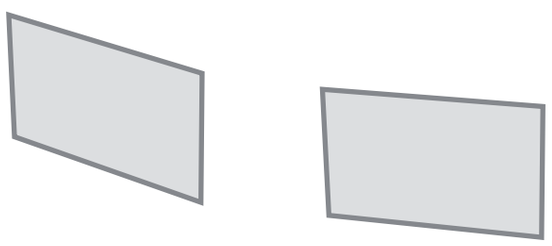
SSY097 - Image Analysis

Lecture 11 - Generative Neural Networks

Torsten Sattler

Last Lecture

Initialize motion from two views

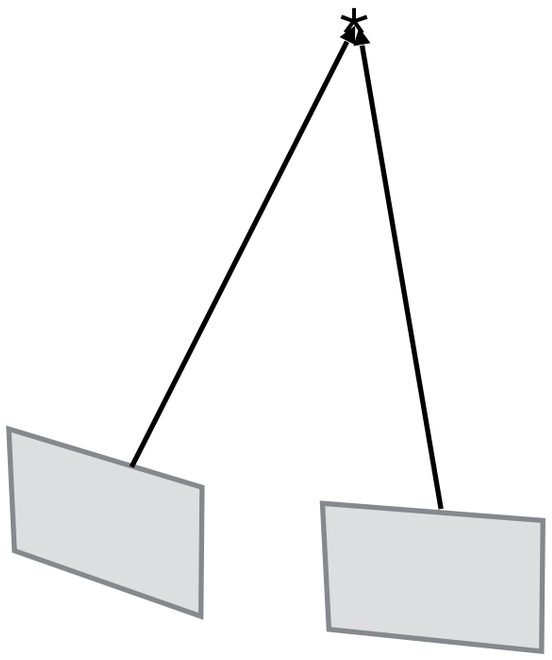


Relative pose for two images

Last Lecture

Initialize motion from two views

Initialize structure from two views

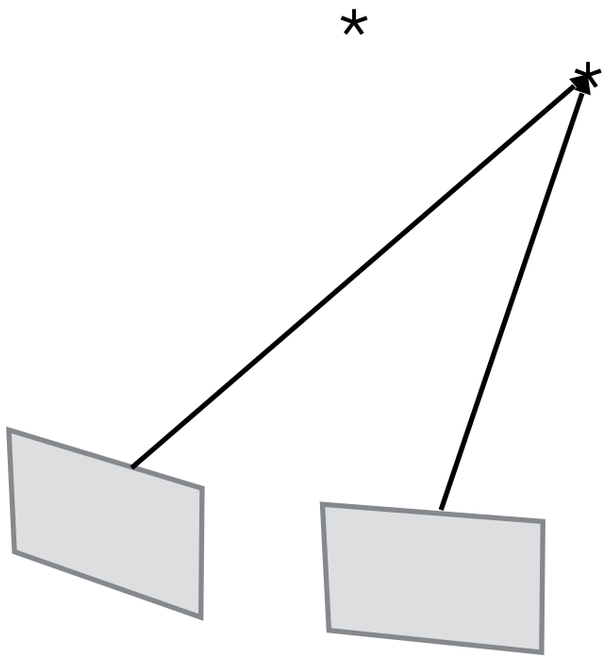


Triangulate 3D points

Last Lecture

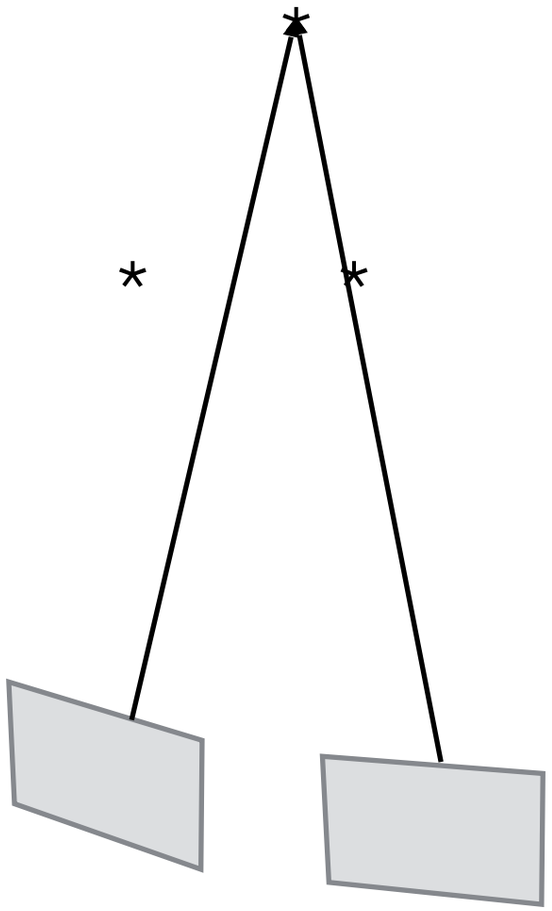
Initialize motion from two views

Initialize structure from two views



Triangulate 3D points

Last Lecture

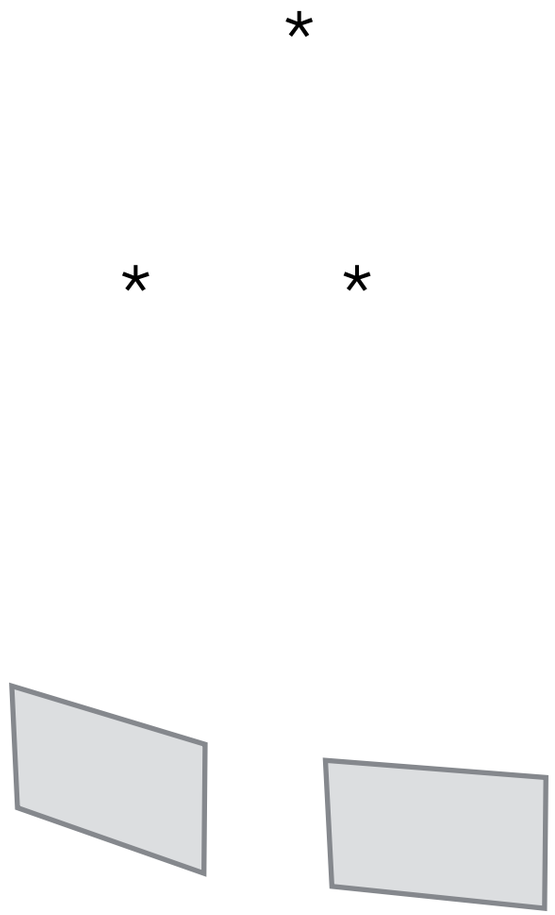


Initialize motion from two views

Initialize structure from two views

Triangulate 3D points

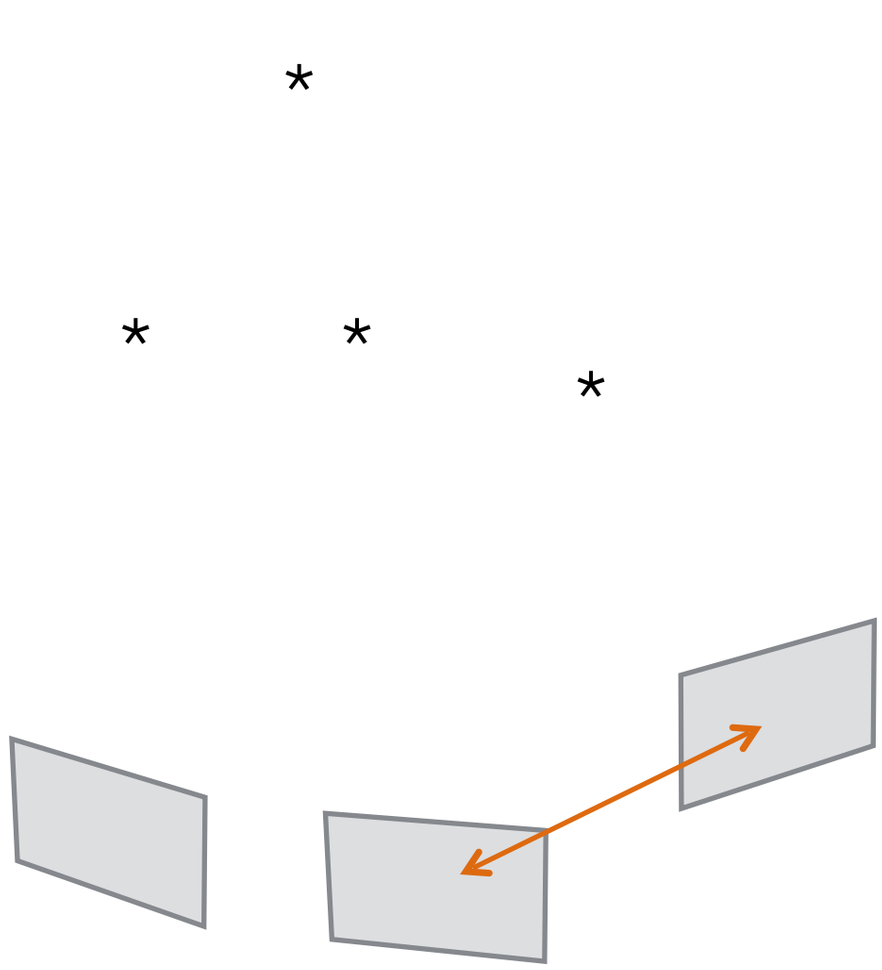
Last Lecture



Initialize motion from two views

Initialize structure from two views

Last Lecture



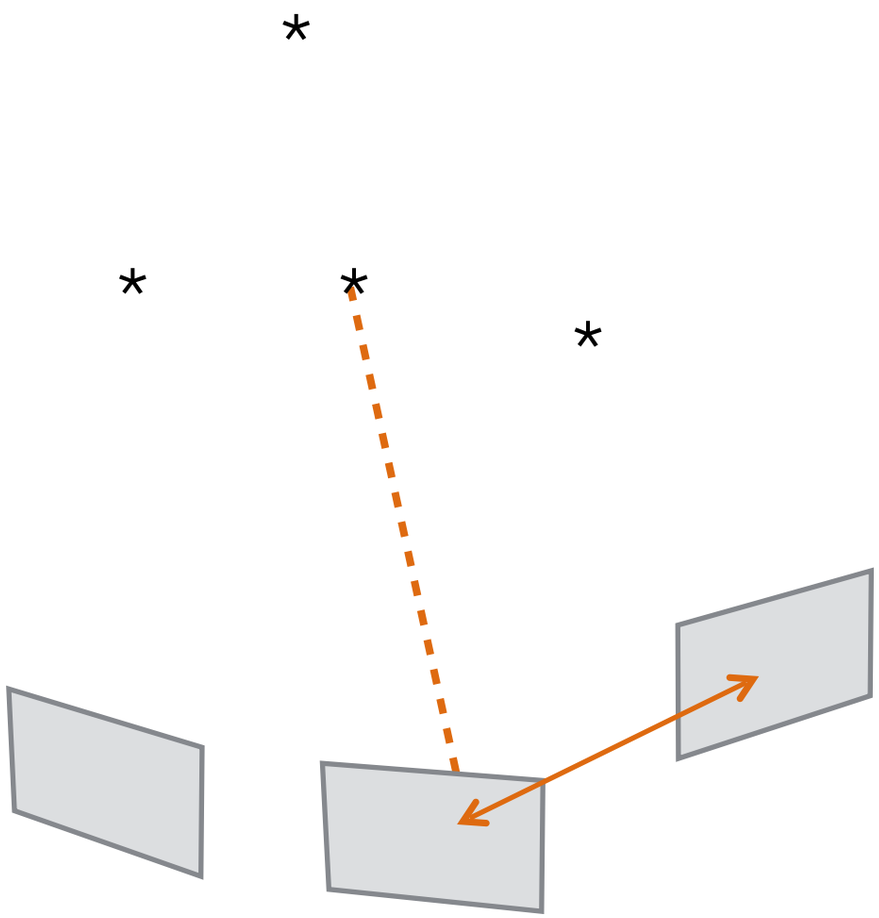
Initialize motion from two views

Initialize structure from two views

Extend motion

Match features

Last Lecture



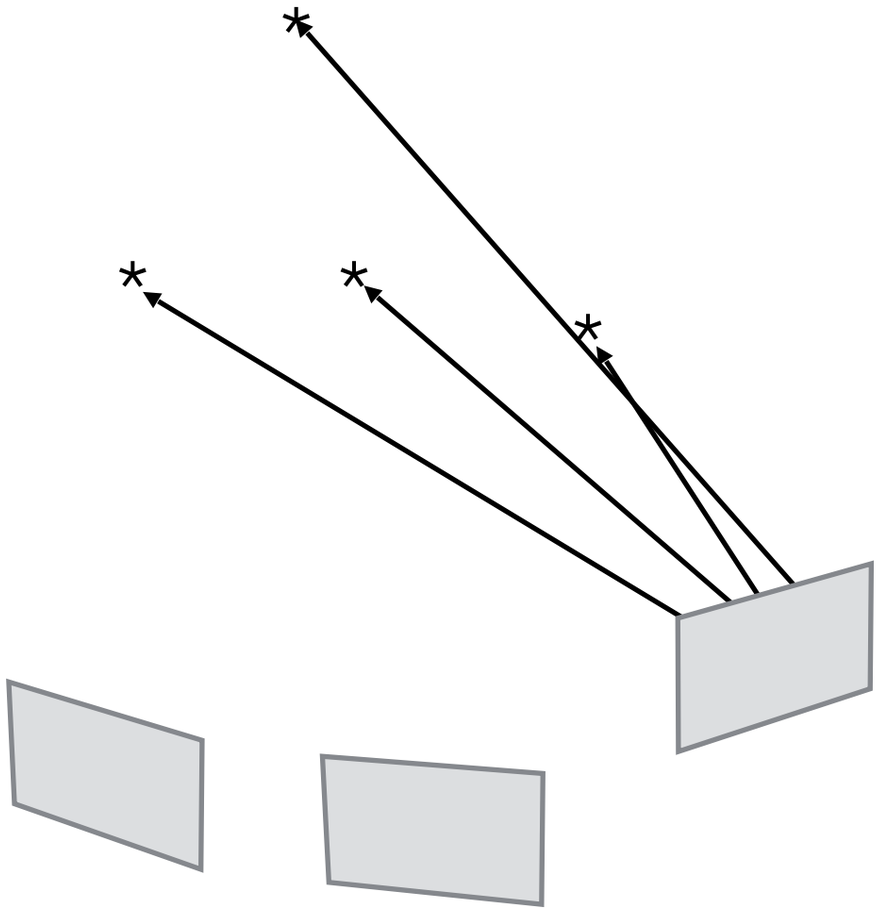
Initialize motion from two views

Initialize structure from two views

Extend motion

Transfer matches to 3D

Last Lecture



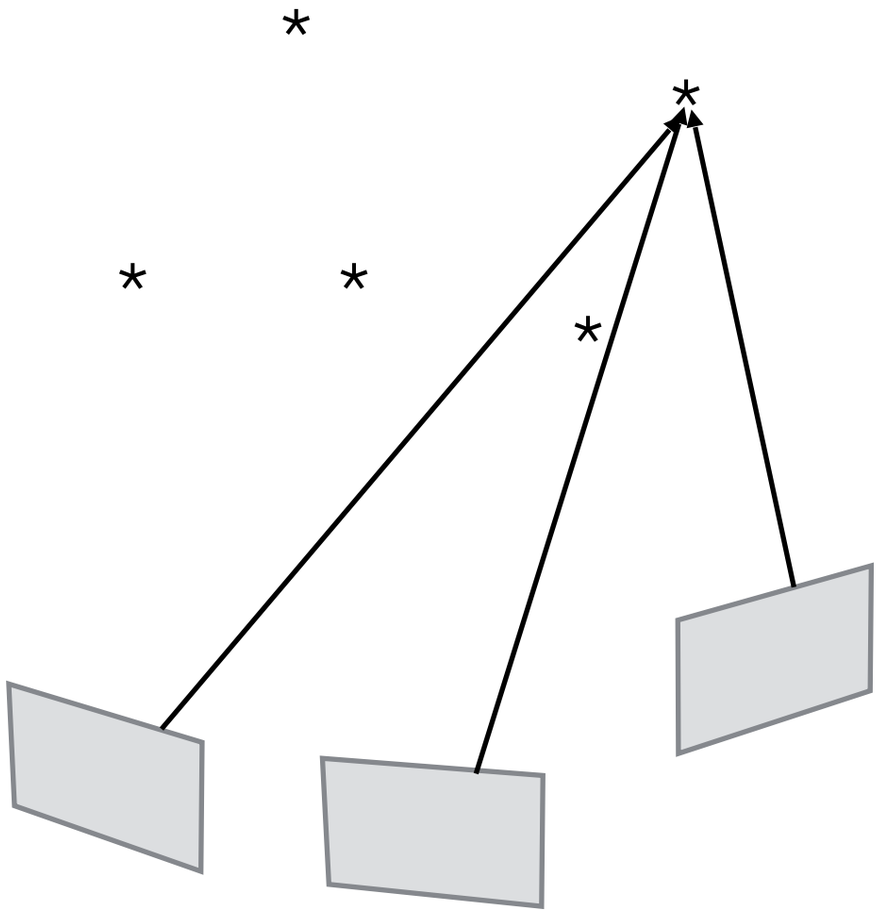
Initialize motion from two views

Initialize structure from two views

Extend motion

Camera pose for third camera

Last Lecture



Initialize motion from two views

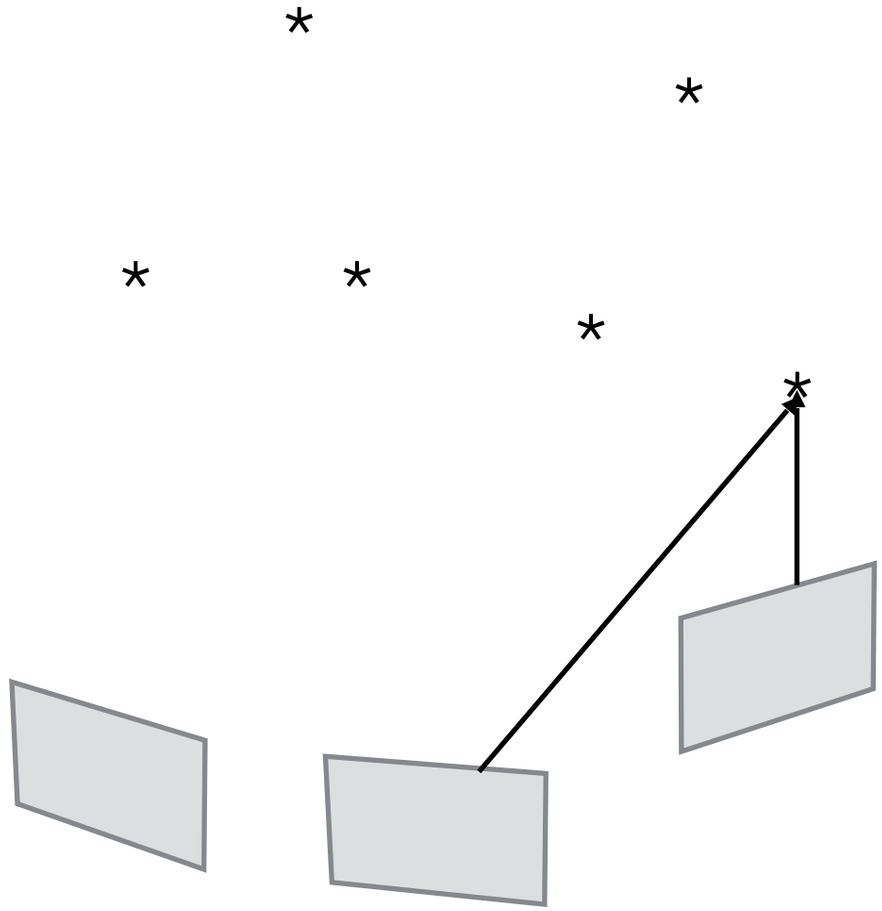
Initialize structure from two views

Extend motion

Extend structure

Triangulate points

Last Lecture



Initialize motion from two views

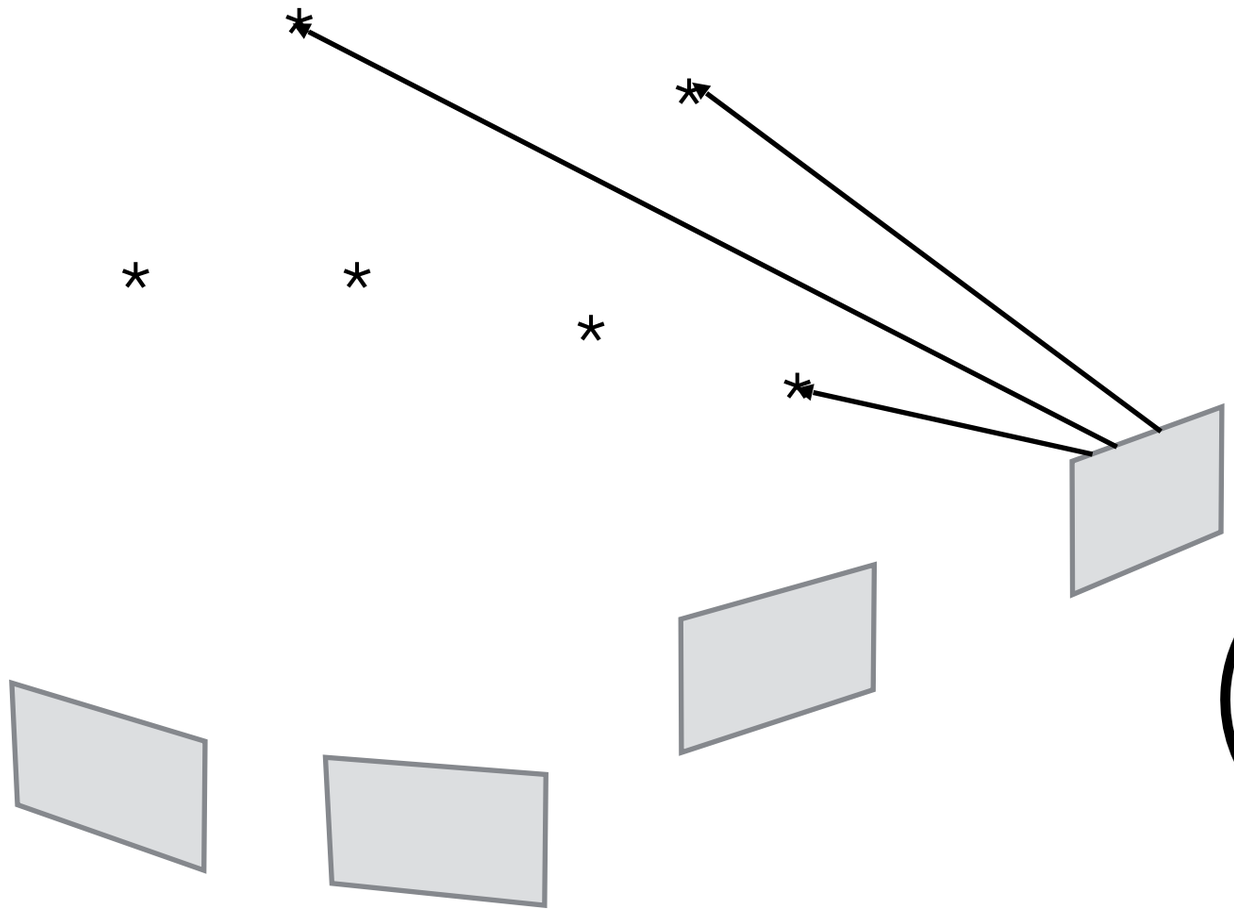
Initialize structure from two views

Extend motion

Extend structure

Triangulate points

Last Lecture



Initialize motion from two views

Initialize structure from two views

Extend motion

Extend structure

Camera pose for fourth image

Last Lecture

3D point $\mathbf{X}$ seen from camera with pose $\mathbb{R}$, $\mathbf{t}$, intrinsics $\mathbb{K}$

$$\mathbb{K}\left(\mathbb{R}\mathbf{X}+\mathbf{t}\right) \quad = \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

3D reconstructions defined up to scale

Last Lecture

3D point $\mathbf{X}$ seen from camera with pose $\mathbf{R}$, $\mathbf{t}$, intrinsics $\mathbf{K}$

$$\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) \quad = \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

Scale 3D scene by arbitrary factor $s \neq 0$

$$\mathbf{K}\left(\mathbf{R}(s\mathbf{X})+s\mathbf{t}\right) \quad = \quad s\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right)$$

3D reconstructions defined up to scale

Last Lecture

3D point $\mathbf{X}$ seen from camera with pose $\mathbf{R}$, $\mathbf{t}$, intrinsics $\mathbf{K}$

$$\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

Scale 3D scene by arbitrary factor $s \neq 0$

$$\begin{aligned} \mathbf{K}\left(\mathbf{R}(s\mathbf{X})+s\mathbf{t}\right) &= s\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) \\ &= s\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{sx}{sz} \\ \frac{sy}{sz} \end{pmatrix} \end{aligned}$$

3D reconstructions defined up to scale

Last Lecture

3D point $\mathbf{X}$ seen from camera with pose $\mathbf{R}$, $\mathbf{t}$, intrinsics $\mathbf{K}$

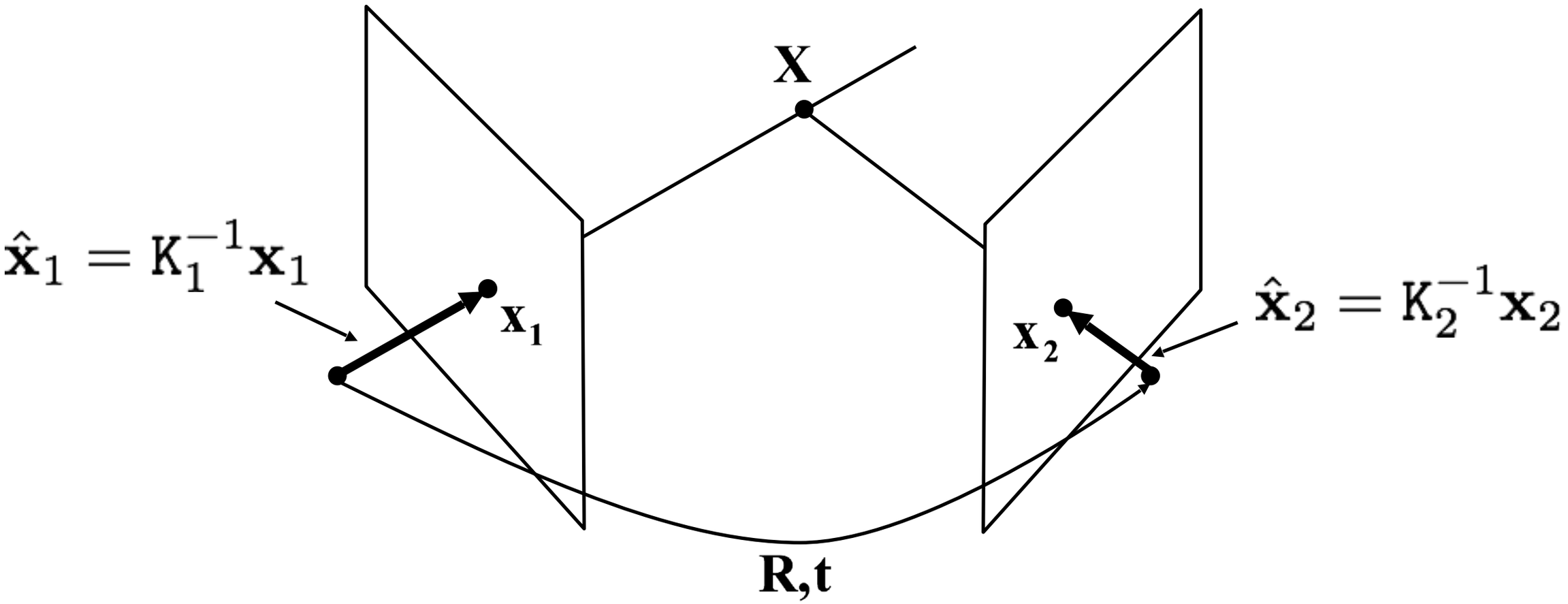
$$\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) \quad = \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

Scale 3D scene by arbitrary factor $s \neq 0$

$$\begin{aligned} \mathbf{K}\left(\mathbf{R}(s\mathbf{X})+s\mathbf{t}\right) &= s\mathbf{K}\left(\mathbf{R}\mathbf{X}+\mathbf{t}\right) \\ &= s\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{sx}{sz} \\ \frac{sy}{sz} \end{pmatrix} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \end{aligned}$$

