SSY098 - Image Analysis

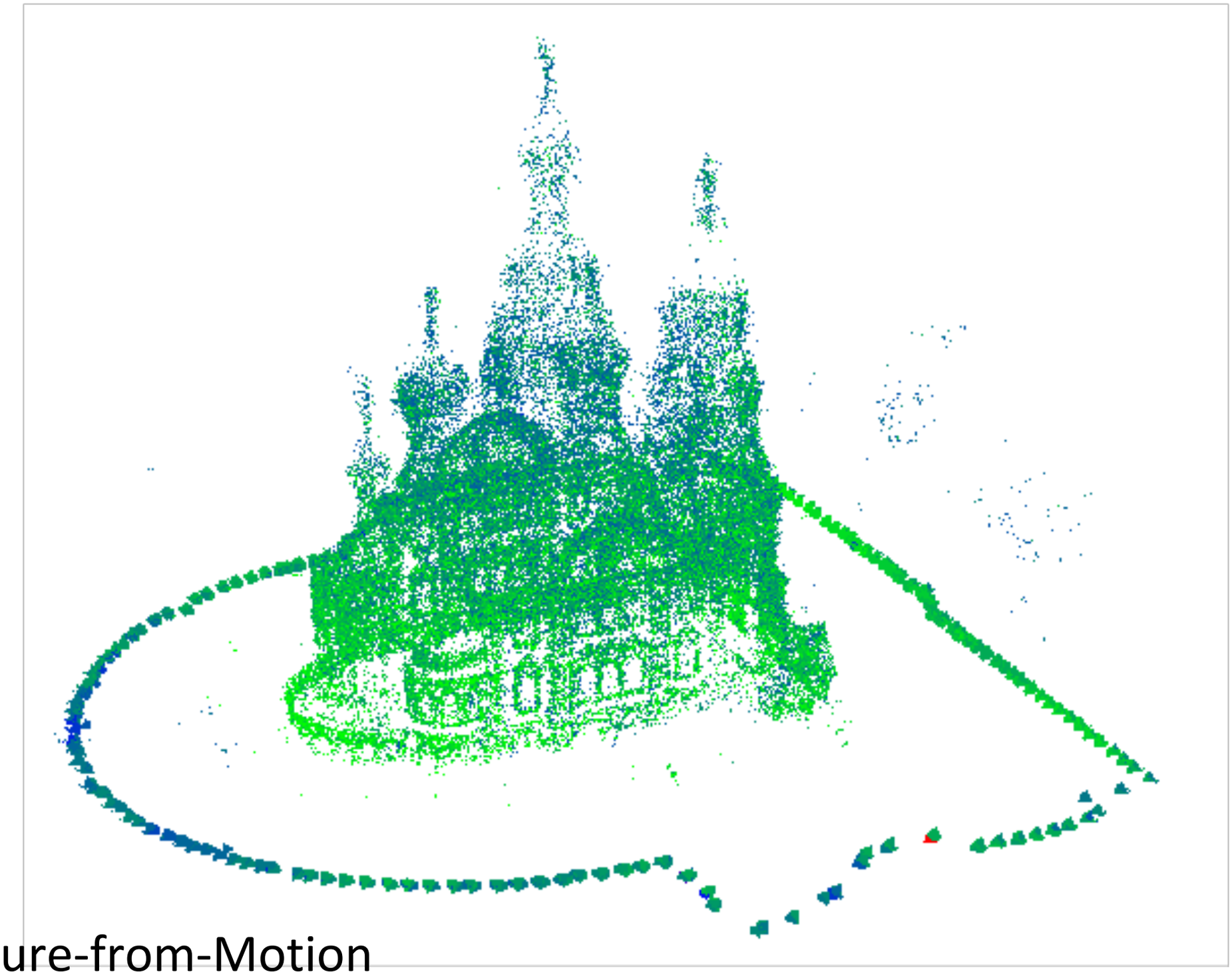
Lecture 11 - Generative Neural Networks

Torsten Sattler

Last Lecture

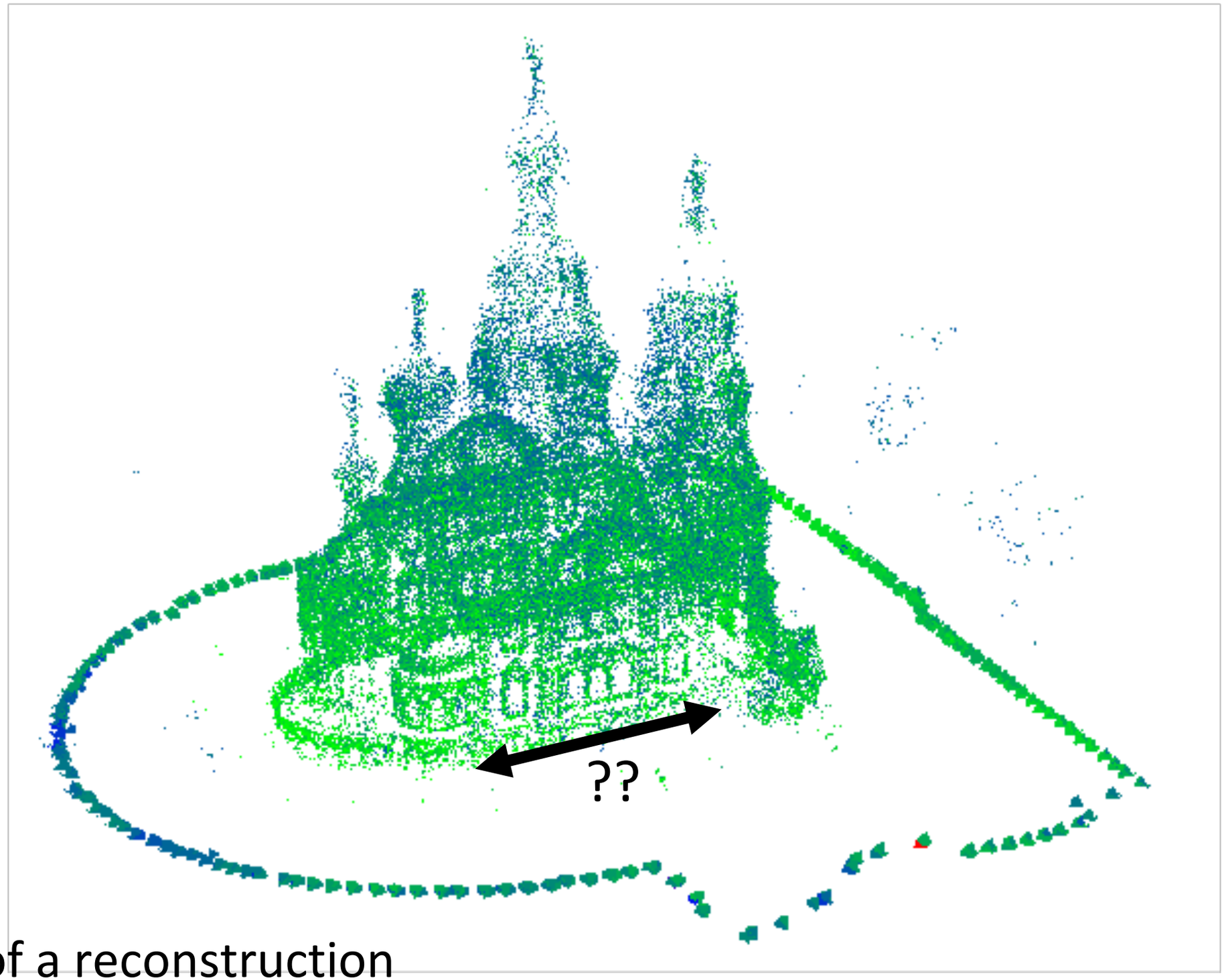
Jan. 20	Introduction, Linear classifiers and filtering	
Jan. 23	Filtering, gradients, scale	Lab 1
Jan. 27	Local features	
Jan. 30	Learning a classifier	Lab 2
Feb. 3	Convolutional neural networks	
Feb. 6	More convolutional neural networks	
Feb. 10	Robust model fitting and RANSAC	Lab 3
Feb. 13	Image registration	
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	

Last Lecture



Structure-from-Motion

Last Lecture



Scale of a reconstruction

Last Lecture

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

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

3D reconstructions defined up to scale

Last Lecture

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

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

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

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

3D reconstructions defined up to scale

Last Lecture

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

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

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

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

3D reconstructions defined up to scale

Last Lecture

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

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

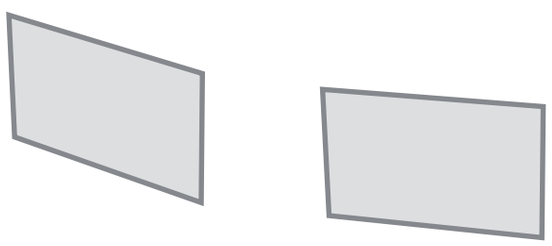
Scale 3D scene by arbitrary factor $\mathtt{s}\neq 0$

