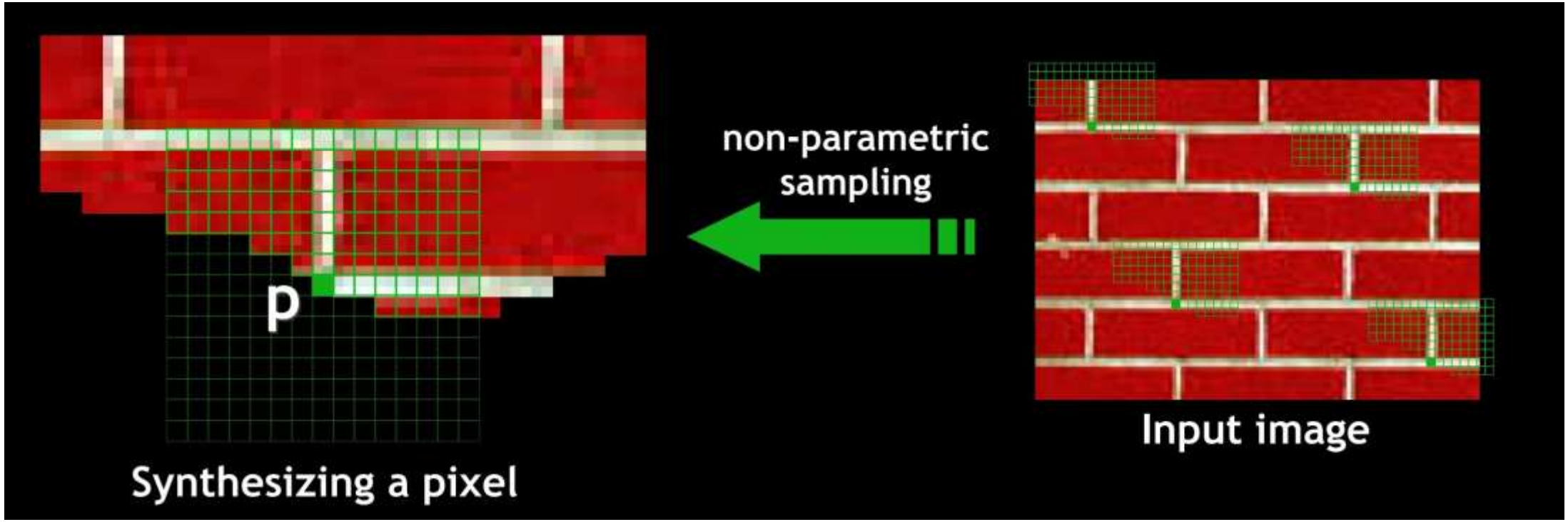
Recap.

Generative Models before the “GenAI” Era

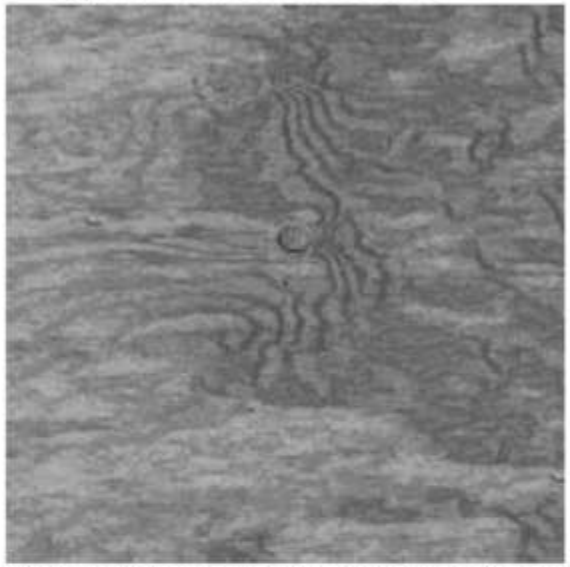


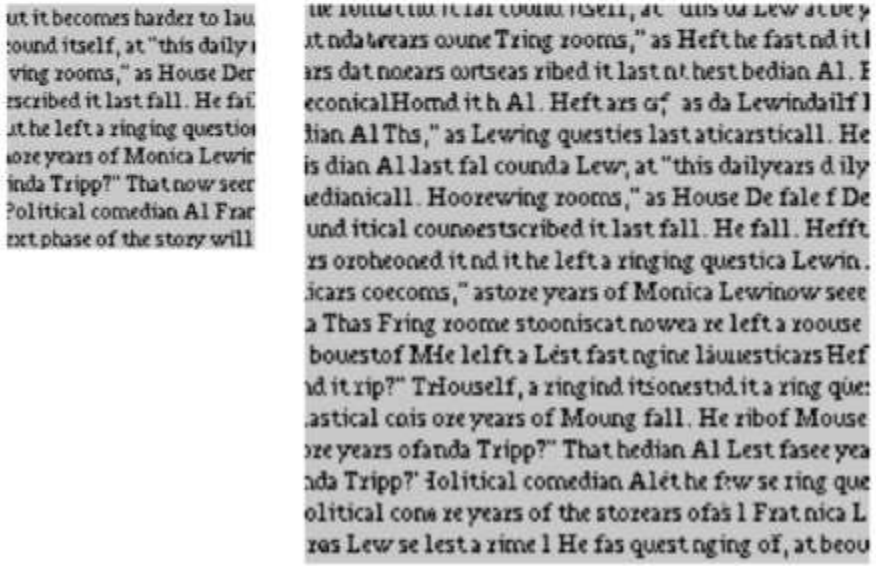
Texture Synthesis by Non-parametric Sampling, ICCV 1999

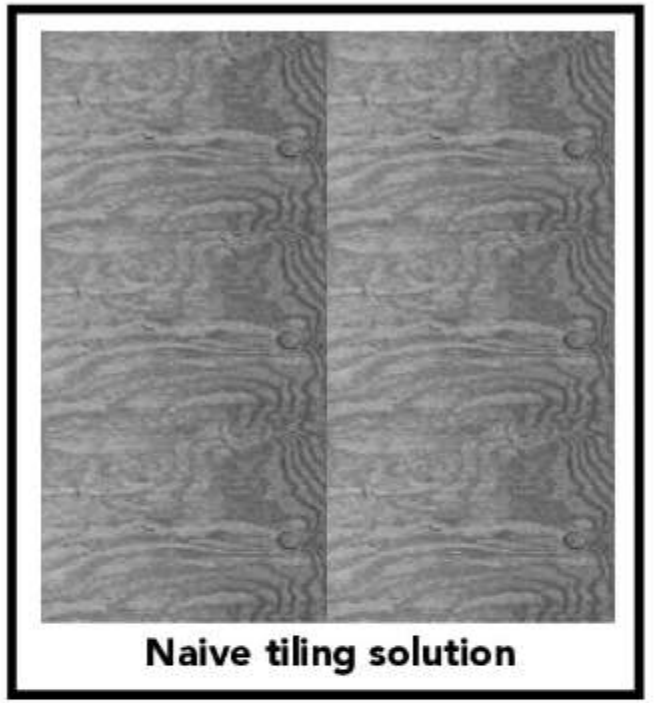
Generative Models before the “GenAI” Era

Source textures

Synthesized Textures







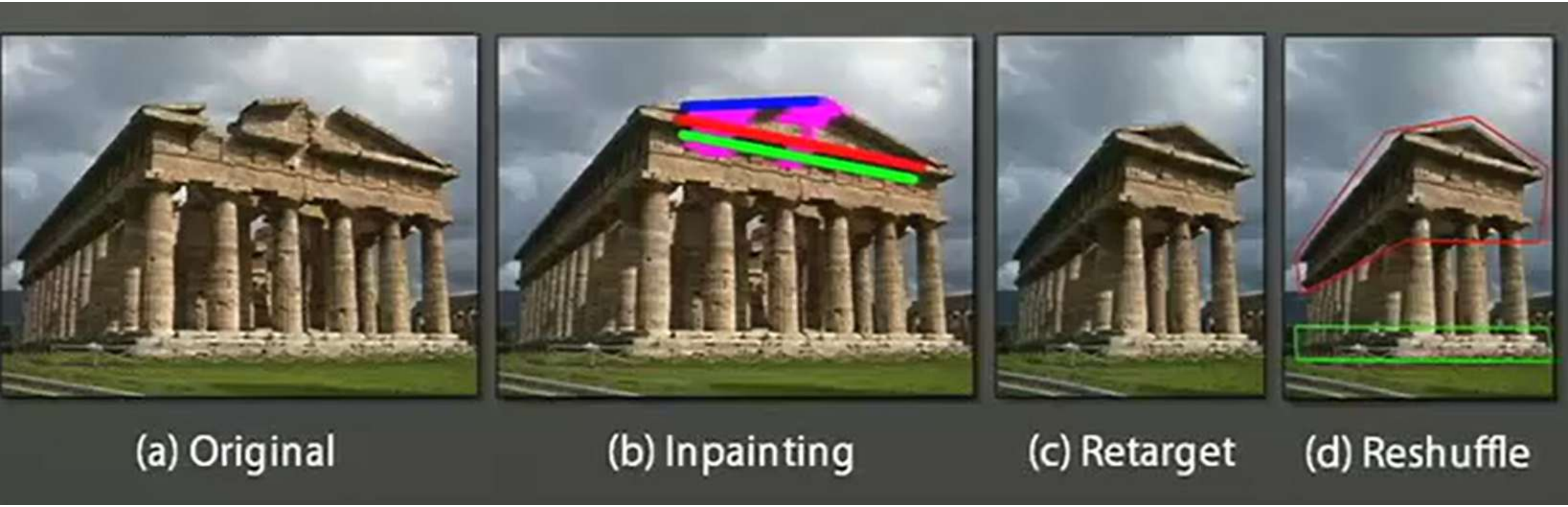
Naive tiling solution

[Efros and Leung 99]

Texture Synthesis by Non-parametric Sampling, ICCV 1999

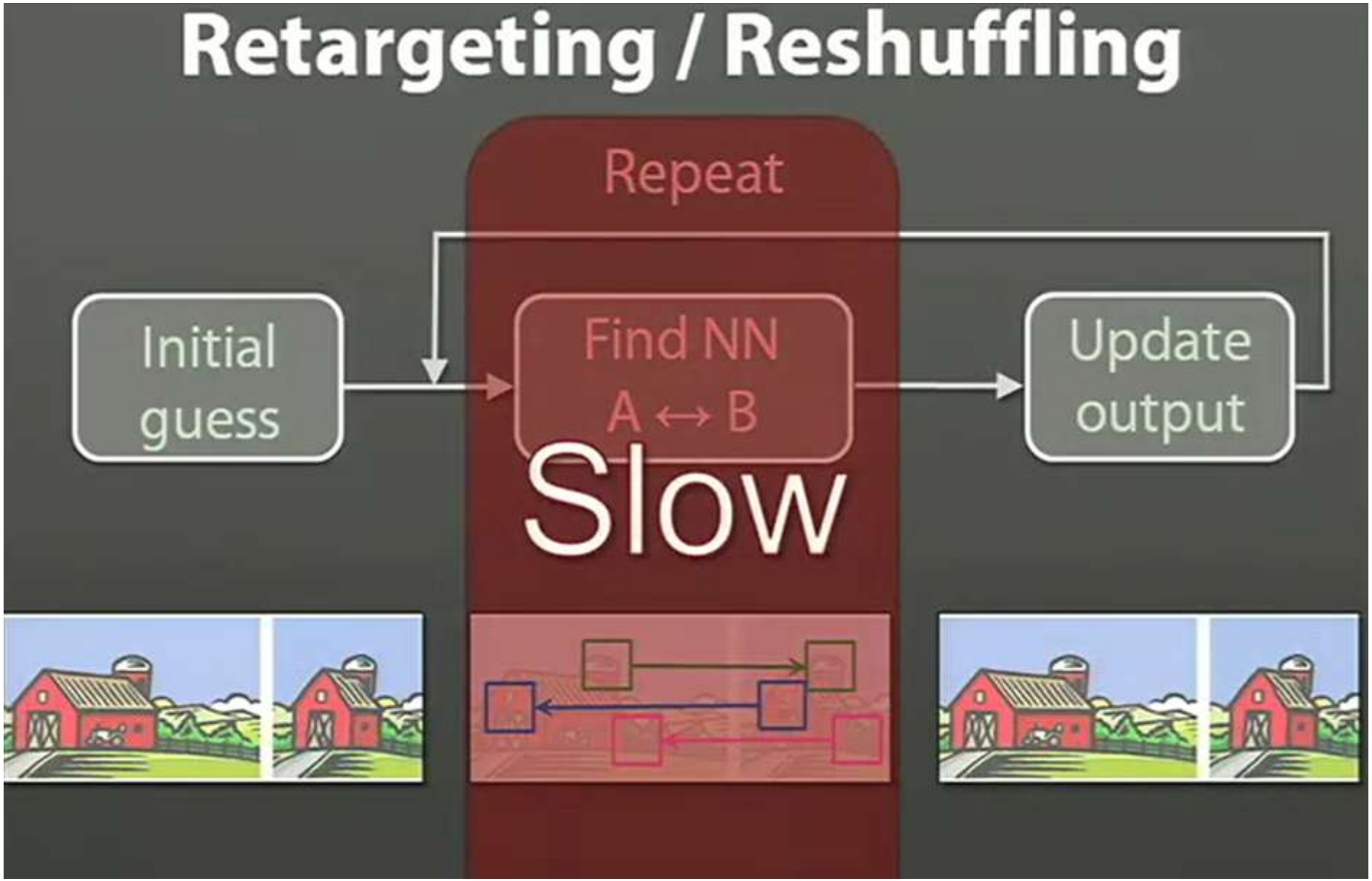
PatchMatch

* C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. TOG, 2009.



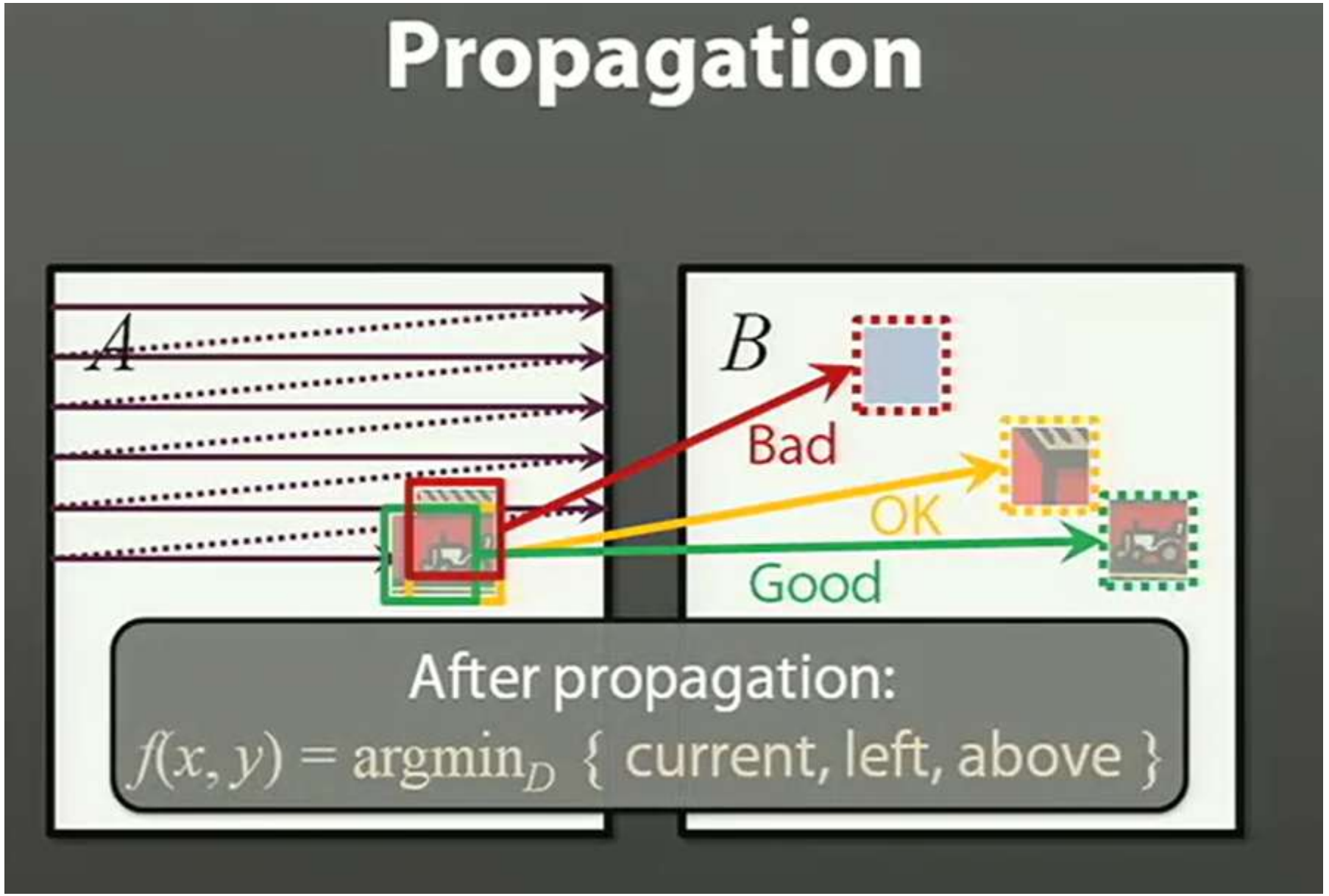
PatchMatch

* C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. TOG, 2009.



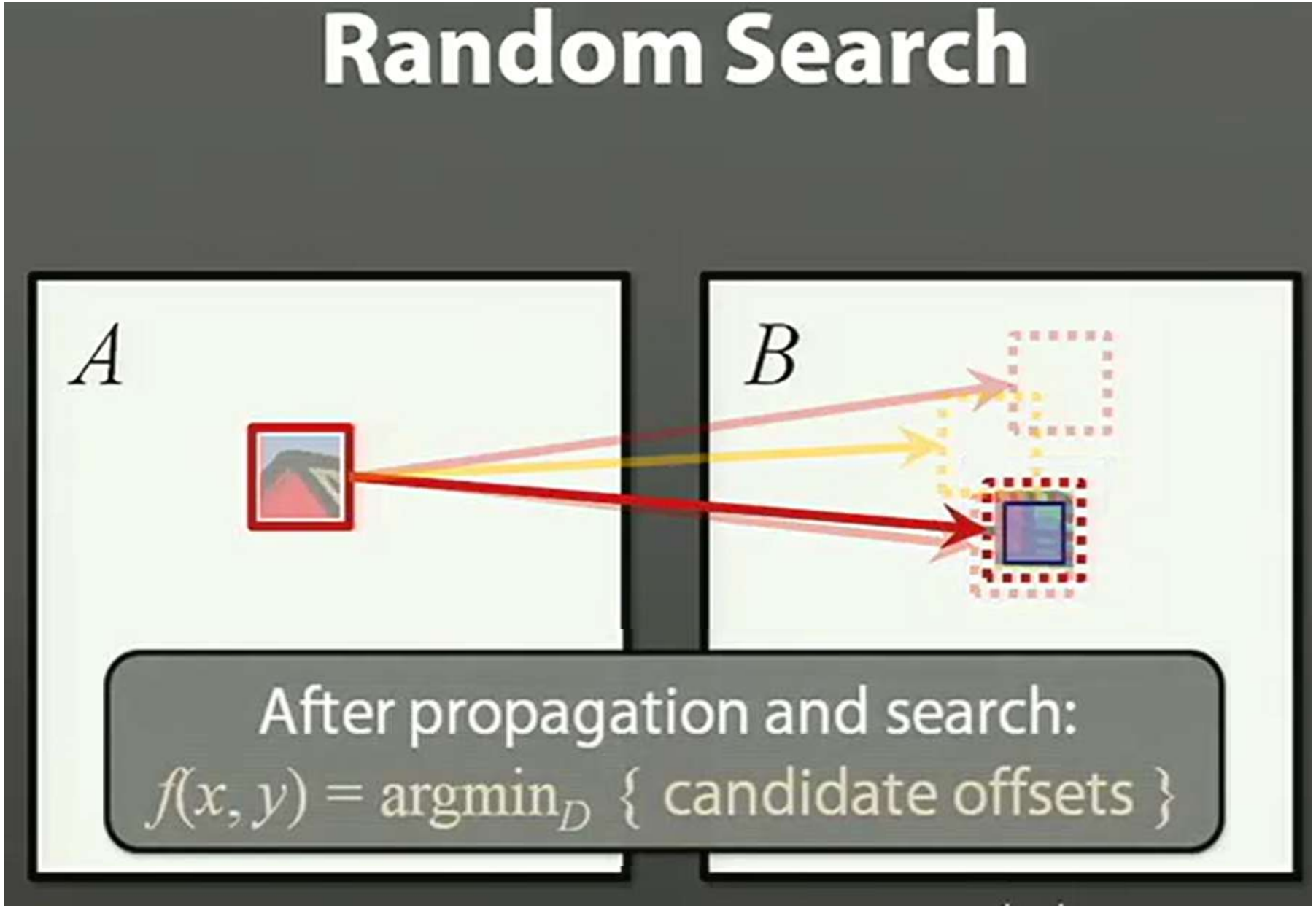
PatchMatch

* C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. TOG, 2009.



PatchMatch

* C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. TOG, 2009.

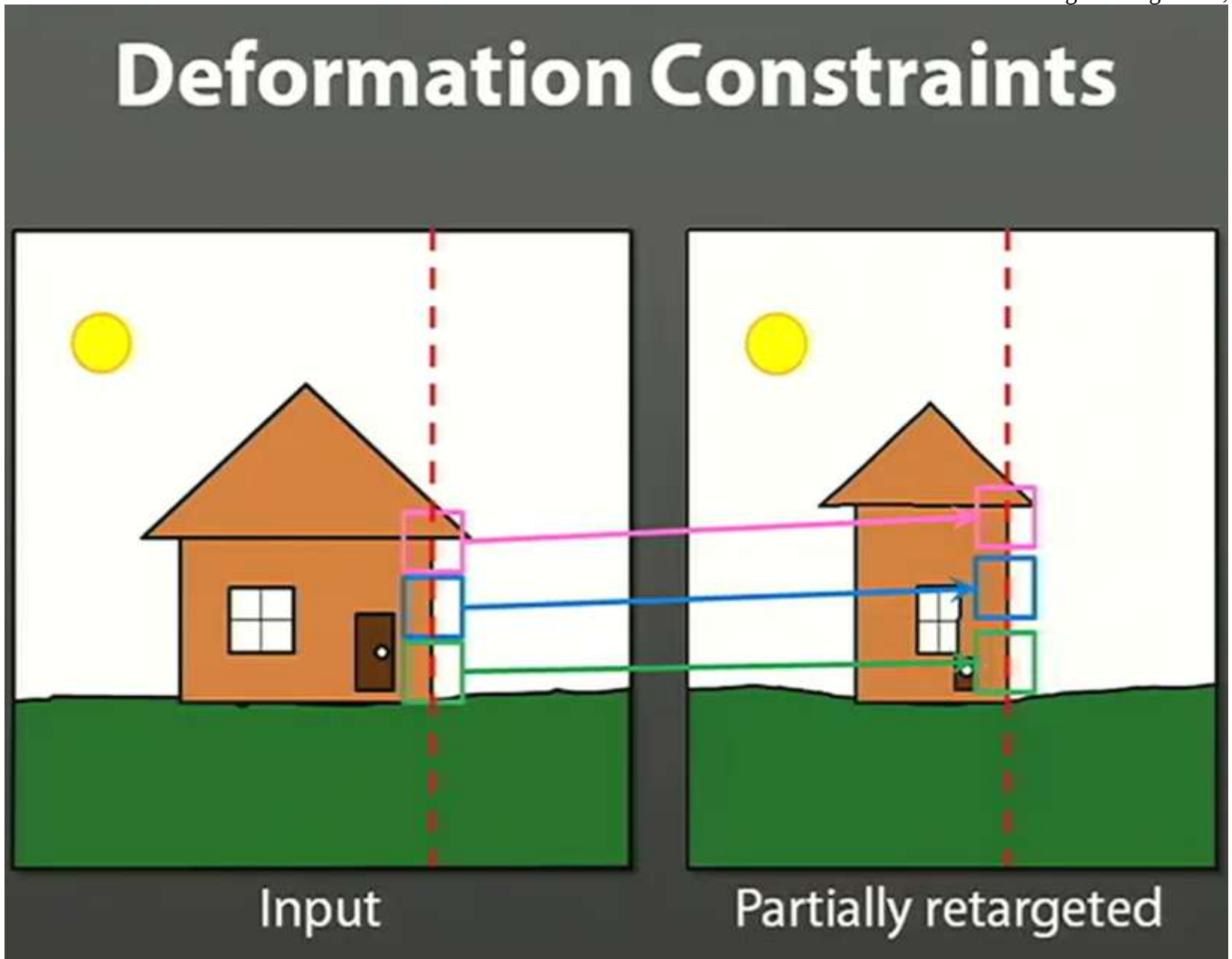


PatchMatch

* C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. TOG, 2009.

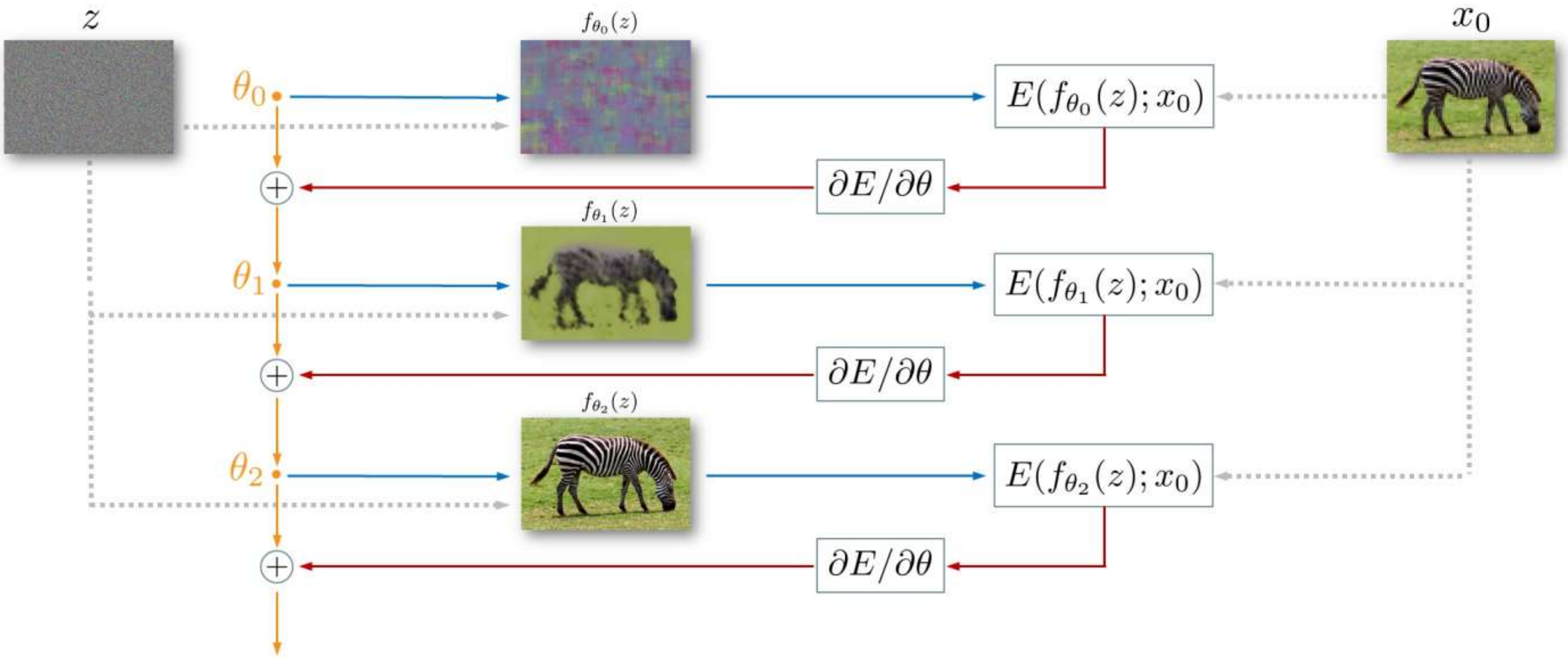
PatchMatch

* C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. TOG, 2009.



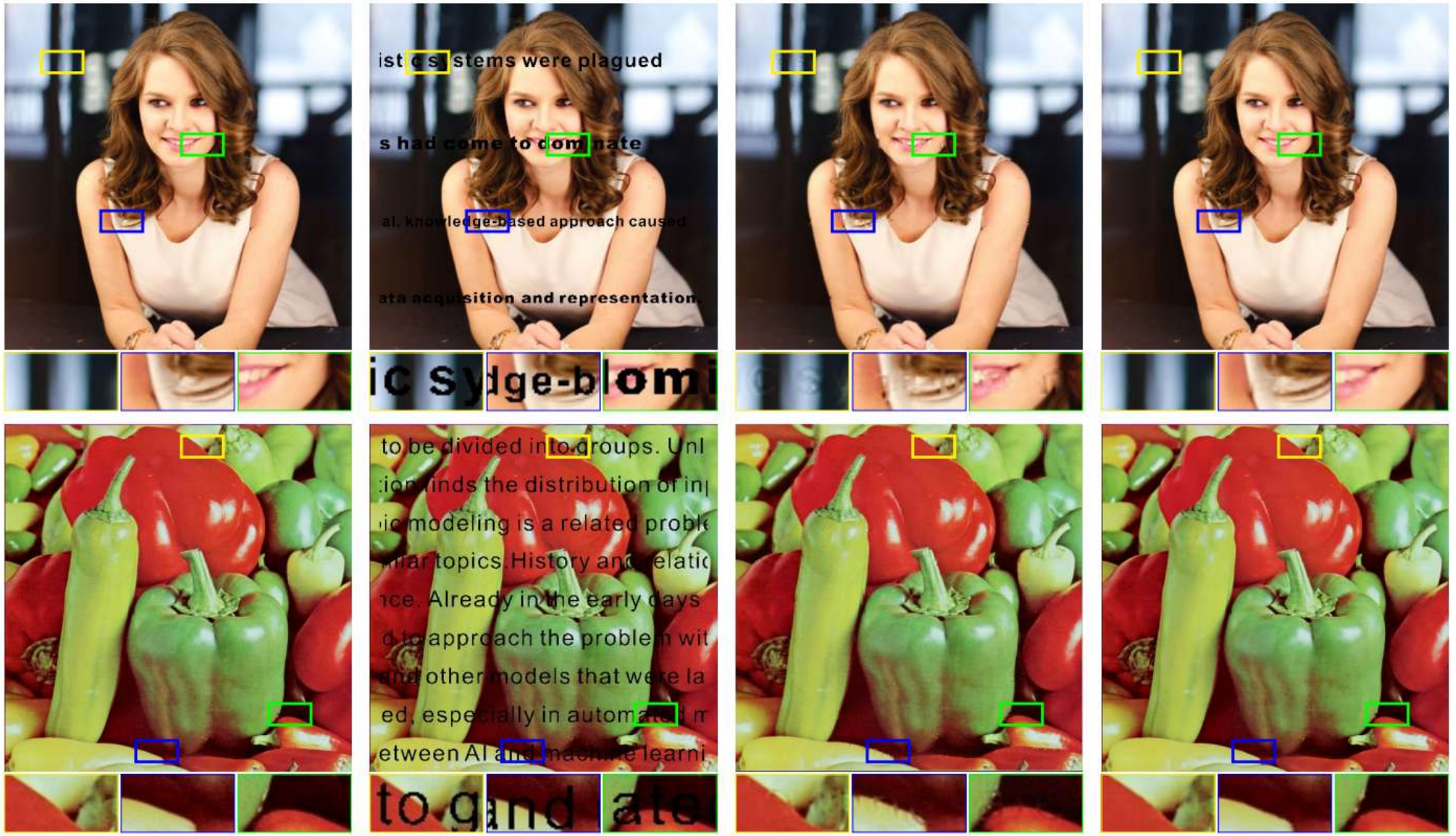
Deep Image Prior

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.



Deep Image Prior

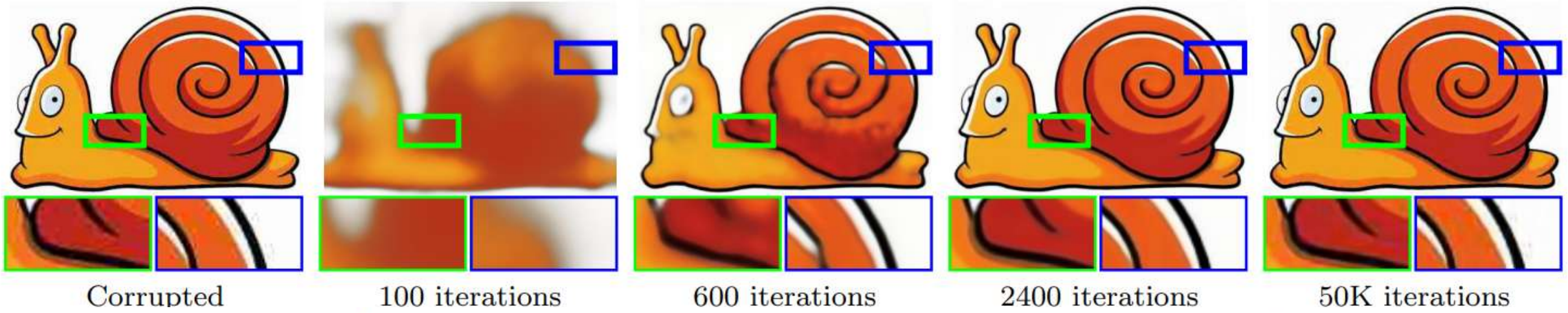
Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

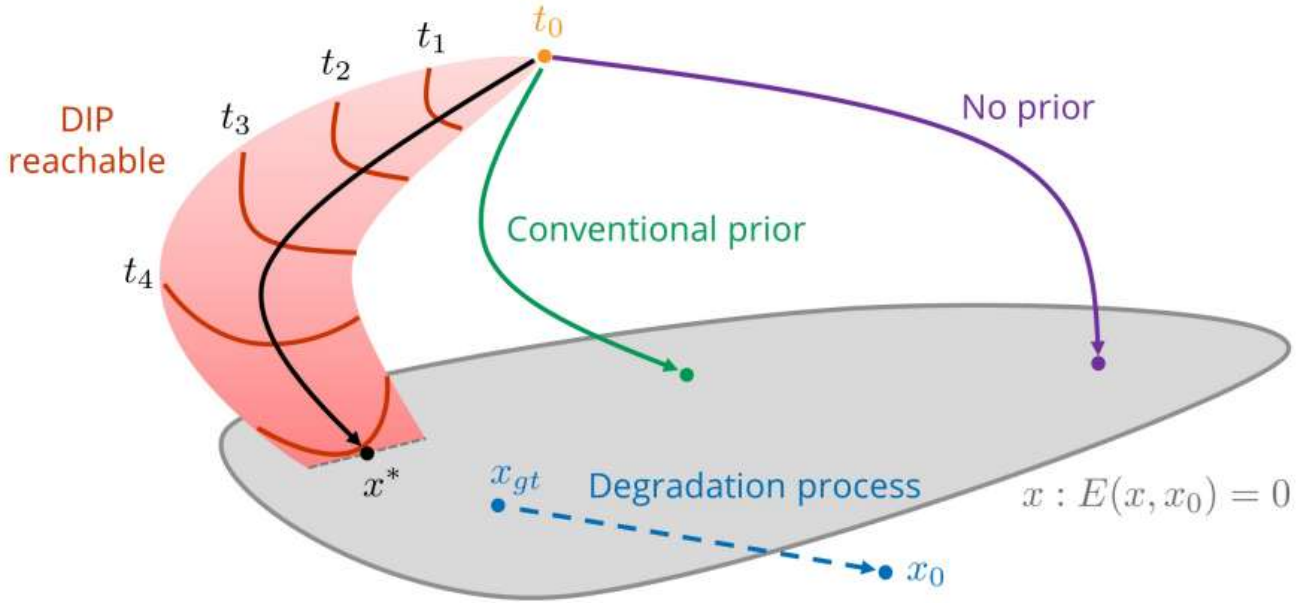


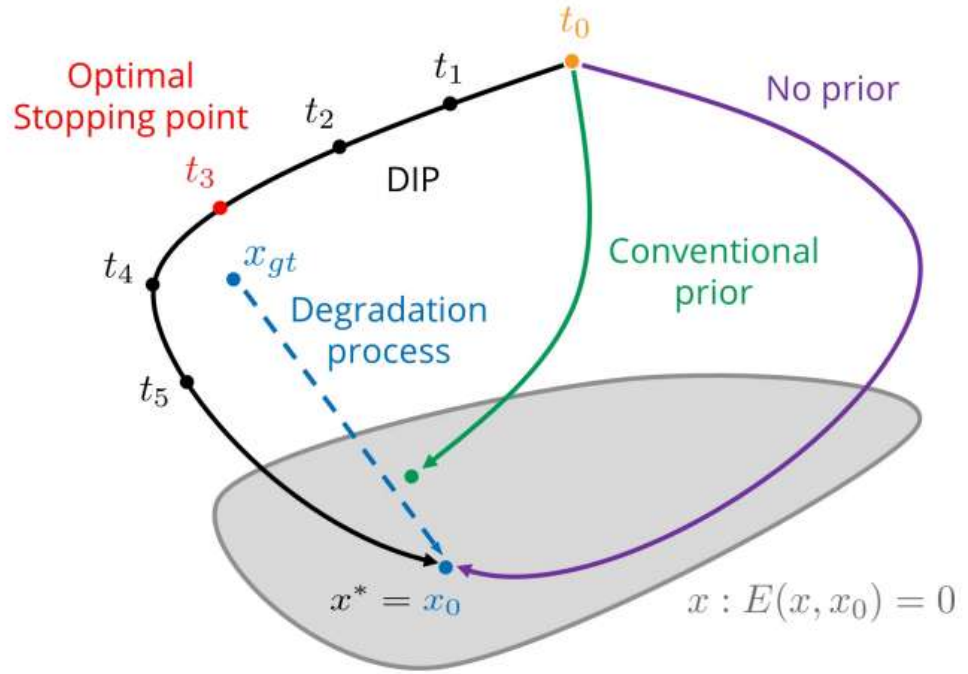
(a) Original image (b) Corrupted image (c) Shepard networks [44] (d) Deep Image Prior

Deep Image Prior

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.







Create New Content

- Patchmatch and Deep Image Prior can’t provide new images
  - PatchMatch: borrow new content from other images
  - DIP: new noise? Totally out of control

- How?

