Variational Autoencoder

https://hku-data8018.github.io/

Bo Dai

Spring 2026

- First come first serve

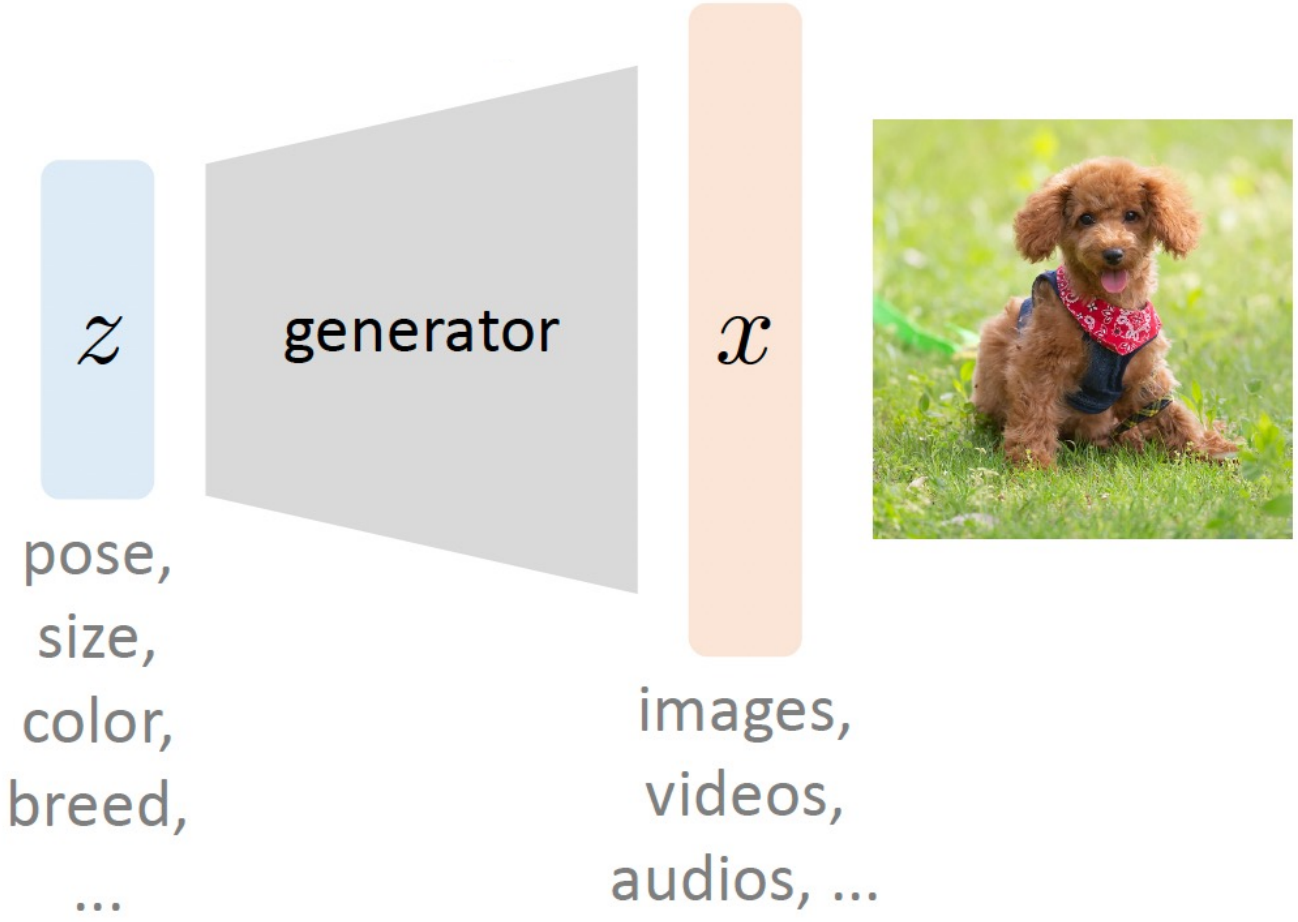
- Student Presentation Schedule
  - https://hkuhk-my.sharepoint.com/:x:/g/personal/bdai_hku_hk/IQCw7NjfThocQZkrtKkYWOexAUxFssmjnC6OhlxUvSBITEo?e=NbkAHd

- Group Info
  - https://hkuhk-my.sharepoint.com/:x:/g/personal/bdai_hku_hk/IQCCN_t6WAxwR4MGYn4QK_LRAeFud4miJGwrJNAYJWO7kU0?e=FVeeVu

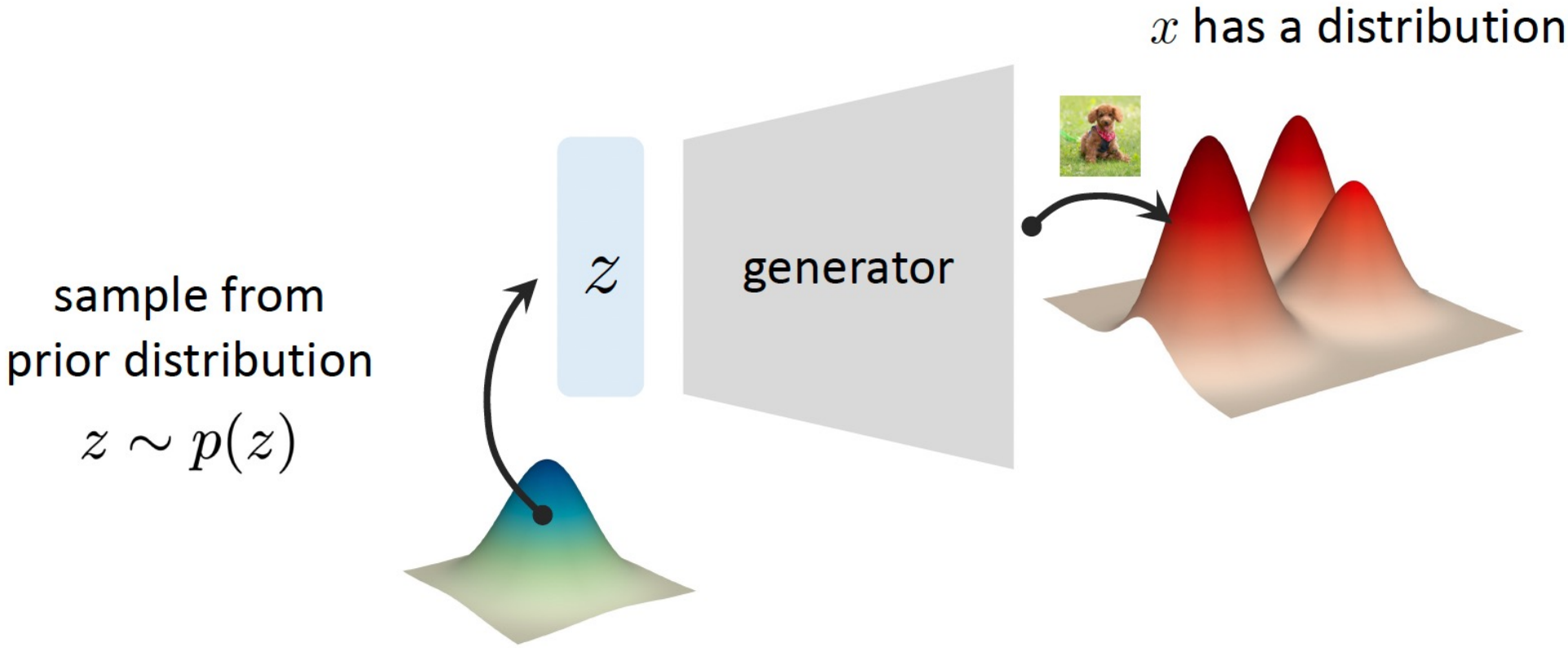
- Final Group Presentation Schedule
  - https://hkuhk-my.sharepoint.com/:x:/g/personal/bdai_hku_hk/IQBpwEA-lzQUSorbZL4MNCi-Ae06e_B-tWlQsj2JxTGxgHo?e=3ntO3G

- Slides reference
  - MIT EECS Fall 2024, 6.S978 Deep Generative Models, https://mit-6s978.github.io/
  - Stanford Fall 2023, CS236 Deep Generative Models, https://deepgenerativemodels.github.io/

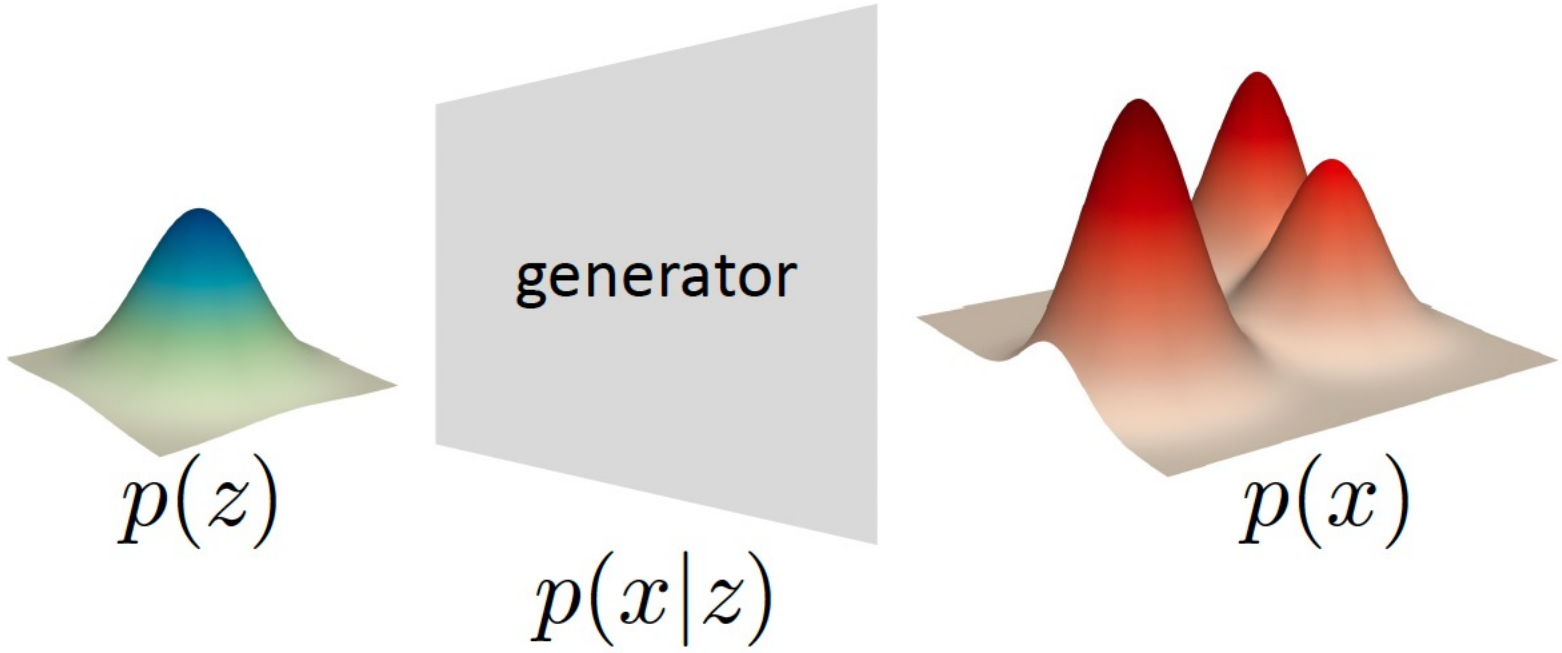
Assuming a data generation process:



Assuming a data generation process:

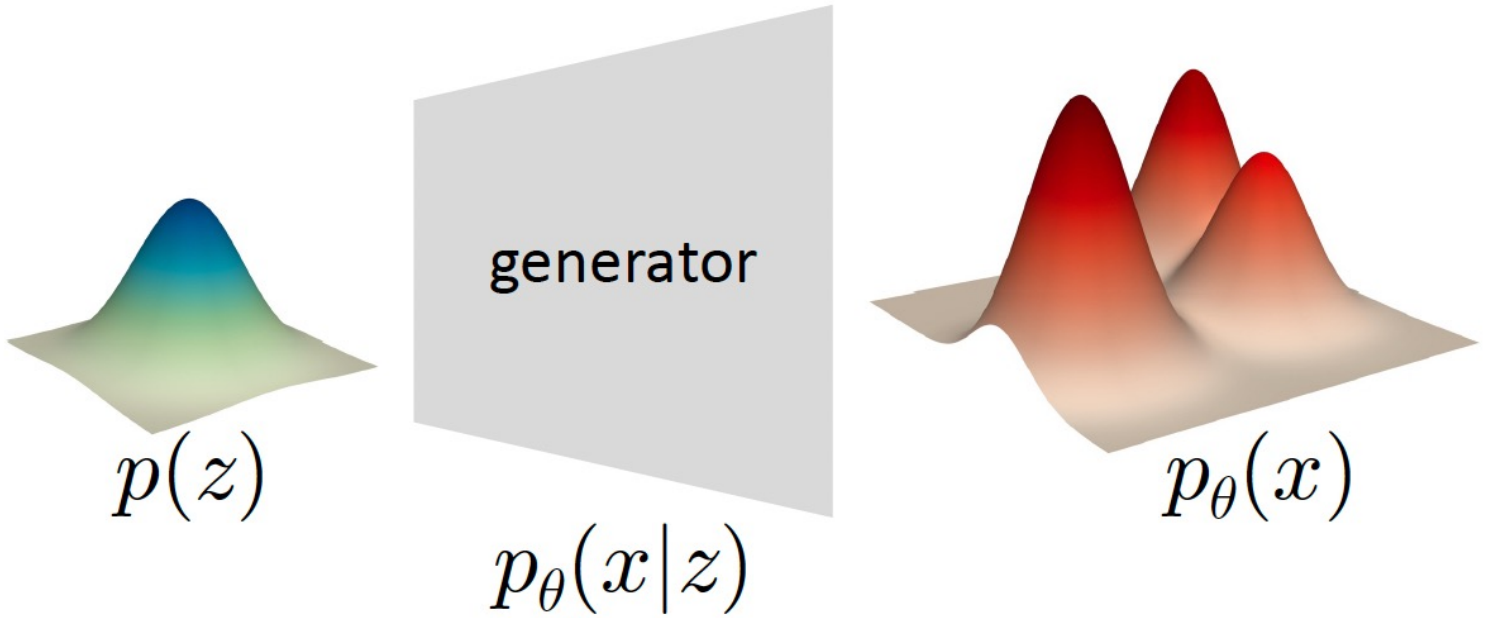


Assuming a data generation process:



Assuming a data generation process:

- Represent a distribution $p_\theta(x|z)$ by a neural network
  - Learnable parameters: θ



Measuring how good our learned distribution is

- Recall our setting:
  - Given a training set of examples, e.g., images of dogs, following a underlying distribution p_{data}.
  - Learn a probability distribution $p(x)$ over samples x such that
    - Generation: if we sample $x_{new} \sim p(x)$, x_{new} should look like a dog
    - Density estimation: $p(x)$ should be high if x looks like a dog, and low otherwise

Measuring how good our learned distribution is

- Recall our setting:
  - Given a training set of examples, e.g., images of dogs, following a underlying distribution p_{data}.
  - Learn a probability distribution $p(x)$ over samples x such that
    - Generation: if we sample $x_{new} \sim p(x)$, x_{new} should look like a dog
    - Density estimation: $p(x)$ should be high if x looks like a dog, and low otherwise
  - The standard assumption is that the data samples are independent and identically distributed (IID)

