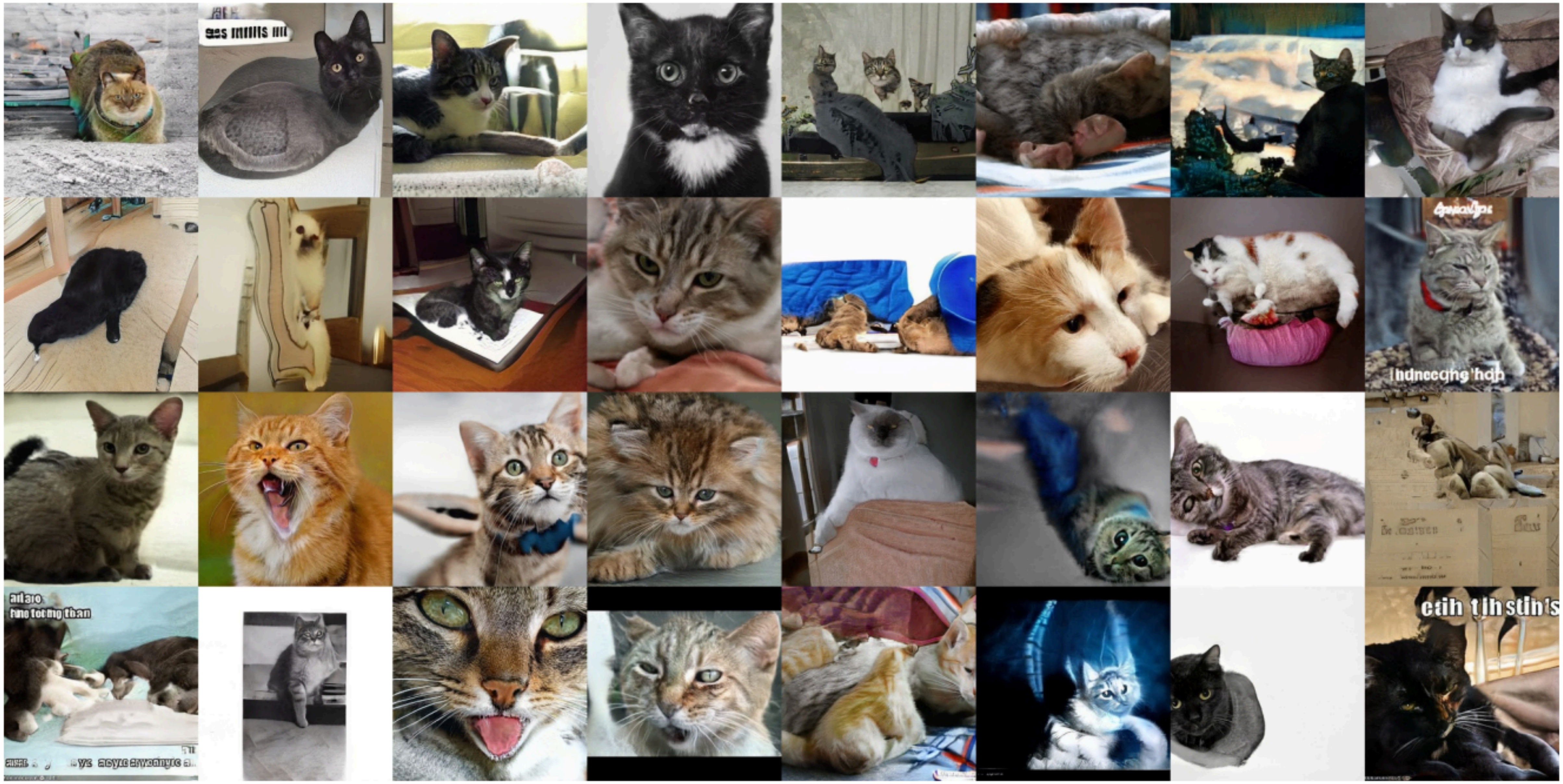


Perceptual Loss, GANs (part I)

Jun-Yan Zhu

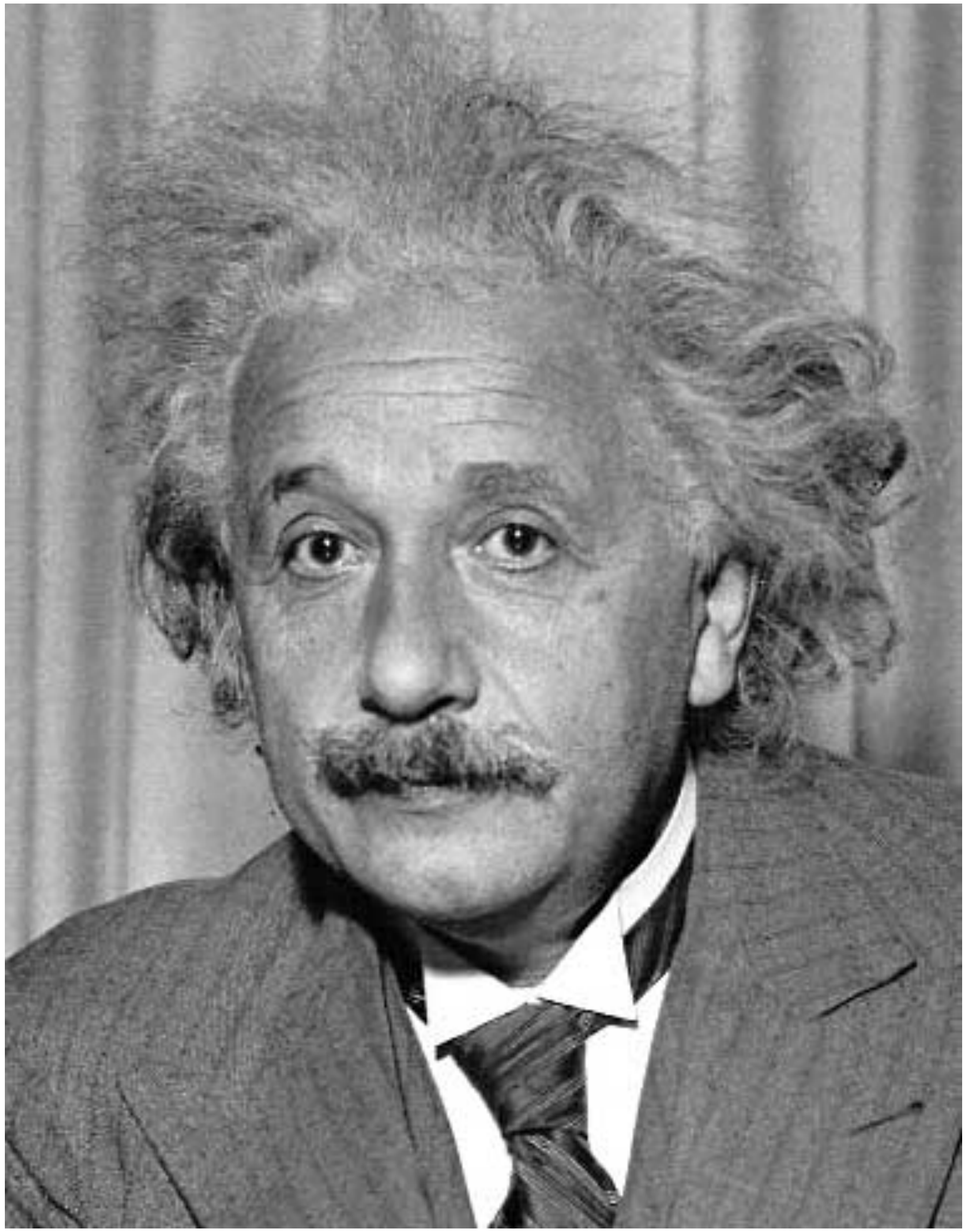
16-726 Learning-based Image Synthesis, Spring 2021

many slides from Phillip Isola, Richard Zhang, Alyosha Efros

HW1 (hints)

Template matching

- Goal: find  in image
- Main challenge: What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation



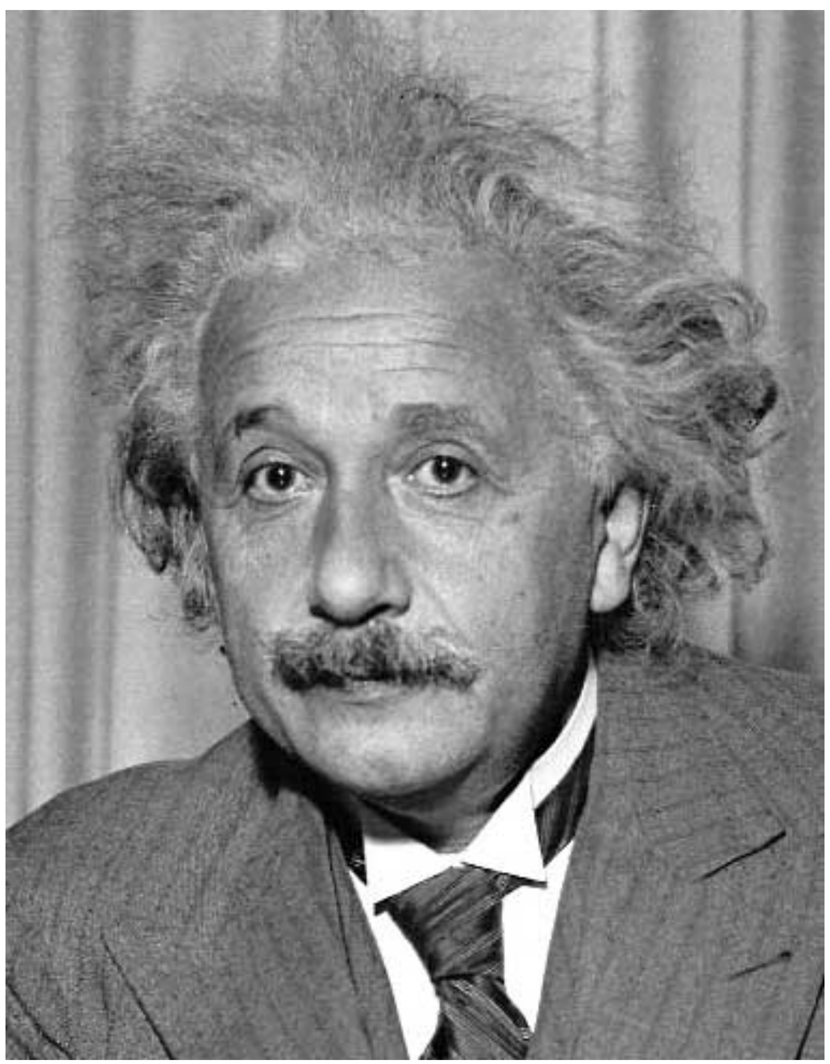
Side by Derek Hoiem

Matching with filters

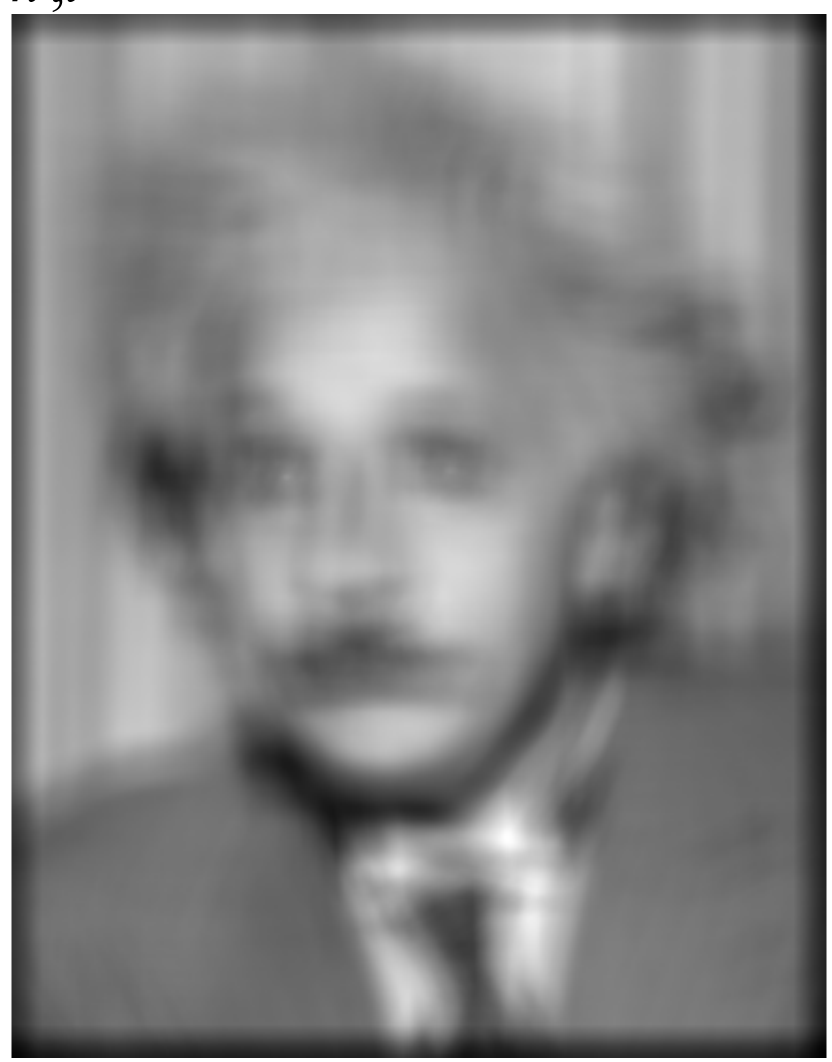
- Goal: find  in image
- Method 0: filter the image with eye patch

$$h[m,n]=\sum_{k,l} g[k,l]\, f[m+k,n+l]$$

f = image
g = filter



Input



Filtered Image

What went wrong?

Side by Derek Hoiem

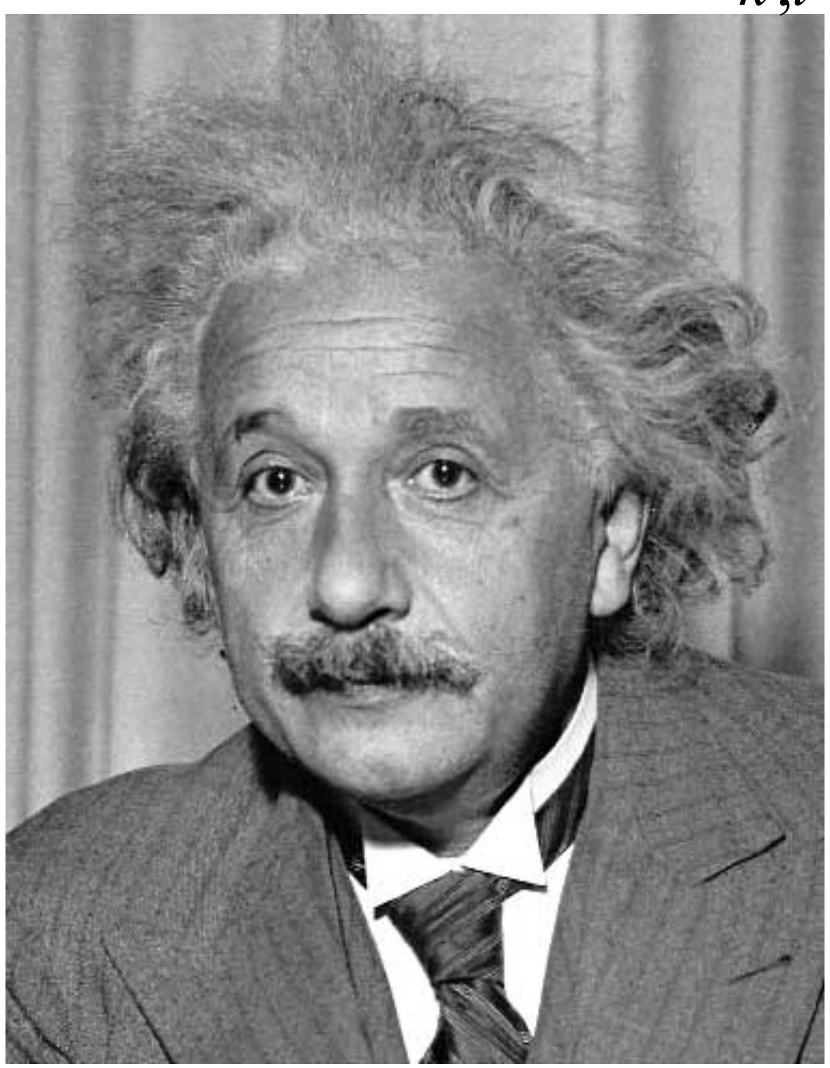
Matching with filters

- Goal: find  in image
- Method 1: filter the image with zero-mean eye

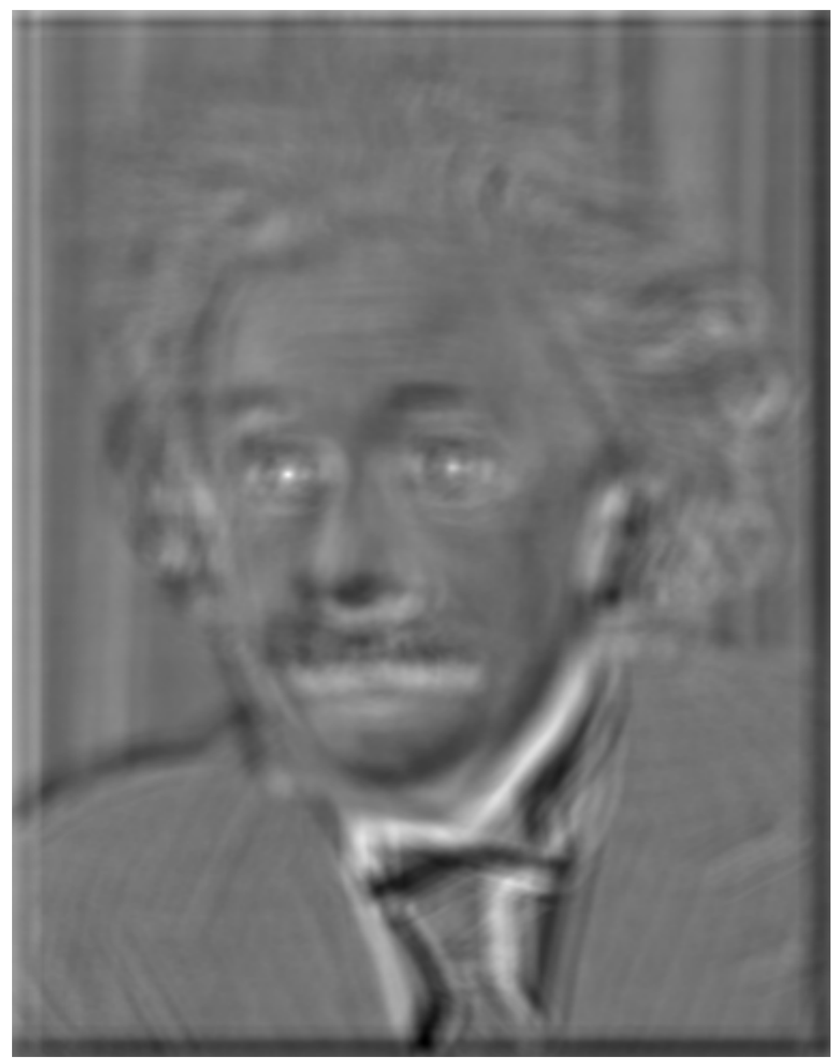
$$h[m,n] = \sum_{k,l} (f[k,l] - \bar{f})\,(g[m+k, n+l])$$

