Generative Latent Variable Models: The Variational Autoencoder

Danilo Comminiello

Generative Deep Learning 2021/2022

June 13, 2022

Sapienza
Università di Roma

Lecture content highlights

- The unsupervised learning of latent representations of an observation dataset is a key point of a successful generation process.
- The variational autoencoder (VAE) is a generative deep learning method, which involves an encoder to learn the deep latent variables and a decoder to generate new data.
- The inference model of the VAE is parameterized by using deep neural networks.
- A differentiable loss function for the VAE is achieved by using a reparameterization trick.

Table of contents

1 Latent Variable Models

Generating by latent variables

Previously, we discussed generative autoregressive models, whose approach to learn $p(\mathbf{x})$ is to *directly* model the likelihood function by parameterizing conditional distributions.

Now we discuss generative models with a different approach based on latent variables.

Let us suppose that we have a collection of images with horses and we want to learn $p(\mathbf{x})$ to generate new images [1].

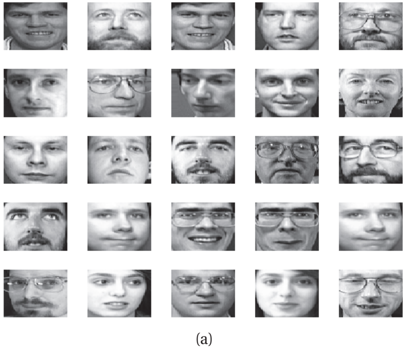
As a first step, we can ask ourselves *how* we should generate a horse.

There are some *factors* in data (e.g., a *silhouette*, a *color*, a *background*) that are crucial for generating an object (here, a horse).

Once we decide about these factors, we can generate them by adding details. Like painting.

Extracting latent factors from data

Although data size may appear rather high, there may only be a small number of degrees of variability, corresponding to **latent factors**.



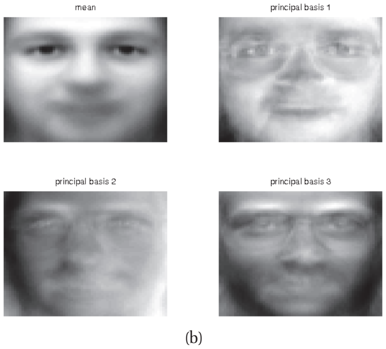


Figure 1: As an example, when modeling the appearance of face images, there may only be a few underlying latent factors which describe *most of the variability*, such as lighting, pose, identity, etc. Here we have (a) 25 randomly chosen 64×64 pixel images from the Olivetti face database. (b) The mean and the first three principal component basis vectors (eigenfaces) [2].

Low-dimensional latent space

We can denote the collection of images as $\mathbf{x} \in \mathcal{X}^D$ and the low-dimensional latent variables as $\mathbf{z} \in \mathcal{Z}^M$, where $\mathcal{Z}^M$ represents the *manifold*.

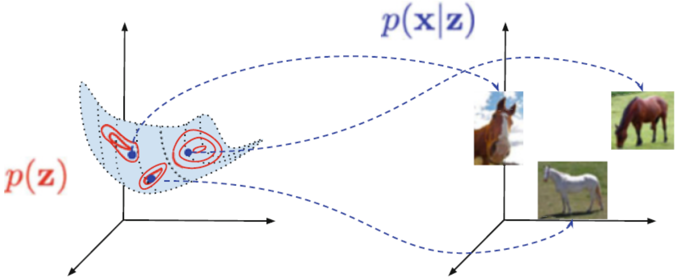


Figure 2: Representation of a latent variable model $\mathbf{z} \sim p(\mathbf{z})$ (in red) and the generative process $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$ (in blue) [1].

In plain words, we first sample $\mathbf{z}$ (e.g., size, shape, and color of a horse) and then create an image with all necessary details, i.e., we sample $\mathbf{x}$ from the conditional distribution $p(\mathbf{x}|\mathbf{z})$.

Factorization of latent variable models

Given the latent variables $\mathbf{z}$, the joint distribution is factorized as:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})$$

For training we have access only to $\mathbf{x}$, thus we should *sum out* the unknown, i.e., $\mathbf{z}$, by considering the marginal likelihood function:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}. \tag{1}$$

A natural question now is how to calculate this integral.

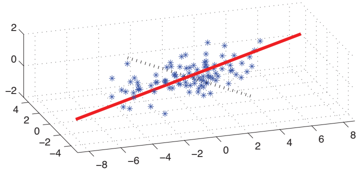
- The easiest case is that the integral is tractable.
- If not tractable, we need to compute it by utilizing a specific *approximate inference*, known as variational inference.

2 Probabilistic Principal Component Analysis

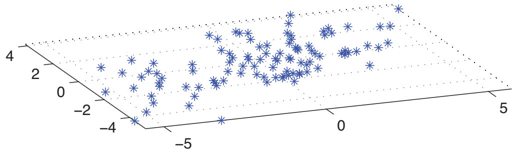
Projecting data in a lower-dimensional latent space

As a first step, we need a latent variable model that projects high-dimensional data $\mathbf{x}$ in a lower-dimensional latent space.

The dimensionality reduction goal is to identify manifold structures and capture the “essence” of the data.



