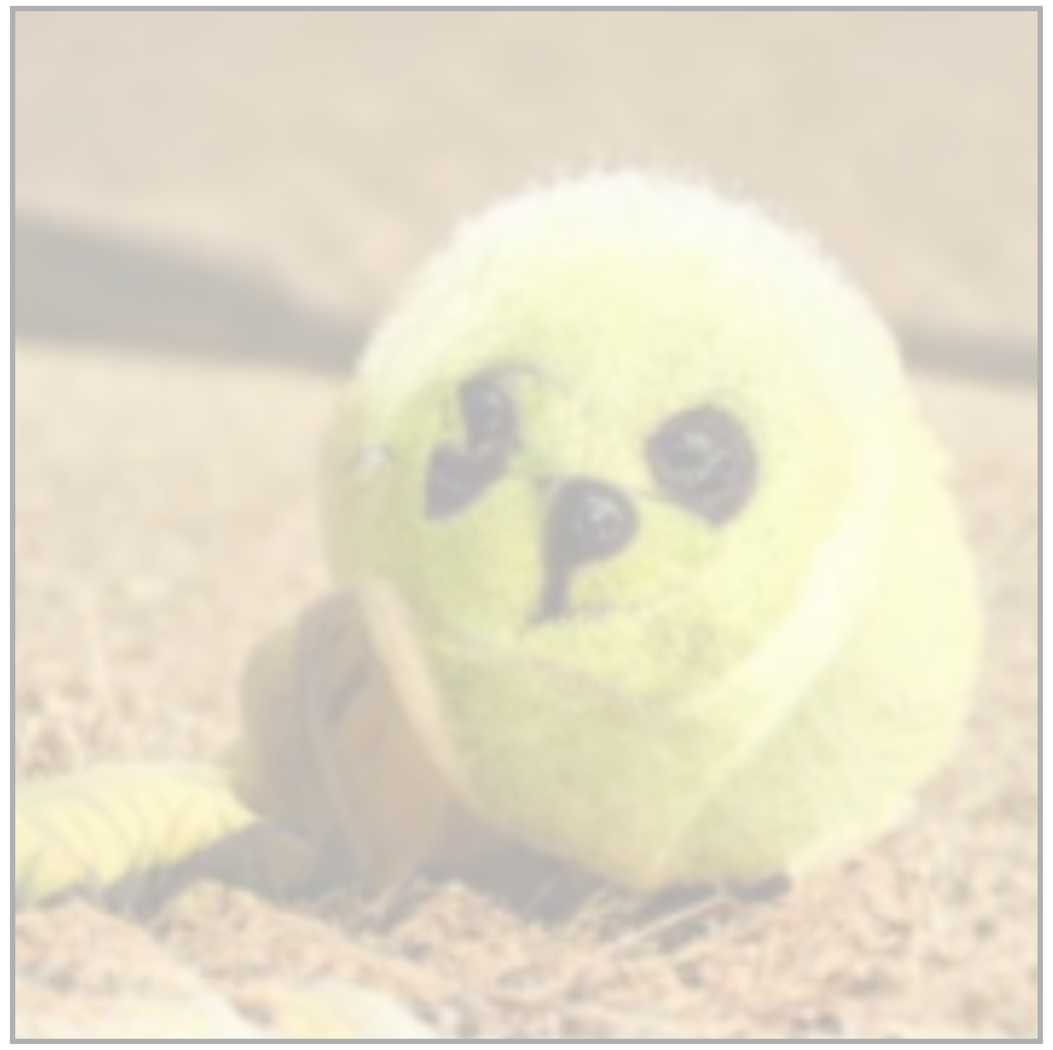
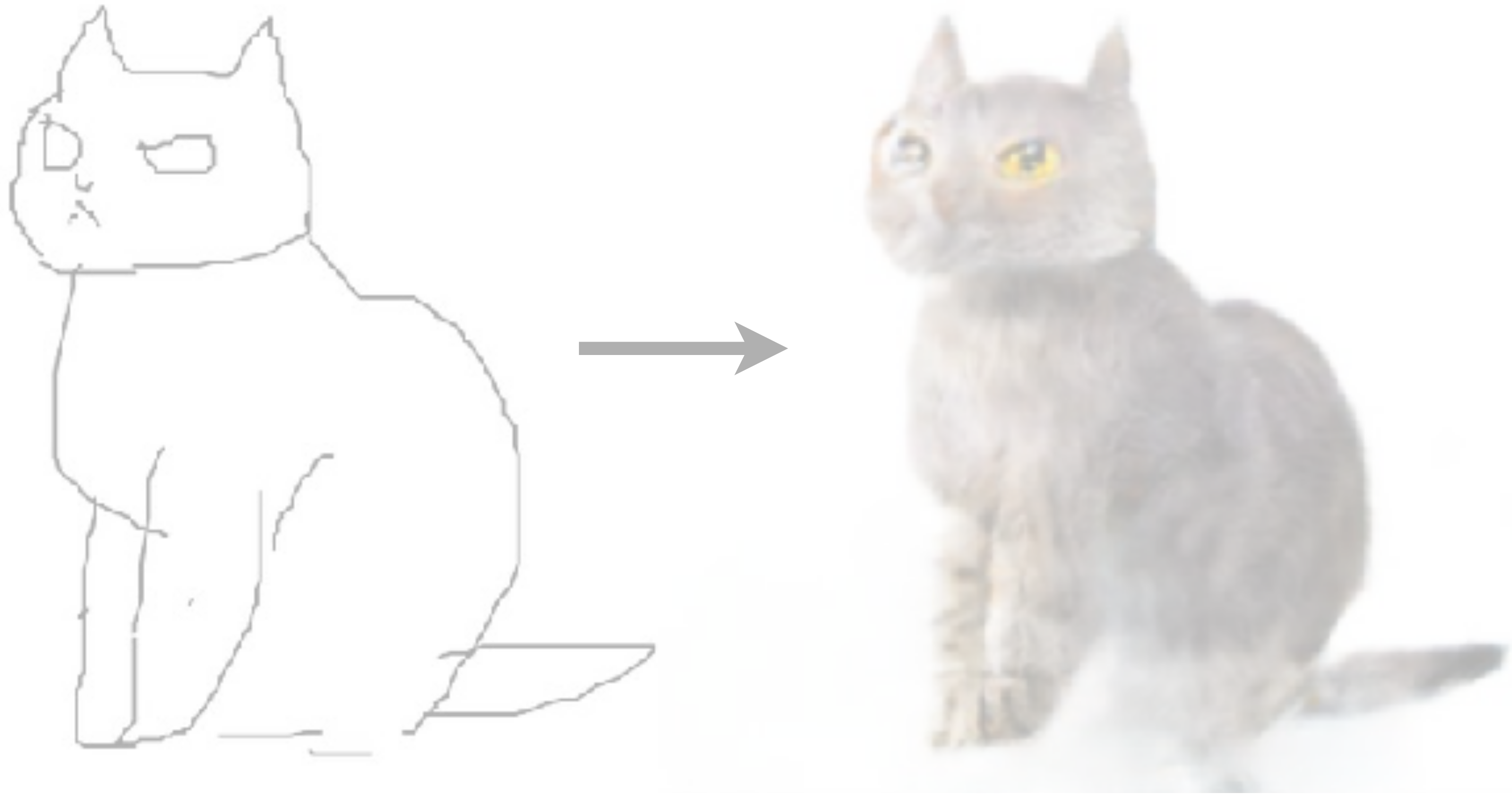




Lecture 18

Image Synthesis

Bill Freeman, Phillip Isola

Image classification

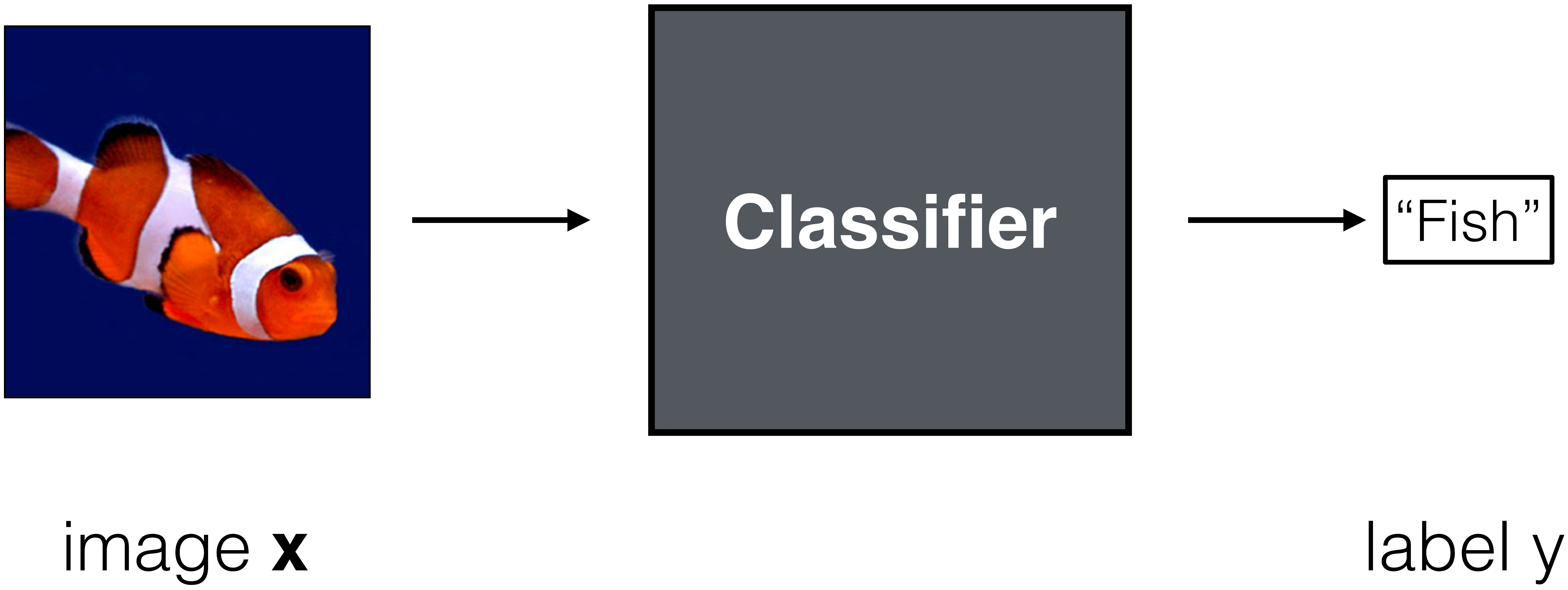


Image classification

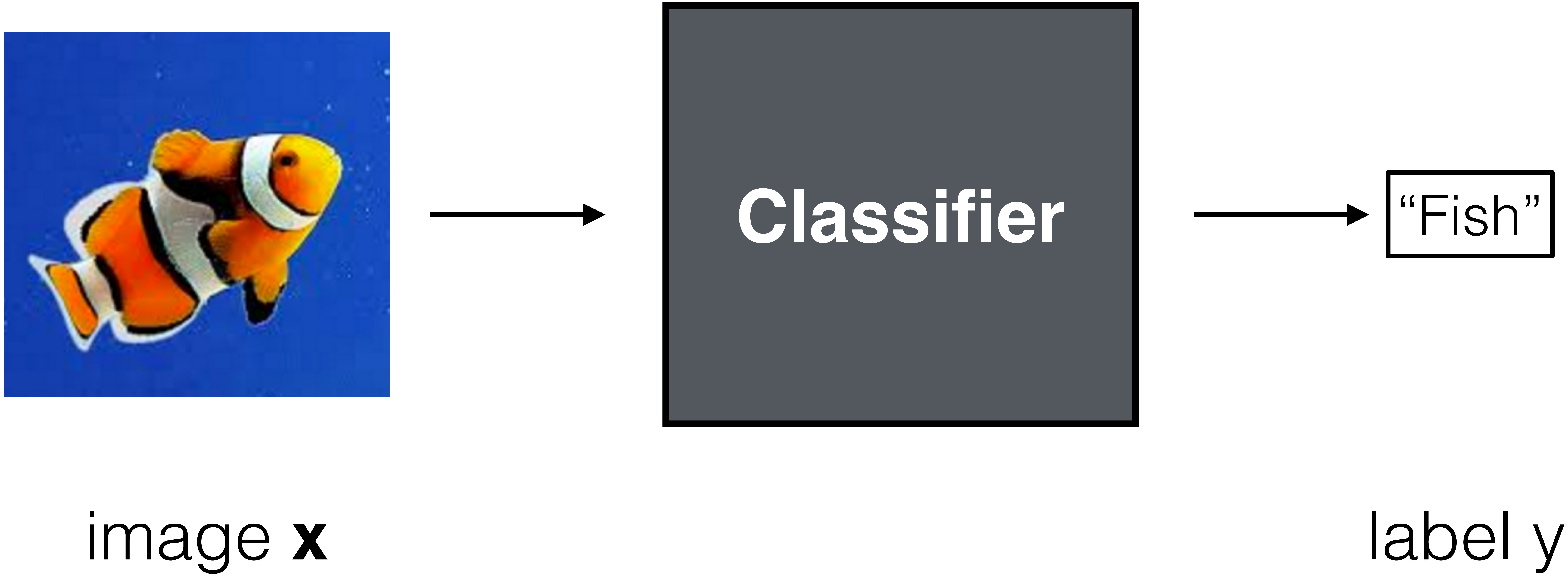


Image classification

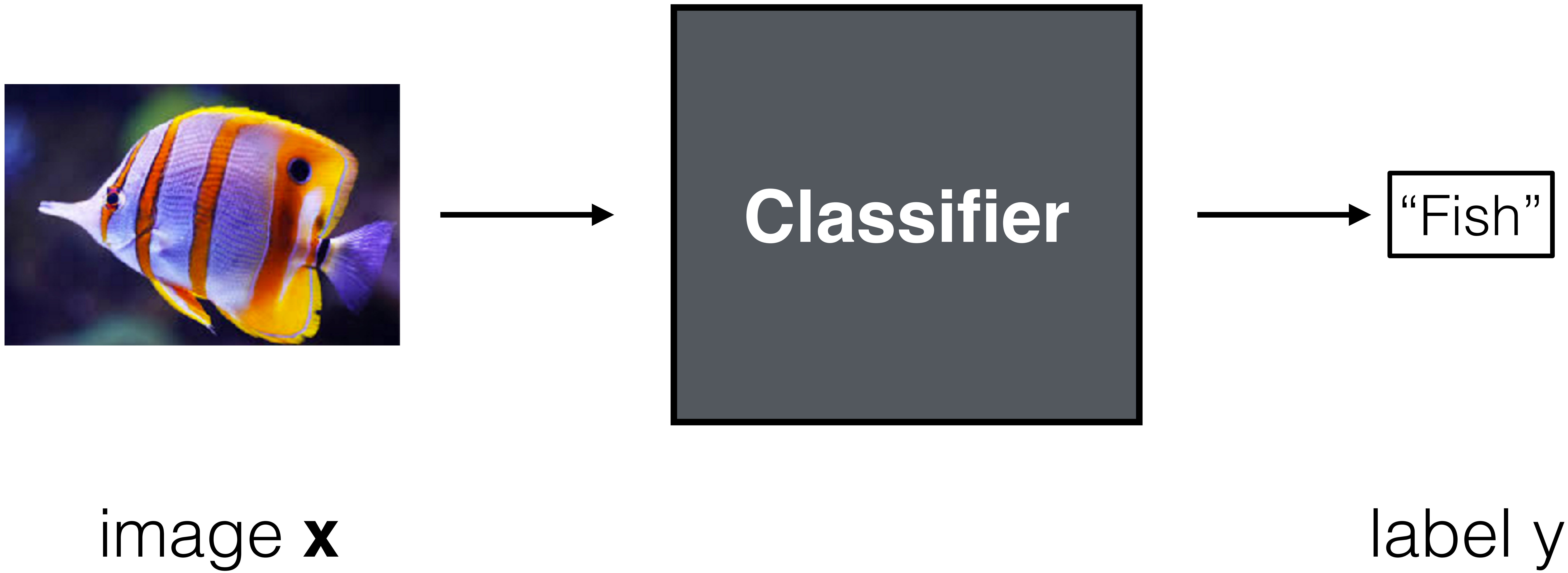


Image classification

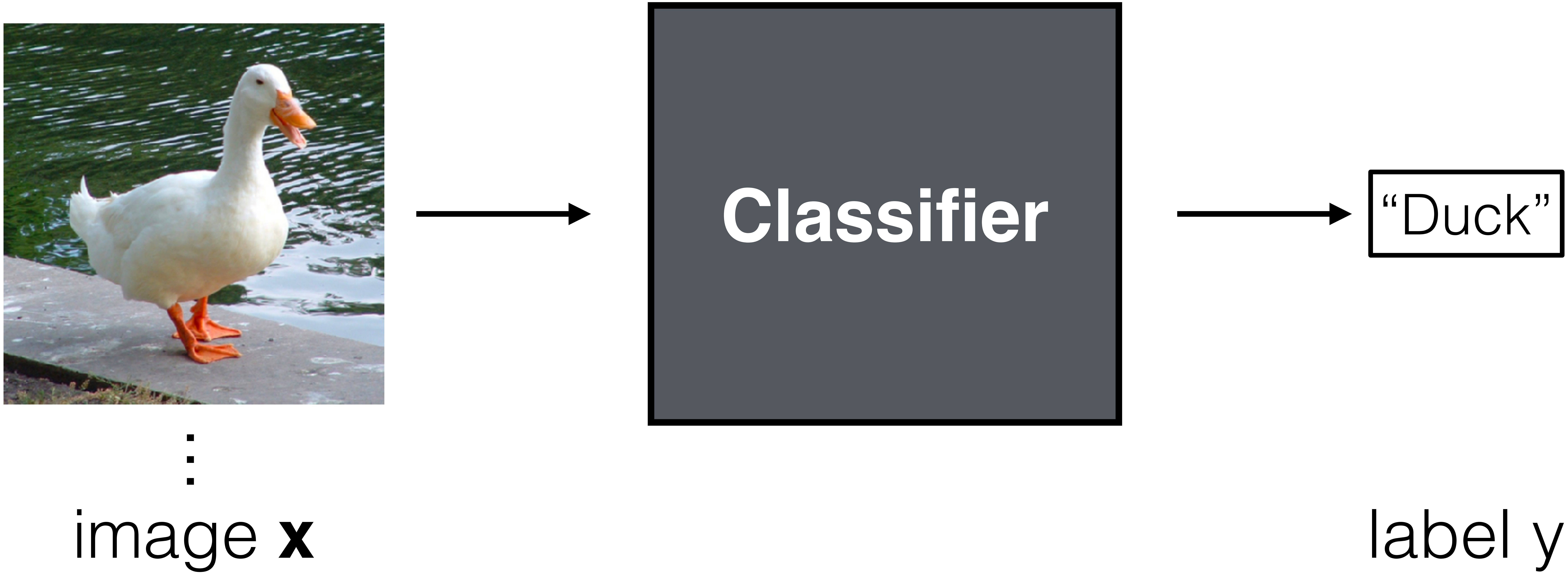


Image synthesis

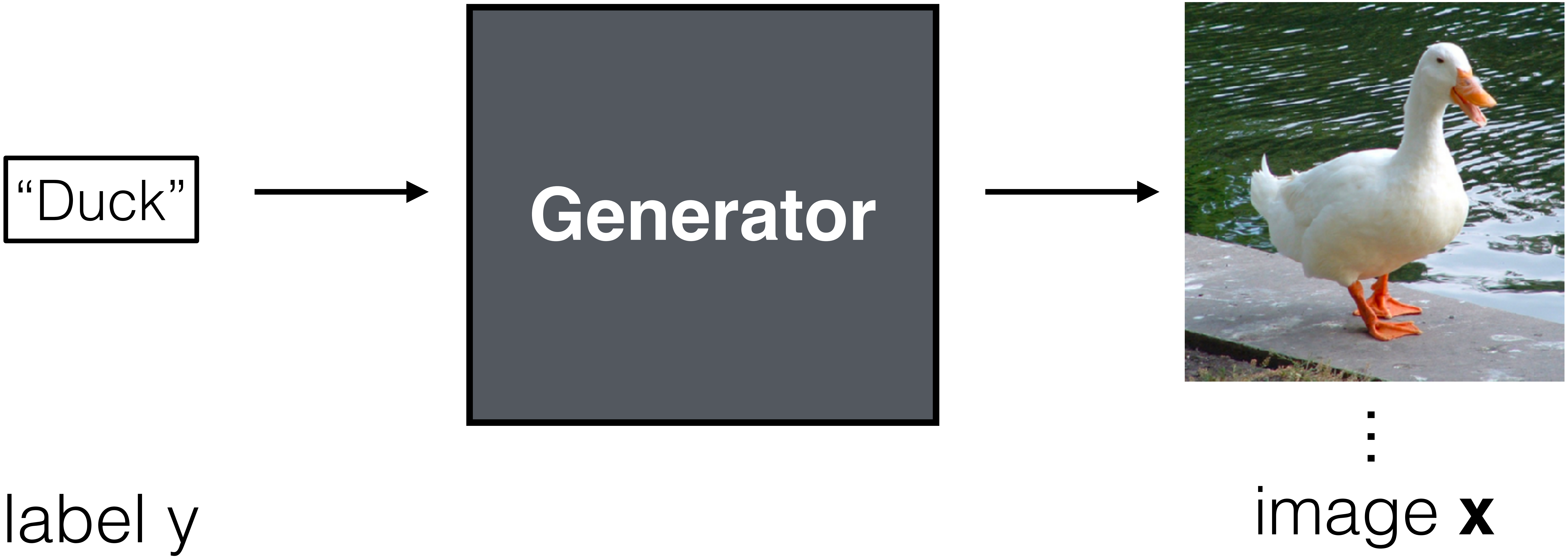


Image synthesis

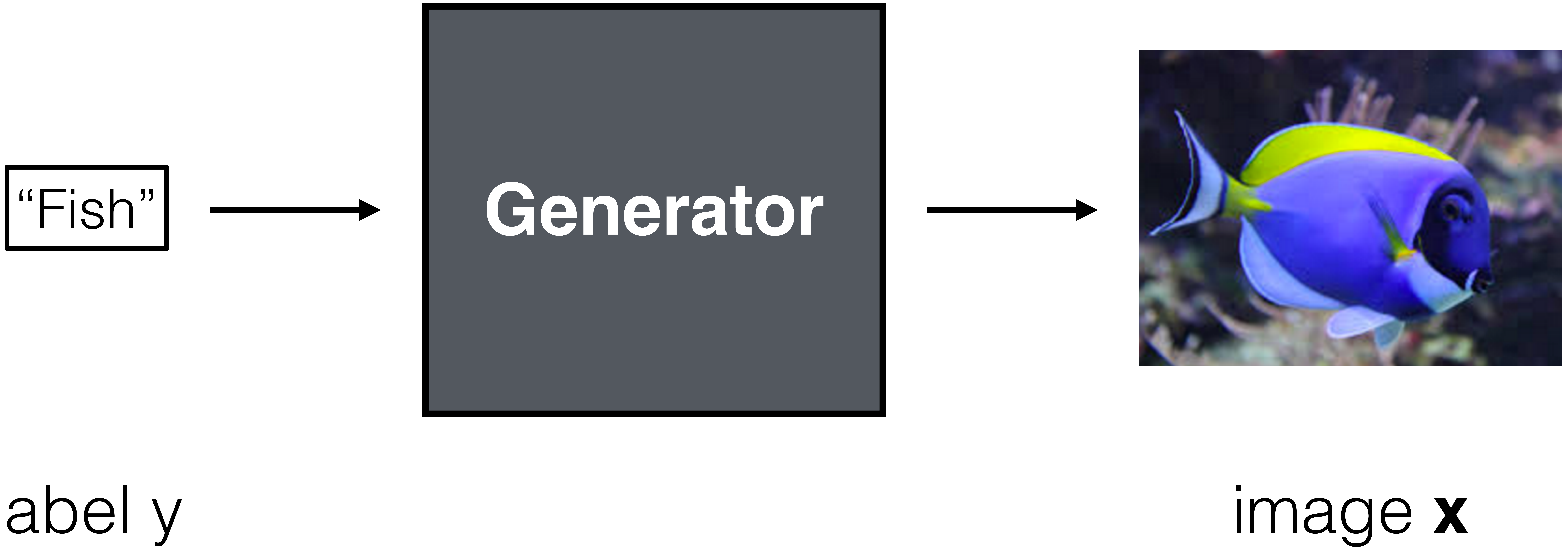


Image translation

User sketch

Photo

Image translation

Google Map

Satellite photo

Image translation

→ **Translator** →

BW image

Color image

Image synthesis via **generative modeling**

X is high-dimensional!

Model of high-dimensional structured data $P(\mathbf{X}|\mathbf{Y} = \mathbf{y})$

In vision, this is usually what we are interested in!

What can you do with generative models?

1. Image synthesis

2. Structured prediction

3. Domain mapping

4. (Representation learning)

5. (Model-based intelligence)

1. **Image synthesis**
2. Structured prediction
3. Domain mapping

[Images: https://ganbreeder.app/]

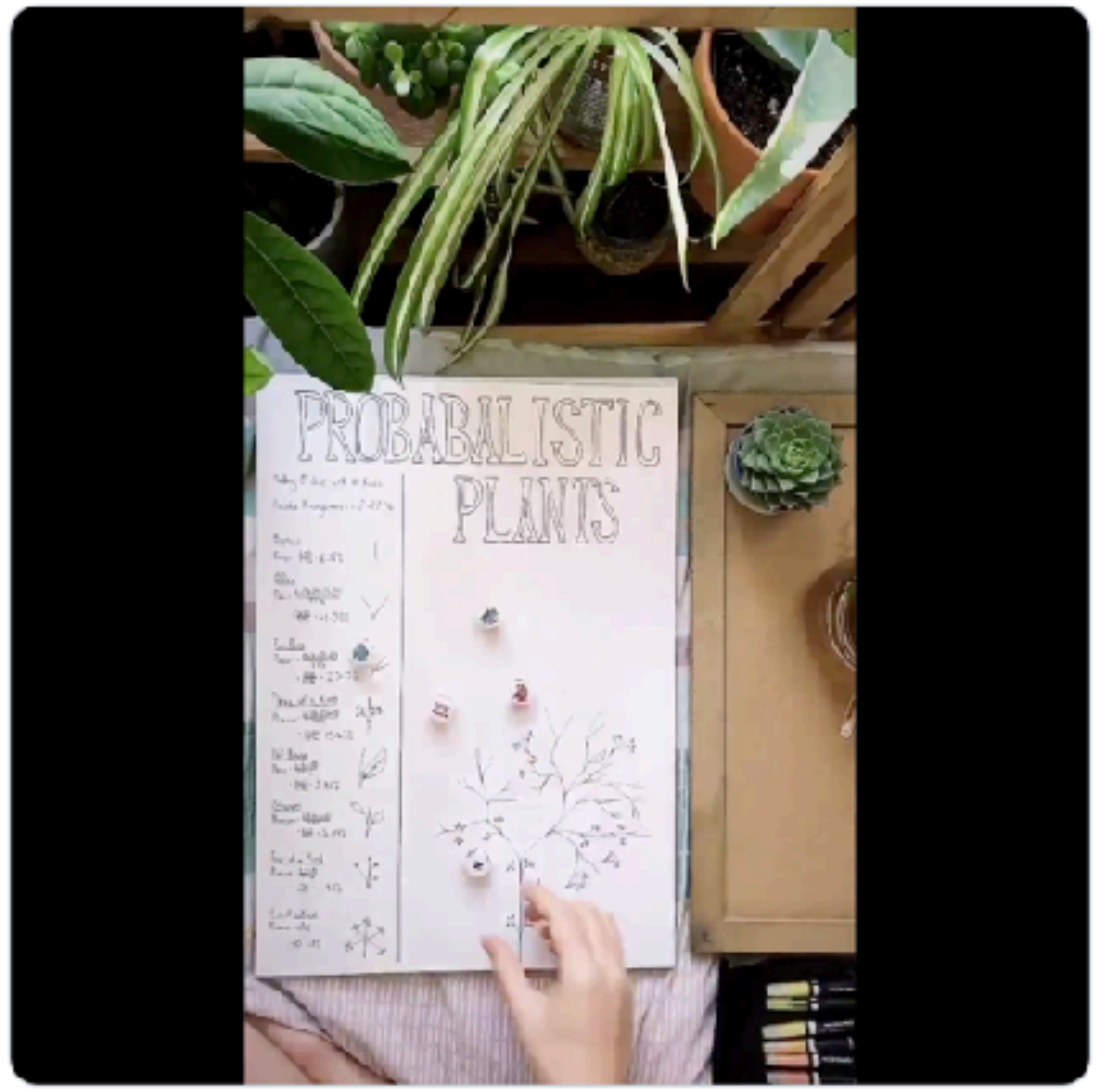
Image synthesis

Procedural graphics

[Anders Scheil]

Ayliean @Ayliean · Nov 17

Made up a set of rules and rolled some dice to decide how this plant would grow. I never did get that five of a kind, as expected, but I was still hopeful! 🌱🍀☘️



52 1.1K 4.5K

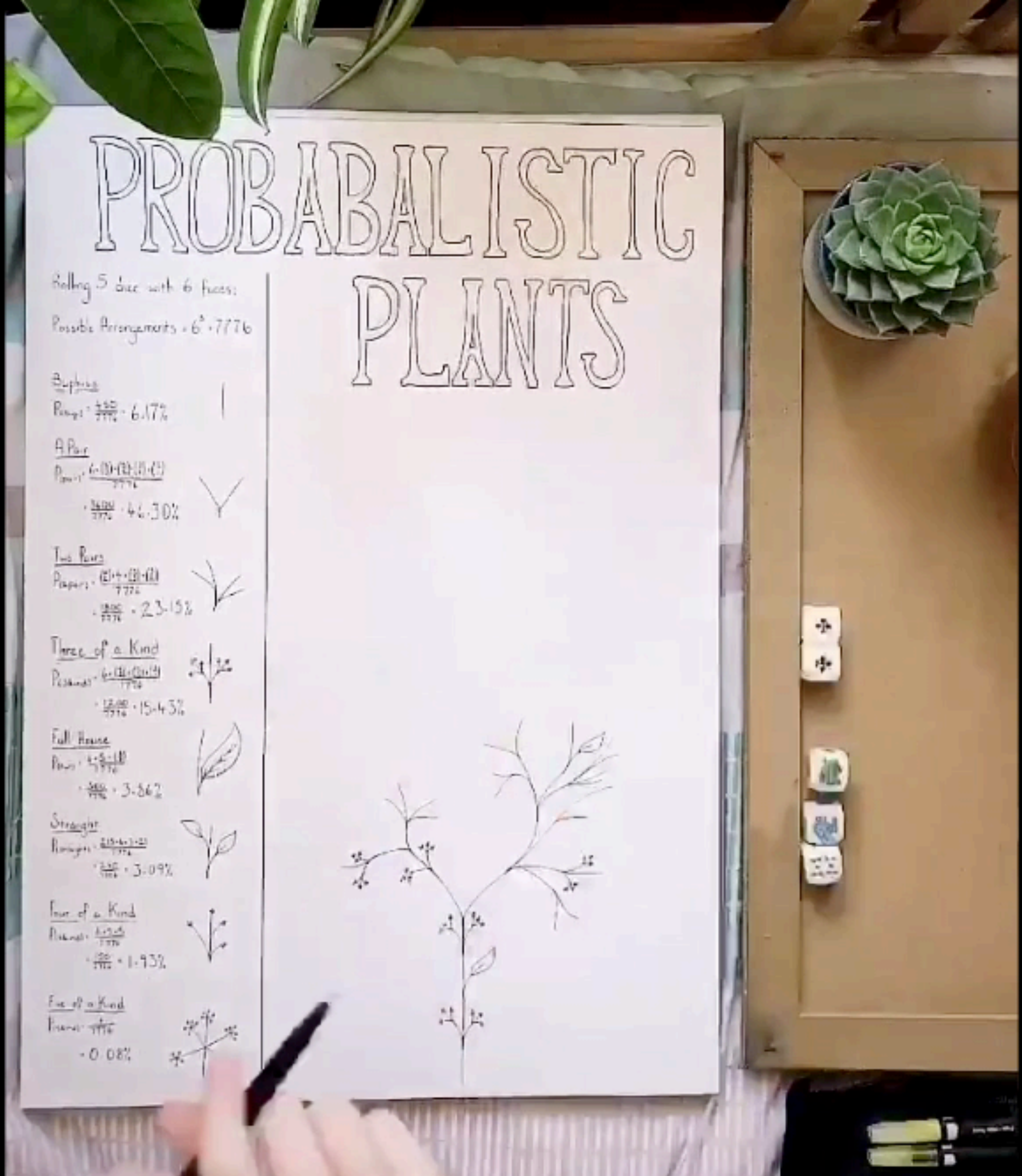
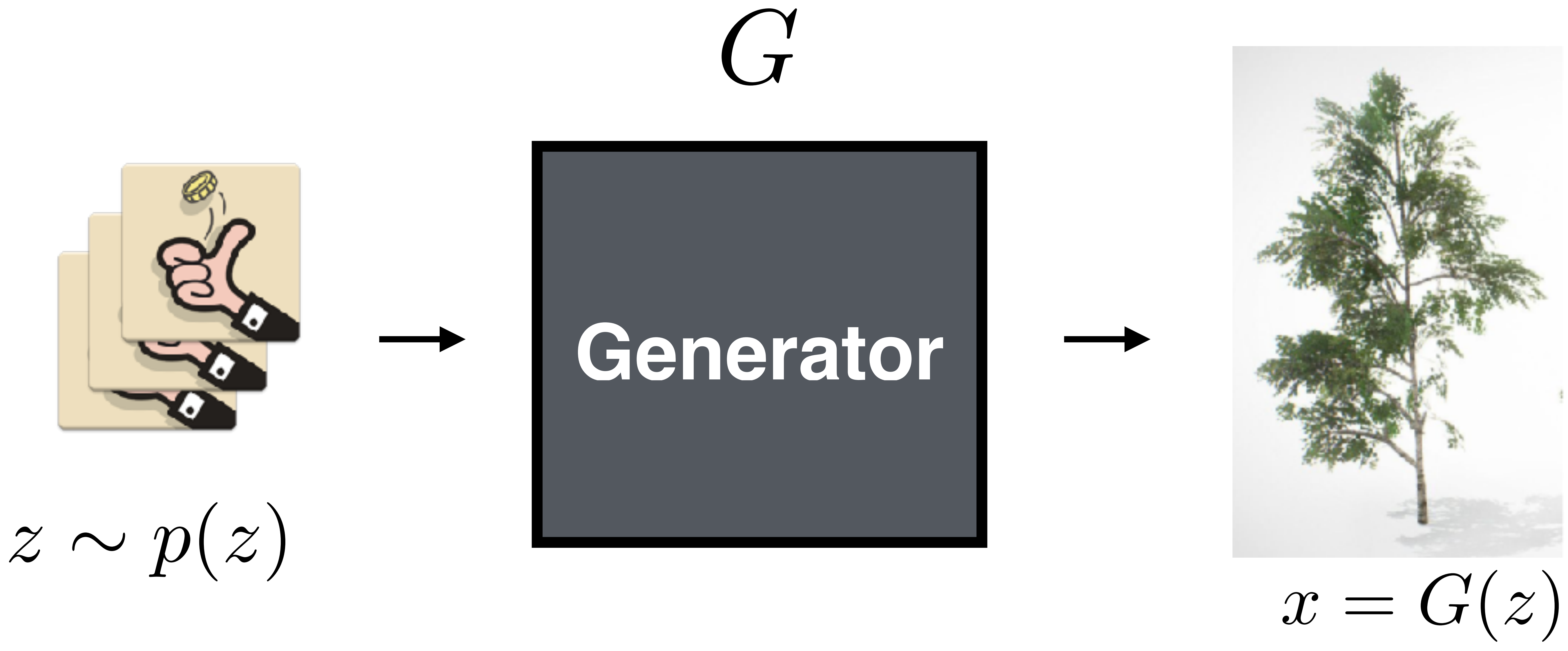


Image synthesis from “noise”



