Modèles à Variables Latentes Profonds : des propriétés aux structures

Victor Berger

université
PARIS-SACLAY

Modèles à Variables Latentes: des propriétés aux structures

Deep Latent Variable Models: from properties to structures

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique
Unité de recherche: Université Paris-Saclay, CNRS, Laboratoire interdisciplinaire des sciences du numérique, 91405, Orsay, France
Référent: Faculté des Sciences d'Orsay

Thèse présentée et soutenue à Paris-Saclay, le 13/10/2021, par

Victor BERGER

Composition du jury

Anne Vilnat Professeure des universités, LISN, Université Paris-Saclay	Présidente
Danilo Jimenez Rezende Senior Staff Research Scientist, DeepMind	Rapporteur & Examinateur
Yann Chevaleyre Professeur des universités, LAMSADE, Université Paris Dauphine	Rapporteur & Examinateur
Stéphane Canu Professeur des universités, LITIS, INSA de Rouen	Rapporteur & Examinateur

Direction de la thèse

Michèle Sebag Directrice de Recherche, LISN, CNRS	Directrice

Thèse de doctorat

NNT: 2021UPASG053

Résumé / Abstract

Résumé

Les Modèles à Variables Latentes Profonds sont des modèles génératifs combinant les Réseaux Bayésiens avec l'apprentissage profond, illustrés par le célèbre *Auto-encodeur Variationnel*. Cette thèse se focalise sur leur structure, entendue comme la combinaison de 3 aspects : le graphe du Réseau Bayésien, le choix des familles probabilistes des variables, et l'architecture des réseaux de neurones. Nous démontrons que de nombreux aspects et propriétés de ces modèles peuvent être compris et contrôlés par cette structure, sans altérer l'objectif d'entraînement construit sur l'*Evidence Lower Bound*.

La première contribution concerne l'impact du *modèle d'observation* – la modélisation probabiliste des variables observées – sur le processus d'entraînement : comment il détermine la séparation entre signal et bruit, ainsi que son impact sur la dynamique de l'entraînement lorsque son paramètre d'échelle est appris plustôt que fixé, où il agit alors comme un processus de recuit simulé.

La seconde contribution, *CompVAE*, est centrée sur la structure hiérarchique des variables latentes : un modèle génératif conditionné par un multi-ensemble d'élements à combiner dans la génération finale. CompVAE démontre comment des propriétés globales – des manipulations ensemblistes dans ce cas – peuvent être atteintes par la seule conception structurale. Ce modèle est de plus validé empiriquement sur des données réelles, pour la génération de courbes de consommation électrique.

La troisième contribution, *Boltzmann Tuning of Generative Models (BTGM)*, est un cadre permettant d'ajuster un modèle génératif pré-entraîné selon un critère extérieur, en trouvant les ajustements minimaux nécessaire. Ceci est fait tout en contrôlant finement quelles variables latentes sont ajustées, et comment elles le sont. Nous démontrons empiriquement comment BTGM peut être utilisé pour spécialiser un modèle déjà entraîné, ou pour explorer les parties extrêmes d'une distribution générée.

Abstract

Deep Latent Variable Models are generative models combining Bayesian Networks and deep learning, illustrated by the renowned *Variational Autoencoder*. This thesis focuses on their structure, understood as the combination of 3 aspects: the Bayesian Network graph, the choice of probability distribution families for the variables, and the neural architecture. We show that and how several aspects and properties of those models can be understood and controlled through this structure, without altering the training objective constructed from the *Evidence Lower Bound*.

The first contribution concerns the impact of the *observation model* – the probabilistic modeling of the observed variables – on the training process: how it determines the demarcation between signal and noise and its impact on training dynamic when its scale parameter is learned rather than fixed. It then behaves similarly to a simulated annealing process.

The second contribution, *CompVAE*, is centered on the hierarchical structure of latent variables: a generative model conditioned by a multi-set of elements to be combined in the final generation. CompVAE demonstrates how global properties – ensemblist manipulations in this case – can be achieved by solely structural design. The model is furthermore empirically validated on real data to generate electrical consumption curves.

The third contribution, *Boltzmann Tuning of Generative Models (BTGM)*, is a framework for adjusting trained generative models according to an externally provided criterion while finding the minimal required adjustments. This is done while finely controlling which latent variables are adjusted and how the are. We empirically demonstrate how BTGM can be used to specialize a trained model or to explore the extreme parts of a generative distribution.

Remerciements

Je tiens tout d'abord à remercier Michèle Sebag, ma directrice de thèse, qui m'a offert l'opportunité d'entreprendre cette aventure et m'a guidé durant ces quatre années. Merci pour ton attention, tes conseils avisés et le temps que tu as su m'accorder.

Je souhaite également remercier Danilo Jimenez Rezender, Yann Chevaleyre, Stéphane Canu et Anne Vilnat, qui ont accepté de rapporter et examiner mon travail.

Je remercie Artelys, en collaboration de qui ces travaux de recherche ont été effectués, et en particulier Vincent Renaud qui a assuré le lien. Je remercie également l'ADEME, qui a financé le projet dans lequel ma thèse s'inscrit.

Merci à l'école doctorale STIC pour son suivi, et tout particulièrement à Stéphanie Druetta, secrétaire du pôle B, pour son aide régulière.

Un très grand merci à toute l'équipe Apprentissage et Optimisation pour son accueil chaleureux, ses séminaires stimulants et ses pauses café aux mathématiques interminables. Merci Marc, Guillaume, Cécile, Cyril, Flora, François, Aurélien, Yann, Théophile, Victor, Giancarlo, Diviyan, Guillaume, Éléonore, Adrien, Loris, Nilo, Zhengying, Armand, Marine pour les nombreuses discussions toutes plus passionnantes les unes que les autres, les sorties escalade, pique-niques et autres soirées tartiflette.

Je ne suis bien évidemment pas un chercheur solitaire, et je remercie Corentin et Pierre avec qui j'ai partagé mon bureau et on été une grande source d'échanges éclairantes pendant tout ce temps. Merci également à Adrian et Balthazar pour les collaborations enrichissantes. Et un merci à Laurent pour m'avoir fait découvrir les travaux d'E.T. Jaynes.

Un merci spécial aux personnes ~~à cause~~ grâce auxquelles j'ai pris le chemin de la thèse : Gaëtan, Laurent, Arsène et Philippe, tout ça c'est aussi grâce à vous.

Enfin un grand merci à toutes les personnes qui m'ont accompagné durant cette thèse. Merci à ma famille qui m'a aidé et soutenu dans ce projet un peu fou. Merci à toi Élodie. Merci aux peuplades de Rézosup et notamment #doctorat. Merci aux camarades du serveur Discord d'entraide et de shitpost.

Et finalement merci à vous, qui prenez le temps de lire ces pages.

Contents

Publications

Published articles

[Pol+19] Adrian Pol, Victor Berger, Gianluca Cerminara, Cécile Germain, and Maurizio Pierini. "Anomaly Detection With Conditional Variational Autoencoders". In: ICMLA 2019 - 18th IEEE International Conference on Machine Learning and Applications. Dec. 16, 2019 (cit. on pp. 8, 103, 109–113).

[BS20a] Victor Berger and Michèle Sebag. "From Abstract Items to Latent Spaces to Observed Data and Back: Compositional Variational Auto-Encoder". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 274–289. ISBN: 978-3-030-46150-8 (cit. on pp. 8, 115, 122).

Pre-Prints

[BS20b] Victor Berger and Michèle Sebag. "Variational Auto-Encoder: not all failures are equal". In: *arXiv:2003.01972 [cs, eess, stat]* (Mar. 4, 2020). arXiv: 2003.01972 (cit. on pp. 8, 73, 87).

[BS21] Victor Berger and Michele Sebag. "Boltzmann Tuning of Generative Models". In: *arXiv:2104.05252 [cs]* (Apr. 12, 2021). arXiv: 2104.05252 (cit. on pp. 8, 135).

Introduction

Generative modeling became a very hot topic in the field of deep learning in 2014, with both seminal papers presenting *Variational Auto-Encoders (VAEs)* [KW14; RMW14] and *Generative Adversarial Networks (GANs)* [Goo+14]. Each paper gave rise to a flourishing framework integrating probabilistic generative principles within deep learning methods.

This thesis focuses on the VAE framework and more specifically deep Latent Variable Models (LVMs), that elegantly combine the mathematical formalism of *Bayesian Networks* with the learning methods of artificial deep neural networks. This framework allows one to build powerful and flexible models, naturally amenable to the integration of prior knowledge about the considered data domain, as we aim to show in the following chapters.

A main research question guiding the organization of the manuscript is how to design deep Latent Variable Models depending on both the specifics of the data and the intended usage of the models, and how to understand their can and their can't. This analysis is conducted in the perspective of *Probabilistic Graphical Models* (PGM) first, and deep learning second. Focusing on the probabilistic structure of the models allows for a unified understanding of most other aspects, as this structure governs the training procedure and the relation of the model to its training data.

The applicative motivation for this research is the study of smart energy policies and dimensioning of electrical networks, more specifically the need for programmable generative models (PGMs) in order to produce realistic and exploratory simulations of electrical consumption in power systems[1]. Principled models built in the PGM-oriented perspective, presented in the last part of the thesis, aim to answer these needs.

Organization of this manuscript

This manuscript is structured in three parts, respectively devoted to i) the theory of deep LVMs as learned models; ii) the relationships of the deep LVM structure with the data; and iii) the design of latent structures in order to encourage or enforce desirable properties in view of the model intended usages.

Chapter 1 provides a general introduction to probabilistic graphical models (PGMs), at the core of the LVM framework and of this whole manuscript.

Part I revolves around the construction and analysis of deep LVMs. Chapter 2 introduces the concept of latent variables in a PGM, which can be *interpretable* or *abstract*, and their training within the Evidence Lower Bound (ELBO) framework.

[1] Contract NEXT, funded by ADEME, French *Agence de la Transition Ecologique*.

Chapter 3 pushes this construction further by combining the ELBO criterion with deep learning in the spirit of the *Variational Auto-Encoder* (VAE) [KW14; RMW14], and analyses the interaction of the *inference model* with the training process. Chapter 4 introduces the concept of *hierarchical deep LVMs*, which involves multiple latent variables organized in a hierarchy, and discusses the difficulties encountered when training such models.

Part II focuses on the relationship between the model and the available data, embodied in the so-called *observation model*. Chapter 5 presents a few sophisticated observation models from the literature, and discusses the problem of *posterior collapse* encountered by some of these models. Chapter 6 focuses on a particular type of observation model referred to as *quasi-deterministic*, which acts as a mere translation layer between the latent space of the model and the observed data, ensuring that all relevant information is captured by the latent variables. By relating these models to the *Manifold Hypothesis*, this chapter investigates the link between the observation model and the data information, and whether it can (or cannot) be exploited by the deep LVM. The answer to this question is theoretically and empirically shown to depend on the variance of the observation model [BS20b]. Then, Chapter 7 focuses on this variance and how it can be learned; it illustrates the profound impact of such a learning on the training dynamics of the whole LVM.

Part III details how hierarchical deep LVMs can be structurally designed to enforce desirable properties from the whole model. Chapter 8 analyses a few examples from the literature, and presents the usage of such models to achieve anomaly detection; this application was motivated by the identification faulty sensors in the CMS experiment at CERN [Pol+19]. Then, Chapter 9 presents CompVAE [BS20a], a deep LVM structure designed to represent a programmable *compositional* generative model, meant to generate instances aggregating a variable number of entities. This model is detailed and analyzed on 1D and 2D artificial problems, and applied to real world data in the context of electrical distribution networks. Finally, Chapter 10 presents the *Boltzmann Tuning of Generative Models* (BTGM) approach [BS21], aimed to *a posteriori* refining an already trained LVM and to oversample a part of the corresponding distribution depending on an externally provided criterion. The BTGM approach, stemmed from the practical motivation of identifying the electrical consumption peak, constitutes a principled alternative to rejection-based sampling.

Notations

In the manuscript, random variables are denoted with capital letters (e.g. A, X, Z); their instanciations are denoted with lowercase letters (e.g., a, x, z).

The probabilistic models are generally denoted p or q; the probability (or probability density) $p(X = x, Y = y)$ associated with a variable assignation is noted $p(x, y)$ for simplicity. By slight abuse of notation, the conditional and marginal distributions derived from this model are noted in the same way, e.g., $p(x, z|w)$ represents the distribution under model p, conditioned on $W = w$, and where all variables but X and Z have been marginalized.

Many probabilistic models considered in this work are parameterized; their vector of parameters is generally noted θ or ϕ, and the parameterized model is

correspondingly noted p_θ or q_ϕ. The notation similarly extends to conditional and marginal distributions, like $p_\theta(x, z|w)$.

Chapter 1
Introduction to Probabilistic Graphical Models

Contents

This chapter presents the general framework of the research conducted during my PhD: probabilistic models.

Models play an important role in analysis of the world, and are widely used in various forms across the scientific literature. They help understand relations and interactions between the various quantities of interest in a problem analysis by unveiling structure in the data. In many cases these models are probabilistic, either due to the intrinsic stochasticity of the phenomenon at hand or to account for the fact that this phenomenon is only partially known.

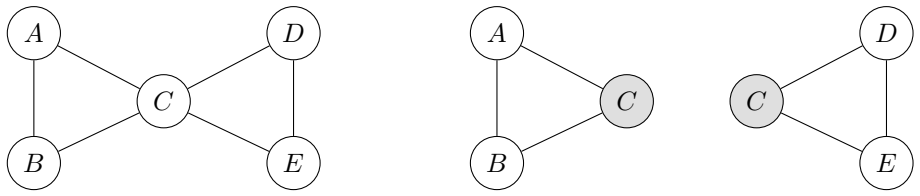
Probabilistic models generally take the form of a joint distribution over descriptive variables[1] $X_1, X_2, \ldots, X_N$. Due to the usually large number of variables involved, such joint distributions can very quickly become incredibly complex and tedious to manipulate. To address this issue, an approach is to introduce explicit structure into the model, e.g. accounting for some known dependency relations between the variables. Probabilistic graphical models aim at representing such relations using graphs.

[1] A notable example is the representation of a statistical link between illnesses and their symptoms. In such case, one can introduce one variable per illness and per symptom.

1.1 Independence relations as graphs

The probabilistic graphical model framework is as follows. Each variable of the model is represented as a node in the graph. Two nodes are linked by an edge if there is some direct dependency relationship between the two variables. In this representation, there exists a path in the graph between two nodes iff there exists some potential dependency between both variables. On the contrary, if no path exists, then the two variables must be independent.

Such a representation allows reasoning about subsets of variables, as illustrated in Figure 1.1. Large graphical models can be analyzed on a local basis if they are sufficiently sparse. This makes it possible to characterize the behavior of the model to some extend without needing to consider the whole joint distribution.



(a) Graphical model before observation **(b)** Equivalent graph with C observed

Figure 1.1: In this example graphical model (a), the observation of the value of a variable can split the graph into independent subgraphs (b): the pairs of variables (A, B) and (D, E) are independent of each other given the value of variable C.

As will be illustrated in the rest of this sections, graphs can directed or undirected, with different interpretations associated with the links. The two main types of graphical models are Bayesian Networks (which are directed graphs) and Markov Random Fields (which are undirected). While the thesis mostly relies on Bayesian Networks, some methods relying on Markov Random Fields are also discussed. Therefore, both frameworks are presented in this chapter, introducing their proper inference and training methods.

1.1.1 Bayesian Networks

Bayesian Networks, a prominent type of probabilistic graphical models [KF09], are based on Directed Acyclic Graphs (DAG): edges among nodes are oriented (represented as arrows) and the graph does not include any directed cycle. An example of such a network is given in Figure 1.2.

Let us introduce and illustrate the classic Bayesian Network terminology on this example. Each edge defines a directed relation between two nodes. The node from which the edge comes is called the *parent* and the node to which the arrow points is called the *child*. In the example graph, S and R are the *children* of C, H and C are the *parents* of S. Nodes H, S, C and R are *ancestors* of W. Similarly, S, R and W are *descendants* of C. There is no particular named relation between H and R, aside the fact that they are not independent.

One should note that the independence relations in such a directed graph are not as straightforward as in an undirected case(Figure 1.1). An important situation is when two nodes share a child, forming what is known as a *v-structure*. Figure 1.2

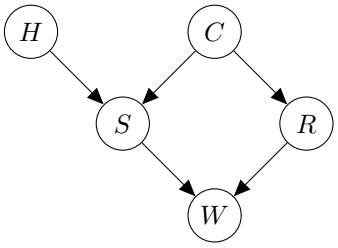


Figure 1.2: Example of a Bayesian Network modeling the possible causes of why the grass outside may be wet (W). The two direct considered causes are the rain (R) and the sprinkler (S). The rain is more likely to occur when the sky is cloudy (C), while the sprinkler is more likely to be used when the sky is clear and the air is hot (H). While the graph may appear cyclic, it is not when you consider the orientation of the edges. For example you cannot loop back to S by only following the arrows.

displays two of them: $H - S - C$ and $S - W - R$. In such a case, the parents are in general *not* independent given their child. In the running example, knowing that the grass is wet means that for sure either its has rained or the sprinkler has run; if one variable is $False$, then the other must be $True$. However, observing a variable still makes its different children independent of each other and its parents independent from its children, assuming there is no other path in the model linking them. In Figure 1.2, observing S and R would make (H, C) independent of W.

The graph representation of a Bayesian Network specifies the structure of the associated probabilistic model. Formally, this joint probability distribution is defined as the product of conditional distributions over the nodes. Letting $\pi(X_i)$ denote the set of parent variables of X_i, then:

$$p(X_1, X_2, \ldots, x_N) = \prod_{i=1}^{N} p(X_i|\pi(X_i)) \tag{1.1}$$

The example network of Figure 1.2 can thus be factorized as:

$$p(A, B, C, D, E) = p(A)p(B)p(C|A, B)p(D|B)p(E|C, D). \tag{1.2}$$

This representation allows for drastically compressing the model. Instead of specifying a full probability table over the 5 variables, we can just specify 5 small tables over at most 3 variables each, making the combinatorics of the model much more manageable. In our example, assuming all 5 variables were binary, the full probability table would have $2^5 = 32$ entries. By contrast, the factorized representation with 5 tables would involve only 24 entries, only half of which would be actually independent parameters. This difference can grow very quickly with the number of variables and the number of different values each variable can take.

Each term $p(x_i|\pi(X_i))$ in the factored distribution can be specified as a probability table in the discrete case. In the following, a more general formulation will be considered to handle both discrete and continuous variables. Each variable $X_i \sim \mathbb{P}_i(X_i; \omega_i))$ follows a given distribution (such as categorical, Gaussian, or Poisson distributions), the parameter of which depends on the value of its parent variables , we associate a parameterized distribution family ($\omega_i = f_i(\pi(X_i))$).

Eventually, the joint distribution $p(X_1 \ldots X_N)$ is fully specified from the DAG structure, the parameterized distribution family $\mathbb{P}_i$ associated to each variable X_i

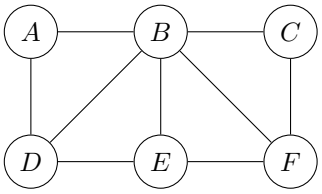


Figure 1.3: Example of a Markov Random Field.

and a parameter function f_i defining how the parameter of $P_i(X_i)$ depends on the parent variables of X_i:

$$pP(X_1 \dots X_N) = \prod_{i=1}^{N} \mathbb{P}_i(X_i \; ; \; \omega_i = f_i(\pi(X_i))) \tag{1.3}$$

1.1.2 Markov Networks

Markov Networks, also called Markov Random Fields (MRFs), are the other main type of graphical models [KF09]. They differ from Bayesian Networks in two ways: firstly, MRFs are represented via undirected graphs (as opposed to directed graphs for BNs). Secondly, while BNs characterize the joint distribution as the product of conditional distributions, MRFs aim to express independence relations, centered on the Markov Property: given a set of variables that splits the graph in two disjoint subsets (a.k.a. separating the graph), the distribution of the two resulting subsets are independent conditionally to the value of the separating variables.

MRFs are notably used in several statistical physics models. For example *Ising networks* represent networks of particles that can each be in two states, denoted *Up* and *Down*. Interactions between these particles are local: each particle tends to settle in a state depending on the states of its neighbors (as represented by the graph). The study of the produced MRFs allows understanding the large-scale behavior of lattices composed of many such particles.

Following the example of Figure 1.3, conditioning on the variables B and E splits the graph in two subgraphs, $\{A, D\}$ and $\{C, F\}$. Thus, the joint conditional distribution $p(A, C, D, F|B, E)$ factors as:

$$p(A, C, D, F|B, E) = p(A, D|B, E)p(C, F|B, E) \tag{1.4}$$

In many cases the probability distribution associated with the graphical model can be represented in a factorized form according to the cliques of the graph[2]. Cliques are fully-connected subgraphs: in the model of Figure 1.3, $\{B, D, E\}$ is a clique, while $\{A, B, D, F\}$ is not (it is missing an edge between A and E).

For each clique $\mathcal{C}$ in the graph, is it possible to define a potential function $\phi_{\mathcal{C}}(X_{\mathcal{C}})$ over the values of the variables contained in this clique, such that the whole probability distribution factors as a product of these potentials. These clique potentials reflect the independence assumptions included in the graph structure.

[2]This is the case if either the distribution is positive (no configuration has a probability of 0) or the graph is chordal (all cycle of size greater than 3 has an internal edge connecting two of its nodes, forming smaller cycles as well).

$$p(x_1, \ldots, x_N) = \frac{1}{Z} \prod_{\mathcal{C}} \phi_{\mathcal{C}}(X_{\mathcal{C}}) \tag{1.5}$$

The normalization constant Z is then defined as the sum of the potential values over all possible assignments of the variables, so that the distribution probability is correctly normalized. As a result, it depends on the potential functions:

$$Z = \sum_{X \in \mathcal{X}} \prod_{\mathcal{C}} \phi_{\mathcal{C}}(X_{\mathcal{C}}) \tag{1.6}$$

In Figure 1.3, $\{B, D, E\}$ is a clique, while $\{A, B, D, F\}$ is not (it is missing an edge between A and E). The joint distribution model thus reads

$$p(A, B, C, D, E, F) = \frac{1}{Z} \phi_{ABD} \phi_{BDE} \phi_{BEF} \phi_{BCF} \tag{1.7}$$

Note that with no loss of generality, one can consider only the largest cliques in the graph (e.g. ϕ_{AB} is accounted for by ϕ_{ABC}).

It is emphasized that potentials ϕ_C need not be normalized probability distributions (as opposed to Bayesian Networks). This gives more representation flexibility[3], at the expense of a loss in interpretability.

By analogy with statistical physics, it is often convenient to represent the clique potentials in logarithmic form: $f_C(x_C) = -\log \phi(x_C)$, in which case the joint distribution of the model takes the form of a Gibbs distribution:

$$p(x_1, \ldots, x_N) = \frac{1}{Z} \exp\left(-\sum f_C(x_C)\right) = \frac{1}{Z} \exp\left(-E(x_1, \ldots, x_N)\right) \tag{1.8}$$

The sum $E(x_1, \ldots, x_N) = \sum f_C(x_C)$ is by analogy named the *energy function* of the model.

The computation of the normalization constant Z is, in general, a hard problem: it requires computing and summing the potentials over all possible values of all variables. As a consequence, many algorithms working with Markov Networks are designed to work with the unnormalized probability, thus side-stepping the need to know the value of Z [KF09].

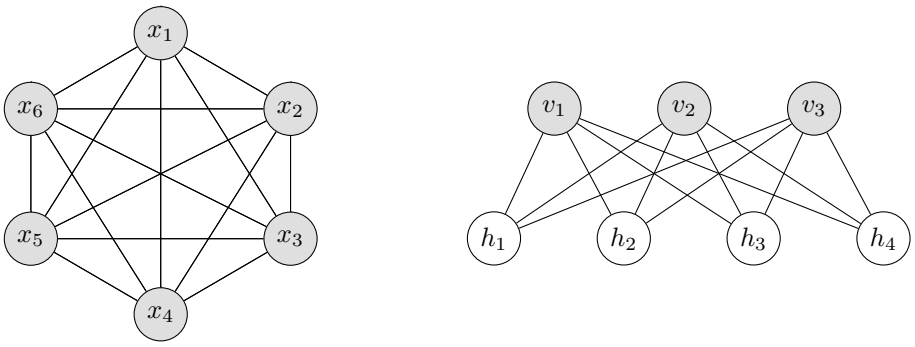
(Restricted) Boltzmann Machines Boltzmann Machines are a special case of Markov Networks where all nodes take binary values, and the energy function decomposes into terms involving only 1 or 2 variables. The general form of a Boltzman Machine energy function thus is:

$$E(x_1, \ldots, x_N) = \sum_{i<j} w_{ij} x_i x_j + \sum_i b_i x_i \tag{1.9}$$

A special case of Boltzmann Machines of special interest in machine learning are the so-called Restricted Boltzmann Machines, or RBMs. They additionally impose that the variables be split into two sets, respectively referred to as "visible" variables

[3] Indeed, any Bayesian Network can be converted into a Markov Networks by using the conditional probabilities associated to each node as a clique potential. On the other hand, not all Markov Networks can be converted into a Bayesian Network over the same set of variables.

$\{v_i\}$ and "hidden" variables $\{h_j\}$ (Figure 1.4(b)), and that visible (resp. hidden) variables are independent from each other. Potentials involving two variables thus always involve one hidden and one visible variable.



(a) Boltzman Machine **(b)** Restricted Boltzmann Machine

Figure 1.4: Examples for graphical representation of a Boltzmann Machine (a) and a Restricted Boltzmann Machine (b).

The strongly constrained structure of RBMs make them significantly easier to manipulate than general Boltzmann Machines or Markov networks. They display significant success as a machine learning tool on a large range of problems [HS06; SMH07; LB08].

1.2 Inference

A fully-specified graphical model provides a description of the underlying phenomenon. It can also be queried to answer questions such as "Assuming variable X takes the value x, what is the associated conditional probability of variable Y, $p(Y|X = x)$?". Answering such queries, referred to as What-If usage, is one of the major assets of graphical models[4]. This kind of questions generally involve a mix of conditioning the model on some variables and marginalizing others, to produce a probability distribution over the variables of interest.

An early example of such a use case is the design of Bayesian Networks to assist medical diagnostic, such as the Pathfinder models [HN92]. The relationships between candidate diseases and the associated symptoms are modeled as a Bayesian Network with the help of medical experts, and inference queries on this network allow to infer the most likely diseases given the observed patient symptoms.

Quite a few approaches have been designed to answer such queries depending on the query and the graph associated to the model. This section presents several of them. They generally apply to both Bayesian Networks and Markov Networks.

1.2.1 Exact inference on discrete variables

In the discrete finite case (every variable taking a finite number of values), any such query can be answered by producing the full probability table of the model, although

[4] Another usage is that of counterfactual reasoning (what if not) [PJS17; PM18]. This is outside of the scope of the presented research.

this approach clearly does not scale up with the number of variables and the size of their domain[5]. Depending on the shape of the graph and the kind of query, various algorithms have been developed to efficiently compute these queries.

Variable Elimination algorithm Some graphs can be processed in an iterative manner, eliminating the variables one after another. Doing so in a carefully chosen order can significantly reduce the total computational cost as it never requires to actually produce the full probability table[ZP94]. For example, following Figure 1.2, the variable A is a good candidate for being marginalized first: being liked to a single other variable doing it is an easy task. On the other hand marginalizing C first would link together variables A, B and E into a complex distribution.

Belief Propagation algorithm The belief propagation (BP) algorithm [Pea82] efficiently applies on tree-structured models. Given any number of observed variables with fixed value, the exact marginal distribution of every other variable is computed. BP can also apply on non tree-structured graphs, in which case it is referred to as *Loopy BP* and is part of the large family of *Approximate Message Passing* algorithms[DMM09]. It does no longer provide any guarantee on the result accuracy and might not converge in the general case, but under some conditions (in particular concerning the graph sparsity), it can still yield satisfactory approximations of the sought results [Wei00].

Junction tree algorithm When considering a non tree-structured BN, the junction tree algorithms proceed by creating additional meta-nodes and grouping initial nodes in such a way that the initial graph induces a tree-structure of the created meta-node graph; the belief propagation algorithm thus applies on the meta-node graph. This general idea comes in diverse variants [LS88; JOA90; She97], differing in how to select and group the nodes: the cost of the belief propagation highly depends on which nodes are grouped together.

1.2.2 Approximate inference with Monte Carlo methods

When the model involves continuous variables, or the query or model does not fit well with the previously described algorithm, one can turn to Markov-Chain Monte-Carlo (MCMC) methods in order to sample the sought distribution. These samples can then be used to compute statistical quantities of this distribution, and more generally to approximate the expectation of an arbitrary function under this distribution.

Markov Chains are the core element of these methods. They are stochastic processes that generate a sequence of values $(x^{(0)}, x^{(1)}, \dots)$. The process is defined by a transition probability function $T(x^{(t+1)}|x^{(t)})$, which describes the probability of the value at step $t+1$ given the value at step t. A visual illustration of such a process can be seen in a board game, where each player rolls the die at their turn and moves accordingly: the resulting position depends on the previous position and the result of the die throw.

[5] This inference problem bears mathematical similarities with that of constraints programming [RVW06], but we focus here on inference queries on probabilistic graphs.

As a result, a sample from the chain only depends on the value of the previous sample. This is another form of the Markov Property. Indeed, a Markov Chain can be represented as a Bayesian Network in which each variable $X^{(t)}$ is linked to the previous and next variables in the sequence, $X^{(t-1)}$ and $X^{(t+1)}$, as illustrated in Figure 1.5.

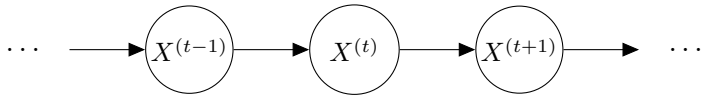


Figure 1.5: Representation of a Markov Chain as a Bayesian Network.

Under some conditions, Markov Chains have a stationary distribution π, defined as:

$$\pi(x') = \mathop{\mathbb{E}}_{x \sim \pi} T(x'|x) \tag{1.10}$$

It means that if $X^{(0)}$ is sampled from π and $X^{(1)}$ is sampled from the transition probability $T(\cdot|X^{(0)})$, then the distribution of $X^{(1)}$ also is π. In the case of an ergodic Markov Chain (i.e. any value can be reached from any starting point in a finite number of steps), then the stationary distribution π is unique and the long-run samples of the Markov Chain converge in law to it.

Markov-Chain Monte-Carlo is a family of methods that can be used to approximately sample a given target distribution p. The common principle is to try and design a Markov Chain whose stationary distribution is p. Samples from this Markov Chain can then by used as proxy for samples of p, with two caveats. First, we need to take into account the time of convergence for the Markov Chain: unless the starting point was sampled directly from the stationary distribution (which cannot be the case here), the initial samples are not distributed according it. This generally implies a "burn-in" time: the first K samples of the Markov Chain are discarded (with typical values of K being around 1000). The second caveat is that the successive samples from the Markov Chain are correlated. In order to approximate independent samples from the target distribution, it is further necessary to discard many samples between each one that is kept (for example keeping 1 sample in 100).

A sufficient condition for p being such a stationary distribution, it that the it satisfies the *detailed balance* property, making the Markov Chain *reversible*:

$$\forall x, x' : T(x'|x)p(x) = T(x|x')p(x') \tag{1.11}$$

In the context of graphical models, a Markov Chain proceeds by generating $X^{(t+1)}$ (the vector of all node values) from $X^{(t)}$. As is illustrated by the algorithms presented in the following sections, a strong merit of MCMC-based methods is that many of them only require an unnormalized target distribution, making them widely applicable.

1.2.2.1 Gibbs sampling

In most models, the sampling complexity is due to the correlation of the variables. However each individual variable x_i often follows a simple conditional distribution

relative to all other variables (noted x_{-i}): when all variables but one are frozen, the effective size of the graphical model is drastically reduced. Gibbs sampling [GG84] takes advantage of this property, by updating each x_i conditionally to the others, thus generating a sequence of intermediate states:

$$(x_0^{(t+1)}, \dots, x_{i-1}^{(t+1)}, \boxed{x_i^{(t)}}, x_{i+1}^{(t)}, \dots, x_n^{(t)}) \to (x_0^{(t+1)}, \dots, x_{i-1}^{(t+1)}, \boxed{x_i^{(t+1)}}, x_{i+1}^{(t)}, \dots, x_n^{(t)})$$

where $x_i^{(t+1)}$ is generated according to $p(x_i|x_{j<i}^{(t+1)}, x_{j>i}^{(t)})$. After all variables have been updated, the next step of the Markov Chain is given as $(x_1^{(t+1)}, \dots, x_n^{(t+1)})$.

Each individual variable updating of this procedure verifies the *detailed balance* property[6], ensuring that the Gibbs sampling procedure does indeed converge towards the sought distribution.

1.2.2.2 Metropolis-Hastings

Another approach is the Metropolis-Hastings algorithm family [Met+53; Has70]. In this context, the sampling of the next value is done in two steps. First, a new candidate value is proposed, by sampling some simple distribution $g(x'|x^{(t)})$, and it is then randomly accepted or refused though a biased coin flip, whose acceptance probability $a(x', x^{(t)})$ depends on the original value and the proposed value. If the sample is refused, the next value is taken as equal to the current value $x^{(t+1)} = x^{(t)}$.

The transition probability is thus given by $T(x'|x) = g(x'|x)a(x', x)$, and a common choice to ensure that the *detailed balance* property is verified is to choose the acceptance ratio as:

$$a(x', x) = \min\left(1, \frac{g(x|x')p(x')}{g(x'|x)p(x)}\right)$$

Note that with this choice, the sampling process only needs to compute probability ratios of p: $p(x')/p(x)$, meaning that it is no necessary to know its normalization constant. In particular when the proposal distribution is symmetric (that is $g(x'|x) = g(x|x')$), then the acceptance process simplifies to an acceptance rate equal to $\frac{p(x')}{p(x)}$ with automatic acceptance if the ratio is greater than one. The proposal is always accepted if it has an higher probability than the previous value, and otherwise is all the more likely to be refused that it has a lower probability according to p.

An example of symmetric proposal probability $g(x'|x)$ would for example be a Normal distribution with some fixed variance. In this context, the Metropolis-Hastings algorithm is similar to a Brownian motion biased towards regions of higher probability for the model p to approximate.

There is a trade-off in the choice of g: if the proposal distribution has a large variance, and proposes new samples that are far from the current value, these samples are likely to fall in a low-probability region and be rejected. On the other hand, having g proposing only samples close to the current value, while increasing the chance that they are accepted, will cause the Markov Chain to explore the space slowly, requiring to discard more samples between two selected ones for these to be considered independent.

[6] If T_i^{Gibbs} is the operation that updates the i-th variable, then $T_i^{Gibbs}(x'|x) = p(x'_i|x_{-i})\delta(x'_{-i} = x_{-i})$, causing both sides of Equation 1.11 to be equal to $p(x'_i|x_{-i})p(x_i|x_{-i})\delta(x'_{-i} = x_{-i})p(x_{-i})$.

1.2.2.3 Langevin Monte-Carlo

One way of choosing a good proposal distribution g in Metropolis-Hastings is based on the Langevin Monte-Carlo (LMC) algorithm [Bes94; RT96; RR98]. This method makes use of the available information about p to propose better candidate samples, based on the following stochastic differential equation (SDE), where dW represents the derivative of a Brownian motion:

$$dX = \frac{1}{2}\nabla_X \log p(X)dt + dW \tag{1.12}$$

This SDE has two interesting properties: it converges towards a stationary distribution that is equal to p, and $\nabla_X \log p(X)$ can be computed without knowing the normalization constant of p. Therefore, it suffices to numerically solve this equation to draw samples according to p. LMC however combines the numerical resolution with a Metropolis-Hastings acceptance step, in order to control for the numerical errors caused by discrete-time solving ofthe SDE.

The resulting process consists in, for some given time step Δt, proposing a candidate sample x' from a current state x as (with $\epsilon \sim \mathcal{N}(0,1)$ drawn after a standard Normal distribution):

$$x' = x + \frac{1}{2}\Delta t \nabla_x \log p(x) + \sqrt{\Delta t}\epsilon \tag{1.13}$$

This corresponds to sampling x' from a Normal distribution of mean $x+\frac{1}{2}\Delta t\nabla_x \log p(x)$ and of variance Δt, making the proposal density easy to compute:

$$g(x'|x) = \frac{1}{\sqrt{2\pi\Delta t}} \exp\left(-\frac{1}{2\Delta t}\left\|x' - x - \frac{1}{2}\Delta t\nabla_x \log p(x)\right\|^2\right) \tag{1.14}$$

This thus defines a proper Metropolis-Hastings scheme, that takes advantage of gradient information from $\log p$ to guide the exploration. As a consequence, it only applies on continuous variables. In general, the LMC enjoys a higher acceptance rate than a mainstream Metropolis Hastings mechanism, despite parameter Δt being very sensitive.

1.2.3 Variational inference

Another approach for approximating distribution p is Variational Inference [SJJ96; Bis+98]. It consists in reframing this approximation problem into an optimization problem: given some convenient class of distributions $\mathcal{Q}$, find $q^\star$ in $\mathcal{Q}$ closest to the target distribution p, where the distance is given by the Kullback-Leibler divergence:

$$q^\star = \arg\min_{q\in\mathcal{Q}} D_{KL}(q\|p) = \arg\min_q \mathop{\mathbb{E}}_{x\sim q} \log\frac{q(x)}{p(x)} \tag{1.15}$$

In the particular case where p is sought as a conditional distribution of z given some other variables x, i.e. the goal is to approximate $p(z|x)$, it takes its most known form[7]:

[7] While $\log p(z|x) = \log p(z,x) - \log p(x)$, note that the later term is a constant, and thus does not affect the optimization process.

$$q^\star = \arg\max_{q \in \mathcal{Q}} \left[H(q) + \mathop{\mathbb{E}}_{z \sim q} \log p(z, x) \right] \tag{1.16}$$

A main challenge in Variational Inference is the choice of the distribution class $\mathcal{Q}$, sufficiently expressive to yield a good approximation of the target distribution while permitting its efficient optimization.

Variational Inference is at the core of the presented research, and is presented and discussed in more detail in the following chapters, notably in Section 2.3.

1.3 Training from data

Informally, the question of learning a graphical model from data can be formulated as: given some dataset $\mathcal{D} \in \mathcal{X}^N$, find a graphical model p that best matches this dataset. This formulation leaves several questions unanswered. How to measure how well does a model p match a dataset? Is the underlying optimization problem a parametric one (find the parameters of a know graph) or a non-parametric one (find the graphical structure as well) ? Can the distribution involve variables that are not observed in the dataset, referred to as hidden variables?

The case where the distribution involves hidden variables, at the core of this manuscript, is considered in next chapters. In the remainder of this chapter, only observed variables are considered.

When learning probabilistic models, it is most often convenient to parameterize the search space with a vector of parameters, traditionally noted θ, concatenating the different vectors of parameters involved in the different elements of the model (the conditional distributions of a Bayesian network or the potentials of a Markov network). Therefore all elements formally depend on the same θ, although each term in the learning criterion only depends on a sub-vector of θ.

1.3.1 Maximum Likelihood training of a graphical model

One natural criterion for evaluating the quality of a graphical model on data is to measure the likelihood this model assigns to the dataset $p(\mathcal{D})$. Given some model class $\mathcal{C}$, the goal thus becomes to find a model p assigning the highest possible probability to the data:

$$p^\star = \arg\max_{p \in \mathcal{C}} p(\mathcal{D}) \tag{1.17}$$

Under the assumption of independent and identically distributed samples (iid), one has $p(\mathcal{D}) = \prod_{x \in \mathcal{D}} p(x)$, making $p(\mathcal{D})$ very small for large $\mathcal{D}$. For numerical stability and general convenience, it is thus customary to instead consider the negative log-likelihood (NLL) of the data:

$$p^\star = \arg\min_{p \in \mathcal{C}} - \log p(\mathcal{D}) = \arg\min_{p \in \mathcal{C}} \sum_{x \in \mathcal{D}} - \log p(x) \tag{1.18}$$

This can be reformulated as an expectation on the dataset:

$$p^\star = \arg\min_{p \in \mathcal{C}} \mathop{\mathbb{E}}_{x \in \mathcal{D}} - \log p(x) \tag{1.19}$$

Along this line, this formulation provides a justification for maximum likelihood training: in the large sample limit, the empirical expectation becomes the expectation w.r.t. the underlying distribution generating the dataset. Equation 1.19 then becomes equivalent to seeking the distribution p that minimizes the cross-entropy from $p_\mathcal{D}$ to p, which is reached by $p^\star = p_\mathcal{D}$. Accordingly, whenever sufficient data is available, maximum likelihood optimization can be used for fitting the parameters of the graphical model.

1.3.1.1 Maximum Likelihood for Bayesian networks

In the Bayesian Network framework, maximum likelihood optimization can conveniently be applied, using the distribution factorization (Equation 1.1):

$$\mathop{\mathbb{E}}_{x \in \mathcal{D}} - \log p(x) = \sum_i \mathop{\mathbb{E}}_{x \in \mathcal{D}} - \log p(x_i | \pi(X_i)) \tag{1.20}$$

Each node/variable of a Bayesian Network can thus be trained independently, only requiring to access the values of the variable and its parents. On large Bayesian Networks, this makes it possible to parallelize the training procedure, with significant computational gains on sparse graphs.

1.3.1.2 Maximum Likelihood for Markov Models

Markov Models do not decompose as nicely. From Equation 1.8, it comes:

$$\mathop{\mathbb{E}}_{x \in \mathcal{D}} - \log p(x) = \mathop{\mathbb{E}}_{x \in \mathcal{D}} E(x) + \log Z = \mathop{\mathbb{E}}_{x \in \mathcal{D}} \sum_C f_C(x_C) + \log Z \tag{1.21}$$

Energy terms f_C do separate from one another, but remain coupled through the normalization constant Z, which depends on all the f_C.

While the normalization constant can be omitted in inference tasks, it is an integral part of the training process, and it is necessary to deal with it. Cases where this normalization constant can be explicitly computed are rare. In some cases however, the model can be trained without explicitly computing Z.

Let parametric model p_θ be defined by the energy function $E(x; \theta)$ with $\theta \in \mathbb{R}^M$, this model can be trained by gradient descent; the trick is that we can compute the full gradient of the NLL with regards to θ without computing the value of Z: simple algebraic manipulations yield $\nabla_\theta \log Z = - \mathbb{E}_{x \sim p_\theta} \nabla_\theta E(x; \theta)$, and finally the full gradient:

$$\nabla_\theta - \log p_\theta(\mathcal{D}) = \mathop{\mathbb{E}}_{x \in \mathcal{D}} \nabla_\theta E(x; \theta) - \mathop{\mathbb{E}}_{x \sim p_\theta} \nabla_\theta E(x; \theta) \tag{1.22}$$

Although the second expectation cannot in general be exactly computed, it can be empirically approximated using the inference methods developed in the previous section.

1.3.2 Bayesian regularization and Maximium A-Posteriori

The Maximum Likelihood method is well founded under the assumption of a very large amount of data, that rarely holds in practice. In this context, one can turn to Bayesian inference, learning a probability distribution over parameters θ to account for the uncertainty due to the lack of data. This is done based on Bayes Theorem:

$$\mathbb{P}(\theta|\mathcal{D}) = \frac{\mathbb{P}(\mathcal{D}|\theta)\mathbb{P}(\theta)}{\mathbb{P}(\mathcal{D})} \tag{1.23}$$

$\mathbb{P}(\theta)$ is a prior distribution over the parameters and assumed given as a model hypothesis. $\mathbb{P}(\mathcal{D}|\theta) = p_\theta(\mathcal{D})$ the likelihood of the data according to the model of parameters θ. Computing the full posterior distribution $\mathbb{P}(\theta|\mathcal{D})$ is in general intractable, in particular due to the normalization factor $\mathbb{P}(\mathcal{D})$. One thus commonly resorts to only finding its optimal value, after the Maximum A-Posteriori (MAP) procedure:

$$\theta^\star = \arg\max_\theta \mathbb{P}(\theta|\mathcal{D}) \tag{1.24}$$

In NLL terms, it comes:

$$\theta^\star = \arg\min_\theta \left[-\log p_\theta(\mathcal{D}) - \log \mathbb{P}(\theta)\right] \tag{1.25}$$

MAP training is thus very similar to Maximum Likelihood, with the addition of the $-\log \mathbb{P}(\theta)$ term in the optimization procedure, acting as a regularizer. Under the iid assumption, dividing Equation 1.25 by the size N of the dataset, it comes:

$$\theta^\star = \arg\min_\theta \left[\mathop{\mathbb{E}}_{x\in\mathcal{D}} \left[-\log p_\theta(x)\right] - \frac{1}{N}\log \mathbb{P}(\theta)\right] \tag{1.26}$$

This shows how the prior term impact naturally decreases as the amount of data increases.

1.3.3 Structure learning

There exists a large body of work devoted to learning the structure of graphical model. refSurvey on learning GM structure As this issue is outside the scope of the presented work, only an overview of the main approaches designed for BN structure learning will be presented, used as an inspiration for structure learning of Deep Learning based models in Section 4.3.

The goal is to learn both the graph structure $\mathcal{G}$ and its parameters given a dataset $\mathcal{D}$. In a fully Bayesian perspective, we are seeking the posterior $\mathbb{P}(\mathcal{G}|\mathcal{D})$ or a workable estimate of its mode. Bayes' Theorem decomposes this posterior as a prior over the graphs $\mathbb{P}(\mathcal{G})$ and a likelihood term $\mathbb{P}(\mathcal{D}|\mathcal{G})$, which can be decomposed as an integral over $\theta_\mathcal{G}$ the parameters of the graphical model defined by $\mathcal{G}$:

$$\mathbb{P}(\mathcal{D}|\mathcal{G}) = \int_{\theta_\mathcal{G}} \mathbb{P}(\mathcal{D}|\theta_\mathcal{G}, \mathcal{G})\mathbb{P}(\theta_\mathcal{G}|\mathcal{G})\, d\theta_\mathcal{G} \tag{1.27}$$

In order to learn $\mathcal{G}$, some scoring measures are designed and used as proxies for this likelihood term (the prior is supposed to be given, as problem dependent). The

main three scoring measures are: the Bayesian Information Criterion, the Akaike Information Criterion, and the Bayesian Dirichlet algorithm.

Bayesian Information Criterion The Bayesian Information Criterion (BIC) [Sch78], is an asymptotic approximation of $\log \mathbb{P}(\mathcal{D}|\mathcal{G})$ in the large sample limit. Let $k_{\mathcal{G}}$ be the number of learnable parameters in the model induced by $\mathcal{G}$, the BIC is formulated as:

$$BIC(\mathcal{G}) = \max_{\theta_{\mathcal{G}}} [\log \mathbb{P}(\mathcal{D}|\theta_{\mathcal{G}}, \mathcal{G})] - \frac{1}{2} k_{\mathcal{G}} \log |\mathcal{D}| \tag{1.28}$$

It is the optimal log-likelihood reached by the model, with a penalization proportional to the number of parameters, and logarithmically growing with the dataset size. The fact that this score does not depend on the exact form of the prior corresponds to the large sample limit assumption. Note also that the likelihood of the dataset grows as $\mathcal{O}(|\mathcal{D}|)$; the likelihood term dominates on very large datasets.

Akaike Information Criterion The Akaike Information Criterion (AIC) [Aka74] is constructed on information theoretic arguments, aimed to find a model achieving the minimal information loss w.r.t. the real distribution.

The information loss is measured with the Kullbak-Leibler divergence from $\mathbb{P}(\mathcal{D}|\mathcal{G})$ to $\mathbb{P}(\mathcal{D})$: $D_{KL}(\mathbb{P}(\mathcal{D})\|\mathbb{P}(\mathcal{D}|\mathcal{G}))$. Like BIC, the AIC considers an approximation in the large sample limit:

$$AIC(\mathcal{G}) = \max_{\theta_{\mathcal{G}}} [\log \mathbb{P}(\mathcal{D}|\theta_{\mathcal{G}}, \mathcal{G})] - k_{\mathcal{G}} \tag{1.29}$$

A main difference compared to BIC is that the penalization does not depend on the dataset size. An extensive comparison of the strengths and weaknesses of both methods is given in [BA98; Yan05; Vri12].