Variational Autoencoder (VAE)

Overview

- Variational Autoencoder (VAE)
- Relation to Expectation-Maximization (EM)
- Vector Quantized VAE (VQ-VAE)

Procedural graphics

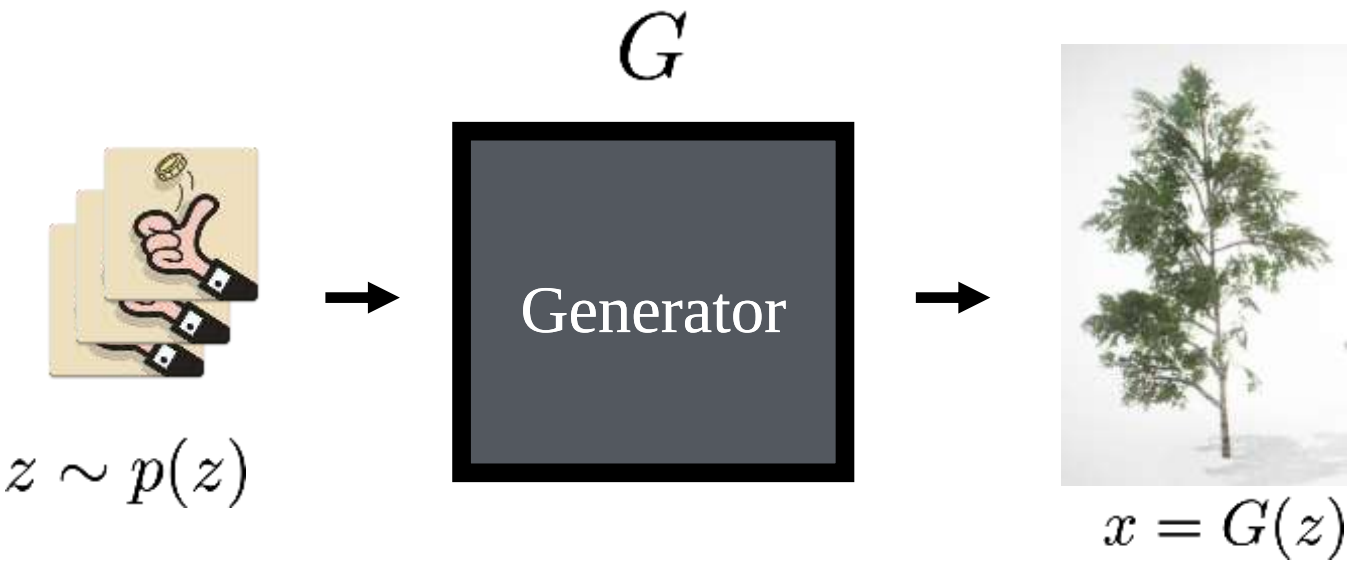
[Anders Scheil]

Aylean @Aylean · Nov 17

Made up a set of rules and rolled some dice to decide how this plant would grow. I never did get that five of a kind, as expected, but I was still hopeful! 🌱🍀🎉

52 1.1K 4.5K

Image synthesis from “noise”



Sampler

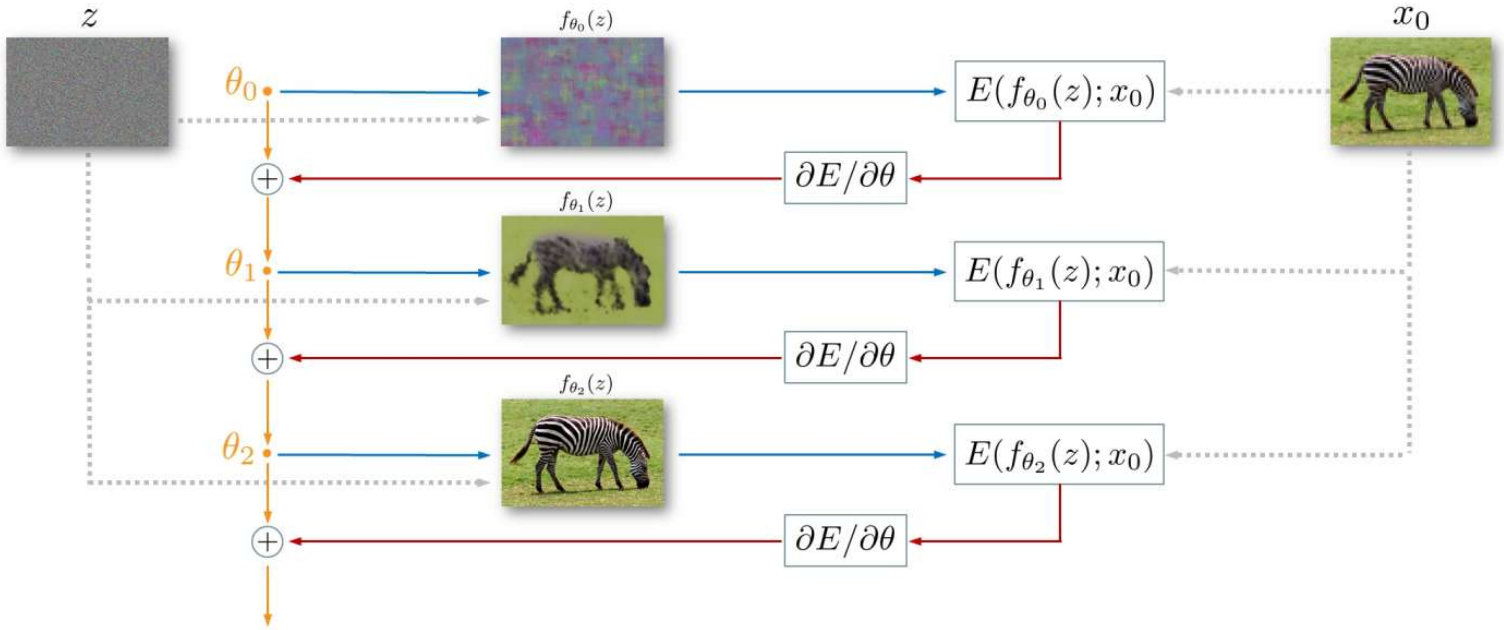
$$G : \mathcal{Z} \rightarrow \mathcal{X}$$

$$z \sim p(z)$$

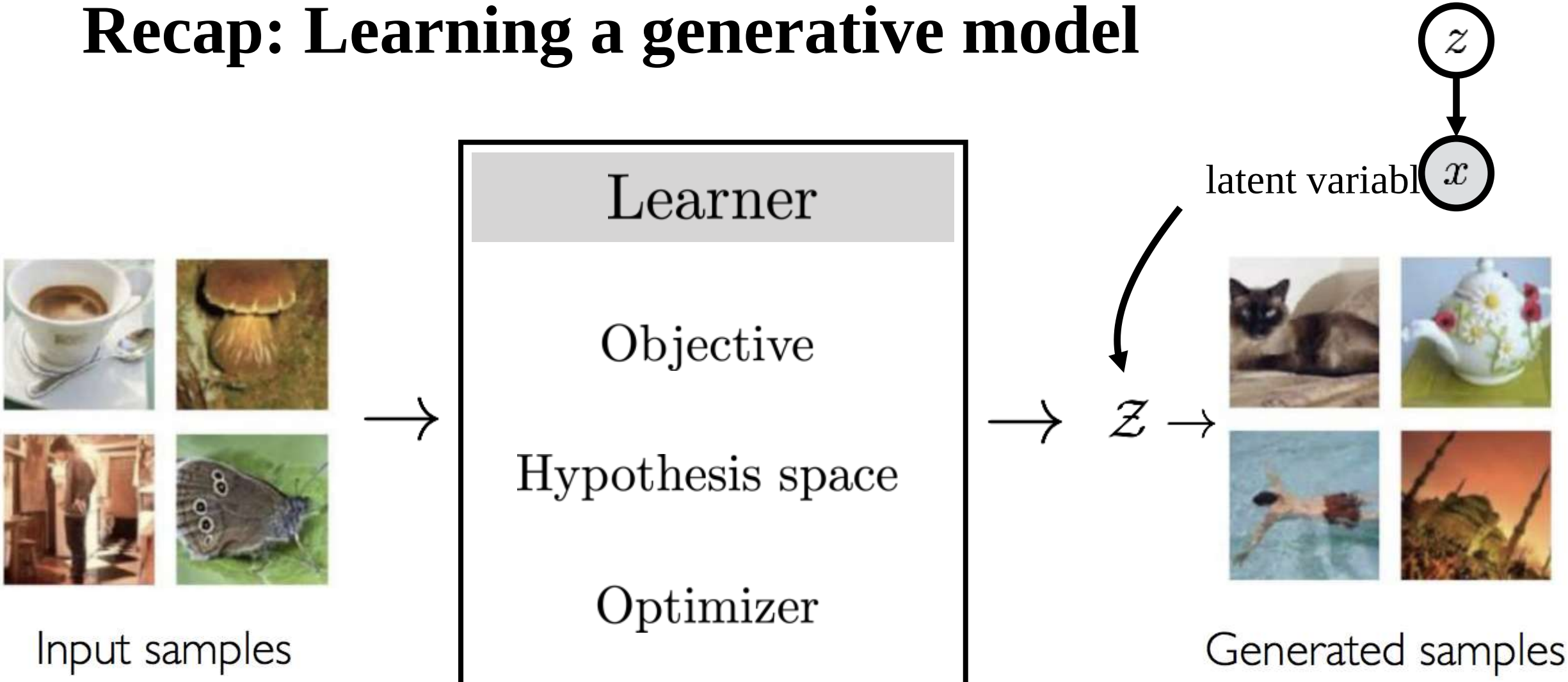
$$x = G(z)$$

Recap: Deep Image Prior

Ulyanov, Dmitry, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

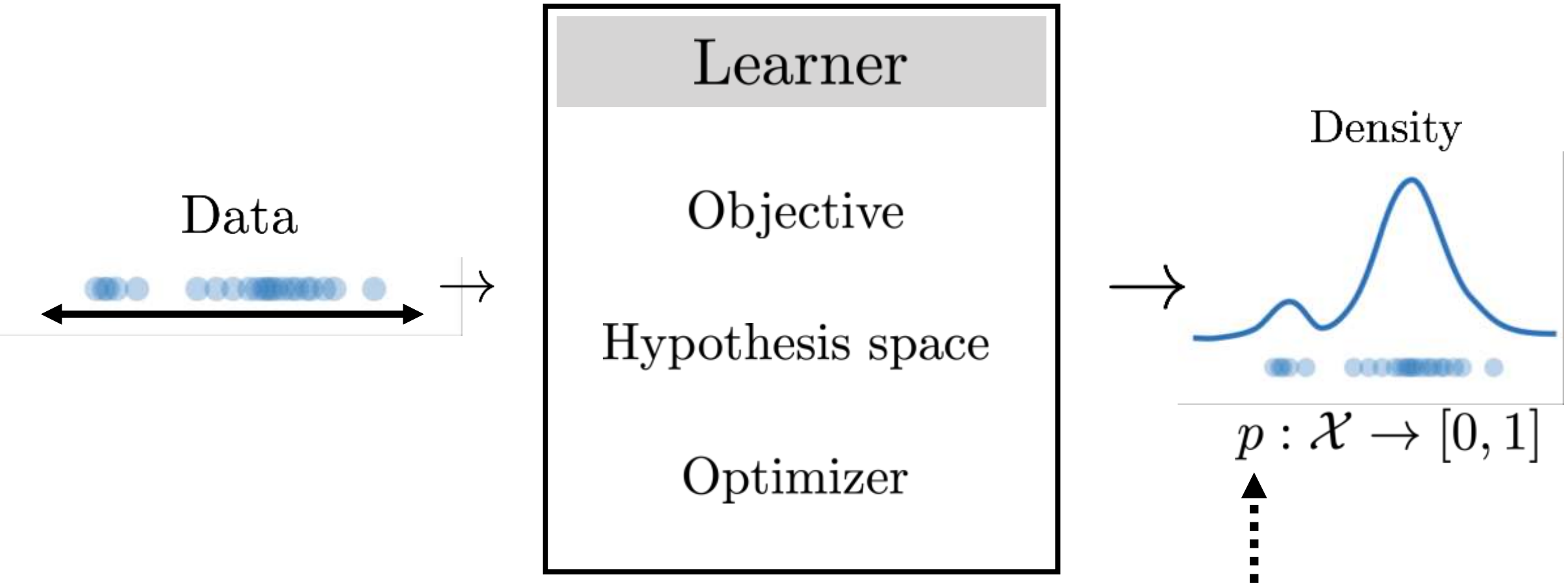


Recap: Learning a generative model



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

Recap: Learning a density model



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

Case study #1: Fitting a Gaussian to data

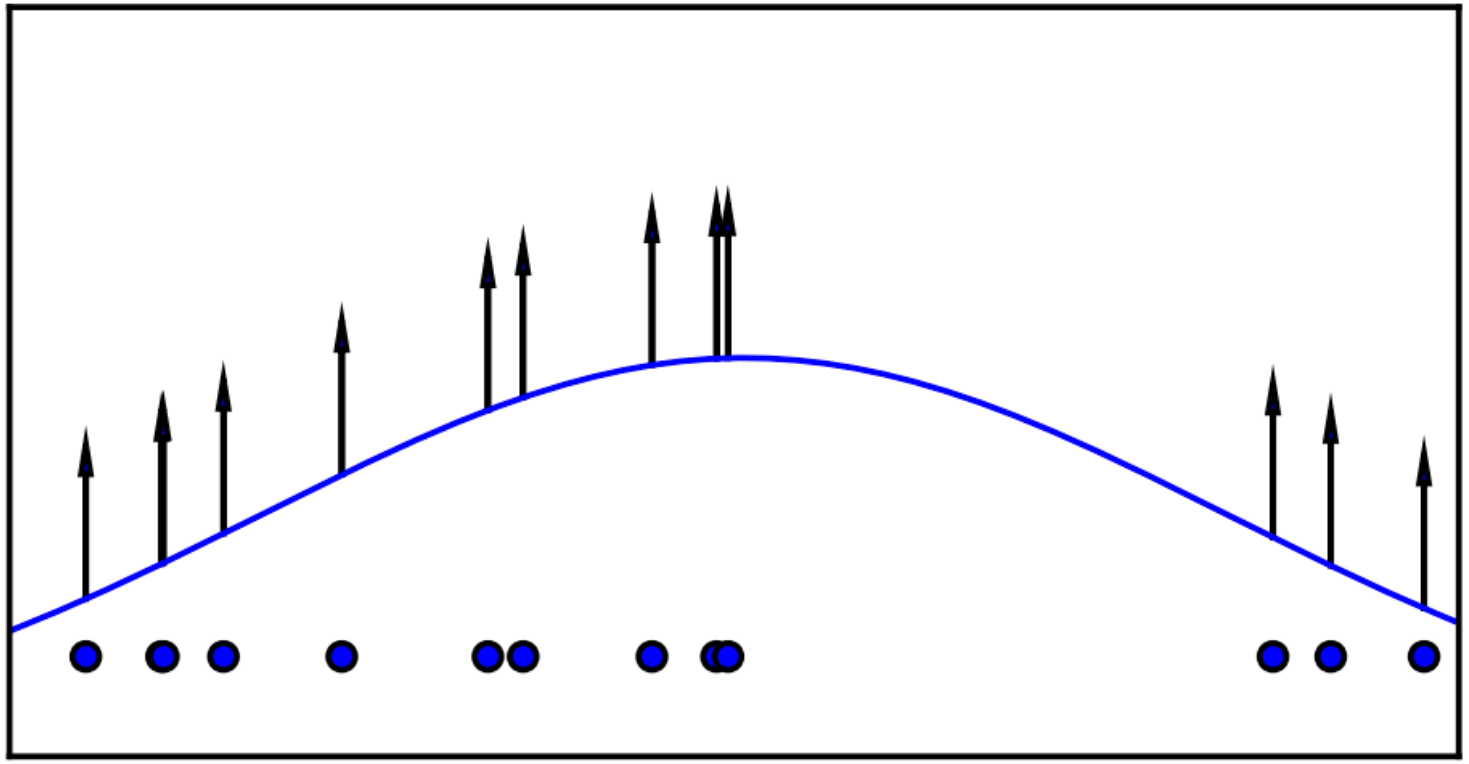


fig from [Goodfellow, 2016]

Max likelihood objective

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

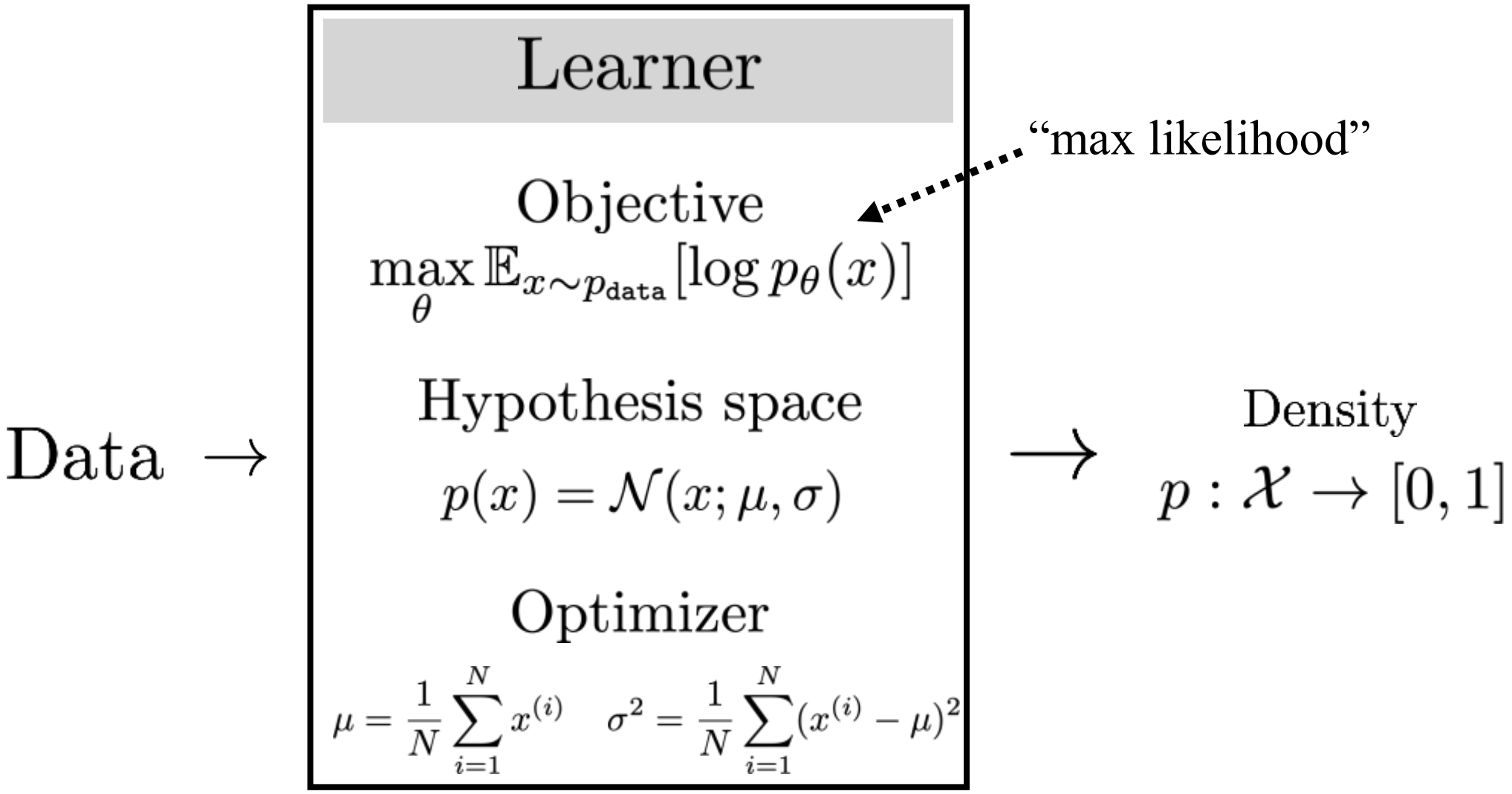
Considering only Gaussian fits

$$p_\theta(x) = \mathcal{N}(x; \mu, \sigma)$$
$$\theta = [\mu, \sigma]$$

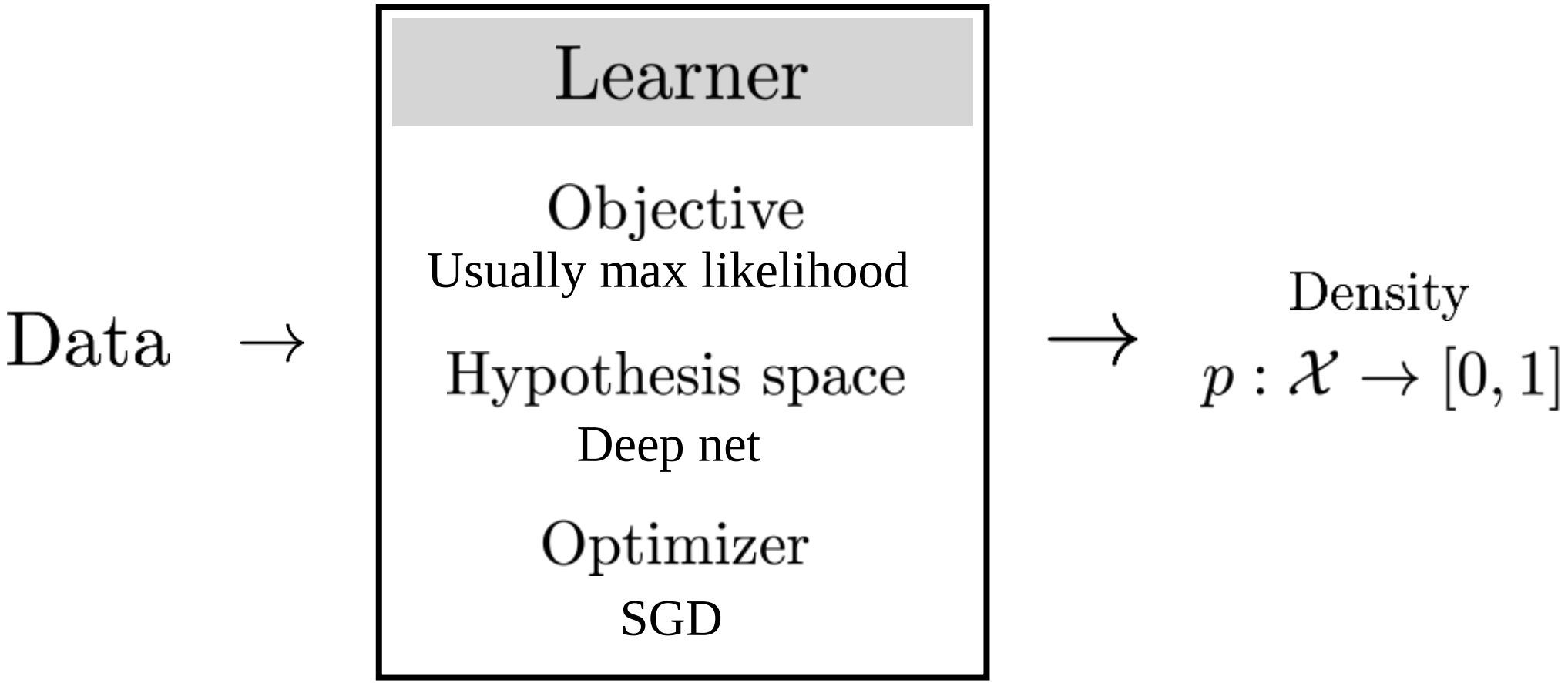
Closed form optimum:

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

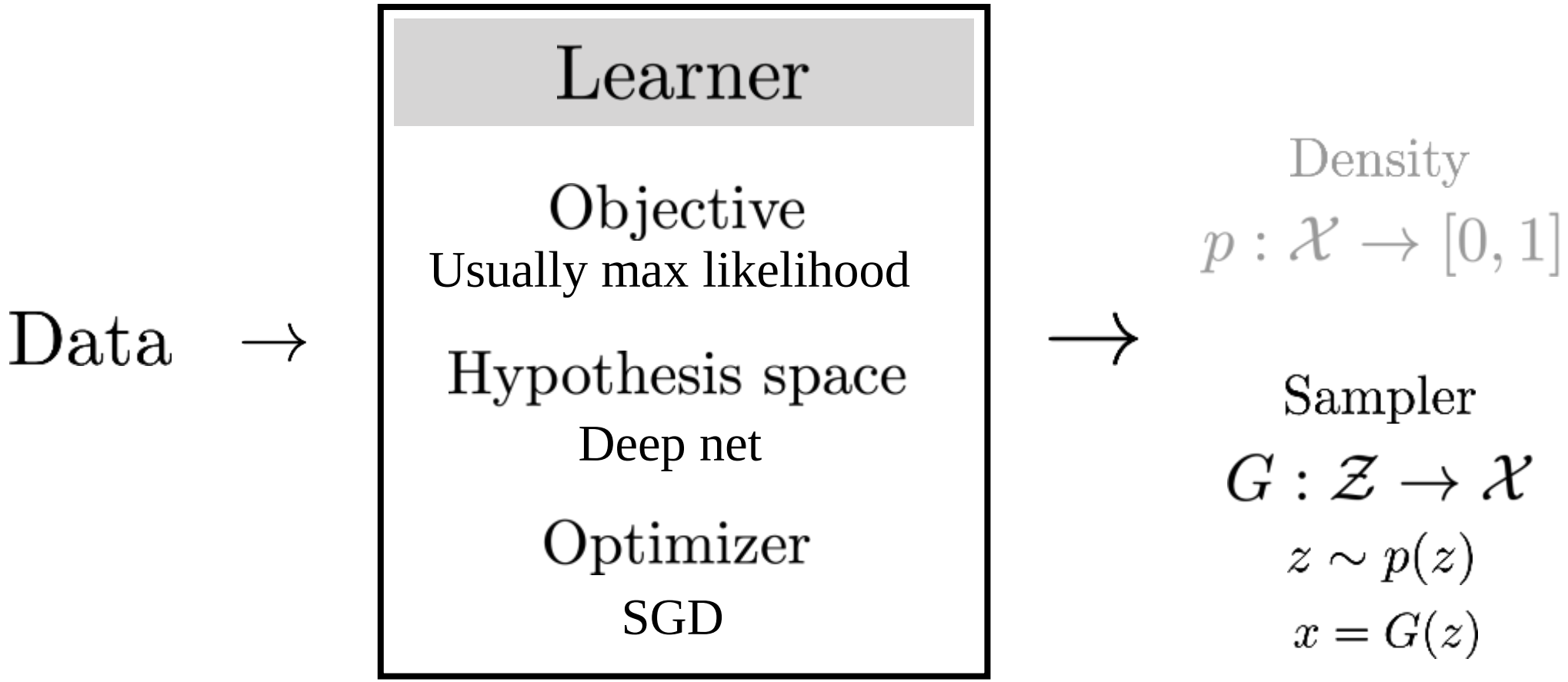
Case study #1: Fitting a Gaussian to data



Case study #2: learning a deep generative model

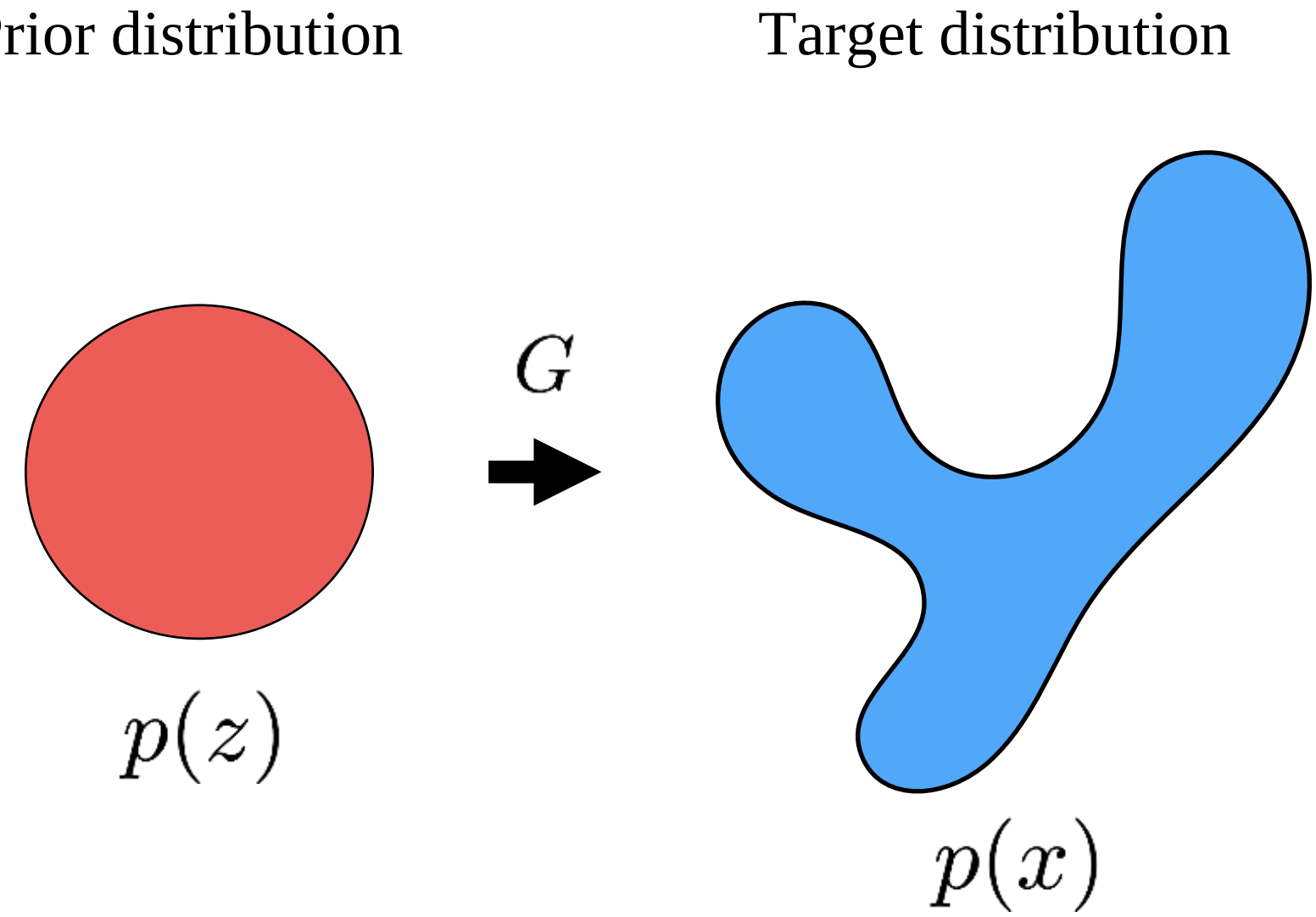


Case study #2: learning a deep generative model

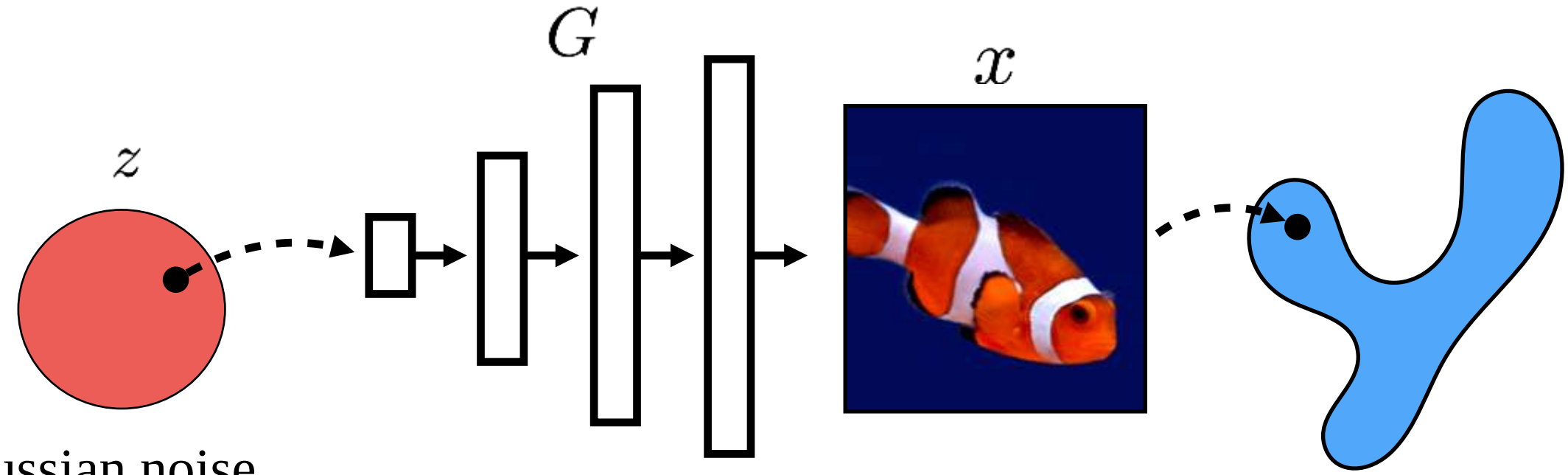


Models that provide a sampler but no density are called **implicit generative models**

Deep generative models are distribution transformers



Deep generative models are distribution transformers

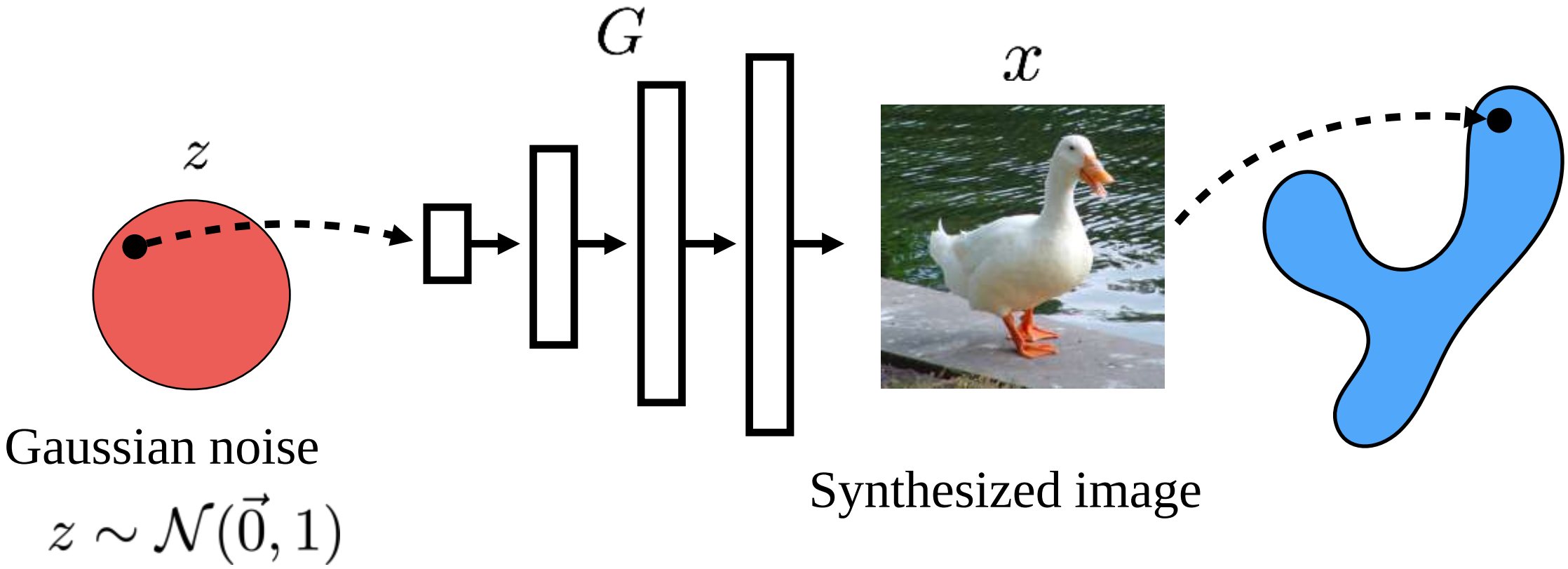


Gaussian noise

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

Synthesized image

Deep generative models are distribution transformers

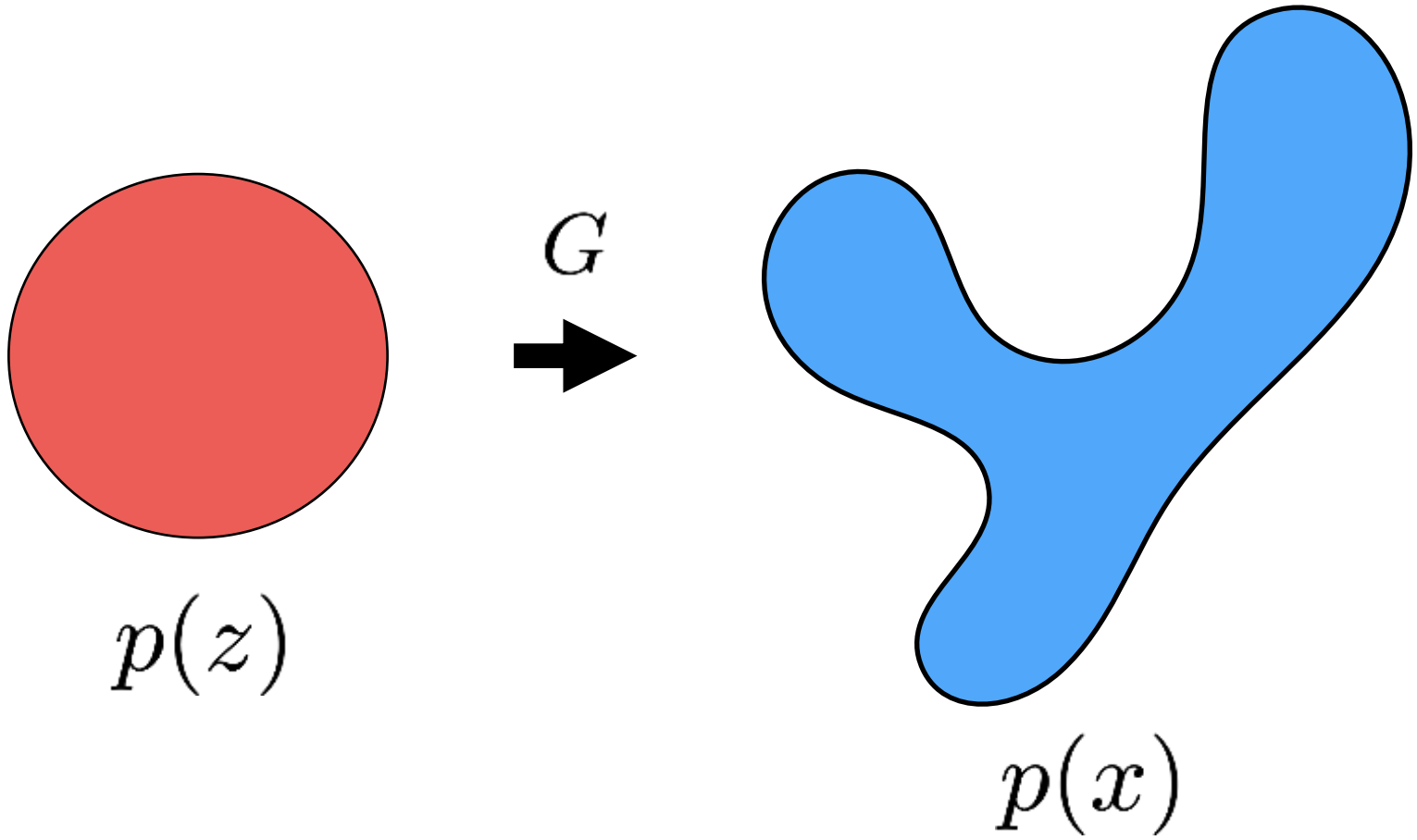


Variational Autoencoders (VAEs)

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

Prior distribution

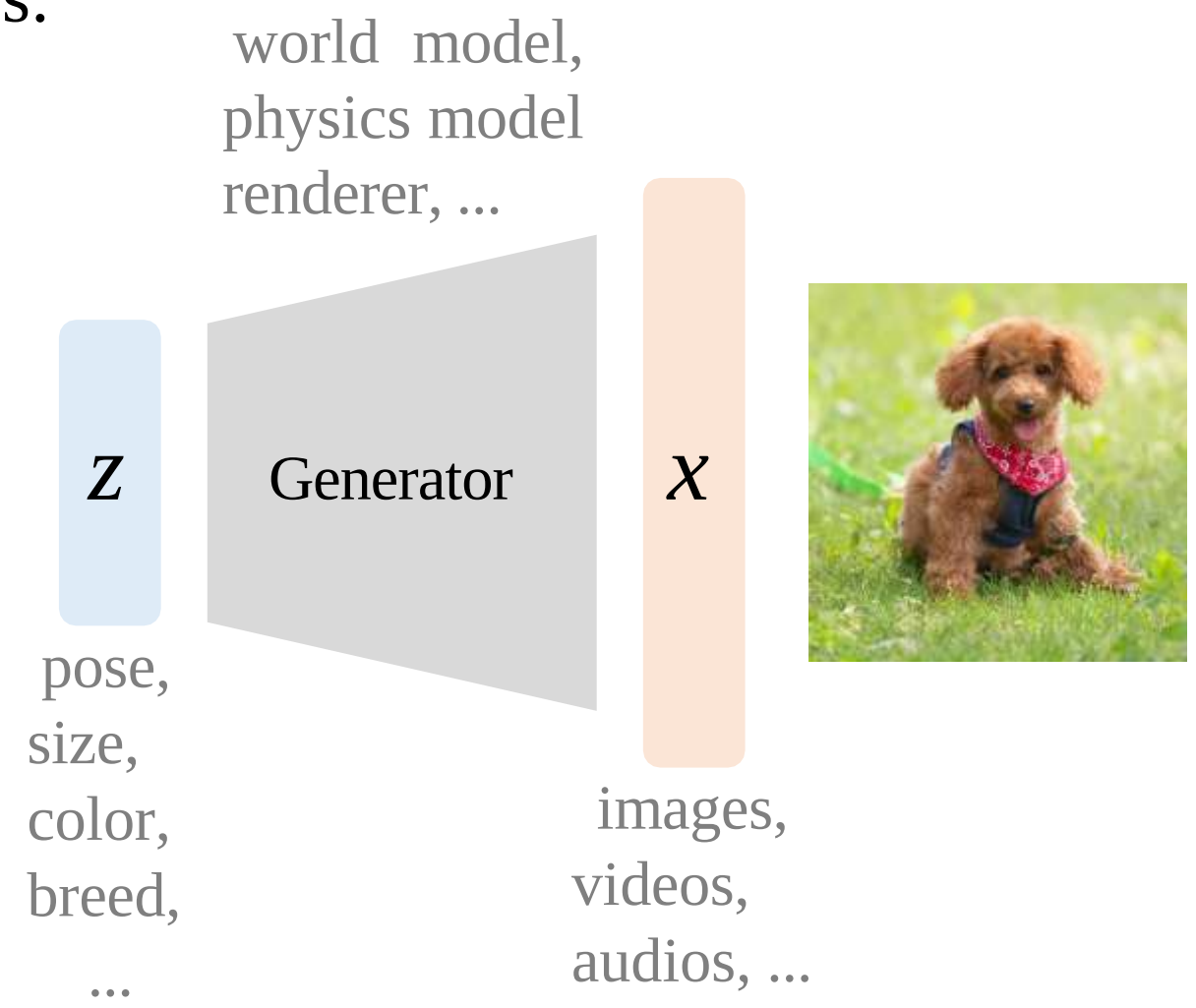
Target distribution



Latent Variable Models

Assuming a data generation process:

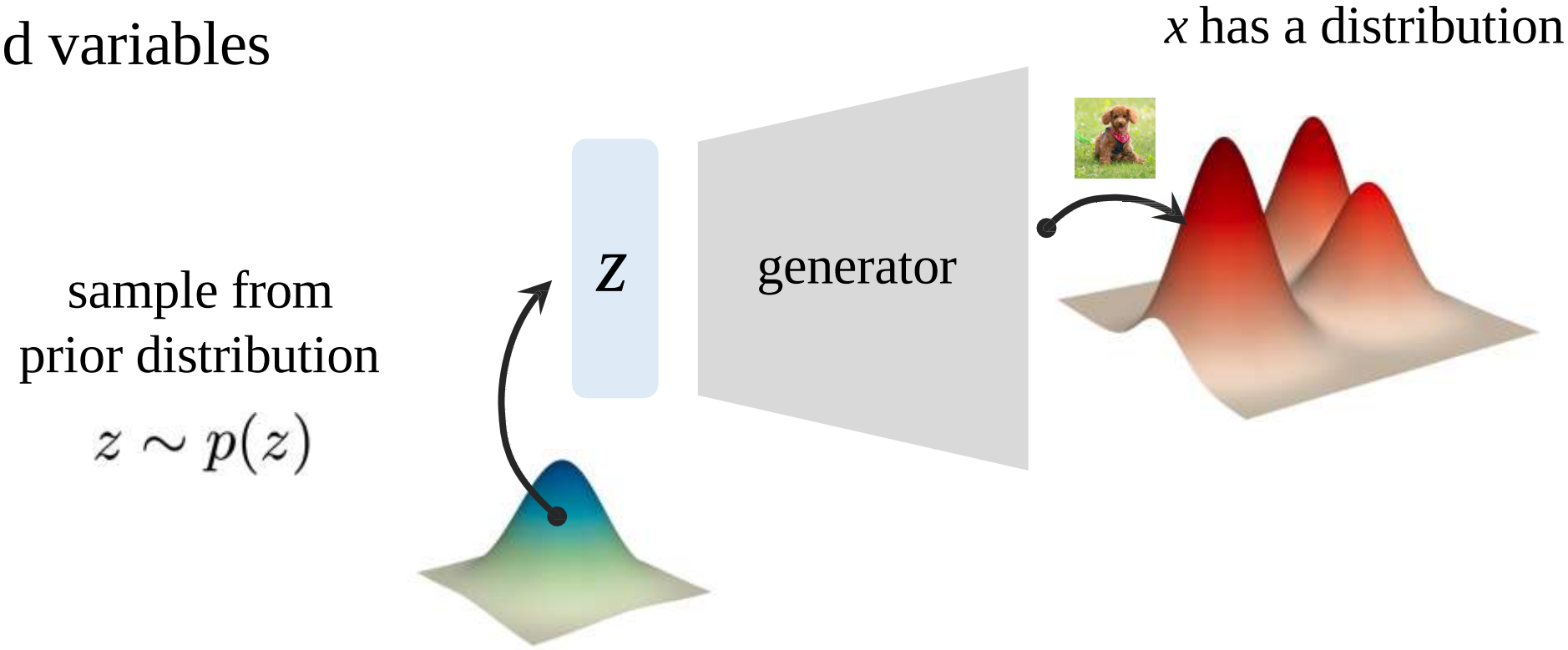
- z - latent variables
- x - observed variables



Latent Variable Models

Assuming a data generation process:

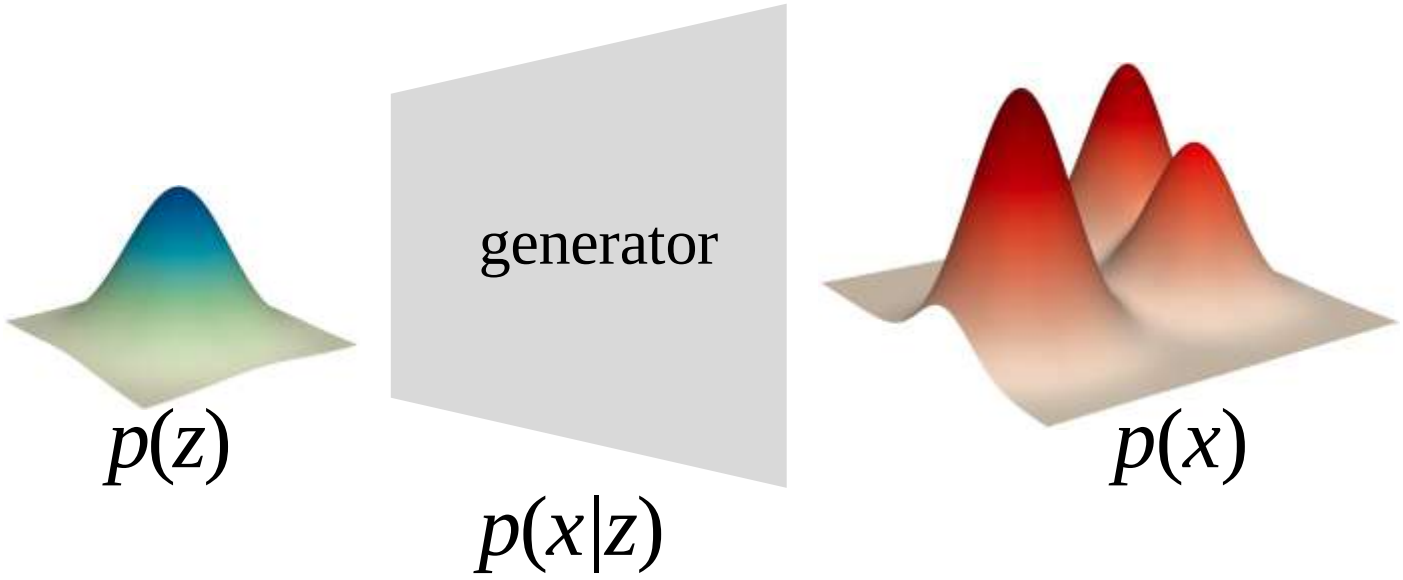
- z - latent variables
- x - observed variables



Latent Variable Models

Assuming a data generation process:

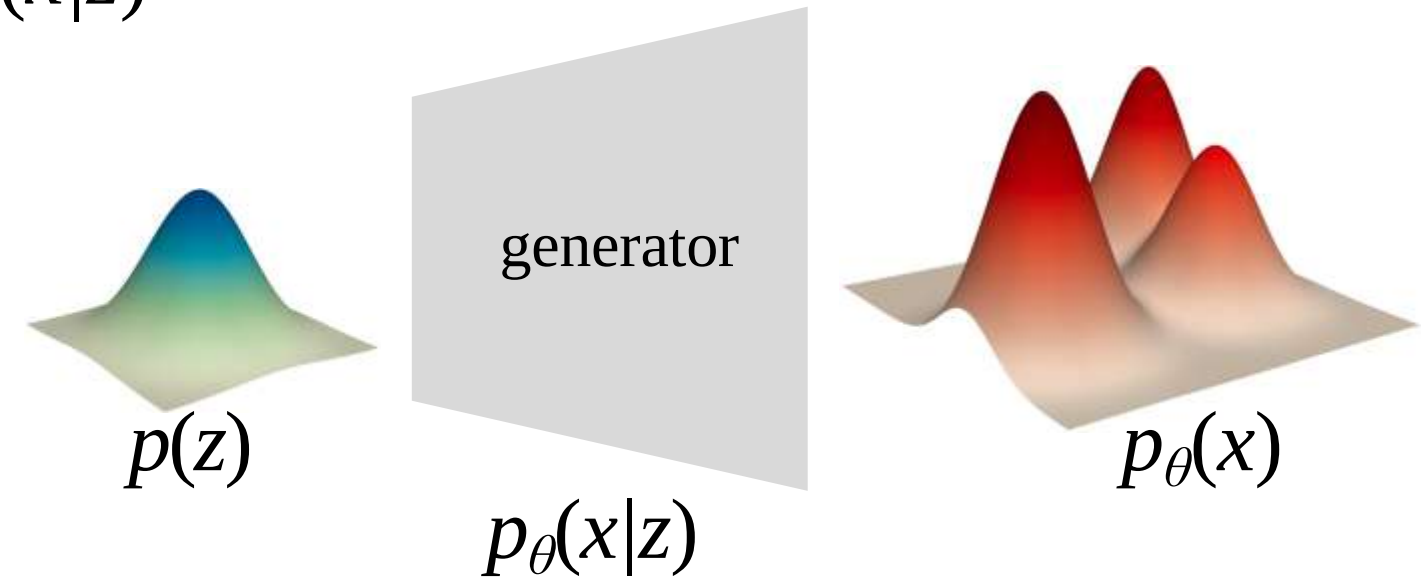
- z - latent variables
- x - observed variables



Latent Variable Models

Represent a distribution by a neural network

- θ - learnable parameters
- represent a function: $p_\theta(x|z)$



Measuring how good a distribution is ...