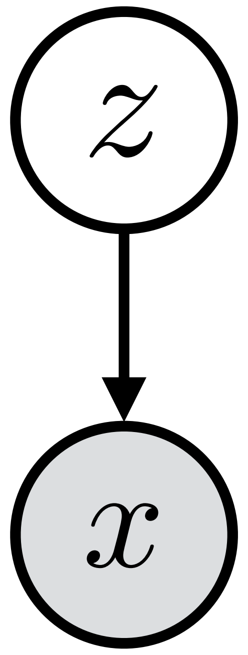
Sampler

$$G : \mathcal{Z} \rightarrow \mathcal{X}$$

$$z \sim p(z)$$

$$x = G(z)$$

Learning a generative model



Input samples

$\rightarrow$

Learner

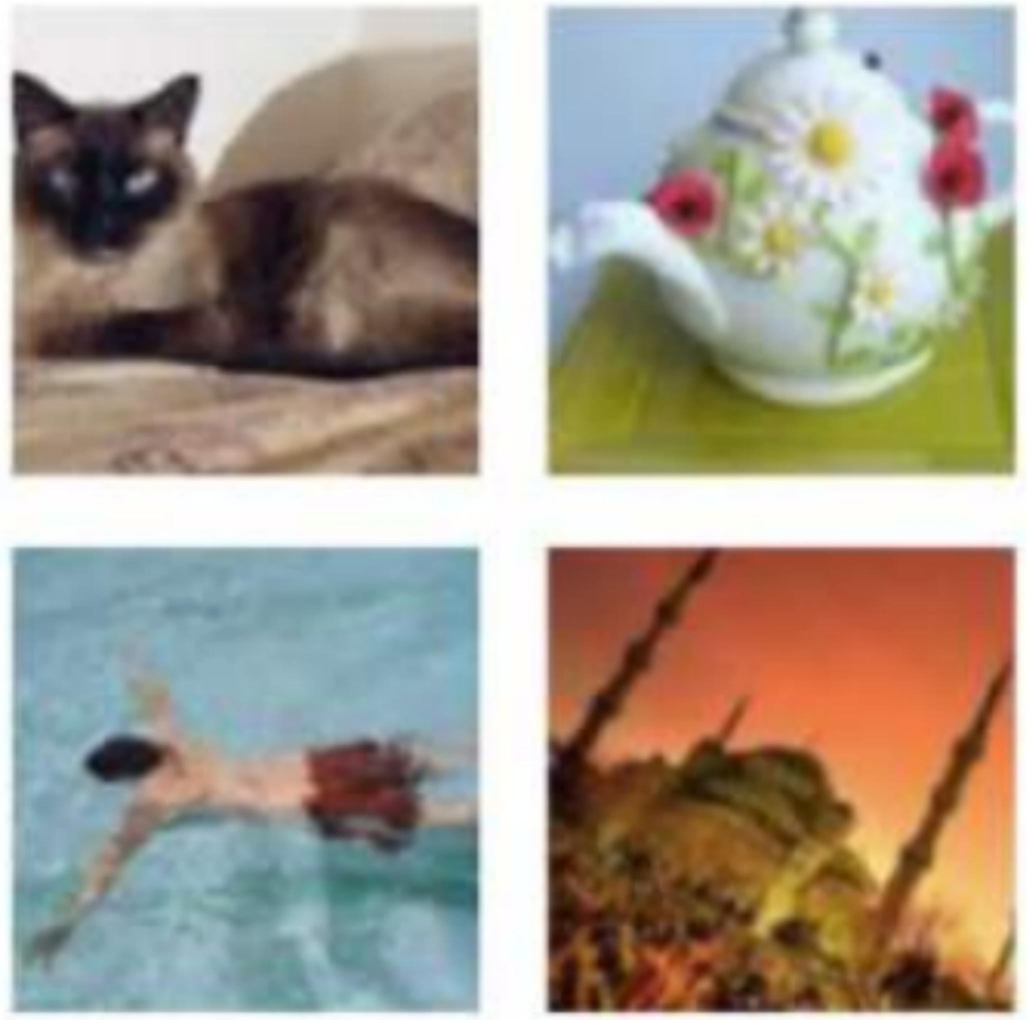
Objective

Hypothesis space

Optimizer

$\rightarrow$ z $\rightarrow$

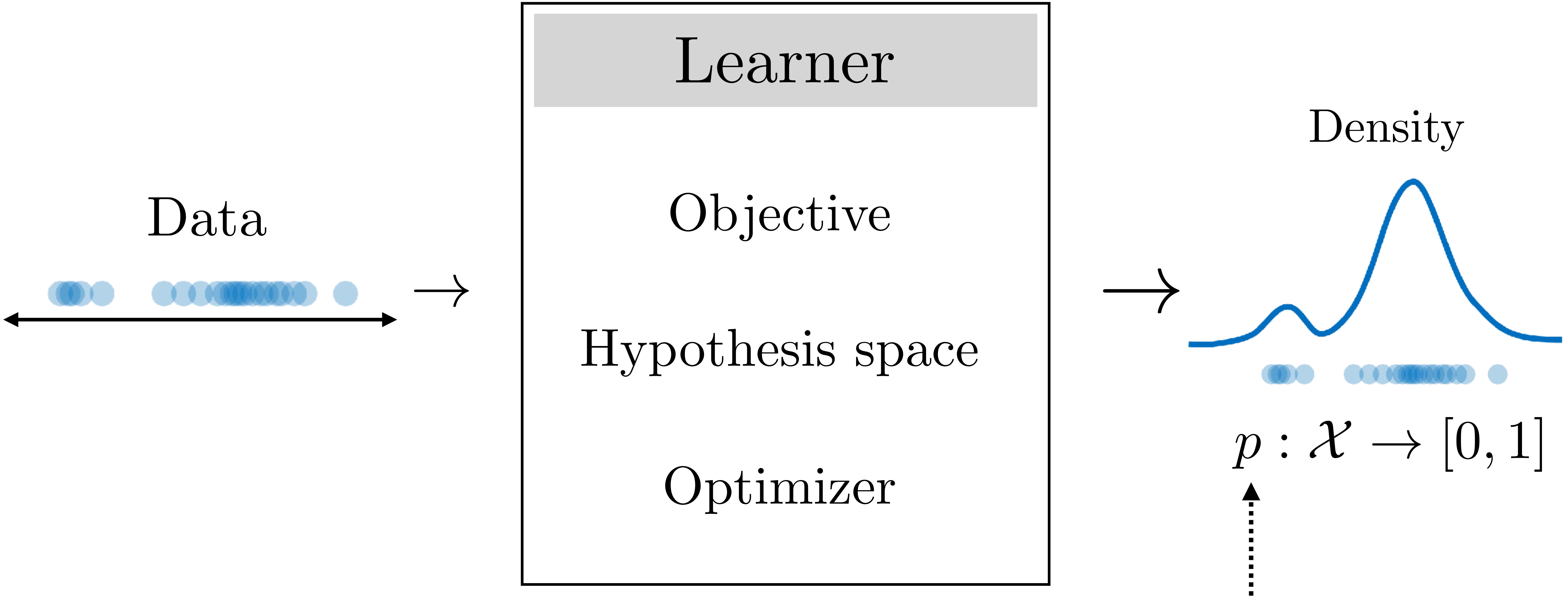
latent variables



Generated samples

[figs modified from: http://introtodeeplearning.com/materials/2019_6S191_L4.pdf]

Learning a density model



Normalized distribution
(some models output unormalized *energy functions*)

[figs modified from: http://introtodeeplearning.com/materials/2019_6S191_L4.pdf]

Case study #1: Fitting a Gaussian to data

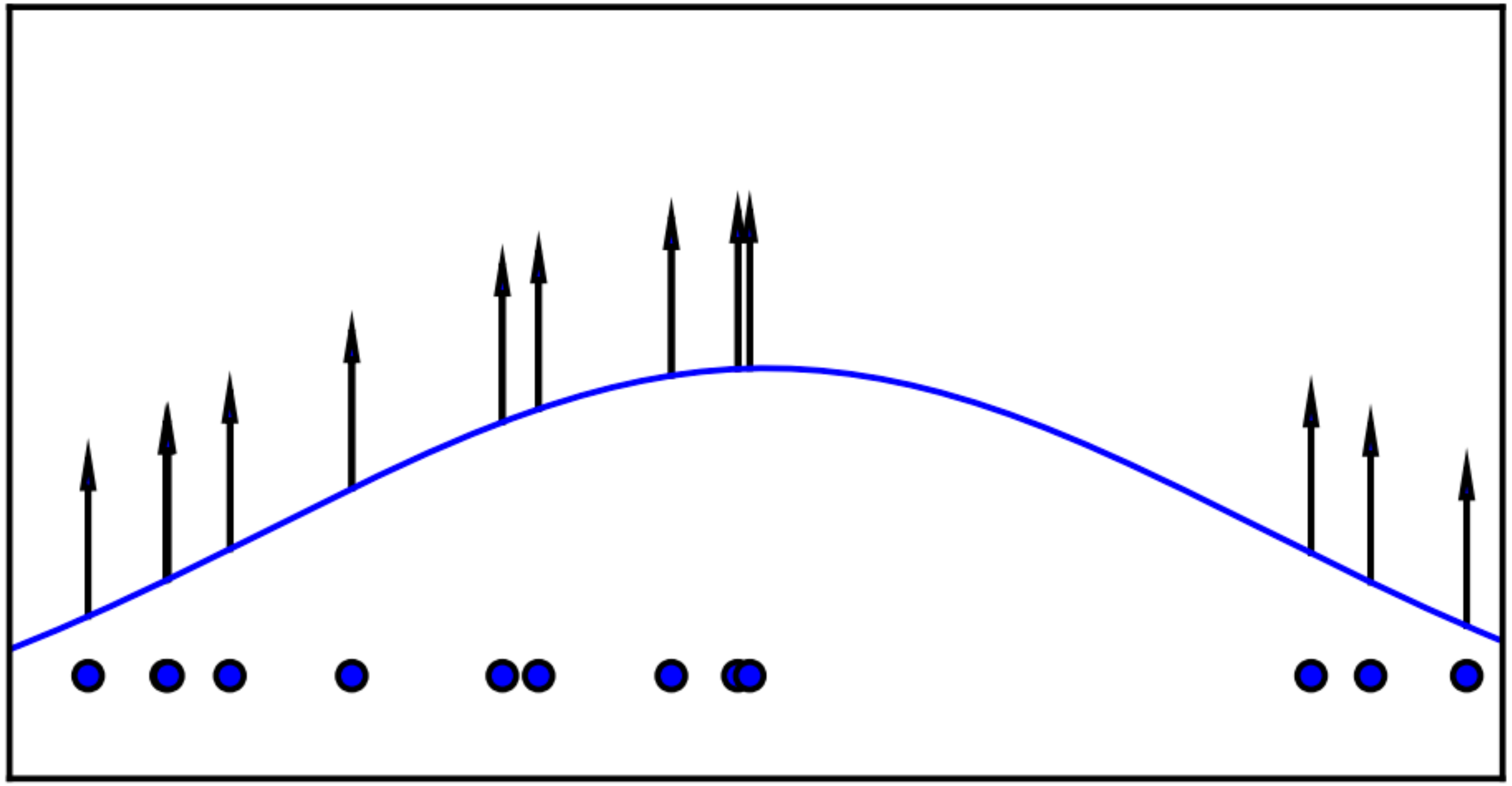


fig from [Goodfellow, 2016]

Max likelihood objective

$$\max_{\theta} \mathbb{E}_{x \sim p_{\texttt{data}}}[\log p_{\theta}(x)]$$

Considering only Gaussian fits

$$p_{\theta}(x) = \mathcal{N}(x; \mu, \sigma)$$

$$\theta = [\mu, \sigma]$$

Closed form optimum:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} \quad \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu)^2$$

Case study #1: Fitting a Gaussian to data

Data $\rightarrow$

Learner

"max likelihood"

Objective

$$\max_{\theta} \mathbb{E}_{x \sim p_{\texttt{data}}}[\log p_{\theta}(x)]$$

Hypothesis space

$$p(x) = \mathcal{N}(x; \mu, \sigma)$$

Optimizer

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x^{(i)} \qquad \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x^{(i)} - \mu)^2$$

$\rightarrow$ Density

$$p : \mathcal{X} \rightarrow [0, 1]$$

Case study #2: learning a deep generative model

Data $\rightarrow$

Learner

Objective
Usually max likelihood

Hypothesis space
Deep net

Optimizer
SGD

$\rightarrow$ Density

$p : \mathcal{X} \rightarrow [0, 1]$

Case study #2: learning a deep generative model

Data $\rightarrow$

Learner

Objective
Usually max likelihood

Hypothesis space
Deep net

Optimizer
SGD

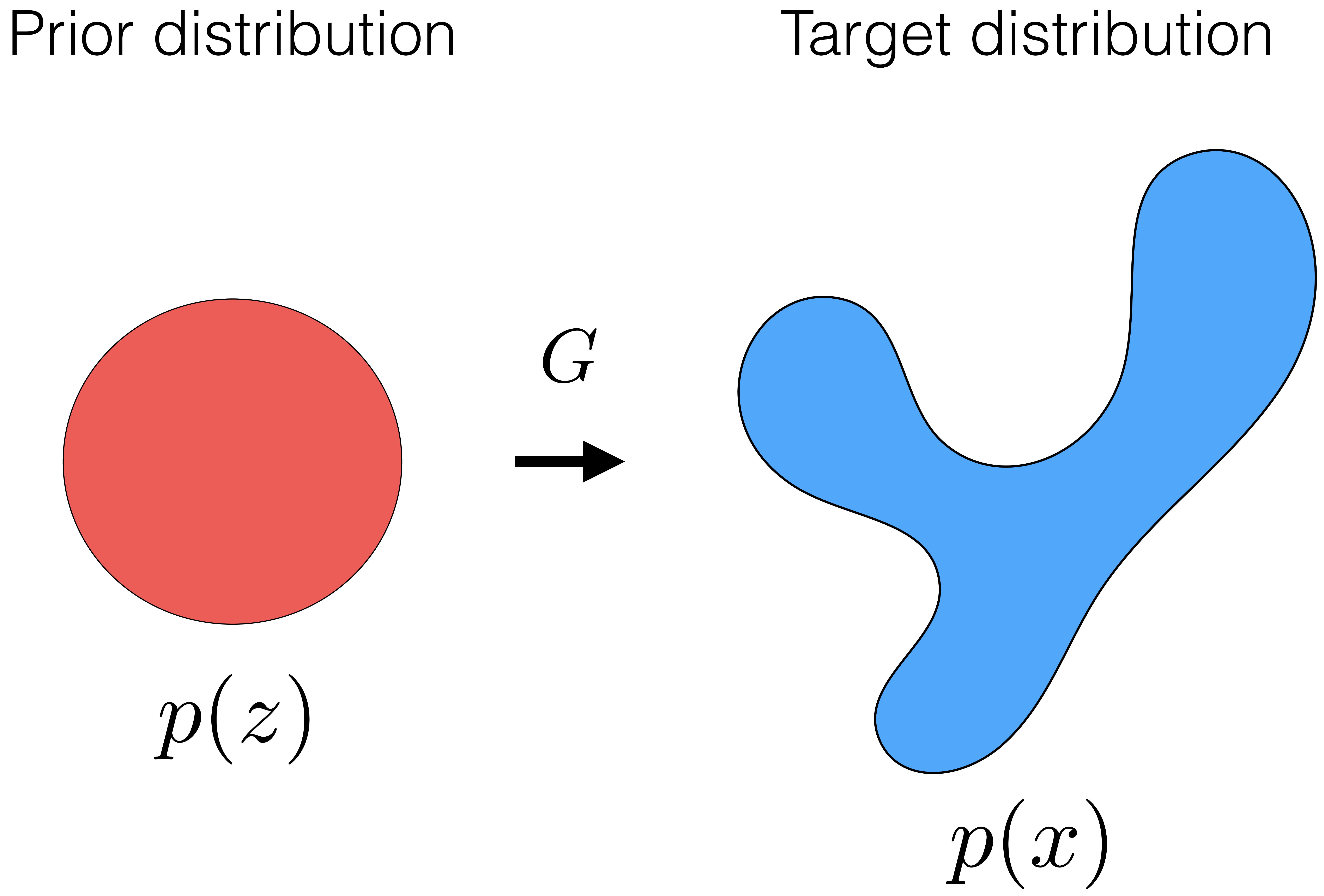
$\rightarrow$

Density
$p : \mathcal{X} \rightarrow [0,1]$

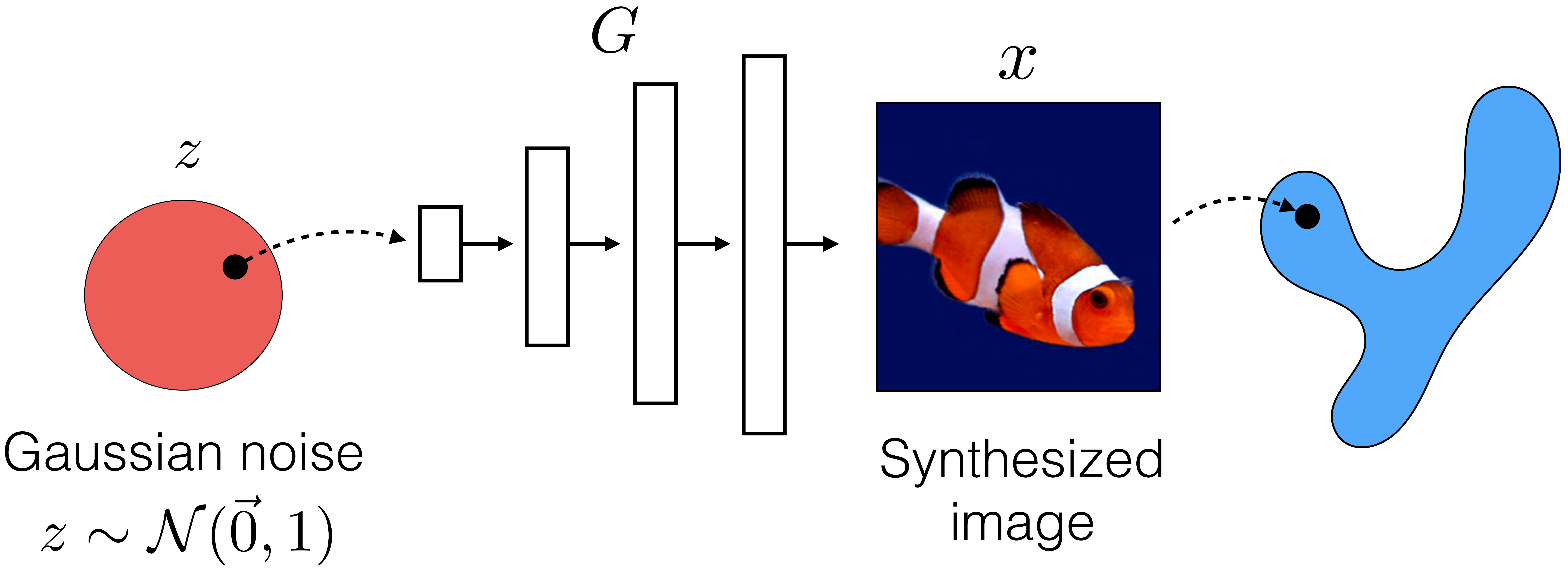
Sampler
$G : \mathcal{Z} \rightarrow \mathcal{X}$
$z \sim p(z)$
$x = G(z)$

Models that provide a sampler but no density are called **implicit generative models**

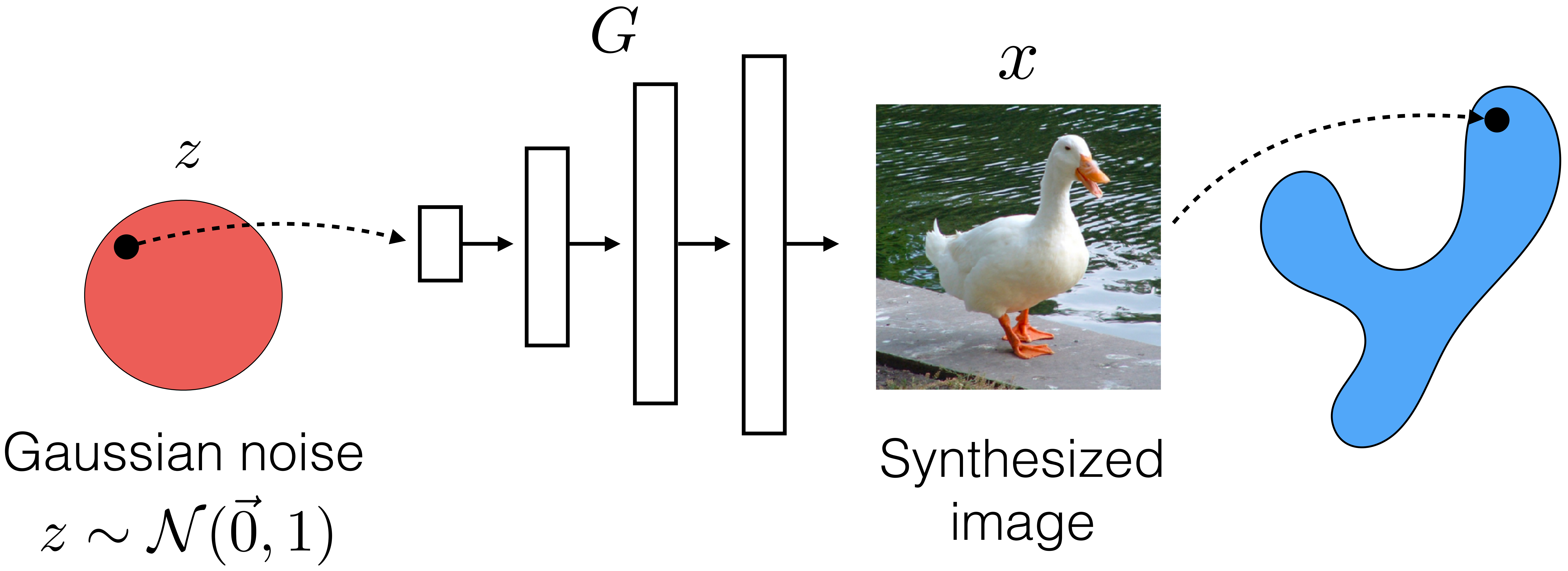
Deep generative models are distribution transformers



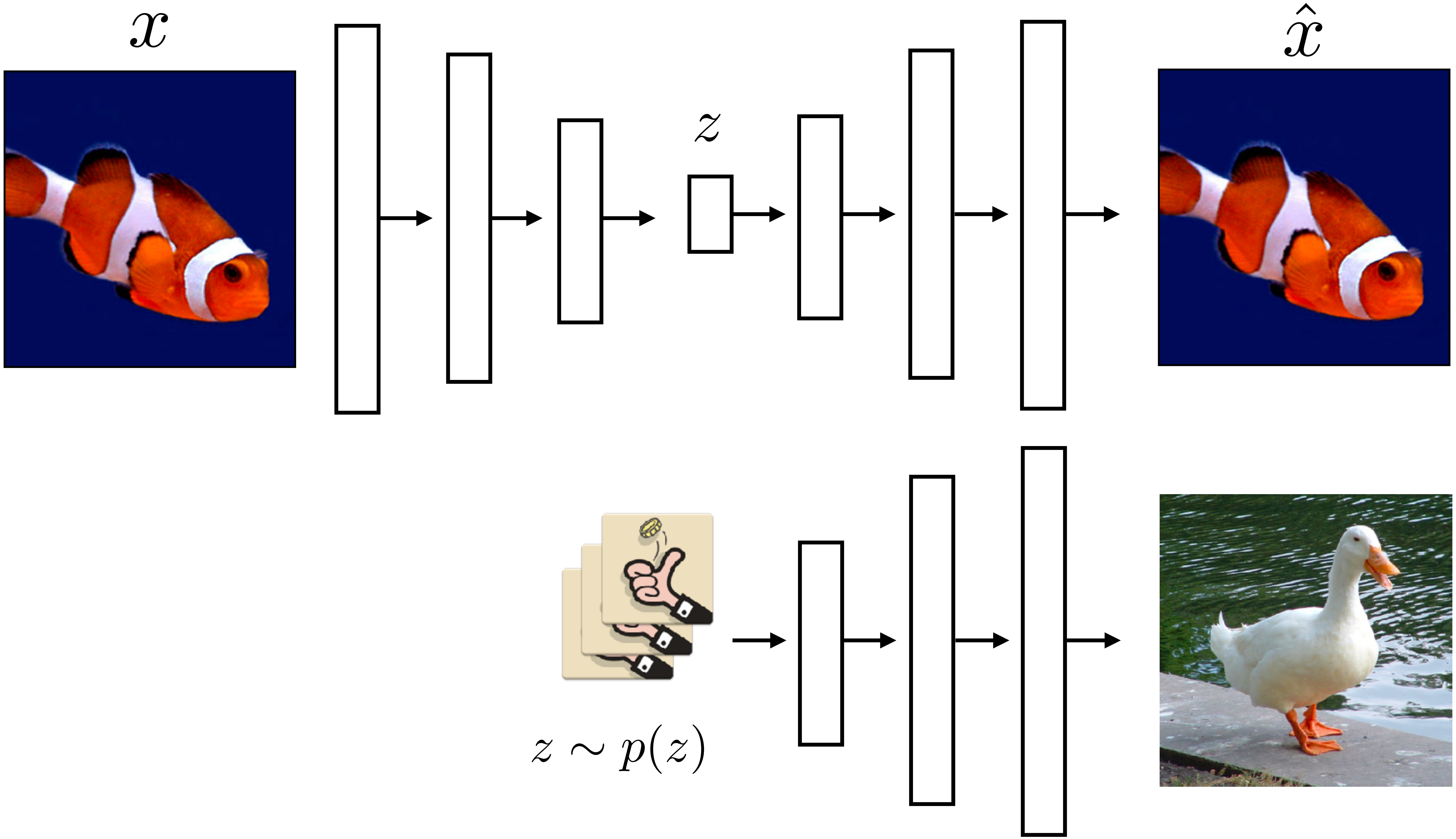
Deep generative models are distribution transformers



Deep generative models are distribution transformers

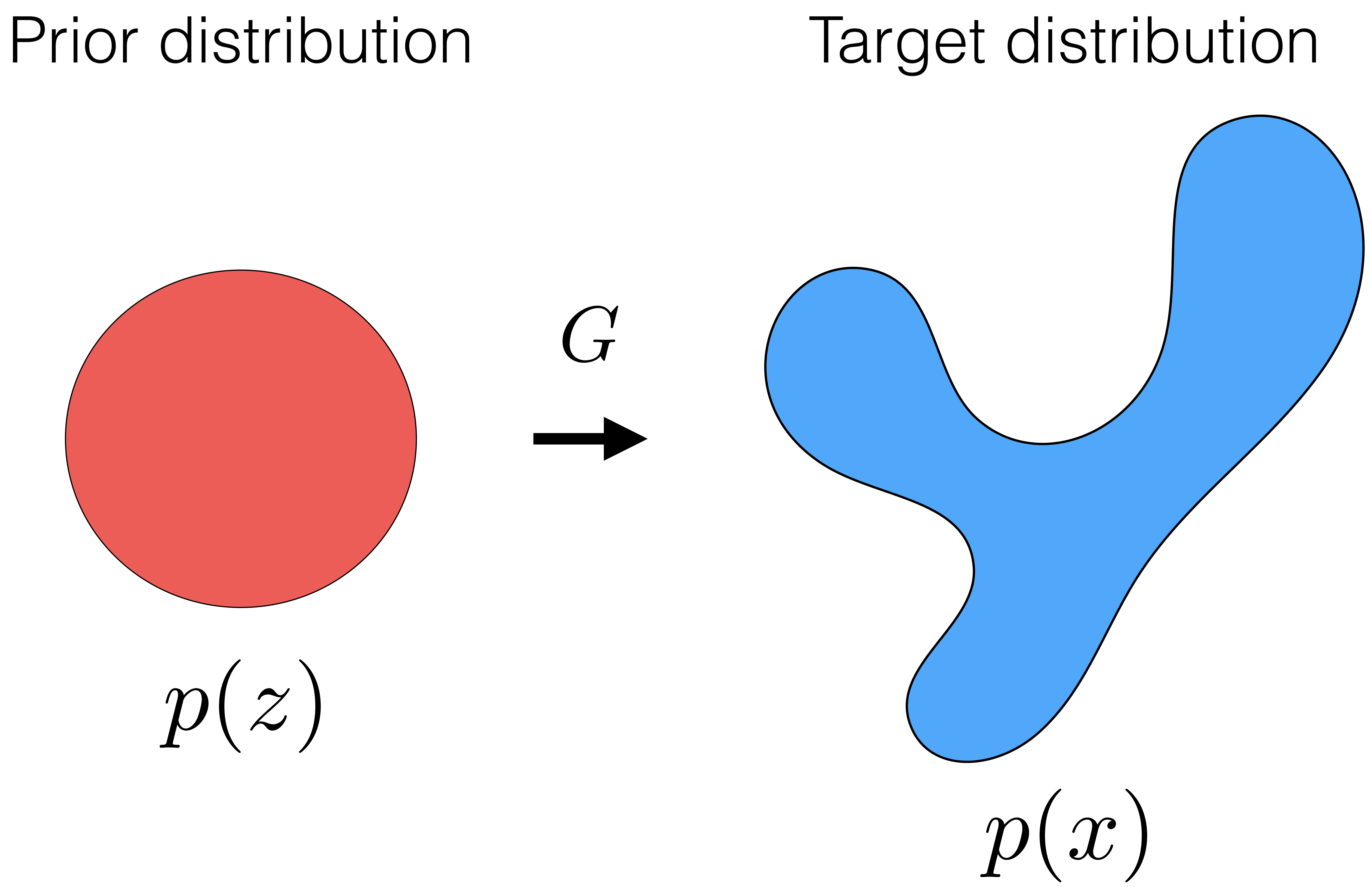


Autoencoder —> Generative model

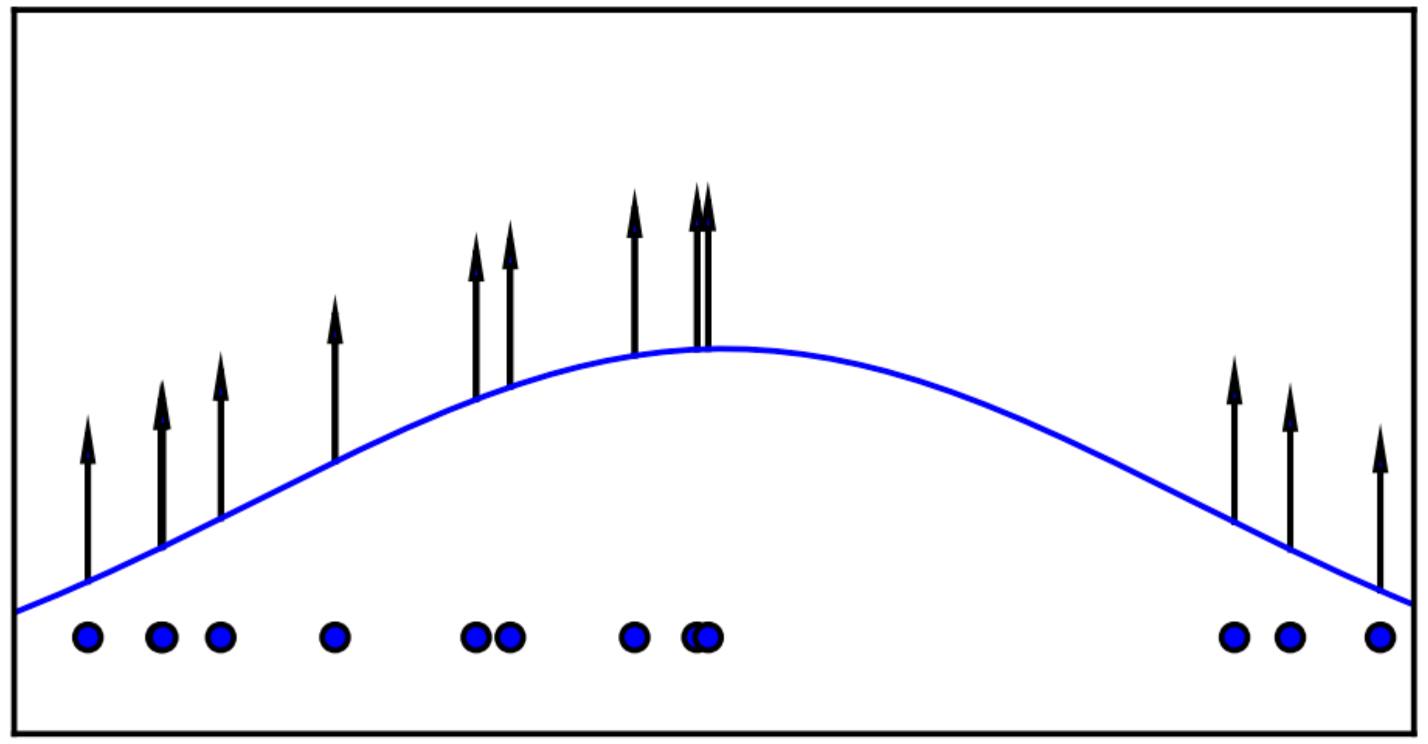


Variational Autoencoders (VAEs)

[Kingma & Welling, 2014; Rezende, Mohamed, Wierstra 2014]

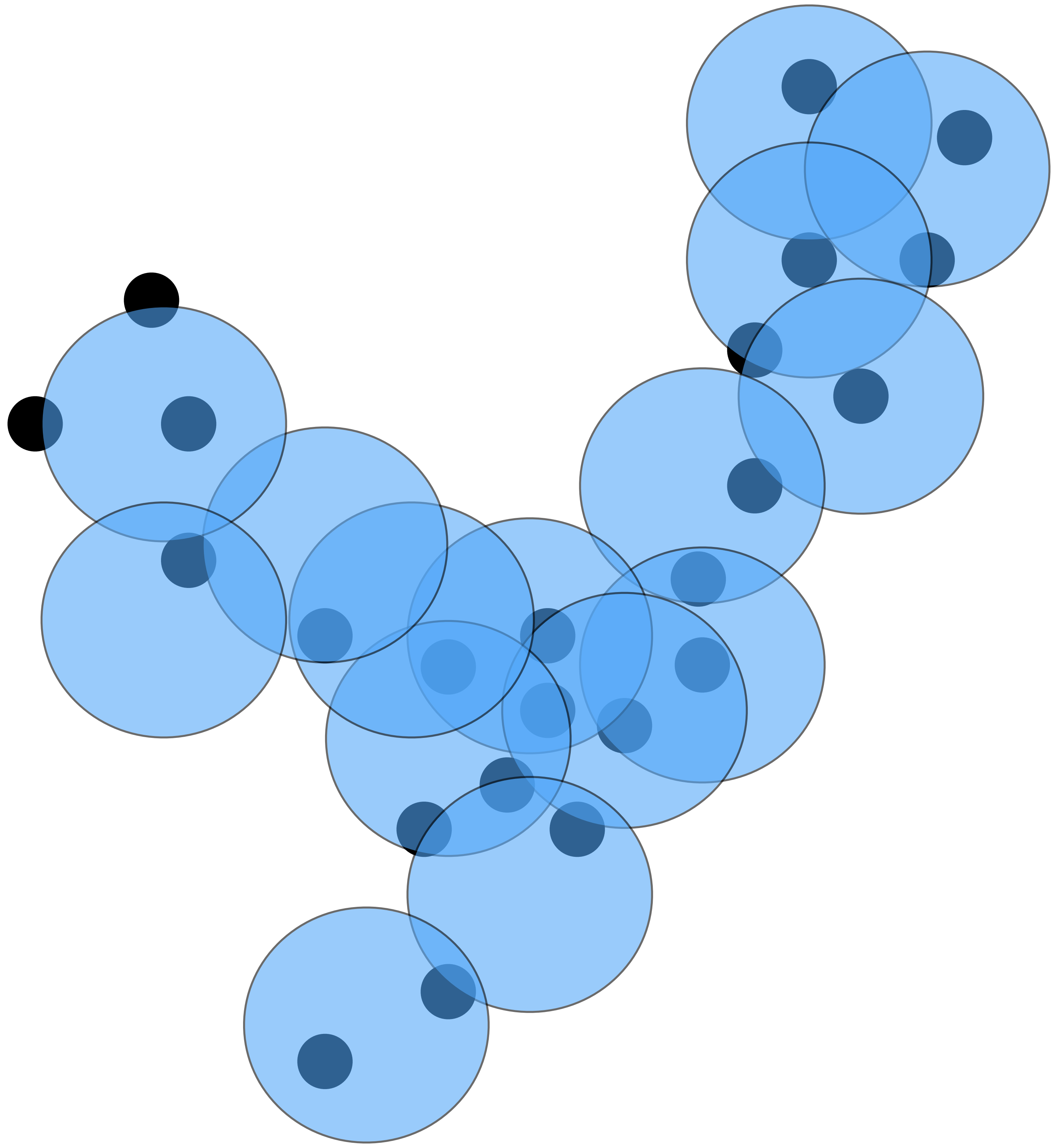


Mixture of Gaussians



$$p_\theta(x) = \sum_{i=1}^{k} w_i \mathcal{N}(x; u_i, \Sigma_i)$$

Target distribution



$$x \sim p_{\texttt{data}}(x)$$

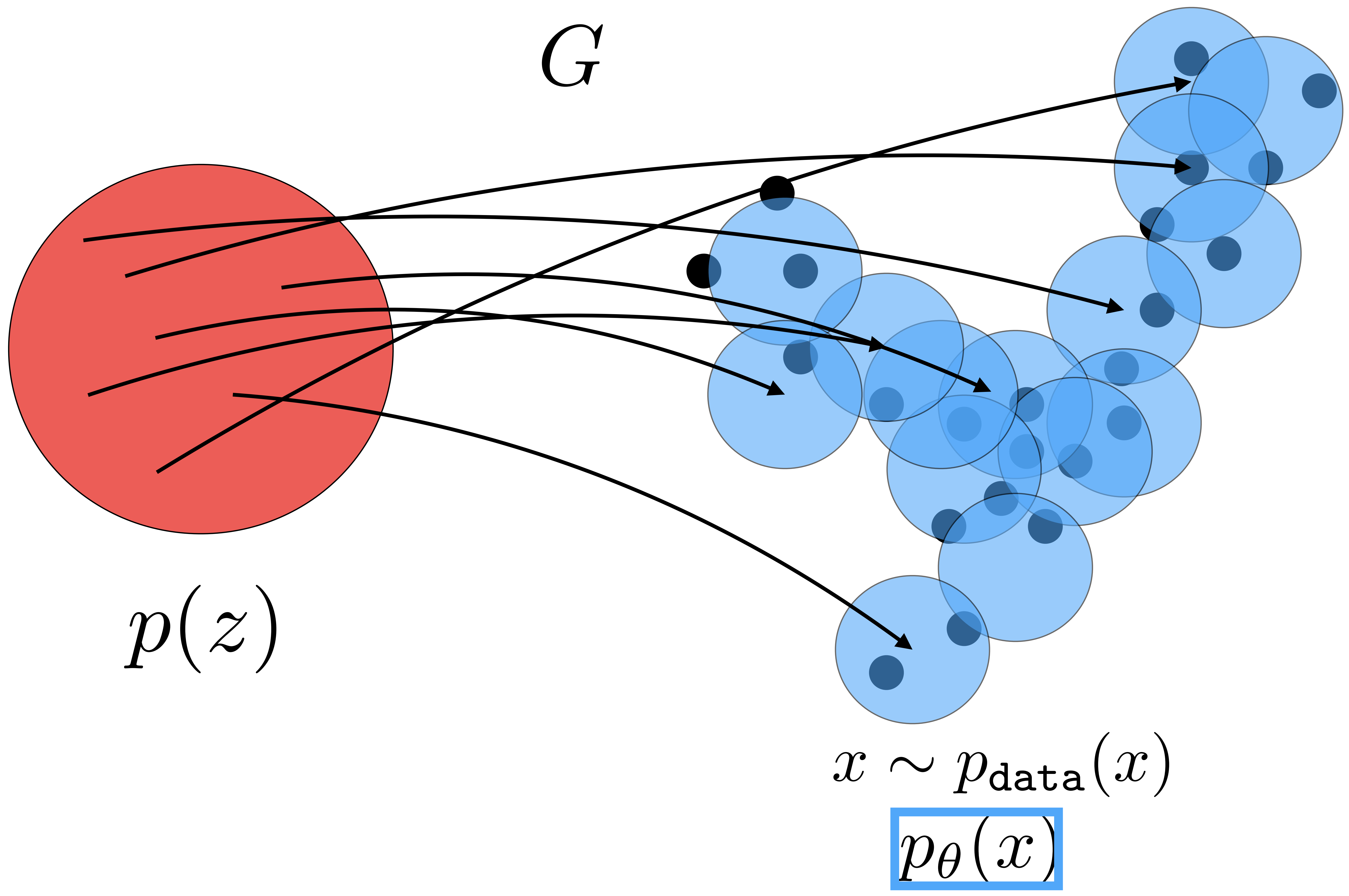
$$p_\theta(x)$$

Variational Autoencoders (VAEs)