Bayesian Dirichlet The Bayesian Dirichlet method [CH92] focuses on Bayesian Networks with discrete variables. In this case all sub-likelihoods $p_\theta(x_i|\pi(X_i))$ are categorical distributions, and admit a Dirichlet distribution as conjugate prior. If such a prior is chosen over the parameters, the whole likelihood factors and the graph likelihood $\mathbb{P}(\mathcal{D}|\mathcal{G})$ can be computed analytically.

The question of how to efficiently explore the space of potential graphs, and find the graph with the best score is known to be NP-hard [CHM04], but several approximate methods give good results [SM06; Jaa+10; CJ11].

1.4 Summary

This chapter presented the *Probabilistic Graphical Models* framework, focusing on the *Bayesian Networks* at the core of the presented work. Graphical models allow one to decompose the relations between variables in a model into a sparse set of local dependencies, where each variable only directly depends on a few others. This both enables a compact representation of the model, and makes it possible to locally reason about them to some extent.

When the goal is to answer queries, e.g. computing conditional distributions in the form $P(Y|X)$ for some subset Y of variables, such models usually require the non-trivial marginalization of some variables. Several algorithms take advantage of the graphical structure to answer the query without needing to produce the full joint distribution, such as the *Variable Elimination Algorithm* or the *Belief propagation Algorithm*. Another approach is the use of *Markov Chain Monte-Carlo* methods to directly produce approximate samples from the sought distribution.

When the goal is to learn graphical models from datasets, one must distinguish the building of the graph structure itself (usually tackled using complexity minimization) and the learning of its parameters. In the last case, the main approaches rely on *Maximum Likelihood*, achieving the (approximate) Bayesian inference of the parameters of the model given its structure.

Part I
Latent Variables modeling

Chapters

Chapter 2
Latent Variables and the ELBO

Contents

In many cases, graphical models involve extra variables beyond those observed in the dataset. These unobserved variables are referred to as "latent" or "hidden" variables, and the models containing such variables are called *Latent Variables Models* (LVMs). This chapter focuses on the motivations for using latent variables, and how to train LVMs.

2.1 Latent variables as a modeling tool

The design of a graphical model is generally driven by our *a priori* knowledge about the problem at hand, implying that the model will generally be designed based on epistemic and computational considerations. Epistemic information informs model design by mapping variables to interpretable quantities from the real-world process under study. Computational considerations incline to retain structures that can easily be manipulated by our current algorithms and computers, while remaining expressive enough to represent the data. Model design must find some trade-off between both types of considerations, and this might lead to include in the model variables that don not map to any observed data.

In this section we shall explore the use of two types of latent variables: interpretable ones and abstract ones.

2.1.1 Interpretability

One immediate reason why one would integrate an interpretable latent variable in a graphical model is if this variable represents a quantity of interest, to be predicted from the data. In this situation the structure of the model is generally based on strong epistemic considerations.

Similarly, unobserved quantities are not rare in scientific models[1], and many of these models can be studied though the lens of graphical models.

In this context, observed and latent variables are generally related in known ways. For instance, in a Bayesian Network, the distribution of an observed variable conditionally to a latent one might be given *a priori*; or at least its structure might be known and depend on a few parameters. For instance, the gravitational law in Physics specifies how the acceleration depends on the mass of a system, via the gravity constant.

In many other cases, latent variables are interpretable by construction. Taking for example *Latent Dirichlet Allocation* for modeling text documents by linking them to topics via their individual words. Each topic k is associated to a latent variable ϕ_k representing the distribution of words associated to it, and each document i is associated to a latent variable θ_i representing the distribution of topics associated to it. Then within a document, each word j is associated to a latent variable $z_{i,j}$ representing its topic and an observed variable $w_{i,j}$ representing the word itself. From this description the sampling process is derived as follows. Each topic k has its word distribution sampled from $p(\phi_k)$. The topic distribution of document i is sampled from $p(\theta_i)$. Then for each word, the topic of that word is sampled from $p(z_{i,j}|\theta_i)$, and finally the word itself is sampled following $p(w_{i,j}|\phi_{k=z_{i,j}})$, yielding the factorization of the whole model and its graphical representation (Figure 2.1):

$$p(\{\phi_k\}_k, \{\theta_i\}_i, \{z_{i,j}\}_{i,j}, \{w_{i,j}\}_{i,j}) = \prod_{i,k}\left[p(\theta_i)p(\phi_k)\prod_j\left[p(z_{i,j}|\theta_i)p(w_{i,j}|\phi_{k=z_{i,j}})\right]\right] \tag{2.1}$$

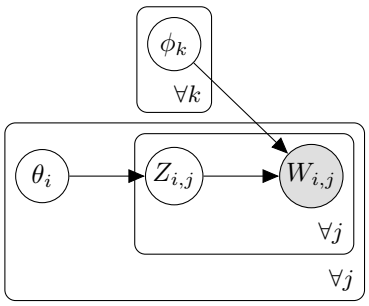


Figure 2.1: Graphical representation of Latent Dirichlet Allocation, the rectangles, referred to as *plates*, represent variables that are repeated.

We generally have an *a priori* understanding of how an interpretable latent variable should behave, either from domain knowledge, theoretical analysis, or preliminary analysis of the data. This understanding can be leveraged to monitor the

[1] We can for example think of the micro-states in statistical physics, or the actual number of contaminated persons (as opposed to the number of reported cases) in an epidemiological model.

behavior of the model and to help training it, depending on the context; for instance (non exhaustive list) the inference algorithm can be adapted to avoid numerical instability if the range of a variable is known in advance, the appropriate number of mixtures in a model can be pre-estimated by eyeballing it from a plot of the data, or appropriate sampling methods can be used depending on the expected rarity of the relevant values.

2.1.2 Abstract variables for expressiveness

In some situations, one might want to consider latent variables with no *a priori* interpretation. Such variables, more loosely coupled with other variables, often serve to drastically increase the expressiveness of a model while keeping its computational complexity controlled.

To illustrate this principle, let us consider a finite Gaussian Mixture. Each component i is defined by a mean μ_i and a variance σ_i^2. The family of all Gaussian mixtures of a given size is significantly more expressive than a single Gaussian distribution while the computational cost associated with it remains very reasonable, proportional to the number of components. Even when the information of which component a given datapoint belongs to is not of interest, using a mixture can be a relevant way of increasing the space of distributions that the model can represent[2], as illustrated by Figure 2.2.

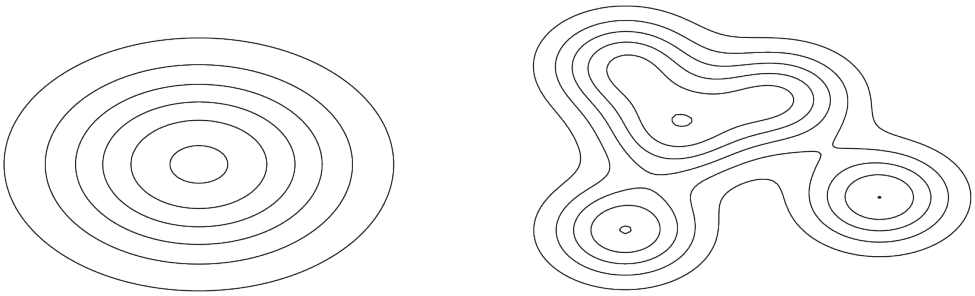


Figure 2.2: Representation of a 2D Gaussian distribution (left) compared to a mixture of 5 Gaussian distributions (right). While the single Gaussian has a simple elliptic shape, Gaussian mixtures can express much more complex distirbutions.

This idea can be generalized though the principle of marginalization. Defining a joint distribution $p(x, z)$ over an observed variable x and a latent variable z might yield a complex marginal distribution[3] $p(x) = \int_z p(x, z)dz$. Such an approach usually involves a latent distribution over the marginalized variable $p(z)$ and a family of simple conditional distributions $p(x|z)$: a graphical model with a latent variable. In this case, latent variable z is *abstract*: it does not have any intrinsic meaning,

[2] In particular, Gaussian functions being universal approximators [MS96], a Gaussian mixture with a sufficiently large number of component would be able to represent any reasonable distribution.

[3] For example, any distribution with a known cumulative distribution function F can be expressed this way: let z by a uniform variable over $[0; 1]$ and x deterministically derived from it: $x = F^{-1}(z)$. $p(z)$ is a simple uniform distribution and $p(x|z)$ is a simple constant distribution, yet the marginal $p(x)$ is the distribution defined by F, which can be arbitrarily complex.

and its only role is to be marginalized out to increase the power of the model over the variable of interest x while keeping the computational cost low. One can then think of $p(x)$ as a mixture model with an infinite (and even continuous) number of components, indexed by z.

2.2 Training challenges of Latent Variable Models

The introduction of latent variables in a graphical model has a significant impact on the training procedures: latent variables are not observed, and appropriate losses must thus be considered. A common approach is based on marginalization, maximizing the likelihood of the observed data according to the (marginalized) model. The marginalization procedure is however generally nontrivial, and poses additional questions in the case of abstract latent variables. By convention, observed variables (respectively latent variables) are noted x_i (resp. z_j) in the following.

2.2.1 Likelihood and marginalization

The graphical model specifies the joint likelihood over all variables $p_\theta(x_1, \dots, x_n, z_1, \dots, z_k)$. Training this model relies on estimating the likelihood $p_\theta(x_1, \dots, x_n)$ of the data over the observed variables; this estimation is at the core of all training procedures, either based on maximum likelihood, MAP, or fully Bayesian training. The likelihood over observed variables is estimated by marginalization over the latent ones:

$$p_\theta(x_1, \dots, x_n) = \int_{z_1,\dots,z_k} p_\theta(x_1, \dots, x_n, z_1, \dots, z_k) dz_1 \dots dz_k \tag{2.2}$$

In some cases, this marginalization can be computed exactly, e.g. in a finite mixture, and the training procedure de scribed in Section 1.3 directly applies. In most cases however, such computation is either intractable or prohibitively expensive, and approximate methods are needed.

In the case of a graphical model parameterized by continuous variables, the gradient of the likelihood can be estimated by Monte-Carlo sampling, by using the following identity (where $\mathbf{x}$ thereafter stands for the set of all observed variables, and $\mathbf{z}$ the set of all latent ones):

$$\nabla_\theta \log p_\theta(\mathbf{x}) = \mathop{\mathbb{E}}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x})} \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) \tag{2.3}$$

The expectation over $p_\theta(\mathbf{z}|\mathbf{x})$ can be estimated using sampling methods (Section 1.2.2). In a large range of problems, the model is trained using the Expectation-Maximization algorithm (see Section 2.4).

2.2.2 Abstract variables and identifiability

Abstract variables usually raise additional challenges, due to the fact that they are often underconstrained. For any model $p_\theta(x, z)$ with z an abstract latent variable, it is generally possible to define a number of equivalent models $p_{\tilde{\theta}}(x, \tilde{z})$ through considering $\tilde{z} = \phi(z)$ with ϕ a bijection on the latent space. Accordingly, the parameter vector θ is not unique, that is, the model is non-identifiable with regard to

its observed variables. Typically in the context of a mixture model, any permutation of the mixture components yields an equivalent model.

This non-identifiability raises difficulties of different types. With respect to the training step, if there exist multiple parameters θ yielding the same marginal likelihood $p_\theta(x)$, then the underlying optimization problem is not well-defined, admitting numerous local optima and offering less guarantees depending on the optimization algorithm, e.g. gradient ascent or estimation of a Bayesian posterior over the parameters[4]. With respect to the interpretation of the results, different models are learned in different runs, hindering the interpretation of the latent variables. While the trained model might accurately represent the observation density $p_\theta(x)$, it offers little insight into the underlying generative process.

Some partial identifiability property can be enforced through setting constraints on either the relationships between observed and latent variables, or the latent distribution $p_\theta(z)$. Such constraints can be guided by *a priori* knowledge on the problem, or by the desired properties of the learned latent representation. Both approaches, at the core of the thesis, are investigated in depth in respectively Part II and Part III.

2.3 Variational Inference

Variational inference is an umbrella for a family of approaches, tackling the approximation of probability distribution through solving an optimization problem. In its most general form, it can be formulated as finding the distribution within the selected distribution space, that is closest to the target distribution in terms of the Kullback-Leibler divergence[5]:

$$q^\star = \arg\min_{q\in\mathcal{Q}} D_{KL}(q\|p) = \arg\min_{q\in\mathcal{Q}} \mathop{\mathbb{E}}_{z\sim q} \log \frac{q(z)}{p(z)} \tag{2.4}$$

A significant advantage of this formulation is that the target distribution p does not need to be sampled, nor does it need to be normalized. This makes the approach particularly applicable to graphical models, in particular in the context of conditional queries where normalization raises critical difficulties.

Given a model $p(\mathbf{x}, \mathbf{z})$, with $\mathbf{x}$ the observed variables and $\mathbf{z}$ the latent ones. Assuming a given value of $\mathbf{x}$, the goal is to approximate the conditional distribution $p(\mathbf{z}|\mathbf{x})$ with $q(z)$. Minimizing the KL divergence from it to the approximating distribution $q(z)$ yields the following optimization problem (constant terms relative to $\mathbf{z}$ omitted):

$$q^\star_{\mathbf{x}} = \arg\max_{q\in\mathcal{Q}} \left[H(q) + \mathop{\mathbb{E}}_{\mathbf{z}\sim q} \log p(\mathbf{x}, \mathbf{z}) \right] \tag{2.5}$$

Accordingly, the optimization of q reflects a trade-off between maximizing the likelihood of the model $\log p(\mathbf{x}, \mathbf{z})$, and maximizing the entropy $H(q)$. For any given $\mathbf{x}$, the global optimum is reached for $q^\star_{\mathbf{x}}(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$.

[4]The Bayesian posterior over the parameters of a non-identifiable model does not converge to a point measure in the large sample limit, while it does converge for an identifiable model.

[5]The choice of the KL-divergence as an optimization objective is in part driven by the fact that it makes the optimization problem rather easy to manipulate and does not require the distribution p to be sampled nor normalized.

2.3.1 The Evidence Lower Bound

Equation 2.5 can be reformulated to express nice insights into the structure of the sought solution. Expanding the KL divergence between $q(z)$ and $p(z|x)$ gives:

$$D_{KL}(q_{\mathbf{x}}(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})) = \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}}\left[\log \frac{q_{\mathbf{x}}(\mathbf{z})}{p(\mathbf{z}|x)}\right] \tag{2.6}$$

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}}\left[\log \frac{q_{\mathbf{x}}(\mathbf{z})p(x)}{p(\mathbf{x},\mathbf{z})}\right] \tag{2.7}$$

$$= \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}}\left[\log \frac{q_{\mathbf{x}}(\mathbf{z})}{p(\mathbf{x},\mathbf{z})}\right] + \log p(\mathbf{x}) \tag{2.8}$$

This can be reformulated as:

$$\log p(\mathbf{x}) = \underbrace{\mathop{\mathbb{E}}_{z\sim q_{\mathbf{x}}}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_{\mathbf{x}}(\mathbf{z})}\right]}_{\text{ELBO}} + D_{KL}(q_{\mathbf{x}}(\mathbf{z})\|p(\mathbf{z}|\mathbf{x})) \tag{2.9}$$

As the KL-divergence is always positive, the first term of the right-hand side of Equation 2.9 sets a lower bound for the log-probability the model assigns to the observed value $\mathbf{x}$, $\log p(\mathbf{x})$. This log-probability is also called the *evidence* of the model, hence giving the name *Evidence Lower Bound (ELBO)* [BG02; GHB12; KW14; RMW14]. Optimizing the ELBO thus supports both approximations of $p(z|x)$ and $\log p(x)$, for any given observation x.

The ELBO can be written in three different ways:

$$\mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q_{\mathbf{x}}(\mathbf{z})}\right] = H(q_{\mathbf{x}}) + \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}} \log p(\mathbf{z},\mathbf{x}) = \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}} \log p(\mathbf{x}|\mathbf{z}) - D_{KL}(q_{\mathbf{x}}(\mathbf{z})\|p(\mathbf{z}))) \tag{2.10}$$

Depending on the context and which terms can be exactly computed, or approximated more efficiently, one formulation can be more convenient than the others. Most approaches in the literature thus focus on one of these three forms.

Overall, the ELBO maximization defines a training objective for the graphical model p [BG06; RGB14]: once decently maximized with regard to q, the value of the ELBO can be used as a estimation of the model evidence $\log p(\mathbf{x})$, and used as an optimization objective. When considering some training dataset $\mathcal{D}$, one associates to each sample $\mathbf{x}$ a distribution $q_{\mathbf{x}}(\mathbf{z})$, optimized to approximate the conditional distribution $p(\mathbf{z}|\mathbf{x})$. The joint ELBO then reads:

$$\log p(\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \log p(\mathbf{x}) \geq \sum_{\mathbf{x}\in\mathcal{D}} \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}} \log \frac{p(\mathbf{x},\mathbf{z})}{q_{\mathbf{x}}(\mathbf{z})} \tag{2.11}$$

Maximizing the ELBO regarding both p and the set $\{q_{\mathbf{x}}\}_{\mathbf{x}\in\mathcal{D}}$ achieves the maximum likelihood training of model p. The key issues are the selection of model class $\mathcal{Q}$ containing the $q_{\mathbf{x}}$, and the optimization methods used to find $q_{\mathbf{x}}$ and p.

The above sum can also be maximized as an expectation over the dataset, as done in Section 1.3.1:

$$\frac{1}{|\mathcal{D}|}\log p(\mathcal{D}) = \mathop{\mathbb{E}}_{\mathbf{x}\in\mathcal{D}} \log p(\mathbf{x}) \geq \mathop{\mathbb{E}}_{\mathbf{x}\in\mathcal{D}} \mathop{\mathbb{E}}_{\mathbf{z}\sim q_{\mathbf{x}}} \log \frac{p(\mathbf{x},\mathbf{z})}{q_{\mathbf{x}}(\mathbf{z})} \tag{2.12}$$

When considering an iterative optimization approach, e.g. based on gradient descent, the expectation over the dataset can be approximated by Monte-Carlo, considering a random subsample (mini-batch) instead of the whole dataset. Only the $q_{\mathbf{x}}$ associated with the sampled datapoints are then computed in each iteration, drastically speeding-up the training of the model on large datasets [Hof+13].

2.3.2 Mean-Field approximation for posteriors

The simplest and most convenient method, known as mean field approximation [SJJ96], considers q as the product of distributions of a single latent variable $q(z) = \prod_j q_j(z_j)$. Each component q_j is sought as a simple (e.g. Gaussian or categorical) distribution, depending on the type of the latent variable. In this case, the entropy term of the ELBO (Equation 2.10) reads as $H(q) = \sum_j H(q_j)$, making its analytical expression generally possible. Likewise, the mean field approximation setting easily supports the Monte-Carlo estimation of $\mathbb{E}_{z\sim q} \log p(z, x)$ in general.

The main and significant limitation of mean field approximation is that it can only efficiently fit mono-modal distributions: only a single mode of the sought distribution is decently approximated.[6]. When considering multi-modal $p(z|x)$, one thus rather models q as a mixture of factorized distributions[Bis+98; GHB12], still supporting an easy sampling and computation of $\log q(z)$. While entropy $H(q)$ is no longer defined in closed form, its Monte-Carlo approximation still applies.

The particular case of continuous (Gaussian) distributions has been further explored. On the one hand, factorized Gaussian models, with diagonal covariance matrix, are often too poor and do not support correlations among latent variables. On the other hand, multivariate Gaussian distributions have a quadratic complexity in the number of latent variables. A trade-off is offered by considering low-rank covariance matrices [ONS18], enabling to capture the main correlations at a reasonable cost.

Note that Gaussian distributions offer many possibilities for mixture approximations of the posterior thanks to their universal approximation property. Adaptive mixture constructions have been proposed along this line, dynamically adding components to the mixture as needed to improve the approximation [Guo+17; MFA17].

2.3.3 Flexible posterior approximation using Normalizing Flows

In order to construct approximations $q(z)$ to the conditional distribution $p(z|x)$, a quite different approach is based on the so-called Normalizing Flows [RM15], where $q(z)$ is based on a standard distribution (e.g. centered Gaussian) π, and z is obtained by applying some transformation $z = f(\epsilon)$, with $\epsilon\ \pi$ a fixed distribution of noise. The optimization problem thus consists in finding an appropriate f.

In order to compute the density $q(z)$, f is required to be invertible. Under this assumption, $q(z)$ can be computed in close form, involving the determinant of the Jacobian matrix $\nabla_\epsilon f$ of f:

[6] If each q_i is a single-mode distribution, then the complete q will have a single mode as well. But even if the q_i can be multi-modal, this forces the modes of each dimension to be independent, which is unlikely to match the real structure of $p(z|x)$, causing the inference model to still choose a single mode and fit it.

$$q(z) = \pi(\epsilon)|\nabla_\epsilon f|^{-1} \tag{2.13}$$

This formula and the choice of a class of functions for which the determinant can easily be computed is at the core of Normalizing Flows (NF). Due to the invertibility constraint, NF only apply to continuous variables. In the NF framework, the ELBO reads:

$$\log p(x) \geq H(\pi) + \mathop{\mathbb{E}}_{z\sim q}\Big[\log|\nabla_\epsilon f| + \log p(x,z)\Big] \tag{2.14}$$

Within this generic NF framework, a number of approaches have been developed, using carefully structured artificial neural networks (e.g. based on autoregressive principles) to model the function f [RM15; DKB15; LW17; Kin+17; Ber+19; Dur+19]. These approaches offer versatile approximations q, well-suited to models with abstract continuous latent variables.

2.4 Model training with Expectation-Maximization

The famed Expectation-Maximization (EM) algorithm is an iterative algorithm, used to train a parametric model p_θ by alternating an "Expectation" step and a "Maximization" step [DLR77]. As will be seen in Section 2.4.2, it is closely related to ELBO training.

Let $\theta^{(t)}$ denote the parameter vector of the model at step t. The Expectation step relies on a score function $\mathcal{L}$, which assigns a score to every possible θ in the parameter space:

$$\mathcal{L}\left(\theta;\theta^{(t)}\right) = \sum_{x\in\mathcal{D}}\left[\mathop{\mathbb{E}}_{z\sim p_{\theta^{(t)}}(z|x)}\log p_\theta(x,z)\right] \tag{2.15}$$

Note that in particular $\mathcal{L}\left(\theta^{(t)};\theta^{(t)}\right) = \log p_{\theta^{(t)}}(\mathcal{D})$ is the evidence of the model for the vector of parameters $\theta^{(t)}$.

Then the Maximization step determines the value of θ maximizing $\mathcal{L}$, and sets $\theta^{(t+1)}$ to this optimal value.

$$\theta^{(t+1)} = \arg\max_\theta \mathcal{L}\left(\theta;\theta^{(t)}\right) \tag{2.16}$$

The Expectation-Maximization scheme is iterated until convergence of θ^t.

The justification of the algorithm goes as follows. $\mathbb{E}_{z\sim q}\log p(x,z)$ is maximal w.r.t. p when $q(z) = p(z|x)$. Therefore, the function $\eta \to \mathcal{L}\left(\theta^{(t+1)};\eta\right)$ reaches its maximum for $\eta = \theta^{(t+1)}$. From the definition of the Maximization step, it then comes:

$$\log p_{\theta^{(t+1)}}(\mathcal{D}) = \mathcal{L}\left(\theta^{(t+1)}; \theta^{(t+1)}\right) \tag{2.17}$$
$$\geq \mathcal{L}\left(\theta^{(t+1)}; \theta^{(t)}\right) \tag{2.18}$$
$$\geq \mathcal{L}\left(\theta^{(t)}; \theta^{(t)}\right) \tag{2.19}$$
$$\log p_{\theta^{(t+1)}}(\mathcal{D}) \geq \log p_{\theta^{(t)}}(\mathcal{D}) \tag{2.20}$$

In other words, the model evidence (the log-likelihood of the data) monotonically increases in each EM iteration. The algorithm thus is guaranteed to converge toward a local optimum, except if the evidence can diverge towards $+\infty$; in this latter case, the algorithm is prone to severe overfitting.

2.4.1 Exact inference in Gaussian Mixture Models

EM classically considers Gaussian Mixture Models as model space, as this space allows analytic solving of both expectation and maximization steps.

Let us consider a mixture model of K multivariate Gaussian distributions, represented as a graphical model with two variables: I the mixture index, and X the observed value. The model thus factors as $p_\theta(x, i) = p_\theta(x|i)p_\theta(i)$. The latent distribution is a categorical one with K values. Let α_i denote its parameters, that is, $p_{\theta^{(t)}}(i) = \alpha_i^{(t)}$. The observation distribution is a multivariate Gaussian of mean μ_i and covariance matrix Σ_i: $p_{\theta^{(t)}}(x|i) = \mathcal{N}(x; \mu_i^{(t)}, \Sigma_i^{(t)})$. The overall parameter vector is thus composed of the K weights of the latent distribution, K mean vectors, and K covariance matrices: $\theta = (\{\alpha_i\}_{i=1}^K, \{\mu_i\}_{i=1}^K, \{\Sigma_i\}_{i=1}^K)$.

In the expectation step, $p_{\theta^{(t)}}(i|x)$ is computed for each datapoint, using Bayes Theorem:

$$p_{\theta^{(t)}}(i|x) = \frac{p_{\theta^{(t)}}(x|i)p_{\theta^{(t)}}(i)}{p_{\theta^{(t)}}(x)} = \frac{\alpha_i^{(t)}\mathcal{N}(x; \mu_i^{(t)}, \Sigma_i^{(t)})}{\sum_{j=1}^K \alpha_j^{(t)}\mathcal{N}(x; \mu_j^{(t)}, \Sigma_j^{(t)})} = w_i^{(t)}(x) \tag{2.21}$$

The objective function of the maximization step reads:

$$\mathcal{L}\left(\theta; \theta^{(t)}\right) = \mathop{\mathbb{E}}_{x\in\mathcal{D}} \sum_{i=1}^K w_i^{(t)}(x)\Big[\log \alpha_i + \log \mathcal{N}(x; \mu_i, \Sigma_i)\Big] \tag{2.22}$$

Note that, thanks to the factorization of the model, parameters α, and each (μ_i, Σ_i) pair can be optimized independently: the latent parameter α_i is set to the mass ratio of the posteriors for the i-th cluster across the dataset and each Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ is fitted by maximum likelihood to the dataset with importance weights $w_i^{(t)}(x)$:

$$\alpha_i^{(t+1)} = \mathop{\mathbb{E}}_{x\in\mathcal{D}} w_i^{(t)}(x) \tag{2.23}$$
$$\mu_i^{(t+1)} = \mathop{\mathbb{E}}_{x\in\mathcal{D}} w_i^{(t)}(x)\, x \tag{2.24}$$
$$\Sigma_i^{(t+1)} = \mathop{\mathbb{E}}_{x\in\mathcal{D}} w_i^{(t)}(x)(x - \mu_i^{(t+1)})(x - \mu_i^{(t+1)})^T \tag{2.25}$$

Along these equations, the algorithm iteratively computes the $|\mathcal{D}| \times K$ matrix of weights $w_i^{(t)}(x)$, and updates the parameters α, μ, Σ, until convergence.

2.4.2 Approximate inference with Variational EM

In some cases, the conditional distribution $p_{\theta^{(t)}}(z|x)$ cannot be exactly computed, preventing the objective function $\mathcal{L}$ from being computed as well. In this context, the EM algorithm can be reframed using the ELBO, defining a Variational EM Algorithm. Note that a number of EM variants have been referred to as "Variational EM"; in the following, Variational EM is meant as a generic variational approach based on the EM principles.

As described in Section 2.3.1, the family $\mathcal{Q}^{(t)}$ made of all $q_x^{(t)}$, variational approximation of $p_{\theta^{(t)}}(z|x)$ for each datapoint x, is maintained.

Given the family $\mathcal{Q}^{(t)}$ and the model parameters $\theta^{(t)}$, the ELBO is used to lower bound the model evidence:

$$\log p_{\theta^{(t)}}(\mathcal{D}) \geq \sum_{x \in \mathcal{D}} \mathop{\mathbb{E}}_{z \sim q_x^{(t)}} \log \frac{p_{\theta^{(t)}}(x,z)}{q_x^{(t)}(z)} = \mathcal{L}_{ELBO}(\mathcal{Q}^{(t)}; \theta^{(t)}) \tag{2.26}$$

The two steps of the EM algorithm then naturally appear from this formulation: the Expectation step maximizes $\mathcal{L}_{ELBO}$ with regard to $\mathcal{Q}$, and the Maximization step optimizes it with regard to θ:

$$\mathcal{Q}^{(t+1)} = \arg\max_{\mathcal{Q}} \mathcal{L}_{ELBO}(\mathcal{Q}; \theta^{(t)}) \quad ; \quad \theta^{(t+1)} = \arg\max_{\theta} \mathcal{L}_{ELBO}(\mathcal{Q}^{(t+1)}; \theta) \tag{2.27}$$

This formulation makes it clear that the ELBO monotonically increases as the algorithm unfolds. Variational EM thus appears as a special case of ELBO training, alternatively optimizing the variational approximations q_x and the model p_θ.

When the exact conditional distribution of the latent variables can be exactly computed, the ELBO boils down to the model evidence, and Variational EM coincides with EM.

2.5 Summary

This chapter describes why and how to use *latent variables* in graphical models. Latent variables, not observed in the data, are meant to either reflect hidden factors (considering the phenomenon at hand as a partially observed one) or increase the expressive power of the model while keeping its computational cost low.

Unless latent variables can be analytically marginalized to compute the model evidence over the dataset $\log p_\theta(\mathcal{D})$, one resorts to estimating it using the *Evidence Lower Bound (ELBO)* with a family of auxiliary inference distributions $\{q_x\}_{x \in \mathcal{D}}$ (Equation 2.11). Optimizing the ELBO with regards to both p_θ and $\{q_x\}_{x \in \mathcal{D}}$ allows one to both train the generative model by likelihood optimization, and build an approximation of the posterior distribution of the latent variables given the observed ones, as the set $\{q_x\}_{x \in \mathcal{D}}$.

Notably, the *Expectation-Maximization (EM)* algorithm can be seen as a special case of this principle, where the model and the inference distributions are trained

iteratively, maximizing the ELBO with regards to either one alternatively. The mainstream EM algorithm corresponds to the case where both maximization steps can be solved analytically.

Chapter 3
Deep Latent Variable Models

Contents

Latent Variable Models took a new turn with the rise of Deep Learning, yielding the famed Deep Latent Variable Models. Note that "deep" refers to the use of deep artificial networks to implement and learn the functions underlying the graphical model, and not to the topology of the graph. The most typical example of these Deep LVMs is the *Variational Auto-Encoder*.

3.1 The Variational Auto-Encoder

The Variational Auto-Encoder (VAE) [RMW14; KW14] is an instantiates the simplest LVM structure: an observed variable X and a latent variable Z, linked as a Bayesian Network $Z \to X$. The model thus factors as $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$. Deep learning is leveraged to implement $p_\theta(x|z)$ (Section 1.1.1): given a parametric family $\mathbb{P}_\omega(x)$ of distributions (e.g. Gaussian), a function $f_\theta : z \to \omega$ is used to map the latent variable z onto the parameters ω of the chosen family, as illustrated by Figure 3.1. It results that $p_\theta(x|z) = \mathbb{P}_{\omega=f_\theta(z)}(x)$. The function f_θ is implemented using an artificial neural network and trained by maximizing the ELBO. The main two ideas used in the VAE concern the construction of the inference model q which approximates $p_\theta(z|x)$: *Amortized Inference* and the *Reparameterization Trick*.

3.1.1 Amortized Inference

While Bayesian inference is in general a hard and computationally expensive problem, it is believed that human cognition performs it (at least approximately) routinely

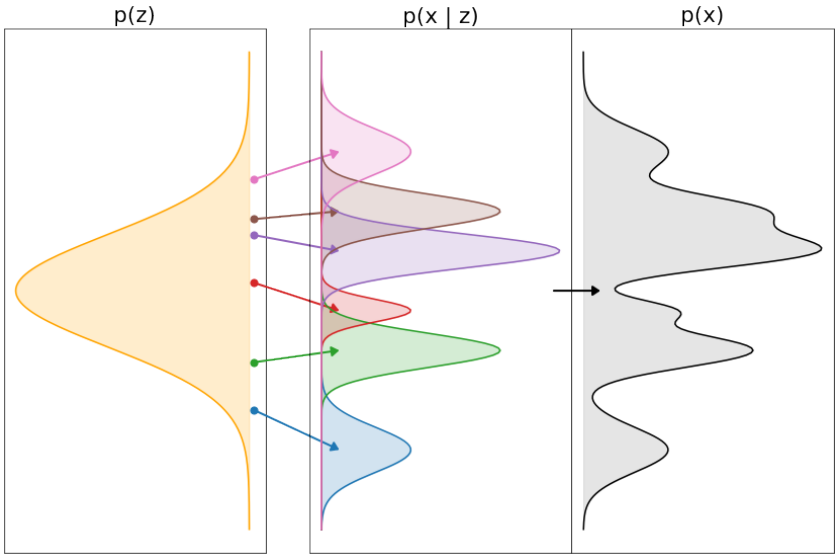


Figure 3.1: Illustration of the generative behavior of a Gaussian VAE: each latent value z (left) is mapped to a Gaussian distribution in the data space $p_\theta(x|z)$ (middle). The mixture of these individual Gaussian distributions makes the complete generative distribution $p_\theta(x)$ (right).

with much efficiency. Human beings generally face and solve many instances of very similar problems (such as recognizing an object in a visual scene). Under the assumption that similar problems have similar solutions, results of past inferences can generally be re-used to solve future ones more efficiently. This action of re-using past inference is referred to as *amortized inference*, as the cost is the initial inferences is amortized by its subsequent reuse. Some evidence has been presented that human beings do actually rely on such re-use, at least to some extent [GG14].

The general principle of amortized inference in a Bayesian Netowrk setting consists in analyzing the graphical model beforehand and inverting it, thus making it possible to perform quick approximate inference queries using the resulting inverted graph [STG13]. The more queries are answered using the inverted graph, the more the inverting cost is amortized.

In the VAE setting, the assumption translates to considering that $p_\theta(z|x)$ varies smoothly with x. In other words, letting x and x' denote two similar observed instances, the associated conditional distributions $p_\theta(z|x)$ and $p_\theta(z|x')$ should also be similar to each other. This prompts the idea of jointly learning all $q_x(z)$ approximations. Like for $p_\theta(x|z)$, the conditional distribution $p_\theta(z|x)$ is approximated by choosing a parametric family and training a neural network mapping x to the parameters of the considered family. By opposition to the generative model, this construct is traditionally named the *inference model*, and noted $q_\phi(z|x)$, with ϕ the trainable parameters of the associated neural network.

3.1.2 The Reparametrization Trick

The Reparameterization Trick is a method to compute the gradient of an expectation with respect to the parameters of the distribution: $\nabla_\phi \mathbb{E}_{z\sim q_\phi} g(z)$, as appears in the

ELBO. In order to train the inference model using stochastic gradient descent (as usual for neural networks), then this gradient needs to be computed. A well known method to do so is the so-called Log-Trick, which relies on the following identity:

$$\nabla_\phi \mathop{\mathbb{E}}_{z\sim q_\phi} g(z) = \mathop{\mathbb{E}}_{z\sim q_\phi} \Big[g(z) \nabla_\phi \log q_\phi(z) \Big] \tag{3.1}$$

This identity however leads to an estimation of the gradient with very high variance[1], making it a poor candidate to do Monte-Carlo estimation. The Reparametrization Trick [KW14] instead tries to express the distribution q_ϕ as some noise sampled from a fixed base distribution $\epsilon \sim \pi$, and then transformed by a function $z = h_\phi(\epsilon)$. This is similar to Normalizing Flows (Section 2.3.3), which were inspired from it. Accordingly, the expectation is expressed over the base distribution π, which does not depend on the parameters ϕ, allowing the gradient operator to commute with it:

$$\nabla_\phi \mathop{\mathbb{E}}_{z\sim q_\phi} g(z) = \nabla_\phi \mathop{\mathbb{E}}_{\epsilon\sim \pi} g(h_\phi(\epsilon)) = \mathop{\mathbb{E}}_{\epsilon\sim \pi} \nabla_\phi [g(h_\phi(\epsilon))] \tag{3.2}$$

This estimator has much better variance characteristics [Xu+19], and can in practice be efficiently implemented in deep learning software.

3.1.3 Link with Auto-Encoders

Combining the above, the VAE proceeds as illustrated on Figure 3.2: first, the datapoint x is processed by the inference network, predicting the inference distribution $q_\phi(z|x)$ from which a latent sample z is sampled. This sample is then processed by the generative network to compute the evidence $\log p_\theta(x|z)$. Both networks are jointly trained by stochastic gradient descent to maximize the ELBO, where the expectation over z is often approximated by a single sample of $q_\phi(z|x)$:

$$ELBO(\theta, \phi) = \mathop{\mathbb{E}}_{x\in\mathcal{D}} \left[\underbrace{\mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} \log p_\theta(x|z)}_{\text{Reconstruction loss}} - \underbrace{D_{KL}(q_\phi(z|x) \| p_\theta(z))}_{\text{Latent regularization}} \right] \tag{3.3}$$

The inference and generative networks can respectively be viewed as probabilistic encoder and decoder. Along this line, the ELBO decomposes into a reconstruction loss and a latent regularization term (Equation 3.3). The training process aims to trading-off the reconstruction loss and the compression of the latent information, as measured by the D_{KL} term, akin a regularized auto-encoder. This parallel explains the origin of the VAE name.

Gaussian VAE VAEs classically use Gaussians as base distributions, with $p_\theta(z)$ set to $\mathcal{N}(0; I)$, and $p_\theta(x, z)$ set to $\mathcal{N}(dec_\theta(x), \sigma^2)$, with dec_θ the output of the decoder network; σ is a hyperparameter of the model. Likewise, the inference model $q_\phi(z|x)$ is classically implemented as $\mathcal{N}(\mu_\phi(x), \sigma^2_\phi(x)$, with $\mu_\phi(x)$ and diagonal covariance

[1] While the source of this high variance is not theoretically established, to our best knowledge, it intuitively reflects the fact that this estimator does not use any differential information from g, only its values.

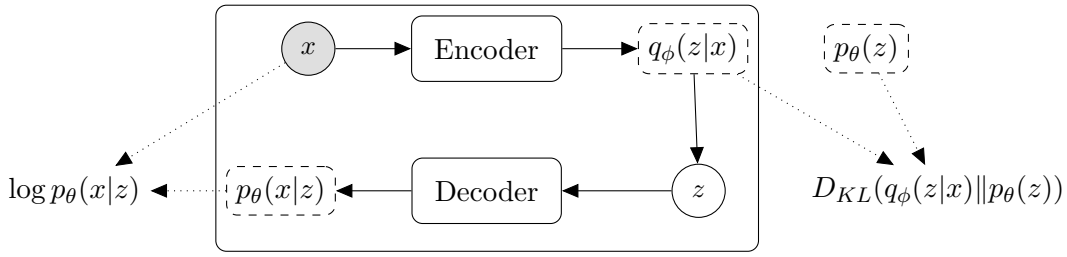


Figure 3.2: VAE architecture: the encoder (resp. decoder) module is a neural network taking x (resp z) as input and computing the parameters of distribution $q_\phi(z|x)$ (resp., $p_\theta(x|z)$). Dashed rectangles represent probability distributions and dotted arrows represent the computation of loss terms.

matrix $\sigma^2_\phi(x)$ the output of the encoder network. In this case, the reparameterization trick simply is:

$$z = \mu_\phi(x) + \epsilon \odot \sigma_\phi(x)$$

with $\epsilon \sim \mathcal{N}(0; I)$ and $\odot$ denoting the element-wise vector product. Likewise, the KL-divergence between $q_\phi(z|x)$ and $p_\theta(z)$ can be computed analytically.

The loss to be minimized is then the opposite of the ELBO, yielding the following loss (where j runs across the dimensions of the latent variable Z):

$$\mathcal{L}(\theta, \phi) = \mathop{\mathbb{E}}_{\substack{x\sim\mathcal{D} \\ \epsilon\sim\mathcal{N}(0,I)}} \left[\underbrace{\frac{1}{2\sigma^2}\|x - dec_\theta(z)\|^2}_{\text{Reconstruction loss}} + \underbrace{\frac{1}{2}\sum_{j=1}^{K}\left(\mu^2_{\phi,j}(x) + \sigma^2_{\phi,j}(x) - \log\sigma^2_{\phi,j}(x) - 1\right)}_{\text{Latent regularization}} \right] \tag{3.4}$$

3.2 Advanced latent models

While Gaussian VAEs are easy to both implement and train, and yield some impressive results [KW14], the Gaussiam structure might fail to represent complex distributions, e.g., images: the too simple approximation of the conditional latent can make the ELBO too loose a bound, in which case the model is susceptible to generate non-realistic samples [AB17].

This limitation raises the question of developing more elaborate representations of the latent distribution. The main two directions explored in the literature to this aim include constructing more powerful representations for $q_\phi(z|x)$, or learning also $p_\theta(z)$ to accommodate for the limited expressiveness of q_ϕ.

3.2.1 Powerful encoders and complex latent spaces

This section provides a (non exhaustive) overview of some approaches aimed to designing a more powerful inference model. Note that these approaches are not necessarily incompatible, and can be combined together.

Conditional Normalizing Flows. A powerful framework for inference models is that of the Normalizing Flows (Section 2.3.3): a flexible and powerful approximation $q_\phi(z|x)$ is obtained via $z = f_\phi(\epsilon, x)$, where f_ϕ is only required to be invertible with regard to ϵ. Note that this usage was the initial motivation for the introduction of Normalizing Flows [RM15]. This formulation drastically increases the expressive power of the inference model, enabling very sharp boundaries in the latent space, as shown by [Kin+17], at the expense of some (significant) increase in the number of layers of the encoder network.

Importance Weighted AE proceeds by replacing the $\frac{p_\theta(x,z)}{q_\phi(z|x)}$ in the ELBO by an empirical average over k z_i samples [BGS16]:

$$\mathcal{L}_k = \sum_{x \in \mathcal{D}} \mathop{\mathbb{E}}_{(z_1,\dots,z_k)\sim q_\phi(\cdot|x)} \log\left(\frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(x,z_i)}{q_\phi(z_i|x)}\right) \tag{3.5}$$

This new formulation, still a lower-bound of the model evidence $\log p_\theta(\mathcal{D})$, becomes tighter and tighter as k increases, as the sum in the logarithm better estimates $p_\theta(x)$. Indeed, when k goes to infinity, it comes:

$$\lim_{k\to\infty}\frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(x,z_i)}{q_\phi(z_i|x)} = \mathop{\mathbb{E}}_{z\sim q_\phi}\frac{p_\theta(x,z)}{q_\phi(z|x)} = \int_z q_\phi(z|x)\frac{p_\theta(x,z)}{q_\phi(z|x)}\,dz = \int_z p_\theta(x,z)\,dz = p_\theta(x)$$

This tighter estimate allows to overcome a poor quality inference model, and increase the quality of the learned model.

While the Importance Weighed AE was originally formulated as an alternative lower bound to the ELBO, it has since then been re-interpreted as equivalent to using the classic ELBO with an augmented inference model [CMD17]: the k samples from $q_\phi(z|x)$ are aggregated into a single sample from an augmented distribution $q_{EW}(z|x)$, which is closer to the model posterior $p_\theta(z|x)$.

MCMC refinement of samples. Noting that neural networks are by construction differentiable, one can compute $\nabla_z p_\theta(x|z)$, opening the door to the use of efficient MCMC methods to directly sample from $p_\theta(z|x)$. Such Monte-Carlo methods allow an extremely tight evaluation of the ELBO, e.g. using Langevin dynamics [Han+17], or using Hamiltonian Monte Carlo to improve the initial proposal of an inference network [CDS18].

Another direction is that of Semi-Amortized VAEs [Kim+18], using stochastic variational inference [Hof+13] to improve the variational parameters of $q_\phi(z|x)$ predicted by the neural network, rather than working directly on the samples.

The use of Monte-Carlo methods however comes at a significant computing cost, requiring back-propagation through the neural networks to compute the gradient for every one of the (many) samples.

Non-euclidean latent spaces. In order to match the complex topology of some datasets, distributions constructed on non-Euclidean spaces have also been proposed recently. The Poincaré VAEs [Mat+19a] build an hyperbolic latent space with

negative curvature, better suited to tree-structured data than a flat Gaussian latent[2]. [Fal+19] introduces a general method for transposing distributions defined on an Euclidean space into an arbitrary Lie Group, including the reparametrization trick.

3.2.2 Learned latent distributions

While above methods might change the structure of the latent space, governing the latent distribution $p_\theta(z)$, this latent distribution remains fixed during the training. Often called a "prior" in the literature, this distribution is only a prior in the Bayesian sense with regards to the inference of z given x, not with regards to the learning of the model parameters θ. It can be learned just as every other part of the model, and this section presents some of the methods developed to do so.

A general incentive for learning the latent distribution is to account for some well-separated modes of the true distribution. If the latent distribution cannot capture several modes (as is the case for Gaussian distributions), this makes it difficult to learn the whole model:
One possibility is that the generative network $p_\theta(x|z)$ learns a quickly varying function, able to separate in the data space values that are close to each other in the latent. This ability requires a significant expressive power from the neural network[3]. Another possibility is that large parts of the latent space with non-zero mass according to $p_\theta(z)$ be avoided by $q_\phi(z|x)$, achieving the separation of the data (Figure 3.3). This option is the one empirically observed when training Gaussian VAEs from multimodal data. When latent z samples fall in these avoided regions, they yield samples $x \sim p_\theta(x|z)$ that are generally unrealistic, illustrating that the ELBO remains loose in this context.

While the methods described below can in principle be combined with the ones from in the previous section, their combination raises difficulties: learning both $q_\phi(z|x)$ and $p_\theta(z)$ as complex distributions often results in training instabilities.

Mixture-based latent distributions. A simpler way to handle multi-modal data is to model $p_\theta(z)$ as a mixture model. As fitting a mixture model by gradient descent is difficult, more elaborate methods have been considered, e.g., modeling the mixture as part of the graphical model [Dil+17; Don+19] (see next chapter for the use of Deep LVM with several nodes) or adapting the ELBO to directly integrate the mixture latent [Guo+20].

Linking the latent to the inference model. As illustrated by the VampPrior [TW18], this approach implicitly defines the latent distribution $p(z)$ from the inference model $q_\phi(z|x)$, as a mixture of inference predictions from few artificial inputs

[2] In negative-curvature spaces distances tend to grow quickly in a non-intuitive way. While the perimeter of a circle in an euclidean space grows linearly with its radius, in a negative-curvature space it will grow quicker. This means that the amount of space at a given maximal distance from the origin is higher in negative-curved spaces, allowing to arrange more datapoints equidistant from each other than would be in an euclidean one. This property is successfully exploited to represent geometrically hierarchically-structured data.

[3] In order to separate two very close latent values, the neural network (the represented function) must allow for abrupt variations in their vicinity. Depending on the optimization algorithm and neural architectures, this might be hardly doable or unstable.