mean of f

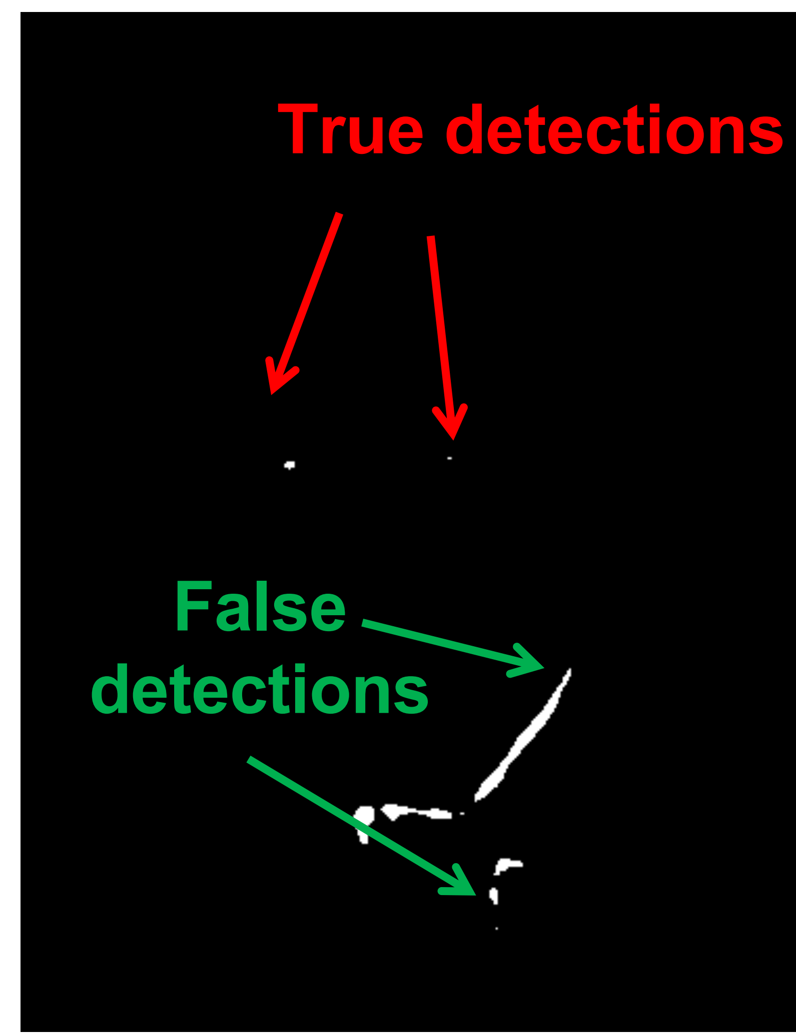
f = image
g = filter



Input



Filtered Image (scaled)



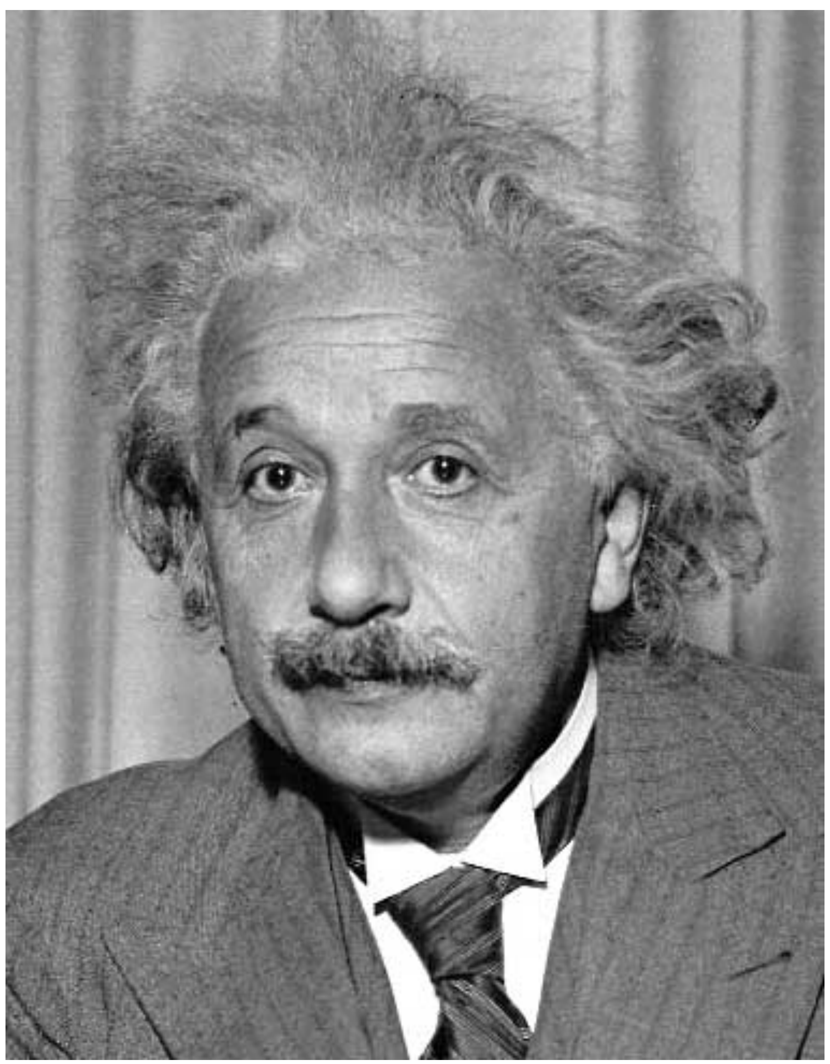
Thresholded Image

Matching with filters

- Goal: find  in image
- Method 2: SSD (Sum Square Difference)

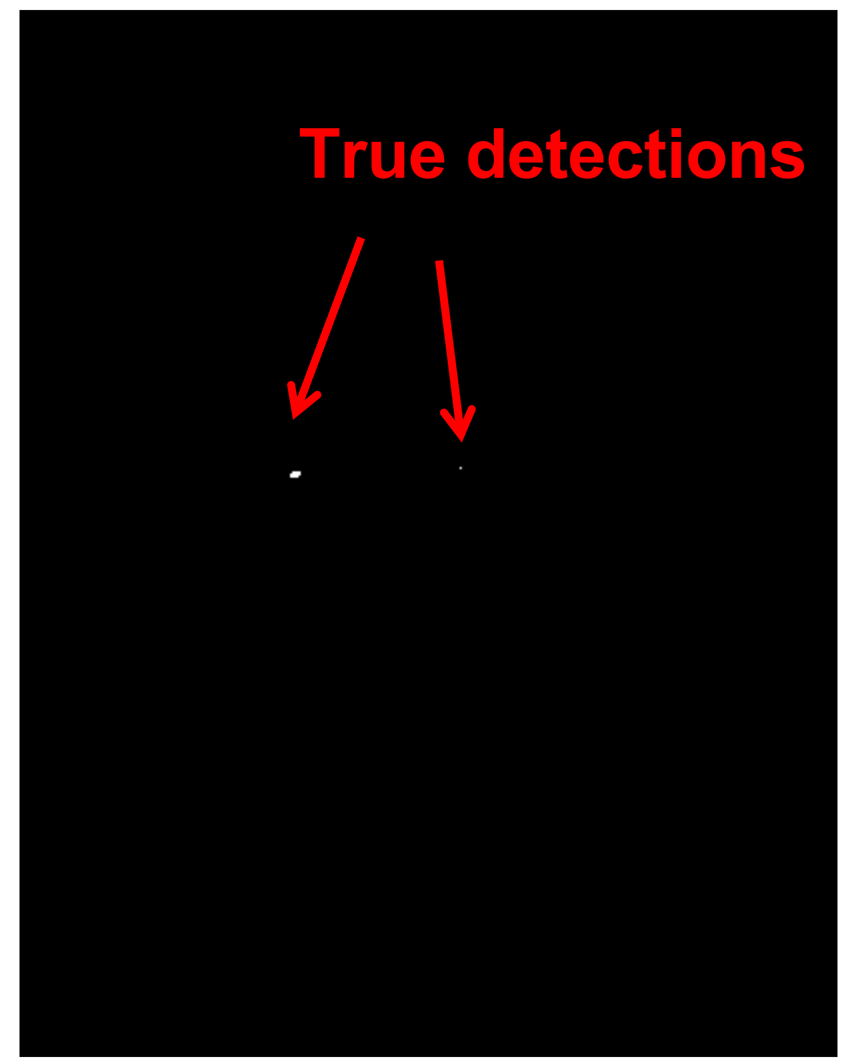
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$

f = image
g = filter



Input

1- sqrt(SSD)



Thresholded Image

Matching with filters

$$h[m,n]=\sum_{k,l}(g[k,l]-f[m+k,n+l])^2$$

f = image
g = filter

- Can SSD be implemented with linear filters?

Side by Derek Hoiem

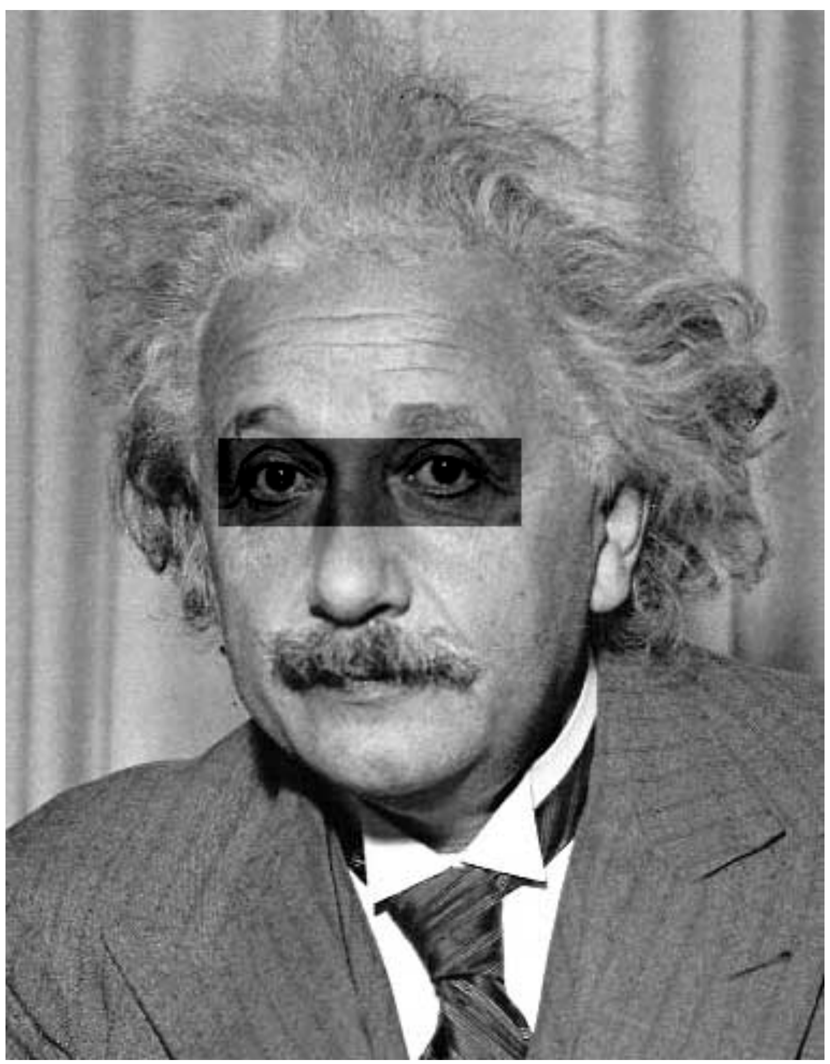
Matching with filters

- Goal: find  in image
- Method 2: SSD (Sum Square Difference)

What's the potential downside of SSD?

$$h[m,n]=\sum_{k,l}(g[k,l]-f[m+k,n+l])^2$$

f = image
g = filter



Input

1- sqrt(SSD)

Side by Derek Hoiem

Matching with filters

- Goal: find  in image
- Method 2: Normalized Cross-Correlation

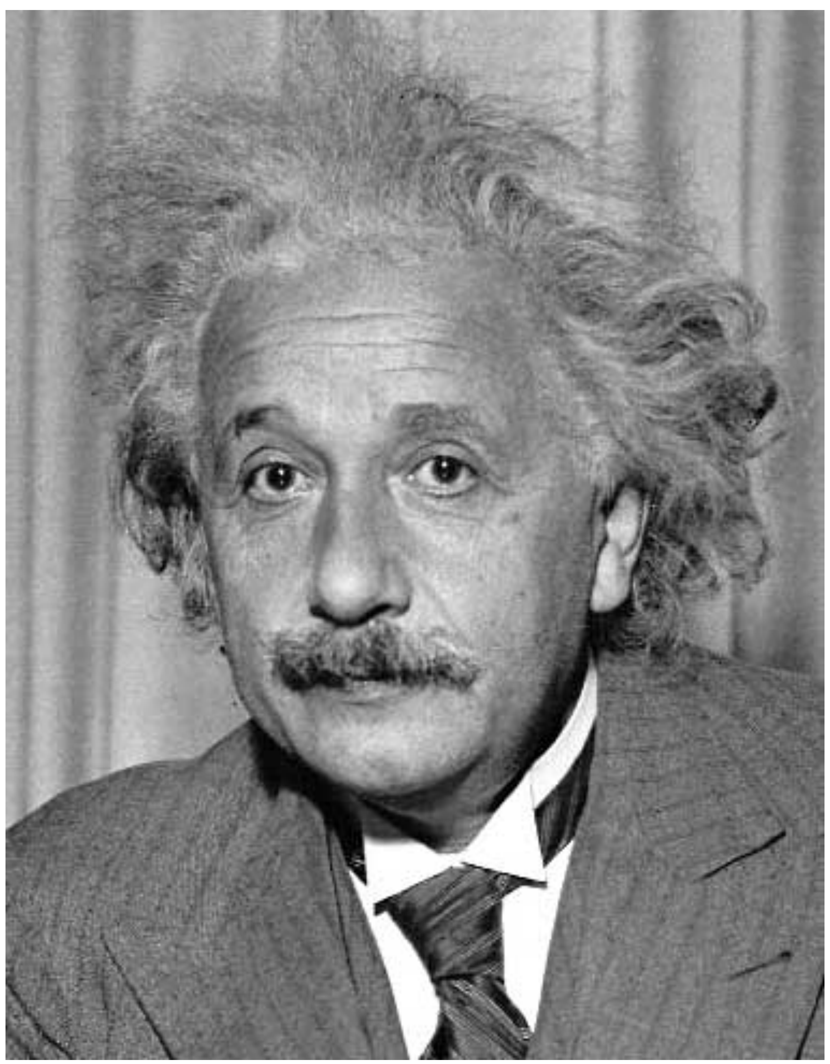
f = image
g = filter

$$h[m,n]=\frac{\sum_{k,l}(g[k,l]-\overset{\text{mean template}}{\overline{g}})(f[m+k,n+l]-\overset{\text{mean image patch}}{\bar{f}_{m,n}})}{\left(\sum_{k,l}(g[k,l]-\overline{g})^2\sum_{k,l}(f[m+k,n+l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Side by Derek Hoiem

