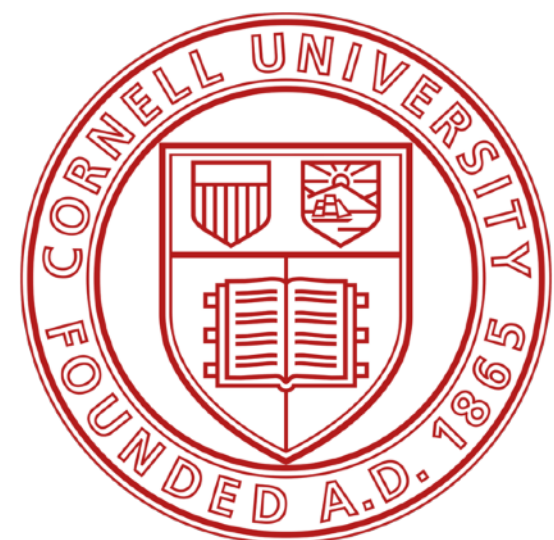


Lecture 7: Generative adversarial networks

CS 5788: Introduction to Generative Models



Many slides from Phillip Isola

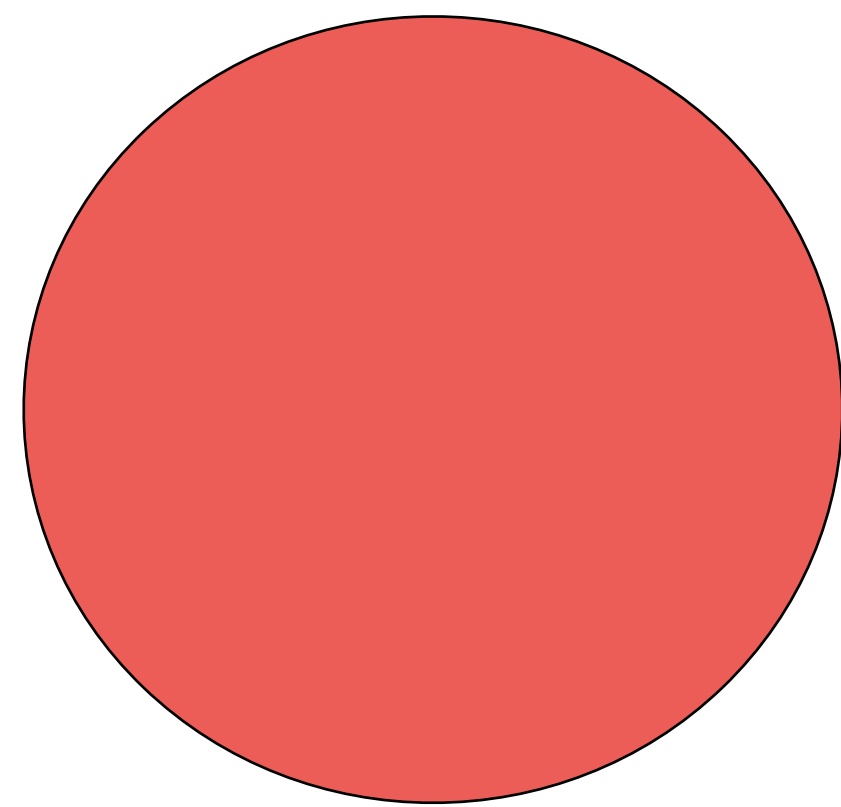
Today

- PS1 deadline pushed back 2 days
- See Ed Discussion for hint about initialization.
- PS2 will be out soon. We'll release it in parts.

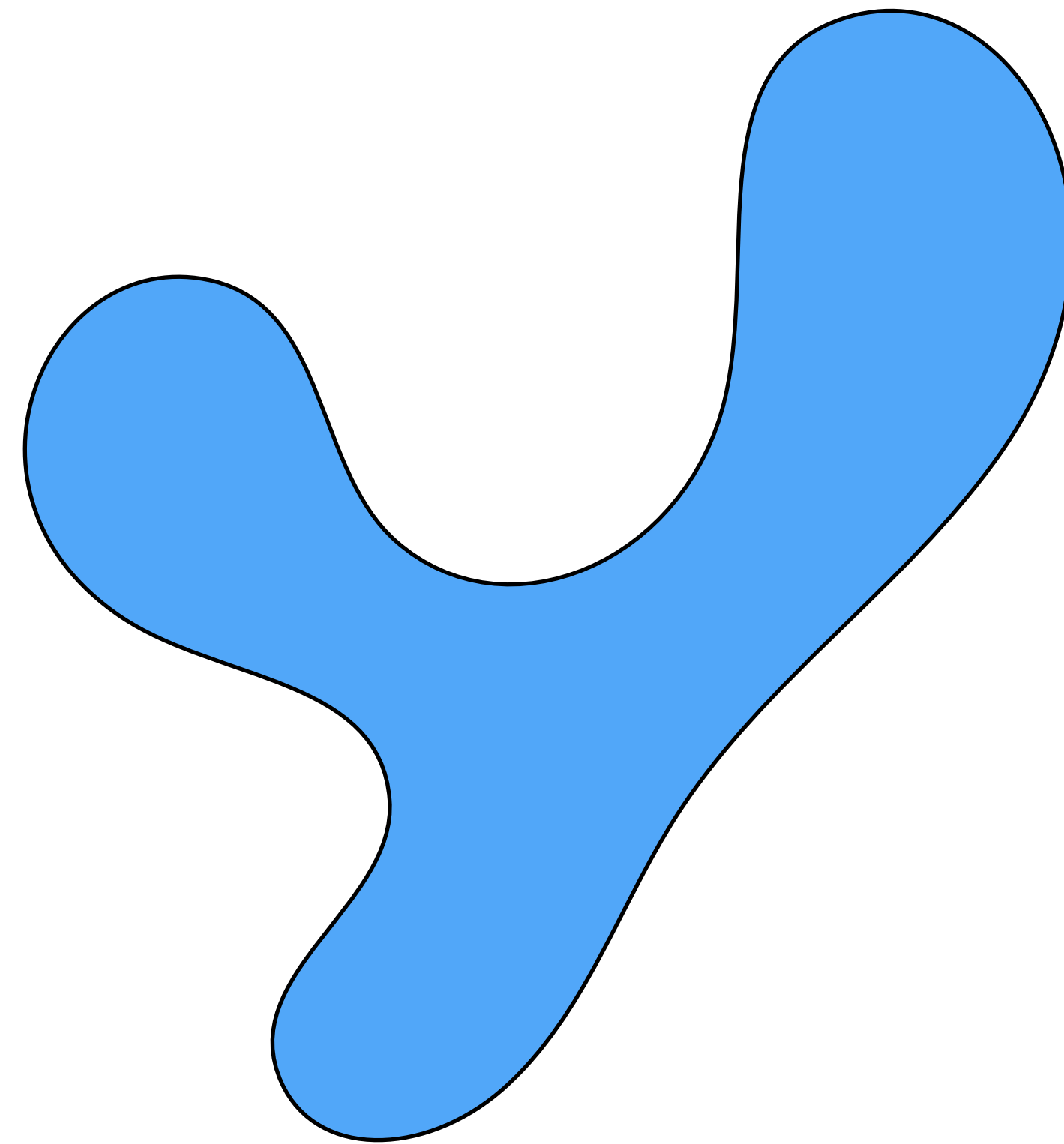
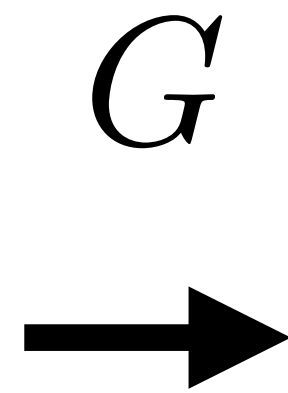
Recall: Neural networks as distribution transformers

Source distribution

Target distribution

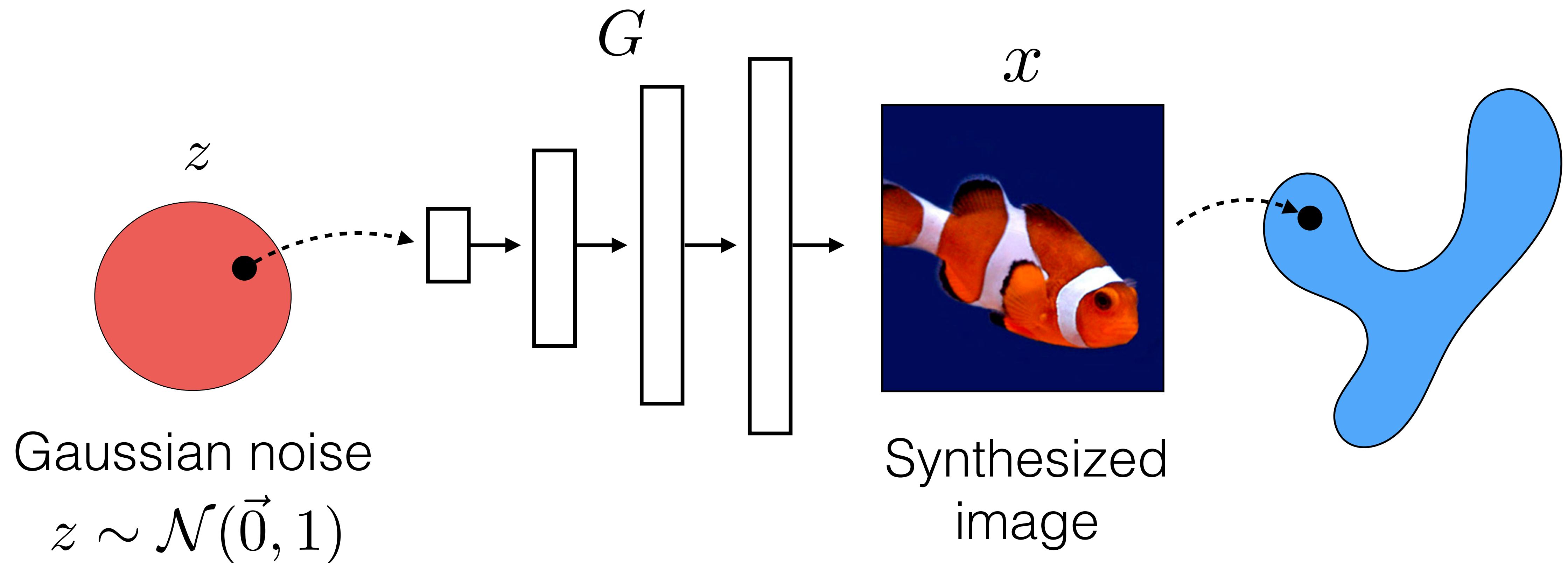


$p(z)$

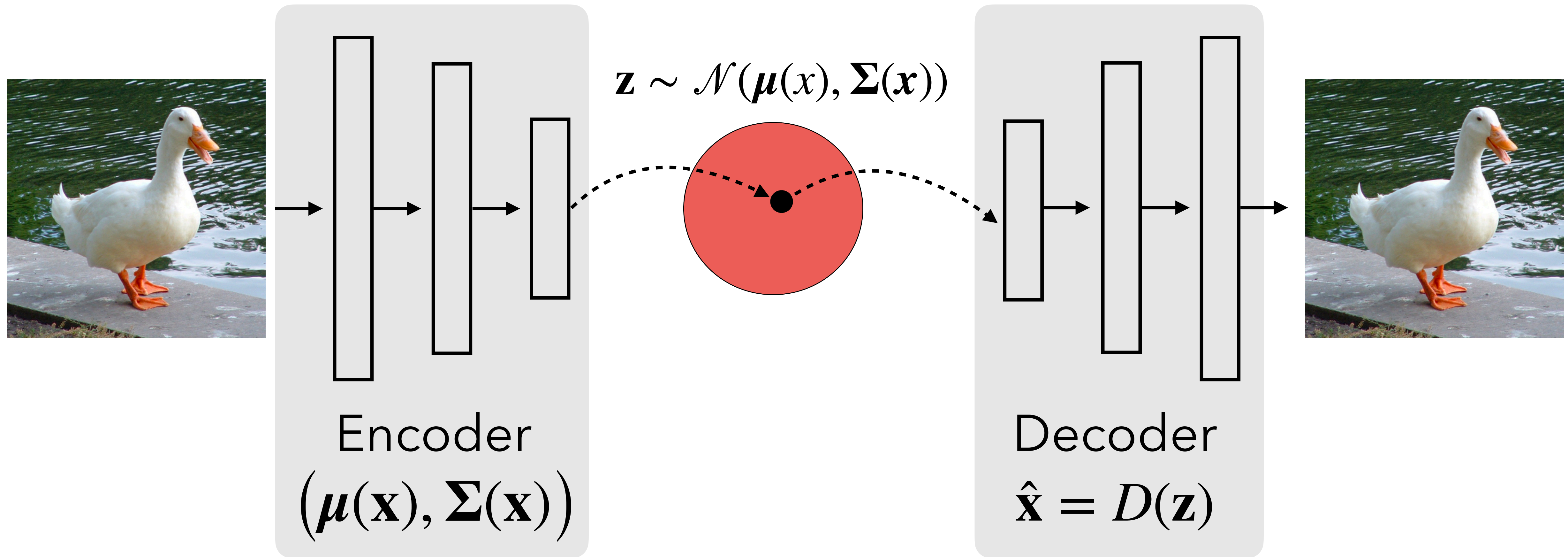


$p(x)$

Recall: Neural networks as distribution transformers

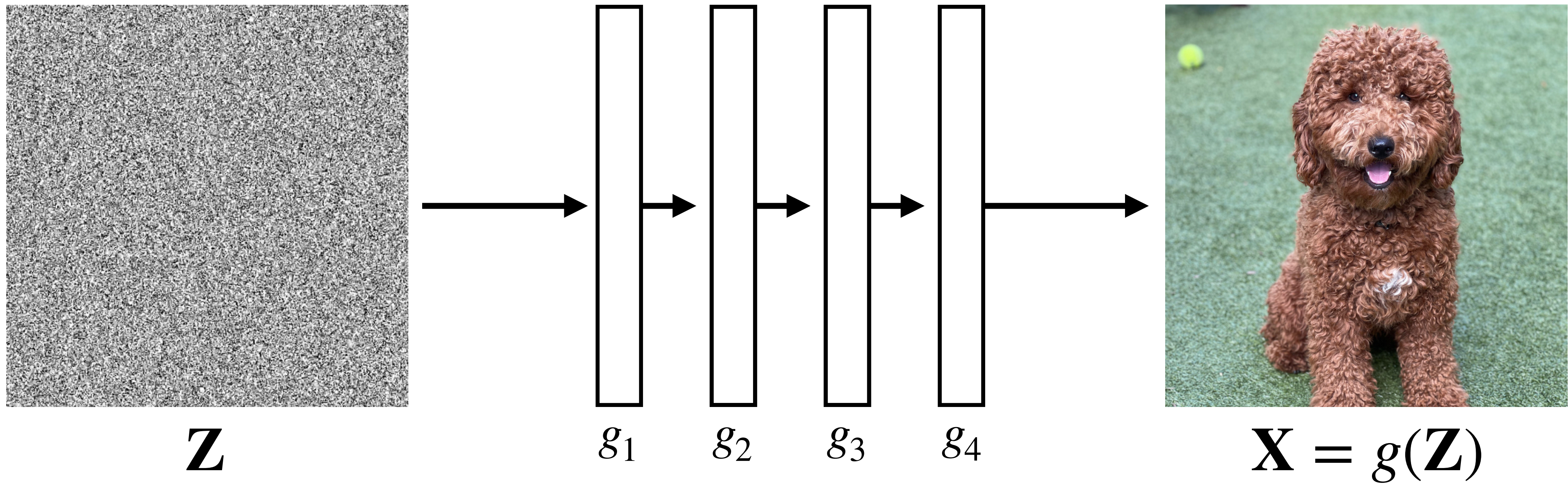


Recall: variational autoencoder



- Encoder net defines a distribution over $\mathbf{z}$: $q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}(x), \boldsymbol{\Sigma}(x))$.
- Decoder net defines a distribution over $\mathbf{x}$: $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\hat{\mathbf{X}}, I)$.