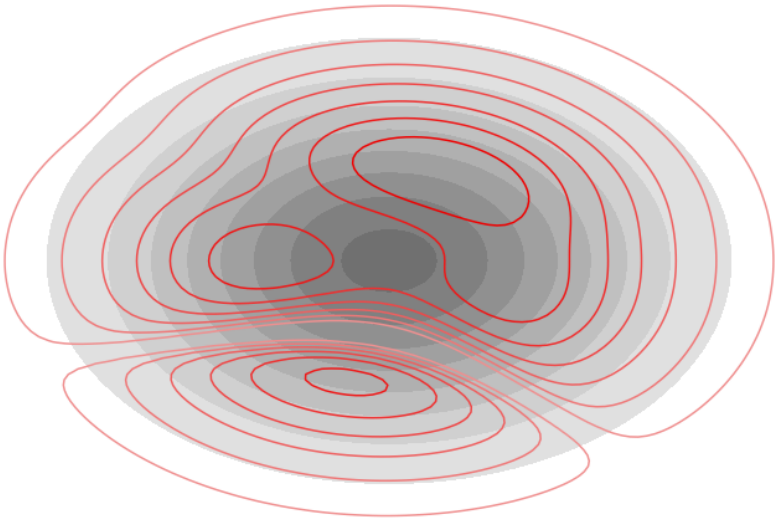


Figure 3.3: Illustration of a possible mismatch in the inference density compared to the latent distribution. The grey background represents the latent density $p_\theta(z)$, while the red lines represent the inference density $q_\phi(z|x)$ averaged over the dataset. The inference density is here split in two regions (top and bottom) separated by a low-probability band, which represent two modes in the dataset that the model couldn't join continuously.

$(\hat{x}_1, \dots, \hat{x}_K)$: $p(z) = \frac{1}{K}\sum_i q_\phi(z|\hat{x}_i)$. Learning the latent distribution thus amounts to learning the artificial inputs, which is done by back-propagation through the inference network.

Implicit latents with an energy model. A very versatile approach is to learn the latent distribution through an energy model: learn some function $E : \mathcal{Z} \to \mathbb{R}$, and implicitly define the distribution as

$$p(z) \propto \exp(-E(z))$$

While this formulation can in principle accommodate most latent structures, it raises two difficulties. Firstly, the latent distribution can hardly be directly sampled, requiring the use of Monte-Carlo methods. Secondly, designing a proper optimization procedure to learn the energy function E is highly non-trivial. Some approaches learn it by estimating the gradient online using Equation 1.22 [Pan+20]; other approaches fit *a posteriori* the energy function on a already trained model to fine-tune it [XYA20].

3.3 Discrete latent variables

All above methods, relying on the reparametrization trick to compute gradients, are limited to continuous latent spaces. However, some datasets are better handled by using discrete structures, raising the issue of whether Deep LVMs can be used with discrete latent variables.

When considering a small latent domain, the expectation over $q_\phi(z|x)$ can be easily and exactly computed from the vector of probabilities of the discrete distribution.

The difficulty arises when considering a large latent domain, e.g. $\mathcal{Z} = \{0, 1\}^K$. In this case, the cost of exactly computing the expectation is prohibitive, being exponential in the dimension K of the latent space.[4]

Two main approaches have stood out to tackle the problem of large discrete latent space: the Gumbel-Softmax, and the so-called Discrete VAE.

Gumbel Softmax This approach relies on the standard Gumbel distribution, whose cumulative distribution function is $G(x) = e^{-e^{-x}}$. It can be used for approximating a one-hot encoding of a categorical distribution[5] [JGP17].

Let us consider a categorical distribution defined by the probability vector $(\pi_1, \pi_2, \dots, \pi_K)$, and let $(g_1, g_2, \dots, g_k)$ be K independent samples from the standard Gumbel distribution. Let vector z be defined by softmax from the vector of coordinates $(g_i + \log \pi_i)/\tau$, with $\tau > 0$ a hyperparameter of the model:

$$z_i = \frac{\exp\left((g_i + \log \pi_i)/\tau\right)}{\sum_{j=1}^{K} \exp\left((g_j + \log \pi_j)/\tau\right)} \tag{3.6}$$

By definition of the softmax, all coordinates of z belong to $(0, 1)$, and they sum to 1. Formally, z converges to a one-hot vector as $\tau \to 0$. Further, the distribution of such zs matches a one-hot encoding of samples from the categorical distribution defined by the probabilities π_i.

This property thus supports a continuous approximation of a one-hot-encoded categorical distribution; this continuous approximation can be used with the reparameterization trick to train a VAE [Lor+19].

Discrete VAE The Discrete VAE [Rol17] and its refinements [Vah+18; VAM18] take a different approach, where the latent space involves a set of K binary variables ($z \in \{0, 1\}^K$). Each latent coordinate z_i is paired with a continuous one $\zeta_i \in [0; 1]$, where z_i depends on ζ_i after some appropriate fixed distribution $p(z_i|\zeta_i)$. In the final generative step, x is generated from the continuous vector ζ, as illustrated on Figure 3.4.

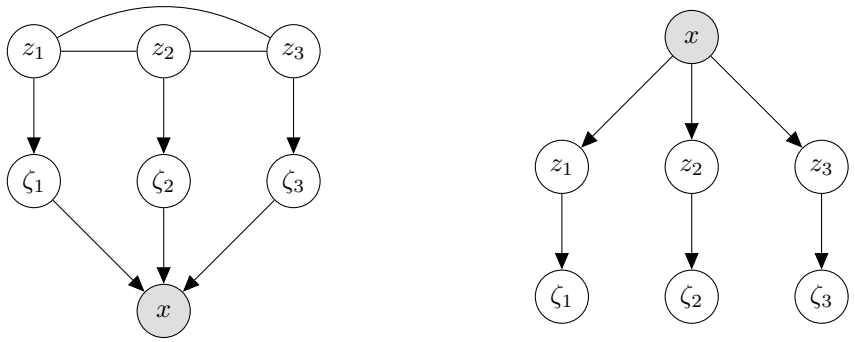
The inference model involves the inference distribution $q_\phi(z_i = 1|x)$. Thanks to the fixed dependency among z_i and ζ_i, $q_\phi(\zeta_i|x)$ can be analytically expressed as a function of $q_\phi(z_i = 1|x)$[6]. The chained dependencies thus allow the gradient to flow and learn q_ϕ through the discrete variable z.

This discrete structure is combined with a latent distribution $p_\theta(z)$ learned as a Restricted Boltzmann Machine (Section 1.1.2), in order to capture complex correlations. The encoder and decoder are trained as a regular VAE by optimizing the ELBO and the latent RBM is trained at the same time by maximum likelihood using samples from the encoder $q_\phi(z|x)$ as a dataset.

[4] Unfortunately, the log-trick yields poor estimates of the gradient in this context.

[5] Letting k be sampled from a categorical distribution with K values, its one-hot encoding is the K-dimensional vector with all coordinates set to 0 except for the k-th, set to 1.

[6] This properties relies on the specific choice of $p(\zeta_i|z_i)$ made in DVAE, and the re-use of that same distribution in q_ϕ.



(a) Generative model **(b)** Inference model

Figure 3.4: Discrete VAE: Graphical representation of: (a) the generative model; and (b) the inference model.

3.4 Impact of the Inference Model

In general[7], q_ϕ lies in some selected class of distributions $\mathcal{Q}$. This restriction has an impact on the training dynamics of the model, as it controls the quality of approximation of $p_\theta(z|x)$, permitted by $\mathcal{Q}$. This impact can be understood as a kind of *posterior regularization* [Gan+10], as analyzed by [Shu+18].

Considering the training ELBO for the whole dataset:

$$ELBO(\theta,\phi) = \sum_{x\in\mathcal{D}} \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} \log \frac{p_\theta(x,z)}{q_\phi(z|x)} = \log p_\theta(\mathcal{D}) - \sum_{x\in\mathcal{D}} D_{KL}(q_\phi(z|x)\|p_\theta(z|x)) \tag{3.7}$$

Let us introduce for any distribution $r(z)$ its "bias" relative to the class $\mathcal{Q}$ as:

$$D_{\mathcal{Q}}(r) = \min_{q\in\mathcal{Q}} D_{KL}(q(z)\|r(z)) \tag{3.8}$$

By training the inference model until ELBO-optimality (reaching the optimal parameters $\phi^\star(\theta)$) then for all x, the term $D_{KL}(q_\phi(z|x)\|p_\theta(z|x))$ is minimized and thus equal to $D_{\mathcal{Q}}(p_\theta(\cdot|x))$. The ELBO can thus be reformulated as:

$$ELBO(\theta,\phi^\star(\theta)) = \log p_\theta(\mathcal{D}) - \sum_{x\in\mathcal{D}} D_{\mathcal{Q}}(p_\theta(\cdot|x)) \tag{3.9}$$

Along this line, using a restricted class of inference model is thus similar to Posterior Regularization: the model is trained to maximize the likelihood of the dataset $\log p_\theta(\mathcal{D})$, augmented with a penalization reflecting the bias of its latent conditional relative to $\mathcal{Q}$. The more expressive $\mathcal{Q}$, the weaker the regularization is, to the point of completely disappearing for very expressive inference models such as MCMC. Further restricting the class $\mathcal{Q}$ has been additionally shown as a way to improve generalization in VAEs [Shu+18].

This regularization, constraining the abstract latent variable z, can serve to mitigate the pitfalls linked to non-identifiable latent variables (Section 2.2.2). Imposing a restrictive class $\mathcal{Q}$ can thus be used to enforce desirable properties in the learned model. For example if $\mathcal{Q}$ only contains distributions with a single mode, then

[7] Unless powerful inference models, e.g. Normalizing Flows or MCMC-based, are considered.

the model is drawn toward learning a latent representation such that $p_\theta(z|x)$ has a single mode as well. Several approaches to exploit this regularization mechanism are discussed by [Shu+18].

Likewise, disentanglement of the latent factors has been observed in factorized Gaussian VAEs [Mat+19b]. It has been suggested that inductive biases are in fact necessary to encourage such desirable properties in the latent representations [Loc+19]. Along this line, the choice of a restrictive class $\mathcal{Q}$ is viewed as an alternative (compared to modifying the ELBO criterion) to enforce the desired biases [Hig+17; Che+18a].

3.5 Summary

This chapter introduces the *Variational Auto-Encoder (VAE)*, transposing latent variable models in the deep learning setting. In the basic case of a Bayesian Network with a single observed and a single latent variable (both of which can be multi-dimensional), VAE mainly relies on: i) *amortized inference*, consisting in learning the inference model q as a function of the observed variable x (as opposed to learning a different q_x approximation for each datapoint); ii) the *reparameterization trick*, aimed to express expectations over $q(z|x)$ as expectations over a fixed base distribution, whose samples are transformed by a differentiable operation, significantly lowering the variance of the gradient estimation.

The VAE can thus be viewed as a regularized Auto-Encoder with a distribution / sampling mechanism at its core. Numerous efforts have been made to address the limitations related to the original use of Gaussian distribution, and design appropriate inference model $q_\phi(z|x)$ and latent distribution $p_\theta(z)$, depending on the data domain.

An aspect of the chapter is to show how, within the *Posterior Regularization* framework, the use of a limited inference model q_ϕ can shape the distribution and favor desirable properties such as latent disentanglement.

Chapter 4 Hierarchical Deep LVMs

Contents

This chapter extends the Variational Auto-Encoder framework and the ELBO learning criterion, to the general case of *Hierarchical Deep LVMs*, where several latent and/or observed variables are structured into a Deep LVM.

4.1 The ELBO with hierarchical latent variables

In the previous sections, the ELBO involves two (multi-dimensional) variables x and z. However this framework can be generalized and involve an arbitrary number of observed and latent variables. Let p denote a model with K observed variables $x_1, \ldots, x_K$ and L hidden variables $z_1, \ldots, z_L$, then for any distribution q over the hidden variables:

$$\log p(x_1, \ldots, x_K) \geq \mathop{\mathbb{E}}_{z1,\ldots,z_L \sim q} \log \frac{p(x1, \ldots, x_K, z_1, \ldots, z_L)}{q(z1, \ldots, z_L)} \tag{4.1}$$

Along this more general formulation, the VAE is extended to handle complex relations among latent and observed variables, referred to *Hierarchical Deep LVMs*, where the hierarchy relates to the structure involving the (usually latent) variables.

Such a generative model is specified as a generic graphical model, usually a Bayesian Network, defining some factorization of the distribution p_θ. The main specificity is that conditional distributions are implemented and trained as neural networks, akin to the VAE decoder. After the amortized inference principle (Section 3.1.1), an inference model is sought as $q_\phi(z_1, \ldots, z_L | x_1, \ldots, x_L)$.

The inference model is generally also factorized a a Bayesian Network, for considering a joint probability over all latent variables can be very impractical. The main design issue is that of the most appropriate factorization of the inference

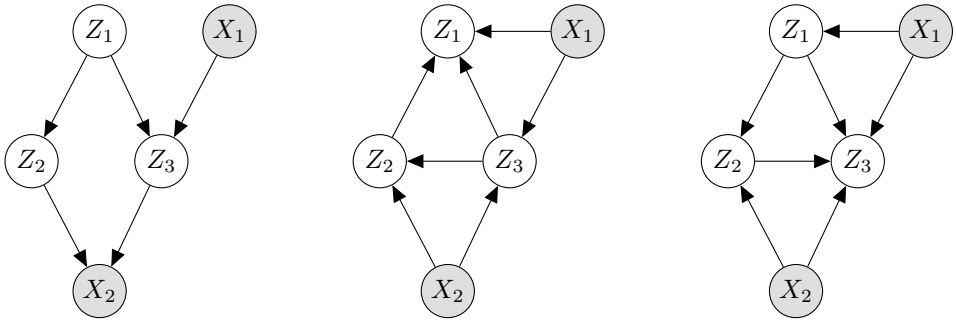
model. While the factorization of the generative model p_θ is guided by domain knowledge and other epistemic considerations, the factorization of the inference model q_ϕ primarily aims at efficiently approximating the conditional distribution $p_\theta(z_1, \dots, z_L | x_1, \dots, x_K)$.

A principled approach for designing the inference model is the *Natural Minimal I-map generator* (NaMI) algorithm [Web+18]. This principle operates on the graph defining p_θ in order to produce a new graph usable for q_ϕ that is *minimally faithful* with regard to p_θ.

The graphical model representing q_ϕ is said to be *faithful* to the one underlying p_θ if all independence relations of q_ϕ are also present in p_θ: in other words if q_ϕ does not introduce *new* independence relations between variables relative to p_θ. It is however allowed to have fewer such relations. For example, having q_ϕ be a fully connected graph is a trivial way to make it faithful.

In addition, NaMI seeks to produce a graph q_ϕ that is *minimally faithful*, that is that has the smallest possible number of edge while still being faithful to p_θ. A graph is said to be minimally faithful if it is faithful, but removing any of its edge would make it unfaithful. Note that this is a local property, not a global one. Hence there can exist many graphs that are each minimally faithful with regard to p_θ. NaMI thus seeks to create one such graph.

The general process of NaMI can be summarized as follows. Starting from the original graphical model (Figure 4.1(a)), first the arrows on all edges are removed to produce an undirected graph. Then this graph is *moralized*: an edge is added between each pair of node which share a child in the original graph. The resulting undirected graph is then the skeleton on which the graph for the inference model q_ϕ is built: the last stage consists in directing the edges to produce a new Bayesian Network. This is done by iterating on all *latent* nodes of the graph and for each directing all not-yet-directed of its edge towards it. For example, in the graph presented on Figure 4.1(b), the latent nodes were visited in the order Z_1, Z_2, Z_3, while on figure Figure 4.1(c) the order was Z_3, Z_2, Z_1.



(a) Generative graph **(b)** Forward-NaMI **(c)** Reverse-NaMI

Figure 4.1: Example of a generative model graph (a), the associated inference model generated by *forward-NaMI* (b), and the one generated by *reverse-NaMI* (c).

Two main properties of this procedure can be noted. First of all, in the produced graph the observed variables don't have any parent: this reflects the fact that this graph represents a conditional distribution of the latent variables given the observed one. Secondly, the resulting graph depends on the order in which the latent variables are processed: each ordering potentially produces a different Bayesian Network.

NaMI identifies two special ordering, referred to as *forward*-NaMI and *reverse*-NaMI, both relying on a topological ordering of the nodes in the original graph[1]. *Forward*-NaMI processes the latent nodes in the same order as the topological ordering, while *reverse*-NaMI processes them in reverse order. Note that as a result, *Forward*-NaMI tends to overall reverse the structure of the original graph and *reverse*-NaMI tends to follow it, as illustrated in Figure 4.1.

Note that one might prefer considering more sparse graphs than the NaMI ones: by using a graph with less edges, one sets more constraints on the learned model, as discussed in Section 3.4

4.2 Optimization of hierarchical structures

While the vanilla VAE (Chapter 3) can usually be trained in an easy and stable way using stochastic gradient descent, the training of hierarchical models raises stability issues, some typical from Deep Learning in general, and some specific to the LVM structures.

4.2.1 Gradient flow

A main stability issue faced by Deep Learning is related to the so-called *vanishing gradients* phenomenon [Hoc+01]. While it is required that the gradients can flow efficiently through the neural network along back-propagation, the vanishing gradients cause the deepest layers of the network to not receive enough gradient information to be efficiently trained. The vanishing gradients are ultimately blamed on the classical activation functions (e.g. sigmoid), with saturating regions where the gradient is vanishingly small (the derivative of the activation function is close to 0). Stacking many layers and activation functions thus makes gradients quickly converging to 0 as they back-propagate. More recent activation functions, like the ReLU [GBB11] somewhat mitigate this issue; however, as quite some RELU neurons are saturated in each layer, a non-negligible fraction of the gradient information is lost at each step.

Deep Hierarchical LVM are also susceptible to this problem, as introducing a hierarchy of latent variables implies that the gradient will need to flow though more neural networks (one per edge in the Bayesian Network defining the model). This can lead the latent variables which are farthest to the observed ones in the graph to not train well, if at all [Sø +16], and as such the resulting model cannot exploit its capacity to the fullest.

A common mitigation for this problem in the Deep Learning literature is the use of residual networks[He+16]. These constructs rework the structure of neural network in such a way that several paths lead to each neurons, and can ensure that at least one of these paths only consists in affine transformations, ensuring meaningful gradient always reaches all parameters in the neural network. Residual networks can in general be an answer to this issue for Deep Hierarchical LVM as well, though some care needs to be taken when incorporating them due to the stochastic nature of the training, which can cause stability issues.

[1] A topological ordering is the definition of an ordering relation of the graph compatible with its topological structure, meaning that if node B is a descendant of node A, then the ordering must have $A \preccurlyeq B$. A topological ordering for a given DAG is not necessarily unique.

4.2.2 Stability problems

The ELBO training objective is computed based on two expectations: one over the examples in the dataset, and another over the latent variables, sampled from the inference model q_ϕ. The first expectation is approximated using mini-batches, leading to training via stochastic gradient descent [Lec+98b; Hof+13]. The second expectation is generally approximated by Monte-Carlo: q_ϕ is sampled a few times and the ELBO is computed on the basis of this average. Though often the expectation is approximated by a single sample with the assumption that the noise will average out across the many training iterations.

This however significantly increases the noise of the gradient estimation during the training, making in general VAE almost impossible to train with plain stochastic gradient descent without diverging. Adaptive optimizers like Adam [KB17] are generally used[2] for their momentum properties, which reduce the noise in parameter updates.

The use of Hierarchical Deep LVMs pose the additional difficulties that, for internal latent variables, both q_ϕ and p_θ are generated by the output of a neural network each. Due to the stochasticity of the optimization process, it can happen that at some point a minibatch is evaluated for which predicted distributions are quite far from each other at some variable z_i, causing the probability ratio $\frac{p_\theta(z_i|\dots)}{q_\phi(z_i|\dots)}$ in the ELBO to be extremely small. This in turns causes a large spike in the ELBO value for that minibatch, causing very large gradients to correct for it, excessively disrupting the internal state of the optimizer and causing the training process to diverge.

This is mostly a problem of training dynamics: due to the large number of parameters in artificial neural networks, even a small update to each (as enforced by Adam) can still have a dramatic effect on the output of the network at the next iteration. The more hierarchical variables in the model, the more likely that at least one of them, when sampled, produces a value in an unlikely region of the space, creating a cascade of unusual values entering the subsequent networks, entailing a spike of the loss and generating extreme gradients.

This instability could be mitigated by considering a sufficient number of samples drawn after q_ϕ and taking their average to better approximate the expectation; but this approach is hardly affordable.

4.2.3 Alternative training formulations

The stability issue can also be handled by adjusting the training procedure and the ELBO optimization. Two approaches investigated in the literature include considering an inference model structure tightly connected to that of the generative model, and sequentially learning the latent variables. Both approaches have mainly considered unary tree-structures, as illustrated in Figure 4.2, factoring as:

$$p_\theta(x, z1, \dots z_K) = p_\theta(x|z_K)p_\theta(z_K|z_{K-1}) \dots p_\theta(z_2|z_1) \tag{4.2}$$

Here, the hierarchy of latent variables mostly aims at a more powerful representation: with numerous latent variables stacked on top of each other, even if each

[2] It is classic to use Adam with lower momentum parameters than the standard, such as $(\beta_1, \beta_2) = (0.5, 0.9)$ rather than $(0.9, 0.999)$, the later being often unstable when used with VAEs.

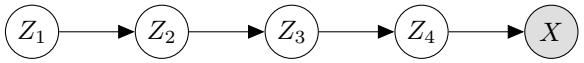
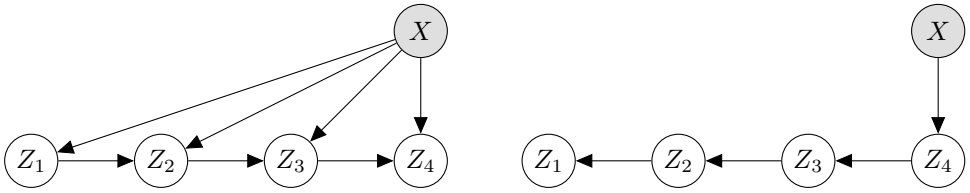


Figure 4.2: Hierarchical graphical model whose 4 latent variables are structured as a unary tree.



(a) Inference model of LadderVAE. **(b)** Output of *forward*-NaMI.

Figure 4.3: Comparison of the inference model used by Ladder-VAE and the on that NaMI would produce for this generative model.

variable has simple neural networks encoding them, complex marginal distributions can be reached for the last one z_K.

Coupling the inference and generative models Introduced by [Sø +16], the LadderVAE builds an inference model with same topological ordering as the generative model (thus similar to *reverse-NaMI*). It relies on a Gaussian parametrization of the latent variables: each layer z_i of the generative model is modeled by a neural network with output $\mu_{p,i}(z_{i-1}), \sigma^2_{p,i}(z_{i-1})$. Finally, $p_\theta(z_i|z_{i-1}) = \mathcal{N}(\mu_{p,i}(z_{i-1}), \sigma^2_{p,i}(z_{i-1}))$.

The specificity of the LadderVAE lies in the definition of its inference model. first, a deterministic neural network takes the data x as input and produces parameters $\hat{\mu}_{q,i}(x), \hat{\sigma}^2_{q,i}(x)$ for all latent variables as once, and then the actual q distribution is defined by combining these parameters with the associated parameters from p to build $q_\phi(z_i|x, z_{i-1}) = \mathcal{N}(\mu_{q,i}, \sigma^2_{q,i})$:

$$\sigma^2_{q,i} = \frac{1}{\sigma^{-2}_{p,i} + \hat{\sigma}^{-2}_{q,i}} \quad \text{and} \quad \mu_{i,q} = \frac{\mu_{p,i}\sigma^{-2}_{p,i} + \hat{\mu}_{q,i}\hat{\sigma}^{-2}_{q,i}}{\sigma^{-2}_{p,i} + \hat{\sigma}^{-2}_{q,i}} \tag{4.3}$$

The above coupling of q_ϕ and p_θ defines the inference model as an iterative refinement of the prediction of the generative model, using information extracted from the target value x, with the variance predictions as mixture weights.
This formulation avoids a major source of instability as it ensures that $\sigma_{q,i} \leq \sigma_{p,i}$. Quite the contrary, when $\sigma_{q,i} >> \sigma_{p,i}$ in a regular VAE, thethis incurs a high Kullback-Leibler divergence, strongly penalizing the model and generating large gradients. LadderVAE is by construction immune to this risk of instability.

Note that in the LadderVAE all latent predictions from the inference model directly depend on x (as opposed to e.g. NaMI where each variable depends on its direct neighbor variables, both cases are compared on Figure 4.3), enabling the information to flow through the model more efficiently.

Sequential training of latent variables The Two Stage VAE [DW19] addresses the following issue. When training a Gaussian non-hierarchical VAE, the marginal latent distribution of the inference model $q_\phi(z) = \mathbb{E}_{x\in\mathcal{D}}\, q_\phi(z|x)$, also referred to as

aggregated posterior in the literature, does not in general correctly match the generative latent distribution $p_\theta(z)$. This mismatch causes the generation of unrealistic samples: when z is sampled from p_θ in an improbable region after q_ϕ, the generative neural network has not been trained in such regions and it fails to generate a realistic sample.

The authors argue that this failure happens when the data is concentrated in a low-dimensional manifold of the data space [DW19]. This property, known as the *Manifold Hypothesis*, is discussed further in Section 6.1. Accordingly, the VAE needs to find said manifold, an ill-defined problem (except in the large sample limit). In this situation, the VAE can optimize its ELBO well without necessarily converging to an aggregated posterior that accurately matches the latent distribution $p_\theta(z)$.

Even though, the aggregated posterior $q_\phi(z)$ still covers most of the latent space $\mathcal{Z}$, not being restricted to a low-dimensional manifold. The Two Stage VAE precisely exploits this $q_\phi(z)$ and trains a second VAE, using another latent variable u to approximate $q_\phi(z)$. The eventual model is a Hierarchical LVM $p_\theta(x, z, u) = p_\theta(x|z)p_\theta(z|u)p(u)$. One can think of the first stage as "smoothing" the dataset in view of the second stage.

4.2.4 Key design considerations

Hierarchical models can still be trained in a stable way using gradient descent on the standard ELBO, provided that some care is taken of the neural network architecture, their initialization, and the training dynamics.

Neural Network architecture To allow the gradient to flow across the whole architecture and efficiently train the model, the use of residual structures is necessary. Specifically, for every neural network involved in either the inference or the generative models, there must exists one linear path from the input to the output neurons (involving no non-linear activation function). This direct path help ensuring the flow of the gradient information up to the deepest parts of the model.

The information flow towards the deepest part of the model can also be improved by augmenting the NaMI-produced graph for the inference model, adding additional edges from the observed variables to the latent ones, much like the LadderVAE does. The increased cost of these additional edges can be mitigated by implementing partial weight-sharing between the neural networks working on these inputs.

Model initialization The initialization of the various neural networks interacting during the training of a Hierarchical Deep LVM can have a dramatic impact on its training stability. A random initialization might result in generating poorly aligned samples from p_θ and q_ϕ, yielding a poor initial ELBO and (very) large gradients, as discussed in Section 4.2.2.

An option is to initialize each neural network in such a way that it yields a constant output, not depending on its inputs. This is achieved by setting to 0 the weight matrix of the last layer, with a constant bias adjusted to yield a sensible initial value depending on the considered distribution. For example, if the network is to predict a Gaussian distribution, it could be initialized to always rpedict a mean of 0 and a variance of 1. This way the rest of the network will still compute random

features of the input, allowing the gradient computation to pick up correlations and initiate the training. This targeted initialization of the last layer is part of the Fixup Initialization scheme [ZDM18].

The same initialization strategy also applies on the inference model, albeit with a different rationale. Having q_ϕ predict random distributions right from the beginning imposes an initial organization of the latent space of abstract variables[3]. If this organization is suboptimal (which is likely the case) the network can nevertheless be stuck with it, as it can form a local minimum of the training objective. Initializing q_ϕ to have constant predictions equal to that of p_θ avoids this, as no organization is imposed. The natural stochasiticity of the gradient estimation via Monte-Carlo is enough to break the symmetry of this initial configuration and allow training.

Finally, when using complex residual networks, one must preserve the initial variance (similarly to mainstream neural networks [Lec+98b]). Hierarchical Deep LVM are particularly sensitive to an initialization that increases the variance, as this effect can cascade through the latent variables hierarchy. The use of standard initializations within residual networks however tends to increase the output variance, due to the multiple paths. An empirically efficient alternative is the Fixup Initialization [ZDM18], which is designed to compensate this variance increase[4].

Noise reduction The noise in the estimation of the gradient through the reparameterization trick can be further reduced by using a Path Derivative estimator [RWD17]. Using the reparameterization trick, a latent variable z sampled from q_ϕ is written as a function of some standard noise ϵ, the parameters of the inference network ϕ, and the input x: $z(\epsilon, \phi, x)$. Then, the gradient estimate produced by the reparameterization trick decomposes as:

$$\nabla_\phi \mathop{\mathbb{E}}_{z\sim q_\phi} \log \frac{p_\theta(x,z)}{q_\phi(z|x)} = \mathop{\mathbb{E}}_{z\sim q_\phi} \left[\underbrace{\nabla_z \log \left(\frac{p_\theta(x,z)}{q_\phi(z|x)} \right) \cdot \nabla_\phi z}_{\text{path derivative}} - \nabla_\phi \log q_\phi(z|x) \right] \tag{4.4}$$

The last term of this equation, $\nabla_\phi \log q_\phi(z|x)$ has a null expectation:

$$\mathop{\mathbb{E}}_{z\sim q_\phi} \nabla_\phi \log q_\phi(z|x) = \int_z q_\phi(z|x) \frac{\nabla_\phi q_\phi(z|x)}{q_\phi(z|x)} dz = \int_z \nabla_\phi q_\phi(z|x) dz = \nabla_\phi \int_z q_\phi(z|x) dz = 0 \tag{4.5}$$

However, this term introduces variance in the gradient estimation, which tends to drown the useful signal during the latter stages of the training. It can however be removed from the gradient estimation using a stop-gradient instruction in the computation graph of the model (which is supported by most deep learning frameworks). A detailed discussing about when this estimator can be preferable to the simple reparameterization trick, and how to do it, is presented in [RWD17].

[3] This argument also applies to $p\theta$ for latent variable which have parents: if $p_\theta(z_i|\pi(Z_i))$ is randomly initialized this gives a very opinionated shape to the associated latent space.

[4] The variance of the random initialization is reduced by taking into account the number of different paths reaching a certain point of the network. This ensures that at initialization, the network as a whole preserves the input variance.

Training dynamics In some cases, the stability can be improved by using two different optimizers for the inference model q_ϕ and the generative model p_θ. In particular, choosing an optimizer with more inertia and a smaller stepsize for p_θ permits q_ϕ to converge quicker; this ensures that a tighter ELBO is used to compute the training gradients for p_θ. A slow-moving p_θ also prevents it from getting suddenly too far from the prediction of q_ϕ and generating huge gradients, as discussed in Section 4.2.2.

4.3 Graph Structure Learning

A question is whether same methods can be applied to learn the structure of Bayesian Networks (Section 1.3.3) and that of Hierarchical Deep LVM. This question however raises two difficulties: besides the general issue of learning the structure of an LVM, comes the additional complexity of structure choice in deep learning.

Two works related to learning Hierarchical Deep LVMs will be discussed (the litterature on the topic being scarce and recent): Graph-VAE [He+18] and LT-VAE[Li+19]. Both approaches center on learning an internal structure of Gaussian latent variables z_i, which are then all given as input to the final stage of the generative model $p_\theta(x|z_1, \dots z_k)$.

Graph-VAE This approach considers a multivariate latent variable $z = (z_1, z_2, \dots, z_K)$ and factors its latent distribution such that the set of parents of each z_j is a subset of $\{z_i, i < j\}$. For each pair (z_i, z_j) with $i < j$, a binary variable $c_{i,j}$ models whether an edge from i to j should exist or not, defining a conditional generative model on the vector c:

$$p_\theta(x, z|c) = p_\theta(x|z) \prod_j p_\theta(z_j|z_i \ s.t. \ c_{i,j} = 1) \tag{4.6}$$

In practice, the neural network structure is fixed and the conditioning on c is achieved by multiplying z_i by $c_{i,j}$ before feeding it as input to the neural network handling z_j. The inference model q_ϕ is defined using the same graphical structure and conditioning, but giving x as input to each distribution:

$$q_\phi(z|x, c) = \prod_j q_\phi(z_j|x, z_i \ s.t. \ c_{i,j} = 1) \tag{4.7}$$

For each c, the above formula defines an ELBO training of $p_\theta(x|c)$, where c fully encodes the graphical latent structure to be learned. c itself is learned by defining a distribution $p_\mu(x)$ as a product of Bernoulli distribution parameterized by a vector μ of their means: $p_\mu(c_{i,j}) = \mathcal{B}(\mu_{i,j})$. The final training objective is then defined by taking the expectation of the c-conditioned ELBO over $p_\mu(c)$, and estimating said expectation via Gumbel-Softmax parametrization to learn μ via gradient descent along θ and ϕ:

$$\mathcal{L} = \mathop{\mathbb{E}}_{c \sim p_\mu} \left[\mathop{\mathbb{E}}_{z \sim q_\phi(z|x,c)} \log \frac{p_\theta(x, z|c)}{q_\phi(z|x, c)} \right] \tag{4.8}$$

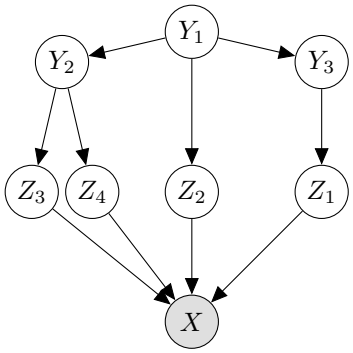


Figure 4.4: Example of a latent structure that could be learned by LTVAE. At the end of the training, if several Z_i variables are attached to the same Y_j (like Z_3 and Z_4 here), they can be fused into a single variable.

Although the resulting loss is not a lower-bound *per se*, the authors report that the approach works well in practice and that p_μ reliably converges to be a constant distribution, meaning the model does settle for a single latent structure.

Latent Tree VAE (LTVAE) This model learns the structure of the latent variables, as illustrated in Figure 4.4. The number of latent dimensions is still fixed in advance; the z_i are partitioned into clusters, multivariate variables noted $Z_1, \ldots, Z_B$, where B is not fixed in advance. Discrete latent variables $Y_1, \ldots Y_\ell$ are learned too: these Y_j are tree-structured, and each Z_i variable is linked to a single Y_j as its parent. The learning procedure concerns: the number of Y_j, their organization as a tree, the number of Z_i, the dimensionality of each, and which Y_j governs a Z_i.

The inference model $q_\phi(z|x)$ is a diagonal multivariate distribution over the whole joint variable z, defining an elaborate latent distribution $p\theta(z)$. While $p_\theta(x|z)$ and $q_\phi(z|x)$ are learned like in a mainstream VAE, the latent distribution $p_\theta(z)$ is learned via an EM-like algorithm, optimizing its parameters and computing the gradient $\nabla_z \log p_\theta(z)$ by message passing. The latent structure itself is also continuously optimized during the training search, modifying the graph via graph operators (adding a Y_i, splitting a Z_j into two new variables, changing the parent of a Z_j, ...) and evaluating the quality of a structure via the Bayesian Information Criterion (BIC) (Section 1.3.3).

4.4 Summary

This chapter describes the *Hierarchical Deep LVMs* framework, and presents the *Natural Minimal I-map* (NaMI) algorithm to construct an inference model for any Bayesian network, that captures all relevant dependencies while remaining as sparse as possible.

These approaches face similar challenges as deep networks (regarding the training stability and the flow of information through the model during training), still exacerbated by the intrinsically stochastic nature of the ELBO training criterion. In order to mitigate these issues, an option is to couple more tightly the structures of the inference and the generative model. The *NaMI* graph can be augmented by

adding direct edges from the observed variables to latent ones, improving information flow at the expense of the computational cost. Mainstream architectures require specific care in the design of the neural networks and their initialization.

Regarding the learning of the Bayesian Network graph, some encouraging preliminary results have been presented in the literature [He+18; Li+19].

Part II
Observation models

Chapters

Chapter 5
Probabilistic interpretation of observed variables

Contents

This chapter discusses the state of the art related to observation models. While latent variables can be arbitrarily modeled following the desired properties (as will be discussed in Part III), the modeling of observed variables is directly linked to their probabilistic interpretation, defining the *observation model*. The observation model can be best designed from the known internal structure of the data (sound, images, text...) as in Machine Learning in general, via pre-processing (e.g., normalization of numerical values, embedding of semantic values or manual feature construction), or based on the structure of the data (e.g. convolutional networks for image processing [LB98] or long short-term memory (LSTM) structures for sequential data [HS97]).

Section 5.1 presents *perceptual distances*, which adjust a Gaussian observation model w.r.t. a specific distance function. Section 5.2 is centered on the use of *auto-regressive* observation models, which decompose multi-dimensional examples as a sequence of conditionally-produced values. These very expressive models are notably applied to represent temporal or spatial structures (such as sound or images). Then, Section 5.3 focuses on an extremely powerful class of observation models built upon normalizing flows: *RealNVP*. Finally, Section 5.4 discusses the *posterior collapse* phenomenon, a not infrequent pitfall related to the use of very expressive observation models.

5.1 Perceptual distances for images

The Gaussian observation model is in general an intuitive choice: it represents the fact that the value x was measured with some (known or not) uncertainty. If

the measurement apparatus that produced the dataset has a known precision, this can be reflected in the model by specifying a Gaussian observation model with an appropriate fixed variance, reflecting the fact that making predictions that are more precise than the experimental apparatus is meaningless.

When applied to images, this interpretation is not always satisfying. A diagonal Gaussian distribution would make sense by representing the intrinsic noise of the photographic captor. This noise is however extremely small, implying a variance so small that the observation model is then essentially deterministic, defeating the purpose of integrating data structure into the model.

Work has been done to address this problem by trying to use the observation model to represent *semantic uncertainty* in the image data, by differentiating information that is semantically relevant from what could be called semantic noise. This is achieved by considering different distance measures in image-space than the euclidean distance induced by the pixel-wise Gaussian distribution.

5.1.1 Gaussian observation and choice of distance

The normal distribution can be specified to explicit its dependency with regards to the underlying metric. Let $\mathcal{X}$ denote the instance space, with a distance function on this space $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$. A normal-like distribution of parameters $\hat{x} \in \mathcal{X}$ and $\sigma^2 > 0$ is defined as:

$$\mathcal{N}_d(x|\hat{x}, \sigma^2) \propto \exp\left(-\frac{1}{2\sigma^2} d(x, \hat{x})^2\right) \tag{5.1}$$

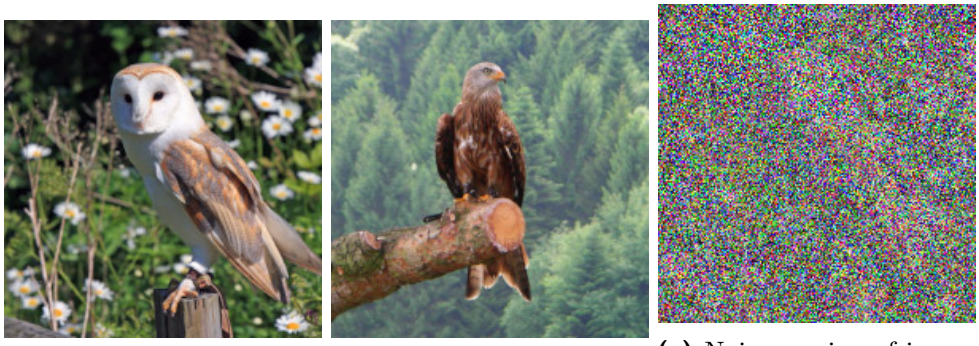
The usual isotropic normal distribution is then recovered for $d(x, y) = \|x - y\|_2$.

This formulation is appealing for the VAE, especially when viewed as an auto-encoder. The reconstruction term (Equation 3.3) then boils down to the expectation of the squared distance d^2 between the input datapoint x and the value predicted by the decoder neural network $\hat{x}$. The key issues are to define the appropriate distance measure, and the scale parameter σ^2.

Along this line, an appropriate distance is one making a real picture far from any image filled with noise, while making close two similar real images. The pixel-based Euclidean distance, $d(x, \hat{x})^2 = \sum_i (x_i - \hat{x}_i)^2$ does not satisfy these requirements, as illustrated on Figure 5.1.

The question thus becomes to find *perceptual distances*, reflecting the human perception of similarities and dissimilarities between images. Such similarity measures have been explored over the last decades, notably for content-addressed search in image databases [NG06; Bed+16], but most are inadequate to support the optimization and training of deep networks, not being differentiable.

Note that the use of an arbitrary distance function raises another issue, that of directly sampling the distribution $\mathcal{N}_d$. Actually, most models based on such $\mathcal{N}_d$ take the output $\hat{x}$ of the decoder NN *in lieu* of generated sample. While this approximation makes sense for $\sigma^2 << 1$, it might be abused in practice, entailing most undesirable effects. We shall come back to this important issue in Section 6.2.



(a) Image A **(b)** Image B **(c)** Noisy version of image A

Figure 5.1: Pixel-based Euclidean distance on images: the distance between the noisy and clean versions of image A is the same as the distance between image A and image B.

5.1.2 NN-based perceptual distances

Perceptual distances often rely on the use of neural networks (thus defining an implicit differentiable distance function). Formally, a NN is used to embed the images onto a vector feature space; the Euclidean distance in this feature space is used as perceptual distance, under the assumption that the feature space induces a more meaningful Euclidean distance than the pixel space. The question thus becomes finding an appropriate feature space as well as a tractable way to compute it.

Quite a few approaches reuse some intermediate representation learned by a neural network trained to achieve image recognition. Along this line, DeepSIM [DB16] uses an internal layer of the classifier AlexNet [KSH12], trained for image classification on the ImageNet [Den+09] dataset. The Euclidean distance computed in this internal feature space is combined with the Euclidean distance in pixel space and an adversarial feedback based on Generative Adversarial Networks (GANs)[1] [Goo+14; RMC16] to produce a hybrid training criterion, mixing the perceptual distance and the discriminant information, eventually yielding more realistic generated images.

Similarly, [Hou+17] use the distance defined from a VGGNet classifier [SZ15] also trained on ImageNet; they consider the Gaussian distribution built upon this distance to train a generative model on the CelebA faces dataset [Liu+15], by using a combination of several internal deep and shallow layers of the classifier network. The intuition is that, although the classifier has not been trained on the same data, both datasets are natural images; therefore the same feature space should be relevant to both. Moreover, using a classifier trained on a more general dataset than the generative model can also reduce overfitting.

Another approach is that of VAEGAN [Lar+16], that uses the latent representation of an adversarial discriminator (trained to discriminate the generated samples from the initial data). By construction, the discriminator aims to distinguish real images from generated ones, thus expectedly creating a feature space where the

[1]The main mechanism of GANs is the use of a so-called "discriminator network": a binary classifier aims to distinguish images from the real dataset from the ones produced by the generative model. The generative model is then trained by reversing the discriminator gradient, in order to fool the discriminator.

Euclidean distance would be relevant.

5.2 Autoregressive observation models

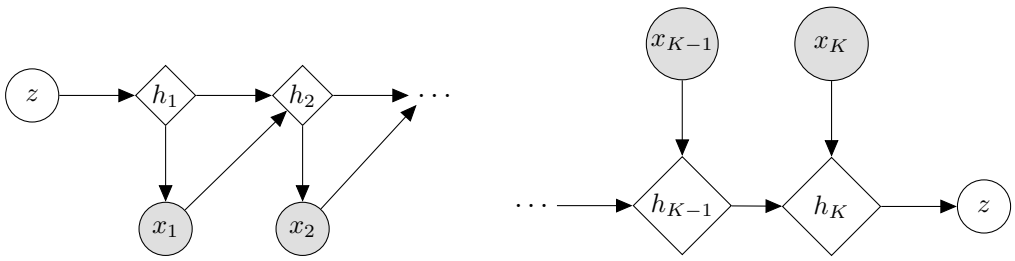
Rather than building further on the Gaussian observation model, another very general approach for handling handling multi-dimensional variables consists in splitting them into a list of scalar variables, with a chained distribution structure:

$$p_\theta(x|z) = \prod_i p_\theta(x_i|z, x_0, \ldots, x_{i-1}) \tag{5.2}$$

Such models, referred to as *autoregressive models*, can be extremely expressive. Actually, many such models were developed with no latent variables: the auto-regressive component $p_\theta(x_i|x_0, \ldots, x_{i-1})$ can be powerful enough to represent a complex distribution, and it can be trained directly using explicit maximum likelihood. Such models are especially well suited to data with a temporal or spatial structure, such as sound, images or language.

5.2.1 Recurrent Neural Networks for sequential data

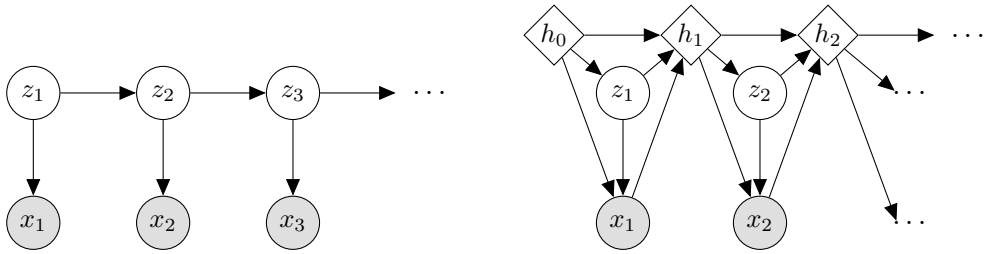
Auto-regressive observation models can be most simply implemented using recurrent neural networks such as LSTMs [HS97]. The latent variable z is used to define the initial state h_0 of the LSTM decoder, that yields eventually x as illustrated in Figure 5.2. In this approach [FA15; Bow+15], the associated inference network similarly relies on an LSTM.



(a) Generative model **(b)** Inference model

Figure 5.2: Recursive structures: (a) a recurrent neural network is used to represent an observation model from a single latent variable; (b) the associated inference model. Diamond-shaped nodes represent deterministic latent variables.

Another approach uses a recurrent structure for the latent variables themselves, akin a Hidden Markov Model (illustrated on Figure 5.3(a)). Introducing a sequence of latent variables can drastically improve the expressiveness of the model, accurately reflecting the sequential nature of the data. In practice, Deep LVMs built on recurrent architectures often use a combination of latent and deterministic nodes, such as the *Variational Recurrent Neural Network* (VRNN) [Chu+15] illustrated in Figure 5.3(b), that has been used for speech modeling. Even more complex structures have been proposed, such as the *Stochastic Recurrent Neural Network* (SRNN) [Fra+16] that uses a hierarchy of two sequences of latent variables linked to the sequence of observations.



(a) Hidden Markov Model **(b)** VRNN

Figure 5.3: Comparison of a Hidden markov Model (a) and a more complex latent structure as proposed by the VRNN [Chu+15] (b)

5.2.2 PixelRNN and PixelCNN for image generation

When considering an autoregressive factorization for image generation, one naturally considers each channel of each pixel of the image as different variables. Along this line, PixelRNN model [OKK16] fully embraces the discrete nature of computer images by modeling the variable of each channel as a 256-values discrete variable, parameterized using a softmax output. With no latent variables, this model can be trained directly to maximize the data likelihood.

In its simplest mode PixelRNN generates the image row by row (denoted Row-LSTM). A more powerful approach, BiLSTM, implements a recurrent structure w.r.t. both dimensions of the image, ensuring a better coherence of the generated image. The third variant, PixelCNN, has proved the most popular one, where the generation of a given pixel x_i depends only on its neighbor pixels, following the receptive field of a convolutional layer. The whole neural network thus is structured using convolutional neural networks (as opposed to recurrent ones, with significant improvement of the runtime performances at the cost of some expressiveness). Both variants are shown in Figure 5.4.

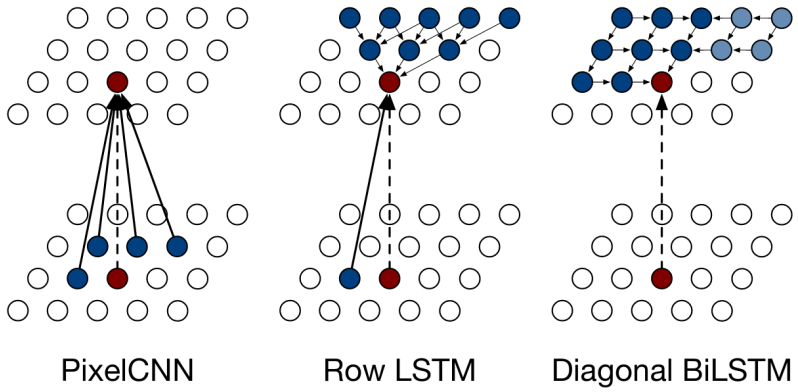


Figure 5.4: The 3 variants of PixelRNN. Top layer is the output, bottom layer is the input. Figure from [OKK16].