Measuring how good our learned distribution is

- Recall our setting:
  - Given a training set of examples, e.g., images of dogs, following a underlying distribution p_{data}.
  - Learn a probability distribution p(x) over samples x such that
    - Generation: if we sample $x_{new} \sim p(x)$, x_{new} should look like a dog
    - Density estimation: p(x) should be high if x looks like a dog, and low otherwise
  - The standard assumption is that the data samples are independent and identically distributed (IID)
- This is in general hard because of
  - limited data only provides a rough approximation
  - Example: *Suppose we represent each image with a vector X of 784 binary variables (black vs. white pixel). How many possible states (= possible images) in the model?* $2^{784} \approx 10^{236}$. *Even* 10^7 *training examples provide extremely sparse coverage!*
- Select the "best" approximation p(x)

Measuring how good our learned distribution is

- KL-divergence is often used to measure distance between two distributions p and q.

$$D(p\|q) = \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}$$

  - $D(p \,\|\, q) \geq 0$ for all p, q, with equality if and only if p = q.
  - KL-divergence is asymmetric, $D(p\|q) \neq D(q\|p)$

Measuring how good our learned distribution is

- KL-divergence is often used to measure distance between two distributions p and q.

$$D(p\|q) = \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}$$

  - Intuitive explanation: if your data comes from p, but you use a scheme optimized for q, the divergence is the number of extra bits needed on average

Measuring how good our learned distribution is

- KL-divergence is often used to measure distance between two distributions p and q.

$$D(p\|q) = \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}$$

- This can be simplified

$$\begin{aligned} \mathbf{D}(P_{\text{data}}||P_\theta) &= \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log\left(\frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}\right)\right] \\ &= \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_{\text{data}}(\mathbf{x})\right] - \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right] \end{aligned}$$

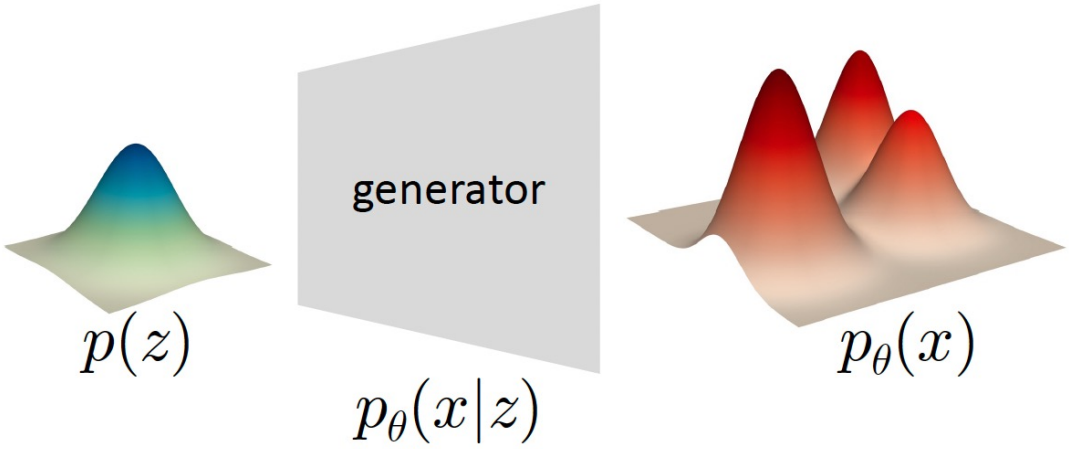
- Thus minimizing KL divergence is equivalent to maximizing the expected log-likelihood

$$\arg\min_{P_\theta} \mathbf{D}(P_{\text{data}}||P_\theta) = \arg\min_{P_\theta} -\mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right] = \arg\max_{P_\theta} \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}}\left[\log P_\theta(\mathbf{x})\right]$$

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

- In our case

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

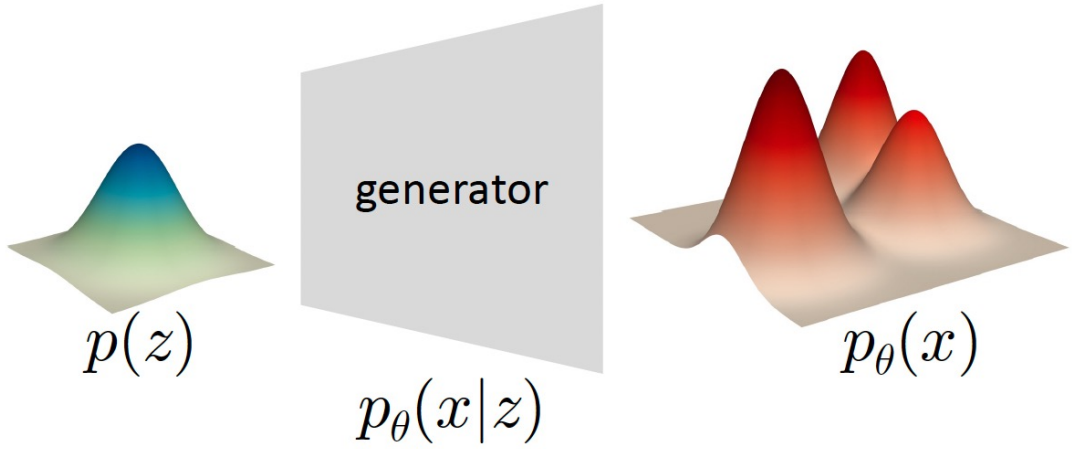


Maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$

- In our case

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

- Two sets of unknowns
  - We need to optimize for θ
  - We cannot control "true" $p(z)$



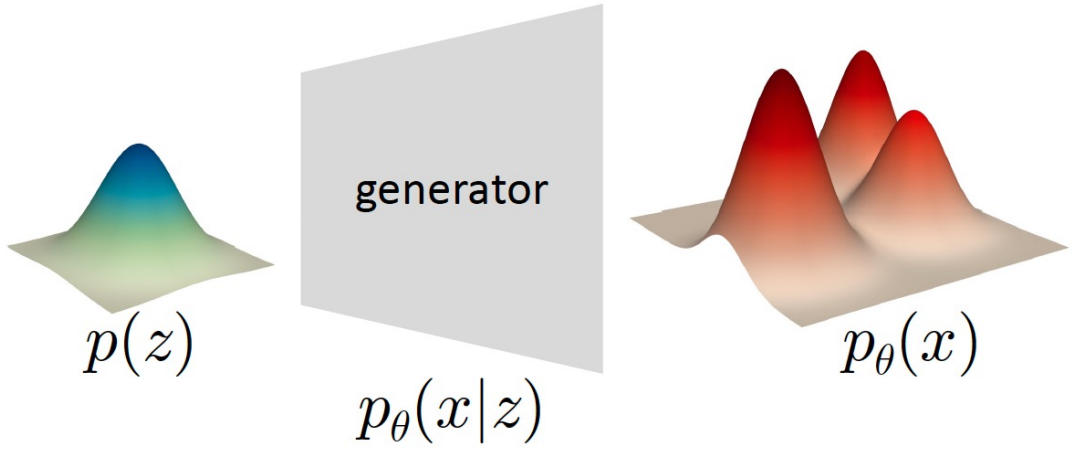
Maximize $\mathbb{E}_{x \sim p_{data}} \log p_\theta(x)$

- In our case

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

- Two sets of unknowns
  - We need to optimize for θ
  - We cannot control "true" $p(z)$

- Solution: introduce a "controllable" distribution $q(z)$



Rewrite log likelihood by latent z

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

- Rewrite log likelihood by latent z
- For any distribution q(z)

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

- Rewrite log likelihood by latent z
- For any distribution q(z)
- Bayes' rule

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

- Rewrite log likelihood by latent z
- For any distribution q(z)
- Bayes' rule

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

- Rewrite log likelihood by latent z

intractable

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

tractable tractable intractable

- Rewrite log likelihood by latent z

intractable

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

tractable tractable intractable

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

tractable tractable

- Rewrite log likelihood by latent z
- This is called Evidence Lower Bound (ELBO), holds for any distribution q(z)

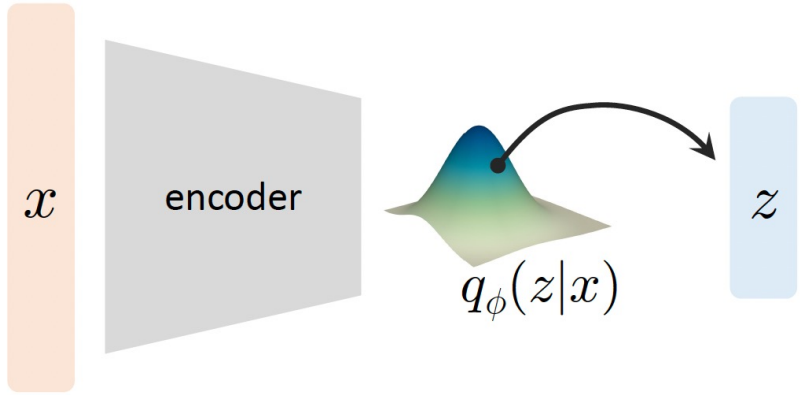
intractable

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

tractable tractable

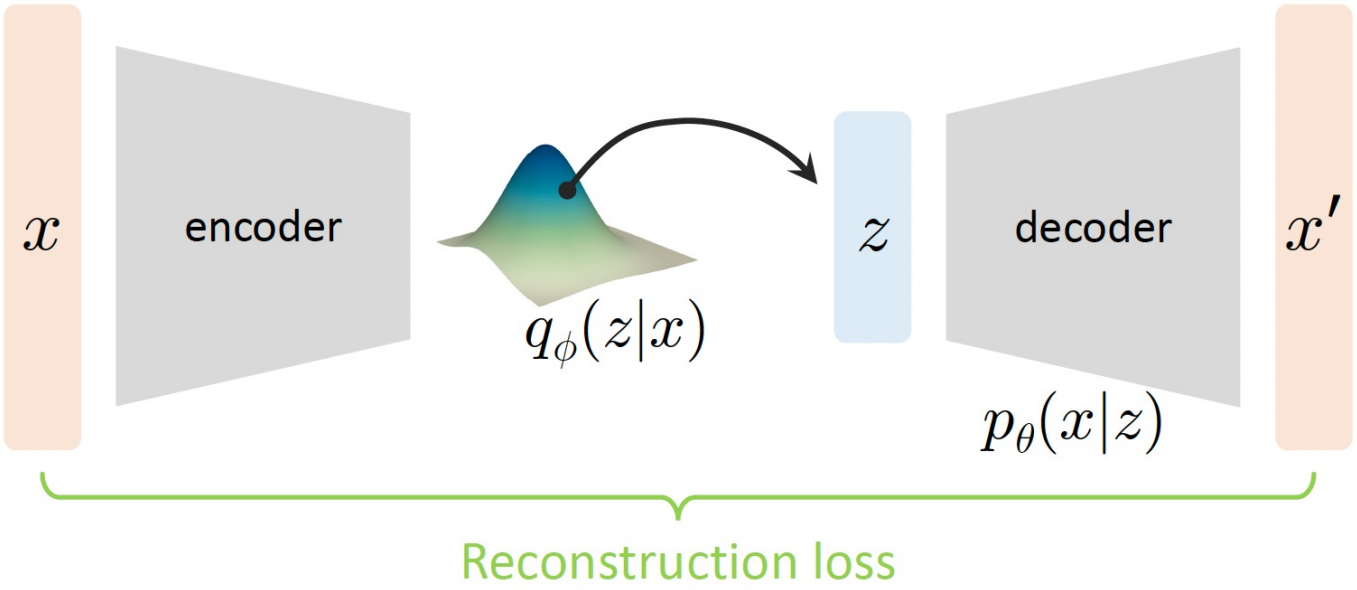
- Variational Autoencoder
  - Parameterize q(z)
  - Let $p_\theta(z)$ be a simple known prior $p(z)$

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



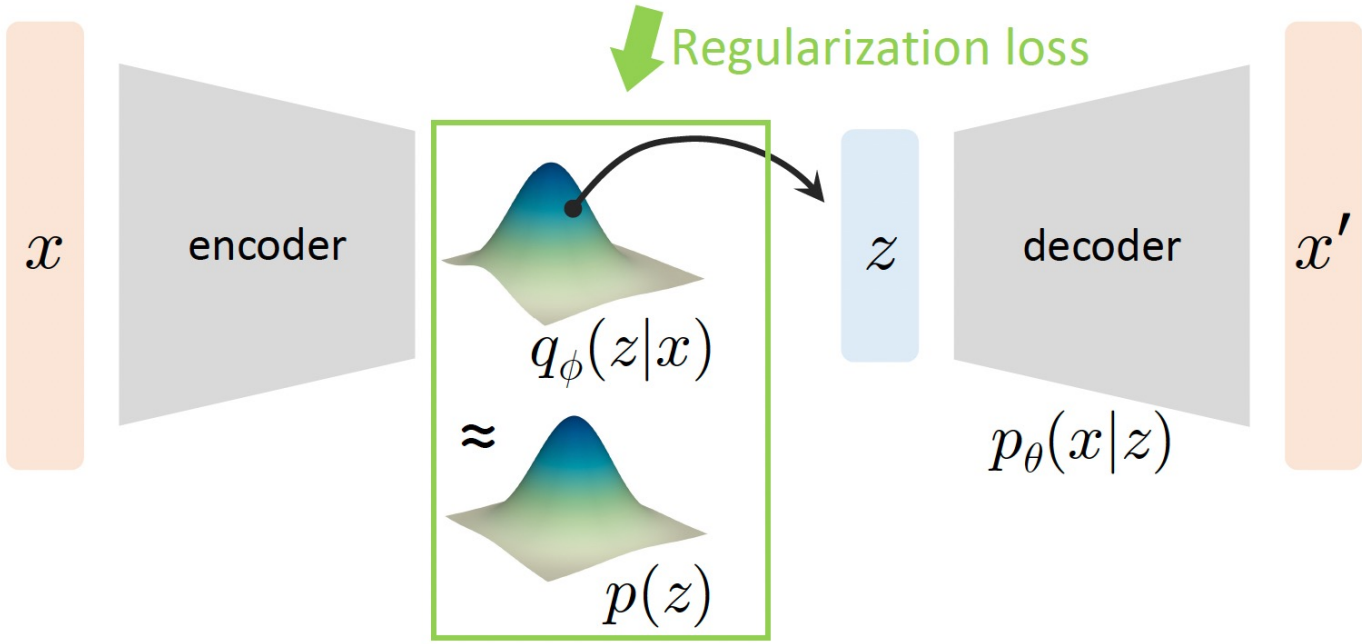
- Variational Autoencoder
  - Minimize

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



- Variational Autoencoder
  - Minimize

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



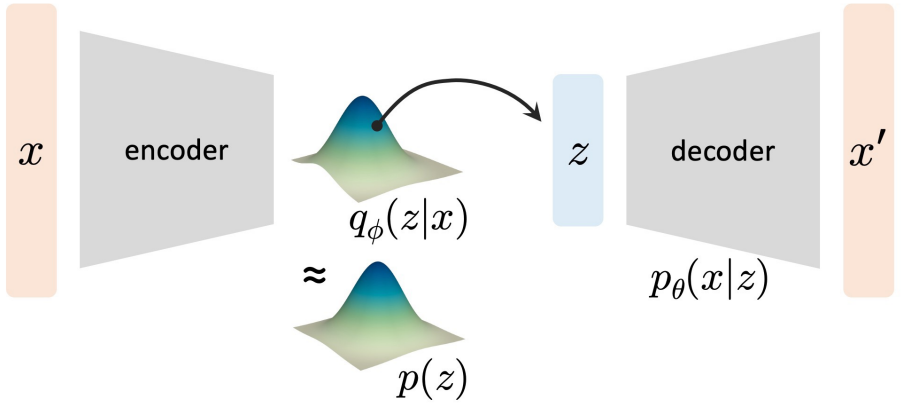
- Variational Autoencoder
  - Minimize

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

Example: L2 loss

- one-step Monte Carlo: $z \sim q_\phi(z|x)$
- map z by decoder net: $g_\theta(z) \to x'$ 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



