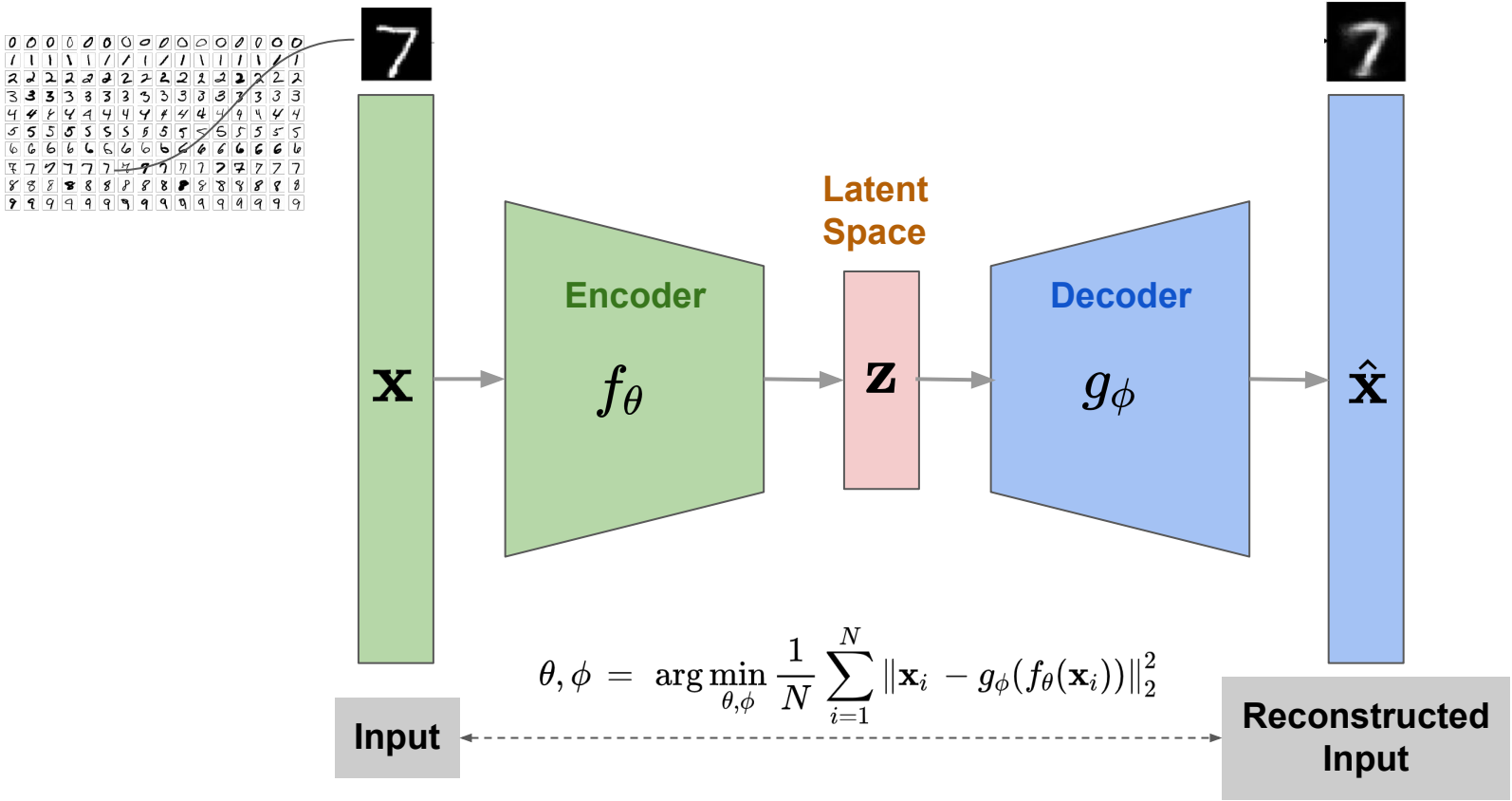
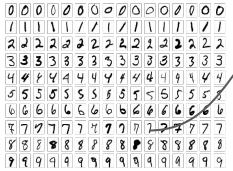
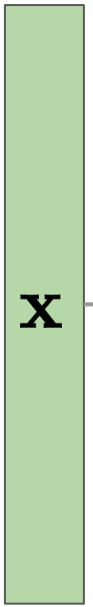
Variational Autoencoder (VAE)

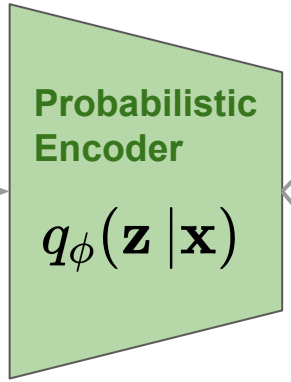
University of Victoria - PHYS-555

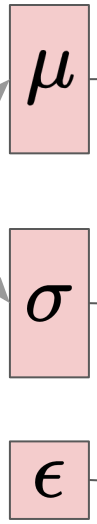

Latent
Space
Encoder
f_θ
Decoder
g_ϕ
$\mathbf{x}$
$\mathbf{z}$
$\hat{\mathbf{x}}$
$\theta, \phi = \arg\min_{\theta,\phi} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - g_\phi(f_\theta(\mathbf{x}_i))\|_2^2$
Input
Reconstructed Input

