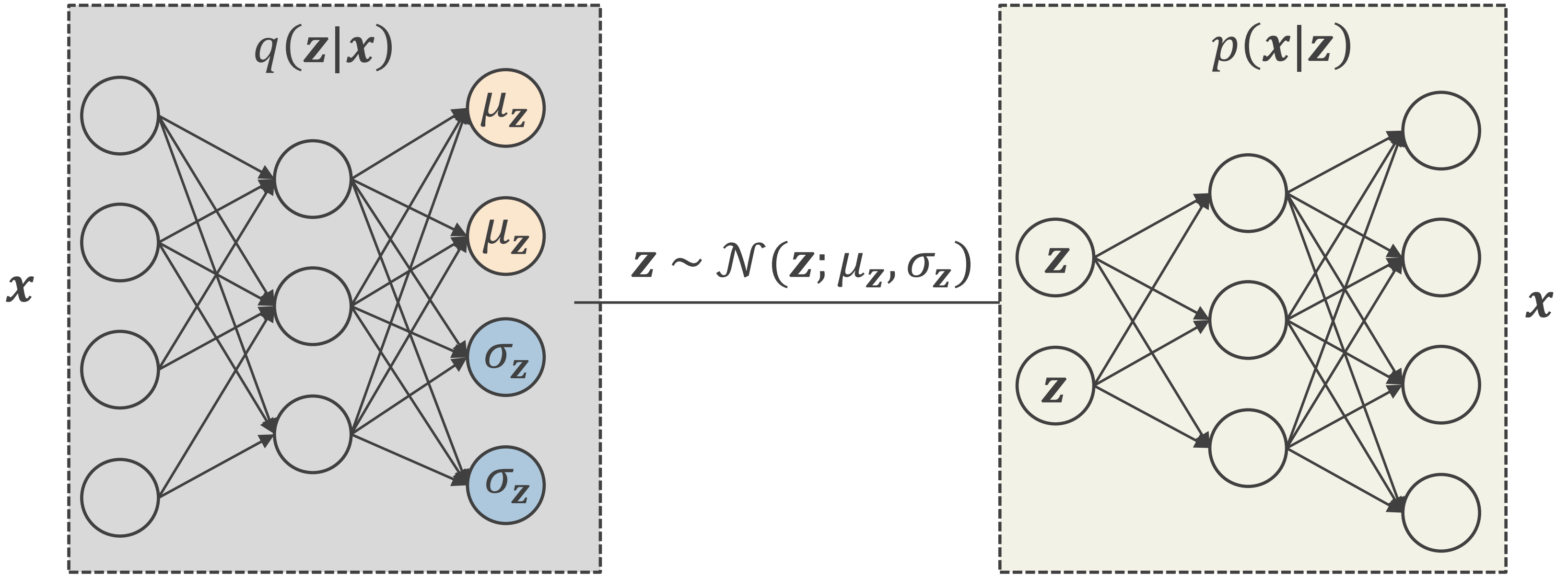
Variational autoencoders

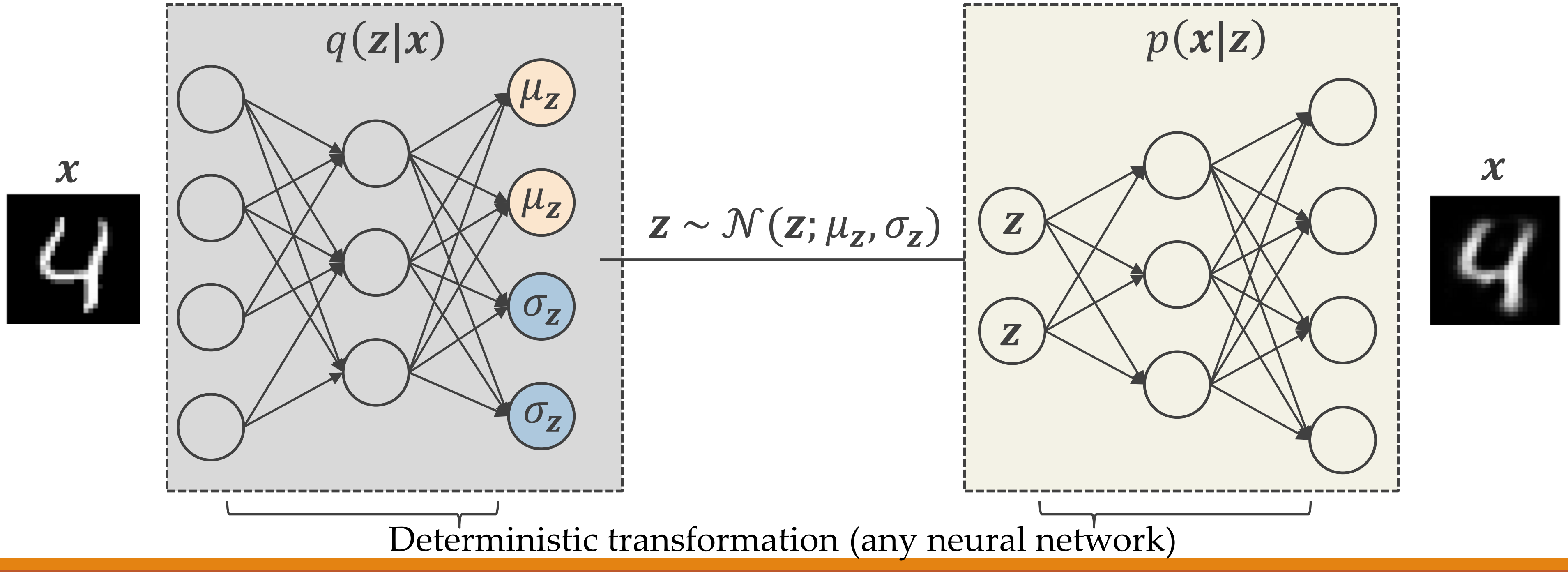


Variational autoencoders

- Variational autoencoders is the neural network implementation of the ELBO

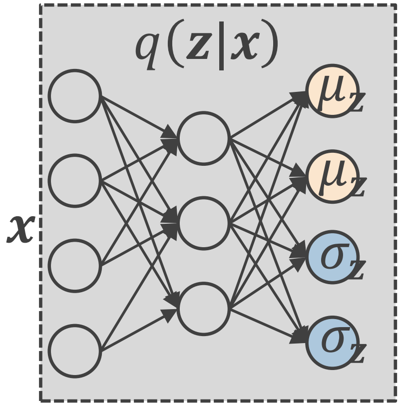
$$\text{ELBO} = \mathbb{E}_{q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \text{KL}\left[q_{\varphi}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right]$$

- In the standard case the approximate posterior is Gaussian $q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \mu_{\boldsymbol{z}}, \sigma_{\boldsymbol{z}})$



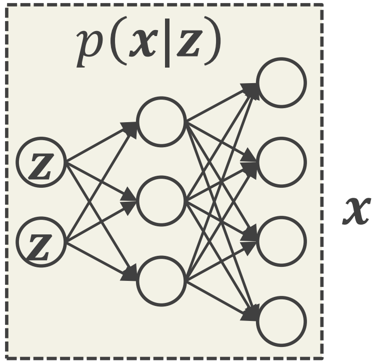
Encoder/inference network ⇔ approximate posterior

- The encoder is any standard neural network
  - Modelling the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$
  - Remember: given input $\boldsymbol{x}$ we have a distribution over latent $\boldsymbol{z}$ (not single value)
  - The KL term $\mathrm{KL}[q(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})]$ encourages the posterior to not deviate too much from the prior $p(\boldsymbol{z})$
- For Gaussian $q(\boldsymbol{z}|\boldsymbol{x})$ we need two neural networks for two outputs $\mu_{\boldsymbol{z}}$, $\sigma_{\boldsymbol{z}}$
  - The $\mu_{\boldsymbol{z}}$ is a neural net encoding the mean of $\boldsymbol{z}$ given $\boldsymbol{x}$
  - The $\sigma_{\boldsymbol{z}}$ is a neural net encoding the stdev of $\boldsymbol{z}$ given $\boldsymbol{x}$
  - The two neural nets can share architecture before the outputs



Decoder network ⇔ generative model

- The decoder model is also a neural network
  - It receives a stochastic input $\boldsymbol{z}$ and returns as output a generation
- The output modelled with a distribution according to the data type
  - For continuous values could be a Gaussian
  - For binary values Bernoulli distribution
- With generative models often convenient to think of the generation process
  - Then the encoder is the variational approximation to ensure tractability
- Check the graphical model
  - Sample $\boldsymbol{z} \sim p(\boldsymbol{z})$ from the prior
  - Given $\boldsymbol{z}$ generate $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z})$



