SSY098 - Image Analysis

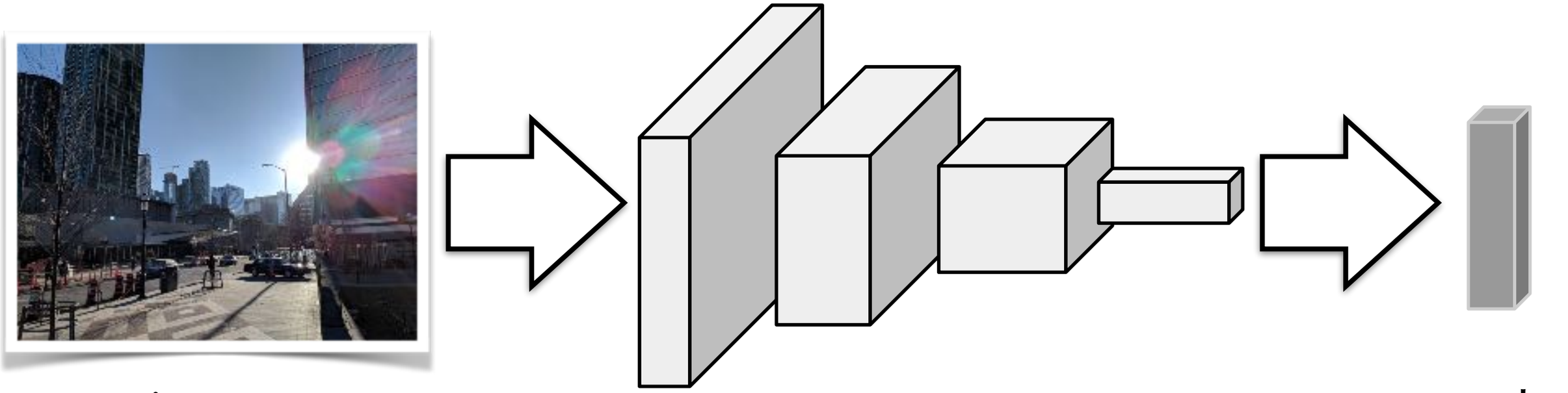
Lecture 12 - Generative Neural Networks

Torsten Sattler

Last Lecture

Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	Filtering, gradients, scale	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	Camera Geometry	Lab 4
Feb. 20	More camera geometry	
Feb. 24	**Generative neural networks**	
Feb. 27	Generative neural networks	
Mar. 2	Visual Localization & Feature Learning	
Mar. 9	No lecture	

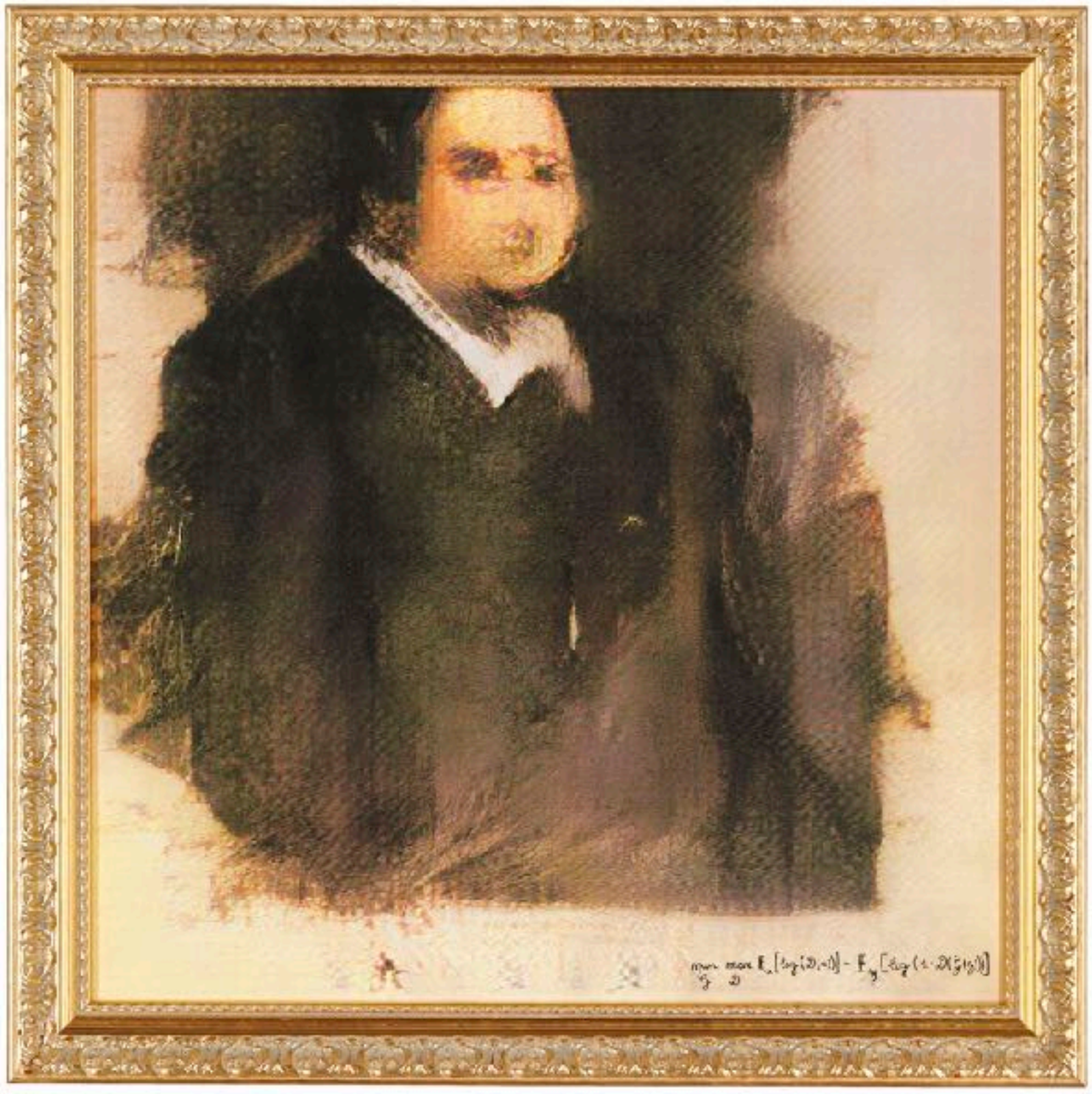
Generative Neural Networks



image

generator / decoder

output

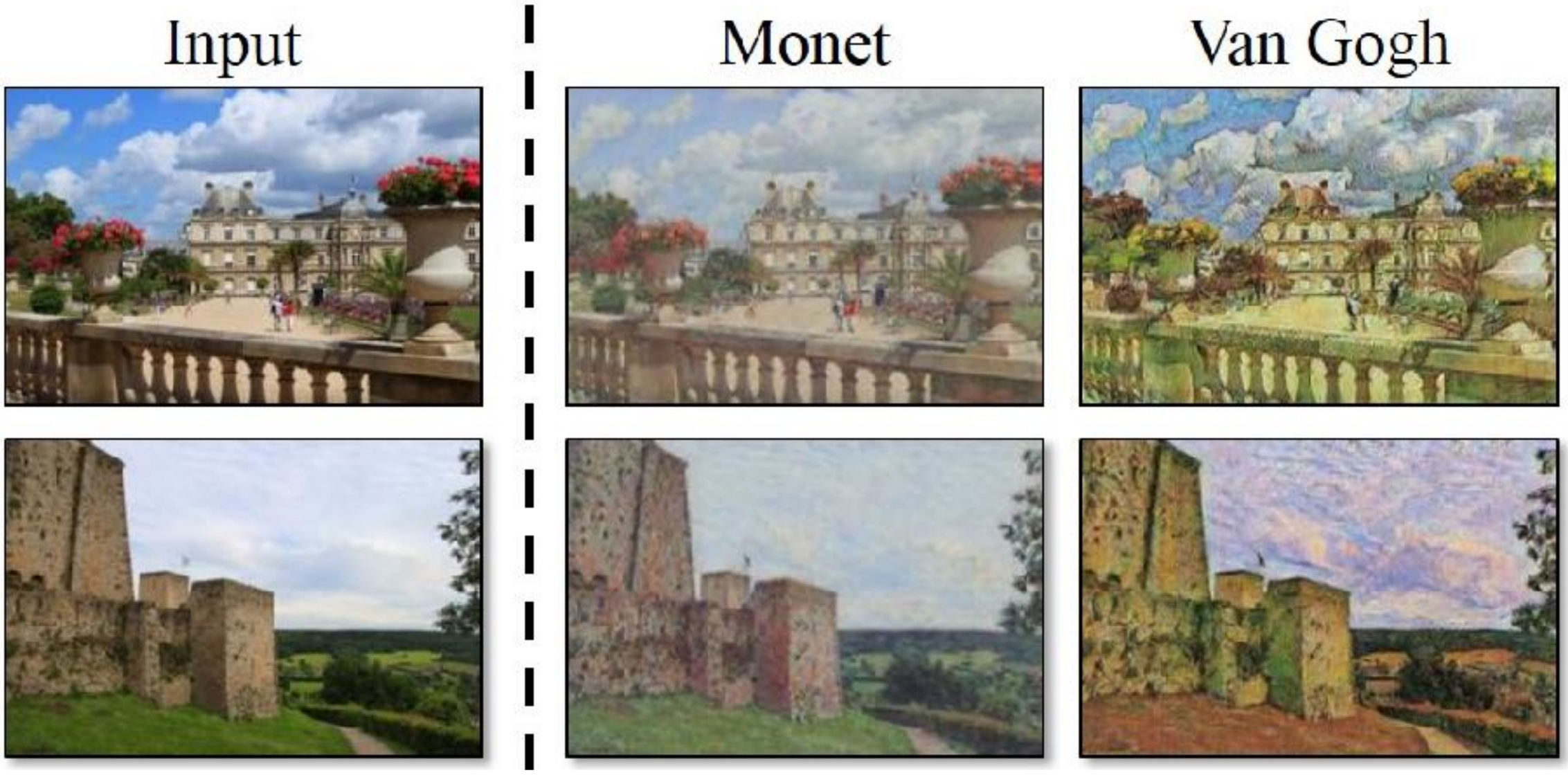


Portrait of Edmond Belamy, 2018. Sold for $432,500 on 25 October at Christie's in New York. Image © Obvious

Image-to-Image Translation

[Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

Artistic Style Transfer



[Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

Training Data Generation

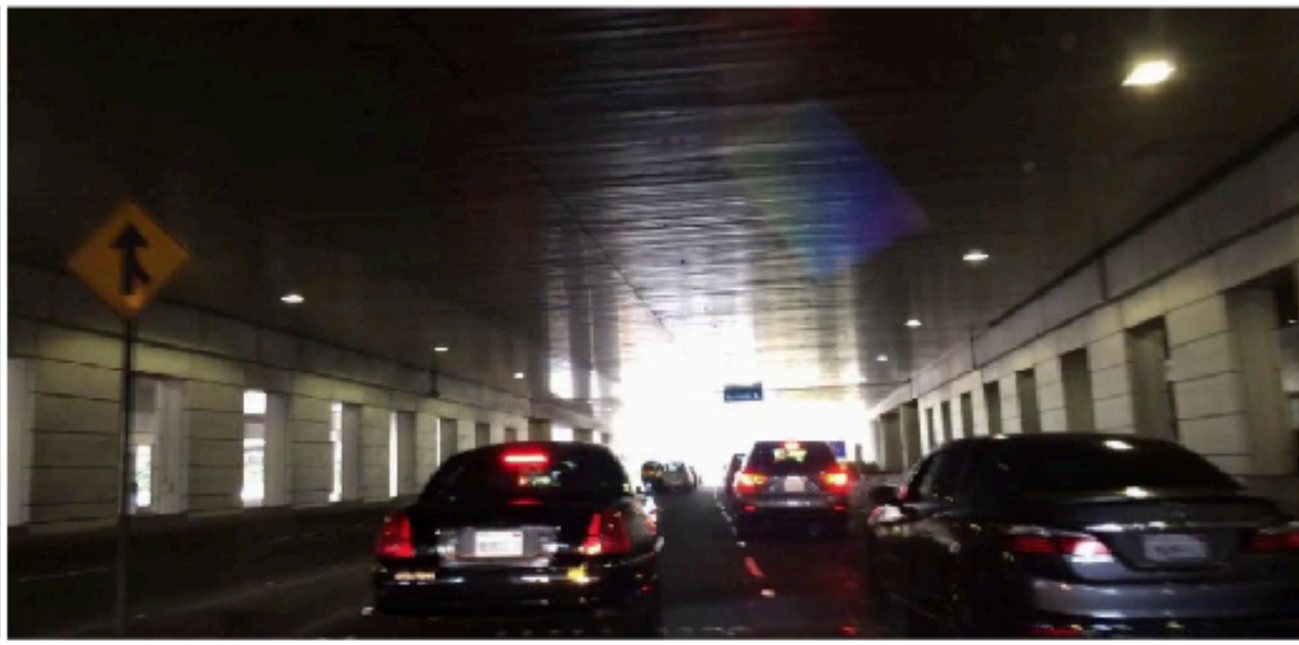
Input sunny image

AI-generated rainy image

[Liu et al., Unsupervised Image-to-Image Translation Networks, NIPS 2017]

Domain Adaptation

CityScapes (Germany)



San Francisco

Source domain:
Plenty of labelled data

Target domain:
Little to no labelled data

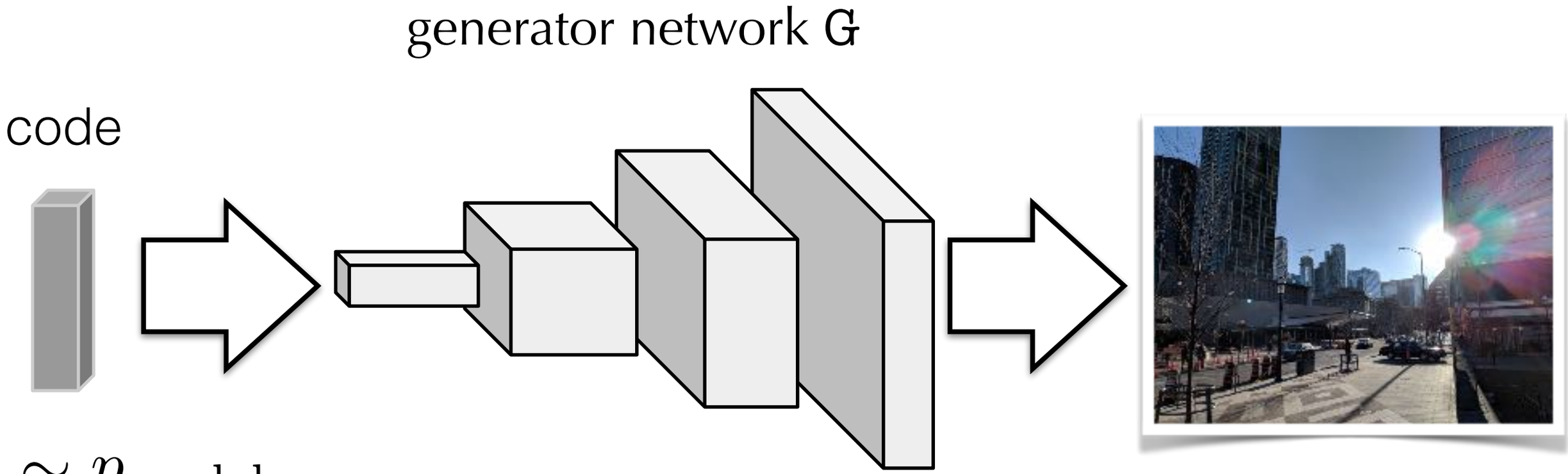
slide credit: Judy Hoffman

Image Generation as Sampling Process

Generate image = pick image from distribution of real images

image from http://3drepresentation.stanford.edu/

Generative Neural Networks



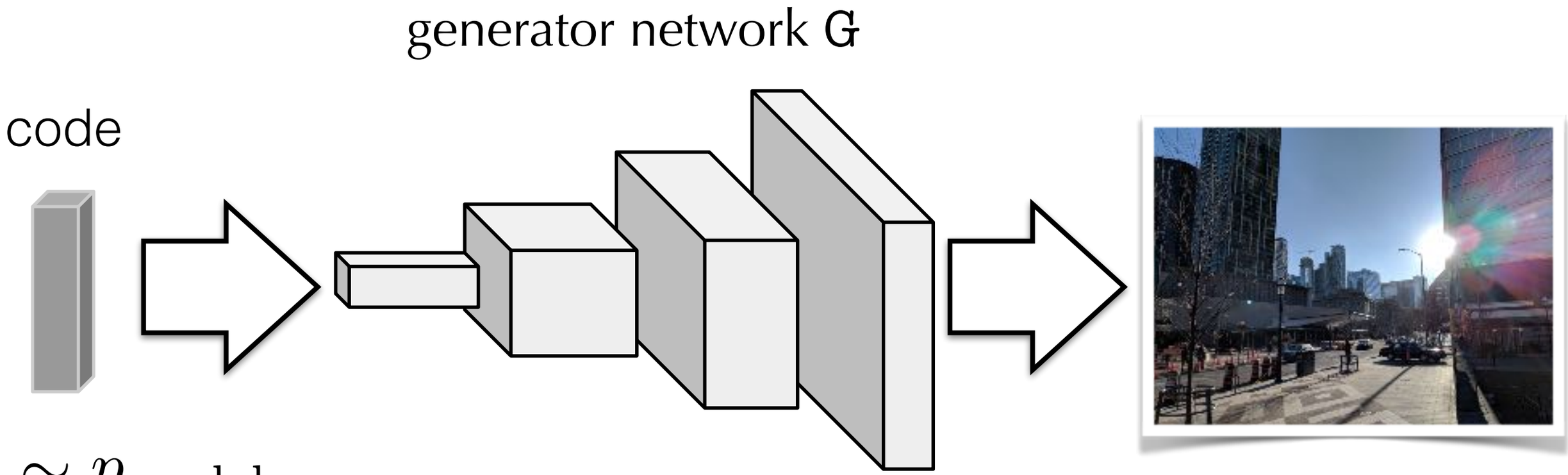
$\mathbf{z} \sim p_{\text{model}}$

normal distribution,
uniform distribution,
…

$\mathbf{x} \sim p_{\text{data}}$

(hopefully)

Generative Neural Networks



$\mathbf{z} \sim p_{\text{model}}$

normal distribution,
uniform distribution,
…

$\mathbf{x} \sim p_{\text{data}}$

(hopefully)

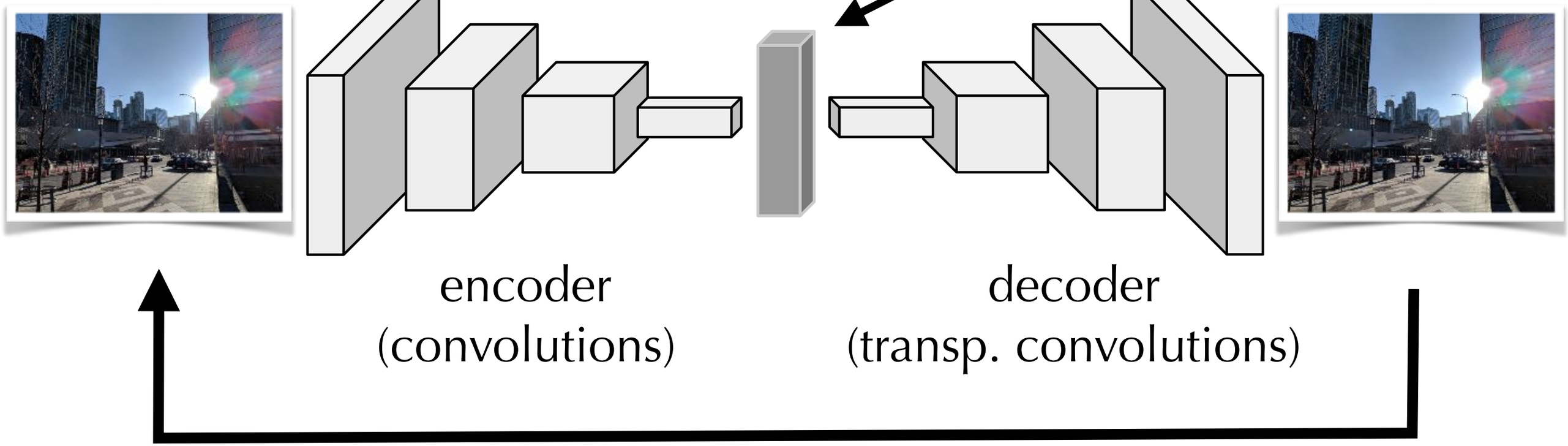
How to make sure that this mapping produces realistic images?

Variational Autoencoder (VAE)

structured latent space

$\mathbf{z} \sim p_{\text{model}}$

lower-dimensional than input (**bottleneck**)



encoder (convolutions)

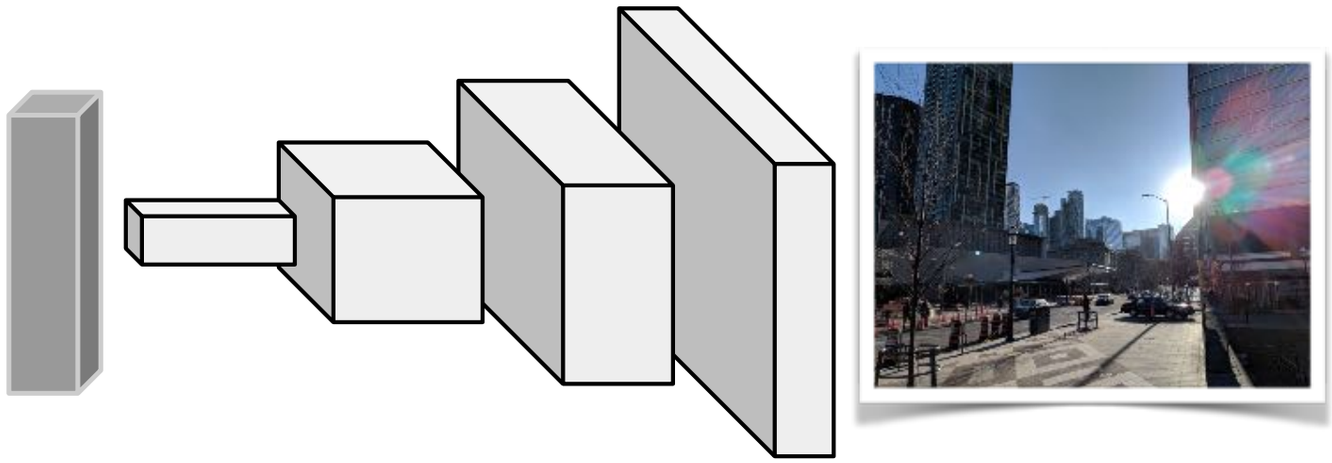
decoder (transp. convolutions)

reconstruction loss

Image Generation via a VAE

sample point in latent space

$$\mathbf{z} \sim p_{\text{model}}$$



decoder
(transp. convolutions)

$$p(\mathbf{x}|\mathbf{z})$$

$$\mathbf{x} \sim p_{\text{data}}$$

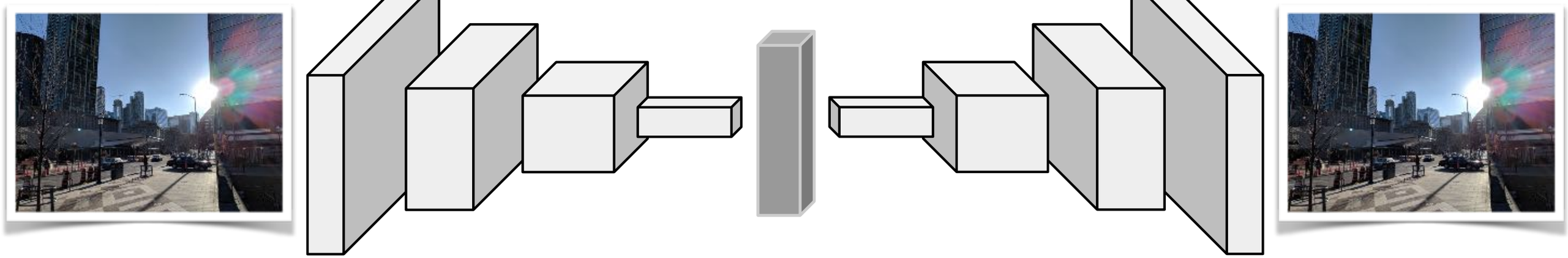
(ideally)

learn mapping from simple
to complex distribution

Image Generation via a VAE

latent space

$$\mathbf{z} \sim p_{\text{model}}$$



encoder
(convolutions)

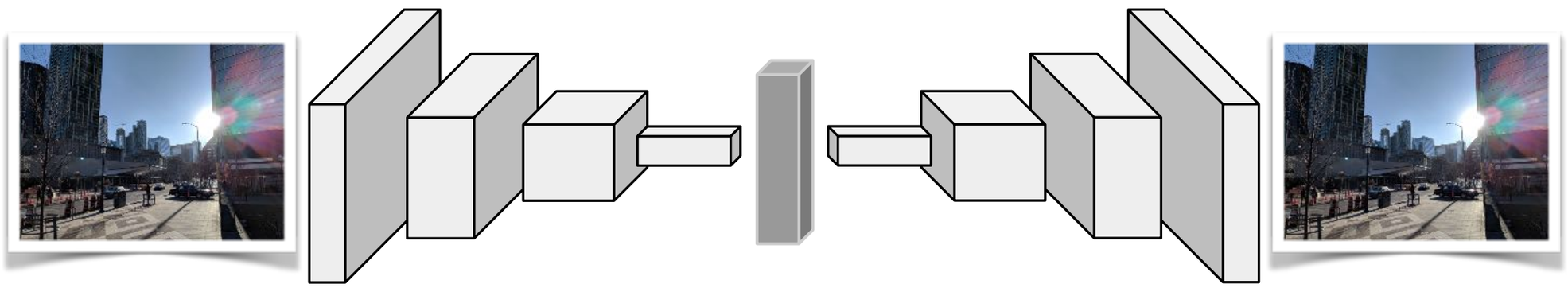
decoder
(transp. convolutions)

$\mathbf{x} \sim p_{\text{data}}$ $q(\mathbf{z}|\mathbf{x})$ $p(\mathbf{x}|\mathbf{z})$ $\mathbf{x} \sim p_{\text{data}}$
(ideally)

learn structure of
latent space

learn mapping from simple
to complex distribution

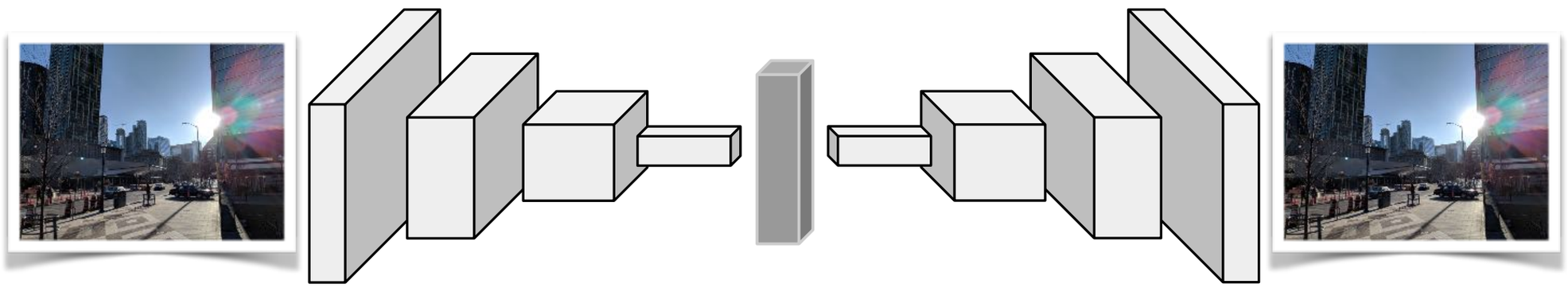
Theory



Evidence Lower BOund (ELBO):

$$\log(p(\mathbf{x})) \geq E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

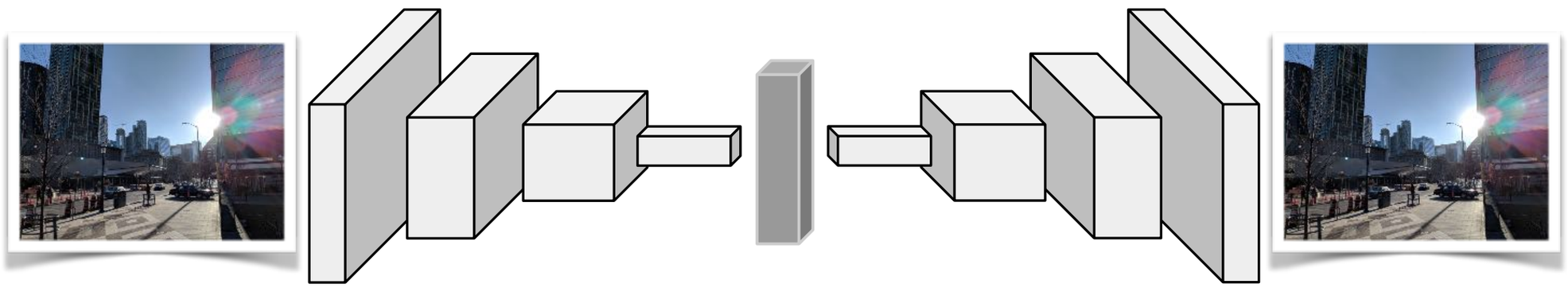
Theory



Evidence Lower BOund (ELBO):

$$\log(p(\mathbf{x})) \geq \underbrace{E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))]}_{\text{likelihood}} - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

Theory

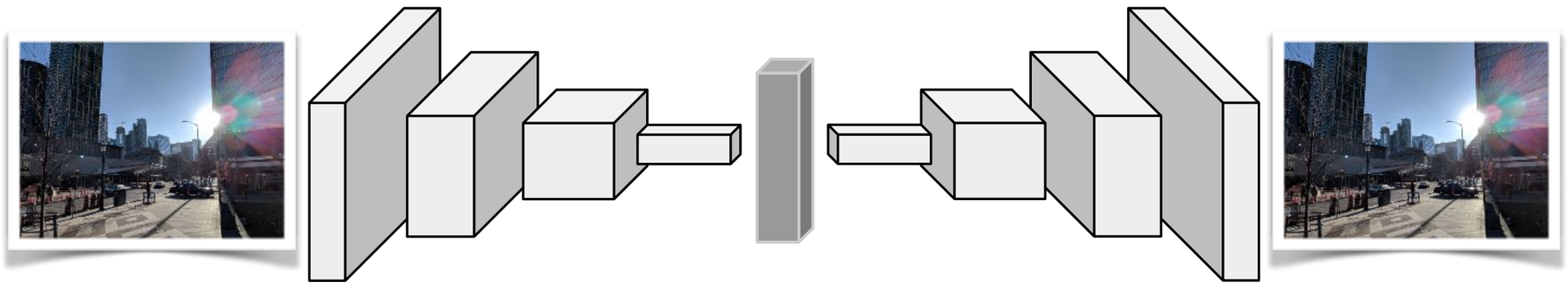


Evidence Lower BOund (ELBO):

$$\log(p(\mathbf{x})) \geq \underbrace{E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))]}_{\text{likelihood}} - \underbrace{D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{KL divergence}}$$