(a)



(b)

Figure 3: A simple example may be represented by the projection of some 3D data down to a 2D plane. The 2D approximation is quite good, since most points lie close to this subspace. Reducing to 1D may result in a rather poor approximation. (a) A set of points that live on a 2D linear subspace embedded in 3D. The solid red line is the first principal component direction. The dotted black line is the second PC direction. (b) 2D representation of the data [2].

Principal component analysis

One of the most popular dimensionality reduction methods is called **principal component analysis** (PCA) or *Karhunen-Loève transform*.

Observed data are generated by a system or process that is driven by a (relatively) small number of *latent* (not directly observed) variables.

The goal of PCA, as any dimensionality reduction methods, is to learn this latent structure.

PCA can be thought of as an unsupervised version of linear regression, where we observe the high-dimensional response $\mathbf{y}$, but not the low-dimensional "cause" $\mathbf{z}$. Thus, $\mathbf{z} \rightarrow \mathbf{y}$.

PCA aims at "inverting the arrow" and inferring the latent low-dimensional $\mathbf{z}$ from $\mathbf{y}$.

Probabilistic Principal Component Analysis

Let us consider a probabilistic formulation of the PCA, assuming:

- $\mathbf{z} \in \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^D$ be continuous random variables;
- the distribution of $\mathbf{z}$ be Gaussian, i.e., $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, \mathbf{I})$;
- $\mathbf{z}$ and $\mathbf{x}$ related by a linear dependency:

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mathbf{b} + \epsilon$$

where $\epsilon \sim \mathcal{N}\left(\epsilon|0, \sigma^2\mathbf{I}\right)$.

Due to the Gaussian distribution we have:

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}\left(\mathbf{x}|\mathbf{W}\mathbf{z} + \mathbf{b}, \sigma^2\mathbf{I}\right),$$

which describes the probabilistic principal component analysis (pPCA) model [3, 1].

Explicit computation of the marginal likelihood

We can take advantage of properties of a linear combination of two vectors of normally distributed random variables to calculate the integral in (1) *explicitly* [3, 1]:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})\, d\mathbf{z} = \int \mathcal{N}\left(\mathbf{x}|\mathbf{W}\mathbf{z}+\mathbf{b}, \sigma^2\mathbf{I}\right) \mathcal{N}(\mathbf{z}|0,\mathbf{I})\, d\mathbf{z}$$
$$= \mathcal{N}\left(\mathbf{x}|\mathbf{b}, \mathbf{W}\mathbf{W}^\mathsf{T} + \sigma^2\mathbf{I}\right)$$

from which we are able to compute $\ln p(\mathbf{x})$.

Moreover, we can also calculate the true posterior over $\mathbf{z}$ analytically:

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{M}^{-1}\mathbf{W}^\mathsf{T}(\mathbf{x}-\mu), \sigma^{-2}\mathbf{M}\right), \qquad \text{where} \quad \mathbf{M} = \mathbf{W}^\mathsf{T}\mathbf{W} + \sigma^2\mathbf{I}$$

Once we find $\mathbf{W}$ that maximize the log-likelihood function, and the dimensionality of the matrix $\mathbf{W}$ is computationally tractable, we can calculate $p(\mathbf{z}|\mathbf{x})$. This is very important as for a given observation $\mathbf{x}$ we can calculate the distribution over the latent factors! [1]

Advantages and drawbacks of the pPCA

The probabilistic PCA is an extremely important latent variable model for two reasons [1]:

1. we can calculate everything *by hand* and, which helps understanding the latent variable models;
2. it is a linear model.

However, in the presence of a nonlinear dependency or other distributions than Gaussians, we would *no longer* be able to calculate the integral exactly, and some sort of *approximation* would be necessary.

Anyhow, pPCA is a model that everyone interested in latent variable models should study in depth to create an *intuition* about probabilistic modeling.

3 Generative Deep Latent Variable Models

Autoencoder

Unsupervised Learning

Autoencoder

In the presence of a larger amount of data or of a more complex problem, involving also a *nonlinear input-output relation*, we need deep learning methods to reduce the dimensionality, with the goal to enhance the representation power:

$$\min_{\mathbf{w}_1,\mathbf{w}_2} \left\| \mathbf{x}_i - g\left(f\left(\mathbf{x}_i;\mathbf{w}_1\right);\mathbf{w}_2\right) \right\|_2^2$$

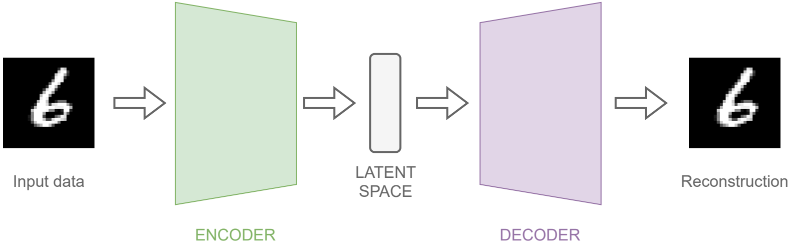


Figure 4: Diagram of an autoencoder, a neural network model composed of: an encoder notework that compresses high-dimensional input data into a lowerdimensional representation vector, and a decoder network that decompresses a given representation vector back to the original domain.