PixelCNN is acknowledged to be a powerful and expressive architecture. It has been used as decoder within an auto-encoder [Oor+16a], and combined with a Deep LVM to yield the PixelVAE [Gul+16]. Further improvements notably replace

the 256-way softmax output with a mixture of discretized logistic distribution[2], drastically reducing the output dimension of the network while retaining sufficient expressiveness in PixelCNN++ [Sal+17]. Later, PixelSNAIL [Che+18b] was proposed, improving the receptive field of the autoregressive process to better handle long-term dependencies.

5.2.3 WaveNet for audio generation

Audio data models lend themselves to a natural sequential factorization along the time steps. Accordingly, Wavenet [Oor+16b] was designed following the principles of PixelRNN. It similarly handles the prediction of the audio waveform x_t as that of a categorical variable, where the continuous signal value is binned into intervals (the width of which follows a logarithmic scale). The major novelty in Wavenet lies in the internal structure of the recurrent network based on dilated convolutions.

Recurrent neural networks are known to struggle with long-term dependencies, and while e.g. LSTM [HS97] mitigates this issue, it remains insufficient for raw audio – usually recorded from 16 kHz up to 44 kHz, or even higher. Tens of thousands of variables for each second of audio is more than any standard recurrent network can hope to remember. Wavenet thus introduces an internal structure specifically designed to handle long-term dependencies using several layers of dilated convolutions, each layer effectively doubling the receptive field of the output with regard to the previously generated values (Figure 5.5).

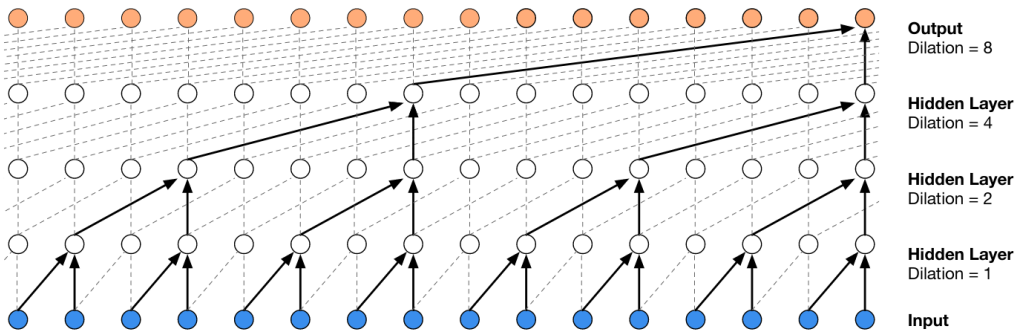


Figure 5.5: Representation of the dilated convolutions used in Wavenet. Each input is the concatenation of the output at previous time step and some conditioning for this time step. Figure from [Oor+16b].

In its initial version Wavenet was rather slow, generating samples x_t one at a time. To alleviate this shortcoming, [Oor+18] introduces the *Parallel Wavenet*. This neural structure similar to Wavenet uses a Normalizing Flow, transforming a sequence of latent values sampled from a fixed distribution $\epsilon_t \sim p(\epsilon)$ into a sequence of observations $x_t \sim p(x_t|\epsilon_1 \dots \epsilon_t)$. The benefit of this structure is that x_t no longer depends on the values of the previous timesteps $x_0, \dots, x_{t-1}$, allowing each value to be sampled in parallel in a much faster way.

[2]The discretized logistic, a distribution similar to the Normal distribution, is likewise parameterized by a location and a scale parameter. A main difference lies in its distribution support, set to an integer interval as opposed to $\mathbb{R}$. The interested reader is refferred to [Sal+17] for a more comprehensive presentation.

However, as discussed in Section 2.3.3, Normalizing Flows are trained by minimizing their Kullback Leibler divergence to an energy model (as opposed to, by maximum likelihood). Accordingly, the Parallel Wavenet is trained using a regular Wavenet as its target energy model. The complete training is thus achieved in two steps: first a regular Wavenet is trained on the dataset, then the Parallel Wavenet is trained using Wavenet as a teacher, akin a Network Distillation scheme [HVD15].

Wavenet is extensively used for speech synthesis, conditioning the synthesis on a phonetic encoding of the sentences to generate, with such a quality that Wavenet was used to voice the Google assistant when it was deployed in Oct. 2017[3].

5.3 RealNVP and flows-based observation models

Normalizing Flows (Section 2.3.3), being very expressive and able to approximate almost any well-behaved distribution, are a very powerful option for building observation models. They however need to be adapted to learn a distribution from a dataset (as opposed to an energy function).

The adaptation is straightforward: instead of defining the transformation as $x = f_\theta(\epsilon)$, it is reversed as $\epsilon = f_\theta(x)$. The density equation then becomes:

$$\log p_\theta(x) - \log |\nabla_x f| = \log \pi(\epsilon) \tag{5.3}$$

This formulation supports a maximum likelihood training from the dataset:

$$\log p_\theta(\mathcal{D}) = \sum_{x \in \mathcal{D}} \log p_\theta(x) = \sum_{x \in \mathcal{D}} \Big[\log |\nabla_x f| + \log \pi(f_\theta(x)) \Big] \tag{5.4}$$

However, this only formulation is limited as $p_\theta(x)$ cannot be efficiently sampled unless f_θ^{-1} can be easily computed. This limitation was initially addressed by the use of *real-valued non-volume preserving* (realNVP) transformations [DSB17], both invertible and suitable for use in Normalizing Flows.

The general idea of realNVP is to split the input and output multi-dimensional variables in two: $x = (x_1, x_2)$ and $\epsilon = (\epsilon_1, \epsilon_2)$, and consider the following transformation f_θ, called an *affine coupling* (where $*$ is an element-wise multiplication):

$$f_\theta : \begin{cases} \epsilon_1 = x_1 \\ \epsilon_2 = x_2 * s_\theta(x_1) + t_\theta(x_1) \end{cases} \tag{5.5}$$

As the transformation is conditioned by x_1, which is fixed, it can be easily inverted (where $\div$ is an element-wise division):

$$f_\theta^{-1} : \begin{cases} x_1 = \epsilon_1 \\ x_2 = (\epsilon_2 - t_\theta(\epsilon_1)) \div s_\theta(\epsilon_1) \end{cases} \tag{5.6}$$

Although this transformation is simple in itself, stacking several instances of it (with different parameters), and alternating the role of the two halves of the variables by permuting them[4] yield powerful representations. In particular for the application

[3] https://deepmind.com/blog/article/wavenet-launches-google-assistant

[4] Permutations of the components of x are invertible operations whose Jacobian determinant is 1.

to images, [DSB17] propose to split the pixels of the image in two groups along a checkerboard pattern; both halves thus cover the whole image, allowing for coherent transformations. The t_θ and s_θ functions are then learned using convolutional neural networks.

Further improvements and variations of realNVP have been proposed:
Glow [KD18] replaces the permutation step by an invertible 1×1 convolution, of which permutations are a special case.
Flow++ [Ho+19] introduces a more complex class of coupling layers, and proposes a new method to address *dequantization* [5] based on the ELBO and using another Normalizing Flow.

5.4 The Posterior Collapse Phenomenon

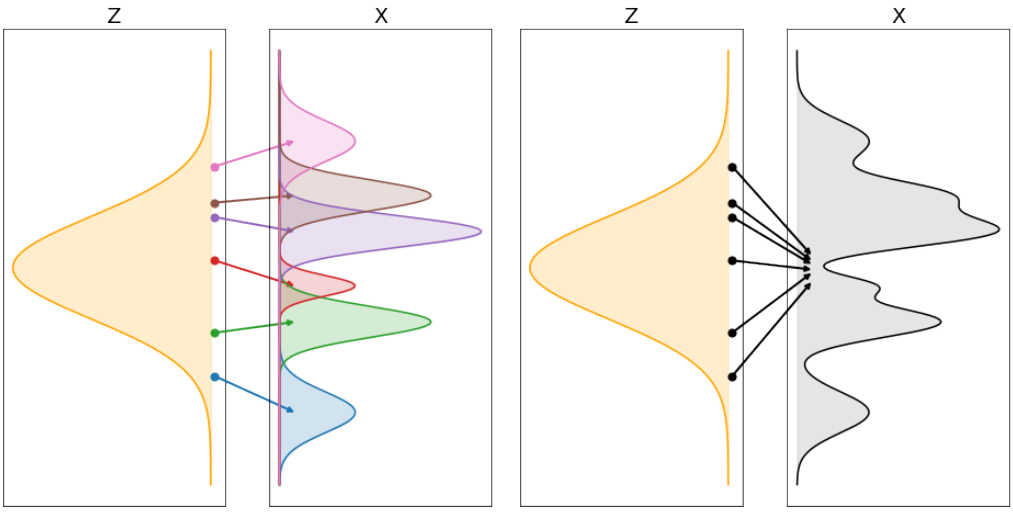
Autoregressive and realNVP-based models, initially introduced as standalone generative models, are mostly used as such. They both have conditional variants, and can be used as observation models within a Deep LVM; along this line PixelVAE [Gul+16] combines VAE with a PixelCNN decoder. However, in general ELBO training of these models proves difficult due to a phenomenon known as *Posterior Collapse* (PC).

The PC occurs when the Deep LVM involves a very powerful observation model, that would be capable of approximating well the dataset while entirely ignoring the value of the latent variables (Figure 5.6). Along this line, the inference model q_ϕ receives almost no feedback from the data while the ELBO draws it to match the latent distribution $p_\theta(z)$, and it *collapses* to this distribution: $\forall x : q_\phi(z|x) = p_\theta(z)$. The model thus reaches a good optimization of the ELBO while completely ignoring the latent variables [Che+17], which generally goes against the intended goal of introducing such variables in the first place.

It is currently believed that the posterior collapse occurs due to the training dynamics of the model [He+19; Luc+19]. In the early training stages the inference model q_ϕ poorly approximates the real posterior distribution $p_\theta(z|x)$ associated to these complex observation models, and the inferred z is thus irrelevant to improve the prediction $p_\theta(x|z)$. The poor quality of this approximation is blamed on the insufficient optimization of the inference model, making it *lag behind* the observation model [He+19].

Several approaches have been developed to mitigate the PC effect. [Luc+19] consider an annealing process on the ELBO, setting a weight α on the term $D_{KL}(q_\phi(z|x)\|p_\theta(z))$, initialized at 0 and gradually increased to 1, then recovering the original ELBO. Therefore the observation model can freely use the z variable in the early training stage, as the inference model is then allowed to deviate a lot from the latent distribution $p_\theta(z)$. The increase of α then slowly drives the latent variable z back to $p_\theta(z)$, while the observation model has already learned to use it,

[5] The dequantization problem arises when training continuous probabilistic models (such as realNVP) on data that have been discretized (such as images whose pixel values are in $[|1; 255|]$). In that case, the target distribution actually lives in a discrete grid-like subspace of the data space, making it a collection of point masses. Such a distribution, beyond the reach of most continuous models, may cause the training to diverge. Dequantization is the process of adding a small continuous noise to the discretized data to make it continuous again.



(a) Gaussian observation VAE **(b)** Model suffering of posterior collapse

Figure 5.6: Illustration of posterior collapse. In the Gaussian VAE (5.6(a)) each latent value Z is mapped to a different distribution in the data space X, the mixture of all of them making the actual generative distribution $p_\theta(x)$ (black outline). When posterior collapse occurs (5.6(b)), all latent values are mapped to the same distribution in the data space and the observation model represents the generative distribution by itself: $\forall z : p_\theta(x|z) = p_\theta(x)$.

and follows the movement. [Raz+19] introduces the δ-VAE, where the distribution classes for $q_\phi(z|x)$ and $p_\theta(z)$ are chosen in such a way that their KL-divergence is lower-bounded by some hyper-parameter $\delta > 0$, preventing the total collapse of the inference model. Likewise, this makes it free for the observation model to somehow use the latent variable and opening the door for more. Finally, [Ale+18] modifies the ELBO training objective by reformulating it as a constrained optimization problem and then generalizing it to design a new training objective that forces the model to use the latent variable.

The most effective results obtained by combining Deep LVMs with powerful observation models rely on the use of discrete latent variables. An example thereof is PixelVAE++ [Sad+19], that improves PixelVAE taking inspiration from PixelCNN++, replacing the latent space with a discrete variable, and learning the latent distribution $p_\theta(z)$ as an RBM, mimicking DVAE [Rol17]. The *Vector-Quantized VAE* (VQ-VAE) [OVK17] and its refinement VQ-VAE-2 [ROV19] significantly modify the model formulation by learning a set of latent embeddings and a discrete distribution over them, effectively handling the inference model $q_\phi(z|x)$ as a mixture of point mass distribution (a mixture of Diracs). This makes it impossible to interpret the learning criterion as a lower bound of the data likelihood. Nevertheless, VQ-VAE reaches very competitive performance in the context of generation of realistic high-definition images.

5.5 Summary

This chapter focuses on the probabilistic interpretation of observed variables. This interpretation (and the associated probabilistic model choice) defines the lens through which the LVM can interact with the data, thus conditioning the success of the training.

Several approaches have been designed to account for data properties or structures: Perceptual distances are meant to emulate the human-perceived similarity between images. Autoregressive models (e.g. Wavenet or PixelRNN/CNN) factor the model as a succession of conditional distributions, yielding very expressive models for discrete sound or image data. Inspired from autoregressive flows, RealNVP builds an expressive class of observation models enabling to generate high-dimensional examples.

In counterpart for these gains in expressivity, the above approaches are highly susceptible to posterior collapse, ignoring the latent variables and learning the data distribution from the only observation model. Several adjustments have been proposed to avoid this issue [Luc+19; Raz+19; Ale+18].

Overall, these expressive observation models tend to weaken the link between the latent variables and the observed data. Specifically, the latent representation can then be less easily exploited in order to analyze the data: expressive observation models can induce, and compensate for, a poorly informative latent representation.

Chapter 6
The Manifold Hypothesis and Quasi-Deterministic Observations

Contents

This chapter presents and discusses the relationship between the model and the data in relation with the so-called Manifold Hypothesis [Rif+11; BWS15; FMN16; CCR21], considering that the data lie near a manifold. Ideally, an optimal latent representation would define a mapping from some (low-dimensional) space onto the data manifold, that is, provide a parametric map of this manifold.

The question thus becomes when and how can an LVM appropriately characterize the data manifold. This chapter presents the first main contribution of this thesis: a case study under the assumptions of a large sample limit and an infinitely expressive latent representation, extending a former article [BS20b].

The subject of information flow throughout the a VAE and its relation to the learning process has been studied thoroughly. One notable angle of analysis is the interpretation the latent space as a noisy communication channel between the encoder and decoder in the light of Information Theory [BK18; RV18; ZSE18; Raz+19; Zhe+19; DSL20]. An other lies on linking the VAE to Principal Component Analysis (PCA) [Dai+18; Luc+19]. This chapter and the next one provide a complementary analysis focusing on the geometrical interpretation of the training objective in light of the Manifold Hypothesis, and the impact of the observation model on the training process.

6.1 The Manifold Hypothesis

Most datasets are endowed with an internal structure, that is implicitly characterized as not every element in the considered embedding space $\mathcal{X}$ is a valid sample: a random matrix of pixels does not correspond to a photorealistic image, a random image does

not represent a person in general, a random sequence of numbers does not encode a voice waveform, etc. (On the other hand, if any sample in a high-dimensional space were a valid one, then in general, the available data resources would be insufficient to support learning).

The structure of the real data samples in a high dimensional space, referred to as *Manifold Hypothesis (MH)*, is formulated as follows:

> The considered dataset in a high-dimensional space is actually concentrated around a low-dimensional manifold embedded in this space.

Note that the dataset is assumed to concentrate near a low-dimensional manifold, as opposed to, be contained in the manifold. The latter assumption would be unrealistic, due to the data noise. The MH thus actually assumes that there exists a low-dimensional representation of the dataset with no significant loss of information; it does not assume that the data exactly lie in some "true" manifold of the embedding space.

The MH is generally assumed in the domain of computer vision: the region of natural images is continuous and connected[1] while most pixel matrices are "visibly" not natural images. The same applies for audio data. The MH can have an adverse impact on some learning algorithms, such as Generative Adversarial Networks: the fact that the data lie in a restricted region of the embedding space can hinder or prevent the training of the model [AB17].

If such an appropriate manifold were known, then any datapoint could be decomposed in two parts: a location on the manifold, plus some small deviation from the manifold. Along this line the semantically useful information would be said location on the manifold, while the deviation would represent a small noise. This decomposition, akin to Manifold Learning [Cay05], motivates the following sections.

6.2 Quasi-deterministic observation models

Latent Variable Models can be understood with respect to the MH. The learned distribution over observed variables, $p_\theta(x)$, can be viewed as a mixture of the observation model, indexed by the latent variables z: $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$. Along this line, the LVM training aims to pave the data manifold with instances of its observation model (Figure 6.1).

If all considered observation models are low-variance distributions, then the learned observation model $p_\theta(x|z)$ itself establishes a mapping from the z space to the data manifold, the transformation $z \to x$ being almost deterministic (in the low variance case). In order for the learned observation model to characterize the data manifold, the latent variables z must contain all relevant information for characterizing a given example within the whole dataset; the latent representation could then support other downstream tasks[2].

[1] There is a very natural and intuitive way to continuously transform a photograph into another: making both of them part of the same filmed video that goes from one point of view to the other.

[2] Such task include for example using the VAE as a dimensionality-reduction algorithm, comparable to a non-linear PCA [Luc+19]. Part III further expands on design of the latent structure of the LVM to extract sought information or enforce desirable properties.

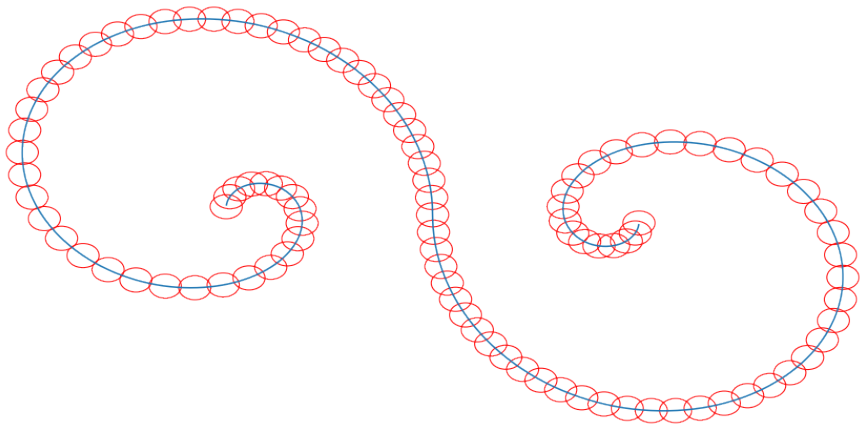


Figure 6.1: Example of approximate paving of a manifold (blue line) with observation models (red circles). Each circle representing for example a small-variance Gaussian distribution.

Let us define the notion of *quasi-deterministic observation model* as an observation model where the generative process of x given z is a deterministic procedure f_θ augmented with some noise ϵ_θ: $x = f_\theta(z) + \epsilon_\theta$. Thus $p_\theta(x|z)$ can be formulated in terms of $p_{\epsilon,\theta}(\epsilon|z)$, the density of the noise model:

$$p_\theta(x|z) = p_{\epsilon,\theta}(\epsilon = x - f_\theta(z)|z) \tag{6.1}$$

Though the noise can in principle depend on z (inducing heteroscedastic models), the idea is that after training, the learned noise model will have a small variance such that the deterministic mapping $\hat{x} = f_\theta(z)$ offers a decent alternative to actually *sampling* $p_\theta(x|z)$.

In the following two examples of quasi-deterministic observation models are analyzed. The simplest one (diagonal Gaussian observation) has limitations that may force the model to fail to differentiate the manifold from the noise. The second illustrates how these limitations can be overcome without leaving the quasi-deterministic context, by building a hierarchical observation model.

6.2.1 The Gaussian observation and its limitations

When considering Gaussian observation models, the observed variable x involves a mean noted $f_\theta(z)$ and a covariance matrix. The low variance of the noise model holds iff this covariance matrix have small eigenvalues compared to the overall variance of the considered dataset.

In practice, three options have been considered in the literature. A first option considers an isotropic covariance matrix $\sigma^2\mathbf{I}$ with a constant σ. This option is retained in many papers, notably in computer vision where the VAE is mostly viewed as a regularized auto-encoder[3] [Hou+17; Dor+17; Hua+18; GSS20; Gho+20], trained from the squared error reconstruction loss $\ell(x, \hat{x}) = \|x - \hat{x}\|^2$. This loss is equivalent

[3]It is actually difficult to measure precisely how widespread it is used: a large fraction of articles about VAEs do not explicit their observation models.

to a Gaussian observation model with a fixed isotropic variance[4] $\sigma^2 = 1/2$. However, a variance of $1/2$ might seem unreasonably large (when considering the image data as a vector whose coordinates are in $[0; 1]$), preventing $f_\theta(z)$ from being a good approximation for $p_\theta(x|z)$. As will be shown in Section 6.3, such a high σ can be considered the main cause for the blurriness long observed in VAE-generated images.

It can be noted that in this context, changing the fixed observation variance σ^2 is equivalent to applying a factor in front of the KL part of the loss, as done by β-VAE[5] [Hig+17]. Empirically, it is observed that increasing the value of β improves the disentanglement of the VAE latent space; in counterpart, the images generated from the decoder ($\hat{x} = f_\theta(z)$) are increasingly blurry, which is coherent with it introducing an observation noise that is too large relative to the data characteristics. A detailed comparison between β-VAE and quasi-deterministic observation models in terms of training dynamics is given in Appendix 7.A.

Other usual options consists in considering an isotropic covariance matrix whose variance is learned as a global parameter of the model [RV18], or as a diagonal covariance matrix, whose components are the output of the decoder neural network alongside the mean [KW14; MF18]. As will be shown in Chapter 7, learning the variance of the observation model has a positive impact on the dynamics of the learning process, compared to considering a fixed variance, even if adjusted using preliminary experiments.

The limitation of all above options is twofold: firstly, they lack the expressiveness of a full-rank covariance matrix; secondly, they assume an uncorrelated noise between the dimensions of x. If the data structure involves some correlated noise, the above models, being unable to model it directly, might settle for a smaller variance than appropriate. Hence, a part of the noise would eventually be encoded in the latent space alongside the actual information, as illustrated by Figure 6.2. On the other hand, the option of considering (learning) a full-rank covariance matrix is impractical, especially so in high-dimensional domains like computer vision.

6.2.2 Hierarchical quasi-deterministic observations

The use of quasi-deterministic observation models can however be combined with taking advantage of the structure of the data: elaborate models can be defined using an adequate LVM structure. While the properties of high dimensional data are counter-intuitive in many cases, the manifold assumption can be invoked to enforce the sought properties or behaviors, even while sticking to the quasi-deterministic framework.

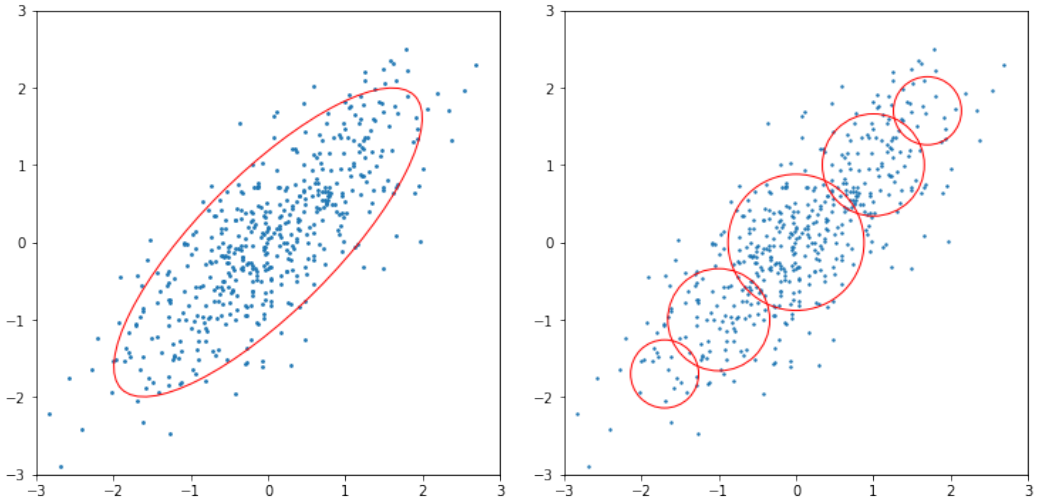
[4] The negative log-likelihood of a Gaussian distribution of mean $\hat{x}$ and of variance $\sigma^2 = 1/2$ reads:

$$-\log p(x|\hat{x}, \sigma^2) = \frac{1}{2\sigma^2}\|x - \hat{x}\|^2 - D\log\sigma = \|x - \hat{x}\|^2 + \frac{D}{2}\log 2$$

The constant term $\frac{D}{2}\log 2$ does not affect the optimization process.

[5] In the isotropic Gaussian observation, the variance σ^2 acts as a global multiplicative parameter in front of the squared error. This makes it possible to interpret it as a hyperparameter reweighting the reconstruction and KL parts of the ELBO. By introducing the parameter $\beta = \sigma^2$, the formulation becomes equivalent to that studied in the β-VAE:

$$\mathcal{L}_{\beta\text{-VAE}} = \beta D_{KL}\big(q_\phi(z|x)\|p_\theta(z)\big) + \mathop{\mathbb{E}}_{z\sim q_\phi}\|x - f_\theta(z)\|^2 \tag{6.2}$$



(a) Full-rank normal distribution **(b)** Mixture of diagonal normal distributions

Figure 6.2: Illustration of the inappropriately small variance imposed by a diagonal observation model. (a) The cluster of points can efficiently be fitted by a single full-rank normal distribution. (b) It requires a mixture of smaller variance normal distributions if these are constrained to be diagonal. The second observation model requires additional information to be captured in the latent variables, compared to the first one.

Such an adequate LVM structure is that of Laplacian Pyramid representations, reflecting the multi-scale structure of images and explored e.g. in LapCVAE [Dor+17]. The article introduces several modifications to the VAE; in the following we focus on the use of intermediate observed variables, which can be considered independently of the other modifications.

The core idea consist in generating the image along a sequence of resolution steps. First a small image x_0 is generated, then this image is upscaled, and the model generates a update δx_1 that is added to the upscaled version to make a larger image with more details: $x_1 = upscale(x_0) + \delta x_1$. The process is iterated until the sought size is reached, generally with an upscaling of a factor 2 in each step. The sequence of generated images $x_0, x_1, \dots$ is thus defined by ($\mathbf{z}$ denoting the set of latent variables):

$$\begin{cases} x_0 & \sim & p(x_0|\mathbf{z}) \\ \delta x_{i+1} & \sim & p(\delta x_{i+1}|x_i, \mathbf{z}) \\ x_{i+1} & = & upscale(x_i) + \delta x_{i+1} \end{cases} \tag{6.3}$$

This is illustrated by Figure 6.3: each observed variable x_{i+1} depends on the previous one x_i and some subset of the latent variable.

Along this setting, for each full image noted x_N, a sequence of x_i (observed variables generated by downscaling x_N) is used in the training procedure. The process, decomposing the data into multiple scales (as appropriate for images), guides the model along this structure.

Finally a hierarchical quasi-deterministic observation model thus builds x_i at

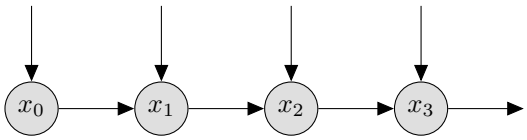


Figure 6.3: Illustration of a multiscale observation model.

each scale as:

$$x_i = \quad upscale(x_{i-1}) + f_{i,\theta}(x_{i-1}, \mathbf{z}) + \epsilon_{i,\theta}(x_{i-1}, \mathbf{z}) \tag{6.4}$$

Both the prediction $f_{i,\theta}$ and noise $\epsilon_{i,\theta}$ functions take as arguments the previous image, x_{i-1} and some subset of the latent variables. This hierarchical observation model can be combined with any latent structure as appropriate for the application domain and goals. LapCVAE for example introduces a hierarchy of latent variables $z_1, z_2, \ldots$ depending on the observed variables, and conditioning them as depicted in Figure 6.4.

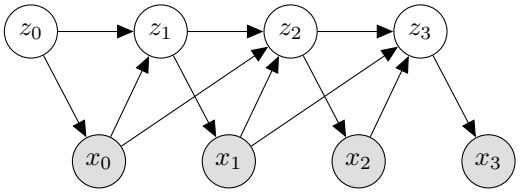


Figure 6.4: LapCVAE. The graphical model illustrates the dependencies among the observed x_i and latent z_i variables. The dependency of z_i on z_{i-1} is achieved through concatenation: each latent variable contains the concatenation of all previous ones, plus new dimensions inferred from the two previous generated images.

6.3 Noise Variance and data resolution

A first contribution of the presented manuscript is to analyze how the structure of the observation model governs the possibilities of the inference model *even under the assumption of a large sample limit and infinite capacity (representation power) of the inference model*. This contribution is analyzed in view of the Manifold Hypothesis.

A primary remark is that the variance σ of the observation model noise governs the resolution and degree of approximation of the data by the model: the training criterion involves the likelihood of the dataset w.r.t. the model at hand, where the noise amplitude is σ. Accordingly, any data pattern that would be made indistinguishable by this noise amplitude, is lost, for the same reasons as discussed in relation with the Posterior Collapse phenomenon (Section 5.4). In the β-VAE case, increasing β amounts to increasing the variance of the noise, which in turn prevents the learned model from grasping the fine-grained patterns of the data. Eventually, such a high variance results both in defining a blurry model, and reducing the amount of information captured by the latent variable, making it in turn more amenable to disentanglement[6].

[6]All structure being fixed, the model needs to pack less information into a latent space of the

The proposed interpretation is backed upon a theoretical result (section 6.3.1). Experiments on synthetic datasets situated on a 1D manifold illustrate the interplay among the model variance (fixed or learned) and the quality of the learned model.

6.3.1 Modeling an hypersphere

The impact of the observation model is examined under three assumptions:

1. The inference model used to train the VAE is assumed to be exact: $q_\phi(z|x) = p_\theta(z|x)$ for all x. Therefore the ELBO is tight, and its maximization boils down to maximizing the likelihood $\mathbb{E}_{x\in\mathcal{D}} \log p_\theta(x)$.

2. The predictor function f_θ of the observation model is sought in a hypothesis space with infinite capacity (encoded by a neural network with arbitrary expressiveness).

3. The optimization process is assumed to reach the exact optimum of the criterion; the dataset is assumed to be an infinite uniform sample on the target manifold.

Therefore, the VAE only depends on the observation model, a Gaussian observation model with an isotropic covariance matrix $\sigma^2\mathbf{I}$, for some fixed value of σ and whose mean is predicted by a neural network f_θ.

Let us further assume that the sought manifold is a hypersphere of radius R of center 0, in dimension D (in $\mathbb{R}^D$).

Then:

Theorem 6.1. *Under the previously described assumptions, if $\sigma \geq \frac{R}{\sqrt{D-1}}$, then the global optimum is reached when f_θ is a constant function at 0.*

In other words, if the variance σ^2 is too large, the designed VAE will learn *nothing* of the dataset, even though it has infinite capacity. The proof of this theorem, detailed in Appendix 6.A, consists in analytically characterizing the optimal distribution $p_\theta(x)$; this characterization is made possible under the considered assumptions. Let μ be the mean of the observation model and output of the predictor neural network f_θ. It is shown that, when optimal, the distribution of μ is uniform over an hypersphere of radius $r \geq 0$, where r is given by maximizing the following quantity:

$$\ell(r) = e^{-\frac{r^2}{2\sigma^2}} \int_{\theta=0}^{\pi} \exp\left(\frac{Rr}{\sigma^2}\cos\theta\right) \sin^{D-2}(\theta) d\theta \tag{6.5}$$

When $\sigma \geq \frac{R}{\sqrt{D-1}}$, ℓ reaches its maximum for $r = 0$, which concludes the proof of Theorem 6.1.

The behavior of ℓ is illustrated in Figure 6.5 (left, for $D = 2$; right, for $D = 10$), plotting $\ell(r)$ in the 2D plane given by R/σ and r/σ, with the curve of optimal r depicted in black. It is seen that the optimal r remains 0 until R/σ grows larger than respectively $\sqrt{2}$ and $\sqrt{10}$. The actual threshold is larger than the one given in Theorem 6.1 ($\sqrt{D}$ rather than $\sqrt{D-1}$). After the threshold, r/σ quickly grows with R/σ, and converges asymptotically to $r = R$. As σ shrinks to 0, r converges to R.

same size, and can afford less precise latent encoding. As a result, it can more easily follow the tendency for disentanglement that a factorized inference model suggests (3.4).

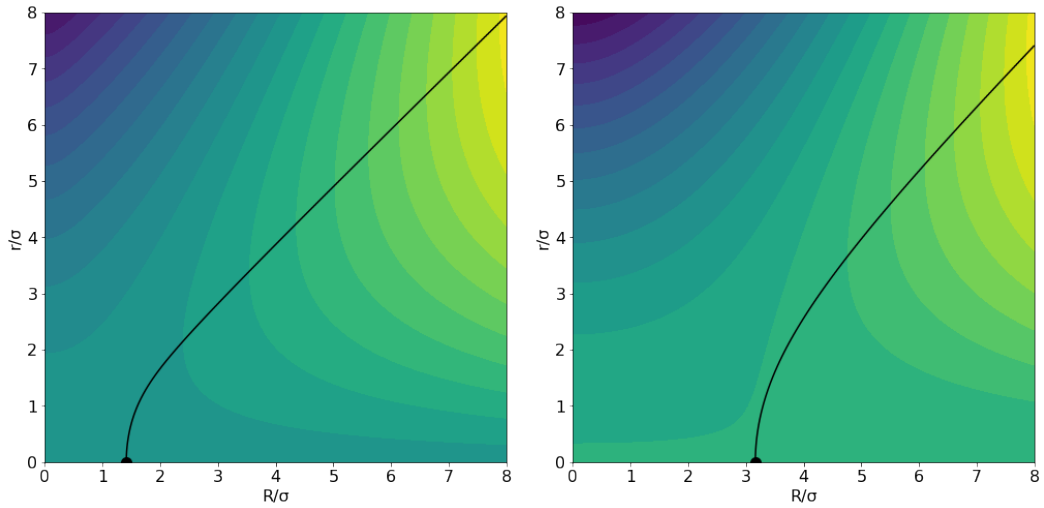


Figure 6.5: Contourplot of ℓ (Equation 6.5) in the 2D plane given by r/σ and R/σ. Left: $D = 2$. Right: $D = 10$. The background color indicates the value of the criterion (increasing from blue to yellow on a logarithmic scale). Black curve indicates the optimum of $\ell(r)$. The threshold value $\sqrt{D}$ is indicated as a black dot on the horizontal axis.

In view of the Manifold Hypothesis, Theorem 6.1 establishes that a Gaussian VAE with fixed variance σ^2 is blind to data structures whose radius of curvature is smaller than[7] $\sigma\sqrt{D-1}$. In other words, the VAE is doomed to miss such "details" of the data distribution if their scale is too small compared to its (fixed) variance; the VAE then yields an overly smooth manifold.

According to this result, the fixed variance of the VAE model σ sets a lower-bound on the smoothness of the manifold the VAE can learn. More precisely, σ defines a trade-off between the noise (missed) and the information (stored in the latent representation): too large, and the fine-grained patterns will be missed; too small and the noise will be stored in the latent representation.

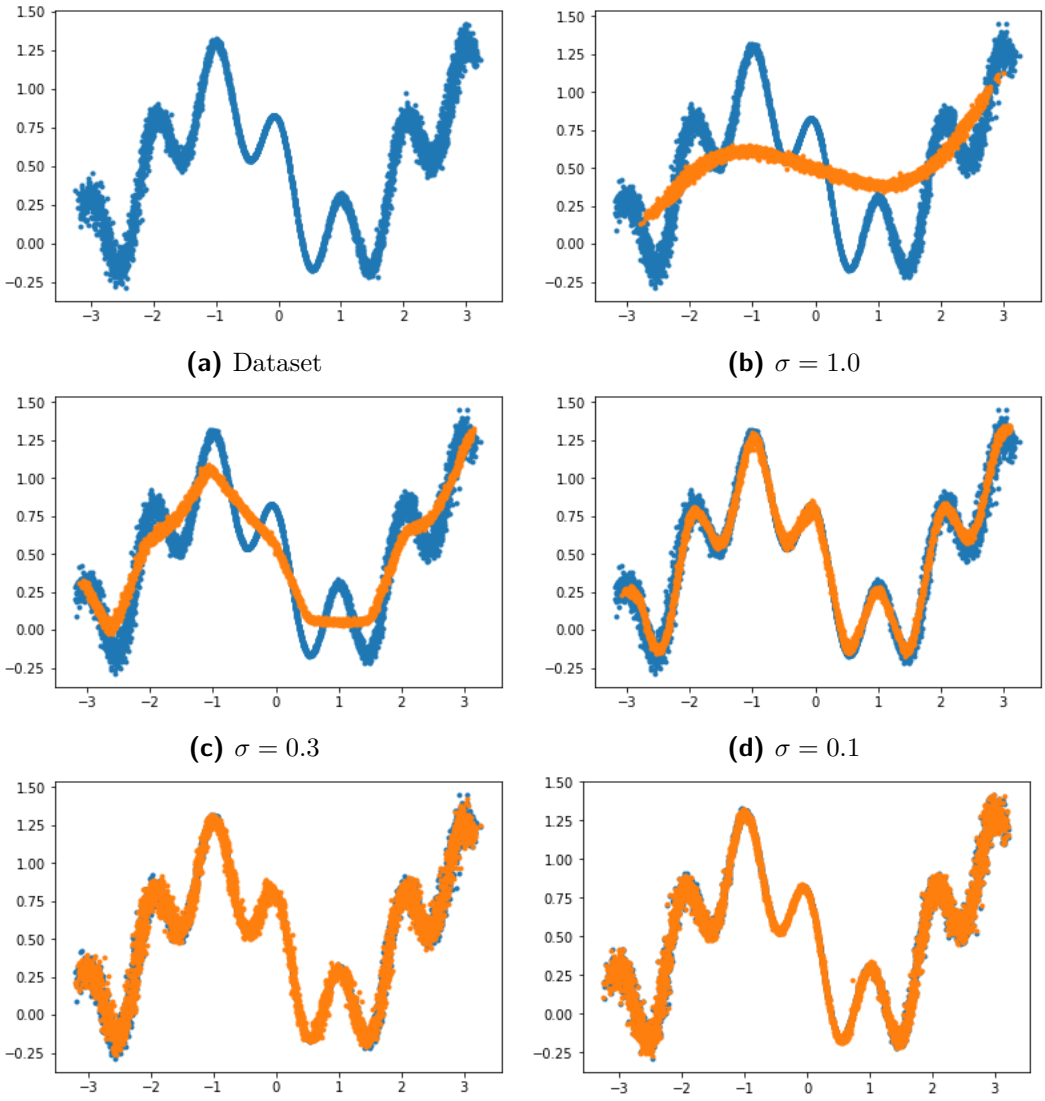
6.3.2 Experimental study of manifold approximation

This claim is experimentally supported on a synthetic 2D dataset Figure 6.6. A 1D sinusoidal manifold is defined in R^2, and the synthetic dataset is generated from this manifold with a isotropic Gaussian noise, the amplitude of which varies along the manifold (heteroscedastic distribution).

A powerful VAE is generated from this dataset along a large sample limit process (10.000 new samples are generated in each epoch) with a Gaussian observation model of various σ values, and an "infinitely powerful" generative model[8].

On this synthetic dataset, the relation between the fixed σ and the curvature of the learned manifold can be summarized as follows:
For a large σ value, the model yields a smoothed version of the manifold; for very small σ values, the noise is interpreted as part of the manifold, increasing the

[7] Note that Figure 6.5 suggests the cutoff is actually at $\sigma\sqrt{D}$.
[8] Using a Resnet-based architecture with a 10 dimensional latent variable z.



(a) Dataset **(b)** $\sigma = 1.0$

(c) $\sigma = 0.3$ **(d)** $\sigma = 0.1$

(e) $\sigma = 0.03$ **(f)** $\sigma = 0.01$

Figure 6.6: Impact of the observation model variance σ^2 on the learned manifold (output of the predictive neural network $\hat{x} = f_\theta(z)$, with $z \sim q_\phi$). (a): The original manifold involves several regions, with a varying curvature, and a varying noise. Original points are in blue, learned manifold in orange. (b-c): Large values of σ result in a smoothed approximation of the manifold, where highly-curved regions have been lost. (d): A well calibrated σ adequately separates the manifold structure from its noise. (e-f) Too small σ values force the model to encode the noise into the latent variable as well; the dimension of the learned manifold increases from 1 to 2.

dimensionality of the latent space. This relation between the approximating manifold and the scale parameter of the observation model and the thresholding effect similar to a phase transition has also been experimentally observed by [RV18].

The precision of the observation model, governing the manifold approximation, also has a dramatic impact on the generative model. The two manifolds $\hat{x} = f_\theta(z)$ respectively obtained by reconstruction ($z \sim q_\phi$, Figure 6.6) and generation ($z \sim p_\theta$,

Figure 6.7) differ depending on σ:
Both manifolds are very similar when σ is adequate or too large. Quite the contrary, when σ is too small, the latent variable z seemingly fails to adequately capture the manifold, resulting in a poor coverage of the latent distribution $p_\theta(z)$ by the aggregated inference model $q_\phi(z)$. This eventually causes the generative process to yield poorly realistic samples.

It must be emphasized that this failure when σ is small is related to the training dynamics; we shall return to this issue in Chapter 7.

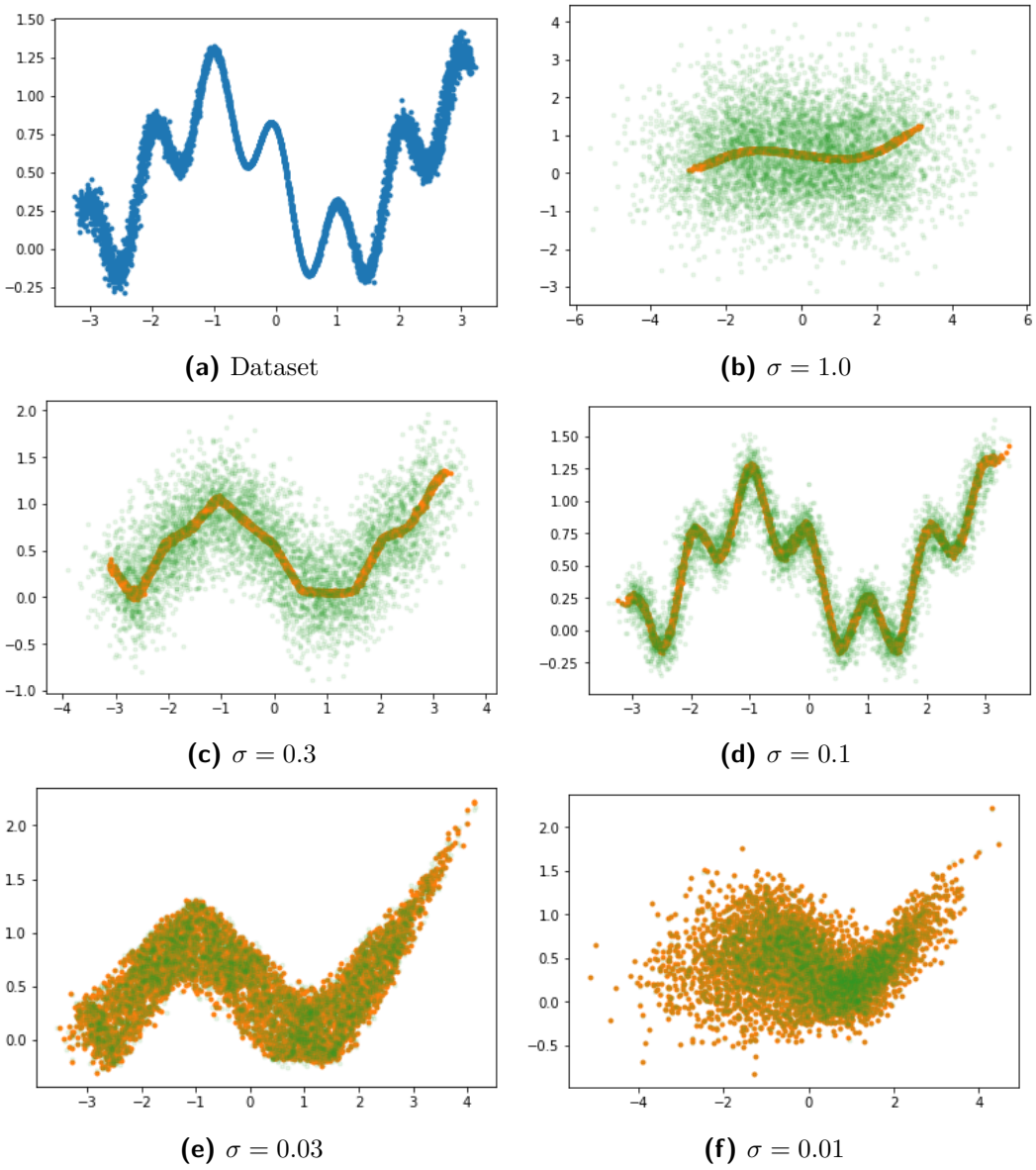
These remarks establish that, when using a quasi-deterministic observation model, the choice of the noise variance σ is a crucial design point. When considering a model with a fixed isotropic covariance, σ acts as regularization weight, balancing the KL-part of the ELBO ($D_{KL}(q_\phi(z|x)\|p_\theta(z))$) and the reconstruction term ($\mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z)$), as classically done in practice in the VAE literature [Hou+17; Lar+16; Hig+17; Hua+18; GSS20].

6.4 Summary

The observation model aims at a probabilistic mapping between the space of latent variables and the observed data space. Under the manifold assumption, one might consider instead a *quasi-deterministic observation model*, involving a deterministic mapping onto the data manifold with an additional small noise. Some examples of this approach, ranging from the Gaussian case to LapCVAE, are described.

Our claim is that the variance of this added noise governs the identification of the data manifold; as theoretically and experimentally shown (Theorem 6.1, Section 6.3.2), manifold regions with a high curvature compared to the noise variance are smoothed and the details are lost. At the other extreme, a too small noise variance leads the model to encode the data noise, increasing the effective dimension of the learned manifold.

The key question thus becomes to identify the noise variance: i) based on prior knowledge (e.g. related to known measurement uncertainties); ii) tuned as a model hyper-parameter; iii) or learned as yet another model parameter. The learning dynamics differs widely depending on the chosen option, as will be shown in Chapter 7.



(a) Dataset **(b)** $\sigma = 1.0$

(c) $\sigma = 0.3$ **(d)** $\sigma = 0.1$

(e) $\sigma = 0.03$ **(f)** $\sigma = 0.01$

Figure 6.7: Impact of the observation model variance σ^2 on the generative process. (a): the original data (in blue). The output of the predictive model ($\hat{x} = f_\theta(z)$ with $z \sim p_\theta(z)$) is in orange and the generated samples ($x = \hat{x} + \epsilon_\theta$) are in green. (b-c): When σ is too large, the generated manifold (orange) matches with the reconstructed one (Figure 6.6), and the added ϵ_θ noise ensures complete coverage of the dataset (note the change of axes scale). (e-f): When σ is too small, the generated manifold is actually quite different from the reconstructed one, suggesting that the latent representation fails to adequately capture the real manifold: $z \sim p_\theta(z)$ often generates unrealistic samples.

6.A Proof of Theorem 6.1

Proof of Theorem 6.1. Under the assumptions stated in Section 6.3.1, the aim is to characterize $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)dz$ at the optimum.

