Bayesian Methods for Machine Learning

Lecture 7 - Deep generative models

Simon Leglaive

CentraleSupélec

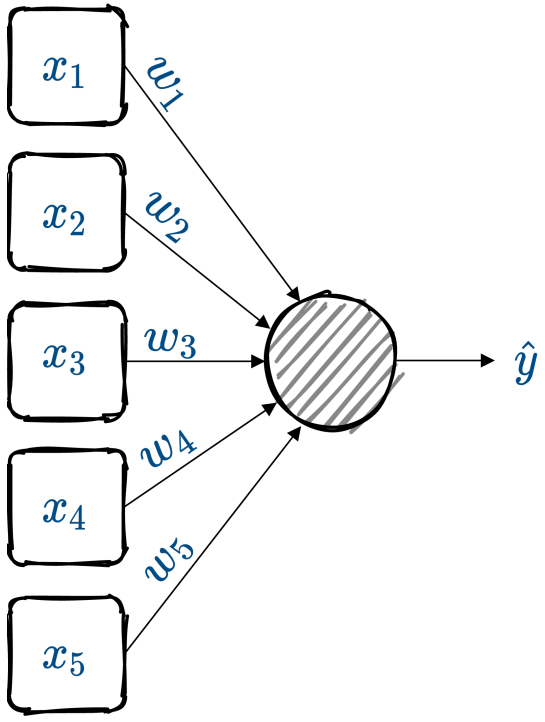
Introduction

Artificial neuron

$$\hat{y} = f(\mathbf{x}; \theta) = \sigma\left(\mathbf{w}^T \mathbf{x} + b\right) = \sigma\left(\sum_i w_i x_i + b\right),$$

where

- $\mathbf{x}$ is the input vector;
- $\hat{y}$ the scalar output;
- $\mathbf{w}$ is the weight vector;
- b is the scalar bias;
- σ is a non-linear activation function;
- $\theta = \{\mathbf{w}, b\}$ are the neuron's parameters.



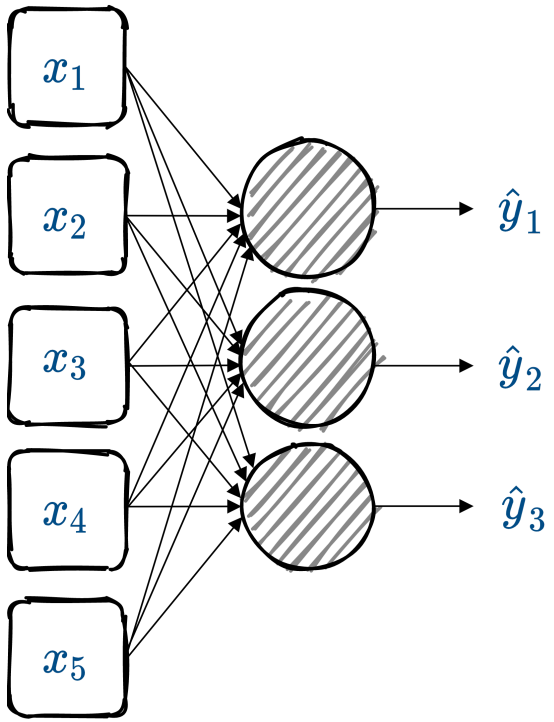
Layer

Neurons can be composed in parallel to form a **layer** with multiple outputs:

$$\hat{\mathbf{y}} = f(\mathbf{x}; \theta) = \sigma\left(\mathbf{W}^T \mathbf{x} + \mathbf{b}\right),$$

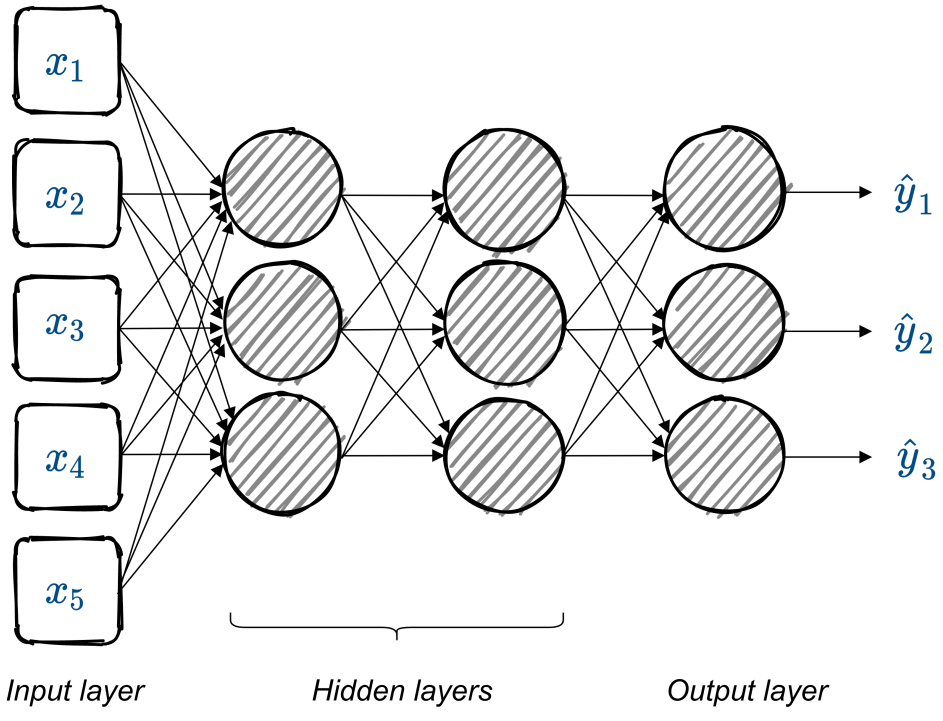
where we have now

- an element-wise activation function,
- an output vector $\hat{\mathbf{y}}$,
- a weight matrix $\mathbf{W}$,
- a bias vector $\mathbf{b}$,
- such that $\theta = \{\mathbf{W}, \mathbf{b}\}$.



Multi-layer Perceptron

Similarly, layers can be composed in series, to form a multi-layer Perceptron, or feed-forward fully-connected neural network.



The model parameters are the weight matrices and bias vectors of all layers.

Supervised learning

- Training dataset (step 0): $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ where the $\mathbf{x}_i$'s are the inputs, and $\mathbf{y}_i$'s the labels.
- Model (step 1): A neural network $f(\cdot; \theta)$.
- Loss function (step 2):

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}} \ell\Big(\mathbf{y}_i, f(\mathbf{x}_i; \theta)\Big),$$

  where $\ell(\cdot, \cdot)$ is task-dependent.
- Training (step 3): Minimize $\mathcal{L}(\theta)$ with (variants of) gradient descent and backpropagation.
- Evaluation (step 4): Test the performance on examples that were not seen during training.

A great number of successful practical applications of machine and deep learning rely on supervised learning methods.

The Cityscapes dataset for semantic segmentation of urban images contains 5k images with high quality pixel-level annotations.

"Annotation and quality control required more than 1.5 h on average for a single image" (Cordts et al., 2016).

More than 300 days of annotation!

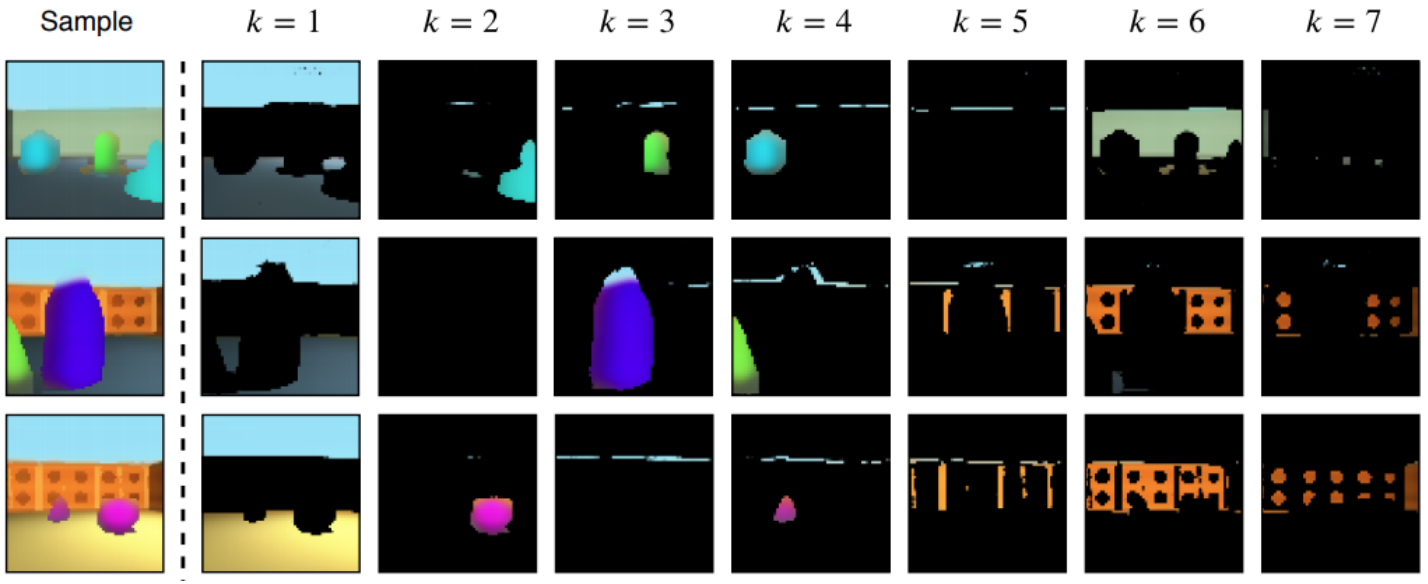
Zhao et al., "ICNet for Real-Time Semantic Segmentation on High-Resolution Images", ECCV 2018

M. Cordts et al., "The Cityscapes Dataset for Semantic Urban Scene Understanding", IEEE CVPR, 2016

Unsupervised learning

We need unsupervised methods that can learn to unveil the underlying (hidden) structure of the data without requiring ground-truth labels.