Unsupervised representation learning

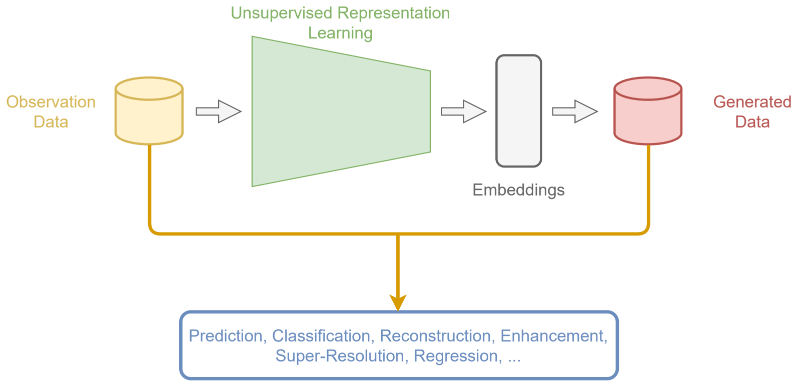

Unsupervised Representation Learning
Observation Data
Embeddings
Generated Data
Prediction, Classification, Reconstruction, Enhancement, Super-Resolution, Regression, ...

Figure 5: Generative modeling can often be thought of as an auxiliary task to a discriminative task.

Unsupervised representation learning refers to a quest for *disentangled*, *semantically meaningful*, *statistically independent* and *causal* factors of variation in data.

Representation learning as implicit regularization

Alternatively, one may view this as an implicit form of regularization.

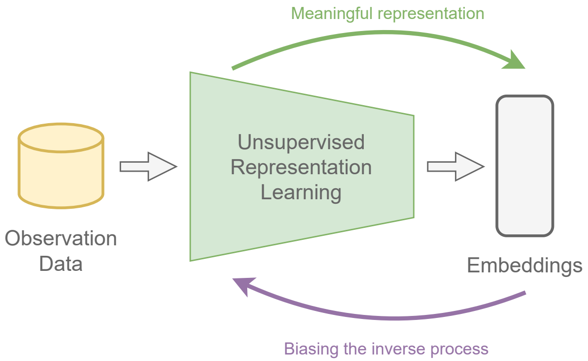

Meaningful representation
Observation Data
Unsupervised Representation Learning
Embeddings
Biasing the inverse process

Figure 6: By forcing the representations to be meaningful for data generation, we bias the inverse of that process, which maps from input to representation, into a certain mould.

4 The Variational Autoencoder Approach

The VAE framework

The framework of **variational autoencoders** (VAEs) [4, 5] provides a principled method for jointly learning deep latent-variable models and corresponding inference models using stochastic gradient descent.

The framework has a wide array of applications, from generative modeling to semi-supervised learning and representation learning.

Two coupled parameterized models

The VAE can be viewed as *two coupled, but independently parameterized models*: the *encoder*, or recognition model or inference model, and the *decoder*, or generative model.

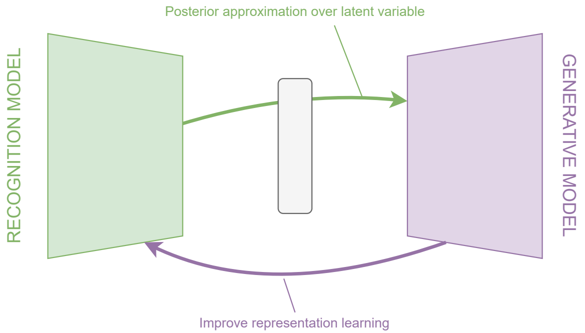

Posterior approximation over latent variable
RECOGNITION MODEL
GENERATIVE MODEL
Improve representation learning

Figure 7: The recognition model is the approximate inverse of the generative model according to Bayes rule.

Advantages of the VAE: amortized inference

In the VAE framework, the recognition model is a stochastic function of the input variables.

This is in contrast to the case in which each data observations has a separate variational distribution, which is inefficient for large data-set.

The recognition model uses one set of parameters to model the relation between input and latent variables and it is referred to as amortized inference [6].

The recognition model can be arbitrary complex but is still reasonably fast because it can be done using a single feedforward pass from input to latent variables.

Advantages of the VAE: reparameterization trick

The parameter learning process of the recognition model introduces gradient noise.

The VAE tackes this problem by the reparameterization trick, a simple procedure to reorganize the gradient computation, thus reducing variance in the gradients.

With respect to other generative models, and like other likelihood-based models, VAEs generate more dispersed samples, but are better density models in terms of the likelihood criterion.

Marrying graphical models and deep learning

The generative model is a Bayesian network of the form $p\left(\mathbf{x}|\mathbf{z}\right)p\left(\mathbf{z}\right)$, where $\mathbf{x}$ represents the input and $\mathbf{z}$ the latent vectors.

Similarly, the recognition model is also a conditional Bayesian network of the form $q\left(\mathbf{z}|\mathbf{x}\right)$.

Each conditional may hide a complex (deep) neural network, as $\mathbf{z}|\mathbf{x} \sim f\left(\mathbf{x}, \boldsymbol{\epsilon}\right)$, with f a neural network mapping and $\boldsymbol{\epsilon}$ a noise random variable.

Its learning algorithm is a mix of classical (amortized, variational) expectation maximization, which ends up backpropagating through the many layers of the deep neural networks through the reparameterization trick.

5 Variational Inference

Probabilistic models

In machine learning, we are often interested in learning probabilistic models of various natural and artificial phenomena from data.

We typically need to assume some level of uncertainty over aspects of the model.

The degree and nature of this uncertainty is specified in terms of (conditional) probability distributions.

