 이미지를 도메인 Y의 이미지로 변환
  Generator F: 도메인 Y의 이미지를 도메인 X의 이미지로 변환
  Discriminator D_X: 도메인 X의 이미지인지 아닌지를 판별
  Discriminator D_Y: 도메인 Y의 이미지인지 아닌지를 판별
- cycle-consistency loss는 generator에 의해서 다른 도메인으로 변환된 이미지가 구조적으로 원본과 같게하기 위해 사용합니다.
- 도메인 X의 이미지 x를 generator G에 의해 도메인 Y의 이미지 $\hat{Y}$로 변환합니다. 이를 다시 generator F에 의해 도메인 X의 이미지 $\hat{x}$로 변환하고 원본 이미지 x와 L1 loss를 취해 같아지도록 만듭니다.
- 도메인 Y로부터 변환되는 경우도 마찬가지의 작업을 수행합니다.

Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Adversarial Loss

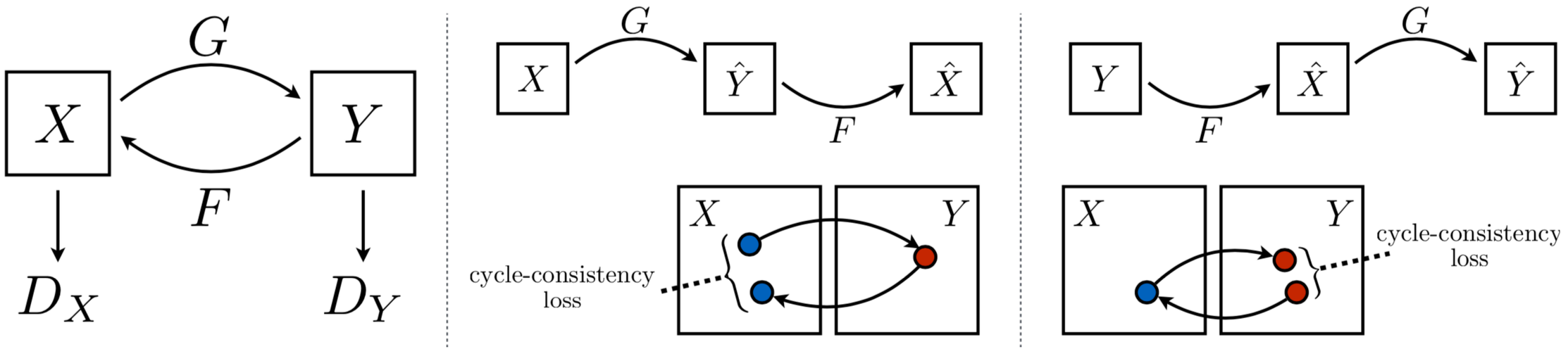
X domain to Y domain

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

Y domain to X domain

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

Cycle Consistency Loss



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

Total Objective Function

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

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

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

