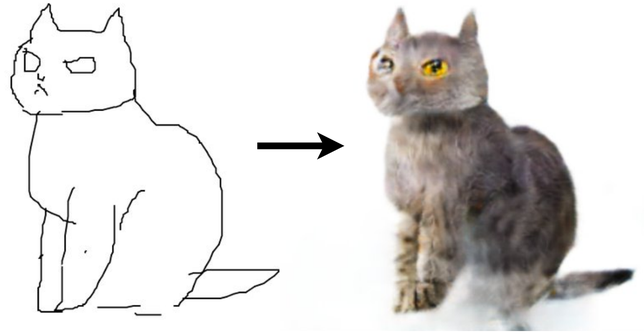
$z \sim \mathcal{N}(\vec{0}, 1)$ →



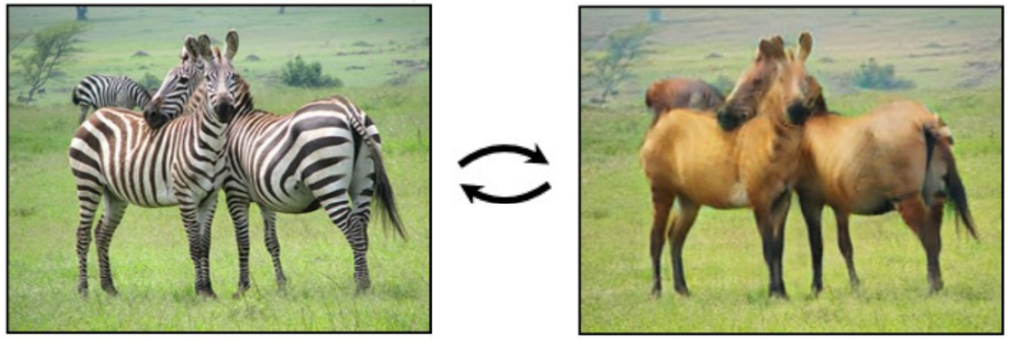


Image synthesis

Bill Freeman, Antonio Torralba, Phillip Isola
6.819 / 6.869

Image classification

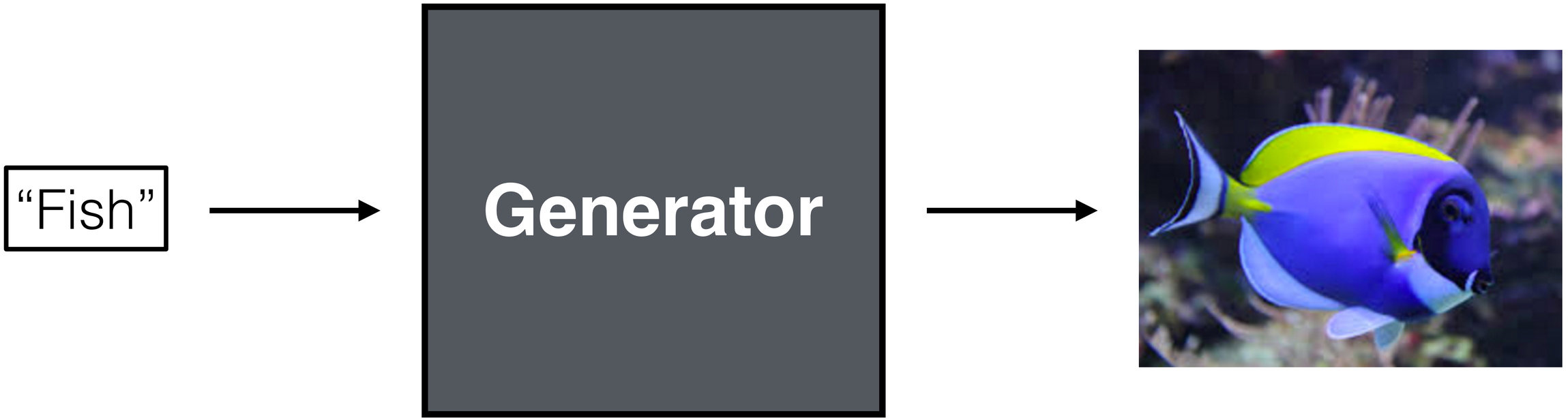
Classifier

“Fish”

image X

label Y

Image synthesis



label Y

image X

Image synthesis via **generative modeling**

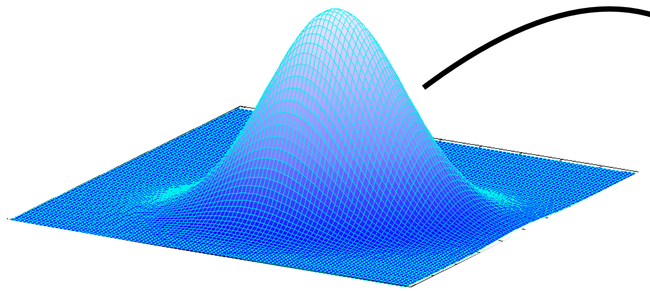
X is high-dimensional!

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

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

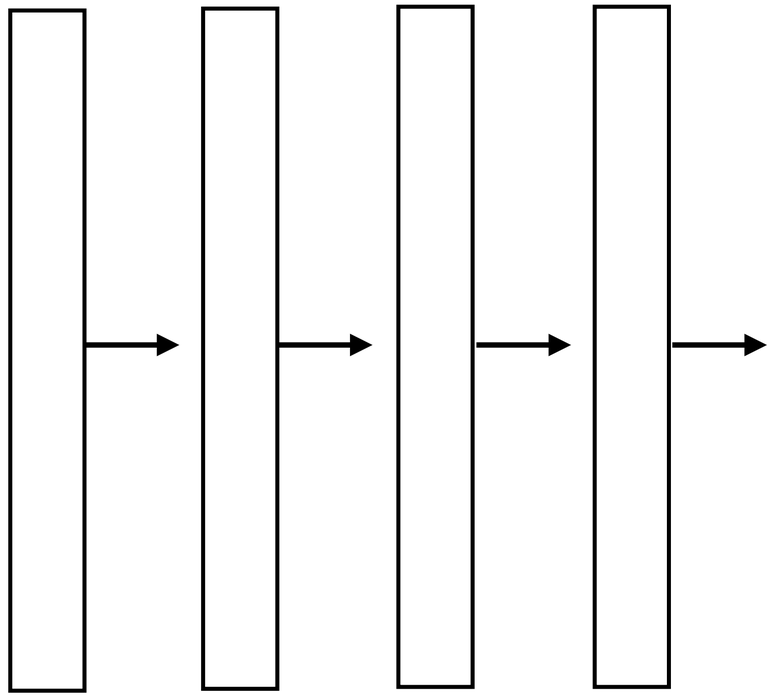
Generative Model

$\mathbf{z}$



Gaussian noise

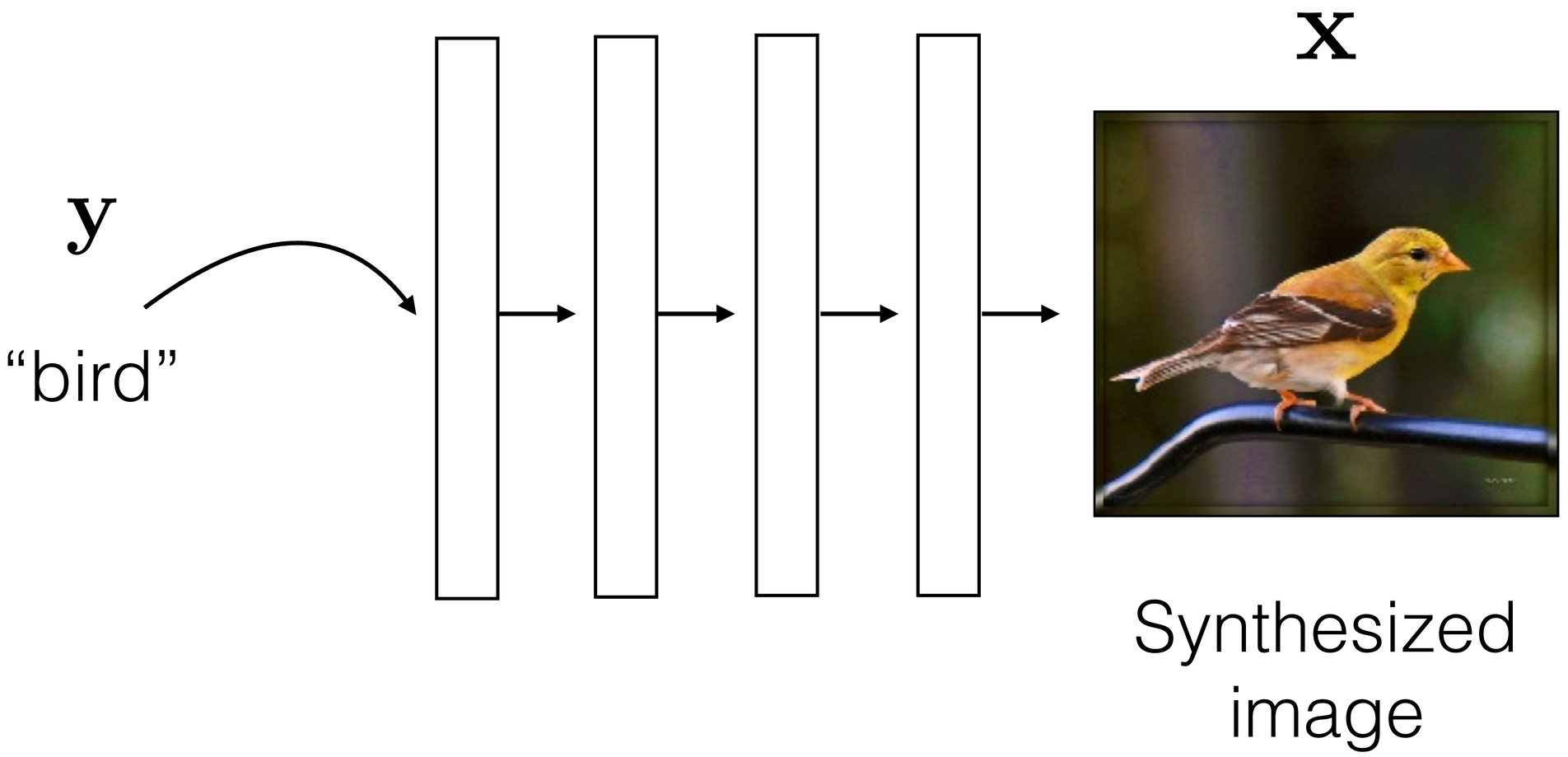
$z \sim \mathcal{N}(\vec{0}, 1)$



$\mathbf{x}$

Synthesized image

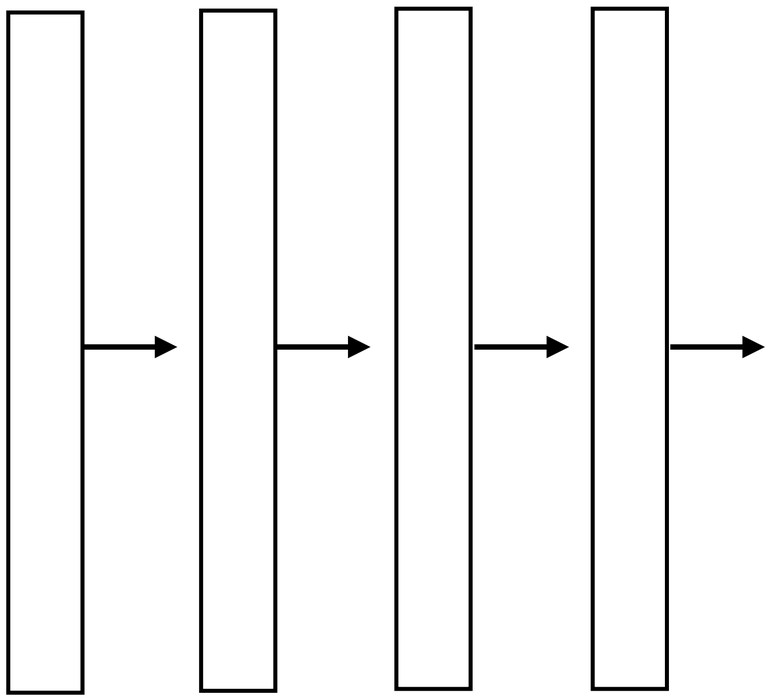
Conditional Generative Model



Conditional Generative Model

$\mathbf{y}$

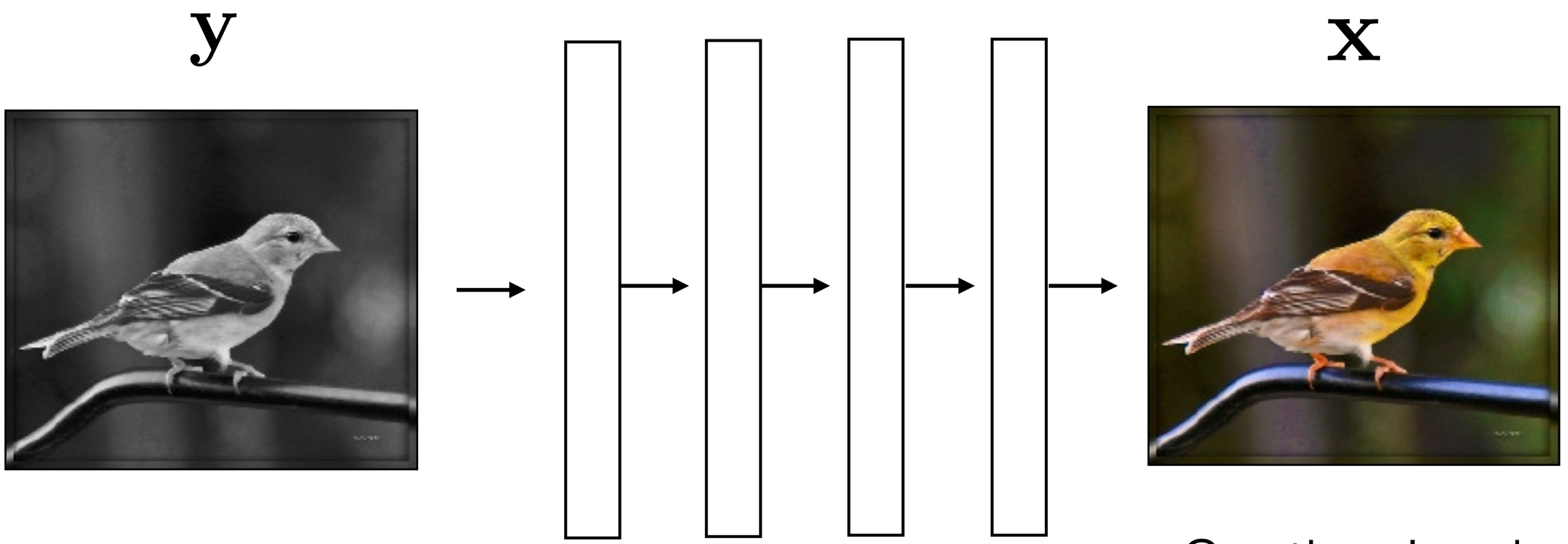
"A yellow bird on a branch"



$\mathbf{x}$

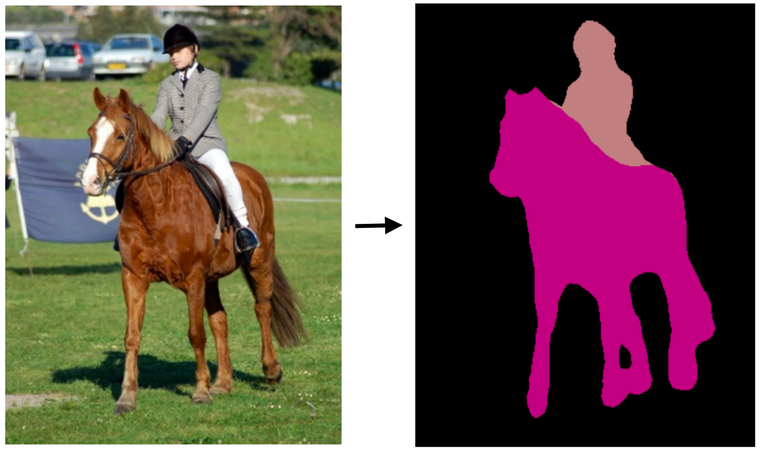
Synthesized image

Conditional Generative Model



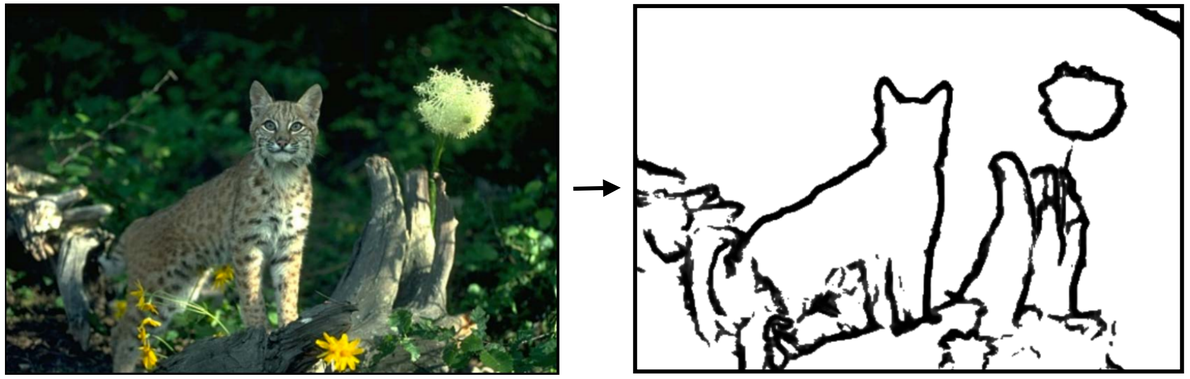
Data prediction problems ("structured prediction")

Semantic segmentation



[Long et al. 2015, …]

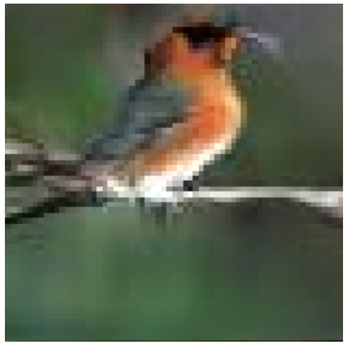
Edge detection



[Xie et al. 2015, …]

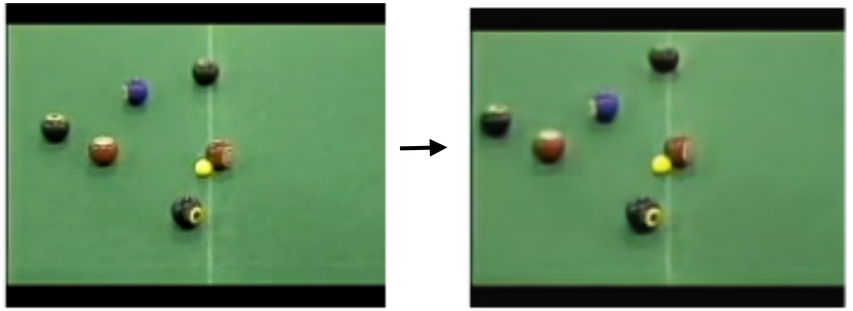
Text-to-photo

"this small bird has a pink breast and crown…"



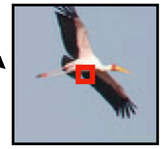
[Reed et al. 2014, …]

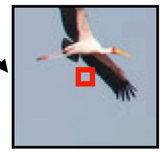
Future frame prediction

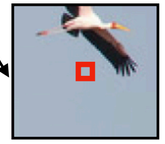


[Mathieu et al. 2016, …]

What's the object class of the center pixel?



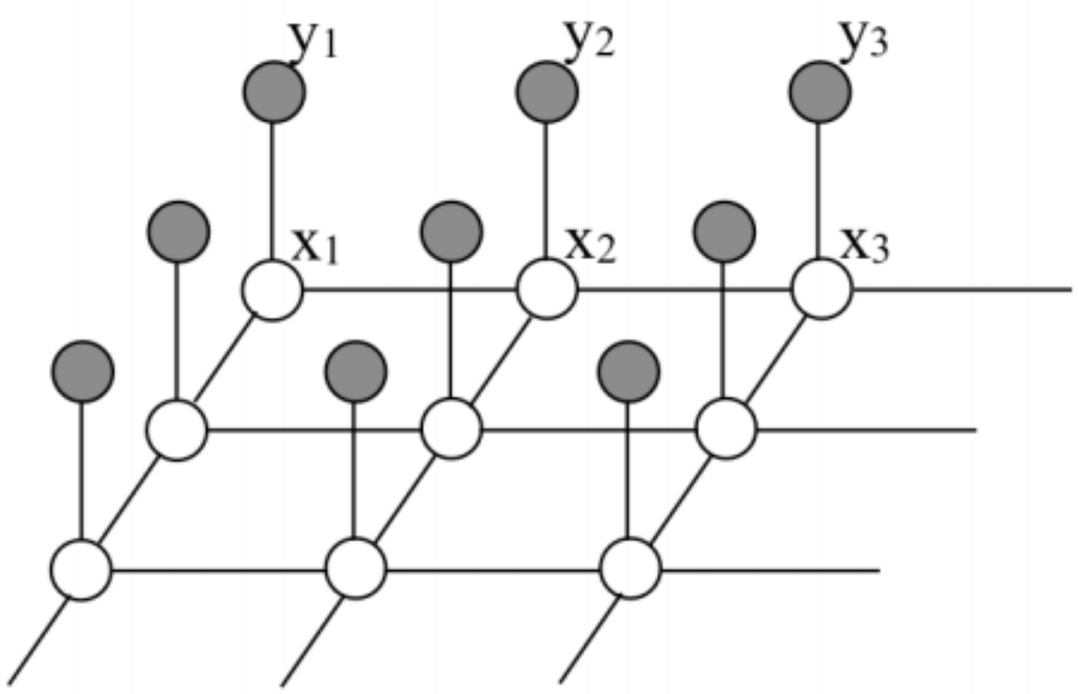




Each prediction is done independently!

Recall from lecture 11 and pset 6:

http://6.869.csail.mit.edu/fa18/notes/L11GraphicalModels.pdf

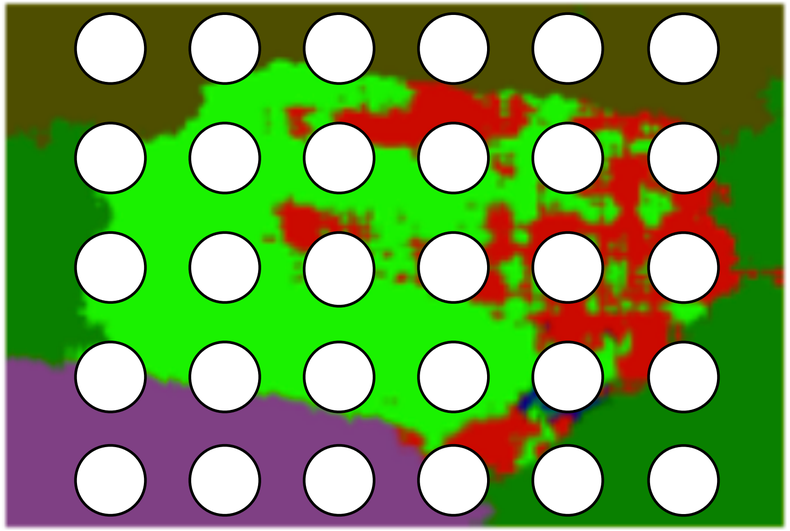


$$P(\mathbf{x}|\mathbf{y}) = \frac{1}{P(\mathbf{y})} \prod_{i,j} \phi_{i,j}(x_i, x_j) \prod_i \psi(x_i, y_i)$$

$$P(\mathbf{x}|\mathbf{y}) = \frac{1}{Z} \prod_i P(x_i|\mathbf{y}_{\mathcal{N}(i)})$$

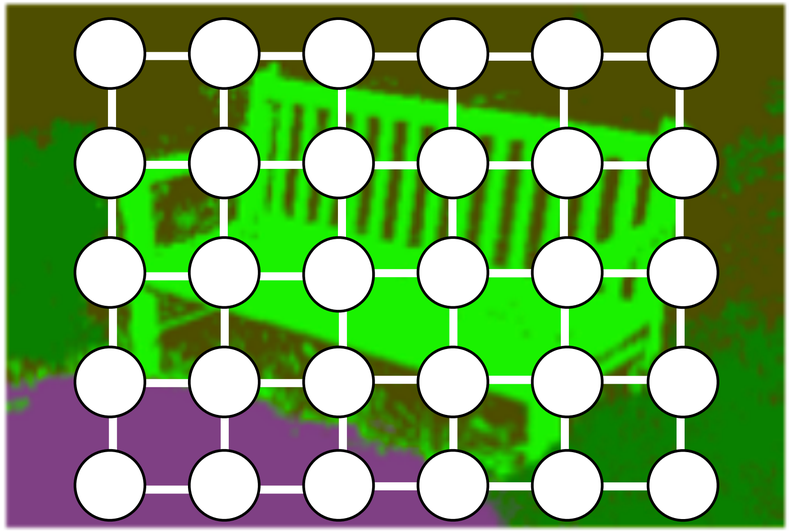
Input

Independent prediction per-pixel



$$\max \prod_i p(y_i|\mathbf{x})$$

Find a configuration of compatible labels

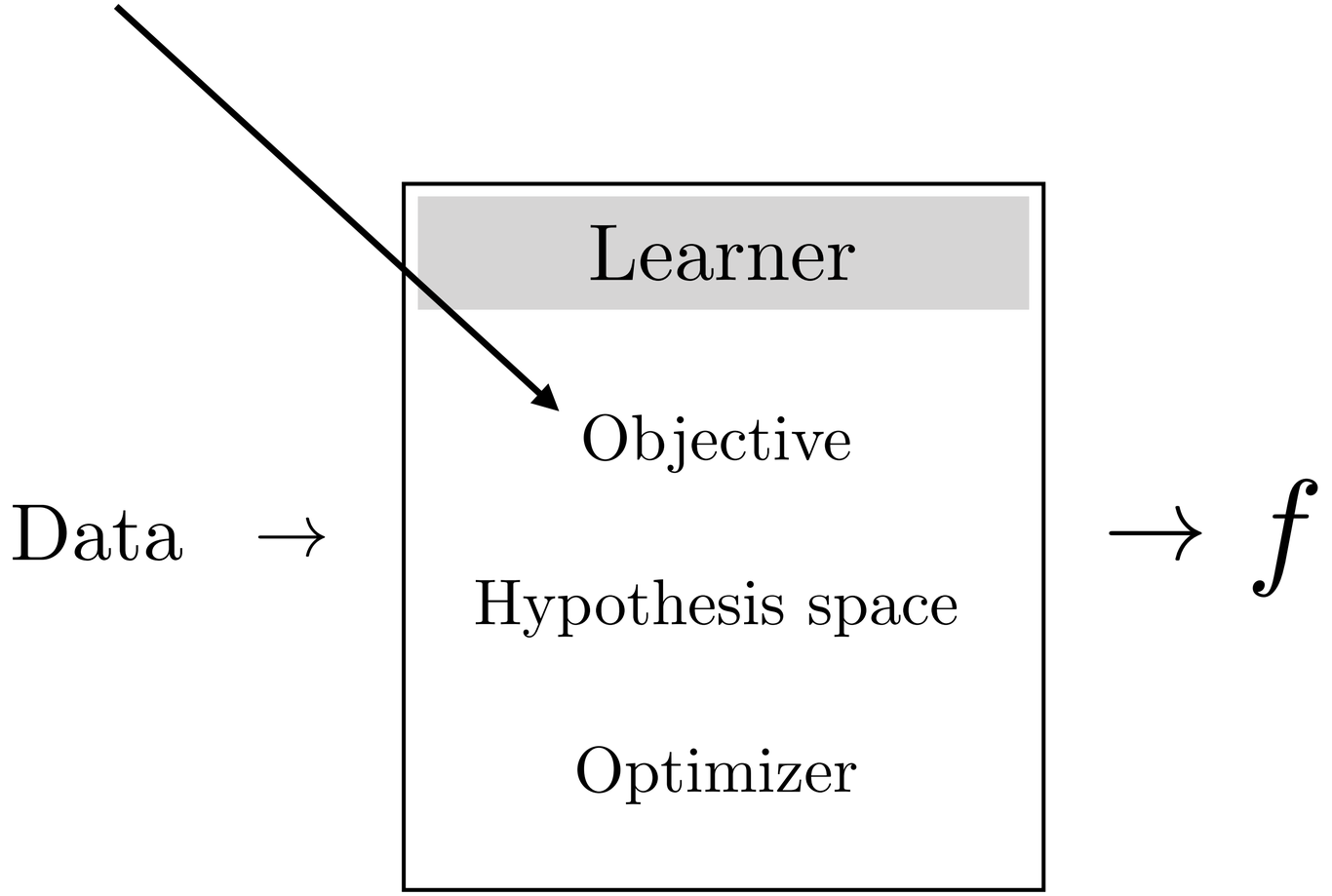


$$\max \frac{1}{Z} \prod_{i,j} p(y_i, y_j|\mathbf{x})$$

["Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials", Krahenbuhl and Koltun, NIPS 2011]

