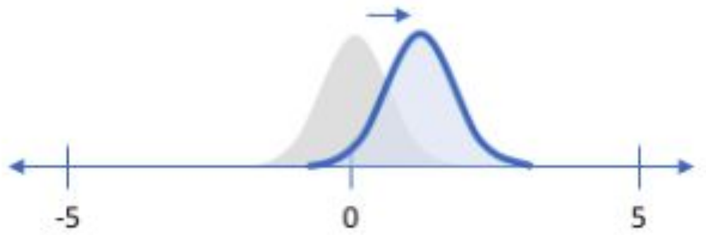
So, how does it work?

Encode

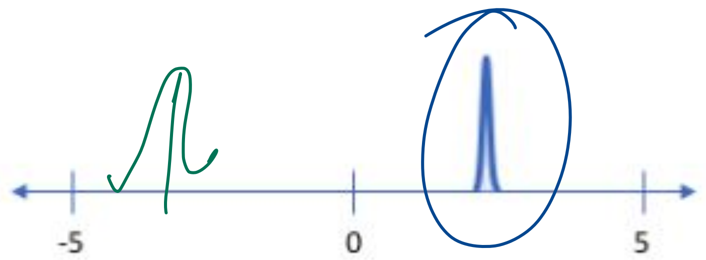
Decode

Penalizing reconstruction loss encourages the distribution to describe the input



Our distribution deviates from the prior to describe some characteristic of the data

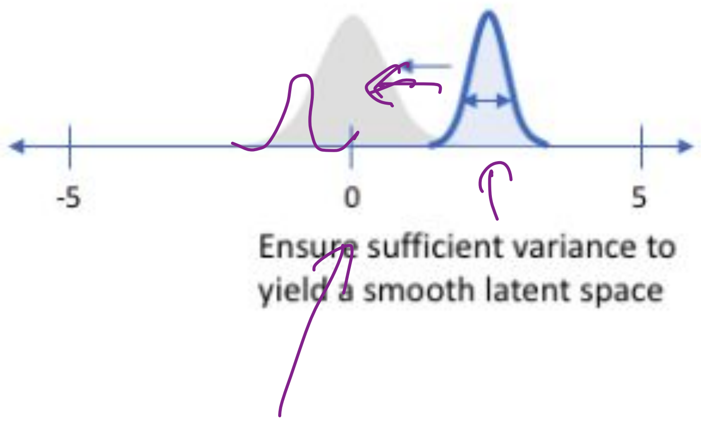
Without regularization, our network can "cheat" by learning narrow distributions



With a small enough variance, this distribution is effectively only representing a single value

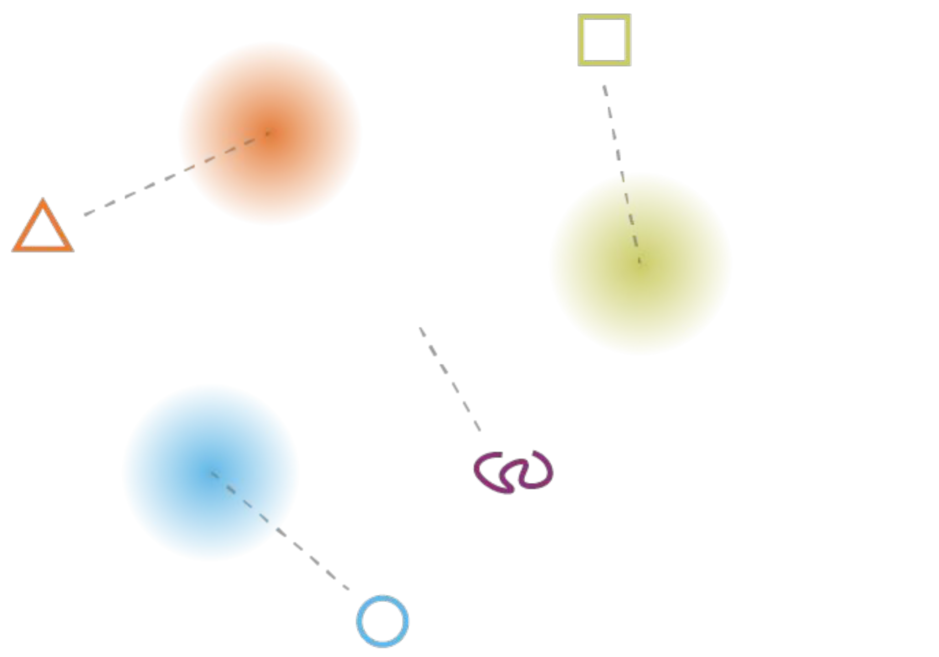
Penalizing KL divergence acts as a regularizing force

Attract distribution to have zero mean

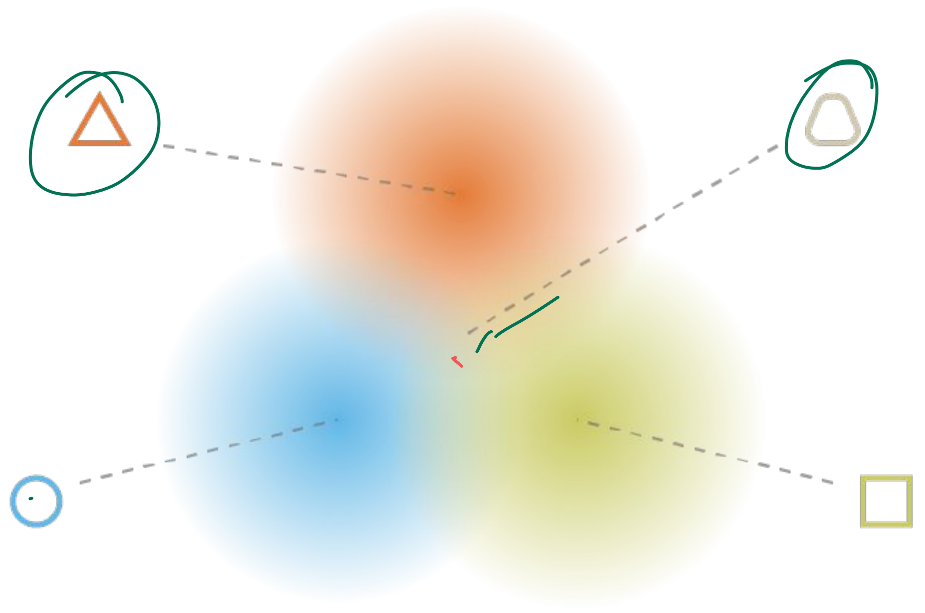


Ensure sufficient variance to yield a smooth latent space

How does KL divergence help?

