What has driven GAN progress?

- Loss functions:

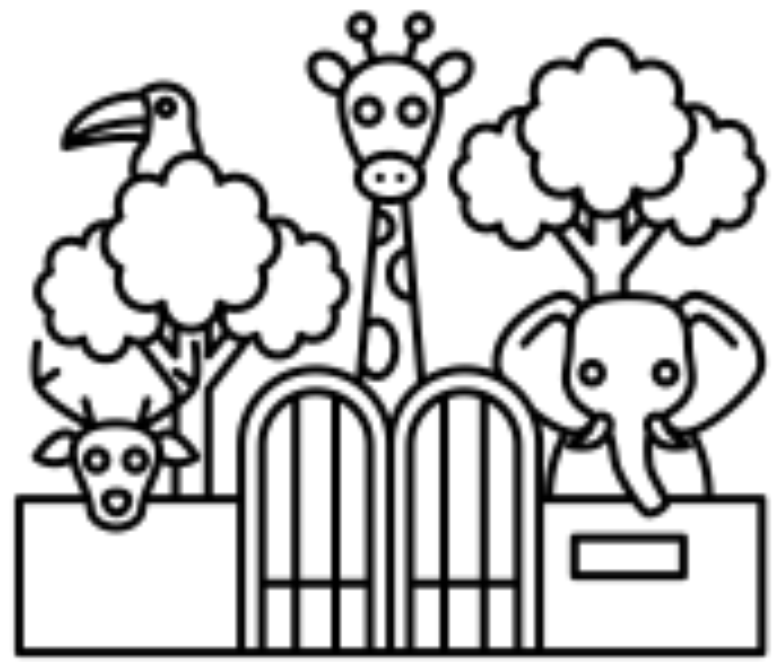
  cross-entropy, least square, Wasserstein loss, gradient penalty, Hinge loss, …
- Network architectures (G/D)

  Conv layers, Transposed Conv layers, modulation layers (AdaIN, spectral norm)
  mapping networks, …
- Training methods

  1. coarse-to-fine progressive training
  2. using pre-trained classifiers (multiple classifiers, random projection)
- Data

  data alignment, data filtering, differentiable augmentation
- GPUs

  bigger GPUs = bigger batch size (stable training) + higher resolution



2

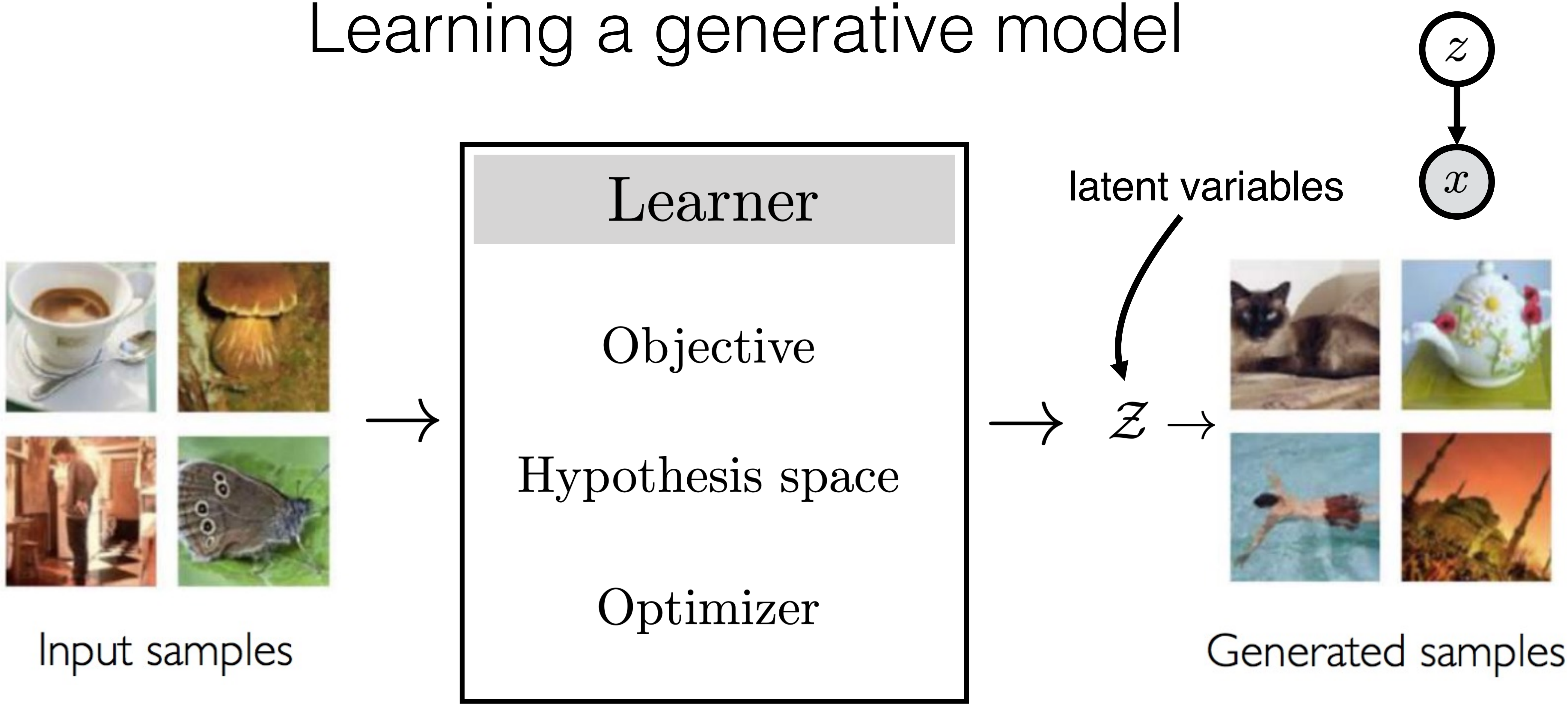
Generative Model Zoo (part I)

Jun-Yan Zhu

16-726 Learning-based Image Synthesis, Spring 2023

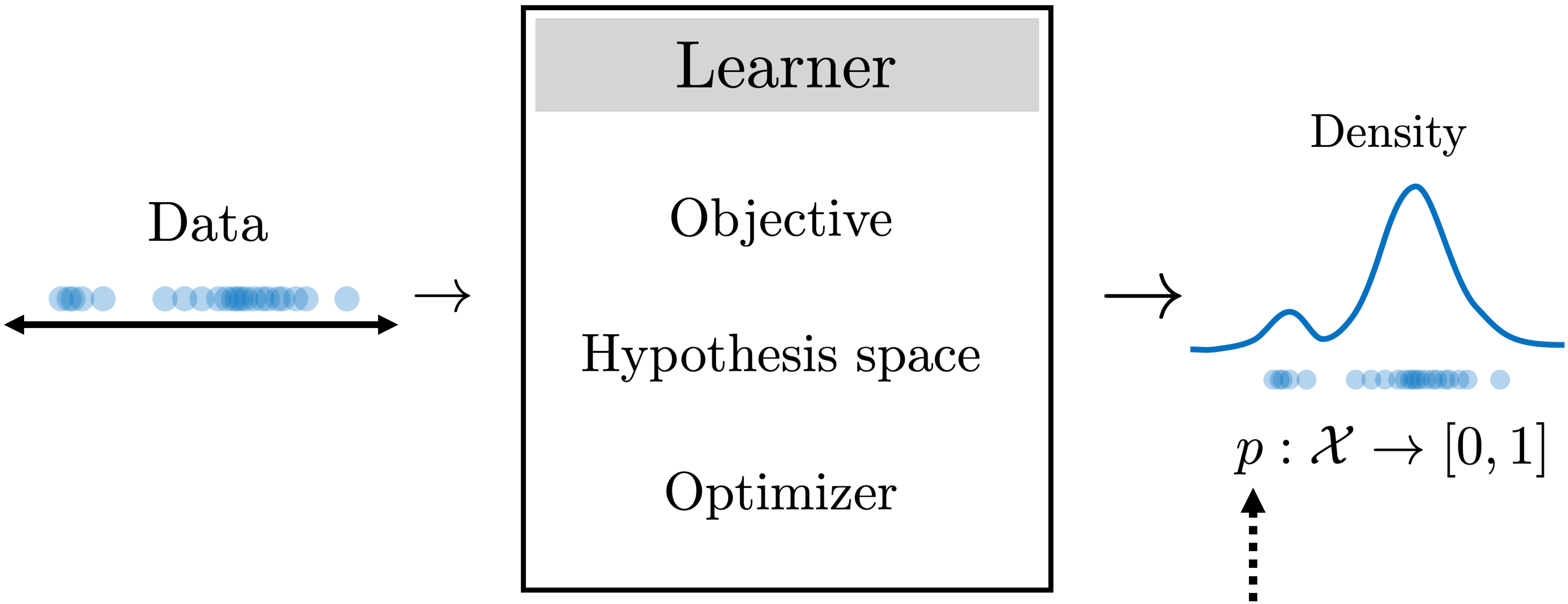
many slides from Phillip Isola, Richard Zhang, Alyosha Efros

Learning a generative model



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

Learning a density model



Integral of probability density function needs to be 1 ⟶ Normalized distribution
(some models output unnormalized ***energy functions***)

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

Useful for abnormality/outlier detection (detect unlikely events)

Case study #1: Fitting a Gaussian to data

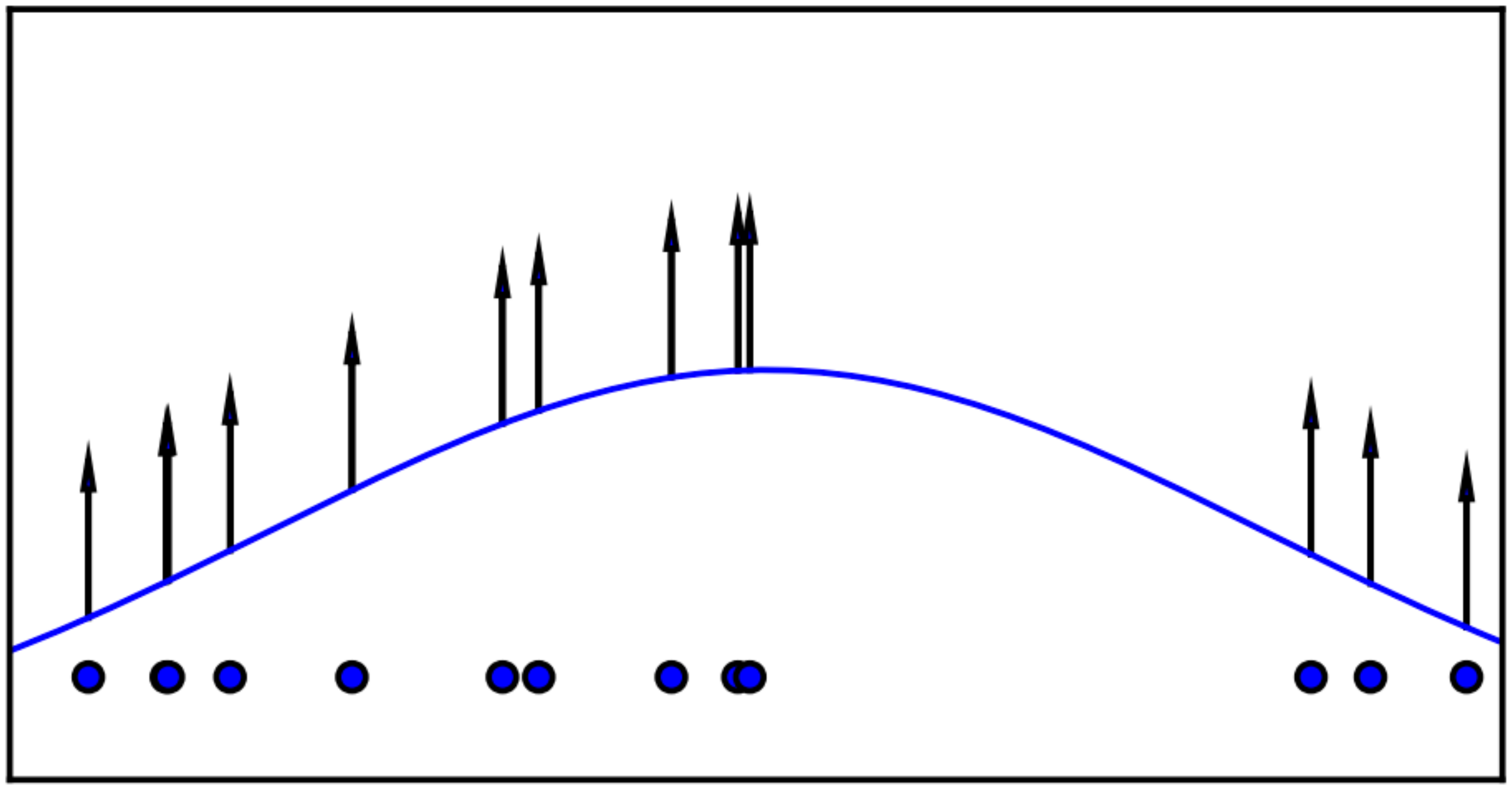


fig from [Goodfellow, 2016]

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:

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

Maximum log likelihood=minimize KLD

KLD (Kullback–Leibler divergence): $\mathcal{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx$

JSD (Jensen–Shannon divergence): $\mathcal{JSD}(p \parallel q) = \frac{1}{2}\mathcal{KL}(p \parallel \frac{p+q}{2}) + \frac{1}{2}\mathcal{KL}(q \parallel \frac{p+q}{2})$

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[\log p_\theta(x)] = \int_x p_{\text{data}}(x) \log p_\theta(x) dx$$

$$\mathcal{KL}(p_{\text{data}}(x)||p_\theta(x)) = \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_\theta(x)} dx$$

$$= \int_x p_{\text{data}}(x) \log p_{\text{data}}(x) dx - \int_x p_{\text{data}}(x) \log p_\theta(x) dx$$

Constant (independent of θ)

Maximize log likelihood=minimize KLD

Case study #2: Generative Adversarial Network

Data $\rightarrow$

Learner

Objective

$$\min_G \max_D \mathbb{E}_z[\log(1-D(G(z))]+\mathbb{E}_x[\log D(x)]$$

Hypothesis space

Deep nets G and D

Optimizer

Alternating SGD on G and D

$\rightarrow$

Critic

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

Sampler

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

$p_g = p_{data}$ is the unique global minimizer of the GAN objective.

Proof

Optimal discriminator given fixed G

$$
\begin{aligned}
C(G) &= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

$$
C(G) = -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right)
$$

$$
C(G) = -\log(4) + 2 \cdot \underbrace{JSD\left(p_{\text{data}} \| p_g\right)}_{\geq 0, \quad 0 \iff p_g = p_{data}} \quad \square
$$

KLD (Kullback–Leibler divergence): $\mathcal{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx$

JSD (Jensen–Shannon divergence): $\mathcal{JSD}(p \parallel q) = \frac{1}{2}\mathcal{KL}(p \parallel \frac{p+q}{2}) + \frac{1}{2}\mathcal{KL}(q \parallel \frac{p+q}{2})$

Case study #3: 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 #3: learning a deep generative model