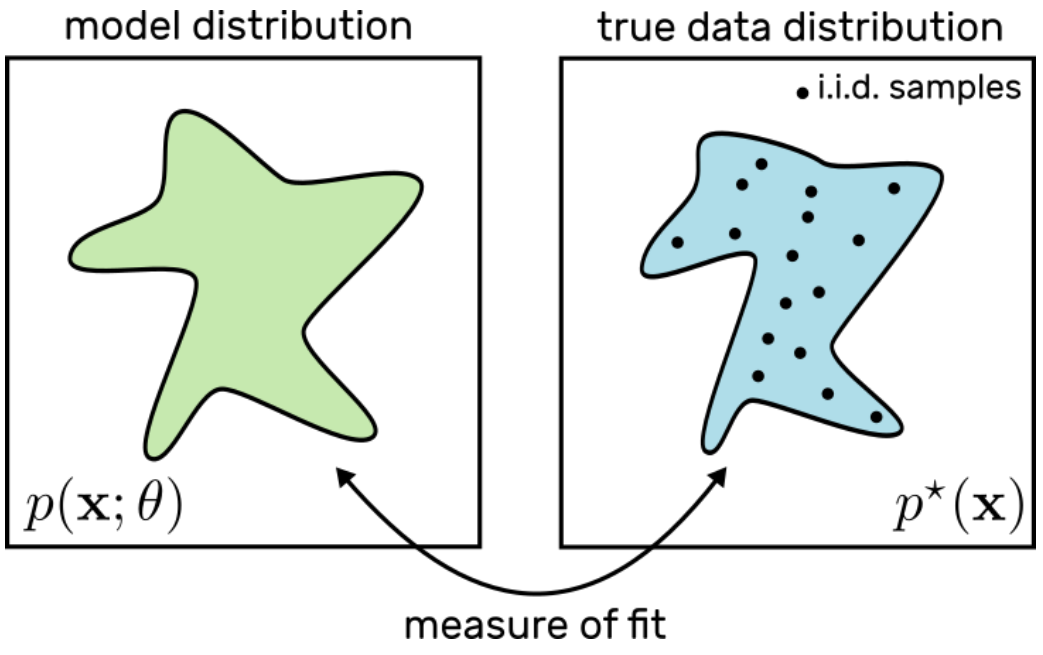
Semi-supervised methods are also of great interest.



Component-by-component scene generation with GENESIS, a generative model of 3D scenes capable of both decomposing and generating scenes by capturing relationships between scene components

Image credits: Engelcke et al., "GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations", ICLR 2020.

Generative modeling

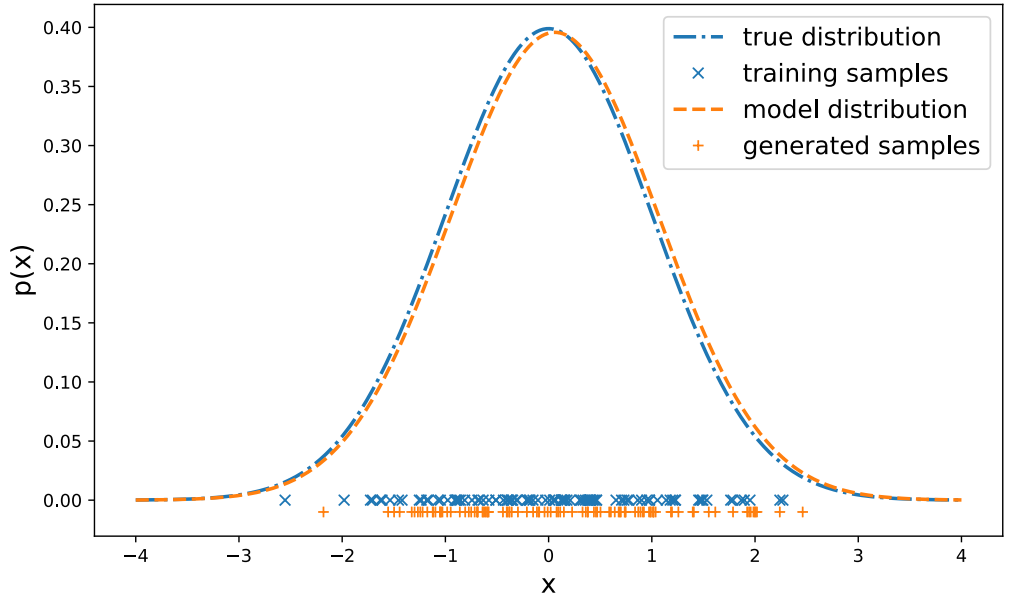


The goal is to tune the parameters θ of the model distribution $p(\mathbf{x};\theta)$ so that it is as close as possible to the true data distribution $p^\star(\mathbf{x})$, according to some measure of fit.

For instance, minimizing the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(p^\star(\mathbf{x}) \parallel p(\mathbf{x};\theta))$ is equivalent to maximum likelihood estimation.

1-dimensional toy example

- We define the true data distribution $p^\star(x)$ as a Gaussian distribution with known mean and variance, but of course usually we do not know this distribution.
- We assume a Gaussian model distribution $p(x;\theta) = \mathcal{N}(x;\mu,\sigma^2)$ where $\theta = \{\mu,\sigma^2\}$.
- The parameters θ are estimated by minimizing (a Monte Carlo estimate of) $D_{\mathrm{KL}}\left(p^\star(x) \parallel p(x;\theta)\right)$, i.e. by maximum likelihood.



10^6-dimensional example

T. Karras et al., Analyzing and Improving the Image Quality of StyleGAN, CVPR 2020.

Deep generative models (DGM)

- The model distribution $p(\mathbf{x}; \theta)$ is somehow defined by means of a neural network.
- Two seminal models: Variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014).
- DGM contain hundreds of thoughsands of parameters, trained in a scalable way using large datasets of unlabeled high-dimensional data.

I. Goodfellow et al., "Generative Adversarial Networks", NeurIPS 2014.

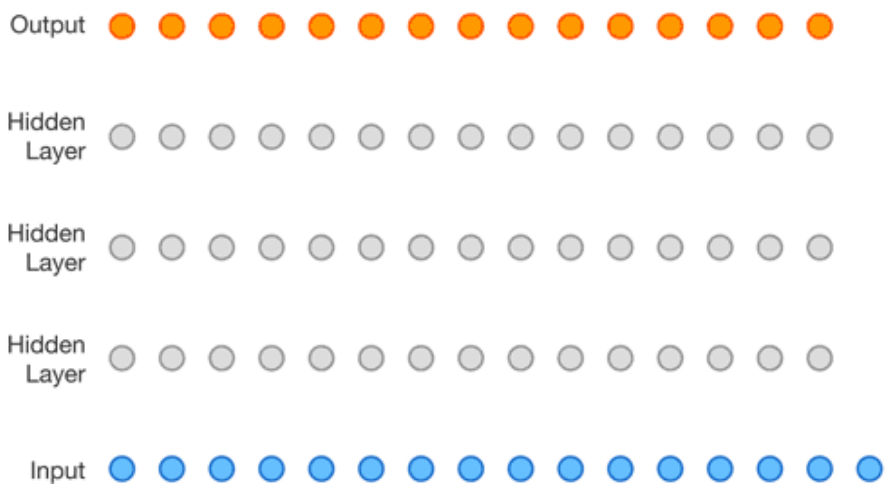
D.P. Kingma and M. Welling, "Auto-Encoding Variational Bayes", ICLR 2014.

D.J. Rezende et. al, " Stochastic backpropagation and approximate inference in deep generative models", ICML 2014.

Several flavors of DGMs

Autoregressive DGMs define the model distribution recursively, using the chain rule:

$$p(\mathbf{x};\theta) = p(x_1)\prod_{i=2}^{D} p(x_i \mid x_{1:i-1};\theta).$$

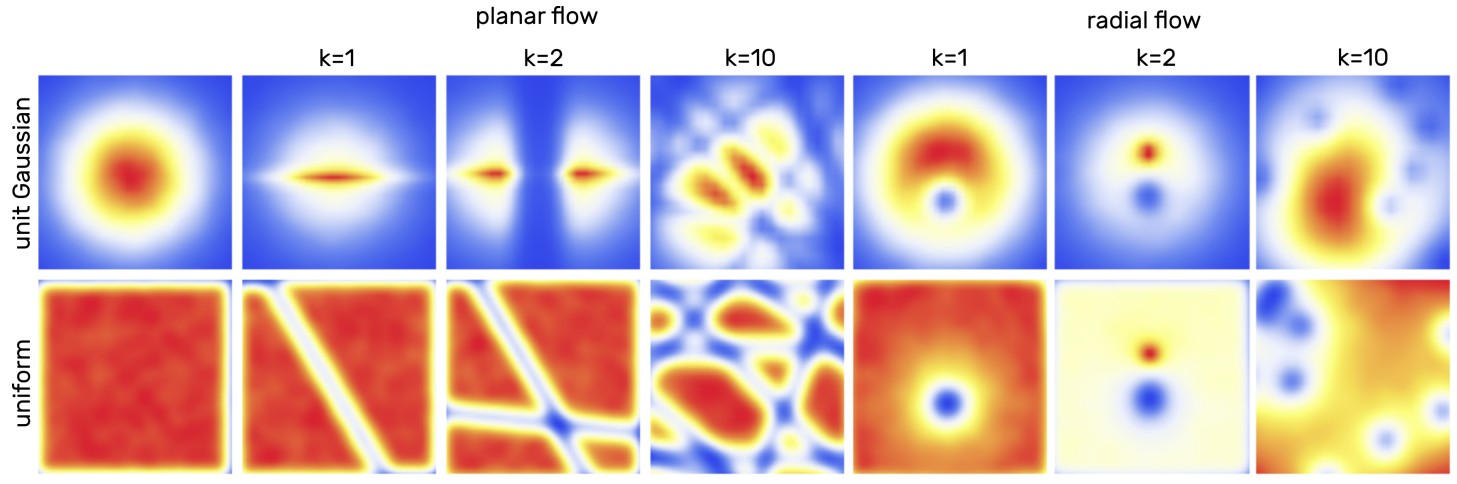


A. van den Oord et al., WaveNet: A generative model for raw audio, arxiv preprint, 2016.

Image credits: https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

Flow-based DGMs transform a simple distribution into a complex one by applying a sequence of invertible mappings:

$$\mathbf{x} = f_K \circ \ldots \circ f_1(\mathbf{z}_0), \qquad \mathbf{z}_0 \sim p_0(\mathbf{z}_0).$$

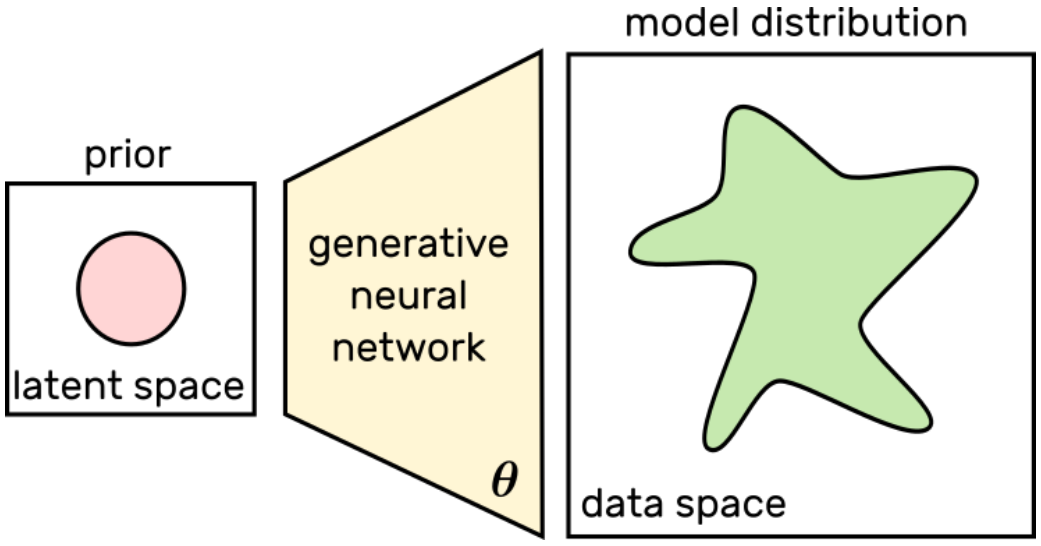


As mappings are invertible, we can express $p(\mathbf{x}; \theta)$ analytically from the initial density $p_0(\mathbf{z}_0)$ and the Jacobian of the inverse transforms.