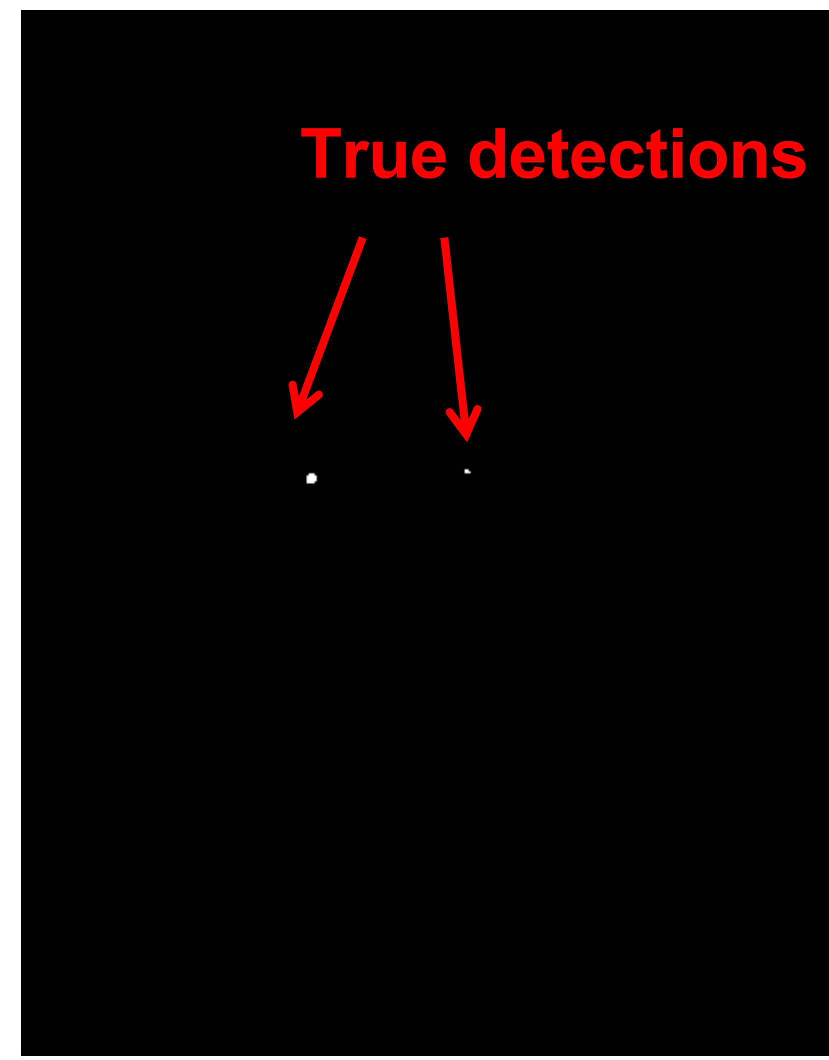
Matching with filters

- Goal: find  in image
- Method 2: Normalized Cross-Correlation



Input

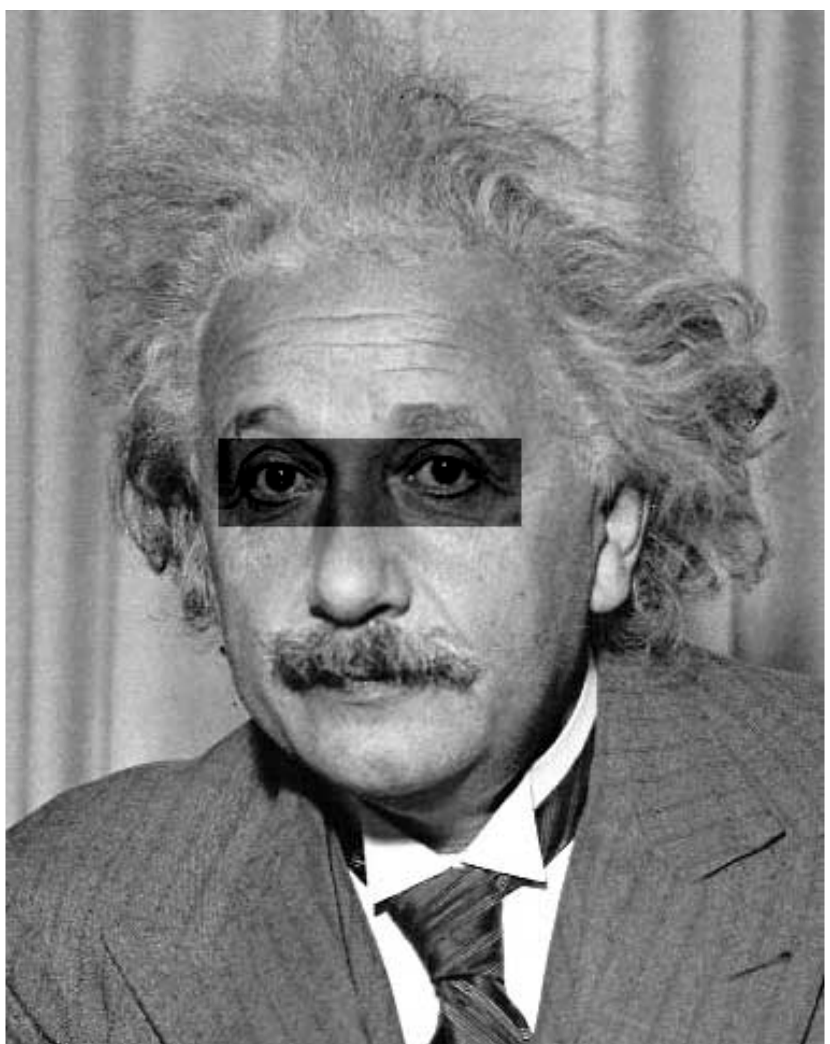
Normalized X-Correlation



Thresholded Image

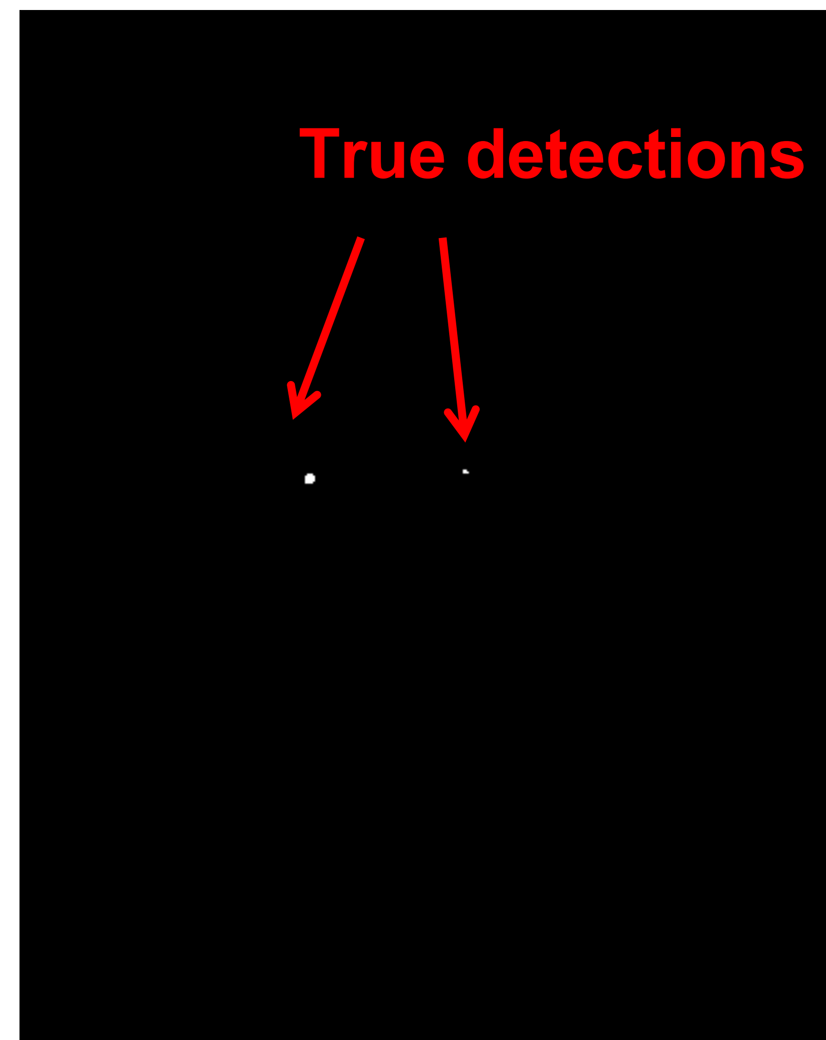
Matching with filters

- Goal: find  in image
- Method 2: Normalized Cross-Correlation



Input

Normalized X-Correlation



Thresholded Image

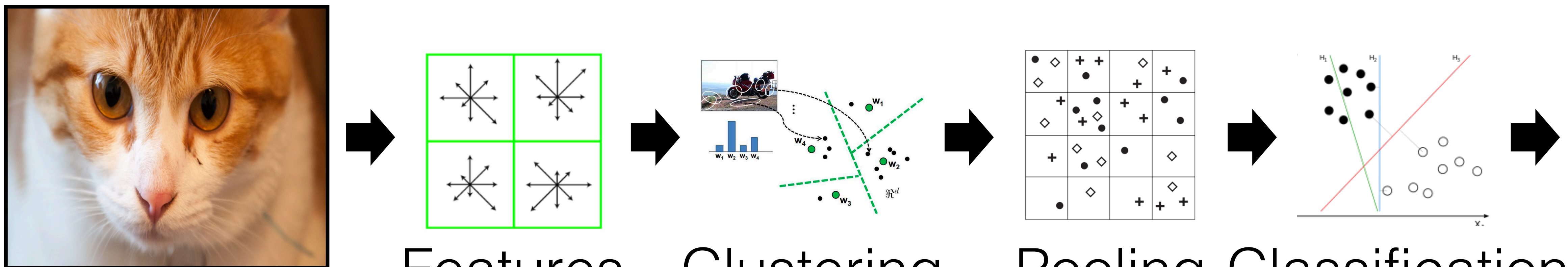
Q: What is the best method to use?

- Answer: Depends
- Zero-mean filter: fastest but not a great matcher
- SSD: next fastest, sensitive to overall intensity
- Normalized cross-correlation: slowest, invariant to local average intensity and contrast

Side by Derek Hoiem

Review
(CNN for Image Synthesis)

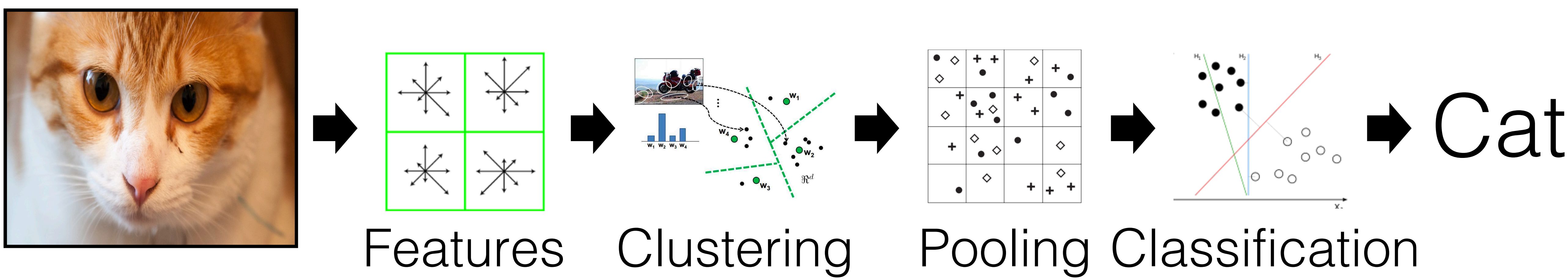
Computer Vision before 2012

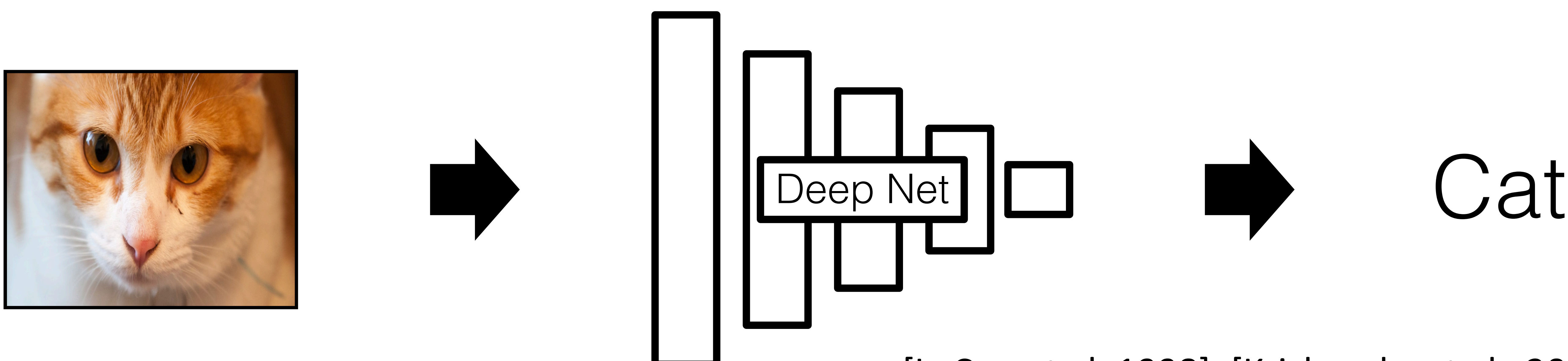


Cat

Features Clustering Pooling Classification

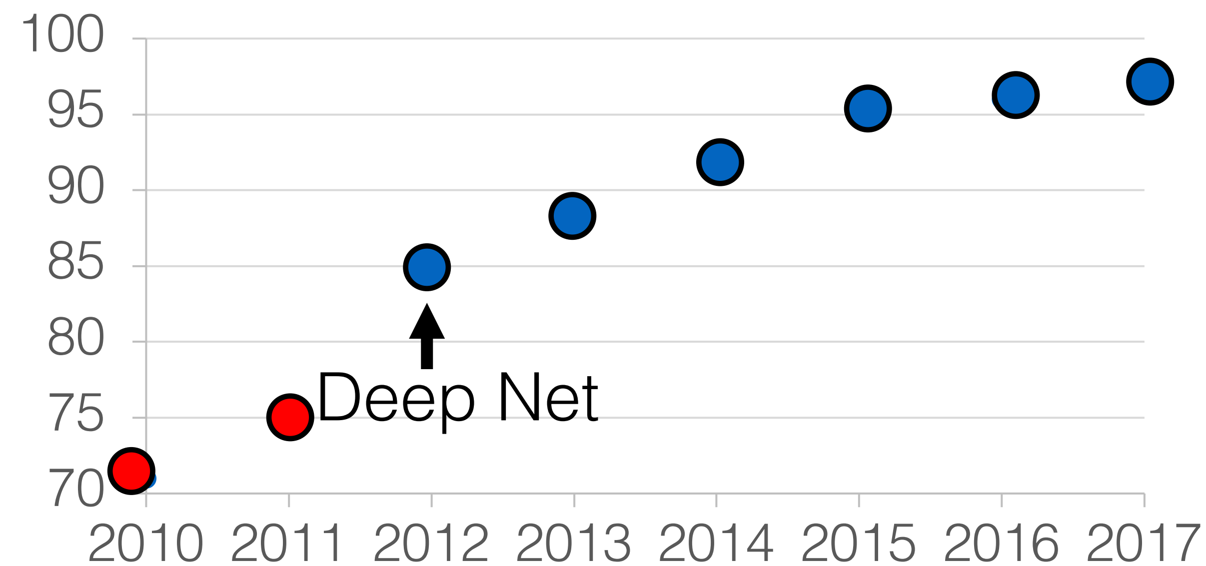
Computer Vision Now



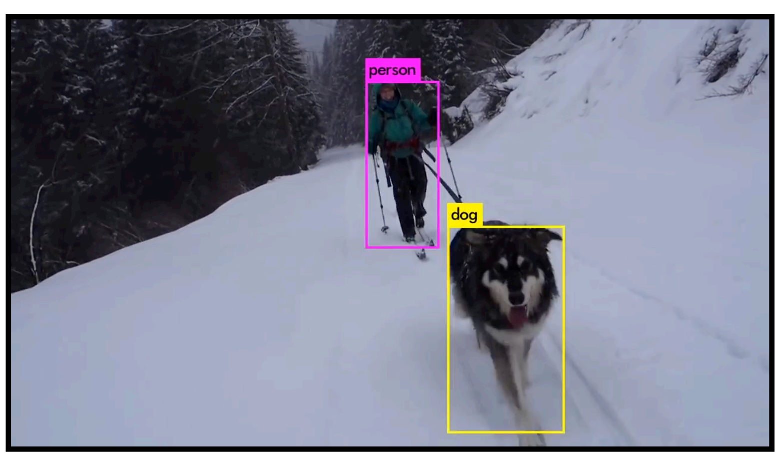


[LeCun et al, 1998], [Krizhevsky et al, 2012]

Deep Learning for Computer Vision

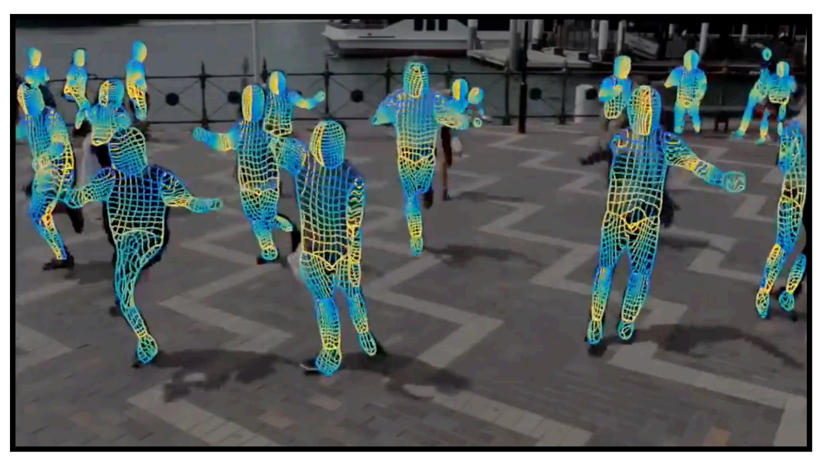


Top 5 **accuracy** on ImageNet benchmark



[Redmon et al., 2018]

Object detection



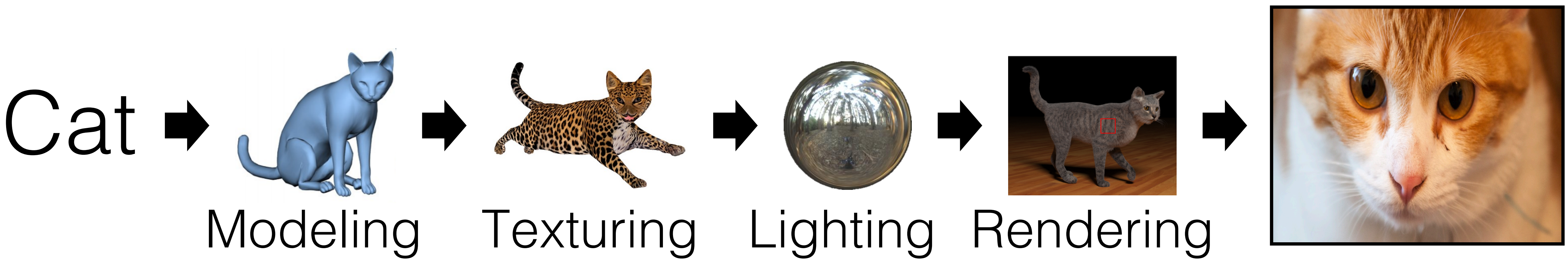
[Güler et al., 2018]

