Deep Learning (BEV033DLE)

Lecture 10: Variational Autoencoders

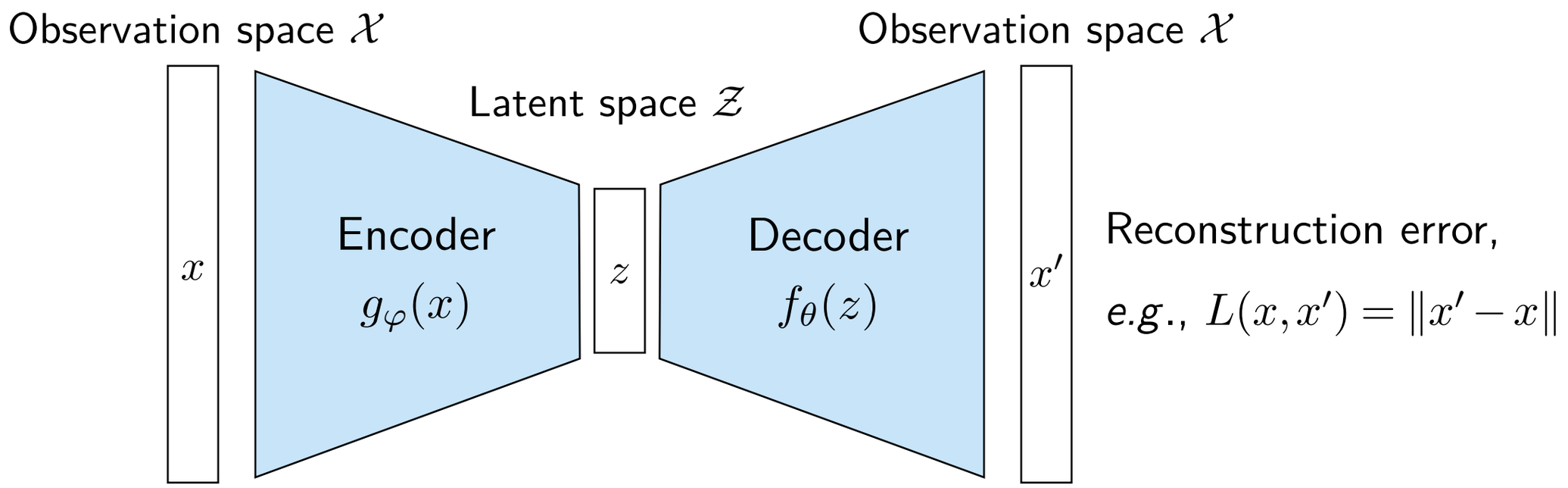
Czech Technical University in Prague

- Background
  - Autoencoders
  - KL divergence
  - Generative models
- Variational Autoencoders
  - Evidence Lower Bound, Variational Optimization, Reparameterization
  - Hierarchical VAEs, Diffusion

Autoencoders

Autoencoders

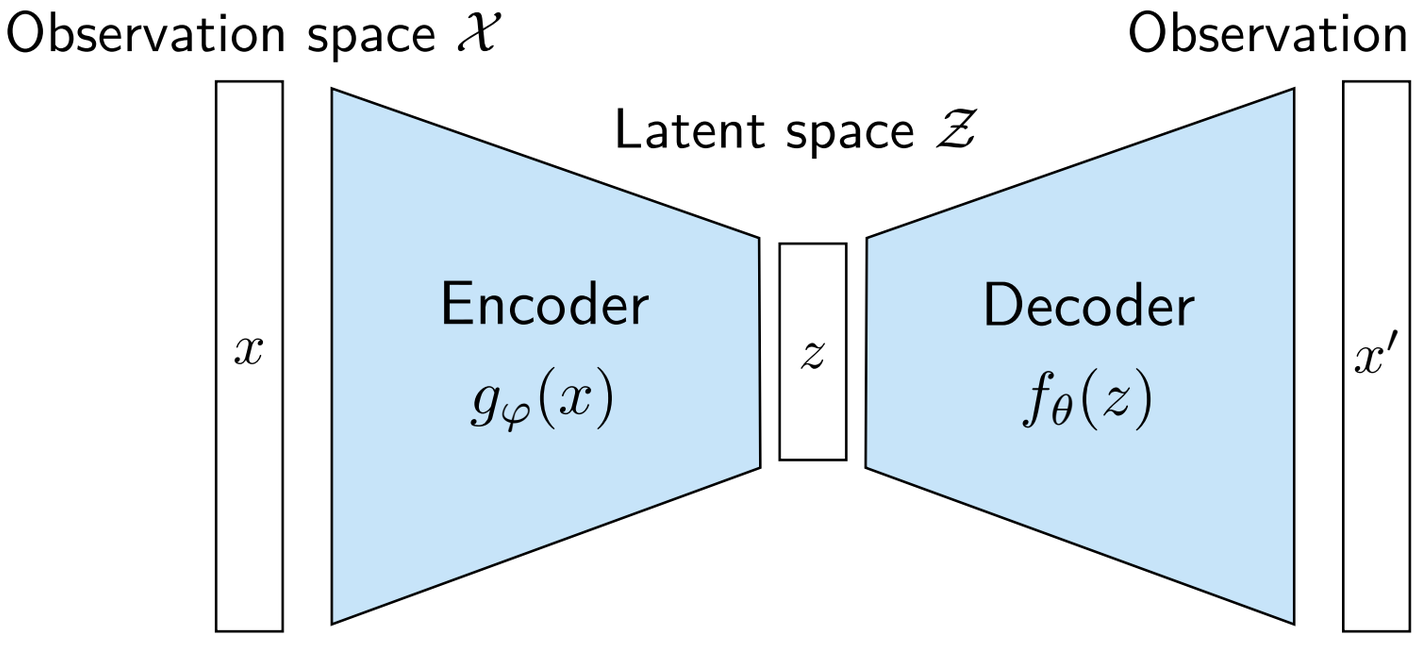
- Motivation / goals:
  - Nonlinear dimensionality reduction (data compression)
  - Learning of features / internal structure in the data in unsupervised way