Objective Function

$$\mathbf{z} \sim p_{\text{model}}$$



encoder
$q(\mathbf{z}|\mathbf{x})$

decoder
$p(\mathbf{x}|\mathbf{z})$

Maximize

$$E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

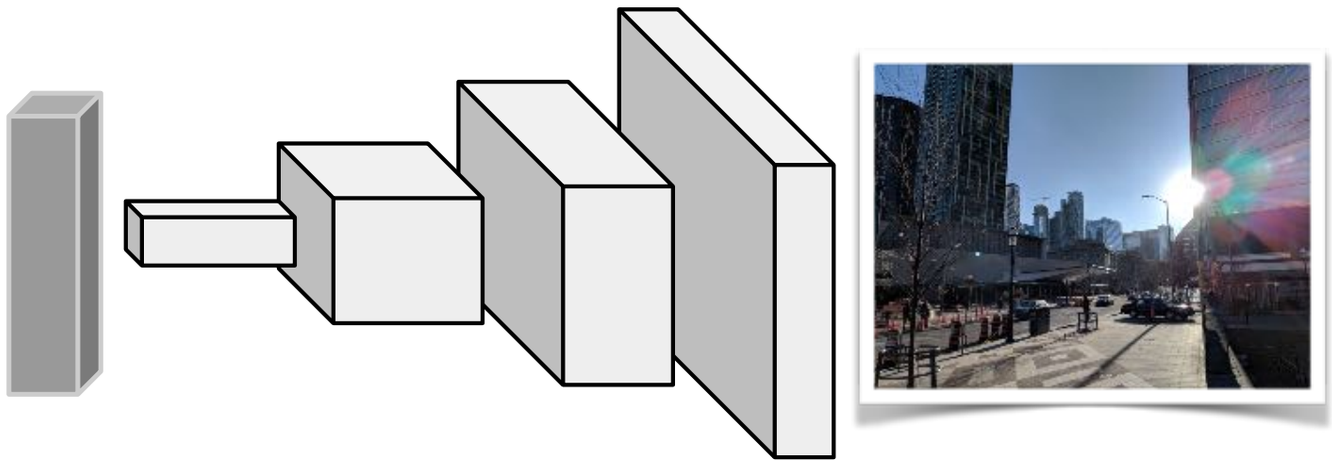
reconstruction ``loss'',
e.g., L2 norm (normal distribution)

similarity between learned
and sampling distribution

At Test Time

sample point from latent space

$$\mathbf{z} \sim p_{\text{model}}$$



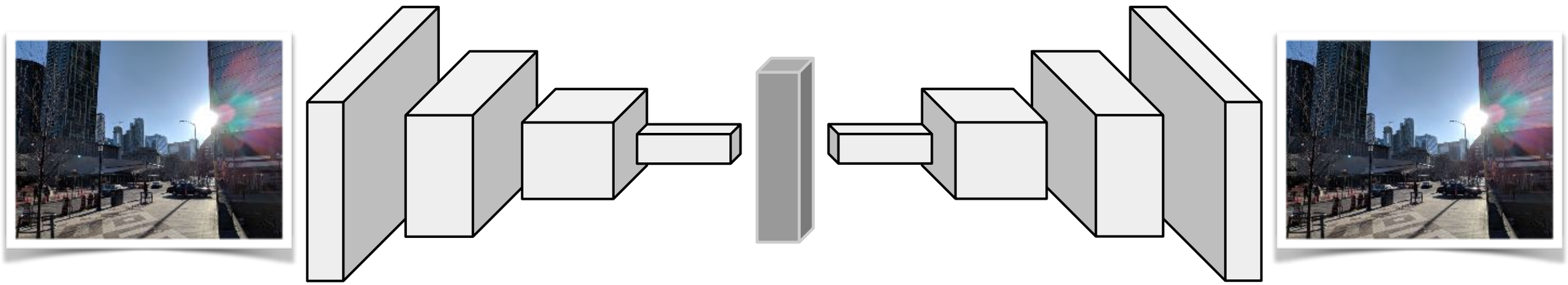
decoder
(transp. convolutions)

$$p(\mathbf{x}|\mathbf{z}) \qquad \mathbf{x} \sim p_{\text{data}}$$

use decoder to ``translate''
to image

VAE - Example

$$\mathbf{z} \sim p_{\text{model}} = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathrm{I})$$



encoder

$$q(\mathbf{z}|\mathbf{x})$$
$$= \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x})\mathrm{I})$$

multi-variate Gaussian distribution

decoder

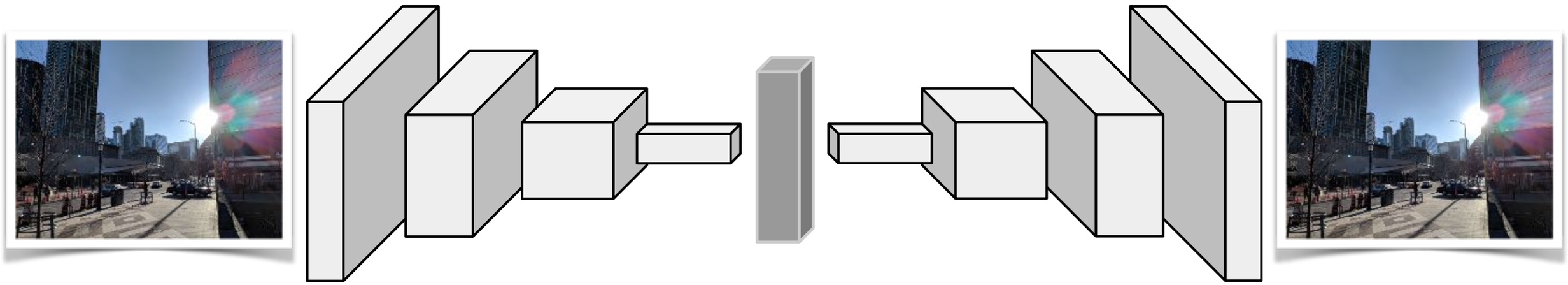
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu', \sigma'^2\mathrm{I})$$

least-squares solution

[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

VAE - Example

$$\mathbf{z} \sim p_{\text{model}} = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathrm{I})$$



encoder

$$q(\mathbf{z}|\mathbf{x})$$

$$= \mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x})\mathrm{I})$$

multi-variate Gaussian distribution

decoder

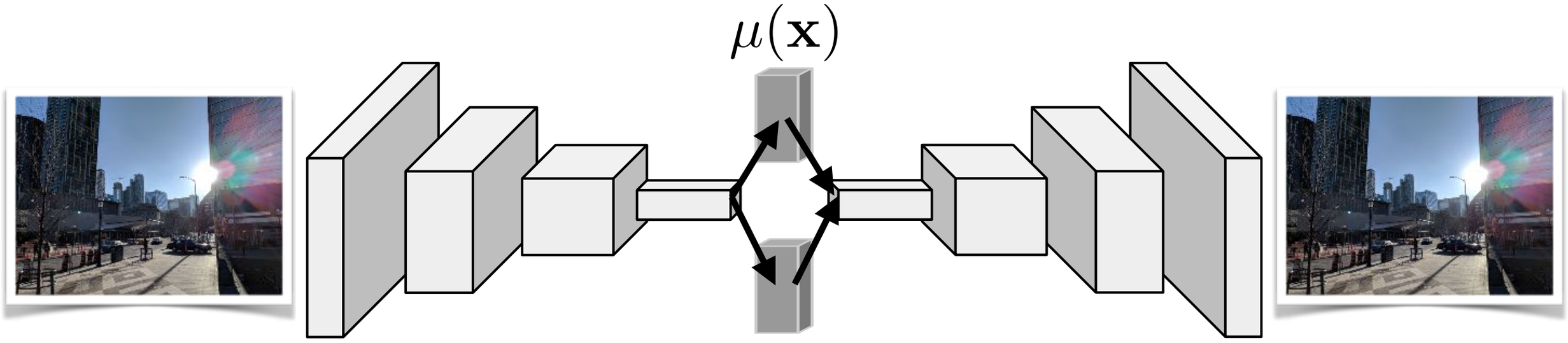
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu', \sigma'^2\mathrm{I})$$

least-squares solution

diagonal co-variance matrix

[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

VAE - Example



encoder $\mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x})\mathtt{I})$

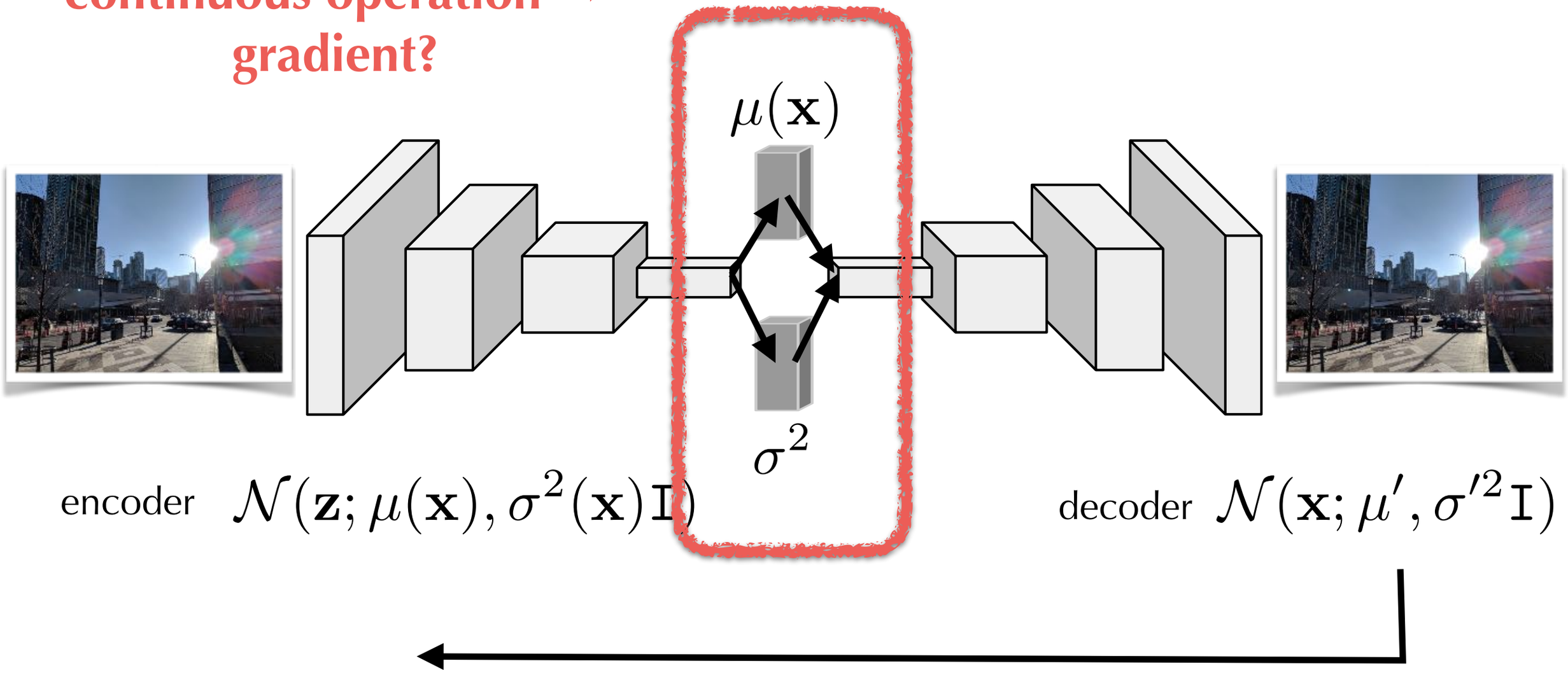
decoder $\mathcal{N}(\mathbf{x}; \mu', \sigma'^2\mathtt{I})$

back-propagation

[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

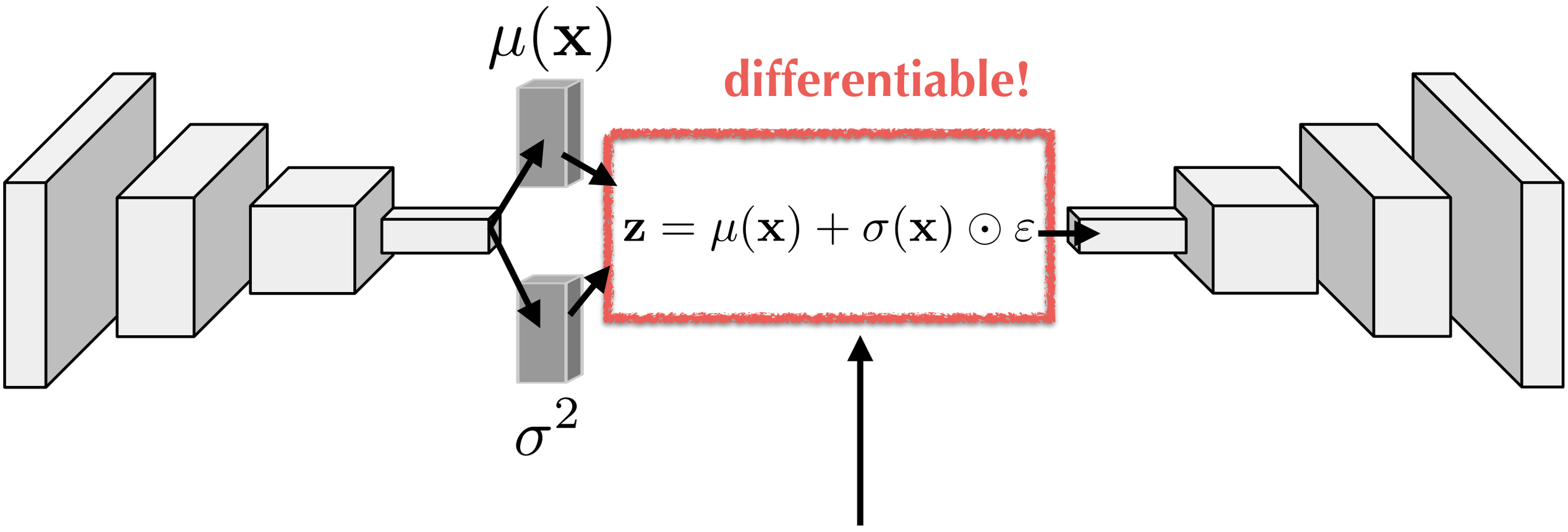
VAE - Example

sampling z is non-continuous operation → gradient?



[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

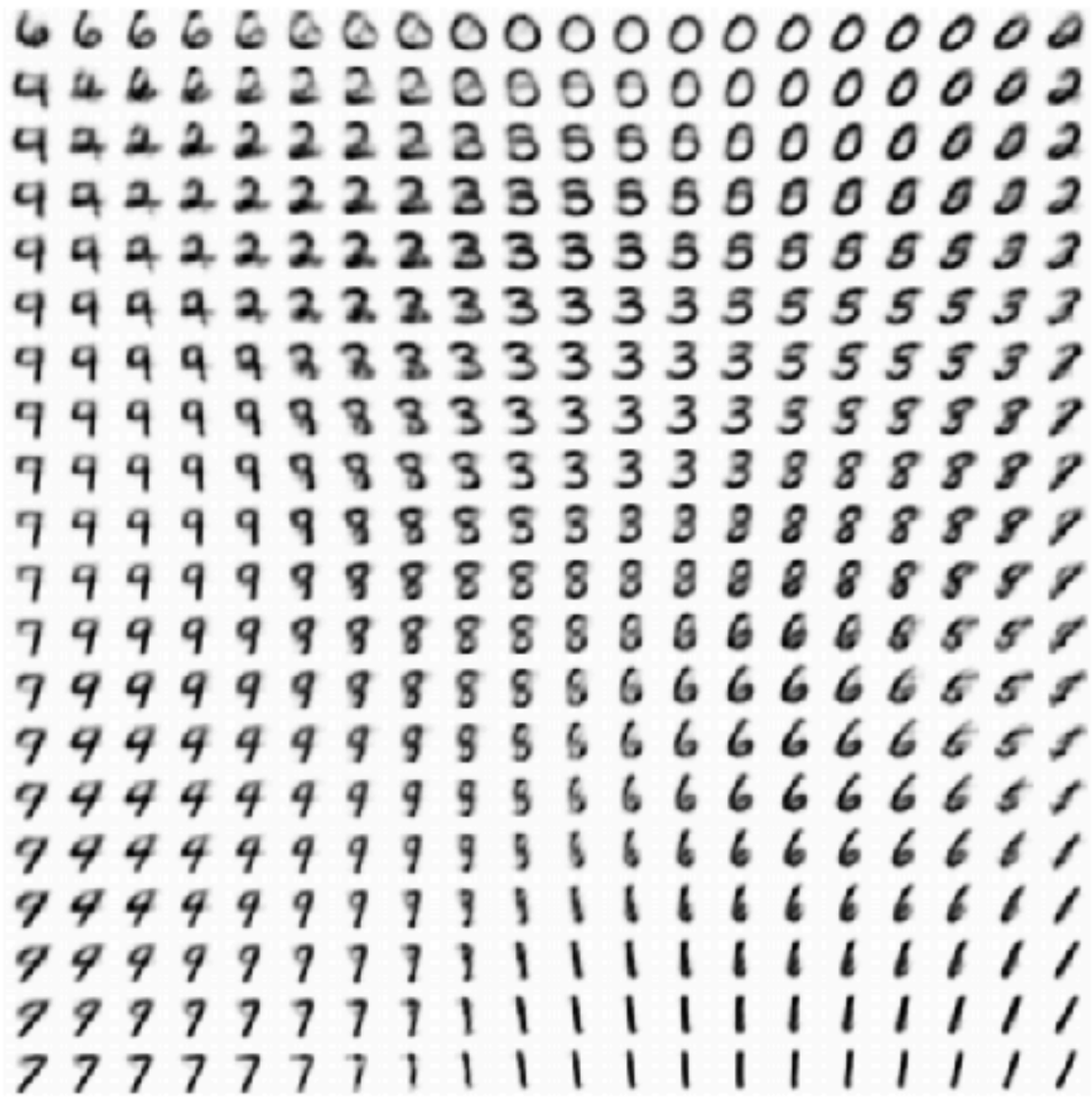
The Reparametrization Trick



[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

VAE Example - Results

Learned manifold (2D latent space)



[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

VAE Example - Results

Learned manifold (2D latent space)

[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

Today

Generative Neural Networks

Today

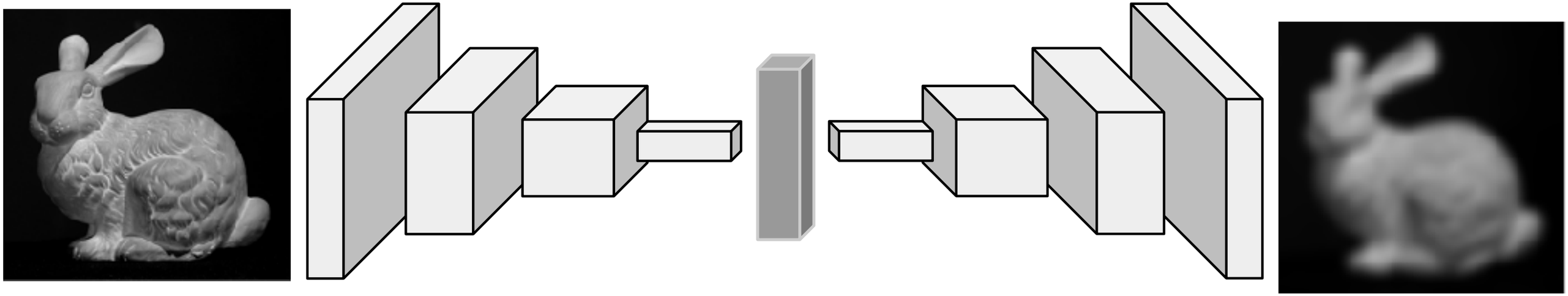
- Generative Adversarial Networks (GANs)
- (Unsupervised) Image Translation

Today

- **Generative Adversarial Networks (GANs)**
- (Unsupervised) Image Translation

Variational Autoencoder

$\mathbf{z} \sim p_{\text{model}}$ latent space



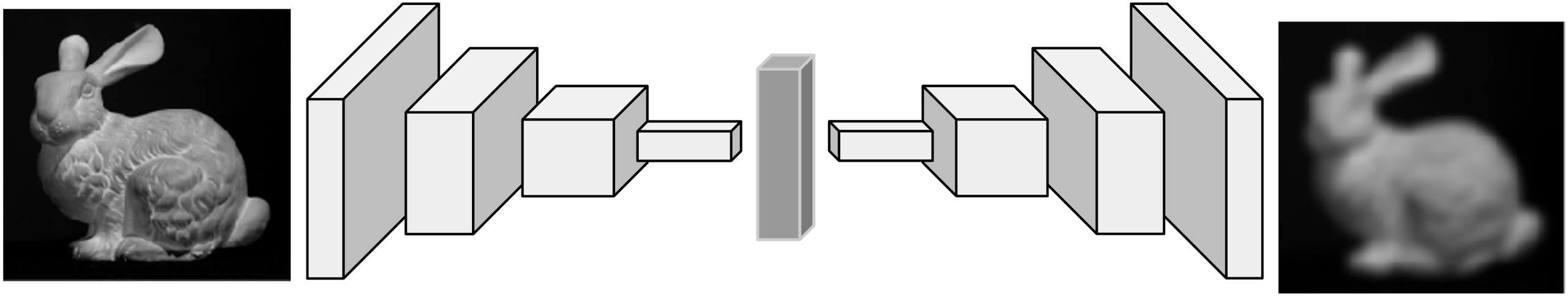
encoder $q(\mathbf{z}|\mathbf{x})$ decoder $p(\mathbf{x}|\mathbf{z})$

Maximize

$$E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \leq \log(p(\mathbf{x}))$$

Variational Autoencoder

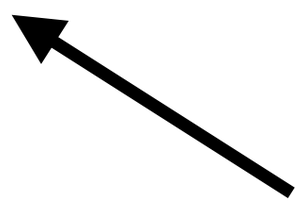
$\mathbf{z} \sim p_{\text{model}}$ latent space



encoder $q(\mathbf{z}|\mathbf{x})$ decoder $p(\mathbf{x}|\mathbf{z})$

Maximize

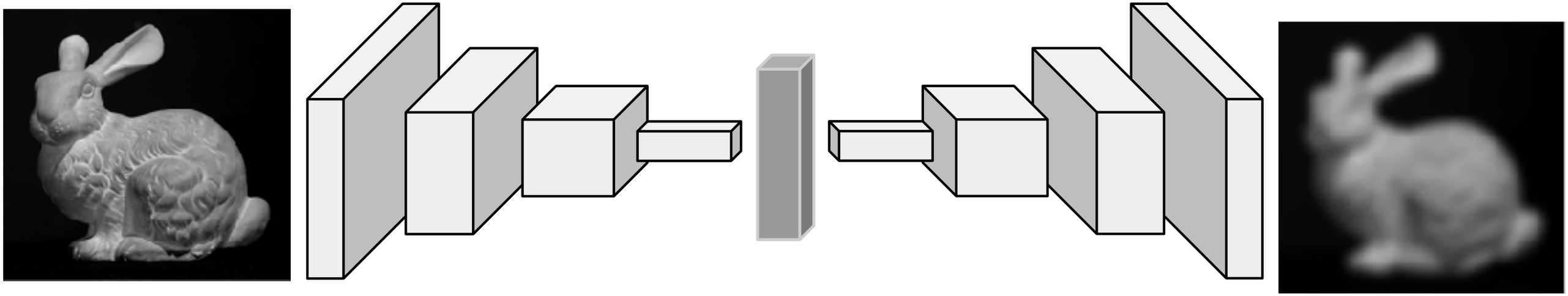
$$E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \leq \log(p(\mathbf{x}))$$



Typically a normal distribution → blurry results

Variational Autoencoder

$\mathbf{z} \sim p_{\text{model}}$ latent space



encoder $q(\mathbf{z}|\mathbf{x})$ decoder $p(\mathbf{x}|\mathbf{z})$

Maximize

$$E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \leq \log(p(\mathbf{x}))$$

Better loss functions?

Typically a normal distribution → blurry results

The Need for a Better Loss Function

original output 1 output 2

[Doersch, Tutorial on Variational Autoencoders, arXiv:1606.05908]

The Need for a Better Loss Function

original output 1 output 2

Output 2 better, but slightly shifted → larger loss under Gaussian distribution

[Doersch, Tutorial on Variational Autoencoders, arXiv:1606.05908]

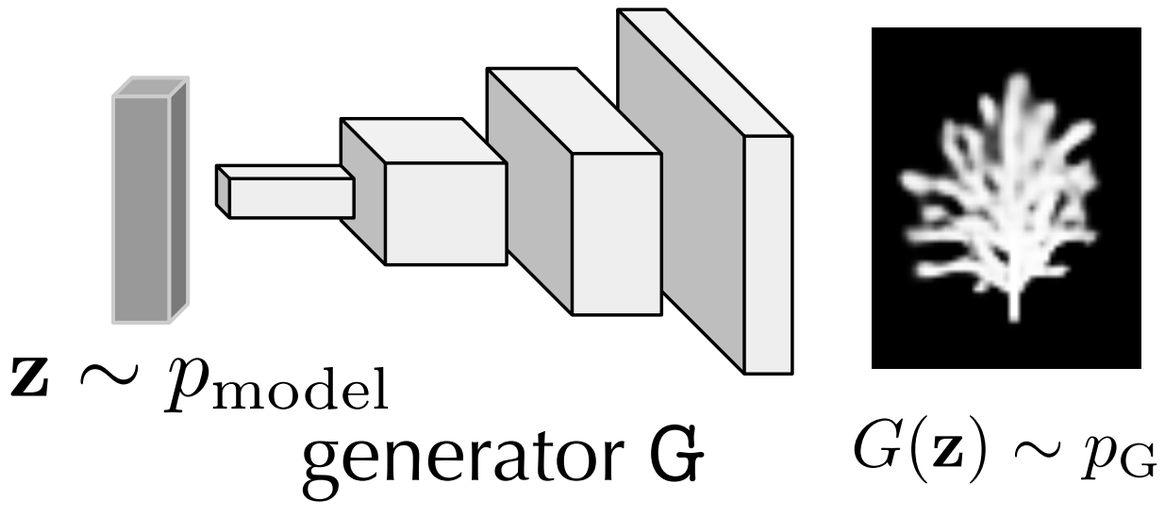
Generative Adversarial Networks (GANs)

Motivation: Learn loss function

[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Generative Adversarial Networks (GANs)

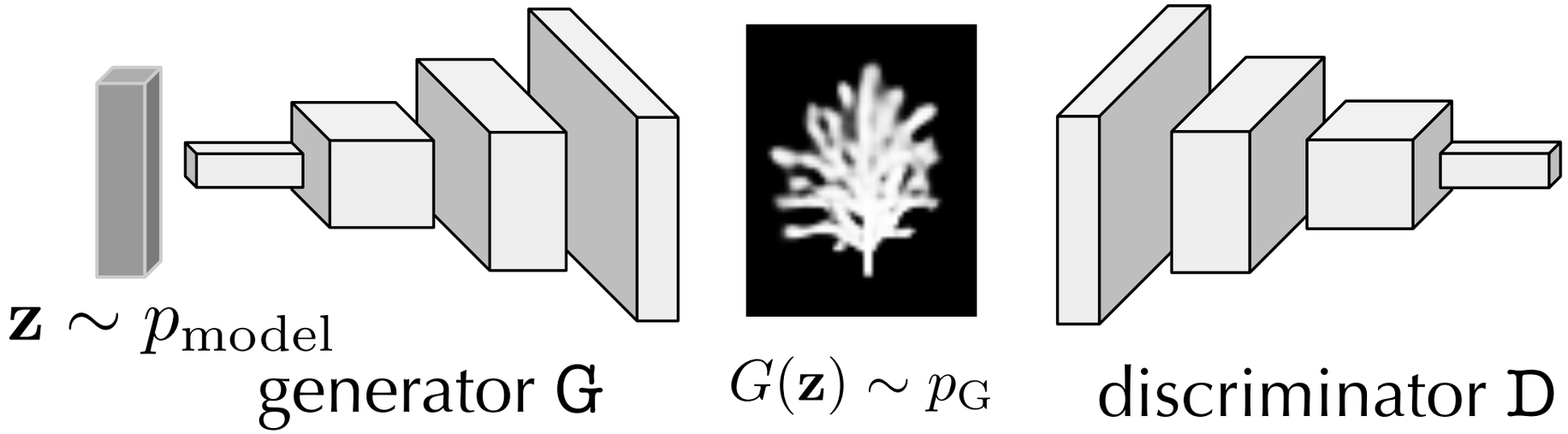
Motivation: Learn loss function



[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Generative Adversarial Networks (GANs)

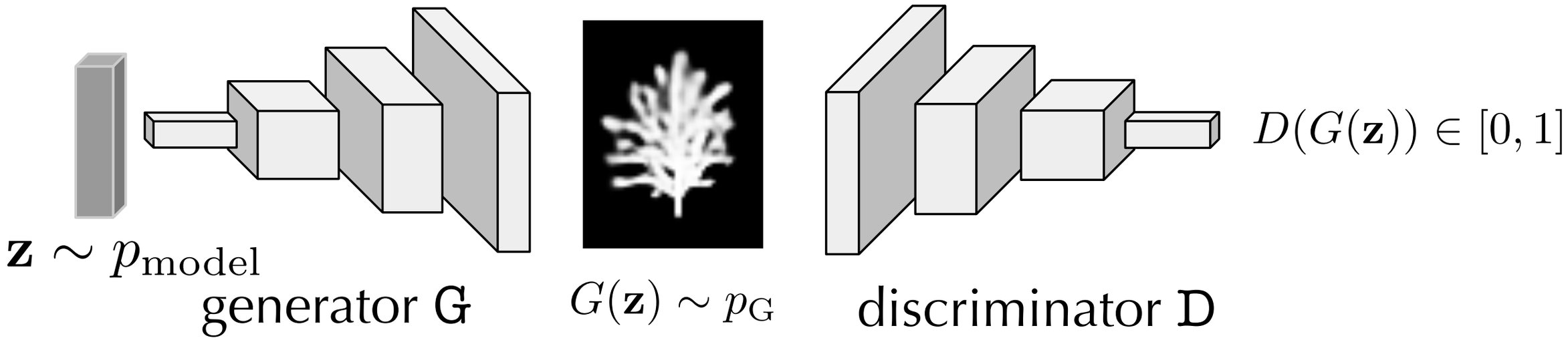
Motivation: Learn loss function



[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Generative Adversarial Networks (GANs)

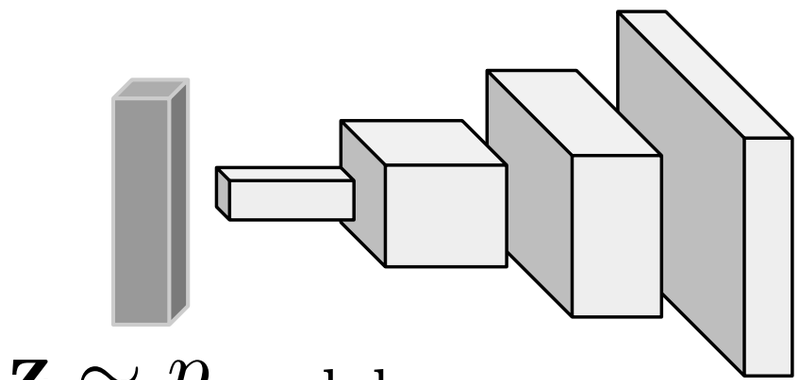
Motivation: Learn loss function



[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Generative Adversarial Networks (GANs)