Human understanding

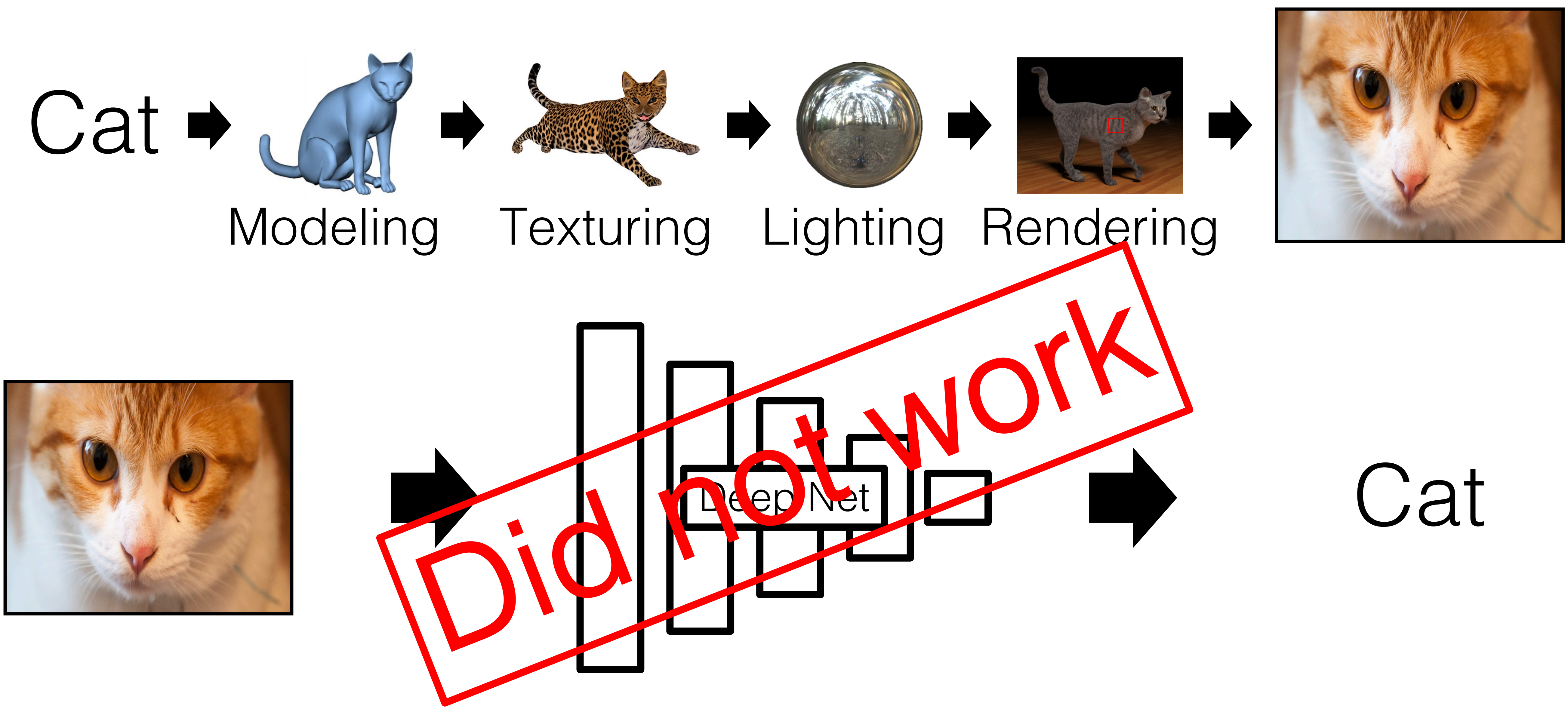
[Zhao et al., 2017]

Autonomous driving

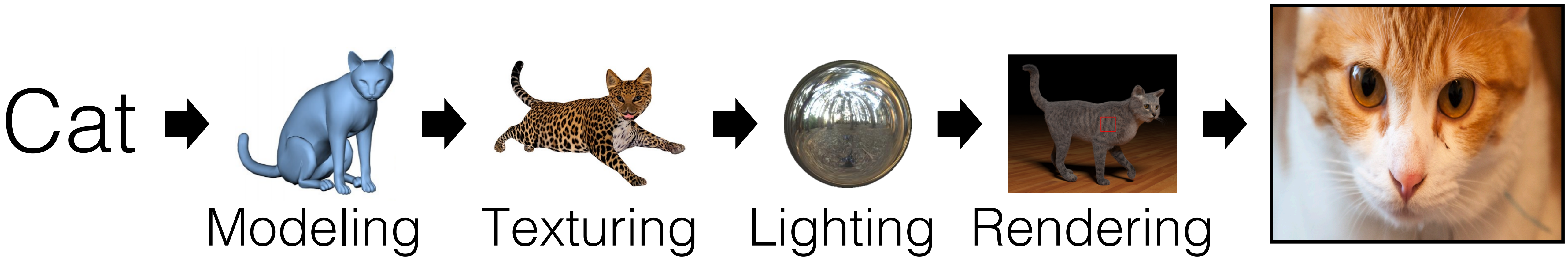
Can Deep Learning **Help** Graphics?

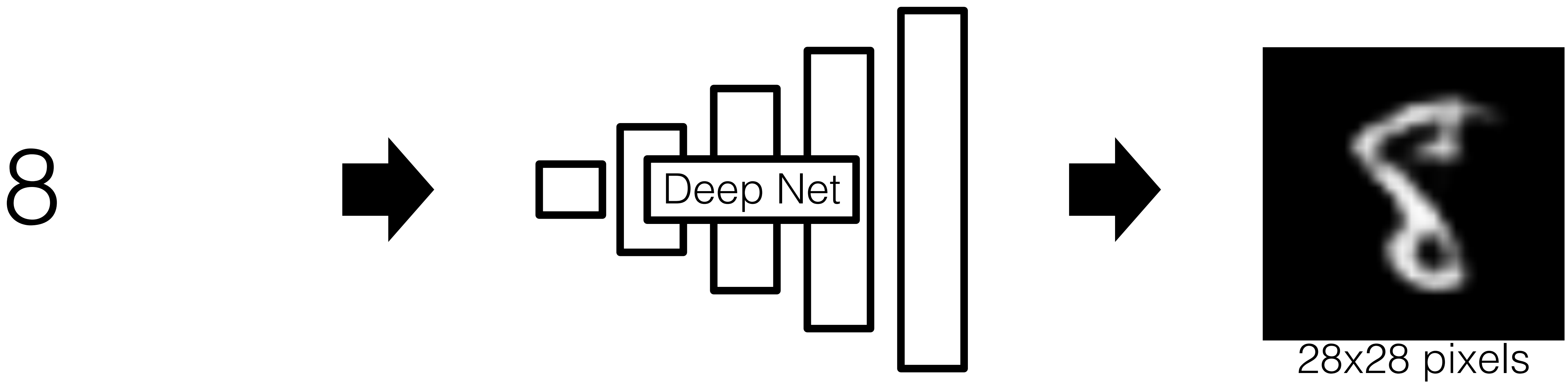


Can Deep Learning **Help** Graphics?



Generating images is hard!





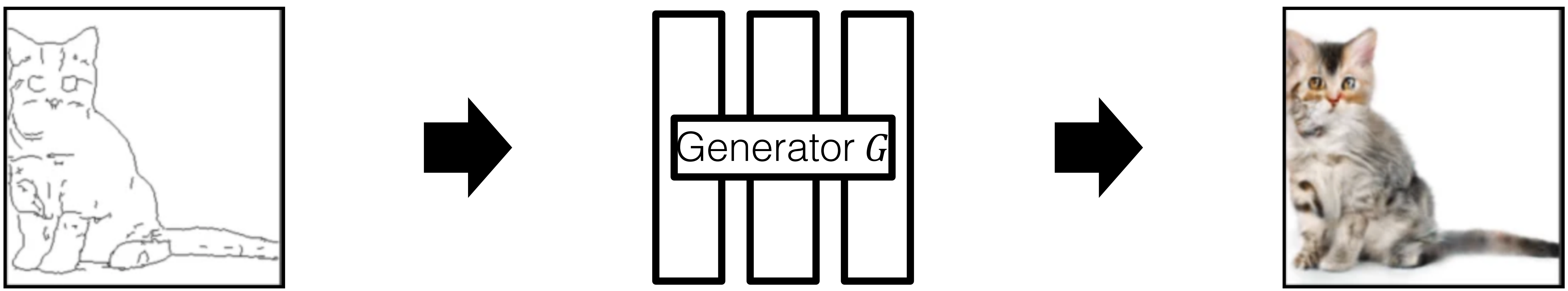
Simple L2 regression doesn't work ☹

Input

Output

Ground truth

Loss functions for Image Synthesis



Input $\mathbf{x}$ Learnable rendering Output Image $\mathbf{G}($

What is a good objective $\mathcal{L}$?

- Capture realism
- Task-agnostic
- Data-dependent

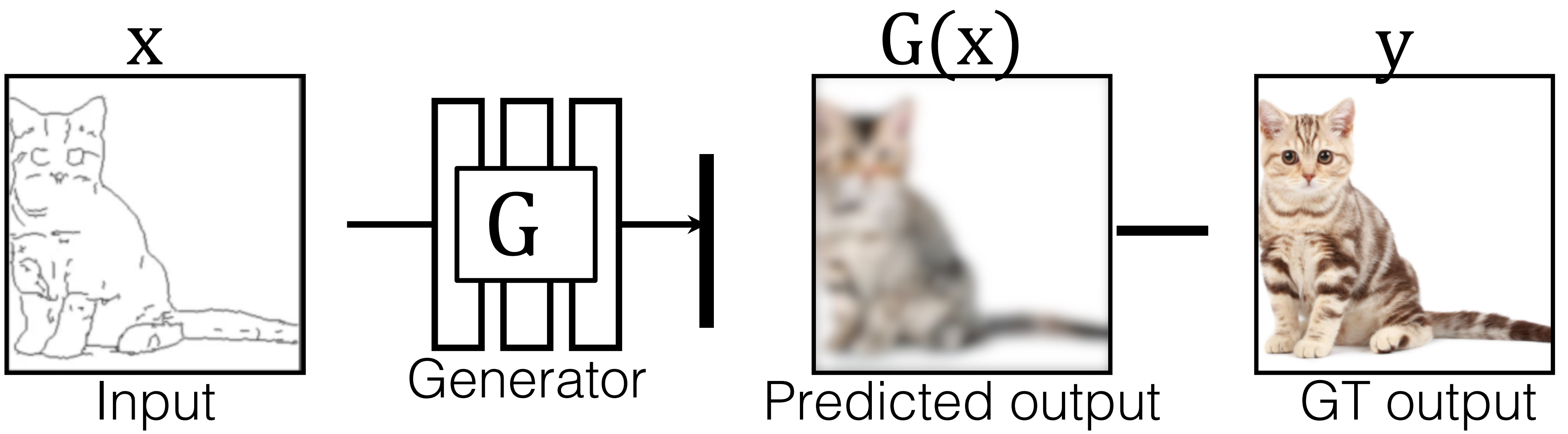
Problem Statement

Loss function

$$\arg\min_{G} \mathcal{L}(G(x), y)$$

Generator Input Output image

Designing Loss Functions

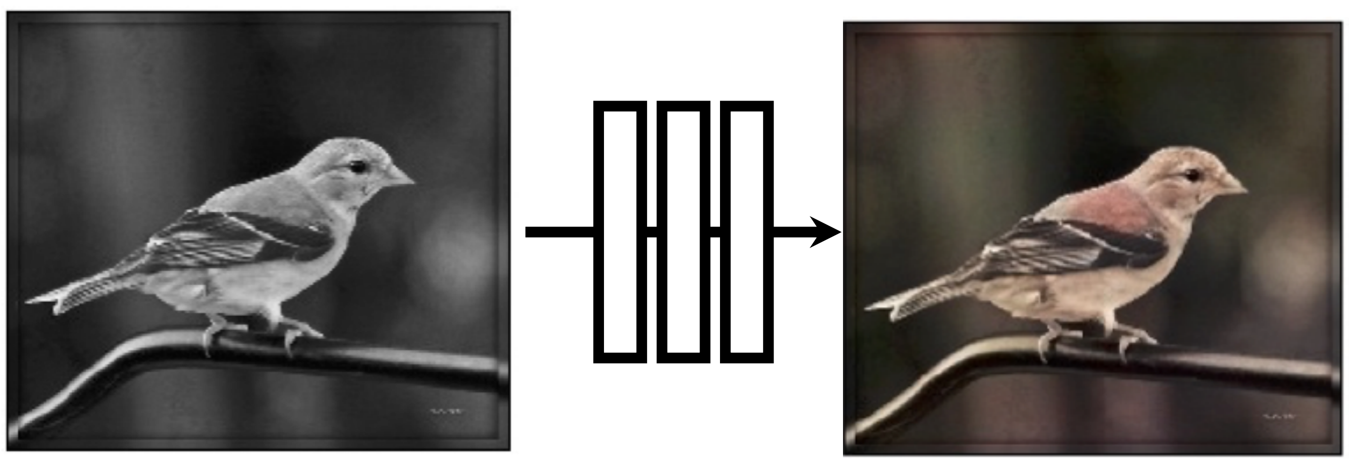


L2 regression

$$\arg\min_{G} \mathbb{E}_{(x,y)}[\,||G(x) - y||\,]$$

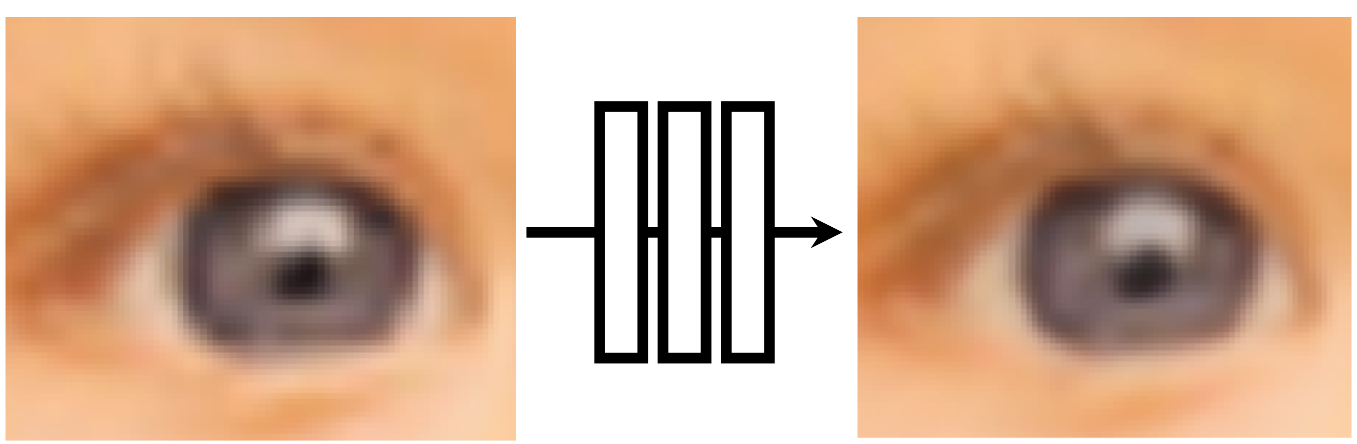
Designing Loss Functions

Image colorization



L2 regression

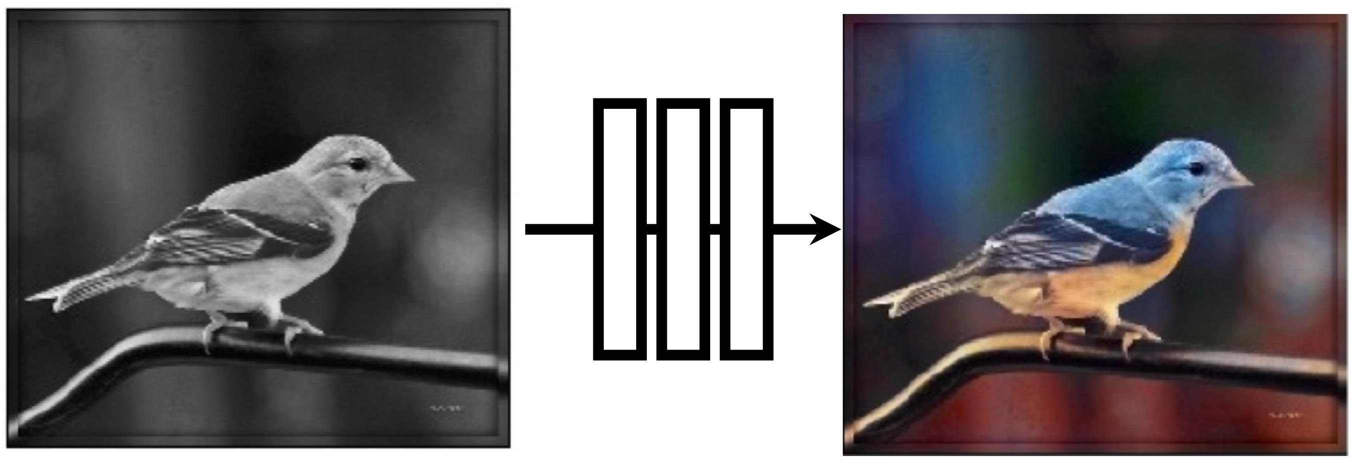
Super-resolution



L2 regression

Designing Loss Functions

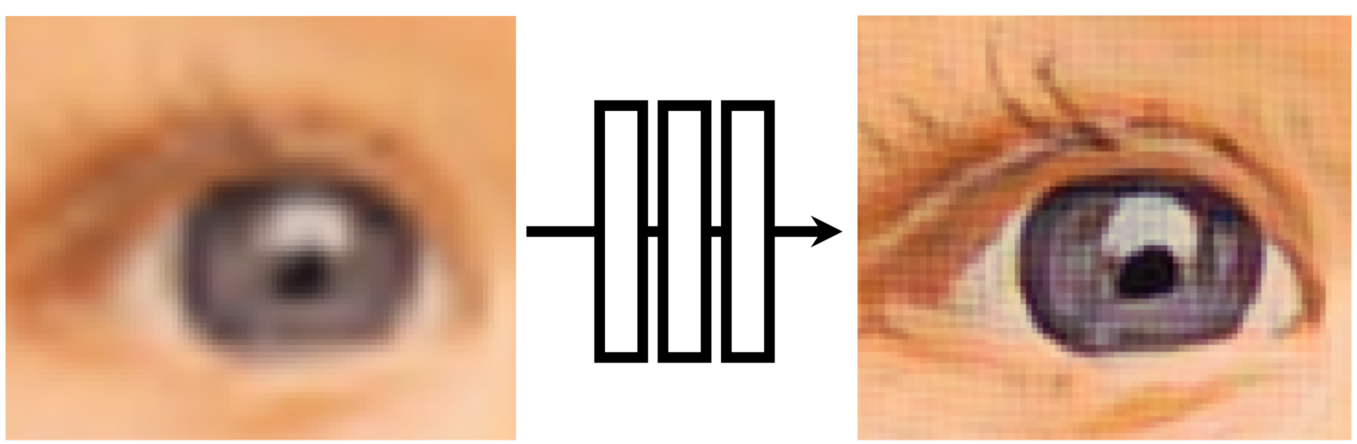
Image colorization



[Zhang et al. 2016]

Classification Loss:
Cross entropy objective,
with colorfulness term

Super-resolution



[Gatys et al., 2016], [Johnson et al. 2016]
[Dosovitskiy and Brox. 2016]

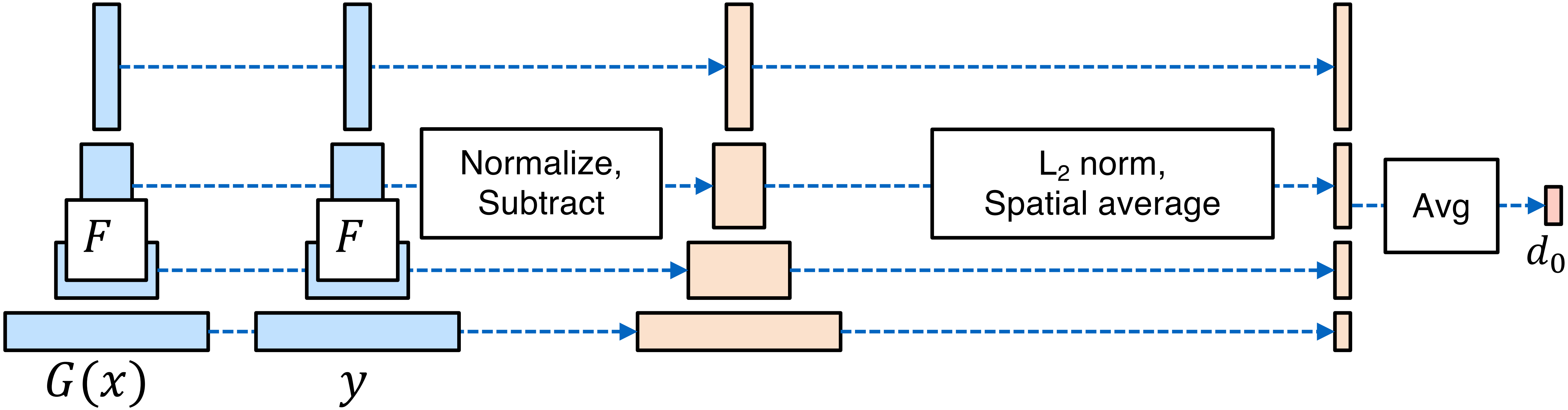
Feature/Perceptual loss
Deep feature covariance
matching objective

“Perceptual Loss”

Gatys et al. In CVPR, 2016.
Johnson et al. In ECCV, 2016.
Dosovitskiy and Brox. In NIPS, 2016.