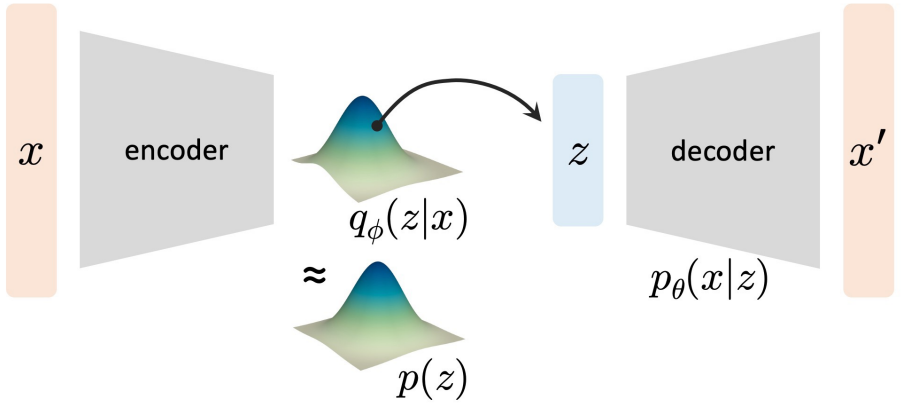
- Variational Autoencoder
  - Minimize

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

Example: Gaussian prior

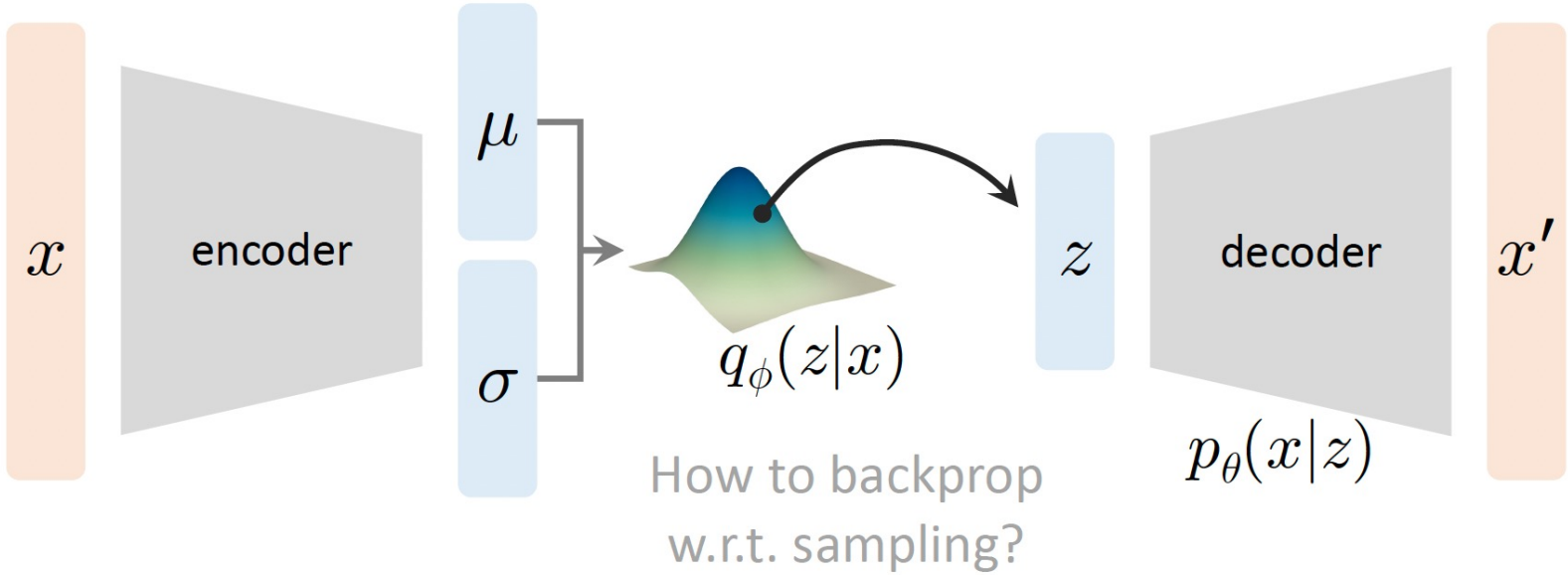
- let $p(z) = \mathcal{N}(z \mid 0, \mathbf{I})$
- model $q_\phi(z|x)$ by Gaussian: $\mathcal{N}(z \mid \mu, \sigma)$
- map x by encoder net: $f_\phi(x) \to \mu, \sigma$ — again, network estimates distribution's parameters
- compute loss analytically: $\mathcal{D}_{\mathrm{KL}}\Big(\mathcal{N}(z \mid \mu, \sigma) \,||\, \mathcal{N}(z \mid 0, \mathbf{I})\Big)$

- fixed covariance ⇒ L2 loss on μ



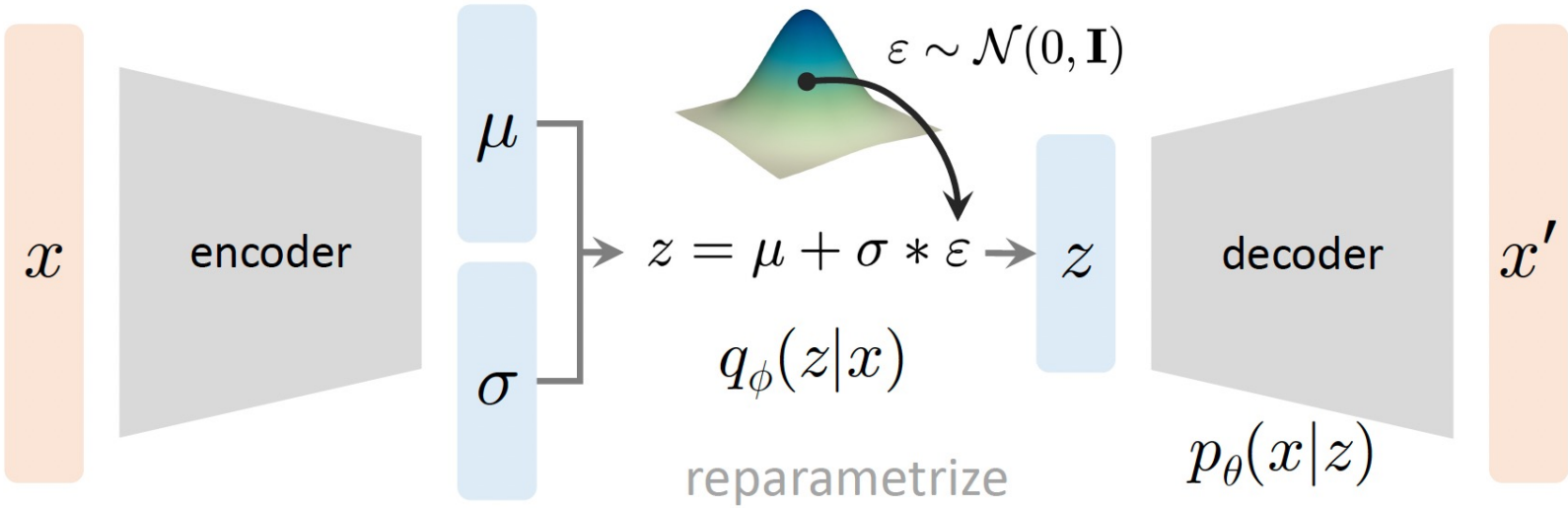
- Variational Autoencoder
  - Minimize

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



- Variational Autoencoder
  - Minimize

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

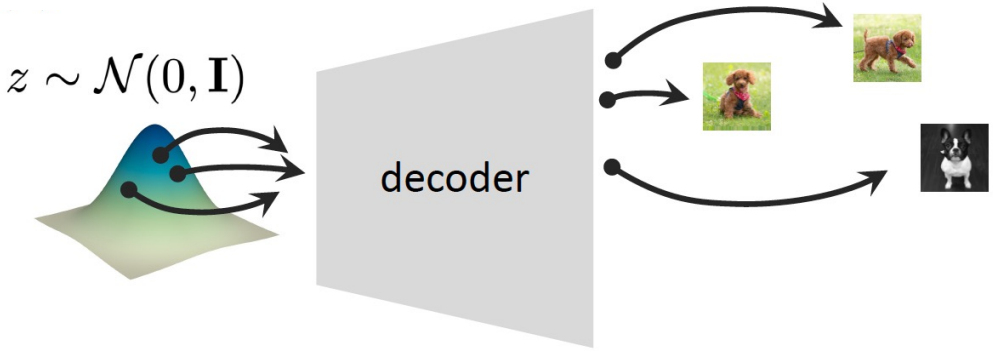


- Variational Autoencoder
  - Minimize across all samples

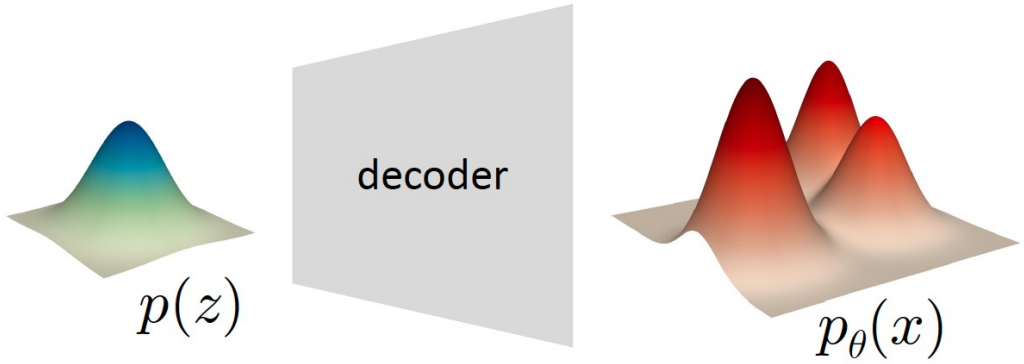
$$\mathcal{L}_{\theta,\phi}(x) = -\mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log p_\theta(x|z)\Big] + \mathcal{D}_{\mathrm{KL}}\Big(q_\phi(z|x)||p(z)\Big)$$

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

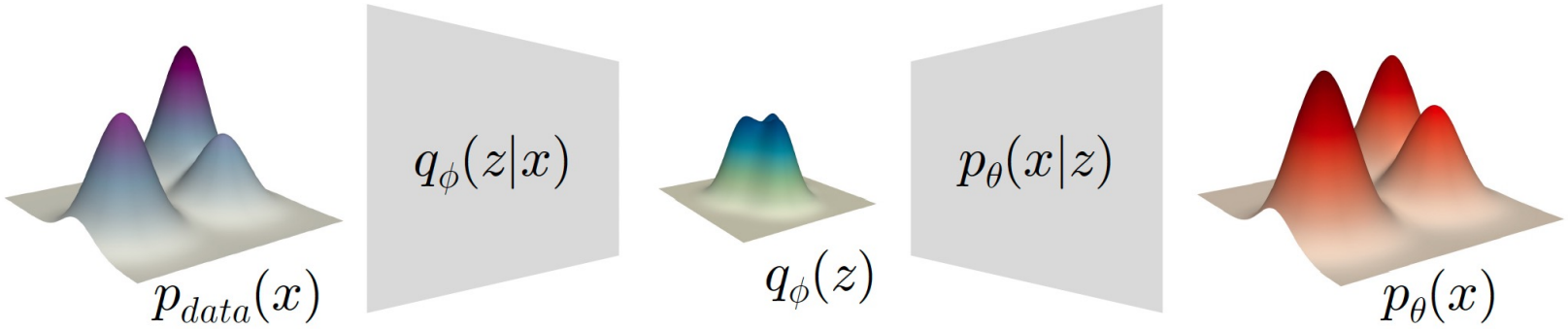
- Variational Autoencoder
  - Inference
    - Sample z, then map z by the decoder



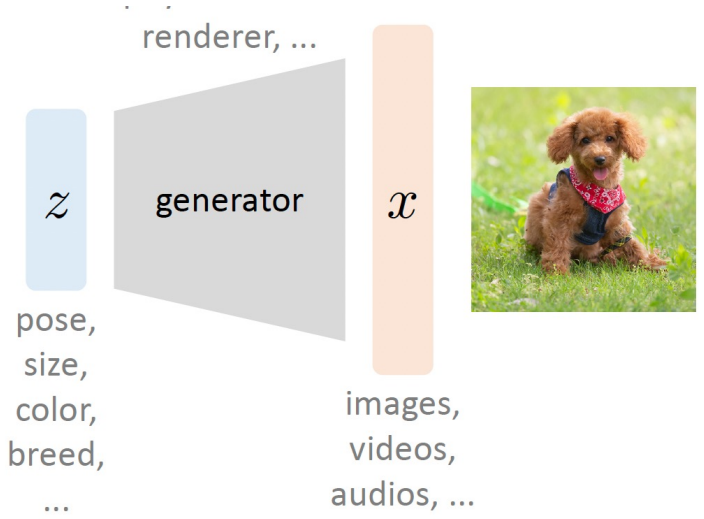
- Variational Autoencoder
  - Inference
    - Sample z, then map z by the decoder
    - Decoder is a deterministic mapping from one distribution to another



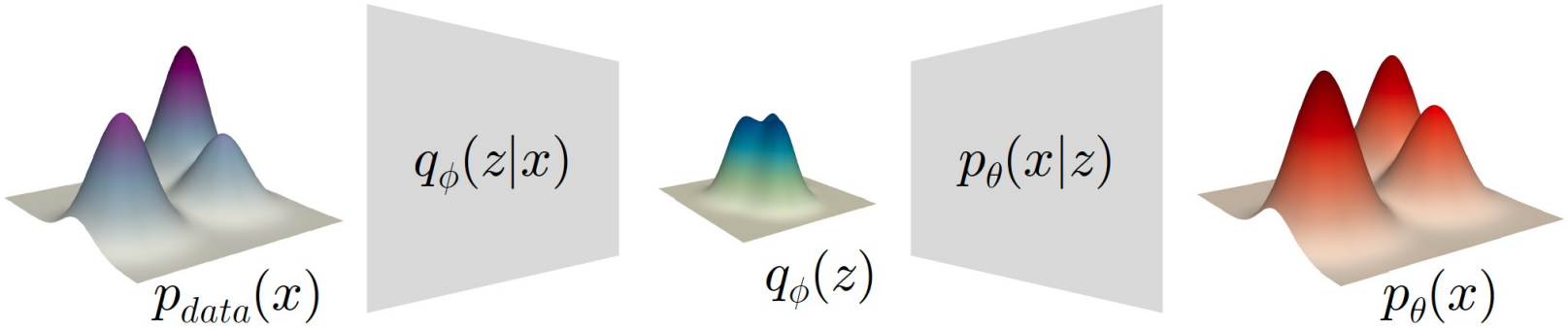
- Variational Autoencoder
  - Decoder maps latent distribution to data distribution
  - Encoder maps data distribution to latent distribution