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

3D reconstructions defined up to scale

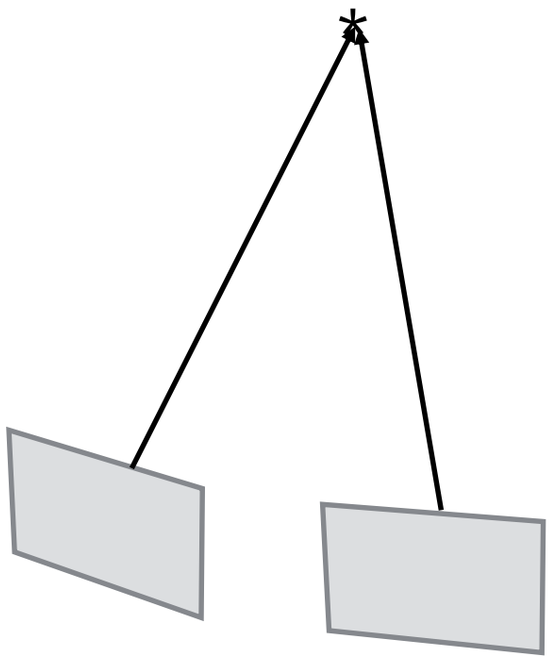
Last Lecture

Initialize motion from two views



Relative pose for two images

Last Lecture

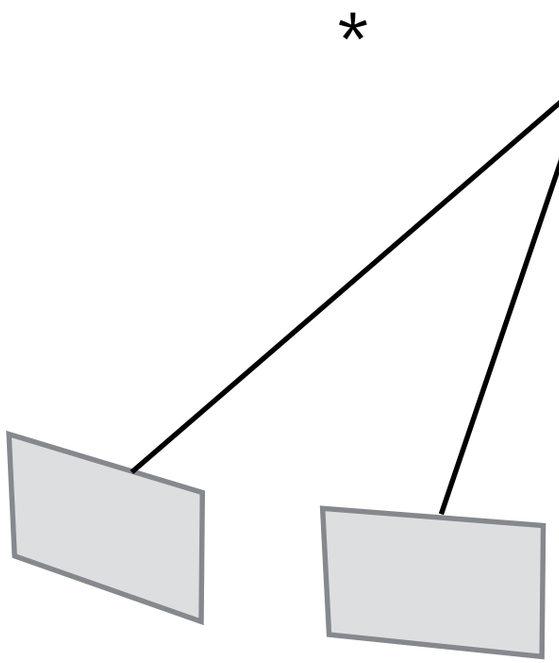


Initialize motion from two views

Initialize structure from two views

Triangulate 3D points

Last Lecture

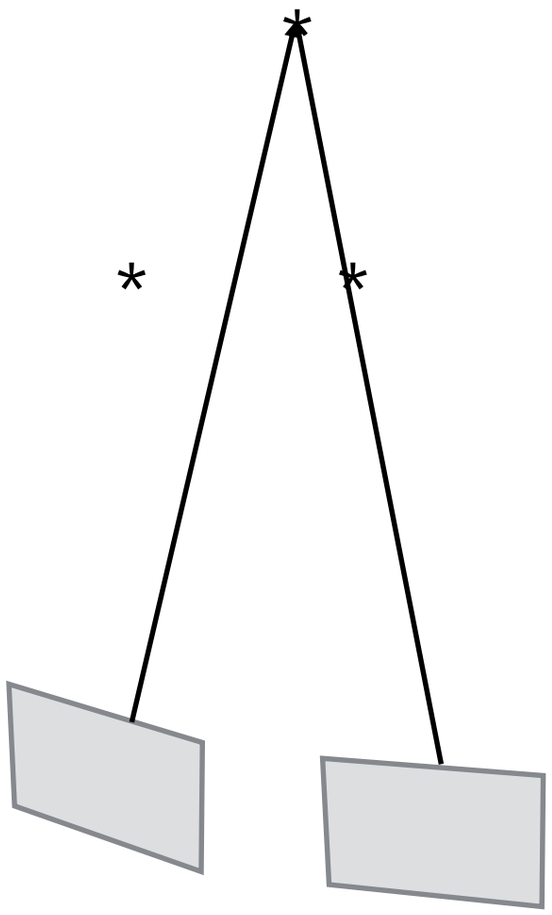


Initialize motion from two views

Initialize structure from two views

Triangulate 3D points

Last Lecture

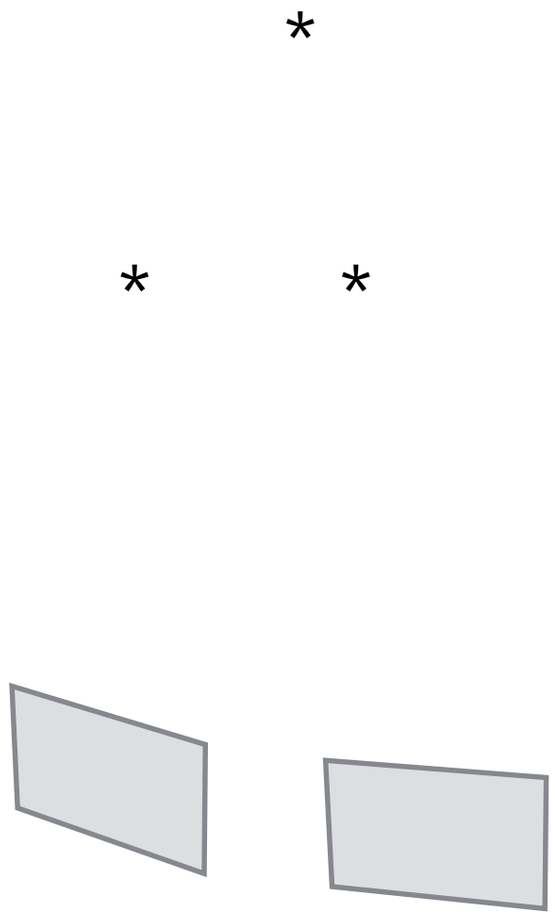


Initialize motion from two views

Initialize structure from two views

Triangulate 3D points

Last Lecture



Initialize motion from two views

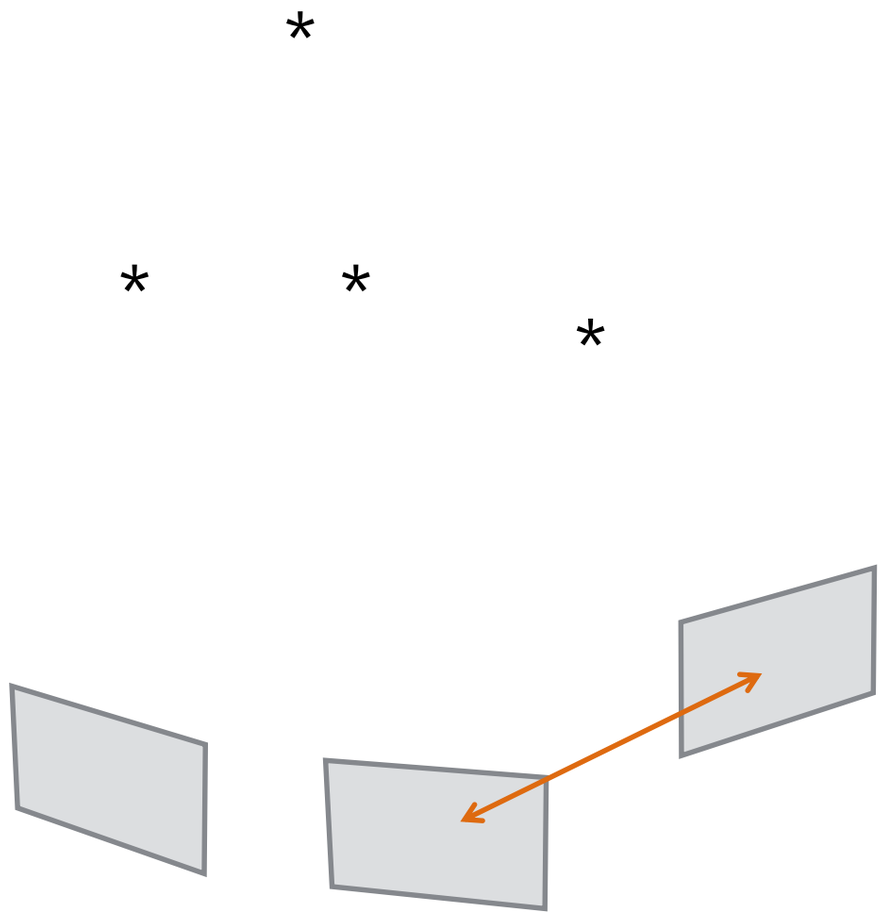
Initialize structure from two views

Last Lecture

Initialize motion from two views

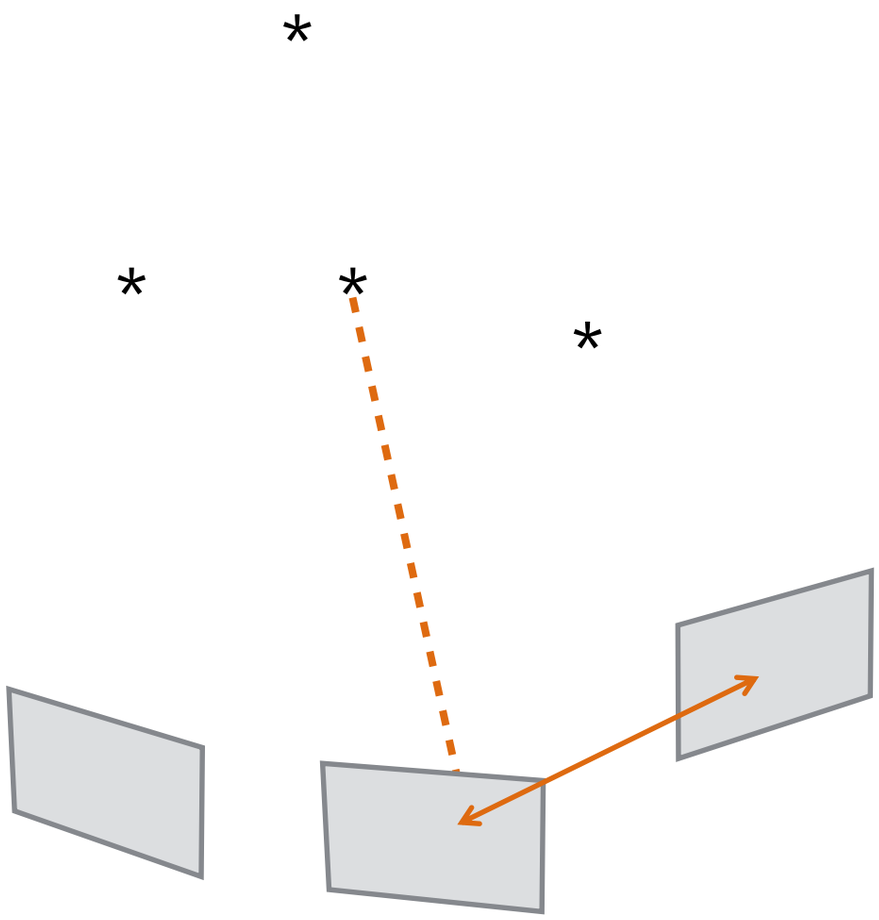
Initialize structure from two views

Extend motion



Match features

Last Lecture



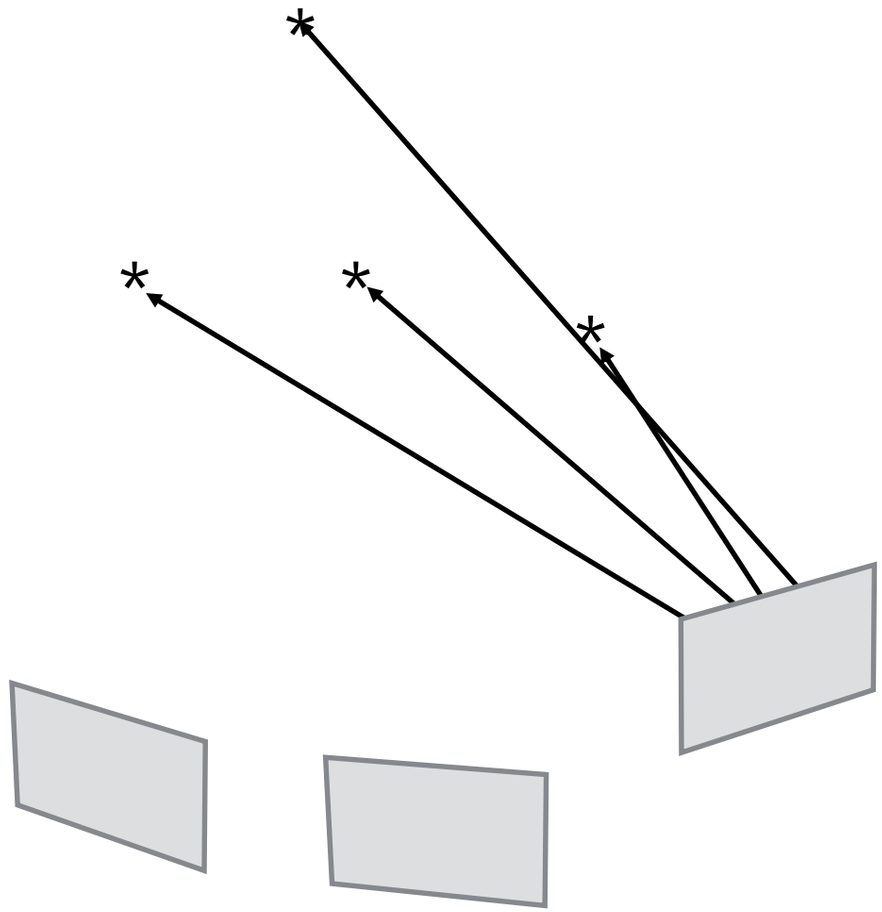
Initialize motion from two views

Initialize structure from two views

Extend motion

Transfer matches to 3D

Last Lecture



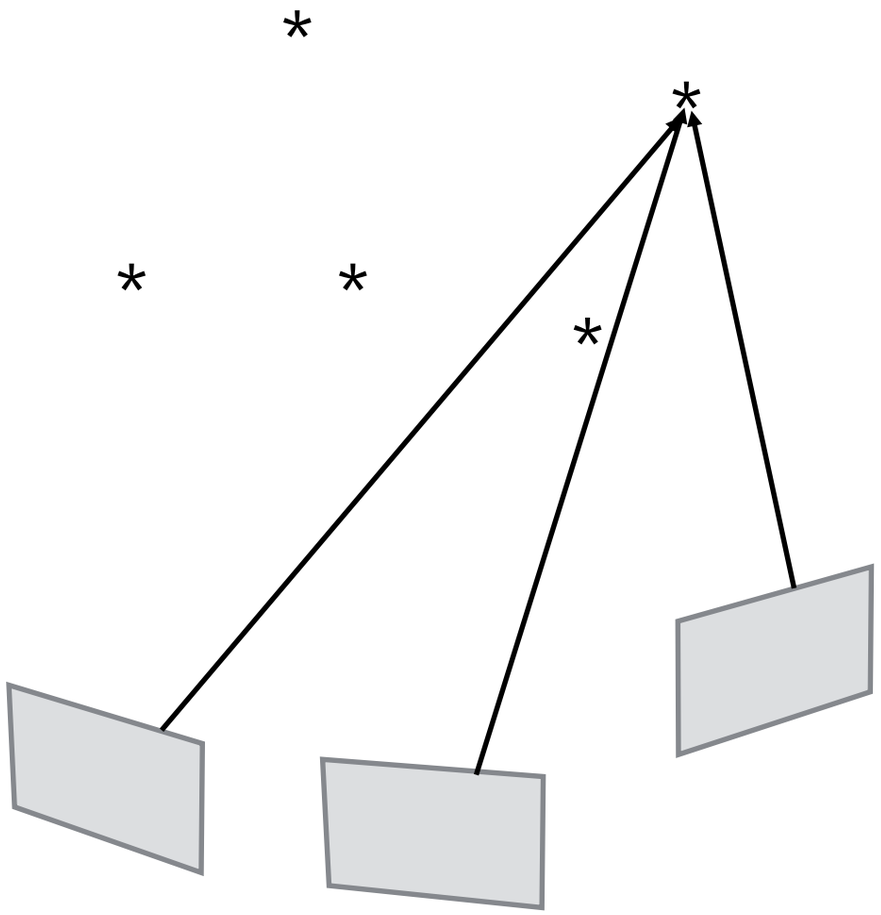
Initialize motion from two views

Initialize structure from two views

Extend motion

Camera pose for third camera

Last Lecture



Initialize motion from two views

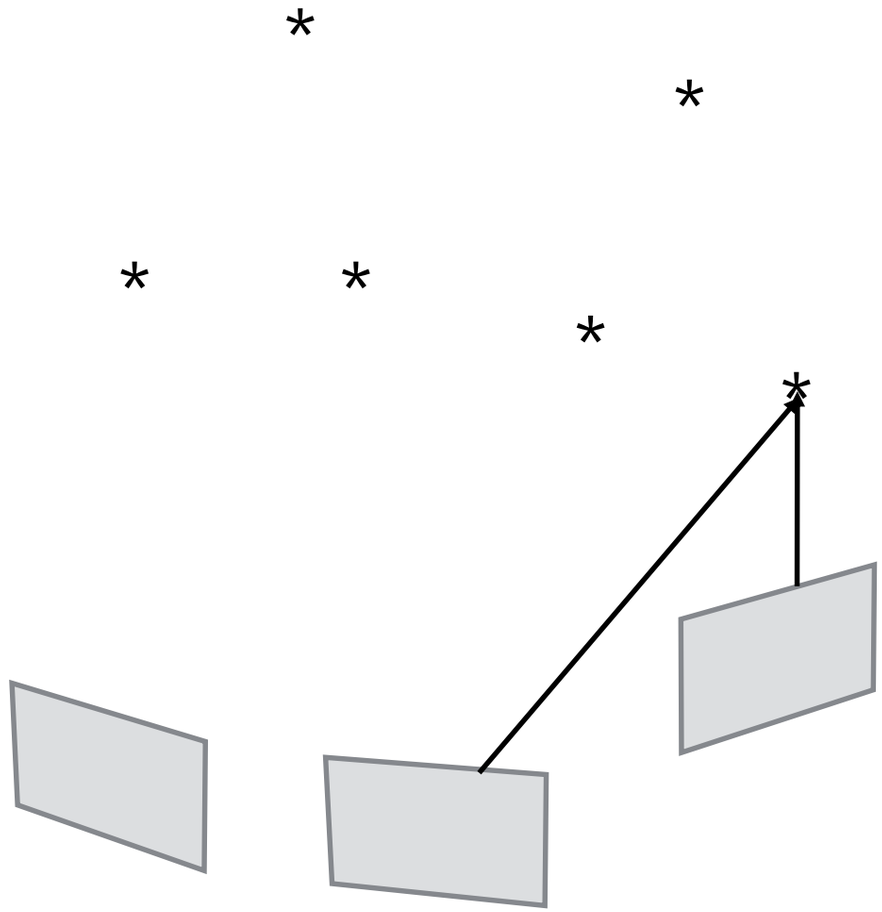
Initialize structure from two views

Extend motion

Extend structure

Triangulate points

Last Lecture



Initialize motion from two views

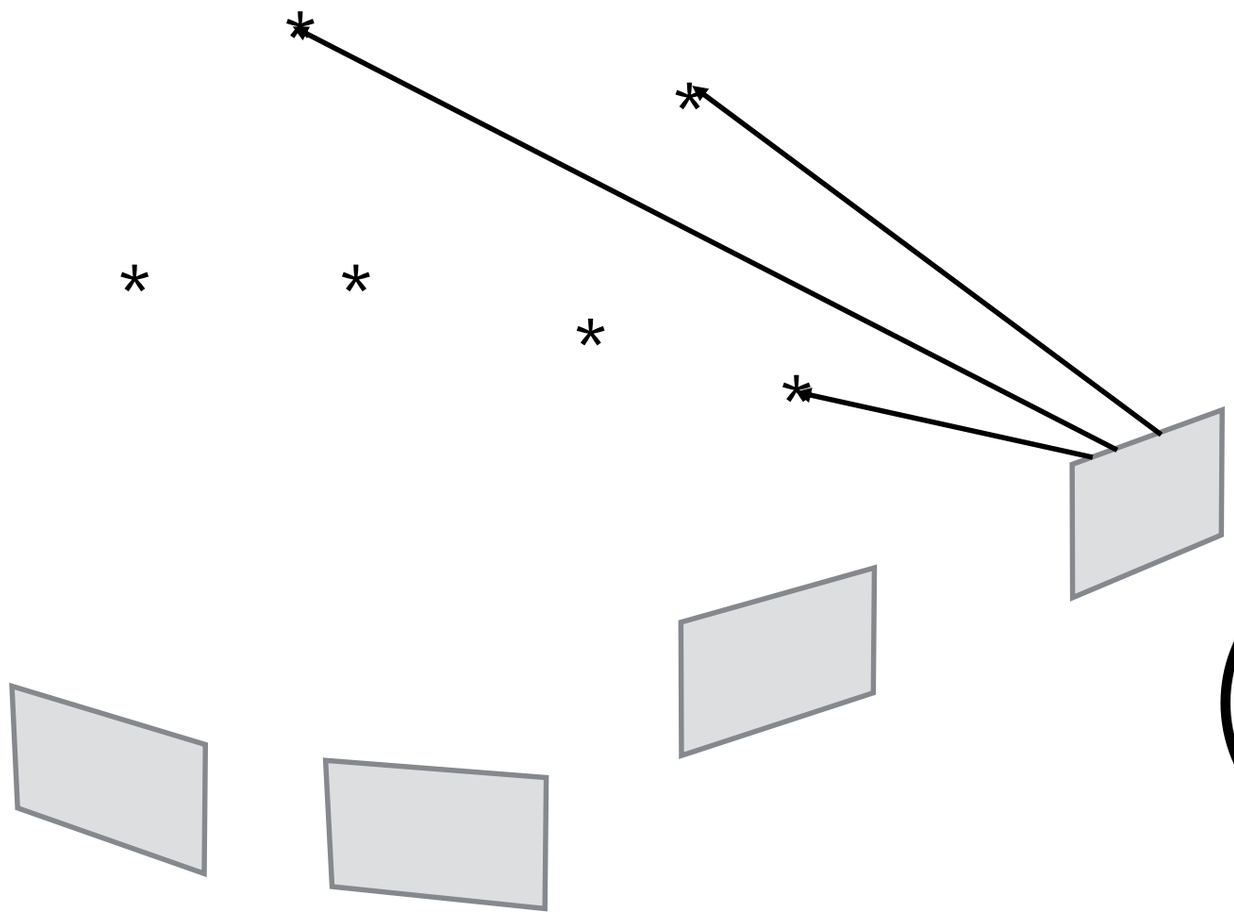
Initialize structure from two views

Extend motion

Extend structure

Triangulate points

Last Lecture



Initialize motion from two views

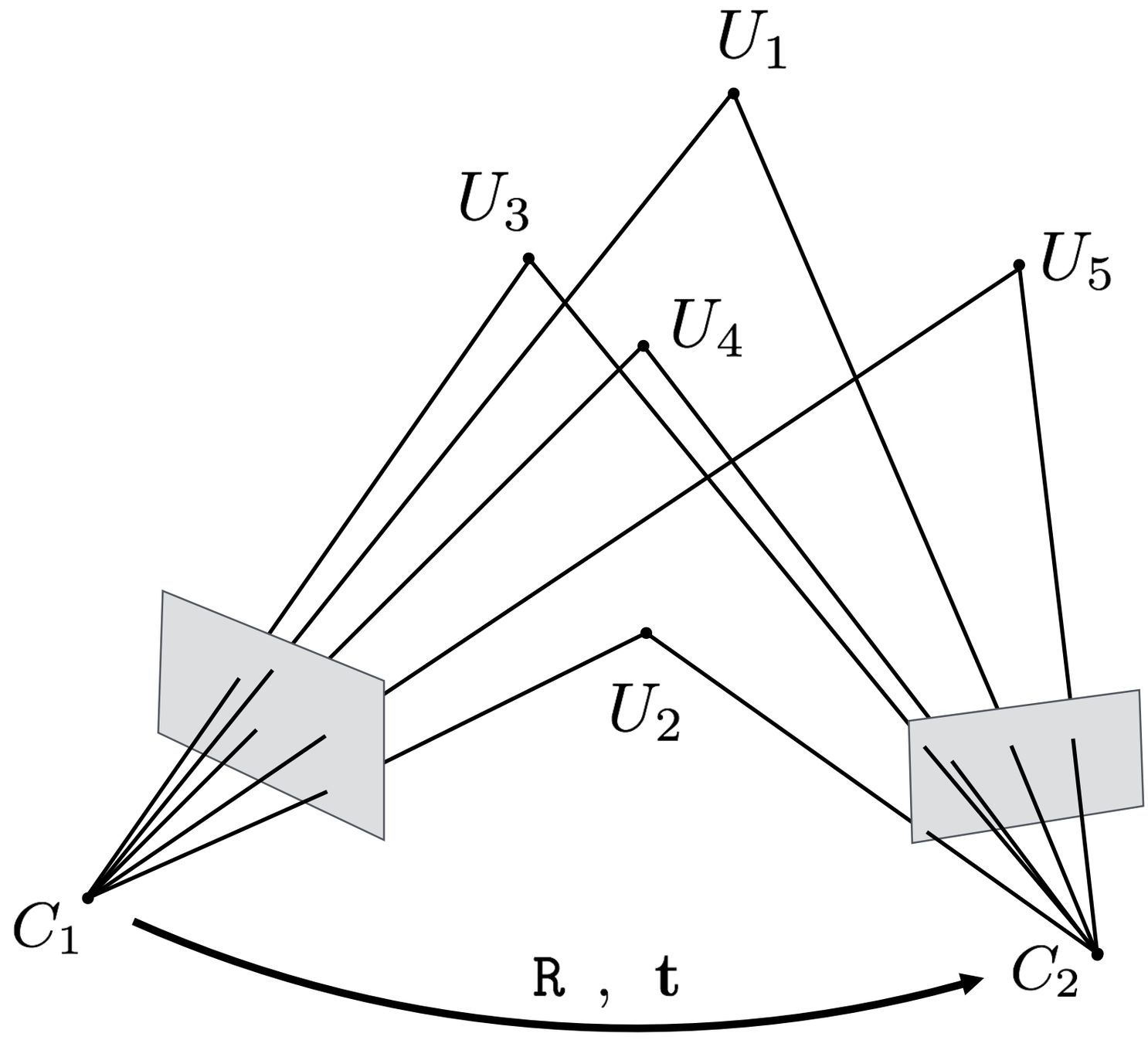
Initialize structure from two views

Extend motion

Extend structure

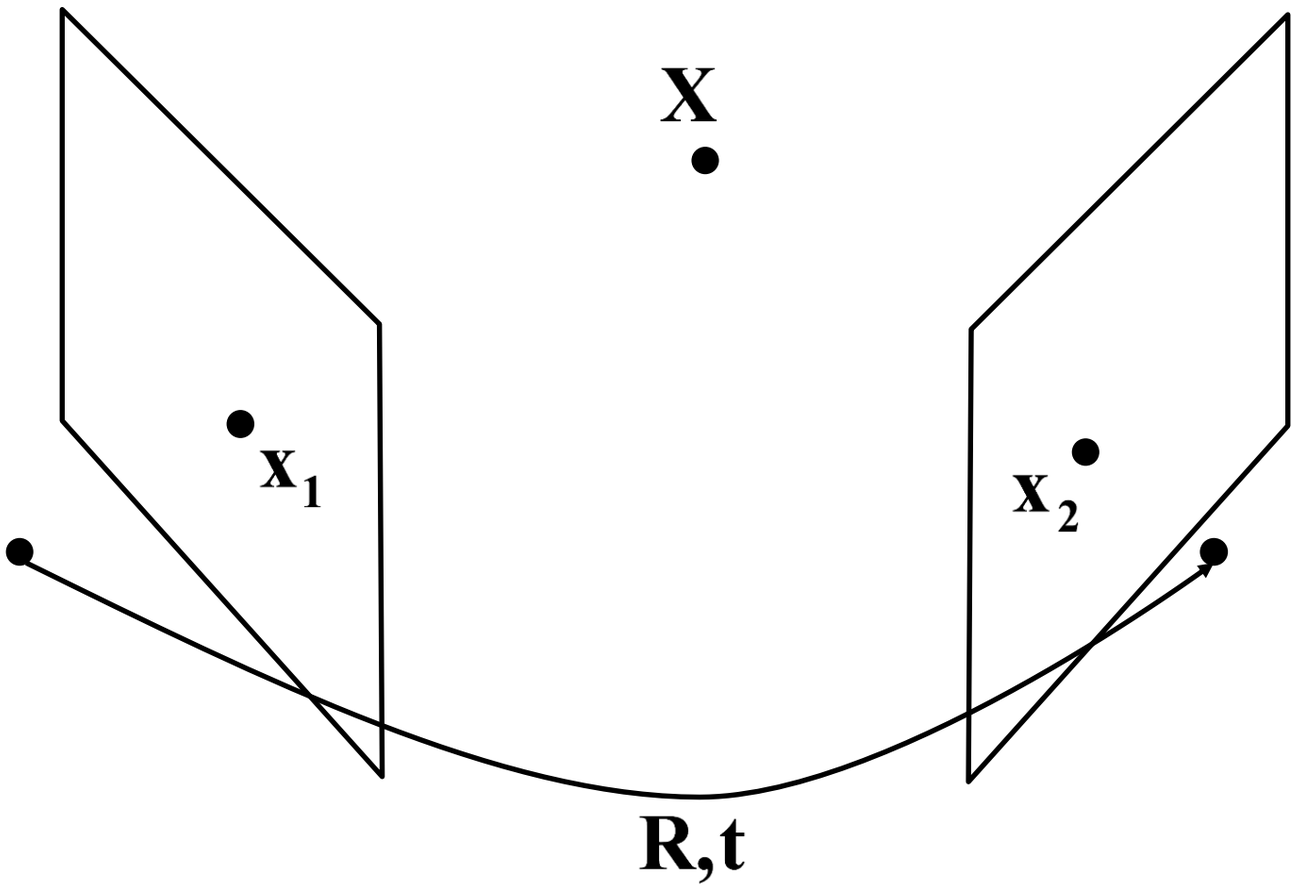
Camera pose for fourth image

Last Lecture



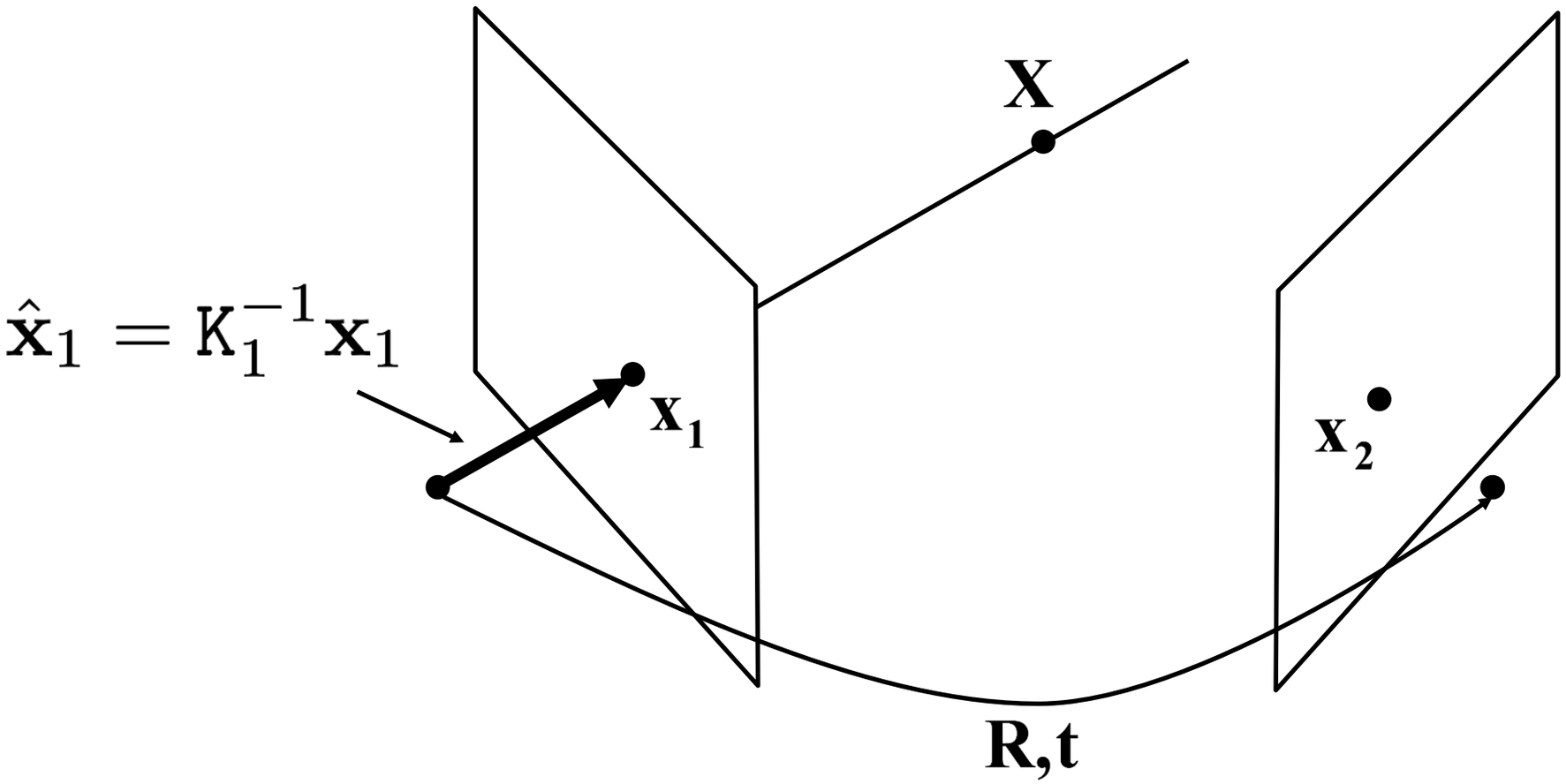
Relative pose estimation

Last Lecture



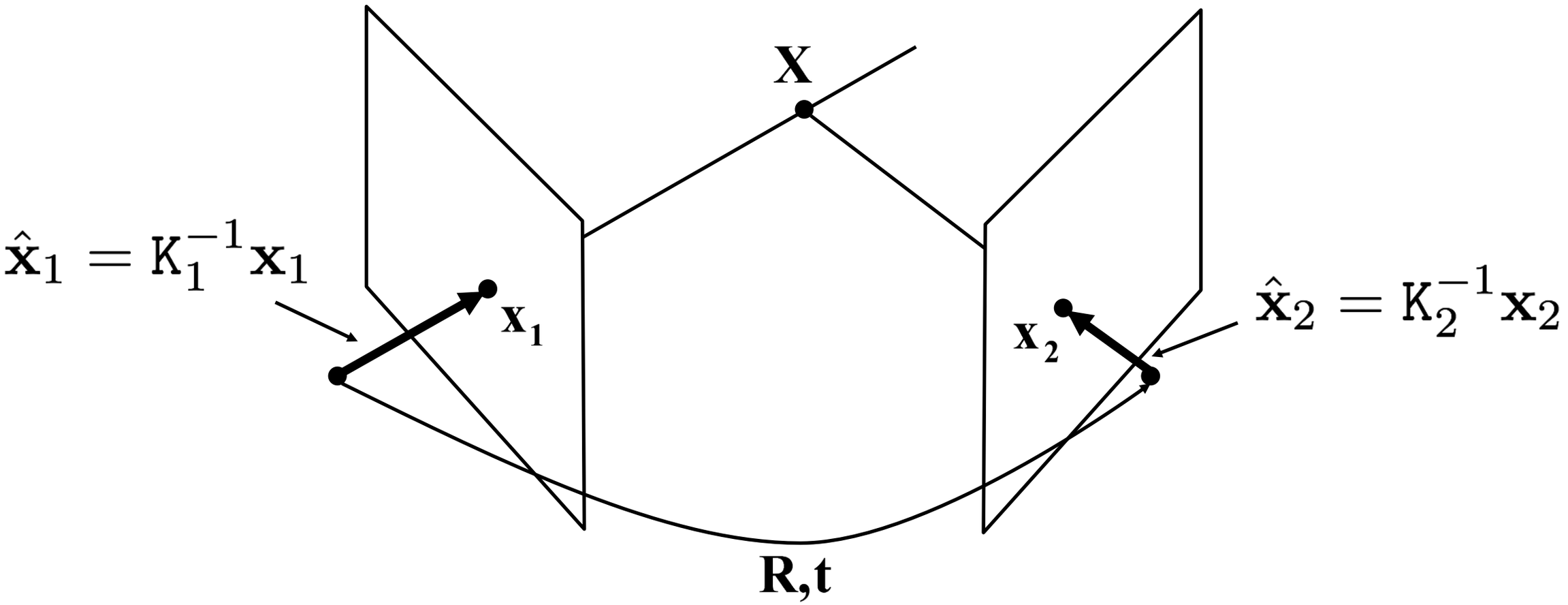
Epipolar geometry

Last Lecture



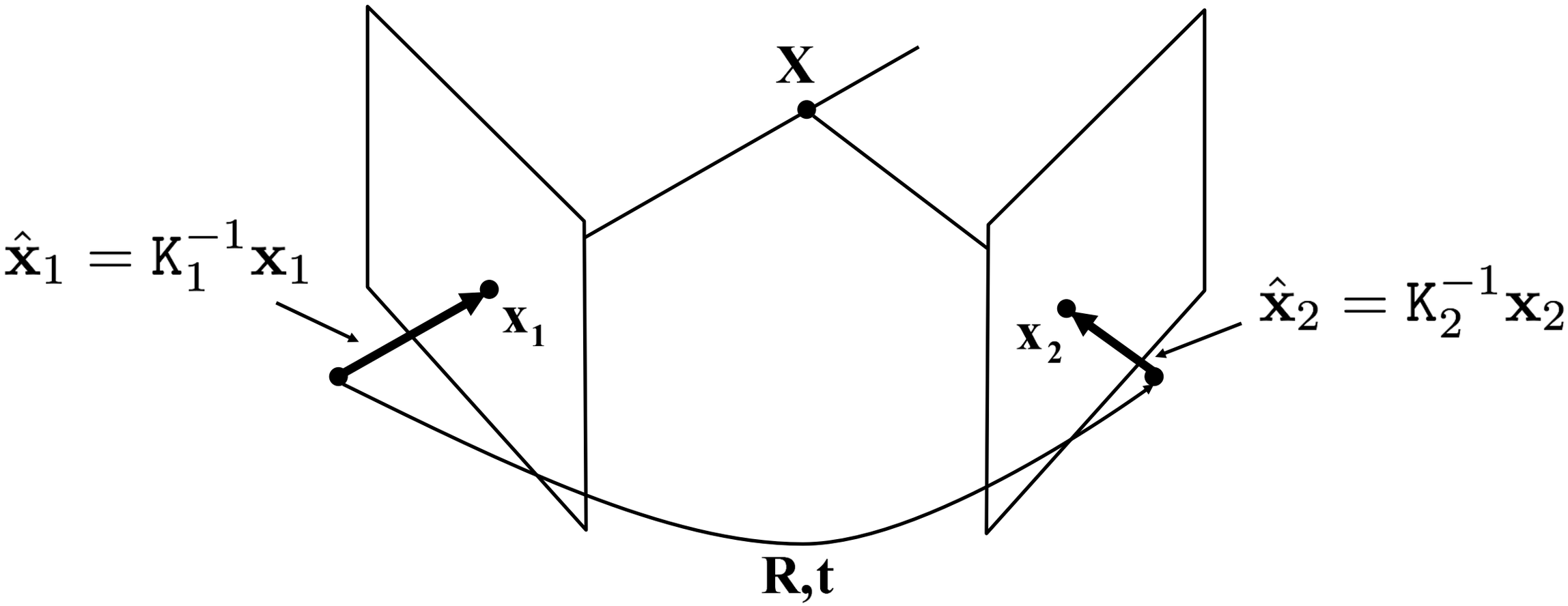
Epipolar geometry

Last Lecture



Epipolar geometry

Last Lecture



epipolar constraint: $0 = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$

Epipolar geometry

Last Lecture

epipolar constraint: $0 = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$

Essential and Fundamental matrices

Last Lecture

epipolar constraint: $0 = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$

Essential matrix: $0 = \hat{\mathbf{x}}_2^T \mathbf{E} \hat{\mathbf{x}}_1$

Essential and Fundamental matrices

Last Lecture

epipolar constraint: $0 = \hat{\mathbf{x}}_2^T [\mathbf{t}]_\times \mathbf{R} \hat{\mathbf{x}}_1$

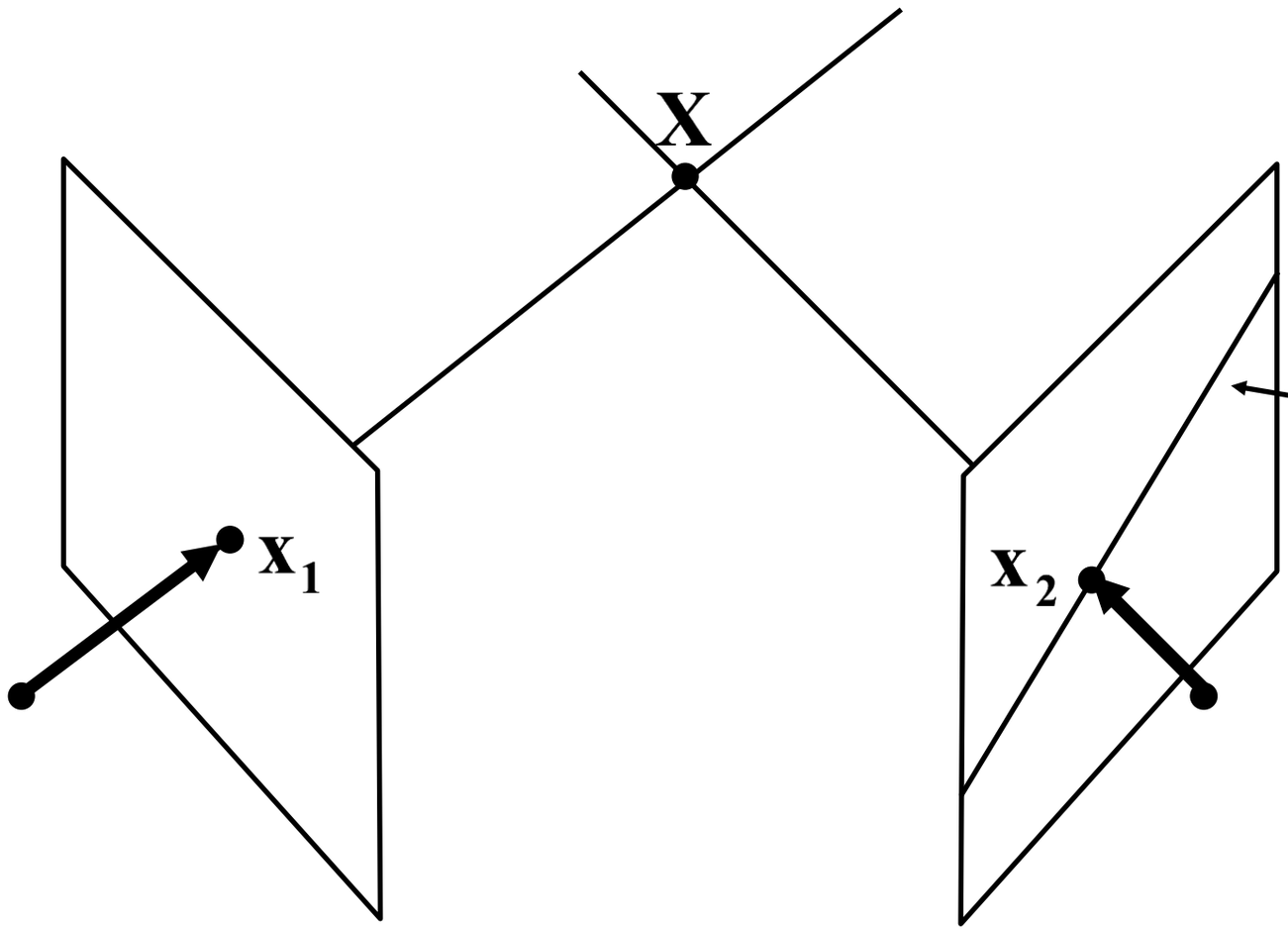
Essential matrix: $0 = \hat{\mathbf{x}}_2^T \mathbf{E} \hat{\mathbf{x}}_1$

Fundamental matrix: $0 = \mathbf{x}_2^T \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \mathbf{x}_1$