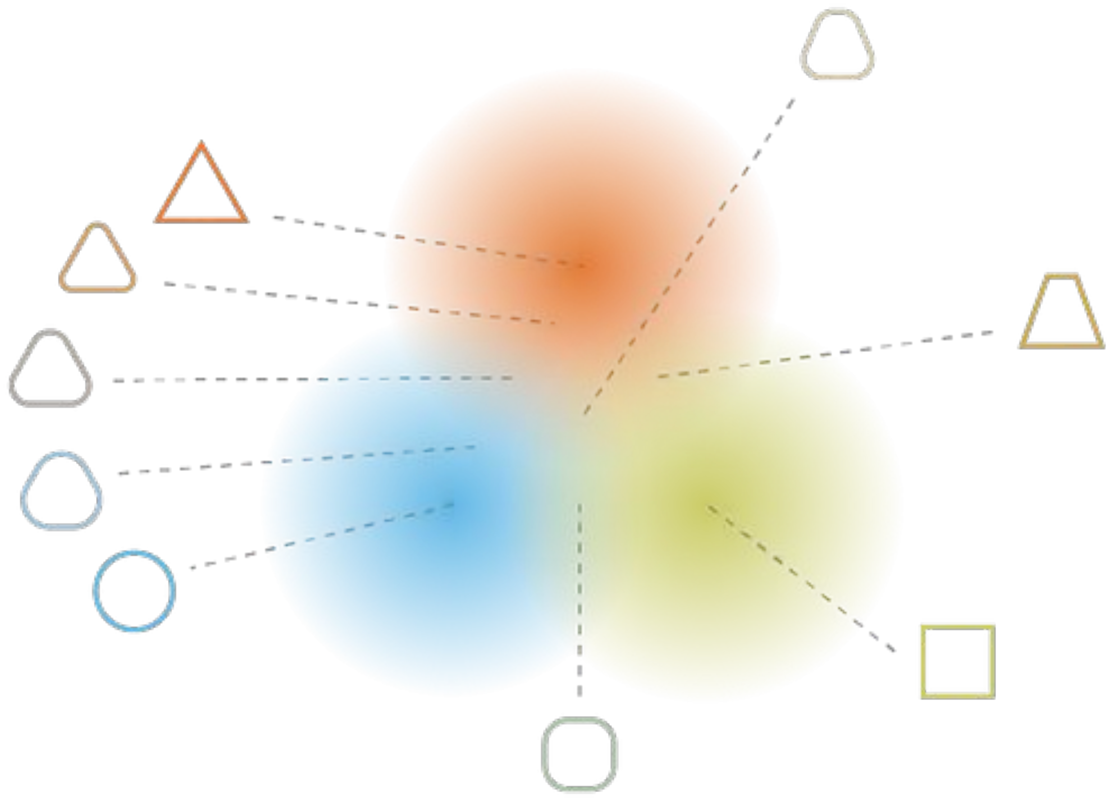


what can happen without regularisation ✗



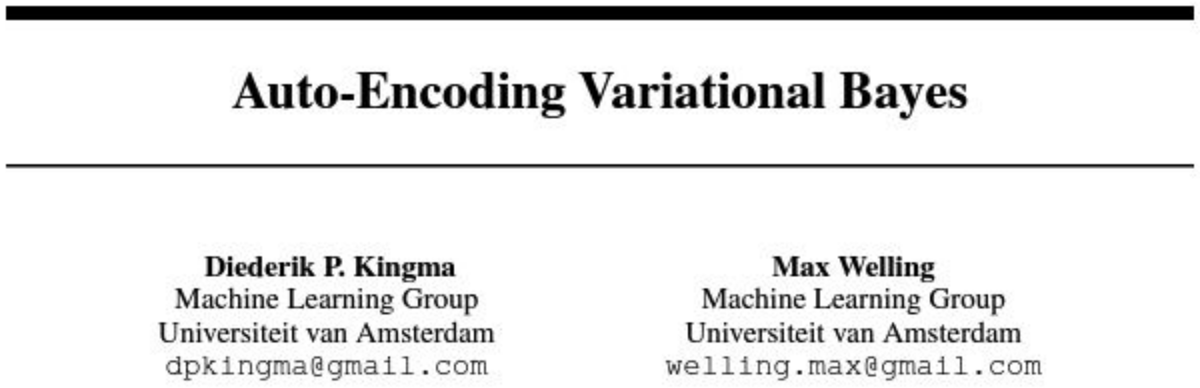
✓ what we want to obtain with regularisation

Continuity and completeness



A point of the latent space that would be halfway between the means of two encoded distributions coming from different training data should be decoded in something that is somewhere between the data that gave the first distribution and the data that gave the second distribution as it may be sampled by the autoencoder in both cases.

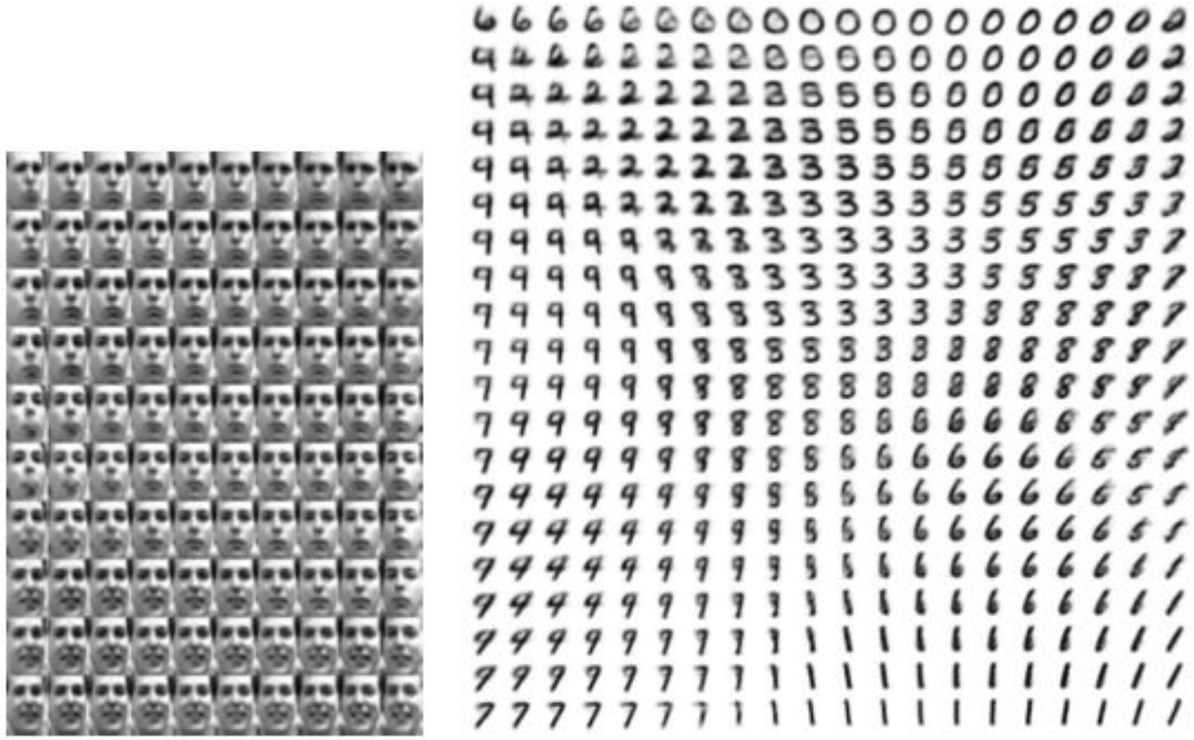
Visualization of the learned data manifold



Auto-Encoding Variational Bayes

Diederik P. Kingma
Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling
Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com



(a) Learned Frey Face manifold (b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Maths behind VAEs