Image credits: D. Rezende et al., Variational Inference with Normalizing Flows, ICML 2015.

Latent-variable-based DGMs define the model distribution as a marginal distribution, by introducing a low-dimensional latent random vector:

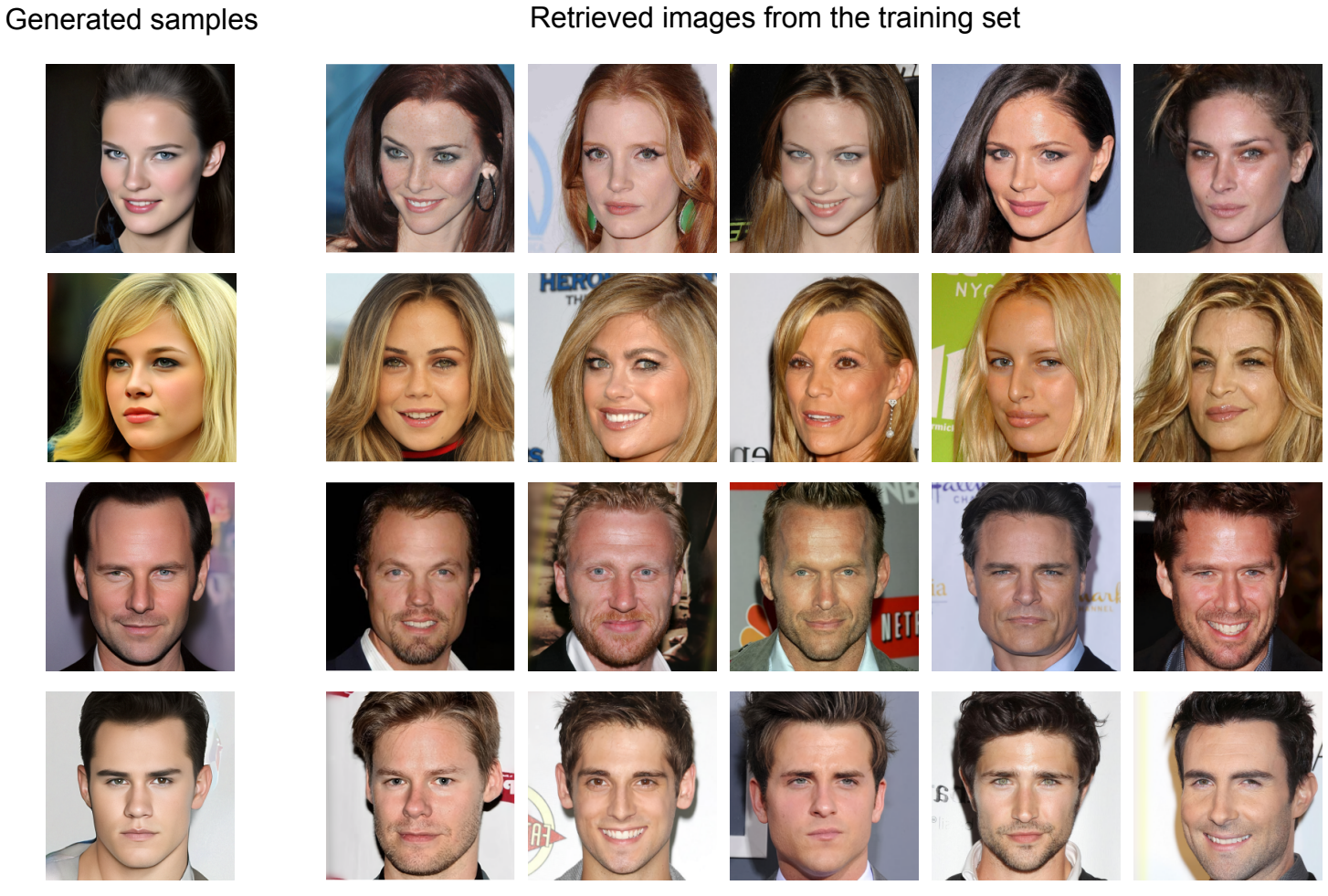
$$p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}.$$



GANs and VAEs are two examples of latent-variable-based DGM, but $p(\mathbf{x}|\mathbf{z};\theta)$ is only defined explicitely (i.e. analytically) for VAEs.

Today we will focus on VAEs.

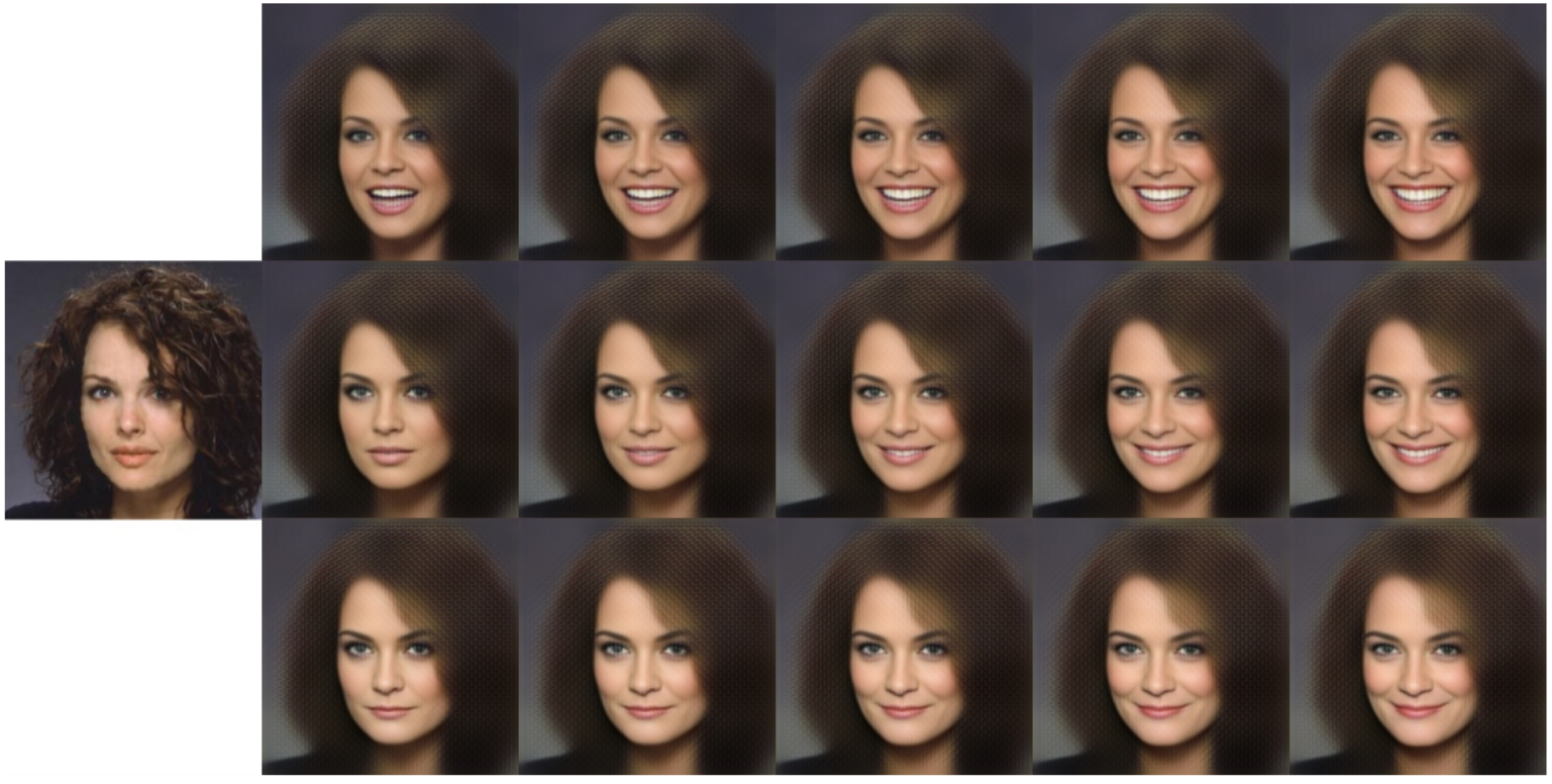
VAE for face generation



Top retrieved images from the training set are visualized for samples generated by NVAE in each row. The generated instances do not exist in the training set.

A. Vahdat and J. Kautz NVAE: A Deep Hierarchical Variational Autoencoder, 2020.

VAE for modifying faces by tuning latent factors



Modifying two latent factors: smiling and mouth open

T. White, Sampling Generative Networks, 2016

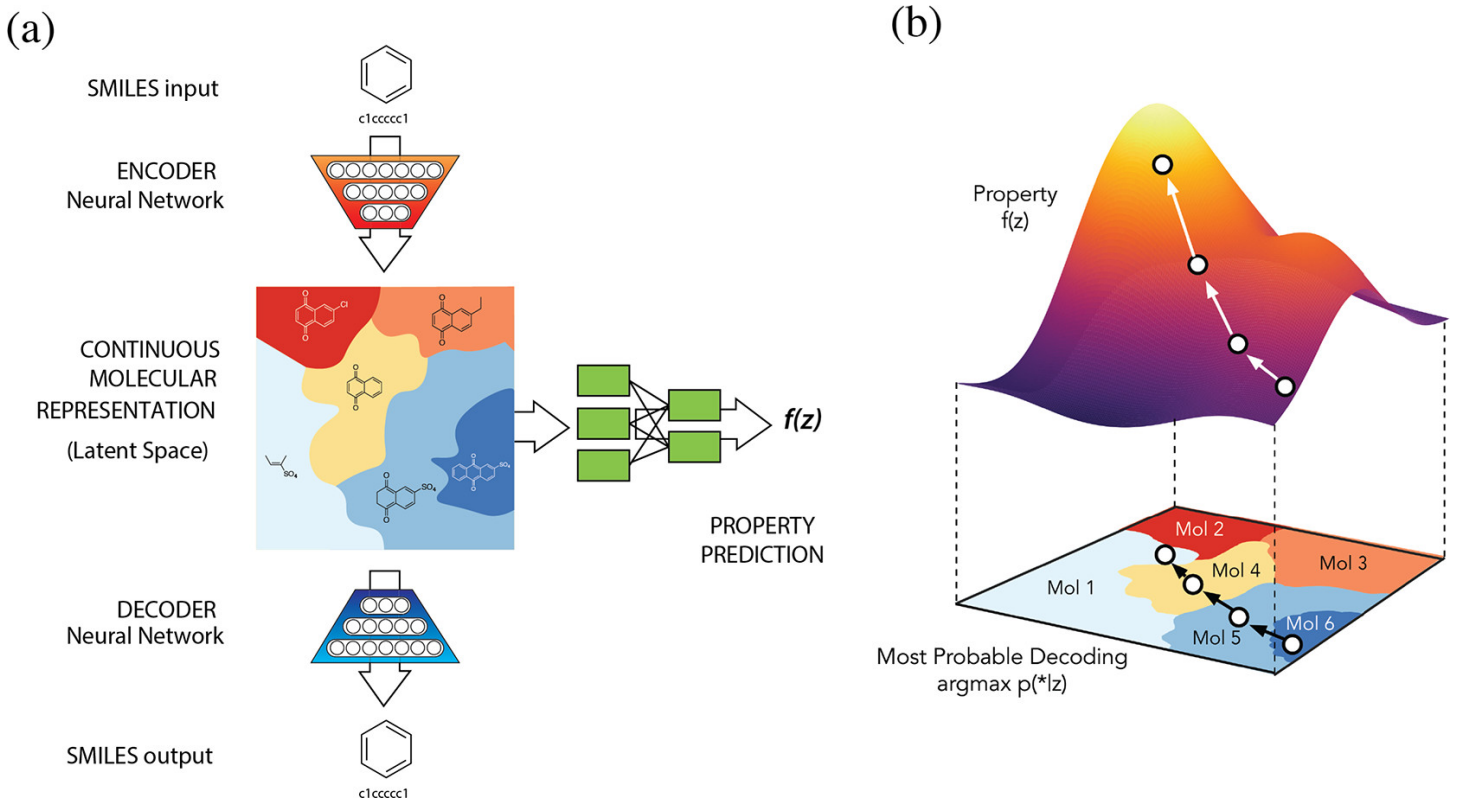
VAE for sentence generation from a continuous space