$0 = \mathbf{x}_2^T \mathbf{F} \mathbf{x}_1$

Essential and Fundamental matrices

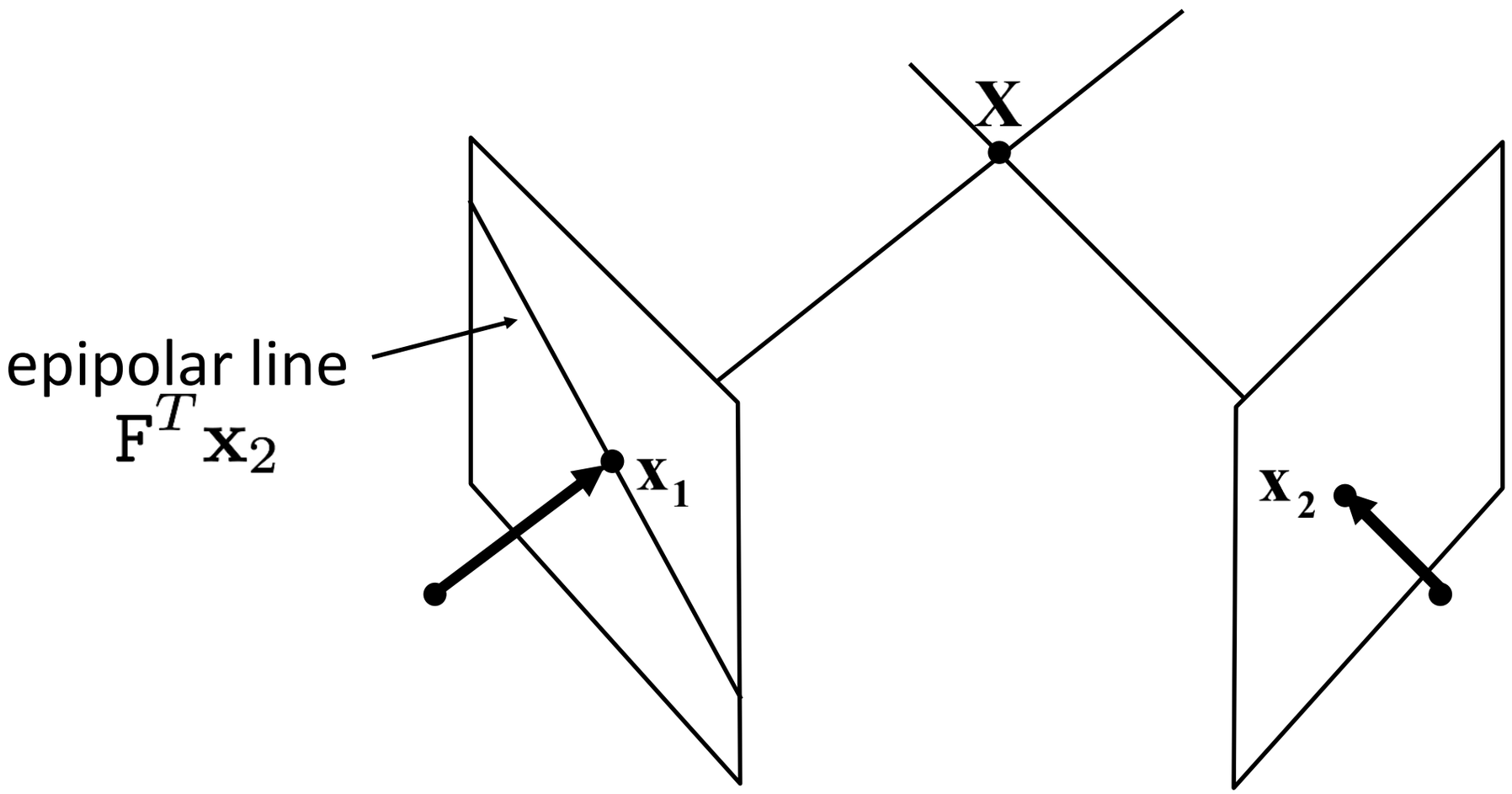
Last Lecture



$$0 = \mathbf{x}_2^T \mathbf{F} \mathbf{x}_1$$

Geometric Interpretation

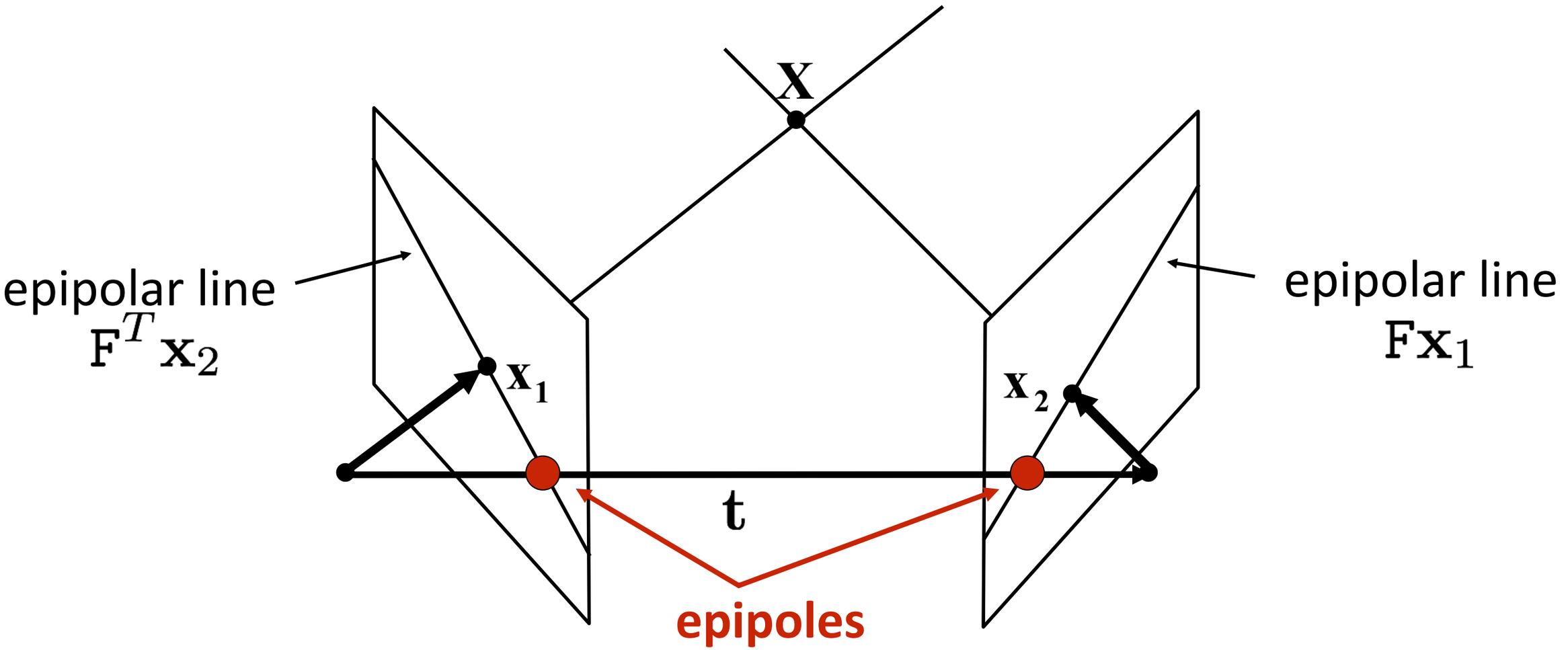
Last Lecture



$$0 = \mathbf{x}_2^T \mathbf{F} \mathbf{x}_1$$

Geometric Interpretation

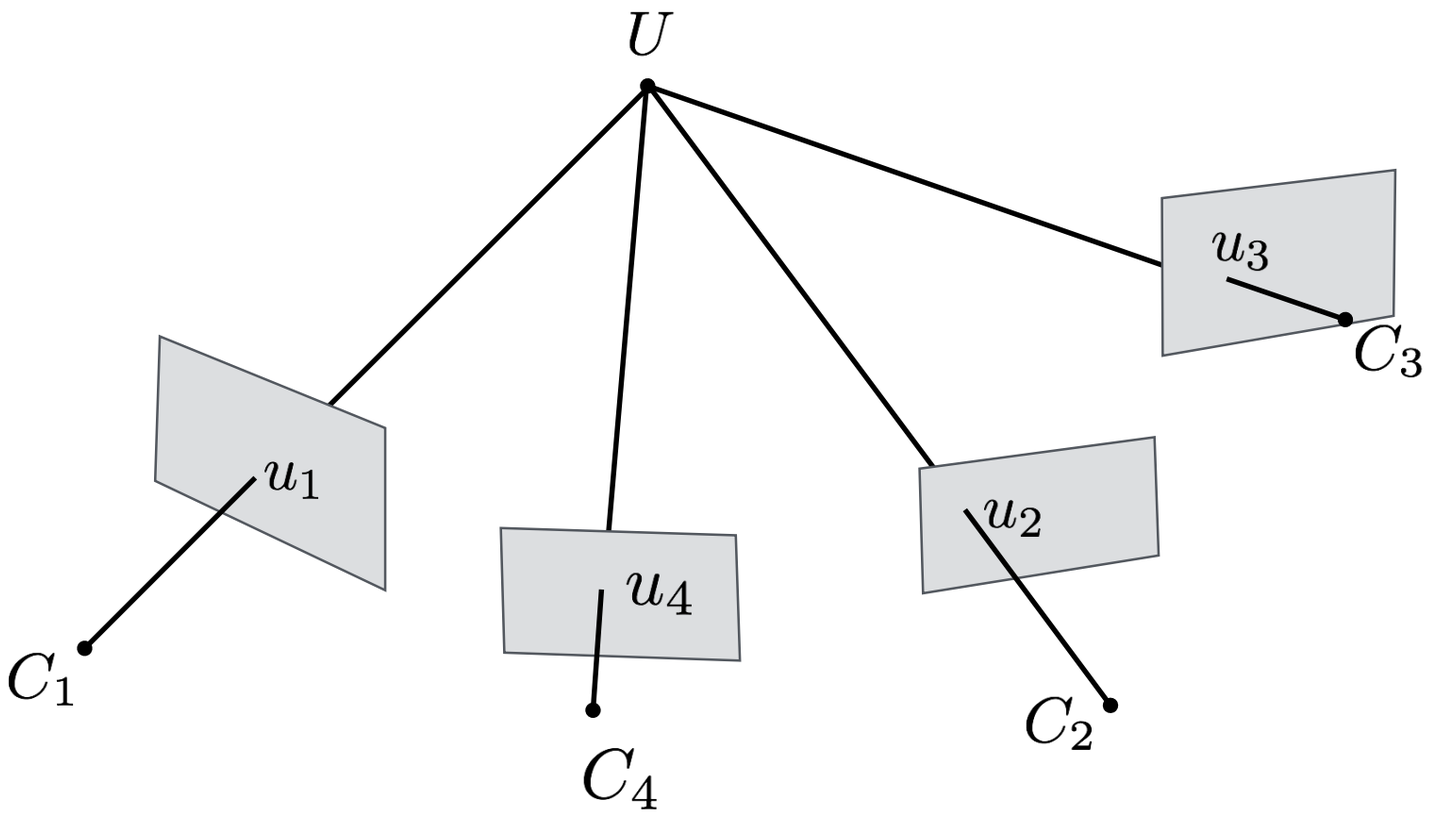
Last Lecture



$$0 = \mathbf{x}_2^T \mathbf{F} \mathbf{x}_1$$

Geometric Interpretation

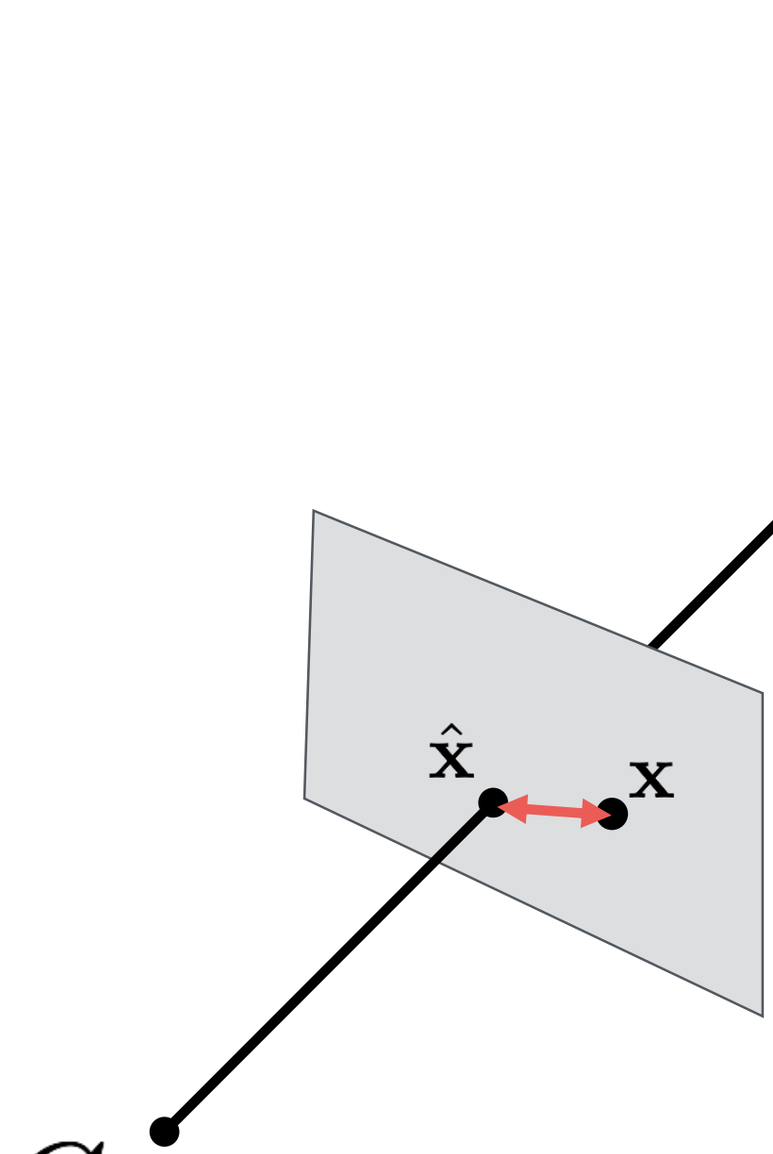
Last Lecture



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

Triangulation

Last Lecture



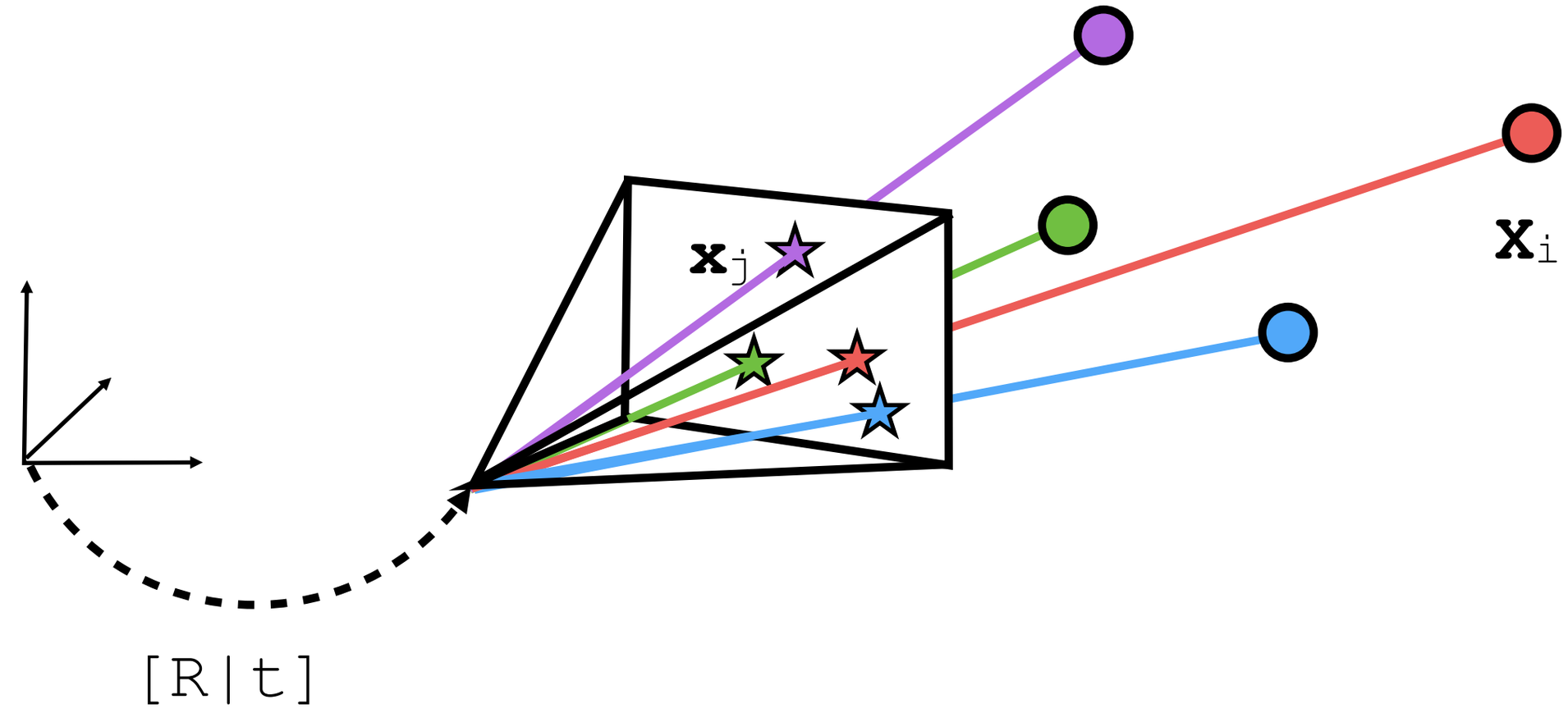
$$\hat{\mathbf{x}} = \begin{pmatrix} \frac{\mathbf{P}_1\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \\ \frac{\mathbf{P}_2\mathbf{X}}{\mathbf{P}_3\mathbf{X}} \end{pmatrix}$$

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

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

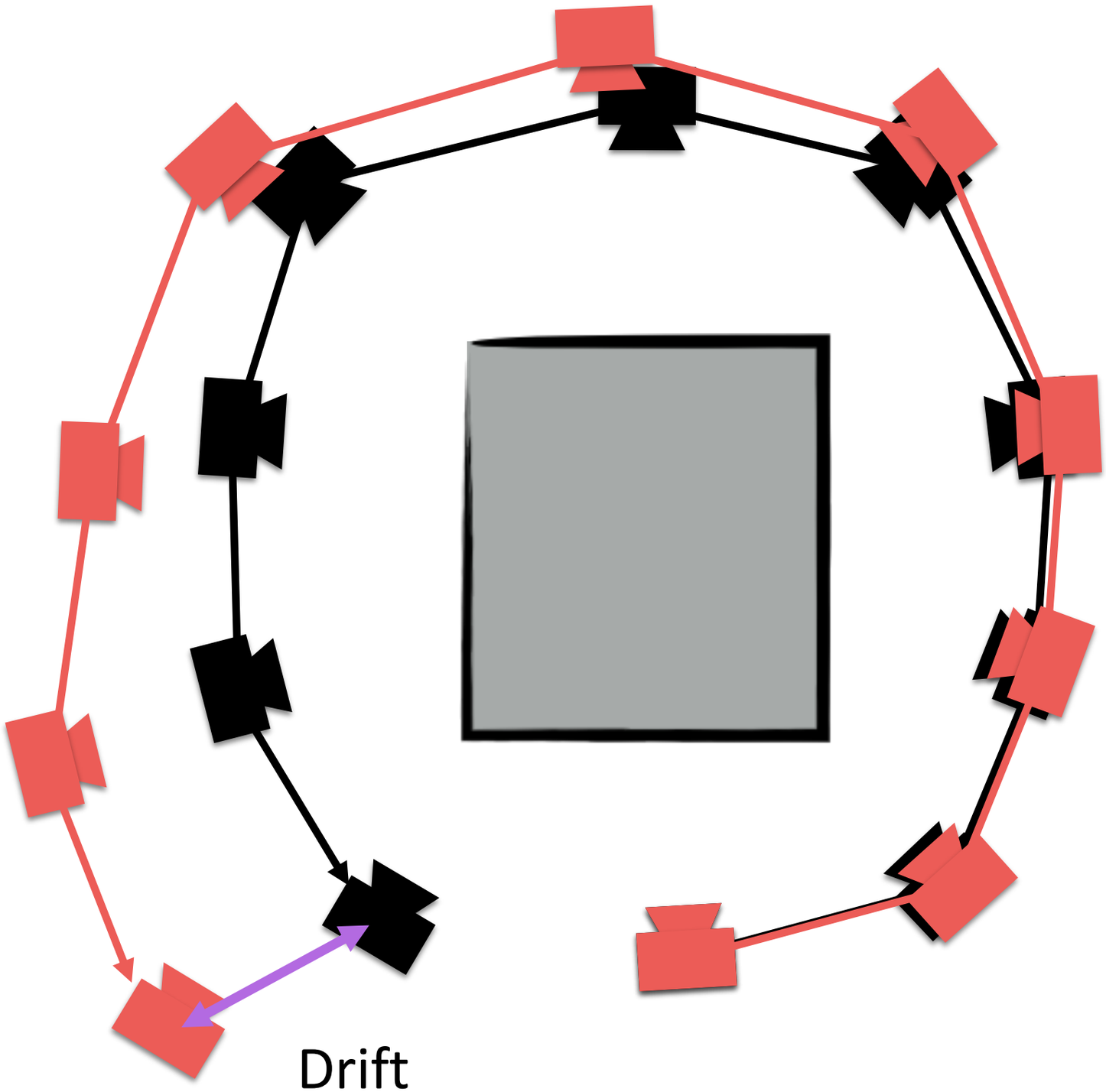
Minimizing reprojection error

Last Lecture

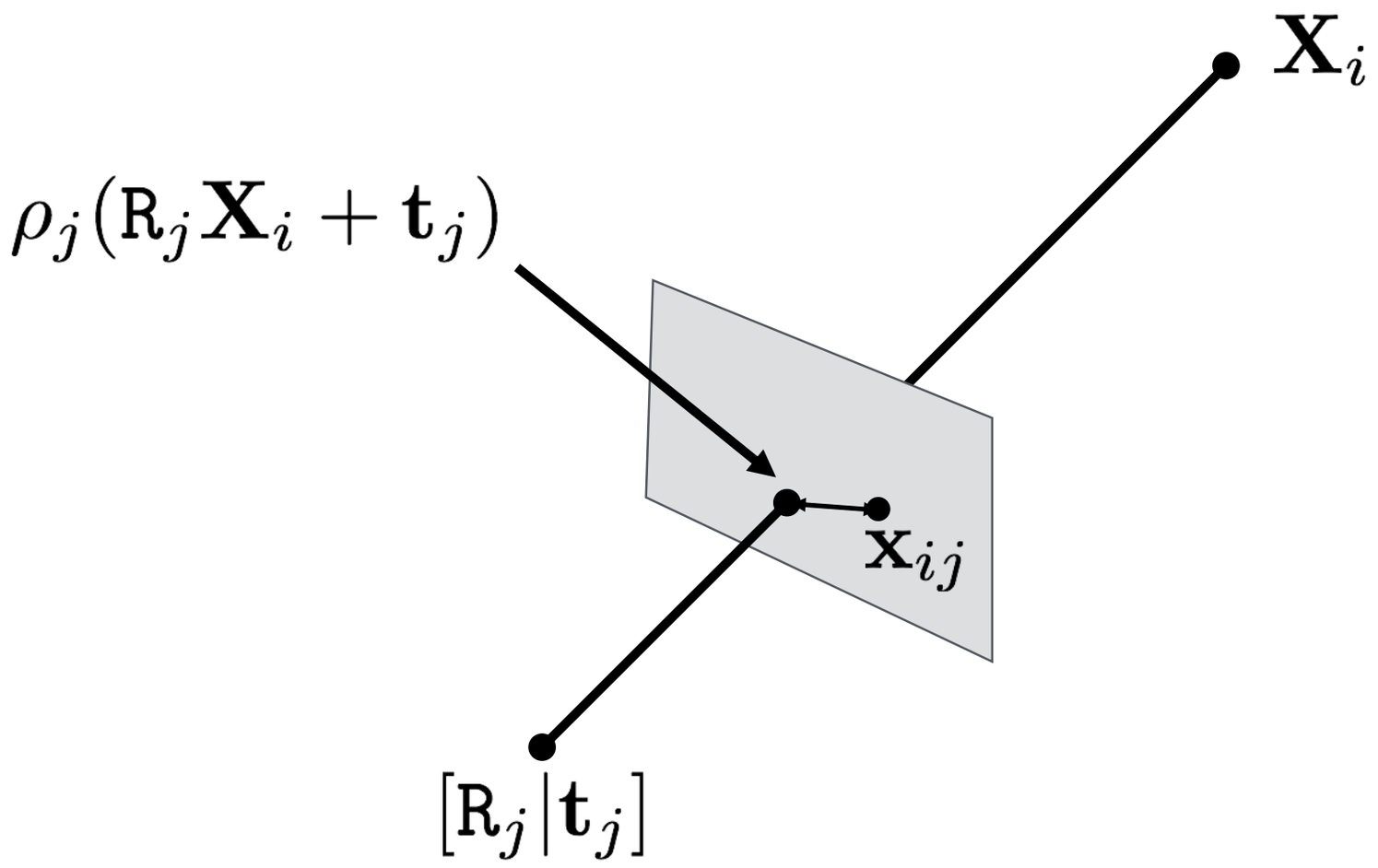


Absolute pose estimation

Last Lecture



Last Lecture



$$\underset{\text{camera poses, 3D points}}{\operatorname{argmin}} \sum_i \sum_j \Delta_{ij} ||\mathbf{x}_{ij} - \rho_j(\mathrm{R}_j\mathbf{X}_i + \mathbf{t}_j)||^2$$

point i visible in image j?

Bundle adjustment

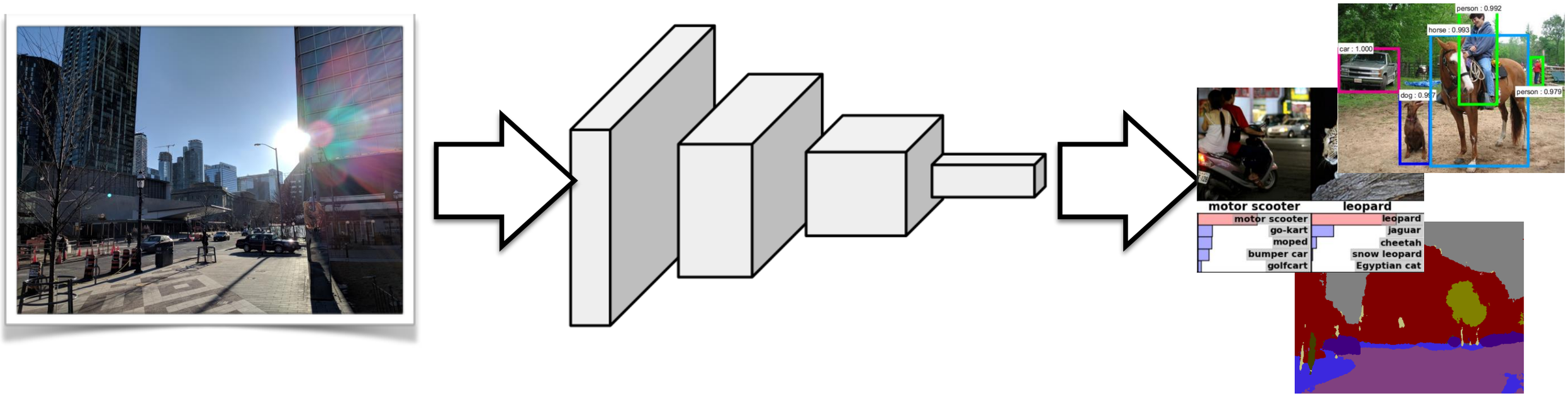
Last Lecture

Bundle adjustment

Today

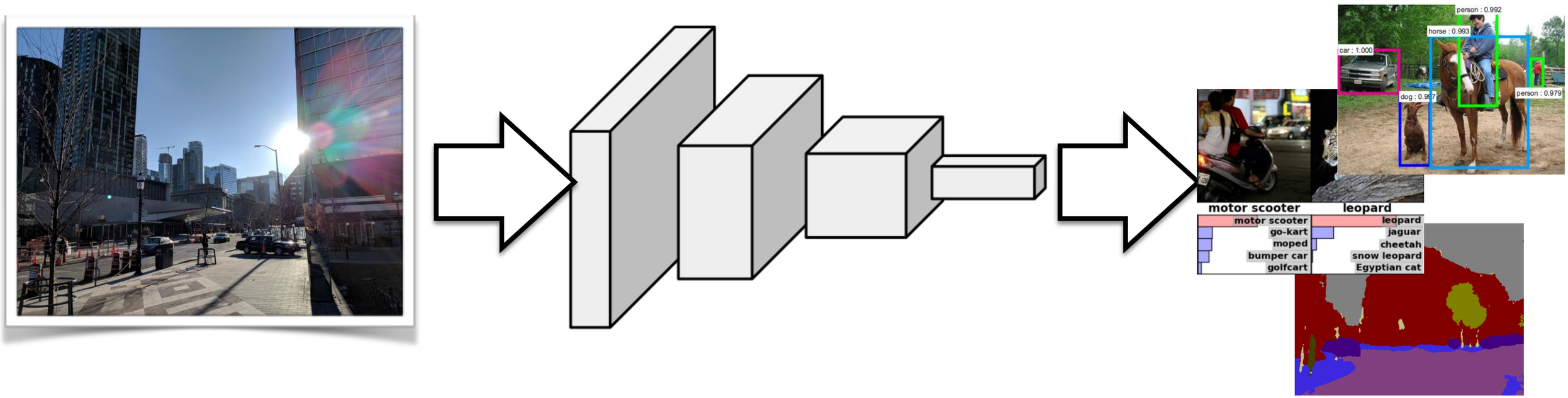
Generative Neural Networks

Recap: CNNs



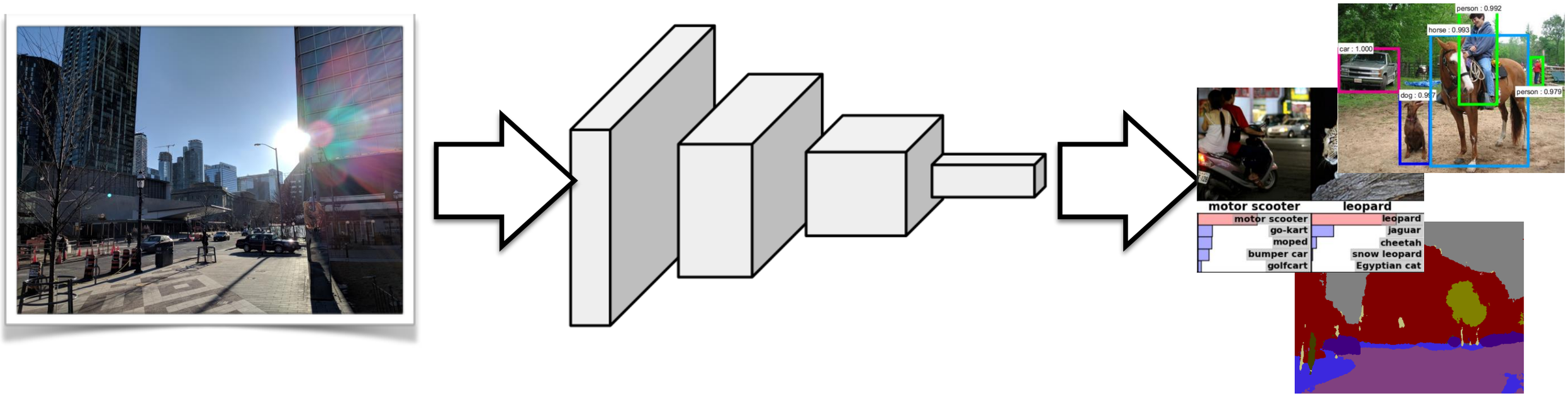
- Convolutional neural networks for classification, segmentation, regression, etc.

Recap: CNNs



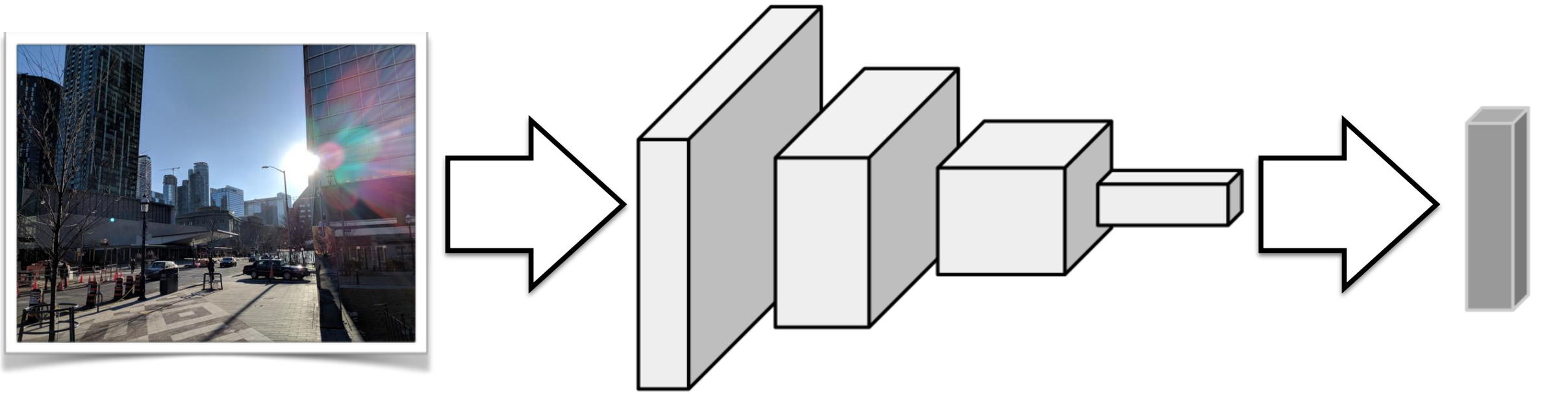
- Convolutional neural networks for classification, segmentation, regression, etc.
- Supervised learning: Training from (image, label) pairs

Recap: CNNs



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

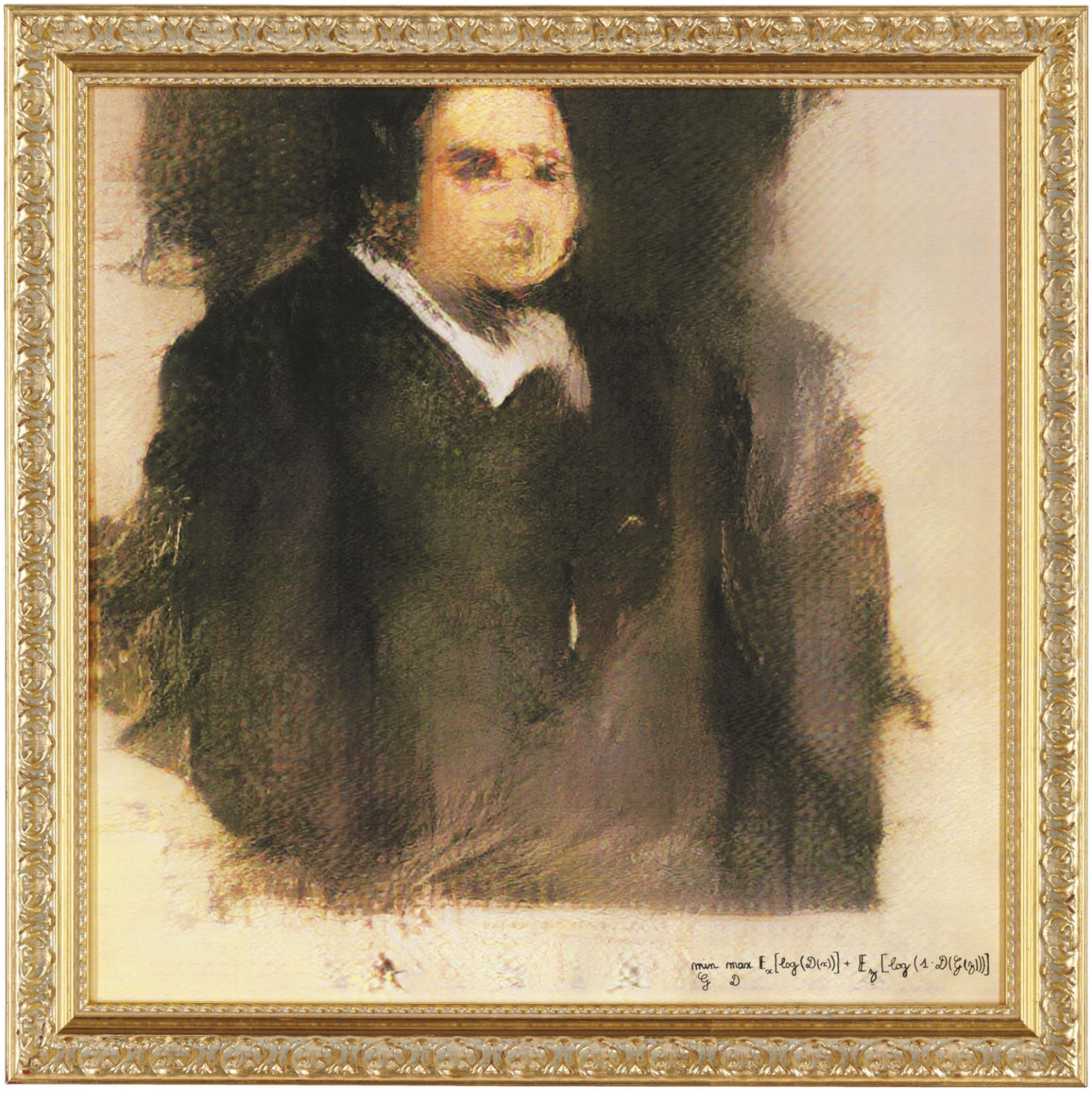
Generative Neural Networks



image

generator / decoder

output



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

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]

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]

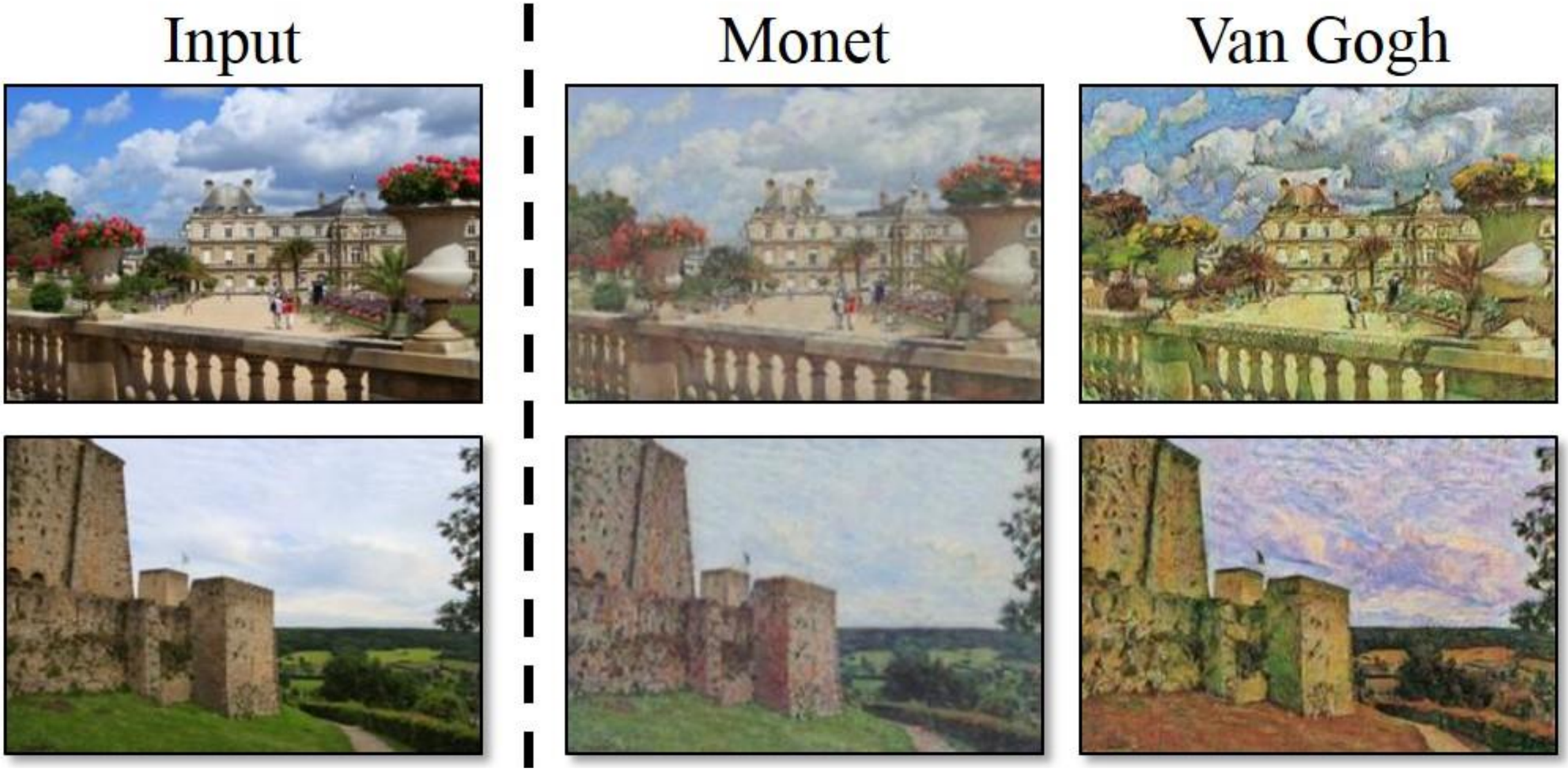
Image-to-Image Translation

[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]

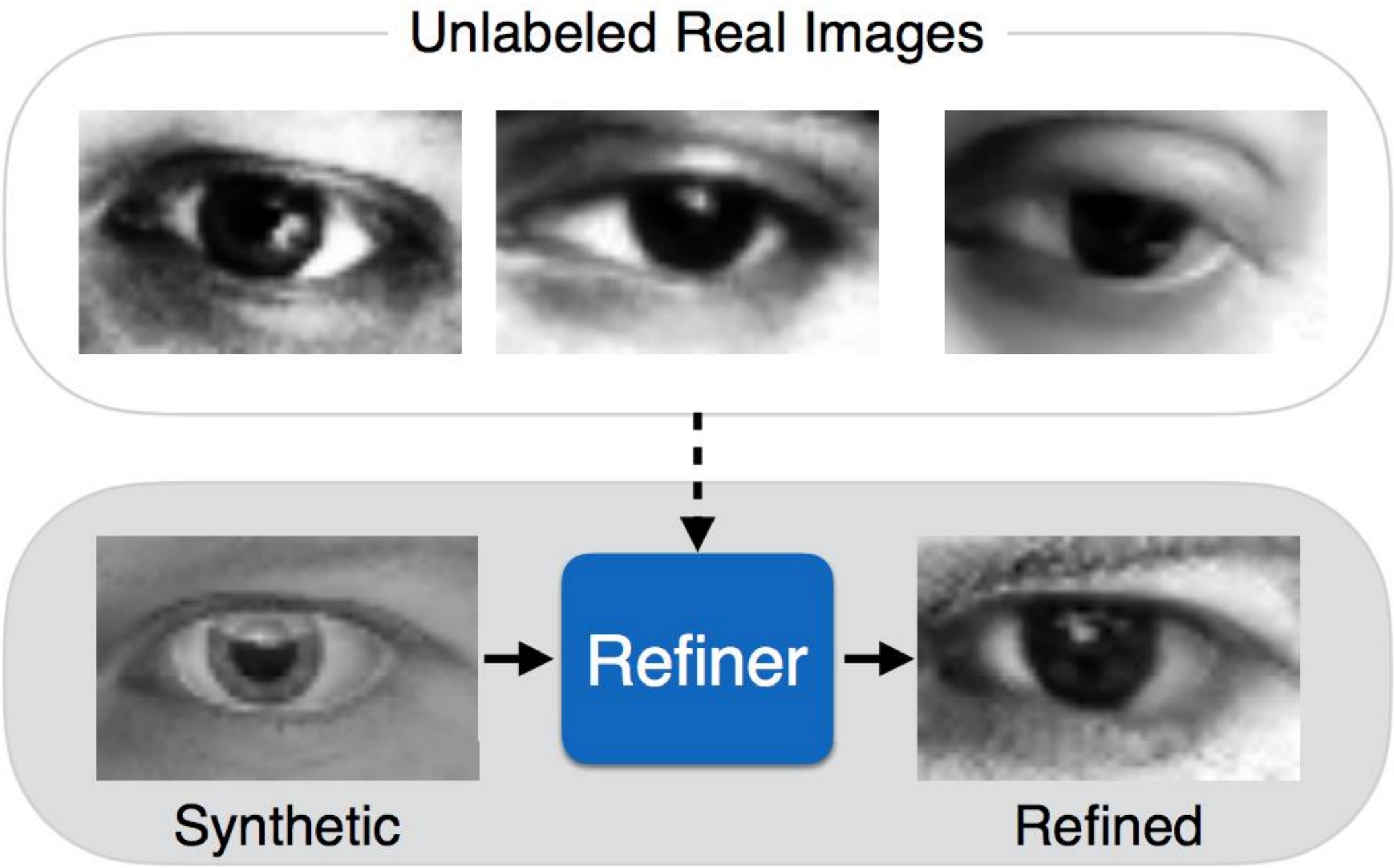
Training Data Generation

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]

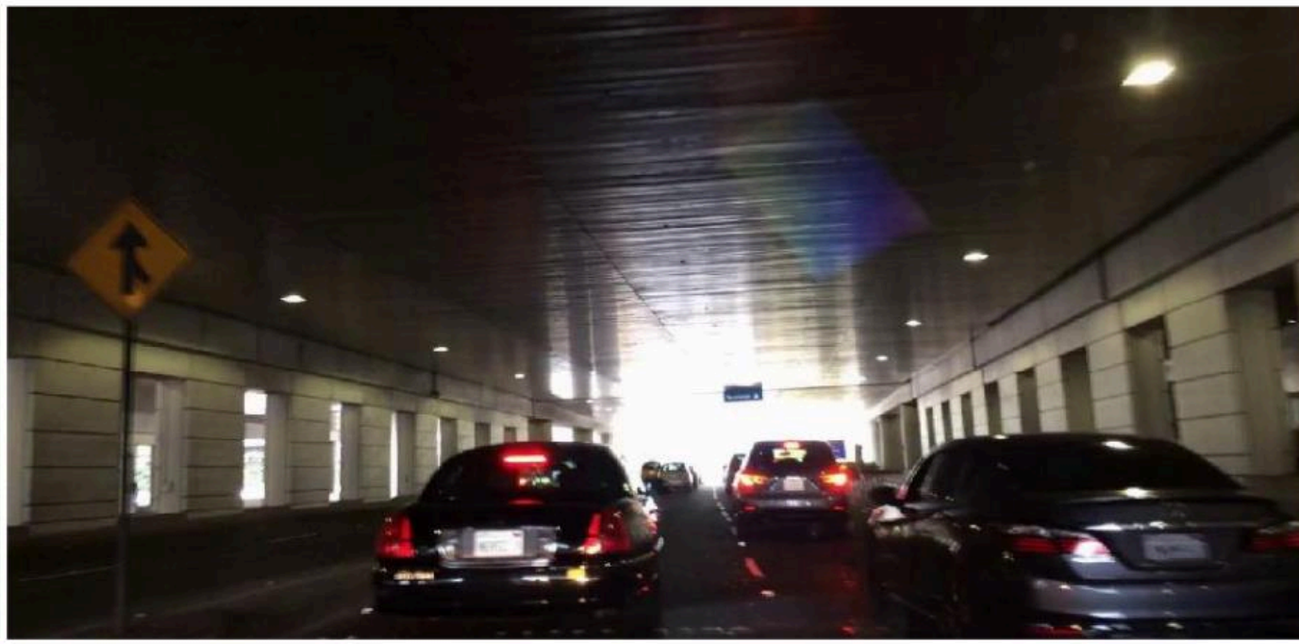
Domain Adaptation

Train

CityScapes (Germany)

Source domain:
Plenty of labelled data

Test



San Francisco

Target domain:
Little to no labelled data

slide credit: Judy Hoffman

Image Generation as Sampling Process

Generate image = pick image from distribution of real images

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

How to Model Distribution of Images?