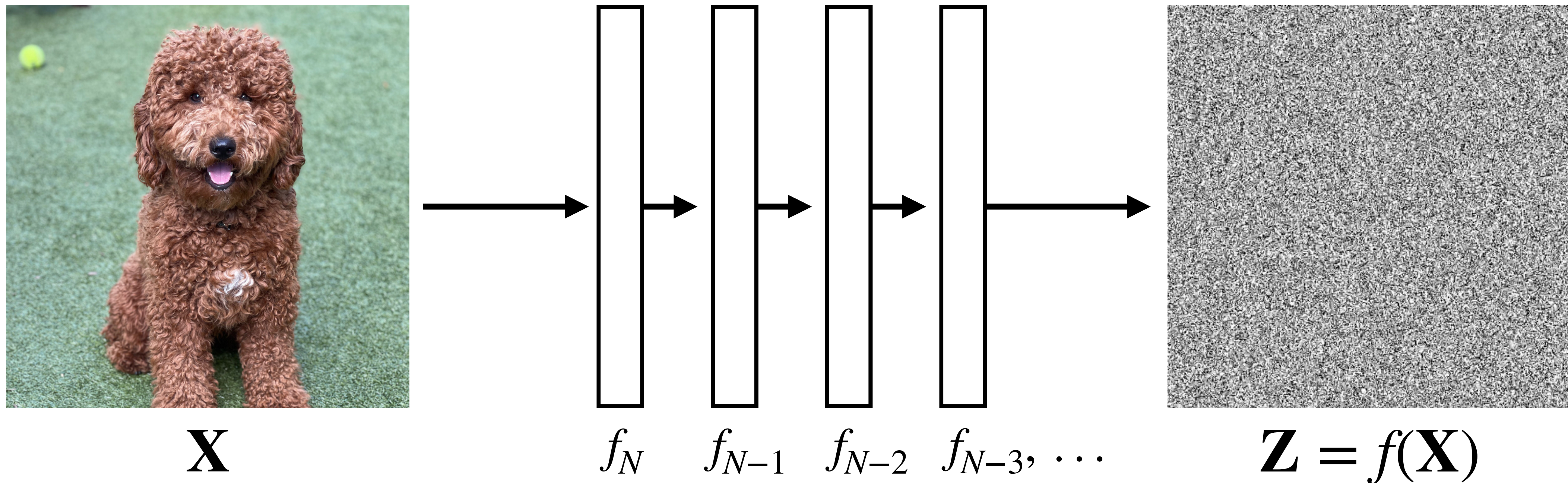
Recall: Normalizing flow model



$$g_N \circ g_{N-1} \circ \dots \circ g_1(\mathbf{z})$$

Recall: Normalizing flow model

For estimating the density and learning:

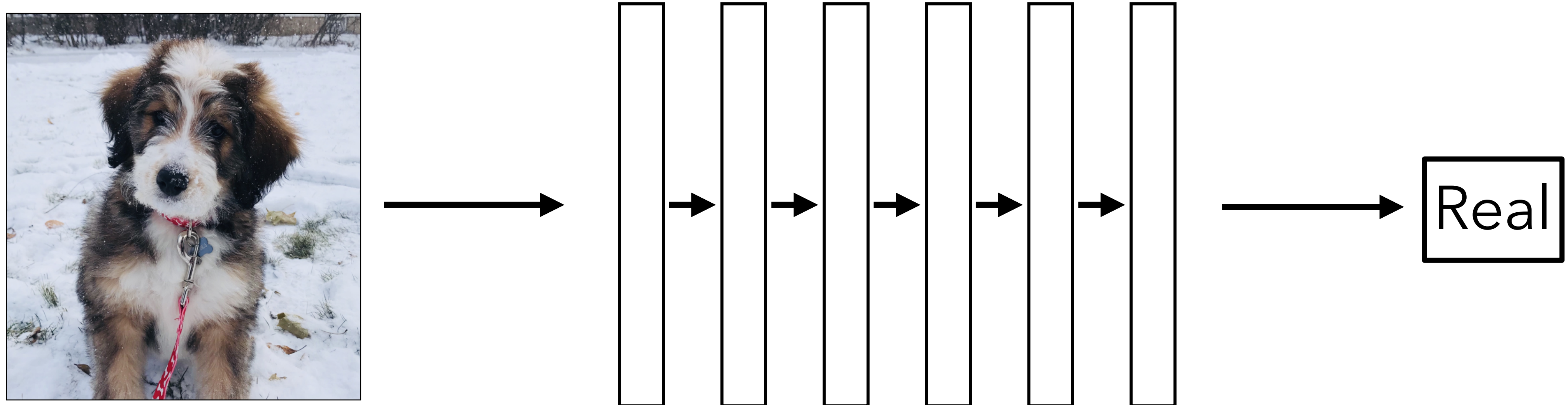


$$\log p_{\theta}(\mathbf{X}) = \sum_{i=1}^M \log p_{\mathbf{Z}}(f(\mathbf{x}_i)) + \log \left| \det(\mathbf{D}f(\mathbf{x}_i)) \right| \quad \text{where } \mathbf{x}_i \text{ is a training example.}$$

Exact likelihood estimation (unlike VAE, which uses lower bound).

Classifying real vs. fake

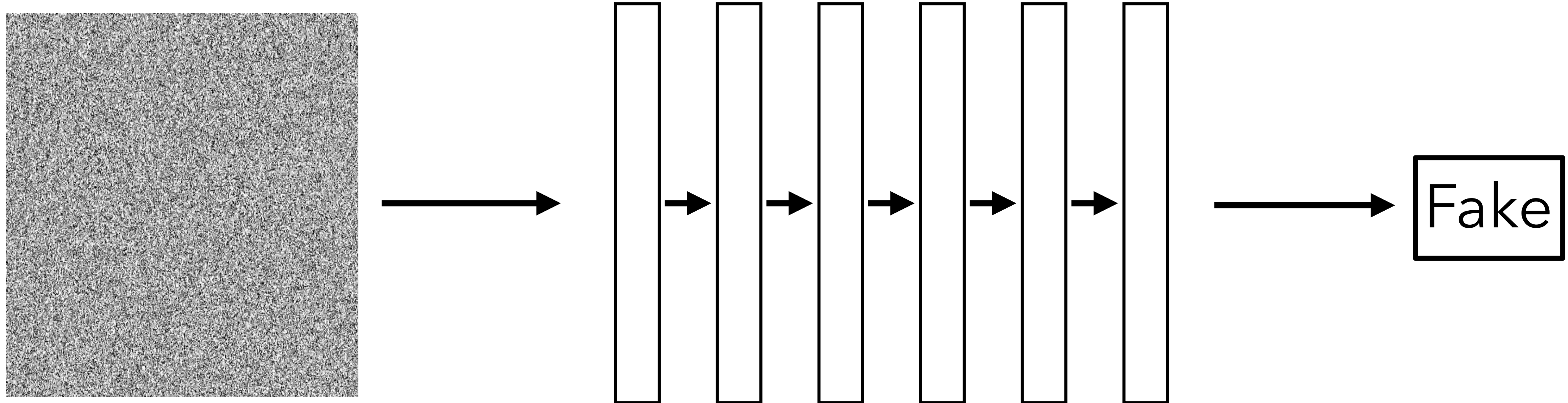
We can successfully *classify* real vs. fake.

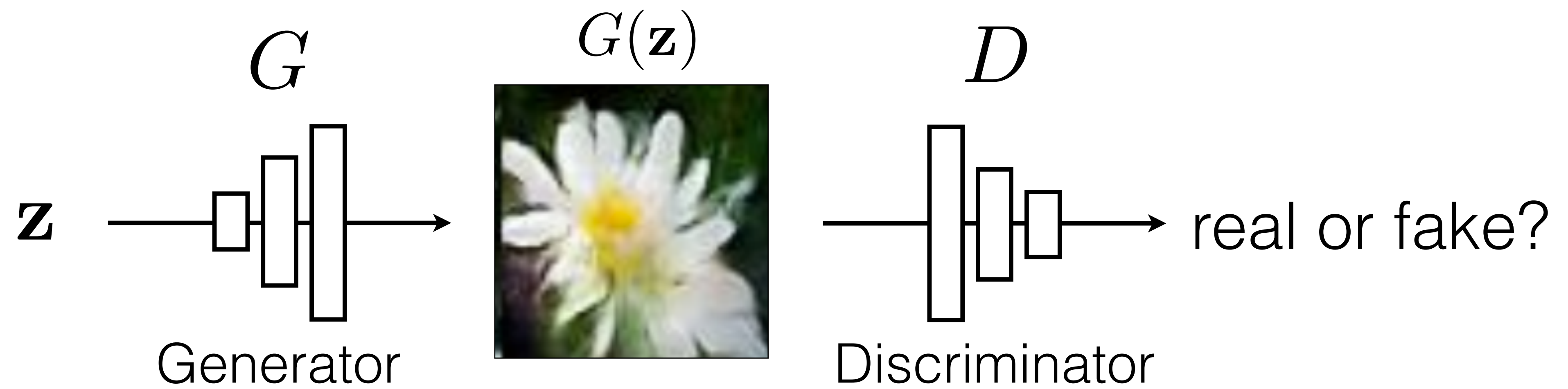


Classifying real vs. fake

We can successfully *classify* real vs. fake.

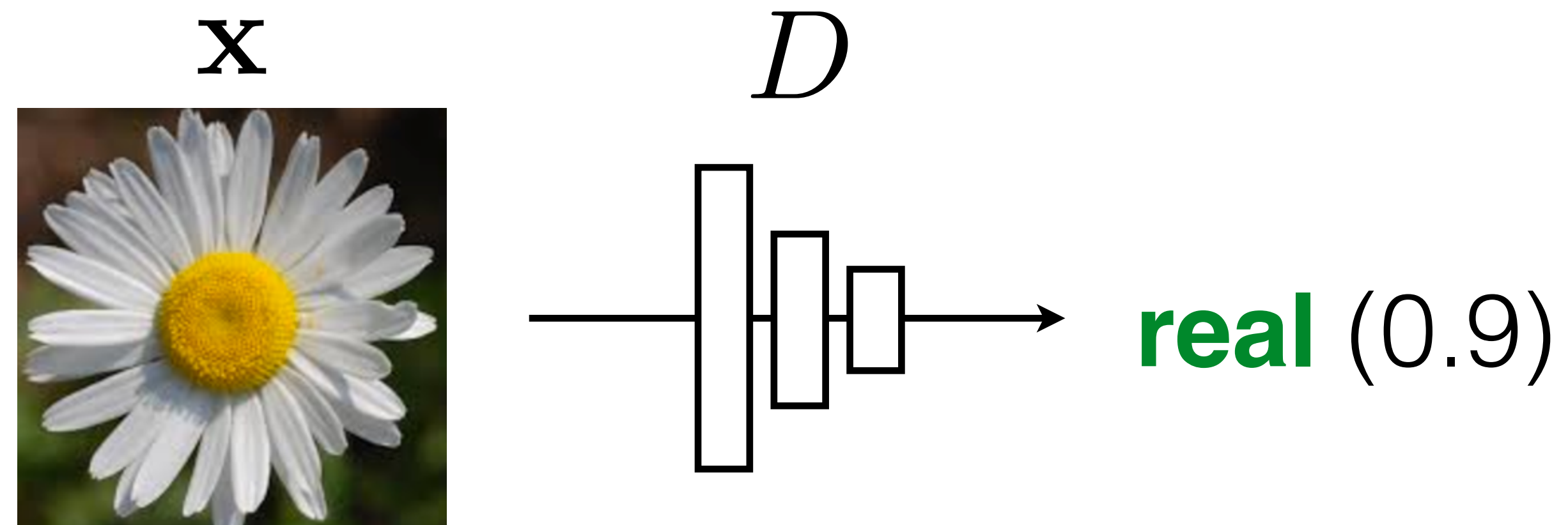
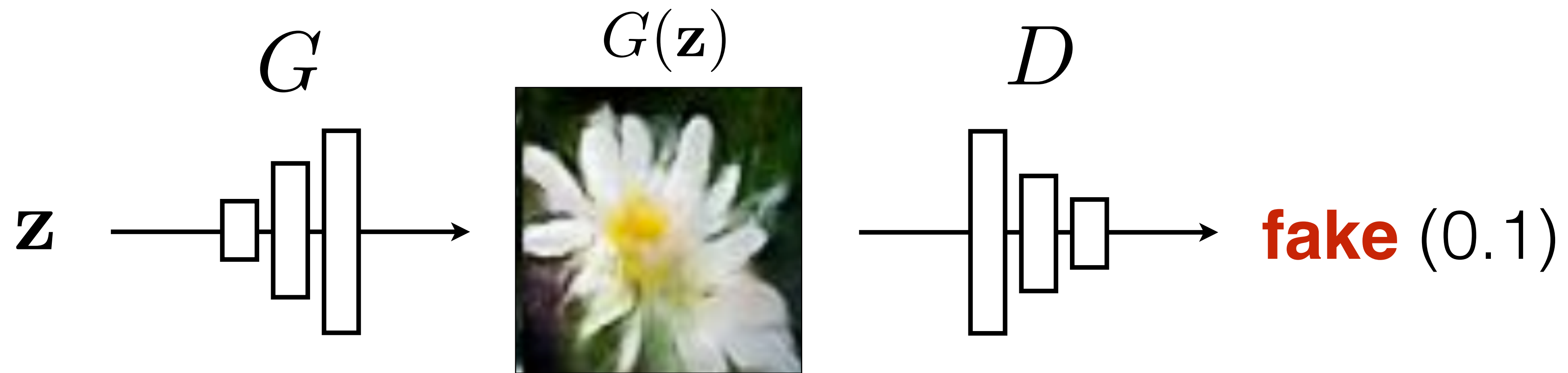
Can we build a generative model from this idea?



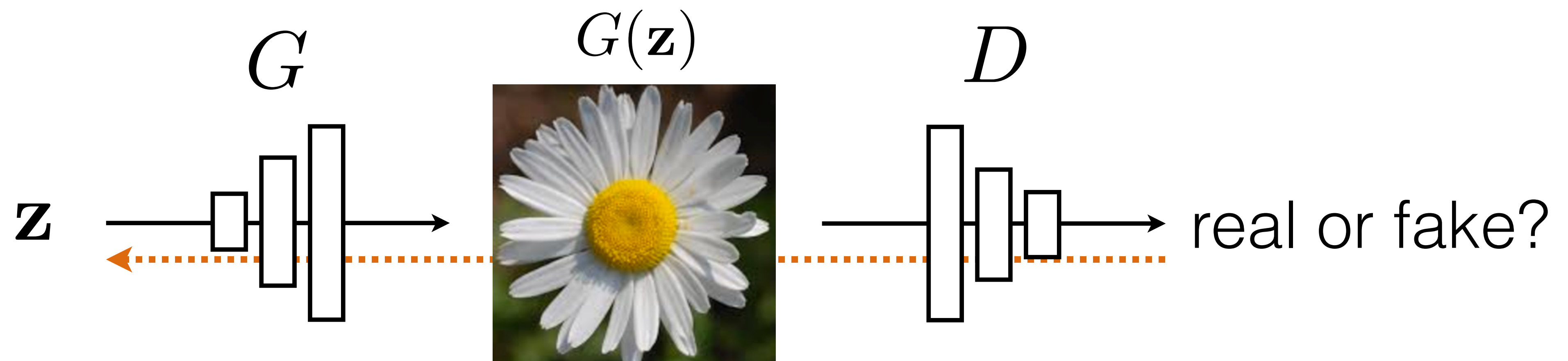


G tries to synthesize fake images that fool **D**

D tries to identify the fakes

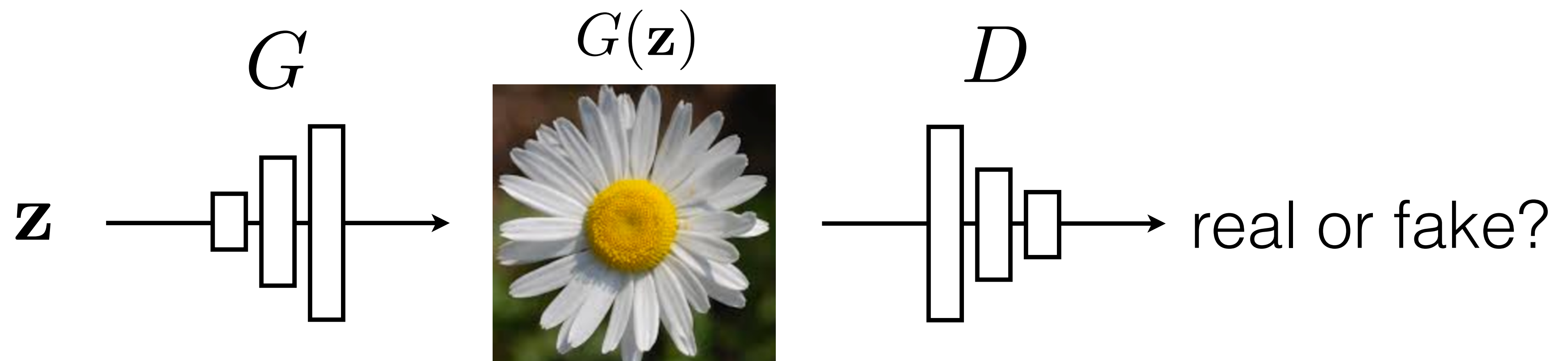


$$\arg \max_D \mathbb{E}_{\mathbf{z}, \mathbf{x}} \left[\log D(G(\mathbf{z})) + \log (1 - D(\mathbf{x})) \right]$$



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

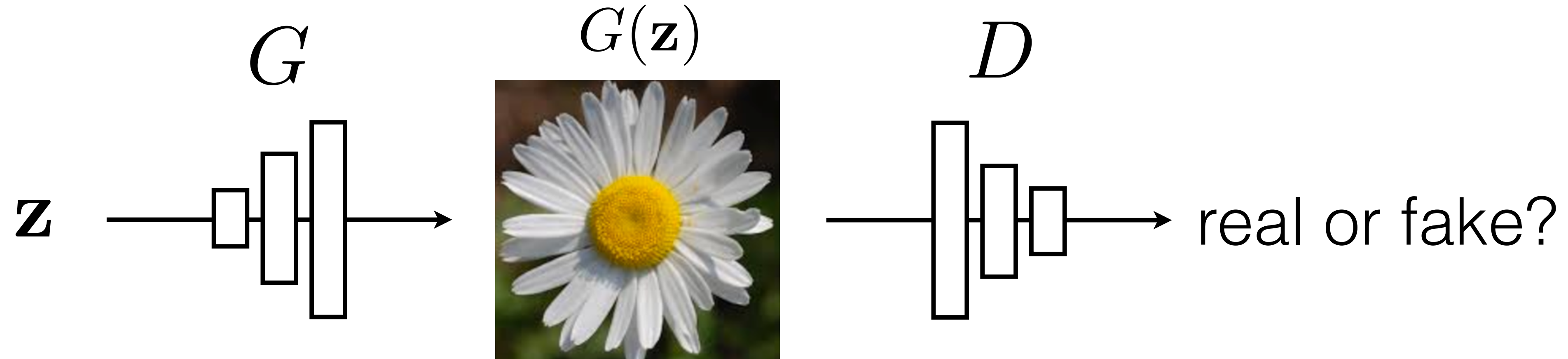
$$\arg \min_{G} \mathbb{E}_{\mathbf{z}, \mathbf{x}} \left[\log D(G(\mathbf{z})) + \log (1 - D(\mathbf{x})) \right]$$



