Advanced Deep Learning

DATA.ML.230

Variational Autoencoders

From: Simon J. D. Prince, Chapter 17 – Understanding Deep Learning, MIT Press (19 May 2025)

Latent variable models

- Learn a probability distribution Pr(**x**) over a multi-dimensional variable **x**:
  - Model a joint distribution $Pr(\mathbf{x}, \mathbf{z})$ of the data **x** and an unobserved hidden (or latent) variable **z**
  - Marginalize the joint probability: $Pr(\mathbf{x}) = \int Pr(\mathbf{x}, \mathbf{z}) d\mathbf{z}$

Latent variable models

- Learn a probability distribution Pr(**x**) over a multi-dimensional variable **x**:
  - Model a joint distribution $Pr(\mathbf{x}, \mathbf{z})$ of the data **x** and an unobserved hidden (or latent) variable **z**
  - Marginalize the joint probability: $Pr(\mathbf{x}) = \int Pr(\mathbf{x}, \mathbf{z}) d\mathbf{z}$

$$Pr(\mathbf{x}) = \int Pr(\mathbf{x}|\mathbf{z}) Pr(\mathbf{z}) d\mathbf{z}$$

Simple Pr(**x**|**z**) and Pr(**z**) can define complex distributions Pr(**x**)

Latent variable models

Example: Mixture of Gaussians

- Discrete latent variable z
- Pr(z) is a categorical distribution

$$\begin{aligned} Pr(z=n) &= \lambda_n \\ Pr(x|z=n) &= \mathrm{Norm}_x\left[\mu_n, \sigma_n^2\right] \end{aligned}$$

Latent variable models

Example: Mixture of Gaussians

- Discrete latent variable z
- Pr(z) is a categorical distribution

$$\begin{aligned} Pr(z=n) &= \lambda_n \\ Pr(x|z=n) &= \text{Norm}_x\left[\mu_n, \sigma_n^2\right] \end{aligned}$$

Marginalizing over z:

$$\begin{aligned} Pr(x) &= \sum_{n=1}^{N} Pr(x, z=n) \\ &= \sum_{n=1}^{N} Pr(x|z=n) \cdot Pr(z=n) \\ &= \sum_{n=1}^{N} \lambda_n \cdot \text{Norm}_x\left[\mu_n, \sigma_n^2\right] \end{aligned}$$

Latent variable models

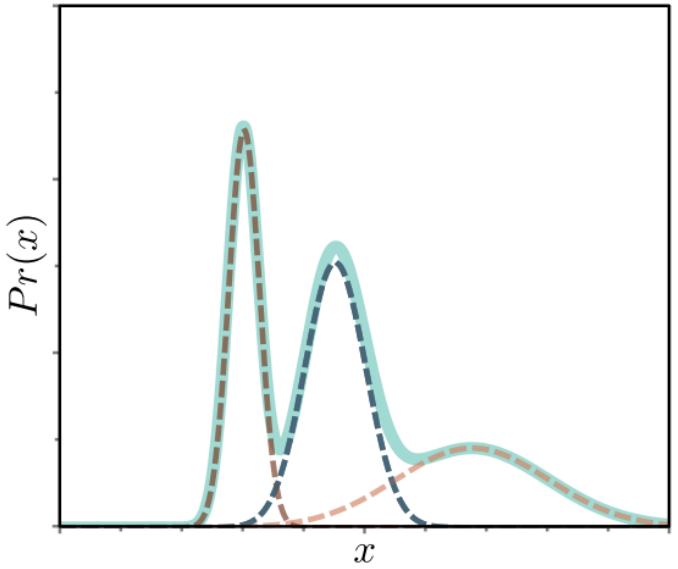
Example: Mixture of Gaussians

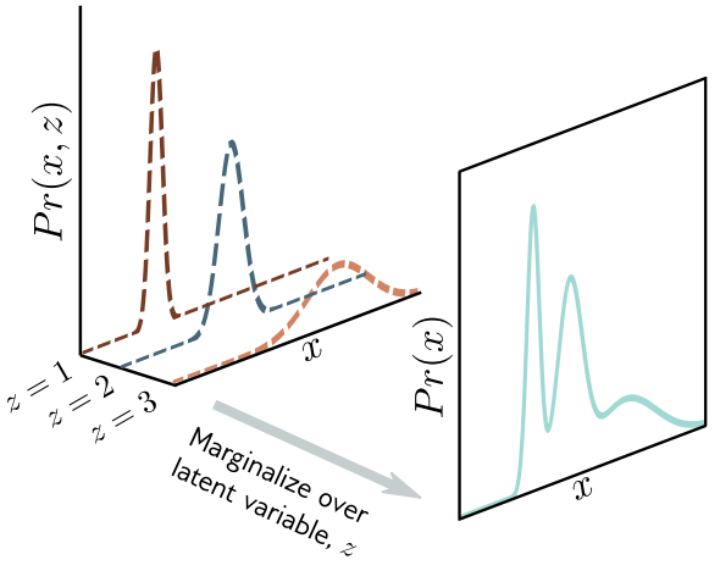
- Discrete latent variable z
- Pr(z) is a categorical distribution

$$\begin{aligned} Pr(z=n) &= \lambda_n \\ Pr(x|z=n) &= \text{Norm}_x\left[\mu_n, \sigma_n^2\right] \end{aligned}$$

Marginalizing over z:

$$\begin{aligned} Pr(x) &= \sum_{n=1}^{N} Pr(x, z=n) \\ &= \sum_{n=1}^{N} Pr(x|z=n) \cdot Pr(z=n) \\ &= \sum_{n=1}^{N} \lambda_n \cdot \text{Norm}_x\left[\mu_n, \sigma_n^2\right] \end{aligned}$$





Latent variable models

Nonlinear latent variable model:

- Both data **x** and latent variable **z** are continuous and multivariate

Latent variable models

Nonlinear latent variable model:

- Both data **x** and latent variable **z** are continuous and multivariate
- Pr(**z**) is a standard normal distribution $Pr(\mathbf{z}) = \mathrm{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$

Latent variable models

Nonlinear latent variable model:

- Both data **x** and latent variable **z** are continuous and multivariate
- Pr(**z**) is a standard normal distribution $Pr(\mathbf{z}) = \mathrm{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$
- Pr(**x**|**z**, $\boldsymbol{\phi}$) is also normally distributed: $Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi}) = \mathrm{Norm}_{\mathbf{x}}\left[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}\right]$

Latent variable models

Nonlinear latent variable model:

- Both data **x** and latent variable **z** are continuous and multivariate
- Pr(**z**) is a standard normal distribution $Pr(\mathbf{z}) = \mathrm{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$
- Pr(**x**|**z**, $\boldsymbol{\phi}$) is also normally distributed: $Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi}) = \mathrm{Norm}_{\mathbf{x}}\left[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}\right]$

A deep neural network

Latent variable models

Nonlinear latent variable model:

- Both data **x** and latent variable **z** are continuous and multivariate
- Pr(**z**) is a standard normal distribution $Pr(\mathbf{z}) = \mathrm{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$
- Pr(**x**|**z**, $\boldsymbol{\phi}$) is also normally distributed: $Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi}) = \mathrm{Norm}_{\mathbf{x}}\left[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}\right]$
- Data probability:

$$\begin{aligned} Pr(\mathbf{x}|\boldsymbol{\phi}) &= \int Pr(\mathbf{x}, \mathbf{z}|\boldsymbol{\phi}) d\mathbf{z} \\ &= \int Pr(\mathbf{x}|\mathbf{z}, \boldsymbol{\phi}) \cdot Pr(\mathbf{z}) d\mathbf{z} \\ &= \int \mathrm{Norm}_{\mathbf{x}}\left[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}\right] \cdot \mathrm{Norm}_{\mathbf{z}}\left[\mathbf{0}, \mathbf{I}\right] d\mathbf{z} \end{aligned}$$

Latent variable models

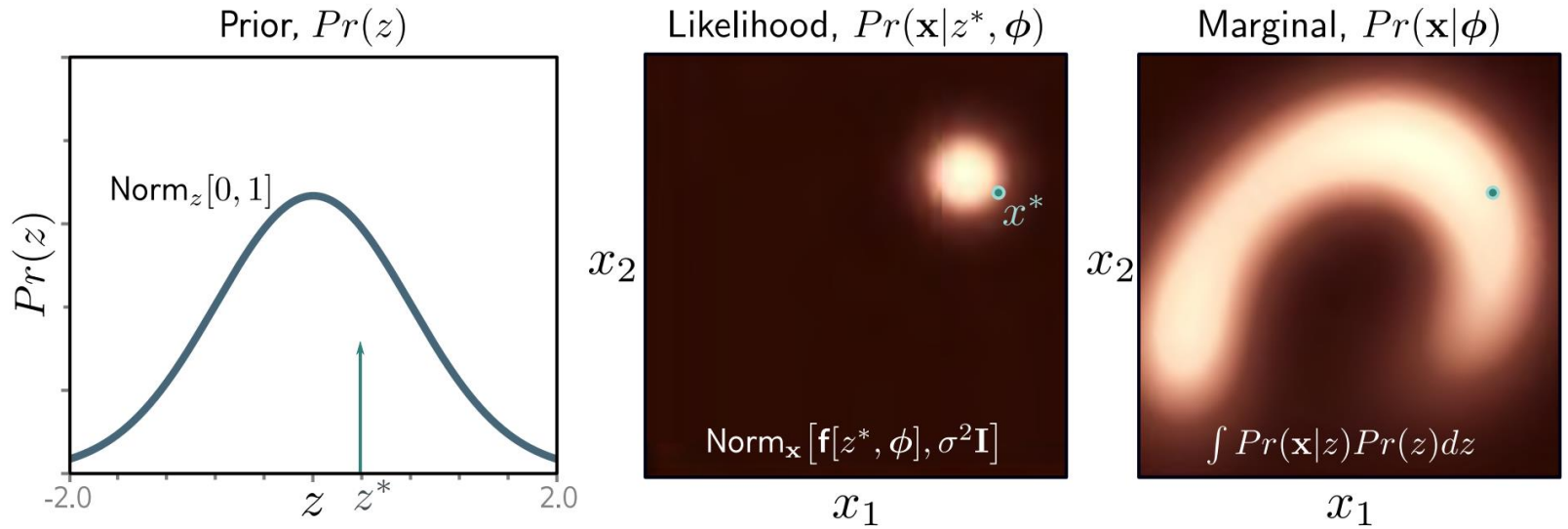
Nonlinear latent variable model:

- Data generation using ancestral sampling:
  - Draw $\mathbf{z}^*$ from the prior $Pr(\mathbf{z})$
  - Pass it through the network $\mathbf{f}[\mathbf{z}^*, \boldsymbol{\phi}]$ to compute the mean of $Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})$
  - Draw the new sample $\mathbf{x}^*$ from $Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})$

Latent variable models

Nonlinear latent variable model:

- Data generation using ancestral sampling:
  - Draw $\mathbf{z}^*$ from the prior $Pr(\mathbf{z})$
  - Pass it through the network $\mathbf{f}[\mathbf{z}^*, \boldsymbol{\phi}]$ to compute the mean of $Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})$
  - Draw the new sample $\mathbf{x}^*$ from $Pr(\mathbf{x}|\mathbf{z}^*, \boldsymbol{\phi})$



Latent variable models

Nonlinear latent variable model:

- Training by maximizing the log-likelihood over a training dataset $\{\mathbf{x}_i\}_{i=1}^{I}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log \left[Pr(\mathbf{x}_i|\boldsymbol{\phi}) \right] \right]$$

where:

- we consider that the variance term is σ^2 and known
- $Pr(\mathbf{x}_i|\boldsymbol{\phi}) = \int \text{Norm}_{\mathbf{x}_i}[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}] \cdot \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}] d\mathbf{z}$

Latent variable models

Nonlinear latent variable model:

- Training by maximizing the log-likelihood over a training dataset $\{\mathbf{x}_i\}_{i=1}^{I}$:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log \left[Pr(\mathbf{x}_i|\boldsymbol{\phi}) \right] \right]$$

where:

- we consider that the variance term is σ^2 and known
- $Pr(\mathbf{x}_i|\boldsymbol{\phi}) = \int \text{Norm}_{\mathbf{x}_i}[\mathbf{f}[\mathbf{z}, \boldsymbol{\phi}], \sigma^2\mathbf{I}] \cdot \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}] d\mathbf{z}$

Intractable solution:

- No closed-form expression for the integral
- No easy way to evaluate for x

Latent variable models

Nonlinear latent variable model:

- Evidence lower bound (ELBO):
  - Define a function that is always less than or equal to the log-likelihood for a given value of $\boldsymbol{\phi}$
  - This function depends on some other parameter $\boldsymbol{\theta}$

Latent variable models

Nonlinear latent variable model:

- Evidence lower bound (ELBO):
  - Define a function that is always less than or equal to the log-likelihood for a given value of $\boldsymbol{\phi}$
  - This function depends on some other parameter $\boldsymbol{\theta}$

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

Latent variable models

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

Nonlinear latent variable model:

- Evidence lower bound (ELBO):
  - Define a function that is always less than or equal to the log-likelihood for a given value of $\boldsymbol{\phi}$
  - This function depends on some other parameter $\boldsymbol{\theta}$

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

Jensen's inequality

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

Latent variable models

Nonlinear latent variable model:

- Evidence lower bound (ELBO):
  - Define a function that is always less than or equal to the log-likelihood for a given value of $\boldsymbol{\phi}$
  - This function depends on some other parameter $\boldsymbol{\theta}$

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

- Training by maximizing the ELBO as a function of both $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$

Latent variable models

Nonlinear latent variable model:

- Evidence lower bound (ELBO):

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

Latent variable models

Nonlinear latent variable model:

- Evidence lower bound (ELBO):

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

Reconstruction accuracy

Degree to which the learned distribution $q(\mathbf{z}|\boldsymbol{\theta})$ matches the prior

Variational autoencoder

- We use a neural network with parameters $\boldsymbol{\phi}$ that computes the ELBO:

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

Variational autoencoder

- We use a neural network with parameters $\boldsymbol{\phi}$ that computes the ELBO:

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

  in which we use the variational approximation of $q(\mathbf{z}|\boldsymbol{\theta})$:

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \text{Norm}_{\mathbf{z}}\left[\mathbf{g}_{\boldsymbol{\mu}}[\mathbf{x}, \boldsymbol{\theta}], \mathbf{g}_{\boldsymbol{\Sigma}}[\mathbf{x}, \boldsymbol{\theta}]\right]$$

Neural network $\mathbf{g}[\mathbf{x}, \boldsymbol{\theta}]$ predicting the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$

Variational autoencoder

- We use a neural network with parameters $\boldsymbol{\phi}$ that computes the ELBO:

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

Intractable integral ➔

Approximate it by sampling

Monte Carlo estimate:

- For any function a[•], and N samples from $q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$

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

Variational autoencoder

- We use a neural network with parameters $\boldsymbol{\phi}$ that computes the ELBO:

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

Intractable integral ➔

Approximate it by sampling

Monte Carlo estimate:

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

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

Variational autoencoder

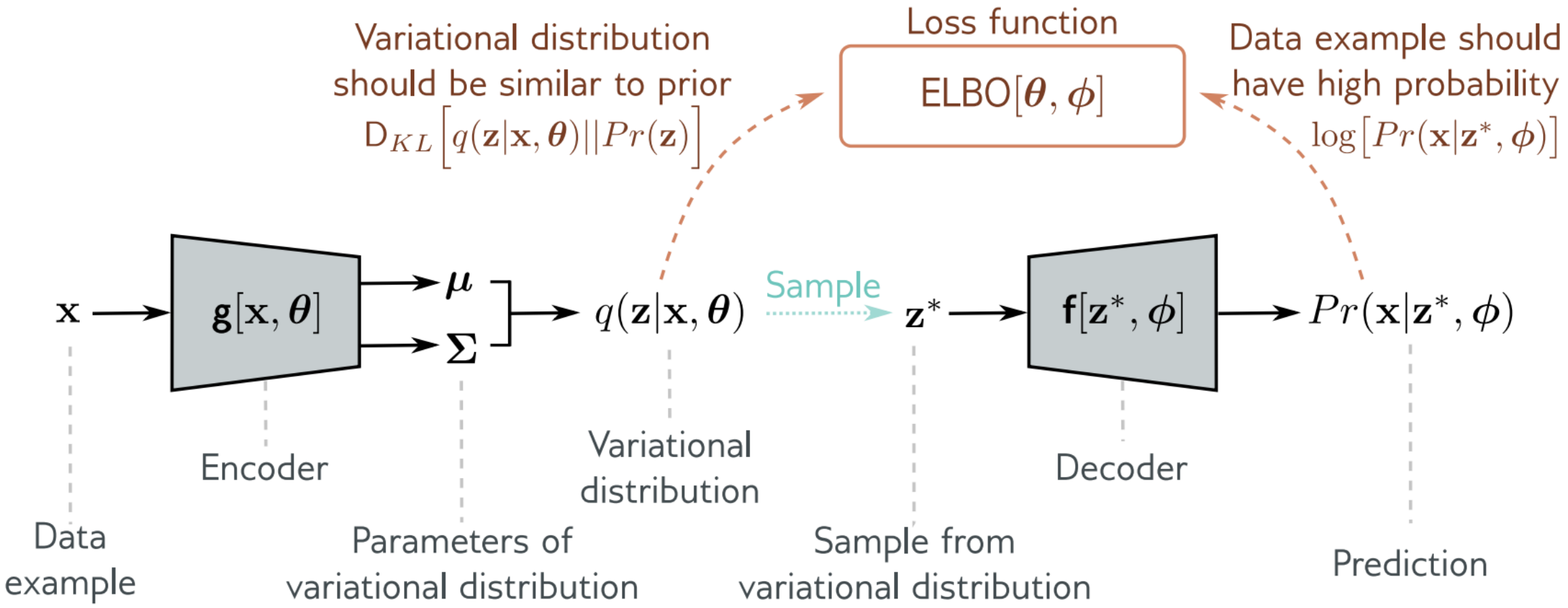
- We use a neural network with parameters $\boldsymbol{\phi}$ that computes the ELBO:

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

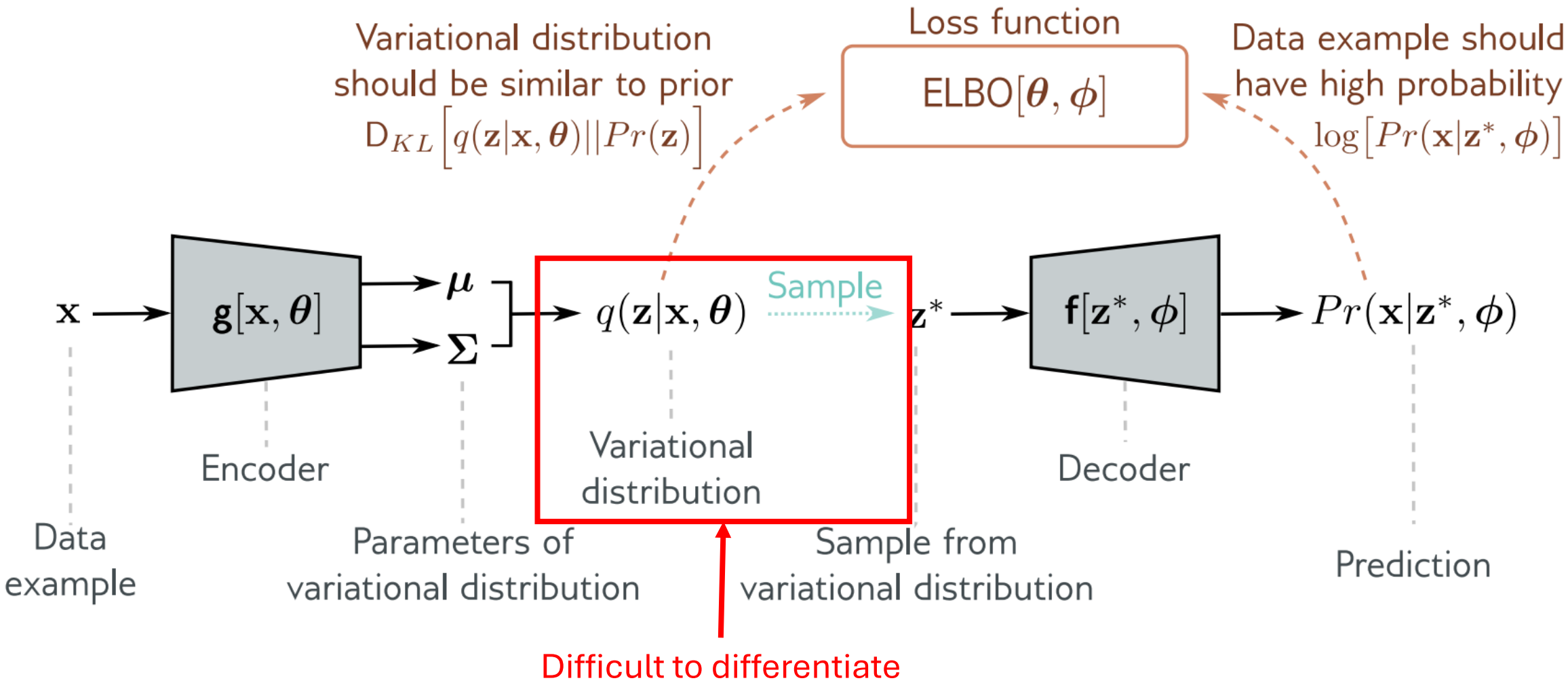
KL Divergence between two Normal distributions:

$$\mathrm{D}_{KL}\left[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\middle|\middle|Pr(\mathbf{z})\right] = \frac{1}{2}\left(\mathrm{Tr}[\boldsymbol{\Sigma}] + \boldsymbol{\mu}^T\boldsymbol{\mu} - D_{\mathbf{z}} - \log\left[\det[\boldsymbol{\Sigma}]\right]\right)$$

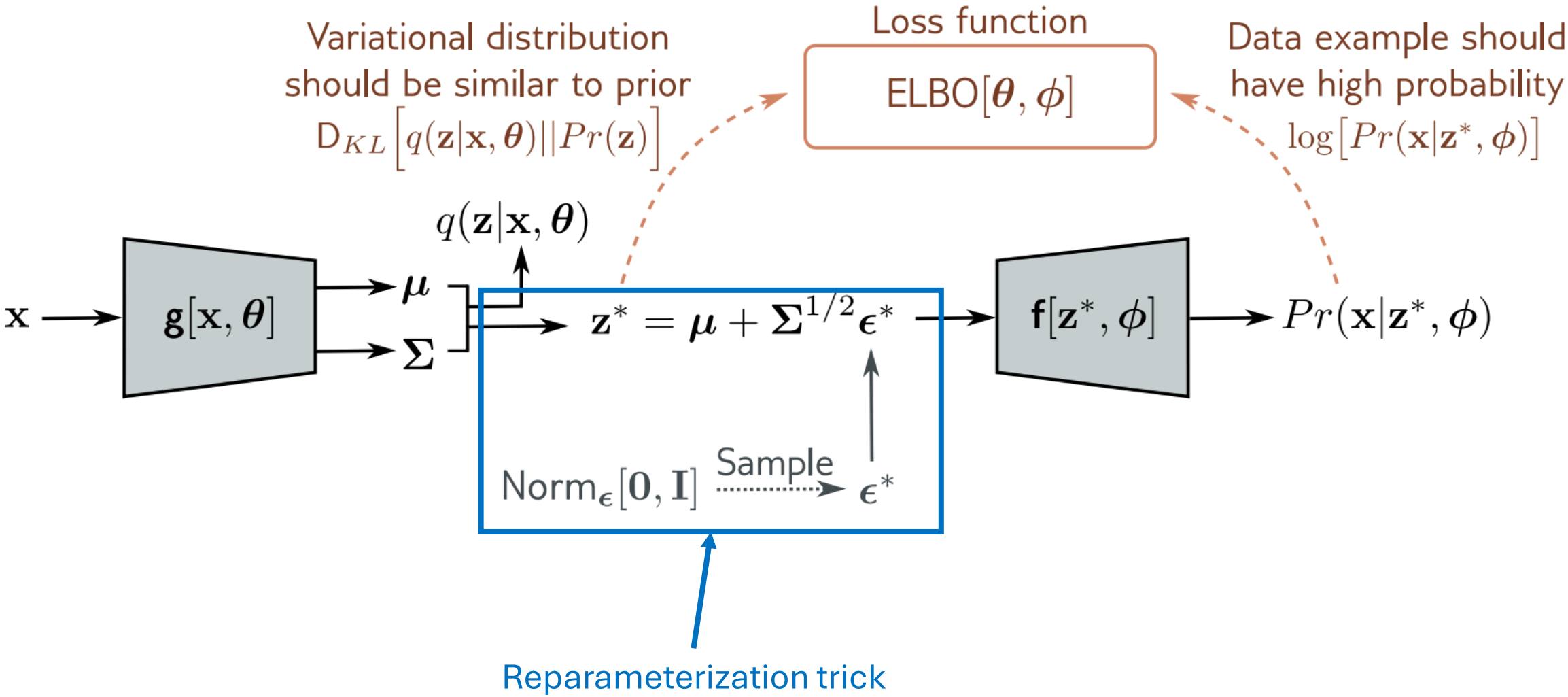
Variational autoencoder



Variational autoencoder



Variational autoencoder



Applications of Variational autoencoders

Approximating sample probability through importance sampling:

- Sample **z** from an auxiliary distribution q(**z**)
- Evaluate Pr(**x**|$\mathbf{z}_n$)
- Rescale the resulting values by the probability q(**z**)

$$
\begin{aligned}
Pr(\mathbf{x}) &= \int Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})d\mathbf{z} = \int \frac{Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z})d\mathbf{z} \\
&= \mathbb{E}_{q(\mathbf{z})}\left[\frac{Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})}{q(\mathbf{z})}\right] \approx \frac{1}{N}\sum_{n=1}^{N} \frac{Pr(\mathbf{x}|\mathbf{z}_n)Pr(\mathbf{z}_n)}{q(\mathbf{z}_n)}
\end{aligned}
$$

Applications of Variational autoencoders

Sample generation:

- Draw **z** from the prior Pr(**z**) or from the *aggregated posterior*

$$q(\mathbf{z}|\boldsymbol{\theta}) = (1/I)\sum_i q(\mathbf{z}|\mathbf{x}_i, \boldsymbol{\theta})$$

- Pass it through the decoder f[**z**, $\boldsymbol{\phi}$]
- Add noise according to Pr(**x**|**f**[**z**, $\boldsymbol{\phi}$])

Applications of Variational autoencoders

Sample generation

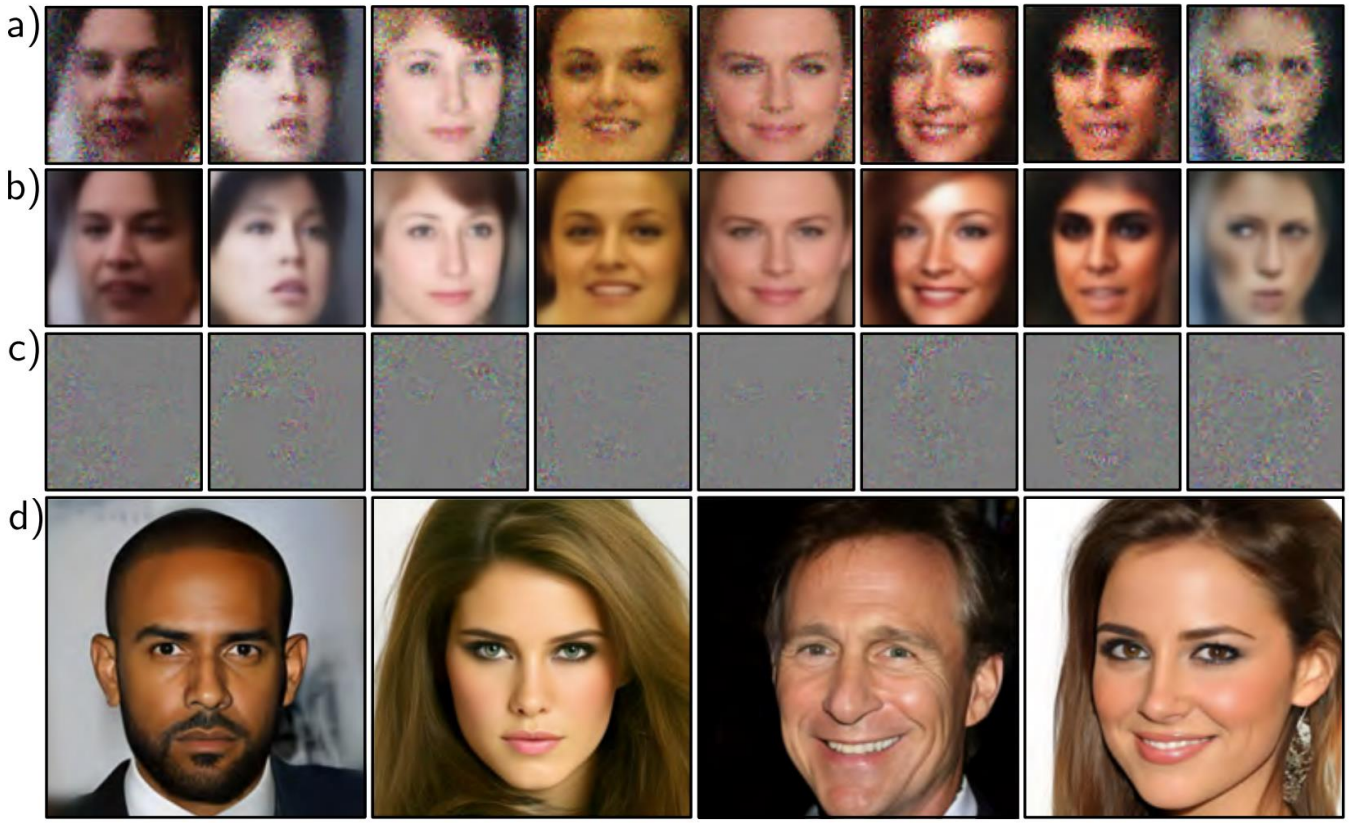


Figure 17.12 Sampling from a standard VAE trained on CELEBA. In each column, a latent variable $\mathbf{z}^*$ is drawn and passed through the model to predict the mean $\mathbf{f}[\mathbf{z}^*, \boldsymbol{\phi}]$ before adding independent Gaussian noise (see figure 17.3). a) A set of samples that are the sum of b) the predicted means and c) spherical Gaussian noise vectors. The images look too smooth before we add the noise and too noisy afterward. This is typical, and usually, the noise-free version is shown since the noise is considered to represent aspects of the image that are not modeled. Adapted from Dorta et al. (2018). d) It is now possible to generate high-quality images from VAEs using hierarchical priors, specialized architecture, and careful regularization. Adapted from Vahdat & Kautz (2020).