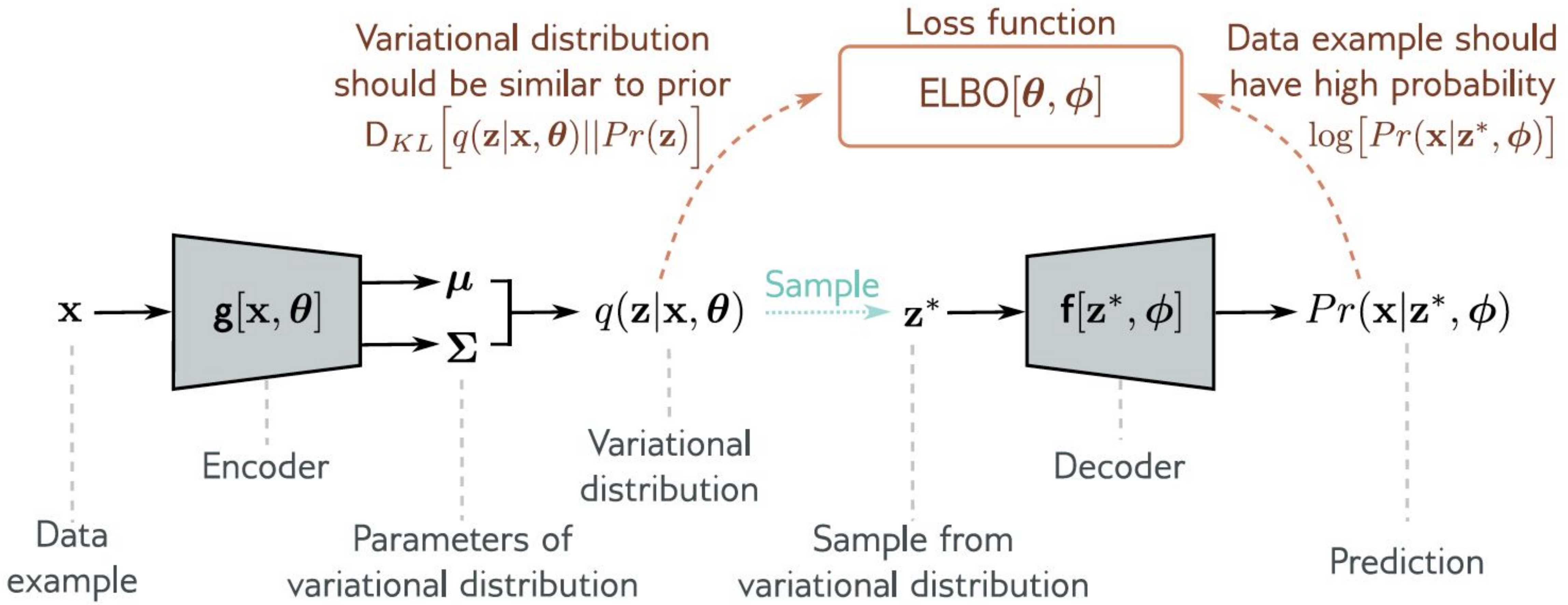
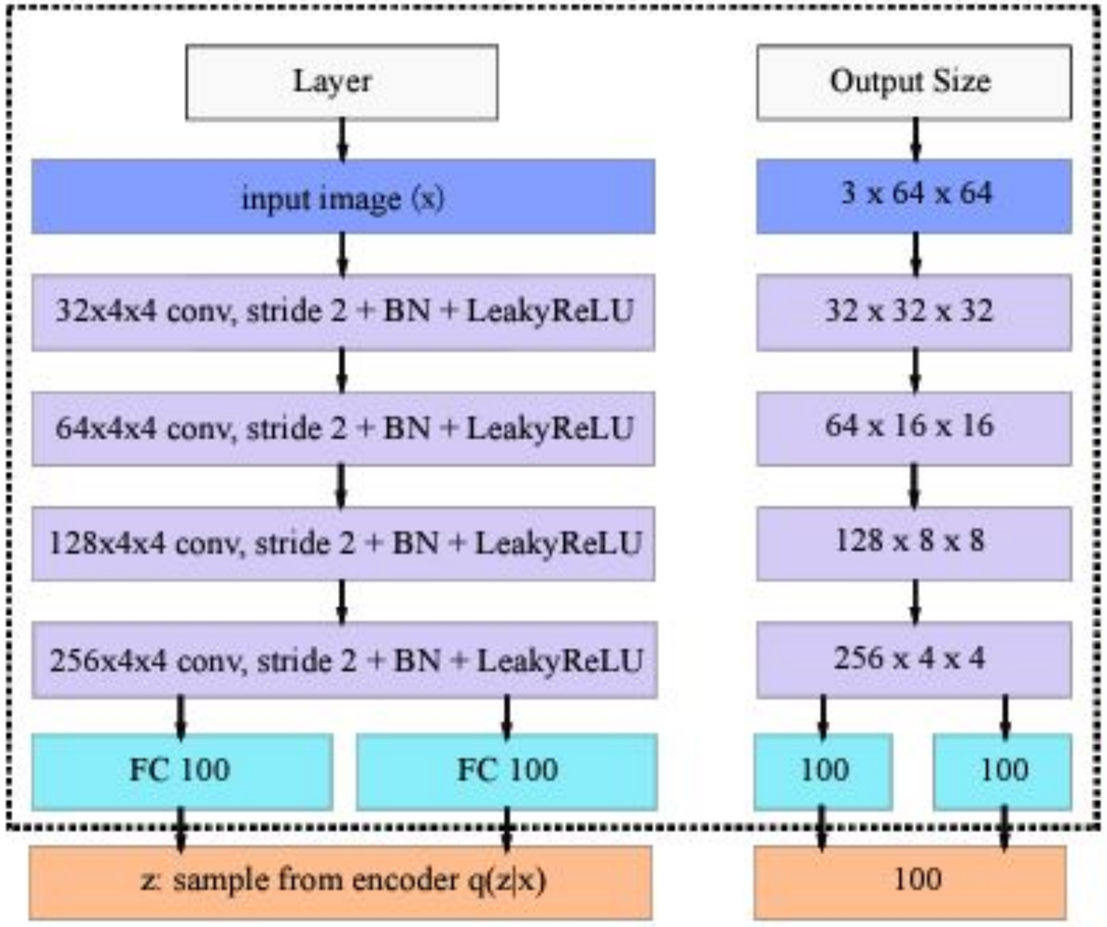
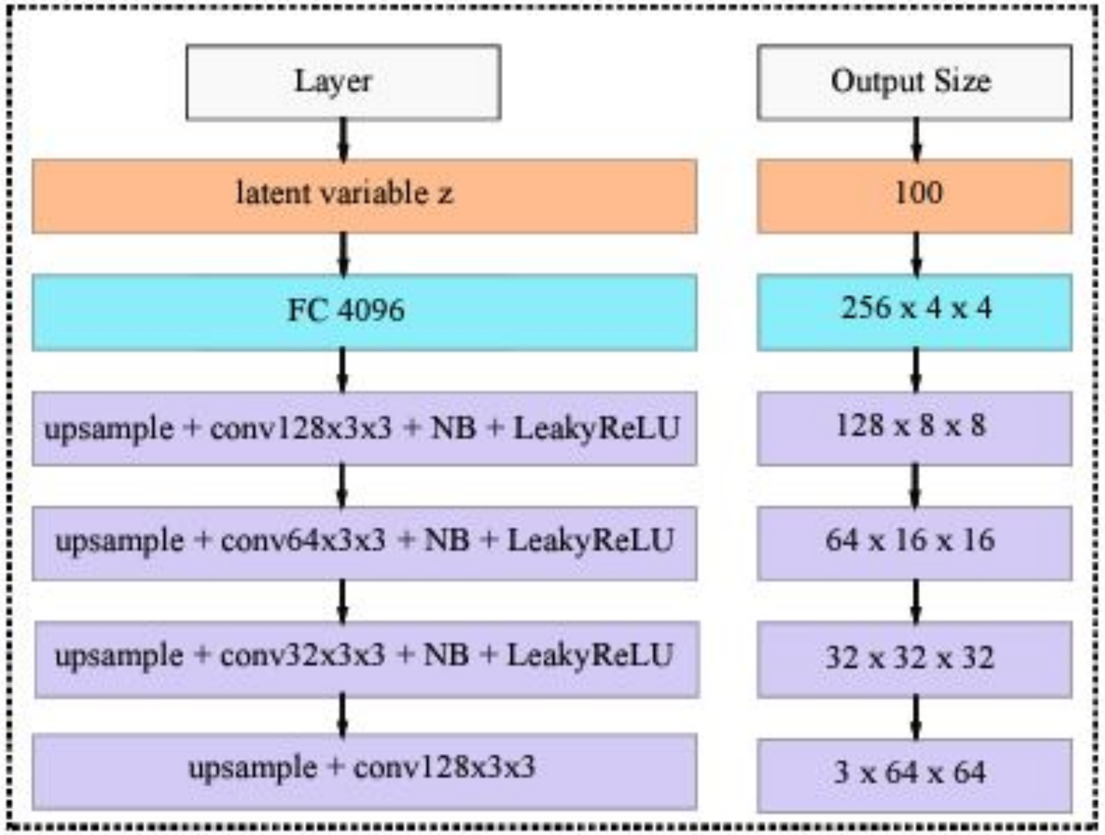