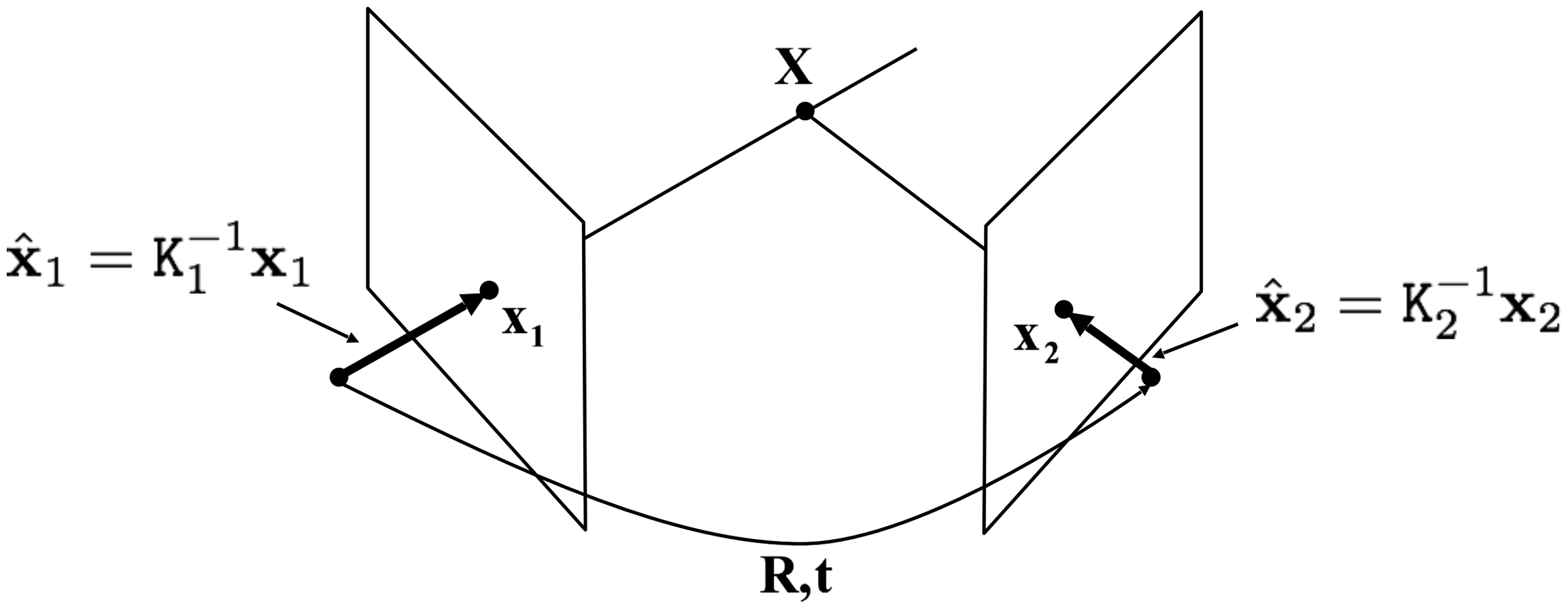
3D reconstructions defined up to scale

Last Lecture



Epipolar geometry

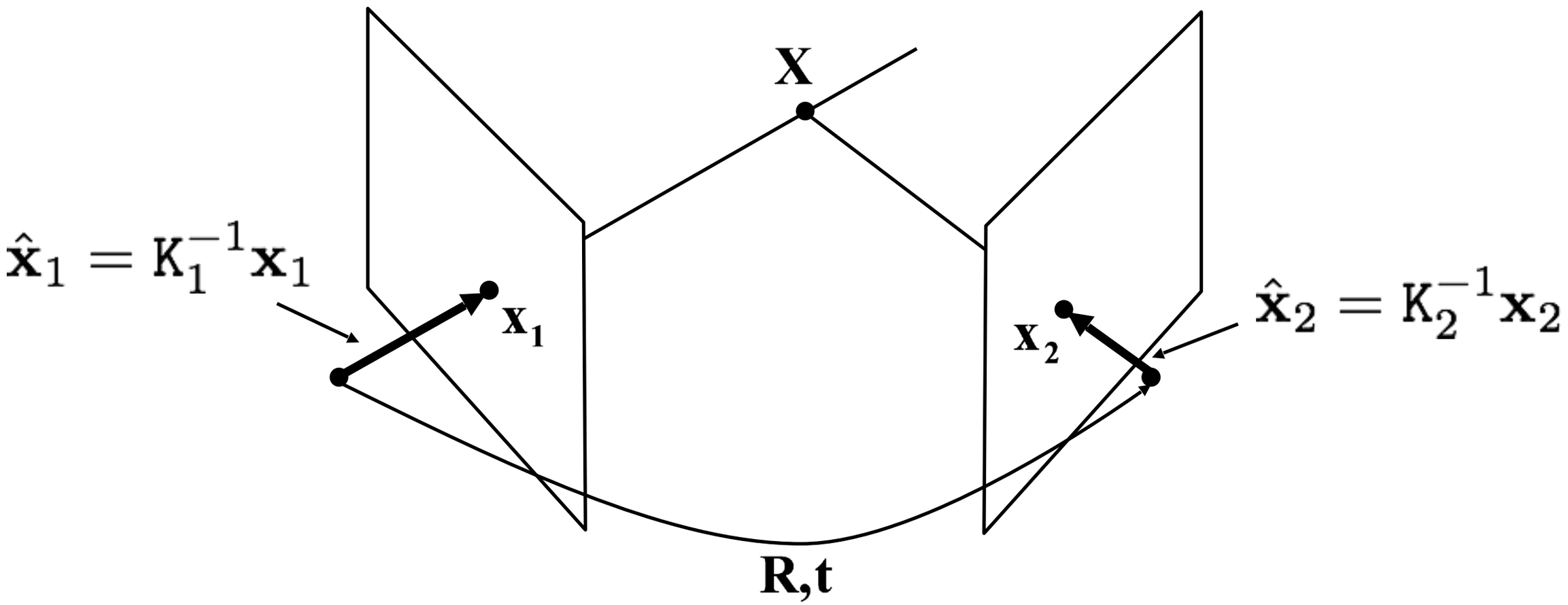
Last Lecture



$$\mathbf{0} = \hat{\mathbf{x}}_2^T \mathrm{E} \hat{\mathbf{x}}_1$$

Epipolar geometry

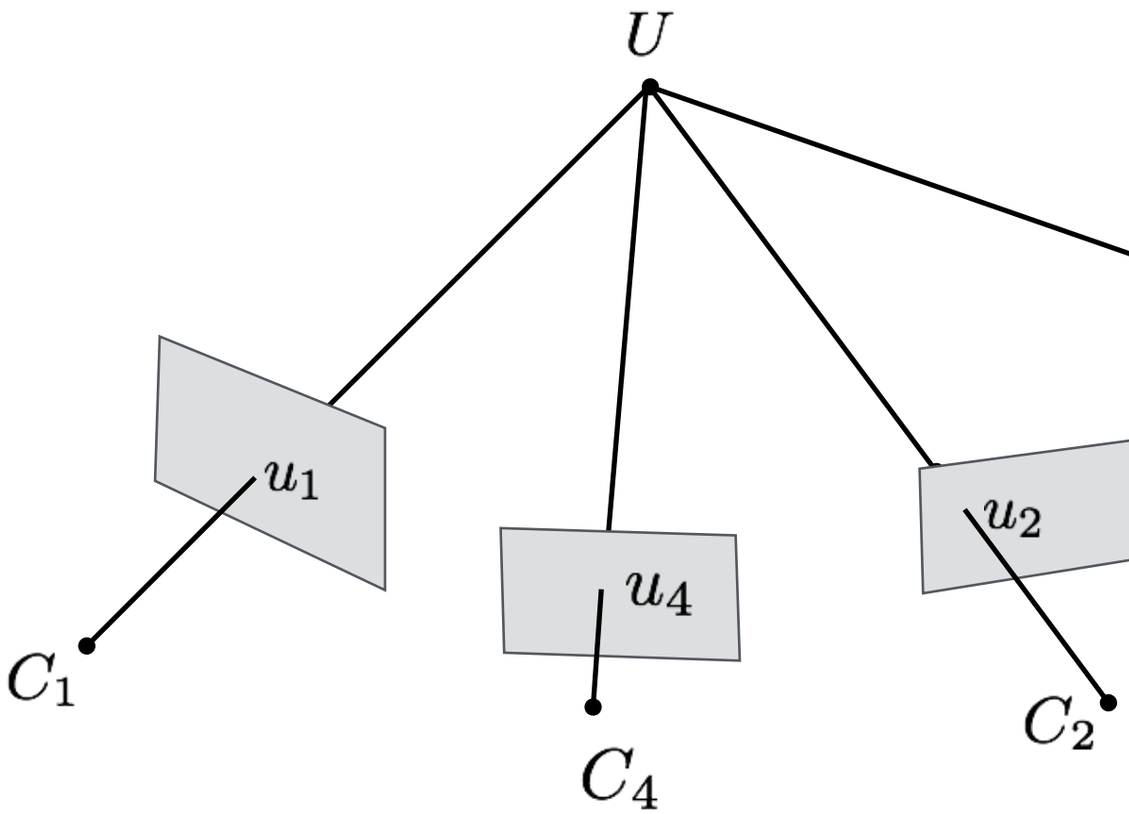
Last Lecture



$$\mathbf{0} = \hat{\mathbf{x}}_2^T \mathrm{E} \hat{\mathbf{x}}_1 \qquad \mathbf{0} = \mathbf{x}_2^T \mathrm{F} \mathbf{x}_1$$

Epipolar geometry

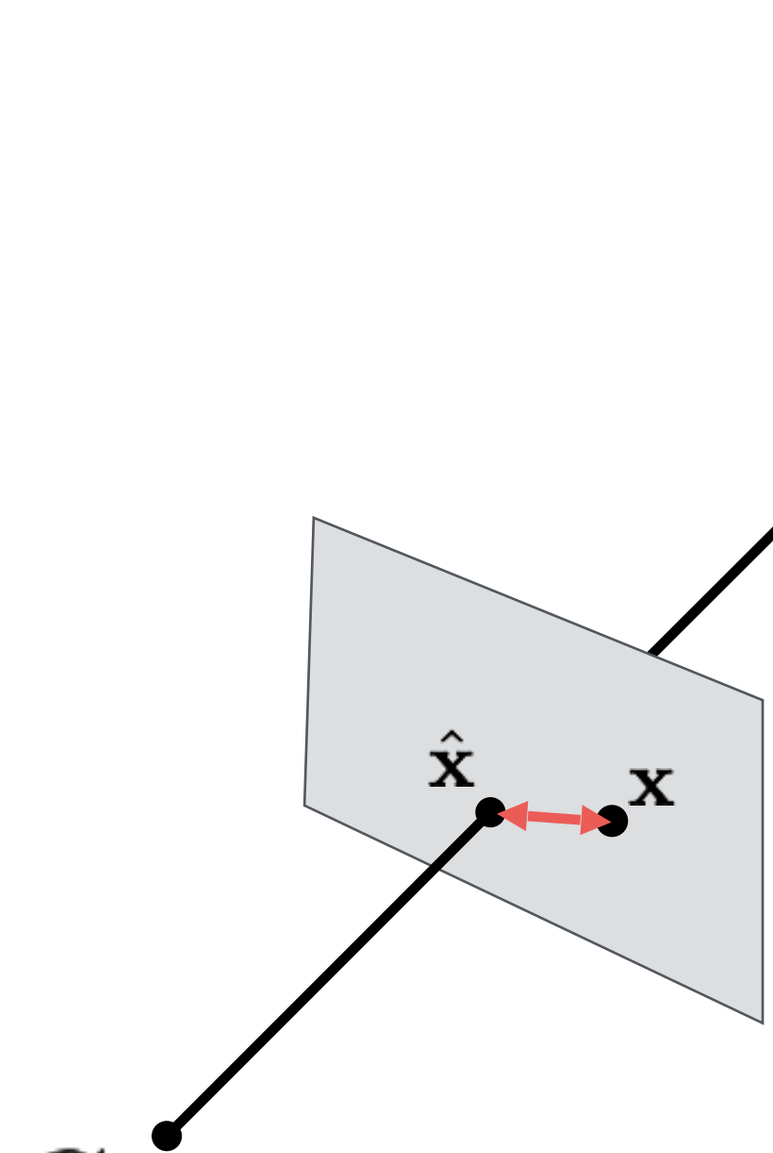
Last Lecture



$$\begin{pmatrix} \mathbf{P}_3 x - \mathbf{P}_1 \\ \mathbf{P}_3 y - \mathbf{P}_2 \\ \mathbf{P}'_3 x - \mathbf{P}'_1 \\ \mathbf{P}'_3 y - \mathbf{P}'_2 \end{pmatrix} \mathbf{X} = \mathbf{0}$$

Triangulation

Last Lecture



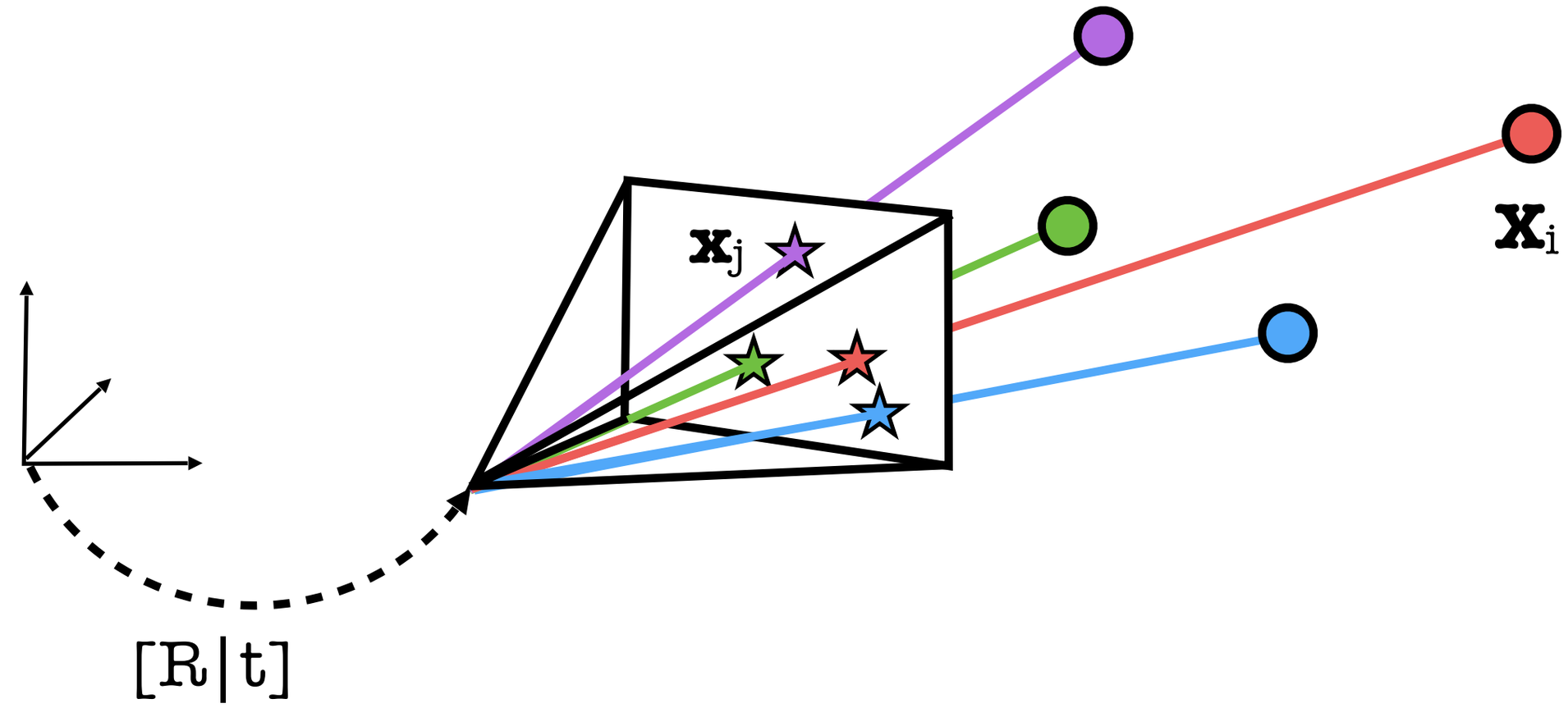
$$\hat{\mathbf{x}} = \begin{pmatrix} \frac{\mathbf{P}_1\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \\ \frac{\mathbf{P}_2\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \end{pmatrix}$$

reprojection error $||\mathbf{x} - \hat{\mathbf{x}}||$

$$\min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 = \min_{\mathbf{X}} \sum_i ||\mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i\mathbf{X}}{\mathbf{P}_3^i\mathbf{X}} \\ \frac{\mathbf{P}_2^i\mathbf{X}}{\mathbf{P}_3^i\mathbf{X}} \end{pmatrix} ||^2$$

Minimizing reprojection error

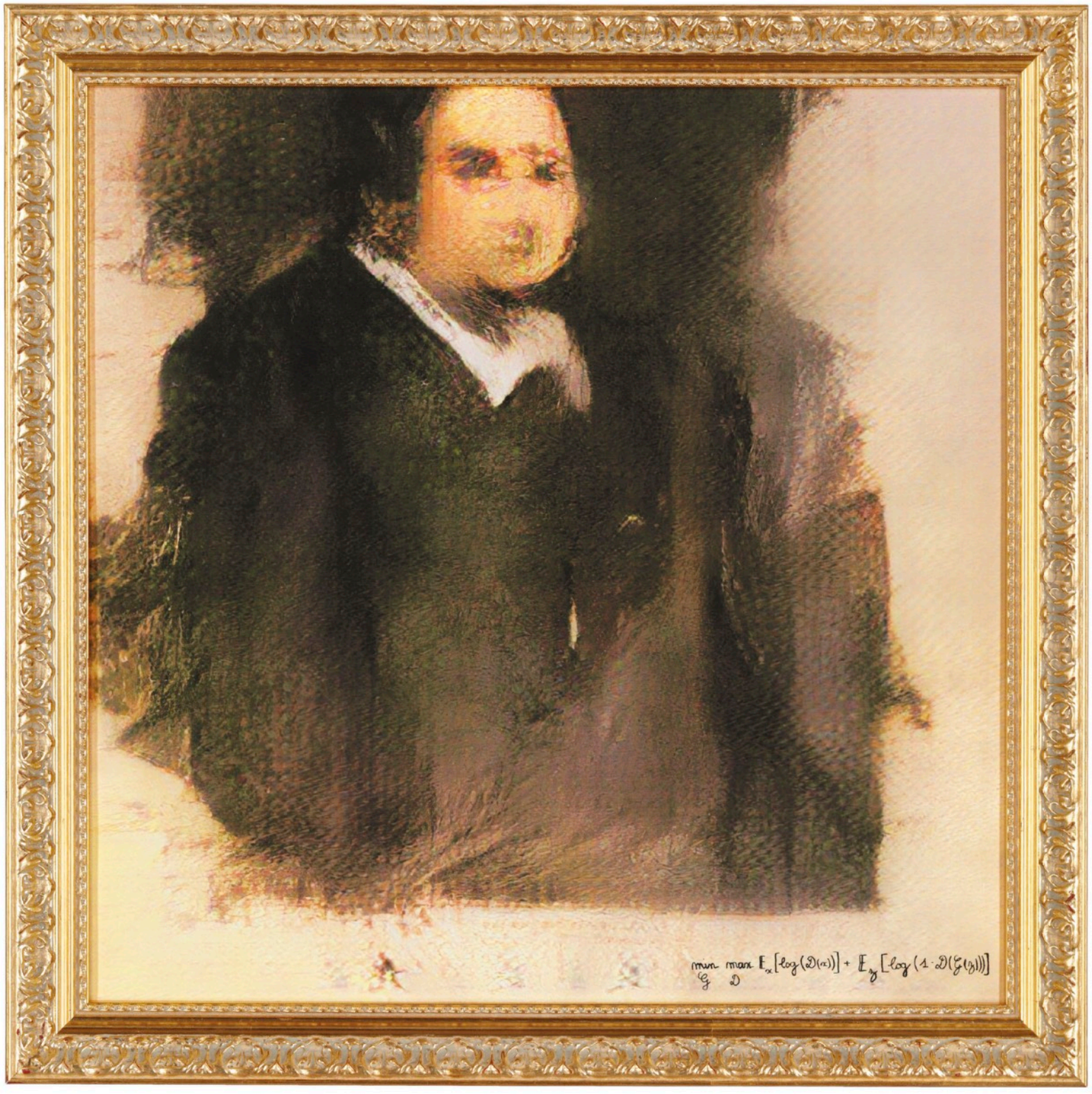
Last Lecture



Absolute pose estimation

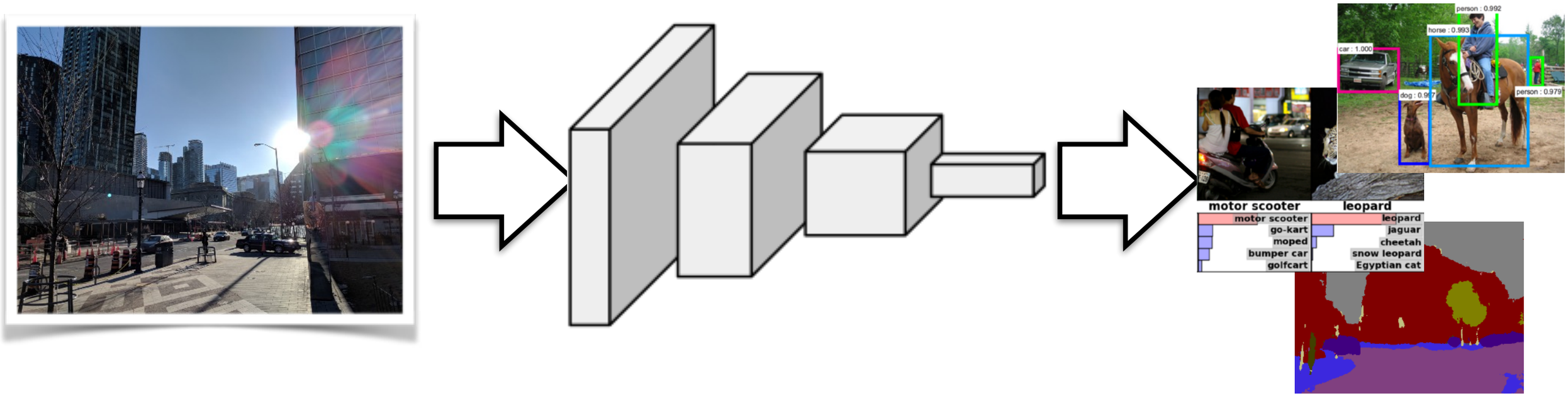
Today

Generative Neural Networks



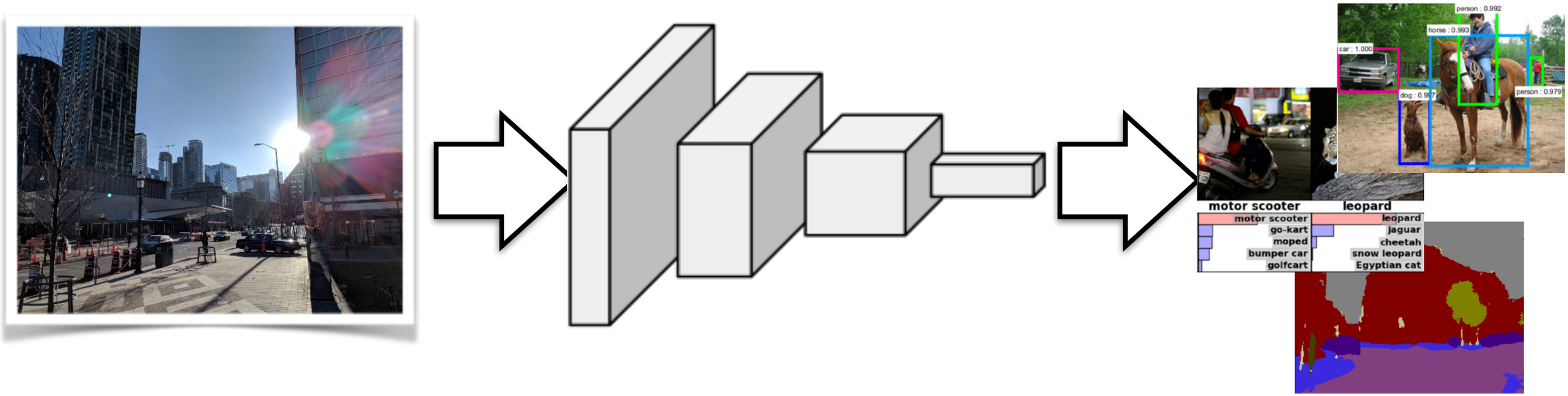
Portrait of Edmond Belamy, 2018. Sold for $432,500 on 25 October at Christie's in New York. Image © Obvious

Recap: CNNs



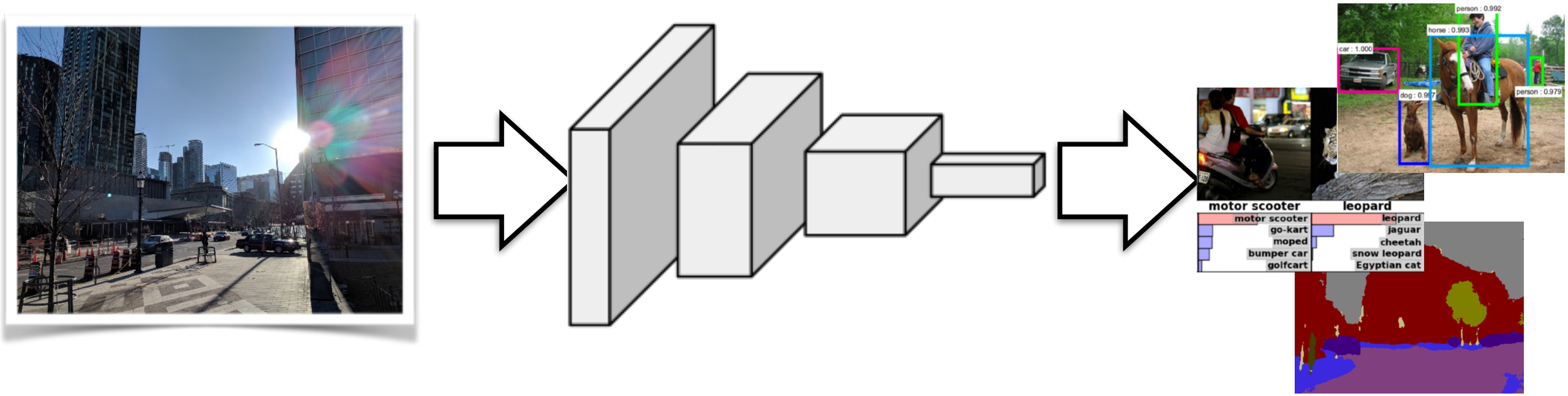
- Convolutional neural networks for classification, segmentation, regression, etc.

Recap: CNNs



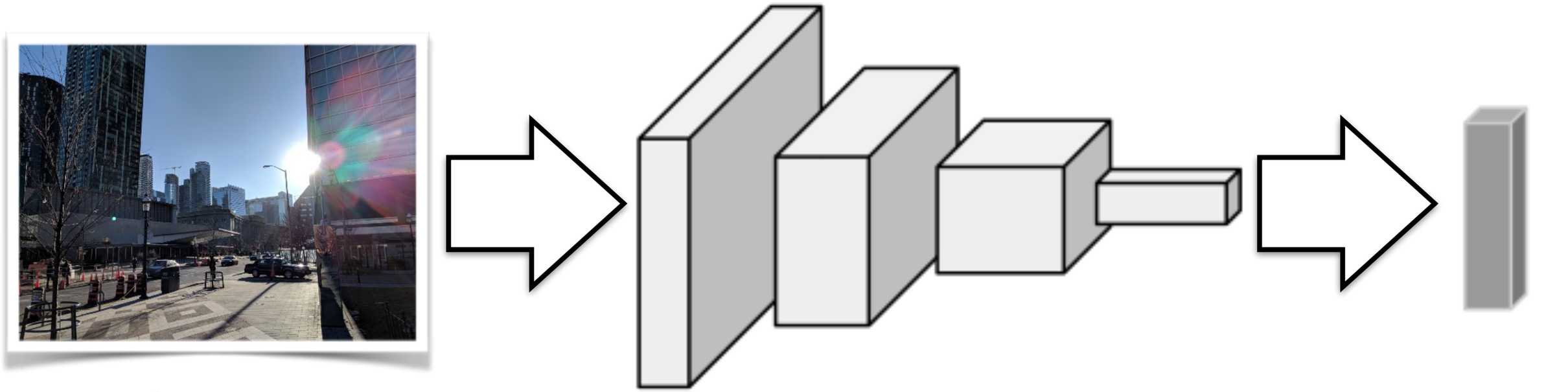
- Convolutional neural networks for classification, segmentation, regression, etc.
- Supervised learning: Training from (image, label) pairs

Recap: CNNs



- Convolutional neural networks for classification, segmentation, regression, etc.
- Supervised learning: Training from (image, label) pairs
- Training via stochastic gradient descent (or variants)

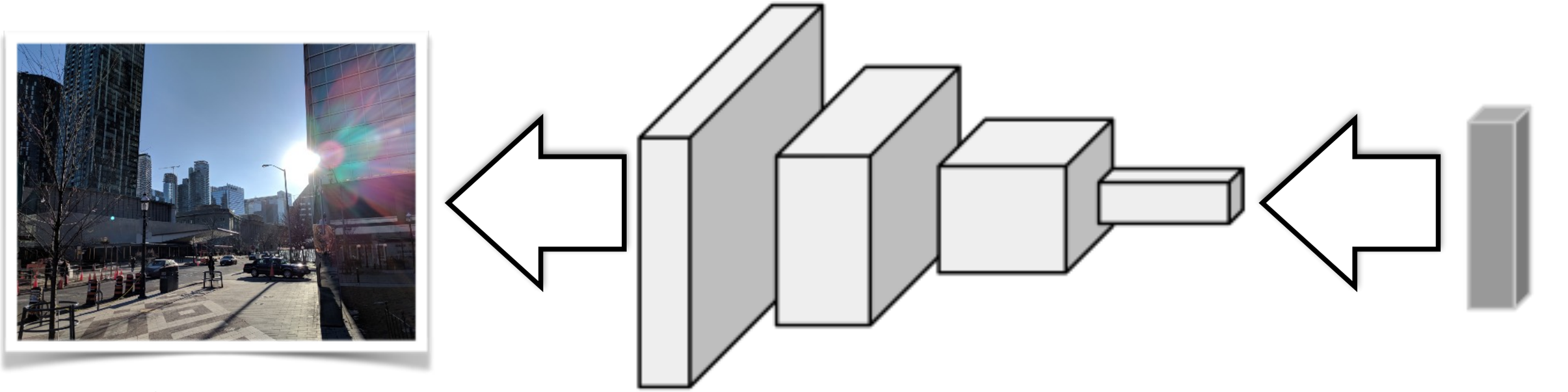
Generative Neural Networks



image

output

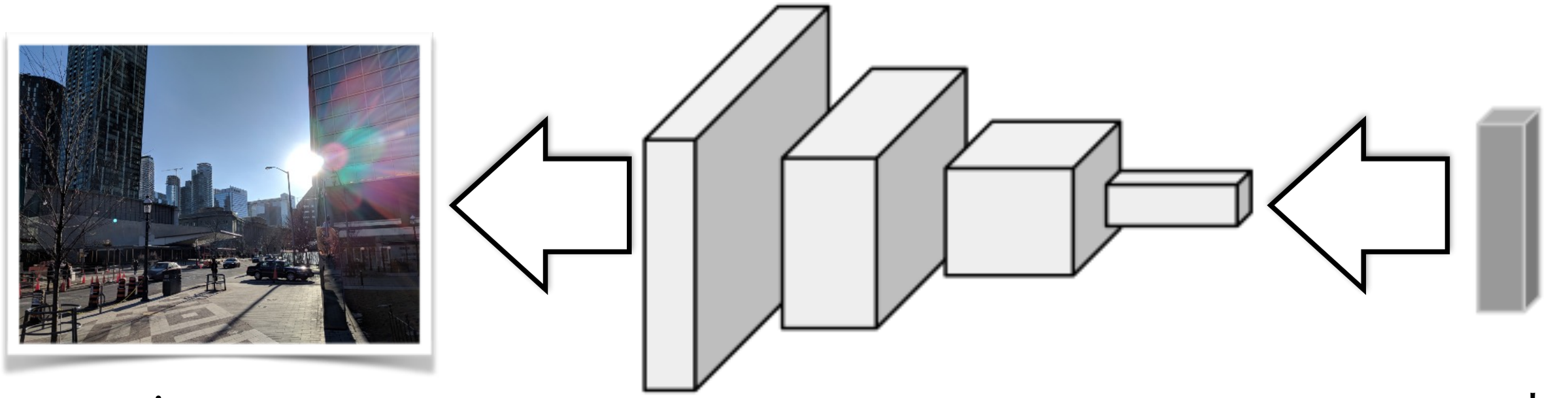
Generative Neural Networks



image

output

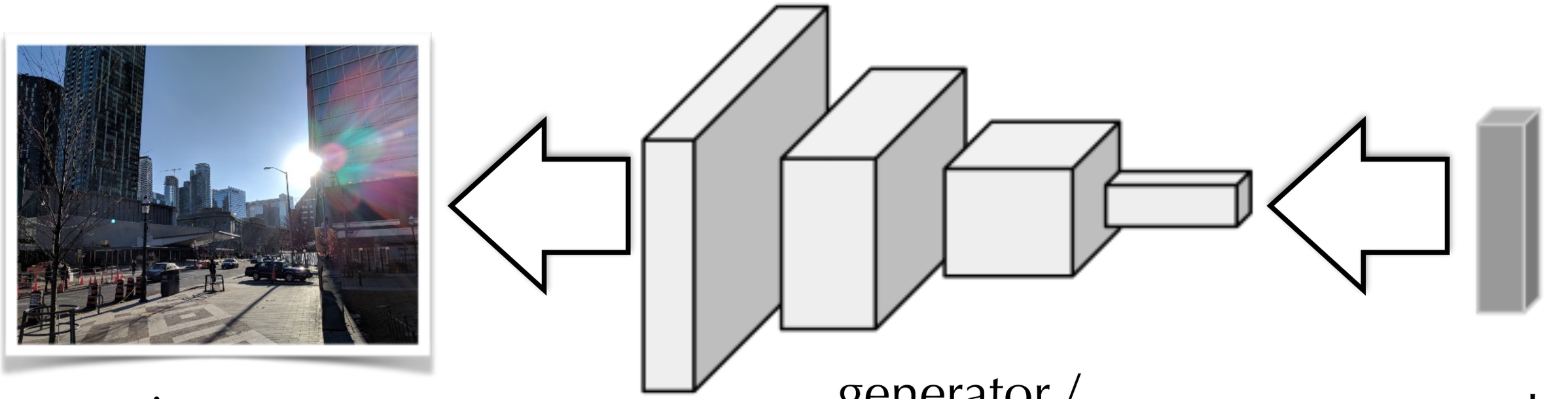
Generative Neural Networks



image

code

Generative Neural Networks



image

generator / decoder

code

Automatic Image Generation

[Nguyen et al., Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space, CVPR 2017]

Commercial Applications

(vue.ai, covered in Quartz)

[slide credit: Ian Goodfellow]