Structured prediction

Define an objective that penalizes bad structure! (e.g., a graphical model)



Unstructured prediction

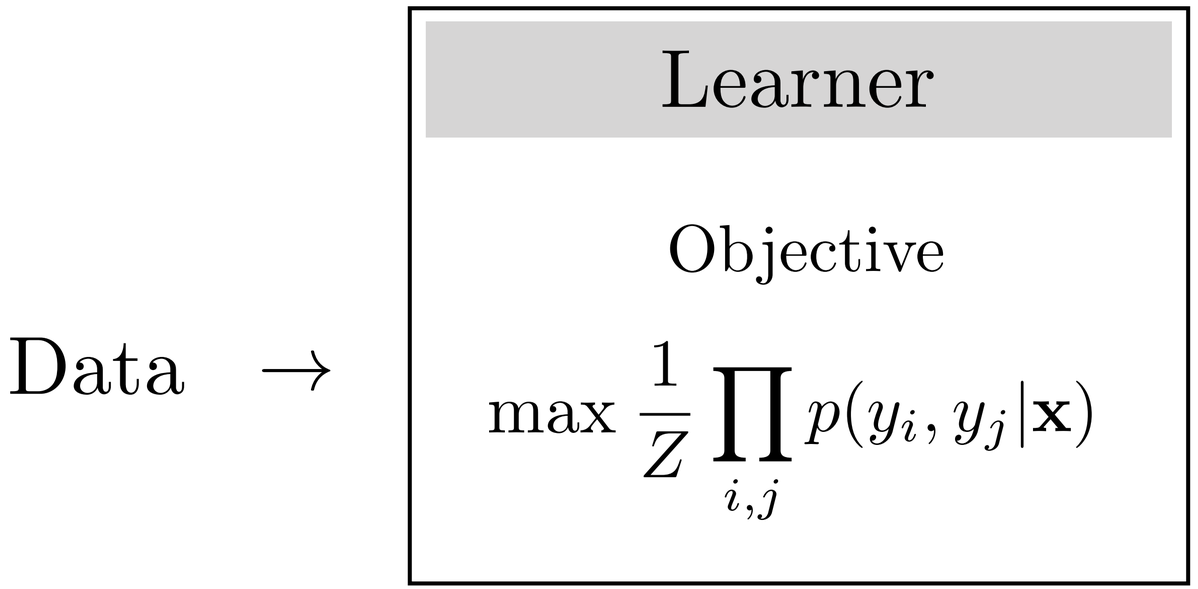
$$\text{Data} \rightarrow \boxed{\begin{array}{c}\text{Learner} \\ \\ \text{Objective} \\ \\ \max \prod_i p(y_i|\mathbf{x})\end{array}} \rightarrow f$$

All learning objectives we have seen in this class so far had this form!

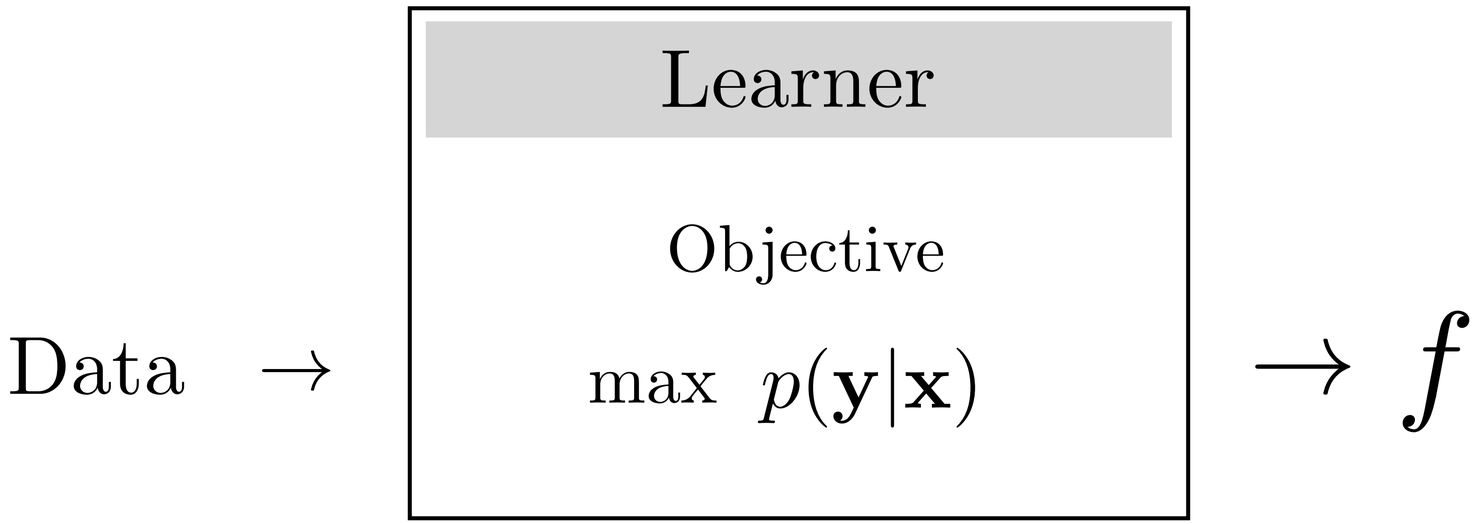
Per-datapoint least-squares regression: $\theta^* = \arg\min_\theta \sum_{i=1}^{N} (f_\theta(x_i) - y_i)^2$

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

Structured prediction with a conditional random field



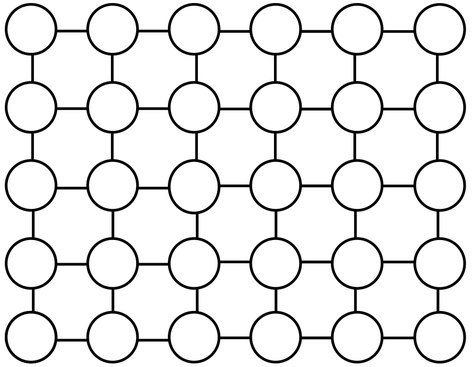
Structured prediction with a generative model



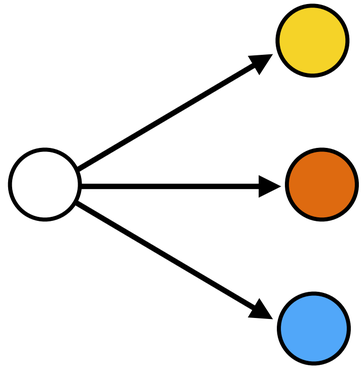
Model *joint* configuration of all outputs **y**

Challenges in visual prediction

1. Output is a high-dimensional, structured object

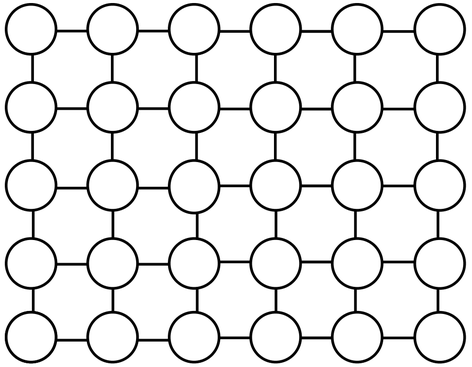


2. Uncertainty in the mapping, many plausible outputs



Properties of generative models

1. Model high-dimensional, structured output



2. Model uncertainty; a whole distribution of possible outputs

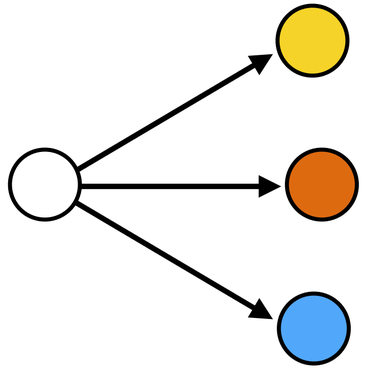


Image-to-Image Translation

Input $\mathbf{x}$

$\mathcal{F}$

Output $\mathbf{y}$

Training data

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

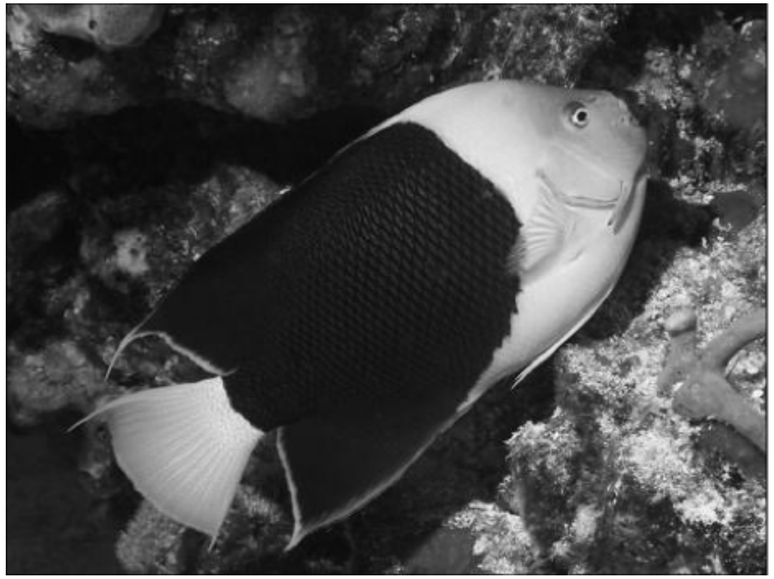
$$\arg\min_{\mathcal{F}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[L(\mathcal{F}(\mathbf{x}), \mathbf{y})]$$

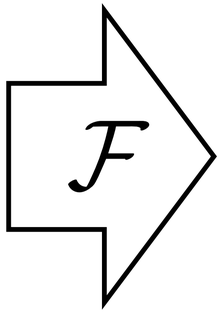
Objective function (loss)

Neural Network

Image-to-Image Translation

Input $\mathbf{x}$





Output $\mathbf{y}$

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

"**What** should I do"

"**How** should I do it?"

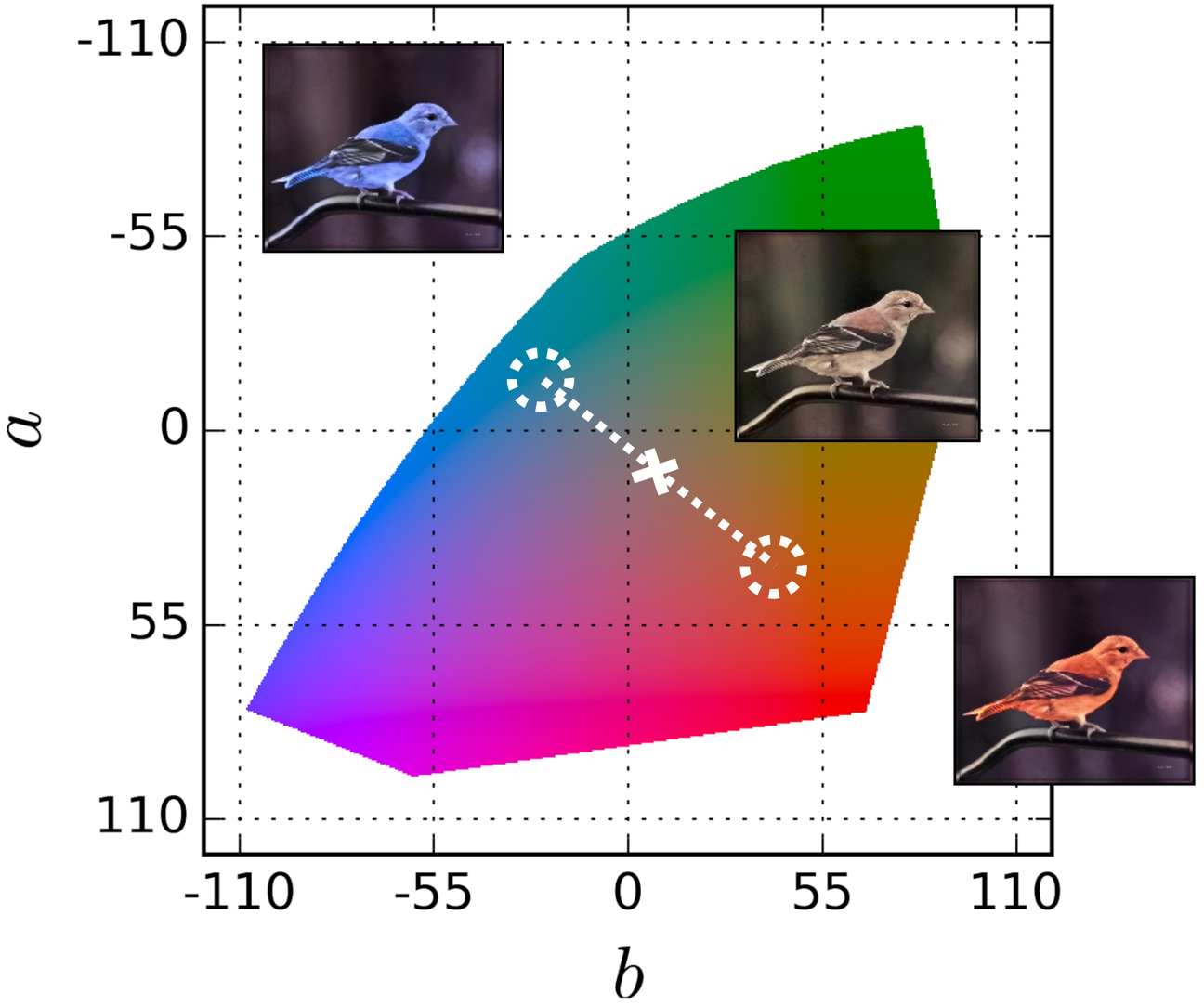
Designing loss functions

Input

Output

Ground truth

$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2}\sum_{h,w} \|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$



$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2}\sum_{h,w} \|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$

Designing loss functions

Input

Zhang et al. 2016

Ground truth

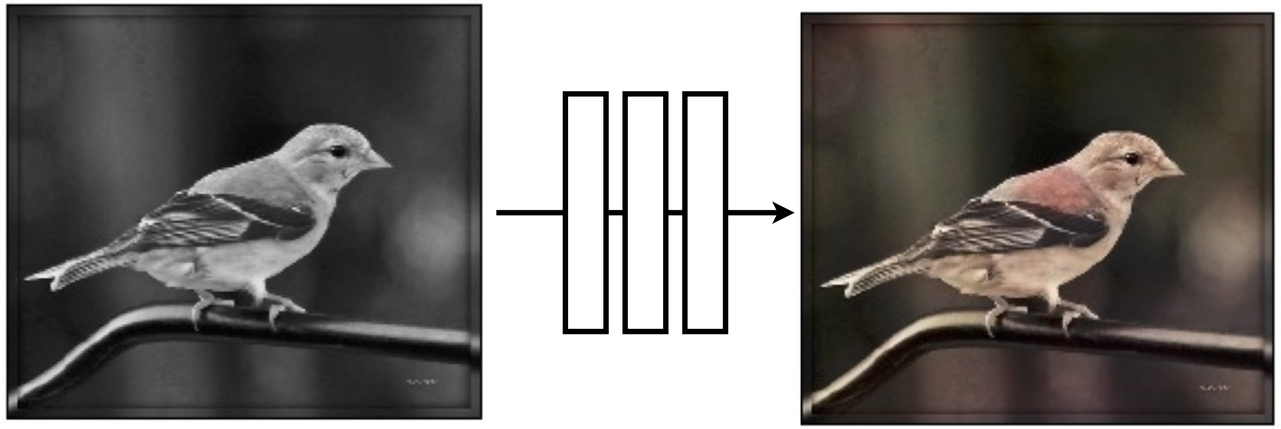
Color distribution cross-entropy loss with colorfulness enhancing term.

Designing loss functions

Be careful what you wish for!

Designing loss functions

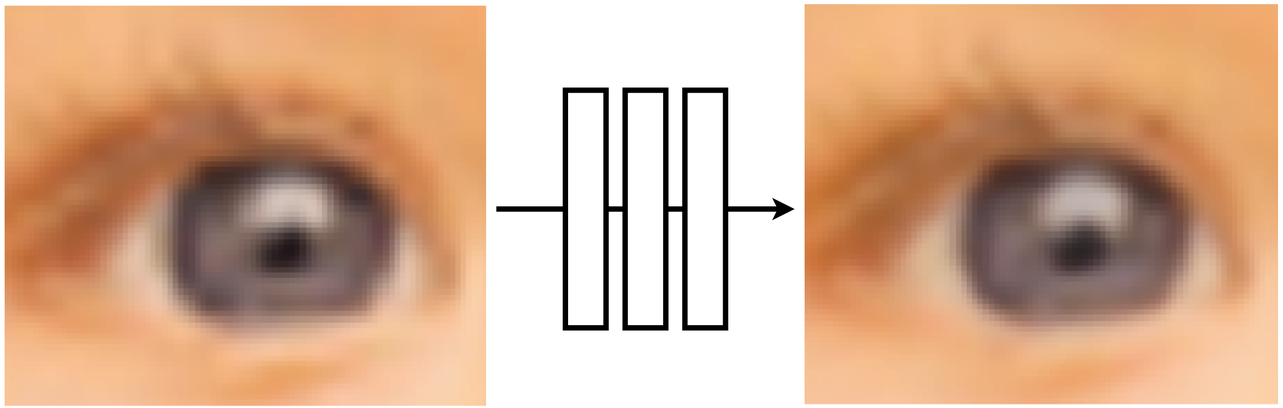
Image colorization



[Zhang, Isola, Efros, ECCV 2016]

L2 regression

Super-resolution

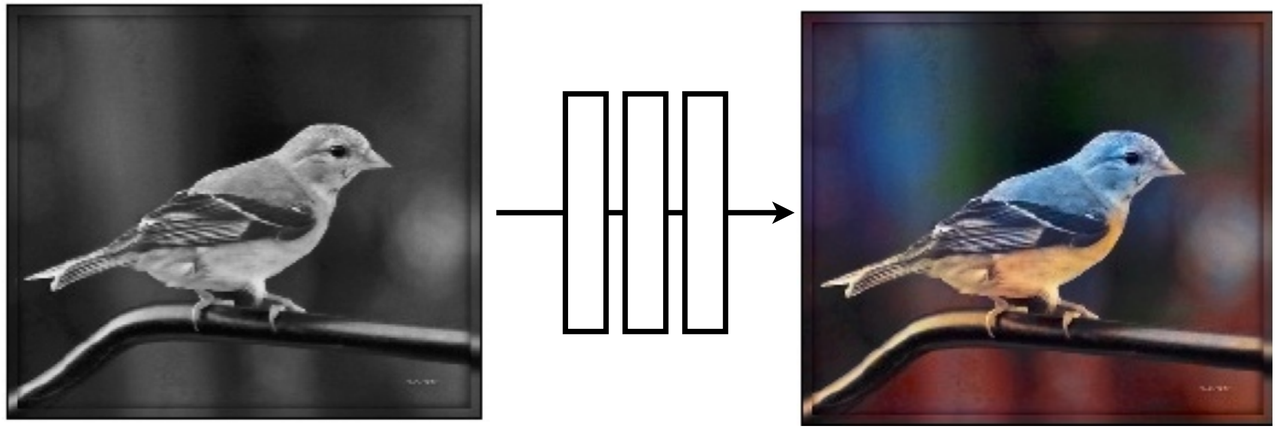


[Johnson, Alahi, Li, ECCV 2016]

L2 regression

Designing loss functions

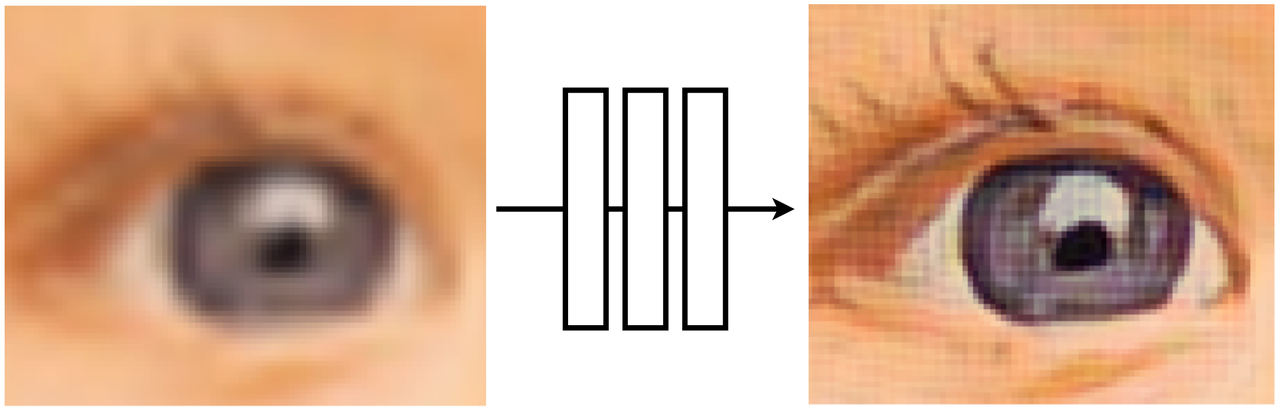
Image colorization



[Zhang, Isola, Efros, ECCV 2016]

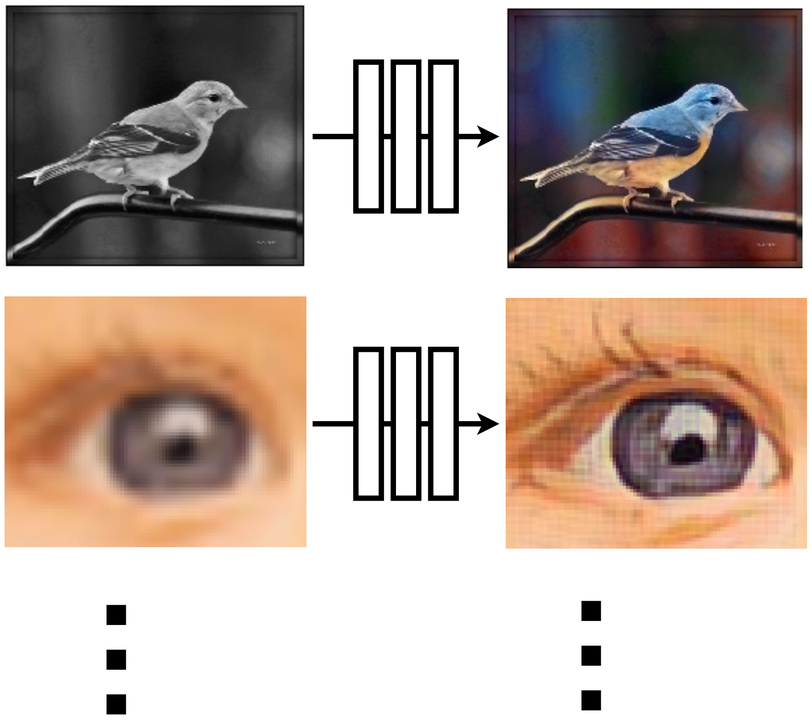
Cross entropy objective,
with colorfulness term

Super-resolution



[Johnson, Alahi, Li, ECCV 2016]

Deep feature covariance
matching objective



Universal loss?