Loss function
$\mathrm{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}]$
Variational distribution should be similar to prior
$\mathrm{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})||Pr(\mathbf{z})\Big]$
Data example should have high probability
$\log\big[Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})\big]$
$\mathbf{x}$
$\mathbf{g}[\mathbf{x}, \boldsymbol{\theta}]$
$\boldsymbol{\mu}$
$\boldsymbol{\Sigma}$
$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$
Sample
$\mathbf{z}^*$
$\mathbf{f}[\mathbf{z}^*, \boldsymbol{\phi}]$
$Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})$
Data example
Encoder
Parameters of variational distribution
Variational distribution
Sample from variational distribution
Decoder
Prediction

Figure 17.9 Variational autoencoder. The encoder $\mathbf{g}[\mathbf{x}, \boldsymbol{\theta}]$ takes a training example $\mathbf{x}$ and predicts the parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ of the variational distribution $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$. We sample from this distribution and then use the decoder $\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}]$ to predict the data $\mathbf{x}$. The loss function is the negative ELBO, which depends on how accurate this prediction is and how similar the variational distribution $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$ is to the prior $Pr(\mathbf{z})$ (equation 17.21).

Using Deep Networks


Layer
Output Size
input image (x)
3 x 64 x 64
32x4x4 conv, stride 2 + BN + LeakyReLU
32 x 32 x 32
64x4x4 conv, stride 2 + BN + LeakyReLU
64 x 16 x 16
128x4x4 conv, stride 2 + BN + LeakyReLU
128 x 8 x 8
256x4x4 conv, stride 2 + BN + LeakyReLU
256 x 4 x 4
FC 100
FC 100
100
100
z: sample from encoder q(z|x)
100



Deriving the bound

Objective: maximize log [*Pr(x|ϕ)*]

Which can be written by marginalizing over z

$$\log[Pr(\mathbf{x}|\phi)] \quad = \quad \log\left[\int Pr(\mathbf{x},\mathbf{z}|\phi)d\mathbf{z}\right]$$

We now multiply and divide the log-likelihood by an arbitrary probability distribution q(z) over the latent variables:

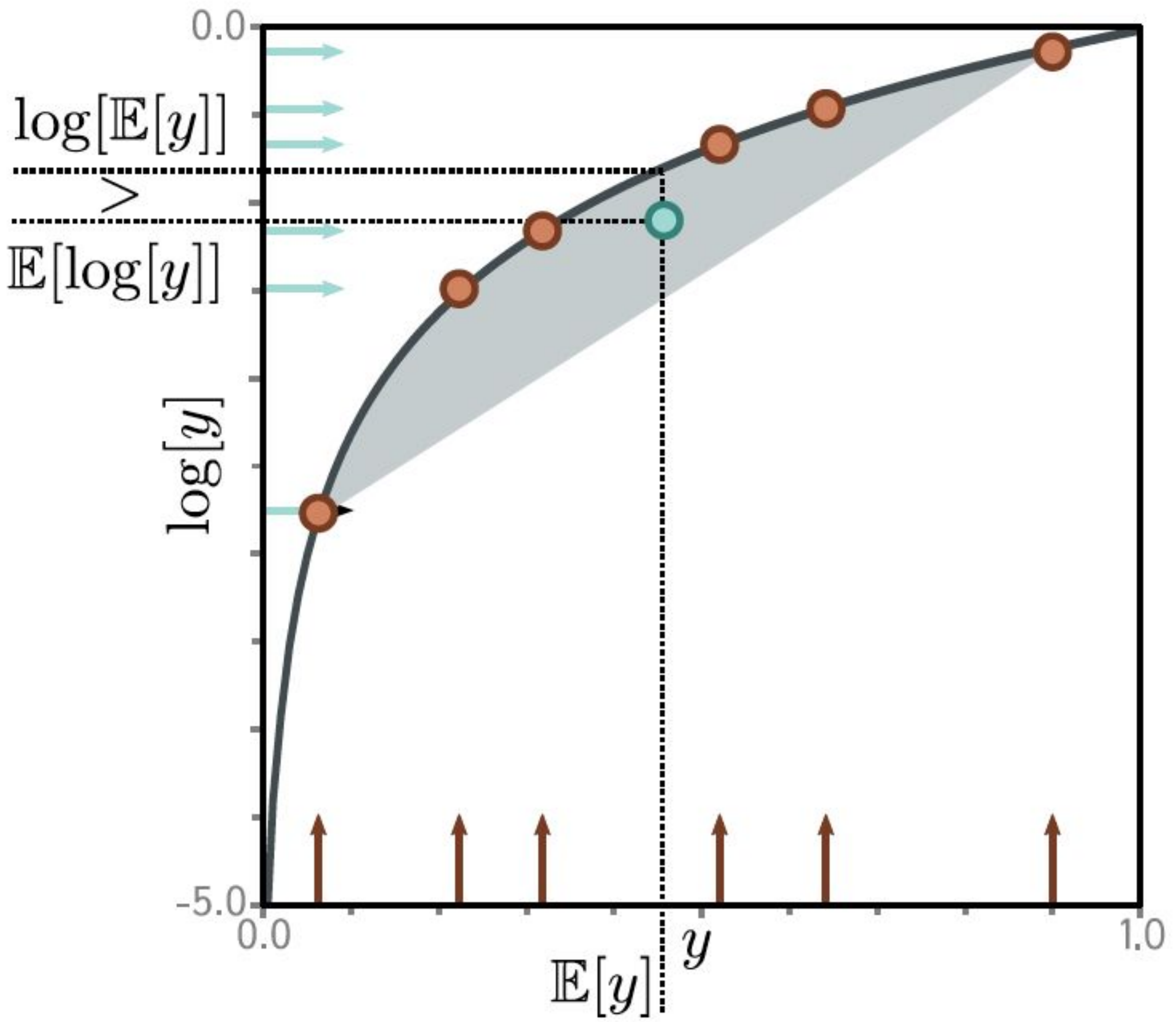
$$= \quad \log\left[\int q(\mathbf{z})\frac{Pr(\mathbf{x},\mathbf{z}|\phi)}{q(\mathbf{z})}d\mathbf{z}\right]$$