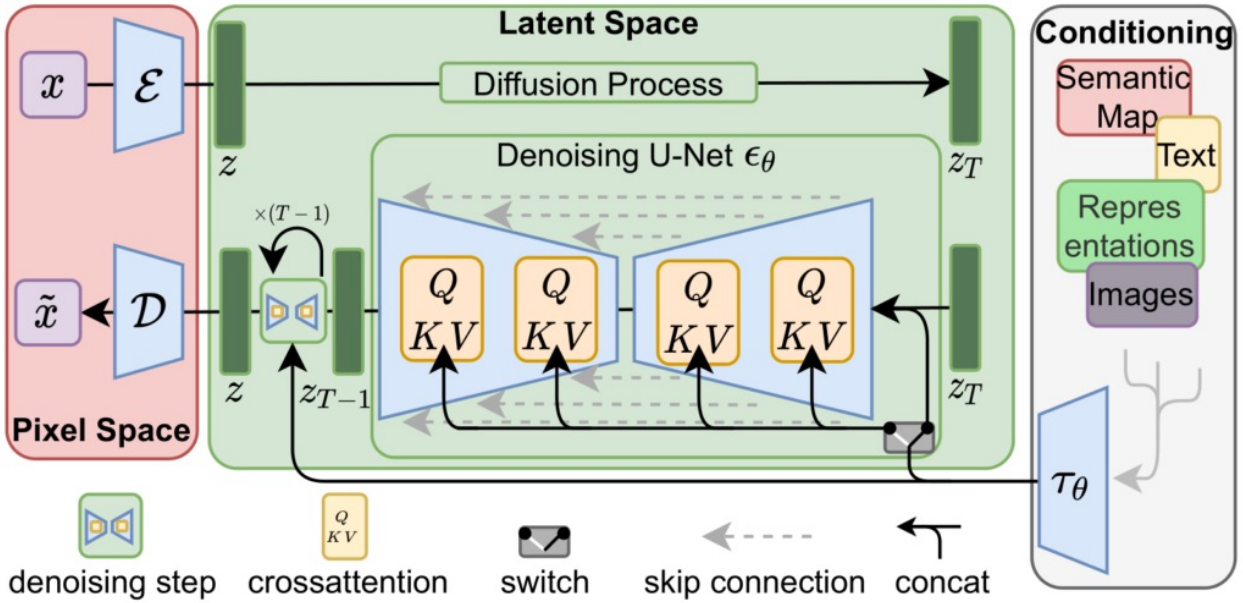
- Variational Autoencoder
  - We start this lecture by assuming x is governed by some latent variables
  - Question: we don't know whether this assumption holds in practice, and even it holds we don't have the ground-truth for these variables



- Variational Autoencoder
  - ~~We start this lecture by assuming x is governed by some latent variables~~
  - VAE is often used for complexity reduction, where dim(z) << dim(x)



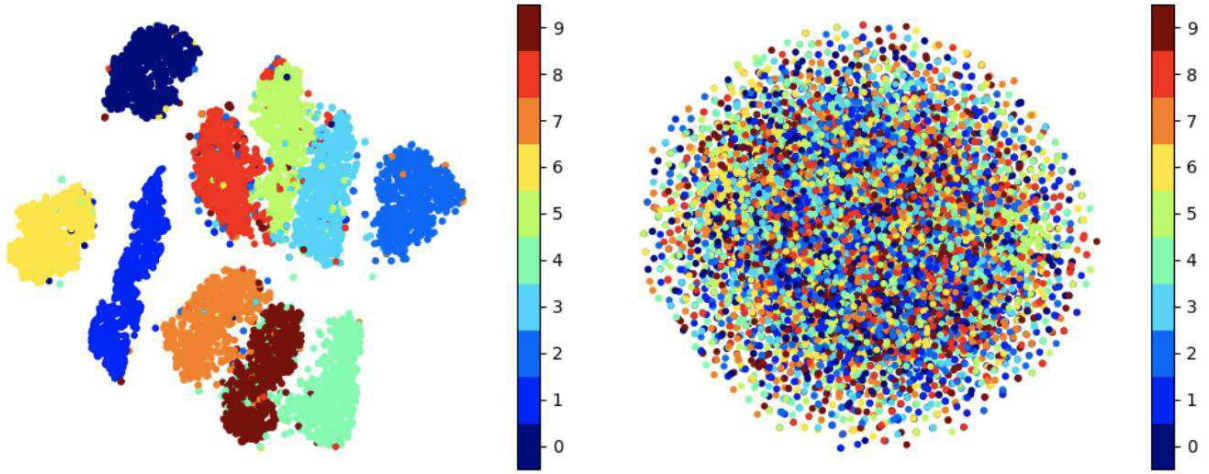
- Variational Autoencoder
  - ~~We start this lecture by assuming x is governed by some latent variables~~
  - VAE is often used for complexity reduction, where dim(z) << dim(x)
  - In this perspective, z captures the most important semantics of x, just like PCA



High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022

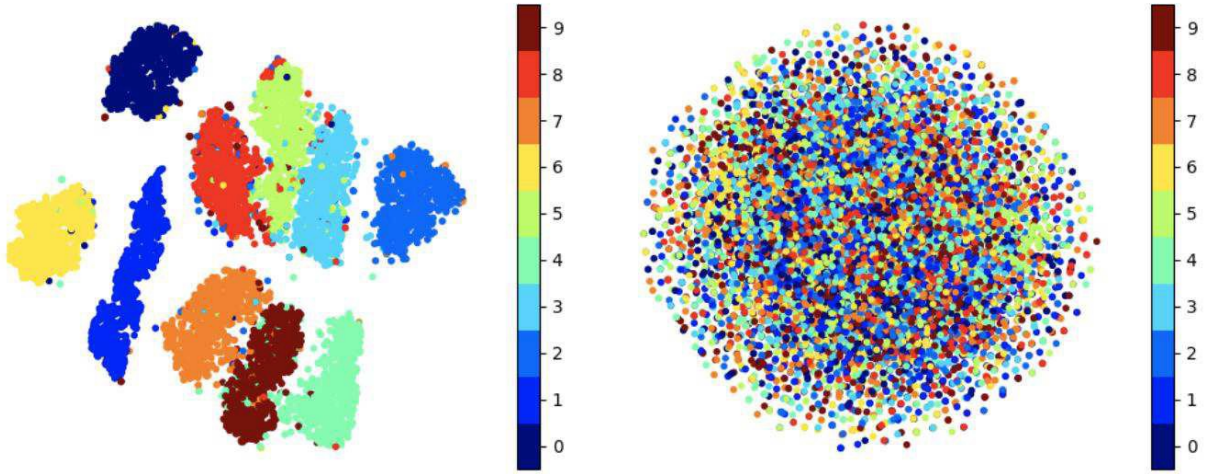
https://arxiv.org/abs/2112.10752

- Variational Autoencoder
  - Issues:
    - Posterior collapse: $\mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$ is very small, yet this is achieved by encoder completely ignoring x



https://openaccess.thecvf.com/content_ECCV_2018/papers/Ananya_Harsh_Jha_Disentangling_Factors_of_ECCV_2018_paper.pdf

- Variational Autoencoder
  - Issues:
    - Posterior collapse: $\mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$ is very small, yet this is achieved by encoder completely ignoring x
      - Encoder is too week; decoder is too strong; over-emphasizing the KL loss term; …

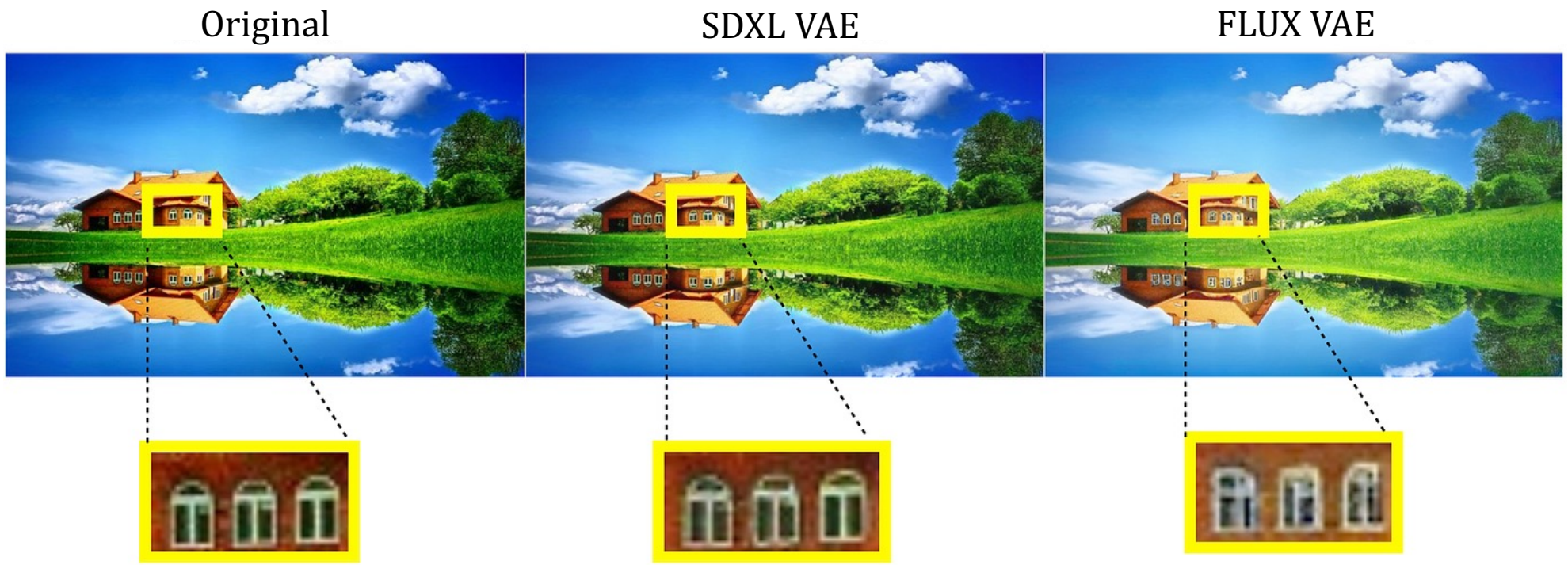


https://openaccess.thecvf.com/content_ECCV_2018/papers/Ananya_Harsh_Jha_Disentangling_Factors_of_ECCV_2018_paper.pdf

- Variational Autoencoder
  - Issues:
    - Posterior collapse: $\mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$ is very small, yet this is achieved by encoder completely ignoring x
    - Blurry outputs:

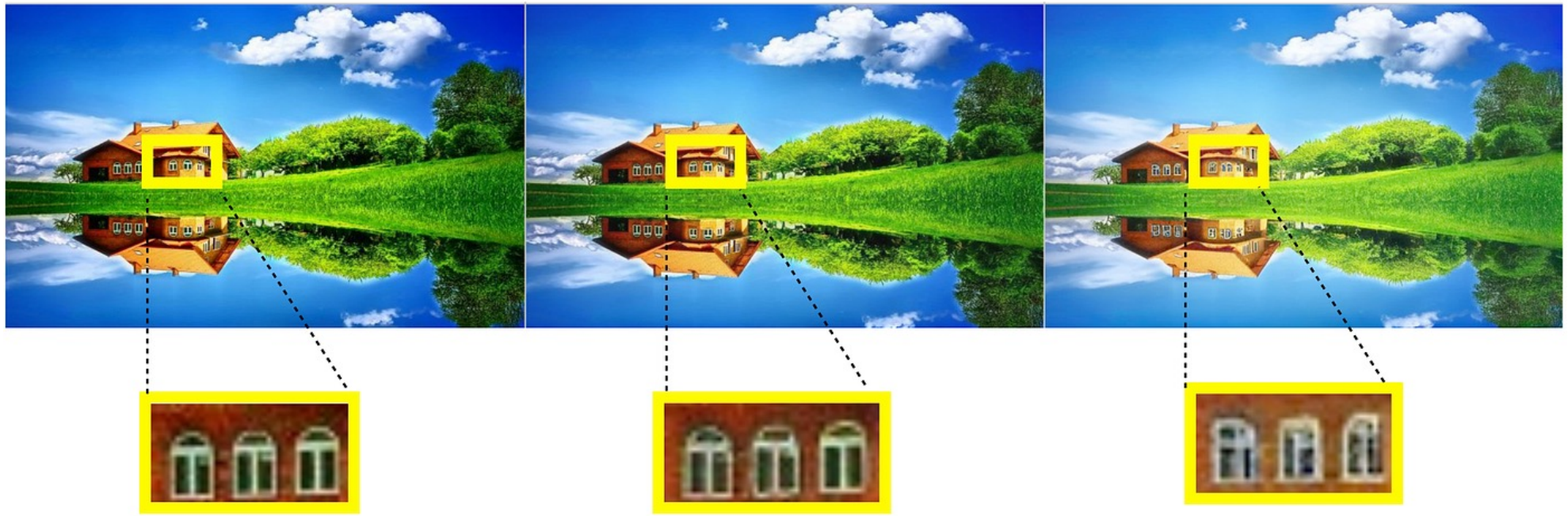
https://medium.com/@efrat_taig/vae-the-latent-bottleneck-why-image-generation-processes-lose-fine-details-a056dcd6015e

- Variational Autoencoder
  - Issues:
    - Posterior collapse: $\mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$ is very small, yet this is achieved by encoder completely ignoring x
    - Blurry outputs:



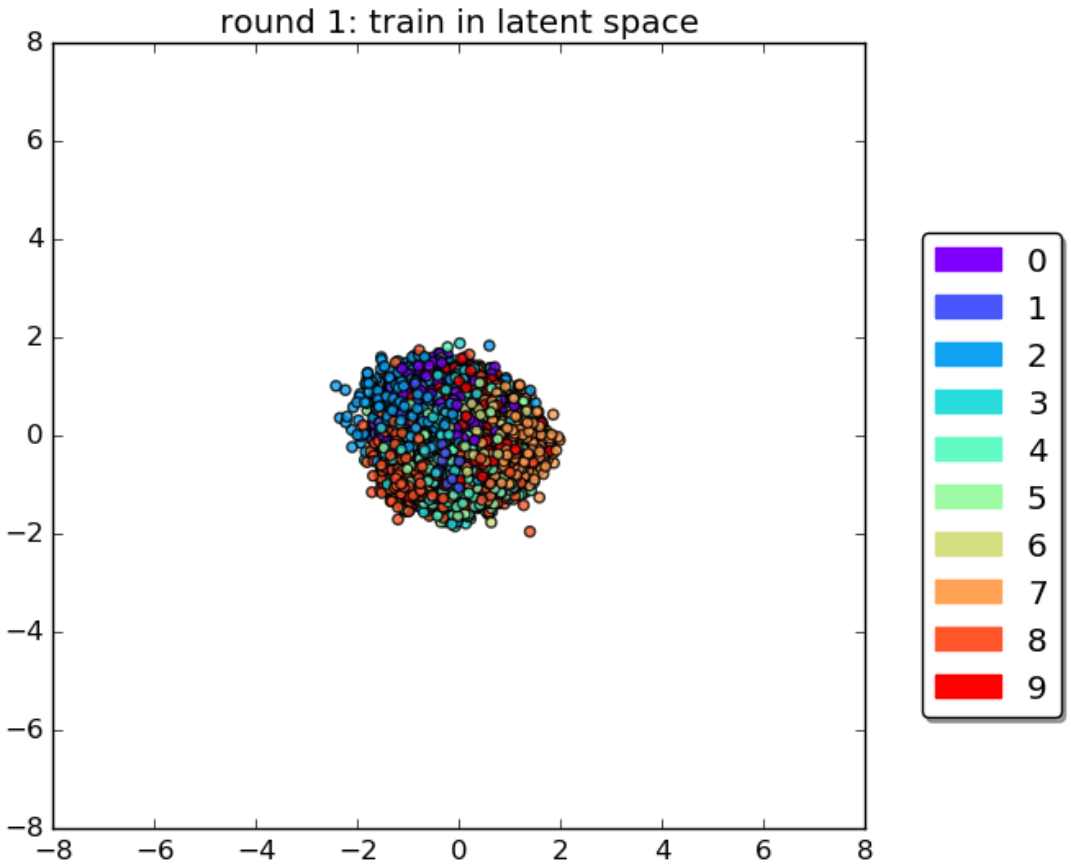
https://medium.com/@efrat_taig/vae-the-latent-bottleneck-why-image-generation-processes-lose-fine-details-a056dcd6015e