Data $\rightarrow$

Learner

Objective
max likelihood
/distribution matching

Hypothesis space
Deep net

Optimizer
SGD

$\rightarrow$

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

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**

Case study #3: learning a deep generative model

Data $\rightarrow$

Learner

Objective
max likelihood
/distribution matching

Hypothesis space
Deep net

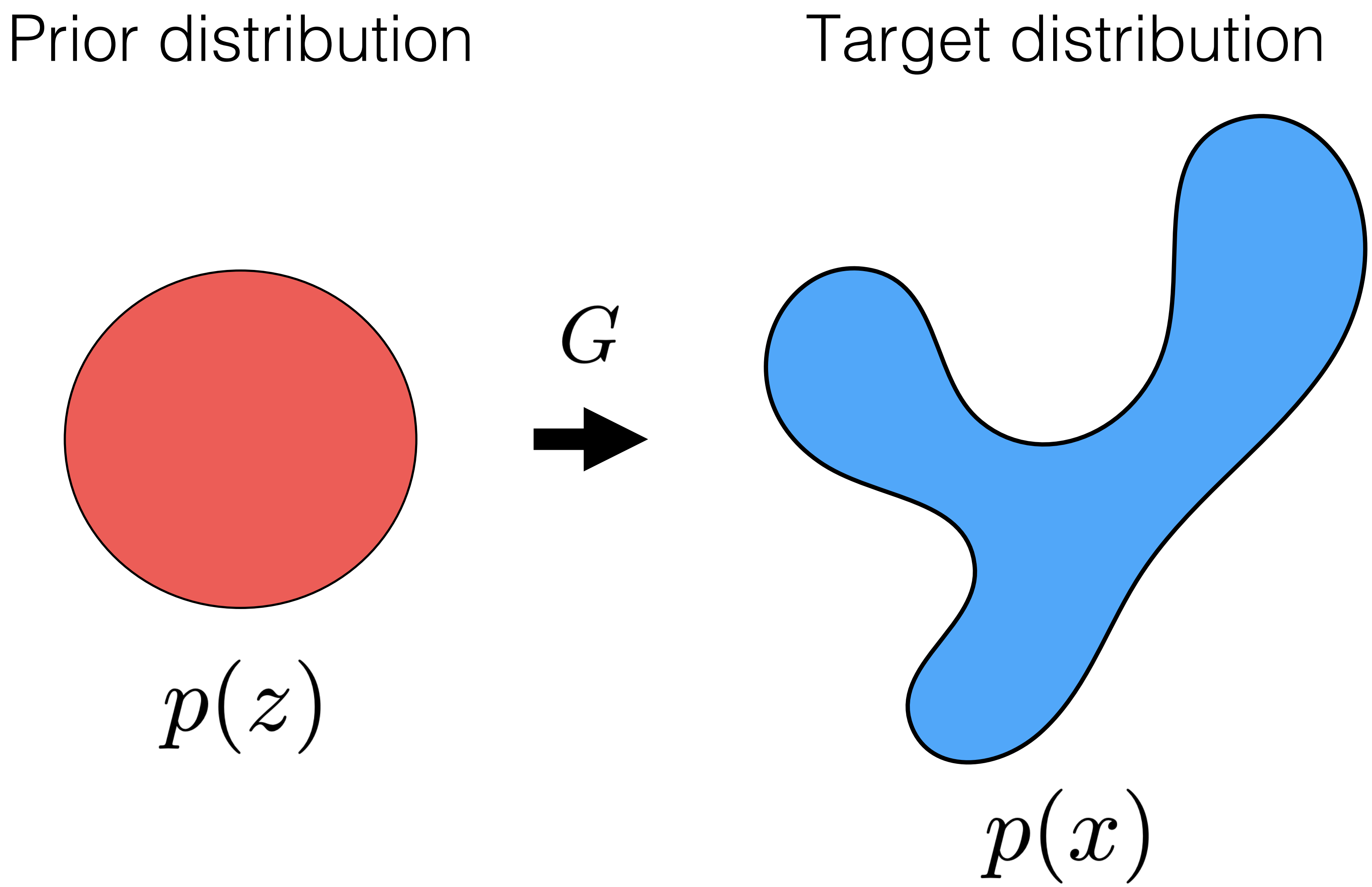
Optimizer
SGD

$\rightarrow$ Density

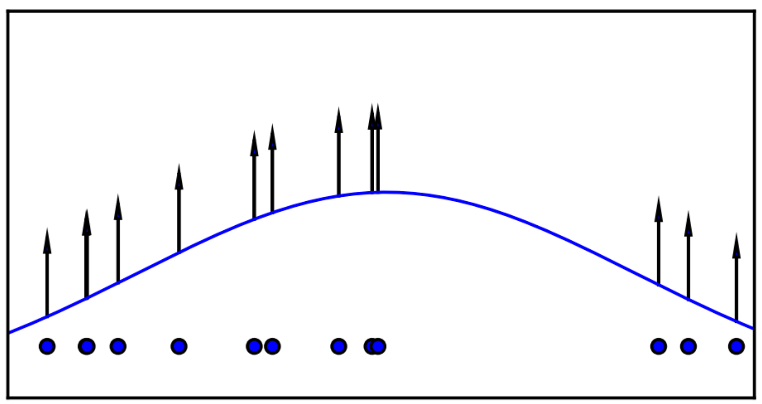
$p : \mathcal{X} \rightarrow [0, 1]$

Variational Autoencoders (VAEs)

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

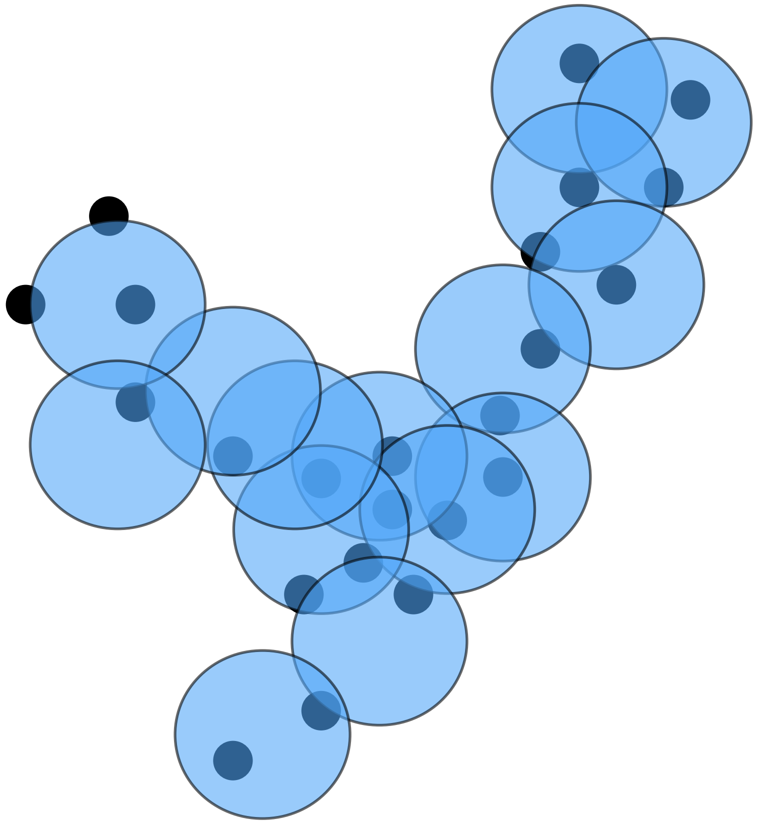


Mixture of Gaussians



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

Target distribution



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

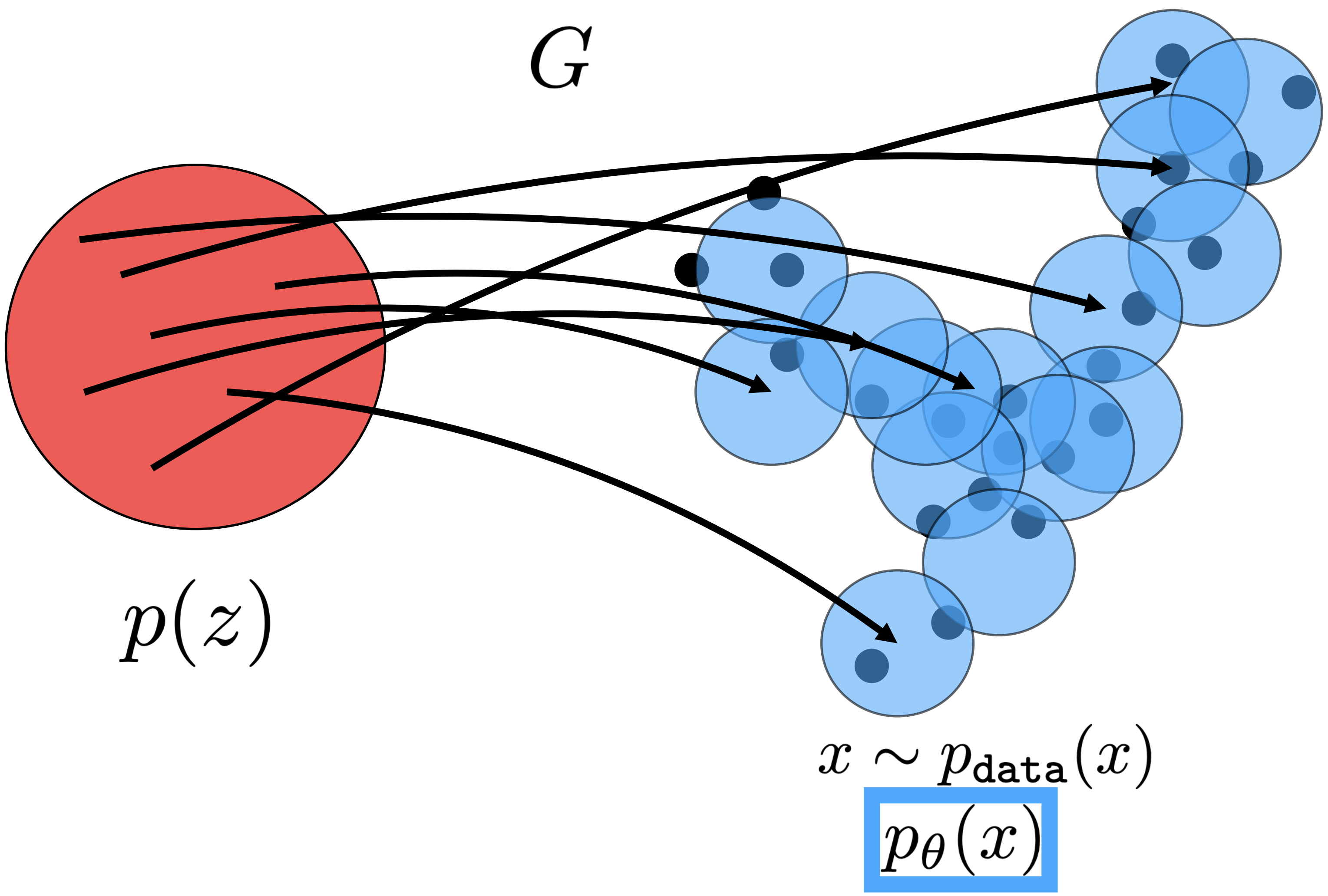
$p_\theta(x)$

Variational Autoencoders (VAEs)

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

Prior distribution

Target distribution



Density model:

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

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

Sampling:

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

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

Variational Autoencoder (VAE)

Data $\rightarrow$

Learner

Objective

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

Hypothesis space

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

$$x = G^{\mu}_{\theta}(z) + G^{\sigma}_{\theta}(z)\epsilon$$

$\rightarrow$

~~Density~~

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

Sampler

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

Variational Autoencoders (VAEs)

Fitting a model to data requires computing $p_\theta(x)$

How to compute $p_\theta(x)$ efficiently?

$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$ ⟵ almost all terms are near zero

Train “inference network” $q_\psi(z|x)$
to give distribution over the z’s that are likely to produce x

Approximate $p_\theta(x)$ with $\mathbb{E}_{q_\psi(z|x)}[p_\theta(x|z)]$

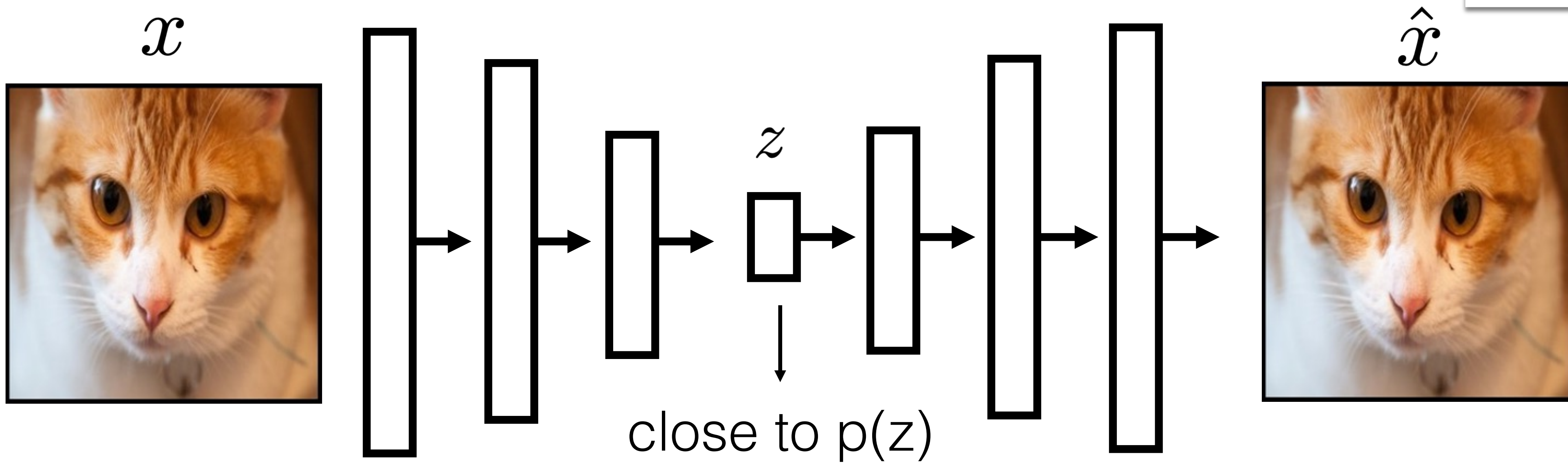
[Kingma and Welling, 2014]

Tutorial on VAEs [Doersch, 2016]

Variational Autoencoders (VAEs)

encoder $q_\psi(z|x)$ $z = E^\mu_\psi(x) + E^\sigma_\psi(x) \cdot \epsilon_z$

generator $p_\theta(x|z)$ $\hat{x} = G^\mu_\theta(z)$



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

Multi-variate Gaussian

$$\geq \max_{\theta,\psi} \mathbb{E}_{x_i\sim p_{\texttt{data}}}[\mathbb{E}_{q_\psi(z|x_i)}[p_\theta(x|z)] - \mathrm{KL}(q_\psi(z|x_i)||p(z))]$$

reconstruction loss

$$||x - \hat{x}||_2$$

KLD loss

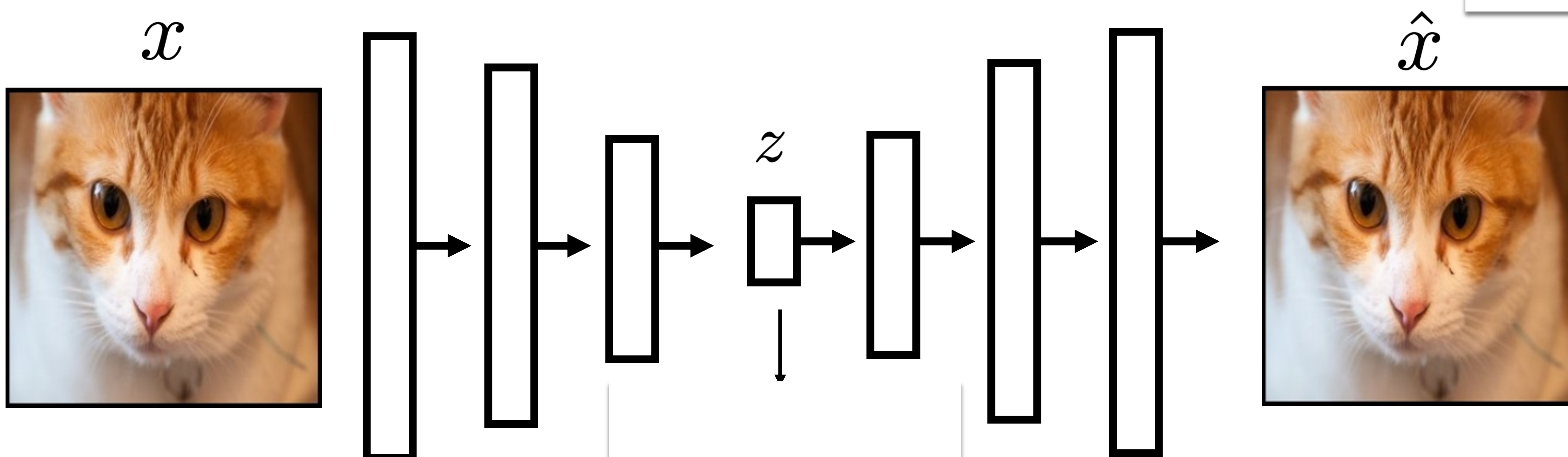
$$\mathrm{KLD}(\mathcal{N}(E^\mu_\psi(x), E^\sigma_\psi(x)) \mid \mathcal{N}(0, I))$$

[Kingma and Welling, 2014]