[Kingma & Welling, 2014; Rezende, Mohamed, Wierstra 2014]

Prior distribution

Target distribution



Density model:

$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$

$$p(x|z;\theta) \sim \mathcal{N}(x; G^\mu_\theta(x), G^\sigma_\theta(x))$$

Sampling:

$$z \sim p(z) \quad \epsilon \sim \mathcal{N}(0,1)$$

$$x = G^\mu_\theta(z) + G^\sigma_\theta(z)\epsilon$$

Variational Autoencoder (VAE)

Data $\rightarrow$

Learner

Objective

$$\max_{\theta} \mathbb{E}_{x \sim p_{\texttt{data}}} [\log p_\theta(x)]$$

Hypothesis space

$$p_\theta(x) = \int p(x|z;\theta) p(z) dz$$

$$x = G_\theta^\mu(z) + G_\theta^\sigma(z)\epsilon$$

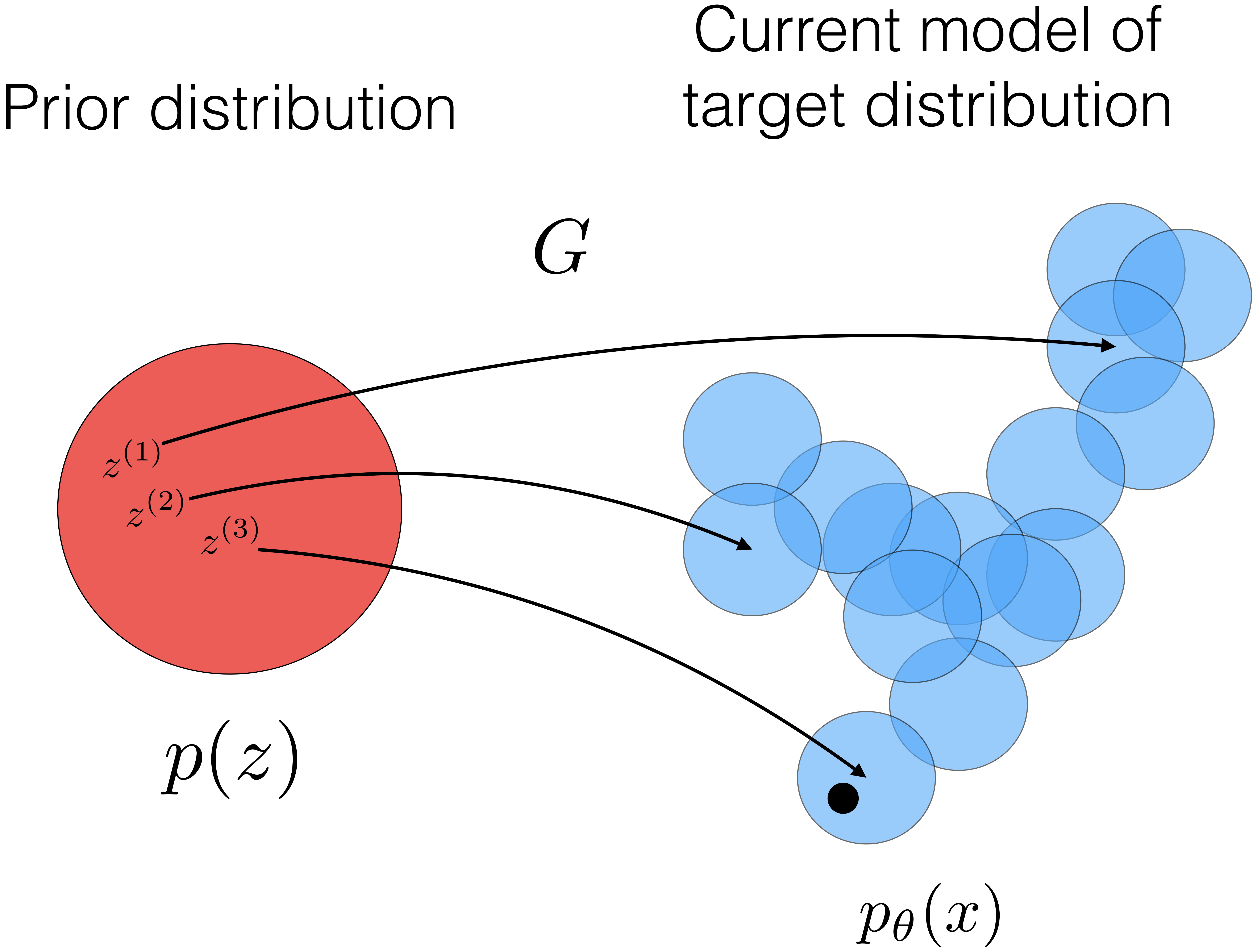
$\longrightarrow$

Density

$$p_\theta : \mathcal{X} \rightarrow [0, 1]$$

Sampler

$$G_\theta : \mathcal{Z} \rightarrow \mathcal{X}$$



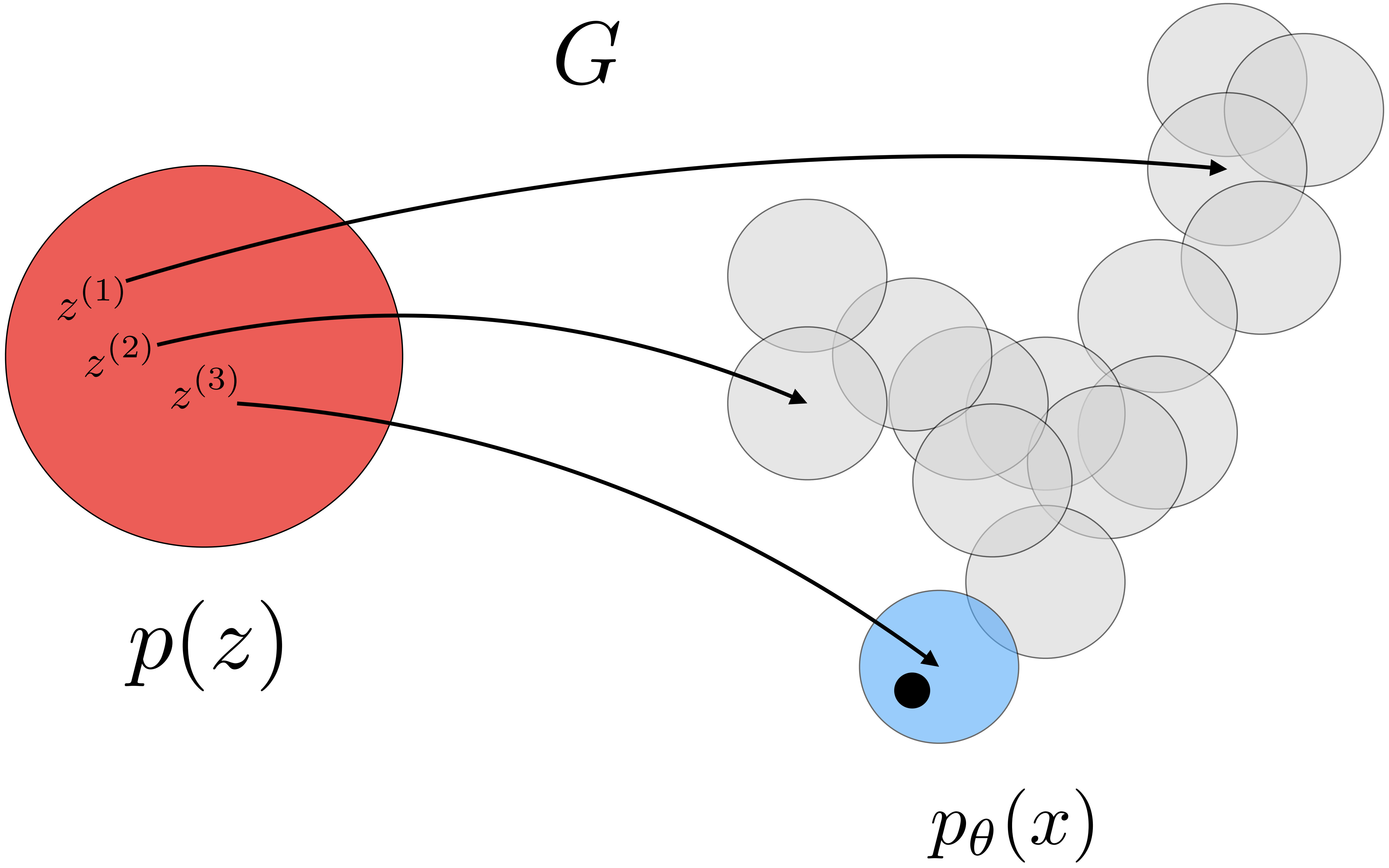
In order to optimize our model, we need to measure the likelihood it assigns to each datapoint x

$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$

$$= p(x|z^{(1)})p(z^{(1)})dz +$$
$$p(x|z^{(2)})p(z^{(2)})dz +$$
$$p(x|z^{(3)})p(z^{(3)})dz + \ldots$$

Prior distribution

Current model of target distribution



In order to optimize our model, we need to measure the likelihood it assigns to each datapoint x

$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$

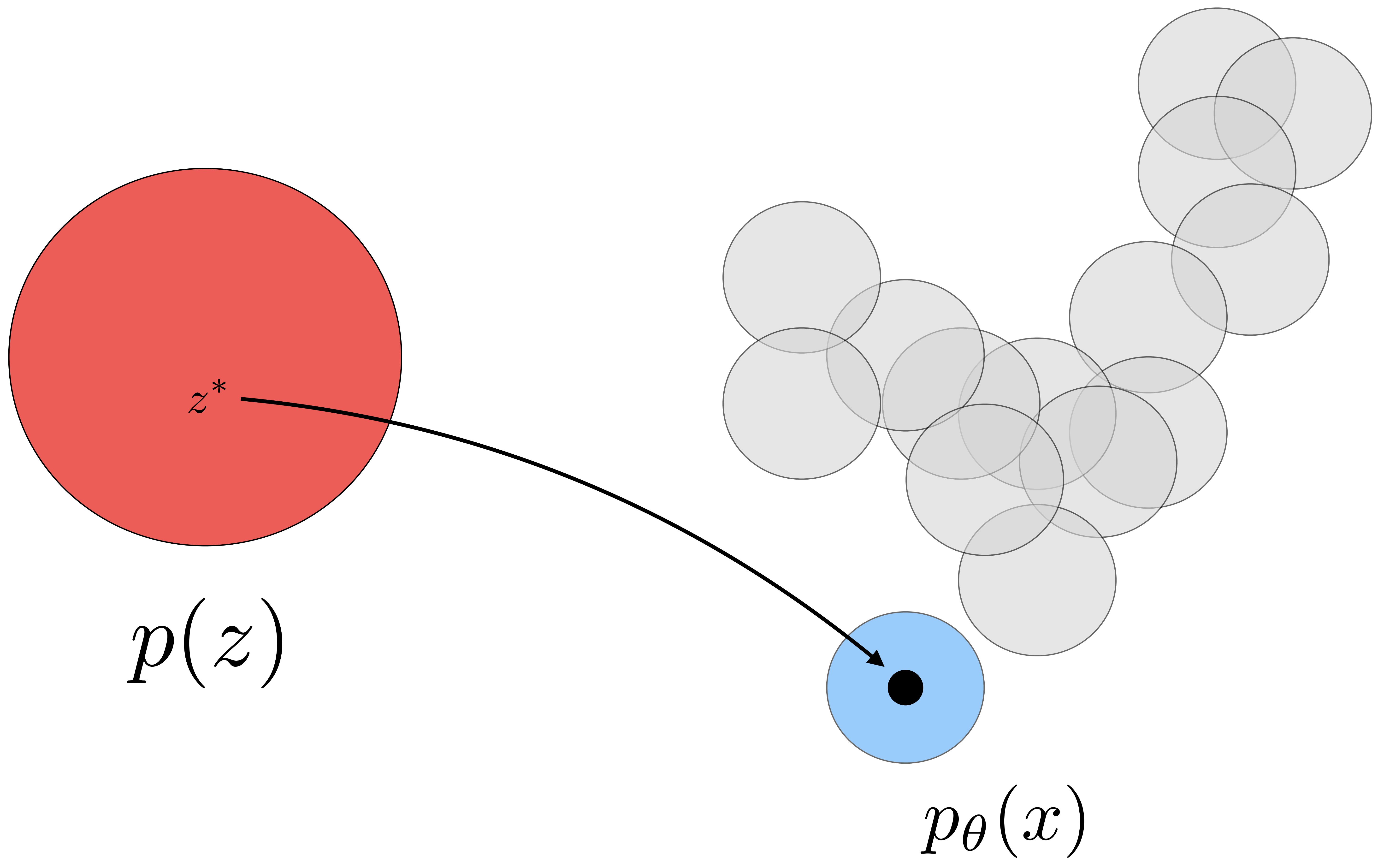
$$= \sim 0+$$

$$\sim 0+$$

$$p(x|z^{(3)})p(z^{(3)})dz + ...$$

Prior distribution

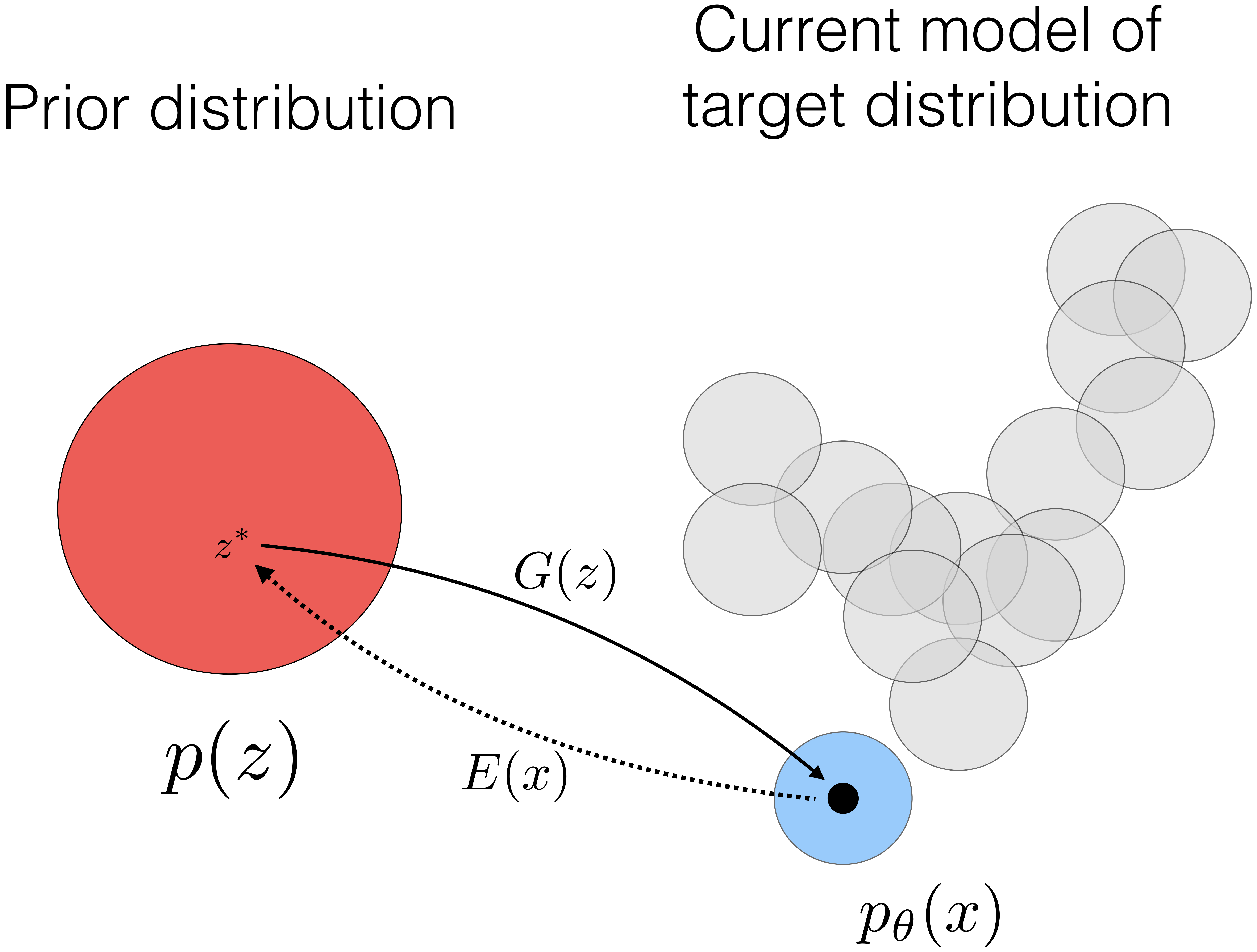
Current model of target distribution



If only we knew z*, we wouldn't need the integral…

$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$

$$\approx p(x|z^*;\theta)p(z^*)$$



If only we knew z*, we wouldn't need the integral…

$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$

$$\approx p(x|z^*;\theta)p(z^*)$$

So, we simply try to predict z* for the given x!

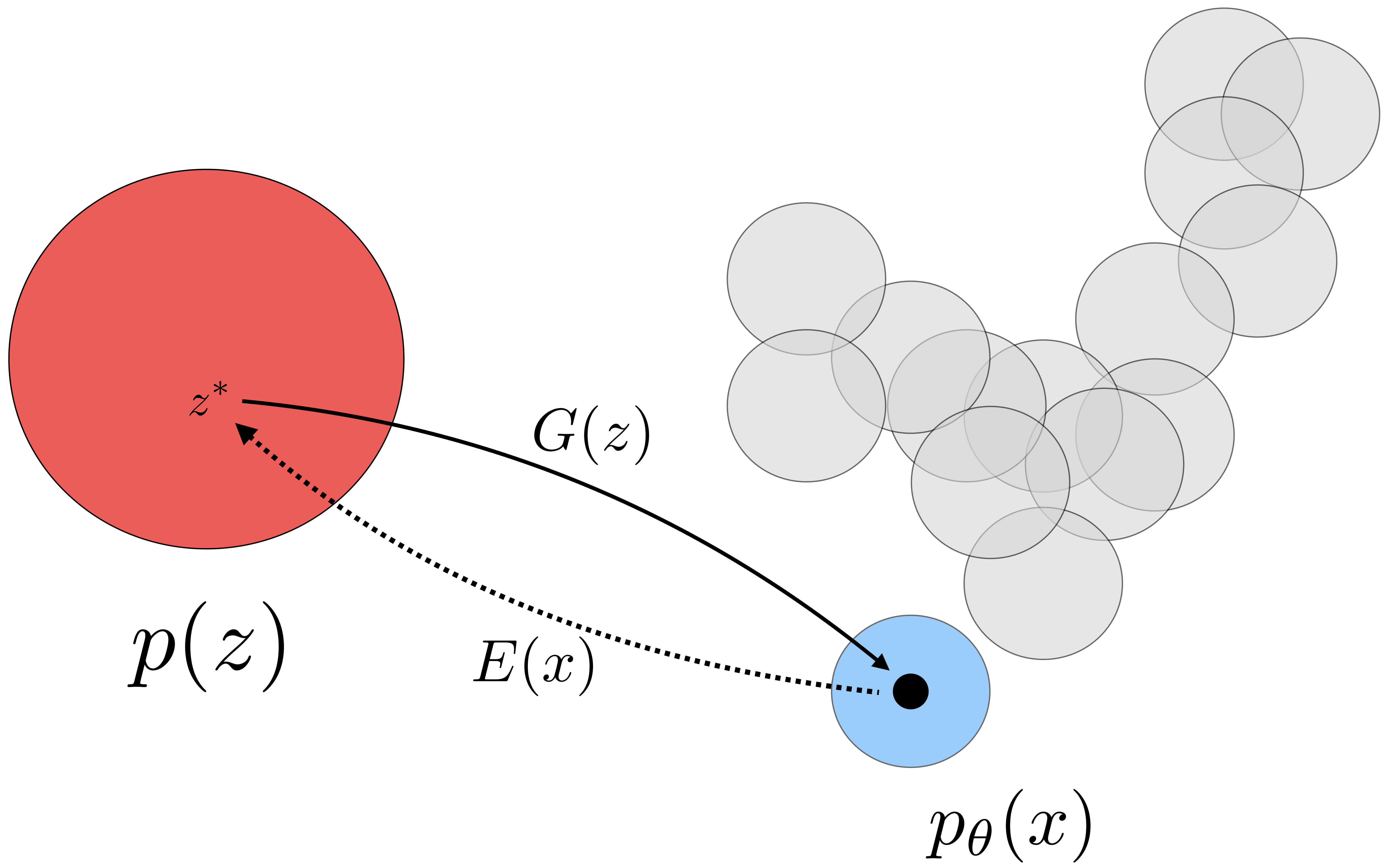
$$z^* = E(x)$$

$$\arg\max_E p(x|E(x);\theta)p(E(x))$$

Technical note: for the continuous math to actually work out, z ~ E(x) needs to be a distribution (typically set to Gaussian), but here we (incorrectly) treat it as deterministic for simplicity.*

Prior distribution

Current model of target distribution



If only we knew z*, we wouldn't need the integral…

$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$

$$\approx p(x|z^*;\theta)p(z^*)$$

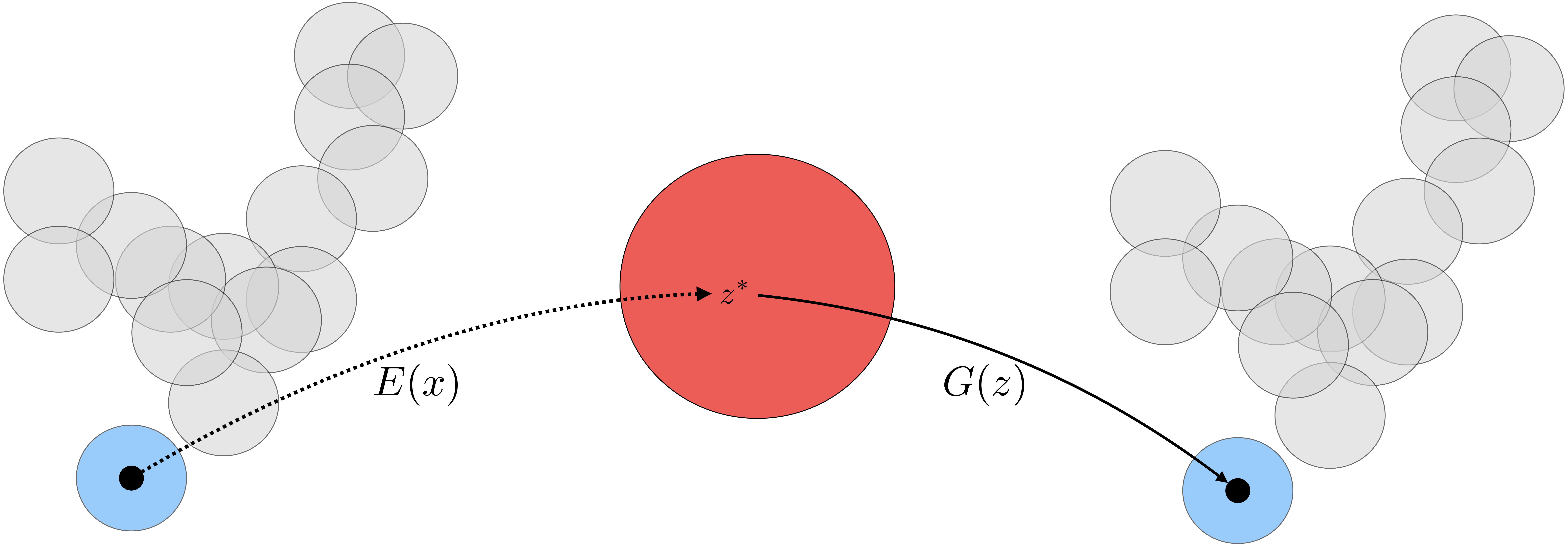
So, we simply try to predict z* for the given x!

$$z^* = E(x)$$

(assuming unit Gaussian prior, isotropic Gaussian likelihood model) →

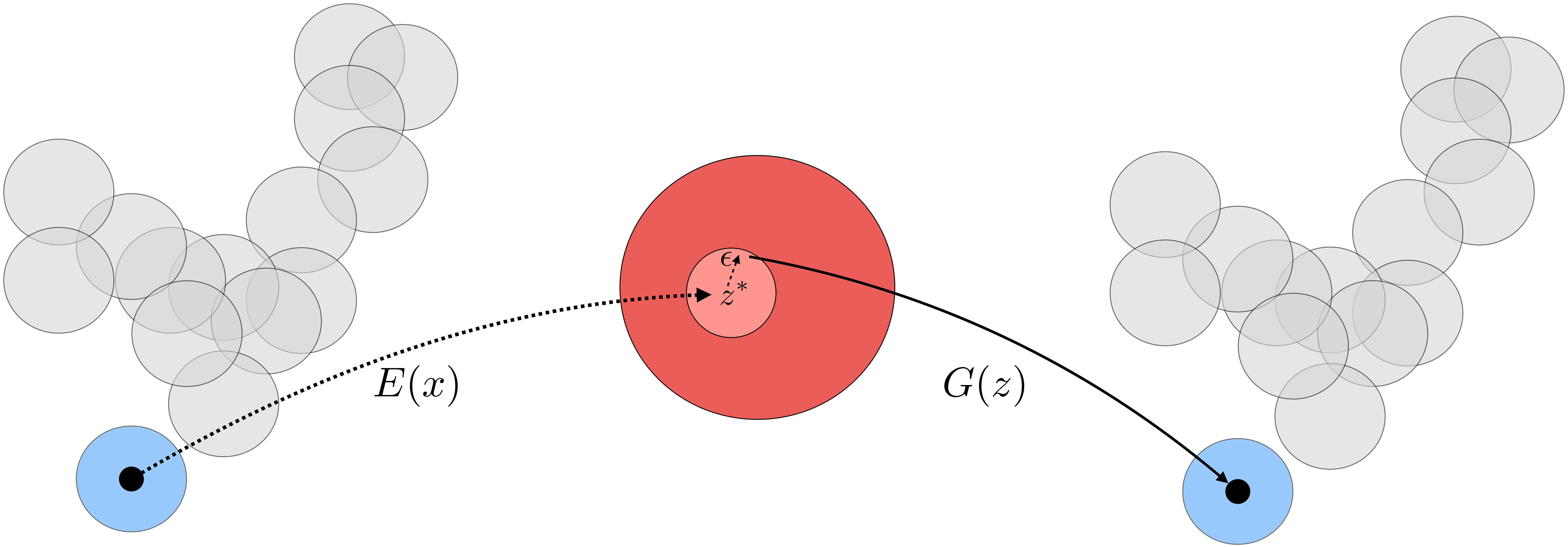
$$\arg\min_E \|G(E(x)) - x\|_2^2 + ||E(x)||_2^2$$

Autoencoder!



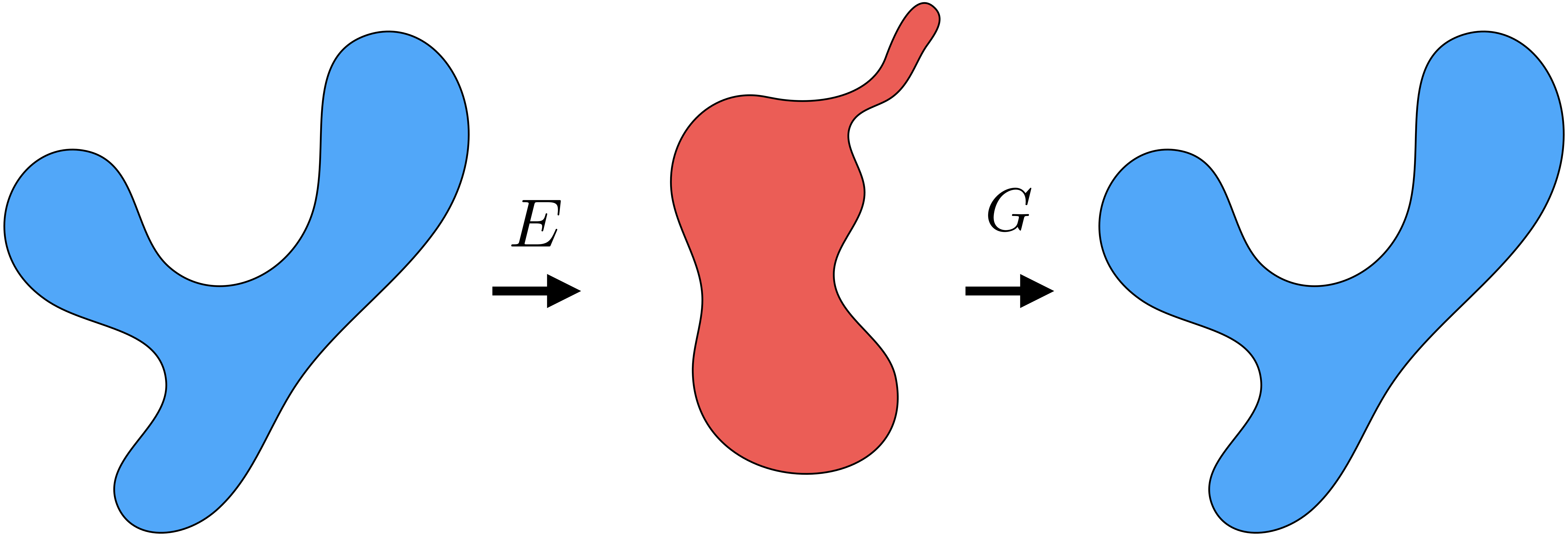
$$\arg\min_{G,E} \|G(E(x)) - x\|_2^2 + \|E(x)\|_2^2$$

Autoencoder!



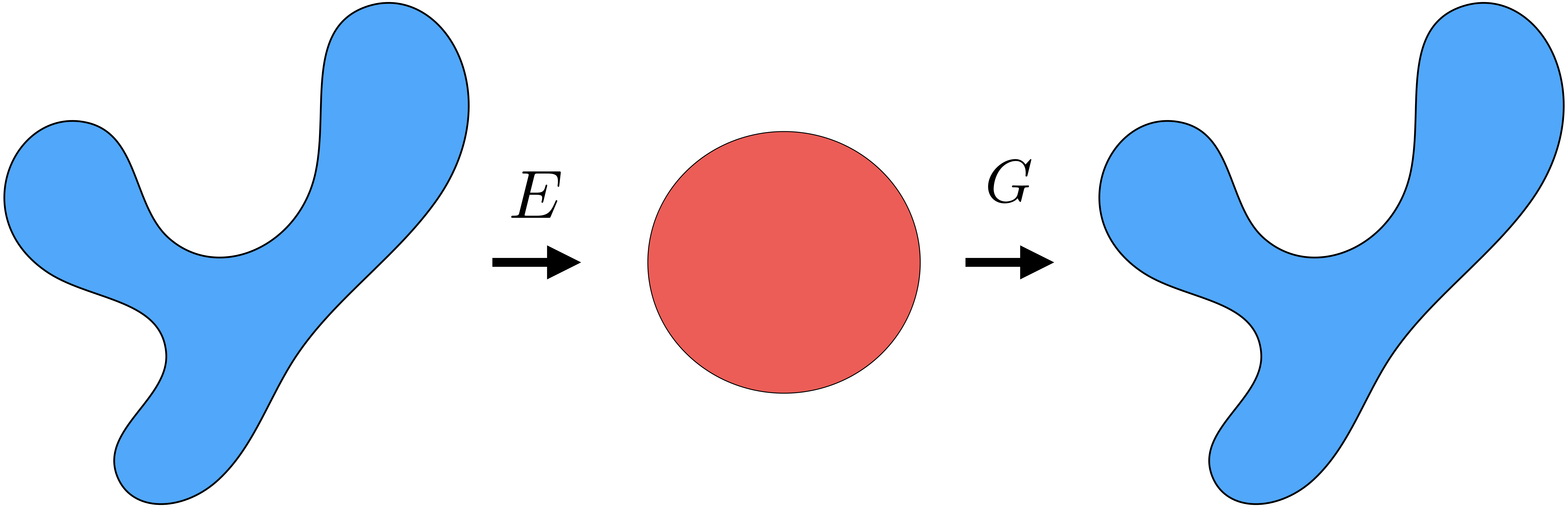
$$\arg\min_{G,E} \mathbb{E}_{x,\epsilon}[\|G(E(x+\epsilon)) - x\|_2^2 + \|E(x+\epsilon)\|_2^2]$$

Classical Autoencoder



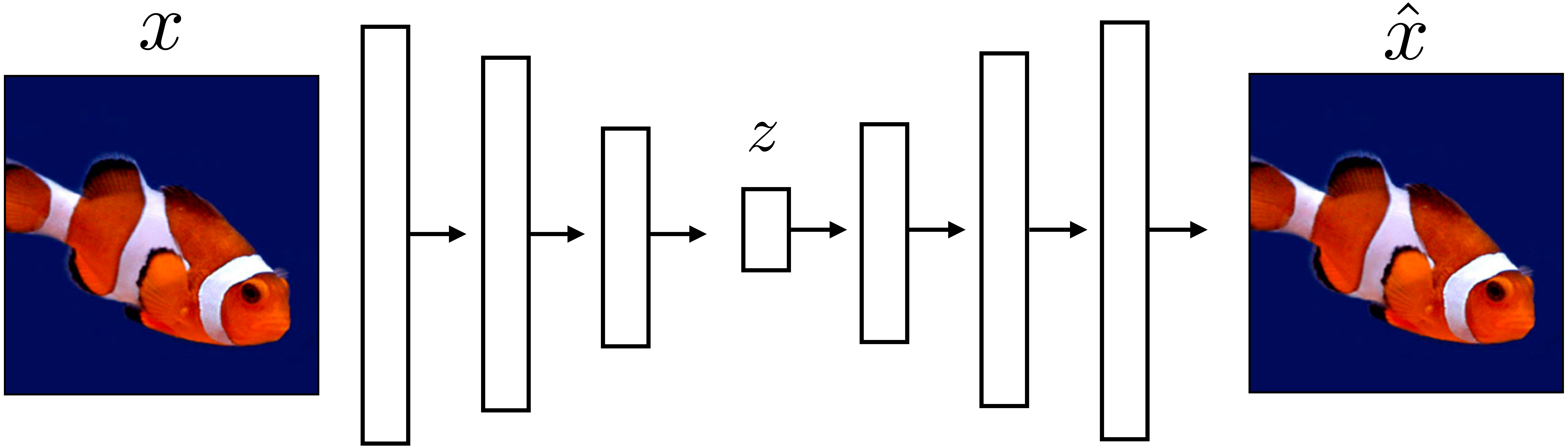
$$\arg\min_{G,E} \mathbb{E}_x[\|G(E(x)) - x\|_2^2]$$

Variational Autoencoder

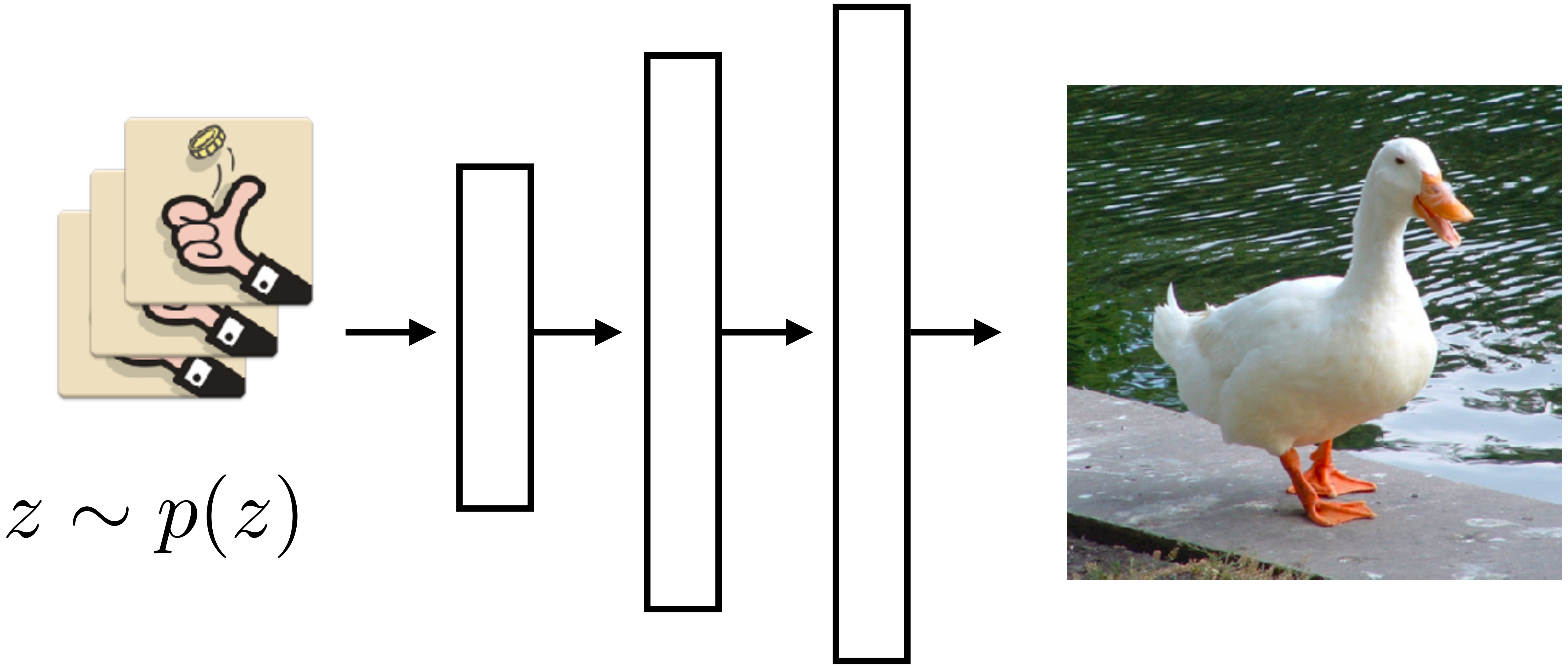


$$\arg\min_{G,E} \mathbb{E}_{x,\epsilon}[\|G(E(x+\epsilon)) - x\|_2^2 + \|E(x+\epsilon)\|_2^2]$$