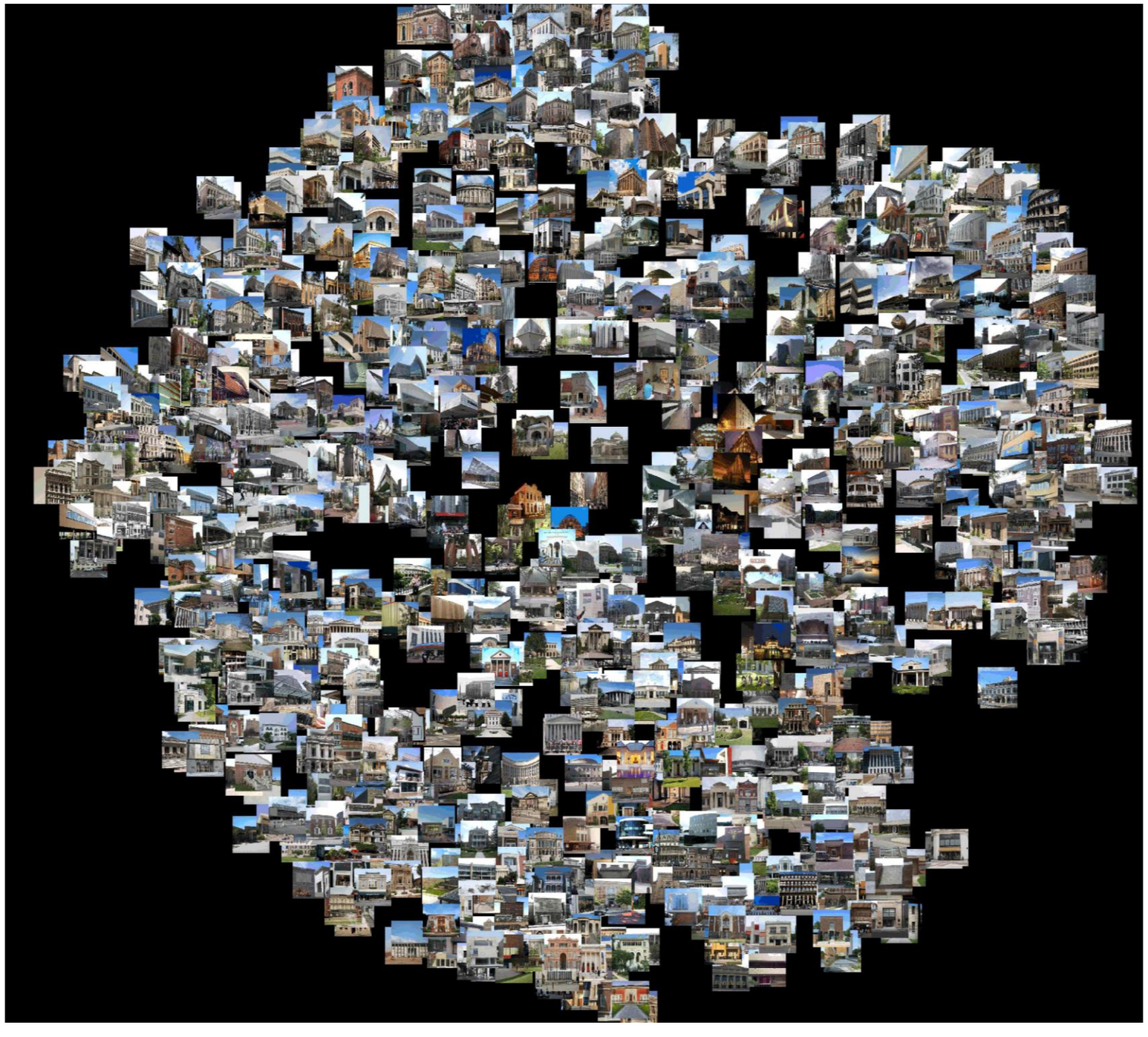
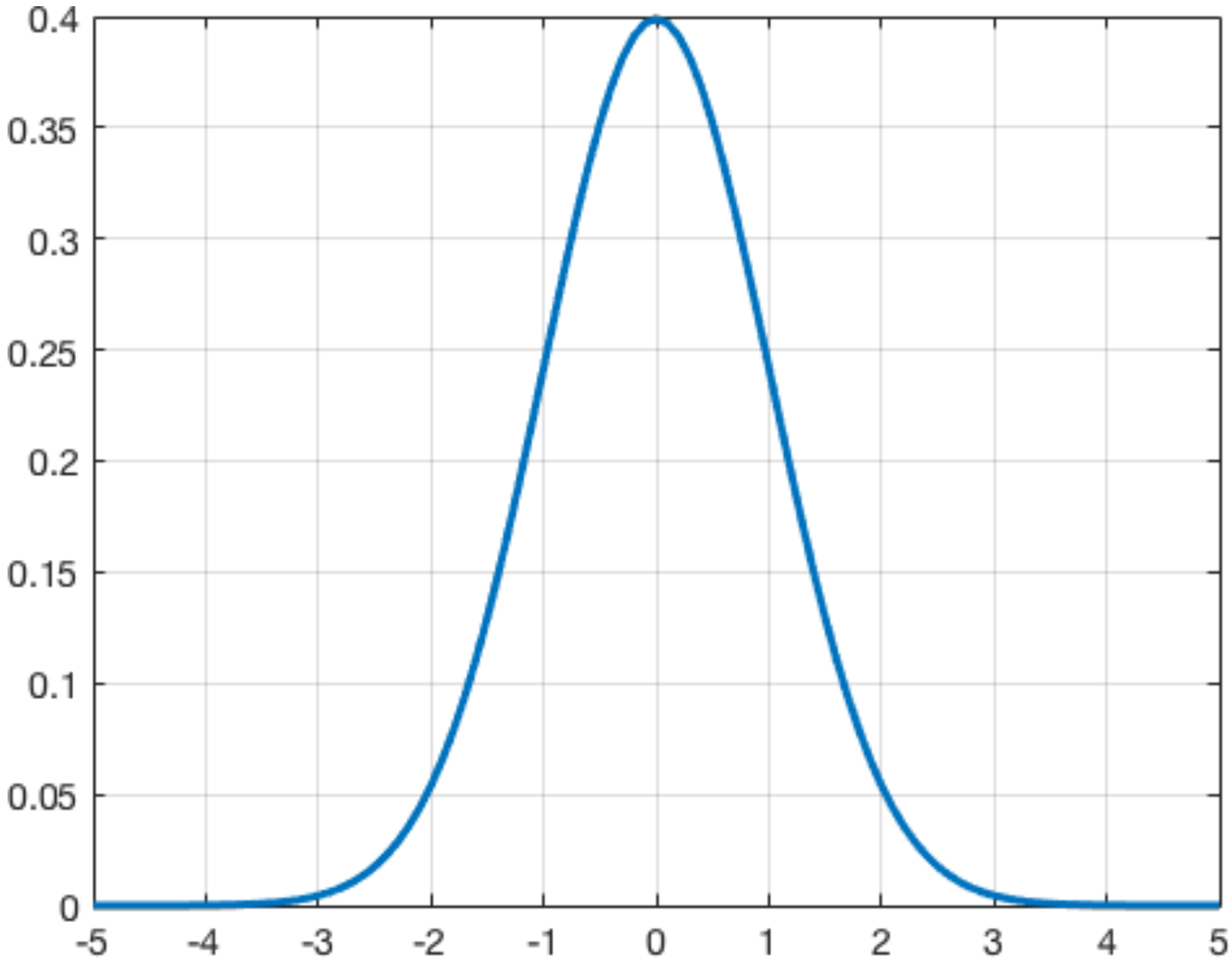


image from http://3drepresentation.stanford.edu/

Generation via Sampling

Example: Sampling from normal distribution



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: $\mathtt{U}$, $\mathtt{V}$ drawn uniformly from $\mathtt{(0,1)}$
- 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:

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

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:

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

Key idea: Sample from simple distribution, map to complex one

Generative Neural Networks

code

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

Generative Neural Networks

code

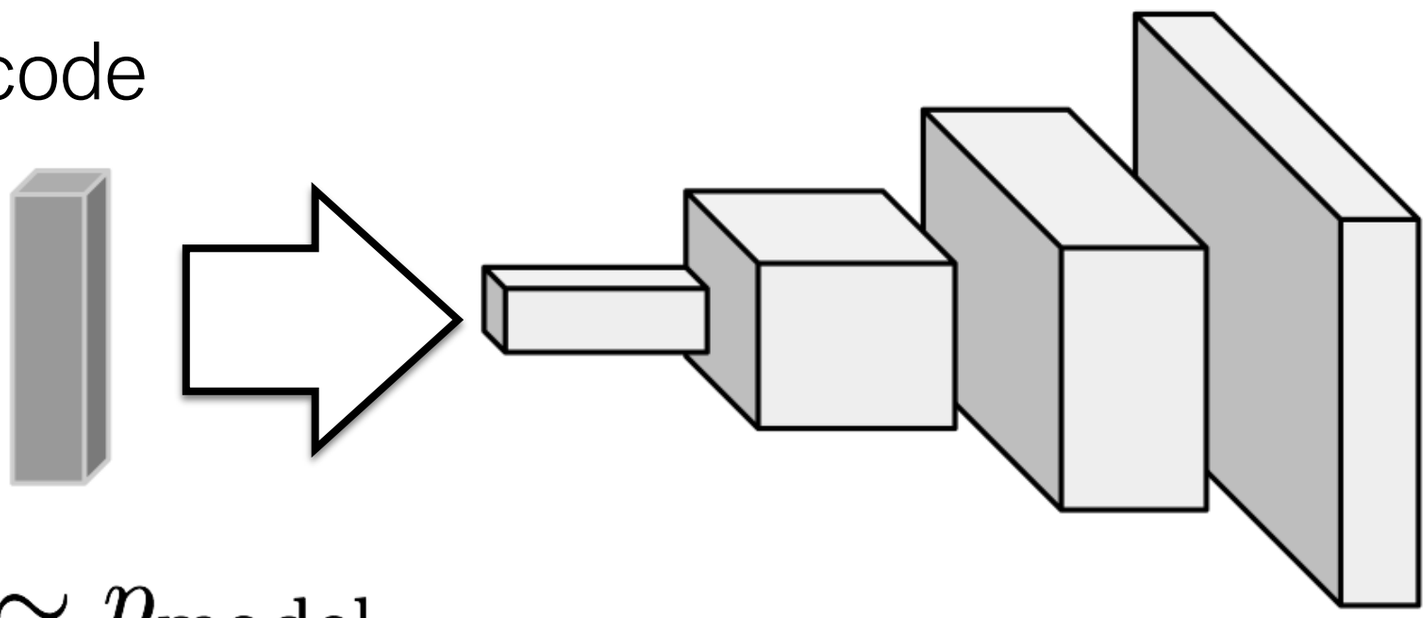
$$\mathbf{z} \sim p_{\text{model}}$$

normal distribution,
uniform distribution,
…

Generative Neural Networks

generator network $\mathbb{G}$

code



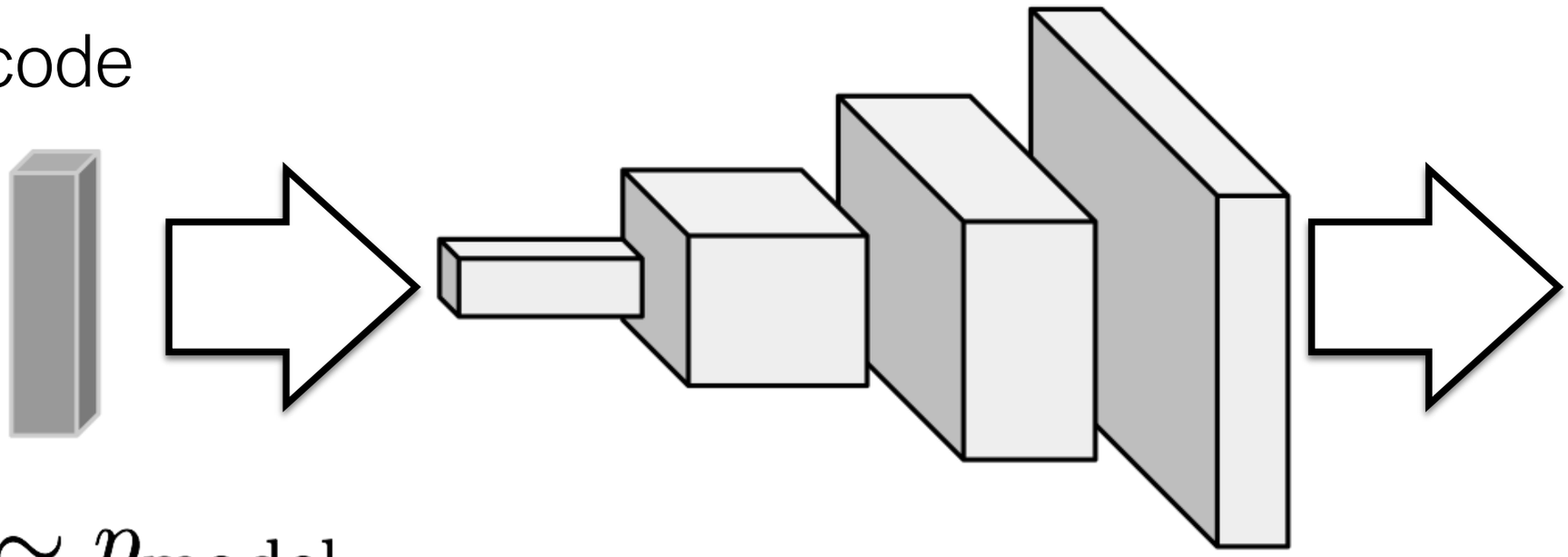
$\mathbf{z} \sim p_{\text{model}}$

normal distribution,
uniform distribution,
…

Generative Neural Networks

generator network $\mathbb{G}$

code

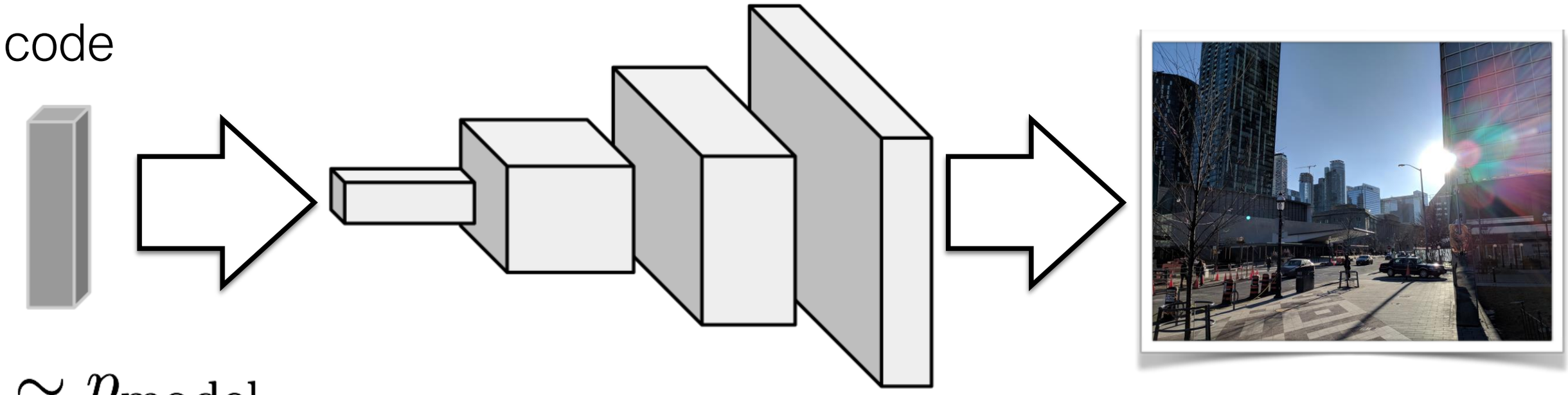


$\mathbf{z} \sim p_{\text{model}}$

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

normal distribution,
uniform distribution,
…

Generative Neural Networks



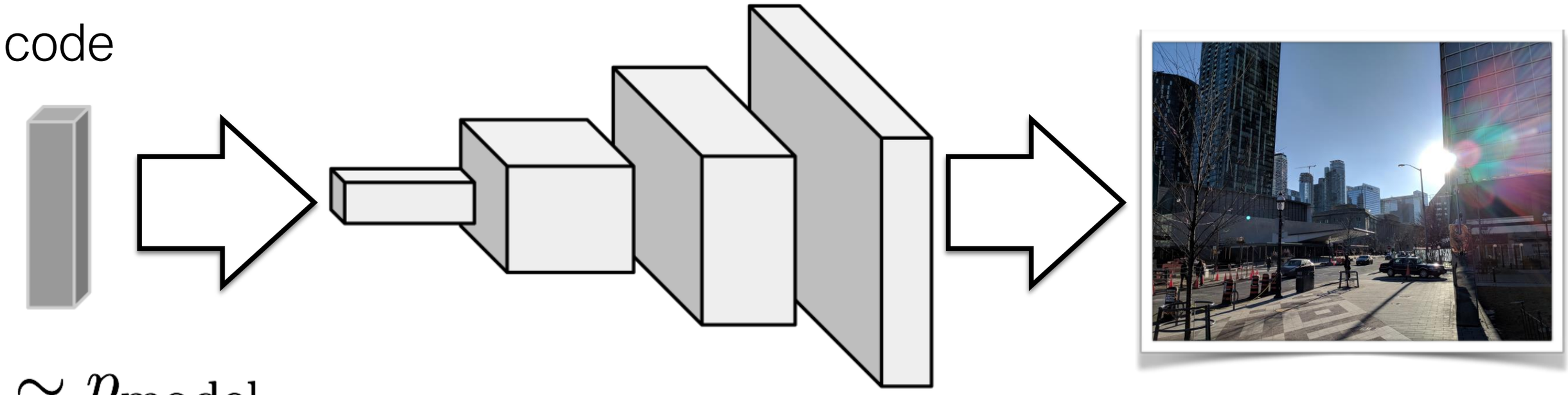
$\mathbf{z} \sim p_{\text{model}}$

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?

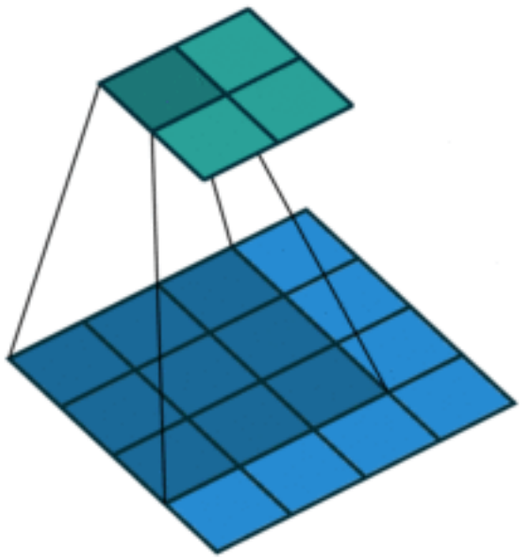
Today

- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GANs)

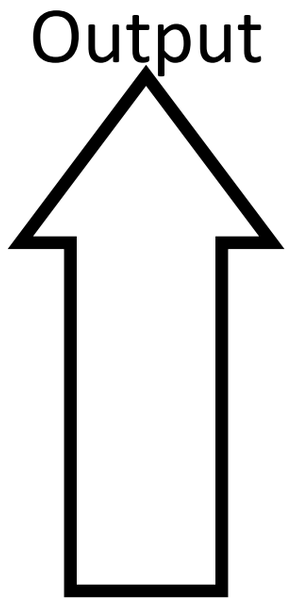
Today

- **Variational Autoencoders (VAE)**
- Generative Adversarial Networks (GANs)

Recap: (Transposed) Convolutions



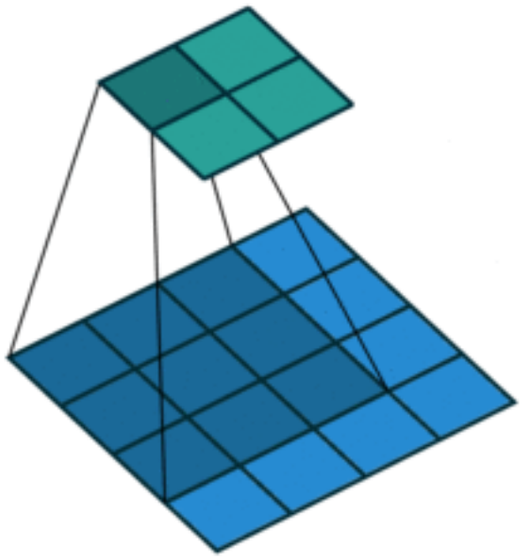
convolution



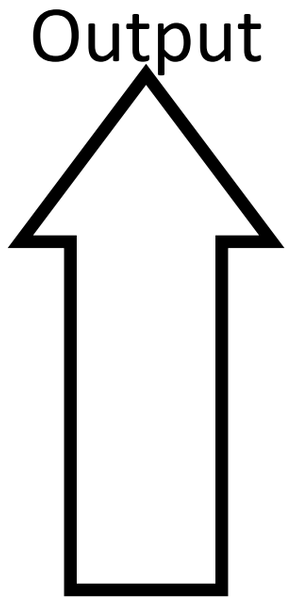
illustrations from https://github.com/vdumoulin/conv_arithmetic

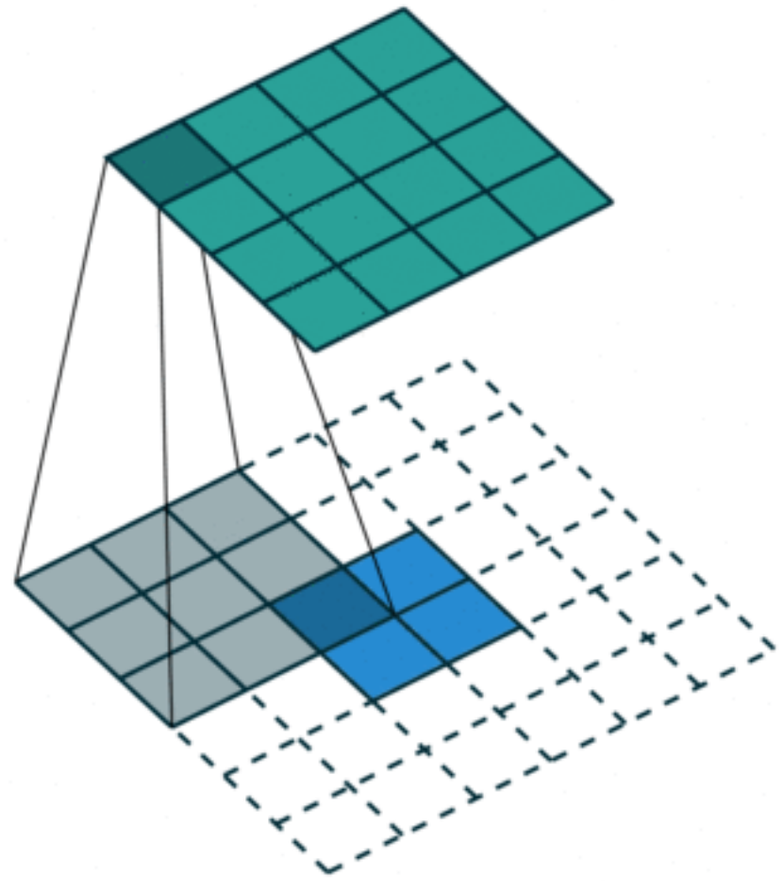
slide credit: Laura Leal-Taixé

Recap: (Transposed) Convolutions



convolution





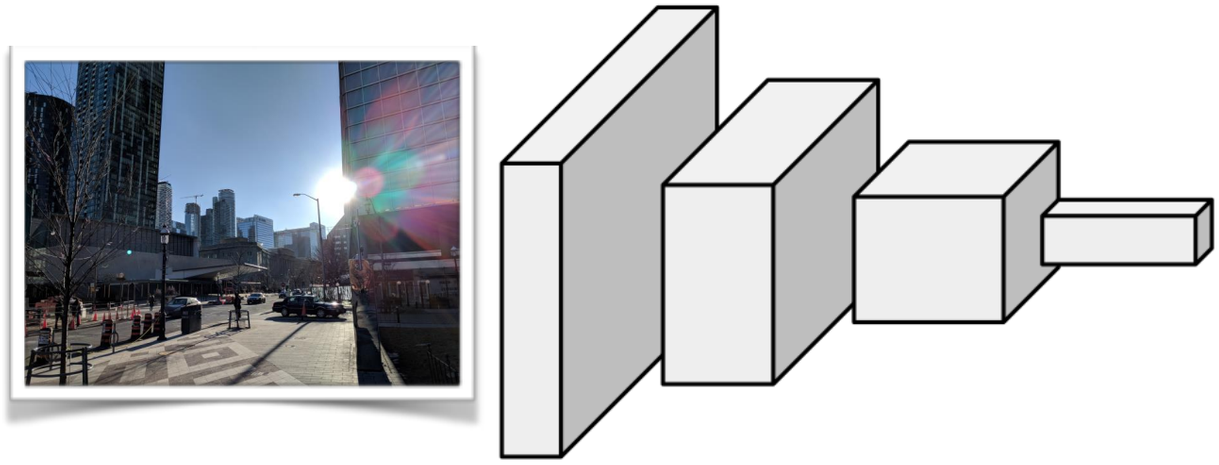
transposed convolution

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

slide credit: Laura Leal-Taixé

Autoencoder (AE)

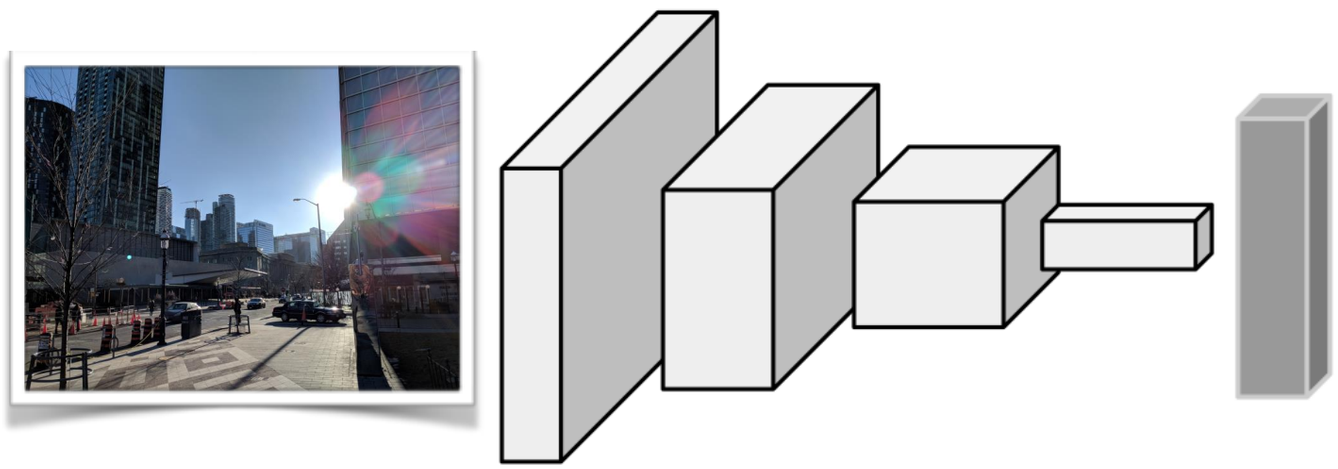
Autoencoder (AE)



encoder
(convolutions)

Autoencoder (AE)

latent
space

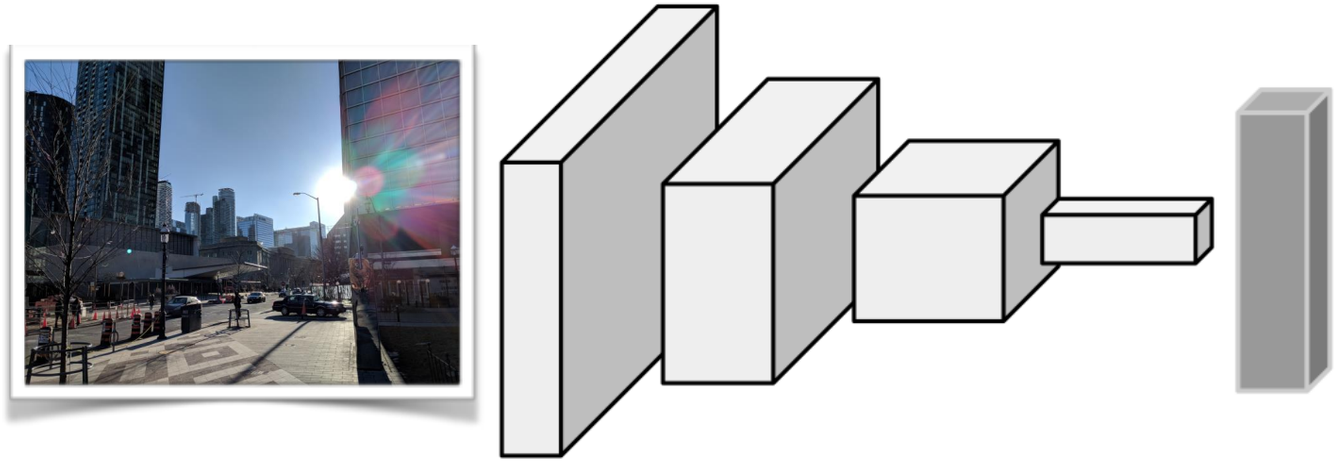


encoder
(convolutions)