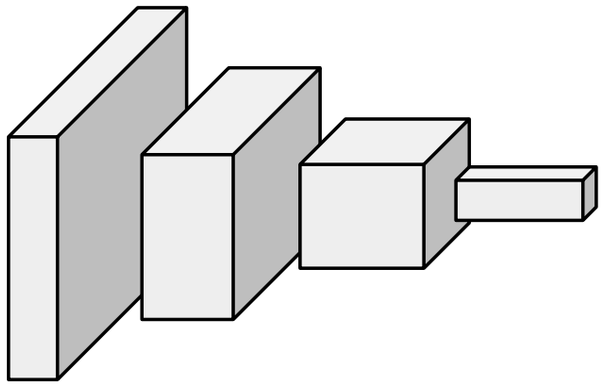
Motivation: Learn loss function



$\mathbf{z} \sim p_{\text{model}}$

generator G

$G(\mathbf{z}) \sim p_{\text{G}}$

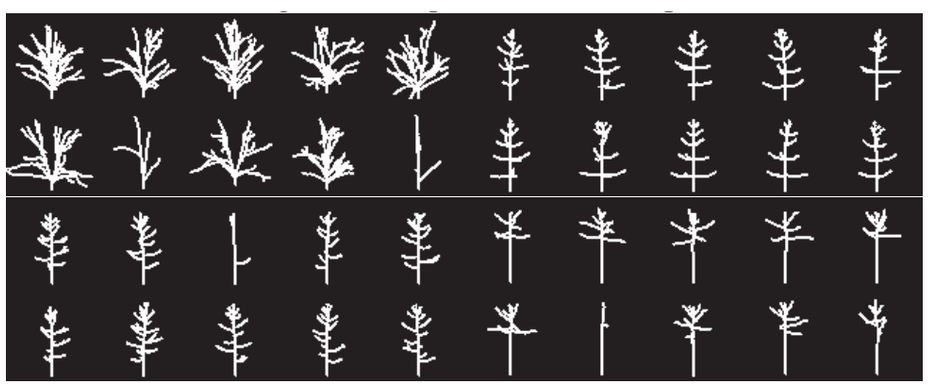


$D(G(\mathbf{z})) \in [0, 1]$

discriminator D

real data

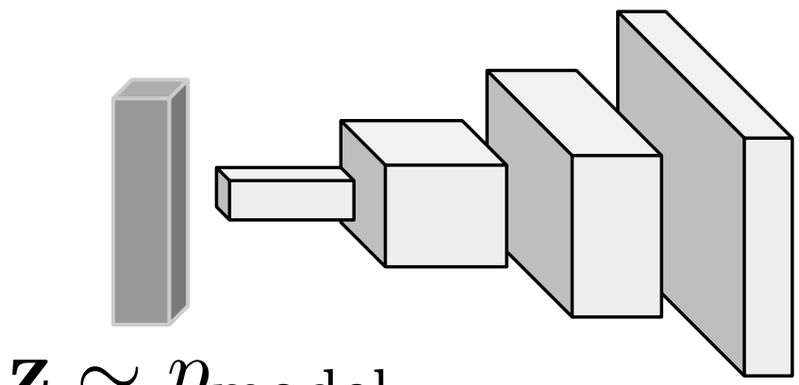
$\mathbf{x} \sim p_{\text{data}}$



[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

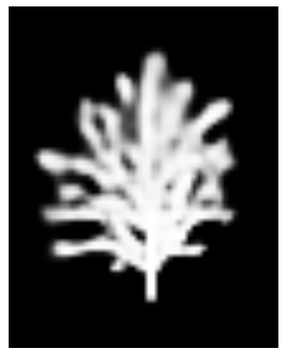
Generative Adversarial Networks (GANs)

Motivation: Learn loss function

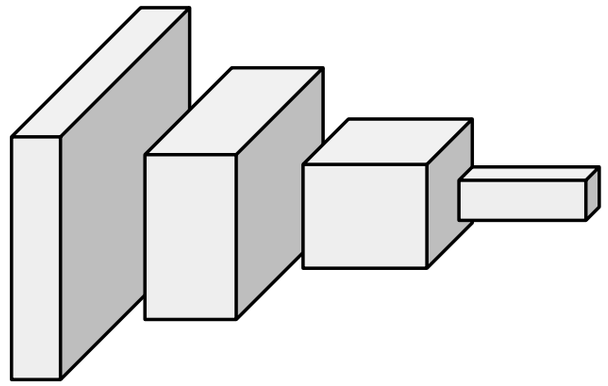


$\mathbf{z} \sim p_{\text{model}}$

generator G



$G(\mathbf{z}) \sim p_{\text{G}}$



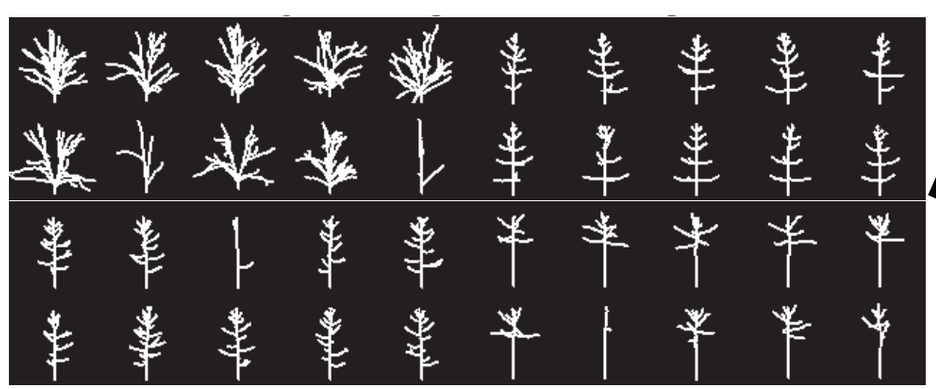
discriminator D

$D(G(\mathbf{z})) \in [0, 1]$

$D(x) \in [0, 1]$

real data

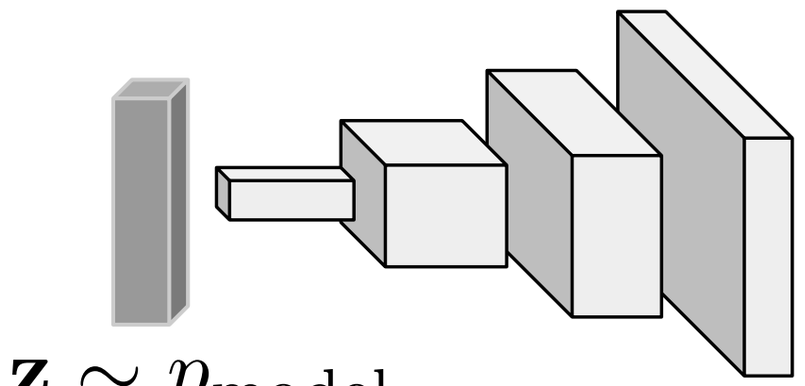
$\mathbf{x} \sim p_{\text{data}}$



[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

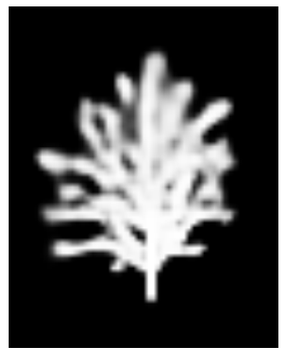
Generative Adversarial Networks (GANs)

Motivation: Learn loss function

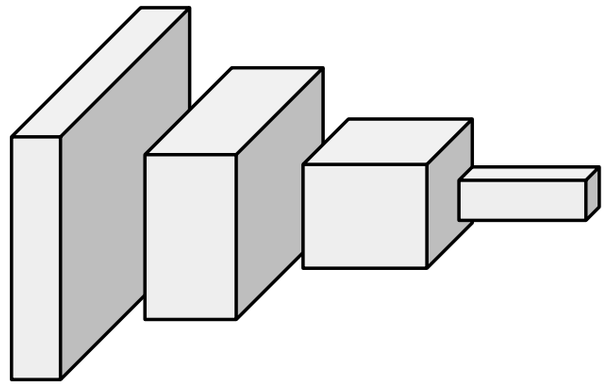


$\mathbf{z} \sim p_{\text{model}}$

generator G



$G(\mathbf{z}) \sim p_{\text{G}}$



discriminator D

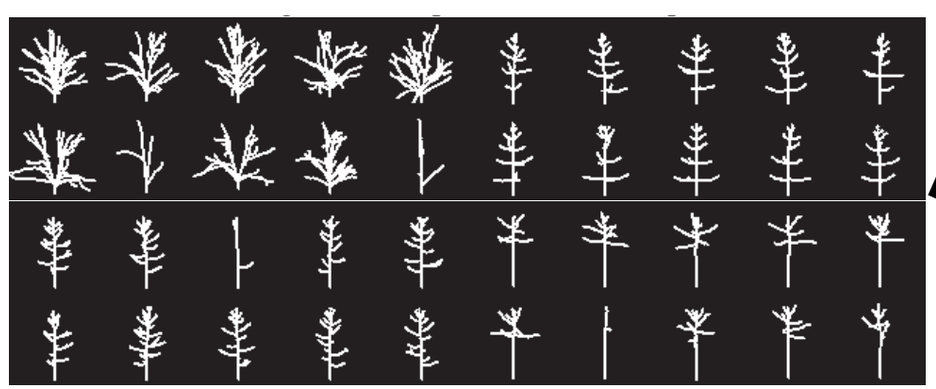
fake or real?

$D(G(\mathbf{z})) \in [0, 1]$

$D(x) \in [0, 1]$

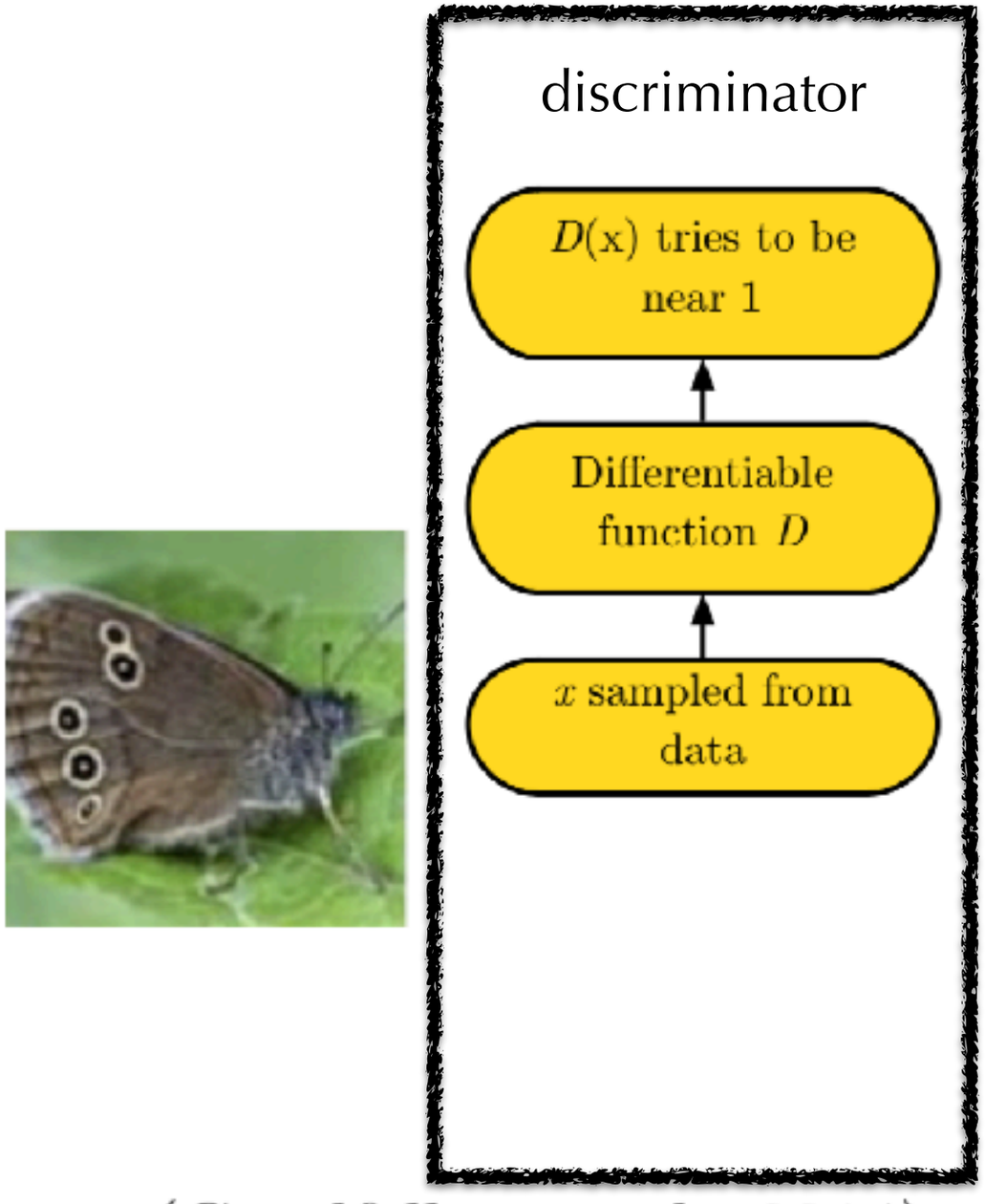
real data

$\mathbf{x} \sim p_{\text{data}}$



[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

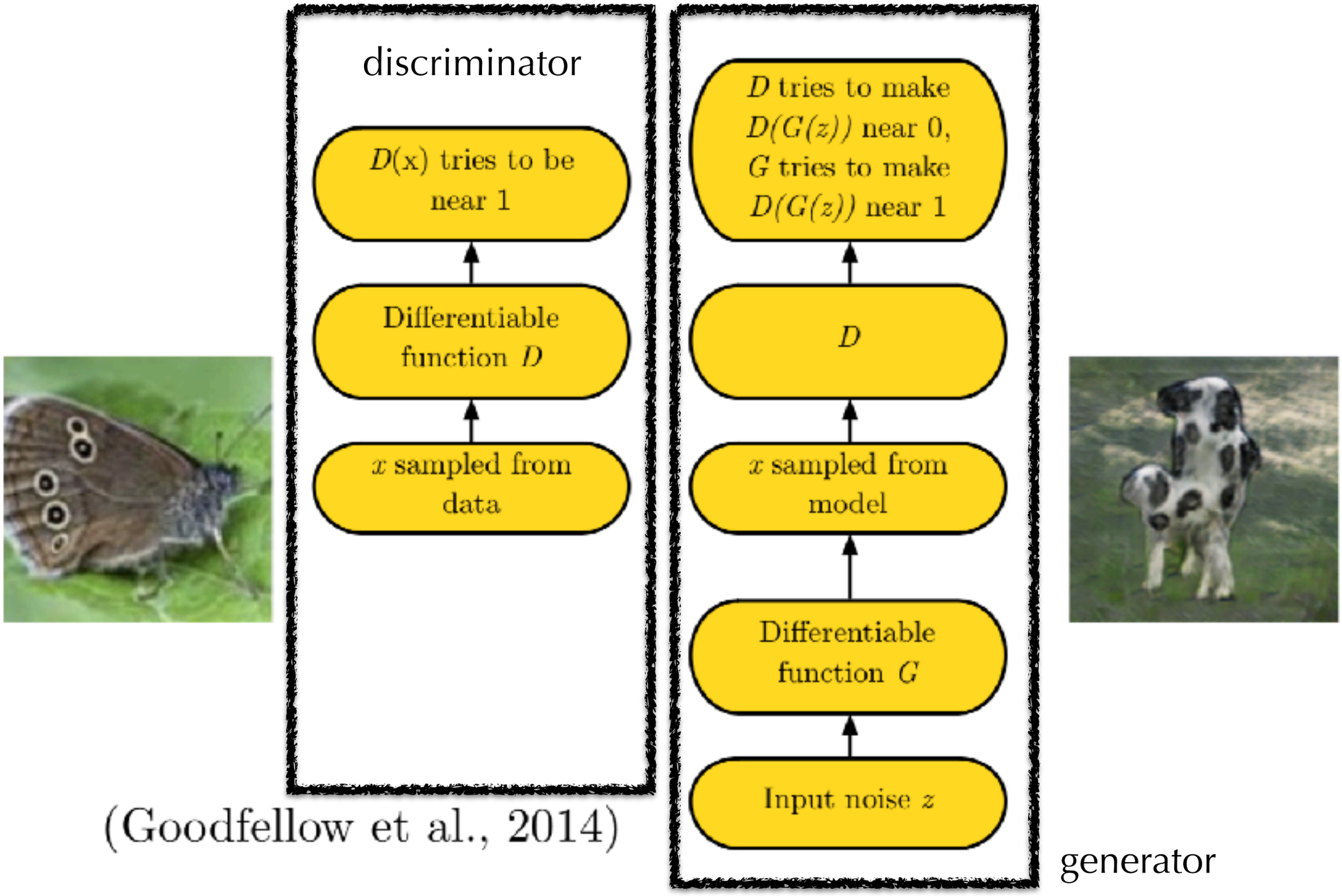
Generative Adversarial Networks (GANs)



(Goodfellow et al., 2014)

slide credit: Ian Goodfellow

Generative Adversarial Networks (GANs)



(Goodfellow et al., 2014)

slide credit: Ian Goodfellow

Game Theoretic View

- Discriminator $\mathbb{D}$ tries to distinguish real and fake images

Game Theoretic View

- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look “as real as possible”

Game Theoretic View

- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look "as real as possible"
- G and D play zero-sum minimax game:

Game Theoretic View

- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look "as real as possible"
- G and D play zero-sum minimax game:

$$\max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$$

Game Theoretic View

- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look "as real as possible"
- G and D play zero-sum minimax game:

$$\max_{D} \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$$

Game Theoretic View

- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look "as real as possible"
- G and D play zero-sum minimax game:

$$\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$$

Game Theoretic View

- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look "as real as possible"
- G and D play zero-sum minimax game:

$$\min_{G} \max_{D} \quad \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z} \sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$$

Game Theoretic View

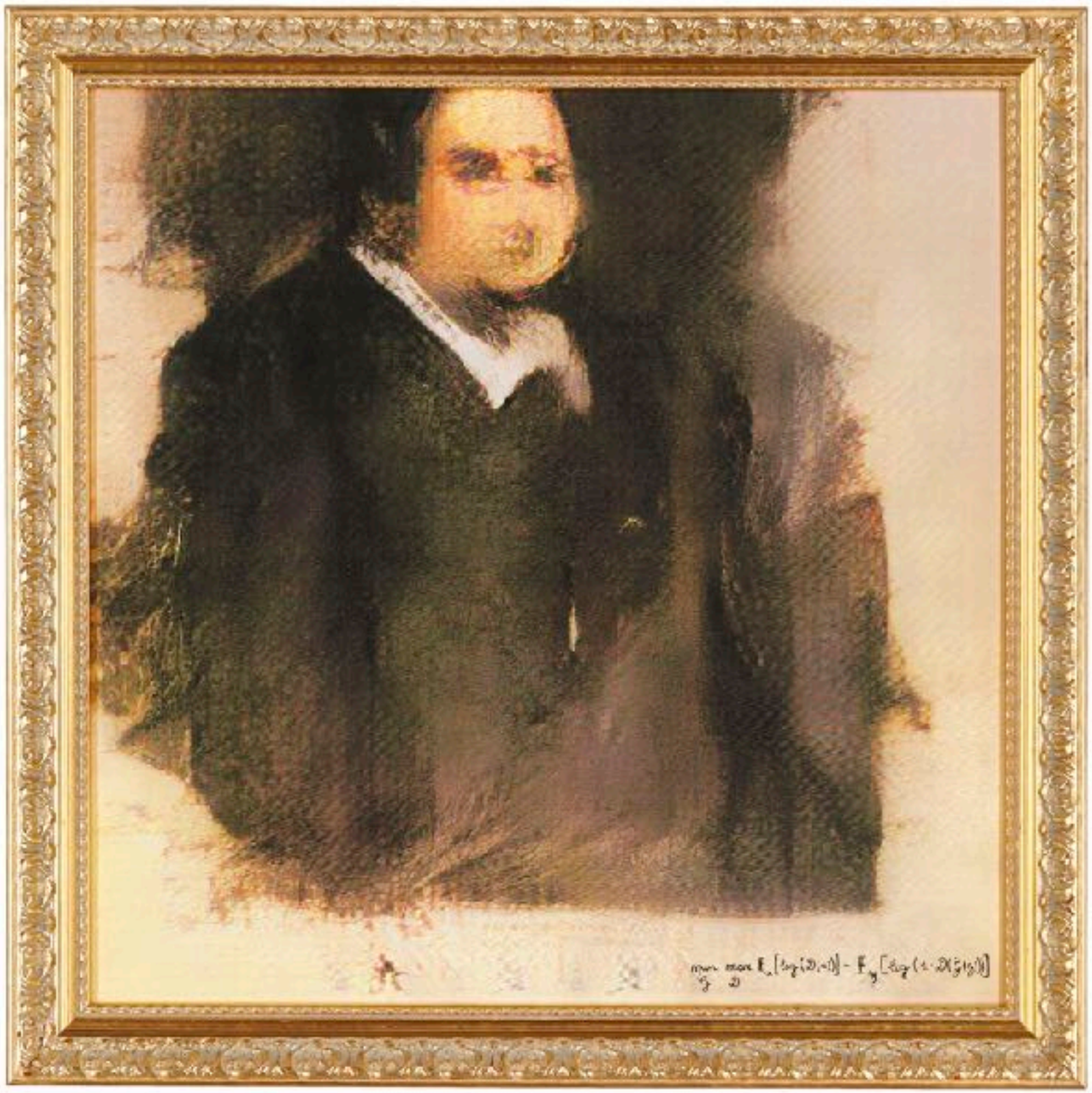
- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look "as real as possible"
- G and D play zero-sum minimax game:

$$\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$$

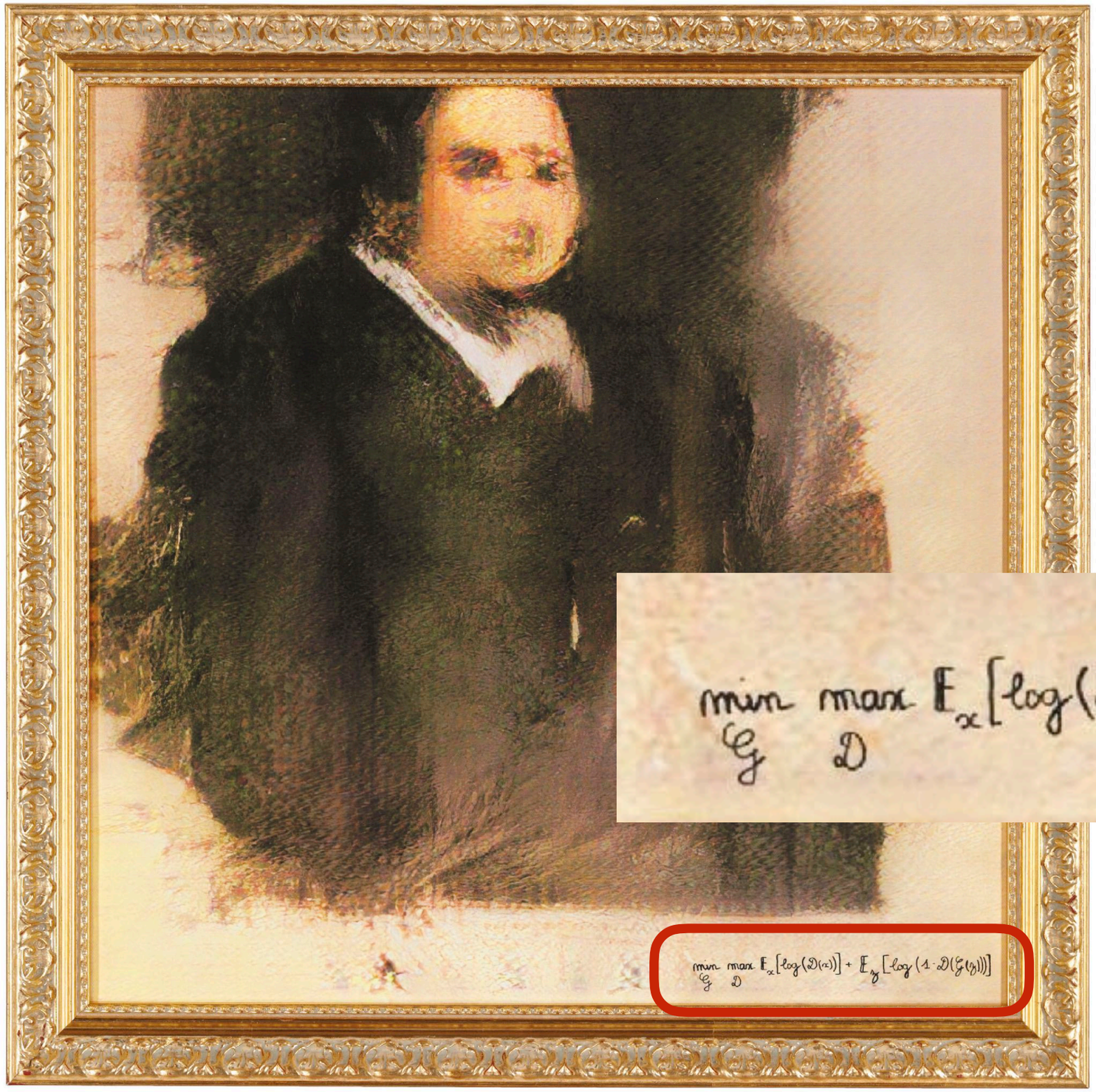
- Equilibrium / global optimum at $p_{\text{data}} = p_G$, i.e., when real and synthetic images are indistinguishable for D:

$$d(\mathbf{x}) = 1/2 \;\; \forall \mathbf{x} \sim p_{\text{data}}, \mathbf{x} \sim p_G$$

Generating Art Revisited



Generating Art Revisited



$$\min_G \max_D \mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))]$$

Training GANs

---

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

Train discriminator D

end for

Train generator G

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

[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Training GANs

for number of training iterations **do**

  for k steps **do**

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

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

  end for

Train generator G

end for

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

[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Training GANs

for number of training iterations **do**

  for k steps **do**

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

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

  end for

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

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

end for

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

[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Training GANs is Hard

- Need to carefully **balance the generator and discriminator**

Training GANs is Hard

- Need to carefully **balance the generator and discriminator**
  - D becomes too powerful too soon → G cannot improve (gradient vanishes)

Training GANs is Hard

- Need to carefully **balance the generator and discriminator**
  - D becomes too powerful too soon → G cannot improve (gradient vanishes)
  - D not powerful enough → G does not improve

Training GANs is Hard

- Need to carefully **balance the generator and discriminator**
  - D becomes too powerful too soon → G cannot improve (gradient vanishes)
  - D not powerful enough → G does not improve
  - Black magic: Schedule training of generator and discriminator

Training GANs is Hard

- Need to carefully **balance the generator and discriminator**
  - D becomes too powerful too soon → G cannot improve (gradient vanishes)
  - D not powerful enough → G does not improve
    - Black magic: Schedule training of generator and discriminator
- **Divergence**: GANs might not train at all

Training GANs is Hard

- Need to carefully **balance the generator and discriminator**
  - D becomes too powerful too soon → G cannot improve (gradient vanishes)
  - D not powerful enough → G does not improve
    - Black magic: Schedule training of generator and discriminator
- **Divergence**: GANs might not train at all
- **Mode collapse**: Generated images are not diverse

Mode Collapse

Generated images span only small subspace

[Arjovsky et al., Wasserstein GAN, arXiv:1701.07875]

Training GANs is Hard

- Need to carefully **balance the generator and discriminator**
  - D becomes too powerful too soon → G cannot improve (gradient vanishes)
  - D not powerful enough → G does not improve
    - Black magic: Schedule training of generator and discriminator
- **Divergence**: GANs might not train at all
- **Mode collapse**: Generated images are not diverse
- List of tips & tricks for training here: https://github.com/soumith/talks/blob/master/2017-ICCV_Venice/How_To_Train_a_GAN.pdf

Why is Training GANs so Hard?

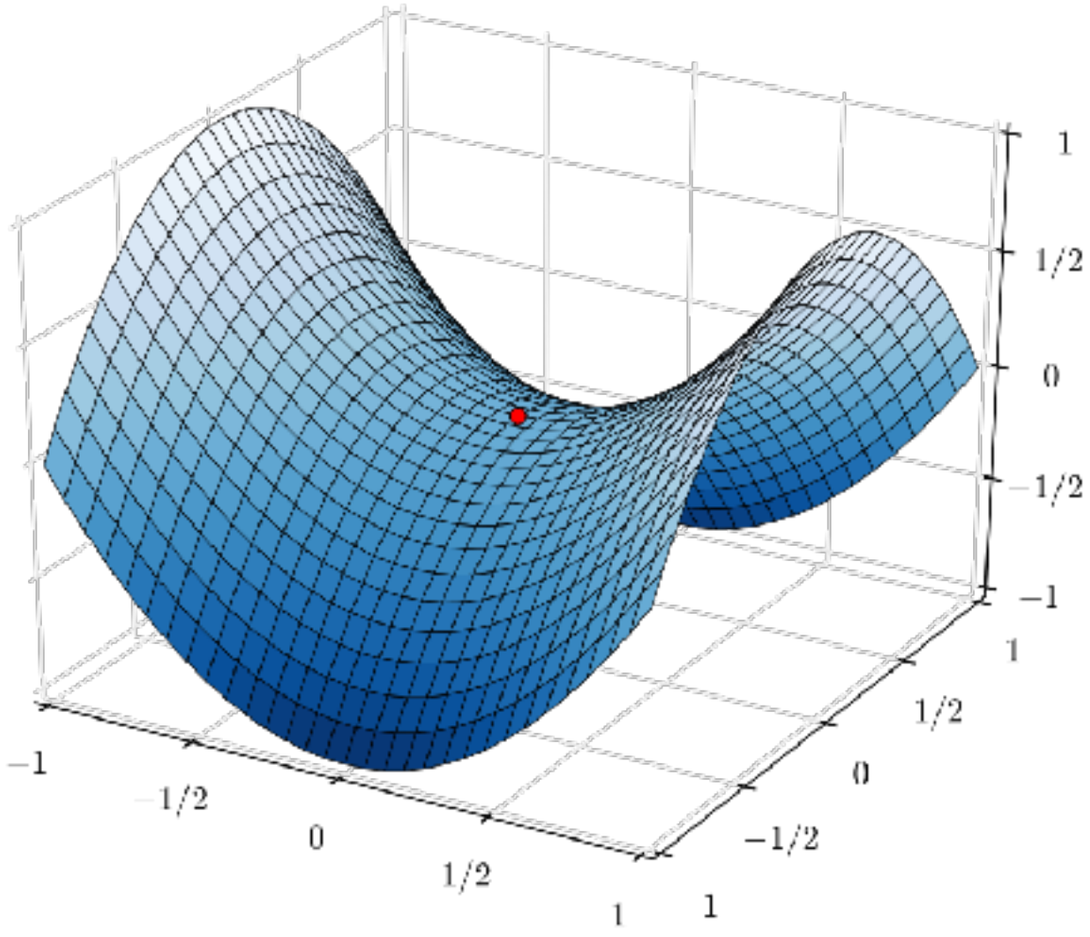


image: © Nicoguaro (CC Attribution Unported 3.0)

GAN equilibria are saddle points

$$\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$$

[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Why is Training GANs so Hard?