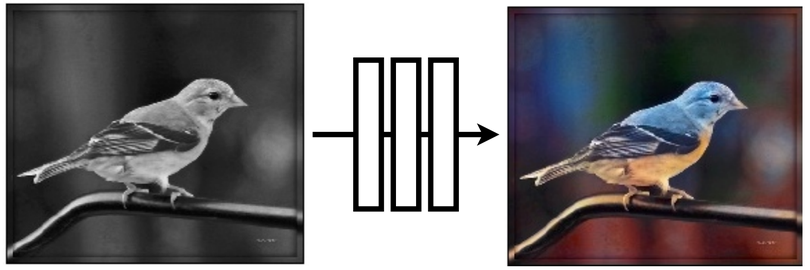
Learner

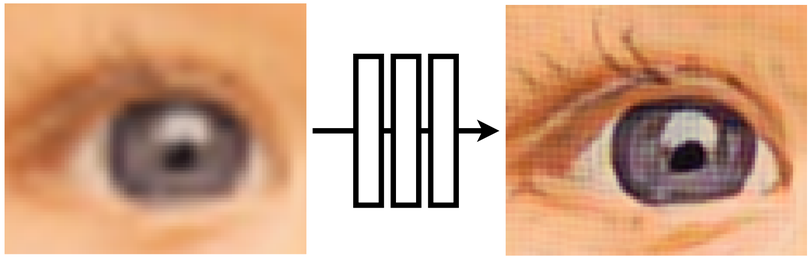
Objective

Hypothesis space

Optimizer

Generated images





“Generative Adversarial Network” (GANs)

Generated vs Real (classifier)

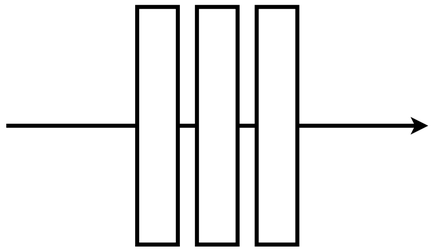
Real photos

...

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

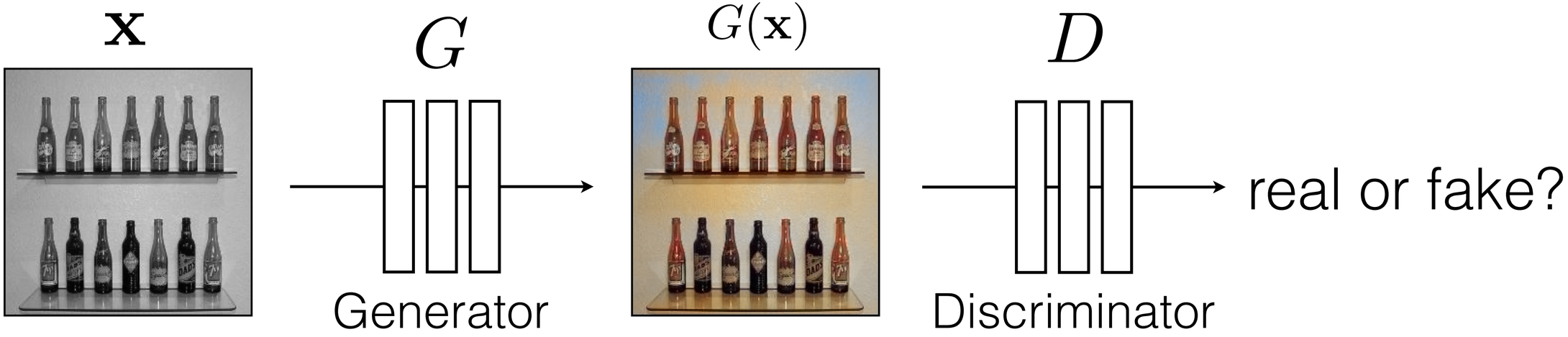
$\mathbf{x}$

G



Generator

$G(\mathbf{x})$



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

D tries to identify the fakes

$\mathbf{x}$

G

$G(\mathbf{x})$

D

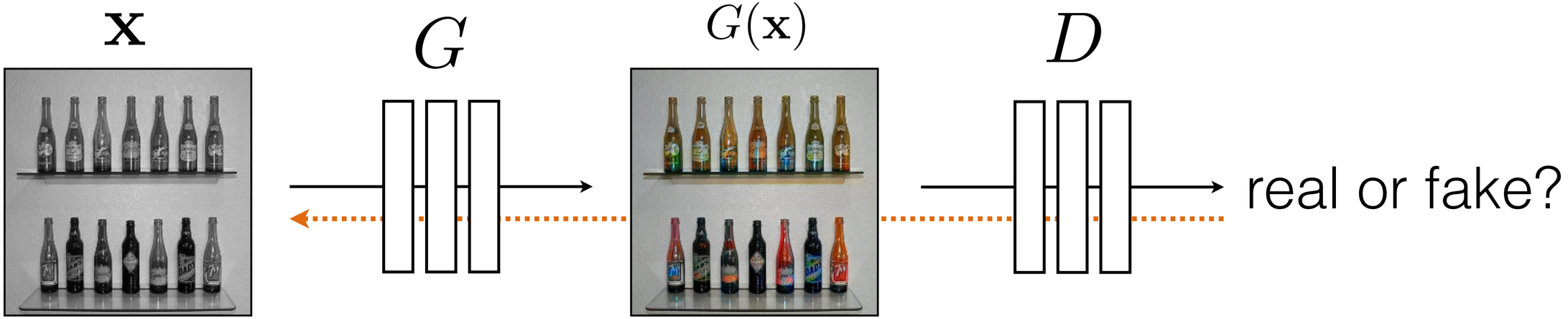
fake (0.9)

$\mathbf{y}$

D

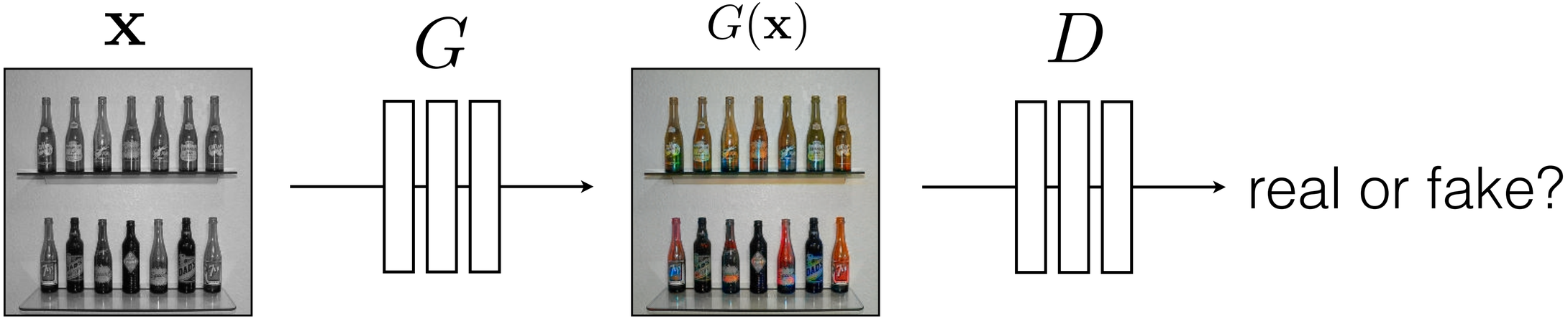
real (0.1)

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



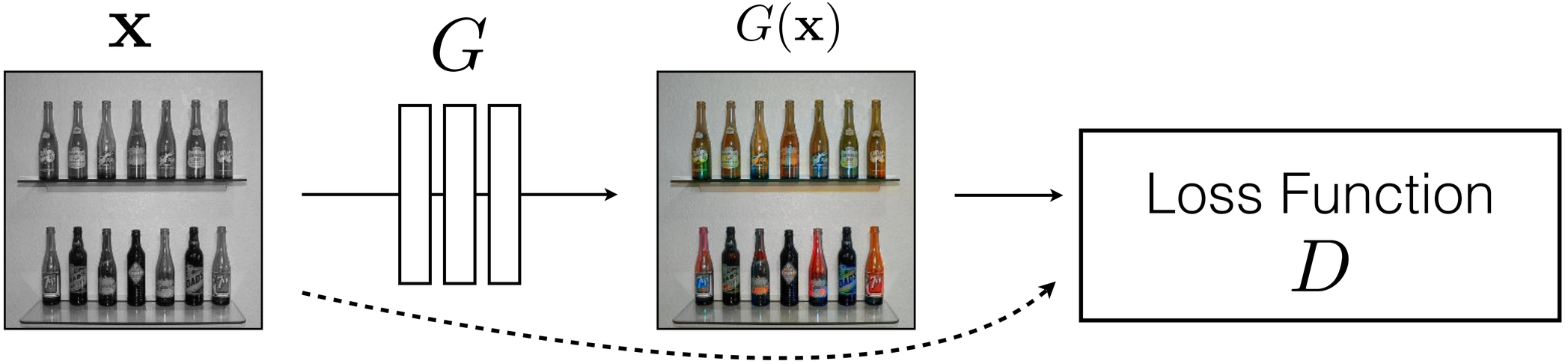
G tries to synthesize fake images that ***fool*** **D**:

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



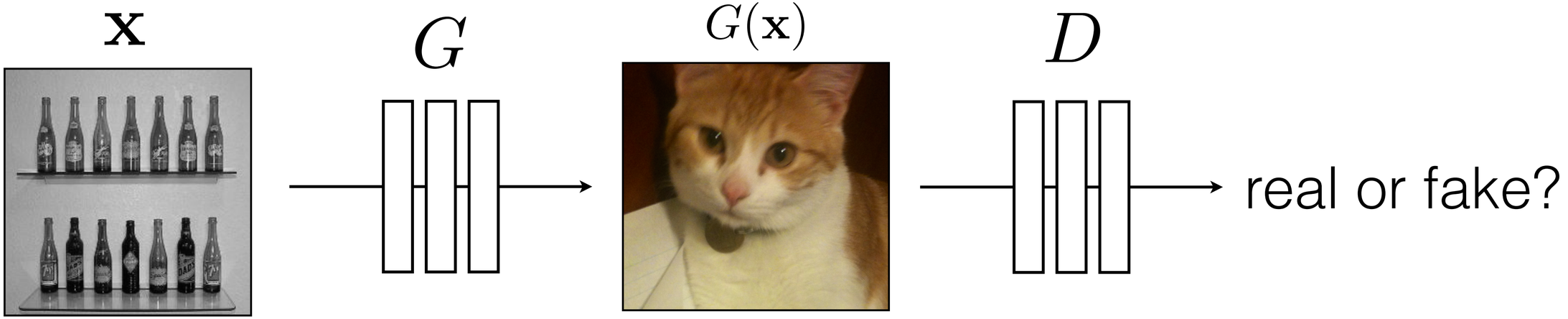
G tries to synthesize fake images that ***fool*** the ***best*** **D**:

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

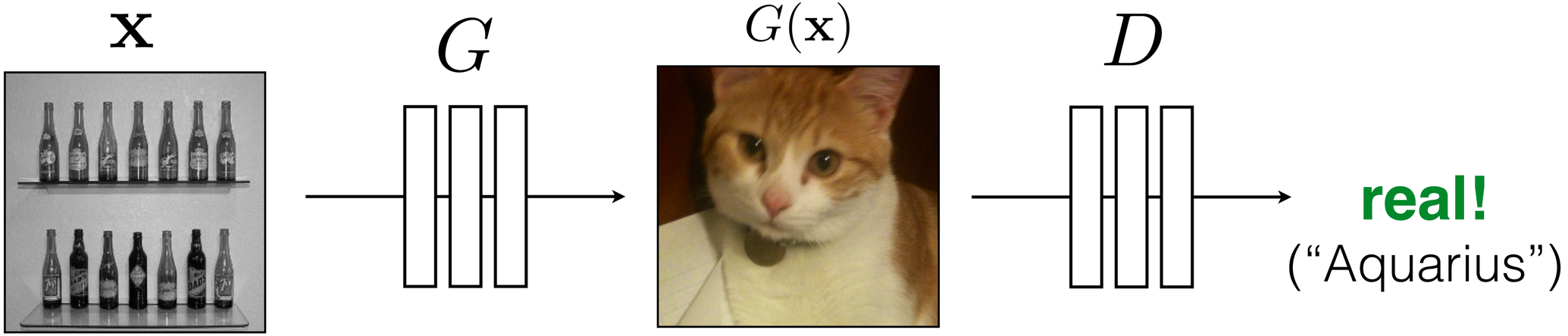


G's perspective: **D** is a loss function.

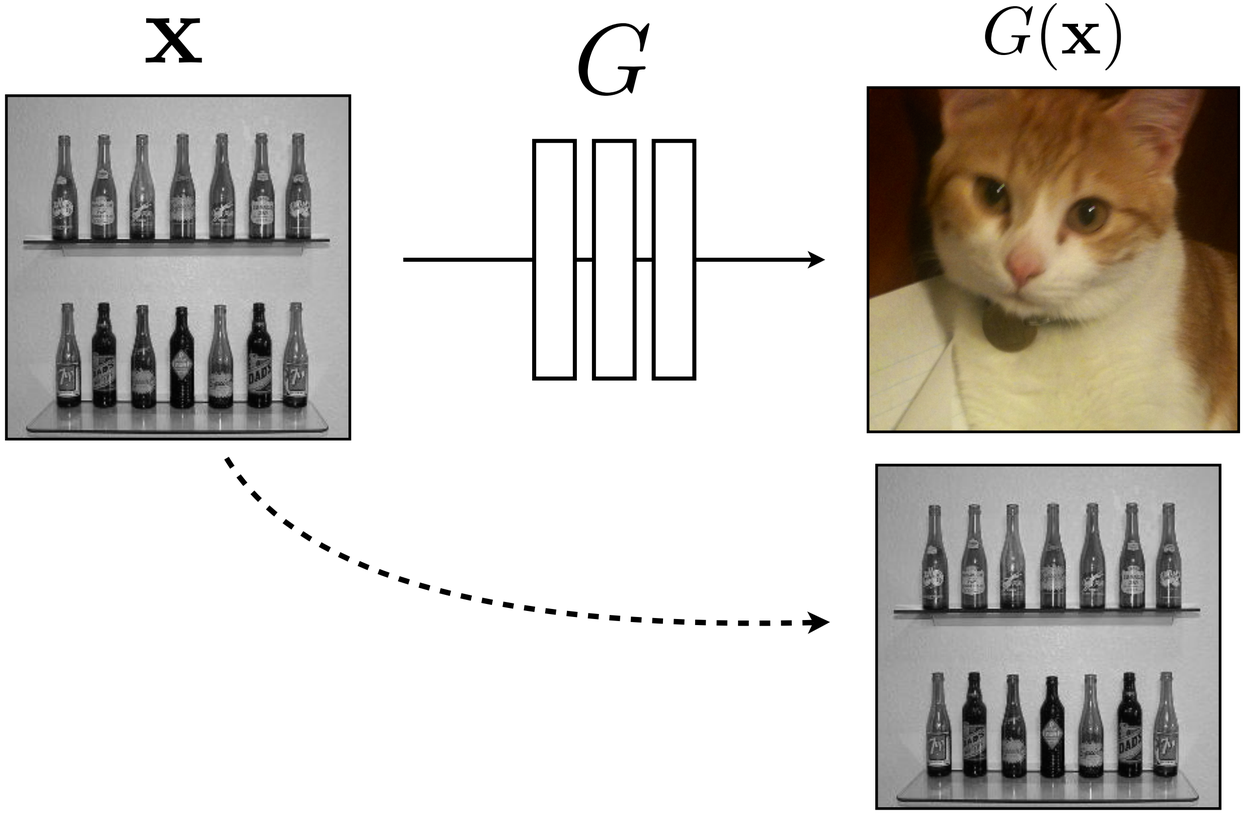
Rather than being hand-designed, it is *learned*.



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



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



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

$\mathbf{x}$

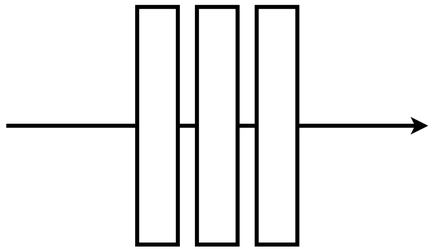
G

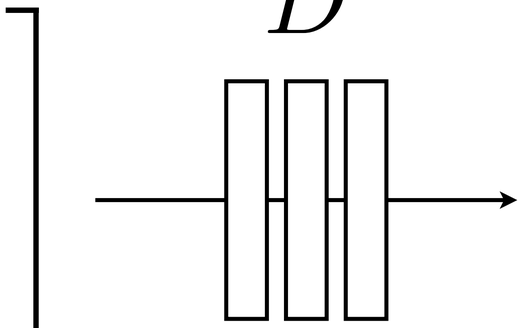
$G(\mathbf{x})$

D

real or fake *pair* ?

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





fake *pair*

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

$\mathbf{x}$

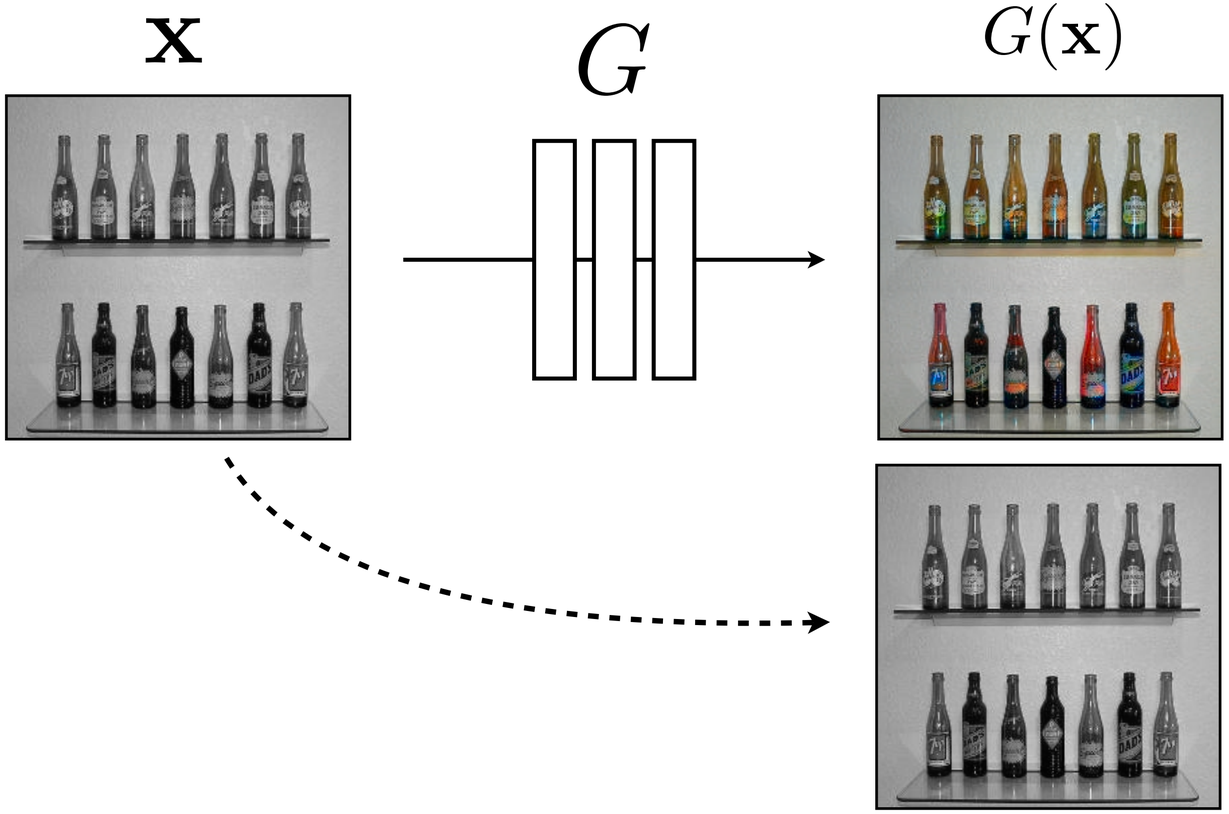
G

$G(\mathbf{x})$

D

real *pair*

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



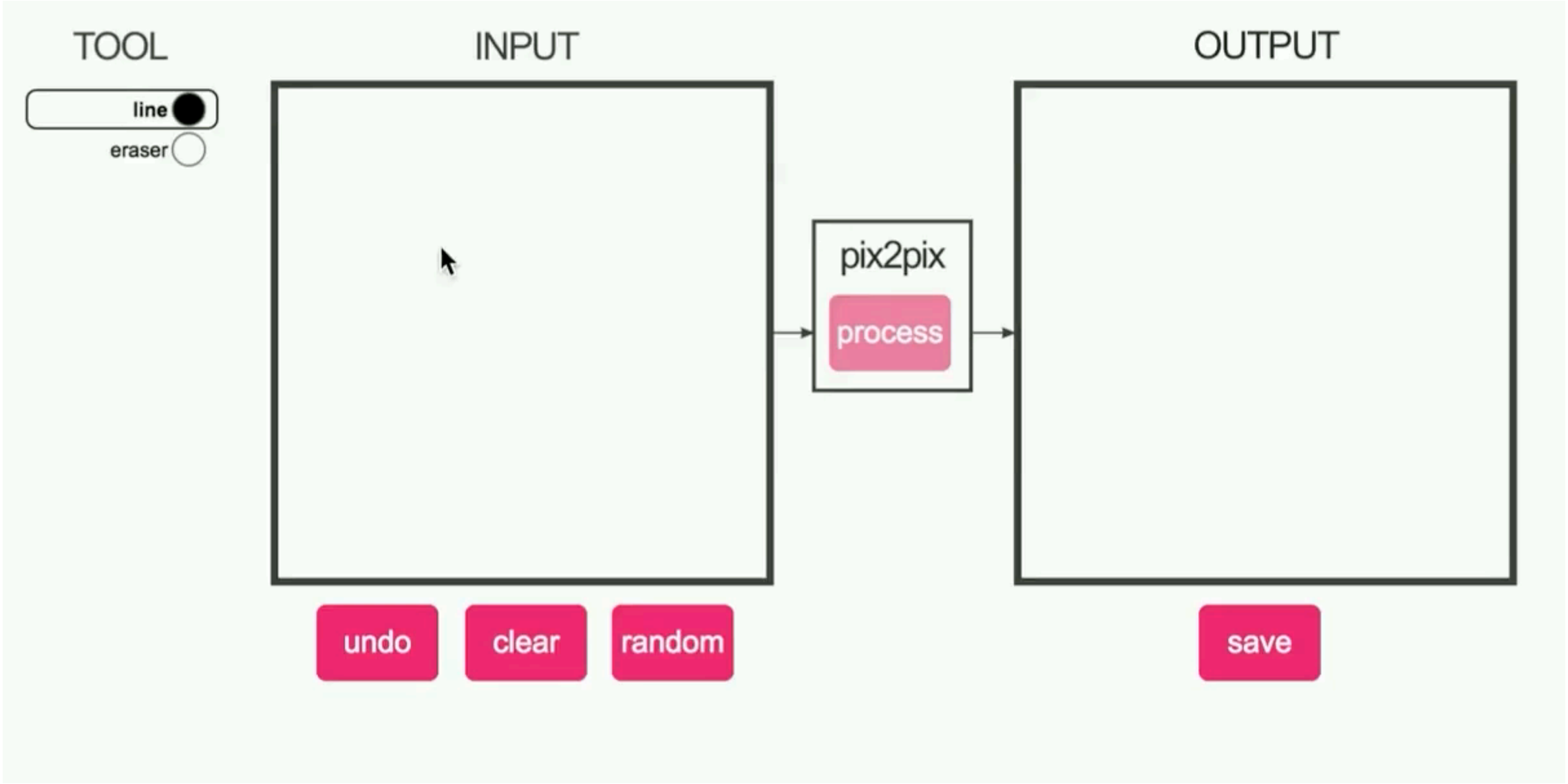
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\;\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;\;]$$

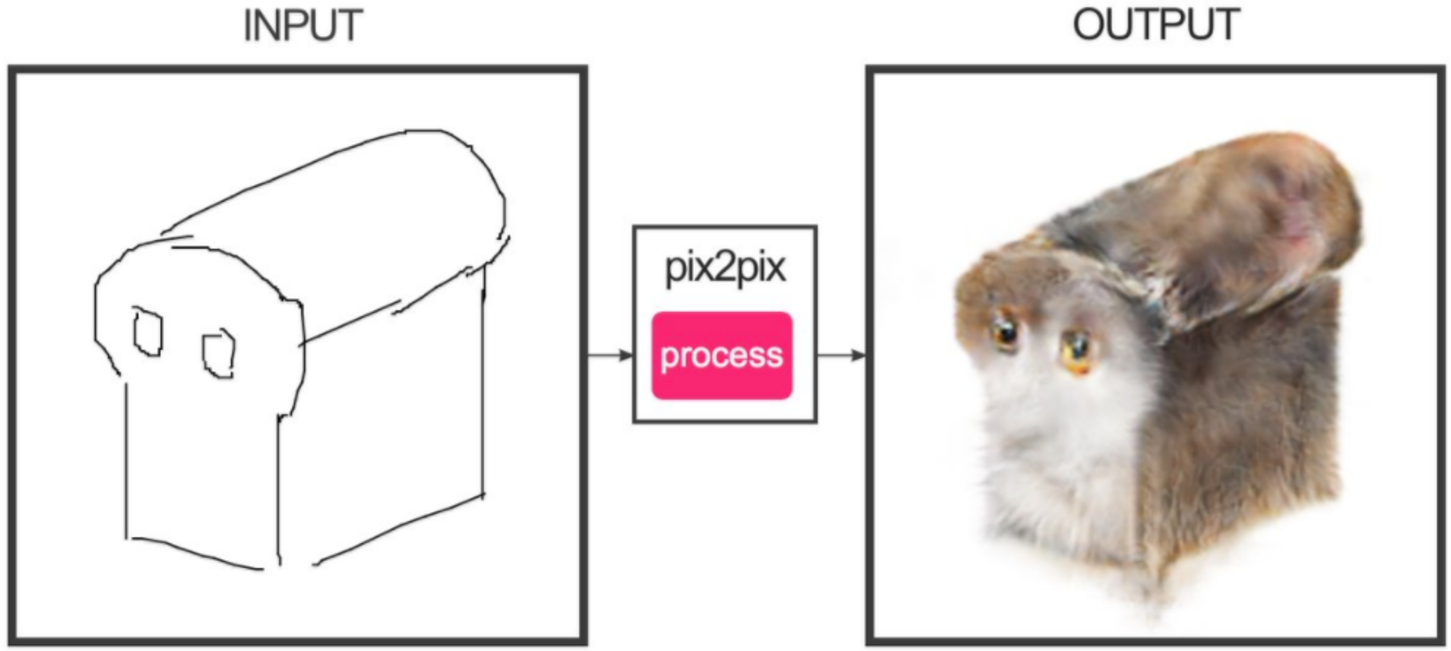
BW → Color

Input Output Input Output Input Output

Data from [Russakovsky et al. 2015]

#edges2cats [Chris Hesse]