- Learning: $\mathbb{E}_{x\sim p^*}\Big[L\big(x, x'\big)\Big] = \mathbb{E}_{x\sim p^*}\Big[L\big(x, f_\theta(g_\varphi(x))\big)\Big] \to \min\limits_{\theta,\varphi}$

- Usage:
  - Efficient representations, unsupervised pretraining, anomaly detection

Autoencoders



Reconstruction error,

e.g., $L(x, x') = \|x' - x\|$

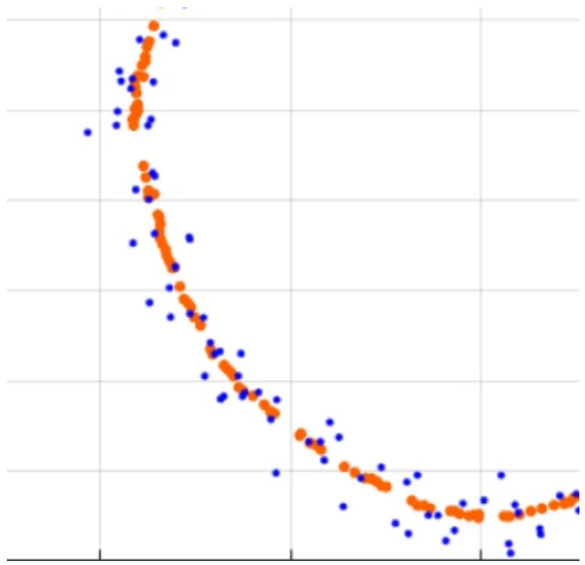
- Variants:

$$\min_{\theta,\varphi} \left(\mathbb{E}_{x\sim p^*} \left[\mathbb{E}_{\delta\sim\mathcal{N}(0,\varepsilon)} \left[L\big(x, f_\theta(g_\varphi(x+\delta))\big) \right] \right] \right. \text{ – denoising}$$

$$\left[\max_{\|\delta\|\le\varepsilon} \left(L\big(x, f_\theta(g_\varphi(x+\delta))\big) \right) \right] \text{ – adversarially robust}$$

$$\left[L\big(x, f_\theta(g_\varphi(x))\big) + \left\| \nabla_\varphi g_\varphi(x) \right\|_F^2 \right] \text{ – contracting}$$

noisy manifold



- Learn representations robust to small changes in the input
- Latent space becomes more smooth (similar codes reconstruct similar images)

Autoencoders

- Example of latent space continuity / smoothness:
  - Interpolation in the latent space between 3 and 8:
    (MNIST data, 2-layer MLPs, 16-dim latent space, training: 10 epochs)

DAE:

CAE:

- Limitations:
  - No principled way to sample new data
  - There may be clusters and gaps in the latent space
  - No probabilistic interpretation

KL Divergence

KL Divergence

- Let $p(x)$ and $q(x)$ be two probability distributions.
- **Kullback–Leibler divergence** of p and q is

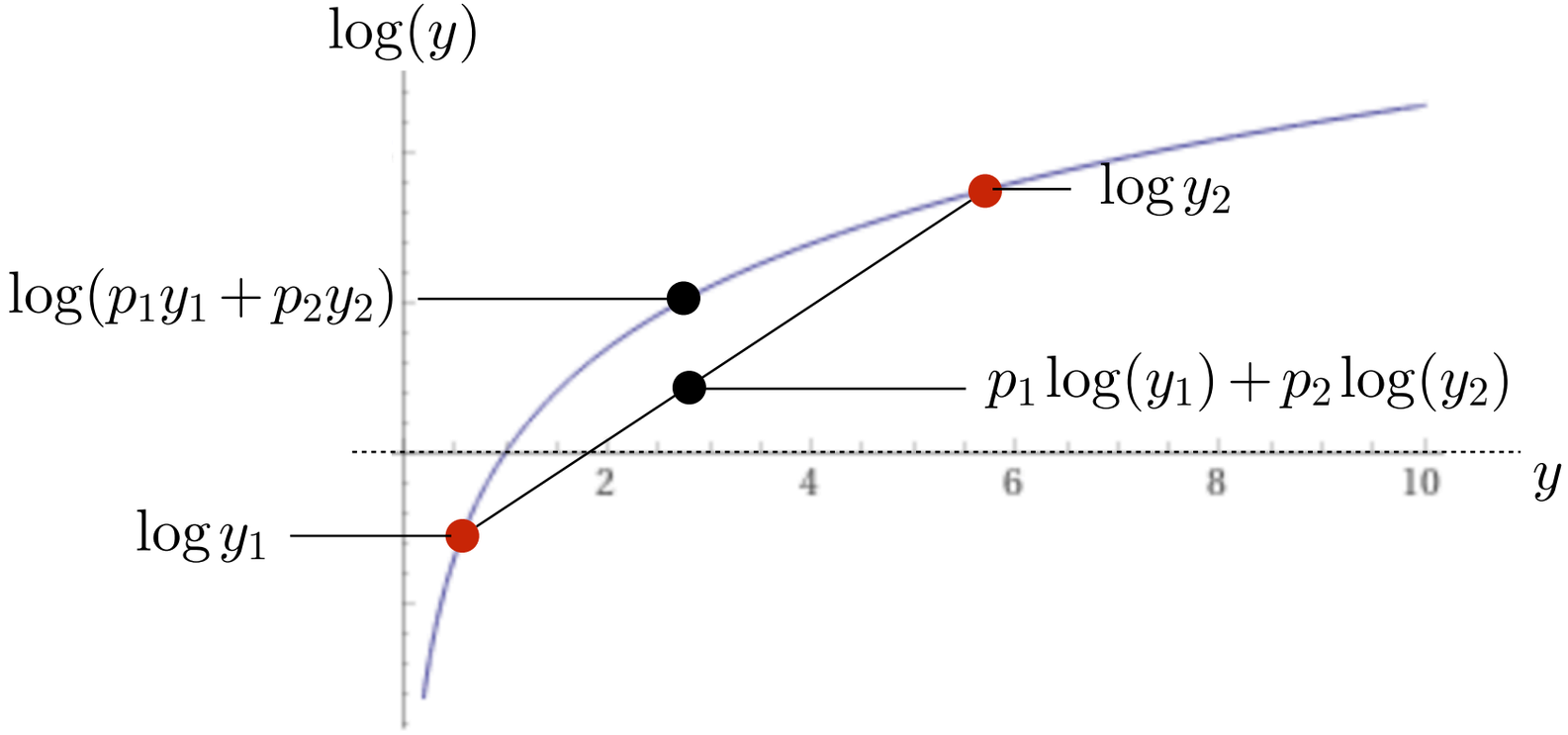
$$D_{\mathrm{KL}}(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