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

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

Training

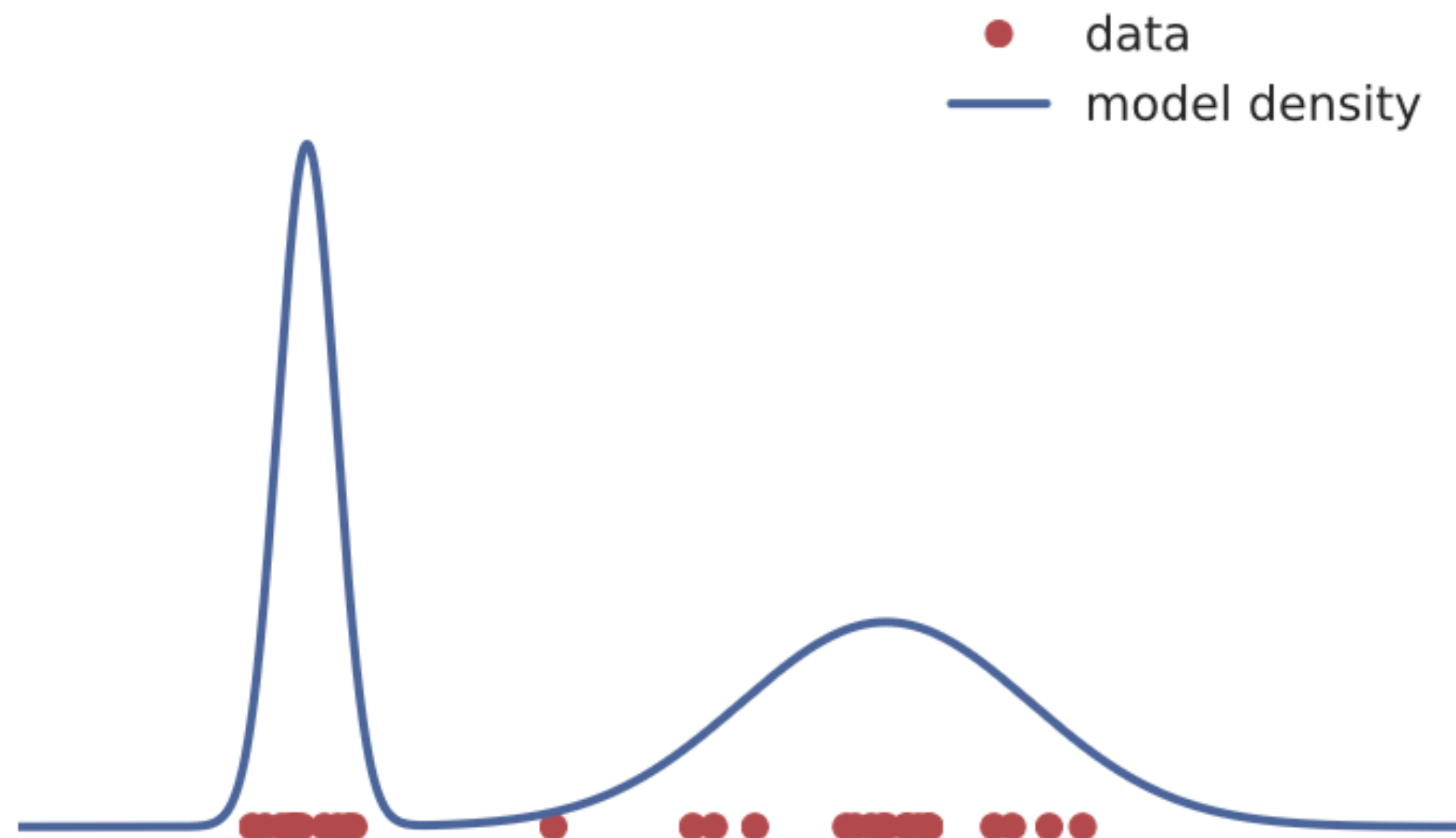


G tries to synthesize fake images that fool **D**

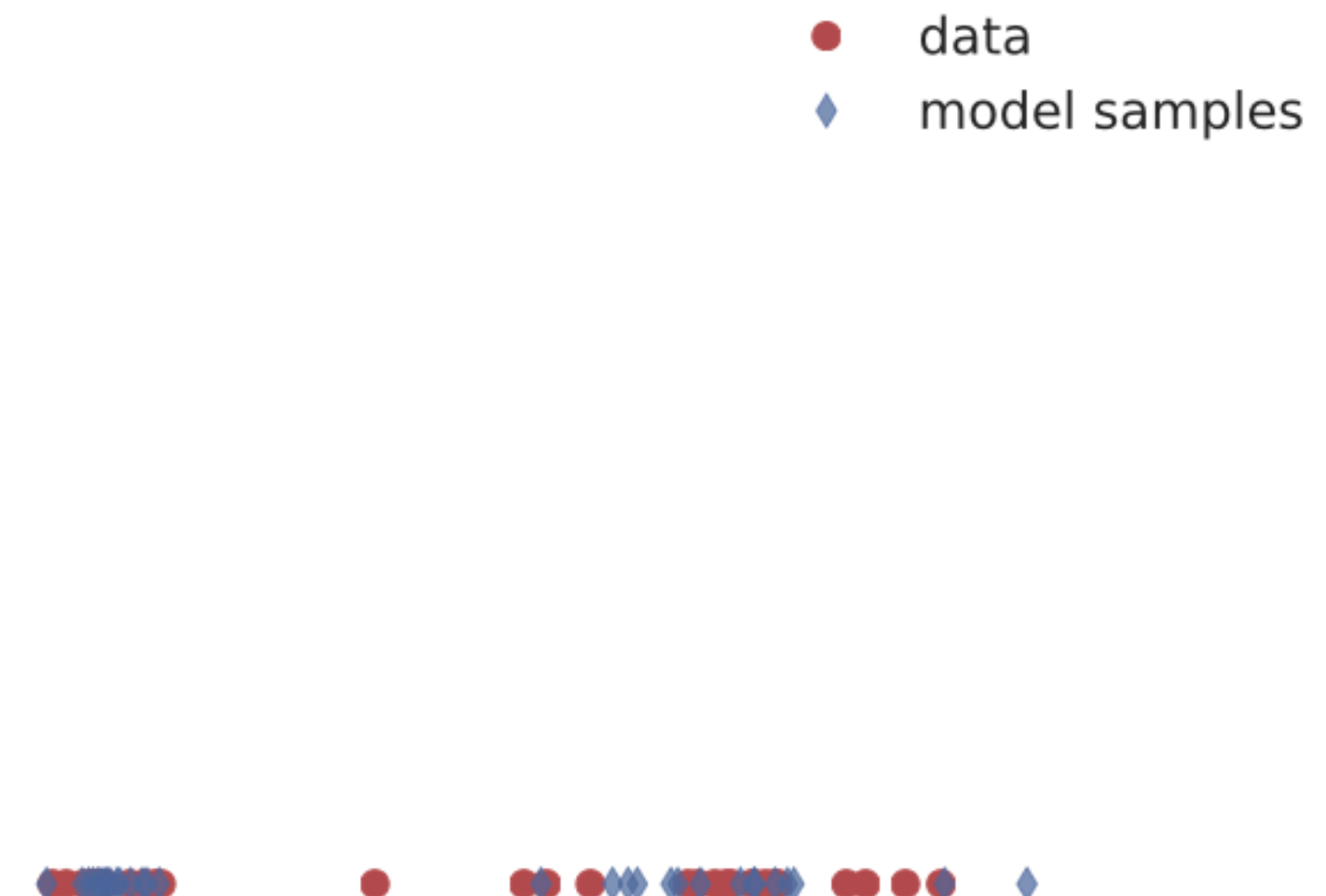
D tries to identify the fakes

- Training: iterate between training **D** and **G** with backprop.
- Global optimum when **G** reproduces data distribution (see book)

GANs are *implicit* generative models



Prescribed generative model
(a density that you can sample from)



Implicit generative model
(only a sampling procedure)

Training pseudocode

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(\mathbf{x}^{(i)}) + \log \left(1 - D(G(\mathbf{z}^{(i)})) \right) \right].$$

end for

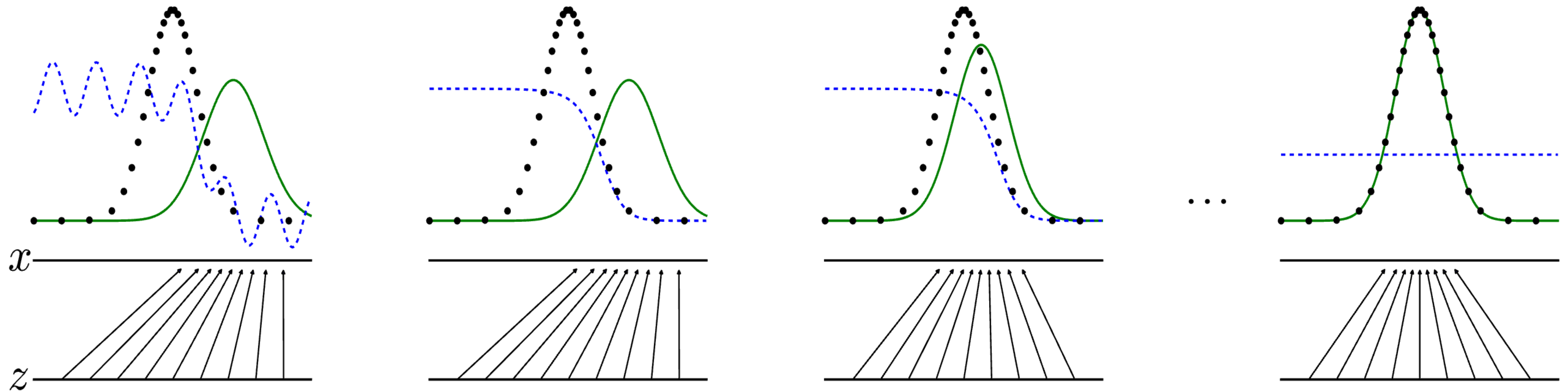
- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log \left(1 - D(G(\mathbf{z}^{(i)})) \right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Toy example



Generator (**green**) and discriminator (**blue**) as training progresses.

Analysis

What is the optimal discriminator?

$$D^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}$$

where p_{data} is the data distribution
and p_g is the generator's distribution (i.e.,
the distribution of $G(\mathbf{z})$)

Given a generator G , we are maximizing:

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} + \int_{\mathbf{z}} p_z(\mathbf{z}) \log(1 - D(G(\mathbf{z}))) d\mathbf{z} \\ &= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \end{aligned}$$

How do you maximize this? For some (a, b) , consider:

$$f(y) = a \log y + b \log(1 - y) \quad \text{set} \quad \frac{df}{dy} = \frac{a}{y} - b \frac{1}{1 - y} = 0$$

$$\implies \frac{1 - y}{y} = \frac{b}{a} \implies \frac{1}{y} = 1 + \frac{b}{a} \implies y = \frac{a}{a + b}$$

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

Proof sketch:

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

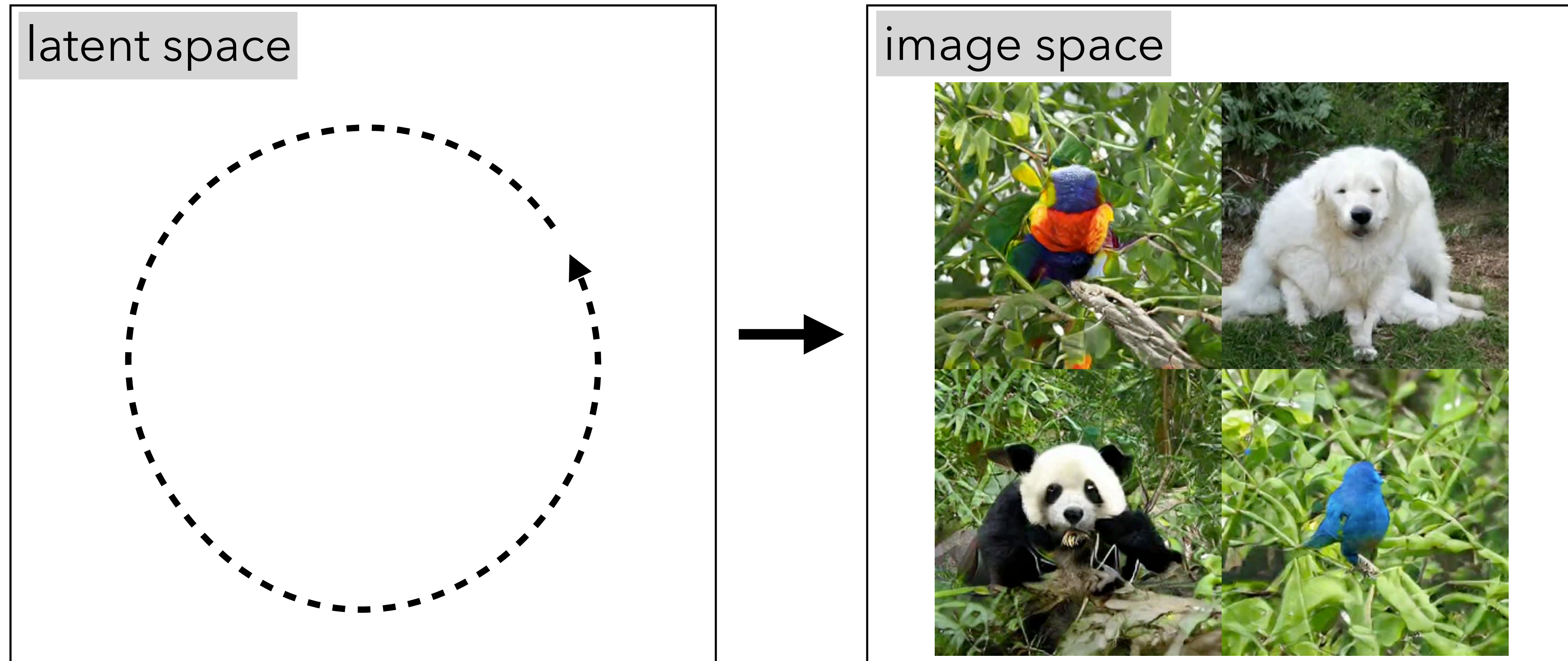
$$C(G) = -\log(4) + KL \left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2} \right\| \right) + KL \left(p_g \left\| \frac{p_{\text{data}} + p_g}{2} \right\| \right)$$

$$C(G) = -\log(4) + 2 \cdot \underbrace{JSD(p_{\text{data}} \| p_g)}_{\geq 0} \quad \text{Jensen-Shannon divergence}$$

$$\geq 0, \quad 0 \iff p_g = p_{data} \quad \square$$

Exploring learned GAN models

Generative models organize the manifold of natural images



Samples from BigGAN

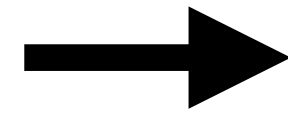
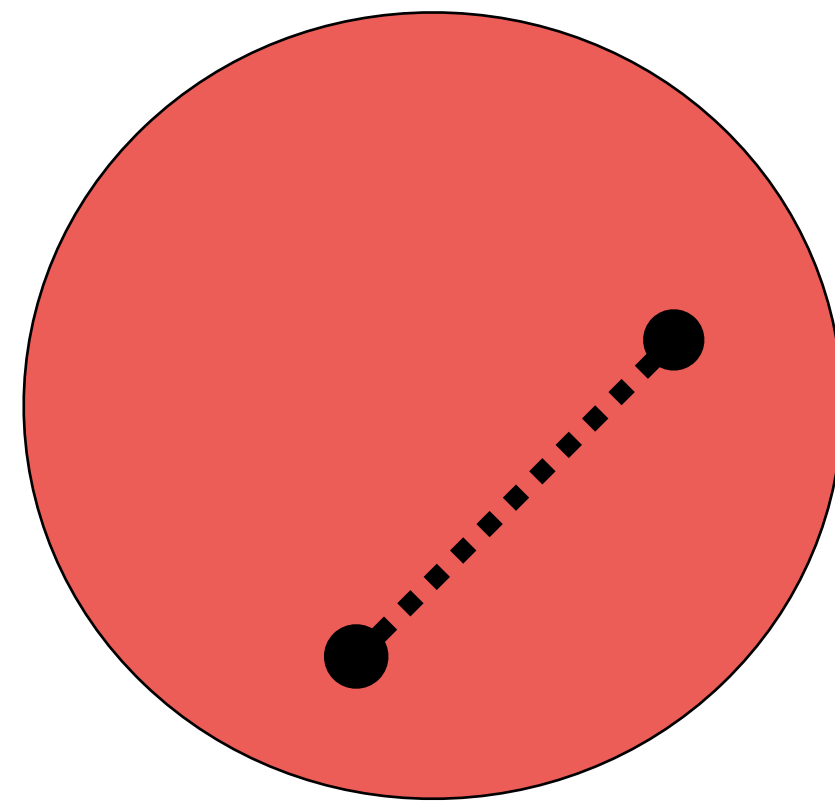
[Brock et al. 2018]