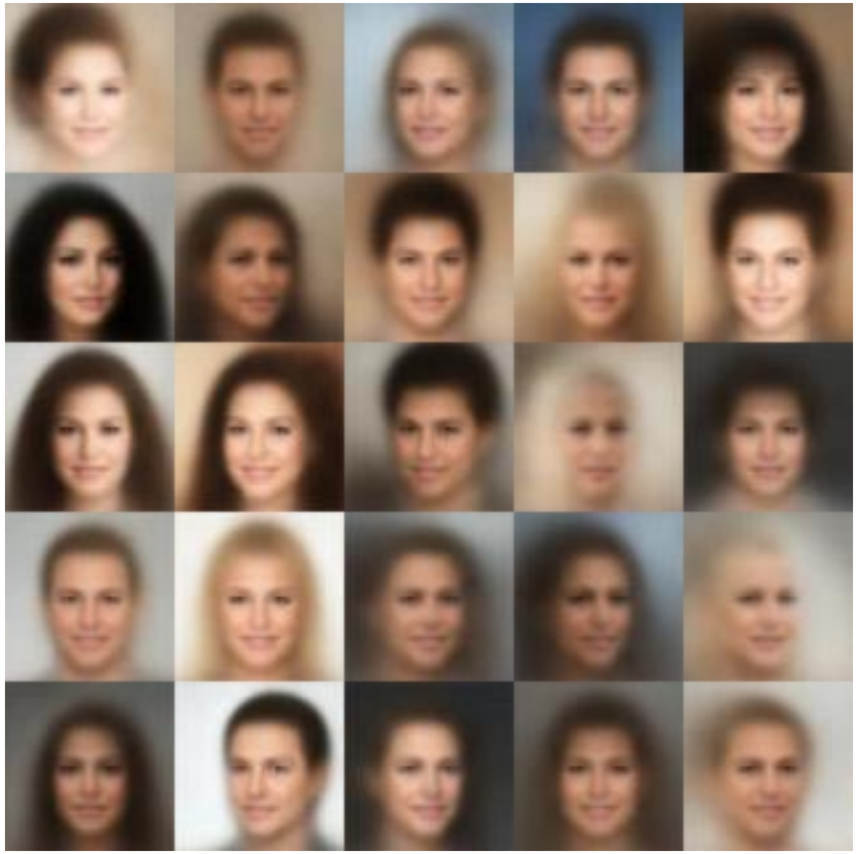
Prior distribution

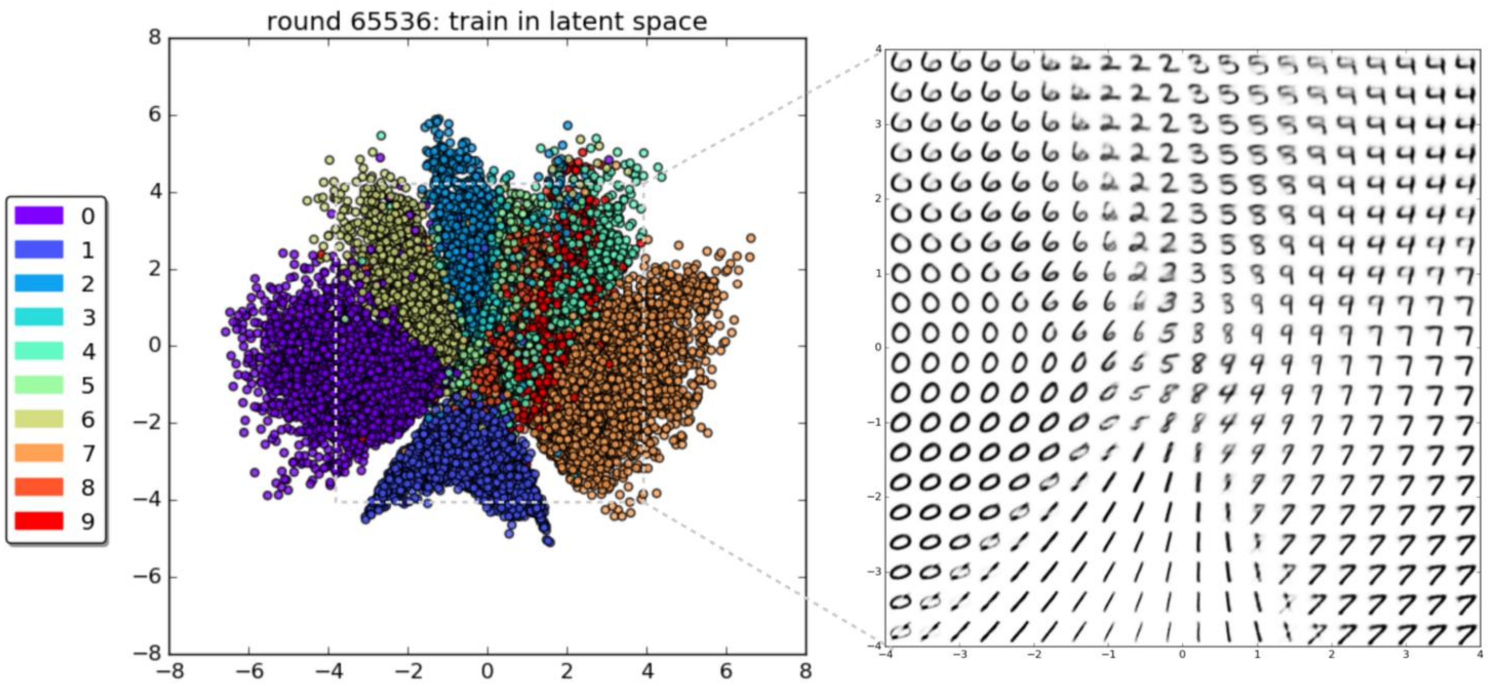
- The prior distribution acts as a regularizer
- The prior $p(\boldsymbol{z})$ is often the unit Gaussian $p(\boldsymbol{z}) \sim \mathcal{N}(0,1)$
- If we expect/desire different nature of $\boldsymbol{z}$, *e.g.*, sparsity or binary latents
  - → pick a different prior
  - The sampled $\boldsymbol{z}$ will be from that prior
  - The KL term will regularize the encoder to be close to the prior

Learning the variational autoencoders

- The variational autoencoder is two neural networks with inputs or outputs that are stochastic (represented by distributions, not single values)
- We must train the neural networks
  - I.e., fit good parameters $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$ for the decoder
- Objectives:
  - We want to predict to good distributions for $\boldsymbol{z}$ for (seen & unseen) inputs $\boldsymbol{x}$
  - We want on average our approximate posterior to be close to the prior
  - We want to reconstruct inputs well
  - We want generations that look 'real' → good extrapolations

Interpolation in the latent VAE space





Training Variational Autoencoders

- Maximize the ELBO

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_{q_\varphi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\varphi(z|x)||p(\mathrm{z}))$$

$$= \int_{\boldsymbol{z}} q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} - \int_{\boldsymbol{z}} q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\, d\boldsymbol{z}$$

- Normally you derive the math between each integral
  - Good exercise: derive the ELBO for Gaussian latents and Bernoulli outputs

- Often, the integrals make some terms intractable. How to train?