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

CycleGAN Results

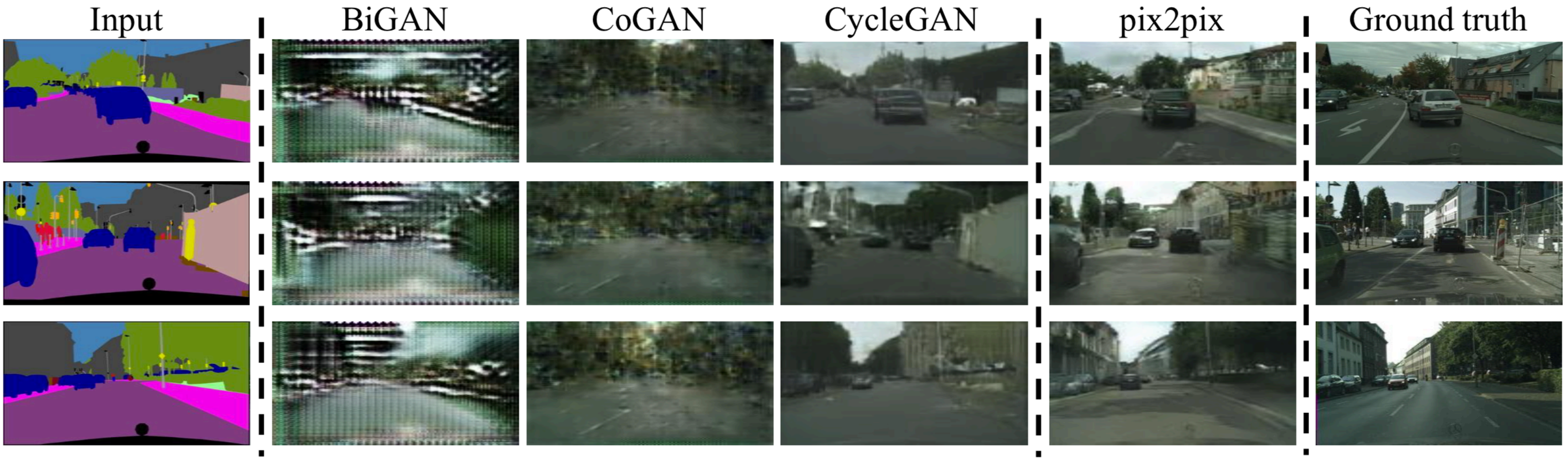


Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN Results

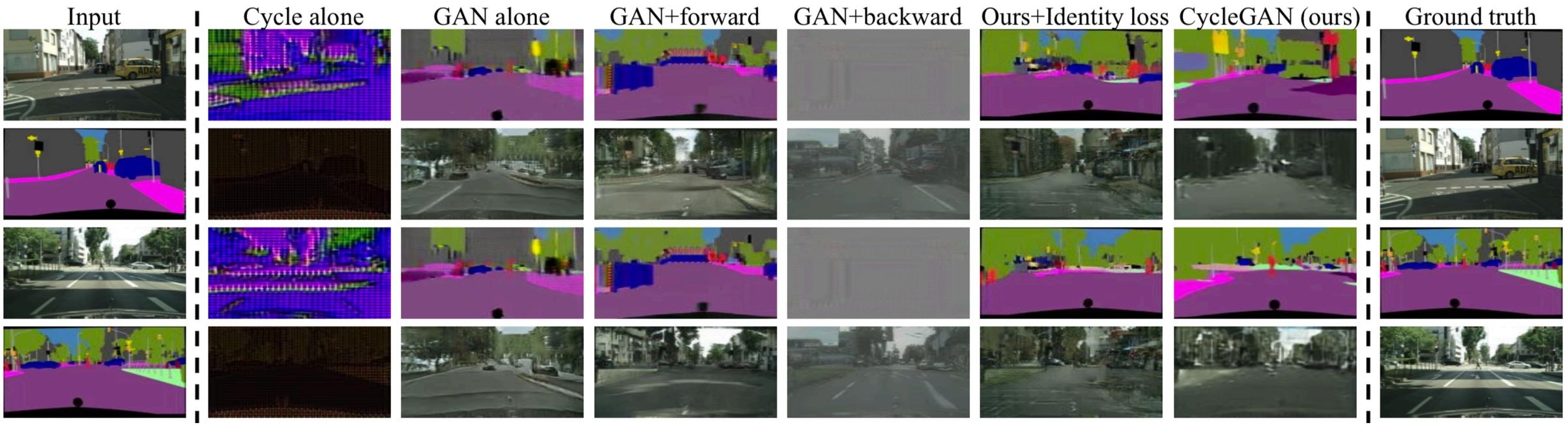


Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN Results

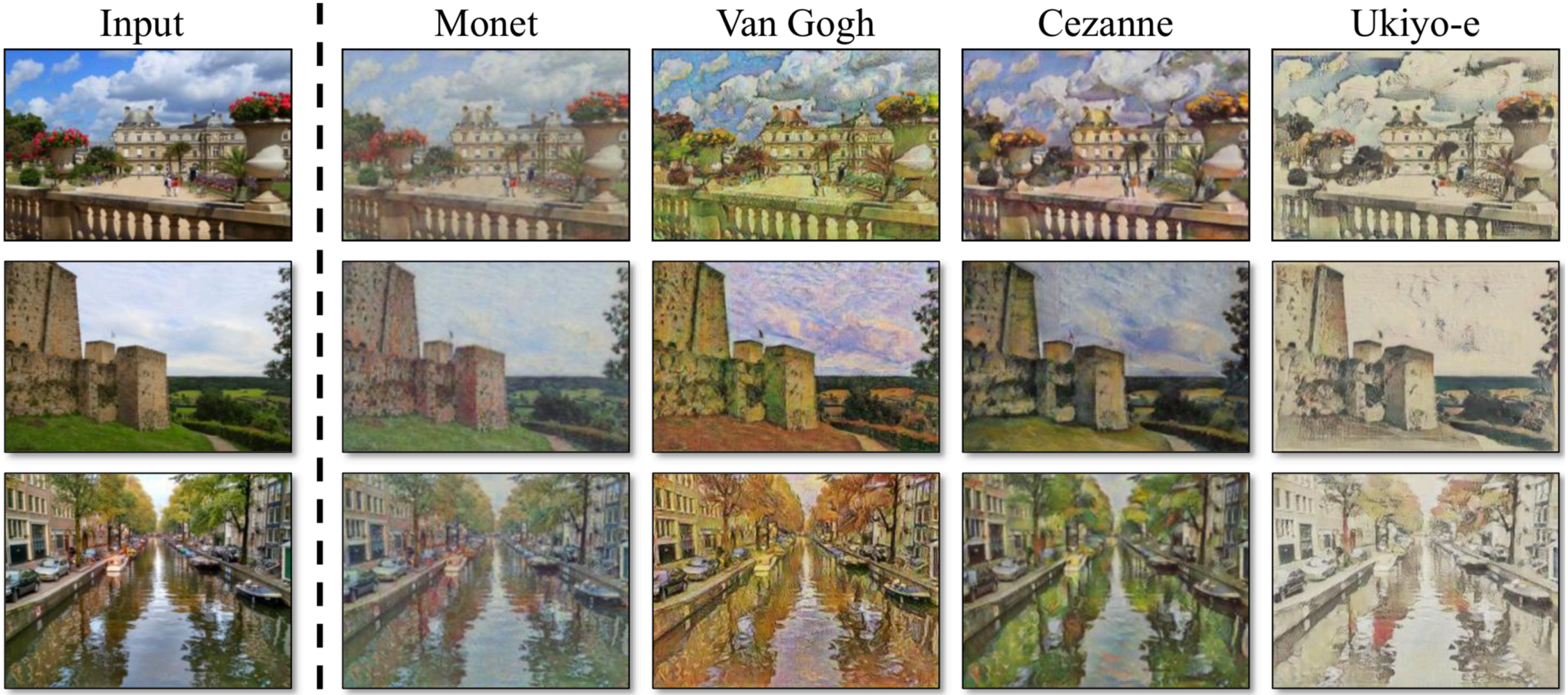


Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN Results

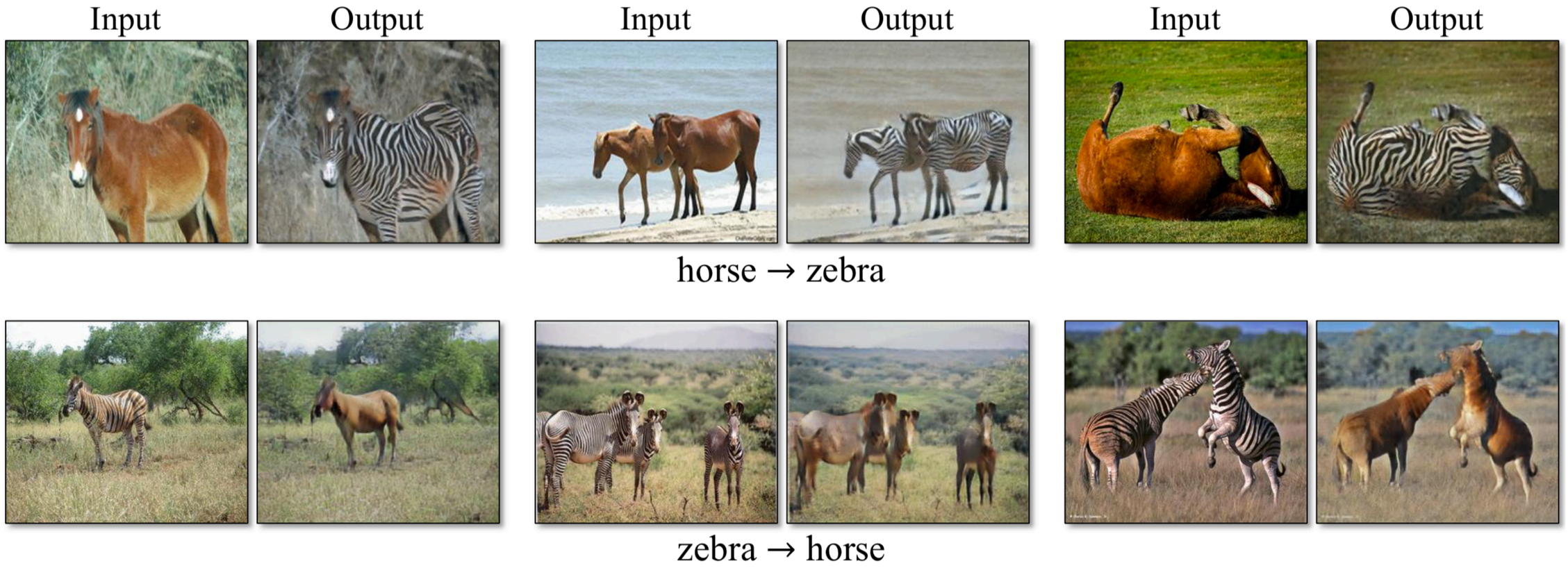


Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

CycleGAN Results

winter Yosemite → summer Yosemite

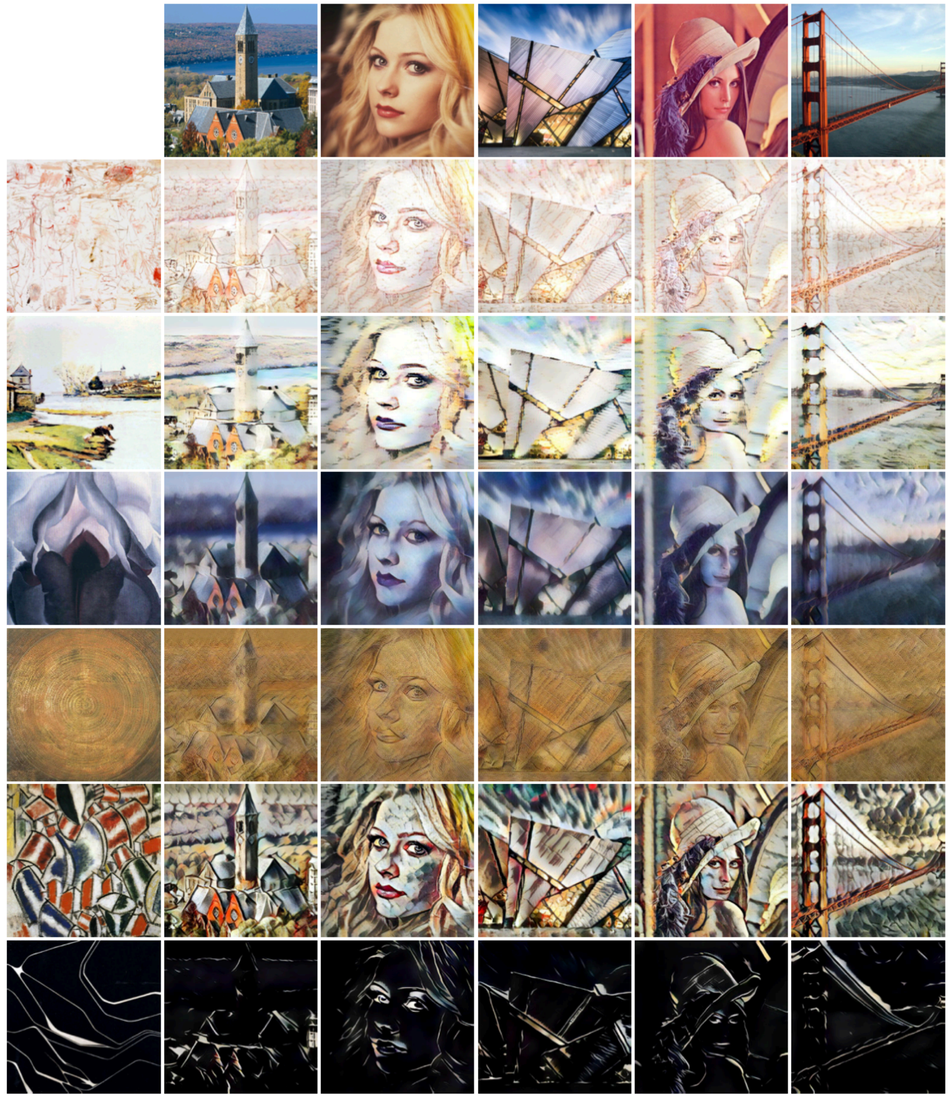
summer Yosemite → winter Yosemite

Figure credit: J-Y. Zhu, et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Style Transfer

Style Transfer

- Style transfer는 GAN 모델은 아니지만 GAN과 함께 2015-2019년 사이에 매우 활발하게 연구되었던 이미지 프로세싱 분야 입니다.
- 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 (Leon A. 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.