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

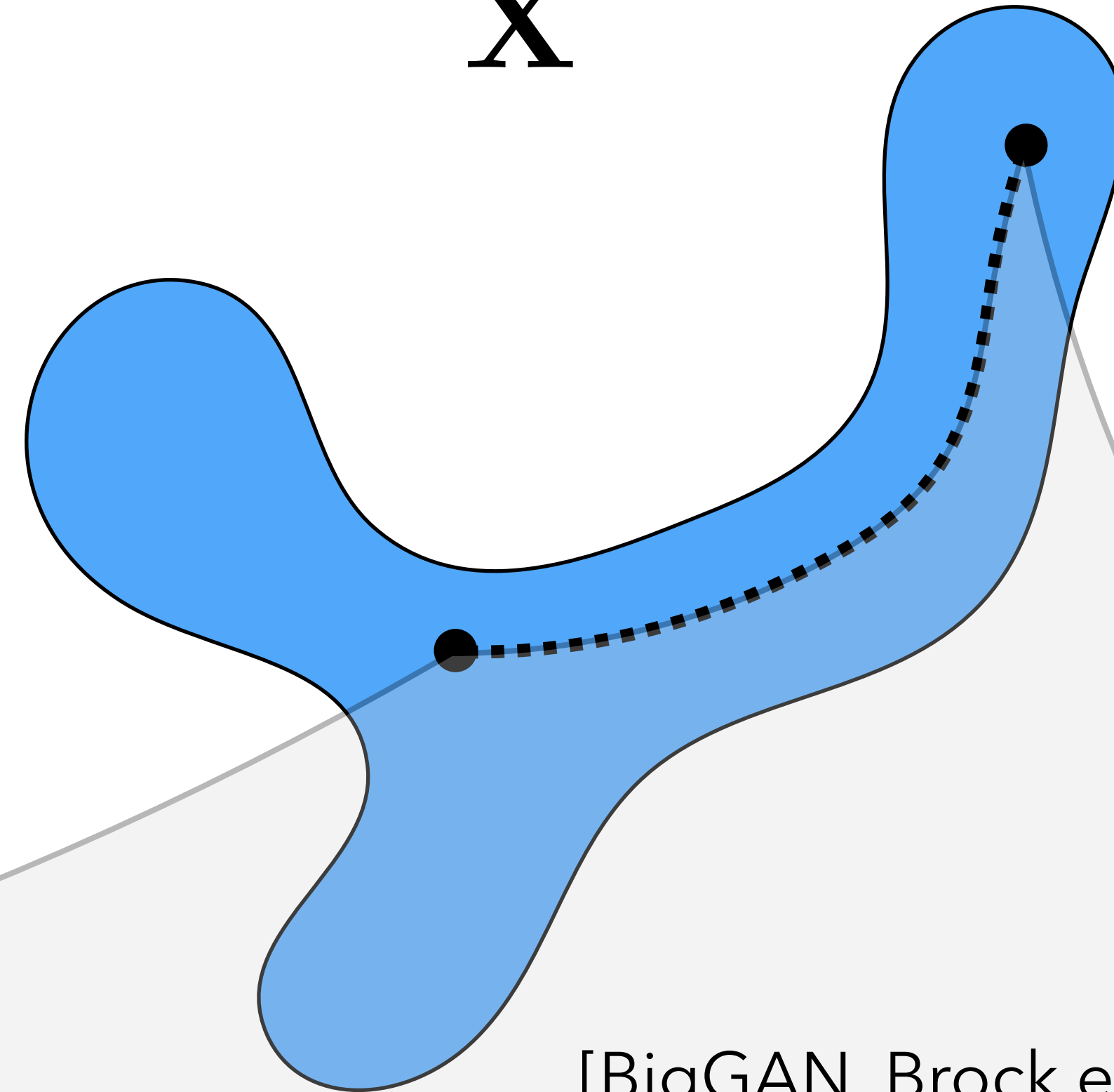
Latent space
(Gaussian)

$\mathbf{Z}$



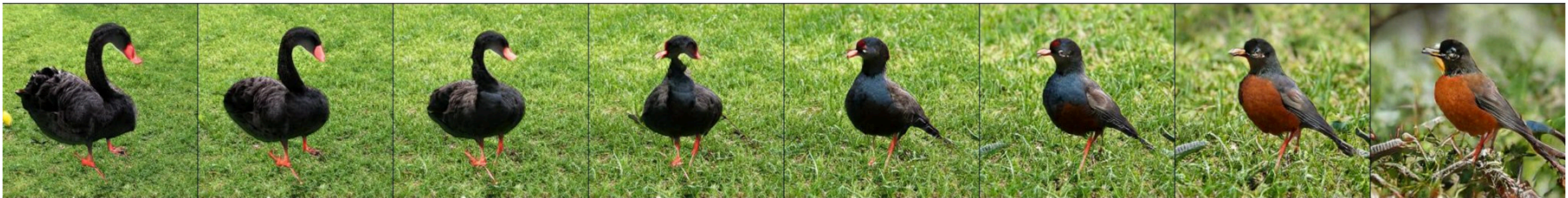
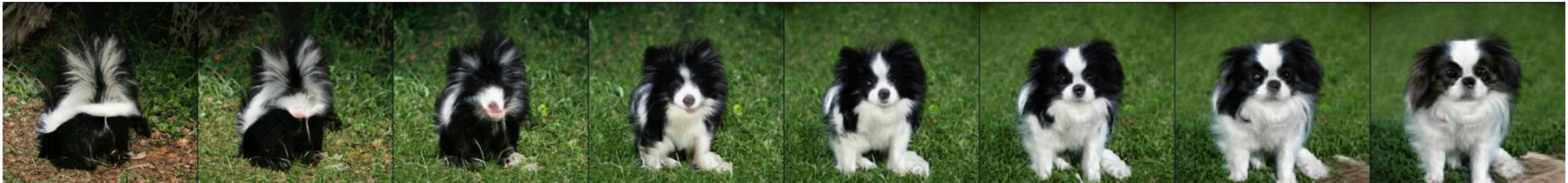
Data space
(Natural image manifold)

$\mathbf{X}$



[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]



StyleGAN3 (Ours)

[Karras et al., "Alias-Free Generative Adversarial Networks", 2021]

GAN variants

Conditional GANs

Unconditional GAN:

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

Conditional GAN:

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

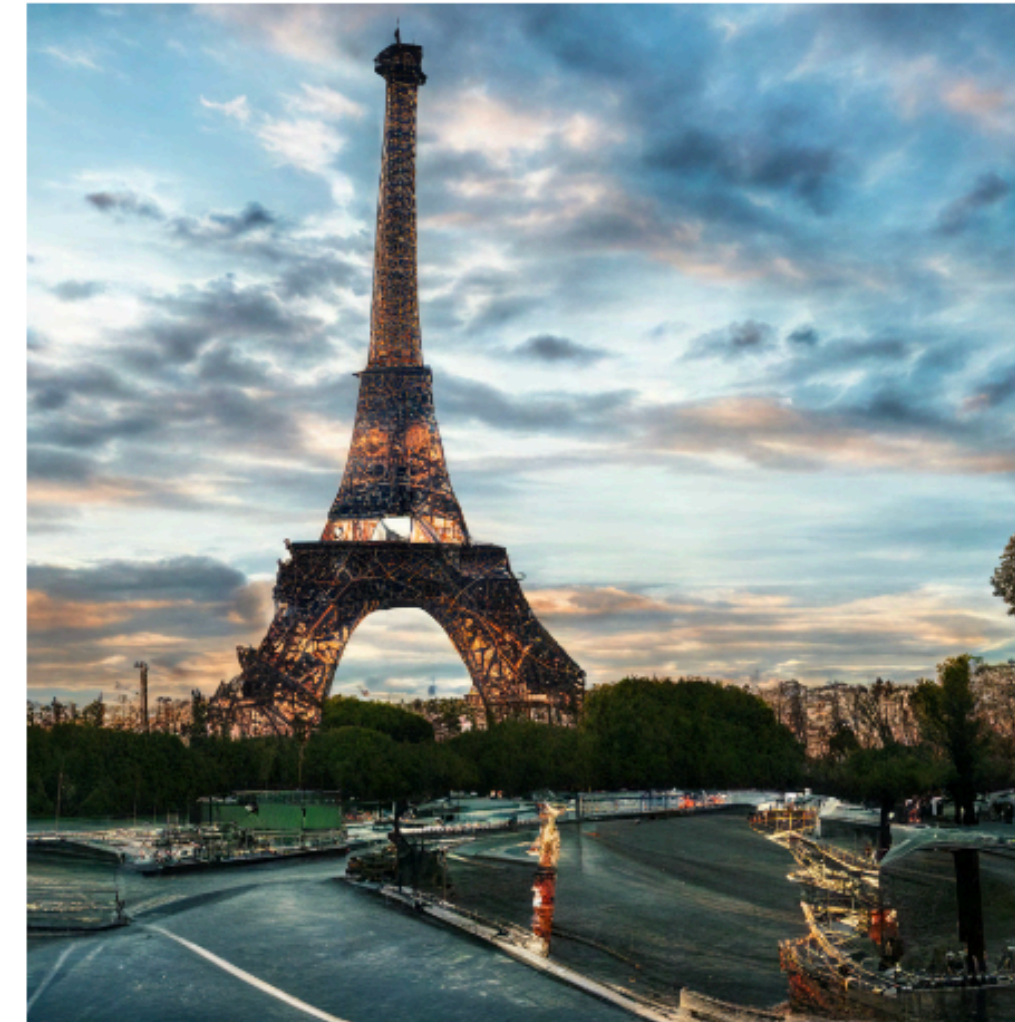
Text-to-image GANs



A living room with a fireplace at a wood cabin. Interior design.



a blue Porsche 356 parked in front of a yellow brick wall.



Eiffel Tower, landscape photography



A painting of a majestic royal tall ship in Age of Discovery.



Isometric underwater Atlantis city with a Greek temple in a bubble.



A hot air balloon in shape of a heart. Grand Canyon



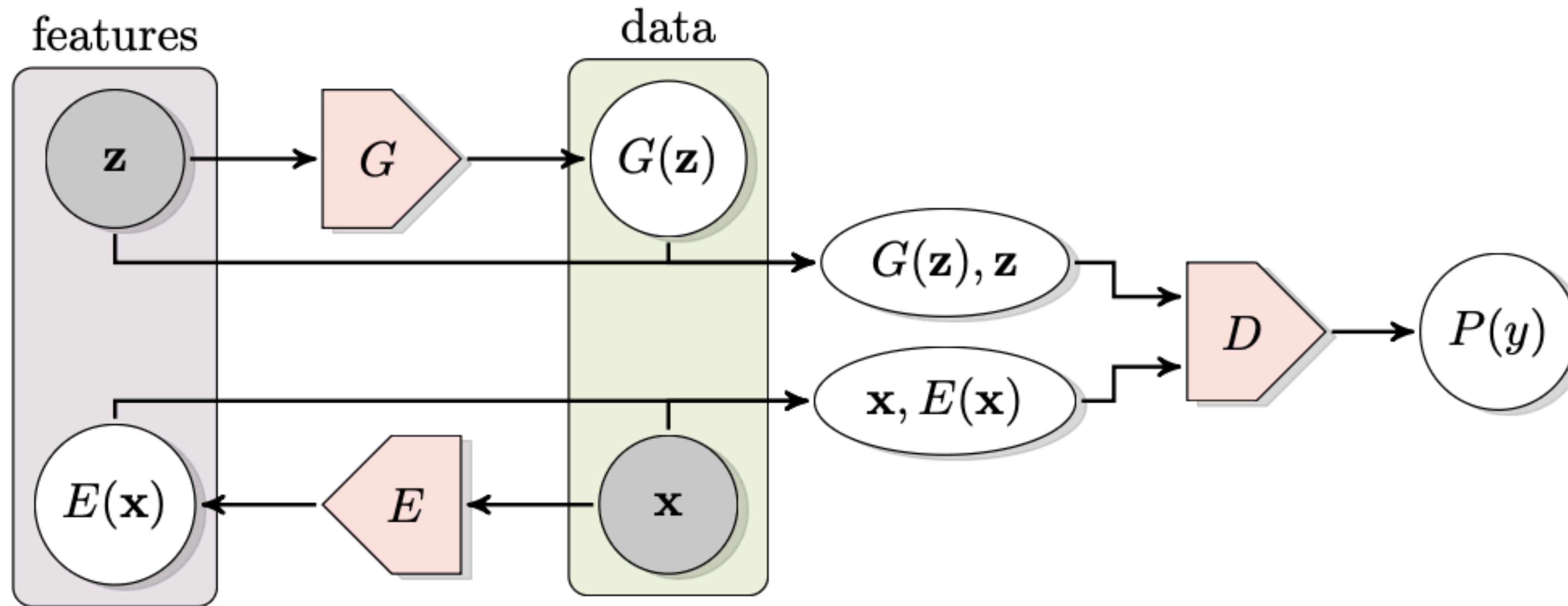
low poly bunny with cute eyes



A cube made of denim on a wooden table

[Kang et al., "Scaling up GANs for Text-to-Image Synthesis", 2023]

Encoder-decoder models



Encoder-decoder models

Real image



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

Challenges of training GANs

Saturation

- Generator loss function: $L_G = \mathbb{E}_{\mathbf{z}} [\log(1 - D(G(\mathbf{z})))]$
- What happens when D gets really high accuracy?
- Generator's loss is nearly flat when $D(\mathbf{x})$ approaches 0. Consequently, gradient of loss w.r.t. generator's parameters $\nabla_{\theta_G} L_G$ will be small.

Saturation

One solution: replace $L_G = \mathbb{E}_{\mathbf{z}} [\log(1 - D(G(\mathbf{z})))]$ with $L'_G = \mathbb{E}_{\mathbf{z}} [-\log(D(G(\mathbf{z})))]$

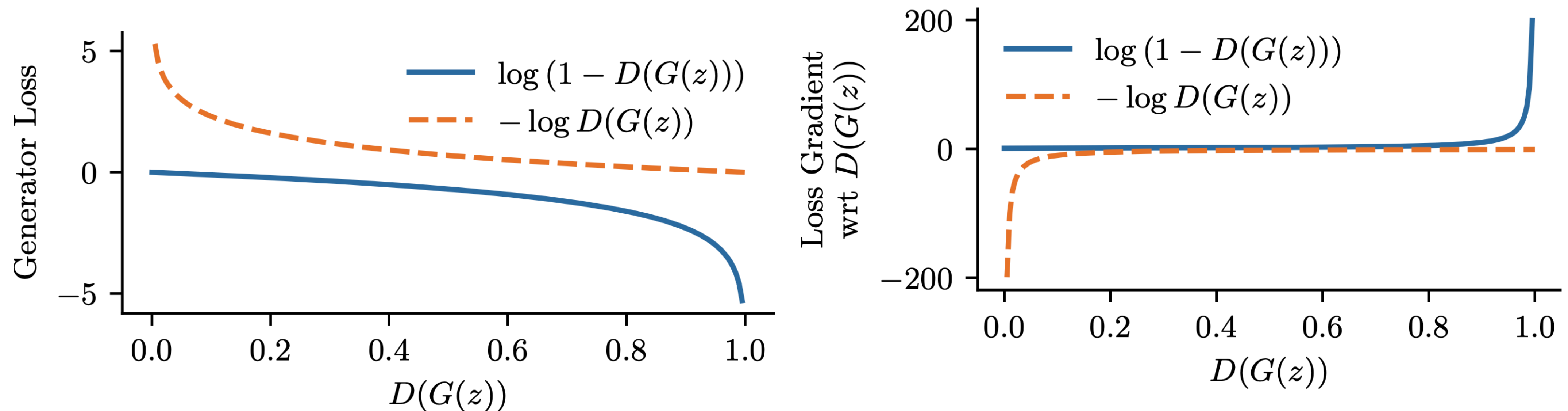


Figure source: [Murphy, PMLAT book]

Lack of diversity



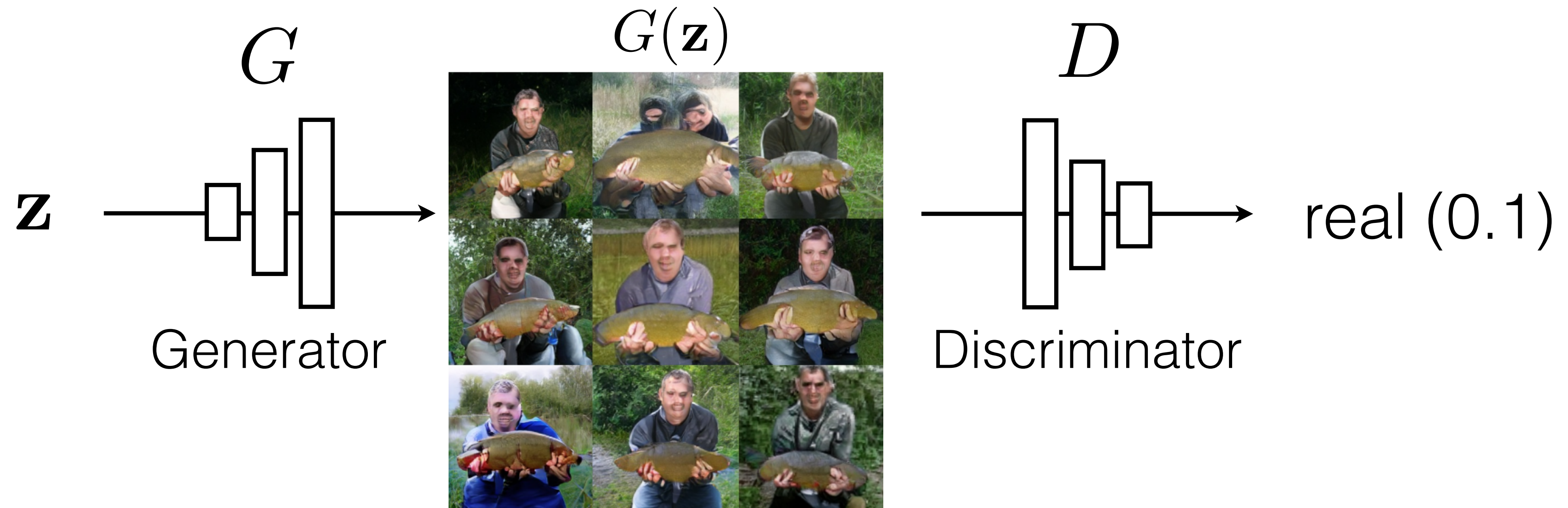
Category: fish

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

Why does this happen?