  - Backpropagation with Monte Carlo (MC) averaging
  - Forward propagation means evaluating the two terms
  - Backpropagation → compute gradients with respect to the $\boldsymbol{\theta}$ and $\boldsymbol{\varphi}$

Training reconstruction term

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \int_{\boldsymbol{z}} q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} - \int_{\boldsymbol{z}} q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})} d\boldsymbol{z}$$

- The first term is an integral (expectation) that we cannot solve analytically
  - Sample from the approximate posterior $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ instead and do MC average
  - Pick a $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ that couples well with log
- With a 'low variance estimator' a single sample $\boldsymbol{z}$ is enough
  - Stochasticity is desirable → reduces overfitting
- Reparameterization trick for low variance estimation

Training KL regularization term

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \int_{\boldsymbol{z}} q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\, d\boldsymbol{z} - \int_{\boldsymbol{z}} q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \log \frac{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\, d\boldsymbol{z}$$

- The second term is an integral which corresponds to KL distance
- For known distributions, *e.g.*, both $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z})$ Gaussians, the KL often reduces to a closed formula → very convenient
  - E.g., compute the KL divergence between a centered $N(0,1)$ and a non-centered $N(\mu, \sigma)$ gaussian
- If closed formula not easy, MC averaging with sampling from $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ is possible