Autoencoders (AEs)

encoder $q_\psi(z|x)$ $z = E^{\mu}_{\psi}(x)$

generator $p_\theta(x|z)$ $\hat{x} = G^{\mu}_{\theta}(z)$



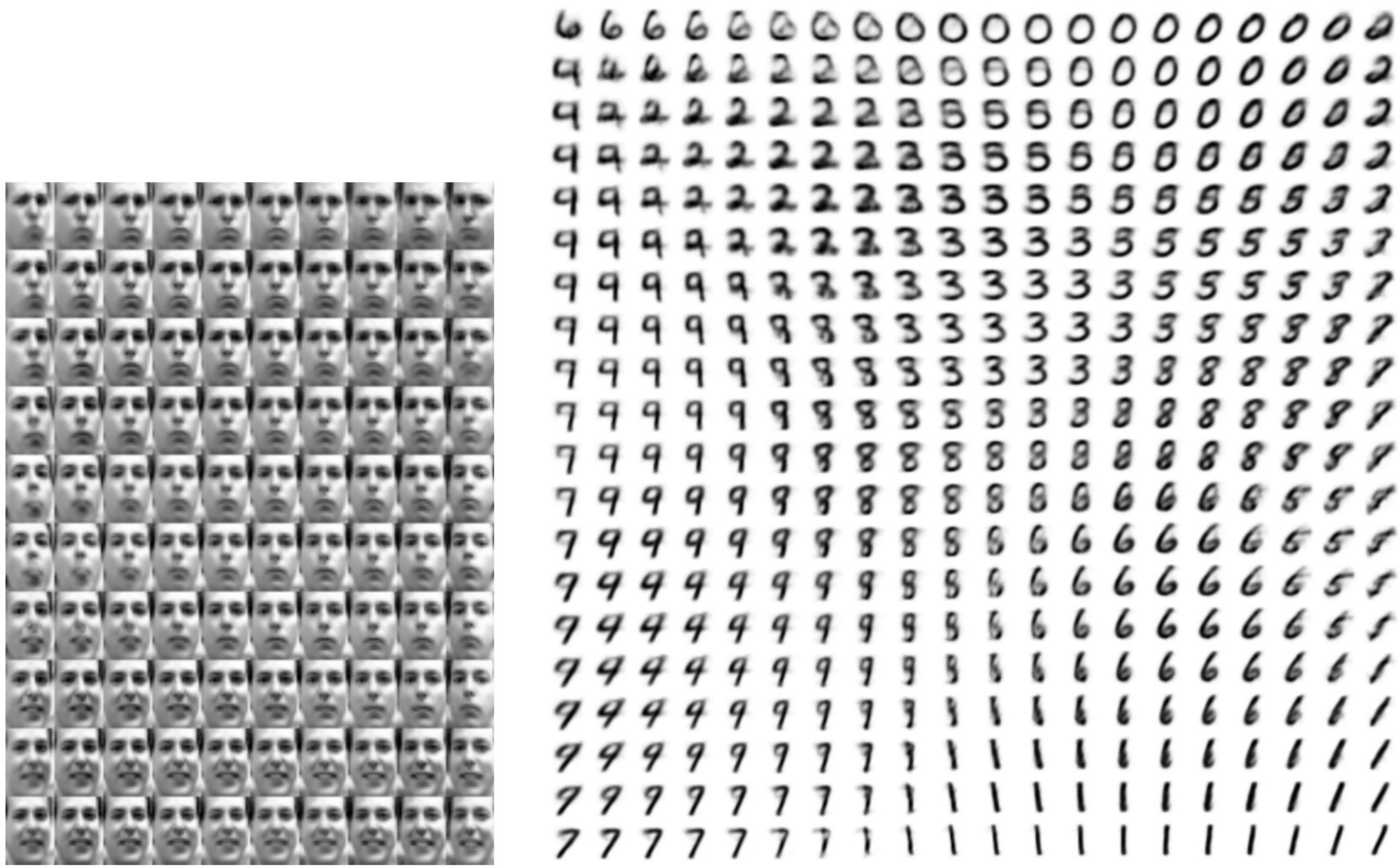
$$\max_{\theta,\psi} \mathbb{E}_{x_i \sim p_{\text{data}}}[\mathbb{E}_{q_\psi(z|x_i)}[p_\theta(x|z)]]$$

reconstruction loss

$$||x - \hat{x}||_2$$

[Hinton and Salakhutdinov, Science 2006]

Variational Autoencoders (VAEs)



VAE with two-dimensional latent space

[Kingma and Welling, 2014]

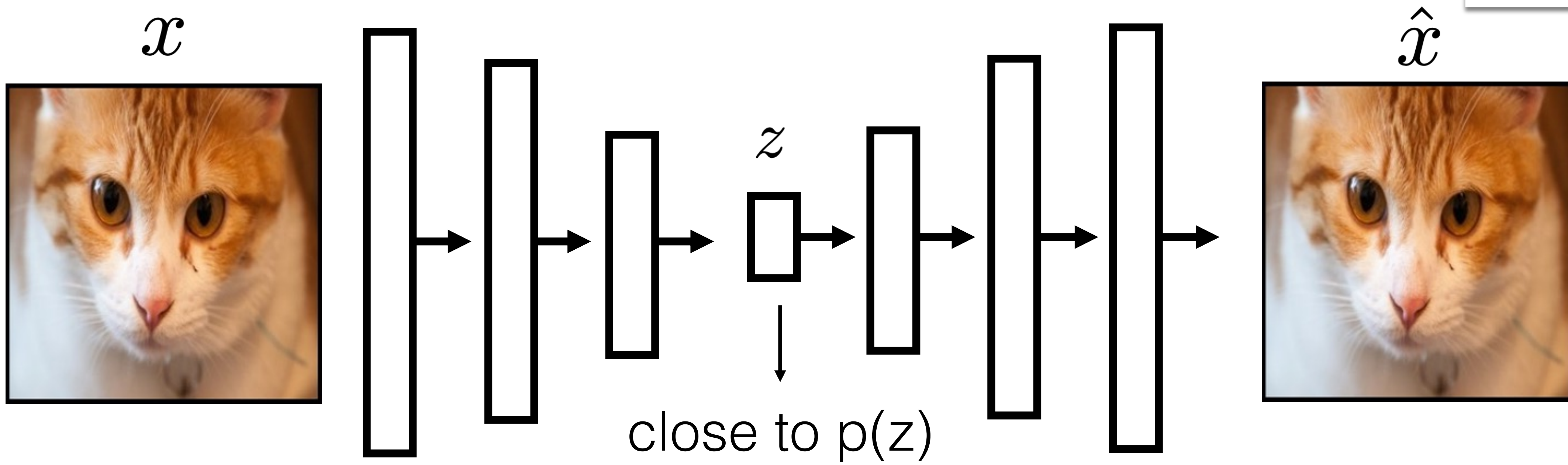
How to improve VAE?

- Why are the results blurry?
    - L2 reconstruction loss?
    - Lower bound might not be tight?

- How can we further improve results?

VAE + Perceptual Loss

encoder $q_\psi(z|x)$ $z = E^\mu_\psi(x) + E^\sigma_\psi(x) \cdot \epsilon_z$ generator $p_\theta(x|z)$ $\hat{x} = G^\mu_\theta(z)$



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

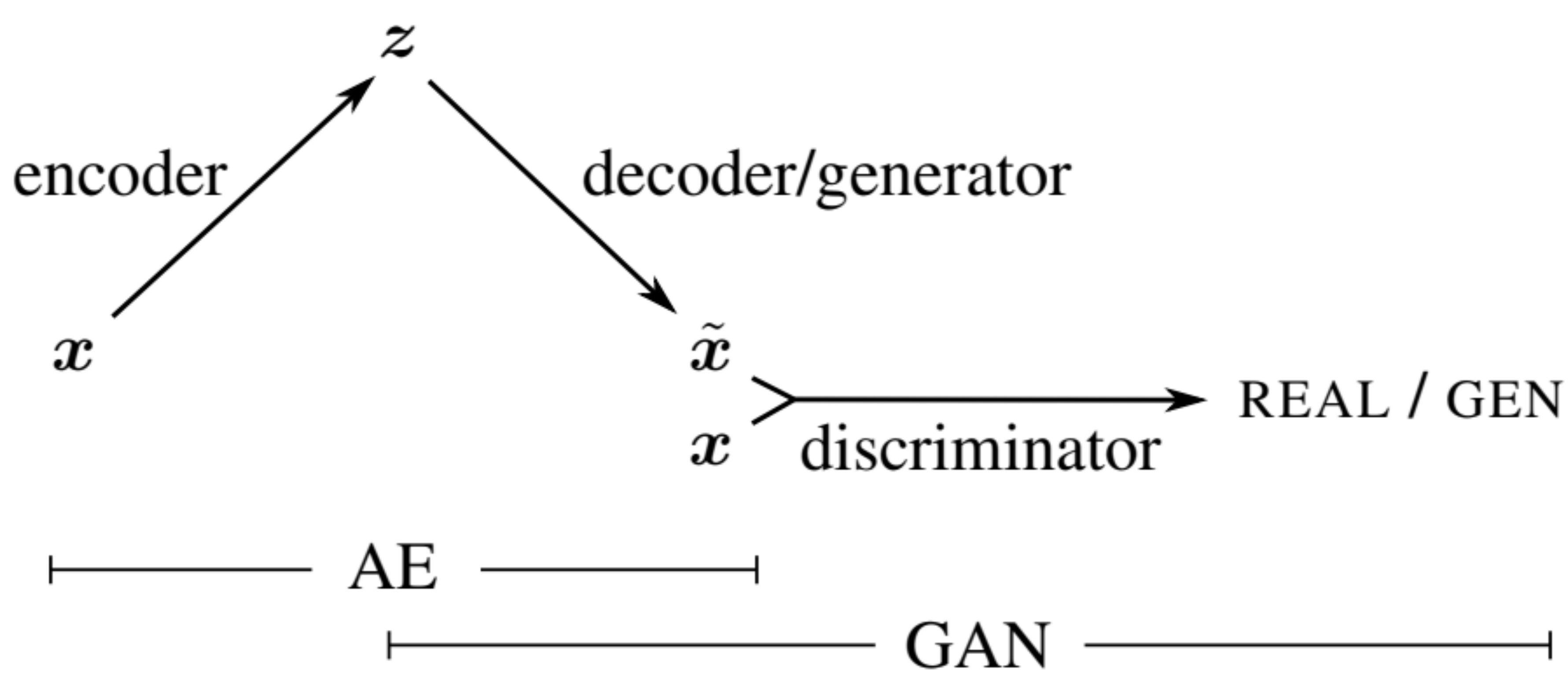
Multi-variate Gaussian

$$\geq \max_{\theta,\psi} \mathbb{E}_{x_i \sim p_{\mathtt{data}}}[\mathbb{E}_{q_\psi(z|x_i)}[p_\theta(x|z)] - \mathrm{KL}(q_\psi(z|x_i)||p(z))]$$

Perceptual loss KLD loss

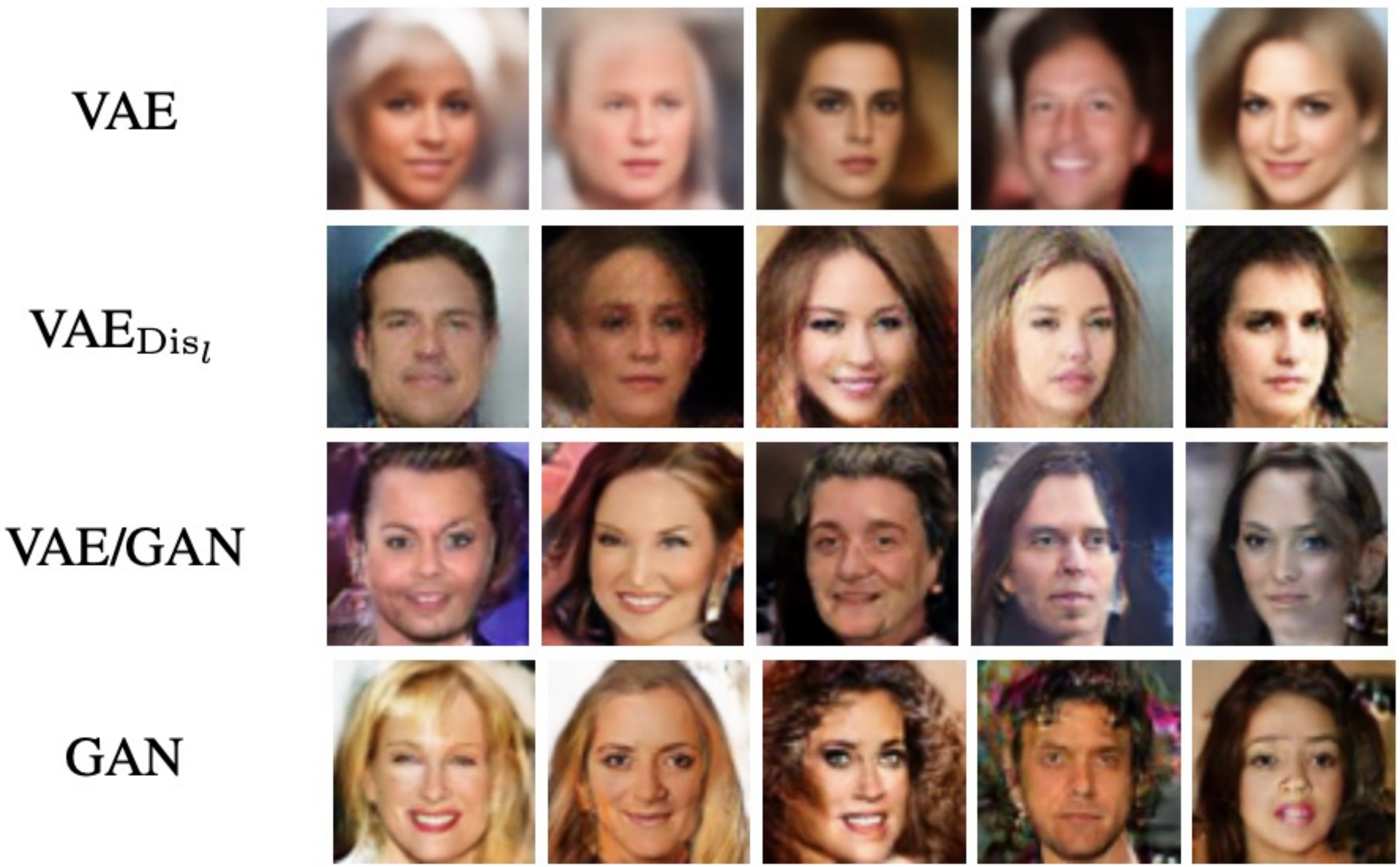
$$||F(x) - F(\hat{x})||_2 \quad \mathrm{KLD}(\mathcal{N}(E^\mu_\psi(x), E^\sigma_\psi(x)) \mid \mathcal{N}(0, I))$$

VAE + GANs



Autoencoding beyond pixels using a learned similarity metric [Larsen et al. 2015]

VAE + GANs



VAE(Disl) = VAE + feature matching loss

[Larsen et al. 2015]

Variational Autoencoder (VAE)

Data $\rightarrow$

Learner

Objective

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

Hypothesis space

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

Optimizer

Variational Bayes

$\longrightarrow$

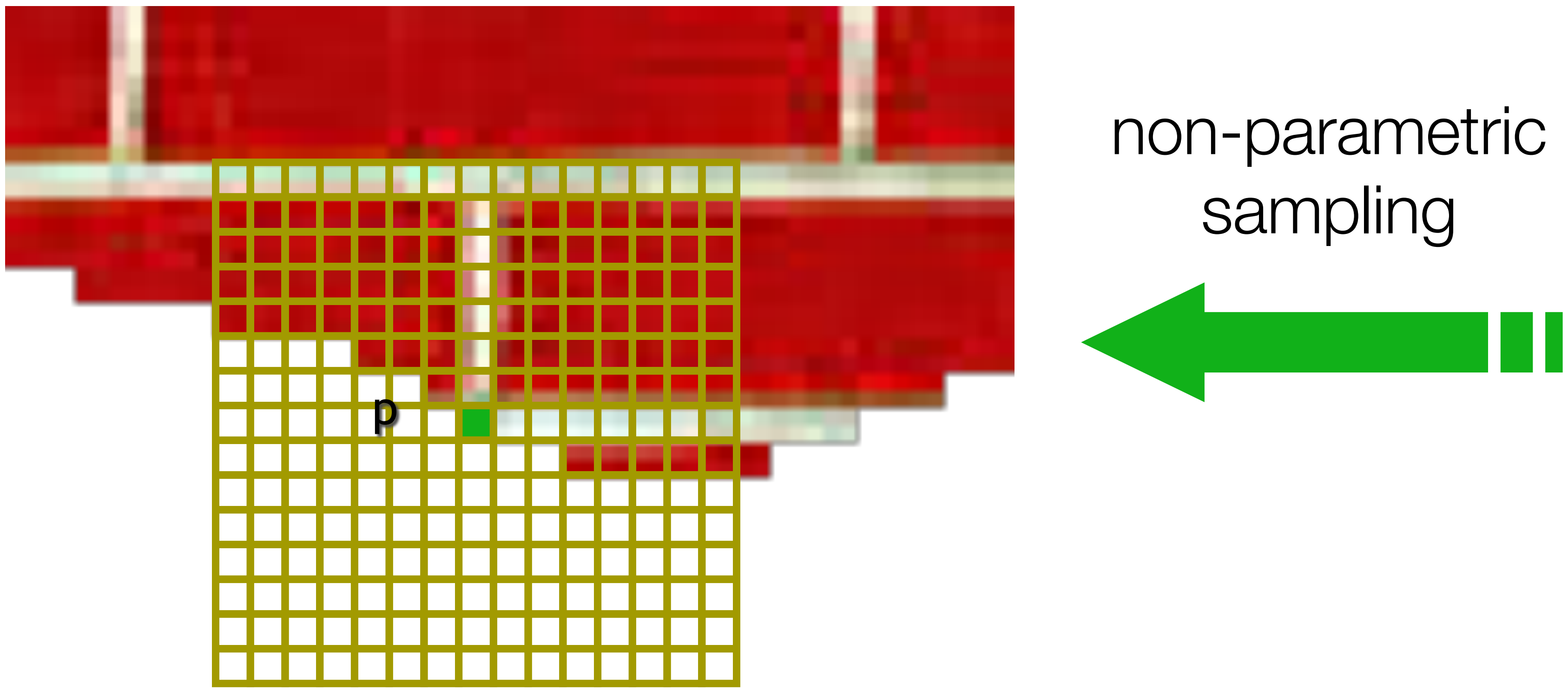
Sampler

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

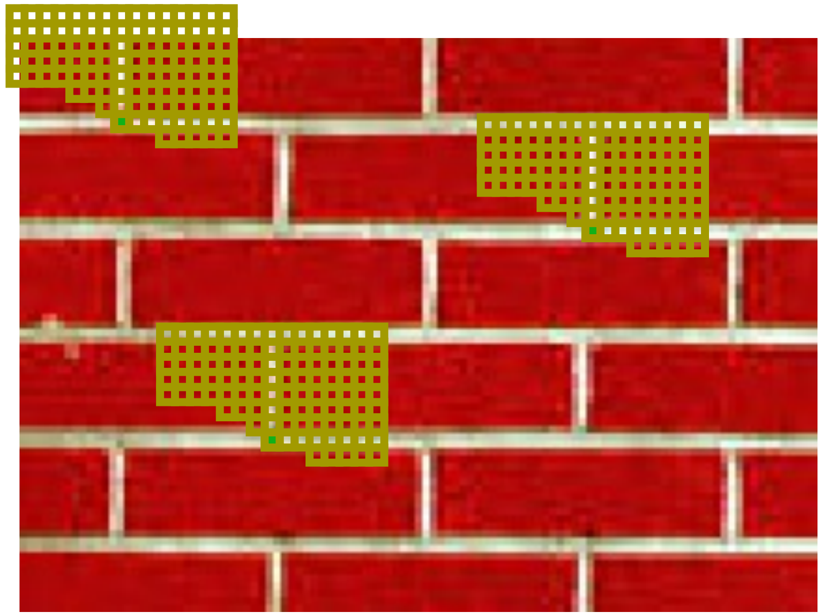
Autoregressive Model

Texture synthesis by non-parametric sampling

[Efros & Leung 1999]