Minimize Kullback–Leibler (KL) divergence:

$$\min_{\theta} \mathcal{D}_{\mathrm{KL}}(p_{data} \,||\, p_\theta)$$

Note: consider other criteria than KL?

⇒ Maximize likelihood:

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

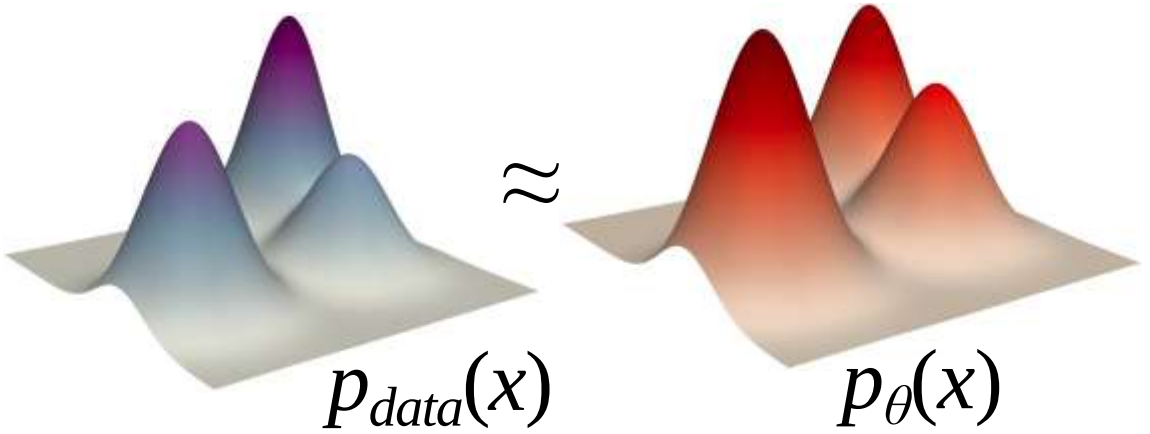
$$\arg\min_{\theta} \mathcal{D}_{\mathrm{KL}}(p_{data} \,||\, p_\theta) \qquad \text{tl; dr}$$

$$= \arg\min_{\theta} \sum_x p_{data}(x) \log \frac{p_{data}(x)}{p_\theta(x)}$$

$$= \arg\min_{\theta} \sum_x -p_{data}(x) \log p_\theta(x) + const$$

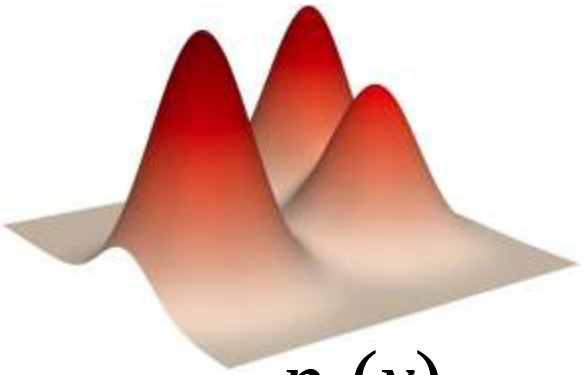
$$= \arg\max_{\theta} \sum_x p_{data}(x) \log p_\theta(x)$$

$$= \arg\max_{\theta} \mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$$



Latent Variable Models

We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$



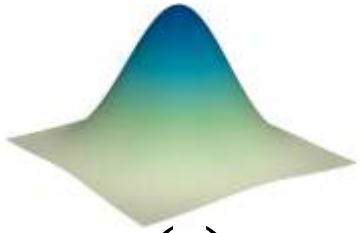
$p_\theta(x)$

Latent Variable Models

We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$

with $p_\theta(x)$ represented as:

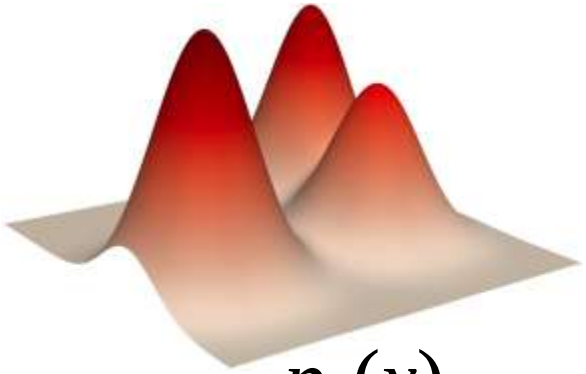
$$p_\theta(x) = \int_z p_\theta(x|z)p(z)dz$$



$p(z)$

generator

$p_\theta(x|z)$



$p_\theta(x)$

Latent Variable Models

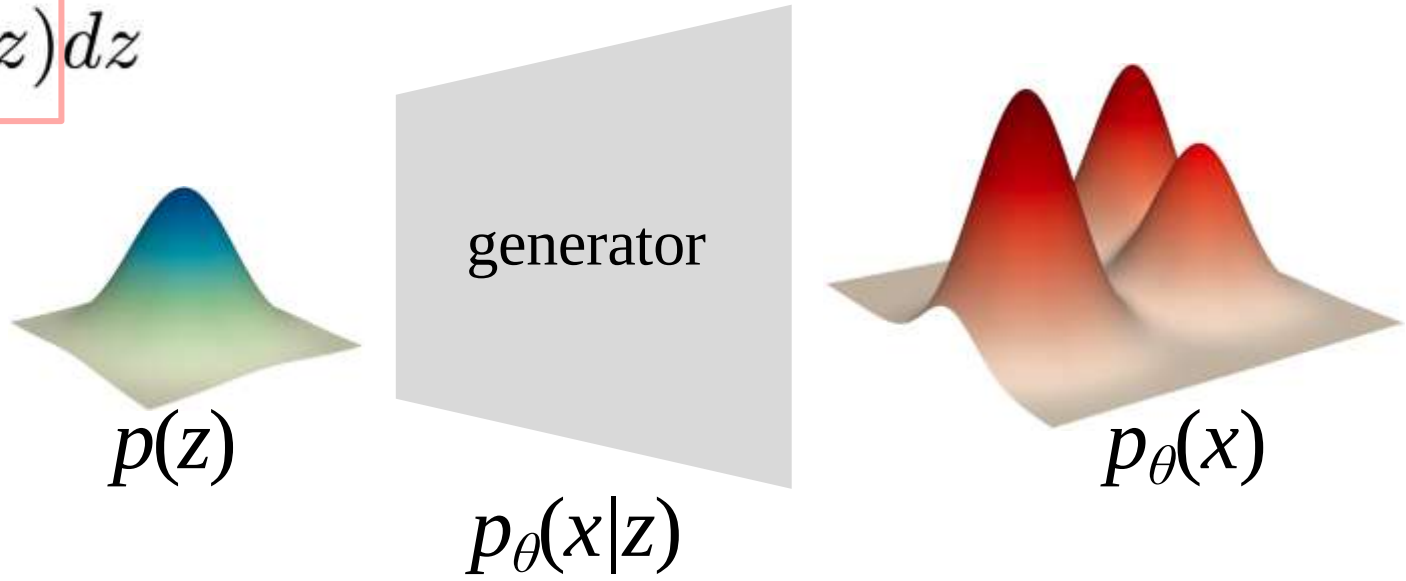
We want to maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$

with $p_\theta(x)$ represented as:

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

Two sets of unknowns:

- We need to optimize for θ
- We can't control "**true**" $p(z)$



Idea: introduce a "controllable" distribution $q(z)$

Latent Variable Models

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \Big(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \Big) dz$$

$$= \mathbb{E}_{z \sim q(z)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z) \Big) + \mathcal{D}_{\mathrm{KL}} \Big(q(z) || p_\theta(z|x) \Big)$$

Rewrite log likelihood by latent *z*

- for any distribution $q(z)$
- Bayes' rule

Latent Variable Models

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Rewrite log likelihood by latent *z*

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Latent Variable Models

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Latent Variable Models

$$\log p_\theta(x)$$

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Latent Variable Models

$$
\begin{aligned}
& \log p_\theta(x) \\
= & \int_z q(z) \log p_\theta(x) dz \\
= & \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz \\
= & \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz \\
= & \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz \\
= & \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)
\end{aligned}
$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule
- just algebra
- just algebra

Latent Variable Models

intractable

$$\log p_\theta(x)$$

Rewrite log likelihood by latent z

- for any distribution $q(z)$
- Bayes' rule

$$= \int_z q(z) \log p_\theta(x) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \right) dz$$

$$= \int_z q(z) \log \left(\frac{p_\theta(x|z) p_\theta(z)}{p_\theta(z|x)} \frac{q(z)}{q(z)} \right) dz$$

$$= \int_z q(z) \left(\log p_\theta(x|z) + \log \frac{p_\theta(z)}{q(z)} + \log \frac{q(z)}{p_\theta(z|x)} \right) dz$$

$$= \mathbb{E}_{z \sim q(z)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z) \right) + \mathcal{D}_{\mathrm{KL}} \left(q(z) || p_\theta(z|x) \right)$$

tractable tractable intractable

Latent Variable Models

intractable $\log p_\theta(x) - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z|x)\right)$ intractable

$= \mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)$

tractable tractable

Latent Variable Models

intractable $\log p_\theta(x) - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z|x)\right)$ intractable

$= \underbrace{\mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)}$

tractable tractable

- This is called Evidence Lower Bound (ELBO)
- Lower bound of $\log p_\theta(x)$

- This equation holds for <u>any</u> distribution $q(z)$

Latent Variable Models

$q_\varphi(z|x)$ $\qquad\qquad$ $q_\varphi(z|x)$

$$\mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)$$

tractable $\qquad$ tractable

- This is called Evidence Lower Bound (ELBO)
- Lower bound of $\log p_\theta(x)$

- This equation holds for <u>any</u> distribution $q(z)$

- Parameterize $q(z)$ by $q_\varphi(z|x)$

Latent Variable Models

$q_\varphi(z|x)$ $\quad$ $q_\varphi(z|x)$ $\quad$ $p(z)$

$$\mathbb{E}_{z\sim q(z)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q(z)||p_\theta(z)\right)$$

tractable $\quad$ tractable

- This is called Evidence Lower Bound (ELBO)
- Lower bound of $\log p_\theta(x)$

- This equation holds for <u>any</u> distribution $q(z)$

- Parameterize $q(z)$ by $q_\varphi(z|x)$
- let $p_\theta(z)$ be a simple known prior $p(z)$

Variational Autoencoder

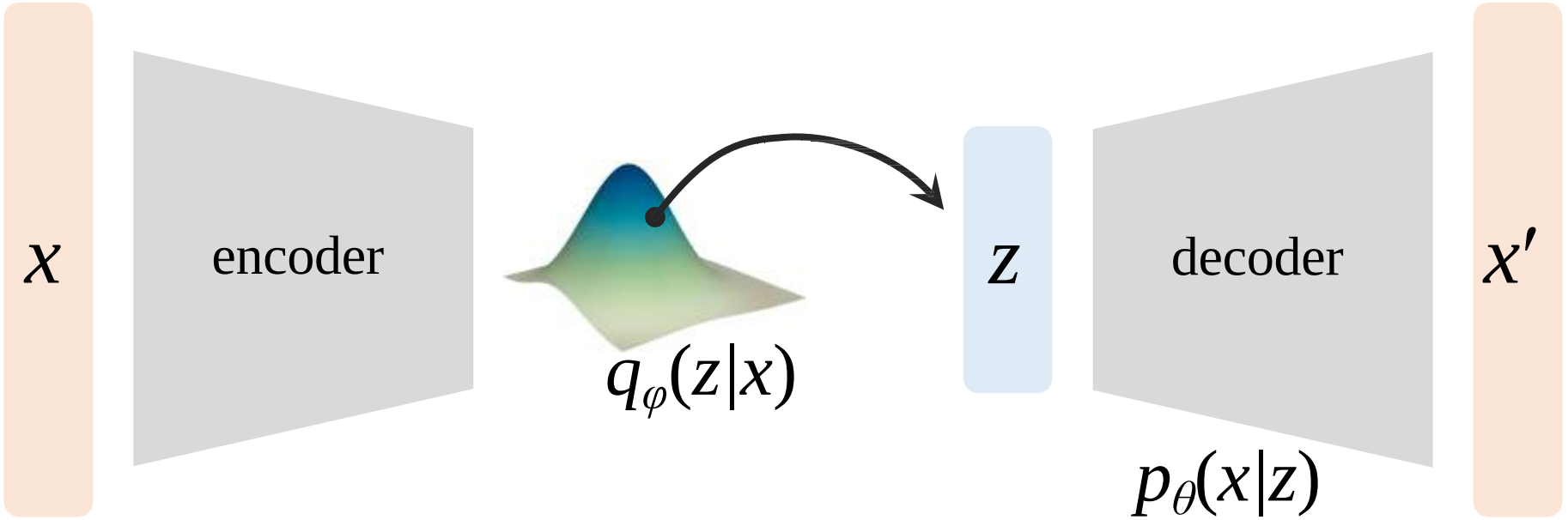
Maximize ELBO ⇒ minimize an objective:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

Variational Autoencoder

Maximize ELBO ⇒ minimize an objective:

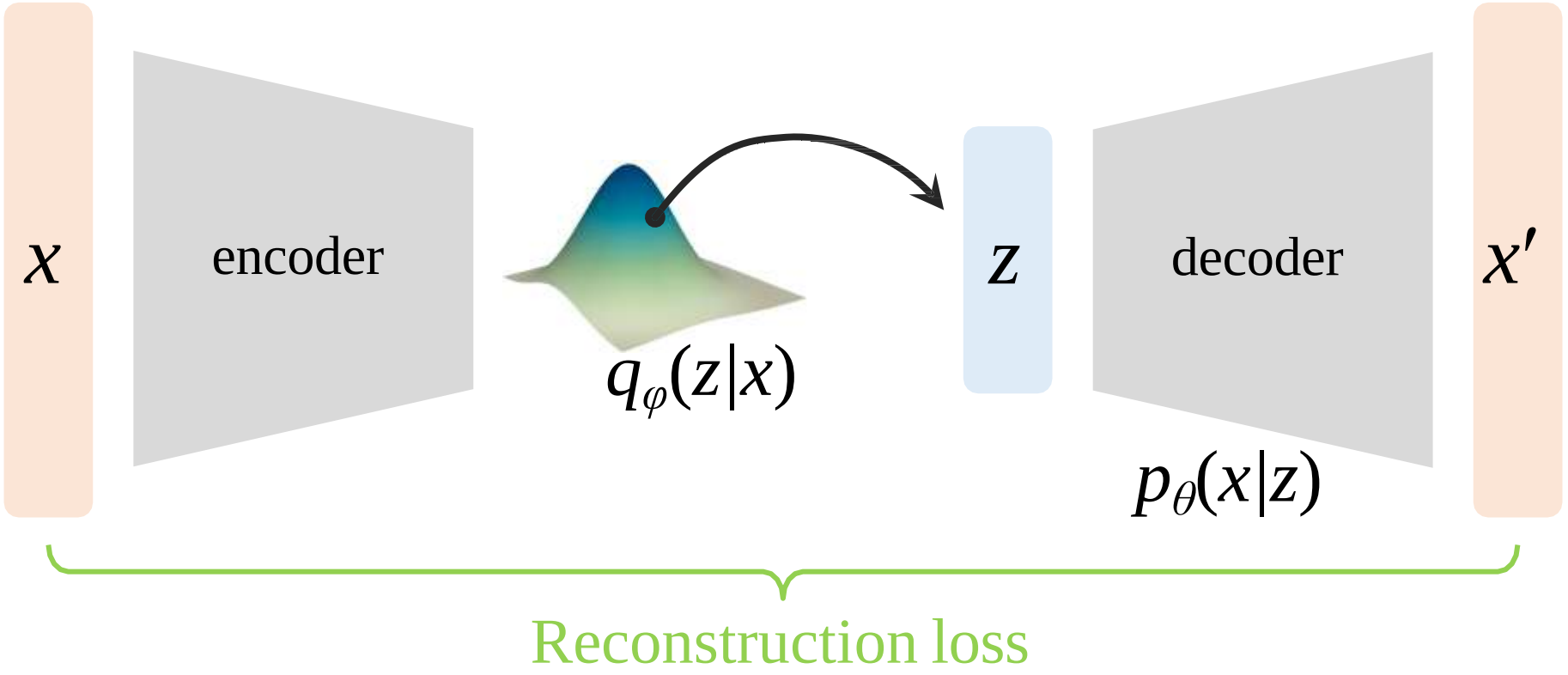
$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$



Variational Autoencoder

Maximize ELBO ⇒ minimize an objective:

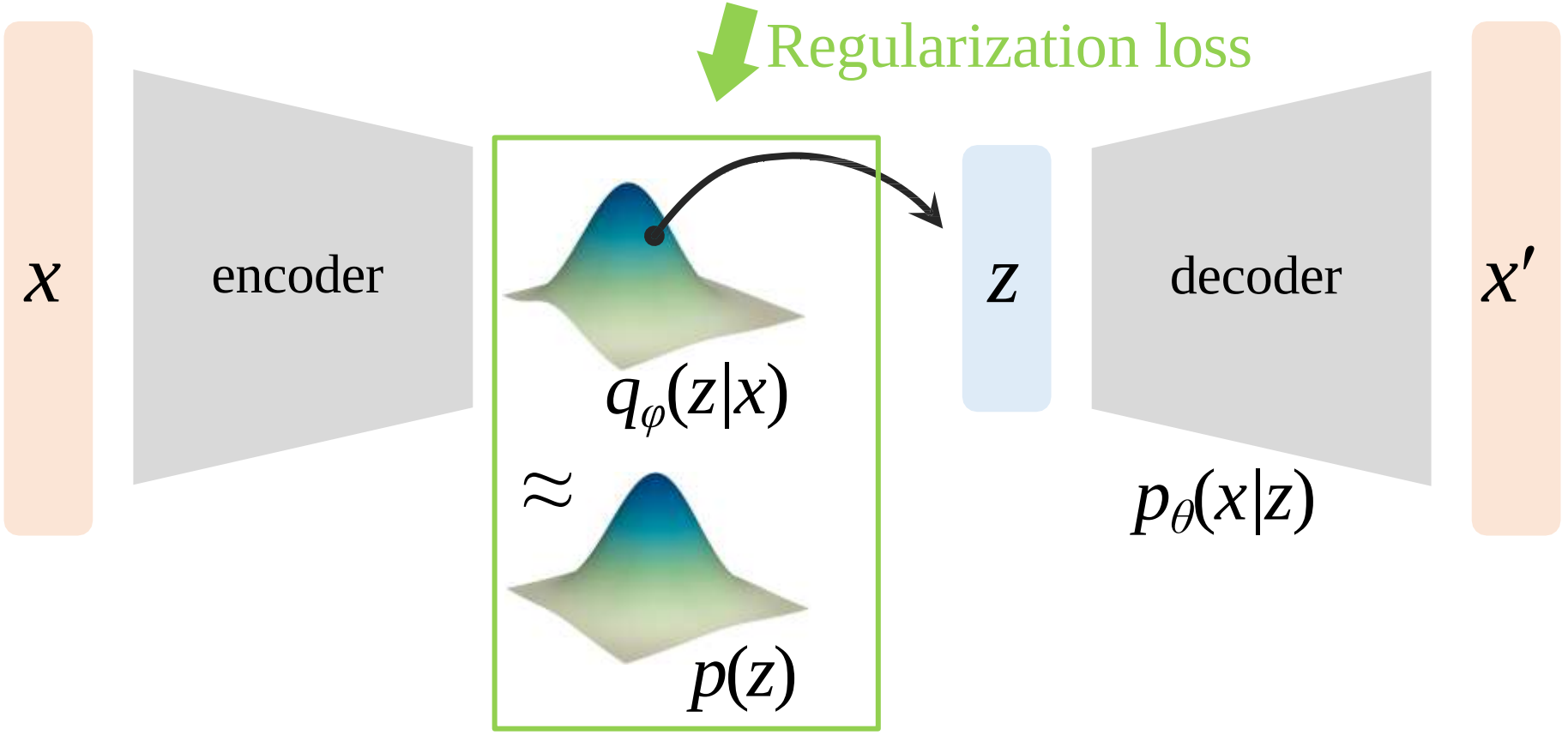
$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$



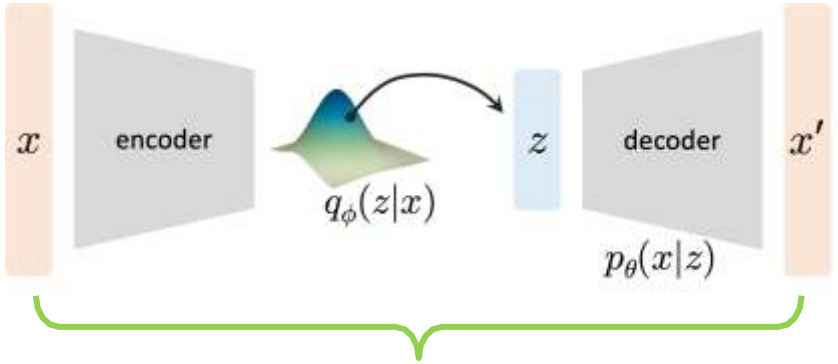
Variational Autoencoder

Maximize ELBO ⇒ minimize an objective:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$



Variational Autoencoder



Reconstruction loss

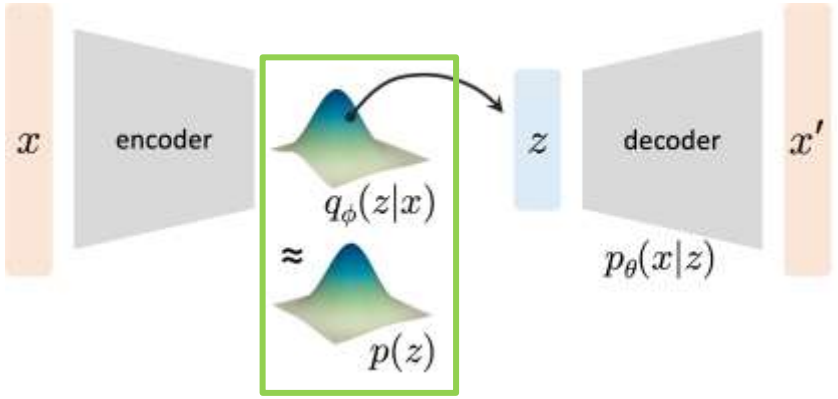
$$\mathcal{L}_{\theta,\phi}(x) = \boxed{-\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big]} + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

Example: L2 loss

- one-step Monte Carlo: $z \sim q_\phi(z|x)$
- map *z* by decoder net: $\boxed{g_\theta(z) \to x'}$ network estimates distribution's parameters
- model $p_\theta(x|z)$ by Gaussian: $p_\theta(x|z) = \mathcal{N}(x \mid x', \sigma_0^2)$ (assume fixed std)
- negative log likelihood: $\frac{1}{2\sigma_0^2}||x - x'||^2 + const$

- L2 loss ⇒ a Gaussian neighborhood around data point *x*

Variational Autoencoder



$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] + \mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$$

Regularization loss

Example: Gaussian prior

- let $p(z) = \mathcal{N}(z \mid 0, \mathbf{I})$
- model $q_\phi(z|x)$ by Gaussian: $\mathcal{N}(z \mid \mu, \sigma)$
- map *x* by encoder net: $f_\phi(x) \rightarrow \mu, \sigma$

again, network estimates distribution's parameters

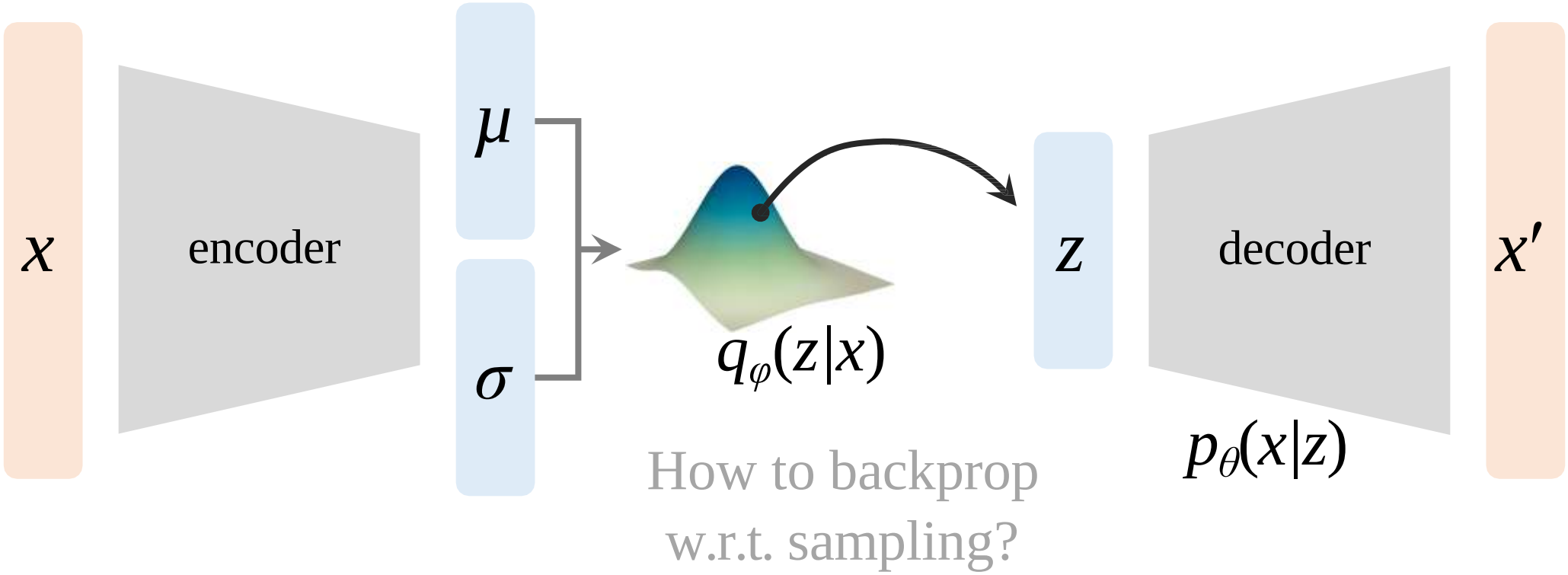
- compute loss analytically: $\mathcal{D}_{\mathrm{KL}}\left(\mathcal{N}(z \mid \mu, \sigma) \,||\, \mathcal{N}(z \mid 0, \mathbf{I})\right)$

- fixed covariance ⇒ L2 loss on μ

Variational Autoencoder

Maximize ELBO ⇒ minimize an objective:

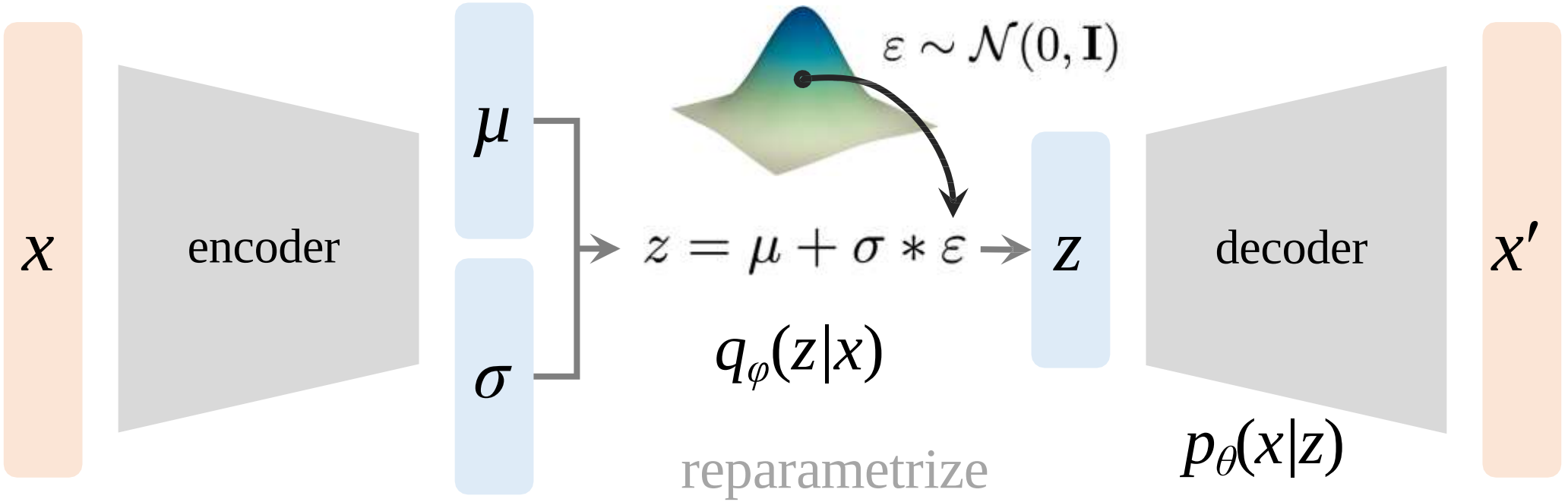
$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$



Variational Autoencoder

Maximize ELBO ⇒ minimize an objective:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$



Variational Autoencoder

... so far, we have discussed an objective on one x:

$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

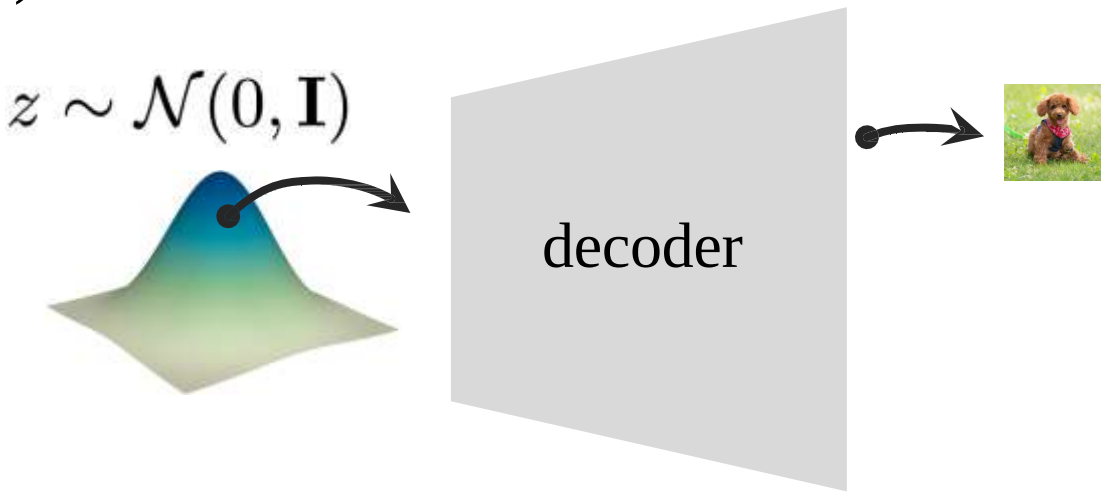
Overall loss is expectation over data:

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{x\sim p_{data}(x)}\Big[-\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)\Big]$$

Variational Autoencoder

Inference (generation):

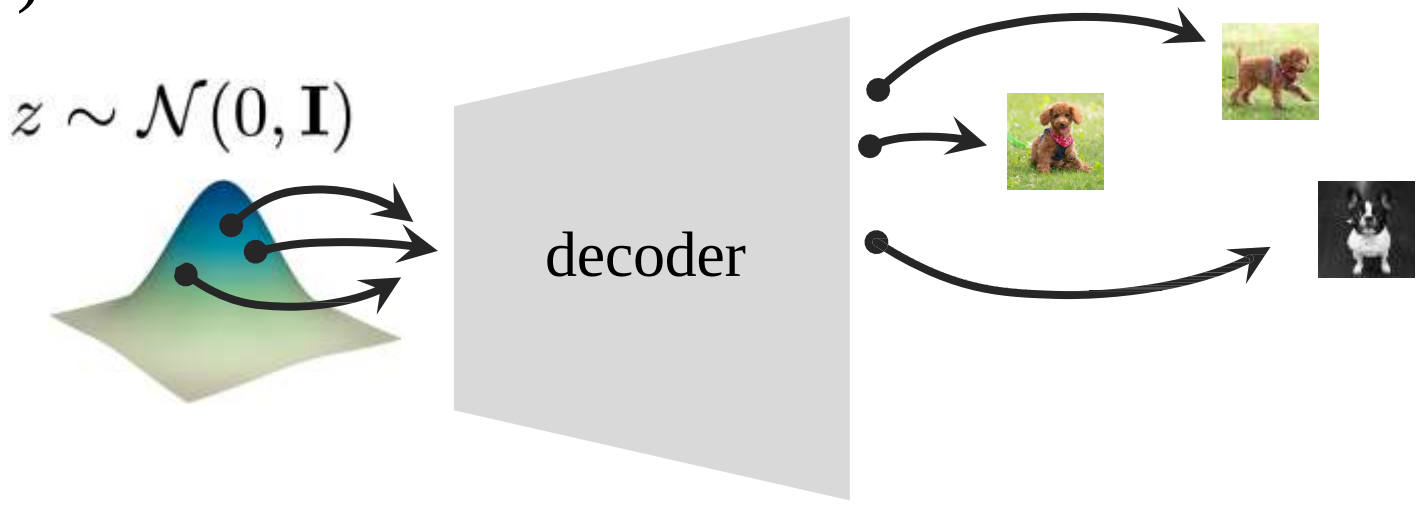
- sample z from: $\mathcal{N}(0, \mathbf{I})$
- map z by decoder net: $g_\theta(z)$



Variational Autoencoder

Inference (generation):

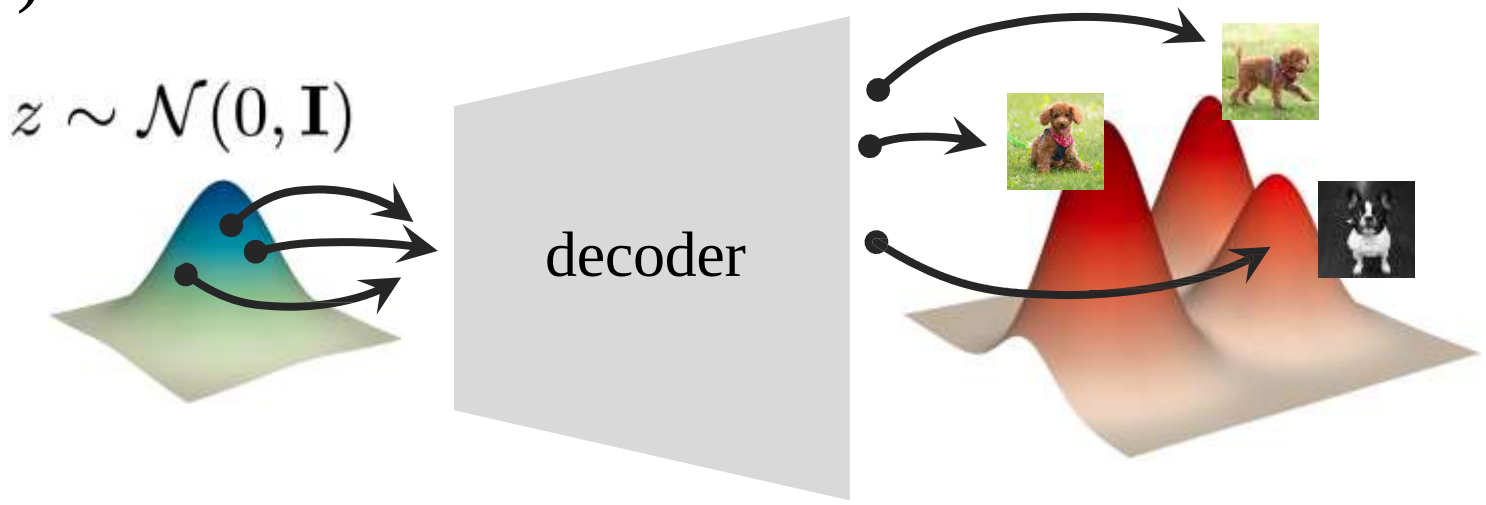
- sample z from: $\mathcal{N}(0, \mathbf{I})$
- map z by decoder net: $g_\theta(z)$



Variational Autoencoder

Inference (generation):

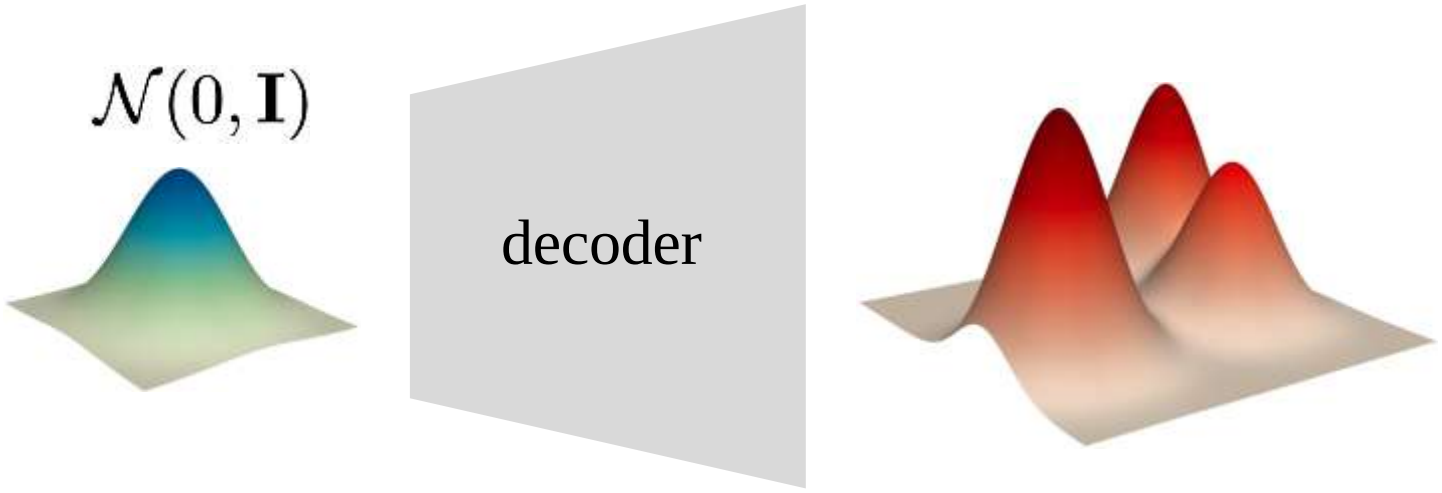
- sample z from: $\mathcal{N}(0, \mathbf{I})$
- map z by decoder net: $g_\theta(z)$



Variational Autoencoder

Inference (generation):

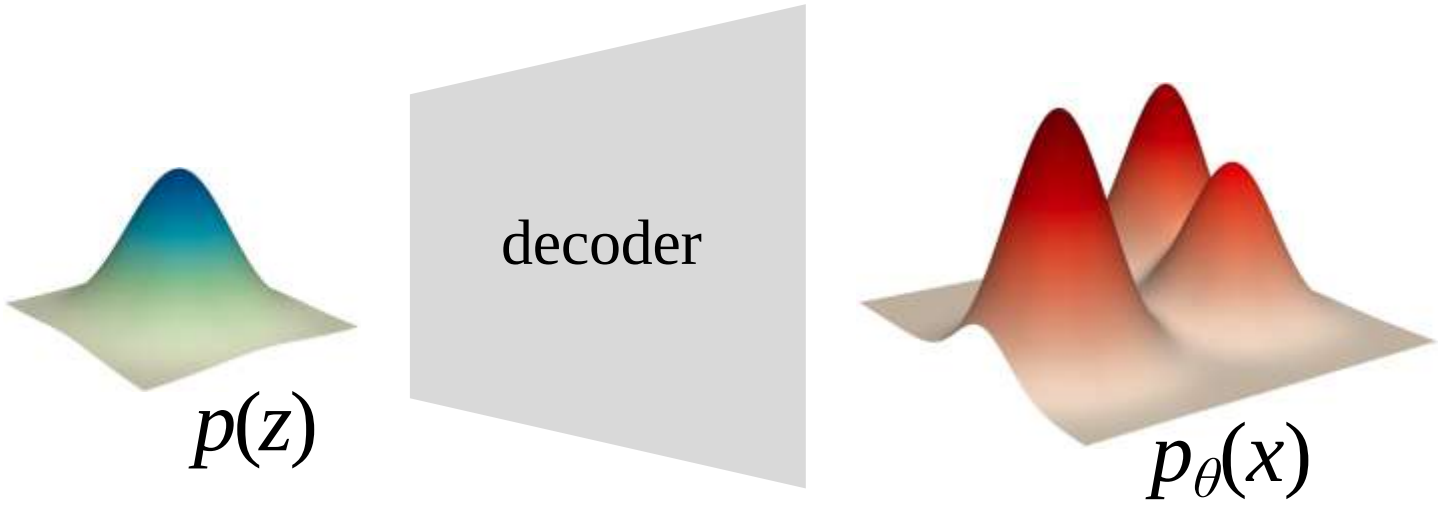
- sample z from: $\mathcal{N}(0, \mathbf{I})$
- map z by decoder net: $g_\theta(z)$



Decoder is a deterministic mapping from one distribution to another.