Image-to-Image Translation

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

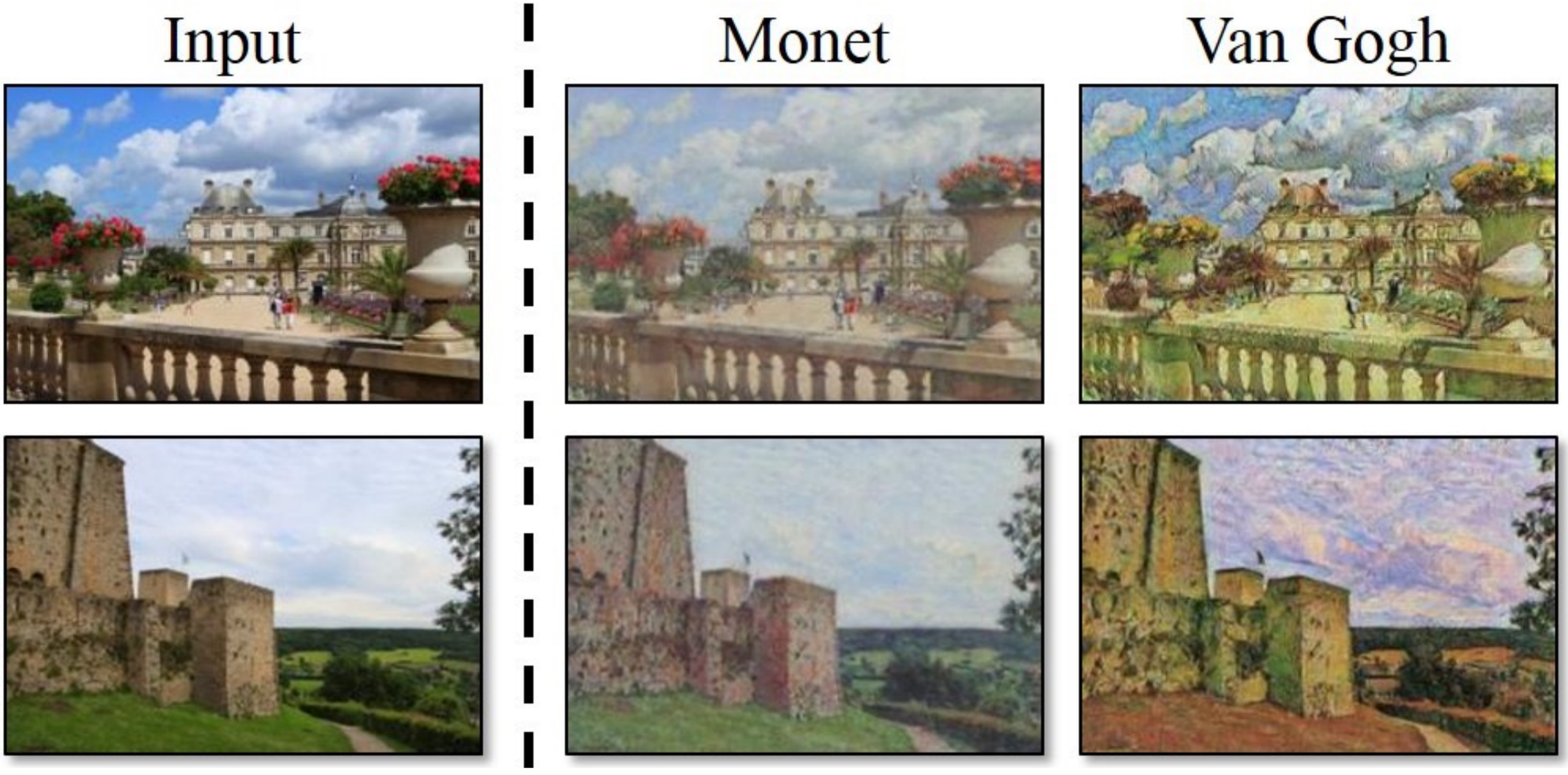
Image-to-Image Translation

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

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]

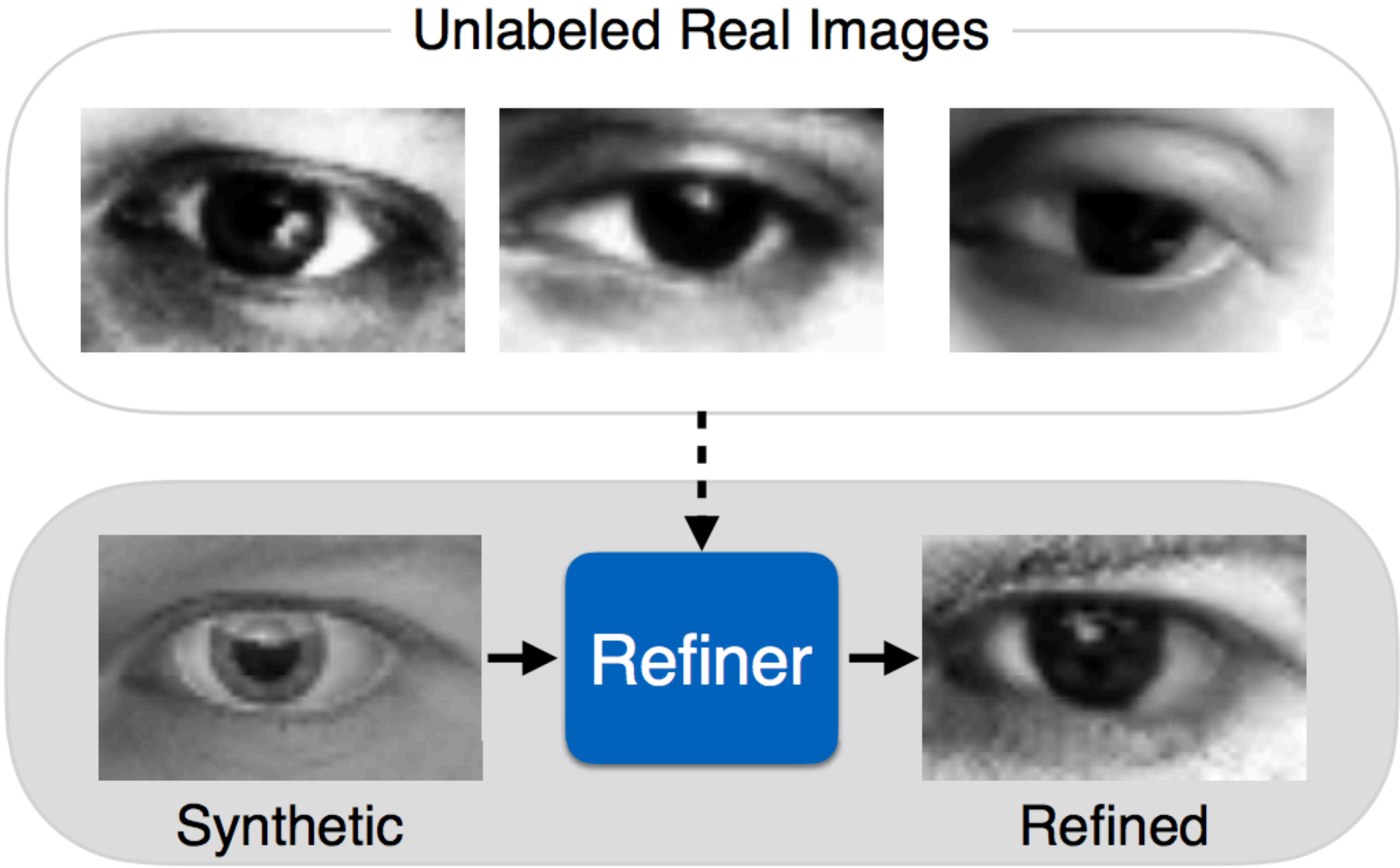
Training Data Generation

Input sunny image

AI-generated rainy image

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

Training Data Generation



[Shrivastava et al., Learning from Simulated and Unsupervised Images through Adversarial Training, CVPR 2017]

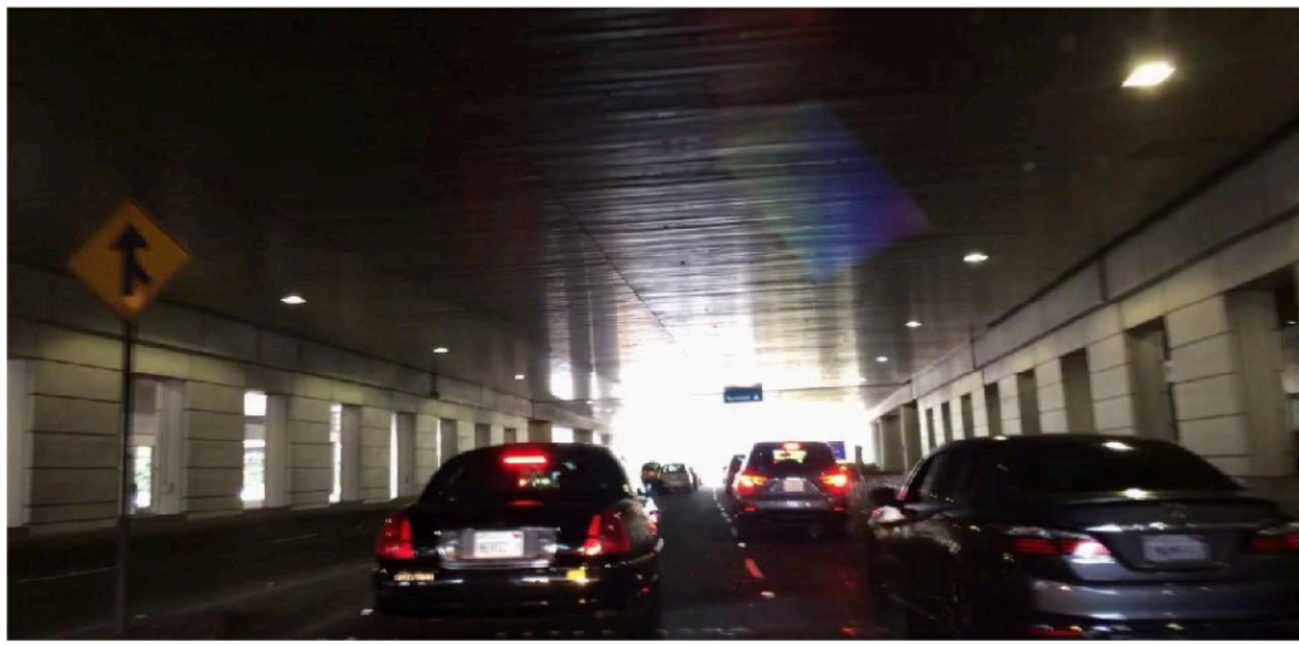
Domain Adaptation

Train

CityScapes (Germany)

Source domain:
Plenty of labelled data

Test



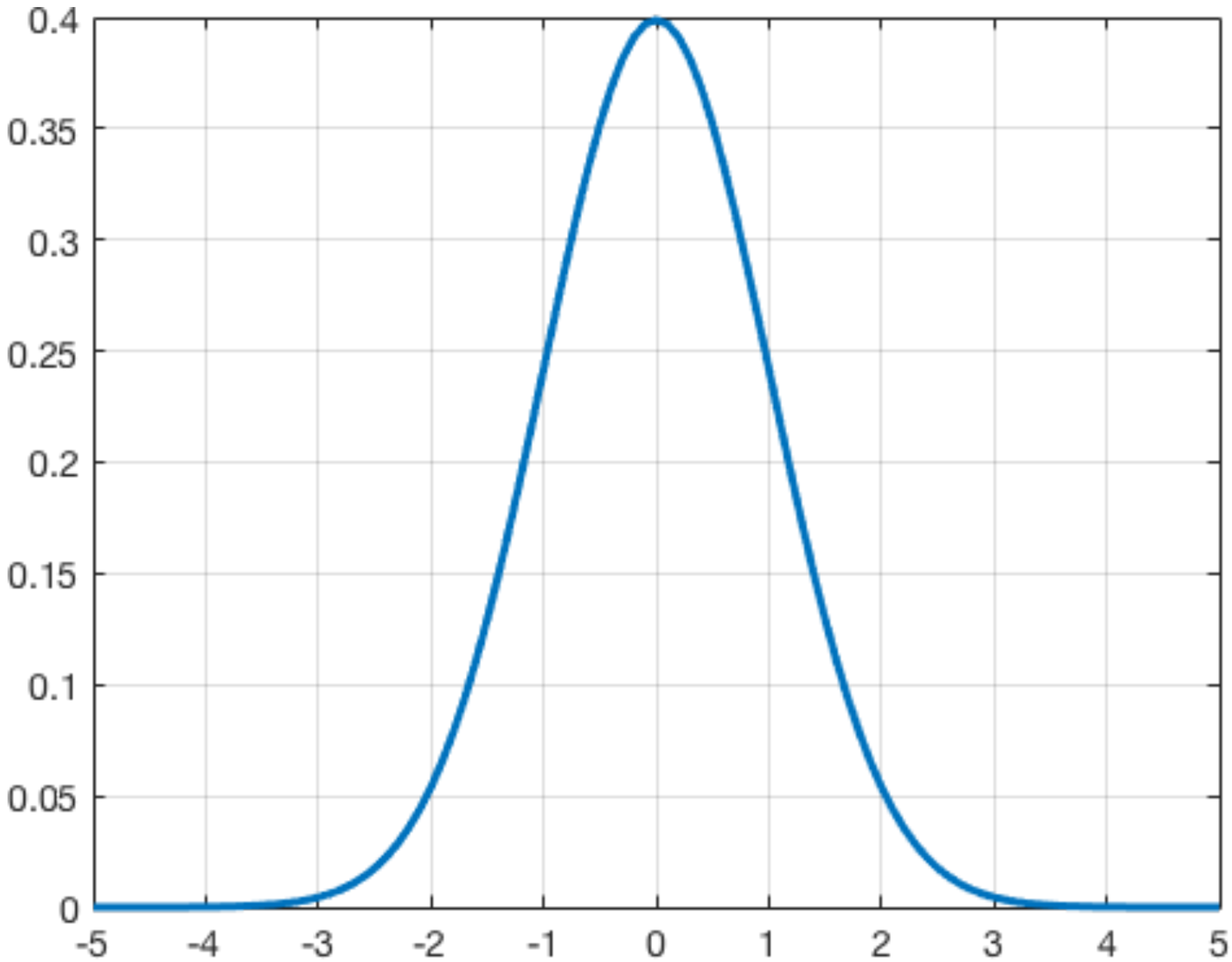
San Francisco

Target domain:
Little to no labelled data

slide credit: Judy Hoffman

Generation via Sampling

Example: Sampling from normal distribution



Generation via Sampling

Example: Sampling from normal distribution

Box-Muller Transform:

Generation via Sampling

Example: Sampling from normal distribution

Box-Muller Transform:

- Input: `U`, `V` drawn uniformly from (`0`,`1`)

Generation via Sampling

Example: Sampling from normal distribution

Box-Muller Transform:

- Input: $\mathtt{U}$, $\mathtt{V}$ drawn uniformly from $(0,1)$
- Output: Random numbers $\mathtt{X}$, $\mathtt{Y}$ drawn from $\mathcal{N}(0,1)$

Generation via Sampling

Example: Sampling from normal distribution

Box-Muller Transform:

- Input: $\mathtt{U}$, $\mathtt{V}$ drawn uniformly from $(0,1)$
- Output: Random numbers $\mathtt{X}$, $\mathtt{Y}$ drawn from $\mathcal{N}(0,1)$
- Algorithm:

$$X = \sqrt{-2\ln U}\cos(2\pi V)$$
$$Y = \sqrt{-2\ln U}\sin(2\pi V)$$

Sampling from Distribution of Images?

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

Generative Neural Networks

code

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

Generative Neural Networks

code

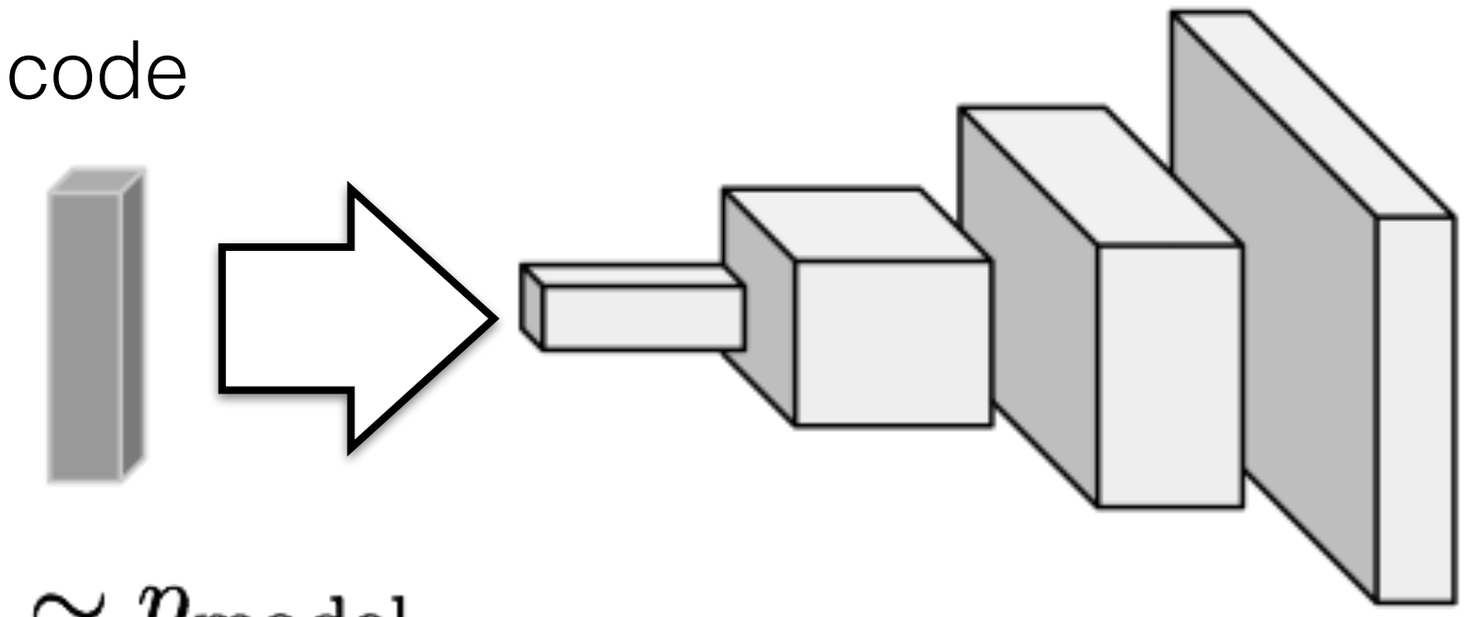
$$\mathbf{z} \sim p_{\text{model}}$$

normal distribution,
uniform distribution,
...

Generative Neural Networks

generator network G

code



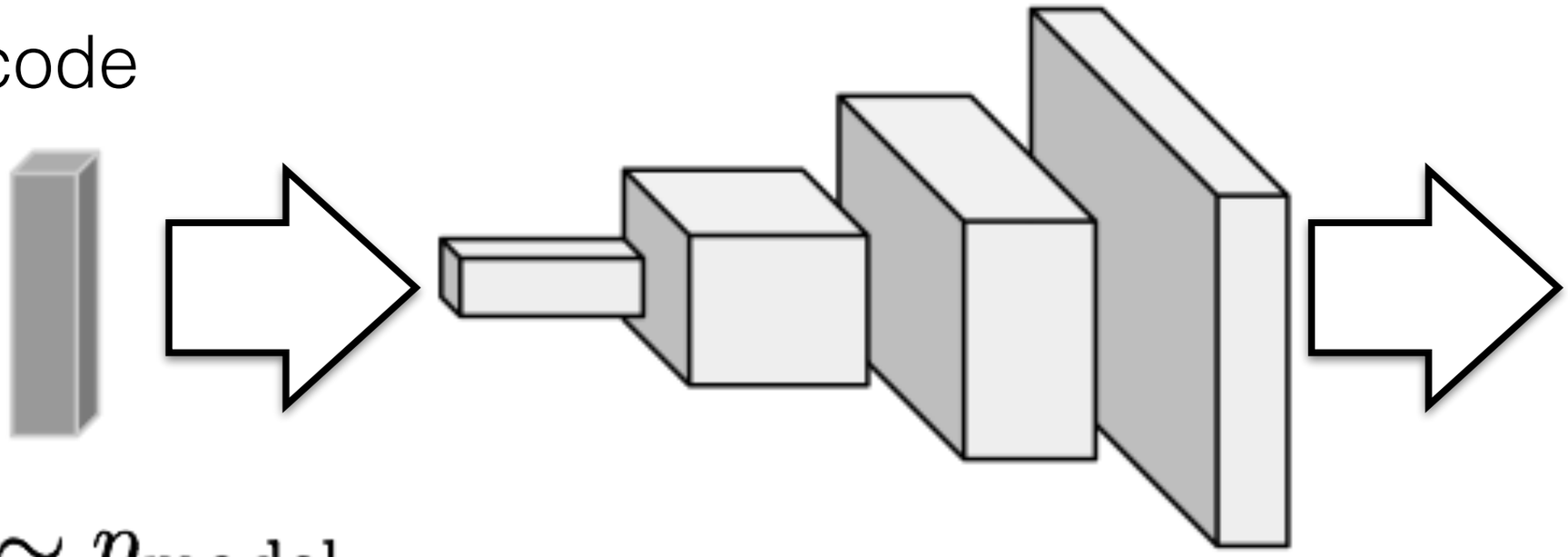
$\mathbf{z} \sim p_{\text{model}}$

normal distribution,
uniform distribution,
…

Generative Neural Networks

generator network G

code

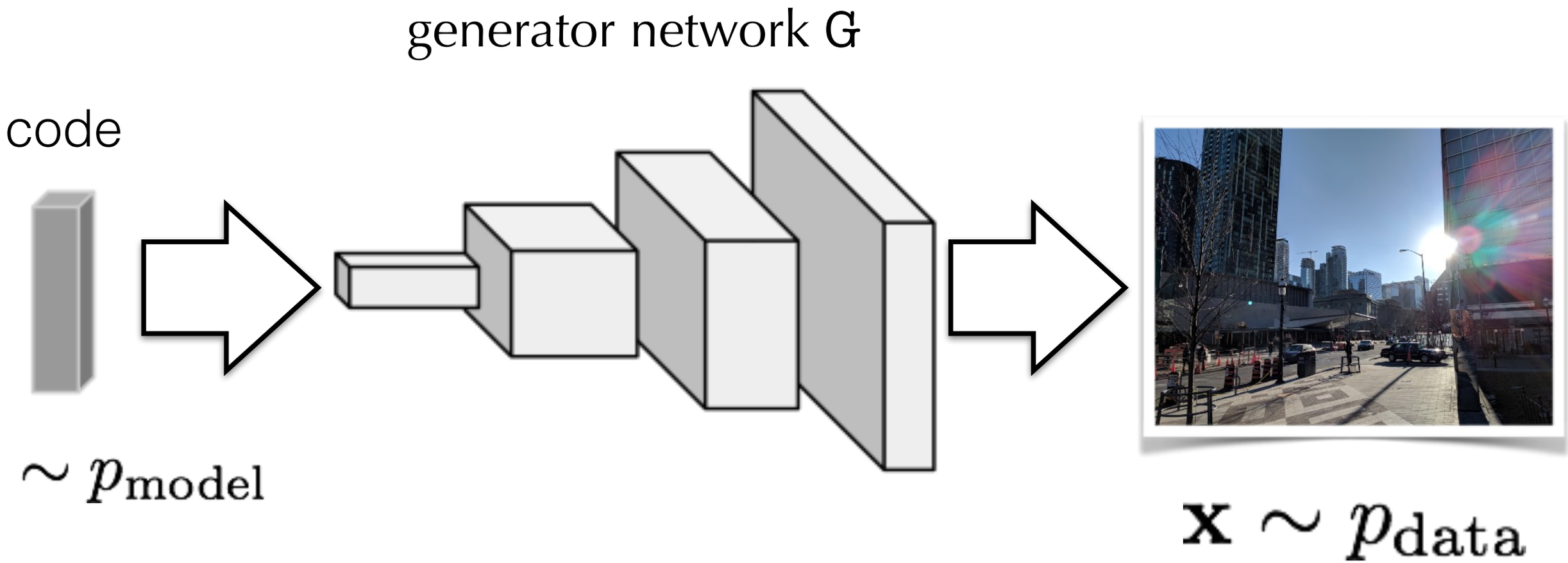


$\mathbf{z} \sim p_{\text{model}}$

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

normal distribution,
uniform distribution,
…

Generative Neural Networks



normal distribution,
uniform distribution,
…

How to learn a realistic distribution?

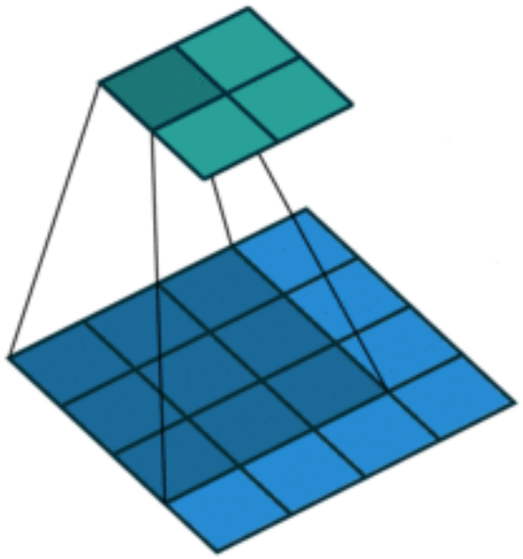
Today

- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GANs)
- Unsupervised Image Translation

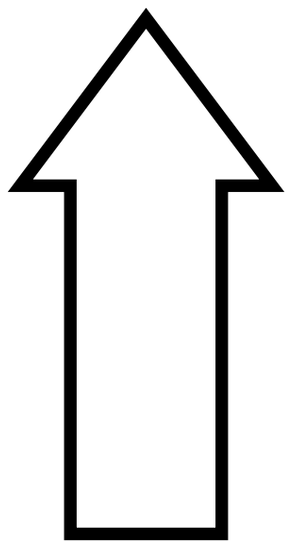
Today

- **Variational Autoencoders (VAE)**
- Generative Adversarial Networks (GANs)
- Unsupervised Image Translation

Recap: (Transposed) Convolutions



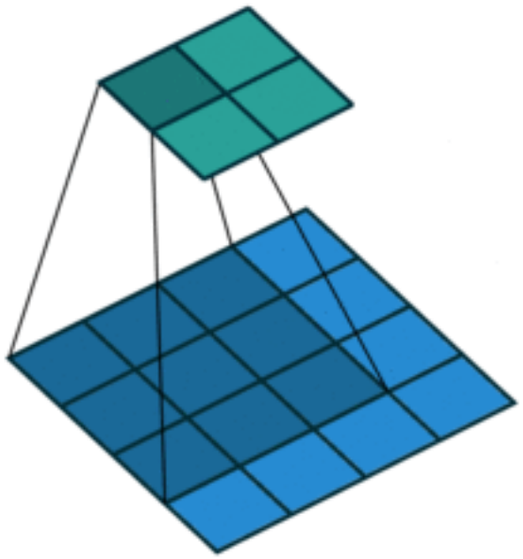
convolution



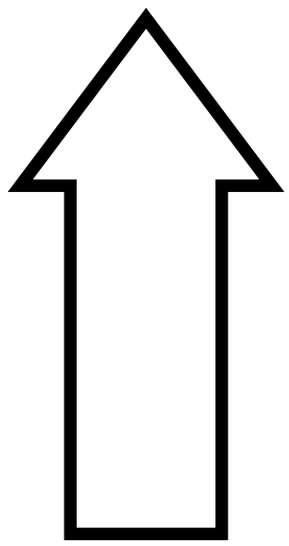
illustrations from https://github.com/vdumoulin/conv_arithmetic

slide credit: Laura Leal-Taixé

Recap: (Transposed) Convolutions



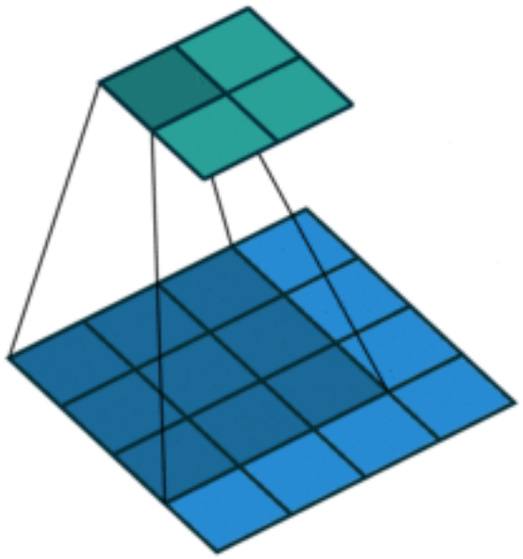
convolution



illustrations from https://github.com/vdumoulin/conv_arithmetic

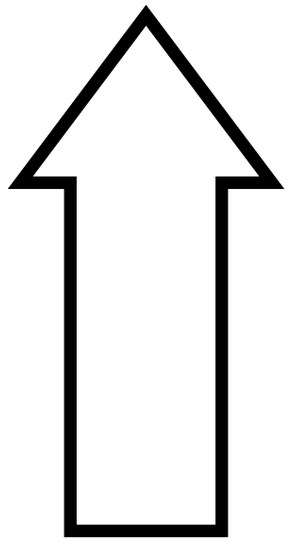
slide credit: Laura Leal-Taixé

Recap: (Transposed) Convolutions

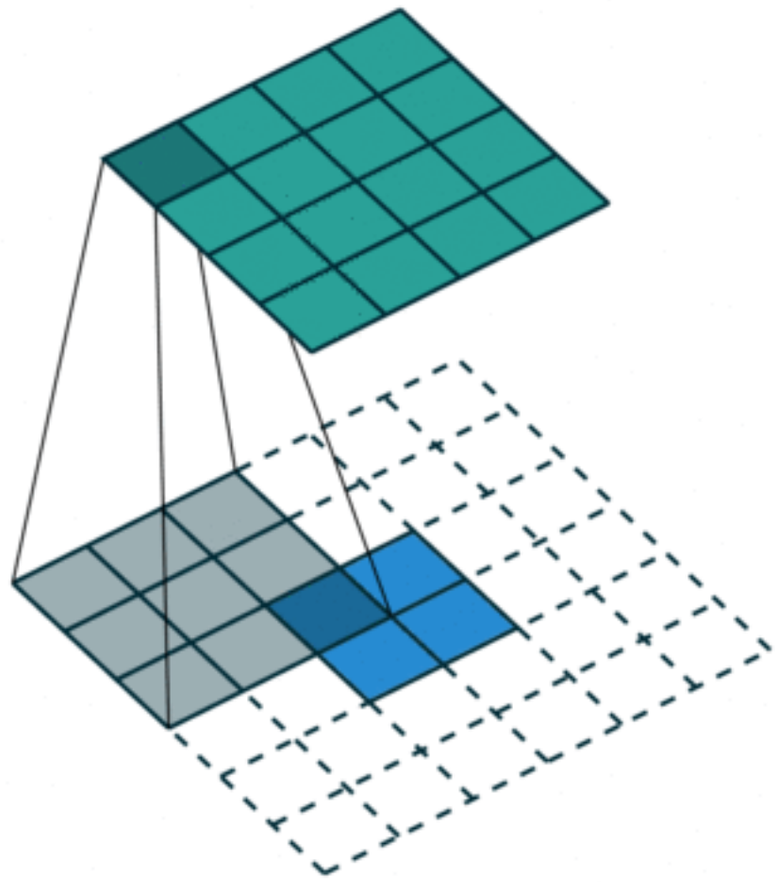


convolution

Output



Input



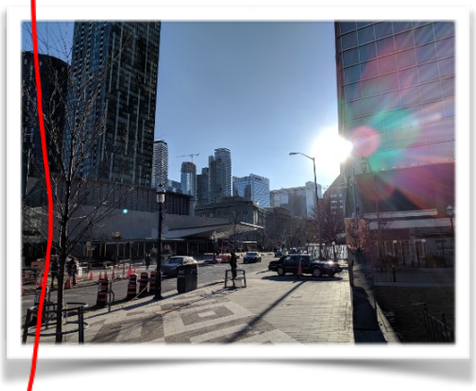
transposed convolution

illustrations from https://github.com/vdumoulin/conv_arithmetic

slide credit: Laura Leal-Taixé

Autoencoder

in practice/
at test time

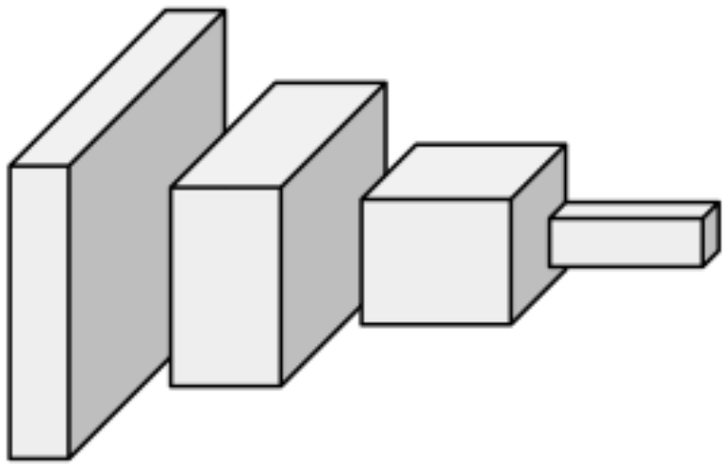


use rather
low-dimensional
space

Autoencoder

in practice / at test time

use rather low-dimensional space



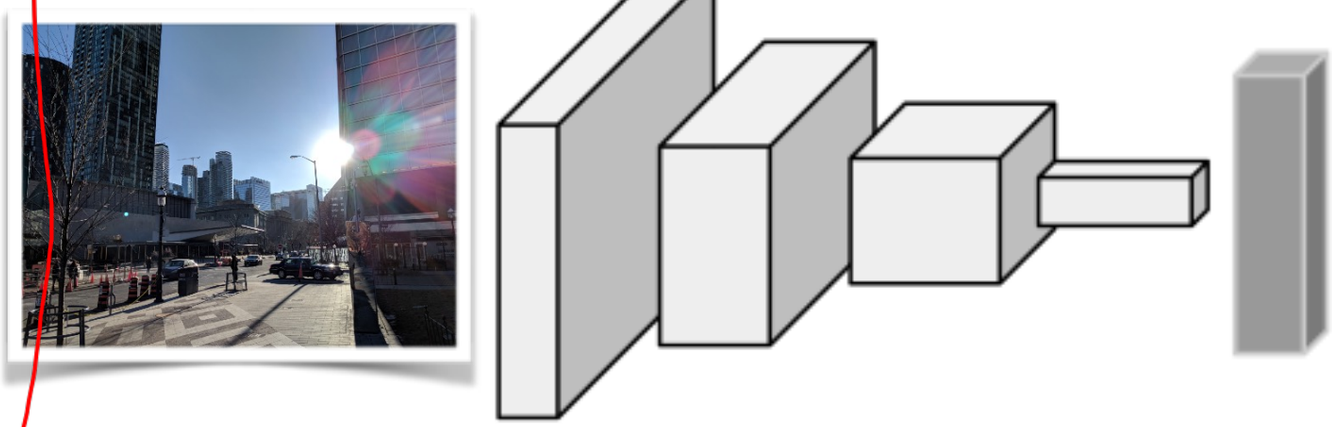
encoder
(convolutions)