The Gaussian observation model is defined by:

$$p_\theta(x|z) = \frac{1}{(2\pi\sigma)^{D/2}} \exp\left(-\frac{1}{2\sigma^2}\|x - f_\theta(z)\|^2\right) \tag{6.6}$$

The prediction function f_θ can be discarded (e.g. through a change of variable $\mu = f_\theta(z)$); only the associated distribution $p_\theta(\mu)$ needs to be considered:

$$p_\theta(x) = \frac{1}{(2\pi\sigma)^{D/2}} \int_\mu \exp\left(-\frac{1}{2\sigma^2}\|x - \mu\|^2\right) p_\theta(\mu)d\mu \tag{6.7}$$

The considered dataset has a spherical symmetry around the origin; the observation model also is isotropic. Therefore at its optimum the distribution $p_\theta(\mu)$ must also have this spherical symmetry, i.e. it only depends on the norm of μ. After normalization, the sought probability distribution thus has the following form:

$$p_\theta(\mu) = \frac{q(\|\mu\|)}{A_{D-1}\|\mu\|^{D-1}} \tag{6.8}$$

with A_{D-1} the area of the unit $D-1$ hypersphere, and q a normalized probability measure ($\int_r q(r)dr = 1$).

Considering the likelihood of the dataset (and dropping constant terms related to q), it comes:

$$\mathcal{L} = \mathop{\mathbb{E}}_{x\in\mathcal{D}} \log p_\theta(x) = \int_{\|x\|=R} \log \int_\mu \frac{q(\|\mu\|)}{\|\mu\|^{D-1}} \exp\left(-\frac{1}{2\sigma^2}\|x - \mu\|^2\right) d\mu\, dx + \dots \tag{6.9}$$

Let r be the norm of μ and θ be the angle between x and μ. In spherical coordinates, the exponential can be reformulated as $\|x - \mu\|^2 = R^2 + r^2 - 2Rr\cos\theta$. The inner integral on μ depends only on R and θ. The integration on the other $D-2$ dimensions yields a multiplicative constant that can be moved out of the logarithm. After this rewriting, the contents of the logarithm no longer depends on x, making the outer integral trivial as well.

$$\mathcal{L} = A_{D-1} \log \int_{r=0}^{+\infty} q(r) \int_{\theta=0}^{\pi} \exp\left(-\frac{r^2 - 2Rr\cos\theta}{2\sigma^2}\right) \sin^{D-2}(\theta)d\theta dr + \dots \tag{6.10}$$

The term in the logarithm is the expectation over q of some function of r, it is thus maximized when q is a Dirac measure at the maximum value of that function. Thus at the optimum, $p_\theta(\mu)$ is the uniform distribution over a hypersphere of center 0. Only its radius r remains to be determined. Maximizing $\mathcal{L}$ is equivalent to maximizing the contents of the logarithm:

$$\ell(r) = e^{-\frac{r^2}{2\sigma^2}} \int_{\theta=0}^{\pi} \exp\left(\frac{Rr}{\sigma^2}\cos\theta\right) \sin^{D-2}(\theta)d\theta \tag{6.11}$$

The derivative of ℓ yields:

$$\frac{d\ell}{dr} = e^{-\frac{r^2}{2\sigma^2}} \int_{\theta=0}^{\pi} \left(\frac{R\cos\theta - r}{\sigma^2}\right) \exp\left(\frac{Rr}{\sigma^2}\cos\theta\right) \sin^{D-2}(\theta)d\theta \tag{6.12}$$

Using integration by parts on the $R\cos\theta$ term, it comes:

$$\int_0^\pi \frac{R\cos\theta}{\sigma^2}\exp\left(\frac{Rr}{\sigma^2}\cos\theta\right)\sin^{D-2}(\theta)d\theta = \underbrace{\left[\frac{R}{(D-1)\sigma^2}\exp\left(\frac{Rr}{\sigma^2}\cos\theta\right)\sin^{D-1}(\theta)\right]_0^\pi}_{=0} + \int_0^\pi \frac{R^2 r}{(D-1)\sigma^4}\exp\left(\frac{Rr}{\sigma^2}\cos\theta\right)\sin^{D}(\theta)d\theta$$

Finally yielding:

$$\frac{d\ell}{dr} = \frac{r}{\sigma^2}e^{-\frac{r^2}{2\sigma^2}}\int_{\theta=0}^\pi \left(\frac{R^2\sin^2\theta}{(D-1)\sigma^2} - 1\right)\exp\left(\frac{Rr}{\sigma^2}\cos\theta\right)\sin^{D-2}(\theta)d\theta \tag{6.13}$$

In particular, if $\frac{R^2}{(D-1)\sigma^2} \leq 1$, the derivative of ℓ as a function of r is always negative, implying that it reaches its maximum for $r = 0$. This means that $p_\theta(\mu)$ is a Dirac-measure at $\mu = 0$, and therefore the predictive function f_θ is the constant function 0. □

Chapter 7
Dynamics of Variance Learning

Contents

This chapter continues the analysis presented in the previous chapter and extending [BS20b], showing that in the quasi-deterministic case the variance of the observation model governs the scale of the data patterns that can be modelled.

The question investigated here concerns how to learn this variance Section 7.1 and its impact on the training dynamics 7.3. Only Gaussian observation models are considered in the chapter for the sake of clarity; the analysis however is general in the sense that it does not rely on the specific structure of the Gaussian model but rather on its quasi-deterministic properties.

7.1 Observation variance fitting

As said (Section 6.3), an observation model with a too high noise variance prevents the model from capturing fine-grained details of the data. Quite the contrary, a too low variance causes the encoding the "natural" data noise into latent variables.

Unless this variance is supplied from prior knowledge, it must thus be learned from the data (and even when it is known beforehand learning it is still beneficial, Section 7.3). Two approaches are found in the literature:

- Considering an isotropic observation model $\mathcal{N}(f_\theta(z), \sigma)$ where σ is optimized along training [RV18] (Section 7.1.1);
- Considering an isotropic $\mathcal{N}(f_\theta(z), \sigma(z)\mathbf{Id})$ or non-isotropic $\mathcal{N}(f_\theta(z), \mathbf{D}(z))$ with $\mathbf{D}$ a diagonal matrix, and the scalar or vector noise variance $\sigma(z)$ being learned by the neural net [KW14; MF18] (section 7.1.2).

7.1.1 Learning a global noise variance σ

In the case where the sought variance σ^2 is constant over the observation space $\mathcal{X} \subset \mathbb{R}^D$, then its optimization can be done analytically. Let $q_\phi(z|x)$ denote the inference model, f_θ the decoder neural network, with D the dimensionality of the observation space. In this case, σ is only involved in the reconstruction part of the ELBO:

$$\mathop{\mathbb{E}}_{x\in\mathcal{D}} \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) = -\mathop{\mathbb{E}}_{x\in\mathcal{D}} \mathop{\mathbb{E}}_{z\sim q_\phi(z|x)} \left[\frac{\|x - f_\theta(z)\|^2}{2\sigma^2} + D\log\sigma\right] \tag{7.1}$$

Let $e^2 = \mathbb{E}_{x\in\mathcal{D}}\mathbb{E}_{z\sim q_\phi(z|x)}\|x - f_\theta(z)\|^2$ be the average squared reconstruction error. The optimal value $\sigma^\star$ is thus reached as:

$$\sigma_\star^2 = \arg\min_{\sigma^2}\left[\frac{e^2}{2\sigma^2} + D\log\sigma\right] \tag{7.2}$$

Simple calculations then give:

$$\sigma_\star^2 = \frac{e^2}{D} \tag{7.3}$$

The optimal variance thus is directly set to the average reconstruction error per coordinate. This result generalizes to non-isotropic observation models, where the noise is defined after a diagonal matrix $\Delta(\sigma_1, \dots, \sigma_D)$: variance σ_i on the i-th coordinate is the average squared reconstruction error of the model on this coordinate[1].

Algorithmically, σ can be computed from the average reconstruction error incurred in the current epoch, and updated for the next epoch. This is proposed with good success in the recent article [RDL21]. In practice however (as said in Section 6.2.1), most articles using a Gaussian observation model consider σ a fixed hyperparameter. Another option is to learn σ by gradient descent like any other parameter, and have it converging to its optimal value. Expectedly, σ gradually decreases along training, as the reconstruction quality improves. This approach appears empirically equivalent to computing the optimal σ at every batch (Figure 7.4). The stability and dynamical implications of this process are discussed in Section 7.2 and Section 7.3.

A third alternative is formulated by the *Generalized ELBO with Constrained Optimization (GECO)* algorithm [RV18], where the search for a VAE proceeds by minimizing the KL divergence in the latent part of the ELBO under the constraint that the average squared reconstruction error be smaller than some prescribed value. This approach is formulated in terms of β-VAE [Hig+17], which is in the Gaussian case equivalent to changing the observation variance, as discussed in Section 6.2.1.

7.1.2 Learning a local noise variance $\sigma(z)$

Most interestingly, the seminal paper on VAE [KW14] advocated the learning of the noise variance as an output of the decoder NN − though this architecture is

[1] Full-rank covariance matrix for the observation model is also analytically solvable and leads to $\mathbf{\Sigma} = \mathbb{E}\left[(x - f_\theta(z))(x - f_\theta(z))^T\right]$, however such models are generally impractical in high-dimensional data spaces as they require large matrix inversions.

also rarely considered in practice. Formally, the decoder neural network yields two vectors of size K, respectively the mean $f_\theta(z)$ and the (diagonal) variance $\sigma^2_\theta(z)$ of the model. As said, the case of a full covariance matrix has been considered for small input dimensions only.

The model is trained by gradient descent to maximize the Gaussian log-probability:

$$\log p_\theta(x|z) = -\sum_i \frac{(x_i - f_\theta(z)_i)^2}{2\sigma^2_\theta(z)_i} - \log \sigma_\theta(z)_i \tag{7.4}$$

This formulation makes it possible to learn different noise amplitudes in different regions of the input space, either reflecting the heteroscedastic noise of the data (as in the artificial example in Chapter 6, Figure 6.6(a)), or reflecting the uncertainty of the model to account for the data in the region. In the latter case, the large variance reflects the fact that the model does not know how to handle this sample.

This gain in expressiveness however comes at a cost in training stability. Depending on the architecture of the underlying neural network and the optimizer, the values of the observation model might vary in a large range, or take extreme values. In such cases the $1/\sigma^2_\theta(z)$ factor occasionally becomes huge[2], severely hindering the training process. To avoid this, the parameterization of the neural network should ensure that $\sigma^2_\theta(z)$ be initialized to a reasonably large value, and evolve slowly enough, avoiding abrupt changes from one iteration to the next. We shall return to the impacts of the training dynamics in Section 7.2 and Section 7.3.

7.1.3 Empirical study

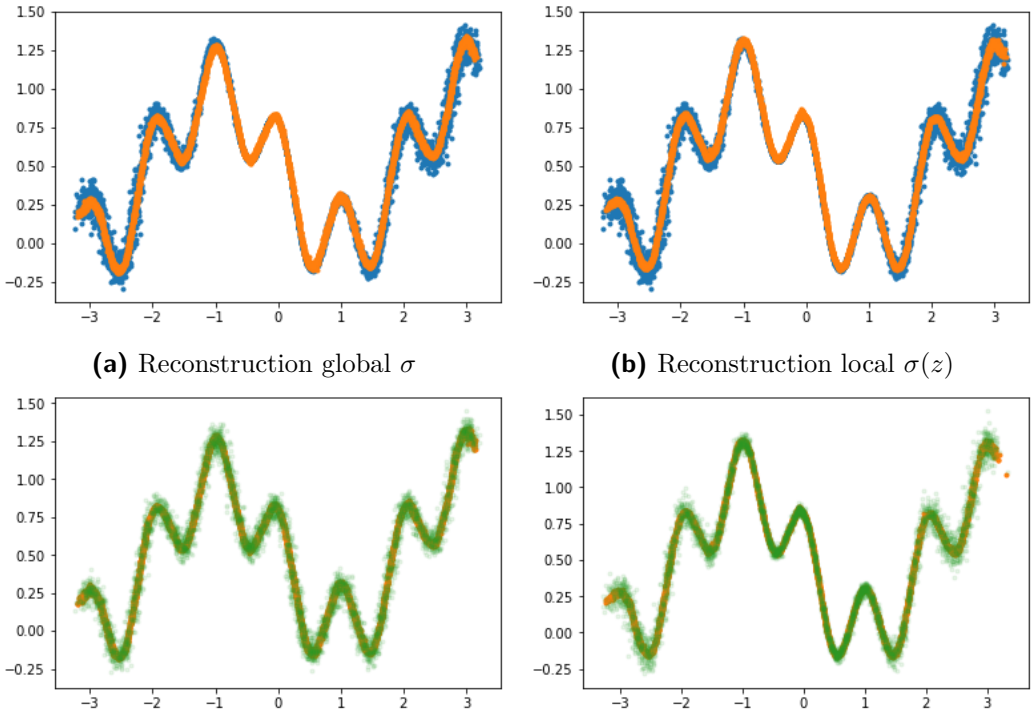
Considering the artificial dataset introduced in the previous chapter, this section presents an empirical comparison of both options of learning a global or local variance (Figure 7.1). The model with a globally optimized value for σ is doomed to find a trade-off, with too low a variance in the high noise regions, and too large a variance elsewhere (Figure 7.1(c)). On the contrary, the model where $\sigma(z)$ is a learned function of z manages to perfectly fit the noise from the dataset (Figure 7.1(d)): the generated samples (in green) display the same dispersion as the original dataset (in blue). The goodness of fit is also visible from the ELBO improvement: the model with learned function $\sigma(z)$ yields 1.62 ± 0.08 while the model with learned global σ only reaches 1.26 ± 0.09.

7.2 The risk of deterministic collapse

Both schemes – learning the variance either directly or as a function of the latent variable – converge toward setting variance σ^2 to the (possibly dimension-wise) reconstruction error $\|x - f_\theta(z)\|^2$. In the former case, σ^2 converges toward the average reconstruction error; in the latter case, $\sigma^2(z)$ converges toward the local reconstruction error.

When considering a quasi-deterministic observation model however, there is a risk of deterministic collapse: if the VAE yields a perfect reconstruction (i.e. $f_\theta(z) = x$

[2] As $\sigma(z)$ must be strictly positive, it is usually sought as the softplus of an expression, and goes to 0 as this expression goes to $-\infty$.



(a) Reconstruction global σ **(b)** Reconstruction local $\sigma(z)$

(c) Generation global σ **(d)** Generation local $\sigma(z)$

Figure 7.1: Results of training the VAE from Section 6.3.2 with σ either learned as a global value or as an output of the decoder. In blue is the original data, in orange the predictions $\hat{x} = f_\theta(z)$, and in green actual samples from p_θ. While both models learn the same approximating manifold (a, b), the second one manages to better fit the heterogeneous noise of the dataset in generation (d) than the first one (c).

everywhere), then the optimal value for σ is 0 (Equation 7.3), and the ELBO goes to $+\infty$. Note that such a perfect reconstruction entails a high cost in terms of $D_{KL}(q_\phi(z|x)\|p_\theta(z))$, since the inference model $q_\phi(z|x)$ is bound to be deterministic as well, making the KL-divergence infinite too[3].

The risk of collapse is explained as follows. In the deterministic limit, both ELBO terms are proportional to the dimensionality of their respective space (data or latent). Typically for a Gaussian VAE, the reconstruction loss is the sum of the squared reconstruction error, and a log-variance term $N \log \sigma_x$. If the reconstruction error is very small, the loss is dominated by the second term, which is proportional to the dimensionality of the input space. The latent KL similarly involves a term which is bounded in the deterministic limit, plus a term of the form $-\sum_i \log \sigma_{z,i}$, with i ranging among the latent coordinates. This latter term is proportional to the number of dimensions where σ_i is significantly smaller than 1 (the variance of $p(z)$). In the deterministic limit, the variances go to 0 and the full behavior of the ELBO is dominated by term $N \log \sigma_x - \sum_i \log \sigma_{z,i}$.

As a result of this limiting behavior, if the number of effectively used dimensions in the latent space is smaller than N, then the ELBO tends to $+\infty$ as all variances converge to 0, resulting in a model converging to a deterministic auto-encoder while

[3]For $q_\phi(z|x)$ a deterministic Dirac distribution, its KL to a non-tereministic prior is $-\infty$.

optimizing its objective. This argument is not specific to the Gaussian case, but relies on the fact as $q_\phi(z|x)$ and $p_\theta(x|z)$ converge to deterministic distributions, $\mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z)$ grows to $+\infty$ faster than $D_{KL}(q_\phi(z|x)\|p(x))$, which is generally the case if the model manages to compress the dataset at least a little.

This risk of collapse, as identified by [RV18; MF18], follows from the fact that training a continuous probabilistic model on a finite dataset is not a well-posed problem, as its optimal solution is a mixture of point masses over the examples. Indeed the ELBO is higher-bounded by the negative entropy of the target distribution. However when considering a continuous observation model, this entropy needs to be computed as a *differential entropy*. When viewed as a continuous distribution, the entropy of a Dirac is $-\infty$, meaning that the differential entropy of any finite dataset is $-\infty$ as well. This results in an optimization problem where the ELBO is not higher-bounded by a finite value, making it ill-defined and entailing the deterministic collapse phenomenon.

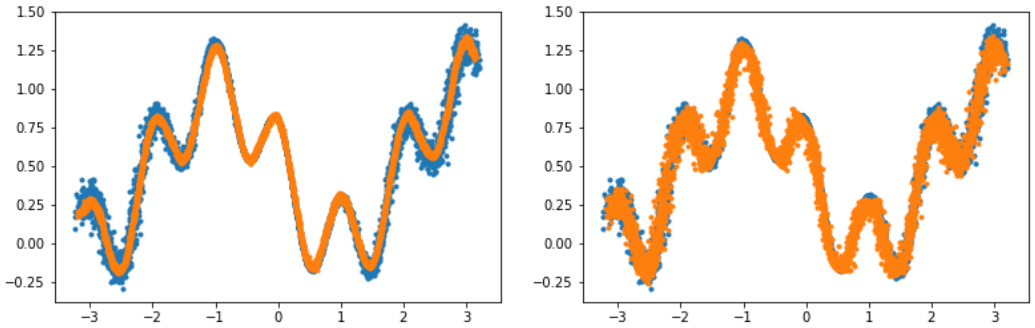
This risk has however to the best of our knowledge remained theoretical: the optimizer dynamics and the limitations of the neural network structure generally prevent the model from reliably reaching a perfect reconstruction, preventing in turn a deterministic collapse. However the model is still driven to decrease the observation variance as much as possible to increase its ELBO, as long as its capacity allows it.

As was analyzed in Section 6.3.2, the variance the model settles on determines the resolution at which it observes the dataset and the amount of detail that is learned in the latent space. It effectively defines the limit between signal and noise, and on a finite dataset the model is driven to consider as much information as possible as signal, as long as it can be compressed at least a little. This is in general counterproductive: as long as the model is expressive enough, it'll be able to learn and compress any noise on a finite dataset [Zha+17]. In such a dynamic, part of the random noise in the dataset would be interpreted as small but significant signal and stored in the latent space. This could be interpreted as some form of overfitting.

In some situations, knowledge about the dataset allows to know in advance a lower-bound for any reasonable value of this observation variance: the uncertainty of the measurement apparatus that produced the data could be known, or smaller details are deemed irrelevant for the task at hand (for example, details that are invisible to the human eye can be irrelevant in a task of image generation). In such cases it seems natural to ensure that the model will not converge to a variance that is smaller than this lower bound, to avoid putting unnecessary pressure on the latent encoding.

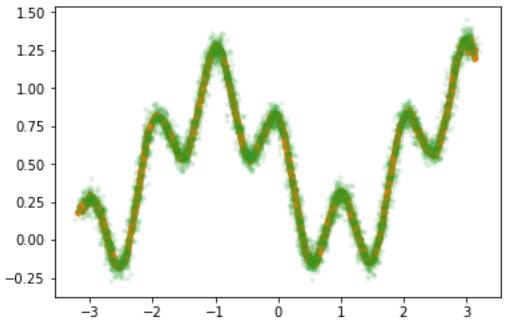
We propose here two simple methods to achieve that. The first method is to parameterize the learning process of the variance such that it cannot decrease below it[4], effectively making this part of the optimization space unreachable for the model. The second method is to augment each training batch with a Gaussian noise with variance equal to said lower-bound. This noise will not be compressible by the model (as it is a real, infinite noise), preventing it from storing it in the latent space. That added noise can also have a regularization behavior on the inference model.

[4]For example, noting σ_{min} the lower-bound, the predicted σ can be passed to a threshold function such as $\sigma \to max(\sigma, \sigma_{min})$ or $\sigma \to \sigma_{min} + softplus(\sigma)$.

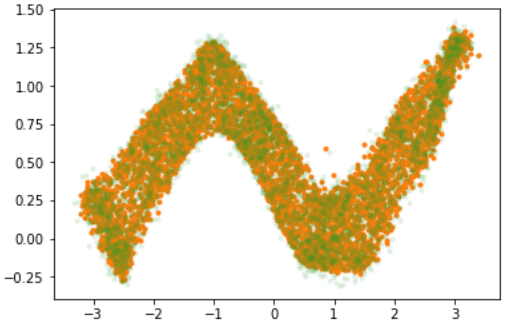


(a) Reconstruction learned global σ

(b) Reconstruction $\sigma = 0.043$



(c) Generation learned global σ



(d) Generation $\sigma = 0.043$

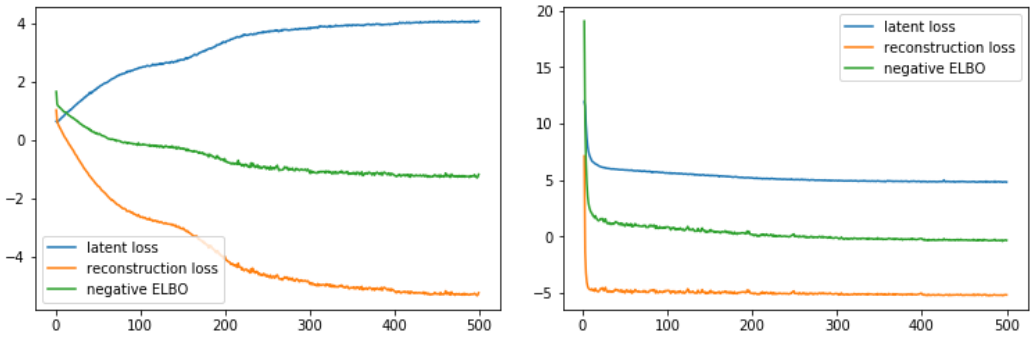
Figure 7.2: Comparing VAE with trained and fixed variance on the artificial problem (Section 6.3.2). (a): Reconstruction of initial samples with learned σ. (b): Reconstruction of initial samples with $\sigma = .043$. (c): Generated samples with learned σ. (d): Generated samples with $\sigma = .043$. Original samples are in blue; Predictions *without observation noise* are in orange; Generated samples are in green. While the model with a learned σ correctly characterizes the dataset, the model with a fixed σ value fails to separate the noise from the underlying manifold (the noise is encoded in the latent space, as seen on (b), resulting in a terrible generation performance as shown in (d).

7.3 The dynamics of variance learning as an annealing process

Chapter 6 illustrates the dramatic impact of the observation variance on the approximation of the data manifold, governing the latent representation.

This impact is investigated in more depth, focusing on the case where the variance is learned. A lesion study is conducted, by comparing the observation model – where the scalar σ converges toward $\sigma = 0.043$ – with the eventual observation model obtained when setting σ to this same value. Unexpectedly, the latter model (fixing $\sigma = 0.043$) fails to correctly characterize the data manifold, as illustrated in Figure 7.2.

The fact that the variance is learned is conjectured to significantly modify the learning dynamics. More specifically, our tentative interpretation is as follows: initializing σ to a large value and letting the training process learn it plays a regularization role comparable to an annealing process, akin KL-annealing [Luc+19].



(a) Losses for learned global σ.

(b) Losses for fixed $\sigma = 0.043$.

Figure 7.3: Trajectories of the reconstruction loss (orange), latent KL loss (blue) and negative ELBO (green) for the model with learned global σ (Fig. 7.3(a)) and fixed $\sigma = 0.043$ (Fig. 7.3(b)). The first model has a very gradual learning trajectory, while the second starts with a high reconstruction loss and converges very quickly toward a local optimum.

Formally, starting with a large σ causes the model to first learn a most simple representation of the data, as discussed in Section 6.3.2. This simplistic representation is very easy to efficiently compress into the latent space, and the model promptly learns an efficient representation. This representation allows the reconstruction quality to be decent enough to afford decreasing σ; the smaller σ in turn enables the model to be aware of smaller details, that are gradually accounted for in the latent representation.

In contrast, freezing σ to a small value puts a large initial weight on the reconstruction part of the ELBO (the $1/\sigma^2$ term dominates), making the model minimizing its reconstruction loss *at all cost*. The optimization, thus entirely driven by local reconstruction quality, fails to establish a global coherent structure on the latent space. As a result, the model remains stuck in a local minima with a poorly organized latent representation.

This interpretation is further assessed by comparing the loss trajectories (Figure 7.3). The training dynamics significantly differ in both cases of learned versus fixed σ, as follows: when σ is learned, the data information is gradually accounted for in the latent space, causing the latent loss (Equation 3.3) to gradually increase while the reconstruction loss gradually decreases, resulting in a gradual reduction of the negative ELBO. On the opposite, the model with fixed $\sigma = 0.043$ suffers a very high reconstruction loss from the start; it quickly reduces it and quickly converges, hardly improving its loss past the first 5 epochs. Note that the gradual dynamics of the losses when σ is learned strongly correlates with the evolution of the value of σ itself, as illustrated on Figure 7.4.

As said, this gradual decrease of σ, viewed as an annealing process, seems to benefit to the shaping of the latent space of a quasi-deterministic observation model. By gradually increasing the amount of detail available to the model, it allows an incremental refinement of the latent representation, which in turn allows the model to converge to a more efficient latent space. The simplest way to achieve this annealing is by having σ be initialized to a large value (of the order of the total variance of the training data), and letting the model learn its value gradually, as discussed in

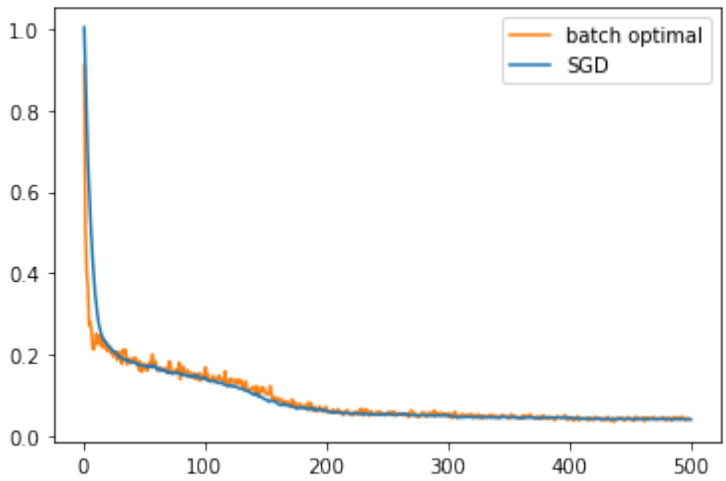


Figure 7.4: Evolution of the learned value for σ during the training procedure. Blue: σ is learned through SGD. Orange: σ is fixed to its optimal value (Equation 7.3) at each minibatch. Both schemes are empirically equivalent.

Section 7.1.

How the final value of the ELBO depends on the fixed value of σ versus learned σ is illustrated on Figure 7.5. 50 independent runs with fixed σ log-uniformly drawn in interval $[0.01; 1]$ have been done. The eventual optimal ELBO shows a bell curve w.r.t. σ, reaching its optimum for $\sigma \approx .1$ (in blue, Figure 7.5). Likewise, 10 runs have been launched to optimize the ELBO with a global learned σ, showing that: i) the variance of the eventual ELBO value is very low (all runs converge toward very close optimal σ); this optimal σ value (.043) is significantly lower than the optimal value for the runs with fixed σ (in red, on Fig. Figure 7.5). Last, 10 runs have been launched with σ learned as a function of the latent variable z, and for each run, the range of these optimal σ is depicted as a segment (in green on Fig. Figure 7.5). To reflect the prediction of multiple different σ values by these last models, the cumulative distribution of these prediction for one such model is presented in Figure 7.6.

7.4 Summary and perspectives

This chapter presents two results concerning the variance of the observation model in the quasi-deterministic case.

Firstly, for a given predictor function $f_\theta(\mathbf{z})$ and inference model q_ϕ, the optimal value of the variance (w.r.t. the ELBO optimization) is the average squared error of the VAE reconstruction[5]. The associated risk of deterministic collapse is analyzed. While this risk is hardly met in practice, it might lead to retain a variance lower than would be appropriate. Two heuristics counter-acting this effect are discussed.

Secondly, a proof of concept is presented, suggesting that *learning the variance* might have beneficial effects beyond finding the appropriate value. Most interestingly, enabling the VAE to learn the variance allows the model to gradually build and

[5]More precisely, the global average if σ is globally optimized by gradient descent; the local average in the image of the latent region if σ is learned as a function of $\mathbf{z}$.

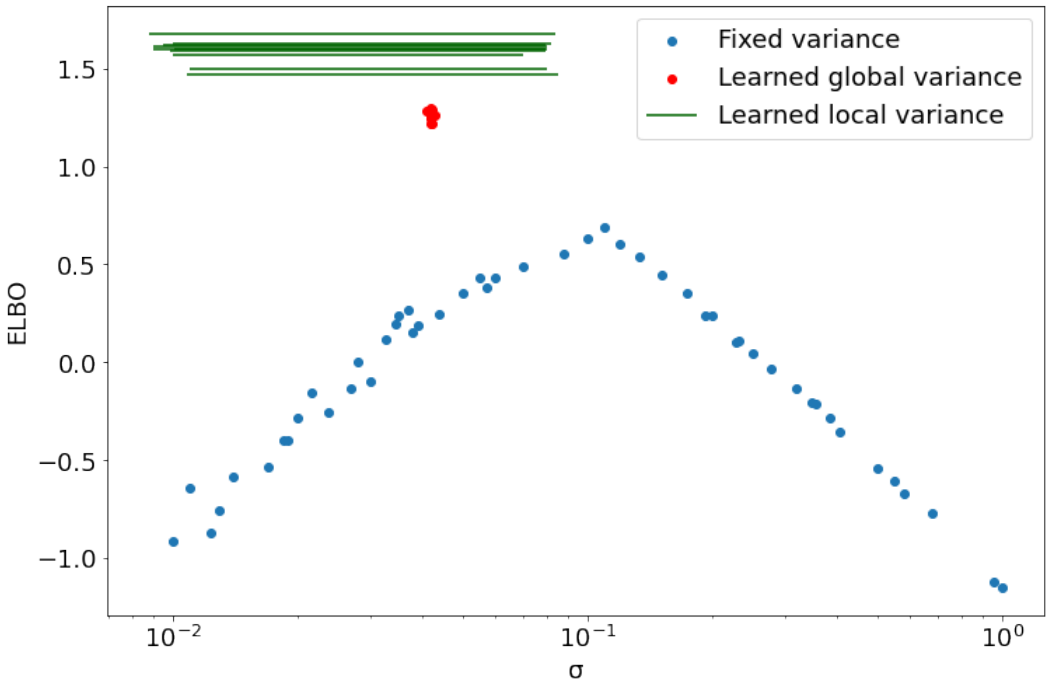


Figure 7.5: Representation of the ELBO value reached by the model as a function of the observation model variance σ^2. Each dot corresponds to one training instance. In blue are models whose variance was fixed at the beginning of the training. In red, models where the variance is learned as a global parameter, the reported variance being the final learned value. In green are represented models where the variance is part of the output of the decoder (and thus depends on the latent variable); each is represented by an horizontal line over the range of values produced by the model.

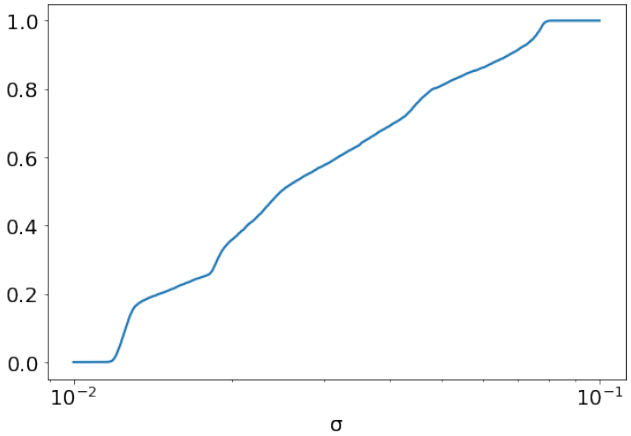


Figure 7.6: Cumulative distribution of the predicted $\sigma(z)$ across the generative distribution $p(z)$ (horizontal axis is in logarithmic scale). A large portion of the $[0.01; 0.08]$ interval is used by the model, reflecting its learning of the heterogeneous noise in the dataset.

compress the latent representation, increasing the amount of detail stored in the latent space, until reaching the point where all remaining information would be more expensive to store in the latent space than the gain provided in terms of reconstruction quality. Overall, this learning dynamics is viewed as an annealing process, adequately separating the noise from the information to the extent permitted by the model capacity.

A perspective for further study thus consists in inspecting how the latent space forms its structure, and how it manages (or fails) to reflect the topological structure of the dataset. Our contribution, showing the impact of the training dynamics, suggests to investigate the adaptive balancing of the reconstruction and the compression efforts.

7.A Observation tempering and link with β-VAE

Chapter 6 detailed the equivalence between setting a fixed variance to a Gaussian observation model and the introduction of a weighting factor between the "reconstruction" and "latent" parts of the ELBO. While this equivalence is no longer true when the variance is a learned parameter[6], in essence the comparison is still applicable, this section will explicit how.

The β-VAE [Hig+17] can be compared to defining the global generative model as:

$$p_\theta(x, z) \;\propto\; p_\theta(x|z)^{1/\beta} p_\theta(z) \tag{7.5}$$

By analogy with the tempered posteriors of Bayesian inference, we will refer to this as *observation tempering*. It is important to note that this observation tempering is not the same as the β-VAE, as the later does not take into account the normalization factor of the modified observation model (which depends on z, θ and β). This is a situation similar to the one analyzed by [LC19]. After studying the impact of observation tempering, we will explicit how it can be compared to the proper β-VAE.

Tempering the observation model with $\beta > 1$ makes it blurrier, converging to a uniform distribution in the limit $\beta \to \infty$. It is to be expected that the analysis of Section 6.3 will overall still apply here. As β increases, the model becomes more and more blind to fine details and highly-curved regions of the data manifold. On the opposite, tempering the observation model with $\beta < 1$ makes it sharper than it normally is, with all the opposite effects as illustrated on Figure 7.7. If the observation model is allowed to learn some scale parameter, it is to be expected that it will converge to a different value of it, compensating the tempering to some extend[7].

The relation of β-VAE to *observation tempering* is that the former does not take into account the renormalization. Let $\hat{p}_\beta$ represent the tempered version of some base distribution p. It is defined by the following identity in log-probabilities, including the normalization term:

$$\log \hat{p}_\beta(x) = \frac{1}{\beta} \log p(x) - \log \int p(x)^{1/\beta} dx \tag{7.6}$$

Now, when considering the gradient of the distribution relative to some value λ (which can be either a model parameter or the value of another variable of the model), the normalization factor gives rise to a non-zero gradient term:

$$\nabla_\lambda \log \hat{p}_\beta(x) = \frac{1}{\beta} \left[\nabla_\lambda \log p(x) - \mathop{\mathbb{E}}_{\hat{x} \sim \hat{p}_\beta} \nabla_\lambda \log p(\hat{x}) \right] \tag{7.7}$$

[6] For example, the Gaussian observation includes an *additive* $\log \sigma$ term to the loss, breaking the equivalence.

[7] The tempered observation model is in general not in the same distribution family as the original model, as a result it cannot be expected that the model would exactly compensate it. Notable exceptions are exponential family distributions (such as categorical, Gaussian or Beta for example), for which the tempering operation does preserve the class, and amounts to a simple reparametrization which learning can in principle compensate. It will however still affect training dynamics.

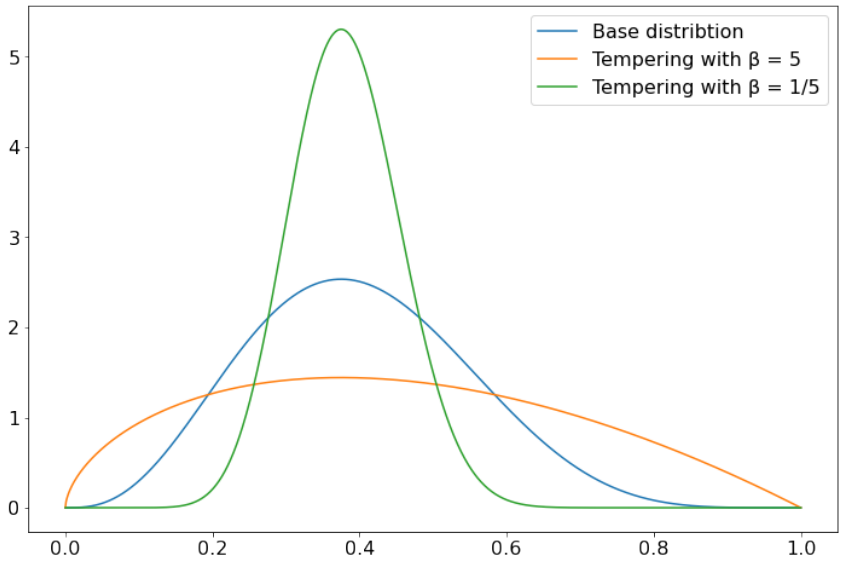


Figure 7.7: Illustration of the effect of tempering on a distribution. The base distribution (blue) is significantly broadened when tempered with a large β (orange, $\beta = 5$), and concentrated when tempered with a small β (green, $\beta = 1/5$).

Isolating the gradient as computed by β-VAE:

$$\underbrace{\frac{1}{\beta}\nabla_\lambda \log p(x)}_{\beta\text{-VAE gradient}} = \underbrace{\nabla_\lambda \log \hat{p}_\beta(x)}_{\text{tempered observation gradient}} + \frac{1}{\beta} \mathop{\mathbb{E}}_{\hat{x}\sim\hat{p}_\beta} \nabla_\lambda \log p(\hat{x}) \tag{7.8}$$

The gradient guiding β-VAE can thus be seen as an approximation of the gradient guiding the tempered observation with an additional term. The analysis of this term will explicit the relation between the two formulations.

The second term of Equation 7.8 is very similar to a known identity[8], which is recovered when $\beta = 1$:

$$\mathop{\mathbb{E}}_{x\sim p} \nabla_\lambda \log p(x) = 0 \tag{7.9}$$

As β is changed from 1, the gradients in the expectation are reweighted relative to each other, changing the result. In particular, when $\beta > 1$, the tempered distribution $\hat{p}_\beta$ is more spread-out than the original distribution p. This affects the expectation such that gradient terms in low-probability regions are over-represented, while gradient terms in high-probability regions are under-represented. With each point-wise gradient directing the distribution towards increasing the probability at this point, they thus combine into a global gradient that tends to increase the spread of the distribution further. A symmetric reasoning can be done when $\beta < 1$, in which case $\hat{p}_\beta$ is more concentrated than p, and the resulting gradient drives the distribution overall towards being even more concentrated.

Finally, Equation 7.8 can be read as follows: the gradient guiding the training in β-VAE is similar to the gradient guiding a tempered observation with the same β, with an additional term that accentuates the effect of β (spreading the distribution

[8]Previously proved in Equation 4.5.

when $\beta > 1$ and concentrating it when $\beta < 1$). This second terms thus prevents the model from effectively compensating the tempering of the observation model by learning different parameters.

As an illustrative example, consider a single-dimensional Gaussian observation of learned variance σ^2. In this case, the additional term can be computed analytically:

$$\frac{1}{\beta}\nabla_\lambda \log p(x) = \nabla_\lambda \log \hat{p}_\beta(x) + \frac{\beta - 1}{\beta}\nabla_\lambda \log \sigma \tag{7.10}$$

In this case, the β-VAE formulation drives the model towards increasing the chosen σ^2 if $\beta > 1$, compared to the usual Gaussian observation[9].

The general interpretation thus holds overall: the β-VAE alters the convergence of the observation model in the training dynamics. When $\beta > 1$, it forces the model to use a smoother observation model than the natural ELBO would converge to, with the consequences explored in Section 6.3. In particular, the higher the β, the more the model is blind to details of the data manifold. This results in less information encoded in the latent variables, and blurrier reconstructions and generations in the context of images. All of this is in accordance with the empirical results observed by [Hig+17].

[9] $\hat{p}_\beta(x)$ is just a reparametrization $\sigma^2 \to \beta\sigma^2$ for which the learning process can compensate exactly.

Part III
Properties of latent structures

Chapters

Chapter 8
Properties-oriented structures

Contents

While the previous chapters focus on the design of deep LVMs, aimed to reflect the data via the observation model, this chapter focuses on the design of Deep LVMs and their latent structure, to enforce model properties guided by either epistemic considerations about the data, or the intended use of the model.

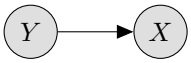
After describing how such properties have been enforced in the literature, either by integrating a deep LVM into a larger model (Section 8.1 & 8.2), or by carefully designing its latent structure (combining probabilistic and deterministic variables, Section 8.3), this chapter describes our contribution related to anomaly detection in the context of the *Compact Muon Solenoid (CMS)* experiment at CERN [Pol+19]. A conditional deep LVM is there used to detect and distinguish two different kinds of anomalies, by leveraging its latent structure (Section 8.4).

8.1 Generative classifiers for robustness

This section focuses on the use of generative models for classification.

The dominant classification approach is discriminative, that is, the ML model is trained to directly approximate $P(Y|X = x)$, the conditional distribution of the class label Y given the description x of an example (Figure 8.1(a)). This approach faces a major challenge: the quality of the learned approximation for a given example x dramatically depends on how close this example is to the training dataset, with little means to evaluate the quality of said approximation *a priori* [Sno+19; Nal+19b].

(a) Discriminative classifier



(b) Generative classifier

Figure 8.1: Graphical representation of a discriminative classifier (a) and a generative one (b).

This challenge is at the heart of the so-called adversarial examples, which are a longstanding research question for deep learning based classification [GSS15; KGB17; CW17; Ath+18; Ily+19].

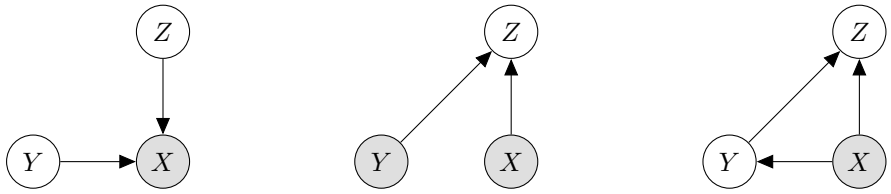
An alternative approach, often referred to as *generative classifiers* or *Bayes classifiers*, consists in reverting the graphical model (Figure 8.1(b)) and instead learning a conditional generative model of the data given the class label $P(X = x|Y = y)$, noted $p(x|y)$ for simplicity.

The actual classification is then achieved by selecting the y value making x most likely. On this two-node model, this amounts to applying Bayes' Theorem: $p(y|x) \propto p(x|y)p(y)$ using the conditional generative model and a prior $p(y)$ on the class labels. Under the assumption of independence of X features, this yields the well known *Naive Bayes classifier*, delivering in practice surprisingly good results (despite this assumption to generally be unrealistic) [DP97; HY01]. The general Bayes principle can also be applied to more complex generative models including latent variables. For instance, in [Sch+18] one VAE per class is trained, and then a tight ELBO estimation of $\log p(x|y)$ is computed at test time.

The generative approach is significantly more computationally expensive than the discriminative one, as it requires the computation of a conditional query on a graphical model. Its main merit is its *robustness* property. The prediction of a generative classifier combines a prior (where the prior $p(y)$ can be simply estimated from the class frequencies in the training data) $p(y)$ and the generative likelihood for each class $p(x|y)$. Its behavior can thus be thought of in terms of mixture models, making it less prone to overfitting than discriminative classifiers. Indeed, examples far from the training set are expected to yield low generative probabilities $p(x|y)$ for all classes, resulting in a final prediction somewhat close to the prior. Furthermore, this setting makes room for evaluating the probability of the sample $p(x)$, that can be used to estimate how close this example is to the training dataset. This makes it possible to discard examples with a very low probability, enabling the classifier to abstain instead of reporting a low confidence classification.

These two properties of resistance to overfitting and identification of untypical examples have been experimentally and thoroughly explored in [LBS19]. This article compares several structures of generative and discriminative classifiers against state of the art adversarial attacks. The effectiveness of different methods to detect such adversarial attacks are also compared. Their conclusions are that generative classifiers are indeed less prone to making confident but wrong predictions, and that even when they do, the evaluation of $p(x)$ is a robust way to detect out-of-distribution examples.

This last issue, the detection of out-of-distribution examples, is still debated in the state of the art. The simplest method of detecting out-of-distribution samples, thresholding $p(x)$, has been empirically shown to be unreliable in high-dimensional data spaces: out-of-distribution samples can have a higher probability assigned by the model than actual samples from the training dataset. Indeed the curse of



(a) Generative model **(b)** Labeled inference model **(c)** Unlabeled inference model

Figure 8.2: The generative semi-supervised model combines a generative and an inference model [Kin+14]

dimensionality makes the probability density a poor indicator of whether a given sample does lie in the *typical set* of a distribution. This phenomenon and its implications are notably analyzed by [Nal+19b; CJA19; Nal+19a]. We shall return to this point in Section 8.4.

How to use generative classifiers for detecting out-of-distribution samples has been further examined within the *Linear Discriminant Analysis (LDA)* framework: by introducing a Gaussian assumption on the form of the generative models $p(x|y)$, the application of Bayes' Theorem is reframed as finding the class that minimizes a distance in a feature space previously learned by a discriminative classifier, improving the prediction robustness [Lee+18]. Notably the introduction of a learned covariance matrix for the Gaussian model allows to geometrically reshape the latent space, improving the quality of the used distance function. In this framework, [PDZ18] further proposes to directly learn the feature space in which LDA is performed, rather than rely on a previously trained one.

8.2 Semi-supervised learning with VAEs

This section focuses on the combined used of labeled and unlabed data, to support a more robust classification approach.

Semi-supervised learning (SSL) encompasses diverse algorithms, able to harness a (supposedly large) unlabeled dataset to improve the predictive capacity of a model trained on a smaller labeled dataset. SSL relies on the assumption that unlabeled data yield meaningful information about the domain structure, such as the presence of natural clusters, which can be used for the prediction task. Many such methods consist in implicitly labeling the unlabeled samples, for example by propagating labels from the labeled ones using geometrical properties.

The so-called *Generative semi-supervised model (M2)* procedure, proposed in [Kin+14], consists in jointly training a classifier and the inference model of a Deep LVM, that is itself trained on both the labeled and unlabeled dataset.

Consider the 3-node LVM represented in Figure 8.2(a), where the X node represents the data, Y its label, and Z an abstract latent variable. This model, factored as $p_\theta(x, y, z) = p_\theta(x|z, y)p_\theta(y)p_\theta(z)$, is trained simultaneously on labeled and unlabeled data using two different inference models.

On labeled dataset $\mathcal{D}$, the model is trained using Y and X as observed variables, combined with an inference model $q_\phi(z|x, y)$, represented as Figure 8.2(b). Applying the ELBO on this model yields the following training objective:

$$\mathcal{L}_{\text{labeled}} = \sum_{(x,y)\in\mathcal{D}} \left[\log p_\theta(y) + \mathop{\mathbb{E}}_{z\sim q_\phi(z|x,y)} \log \frac{p_\theta(x|z,y)p_\theta(z)}{q_\phi(z|x,y)} \right] \tag{8.1}$$

The above loss, decoupling $p_\theta(y)$ from the rest of the generative model, makes it to reflect the probabilities of the different classes in the dataset. Independently, the other two parts of the model $p_\theta(x|z,y)$ and $p_\theta(z)$ are trained jointly as a conditional VAE to reflect the distribution of X given Y.