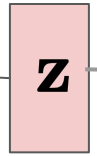


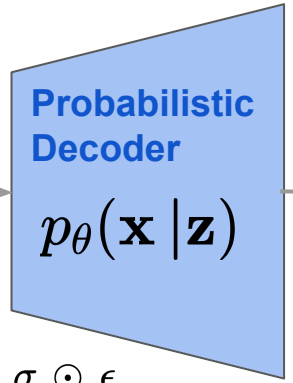
$$\mathcal{L}(x;\theta,\phi) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction term}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{prior term}}$$











$$\mathbf{z} = \mu + \sigma \odot \epsilon$$

Why do we need a VAE?

VAE are generative models: sample a latent vector, decode to get a new sample.

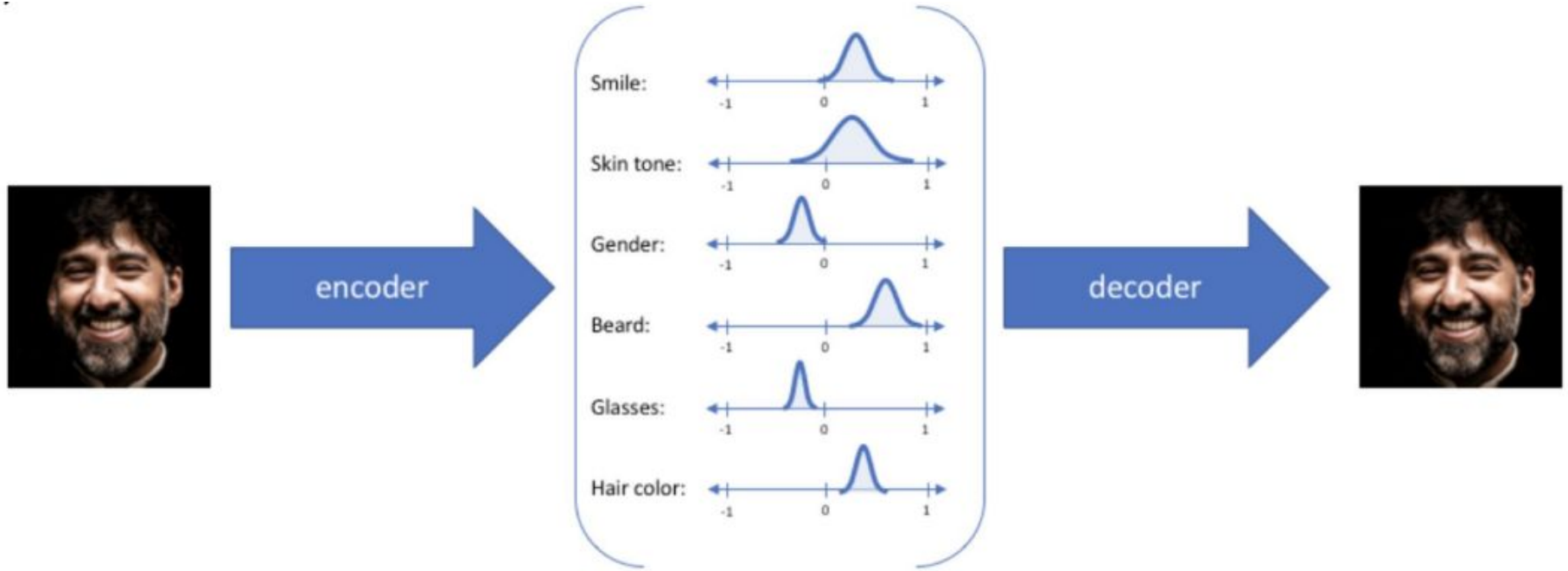
Q: Why can't we use a (deterministic) autoencoder?
A: If we choose an arbitrary latent vector, we are not guaranteed to be close to any points in the training set and the reconstruction is garbage

Q: How can we avoid it?
A: Have a more compact latent space.