Autoencoder

in practice / at test time

latent space

use rather low-dimensional space



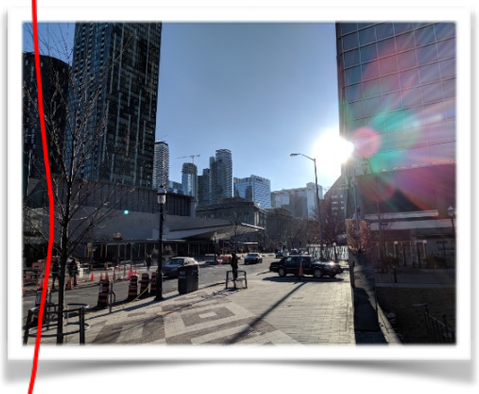
encoder (convolutions)

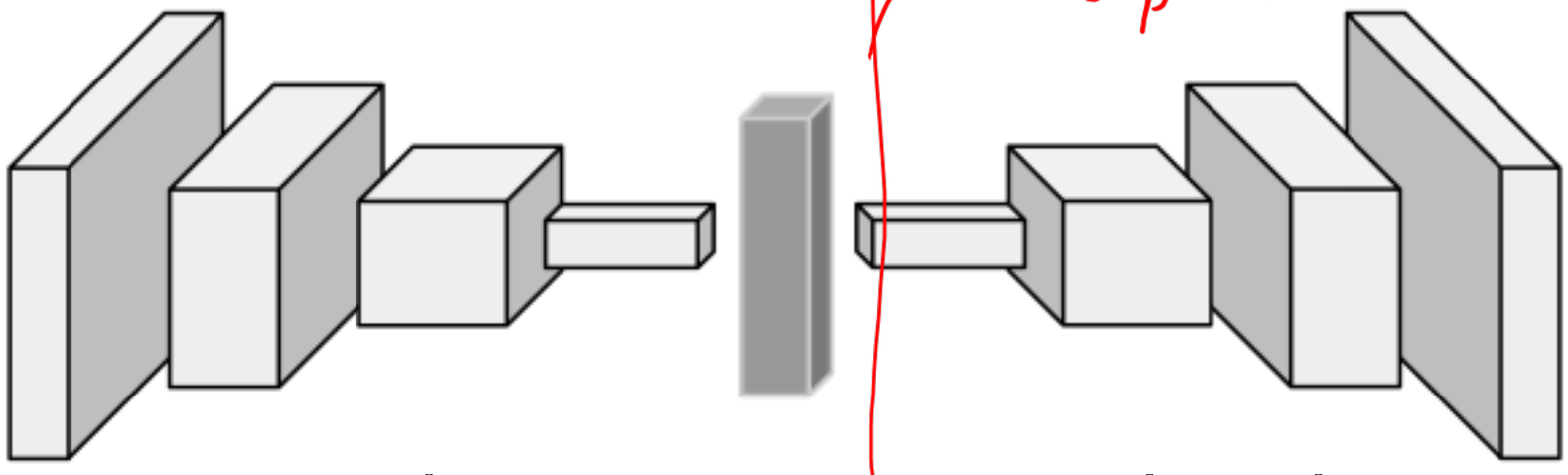
Autoencoder

in practice / at test time

latent space

use rather low-dimensional space





encoder (convolutions)

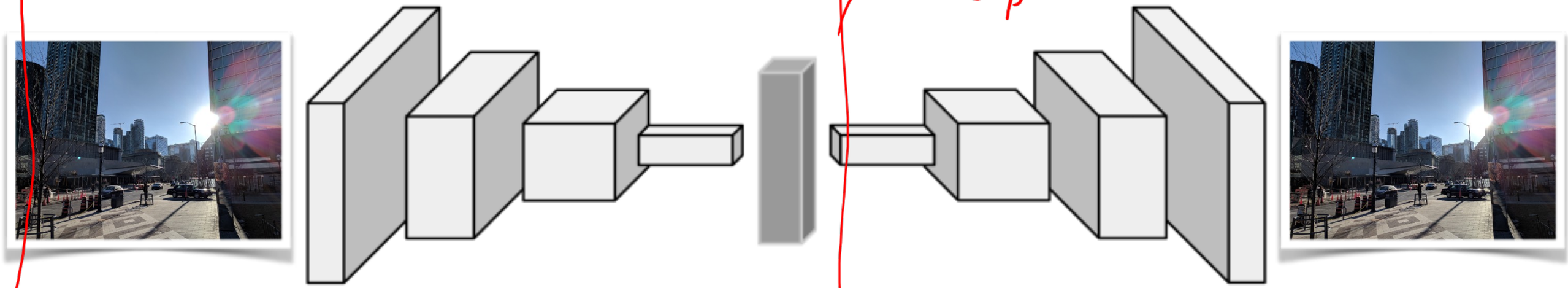
decoder (transp. convolutions)

Autoencoder

in practice / at test time

latent space

use rather low-dimensional space



encoder (convolutions)

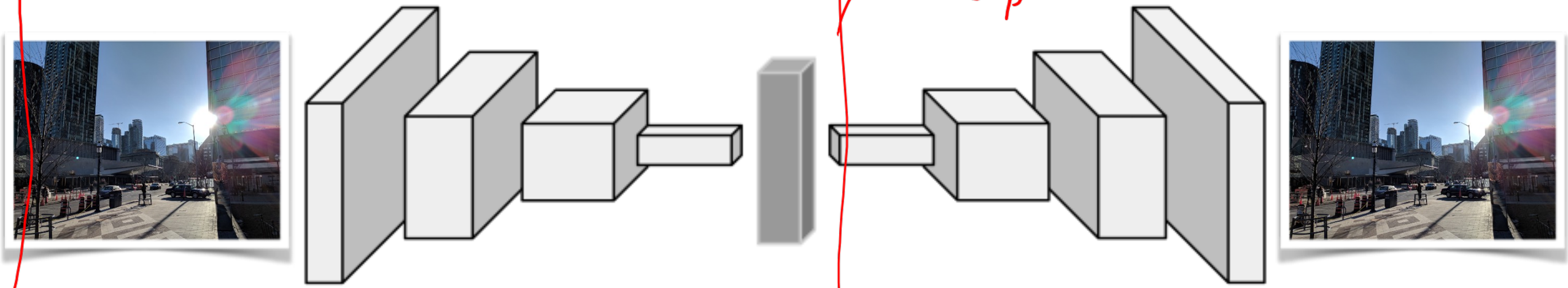
decoder (transp. convolutions)

Autoencoder

in practice / at test time

latent space

use rather low-dimensional space



encoder (convolutions)

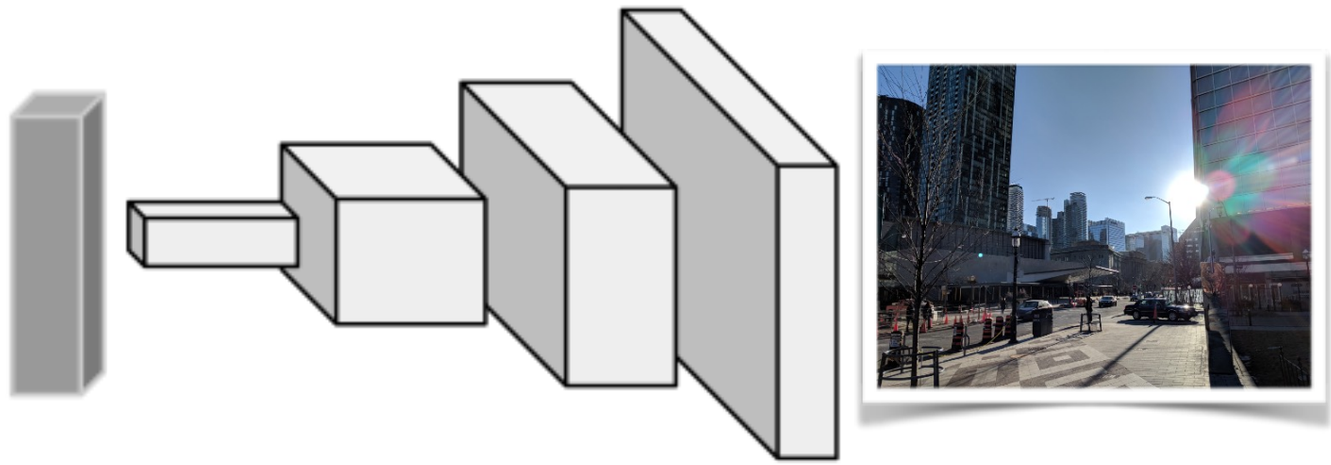
decoder (transp. convolutions)

reconstruction loss

Variational Autoencoder

sample point latent space

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



decoder
(transp. convolutions)

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

learn mapping from simple
to complex distribution

Theory

training example

Want to maximize

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

Theory

Want to maximize

training example

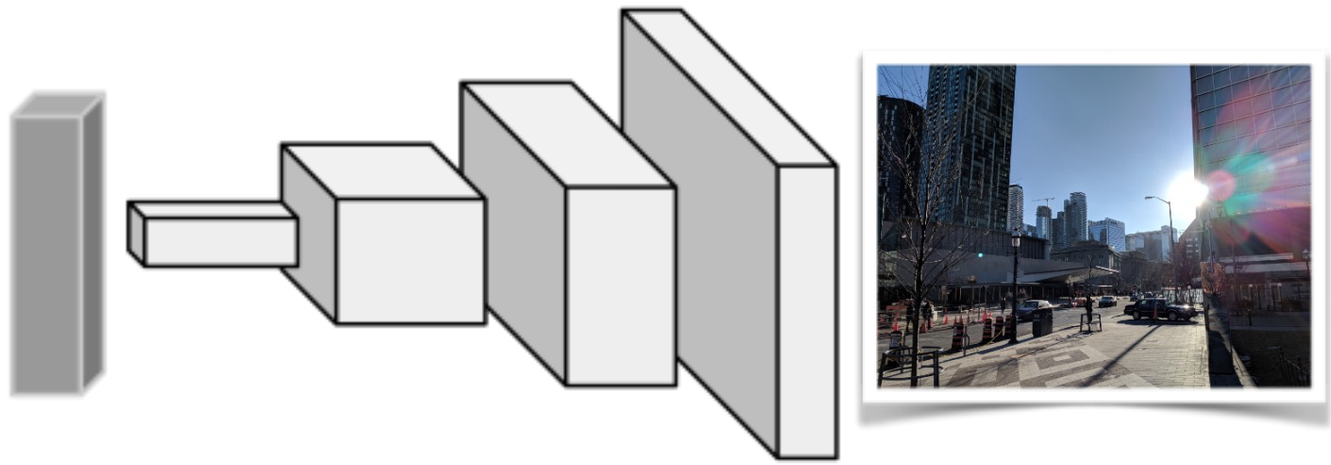
$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

Intractable due to high dimensionality of latent space

Variational Autoencoder

latent space

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



decoder
(transp. convolutions)

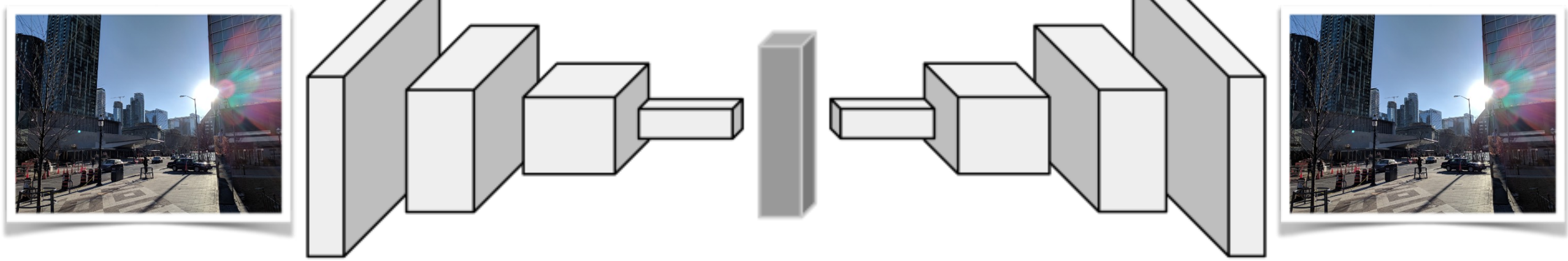
$$p(\mathbf{x}|\mathbf{z}) \qquad \mathbf{x} \sim p_{\text{data}}$$

learn mapping from simple
to complex distribution

Variational Autoencoder

latent space

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



encoder
(convolutions)

decoder
(transp. convolutions)

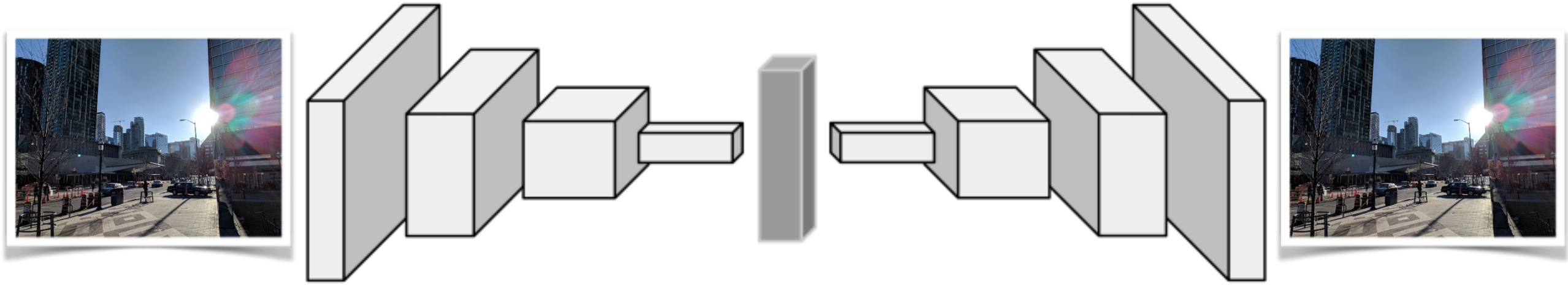
$$p(\mathbf{x}|\mathbf{z}) \qquad \mathbf{x} \sim p_{\text{data}}$$

learn mapping from simple
to complex distribution

Variational Autoencoder

latent space

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



encoder
(convolutions)

decoder
(transp. convolutions)

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

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

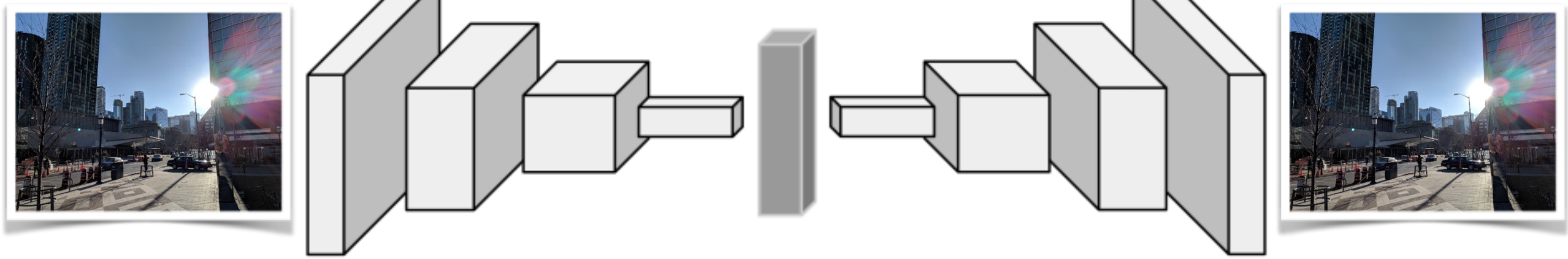
$\mathbf{x} \sim p_{\text{data}}$

learn mapping from simple
to complex distribution

Variational Autoencoder

latent space

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



encoder
(convolutions)

decoder
(transp. convolutions)

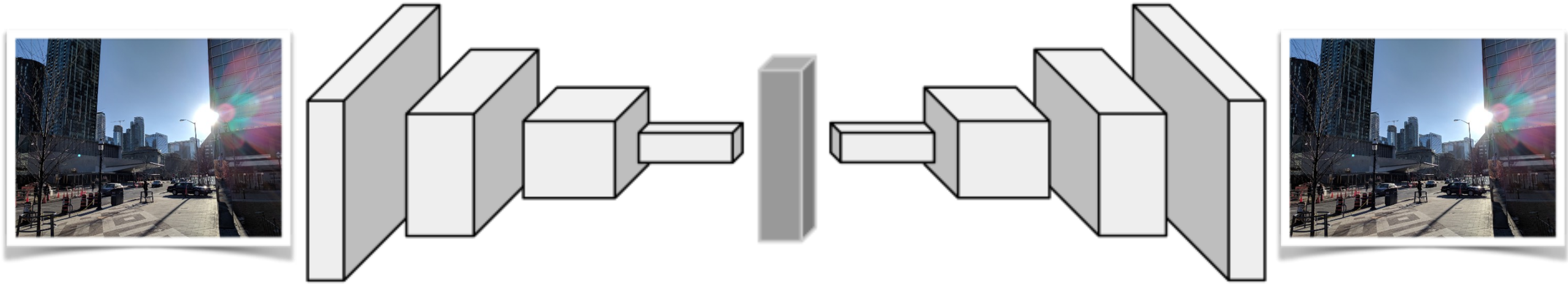
$\mathbf{x} \sim p_{\text{data}}$ $q(\mathbf{z}|\mathbf{x})$ $p(\mathbf{x}|\mathbf{z})$ $\mathbf{x} \sim p_{\text{data}}$

learn mapping from simple
to complex distribution

Variational Autoencoder

latent space

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



encoder
(convolutions)

decoder
(transp. convolutions)

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

learn structure of
latent space

learn mapping from simple
to complex distribution

Theory

Use encoder to generate probable samples, simplifying

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

to

$$p(\mathbf{x}) = E_{\mathbf{z}\sim q}p(\mathbf{x}|\mathbf{z})d\mathbf{z}$$

Theory

Use encoder to generate probable samples, simplifying

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

to

$$p(\mathbf{x}) = E_{\mathbf{z}\sim q}p(\mathbf{x}|\mathbf{z})d\mathbf{z}$$

Problem: Learned distribution $\mathbf{q}$ might not follow $\mathbf{p}_{\text{model}}$

KL Divergence

Kullback-Leibler divergence: ``distance'' between probability distributions

$$D_{\mathrm{KL}}(P||Q) = \int P(x) \log\left(\frac{P(x)}{Q(x)}\right) dx \quad \geq 0$$

$= 0 \Leftrightarrow P(x) = Q(x)$ $\forall x$

$D_{KL}(P||Q) \neq D_{KL}(Q||P)$

KL Divergence

Kullback-Leibler divergence: ``distance'' between probability distributions

$$D_{\mathrm{KL}}(P||Q) = \int P(x) \log\left(\frac{P(x)}{Q(x)}\right) dx \geq 0$$

$= 0 \Leftrightarrow P(x) = Q(x) \quad \forall x$

Equivalent to (when sampling):

$$D_{\mathrm{KL}}(P||Q) = E_{x\sim P}[\log(P(x)) - \log(Q(x))]$$

$D_{KL}(P||Q) \neq D_{KL}(Q||P)$

KL Divergence

Kullback-Leibler divergence: ``distance'' between probability distributions

$$D_{\mathrm{KL}}(P||Q) = \int P(x) \log\left(\frac{P(x)}{Q(x)}\right) dx \geq 0$$

$= 0 \Leftrightarrow P(x) = Q(x) \quad \forall x$

Equivalent to (when sampling):

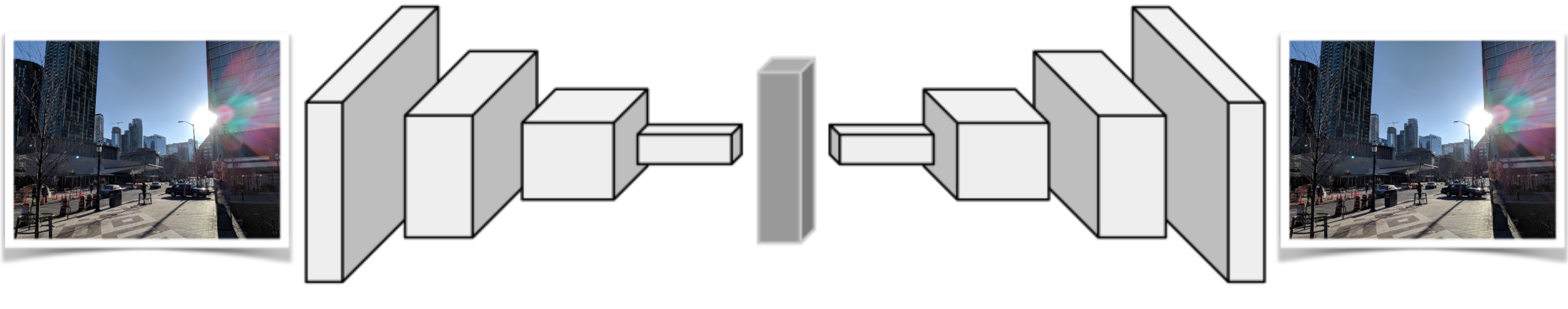
$$D_{\mathrm{KL}}(P||Q) = E_{x\sim P}[\log(P(x)) - \log(Q(x))]$$

Notice that KL divergence is asymmetric

$D_{KL}(P||Q) \neq D_{KL}(Q||P)$

Theory

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

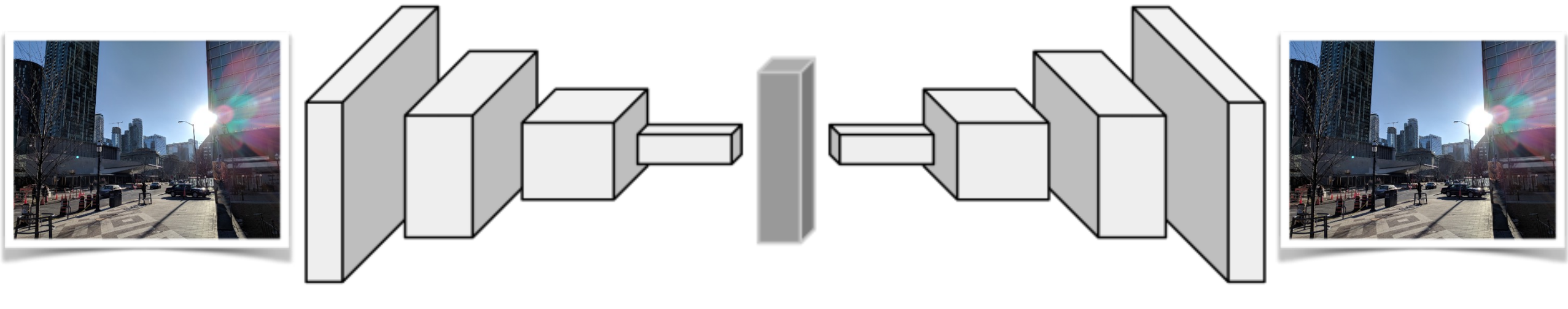


encoder
$q(\mathbf{z}|\mathbf{x})$

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

Theory

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



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

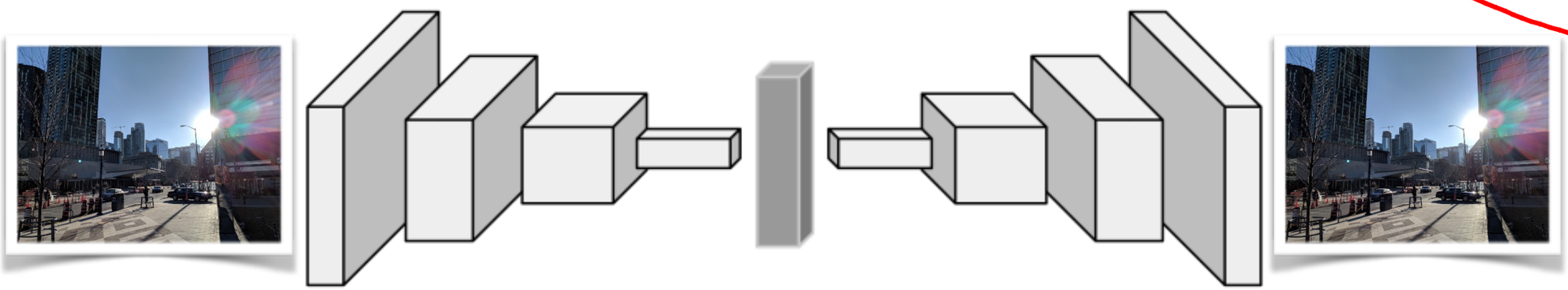
decoder
$p(\mathbf{x}|\mathbf{z})$

Want to ensure that $\mathtt{q(z|x)}$ and $\mathtt{p(z|x)}$ are the same:

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

Theory

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



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

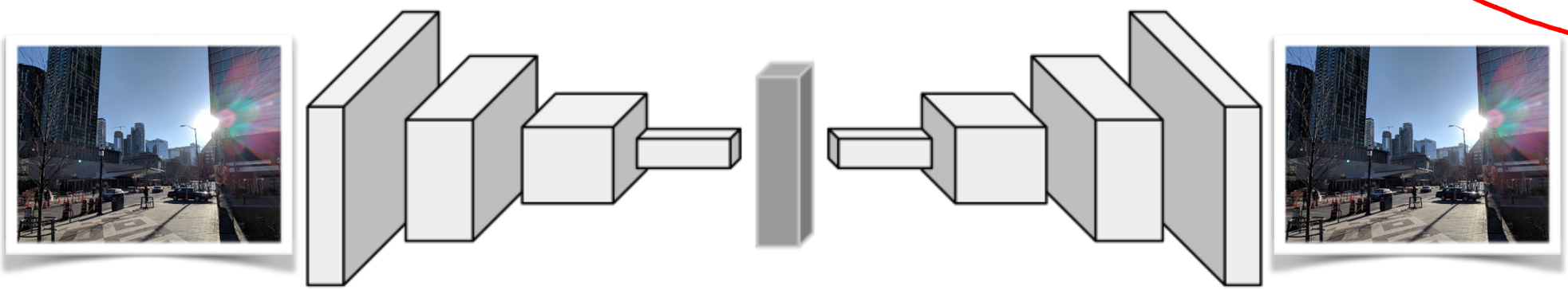
decoder
$p(\mathbf{x}|\mathbf{z})$

Use Bayes rule to include p(**x**) and p(**x**|**z**):

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

Theory

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



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

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

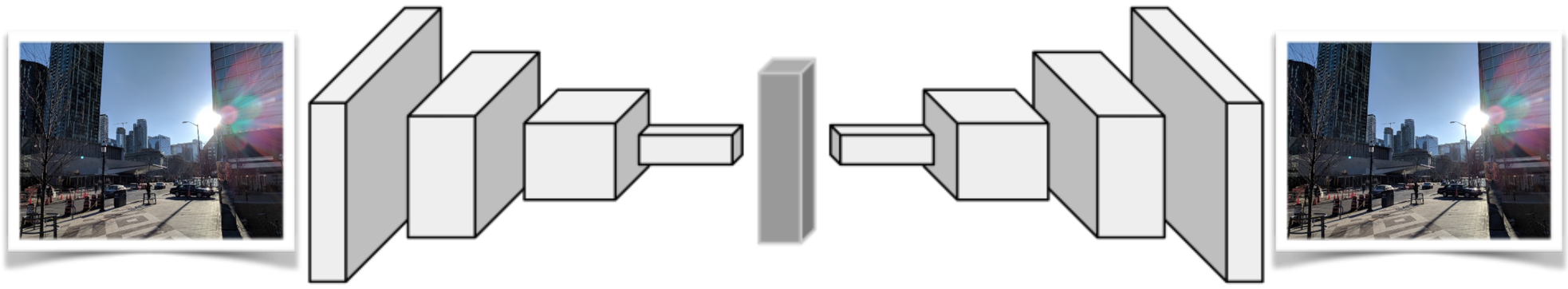
Use Bayes rule to include p(**x**) and p(**x**|**z**):

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

$$D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log\left(\frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}\right)]$$

Theory

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



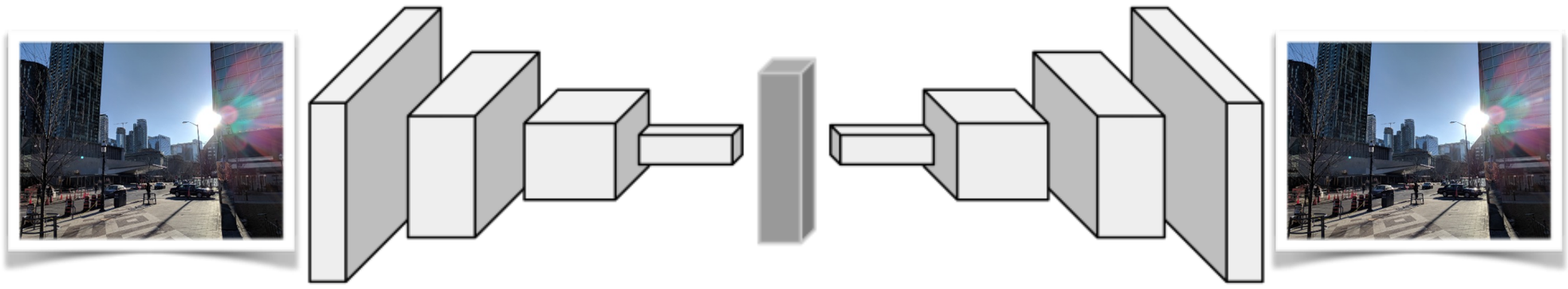
encoder
$q(\mathbf{z}|\mathbf{x})$

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

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

Theory

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



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

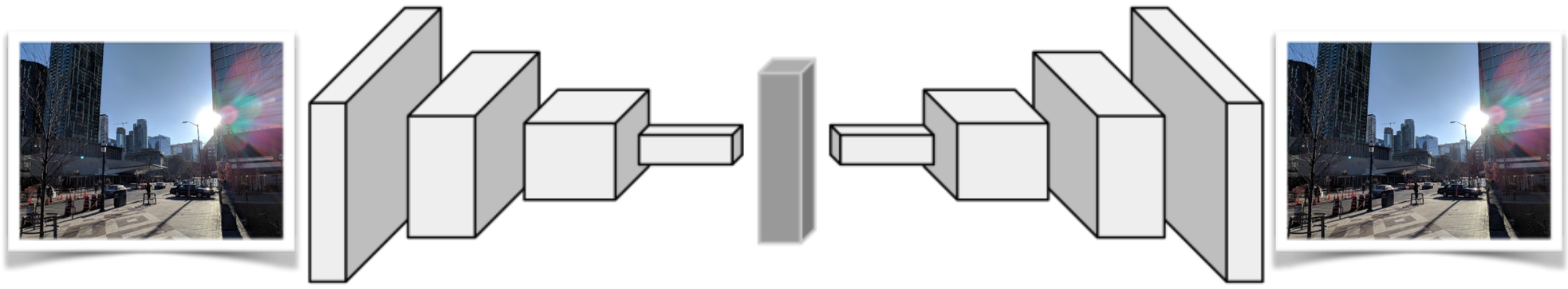
decoder
$p(\mathbf{x}|\mathbf{z})$

$$D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log\left(\frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}\right)]$$

$$D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log\left(p(\mathbf{x}|\mathbf{z})\right) - \log(p(\mathbf{z})) + \log(p(\mathbf{x})))]$$

