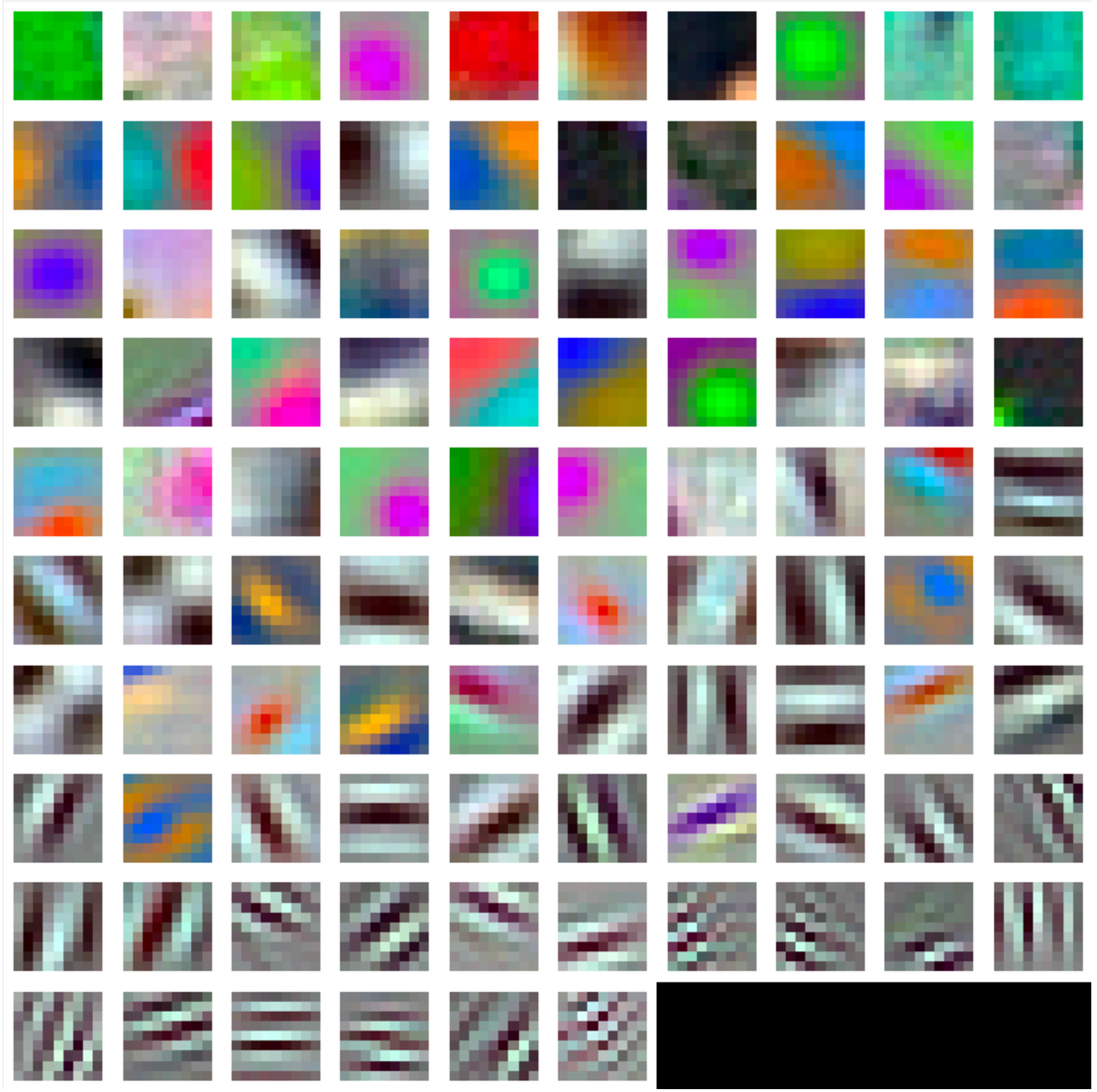
Lecture 13: GANs and autoregressive models

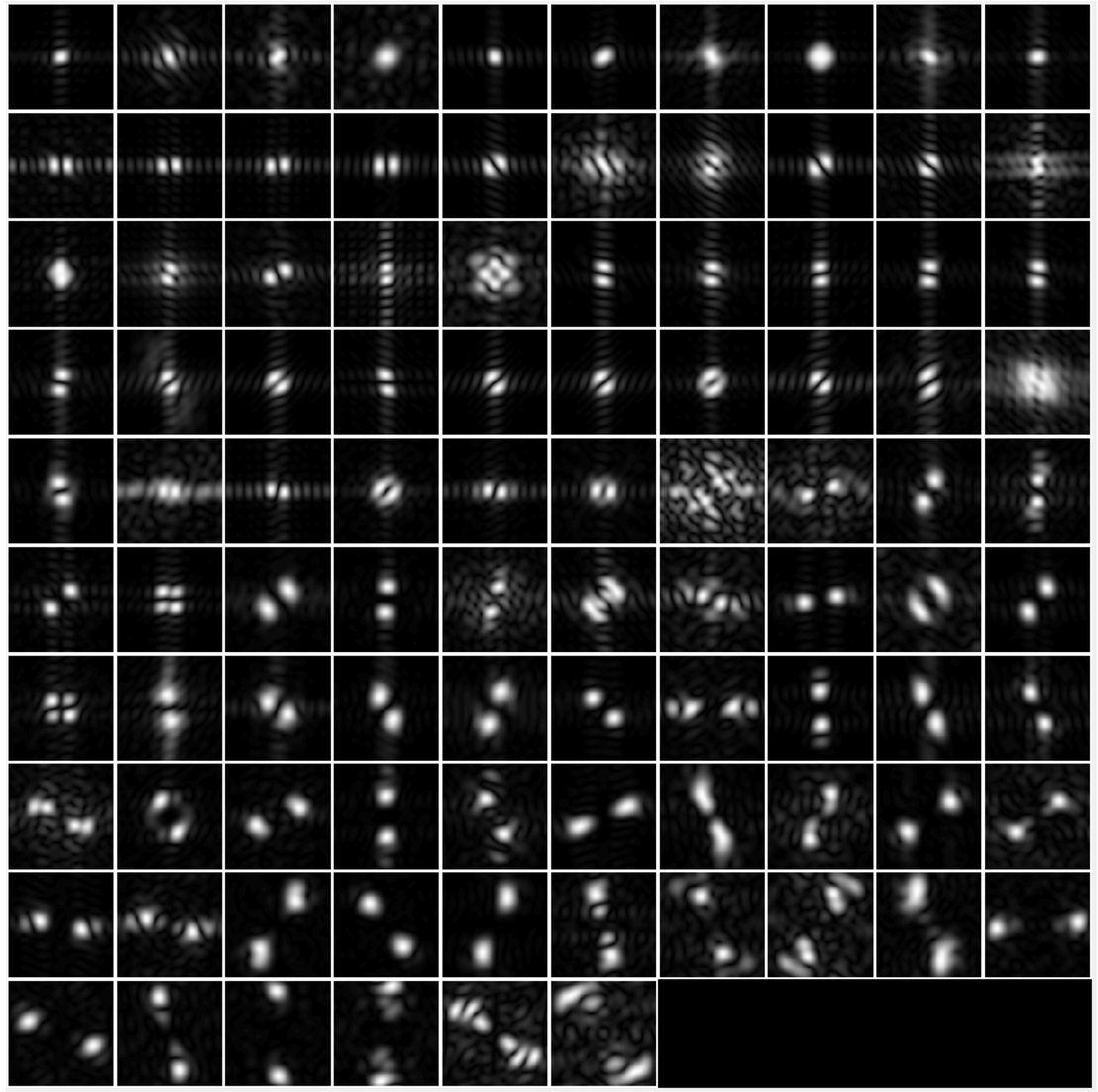
Today

- Texture synthesis
- Generative adversarial nets (GANs)
- Autoregressive models

Recall: neural net feature visualization

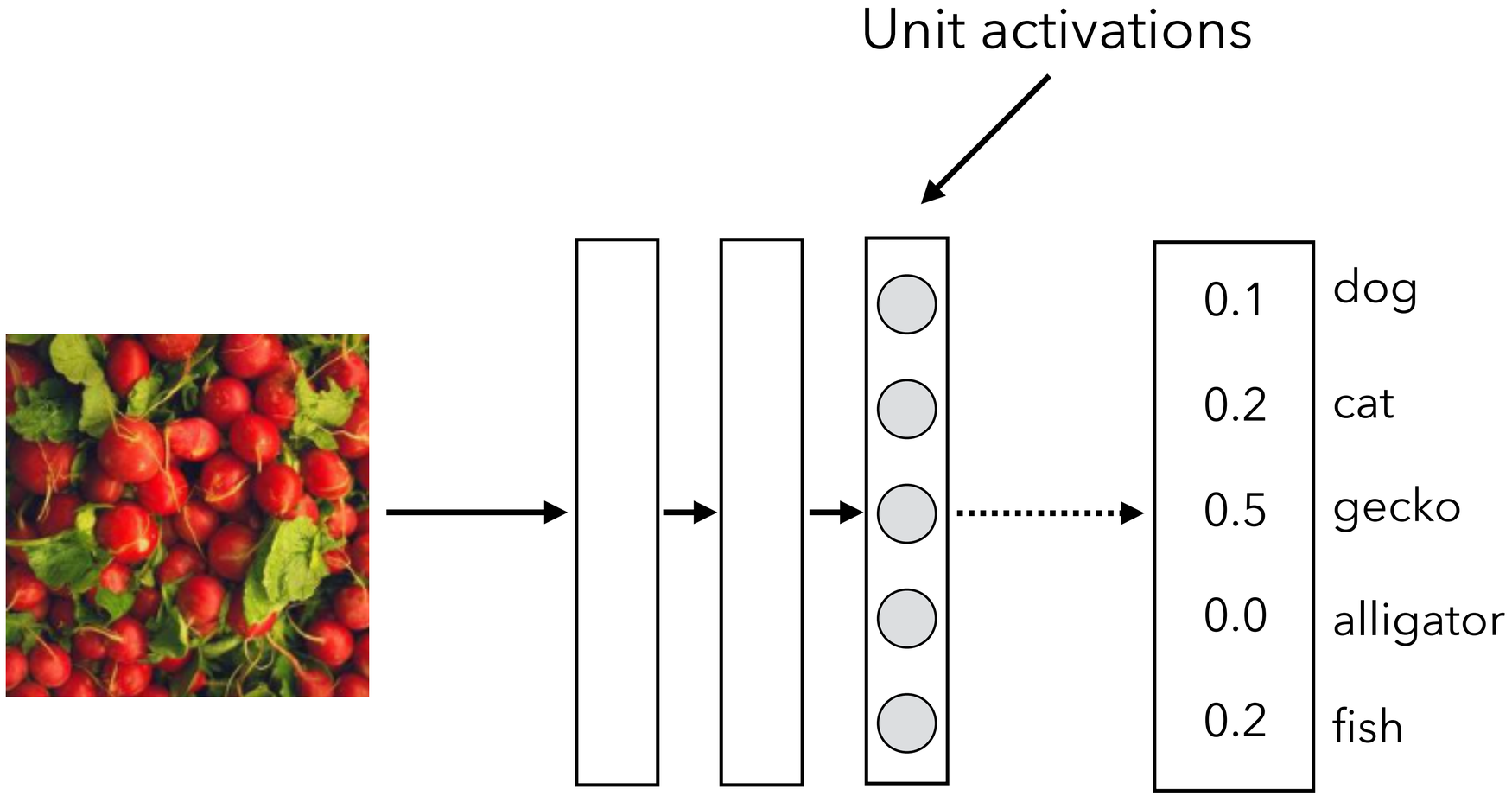


96 Units in conv1

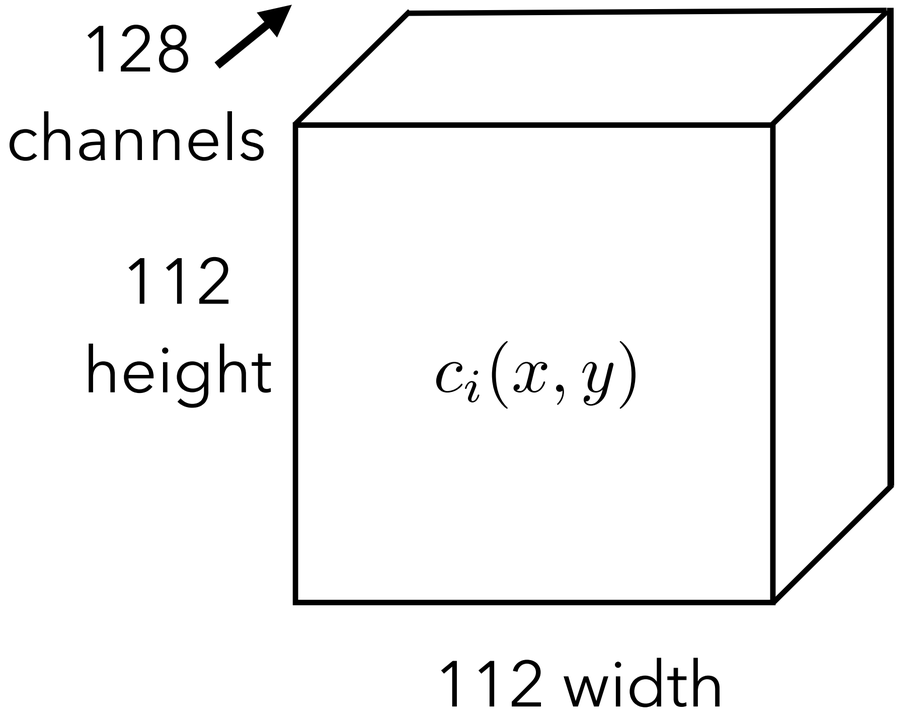


Source: Isola, Torralba, Freeman

Extracting features from a trained network



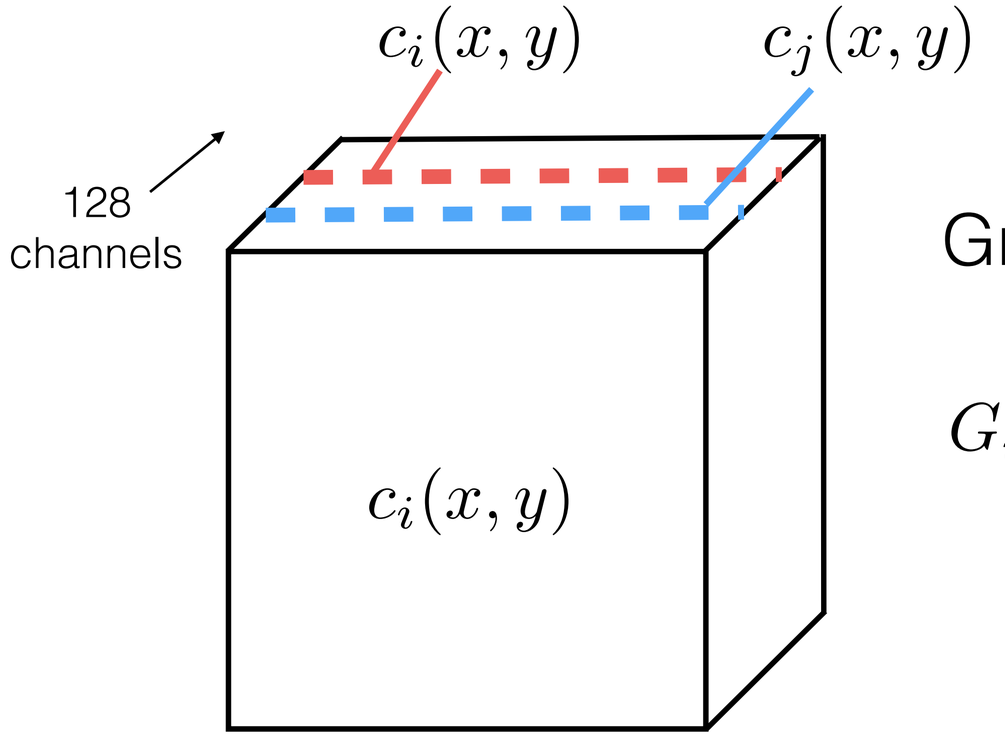
Extracting neural net features



conv2 feature activations

(112 x 112) x 128 for conv2 of VGG19

Capturing feature correlations



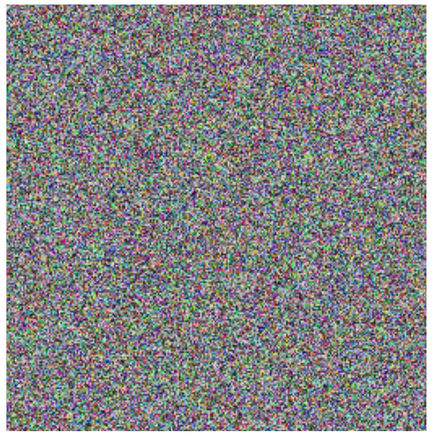
Gram (≈ covariance) matrix:

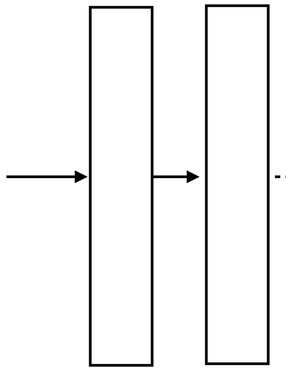
$$G_{ij} = \sum_{x=1}^{w} \sum_{y=1}^{h} c_i(x, y) c_j(x, y)$$

[Gatys et al. 2016]

Idea: correlations between unit activations convey texture.
Discard global spatial information.

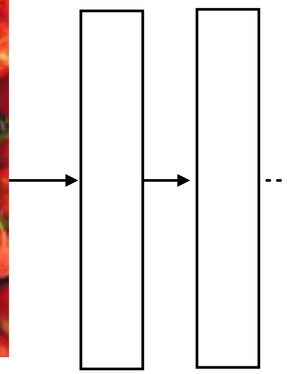
Matching image statistics





$G_{ij}(\hat{I})$

Target



$G_{ij}(I)$

Find $\hat{I}$ by minimizing:

$$\sum_{i=1}^{128}\sum_{j=1}^{128}(G_{ij}(I) - G_{ij}(\hat{I}))^2$$

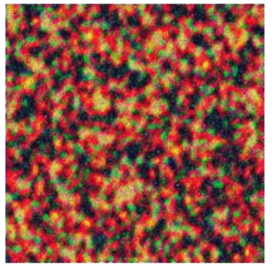
Implementation details:

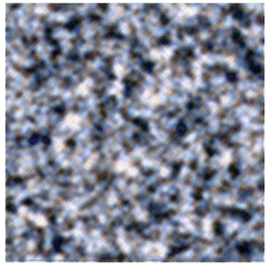
- Minimize with gradient descent
- Use many layers of network