When considering the unlabeled dataset $\mathcal{U}$, the inference model is augmented with a component $q_\phi(y|x)$ predicting y, yielding the model presented in Figure 8.2(c). The associated ELBO training objective is thus:

$$\mathcal{L}_{\text{unlabeled}} = \sum_{x\in\mathcal{U}} \mathop{\mathbb{E}}_{y\sim q_\phi(y|x)} \left[\log \frac{p_\theta(y)}{q_\phi(y|x)} + \mathop{\mathbb{E}}_{z\sim q_\phi(z|x,y)} \log \frac{p_\theta(x|z,y)p_\theta(z)}{q_\phi(z|x,y)} \right] \tag{8.2}$$

Note that the expectation over Y cannot be evaluated using the reparametrization trick as Y is a discrete variable. It is thus computed exactly, evaluating the ELBO of the conditional generative model for each possible value of the label Y, and weighting these values according to $q_\phi(y|x)$.

Lastly, the $q_\phi(y|x)$ part of the inference model acts as a probabilistic classifier. This most interesting part, trained (after Equation 8.2) to approximate[1] $p_\theta(y|x)$, can be seen as a generative classifier based on the Deep LVM p_θ. The whole approach thus builds a generative model consistent with both the labeled dataset $\mathcal{D}$ and the unlabeled dataset $\mathcal{U}$, by maximizing the joint objective $\mathcal{L}_{\text{labeled}} + \mathcal{L}_{\text{unlabeled}}$, and delivers $q_\phi(y|x)$, a good approximation of the generative classifier defined by this model.

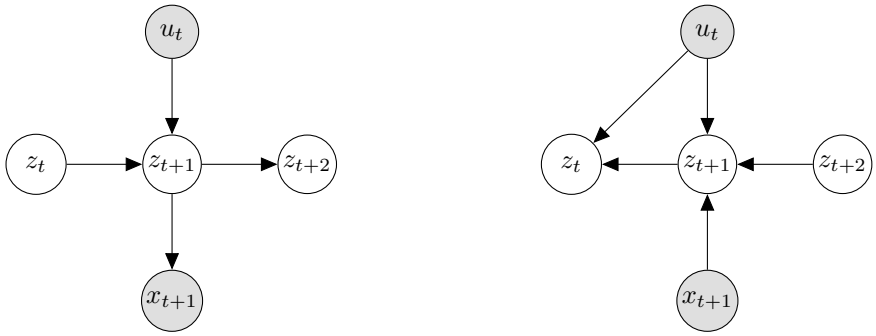
Moreover, one can also directly train $q_\phi(y|x)$ as a regular classifier on the labeled dataset [Kin+14], introducing a third term in the overall loss with hyperparameter weight α, and considering the full training objective $\mathcal{L}_{\text{labeled}} + \mathcal{L}_{\text{unlabeled}} + \alpha\, \mathcal{L}_{\text{classifier}}$.

8.3 Combining probabilistic and deterministic latent variables

This section is interested in the modeling of structured data, focusing on the particular case of temporal data and the identification of the underlying dynamics of the considered system. The claim is that such models are more effectively identified by combining probabilistic and deterministic variables.

As will be shown in this section, a finer control of the latent representation can be obtained through introducing deterministic variables in the graphical model. Such variables play a very different role compared to the probabilistic ones: they do not appear in any density distribution and they only affect the training objective indirectly, by guiding the information flow through the model, as illustrated by the Deep Variational Bayes Filter [Kar+17].

[1] Optimizing the ELBO drives $q_\phi(z|x,y)q_\phi(y|x)$ to be a good approximation of $p_\theta(z,y|x)$, which in turns implies that $q_\phi(y|x)$ is trained to be a good approximation of $p_\theta(y|x)$.



(a) Generative model **(b)** Inference model

Figure 8.3: Intuitive generative and inference models for sequence modeling.

The model, aimed to learn the behavior of a controlled dynamical system, is trained from a sequence (x_t, u_t), with x_t the observed state of (resp. u_t the control applied on) the system at time step t. For instance, x_t might stand for the location of the considered vehicle, and u_t for the amplitude of acceleration and steering angle. A natural modeling approach is to introduce a sequence of latent variables z_t akin a hidden Markov model (Figure 8.3). However, [Kar+17] notes that this model structure puts the stress on accurately reconstructing x_t from z_t (in the auto-encoding step of ELBO training); but empirically, it fails to accurately capturing the desired dynamics of the z_t sequence.

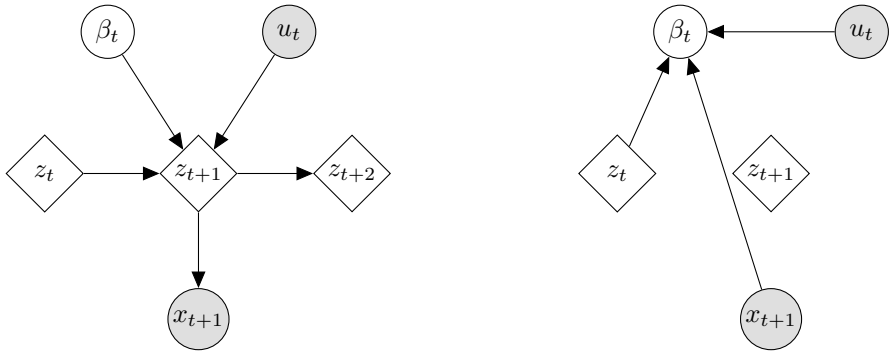
8.3.1 Failure of the fully probabilistic approach

The overall approach, as depicted in Figure 8.3, assumes the Markovian property of the z_t sequence: z_t is supposed to yield all necessary information for predicting x_t, as well as, together with u_t, predicting z_{t+1}. While [Kar+17] report that this is not the case, they offer no tentative interpretation for this failure. This failure is unexpected as one might think that minimizing the gap between the prediction of the inference and generative models corresponds to an optimal ELBO.

Let us suggest a tentative interpretation for this failure, based on analyzing the training dynamics along the same lines as in Section 7.3.

In the early training stage, the optimization process is generally mostly driven by the reconstruction term of the loss (as discussed in Section 7.3), that drives $p_\theta(x_t|z_t)$ and $q_\phi(z_t|x_t, z_{t+1}, u_t, u_{t+1})$ along an auto-encoding scheme. This causes the model to store into z_t all information needed to reconstruct x_t on a per-timestep basis. In a further stage, when the reconstruction loss is decent enough, the reconstruction term of the loss no longer dominates, and the optimization process should ideally proceed to compress the latent representation, taking advantage of the Markov relation between the z_t variables.

However, at this stage, z_t is unlikely to contain the necessary information for predicting z_{t+1}, as it has mostly been optimized for predicting x_t. If the system involves some latent information required for (short or long term) prediction, that is not required for instant prediction (of x_t), then p_θ does not have access to this relevant information, preventing the prediction of z_{t+1} given z_t and u_t. In essence, the model is stuck in a local minimum where it misses the temporal relation between



(a) Generative model **(b)** Inference model

Figure 8.4: Generative and inference models for the Deep Variational Bayes Filter. Diamond-shaped nodes represent deterministic variables.

the variables. In the exemple of the vehicle dynamics, x_t contains the location of the vehicle; but in order to predict its dynamics, one needs z_t to encode both its location and its momentum. If the latter information is missing, one cannot predict z_{t+1} from z_t.

8.3.2 Deep Variational Bayes Filter

The above issue can be avoided by modifying the structure of the generative model, and making the latent variables z_t deterministic, as proposed by [Kar+17], with

$$z_{t+1} = f_\theta^{(t)}(z_t, u_t, \beta_t)$$

where β_t is a latent variable accounting for the stochasticity of the overall dynamics (Figure 8.4). The variable z_{t+1} is the only input of the decoder $p_\theta(x_{t+1}|z_{t+1})$, thus to achieve good prediction of x_{t+1} during training, z_{t+1} must contain all the necessary information. In turn, z_{t+1} being a deterministic variable, the inference model q_ϕ does not directly predicts its value, the information pressure thus flows up to the parents of z_{t+1}: the triplet (z_t, u_t, β_t) must contain the necessary information to predict z_{t+1} (and thus x_{t+1}).

Note that this new formalization radically changes the training dynamics: the z_t now being deterministic variables, the ELBO no longer penalizes storing information into them (as opposed to β_t). The training objective thus enables storing as much information in z_t as needed to predict z_{t+1}, ensuring a good modeling of the system dynamics[2]. The good experimental results confirm the merits of the approach, combining probabilistic and deterministic latent variables: within this formalization, [Kar+17] accurately model the dynamics of simple physical systems, such as a pendulum or a ball bouncing in a box, and can predict dynamics of the system on timelines significantly longer than the length of training sequences. However to the best of our knowledge, evaluation of this approach on more complex problems

[2]To enforce the interpretability of the latent representation, [Kar+17] further requires $f_\theta^{(t)}$ to be a mixture of linear functions of z_t, u_t and β_t. Formally, the matrices encoding these linear functions are globally learned as function of t, and the mixture parameters are learned functions of (z_t, u_t) (but not directly of t).

remains to be done: [Kar+17] heavily rely on the fact that the dynamical equation of their test systems can be represented linearly in the latent space.

This experiment illustrates how a change of the latent structure in the generative model can have a dramatic impact on what can be learned. Using deterministic variables as the backbone of the recurrent structure rather than a Markovian construct allows the gradient information to flow much more efficiently through the model, thus avoiding local optima where the temporal relation is not learned.

8.4 Typed anomaly detection

As discussed in Section 8.1, LVMs can to some extent be used to detect out-of-distribution samples. I applied this property to achieve anomaly detection in the domain of particle physics, in collaboration with Adrian Alan Pol [Pol+19]. The context of application is the Compact Muon Solenoid (CMS), a large detector on the Large Hadron Collider (LHC) at CERN, tasked with detecting particles from high-energy physics experiments. The CMS notably took part in the confirmation of existence of the Higgs Boson.

The CMS trigger system aims to prune the raw data stream from the particle detectors to manageable amounts by filtering out non-interesting events. It is made of a number of rules that can be evaluated online and triggered depending on the content of the events; it governs whether these events will be retained for further analysis. The experiment produces around 40 million events per second, which is much too large an amount of data for exhaustive processing. A first level of fast hardware-implemented filters (named *L1 triggers*) reduces this to 100 thousand events per second. Then a second level of software triggers (named *High level trigger (HLT)*) further reduces this to 1000 events per second. Each HLT processes the events selected by a pre-defined subset of L1 triggers, making a hierarchical relation between L1 and HLT.

The CMS system is hierarchical, with a second layer of (more complex) trigger rules exploiting the events of the first (simpler) ones.

The applicative goal is to detect anomalies in the measurement data streams to identify hardware or software failures in the CMS trigger system. Failures can expectedly be detected by observing unusual rates of acceptance in the triggers [Pol20].

This anomaly detection task has two specific features: i) two kinds of anomalies need to be distinguished; ii) the anomalous status of a datapoint is context-dependent.

These features can be efficiently accounted for in a specific deep LVM, as will be detailed below.

8.4.1 The two kinds of anomalies

In the following, each datapoint is a vector of numerical values. In the context of the CMS data, each of these values correspond to the trigger rate of a single filter during a physical experiment. As the triggers are related by a hierarchical relation, depending on where the fault occurs (at the level of a L1 trigger, or at the level of a HLT) the final vector of trigger rates is affected differently.

The first kind of anomaly, referred to as **type A** anomaly, manifests itself as a large change of value in a single feature of the vector. The second kind of anomaly, referred to as **type B** anomaly, manifests itself as a small but correlated change over a group of features. Both types of anomaly can be emulated in an LVM in a natural way, via perturbing different variables in the generative process. A **type A** anomaly corresponds to a perturbation of a single observed variable, while the generative process is untouched as a whole. A **type B** anomaly can be thought of as perturbing a single latent variable, with cascading consequences, leading to a host of small correlated changes at the observation level. In relation to the CMS data, **Type A** anomalies are anomalies related to a fault in a single HLT, while **Type B** anomalies are linked to a fault at the L1 level, cascading over to all HLT feeding on this particular L1 trigger.

More formally, the behavior of the LVM on anomalous observations differs depending on the kind of anomaly.

Type A anomalies are generally errors that cannot be modeled by the generative model, as they do not fit the generative process of the training data at all. It is thus expected that the auto-encoding process would fail to account for these anomalies. As a result, the reconstruction term of the ELBO, $\mathbb{E}_{z\sim q_\phi} \log p_\theta(x|z)$, would have a value much lower than for regular data.

On the other hand, **type B** anomalies would somewhat align with the generative process and the inference model might reflect **type B** anomalies through slightly modifying the latent variable values according to the existing generative process ($p_\theta(x|z)$). In this case, the reconstruction term of the ELBO would have a usual amplitude, but the inferred values for the latent variables would cause a very low probability for the latent term $\mathbb{E}_{z\sim q_\phi} \log p_\theta(z)$.

In a nutshell, the general intuition is that when evaluating the ELBO on an anomalous datapoint, at least one of the variables of the model will take an unexpected value. Depending on which variable does, one can identify the kind of anomaly presented to the LVM. Furthermore the (usual) dimensionality reduction from X to Z alleviates the curse of dimensionality encountered when assessing whether the variable is out-of-distribution.

In [Pol+19] we focus on a model with a single latent variable Z and a single observed variable X; both are multi-dimensional Gaussian variables whose dimensions depend on the problem considered. The definition of what counts as a **Type B** anomaly amounts to putting a threshold on the value of the latent KL divergence, $D_{KL}(q_\phi(z|x)\|p_\theta(z))$, while **Type A** anomalies are found by comparing the real feature vector x to the one predicted by the observation model $\mu_\theta(z)$, and rescaled by the predicted standard deviations $\sigma_\theta(z)$. This amounts to computing the max norm of the difference vector (i running on the dimensions of the space):

$$\max_i \frac{|x_i - \mu_\theta(z)_i|}{\sigma_\theta(z)_i} \tag{8.3}$$

That criterion is triggered if a feature of the vector x deviates from the predicted value $\mu_\theta(z)$ significantly more than the predicted standard deviation $\sigma_\theta(z)$, allowing e.g. to characterize 3-sigma deviations (being reminded that type A anomalies intervene on a single coordinate of the observations).

(a) Generative model **(b)** Inference model

Figure 8.5: Generative and inference models associated with the conditional VAE presented in [Pol+19].

Limitations. The potential limitation of the approach is twofold − although these limitations were not experimentally encountered in [Pol+19]. The first limitation (regarding the detection of type B anomalies) lies in assessing whether the KL term is unusually high ($\log p_\theta(z)$ unusually low). As discussed in Section 8.1, the use of the log-likelihood is not always a good indicator, especially in high-dimensional spaces [Nal+19b; CJA19; Nal+19a]. However, the fact that one considers here the latent Z as opposed to the observed X alleviates this issue.

The second limitation is when, even though the ELBO presents an abnormally low value, the gap is equally spread among several variables, making it impossible to precisely identify the faulty variable. This situation might arise in a hierarchical generative process where several nodes take a slightly unusual value, and their combined effects lead to a very unexpected end result. The question of whether these cases should be treated as anomalies or not is generally problem-dependent.

8.4.2 Conditional anomaly detection

In order to detect malfunctioning sensors at the CERN, another constraint needs to be considered: the question of whether a datapoint is an anomaly or not depends on some external factors, K, e.g. reflecting the nature of the ongoing LHC experiments. Depending on the current experiment, we have some expectations about the behavior of the trigger system.

Accordingly, the data distribution should be learned as $p_\theta(x|k)$, prompting for the use of a conditional deep LVM (Figure 8.5).

This conditional formulation plays a crucial role in the use of this model for anomaly detection. As discussed in Section 8.4.1, K should rather be viewed as a deterministic variable, available to both generative and inference models. This ensures the model uses Z to only encode the relevant information not already contained in K. This role of Z is enforced by the KL term of the ELBO, that drives $q_\phi(z|x,k)$ (and thus $p_\theta(z|x,k)$ as well, Section 3.4) to be as close as possible to the fixed latent distribution $p(z)$.

The KL information pressure is such that Z is as parcimonious as possible, thus avoids duplicating any information already contained in K,[3] as long as the decoder

[3]The value of the latent KL loss of a VAE can be interpreted as the amount of information the decoder has to specify to distinguish the particular sample at hand in the whole generative latent distribution$p_\theta(z)$. ELBO training thus drives the model to build a latent representation which allows this using as little information as possible. It follows from this that any information provided to the model via the K variable but still encoded in Z by the inference model will cause an ELBO penalty compared to the optimal model.

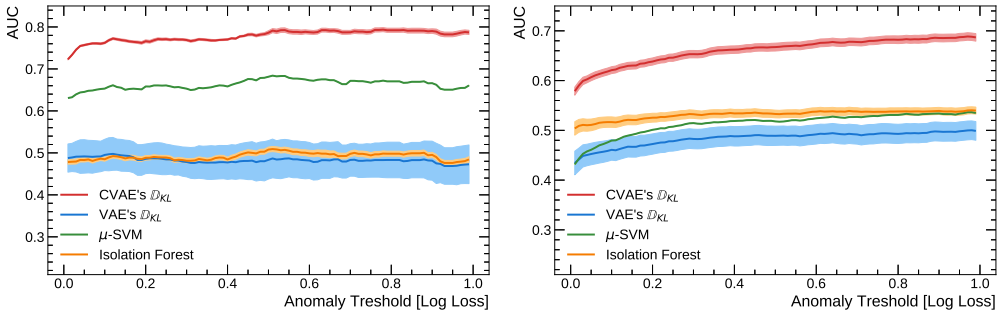


Figure 8.6: Reported ROC area under curve (AUC) for MNIST (left) Fashion-MNIST (right) datasets and different anomaly detection algorithms as a function of varying anomaly threshold t based on LeNet-5 [Lec+98a] model classification log loss. Overall classifier accuracy is 98.95% and 89.62% for MNIST and Fashion-MNIST respectively. The curves stay relatively flat due to high performance of the classifier: most of the test samples have log loss smaller than 0.01. The variance is computed over 5 independent runs.

neural network implementing $p_\theta(x|z,k)$ can appropriately mix Z and K into the final prediction X. As above, Z being concise makes it less susceptible to dimensionality issues.

8.4.3 Empirical validation

This method for anomaly detection was empirically validated on both toy examples (a synthetic dataset, MNIST, and Fashion-MNIST) as well as the proprietary data from the CMS experiment [Pol+19], confirming the above considerations as follows. The Conditional VAE is referred in the figures as *CVAE*.

In the context of the MNIST and Fashion-MNIST datasets, the notion of conditional anomaly is intuitively defined as an image that does not look like its class, either due to a labeling error, or because the image is fundamentally borderline (for example, a very deformed digit, which can be hard to classify). It is observed that the conditional VAE (CVAE, in Figure 8.6) does detect such examples as **type B** anomalies, and that this classification strongly correlates with the uncertainty of a classifier trained on the same dataset: anomalous datapoints are the most difficult ones to classify. In particular, one observes that the conditional part of the VAE significantly improves the anomaly detection, further backing the claim that it is easier for the model to decide whether an image is weird-looking when knowing its alleged class.

The tests on the synthetic and CMS data allow one to assess the behavior of the model of **type A** and **type B** anomalies compartively to the ground truth (Figure 8.7). The synthetic data used 100-dimensional X values, generated from a 5-dimensional K external factor. In the case of the CMS data, X is composed of the trigger rates of 24 HLT, while K is composed of the trigger rates of the 4 L1 triggers that feed them.

It is observed that **type A** anomalies are very easy to detect (being of large amplitude in both datasets), while **type B** anomalies are more challenging. However, the conditional VAE significantly outperforms the vanilla VAE, illustrating the

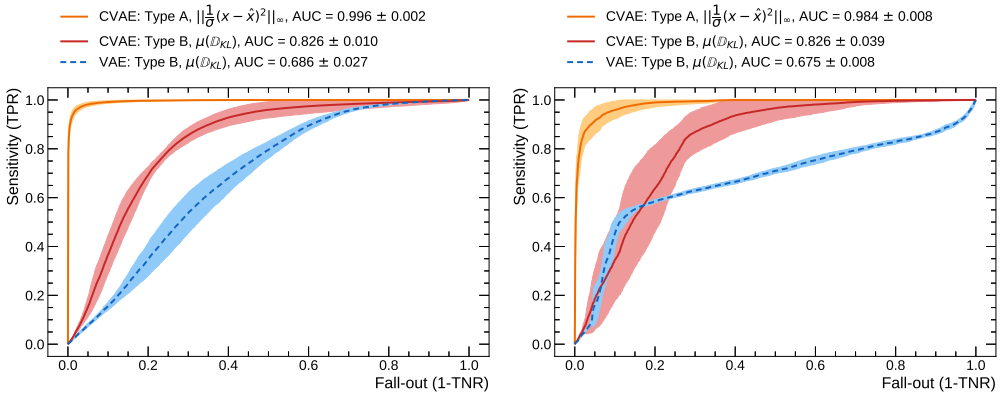


Figure 8.7: Anomaly detection: ROC curves for the synthetic test dataset (left) and the CMS trigger rates dataset (right). The represented variance corresponds to 5 independent runs. The anomaly score for **type B** reports the average D_{KL} of z. For CMS trigger case with low false-positive rate (fall-out), VAE slightly outperforms CVAE but remains within the training variance. On **type A** anomalies, VAE and CVAE have very similar performances.

positive impact of using K and letting the latent variable Z focus on the anomaly detection.

8.5 Summary

This chapter focuses on the design of latent structures in order to achieve different goals: supervised and semi-supervised learning; structure-based learning; and anomay detection.

Classification can be achieved by the use of conditional generative models in the context of *generative classification*, producing models that are more resilient to adversarial attacks and out-of-distribution samples, by avoiding unreasonably confident predictions, making it easier to detect anomalous examples and abstaining from classifying them. The integration of the conditional generative model in a larger structure [Kin+14] allows to expand this construct to semi-supervised learning, and train a discriminative classifier using the generative one as a teacher. This approach also nicely illustrates how multiple datasets with different observed variables can be used to jointly train a single model.

Regarding data structures, the introduction of deterministic latent variables in the LVM [Kar+17] can significantly impact the training dynamics of the model, and allow finer control over the flow of information through the variables. In particular, sequential models can more efficiently learn temporal dependencies through a backbone of deterministic variables, rather than probabilistic ones for which the training dynamics can easily get stuck in local minima.

Finally, formulating the question of anomaly detection in the context of generative graphical models enables to finely distinguish among different types of anomalies. In the presented contribution [Pol+19], we show that the generative model (via the inference model) can be structured so as to distinguish several parts in the data producing system, here enabling to identify faulty behavior from two different

systems in the CMS experiment at CERN. The ability of deep LVMs to represent data using low-dimensional variables also seems to make them more robust to the curse of dimensionality.

Chapter 9
Compositional VAE: structure-enforced properties

Contents

This chapter describes our second main scientific contribution, the Compositional Variational Autoencoder (CompVAE), first presented in [BS20a]. This contribution originates from the context of smart energy policy design and infrastructure dimensioning. More precisely, the goal is to generate the electrical consumption curves aggregating the consumption of a few dozen households, based on little metadata information about these individual households, under various usage scenarii. The difference, compared to other approaches related with residential consumption forecasting [Cia+13; Ort+14; Zha+18], is that the latter aim to precise short or midterm forecasting built on probabilistic models of electrical consumption of individual appliances.

The considered goal was formulated and addressed at a general level: define compositional models, trained from and able to generate whole instances, involving a varying number of entities. In other words, the point is to define a programmable probabilistic simulator, trained from aggregated instances. The presented approach, tackling the compositional generative goal, faces three challenges:
Firstly, the number of households is not fixed in advance, and the model needs to potentially accommodate a large number (a few hundred) of households based on their metadata.
Secondly, the aggregated consumption should be invariant w.r.t. permutation of the

households.
Thirdly, the aggregated model should account for the fact that the individual curves are *not independent*: they are all conditioned by same global factors, e.g. weather or holidays.

These requirements are addressed using a generative model conditioned by a *multi-set*[1], hence the name *Compositional Variational Auto-Encoder*. This approach is more generally designed to enable generating instances of some "whole" based on a multi-set description of its parts.

9.1 A latent space supporting composition

A main specific requirement of the tackled problem is to control the generated output based on a multi-set, denoted $\{\ell_i\}$. As detailed in Section 5.2.1, quite a few Latent Variable Models can handle sequential data [FA15; Bow+15; Chu+15; Fra+16; OKK16; Oor+16b]. The goal here is to ensure that the eventual generated output does not depend on the arbitrary order on the ℓ_i elements, that is, achieving a permutation-invariant representation: the behavior of the model is not affected by a permutation is applied on its inputs. Encoding such invariances in the structure of the model and the architecture of the neural networks has a significant positive effect on the training and model quality [RSP17b; BPC19]. In CompVAE, this invariance property is sought using a 2-level representation, as follows.

9.1.1 Definition of the latent structure

Let C denote the set of external factors (e.g. weather or holidays) generally conditioning all simultaneous consumption curves.

Besides the source of variations represented by C, each i-th part of the whole has some additional *internal* variation on the top of its description via ℓ_i. In the case of electrical consumption for instance, ℓ_i might be the type of electrical contract of the i-th household; this type of contract does not entirely characterize the behavior of a household.[2] This remark leads to associate each i-th part with a continuous latent variable W_i, aimed to capture this internal variation. Along this line, the model learns an associated distribution $p_\theta(w_i|\ell_i)$.

Secondly, as said, the different parts involved in the whole depend on some *global* factor C. For electrical consumption, the instanciation of C might represent an external event that jointly affects the consumption of all households (a match on the TV). For image generation, C might indicate the general light and light orientation conditions. For music generation it could capture the synchronization of the different instruments. This lack of independence is modelled by introducing another latent variable Z, depending on C (the global factors) and possibly the $\{W_i\}$ as well: $p_\theta(z|c, \{w_i\})$.

[1] A *multi-set* differs from a standard mathematical set as it can contain multiple copies of some elements.

[2] Likewise, when generating an image based on the involved entities, ℓ_i might specify that the i-th entity s a chair; still there are many instances of chair.

Eventually, the output X is generated by combining Z and the $\{W_i\}$ in a last step, $p_\theta(x|z, \{w_i\})$, yielding the following factored distribution:

$$p_\theta(x, z, \{w_i\}|c, \{\ell_i\}) = p_\theta(x|z, \{w_i\})\, p_\theta(z|c, \{w_i\}) \prod_i p_\theta(w_i|\ell_i) \tag{9.1}$$

The next step is to describe how X and Z depend on the multi-set $\{W_i\}$. The use of recurrent neural networks is not appropriate: they impose an arbitrary ordering of the elements, and pose several optimization challenges when considering a large number of elements. For this reason CompVAE takes inspiration from the *encoder-aggregator-decoder* structure that has been used in previous work dealing with set-structured data [ES17; Esl+18; Gar+18; Lee+19] and introduces a specific *deterministic* latent variable $\widetilde{W}$ that aggregates the $\{W_i\}$, and conditions X and Z with $\widetilde{W}$ (Figure 9.1):

$$\widetilde{w} = \Psi(w_1, w_2, \dots w_K) \tag{9.2}$$

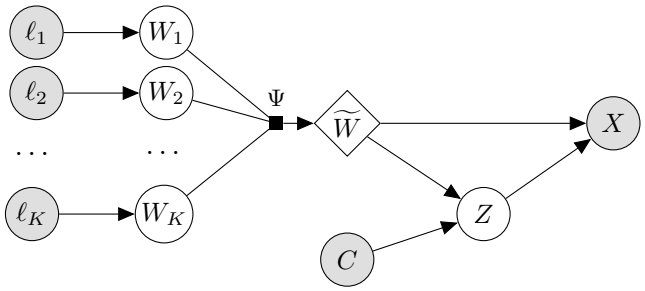


Figure 9.1: Representation of the CompVAE generative model.

The aggregation function Ψ must apply on any number of inputs, be invariant with regard to their order, and capture the multiplicity of a same element in the multi-set.

The straigtforward approach, used in CompVAE, is to consider their sum:

$$\widetilde{w} = \sum_i w_i$$

The investigation of other aggregators is left for further work. This aggregation of the individual w_i has important implications on the inference model, and the choice of an additive aggregation is in large part driven by the fact that there are efficient and simple ways to control for the sum of a set of random variables (more in Section 9.2).

9.1.2 Handling the variable number of parts in neural architecture

A key issue in CompVAE is to account for the fact that the generated whole X must reflect the number K of pats involved in the composition. Two main situations are considered, depending on whether the general amplitude of X is proportional to K (e.g. in the case of the consumption curves) or not (e.g. in images, each pixel varies in the same range, and the number of pixels remain the same, whatever the number of objects in the scene).

Since the 2010s, many neural networks are based on activation functions akin the ReLU [GBB11], with value linear in their input unless they are saturated. In this case, when the amplitude of the input is doubled, the overall amplitude of the output is approximately doubled as well. Accordingly, when using such networks for modeling p_θ in CompVAE, the amplitude of Z and X is proportional to that of $\widetilde{W}$, and thus to K, the number of parts.[3]

9.2 Inference model over multi-sets

Symmetrically, the design of the CompVAE inference model poses the inverse problem of that of the generative model: the information contained in the variable X needs to be split into the different $\{W_i\}$ variables. This inference model naturally factors as:

$$q_\phi(z, \{w_i\}|x, c, \{\ell_i\}) = q_\phi(z|c, x)q_\phi(\{w_i\}|x, c, z, \{\ell_i\}) \tag{9.3}$$

The question is how to implement $q_\phi(\{w_i\}|x, c, z, \{\ell_i\})$.

9.2.1 The recurrent network approach

Our first attempt to build the inference model relied on factorizing further the q_ϕ distribution by introducing an arbitrary ordering of the elements, as follows:

$$q_\phi(\{w_i\}|x, c, z, \{\ell_i\}) = \prod_i q_\phi(w_i|x, c, z, \ell_i, \{w_{<i}\}) \tag{9.4}$$

Such a factorization can easily be implemented using a recurrent neural network: at each step, the network is given a triplet (x, c, ℓ_i) as input and predicts a distribution on w_i as output, the dependency on $\{w_{<i}\}$ being handled by the recurrent state of the network. This approach however does not scale well to large numbers of elements (due to long-term dependencies problems with RNNs, as discussed in Section 5.2.3). We also observed that, even with a small number of elements, the information does not in practice split equally between the different w_i; quite the contrary it is observed that the w_is are almost insignificant for $i \geq 3$. While the RNN achieves a good reconstruction, the model thus fails to capture the link between w_i and ℓ_i, resulting in a very poor generative quality.

Our second attempt, aimed to mitigate the above loss of signal issue, was to explicit the recurrent state of the model. In the former attempt the recurrent state does not directly depend on the sampled value of each variable w_i, blurring the dependency on $\{w_{<i}\}$. The second attempt thus made it explicit that the relevant recurrent state is the sum of the previously extracted w_i, hopefully enabling the extraction of w_i. The sought distribution thus reads:

$$q_\phi(\{w_i\}|x, c, z, \{\ell_i\}) = \prod_i q_\phi\left(w_i \middle| x, c, z, \ell_i, \sum_{j<i} w_j\right) \tag{9.5}$$

[3] If such a proportionality is not desirable, the structure of the neural networks needs to be adjusted accordingly, for example by introducing (at least) one layer of saturating activation functions like the sigmoid or the hyperpolic tangent, as done in section 9.3.2.

The above extraction procedure behaves like a hand-rolled RNN: an accumulator variable, initialized to 0, holds the partial sum $\sum_{j<i} w_j$; then at each step this accumulator is given as input to the neural network, in addition to x, c, z and ℓ_i, and its output is the probabilistic prediction for w_i, from which a value is sampled and added to the accumulator, as illustrated by Figure 9.2.

This second attempt however faced the same issue as the first one, most of the information being captured by the first few variables. Another, non-recurrent approach was thus considered.

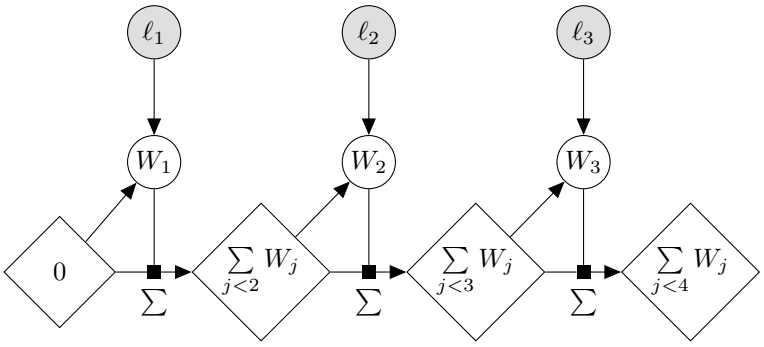


Figure 9.2: Graphical representation of the recurrent inference model with explicit recurrent state. The dependency of each w_i node on (x, c, z) is omitted for clarity.

9.2.2 Correlated Gaussian prediction

The failure of the inference model to adequately split the information between the different W_i variables is blamed on the choice to process them sequentially, rather than as a whole. Therefore, we opted for designing an inference model that directly predicts a joint distribution over the set $\{W_i\}$.

In order to ensure an accurate reconstruction of X, the key issues are to identify $\widetilde{W}$ and Z; the W_is are only directly involved in the KL term of the ELBO.

For each (vectorial) W_i in the latent space, its coordinates are assumed to be independent; for simplicity, W_i will be considered as a scalar in the remainder (its prediction actually proceeds coordinate-wise). The simplest model considers a diagonal Gaussian distribution over the vector $(w_1, w_2, \ldots, w_K)$, where the neural network predicts a mean and variance pair (μ_i, σ_i^2) for each w_i. We modify the diagonal covariance matrix to allow the model to control for the variance of the sum of these variables: the motivation is to enforce a small variance on the prediction of the value $\widetilde{w}$ while enabling a larger variance on the prediction of each w_i, to allow a tighter ELBO.

The motivation for this requirement (enabling a high variance for each individual W_i while enforcing a low variance on $\widetilde{W}$ through correlating the W_is) is the following. On the one hand the low variance on the prediction of $\widetilde{W}$ is required for a good reconstruction of X by the model. On the other hand, the inference model should not be required to identify the individual contributions of the parts. Alleviating the need for individual predictions allows one to increase the entropy of the inference prediction (thus improving the ELBO), and avoids the need for the neural network to break the symmetry between similar parts, a far from trivial task.

To control the variance of $\widetilde{W}$, a new parameter $0 \leq \rho_i \leq 1$ for $i = 1$ to K is introduced, informally defining how much W_i varies conditionally to $\widetilde{W}$; $\rho_i > 0$ enforces a negative correlation between W_i and $\widetilde{W}$, as formalized in Equation 9.6.

Let ϵ_i and $\mu_i + \sigma_i\epsilon_i$ be samples respectively drawn after $\mathcal{N}(0;1)$ and $\mathcal{N}(\mu_i;\sigma_i^2)$). The ρ_i parameter is used in the sampling procedure of w_i, as:

$$w_i = \mu_i + \sigma_i\epsilon_i - \rho_i \sum_j \sigma_j\epsilon_j \tag{9.6}$$

The above sampling procedure defines a multivariate normal distribution of all W_is, such that the variance of $\widetilde{W} = \sum_i W_i$ is controlled by the ρ_i parameters with:

$$Var\left(W_i\right) = (1-\rho_i)^2\sigma_i^2 + \sum_{j\neq i} \rho_j^2\sigma_j^2 \tag{9.7}$$

$$Var\left(\widetilde{W}\right) = \left(1 - \sum_i \rho_i\right)^2 \left(\sum_i \sigma_i^2\right) \tag{9.8}$$

thus, the variance of W_i can be high while the variance of $\widetilde{W}$ is low. In the case where $\sum_i \rho_i = 1$, the variance of $\widetilde{W}$ is 0, making its prediction deterministic. In the particular case where $\rho_i = 1/N$, with N the number of parts in the whole, then one has as desired the variance of $\widetilde{W}$ set to 0, while the variance of each W_i is $(1-\frac{2}{N})\sigma_i^2 + \frac{1}{N}\sum_j \sigma_j^2$, which can still be relatively large.

The density of the w_i distribution reads, with $|\Sigma|$ the determinant of the covariance matrix of the w_i:

$$\log q_\phi(\{w_i\}|\{\mu_i\},\{\sigma_i\},\{\rho_i\}) = -\frac{1}{2}\sum_i \epsilon_i^2 - \frac{1}{2}\log|\Sigma| - \frac{K}{2}\log 2\pi \tag{9.9}$$

This determinant can be computed analytically (with detailed derivation in Appendix 9.A):

$$\frac{1}{2}\log|\Sigma| = \log\left(1 - \sum_i \rho_i\right) + \sum_i \log\sigma_i \tag{9.10}$$

A non-admissible case is when $\sum_i \rho_i = 1$, where the variance of $\widetilde{W}$ is zero and the determinant of the covariance matrix Σ is 0. This case is avoided by choosing an appropriate parameterization, e.g. by setting ρ_i as the softmax of some ν_i in the real space:

$$\rho_i = \frac{e^{\nu_i}}{1 + \sum_j e^{\nu_j}} \tag{9.11}$$

ensuring that both the sum of the ρ_i is less than 1 and ρ_i is in interval $(0,1)$.

This parameterization also interestingly supports the closed form computation of the KL-divergence between $q_\phi(\{w_i\}|z,x,c,\{\ell_i\})$ and $p_\theta(\{w_i\}|\{\ell_i\})$ (detailed in Appendix 9.B).

9.2.3 Using graph neural networks

The neural network defining $q_\phi(\{w_i\}|z, x, c, \{\ell_i\})$ thus yields three values μ_i, σ_i and ν_i for each coordinate of W_i. The chosen architecture, easily scaling up with the dimension size of the latent W space, is that of graph neural networks (GraphNN) [Sca+09; Gil+17; Wu+21].

In GraphNN, each layer of the network is structured according to a same graph (Figure 9.3): each node in a layer receives as input its previous state and the output of its neighbor nodes computed in the former layer; the same activation function is used in each node for a given layer. Formally, letting L denote the layer index, with h_i^L the state of node i at layer L, and f_L the function encoded by the neural network on the L-th GraphNN layer, then:

$$h_i^{L+1} = f_L(h_i^L, \{h_j^L\}_{j \in N_i}) \tag{9.12}$$

for N_i the set of neighbor node of the i-th node.

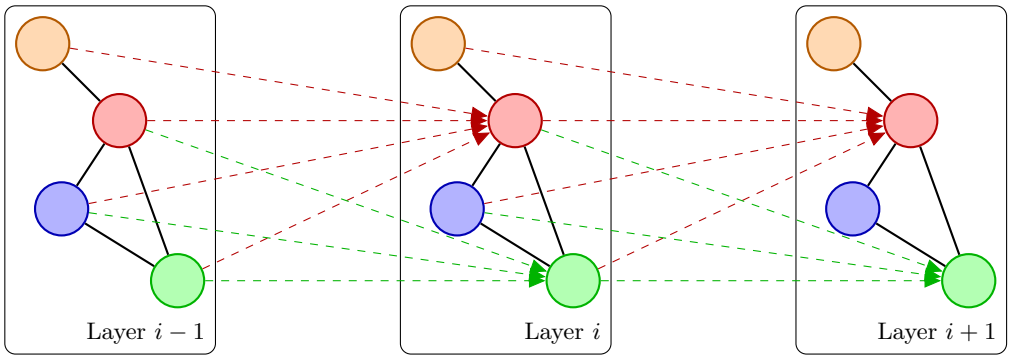


Figure 9.3: A GraphNN: within each layer, the neurons (circles) are connected using a same graph (black lines). The value of each i-th neuron is computed from the values of the i-th neuron and its neighbors in the previous layer. For readability, the computation graph is only shown for the red and green neurons.

In order for the function definition to be independent of the number of neighbors, the function f_L is implemented as taking the sum (or average) of some projection of the state at the neighbors via a function Φ_L:

$$h_i^{L+1} = f_L\left(h_i^L, \sum_{j \in N_i} \Phi_L(h_j^L)\right) \tag{9.13}$$

where f^L and Φ^L are implemented and learned as standard neural networks.

In CompVAE, the GraphNN is built on a fully connected layer graph[4]. It gradually exploits and refines individual parts and whole information (respectively, the $\{\ell_i\}$ element metadata, the state of the other variables, and the global (x, z, c))), as illustrated by Figure Figure 9.4. Note that the global (x, z, c) is pre-processed by

[4]Even with a fully-connected graph, the computational cost remains reasonable thanks to the chosen structure: Φ_L needs only be computed once, and the sum on all neighbors can be efficiently computed by first computing the sum on all nodes, and then subtracting the value from the current node from that total sum. The complete process thus remains linear in the number of elements of the composition, not quadratic.

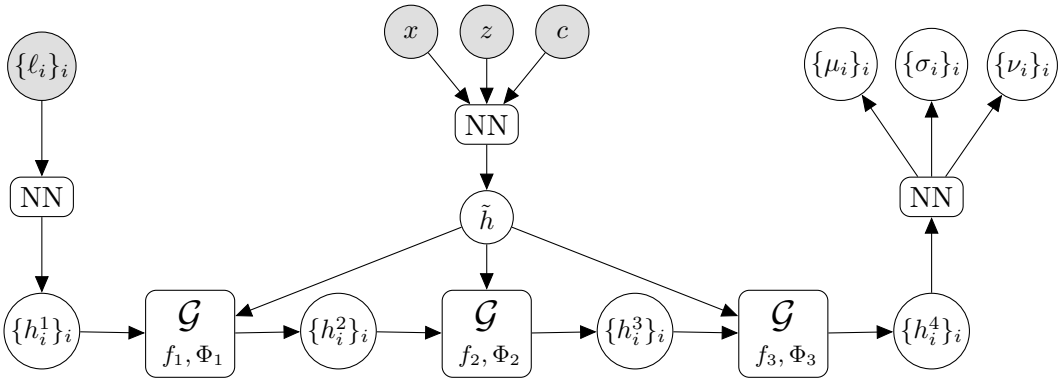


Figure 9.4: Graphical representation of the GraphNN used in the inference model of CompVAE with 3 GraphNN layers($\mathcal{G}$). Each "NN" rectangle represents a neural network independent of the graphical structure. When applied to a set of variables $\{\dots\}_i$, these neural networks are applied independently on each element to produce a the output set of variables.

another neural network into a latent value $\tilde{h}$, which is provided as input to both f_L and Φ_L in each layer of the GraphNN:

$$h_i^{L+1} = f_L\left(h_i^L, \tilde{h}, \sum_{j \in N_i} \Phi_L(h_j^L, g)\right) \tag{9.14}$$

9.3 Empirical results

The proposed CompVAE architecture is validated on two artificial problems and the real-world problem motivating the design of the approach, the generation of electrical consumption curves. Preliminary results on both artificial problems in [BS20a]; the validation real-world data has been completed since and is also provided here.

9.3.1 1D artificial problem

In the first artificial problem, a part is a sine curve; the whole is made of the non-linear sum of these curves. In this problem, the output amplitude is proportional to the number K of parts, and each part affects all coordinates of the whole.

Each part (sine curve) is defined by its frequency, amplitude and phase. The associated meta-data consists of the single frequency ℓ_i (and thus is known when generating the curves); the amplitude a_i and phase κ_i are meant to be captured by the latent variables w_i. The whole curve is obtained by applying a *tanh* function on the sum of the K curves, inducing a saturating behavior controlled by a hyper-parameter λ. The whole curve is then sampled according to some time resolution T to produce a vector of values:

$$x[t] = K \tanh\left(\frac{\lambda}{K} \sum_{i=0}^{K} a_i \cos\left(\frac{2\pi\ell_i}{T}t + \kappa_i\right)\right) \tag{9.15}$$

The observation model thus is defined as a diagonal Gaussian observation on the vector $x[t]$, for $t = 0$ to T.

The frequency ℓ_i of each sine curve is randomly sampled in the finite set $\{1, 2, \ldots 10\}$; amplitude a_i is sampled from a normal distribution $\mathcal{N}(1, 0.3)$; phase κ_i is sampled from a normal distribution $\mathcal{N}(0, \pi/2)$. Two global curves generated from the same four sine curves are depicted in Figure 9.5, in order to compare the curve obtained by the tanh of the sum and that of the sum of the tanh. The point of this artificial problem is to investigate the case where the whole does not boil down to the sum of the parts.

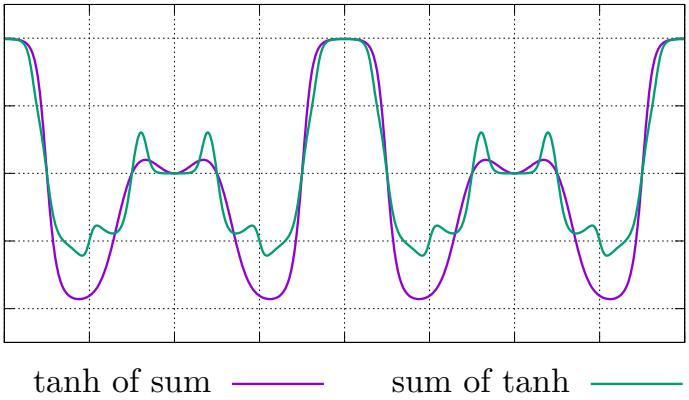


Figure 9.5: Non-linear part-to-whole aggregation (purple) compared to the sum of non-linear perturbations of the parts (green). Both curves involve a non-linear transform factor $\lambda = 3$.

The compositional behavior of the model learned by CompVAE is illustrated on Figure 9.6, displaying the whole curve generated from an increasing number of the individual sine curves. All individual w_is are sampled from the generative model $p_\theta(w_i|\ell_i)$ and the corresponding individual curves are displayed (Figure 9.6, left). The overall curve (Figure 9.6, right) is generated from the partial sums: from top to bottom, one sees the overall curve generated from w_1 alone, then $w_1 + w_2$, then $w_1 + w_2 + w_3$, forming a coherent composition of the different individual parts.

The model used in this problem is built using a 10-layers residual network for the generative model and 3 GraphNN layers combined with 6 residual layers for the inference network. It takes approximately 10 hours to train using a GTX1080 GPU.

9.3.2 2D artificial problem

The second artificial problem is concerned with the generation of an image (the whole) combining parts described as color anchors. Formally, a part (color anchor) is defined from its location, color, and intensity. The associated meta-data ℓ_i consists of both the color varying in $\{white, black, red, green, blue\}$ and the location of the anchor point encoded as a 2D vector in $[0; 1]^2$; the intensity of the anchor point is left to be captured by w_i. Each part induces a color gradient on the blank canvas. The whole is an image alike a Voronoi diagram (Figure 9.7) where the color in each pixel is set to a weighted combination of the anchor colors, weighted by the distance of the pixel to the anchor and the intensity thereof.