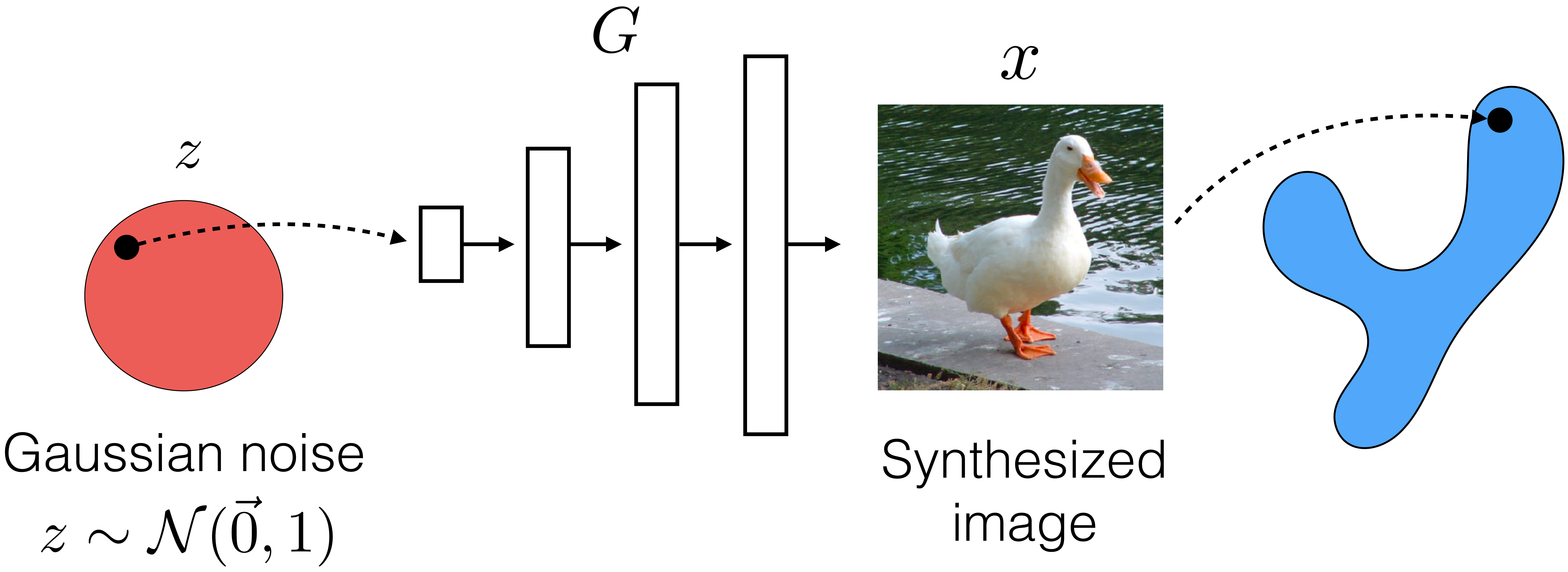
Variational Autoencoder

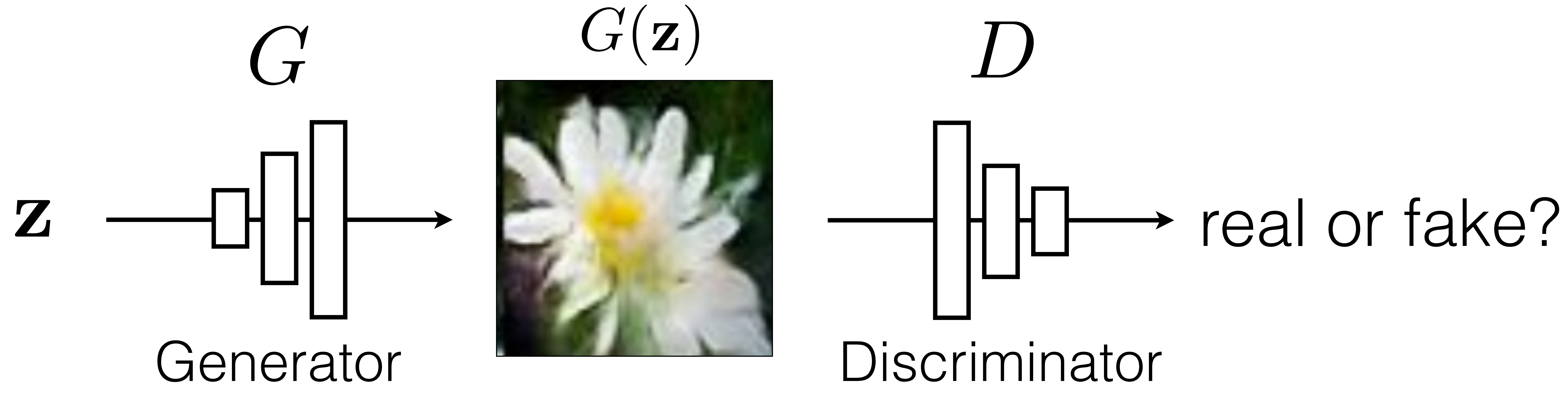


All of that math was basically just to make z have a Gaussian distribution, so that we sample random images by inputing random Gaussian noise.



Generative Adversarial Networks (GANs)

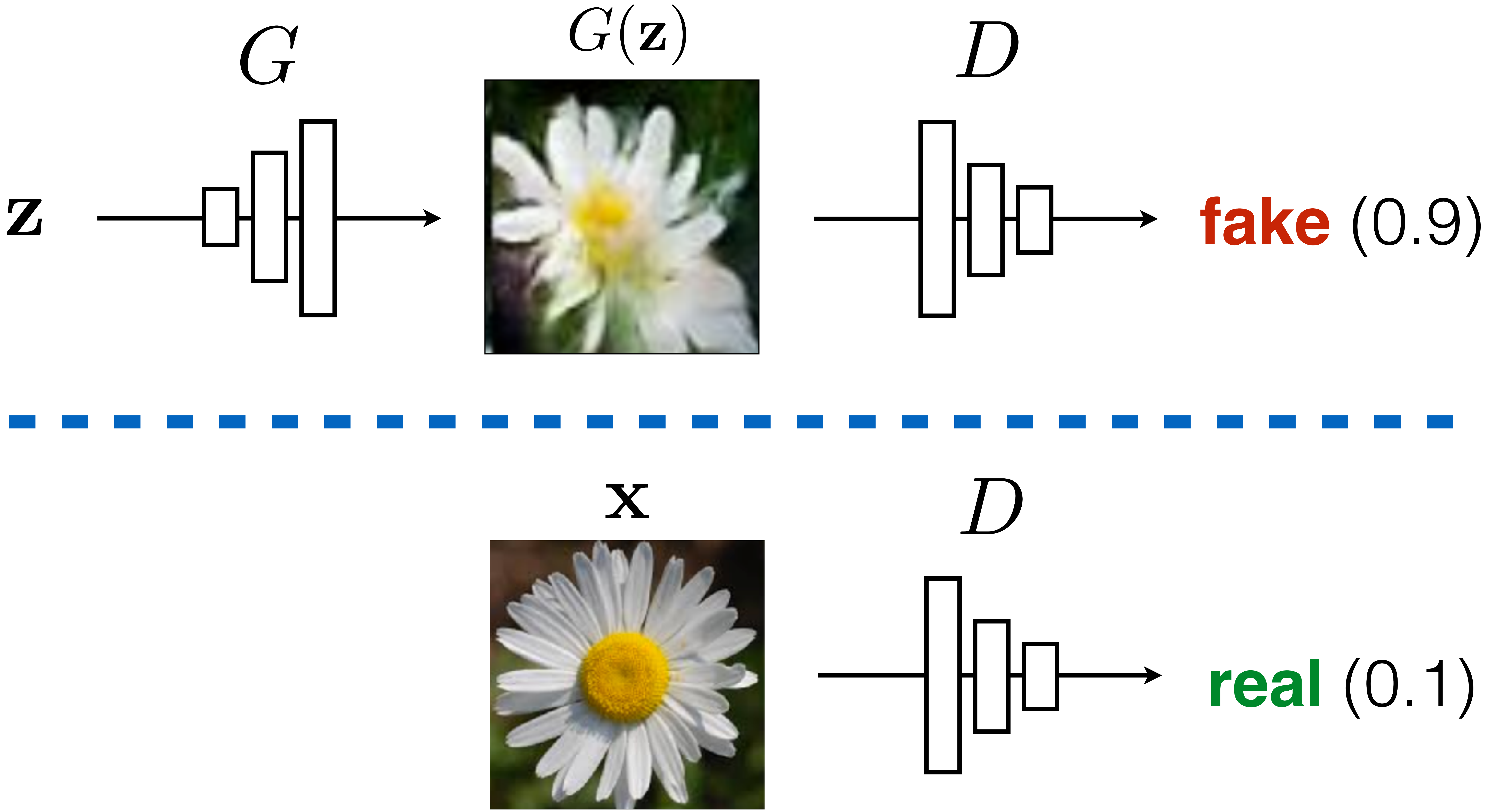




G tries to synthesize fake images that fool **D**

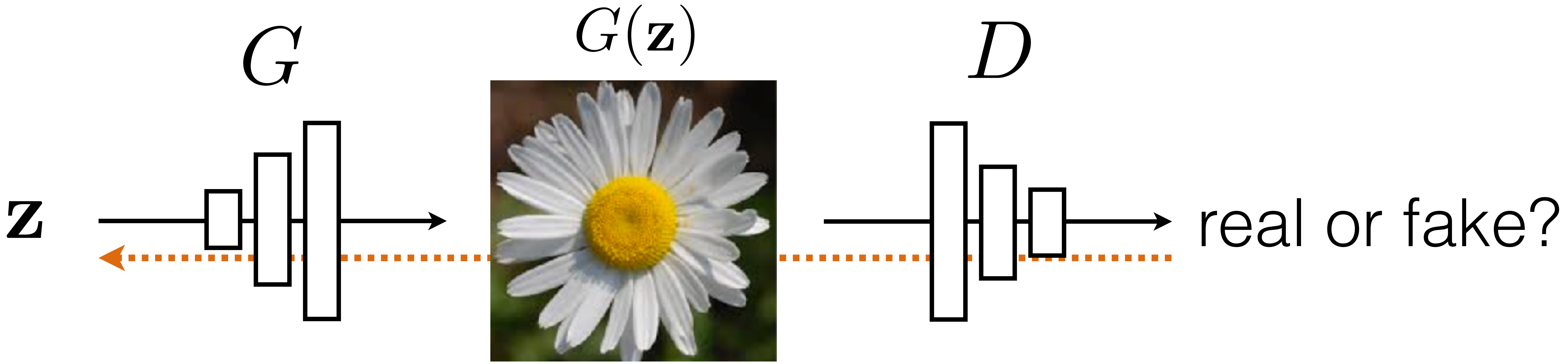
D tries to identify the fakes

[Goodfellow et al., 2014]



$$\arg\max_{D} \mathbb{E}_{\mathbf{z},\mathbf{x}}[\ \log D(G(\mathbf{z})) \ + \ \log(1 - D(\mathbf{x}))\ ]$$

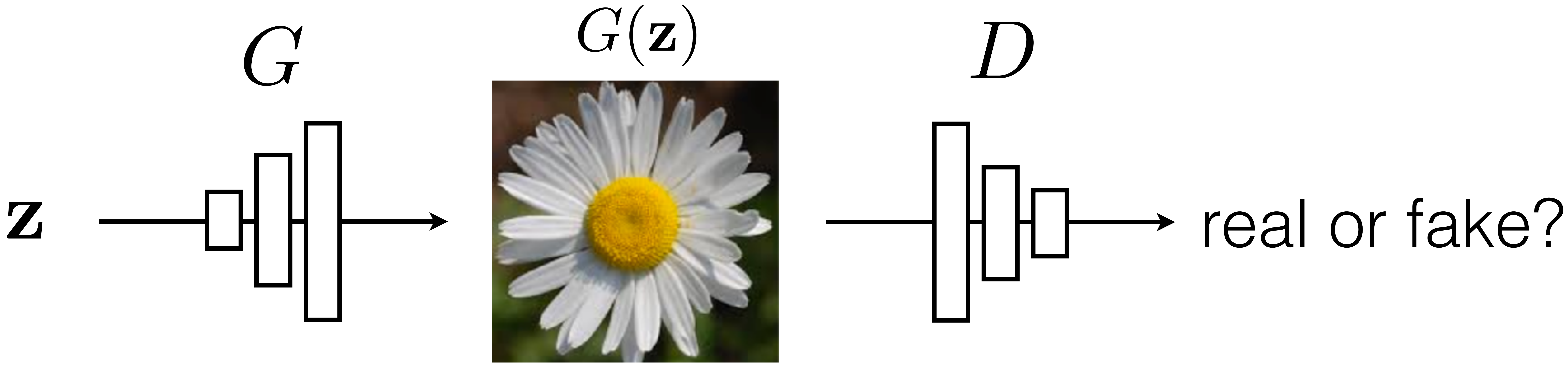
[Goodfellow et al., 2014]



G tries to synthesize fake images that ***fool*** **D**:

$$\arg\min_{G} \; \mathbb{E}_{\mathbf{z},\mathbf{x}}[\; \log D(G(\mathbf{z})) \; + \; \log(1 - D(\mathbf{x})) \;]$$

[Goodfellow et al., 2014]

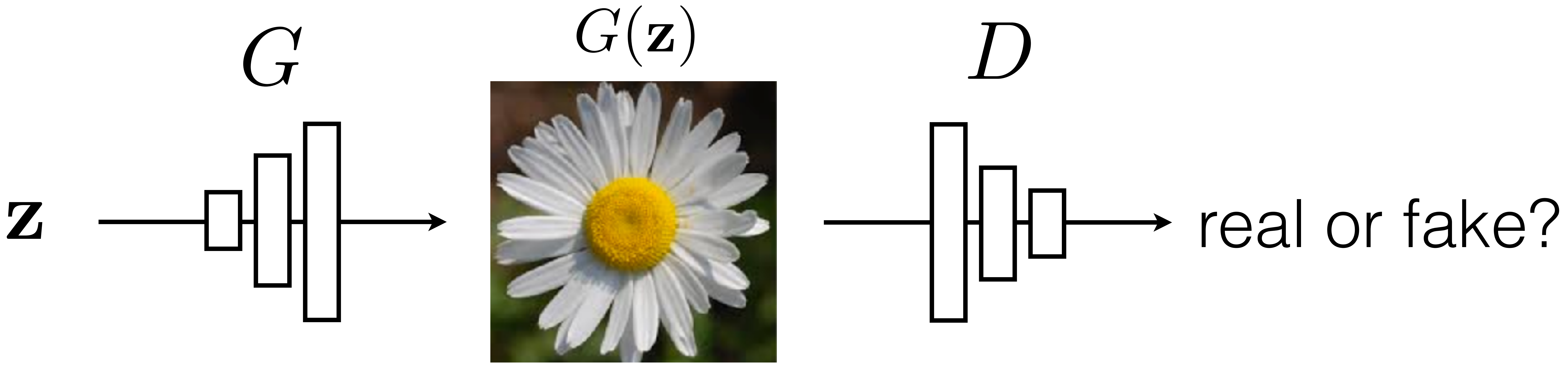


G tries to synthesize fake images that ***fool*** the ***best*** **D**:

$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{z},\mathbf{x}}\left[\ \log D(G(\mathbf{z}))\ +\ \log\left(1 - D(\mathbf{x})\right)\ \right]$$

[Goodfellow et al., 2014]

Training



G tries to synthesize fake images that fool **D**

D tries to identify the fakes

- Training: iterate between training D and G with backprop.
- Global optimum when G reproduces data distribution.

[Goodfellow et al., 2014]

Samples from BigGAN

[Brock et al. 2018]

Generative Adversarial Network

Data $\rightarrow$

Learner

Objective

$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{z},\mathbf{x}}[\; \log D(G(\mathbf{z})) \;+\; \log(1 - D(\mathbf{x})) \;]$$

Hypothesis space

Deep nets G and D

Optimizer

Alternating SGD on G and D

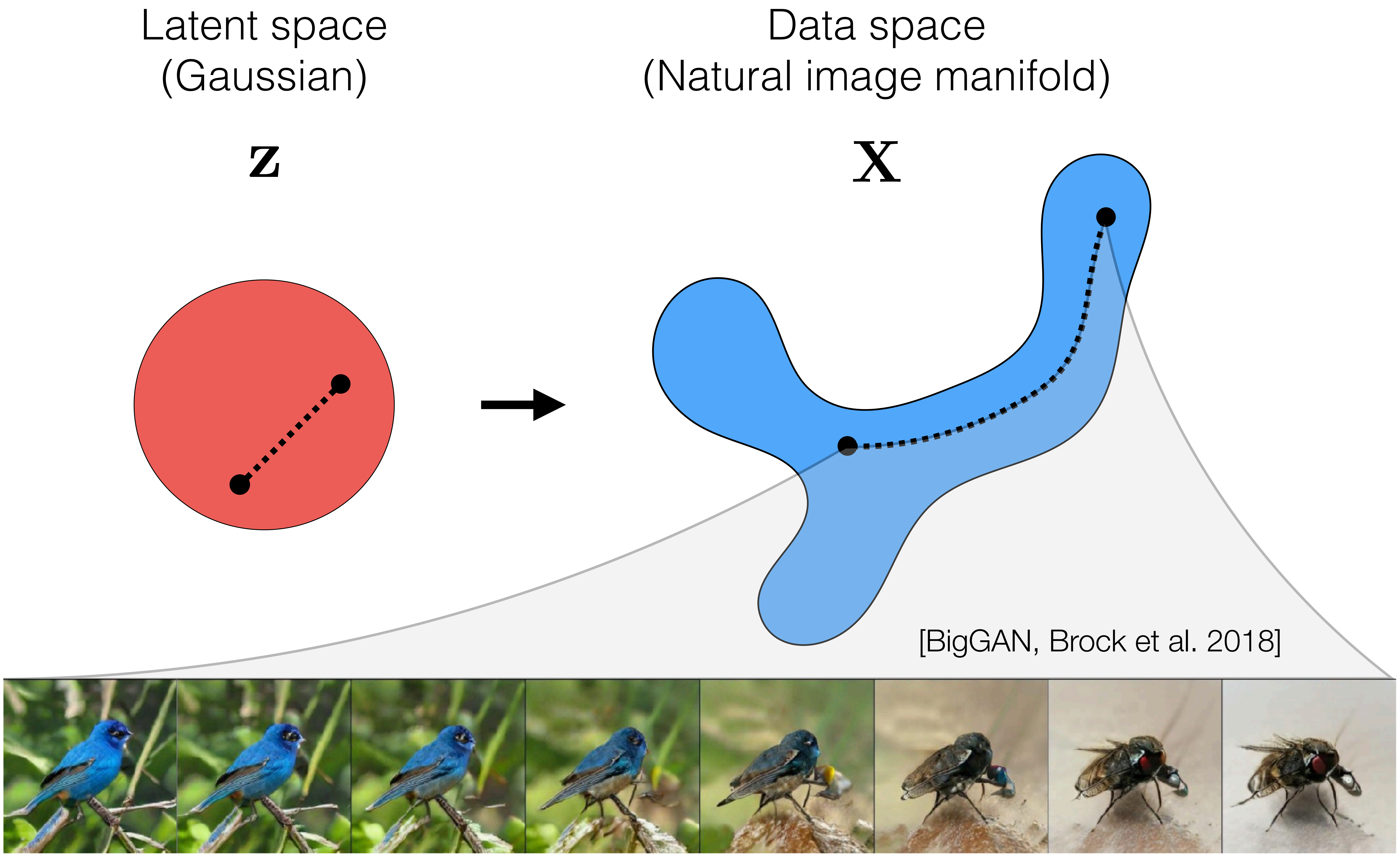
$\rightarrow$

Critic

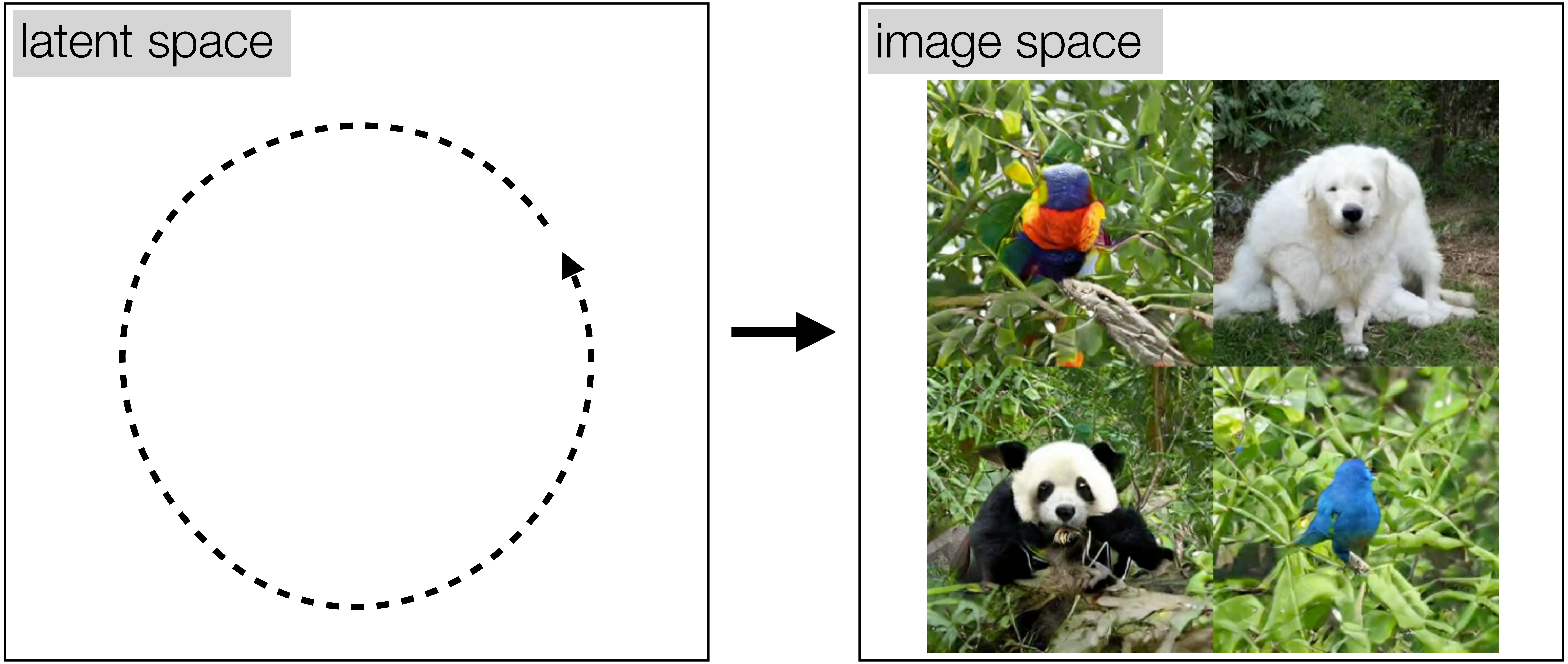
$D : \mathcal{X} \rightarrow [0,1]$

Sampler

$G : \mathcal{Z} \rightarrow \mathcal{X}$


Latent space
(Gaussian)
z
Data space
(Natural image manifold)
x
[BigGAN, Brock et al. 2018]

Generative models organize the manifold of natural images



VAEs

Pros: Cheap to sample, good coverage

Cons: Blurry samples (in practice)

GANs

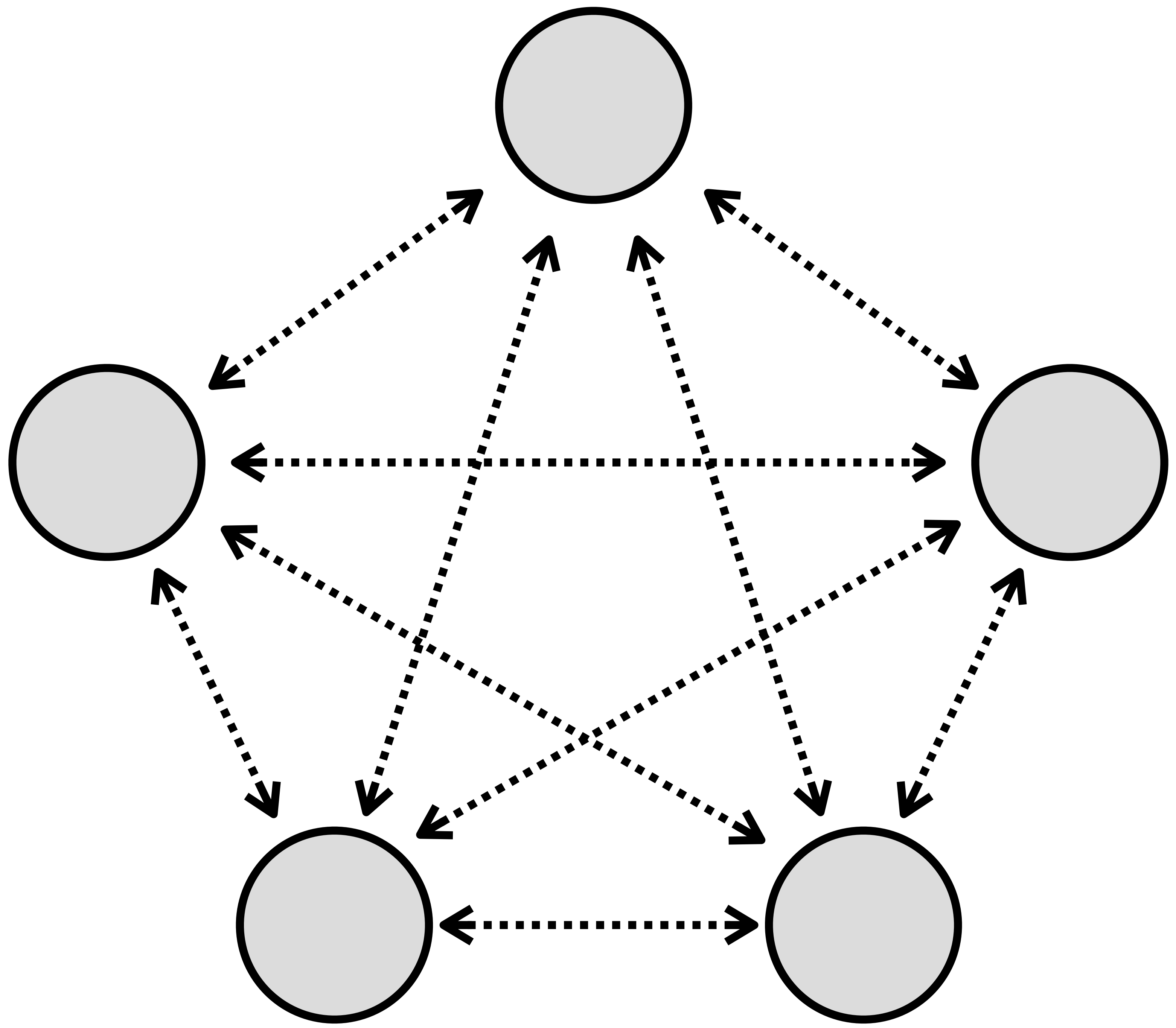
Pros: Cheap to sample, fast to train, require little data

Cons: No likelihoods, bad coverage (mode collapse), finicky to train (minimax)

Other deep generative models:
Autoregressive models, Normalizing flows, **Energy-based models**

[adapted from slide by David Duvenaud]

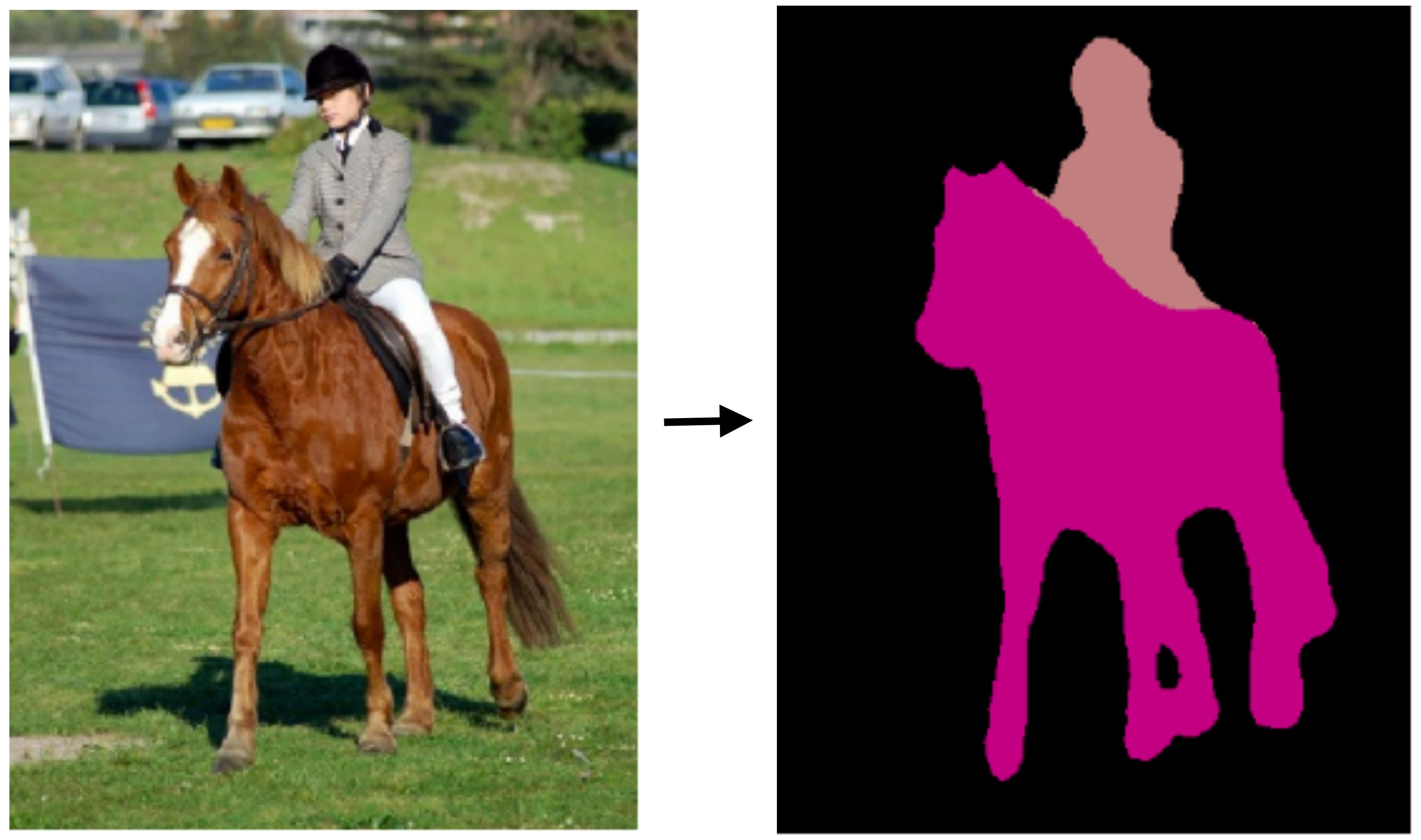
1. Image synthesis
2. **Structured prediction**
3. Domain mapping



Strutured Prediction

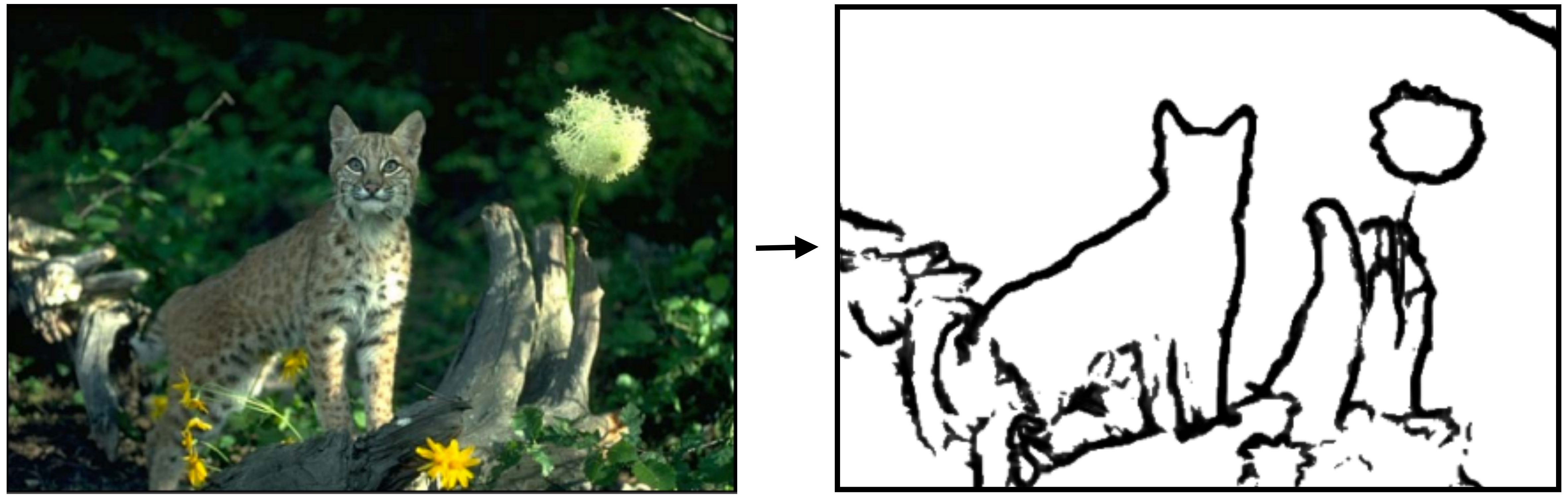
Data prediction problems (“structured prediction”)

Semantic segmentation



[Long et al. 2015, …]

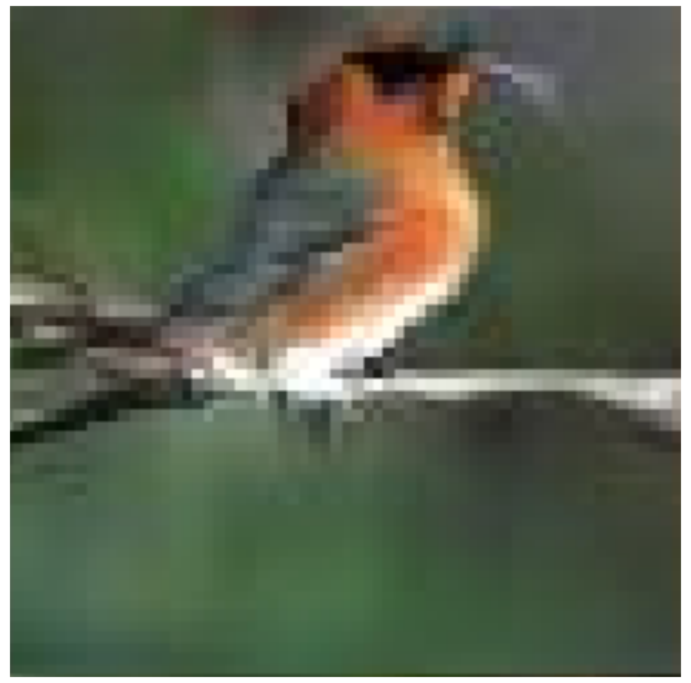
Edge detection



[Xie et al. 2015, …]

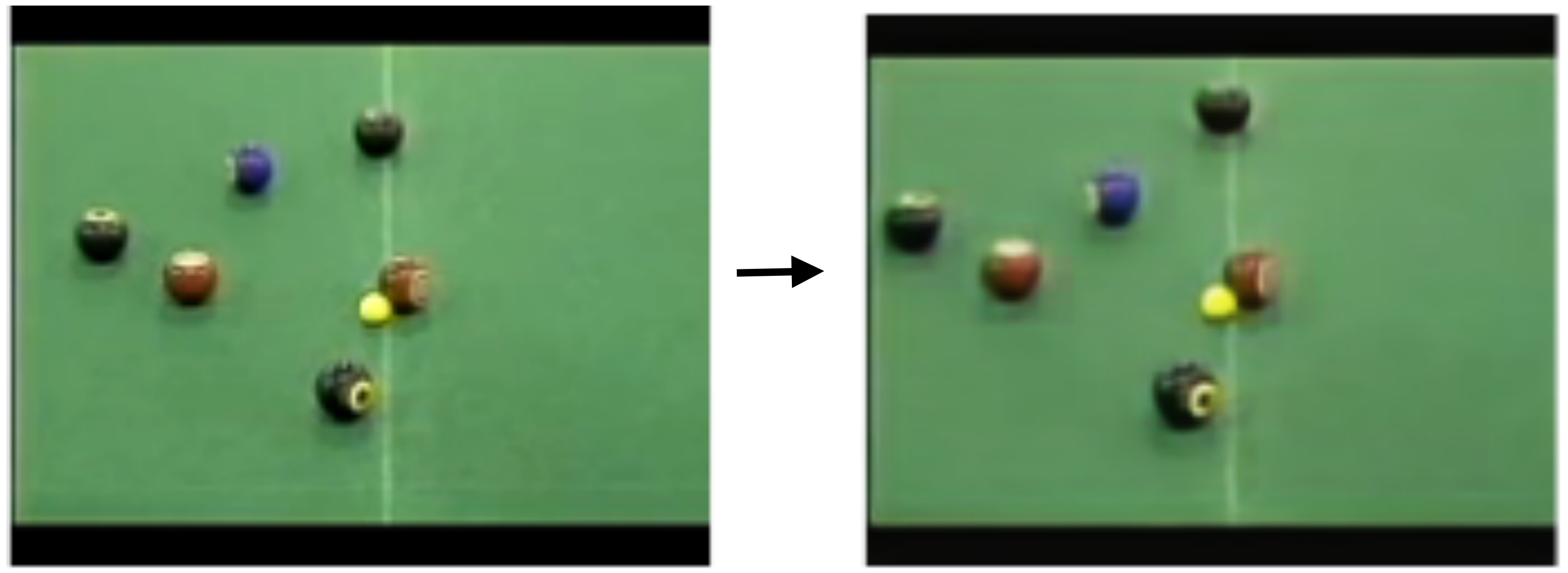
Text-to-photo

“this small bird has a pink breast and crown…” →



[Reed et al. 2014, …]