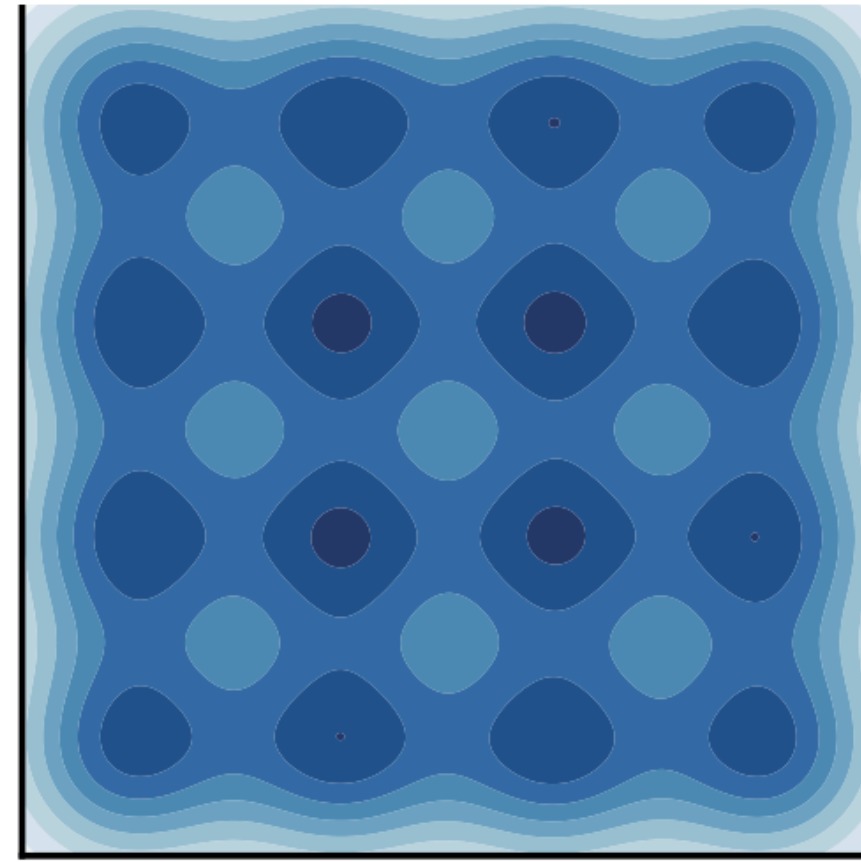
Figure source: [Razavi et al., 2019]

Lack of diversity



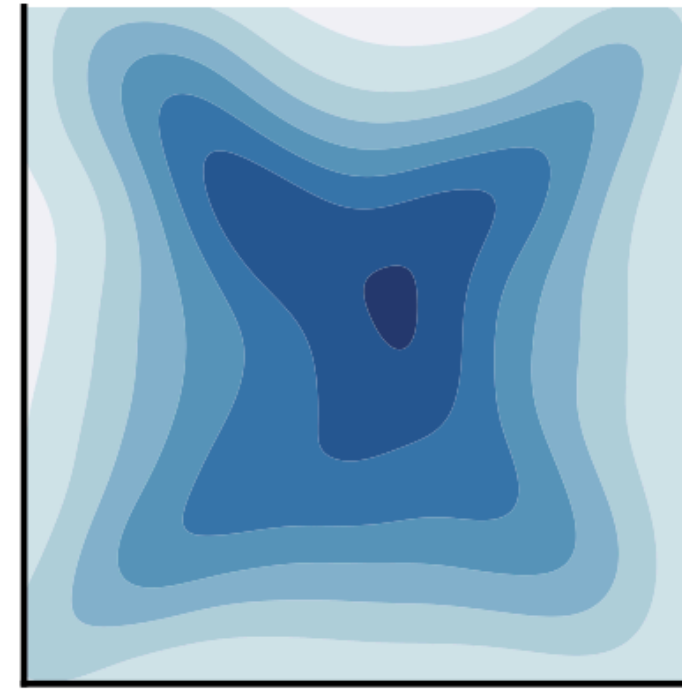
Don't need to cover the full distribution. Only need to fool the discriminator!