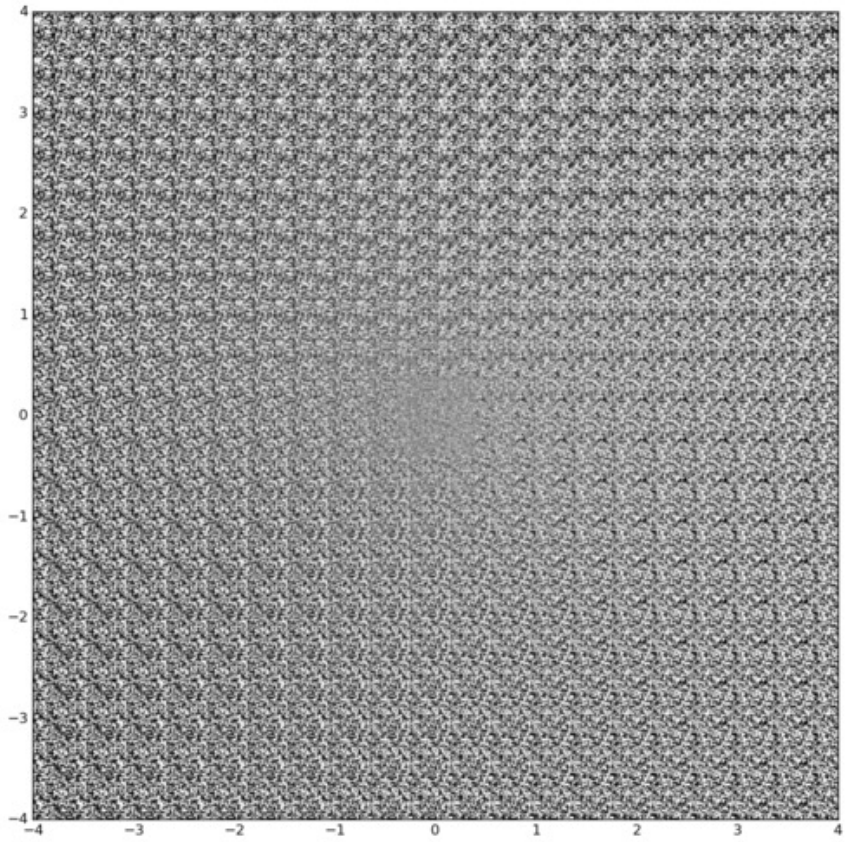
- Variational Autoencoder
  - Issues:
    - Posterior collapse: $\mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$ is very small, yet this is achieved by encoder completely ignoring x
    - Blurry outputs:
      - Compression ratio is too high; issue of the training objective; …



https://medium.com/@efrat_taig/vae-the-latent-bottleneck-why-image-generation-processes-lose-fine-details-a056dcd6015e

- Structure of the latent space



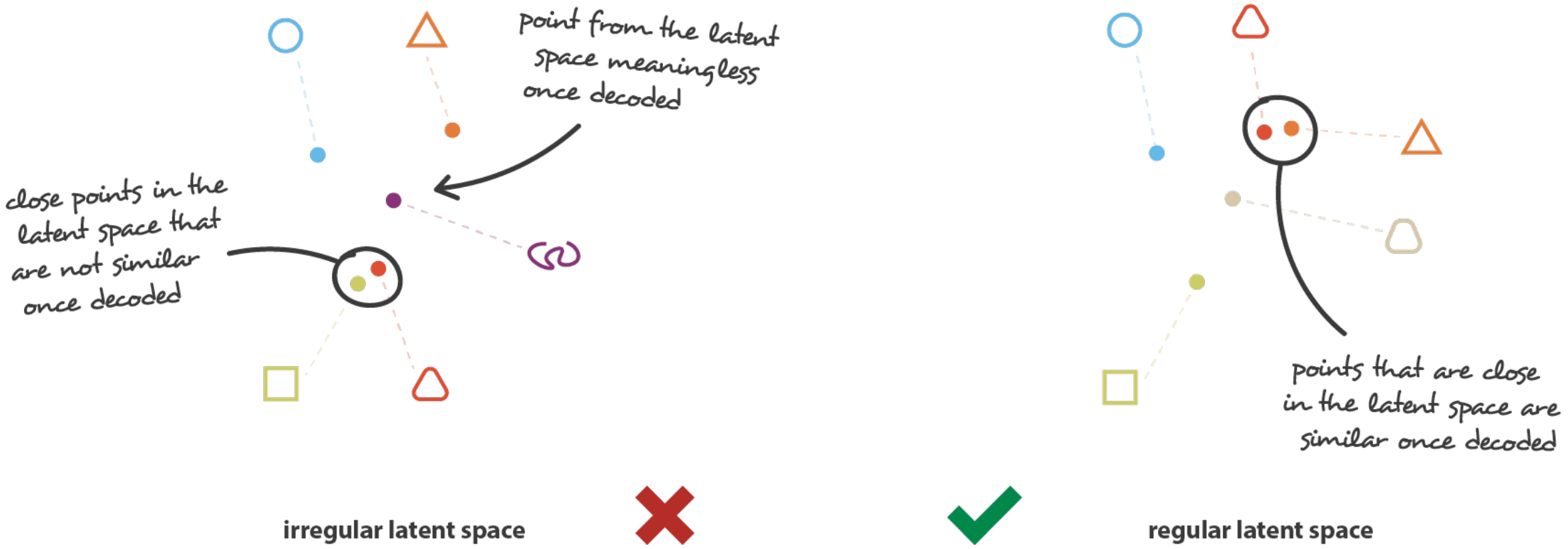


https://blog.fastforwardlabs.com/2016/08/12/introducing-variational-autoencoders-in-prose-and-code.html

- Structure of the latent space

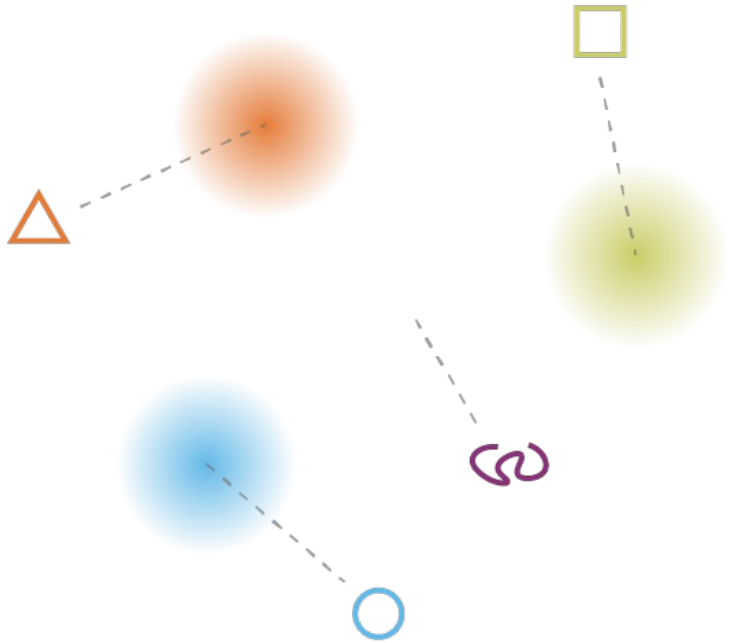
Auto-Encoding Variational Bayes, ICLR 2014

- Structure of the latent space

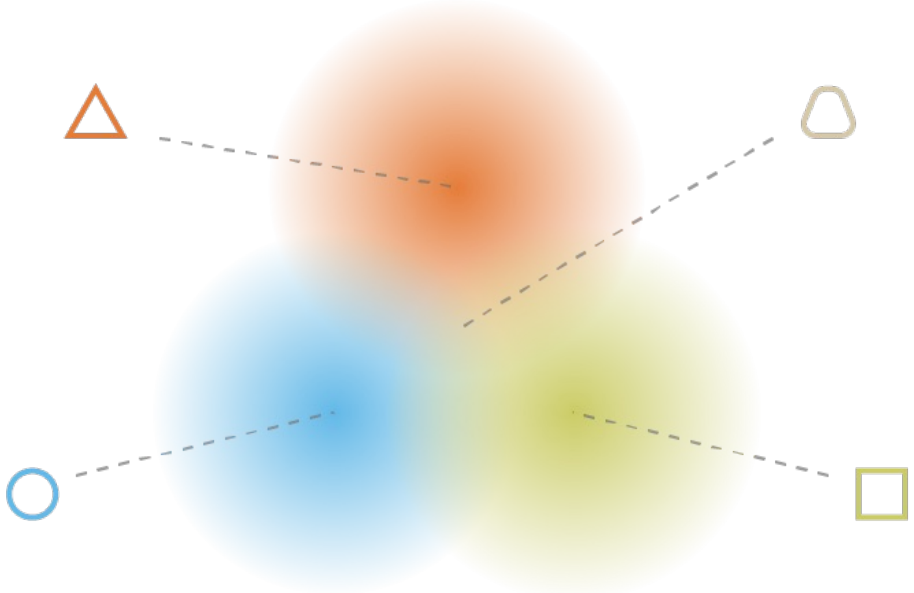


https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/

- Structure of the latent space



what can happen without regularisation



what we want to obtain with regularisation

https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/

- Structure of the latent space

https://tayden.github.io/VAE-Latent-Space-Explorer/

- Structure of the latent space
  - How to separate important subspaces?

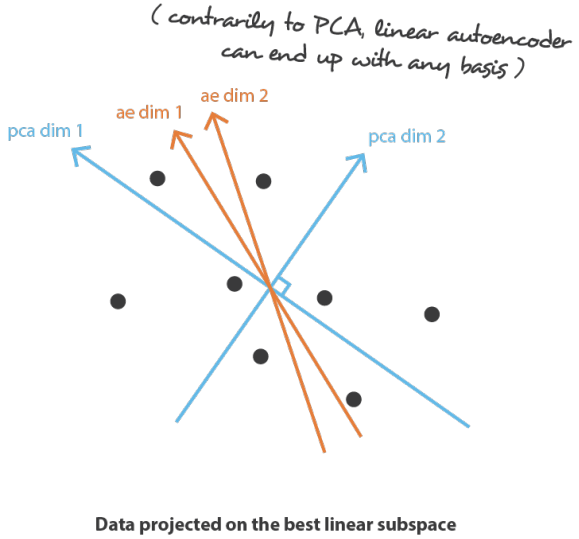
$$\max_{\phi,\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})]$$
$$\text{subject to } D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) < \delta$$

$$\begin{aligned}
\mathcal{F}(\theta,\phi,\beta) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta(D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) - \delta) \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) + \beta\delta \\
&\geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) \qquad \text{; Because } \beta,\delta\geq 0
\end{aligned}$$

$$L_{\mathrm{BETA}}(\phi,\beta) = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$

β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK, ICLR 2017

- Structure of the latent space
  - How to separate important subspaces?
  - When beta > 1, it applies a stronger constraint on the latent bottleneck and limits the representation capacity of z
  - Disentangled representation is more efficient

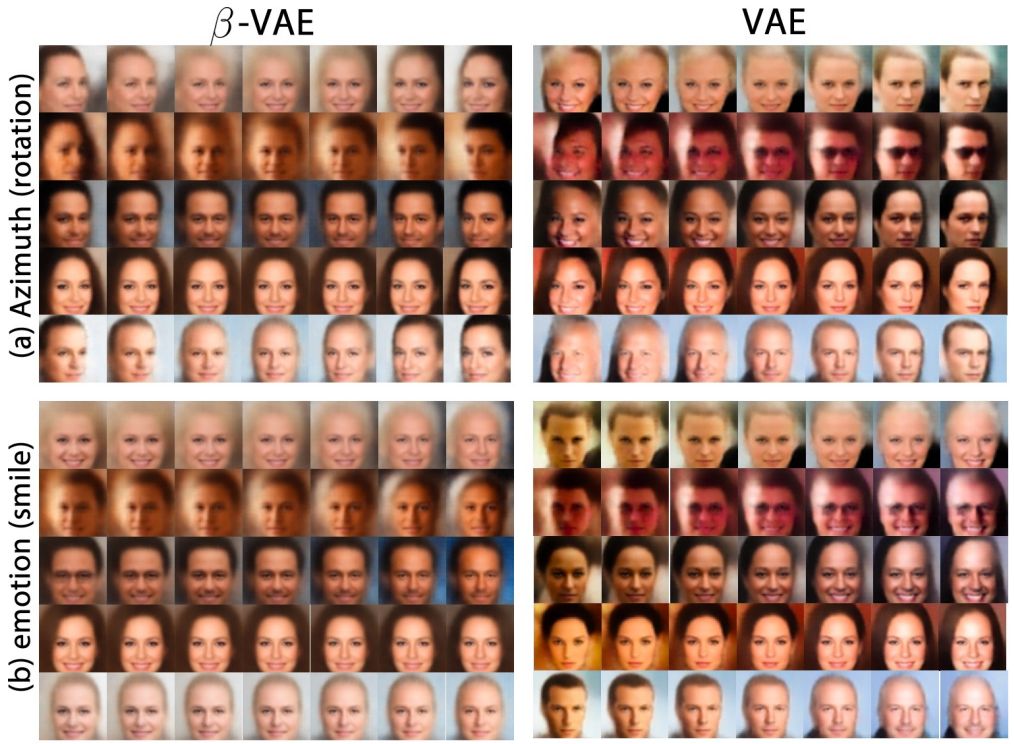


$$L_{\mathrm{BETA}}(\phi, \beta) = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}))$$

β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK, ICLR 2017

https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/

- Structure of the latent space
  - How to separate important subspaces?
  - When beta > 1, it applies a stronger constraint on the latent bottleneck and limits the representation capacity of z



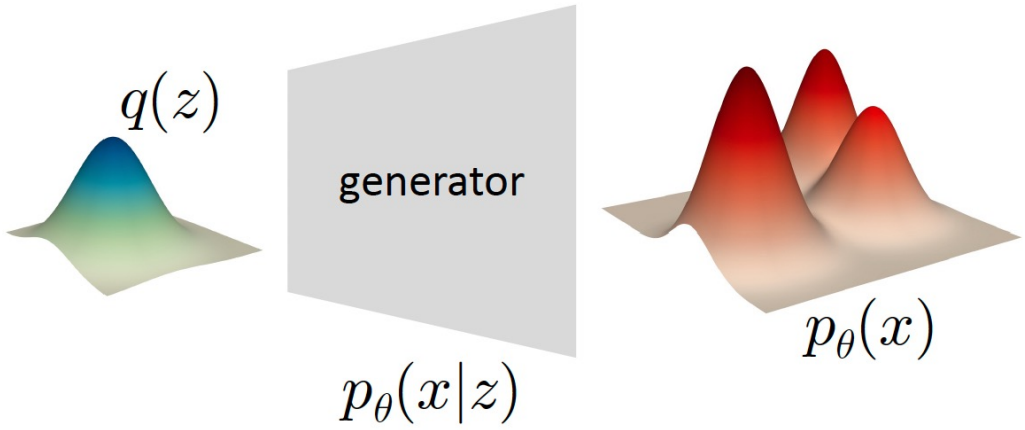
β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK, ICLR 2017

- Structure of the latent space
  - Relationship to Expectation-Maximization (EM)

  - Two sets of variables
    - q: distribution of latent
    - θ: parameters of generator

  - VAE:
    - Parameterize q by a neural network
    - Stochastic gradient decent

  - EM:
    - Often parameterize q analytically
    - Coordinate descent (alternative optimization)