Jensen’s inequality says that a concave function g[•] of the expectation of data y is greater than or equal to the expectation of the function of the data

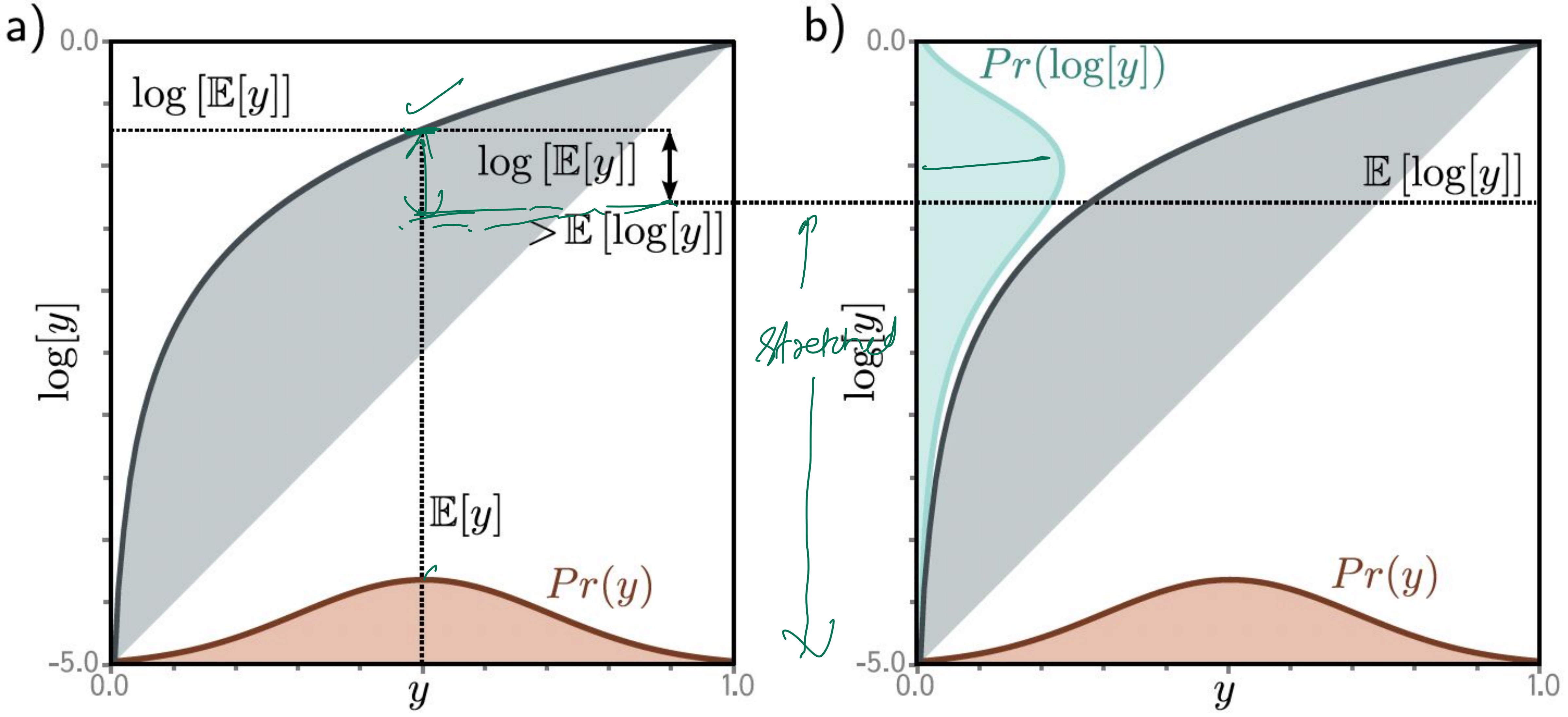
$$\log\left[\int Pr(y)y dy\right] \geq \int Pr(y)\log[y]dy.$$

Graphical Intuition: Jensen’s inequality

Figure 17.4 Jensen’s inequality (discrete case). The logarithm (black curve) is a concave function; you can draw a straight line between any two points on the curve, and this line will always lie underneath it. It follows that any convex combination (weighted sum with positive weights that sum to one) of the six points on the log function must lie in the gray region under the curve. Here, we have weighted the points equally (i.e., taken the mean) to yield the cyan point. Since this point lies below the curve, $\log[\mathbb{E}[y]] > \mathbb{E}[\log[y]]$.



Graphical Intuition: Jensen's inequality



Deriving the bound

Jensen's inequality says that a concave function g[•] of the expectation of data y is greater than or equal to the expectation of the function of the data

$$\boxed{\log\left[\int Pr(y)y\,dy\right] \geq \int Pr(y)\log[y]\,dy.}$$ ✓

In fact, the slightly more general statement is true:

$$\log\left[\int Pr(y)\underbrace{h[y]}\,dy\right] \geq \int Pr(y)\underbrace{\log[h[y]]}\,dy.$$

where h[y] is a function of y. This follows because h[y] is another random variable with a new distribution. Since we never specified Pr(y), the relation remains true.

Deriving the bound

Using Jensen’s inequality

$$\log\left[\int q(\mathbf{z})\frac{Pr(\mathbf{x},\mathbf{z}|\phi)}{q(\mathbf{z})}d\mathbf{z}\right] \geq \int \underline{q}(\mathbf{z})\log\left[\frac{Pr(\mathbf{x},\mathbf{z}|\phi)}{q(\mathbf{z})}\right]d\mathbf{z}.$$

Evidence Lower Bound (ELBO)

In practice, the distribution *q(z)* has parameters *θ*, so the ELBO can be written as:

$$\mathrm{ELBO}[\theta,\phi] = \int \underline{q(\mathbf{z}|\theta)}\log\underline{\left[\frac{Pr(\mathbf{x},\mathbf{z}|\phi)}{q(\mathbf{z}|\theta)}\right]}d\mathbf{z}.$$

ELBO as reconstruction loss - KL divergence

$$\begin{aligned}
\mathrm{ELBO}[\boldsymbol{\theta},\phi] &= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{x},\mathbf{z}|\phi)}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{x}|\mathbf{z},\phi)Pr(\mathbf{z})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[Pr(\mathbf{x}|\mathbf{z},\phi)\right] d\mathbf{z} + \int q(\mathbf{z}|\boldsymbol{\theta}) \log\left[\frac{Pr(\mathbf{z})}{q(\mathbf{z}|\boldsymbol{\theta})}\right] d\mathbf{z} \\
&= \int q(\mathbf{z}|\boldsymbol{\theta}) \log\Big[Pr(\mathbf{x}|\mathbf{z},\phi)\Big] d\mathbf{z} - \mathrm{D}_{KL}\Big[q(\mathbf{z}|\boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big],
\end{aligned}$$