Applications of Variational autoencoders

Resynthesis:

1. Project multiple images representing two different sets (e.g., “neutral face” and “smiling face”)

Applications of Variational autoencoders

Resynthesis:

1. Project multiple images representing two different sets (e.g., “neutral face” and “smiling face”)

To project a data point **x** into the latent space:

- take the mean of the distribution predicted by the encoder, or
- find the latent variable z that maximizes the posterior probability, which is proportional to $Pr(\mathbf{x}|\mathbf{z})Pr(\mathbf{z})$

Applications of Variational autoencoders

Resynthesis:

1. Project multiple images representing two different sets (e.g., “neutral face” and “smiling face”)
2. Calculate the vector that represents the change between the two sets of images

Applications of Variational autoencoders

Resynthesis:

1. Project multiple images representing two different sets (e.g., “neutral face” and “smiling face”)
2. Calculate the vector that represents the change between the two sets of images

Calculate the difference between the vector means of the two groups

Applications of Variational autoencoders

Resynthesis:

1. Project multiple images representing two different sets (e.g., “neutral face” and “smiling face”)
2. Calculate the vector that represents the change between the two sets of images
3. Project the image of interest to the latent space

Applications of Variational autoencoders

Resynthesis:

1. Project multiple images representing two different sets (e.g., “neutral face” and “smiling face”)
2. Calculate the vector that represents the change between the two sets of images
3. Project the image of interest to the latent space
4. Add to (or subtract from) latent space representation of the image of interest the change vector

Applications of Variational autoencoders

Resynthesis:

1. Project multiple images representing two different sets (e.g., “neutral face” and “smiling face”)
2. Calculate the vector that represents the change between the two sets of images
3. Project the image of interest to the latent space
4. Add to (or subtract from) latent space representation of the image of interest the change vector
5. Use the decoder to map the new latent space representation to a new image

Applications of Variational autoencoders

Resynthesis

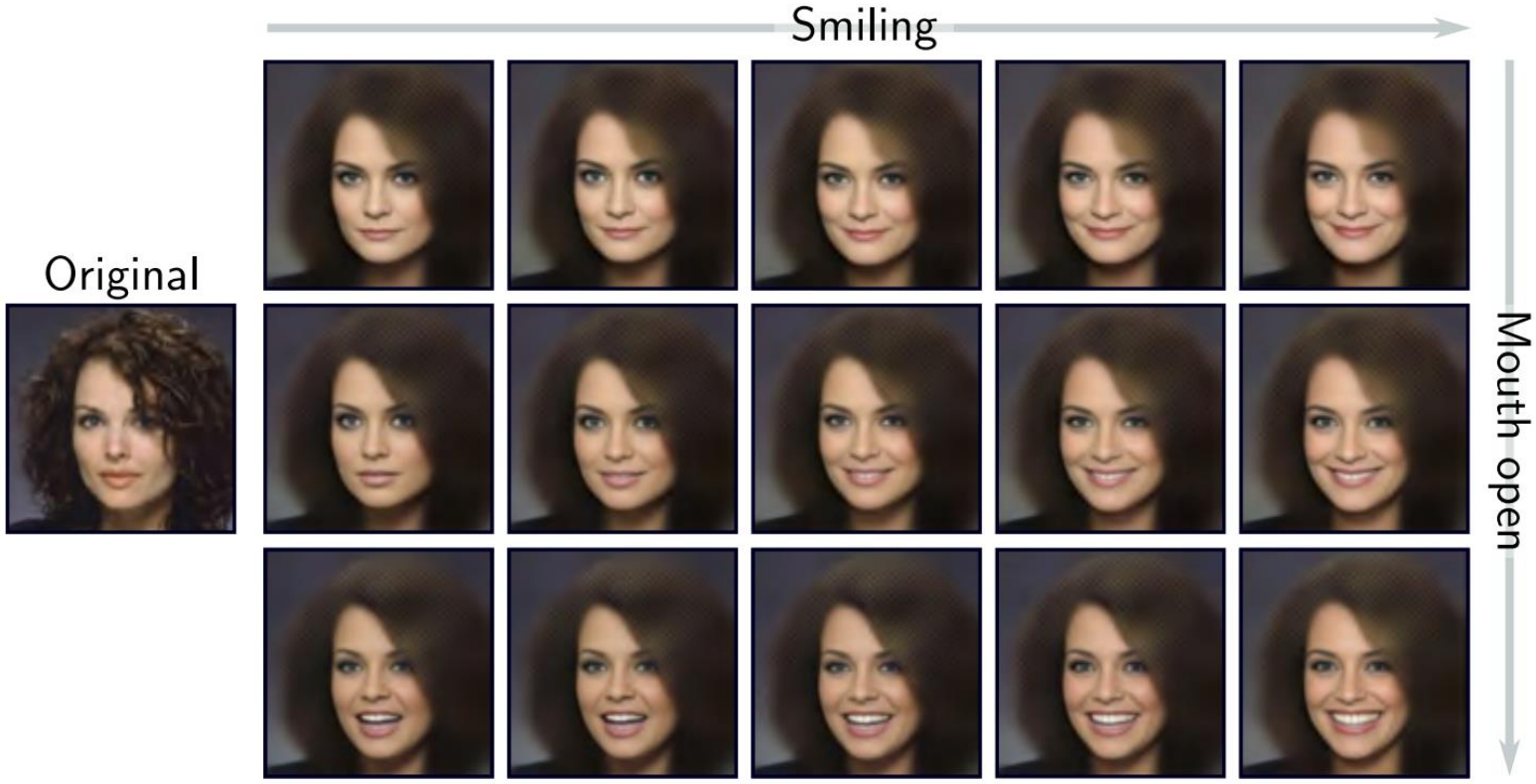


Figure 17.13 Resynthesis. The original image on the left is projected into the latent space using the encoder, and the mean of the predicted Gaussian is chosen to represent the image. The center-left image in the grid is the reconstruction of the input. The other images are reconstructions after manipulating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

Applications of Variational autoencoders

Disentanglement:

- Instead of using labeled data to determine the directions of meaningful properties (e.g., neutral, smiling) *learn* coordinate directions corresponding to properties

  → learn a disentangled latent space

Applications of Variational autoencoders

Disentanglement:

- Instead of using labeled data to determine the directions of meaningful properties (e.g., neutral, smiling) *learn* coordinate directions corresponding to properties
- Use appropriate regularization terms:

$$L_{\text{new}} = -\text{ELBO}[\boldsymbol{\theta}, \boldsymbol{\phi}] + \lambda_1 \mathbb{E}_{Pr(\mathbf{x})}\Big[\text{r}_1\big[q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})\big]\Big] + \lambda_2 \text{r}_2\big[q(\mathbf{z}|\boldsymbol{\theta})\big]$$