Autoencoder (AE)

latent space

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

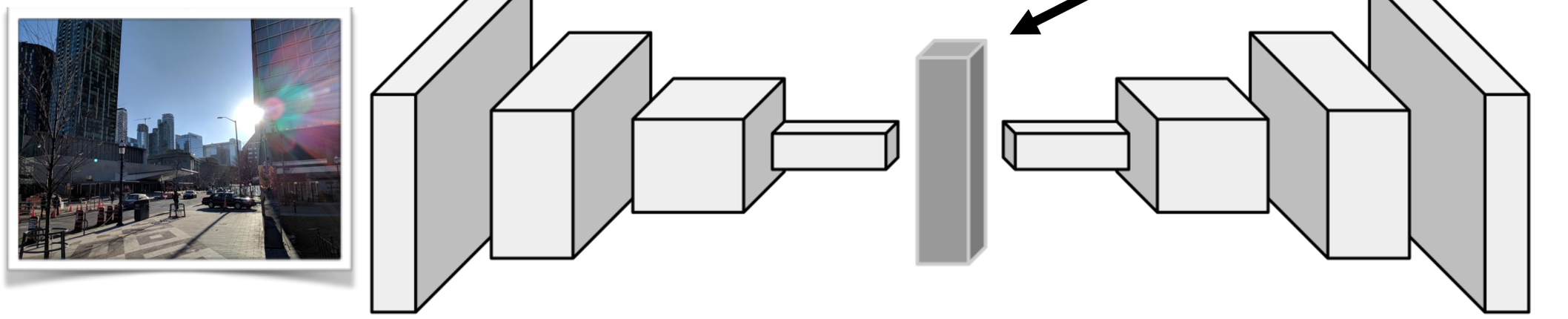


encoder (convolutions)

Autoencoder (AE)

latent space

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



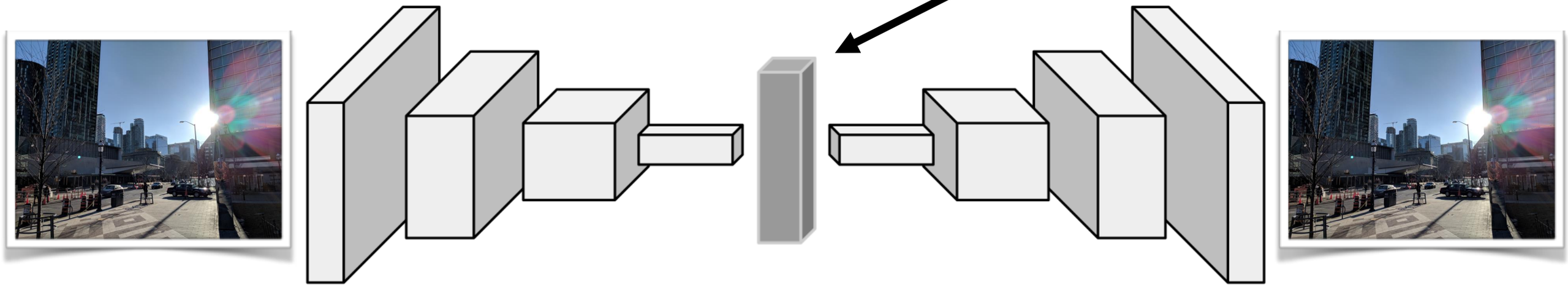
encoder (convolutions)

decoder (transp. convolutions)

Autoencoder (AE)

latent space

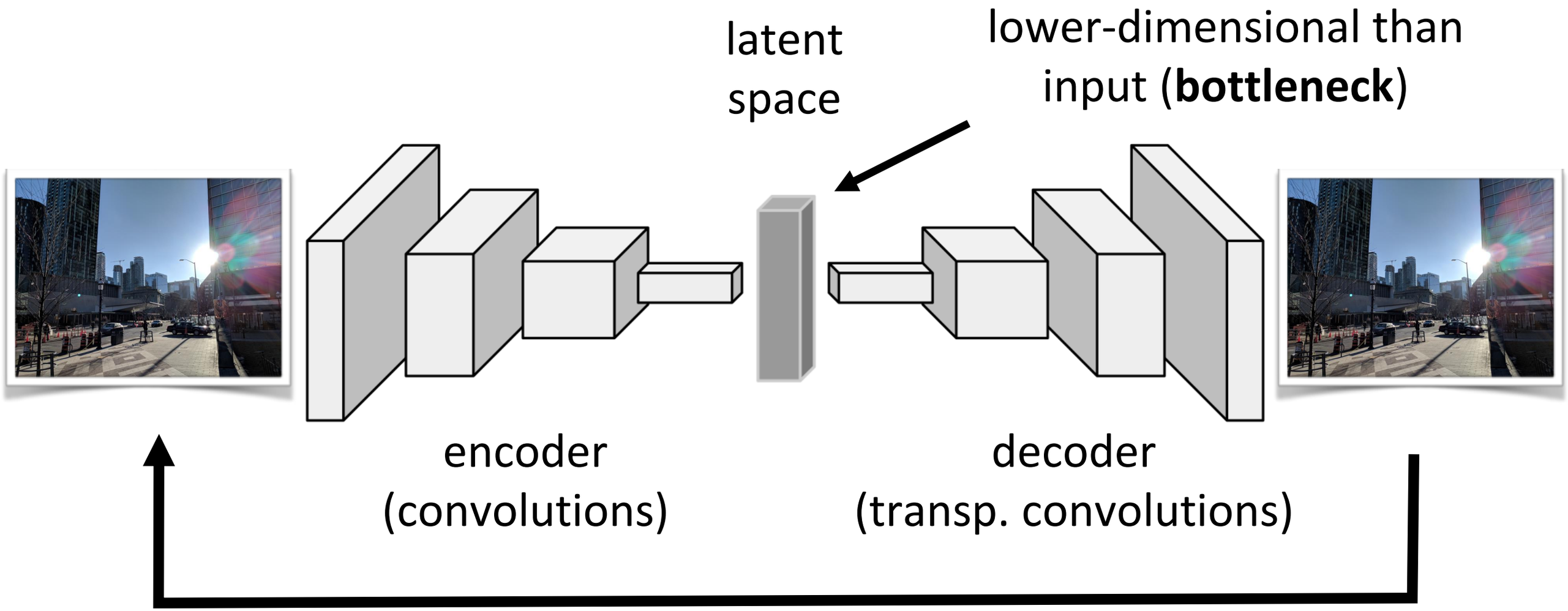
lower-dimensional than input (**bottleneck**)



encoder (convolutions)

decoder (transp. convolutions)

Autoencoder (AE)

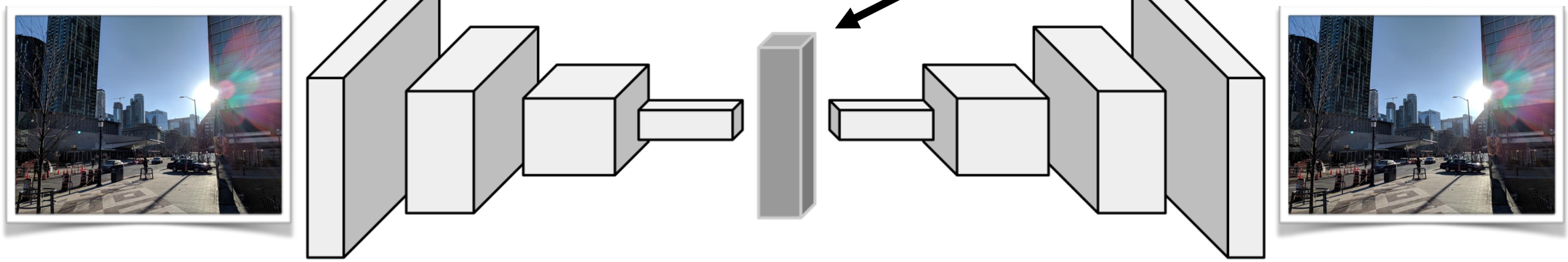


Variational Autoencoder (VAE)

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

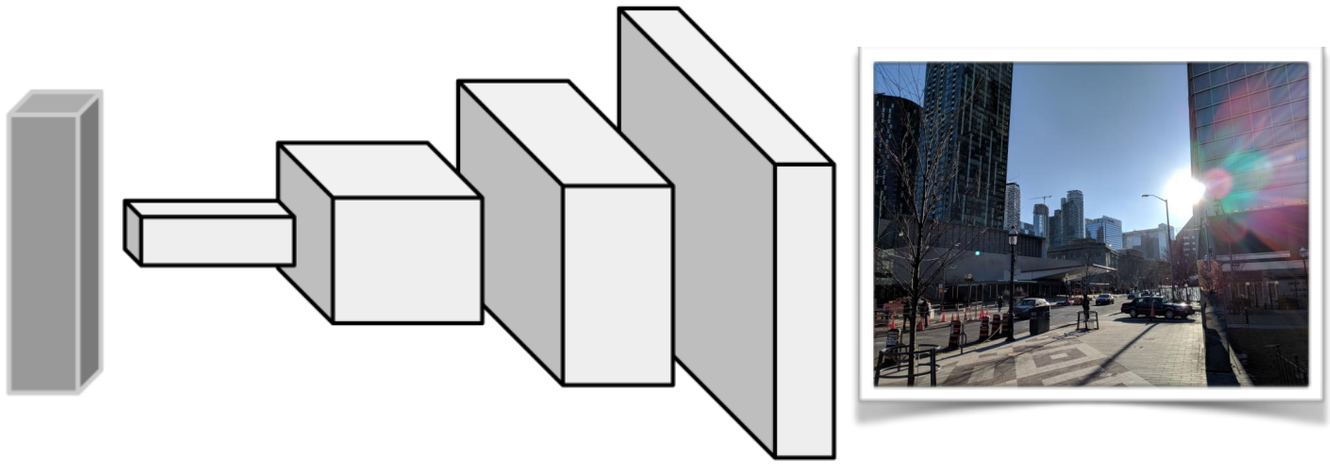
sample point in latent space

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

Image Generation via a VAE

sample point in latent space

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

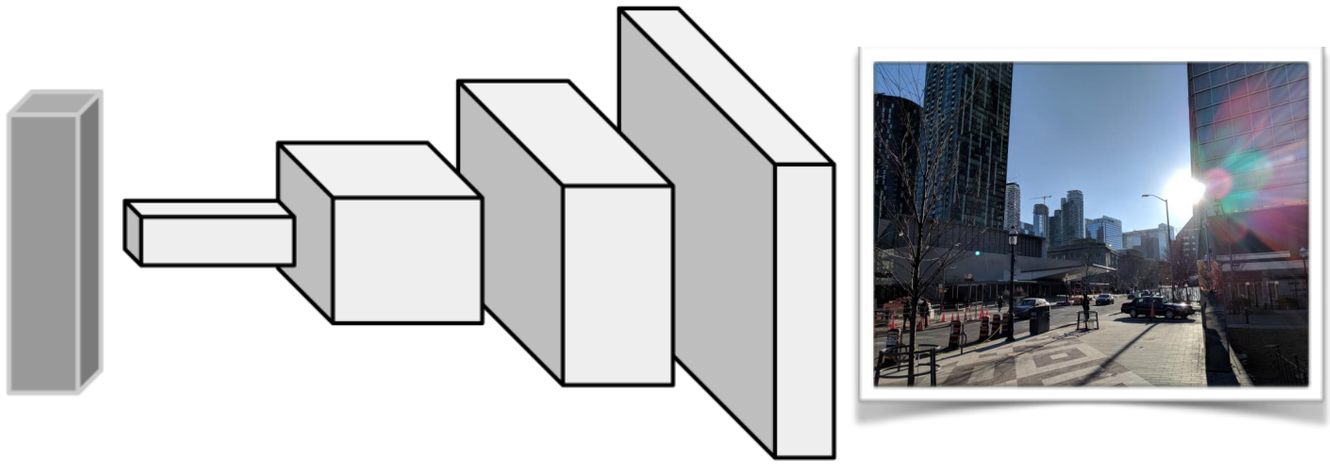


decoder
(transp. convolutions)

Image Generation via a VAE

sample point in latent space

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



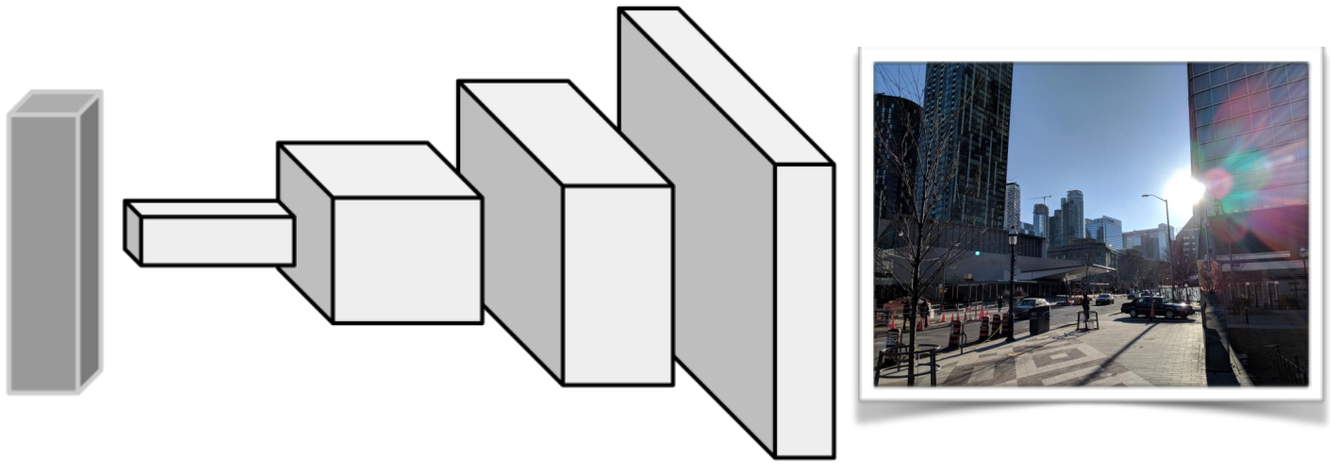
decoder
(transp. convolutions)

learn mapping from simple
to complex distribution

Image Generation via a VAE

sample point in latent space

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



decoder
(transp. convolutions)

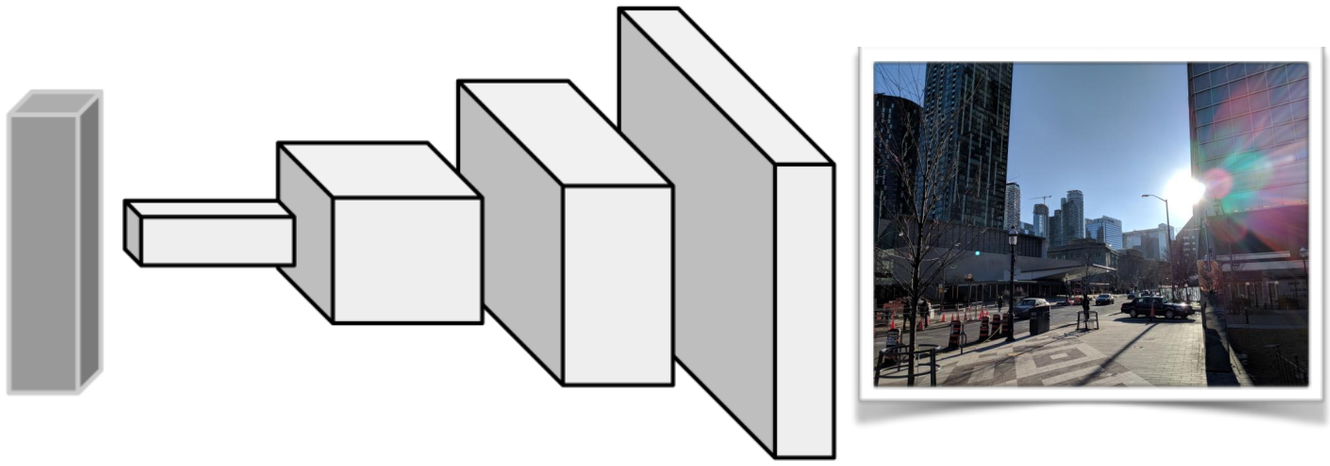
$$p(\mathbf{x}|\mathbf{z})$$

learn mapping from simple
to complex distribution

Image Generation via a VAE

sample point in latent space

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



decoder
(transp. convolutions)

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

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

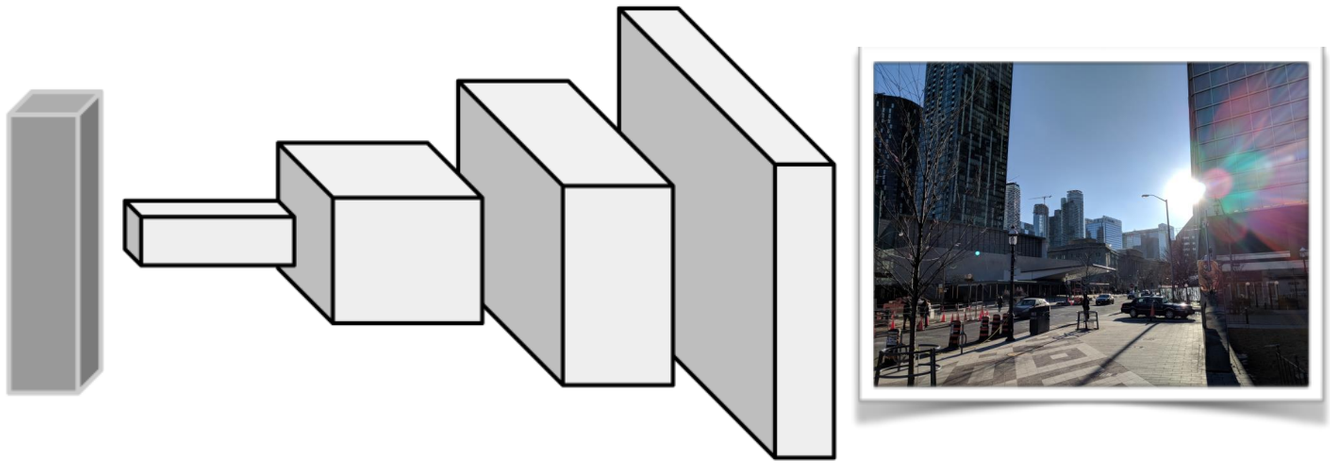
(ideally)

learn mapping from simple
to complex distribution

Image Generation via a VAE

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)

How to ensure that samples in latent space lead to meaningful images?

learn mapping from simple to complex distribution

Theory

Want to maximize

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

Theory

Want to maximize

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

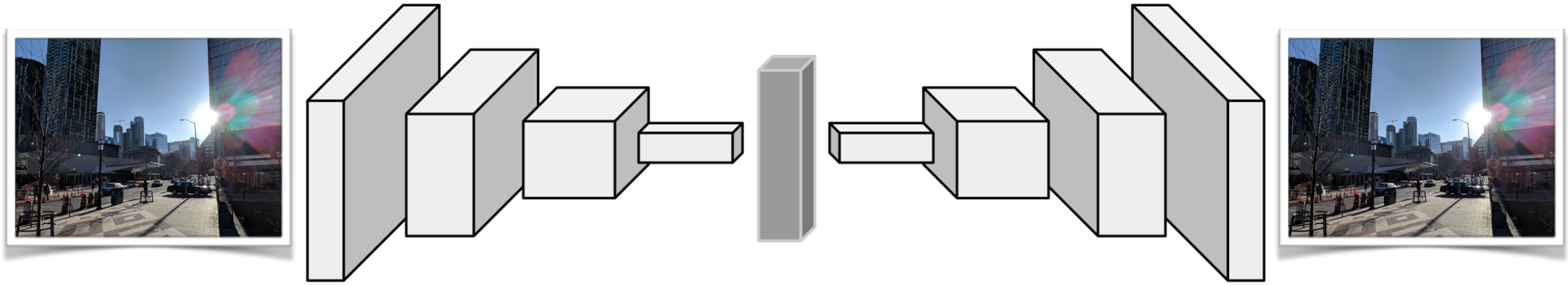
Intractable:

- Latent space is vector space
- Need to integrate over full space

Image Generation via a VAE

latent space

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



encoder
(convolutions)

decoder
(transp. convolutions)

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

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

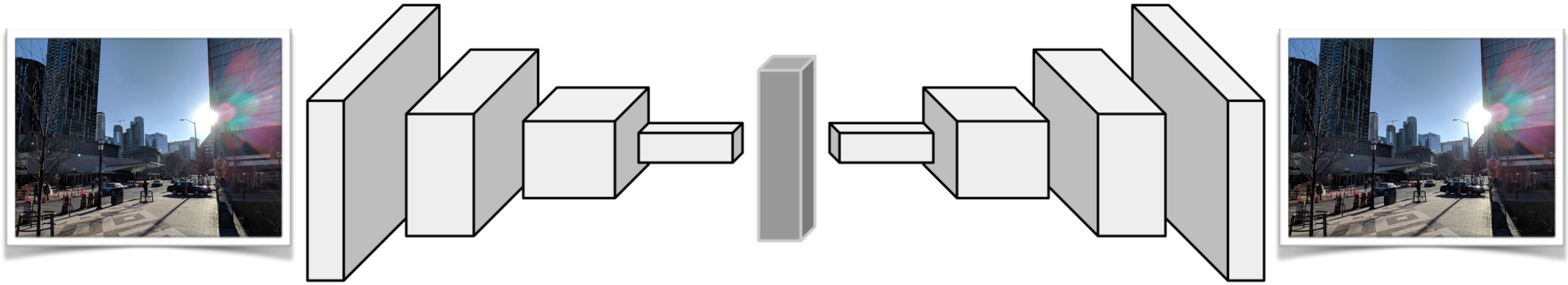
(ideally)

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)

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

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

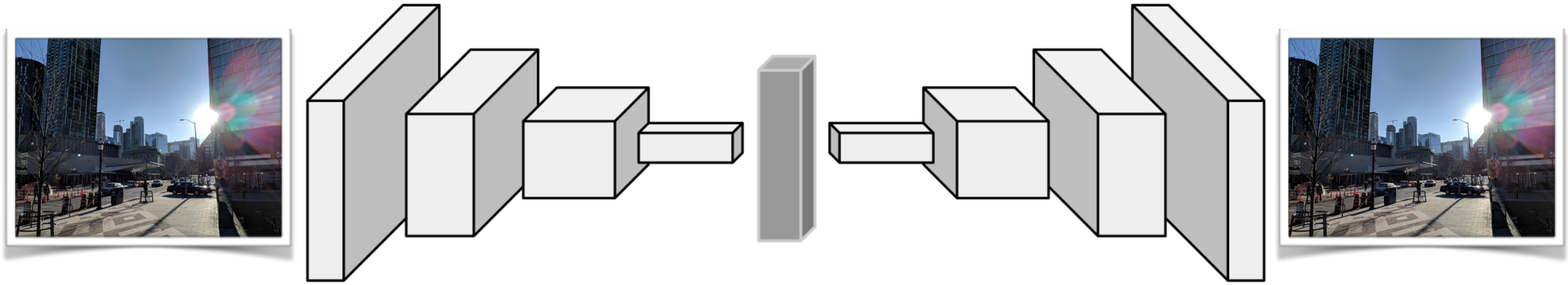
$\mathbf{x} \sim p_{\text{data}}$
(ideally)

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)

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

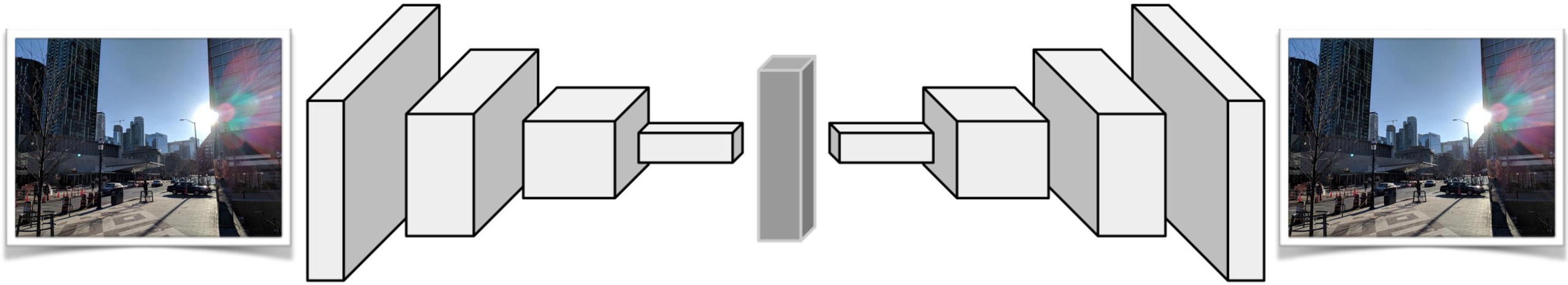
(ideally)

learn 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

Use encoder to generate probable samples, simplifying

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

Theory

Use encoder to generate probable samples, simplifying

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

to

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

Theory

Use encoder to generate probable samples, simplifying

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

to

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

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

KL Divergence

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

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

KL Divergence

Kullback-Leibler divergence: ``distance" between probability distribution

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

Properties:

KL Divergence

Kullback-Leibler divergence: ``distance" between probability distribution

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

Properties:

- KL divergence is non-negative: $D_{\mathrm{KL}}(P||Q) \geq 0$

KL Divergence

Kullback-Leibler divergence: ``distance" between probability distribution

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

Properties:

- KL divergence is non-negative: $D_{\mathrm{KL}}(P||Q) \geq 0$
- KL divergence is 0 exactly if: $D_{\mathrm{KL}}(P||Q) = 0 \Leftrightarrow P(x) = Q(x) \forall x$

KL Divergence

Kullback-Leibler divergence: ``distance" between probability distribution

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

Properties:

- KL divergence is non-negative: $D_{\mathrm{KL}}(P||Q) \geq 0$
- KL divergence is 0 exactly if: $D_{\mathrm{KL}}(P||Q) = 0 \Leftrightarrow P(x) = Q(x) \forall x$
- KL divergence is asymmetric: $D_{\mathrm{KL}}(P||Q) \neq D_{\mathrm{KL}}(Q||P)$

KL Divergence

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

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

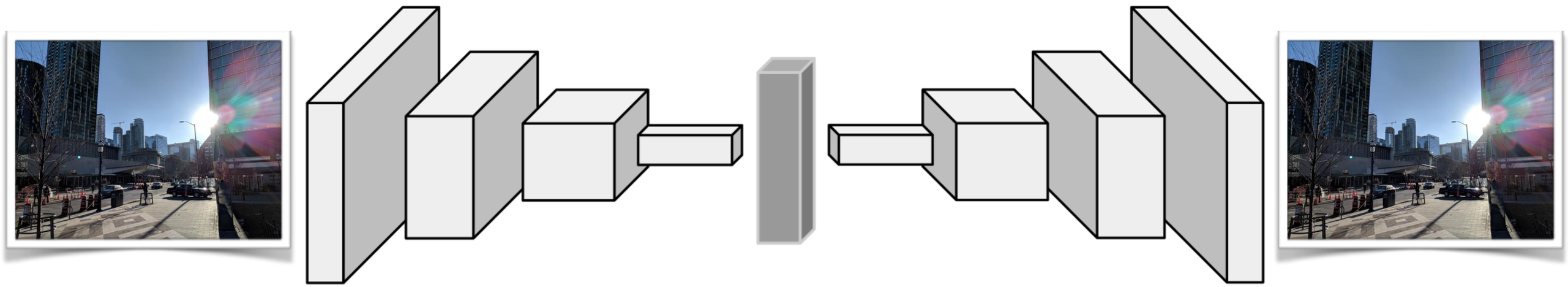
Equivalent to (when sampling):

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

Theory

latent space

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



encoder

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

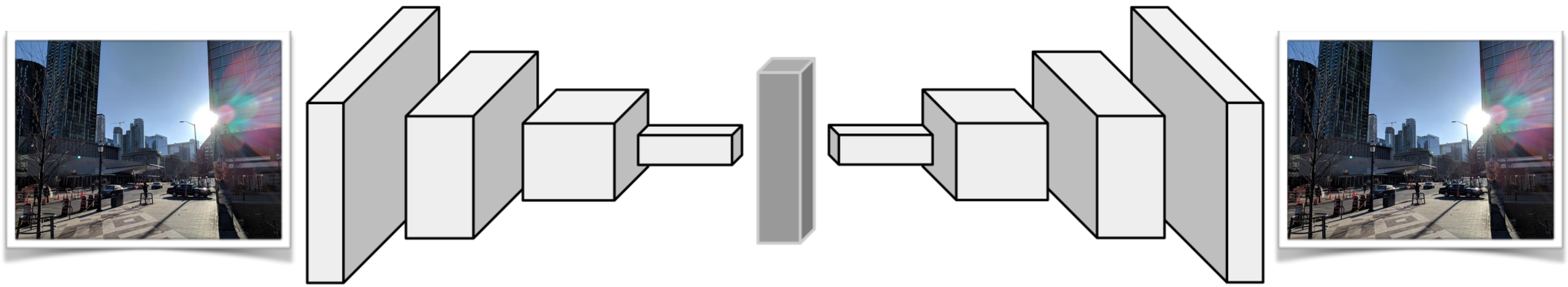
decoder

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

Theory

latent space

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



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

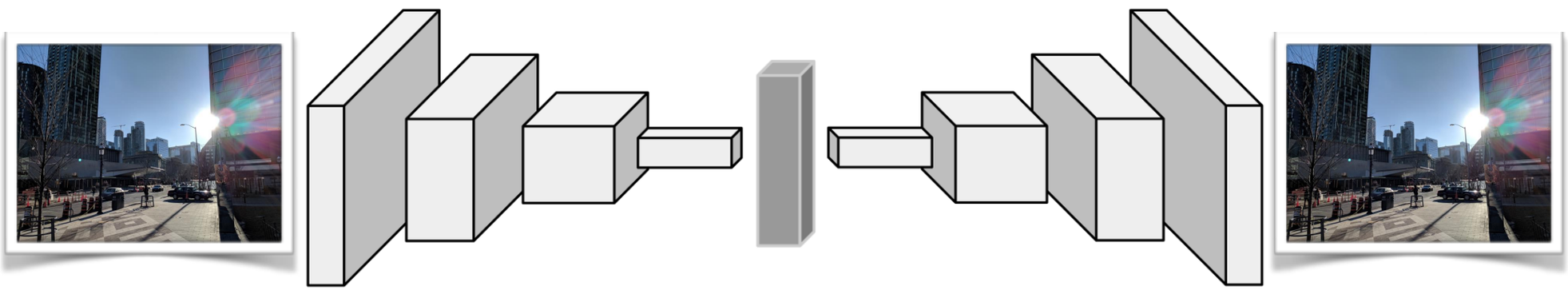
decoder
$p(\mathbf{x}|\mathbf{z})$

Want to ensure that `q(z|x)` and `p(z|x)` are the same:

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

Theory

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



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

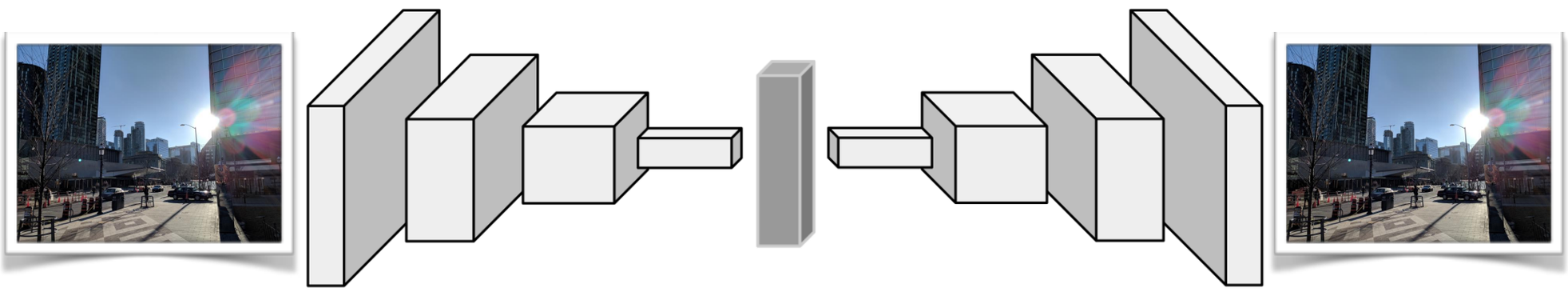
decoder $p(\mathbf{x}|\mathbf{z})$

Use Bayes rule to include $p(\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$:

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

Theory

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



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

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

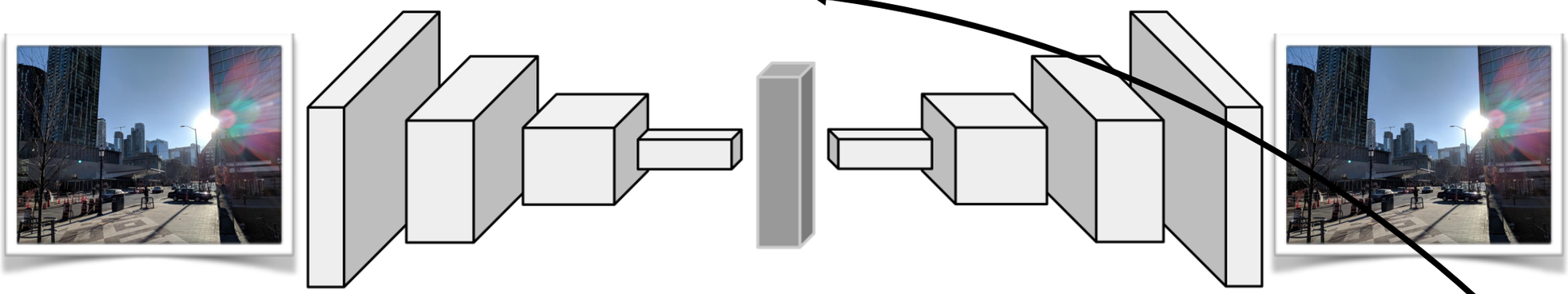
Use Bayes rule to include $p(\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$:

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

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

Theory

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



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

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

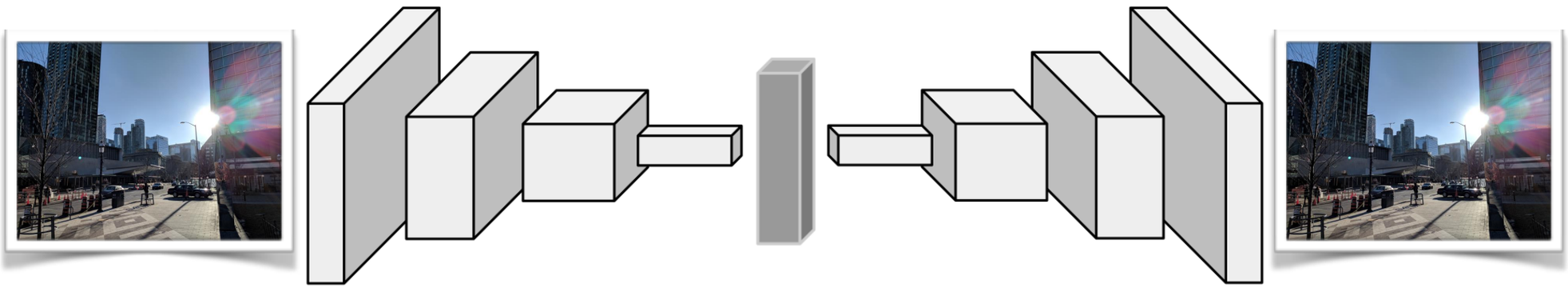
Use Bayes rule to include $p(\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$:

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

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

Theory

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



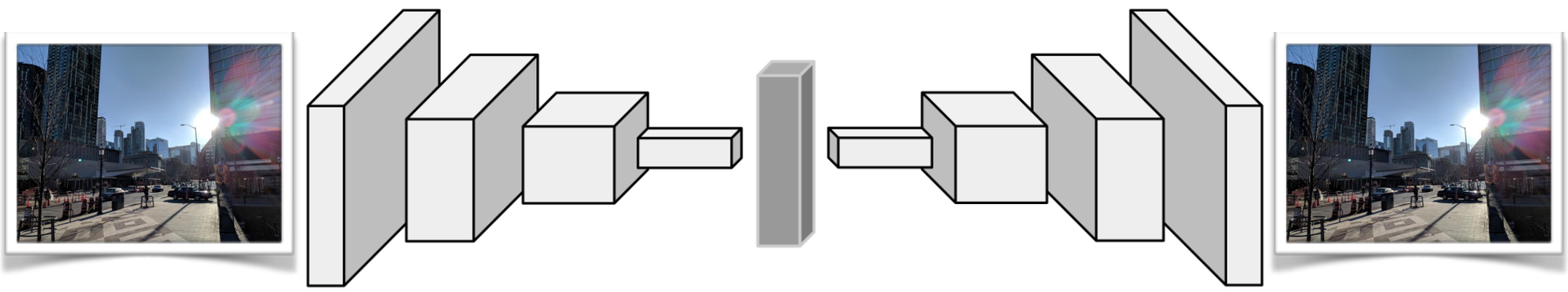
encoder
$q(\mathbf{z}|\mathbf{x})$

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

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

Theory

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



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

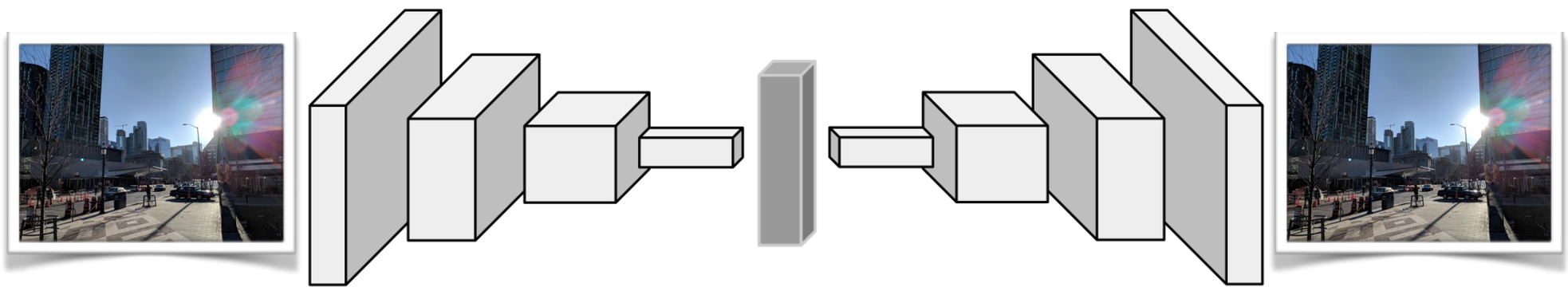
decoder
$p(\mathbf{x}|\mathbf{z})$

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

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

Theory

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



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

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

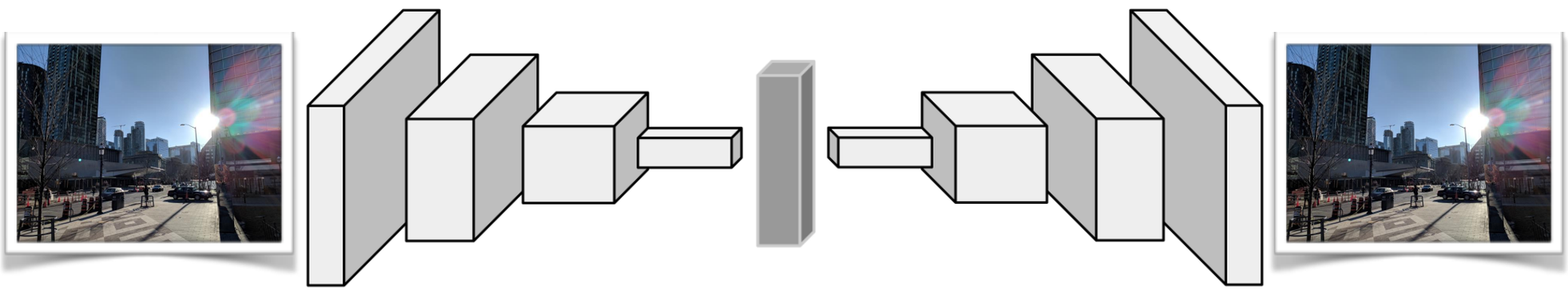
$$D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log\left(\frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}\right)]$$

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

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

Theory

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



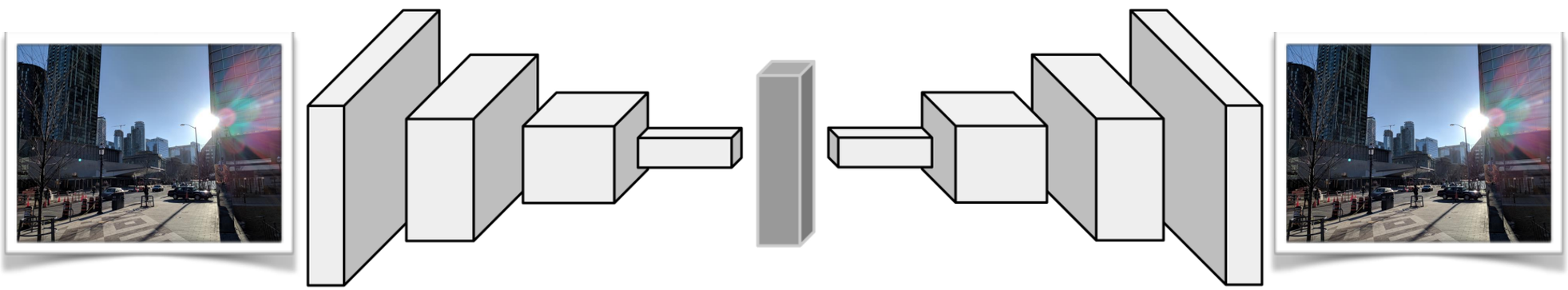
encoder
$q(\mathbf{z}|\mathbf{x})$

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

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

Theory

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



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

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

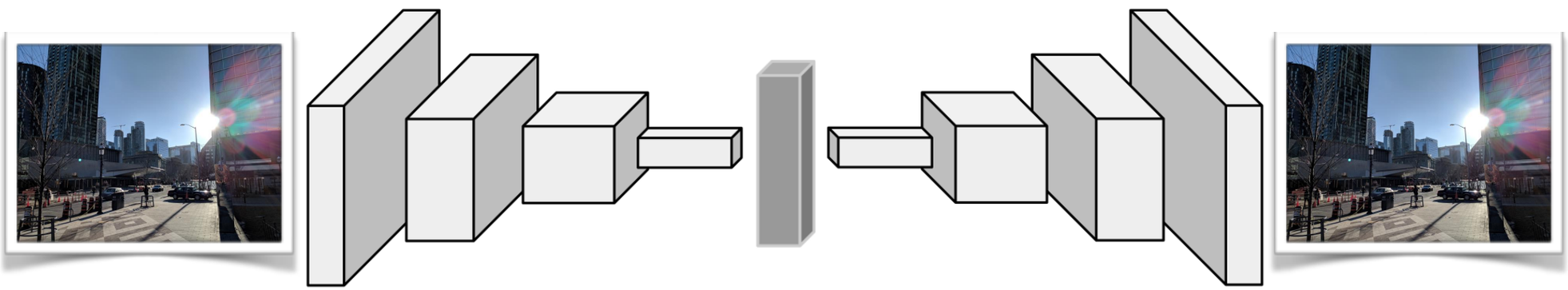
$$D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = -E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] + E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log(p(\mathbf{z}))] + \log(p(\mathbf{x}))$$

Rearrange:

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

Theory

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



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

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

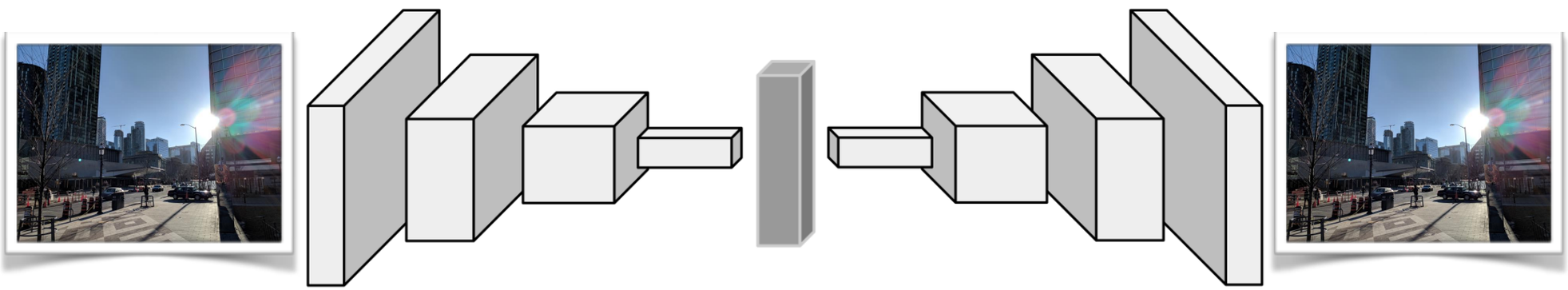
$$D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = -E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] + E_{\mathbf{z}\sim q}[\log(q(\mathbf{z}|\mathbf{x})) - \log(p(\mathbf{z}))] + \log(p(\mathbf{x}))$$

Rearrange:

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

Theory

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



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

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

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

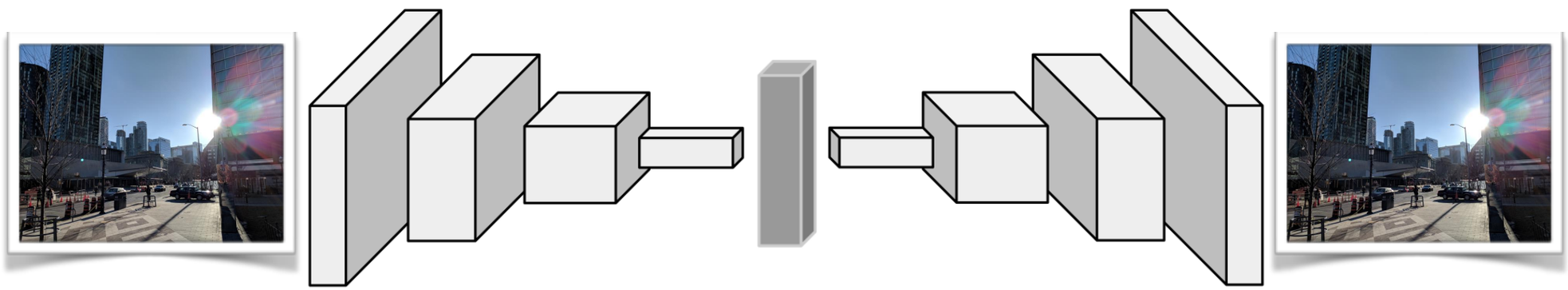
Rearrange:

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

decoder

Theory

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



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

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

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

Rearrange:

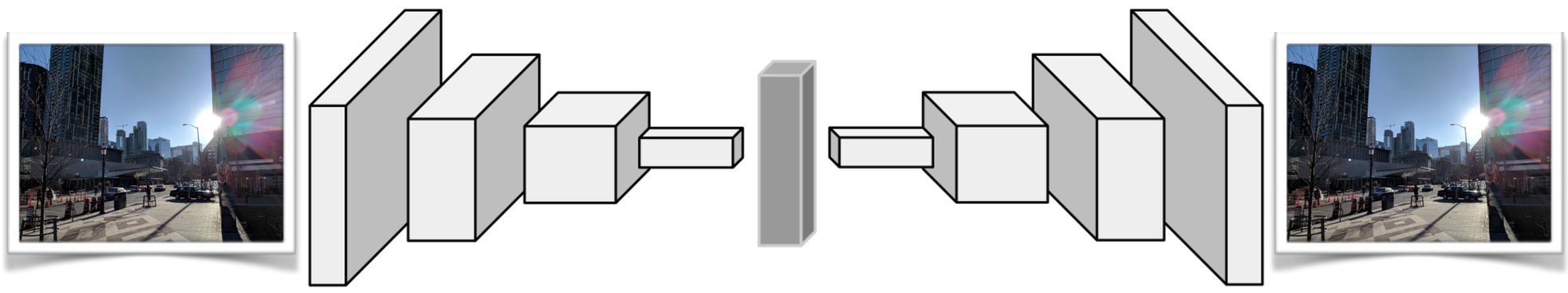
$$\log(p(\mathbf{x})) = D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) + E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

how to compute?

decoder

Theory

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



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

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

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

Rearrange:

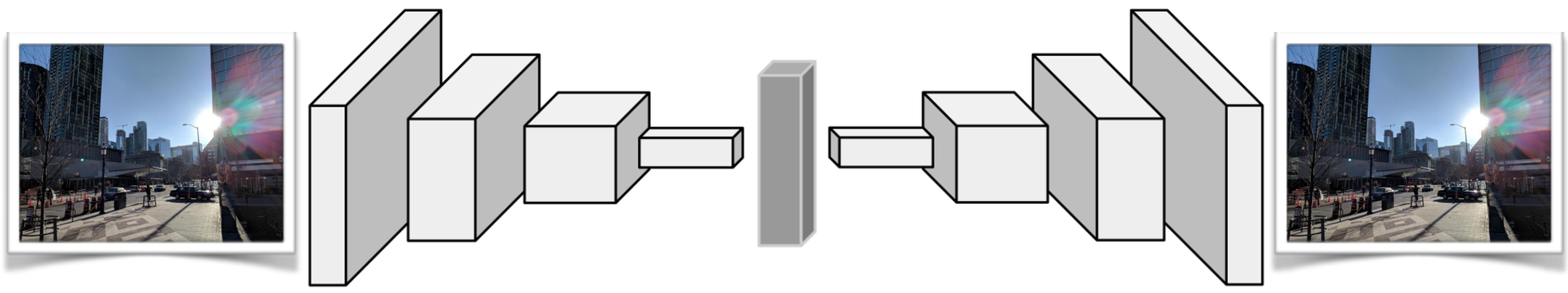
$$\log(p(\mathbf{x})) = D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) + E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

how to compute?
no idea!

decoder

Theory

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



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

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

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

Rearrange:

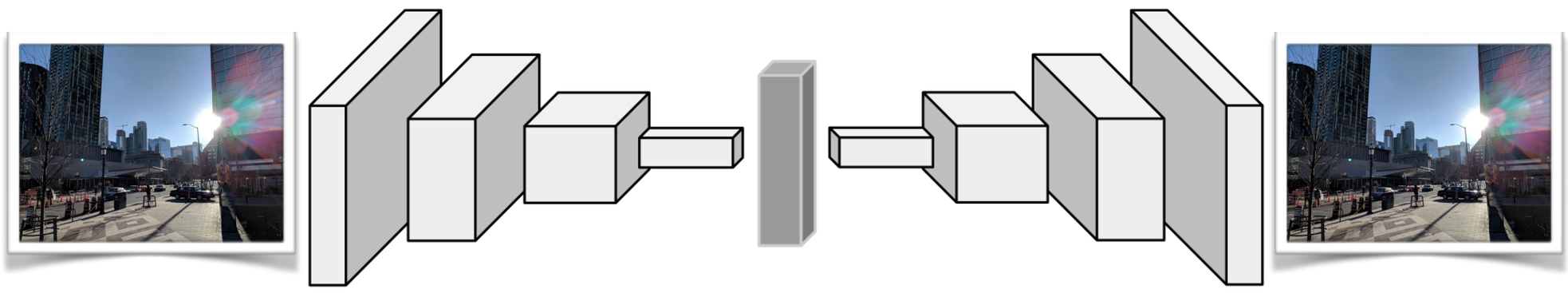
$$\log(p(\mathbf{x})) = D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) + E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

how to compute?
no idea! But ≥0

decoder

Theory

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



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

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

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

Rearrange:

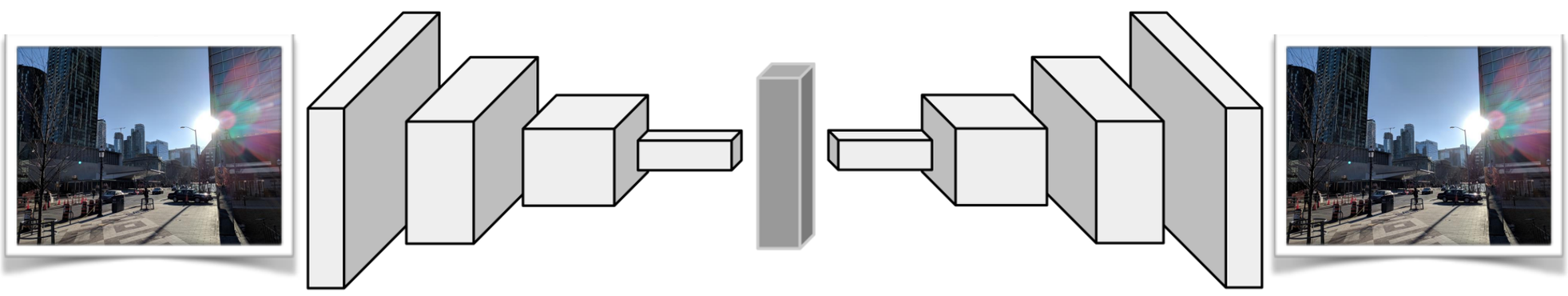
$$\log(p(\mathbf{x})) = D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) + E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

how to compute?
no idea! But ≥0

decoder

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

Theory

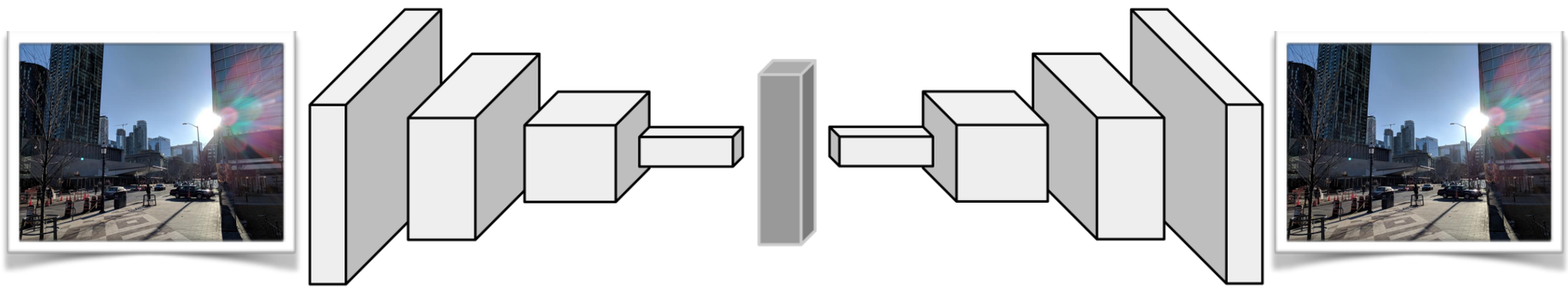


Evidence Lower BOund (ELBO):

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

Theory

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



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

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

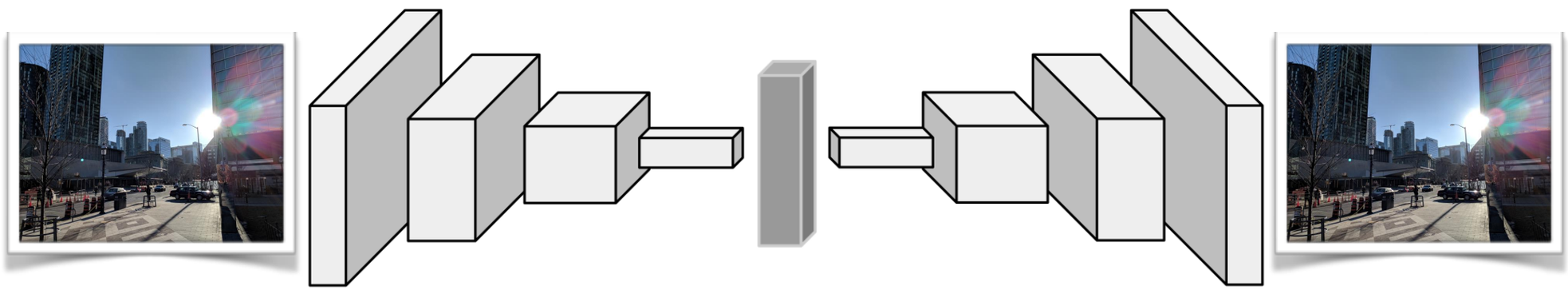
Evidence Lower BOund (ELBO):

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

likelihood

Theory

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



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

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

Evidence Lower BOund (ELBO):

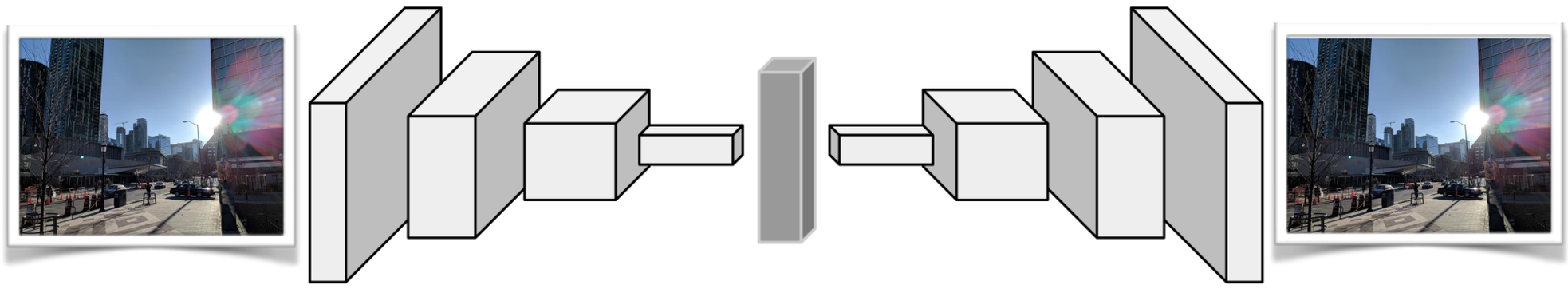
$$\log(p(\mathbf{x})) \geq E_{\mathbf{z}\sim q}[\log(p(\mathbf{x}|\mathbf{z}))] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

likelihood

KL divergence

Objective Function

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



encoder

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

decoder

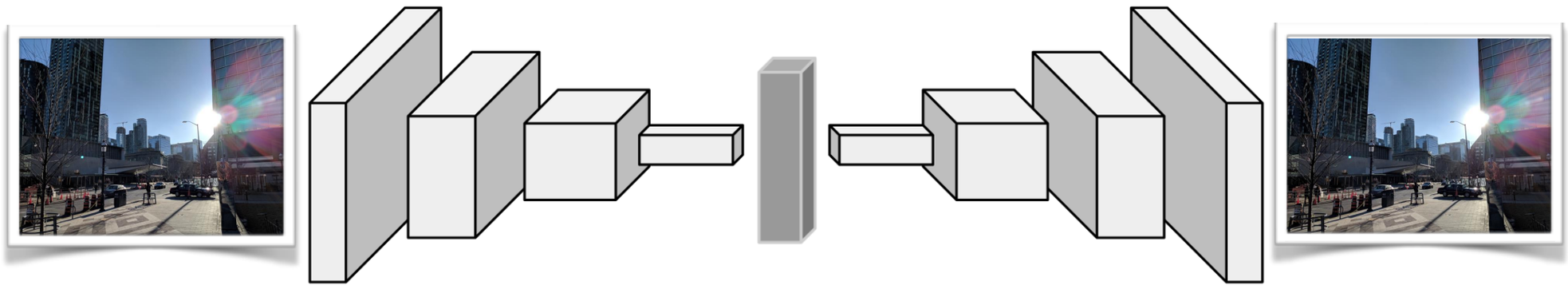
$p(\mathbf{x}|\mathbf{z})$

Maximize

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

Objective Function

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



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

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

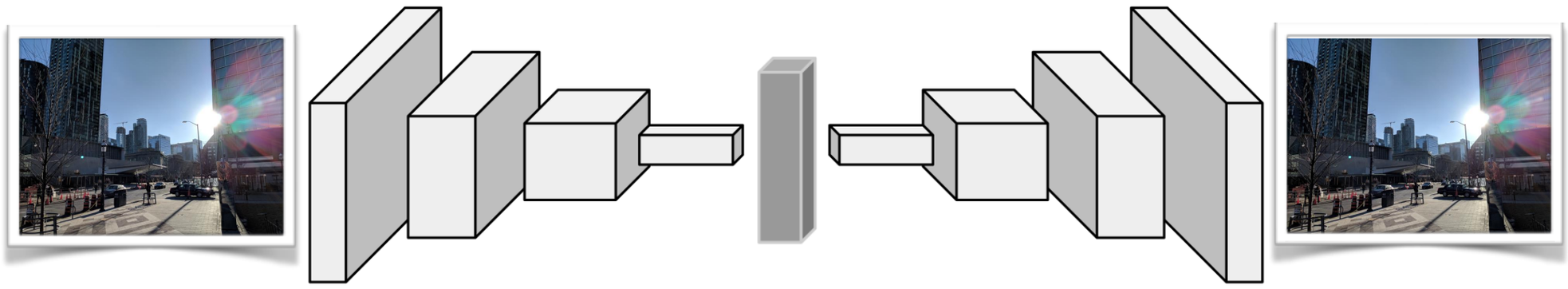
Maximize

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

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

Objective Function

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



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

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

Maximize

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

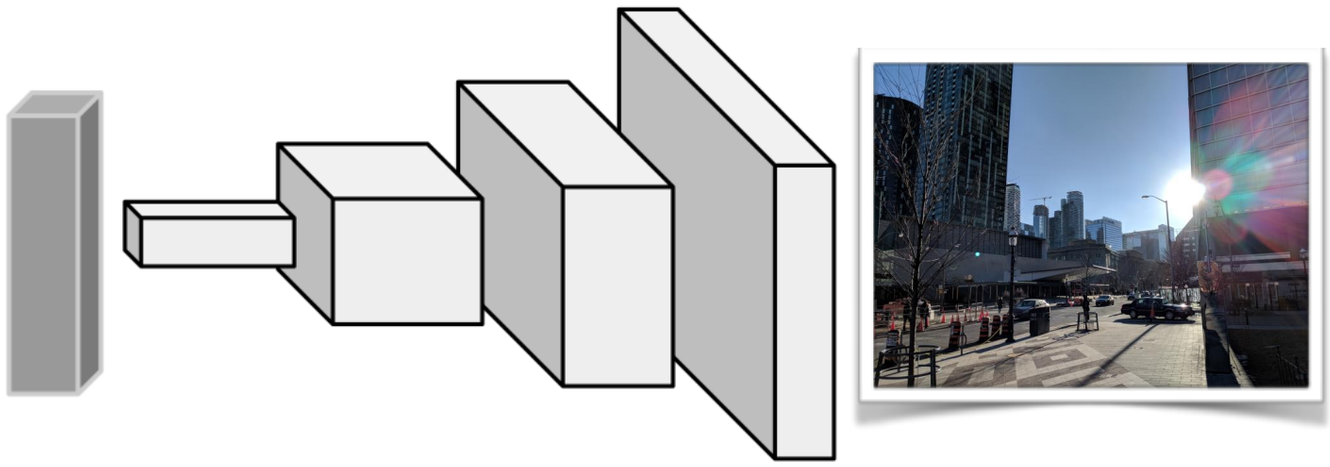
reconstruction ``loss'',
e.g., L2 norm (normal distribution)

similarity between learned
and sampling distribution

At Test Time

sample point from latent space

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



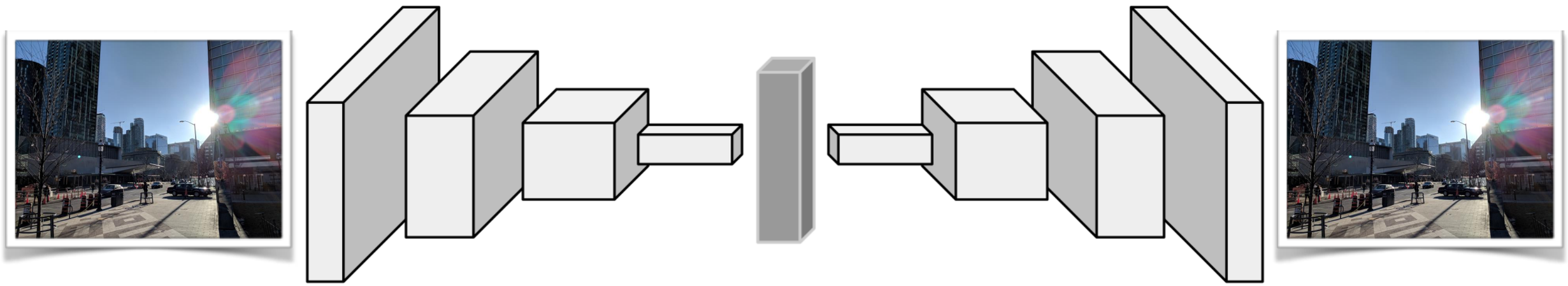
decoder
(transp. convolutions)