For our network: *q(z|θ) = q(z|x, θ)*

$$\mathrm{ELBO}[\boldsymbol{\theta},\phi] = \int q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta}) \log\Big[Pr(\mathbf{x}|\mathbf{z},\phi)\Big] d\mathbf{z} - \mathrm{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big]$$

Approximating ELBO

$$\mathrm{ELBO}[\boldsymbol{\theta},\phi] = \int q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})\log\left[Pr(\mathbf{x}|\mathbf{z},\phi)\right]d\mathbf{z} - \mathrm{D}_{KL}\left[q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})\middle|\middle|Pr(\mathbf{z})\right]$$

Still intractable

Can you think of a trick to approximate the first term?

$$\mathbb{E}_{\mathbf{z}}\left[\mathrm{a}[\mathbf{z}]\right] = \int \mathrm{a}[\mathbf{z}]q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})d\mathbf{z} \approx \frac{1}{N}\sum_{n=1}^{N}\mathrm{a}[\mathbf{z}_n^*]$$

Monte Carlo Markov Sampling

For a very approximate estimate, we can just use a single sample $\mathbf{z}^*$ from $q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$

$$\mathrm{ELBO}[\boldsymbol{\theta},\phi] \approx \log\left[Pr(\mathbf{x}|\mathbf{z}^*,\phi)\right] - \mathrm{D}_{KL}\left[q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})\middle|\middle|Pr(\mathbf{z})\right]$$

KL divergence can be calculated in closed form

Computing KL divergence

→ MSE ✓

$$\text{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] \approx \underline{\log\Big[Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})\Big]} - \text{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big]$$

The second term is the KL divergence between the variational distribution $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \text{Norm}_{\mathbf{z}}[\boldsymbol{\mu}, \boldsymbol{\Sigma}]$ and the prior $Pr(\mathbf{z}) = \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$. The KL divergence between two normal distributions can be calculated in closed form. For the special case where one distribution has parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and the other is a standard normal, it is given by:

$$\boxed{\text{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big] = \frac{1}{2}\Big(\text{Tr}[\boldsymbol{\Sigma}] + \boldsymbol{\mu}^T\boldsymbol{\mu} - D_{\mathbf{z}} - \log\Big[\text{det}[\boldsymbol{\Sigma}]\Big]\Big).} \tag{17.24}$$

where $D_{\mathbf{z}}$ is the dimensionality of the latent space.

VAE Algorithm

We aim to build a model that computes the evidence lower bound for a point $\boxed{x.}$
Then we use an optimization algorithm to maximize this lower bound over the dataset and hence improve the log-likelihood.
To compute the ELBO, we: ✓ ✓

- compute the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ of the variational posterior distribution $q(\mathbf{z}|\boldsymbol{\theta}, \mathbf{x})$ for this data point $\mathbf{x}$ using the network $\boxed{\mathbf{g}[\mathbf{x}, \boldsymbol{\theta}],}$ Encoder
- draw a sample $\mathbf{z}^*$ from this distribution, and
- compute the ELBO using equation 17.23.

$$\text{ELBO}[\boldsymbol{\theta}, \phi] \approx \log\Big[Pr(\mathbf{x}|\mathbf{z}^*, \phi)\Big] - \text{D}_{KL}\Big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\Big|\Big|Pr(\mathbf{z})\Big] \quad \leftarrow$$

To maximize this bound, we run mini-batches of samples through the network and update these parameters with an optimization algorithm such as SGD or Adam