Huang, Xun, and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization." Proceedings of the IEEE International Conference on Computer Vision. 2017.

Style Transfer

- Style transfer는 A neural algorithm of artistic style 논문에서 최초로 시도 되었습니다
- 이 논문에서는 별도의 모델을 트레이닝하지 않고, pretrained된 VGG모델만을 이용하여 style transfer를 수행합니다.
- 먼저 Content 이미지와 style 이미지를 각각 pertained VGG 모델에 넣고 content representation들과 style representation들을 뽑습니다.
- 변환시킬 이미지를 새로 준비하고 (대개 content 이미지로 initialize) 앞에서 처럼 VGG 모델에 넣어 representation들을 뽑습니다.
- Content loss : 원본 content representation과 변환시킬 이미지의 content representation끼리 MSE loss를 취해 원본 content 이미지의 모양을 따라가도록 만듭니다.
- Style loss : 원본 style representation과 변환시킬 이미지의 style representation끼리 style loss를 취해 원본 style 이미지의 텍스쳐를 따라가도록 만듭니다. style los는 representation으로 Gram matrix를 만들고 MSE loss를 취해서 구합니다.
- 두 loss가 작아지도록 backpropagation하되, VGG의 파라메터는 업데이트되지 않도록 내버려두고 변환시킬 이미지가 업데이트 되도록하면 점차 content 이미지의 형태를 따라가고, style 이미지의 텍스쳐를 따라가게 됩니다.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." ***arXiv preprint arXiv:1508.06576*** **(2015).**

Style Transfer - Loss Function

$$\mathscr{L}_{\textbf{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathscr{L}_{\textbf{content}}(\vec{p}, \vec{x}) + \beta \mathscr{L}_{\textbf{style}}(\vec{a}, \vec{x})$$

$$\mathscr{L}_{\textbf{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l} \right)^2$$

$$\mathscr{L}_{\textbf{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

$$E_l = \frac{1}{4 N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l} \right)^2$$

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." ***arXiv preprint arXiv:1508.06576*** **(2015).**

Style Transfer

- VGG는 본래 image recognition을 위한 모델로 일반적으로 알려져있는 Convolution + Fully Connected Layer 형태를 가지고 있습니다.
- Style transfer에서는 이중 convolution layer의 중간 출력 결과를 이용합니다.
- VGG가 image recognition을 위해 트레이닝 되었기 때문에 convolution layer는 이미지의 형태를 분석하는 역할을 하고 있다고 볼 수 있습니다.

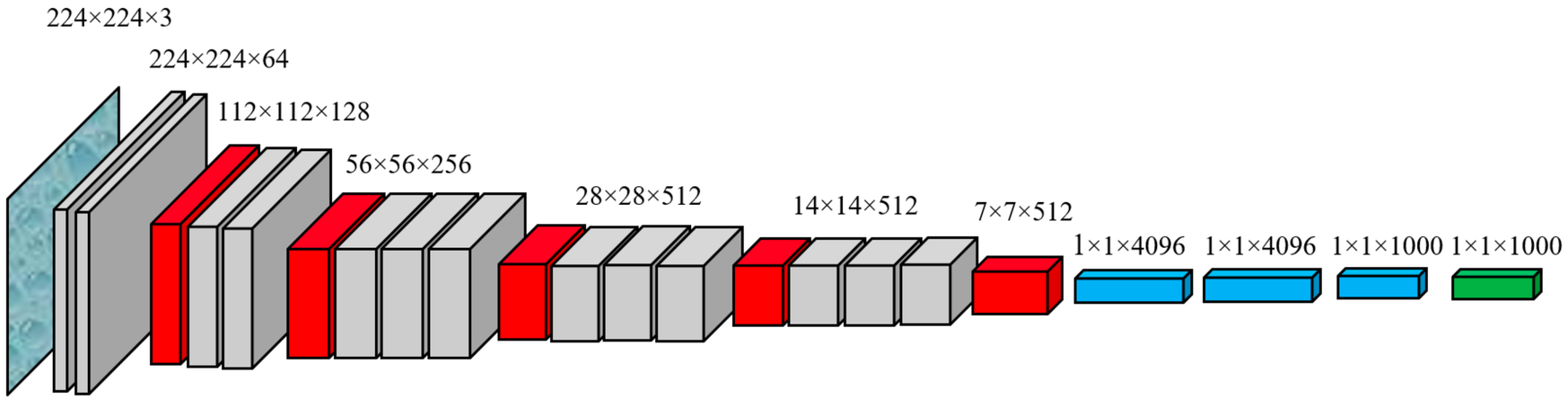
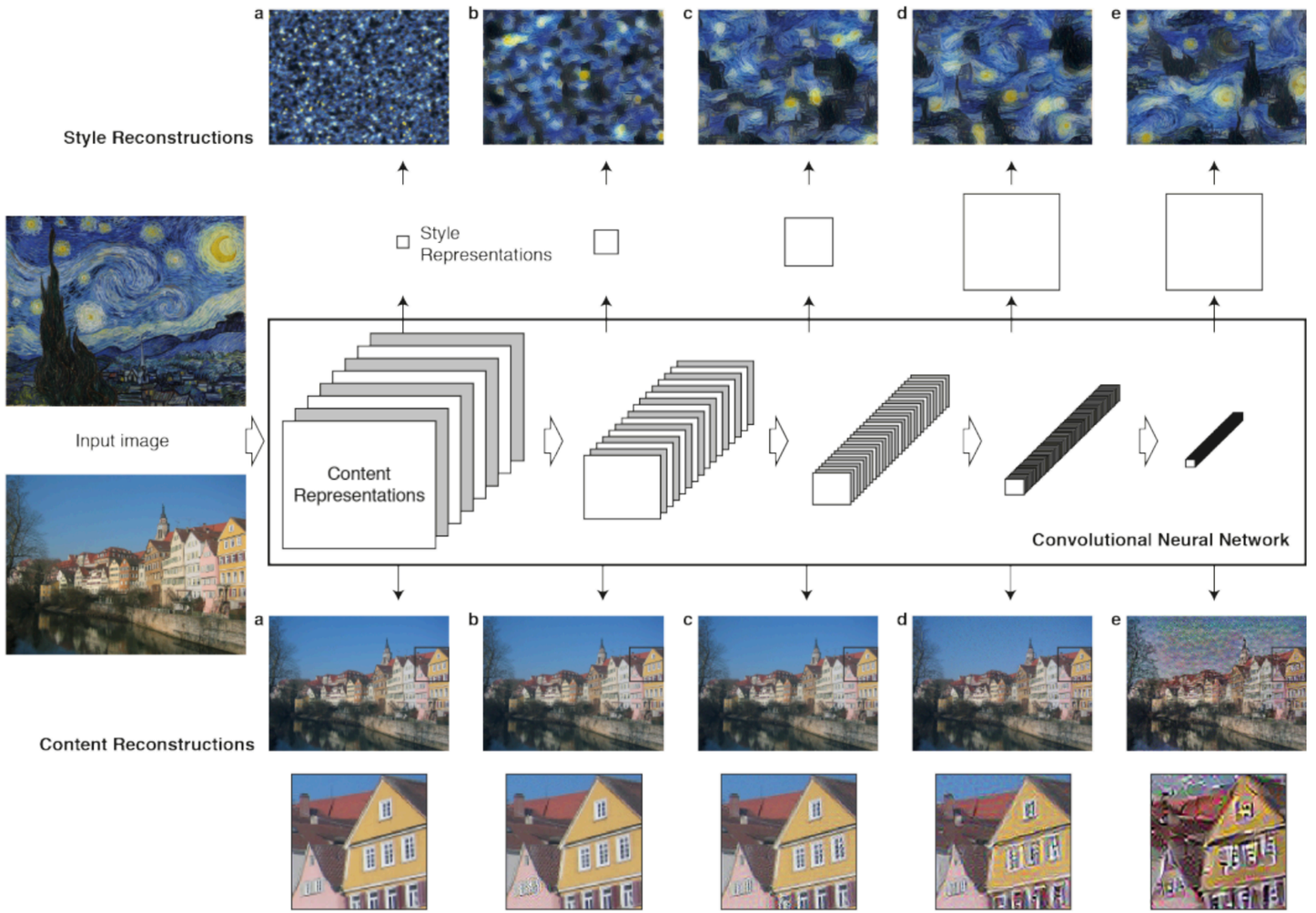