  - Definition allows $p(x) = 0$ by the extension $\lim_{p \to 0} p \log p = 0$
  - Defined when $\mathrm{supp(p)} \subseteq \mathrm{supp(q)}$, i.e. $q(x) = 0 \Rightarrow p(x) = 0$

- **Properties:**
  - D_{KL} is a *divergence*: $D_{\mathrm{KL}}(p \,\|\, q) \geq 0$ with equality iff $q = p$
  - Non-symmetric: $D_{\mathrm{KL}}(p \,\|\, q) \neq D_{\mathrm{KL}}(q \,\|\, p)$
  - Invariant under change of variables

KL non-negativity

- Non-negativity: $D_{\mathrm{KL}}(p\|q) \geq 0$
  - let $y(x) = \frac{q(x)}{p(x)}$
  - The inequality $\sum_x p(x) \log \frac{p(x)}{q(x)} \geq 0$ is equivalent to $\sum_x p(x) \log y(x) \leq 0$
  - Observe that $\log$ is concave, apply Jensen's inequality:
  - $\sum_x p(x) \log y(x) \leq \log \sum_x p(x) y(x) = \log \sum_x q(x) = \log 1 = 0.$
- From strict concavity follows that $D_{\mathrm{KL}}(p\|q) = 0$ iff $p = q$



KL Asymmetry

Minimizing **forward KL** divergence:

$$\min_q D_{\mathrm{KL}}(p\|q)$$

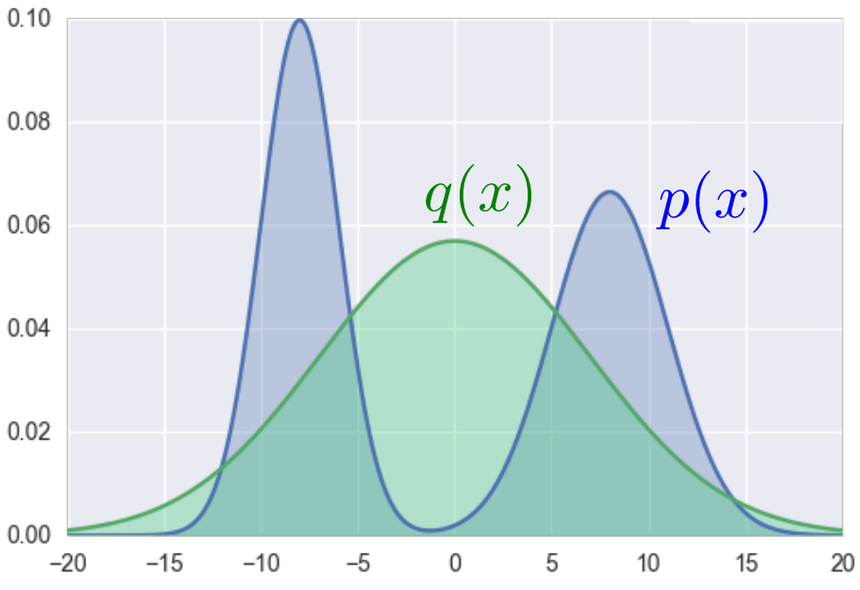
$$\min_q \int p(x)(\log p(x) - \log q(x))dx$$

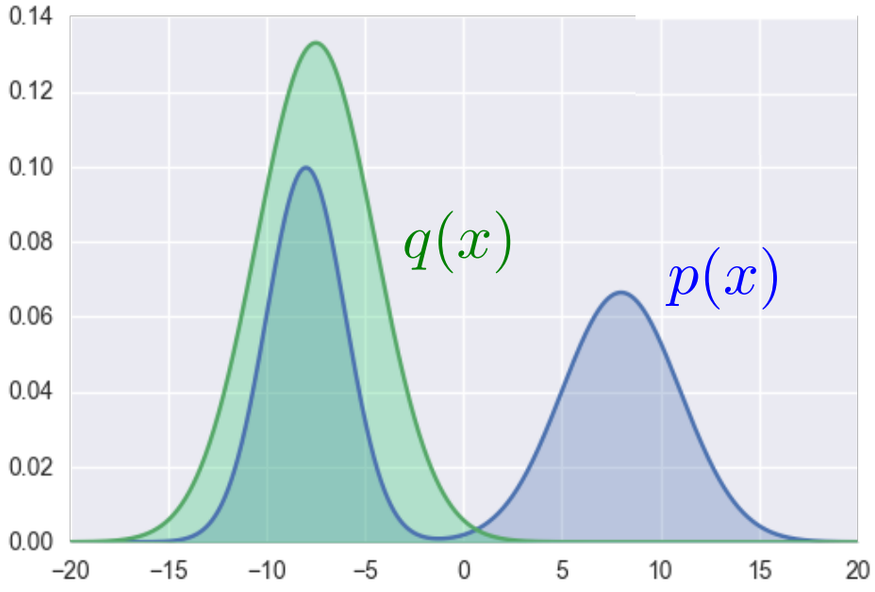
Minimizing **reverse KL** divergence:

$$\min_q D_{\mathrm{KL}}(q\|p)$$

$$\min_q \int q(x)(\log q(x) - \log p(x))dx$$

Example: q is Gaussian





- Approximates well on average in p
- Matches moments for q in EF (e.g. Gaussian)
- Suffices to sample from $p(x)$