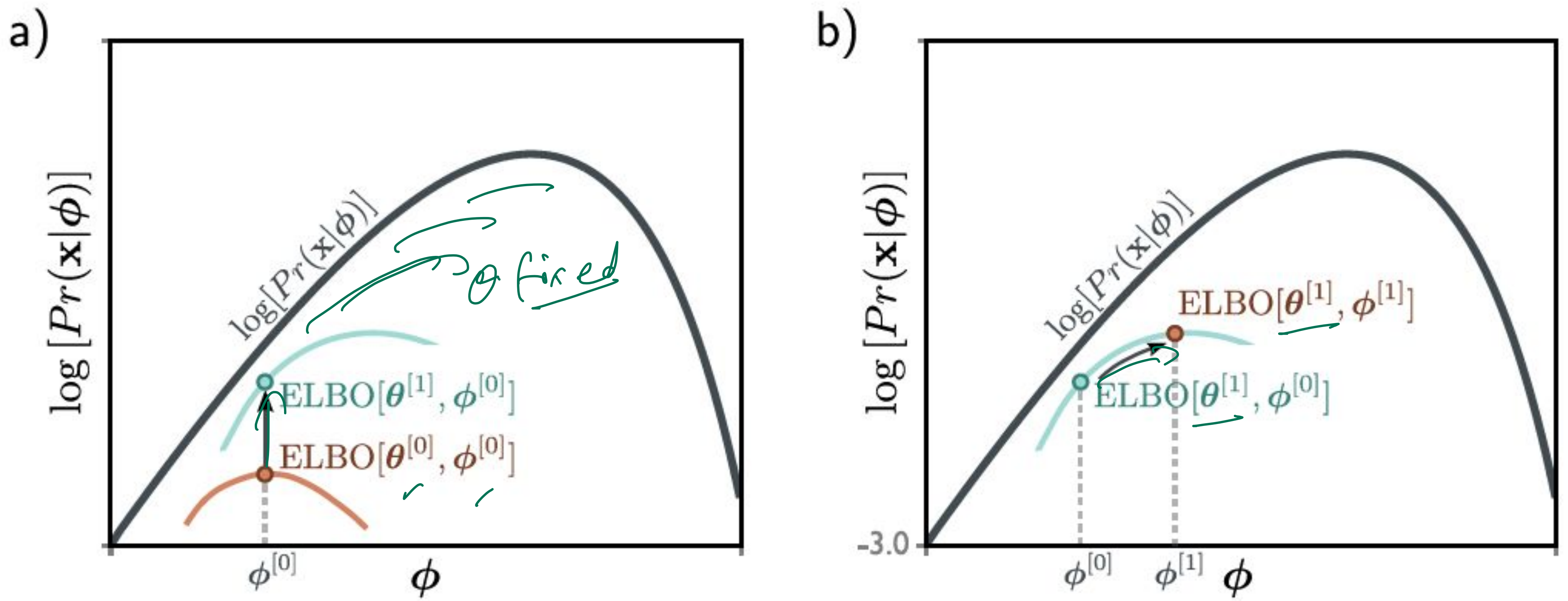
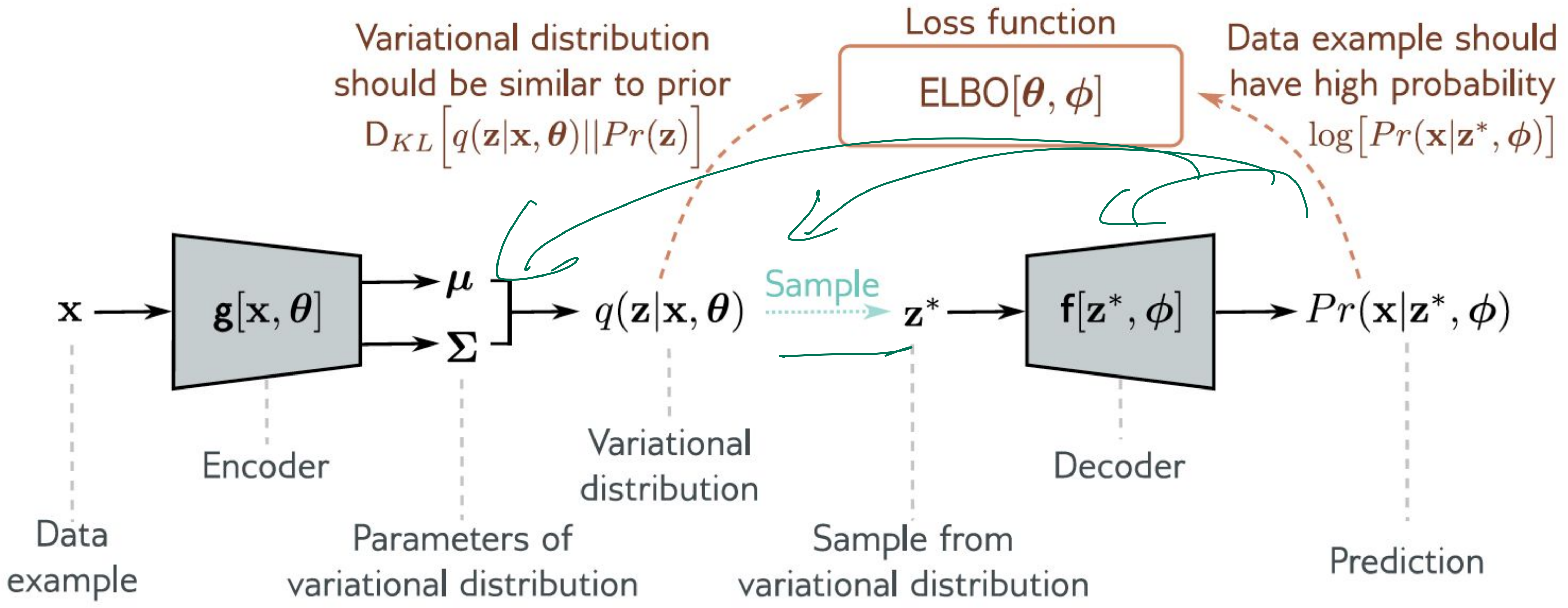
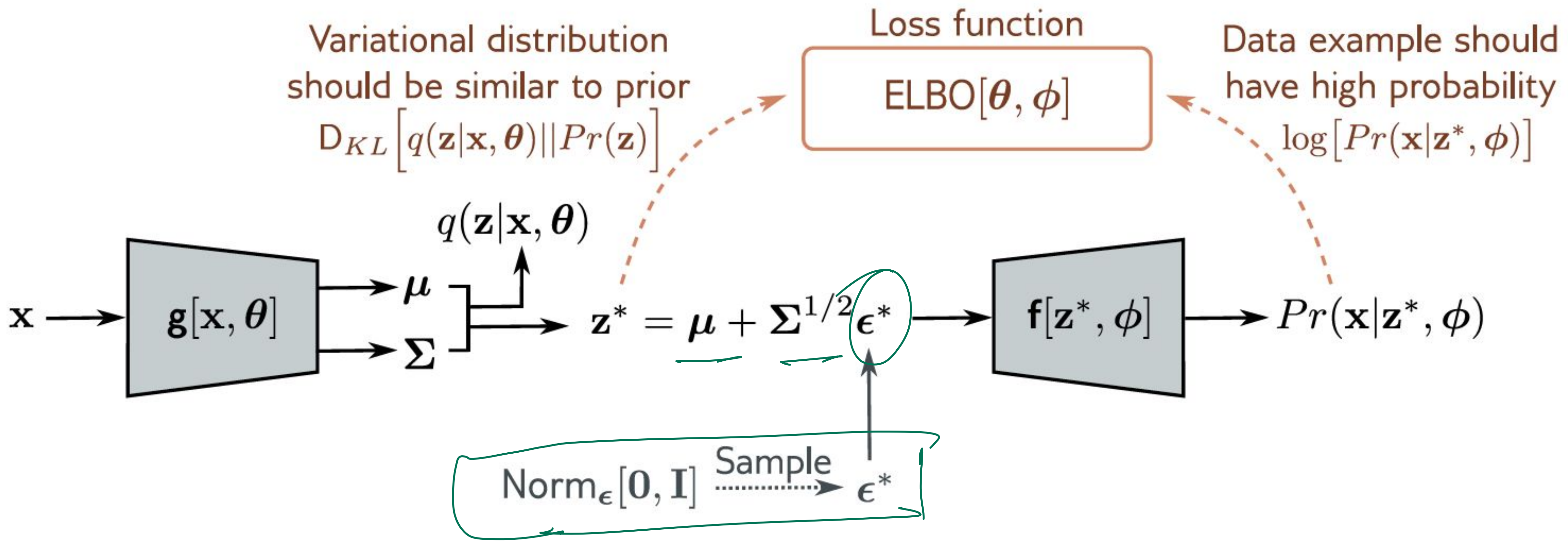


Figure 17.6 Evidence lower bound (ELBO). The goal is to maximize the log-likelihood $\log[Pr(\mathbf{x}|\boldsymbol{\phi})]$ (black curve) with respect to the parameters $\boldsymbol{\phi}$. The ELBO is a function that lies everywhere below the log-likelihood. It is a function of both $\boldsymbol{\phi}$ and a second set of parameters $\boldsymbol{\theta}$. For fixed $\boldsymbol{\theta}$, we get a function of $\boldsymbol{\phi}$ (two colored curves for different values of $\boldsymbol{\theta}$). Consequently, we can increase the log-likelihood by either improving the ELBO with respect to a) the new parameters $\boldsymbol{\theta}$ (moving from colored curve to colored curve) or b) the original parameters $\boldsymbol{\phi}$ (moving along the current colored curve).



Variational distribution should be similar to prior
$D_{KL}\left[q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})||Pr(\mathbf{z})\right]$
Loss function
$\text{ELBO}[\boldsymbol{\theta},\phi]$
Data example should have high probability
$\log\left[Pr(\mathbf{x}|\mathbf{z}^*,\phi)\right]$
$\mathbf{x}$
$\mathbf{g}[\mathbf{x},\boldsymbol{\theta}]$
$\boldsymbol{\mu}$
$\boldsymbol{\Sigma}$
$q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$
Sample
$\mathbf{z}^*$
$\mathbf{f}[\mathbf{z}^*,\phi]$
$Pr(\mathbf{x}|\mathbf{z}^*,\phi)$
Encoder
Variational distribution
Decoder
Data example
Parameters of variational distribution
Sample from variational distribution
Prediction