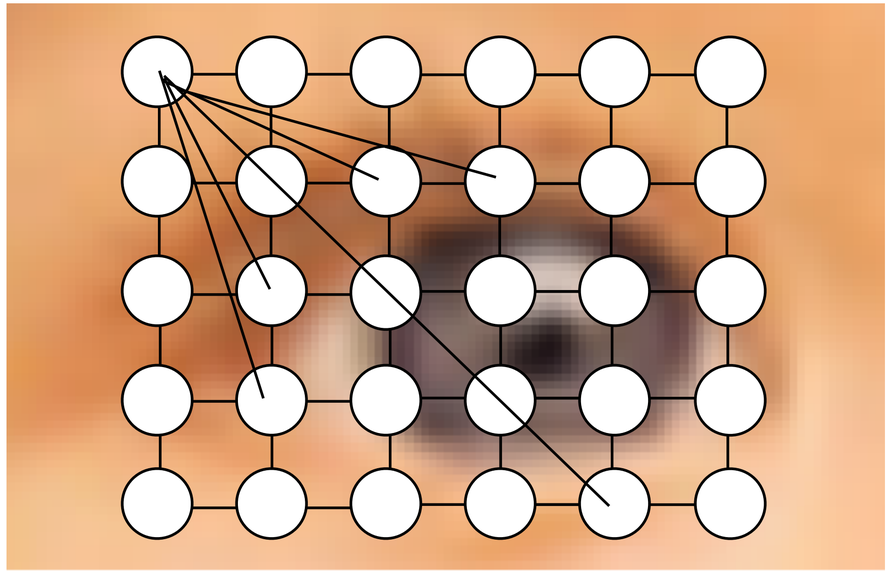




Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

Structured Prediction



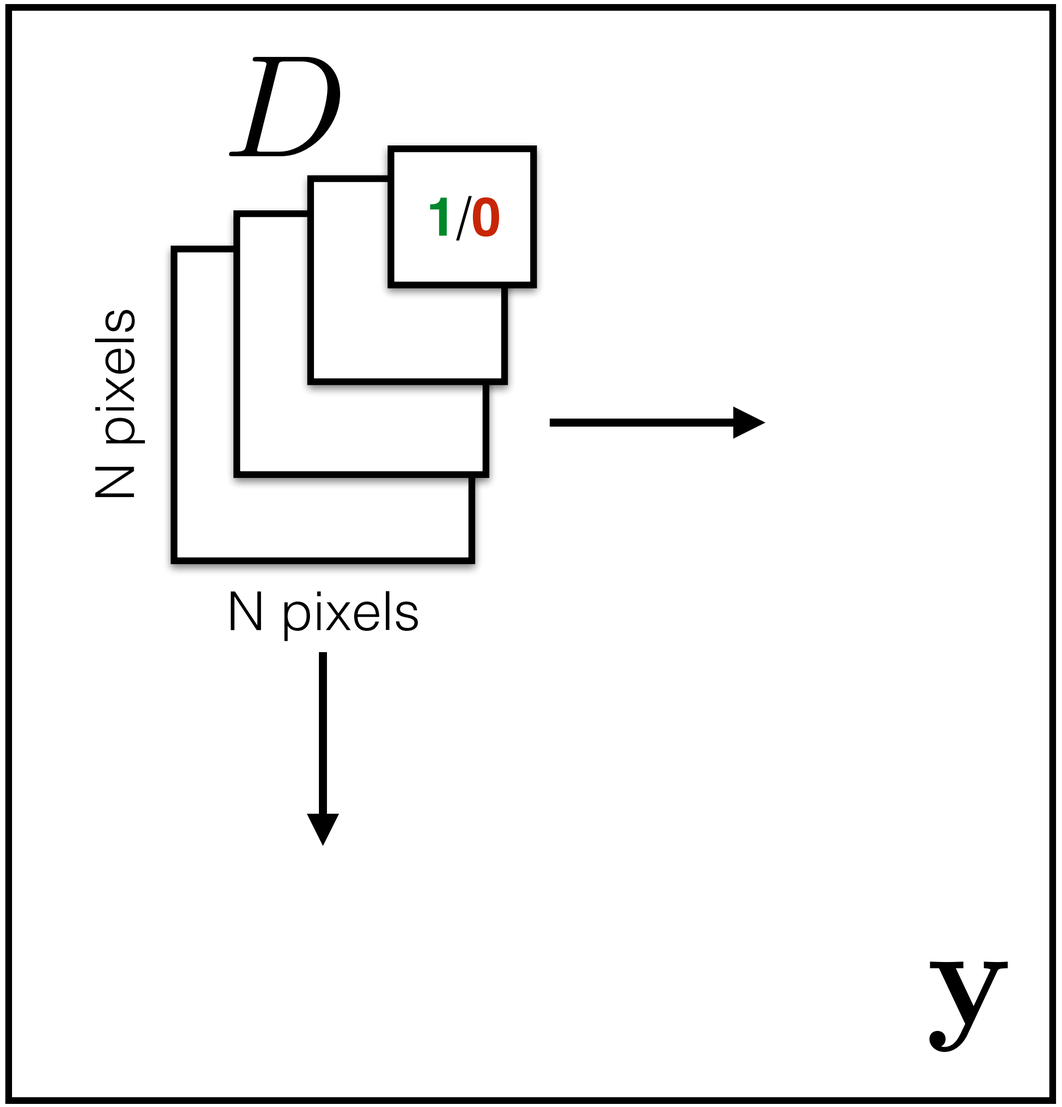
Model *joint* configuration of all pixels

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

A GAN, with sufficient capacity, samples from the full joint distribution when perfectly optimized.

Most generative models have this property! Give them **sufficient capacity** and **infinite data**, and they are the complete solution to prediction problems.

Shrinking the capacity: Patch Discriminator

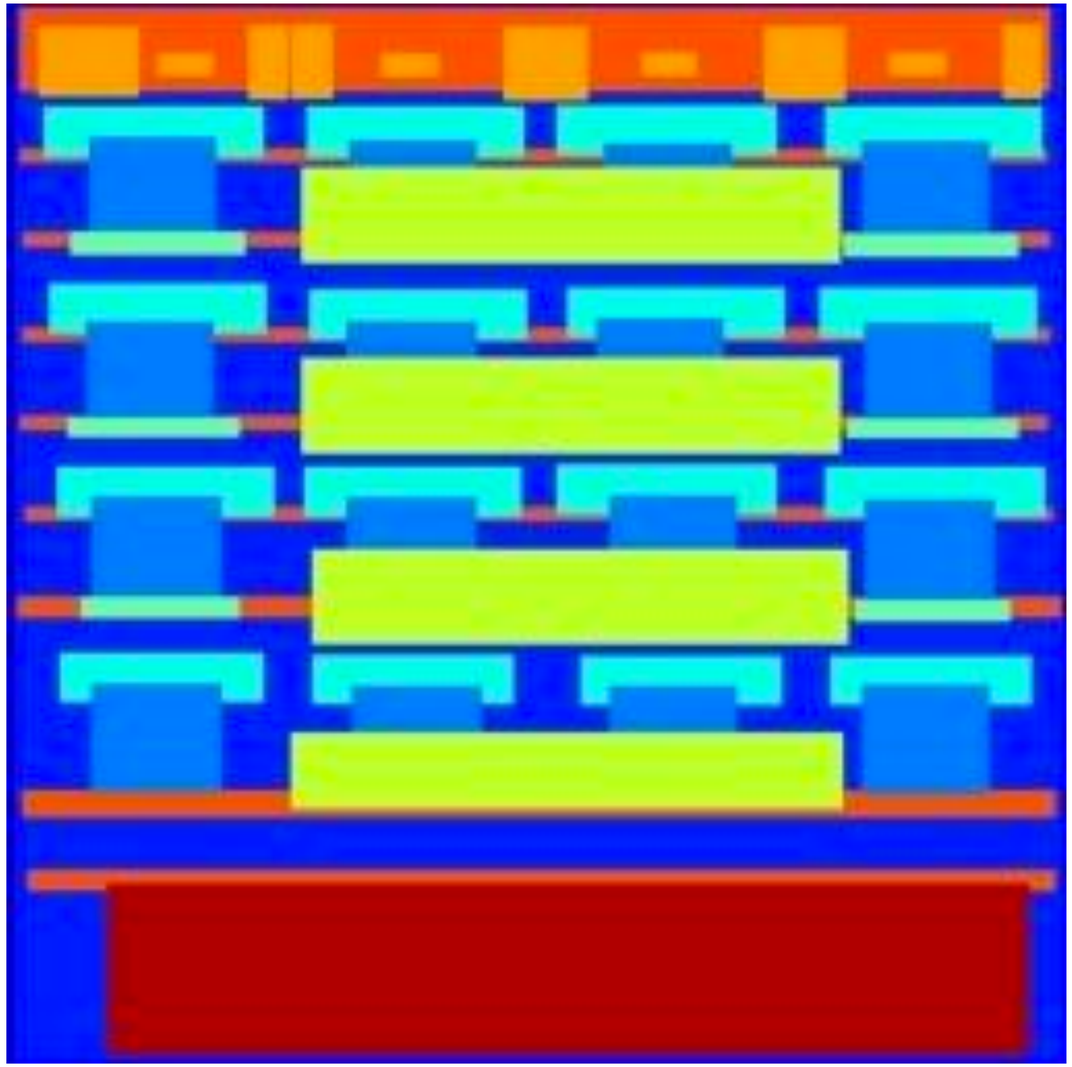


Rather than penalizing if output *image* looks fake, penalize if each overlapping *patch* in output looks fake

[Li & Wand 2016]
[Shrivastava et al. 2017]
[Isola et al. 2017]

Labels → Facades

Input

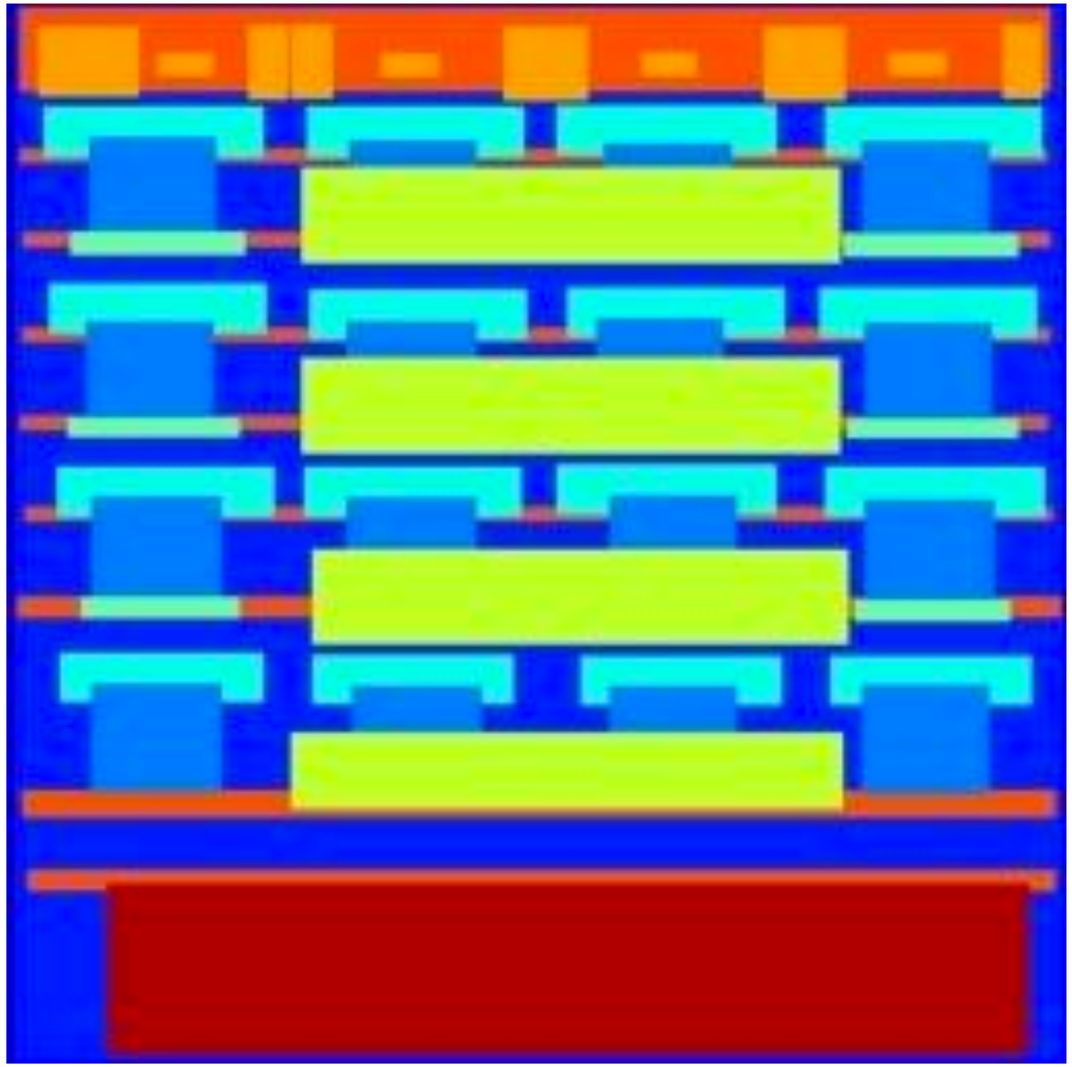


1x1 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

Input

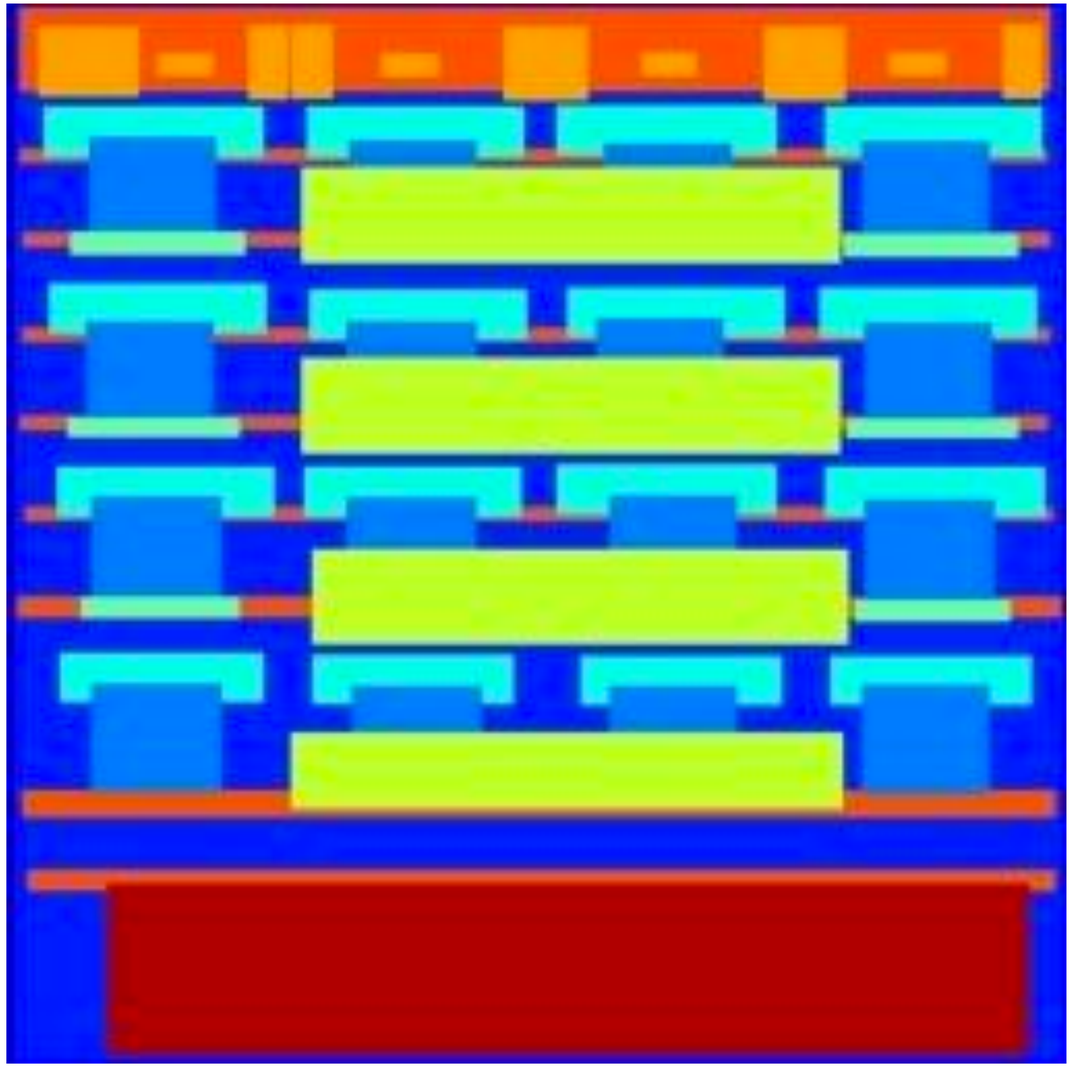


16x16 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

Input

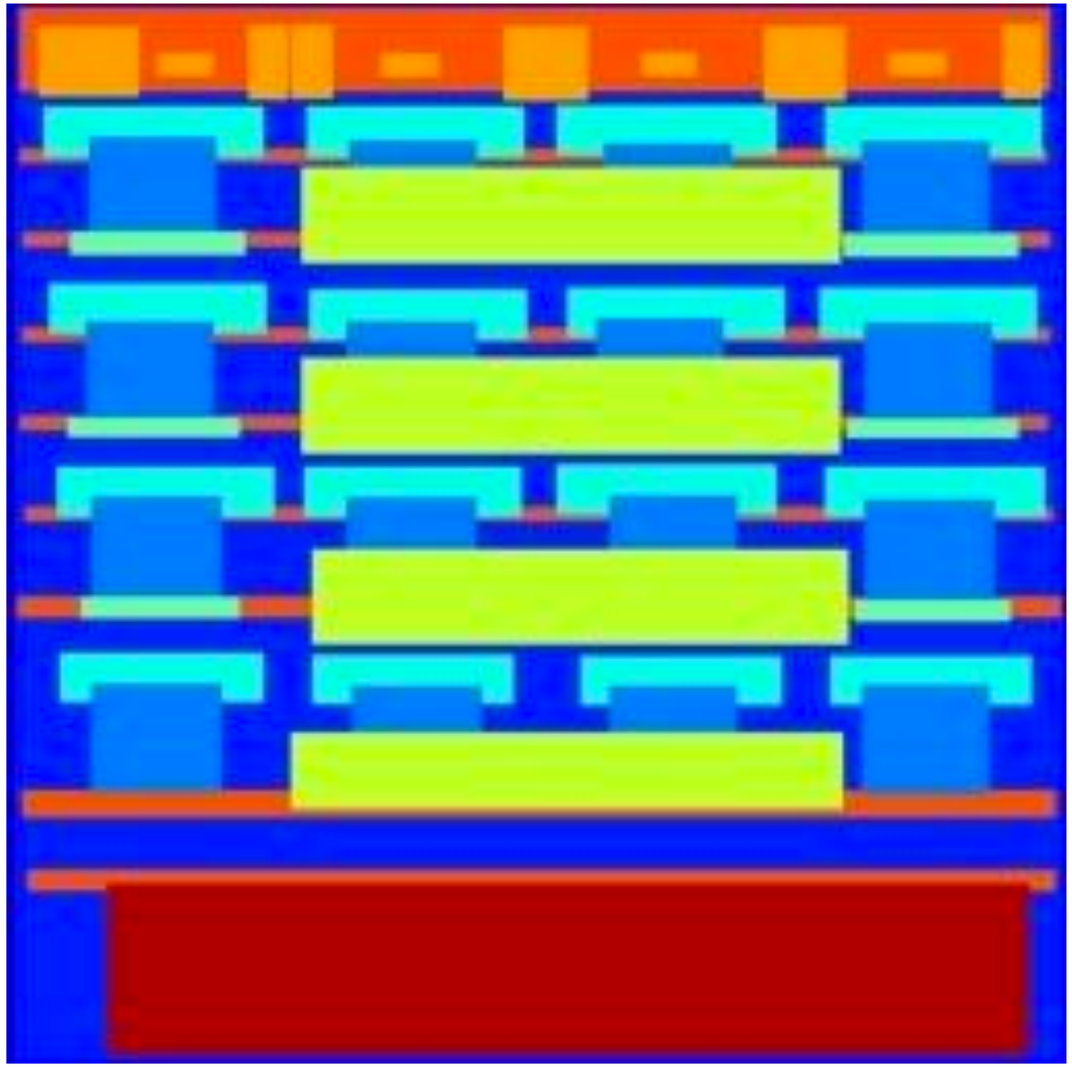


70x70 Discriminator

Data from [Tylecek, 2013]

Labels → Facades

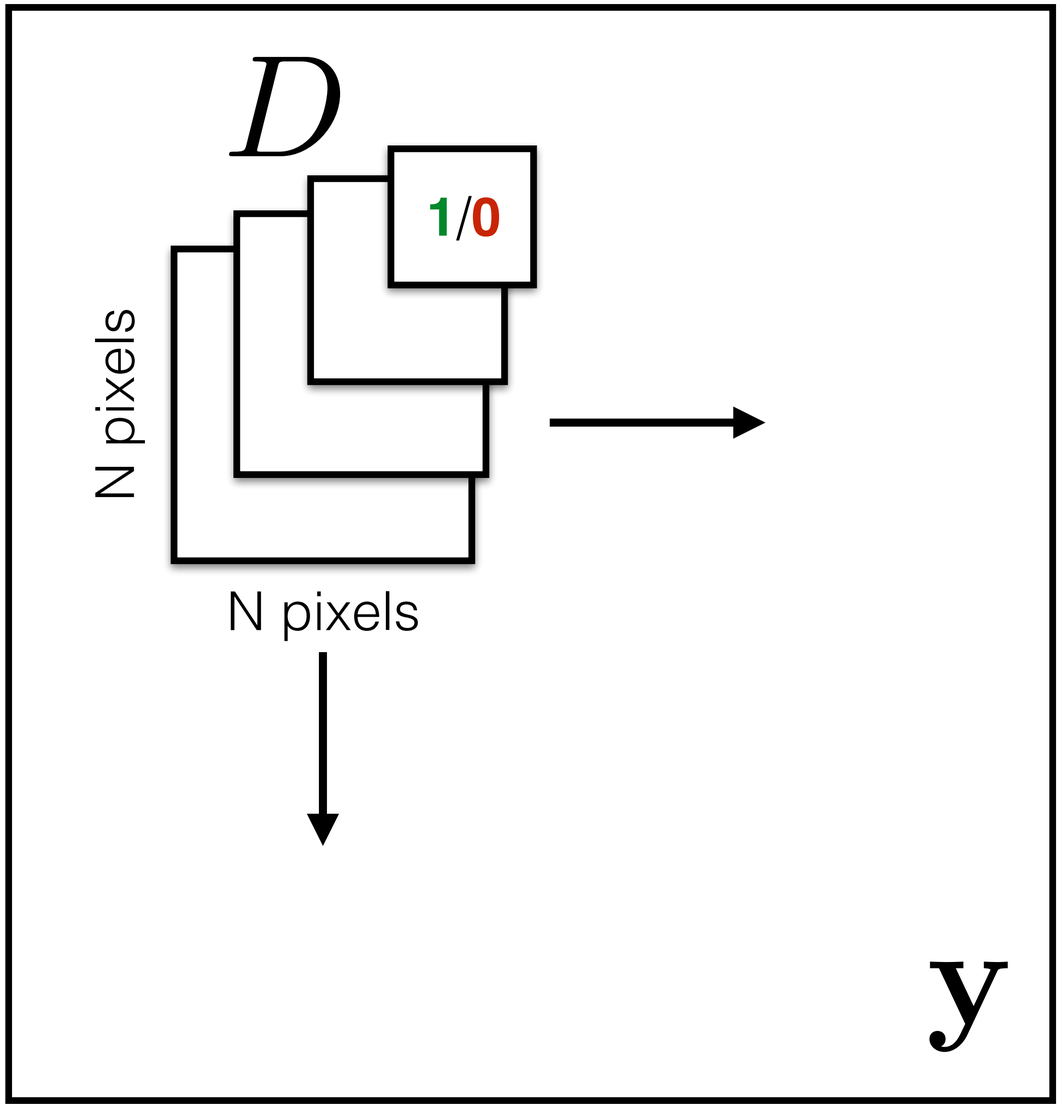
Input



Full image Discriminator

Data from [Tylecek, 2013]

Patch Discriminator



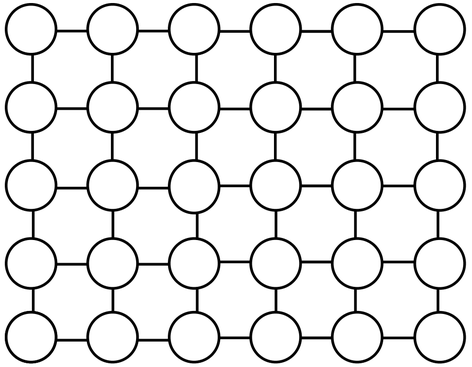
Rather than penalizing if output *image* looks fake, penalize if each overlapping *patch* in output looks fake

- Faster, fewer parameters
- More supervised observations
- Applies to arbitrarily large images

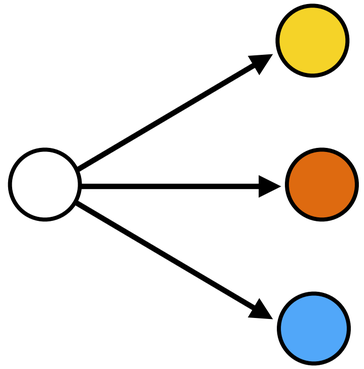
Properties of generative models

1. Model high-dimensional, structured output

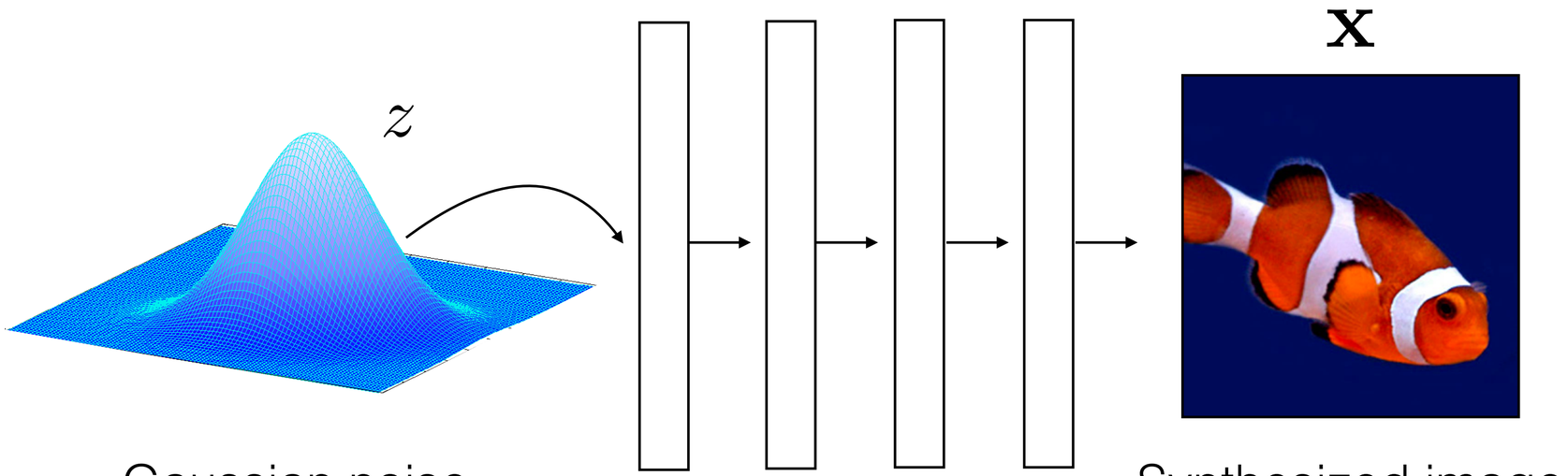
   —> Use a deep net, D, to model output!