Future frame prediction



[Mathieu et al. 2016, …]

Structured prediction

X is high-dimensional

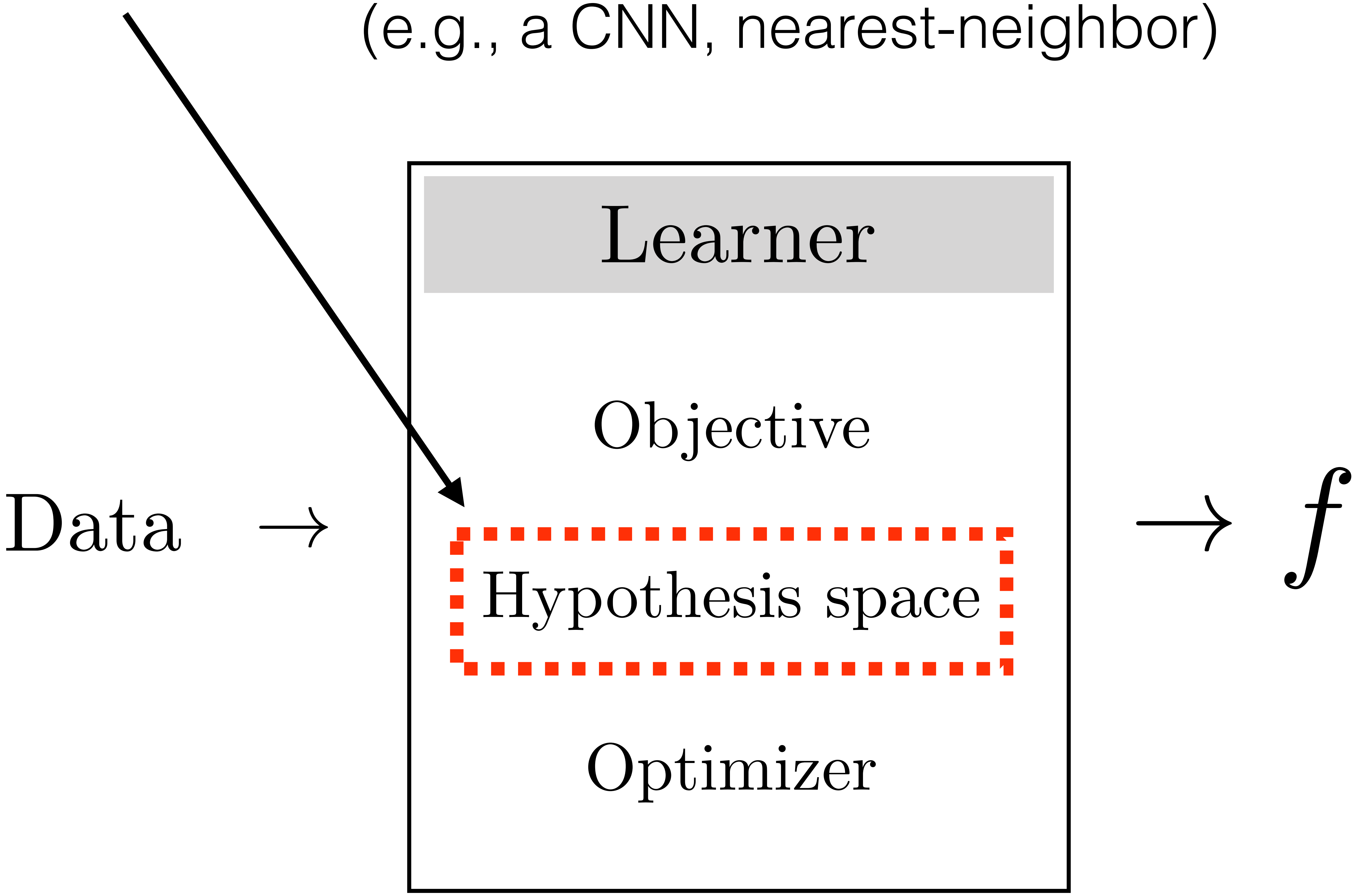
Model *joint* distribution of high-dimensional data $P(\mathbf{X}|\mathbf{Y}=\mathbf{y})$

In vision this is usually what we are interested in

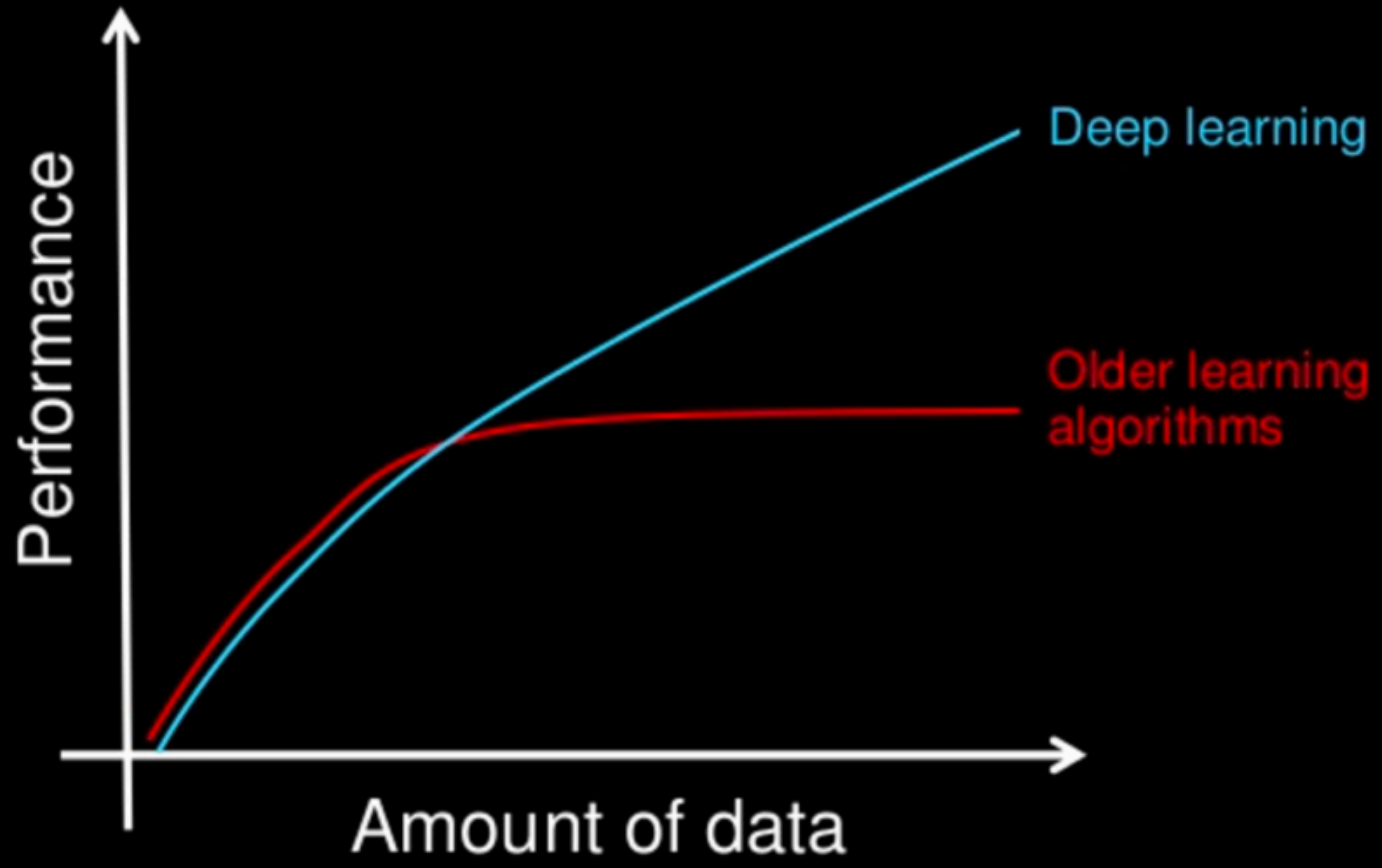
Unstructured: $\prod_i p(X_i|\mathbf{Y}=\mathbf{y})$

Deep learning in 2012

Use a **hypothesis space** that can model complex structure
(e.g., a CNN, nearest-neighbor)

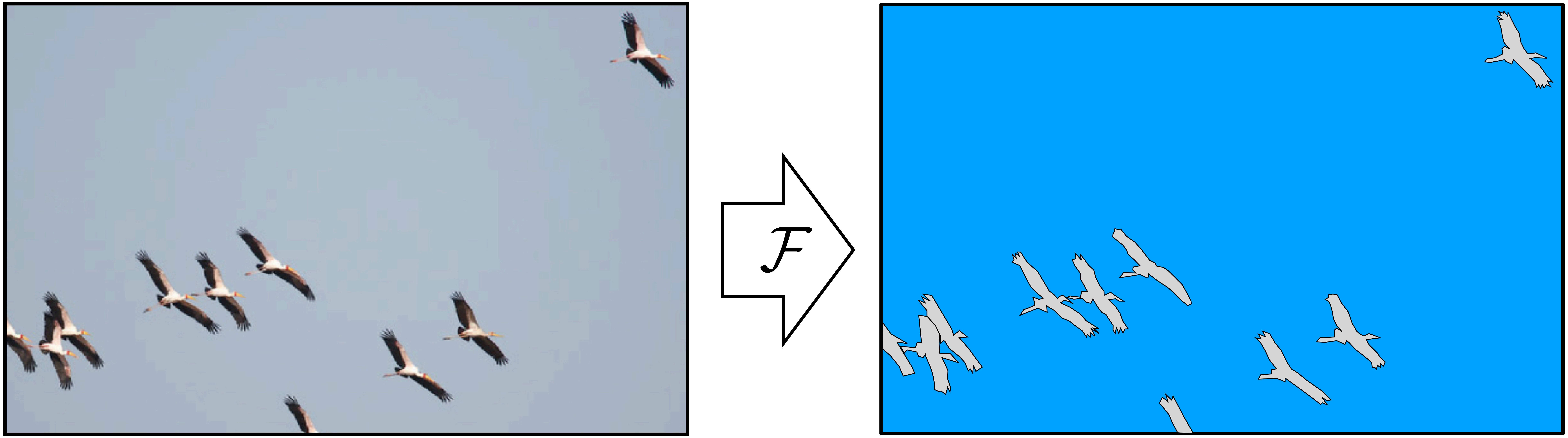


Why deep learning



How do data science techniques scale with amount of data?

[Slide credit: Andrew Ng]



[Photo credit: Fredo Durand]

(Colors represent one-hot codes)

$$\arg\min_{\mathcal{F}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[L(\mathcal{F}(\mathbf{x}),\mathbf{y})]$$

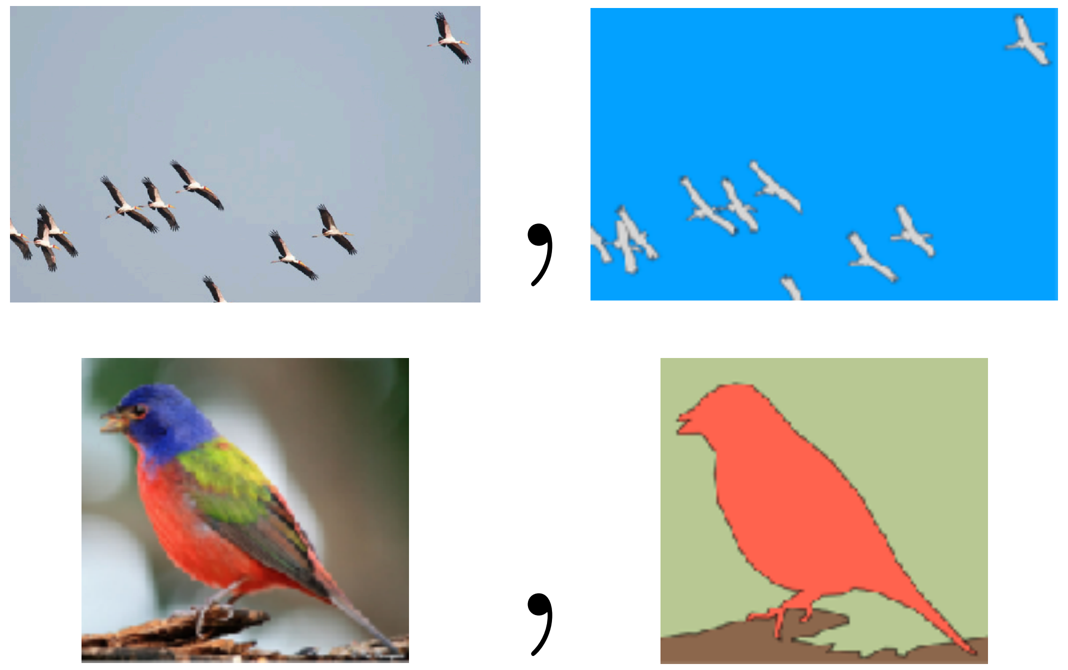
Hypothesis space

Objective function
(loss)

Semantic Segmentation

Data

$\mathbf{x}$ $\mathbf{y}$



⋮

$\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$

$\mathbf{y} \in \mathbb{R}^{H \times W \times K}$

$\rightarrow$

Learner

Objective

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} H(\mathbf{y}_i, \hat{\mathbf{y}}_i)$$

Hypothesis space

Convolutional neural net

Optimizer

Stochastic gradient descent

$\rightarrow \quad f$

Sat2Map

Data

x **y**

$\vdots$

$$\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$$
$$\mathbf{y} \in \mathbb{R}^{H \times W \times 3}$$

$\rightarrow$

Learner

Objective

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} (f_\theta(\mathbf{x})_i - y_i)^2$$

Hypothesis space

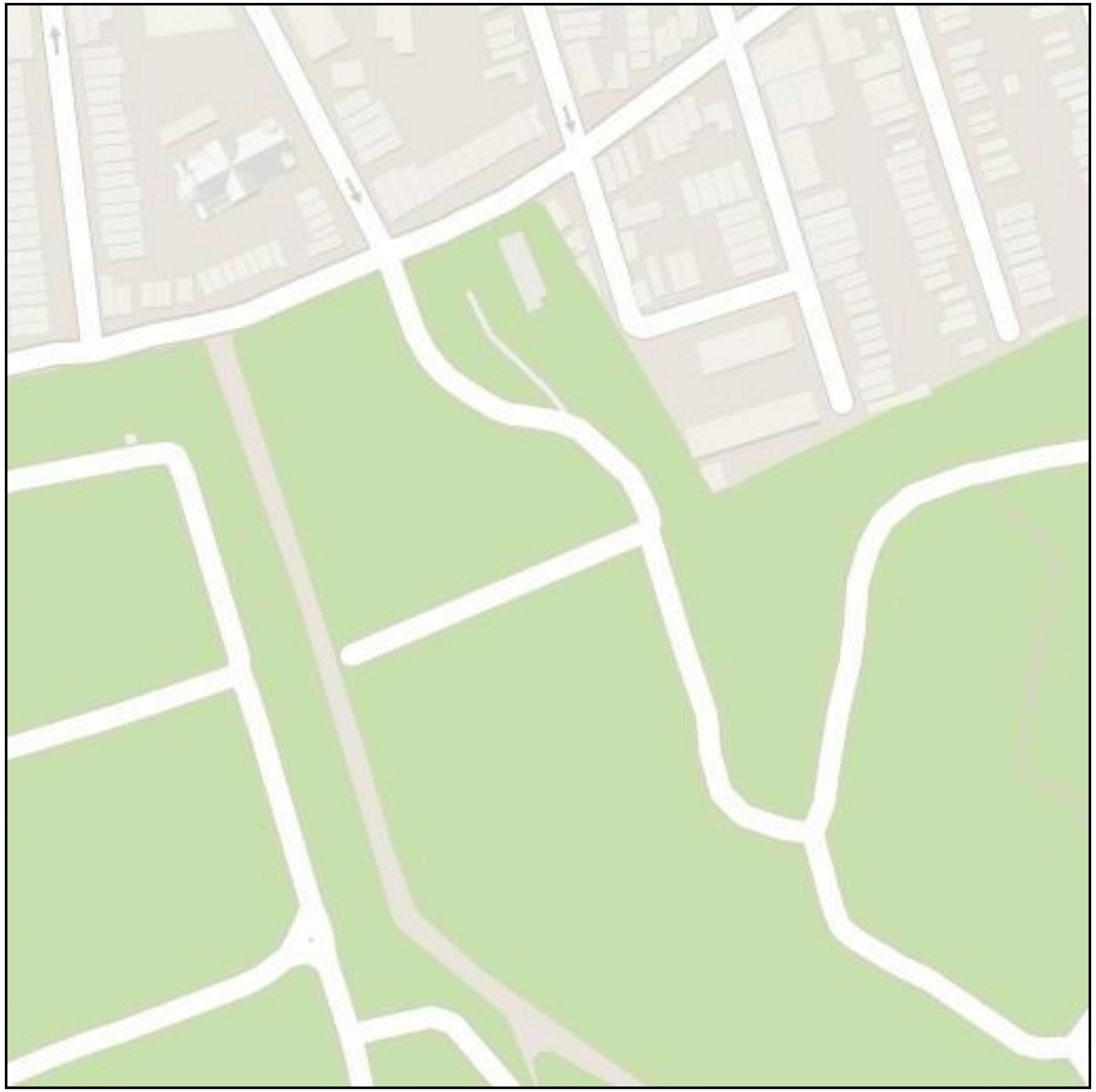
Convolutional neural net

Optimizer

Stochastic gradient descent

$\rightarrow f$

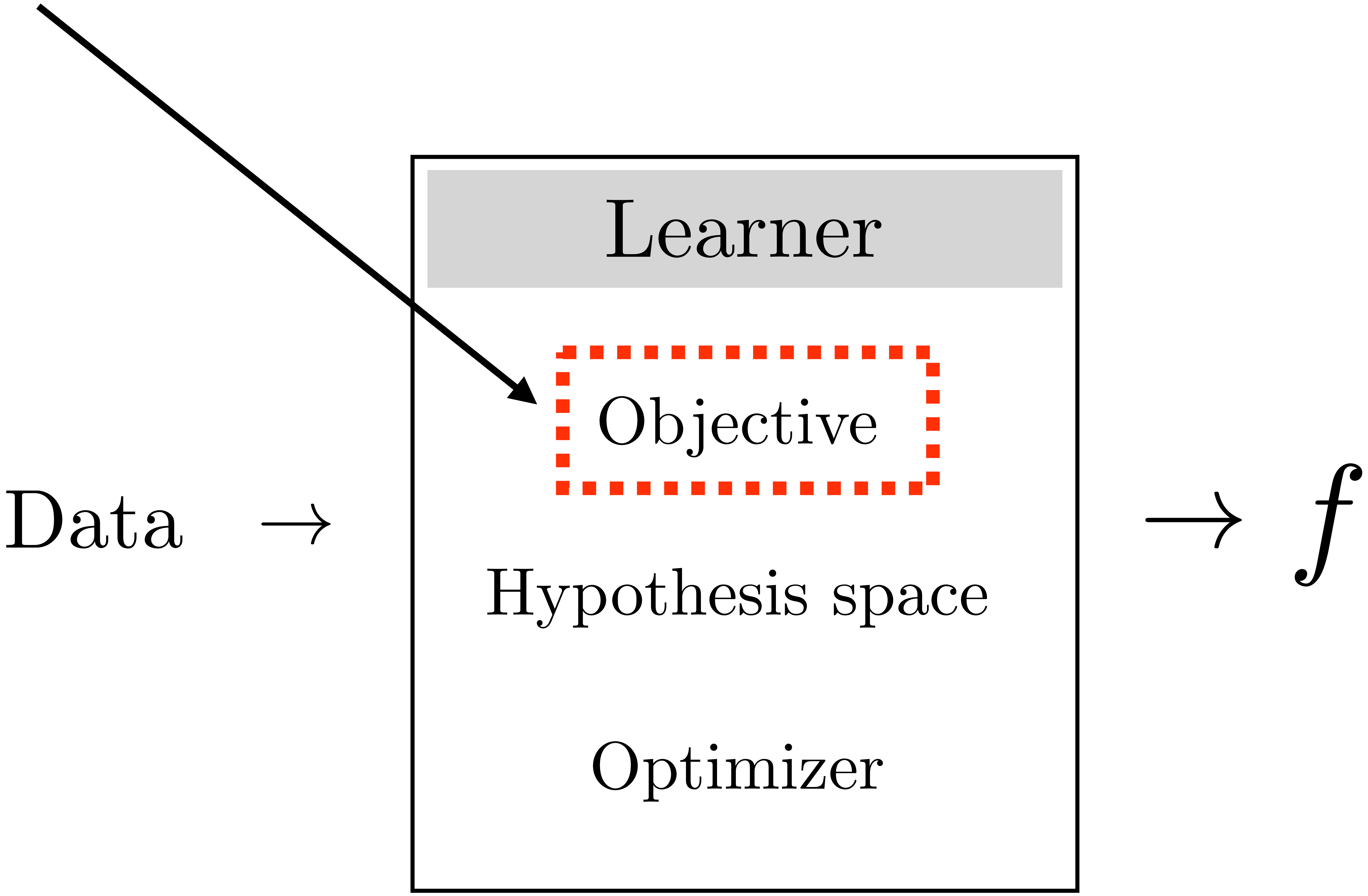
Input

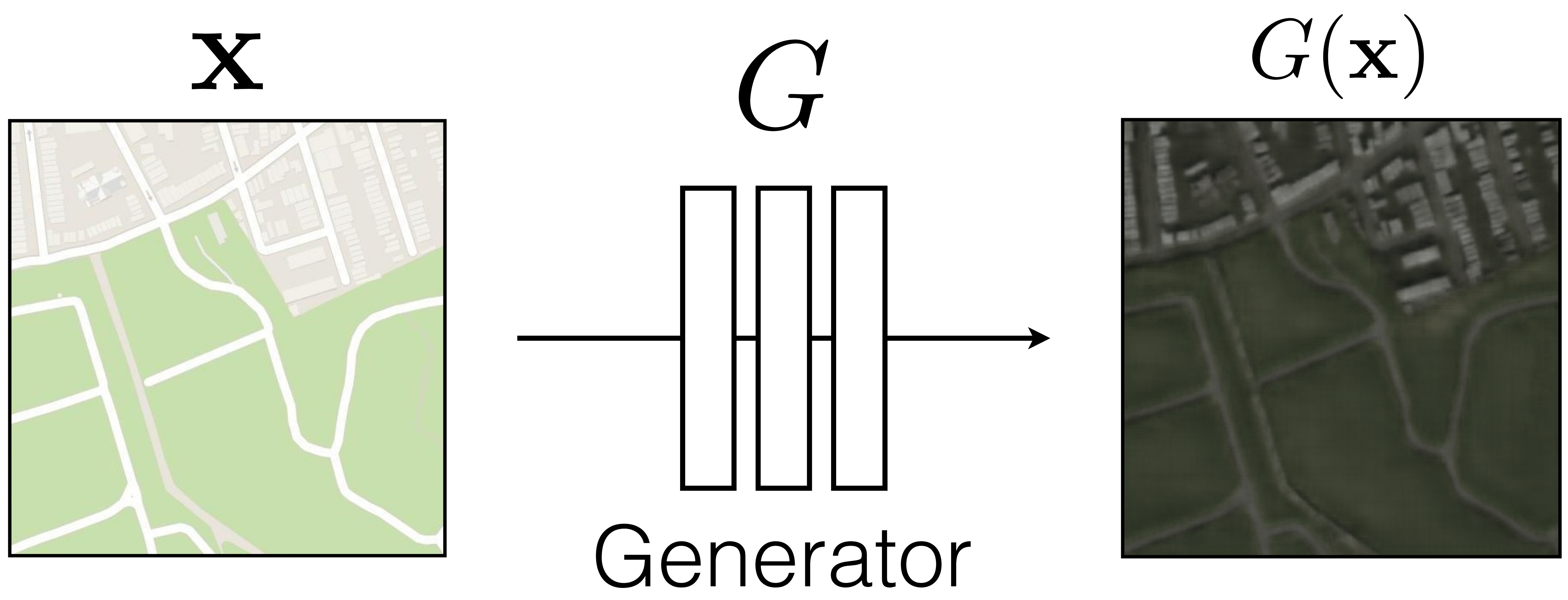


Deep net output

Structured prediction

Use an **objective** that can model structure! (e.g., a graphical model, a GAN, etc)

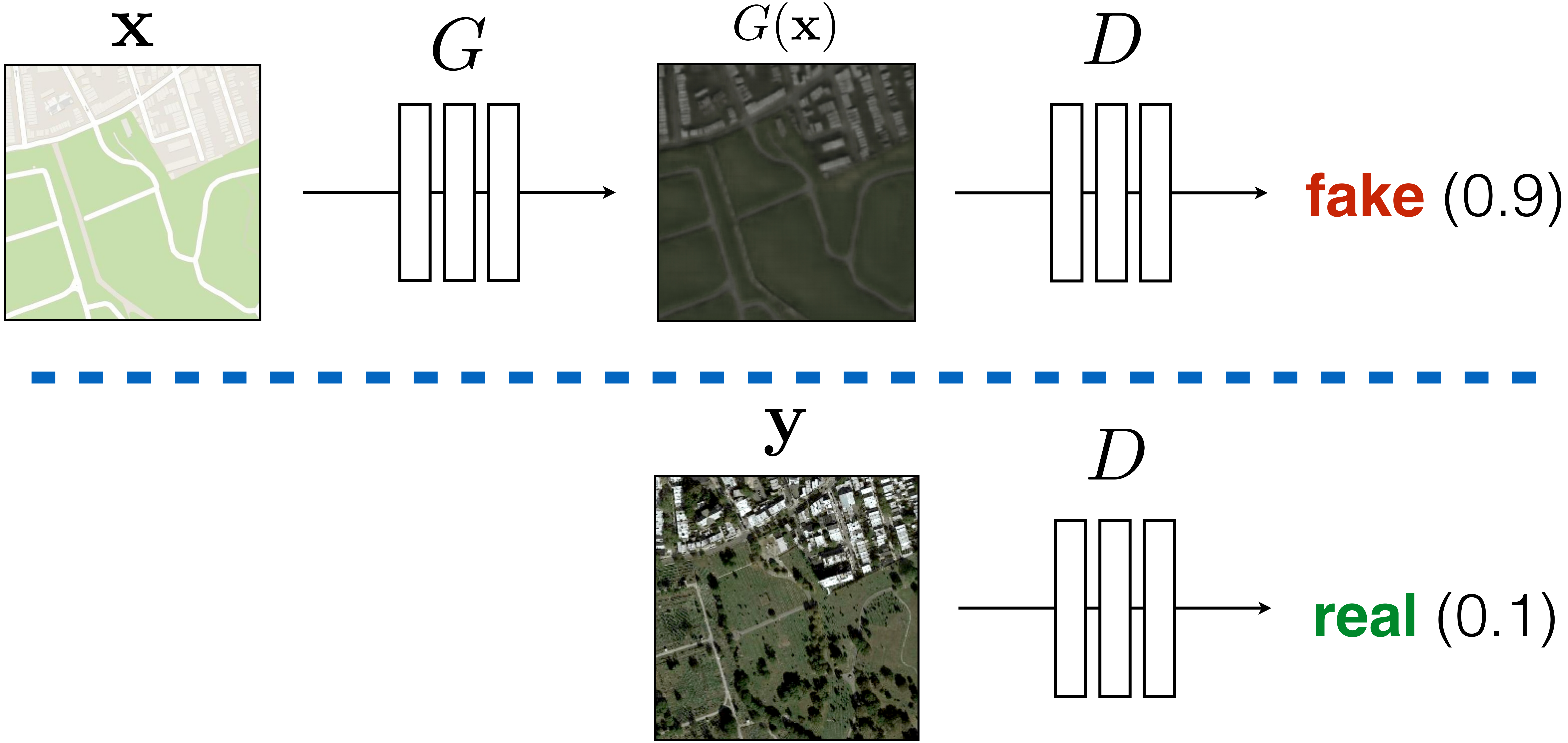




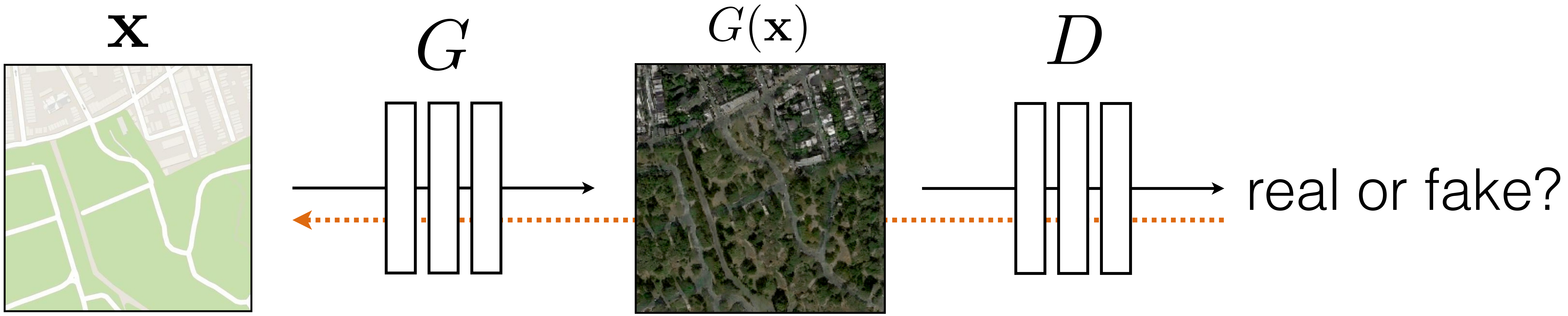
$\mathbf{x}$
G
Generator
$G(\mathbf{x})$

G tries to synthesize fake images that fool **D**

D tries to identify the fakes

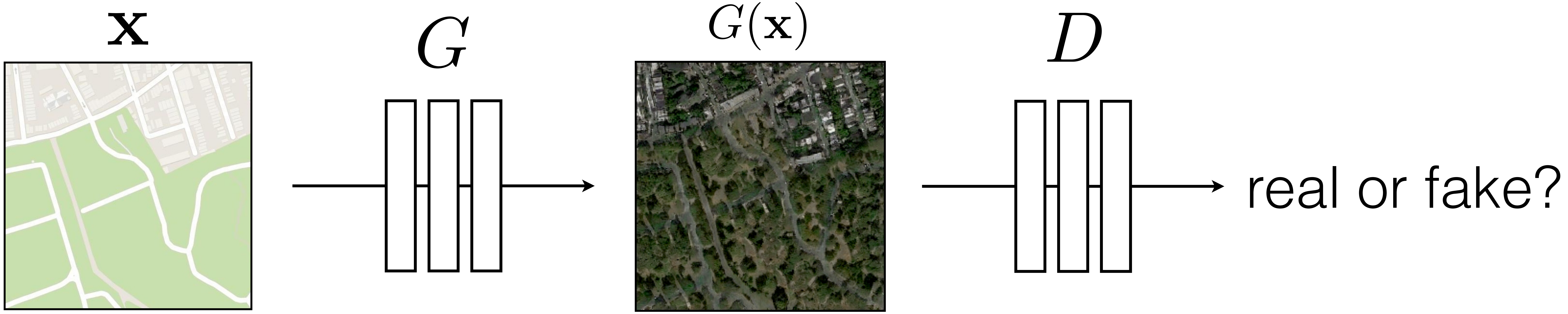


$$\arg\max_{D} \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \ + \ \log(1 - D(\mathbf{y}))\ ]$$



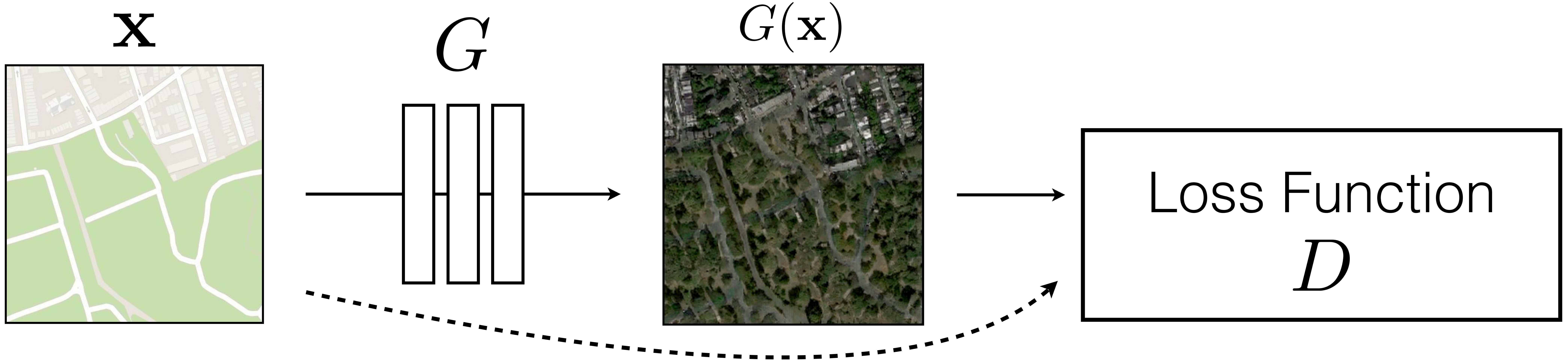
G tries to synthesize fake images that ***fool*** **D**:

$$\arg\min_{G} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$



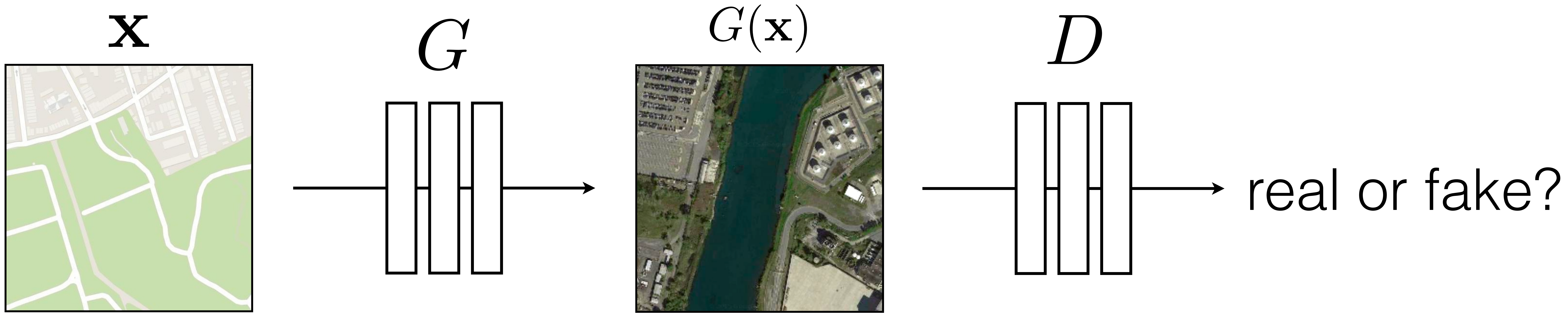
G tries to synthesize fake images that ***fool*** the ***best*** **D**:

$$\arg \min_{G} \max_{D} \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \ + \ \log(1 - D(\mathbf{y}))\ ]$$

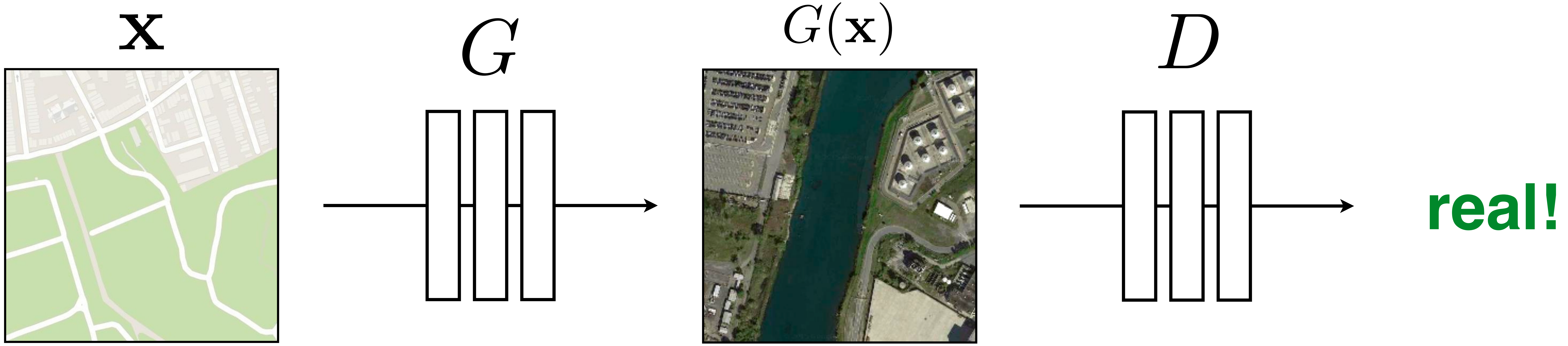


G’s perspective: **D** is a loss function.