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Interpolation in a recurrent VAE latent space

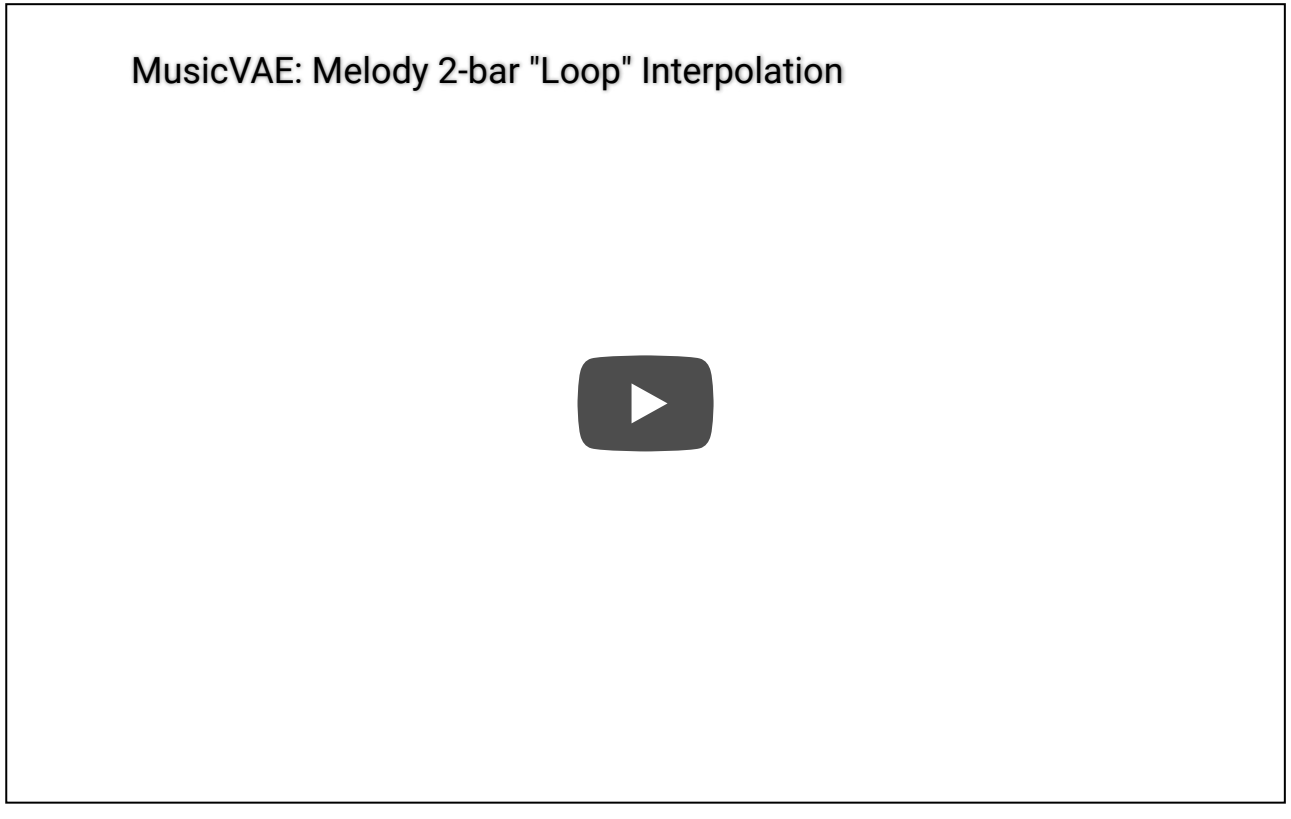
S.R. Bowman et al., Generating Sentences from a Continuous Space, 2015

VAE for automatic design of new molecules



R. Gómez-Bombarelli et al., Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules, 2018

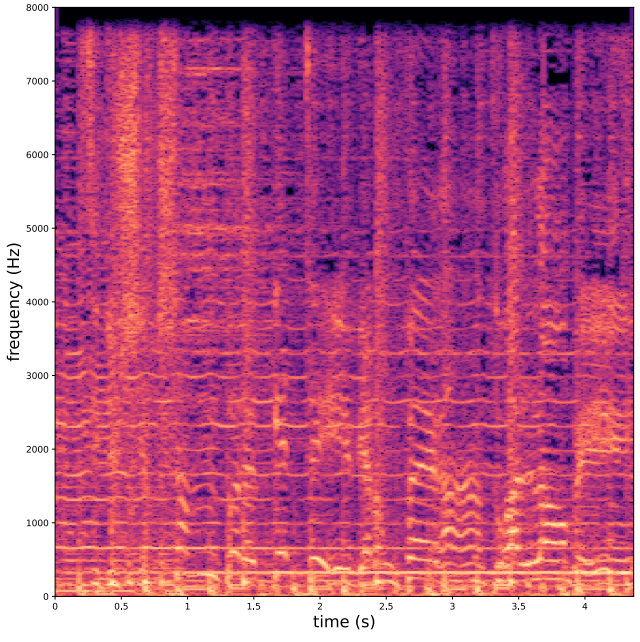
VAE for symbolic music generation

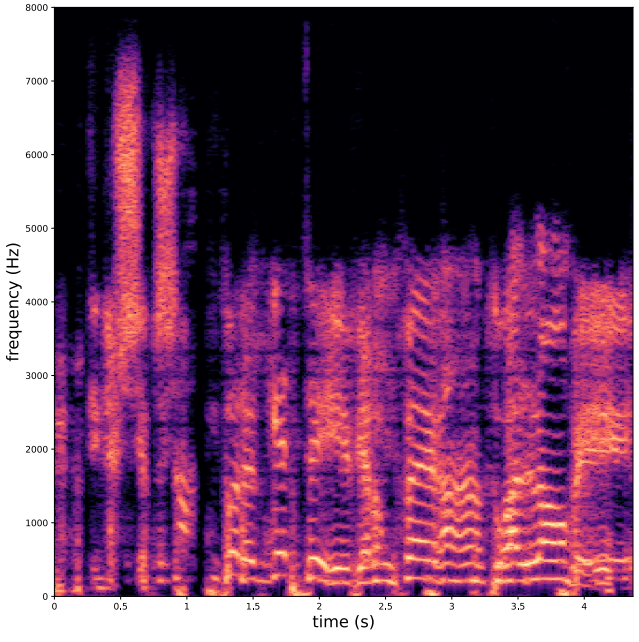


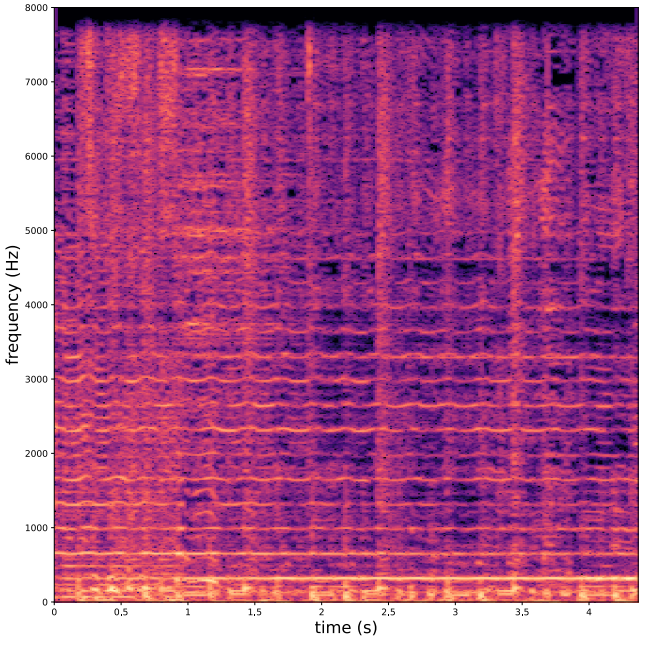
Gradual blending of 2 different melodies.

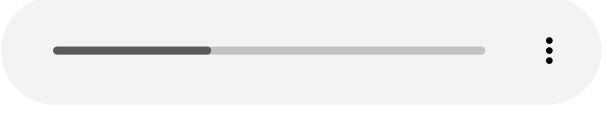
A. Roberts et al., A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music, 2018

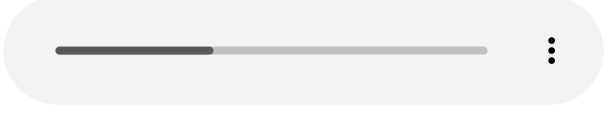
VAE for singing voice separation in music

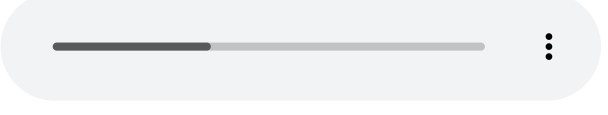












S. L., X. Alameda-Pineda, L. Girin, R. Horaud, A Recurrent Variational Autoencoder for Speech Enhancement, IEEE ICASSP, 2020.

VAE for audio synthesis

- Unconditional generation

Dataset	Generation
Darbouka (stereo)	0:00 / 2:04
VCTK (mono)	0:00 / 0:38

- Strings to speech transfer

original	reconstructed
0:00	0:00

- Speech to strings transfer

original	reconstructed
0:00	0:00

A. Caillon, P. Esling, RAVE: A variational autoencoder for fast and high-quality neural audio synthesis, arXiv, 2021.

Variational autoencoders

D.P. Kingma and M. Welling, Auto-Encoding Variational Bayes, ICLR 2014.