- GAN objective:
  $\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$

Why is Training GANs so Hard?

- GAN objective:
  $\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$
- Optimal discriminator D* for given G:

$$D^*_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}$$

Why is Training GANs so Hard?

- GAN objective:
  $\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$
- Optimal discriminator D* for given G:

$$D^*_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}$$

- GAN objective becomes

$$\min_G \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log\left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right) + \mathbb{E}_{\mathbf{x}\sim p_G} \log\left(\frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right)$$

Why is Training GANs so Hard?

- GAN objective:
  $\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$
- Optimal discriminator D* for given G:

$$D^*_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}$$

- GAN objective becomes

$$\min_G \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log\left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right) + \mathbb{E}_{\mathbf{x}\sim p_G} \log\left(\frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right)$$

- Utilizing the Kullback-Leibler (KL) divergence

Why is Training GANs so Hard?

- GAN objective:
  $\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$
- Optimal discriminator D* for given G:

$$D^*_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}$$

- GAN objective becomes

$$\min_G \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log\left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right) + \mathbb{E}_{\mathbf{x}\sim p_G} \log\left(\frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right)$$

- Utilizing the Kullback-Leibler (KL) divergence

$$\min_G \quad \text{KL}\left(p_{\text{data}} || \frac{p_{\text{data}} + p_G}{2}\right) + \text{KL}\left(p_G || \frac{p_{\text{data}} + p_G}{2}\right) - \log(4)$$

Why is Training GANs so Hard?

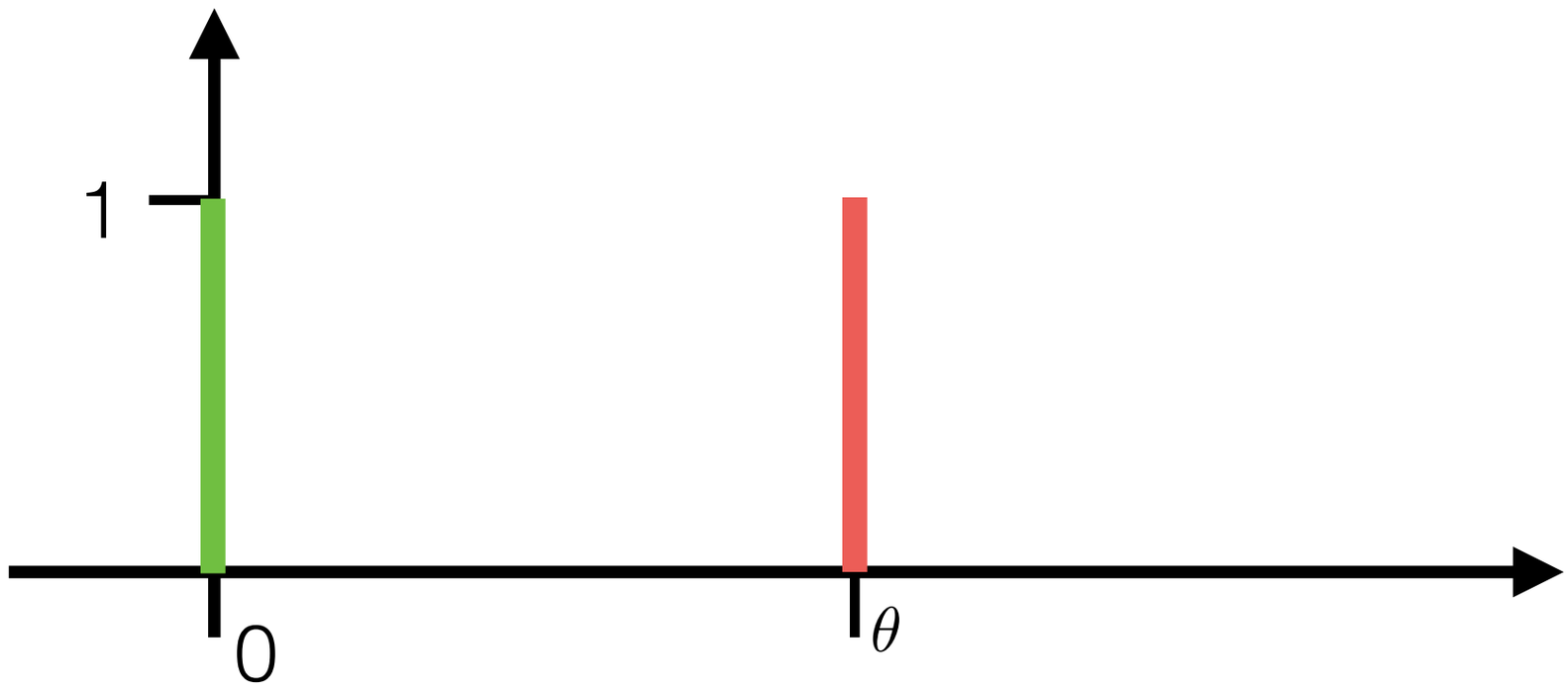
- GAN objective:
$$\min_G \max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log(d(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_{\text{model}}} \log(1 - d(g(\mathbf{z})))$$
- Optimal discriminator D* for given G:
$$D^*_G(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}$$
- GAN objective becomes
$$\min_G \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log\left(\frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right) + \mathbb{E}_{\mathbf{x}\sim p_G} \log\left(\frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{G}}(\mathbf{x})}\right)$$
- Utilizing the Kullback-Leibler (KL) divergence
$$\min_G \quad \underbrace{\text{KL}\left(p_{\text{data}}||\frac{p_{\text{data}} + p_G}{2}\right) + \text{KL}\left(p_G||\frac{p_{\text{data}} + p_G}{2}\right)}_{2 \cdot \text{JSD}(p_{\text{data}}||p_G) \quad \text{Jensen-Shannon divergence}} - \log(4)$$

Why is Training GANs so Hard?

Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent

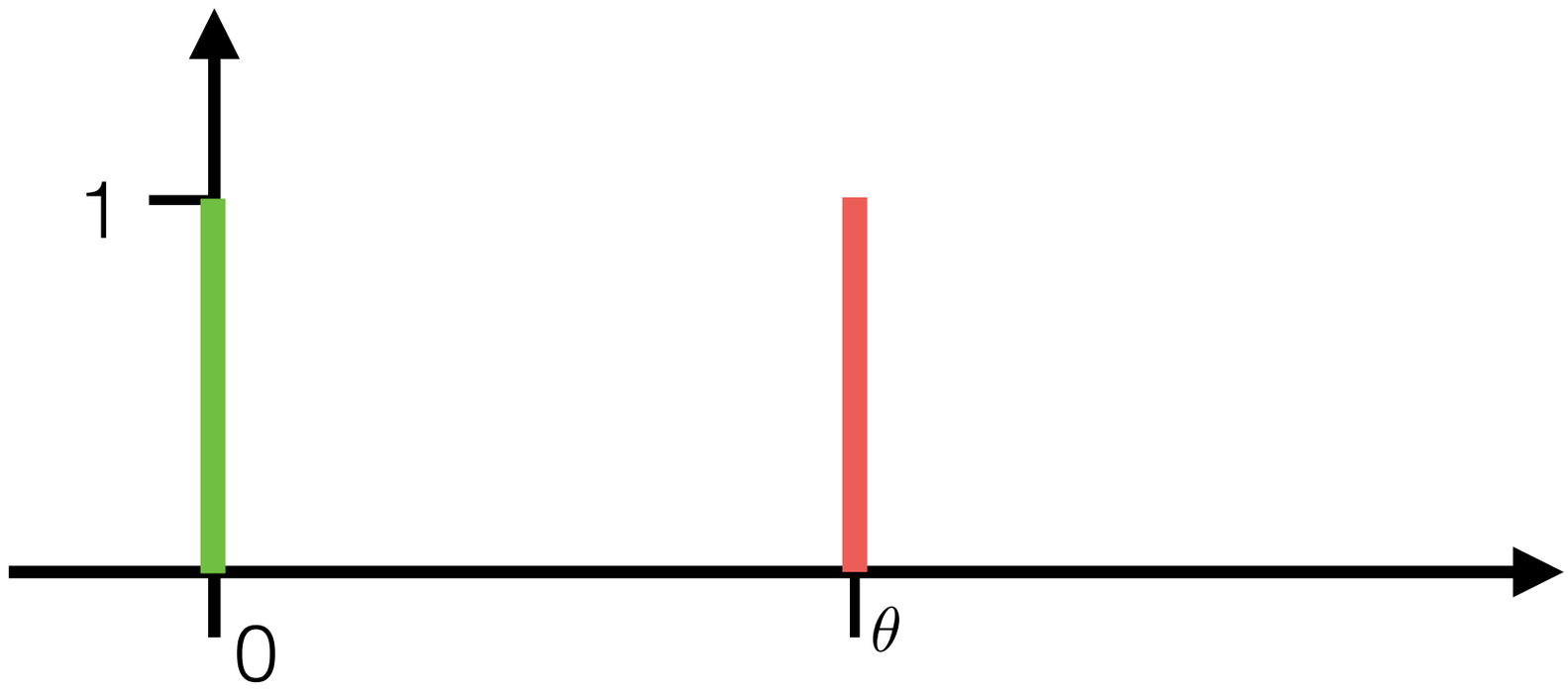
Why is Training GANs so Hard?

Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent



Why is Training GANs so Hard?

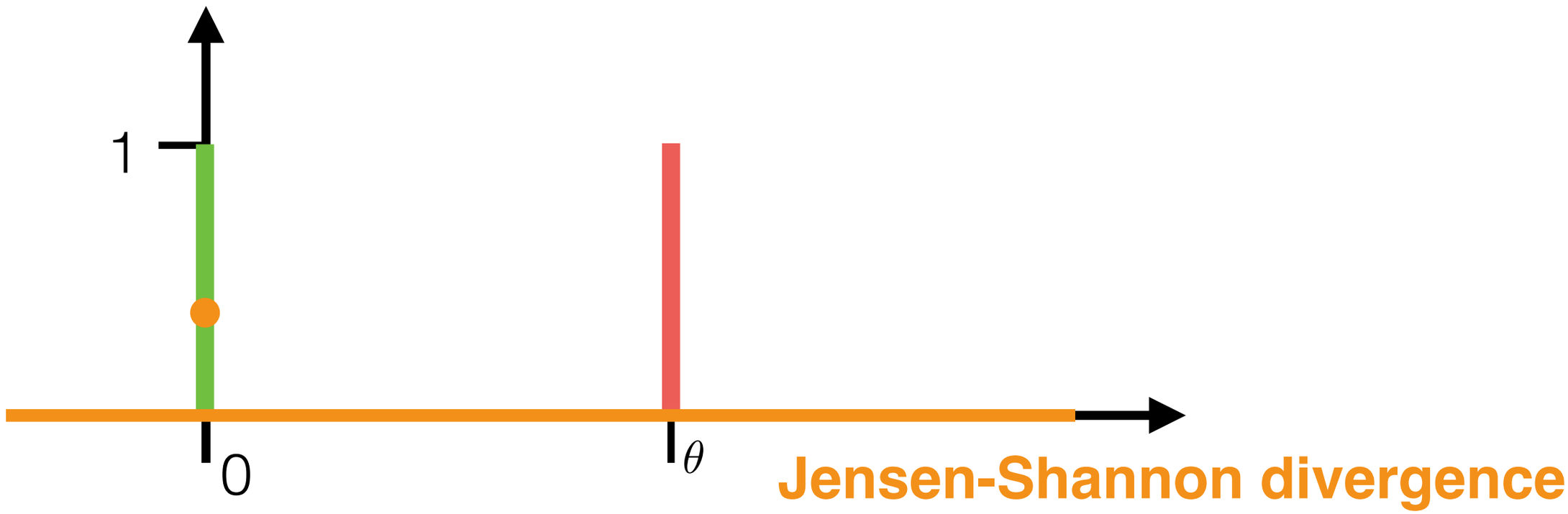
Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent



Simple one-parameter distribution $p_\theta(z) = (\theta, z)$, with $z \sim U[0, 1]$

Why is Training GANs so Hard?

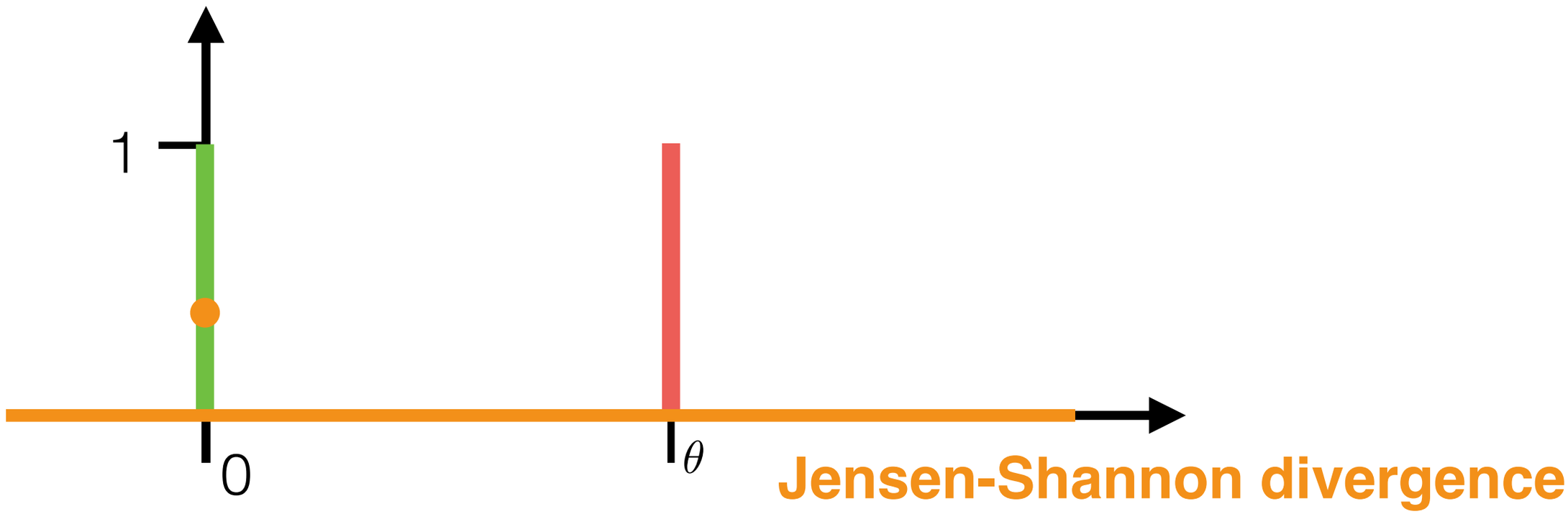
Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent



Simple one-parameter distribution $p_\theta(z) = (\theta, z)$, with $z \sim U[0, 1]$

Why is Training GANs so Hard?

Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent

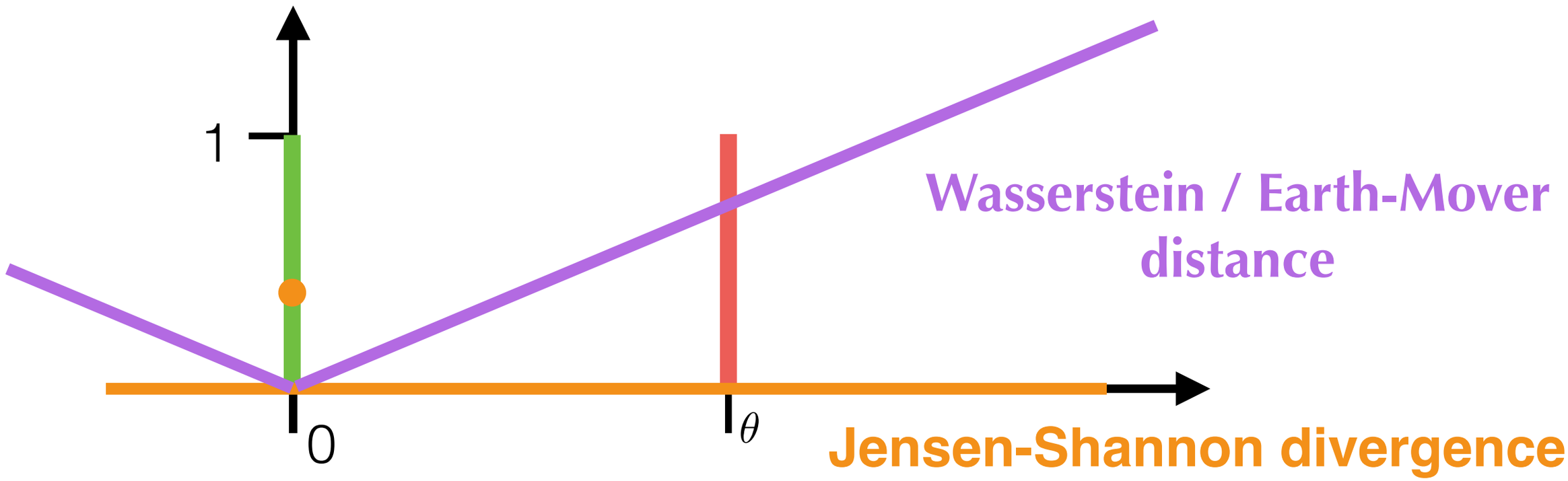


Simple one-parameter distribution $p_\theta(z) = (\theta, z)$, with $z \sim U[0, 1]$

Practical relevance: Operating on disjoints manifolds

Why is Training GANs so Hard?

Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent

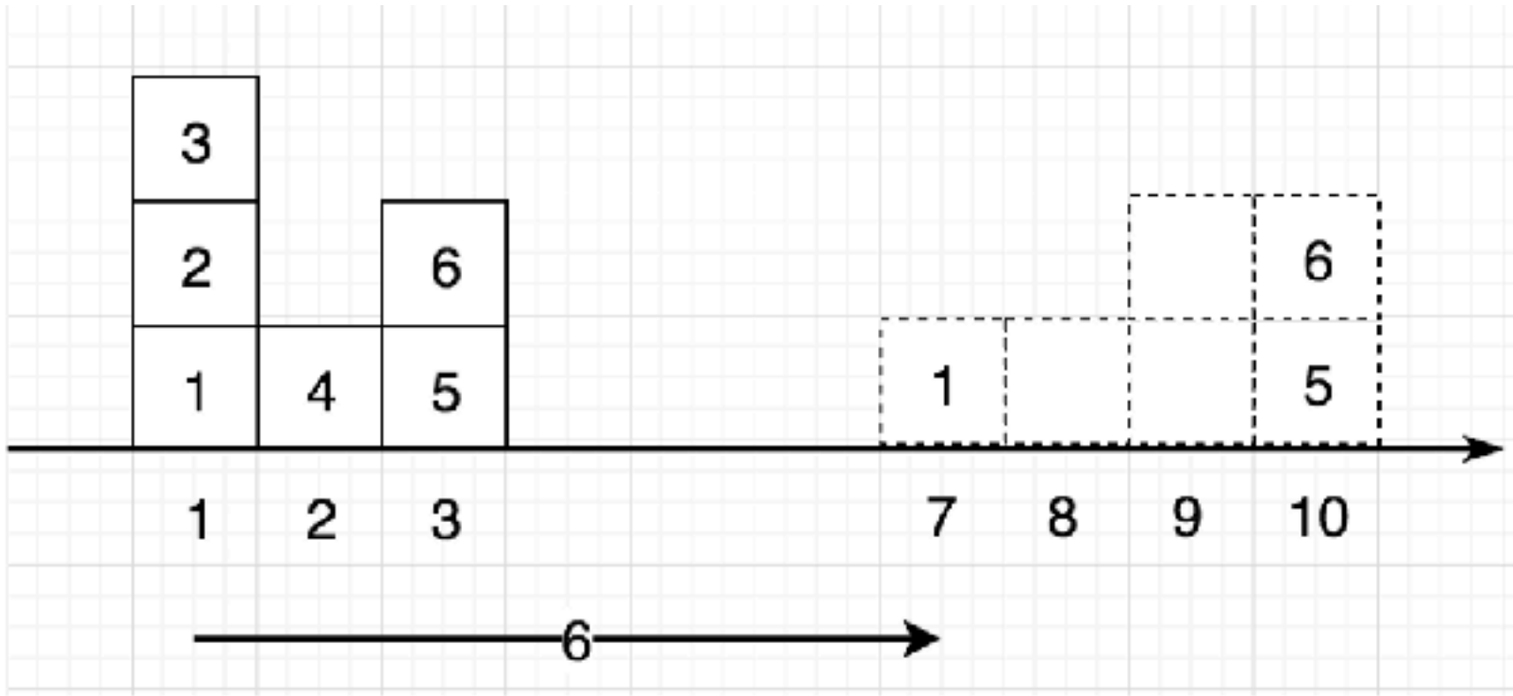


Simple one-parameter distribution $p_\theta(z) = (\theta, z)$, with $z \sim U[0, 1]$

Practical relevance: Operating on disjoints manifolds

Wasserstein GANs (WGANs)

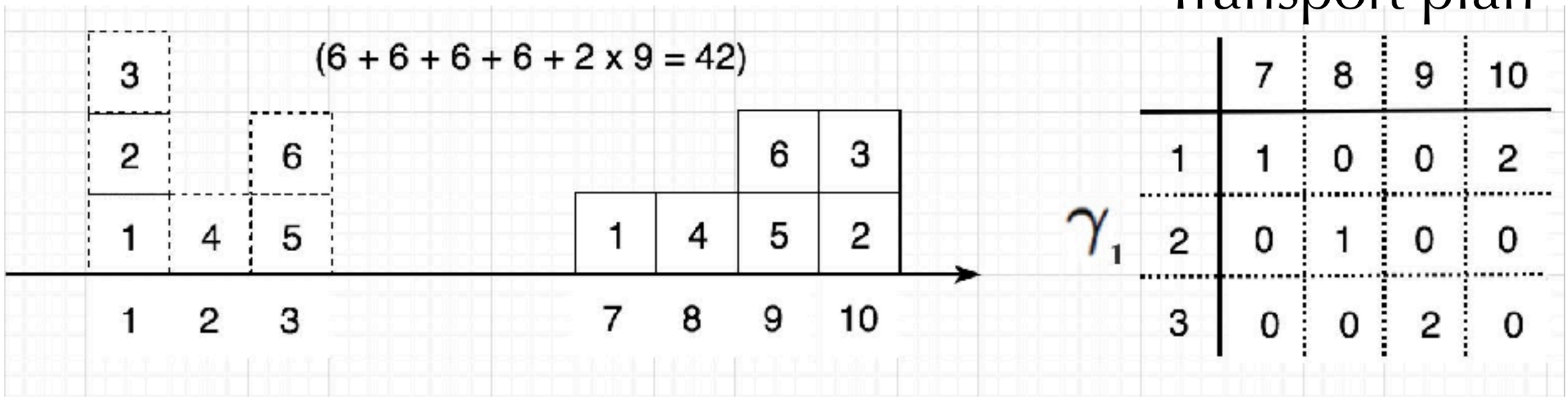
Earth-Mover distance: Minimum amount of work to move weight from $p(x)$ to $q(x)$



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

Wasserstein GANs (WGANs)

Earth-Mover distance: Minimum amount of work to move weight from $p(x)$ to $q(x)$

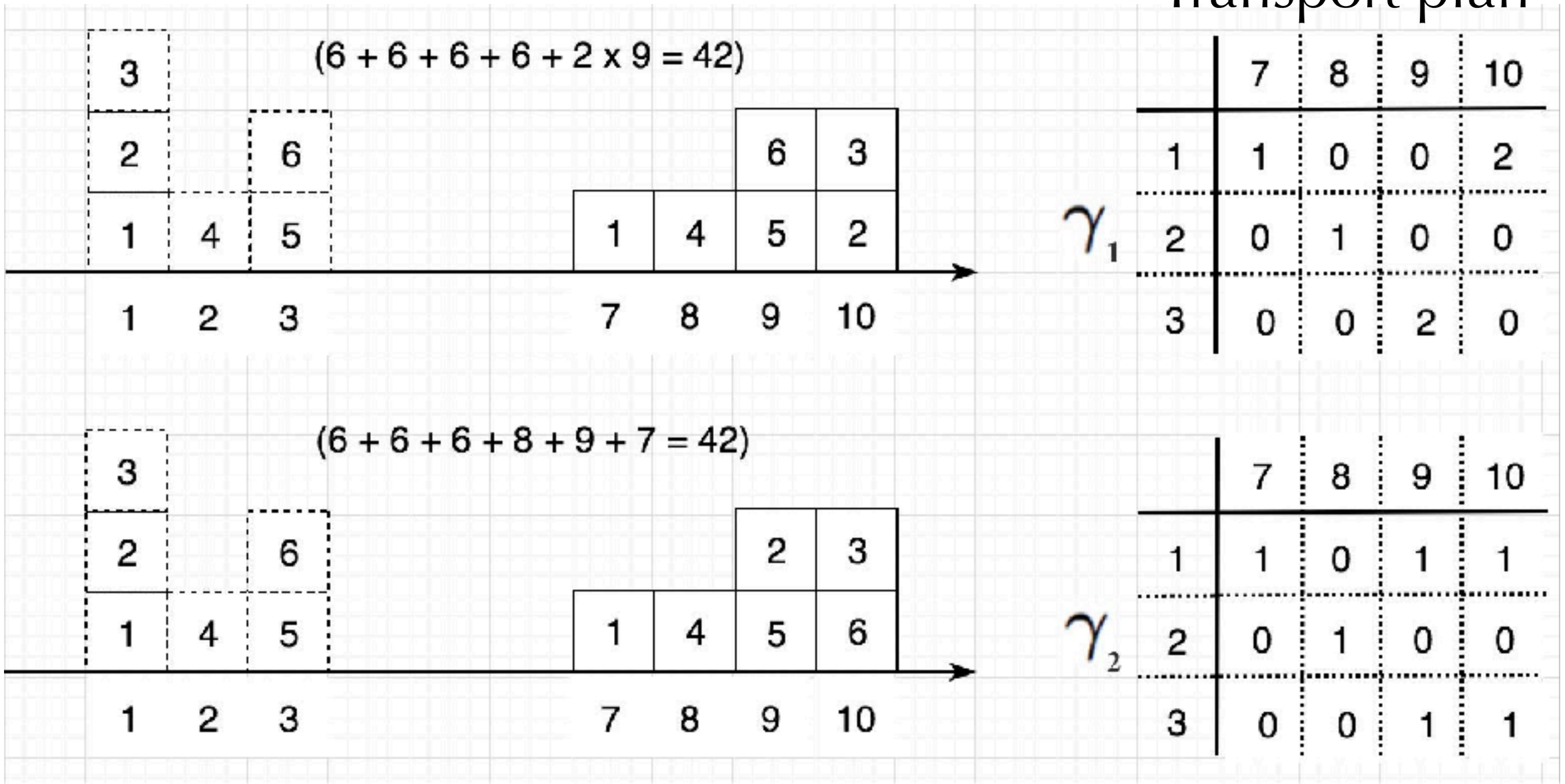


https://medium.com/@jonathan_hui/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490

Wasserstein GANs (WGANs)

Earth-Mover distance: Minimum amount of work to move weight from $\mathbf{p(x)}$ to $\mathbf{q(x)}$

Transport plan



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

Wasserstein GANs (WGANs)

Earth-Mover distance: Minimum amount of work to move weight from $p(x)$ to $q(x)$

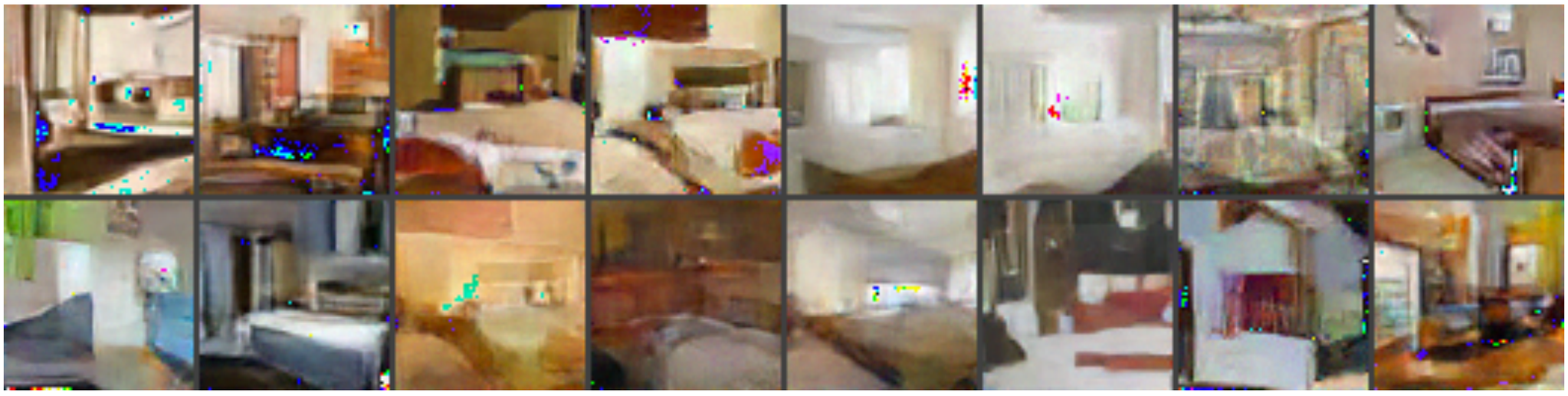
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\, \|x - y\| \,\right],$$

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

WGANs vs. GANs

GAN

WGAN



[Arjovsky et al., Wasserstein GAN, arXiv:1701.07875]