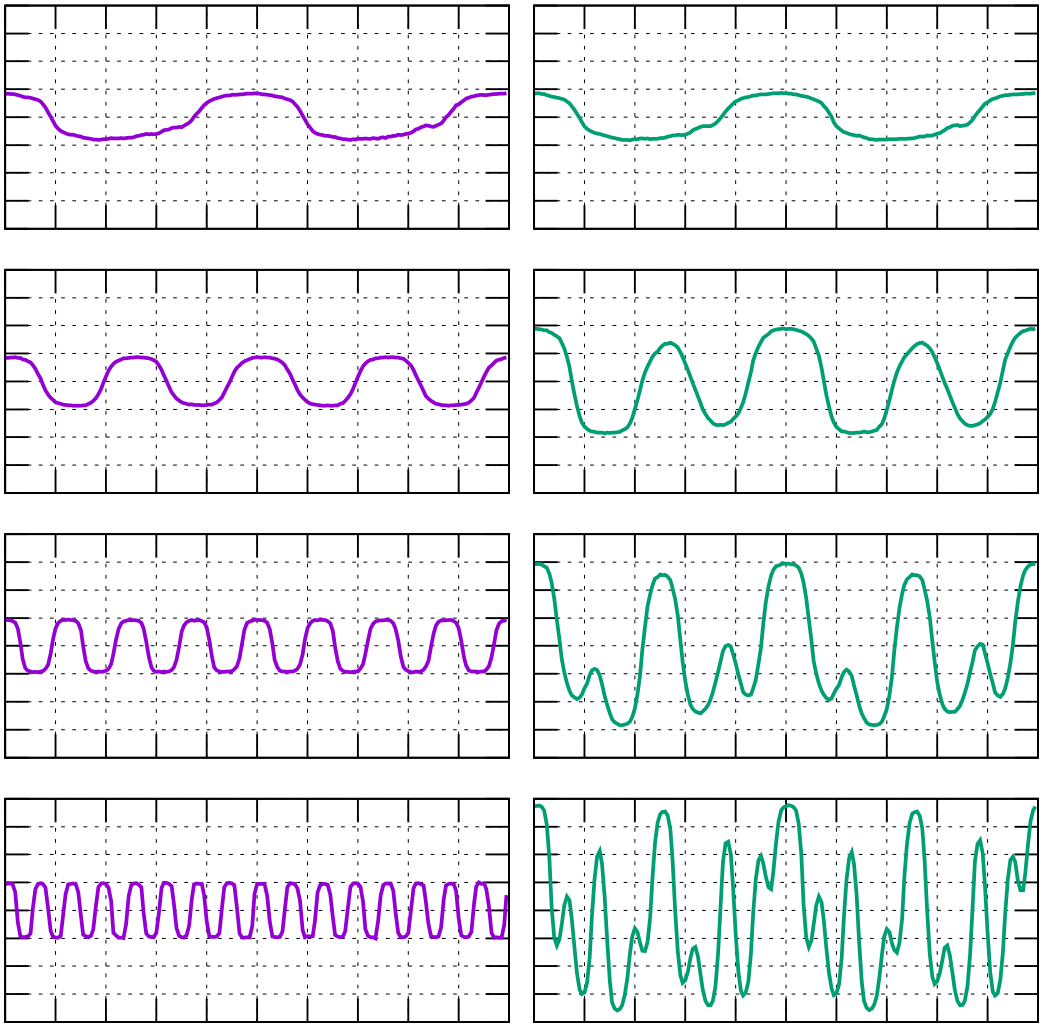


Figure 9.6: The compositional CompVAE generative model on the 1D artificial problem: On each row is displayed a part (left) and the whole (right) made of this part and all above parts.

This 2D problem presents two additional difficulties compared to the former 1D problem. Firstly, the whole output (the image) is of constant amplitude and does not depend on the number K of parts; pixel values lie in the $[0; 1]$ interval whatever K is. Secondly, each part only has a local impact on the whole.

The first issue implies that the generative model needs to include a saturating mechanism. Two approaches have been considered, based on the combination function ψ and the observation model.

The aggregation function Ψ (Equation 9.2) is modified by applying, on the additive aggregation of the W_is, the following saturating activation function:

$$x \to \chi(x) = \frac{x}{1 + \gamma|x|} \tag{9.16}$$

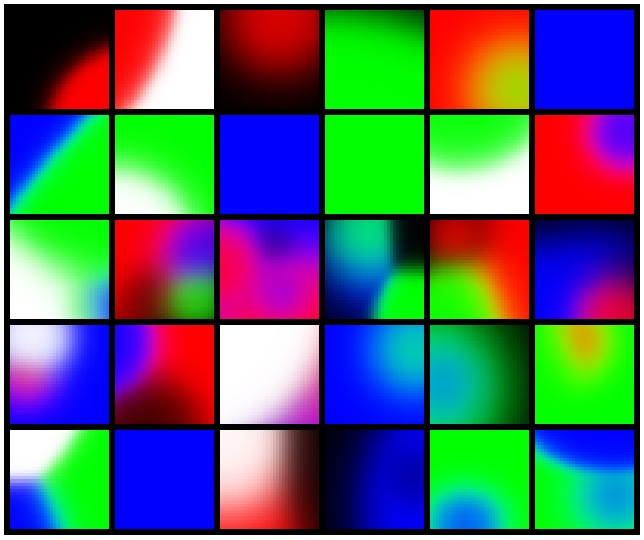


Figure 9.7: Examples of training images for the color gradient composition problem.

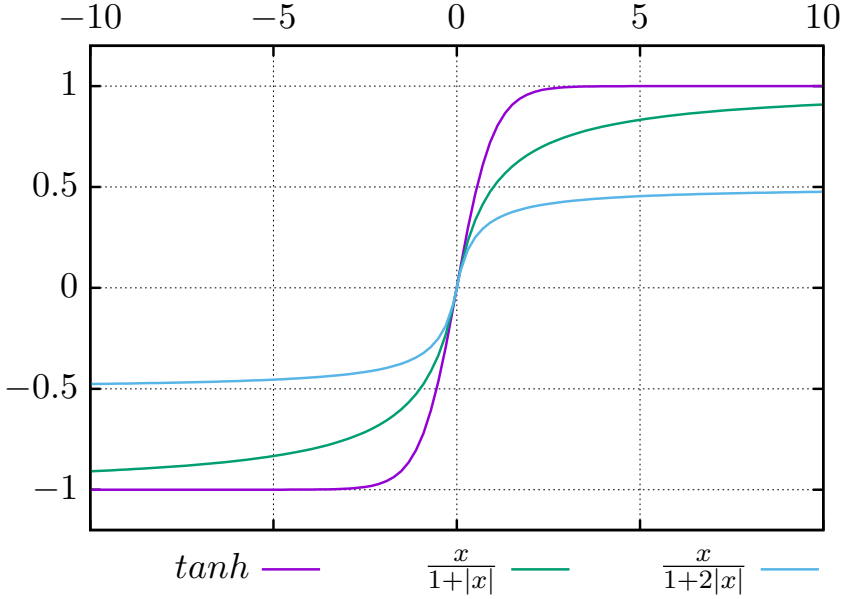


Figure 9.8: Comparison of the *tanh* activation to $x \to \frac{x}{1+\gamma|x|}$.

and finally,

$$\widetilde{W} = \chi\left(\sum_i W_i\right) \tag{9.17}$$

The coefficient γ is learned as a global parameter, and initialized to 1.0. This activation function has a saturating behavior similar to that of tanh but is slower to saturate (Figure 9.8), preventing gradient vanishing in the early learning stages, when $\widetilde{w}$ still has a large variance.

Additionally, the observation model is built using the discretized logistic model [Sal+17] (Section 5.2.2). This model generally behaves as a *quasi-deterministic* observation model (Section 6.2), but it is built on the discrete domain of pixels (that is, $\{0, 1, 2, \ldots, 255\}$). It thus acts as a second saturating layer, enforcing by design

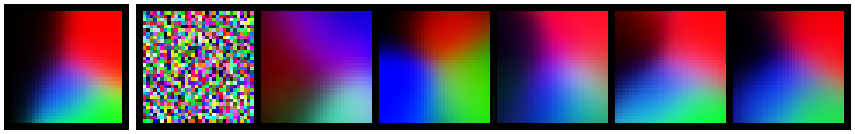


Figure 9.9: CompVAE on the 2D problem, composition of color anchors. Leftmost: the ground truth image. Left to right, generated output at epoch {0, 1, 2, 3, 4 5} $\times 100,000$. The right colors are identified around epoch 200.000; the right locations are identified much later, around epoch 500,000.

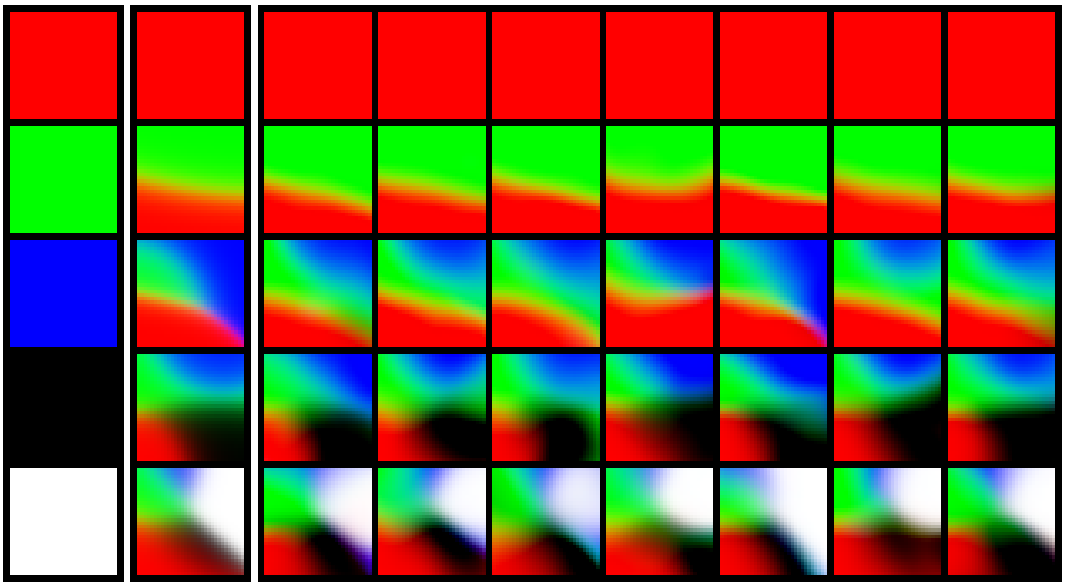


Figure 9.10: The compositional CompVAE generative model on the 2D artificial problem: On each row is displayed a part (column 1), the ground truth whole made of this part and all above parts (column 2) and generated images (columns 3 to 9) generated from this part and all above parts.

the fact that every generated pixel belongs to the proper domain.

The second difficulty is that each part (anchor) only locally affects the whole image locally. The learning process thus must manage to identify which part of the whole image is associated with an ℓ_i part; the training information is more sparse. In contrast, in the 1D problem, every ℓ_i part had an impact on all coordinates of the whole curve. This sparsity of the training signal overall results in a significantly longer training process. Typically, the last thing the model learns is the link between the location of the parts and their effect on the whole image, as shown on Figure 9.9.

The whole process is illustrated on Figure 9.10, displaying the whole image generated from an increasing number of the individual color anchors. As in the 1D problem, all individual w_is are sampled from the generative model $p_\theta(w_i|\ell_i)$ and the overall image (Figure 9.10, right) is generated from the partial sums: from top to bottom, one sees the overall image generated from w_1 alone, then $w_1 + w_2$, then $w_1 + w_2 + w_3$, forming a coherent composition of the different individual parts.

The model used for this experiment is a 20-layer residual network for the generative model, and uses 5 GraphNN layers with 10 additional residual layers for the inference model. Its full training took 2 days on a GTX1080 GPU.

9.3.3 Electrical curves composition

The applicative motivation of CompVAE is concerned with smart energy control policies, and more specifically the dimensioning of infrastructures along diverse usage scenarii. In the context of the NEXT contract (ADEME funding, coordinator Artelys), our industrial partner needs to assess the consumption peak of sets of households under different energy demand settings (weather, electric car). More abstractly, the question is to generate energy consumption curves corresponding to a set of customers (households or factories), where each customer is associated with its contract metadata.

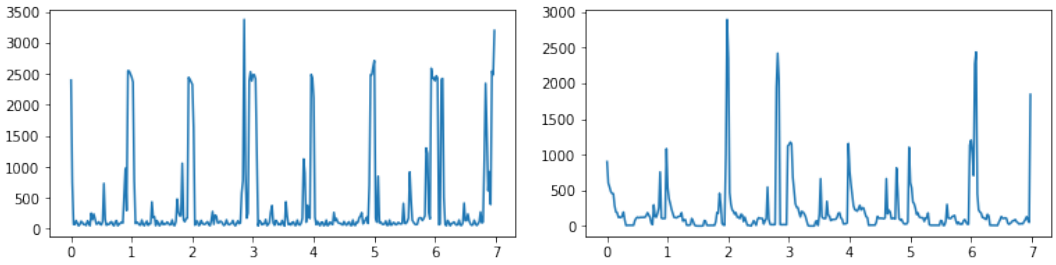
Our input data consists of anonymized consumption curves from around 500 households, covering a span of 5 years for the longest. The consumption is measured bi-hourly, resulting in 48 values per day. Examples of the consumption curve of a single household over a week are displayed in Figure 9.11. These curves illustrate the erratic behavior of household-level consumption in general: the baseline is very low, with spikes of consumption when persons are active, turning lights on and using electrical appliances. Note that the aggregated curves of circa 100 households show a much smoother behavior than individual curves (Figure 9.12). As expected, the aggregation of independent random variables is smoother than the individual variables. Note however that the behavior of different households (the random variables) are independent only to some extent: they are correlated as the households face the same weather and external temperature, the same holiday periods, the same matches on the TV.

However, the comparatively smoother behavior of aggregated households motivates the presented approach: training a generative modeling to directly produce aggregated curves, associated with the multi-set of entity (household) descriptions.

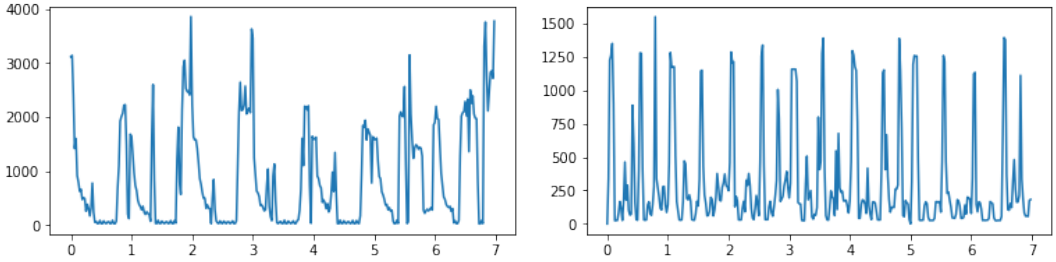
The metadata available for each entity (household) is composed of the type of power contract (categorical variable), the power subscribed (numerical variable), and possibly the group of off-peak hours[5] (also described with a categorical variable). These three values thus compose the ℓ_i label associated with household i. The objective is to build a model that can generate aggregate consumption curves of circa 50-150 households. In addition, the hourly measure of temperature is given as a global metadata C.

CompVAE uses a specific observation model $p_\theta(x, \widetilde{w}, z)$ here, reflecting the multi-scale structure of aggregated electrical curves (Figure 9.12): a strongly regular baseline with a daily regularity, decorated with a fast varying component. This structure leads to the use of a hierarchical observation model as presented in Section 6.2.2. Formally, curve X is decomposed into two observed variables $X1$ and $X2$, where $X2$ is the full-resolution curve, and $X1$ is a smoothed version, sub-sampled by a factor of 2. The model then decomposes as $p_\theta(x2|x1, \widetilde{w}, z)p_\theta(x1|\widetilde{w}, z)$. As in the Laplacian pyramid structure, $X2$ is predicted as a correction to an upscaled version of $X1$. The neural network structure itself is built using 1D convolutional and residual networks: the generator model is composed of 14 residual layers, while the inference model has 5 GraphNN layers, associated with 7 residual layers. The total training time is of approximately 1 day using a GTX1080 gpu.

[5]The electricity provider proposes electrical contracts where the electricity is cheaper during some time range (usually during the night), as an incentive to shift part of the consumption (such as water heaters or washing machines) during off-peak hours.

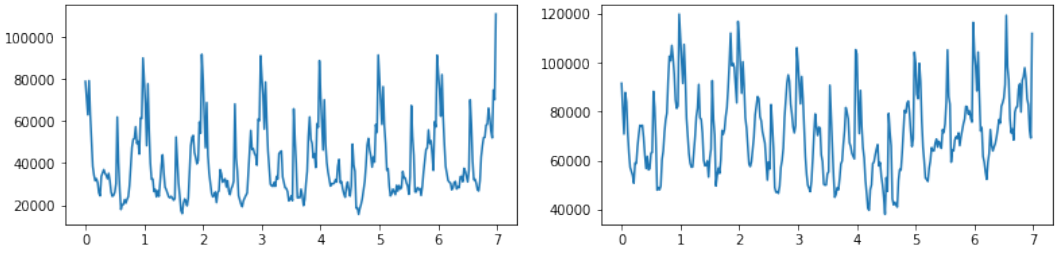


(a) Consumption curve of a household. **(b)** Consumption curve of a household.

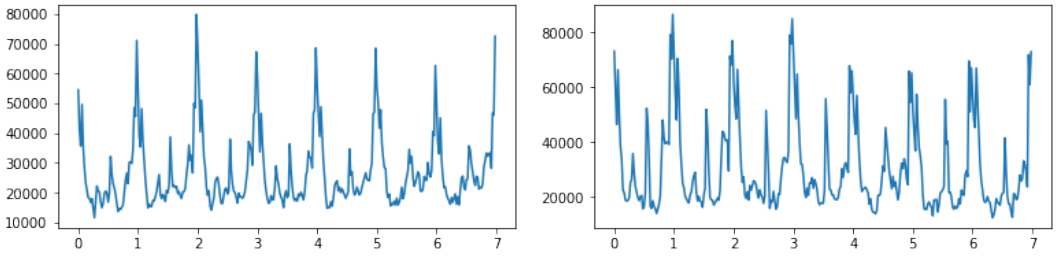


(c) Consumption curve of a household. **(d)** Consumption curve of a household.

Figure 9.11: Examples of consumption curves of single households. Horizontal axis is labeled in days, starting at Monday morning and ending at Sunday evening. Vertical axis is labeled in Watts.

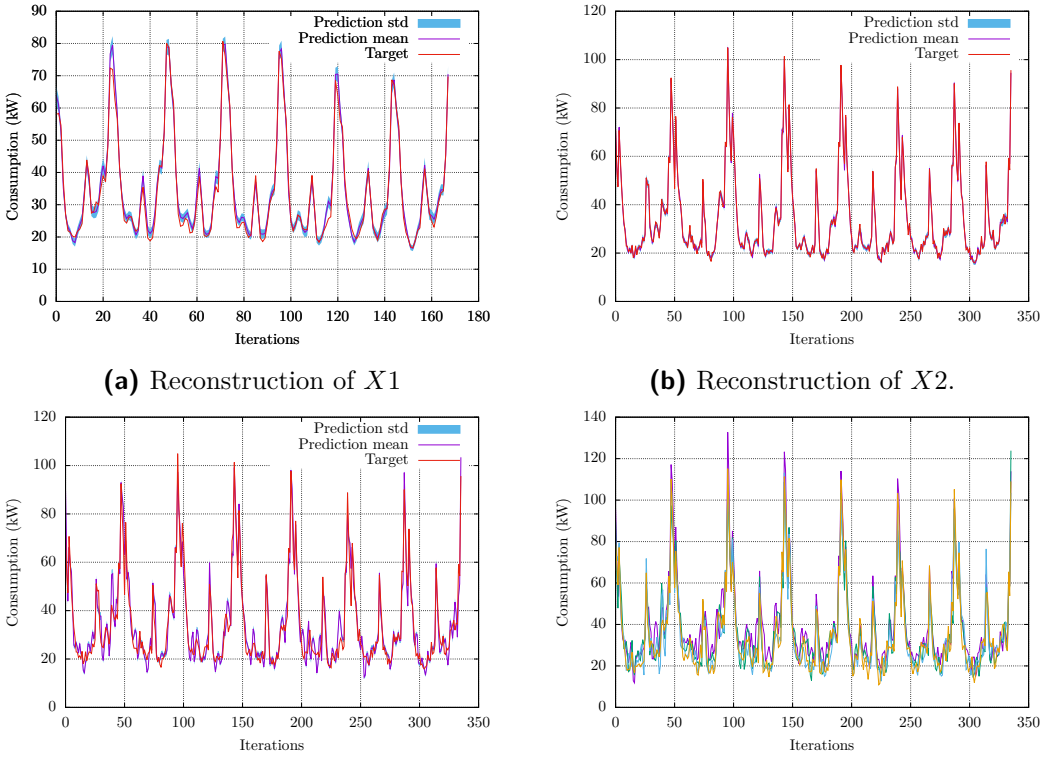


(a) Consumption curve of 100 household. **(b)** Consumption curve of 100 household.



(c) Consumption curve of 100 household. **(d)** Consumption curve of 100 household.

Figure 9.12: Examples of aggregated consumption curves of a hundred random households. Horizontal axis is labeled in days, starting at Monday morning and ending at Sunday evening. Vertical axis is labeled in Watts.



(a) Reconstruction of $X1$

(b) Reconstruction of $X2$.

(c) Stochastic reconstruction of $X2$.

(d) Fully generated samples.

Figure 9.13: CompVAE performance on electrical consumption curves. The reconstructions (a,b,c) show the target curve in red, and the Gaussian model prediction by its mean (purple) and standard deviation (light blue). (a) shows the reconstructed $X1$, (b) the reconstructed $X2$ given the real $X1$, and (c) the reconstructed $X2$ given the reconstructed $X1$. (d) shows 5 generated curves from the same set of $\{\ell_i\}$ metadata.

The training procedure uses aggregated curves that are generated on the fly for each minibatch, uniformly selecting a week and a set of households in the set of 500 households and (at most) 250 weeks. One can thus consider that the training data is virtually infinite[6], which significantly helps avoiding overfitting.

The behavior of the trained model is presented in Figure 9.13. In particular, Figure 9.13(a) and Figure 9.13(b) illustrate the reconstruction quality reached by CompVAE on these data: $X2$ is near-perfect given the real $X1$, and $X1$ itself captures well the shape of the curve, with most of the uncertainty being on the exact height of the peaks and lows. Figure 9.13(c) illustrates what would be a "full reconstruction" from the latent values: Z and $\{W_i\}$ are given from the inference model, but $X1$ is sampled from $p_\theta(x1|\widetilde{w}, z)$ before being given to $p_\theta(x2|x1, \widetilde{w}, z)$, the later being represented on the plot. The resulting prediction, while still fitting the overall shape of the curve, now displays small but significant deviations from it. While the error between the reconstructed curves and the real ones is of $3.8kW(\pm 1.5kW)$

[6] As the number of ways to randomly choose between 50 and 150 elements from a set of 500 or magnitude comparable to 10^{130}

and $4.9kW(\pm 1.6kW)$ per household[7] for individual $X1$ and $X2$ reconstructions, it reaches around $13.9kW(\pm 5.3kW)$ per household for the "full reconstruction". This illustrates how the uncertainty is shared between the two variables $X1$ and $X2$ by the hierarchical observation model.

In generative mode, Figure 9.13(d) displays 5 curves produced by CompVAE upon receiving the same $\{\ell_i\}$ 100-element multiset. All generated curves have same general shape and amplitude as the original one: the average distance between the original curve and a generated one is $20kW(\pm 11kW)$ per household (using the same metadata), to be compared with $92kW(\pm 73kW)$ when the individual curves are independent, that is, the same distance as between two independent real curves. This result confirms that CompVAE correctly identifies the dependency between the metadata and the curve.

The quality of the generated curves is evaluated and compared to real data according to three domain-specific metrics, related to the goal of balancing the electrical network. The first metric is the peak consumption: the maximum value reached by the curve during the week. A too high peak value can trigger some security mechanisms on the grid. A second metric is the peak sustained consumption for a given duration: for example the maximum value that is reached for at least 8h in total during the whole week. Sustained high consumption can also adversely affect the network, e.g. by causing overheating in some parts. A third metric is the *flexibility* of the curve, a technical indicator defined as the L1 distance between the curve and the constant curve of same average consumption over the week. At the grid level, consumption and production must always be balanced; in this respect, the easiest case is that of constant consumption curves. The flexibility thus measures how far the curve is from a constant curve with the same total energy consumed, using the earth-mover's (or *energy*-mover's) distance.

The overall distribution (averaged over the number of households, household metadata and temperature profiles) of these three metrics are compared in Figure 9.14. The provided histograms are computed from 100,000 real aggregated curves and 100,000 curves generated by CompVAE. One observes a small but noticeable bias in CompVAE, that generates curves with slightly higher peaks and a slightly higher flexibility in average, as shown in Fig. 9.14. Following the discussion in Chapter 7, a tentative interpretation might be that, as the generative model did not fully capture the structure of the curves, it provisionned the intrinsic variability of the curves through random noise; this noise might explain the wider range of the curve values, increasing the top-1 value (peak) and its variance (flexibility).

The impact of providing the temperature as global metadata C is assessed through a lesion study, comparing the latent part of the losses for a model trained with and without this metadata, as summarized in Table 9.1. These values can be interpreted as the amount of data the encoder needs to store in the latent variables in addition to the one already provided by the latent model $p_\theta(z, \{w_i\}|\{\ell_i\})$. CompVAE trained without the temperature information converges to storing on average 12 bits in the set of $\{w_i\}$ as a whole, and 25 bits in z, so 37 bits total. This is already quite a small value compared to the size of the data (X is a vector of 336 values), illustrating how

[7] The error is computed as the sum of absolute error across time, divided by the number of households in the composition, so that values associated with different number of households can still be compared.

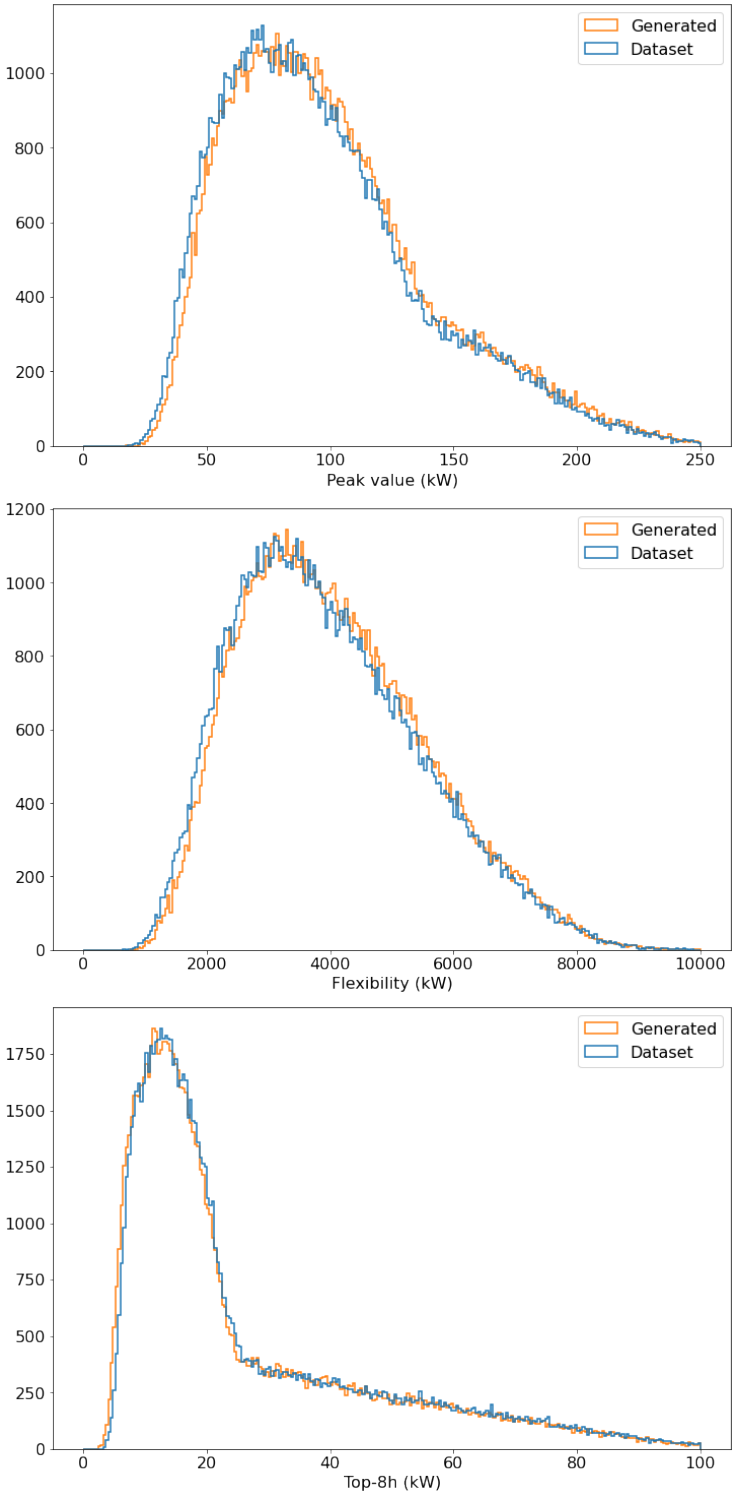


Figure 9.14: Histograms of the overall distribution of generated curves (orange) and the dataset (blue) according to 3 domain-specific metrics: the peak value, the flexibility, and the maximum-value sustained over 8 hours (top-8h).

	$\{W_i\}_i$ loss (bits)	Z loss (bits)	Total latent loss (bits)
Without temperature input	12	25	37
With temperature input	8	18	26

Table 9.1: Lesion study, impact of providing the weather temperature as input to CompVAE: Amount of information stored by the model in the latent variables, as interpreted from the loss values.

X can indeed be well predicted from the set $\{\ell_i\}$. However, the model trained with the temperature input as C converges to around 8 bits in the set of $\{w_i\}$ and 18 bits in z, for a total of 26 bits. The addition of the temperature input thus allows the model to reduce by around 30% the amount of information the inference model needs to provide on top of the trained model to accurately reconstruct the curve, illustrating the expected link between weather and electrical consumption (being reminded that electrical heating is quite common in France). It is interesting to note that even though the temperature is only injected at the level of the Z variable in the model, its usage enables to reduce the amount of information stored both in Z and in the $\{W_i\}$.

9.4 Summary and perspectives

Our second main contribution, CompVAE is a Deep LVM with a specific latent structure aimed to enforce additive semantics in latent space. This structure makes CompVAE able to accurately learn and exploit a generative model conditioned on a *multi-set* of variables, addressing the applicative need of generating aggegated electrical consumption curves for a neighborhood given the set of households it contains. Its empirical validation on synthetic and real-world problems demonstrates its generality and its potential[8].

The CompVAE structure involves clearly separated observation and latent models – as generally advocated throughout the manuscript. A perspective for further study is thus to combine CompVAE with more sophisticated observation models (e.g. Wavenet or PixelCNN), and extend compositional generation to high-dimensional complex data.

Another perspective concerns the aggregation function Ψ used to combine the $\{W_i\}$ variables in CompVAE. All considered models rely on the sum (or a variation thereof); the rationale is to allow the inference model to infer the $\{W_i\}$s while controlling for their sum, using a multivariate Gaussian distribution. Other aggregation functions could be considered, e.g. an element-wise max, or a learned combination function built on invariant neural networks [RSP17a; BPC19]. How to adjust the inference model to capture the behavior of these aggregation functions opens the path for further research.

[8]Though a slight bias, interpreted as the amplification of the entity interdependencies, is observed on the electrical curves case study.

9.A Determinant of the covariance matrix

The sampling procedure is defined by:

$$w_i = \mu_i + \sigma_i \epsilon_i - \rho_i \sum_j \sigma_j \epsilon_j \tag{9.18}$$

Let $\mathbf{D}$ be the diagonal matrix $(\sigma_1, \sigma_2, \ldots, \sigma_K)$, $\mathbf{1}$ be the column vector $(1, 1, \ldots, 1)$, and $\mathbf{w}$, ρ, ϵ and μ be the column vectors defined by the $\{w_i\}$, $\{\rho_i\}$ $\{\epsilon_i\}$ and $\{\mu_i\}$ respectively. The sampling procedure can be reformulated using these as:

$$\mathbf{w} = \mu + \mathbf{D}\epsilon - \rho(\mathbf{1}^T \mathbf{D}\epsilon) = \mu + (\mathbf{I} - \rho \mathbf{1}^T)\mathbf{D}\epsilon \tag{9.19}$$

The covariance matrix is thus given as a product of four matrices:

$$\Sigma = (\mathbf{I} - \rho \mathbf{1}^T)\mathbf{D}\mathbf{D}(\mathbf{I} - \rho \mathbf{1}^T)^T \tag{9.20}$$

The determinant of $\mathbf{D}$ is $\prod_i \sigma_i$, remains the determinant of $(\mathbf{I} - \rho \mathbf{1}^T)$ to compute. This can be done using the matrix determinant lemma[9]:

$$|\mathbf{I} - \rho \mathbf{1}^T| = 1 - \mathbf{1}^T \rho = 1 - \sum_i \rho_i \tag{9.22}$$

Finally the determinant is given as:

$$|\Sigma| = \left(1 - \sum_i \rho_i\right)^2 \prod_i \sigma_i^2 \tag{9.23}$$

9.B Computing the KL divergence on $\{W_i\}$

In the context of the CompVAE model, $p_\theta(\{w_i\}|\{\ell_i\})$ factorizes as a product $\prod_i p_\theta(w_i|\ell_i)$, each term of this product being a normal distribution.

The KL divergence thus decomposes as (conditioning variables are ommited for brevity):

$$D_{KL}(q_\phi(\{w_i\}|\ldots)\|p_\theta(\{w_i\}|\ldots)) = -H(q_\phi(\{w_i\}|\ldots)) - \sum_i \mathop{\mathbb{E}}_{w_i \sim q_\phi} \log p_\theta(w_i|\ldots) \tag{9.24}$$

The entropy of the q_ϕ distribution is easily known from its covariance matrix:

$$H(q_\phi(\{w_i\}|\ldots)) = \frac{K}{2} + \frac{K}{2}\log 2\pi + \frac{1}{2}\log|\Sigma| \tag{9.25}$$

$$= \frac{K}{2} + \frac{K}{2}\log 2\pi + \log\left(1 - \sum_i \rho_i\right) + \sum_i \log \sigma_i \tag{9.26}$$

[9] For any invertible square matrix $\mathbf{A}$ and vectors $\mathbf{u}$ and $\mathbf{v}$:

$$|\mathbf{A} + \mathbf{u}\mathbf{v}^T| = (1 + \mathbf{v}^T \mathbf{A}^{-1}\mathbf{u})|\mathbf{A}| \tag{9.21}$$

To compute the second half of the KL, let m_i be the mean of $p_\theta(w_i|\ell_i)$, and s_i^2 be its variance. The log-density is thus expressed as:

$$\log p_\theta(w_i|\ell_i) = -\frac{(w_i - m_i)^2}{2s_i^2} - \log s_i - \frac{1}{2}\log 2\pi \tag{9.27}$$

The expectation is then computed using the sampling definition of w_i, and by averaging over the standard normal variables $\epsilon_1, \dots, \epsilon_K$:

$$\begin{aligned}
\mathop{\mathbb{E}}_{w_i \sim q_\phi} \log p_\theta(w_i|\dots) &= -\frac{1}{2s_i^2} \mathop{\mathbb{E}}_{\epsilon_1,\dots,\epsilon_K} \left(\mu_i + \sigma_i\epsilon_i - \rho_i \sum_j \sigma_j \epsilon_j - m_i\right)^2 - \log s_i - \frac{1}{2}\log 2\pi \\
&= -\frac{1}{2s_i^2}\left[(\mu_i - m_i)^2 + (1-\rho_i)^2\sigma_i^2 + \rho_i^2 \sum_{j\neq i} \sigma_j^2\right] - \log s_i - \frac{1}{2}\log 2\pi \\
&= -\frac{1}{2s_i^2}\left[(\mu_i - m_i)^2 + (1-2\rho_i)\sigma_i^2 + \rho_i^2 \sum_{j} \sigma_j^2\right] - \log s_i - \frac{1}{2}\log 2\pi
\end{aligned}$$

Putting all terms together, the final result is similar to the usual divergence between two diagonal normal distribution, with a few additional terms involving the ρ_i parameters:

$$\begin{aligned}
D_{KL}(\dots) = \sum_i \left[\frac{(\mu_i - m_i)^2 + (1-2\rho_i)\sigma_i^2 + \rho_i^2 \sum_j \sigma_j^2}{2s_i^2} + \log\frac{s_i}{\sigma_i}\right] \\
- \log\left(1 - \sum_i \rho_i\right) - \frac{K}{2}
\end{aligned} \tag{9.28}$$

Chapter 10
Latent manipulation from Boltzmann principles

Contents

This chapter presents the third main contribution of the manuscript, concerned with the control of the generative process. Many related works [Mat+16; RMC16; Lar+16; Che+18a; Moy+18; Mat+19b] are based on the explicit interpretation and control of the latent variables. Another approach is proposed here, where the latent distribution is altered using an explicit loss on the observed variables, as this second formulation is more in line with some problems. A first version of the approach, named *Boltzmann Tuning of Generative Models (BTGM)*, was presented in [BS21]. This chapter describes an extended version of BTGM.

Considering a given, already trained generative model $p(x)$, and an external criterion $f : \mathcal{X} \to \mathbb{R}$, the goal is to refine the model and define a new model $\hat{p}(x)$ that maximizes $\mathbb{E}_{x\sim\hat{p}} f(x)$ while remaining as close to p as possible. This goal, akin model fine-tuning, is formalized as a tractable optimization problem, and a sound procedure is proposed to tackle it.

This problem is also motivated by an application from the smart energy control domain, and specifically the estimation of the consumption peak. More formally, in the same context as in the previous chapter, the goal is to generate extreme though realistic electrical consumption curves; these curves will be used to evaluate the behavior of the electrical grid in specific scenarii and the associated risk attached.

BTGM proposes a general answer to the above problem. At an abstract level, the point is to bias the generative model toward the region of extreme values w.r.t. the considered external criterion.

10.1 Boltzmann distributions and Pareto exploration

A most appropriate framework to express constraints in a probabilistic setting is the celebrated Principle of Maximum-Entropy (MaxEnt), originated from statistical physics [Jay57] and now widely used in Machine Learning. In Bayesian terms, this principle concerns the appropriate choice of a prior distribution, as being the one with largest entropy among the admissible distributions (complying with the constraints).

10.1.1 Principle of maximum (relative) entropy

Most considered constraints either restrict the support of the distribution, or require the expectation of some function over the distribution take a given value: $\mathbb{E}\, f = C$. In both cases, the MaxEnt principle can be formulated as a constrained optimization problem:

$$\arg\max_{\hat{p}} H(\hat{p}) \quad \text{s.t.} \quad \forall i : \mathop{\mathbb{E}}_{x\sim\hat{p}} f_i(x) = C_i \tag{10.1}$$

This constrained optimization problem is solved by using Lagrange multipliers [Jay57], yielding a solution of the form (complete derivation in Appendix 10.A):

$$\hat{p}(x) = \frac{1}{Z} \exp\left(\sum_i \lambda_i f_i(x)\right) \tag{10.2}$$

where the values of the Lagrange multipliers λ_i depend on constant C_is, and the normalization constant Z also is a function of the Lagrange multipliers.

Interestingly though unsurprisingly, energy-based models (Section 1.1.2) take the same functional form. Both approaches take inspiration from statistical physics: In both cases, the principle of maximum entropy of the distribution over the physical states of the studied system, subject to the expectation of the total energy taking a known value [Jay57], yields the Boltzmann distribution (where the inverse temperature plays the role of the Lagrange multiplier).

The principle of maximum entropy has been further generalized by [Cat07], combined with Bayes rule to define the *principle of maximum relative entropy*. This latter principle states that, given a prior distribution $p(x)$, the refinement of p subject to a set of constraints of the form $\mathbb{E}\, f = C$ is properly formulated through minimizing the KL-divergence of the sought distribution $\hat{p}$ and p (or *relative entropy* of $\hat{p}$):

$$\arg\min_{\hat{p}} D_{KL}(\hat{p}\|p) \quad \text{s.t.} \quad \forall i : \mathop{\mathbb{E}}_{x\sim\hat{p}} f_i(x) = C_i \tag{10.3}$$

This new problem can be solved in the same way as MaxEnt, and yields a similar solution:

$$\hat{p}(x) = \frac{p(x)}{Z} \exp\left(\sum_i \lambda_i f_i(x)\right) \tag{10.4}$$

10.1.2 Exploration of the Pareto front

This formulation interestingly allows for exchanging the roles of the optimization objective and the constraints, up to a reparameterization of the Lagrange multipliers. In short, optimizing the (relative) entropy under the constraint that the expectation of a function f is fixed to a constant value, is equivalent to maximizing $\mathbb{E} f$ under the constraint of a given value for the (relative) entropy[1]. More generally, the Lagrange multipliers can be interpreted as a parameterization of the Pareto front corresponding to the multi-objective optimization of the various f_i criteria and the (relative) entropy. This Pareto front, describing the possible trade-offs between the considered objectives, can be explored via varying the λ_is.

This exploration is illustrated on a simple case with a single criterion $f(x)$. Let $\hat{p}_\lambda$ be the distribution defined as:

$$\hat{p}_\lambda(x) = \frac{p(x)}{Z(\lambda)} e^{\lambda f(x)} \tag{10.5}$$

Then, the local behavior of the criteria seen as functions of λ can be expressed using statistical quantities evaluated over $\hat{p}_\lambda$, as follows:

$$\frac{d}{d\lambda} D_{KL}(\hat{p}_\lambda \| p) = \lambda Var_{\hat{p}_\lambda}\Big(f\Big) \tag{10.6}$$

$$\frac{d}{d\lambda} \mathop{\mathbb{E}}_{\hat{p}_\lambda} f = Var_{\hat{p}_\lambda}\Big(f\Big) \tag{10.7}$$

In particular, when restricted to $\lambda > 0$, both $D_{KL}(\hat{p}_\lambda \| p)$ and $\mathbb{E}_{\hat{p}_\lambda} f$ are monotonic functions of λ. The $\lambda < 0$ case is symmetrically handled (by minimizing f instead of maximizing it). This confirms that exploring the different values of λ reliably explores the Pareto front balancing these two objectives.

Similar identities hold for second derivatives (with proofs in Appendix 10.B):

$$\frac{d^2}{d\lambda^2} D_{KL}(\hat{p}_\lambda \| p) = Var_{\hat{p}_\lambda}\Big(f\Big) + \lambda \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f \right)^3 \tag{10.8}$$

$$\frac{d^2}{d\lambda} \mathop{\mathbb{E}}_{\hat{p}_\lambda} f = \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f \right)^3 \tag{10.9}$$

From the above equations, it follows that if the first, second, and third moments of f under $\hat{p}_\lambda$ can be empirically estimated (assuming that one can sample the instance space according to $\hat{p}_\lambda$), these estimates can be used to efficiently explore the said Pareto front.[2].

The sought approximation of $\hat{p}_\lambda$ can be found using variational inference (presented in Section 2.3); it reads:

$$\hat{p}_\lambda = \arg\min_{\hat{p}} \; D_{KL}(\hat{p} \| \hat{p}_\lambda) \tag{10.10}$$

[1] Both associated Lagrangians (see details in Appendix 10.A) are very similar, the only difference being whether the Lagrange multiplier λ applies on f or of the KL-divergence. As a result, they yield the same solution, up to a reparameterization of λ into $1/\lambda$.

[2] In the case where more than one f_i criterion is considered, the approach can be extended to characterize in closed form the gradient of the criteria w.r.t. the $(\lambda_i)_i$ vector, as well as their Hessian matrix.

Then, replacing $\hat{p}_\lambda$ with its formal definition (Equation 10.5) and dropping the constant terms:

$$\hat{p}_\lambda = \arg\max_{\hat{p}} \; H(\hat{p}) + \mathop{\mathbb{E}}_{x\sim\hat{p}} \left[\lambda f(x) + \log p(x)\right] \tag{10.11}$$

This optimization problem can be solved within the variational inference framework, depending on the considered f and p. This optimization can be combined with a second-order method to efficiently find the value of λ corresponding to a given target value of $D_{KL}(\hat{p}_\lambda \| p)$ or $\mathbb{E}_{\hat{p}_\lambda} f$. The approach is described in Algorithm 10.1 and Algorithm 10.2.

Algorithm 10.1 Maximizing $\mathbb{E}_{\hat{p}_\lambda} f$ subject to $D_{KL}(\hat{p}_\lambda \| p)$

$D \leftarrow$ Target value for $D_{KL}(\hat{p}_\lambda \| p)$
$\lambda \leftarrow 0$
repeat
  $\hat{p}_\lambda \leftarrow \arg\max_{\hat{p}} \; H(\hat{p}) + \mathbb{E}_{x\sim\hat{p}} \left[\lambda f(x) + \log p(x)\right]$ (Equation 10.11)
  Estimate $D_{KL}(\hat{p}_\lambda \| p)$ by Monte-Carlo
  Estimate $\frac{d}{d\lambda} D_{KL}(\hat{p}_\lambda \| p)$ using Equation 10.6 by Monte-Carlo
  Estimate $\frac{d^2}{d\lambda^2} D_{KL}(\hat{p}_\lambda \| p)$ using Equation 10.8 by Monte-Carlo
  $\lambda \leftarrow$ Second order update to bring $D_{KL}(\hat{p}_\lambda \| p)$ towards D
until convergence of λ
return $\hat{p}_\lambda$

Algorithm 10.2 Minimizing $D_{KL}(\hat{p}_\lambda \| p)$ subject to $\mathbb{E}_{\hat{p}_\lambda} f$

$F \leftarrow$ Target value for $\mathbb{E}_{\hat{p}_\lambda} f$
$\lambda \leftarrow 0$
repeat
  $\hat{p}_\lambda \leftarrow \arg\max_{\hat{p}} \; H(\hat{p}) + \mathbb{E}_{x\sim\hat{p}} \left[\lambda f(x) + \log p(x)\right]$ (Equation 10.11)
  Estimate $\mathbb{E}_{\hat{p}_\lambda} f$ by Monte-Carlo
  Estimate $\frac{d}{d\lambda} \mathbb{E}_{\hat{p}_\lambda} f$ using Equation 10.7 by Monte-Carlo
  Estimate $\frac{d^2}{d\lambda^2} \mathbb{E}_{\hat{p}_\lambda} f$ using Equation 10.9 by Monte-Carlo
  $\lambda \leftarrow$ Second order update to bring $\mathbb{E}_{\hat{p}_\lambda} f$ towards F
until convergence of λ
return $\hat{p}_\lambda$

In both algorithms, the computational cost is dominated by the optimization of Equation 10.11. The Monte-Carlo estimation of the objective value and its two derivatives with a large number of samples is comparatively inexpensive; hence the estimation error comes from the error on $\hat{p}_\lambda$ itself (not from the Monte-Carlo estimation).

10.2 The Boltzmann tuning of Generative Models

The above general principles aimed to finding $\hat{p}_\lambda$ from p can be adapted in the deep LVM framework, defining our third contribution, the *Boltzmann Tuning of*

Generative Models (BTGM). This algorithm (Algorithm 10.1 & 10.2) involves an inner loop (finding $\hat{p}_\lambda$ for a given value of λ and an outer loop, adjusting the value of λ complying with the constraint w.r.t. the current p_λ, until convergence of λ.

10.2.1 Generalizing to multiple variables

When transposing the optimization problem from Equation 10.1 to a LVM, a first issue regards the definition of the relevant variables, and specifically which variables should be marginalized before computing the KL-divergence between $\hat{p}_\lambda$ and p.

Let us consider the case of distribution $p(x, z)$ with X the observed variable and Z an abstract latent variable. The quantity of interest clearly is the distribution $p(x)$ over the observed variable.
One would thus like to minimize $D_{KL}(\hat{p}_\lambda(x)\|p(x))$ (as opposed to, $D_{KL}(\hat{p}_\lambda(x, z)\|p(x, z))$, that is in general different). However, the optimization of $D_{KL}(\hat{p}_\lambda(x, z)\|p(x, z))$ is straightforward to estimate from the LVM, while $D_{KL}(\hat{p}_\lambda(x)\|p(x))$ is much more expensive, as it requires marginalizing out the latent variable Z.
The question thus becomes of comparing both optimization problems (marginalized and non marginalized) and estimate the cost of the non-marginalized approximation.

Formally, the KL-divergence can be decomposed as:

$$D_{KL}(\hat{p}_\lambda(x, z)\|p(x, z)) = D_{KL}(\hat{p}_\lambda(x)\|p(x)) + \mathop{\mathbb{E}}_{x\sim\hat{p}_\lambda} D_{KL}(\hat{p}_\lambda(z|x)\|p(z|x)) \quad (10.12)$$

The non marginalized KL-divergence thus is a higher-bound of the marginalized one. Accordingly, minimizing the non-marginalized KL-divergence also brings the marginalized down, but the tightness of the bound is unknown and there is no guarantee regarding the difference between both optimal solutions. In the marginalized optimization problem, the model can flexibly reorganize the latent space as it does not constrain in any way the distribution of the latent variable Z. Quite the contrary, the non-marginalized optimization problem directly takes into account the latent space: it forces both $p_\lambda(z)$ to remain similar to $p(z)$ and $p_\lambda(x|z)$ to remain similar to $p(x|z)$.