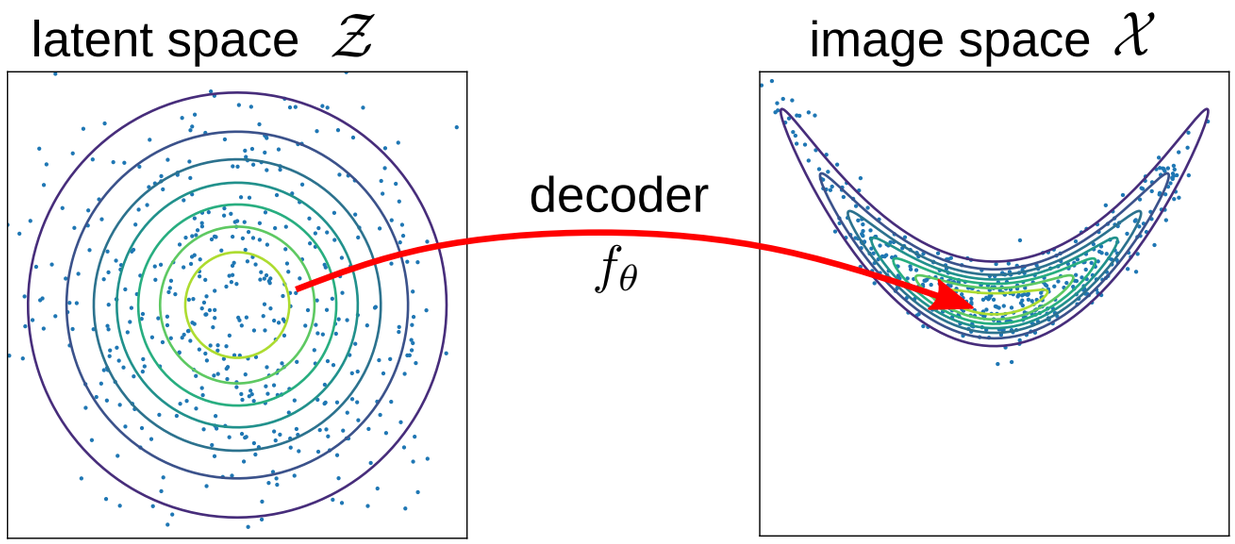
- Approximates well on average in q
- Selects a mode
- Requires $\log(p)$

Generative Models

Generative models: Given training data $\mathcal{T} = \{x_i \,|\, i = 1, \ldots, N\}$ drawn i.i.d. from an unknown distribution $p^*(x)$, the goal is to learn a model that allows to generate random instances of x similar to $x \sim p^*(x)$.

Approach this task by using *latent variable models:*

- fix a latent noise space $\mathcal{Z}$ and a distribution $p(z)$ on it,
- design a neural network f_θ that maps $\mathcal{Z}$ to the feature space $\mathcal{X}$,
- learn its parameters θ so that the resulting distribution $p_\theta(x)$ "reproduces" the data distribution.