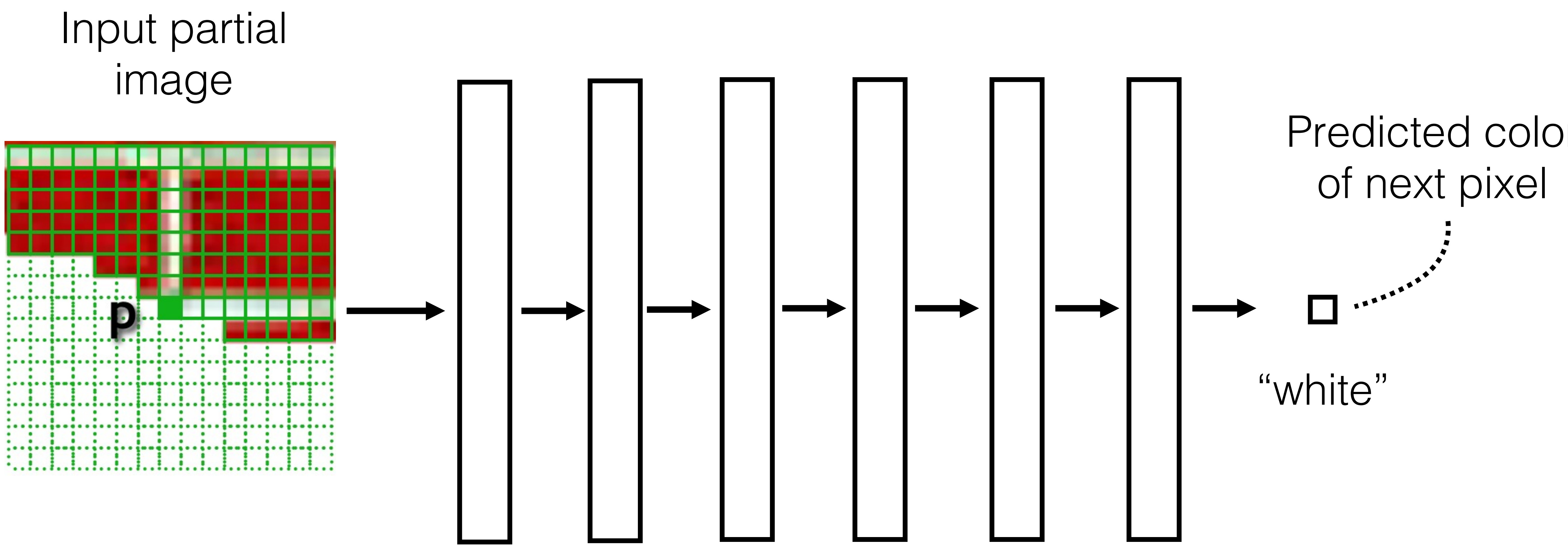
Synthesizing a pixel



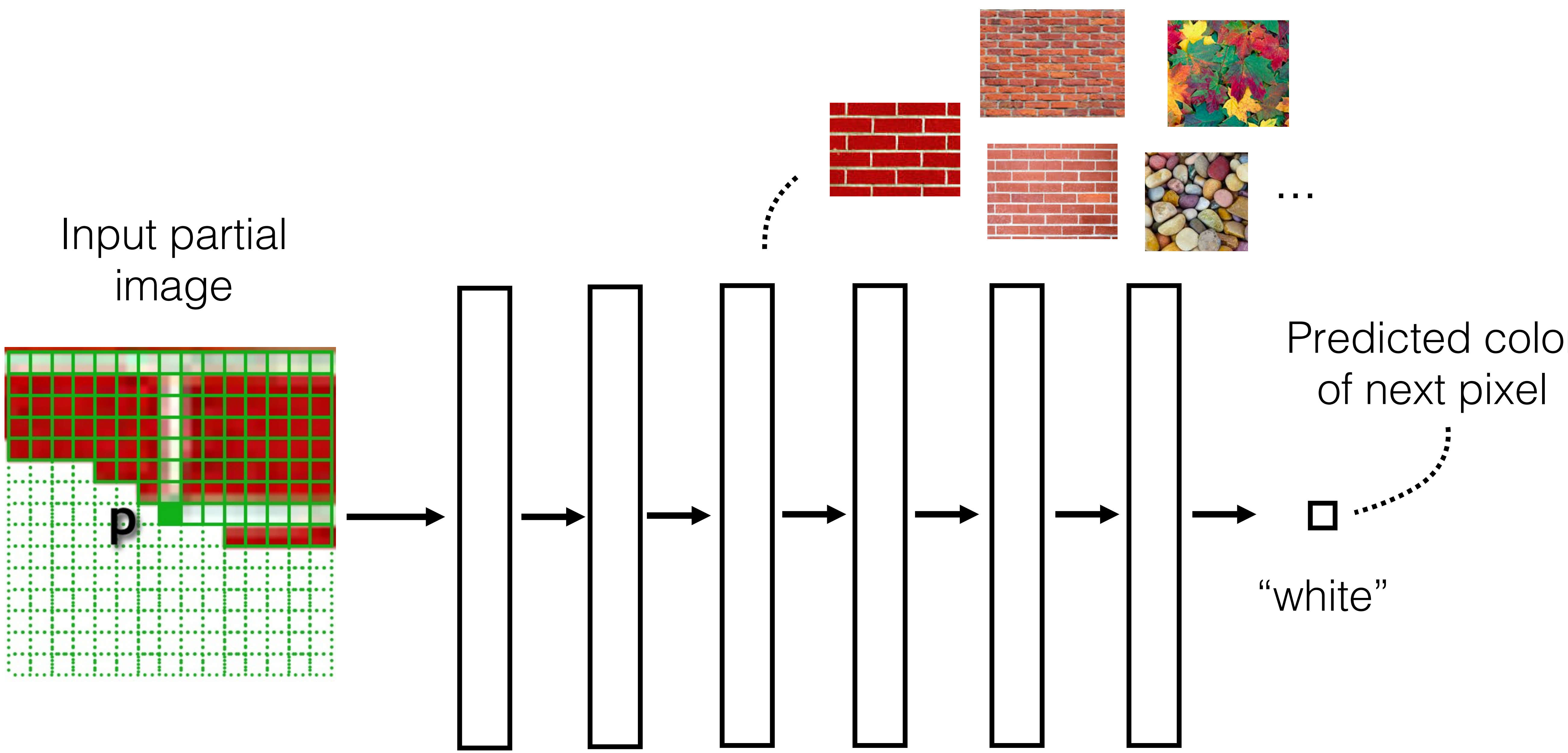
Input image

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

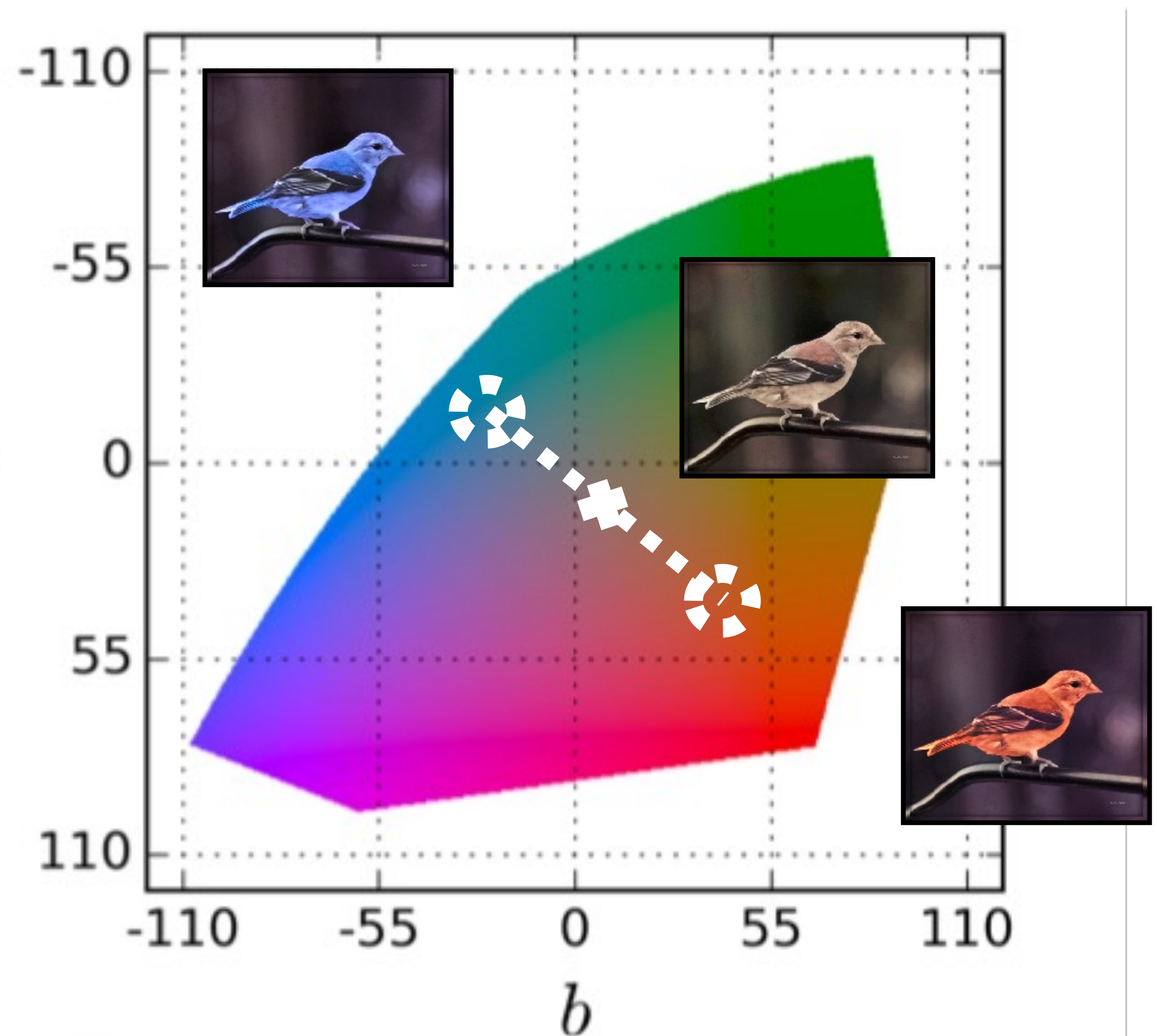
Autoregressive image synthesis



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



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



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

Classification Loss

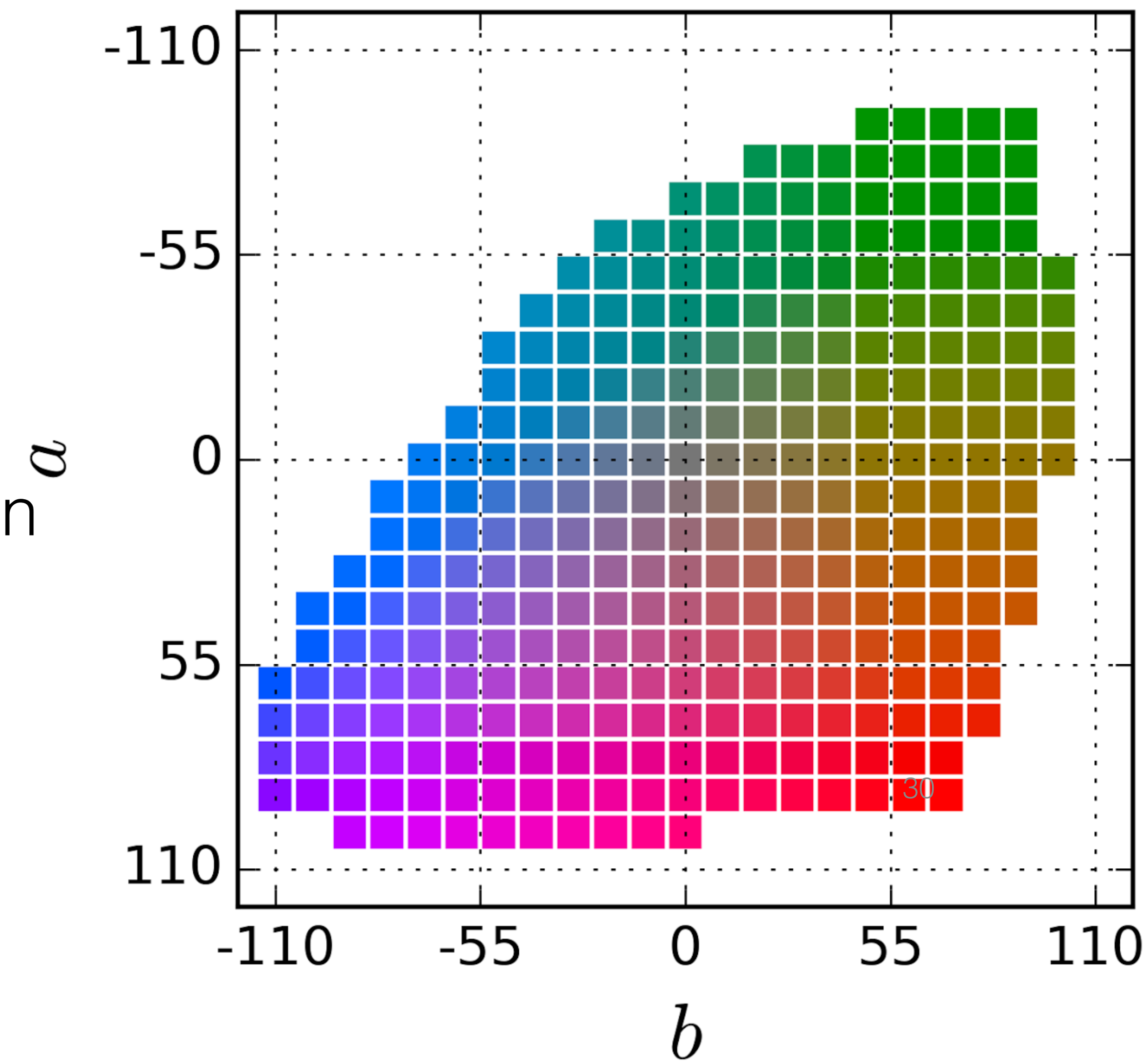
$$\theta^* = \arg\min_{\theta} \ell(\mathcal{F}_\theta(\mathbf{X}), \mathbf{Y})$$