Mode collapse and mode hopping



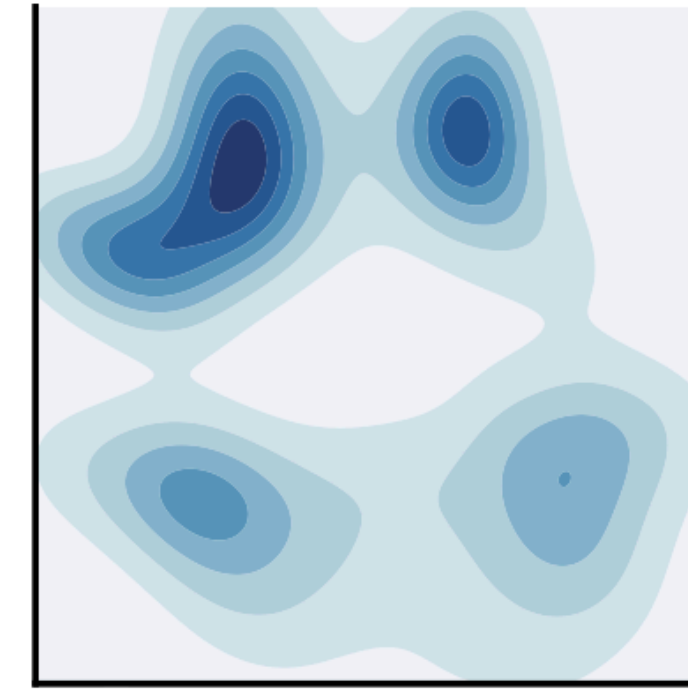
(a) Data

Disc Loss: 1.252
Gen Loss: 0.753



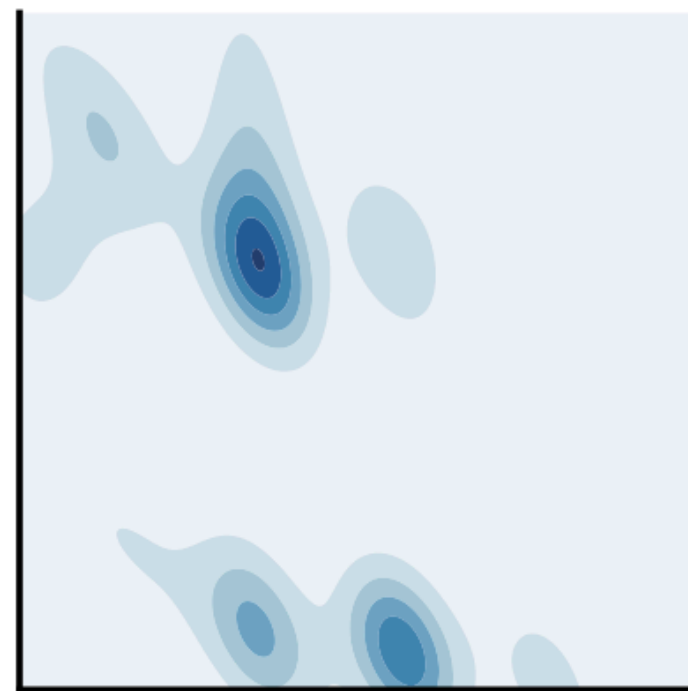
(b) 2000 iterations

Disc Loss: 1.136
Gen Loss: 0.740



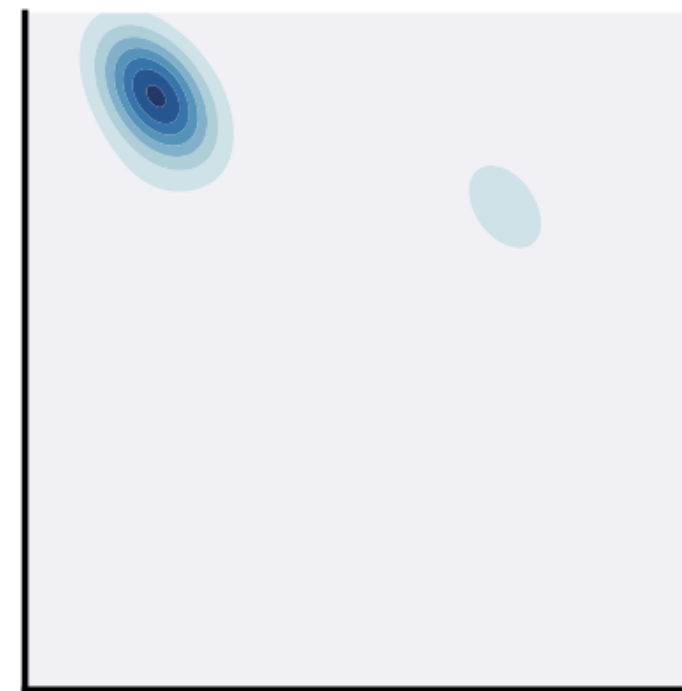
(c) 4000 iterations

Disc Loss: 0.892
Gen Loss: 1.852



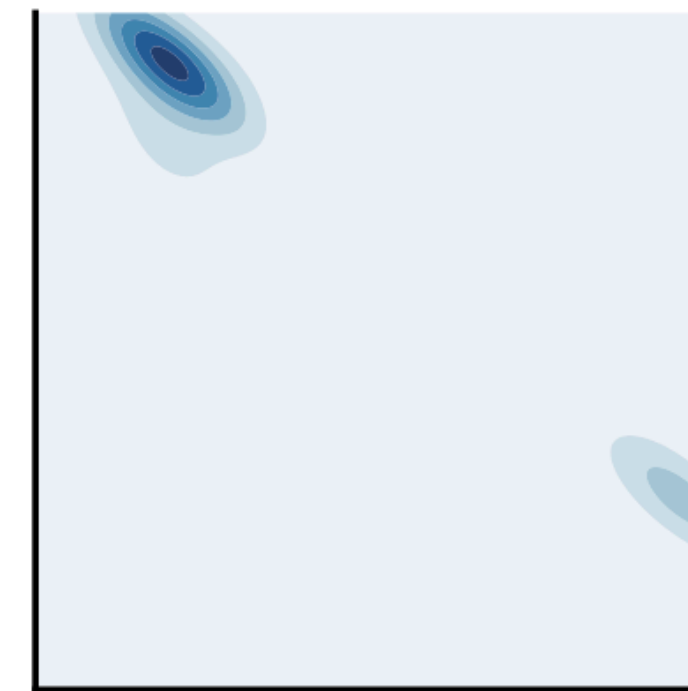
(d) 8000 iterations

Disc Loss: 0.457
Gen Loss: 3.258



(e) 12000 iterations

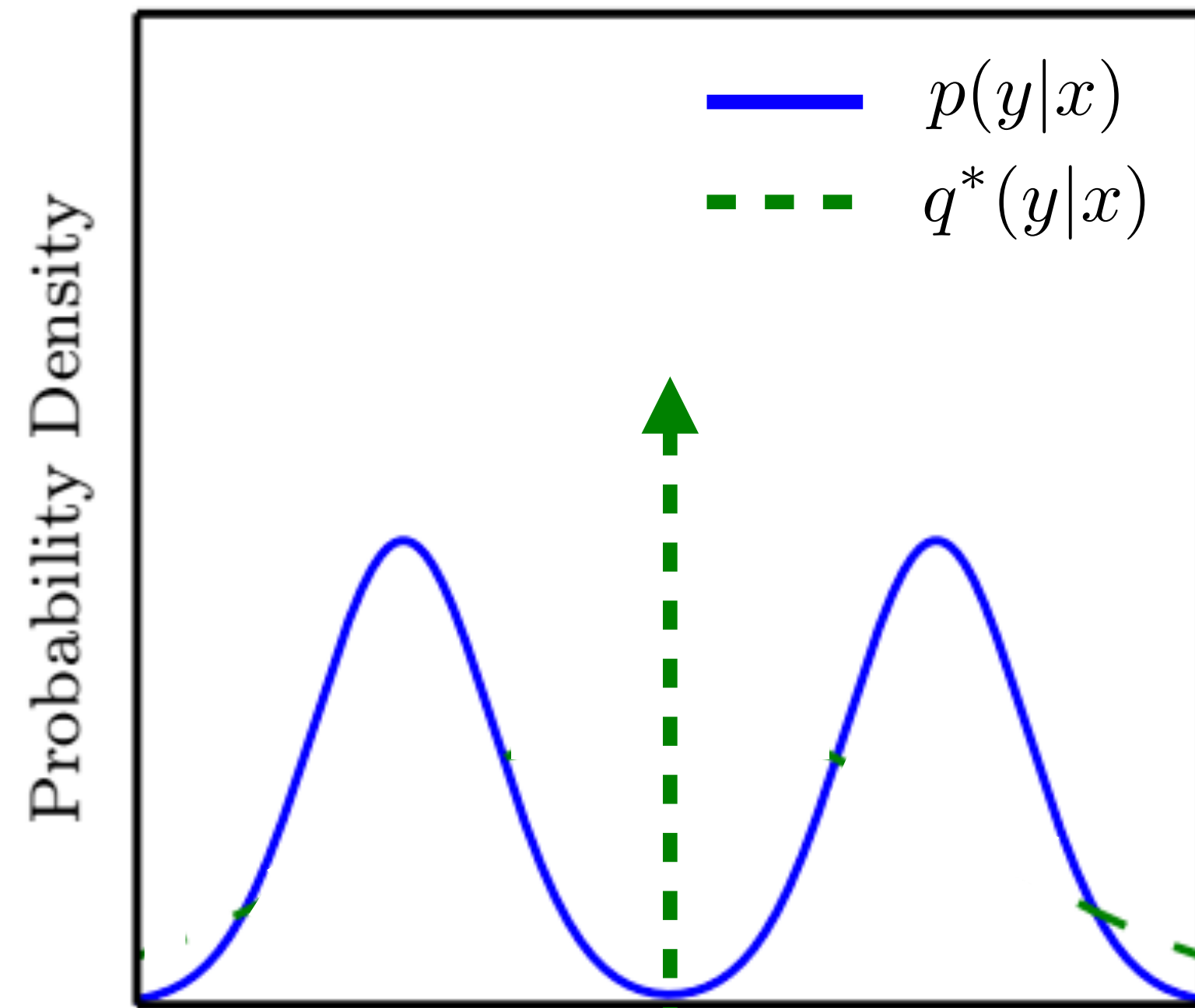
Disc Loss: 0.305
Gen Loss: 3.016



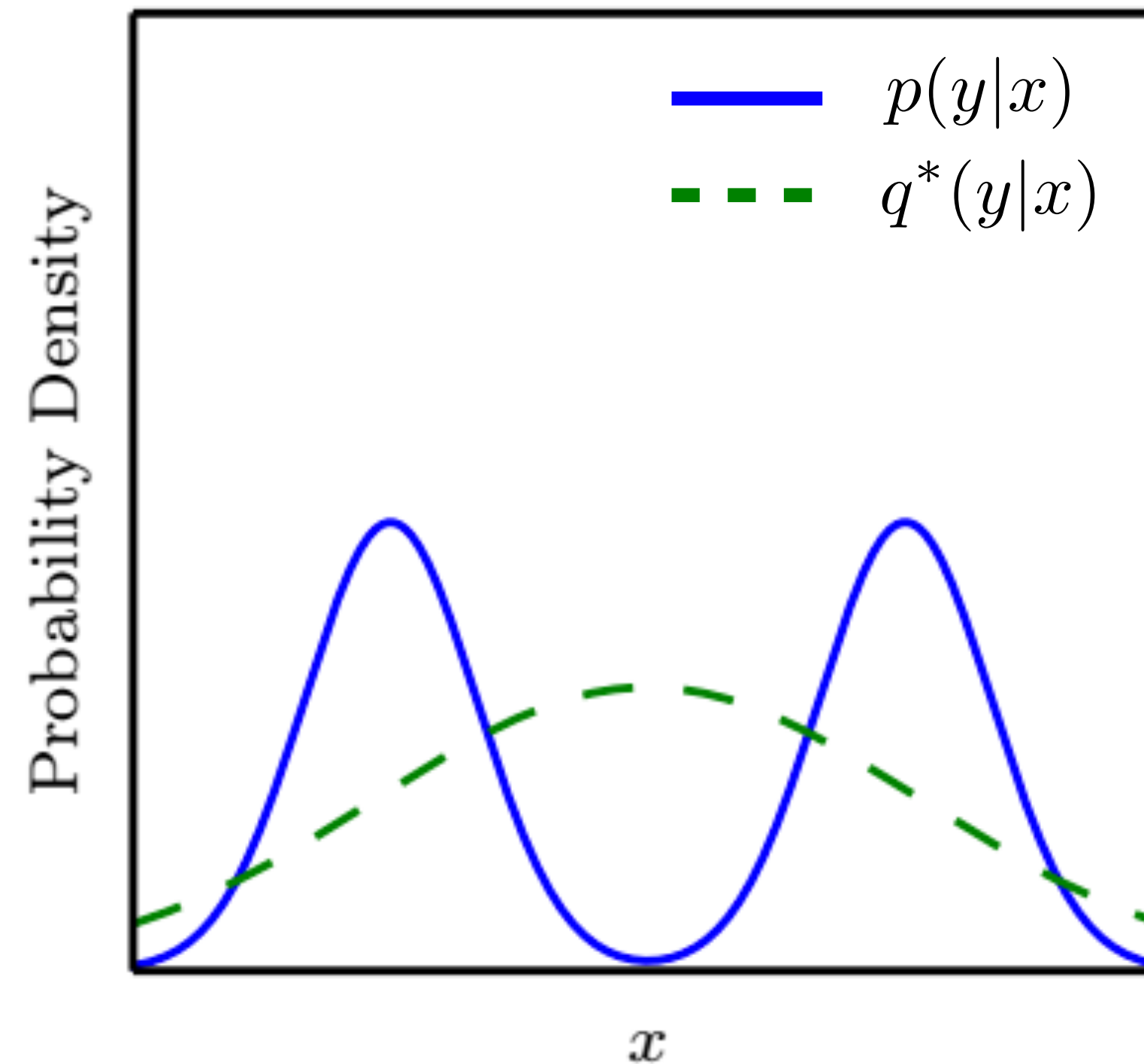
(f) 14000 iterations

Mode seeking property

Point estimate

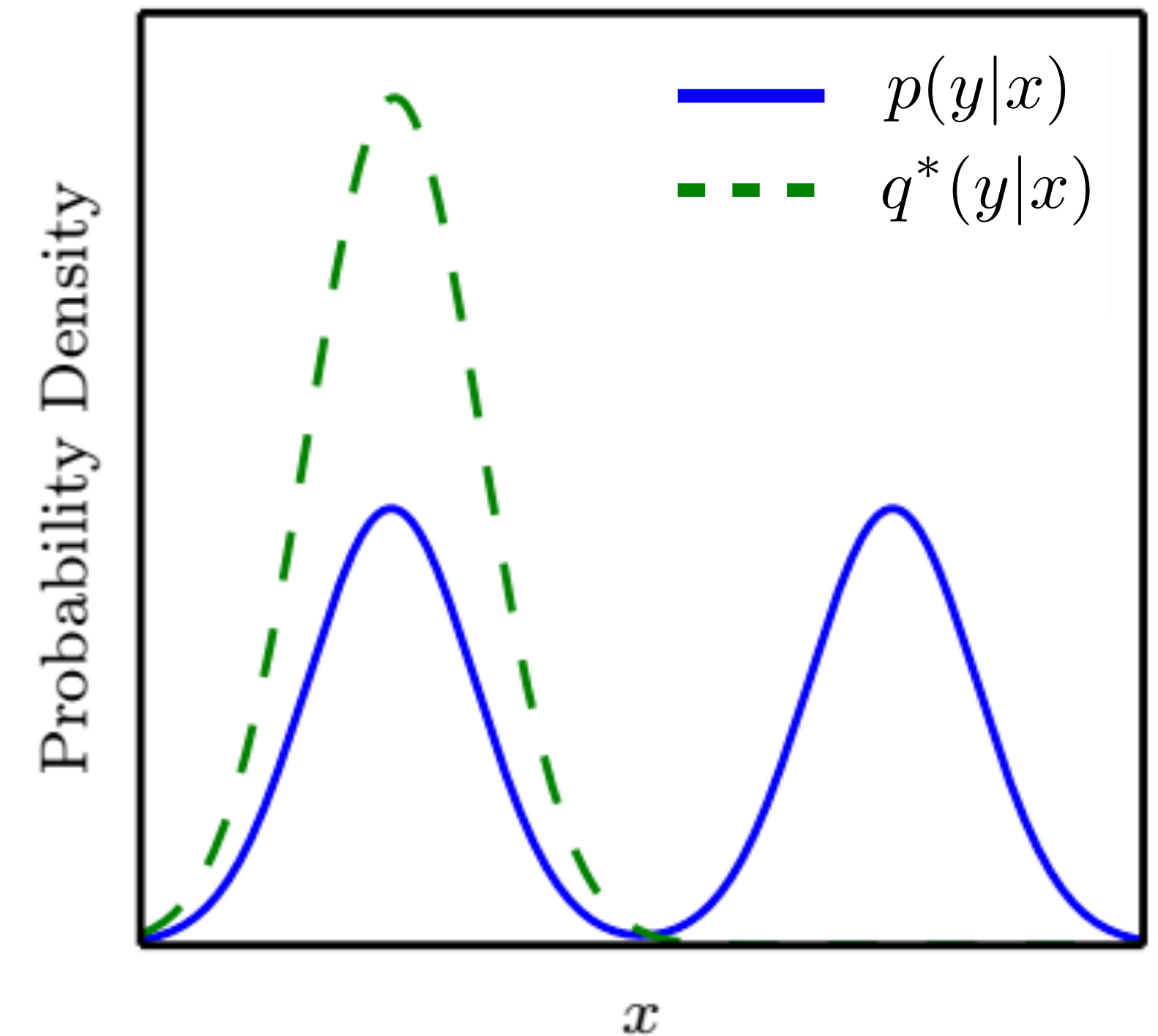


$$q^* = \operatorname{argmin}_q D_{\text{KL}}(p||q)$$



Maximum likelihood

$$q^* = \operatorname{argmin}_q D_{\text{KL}}(q||p)$$



Reverse KL

adapted from [Goodfellow, 2016]

Source: Isola, Freeman, Torralba

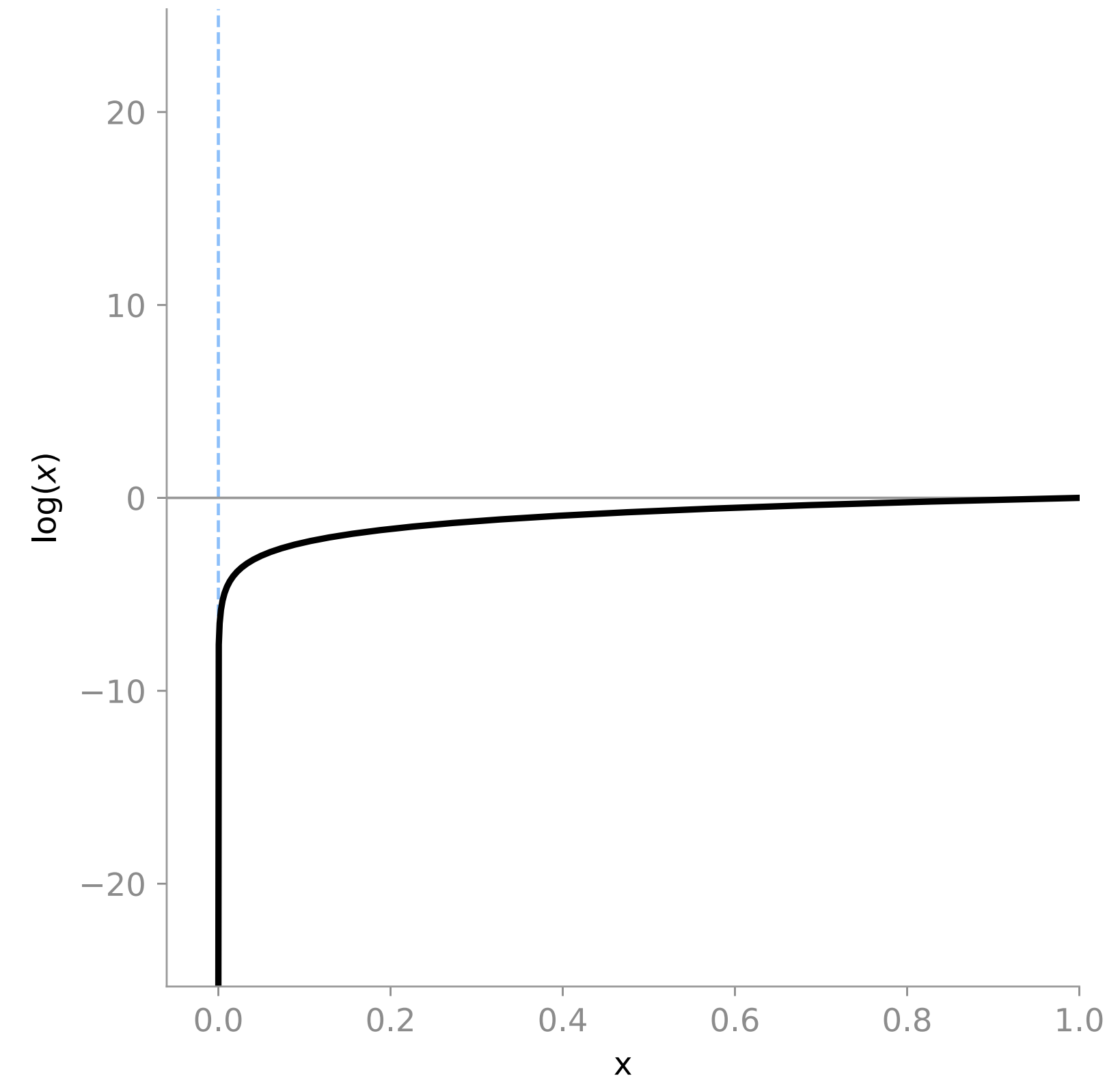
Mode seeking property

KL:

$$D_{KL}(p \parallel q) = \int_x p(x) \log \left(\frac{p(x)}{q(x)} \right) dx$$

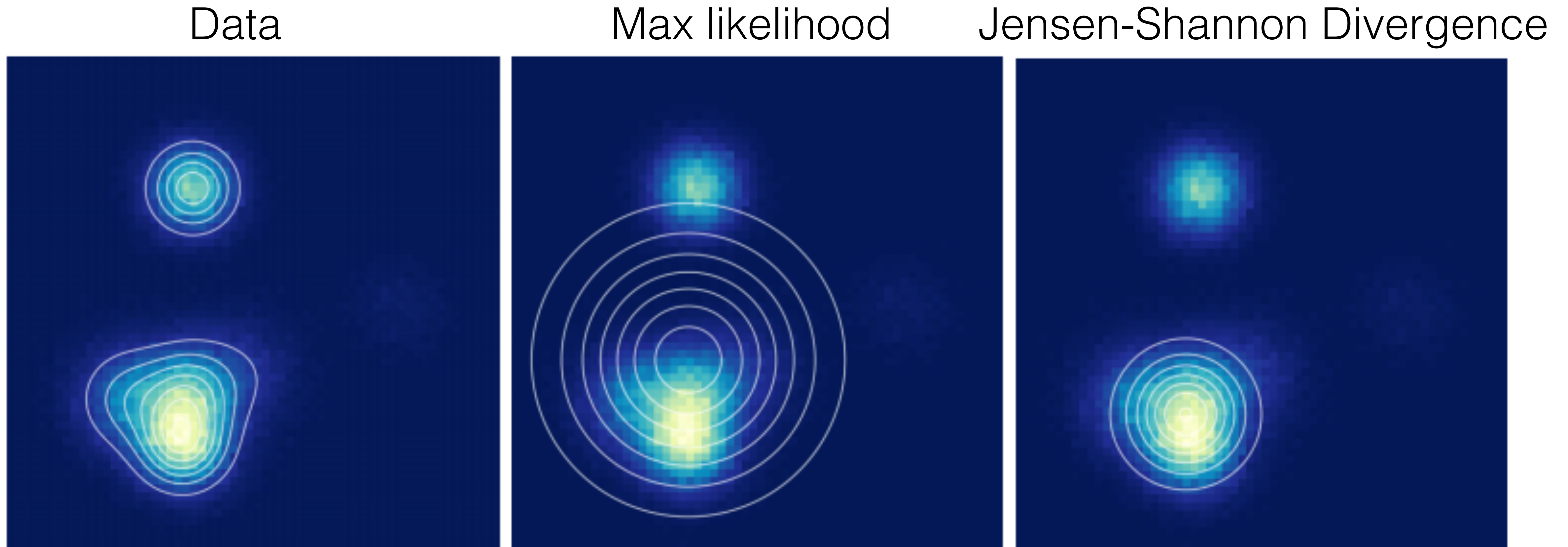
Reverse KL:

$$D_{KL}(q \parallel p) = \int_x q(x) \log \left(\frac{q(x)}{p(x)} \right) dx$$



What happens when you miss a mode, i.e., $p(x) > 0$ and $q(x)$ is near zero.

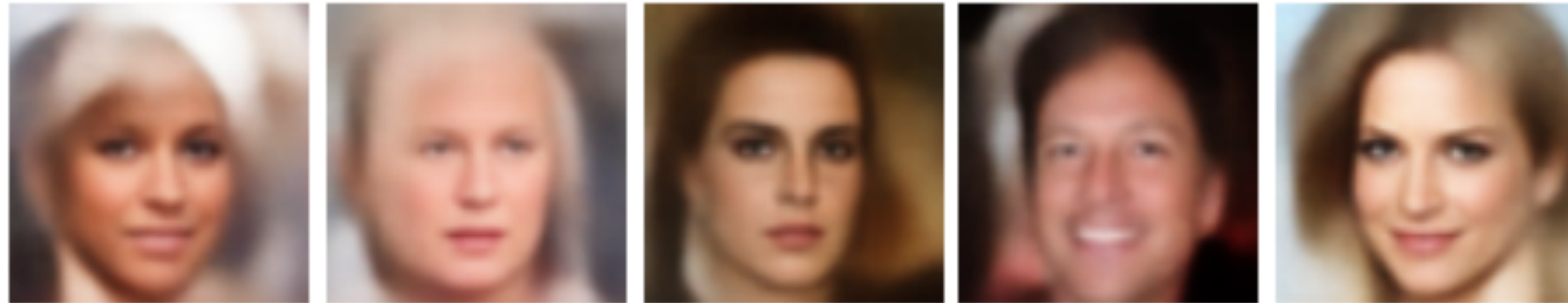
Behavior under model misspecification



["A note on the evaluation of generative models", Theis et al. 2016]

Mode covering versus mode seeking

VAE



GAN



[Larsen et al. 2016]

Does maximum likelihood lead to good samples?

- Suppose that we mix a really good model of a distribution, $p(x)$, with pure random noise, $q(x)$.

$$p_M(x) = 0.01p(x) + 0.99q(x)$$

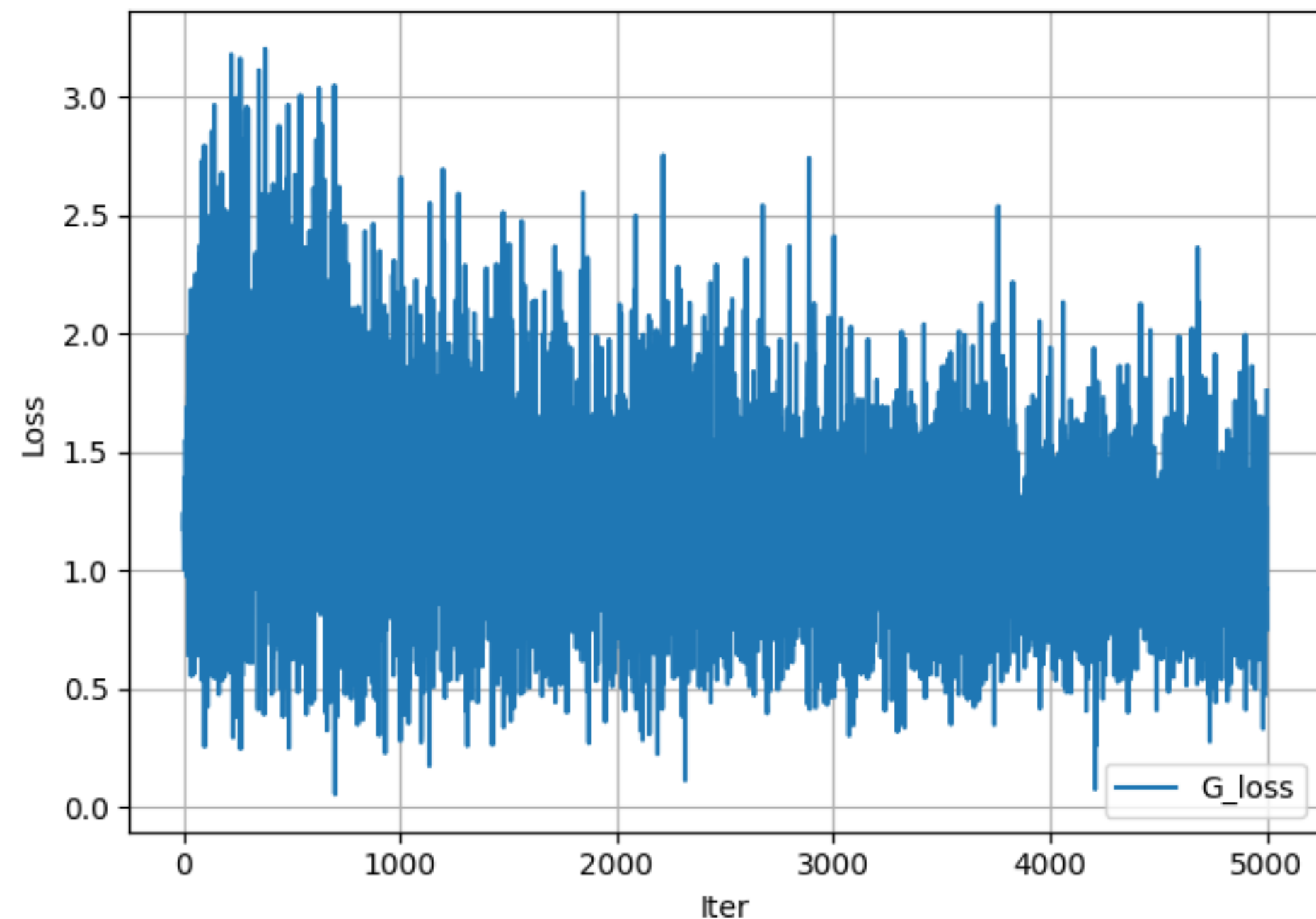
- What happens to the log likelihood for high dimensional data?