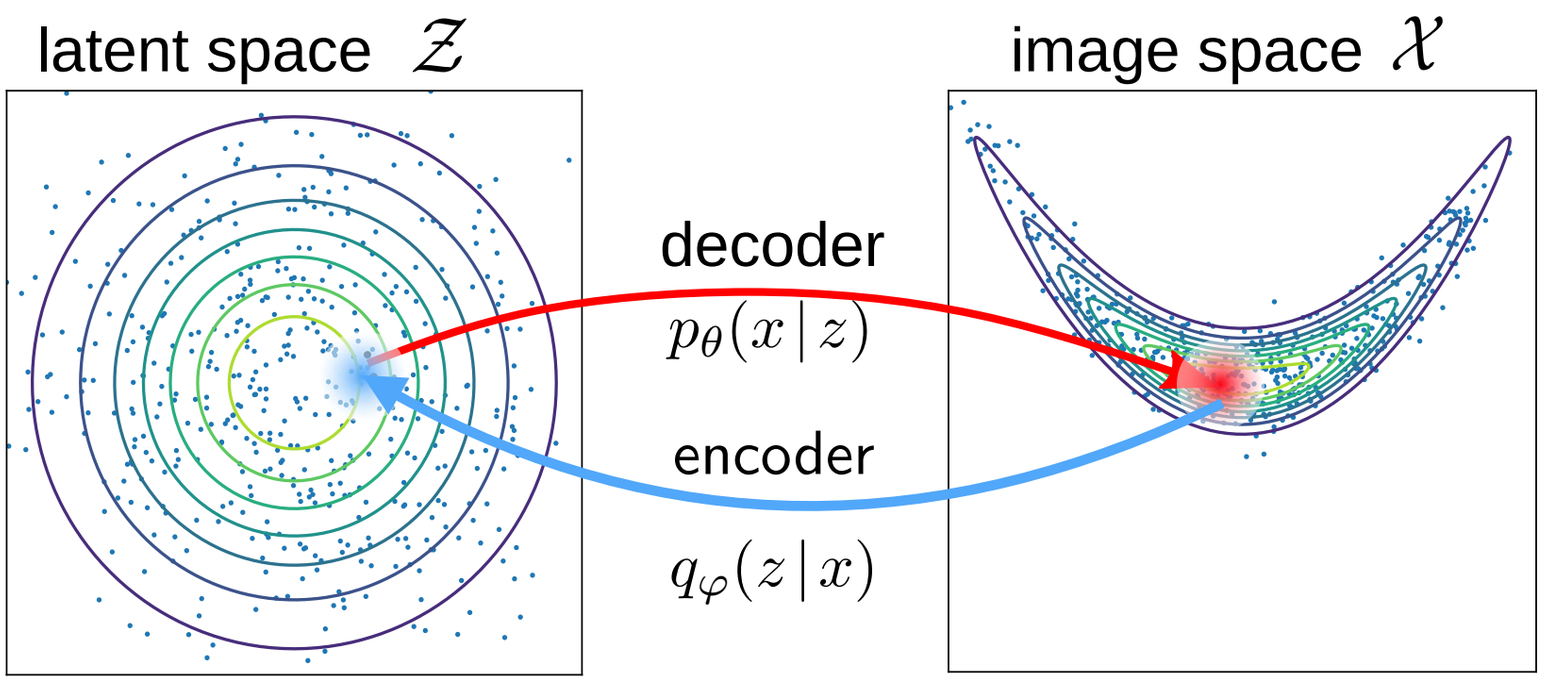


- E.g. maximum likelihood learning: $\log p_\theta(x) = \log p_{\mathcal{Z}}(f_\theta^{-1}(x)) + \log \det \frac{\mathrm{df}_\theta^{-1}(\mathrm{x})}{\mathrm{dx}}$

  — normalizing flow model, need invertible f_θ with tractable determinant Jacobian

(Gaussian) Variational Autoencoders

- Prior distribution $p(z)\colon \mathcal{N}(0, \mathbb{I})$
- **Decoder:** conditional distribution $p_\theta(x \,|\, z)\colon \mathcal{N}(f_\theta(z), \sigma^2 \mathbb{I})$
  - $f_\theta(z)$ is a (deep, convolutional) *decoder network* $\mathcal{Z} \to \mathcal{X}$.
- **Sampling:** $z \sim p(z)$, $x \sim p(x \,|\, z)$, induced marginal distribution $p_\theta(x) = \mathbb{E}_{z \sim p(z)}\big[p(x|z)\big]$



- **Encoder (inference model):** $q_\varphi(z|x)$ will be learned to approximate the decoder posterior $p_\theta(z|x)$ (i.e. probabilistic inverse)
- **Likelihood:** $\log p_\theta(x) = \mathbb{E}_{z \sim q_\varphi(z|x)}\Big[\log p_\theta(x|z)p(z) - \log q_\varphi(z|x)\Big]$ if $q_\varphi(z|x) = p_\theta(z|x)$

Variational Evidence Lower Bound (ELBO)

- Want to maximize the log-likelihood of the data evidence (θ ommited):

$$L = \underbrace{\sum_i \log p(x_i)}_{\text{Evidence}} = \sum_i \log \underbrace{\sum_z p(x_i \,|\, z)\, p(z)}_{\text{difficult in general}}$$

$$= \sum_i \log \sum_z q(z \,|\, x_i) \frac{p(x_i|z|p(z)}{q(z \,|\, x_i)} \geq \underbrace{\sum_i \sum_z q(z \,|\, x_i) \log \frac{p(x_i \,|\, z)\, p(z)}{q(z \,|\, x_i)}}_{\text{Evidence Lower Bound (ELBO)}}$$

Holds for any distribution $q(z \,|\, x_i)$ by Jensen inequality

- Proof using KL (omitting the outer sum in i):

$$\underbrace{\log p(x)}_{\text{Evidence term}} - \underbrace{\sum_z q(z|x) \log \frac{p(x,z)}{q(z|x)}}_{\text{ELBO term}} = \sum_z q(z|x) \Big(\log p(x) - \log \frac{p(x,z)}{q(z|x)} \Big)$$

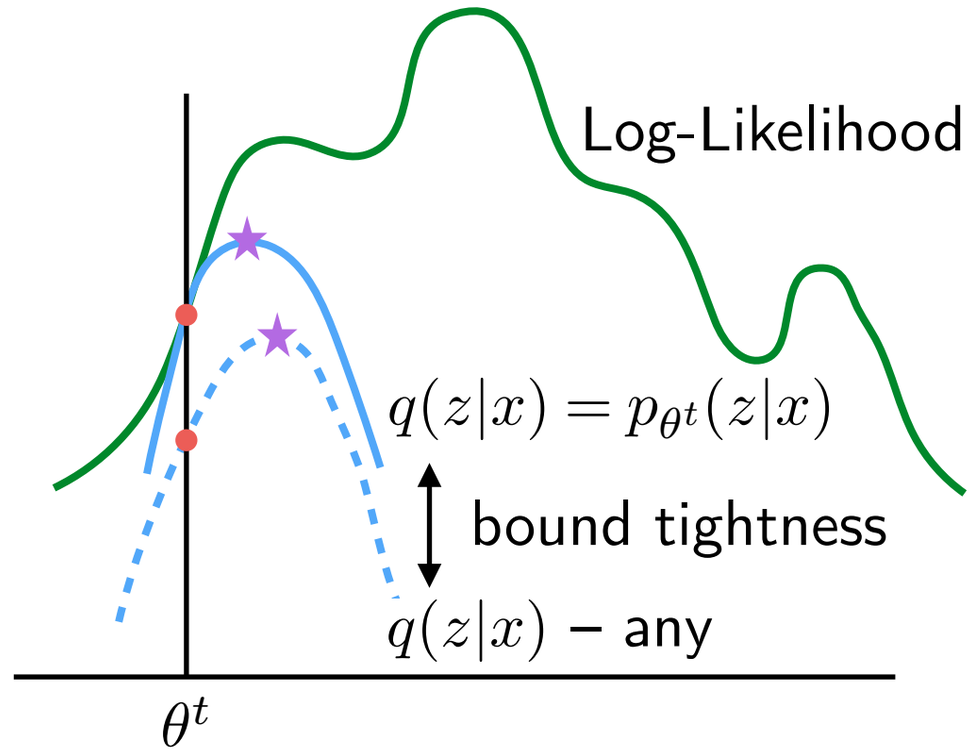
$$= \sum_z q(z|x) \Big(- \log \frac{p(x,z)}{p(x) q(z|x)} \Big)$$

$$= \sum_z q(z|x) \log \frac{q(z|x)}{p(z|x)} = D_{\mathrm{KL}}(q(z|x) \,\|\, p(z|x)) \geq 0.$$