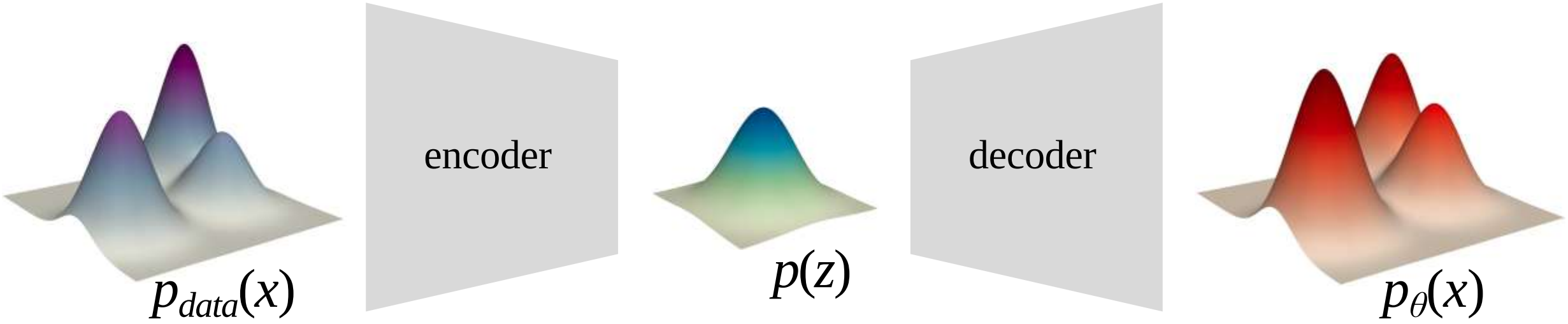
A view of “Autoencoding Distributions”

- decoder: maps latent distribution to data distribution



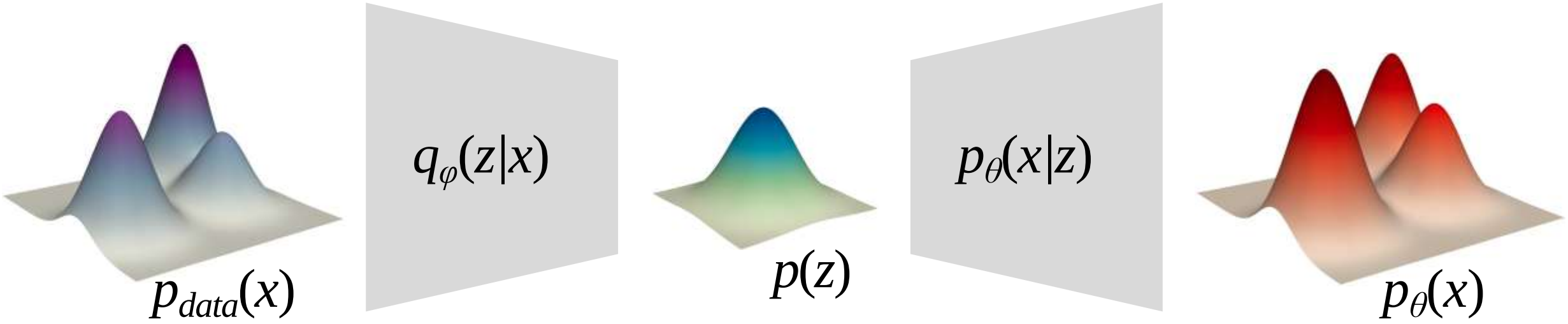
A view of “Autoencoding Distributions”

- encoder: maps data distribution to latent distribution
- decoder: maps latent distribution to data distribution



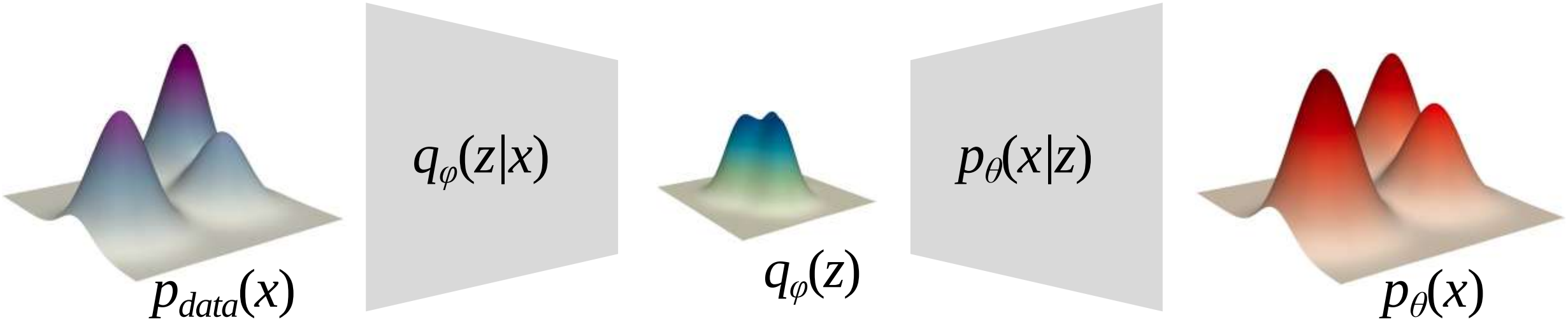
A view of “Autoencoding Distributions”

- encoder: maps data distribution to latent distribution
- decoder: maps latent distribution to data distribution



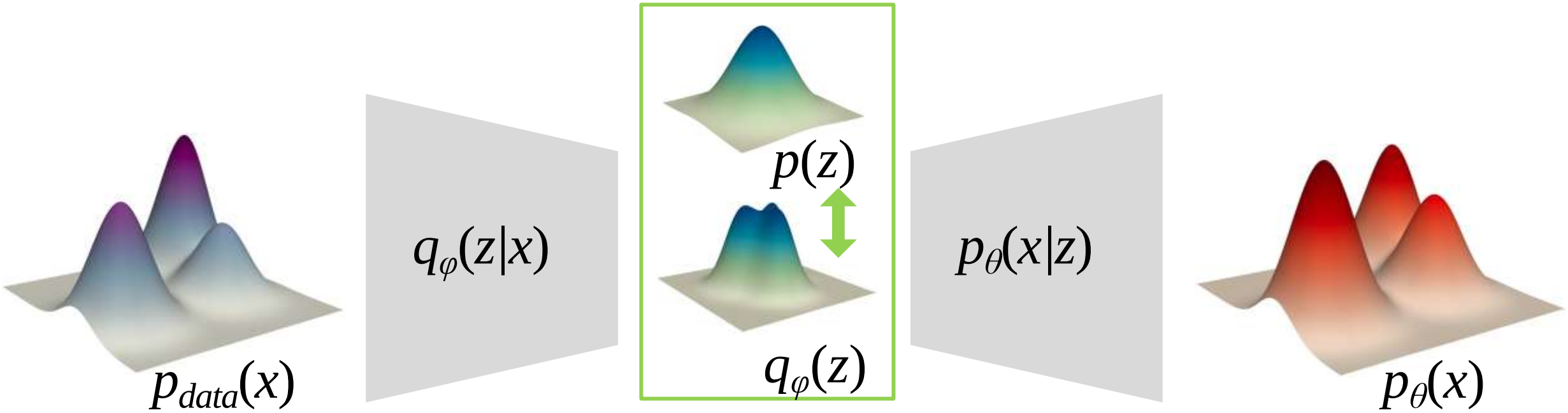
A view of “Autoencoding Distributions”

- encoded latent distribution: $q_\phi(z) = \int_x q_\phi(z|x) p_{data}(x) dx$



A view of "Autoencoding Distributions"

- encoded latent distribution: $q_\phi(z) = \int_x q_\phi(z|x) p_{data}(x) dx$

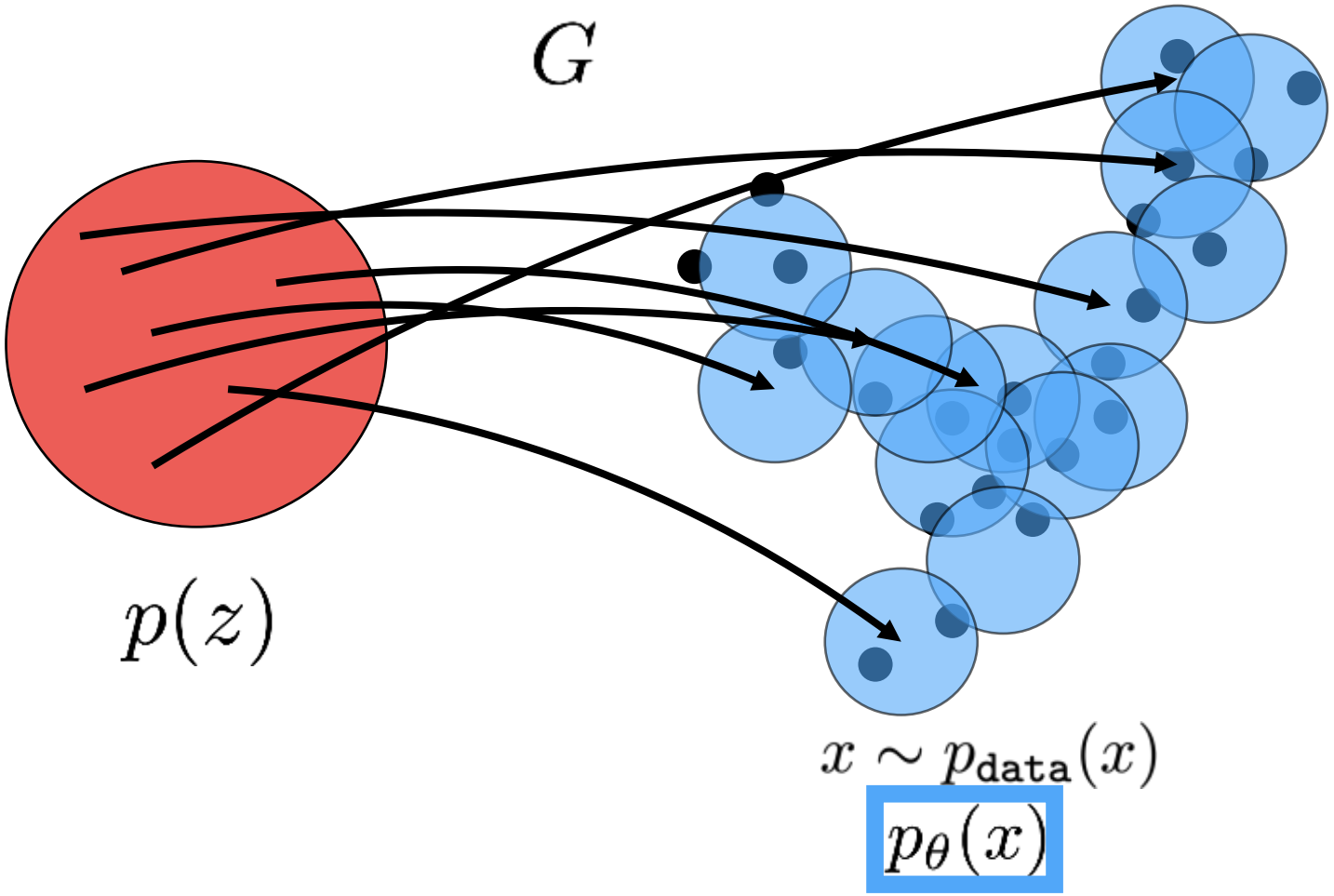


E.g., see "*InfoVAE: Information Maximizing Variational Autoencoders*", 2017

Variational Autoencoders (VAEs)

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

Prior distribution Target distribution



Density model:

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

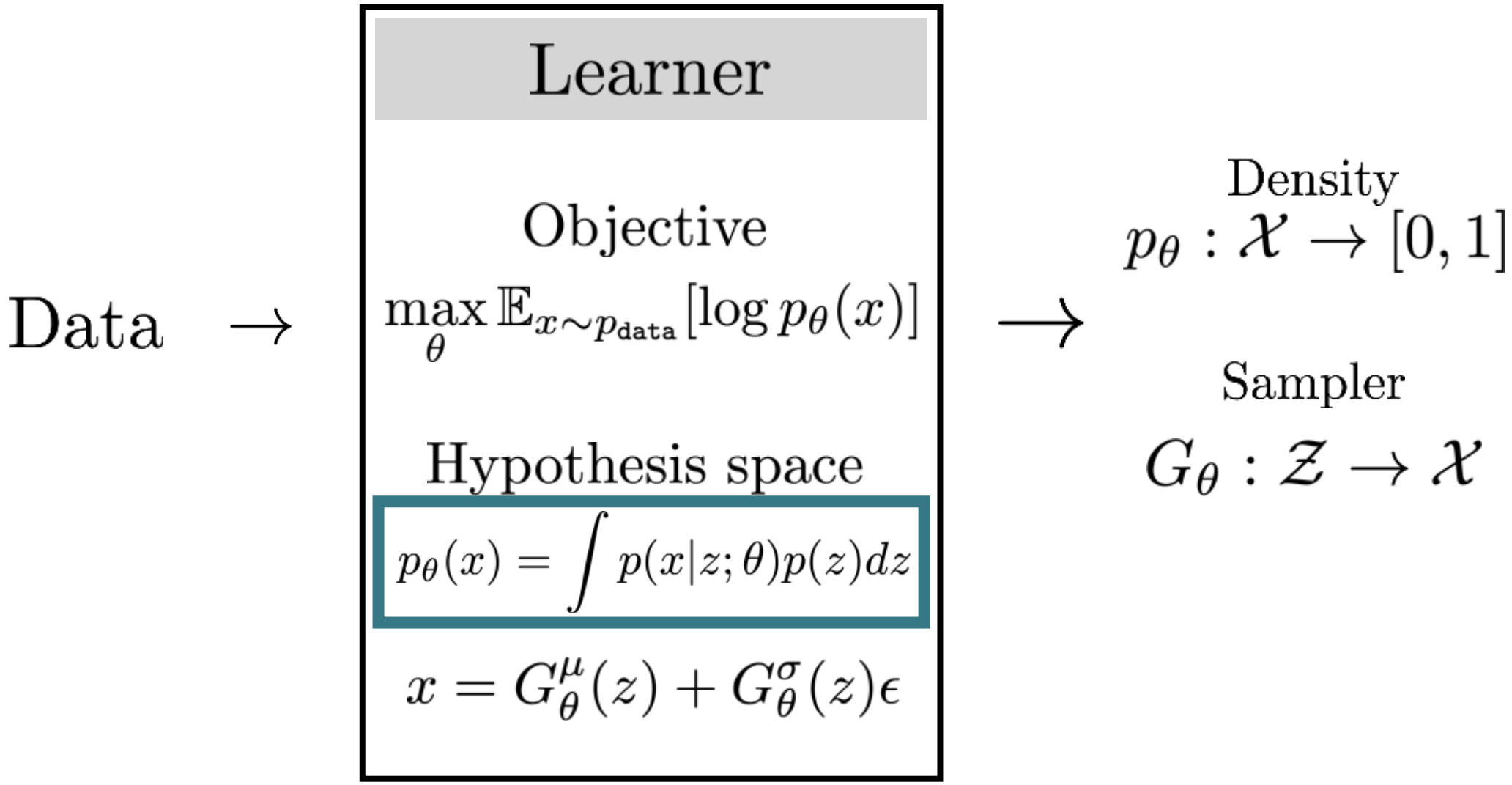
$$p(x|z;\theta) \sim \mathcal{N}(x; G^\mu_\theta(x), G^\sigma_\theta(x))$$

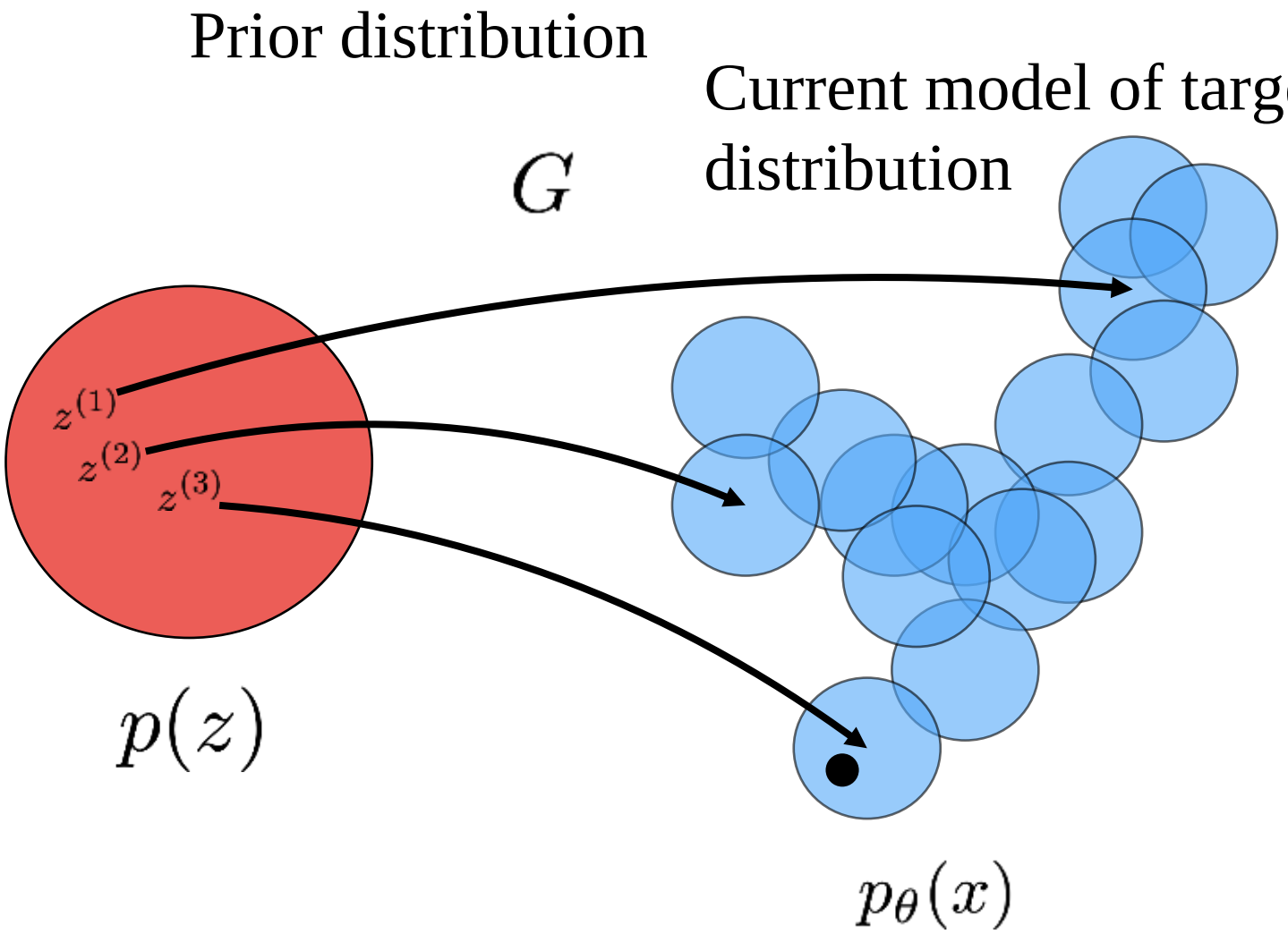
Sampling:

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

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

Variational Autoencoder (VAE)

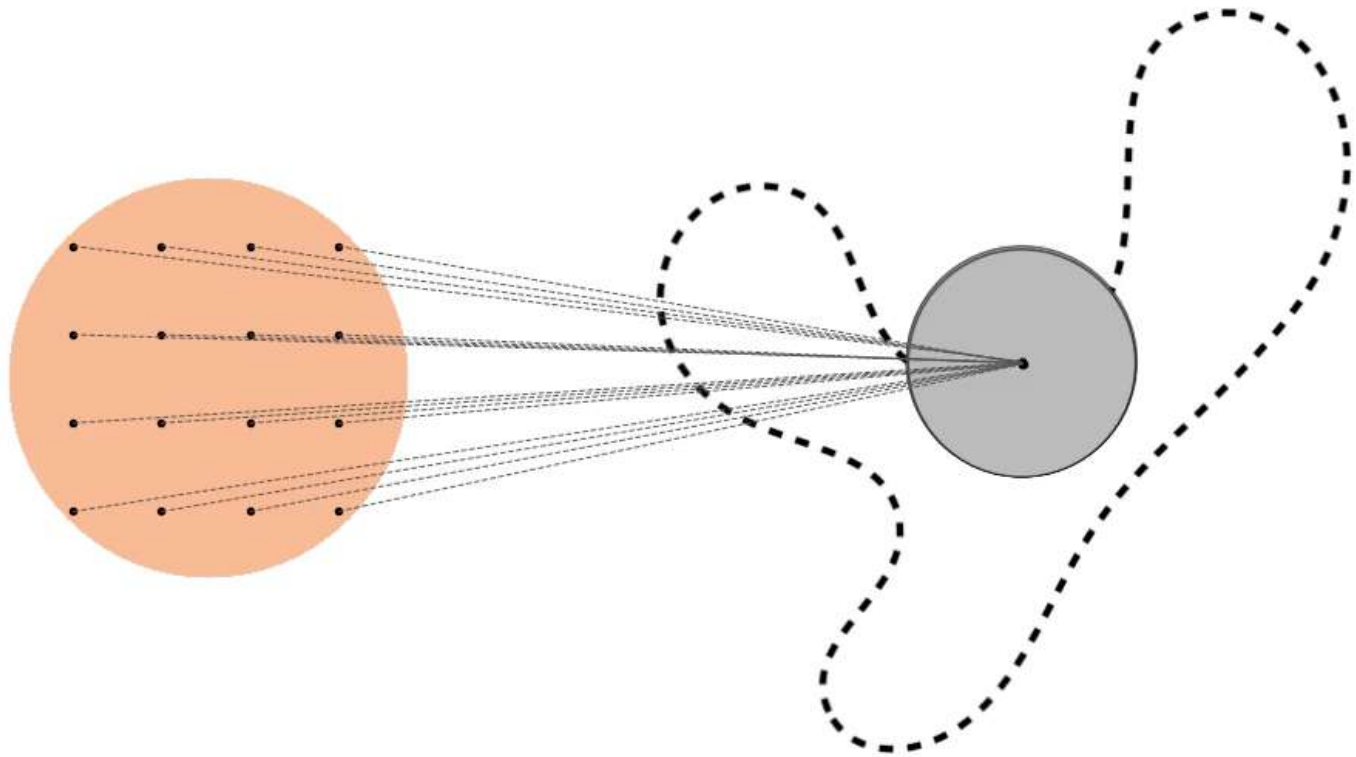




In order to optimize our model, we need to measure the likelihood it assigns to each datapoint x

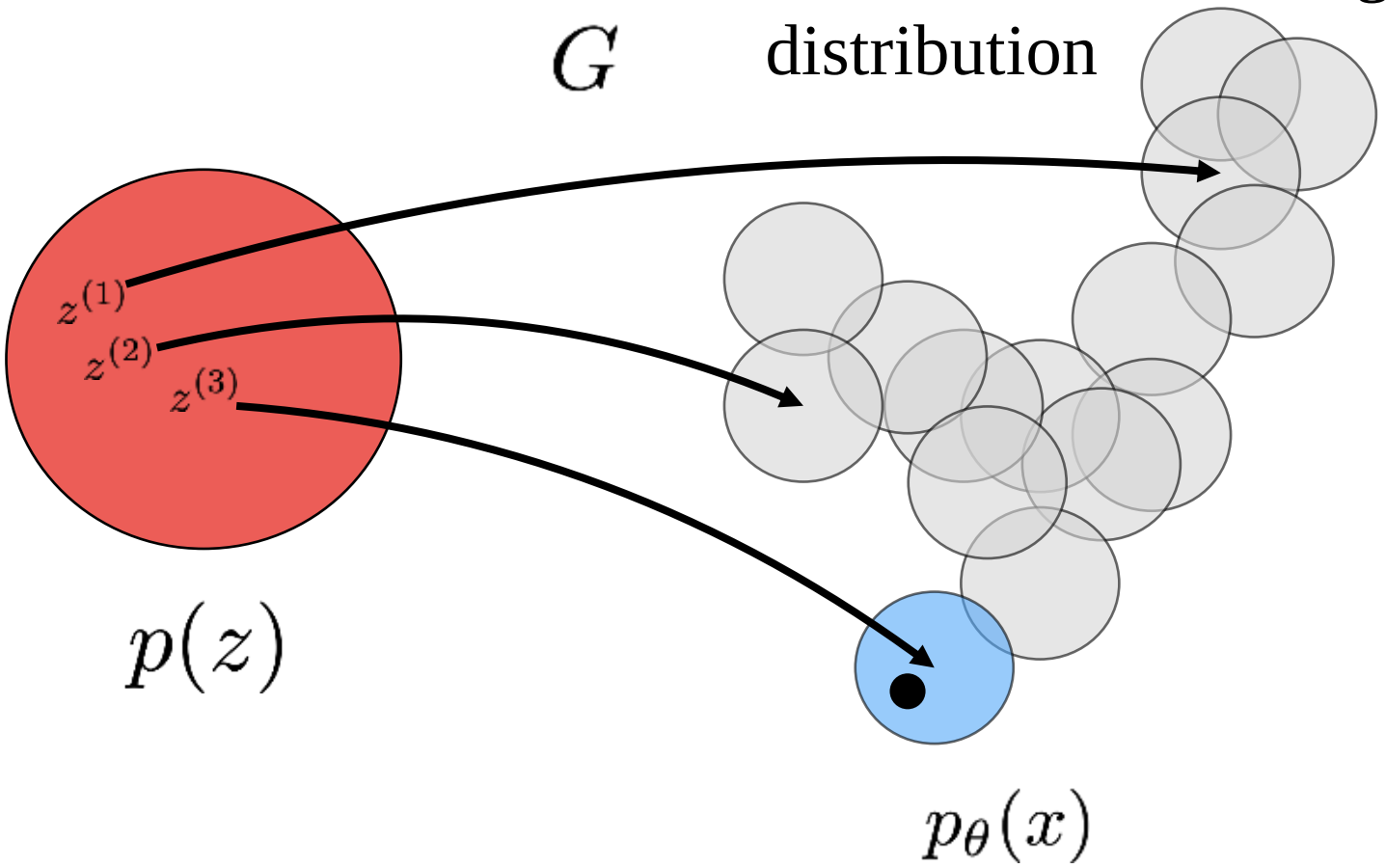
$$p_\theta(x) = \int p(x|z;\theta)p(z)dz$$

$$= p(x|z^{(1)})p(z^{(1)})dz +$$
$$p(x|z^{(2)})p(z^{(2)})dz +$$
$$p(x|z^{(3)})p(z^{(3)})dz + \ldots$$



Prior distribution

Current model of target distribution



In order to optimize our model, we need to measure the likelihood it assigns to each datapoint x

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

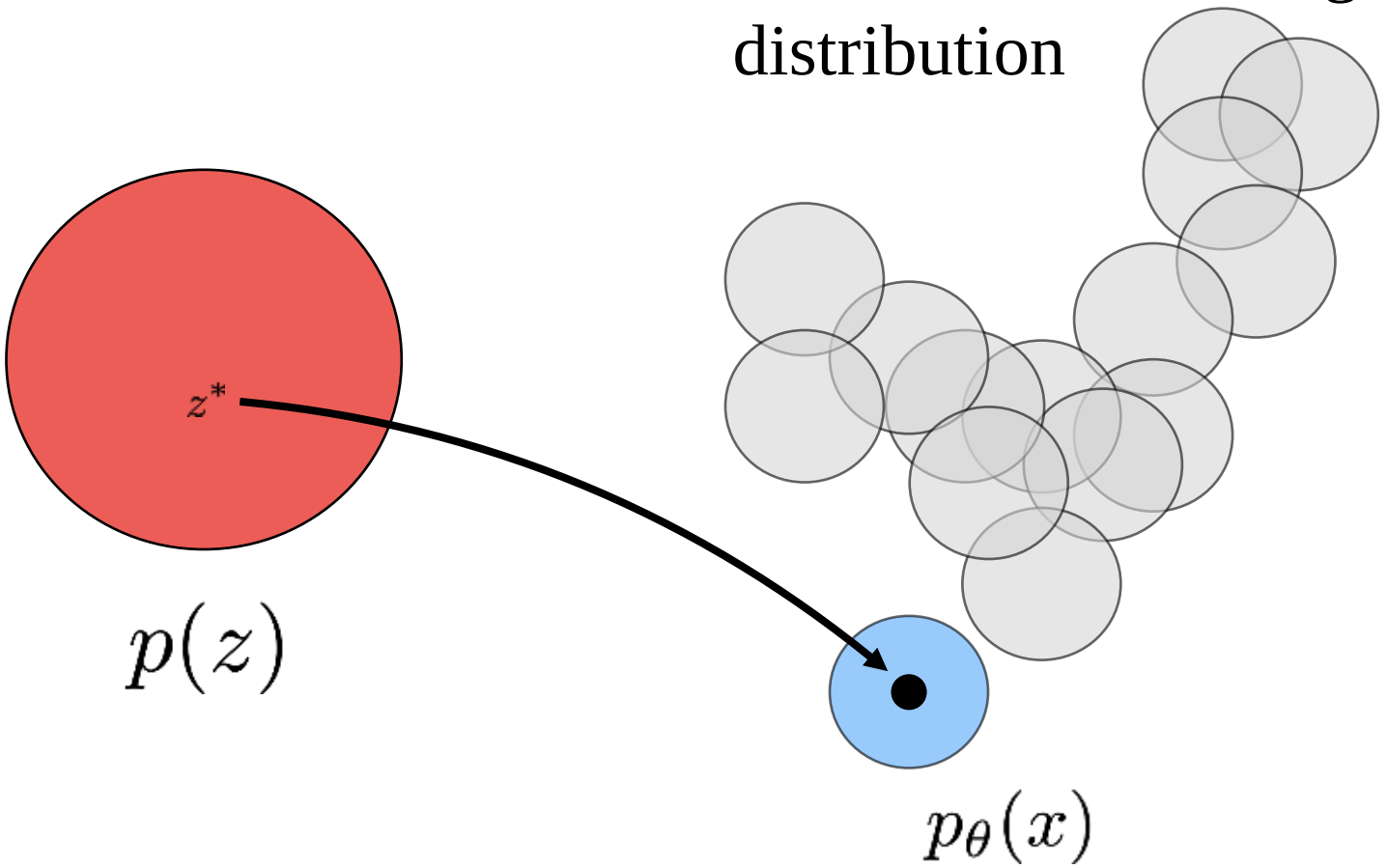
$$= \sim 0+$$

$$\sim 0+$$

$$p(x|z^{(3)})p(z^{(3)})dz + \ldots$$

Prior distribution

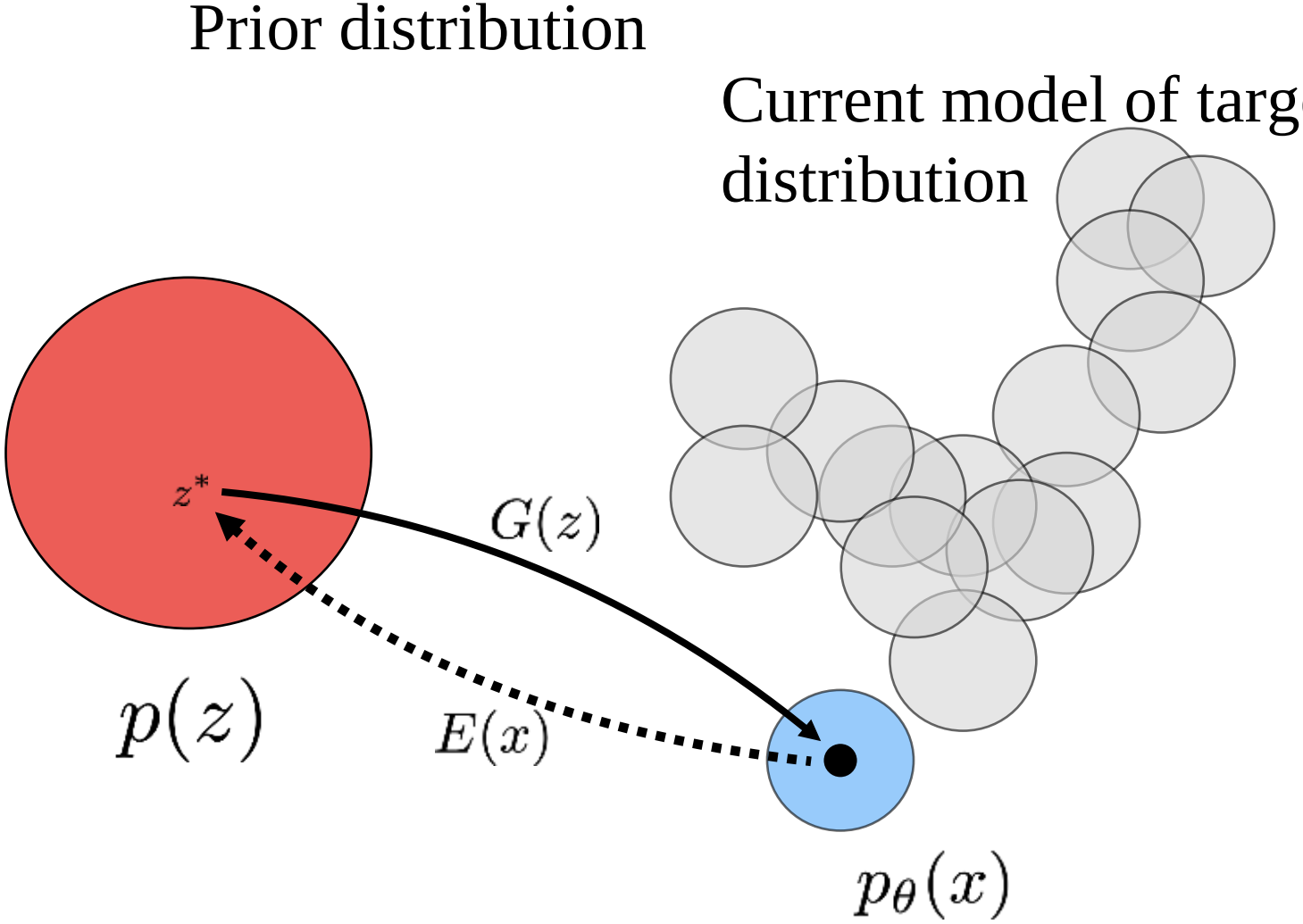
Current model of target distribution



If only we knew z*, we wouldn't need the integral…

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

$$\approx p(x|z^*;\theta)p(z^*)$$



Technical note: for the continuous math to actually work out, z ~ E(x) needs to be a distribution (typically set to Gaussian), but here we (incorrectly) treat it as deterministic for simplicity.*

If only we knew z*, we wouldn't need the integral…

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

$$\approx p(x|z^*;\theta)p(z^*)$$

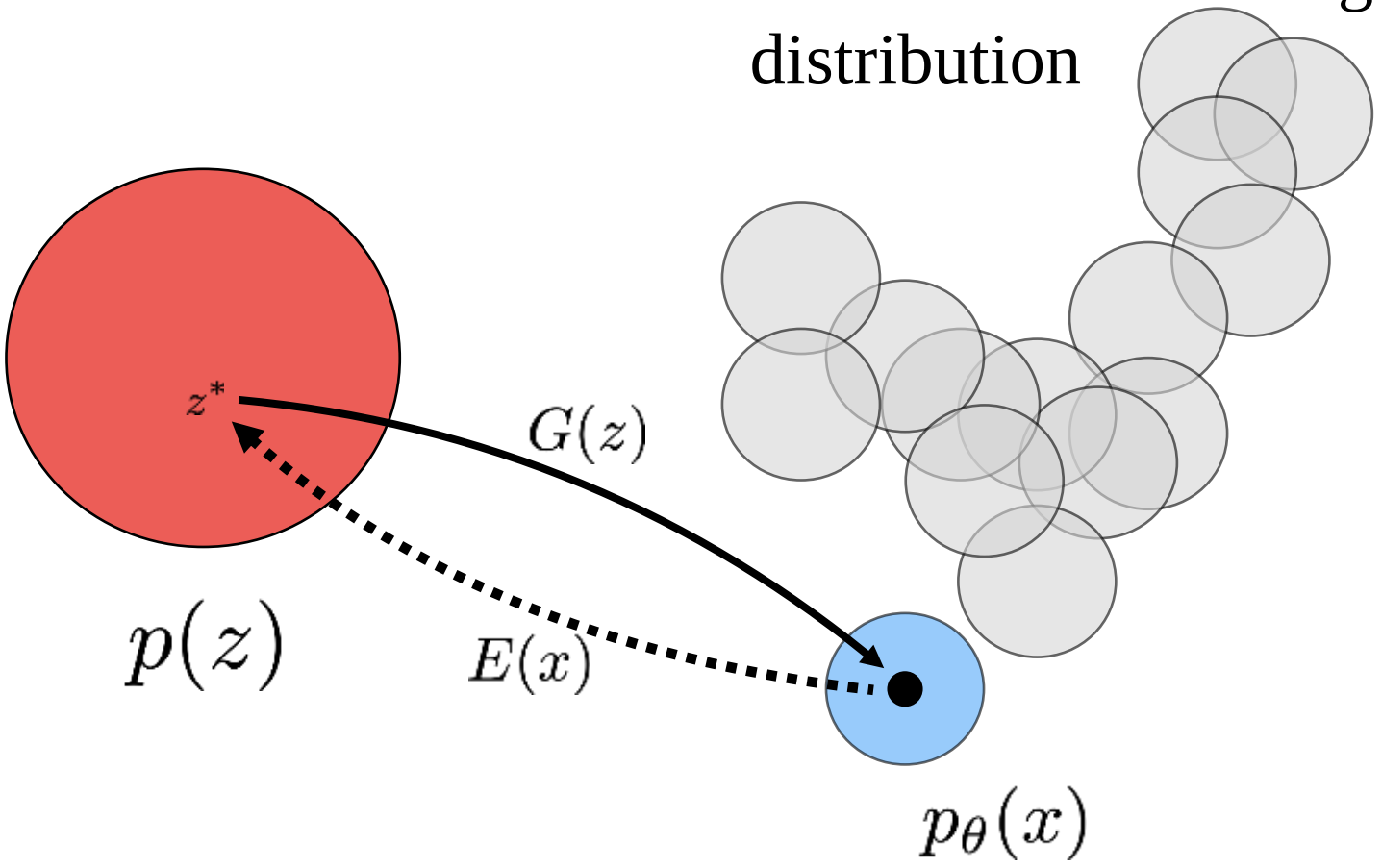
So, we simply try to predict z* for the given x!

$$z^* = E(x)$$

$$\arg\max_E p(x|E(x);\theta)p(E(x))$$

Prior distribution

Current model of target distribution



If only we knew z*, we wouldn't need the integral…

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

$$\approx p(x|z^*;\theta)p(z^*)$$

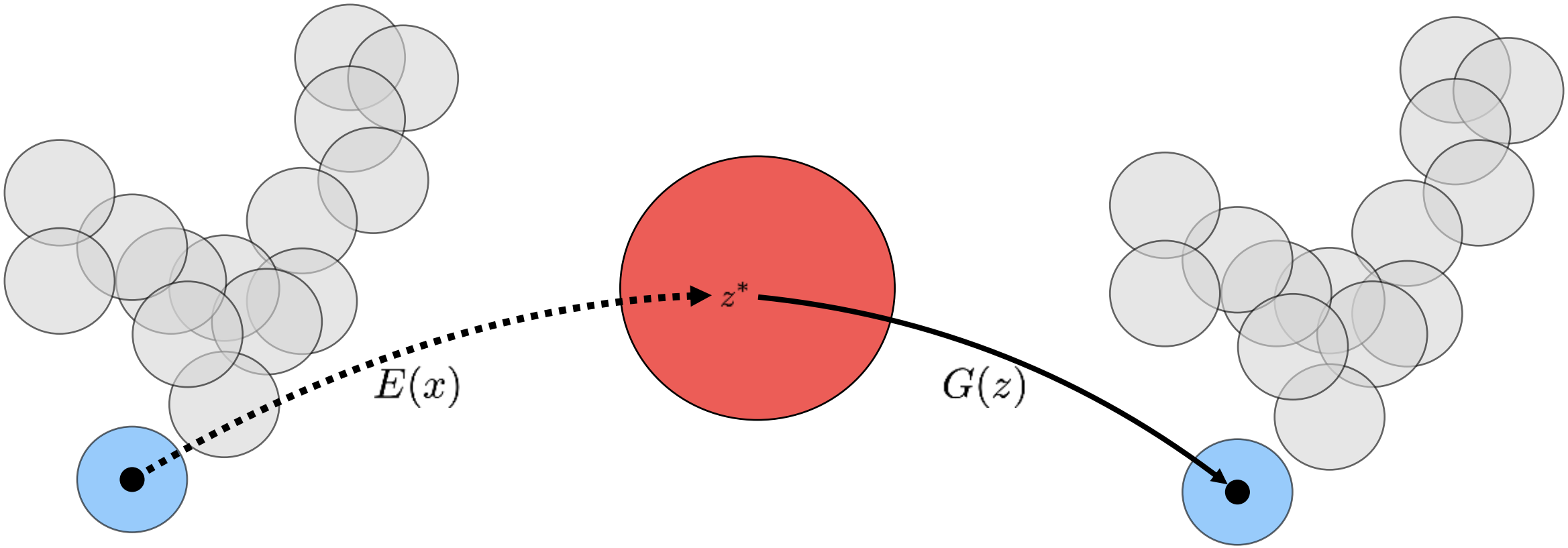
So, we simply try to predict z* for the given x!