Conditional models

Often, we are not interested in learning an unconditional model $p_{\boldsymbol{\theta}}(\mathbf{x})$, but a conditional model $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ such that:

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \approx p_{\text{data}}(\mathbf{y}|\mathbf{x}) .$$

Conditional models become more difficult to learn when the predicted variables are very high-dimensional, such as images, video or sound.

To avoid notational clutter, unconditional modeling is often assumed but we can generalize those models to conditional ones.

Parameterizing conditional distributions with neural networks

Neural networks are a particularly flexible and computationally scalable type of function approximator, denoted as $f\left(\cdot\right)$.

In case of neural network based image classification [7, 8], neural networks parameterize a categorical distribution $p_{\boldsymbol{\theta}}\left(y|\mathbf{x}\right)$ over a class label l, conditioned on an image $\mathbf{x}$:

$$\mathbf{y} = f\left(\mathbf{x}\right)$$
$$p_{\boldsymbol{\theta}}\left(l|\mathbf{x}\right) = \text{Categorical}\left(l;\mathbf{y}\right)$$

where the last operation of $f\left(\cdot\right)$ is typically a softmax function such that $\sum_i y_i = 1$.

Structured probabilistic models

Probability distributions over a very large number of random variables involve direct interactions between relatively few variables [9].

Using a single function to describe the entire joint probability distribution can be very inefficient (both computationally and statistically).

Instead of using a single function to represent a probability distribution, we can split a probability distribution into many factors that we multiply together.

These factorizations can greatly reduce the number of parameters needed to describe the distribution.

When we represent the factorization of a probability distribution with a graph, we call it a structured probabilistic model or *graphical model*.

Probabilistic graphical models and neural networks I

Directed graphical models, or *Bayesian networks*, are a type of probabilistic models where all the variables are topologically organized into a *directed acyclic graph*.

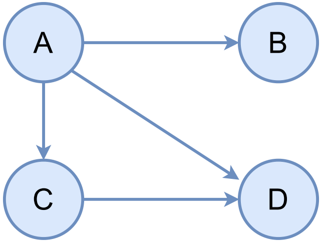


Figure 8: Example of a directed acyclic graph on four vertices.

Probabilistic graphical models and neural networks II

The joint distribution over the variables of such models factorizes as a product of prior and conditional distributions:

$$p_{\boldsymbol{\theta}}\left(\mathbf{x}_1, \ldots, \mathbf{x}_M\right) = \prod_{j=1}^{M} p_{\boldsymbol{\theta}}\left(\mathbf{x}_j | A\left(\mathbf{x}_j\right)\right)$$

where $A\left(\mathbf{x}_j\right)$ is the set of parent variables of node j in the directed graph.

Traditionally, $p_{\boldsymbol{\theta}}\left(\mathbf{x}_j | A\left(\mathbf{x}_j\right)\right)$ is parameterized as a lookup table or a linear model [10].

A more flexible parameterization involves neural networks:

$$\boldsymbol{\eta} = f\left(A\left(\mathbf{x}\right)\right)$$
$$p_{\boldsymbol{\theta}}\left(\mathbf{x} | A\left(\mathbf{x}\right)\right) = p_{\boldsymbol{\theta}}\left(\mathbf{x} | \boldsymbol{\eta}\right)$$

Learning in fully observed models with neural networks

If all variables in the directed graphical model are observed in the data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$, under the *i.i.d. assumption*, then we can compute and differentiate the log-probability of the data under the model:

$$\log p_{\boldsymbol{\theta}}(\mathcal{D}) = \sum_{\mathbf{x} \in \mathcal{D}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

Under the ML criterion, we attempt to find the parameters $\boldsymbol{\theta}$ that maximize the sum, or equivalently the average, of the log-probabilities assigned to the data by the model.

ML maximization is important due to its equivalence to Kullback Leibler (KL) divergence minimization.

Unbiased stochastic gradients

We optimize the ML objective by using a stochastic gradient descent (SGD), which uses randomly drawn minibatches of data $\mathcal{M} \subset \mathcal{D}$ of size $N_{\mathcal{M}}$:

$$\frac{1}{N_{\mathcal{D}}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathcal{D}) \simeq \frac{1}{N_{\mathcal{M}}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathcal{M}) = \frac{1}{N_{\mathcal{M}}} \sum_{\mathbf{x} \in \mathcal{M}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

The symbol $\simeq$ means that one of the two sides is an unbiased estimator of the other side: the two sides are equal when averaged over the noise distribution.

From a Bayesian perspective, we can improve upon ML by using the maximum a posteriori (MAP) estimation.

Directed models with latent variables

Latent variables $\mathbf{z}$ are variables that are part of the model, but which we *don't observe*, and are therefore not part of the dataset.

The marginal distribution over the observed variables $p_{\boldsymbol{\theta}}(\mathbf{x})$ is given by:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\, d\mathbf{z} \tag{2}$$

where the joint probability $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ is represented by a directed graphical model.

$p_{\boldsymbol{\theta}}(\mathbf{x})$ is also known as marginal likelihood and it is a *flexible* implicit distribution, since it can be represented by mixtures of distributions.

Deep latent variable models

We refer as **deep latent variable model** (DLVM) to a latent variable model $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ whose distributions are parameterized by neural networks.

Even when each factor in the directed model is relatively simple, the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$ can be very complex.

This expressivity makes DLVMs attractive for approximating $p_{\text{data}}(\mathbf{x})$.