2. Model uncertainty; a whole distribution of possible outputs



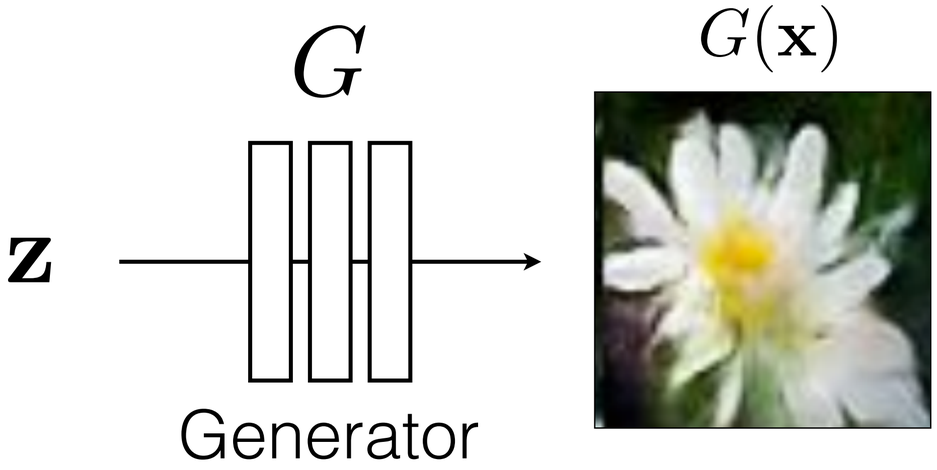
Can we generate images from scratch?



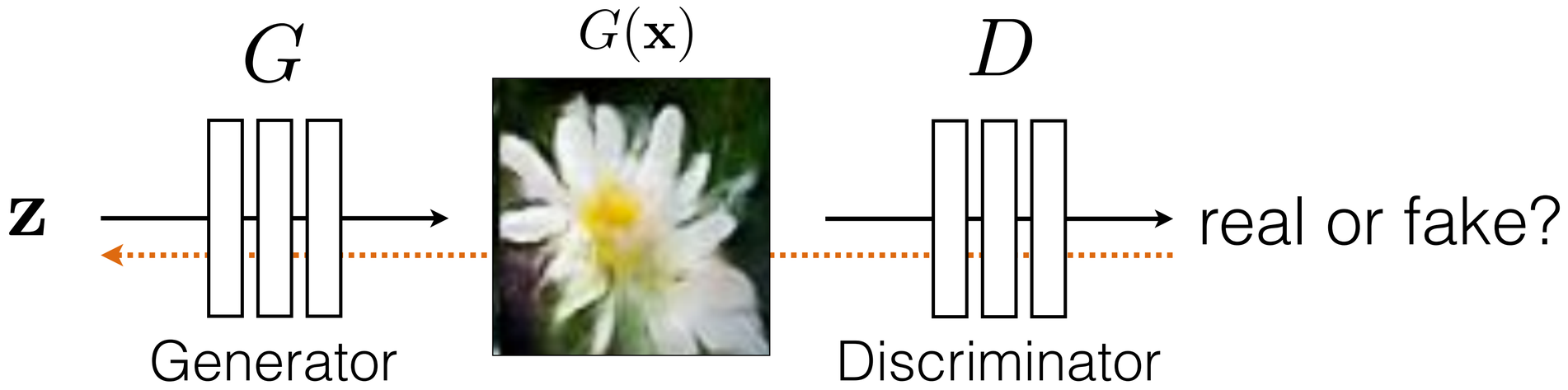
Gaussian noise

Synthesized image

$$z \sim \mathcal{N}(\vec{0}, 1)$$



[Goodfellow et al., 2014]



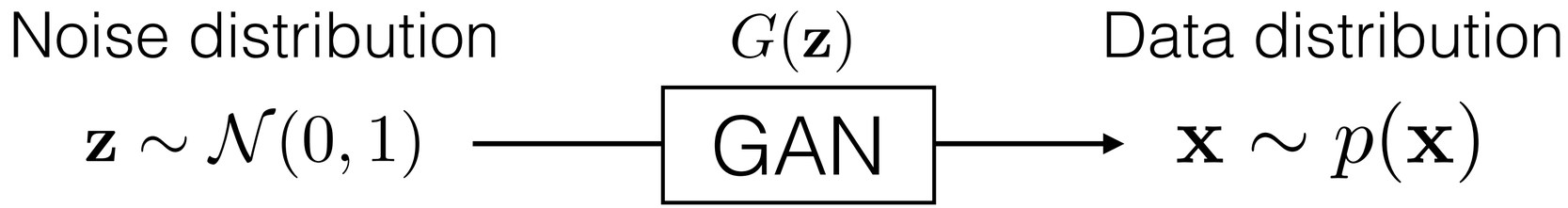
G tries to synthesize fake images that fool **D**

D tries to identify the fakes

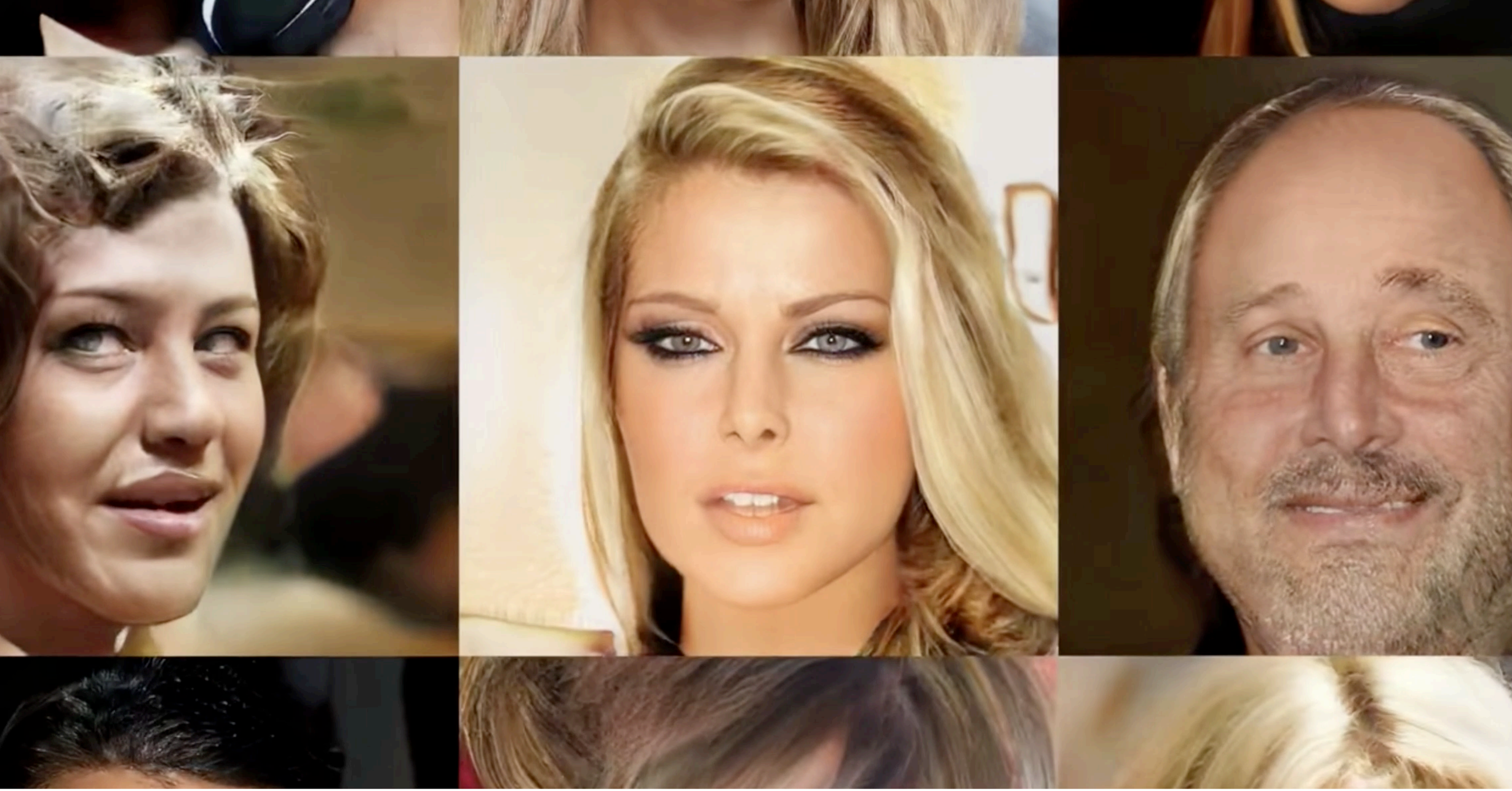
[Goodfellow et al., 2014]

GANs are implicit generative models

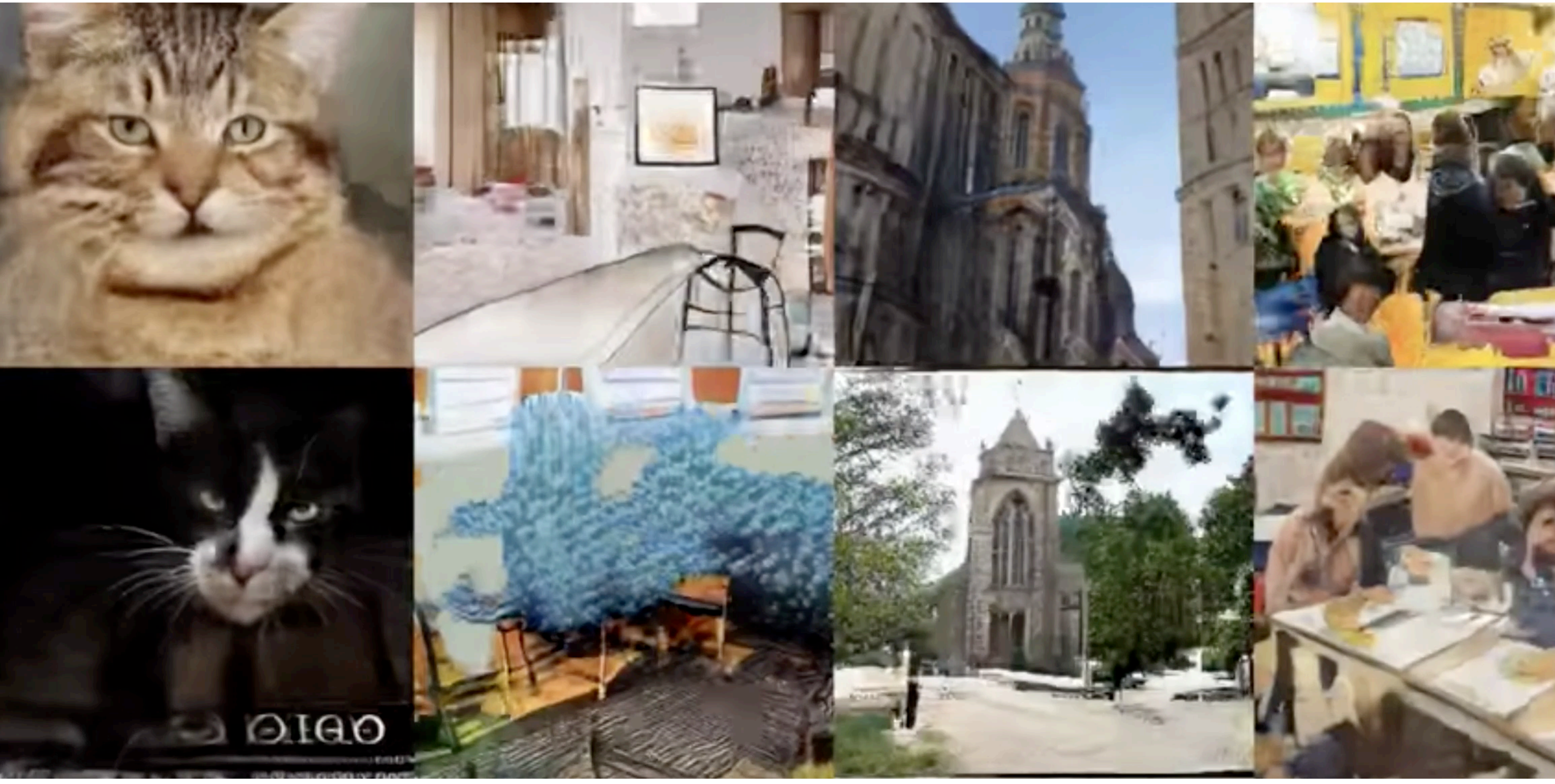
$p(\mathbf{x})$ ⟵ "generative model" of the data $\mathbf{x}$



$G(\mathbf{z}) \sim p(\mathbf{x})$ ⟵ Samples from a perfectly optimized GAN are samples from the data distribution



Progressive GAN [Karras et al., 2018]

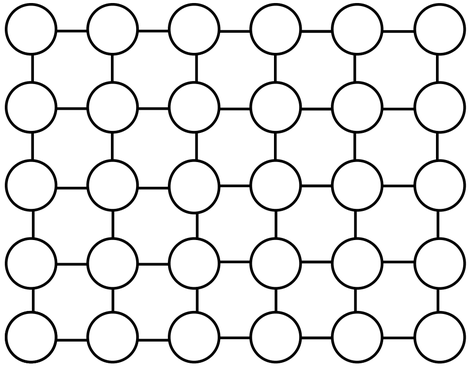


Progressive GAN [Karras et al., 2018]

Properties of generative models

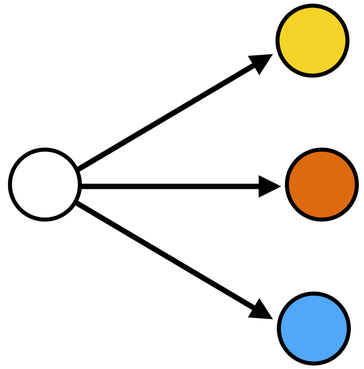
1. Model high-dimensional, structured output

   —> Use a deep net, D, to model output!

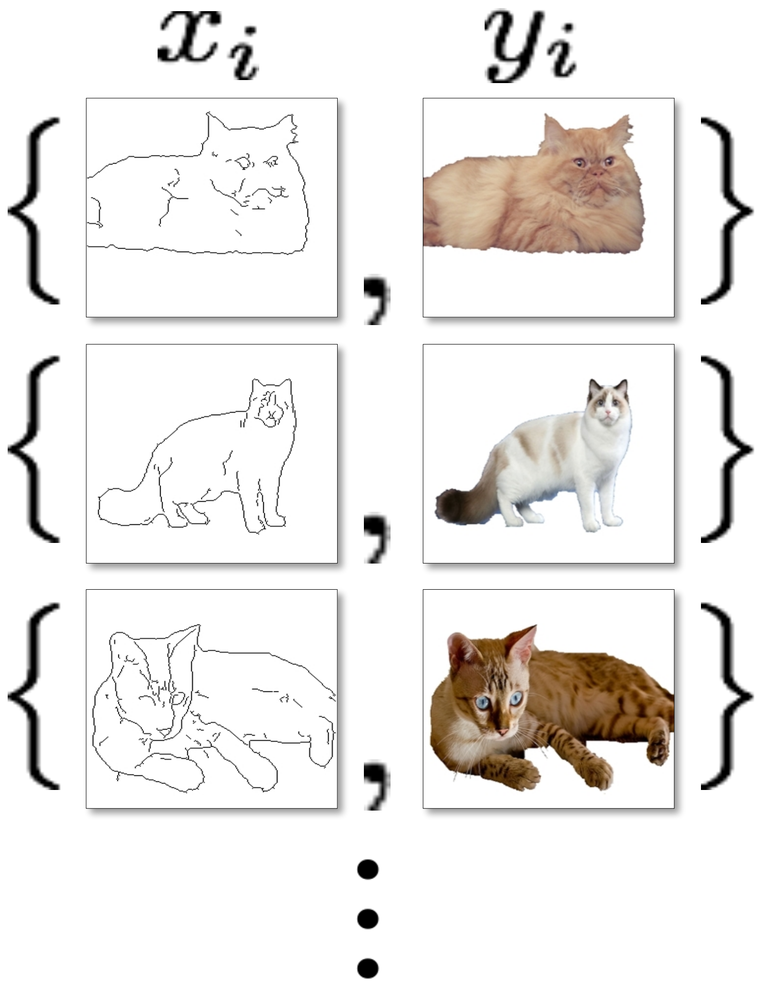


2. Model uncertainty; a whole distribution of possible outputs

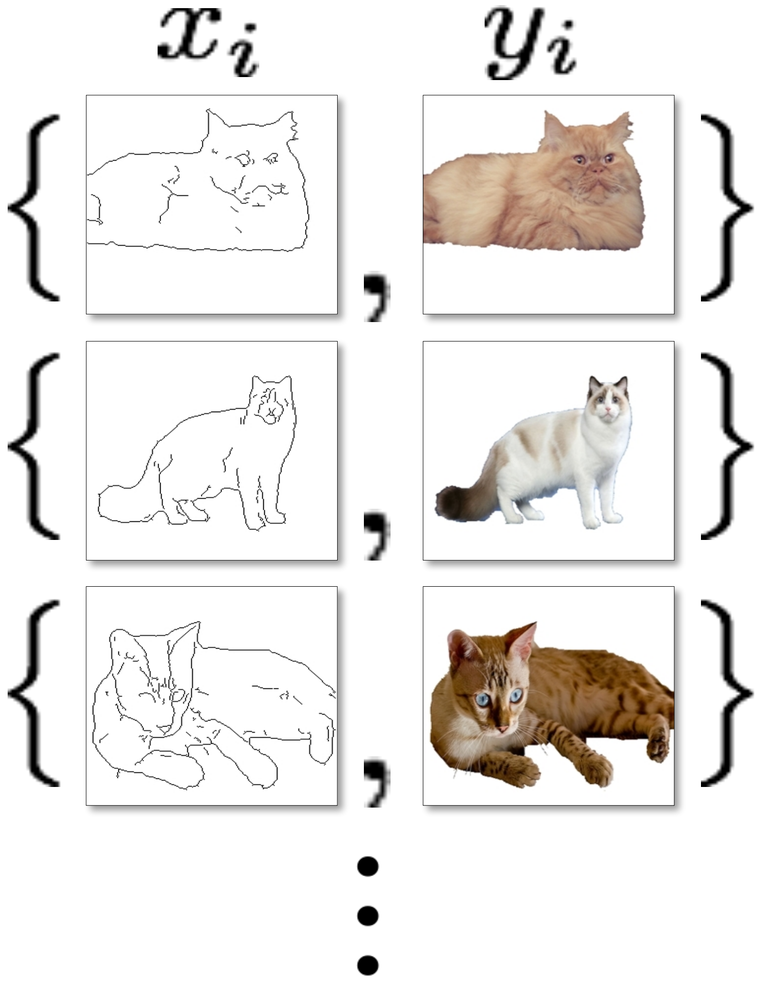
   —> Generator is stochastic, learns to match data distribution



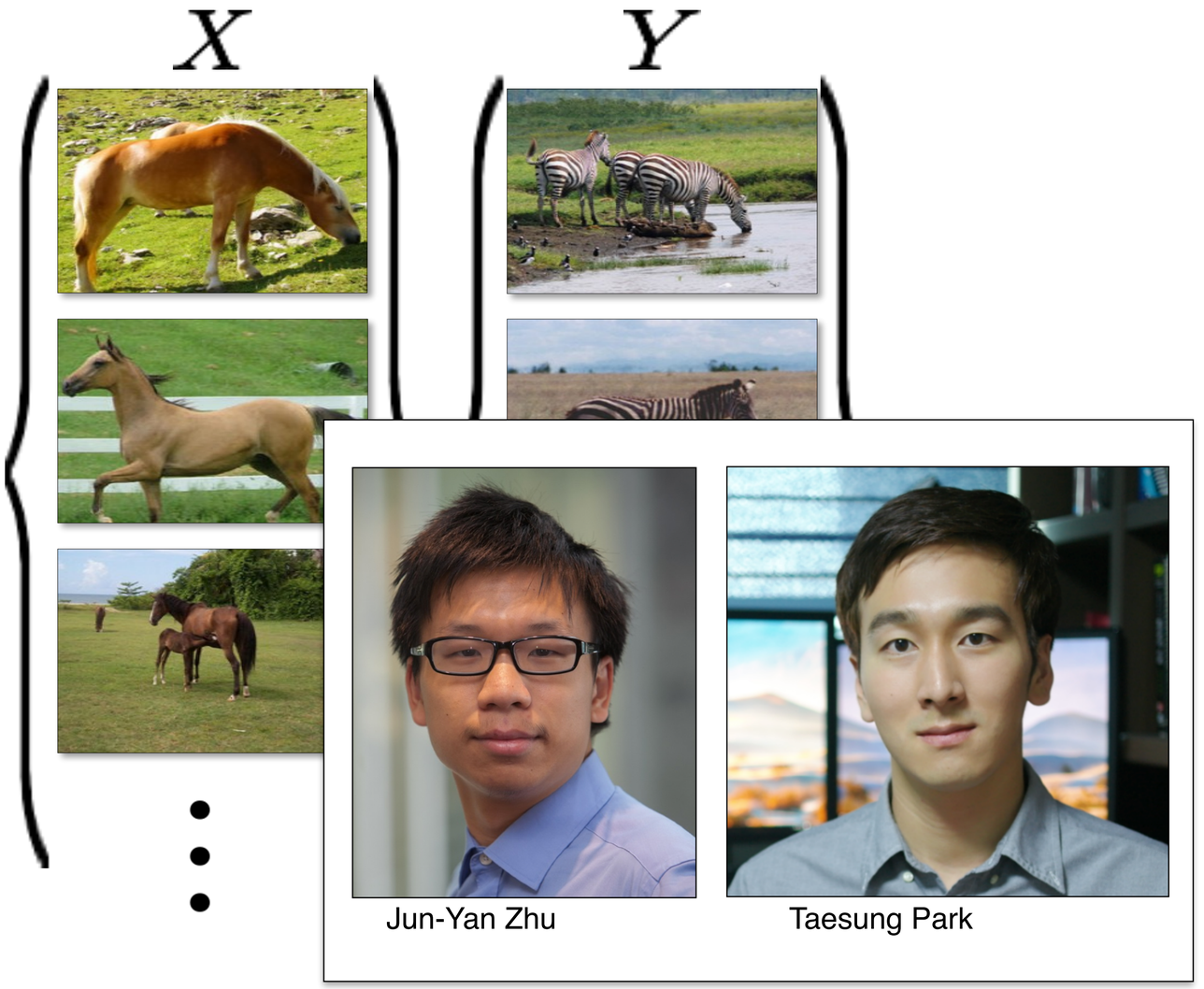
Paired data

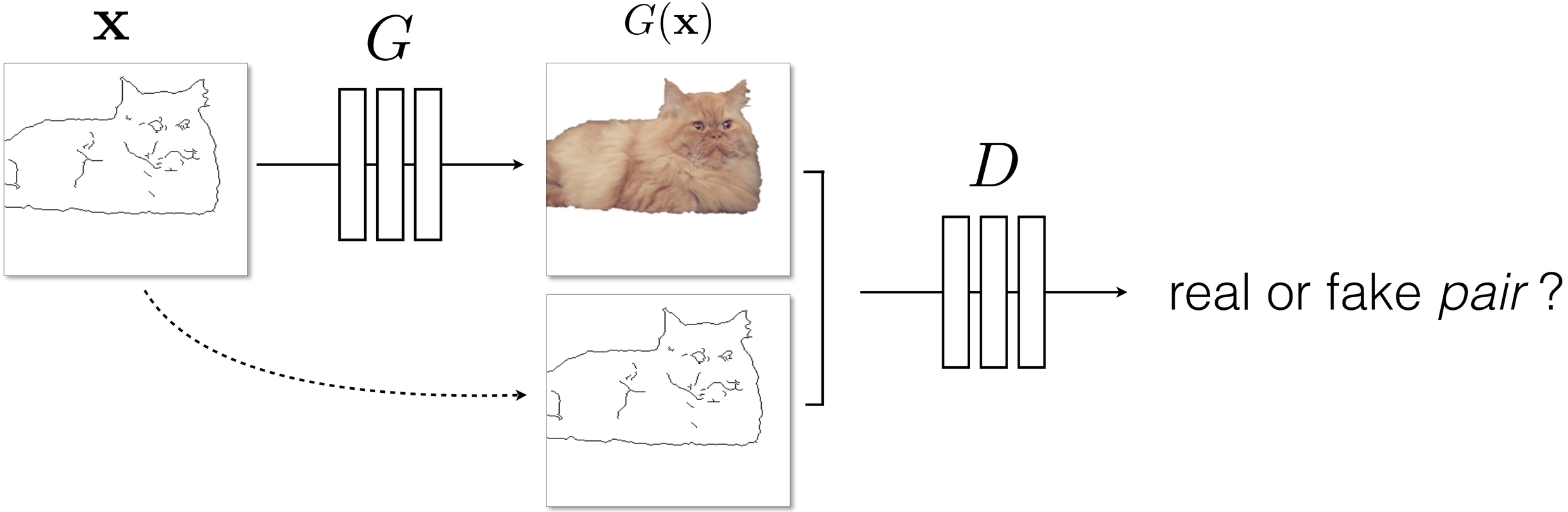


Paired data

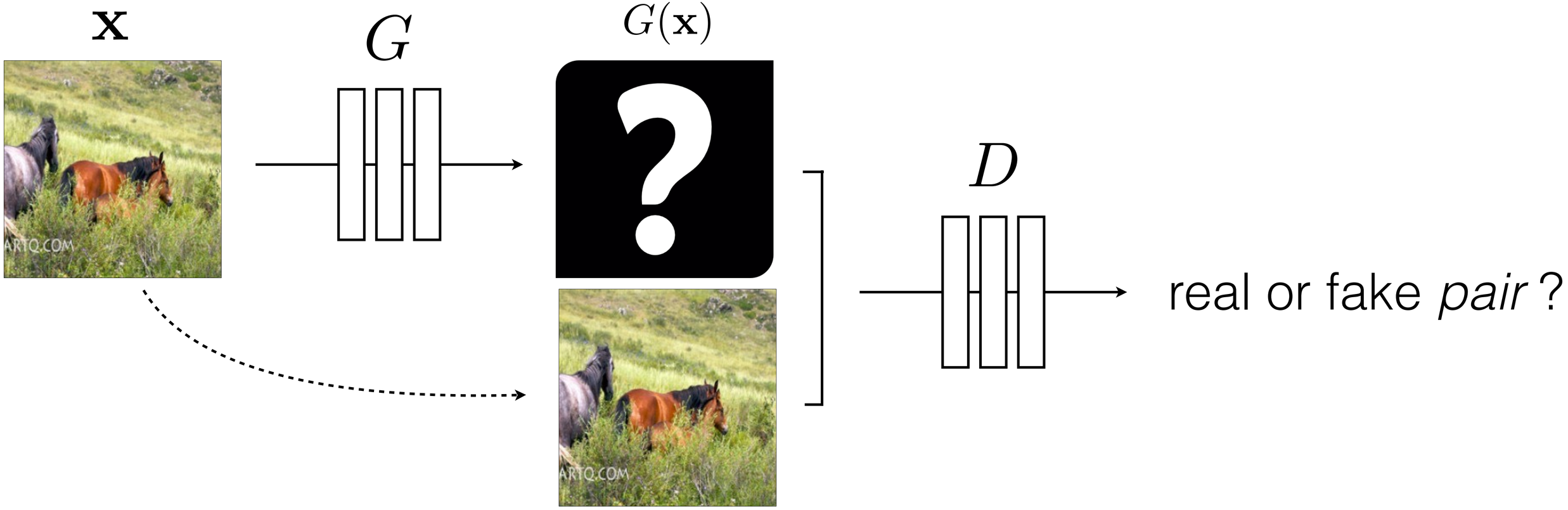


Unpaired data



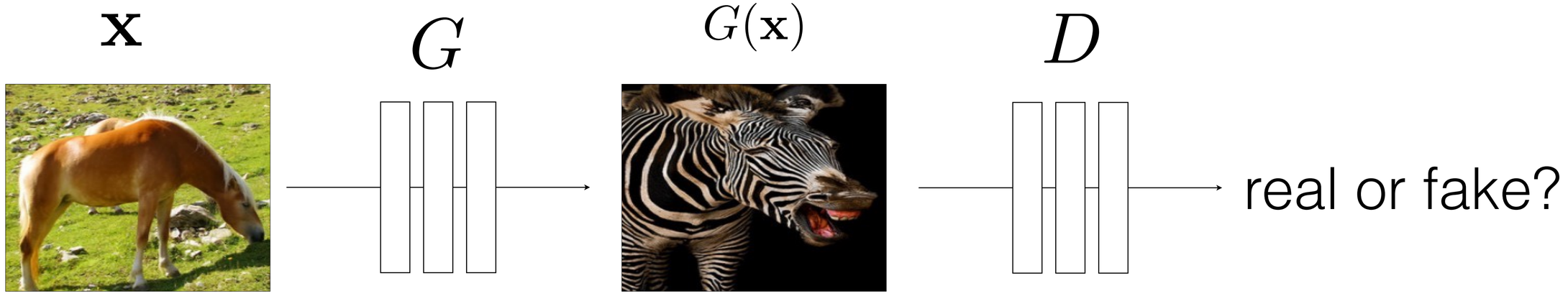


$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\;\; \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x}, \mathbf{y}))\;\;]$$



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

No input-output pairs!