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

use decoder to ``translate”
to image

VAE - Example

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



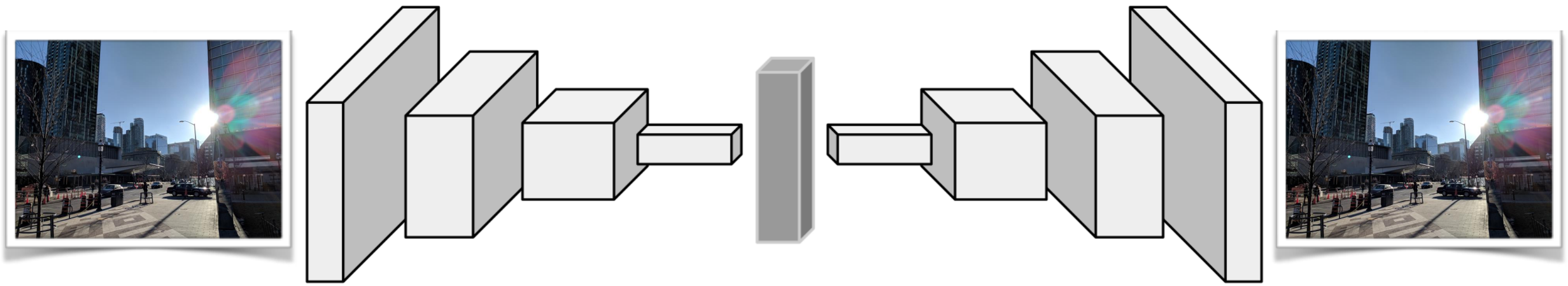
encoder
$q(\mathbf{z}|\mathbf{x})$

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

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

VAE - Example

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



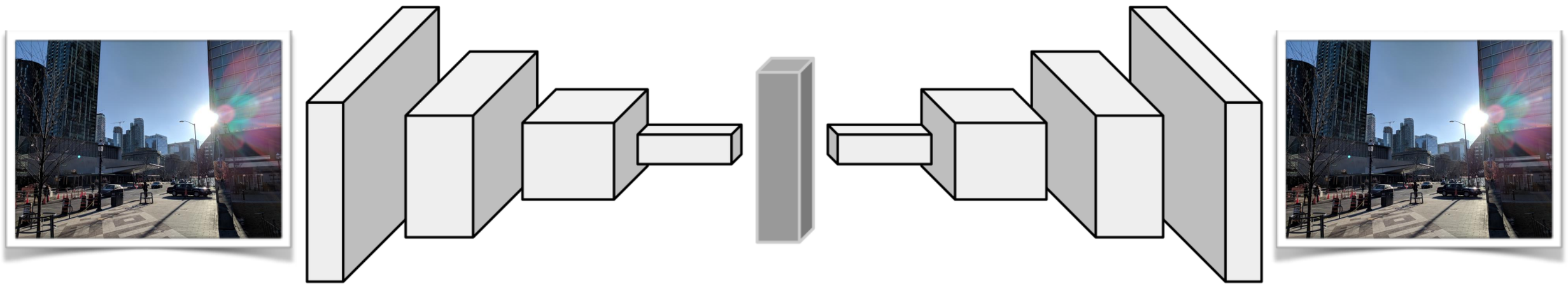
encoder $q(\mathbf{z}|\mathbf{x})$

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

[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

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

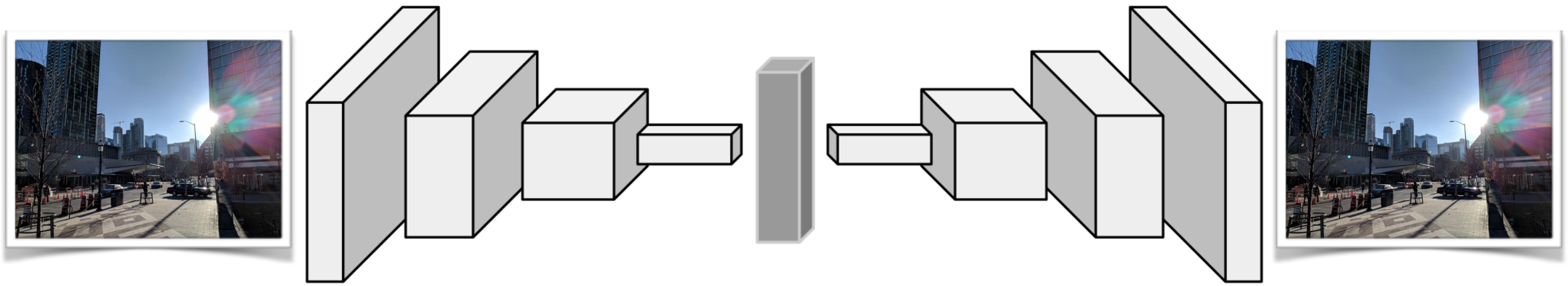
decoder

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

[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

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

decoder

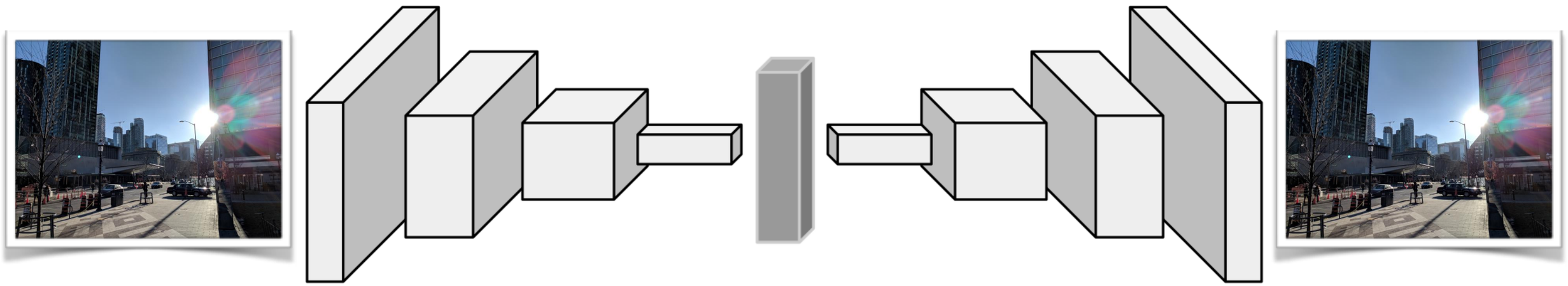
$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu', \sigma'^2 \mathrm{I})$

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

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

decoder

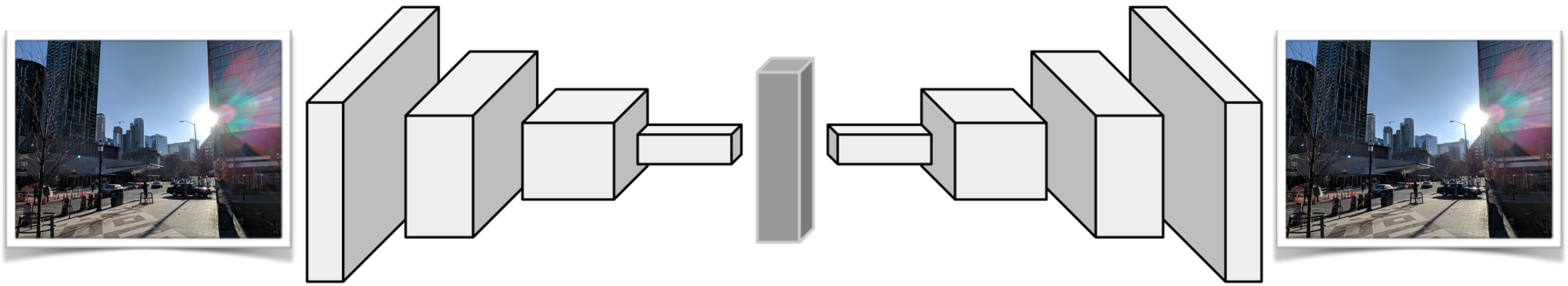
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu', \sigma'^2\mathrm{I})$$

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

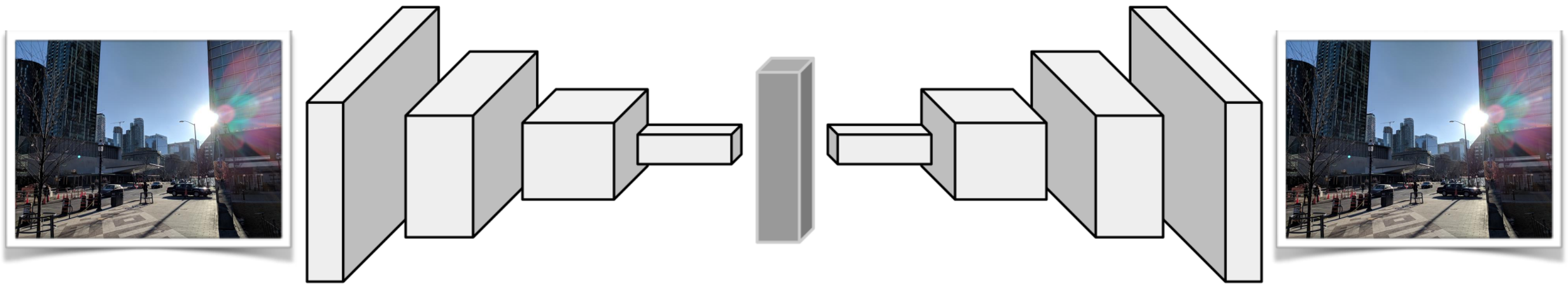
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu', \sigma'^2\mathrm{I})$$

least-squares solution

[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

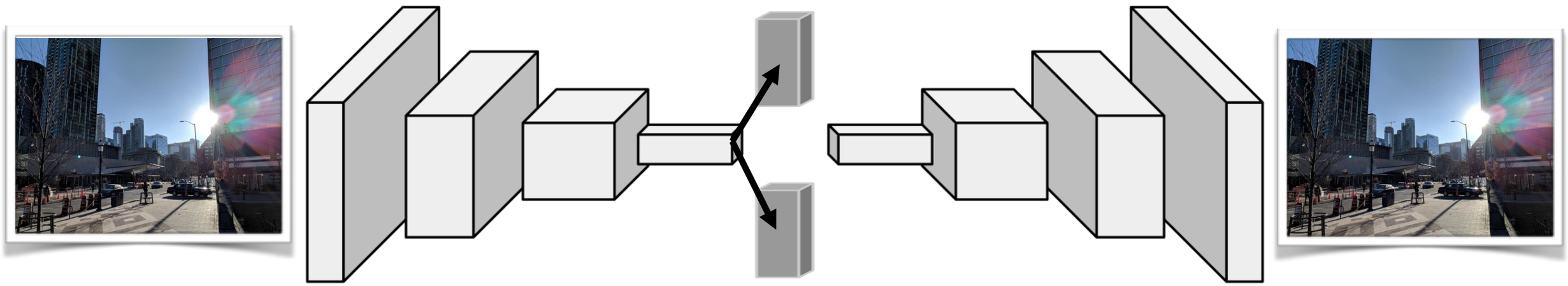
$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \mu', \sigma'^2\mathrm{I})$$

least-squares solution

diagonal co-variance matrix

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

VAE - Example

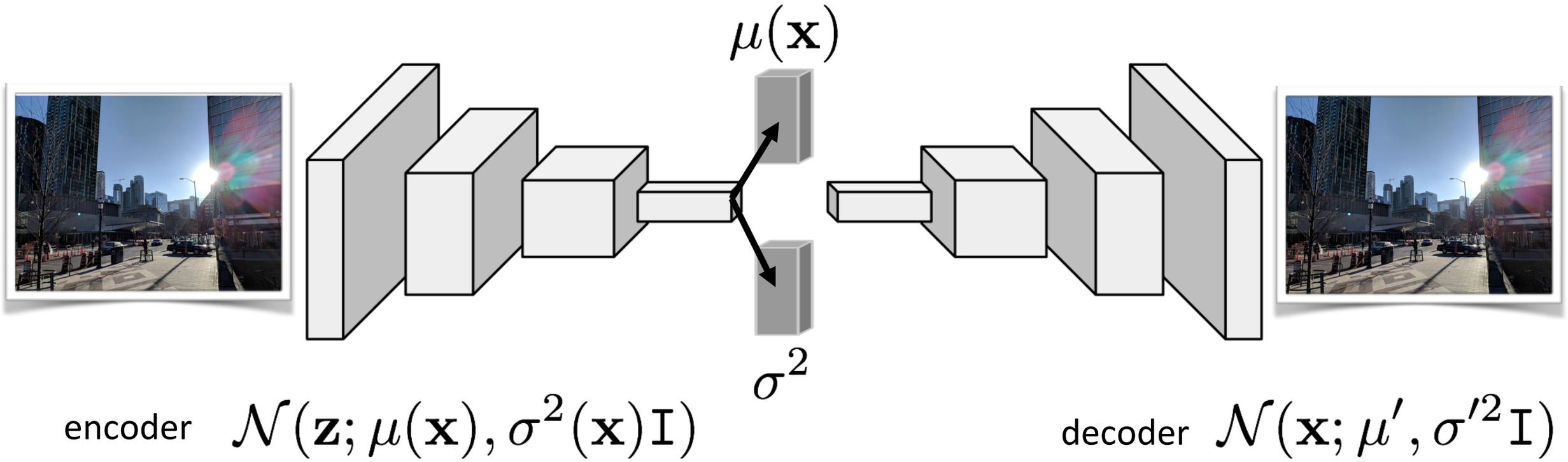


encoder $\mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x})\mathrm{I})$

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

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

VAE - Example

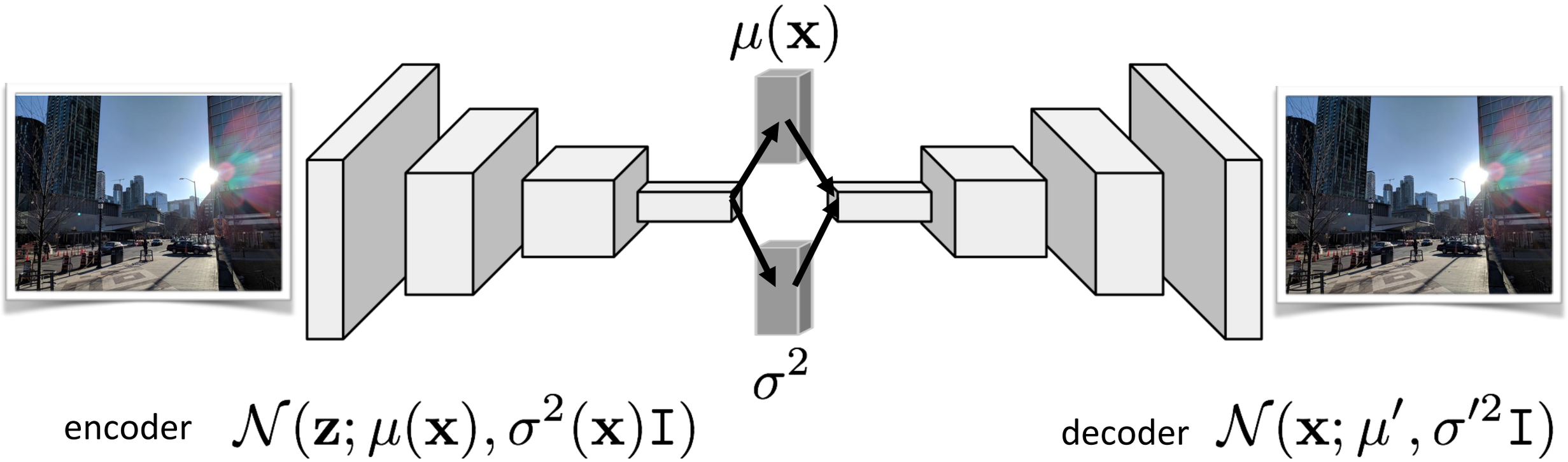


encoder $\mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x})\mathrm{I})$

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

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

VAE - Example

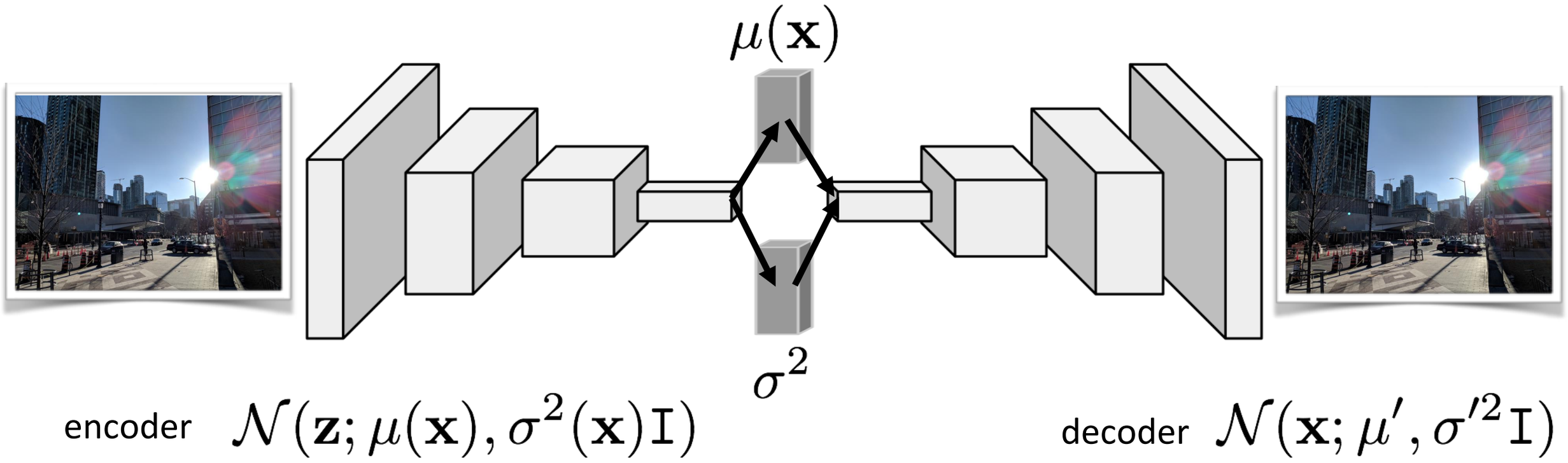


encoder $\mathcal{N}(\mathbf{z}; \mu(\mathbf{x}), \sigma^2(\mathbf{x})\mathrm{I})$

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

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

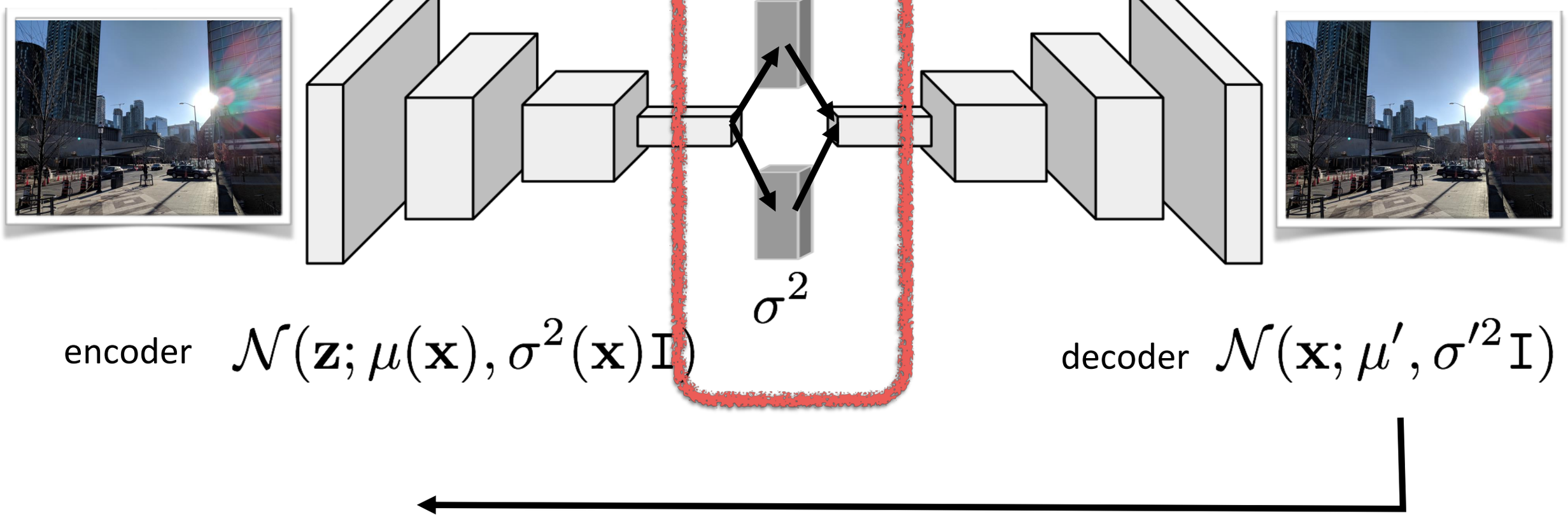
VAE - Example



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

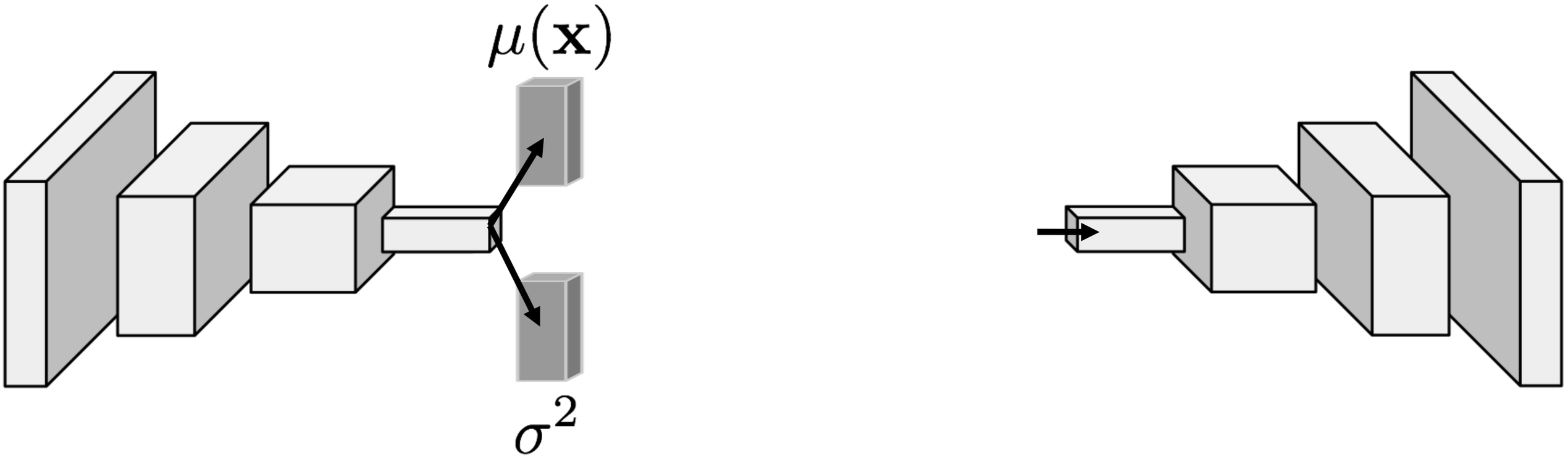
VAE - Example

sampling z is non-continuous operation → gradient?



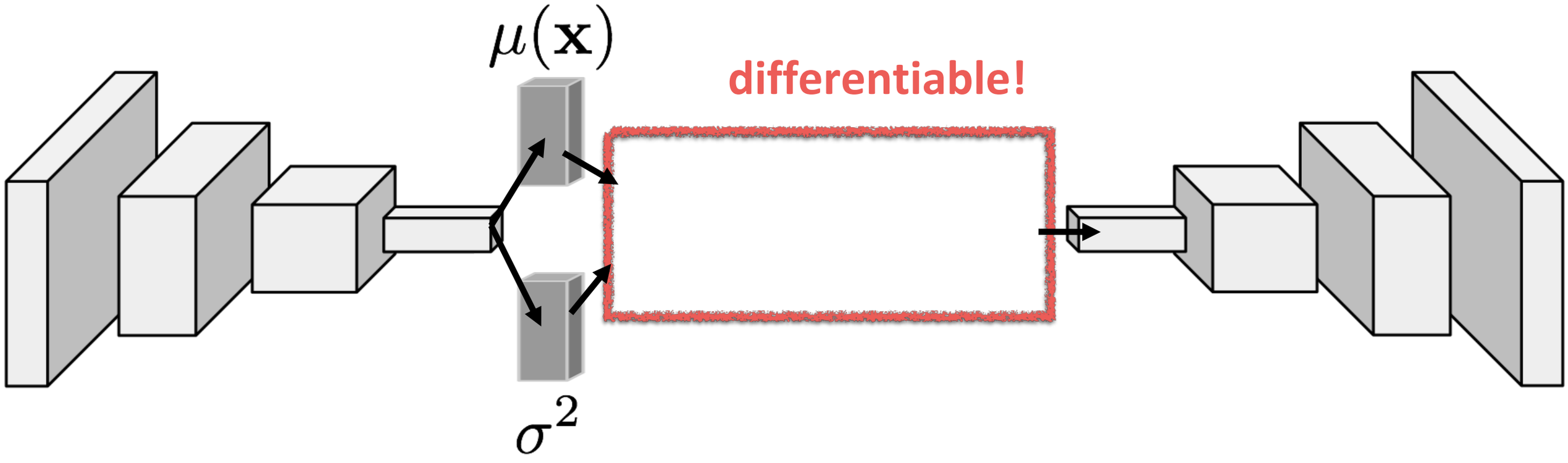
[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

The Reparametrization Trick



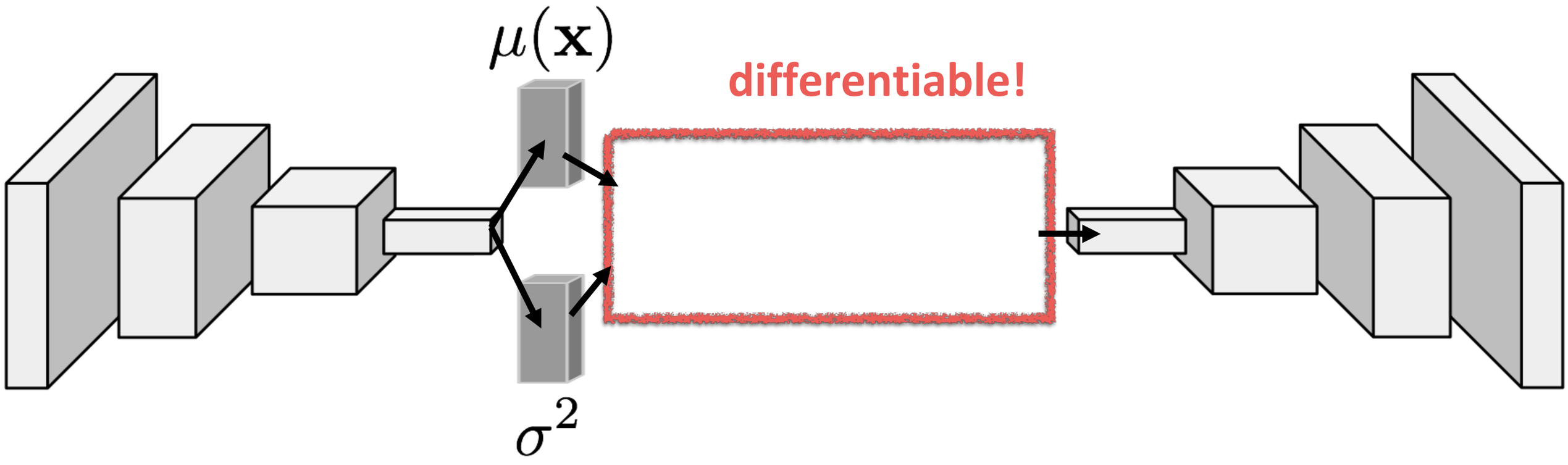
[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

The Reparametrization Trick



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

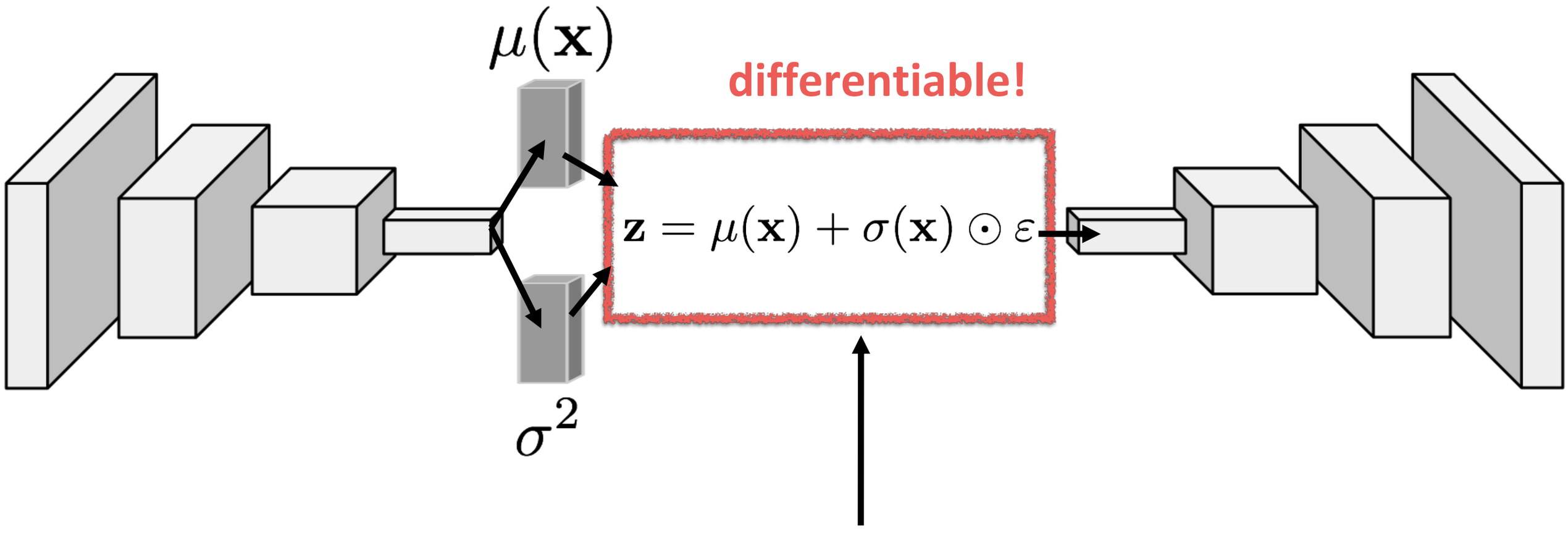
The Reparametrization Trick



external sampling: $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$

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

The Reparametrization Trick



external sampling: $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

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

VAE Example - Results

(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

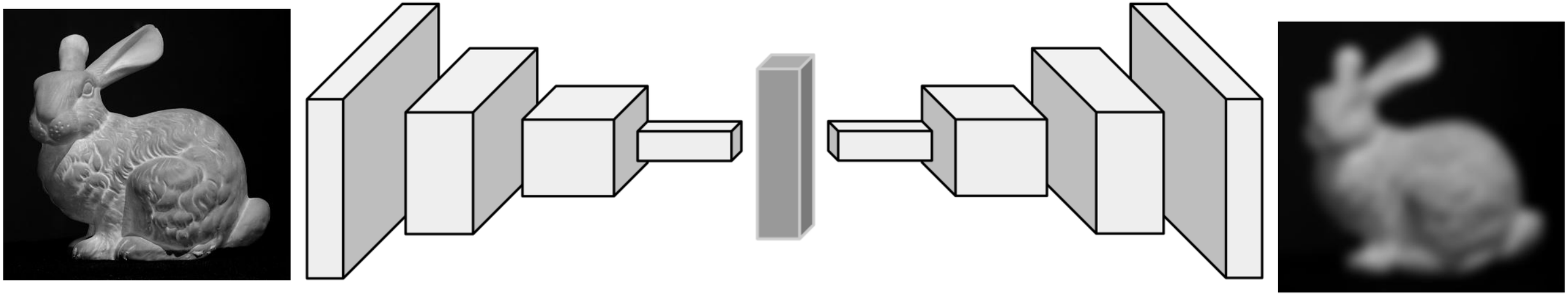
[Kingma & Welling, Auto-Encoding Variational Bayes, ICLR 2014]

Today

- Variational Autoencoders (VAE)
- **Generative Adversarial Networks (GANs)**

Variational Autoencoder

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



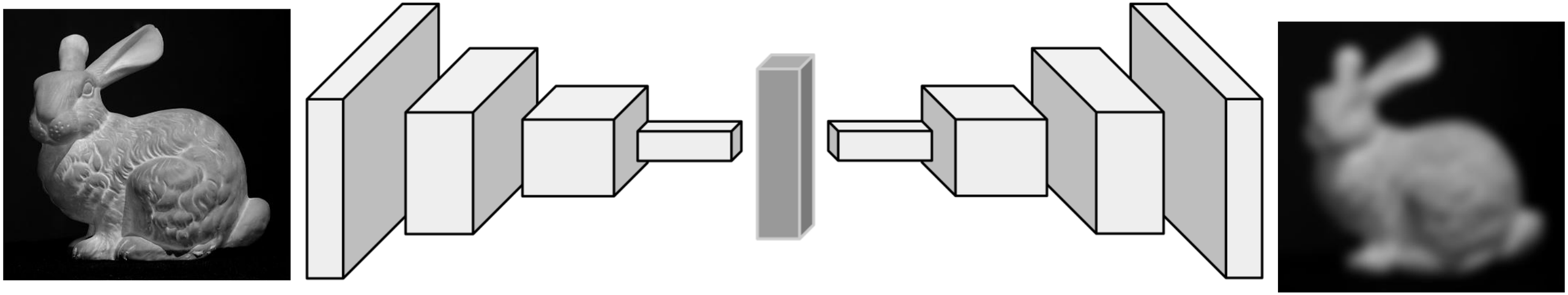
encoder $q(\mathbf{z}|\mathbf{x})$ decoder $p(\mathbf{x}|\mathbf{z})$