ELBO Optimization

$$L(\theta) \leq \mathrm{ELBO}(\theta, q) = \sum_i \sum_z q\left(z \,|\, x_i\right) \log \frac{p_\theta\left(x_i \,|\, z\right) p(z)}{q\left(z \,|\, x_i\right)}$$

$$= \underbrace{\mathbb{E}_{x \sim p^*, z \sim q(z\,|\,x)} \left[\log p_\theta(x \,|\, z)\right]}_{\text{data fitness part}} + \underbrace{\mathbb{E}_{x \sim p^*} \left[D_{\mathrm{KL}}(q(z \,|\, x) \parallel p(z))\right]}_{\text{“regularizer”}} \qquad (1)$$



- Variational EM:
  - For current q maximize in θ

    — like supervised learning from pairs x, z
  - For current θ maximize in q:

In practice we will optimize (1) but due to identity

$$\mathrm{ELBO}(\theta, q) = L(\theta) - \mathbb{E}_{x \sim p^*} \left[D_{\mathrm{KL}}(q(z \,|\, x) \parallel p_\theta(z|x))\right] \qquad (2)$$

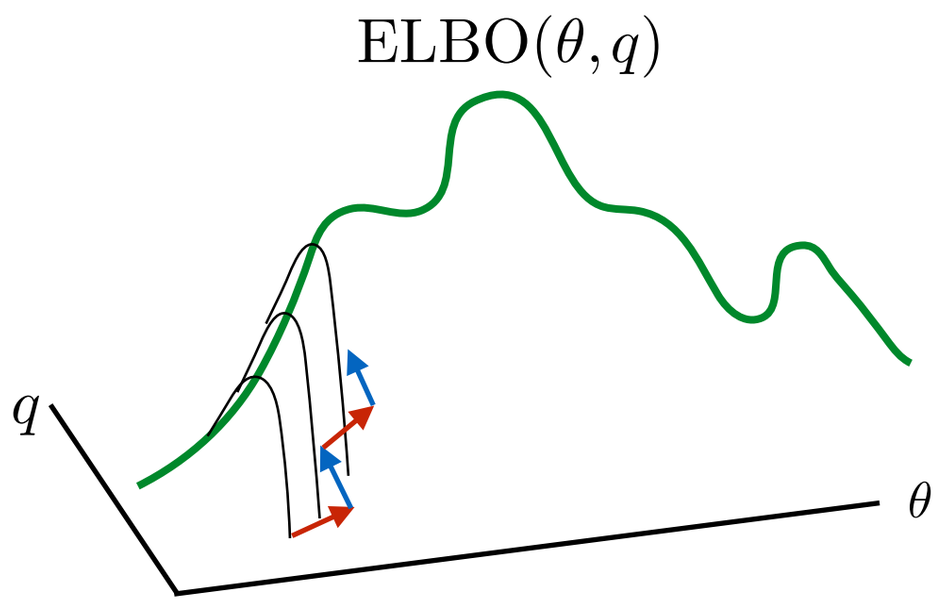
this is equivalent to approximating the decoder posterior in the reverse KL

ELBO Optimization: Amortized Inference

$$L(\theta) \leq \mathrm{ELBO}(\theta, q) = \sum_i \sum_z q\,(z \,|\, x_i) \log \frac{p_\theta\,(x_i \,|\, z)\, p(z)}{q\,(z \,|\, x_i)}$$

$$= \underbrace{\mathbb{E}_{x \sim p^*,\, z \sim q(z\,|\,x)} \big[\log p_\theta(x \,|\, z)\big]}_{\text{data fitness part}} + \underbrace{\mathbb{E}_{x \sim p^*} \big[D_{\mathrm{KL}}(q(z \,|\, x) \,\|\, p(z))\big]}_{\text{“regularizer”}} \qquad (1)$$

Inexact steps are Ok --
we have a global lower bound!



- Consider parametric encoder $q_\varphi(z \,|\, x) = \mathcal{N}(z, \mu_\varphi(x), \sigma^2_\varphi I)$, modeled in terms of a (deep, convolutional) *encoder network* $e_\varphi(x) = (\mu_\varphi(x), \sigma_\varphi(x))$ — amortized inference model.
- $D_{\mathrm{KL}}(q_\varphi(z \,|\, x) \,\|\, p(z))$: since both Gaussians factorize, can be computed in closed form. ✓
- Remains to estimate (stochastically) gradients in θ and φ

ELBO Optimization: Reparameterization Trick

$$\mathrm{ELBO}(\theta, \varphi) = \mathbb{E}_{x \sim p^*} \Big[\mathbb{E}_{q_\varphi(z \,|\, x)} \log p_\theta(x \,|\, z) - D_{KL}(q_\varphi(z \,|\, x) \,\|\, p(z)) \Big]$$

- $\nabla_\theta \mathbb{E}_{q_\varphi(z \,|\, x)} \log p_\theta(x \,|\, z)$: use SGD by sampling $z \sim q_\varphi(z \,|\, x)$. ✓
- $\nabla_\varphi \mathbb{E}_{q_\varphi(z \,|\, x)} \log p_\theta(x \,|\, z)$: this gradient is *critical*.