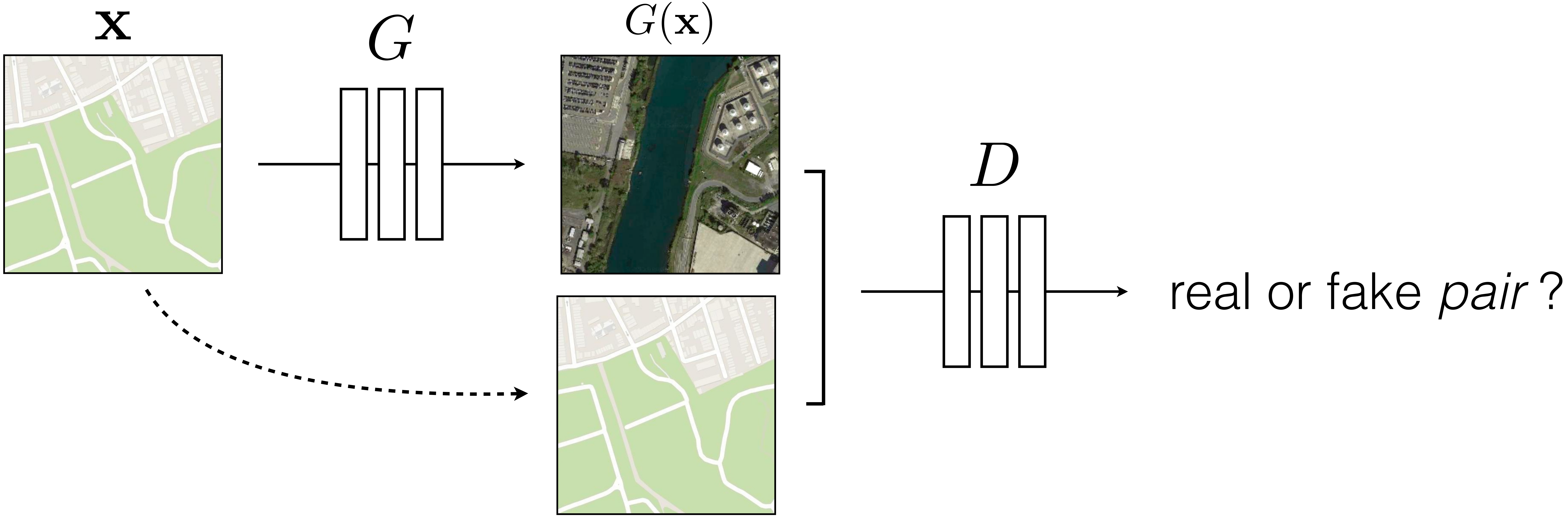
Rather than being hand-designed, it is *learned* and *highly structured*.



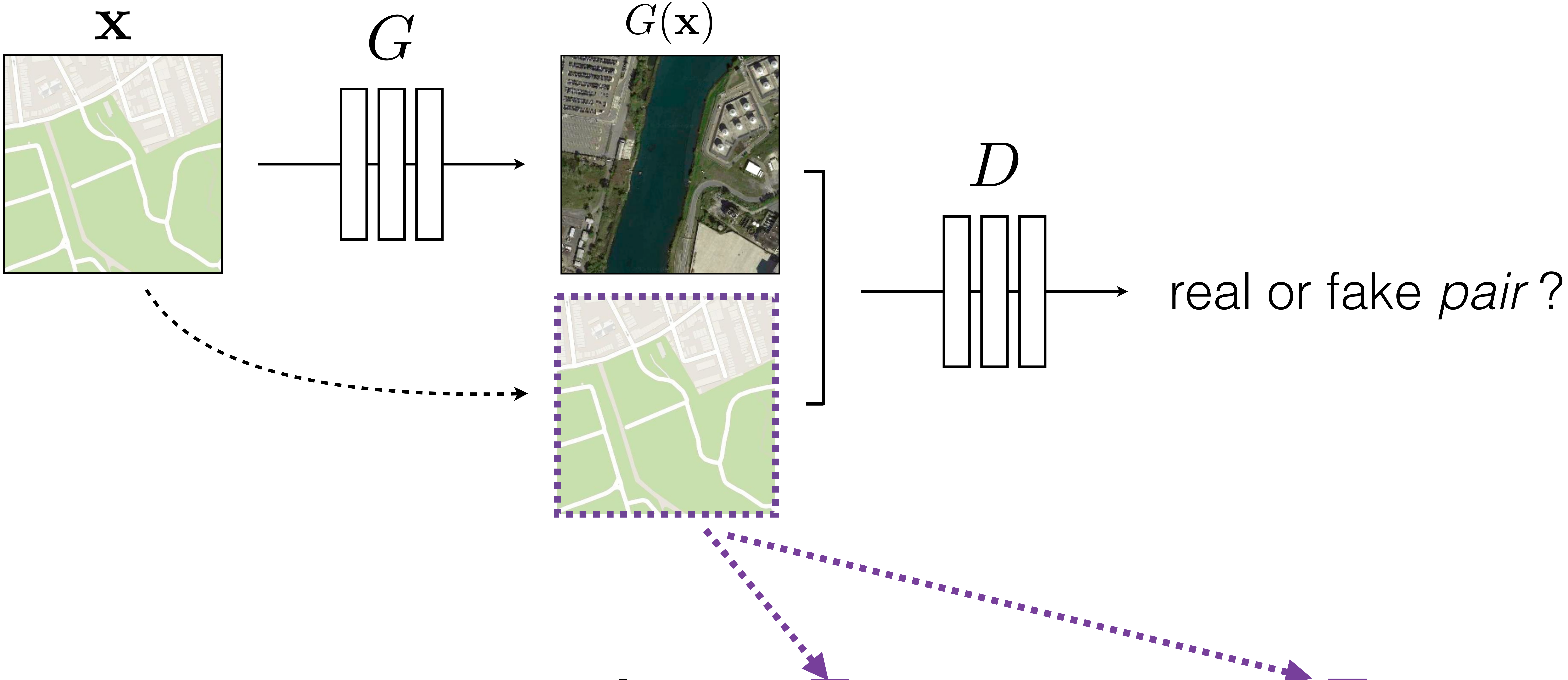
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$



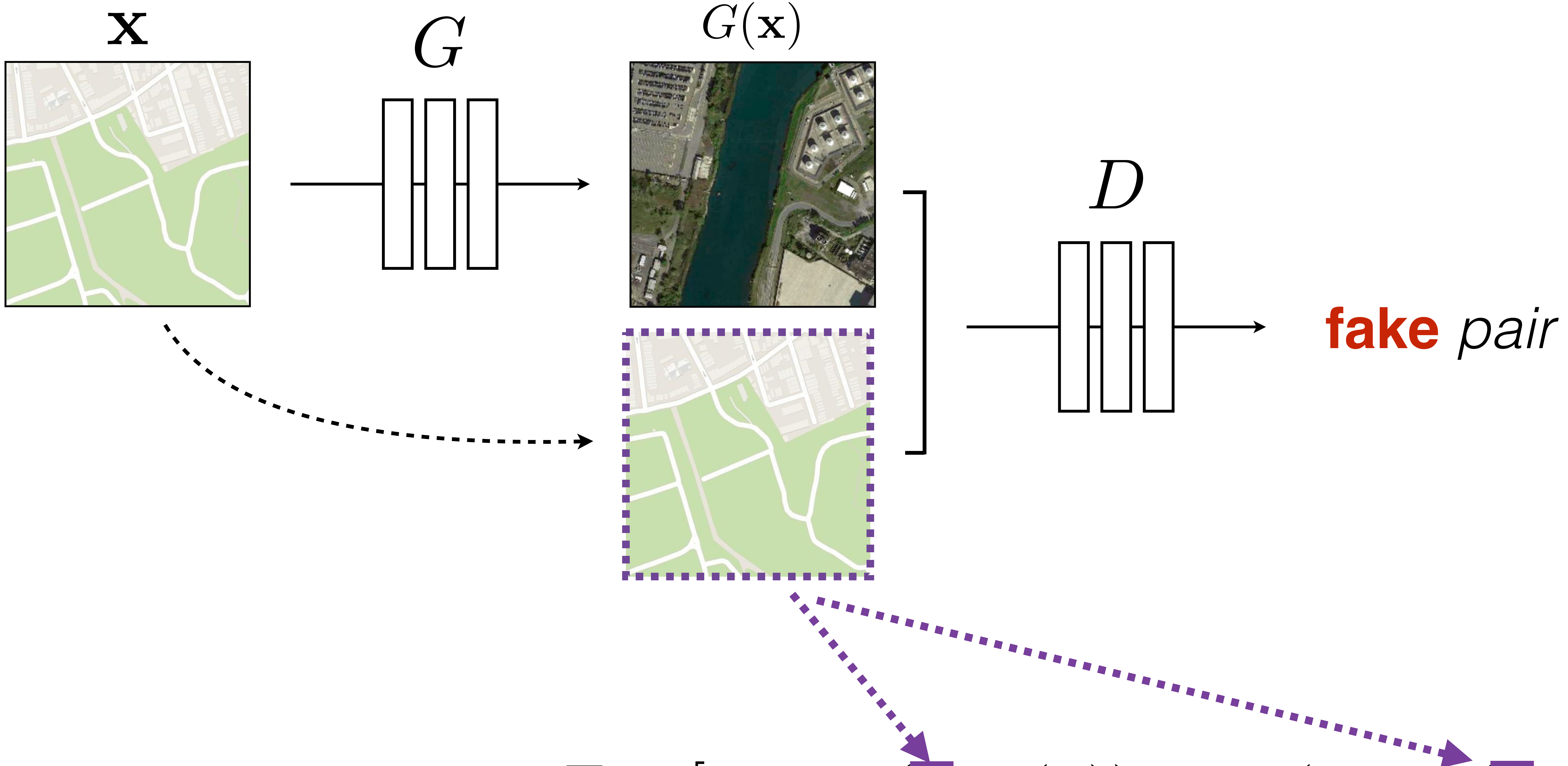
$$\arg\min_G \max_D \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$



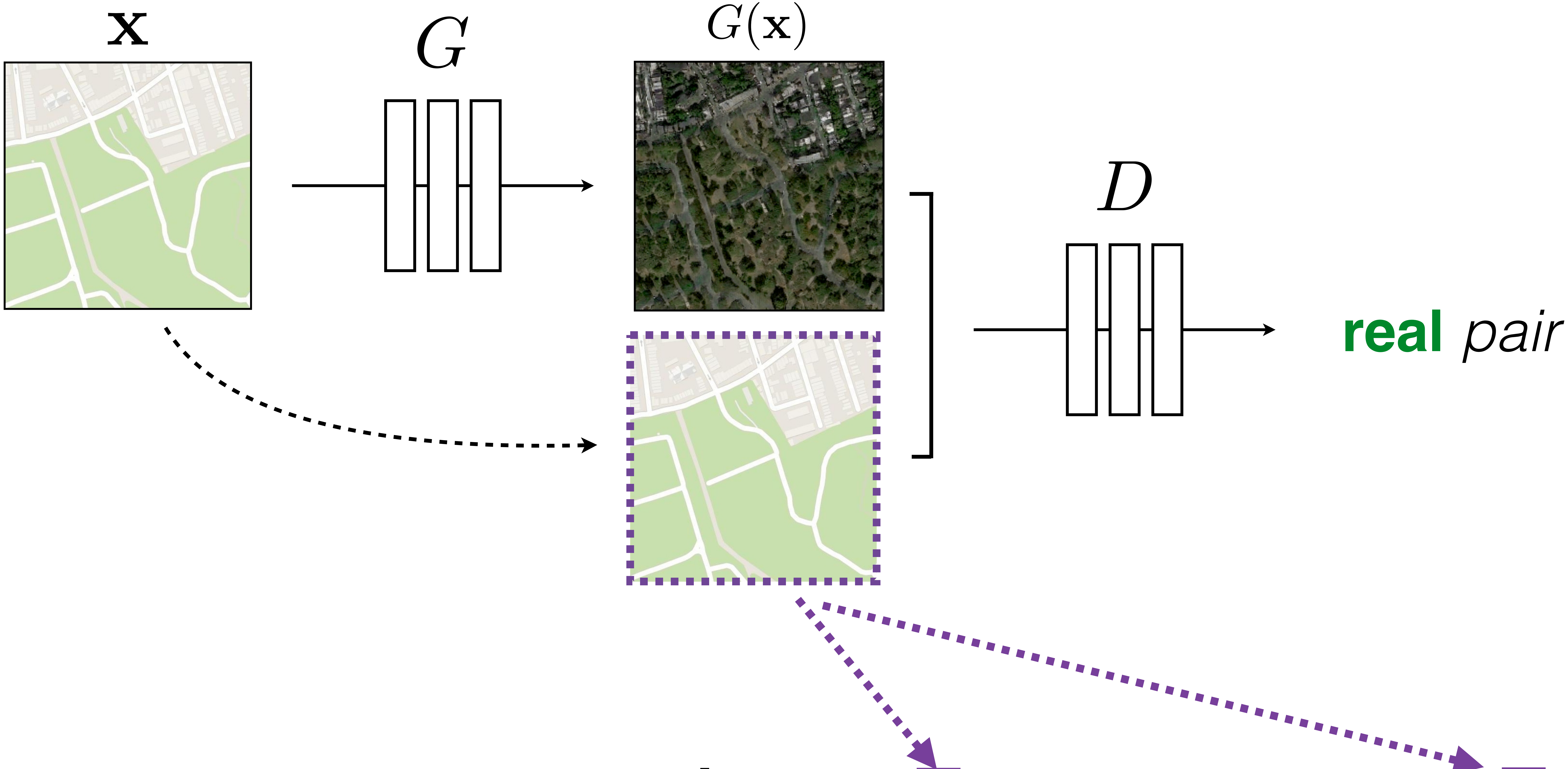
$$\arg\min_G \max_D \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \ + \ \log(1 - D(\mathbf{y}))\ ]$$



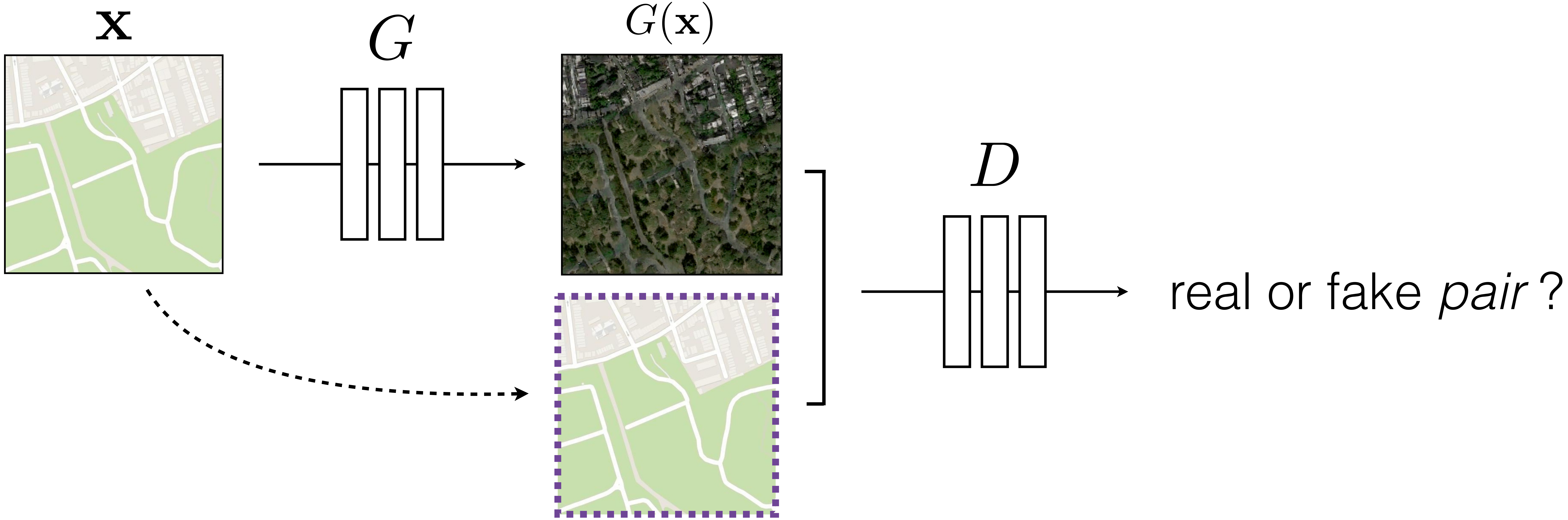
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$



$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y})) \;]$$



$$\arg\min_{G}\max_{D} \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y})) \;]$$



$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x},\mathbf{y}))\ ]$$

Training Details: Loss function

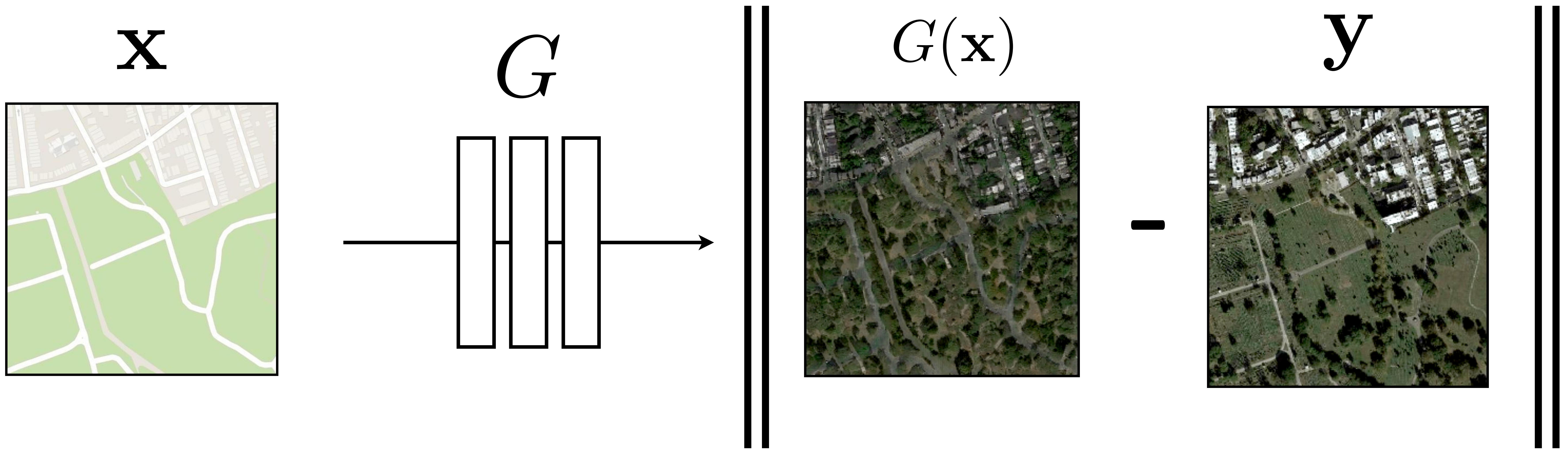
Conditional GAN

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

Training Details: Loss function

Conditional GAN

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



Stable training + fast convergence

[c.f. Pathak et al. CVPR 2016]

Input Output Groundtruth

Data from [maps.google.com]

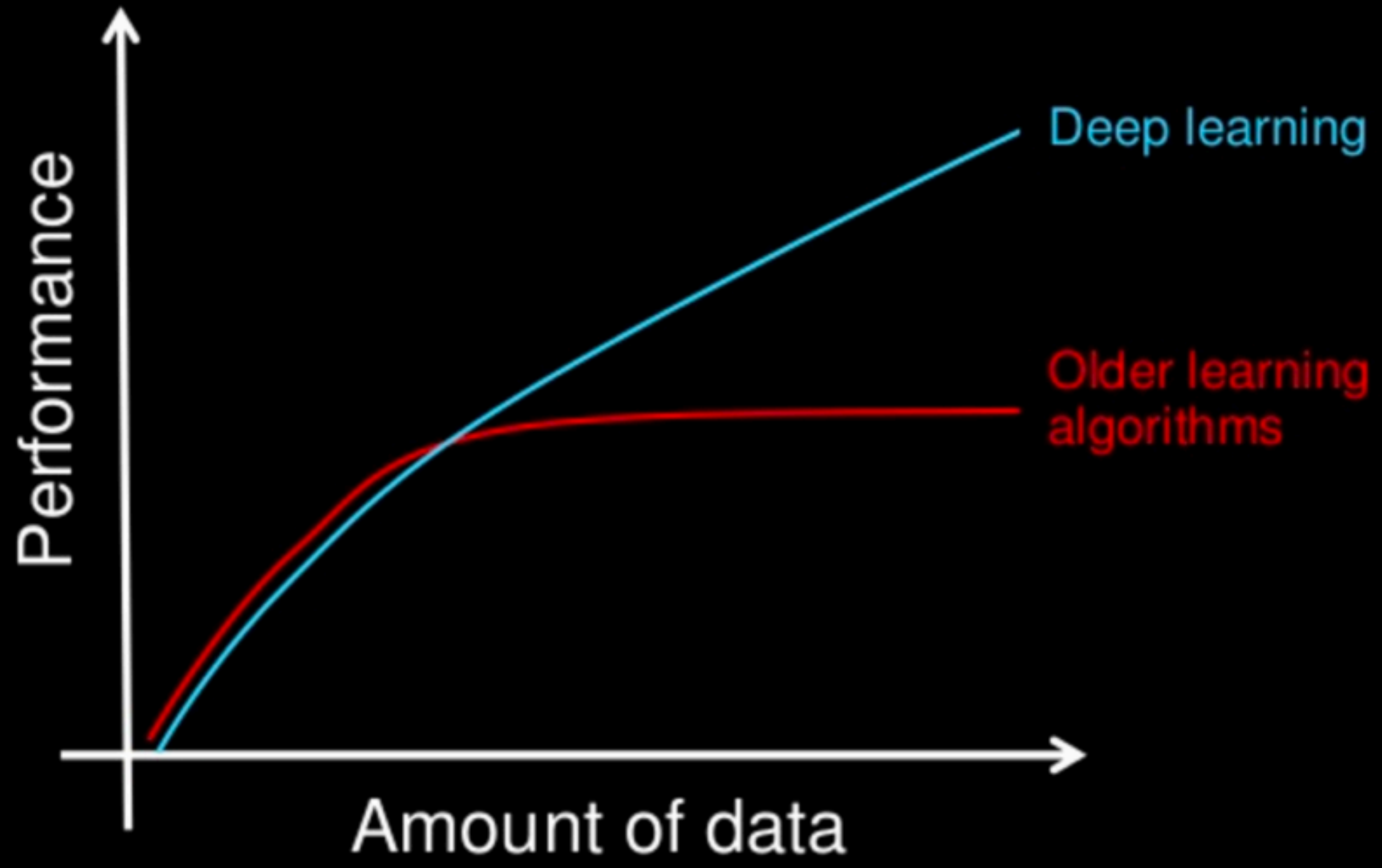
Input

Output

Groundtruth

Data from [maps.google.com]

Why deep learning

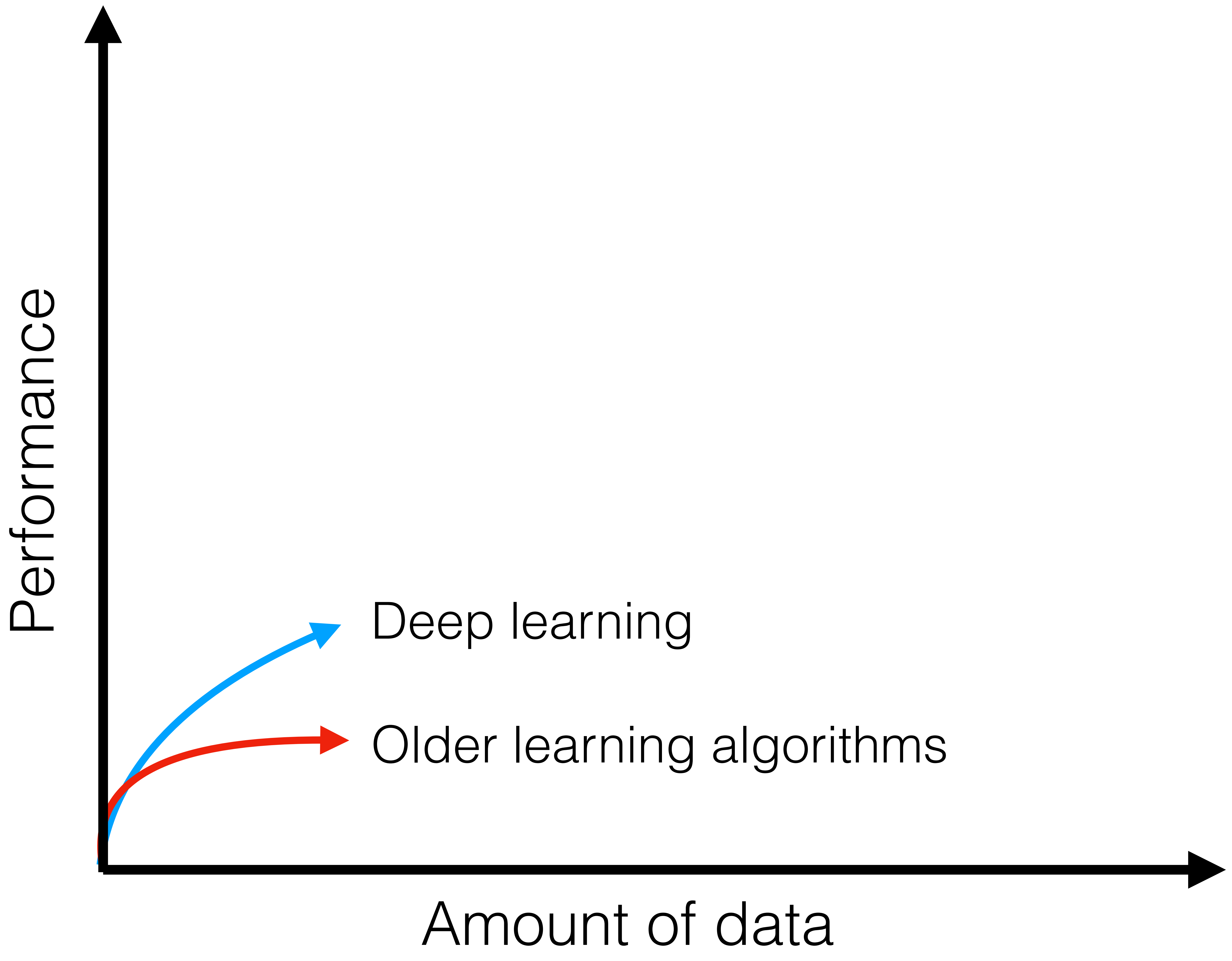


How do data science techniques scale with amount of data?

[Slide credit: Andrew Ng]

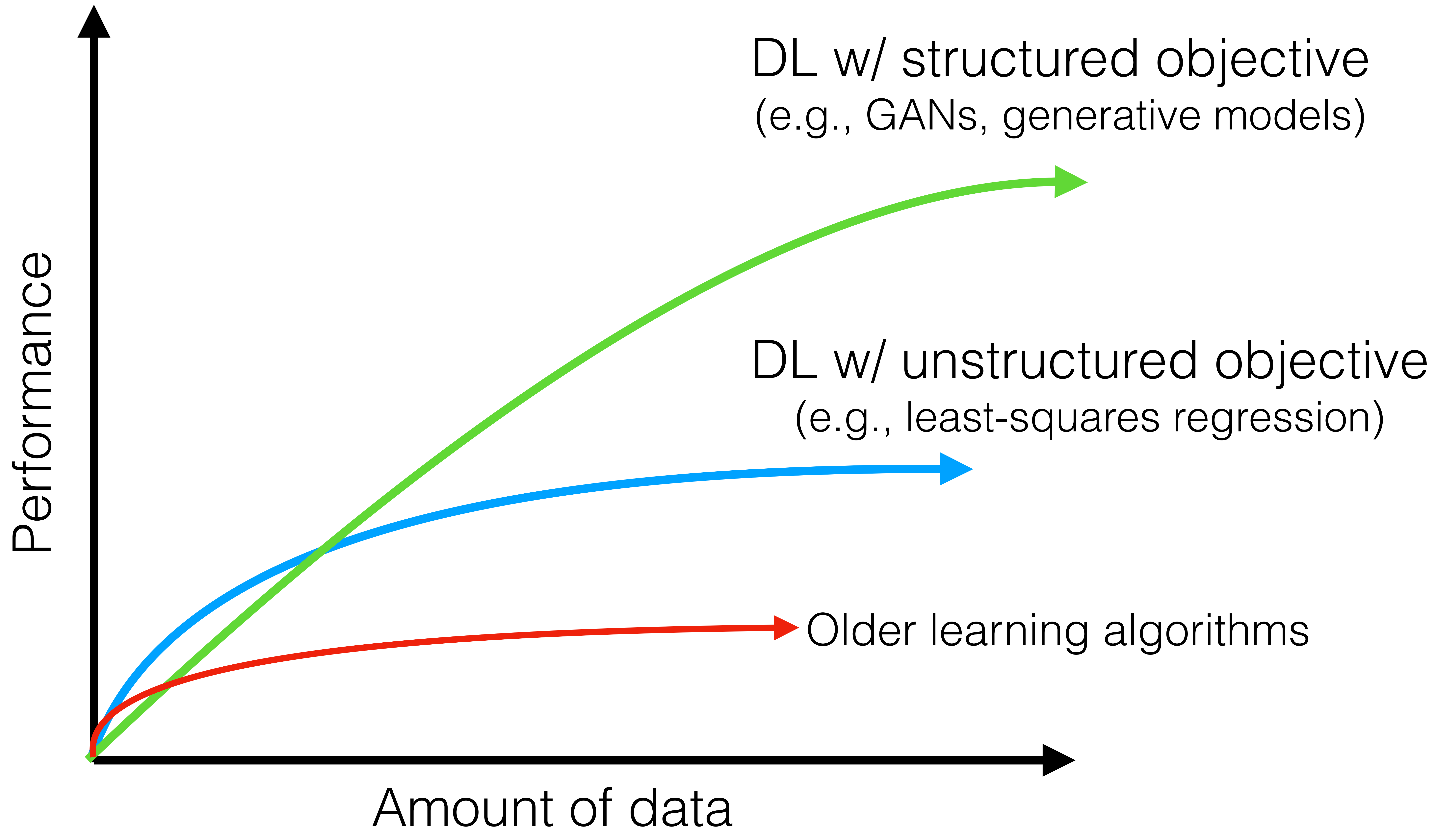
Why structured objectives

(cartoon)

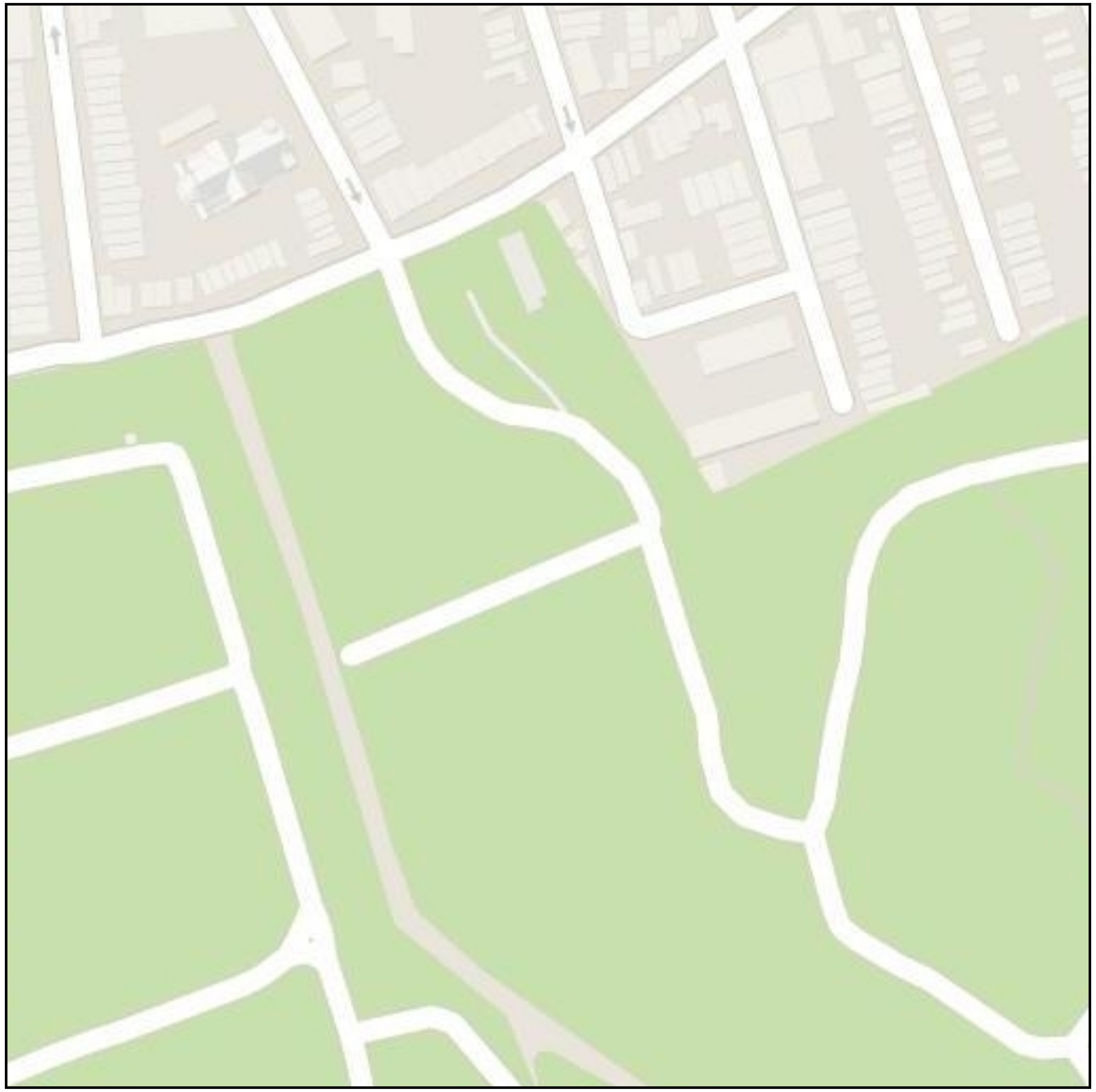


Why structured objectives

(cartoon)

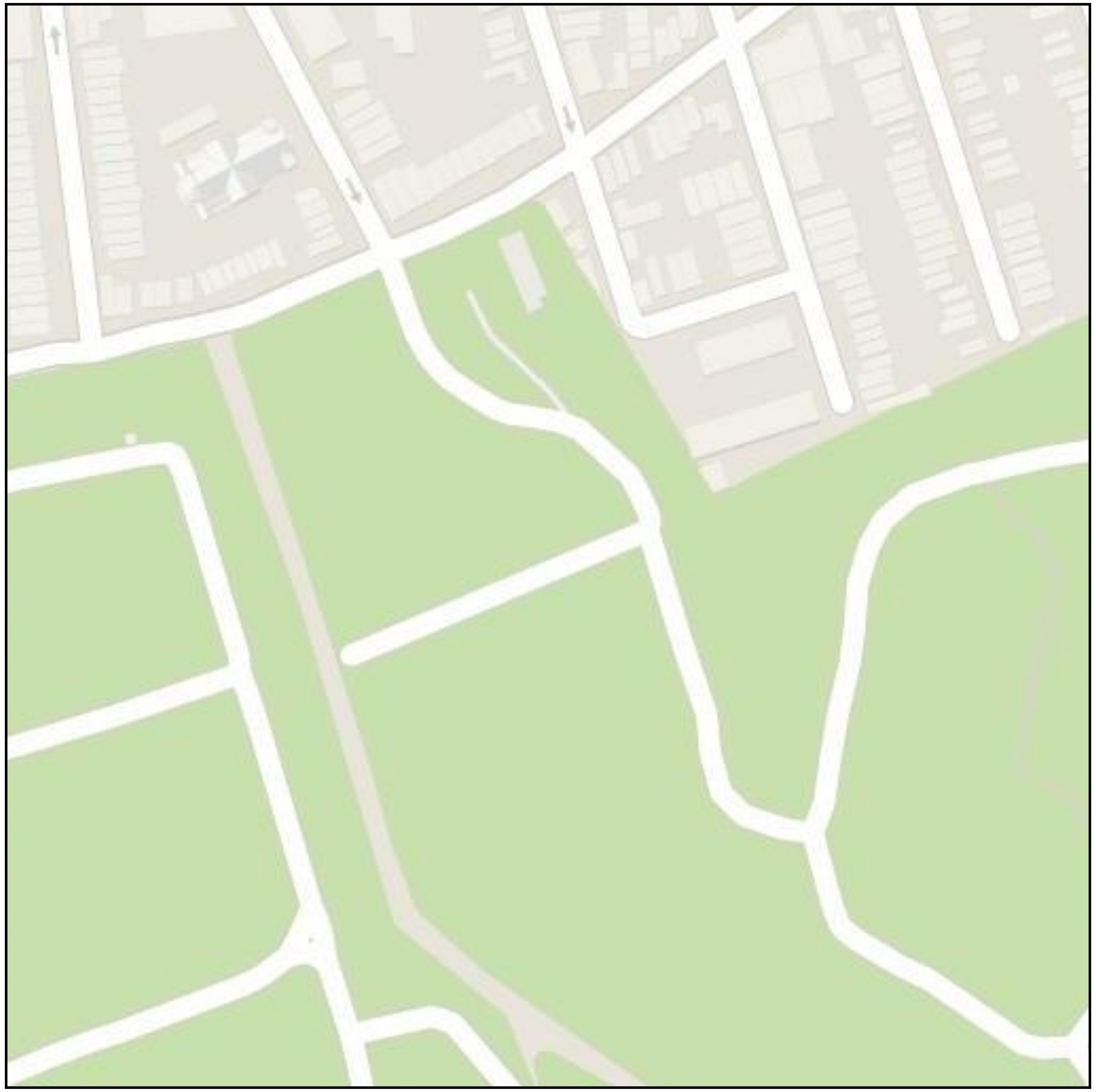


Input



Unstructured prediction (L1)