Figure credit : https://en.wikipedia.org/wiki/File:VGG_neural_network.png

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

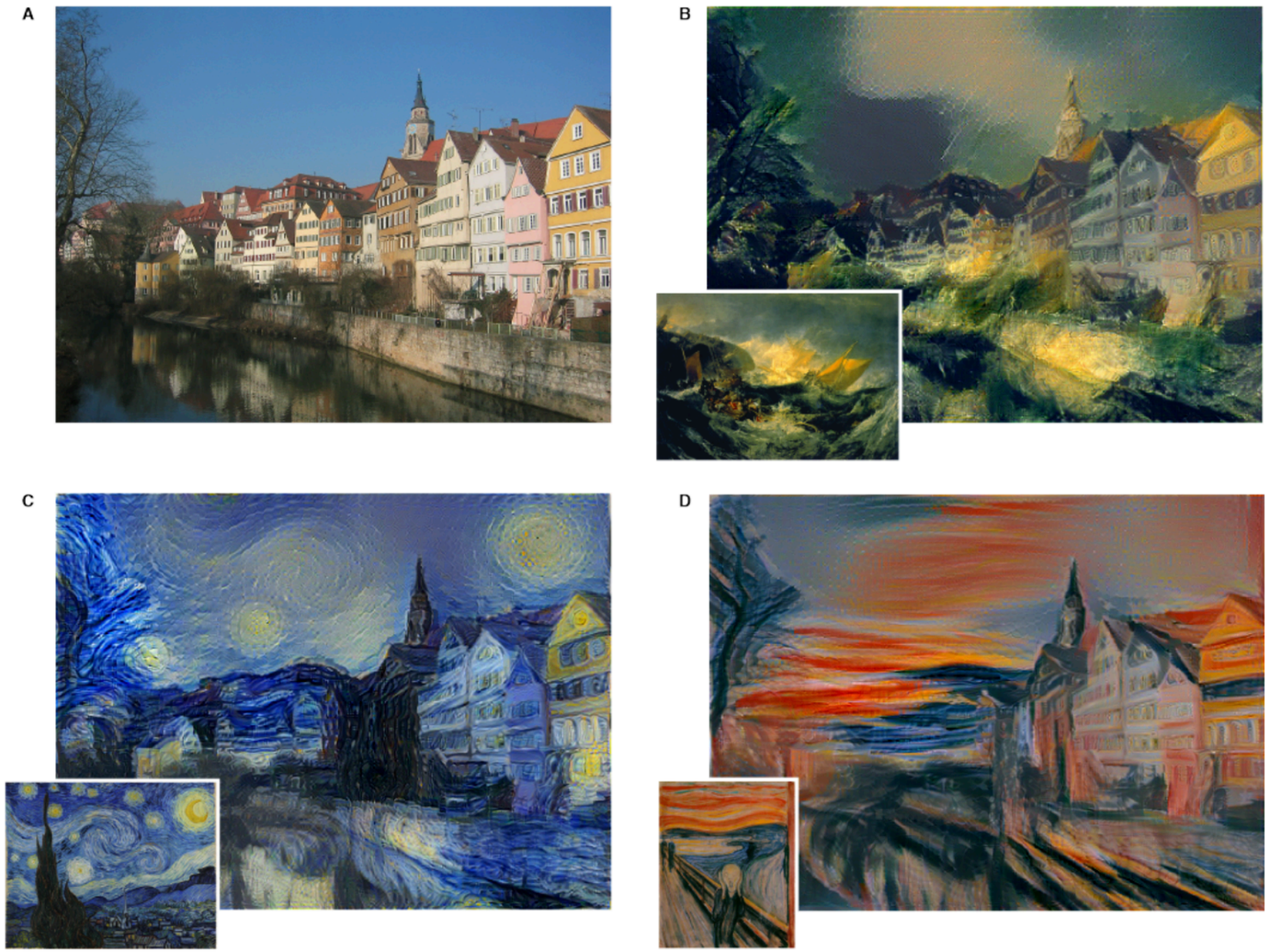
Style Transfer



- VGG의 convolution layer들중 어떤 layer의 출력 값으로 style loss를 잡으냐에 따라 texture가 자잘하거나 굵어질 수 있습니다.
- 입력층에 가까울 수록 local한 정보를 분석하므로 자잘한 texture모양이 나오고, convolution이 많이 진행된 출력으로 style loss를 잡으면 global한 정보를 분석하므로 굵은 texture가 만들어 집니다.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576* (2015).

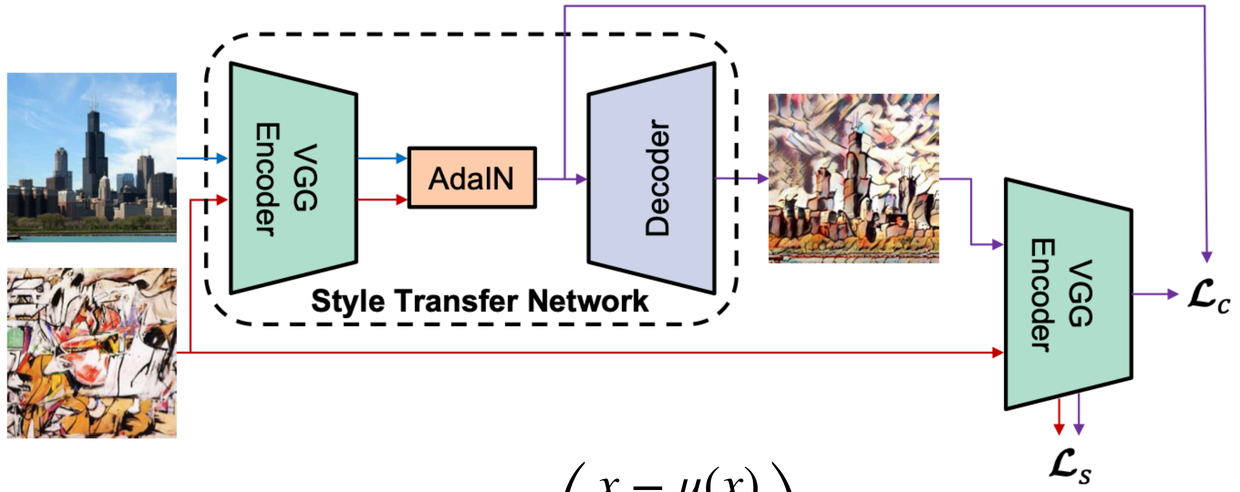
Style Transfer



Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576* (2015).

Style Transfer

- Gatys의 방법은 loss를 잡고 여러번 backpropagation 시키느라 시간이 오래 걸립니다.
- 이런 단점을 극복하기 위해 adaptive instance normalization (AdaIN) 방법이 개발되 었습니다.
- 이 모델에서도 VGG를 encoder로 이용하며, 추가적인 decoder를 트레이닝 합니다.
- AdaIN은 content 이미지의 representation을 standardization하고, style 이미지의 representation에 대한 statistics를 적용합니다.



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

Huang, Xun, and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization." Proceedings of the IEEE International Conference on Computer Vision. 2017.

ProgressiveGAN

ProgressiveGAN

- ProgressiveGAN은 Progressive growing of gans for improved quality, stability, and variation 논문에서 제안되었습니다.
- 1024x1024 고해상도의 얼굴 이미지를 얻기 위해 저해상도의 이미지부터 고해상도의 이미지까지 점차 해상도를 높히면서 만들어 나갑니다.
- 각 해상도의 이미지를 만들기 위한 generator와 discriminator가 각각 존재합니다.
- 저해상도의 이미지는 다음 고해상도의 이미지를 만드는 generator의 입력으로 사용됩니다.
- 고해상도 이미지를 만들 때 트레이닝 초기에는 generator가 의미없는 노이즈 이미지를 내보내게 됩니다. 이를 막기 위해 저해상도 이미지를 nearest neighbor 방식을 통해 upsampling한 이미지와 generator의 출력 이미지를 linear interpolation한 중간 이미지를 구해 discriminator로 보냅니다.
- 트레이닝이 진행됨에 따라 interpolation coefficient를 조정하여 generator가 출력한 이미지만을 내보내게 만듭니다.
- 모든 generator와 discriminator의 트레이닝이 완료되면, generator를 직렬로 사용하여 저해상도부터 고해상도의 이미지까지 점차 만들어나갈 수 있습니다.