D.J. Rezende et. al, Stochastic backpropagation and approximate inference in deep generative models, ICML 2014.

Generative model

Let $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{z} \in \mathbb{R}^K$ be two random vectors (typically $K \ll D$). The generative model is defined by:

$$p(\mathbf{x}; \theta) = \int p(\mathbf{x}|\mathbf{z}; \theta) p(\mathbf{z}) d\mathbf{z}.$$

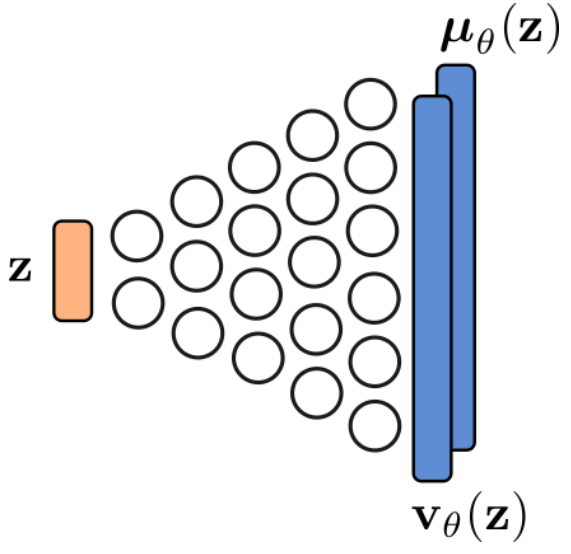
- The prior is a standard Gaussian distribution:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}).$$

- The likelihood is parametrized with a generative/decoder neural network, e.g.

$$p(\mathbf{x}|\mathbf{z}; \theta) = \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z}), \operatorname{diag}\left\{\mathbf{v}_\theta(\mathbf{z})\right\}\right),$$

  where θ denotes the parameters of the decoder network.



Pause on the notebook

How to estimate the VAE generative model parameters?

KL divergence minimization

- We have defined our model distribution $p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$.

KL divergence minimization

- We have defined our model distribution $p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$.
- We want to estimate the model parameters θ so that $p(\mathbf{x};\theta)$ is as close as possible to the true unknown data distribution $p^\star(\mathbf{x})$.

KL divergence minimization

- We have defined our model distribution $p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$.
- We want to estimate the model parameters θ so that $p(\mathbf{x};\theta)$ is as close as possible to the true unknown data distribution $p^\star(\mathbf{x})$.
- We take the Kullback-Leibler (KL) divergence as a measure of fit:

$$D_{\mathrm{KL}}(p \parallel q) = \mathbb{E}_p[\ln(p) - \ln(q)] \geq 0.$$

KL divergence minimization

- We have defined our model distribution $p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$.
- We want to estimate the model parameters θ so that $p(\mathbf{x};\theta)$ is as close as possible to the true unknown data distribution $p^\star(\mathbf{x})$.
- We take the Kullback-Leibler (KL) divergence as a measure of fit:

$$D_{\mathrm{KL}}(p \parallel q) = \mathbb{E}_p[\ln(p) - \ln(q)] \geq 0.$$

- We want to solve:

$$\begin{aligned} & \min_\theta D_{\mathrm{KL}}(p^\star(\mathbf{x}) \parallel p(\mathbf{x};\theta)) \\ \Leftrightarrow \quad & \min_\theta \mathbb{E}_{p^\star(\mathbf{x})}[\ln p^\star(\mathbf{x}) - \ln p(\mathbf{x};\theta)] \\ \Leftrightarrow \quad & \max_\theta \mathbb{E}_{p^\star(\mathbf{x})}[\ln p(\mathbf{x};\theta)] \end{aligned}$$

KL divergence minimization

- We have defined our model distribution $p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z}$.
- We want to estimate the model parameters θ so that $p(\mathbf{x};\theta)$ is as close as possible to the true unknown data distribution $p^\star(\mathbf{x})$.
- We take the Kullback-Leibler (KL) divergence as a measure of fit:

$$D_{\mathrm{KL}}(p \parallel q) = \mathbb{E}_p[\ln(p) - \ln(q)] \geq 0.$$

- We want to solve:

$$\begin{aligned} & \min_\theta D_{\mathrm{KL}}(p^\star(\mathbf{x}) \parallel p(\mathbf{x};\theta)) \\ \Leftrightarrow \quad & \min_\theta \mathbb{E}_{p^\star(\mathbf{x})}[\ln p^\star(\mathbf{x}) - \ln p(\mathbf{x};\theta)] \\ \Leftrightarrow \quad & \max_\theta \mathbb{E}_{p^\star(\mathbf{x})}[\ln p(\mathbf{x};\theta)] \end{aligned}$$

Equivalent to maximum marginal likelihood estimation.

Empirical risk minimization

- We do not have acces to the true data distribution $p^{\star}(\mathbf{x})$...

Empirical risk minimization

- We do not have acces to the true data distribution $p^\star(\mathbf{x})$...
- ... but we have access to a dataset $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}\}_{i=1}^N$ of independent and identically distributed (i.i.d) samples drawn from $p^\star(\mathbf{x})$:

$$\mathbf{x}_n \overset{i.i.d}{\sim} p^\star(\mathbf{x}), \qquad \forall n \in 1, ..., N.$$

Empirical risk minimization

- We do not have acces to the true data distribution $p^\star(\mathbf{x})$...
- ... but we have access to a dataset $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}\}_{i=1}^N$ of independent and identically distributed (i.i.d) samples drawn from $p^\star(\mathbf{x})$:

$$\mathbf{x}_n \overset{i.i.d}{\sim} p^\star(\mathbf{x}), \qquad \forall n \in 1, ..., N.$$

- Monte Carlo estimate: The expectation is approximated by an empirical average, using i.i.d samples drawn from the true intractable distribution,

$$\mathbb{E}_{p^\star(\mathbf{x})}[f(\mathbf{x}; \theta)] \approx \frac{1}{N} \sum_{i=1}^{N} [f(\mathbf{x}_n; \theta)].$$

  The approximation error decreases as N increases (law of large numbers).

Empirical risk minimization

- We do not have acces to the true data distribution $p^\star(\mathbf{x})$...
- ... but we have access to a dataset $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{R}\}_{i=1}^N$ of independent and identically distributed (i.i.d) samples drawn from $p^\star(\mathbf{x})$:

$$\mathbf{x}_n \overset{i.i.d}{\sim} p^\star(\mathbf{x}), \qquad \forall n \in 1, ..., N.$$

- Monte Carlo estimate: The expectation is approximated by an empirical average, using i.i.d samples drawn from the true intractable distribution,

$$\mathbb{E}_{p^\star(\mathbf{x})}[f(\mathbf{x}; \theta)] \approx \frac{1}{N} \sum_{i=1}^{N} [f(\mathbf{x}_n; \theta)].$$

  The approximation error decreases as N increases (law of large numbers).

- This is a general principled called empirical risk minimization in statistical learning theory.

Maximum likelihood estimation

- To sum up, we want to minimize the KL divergence between $p^\star(\mathbf{x})$ and $p(\mathbf{x};\theta)$ w.r.t θ (the weights and biases of the generative network), which is equivalent to solving:

$$\max_\theta \left\{ \mathbb{E}_{p^\star(\mathbf{x})} [\ln p(\mathbf{x};\theta)] \approx \frac{1}{N} \sum_{i=1}^{N} \ln p(\mathbf{x}_n;\theta) \right\} .$$

Maximum likelihood estimation

- To sum up, we want to minimize the KL divergence between $p^\star(\mathbf{x})$ and $p(\mathbf{x};\theta)$ w.r.t θ (the weights and biases of the generative network), which is equivalent to solving:

$$\max_\theta \left\{\mathbb{E}_{p^\star(\mathbf{x})}\left[\ln p(\mathbf{x};\theta)\right] \approx \frac{1}{N}\sum_{i=1}^{N} \ln p(\mathbf{x}_n;\theta)\right\}.$$

- Let's try to develop the expression of the marginal likelihood:

$$p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \int \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu}_\theta(\mathbf{z}),\operatorname{diag}\left\{\mathbf{v}_\theta(\mathbf{z})\right\}\right)\mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})d\mathbf{z}.$$

Maximum likelihood estimation

- To sum up, we want to minimize the KL divergence between $p^\star(\mathbf{x})$ and $p(\mathbf{x};\theta)$ w.r.t θ (the weights and biases of the generative network), which is equivalent to solving:

$$\max_\theta \left\{\mathbb{E}_{p^\star(\mathbf{x})}\left[\ln p(\mathbf{x};\theta)\right] \approx \frac{1}{N}\sum_{i=1}^{N} \ln p(\mathbf{x}_n;\theta)\right\}.$$

- Let's try to develop the expression of the marginal likelihood:

$$p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \int \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu}_\theta(\mathbf{z}),\operatorname{diag}\left\{\mathbf{v}_\theta(\mathbf{z})\right\}\right)\mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})d\mathbf{z}.$$

- We cannot compute this integral analytically, because the integrand is highly non-linear in $\mathbf{z}$.

Maximum likelihood estimation

- To sum up, we want to minimize the KL divergence between $p^\star(\mathbf{x})$ and $p(\mathbf{x};\theta)$ w.r.t θ (the weights and biases of the generative network), which is equivalent to solving:

$$\max_\theta \left\{\mathbb{E}_{p^\star(\mathbf{x})}\left[\ln p(\mathbf{x};\theta)\right] \approx \frac{1}{N}\sum_{i=1}^{N} \ln p(\mathbf{x}_n;\theta)\right\}.$$

- Let's try to develop the expression of the marginal likelihood:

$$p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})d\mathbf{z} = \int \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu}_\theta(\mathbf{z}),\mathrm{diag}\left\{\mathbf{v}_\theta(\mathbf{z})\right\}\right)\mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})d\mathbf{z}.$$

- We cannot compute this integral analytically, because the integrand is highly non-linear in $\mathbf{z}$.
- We will resort to variational inference techniques, that we have already encountered in this course.

Variational inference

- For any distribution $q(\mathbf{z}|\mathbf{x};\phi)$, we have (Neal and Hinton, 1999; Jordan et al. 1999)

$$\ln p(\mathbf{x};\theta) = \mathcal{L}(\mathbf{x};\phi,\theta) + D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z}|\mathbf{x};\theta)),$$

  where $\mathcal{L}(\mathbf{x};\phi,\theta)$ is the evidence lower bound (ELBO), defined by

$$\mathcal{L}(\mathbf{x};\phi,\theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x},\mathbf{z};\theta) - \ln q(\mathbf{z}|\mathbf{x};\phi)].$$

R.M. Neal and G.E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants", in M. I. Jordan (Ed.), *Learning in graphical models*, Cambridge, MA: MIT Press, 1999.

M.I. Jordan et al., "An introduction to variational methods for graphical models", Machine learning, 1999.