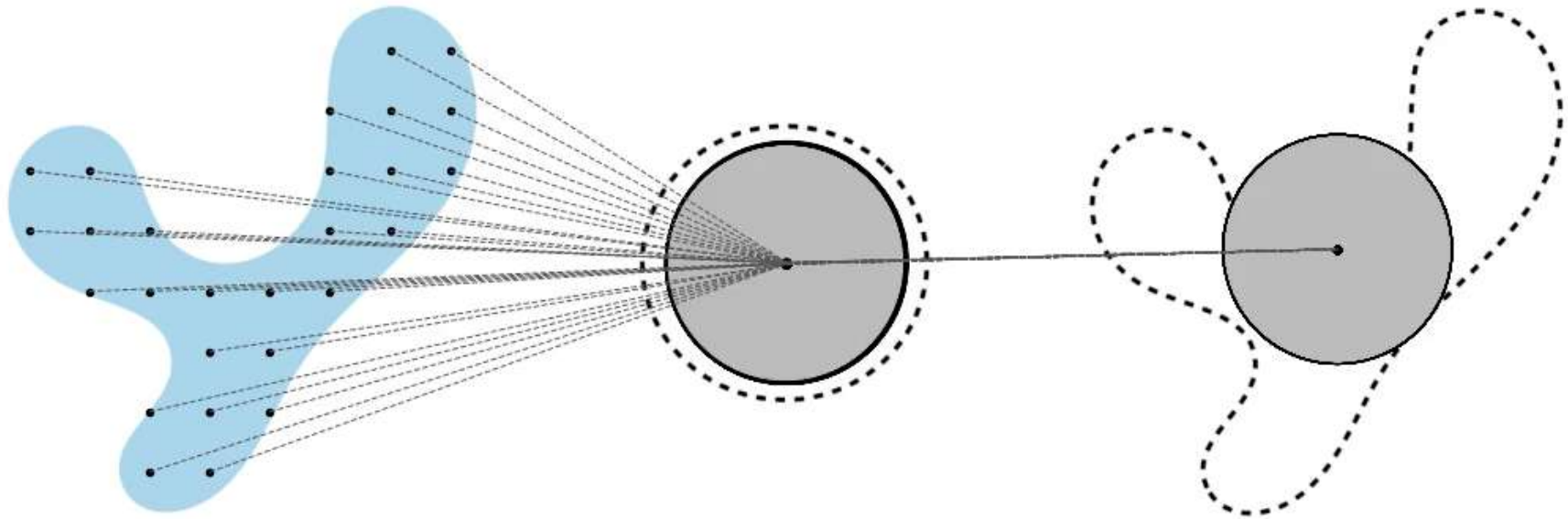
$$z^* = E(x)$$

(assuming unit Gaussian prior, isotropic Gaussian likelihood model) →

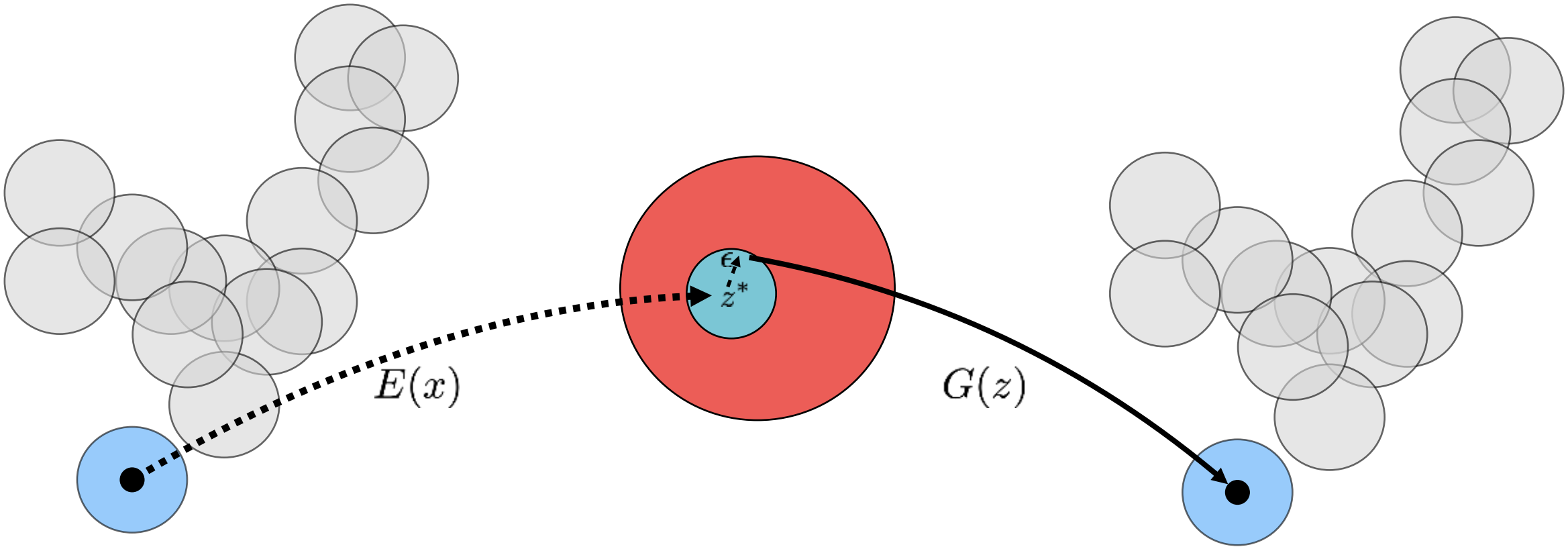
$$\arg\min_E \|G(E(x)) - x\|_2^2 + ||E(x)||_2^2$$

Autoencoder!



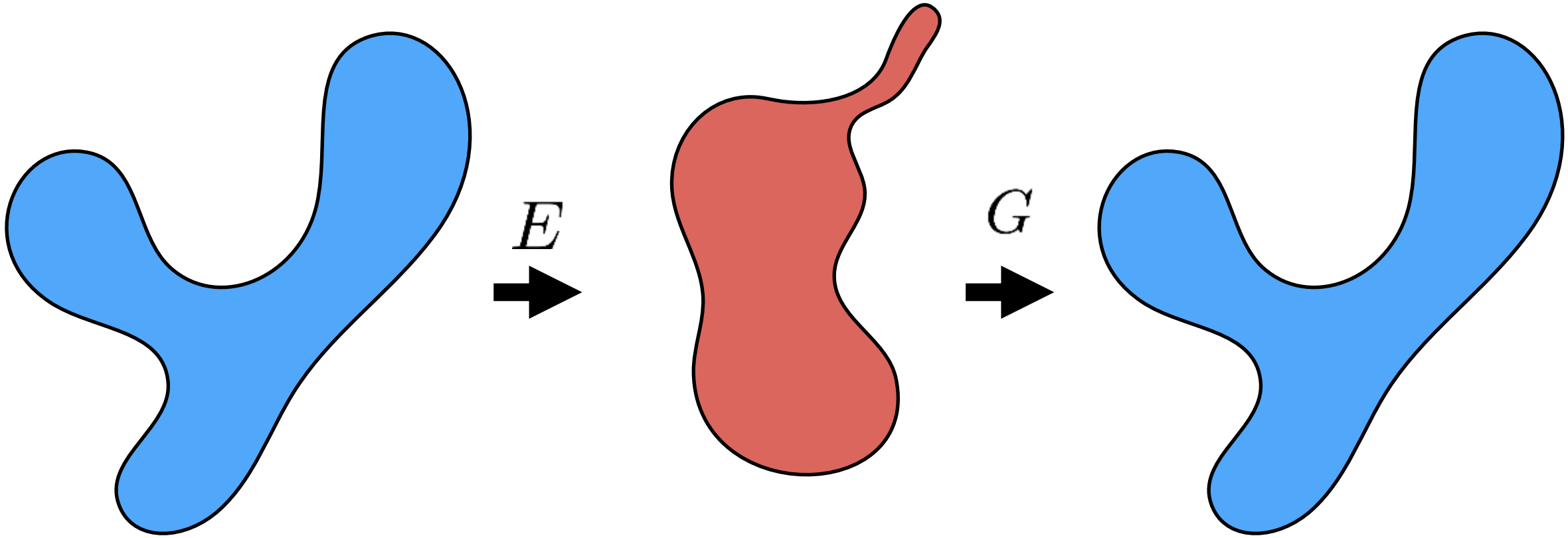


Autoencoder!



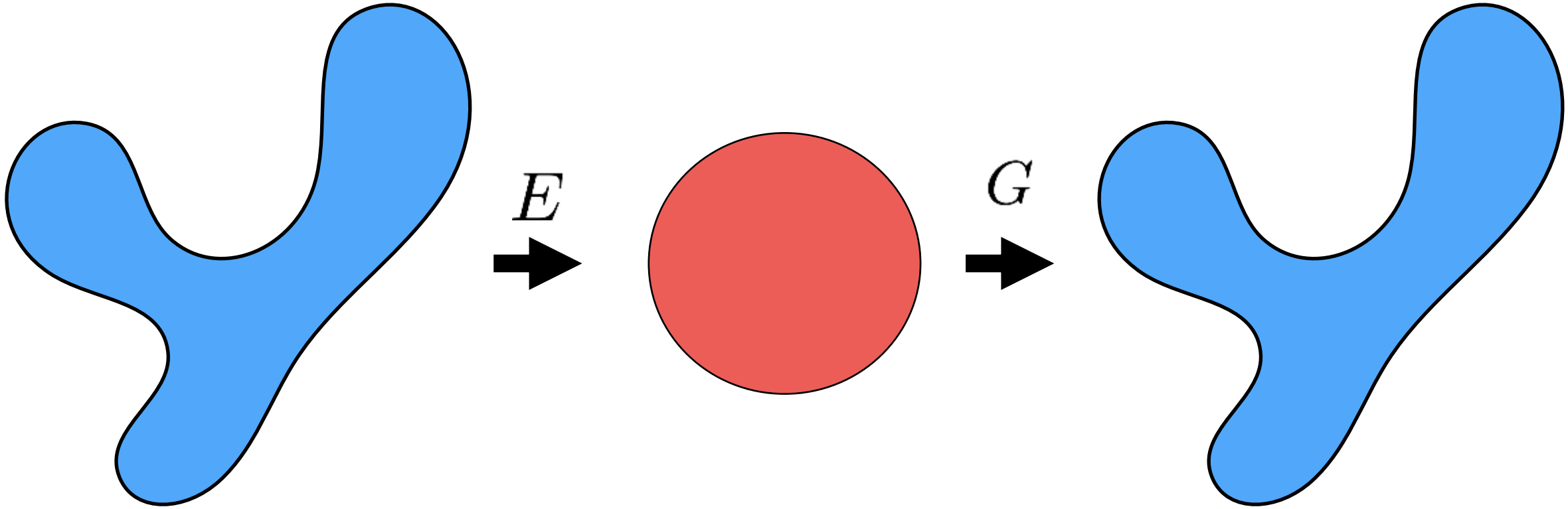
$$\arg\min_{G,E} \mathbb{E}_{x,\epsilon}[\|G(E(x+\epsilon)) - x\|_2^2 + \|E(x+\epsilon)\|_2^2]$$

Classical Autoencoder



$$\arg\min_{G,E} \mathbb{E}_x[\|G(E(x)) - x\|_2^2]$$

Variational Autoencoder

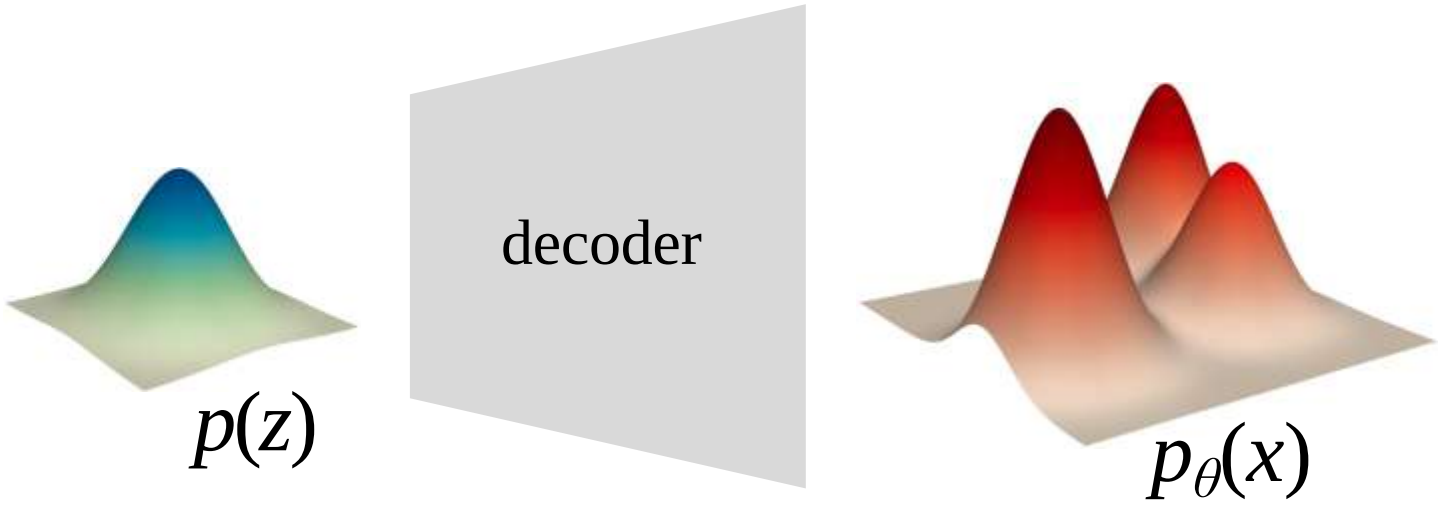


$$\arg\min_{G,E} \mathbb{E}_{x,\epsilon}[\|G(E(x+\epsilon)) - x\|_2^2 + \|E(x+\epsilon)\|_2^2]$$

Recap.

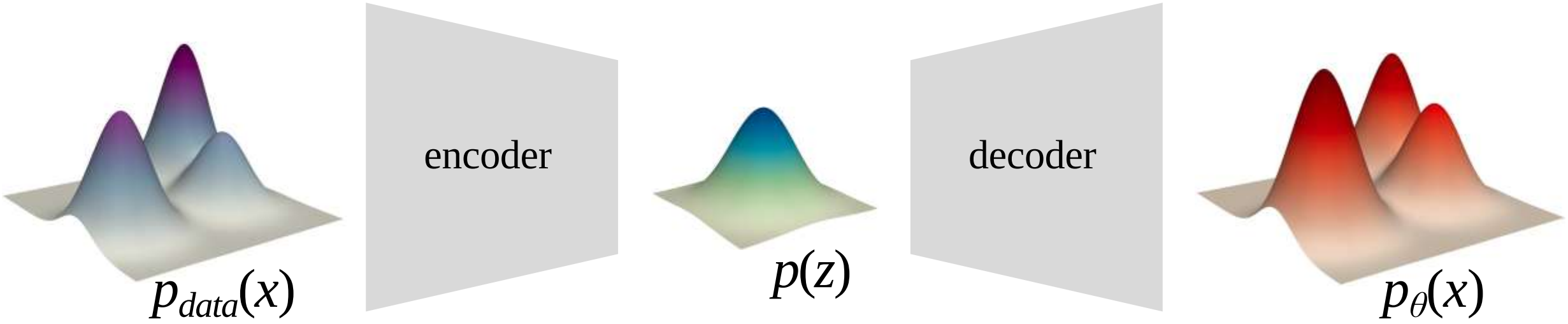
A view of “Autoencoding Distributions”

- decoder: maps latent distribution to data distribution



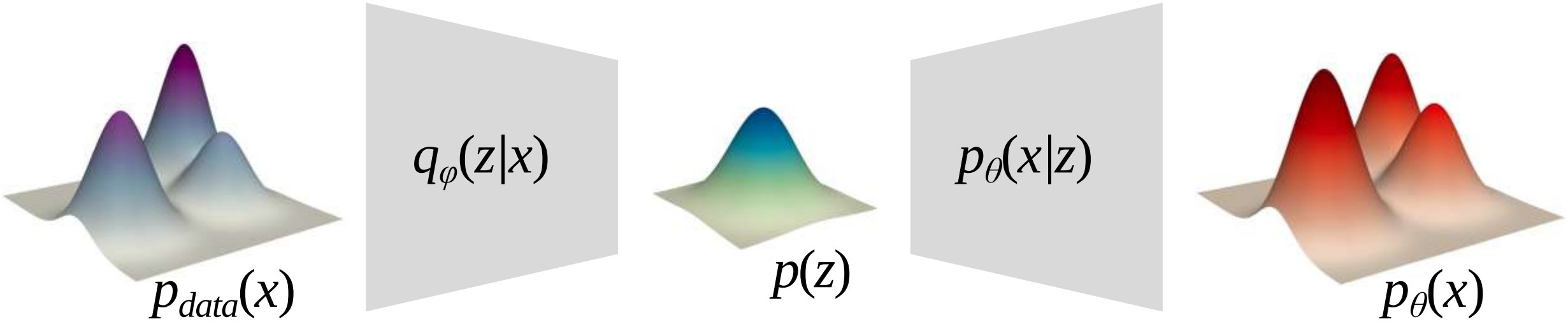
A view of “Autoencoding Distributions”

- encoder: maps data distribution to latent distribution
- decoder: maps latent distribution to data distribution



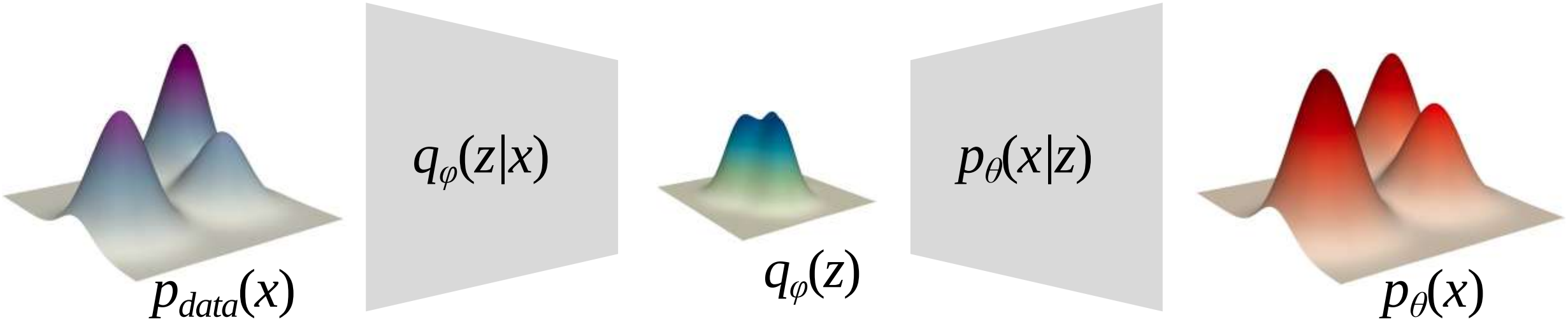
A view of “Autoencoding Distributions”

- encoder: maps data distribution to latent distribution
- decoder: maps latent distribution to data distribution



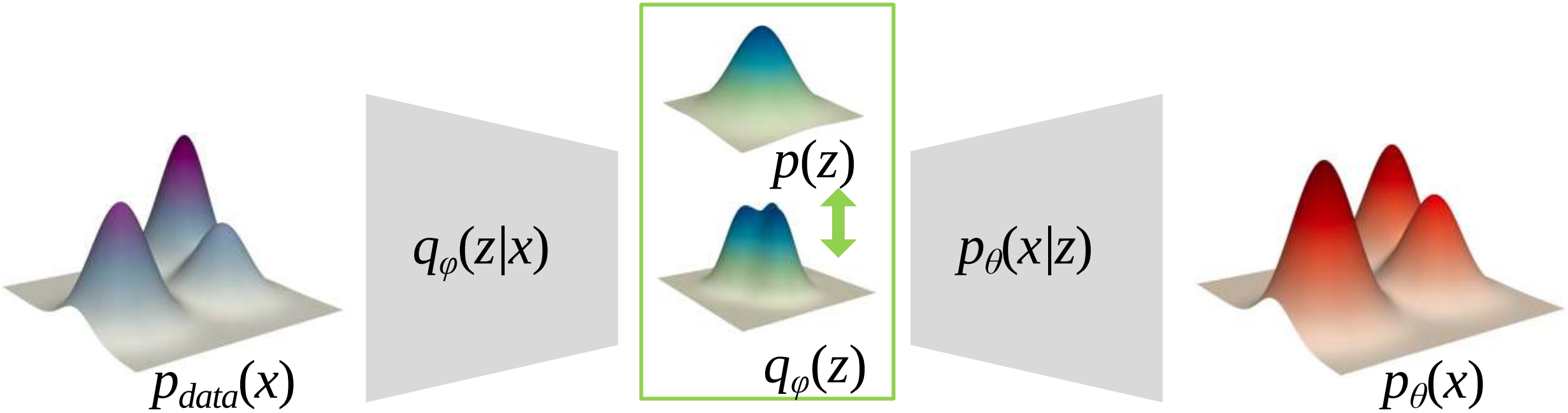
A view of “Autoencoding Distributions”

- encoded latent distribution: $q_\phi(z) = \int_x q_\phi(z|x) p_{data}(x) dx$



A view of "Autoencoding Distributions"

- encoded latent distribution: $q_\phi(z) = \int_x q_\phi(z|x) p_{data}(x) dx$



E.g., see "*InfoVAE: Information Maximizing Variational Autoencoders*", 2017

Illustration

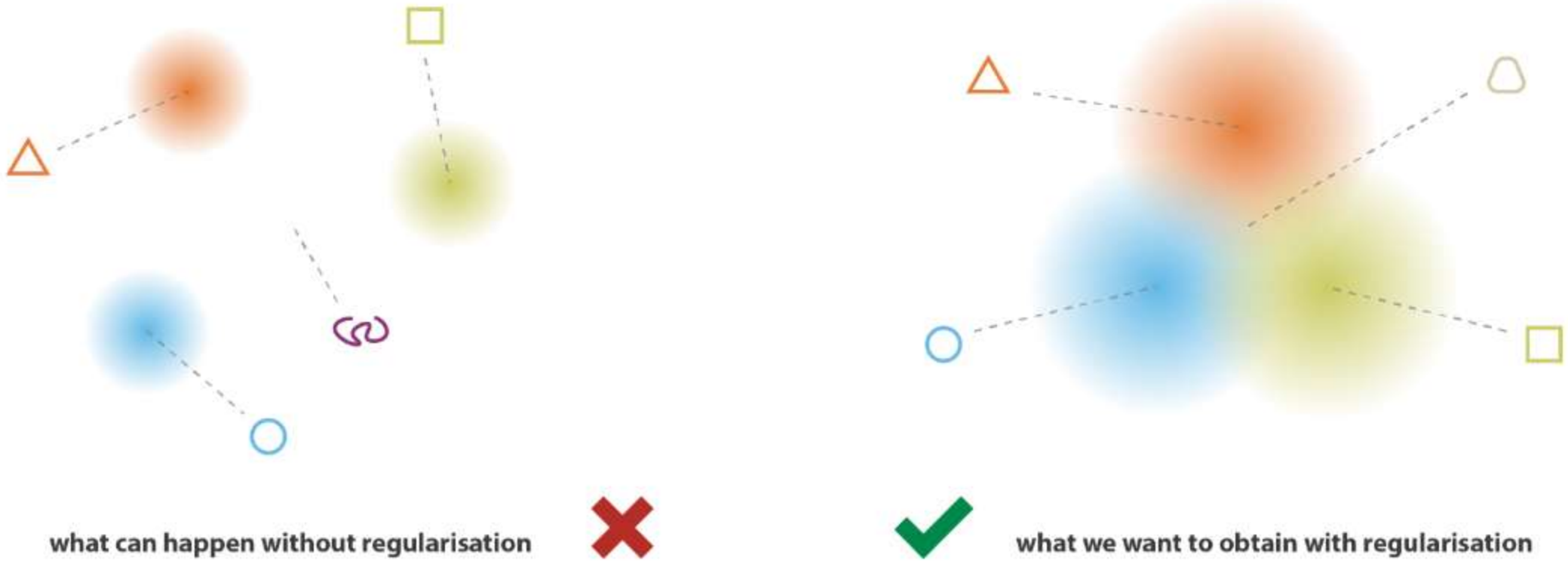


Figure adapted from: Joseph Rocca “Understanding Variational Autoencoders (VAEs)” https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Illustration

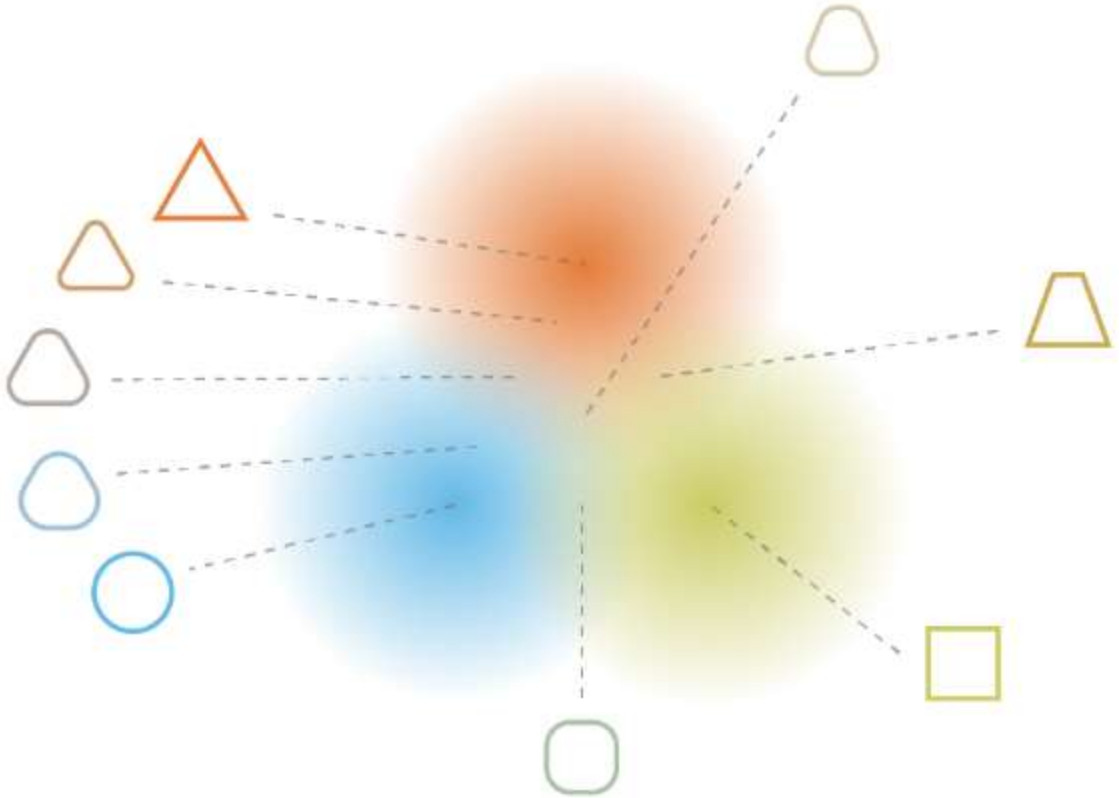


Figure adapted from: Joseph Rocca “Understanding Variational Autoencoders (VAEs)”
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Illustration

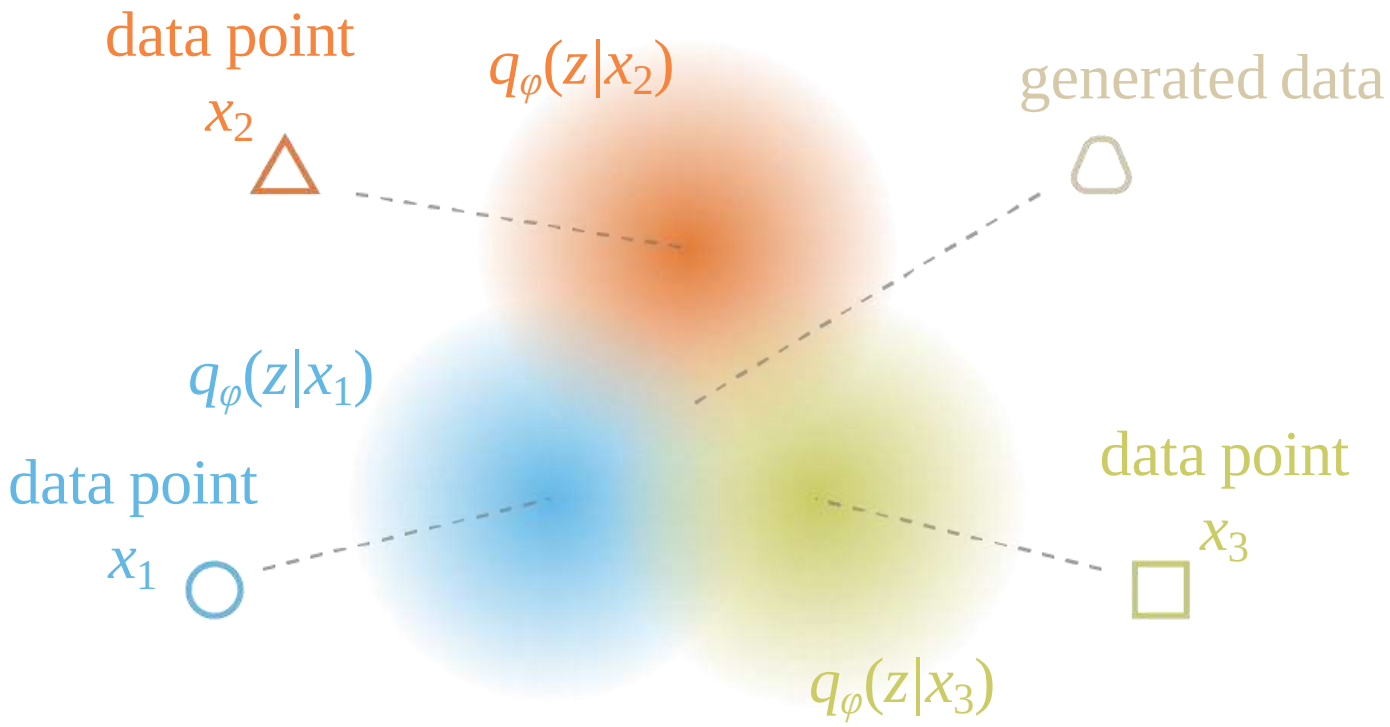
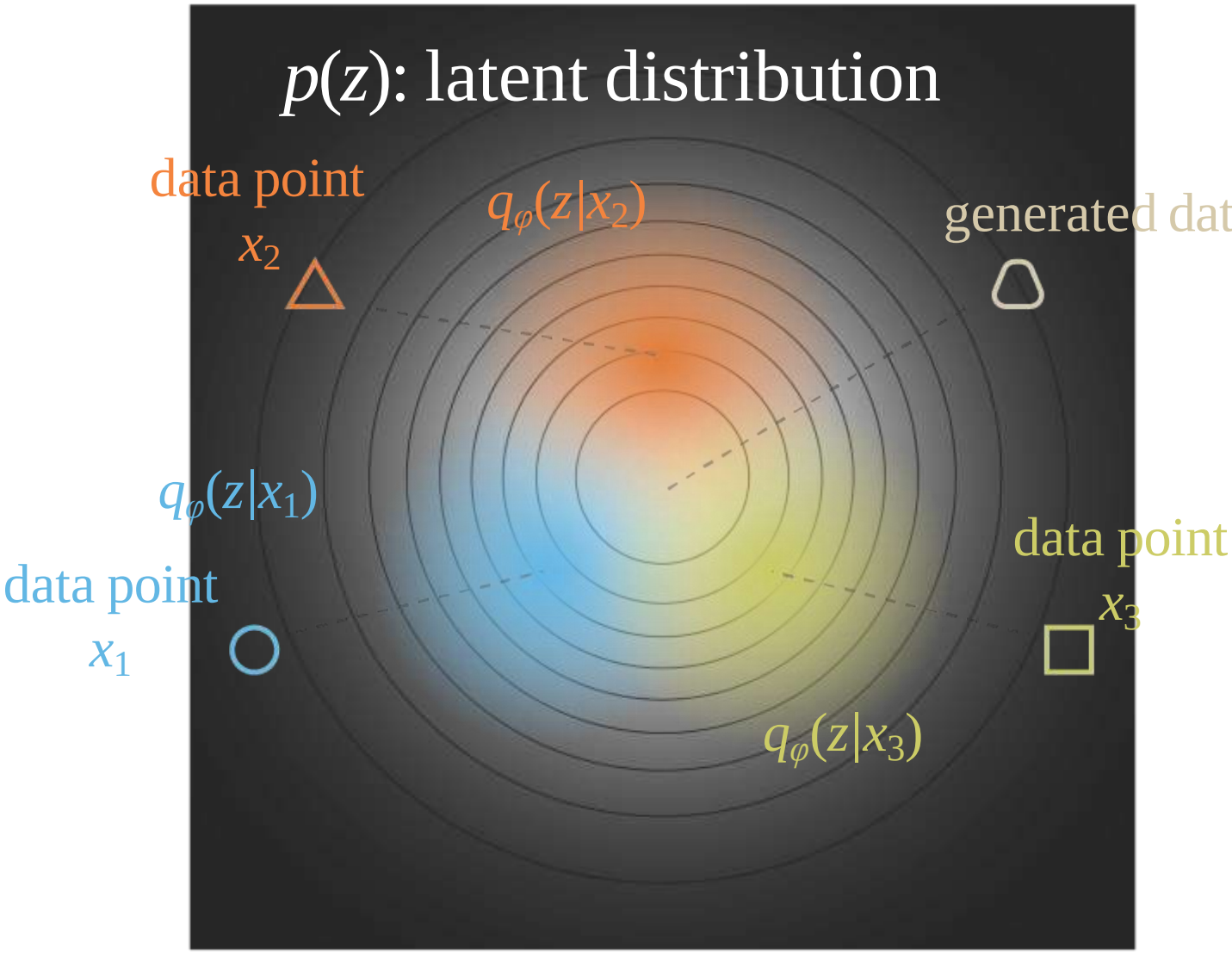


Figure adapted from: Joseph Rocca “Understanding Variational Autoencoders (VAEs)”
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Illustration



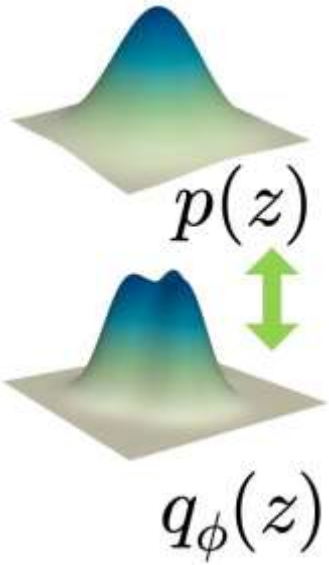
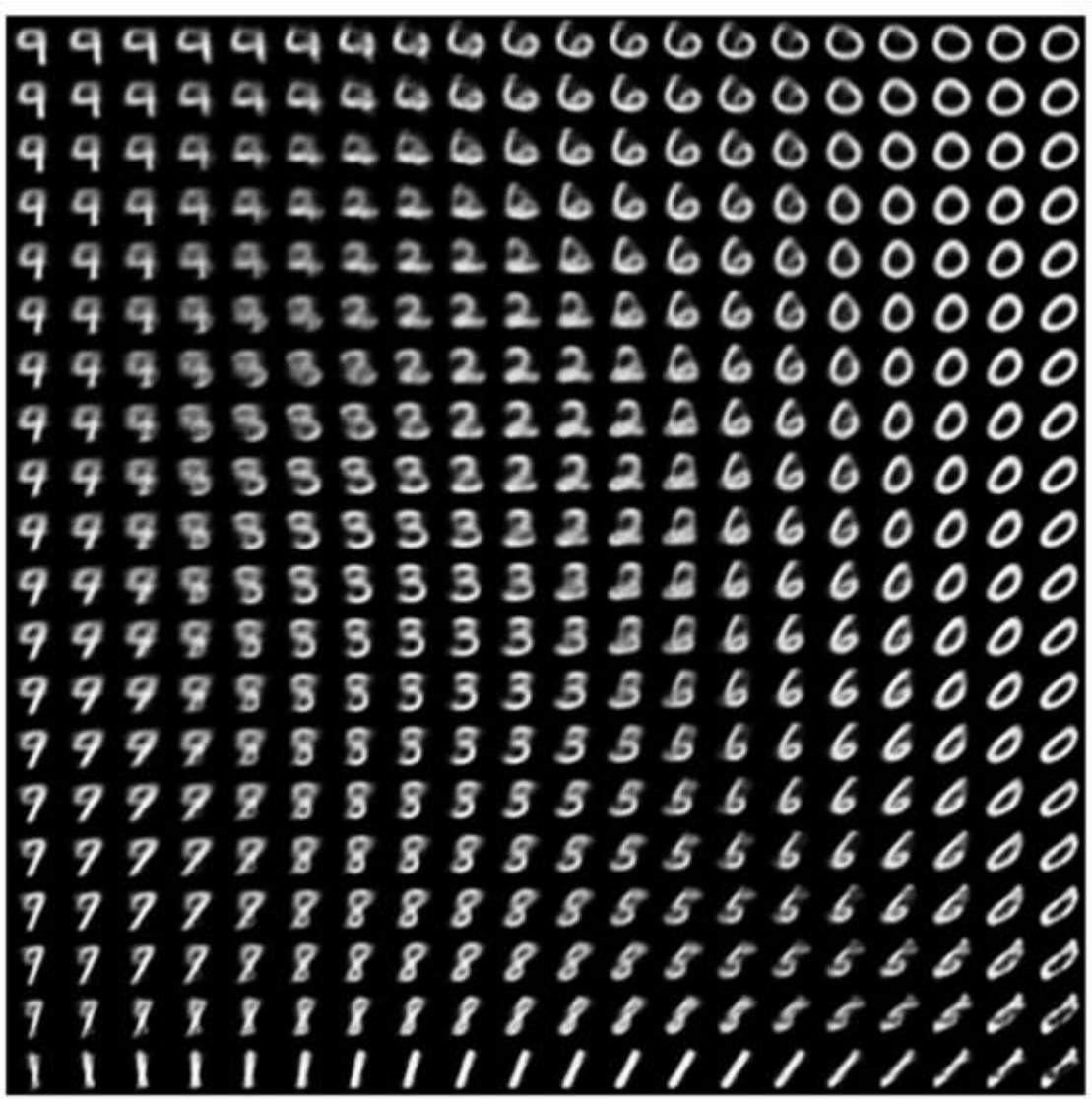


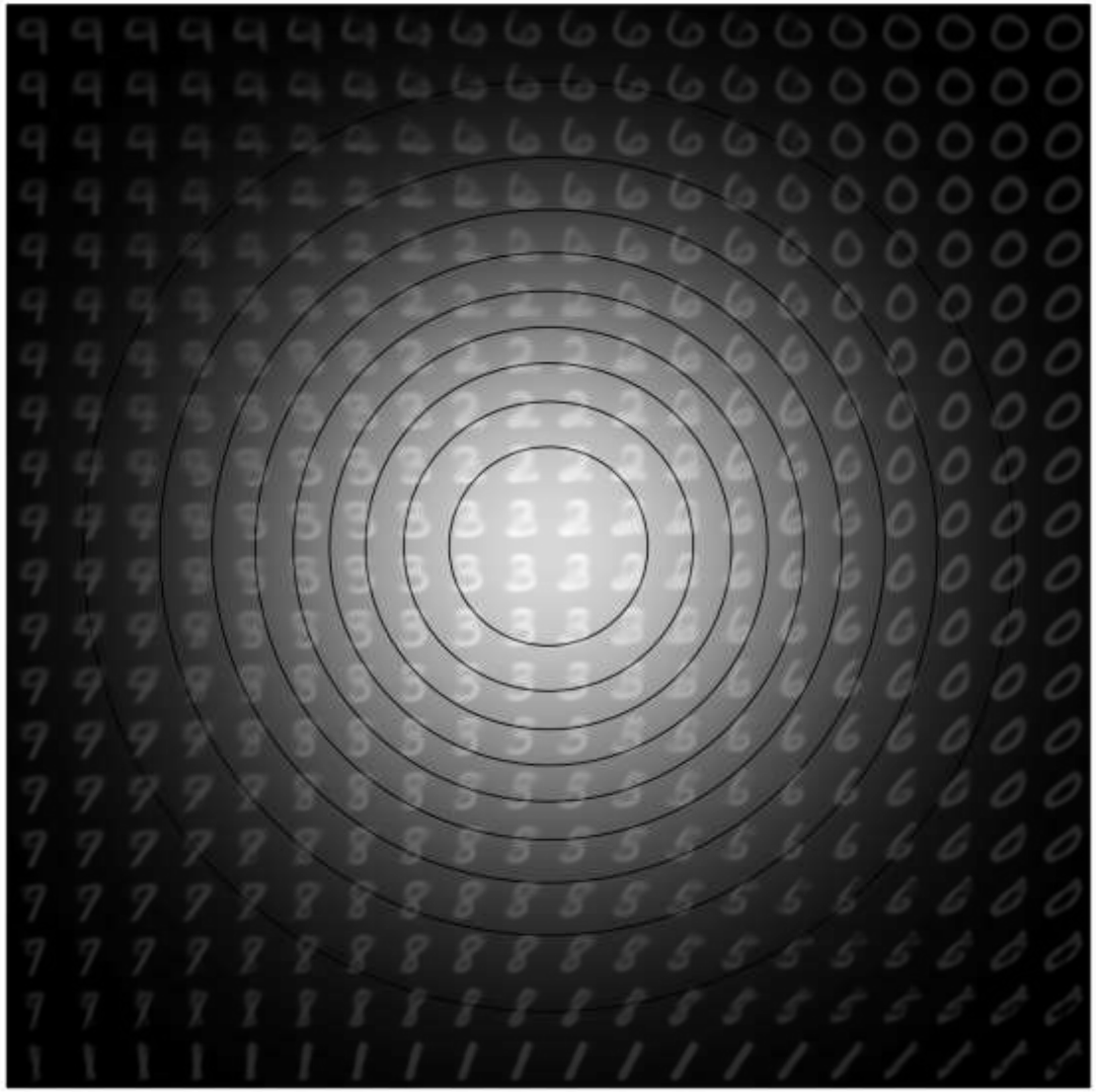
Figure adapted from: Joseph Rocca "Understanding Variational Autoencoders (VAEs)"
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