Karras, Tero, et al. "Progressive growing of gans for improved quality, stability, and variation." arXiv preprint arXiv:1710.10196 (2017).

ProgressiveGAN networks

Start with low-resolution images,
And then progressively increase the resolution by adding layers to the networks.

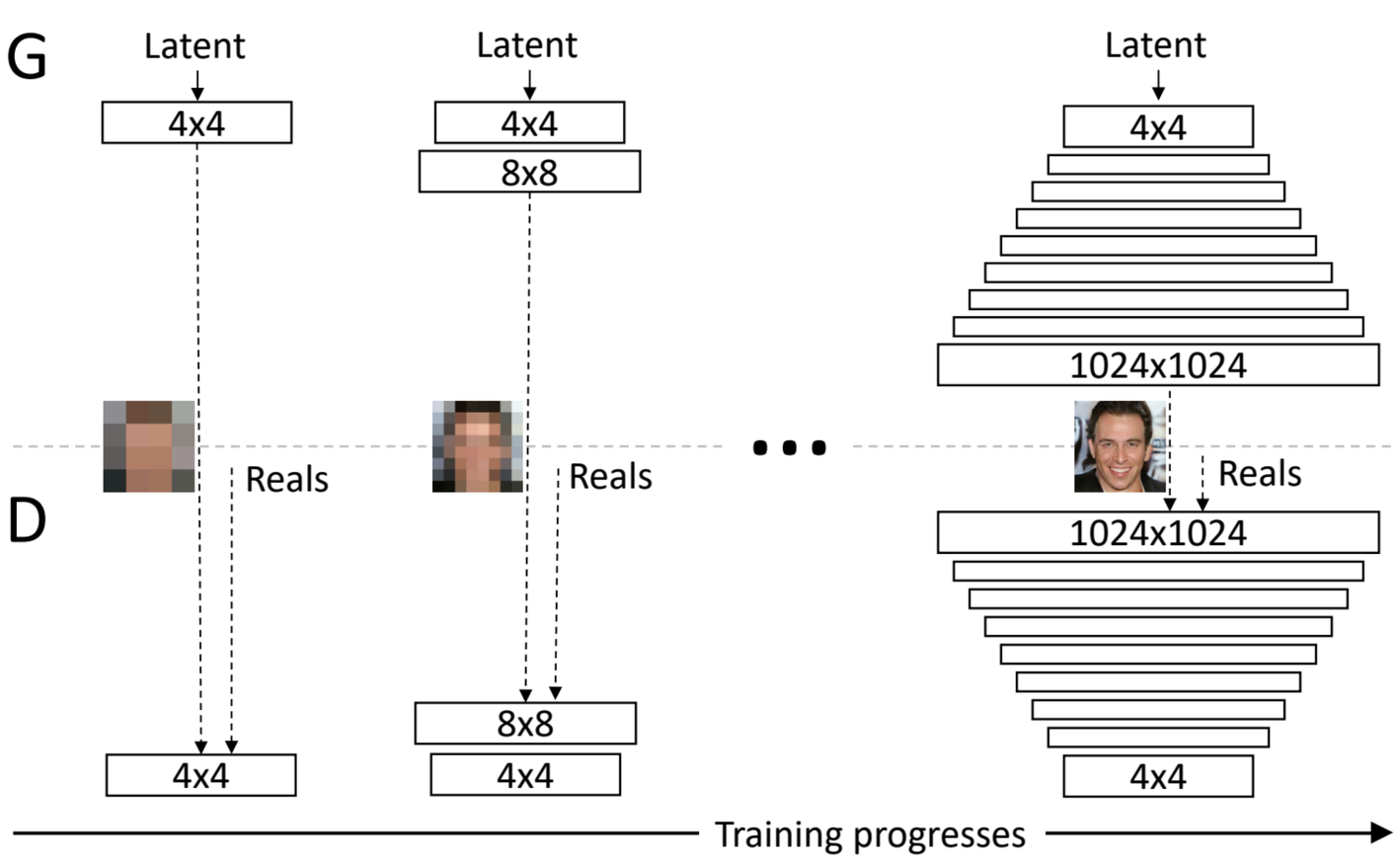
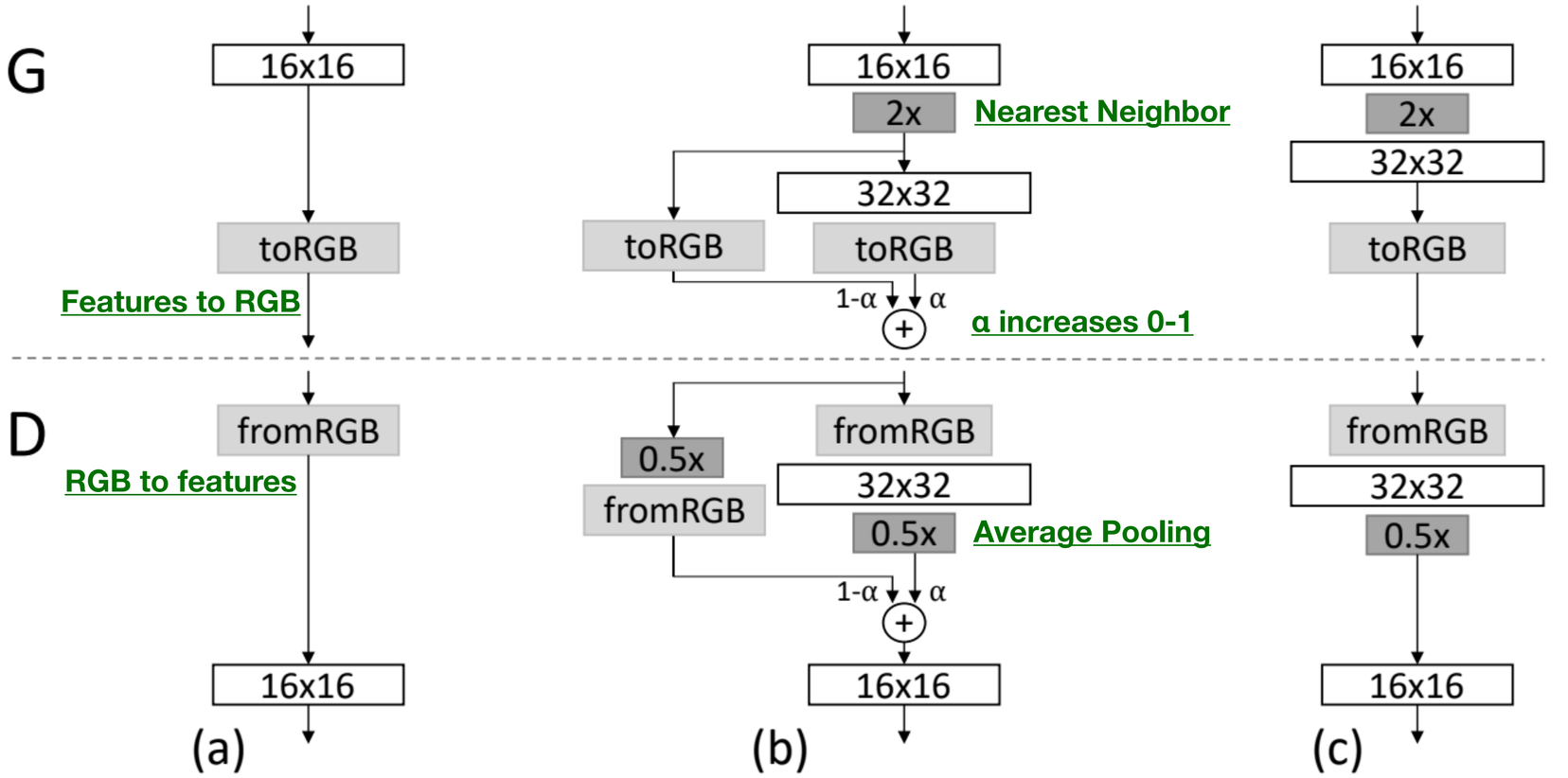


Figure credit: Tero Karras, et al., Progressive Growing of GANs for Improved Quality, Stability, and Variation

ProgressiveGAN training idea

To avoid sudden shocks to already well-trained,
Fade in the new layers smoothly.