Theory

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



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

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

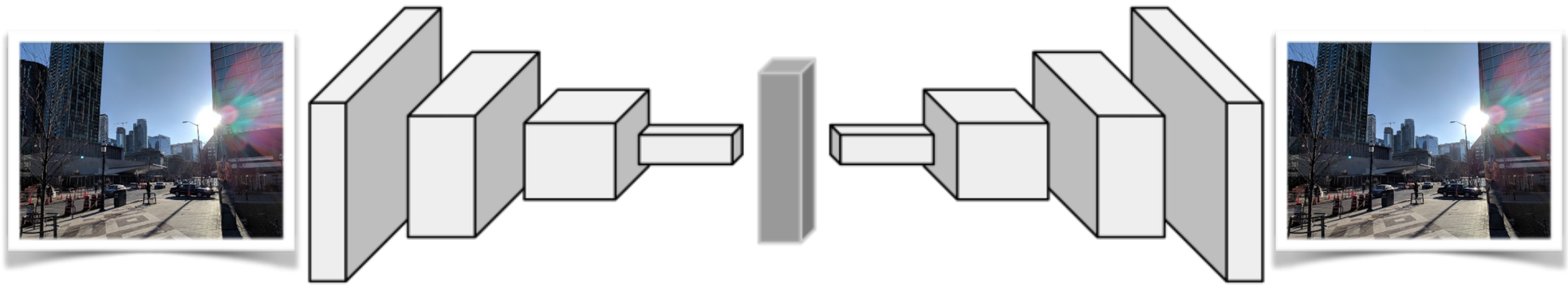
$$D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log\left(\frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}\right)]$$

$$D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log\left(p(\mathbf{x}|\mathbf{z})\right) - \log(p(\mathbf{z})) + \log(p(\mathbf{x}))]$$

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

Theory

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



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

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

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

≥ 0

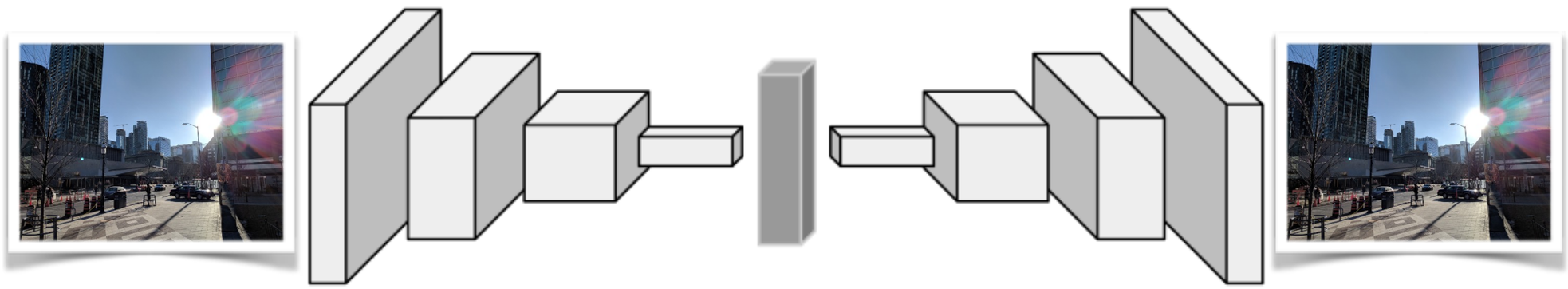
no idea

decoder

$\checkmark$

Theory

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



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

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

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

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

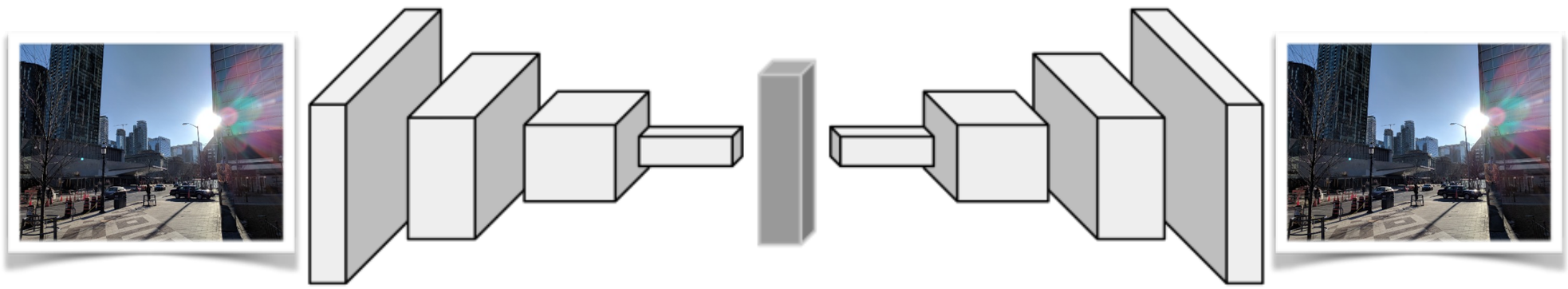
≥ 0

no idea

decoder

Theory

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



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

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

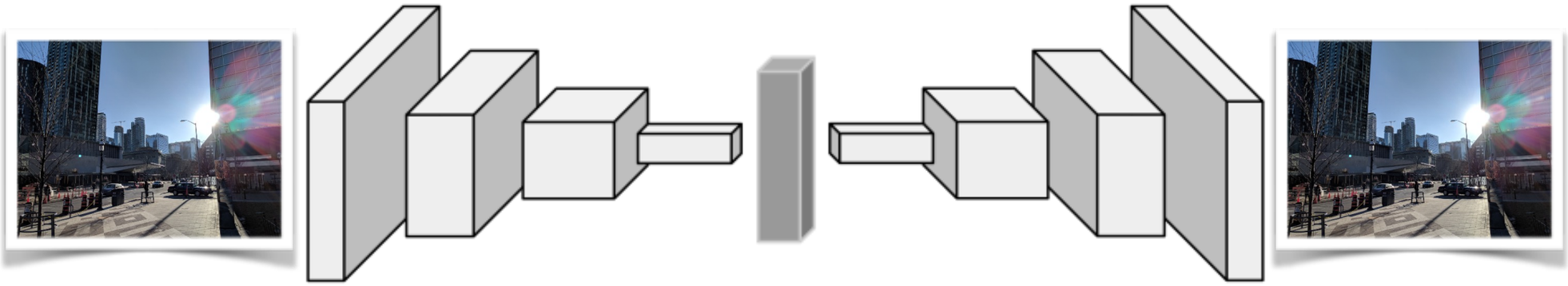
$$D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = -E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] + E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log(p(\mathbf{z}))] + \log(p(\mathbf{x}))$$

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

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

Objective Function

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



encoder

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

decoder

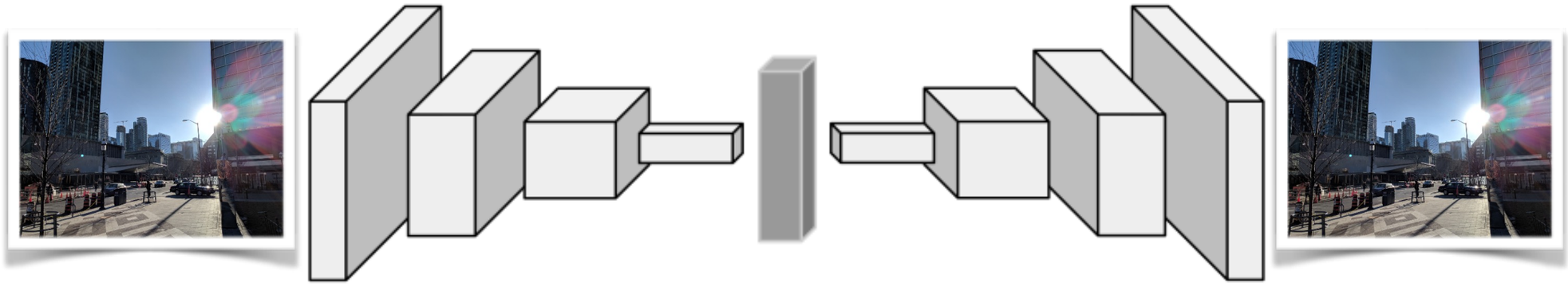
$p(\mathbf{x}|\mathbf{z})$

Maximize

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

Objective Function

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



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

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

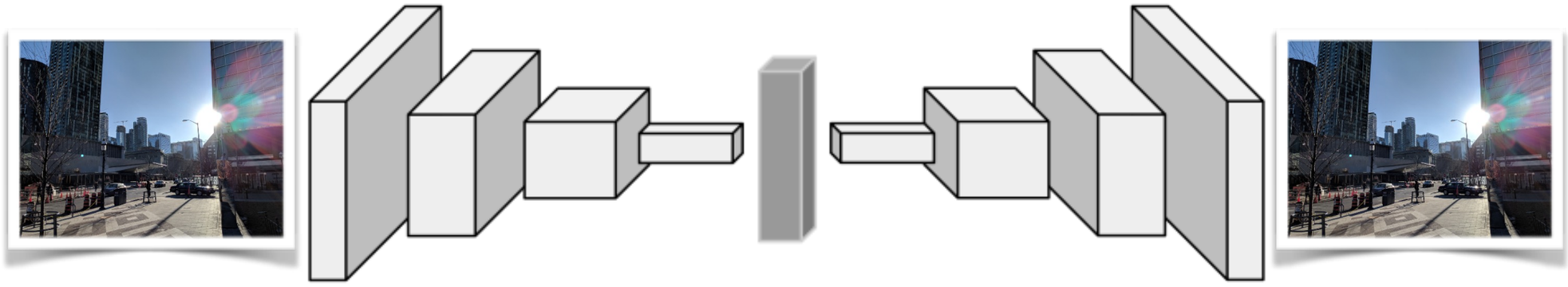
Maximize

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

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

Objective Function

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



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

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

Maximize

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

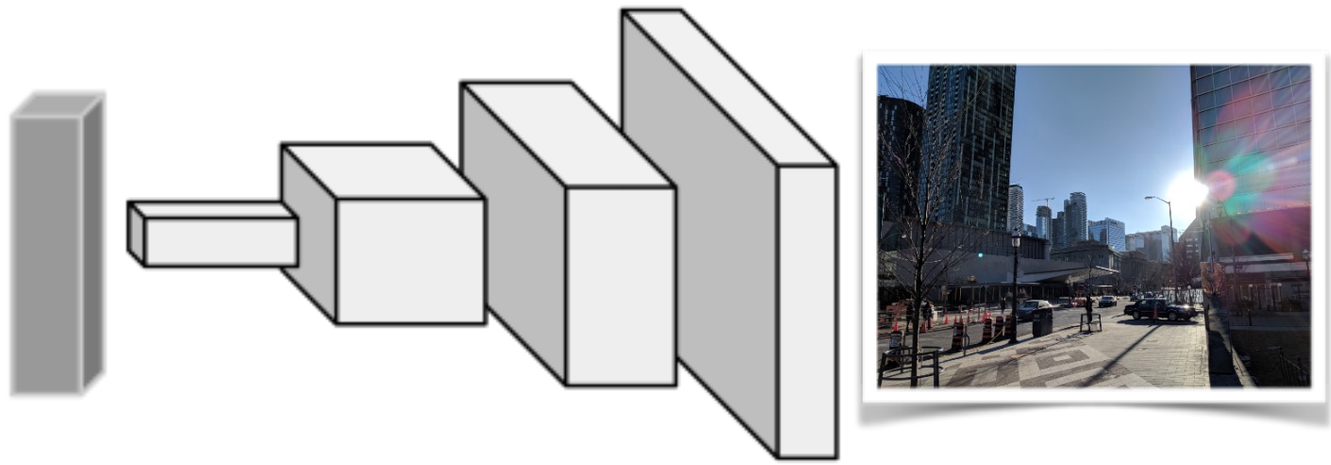
reconstruction ``loss'',
e.g., L2 norm (normal distribution)

similarity between learned
and sampling distribution

At Test Time

sample point latent space

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



decoder
(transp. convolutions)

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

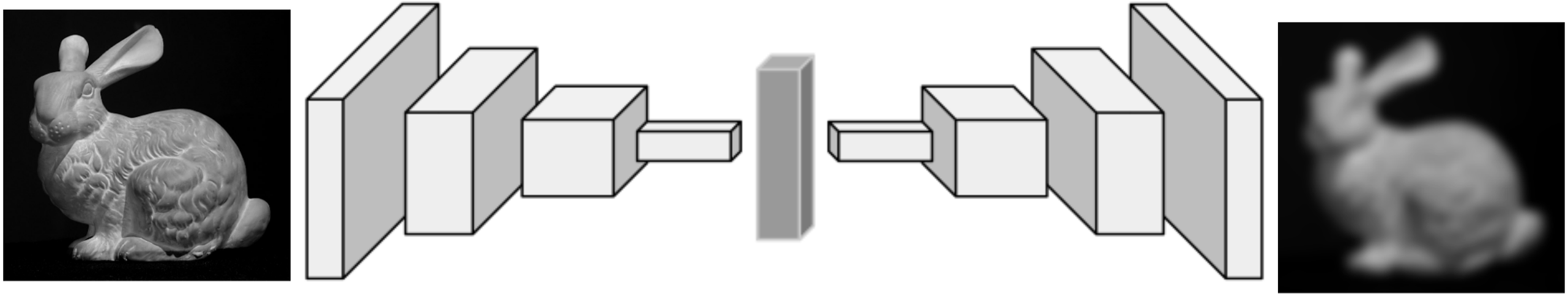
use decoder to ``translate''
to image

Today

- Variational Autoencoders (VAE)
- **Generative Adversarial Networks (GANs)**
- Unsupervised Image Translation

Variational Autoencoder

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



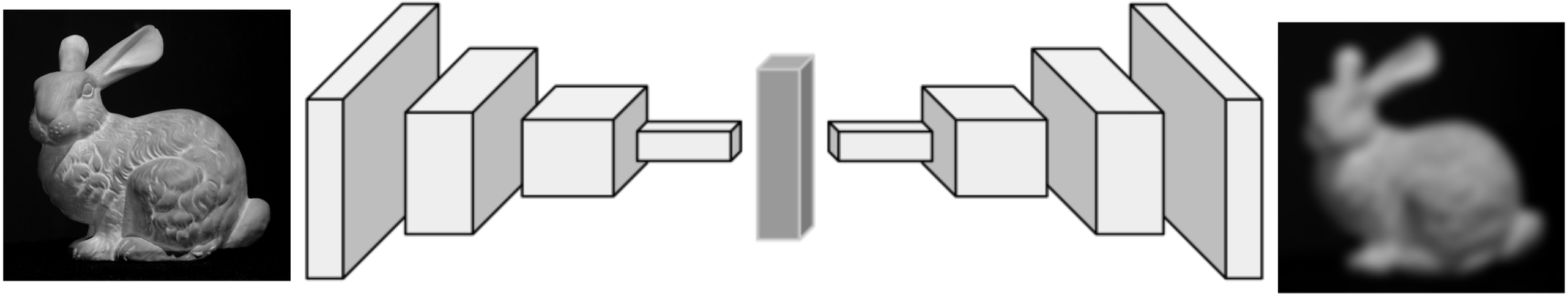
encoder $q(\mathbf{z}|\mathbf{x})$ decoder $p(\mathbf{x}|\mathbf{z})$

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

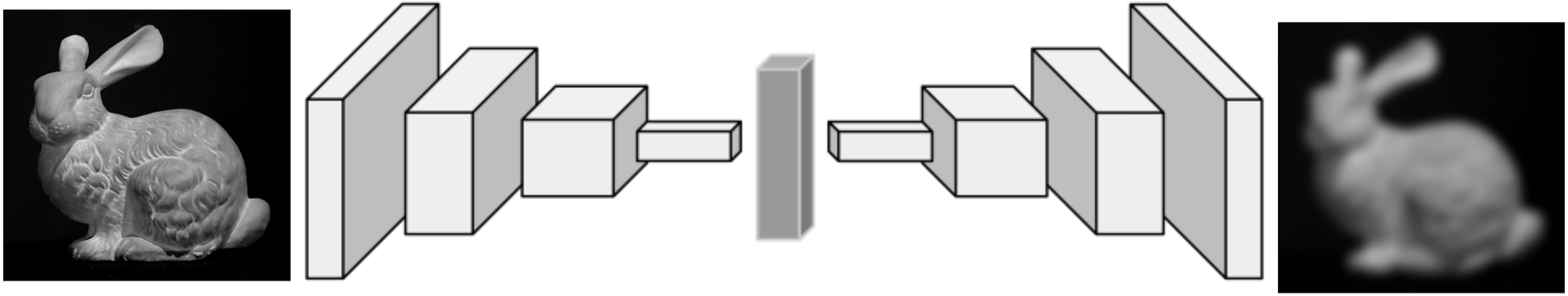
Maximize

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

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

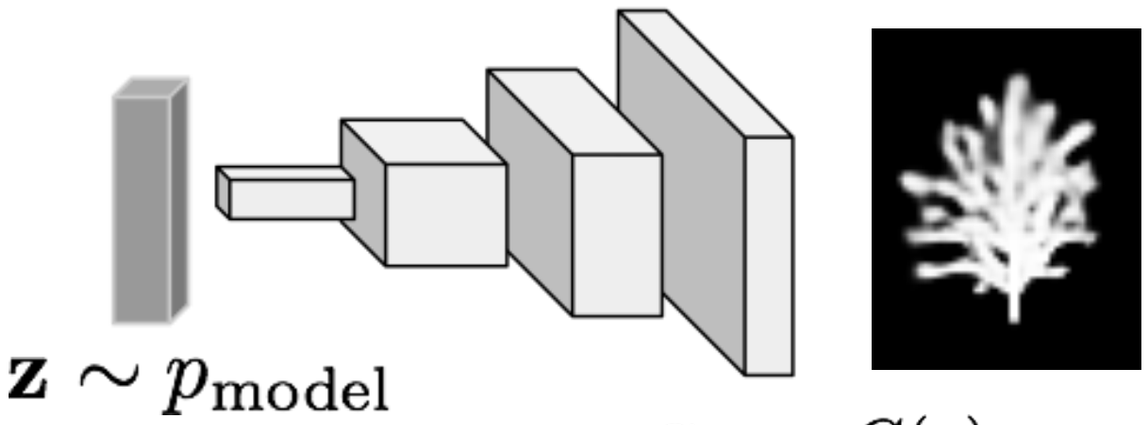
Generative Adversarial Networks (GANs)

Motivation: Learn loss function

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

Generative Adversarial Networks (GANs)

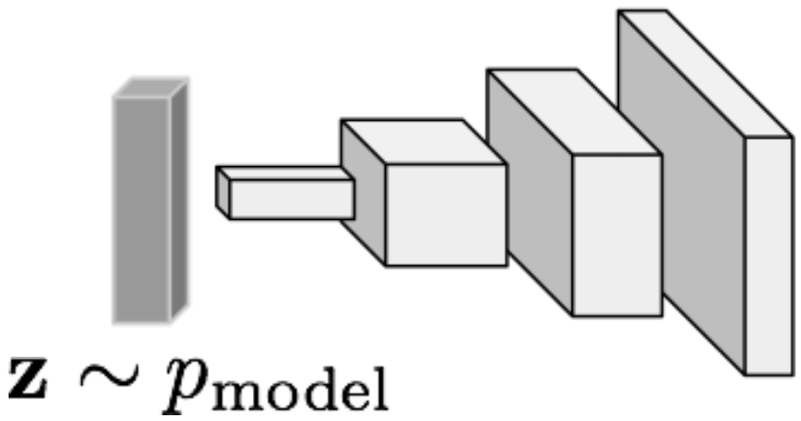
Motivation: Learn loss function



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

Generative Adversarial Networks (GANs)

Motivation: Learn loss function



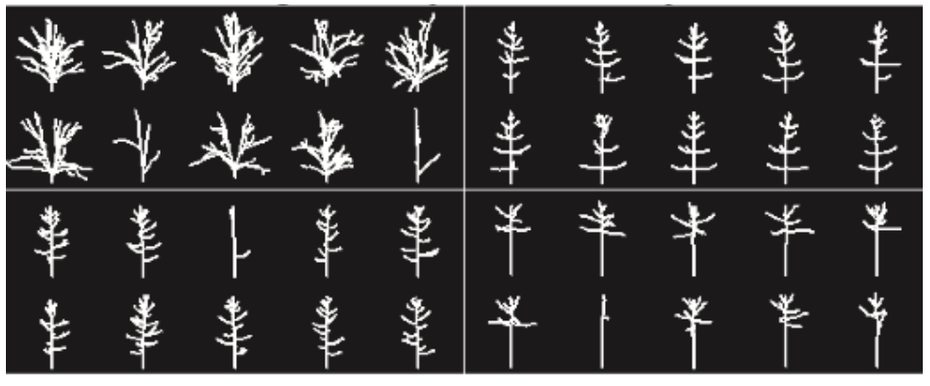
$\mathbf{z} \sim p_{\text{model}}$

generator G

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

real data

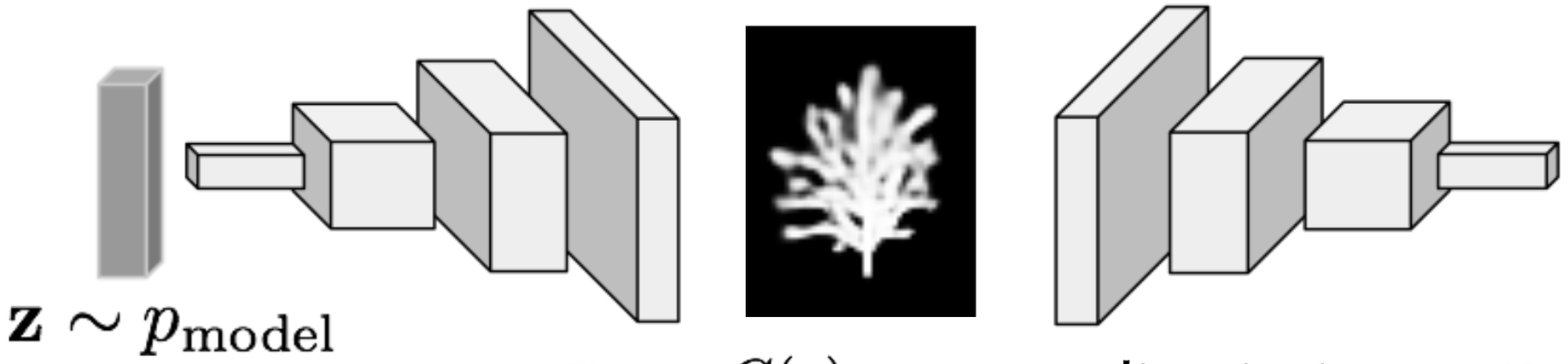
$\mathbf{x} \sim p_{\text{data}}$



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

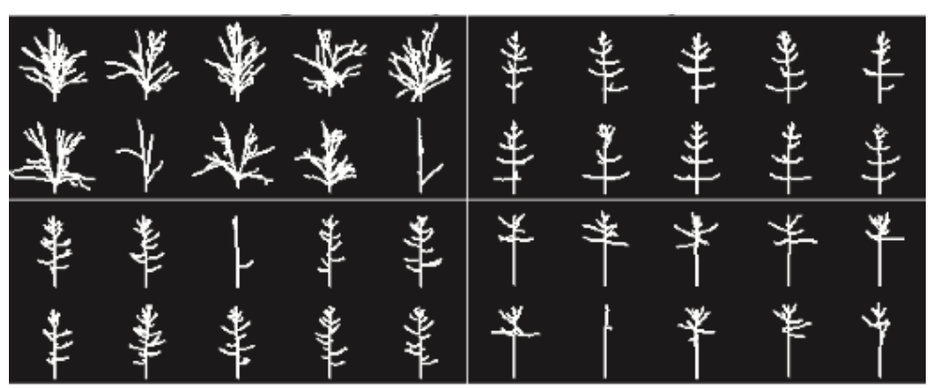
Generative Adversarial Networks (GANs)

Motivation: Learn loss function



real data

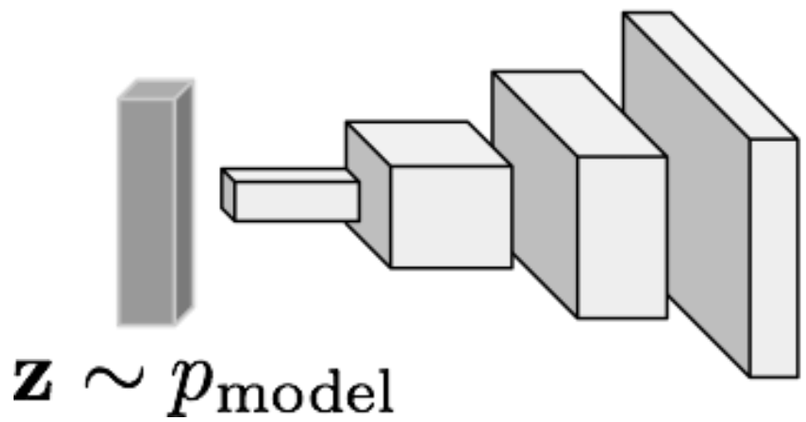
$\mathbf{x} \sim p_{\text{data}}$



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

Generative Adversarial Networks (GANs)

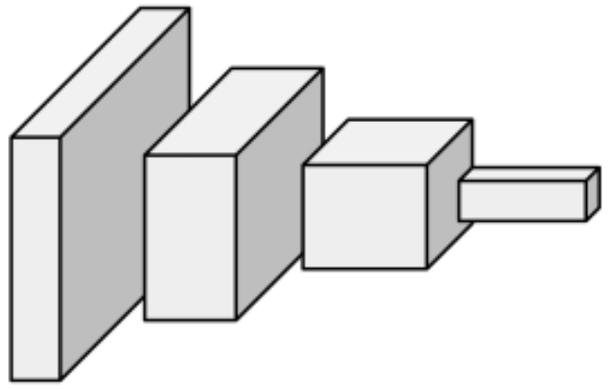
Motivation: Learn loss function



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

generator G

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

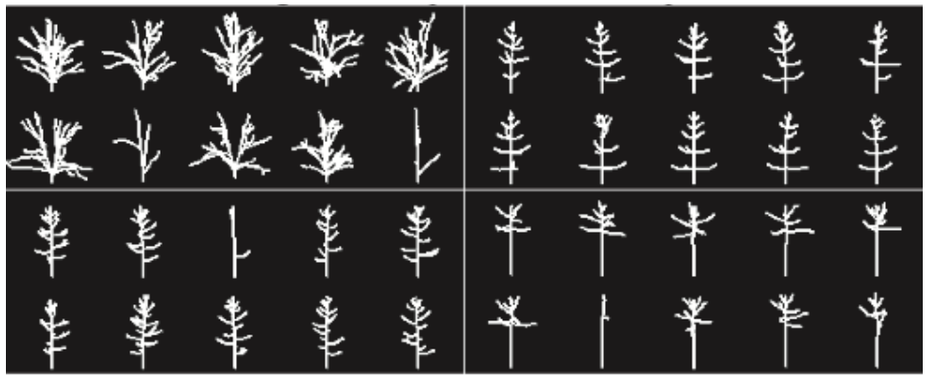


$D(G(\mathbf{z})) \in [0, 1]$

discriminator D

real data

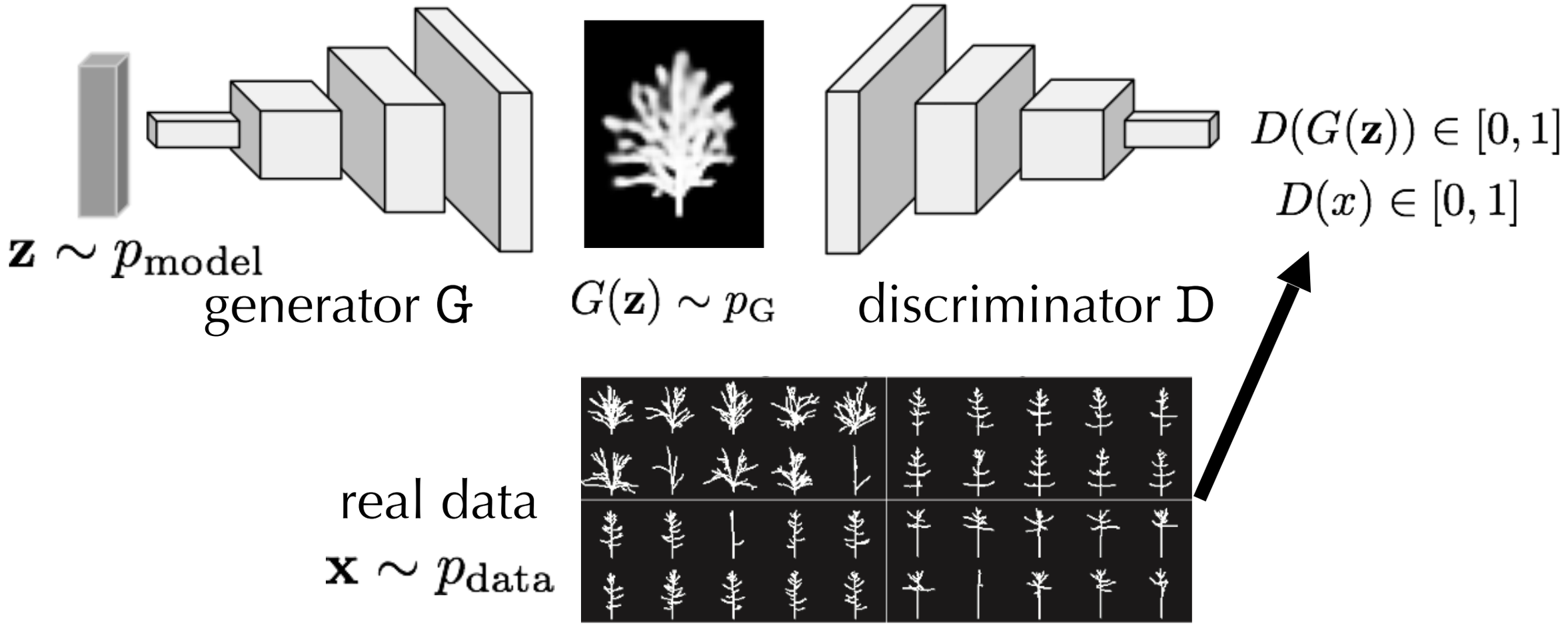
$\mathbf{x} \sim p_{\text{data}}$



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

Generative Adversarial Networks (GANs)

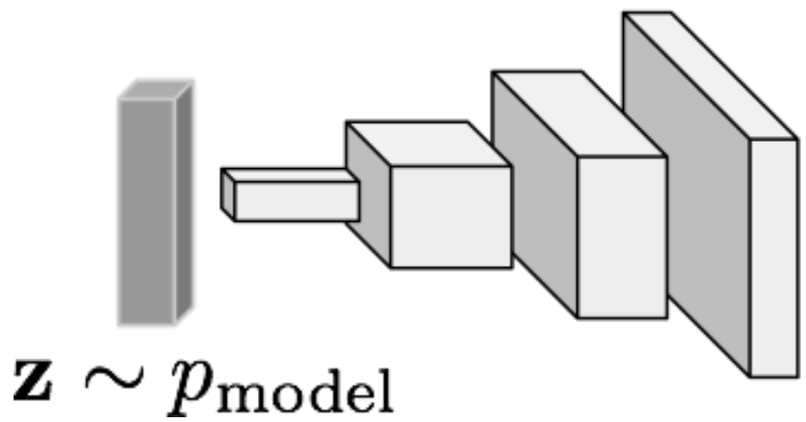
Motivation: Learn loss function



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

Generative Adversarial Networks (GANs)

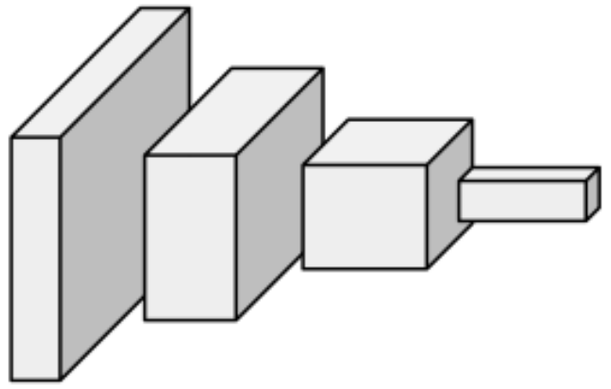
Motivation: Learn loss function



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

generator G

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



discriminator D

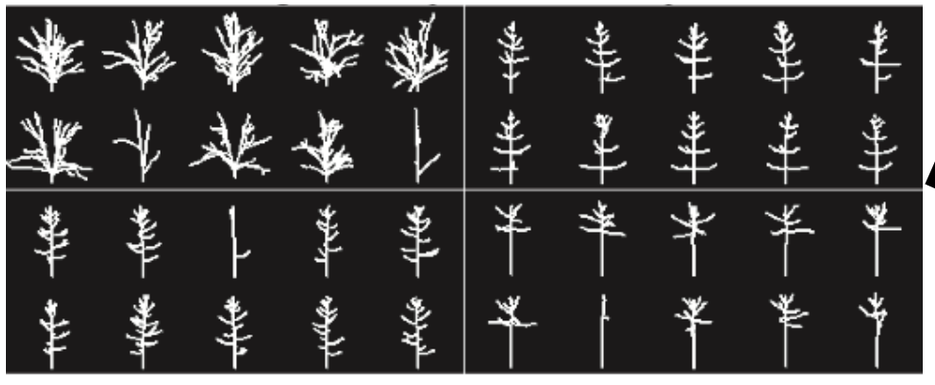
fake or real?