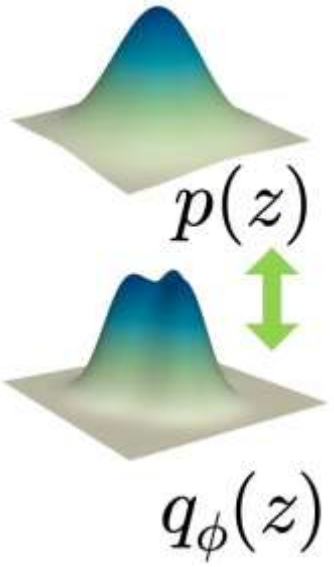
VAE: 2D latent space on MNIST



"Convolutional Variational Autoencoder"
https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/cvae.ipynb

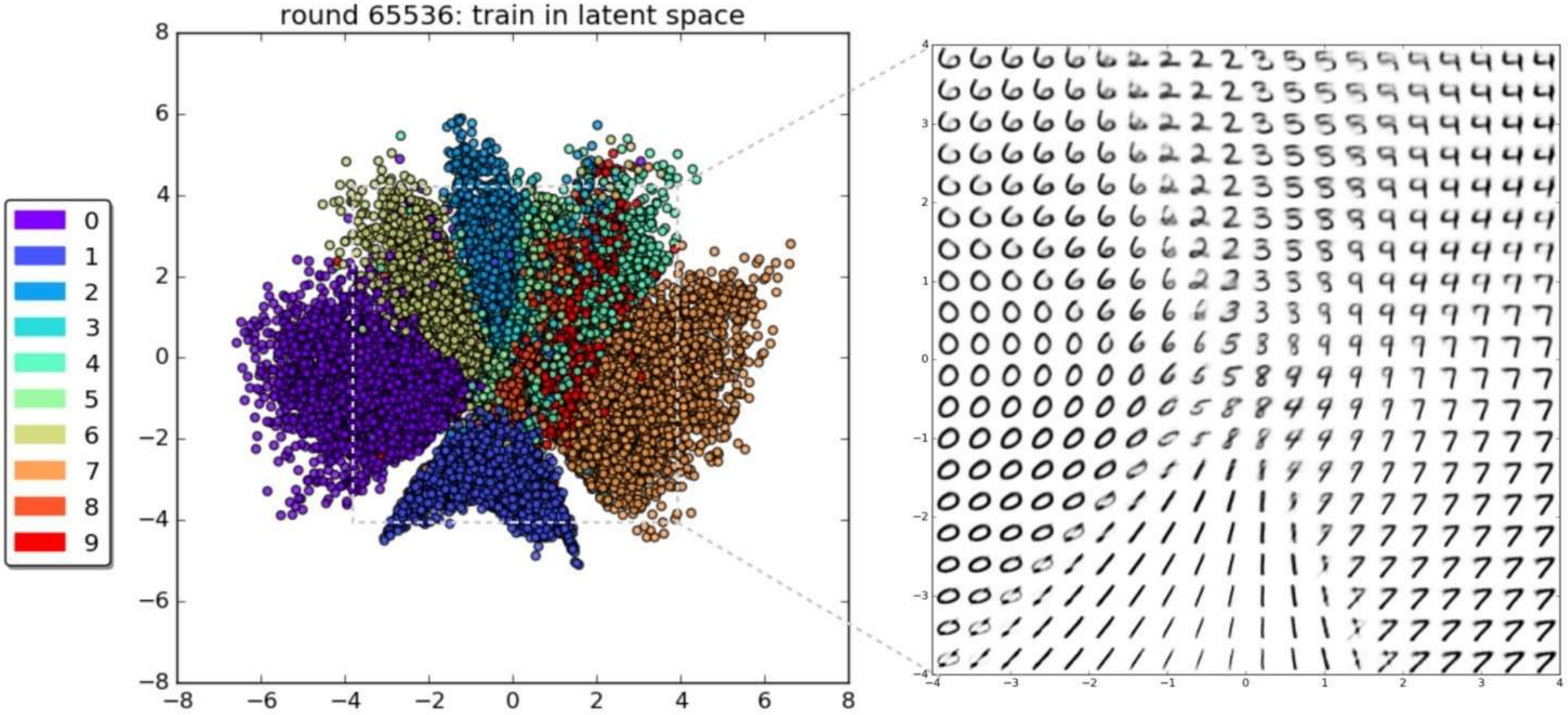
VAE: 2D latent space on MNIST





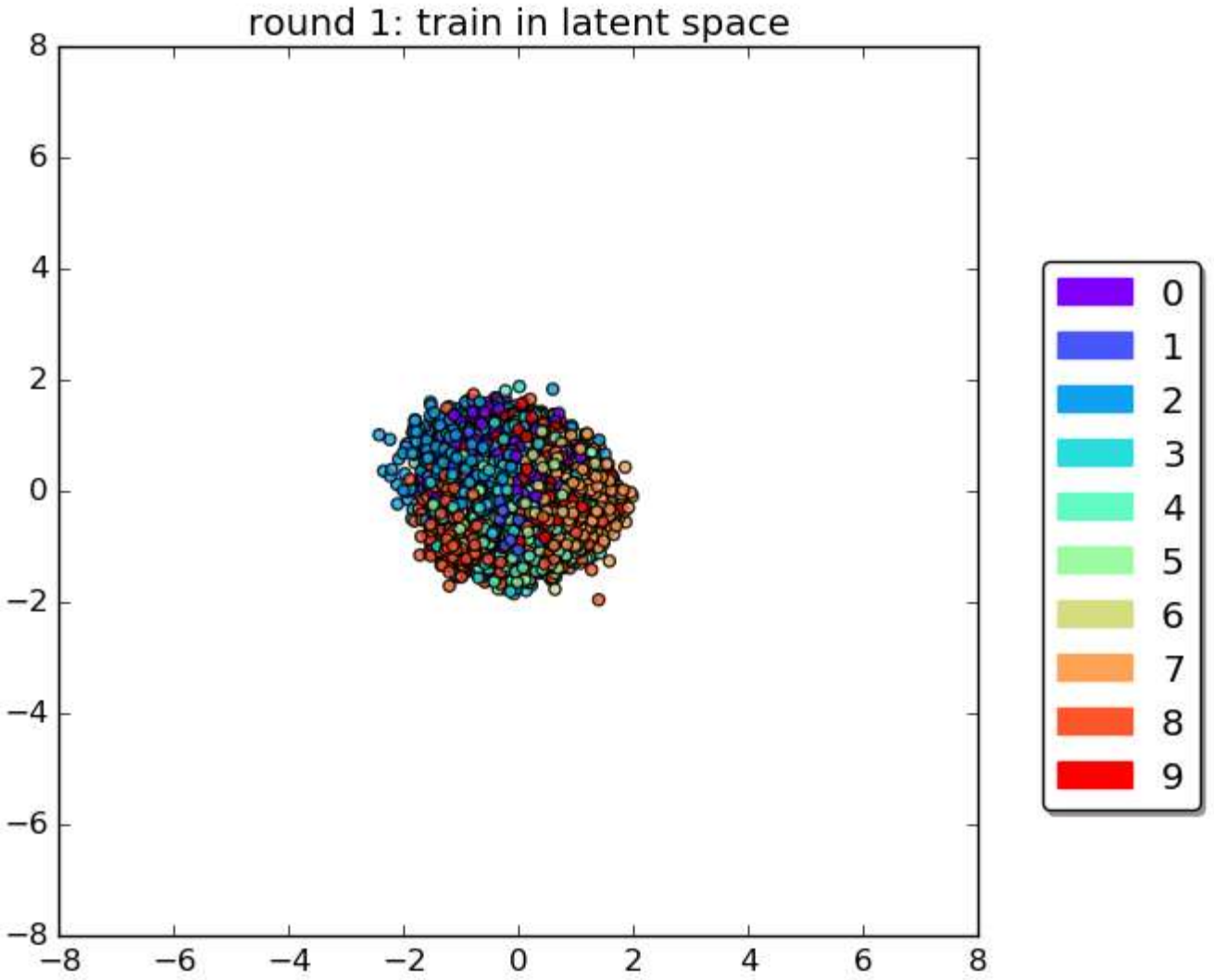
"Convolutional Variational Autoencoder"
https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/cvae.ipynb

VAE: 2D latent space on MNIST



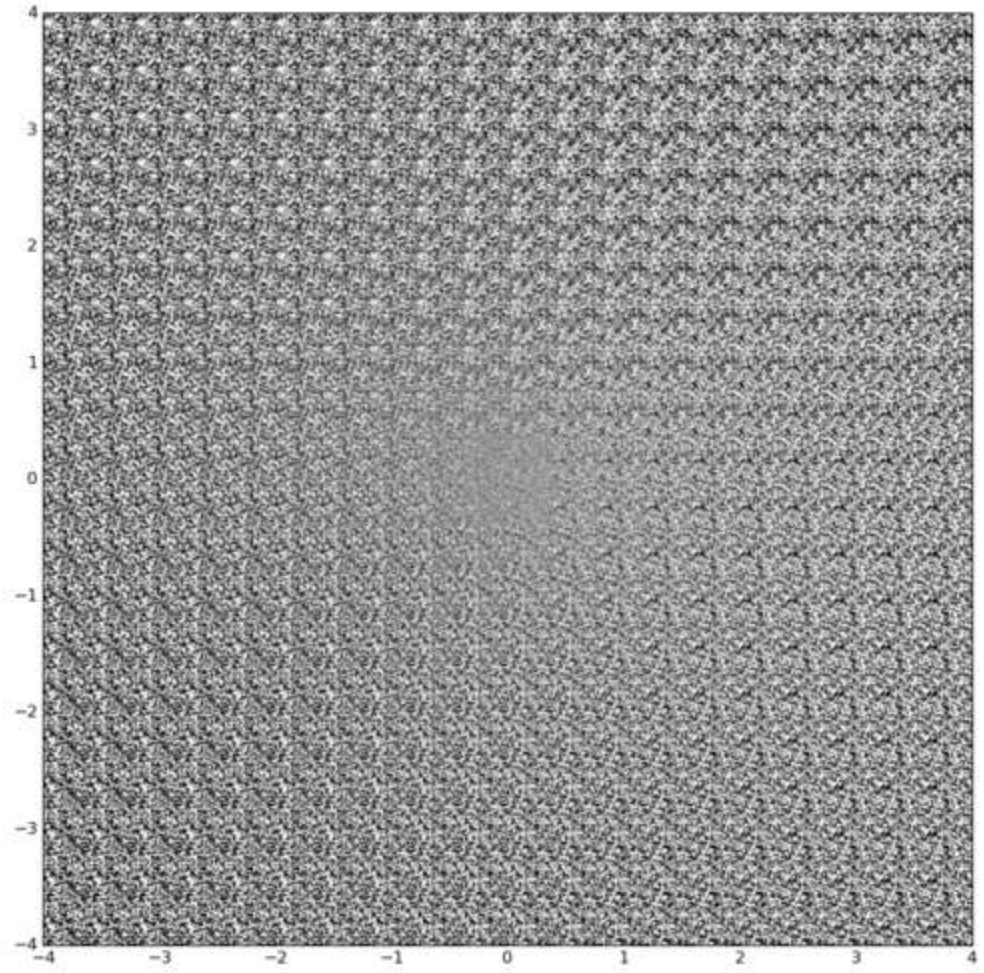
"Introducing Variational Autoencoders (in Prose and Code)"
https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

VAE: 2D latent space on MNIST



"Introducing Variational Autoencoders (in Prose and Code)"
https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

VAE: 2D latent space on MNIST

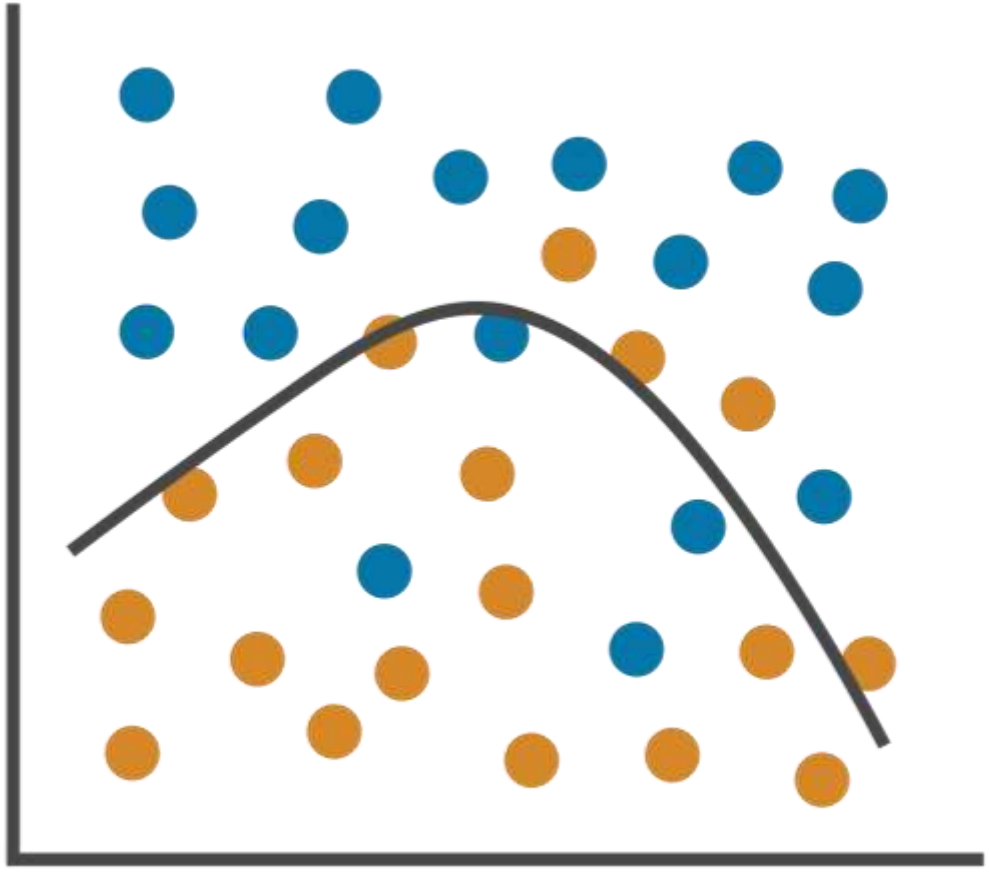


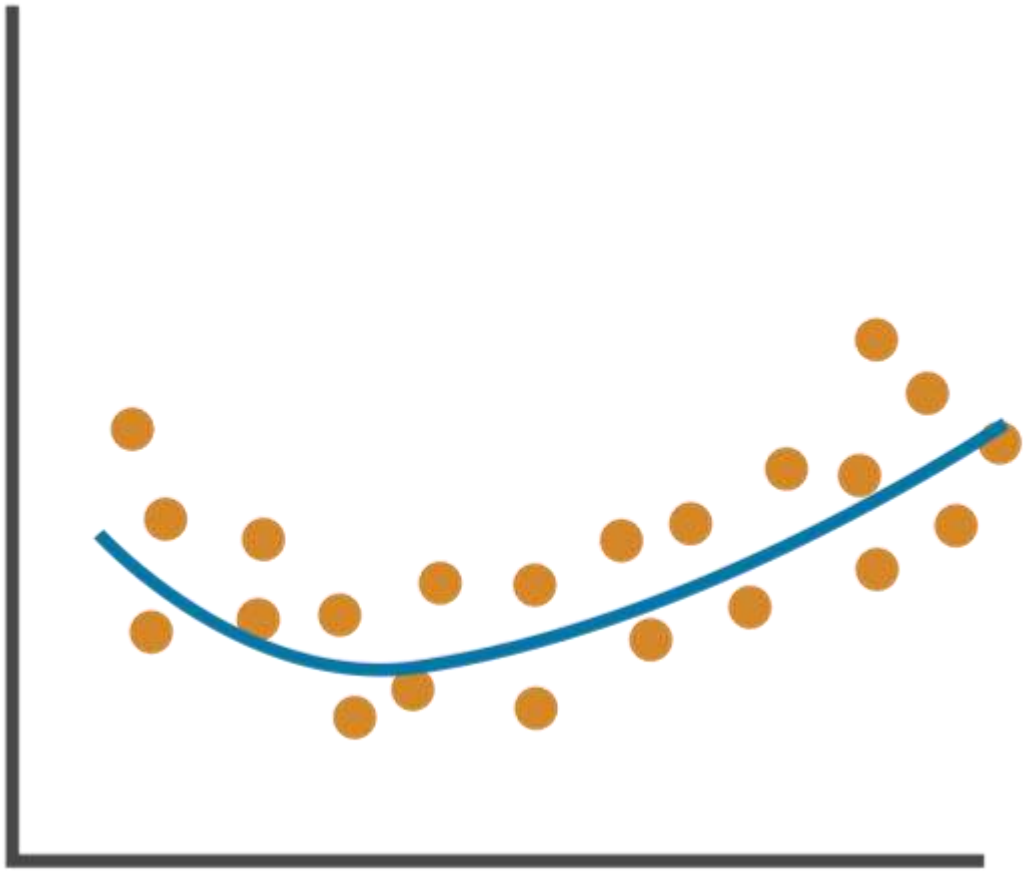
“Introducing Variational Autoencoders (in Prose and Code)”
https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

VAE: 2D latent space on “Frey Face” dataset

Kingma and Welling. Auto-Encoding Variational Bayes, ICLR 2014

About How to Optimize





Two approaches:

- *Classification: Prob. Distribution*
- *Regression: Why we choose this approach? （self-supervised）*

Relation to Expectation-Maximization (EM)

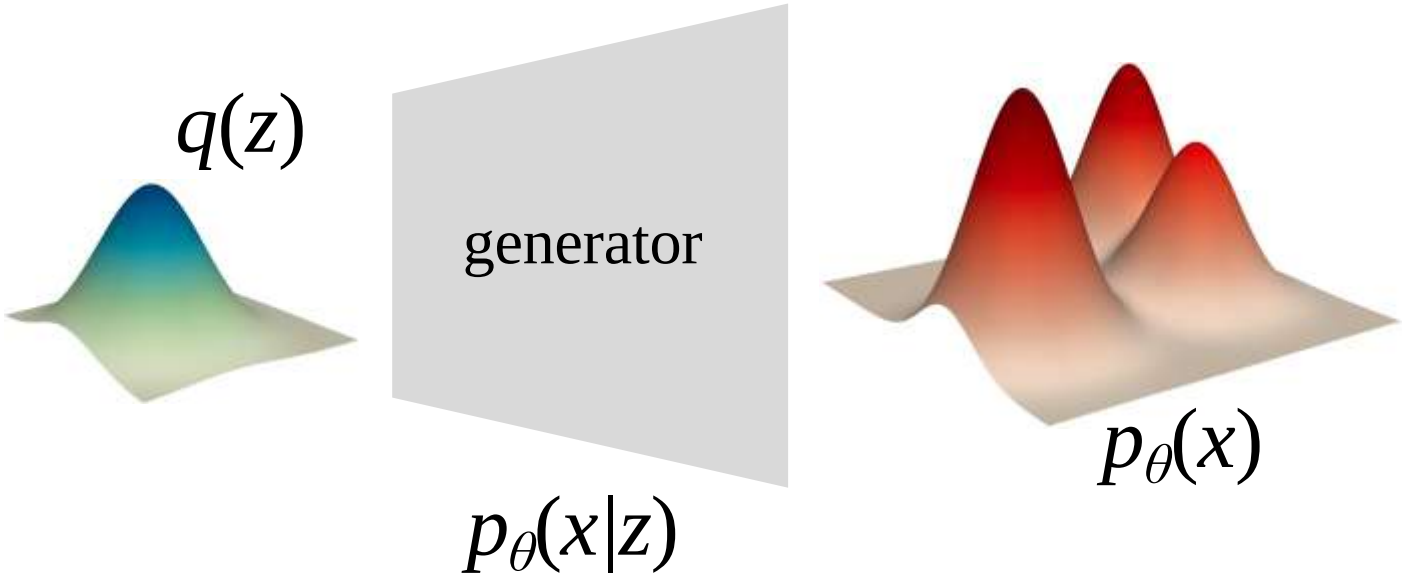
Recap: Latent Variable Models

Two sets of variables:

- q: distribution of latent
- θ: parameters of generator

VAE:

- parametrize q by a network
- stochastic gradient decent



Expectation-Maximization (EM):

- often parametrize q analytically
- coordinate descent (i.e., alternating optimization)

EM – A Match x

EM – Before Match q

EM – Before Match θ

EM – First Half x_t, q, θ

EM – Intermission $q^{(t)}$ and $p_\theta(z|x)$

EM – Second Half $p_\theta(x_{t+1}|z)$

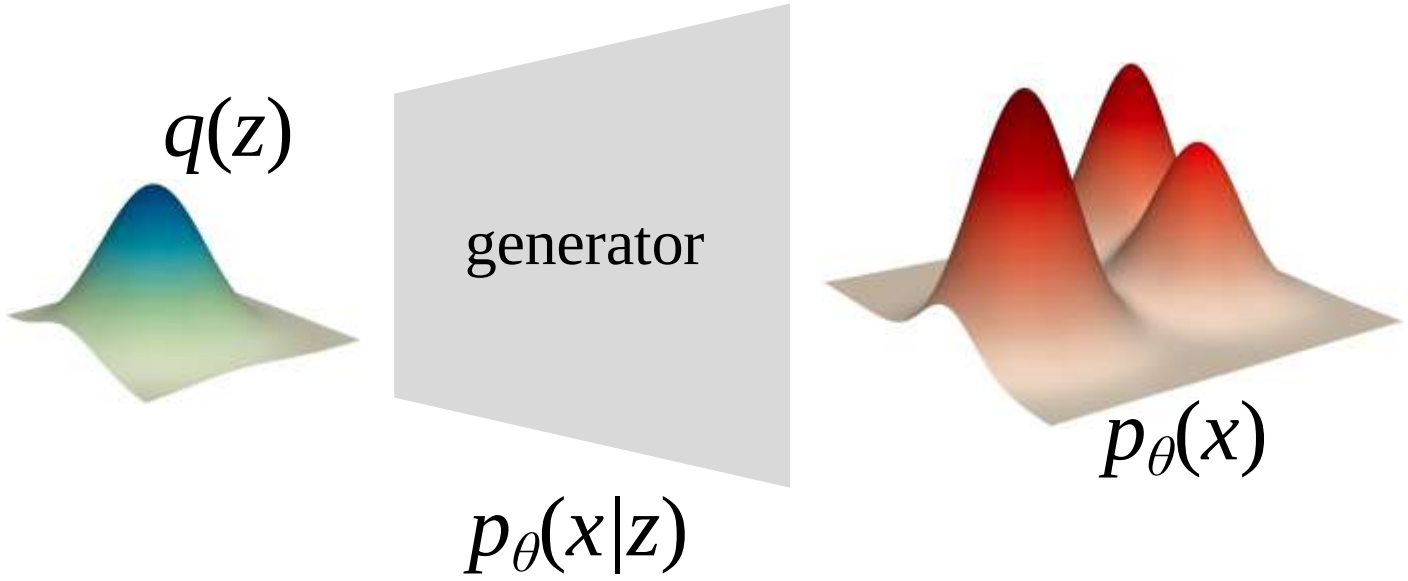
EM – Maximize $p_\theta(x)$

Posterior $p_\theta(z|x)$

Posterior $p_\theta(z|x)$

EM as A Max-Max Procedure

$$\text{ELBO} = \mathbb{E}_{x \sim p_{data}(x)} \Big[\mathbb{E}_{z \sim q(z|x)} \Big[\log p_\theta(x|z) \Big] - \mathcal{D}_{\text{KL}} \Big(q(z|x) || p(z) \Big) \Big]$$



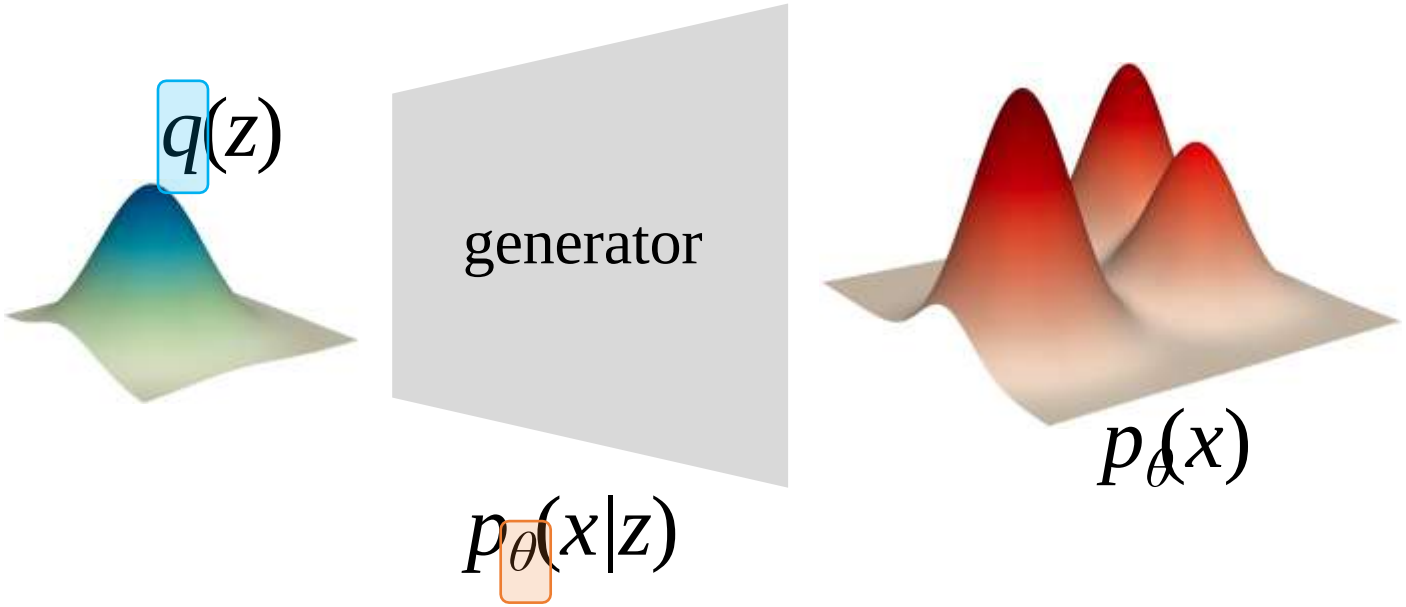
EM as A Max-Max Procedure

$$\text{ELBO} = \mathbb{E}_{x \sim p_{data}(x)} \left[\mathbb{E}_{z \sim q(z|x)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\text{KL}} \left(q(z|x) || p(z) \right) \right]$$

$$\max_{\theta, q} \text{ELBO}(\theta, q(\cdot))$$

Two sets of variables:

- q - distribution of latent
- θ - parameters of generator



Coordinate descent:

- max-max procedure (GAN: max-min)

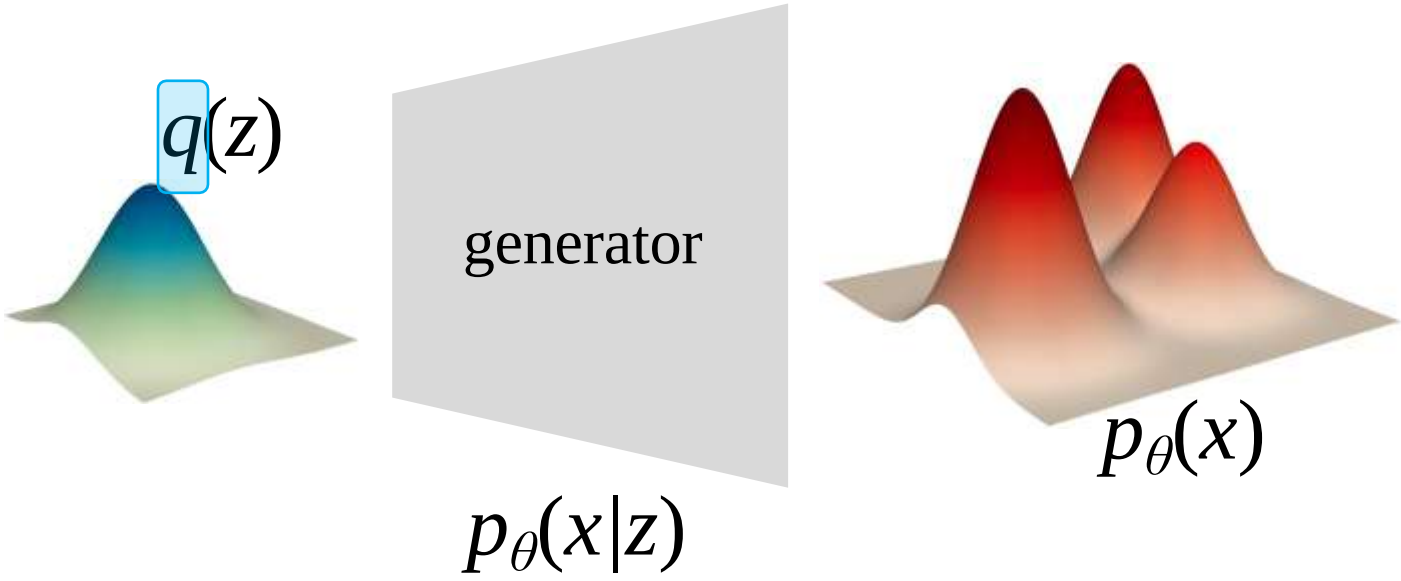
EM as A Max-Max Procedure

$$\text{ELBO} = \mathbb{E}_{x \sim p_{data}(x)} \left[\mathbb{E}_{z \sim q(z|x)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\text{KL}} \left(q(z|x) || p(z) \right) \right]$$

$$\max_{\theta, q} \text{ELBO}(\theta, q(\cdot))$$

E-step: optimize for q

$$q^{(t)} = p_{\theta^{(t)}}(z|x)$$



Details can be found in: Hastie et al. “The Elements of Statistical Learning”.

EM as A Max-Max Procedure

$$\text{ELBO} = \mathbb{E}_{x \sim p_{data}(x)} \left[\mathbb{E}_{z \sim q(z|x)} \left[\log p_\theta(x|z) \right] - \mathcal{D}_{\text{KL}} \left(q(z|x) || p(z) \right) \right]$$

$$\max_{\theta, q} \text{ELBO}(\theta, q(\cdot))$$

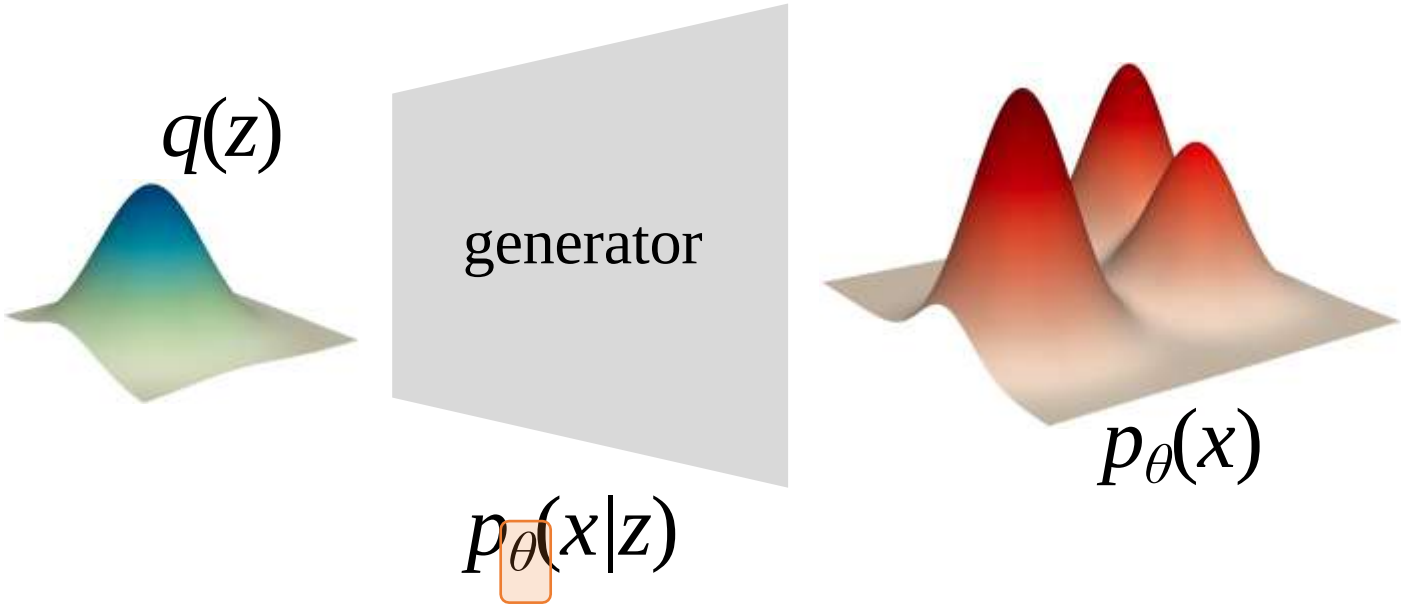
E-step: optimize for q

$$q^{(t)} = p_{\theta^{(t)}}(z|x)$$

M-step: optimize for θ

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$

with sub-objective defined as: $Q(\theta|\theta^{(t)}) = \mathbb{E}_{p_{data}(x)} \mathbb{E}_{p_{\theta^{(t)}}(z|x)} [\log p_\theta(x, z)]$



Details can be found in: Hastie et al. “The Elements of Statistical Learning”.

EM as A Max-Max Procedure

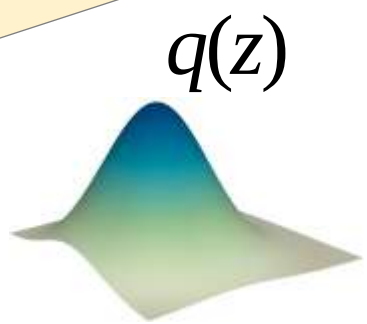
$$\text{ELBO} = \mathbb{E}_{x \sim p_{data}(x)} \left[\mathbb{E}_{z \sim q(z|x)} \left[\log p_\theta \right. \right.$$

$$\max_{\theta, q} \text{ELBO}(\theta, q(\cdot))$$

q: often in analytical forms
- Gaussian Mixtures
- **K-means**

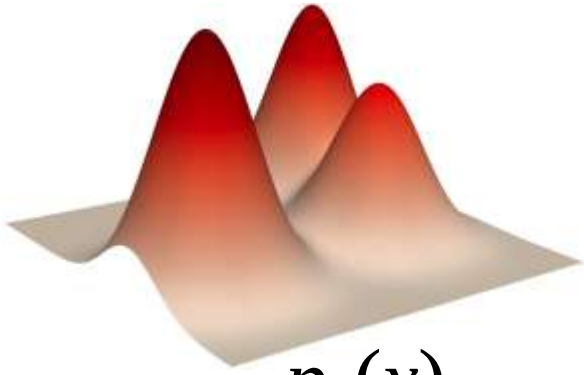
E-step: optimize for q

$$q^{(t)} = p_{\theta^{(t)}}(z|x)$$



generator

$p_\theta(x|z)$



M-step: optimize for θ

$$\theta^{(t+1)} = \arg\max_\theta Q(\theta|\theta^{(t)})$$

with sub-objective defined as: $Q(\theta|\theta^{(t)}) = \mathbb{E}_{p_{data}(x)} \mathbb{E}_{p_{\theta^{(t)}}(z|x)} [\log p_\theta(x, z)]$

Details can be found in: Hastie et al. “The Elements of Statistical Learning”.