The simplest, and most common, DLVM is defied by the following factorization:

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$$

where the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ and/or $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ are specified.

Example DLVM for multivariate Bernoulli data

A simple DLVM example [4] for binary data $\mathbf{x}$ involves a spherical Gaussian latent space and a factorized Bernoulli observation model $p(\mathbf{x}|\mathbf{z})$:

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; 0, \mathbf{I}) \\
\mathbf{y} &= f_{\boldsymbol{\theta}}(\mathbf{z}) \\
\log p(\mathbf{x}|\mathbf{z}) &= \sum_{j=1}^{D} \log p(x_j|\mathbf{z}) = \sum_{j=1}^{D} \log \text{Bernoulli}(x_j; y_j) \\
&= \sum_{j=1}^{D} x_j \log y_j + (1 - x_j) \log (1 - y_j) .
\end{aligned}
$$

Intractabilities

The main difficulty of ML learning in DLVMs is that the marginal probability $p_{\boldsymbol{\theta}}(\mathbf{x})$ of data under the model is typically intractable.

This is due to absence of any analytical solution of the integral in (2).

Thus, differentiation is not possible, contrary to what is possible with fully observable data.

The intractability of $p_{\boldsymbol{\theta}}(\mathbf{x})$ is related to the intractability of the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, as:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}.$$

Approximate inference techniques allow us to approximate $p_{\boldsymbol{\theta}}(\mathbf{x})$ and $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ in DLVMs.

6 Formalizing Variational Autoencoders

Encoder or approximate posterior

The VAE framework provides a computationally efficient way for optimizing DLVMs jointly with a corresponding inference model using SGD.

To turn the DLVM's intractable posterior into tractable problems, we introduce a parametric inference model $q_{\phi}\left(\mathbf{z}|\mathbf{x}\right)$.

This model is also called an **encoder** or *recognition model*.

The model parameters ϕ are known as variational parameters, which should be optimized such that:

$$q_{\phi}\left(\mathbf{z}|\mathbf{x}\right) \approx p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{x}\right).$$

Parameterizing the inference model

Like a DLVM, the inference model can be any directed graphical model:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = q_{\boldsymbol{\phi}}(\mathbf{z}_1, \ldots, \mathbf{z}_M|\mathbf{x}) = \prod_{j=1}^{M} q_{\boldsymbol{\phi}}(\mathbf{z}_j|A(\mathbf{z}_j), \mathbf{x})$$

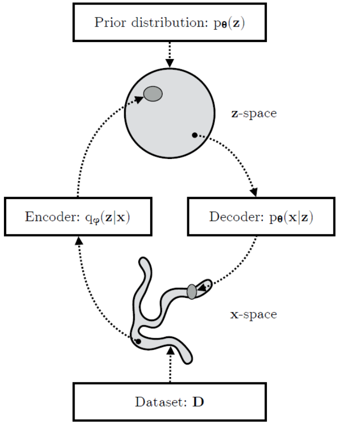
where $A(\mathbf{z}_j)$ is the set of parent variables of $\mathbf{z}_j$ in the directed graph.

And also similar to a DLVM, the distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can be parameterized using deep neural networks, e.g.:

$$\begin{aligned}(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) &= f_{\boldsymbol{\phi}}(\mathbf{x}) \\ q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}))\end{aligned}$$

Tipically, a single encoder neural network $f_{\boldsymbol{\phi}}(\cdot)$ is used to perform posterior inference over all of the datapoints (amortized variational inference strategy [6]).

The VAE's stochasting mapping process



Prior distribution: $p_{\boldsymbol{\theta}}(\mathbf{z})$
z-space
Encoder: $q_{\varphi}(\mathbf{z}|\mathbf{x})$
Decoder: $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$
x-space
Dataset: $\mathbf{D}$

Figure 9: A VAE learns stochastic mappings between an observed $\mathbf{x}$-space, whose empirical distribution $q_{\mathcal{D}}(\mathbf{x})$ is typically complicated, and a latent $\mathbf{z}$-space, whose distribution can be relatively simple. The generative model learns a joint distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ that is often (but not always) factorized as $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, with a prior distribution over latent space $p_{\boldsymbol{\theta}}(\mathbf{z})$, and a stochastic decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. The stochastic encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, also called inference model, approximates the true but intractable posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ of the generative model [8].

Maximization of the marginal log-likelihood

The optimization objective of the VAE is the **evidence lower bound** (ELBO).

For any choice of the inference model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, the maximization of the log-likelihood can be expressed as:

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log p_{\boldsymbol{\theta}}(\mathbf{x})\right\} \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right)\right\} \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right)\right\} \\
&= \underbrace{\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right)\right\}}_{\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})} + \underbrace{\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log\left(\frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right)\right\}}_{\mathcal{D}_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))}
\end{aligned}
\tag{3}
$$

where $\mathcal{D}_{\text{KL}}(\cdot)$ the KL divergence and $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\cdot)$ is the ELBO.