Q: How can we do this:
A: 1) encode in balls rather than points 2) bring balls closer together

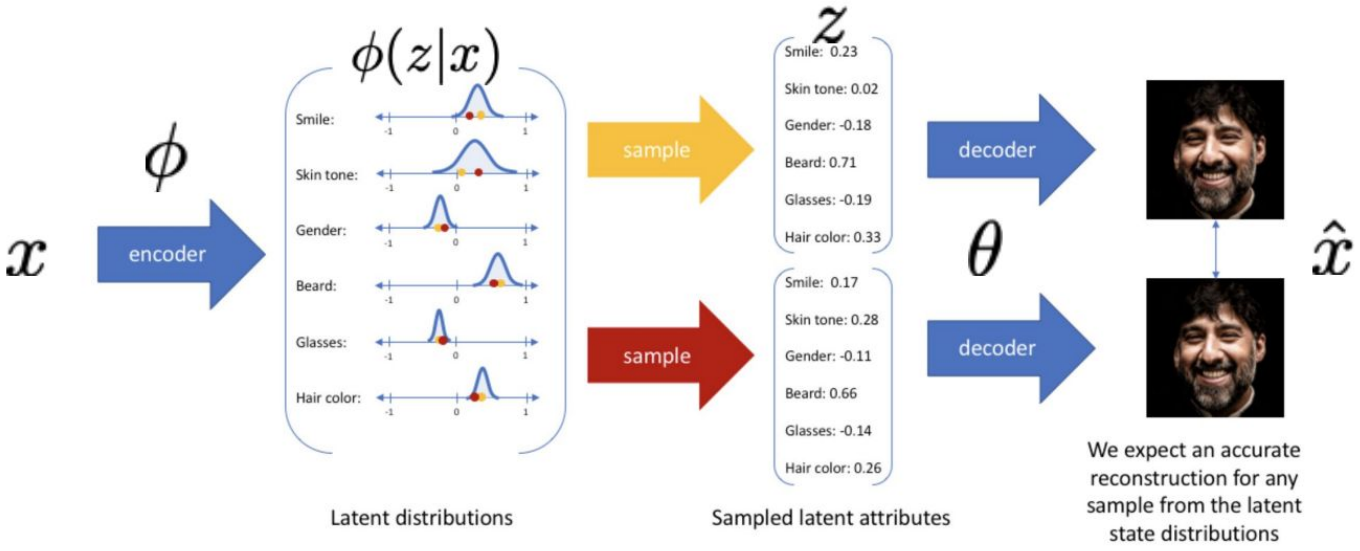
VAE: General Idea

Map the complicated data distribution to a simpler distribution (encoder) we can sample from (Kingma & Welling 2014) to generate images (decoder)



Encode into Distributions

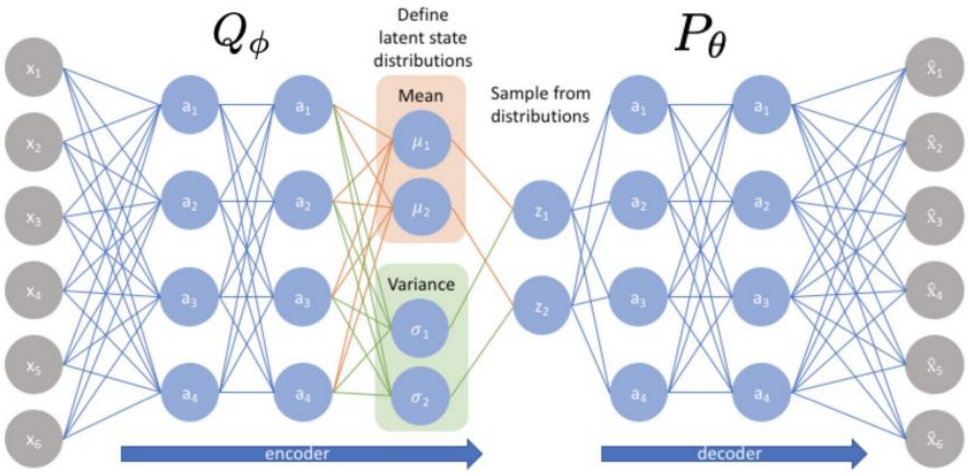
Q: Why encode into distributions rather than deterministic values?
A: This creates balls in latent space and ensures that close points in latent space lead to the same reconstruction and we expect to give more “meaning” to the latent space.



Impose Structure

Q : How can I bring the balls together to compactify latent space ?

A : Make sure that **q**(z|x) for different x's are close together !



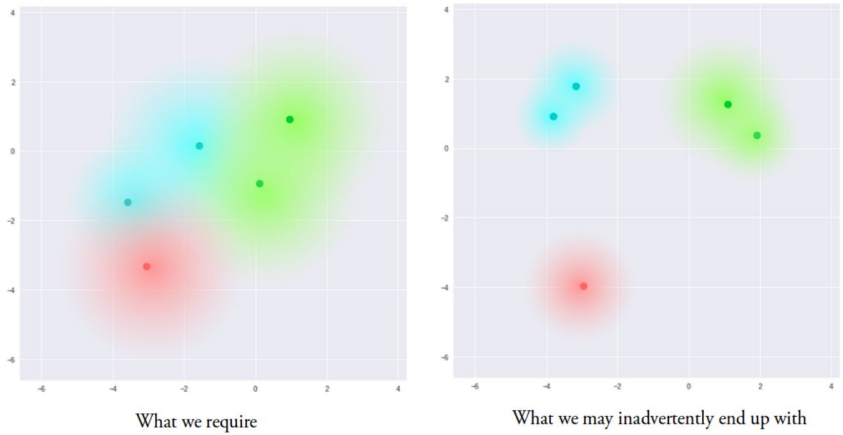
Impose Structure

Q : How do we keep the balls close together?