- Regression with L2 loss inadequate

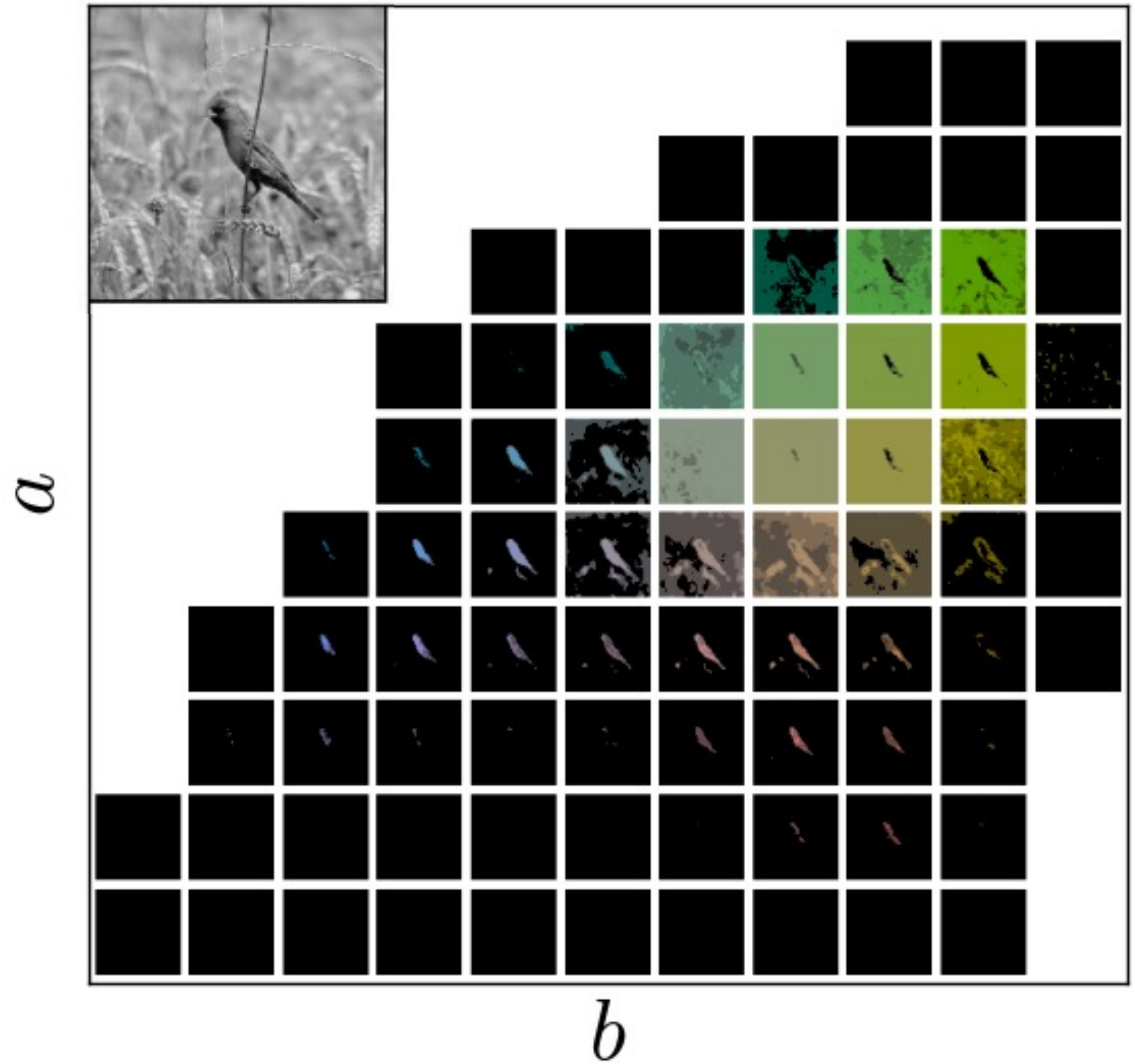
$$\mathrm{L}_2(\widehat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \widehat{\mathbf{Y}}_{h,w}\|_2^2$$

- Use per-pixel multinomial classification

$$\mathrm{L}(\widehat{\mathbf{Z}}, \mathbf{Z}) = -\frac{1}{HW} \sum_{h,w} \sum_{q} \mathbf{Z}_{h,w,q} \log(\widehat{\mathbf{Z}}_{h,w,q})$$



Colors in *ab* space
(discrete)


a
b

Designing loss functions

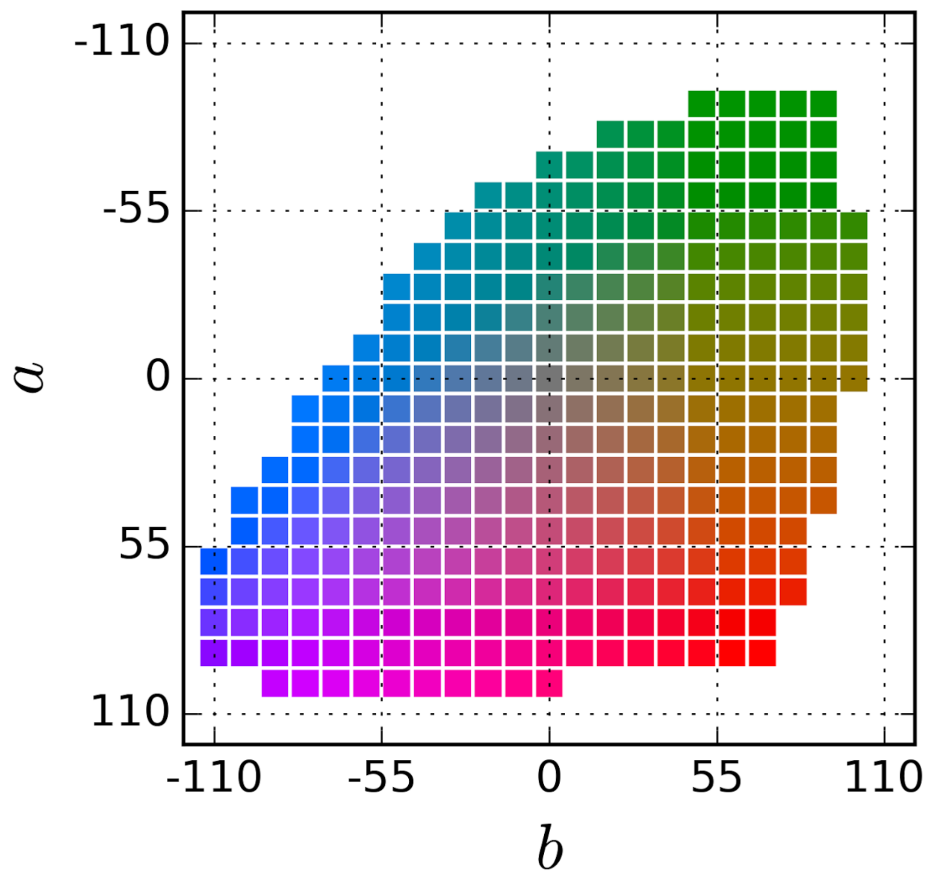
Input Zhang et al. 2016 Ground truth

Color distribution cross-entropy loss with colorfulness enhancing term.

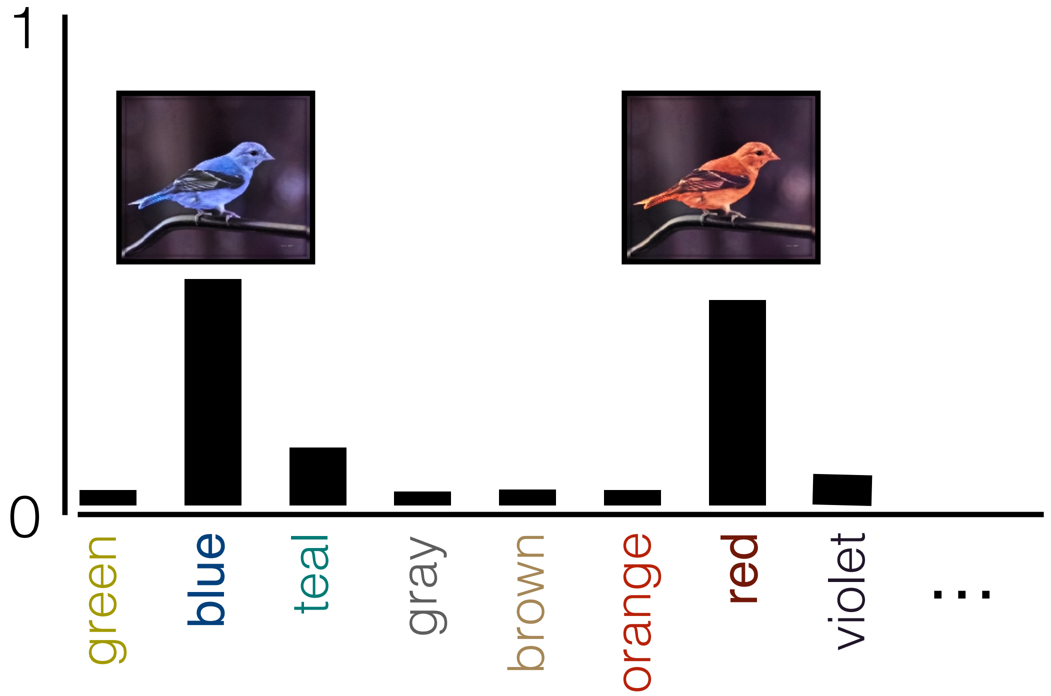
[Zhang, Isola, Efros, ECCV 2016]

Recall: we can represent colors as discrete classes

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



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**

One-hot vector

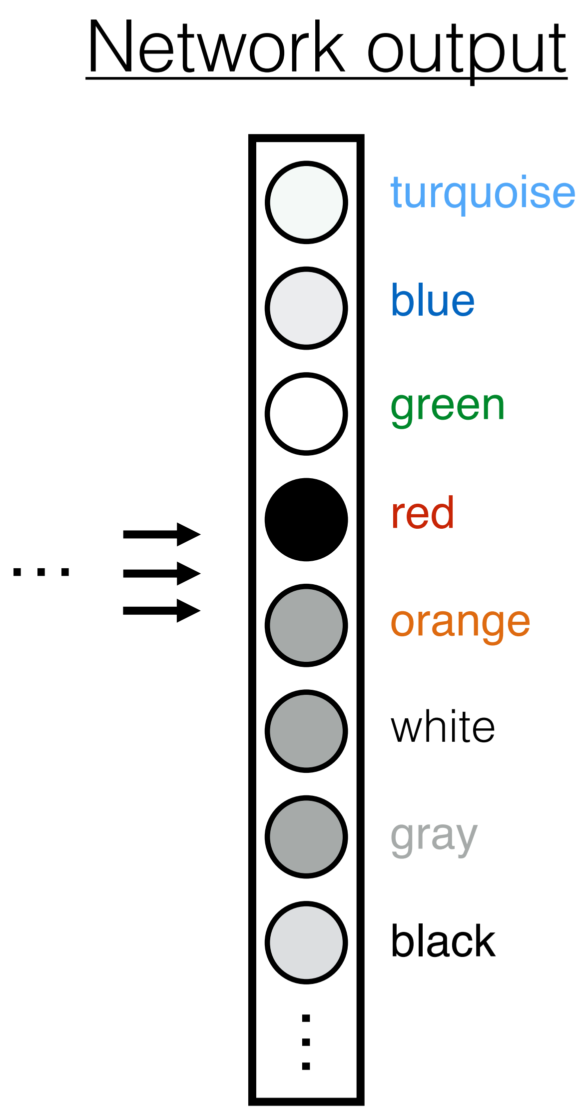
$$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

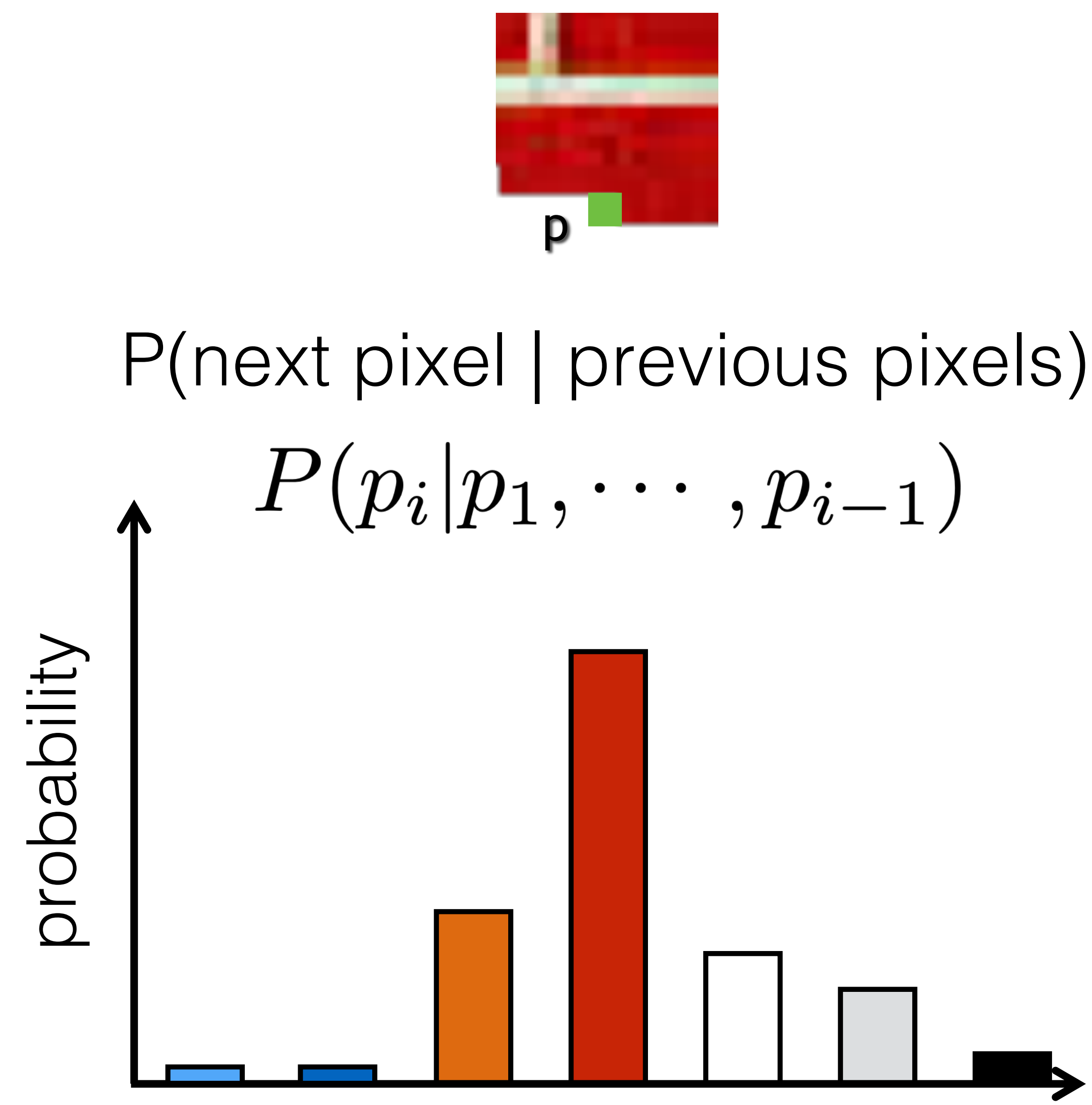
← picks out the -log likelihood of the ground truth class $\mathbf{y}$ under the model prediction $\hat{\mathbf{y}}$

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

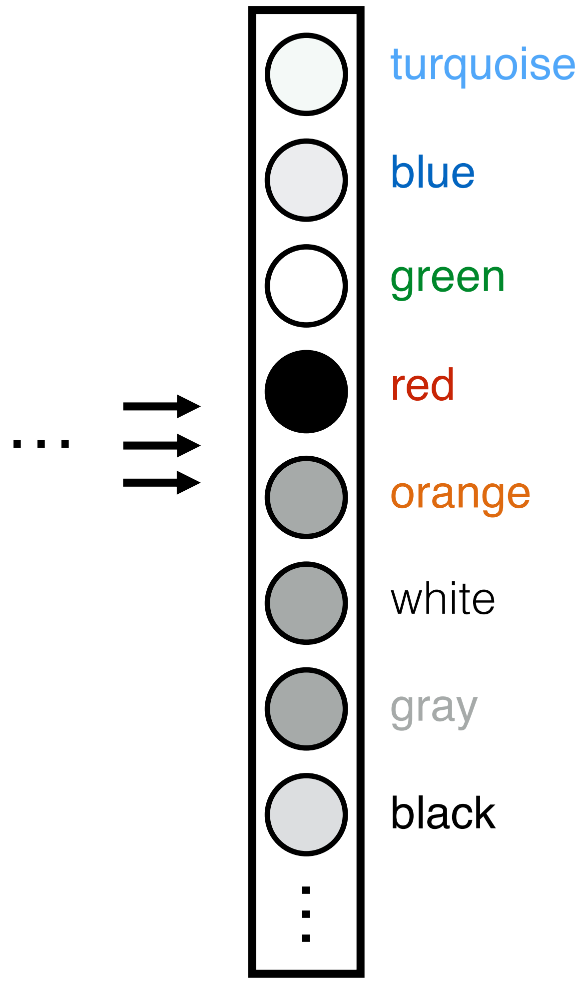
← max likelihood learner!