$$\log [0.01p(x) + 0.99q(x)] \geq \log(0.01p(x)) = \log p(x) - \log 100$$

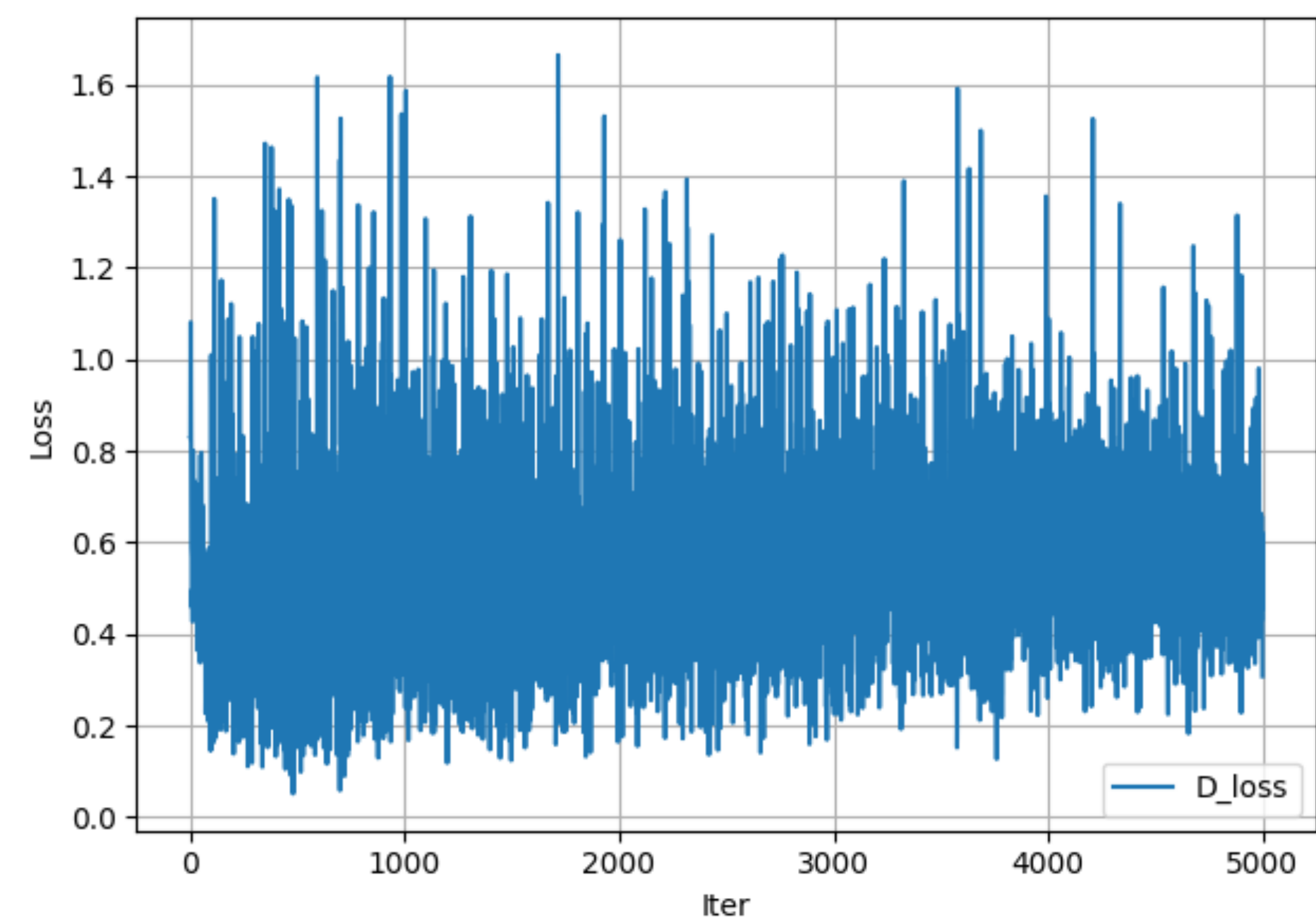
- But $\log p(x)$ is proportional d and $\log(100) \approx 4.61$ is a constant.
- Different models might differ in $\log(p(x))$ by $10000 \times$
- So, a model can generate bad samples most of the time without affecting log likelihood much.

Training dynamics of GANs

Loss doesn't necessarily go down! G and D play a two-player game.



Generator loss



Discriminator loss

Training dynamics of GANs

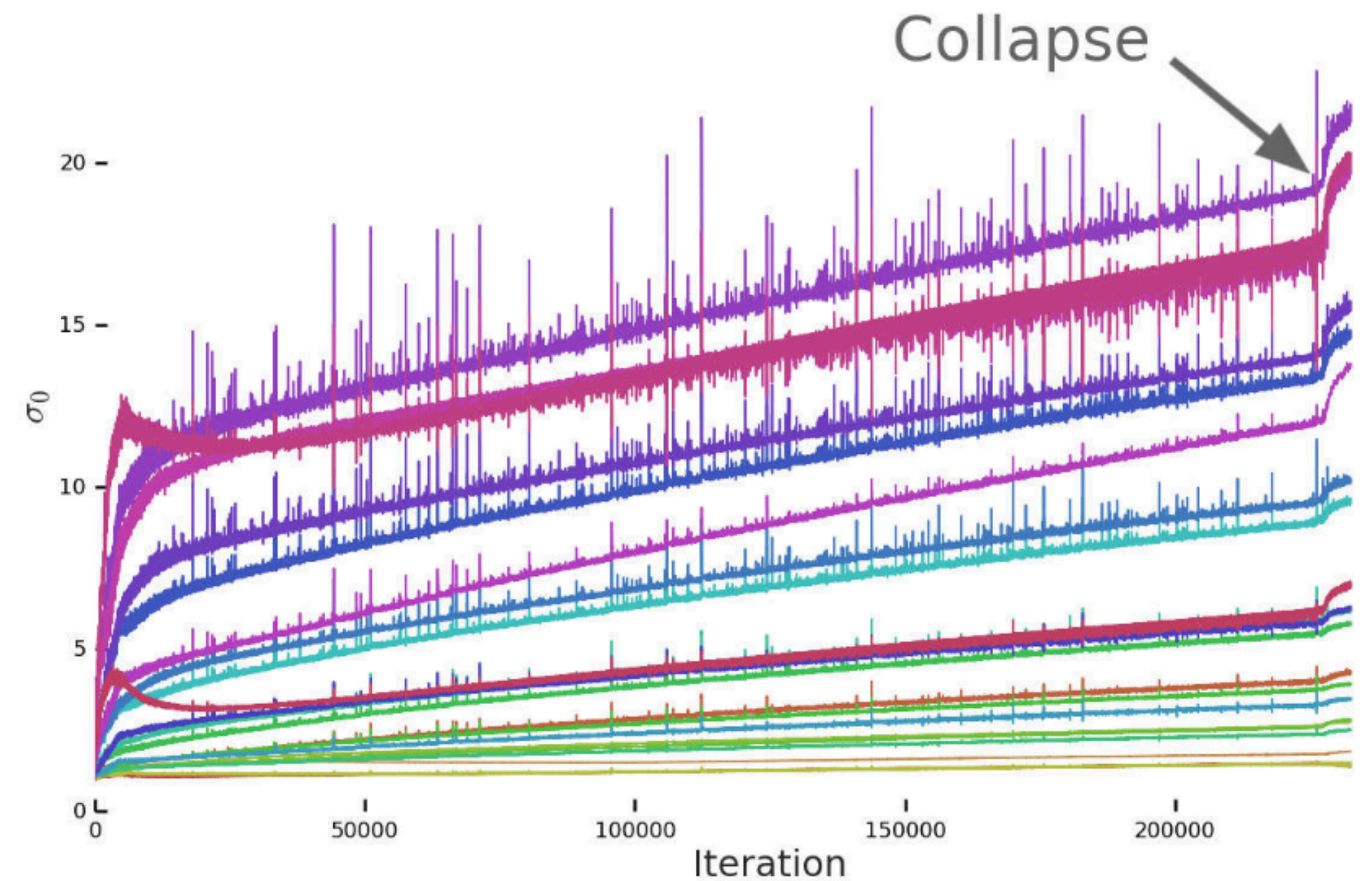
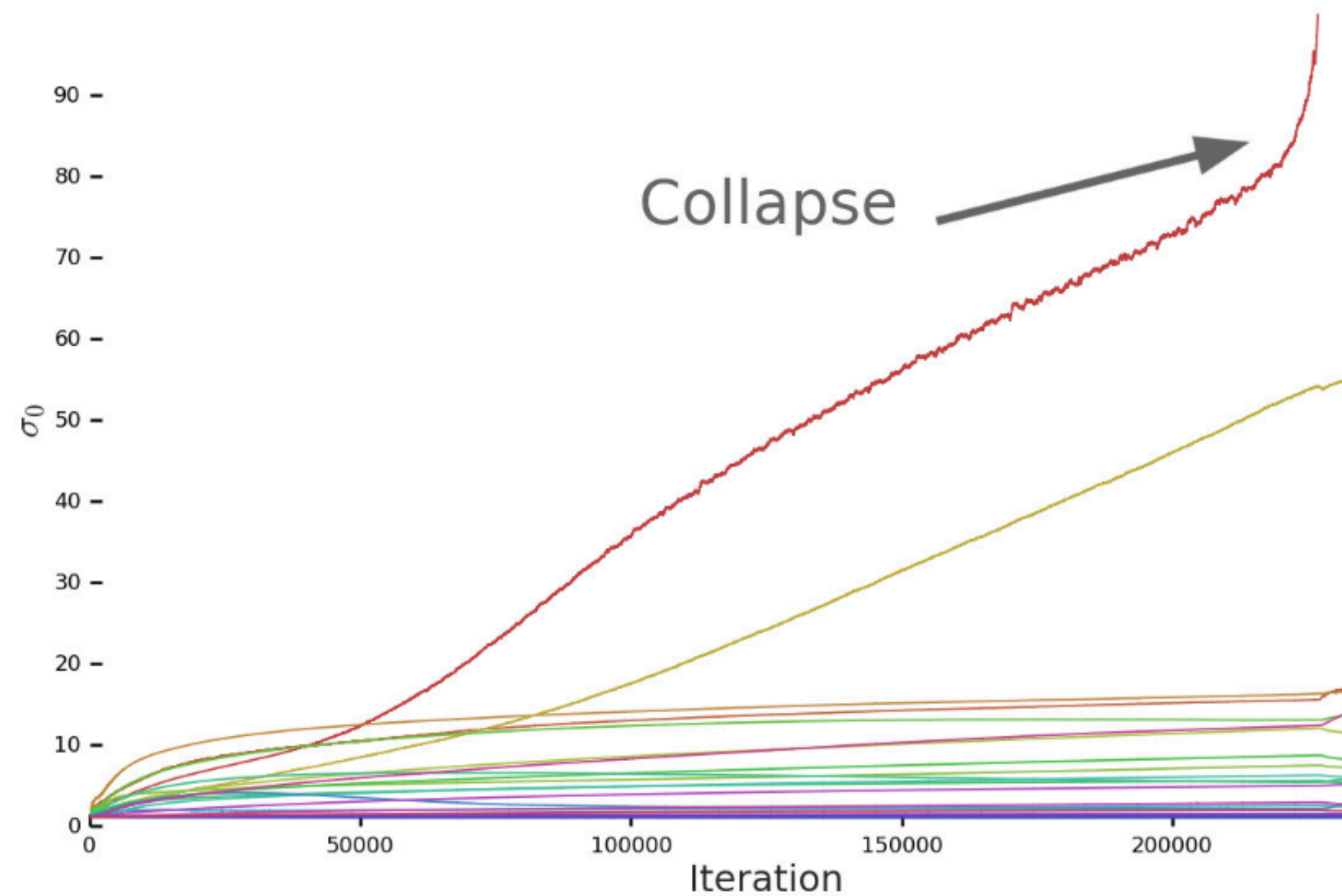
Some differences between GANs and other models we've seen:

- Not optimizing a single loss function.
- The two networks are constantly trying to fool each other.
- G backpropagates through D and gets feedback about what's wrong.
- Conceptually, D is a detective, while G is an art forger that gets to read the mind of D [Lethinen, 2017]

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

Training instabilities

Model can “collapse”: sample quality suddenly drops



Singular values of weight matrix

Rapid improvements in GAN quality (from 2014-2018)



Figure source: Ian Goodfellow

Better objectives? Optimization?

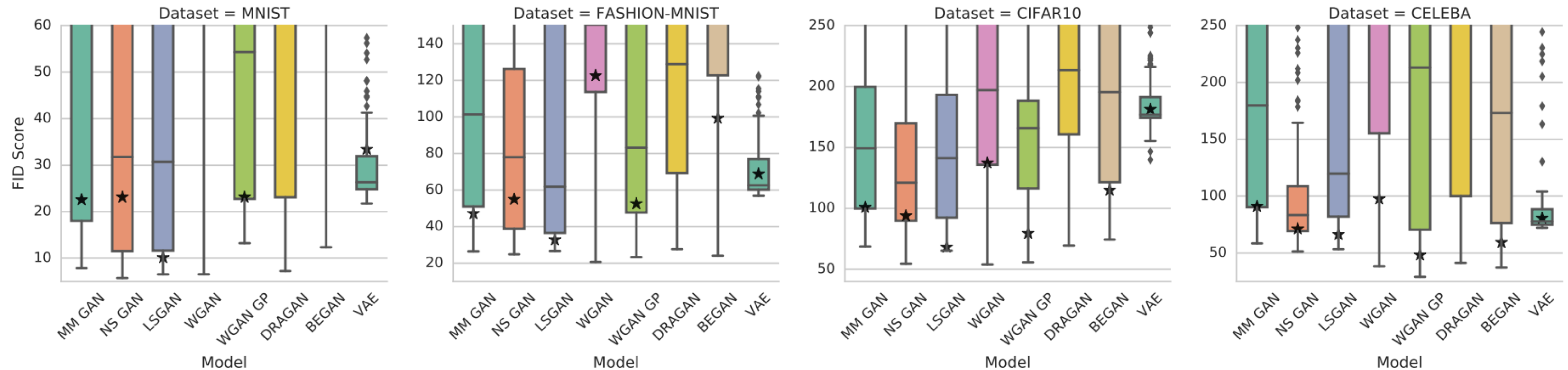


Figure 4: A *wide range* hyperparameter search (100 hyperparameter samples per model). Black stars indicate the performance of suggested hyperparameter settings. We observe that GAN training is extremely sensitive to hyperparameter settings and there is no model which is significantly more stable than others.

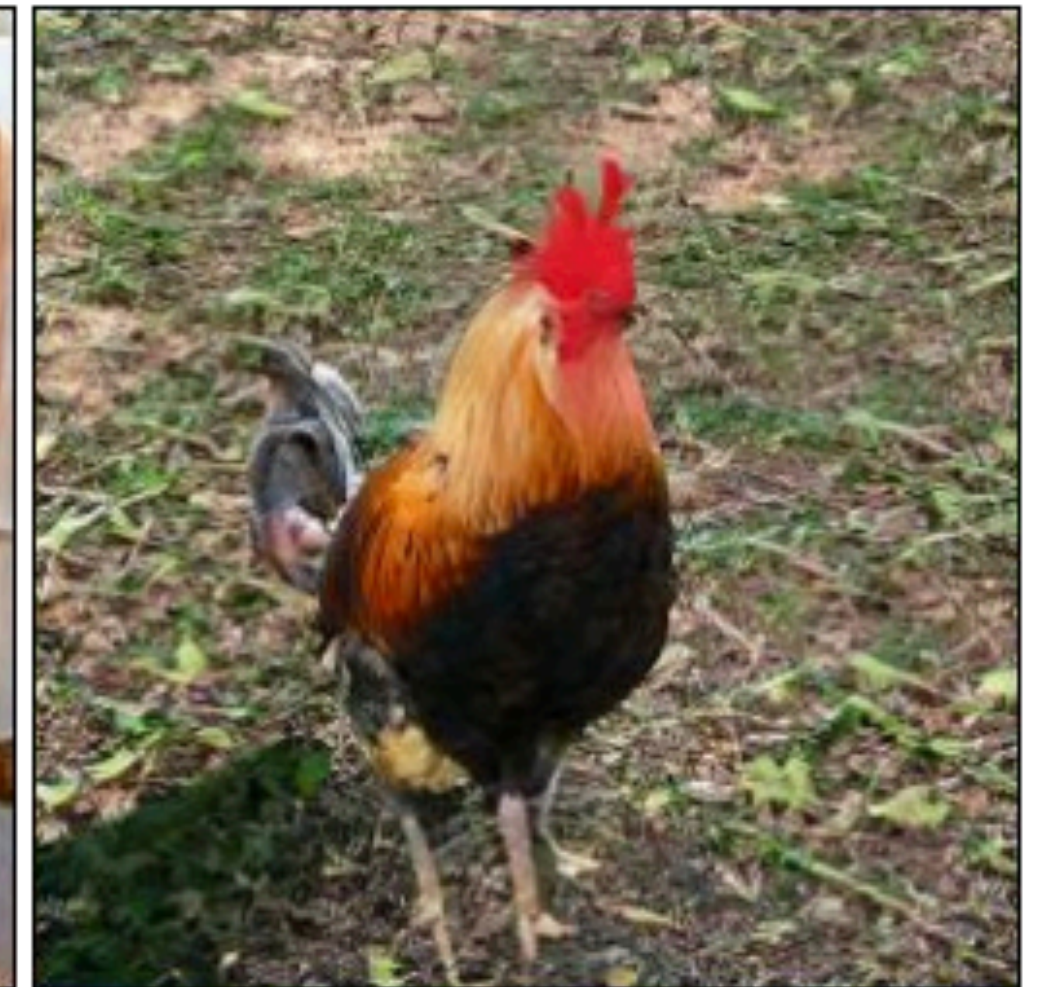
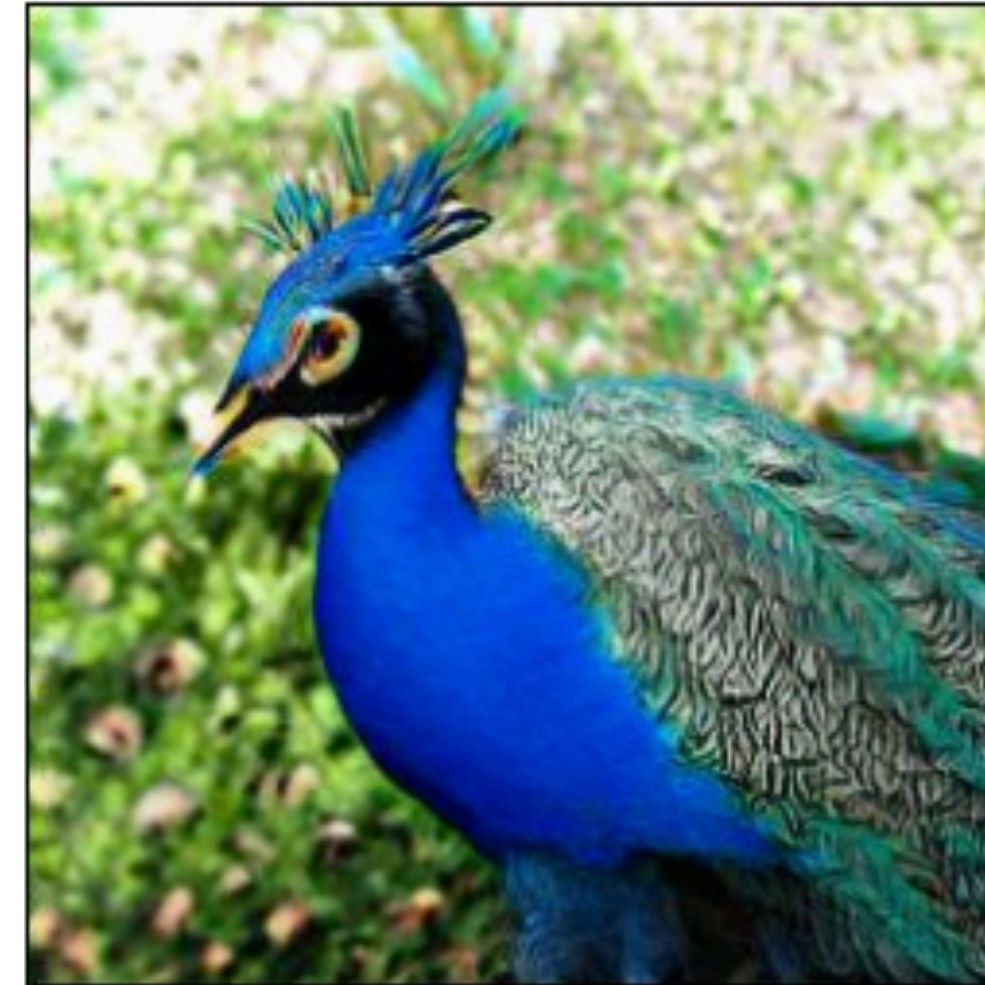
[“Are all GANs Created Equal?”, Lucic*, Kurach*, et al. 2018]

More data?