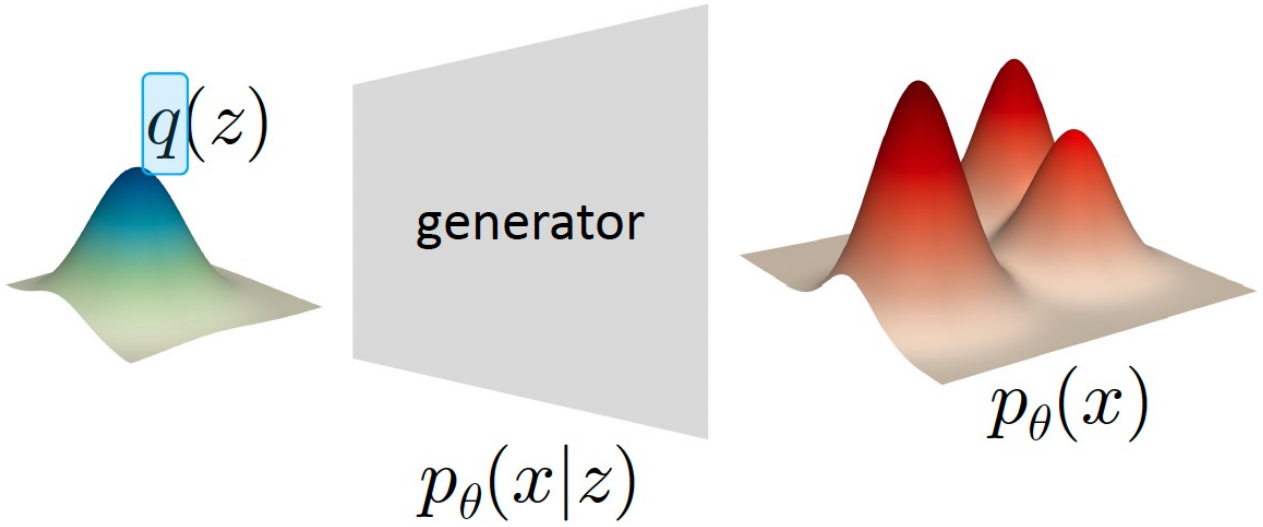
- Structure of the latent space
  - Relationship to Expectation-Maximization (EM)

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

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

E-step: optimize for q

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



- Structure of the latent space
  - Relationship to Expectation-Maximization (EM)

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

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

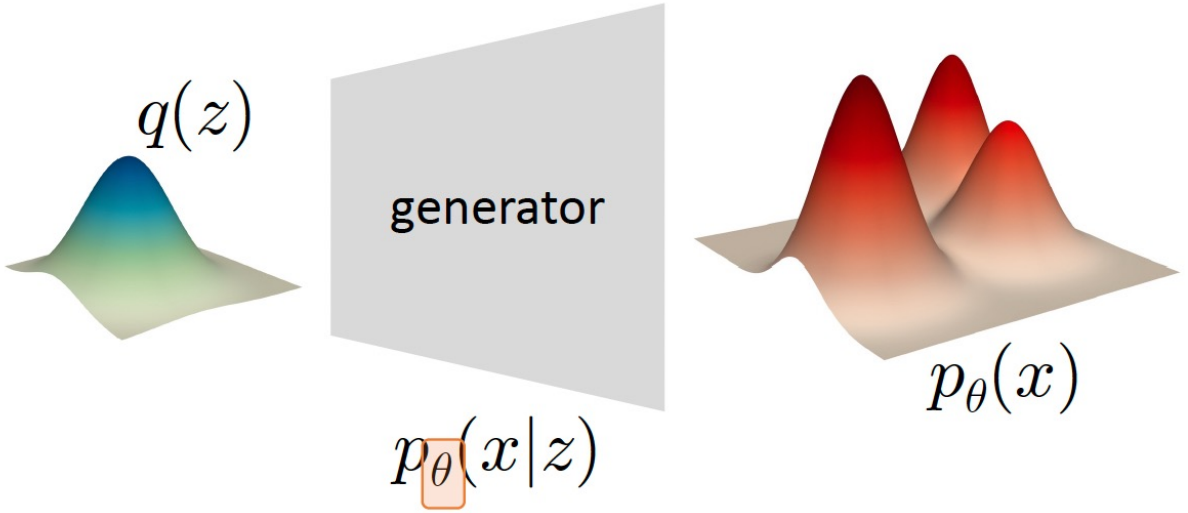
E-step: optimize for q

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

M-step: optimize for θ

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

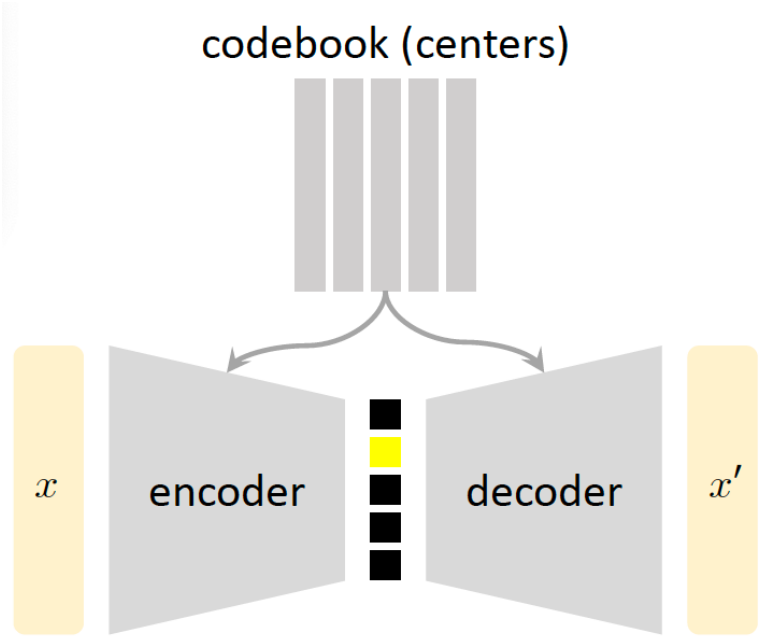
with sub-objective defined as: $Q(\theta|\theta^{(t)}) = \mathbb{E}_{p_{data}(x)}\mathbb{E}_{p_{\theta^{(t)}}(z|x)}[\log p_\theta(x,z)]$



- Structure of the latent space
  - An example with K-means

https://kkevsterrr.github.io/K-Means/

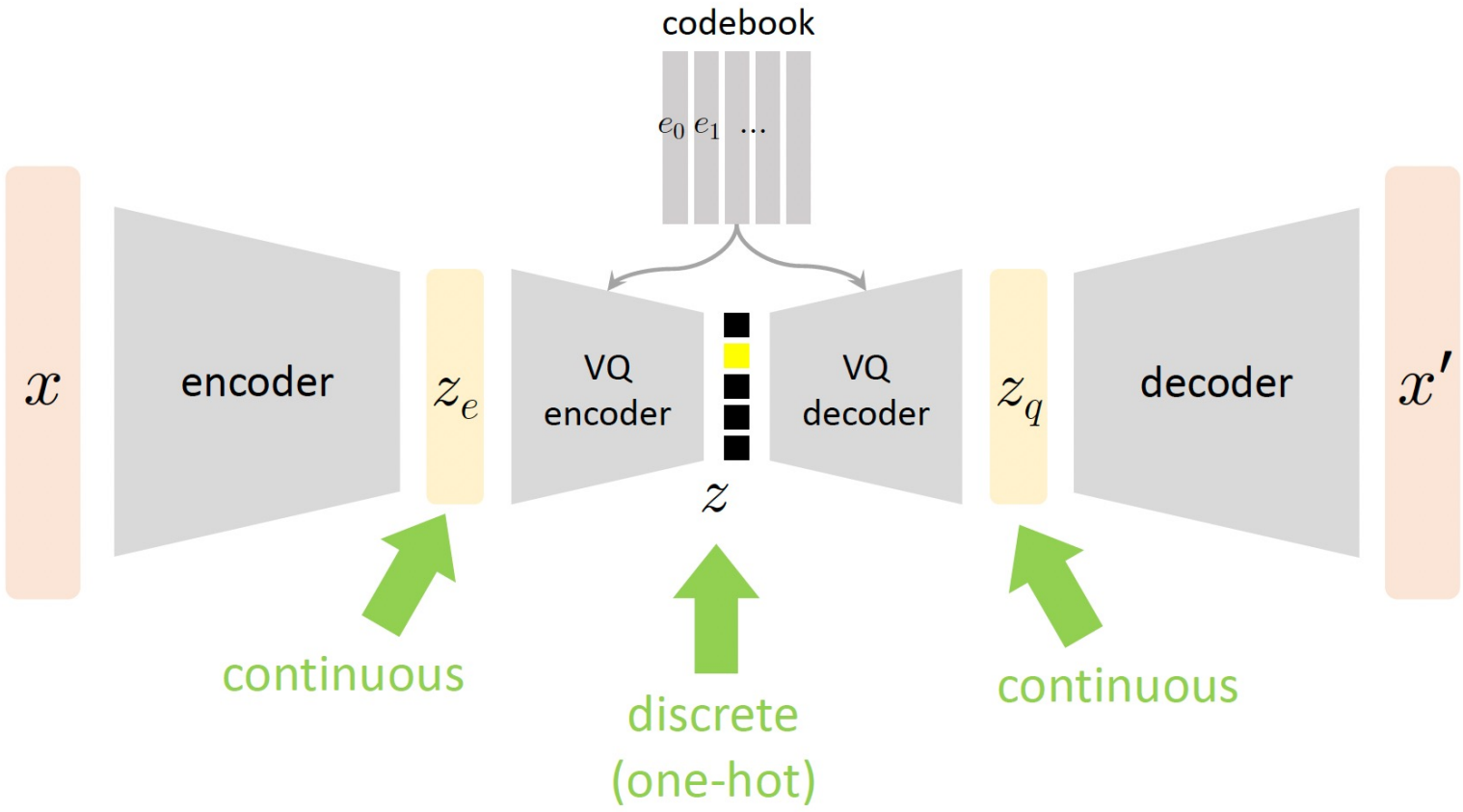
- Structure of the latent space
  - K-means as Autoencoder
    - Encoder: map x to one-hot
    - Decoder: map one-hot to x'
    - x' is a center



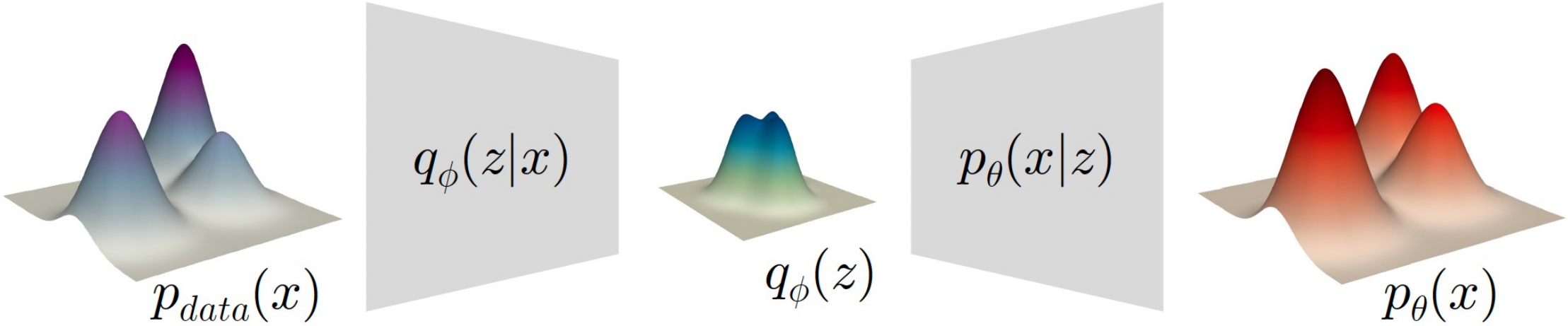
$$\min \sum_{x} ||x - Dec(Enc(x))||^2$$

- Structure of the latent space
  - K-means as Autoencoder
  - From autoencoder to variational autoencoder ?

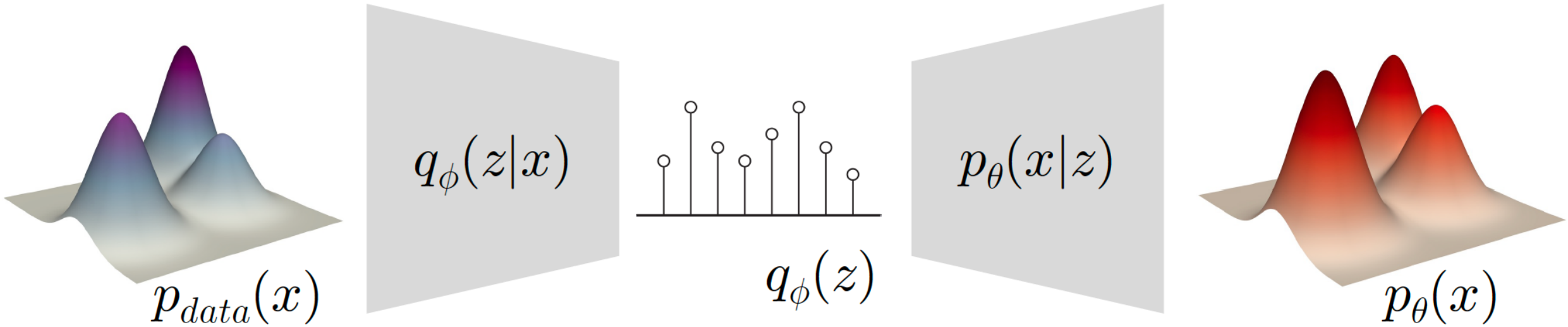
- Vector Quantized VAE (VQ-VAE)



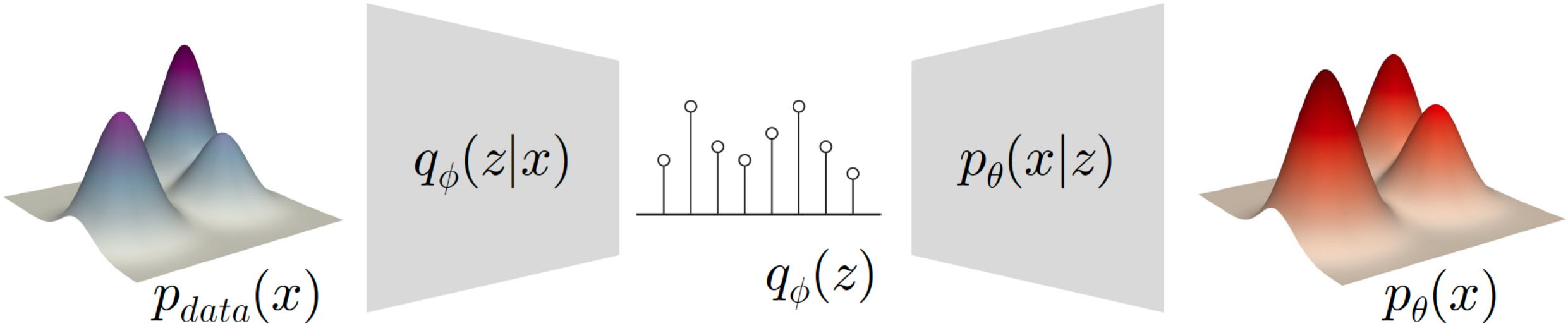
- Vector Quantized VAE (VQ-VAE)
  - Original VAE: latent variables are continuous



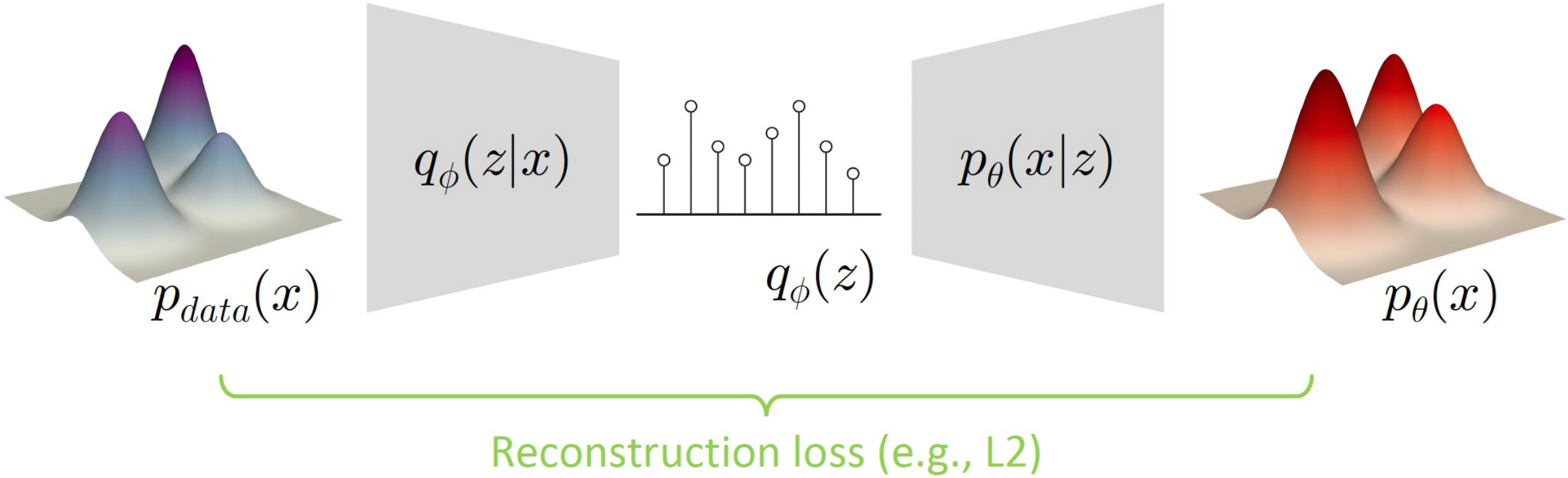
- Vector Quantized VAE (VQ-VAE)
  - Original VAE: latent variables are continuous
  - VQ-VAE: discrete latent variables
    - Categorical: no particular relation between numbers (SSN, zip code, ...)
    - Symbolic: language, speech, planning, ...