Chen and Koltun. In ICCV, 2017.

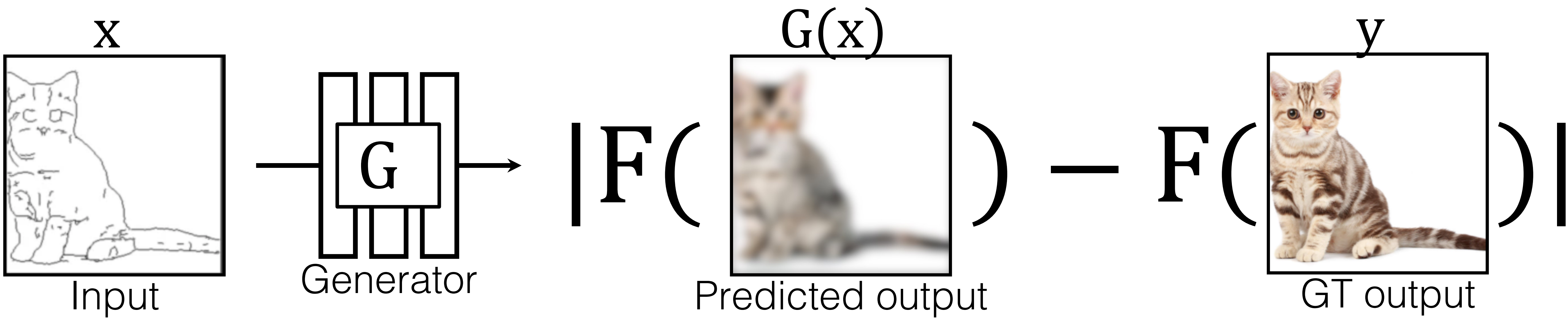
CNNs as a Perceptual Metric



(1) How well do “perceptual losses” describe perception?

c.f. Gatys et al. CVPR 2016. Johnson et al. ECCV 2016. Dosovitskiy and Brox. NIPS 2016.

CNNs as a Perceptual Metric



F is a deep network (e.g., ImageNet classifier)

Perceptual Loss

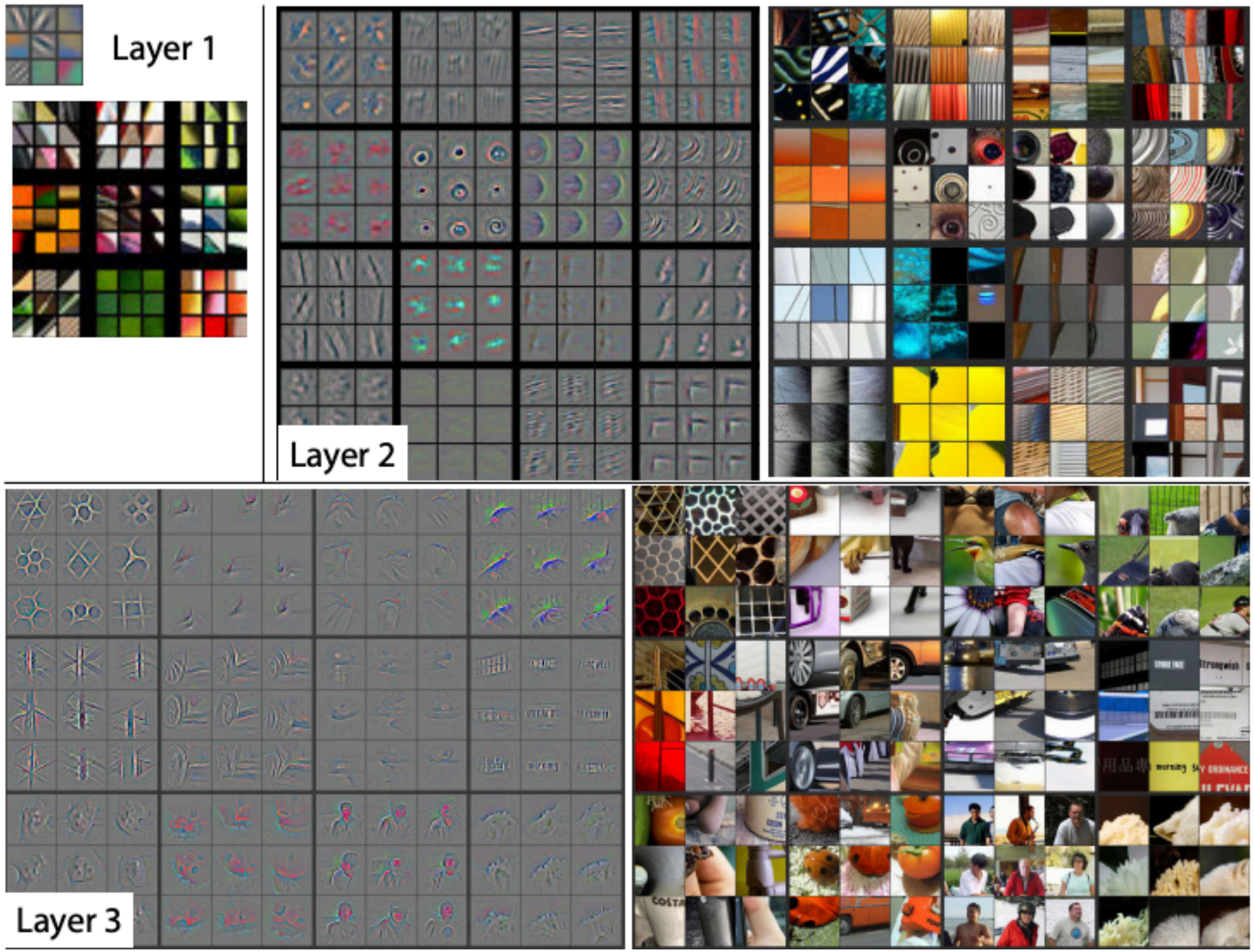
weight

(i)-th layer

$$\arg\min_{G} \mathbb{E}_{(x,y)} \sum_{i=1}^{N} \lambda_i \frac{1}{M_i} ||F^{(i)}(G(x)) - F^{(i)}(y)||_2^2$$

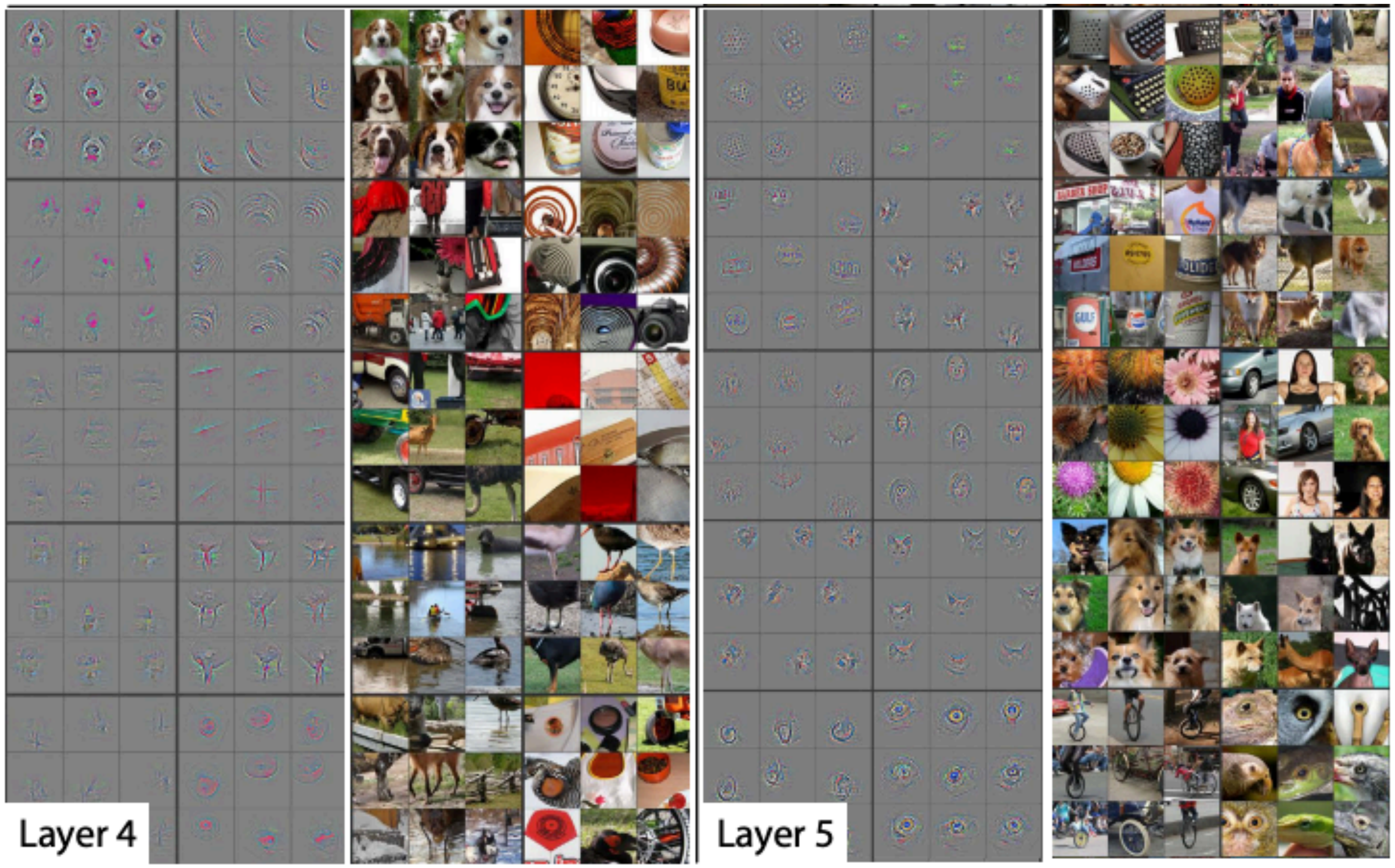
The number of elements in the (i)-th layer

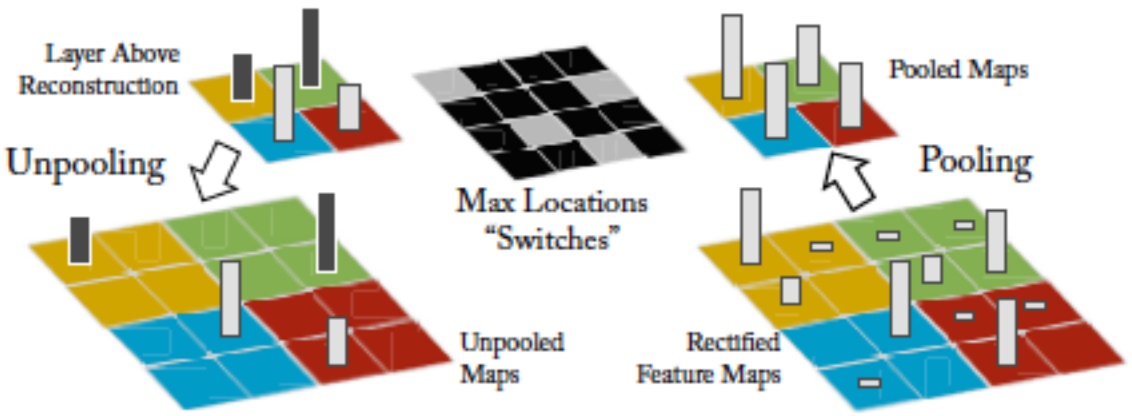
What has a CNN Learned?



Zeiler and Fergus. In ECCV, 2014.

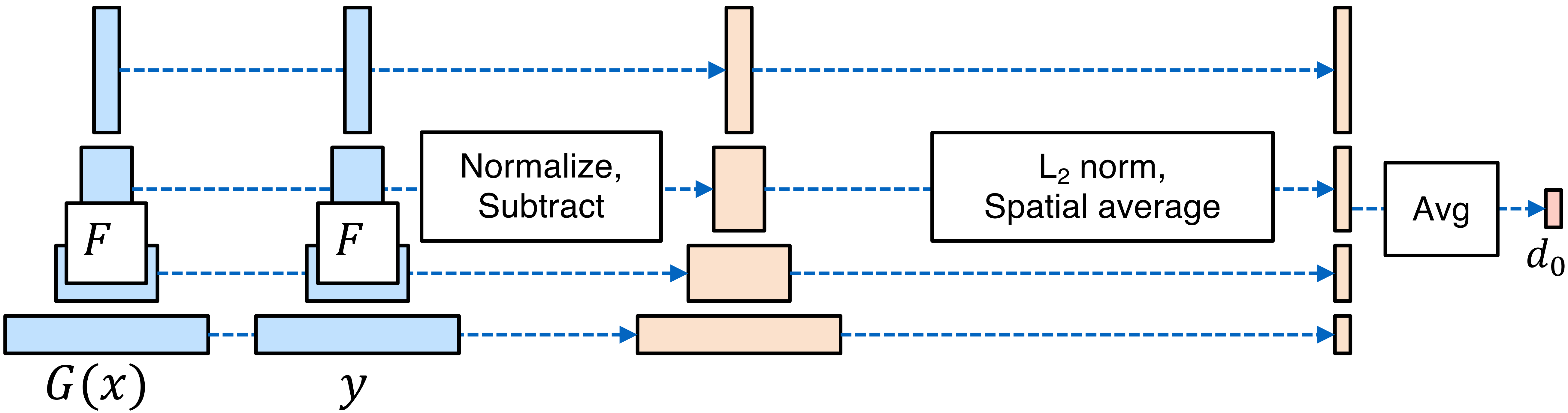
What has a CNN Learned?





Zeiler and Fergus. In ECCV, 2014.

CNNs as a Perceptual Metric



Perceptual Loss

$$\arg\min_{G} \mathbb{E}_{(x,y)} \sum_{i=1}^{N} \lambda_i \frac{1}{M_i} ||F^{(i)}(G(x)) - F^{(i)}(y)||_2^2$$

weight

(i)-th layer

The number of elements in the (i)-th layer

How Different are these Patches?

$D\Big($  ,  $\Big)$

Zhang, Isola, Efros, Shechtman, Wang.
The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018.

Which patch is more similar to the middle?