Before adding new layers **Training new layers** **Training finished**

Figure credit: Tero Karras, et al., Progressive Growing of GANs for Improved Quality, Stability, and Variation

ProgressiveGAN results

1024x1024 images generated using the CELEBA-HQ dataset.

Figure credit: Tero Karras, et al., Progressive Growing of GANs for Improved Quality, Stability, and Variation

StyleGAN

StyleGAN

- StyleGAN은 A style-based generator architecture for generative adversarial networks 논문에서 제안되었습니다.
- ProgressiveGAN의 좋은 결과물들은 많은 결과물들 중에서 골라낸 (curated)것인데 반해 styleGAN은 항상 좋은 결과물을 내보내는 장점을 가지고 있습니다.
- StyleGAN은 coarse style부터 mid style과 fine style까지 style을 선택할 수 있습니다.
- StyleGAN은 progressiveGAN과 다르게 저해상도부터 고해상도까지 순차적으로 만들지 않고, 하나의 네트워크로 출력합니다.
- Layer마다 style을 주입시키기 위해 style transfer에서 활용되던 AdaIN이라는 기법을 사용합니다.
- Style을 결정하는 noise vector는 각 layer마다 주입되며 샘플링된 z vector를 FC layer들로 변환한 w vector를 사용합니다.

Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

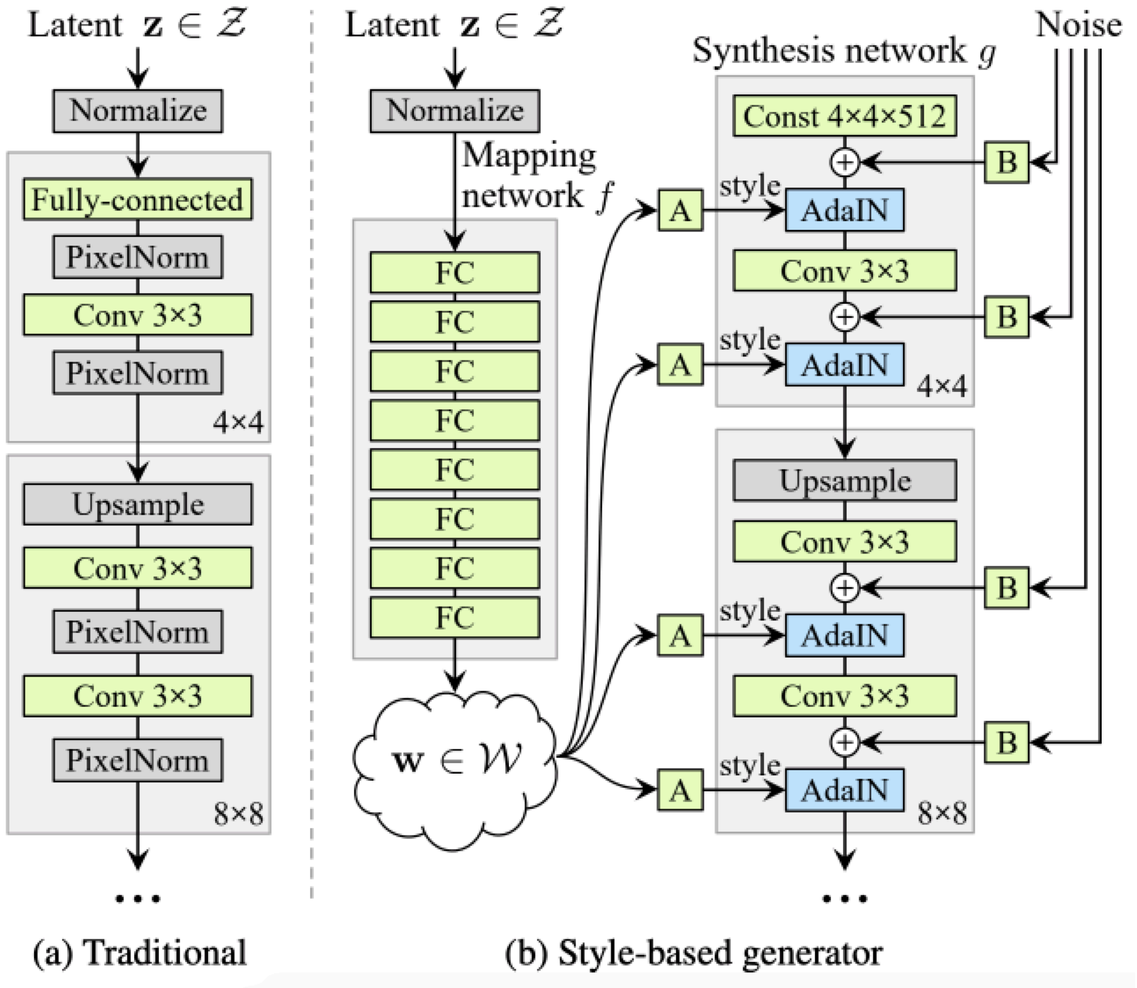
StyleGAN Generated Images

Figure 2. Uncurated set of images produced by our style-based generator (config F) with the FFHQ dataset. Here we used a variation of the truncation trick [42, 5, 34] with $\psi = 0.7$ for resolutions $4\ 2 - 322$. Please see the accompanying video for more results.

StyleGAN Coarse Styles

StyleGAN Middle, Fine Styles

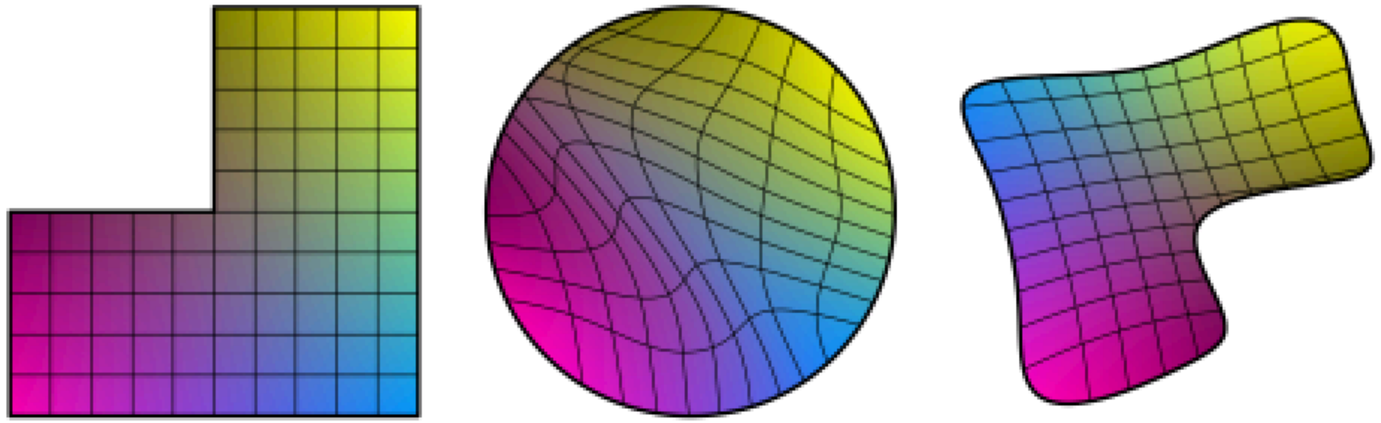
StyleGAN networks



1. **Map non-linearly the latent input z into intermediate w.**
2. **W controls the generator through adaptive instance normalization(AdaIN) at each convolutional layer.**
3. **"A" stands for a learned affine transform.**
4. **"B" applies learned per-channel scaling factors to the noise input.**

Figure credit: Tero Karras, et al., A Style-Based Generator Architecture for Generative Adversarial Networks

StyleGAN Disentanglement



(a) Distribution of features in training set

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

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

(a) An example training set where some combination (e.g., long haired males) is missing.

(b) This forces the mapping from Z to image features to become curved so that the forbidden combination disappears in Z to prevent the sampling of invalid combinations.

(c) The learned mapping from Z to W is able to “undo” much of the warping.

Figure credit: Tero Karras, et al., A Style-Based Generator Architecture for Generative Adversarial Networks

StyleGAN Adaptive Instance Norm.