A : By adding springs the balls which pull them towards the center

Q : How?

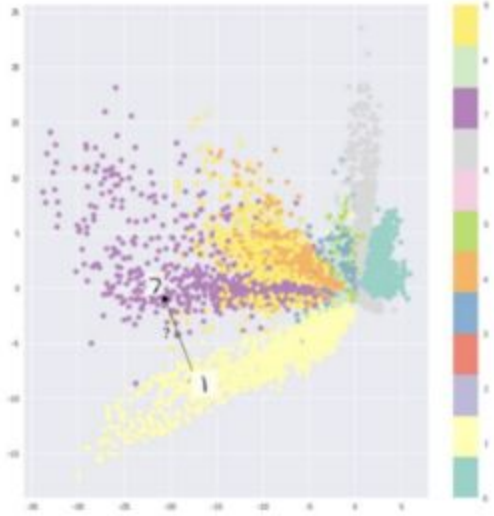
A : KL divergence with N(0,1) prior. $\mathcal{L}_{KL} = \mathbb{E}_{x \sim dataset} \left[D_{KL} \{ Q_\phi(z|x) || p(z) \} \right], p \sim \mathcal{N}(0,1)$



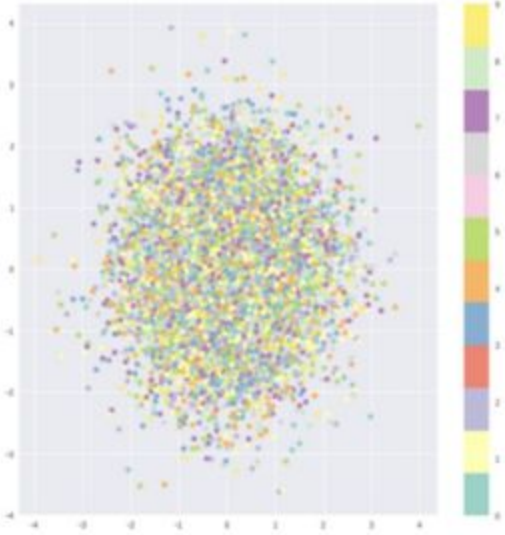
What we require

What we may inadvertently end up with

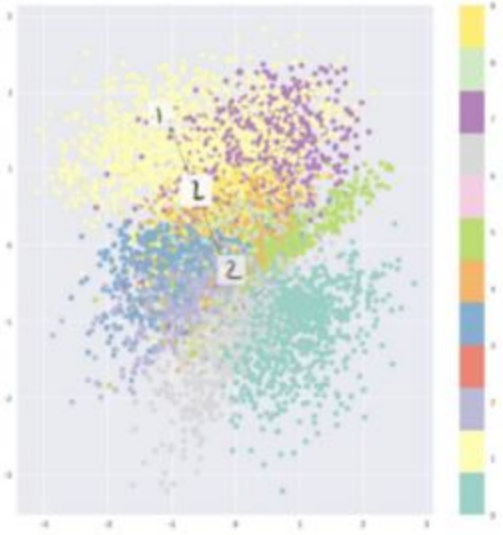
Only reconstruction loss



Only KL divergence

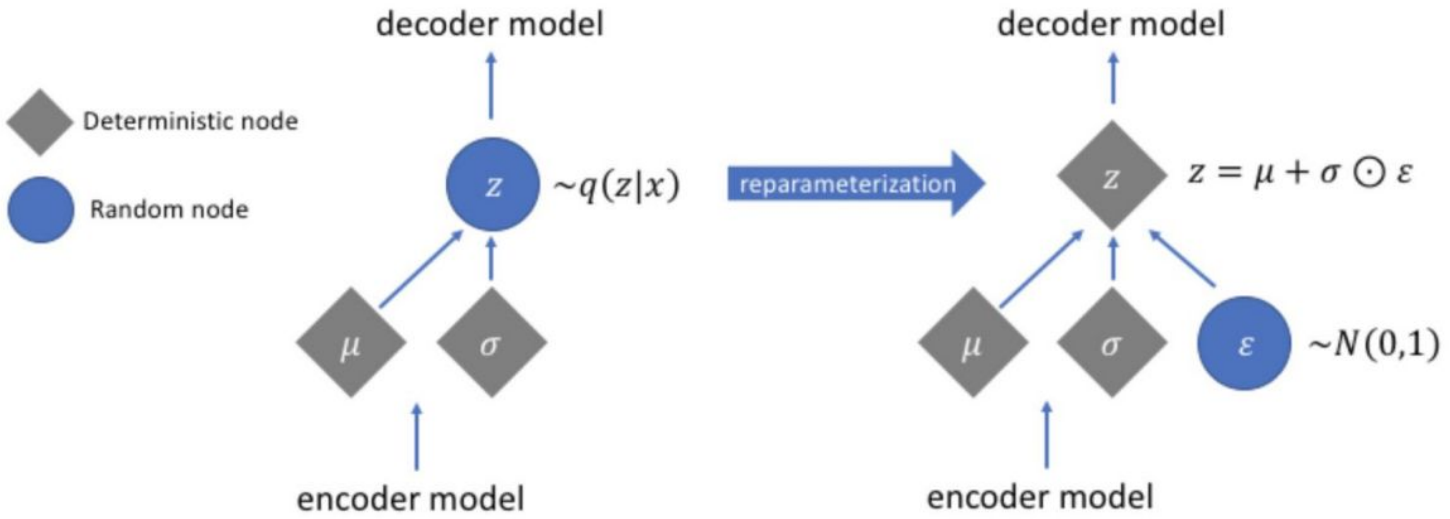


Combination



The Reparameterization Trick

Q: How can we backpropagate when one of the nodes is non-deterministic?
A: Put the random process outside the network



Variational inference

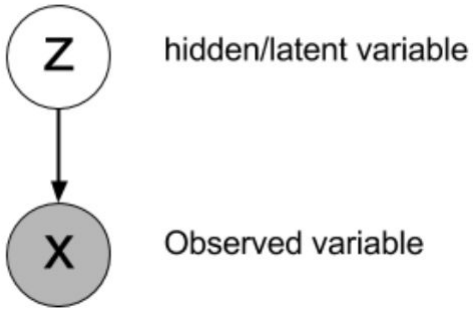
- Latent variable model:

likelihood of the observation X given Z

prior

$$p(X, Z) = p(Z)p(X|Z)$$

- **X** – observed data, **Z** – latent variable



Variational inference

- Variational inference problem:

posterior distribution

$$p(Z|X) = \frac{p(X,Z)}{p(X)}$$

evidence

- Computing marginal distribution of the data:

$$p(X) = \int p(X,Z)dz$$

- Intractable in many cases

Approximate posterior

- Goal of variational inference: find approximate posterior distribution similar to the true posterior distribution
- KL divergence measures how two distributions diverge