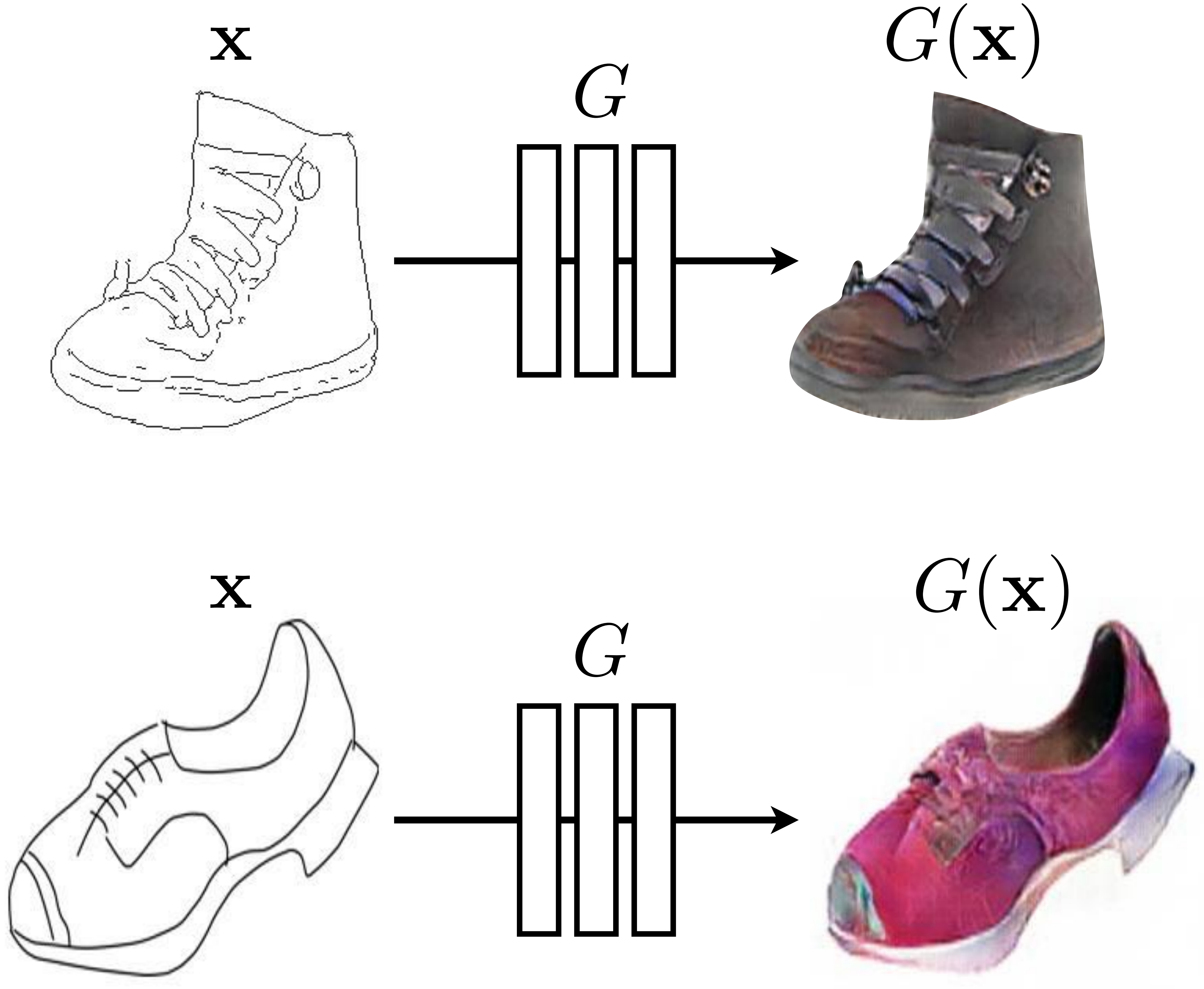
Input



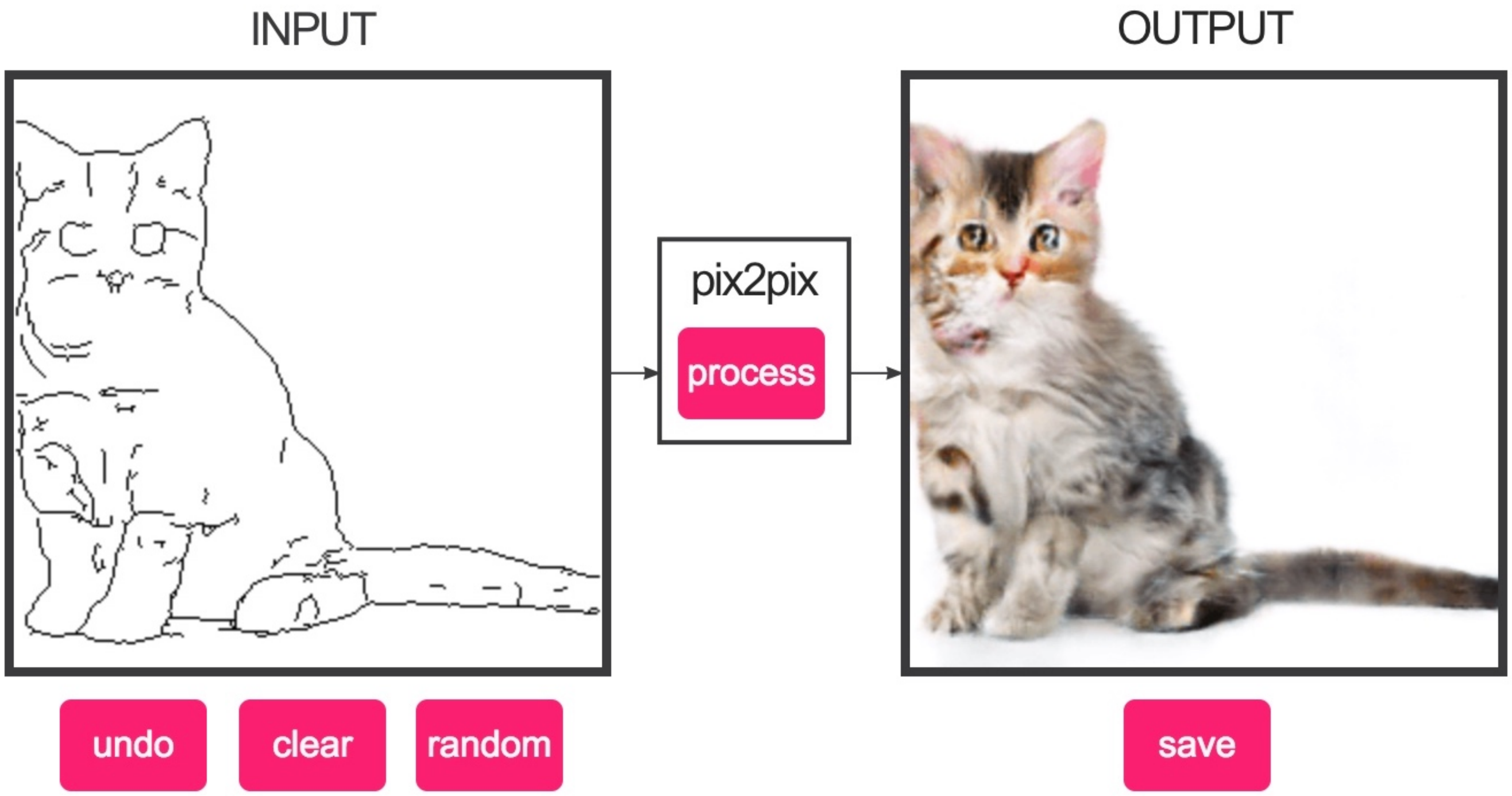
Structured Prediction (cGAN)

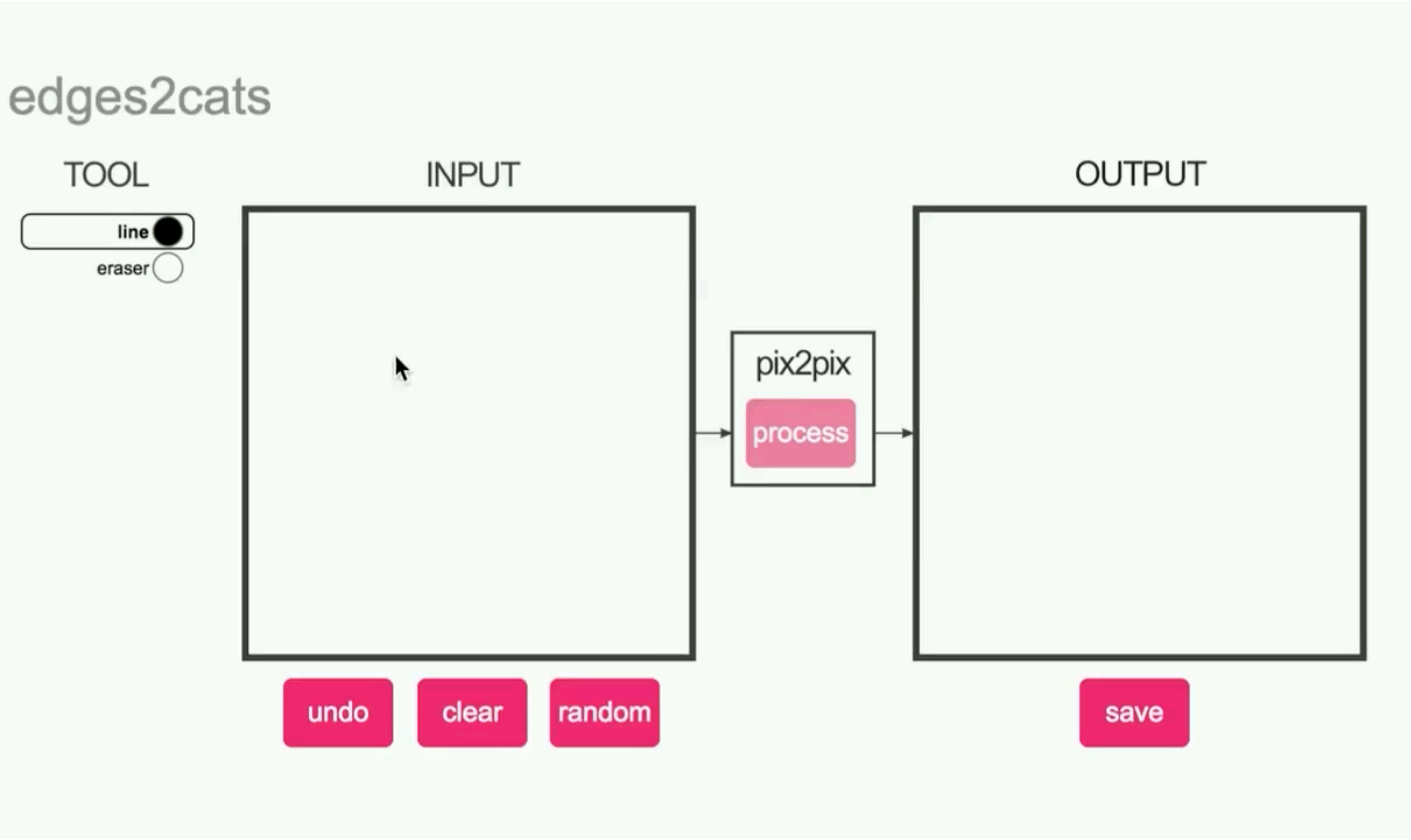
Training data

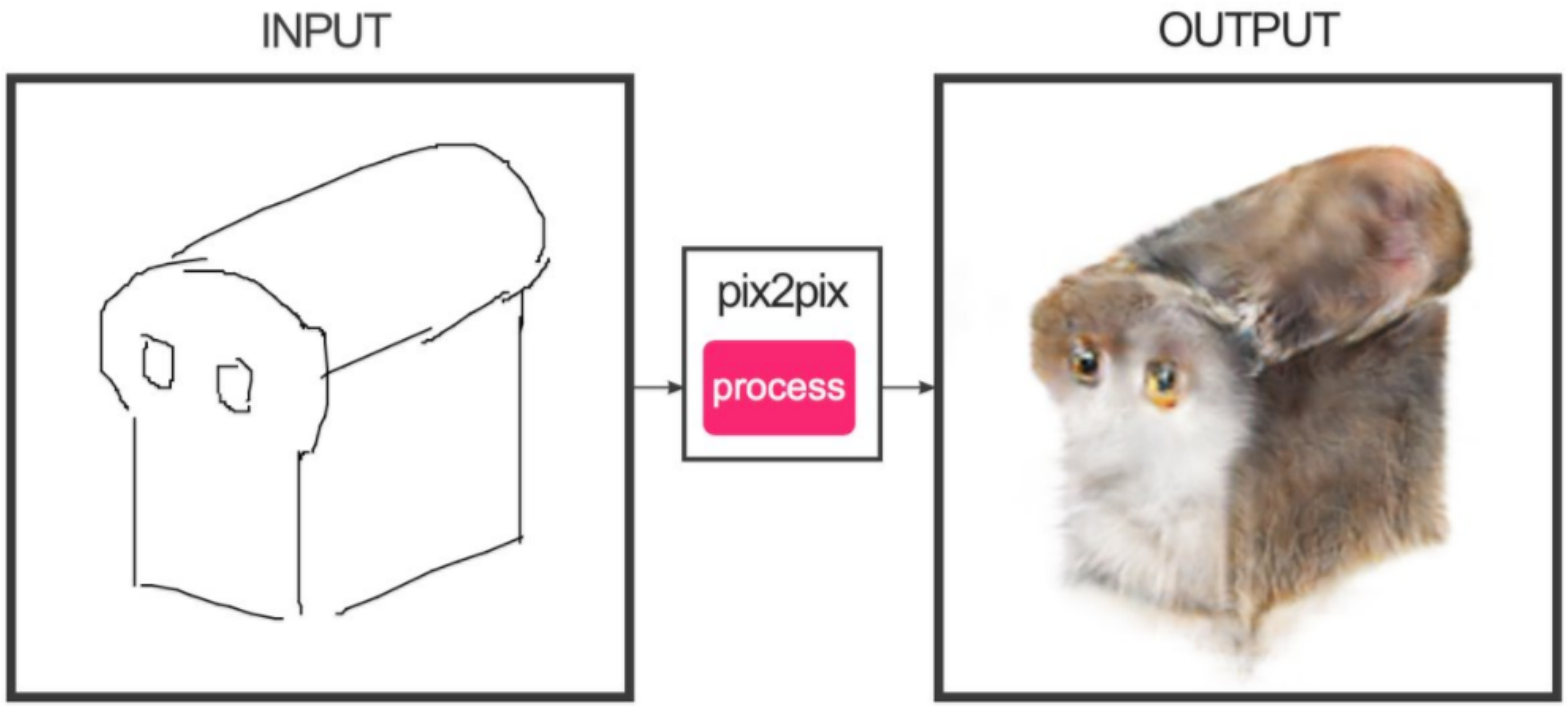
[HED, Xie & Tu, 2015]



#edges2cats [Chris Hesse]



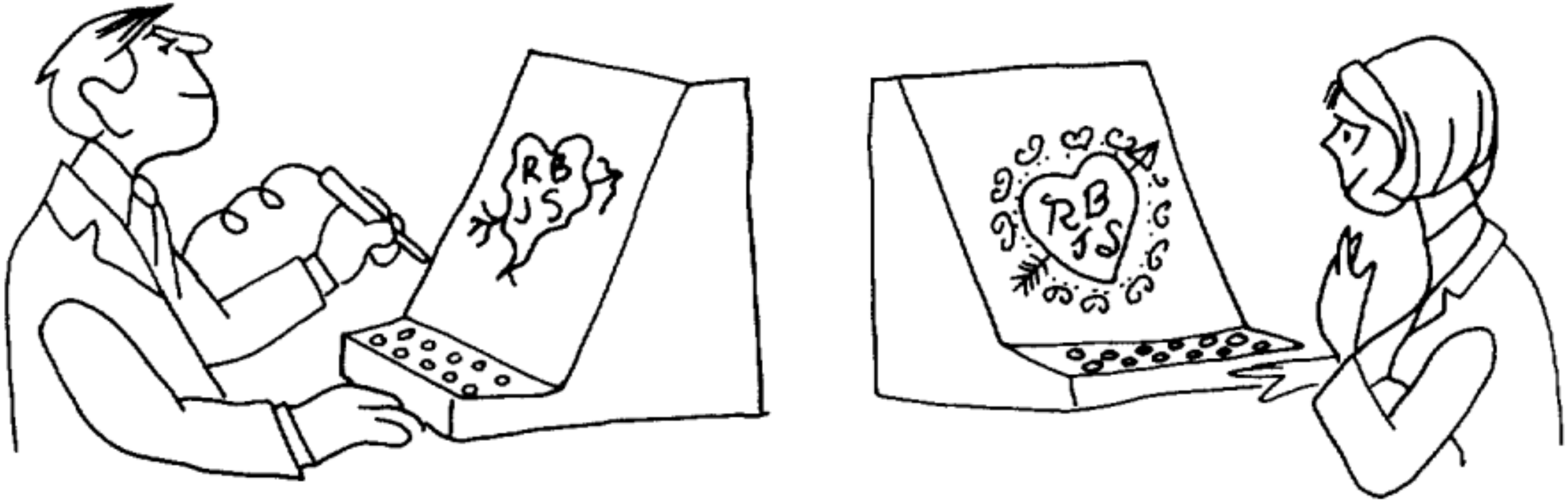

edges2cats
TOOL
line
eraser
INPUT
pix2pix
process
OUTPUT
undo
clear
random
save



Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

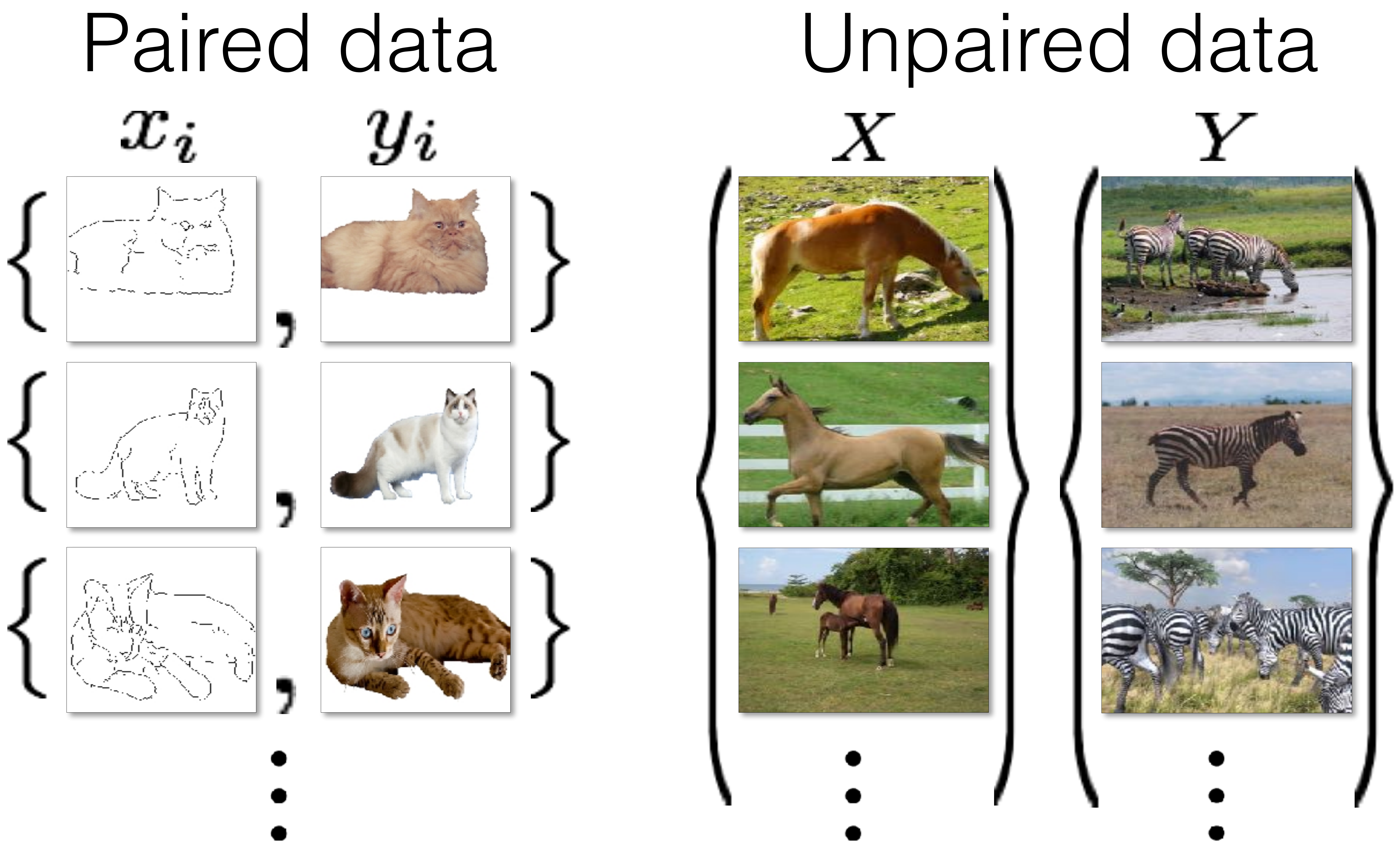
1. Image synthesis
2. Structured prediction
3. **Domain mapping**

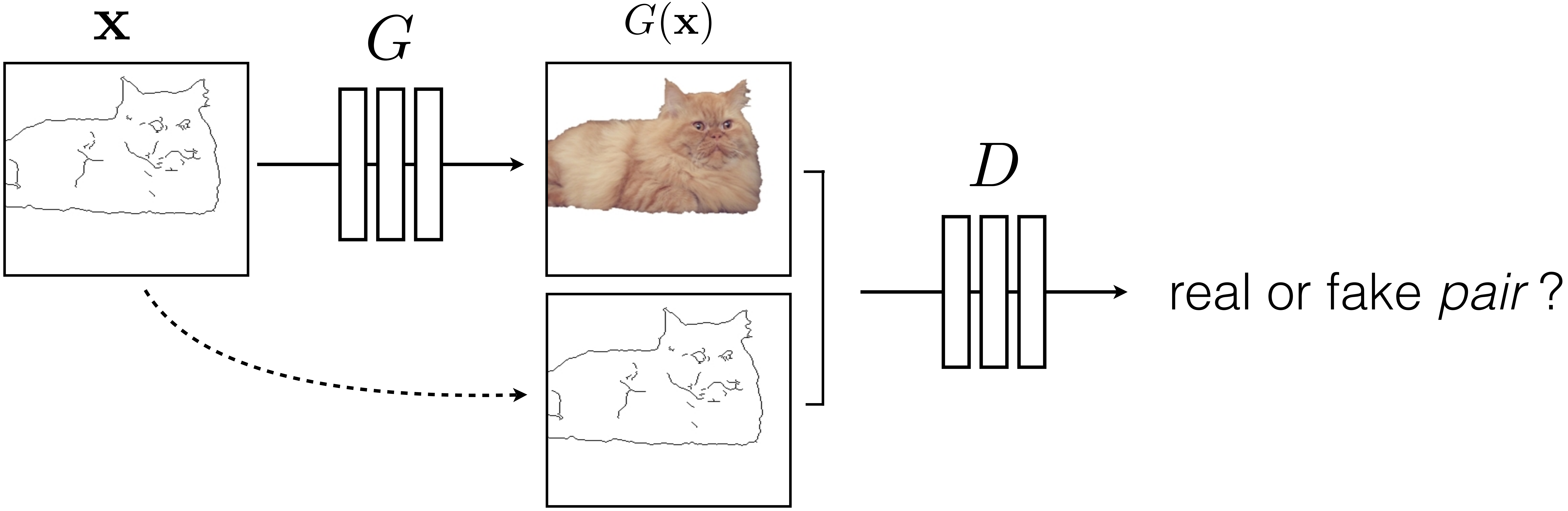


Domain mapping

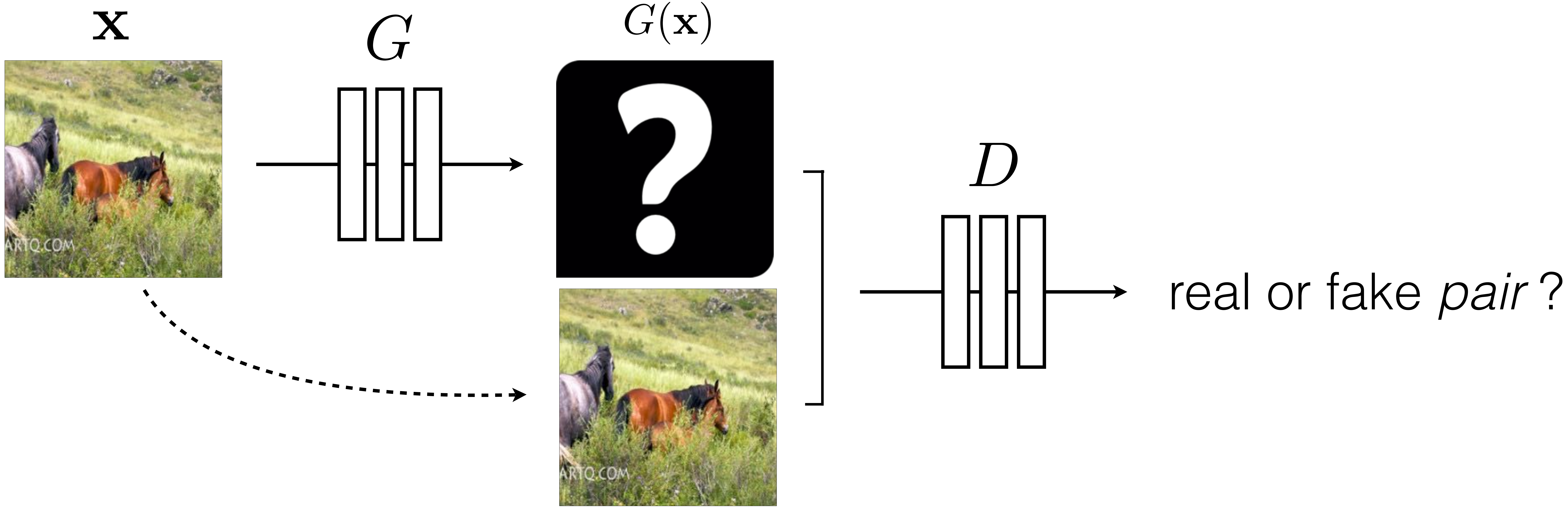
[Includes slides from Jun-Yan Zhu, Taesung Park]

[Cartoon: The Computer as a Communication Device, Licklider & Taylor 1968]


Paired data
Unpaired data
x_i
y_i
X
Y

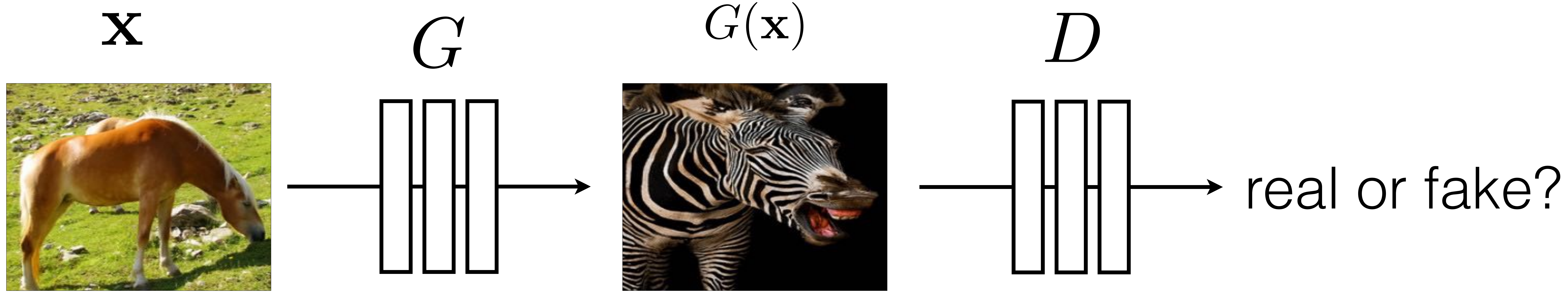


$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\ ]$$



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

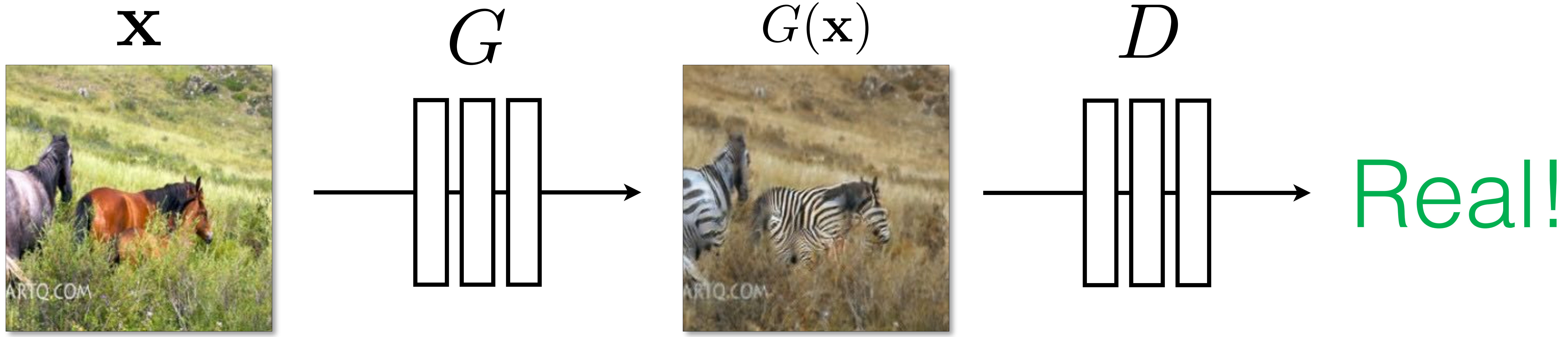
No input-output pairs!

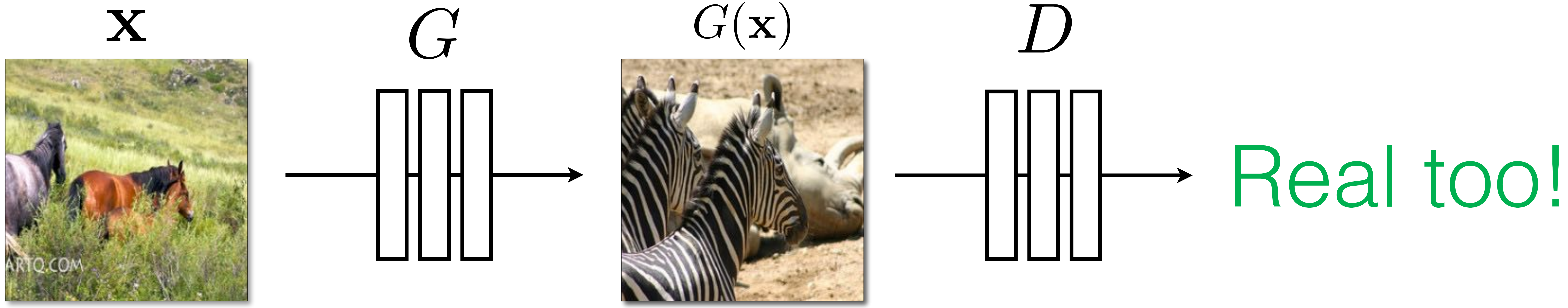


$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$

Usually loss functions check if output matches a target *instance*

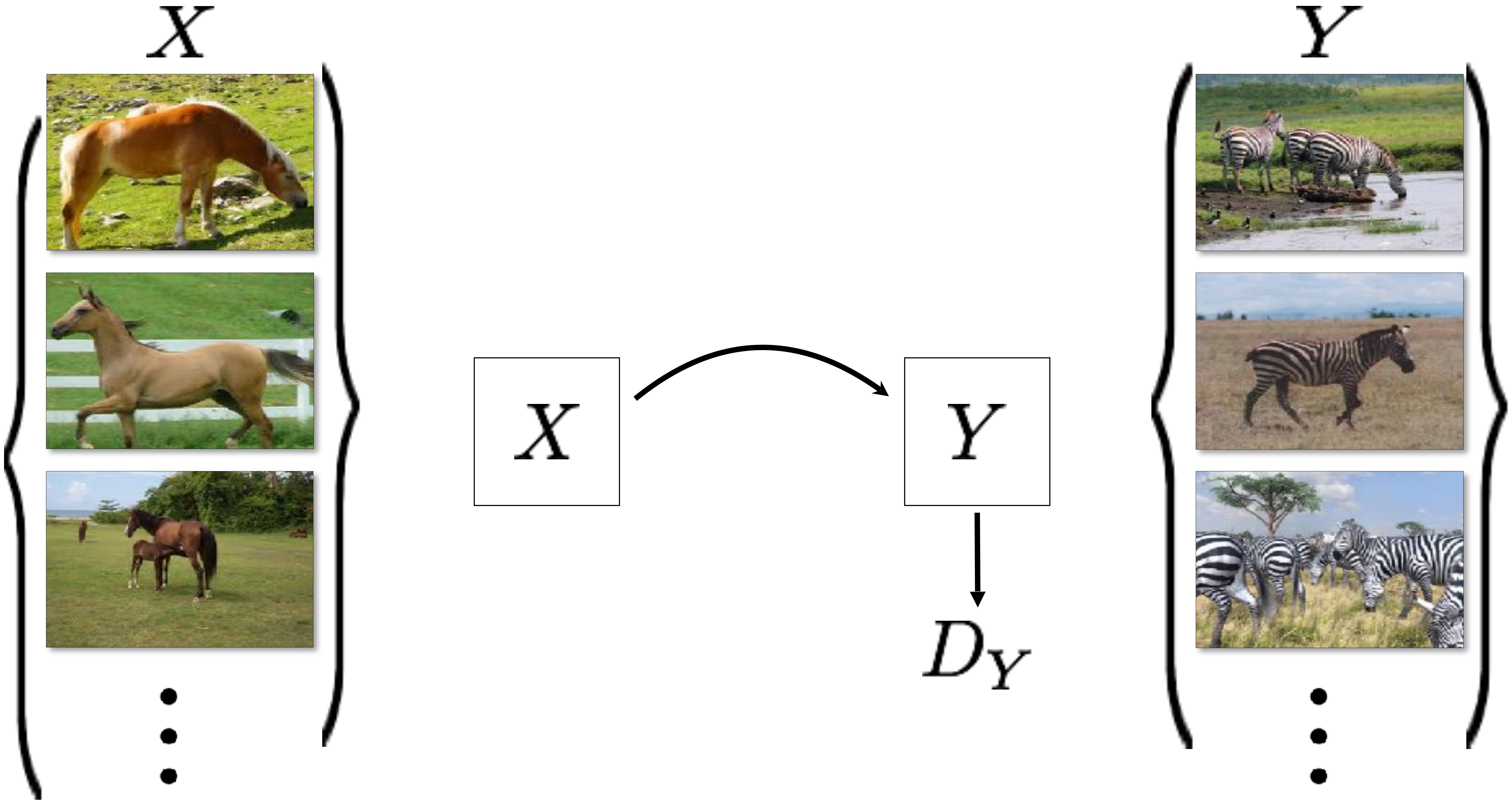
GAN loss checks if output is part of an admissible *set*


x
G
G(x)
D
Real!