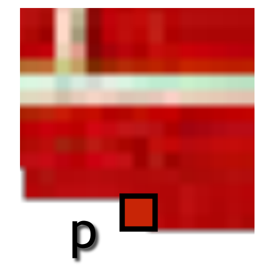
Cross-entropy loss


Network output
turquoise
blue
green
red
orange
white
gray
black

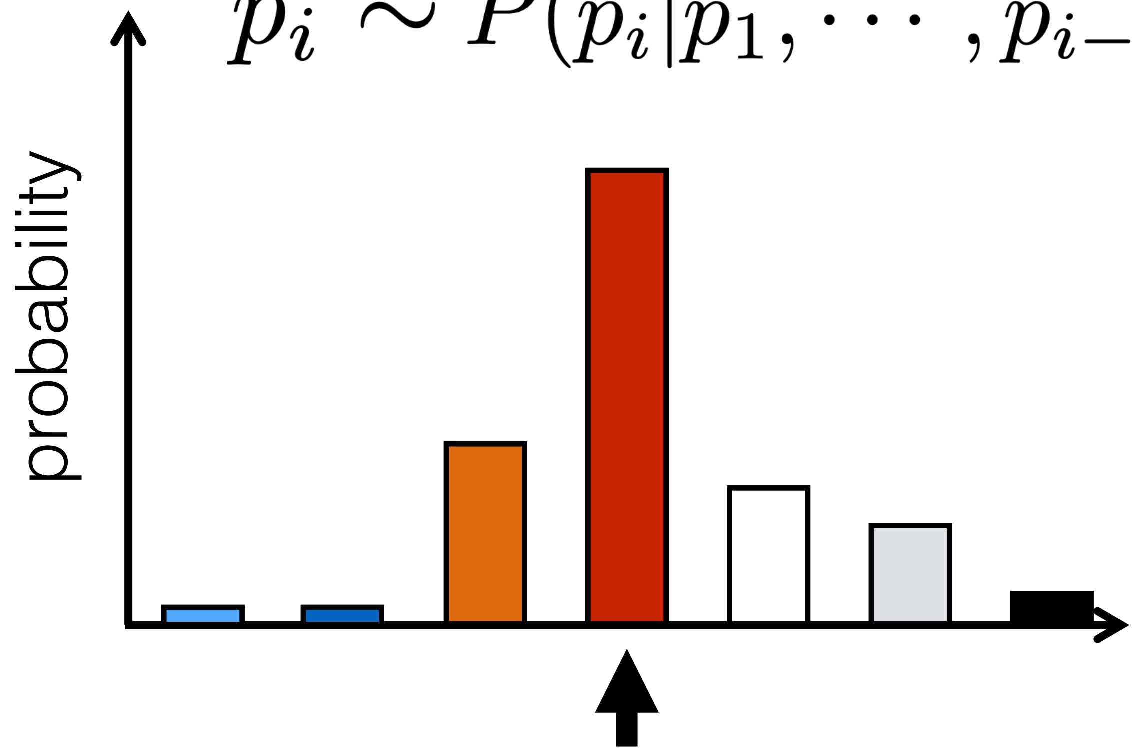

p
P(next pixel | previous pixels)
$P(p_i|p_1, \cdots, p_{i-1})$
probability

Network output

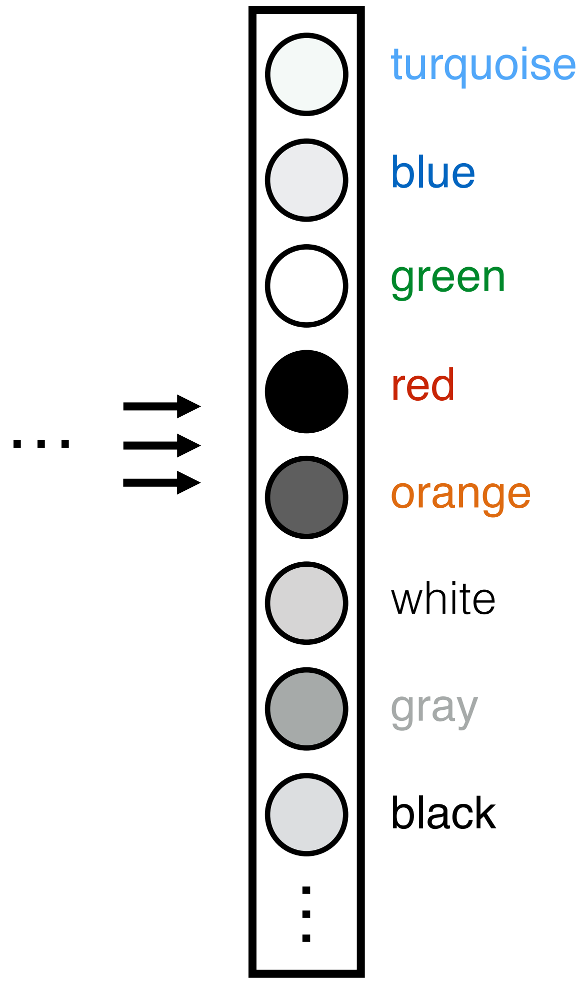




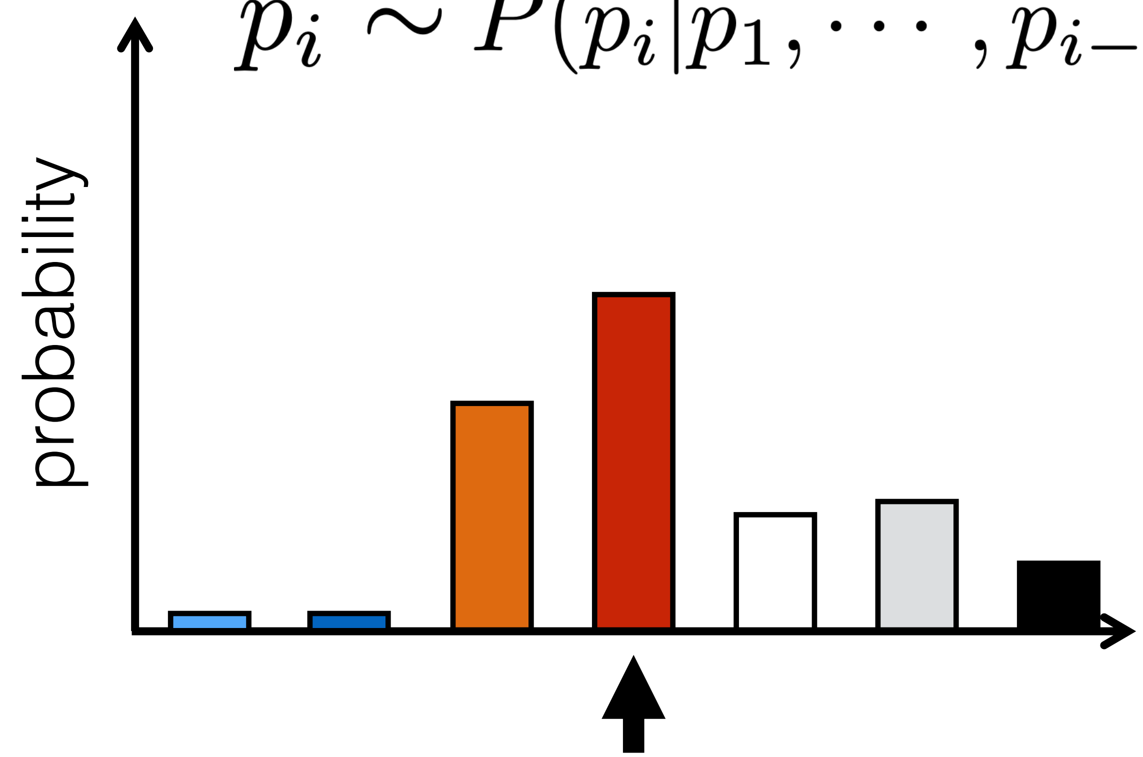
$$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$$



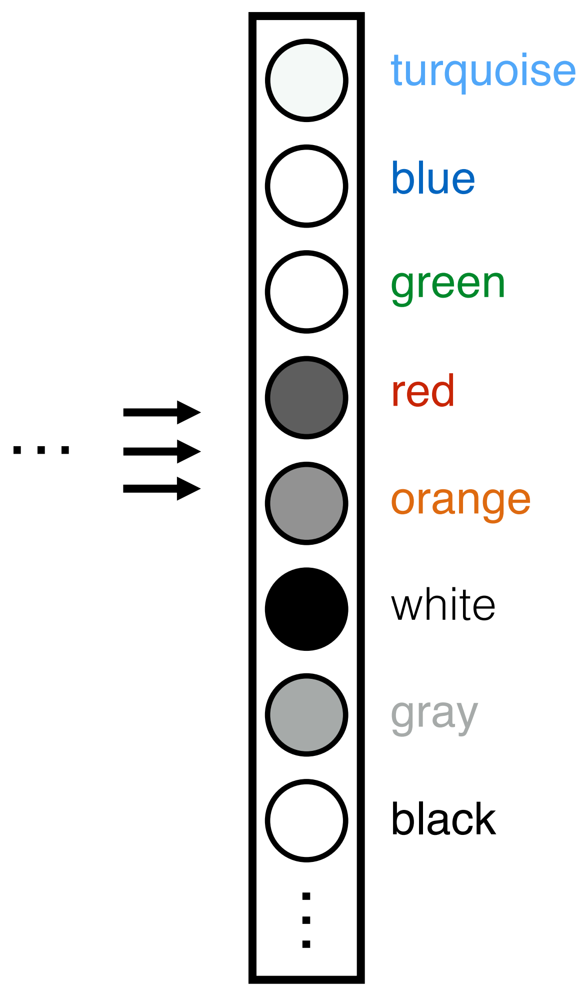
Network output



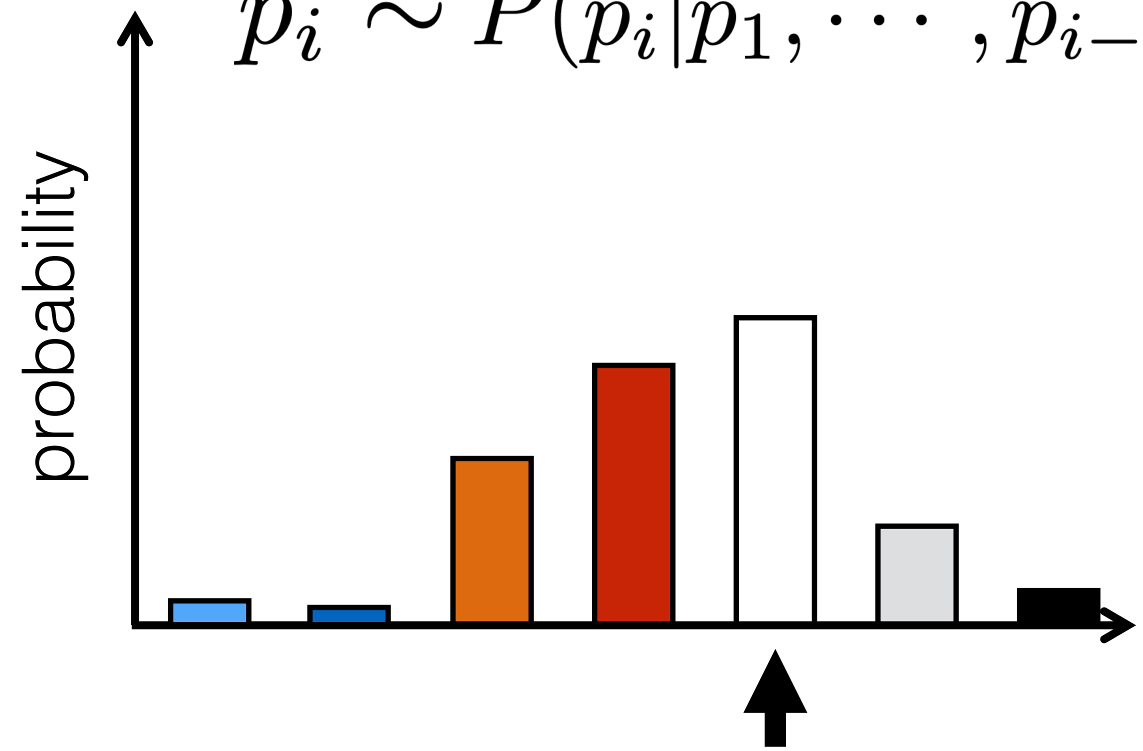
$$p_i \sim P(p_i|p_1, \cdots, p_{i-1})$$

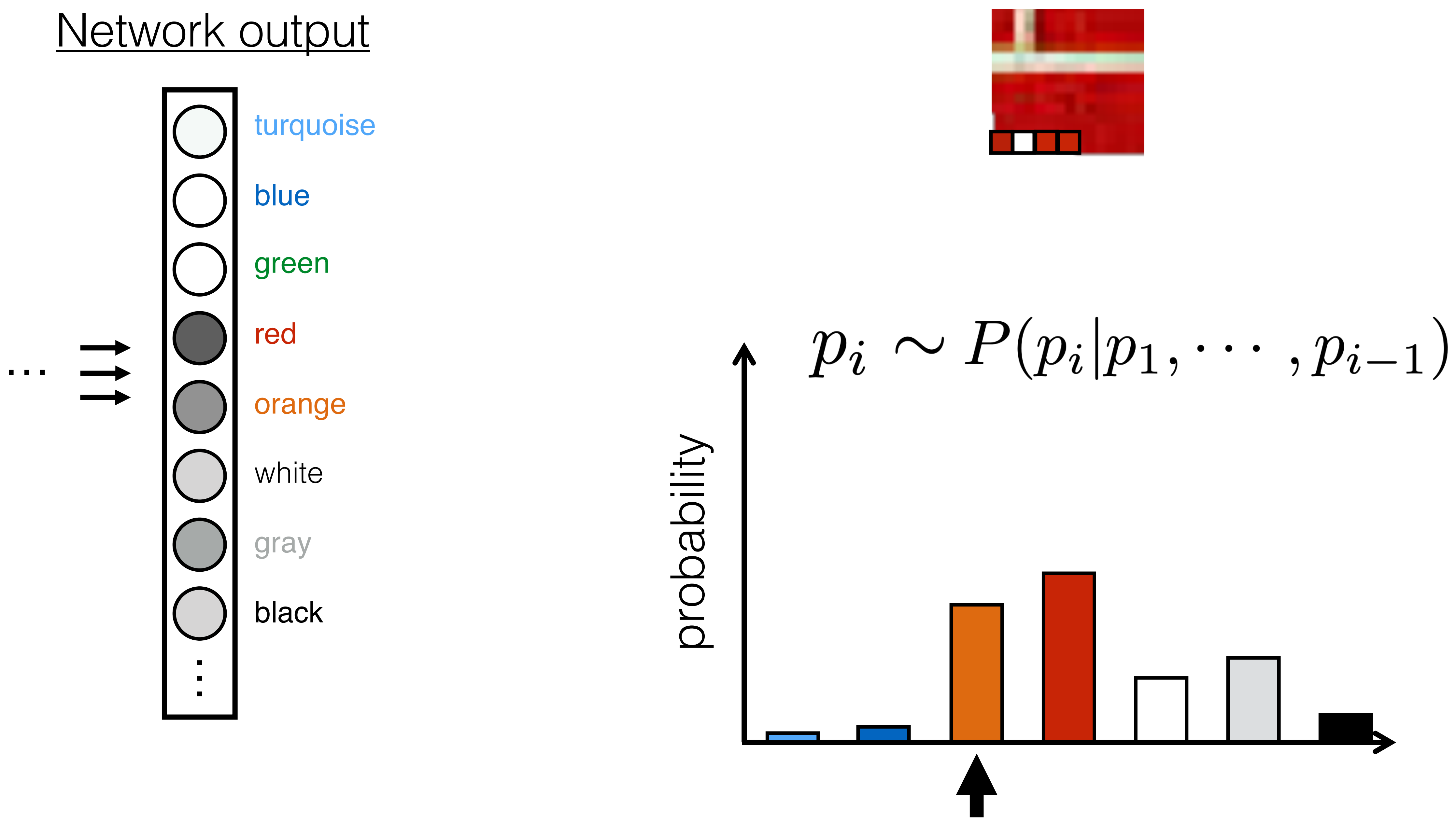


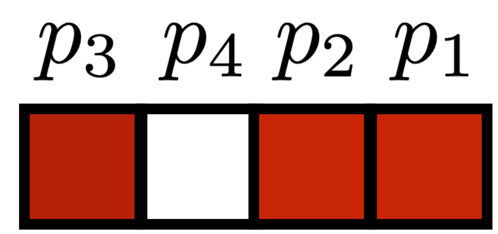
Network output



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




Network output
turquoise
blue
green
red
orange
white
gray
black
$p_i \sim P(p_i | p_1, \cdots, p_{i-1})$
probability