  - Discrete:

$$\mathrm{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

  - Continuous:

$$\mathrm{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

- Minimize the KL divergence between the approximate posterior distribution Q(Z) and the true posterior distribution P(Z)

Approximate posterior

- From a family of distributions τ find a distribution q(Z) that minimizes KL divergence with the true posterior distribution p(Z|X):

$$q^*(Z) = \underset{q(Z)\in\tau}{\operatorname{argmin}} \quad \mathrm{KL}(q(Z)||p(Z|X)) \qquad [1]$$

- Can’t optimize directly because of p(Z|X)

- Rewrite Eq. [1] as:

$$\begin{aligned}\mathrm{KL}(q(Z)||p(Z|X)) &= \mathbb{E}_{q(Z)}\left[\log \frac{q(Z)}{p(Z|X)}\right] \\ &= \mathbb{E}_{q(Z)}\left[\log q(Z)\right] - \mathbb{E}_{q(Z)}\left[\log p(Z|X)\right] \\ &= \mathbb{E}_{q(Z)}\left[\log q(Z)\right] - \mathbb{E}_{q(Z)}\left[\log \frac{p(Z,X)}{p(X)}\right] \\ &= \mathbb{E}_{q(Z)}\left[\log q(Z)\right] - \mathbb{E}_{q(Z)}\left[\log p(Z,X)\right] + \mathbb{E}_{q(Z)}\left[\log p(X)\right]\end{aligned} \quad [2]$$

- Rearrange:

Evidence Lower Bound (ELBO)

log evidence

$$\left\{\mathbb{E}_{q(Z)}\left[\log p(Z,X)\right] - \mathbb{E}_{q(Z)}\left[\log q(Z)\right]\right\} + \mathrm{KL}(q(Z)||p(Z|X)) = \log p(X) \quad [3]$$

Evidence Lower Bound (ELBO)