Variational inference

- For any distribution $q(\mathbf{z}|\mathbf{x};\phi)$, we have (Neal and Hinton, 1999; Jordan et al. 1999)

$$\ln p(\mathbf{x};\theta) = \mathcal{L}(\mathbf{x};\phi,\theta) + D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z}|\mathbf{x};\theta)),$$

  where $\mathcal{L}(\mathbf{x};\phi,\theta)$ is the evidence lower bound (ELBO), defined by

$$\mathcal{L}(\mathbf{x};\phi,\theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x},\mathbf{z};\theta) - \ln q(\mathbf{z}|\mathbf{x};\phi)].$$

Problem #1

$$\max_{\theta} \mathcal{L}(\mathbf{x};\phi,\theta),$$

where $\mathcal{L}(\mathbf{x};\phi,\theta) \leq \ln p(\mathbf{x};\theta)$

R.M. Neal and G.E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants", in M. I. Jordan (Ed.), *Learning in graphical models*, Cambridge, MA: MIT Press, 1999.

M.I. Jordan et al., "An introduction to variational methods for graphical models", Machine learning, 1999.

Variational inference

- For any distribution $q(\mathbf{z}|\mathbf{x};\phi)$, we have (Neal and Hinton, 1999; Jordan et al. 1999)

$$\ln p(\mathbf{x};\theta) = \mathcal{L}(\mathbf{x};\phi,\theta) + D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z}|\mathbf{x};\theta)),$$

  where $\mathcal{L}(\mathbf{x};\phi,\theta)$ is the evidence lower bound (ELBO), defined by

$$\mathcal{L}(\mathbf{x};\phi,\theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x},\mathbf{z};\theta) - \ln q(\mathbf{z}|\mathbf{x};\phi)].$$

Problem #1

$$\max_{\theta} \mathcal{L}(\mathbf{x};\phi,\theta),$$

where $\mathcal{L}(\mathbf{x};\phi,\theta) \leq \ln p(\mathbf{x};\theta)$

Problem #2

$$\max_{\phi} \mathcal{L}(\mathbf{x};\phi,\theta)$$

$$\Leftrightarrow \min_{\phi} D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z}|\mathbf{x};\theta))$$

R.M. Neal and G.E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants", in M. I. Jordan (Ed.), *Learning in graphical models*, Cambridge, MA: MIT Press, 1999.

M.I. Jordan et al., "An introduction to variational methods for graphical models", Machine learning, 1999.

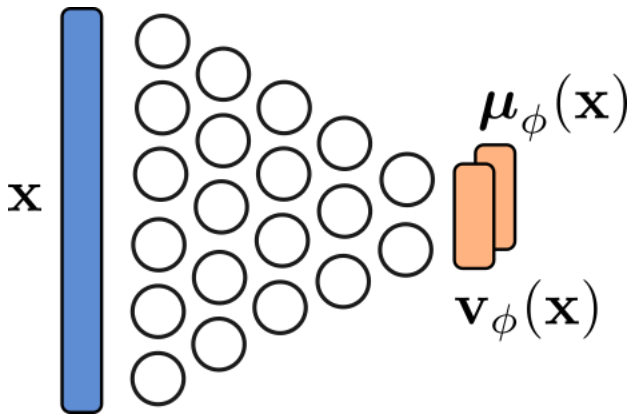
To fully define the objective function, we need to specify the inference model $q(\mathbf{z}|\mathbf{x};\phi)$.

Inference model

The inference model (approximate posterior) is typically defined by:

$$q(\mathbf{z}|\mathbf{x};\phi) = \mathcal{N}\left(\mathbf{z};\boldsymbol{\mu}_\phi(\mathbf{x}), \operatorname{diag}\left\{\mathbf{v}_\phi(\mathbf{x})\right\}\right),$$

where the mean and variance vectors are provided by the encoder neural network.



ELBO

The ELBO is now fully defined:

$$\begin{aligned}\mathcal{L}(\mathbf{x};\phi,\theta) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x},\mathbf{z};\theta) - \ln q(\mathbf{z}|\mathbf{x};\phi)\right] \\ &= \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x}|\mathbf{z};\theta)\right]}_{\text{reconstruction accuracy}} - \underbrace{D_{\mathrm{KL}}\left(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z})\right)}_{\text{regularization}}.\end{aligned}$$

- prior: $p(\mathbf{z}) = \mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})$
- likelihood model: $p(\mathbf{x}|\mathbf{z};\theta) = \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu}_\theta(\mathbf{z}),\operatorname{diag}\left\{\mathbf{v}_\theta(\mathbf{z})\right\}\right)$
- inference model: $q(\mathbf{z}|\mathbf{x};\phi) = \mathcal{N}\left(\mathbf{z};\boldsymbol{\mu}_\phi(\mathbf{x}),\operatorname{diag}\left\{\mathbf{v}_\phi(\mathbf{x})\right\}\right)$

Pause on the notebook

Back to the computation of the ELBO

$$\mathcal{L}(\mathbf{x}; \phi, \theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x}|\mathbf{z};\theta)] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z})).$$

Let's first focus on the second term of the ELBO which is called the regularization term because it constrains the inference model to be not too far from the prior.

$$\begin{aligned}
D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z})) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln q(\mathbf{z}|\mathbf{x};\phi) - \ln p(\mathbf{z})\right] \\
&= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}\left\{\mathbf{v}_\phi(\mathbf{x})\right\}\right) - \ln \mathcal{N}\left(\mathbf{z}; \mathbf{0}, \mathbf{I}\right)\right] \\
&= \sum_{k=1}^{K} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln \mathcal{N}\left(z_k; \mu_{k,\phi}(\mathbf{x}), v_{k,\phi}(\mathbf{x})\right) - \ln \mathcal{N}\left(z_k; 0, 1\right)\right] \\
&= \dots
\end{aligned}$$

$$\mathcal{L}(\mathbf{x};\phi,\theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x}|\mathbf{z};\theta)] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z})).$$

Let's now focus on the second term, which is called the reconstruction accuracy term.

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x}|\mathbf{z};\theta)] &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu}_\theta(\mathbf{z}), \operatorname{diag}\{\mathbf{v}_\theta(\mathbf{z})\}\right)\right] \\
&= \sum_{d=1}^{D} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln \mathcal{N}\left(x_d;\mu_{d,\theta}(\mathbf{z}), v_{d,\theta}(\mathbf{z})\right)\right] \\
&= -\frac{1}{2}\sum_{d=1}^{D} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln\left(2\pi\, v_{d,\theta}(\mathbf{z})\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z})\right)^2}{v_{d,\theta}(\mathbf{z})}]
\end{aligned}$$

$$\mathcal{L}(\mathbf{x};\phi,\theta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x}|\mathbf{z};\theta)\right] - D_{\mathrm{KL}}\left(q(\mathbf{z}|\mathbf{x};\phi) \parallel p(\mathbf{z})\right).$$

Let's now focus on the second term, which is called the reconstruction accuracy term.

$$\begin{aligned}
\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x}|\mathbf{z};\theta)\right] &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln \mathcal{N}\left(\mathbf{x};\boldsymbol{\mu}_\theta(\mathbf{z}), \operatorname{diag}\left\{\mathbf{v}_\theta(\mathbf{z})\right\}\right)\right] \\
&= \sum_{d=1}^{D} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln \mathcal{N}\left(x_d;\mu_{d,\theta}(\mathbf{z}), v_{d,\theta}(\mathbf{z})\right)\right] \\
&= -\frac{1}{2}\sum_{d=1}^{D} \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln\left(2\pi\, v_{d,\theta}(\mathbf{z})\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z})\right)^2}{v_{d,\theta}(\mathbf{z})}\right]
\end{aligned}$$

A common approach consist in choosing $v_{d,\theta}(\mathbf{z}) = 1$ for all $d \in \{1, \ldots, D\}$, such that the reconstruction accuracy term involves the mean squared error between the data $\mathbf{x}$ and the mean vector $\boldsymbol{\mu}_\theta(\mathbf{z})$ provided by the decoder network.

Reconstruction accuracy term

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x}|\mathbf{z};\theta)] = -\frac{1}{2}\sum_{d=1}^{D}\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln\left(2\pi\, v_{d,\theta}(\mathbf{z})\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z})\right)^2}{v_{d,\theta}(\mathbf{z})}].$$

Reconstruction accuracy term

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x}|\mathbf{z};\theta)] = -\frac{1}{2}\sum_{d=1}^{D}\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln\left(2\pi\, v_{d,\theta}(\mathbf{z})\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z})\right)^2}{v_{d,\theta}(\mathbf{z})}].$$

- **Problem 1** - The expectation cannot be computed analytically.

Reconstruction accuracy term

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x}|\mathbf{z};\theta)] = -\frac{1}{2}\sum_{d=1}^{D}\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln\left(2\pi\, v_{d,\theta}(\mathbf{z})\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z})\right)^2}{v_{d,\theta}(\mathbf{z})}].$$

- **Problem 1** - The expectation cannot be computed analytically.
- **Solution 1** - Approximate it with a Monte Carlo estimate:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\ln p(\mathbf{x}|\mathbf{z};\theta)] \approx -\frac{1}{2R}\sum_{d=1}^{D}\sum_{r=1}^{R}[\ln\left(2\pi\, v_{d,\theta}(\mathbf{z}_r)\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z}_r)\right)^2}{v_{d,\theta}(\mathbf{z}_r)}],$$

where $\{\mathbf{z}_r\}_{r=1}^{R}$ are i.i.d samples drawn from $q(\mathbf{z}|\mathbf{x};\phi)$.

Note that in practice, we choose $R = 1$.

Reconstruction accuracy term

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x}|\mathbf{z};\theta)\right] = -\frac{1}{2}\sum_{d=1}^{D}\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\,\ln\left(2\pi\, v_{d,\theta}(\mathbf{z})\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z})\right)^2}{v_{d,\theta}(\mathbf{z})}].$$

- **Problem 1** - The expectation cannot be computed analytically.
- **Solution 1** - Approximate it with a Monte Carlo estimate:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x}|\mathbf{z};\theta)\right] \approx -\frac{1}{2R}\sum_{d=1}^{D}\sum_{r=1}^{R}[\,\ln\left(2\pi\, v_{d,\theta}(\mathbf{z}_r)\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z}_r)\right)^2}{v_{d,\theta}(\mathbf{z}_r)}],$$

  where $\{\mathbf{z}_r\}_{r=1}^{R}$ are i.i.d samples drawn from $q(\mathbf{z}|\mathbf{x};\phi)$.

  Note that in practice, we choose $R = 1$.

- **Problem 2** - This sampling operation is not differentiable w.r.t. ϕ.

Reconstruction accuracy term

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x}|\mathbf{z};\theta)\right] = -\frac{1}{2}\sum_{d=1}^{D}\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\,\ln\left(2\pi\, v_{d,\theta}(\mathbf{z})\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z})\right)^2}{v_{d,\theta}(\mathbf{z})}].$$

- **Problem 1** - The expectation cannot be computed analytically.
- **Solution 1** - Approximate it with a Monte Carlo estimate:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}\left[\ln p(\mathbf{x}|\mathbf{z};\theta)\right] \approx -\frac{1}{2R}\sum_{d=1}^{D}\sum_{r=1}^{R}[\,\ln\left(2\pi\, v_{d,\theta}(\mathbf{z}_r)\right) + \frac{\left(x_d - \mu_{d,\theta}(\mathbf{z}_r)\right)^2}{v_{d,\theta}(\mathbf{z}_r)}],$$

  where $\{\mathbf{z}_r\}_{r=1}^{R}$ are i.i.d samples drawn from $q(\mathbf{z}|\mathbf{x};\phi)$.