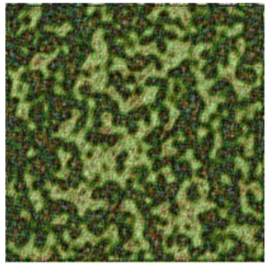
High → Low

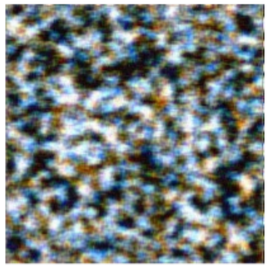
original

conv1_1

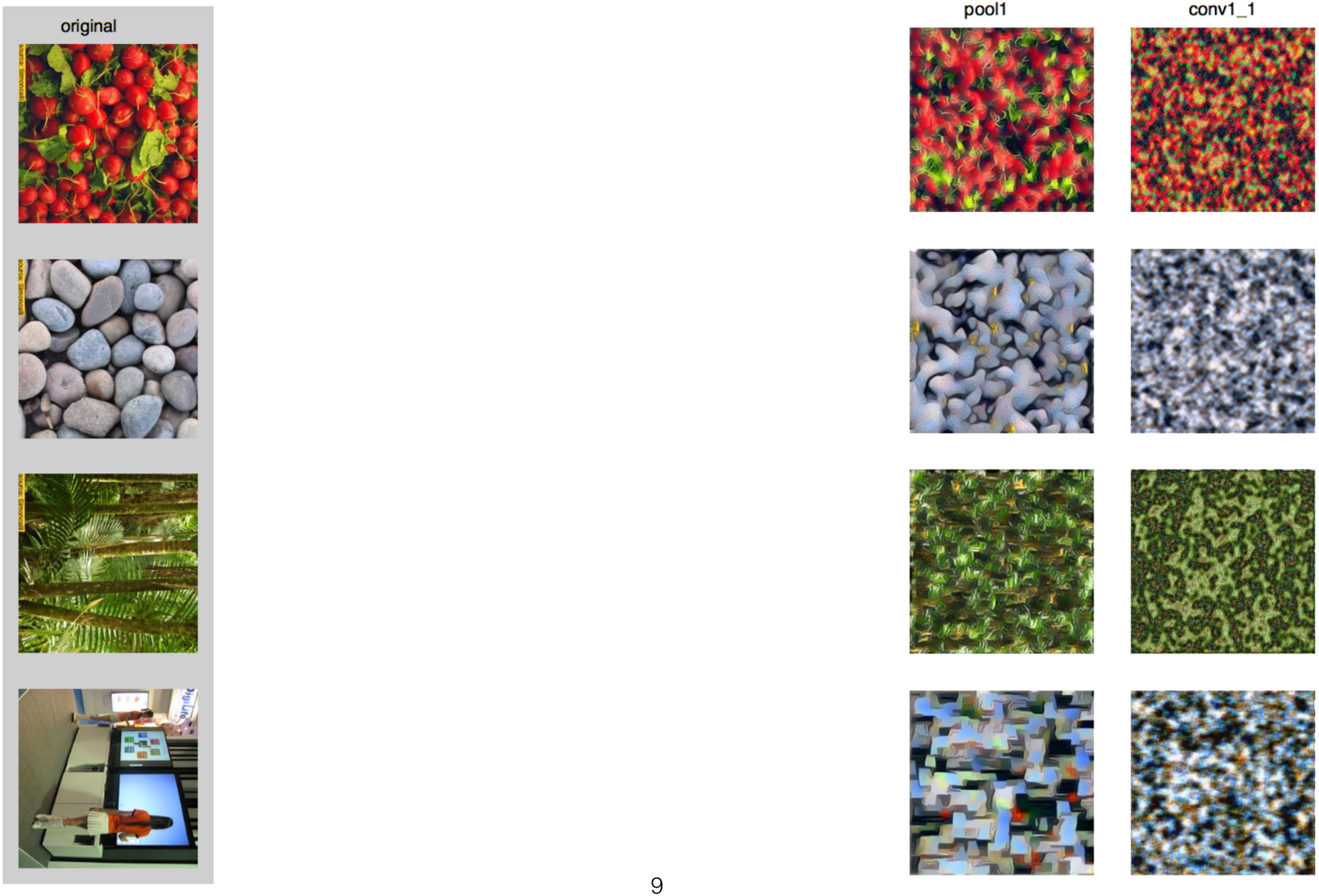




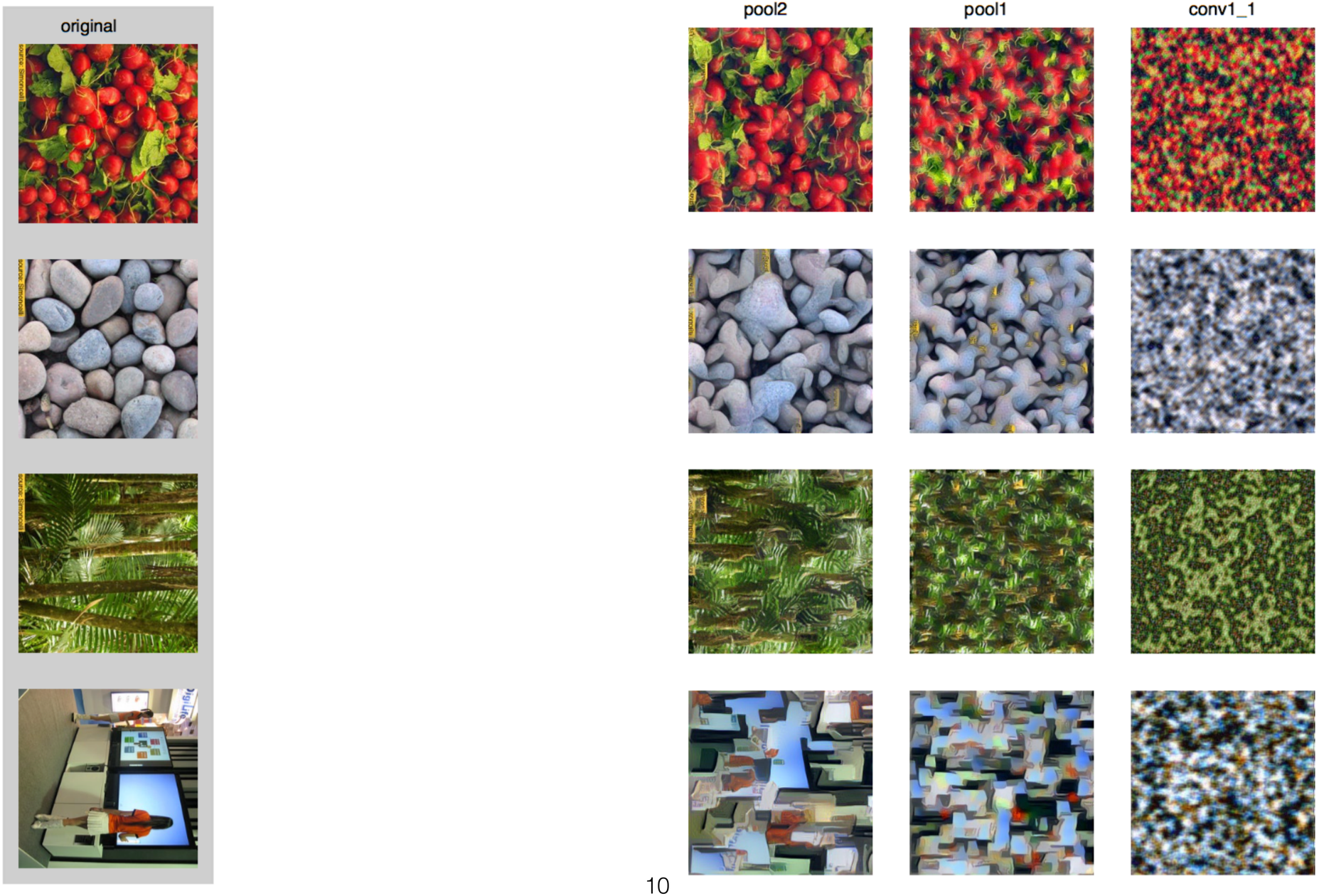




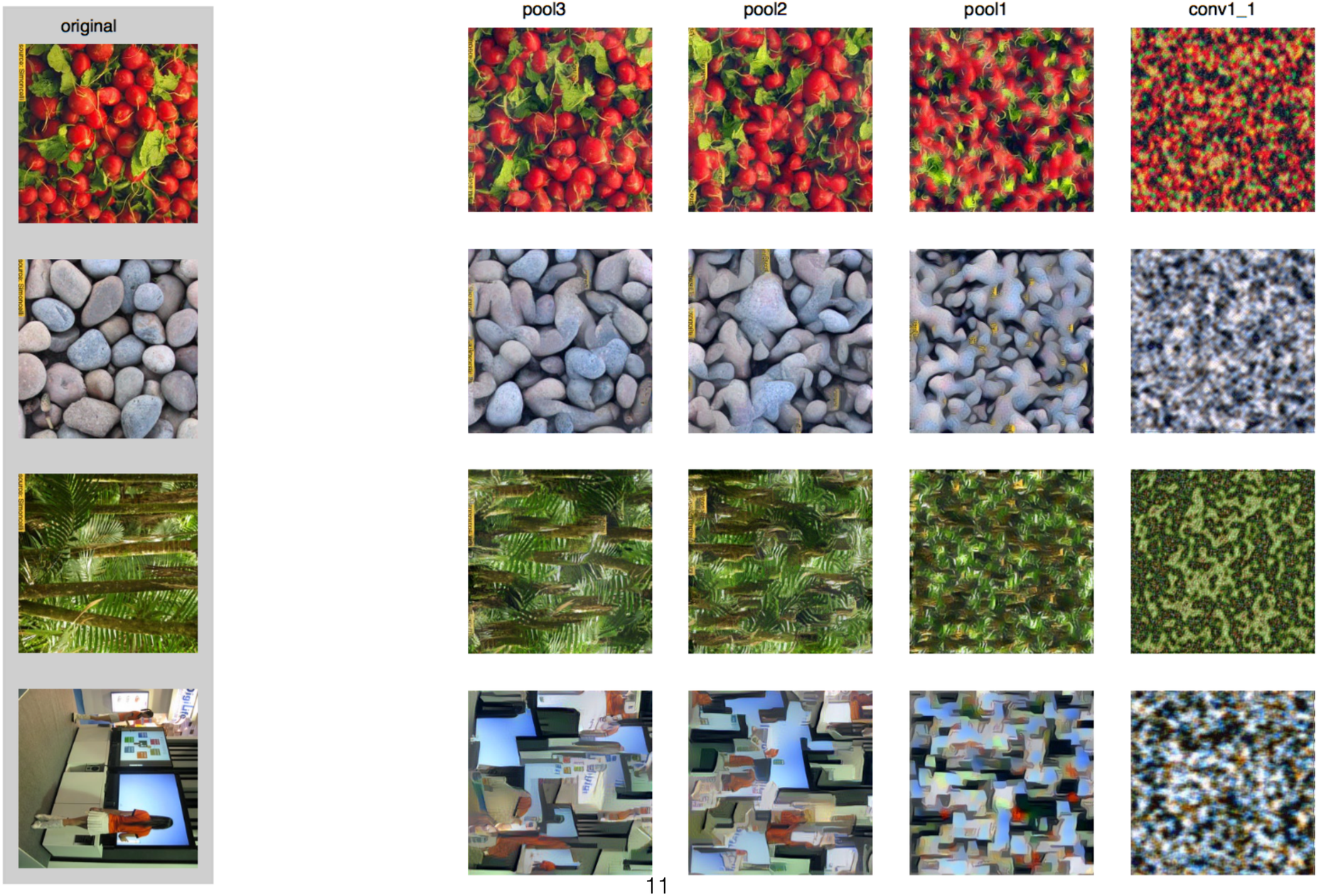
High → Low



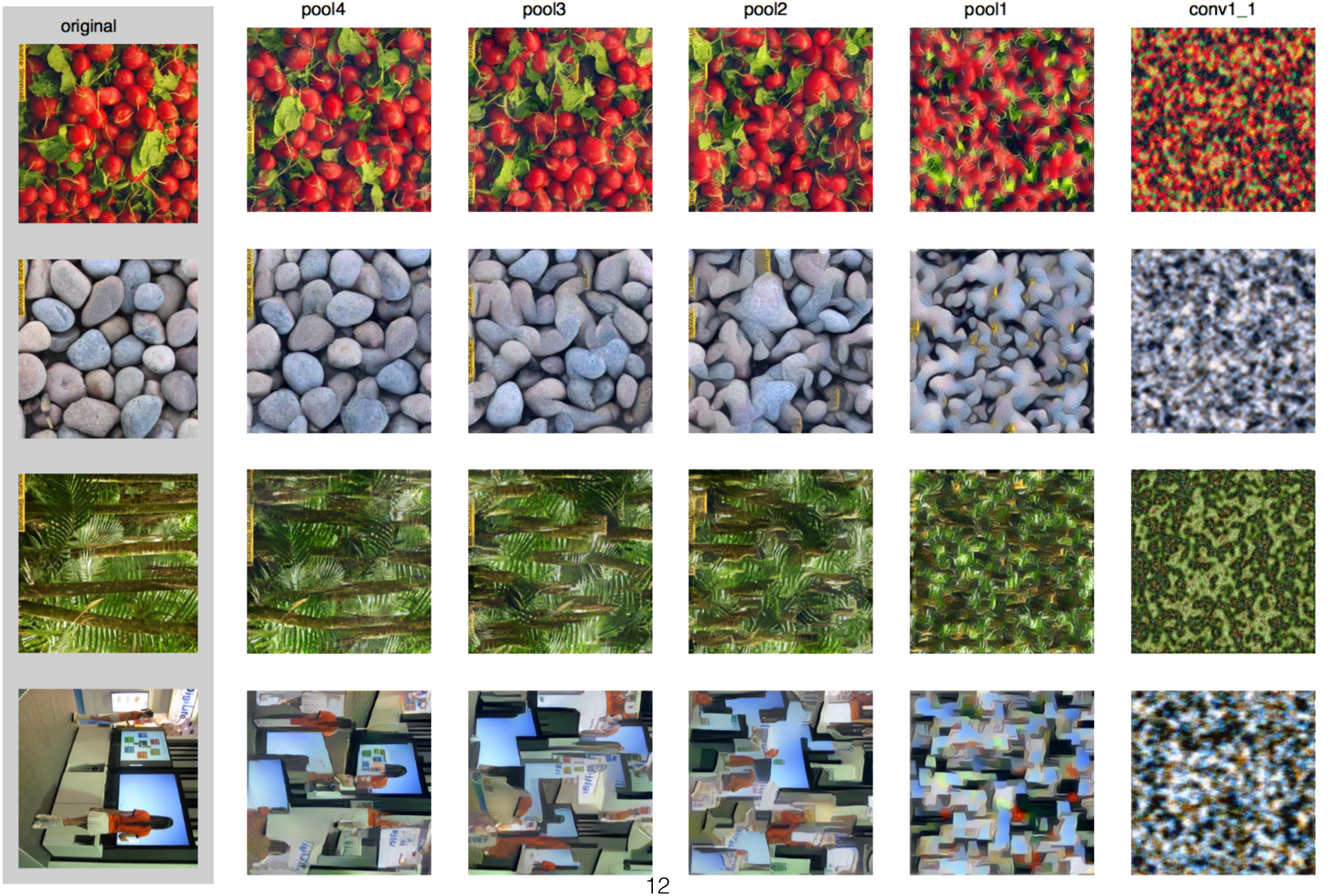
High → Low



High → Low



High → Low



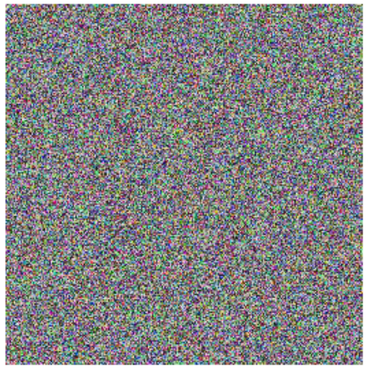
pool4
original

Texture captures artistic style

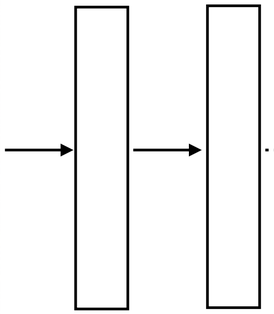
Can we transfer the style of a painting to a photo?

[Gatys et al. 2016]

Match the **style** of the painting.



Synthesized image



$G_{ij}(\hat{I})$

$$\sum_{i=1}^{128}\sum_{j=1}^{128}(G_{ij}(\hat{I}) - G_{ij}(I))^2$$

Perceptual loss:

usually distance in feature space

… and the **content** of the photo.

$$\sum_{i}\sum_{x,y}(c_i(x,y) - \hat{c}_i(x,y))^2$$

$c_i(x,y)$

$\hat{c}_i(x,y)$

London during the day.

New York at night.

Neural networks that generate images

Image classification

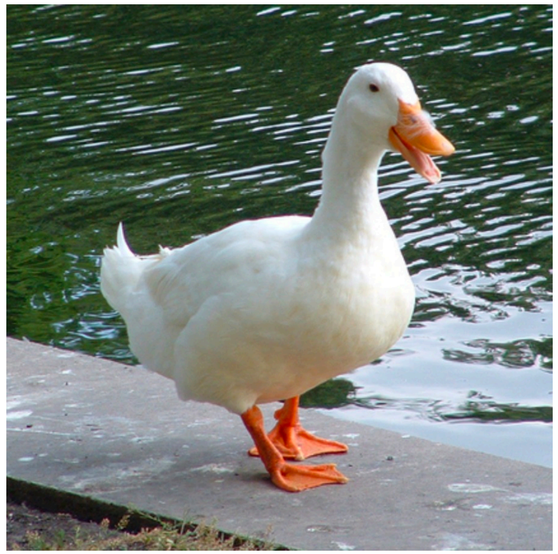


image **x**

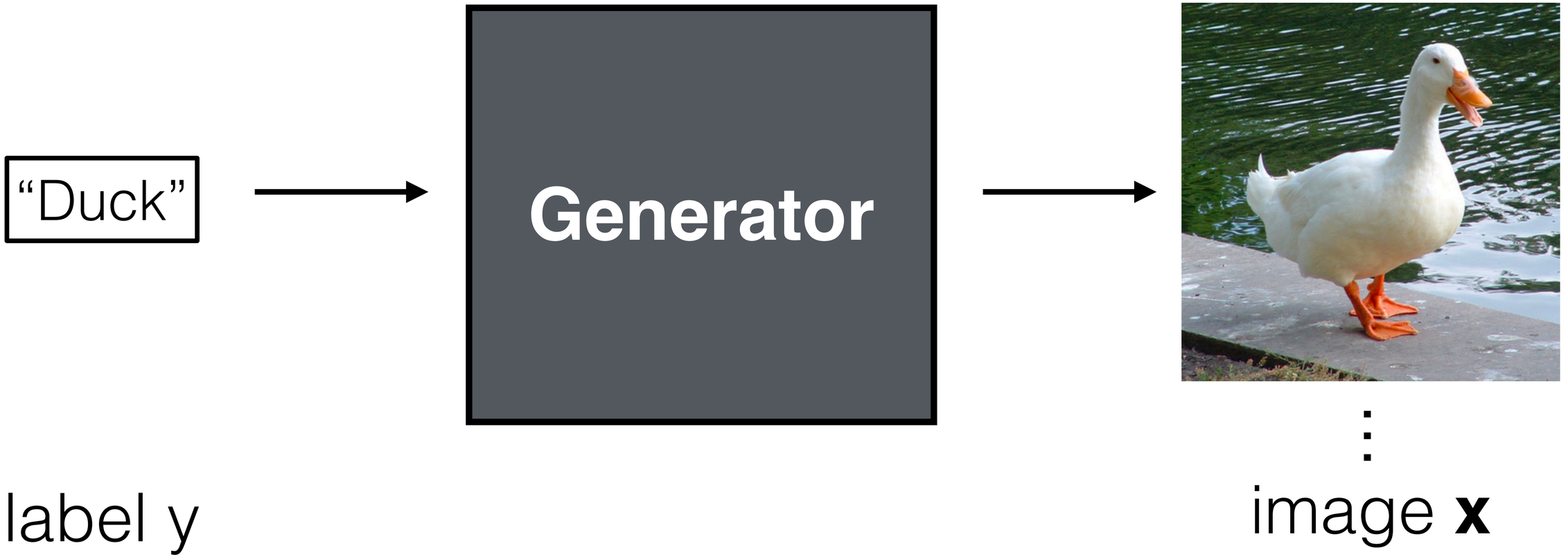
Classifier

"Duck"

label y

Source: Isola, Freeman, Torralba

Image synthesis

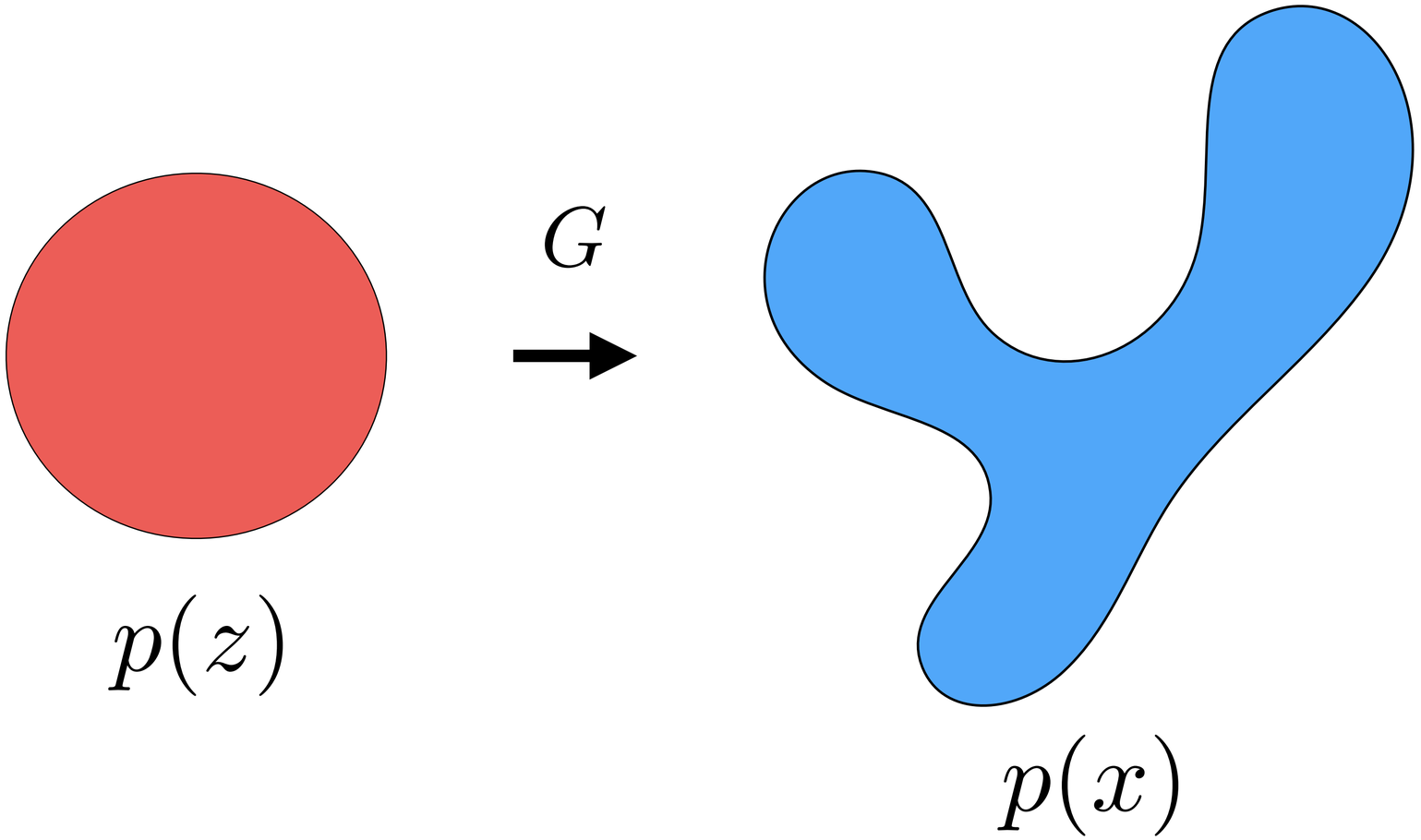


Source: Isola, Freeman, Torralba

Neural networks as distribution transformers

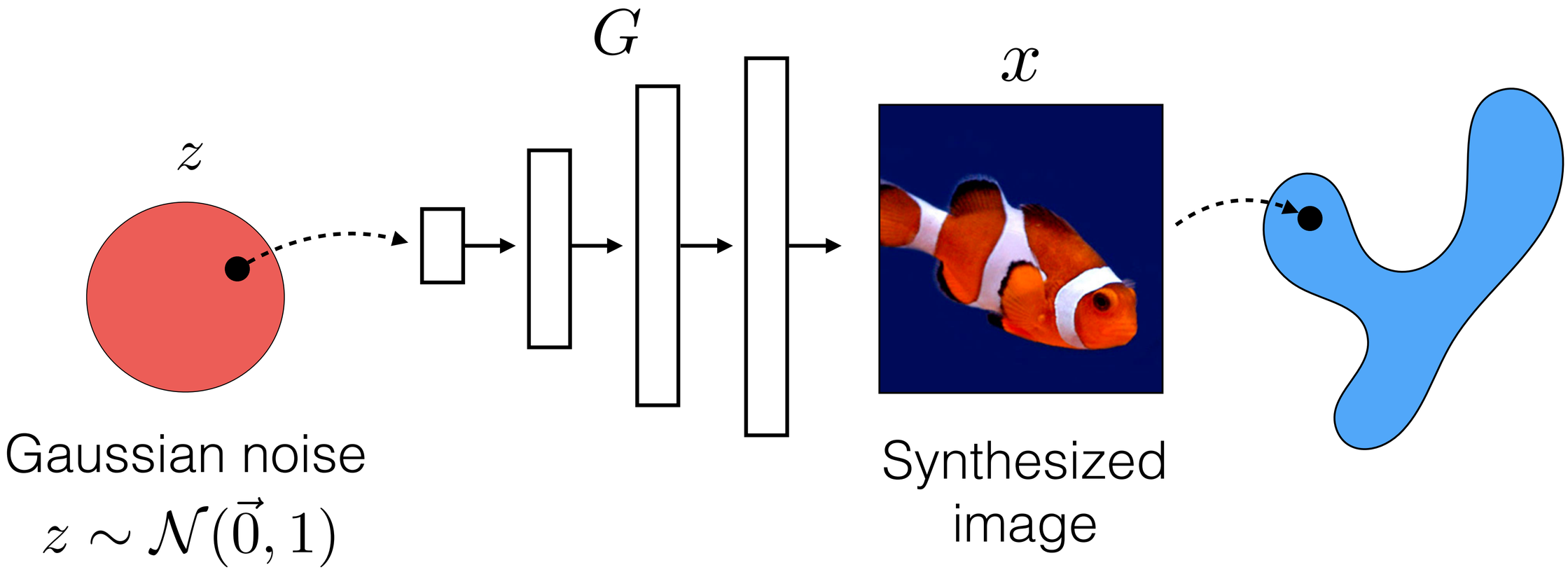
Source distribution

Target distribution



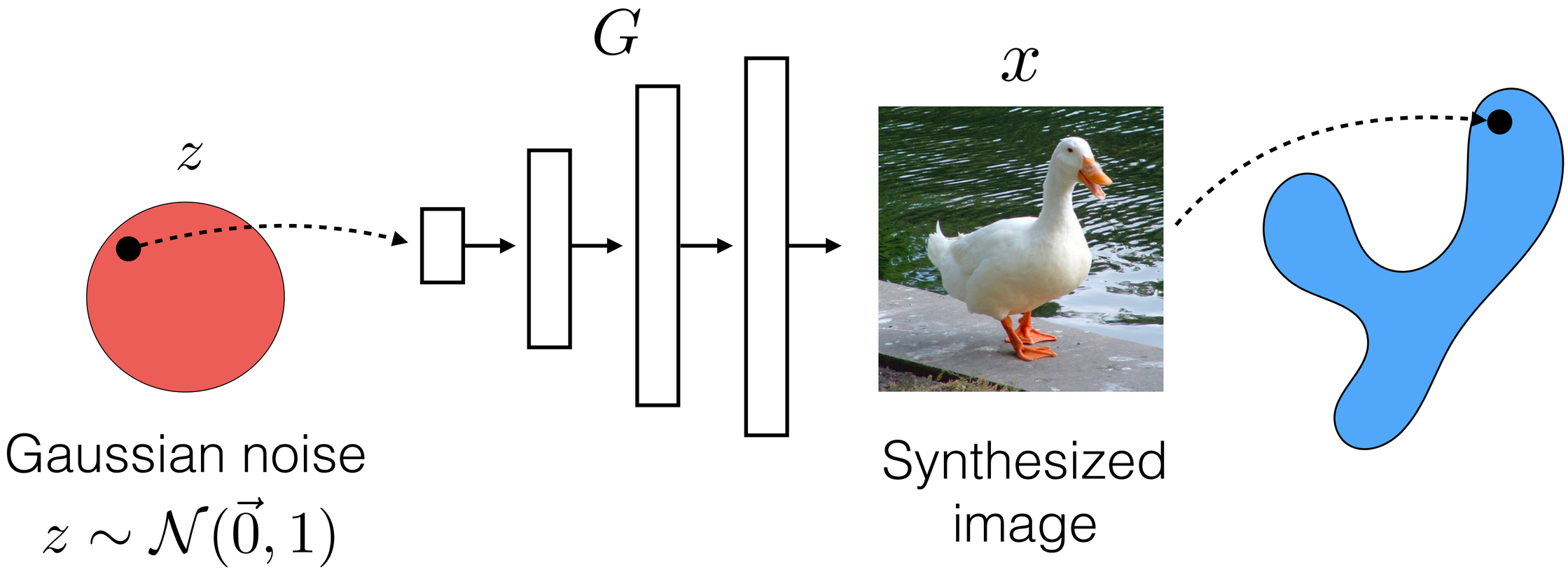
Source: Isola, Freeman, Torralba

Neural networks as distribution transformers



Source: Isola, Freeman, Torralba

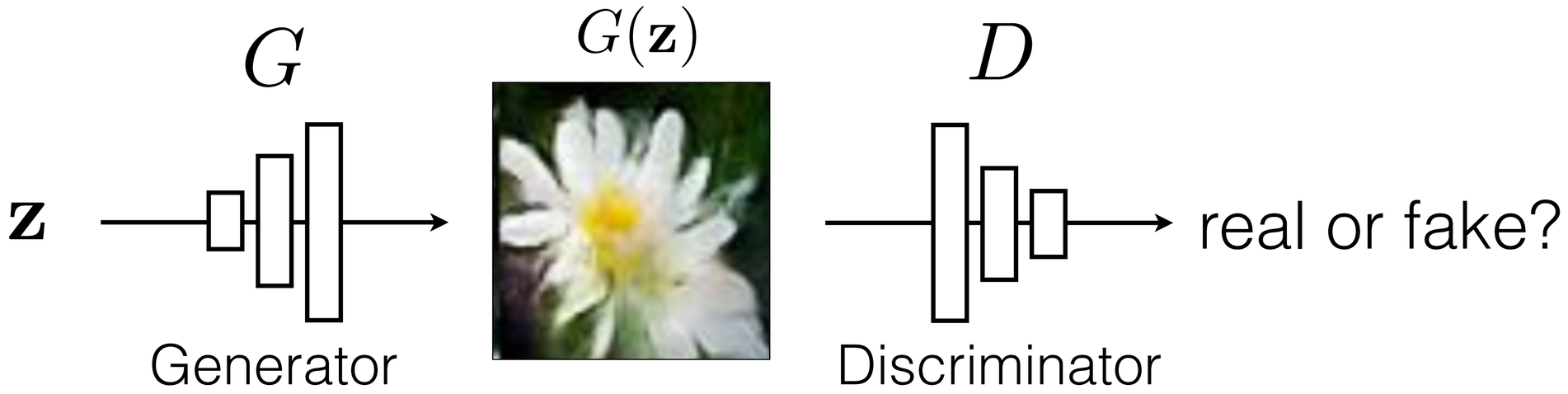
Neural networks as distribution transformers



Source: Isola, Freeman, Torralba

Today

- Texture synthesis
- **Generative adversarial nets (GANs)**
- Autoregressive models