$$p_1 \sim P(p_1)$$

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

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

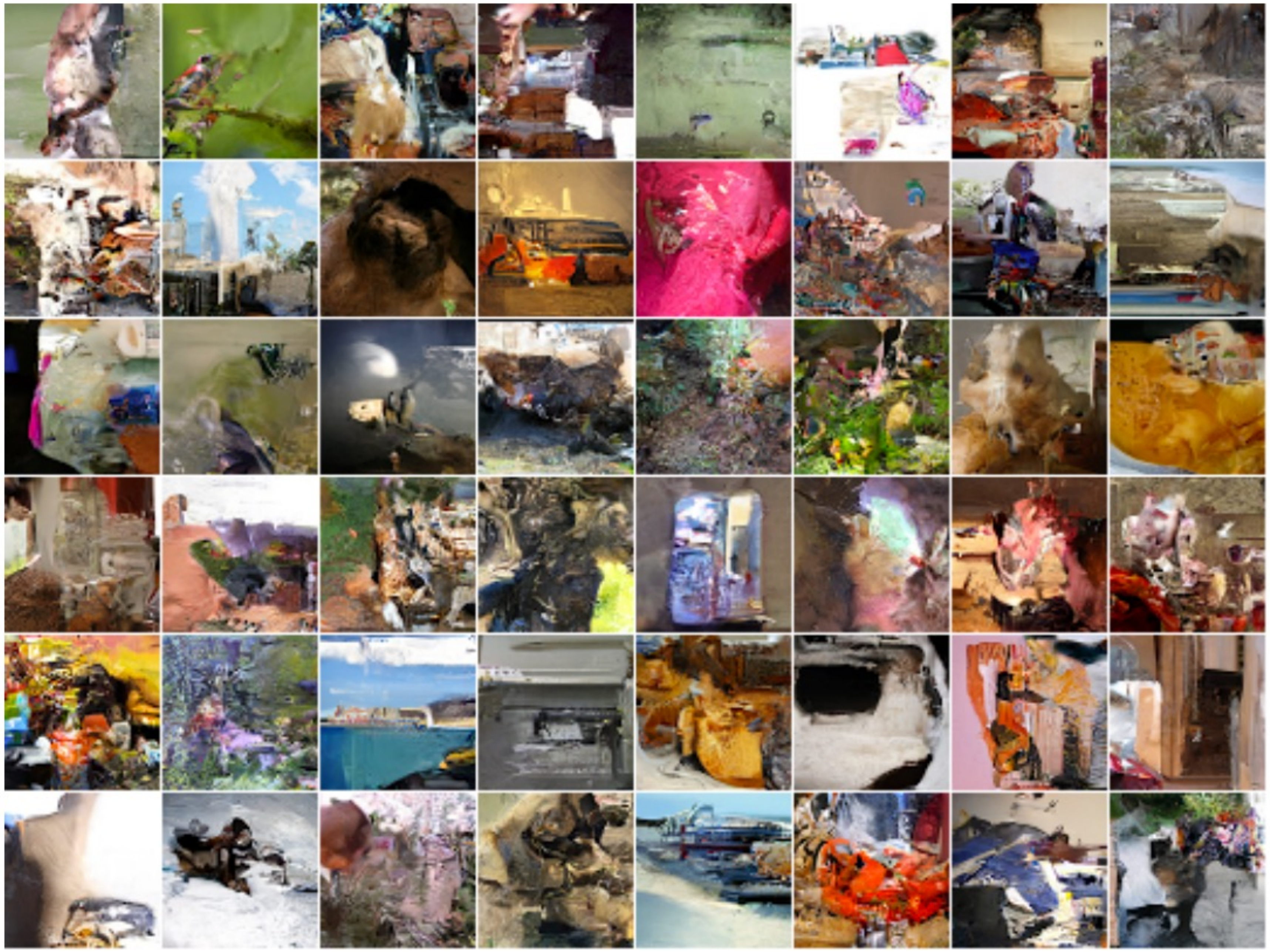
$$p_4 \sim P(p_4|p_1, p_2, p_3)$$

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

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

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

Samples from PixelRNN

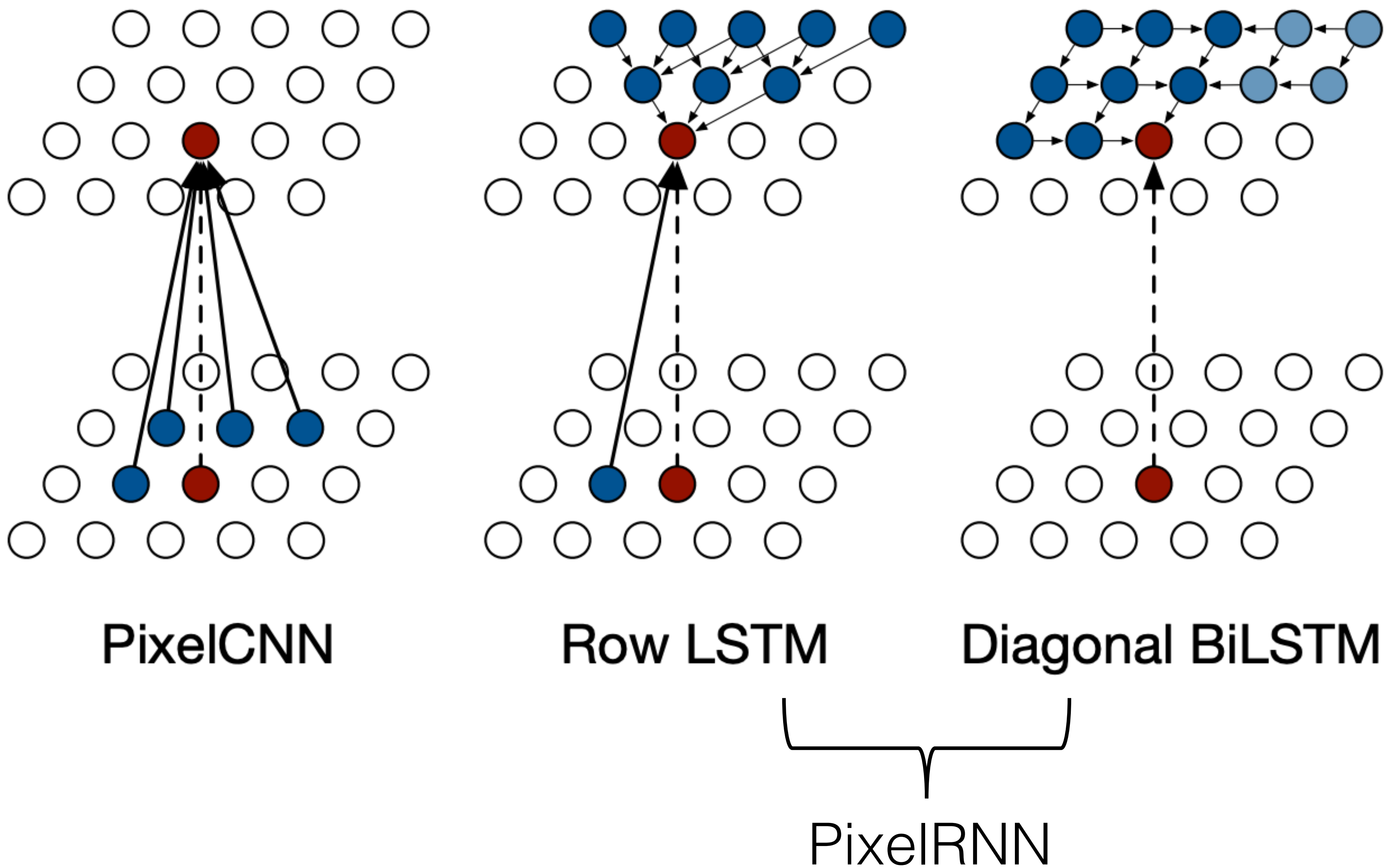


[PixelRNN, van der Oord et al. 2016]

Image completions (conditional samples) from PixelRNN

[PixelRNN, van der Oord et al. 2016]

PixelCNN vs. PixelRNN



Checkout PixelCNN++ [Salimans et al.,2017] (+ coarse-to-fine, ResNet, whole pixels, etc.)

How to improve PixelCNN?

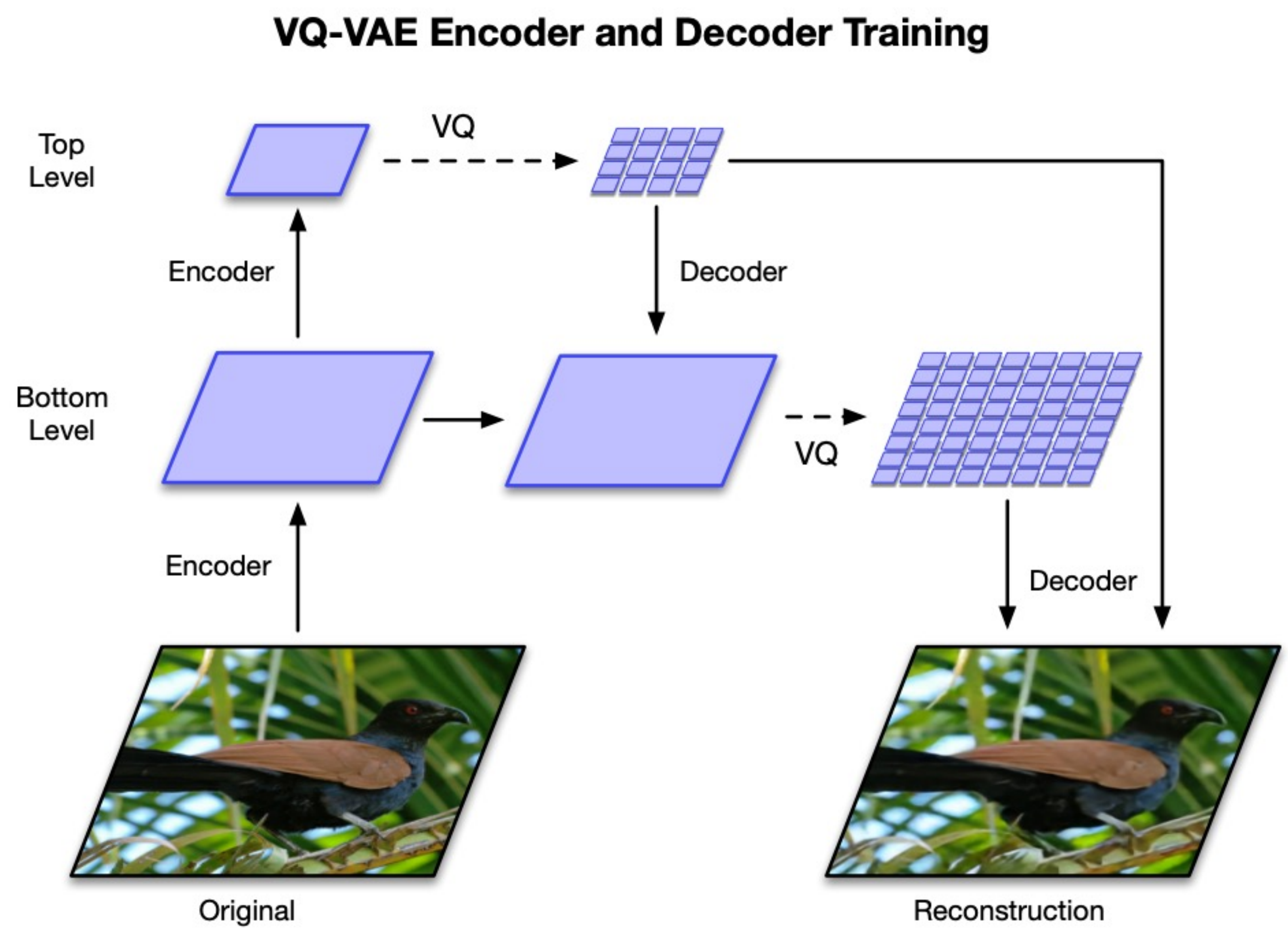
- What are the limitations of PixelCNN/RCN?
  - Slow sampling time.
  - May accumulate errors over multiple steps. (might not be a big issue for image completion)
- How can we further improve results?

VQ-VAE-2 :VAE+PixelCNN

VQ (Vector quantization) maps continuous vectors into discrete codes
Common methods: clustering (e.g., k-means)

Generating Diverse High-Fidelity Images with VQ-VAE-2 [Razavi et al., 2019]

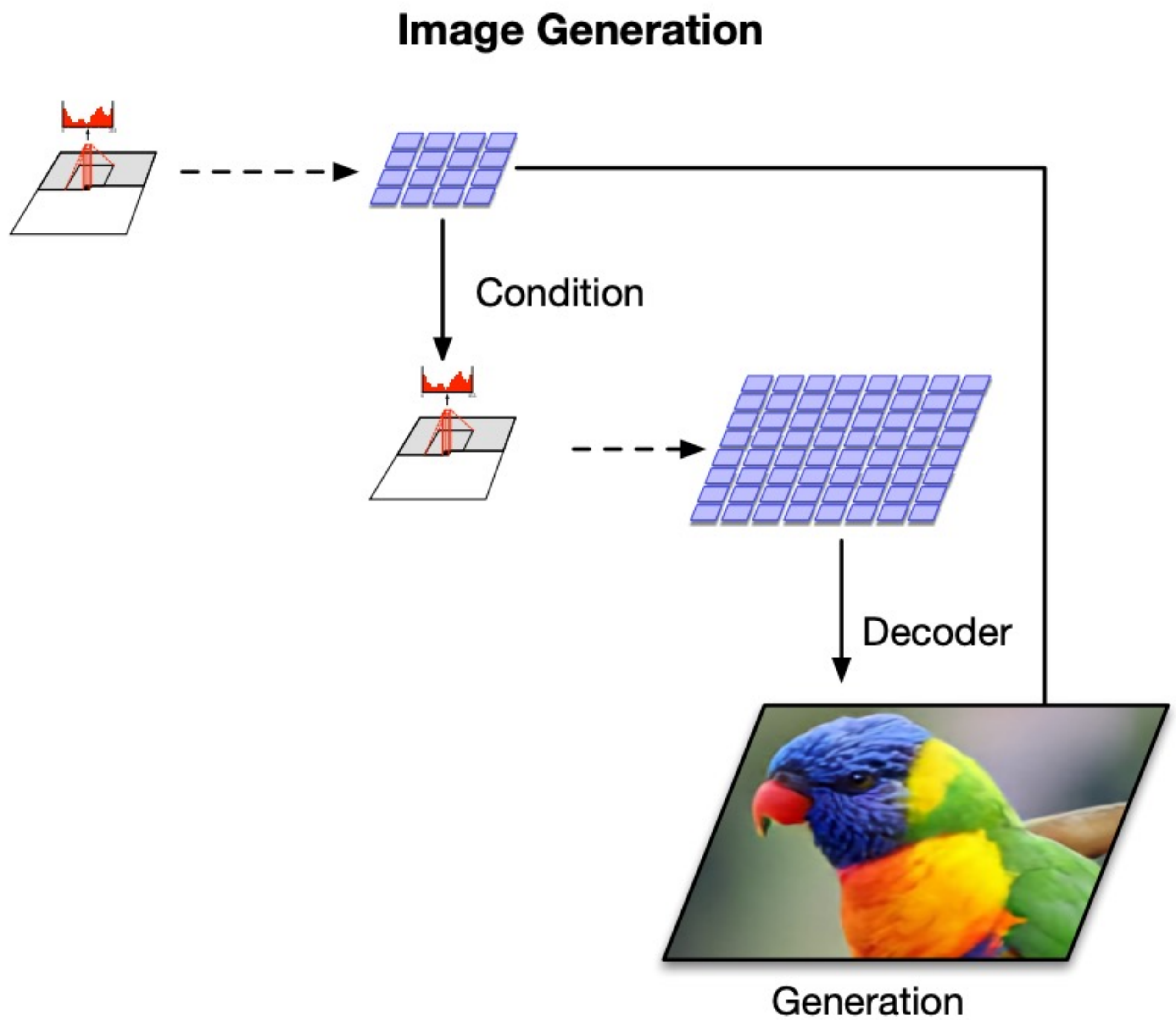
VQ-VAE-2: VAE+PixelCNN



<u>VAE+VQ</u>: learn a more compact codebook for PixelCNN (instead of pixels)
<u>PixelCNN</u>: use a more expressive bottleneck for VAE (instead of Gaussian)

[Razavi et al., 2019]

VQ-VAE-2: VAE+PixelCNN



VAE+VQ: learn a more compact codebook for PixelCNN (instead of pixel colors)
PixelCNN: use a more expressive bottleneck for VAE (instead of Gaussian prior)

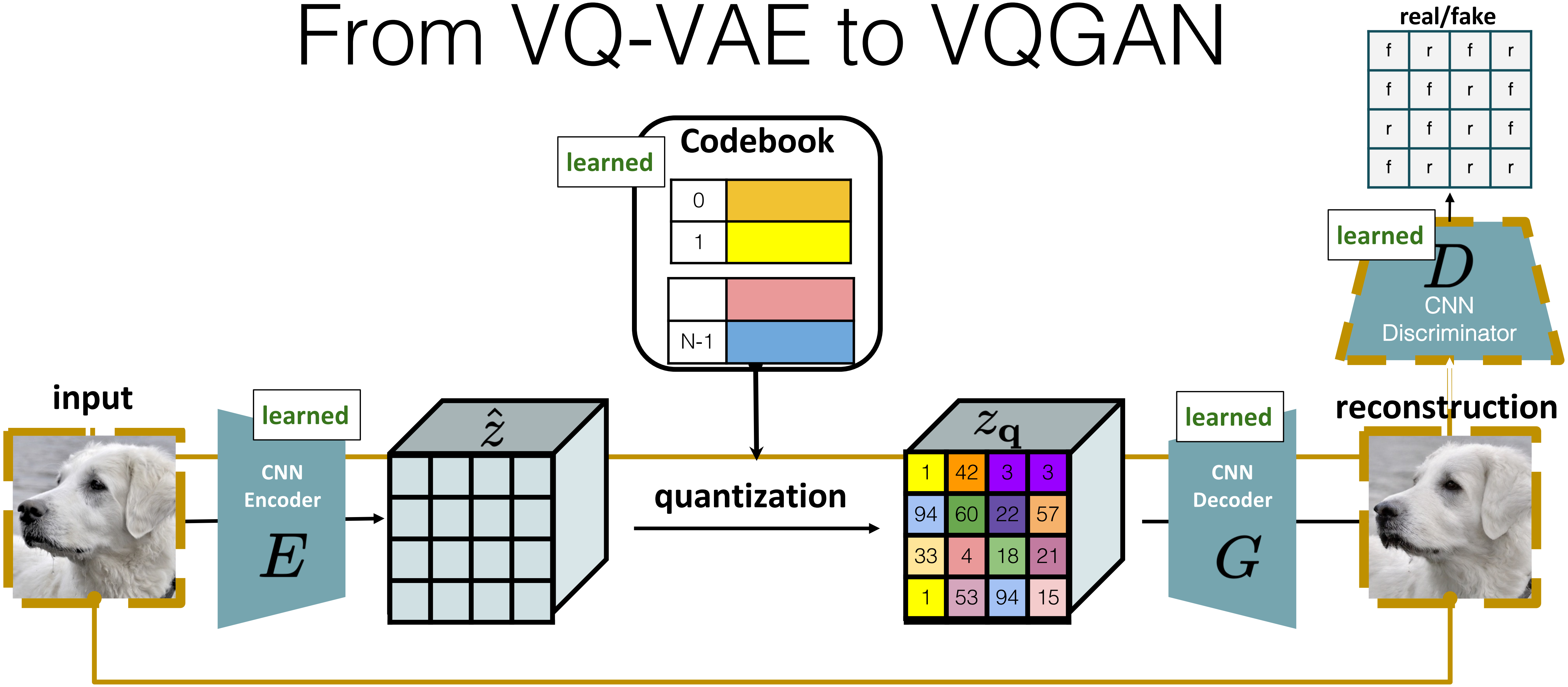
[Razavi et al., 2019]

How to Improve further?