  Note that in practice, we choose $R = 1$.

- **Problem 2** - This sampling operation is not differentiable w.r.t. ϕ.
- **Solution 2** - Use the so-called **reparametrization trick**.

Reparametrization trick

- The reparameterization trick consists in rewritting the sampling operation of $\mathbf{z}_r$ from $q(\mathbf{z}|\mathbf{x};\phi)$ as a invertible transformation of another random sample $\boldsymbol{\epsilon}_r$, drawn from a distribution $p(\boldsymbol{\epsilon})$ which is independent of $\mathbf{x}$ and ϕ.

Reparametrization trick

- The reparameterization trick consists in rewritting the sampling operation of $\mathbf{z}_r$ from $q(\mathbf{z}|\mathbf{x};\phi)$ as a invertible transformation of another random sample $\boldsymbol{\epsilon}_r$, drawn from a distribution $p(\boldsymbol{\epsilon})$ which is independent of $\mathbf{x}$ and ϕ.
- In our case, we have $q(\mathbf{z}|\mathbf{x};\phi) = \mathcal{N}\left(\boldsymbol{\mu}_\phi(\mathbf{x}), \operatorname{diag}\{\mathbf{v}_\phi(\mathbf{x})\}\right)$.

Reparametrization trick

- The reparameterization trick consists in rewritting the sampling operation of $\mathbf{z}_r$ from $q(\mathbf{z}|\mathbf{x};\phi)$ as a invertible transformation of another random sample $\boldsymbol{\epsilon}_r$, drawn from a distribution $p(\boldsymbol{\epsilon})$ which is independent of $\mathbf{x}$ and ϕ.
- In our case, we have $q(\mathbf{z}|\mathbf{x};\phi) = \mathcal{N}\left(\boldsymbol{\mu}_\phi(\mathbf{x}), \operatorname{diag}\{\mathbf{v}_\phi(\mathbf{x})\}\right)$.
- A simple reparametrization is given by:

$$\mathbf{z}_r = \boldsymbol{\mu}_\phi(\mathbf{x}) + \operatorname{diag}\{\mathbf{v}_\phi(\mathbf{x})\}^{1/2} \boldsymbol{\epsilon}_r,$$

  where $\boldsymbol{\epsilon}_r$ is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Finally, the full ELBO

- Putting all together, we end-up with the following expression of the variational free energy:

$$\begin{aligned}\mathcal{L}(\mathbf{x};\phi,\theta) = & -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}})\right)+\frac{\left(x_d-\mu_{d,\theta}(\tilde{\mathbf{z}})\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}})}] \\ & +\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x})-\mu_{k,\phi}^2(\mathbf{x})-v_{k,\phi}(\mathbf{x})\right]+cst(\phi,\theta),\end{aligned}$$

where $\tilde{\mathbf{z}}$ is drawn from $q(\mathbf{z}|\mathbf{x};\phi)$ using the **reparametrization trick**.

Finally, the full ELBO

- Putting all together, we end-up with the following expression of the variational free energy:

$$\begin{aligned}\mathcal{L}(\mathbf{x};\phi,\theta) = & -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}})\right)+\frac{\left(x_d-\mu_{d,\theta}(\tilde{\mathbf{z}})\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}})}] \\ & +\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x})-\mu_{k,\phi}^2(\mathbf{x})-v_{k,\phi}(\mathbf{x})\right]+cst(\phi,\theta),\end{aligned}$$

  where $\tilde{\mathbf{z}}$ is drawn from $q(\mathbf{z}|\mathbf{x};\phi)$ using the **reparametrization trick**.

- This objective function is differentiable with respect to ϕ and θ. It can be optimized with gradient-ascent-based techniques, where gradients are computed by backpropagation.

Finally, the full ELBO

- Putting all together, we end-up with the following expression of the variational free energy:

$$\begin{aligned}\mathcal{L}(\mathbf{x};\phi,\theta) = & -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}})\right)+\frac{\left(x_d-\mu_{d,\theta}(\tilde{\mathbf{z}})\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}})}] \\ & +\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x})-\mu_{k,\phi}^2(\mathbf{x})-v_{k,\phi}(\mathbf{x})\right]+cst(\phi,\theta),\end{aligned}$$

  where $\tilde{\mathbf{z}}$ is drawn from $q(\mathbf{z}|\mathbf{x};\phi)$ using the **reparametrization trick**.

- This objective function is differentiable with respect to ϕ and θ. It can be optimized with gradient-ascent-based techniques, where gradients are computed by backpropagation.

- Do not forget that we actually maximize a sample average over our training dataset:

$$\max_{\phi,\theta}\left\{\mathbb{E}_{p^\star(\mathbf{x})}\left[\mathcal{L}(\mathbf{x};\phi,\theta)\right]\approx\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(\mathbf{x}_n;\phi,\theta)\right\}.$$

Training procedure

Step 0: Do the math

$$\begin{aligned}\mathcal{L}(\mathbf{x}_n;\phi,\theta) = & -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}}_n)\right)+\frac{\left(x_{n,d}-\mu_{d,\theta}(\tilde{\mathbf{z}}_n)\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}}_n)}] \\ & +\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x}_n)-\mu_{k,\phi}^2(\mathbf{x}_n)-v_{k,\phi}(\mathbf{x}_n)\right]+cst(\phi,\theta),\end{aligned}$$

Training procedure

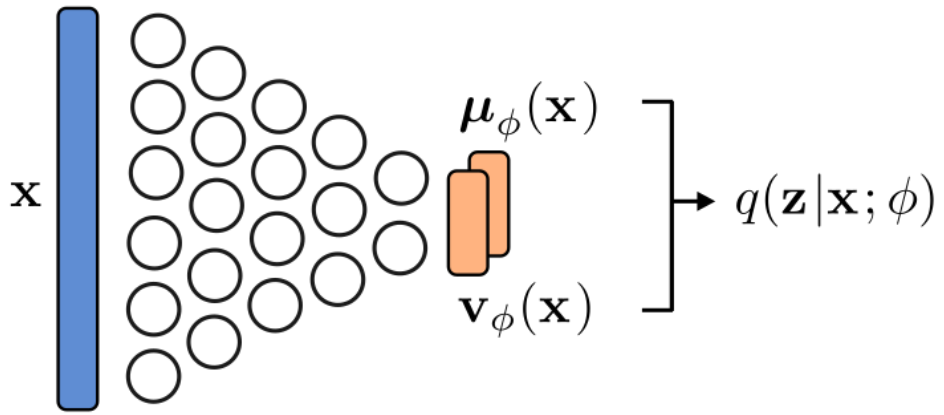
Step 1: Pick an example in the dataset

$$\mathcal{L}(\mathbf{x}_n;\phi,\theta) = -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}}_n)\right) + \frac{\left(x_{n,d}-\mu_{d,\theta}(\tilde{\mathbf{z}}_n)\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}}_n)}]$$
$$+\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x}_n) - \mu_{k,\phi}^2(\mathbf{x}_n) - v_{k,\phi}(\mathbf{x}_n)\right] + cst(\phi,\theta),$$

Training procedure

Step 2: Map through the encoder

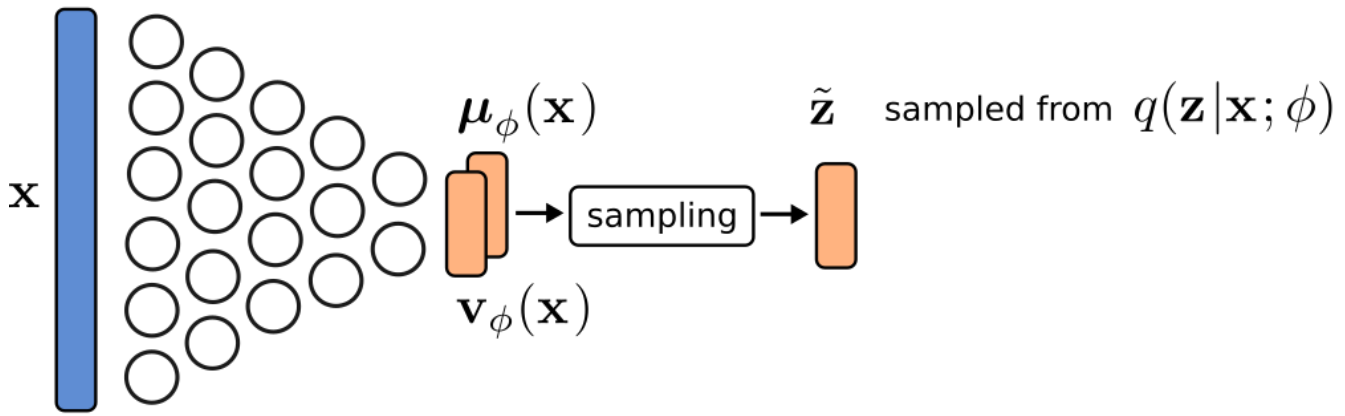
$$\mathcal{L}(\mathbf{x}_n;\phi,\theta) = -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}}_n)\right) + \frac{\left(x_{n,d}-\mu_{d,\theta}(\tilde{\mathbf{z}}_n)\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}}_n)}]$$
$$+\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x}_n) - \mu_{k,\phi}^2(\mathbf{x}_n) - v_{k,\phi}(\mathbf{x}_n)\right] + cst(\phi,\theta),$$