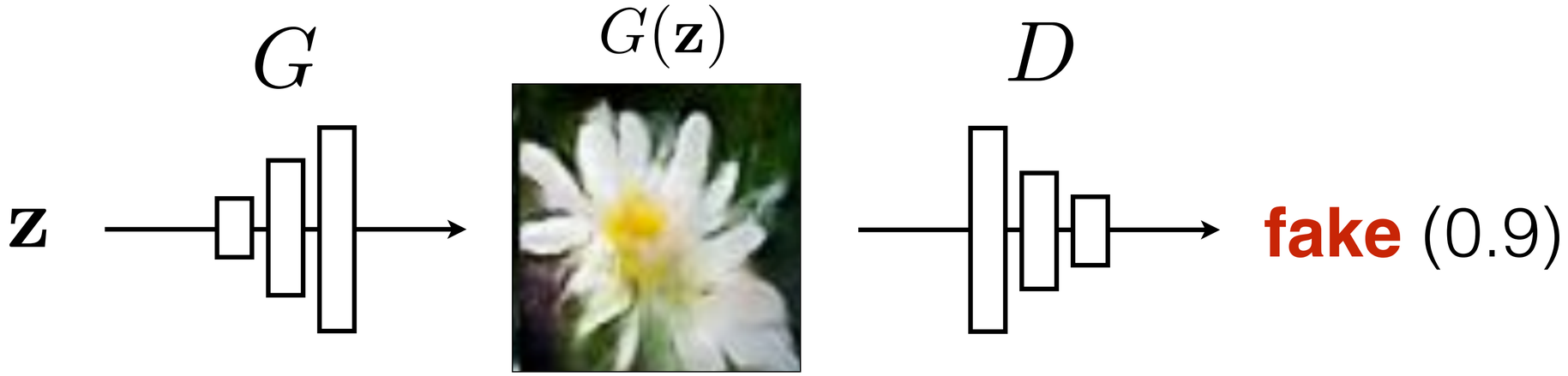


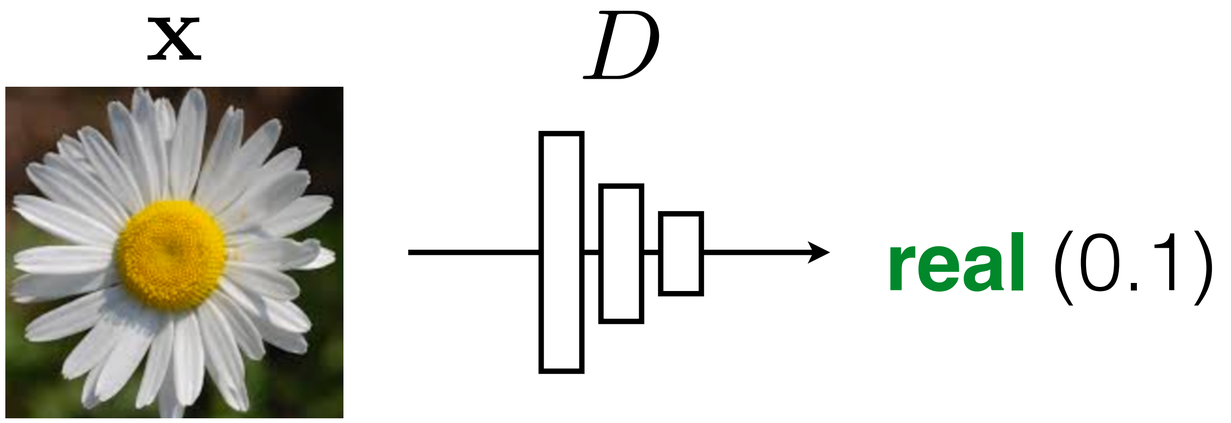
G tries to synthesize fake images that fool **D**

D tries to identify the fakes

[Goodfellow et al., 2014]

Source: Isola, Freeman, Torralba

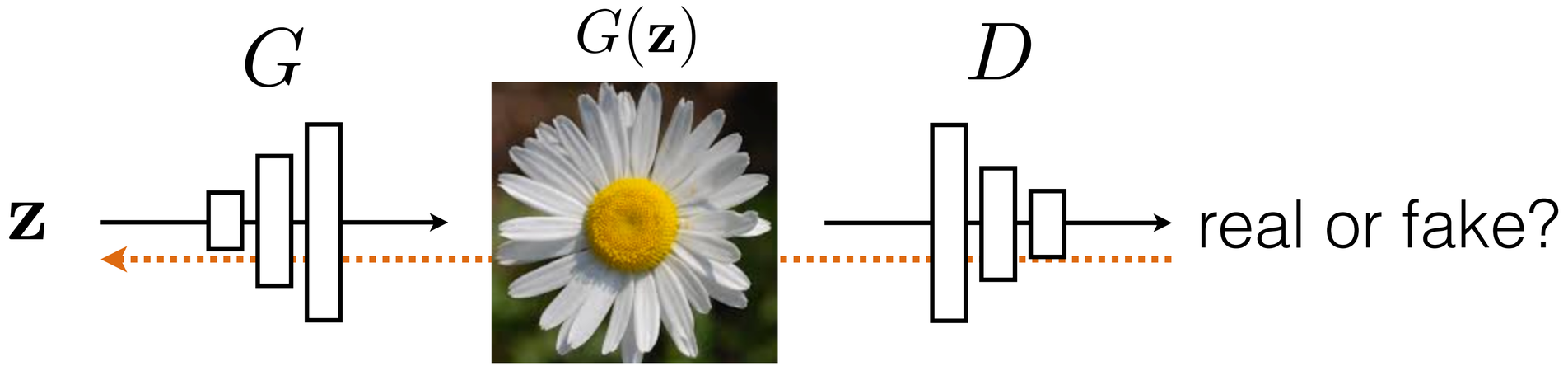




$$\arg\max_D \mathbb{E}_{\mathbf{z},\mathbf{x}}[\ \log D(G(\mathbf{z}))\ +\ \log\left(1 - D(\mathbf{x})\right)\ ]$$

[Goodfellow et al., 2014]

Source: Isola, Freeman, Torralba

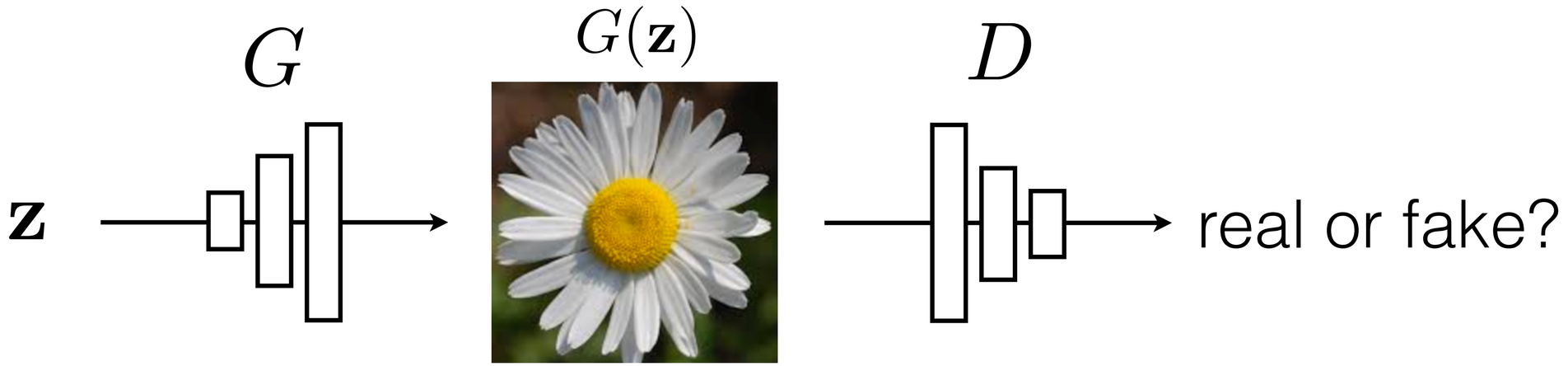


G tries to synthesize fake images that ***fool*** **D**:

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

[Goodfellow et al., 2014]

Source: Isola, Freeman, Torralba



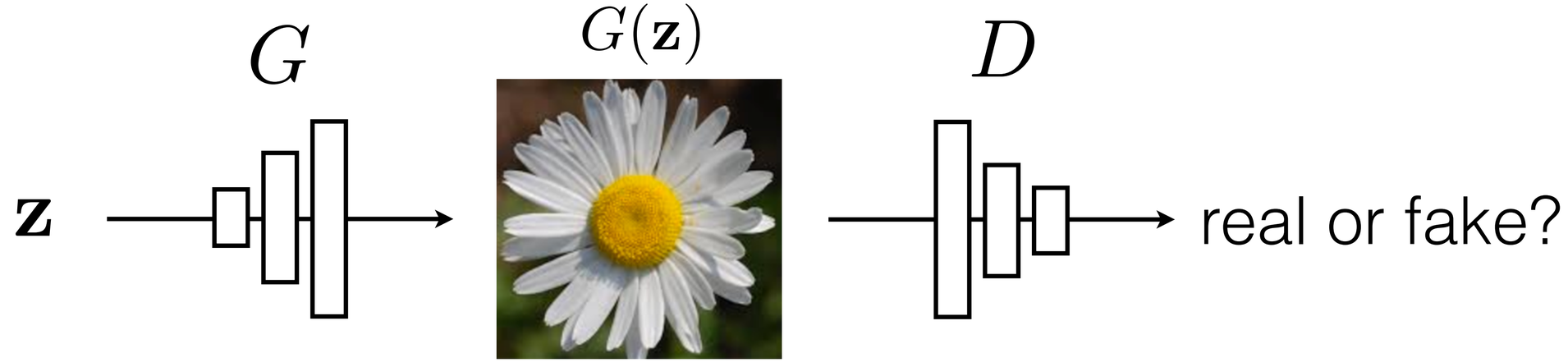
G tries to synthesize fake images that ***fool*** the ***best*** **D**:

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

[Goodfellow et al., 2014]

Source: Isola, Freeman, Torralba

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 (see book)
- Note: when training G, we take derivatives through discriminator!

[Goodfellow et al., 2014]

Source: Isola, Freeman, Torralba

Samples from BigGAN

[Brock et al. 2018]

More here: https://arxiv.org/pdf/1809.11096.pdf

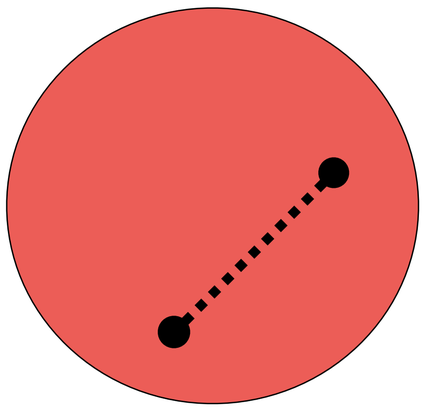
Source: Isola, Freeman, Torralba

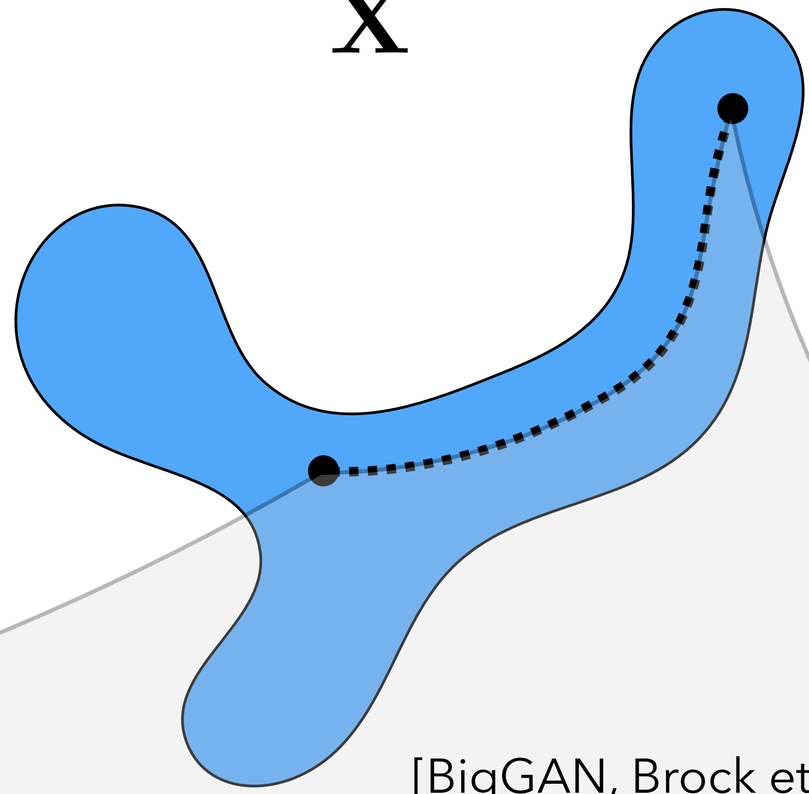
Latent space
(Gaussian)

$\mathbf{z}$

Data space
(Natural image manifold)

$\mathbf{x}$





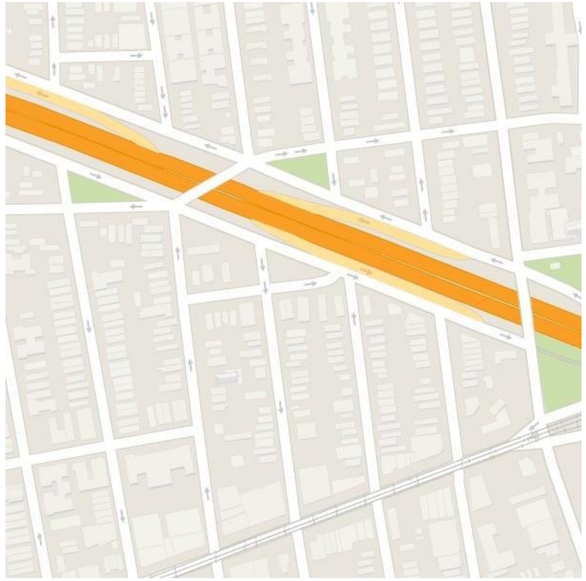
[BigGAN, Brock et al. 2018]

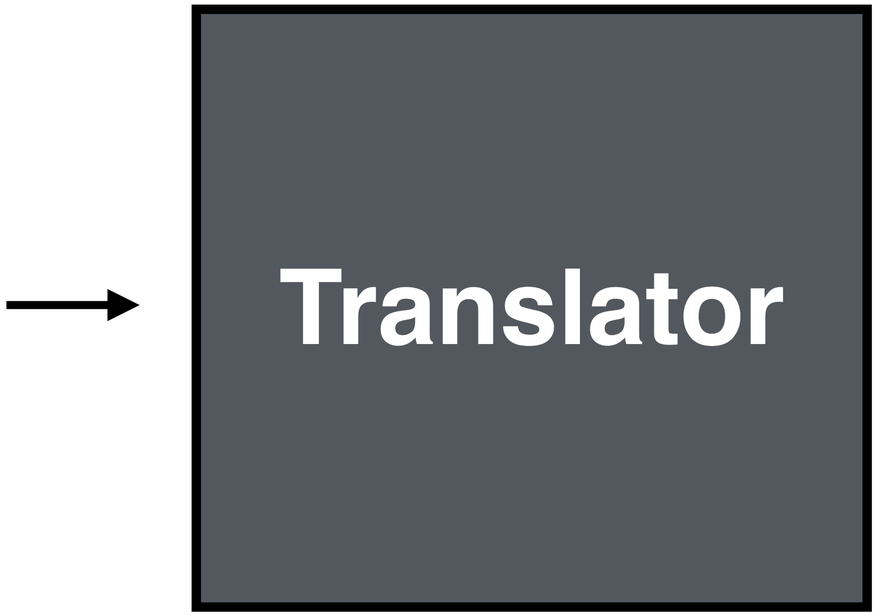
https://github.com/NVlabs/stylegan3 [Karras et al., "Alias-Free Generative Adversarial Networks", 2021

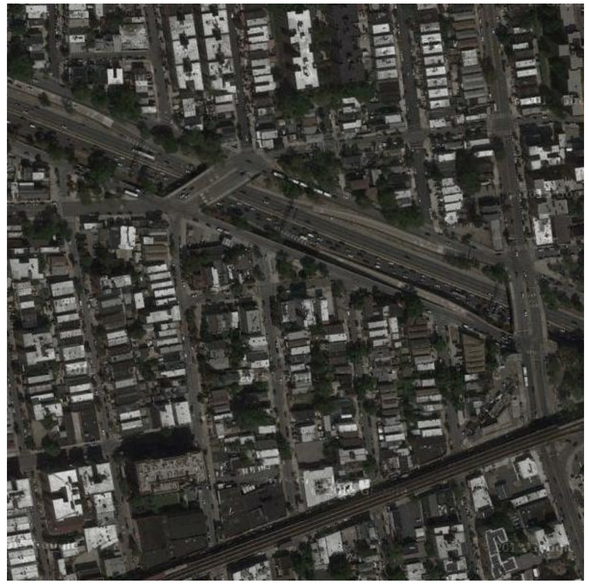
[Karras et al., “Alias-Free Generative Adversarial Networks”, 2021

[Karras et al., “Alias-Free Generative Adversarial Networks”, 2021

Image translation







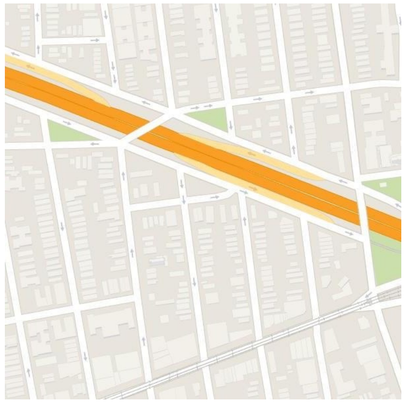
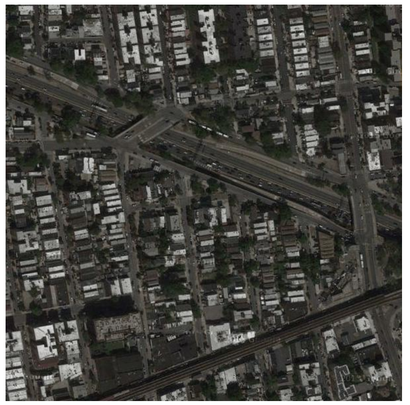
Google Map

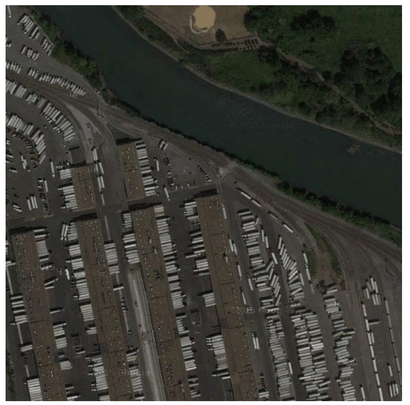
Satellite photo

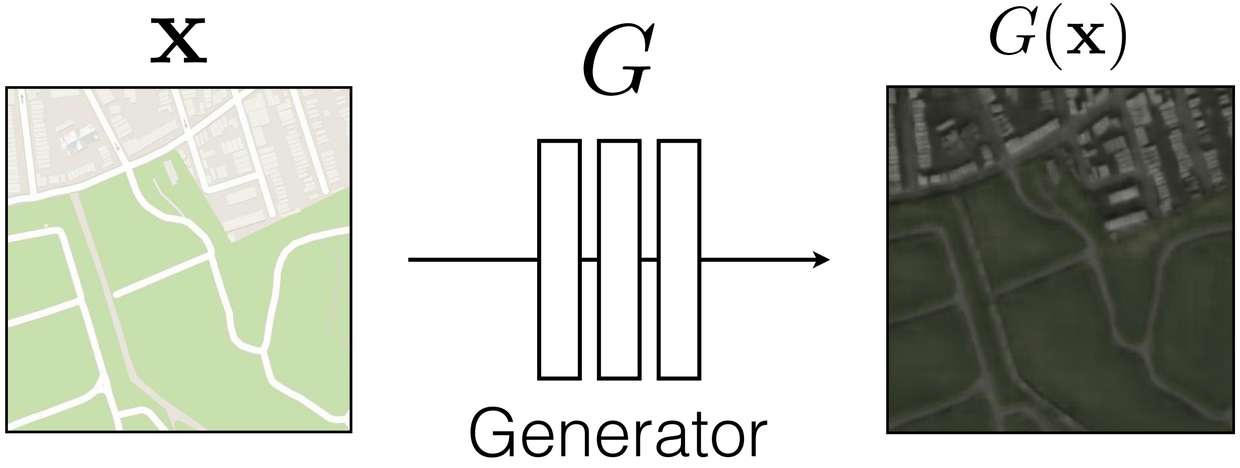
Source: Isola, Freeman, Torralba

Map2Sat

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

{  ,  }

{  ,  }

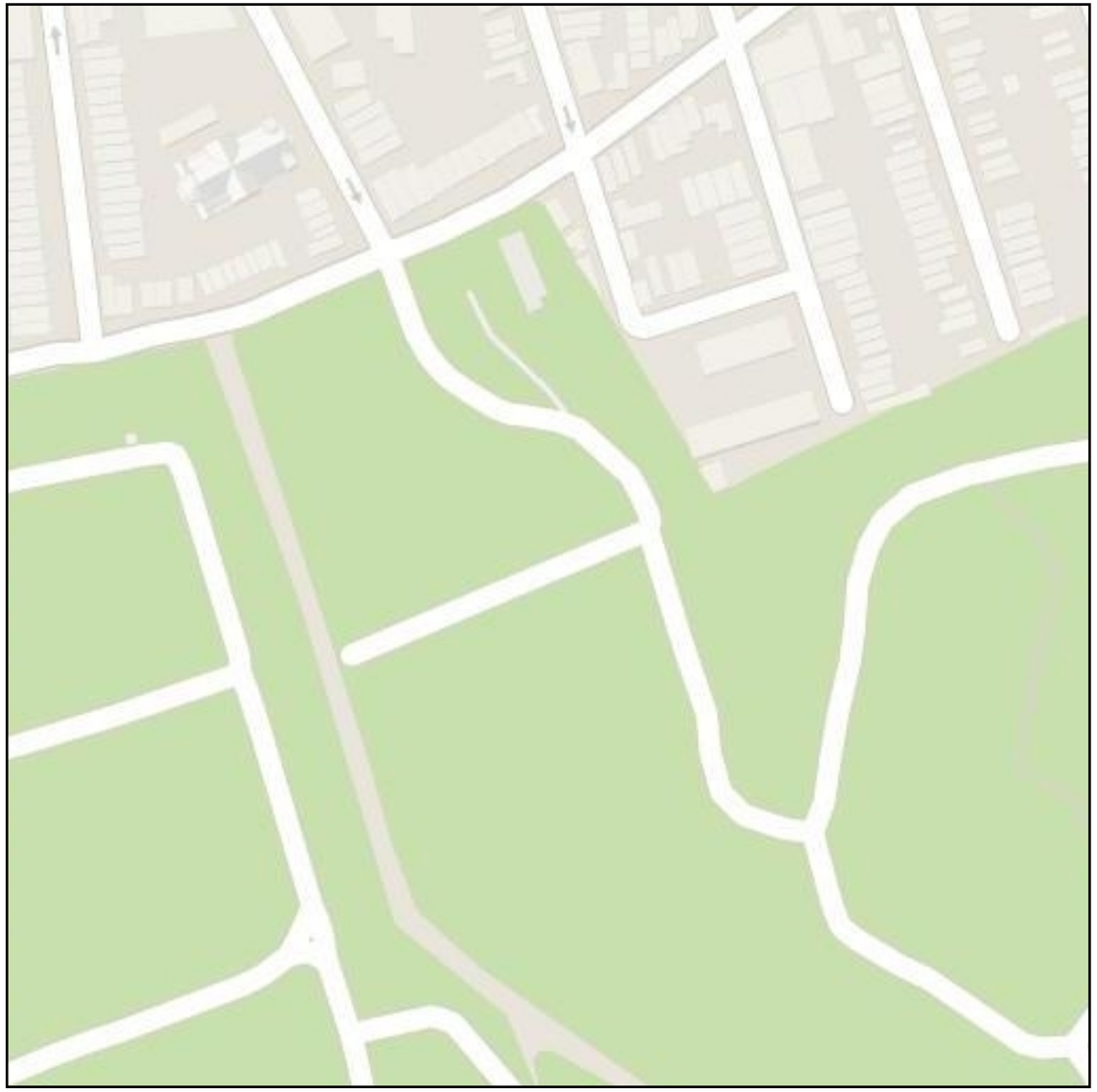


Idea: L1 loss

$$\|G(\mathbf{x}) - \mathbf{y}\|_1$$

Source: Isola, Freeman, Torralba

Input

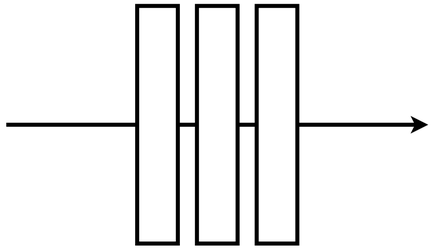


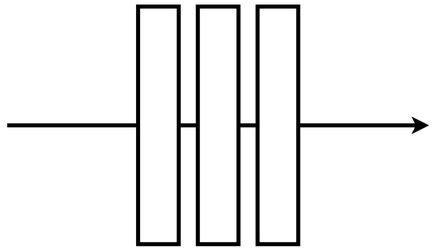
L1 loss

Source: Isola, Freeman, Torralba

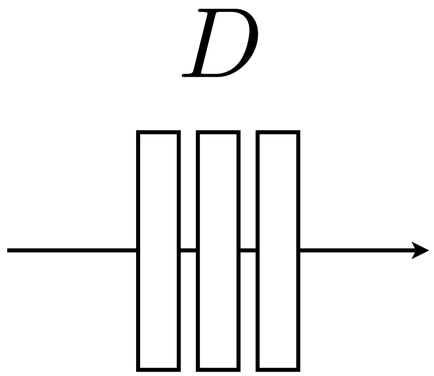
G tries to synthesize fake images that fool **D**

D tries to identify the fakes





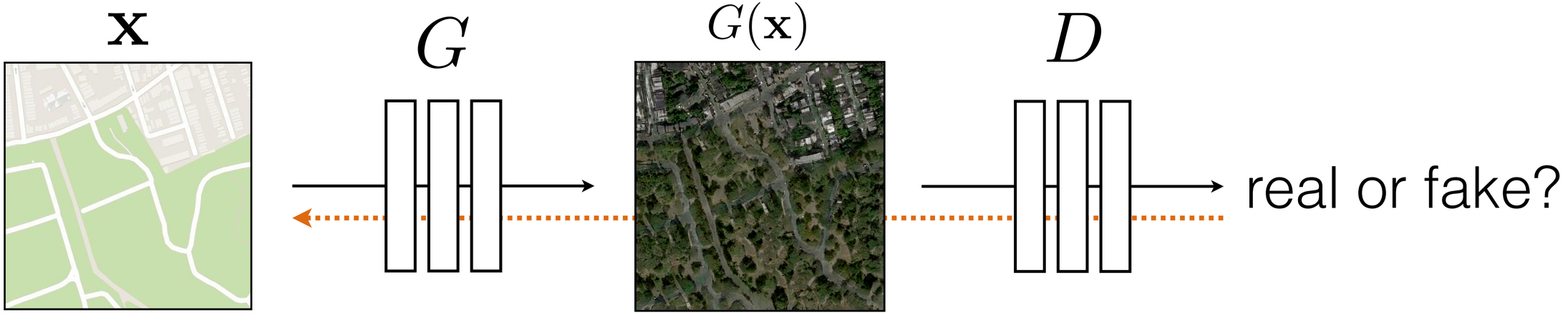
fake (0.9)



real (0.1)