Batch Normalization (Ioffe and Szegedy, 2015.)

$$\mathrm{BN}(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta \quad \mu_c(x) = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \quad \sigma_c(x) = \sqrt{\frac{1}{NHW}\sum_{n=1}^{N}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw}-\mu_c(x))^2+\epsilon}$$

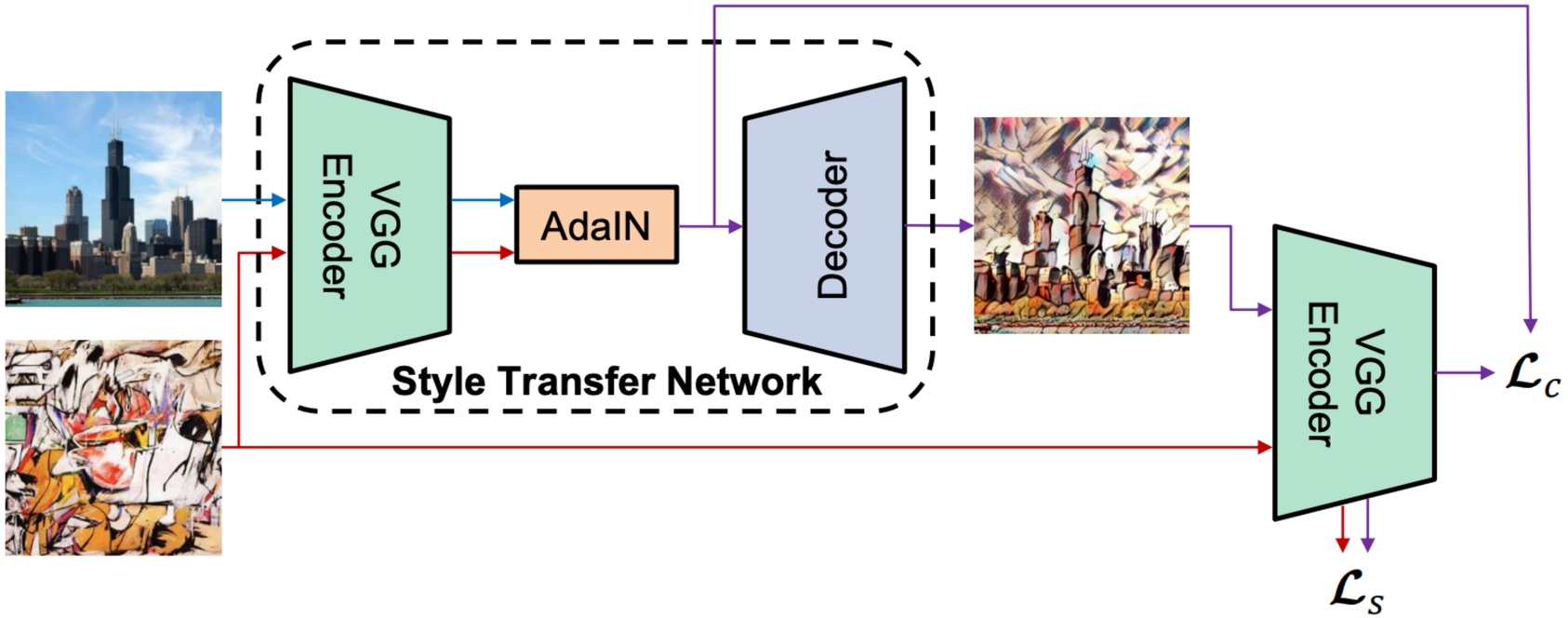
Instance Normalization (Ulyanov et al., 2017.)

$$\mathrm{IN}(x) = \gamma\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta \quad \mu_{nc}(x) = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw} \quad \sigma_{nc}(x) = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw}-\mu_{nc}(x))^2+\epsilon}$$

Conditional Instance Normalization (Dumoulin et al., 2017.)

$$\mathrm{CIN}(x;s) = \gamma^s\left(\frac{x-\mu(x)}{\sigma(x)}\right)+\beta^s, \text{ where s is a style index}, \gamma \; and \; \beta \text{ are trainable parameters}$$

StyleGAN Adaptive Instance Norm.

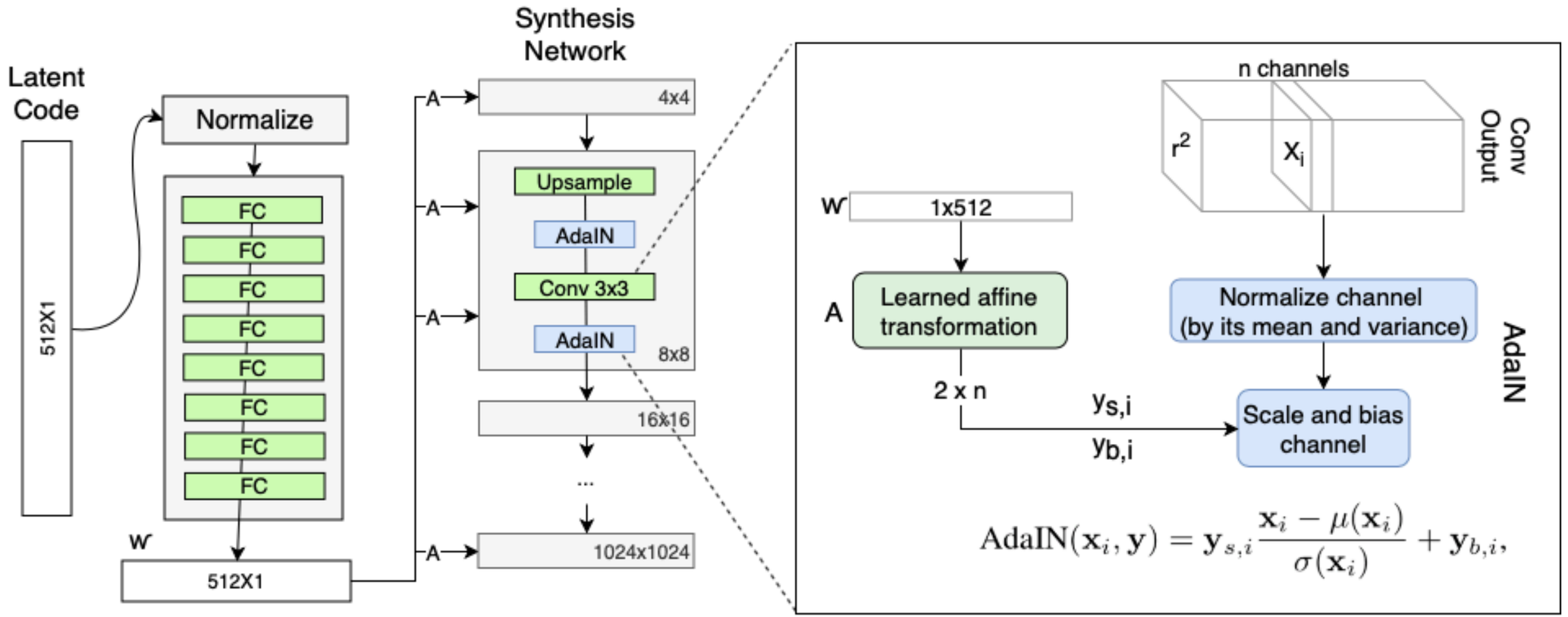


AdaIN receives a content input x and a style input y,
and simply aligns the channelize mean and variance of x to match those of y

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

Figure credit: Huang et al., Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization

StyleGAN Adaptive Instance Norm.