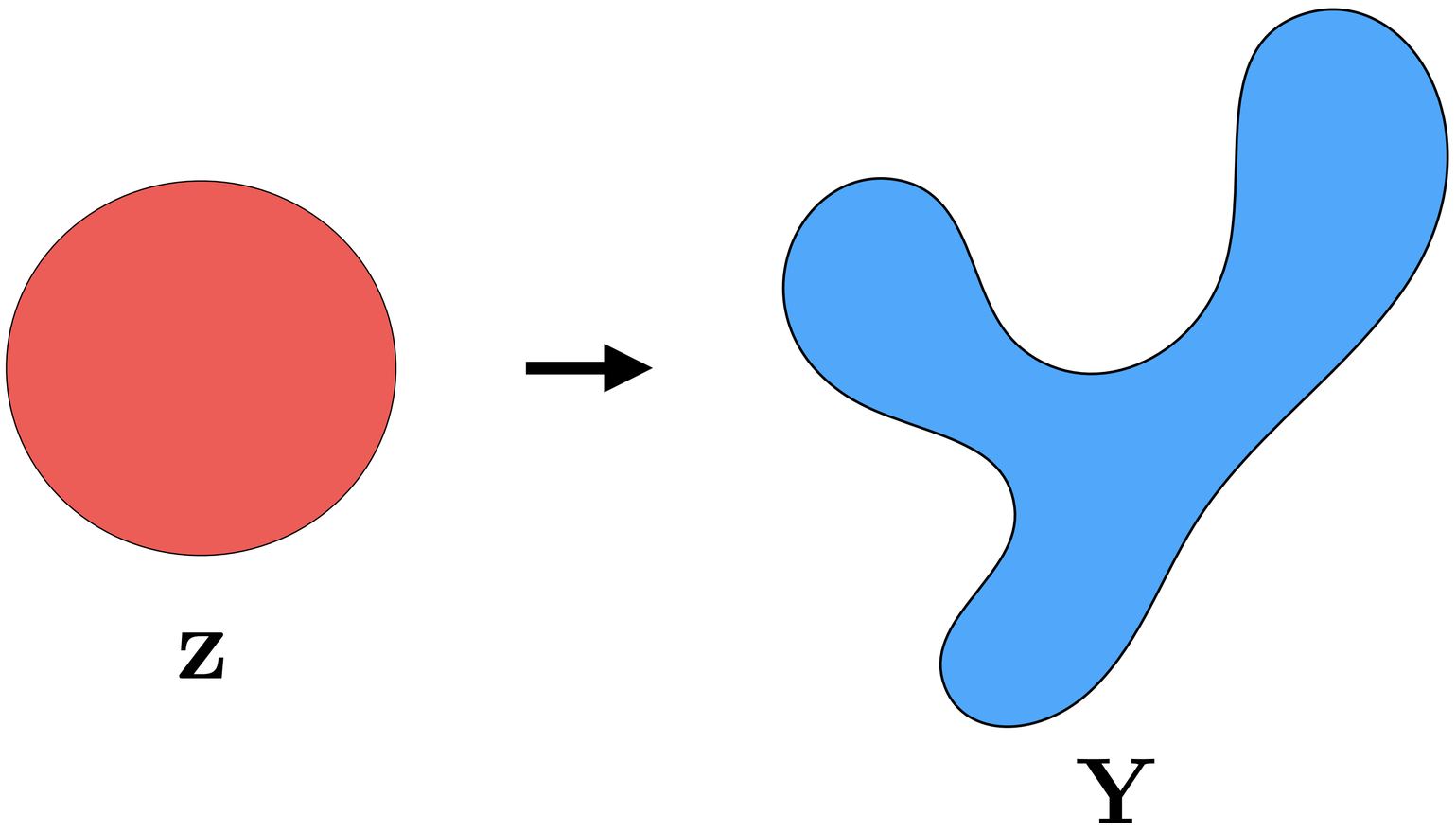
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\;]$$

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

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

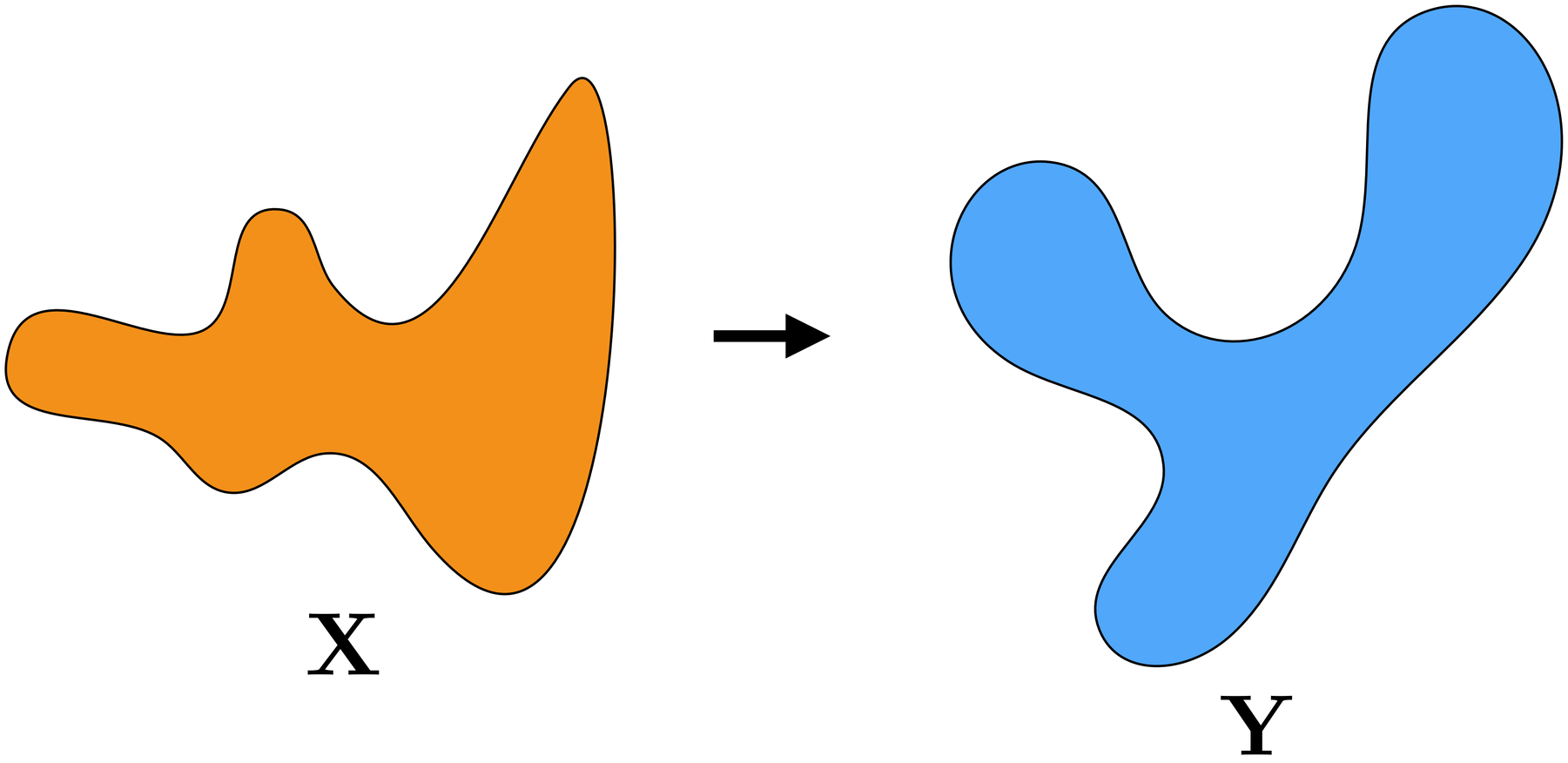
Gaussian

Target distribution



Horses

Zebras

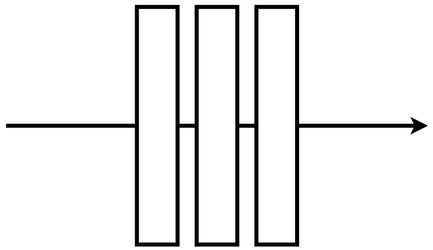


X

Y

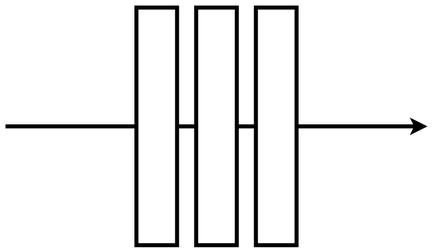
$\mathbf{x}$

G

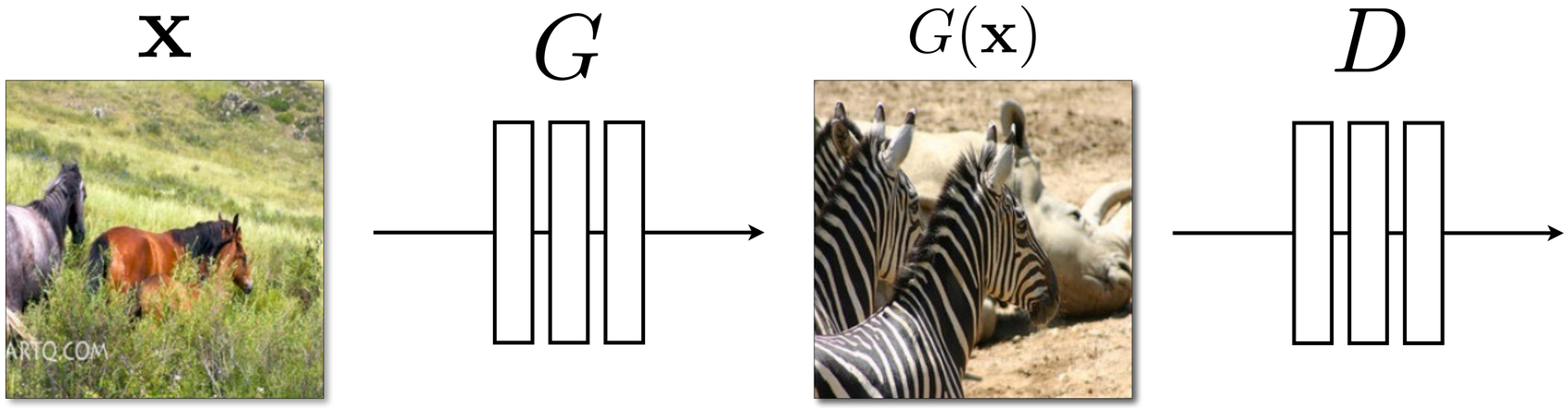


$G(\mathbf{x})$

D

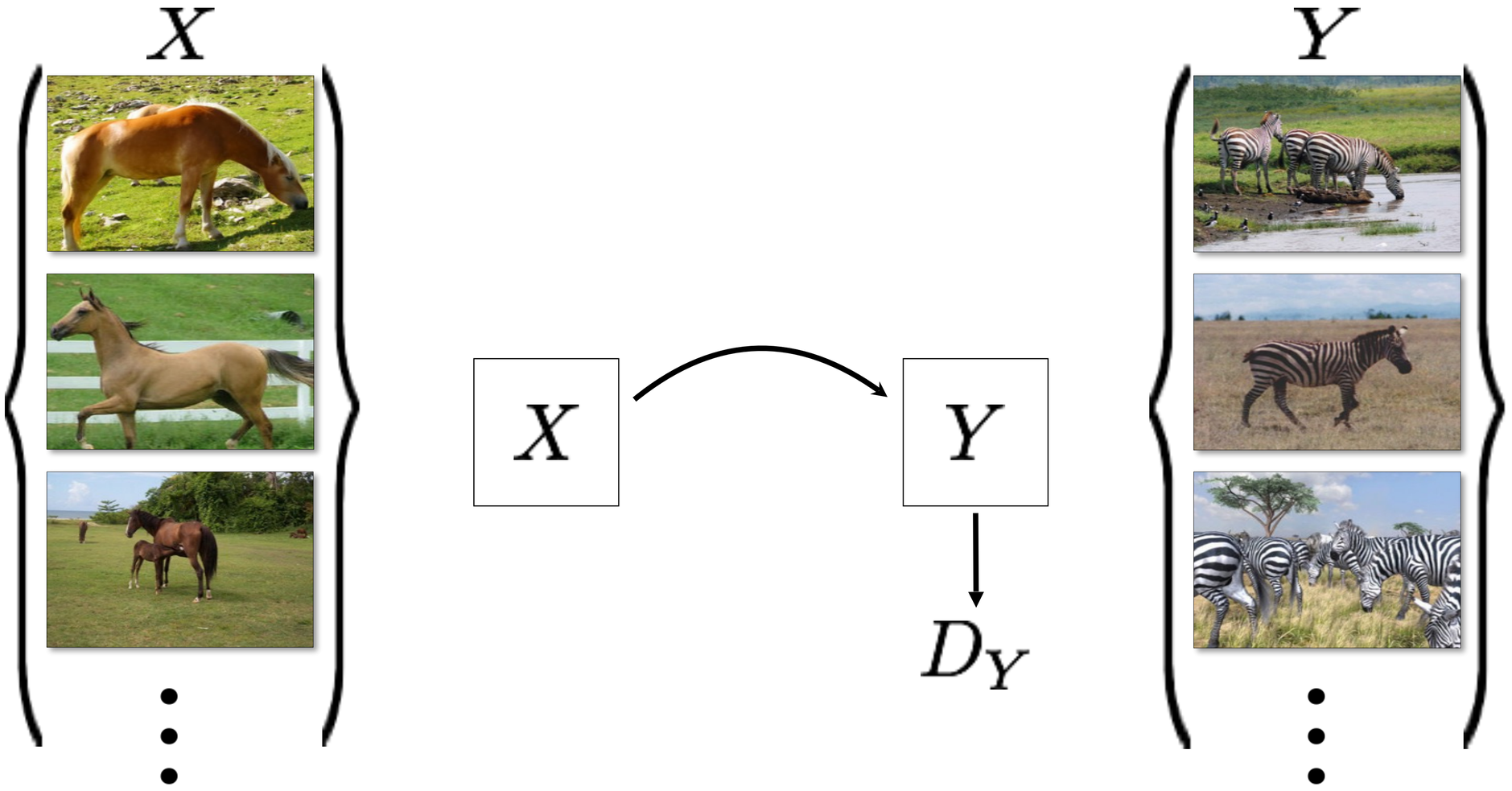


Real!



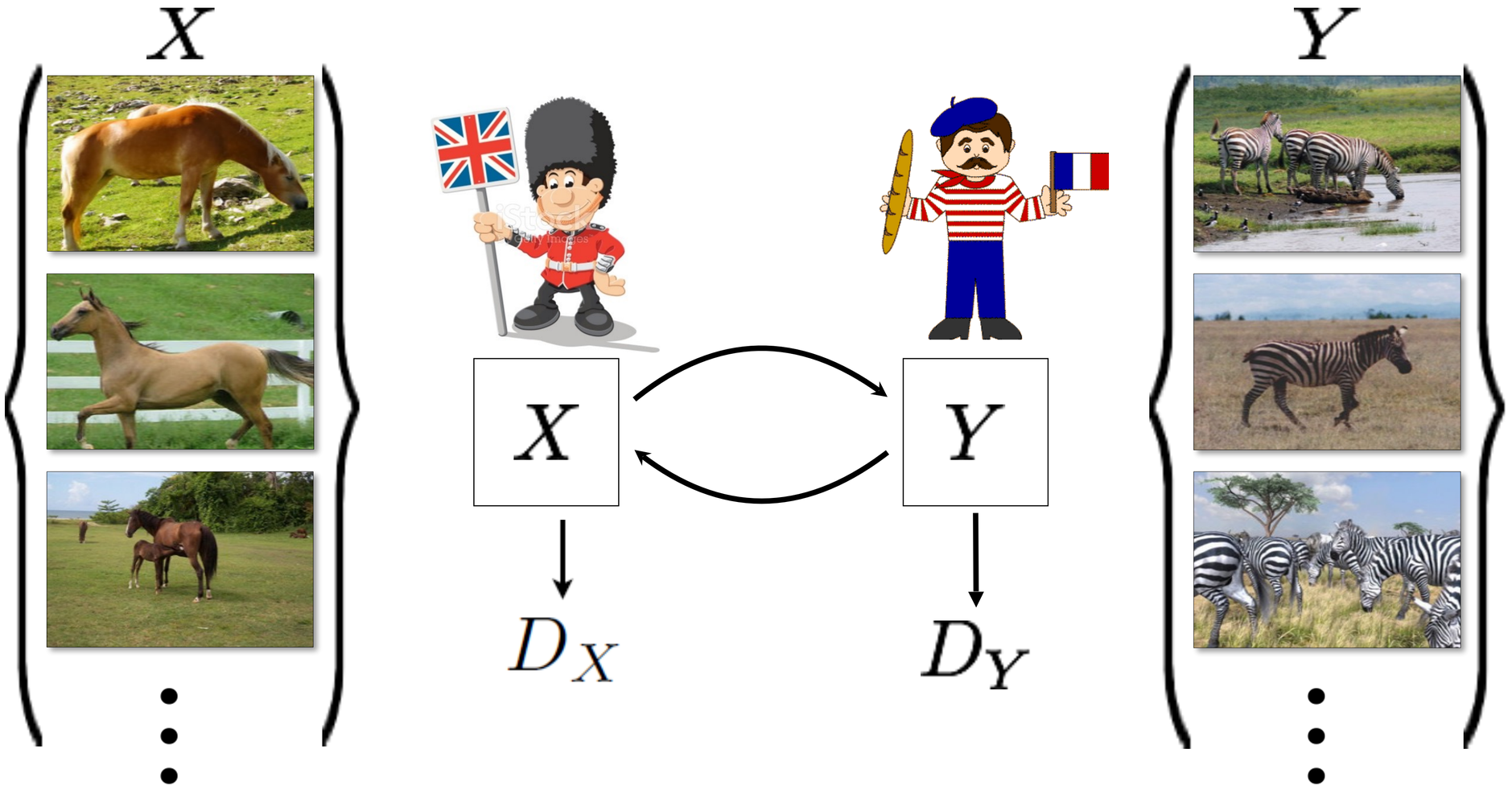
Nothing to force output to correspond to input

Cycle-Consistent Adversarial Networks

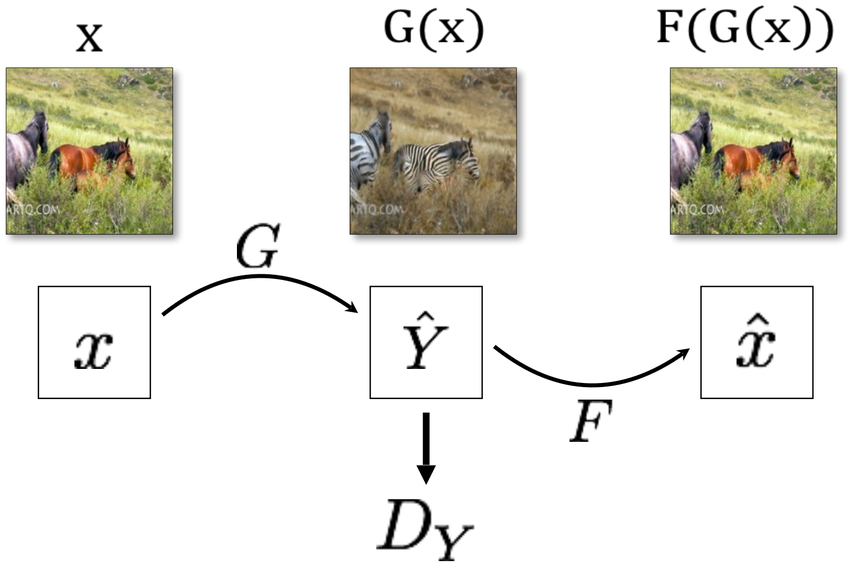


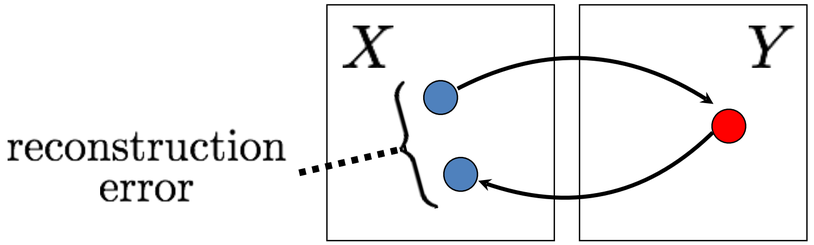
[Zhu et al. 2017], [Yi et al. 2017], [Kim et al. 2017]