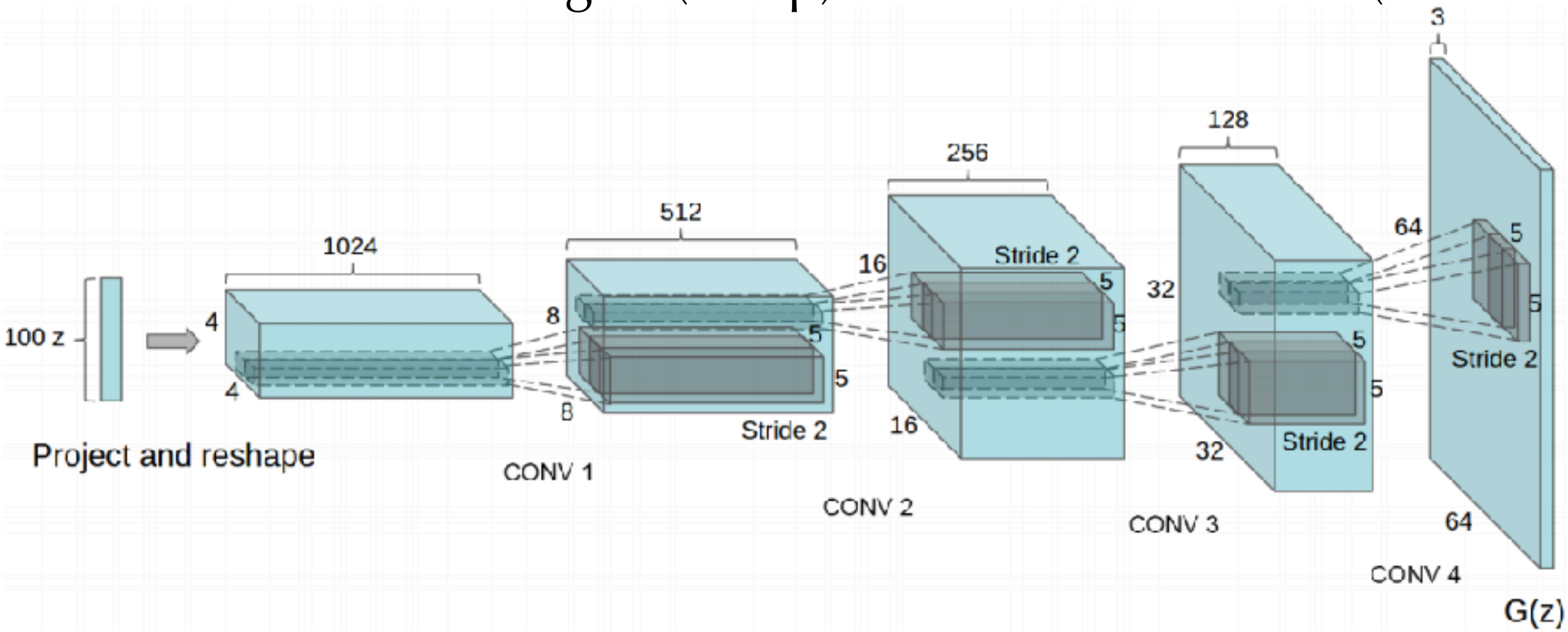
Deep Convolutional GANs (DCGANs)

- GANs are general technique, not restricted to images
- This lecture focus on images: (Deep) Convolutional GANs (DCGANs)

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

Deep Convolutional GANs (DCGANs)

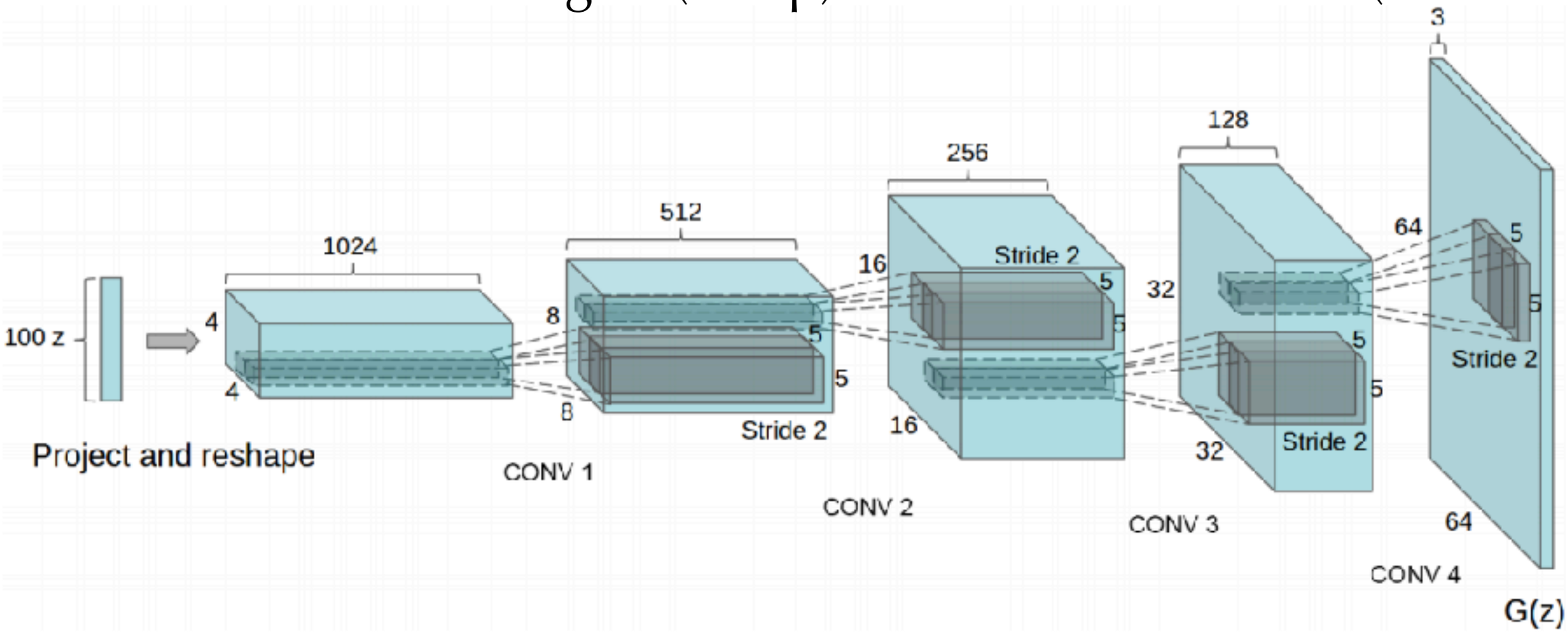
- GANs are general technique, not restricted to images
- This lecture focus on images: (Deep) Convolutional GANs (DCGANs)



[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

Deep Convolutional GANs (DCGANs)

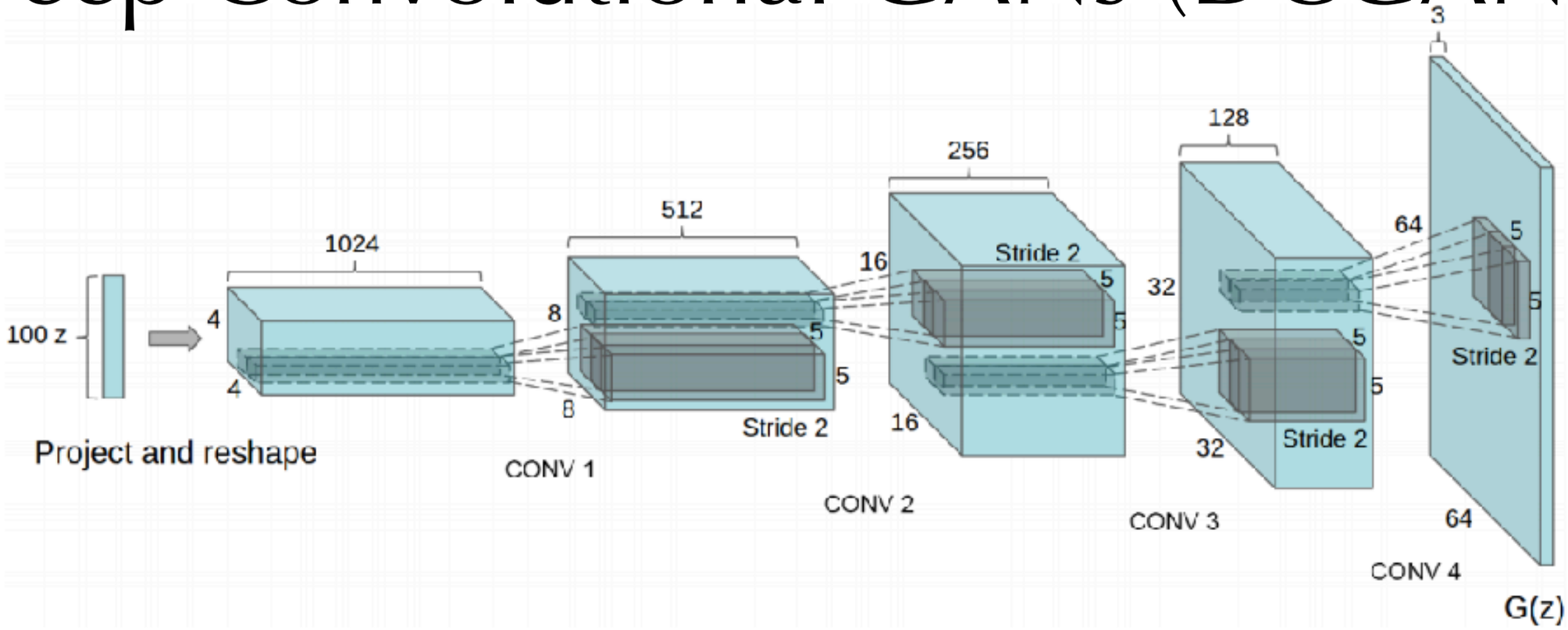
- GANs are general technique, not restricted to images
- This lecture focus on images: (Deep) Convolutional GANs (DCGANs)



- List of recommendations for stable DCGANs

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

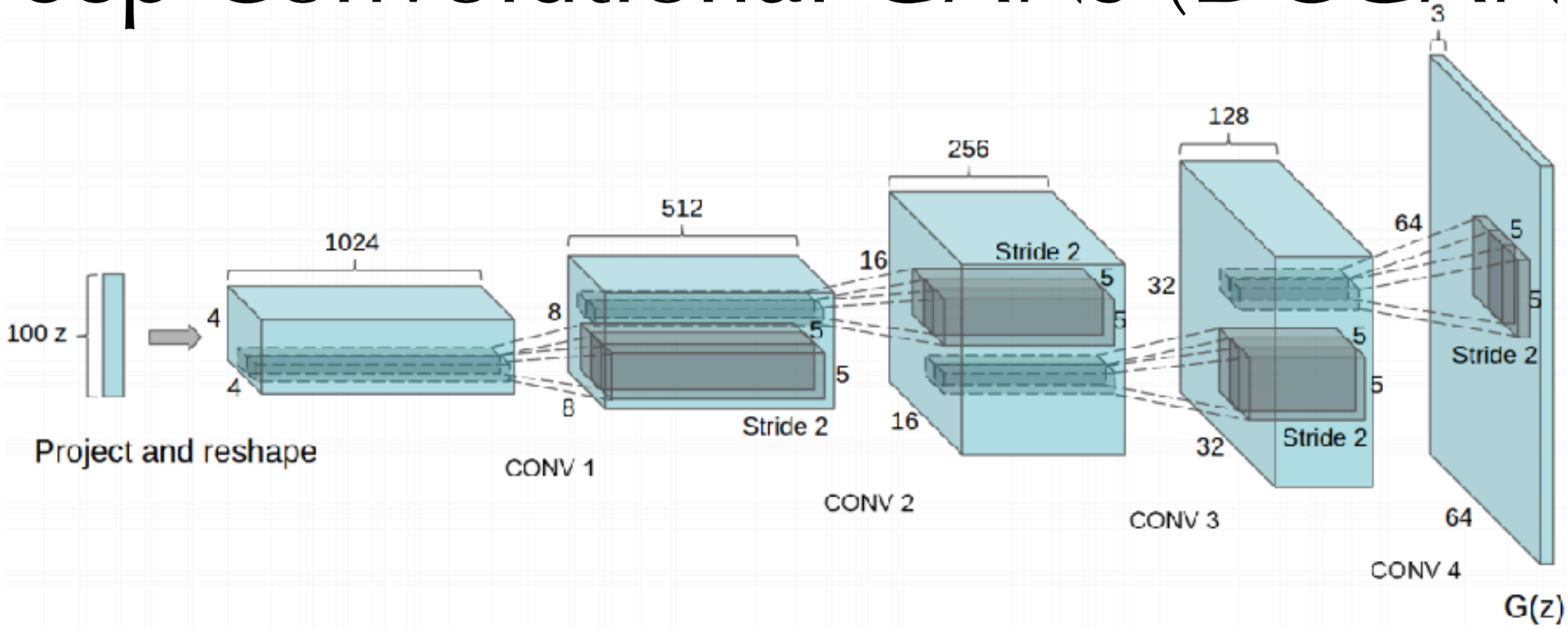
Deep Convolutional GANs (DCGANs)



- Discriminator: Use strided convolutions instead of max-pooling
- Generator: Use transposed convolutions
- Intuition: Learn up- / down-sampling

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

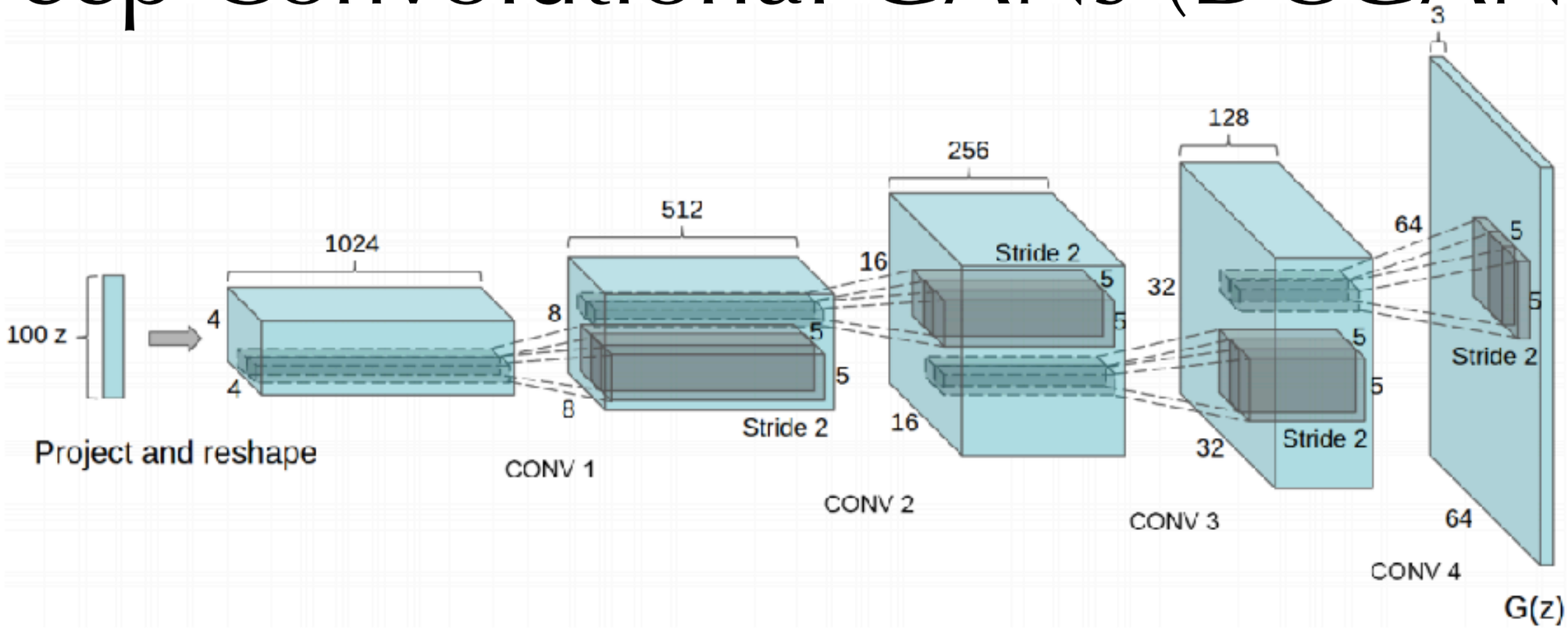
Deep Convolutional GANs (DCGANs)



- No fully connected hidden layers

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

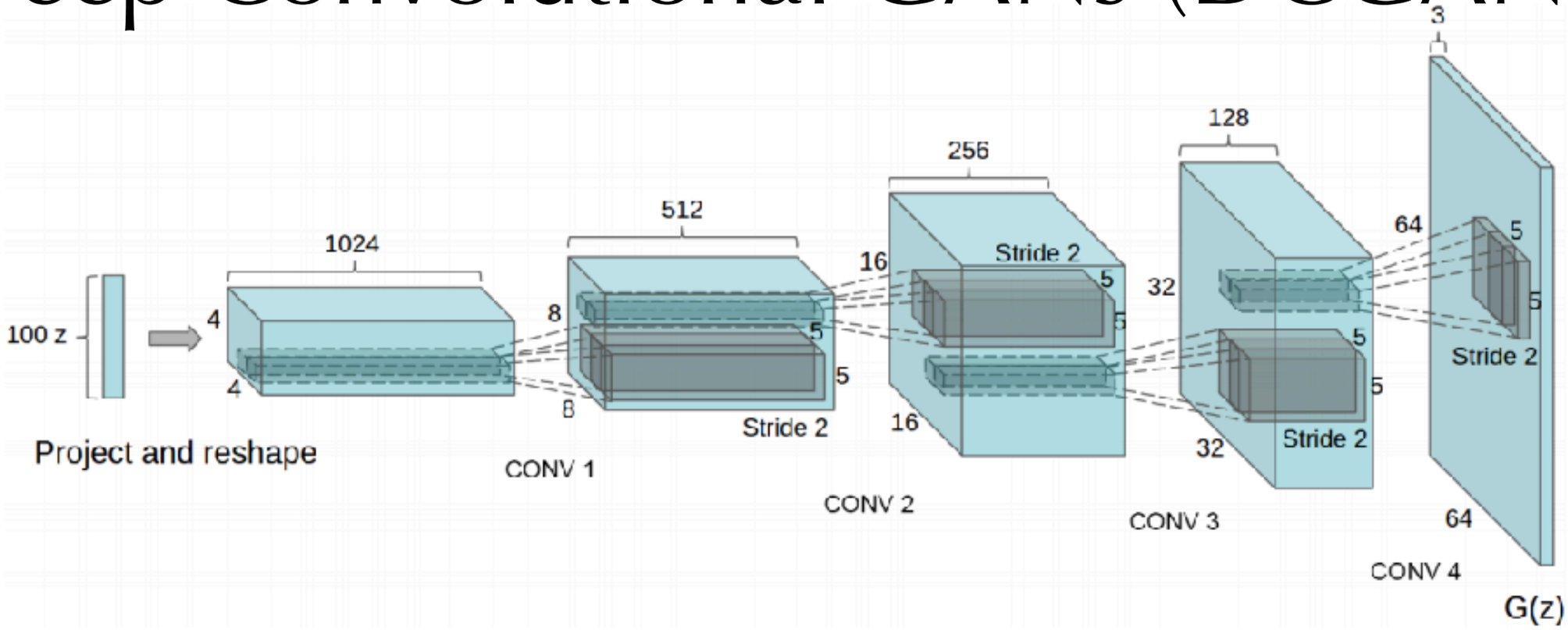
Deep Convolutional GANs (DCGANs)



- No fully connected hidden layers
- Observation: Improves convergence speed = training speed

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

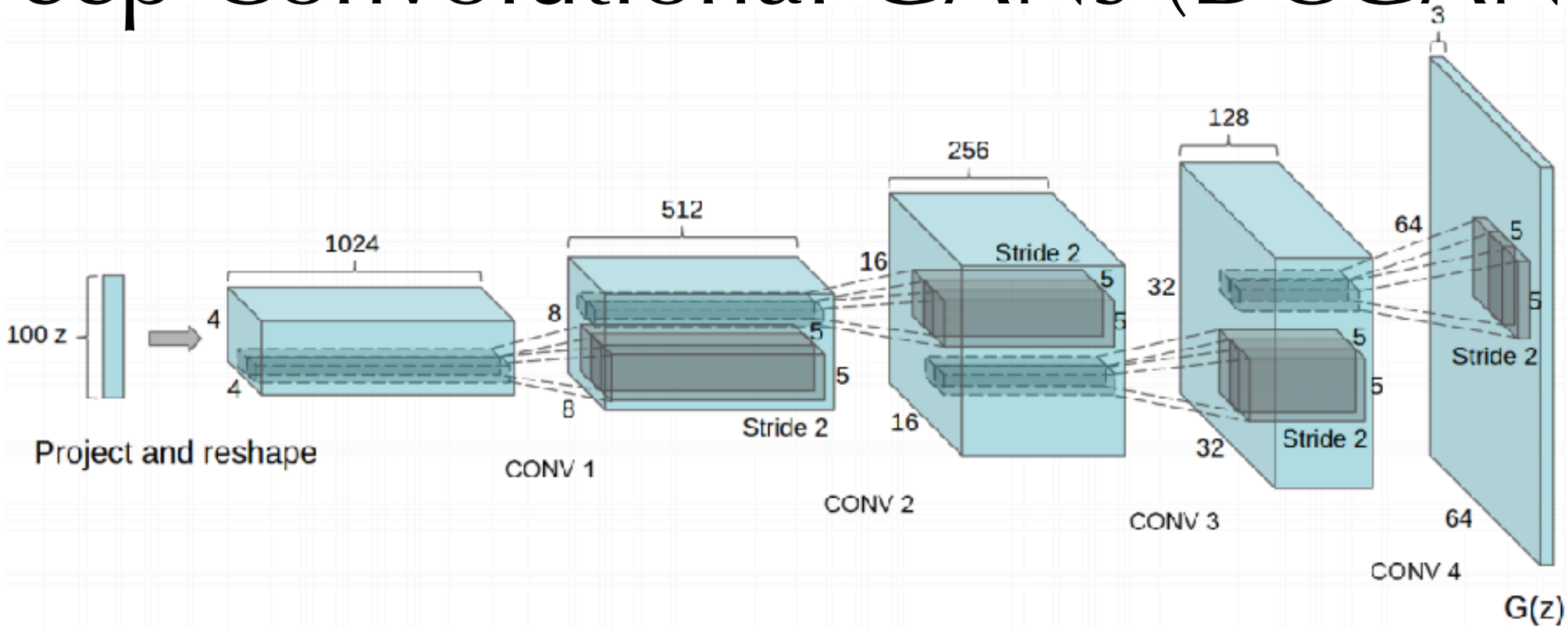
Deep Convolutional GANs (DCGANs)



- Use **batch normalization** (batch norm) in both generator and discriminator (but not in generator output layer and discriminator input layer)

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

Deep Convolutional GANs (DCGANs)



- Use **batch normalization** (batch norm) in both generator and discriminator (but not in generator output layer and discriminator input layer)
- Observation: Helps generator to start training (prevents generator from collapsing all samples to single point)

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

Batch Normalization

- Motivation:
  - Distribution of input of network layers changes over time
  - Distribution potentially changes with each mini batch
  - Later layers are affected by distributions of previous layers
  - Makes training slower and harder

[Ioffe & Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICMl 2015]

Batch Normalization

- Motivation:
  - Distribution of input of network layers changes over time
  - Distribution potentially changes with each mini batch
  - Later layers are affected by distributions of previous layers
  - Makes training slower and harder

- Batch normalization:
  - Normalize the distribution to stabilize training
  - Normalization per mini-batch for efficiency

[Ioffe & Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICMl 2015]

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

[Ioffe & Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICMl 2015]

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

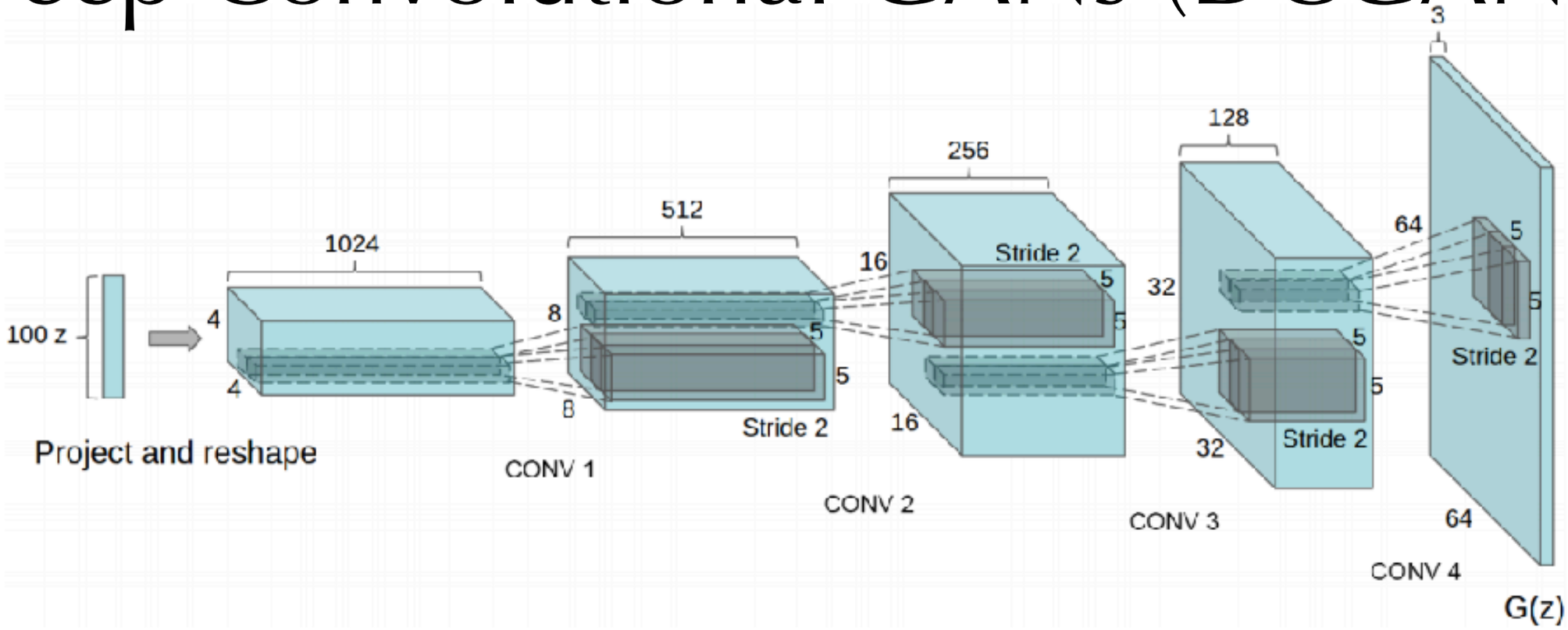
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Observation: Stabilizes and accelerates the training process

[Ioffe & Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICMl 2015]

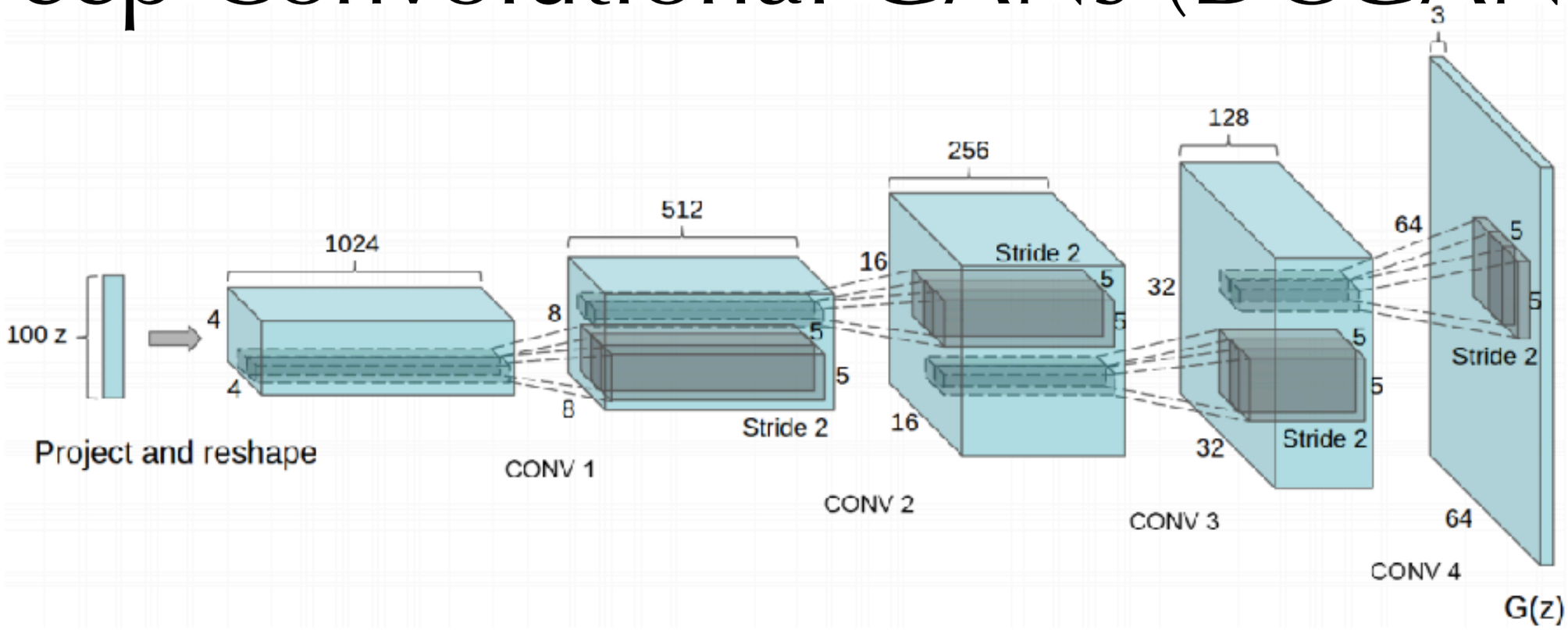
Deep Convolutional GANs (DCGANs)



- Generator: Use ReLU in all layers, but tanh in output layer

[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]
[Maas et al., Rectifier Nonlinearities Improve Neural Network Acoustic Models, ICML 2013]

Deep Convolutional GANs (DCGANs)

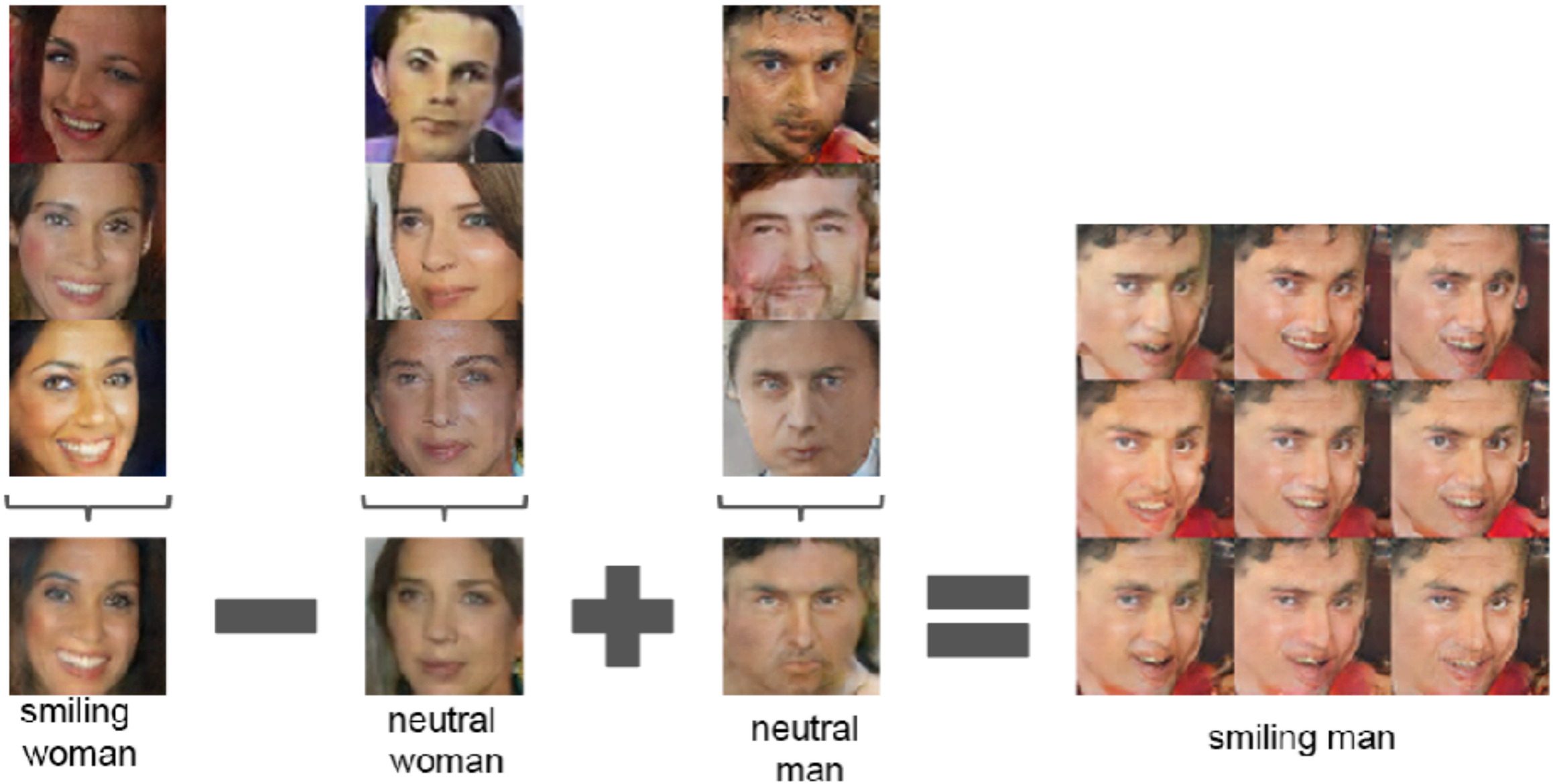


- Generator: Use ReLU in all layers, but tanh in output layer
- Discriminator: Use LeakyReLU