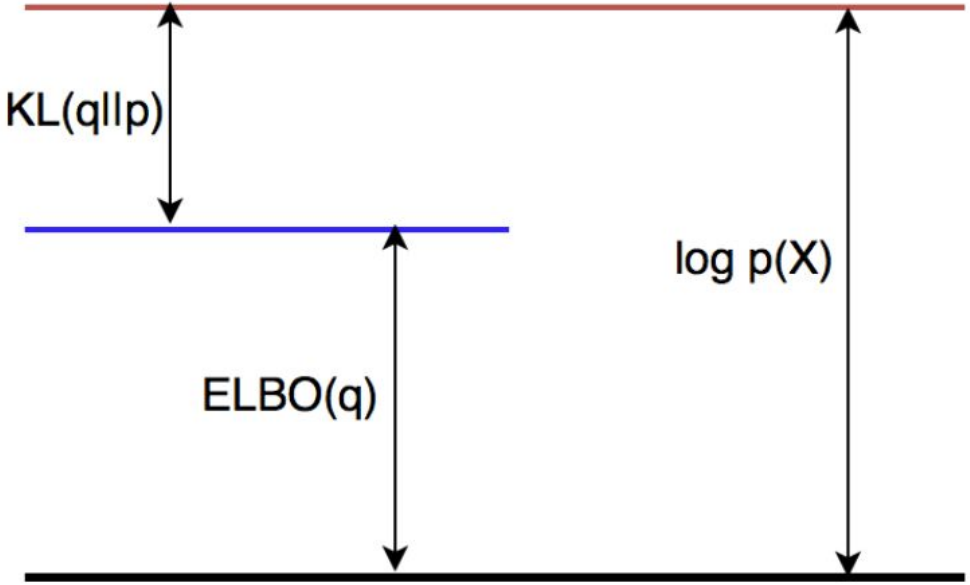


Image source: (Bahuleyan, 2018)

Evidence Lower Bound (ELBO)

- ELBO can be re-written as:

$$
\begin{aligned}
\mathrm{ELBO}(q) &= \mathbb{E}_{q(Z)}\left[\log p(Z,X)\right] - \mathbb{E}_{q(Z)}\left[\log q(Z)\right] \\
&= \mathbb{E}_{q(Z)}\left[\log p(X|Z)p(Z)\right] - \mathbb{E}_{q(Z)}\left[\log q(Z)\right] \\
&= \mathbb{E}_{q(Z)}\left[\log p(X|Z)\right] + \mathbb{E}_{q(Z)}\left[\log p(Z)\right] - \mathbb{E}_{q(Z)}\left[\log q(Z)\right] \\
&= \mathbb{E}_{q(Z)}\left[\log p(X|Z)\right] + \mathbb{E}_{q(Z)}\left[\log \frac{p(Z)}{q(Z)}\right] \\
&= \mathbb{E}_{q(Z)}\left[\log p(X|Z)\right] - \mathrm{KL}(q(Z)||p(Z))
\end{aligned}
$$

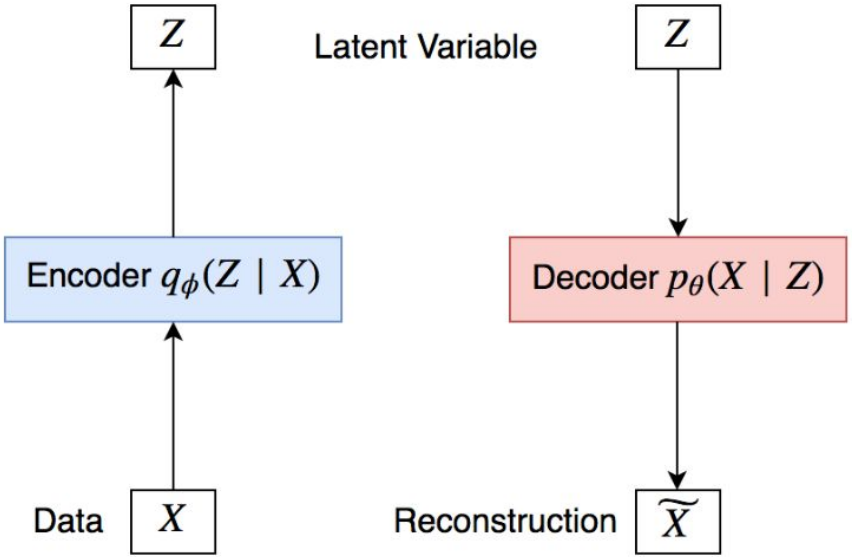
[4]

Expected log likelihood of the data

KL divergence between the approximate posterior and the prior

- Maximize expected log likelihood of data and minimize KL divergence

Variational Autoencoder



Variational Autoencoder

- Replace approximate posterior q(Z) with $q_{\varphi}(Z|X)$
- Given a dataset: $\mathcal{D} = \{x^{(n)}\}_{n=1}^{N}$
- The likelihood of a data point (log evidence) and ELBO are related as follows:

$$\log p_{\boldsymbol{\theta}}(x^{(n)}) \geq \mathbb{E}_{z \sim q_{\boldsymbol{\phi}}(z|x^{(n)})} \left[\log \left\{ \frac{p_{\boldsymbol{\theta}}(x^{(n)}, z)}{q_{\boldsymbol{\phi}}(z|x^{(n)})} \right\} \right]$$

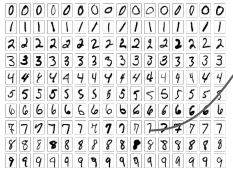
$$= \mathbb{E}_{z \sim q_{\boldsymbol{\phi}}(z|x^{(n)})} \left[\log p_{\boldsymbol{\theta}}(x^{(n)}|z) \right] - \mathrm{KL} \left(q_{\boldsymbol{\phi}}(z|x^{(n)}) || p(z) \right) \quad [5]$$