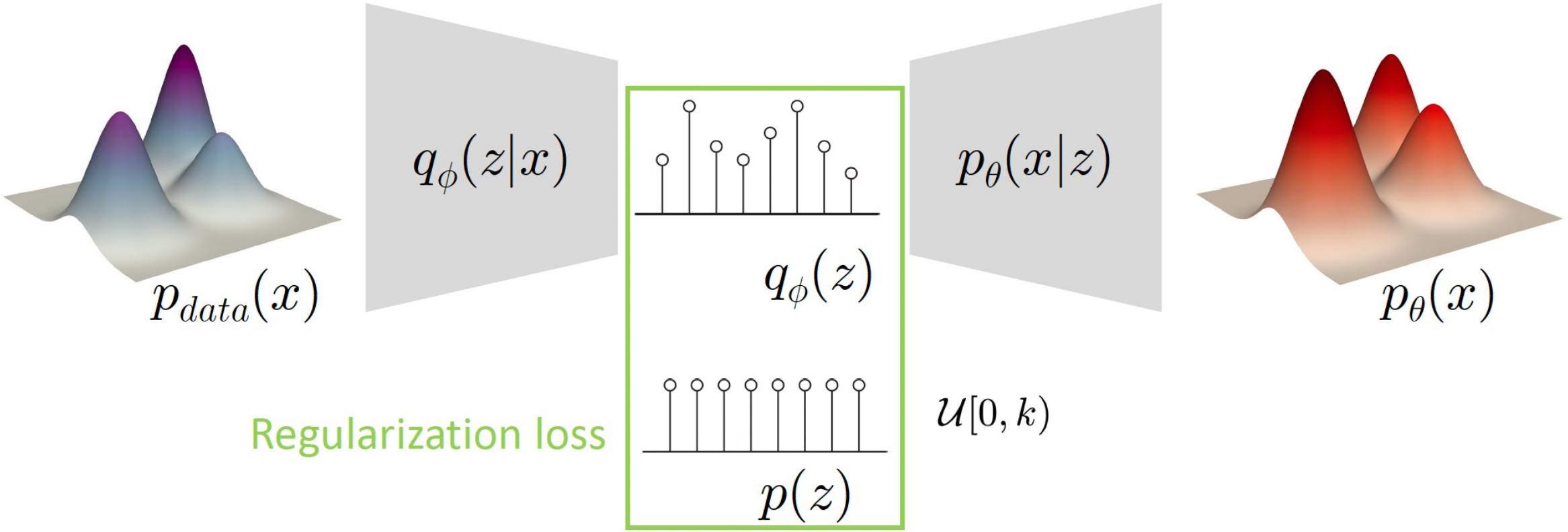
- Vector Quantized VAE (VQ-VAE)
  - Maximize ELBO
    - Reconstruction loss: about x
    - Regularization loss: about z (discrete)



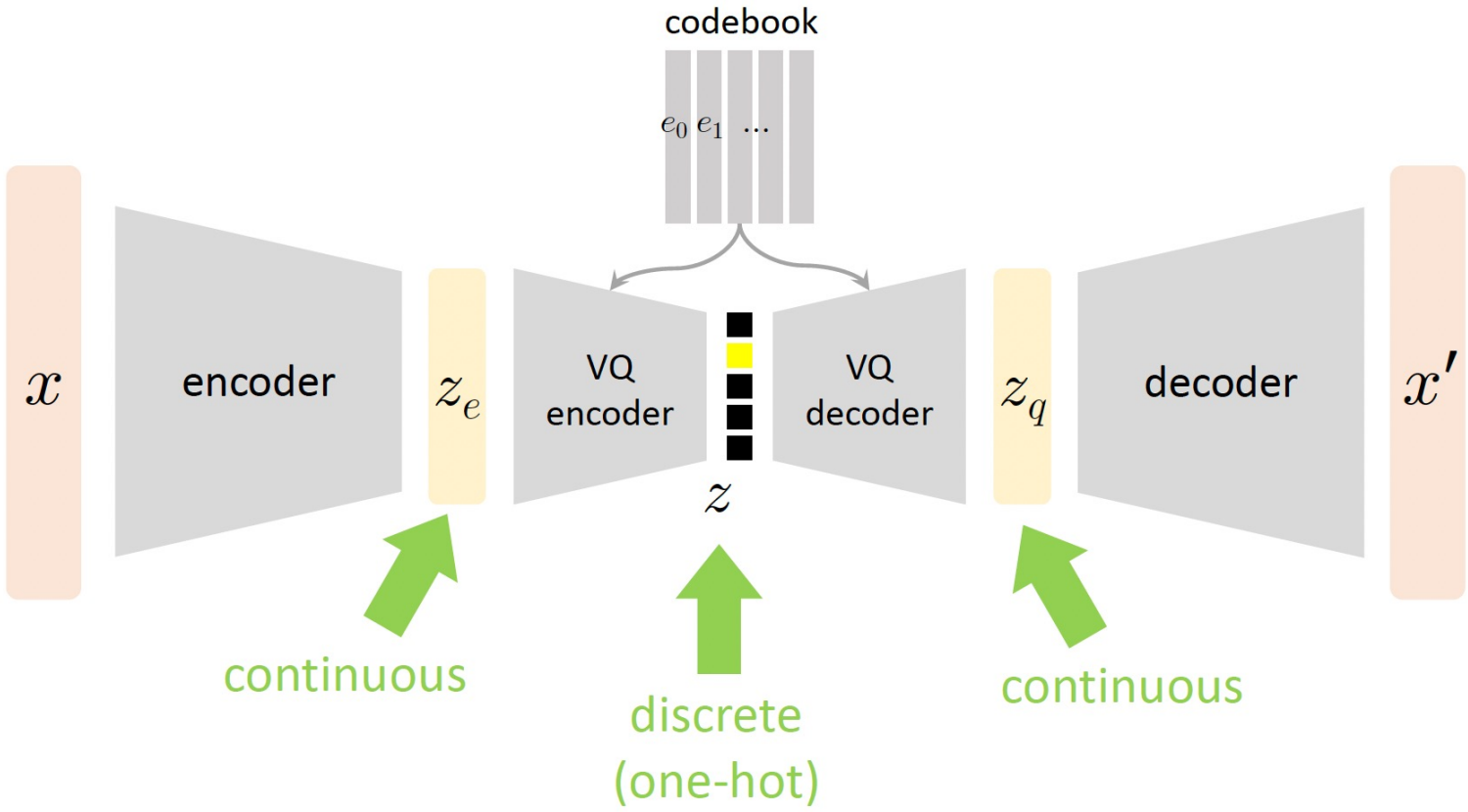
- Vector Quantized VAE (VQ-VAE)
  - Maximize ELBO: reconstruction loss
    - Same as VAE



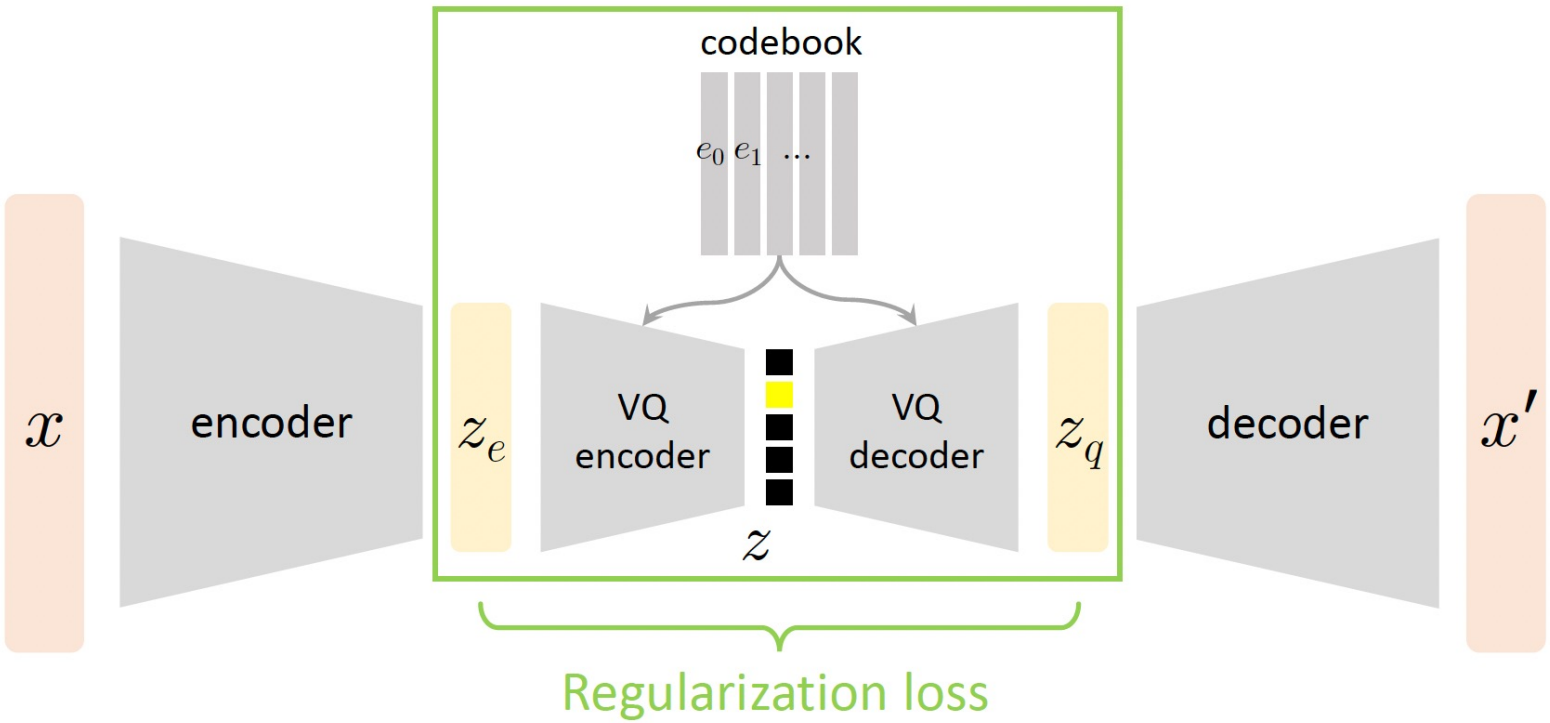
- Vector Quantized VAE (VQ-VAE)
  - Maximize ELBO: regularization loss
    - Conceptually same as VAE, but how can we backprop w.r.t. discrete sampling?



- Vector Quantized VAE (VQ-VAE)
  - Maximize ELBO: regularization loss
    - Conceptually same as VAE, but how can we backprop w.r.t. discrete sampling?
    - K-means



- Vector Quantized VAE (VQ-VAE)
  - Maximize ELBO: regularization loss
    - Conceptually same as VAE, but how can we backprop w.r.t. discrete sampling?
    - K-means

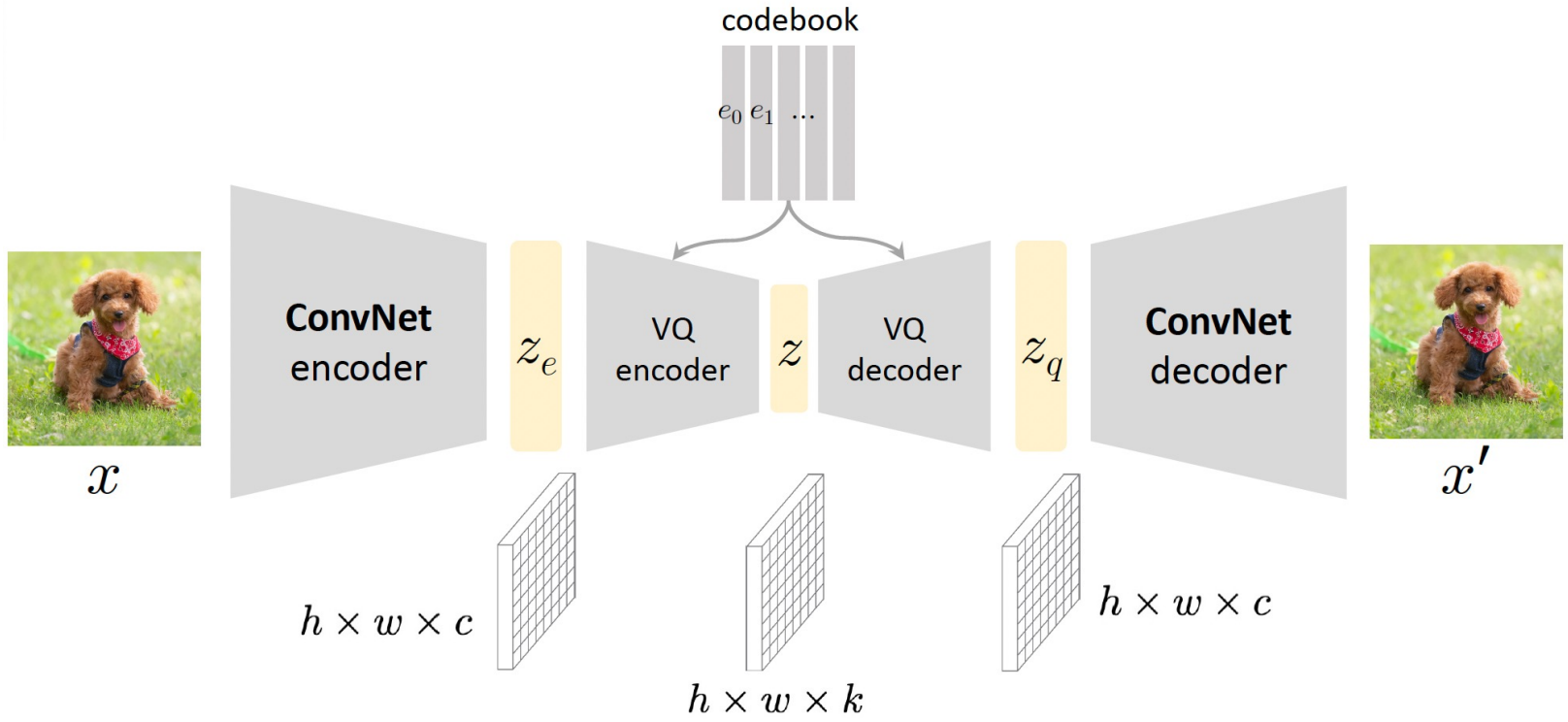


conceptually, this is the K-means reconstruction loss: $\|z_e - z_q\|^2$

- Vector Quantized VAE (VQ-VAE)
  - Maximize ELBO: regularization loss
    - Conceptually same as VAE, but how can we backprop w.r.t. discrete sampling?
    - K-means
    - Backprop through one-hot vector using the 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)`