A running example of EM: K-means

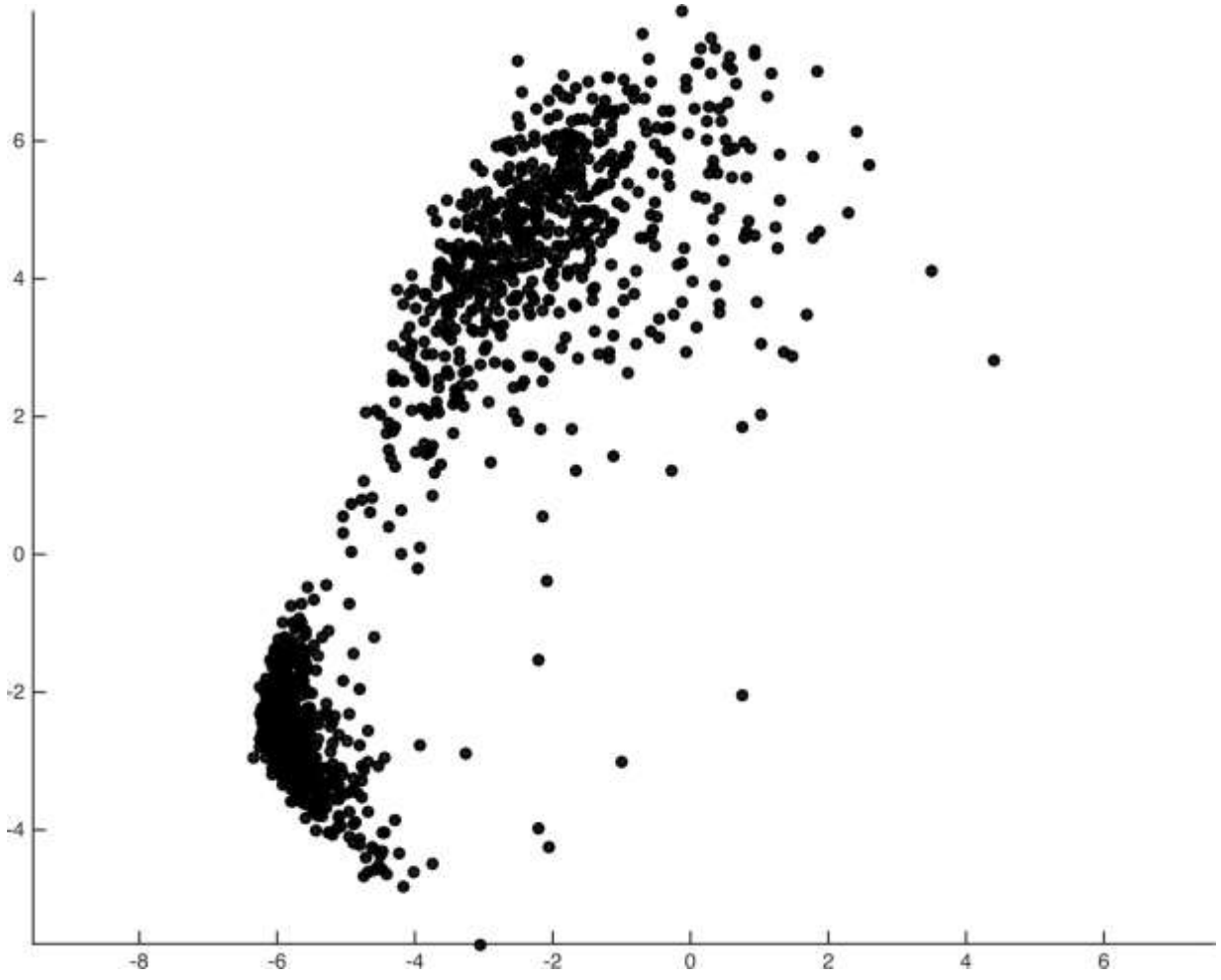


Figure adapted from: Marion Neumann, https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote20.html

A running example of EM: K-means

- cluster centers: θ

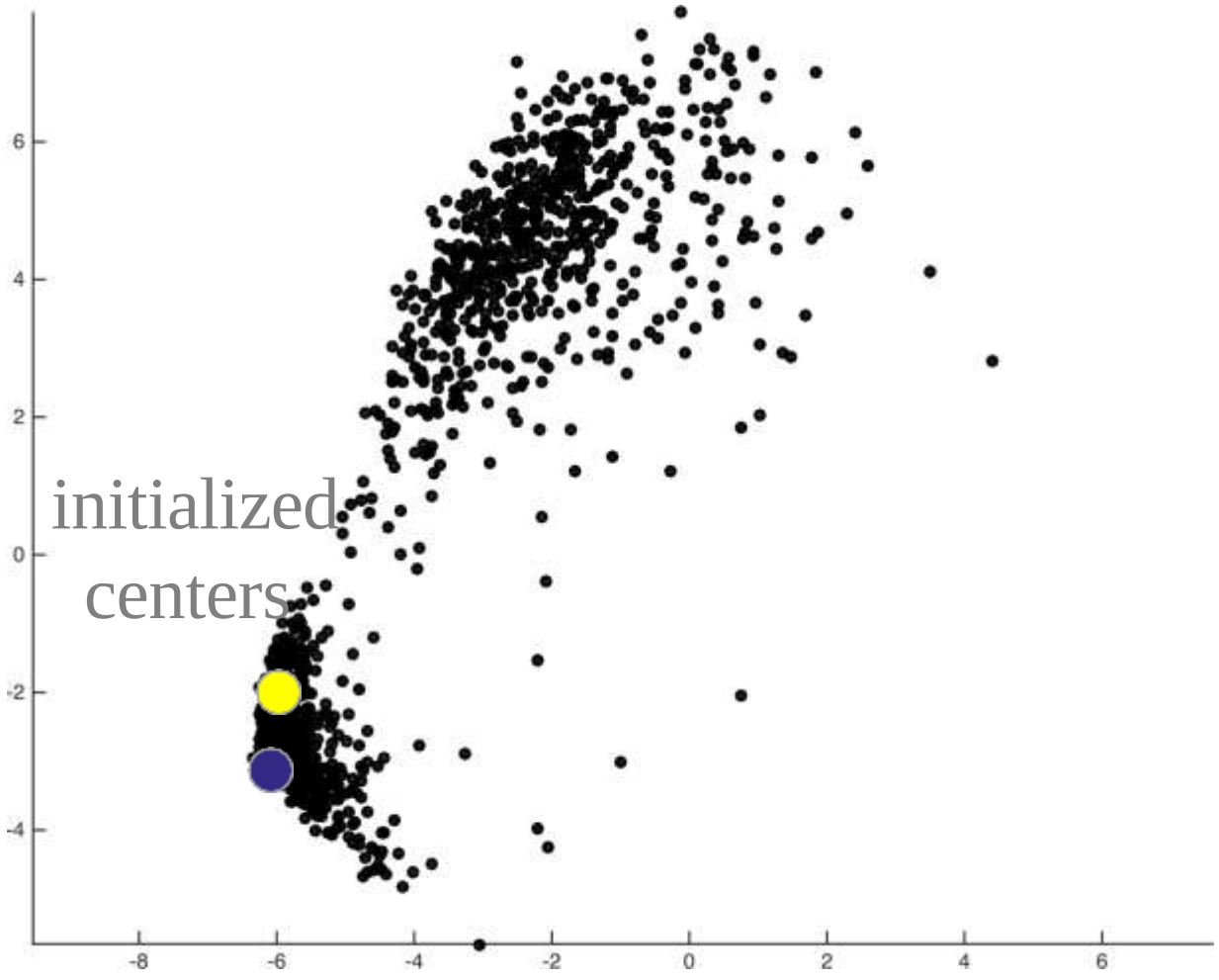


Figure adapted from: Marion Neumann, https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote20.html

A running example of EM: K-means

- cluster centers: θ
- assignment:

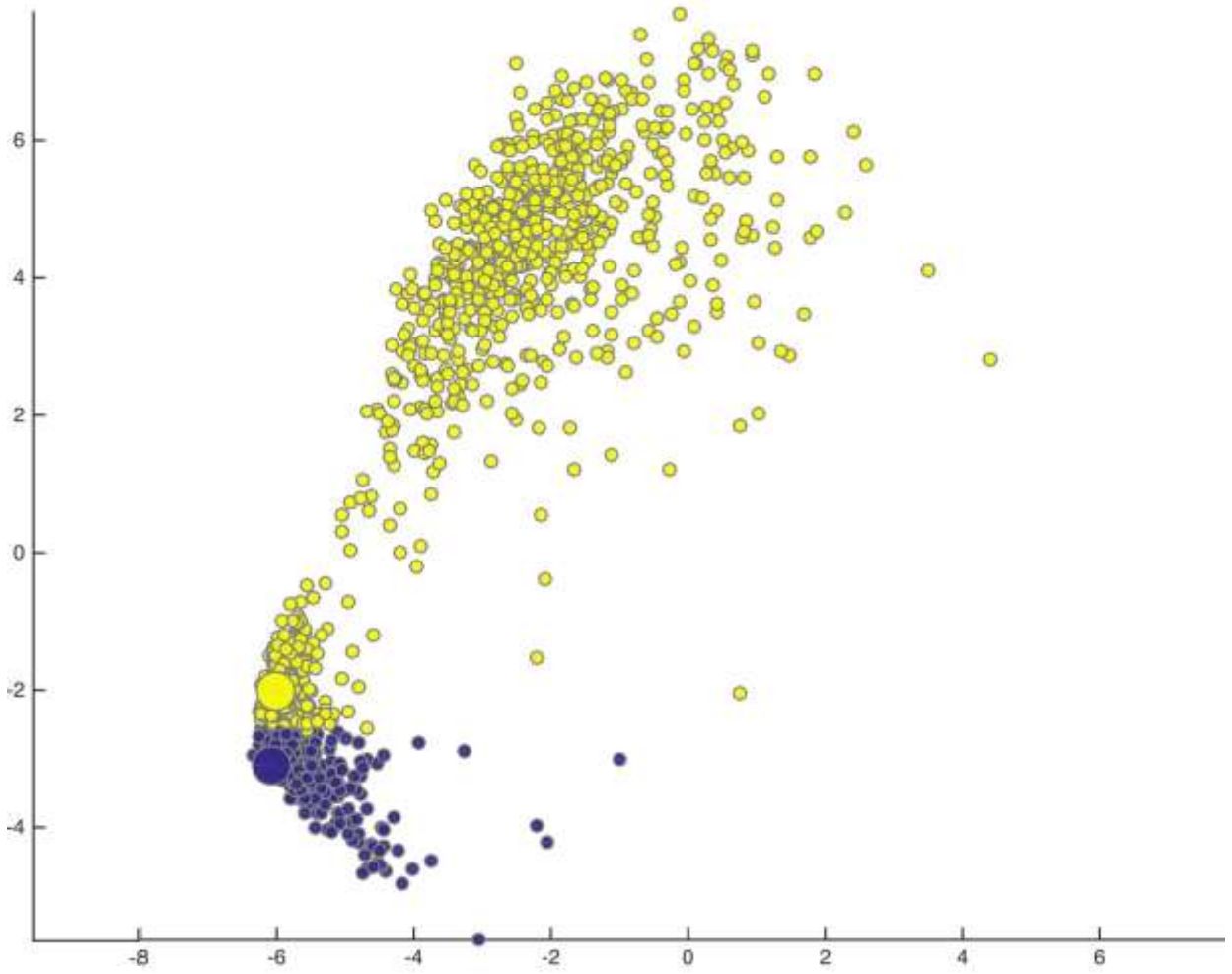


Figure adapted from: Marion Neumann, https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote20.html

A running example of EM: K-means

- cluster centers: θ
- assignment: E-step

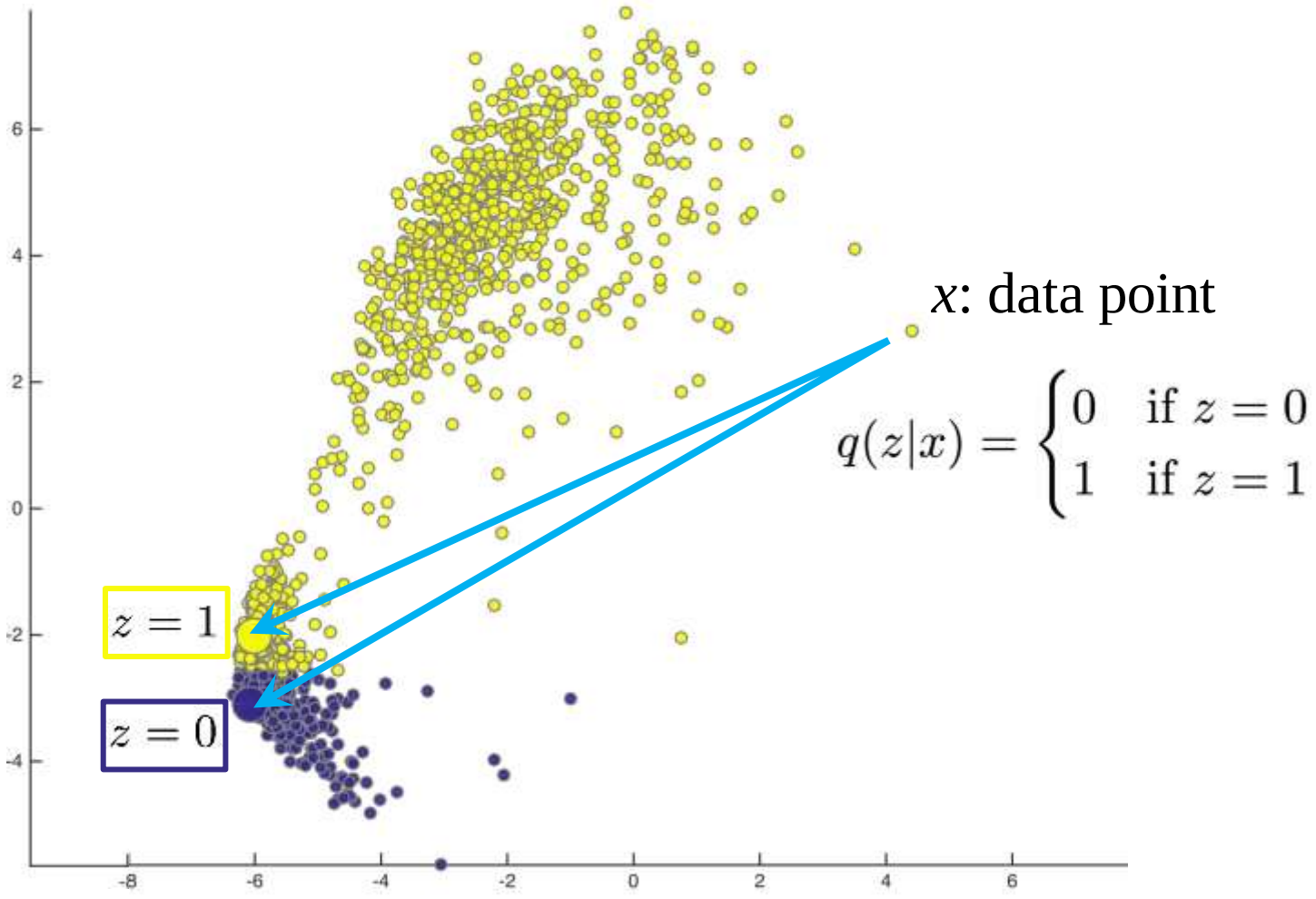


Figure adapted from: Marion Neumann, https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote20.html

A running example of EM: K-means

- cluster centers: θ
- assignment: E-step

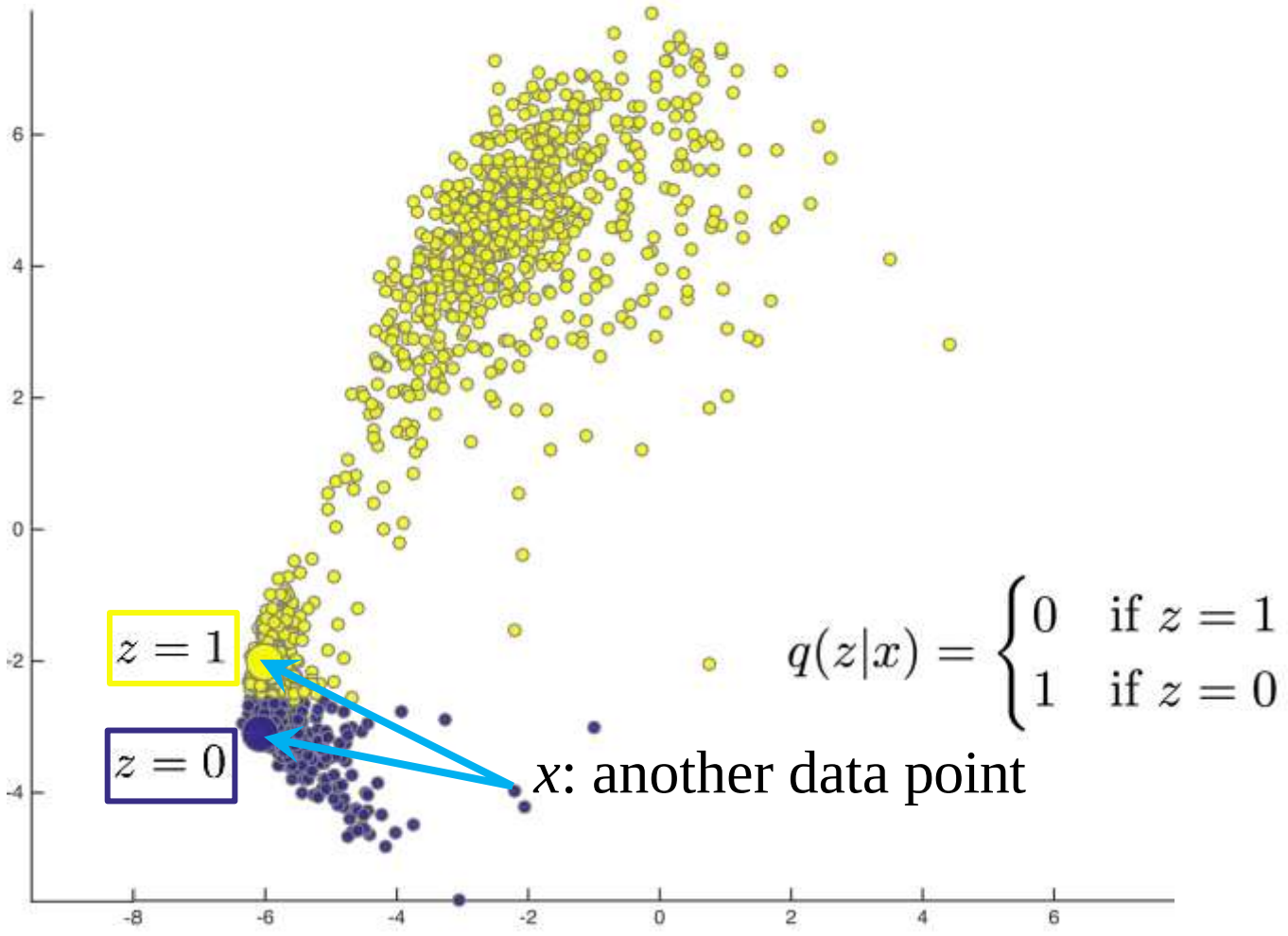
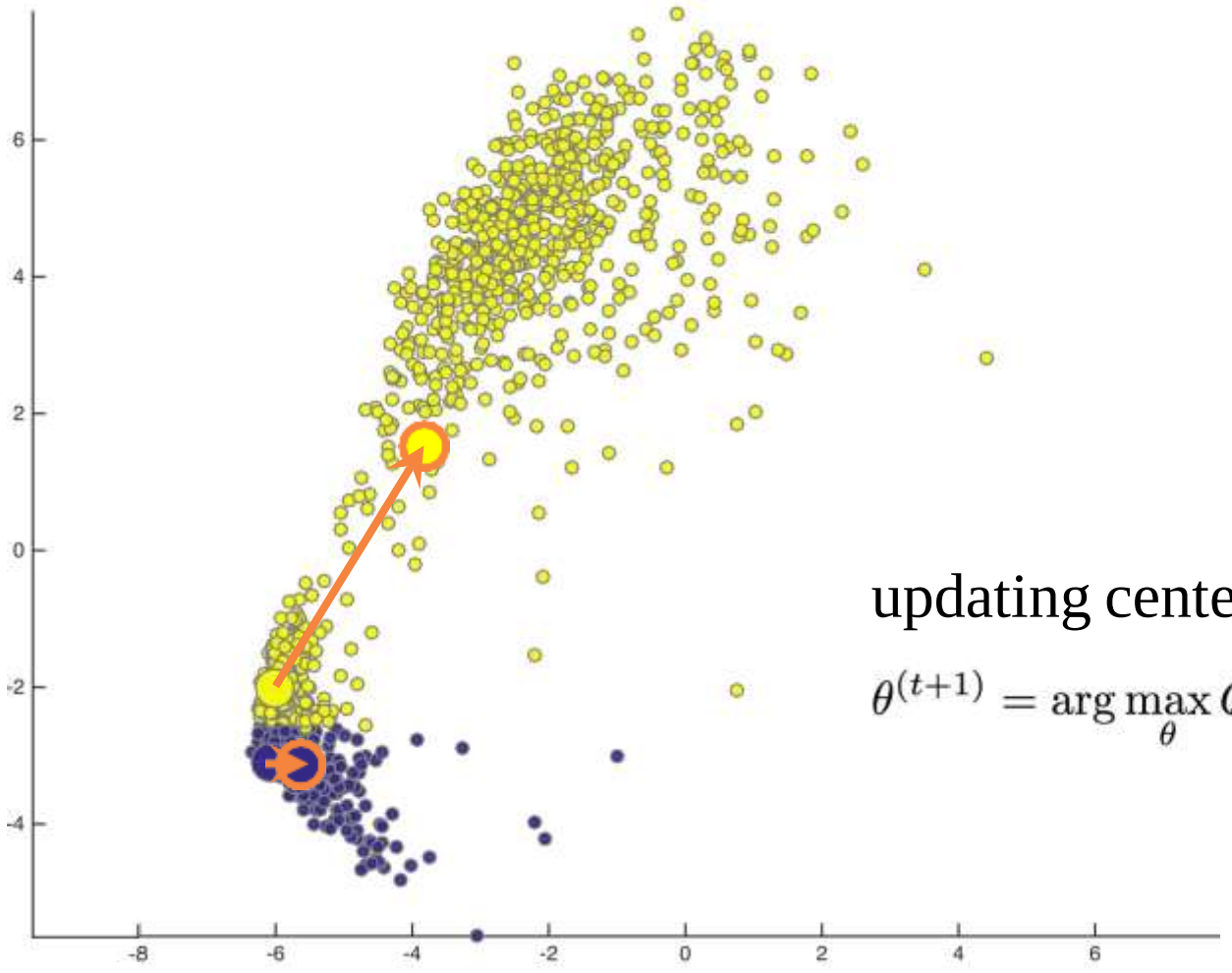


Figure adapted from: Marion Neumann, https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote20.html

A running example of EM: K-means

- cluster centers: θ
- assignment: E-step
- update: M-step



updating centers by:

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)})$$

Figure adapted from: Marion Neumann, https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote20.html

A running example of EM: K-means

- cluster centers: θ
- assignment: E-step
- update: M-step

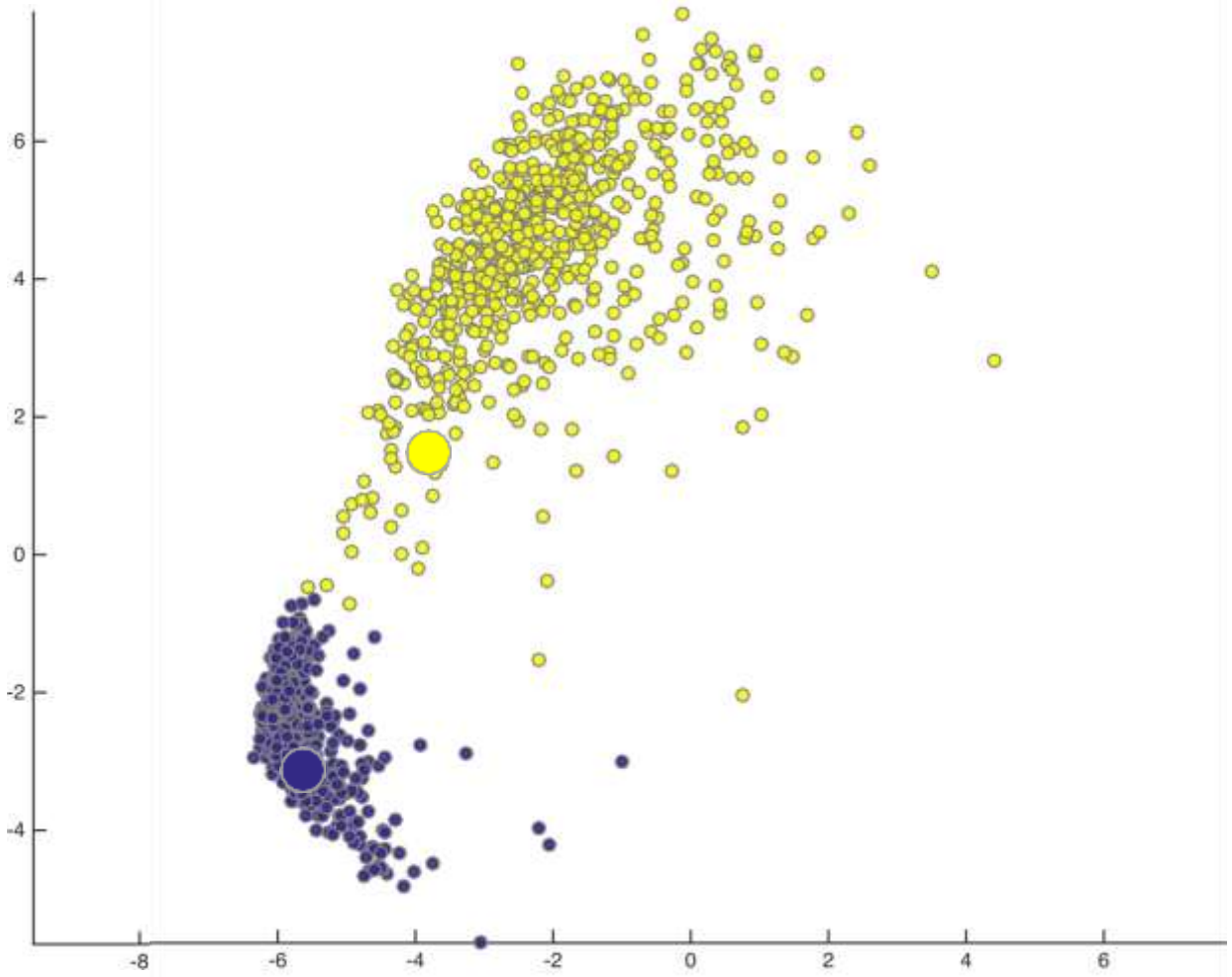
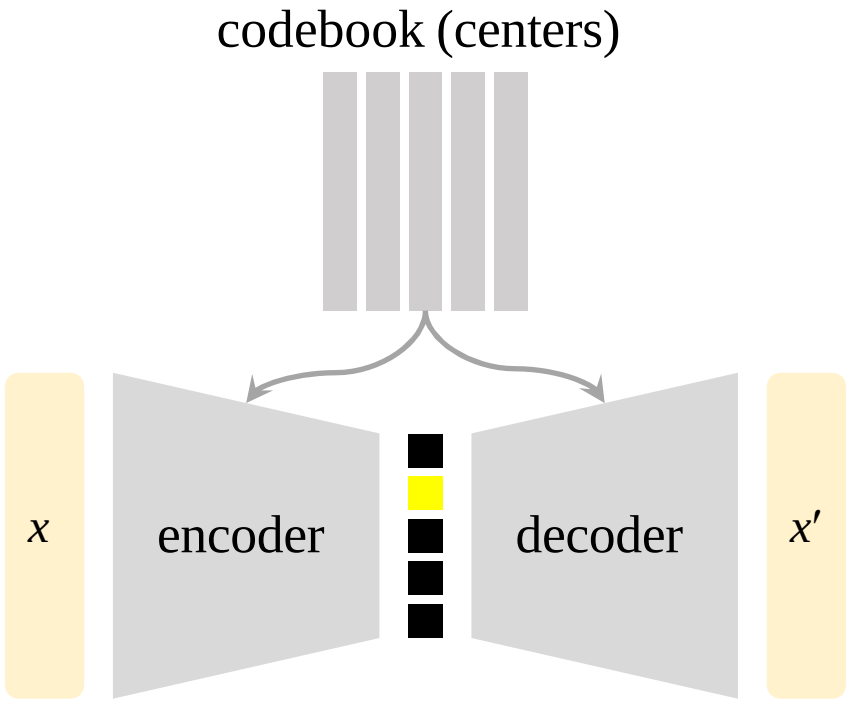
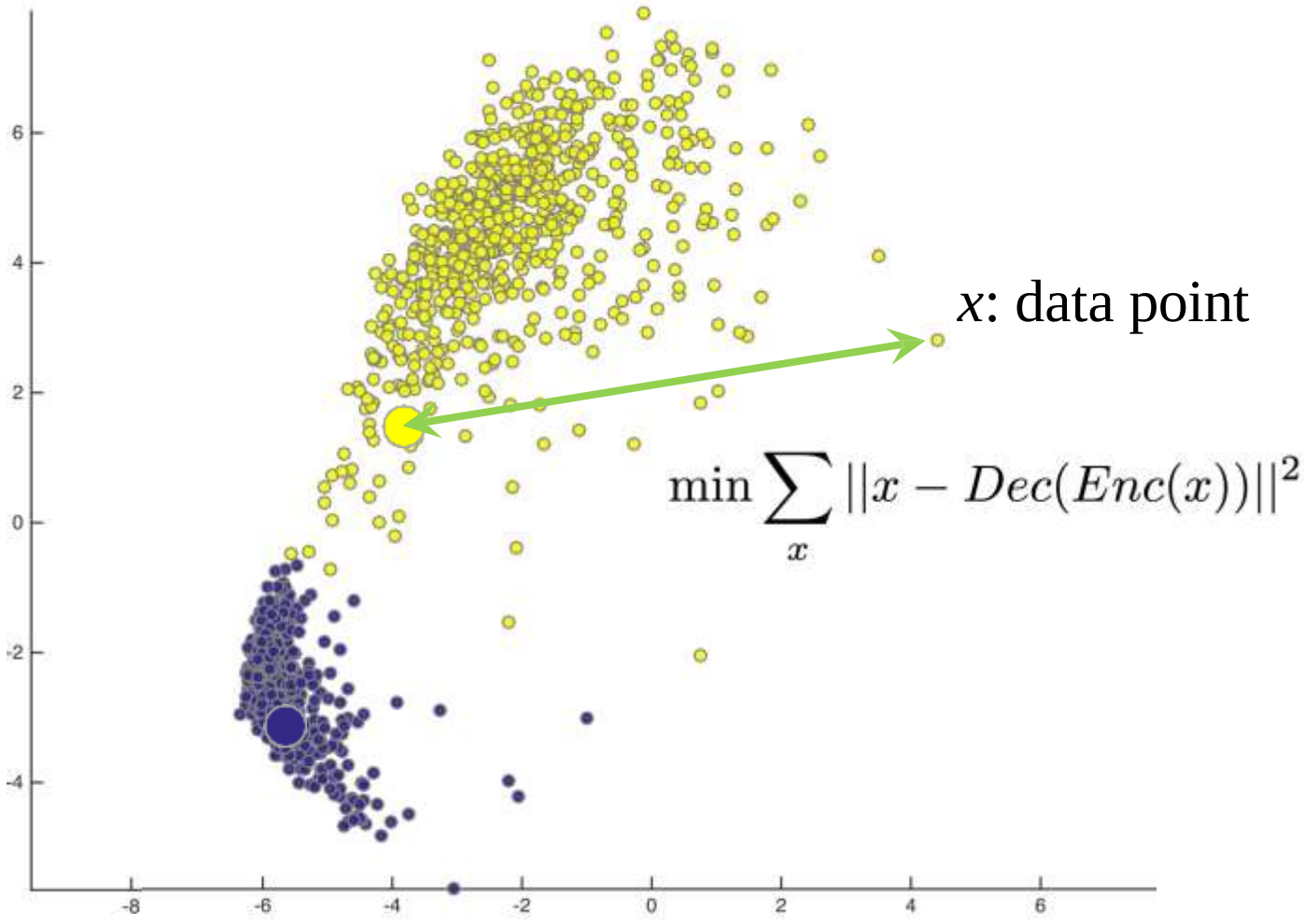


Figure adapted from: Marion Neumann, https://www.cse.wustl.edu/~m.neumann/sp2016/cse517/lecturenotes/lecturenote20.html

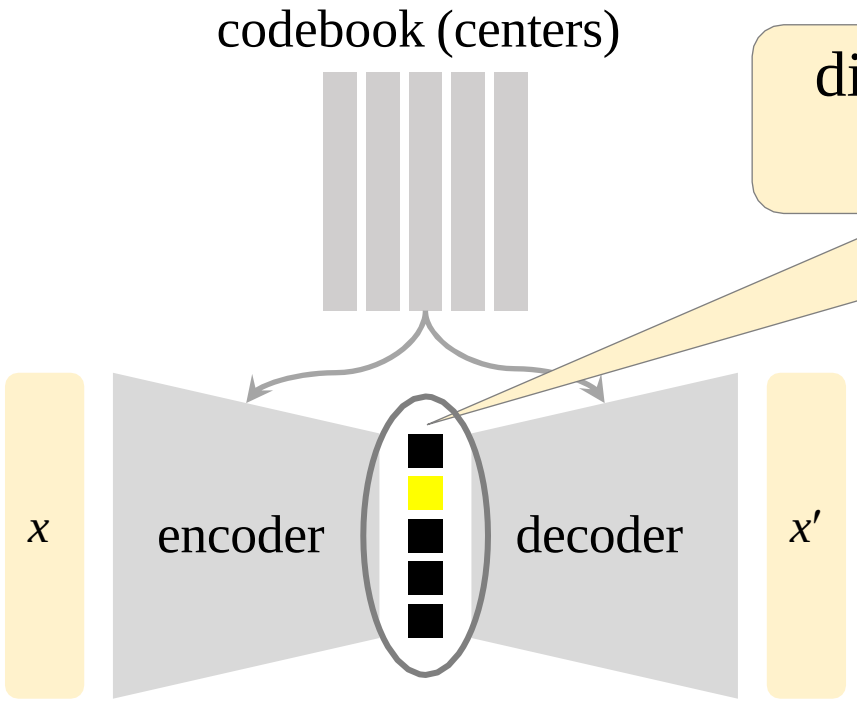
K-means as **Autoencoder**



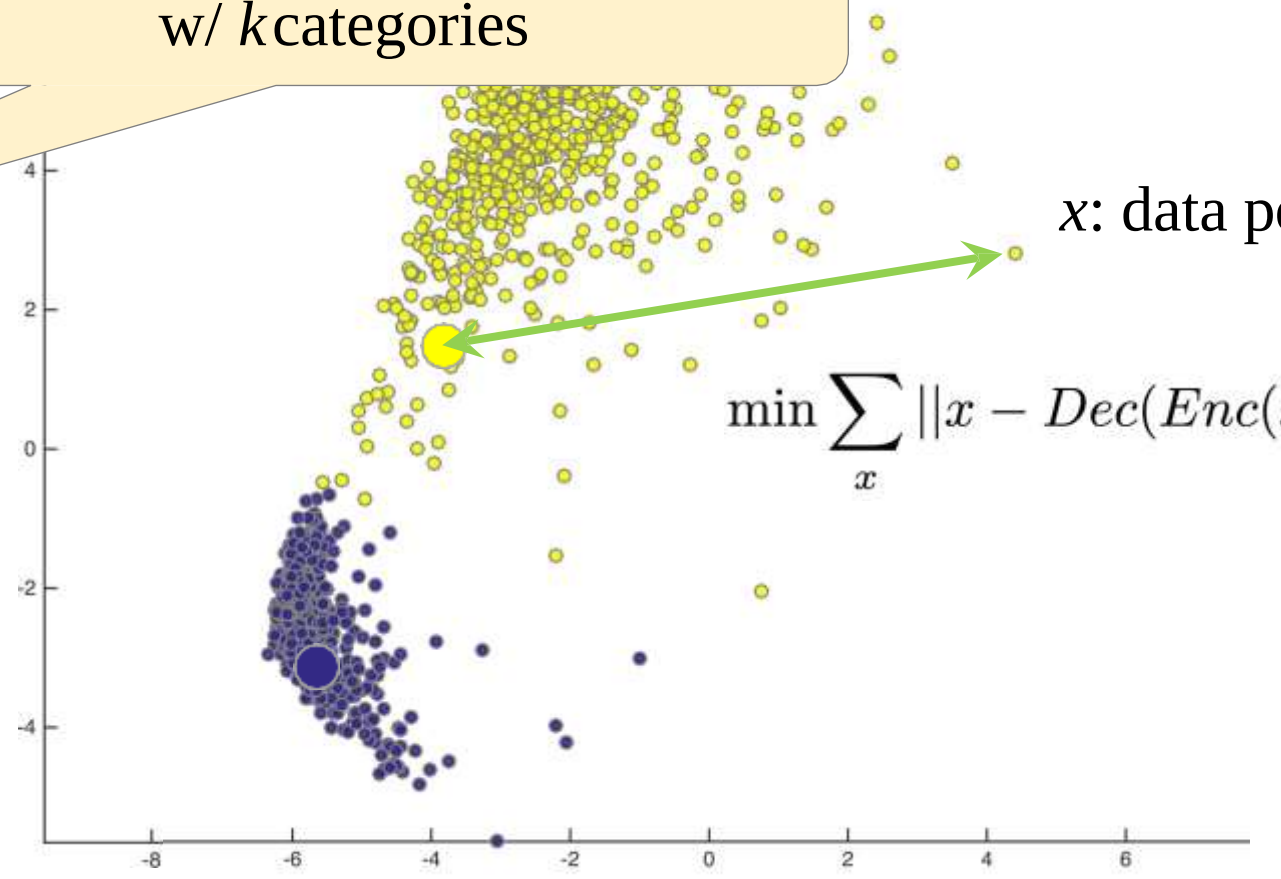
- encode: map x to one-hot
- decode: map one-hot to x'
- x' is a center



K-means as **Autoencoder**



discrete (categorical) distribution
w/ k categories

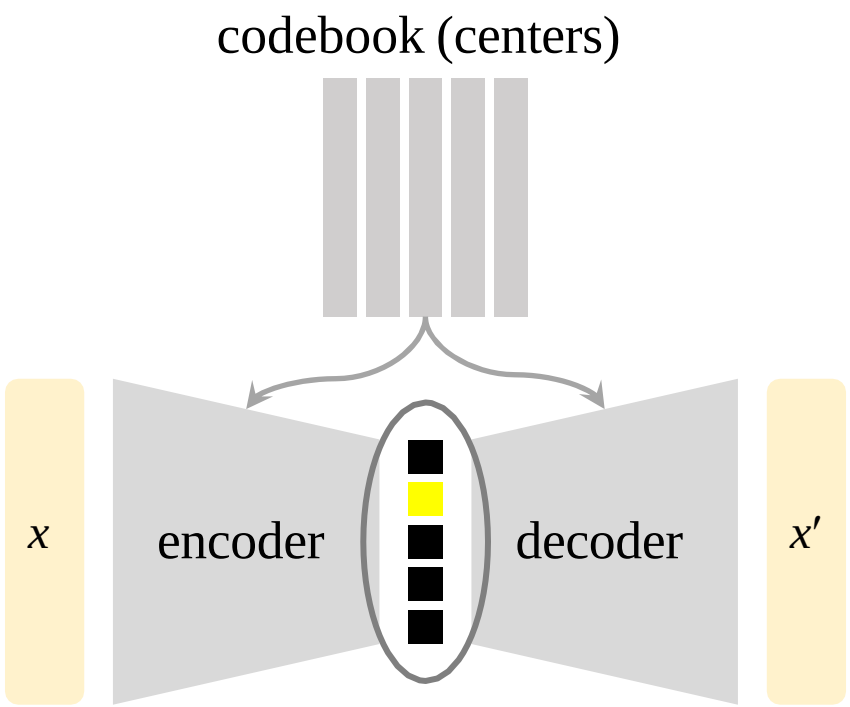


x: data point

$$\min \sum_x ||x - Dec(Enc(x))||^2$$

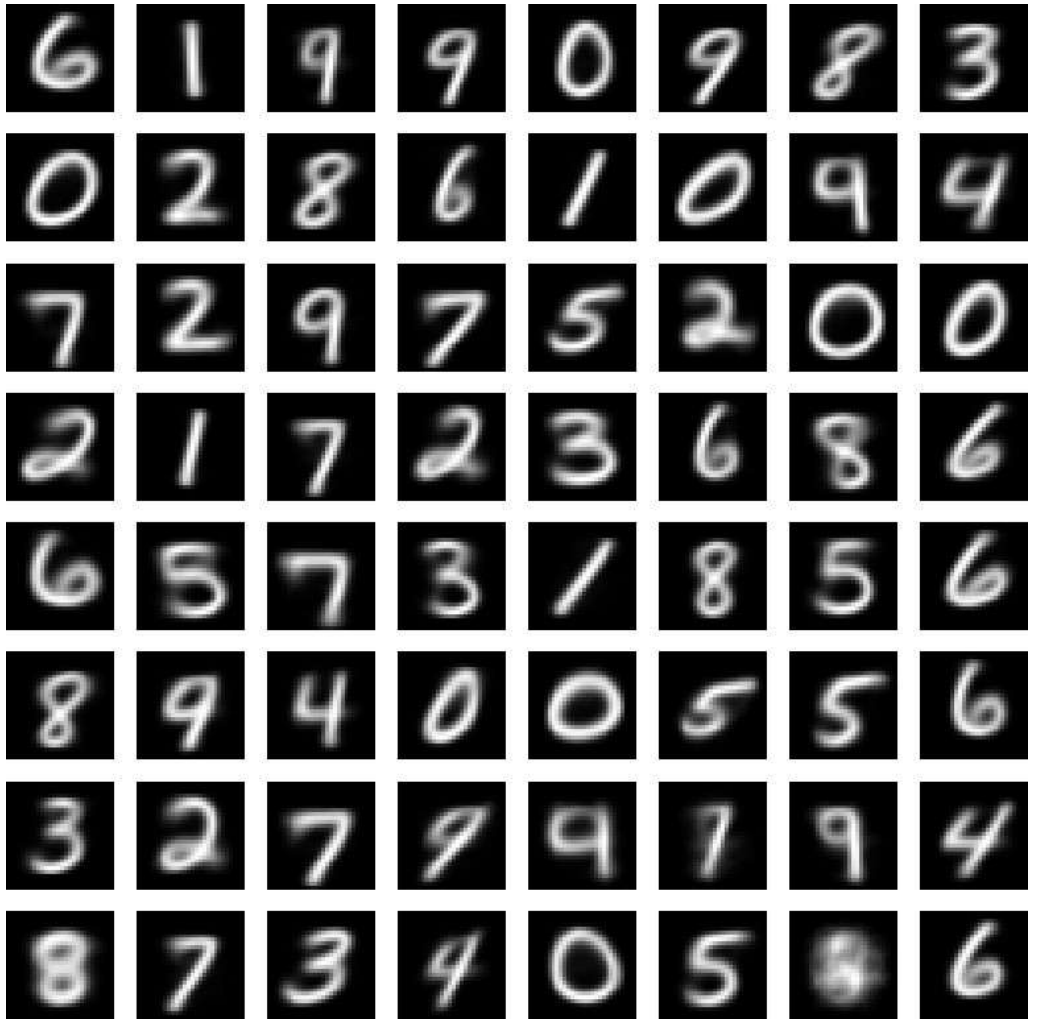
- encode: map x to one-hot
- decode: map one-hot to x'
- x' is a center

K-means as **Autoencoder**



- encode: map x to one-hot
- decode: map one-hot to x'
- x' is a center

codebook on MNIST, $k = 64$

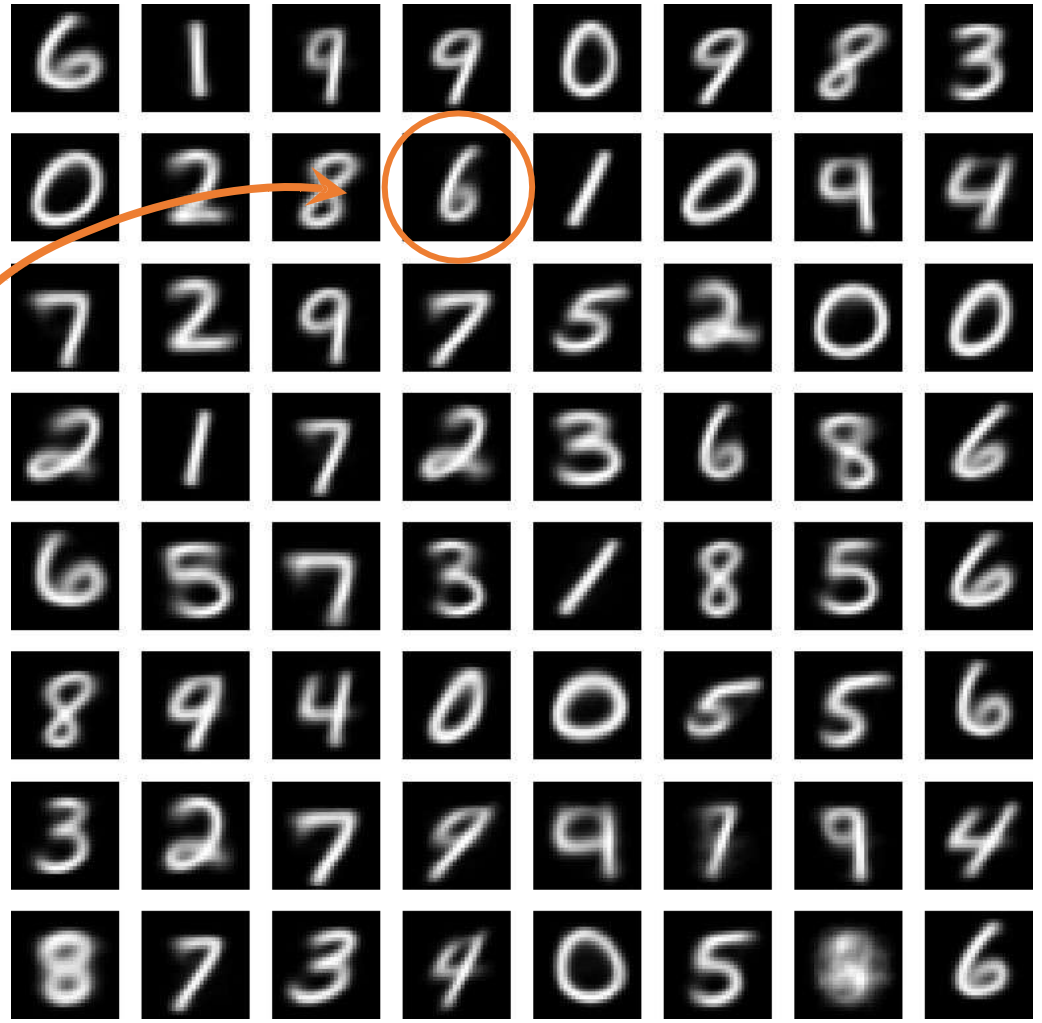


K-means as **Generative Models**

- randomly sample: $z \sim \mathcal{U}[0, k)$
- map z by the decoder
- generation result is one codeword

$z = 11$

codebook on MNIST, $k = 64$



- this is a valid generative model
- but not a “good” one
- but a good thought model

thus far, ...