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

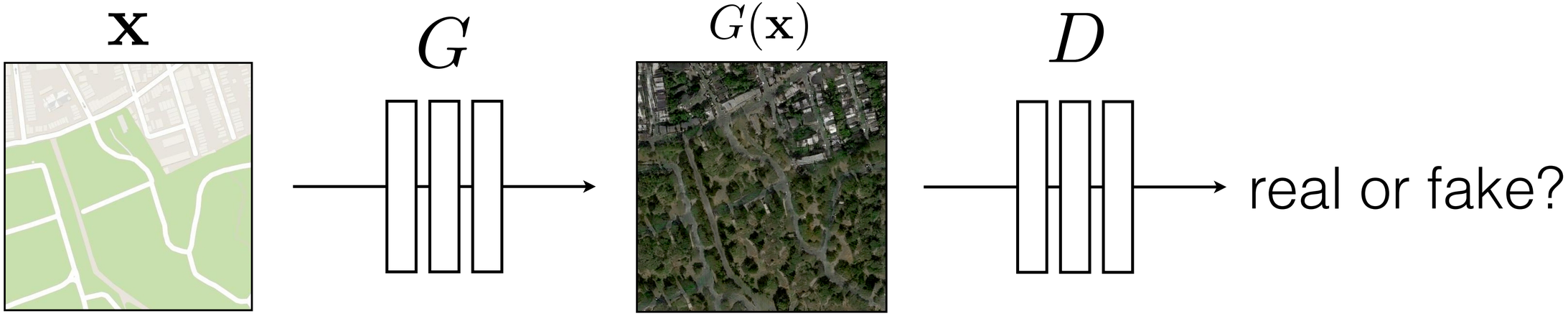
Source: Isola, Freeman, Torralba



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

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

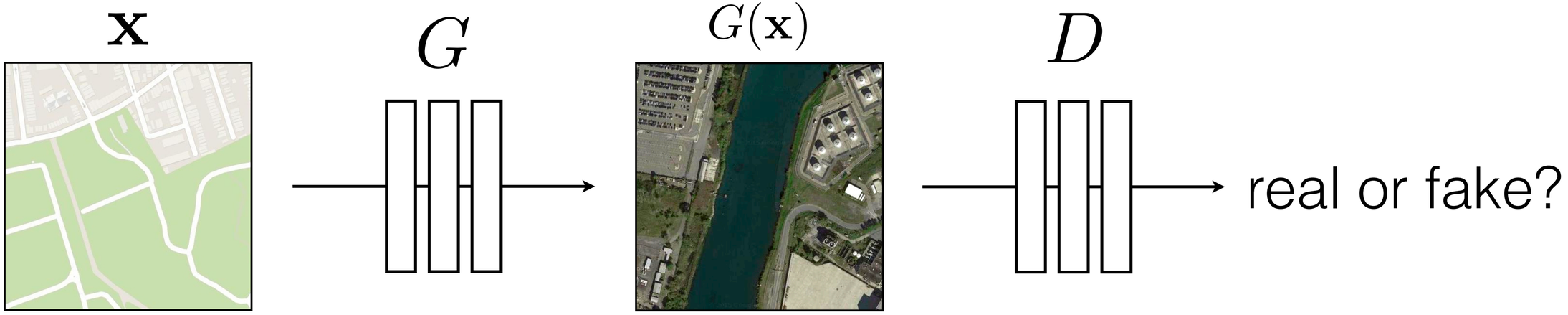
Source: Isola, Freeman, Torralba



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

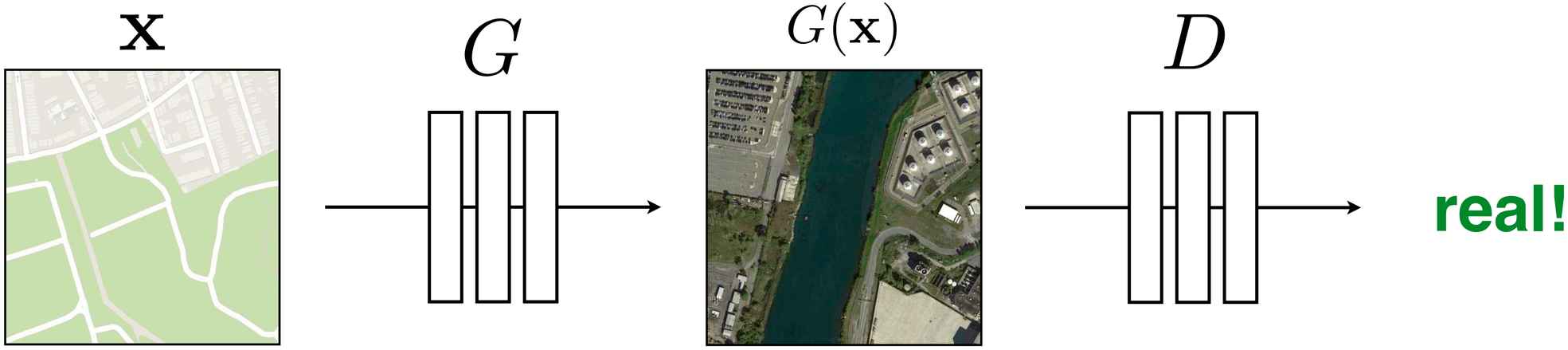
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ ]$$

Source: Isola, Freeman, Torralba



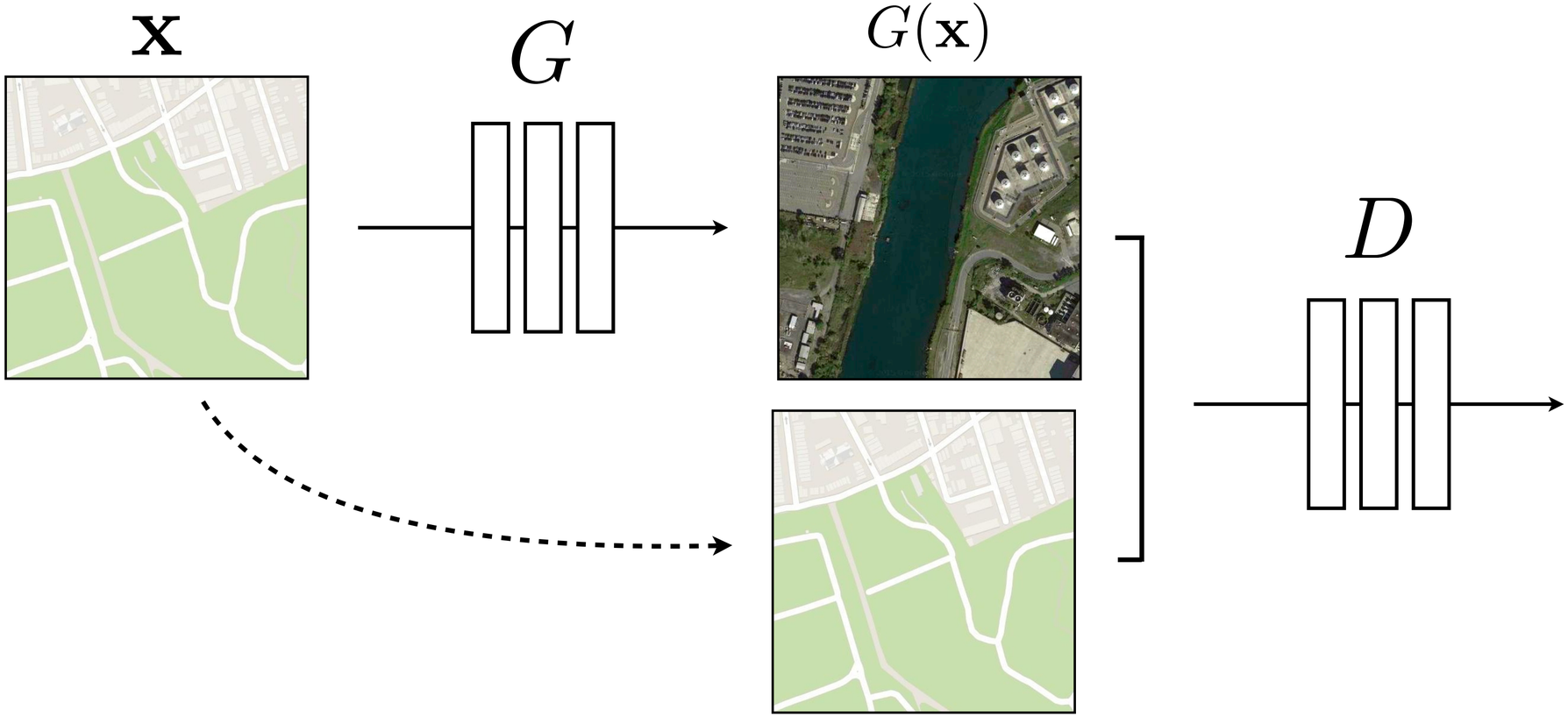
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y})) \;\;]$$

Source: Isola, Freeman, Torralba



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

Source: Isola, Freeman, Torralba



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

Source: Isola, Freeman, Torralba

$\mathbf{x}$

G

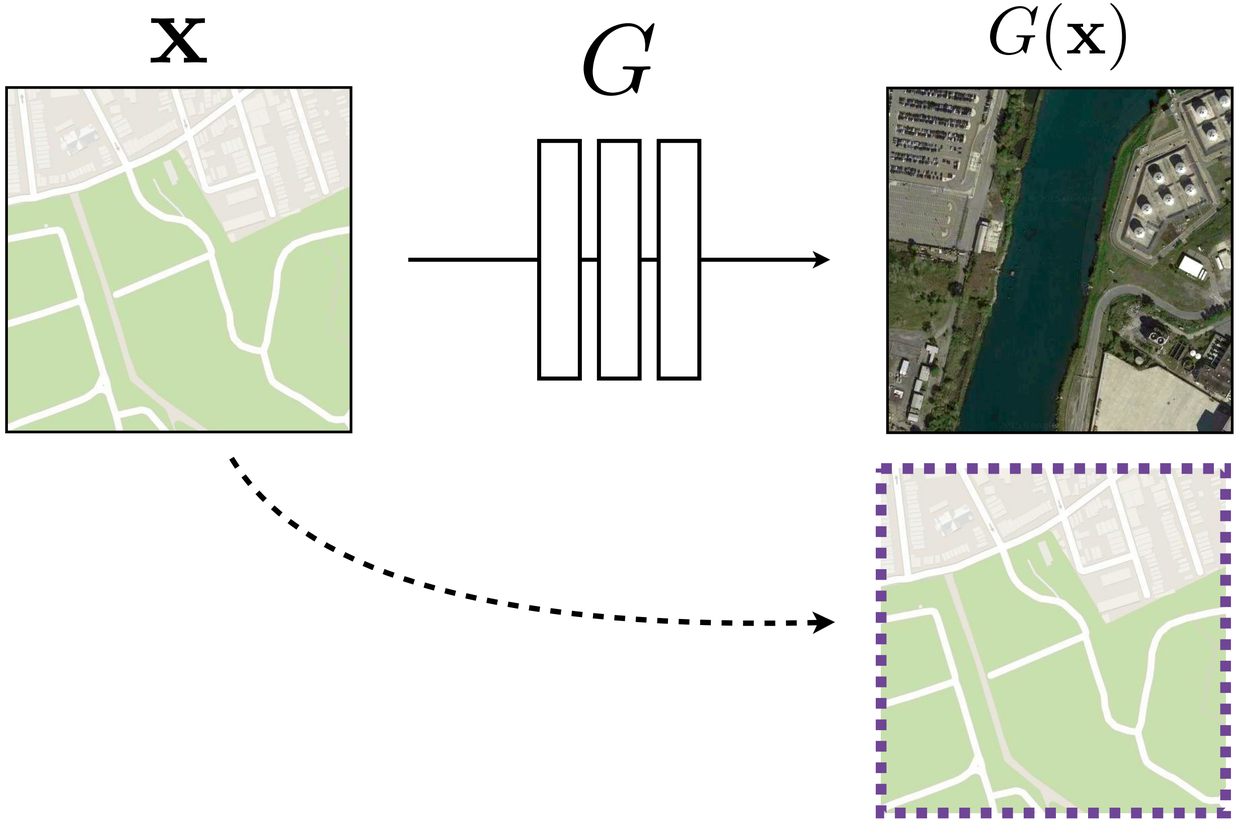
$G(\mathbf{x})$

D

real or fake *pair* ?

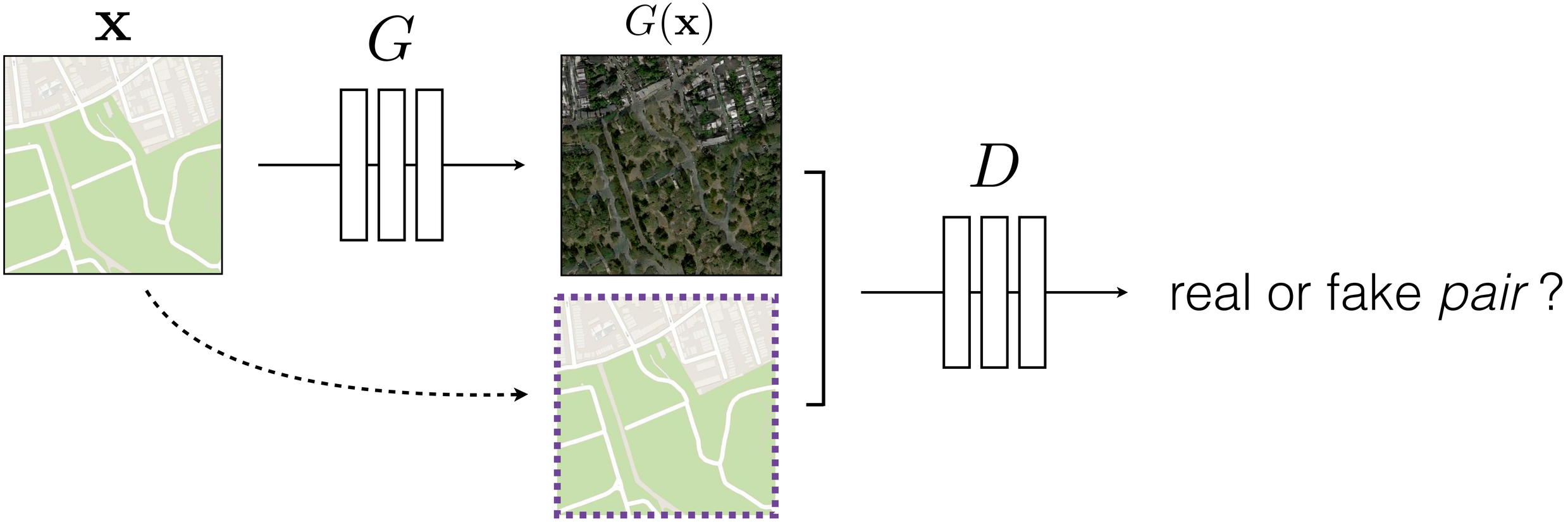
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;]$$

Source: Isola, Freeman, Torralba



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

Source: Isola, Freeman, Torralba



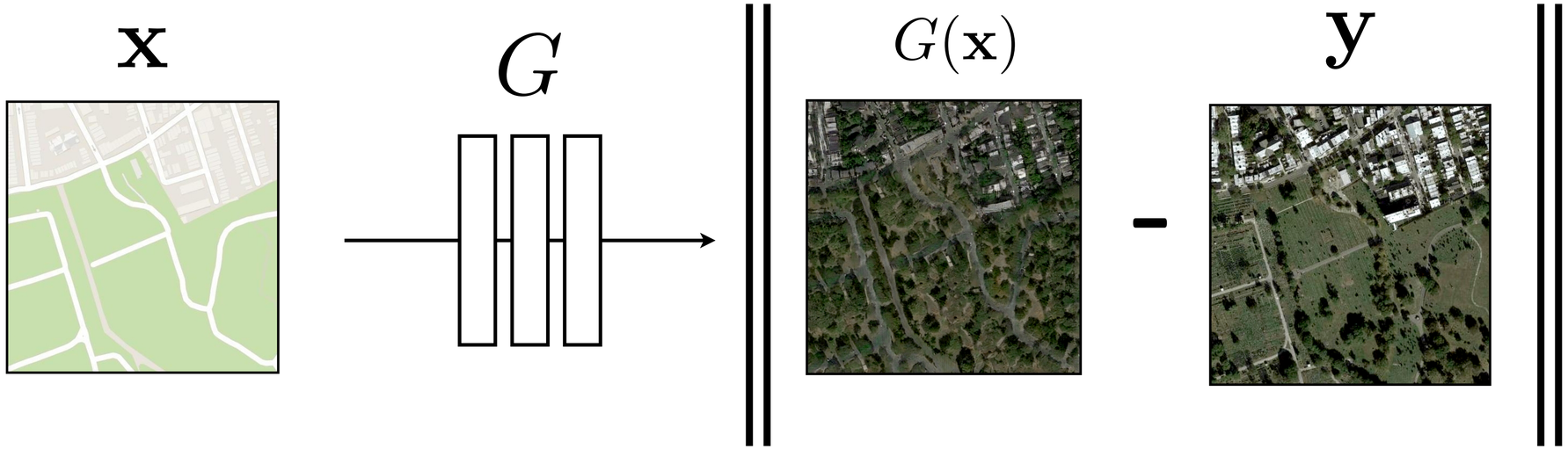
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\;\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;\;]$$

Source: Isola, Freeman, Torralba

Training Details: Loss function

Conditional GAN

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

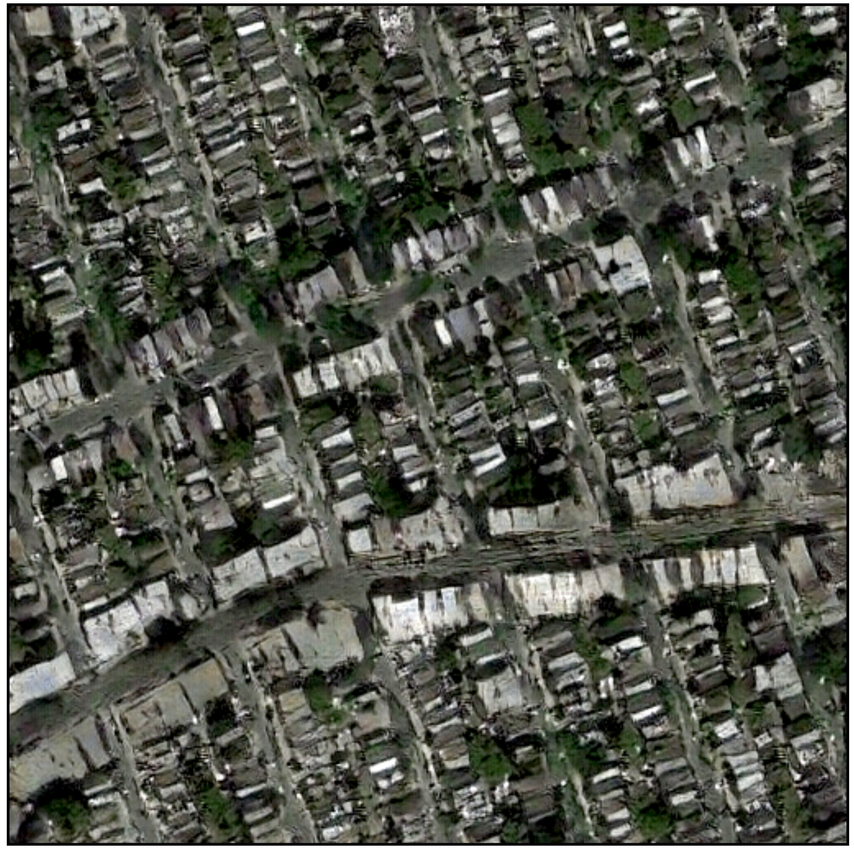


Pixel-wise loss helps stabilize training + faster convergence

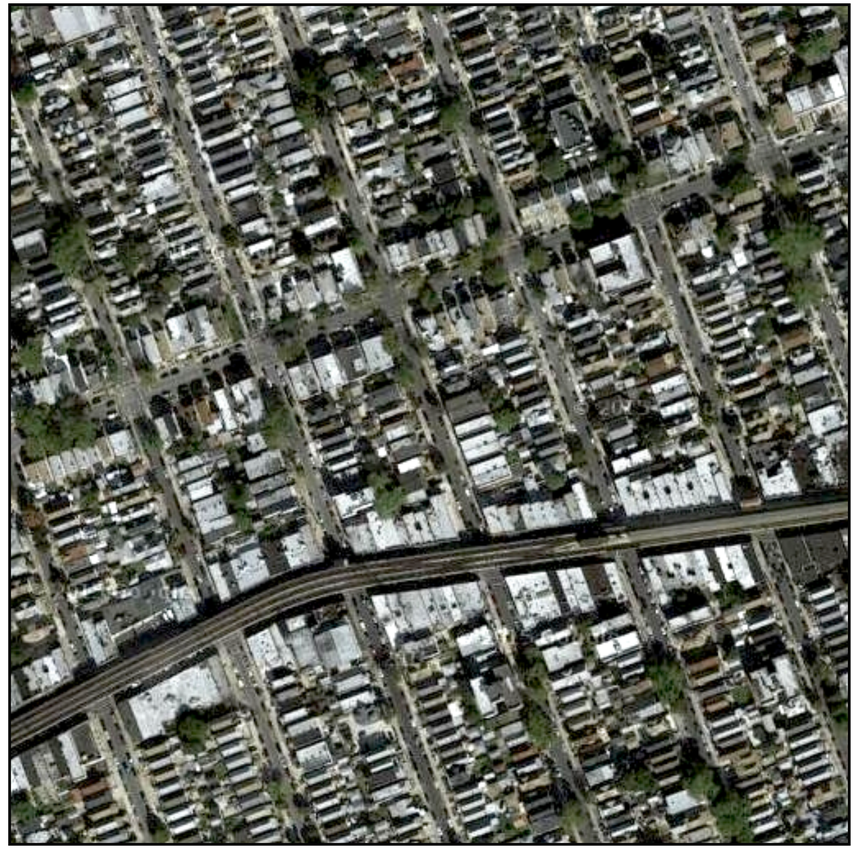
Source: Isola, Freeman, Torralba

Input

Output



Groundtruth



Data from
[maps.google.com]

Source: Isola, Freeman, Torralba

Input

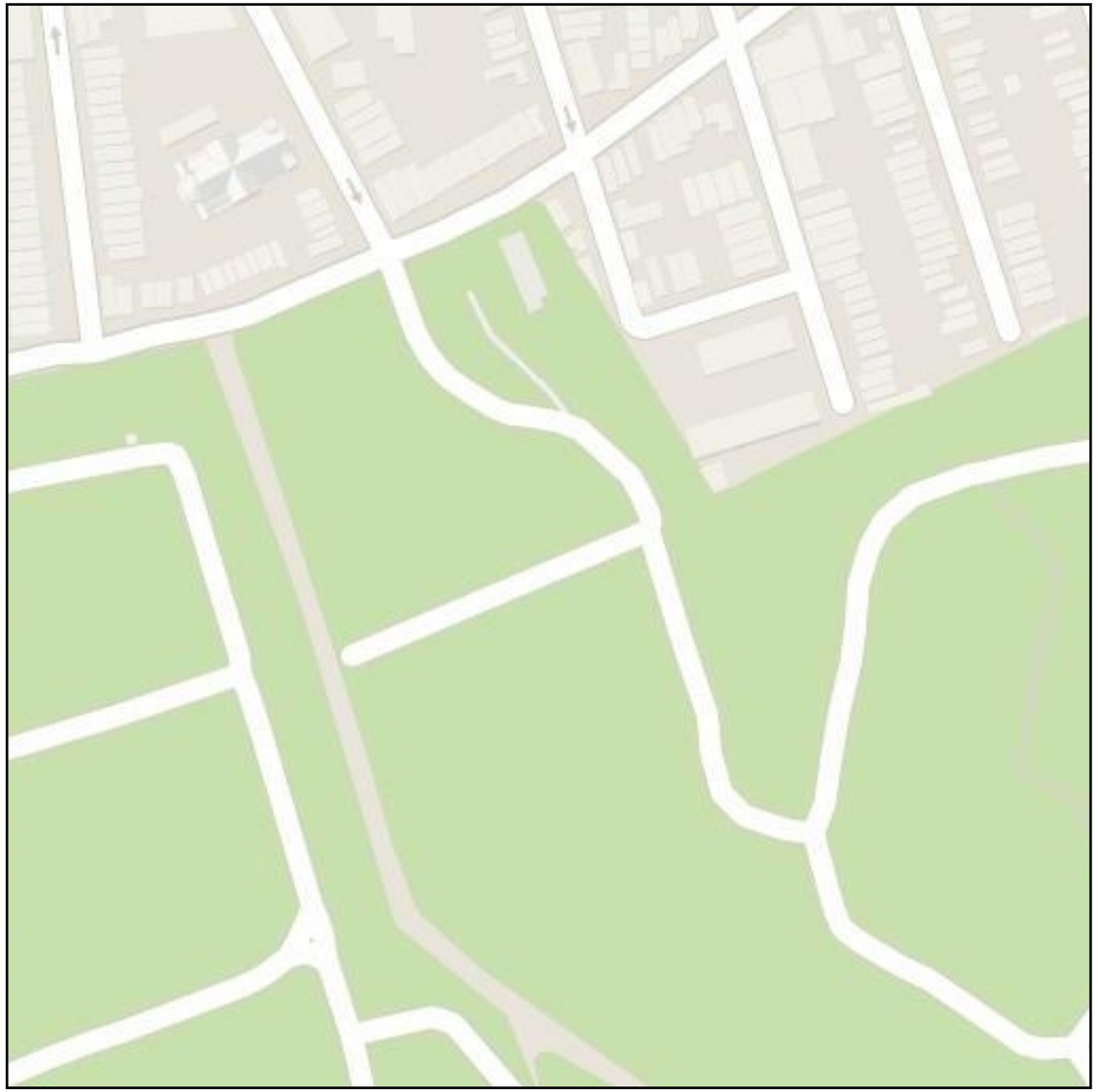
Output

Groundtruth

Data from [maps.google.com]