Training procedure

Step 3: Sample

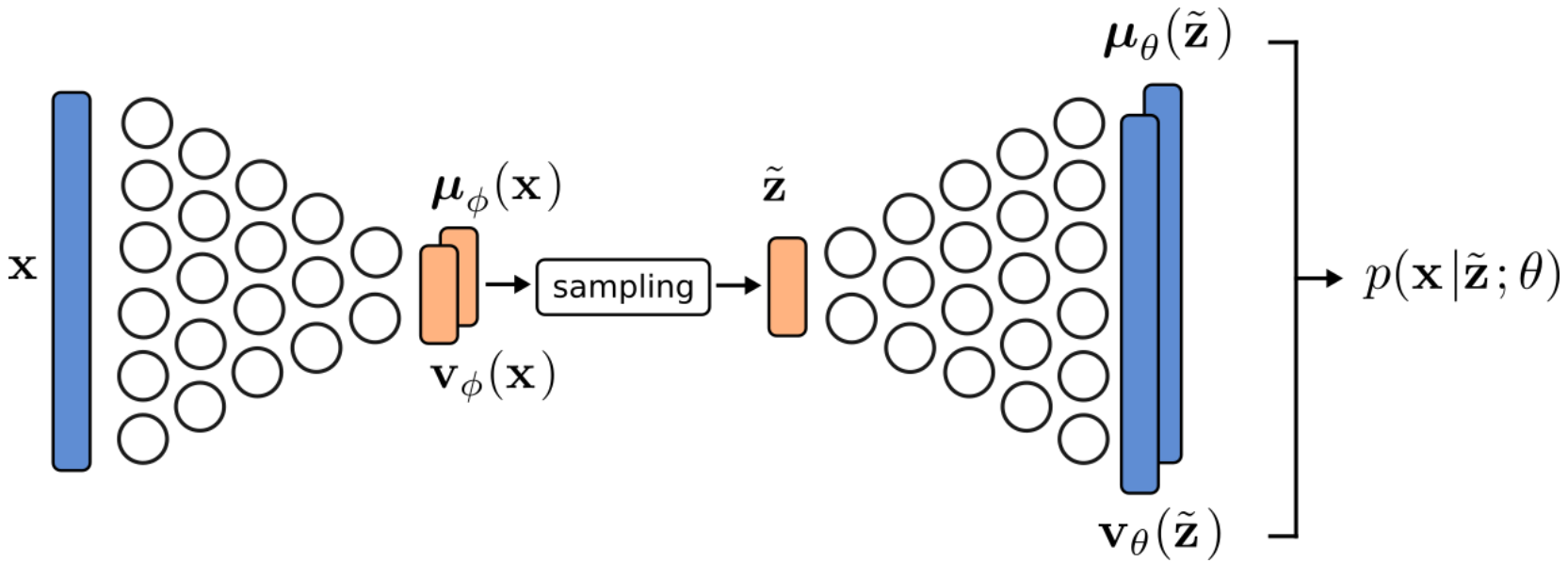
$$\mathcal{L}(\mathbf{x}_n;\phi,\theta) = -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}}_n)\right) + \frac{\left(x_{n,d}-\mu_{d,\theta}(\tilde{\mathbf{z}}_n)\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}}_n)}]$$
$$+\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x}_n) - \mu_{k,\phi}^2(\mathbf{x}_n) - v_{k,\phi}(\mathbf{x}_n)\right] + cst(\phi,\theta),$$



Training procedure

Step 4: Map through the decoder

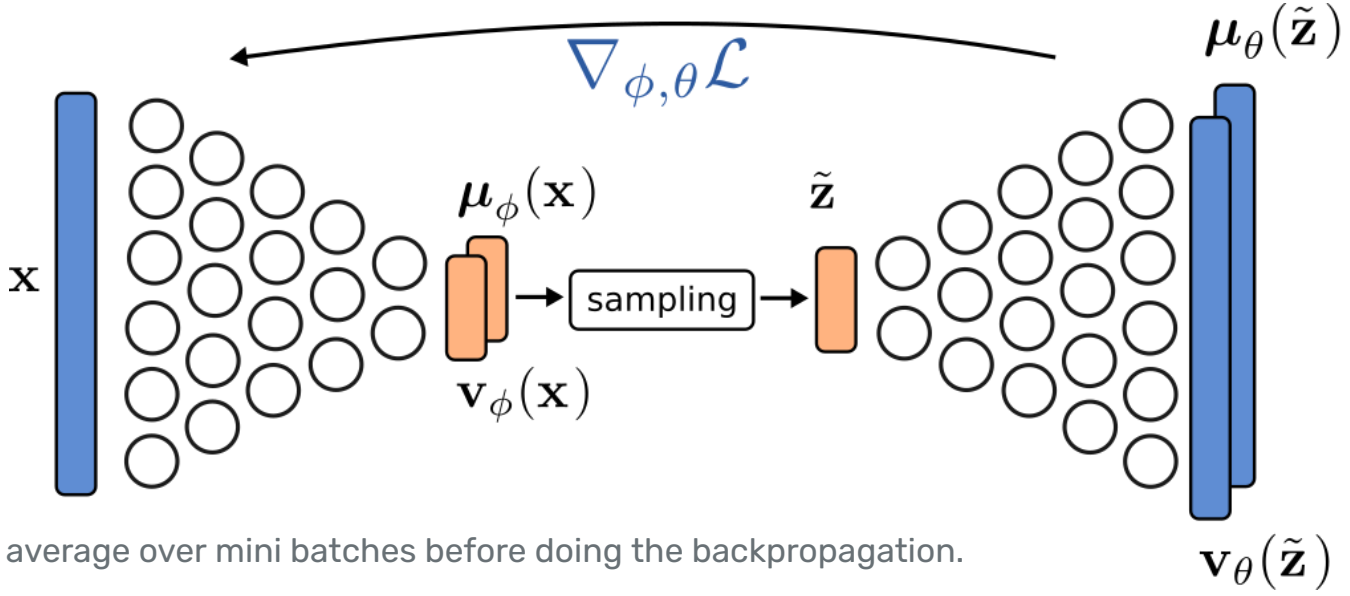
$$\mathcal{L}(\mathbf{x}_n;\phi,\theta) = -\frac{1}{2}\sum_{d=1}^{D}[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}}_n)\right) + \frac{\left(x_{n,d}-\mu_{d,\theta}(\tilde{\mathbf{z}}_n)\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}}_n)}]$$
$$+\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x}_n) - \mu_{k,\phi}^2(\mathbf{x}_n) - v_{k,\phi}(\mathbf{x}_n)\right] + cst(\phi,\theta),$$



Training procedure

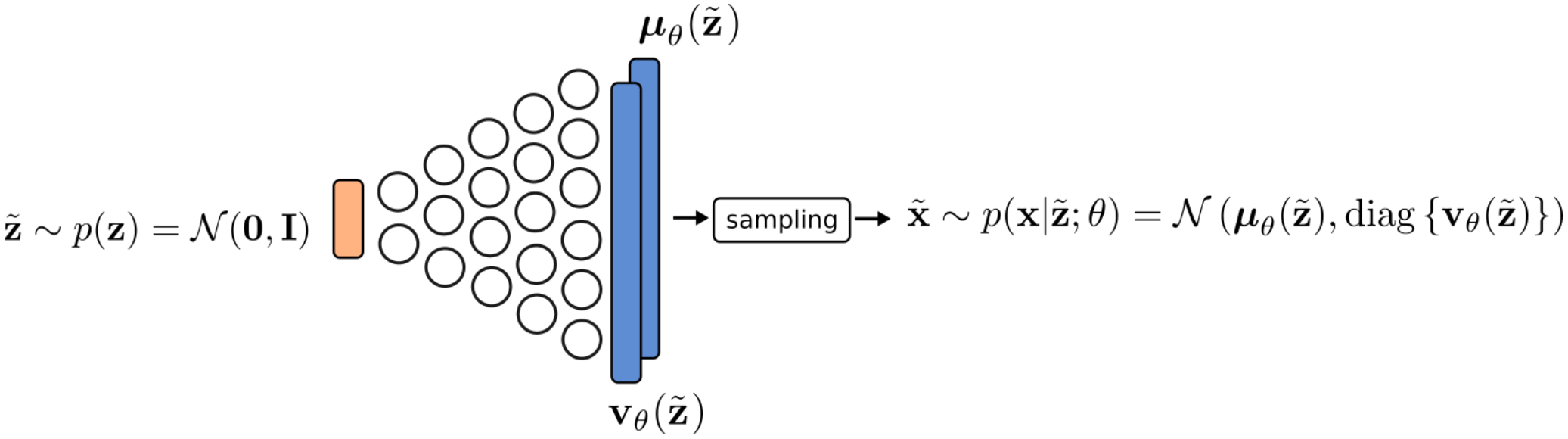
Step 5: Backpropagate and gradient step

$$\mathcal{L}(\mathbf{x}_n;\phi,\theta) = -\frac{1}{2}\sum_{d=1}^{D}\left[\ln\left(v_{d,\theta}(\tilde{\mathbf{z}}_n)\right) + \frac{\left(x_{n,d}-\mu_{d,\theta}(\tilde{\mathbf{z}}_n)\right)^2}{v_{d,\theta}(\tilde{\mathbf{z}}_n)}\right]$$
$$+\frac{1}{2}\sum_{k=1}^{K}\left[\ln v_{k,\phi}(\mathbf{x}_n) - \mu_{k,\phi}^2(\mathbf{x}_n) - v_{k,\phi}(\mathbf{x}_n)\right] + cst(\phi,\theta),$$



You can also average over mini batches before doing the backpropagation.

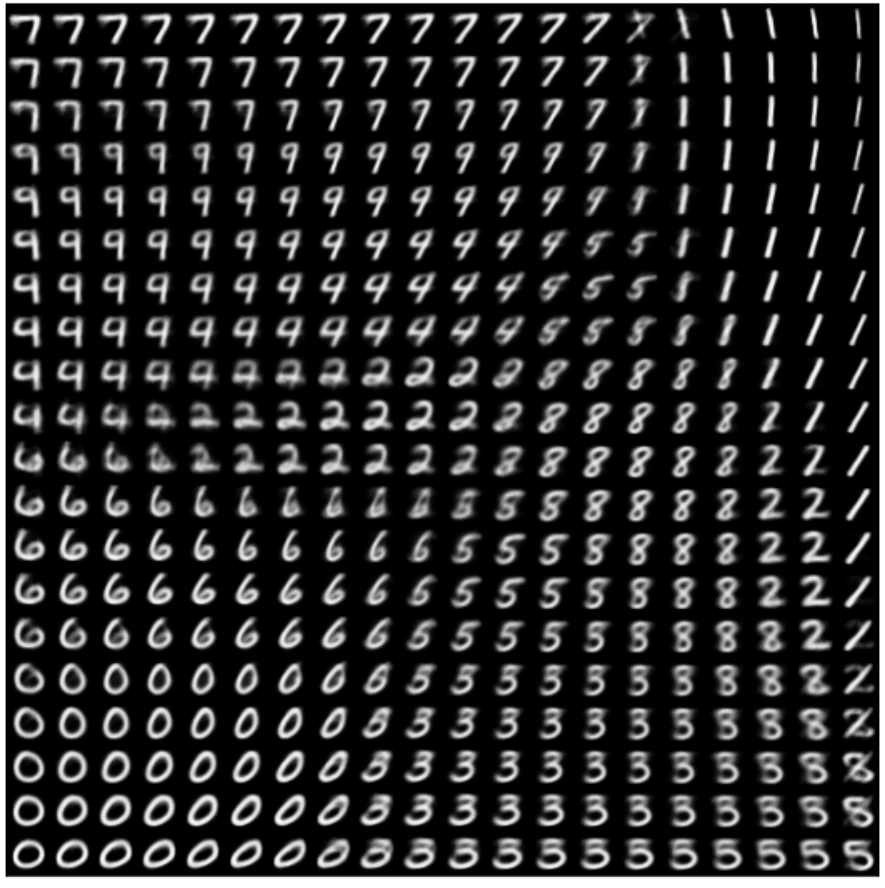
At test time (after training)



- Note that the encoder was only introduced in order to estimate the parameters of the decoder.
- We do not need the encoder for generating new samples.
- But it is useful if we need to do inference.

Generated MNIST samples

VAE with $K = 16$



Linearly spaced coordinates on the unit square are transformed through the inverse cumulative distribution function of the Gaussian to produce values of the latent variables which are then mapped through the decoder network.

Practical session