Variational Autoencoder

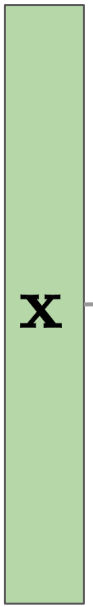
- Maximizing ELBO(q) is equivalent to minimizing –ELBO(q)
- Loss function for the VAE:

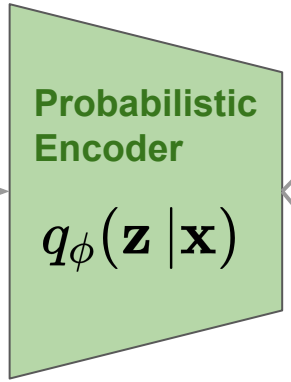
$$
\begin{aligned}
J^{(n)} &= -\mathbb{E}_{z\sim q_{\boldsymbol{\phi}}(z|x^{(n)})}\left[\log p_{\boldsymbol{\theta}}(x^{(n)}|z)\right] + \mathrm{KL}\left(q_{\boldsymbol{\phi}}(z|x^{(n)})\|p(z)\right) \\
&= J_{\mathrm{rec}}(\boldsymbol{\theta}, \boldsymbol{\phi}, x^{(n)}) + \mathrm{KL}\big(q_{\boldsymbol{\phi}}(z|x^{(n)})\|p(z)\big)
\end{aligned}
$$

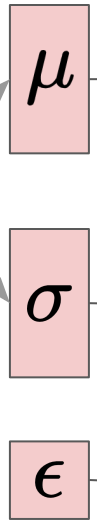
- Reconstruction loss: expected negative log likelihood of the data
- Prior is typically Gaussian $N(0,I)$

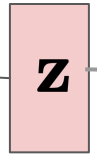


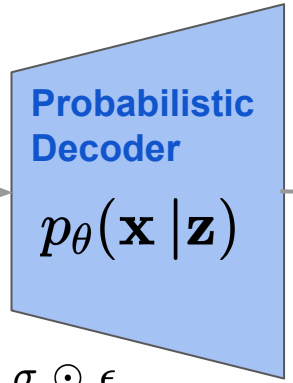
$$\mathcal{L}(x;\theta,\phi) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction term}} - \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{prior term}}$$











$$\mathbf{z} = \mu + \sigma \odot \epsilon$$

VAE Extension: β-VAE

- Simple modification to VAE framework

$$\mathcal{L}(x;\theta,\phi) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction term}} - \beta \underbrace{D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{prior term}}$$

- Encourages disentangled representation in latent code with higher beta at the expense of reconstruction quality

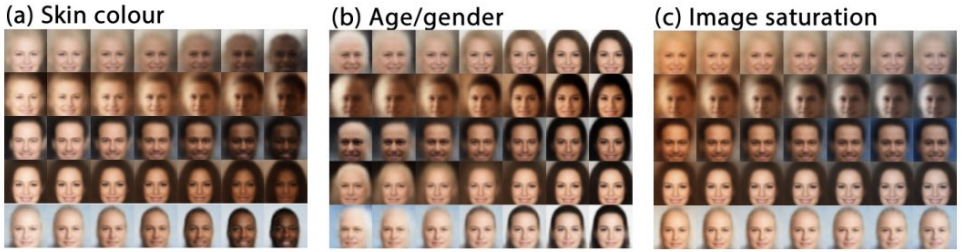
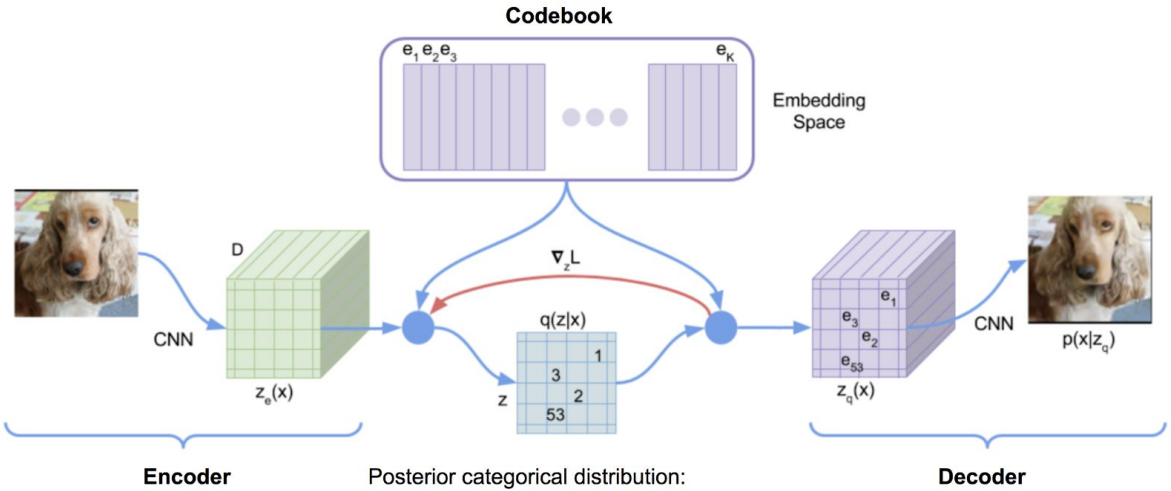


Figure 4: **Latent factors learnt by β-VAE on celebA:** traversal of individual latents demonstrates that β-VAE discovered in an unsupervised manner factors that encode skin colour, transition from an elderly male to younger female, and image saturation.