< Type 1 >

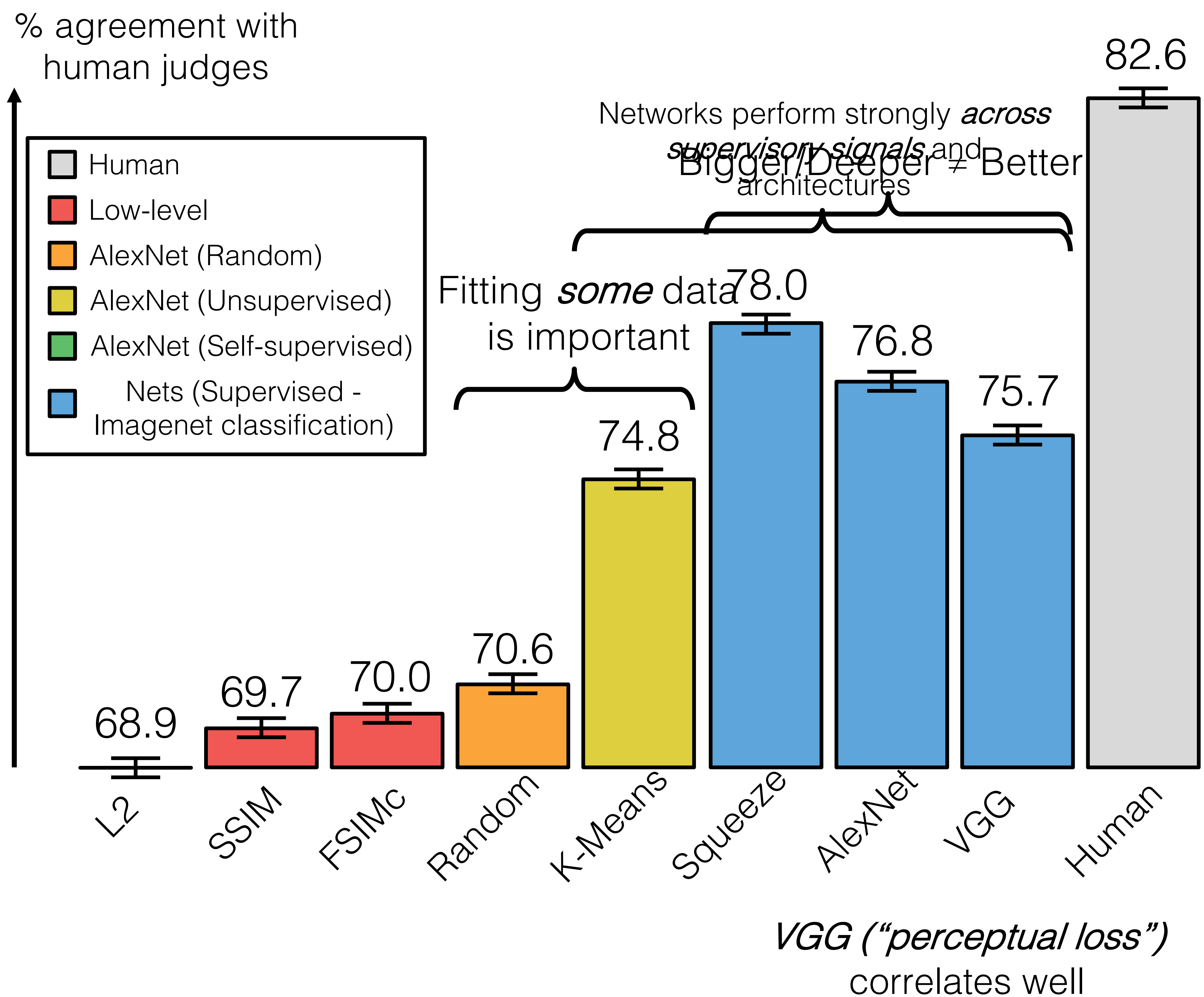
Humans

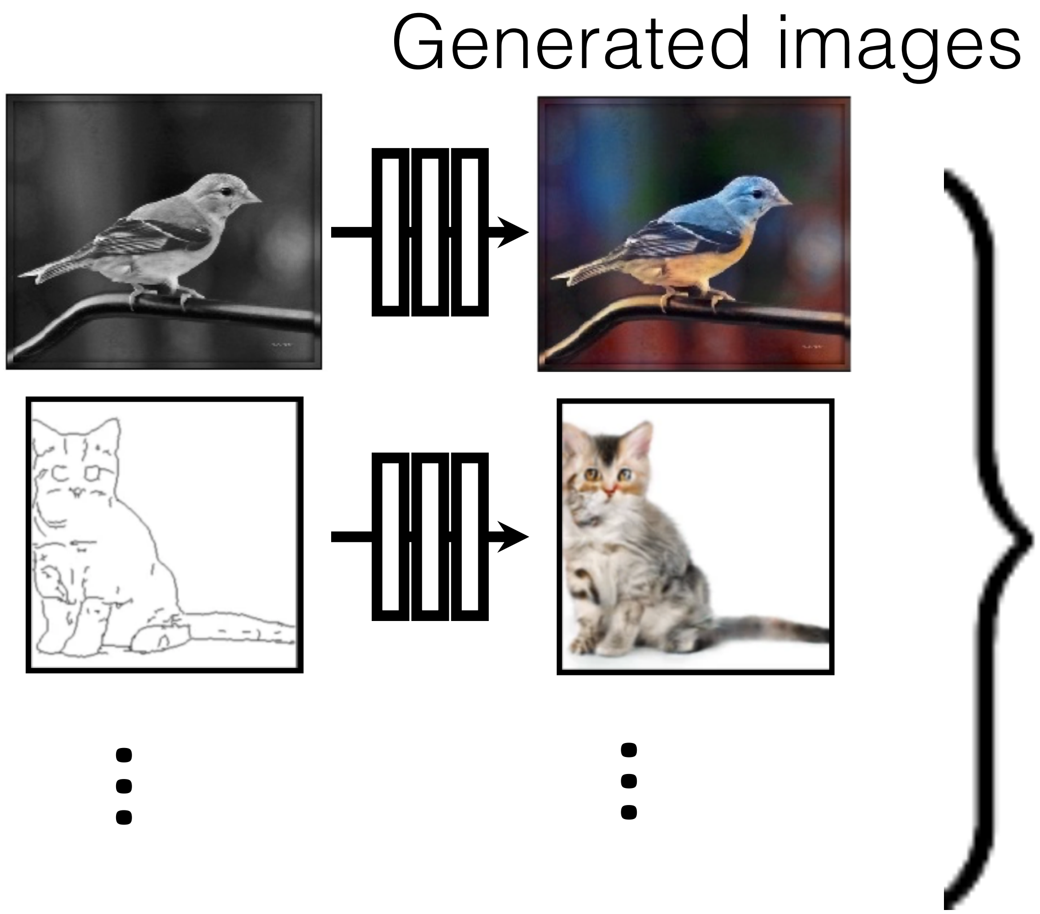
L2/PSNR

SSIM/FSIMc

Deep Networks?

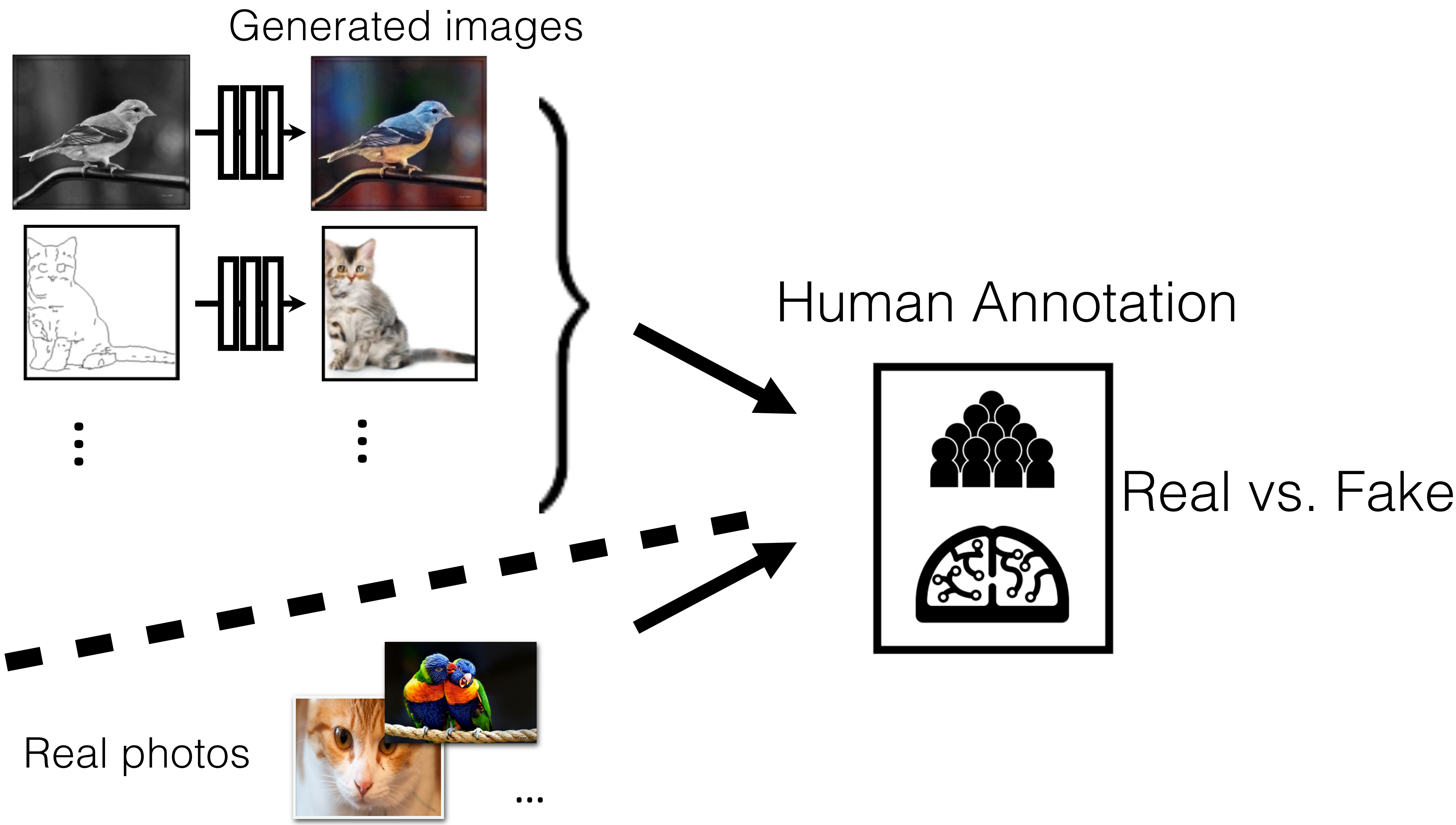
< Type 2 >


% agreement with human judges
Human
Low-level
AlexNet (Random)
AlexNet (Unsupervised)
AlexNet (Self-supervised)
Nets (Supervised - Imagenet classification)
Networks perform strongly *across supervisory signals* and architectures
Bigger/Deeper ≠ Better
Fitting *some* data is important
68.9
69.7
70.0
70.6
74.8
78.0
76.8
75.7
82.6
L2
SSIM
FSIMc
Random
K-Means
Squeeze
AlexNet
VGG
Human
VGG ("perceptual loss") correlates well



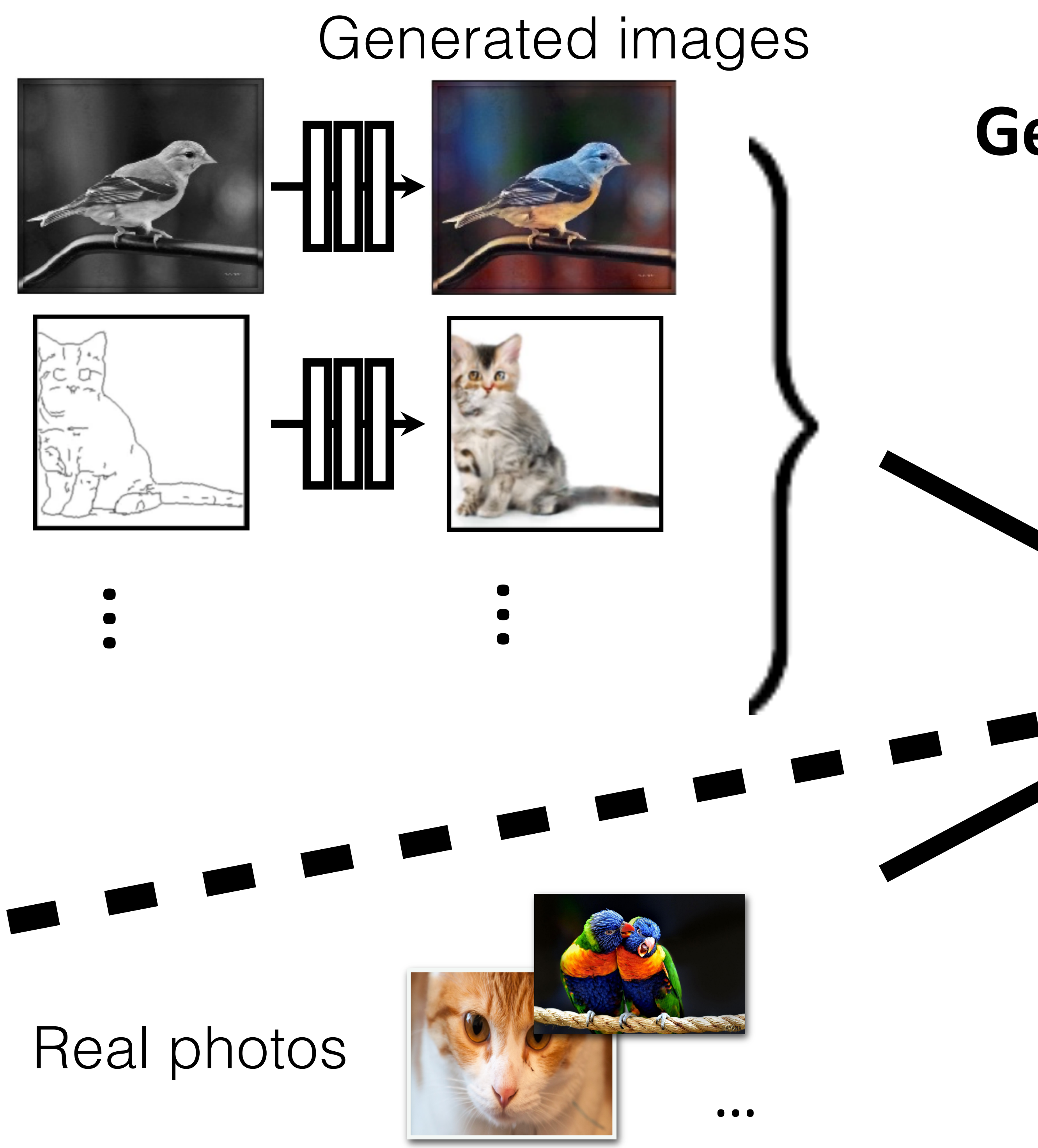
Universal loss?

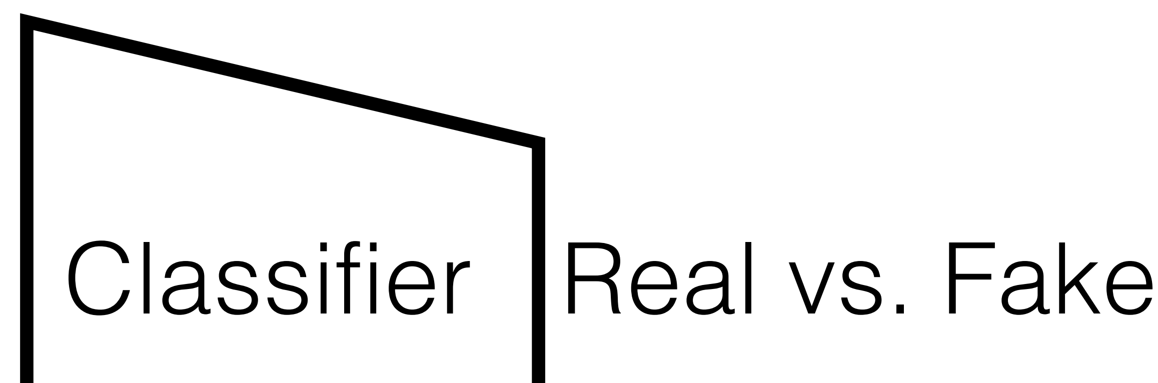
Learning with Human Perception



[Zhu et al. 2014]

Generative Adversarial Network (GANs)





[Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio 2014]

Image synthesis from “noise”

G

$z \sim p(z)$

→ Generator →

$x = G(z)$

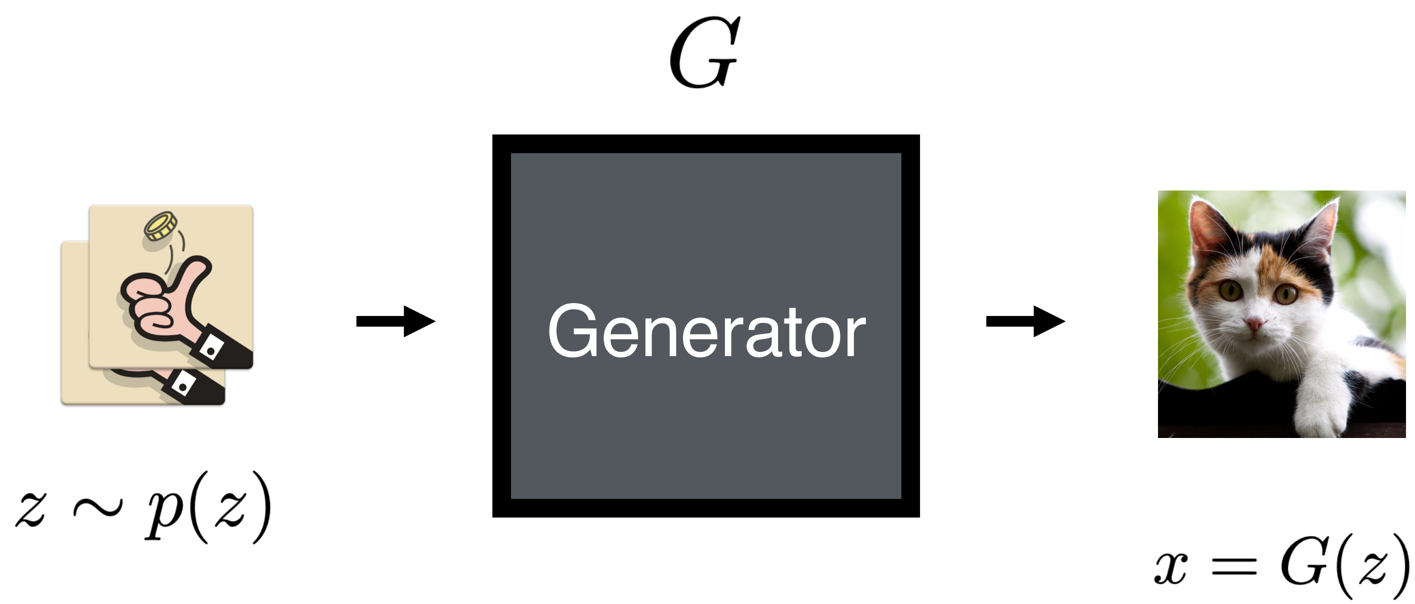
Sampler

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

$$z \sim p(z)$$

$$x = G(z)$$

Image synthesis from “noise”



Sampler

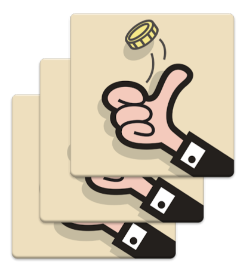
$$G : \mathcal{Z} \rightarrow \mathcal{X}$$

$$z \sim p(z)$$

$$x = G(z)$$

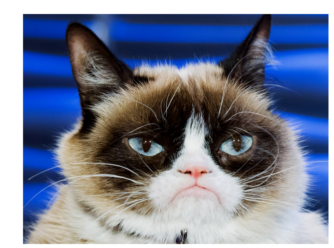
Image synthesis from “noise”