Cycle-Consistent Adversarial Networks



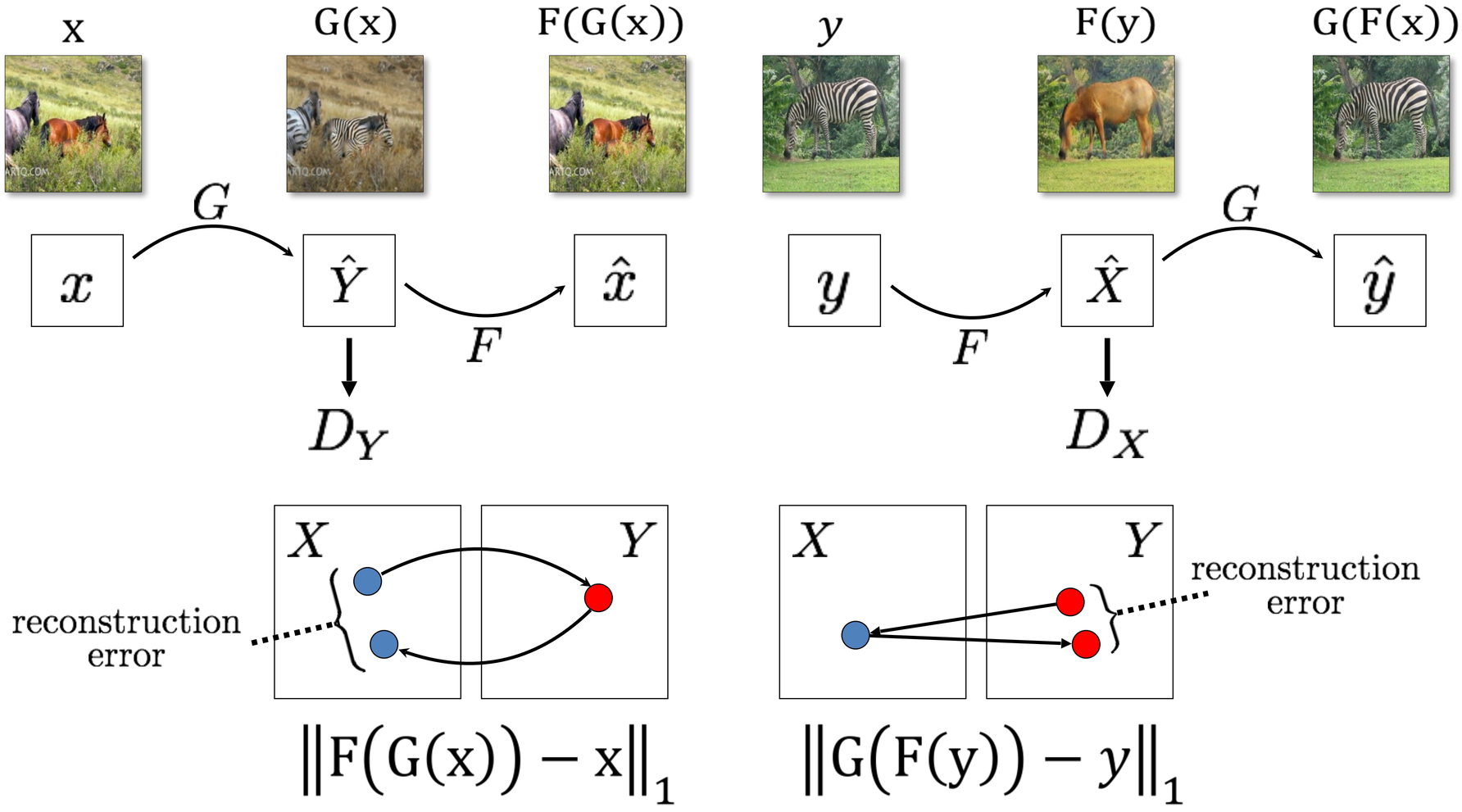
Cycle Consistency Loss

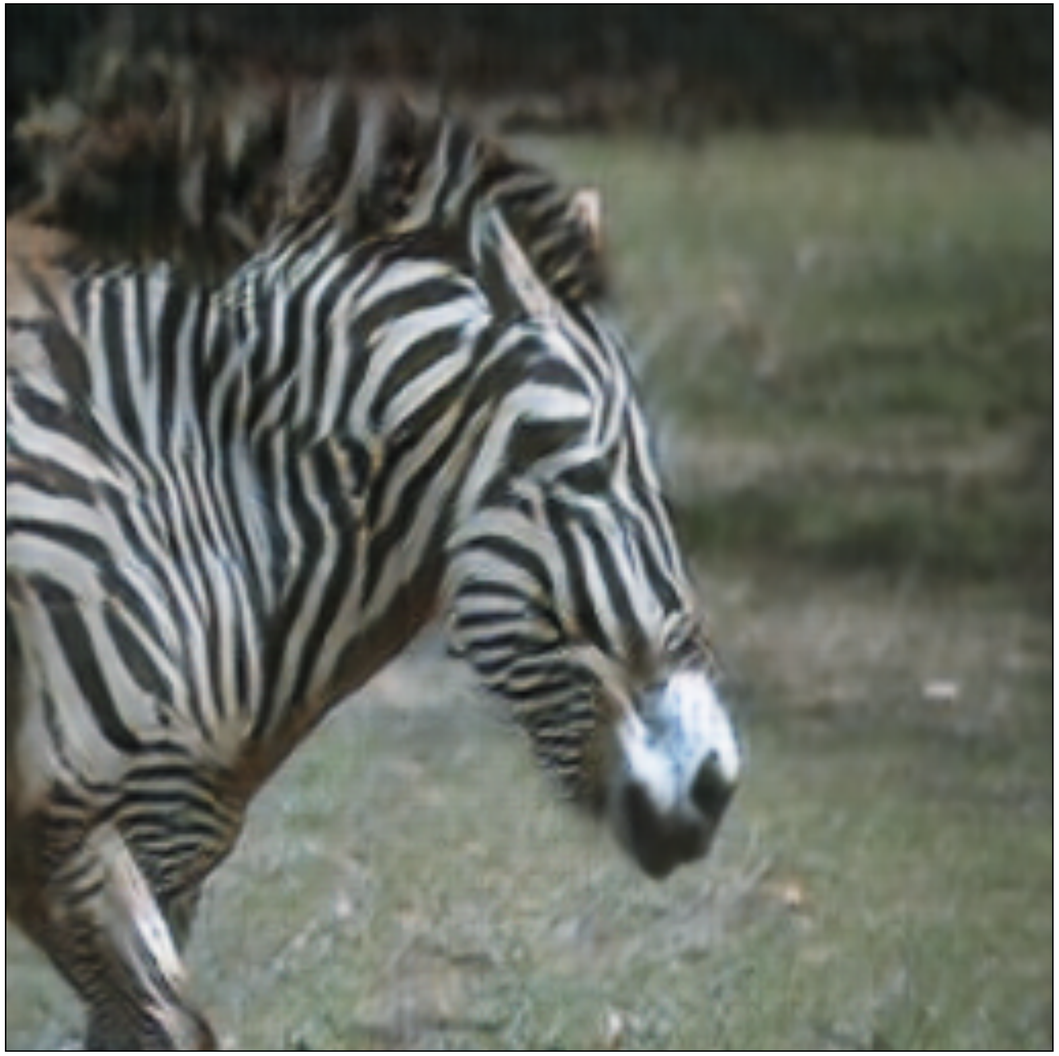




$$\|F(G(x)) - x\|_1$$

Cycle Consistency Loss





Collection Style Transfer

Photograph
@ Alexei Efros

Monet

Van Gogh

Cezanne

Ukiyo-e

Input
Monet
Van Gogh
Cezanne
Ukiyo-e

Monet's paintings → photos

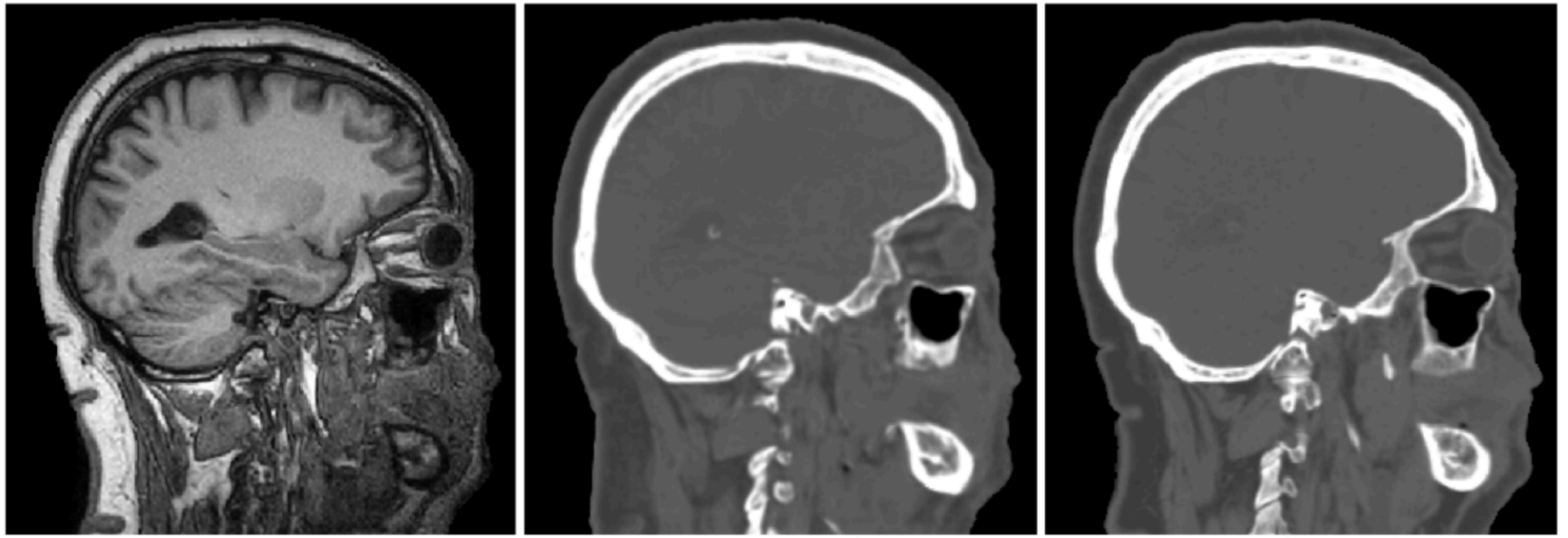
Monet’s paintings → photos

Failure case

Failure case

Medical domain adaptation

MR → CT [Wolterink et al] arxiv: 1708.01155



Input MR | Generated CT | Ground truth CT

- MRI reconstruction [Quan et al.] arxiv:1709.00753
- Cardiac MR images from CT [Chartsias et al. 2017]

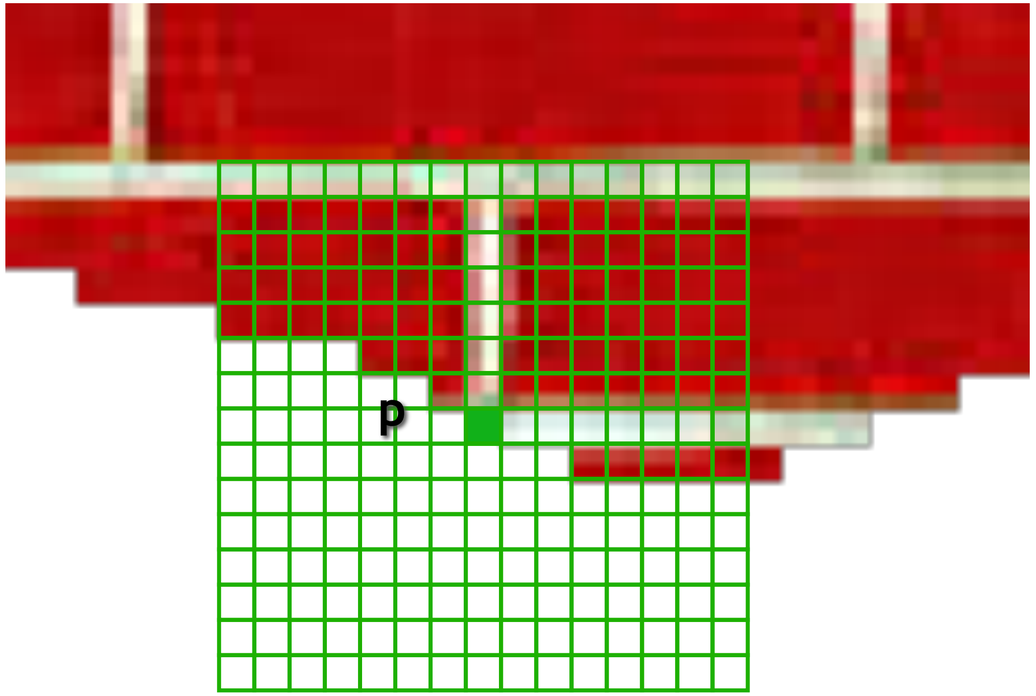
Autoregressive generative models

(putting together a bunch of ideas we have recently seen)

1. Texture synthesis (lecture 10)
2. Sequence modeling (lecture 16)
3. Self-supervised learning (lecture 17)

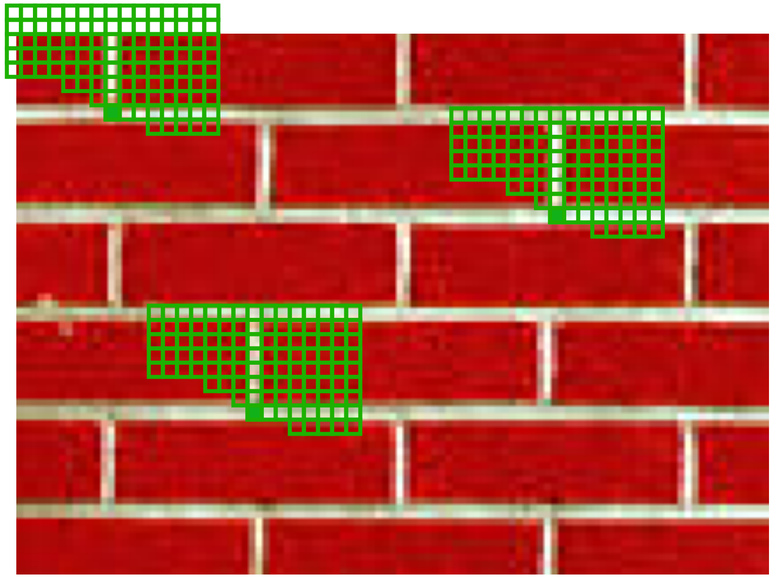
self-supervised image synthesis by sequential pixel prediction

Texture synthesis by non-parametric sampling



Synthesizing a pixel

non-parametric sampling

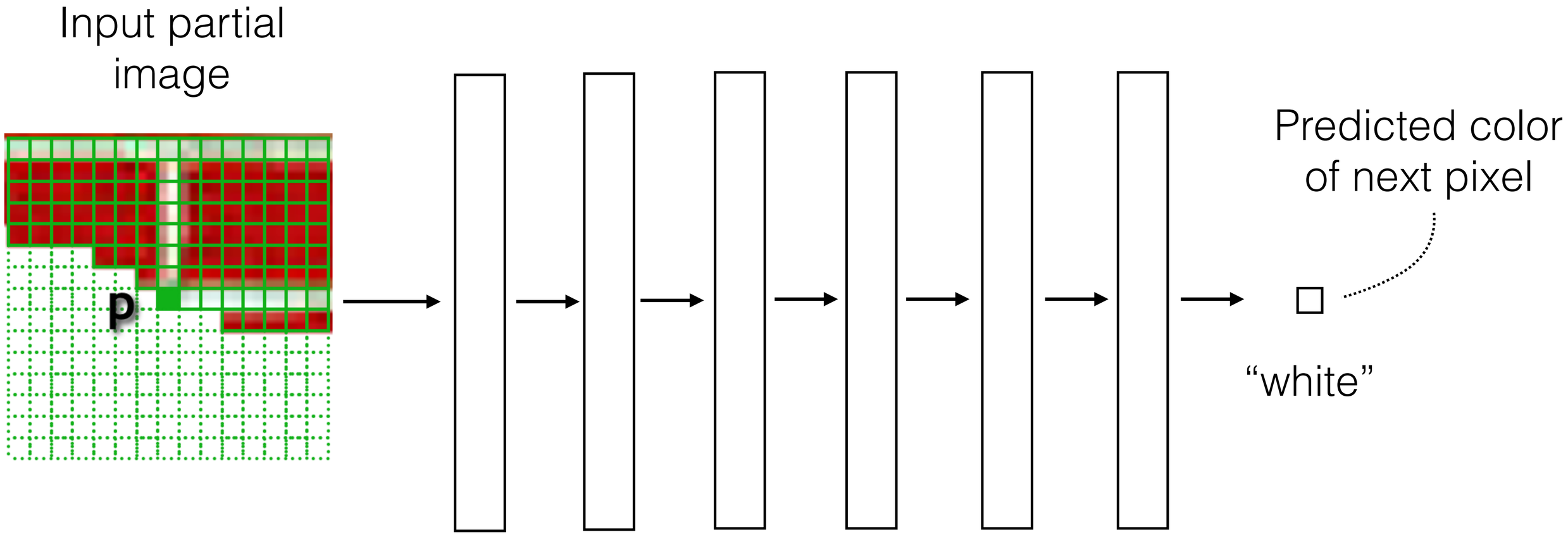


Input image

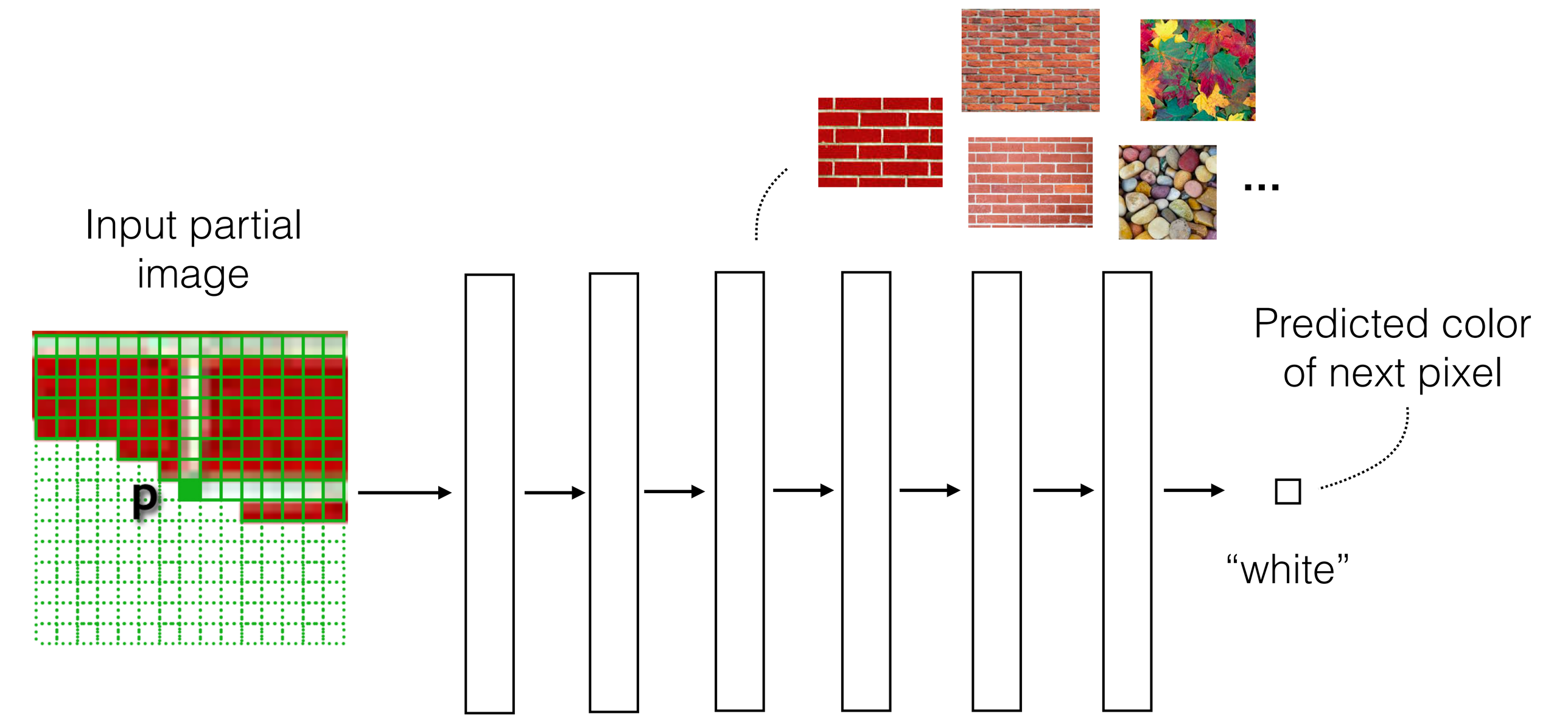
Models $P(p|N(p))$

[Efros & Leung 1999]

Texture synthesis with a deep net



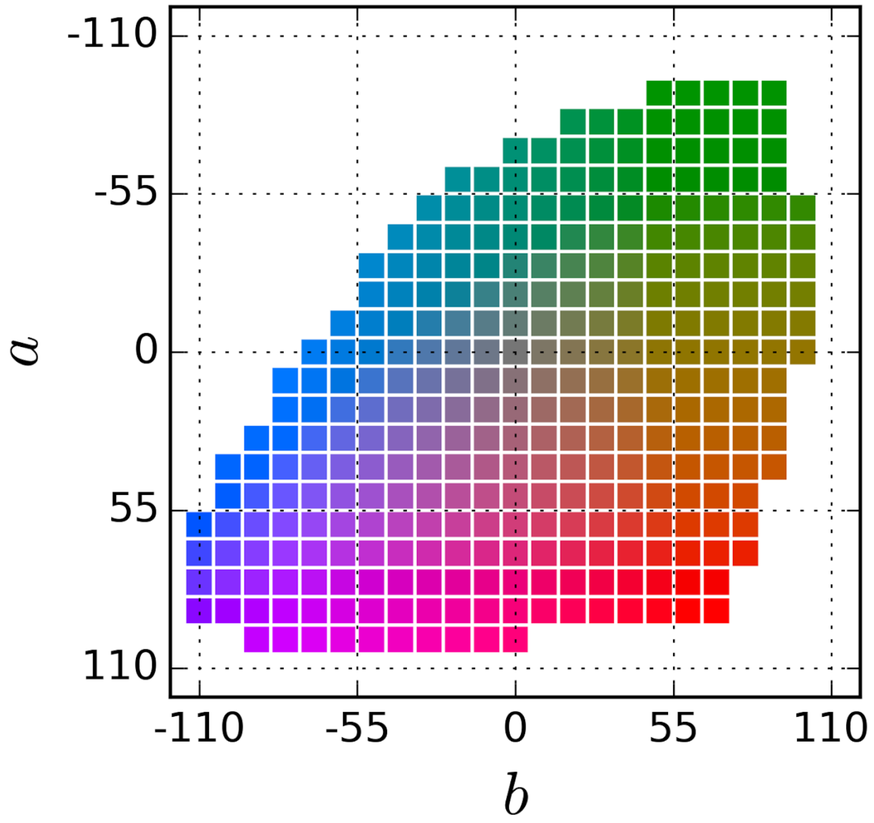
[PixelRNN, PixelCNN, van der Oord et al. 2016]



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

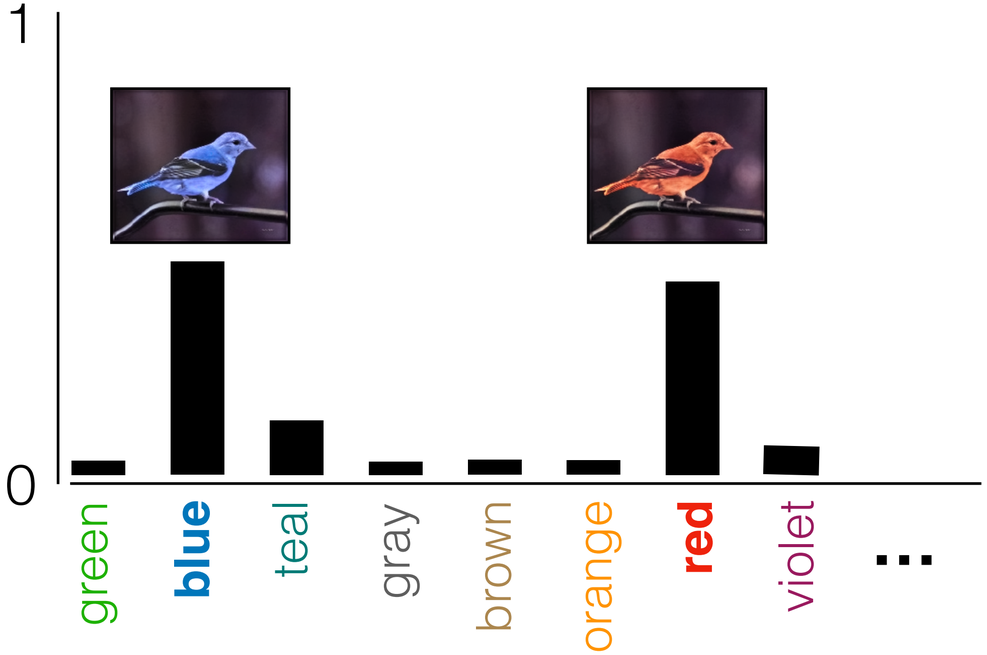
Recall from lecture 12: we can represent colors as discrete classes

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



f

Prediction for a single pixel i,j



$$\mathcal{L}(\mathbf{y}, f_\theta(\mathbf{x})) = H(\mathbf{y}, \texttt{softmax}(f_\theta(\mathbf{x})))$$

And we can interpret the learner as modeling P(next pixel | previous pixels):

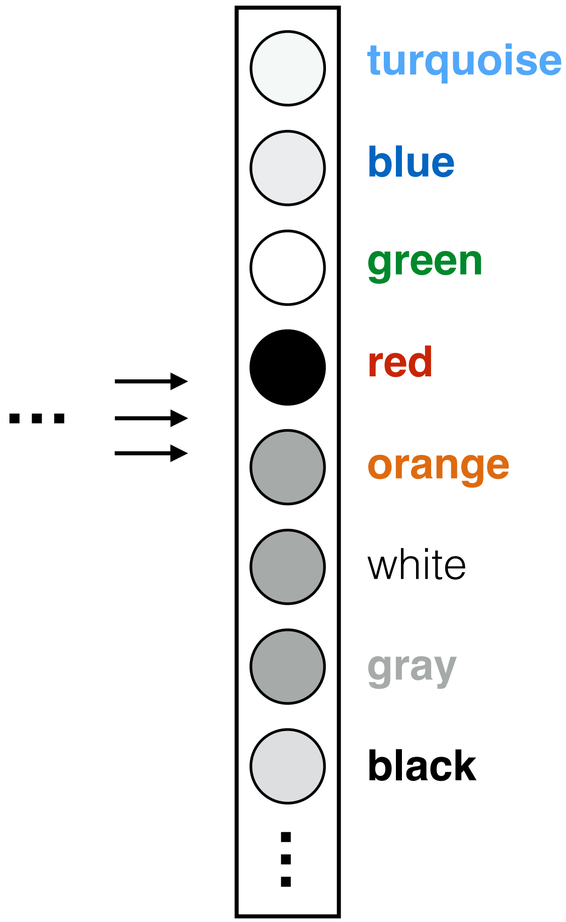
Softmax regression (a.k.a. multinomial logistic regression)

$$\hat{\mathbf{y}} \equiv [P_\theta(Y = 1|X = \mathbf{x}), \ldots, P_\theta(Y = K|X = \mathbf{x})]$$ ← 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 $\hat{\mathbf{y}}$

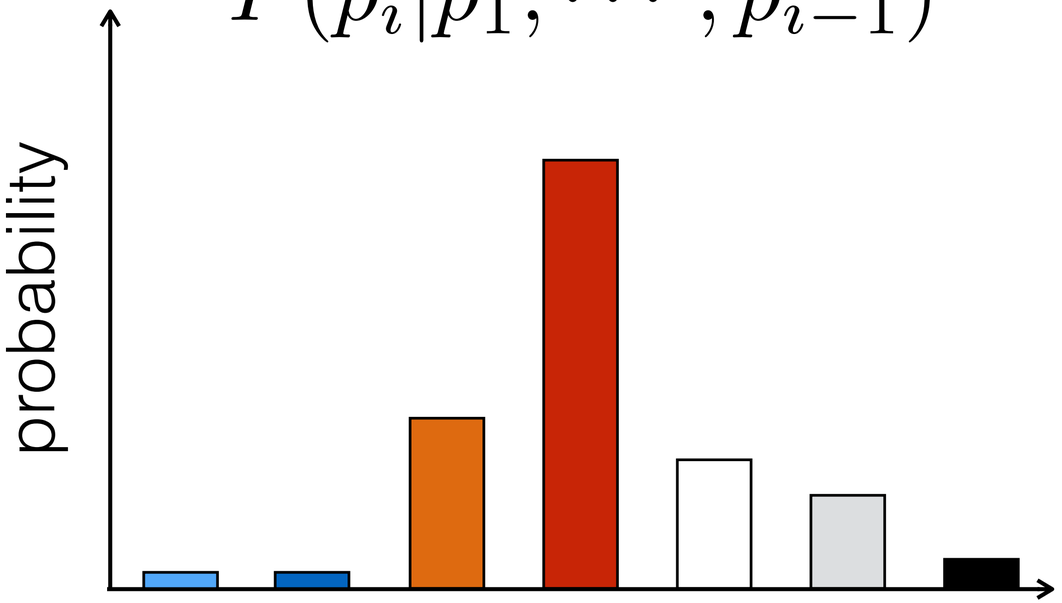
$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} H(\mathbf{y}_i, \hat{\mathbf{y}}_i)$$ ← max likelihood learner!