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

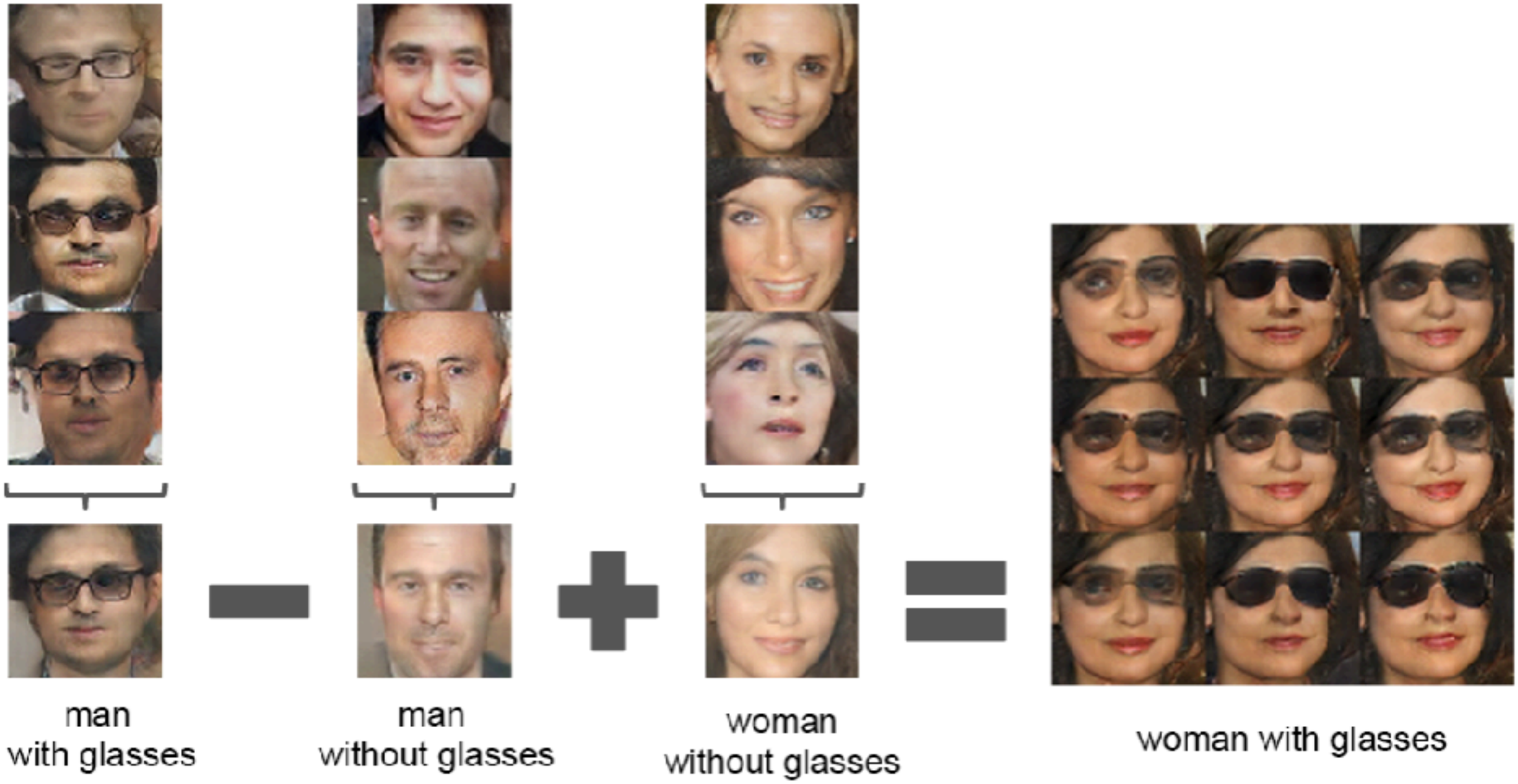
[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]
[Maas et al., Rectifier Nonlinearities Improve Neural Network Acoustic Models, ICML 2013]

Deep Convolutional GANs (DCGANs)



[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

Deep Convolutional GANs (DCGANs)



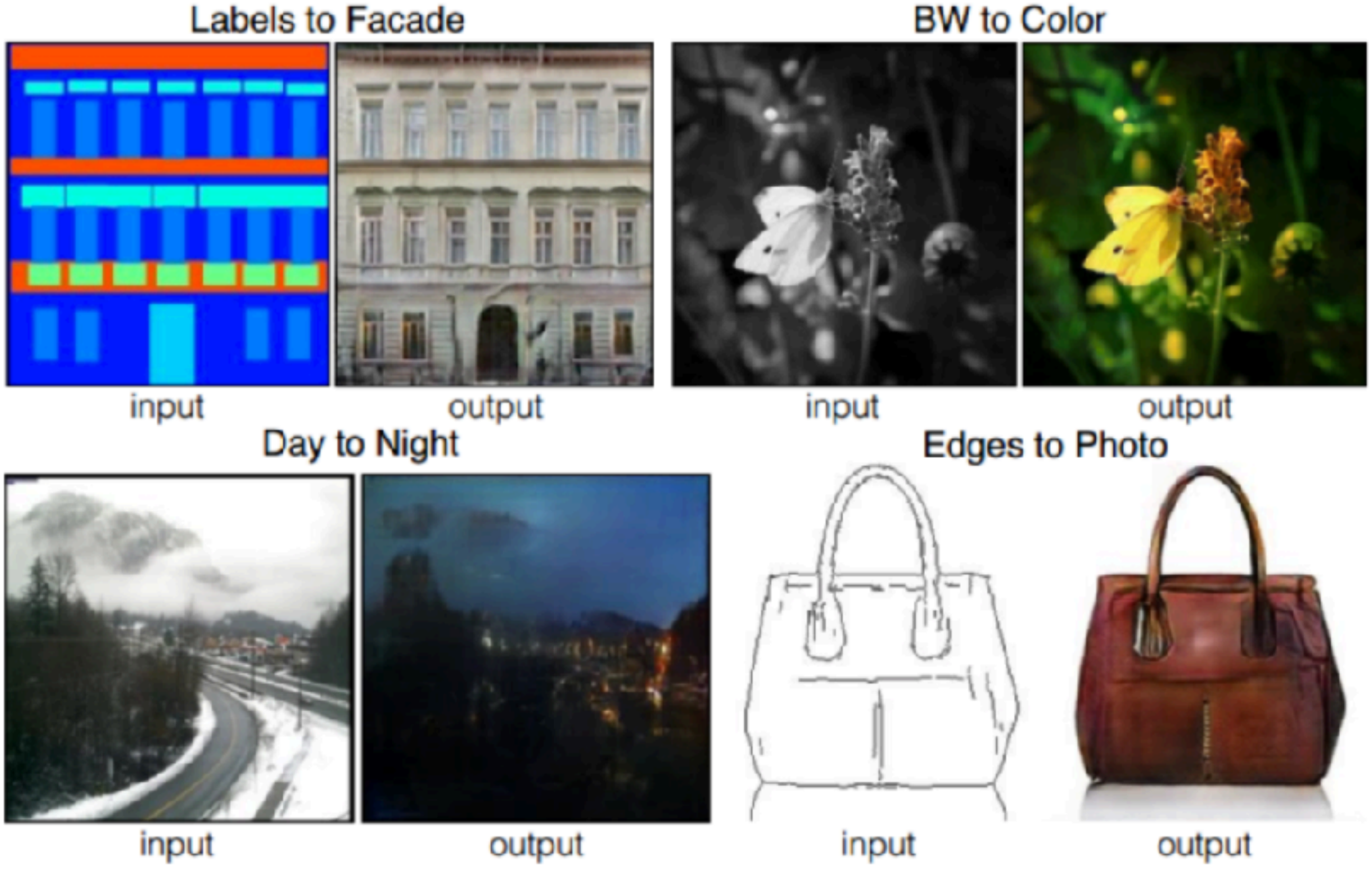
[Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016]

State-of-the-art Results

Today

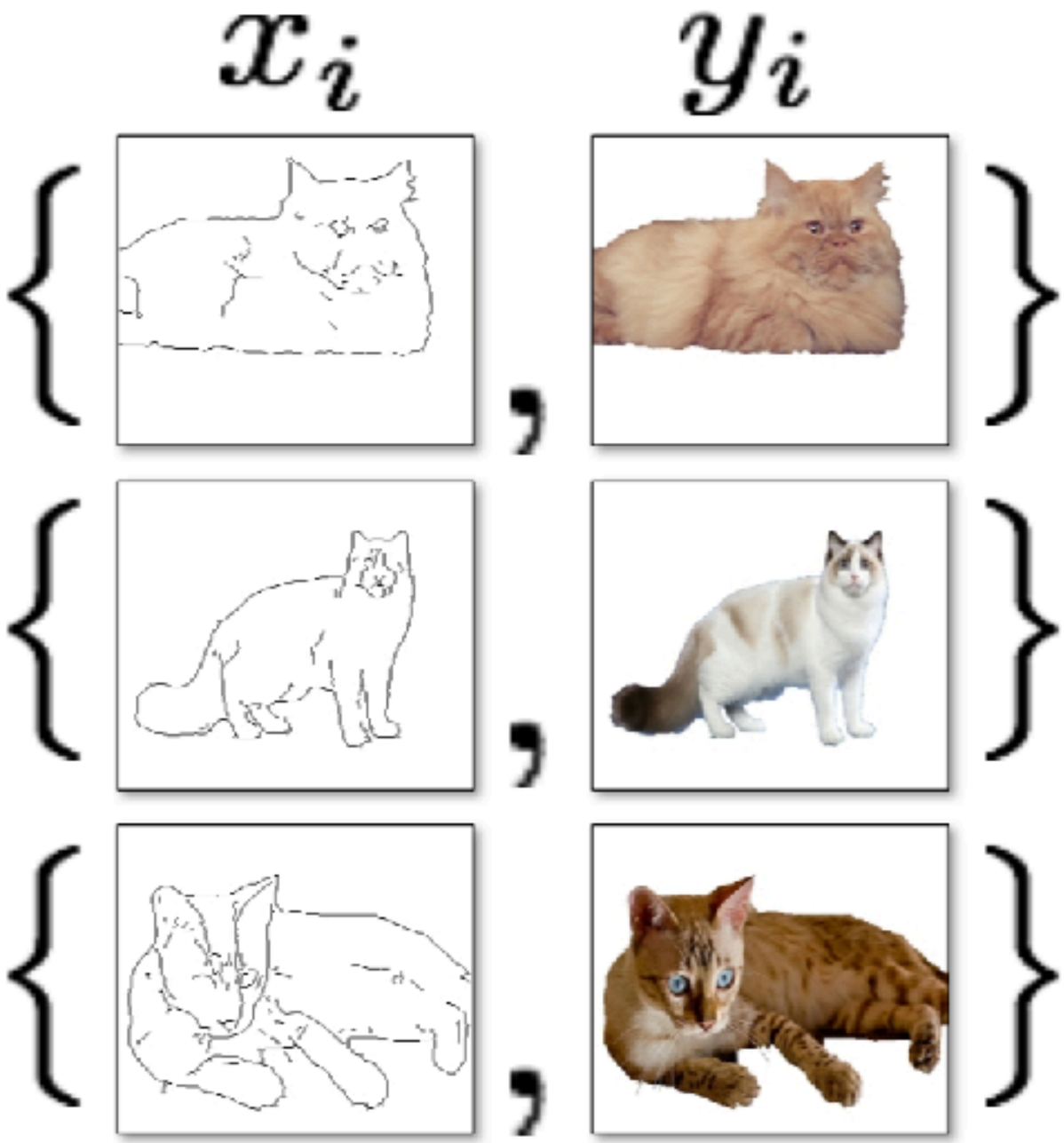
- Generative Adversarial Networks (GANs)
- **(Unsupervised) Image Translation**

Image-To-Image Translation



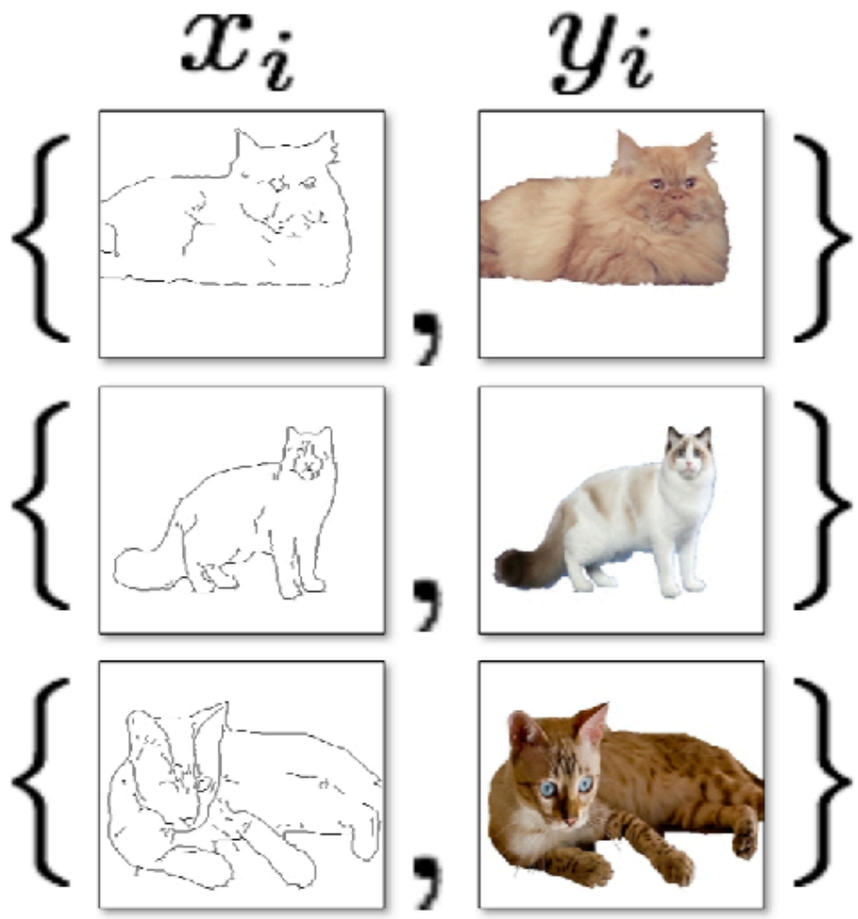
slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



slide credit: Jun-Yan Zhu, Taesung Park

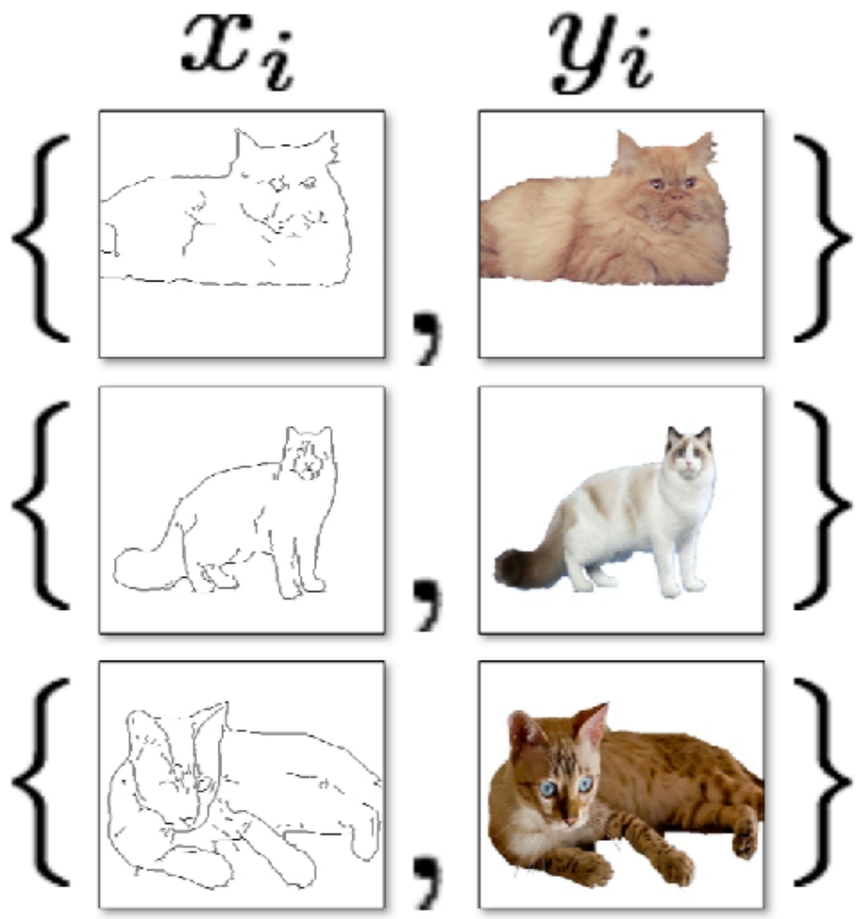
Supervised Image-To-Image Translation



$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation

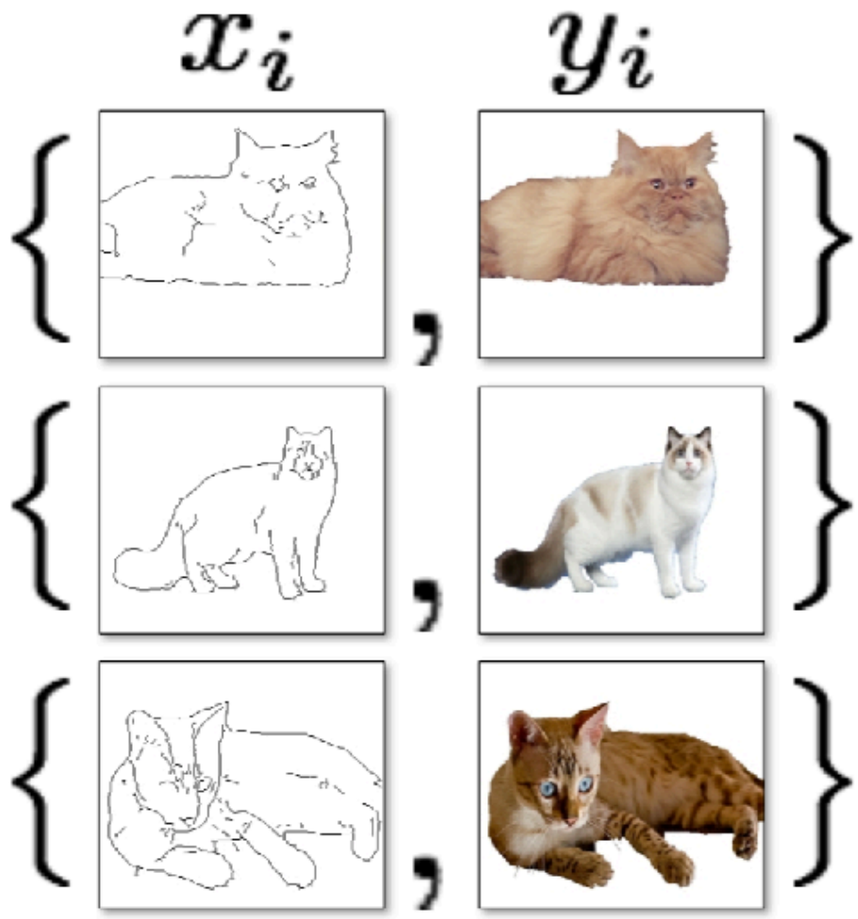


real cats

$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation

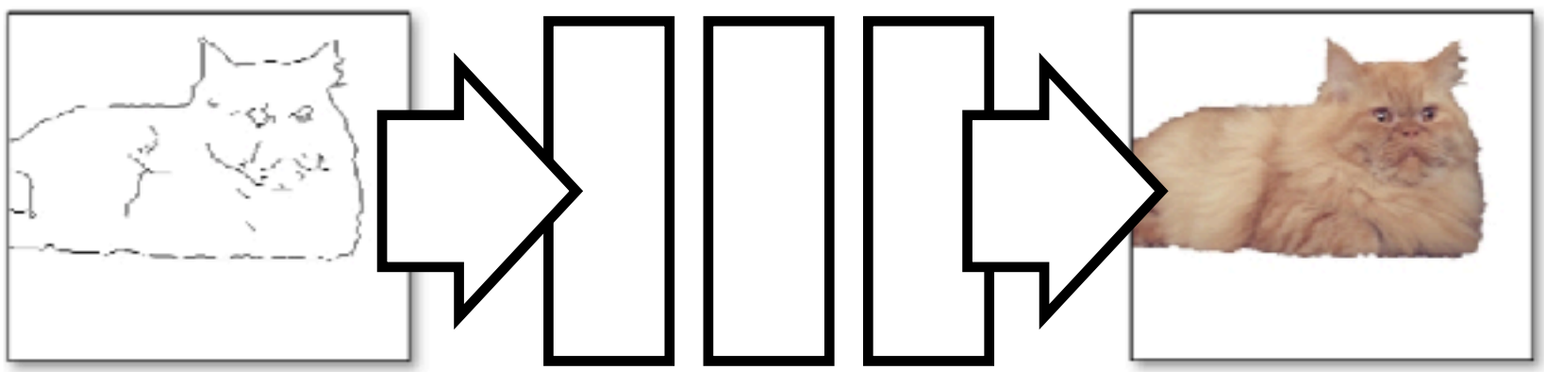


real cats cat sketches

$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation

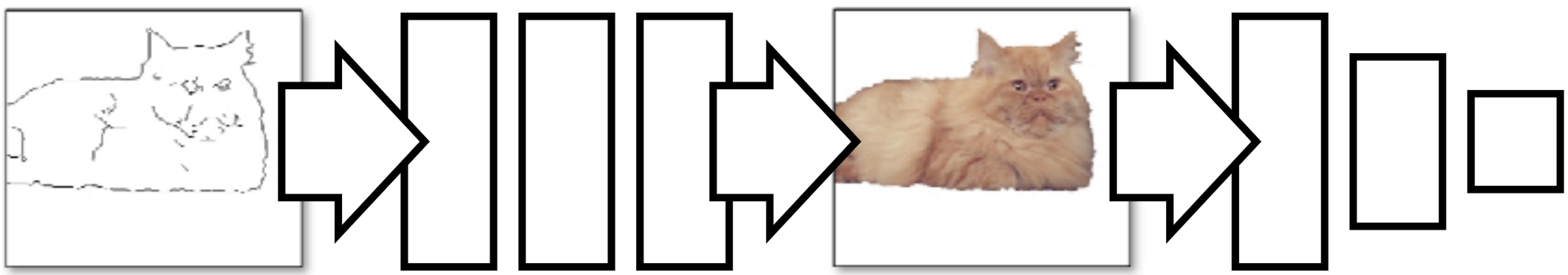


generator G

$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



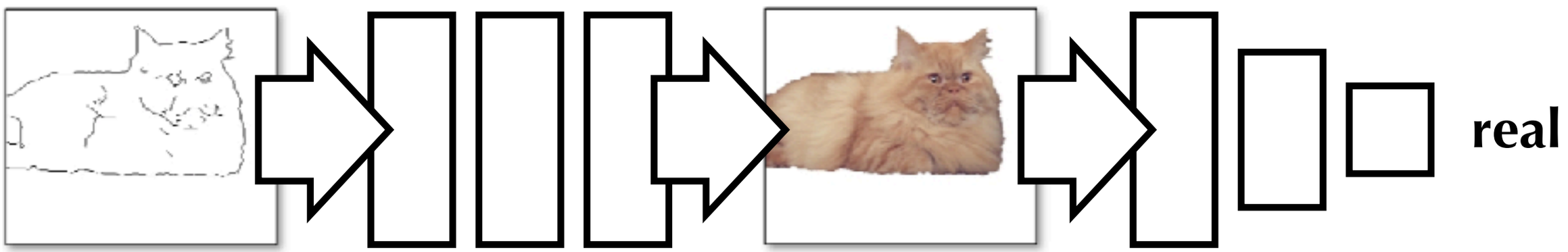
generator G

discriminator D

$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



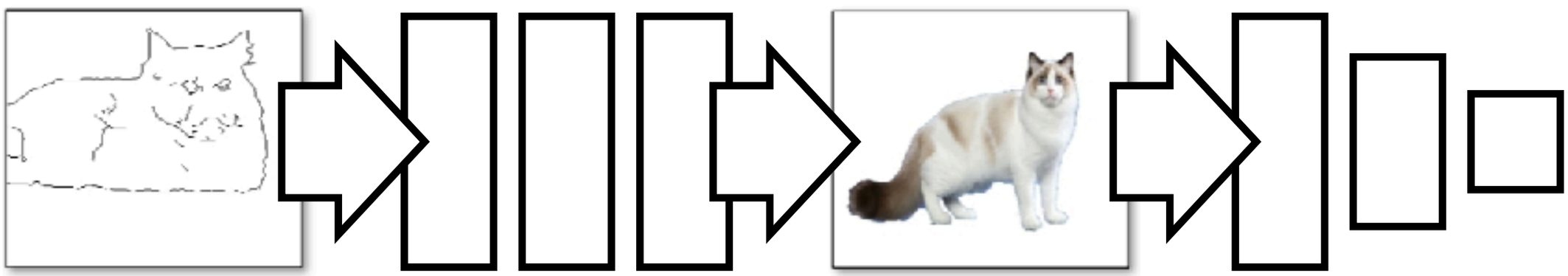
generator G

discriminator D

$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



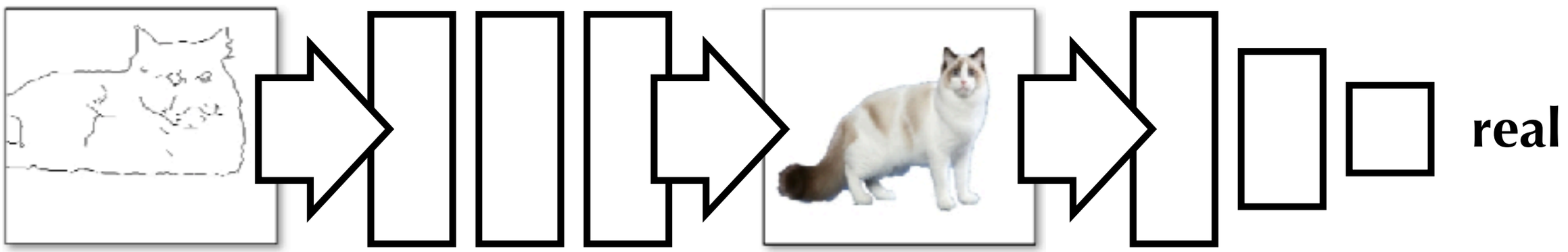
generator G

discriminator D

$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



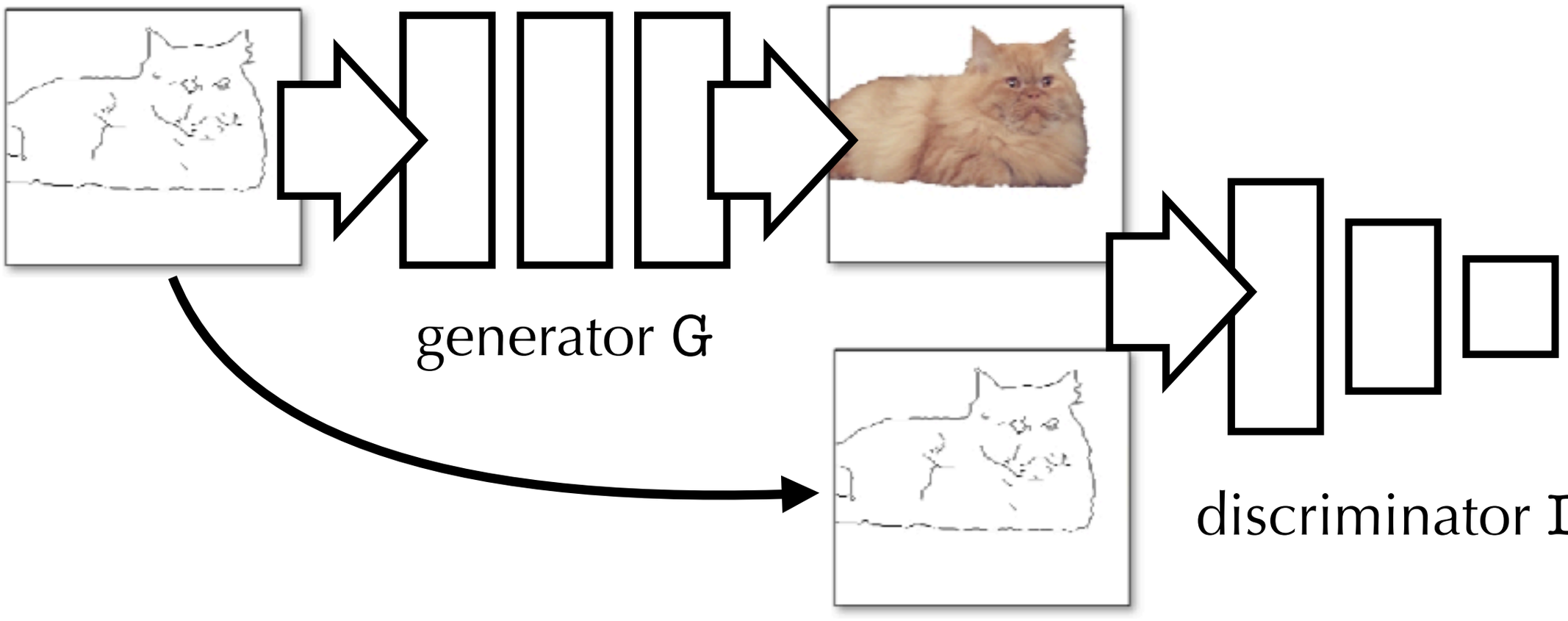
generator G

discriminator D

$$\min_G \max_D E_{\mathbf{y}}[\log(D(\mathbf{y}))] + E_{\mathbf{x}}[\log(1 - D(G(\mathbf{x})))]$$