G



$z \sim p(z)$

Generator



$x = G(z)$

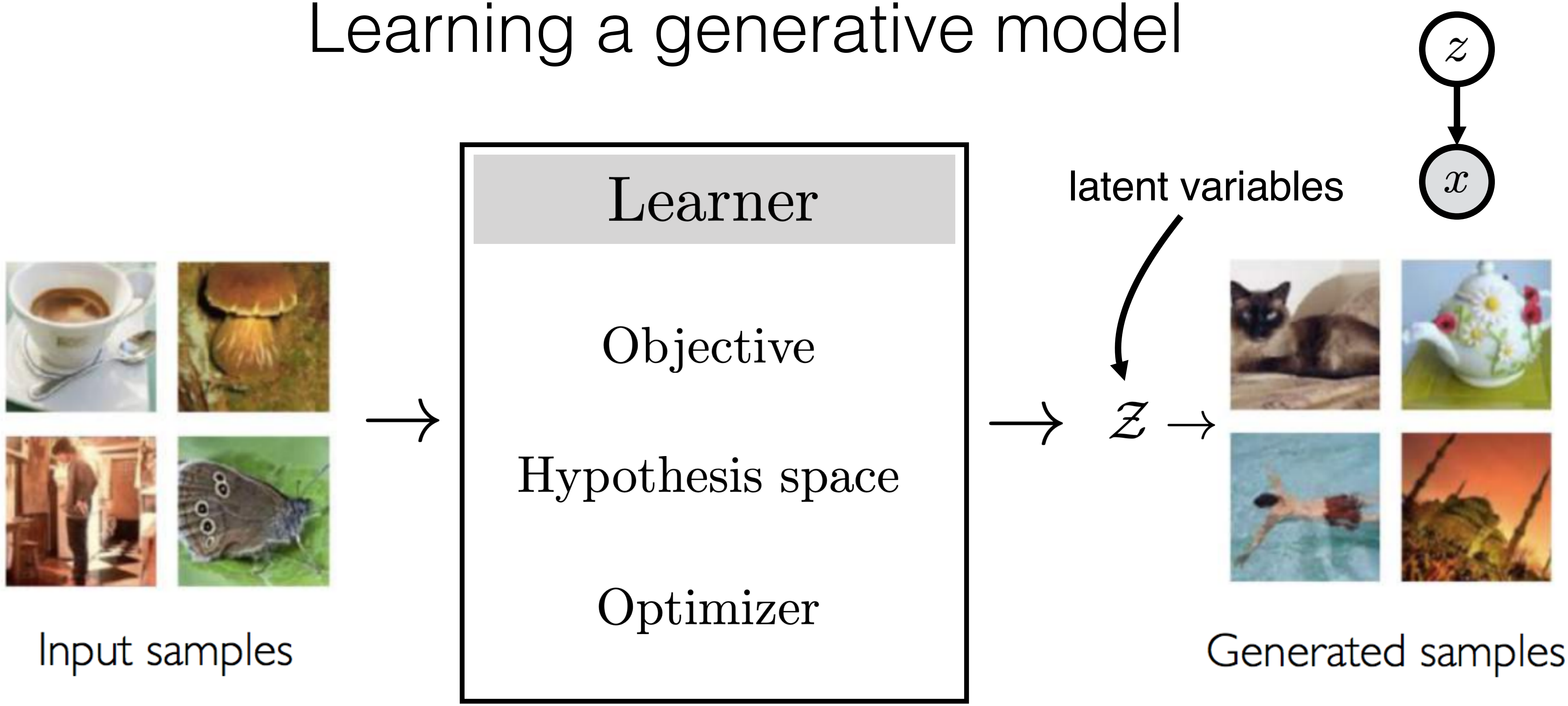
Sampler

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

$$z \sim p(z)$$

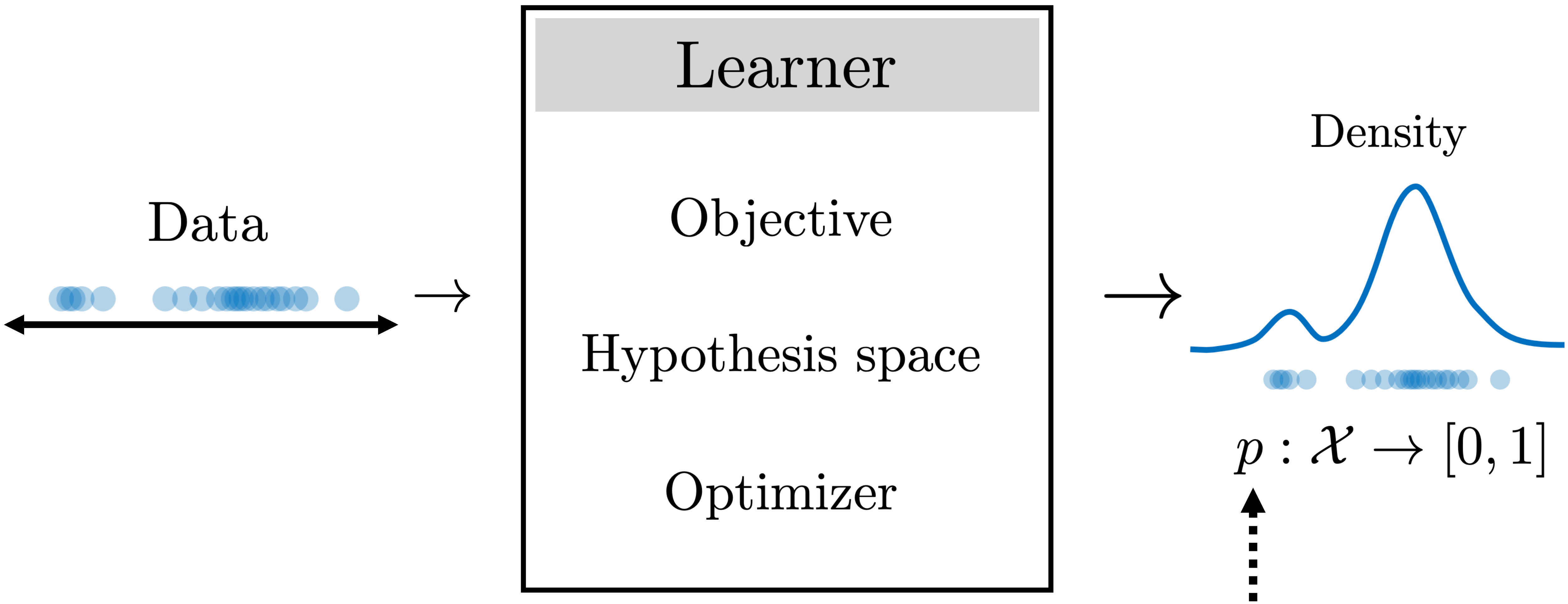
$$x = G(z)$$

Learning a generative model



[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

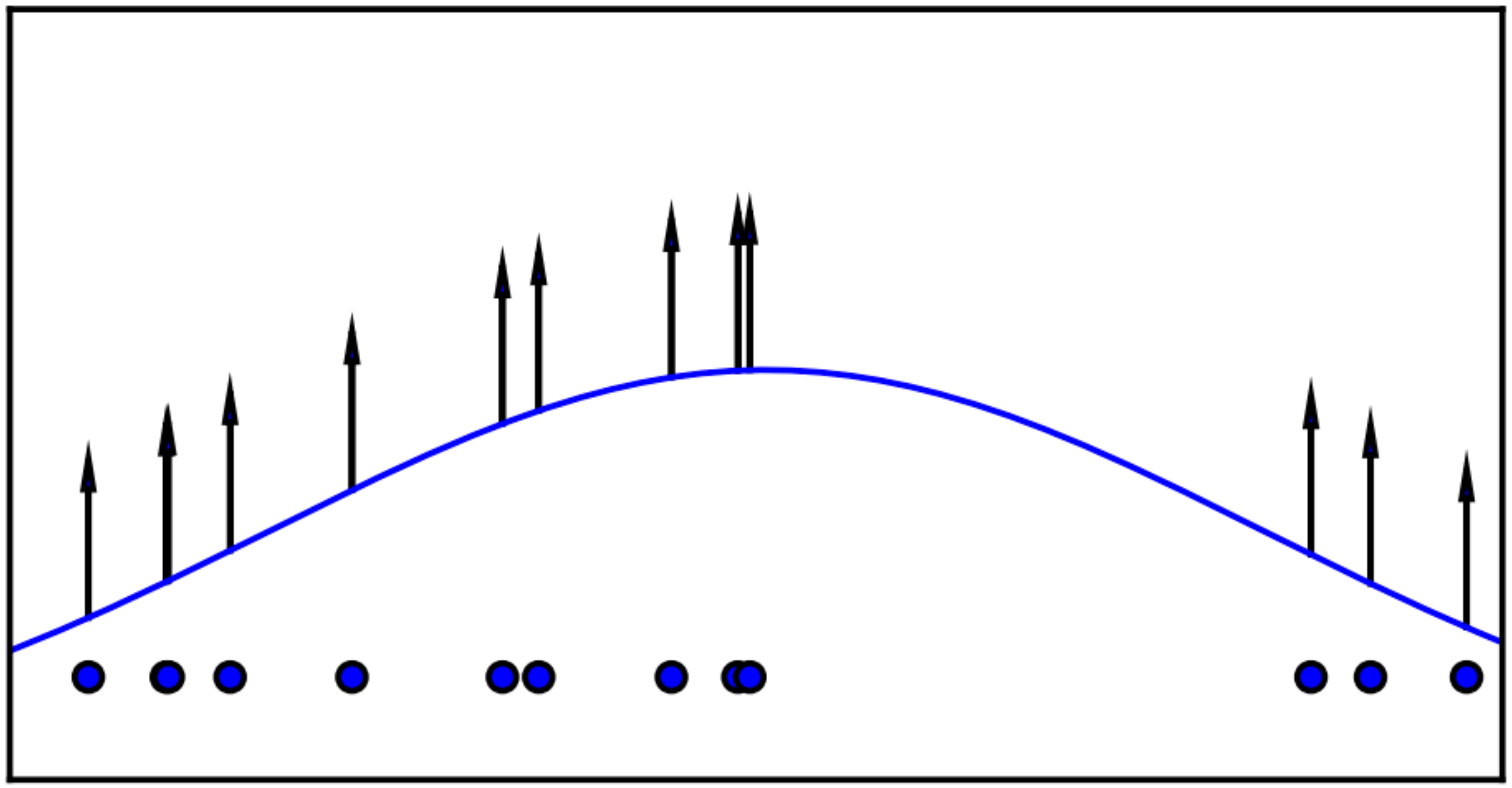


fig from [Goodfellow, 2016]

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:

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

Case study #1: Fitting a Gaussian to data

Data $\rightarrow$

Learner

Objective

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

“max likelihood”

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
max likelihood
/distribution matching

Hypothesis space
Deep net

Optimizer
SGD

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

Case study #2: learning a deep generative model

$\text{Data} \quad \rightarrow$

Learner

Objective
max likelihood
/distribution matching

Hypothesis space
Deep net

Optimizer
SGD

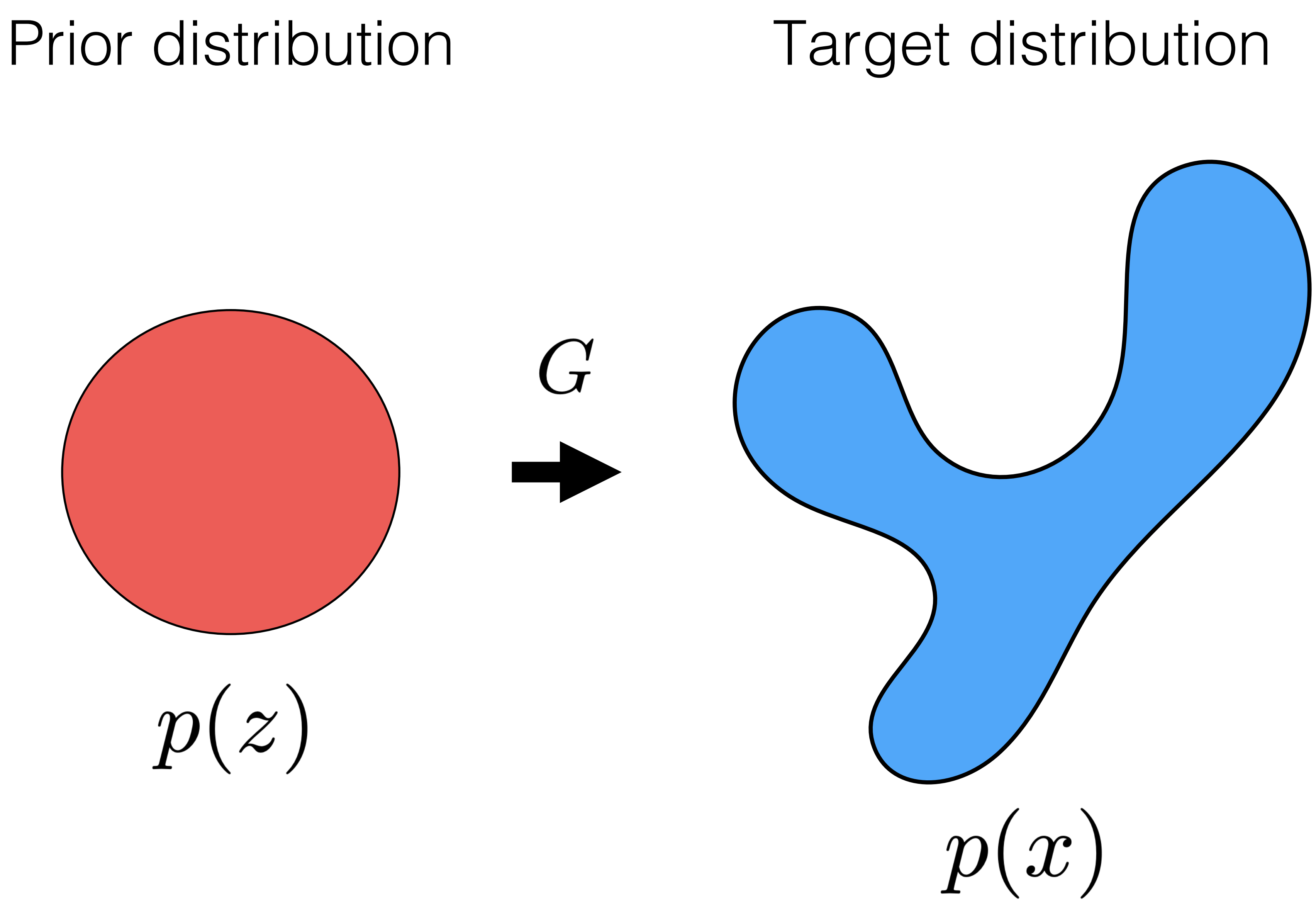
$\longrightarrow$

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

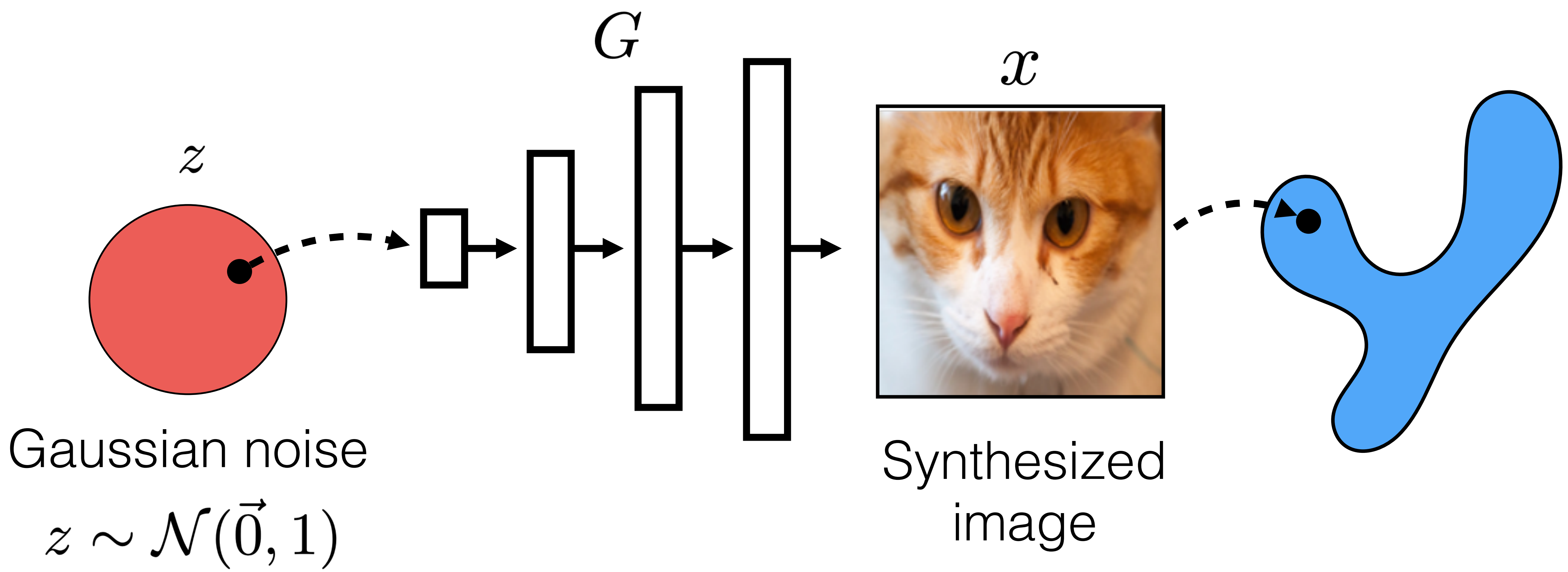
Sampler
$G : \mathcal{Z} \rightarrow \mathcal{X}$
$z \sim p(z)$
$x = G(z)$

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

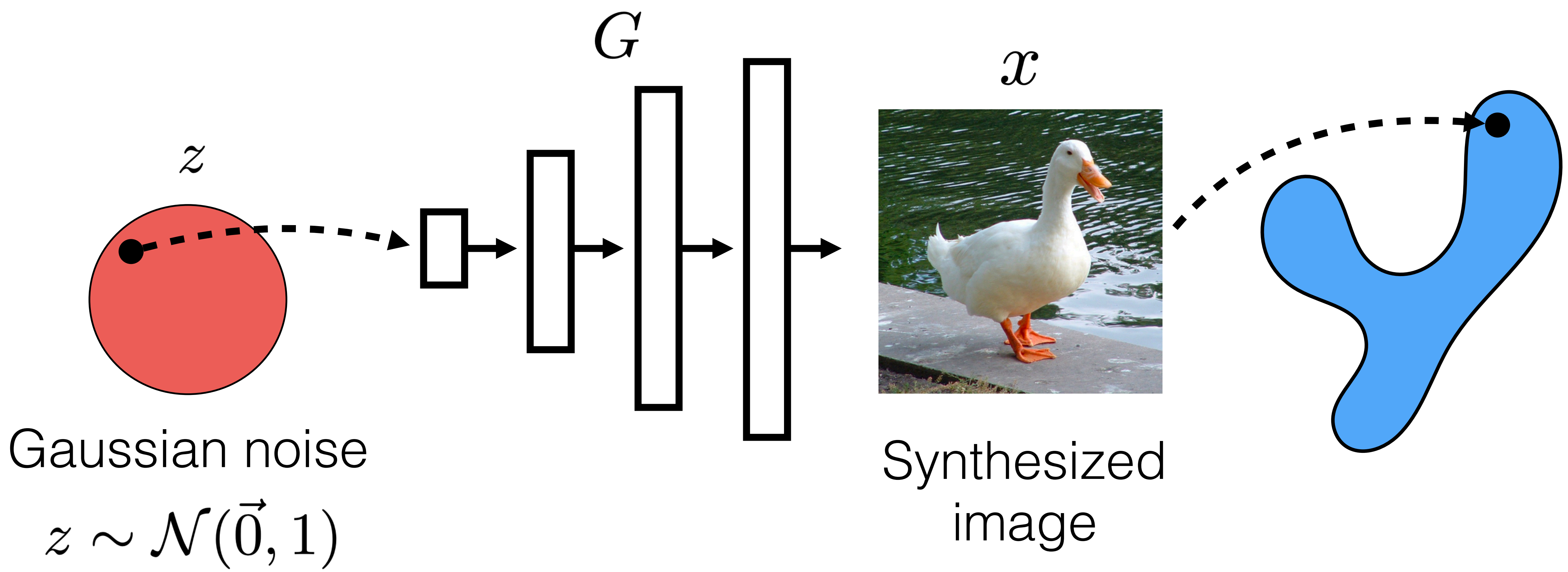
Deep generative models are distribution transformers