Aggregate posterior

Posterior

Applications of Variational autoencoders

Disentanglement

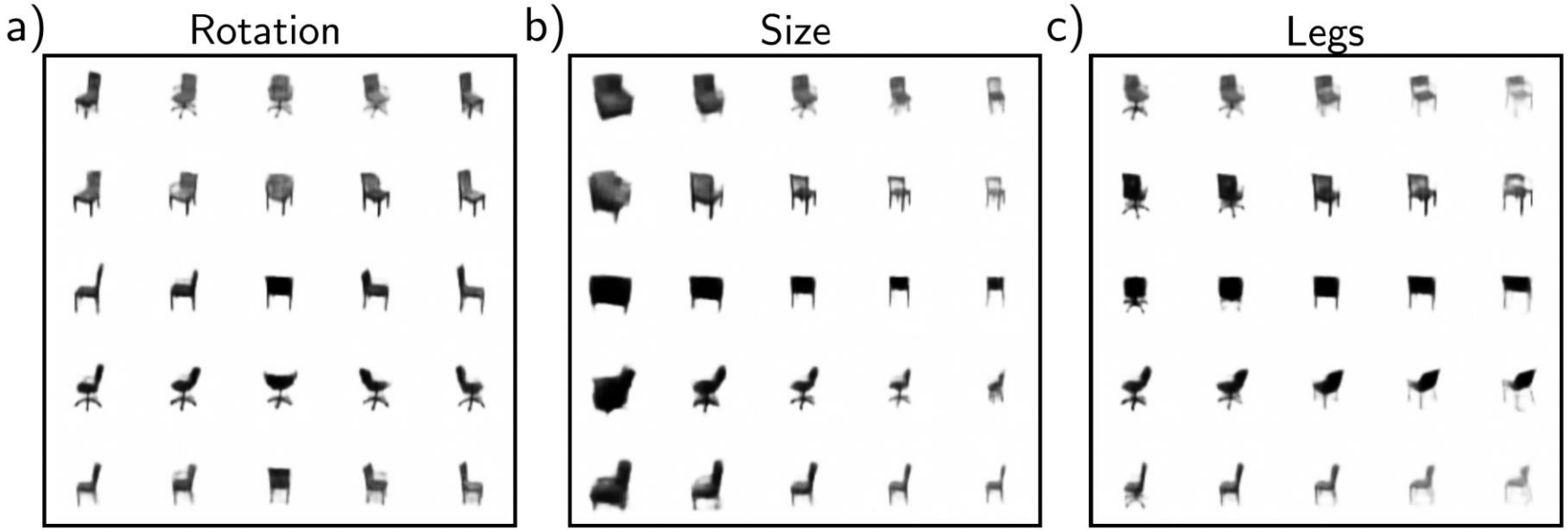
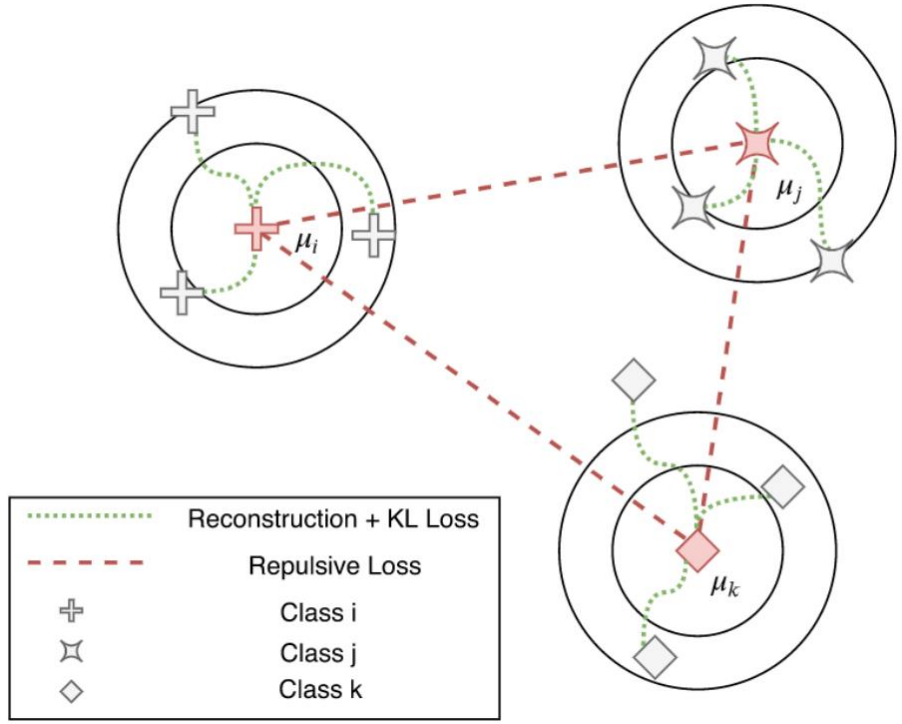


Figure 17.14 Disentanglement in the total correlation VAE. The VAE model is modified so that the loss function encourages the total correlation of the latent variables to be minimized and hence encourages disentanglement. When trained on a dataset of images of chairs, several of the latent dimensions have clear real-world interpretations, including a) rotation, b) overall size, and c) legs (swivel chair versus normal). In each case, the central column depicts samples from the model, and as we move left to right, we are subtracting or adding a coordinate vector in latent space. Adapted from Chen et al. (2018d).

Regularized Discriminative VAE

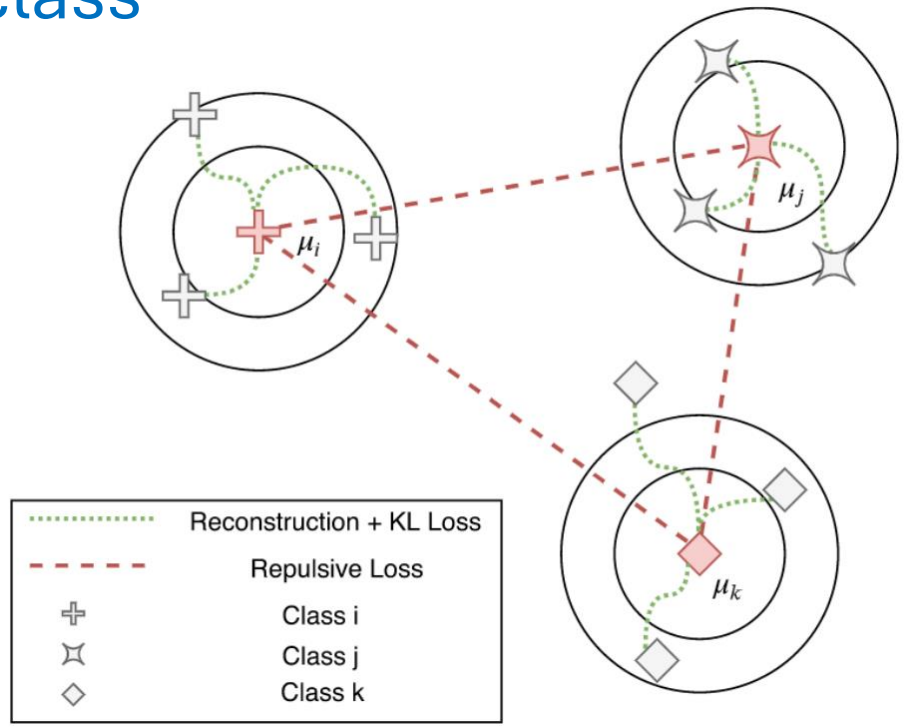
- Capable of learning efficient discriminative representations ➔ Allows for distinguishing different classes