- **VAE**: maximize ELBO
  - parameterize q by network
  - optimize by Stochastic Gradient Descent

- **EM**: maximize ELBO
  - parameterize q analytically
  - optimize by Coordinate Descent

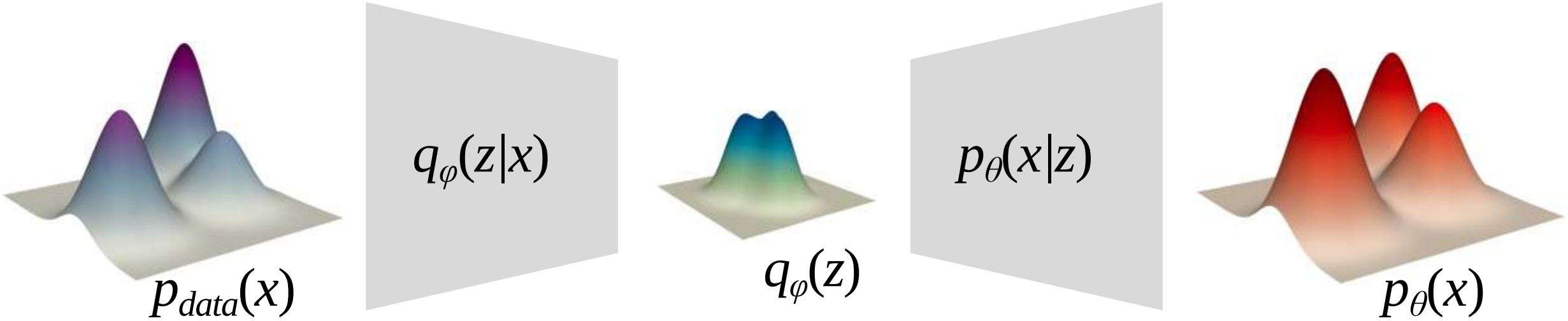
- **K-means**:
  - special case of EM; special case of AE
  - discrete distribution

- next: **VQ-VAE**

Vector Quantized VAE (VQ-VAE)

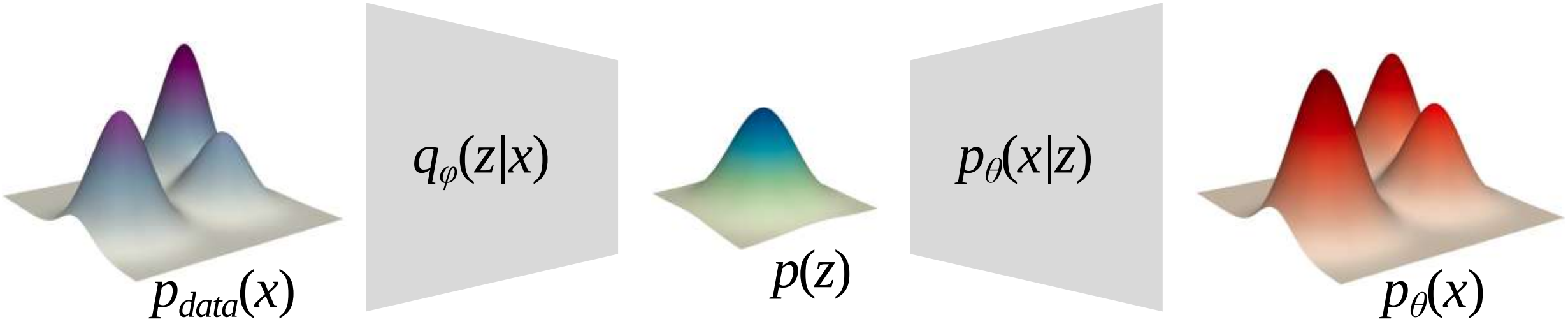
Recap

- Original VAE: latent variables are continuous



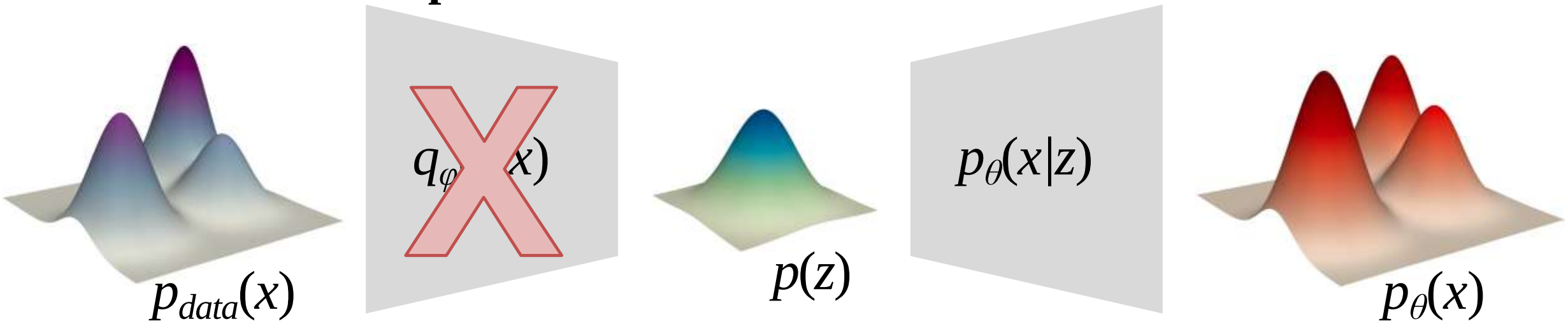
VAE

- encoder: maps data distribution to latent distribution
- decoder: maps latent distribution to data distribution



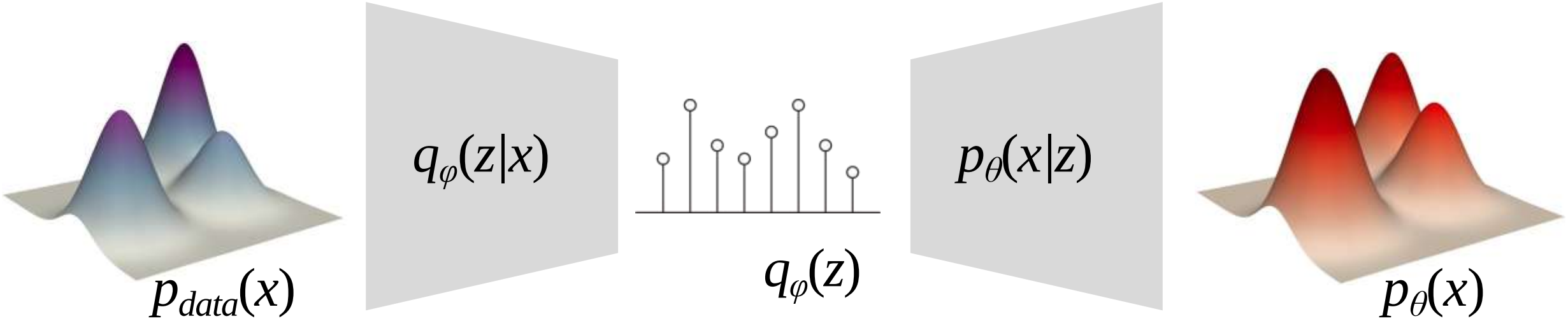
VAE

- encoder: maps data distribution to latent distribution
- decoder: maps latent distribution to data distribution
- **Posterior Collapse**



Discrete Latent Variables

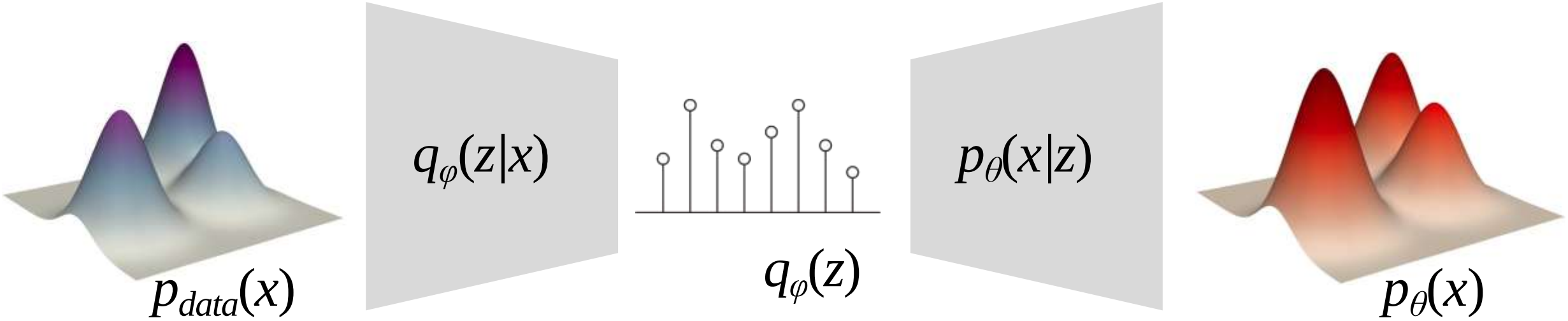
- model **multimodal** distributions
- **categorical**: no particular relation between numbers (SSN, zip code, ...)
- **symbolic**: language, speech, planning, ...



Discrete Latent Variables + VAE

Maximize ELBO

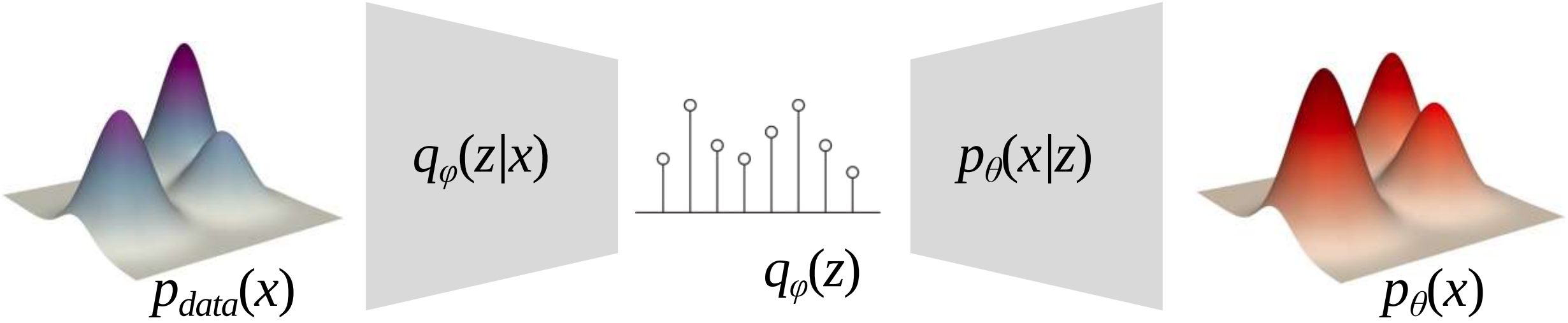
- Reconstruction loss: about x
- Regularization loss: about z (discrete)



Discrete Latent Variables + VAE

Reconstruction loss: about x

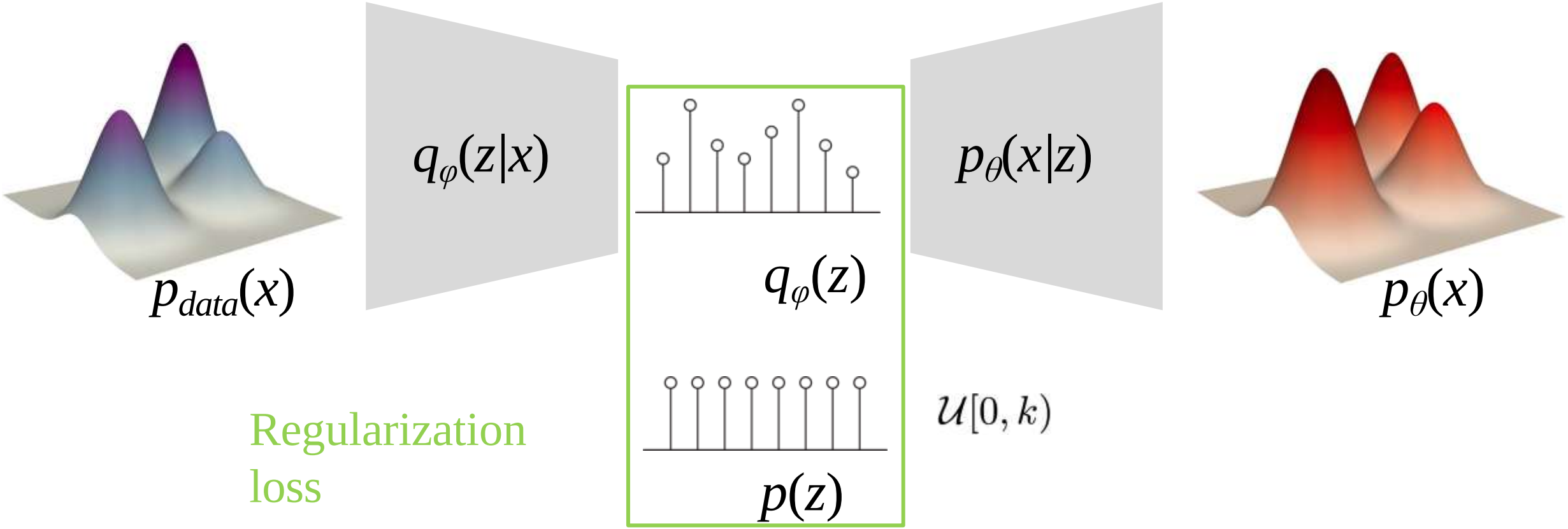
- same as VAE: $-\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right]$



Discrete Latent Variables + VAE

Regularization loss: about z

- conceptually, same as VAE: $\mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$
- but how can we backprop w.r.t. discrete sampling?



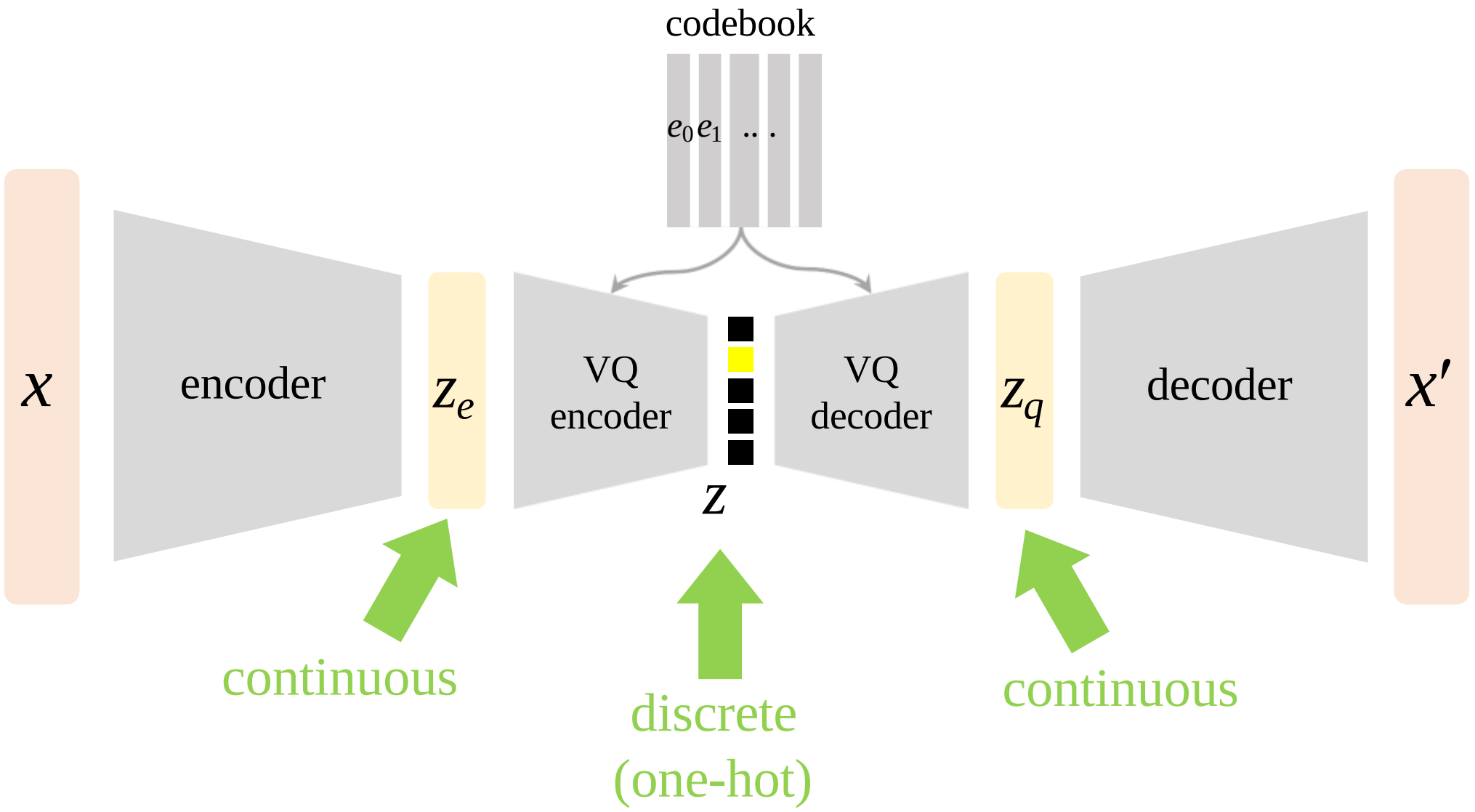
Discrete Latent Variables + VAE

Solution: K-means

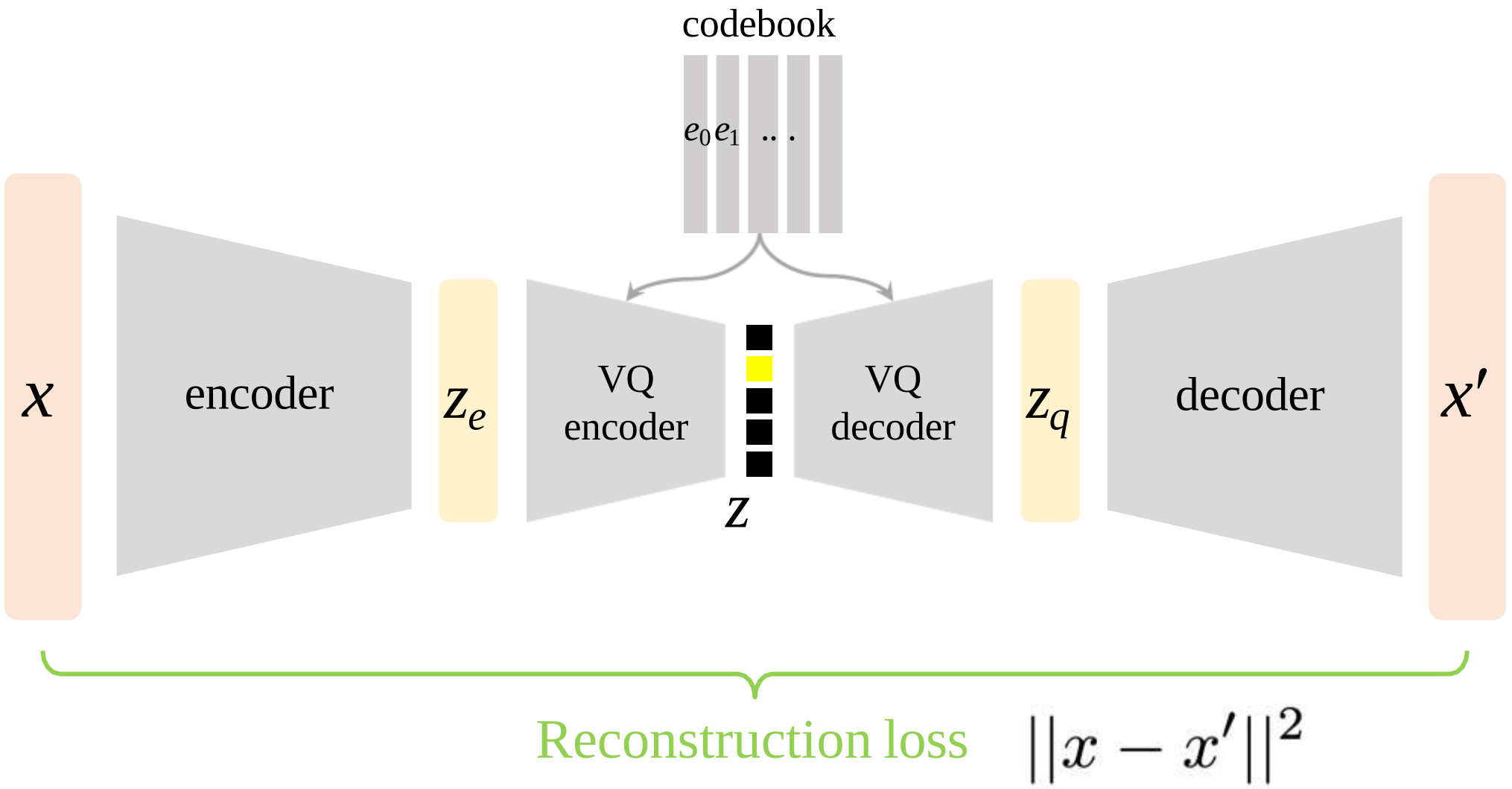
- K-means is autoencoding
- K-means has an objective function (reconstruction loss)
- K-means implicitly encourages codebook uniformity

This leads us to VQ-VAE ...

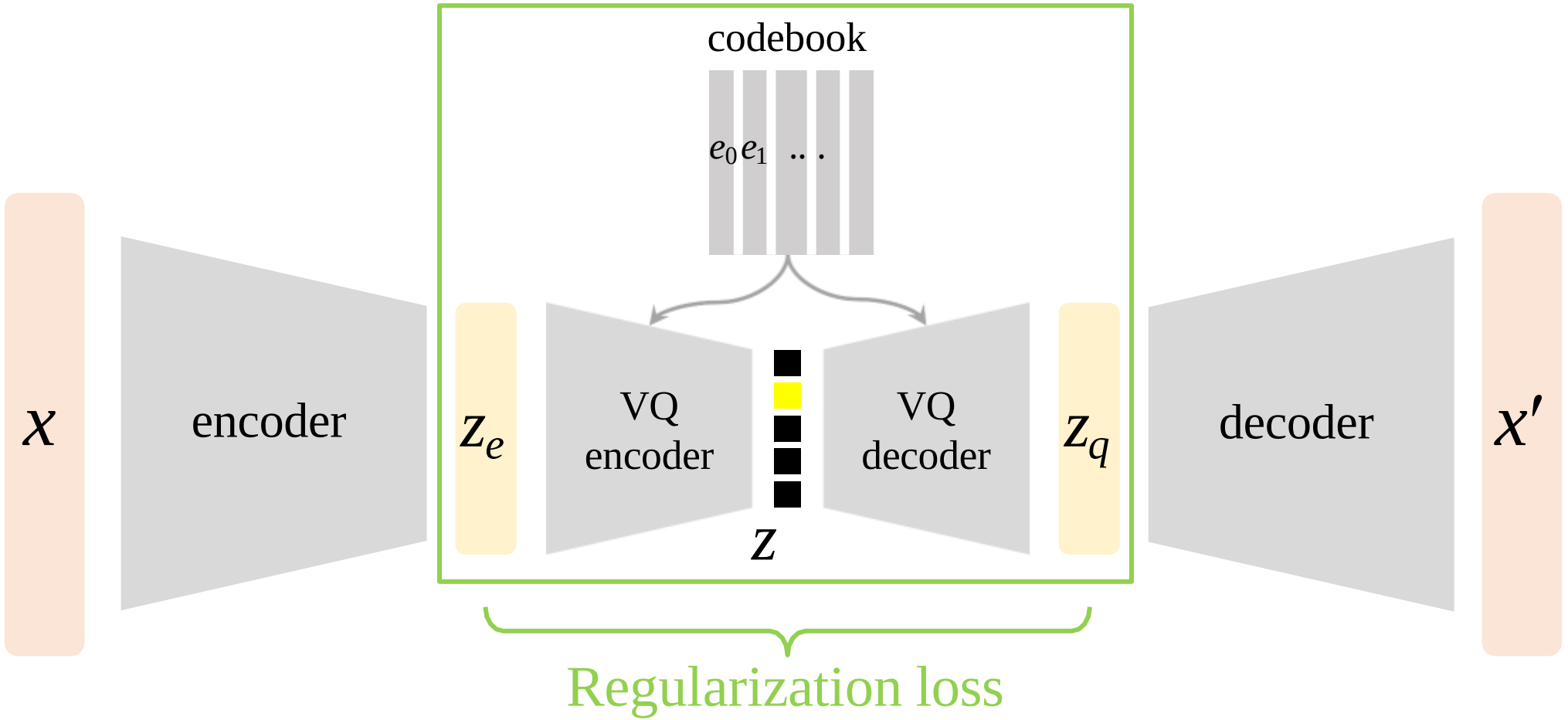
Vector Quantized VAE



Vector Quantized VAE



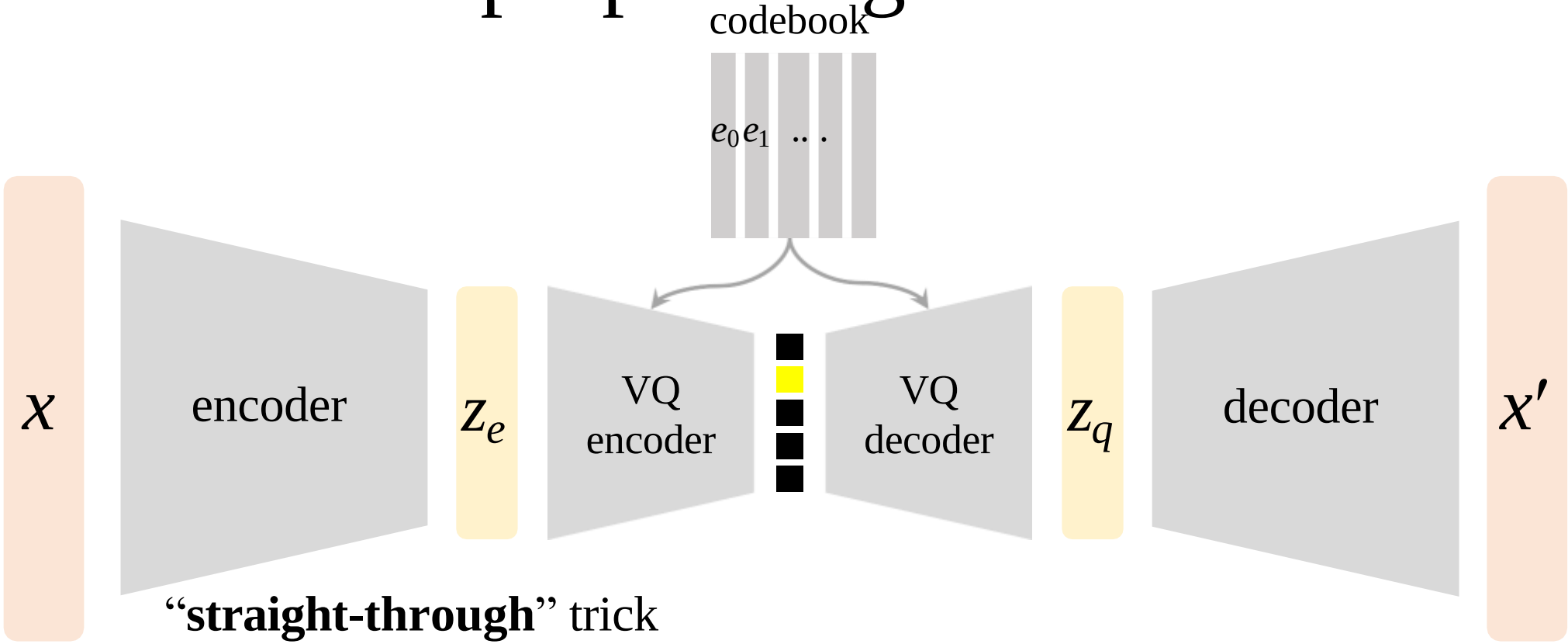
Vector Quantized VAE



conceptually, this is the K-means reconstruction loss: $\|z_e - z_q\|^2$

*The VQ-VAE paper uses $\|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2$ which weights the gradients differently

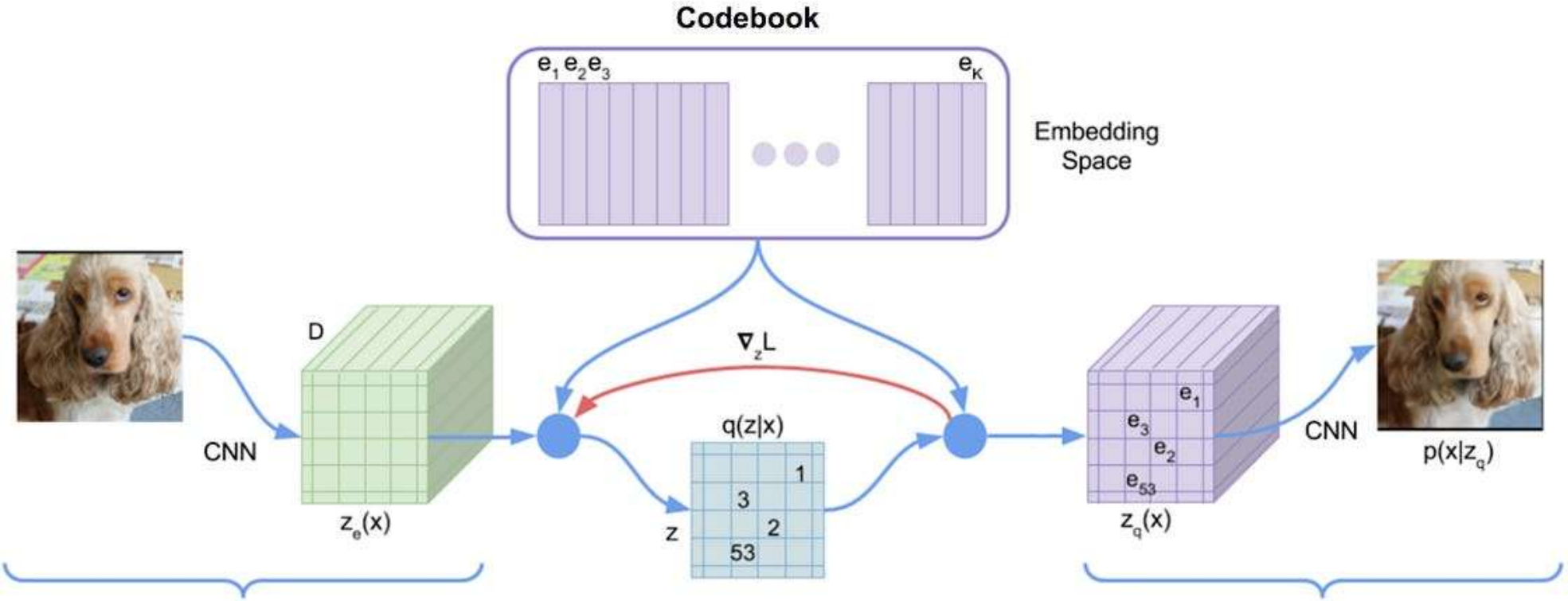
How to backprop through one-hot vector?

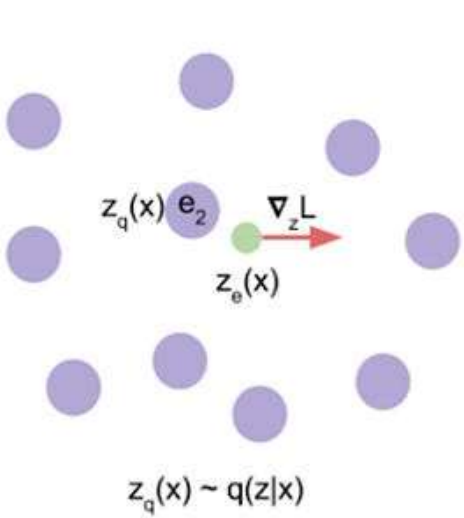


"**straight-through**" trick

- forward: hardmax's output (i.e., argmax and one-hot)
- backward: softmax's gradient
- in code: `stop_grad(hardmax(y) - softmax(y)) + softmax(y)`

VQ-VAE

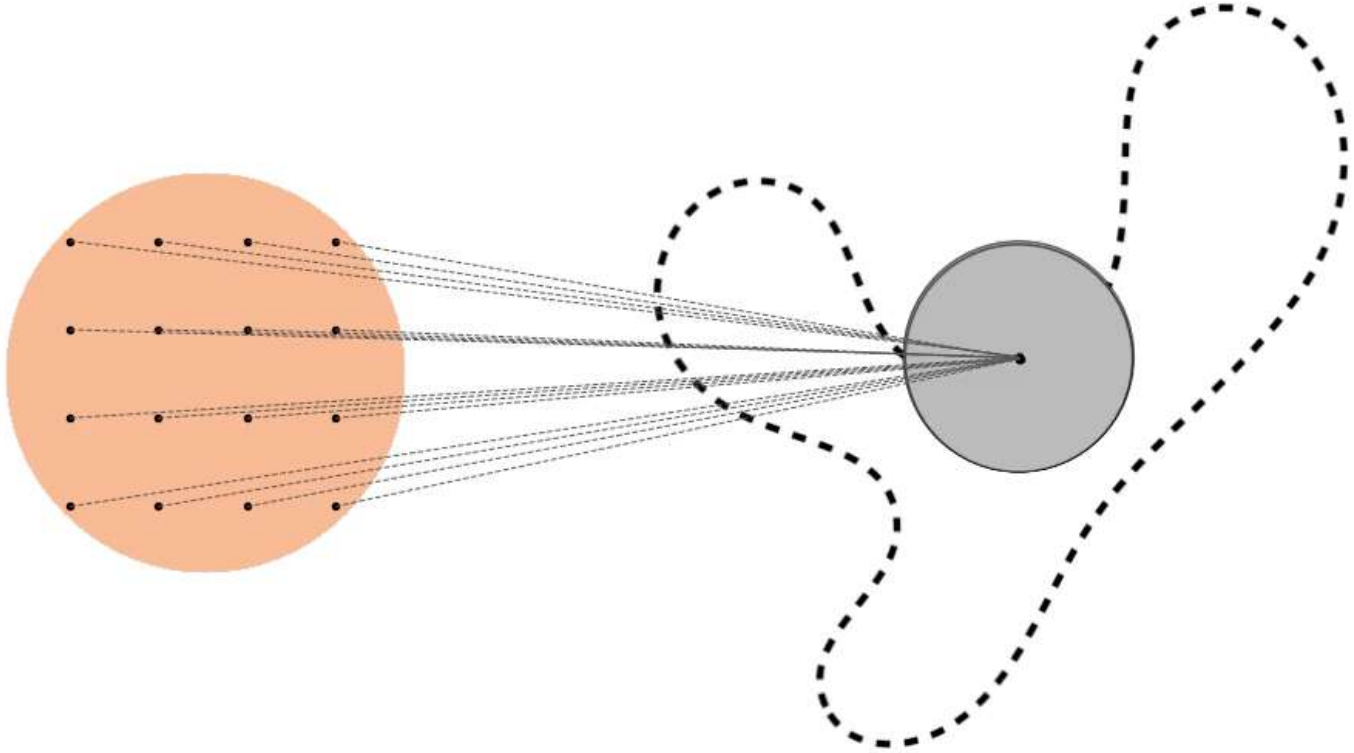




Posterior categorical distribution:

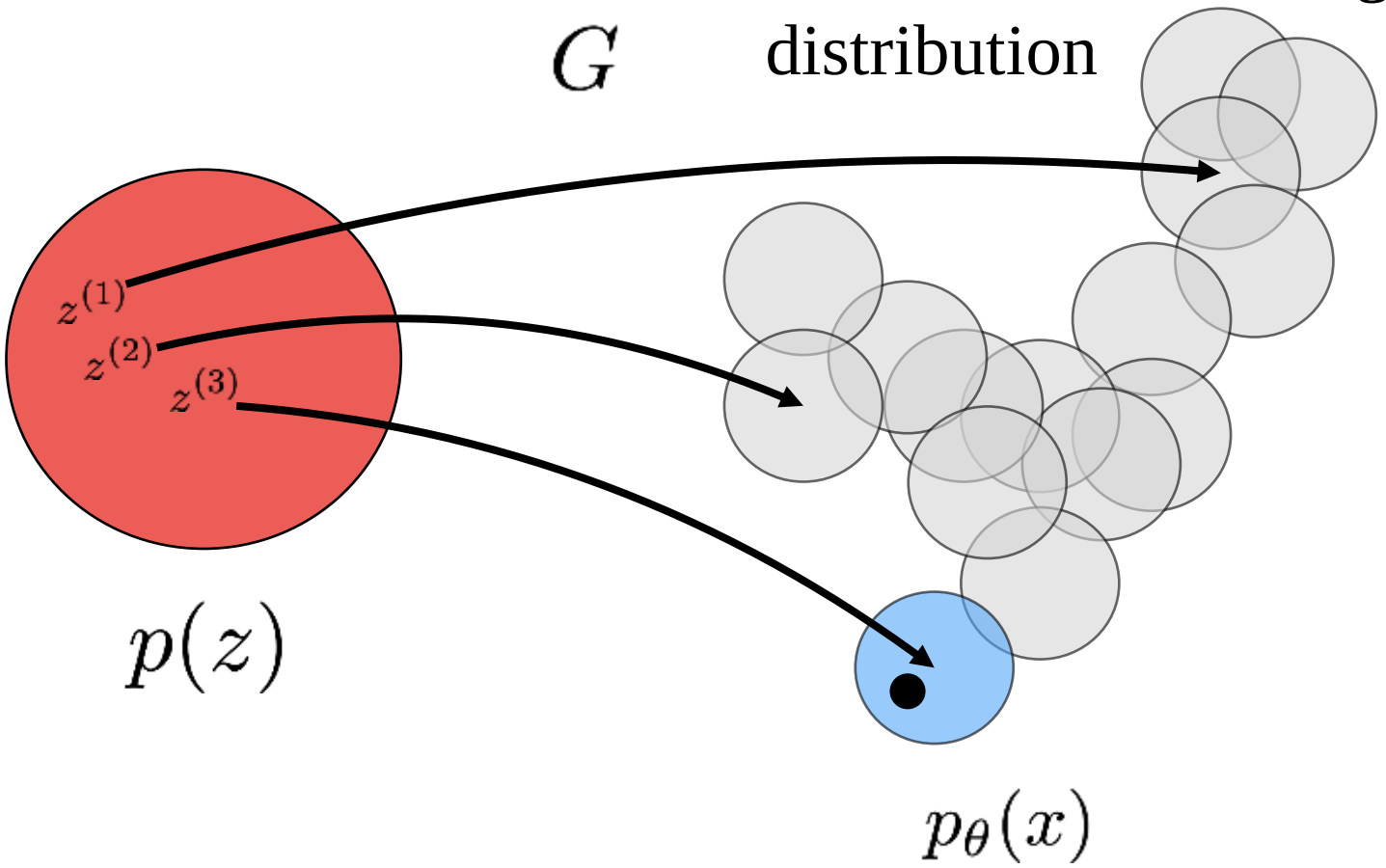
$$q(\mathbf{z}=\mathbf{e}_k|\mathbf{x}) = \begin{cases} 1 & \text{if } k = \arg\min_i \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_i\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

Recap. VAE



Prior distribution

Current model of target distribution



In order to optimize our model, we need to measure the likelihood it assigns to each datapoint x

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

$$= \sim 0+$$

$$\sim 0+$$

$$p(x|z^{(3)})p(z^{(3)})dz + \ldots$$

Vector Quantized VAE

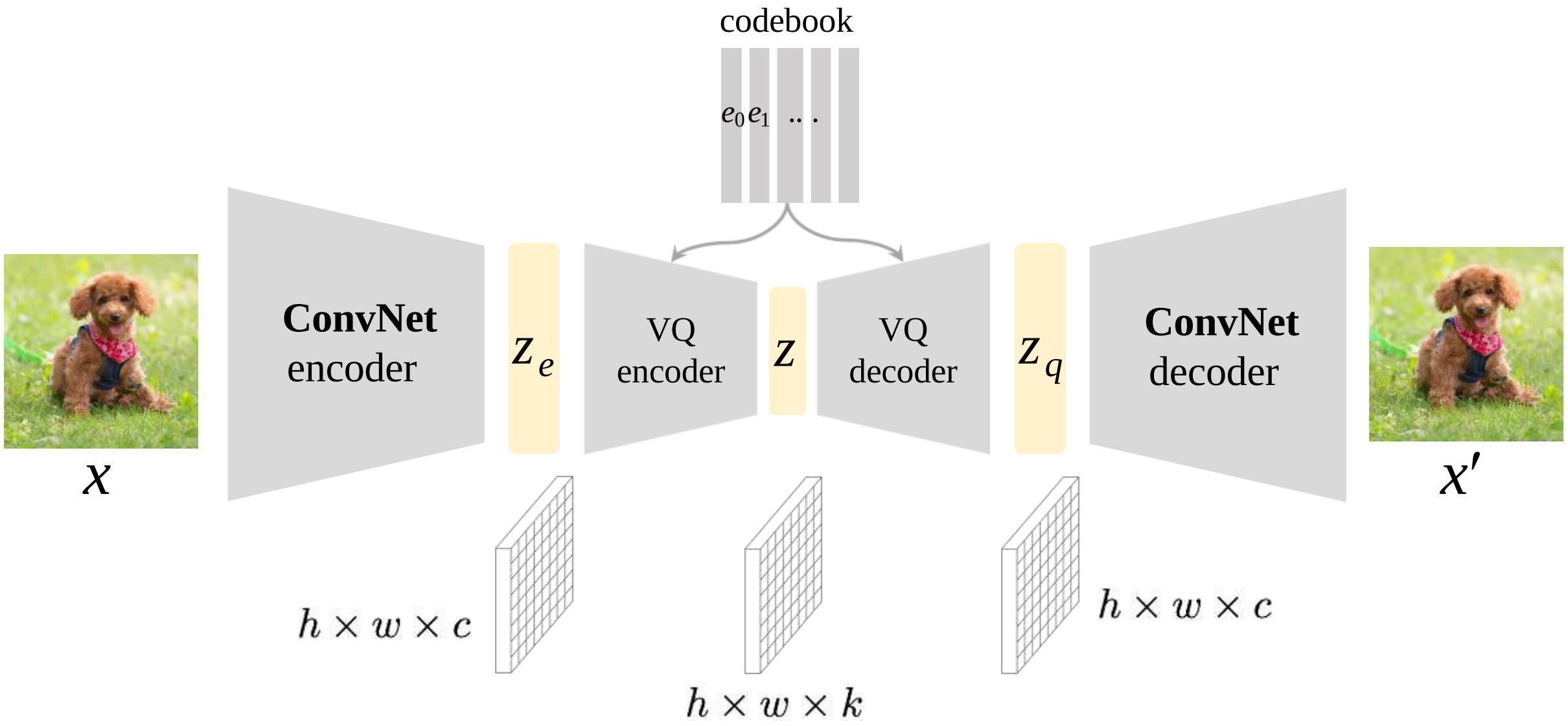
A single one-hot latent is not useful

- it's "deep K-means": with deep encoder/decoder
- a valid generative model; but not a "good" one

VQ-VAE: often used as "**tokenizers**"

- output multiple one-hot vectors
- don't reduce latent spatial/temporal size to 1
- use ConvNet/Transformer as encoder and decoder

VQ-VAE as Tokenizers



Notes

- Both VAE and VQ-VAE can be “tokenizers” (produce spatial latents).

But:

- prior $p(z)$ only models per-token (per-location) distribution
- prior $p(z)$ doesn’t model **joint** distribution across tokens
- spatial tokens are not **independent**
- at inference, we can’t sample from **i.i.d.** prior $p(z)$

Next: modeling joint distribution:

$$p(z) = p(z_1)p(z_2|z_1)p(z_3|z_1, z_2)p(z_4|z_1, z_2, z_3)\dots$$

- Autoregressive models
- Masked models
- Diffusion models

Main References

- Kingma and Welling. “Auto-Encoding Variational Bayes”, ICLR 2014
- Neal and Hinton. “A view of the EM algorithm that justifies incremental, sparse, and other variants”, 1999
- Hastie, et al. “The Elements of Statistical Learning”, 2001
- van den Oord, et al. “Neural Discrete Representation Learning”, NeurIPS 2017