  We can not replace $\mathbb{E}_{q_\varphi(z \,|\, x)}$ by a sample $z \sim q_\varphi(z \,|\, x)$, because it will depend on φ!

Re-parametrisation trick: Simple solution for Gaussians:

$$z \sim \mathcal{N}(\mu, \sigma^2) \iff \varepsilon \sim \mathcal{N}(0,1) \text{ and } z = \sigma\varepsilon + \mu$$

Now, if μ and σ depend on φ:

$$\nabla_\varphi \mathbb{E}_{z \sim \mathcal{N}(\mu_\varphi, {\sigma_\varphi}^2)} [f(z)] = \nabla_\varphi \mathbb{E}_{\varepsilon \sim \mathcal{N}(0,1)} \big[f(\sigma_\varphi \varepsilon + \mu_\varphi) \big] = \mathbb{E}_{\varepsilon \sim \mathcal{N}(0,1)} \big[\nabla_\varphi f(\sigma_\varphi \varepsilon + \mu_\varphi) \big]$$

ELBO Optimization

Overall, the learning step for a (Gaussian) VAE is pretty simple:

Fetch a mini-batch x from training data

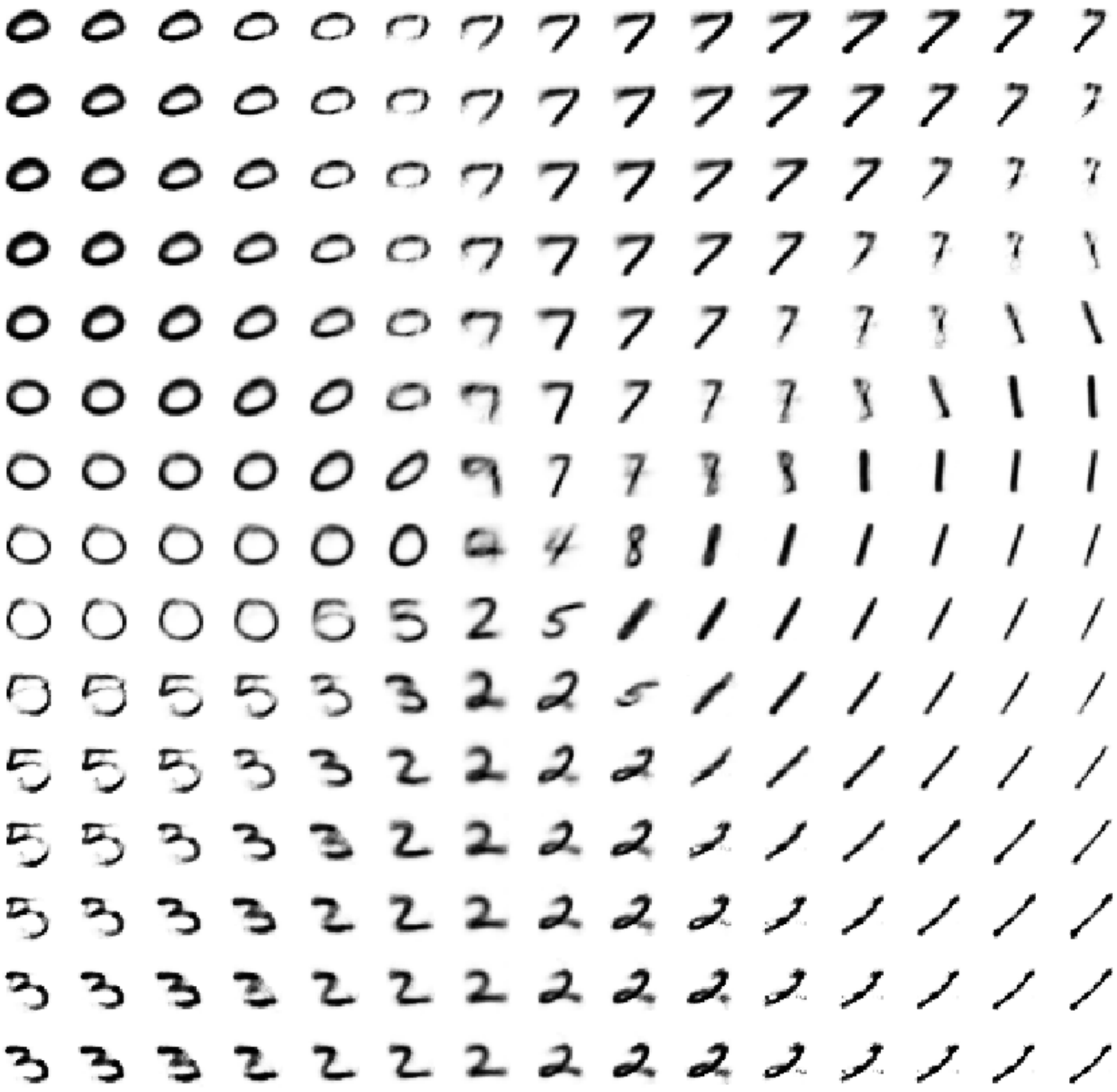
1. apply the encoder network $g_\varphi(x) \mapsto \big(\mu_\varphi(x), \sigma_\varphi(x)\big)$
2. compute the KL-divergence $D_{KL}(q_\varphi(z\,|\,x) \,\|\, p(z))$
3. sample a batch $z \sim q_\varphi(z\,|\,x)$ with reparameterization
4. apply the decoder network $f_\theta(z) \mapsto \mu_\theta(z)$ and compute $\log p_\theta(x\,|\,z)$
5. combine the ELBO terms and let PyTorch compute the derivatives and make an SGD step.