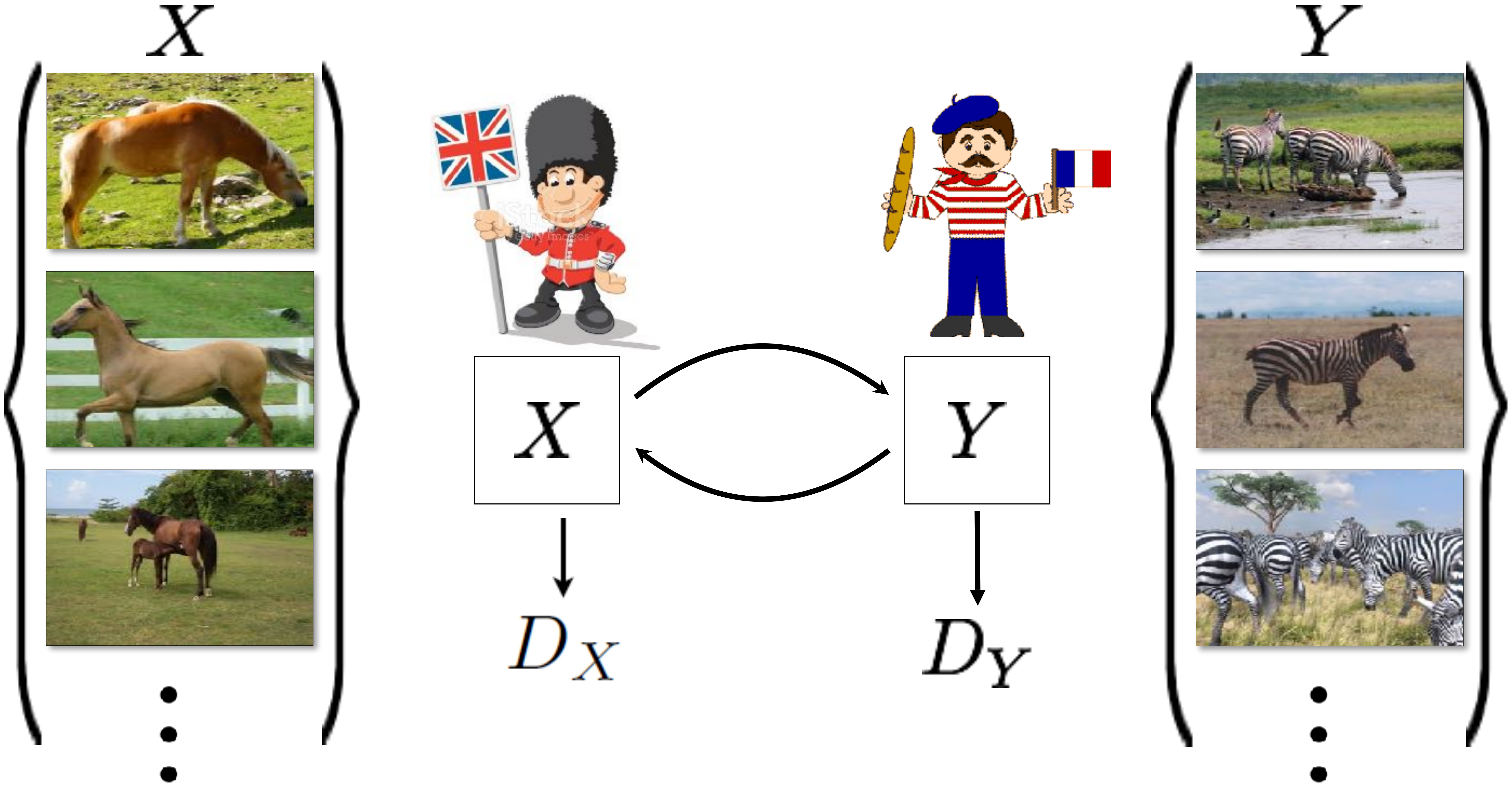
Nothing to force output to correspond to input

CycleGAN, or there and back aGAN

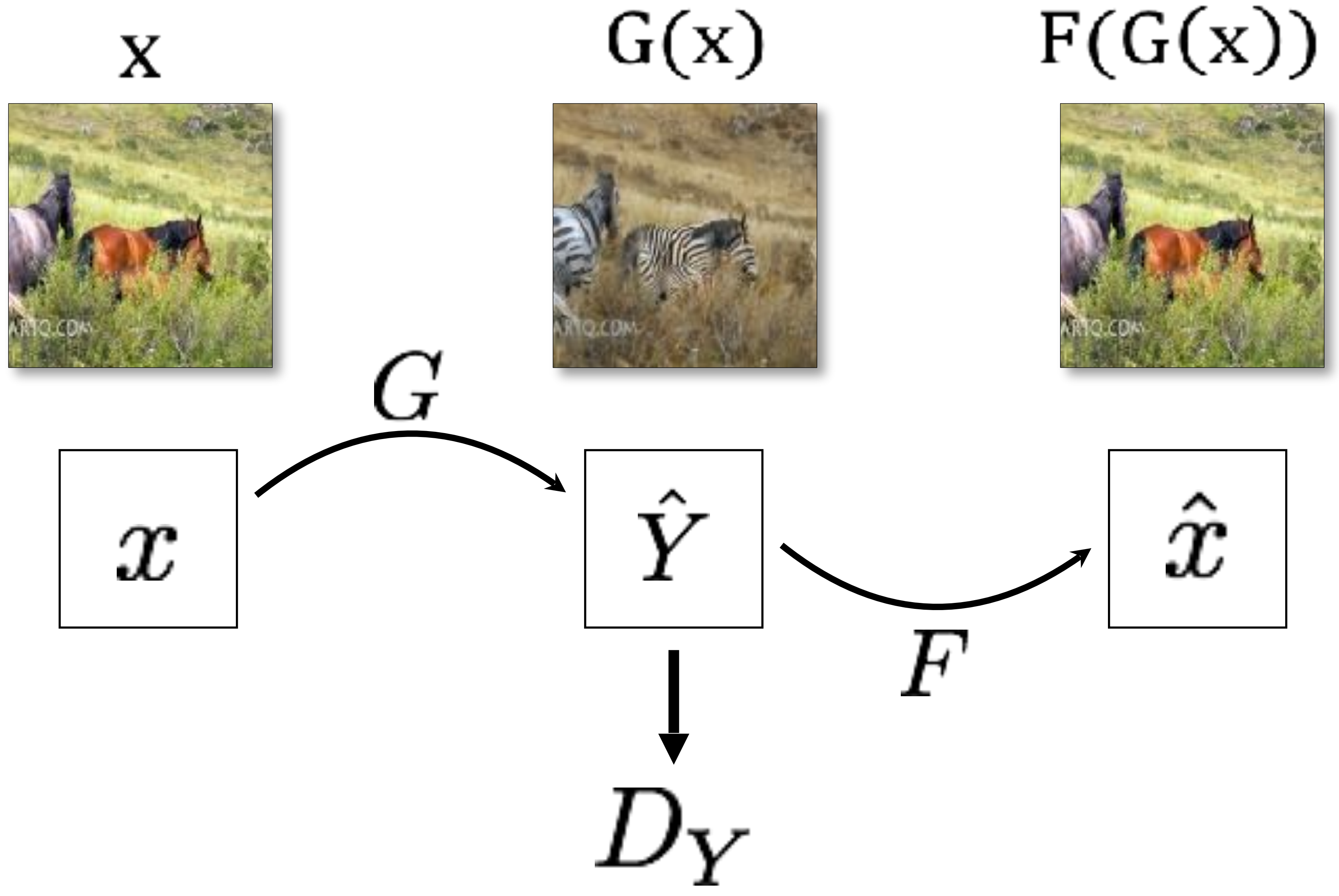


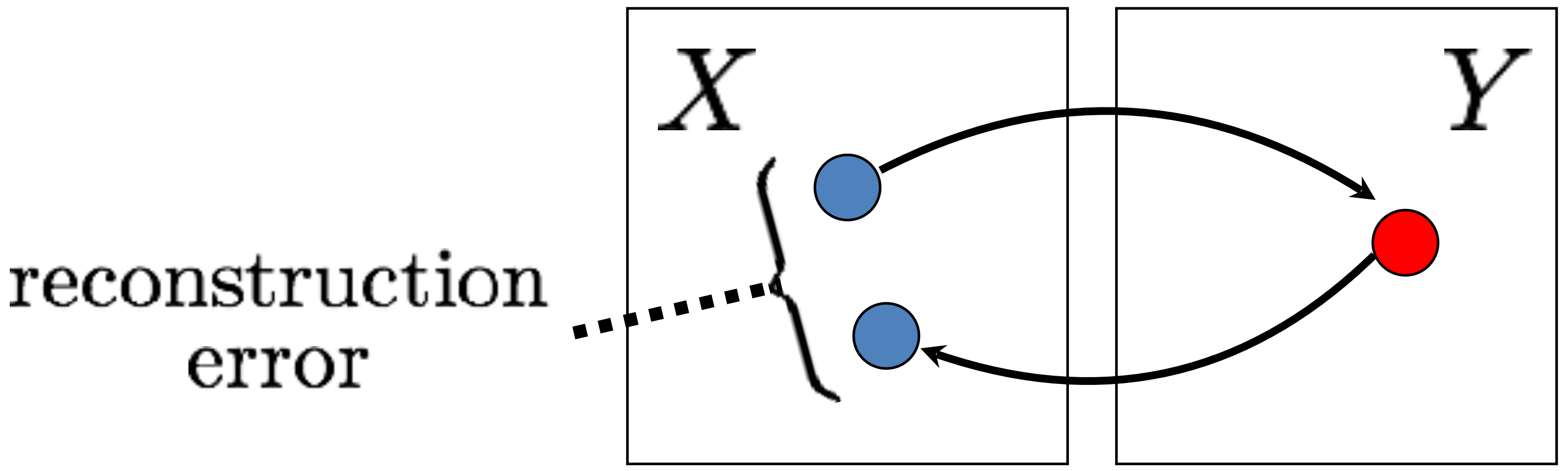
[Zhu*, Park* et al. 2017], [Yi et al. 2017], [Kim et al. 2017]

CycleGAN, or there and back aGAN



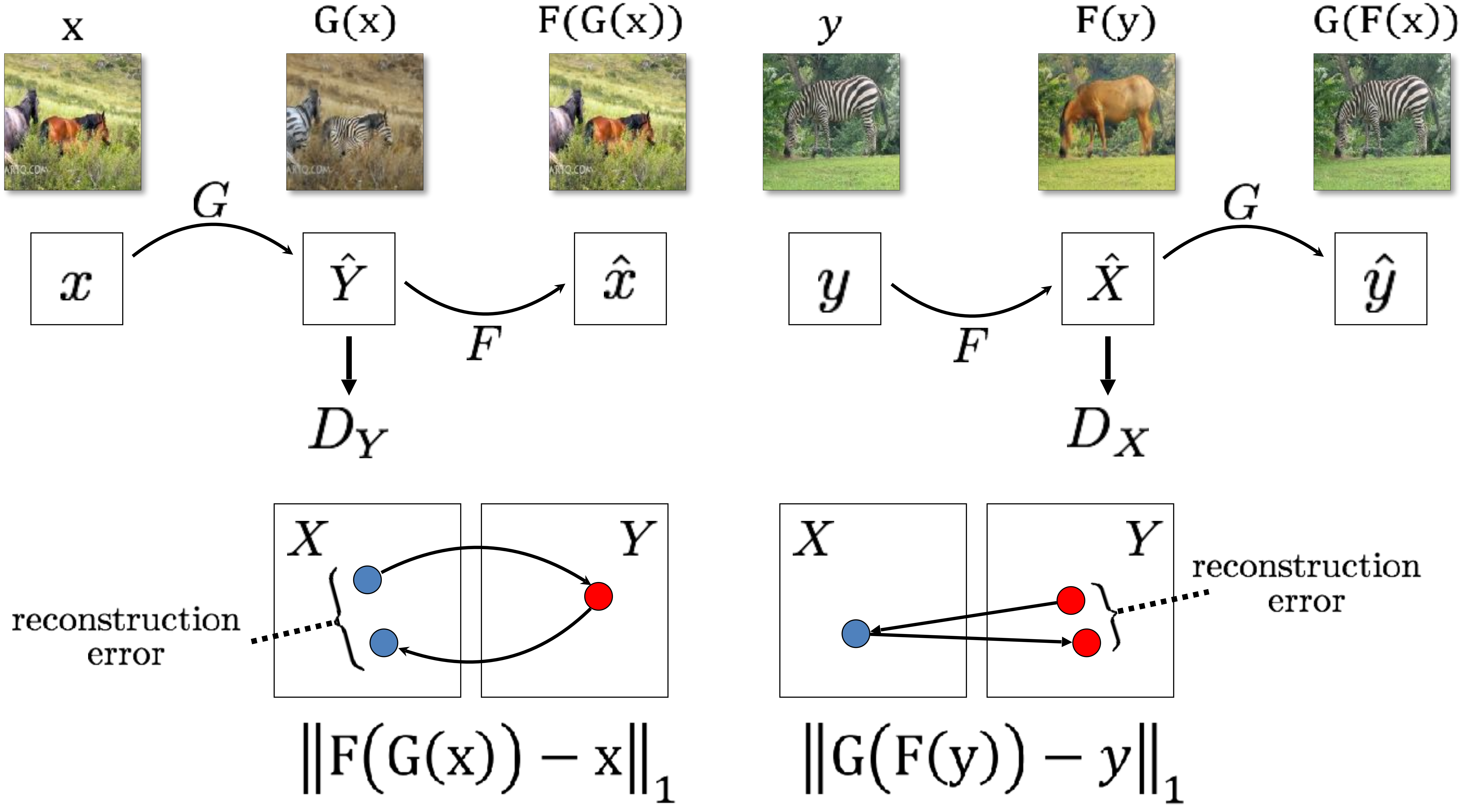
Cycle Consistency Loss



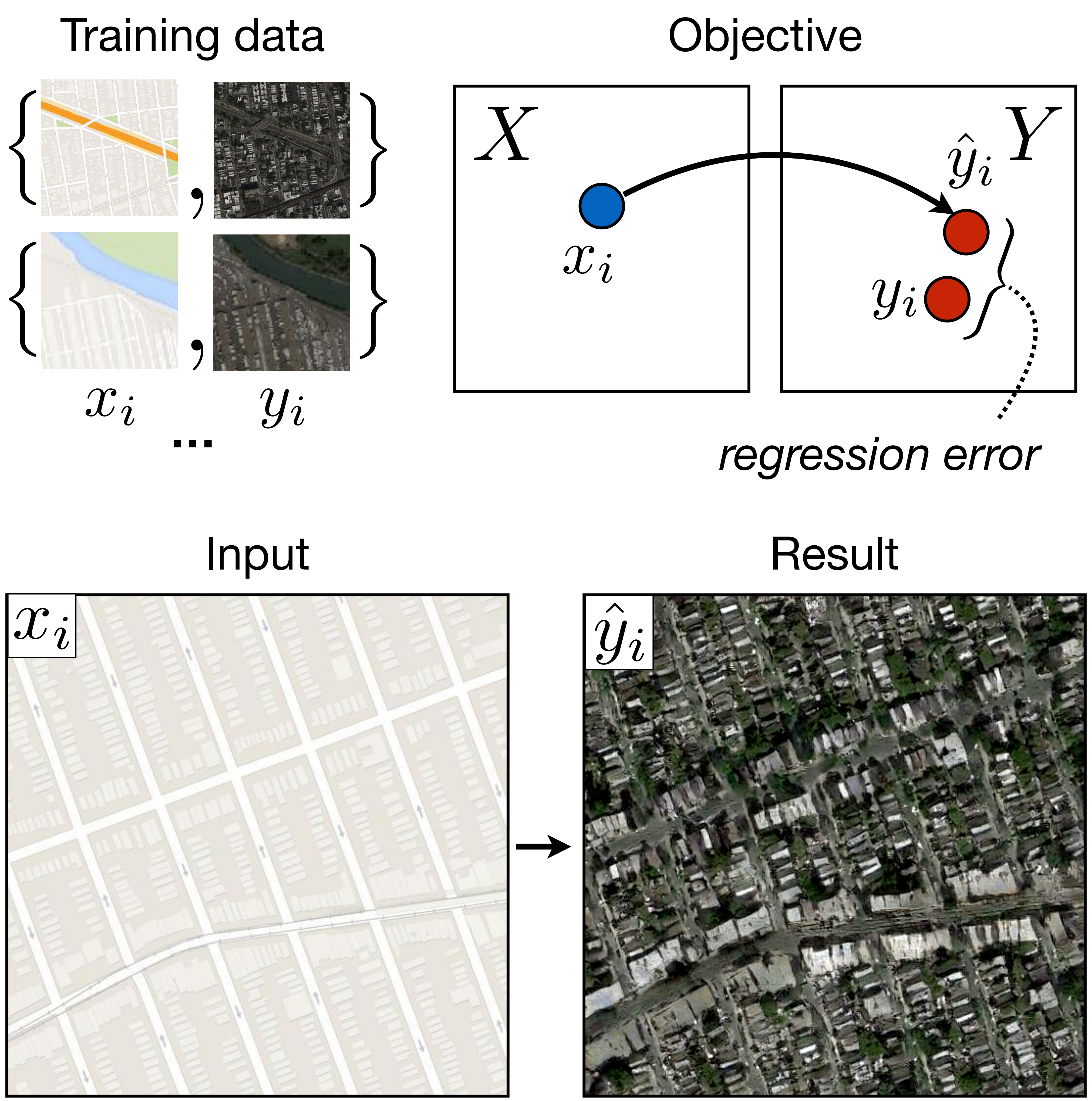


$$\|F(G(x)) - x\|_1$$

Cycle Consistency Loss

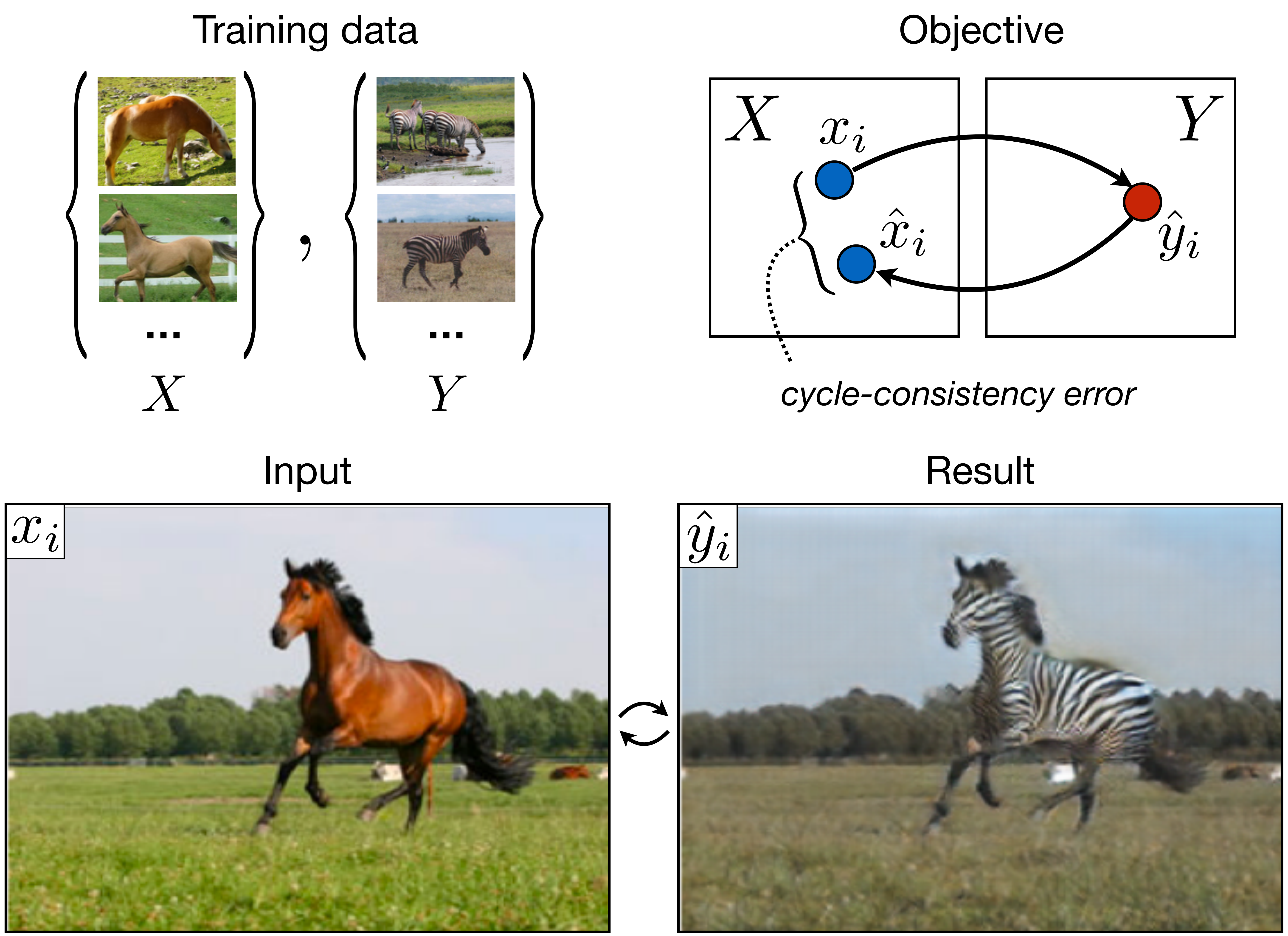


Paired translation

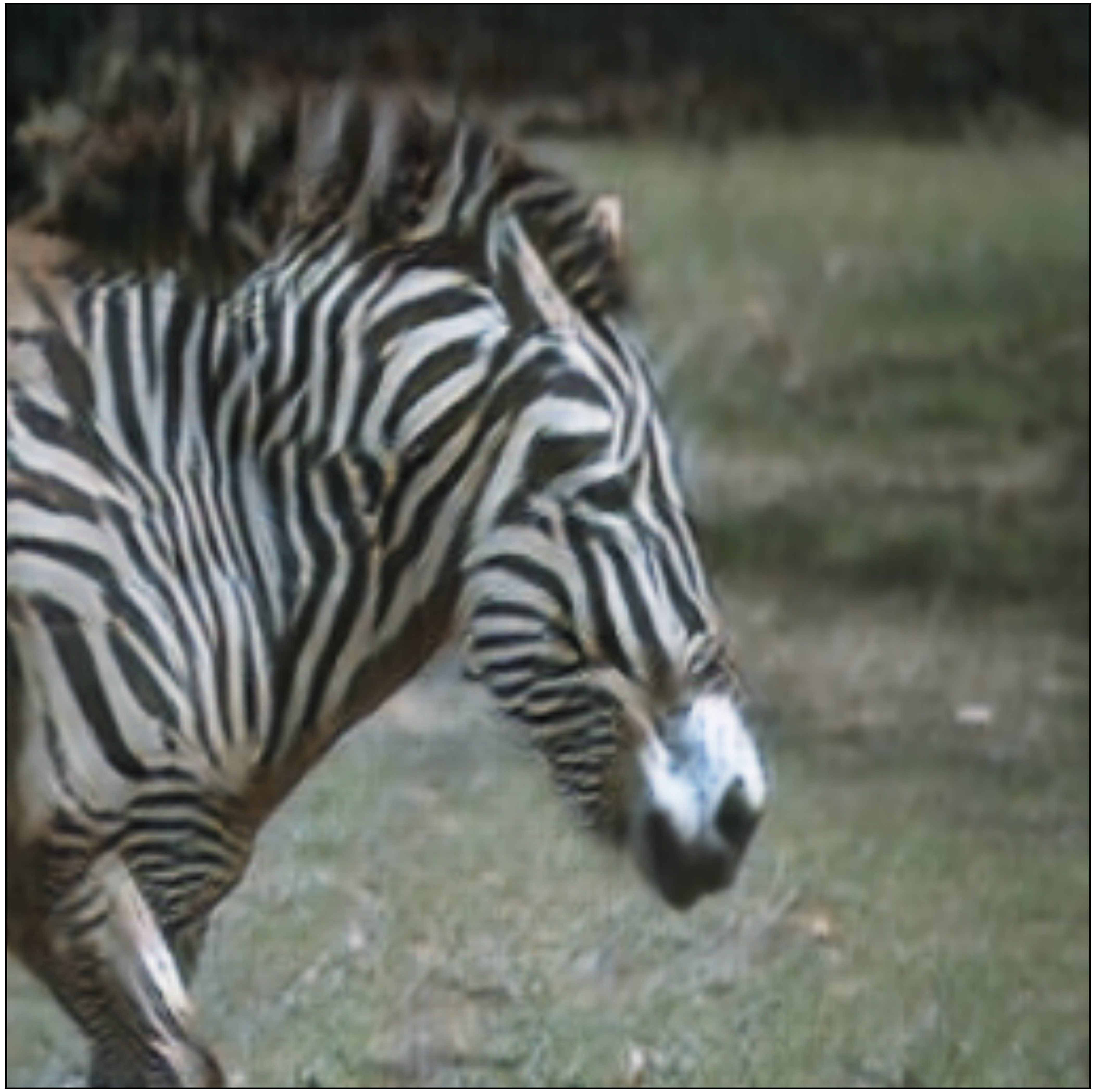


[“pix2pix”, Isola, Zhu, Zhou, Efros, 2017]

Unpaired translation



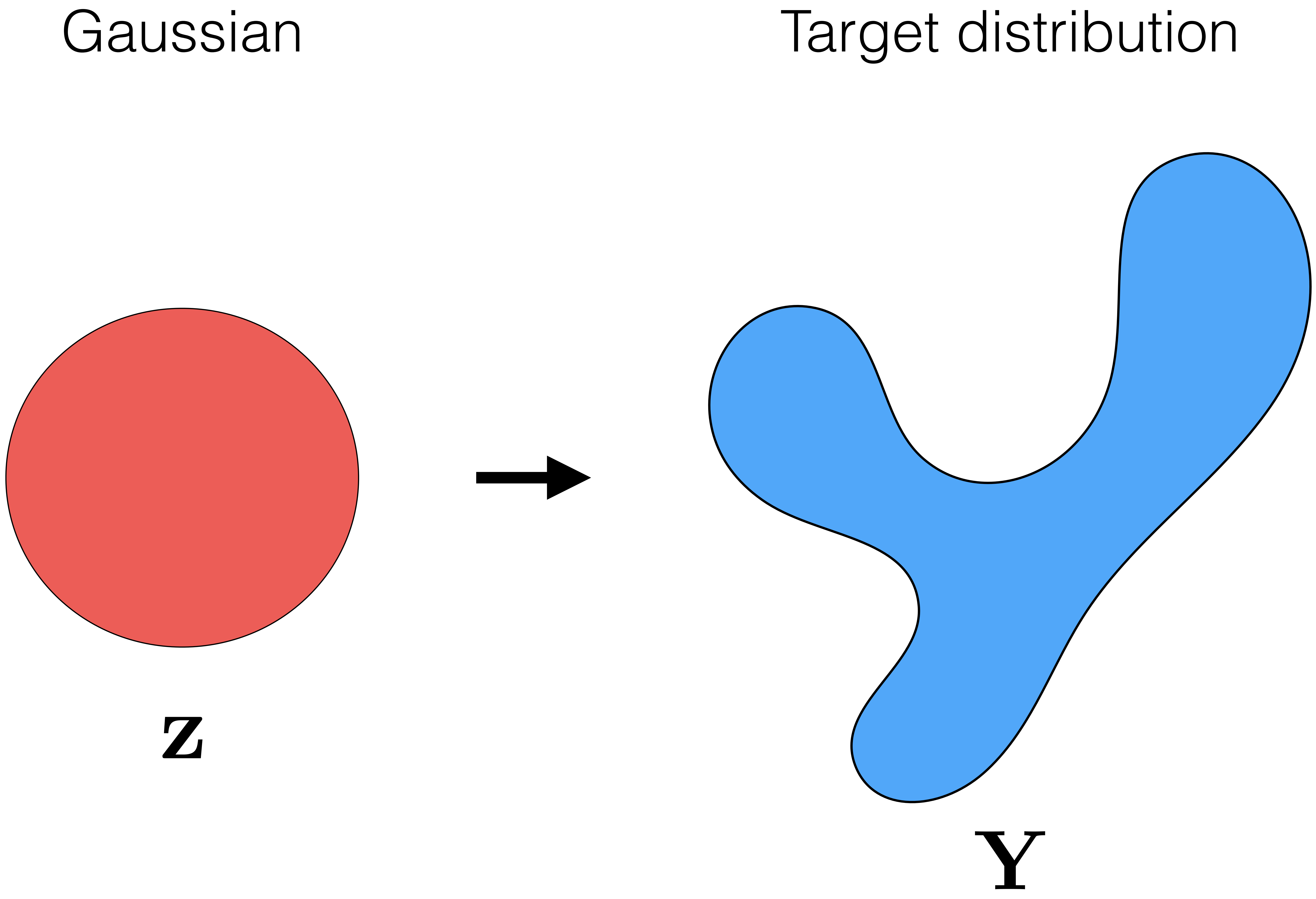
[“CycleGAN”, Zhu*, Park*, Isola, Efros, 2017]



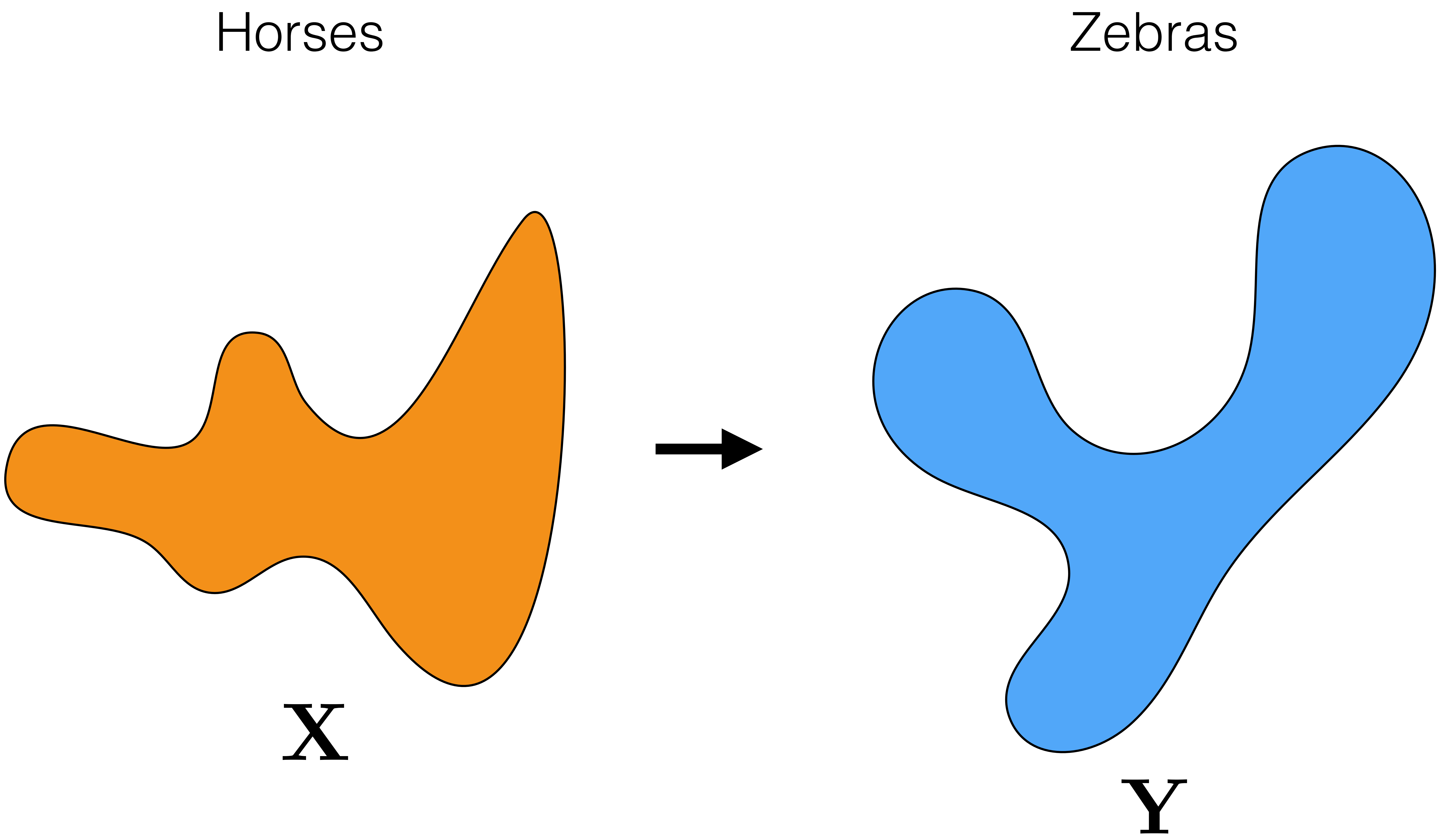


Input
Monet
Van Gogh
Cezanne
Ukiyo-e

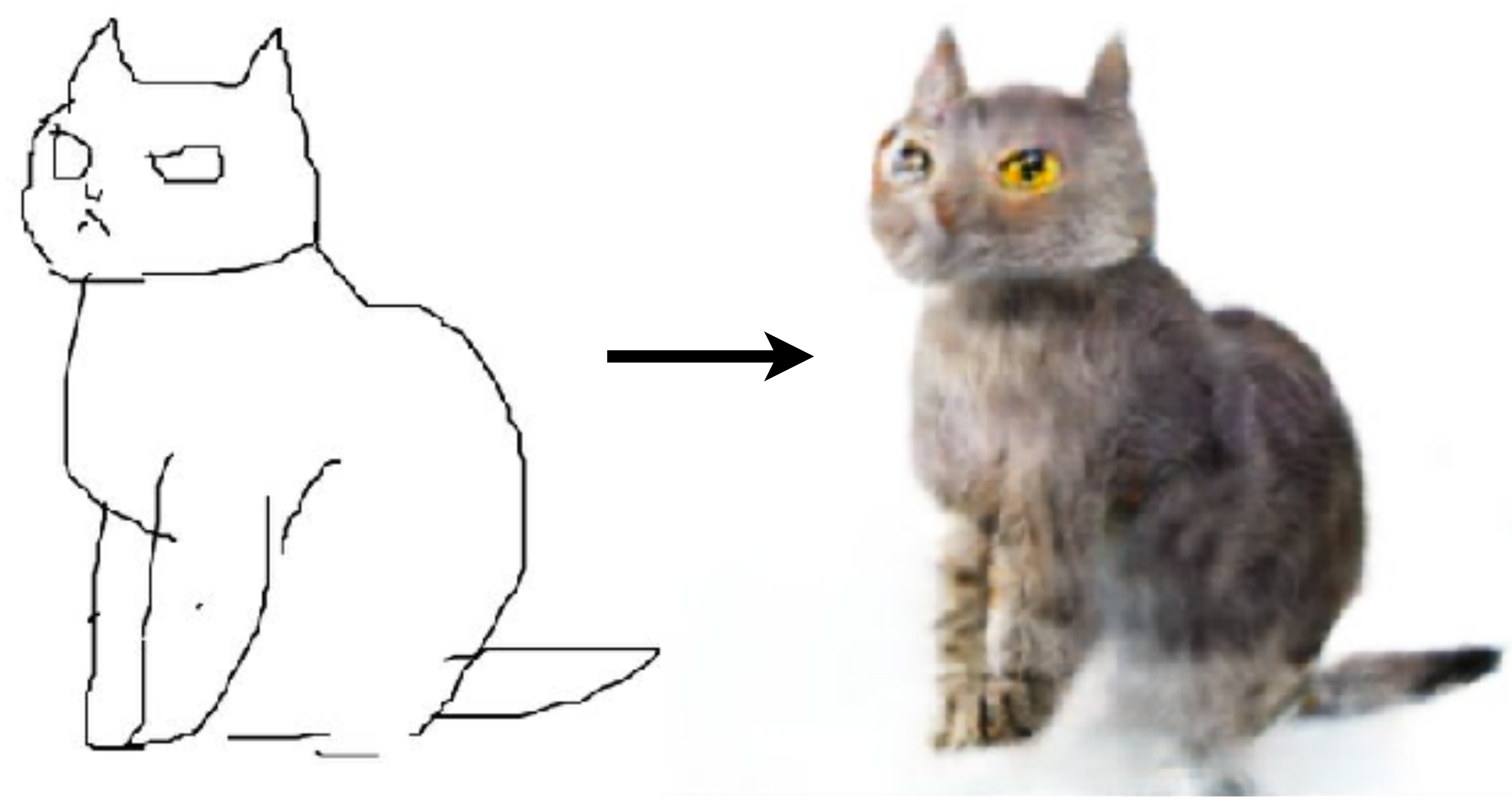
GANs

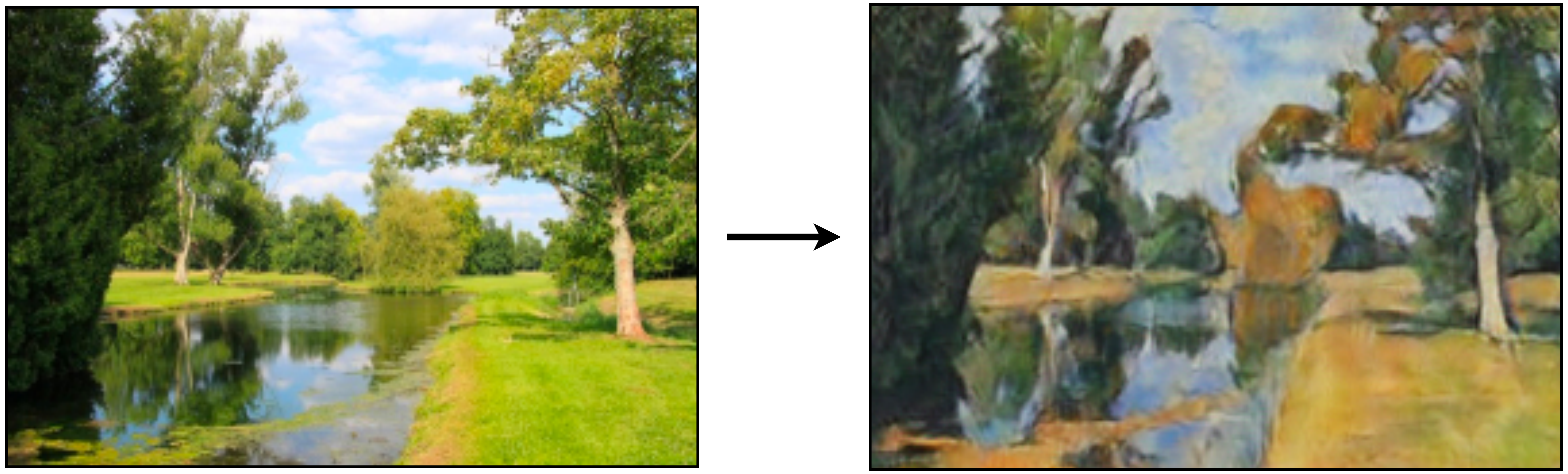


CycleGAN



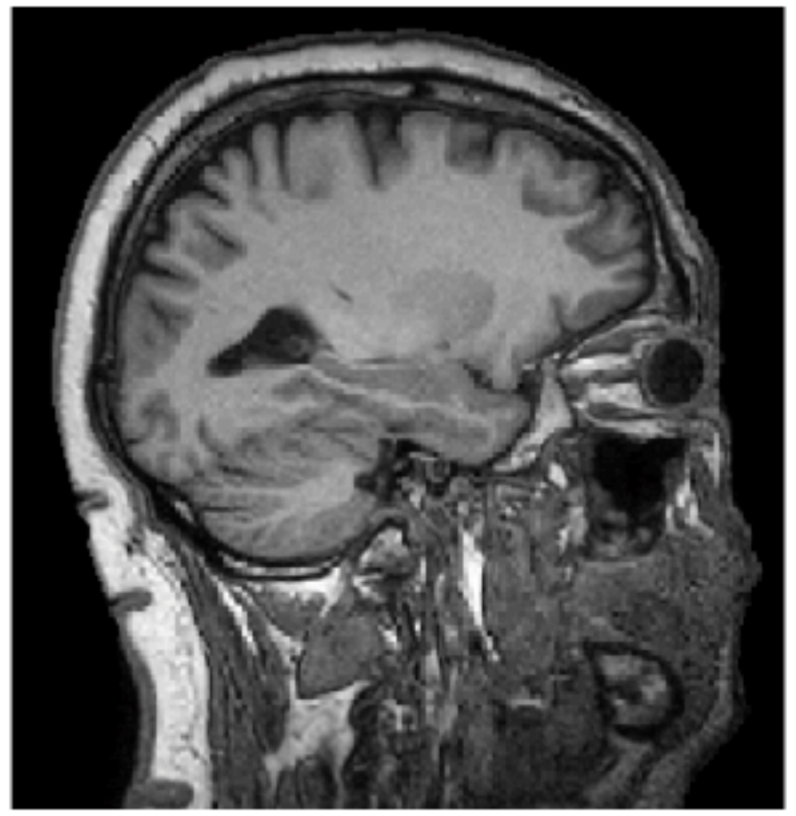
What would it look like if…?





What would it look like if…?

MRI



CT

[Wolterink et al, 2017]

Sim

“Real”

[Hoffman et al, 2018]