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

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

real data

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

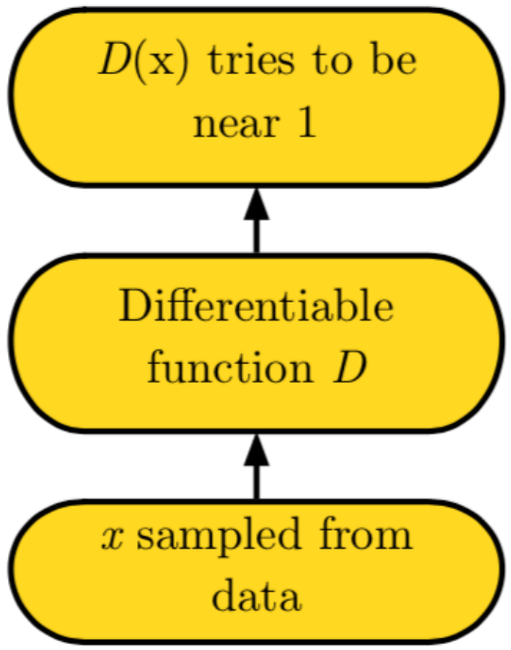


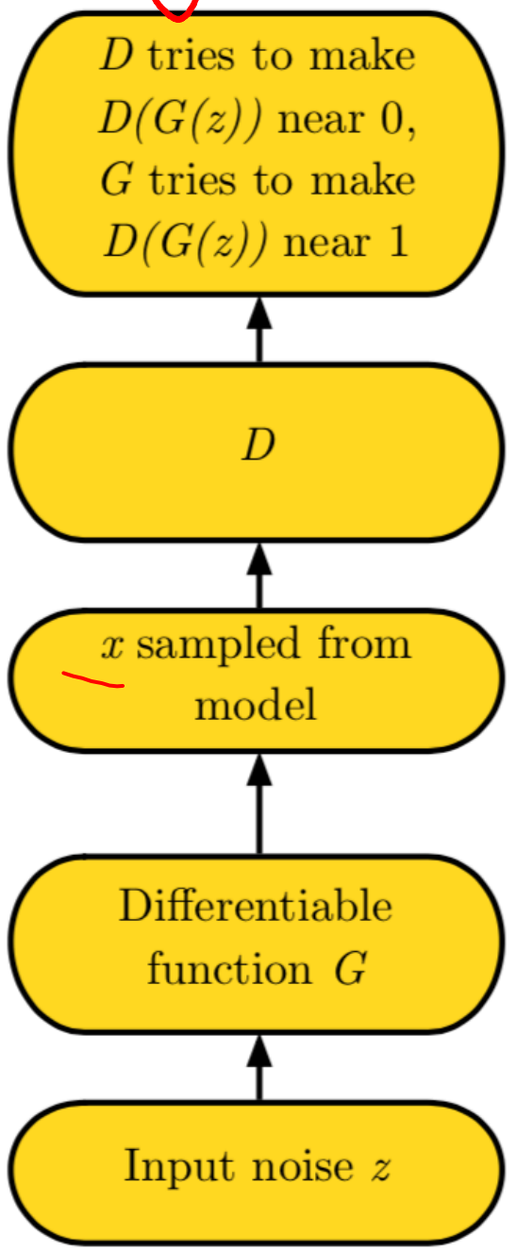
[Goodfellow et al., Generative Adversarial Nets, NIPS 2014]

Generative Adversarial Networks (GANs)

discriminator

generator





(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:

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

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

Game Theoretic View

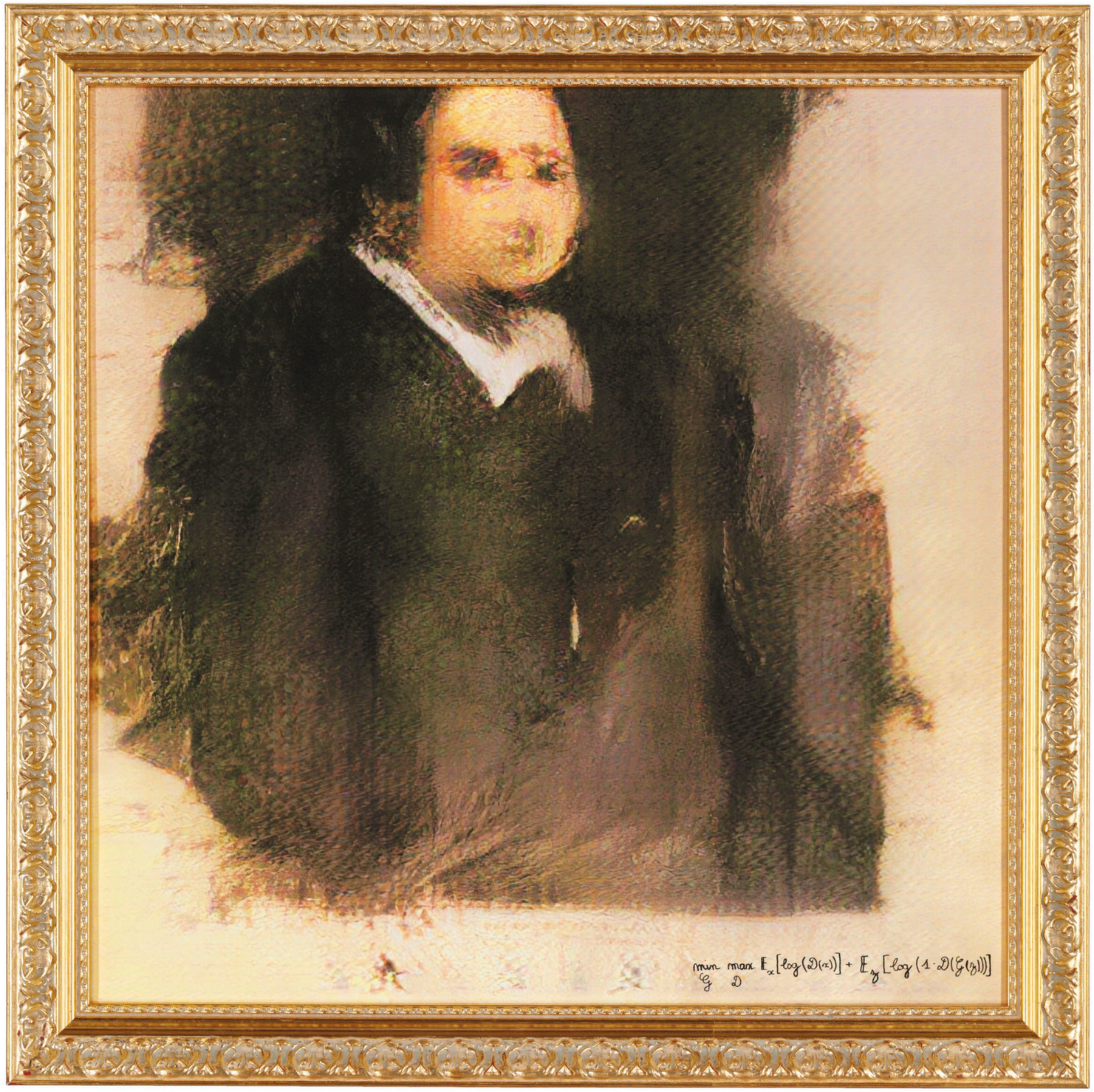
- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look "as real as possible"
- G and D play zero-sum minimax game:

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

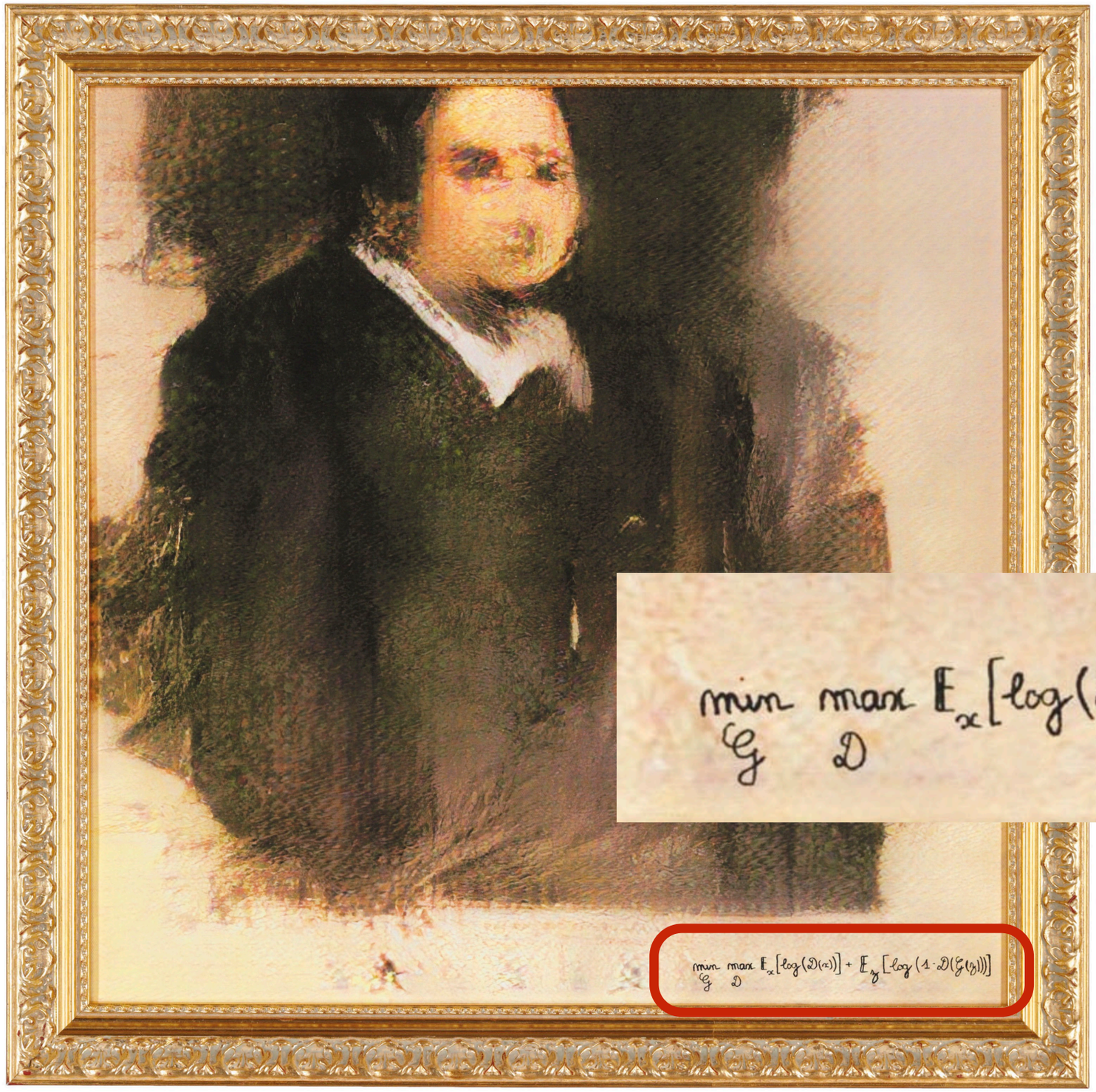
- Equilibrium / global optimum at $p_{\text{data}} = p_G$, i.e., when real and synthetic images are indistinguishable for D:

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

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
- List of tips & tricks for training

Mode Collapse

Generated images span only small subspace

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

Why is Training GANs so Hard?

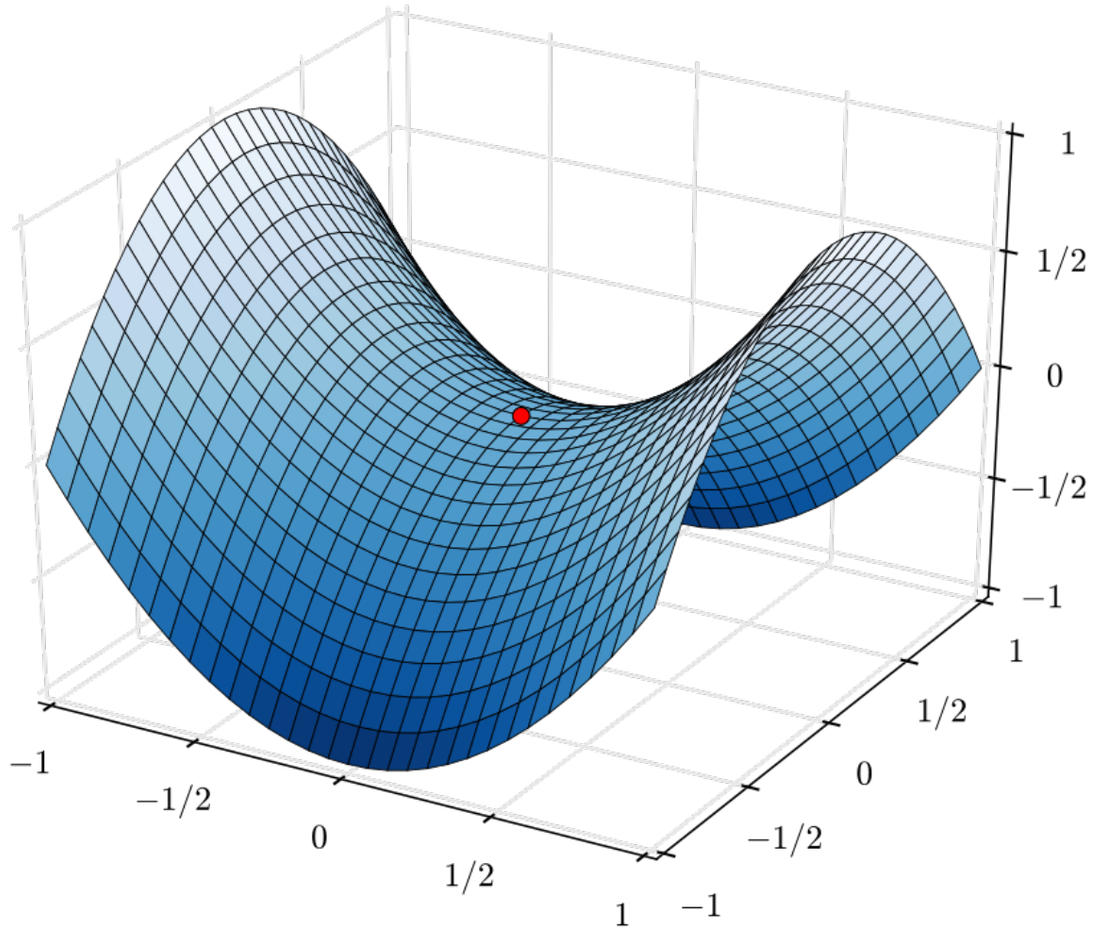


image: © Nicoguaro (CC Attribution Unported 3.0)

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

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

- 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

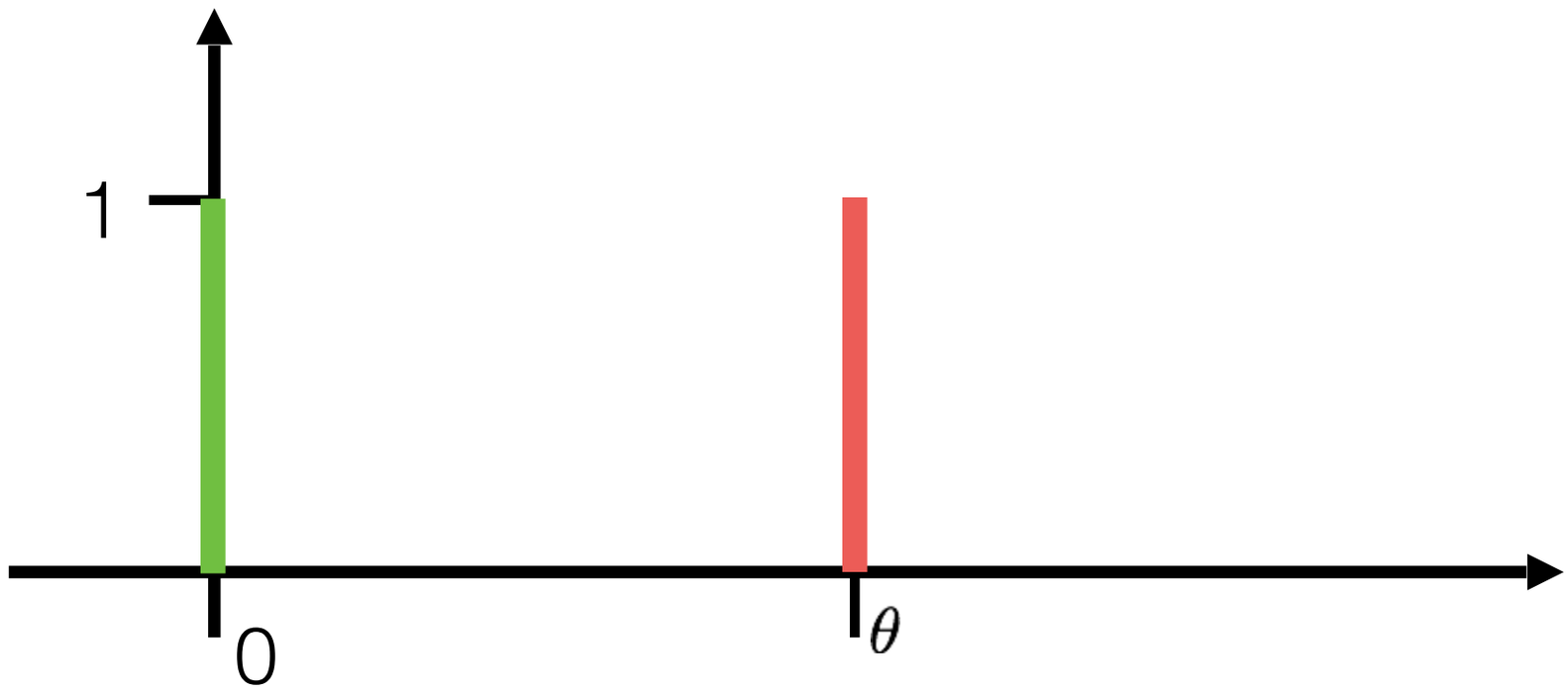
$$\min_G \quad \underbrace{\text{KL}\left(p_{\text{data}}||\frac{p_{\text{data}} + p_G}{2}\right) + \text{KL}\left(p_G||\frac{p_{\text{data}} + p_G}{2}\right)}_{2 \cdot \text{JSD}(p_{\text{data}}||p_G) \quad \text{Jensen-Shannon divergence}} - \log(4)$$

Why is Training GANs so Hard?

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

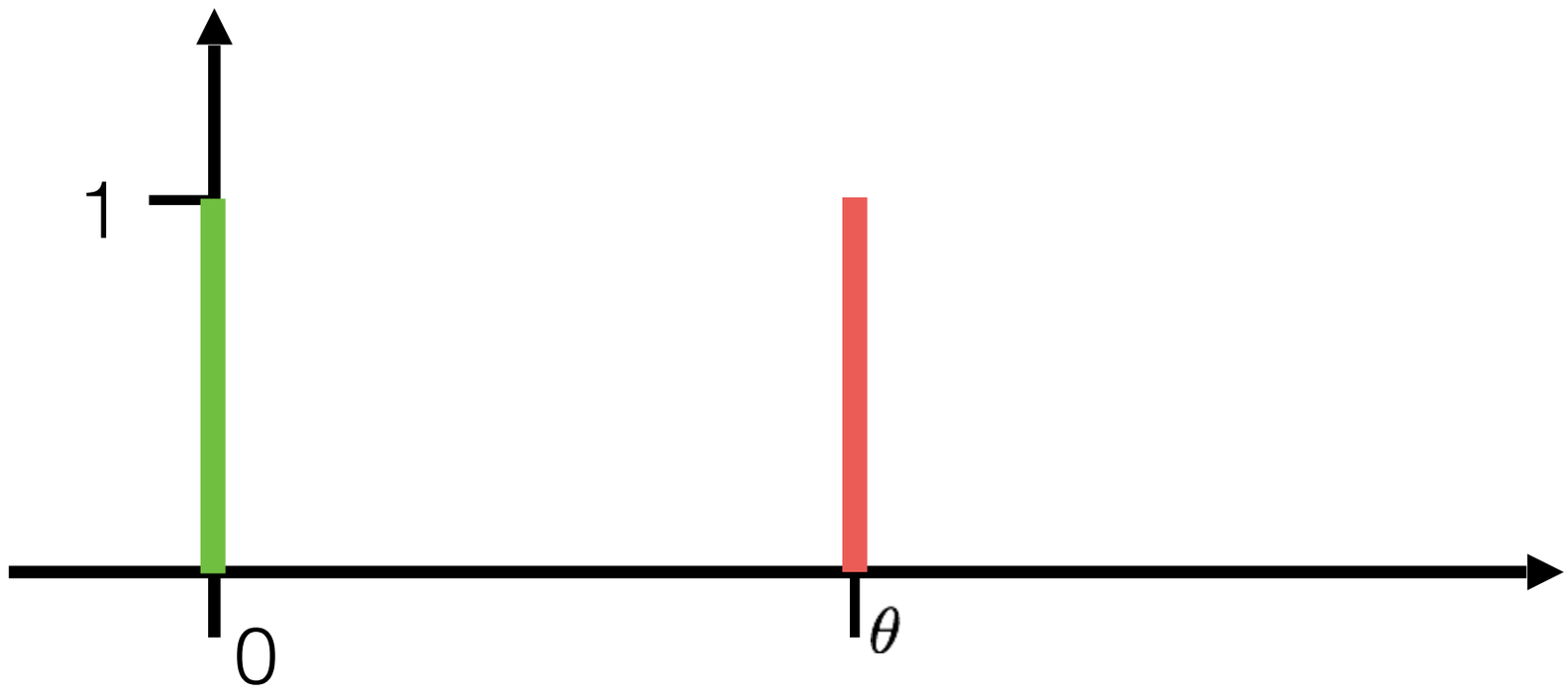
Why is Training GANs so Hard?

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



Why is Training GANs so Hard?

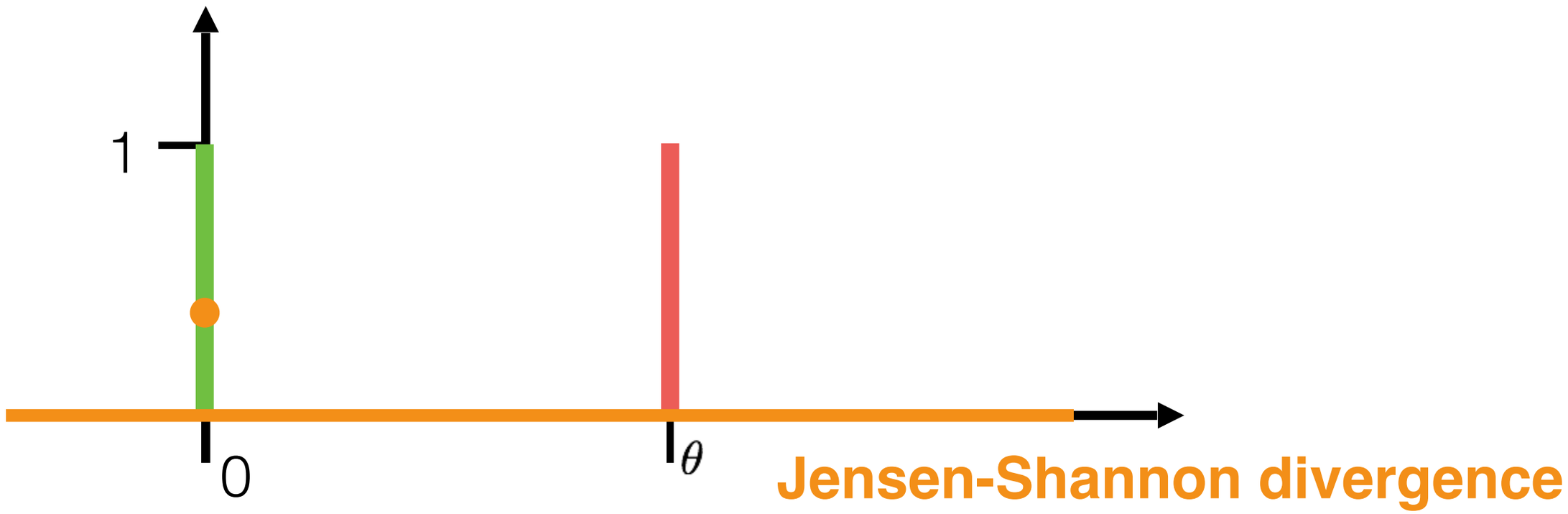
Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent



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

Why is Training GANs so Hard?

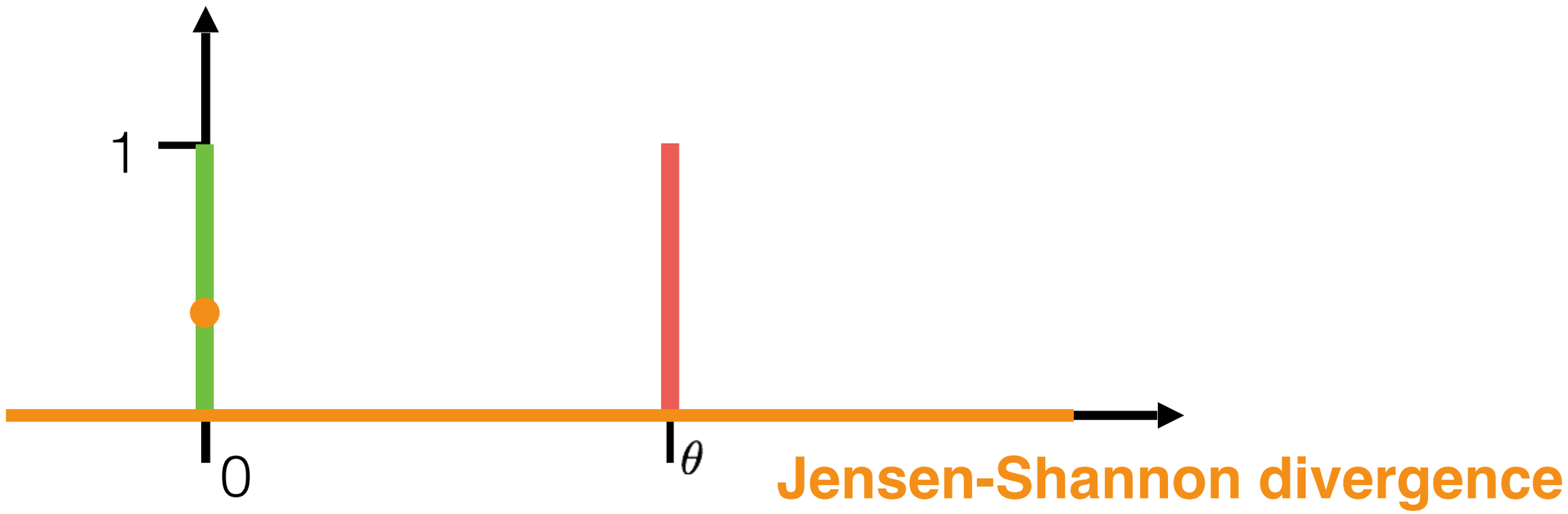
Problem: Jensen-Shannon divergence gradient not always useful → problem when using gradient descent



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

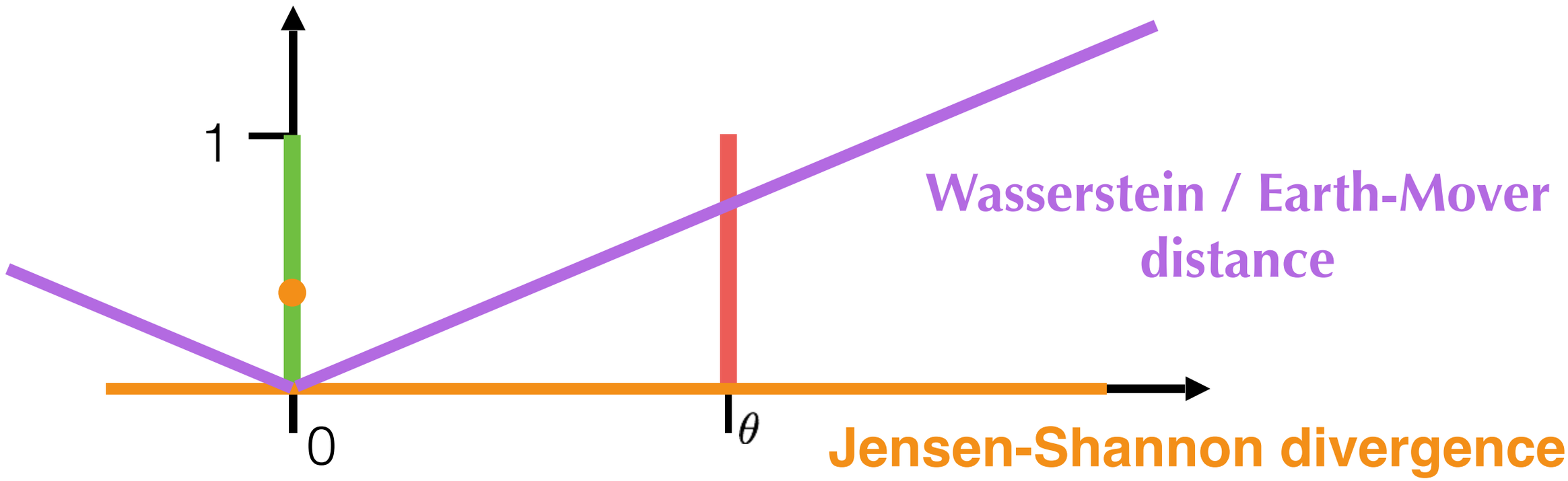


Simple one-parameter distribution $p_\theta(z) = (\theta, z)$, with $z \sim U[0, 1]$

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

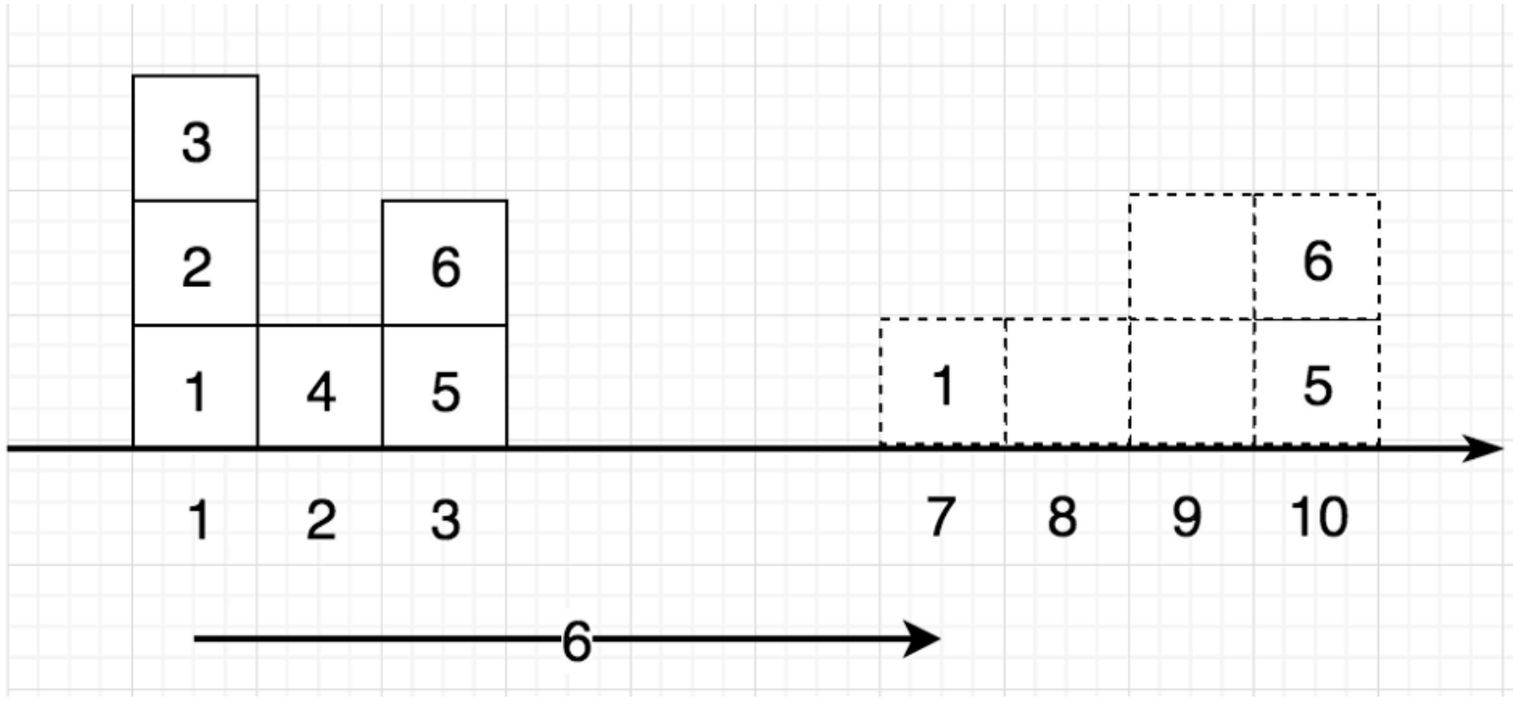


Simple one-parameter distribution $p_\theta(z) = (\theta, z)$, with $z \sim U[0, 1]$