- Better architectures
- Better loss functions for encoder-decoder

[Razavi et al., 2019]

From VQ-VAE to VQGAN

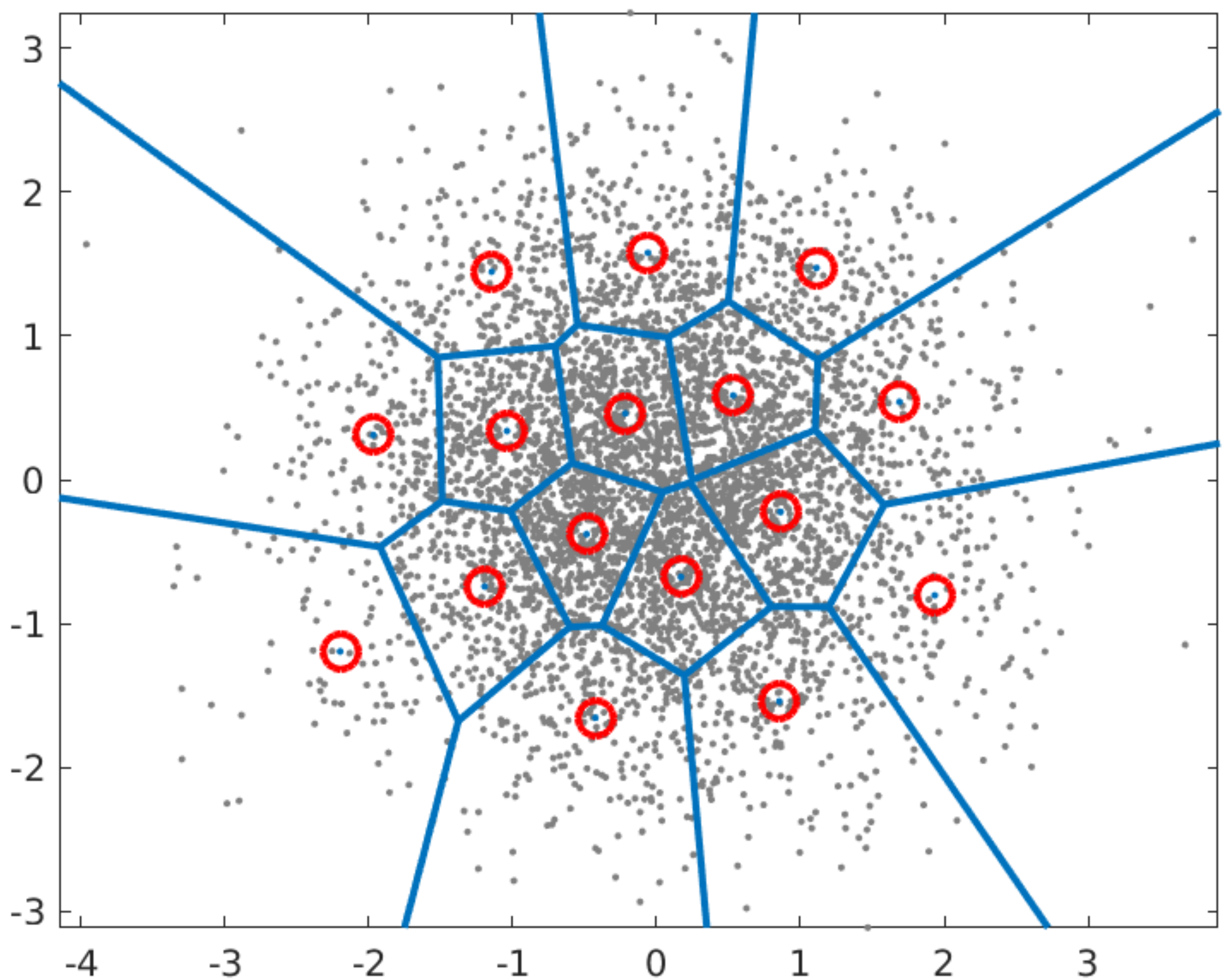


1) replace L2/L1 rec. loss with Perceptual loss (includes pixel-level)

2) add Discriminator to favor realism over per-pixel reconstruction

Slide credit: Robin Rombach

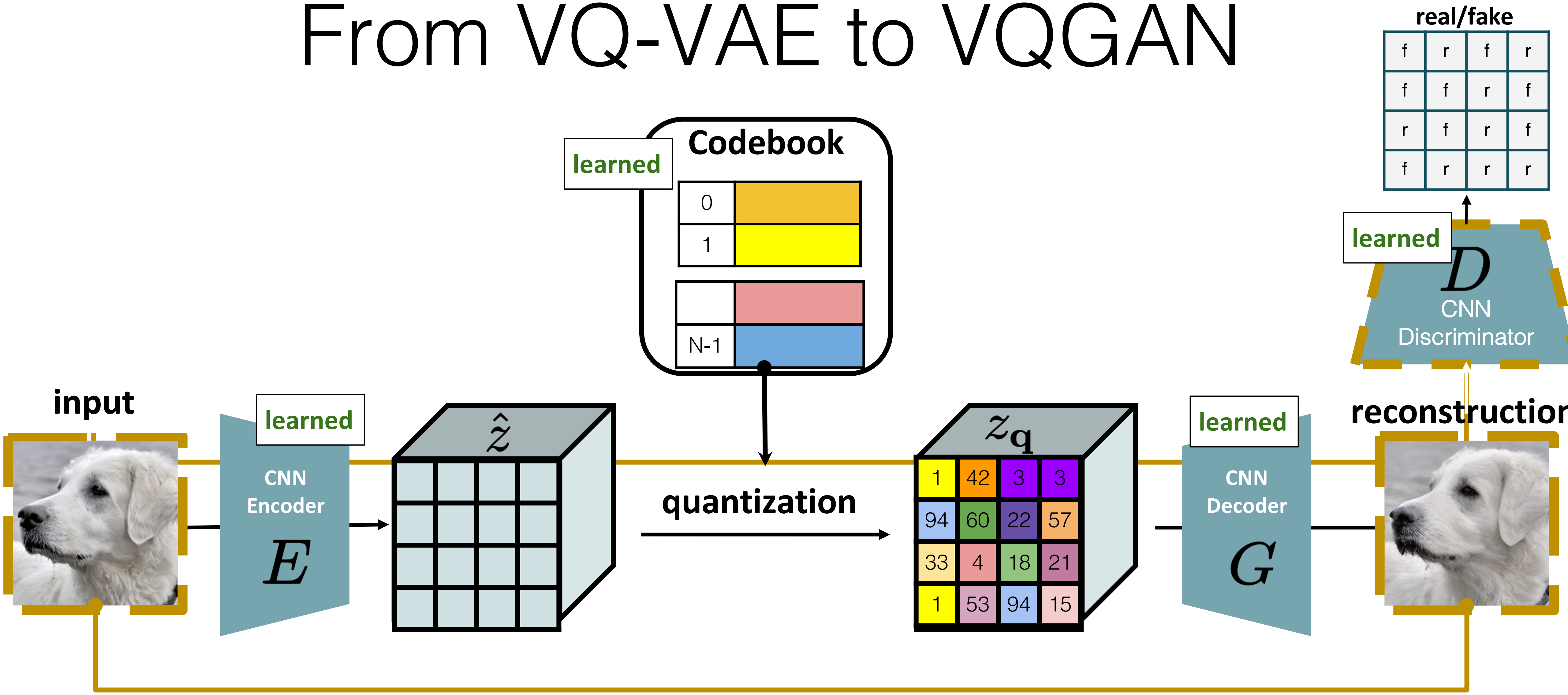
Vector Quantization (VQ)



K-means, EM (GMM), end-to-end learning

https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153

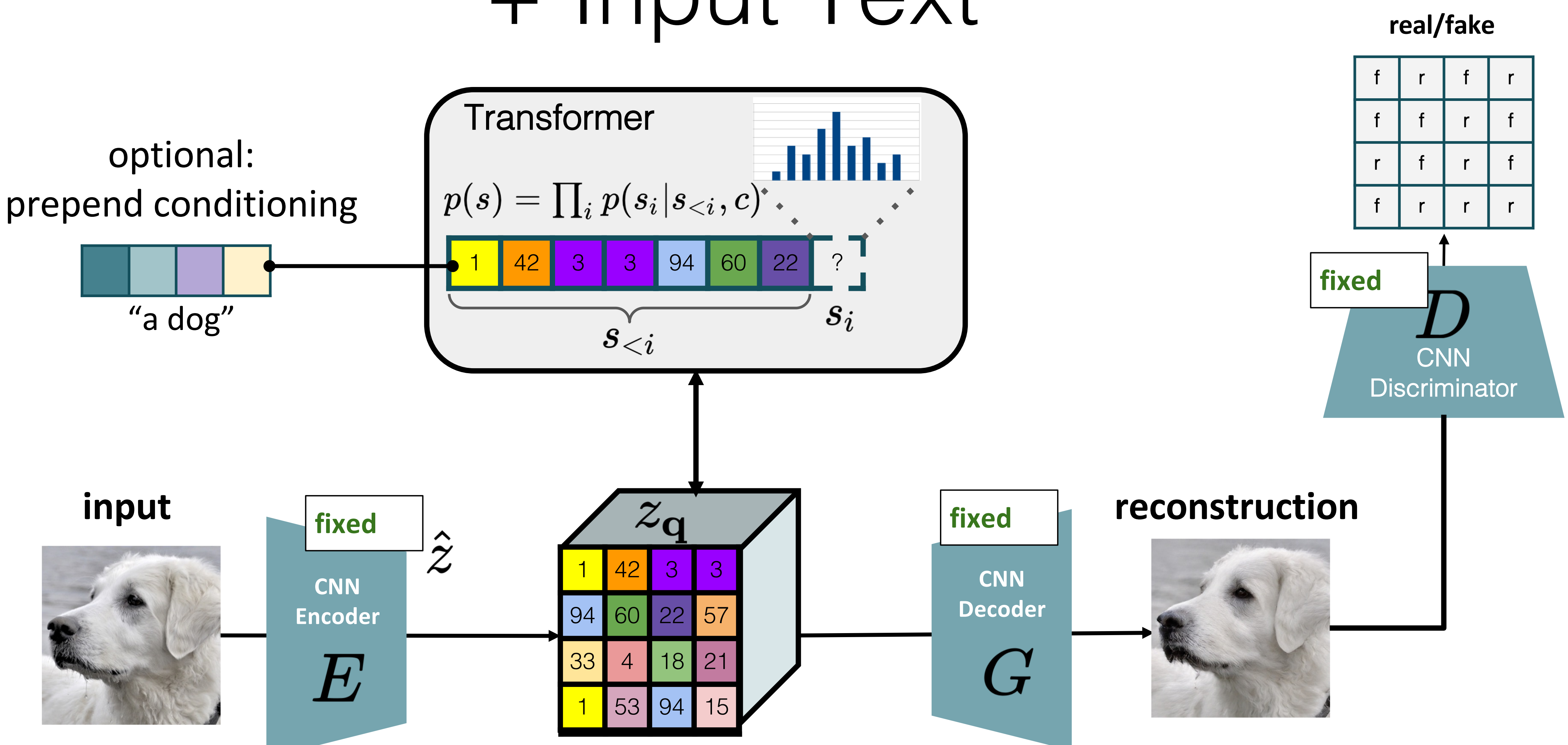
From VQ-VAE to VQGAN



$$\mathcal{L}_{\text{total}} = \mathcal{L}_{rec} + \mathcal{L}_{VQ} + \lambda \mathcal{L}_{GAN} \quad \text{where} \quad \lambda = \frac{\nabla_{G_L}[\mathcal{L}_{rec}]}{\nabla_{G_L}[\mathcal{L}_{GAN}] + \delta}$$

Slide credit: Robin Rombach

Transformer Training + Input Text



Slide credit: Robin Rombach

Slide credit: Robin Rombach

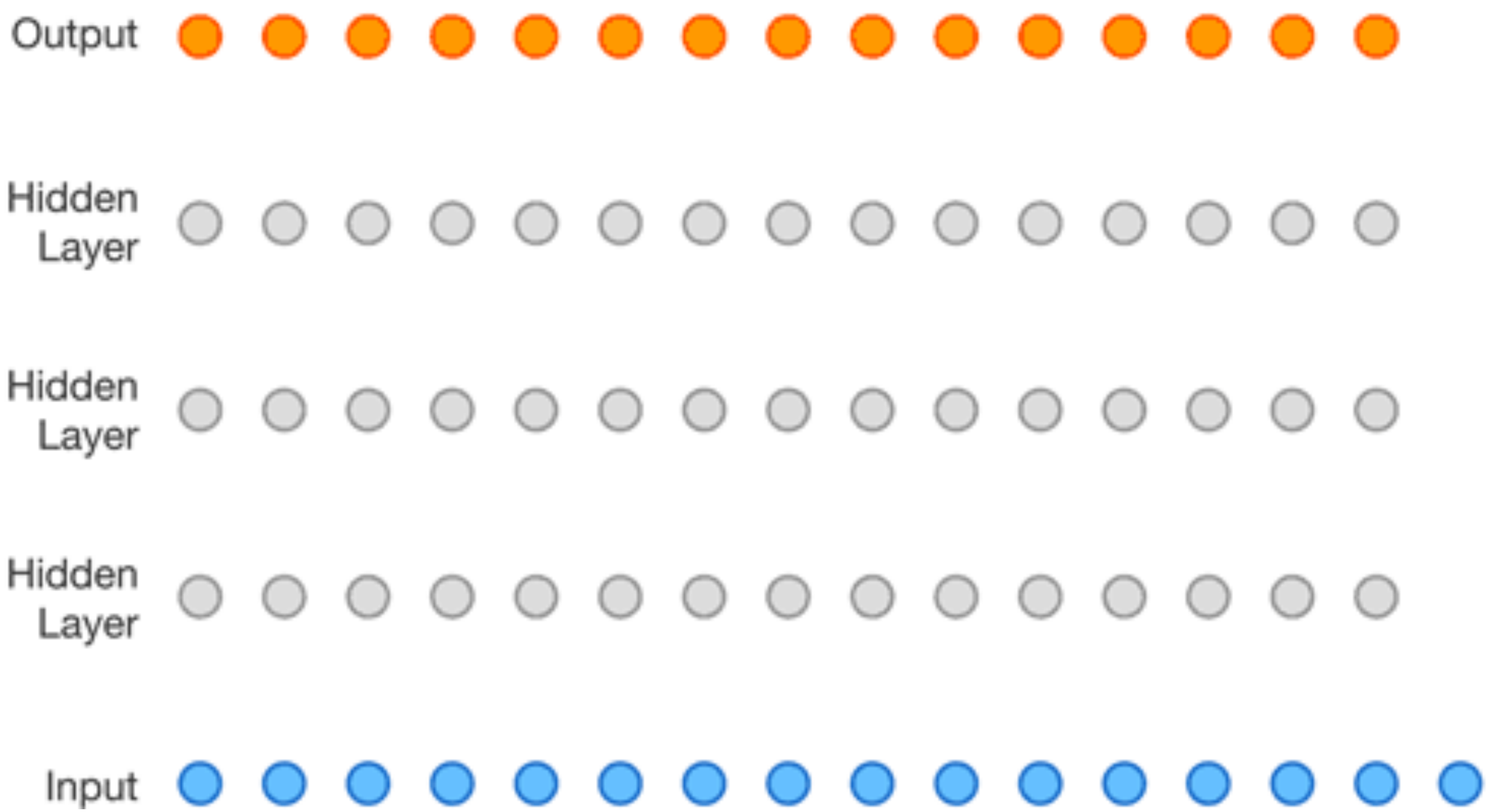
Slide credit: Robin Rombach

Class-Conditional Synthesis on ImageNet

1.4B Model trained on single A100

Slide credit: Robin Rombach

WaveNet



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

Auto-regressive models works extremely well for audio/music data.

Autoregressive Model

Data $\rightarrow$

Learner

Objective

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

Hypothesis space

$$p_\theta(x) = \prod_{i=1}^{N} p_\theta^{\texttt{auto}}(x_i | x_1, \ldots, x_{i-1})$$

$$x = [f^1(z^1), f^2(f^1, z^2), \ldots, f^n(f^1, \cdots, f^{n-1}, z^n)]$$

$\rightarrow$

Density

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

Sampler

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

Thank You!

16-726, Spring 2023

https://learning-image-synthesis.github.io/sp23/