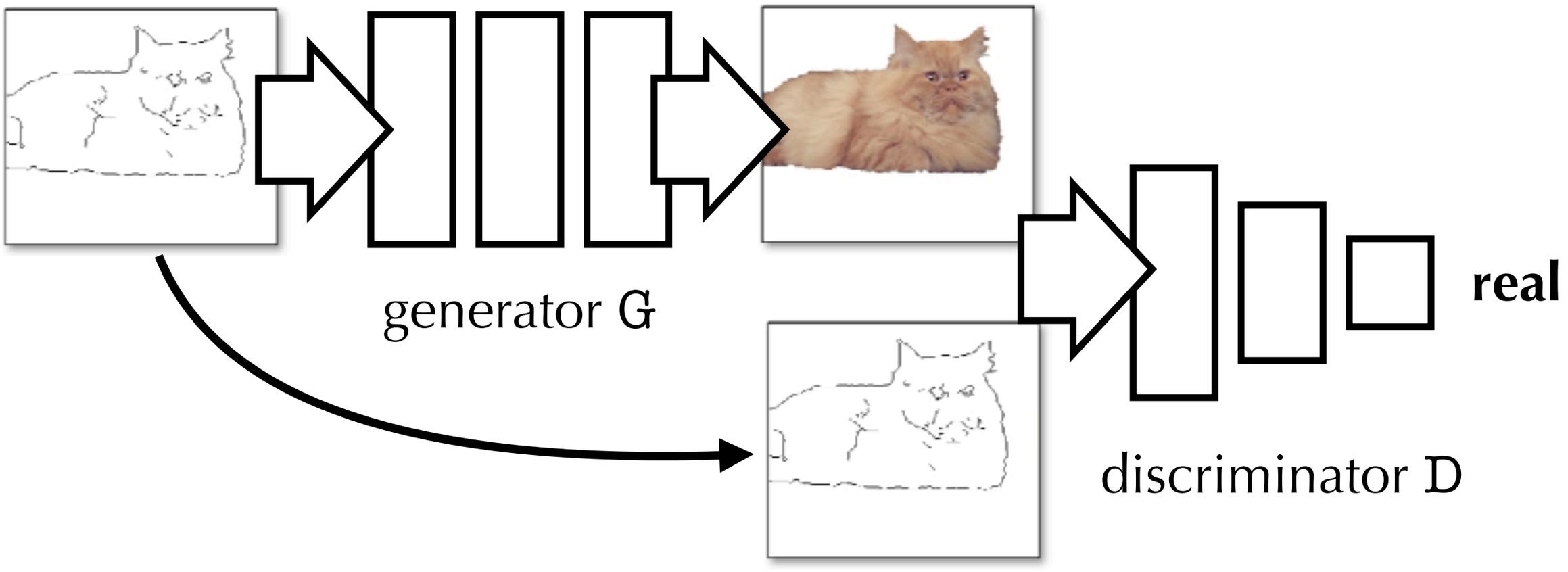
slide credit: Jun-Yan Zhu, Taesung Park

Conditional GANs



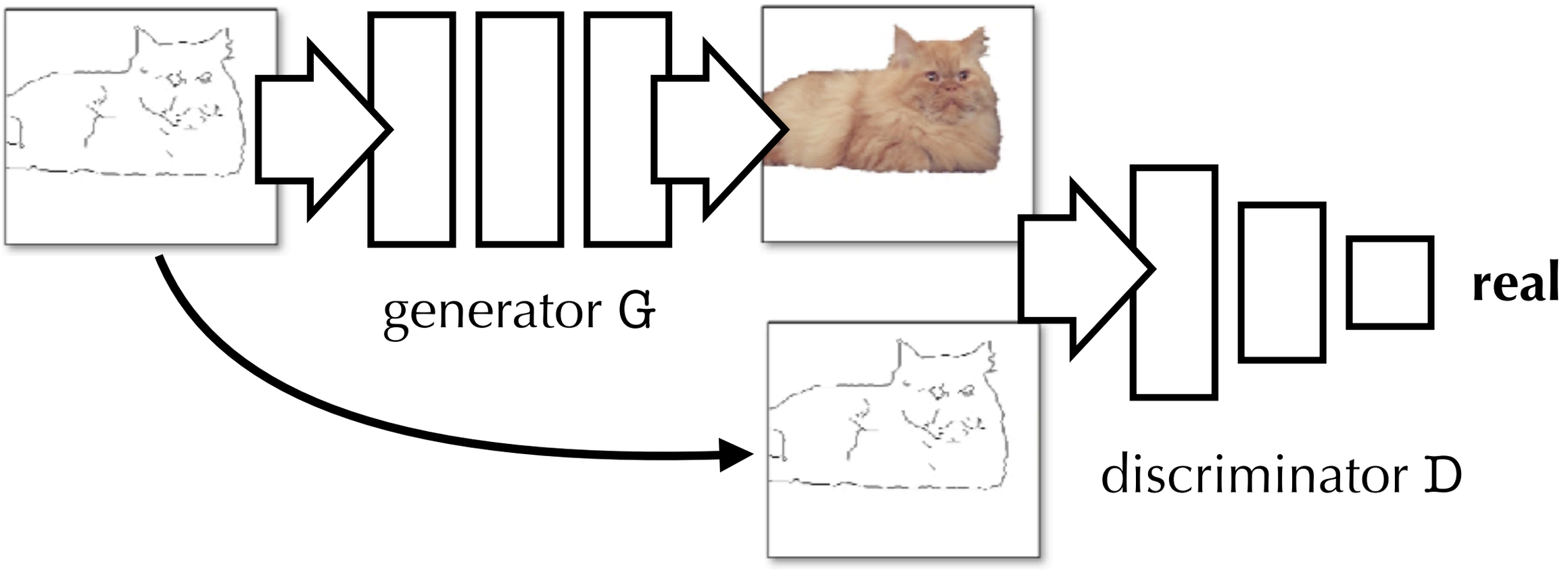
[Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017]

Conditional GANs



[Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017]

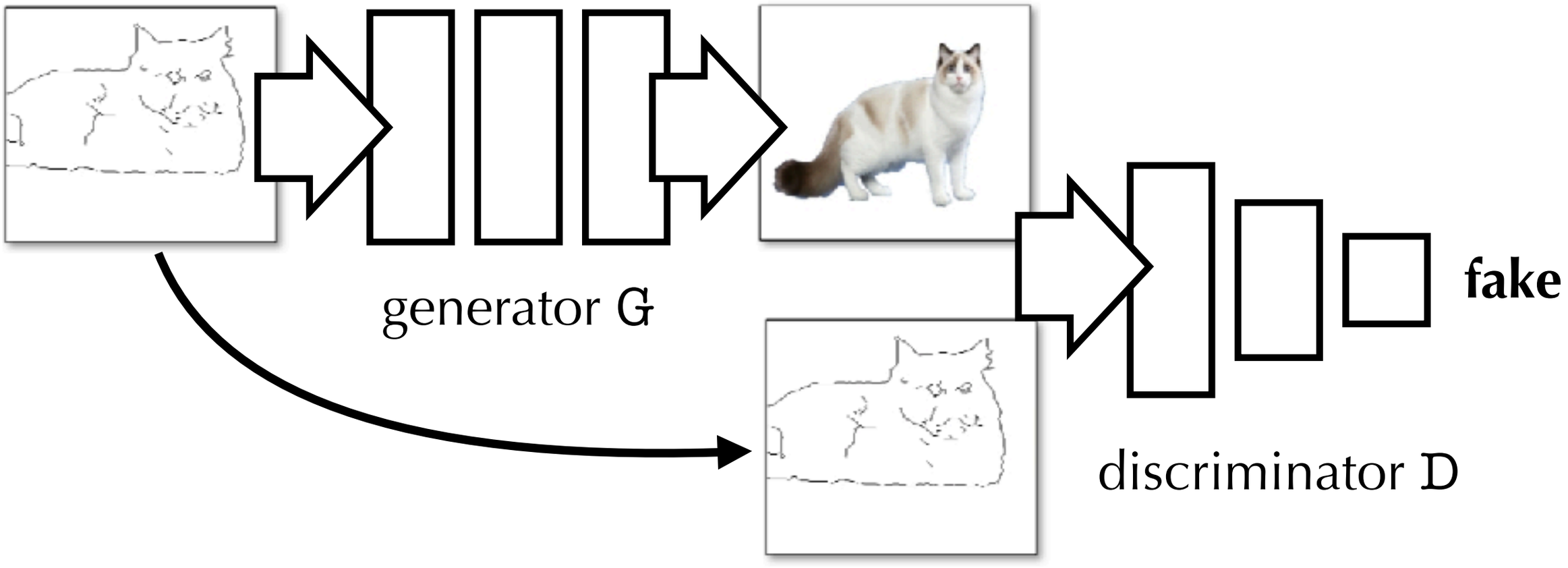
Conditional GANs



$$\min_G \max_D E_{\mathbf{x},\mathbf{y}}[\log(D(\mathbf{x},\mathbf{y}))] + E_{\mathbf{x},\mathbf{y}}[\log(1 - D(\mathbf{x}, G(\mathbf{x})))]$$

[Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017]

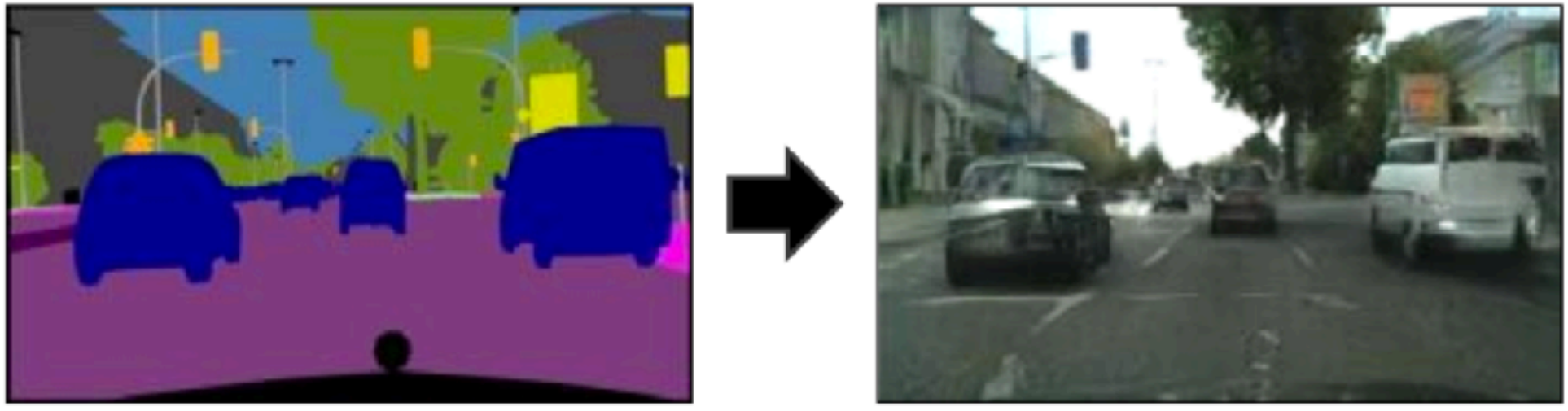
Conditional GANs



$$\min_G \max_D E_{\mathbf{x},\mathbf{y}}[\log(D(\mathbf{x},\mathbf{y}))] + E_{\mathbf{x},\mathbf{y}}[\log(1 - D(\mathbf{x}, G(\mathbf{x})))]$$

[Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017]

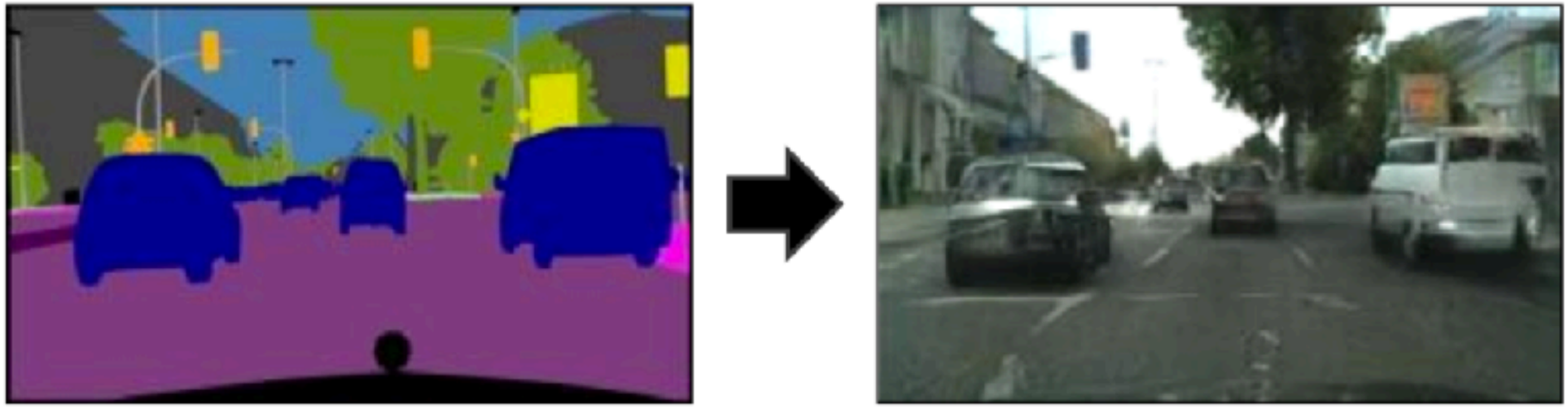
Supervised Image-To-Image Translation



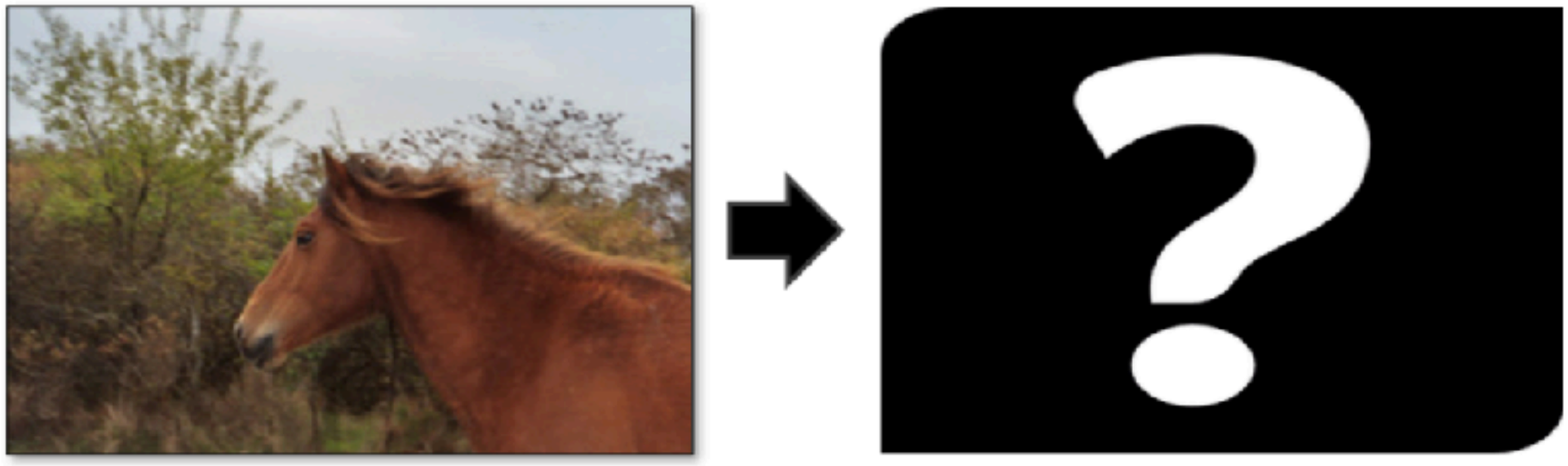
Label ↔ photo: per-pixel labeling

slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



Label ↔ photo: per-pixel labeling



Labelled data often hard or impossible to obtain

slide credit: Jun-Yan Zhu, Taesung Park

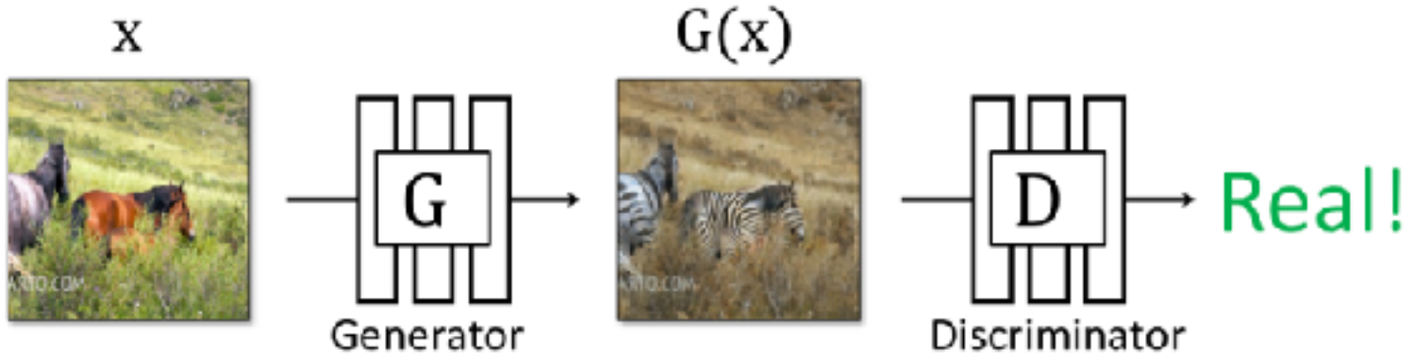
Unsupervised Image-To-Image Translation

independent sets,
no pairing!

slide credit: Jun-Yan Zhu, Taesung Park

Unsupervised Image-To-Image Translation

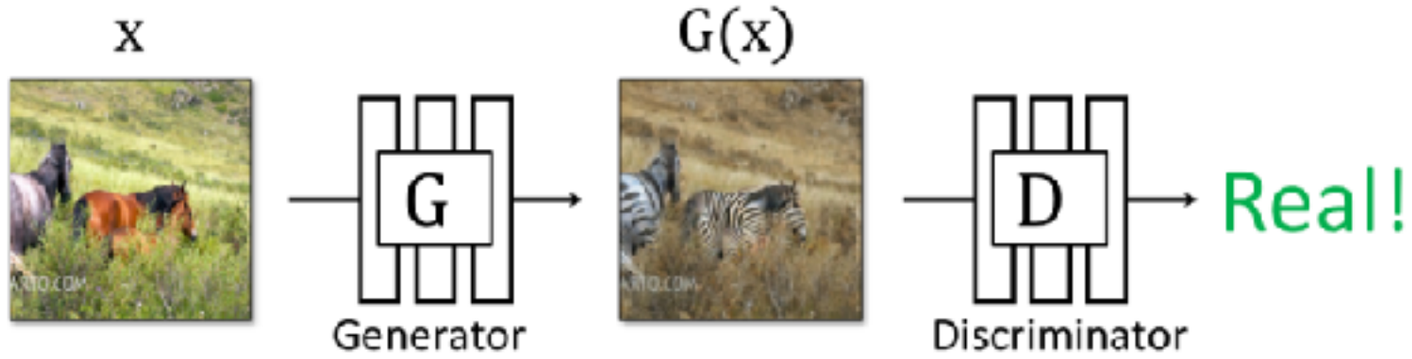
independent sets,
no pairing!

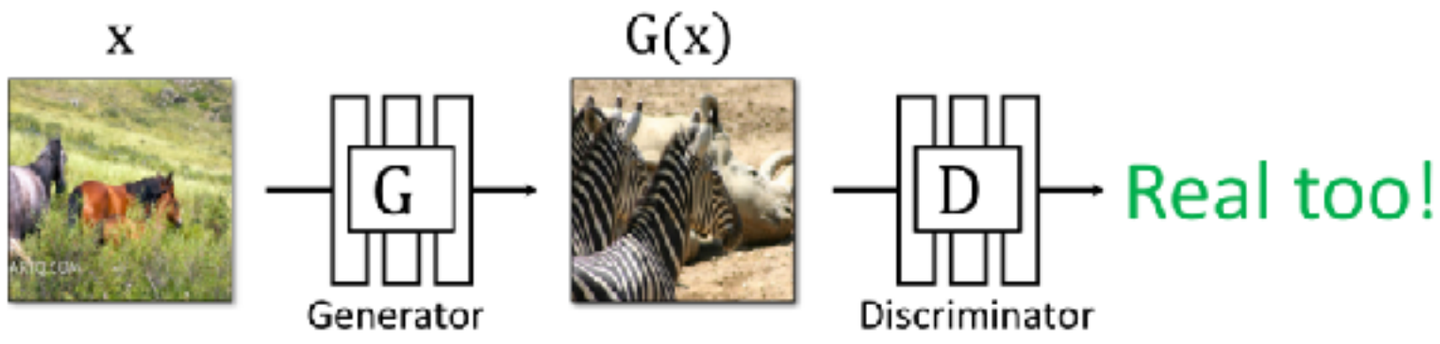


slide credit: Jun-Yan Zhu, Taesung Park

Unsupervised Image-To-Image Translation

independent sets,
no pairing!

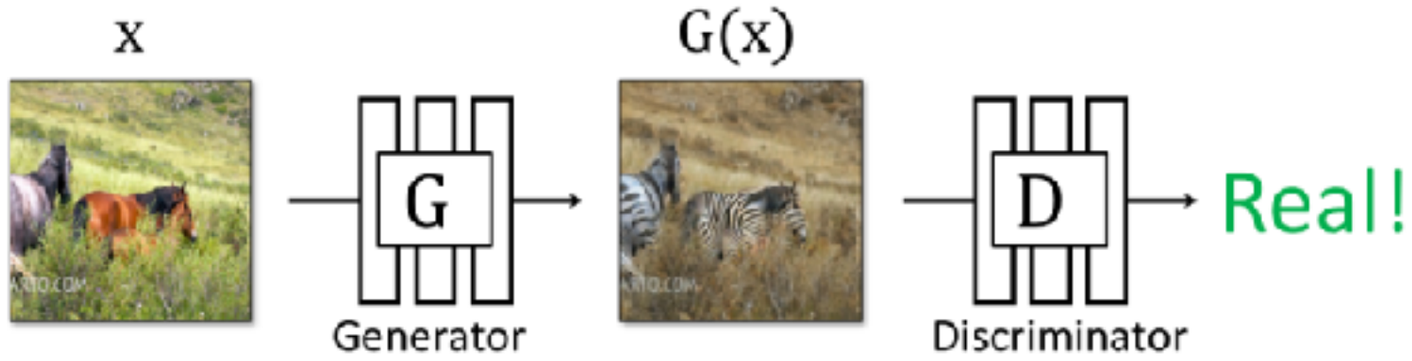


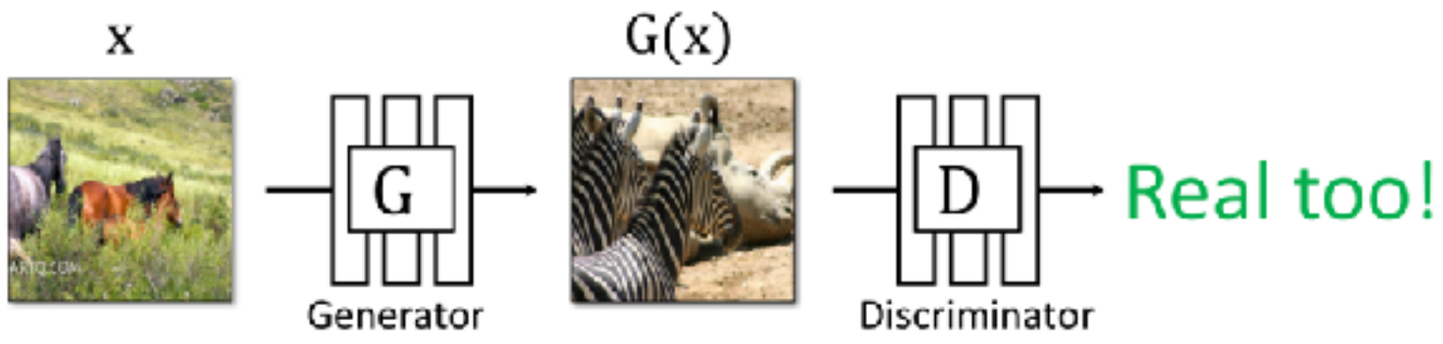


slide credit: Jun-Yan Zhu, Taesung Park

Unsupervised Image-To-Image Translation

independent sets,
no pairing!

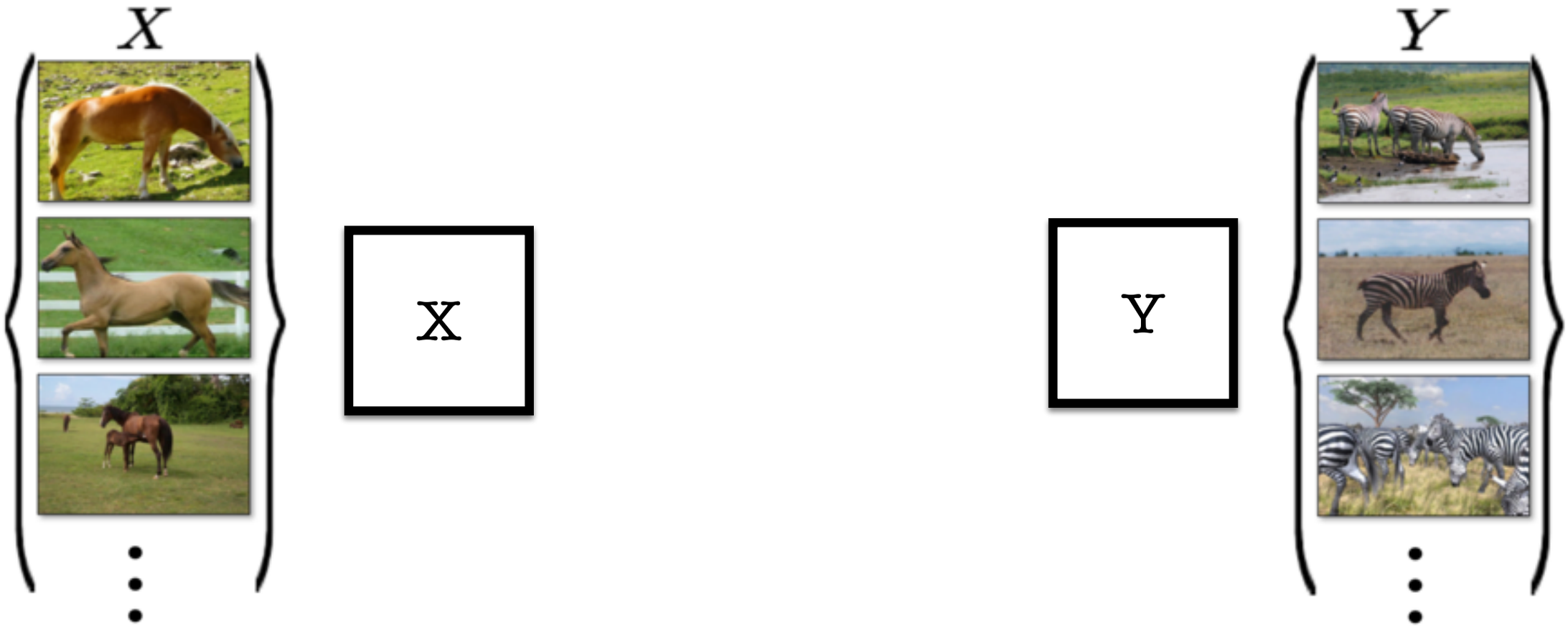




mode collapse

slide credit: Jun-Yan Zhu, Taesung Park

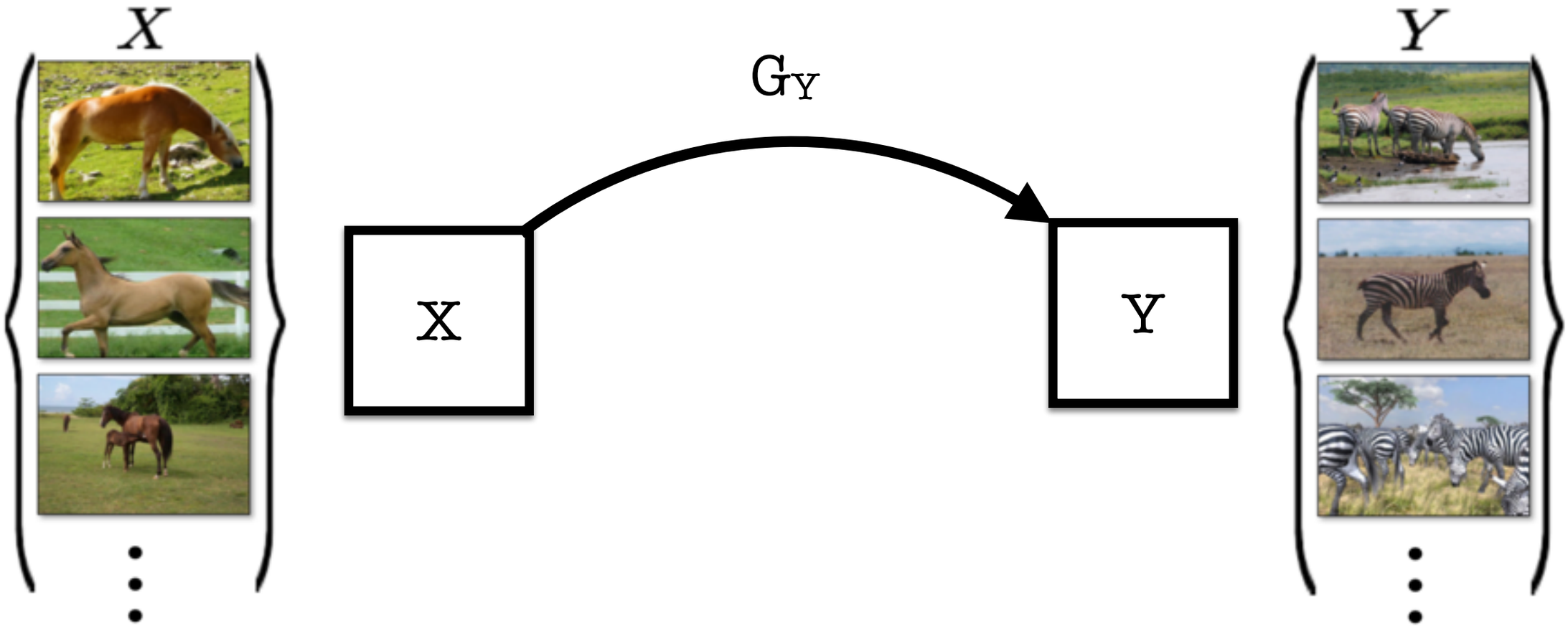
CycleGan: Cycle -Consistent Adversarial Networks