Passalis, Iosifidis, Gabbouj, Tefas, “Variance-preserving Deep Metric Learning for Content-based Image Retrieval”, Pattern Recognition Letters, 2020

Regularized Discriminative VAE

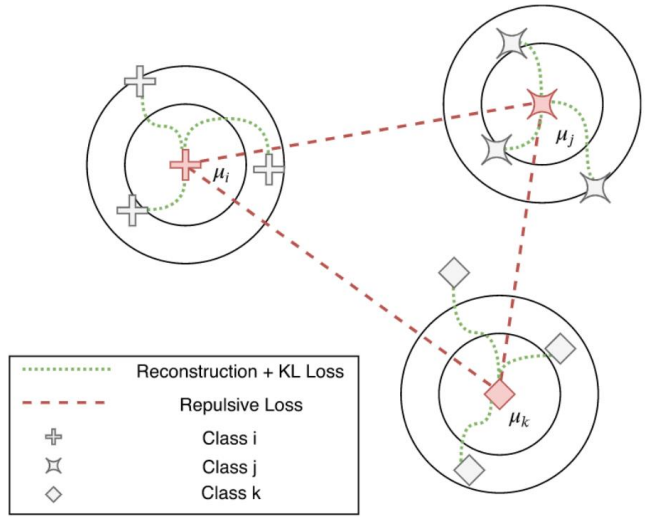
- Capable of learning efficient discriminative representations ➔ Allows for distinguishing different classes
- Works by modeling the latent generative factors for each class ➔ Keep the important information of each class



Passalis, Iosifidis, Gabbouj, Tefas, “Variance-preserving Deep Metric Learning for Content-based Image Retrieval”, Pattern Recognition Letters, 2020

Regularized Discriminative VAE

- Capable of learning efficient discriminative representations ➔ Allows for distinguishing different classes
- Works by modeling the latent generative factors for each class ➔ Keep the important information of each class
- Represents both sub-classes of the in-domain classes, as well as samples that belong to out-of-domain classes



Passalis, Iosifidis, Gabbouj, Tefas, “Variance-preserving Deep Metric Learning for Content-based Image Retrieval”, Pattern Recognition Letters, 2020

Regularized Discriminative VAE

- Capable of learning efficient discriminative representations ➔ Allows for distinguishing different classes
- Works by modeling the latent generative factors for each class ➔ Keep the important information of each class
- Represents both sub-classes of the in-domain classes, as well as samples that belong to out-of-domain classes

This is achieved by adding a repulsive loss to the function:

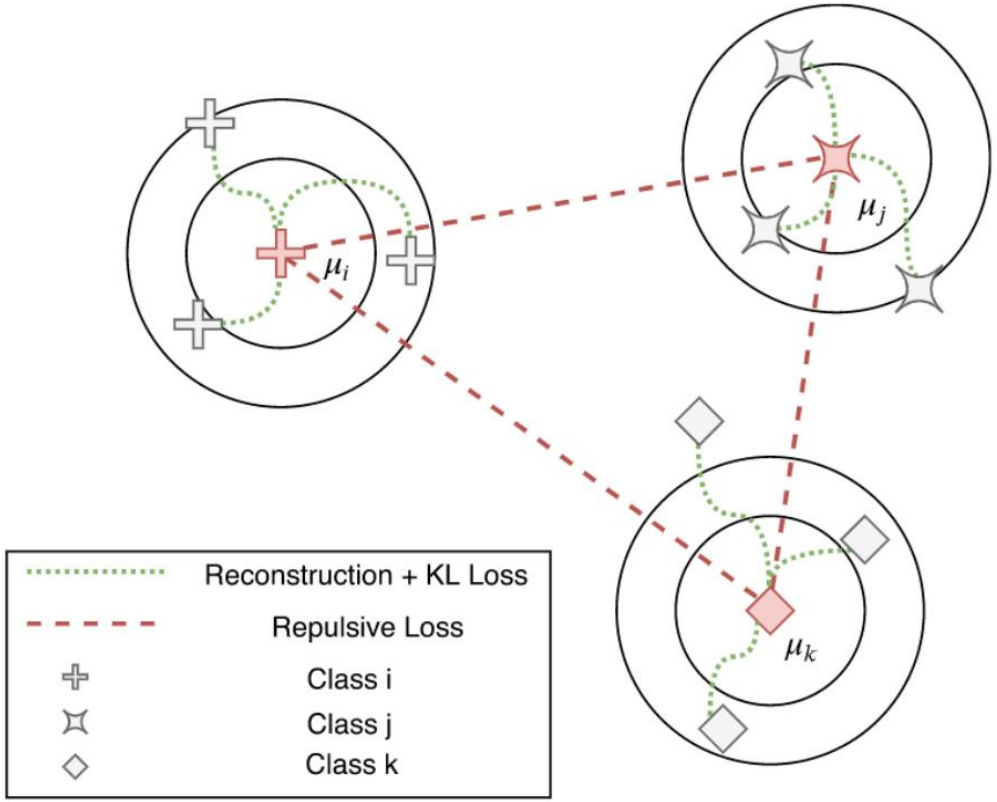
Total loss = Reconstruction loss + KL loss + Repulsive loss

Passalis, Iosifidis, Gabbouj, Tefas, “Variance-preserving Deep Metric Learning for Content-based Image Retrieval”, Pattern Recognition Letters, 2020

Regularized Discriminative VAE

Considering: N_C as the number of Gaussians and ρ as the minimum distance between them:

$$L_{rep} = \frac{1}{\rho} \sum_{i=1}^{N_C} \sum_{i=1, j \neq i}^{N_C} max(0, \rho - \|\mu_i - \mu_j\|_2^2)^2$$



Passalis, Iosifidis, Gabbouj, Tefas, “Variance-preserving Deep Metric Learning for Content-based Image Retrieval”, Pattern Recognition Letters, 2020