Source: Isola, Freeman, Torralba

Input

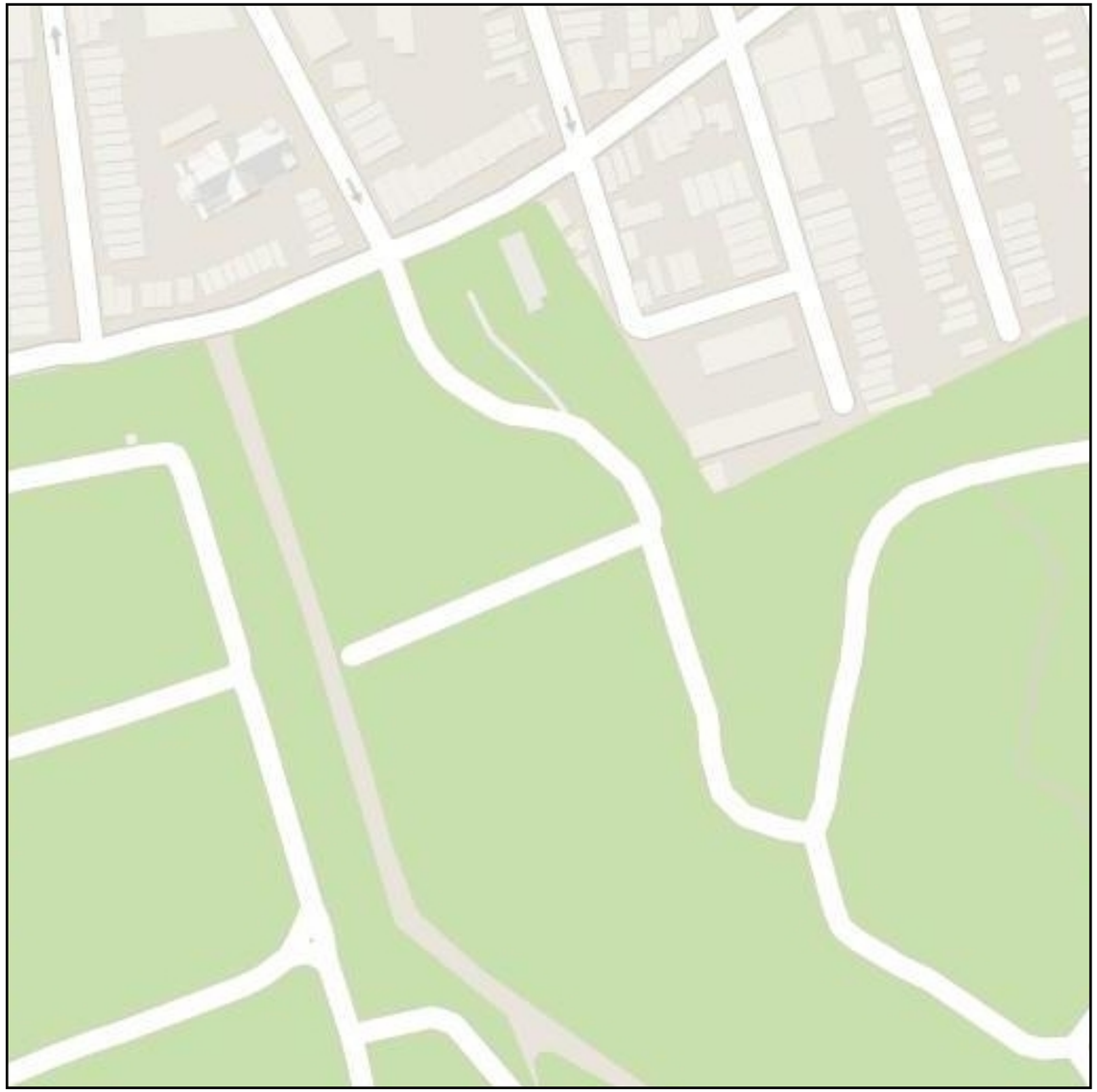


L1 loss only

Source: Isola, Freeman, Torralba

Input

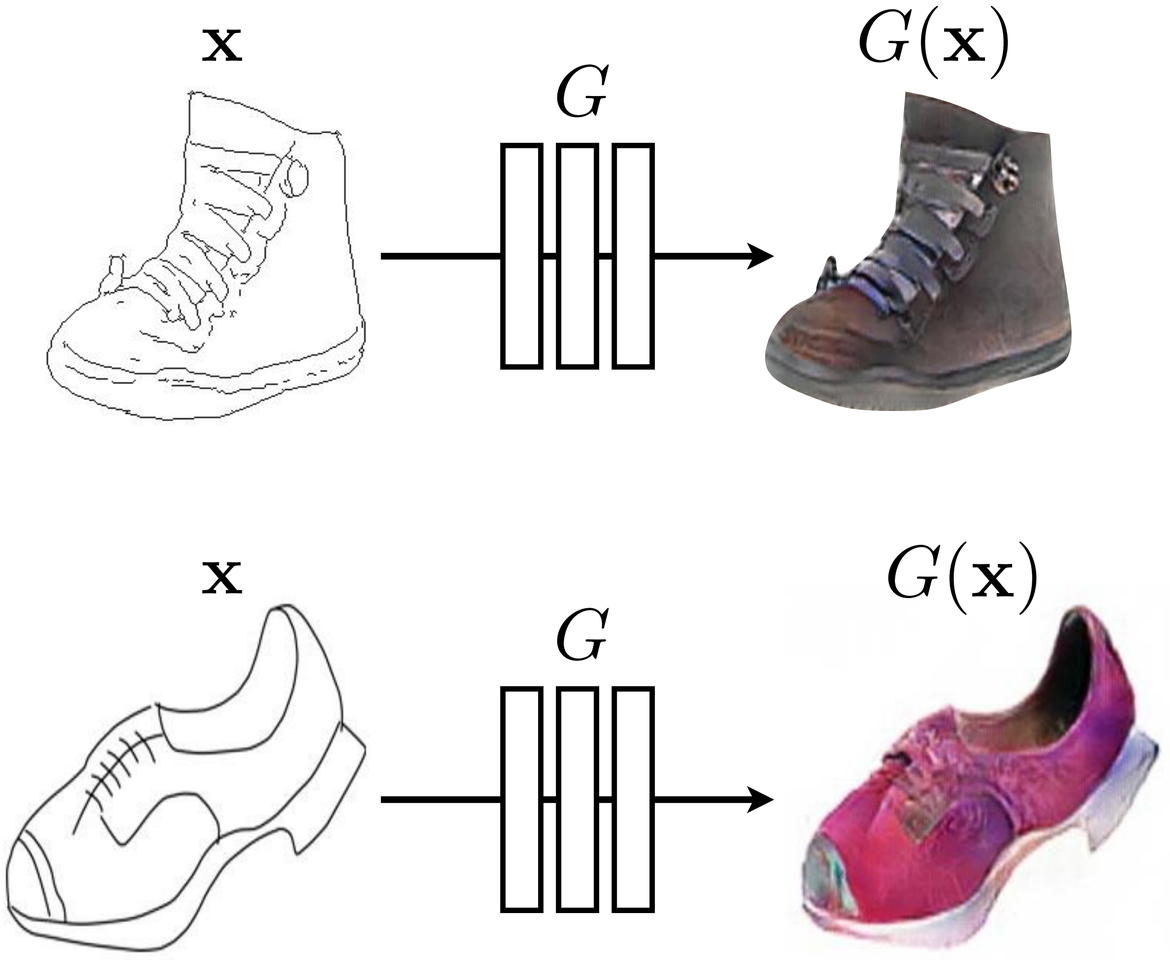
L1 loss + discriminator



Source: Isola, Freeman, Torralba

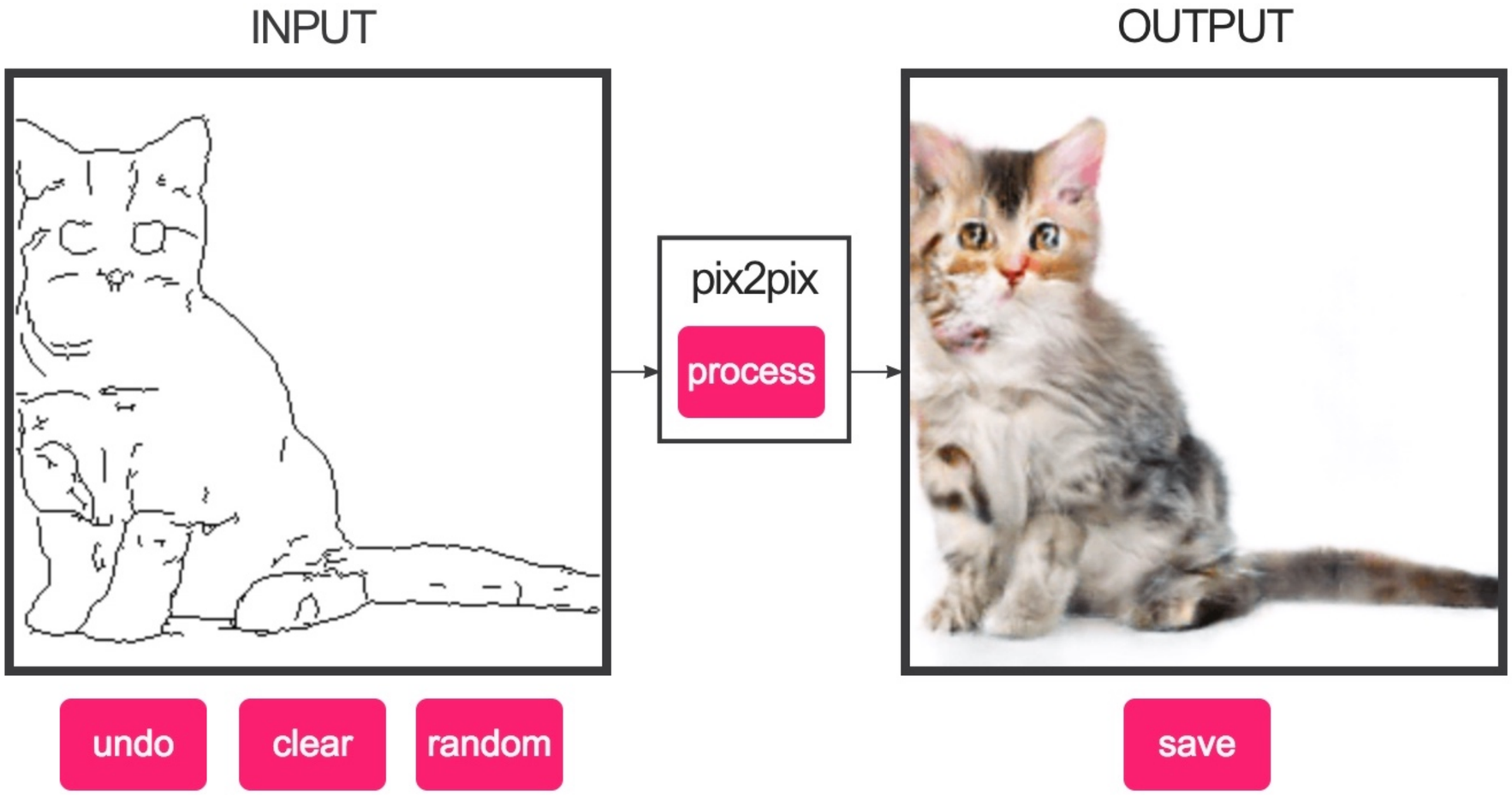
Training data

[HED, Xie & Tu, 2015]



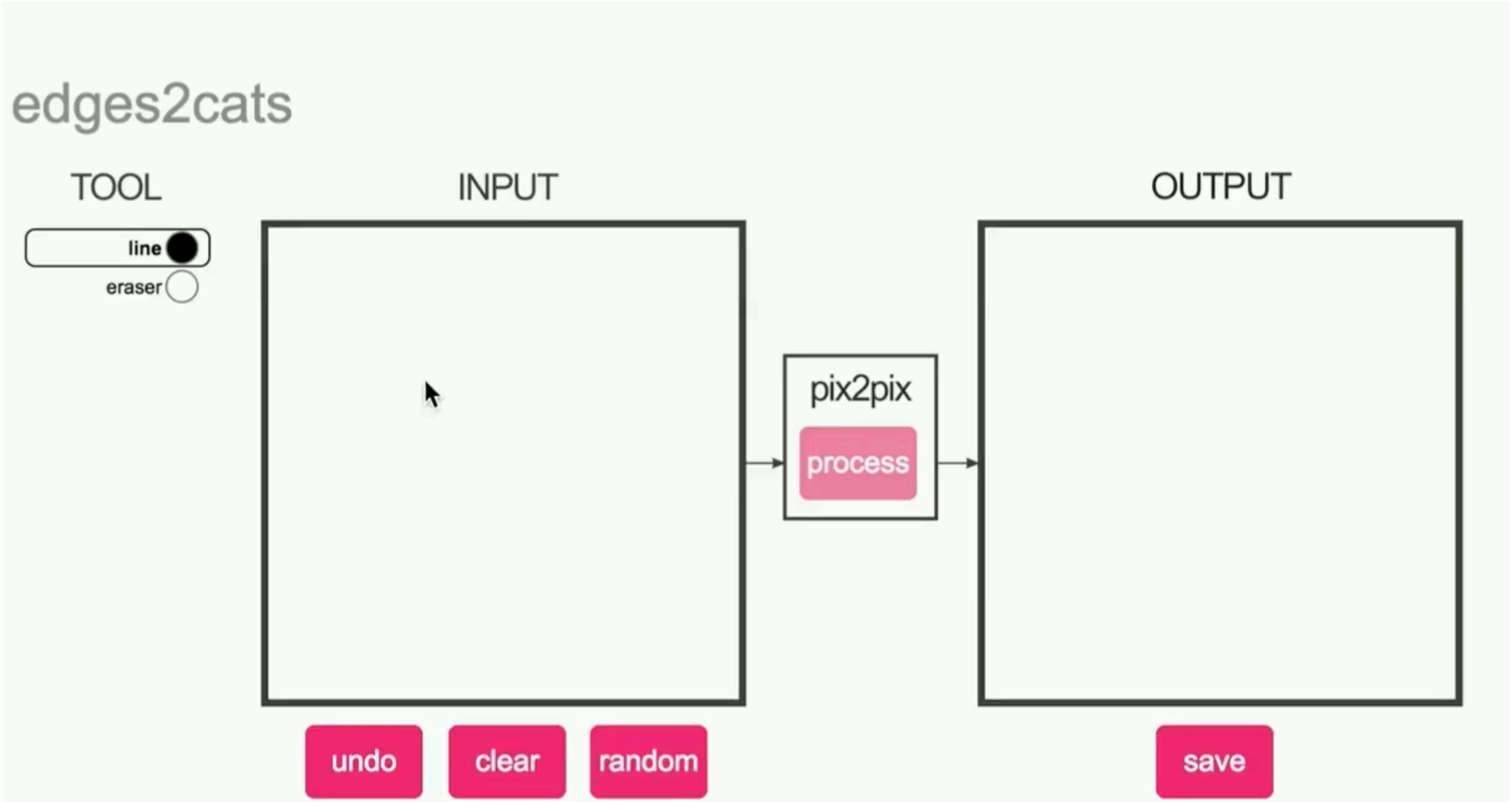
Source: Isola, Freeman, Torralba

edges2cats

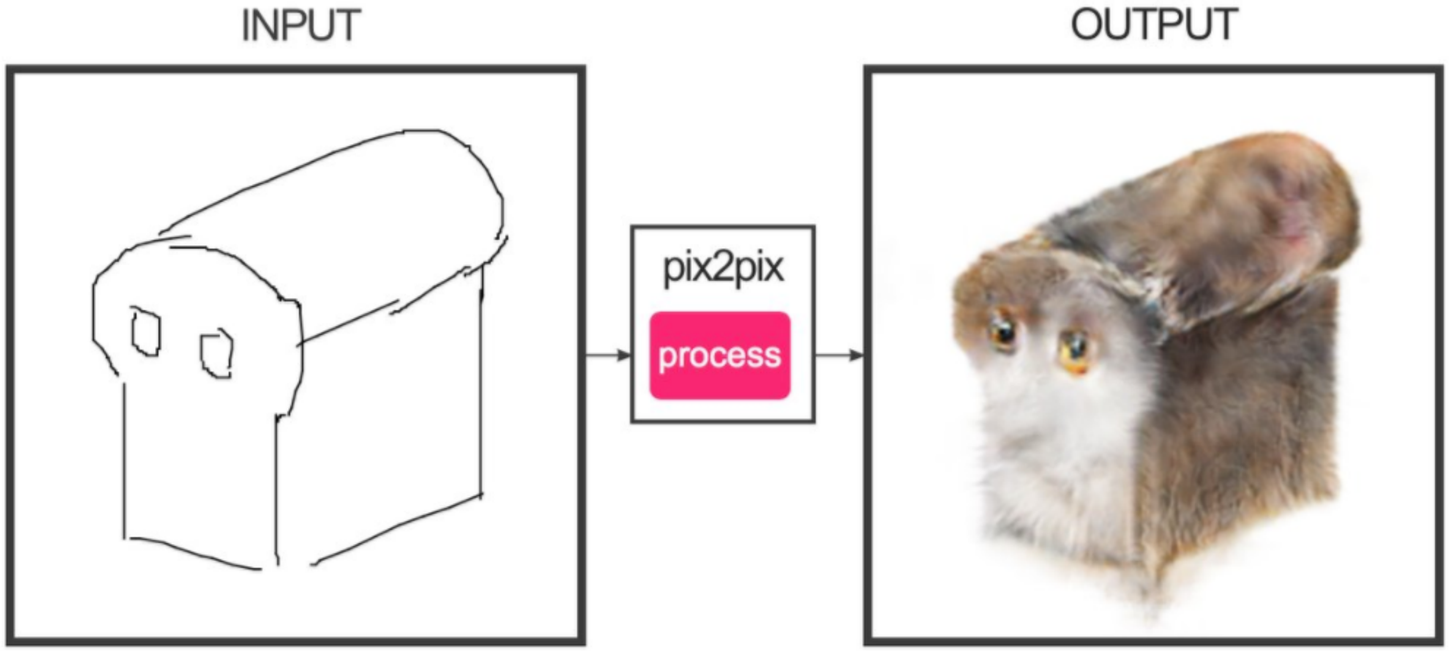


[Chris Hess, edges2cats]

Source: Isola, Freeman, Torralba



Source: Isola, Freeman, Torralba



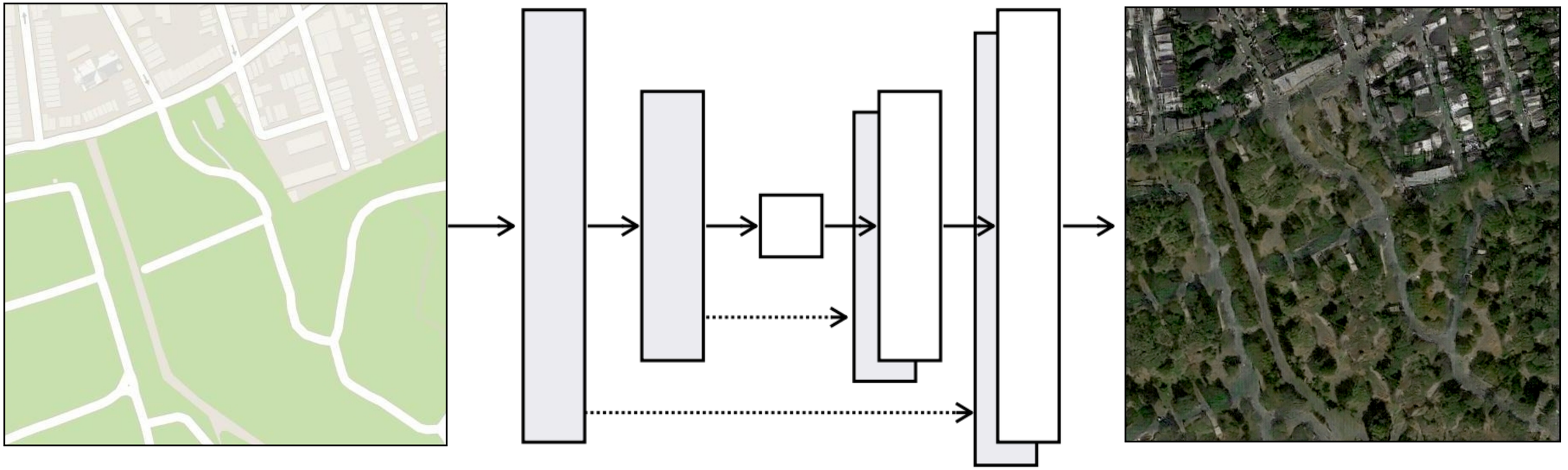
Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

Source: Isola, Freeman, Torralba

Architectures

Generator: U-Net

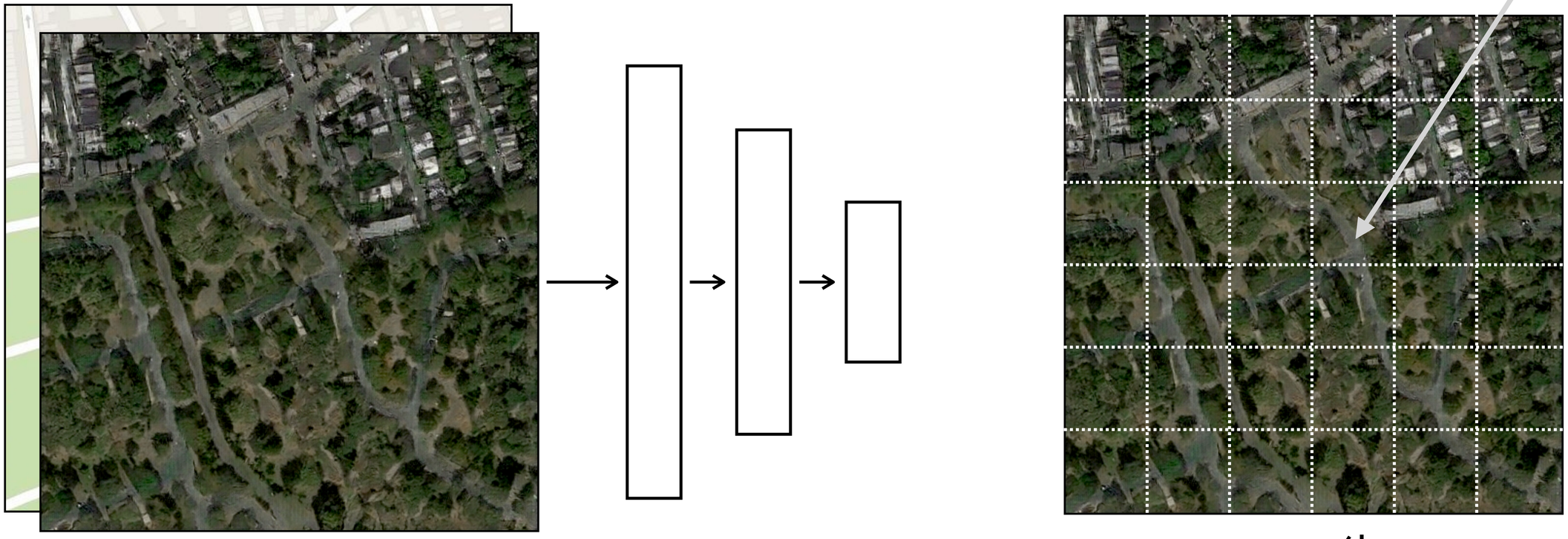


Skip connections between encoder and decoder layers

Figure from [Isola et al., “Image-to-Image Translation with Conditional Adversarial Networks”, 2017]