Strengths and weaknesses of VAEs

- concise model, simple objective (ELBO), can be optimised by SGD ✓
- local optima, *posterior collapse*: some latent components collapse to $q_\varphi(z_i\,|\,x) = p(z_i)$, and decoder does not depend on them, *i.e.* they carry no information. ✗
- amortised inference models $q_\varphi(z\,|\,x)$ have not enough expressive power to close the gap between $L(\theta)$ and $\mathrm{ELBO}(\theta, \varphi)$ for complex data distributions ✗

Example: Latent Space Smoothness

- MNIST latent space bilinear interpolation



Hierarchical Variational Autoencoders

Closing the gap between $L(\theta)$ and $L_B(\theta, \varphi)$:

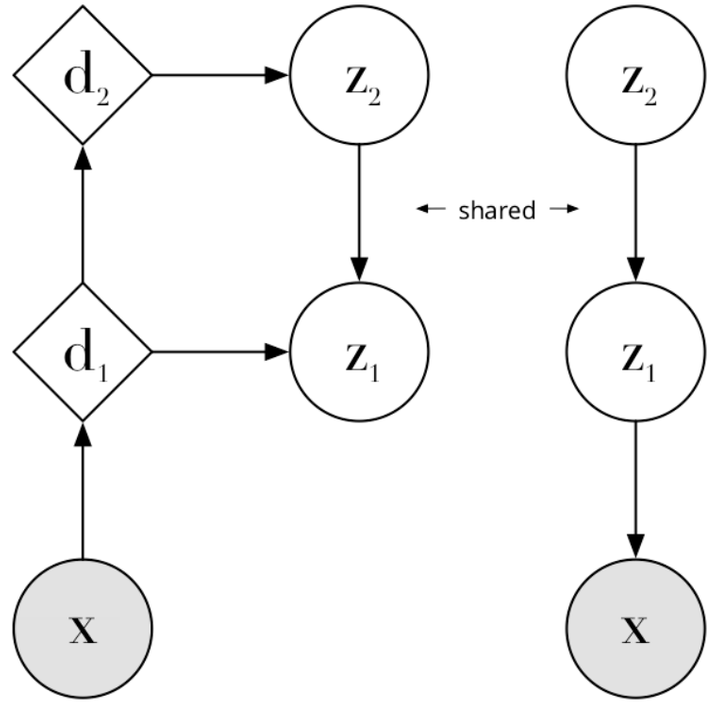
The latent state z consists of variable groups $z_1, \ldots, z_m$.

$$p_\theta(x, z) = p(z_m) \prod_{i=1}^{m-1} p_\theta(z_i \,|\, z_{>i})\, p_\theta(x \,|\, z); \quad q_\varphi(z \,|\, x) = q_\varphi(z_m \,|\, x) \prod_{i=1}^{m-1} q_\varphi(z_i \,|\, z_{>i}, x).$$

The encoder shares parameters with the decoder, by assuming

$$q_{\theta,\varphi}(z_i \,|\, z_{>i}, x) \propto p_\theta(z_i \,|\, z_{>i}) d_i(z_i, x, \varphi),$$

where the functions d_i are hidden layer outputs of a deterministic encoder network whose forward direction is reverse to the factorisation order of the model.



Hierarchical Variational Autoencoders

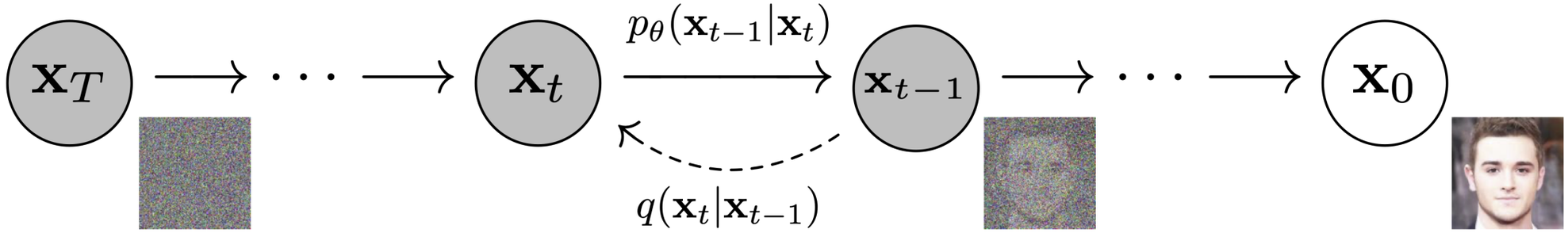
Hierarchical VAEs can be learned by maximising ELBO.

For instance

$$D_{KL}(q_\varphi(z\,|\,x)\,\|\,p(z)) = D_{KL}(q_\varphi(z_m\,|\,x)\,\|\,p(z_m))+ \int dz_m\, q_\varphi(z_m\,|\,x) D_{KL}(q_\varphi(z_{m-1}\,|\,z_m,x)\,\|\,p_\theta(z_{m-1}\,|\,z_m)) + \ldots$$

A. Vahdat et al., NeurIPS 2020: A Deep Hierarchical VAE trained on CelebA data.

Diffusion Models



Diffusion models are homogeneous hierarchical VAEs with the latent spaces same as image: $\mathcal{Z}_i = \mathcal{X}$.

- The encoder $q(x_t, |\, x_{t-1}) = \mathcal{N}(x_t, \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ is fixed and gradually adds Gaussian noise to the data. (diffusion forward process)
- The decoder is given by $p_\theta(x_{t-1}\,|\,x_t, \beta_t) \sim \mathcal{N}(x_{t-1}, \mu_\theta(x_t, t), \sigma_t^2 I)$ and is implemented by a deep network (typically a UNet). Its parameters θ are shared for all t.

The limiting distribution of the encoder (for $t \to \infty$) is pixel-wise independent Gaussian noise.

The limiting distribution of the trained decoder matches the data distribution.

Diffusion Models

J. Ho et al., NeurIPS 2020, Denoising diffusion probabilistic models