KL divergence and ELBO

The KL divergence between $q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)$ and $p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{x}\right)$ in (3) is nonnegative:

$$\mathcal{D}_{\mathsf{KL}}\left(q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)||p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{x}\right)\right) \geq 0. \tag{4}$$

The ELBO in (3) can be expressed as:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}\left(\mathbf{x}\right) = \mathrm{E}_{q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)}\left\{\log p_{\boldsymbol{\theta}}\left(\mathbf{x},\mathbf{z}\right) - \log q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)\right\}$$

Due to (4), the ELBO is a lower bound on the log-likelihood of the data:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}\left(\mathbf{x}\right) = \log p_{\boldsymbol{\theta}}\left(\mathbf{x}\right) - \mathcal{D}_{\mathsf{KL}}\left(q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)||p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{x}\right)\right) \leq \log p_{\boldsymbol{\theta}}\left(\mathbf{x}\right)$$

Role of KL divergence and ELBO

The KL divergence $\mathcal{D}_{\text{KL}}\left(q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)||p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{x}\right)\right)$ determines two distances:

1. the KL divergence of the approximate posterior from the true posterior, by definition;
2. the tightness of the bound, i.e., the gap between the ELBO $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}\left(\mathbf{x}\right)$ and the marginal likelihood $\log p_{\boldsymbol{\theta}}\left(\mathbf{x}\right)$.

Maximizing the ELBO concurrently leads to:

1. maximizing the marginal likelihood $p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{x}\right)$, which improves the generation;
2. minimizing the KL divergence, which improves the approximation of $q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)$ to $p_{\boldsymbol{\theta}}\left(\mathbf{z}|\mathbf{x}\right)$.

Computational flow in the VAE

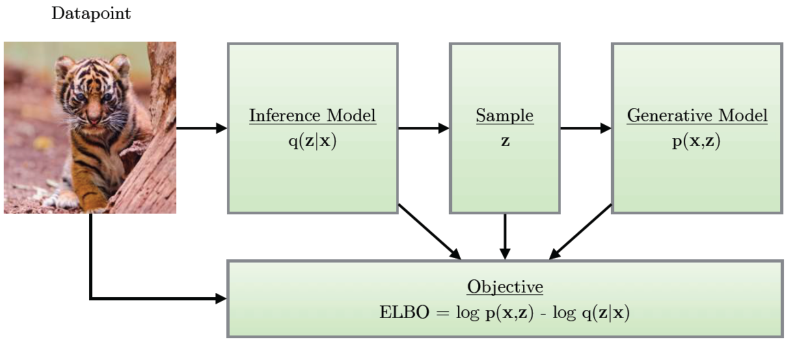


Figure 10: Simple schematic of computational flow in a variational autoencoder [8].

Stochastic gradient-based optimization of the ELBO I

The ELBO allows joint optimization w.r.t. all parameters ($\boldsymbol{\theta}$ and $\boldsymbol{\phi}$) using SGD.

The gradient of individual-datapoint ELBO $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$ is in general intractable.

Unbiased gradients of the ELBO w.r.t. parameters $\boldsymbol{\theta}$ are simple to obtain:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\theta}}\,\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right\} \\
&= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\nabla_{\boldsymbol{\theta}}\left(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right)\right\} \\
&\simeq \nabla_{\boldsymbol{\theta}}\left(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right) \\
&= \nabla_{\boldsymbol{\theta}}\left(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\right)
\end{aligned}$$

Stochastic gradient-based optimization of the ELBO II

Unbiased gradients of the ELBO w.r.t. *variational* parameters ϕ are more difficult to obtain, since the expectation depends on ϕ and cannot be approximated:

$$
\begin{aligned}
\nabla_{\phi}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}\left(\mathbf{x}\right) &= \nabla_{\boldsymbol{\phi}}\,\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log p_{\boldsymbol{\theta}}\left(\mathbf{x},\mathbf{z}\right) - \log q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)\right\} \\
&\neq \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\nabla_{\boldsymbol{\theta}}\left(\log p_{\boldsymbol{\theta}}\left(\mathbf{x},\mathbf{z}\right) - \log q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right)\right)\right\}
\end{aligned}
\tag{5}
$$

To solve this problem, we can use a change of variables, also called the reparameterization trick [4, 5].

Change of variables

First, we express the random variable $\mathbf{z} \sim q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ as some differentiable (and invertible) transformation of another random variable $\boldsymbol{\epsilon}$, given $\mathbf{z}$ and $\boldsymbol{\phi}$:

$$\mathbf{z} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x}) \tag{6}$$

where the distribution of the random variable $\boldsymbol{\epsilon}$ is independent of $\mathbf{x}$ or $\boldsymbol{\phi}$.

Thus, the expectation $\mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\{\cdot\}$ can be rewritten in terms of $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\{f(\mathbf{z})\} &= \nabla_{\boldsymbol{\phi}} \mathrm{E}_{p(\boldsymbol{\epsilon})}\{f(\mathbf{z})\}\\ &= \mathrm{E}_{p(\boldsymbol{\epsilon})}\{\nabla_{\boldsymbol{\phi}} f(\mathbf{z})\}\\ &\simeq \nabla_{\boldsymbol{\phi}} f(\mathbf{z}).\end{aligned}$$