Architectures

Discriminator: fully convolutional network

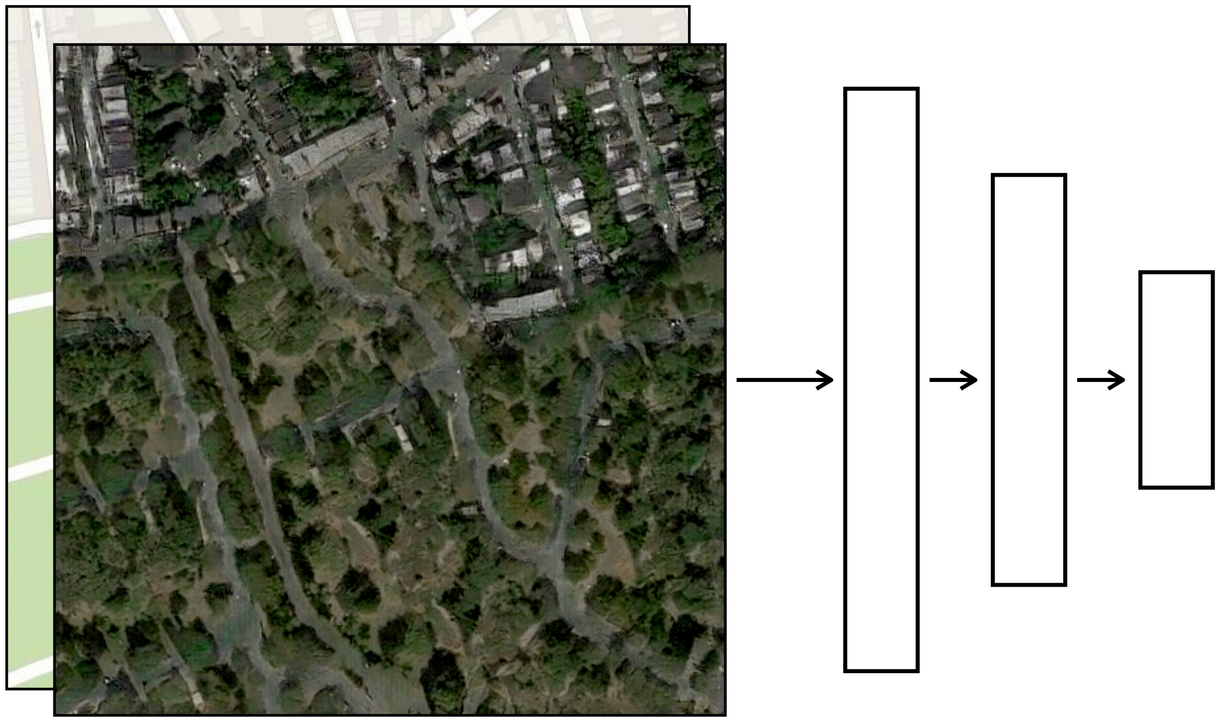


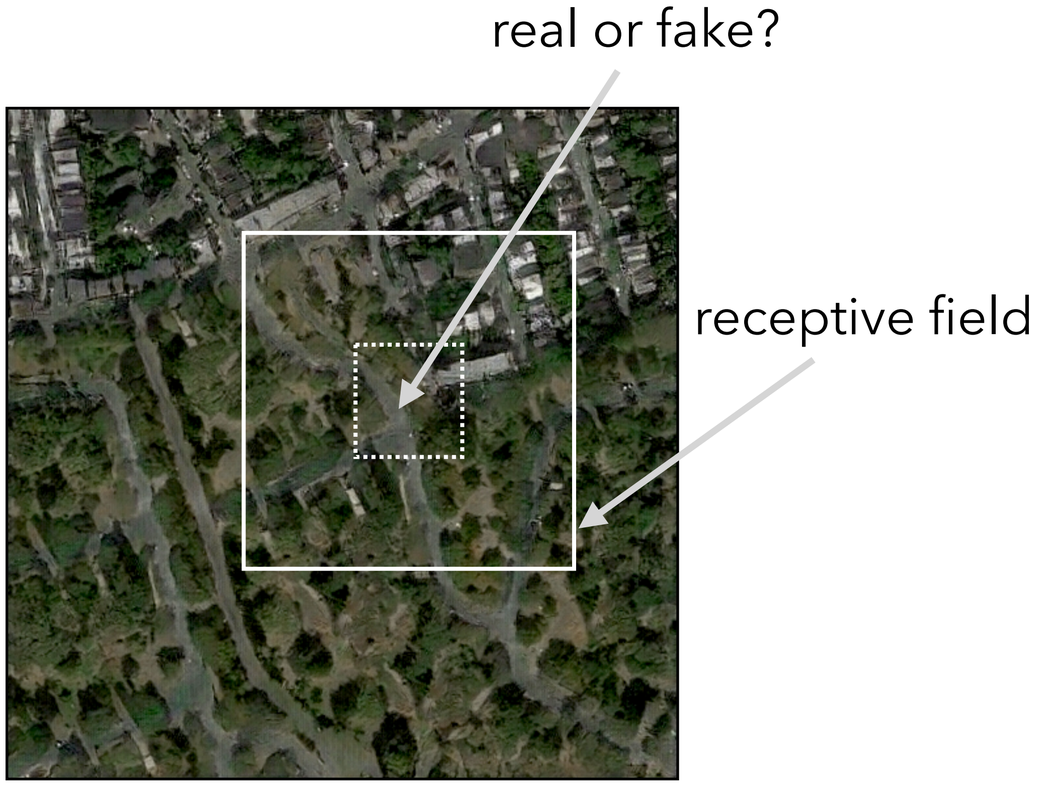
$n \times n$ output map (last conv. layer)

Sequence of strided convolutions

Architectures

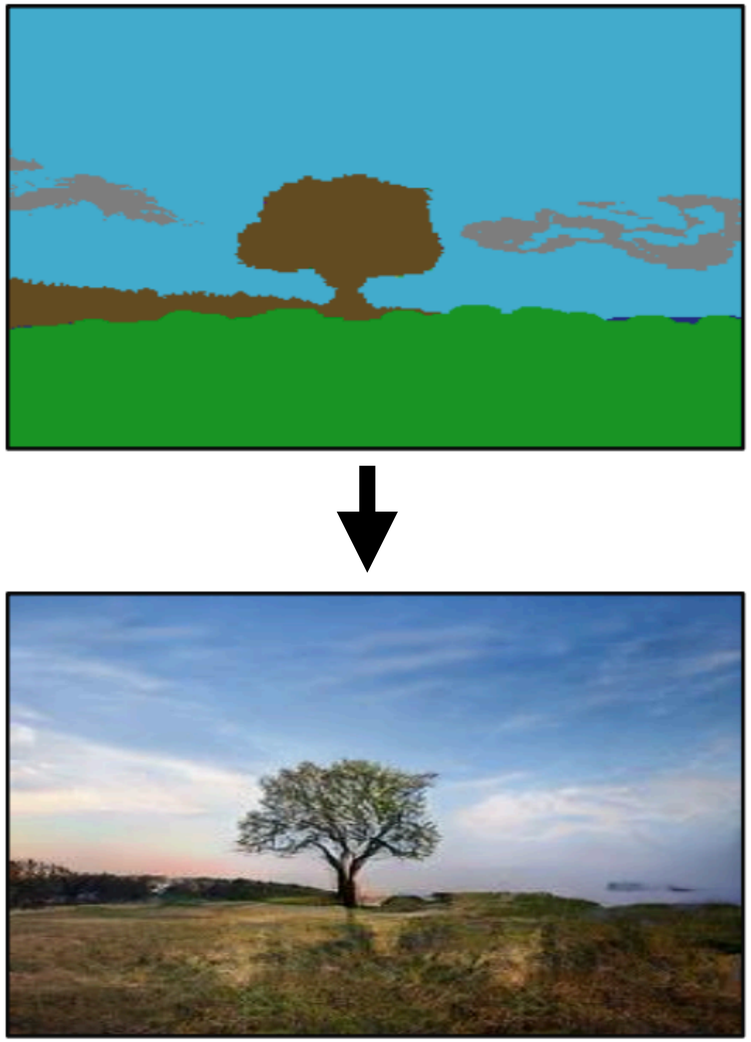
Discriminator: fully convolutional network

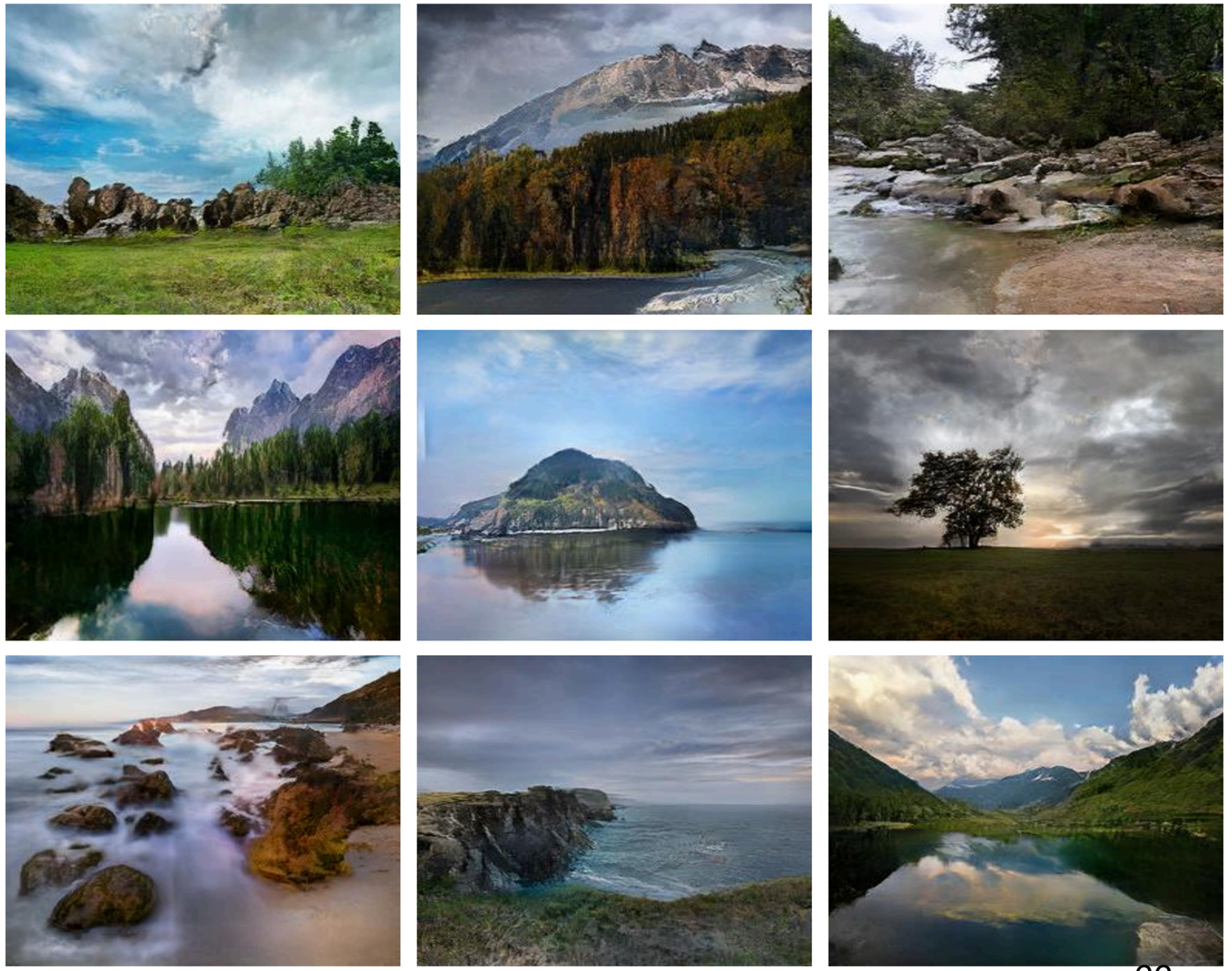




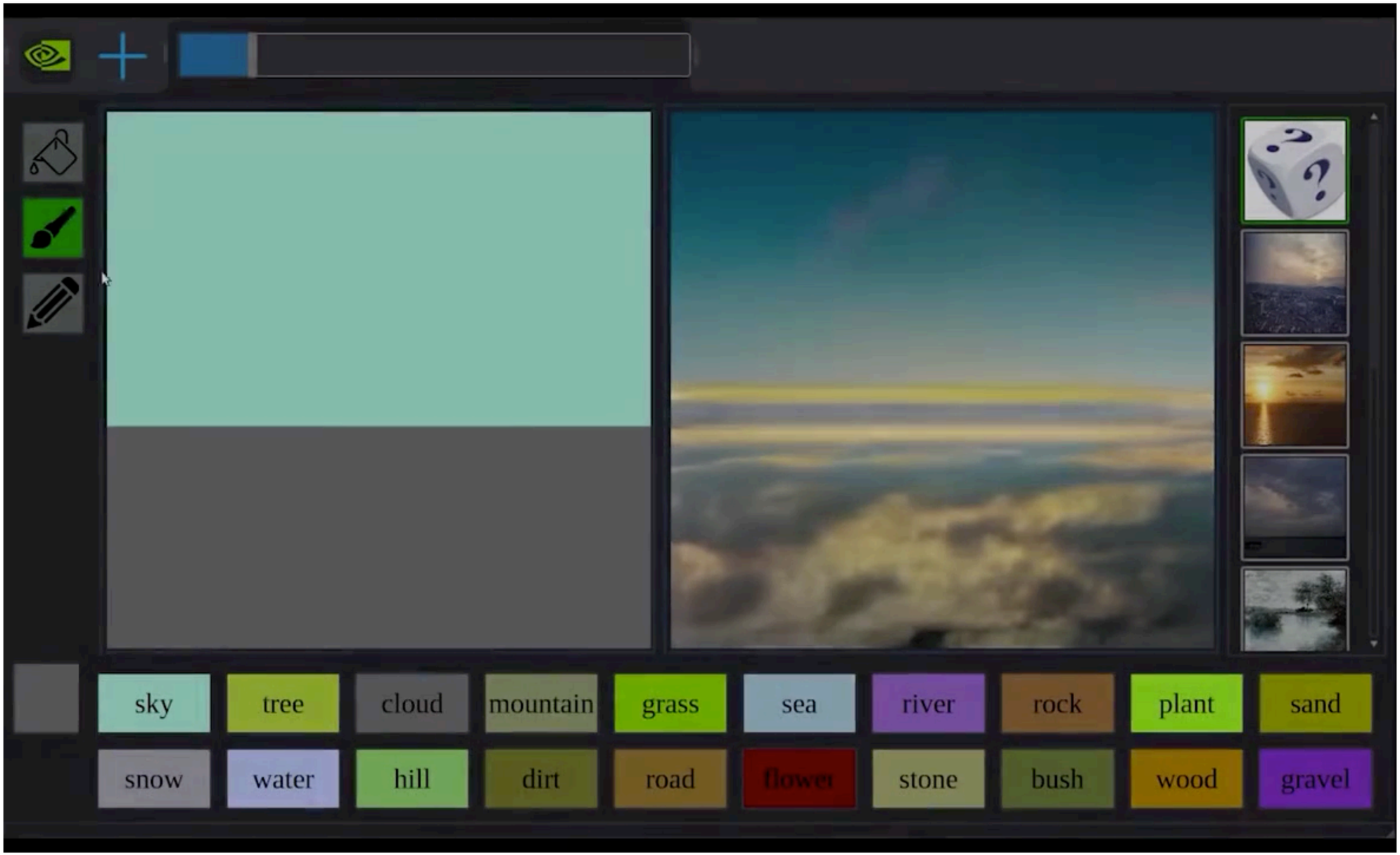
Sometimes called a Patch GAN, since it effectively only looks at patches

More recent architectures





["GauGAN", Park et al., CVPR 2019]



["GauGAN", Park et al., CVPR 2019]

What's wrong with GANs?

Category: fish

[Brock et al., "BigGAN", 2018]

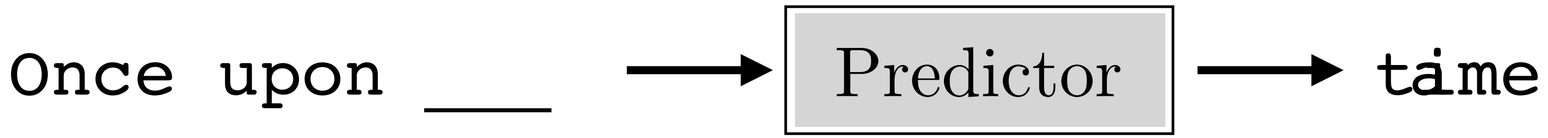
- Optimization challenges
- Lack of diversity
- Mode collapse: only models one "mode" of the probability distribution

Figure source: [Razavi et al., 2019]

Today

- Texture synthesis
- Generative adversarial nets (GANs)
- **Autoregressive models**

Recall: autoregressive models for text



Can we do something analogous for images?

Source: P. Isola

Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

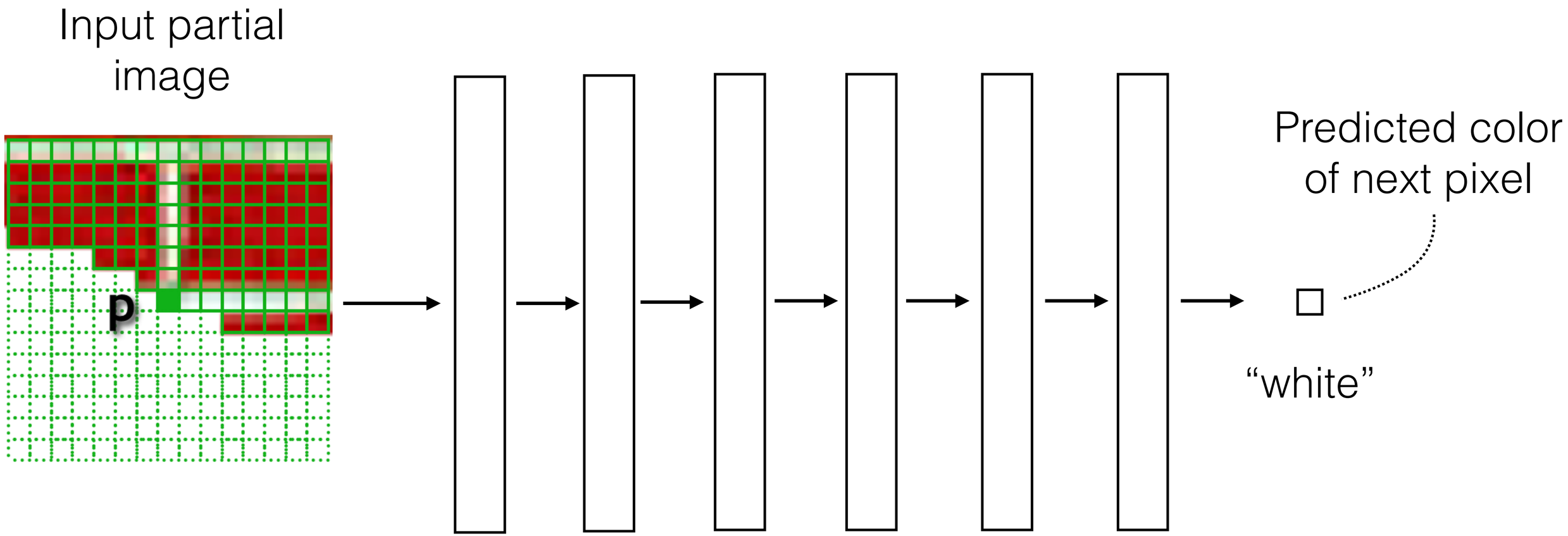
Generating images one pixel at a time

Generating images one pixel at a time

Generating images one pixel at a time

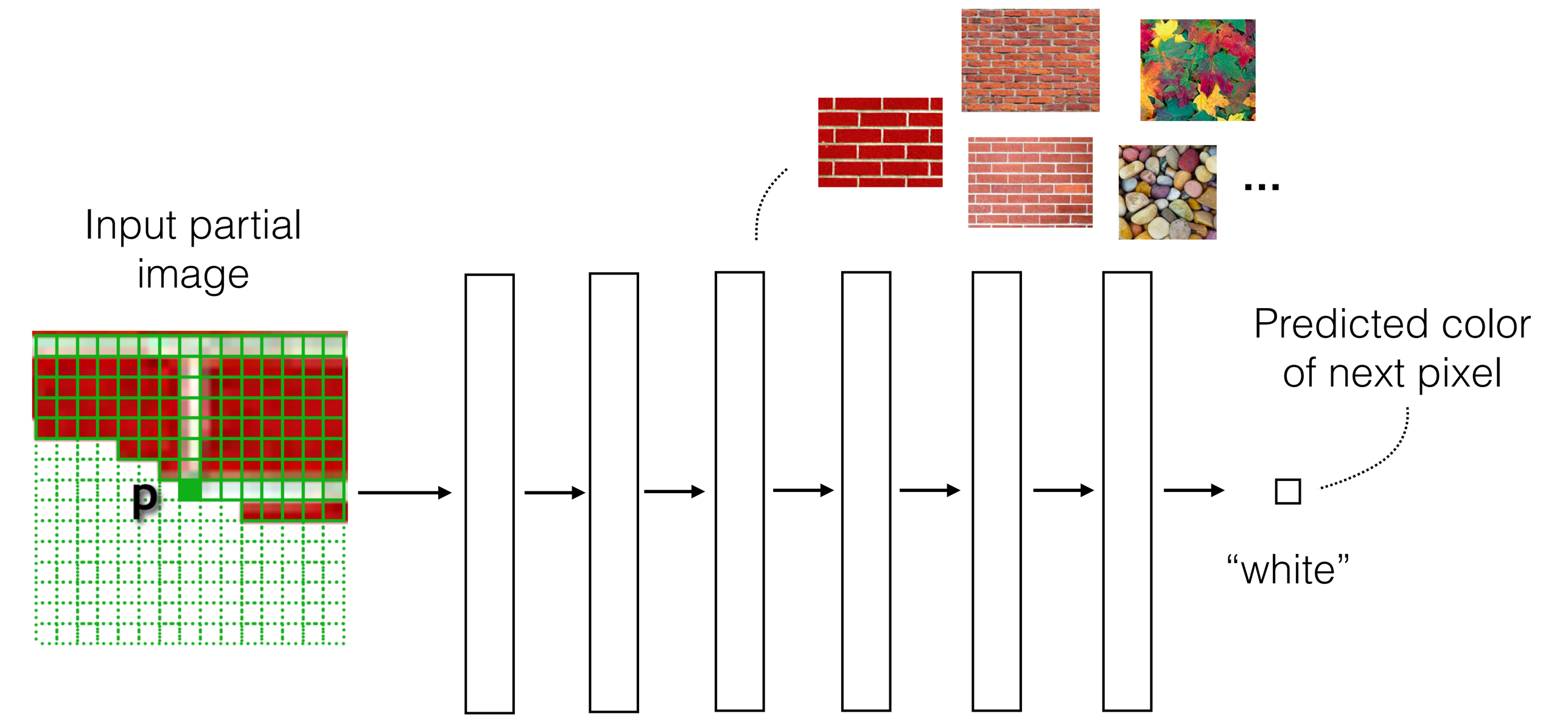
Generating images one pixel at a time

Texture synthesis with a neural network



[PixelRNN, PixelCNN, van der Oord et al. 2016]

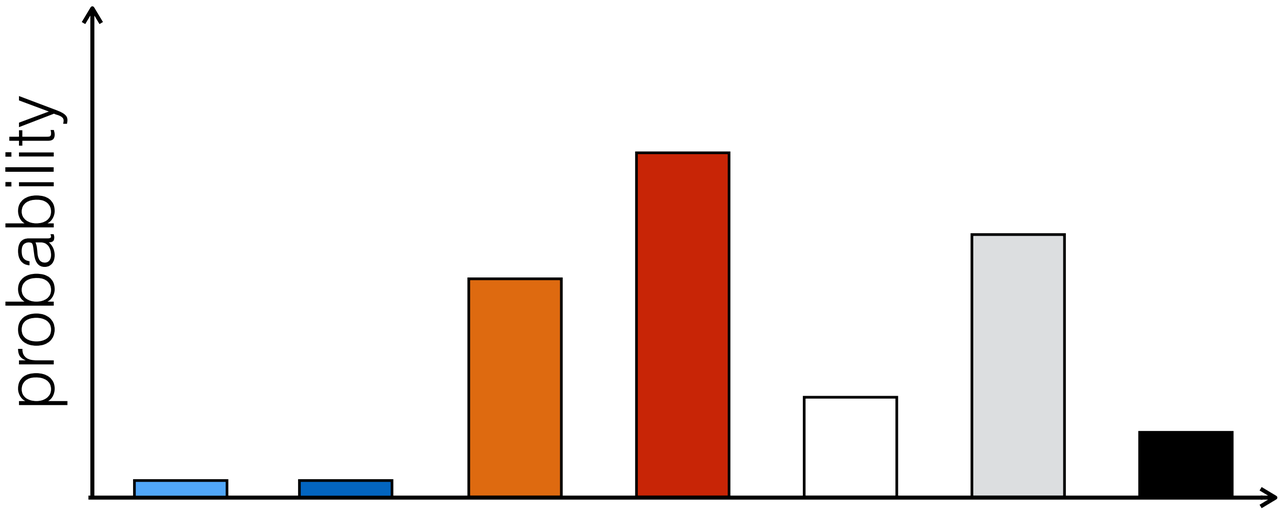
Source: Torralba, Freeman, Isola



[PixelRNN, PixelCNN, van der Oord et al. 2016]

Source: Torralba, Freeman, Isola

Represent colors as discrete categories



- One class for each intensity value, e.g. 256 classes
- One label per color channel
- Newer methods use mixture density networks

Source: Torralba, Freeman, Isola

Represent colors as discrete categories

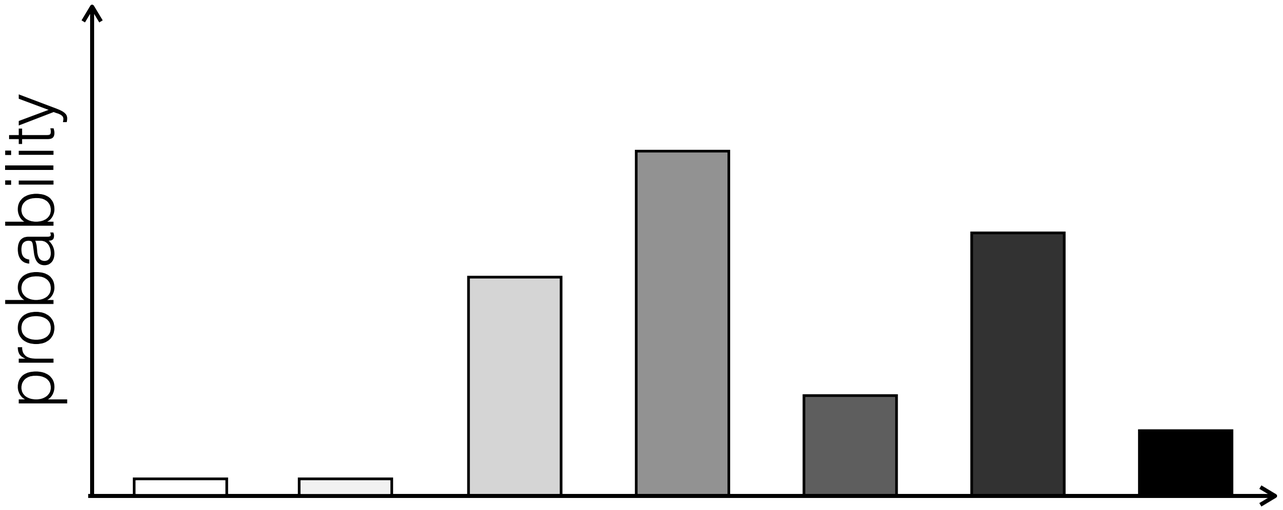
The model is learning: P(next pixel | previous pixels):

$$\hat{\mathbf{y}} \equiv [P_\theta(Y = 1|X = \mathbf{x}), \ldots, P_\theta(Y = K|X = \mathbf{x})]$$ ← predicted probability of each class given input **x**