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

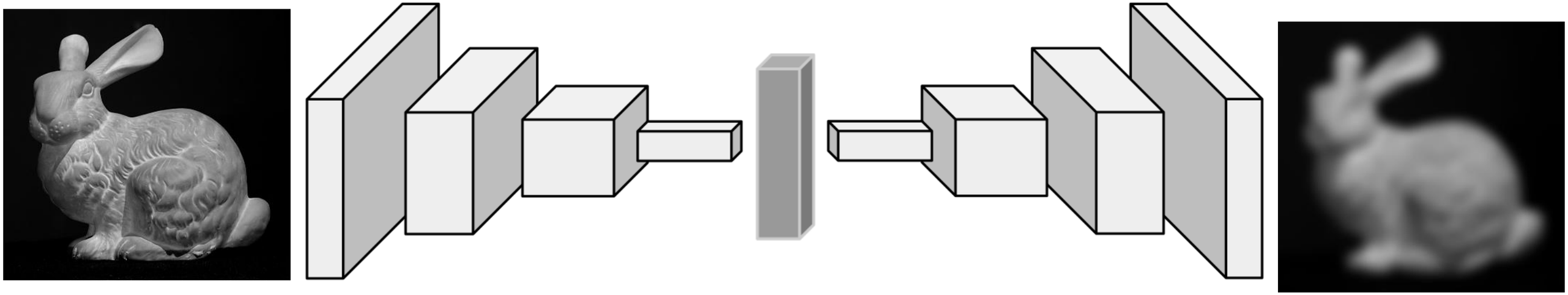
Maximize

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

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

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]

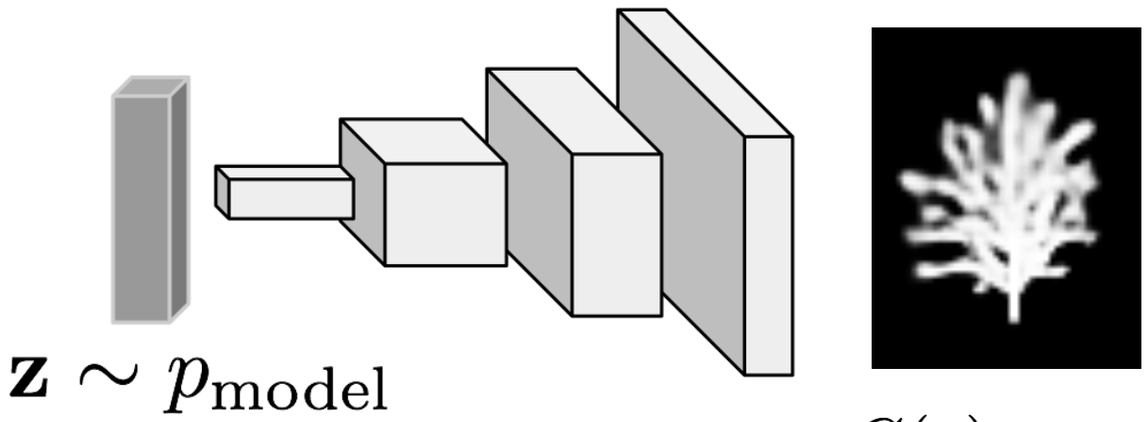
Generative Adversarial Networks (GANs)

Motivation: Learn loss function

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

Generative Adversarial Networks (GANs)

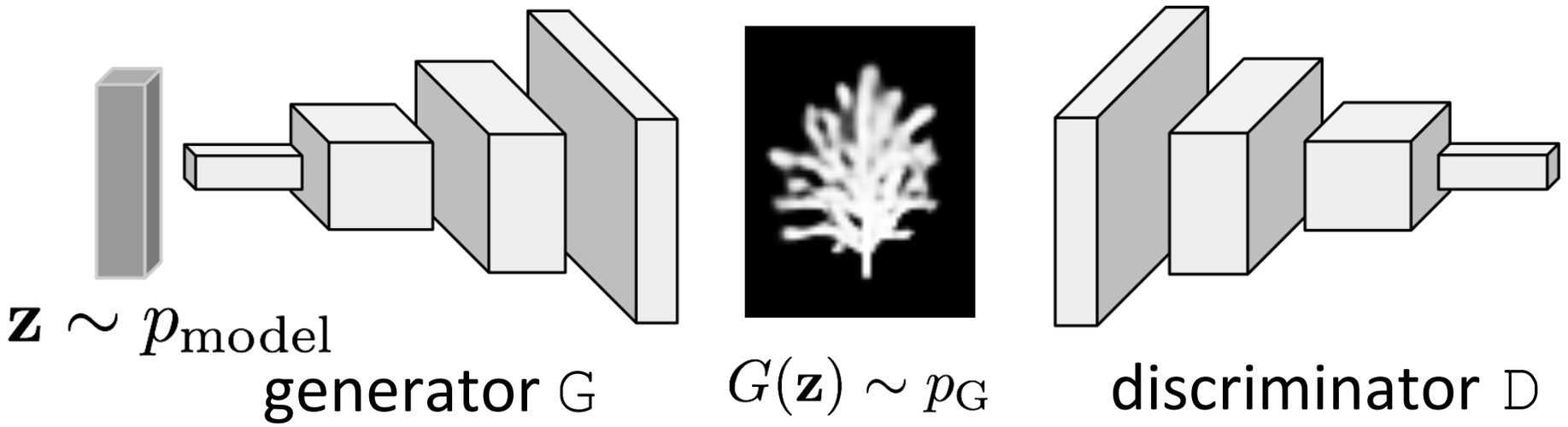
Motivation: Learn loss function



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

Generative Adversarial Networks (GANs)

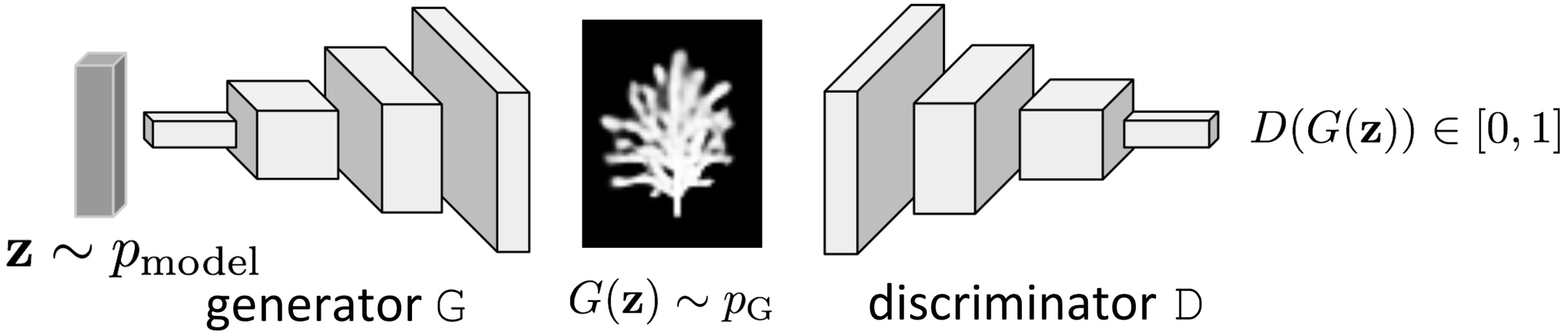
Motivation: Learn loss function



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

Generative Adversarial Networks (GANs)

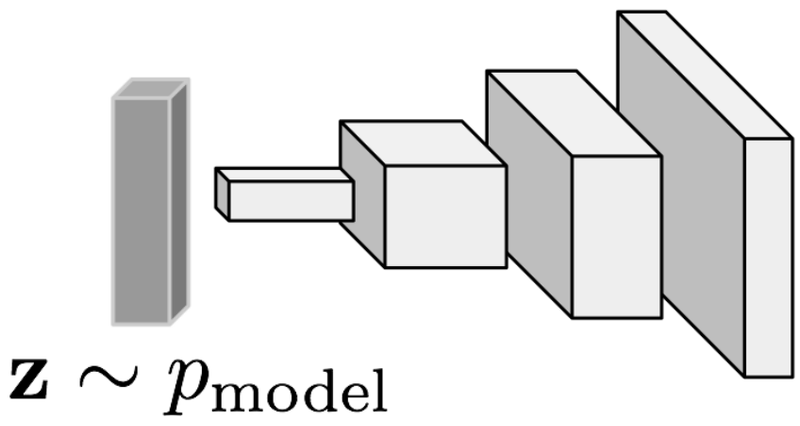
Motivation: Learn loss function



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

Generative Adversarial Networks (GANs)

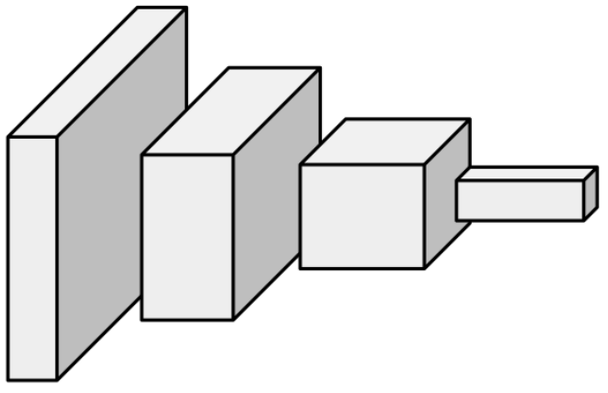
Motivation: Learn loss function



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

generator G

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

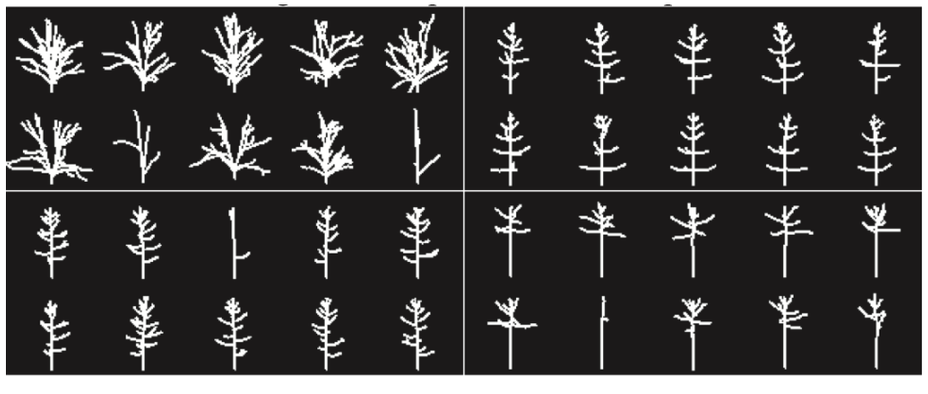


discriminator D

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

real data

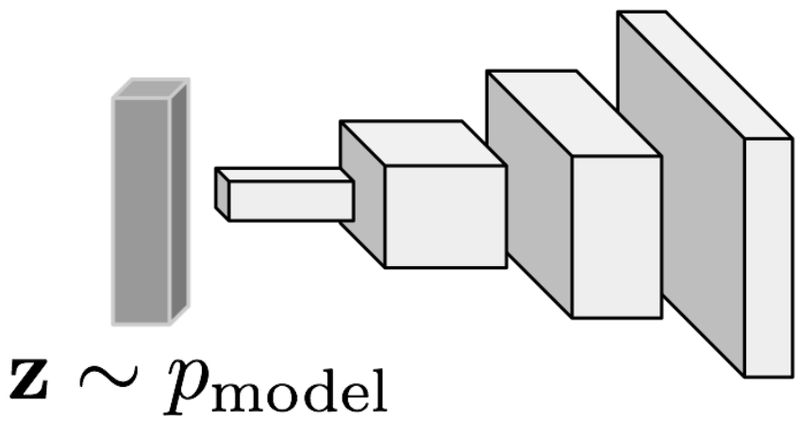
$\mathbf{x} \sim p_{\text{data}}$



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

Generative Adversarial Networks (GANs)

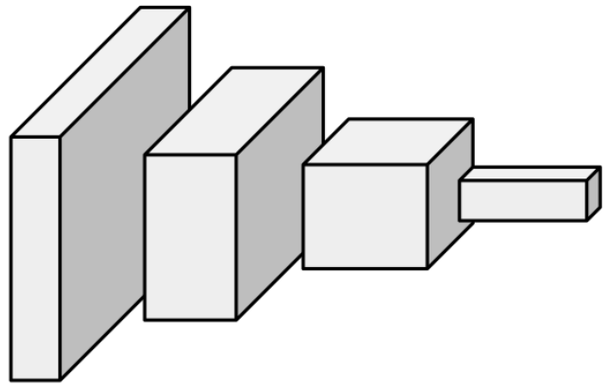
Motivation: Learn loss function



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

generator G

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



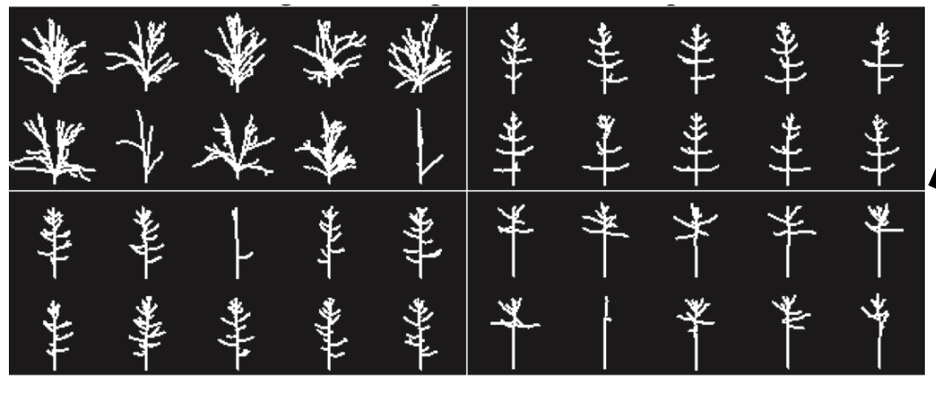
discriminator D

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

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

real data

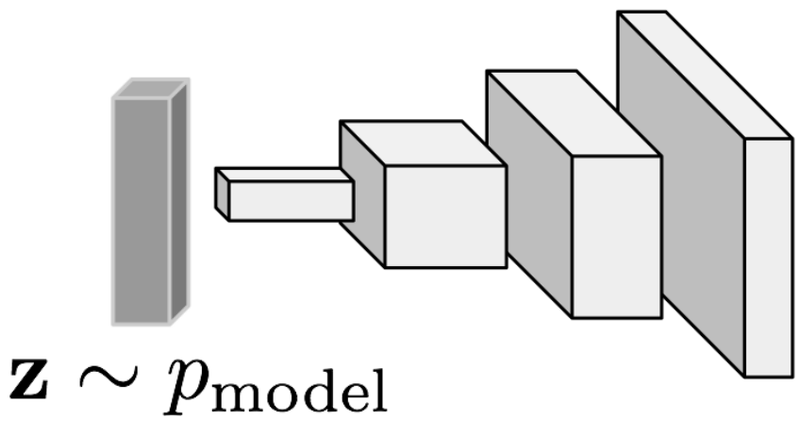
$\mathbf{x} \sim p_{\text{data}}$



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

Generative Adversarial Networks (GANs)

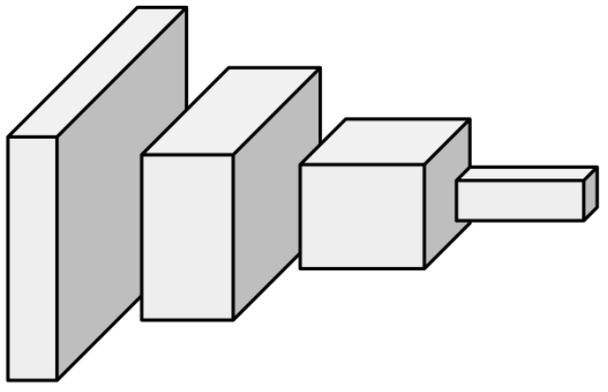
Motivation: Learn loss function



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

generator G

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



discriminator D

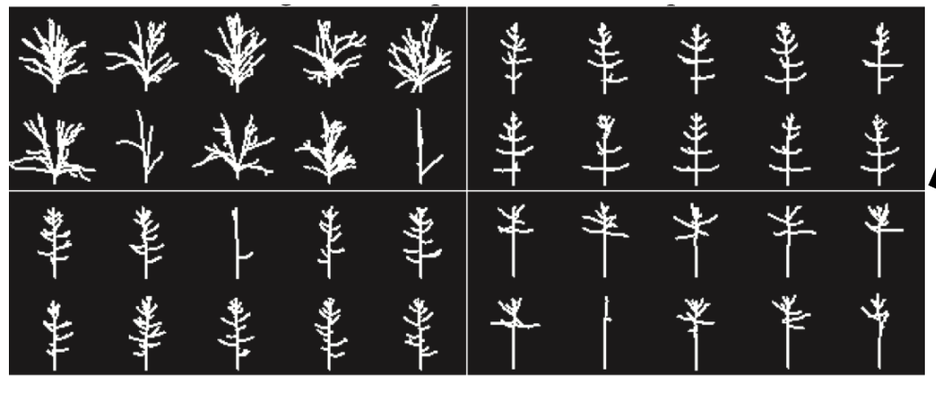
fake or real?

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

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

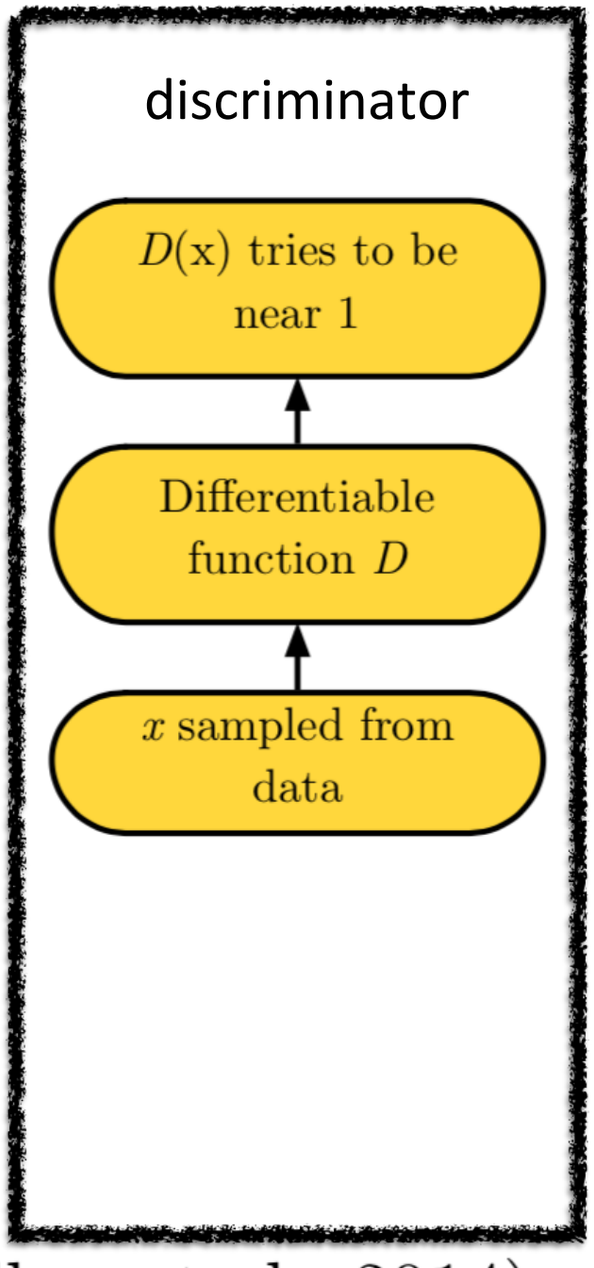
real data

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



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

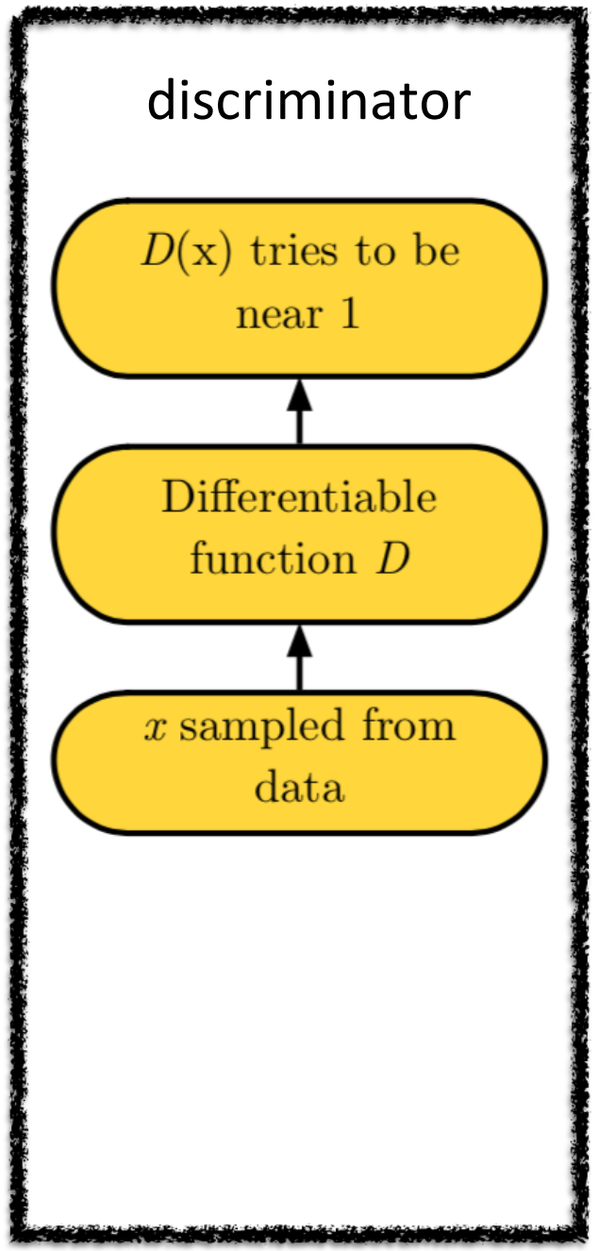
Generative Adversarial Networks (GANs)

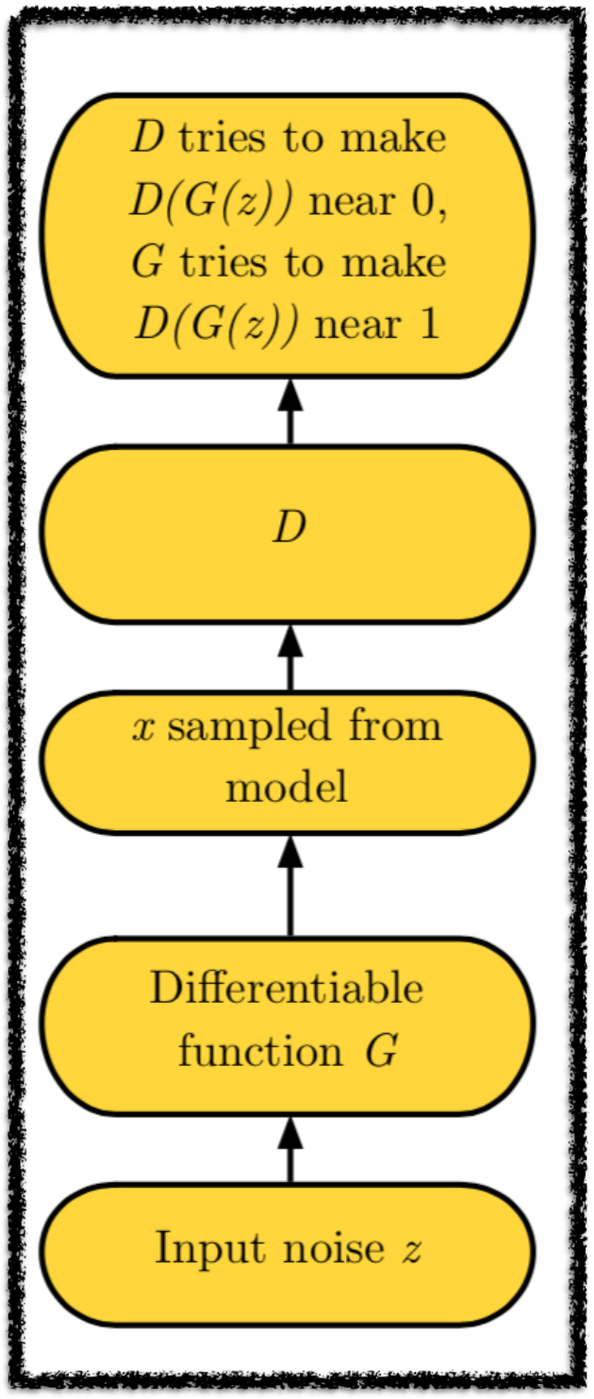


(Goodfellow et al., 2014)

slide credit: Ian Goodfellow

Generative Adversarial Networks (GANs)





(Goodfellow et al., 2014)

generator

slide credit: Ian Goodfellow

Game Theoretic View

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

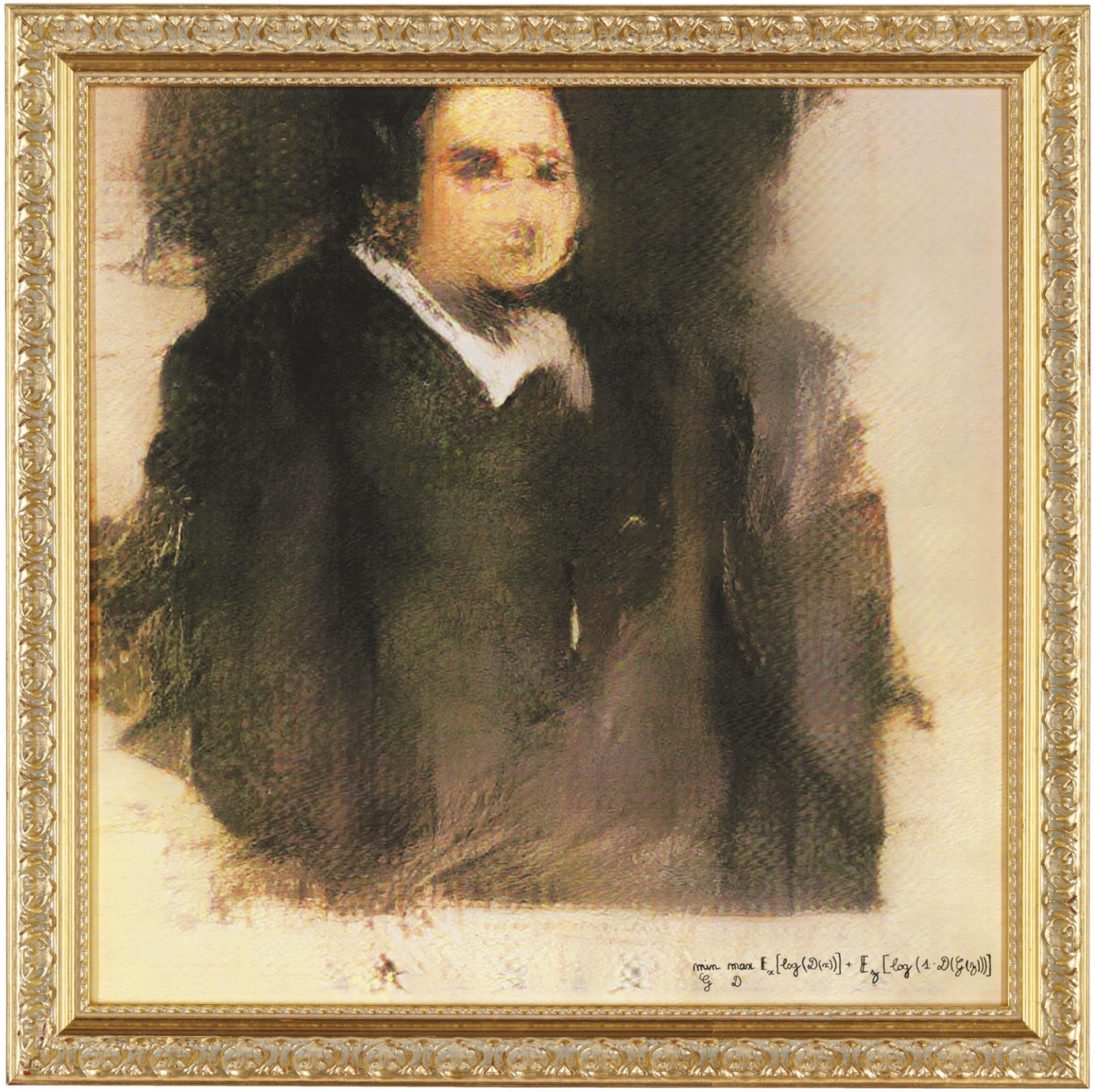
- Discriminator D tries to distinguish real and fake images
- Generator G tries to fool D, tries to make synthetic images look “as real as possible”
- G and D play zero-sum minimax game:

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

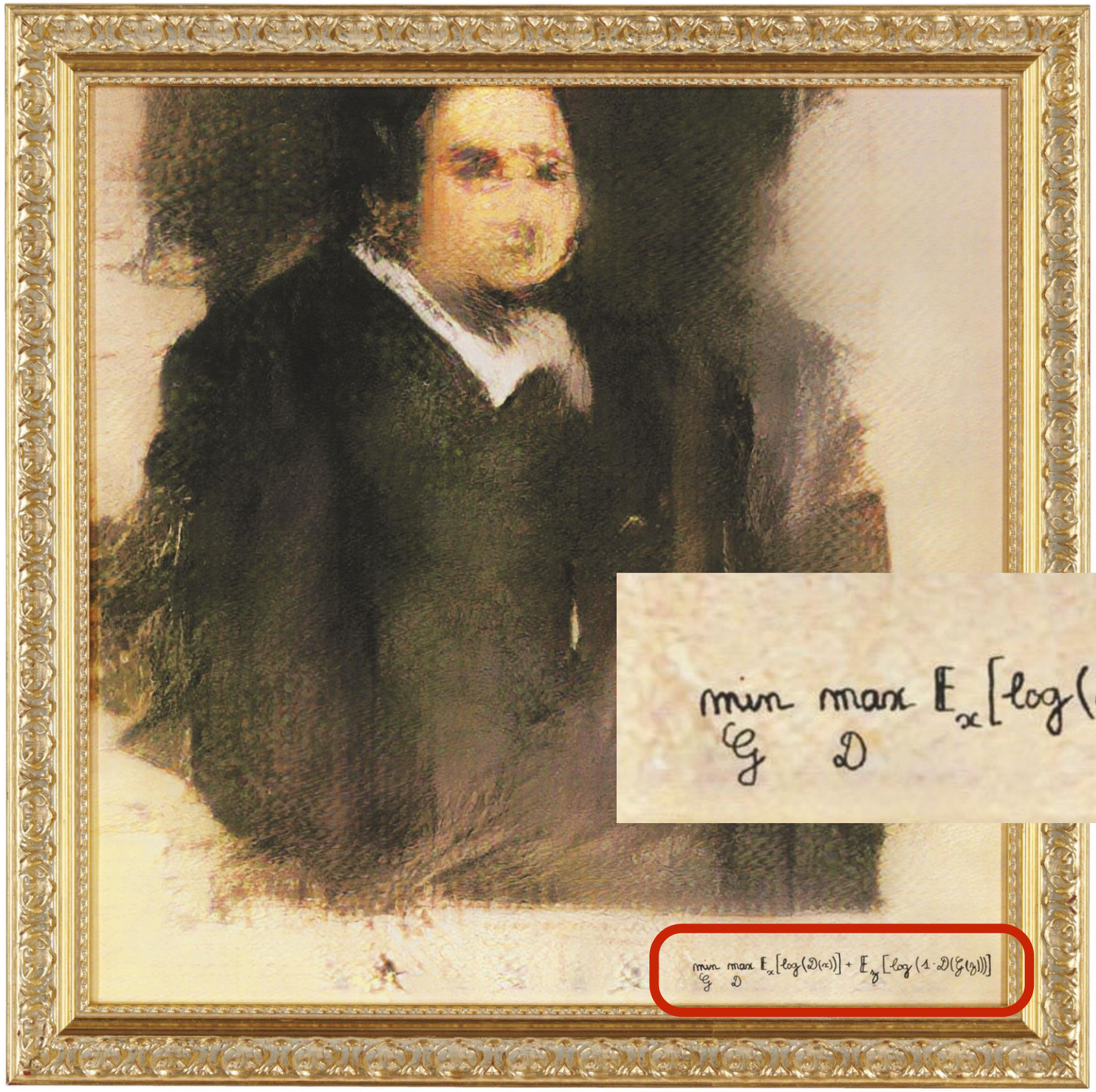
- Equilibrium / global optimum at $p_{\text{data}} = p_G$, i.e., when real and synthetic images are indistinguishable for D:

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

Generating Art Revisited



Generating Art Revisited



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
- Next lecture: More generative modeling

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

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	