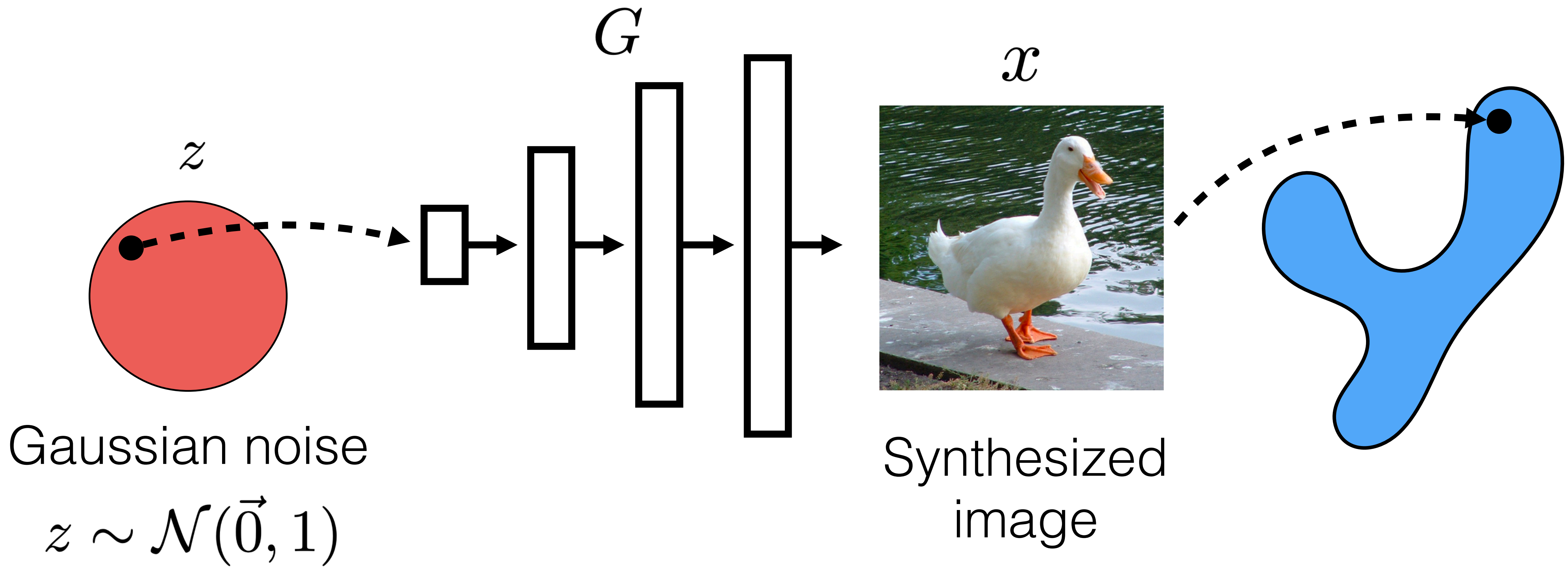
Deep generative models are distribution transformers

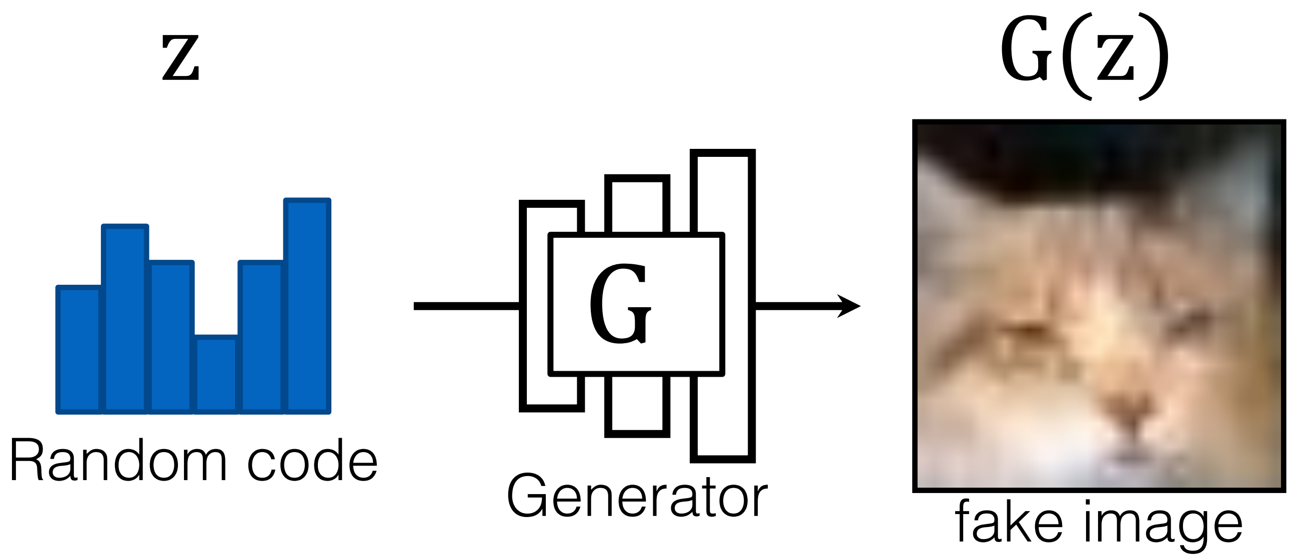


Deep generative models are distribution transformers

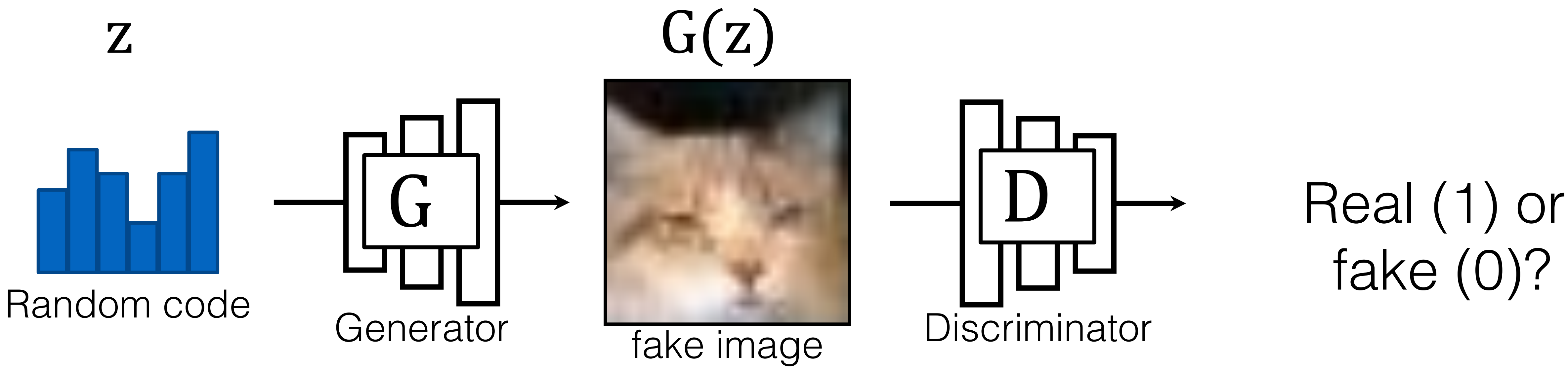


Generative Adversarial Networks (GANs)





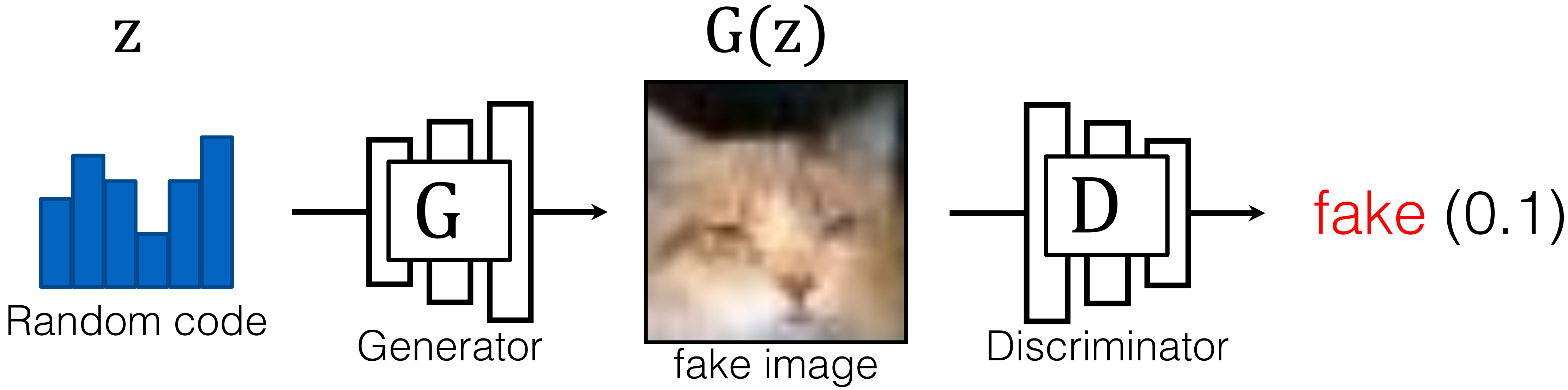
[Goodfellow et al. 2014]



A two-player game:

- G tries to generate fake images that can fool D.
- D tries to detect fake images.

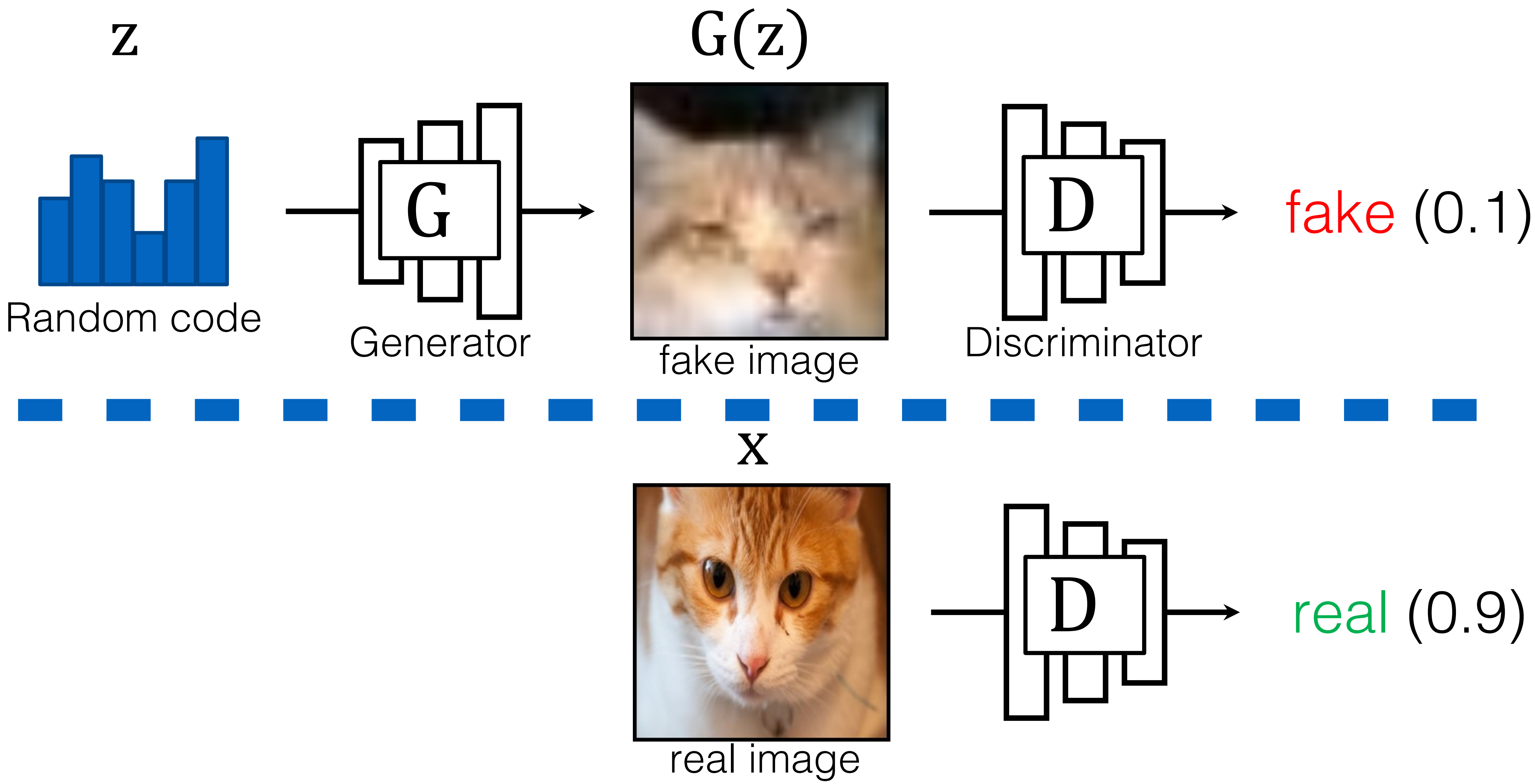
[Goodfellow et al. 2014]



Learning objective (GANs)

$$\min_{G} \max_{\boxed{D}} \mathbb{E}[\boxed{\log(1-D(G(z)))}]$$

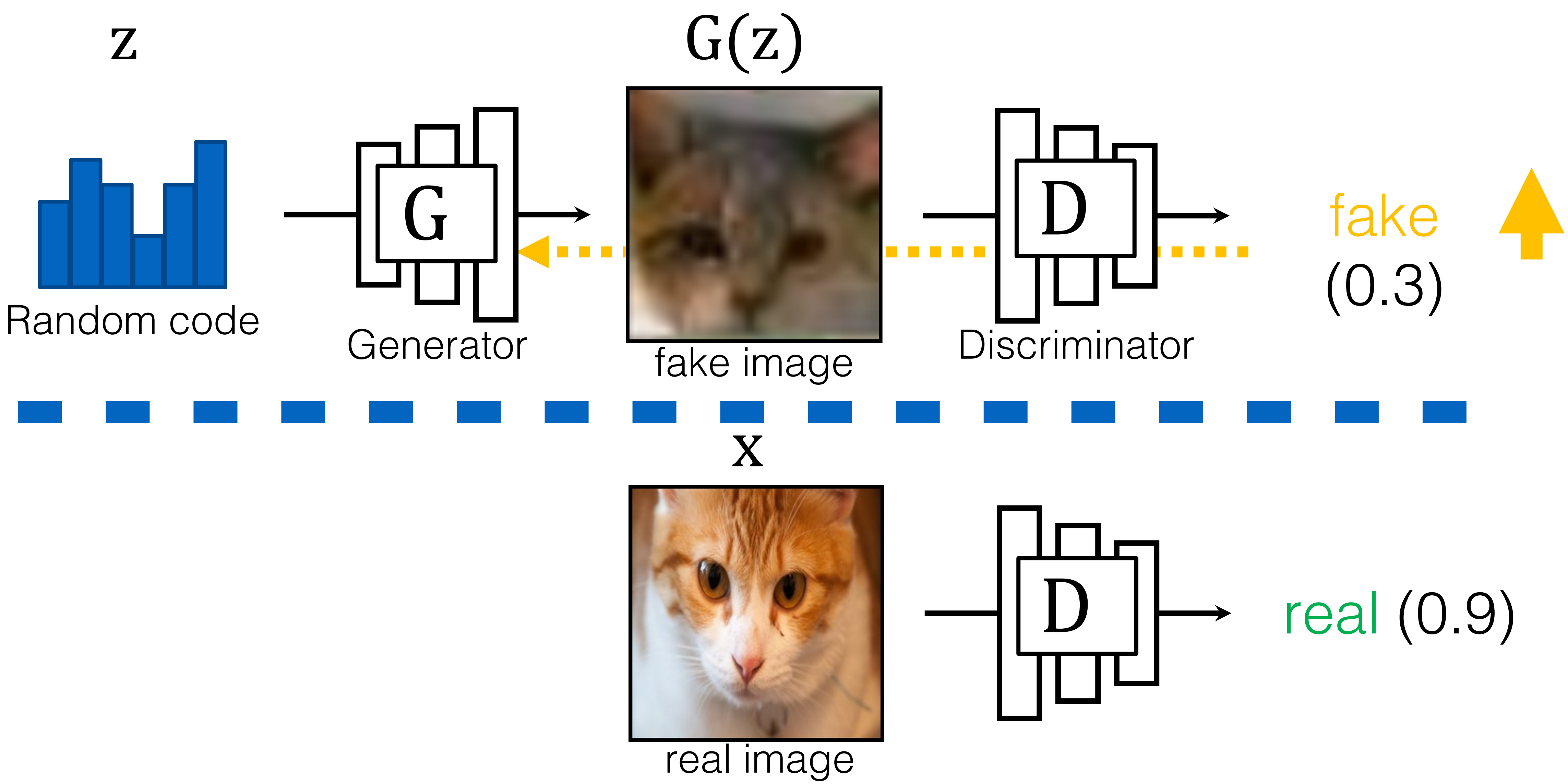
[Goodfellow et al. 2014]



Learning objective (GANs)

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

[Goodfellow et al. 2014]



Learning objective (GANs)

$$\min_{\boxed{G}} \max_{D} \mathbb{E}[\boxed{\log(1-D(G(z)))}+\log D(x)]$$

[Goodfellow et al. 2014]

GANs 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]

Thank You!

16-726, Spring 2021

https://learning-image-synthesis.github.io/