Illustration of the reparameterization trick

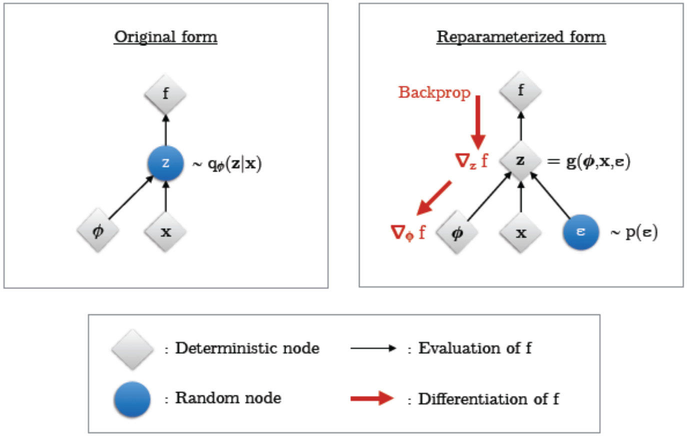


Figure 11: The variational parameters ϕ affect the objective f through the random variable $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. In the original form (left), we cannot differentiate f w.r.t. ϕ, because we cannot directly backpropagate gradients through the random variable $\mathbf{z}$. We can “externalize” the randomness in $\mathbf{z}$ by re-parameterizing the variable as a deterministic and differentiable function of ϕ, $\mathbf{x}$, and a newly introduced random variable ϵ. This allows us to “backprop through $\mathbf{z}$” and compute gradients $\nabla_\phi f$ [8].

Stochastic gradient of the ELBO under reparameterization

We can now replace an expectation w.r.t. $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, where $\mathbf{z} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$, with one w.r.t. $p(\boldsymbol{\epsilon})$, thus the ELBO becomes:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left\{\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right\} \\ &= \mathrm{E}_{p(\boldsymbol{\epsilon})}\left\{\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right\}\end{aligned}$$

Thus, we can form a simple Monte Carlo estimator $\widetilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$ where we use a single noise sample $\boldsymbol{\epsilon}$ from $p(\boldsymbol{\epsilon})$:

$$\begin{aligned}\boldsymbol{\epsilon} &\sim p(\boldsymbol{\epsilon}) \\ \mathbf{z} &= g(\boldsymbol{\phi}, \mathbf{x}, \boldsymbol{\epsilon}) \\ \widetilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \end{aligned} \tag{7}$$

The reparameterized ELBO estimator is referred to as the stochastic gradient variational Bayes (SGVB) estimator [4].

Computation of $\log q_{\phi}(\mathbf{z}|\mathbf{x})$

The computation of the ELBO estimator in (7) requires the computation of $\log q_{\phi}(\mathbf{z}|\mathbf{x})$, which is very simple if we use (6).

As long as $g(\cdot)$ in (6) is an invertible function, $\boldsymbol{\epsilon}$ and $\mathbf{z}$ are related by:

$$\log q_{\phi}(\mathbf{z}|\mathbf{x}) = \log p(\boldsymbol{\epsilon}) - \log d_{\phi}(\mathbf{x}, \boldsymbol{\epsilon})$$

where:

$$\log d_{\phi}(\mathbf{x}, \boldsymbol{\epsilon}) = \log \left| \det \left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right) \right|$$

$$\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} = \frac{\partial (z_1, \ldots, z_k)}{\partial (\epsilon_1, \ldots, \epsilon_k)} = \begin{pmatrix} \frac{\partial z_1}{\partial \epsilon_1} & \cdots & \frac{\partial z_1}{\partial \epsilon_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_k}{\partial \epsilon_1} & \cdots & \frac{\partial z_k}{\partial \epsilon_k} \end{pmatrix}$$

It is possible to build very flexible transformations $g(\cdot)$ for which $\log d_{\phi}(\mathbf{x}, \boldsymbol{\epsilon})$ is simple to compute, resulting in highly flexible inference models $q_{\phi}(\mathbf{z}|\mathbf{x})$.

Factorized Gaussian posteriors I

A common choice is a simple factorized Gaussian encoder $q_{\phi}\left(\mathbf{z}|\mathbf{x}\right)=\mathcal{N}\left(\mathbf{z};\boldsymbol{\mu},\mathrm{diag}\left(\boldsymbol{\sigma}^{2}\right)\right)$:

$$\begin{aligned}\left(\boldsymbol{\mu},\log\boldsymbol{\sigma}\right)&=f_{\phi}\left(\mathbf{x}\right)\\ q_{\phi}\left(\mathbf{z}|\mathbf{x}\right)&=\prod_{i}q_{\phi}\left(z_i|\mathbf{x}\right)=\prod_{i}\mathcal{N}\left(z_i;\mu_i,\sigma_i^2\right)\end{aligned}$$

After reparameterization, we can write:

$$\begin{aligned}\boldsymbol{\epsilon}&\sim\mathcal{N}\left(0,\mathbf{I}\right)\\ \left(\boldsymbol{\mu},\log\boldsymbol{\sigma}\right)&=f_{\phi}\left(\mathbf{x}\right)\\ \mathbf{z}&=\boldsymbol{\mu}+\boldsymbol{\sigma}\odot\boldsymbol{\epsilon}\end{aligned}$$

Factorized Gaussian posteriors II

The log determinant of the Jacobian is:

$$\log d_{\boldsymbol{\phi}}\left(\mathbf{x}, \boldsymbol{\epsilon}\right) = \log \left|\det\left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}}\right)\right| = \sum_i \log \sigma_i$$

where $\partial \mathbf{z}/\partial \boldsymbol{\epsilon} = \operatorname{diag}\left(\boldsymbol{\sigma}\right)$.