Practical relevance: Operating on disjoints manifolds

Wasserstein GANs (WGANs)

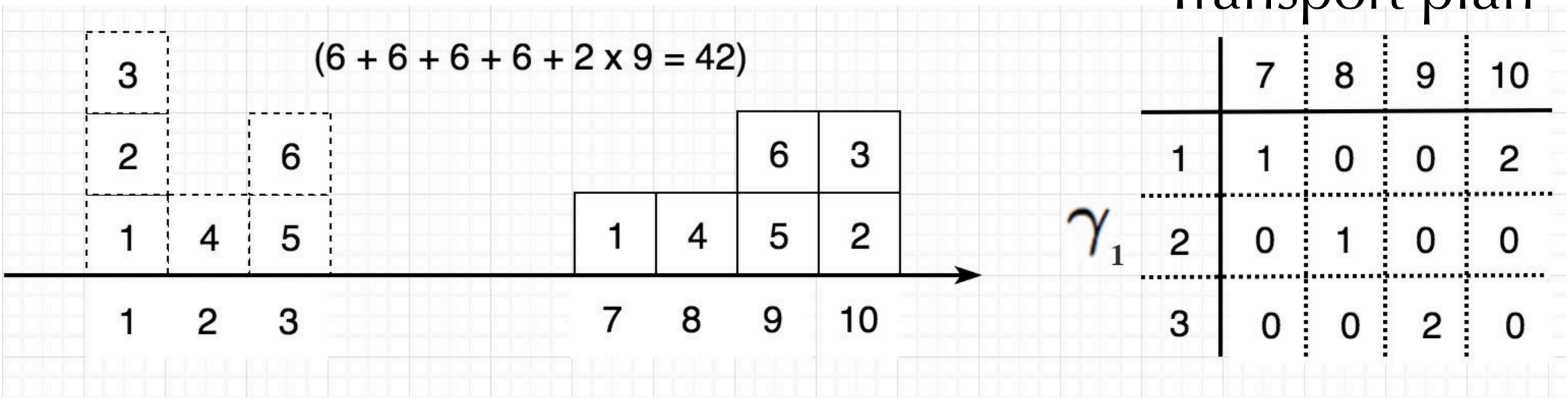
Earth-Mover distance: Minimum amount of work to move weight from $p(x)$ to $q(x)$



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

Wasserstein GANs (WGANs)

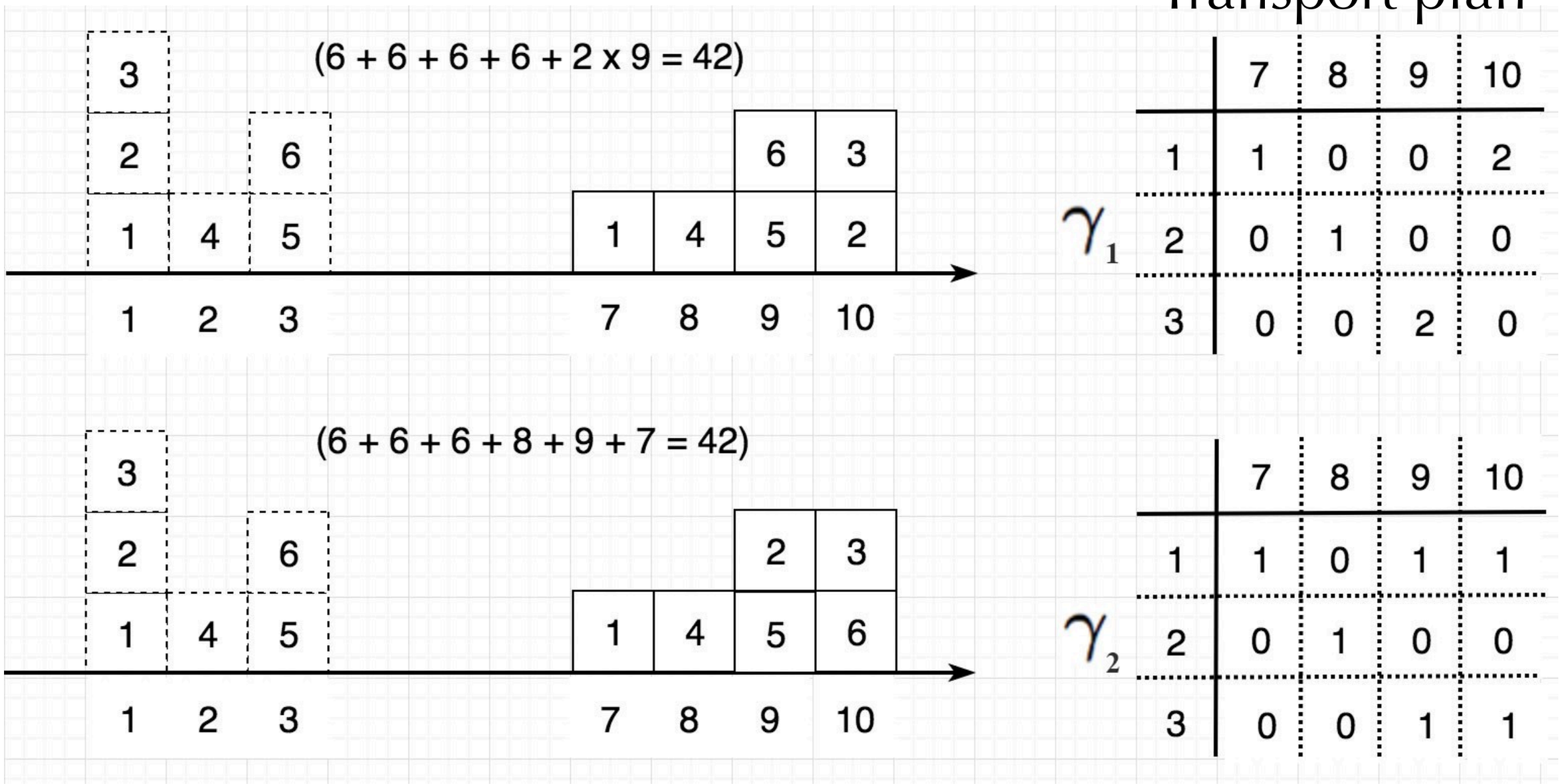
Earth-Mover distance: Minimum amount of work to move weight from p(x) to q(x)



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 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 $\mathtt{p(x)}$ to $\mathtt{q(x)}$

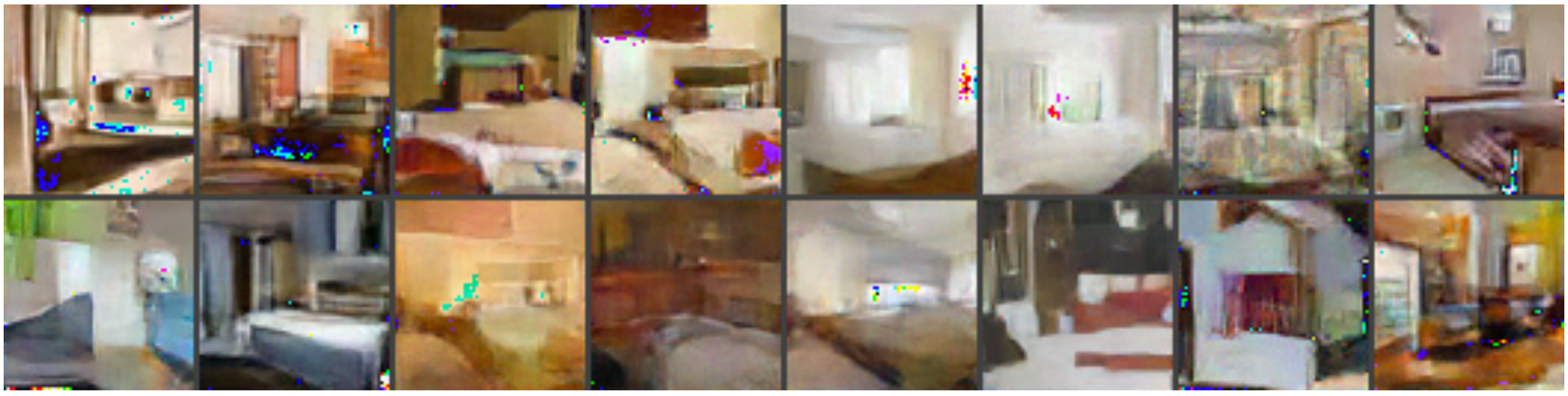
$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\left[\|x - y\| \right]$$

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

WGANs vs. GANs

GAN

WGAN



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

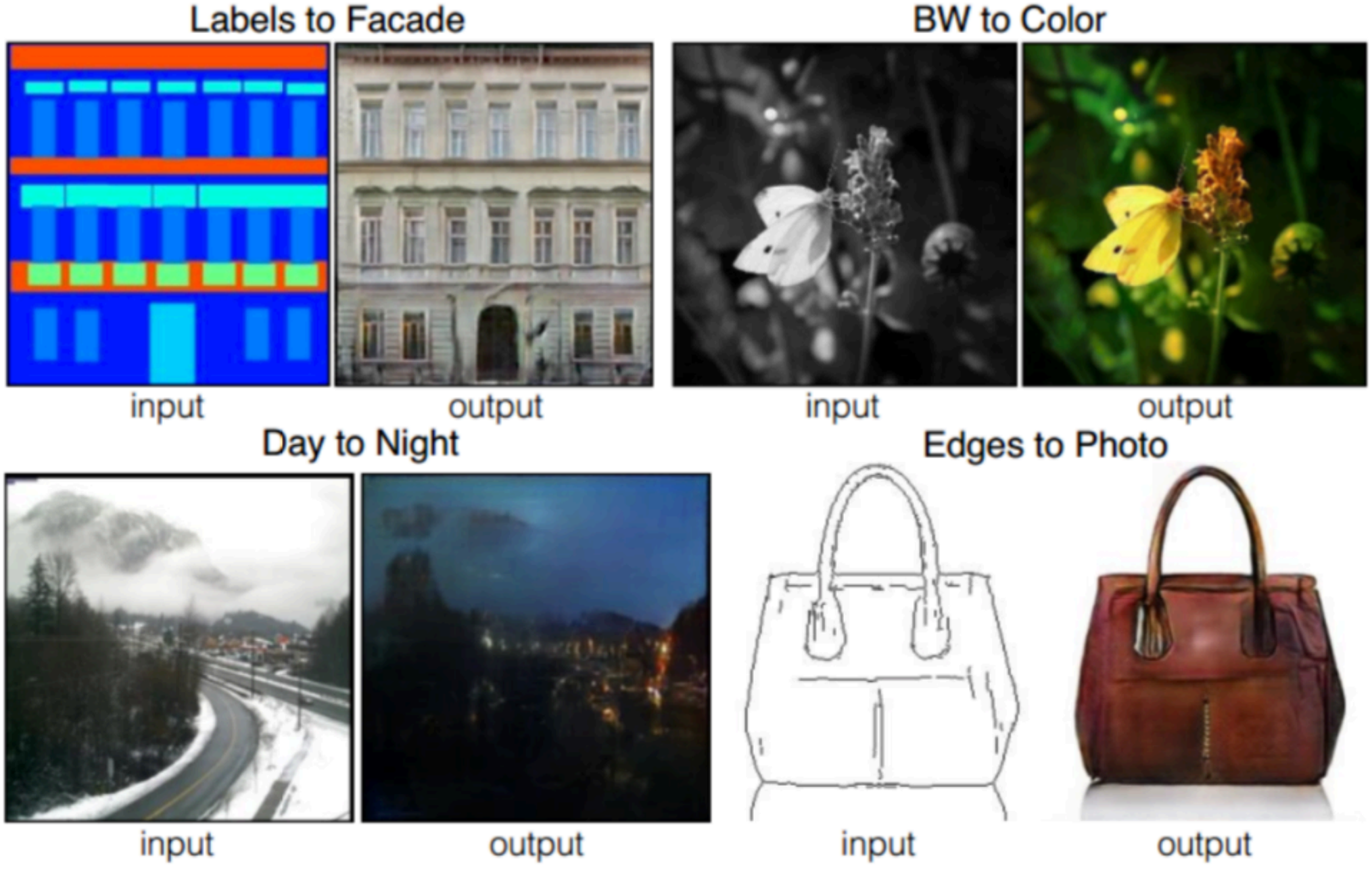
State-of-the-art Results

State-of-the-art Results

Today

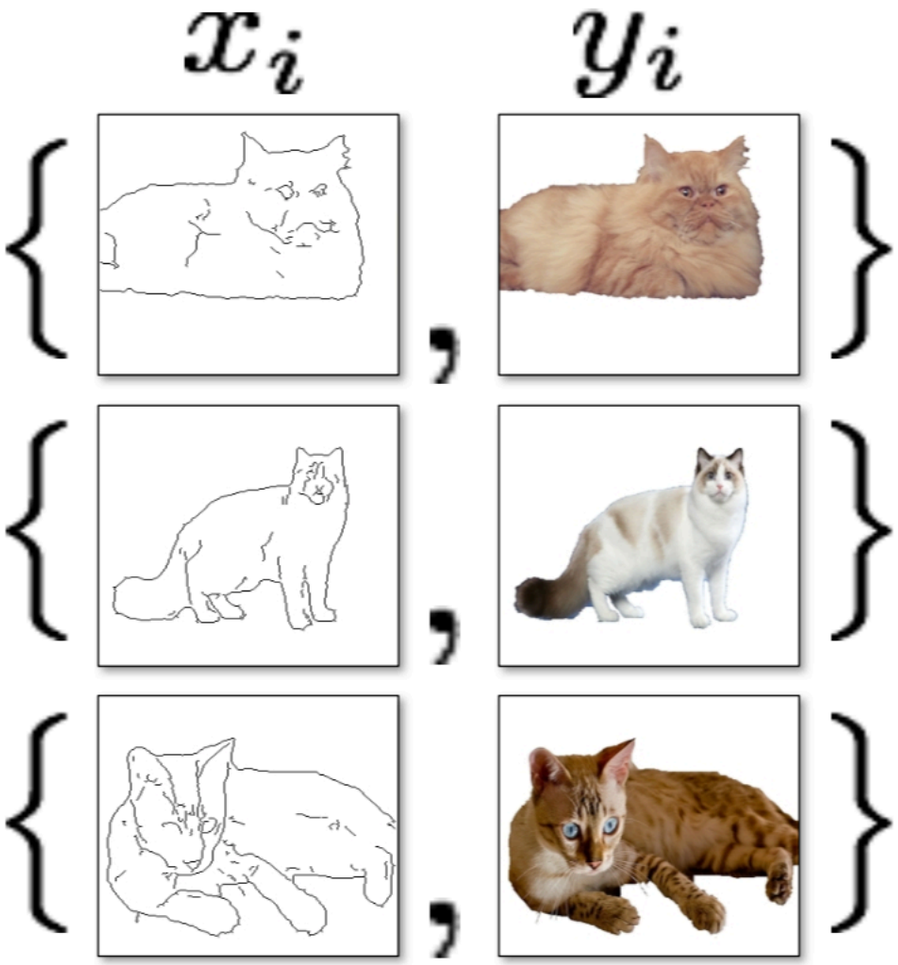
- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GANs)
- **Unsupervised Image Translation**

Image-To-Image Translation



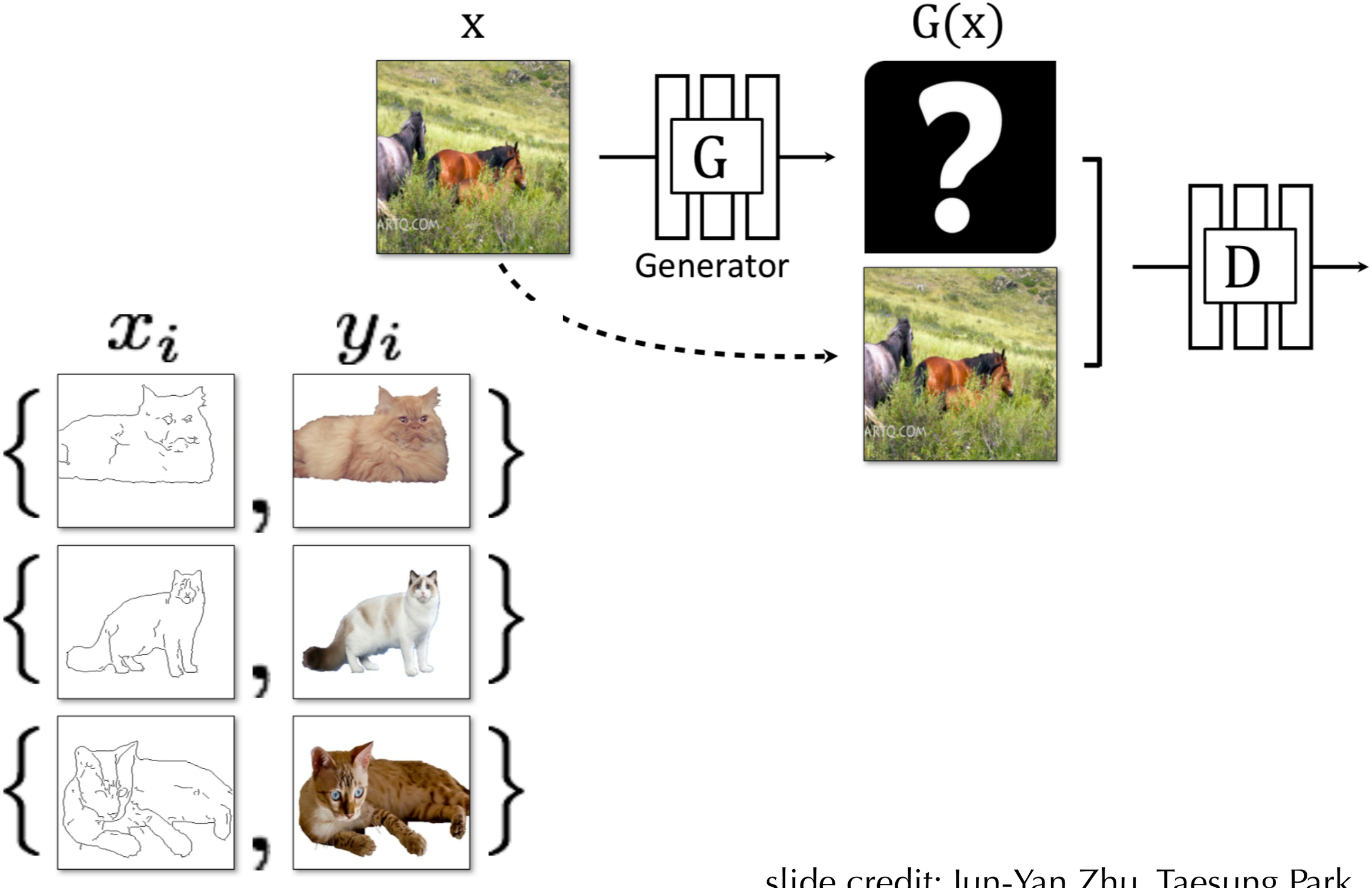
slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



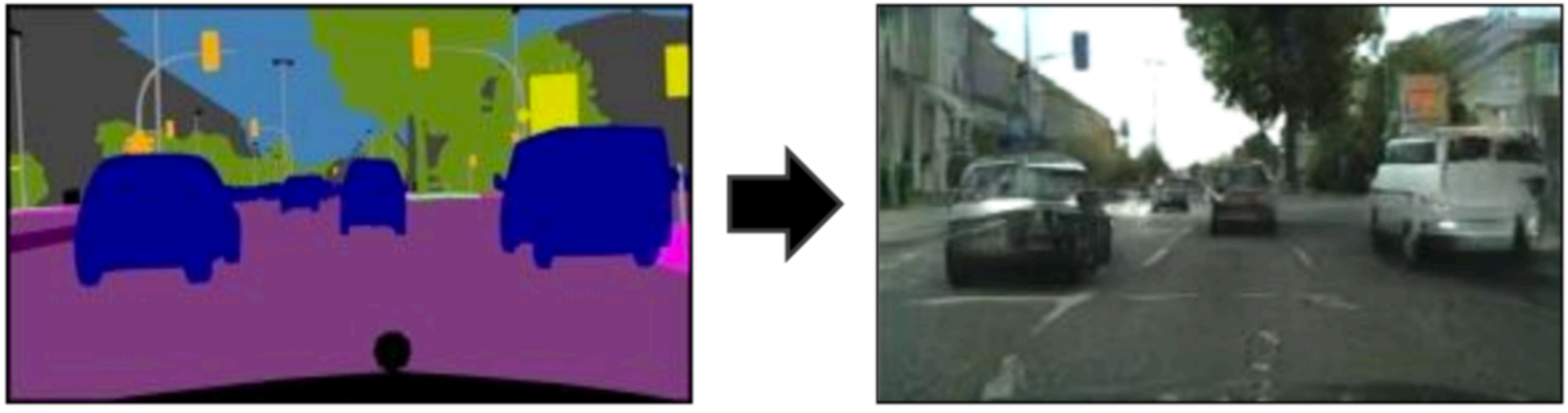
slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation

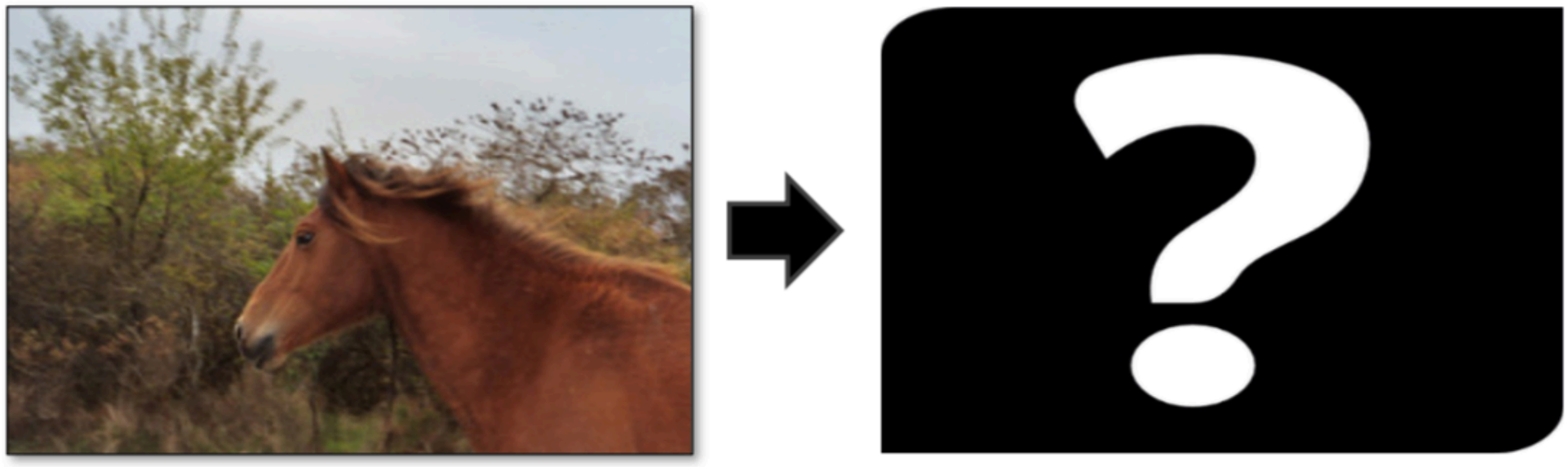


slide credit: Jun-Yan Zhu, Taesung Park

Supervised Image-To-Image Translation



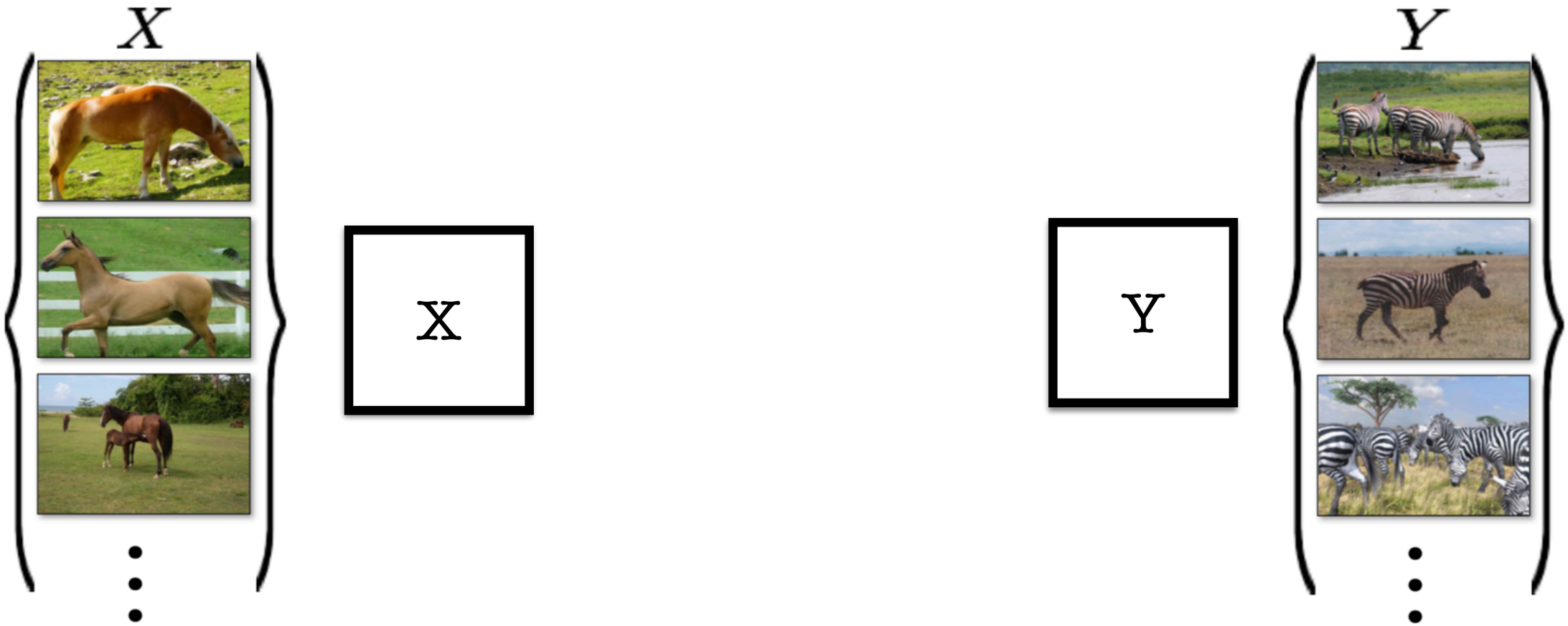
Label ↔ photo: per-pixel labeling



Labelled data often hard or impossible to obtain

slide credit: Jun-Yan Zhu, Taesung Park

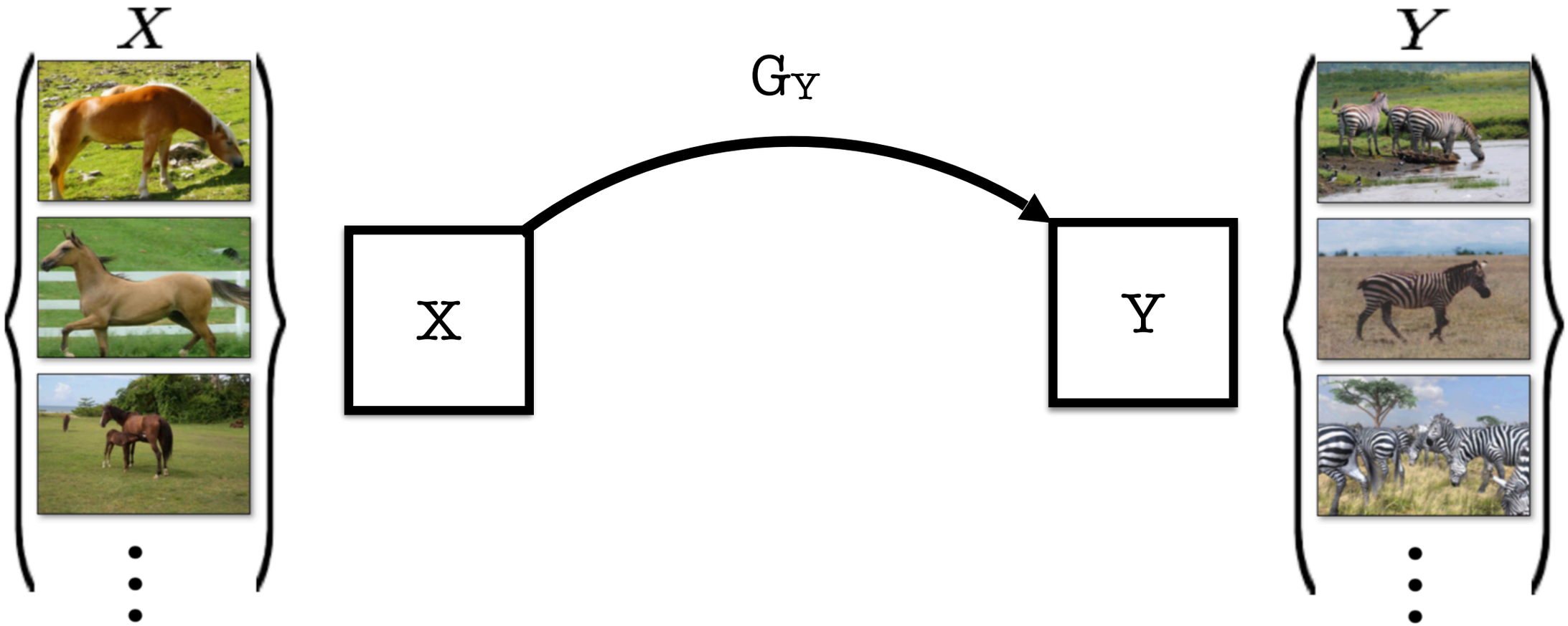
CycleGan: Cycle -Consistent Adversarial Networks



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

slide credit: Jun-Yan Zhu, Taesung Park

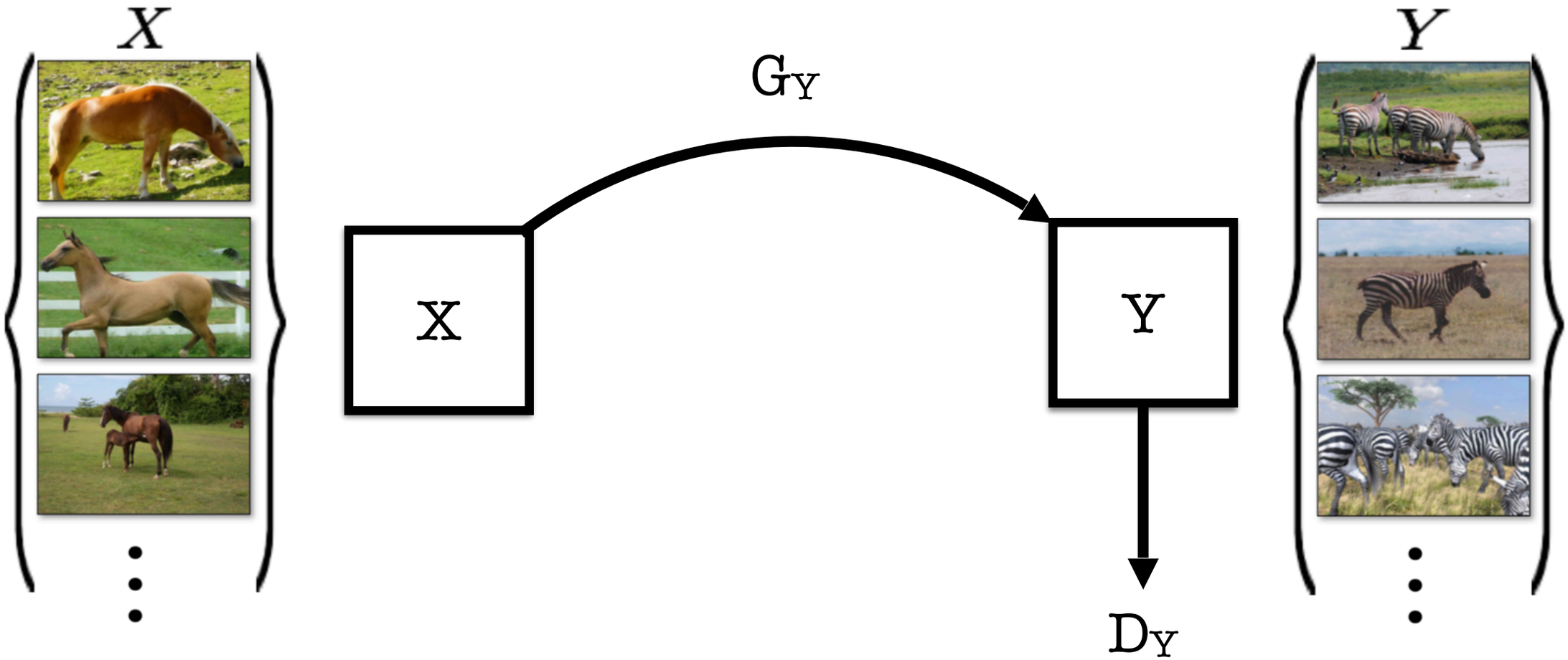
CycleGan: Cycle -Consistent Adversarial Networks