$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$ ← picks out the -log likelihood of the ground truth class $\mathbf{y}$ under the model prediction $\hat{\mathbf{y}}$

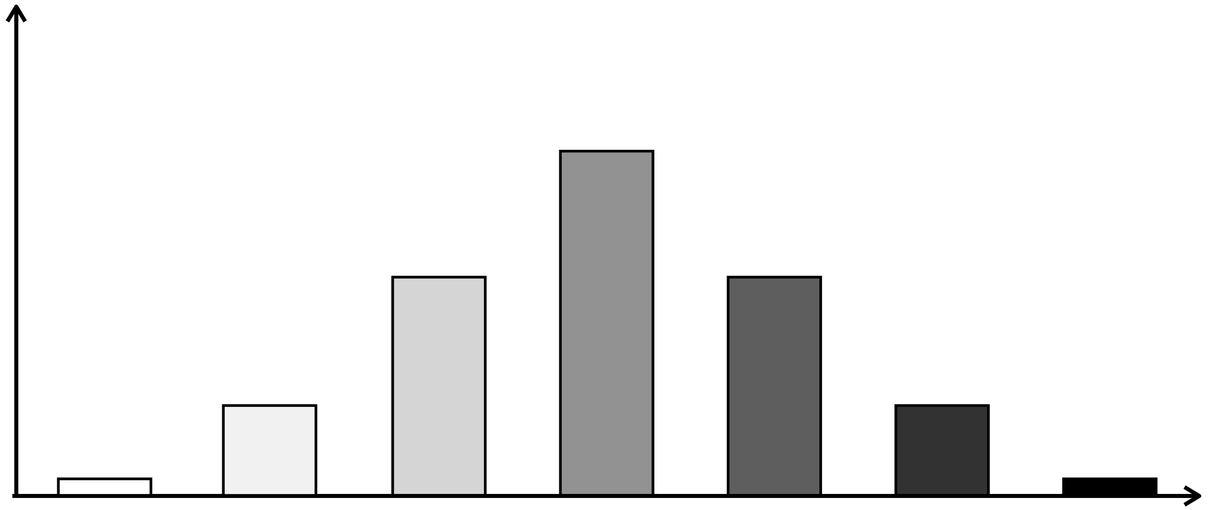
Source: Torralba, Freeman, Isola

Representing distributions



Distribution has multiple modes

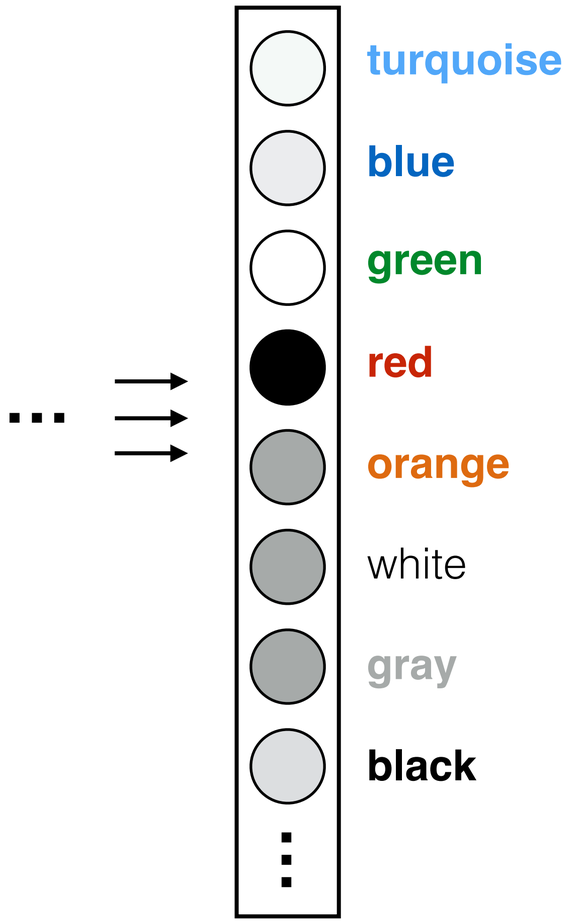
$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$



Single mode

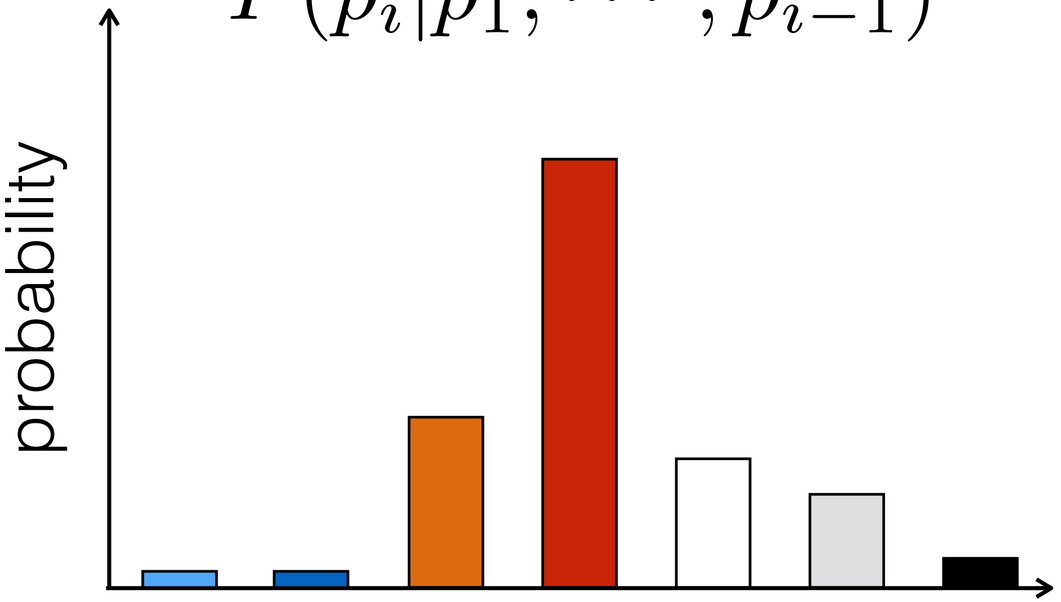
$$L_2(y, \hat{y}) = -\log(P_{\text{gauss}}(\hat{y}|y)) \propto (y - \hat{y})^2$$

Network output



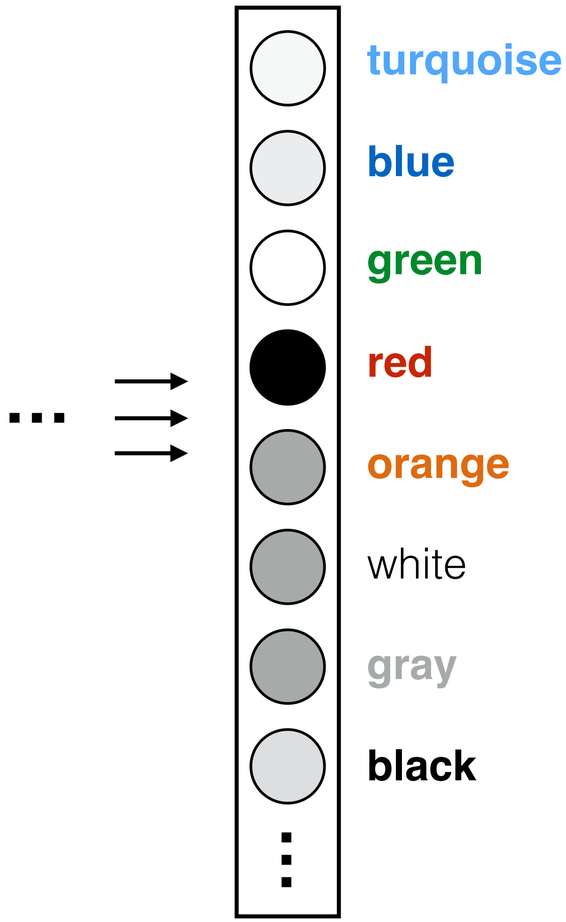
P(next pixel | previous pixels)

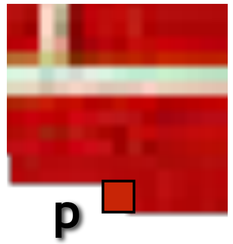
$$P(p_i | p_1, \cdots, p_{i-1})$$



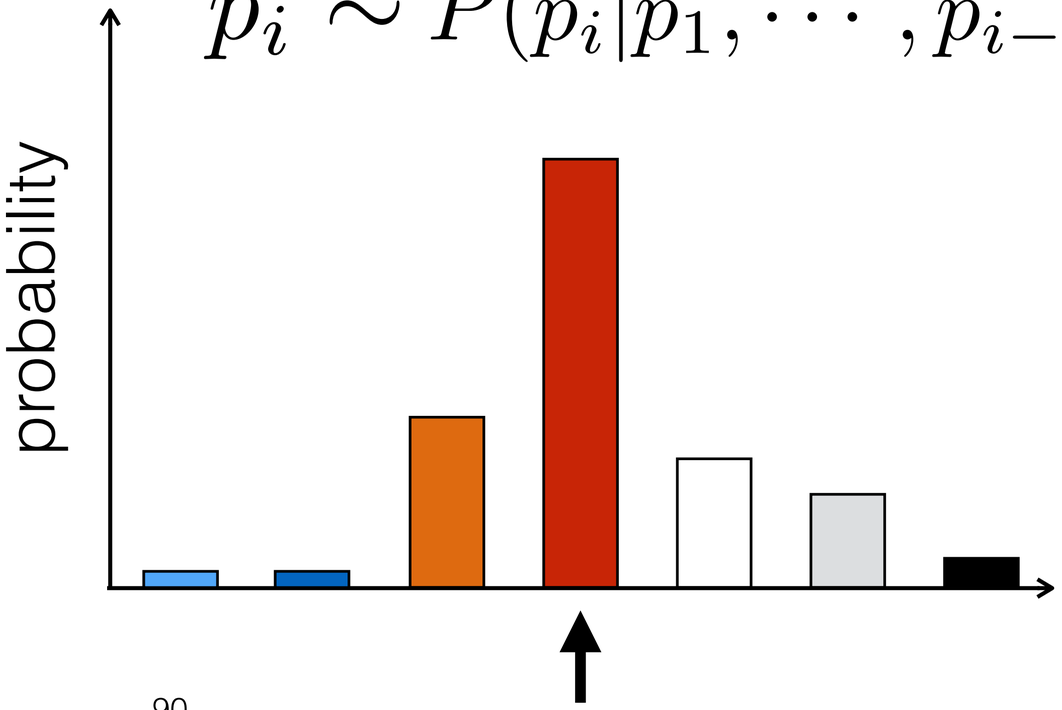
Source: Torralba, Freeman, Isola

Network output



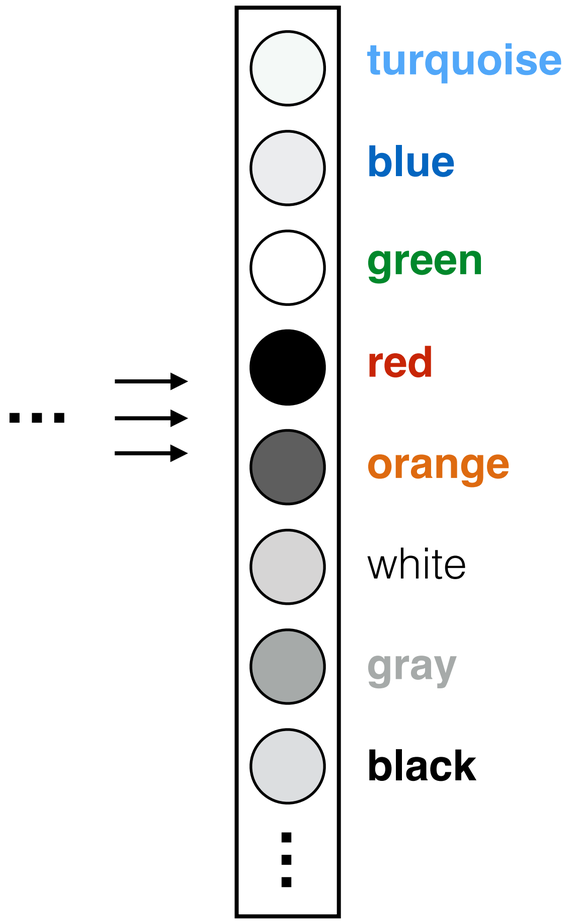


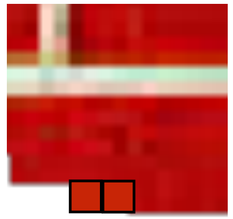
$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



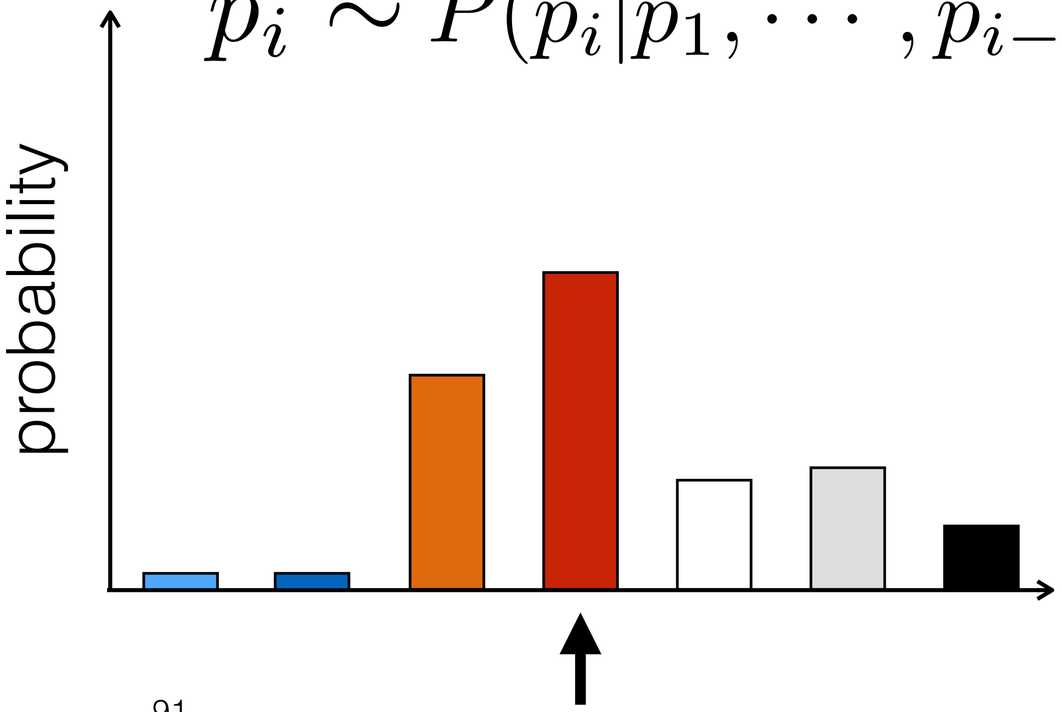
Source: Torralba, Freeman, Isola

Network output



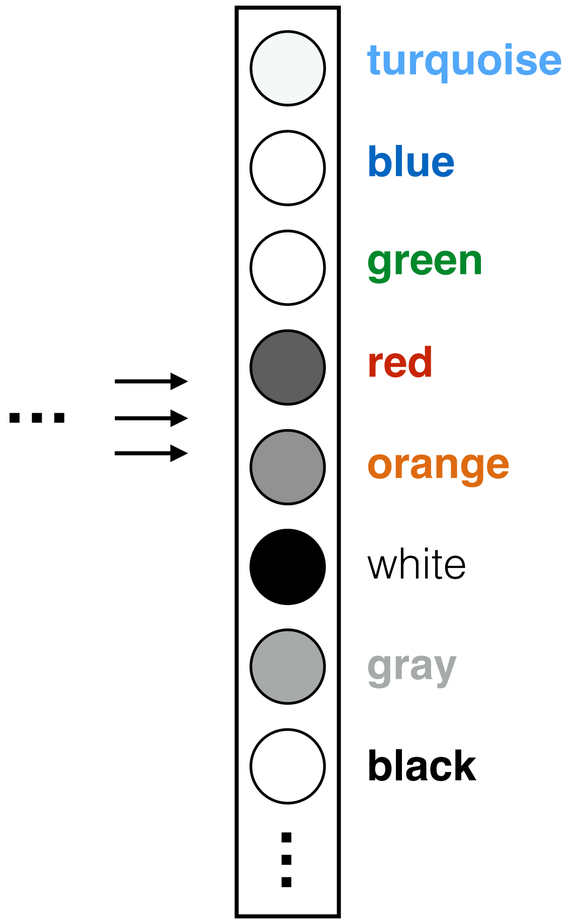


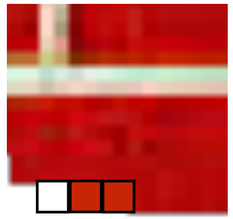
$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



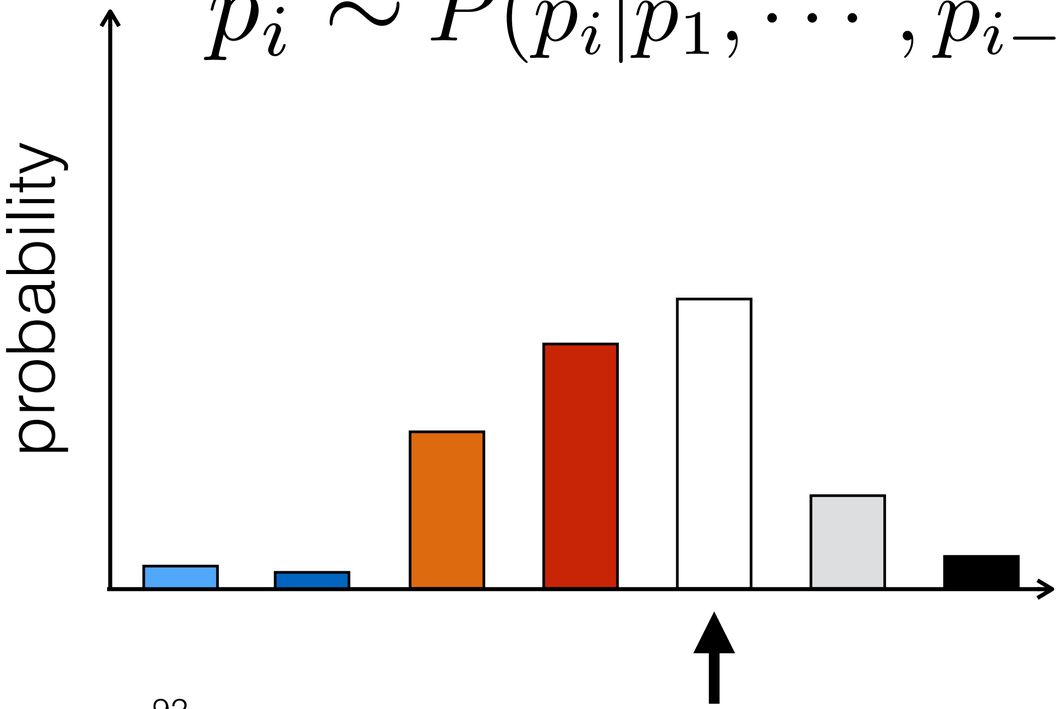
Source: Torralba, Freeman, Isola

Network output



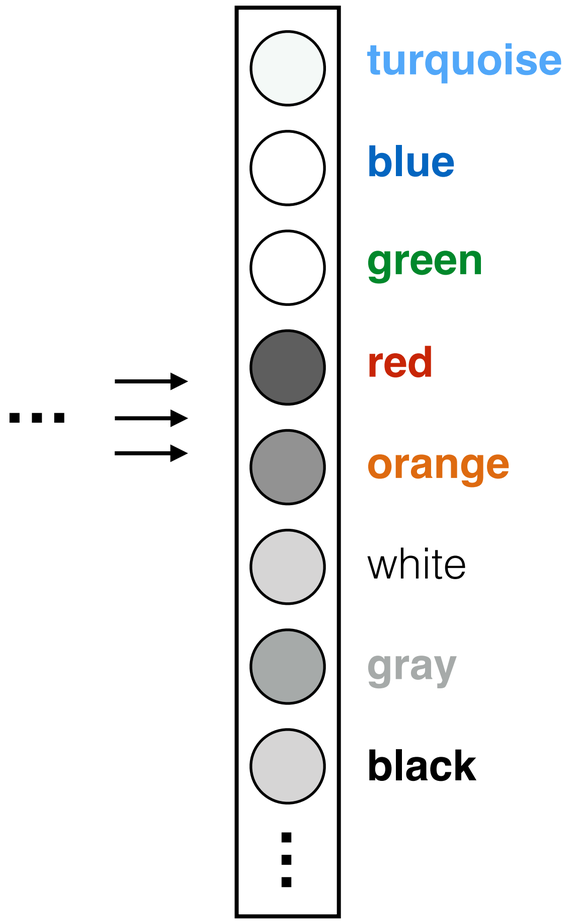


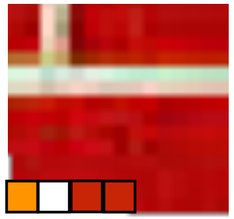
$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



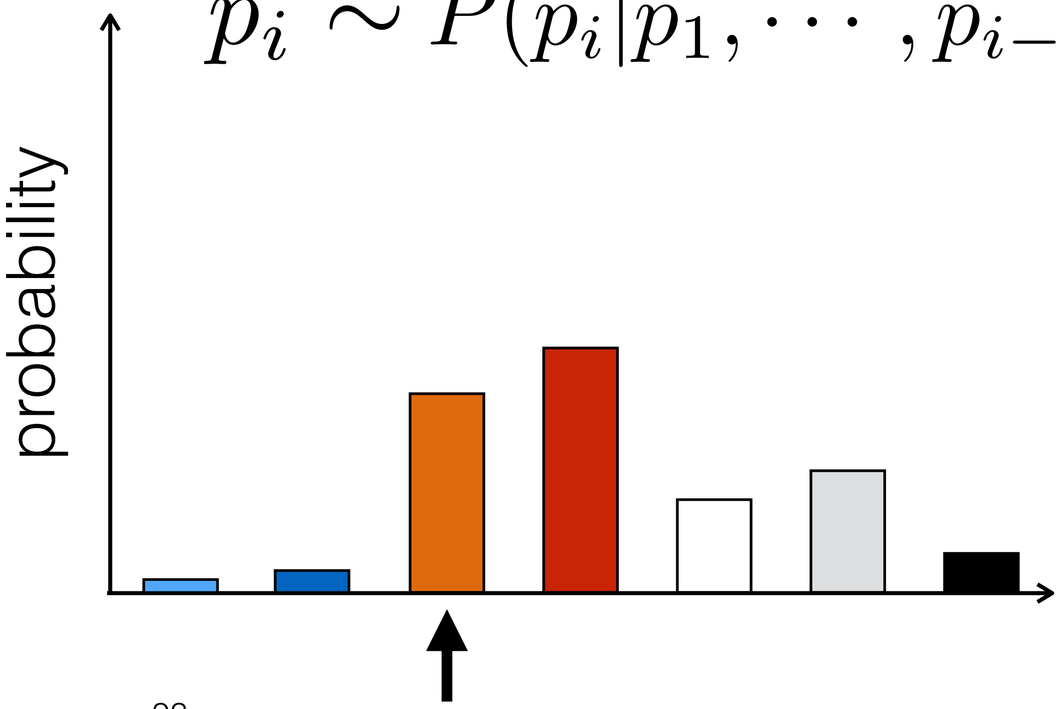
Source: Torralba, Freeman, Isola

Network output





$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



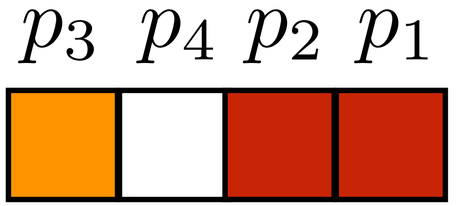
Source: Torralba, Freeman, Isola

$$p_1 \sim P(p_1)$$

$$p_2 \sim P(p_2|p_1)$$

$$p_3 \sim P(p_3|p_1, p_2)$$

$$p_4 \sim P(p_4|p_1, p_2, p_3)$$



$$\{p_1, p_2, p_3, p_4\} \sim P(p_4|p_1, p_2, p_3)P(p_3|p_1, p_2)P(p_2|p_1)P(p_1)$$

$$p_i \sim P(p_i|p_1, \ldots, p_{i-1})$$

$$\mathbf{p} \sim \prod_{i=1}^{N} P(p_i|p_1, \ldots, p_{i-1})$$

Source: Torralba, Freeman, Isola

Autoregressive probability model

$$\mathbf{p} \sim \prod_{i=1}^{N} P(p_i | p_1, \ldots, p_{i-1})$$

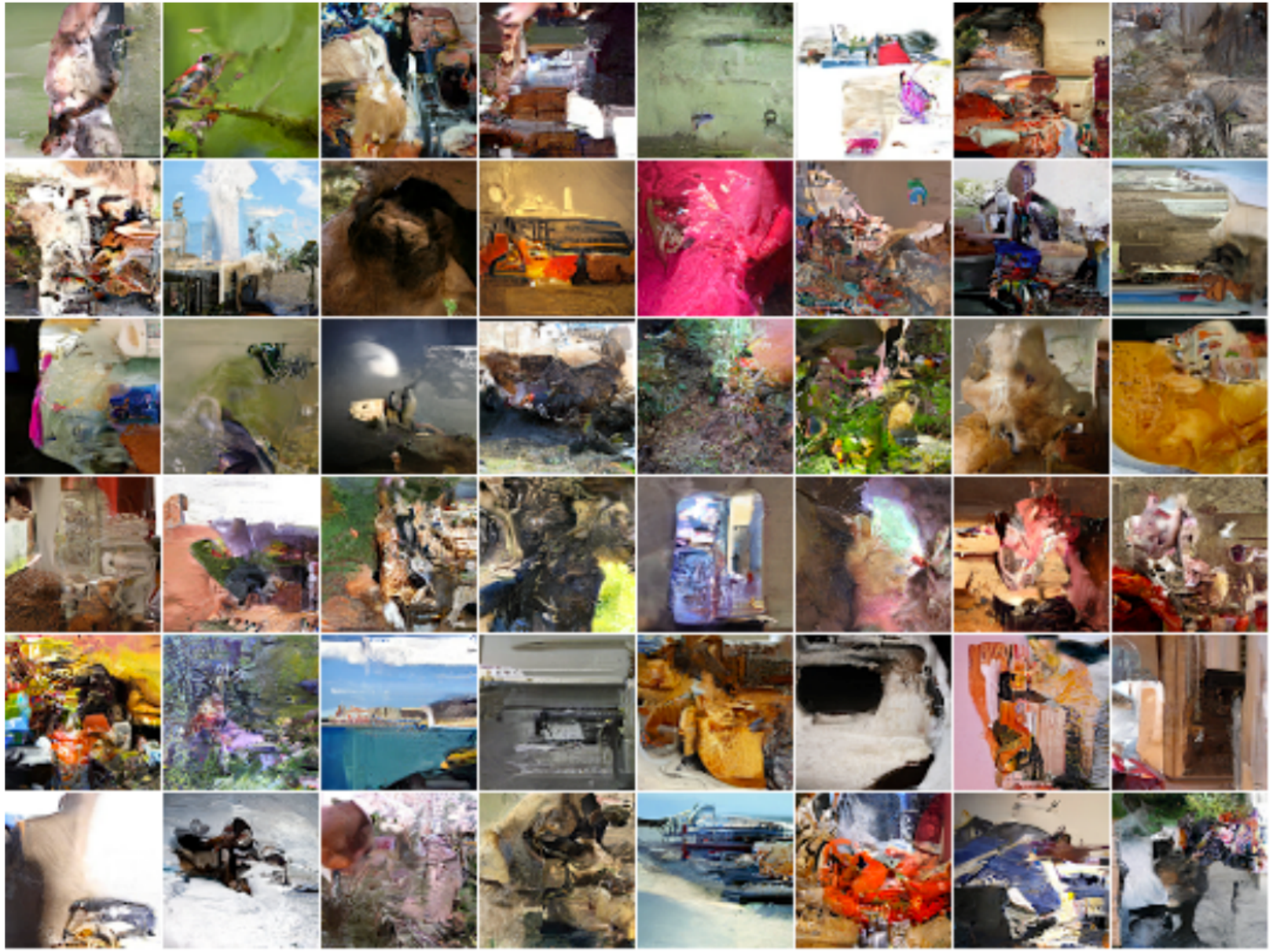
$$P(\mathbf{p}) = \prod_{i=1}^{N} P(p_i | p_1, \ldots, p_{i-1}) \quad \longleftarrow \quad \textbf{General product rule}$$

The sampling procedure we defined above takes exact samples from the learned probability distribution (pdf).