Thus, the posterior density is:

$$\log q_{\boldsymbol{\phi}}\left(\mathbf{z}|\mathbf{x}\right) = \log p\left(\boldsymbol{\epsilon}\right) - \log d_{\boldsymbol{\phi}}\left(\mathbf{x}, \boldsymbol{\epsilon}\right) = \sum_i \log \mathcal{N}\left(\epsilon_i; 0, 1\right) - \log \sigma_i .$$

Estimation of the marginal likelihood

After training a VAE, we can estimate the probability of data under the model using an importance sampling technique [5].

The marginal likelihood of a datapoint can be written as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \mathrm{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \{ p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) / q_{\boldsymbol{\phi}}(\mathbf{z}|x) \} \tag{8}$$

Taking random samples from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, a Monte Carlo estimator of (8):

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \approx \log \frac{1}{L} \sum_{l=1}^{L} p_{\boldsymbol{\theta}}\left(\mathbf{x}, \mathbf{z}^{(l)}\right) / q_{\boldsymbol{\phi}}\left(\mathbf{z}^{(l)}|\mathbf{x}\right) \tag{9}$$

where each $\mathbf{z}^{(l)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|x)$ is a random sample from the inference model.

By making L large, the approximation (9) becomes a better estimate of the marginal likelihood (8).

Variational autoencoders: training stage

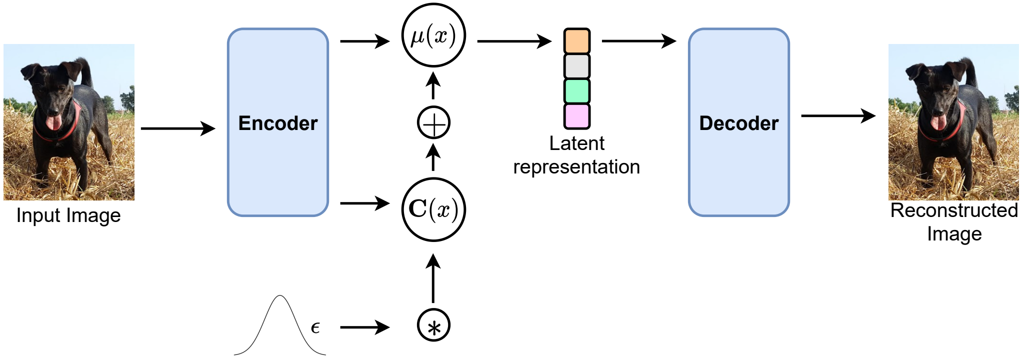


Figure 12: During training, the encoder learns the data statistics to adjust the sample ϵ from a $\mathcal{N}(0, I)$ to build the latent representation. The representation is then passed to the decoder for the reconstruction. (Courtesy of Eleonora Grassucci [11]).

Variational autoencoders: testing stage

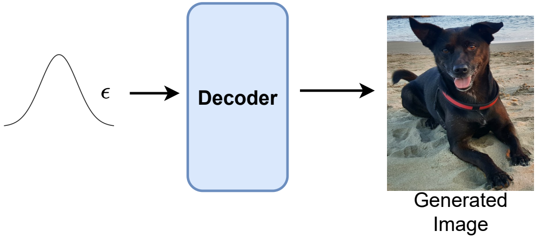


Figure 13: At testing time, a sample is drawn from $\mathcal{N}(0, \mathbf{I})$ and passed to the decoder which constructs the new image. (Courtesy of Eleonora Grassucci [11].)

VAE challenges: optimization issues

Stochastic optimization with the unmodified lower bound objective can gets stuck in an undesirable stable equilibrium.

At the start of training, the likelihood term $\log p\left(\mathbf{x}|\mathbf{z}\right)$ is relatively weak, such that an initially attractive state is where $q\left(\mathbf{z}|\mathbf{x}\right) \approx p\left(\mathbf{z}\right)$, resulting in a stalling.

To this end, a solution is to use an optimization schedule where the weights of the latent cost $\mathcal{D}_{\text{KL}}\left(q\left(\mathbf{z}|\mathbf{x}\right)||p\left(\mathbf{z}\right)\right)$ is slowly annealed from 0 to 1 over many epochs [12].

Alternatively, the free bits method [13] introduces a modification of the ELBO objective, that ensures that on average, a certain minimum number of bits of information are encoded per latent variable.

References I

[1] J. M. Tomczak, *Deep Generative Modeling*. Springer Nature Switzerland AG, 2022.

[2] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv Preprint: arXiv:1312.6114v10*, May 2014.

[5] D. J. Rezende, L. Metz, and S. Chintala, "Stochastic backpropagation in approximate inference in deep generative models," in *Int. Conf. on Machine Learning (ICML)*, Beijing, China, Jun. 2014, pp. 1278–1286.

[6] S. Gershman and N. Goodman, "Amortized inference in probabilistic reasoning," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 36, 2014.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, Nov. 2019.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: The MIT Press, 2016.

[10] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

References II

[11] E. Grassucci, D. Comminiello, and A. Uncini, “An information-theoretic perspective on proper quaternion variational autoencoders,” *Entropy*, vol. 23, no. 7, p. 856, Jul. 2021.

[12] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” *arXiv preprint arXiv:1511.06349*, 2015.

[13] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, “Improved variational inference with inverse autoregressive flow,” in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4743–4751.

Generative Latent Variable Models: The Variational Autoencoder

Generative Deep Learning
2021/2022

Danilo Comminiello

PhD Course in Information and Communication Technology (ICT)

http://danilocomminiello.site.uniroma1.it

danilo.comminiello@uniroma1.it