[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

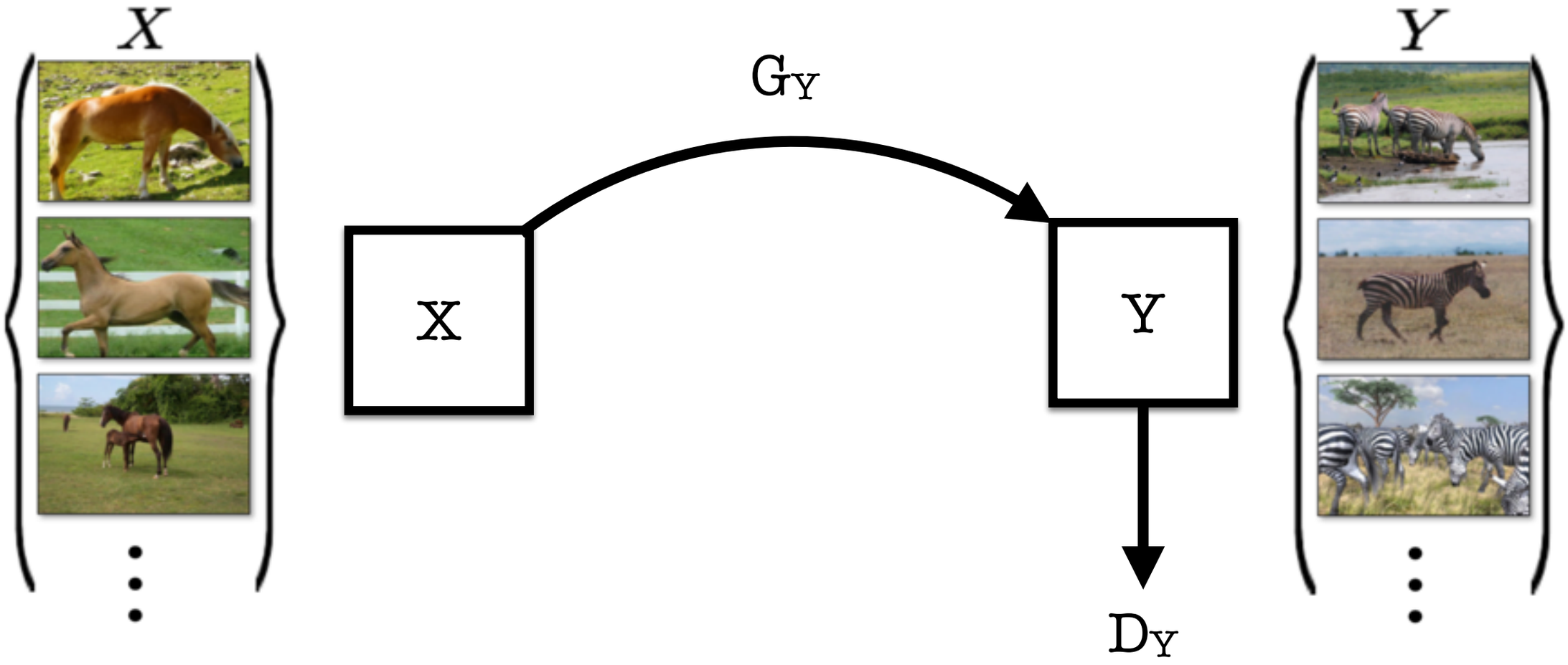
CycleGan: Cycle -Consistent Adversarial Networks



[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

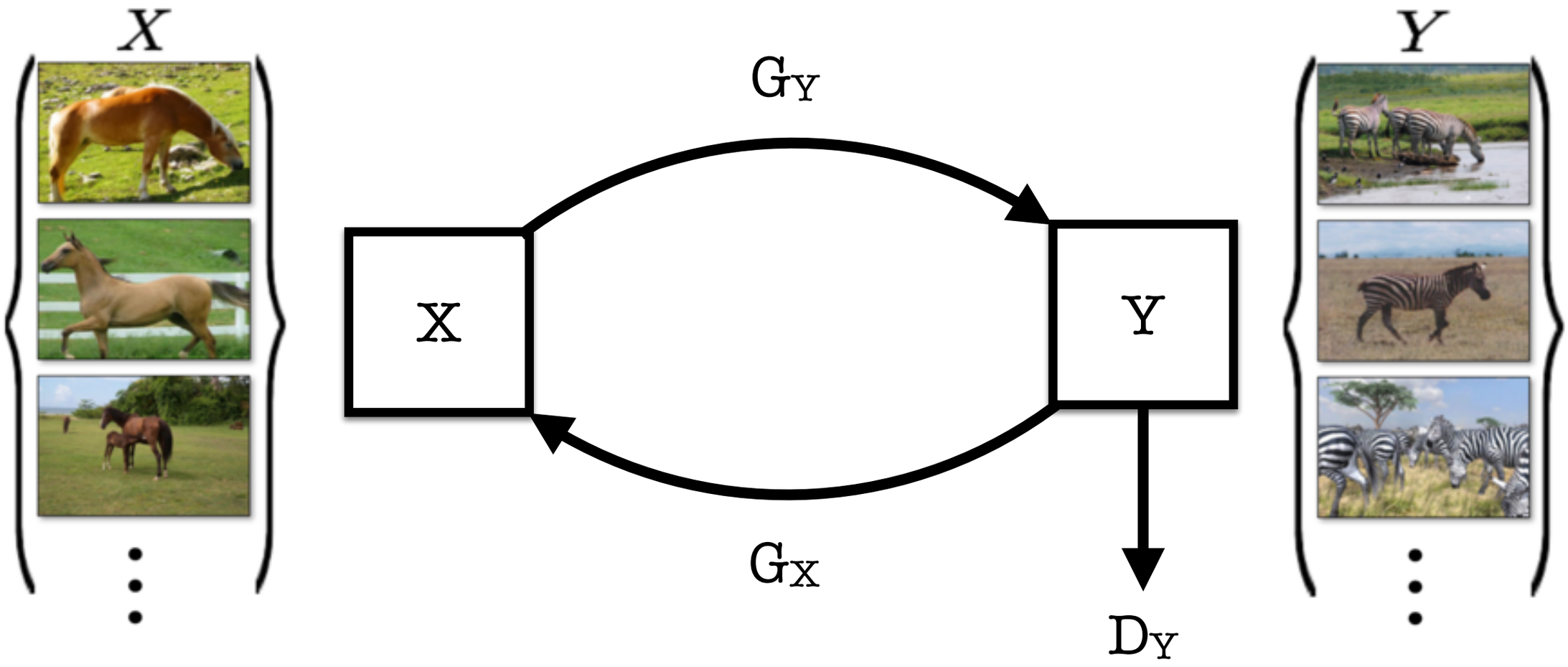
CycleGan: Cycle -Consistent Adversarial Networks



[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

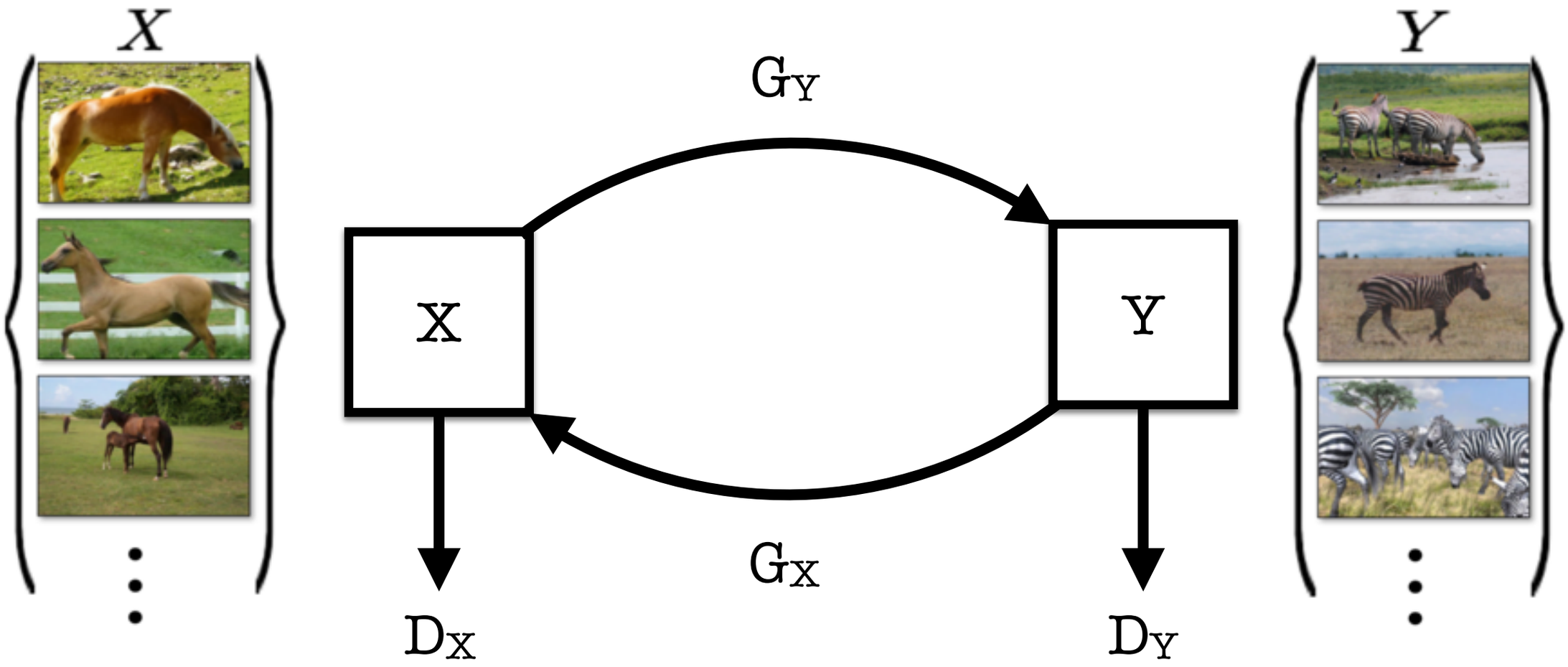
CycleGan: Cycle -Consistent Adversarial Networks



[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

CycleGan: Cycle -Consistent Adversarial Networks

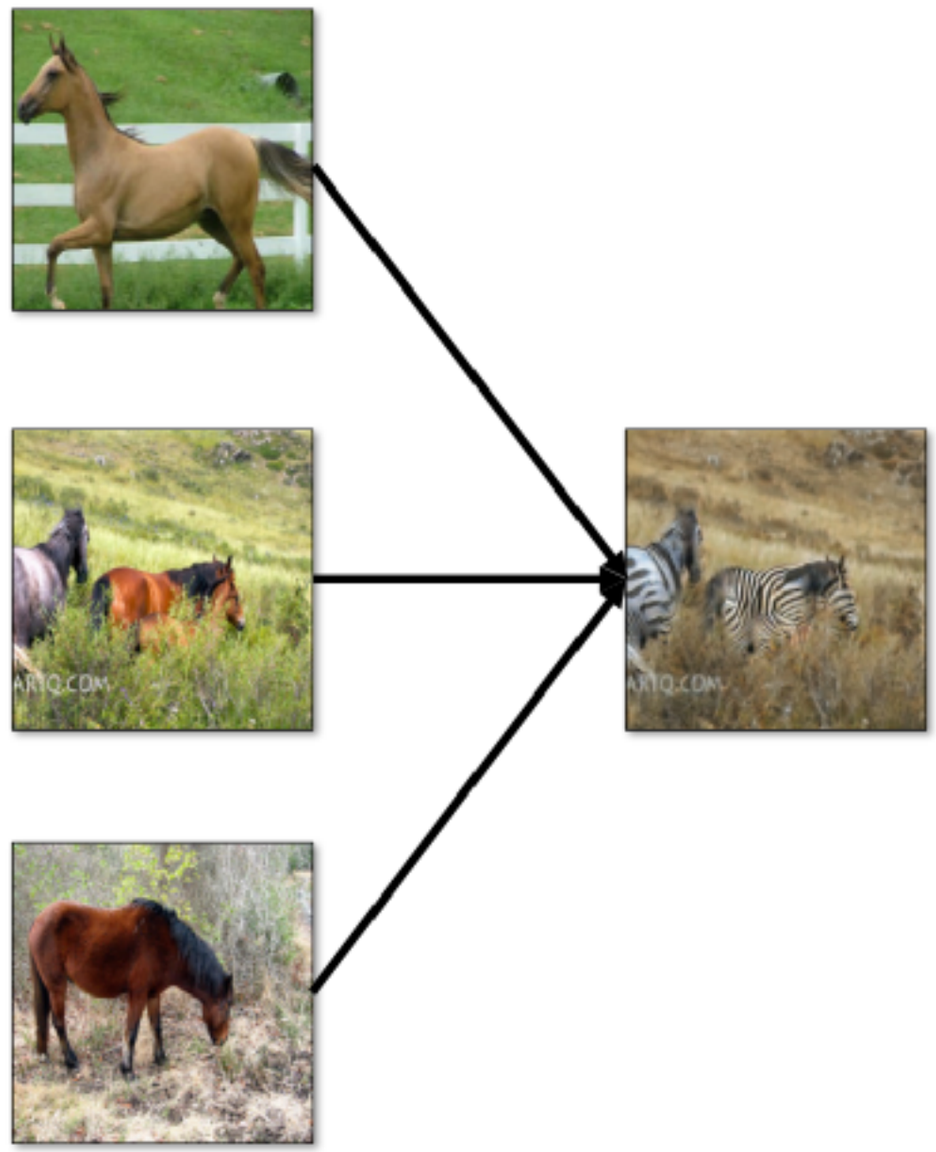


[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

CycleGan: Cycle -Consistent Adversarial Networks

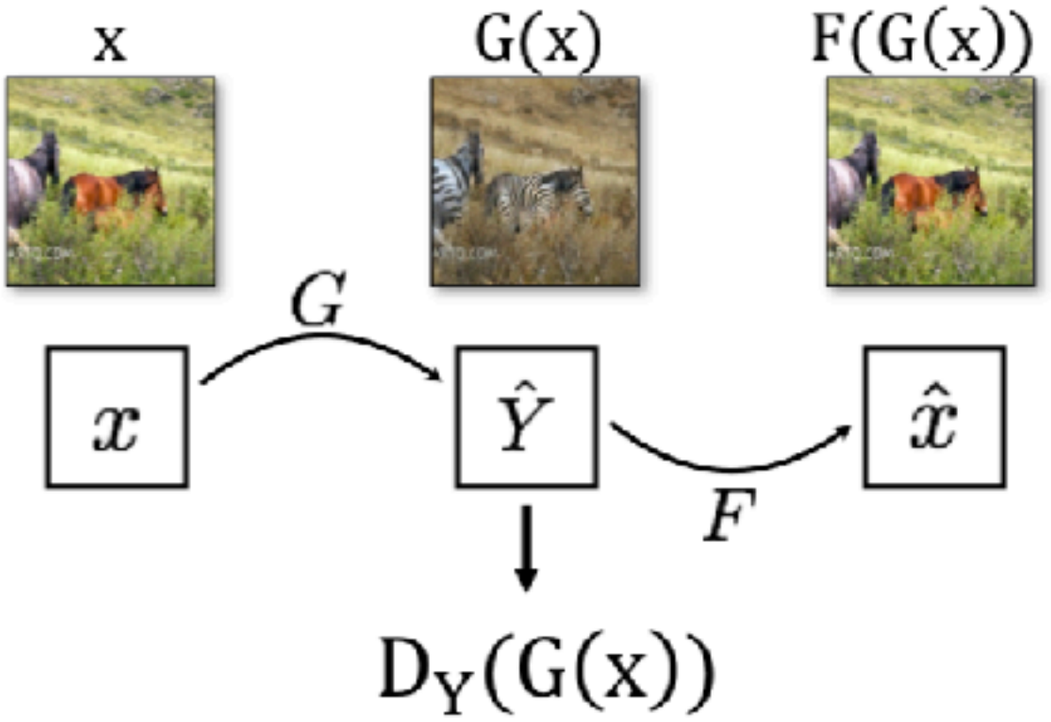
Does not prevent mode collapse



[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

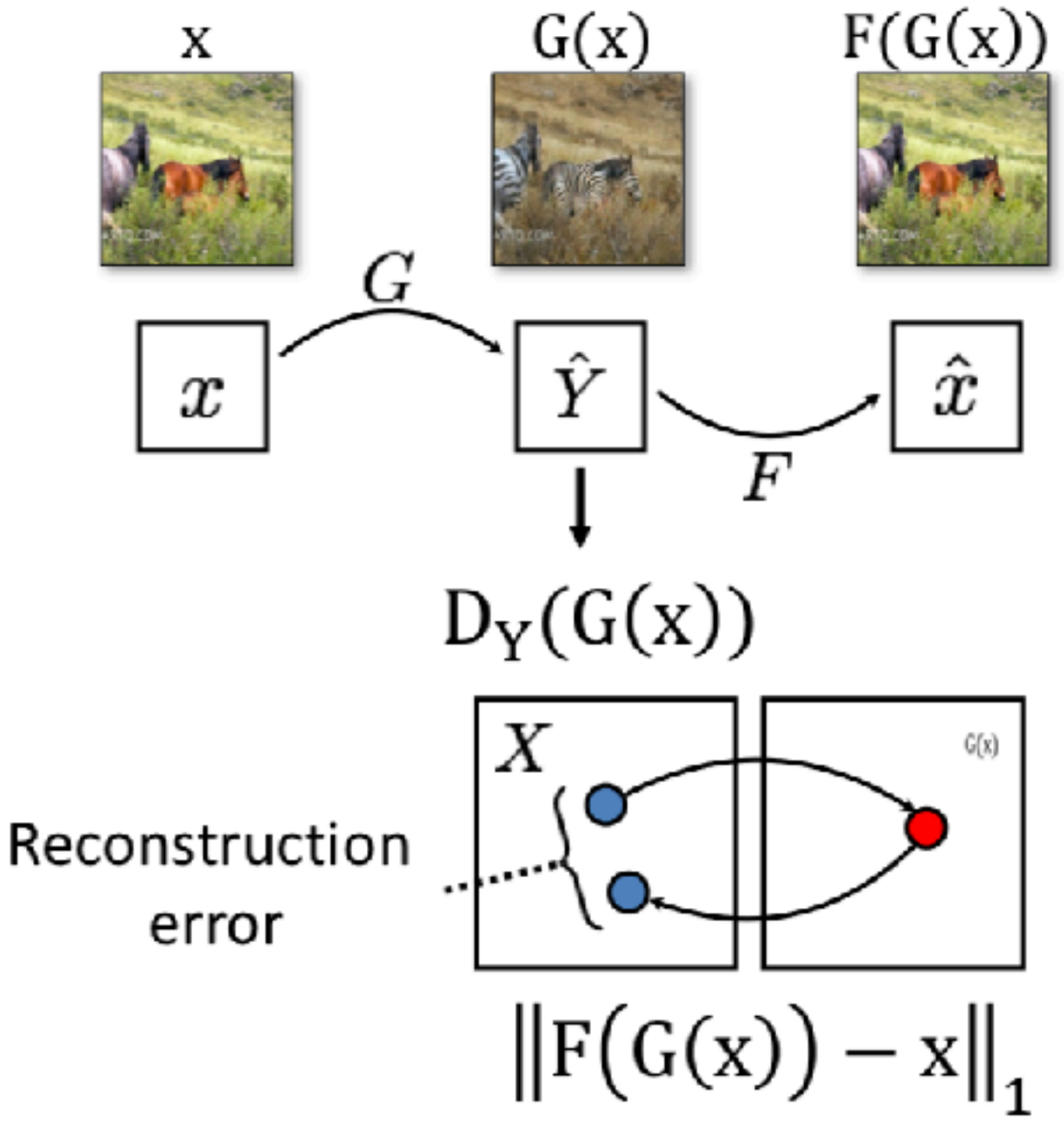
CycleGan: Cycle -Consistent Adversarial Networks



[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

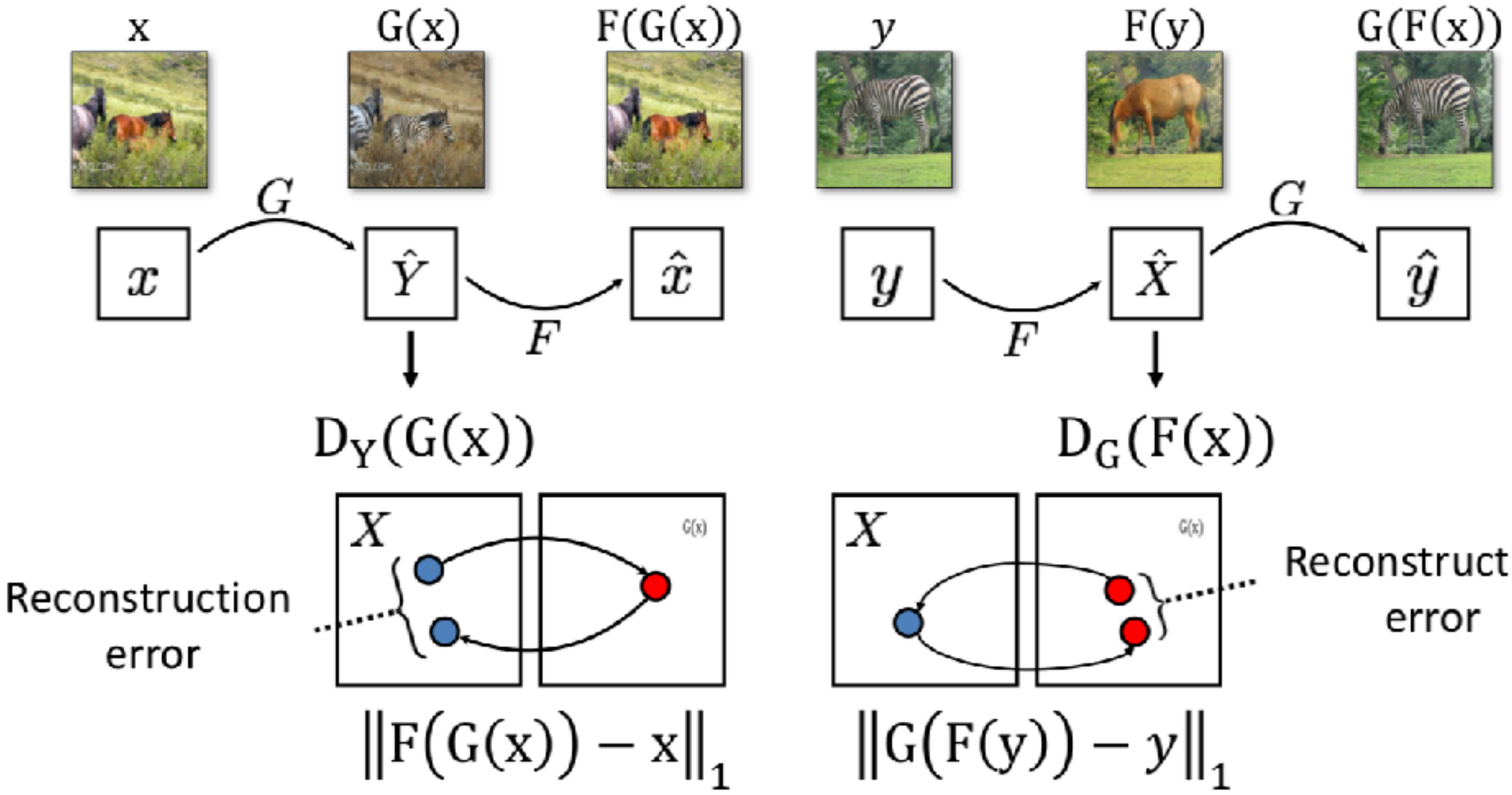
CycleGan: Cycle -Consistent Adversarial Networks



[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

CycleGan: Cycle -Consistent Adversarial Networks

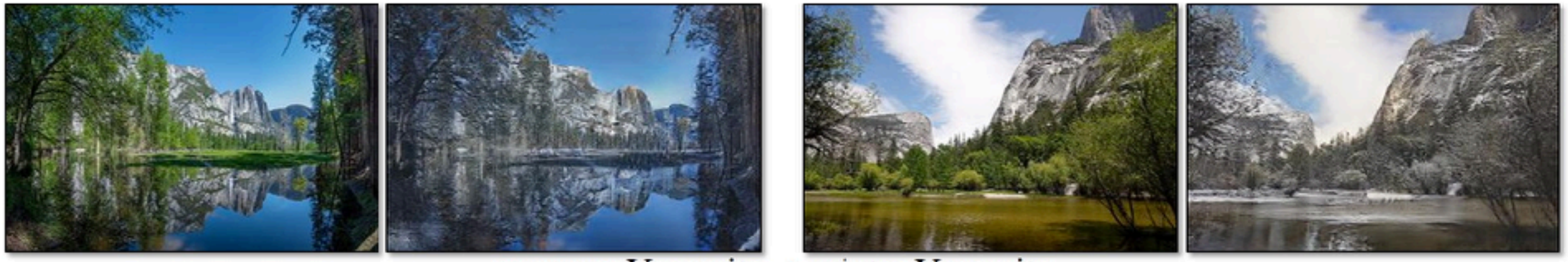


[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

CycleGan: Cycle -Consistent Adversarial Networks

winter Yosemite → summer Yosemite



summer Yosemite → winter Yosemite

[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

CycleGan Application: Rendering to Image

image credit: Markus Müller

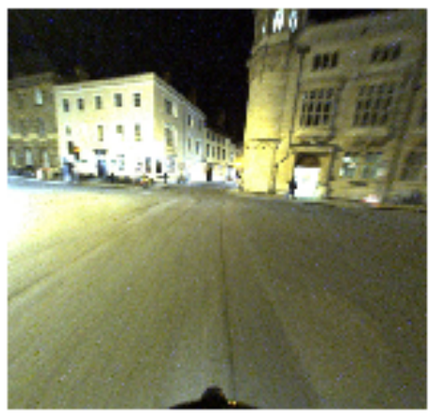
CycleGan Application: Image Retrieval

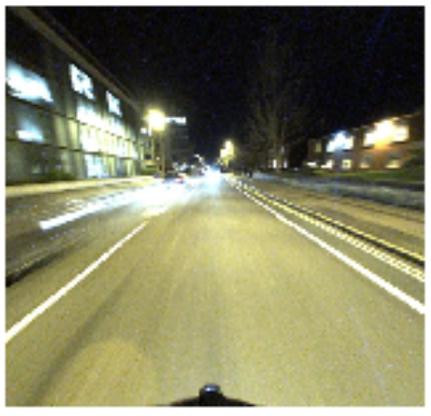
[Maddern, Pascoe, Linegar, Newman, 1 Year, 1000km: The Oxford RobotCar Dataset, IJRR 2017]

[Sattler et al., Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions, CVPR 2018]

CycleGan Application: Image Retrieval

Night-Time Query





Retrieved Day Image

[Anoosheh et al., Night-to-Day Image Translation for Retrieval-based Localization, ICRA 2019]

slide credit: Asha Anoosheh

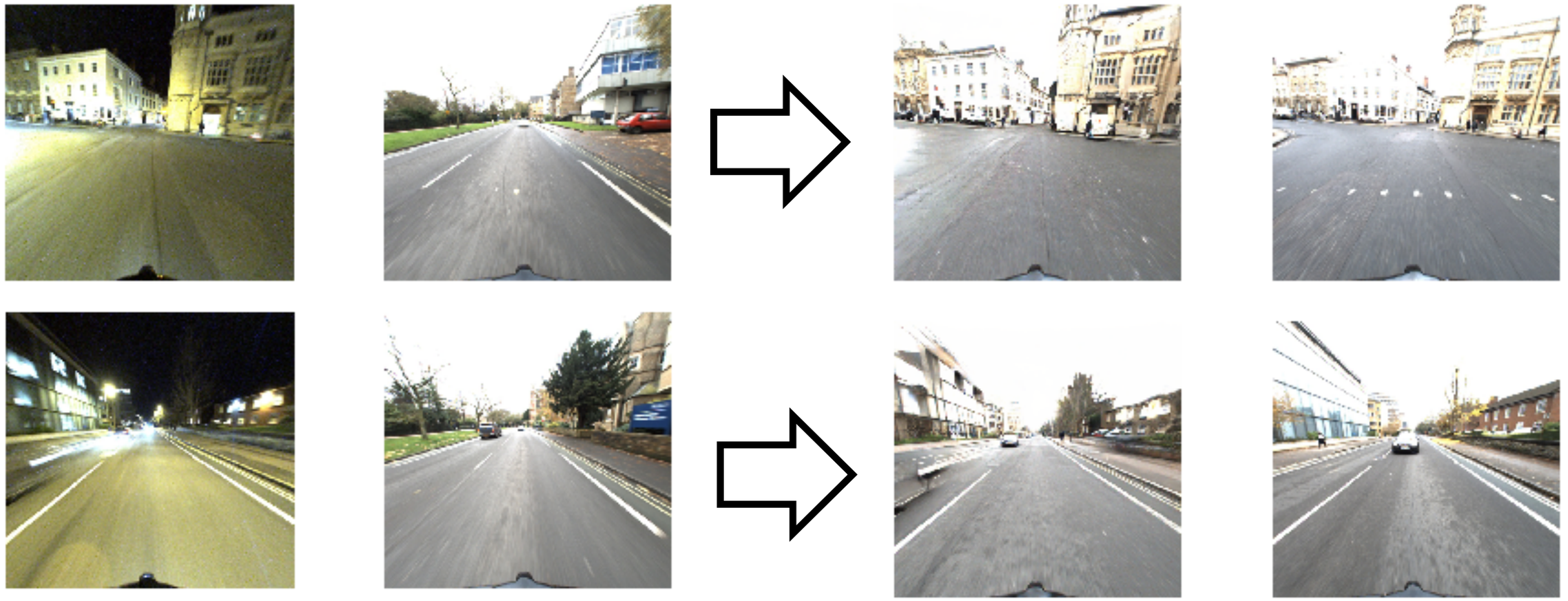
CycleGan Application: Image Retrieval

Night-Time Query

Retrieved Day Image

Translated Query

Retrieved Day Image



[Anoosheh et al., Night-to-Day Image Translation for Retrieval-based Localization, ICRA 2019]

slide credit: Asha Anoosheh

Lessons Learned

- Main lessons from this lecture
  - Generative networks learn mapping from simple probability distribution to target probability distribution
  - Adversarial networks: Learn the loss function
  - Unsupervised image-to-image translation via cycle consistency
- Next lecture: Visual Localization & Feature Learning

Reading Materials

- Start: Goodfellow, Bengio, Courville, Deep Learning, MIT press (Chapter 20.10.4)
- Continue: Blog posts on Wasserstein GANs, Tutorial on VAEs
- Advanced material:
  - ICCV 2017 GAN Tutorial (website with slides and videos)
  - CVPR 2018 GAN Tutorial (website with slides and videos)
  - Tensorflow implementations of various GANs
  - Read the papers listed on the slides
- GANs are an active research topic, be prepared to read a lot

Next Lecture

Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	Filtering, gradients, scale	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
Feb. 17	Camera Geometry	Lab 4
Feb. 20	More camera geometry	
Feb. 24	Generative neural networks	
Feb. 27	Generative neural networks	
Mar. 2	**Visual Localization & Feature Learning**	
Mar. 9	No lecture	