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

slide credit: Jun-Yan Zhu, Taesung Park

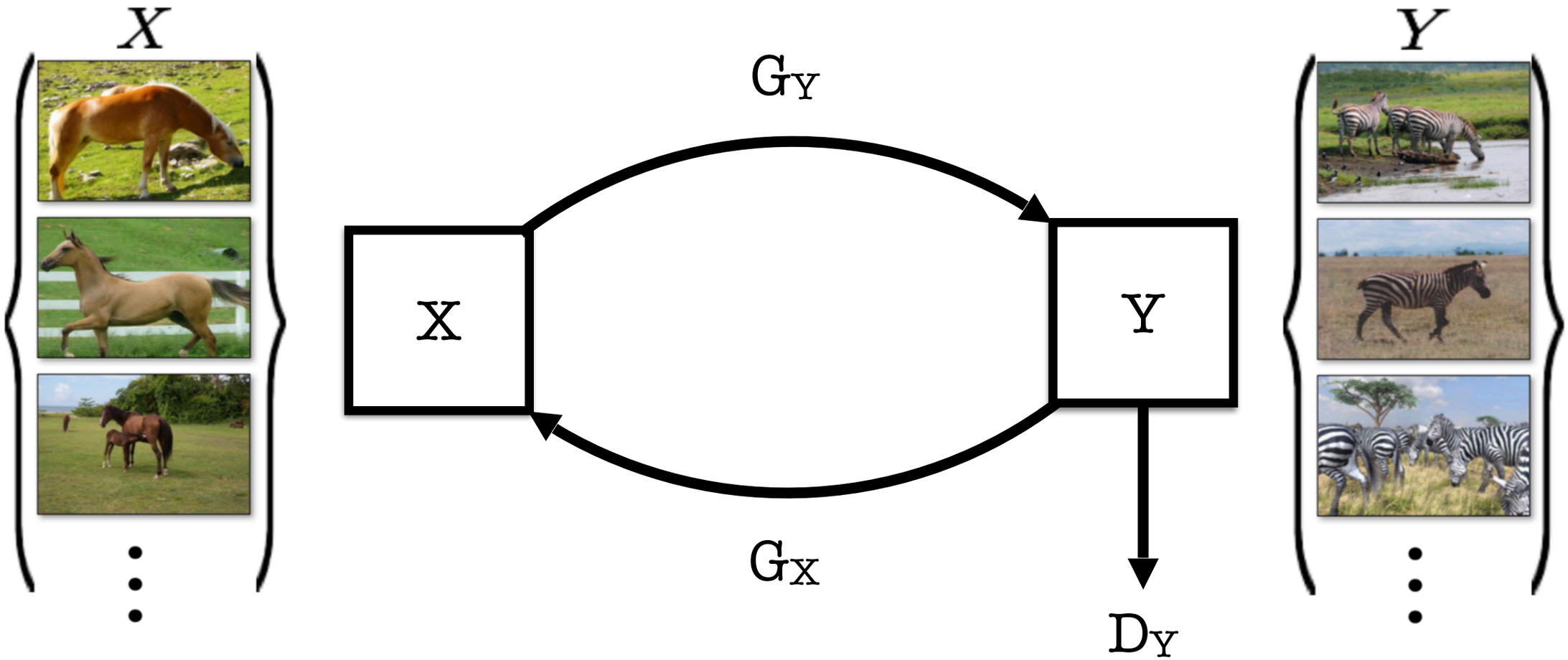
CycleGan: Cycle -Consistent Adversarial Networks



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

slide credit: Jun-Yan Zhu, Taesung Park

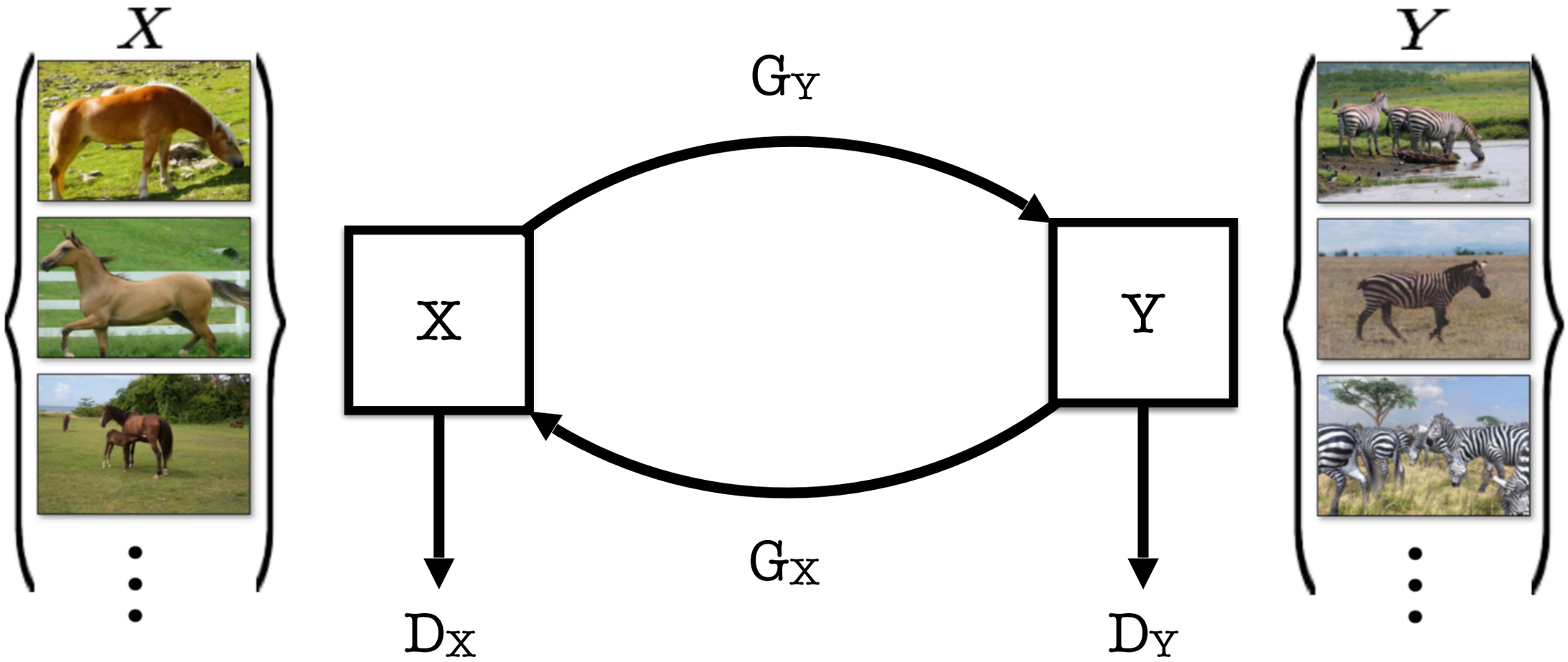
CycleGan: Cycle -Consistent Adversarial Networks



[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



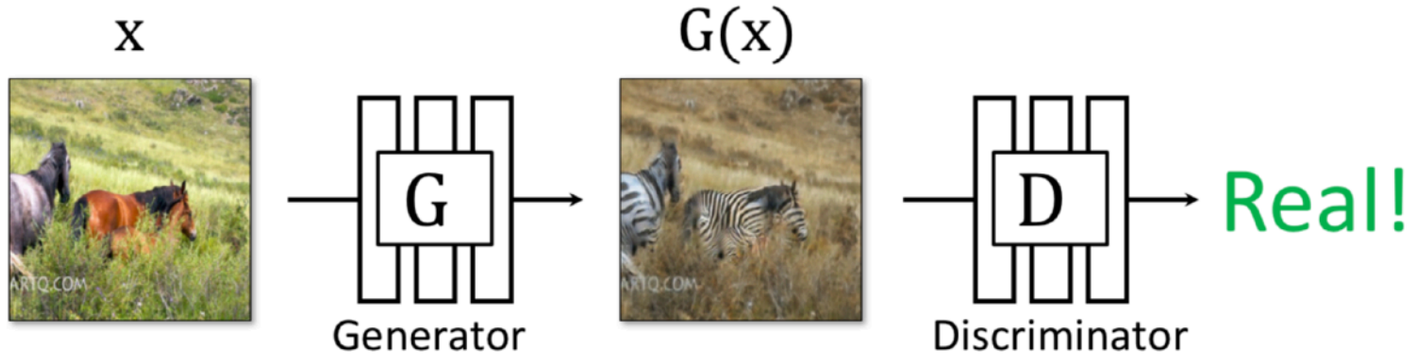
[Zhu & Park et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017]

slide credit: Jun-Yan Zhu, Taesung Park

Unsupervised Image-To-Image Translation

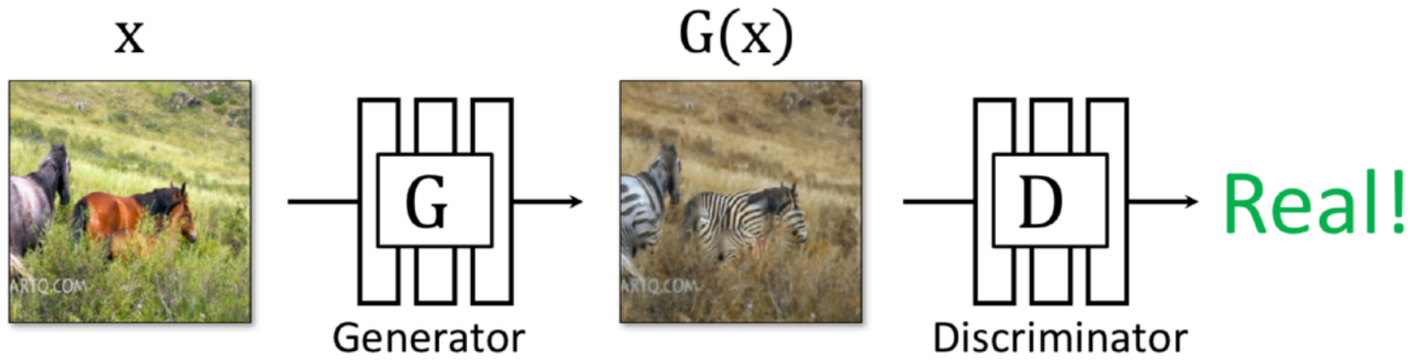
slide credit: Jun-Yan Zhu, Taesung Park

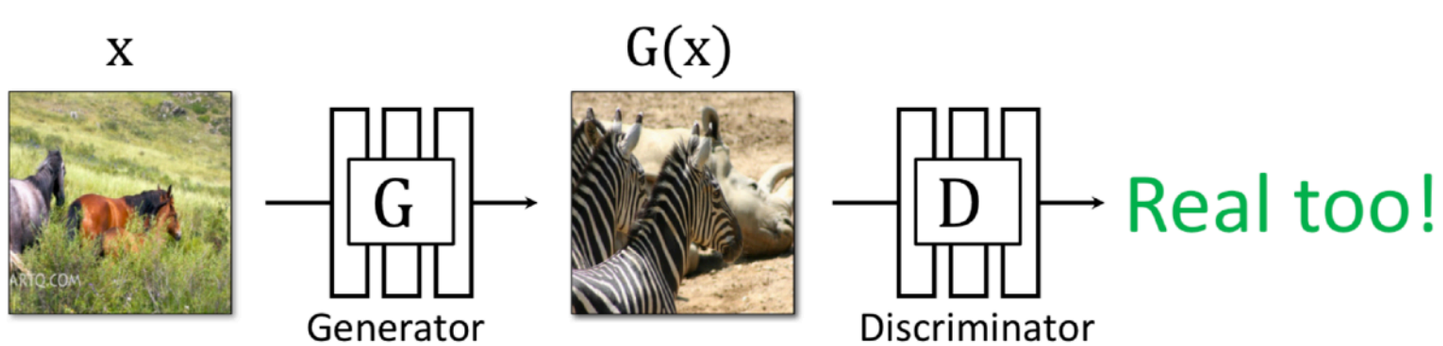
Unsupervised Image-To-Image Translation



slide credit: Jun-Yan Zhu, Taesung Park

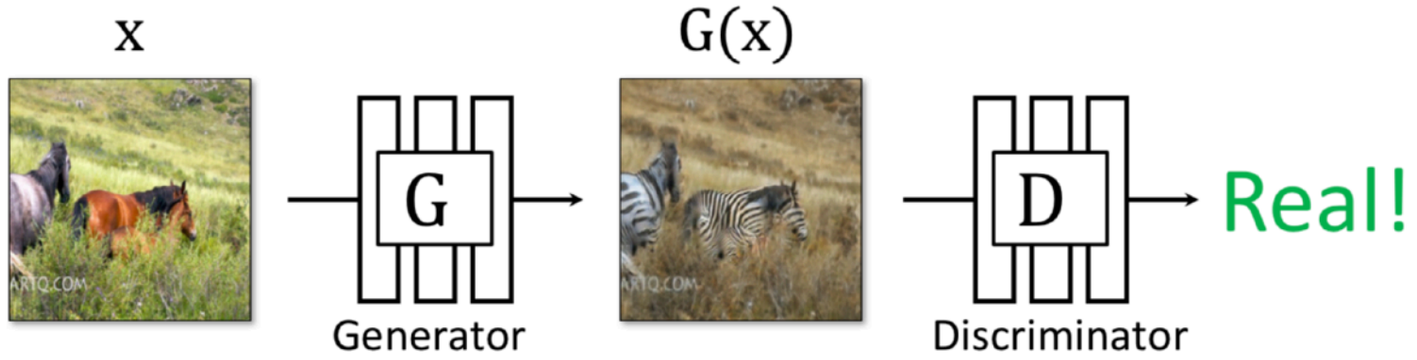
Unsupervised Image-To-Image Translation

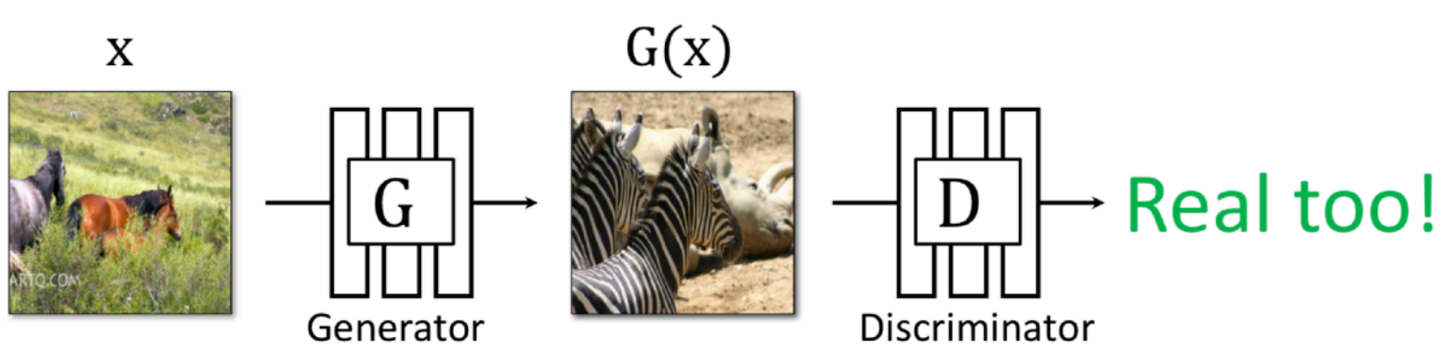




slide credit: Jun-Yan Zhu, Taesung Park

Unsupervised Image-To-Image Translation



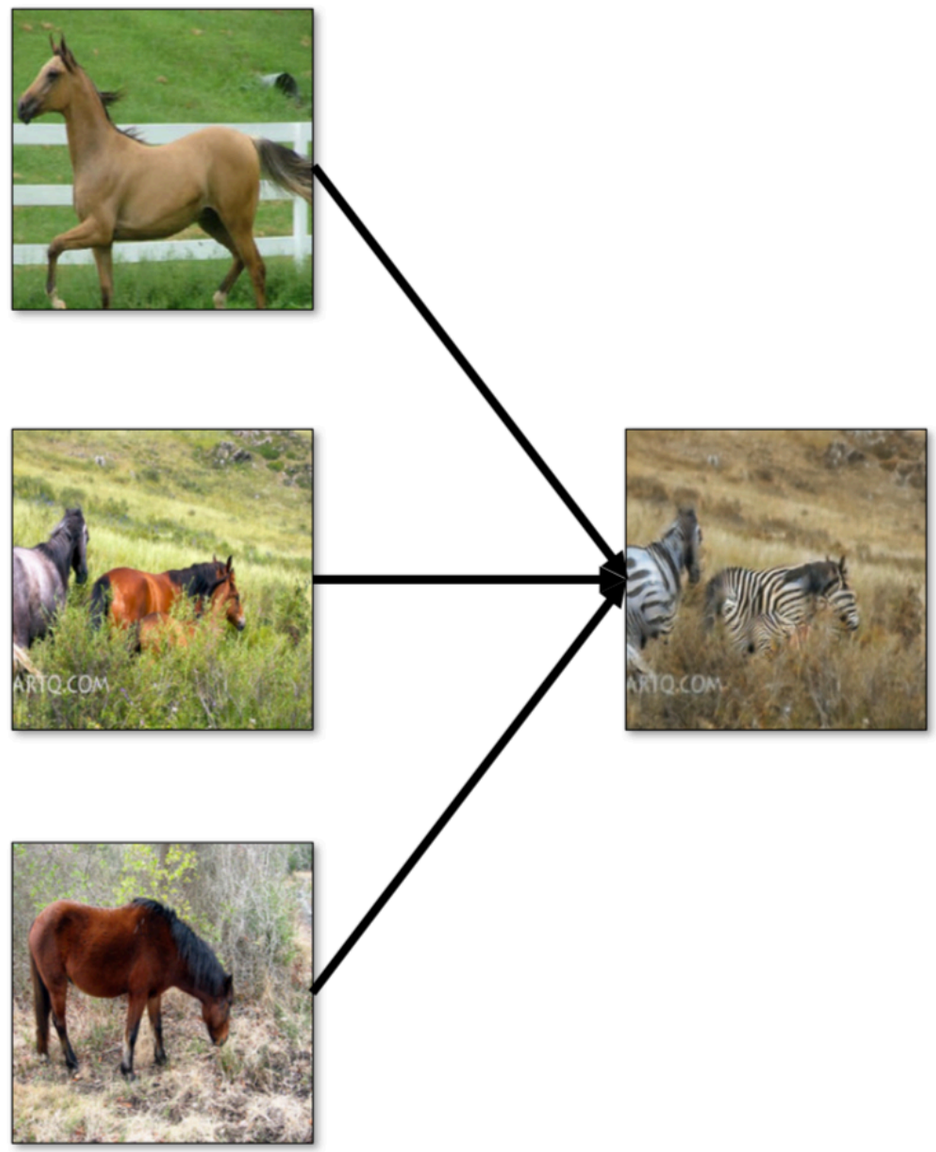


mode collapse

slide credit: Jun-Yan Zhu, Taesung Park

CycleGan: Cycle -Consistent Adversarial Networks

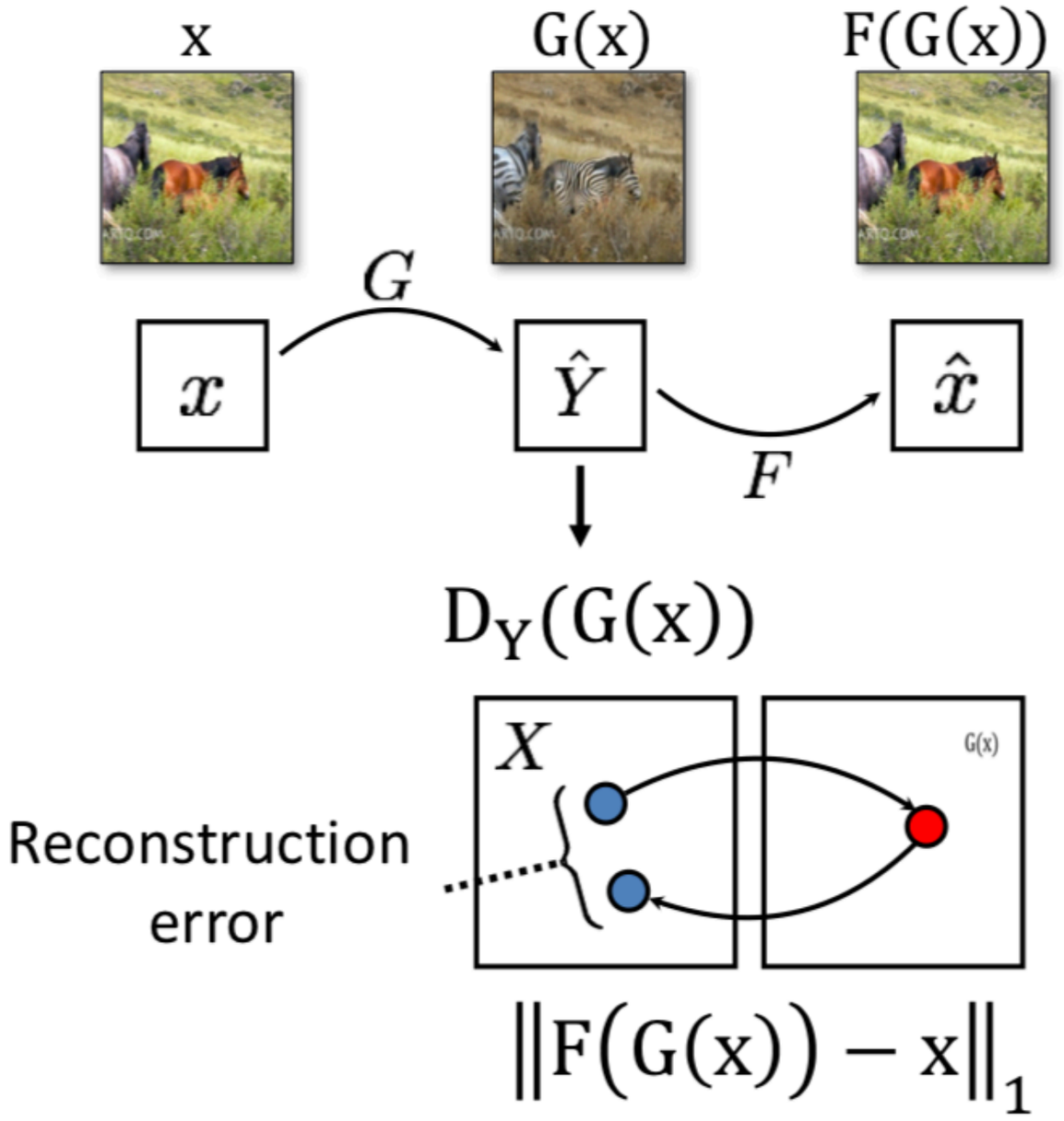
Does not prevent mode collapse



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

slide credit: Jun-Yan Zhu, Taesung Park

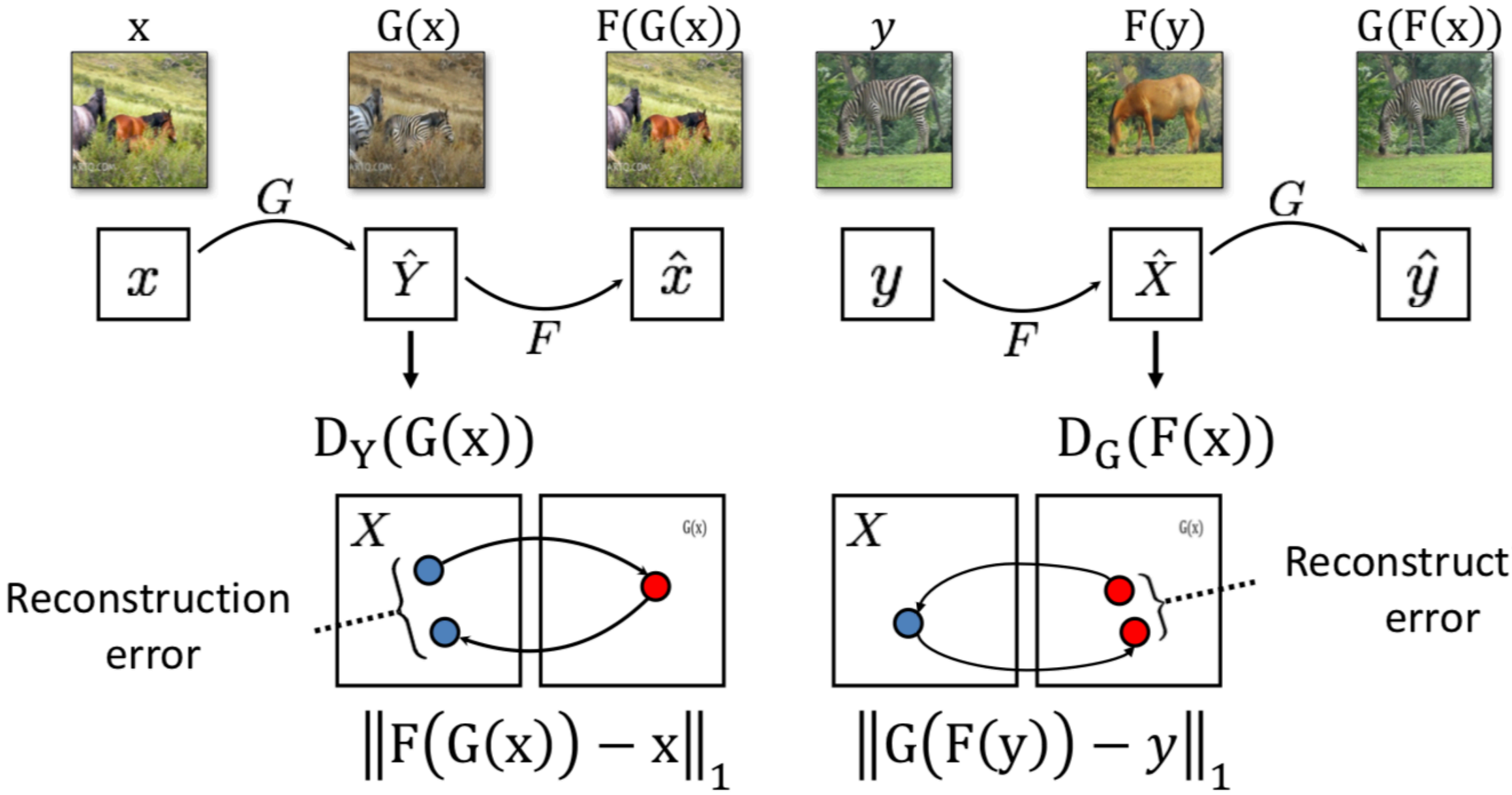
CycleGan: Cycle -Consistent Adversarial Networks



[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 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

[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]

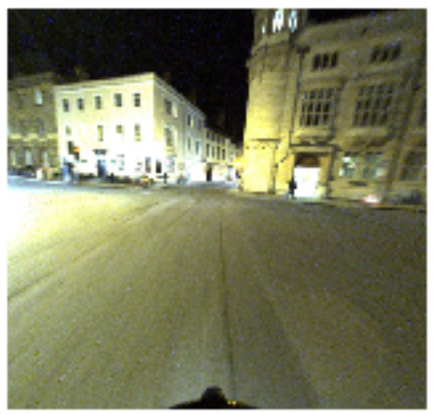
CycleGan Application: Image Retrieval

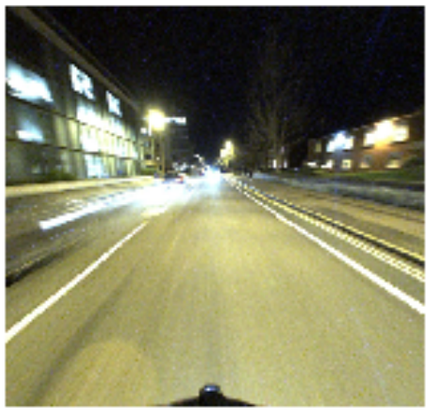
[Maddern, Pascoe, Linegar, Newman, 1 Year, 1000km: The Oxford RobotCar Dataset, IJRR 2017]

[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

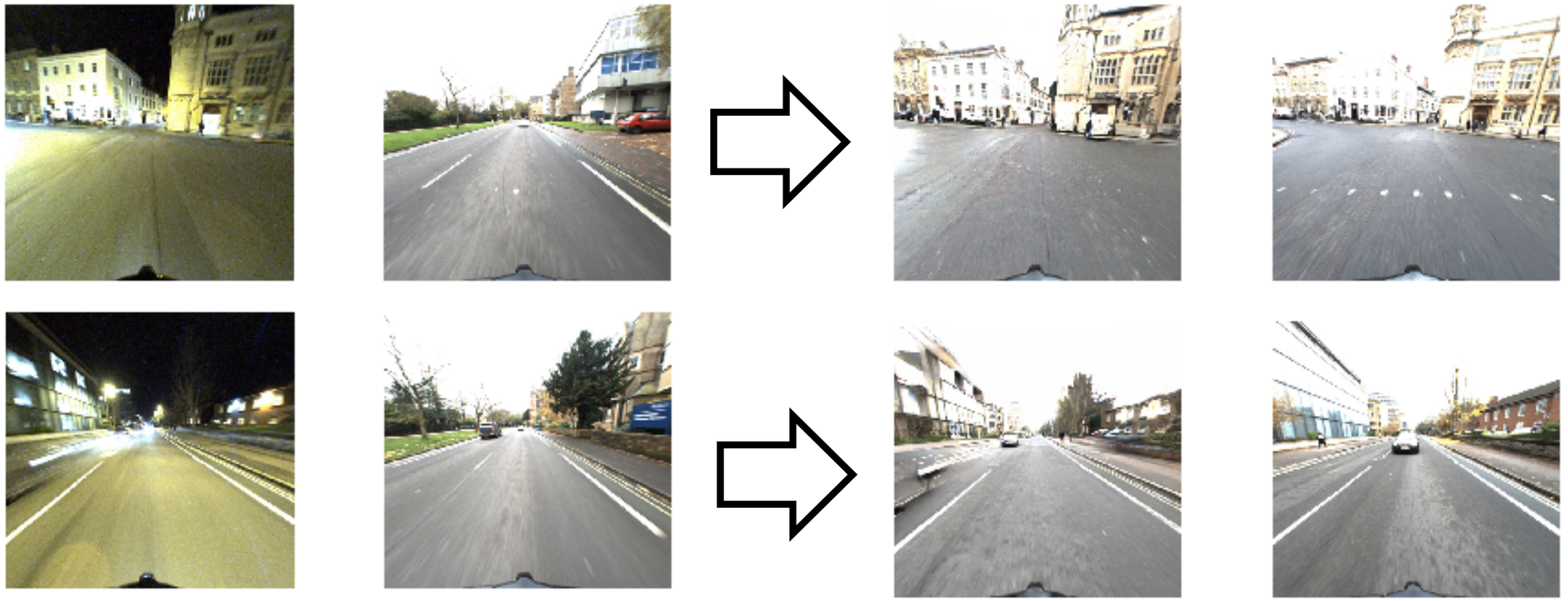
CycleGan Application: Image Retrieval

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 modeling via sampling from probability distributions
  - 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

Lessons Learned

- Main lessons from this lecture
  - Generative modeling via sampling from probability distributions
  - 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: Medical Image Analysis by Jennifer Alvén

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
- GANs are an active research topic, be prepared to read a lot