- Vector Quantized VAE (VQ-VAE)
  - A single one-hot latent is not useful
  - 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



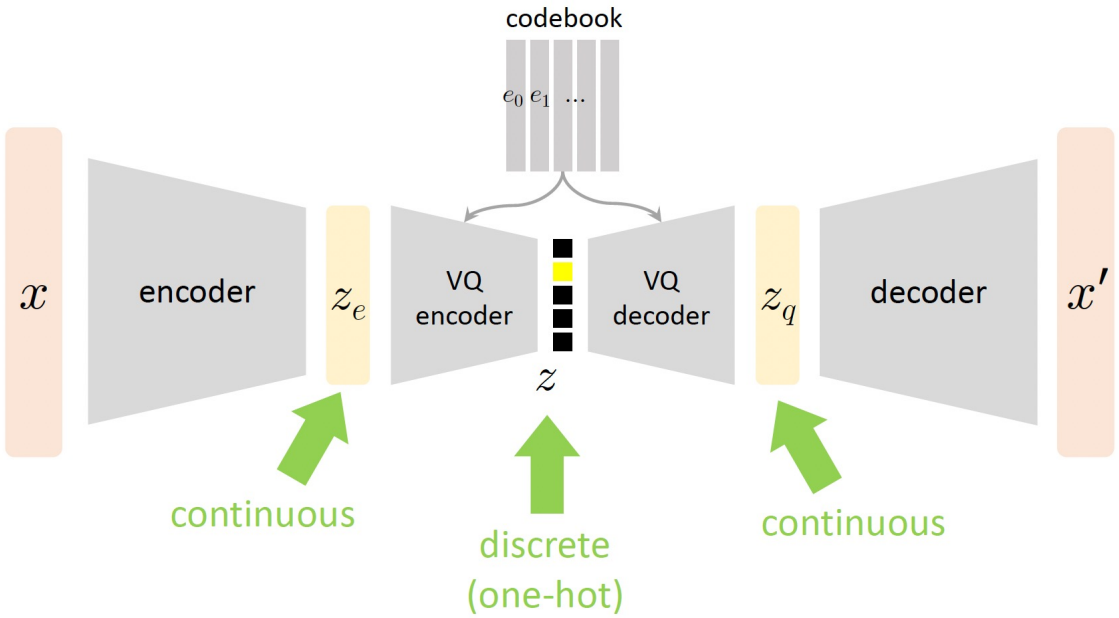
- Vector Quantized VAE (VQ-VAE)
  - A single one-hot latent is not useful
  - VQ-VAE: often used as tokenizers
    - prior p(z) only models per-token (per-location) distribution
    - prior p(z) doesn't model joint distribution across tokens
    - spatial tokens are not independent
    - at inference, we can't sample from i.i.d. prior p(z)

    - Autoregressive models; diffusion models; masked models; …

- Hierarchical VAE
  - Let $p_\theta(z)$ be a simple known prior $p(z)$

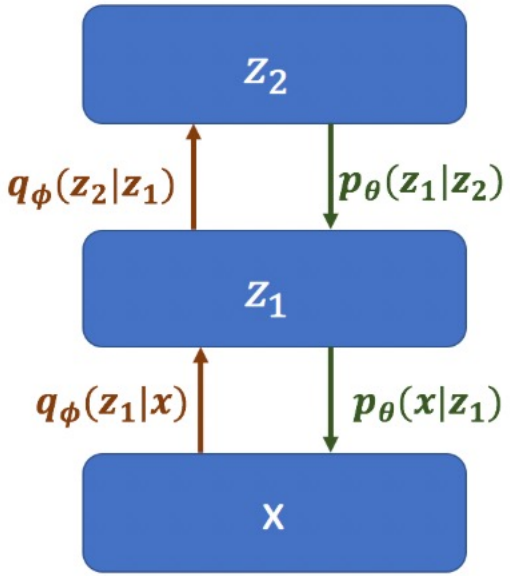
$$\mathbb{E}_{z\sim q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \mathcal{D}_{\mathrm{KL}}\left(q_\phi(z|x)||p(z)\right)$$

tractable tractable



- Hierarchical VAE
  - Stack one VAE on top of another

$$p(x|z) = p(x|z_1) \prod_{\ell=1}^{L-1} p(z_\ell|z_{\ell+1})$$

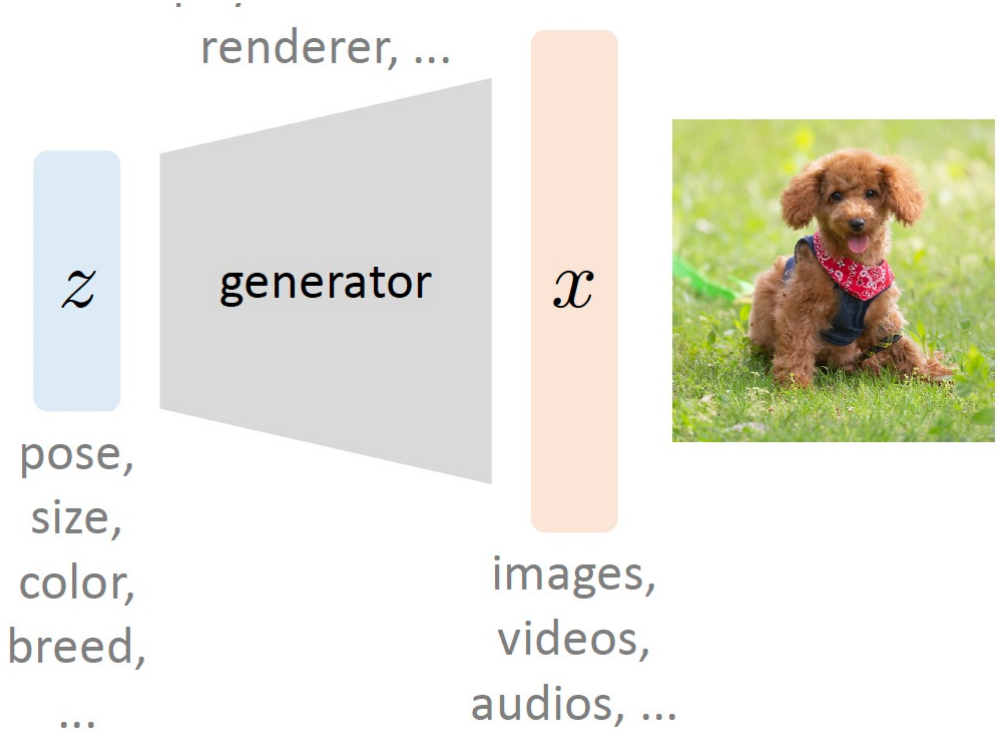


https://ermongroup.github.io/blog/hierarchy/

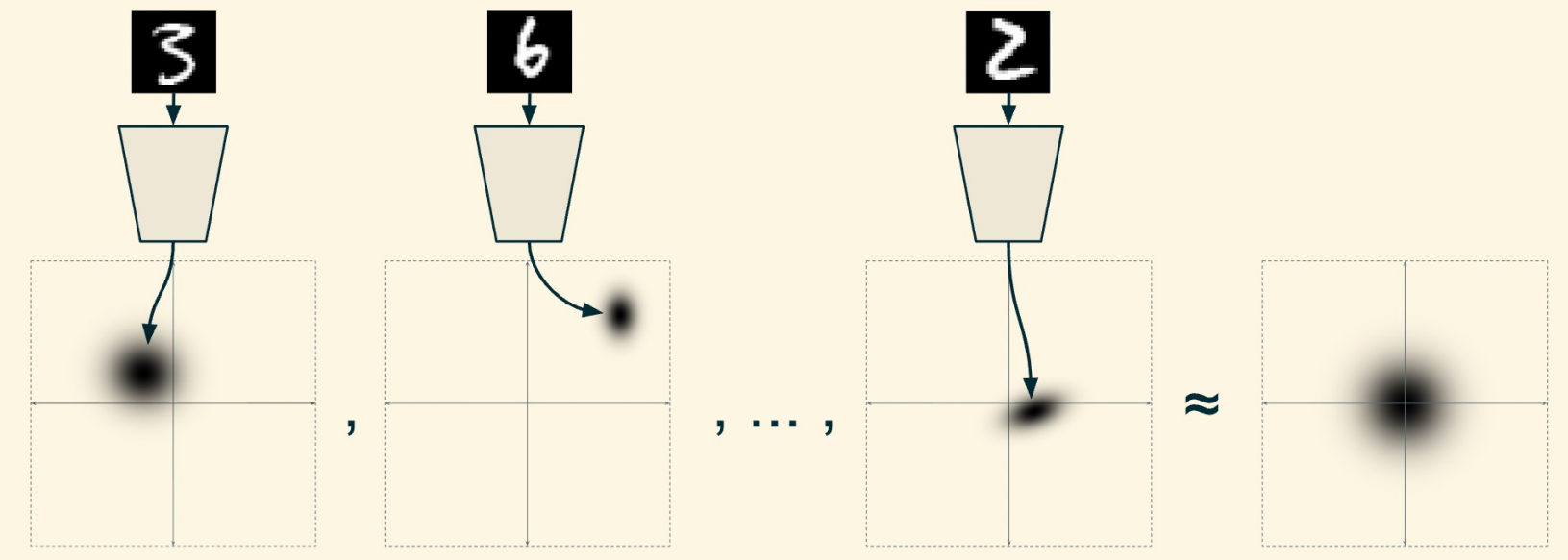
- Hierarchical VAE
  - Stack one VAE on top of another
    - Improved ELBO
    - Hierarchy of latents

$$p(x|z) = p(x|z_1) \prod_{\ell=1}^{L-1} p(z_\ell|z_{\ell+1})$$

- Conditional VAE
  - If some latent variable is known a priori

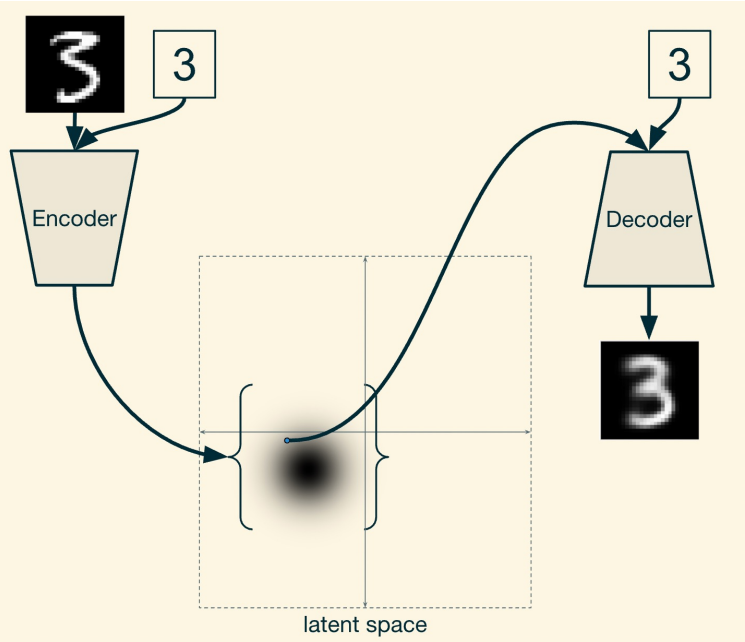


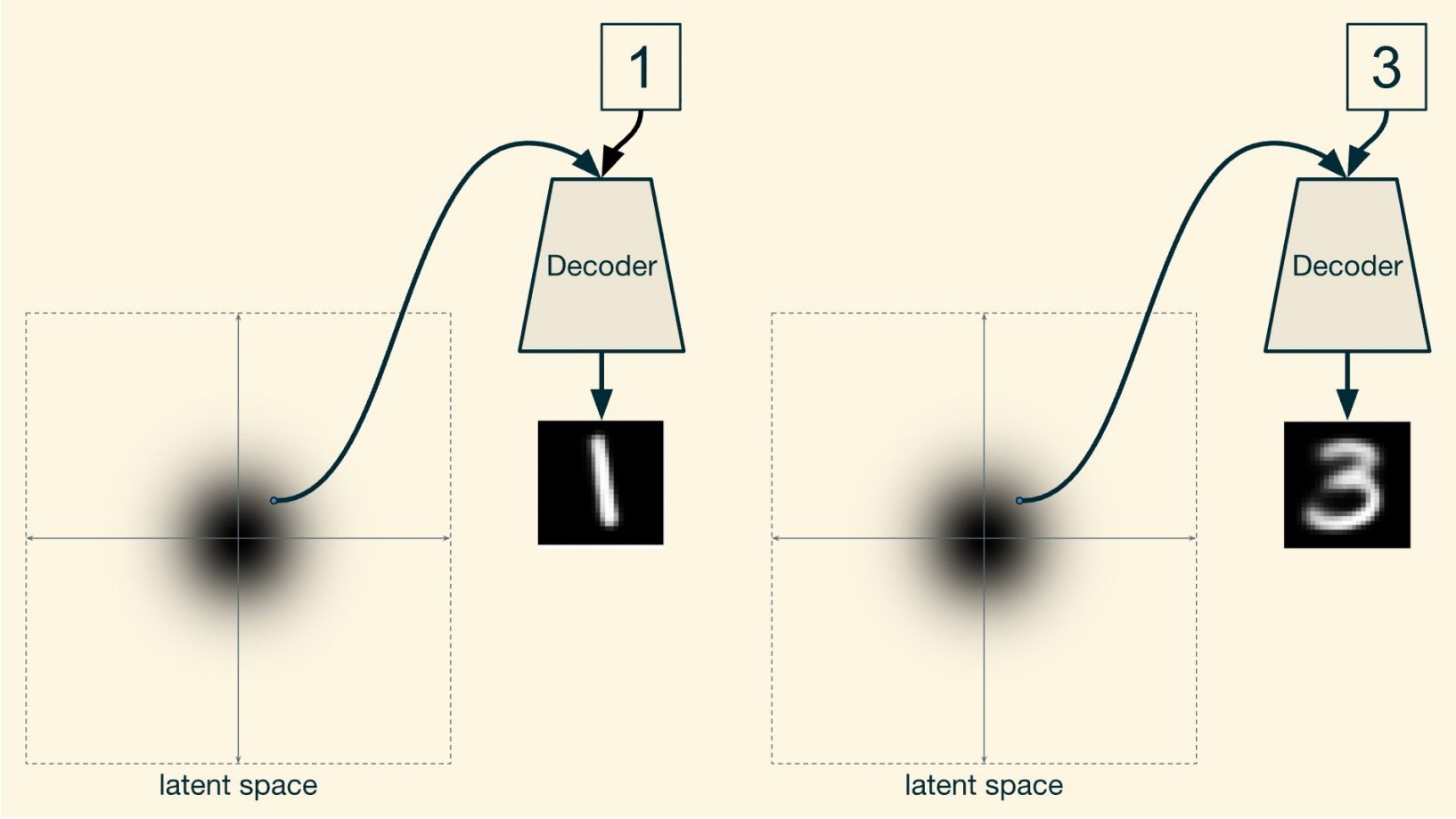
- Conditional VAE
  - If some latent variable is known a priori



https://ijdykeman.github.io/ml/2016/12/21/cvae.html

- Conditional VAE
  - If some latent variable is known a priori, taking it as a conditional input rather than latent variable





https://ijdykeman.github.io/ml/2016/12/21/cvae.html

- Conditional VAE
  - If some latent variable is known a priori, taking it as a conditional input rather than latent variable

https://ijdykeman.github.io/ml/2016/12/21/cvae.html