ACGAN [Odena et al. 2016]



BigGAN [Brock et al. 2018]



50

Both trained on Imagenet

Source: Isola, Freeman, Torralba

Rapid progress from better architectures

DCGAN

[Radford, Metz, Chintala 2016]



StyleGAN2

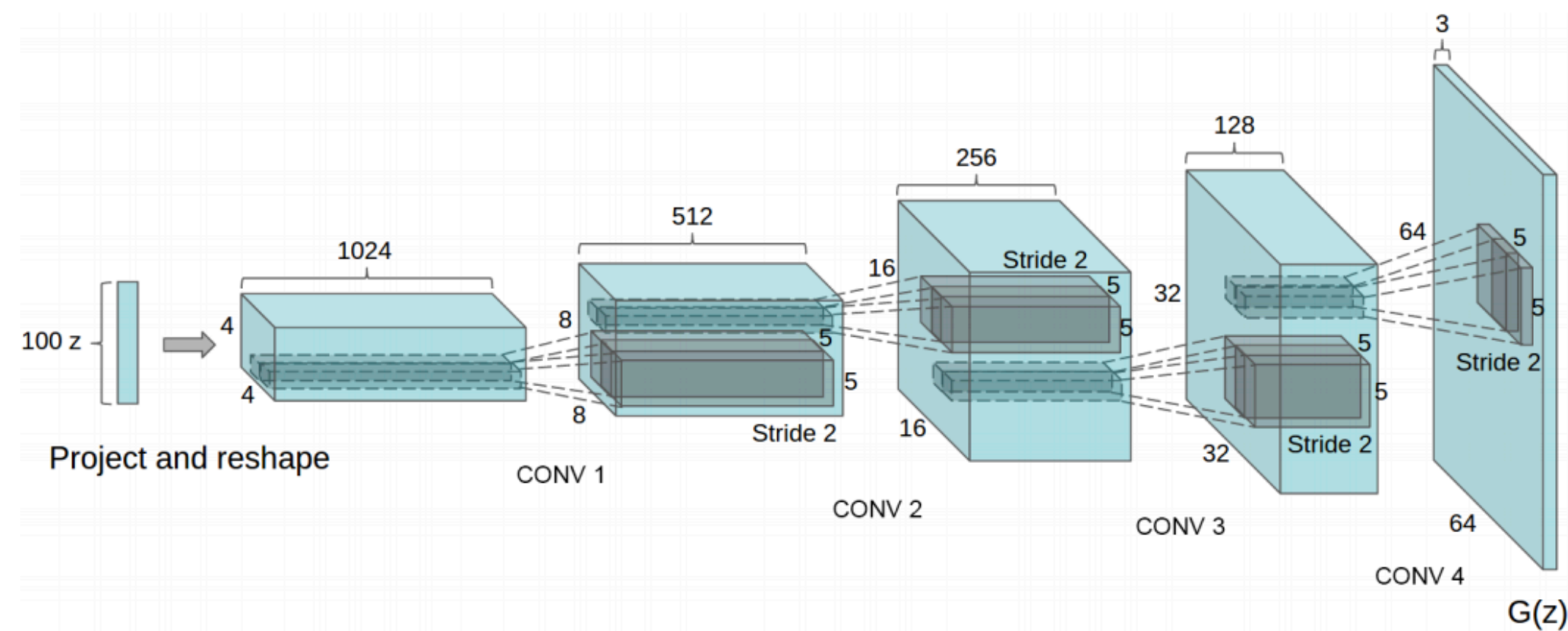
[Karras et al., 2020]



Architectures

DCGAN

[Radford, Metz, Chintala 2016]

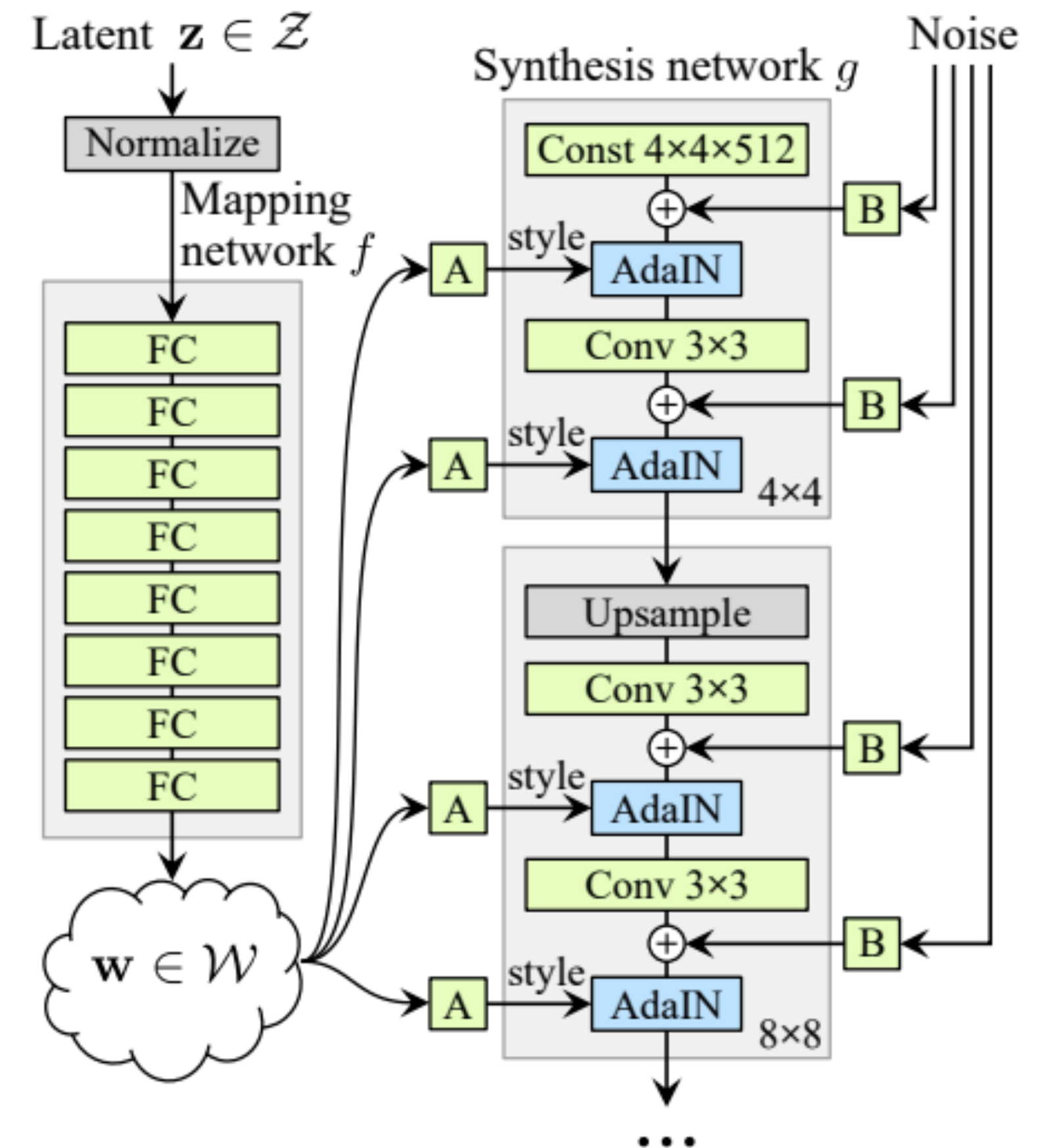


Transpose convolution + batch norm + nonlinearities



StyleGAN

[Karras, Laine, Aila 2019]

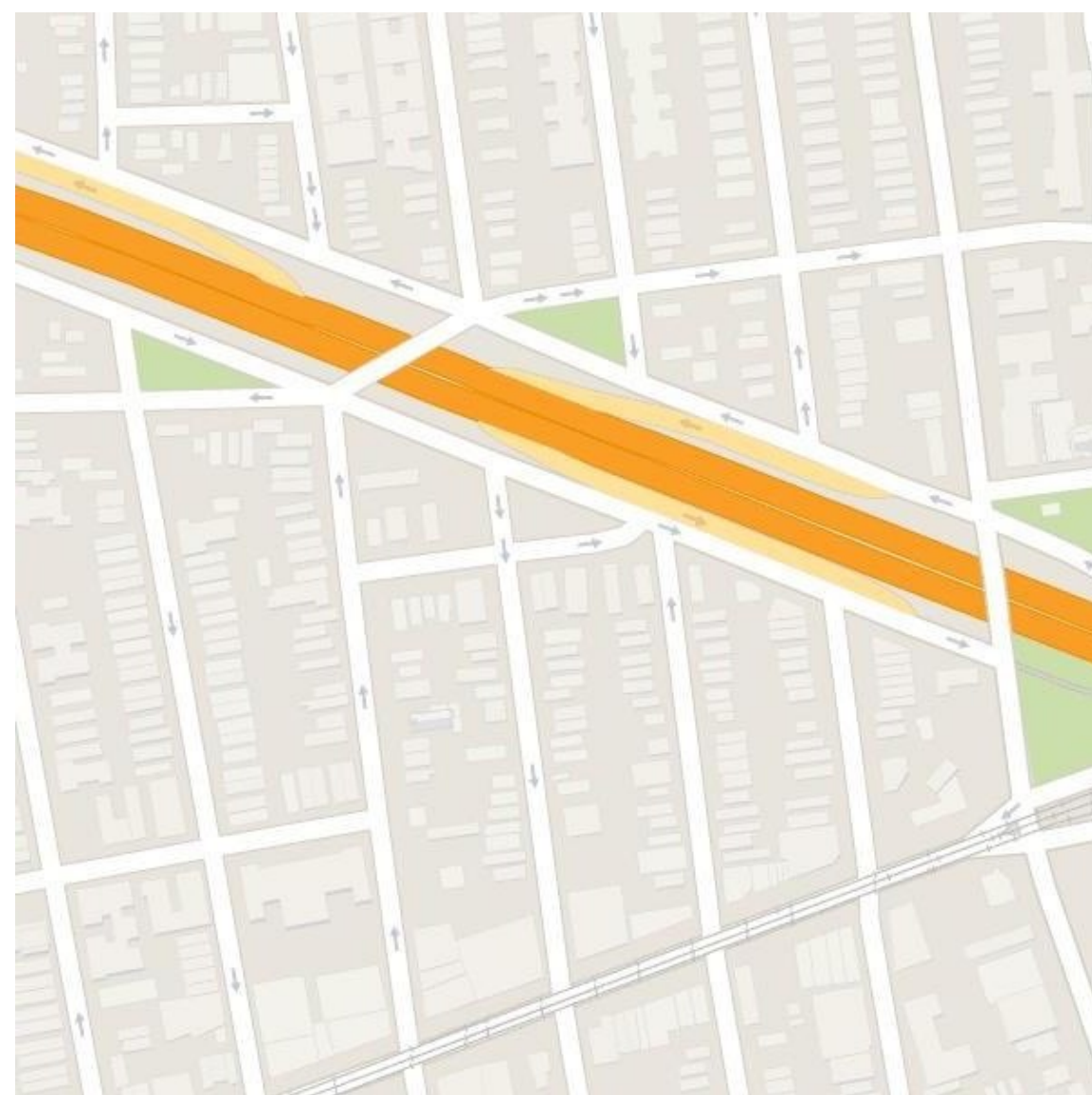


Similar but bigger and lots of engineering details.

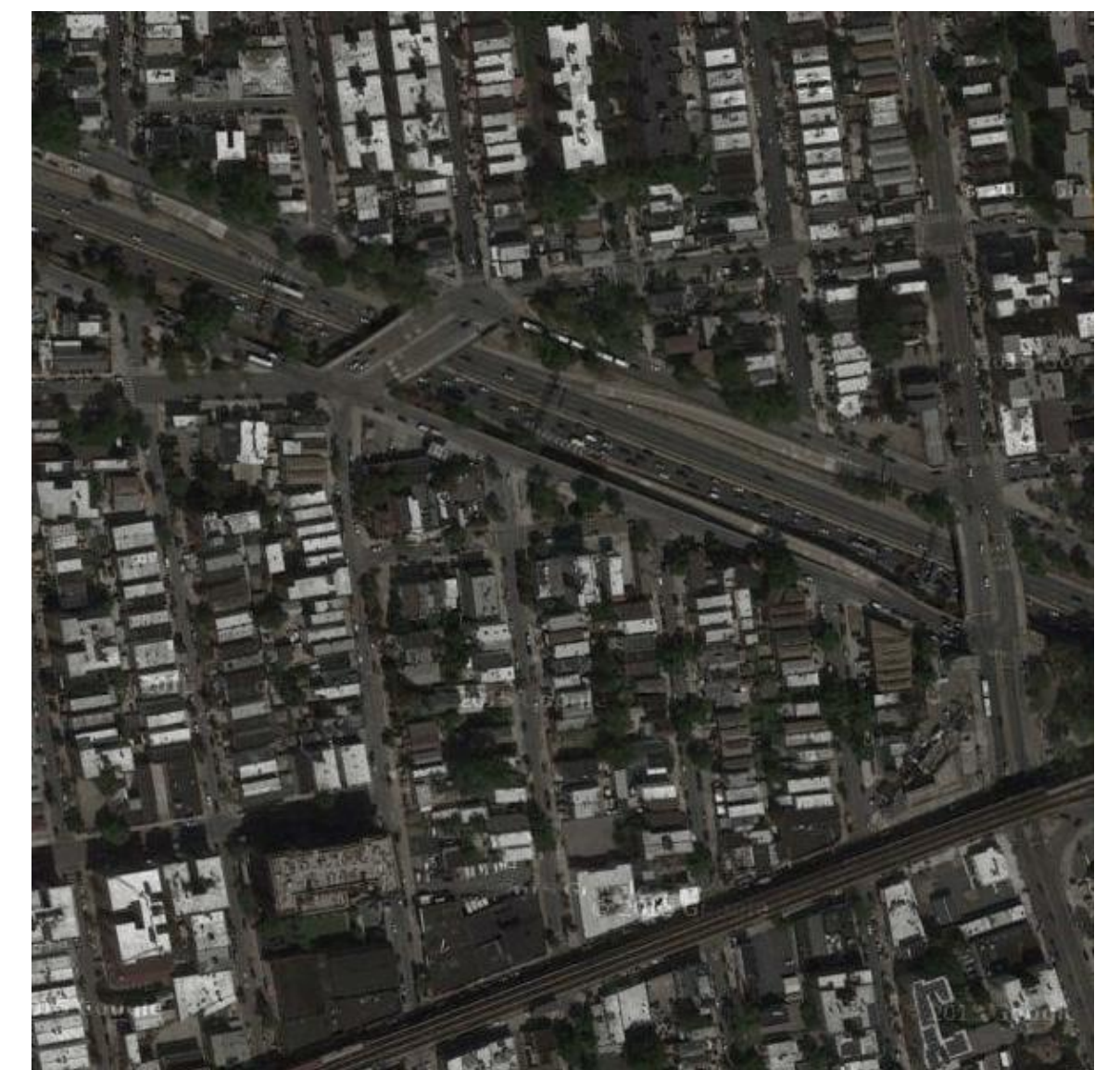
GANs vs. other models we've seen

- Often generate very qualitatively good samples (e.g., crisp images)
- Doesn't necessarily cover the full dataset.
- Train them by training models that play a minimax game.
- Unlike VAEs and normalizing flows, no single loss function that you can use to measure quality.
- Training can collapse (e.g., if one model wins)

Next class: GANs for images



Google Map



Satellite photo