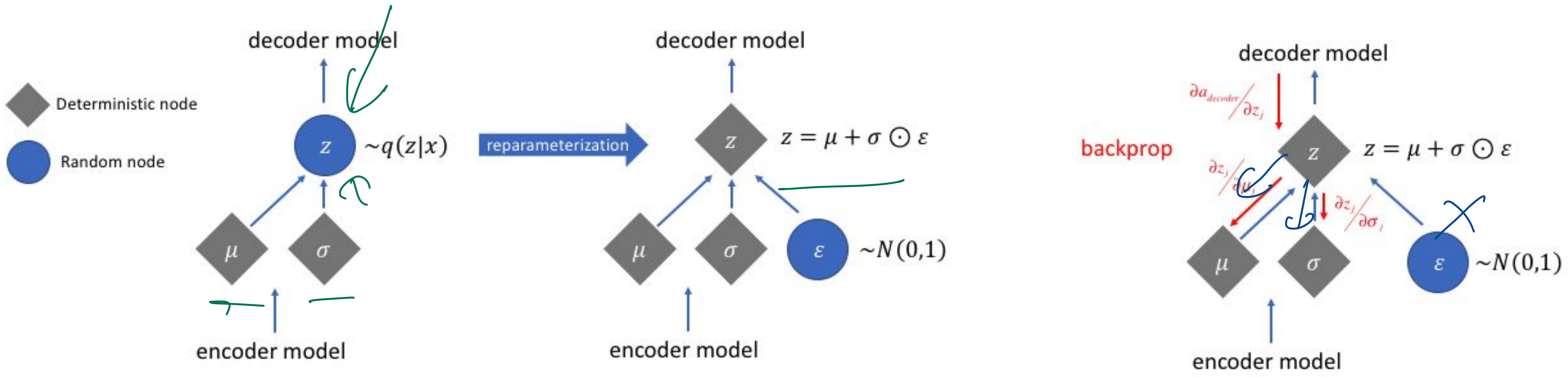
Can you see any problem with the optimization?



Variational distribution
should be similar to prior
$D_{KL}\left[q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})||Pr(\mathbf{z})\right]$
Loss function
$\text{ELBO}[\boldsymbol{\theta},\phi]$
Data example should
have high probability
$\log\left[Pr(\mathbf{x}|\mathbf{z}^*,\phi)\right]$
$q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$
$\mathbf{x}$
$\mathbf{g}[\mathbf{x},\boldsymbol{\theta}]$
$\boldsymbol{\mu}$
$\boldsymbol{\Sigma}$
$\mathbf{z}^* = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{\epsilon}^*$
$\mathbf{f}[\mathbf{z}^*,\phi]$
$Pr(\mathbf{x}|\mathbf{z}^*,\phi)$
$\text{Norm}_{\boldsymbol{\epsilon}}[\mathbf{0},\mathbf{I}]$
Sample
$\boldsymbol{\epsilon}^*$

Figure 17.11 Reparameterization trick. With the original architecture (figure 17.9), we cannot easily backpropagate through the sampling step. The reparameterization trick removes the sampling step from the main pipeline; we draw from a standard normal and combine this with the predicted mean and covariance to get a sample from the variational distribution.

The reparameterization trick



Application

A data point x can be projected into the latent space by taking the mean of the distribution predicted by the encoder

Multiple images labeled as “neutral” or “smiling” are projected into latent space. The vector representing this change is estimated by taking the difference in latent space between the means of these two groups. A second vector is estimated to represent “mouth closed” versus “mouth open.”

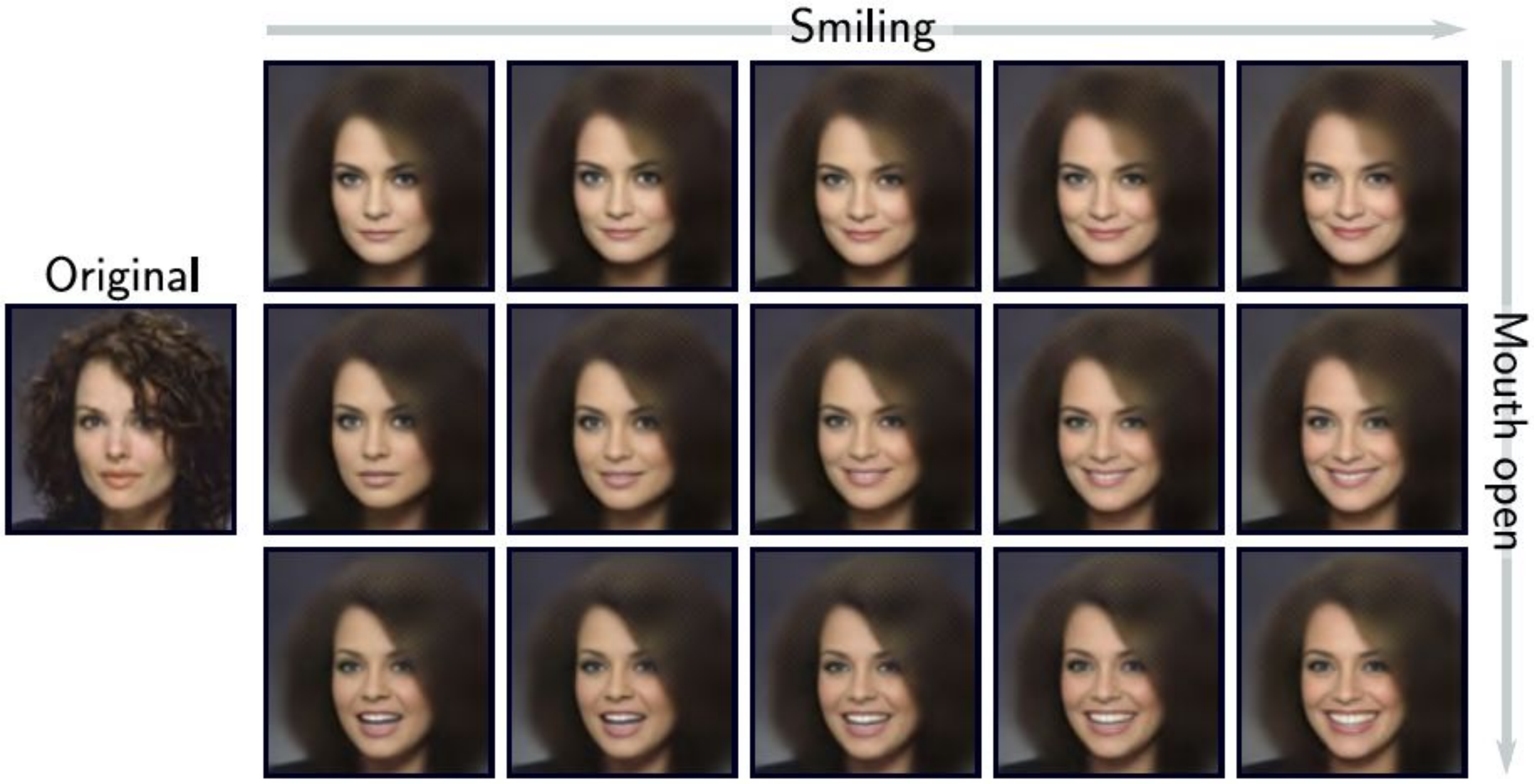


Figure 17.13 Resynthesis. The original image on the left is projected into the latent space using the encoder, and the mean of the predicted Gaussian is chosen to represent the image. The center-left image in the grid is the reconstruction of the input. The other images are reconstructions after manipulating the latent space in directions representing smiling/neutral (horizontal) and mouth open/closed (vertical). Adapted from White (2016).