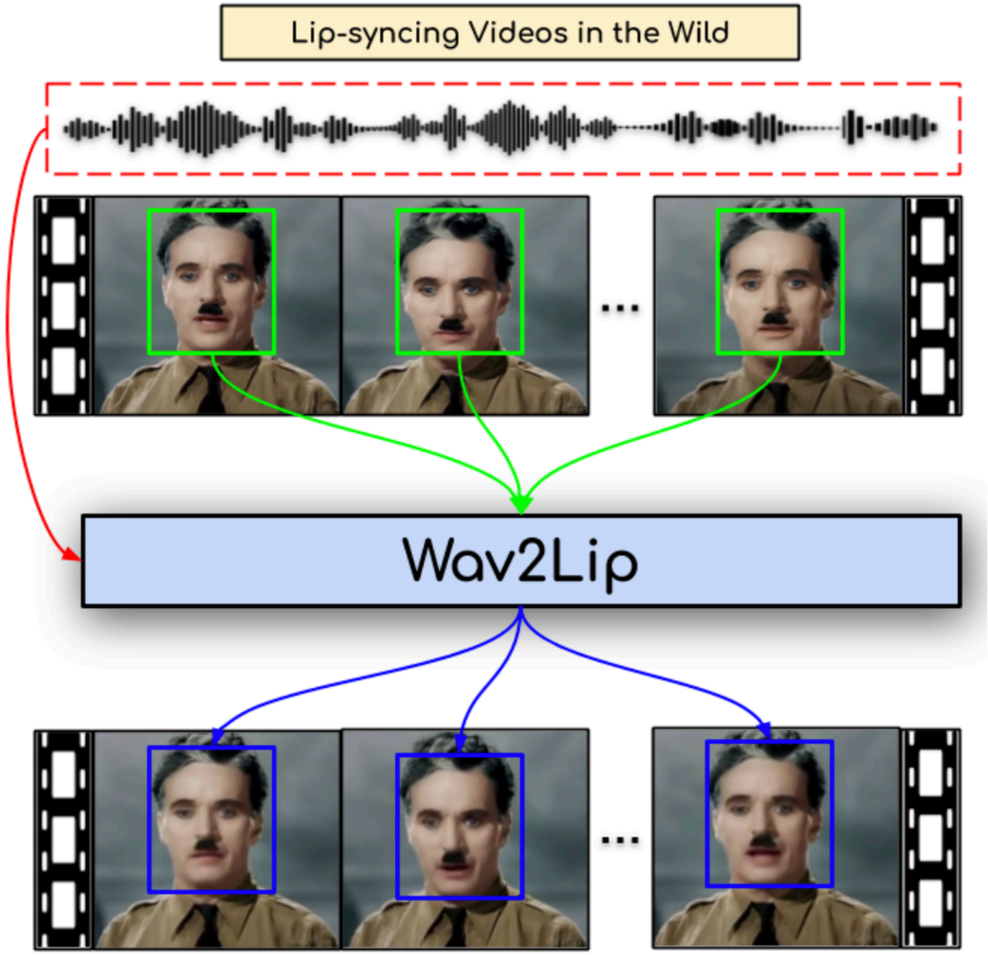
Each feature map x_i is normalized separately, and then scaled and biased using the corresponding scalar components from style y.

Figure credit: Rani Horev, Explained: A Style-Based Generator Architecture for GANs - Generating and Tuning Realistic Artificial Faces
https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431

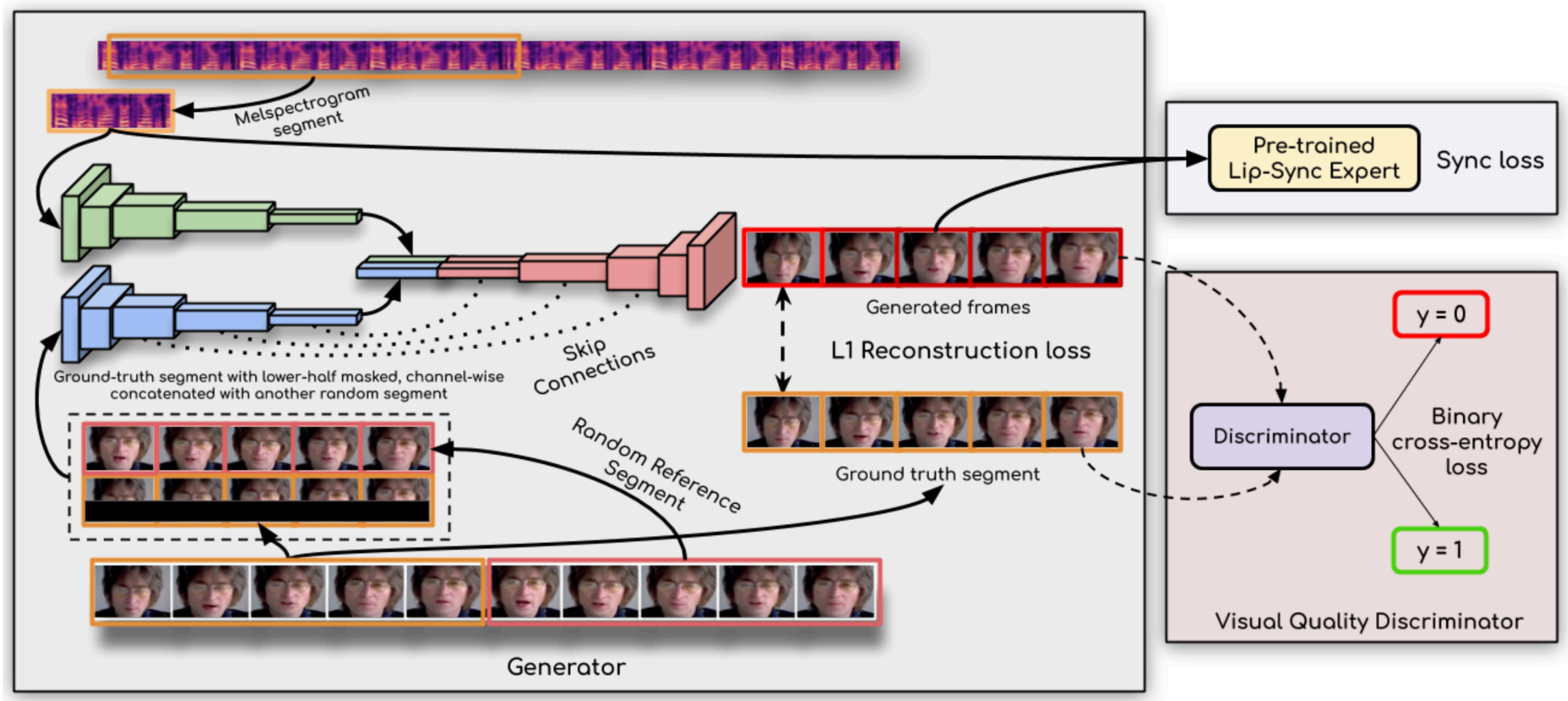
Wav2Lip

Wav2Lip

- Wav2Lip은 A Lip Sync Expert Is All You Need for Speech to Lip GenerationIn The Wild 논문에서 제안된 모델입니다.
- Wave는 음성 신호를 기록한 데이터이고, lip은 입술을 뜻합니다. 즉, wav2lip은 음성 신호에 의해서 주어진 인물 동영상의 입술 부분을 변환하여, 해당 음성 신호를 말하는 것처럼 바꾸어 주는 모델입니다.
- 기본적으로 pix2pix와 같이 source 이미지(원래 입술)를, dest image(wav에 해당하는 입술)로 변환하는 방법을 사용하고 있습니다.
- 추가적으로 음성 신호인 wave를 변환한 mel-spectrogram을 condition으로 사용해 음성과 sync가 맞도록 합니다.
- Demo : https://youtu.be/0fXaDCZNOJc



Wav2Lip



Wav2Lip

- https://github.com/Rudrabha/Wav2Lip/blob/master/models/wav2lip.py

Wav2Lip module
메인이 되는 Wav2Lip 모듈은 face encoder와 audio encoder, 그리고 face decoder로 구성되어 있습니다.
- Face encoder에서는 입력으로 ground truth 프레임들과 out-of-sync 프레임들을 받습니다. 이 때 ground truth 프레임들에서 입술 부분은 mask하여 전달됩니다. 여러 convolution layer들을 거쳐서 인코딩된 벡터 하나를 출력합니다.
- Audio encoder에서는 입력으로 conditional으로 사용되는 mel-spectrogram을 받아, face encoder와 마찬가지로 여러 convolution layer를 거쳐 인코딩된 벡터 하나를 출력합니다.
- face encoder와 audio encoder를 통해 인코딩된 두 벡터를 concat하여 face decoder를 거쳐 새로운 얼굴 프레임들을 출력합니다. Ground truth 프레임과 L2 loss를 사용하여 sync가 맞는 입술 모양이 복원되도록 합니다.

Wav2Lip_disc_qual module
- Pix2pix에서 blurry한 결과를 완화시키기 위해 L1, L2 loss에 더해 GAN loss를 사용하였듯이 wav2lip에서도 GAN loss를 사용하여 화질을 개선합니다.

Wav2Lip

- https://github.com/Rudrabha/Wav2Lip/blob/master/models/syncnet.py

SyncNet_color module

- Wav2Lip과 다른 모델들과의 가장 큰 차이는 입술과 음성간에 싱크를 맞추기 위해 별도의 네트워크를 사용한다는 점입니다. syncnet은 convolution layer로 이루어진 face encoder와 audio encoder로 구성되어 있고, 각 encoder들은 face와 audio를 인코딩하여 벡터로 각각 변환합니다.
- 싱크가 맞지 않는 (입술, 음성) 데이터 쌍은 인코딩된 벡터들의 cosine similarity가 작아지도록 만들고, 싱크가 맞는 데이터 쌍은 커지도록 만듭니다.

$$P_{\text{sync}} = \frac{v \cdot s}{max(\|v\|_2 \cdot \|s\|_2, \epsilon)}$$