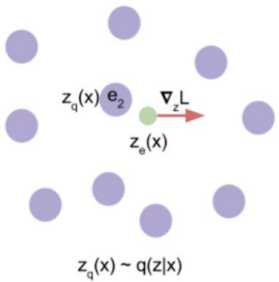
Vector Quantized VAE

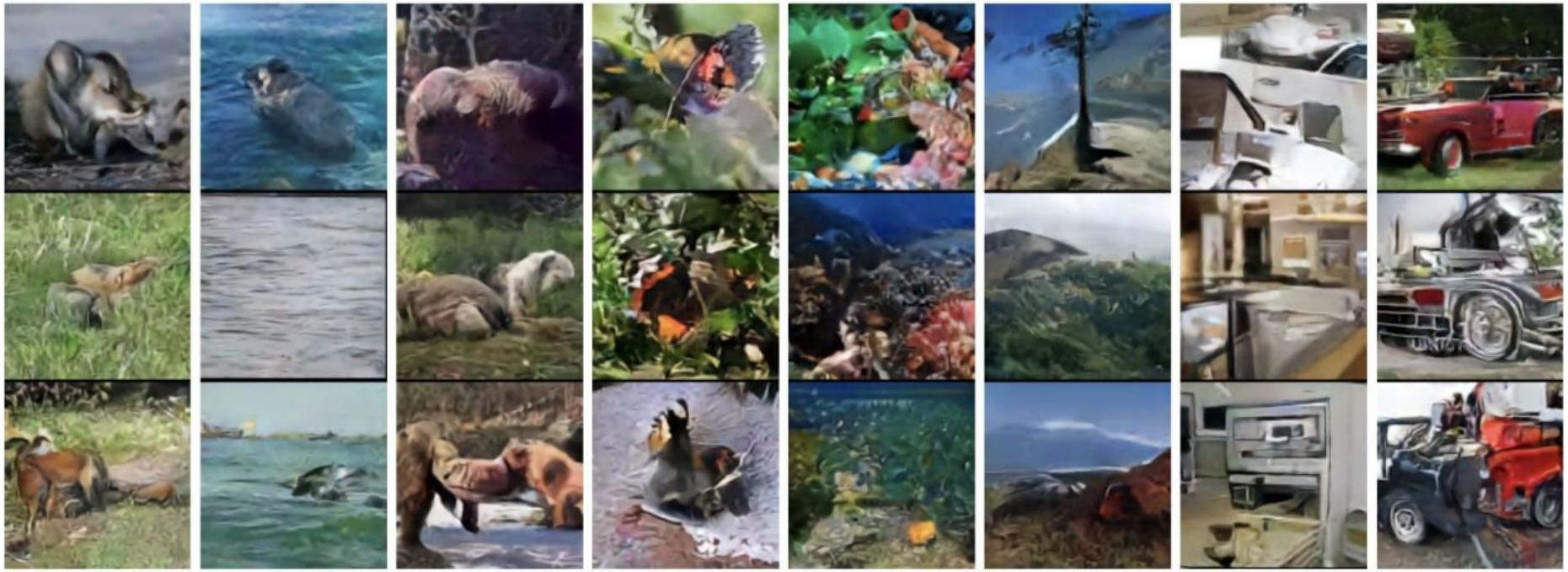
VQ-VAE

- Discrete latent code
- Autoregressive decoder
- Train prior over latent code can be added (VQ-VAE-2)



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





VQ-VAE high quality samples

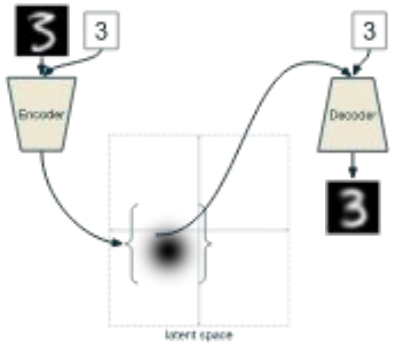
Conditional VAE

If we have labels, we can learn to generate data *given* a label to hopefully improve or guide our generation.

Guide the generator towards more familiar samples.

Implementation: concatenate labels to dense layers in the encoder (or latent code), do the same for the decoder.

Training time



Inference time