Network output

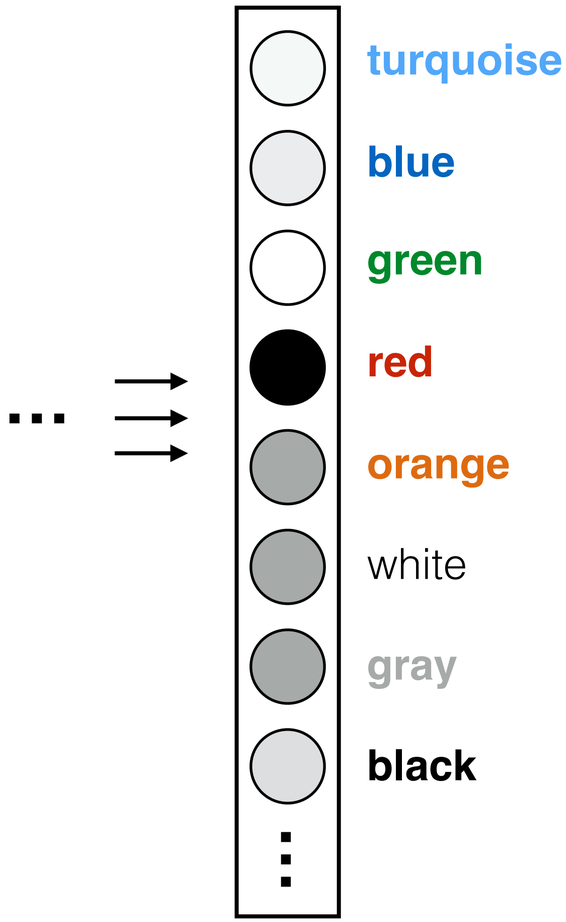


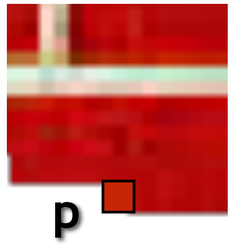
P(next pixel | previous pixels)

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

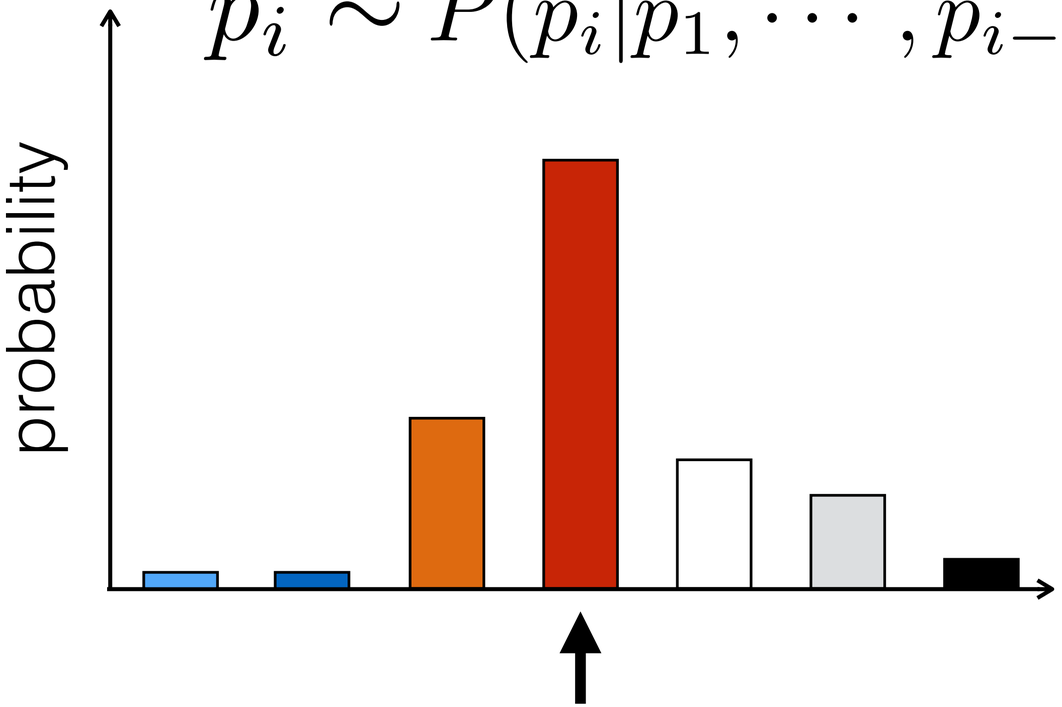


Network output

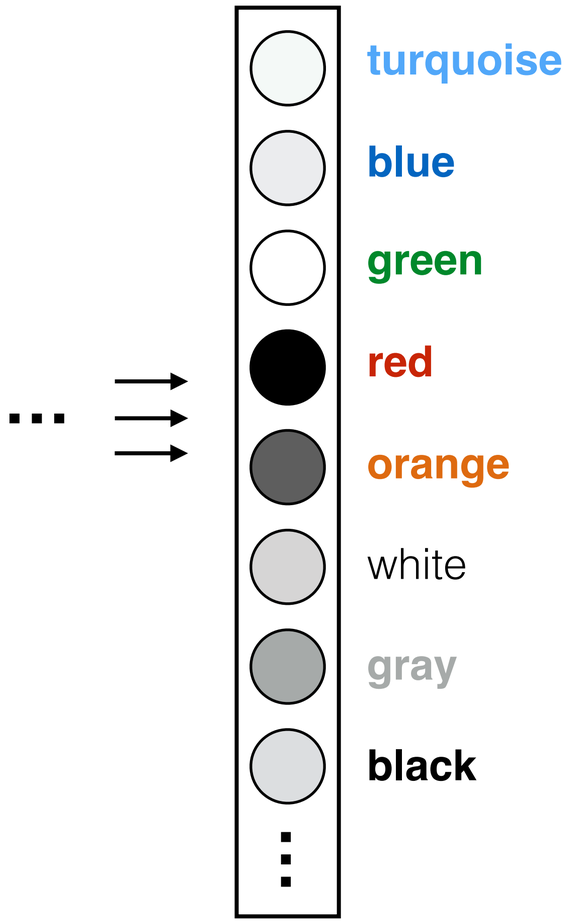


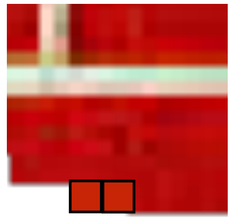


$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$

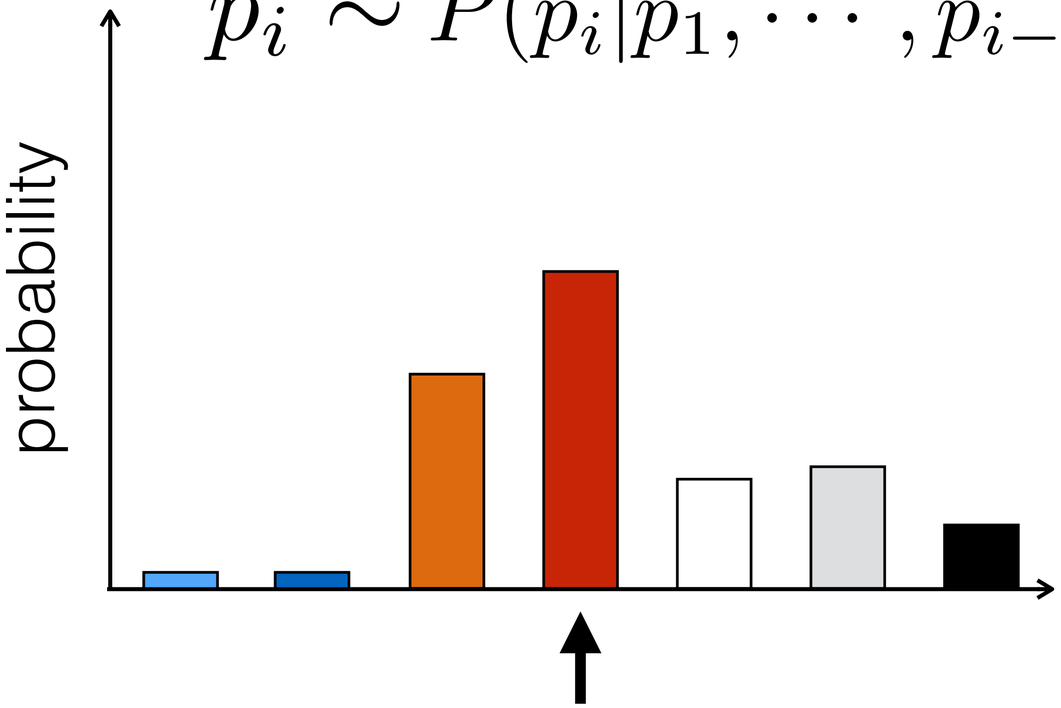


Network output

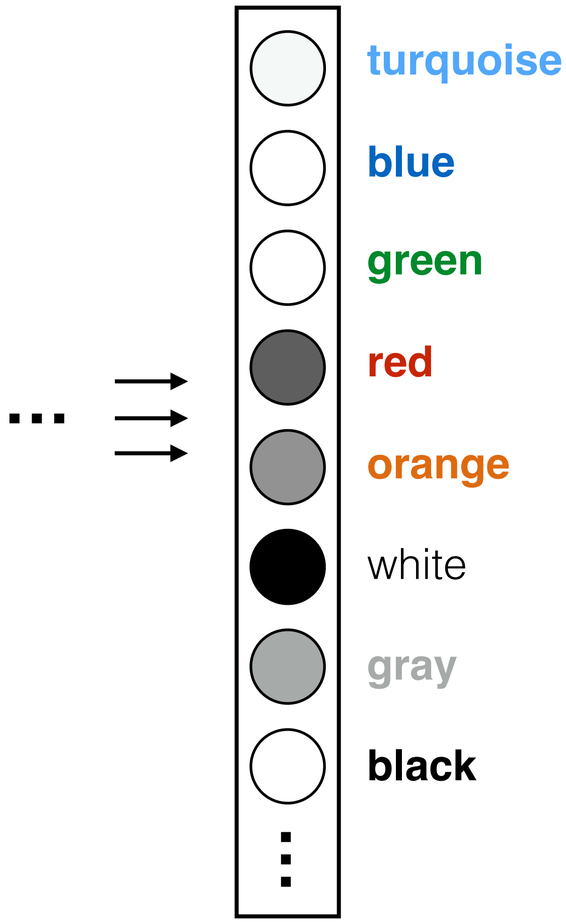


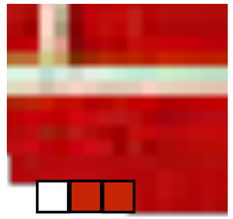


$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$

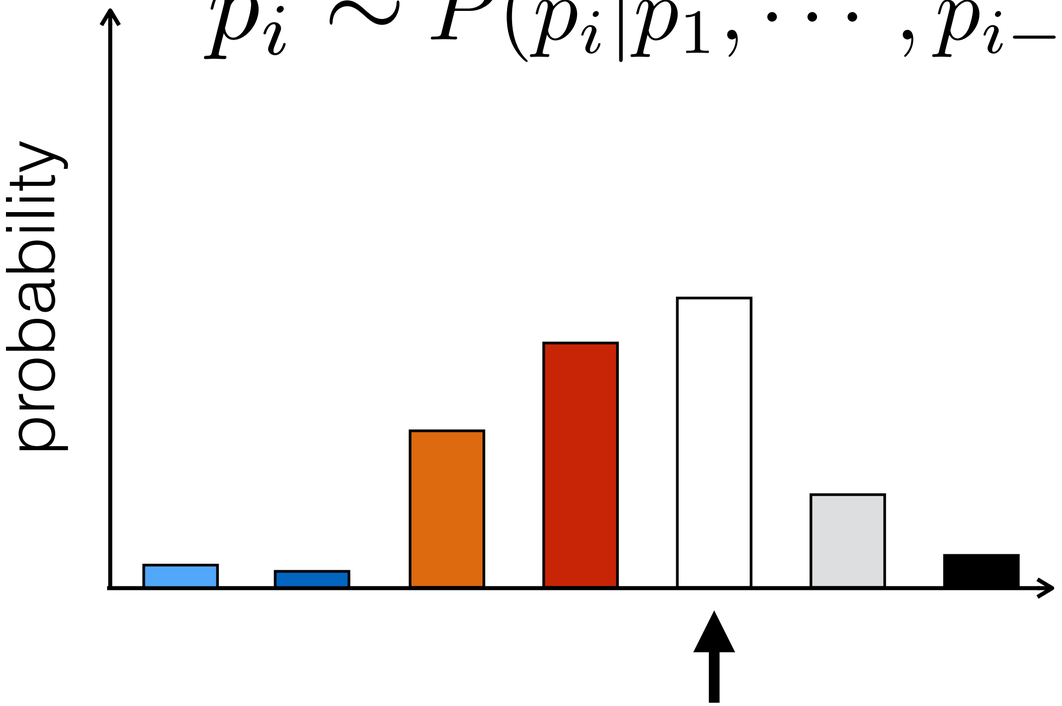


Network output

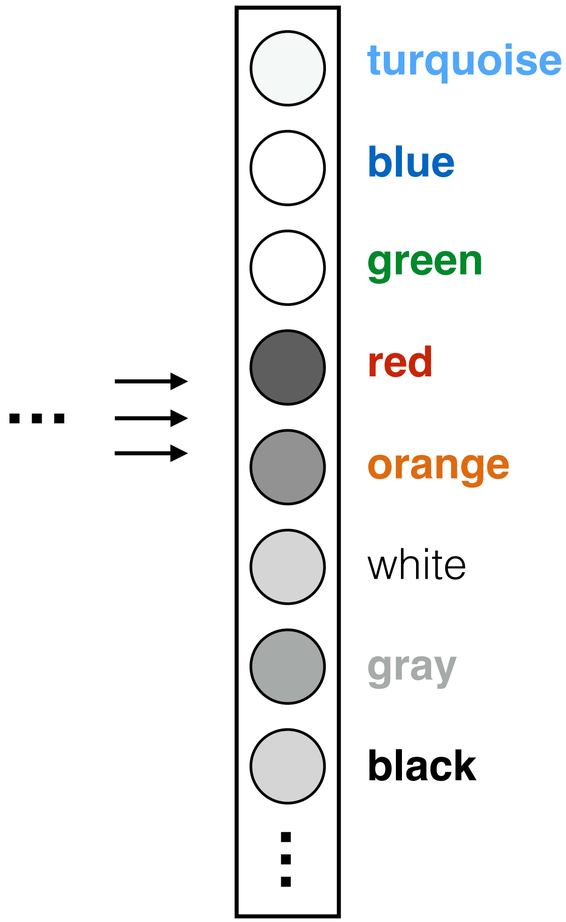


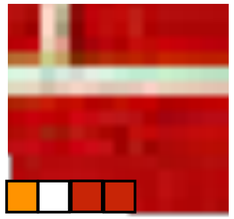


$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$

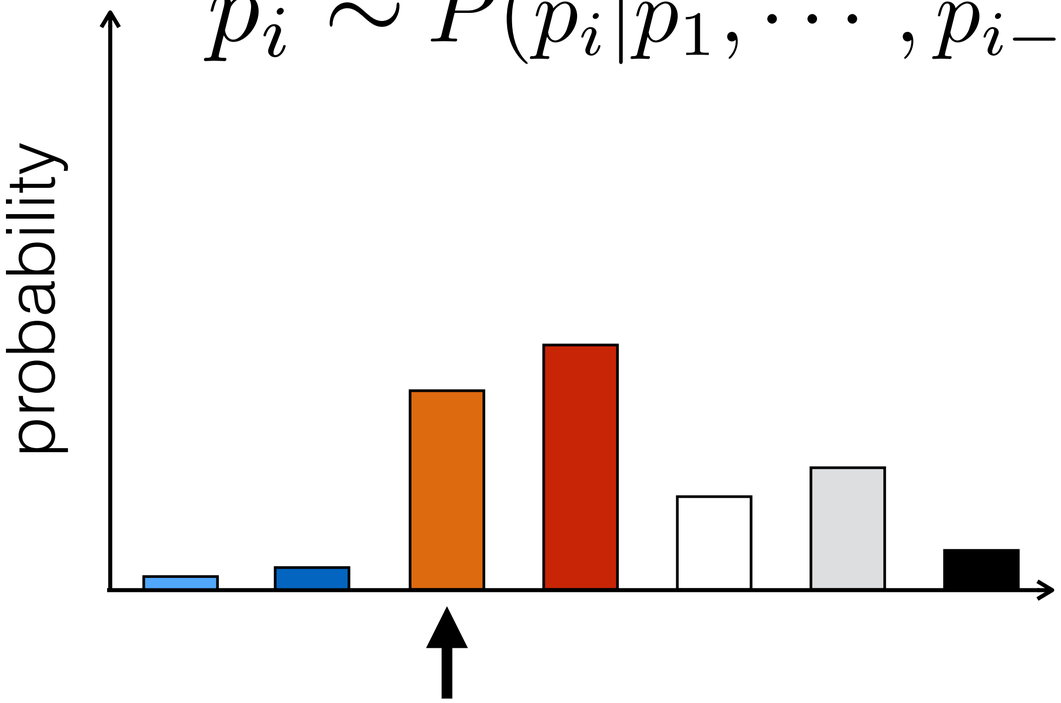


Network output





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

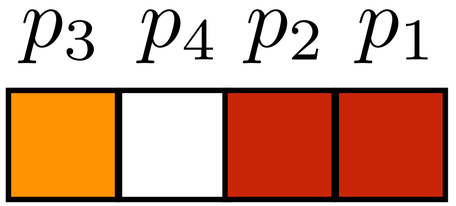


$$p_1 \sim P(p_1)$$

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

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

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



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

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

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

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 (pmf).

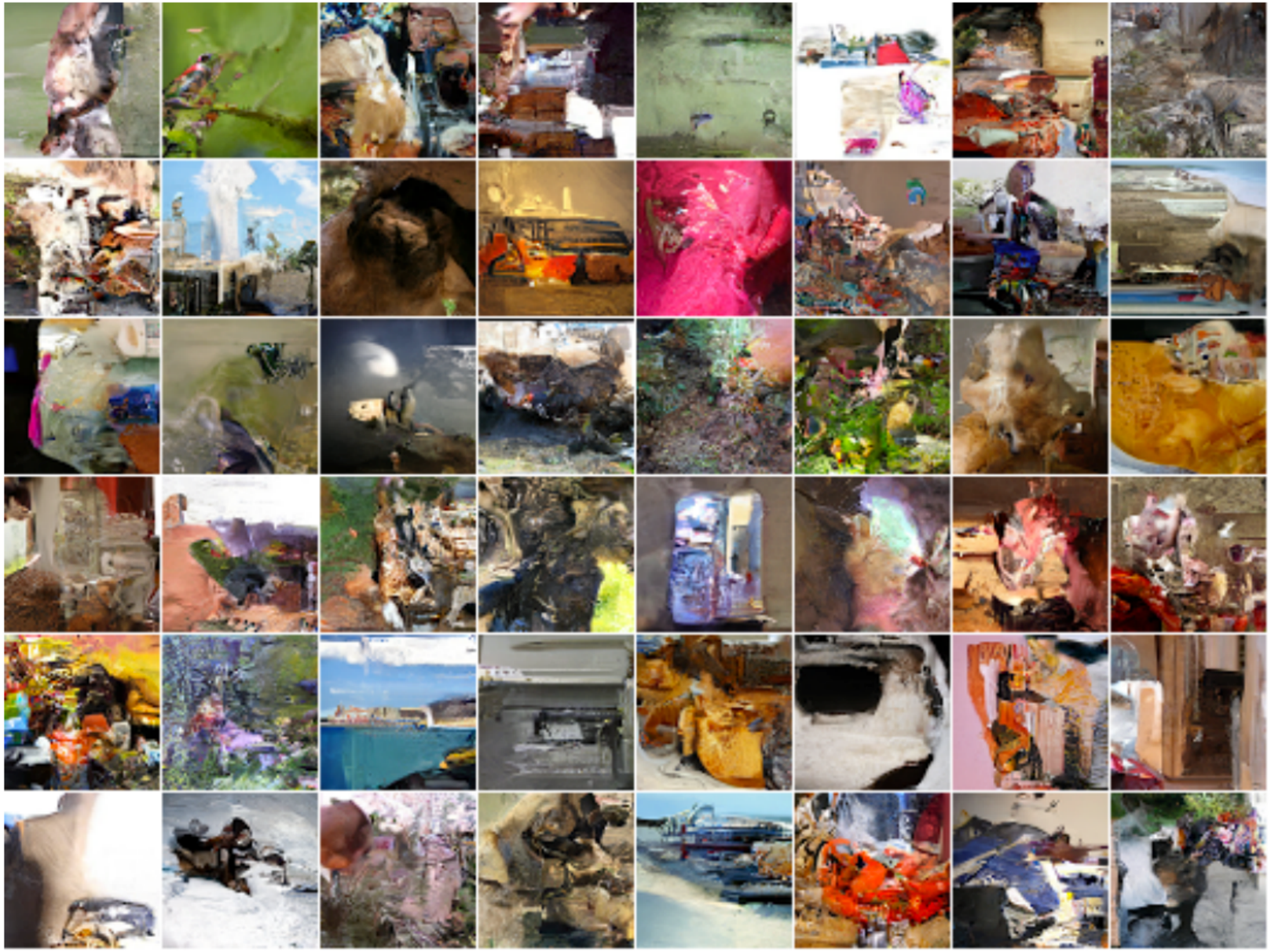
Multiplying all conditionals evaluates the probability of a full joint configuration of pixels.

Generative models

$$\mathbf{p} \sim P(\mathbf{p})$$

Models that allow us to sample, i.e. *generate*, images from scratch are called **generative models**.

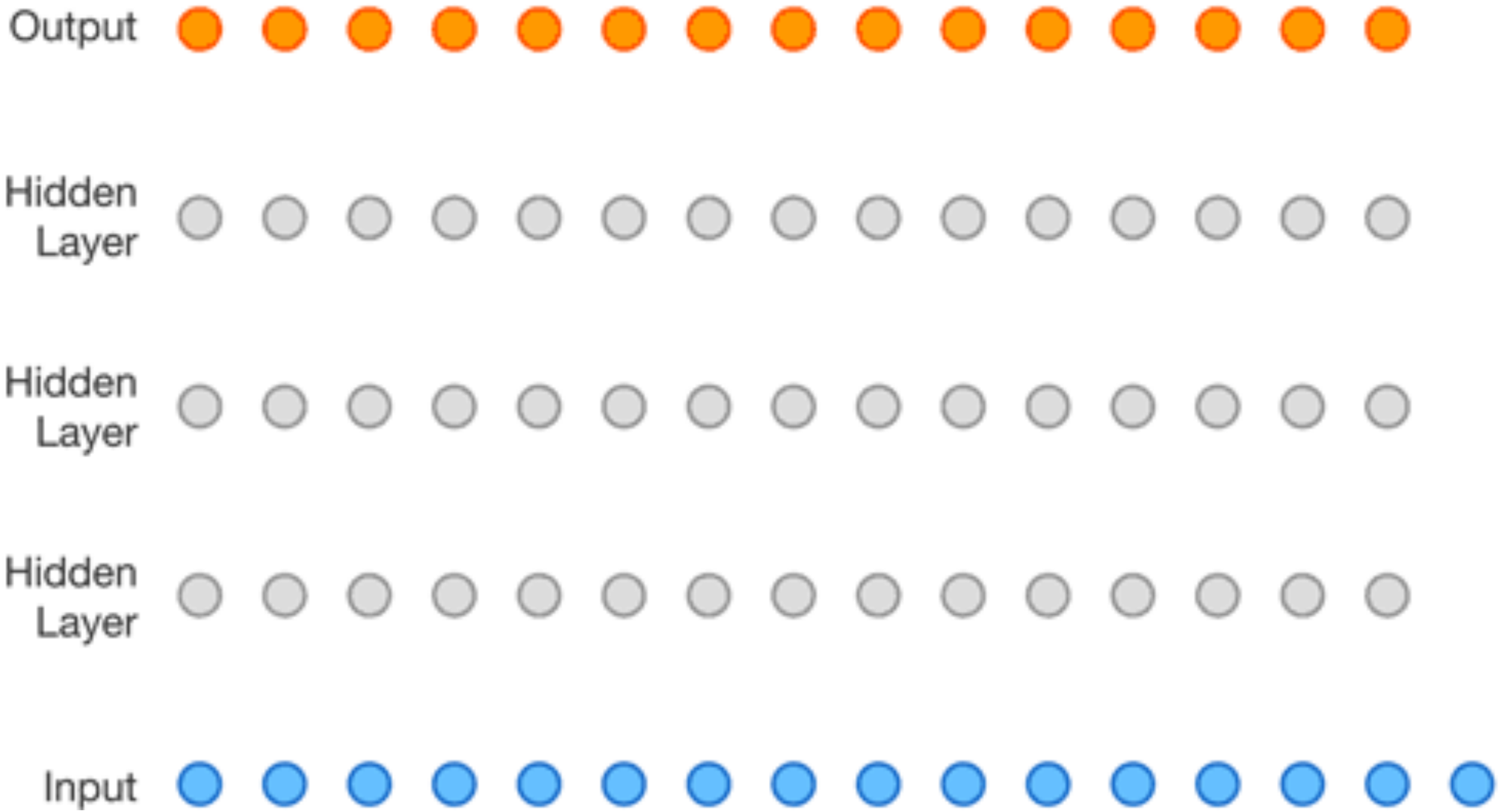
Samples from PixelRNN



[PixelRNN, van der Oord et al. 2016]

Image completions (conditional samples) from PixelRNN

[PixelRNN, van der Oord et al. 2016]



[**Wavenet**, https://deepmind.com/blog/wavenet-generative-model-raw-audio/]