Multiplying all conditionals evaluates the probability of a full joint configuration of pixels.

Source: Torralba, Freeman, Isola

Samples from PixelRNN



[PixelRNN, van der Oord et al. 2016]

Source: Torralba, Freeman, Isola

Image completions (conditional samples) from PixelRNN

occluded

Source: Torralba, Freeman, Isola

[PixelRNN, van der Oord et al. 2016]

Image completions (conditional samples) from PixelRNN

[PixelRNN, van der Oord et al. 2016]

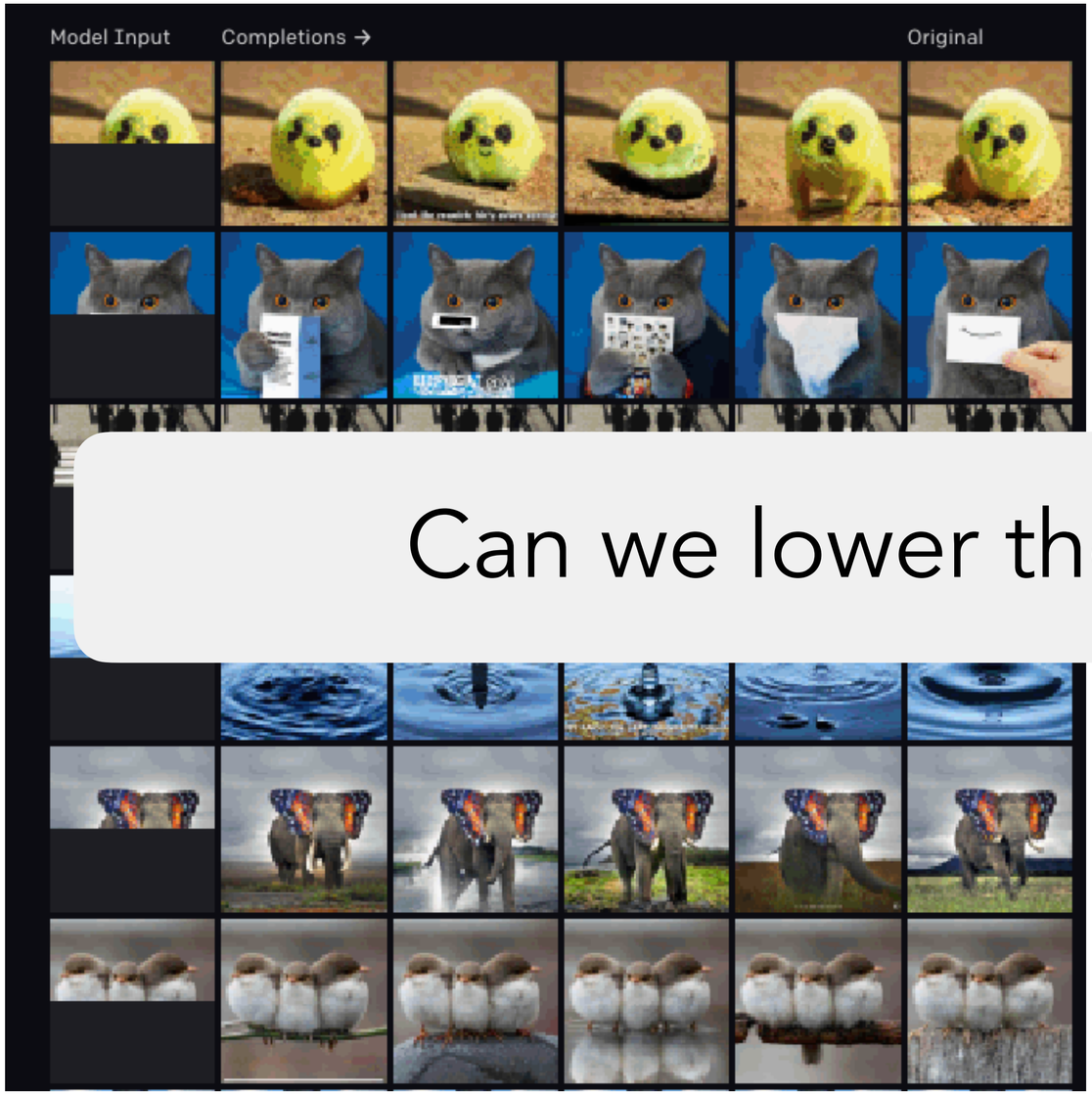
Source: Torralba, Freeman, Isola

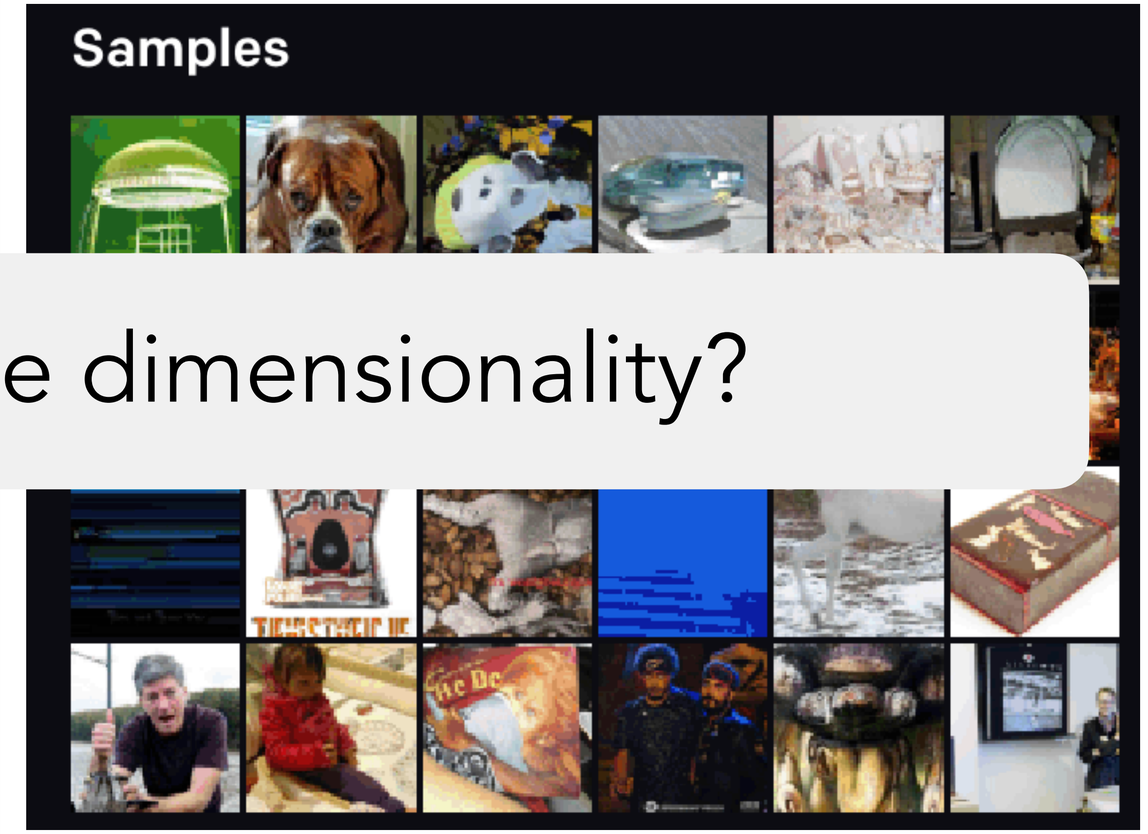
Image completions (conditional samples) from PixelRNN

[PixelRNN, van der Oord et al. 2016]

Source: Torralba, Freeman, Isola

Hard to scale up, though…





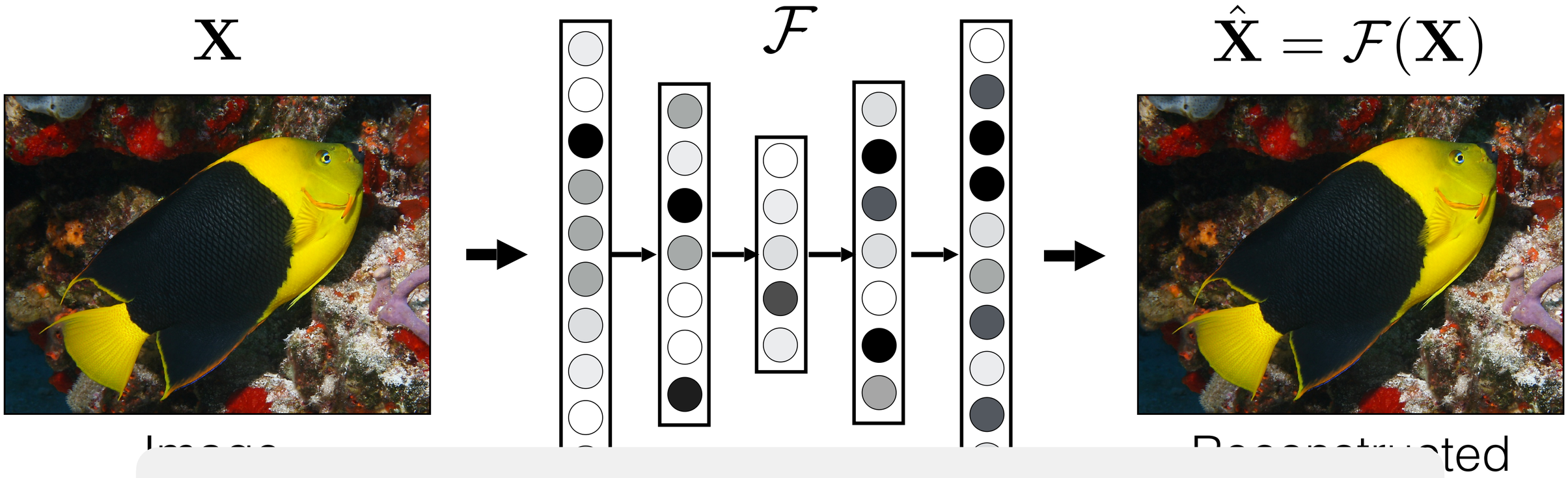
Can we lower the dimensionality?

https://openai.com/blog/image-gpt/

M. Chen et al., Generative pretraining from pixels, ICML 2020

Source: S. Lazebnik

Lower dimensionality using an autoencoder

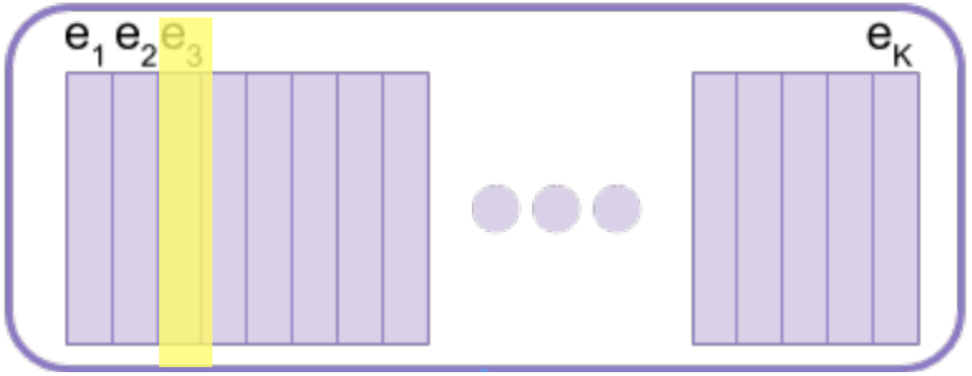


How can we make the output discrete?

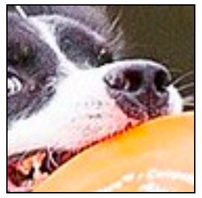
Figure source: Torralba, Freeman, Isola

Vector quantization

Predict a real-valued vector, then "snap" it to a nearest neighbor from a *codebook*.



Codebook



Patch **x**

E(**x**) → **z** → Quantization → $\mathbf{e}_k$

$$\text{Quantize}(E(\mathbf{x})) = \mathbf{e}_k \quad \text{where } k = \arg\min_j ||E(\mathbf{x}) - \mathbf{e}_j||$$

Figure adapted from [van den Oord et al., "VQ-VAE", 2017

Vector quantized (variational) autoencoder

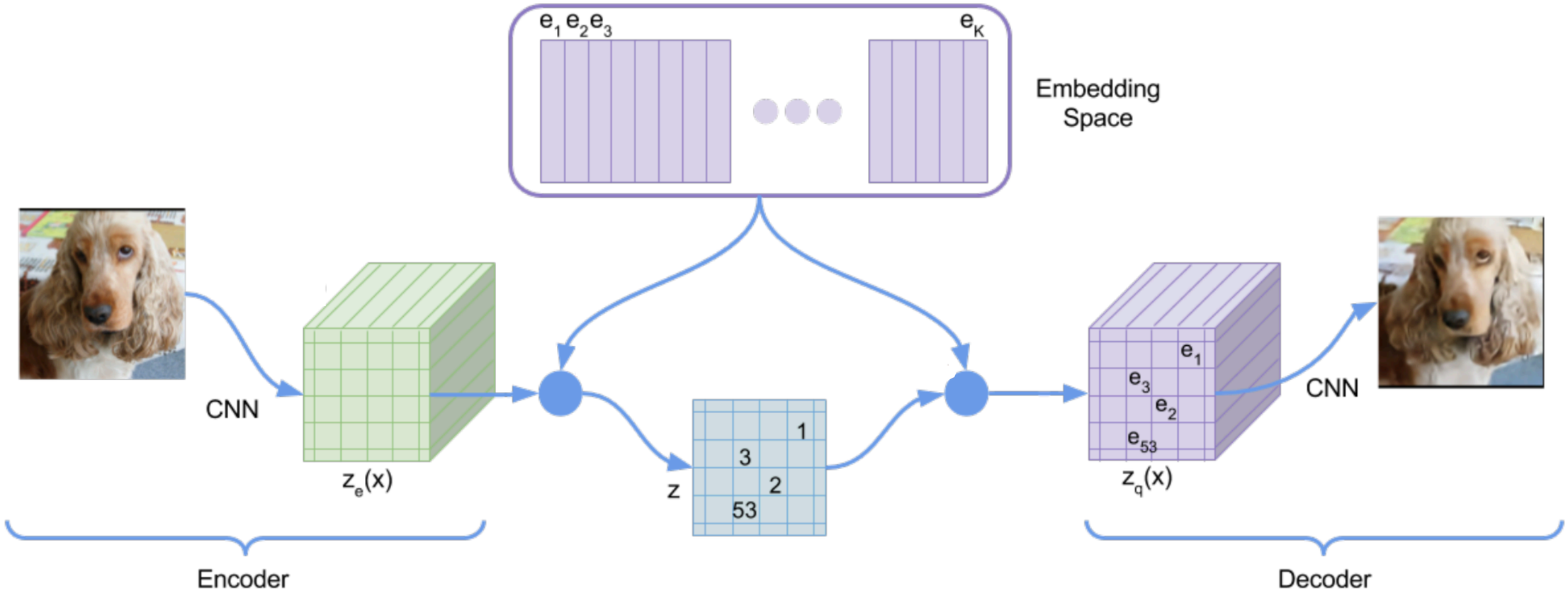
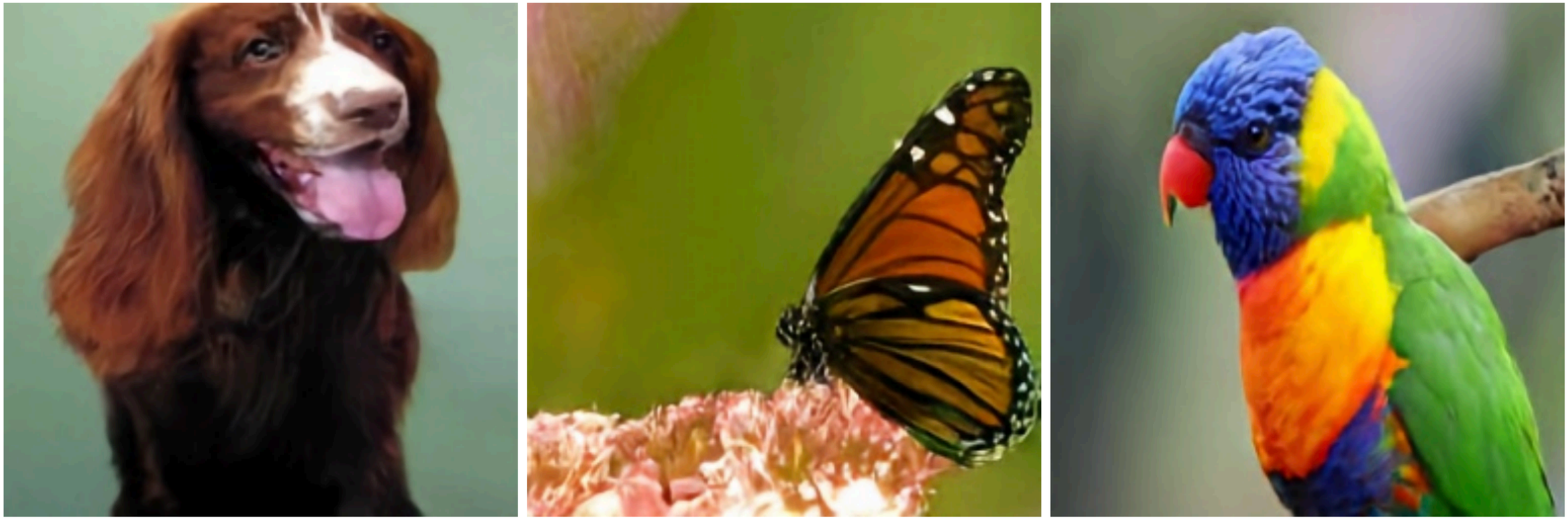


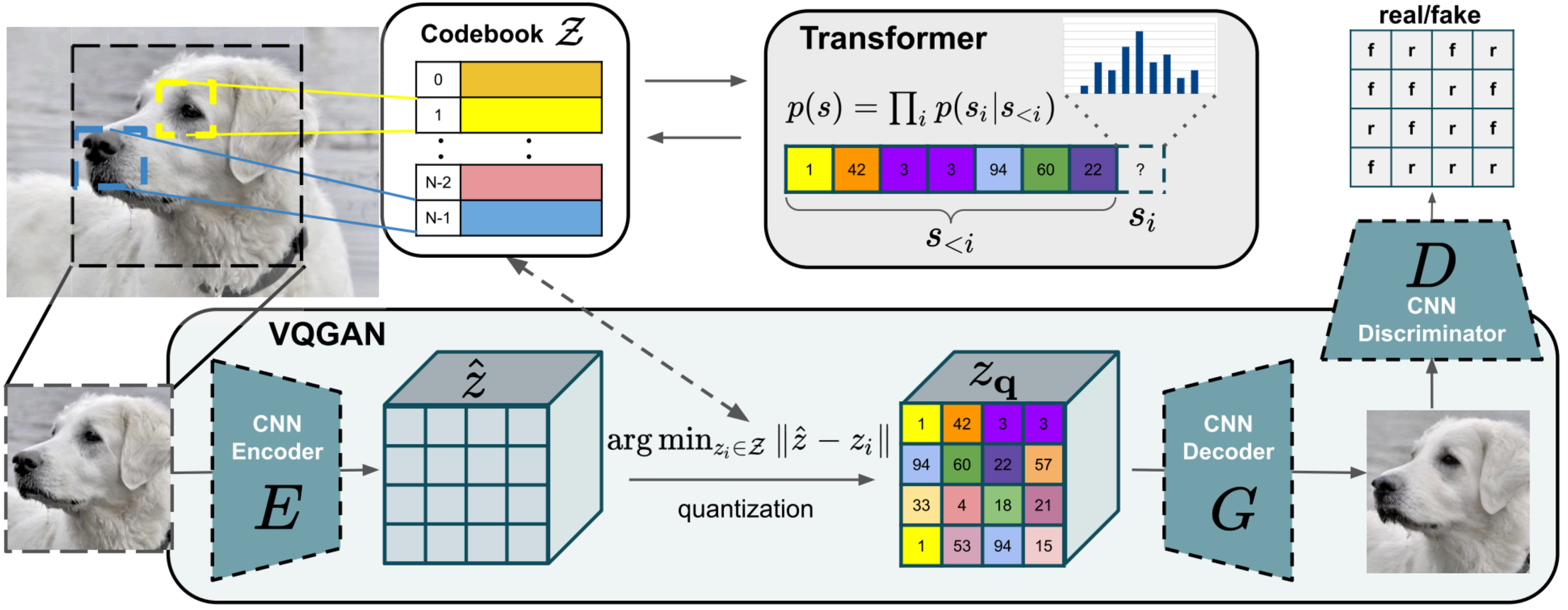
Figure source: [van den Oord et al., “VQ-VAE”, 2017]



Generated images (256 × 256)

Figure source: [van den Oord et al., “VQ-VAE 2”, 2017

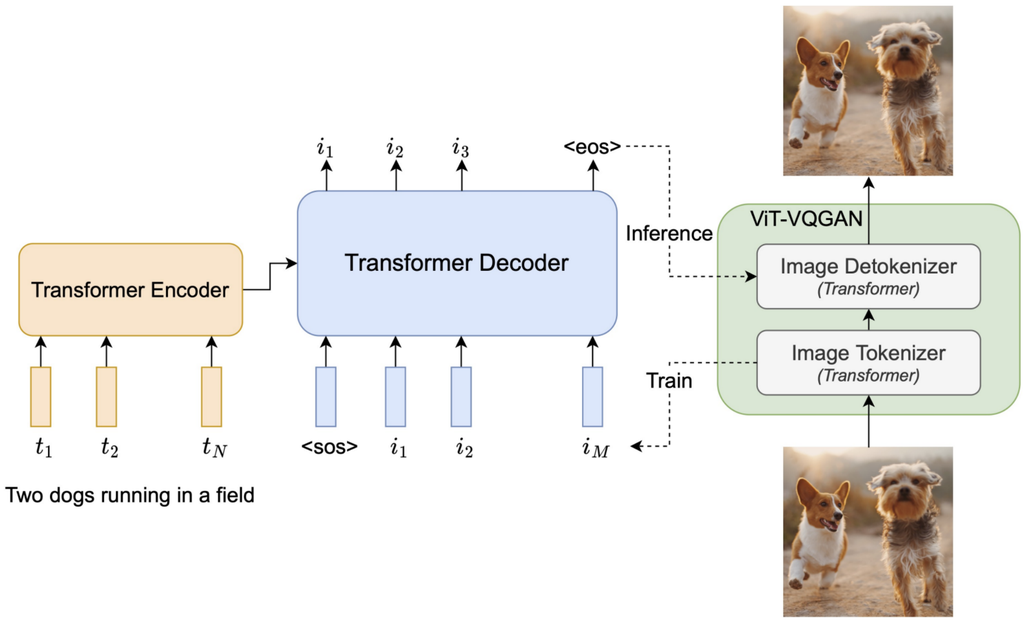
Vector quantized GAN (VQ-GAN)



Add a GAN loss to get crisper images

[Esser et al., "Taming Transformers for High-Resolution Image Synthesis", 2021

Autoregressive text-to-image generation



A. *A photo of a frog reading the newspaper named "Toaday" written on it. There is a frog printed on the newspaper too.*

B. *A portrait of a statue of the Egyptian god Anubis wearing aviator goggles, white t-shirt and leather jacket. The city of Los Angeles is in the background. Hi-res DSLR photograph.*

[Yu et al., "Parti", 2022]

Next time: more image synthesis