The lack of flexibility of the latter optimization problem is however not necessarily a drawback, and might actually be beneficial to the BTGM implementation. In most cases, distribution p is obtained as the (imperfect) result of a training process. Searching for $\hat{p}_\lambda$ while drastically modifying the latent structure of p thus amounts to training from scratch a new model distilling p. On the one hand, this optimization is very costly; on the other hand, it can amplify the errors in p.

The alternative is to only learn a new latent distribution $\hat{p}_\lambda(z)$, while keeping the "decoder" part of the model $p(x|z)$ frozen, where $\hat{p}_\lambda$ is obtained as:

$$\hat{p}_\lambda(x, z) = p(x|z)\hat{p}_\lambda(z) \quad (10.13)$$

The characteristics of the data, what makes p "realistic", are essentially captured in $p(x|z)$, where the latent variable Z only dictates the data region to be sampled from. The only modification of the Z distribution thus ensures that the tuned model $\hat{p}_\lambda$ remains within the domain of realistic data (as sampled by p), and focuses on the appropriate subdomain according to criterion f. Furthermore, the KL-divergence

simplifies when focusing on the only Z distribution optimization:

$$D_{KL}(\hat{p}_\lambda(x,z)\|p(x,z)) = D_{KL}(\hat{p}_\lambda(z)\|p(z)) + \mathop{\mathbb{E}}_{z\sim\hat{p}_\lambda} \underbrace{D_{KL}(p(x|z)\|p(x|z))}_{=0} \tag{10.14}$$

Overall, conducting the optimization on the two variable model $p(x,z)$ with criterion $f(x)$ is equivalent to conducting the optimization on the single-variable model $p(z)$ with criterion $\widetilde{f}(z) = \mathbb{E}_{x\sim p(x|z)} f(x)$.[3]

This reduction enables to exploit models with a deterministic "decoder", such as GANs. It is however still necessary to evaluate the tightness of the upper bound (Equation 10.12).

Note that even though $\hat{p}_\lambda(x|z) = p(x|z)$, this does not imply in general that $\hat{p}_\lambda(z|x) \approx p(z|x)$. After Bayes theorem,

$$\frac{\hat{p}_\lambda(z|x)}{p(z|x)} = \frac{\hat{p}_\lambda(z)}{\hat{p}_\lambda(x)}\frac{p(x)}{p(z)} \tag{10.15}$$

In order to enforce the tightness of the upper bound (Equation 10.12), that is, ensure $\mathbb{E}_{x\sim\hat{p}_\lambda} D_{KL}(\hat{p}_\lambda(z|x)\|p(z|x))$ be close to 0, then the ratio needs to be most of the time close to 1 when $(x,z)\sim\hat{p}_\lambda$:

$$\forall(x,z)\sim\hat{p}_\lambda : \quad \frac{\hat{p}_\lambda(z)}{\hat{p}_\lambda(x)} \approx \frac{p(z)}{p(x)} \tag{10.16}$$

In the case where x is a quasi-deterministic and invertible function of z, the change of probability from p to $\hat{p}_\lambda$ can be mostly controlled by the change of probability of z.

In the VAE case, the use of a quasi-deterministic observation model (Section 6.2) falls in the above category; the mapping from z to x can be made quasi-invertible by using a restricted probabilistic inference model.[4]

In the GAN case, on the one hand they do provide a deterministic mapping; on the other hand, there is *a-priori* no guarantees for this mapping to be invertible; hence, the use of a GAN as base p model might result in a larger gap between $D_{KL}(\hat{p}_\lambda(x,z)\|p(x,z))$ and $D_{KL}(\hat{p}_\lambda(x)\|p(x))$.

Likewise, complex observation models such as PixelCNN (Section 5.2.2) neither provide quasi-determinism nor are quasi-invertible, and thus they are *a priori* poor candidates as base models for BTGM.

When considering a multi-variable deep LVM (with more than two variables), the same methodology can be followed, selecting only some variables X_is of the model to be tuned while the total KL-divergence reads as a sum of KL-divergences over these variables (taking the KL expectation w.r.t. their parent variables noted $\pi(X_i)$):

$$D_{KL}(\hat{p}_\lambda\|p) = \sum_i \mathop{\mathbb{E}}_{\pi(X_i)} D_{KL}(\hat{p}_\lambda(x_i|\pi(X_i))\|p(x_i|\pi(X_i))) \tag{10.17}$$

[3] The criterion f under $\hat{p}_\lambda$ reads $\mathbb{E}_{(x,z)\sim\hat{p}_\lambda} f(x)$. Under the assumption $\hat{p}_\lambda(x,z) = p(x|z)\hat{p}_\lambda(z)$, this can be decomposed as $\mathbb{E}_{z\sim\hat{p}_\lambda}\mathbb{E}_{x\sim p(x|z)} f(x)$. The latter is expressed as the expectation of the alternative criterion $\widetilde{f}(z) = \mathbb{E}_{x\sim p(x|z)} f(x)$ under $\hat{p}_\lambda(z)$, effectively moving the decoder part of the model into the optimization criterion.

[4] If the inference model is limited to be single mode, e.g. Gaussian, then the generative model $p_\theta(x,z)$ is driven to learn a latent representation compatible with this inference model: such that $p_\theta(z|x)$ has a single mode as well (Section 3.4), therefore x is a quasi-invertible function of z.

Does approximating the KL-divergence over observed variables by the full KL-divergence over all variables entail a performance loss? The answer thus depends on whether the observed variables depend on the rest of the model in a quasi-deterministic and quasi-invertible way. In the positive case, the tightness of the upper bound is good after Equation 10.12 (the second term in right hand side is small); otherwise there is no guarantee.

10.2.2 Using normalizing flows

A key issue is to define a good search space for $\hat{p}_\lambda$. For simplicity, let us focus on the single-variable case; the multi-variable case is handled in the same way. While any variational inference method can be used in principle, the choice of flows-based methods (Section 2.3.3) appear to be appropriate: even though $\hat{p}_\lambda$ can be a very complex distribution, its support is bound to be within the support of p for the sake of the realism of the generated samples.

Using normalizing flows (Section 2.3.3) to transform samples from p into samples of another distribution $\hat{p}$ is thus a natural approach. It provides a significant flexibility, while the values of $\hat{p}$ can still be computed in closed form. Letting g denote the learned flow, x a sample from p and $\hat{x} = g(x)$, the generated sample from $\hat{p}$, then it comes:

$$\hat{p}(\hat{x}) = p(x)|J(g)(x)|^{-1} \tag{10.18}$$

with $|J(g)(x)|$ the determinant of the Jacobian matrix of g in x. The optimization problem defined in Equation 10.11 then is rewritten as an optimization on g:

$$\arg\max_g \mathop{\mathbb{E}}_{x\sim p} \left[\lambda f(g(x)) + \log p(g(x)) + \log |J(g)(x)|\right] \tag{10.19}$$

In practice, to ensure a proper coverage of the space and a good exploration of the modes of $\hat{p}_\lambda$, the flow model is initialized close to the identity function, ensuring that the training samples cover the whole support of p from the start.

As said, the approach involves an inner optimization loop (finding p_λ) and an outer loop (adjusting λ). Using a flow-based approximation in the frame of Algorithm 10.1 or Algorithm 10.2 enables a warm-start heuristics in the inner loop: re-using the approximation of g from outer iteration $t-1$ to warm-start g at iteration t. As long as the estimated value of λ increases along the outer loop of optimization, the new target distribution becomes more concentrated around its modes, making the previous one a decent initialization. If λ decreases this strategy is however more risky: one or more modes of the distribution may become eligible, not necessarily covered by the previous approximation. In that case, the safe solution consists in initializing g to identity, or to a previous approximation trained with a lower λ value.

The experiments presented in Section 10.3 rely on this warm-start procedure.

10.2.3 Comparing and selecting the f criterion

A key step in the setup of BTGM is the choice of the criterion f to be maximized. Informally, the expert's requirements can be represented in many different ways, and same solutions should be found when using f or hof for any monotonous scalar function h. The difficulty of the underlying optimization problem however varies

depending on which h is chosen. Let us examine how BTGM behaves depending on f.

Note that the Pareto front defined by the family $\hat{p}_\lambda$ when λ ranges in $\mathbb{R}$ is unaffected when f is composed with any affine h ($h(x) = ax + b$, with $a > 0$). Varying constant b amounts to changing the normalization constant $Z(\lambda)$, and varying coefficient a amounts to changing λ, leaving the Pareto front unchanged. An affine normalization of f is based on this remark: in the remainder, it is assumed with no loss of generality that the expectation of f under the base distribution p is 0, and its variance is 1.

A given f (after the affine normalization) is assessed from the expected difficulty of learning a good approximation of $\hat{p}_\lambda$ with f. Intuitively, if $\hat{p}_\lambda$ involves sharp and intricate boundaries, the task is more difficult. Letting g denote the transformation from p to $\hat{p}_\lambda$, then the determinant $|J(g)|$ of its Jacobian matrix gives a quantitative estimate of how much g needs to warp its working space to transform samples from p into samples from $\hat{p}_\lambda$. The variations of this determinant across the space thus provide a good indicator of the complexity of this transformation.

More formally, the complexity of g at its best (corresponding to an optimal $\hat{p}_\lambda$) can be computed from its theoretical value given by Equation 10.5. The gradient of the logarithm of its determinant is given as:

$$\nabla_x \log |J(g)|^{-1} = \nabla_x \log \frac{\hat{p}_\lambda(x)}{p(x)} = \lambda \nabla_x f(x) \tag{10.20}$$

The difficulty of learning $\hat{p}_\lambda$ for a criterion f is thus given *a priori* by considering the distribution of the gradient of f under the base distribution p. Therefore, two criteria f_1 and f_2 can be compared based on the histograms of their normalized gradients: $\|\nabla_x f(x)\| / \sqrt{Var_p(f)}$, as will be illustrated in Section 10.3.1.

10.3 Case studies

This section illustrates the use of BTGM on three case studies:
In the first one, the criterion f is an externally-trained classifier, and the goal is to bias a generative model to oversample the selected class(es), allowing one to finely control the strength of the bias.
In the second one, BTGM is applied in the context of smart energy policies, continuing the application described in Section 9.3.3; the initial goal is to produce curves and facilitate the estimation of the peak consumption; more generally, one want to produce extreme-but-realistic consumption curves according to some metrics.
The third case study investigates how to use BTGM to combine an overly general generative model (as trained by a VAE) with a discriminator network (as used in GANs) to yield a more realistic distribution sample.

10.3.1 Case 1: Conditioning a distribution

Given a generative model $p(x)$, BTGM is applied to create a conditioned version of it. Let us assume for instance that some classifier $c(y|x)$ is provided, then we might use it to condition *a posteriori* the generative model and e.g., approximate $p(x|y)$ for a chosen class y.

In order to do so, one might consider two criteria, commonly associated to probabilistic classifiers: $f(x) = c(y|x)$ and $f(x) = \log c(y|x)$. From an analytical point of view, the second criterion (log-probability of the class) seems to be more appropriate, yielding the theoretical distribution (after Equation 10.5):

$$\hat{p}_\lambda \propto p(x)\, c(y|x)^\lambda \tag{10.21}$$

If classifier c accurately approximates $p(y|x)$, then setting $\lambda = 1$ gives $\hat{p}_\lambda \approx p(x|y)$ after Bayes theorem. Depending on the value of λ, one might control the strength of the bias, from a slight oversampling of class y ($\lambda < 1$) to a strong preference for examples unambiguously classified into class y ($\lambda > 1$).

Criterion $f(x) = c(y|x)$ does not yield such a principled theoretical solution. However, as discussed in Section 10.2.3, another issue is whether one or the other criterion induces a tractable optimization problem. This question is experimentally investigated as follows. We trained a classifier on the MNIST dataset; the distributions of $\|\nabla_x c(y|x)\|$ and $\|\nabla_x \log c(y|x)\|$ for each of the 10 classes are computed, using a trained VAE as base distribution p. The histograms, estimated from 10.000 samples, are presented as Figure 10.1.

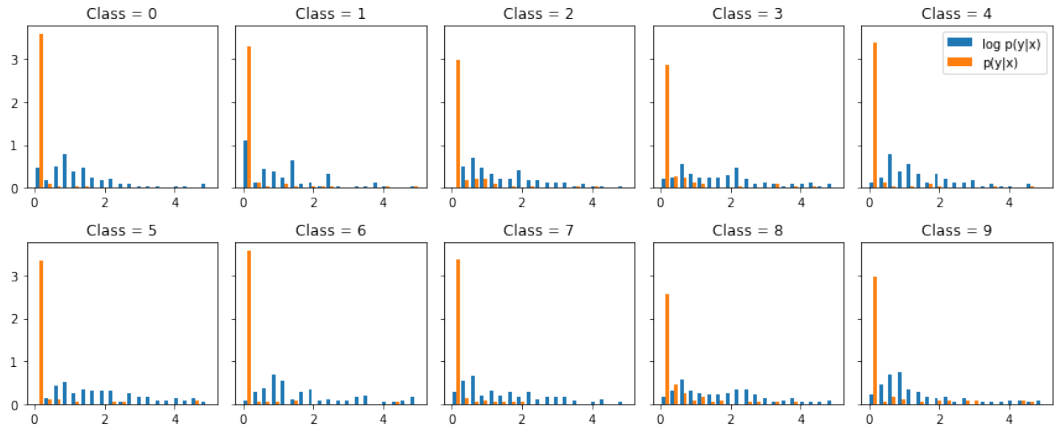


Figure 10.1: Comparing criterion $f(x) = c(y|x)$ (in blue) and $f(x) = \log c(y|x)$ (in orange) on MNIST: binned distribution of their gradient norms. The distribution tails are truncated for the sake of visualization.

In the latter case ($f(x) = c(y|x)$), Figure 10.1 (left) shows a large bar at 0 (for large regions of the space, the gradient norm is close to 0), and values up to 100 are observed (the histogram is truncated for readability). The optimization landscape defined by $f(x) = c(y|x)$ thus expectedly involves large plateaus (regions where the gradient norm is close to 0) separated by large cliffs (regions where the gradient norm is very high). While this landscape reflects the behavior of a sharp classifier, it is expected to define a hard optimization problem for the normalizing flow, as it requires the learned transformation to widely vary at the boundaries among classes (following Equation 10.20). On the opposite, the former case ($f(x) = logc(y|x)$) involves few plateaus (regions with gradients close to 0) and no abrupt transitions (the gradient norm remains less than 15), suggesting a much smoother optimization landscape consisting of hills and valleys.

The above remarks are empirically confirmed by building $\hat{p}_\lambda$ for various values of λ. Figure 10.2 reports $D_{KL}(\hat{p}_\lambda \| p)$ versus $\mathbb{E}_{\hat{p}_\lambda} f$, comparing the empirical value

(in orange) and the theoretical value (in blue) for both $f = c(y|x)$ (Figure 10.2, left) and $f = logc(y|x)$ (Figure 10.2, right). The theoretical values are numerically computed by sampling p and using the following formulas (derivations detailed in Appendix 10.C):

$$\mathop{\mathbb{E}}_{\hat{p}_\lambda} f = \frac{\mathbb{E}_p\, f e^{\lambda f}}{\mathbb{E}_p\, e^{\lambda f}} \tag{10.22}$$

$$D_{KL}(\hat{p}_\lambda \| p) = \frac{\mathbb{E}_p\, f e^{\lambda f}}{\mathbb{E}_p\, e^{\lambda f}} - \log \mathop{\mathbb{E}}_{p} e^{\lambda f} \tag{10.23}$$

As could have been expected, the empirical curves do not perfectly match the theoretical curves. However, the gap is significantly lower when using $f = log(c(y|x))$[5]. More specifically, for low λ values, $\hat{p}_\lambda$ tends to remain close to p; as λ increases, it quickly overshoots the theoretical D_{KL}. This behavior is consistent with an optimization landscape composed of plateaus and abrupt cliffs: for low λ the landscape is mostly flat, and the few small cliffs are missed, and when λ increases only the high plateaus are kept and the rest of the distribution support is suddenly discarded.

In contrast, the evolution of the distribution for $f(x) = log(c(y|x))$ is smoother and more gradual, closer to the theoretical values, thus confirming the expectations based on the regularity of its gradient norm.

The generated samples are illustrated on Figure 10.4 using $f(x) = \log c(y = 4|x)$.

From top to bottom, the λ value increases; each row displays a set of samples from $\hat{p}_\lambda$. As λ increases, the prevalence of the digit 4 increases in the generated samples, and non-4 digits are mostly chosen in the class 9, that most look alike 4. For $\lambda \geq 1$, only 4 samples are generated; for $\lambda = 1.25$, the distribution focuses on the most unambiguous 4 samples.

10.3.2 Case 2: Extreme values of a distribution

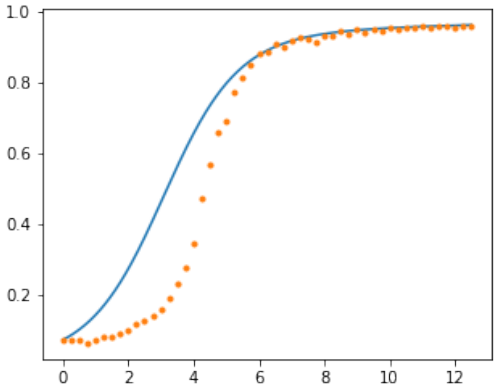
The motivating application used in CompVAE (Chapter 9) is also considered for BTGM.[6] The presented results are based on a regular VAE trained on electrical curves, aggregating 10 households. The rationale for considering a small number of households is to enforce the variability of the generated curves, and produce examples that are visually compelling. The same model, trained on larger aggregations, produces "extreme" curves that are less easily interpretable due to the lower global variance of the dataset.

BTGM is applied on a pre-trained VAE by learning a normalizing flow in its latent space, as described in Section 10.2.1 and Section 10.2.2. The generation of electrical consumption curves considers the same objective functions as presented in Section 9.3.3: the peak value, the flexibility, and the sustained maximum value over 8 hours, characterized as follows (where $x = (x_1, x_2, \dots x_T)$ denotes the generated curve):

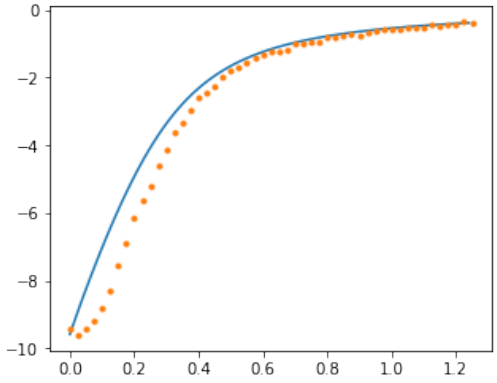
$$f_{\text{peak}}(x) = \max_i(x_i) \tag{10.24}$$

[5]When using the classification probability, the maximum gap between the predicted and reached values for the KL-divergence is 0.58, while it is of only 0.17 when using the log-probability (the values of f are not comparable as they are on different scales).

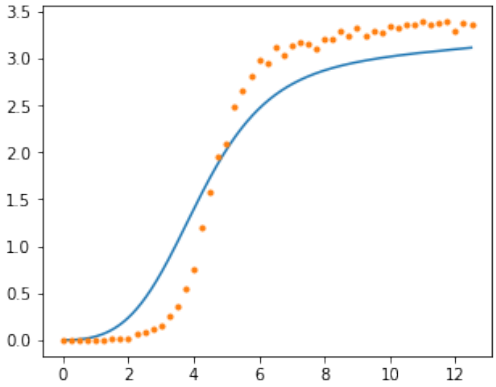
[6]The full integration of CompVAE and BTGM is not achieved at time of writing.



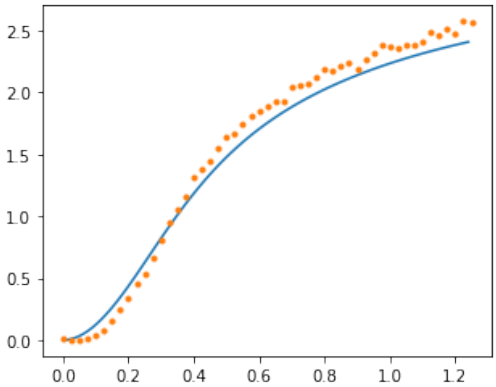
(a) $\mathbb{E}_{\hat{p}_\lambda} f$ as a function of λ using $c(y|x)$.



(b) $\mathbb{E}_{\hat{p}_\lambda} f$ as a function of λ using $\log c(y|x)$.

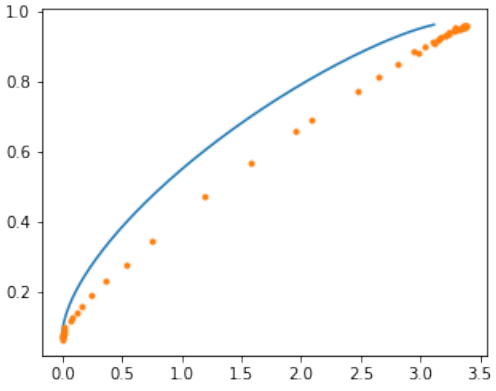


(c) $D_{KL}(\hat{p}_\lambda \| p)$ as a function of λ using $c(y|x)$.

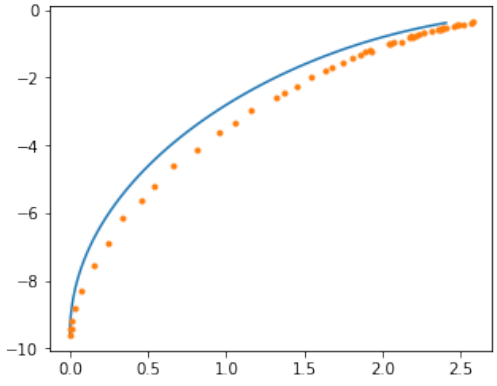


(d) $D_{KL}(\hat{p}_\lambda \| p)$ as a function of λ using $\log c(y|x)$.

Figure 10.2: Comparing the theoretical (in blue) and empirical (in orange) values of the criteria. Left: $f(x) = c(y|x)$. Right: $f(x) = \log c(y|x)$.



(a) $\mathbb{E}_{\hat{p}_\lambda} f$ as a function of $D_{KL}(\hat{p}_\lambda \| p)$ using $c(y|x)$.



(b) $\mathbb{E}_{\hat{p}_\lambda} f$ as a function of $D_{KL}(\hat{p}_\lambda \| p)$ using $\log c(y|x)$.

Figure 10.3: Representation of the theoretical (blue lines) and empirical (orange dots) Pareto front between both criteria, depending on the choice of f, using either $f(x) = c(y|x)$ (left) or $f(x) = \log c(y|x)$ (right).

Figure 10.4: Samples generated from $\hat{p}_\lambda$ biasing a generative model trained on MNIST towards the class 4(using $f(x) = \log c(y = 4|x)$). The top to bottom rows respectively correspond to $\lambda = \{0.0, 0.25, 0.5, 0.75, 1.0, 1.25\}$. In each row are presented samples from $\hat{p}_\lambda$: $\hat{p}_\lambda = p$ for $\lambda = 0$ (top row) and the bottom row corresponds to the theoretical conditional model ($\lambda = 1$).

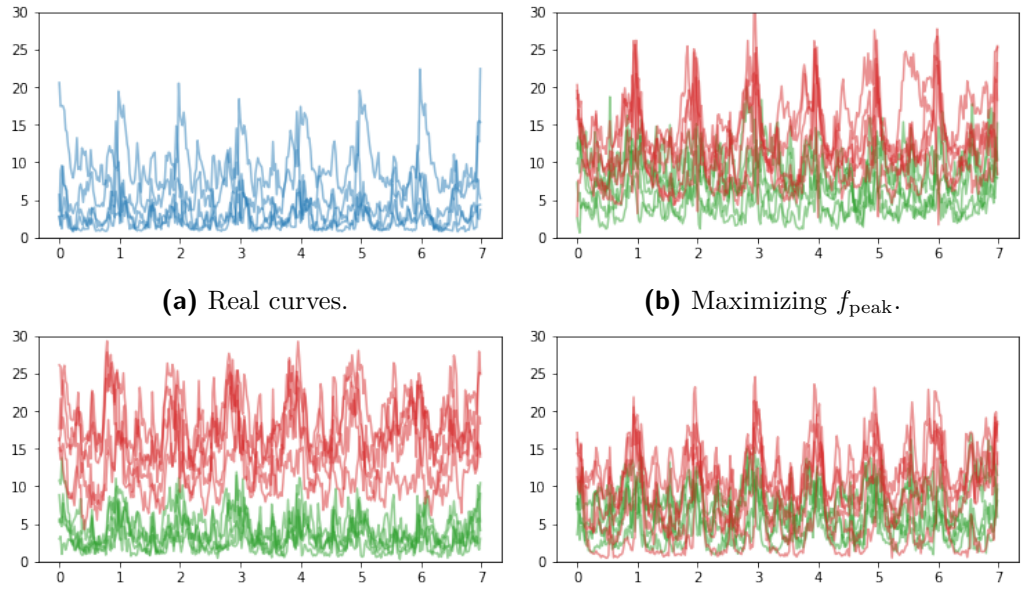
$$f_{\text{sustained}}(x; k) = \underset{i}{\text{top-}k}(x_i) \tag{10.25}$$

$$f_{\text{flexibility}}(x) = \sum_i \left| x_i - \frac{1}{T} \sum_j x_j \right| \tag{10.26}$$

In order to generate visually extreme cases, the target value for D_{KL} is fixed to $-\log(0.001) \approx 6.91$, corresponding to the top 0.1% curves w.r.t. the chosen criterion. In all cases the process converged in less than 10 updates of λ, following Algorithm 10.1. The resulting tuned distributions are shown in Figure 10.5, on which the respective impact of each criterion is clearly visible. The maximization of f_{peak} (Figure 10.5(b)) selects curves that are slightly above average, but with very high short peaks (even going beyond 30kW). In contrast, maximizing $f_{\text{sustained}}$ does not produce as high peaks, but selects curves with higher average value, as desired. Finally, maximizing $f_{\text{flexibility}}$ selects curves with high variance, with periods of both very high and very low consumption. In all cases, the generated curves still have the general characteristic of real curves (such as the daily pattern) and do present significant variety, in concordance with BTGM objective of remaining as close to the base generative model as possible.

10.3.3 Case 3: Fine-tuning a generative model

Another usage of BTGM aims to fine-tuning generative models to improve their generation quality. As discussed in Section 3.2, deep LVMs such as VAEs tend to learn overly general distributions due to the limited capacity of their inference models and their maximum likelihood training [AB17]. Given such a model, BTGM



(a) Real curves.

(b) Maximizing f_{peak}.

(c) Maximizing $f_{\text{sustained}}$.

(d) Maximizing $f_{\text{flexibility}}$.

Figure 10.5: Generation of electrical consumption curves: original curves (in blue), curves generated from the base distribution (in green) and from the biased distribution (in red). (a): Original curves. (b): Curves biased toward high consumption peak. (c): Curves biased toward sustained consumption (over 8 hours, thus biased toward the top 16 highest consumption. (d) Curves biased toward high flexibility. In each case, 5 curves are sampled and superposed to illustrate the variety of the generation.

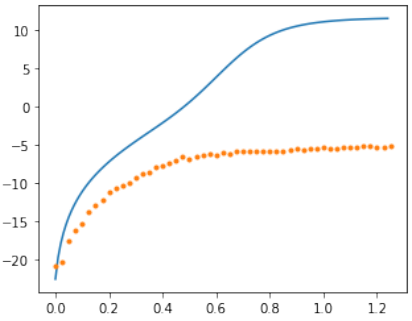
proceeds to refine its latent distribution and better stick to realistic samples using an adversarial mechanism *a posteriori* [Goo+14].

A standard GAN classifier (referred to as discriminator) discriminates among generated samples (drawn after a generative model p_G) and a real dataset p_D. When trained to optimality, the discriminator thus provides an approximation of $\frac{p_D(x)}{p_G(x)+p_D(x)}$. The pre-sigmoid output of the discriminator is thus itself an approximation[7] of $\log \frac{p_D(x)}{p_G(x)}$, making it an interesting candidate for a BTGM f criterion. Using p_G as base distribution, the optimized distribution produced by BTGM reads:

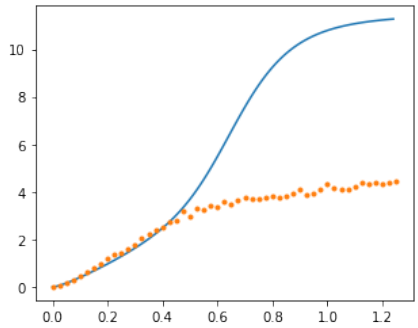
$$\hat{p}_\lambda(x) \propto p_G(x)^{1-\lambda} p_D(x)^\lambda \tag{10.27}$$

Distribution $\hat{p}_\lambda$ gets closer to the data distribution p_D as λ increases, with theoretically $\hat{p}_\lambda = p_D$ for $\lambda = 1$. In practice it is unlikely to fully recover p_D, but one might expect to decently tighten an overly general distribution p_G. Note that the use of a GAN discriminator within BTGM enables to decouple the training of the modules: the discriminator is trained while p_G is frozen, and BTGM is then launched to learn $\hat{p}_\lambda$ with both p_G and the discriminator frozen: this mechanism avoids the concurrent adversarial training of the modules, sidestepping the notorious stability issues of the GANs [AB17].

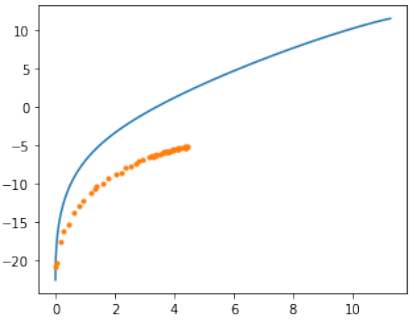
[7] Let y denote the pre-sigmoid activation of the discriminator, then $sigmoid(y) \approx \frac{p_D(x)}{p_G(x)+p_D(x)}$. Solving for y yields $y \approx \frac{p_D(x)}{p_G(x)}$.



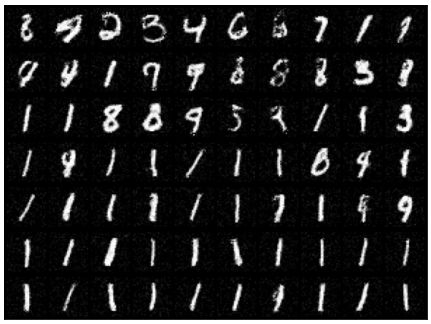
(a) $\mathbb{E}_{\hat{p}_\lambda} f$ (y axis) vs λ (x axis).



(b) $D_{KL}(\hat{p}_\lambda \| p)$ (y axis) vs λ (x axis).



(c) $\mathbb{E}_{\hat{p}_\lambda} f$ (y axis) vs $D_{KL}(\hat{p}_\lambda \| p)$ (x axis).



(d) Generated samples, with the strength λ of the bias increasing from top to bottom rows.

Figure 10.6: Adversarial refinement of p_G (VAE trained on MNIST) based on an adversarial discriminator f, comparison of theoretical (in blue) and empirical results (in orange). 10.6(a), 10.6(b): Evolution of $\mathbb{E}_{\hat{p}_\lambda} f$ and $D_{KL}(\hat{p}_\lambda \| p)$ with λ. 10.6(c): Pareto front of both objectives. 10.6(d): top to bottom rows respectively correspond to λ in $\{0.0, 0.125, 0.250, 0.375, 0.5, 0.625, 0.750\}$; on each row are samples generated from $\hat{p}_\lambda$, showing a mode dropping phenomenon (see text).

A proof of concept for this *a posteriori* adversarial refinement of an overly general distribution is provided on the MNIST dataset. p_G is trained as a Gaussian VAE; f is a classifier trained to distinguish p_G from the dataset, with 99% accuracy. BTGM is applied on the VAE latent space (Section 10.2.1), and the results are presented in Figure 10.6.

An ideal solution is obtained for $\mathbb{E}_{\hat{p}_\lambda} f \approx 0$, i.e. on average, the classifier cannot distinguish between $\hat{p}_\lambda$ and p_D. As expected, BTGM does not manage to reach this ideal solution and the optimization saturates for $\lambda \leq 0.5$, failing to tighten the model further. The D_{KL} value reached at this point is ≈ 4, hinting at a concentration of the model around the 2% most realistic images according to the classifier[8].

The reason for failing to improve further $\hat{p}_\lambda$ likely lies in the shape of the

[8]The reasoning is as follows. Considering some base distribution p, and another distribution q which is proportional to p, with support is restricted to some subset S of the support of p, then one has $D_{KL}(q\|p) = -\log p(S)$, where $p(S)$ represents the total probability mass of the set S according to p. Inversely, if $\hat{p}_\lambda$ restricts its support − while remaining proportional to p on this support (which is untrue in general), then the mass of its support can be estimated from the value of the KL-divergence.

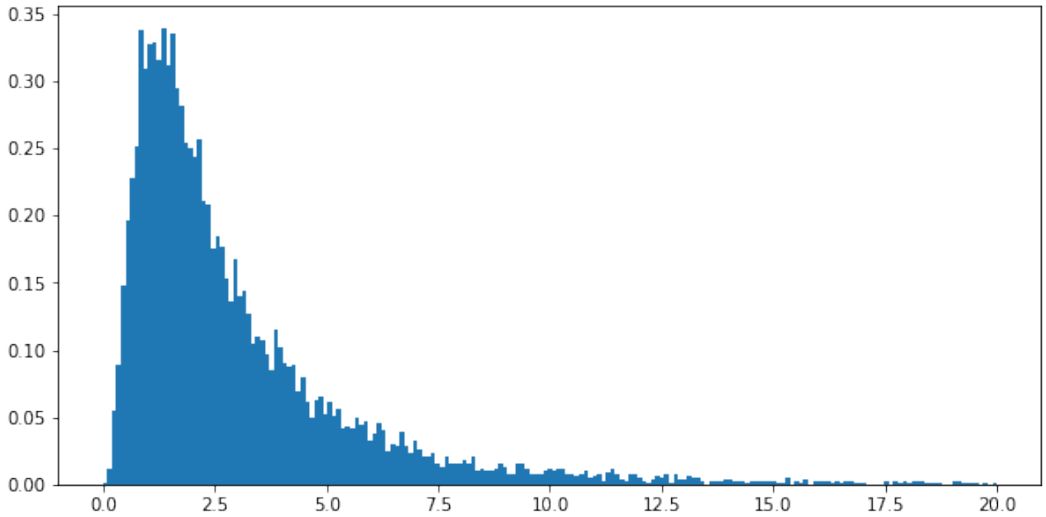


Figure 10.7: Distribution of the norm of the gradient of the objective f (pre-activation output of the discriminator) wrt to the latent variable. The histogram is truncated at a norm of 20 for clarity, but around 1% of the gradients have a norm circa 230.

optimization landscape defined by the discriminator. Figure 10.6(b) shows the optimization plateau at the same value of λ for which the curve for D_{KL} steepens, hinting at a point where the optimization landscape contains large plateaus and high cliffs, as in Section 10.3.1.

This tentative interpretation is confirmed by plotting the histogram of the gradient norm (Figure 10.7): while most gradients are well-behaved, around 1% of them take very high values, up to a norm of 230. This optimization landscape likely contains some very high cliffs, that the normalizing flow fails to capture.

This optimization landscape suggests that the discriminator manages to tightly characterize the support of the real data distribution p_D, leading to sharp boundaries between the classes. As the support of p_D is known to be contained within that of p_G (as VAEs always cover the whole dataset), this suggests the generative model p_G is too spread out, making the discriminator task too easy. Empirically, BTGM fails to bring the expectation of $\log \frac{p_D}{p_G}$ higher than -5, meaning the region is still largely classified as belonging to p_G. It is also likely that the Gaussian observation model of the VAE further introduces some noise over which BTGM does not have control (as it only operates in the latent space) but which is exploited by the discriminator.

This last point underlines the fundamental assumption made in Section 10.2.1: that $p_G(x|z)$ reflects the true distribution. In the case of a Gaussian observation model, this assumption contradicts the analysis presented in Chapter 6 and Chapter 7, showing that there will always be some residual noise that a discriminator could capture. A perspective to mitigate this drawback would be to train the discriminator on a version of the dataset on which a similar noise has been introduced, in the spirit of [Saj+18].

10.4 Summary and perspectives

The *Boltzmann Tuning for Generative Models* (BTGM) presented in this chapter constitutes the third contribution of this thesis. Built on Deep LVMs and the celebrated Maximum Entropy Principle, BTGM makes it possible to selectively revise some parts of a Deep LVM based on an external differentiable criterion f operating on a subset of variables. The presented framework focuses on adjusting the distribution of the latent (Z) variable of a VAE based on a criterion f operating on its observed (X) variable; in the general case, the distribution and criterion can equally handle observed or latent variables.

BTGM can naturally be interpreted in probabilistic terms, e.g. recovering a conditional generative model when criterion f is based on a discriminative classifier. On our motivating application in the domain of electrical consumption curves, it gracefully modifies the original distribution according to the criterion, preserving the realism and general behavior of real electric curves when the base distribution is "sufficiently" appropriate.

A third case study, concerned with the adversarial refinement of a distribution, illustrates the main limitation of the approach, as follows. BTGM freezes some parts of the LVM distribution and modifies the others. But even when training extremely powerful normalizing flows to achieve the modifications, BTGM cannot compensate for deficiencies present in frozen parts of the model.

This remark opens a perspective for further research. As retraining the whole distribution (not freezing any part of it) is expensive, the question is whether and how marginal modifications on some parts of the model can be achieved, avoiding both the rigidity of freezing them and the computational cost of learning them fully.

10.A Derivation of the MaxEnt solution

The constrained optimization problem of Equation 10.1 is restated here:

$$\arg\max_{\hat{p}} H(\hat{p}) \quad \text{s.t.} \quad \forall i : \mathop{\mathbb{E}}_{x\sim\hat{p}} f_i(x) = C_i \tag{10.28}$$

In order to solve it, we use consider the Lagrange multiplier method, defining the Lagrangian as:

$$\mathcal{L} = \underbrace{\int_{\mathcal{X}} \hat{p}(x)\log\hat{p}(x)dx}_{H(\hat{p})} - \eta\underbrace{\left(\int_{\mathcal{X}}\hat{p}(x)dx - 1\right)}_{\text{Normalization of }\hat{p}} - \sum_i \lambda_i \underbrace{\left(\int_{\mathcal{X}}\hat{p}(x)f_i(x)dx - C_i\right)}_{i\text{-th constraint}} \tag{10.29}$$

Then, solving for $\hat{p}$ and the Lagrange multipliers η and $(\lambda_i)_i$ implies requiring that we are at a saddle point of the Lagrangian $\mathcal{L}$ for all of these variables. For $\hat{p}$, this means that, for any small perturbation $\hat{p} \to \hat{p}\delta\hat{p}$ the first-order change of the Lagrangian must be $\delta\mathcal{L} = 0$. Injecting this into the definition and only keeping first-order terms in $\delta\hat{p}$ yields:

$$\delta\mathcal{L} = \int_{\mathcal{X}} \left[\log\hat{p}(x) + 1 - \eta - \sum_i \lambda_i f_i(x)\right]\delta\hat{p}(x)dx \tag{10.30}$$

This expression must be 0 for any perturbation $\delta\hat{p}$, meaning the term between brackets must be 0 for every value of x. This means in turn:

$$\log\hat{p}(x) = \eta - 1 + \sum_i \lambda_i f_i(x) \tag{10.31}$$

And then:

$$\hat{p}(x) = e^{\eta-1}e^{\sum_i \lambda_i f_i(x)} \tag{10.32}$$

The value of η is determined by the constraint that $\hat{p}$ must be a normalized probability distribution. Defining Z as:

$$Z = \frac{1}{e^{\eta-1}} = \int_x e^{\sum_i \lambda_i f_i(x)}dx \tag{10.33}$$

We finally get:

$$\hat{p}(x) = \frac{1}{Z}e^{\sum_i \lambda_i f_i(x)} \tag{10.34}$$

The value of the remaining Lagrange multipliers $(\lambda_i)_i$ must now be determined according to the other constraints: $\mathbb{E}_{\hat{p}}\, f_i = C_i$.

10.B Proofs of the derivative formulas

In order to derive the formulas for the derivatives, let us first derive two intermediate results (assuming a single criterion f and Lagrange multiplier λ):

Lemma 10.1. *The derivative* $\frac{d}{d\lambda}\log Z(\lambda)$ *reads:*

$$\frac{d}{d\lambda}\log Z(\lambda) = \mathop{\mathbb{E}}_{\hat{p}_\lambda} f \tag{10.35}$$

Proof. As $Z(\lambda) = \int_x p(x)e^{\lambda f(x)}dx$ by definition, it follows:

$$\begin{aligned}
\frac{d}{d\lambda}\log Z(\lambda) &= \frac{1}{Z(\lambda)}\frac{d}{d\lambda}Z(\lambda) \\
&= \frac{1}{Z(\lambda)}\frac{d}{d\lambda}\int_{\mathcal{X}} p(x)e^{\lambda f(x)}dx \\
&= \frac{1}{Z(\lambda)}\int_{\mathcal{X}} f(x)p(x)e^{\lambda f(x)}dx \\
&= \int_x f(x)\hat{p}_\lambda(x) \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda} f
\end{aligned} \tag{10.36}$$

□

Lemma 10.2. *Let $h : \mathcal{X} \to \mathbb{R}$ be a function (possibly depending on λ as well as x). The derivative of its expectation on $\hat{p}_\lambda$ wrt λ reads:*

$$\frac{d}{d\lambda}\mathop{\mathbb{E}}_{\hat{p}_\lambda} h = \mathop{\mathbb{E}}_{\hat{p}_\lambda}\left[fh + \frac{\partial h}{\partial \lambda}\right] - \left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)\left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} h\right) \tag{10.37}$$

Proof.

$$\begin{aligned}
\frac{d}{d\lambda}\mathop{\mathbb{E}}_{\hat{p}_\lambda} h &= \frac{d}{d\lambda}\frac{1}{Z(\lambda)}\int_x h(x)p(x)e^{\lambda f(x)}dx \\
&= \frac{1}{Z(\lambda)}\int_x \left(h(x)f(x) + \frac{\partial h}{\partial \lambda}(x)\right)p(x)e^{\lambda f(x)}dx \\
&\quad - \frac{1}{Z(\lambda)^2}\frac{dZ}{d\lambda}\int_x h(x)p(x)e^{\lambda f(x)}dx \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda}\left[hf + \frac{\partial h}{\partial \lambda}\right] - \left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} h\right)\frac{d}{d\lambda}\log Z(\lambda) \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda}\left[fh + \frac{\partial h}{\partial \lambda}\right] - \left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)\left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} h\right)
\end{aligned} \tag{10.38}$$

□

From Lemma 10.1 and Lemma 10.2, we can derive the first and second derivatives of $\mathbb{E}_{\hat{p}_\lambda} f$:

Lemma 10.3. *The first and second derivatives of $\mathbb{E}_{\hat{p}_\lambda} f$ wrt λ read:*

$$\frac{d}{d\lambda}\mathop{\mathbb{E}}_{\hat{p}_\lambda} f = Var_{\hat{p}_\lambda} f \quad \text{and} \quad \frac{d^2}{d\lambda^2}\mathop{\mathbb{E}}_{\hat{p}_\lambda} f = \mathop{\mathbb{E}}_{\hat{p}_\lambda}\left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^3 \tag{10.39}$$

Proof. Replacing h with f in Equation 10.37, and noting that f does not depend on λ, yields the first derivative:

$$\frac{d}{d\lambda}\mathop{\mathbb{E}}_{\hat{p}_\lambda} f = \mathop{\mathbb{E}}_{\hat{p}_\lambda} f^2 - \left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^2 = Var_{\hat{p}_\lambda} f \tag{10.40}$$

Noting that $Var_{\hat{p}_\lambda} f = \mathbb{E}_{\hat{p}_\lambda}\left(f - \mathbb{E}_{\hat{p}_\lambda} f\right)^2$ and replacing h with $\left(f - \mathbb{E}_{\hat{p}_\lambda} f\right)^2$ (that does depend on λ) in Equation 10.37 yields the second derivative:

$$\begin{aligned}
\frac{d^2}{d\lambda^2} \mathop{\mathbb{E}}_{\hat{p}_\lambda} f &= \frac{d}{d\lambda} \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^2 \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left[f \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^2 - 2 \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right) \frac{d}{d\lambda} \mathop{\mathbb{E}}_{\hat{p}_\lambda} f \right] - \left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right) \left(\mathop{\mathbb{E}}_{\hat{p}_\lambda} \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^2\right) \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^3 - 2 \underbrace{\mathop{\mathbb{E}}_{\hat{p}_\lambda} \left[f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f \right]}_{=0} \frac{d}{d\lambda} \mathop{\mathbb{E}}_{\hat{p}_\lambda} f \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^3
\end{aligned} \tag{10.41}$$

□

And similarly the first and second derivatives of $D_{KL}(\hat{p}_\lambda \| p)$:

Lemma 10.4. *The first and second derivatives of $D_{KL}(\hat{p}_\lambda \| p)$ wrt λ read:*

$$\frac{d}{d\lambda} D_{KL}(\hat{p}_\lambda \| p) = \lambda Var_{\hat{p}_\lambda} f \quad \textit{and} \quad \frac{d^2}{d\lambda^2} D_{KL}(\hat{p}_\lambda \| p) = Var_{\hat{p}_\lambda} f + \lambda \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda} f\right)^3 \tag{10.42}$$

Proof. By definition:

$$\begin{aligned}
D_{KL}(\hat{p}_\lambda \| p) &= \mathop{\mathbb{E}}_{\hat{p}_\lambda} \log \frac{\hat{p}_\lambda}{p} \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda} \left[\lambda f - \log Z(\lambda)\right] \\
&= \lambda \mathop{\mathbb{E}}_{\hat{p}_\lambda} [f] - \log Z(\lambda)
\end{aligned} \tag{10.43}$$

Lemmas 1 and 2 thus yield:

$$\begin{aligned}
\frac{d}{d\lambda} D_{KL}(\hat{p}_\lambda \| p) &= \mathop{\mathbb{E}}_{\hat{p}_\lambda} [f] + \lambda \frac{d}{d\lambda} \mathop{\mathbb{E}}_{\hat{p}_\lambda} [f] - \frac{d}{d\lambda} \log Z(\lambda) \\
&= \mathop{\mathbb{E}}_{\hat{p}_\lambda} [f] + \lambda Var_{\hat{p}_\lambda} [f] - \mathop{\mathbb{E}}_{\hat{p}_\lambda} [f] \\
&= \lambda Var_{\hat{p}_\lambda} f
\end{aligned} \tag{10.44}$$

and:

$$\begin{aligned}\frac{d}{d\lambda}D_{KL}(\hat{p}_\lambda\|p) &= Var_{\hat{p}_\lambda}f + \lambda\frac{d}{d\lambda}Var_{\hat{p}_\lambda}f \\ &= Var_{\hat{p}_\lambda}f + \lambda\mathop{\mathbb{E}}_{\hat{p}_\lambda}\left(f - \mathop{\mathbb{E}}_{\hat{p}_\lambda}f\right)^3\end{aligned} \tag{10.45}$$

□

10.C Monte-Carlo Prediction of $\mathbb{E}_{\hat{p}_\lambda} f$ and $D_{KL}(\hat{p}_\lambda\|p)$

Following the theoretical definition of $\hat{p}_\lambda$ (Equation 10.5):

$$\hat{p}_\lambda(x) = \frac{p(x)}{Z(\lambda)}e^{\lambda f(x)} \tag{10.46}$$

One can express $Z(\lambda)$ as:

$$Z(\lambda) = \int_x p(x)e^{\lambda f(x)}dx = \mathop{\mathbb{E}}_{p} e^{\lambda f} \tag{10.47}$$

From this, interpreting the integrals as expectations over p yield the expected results:

$$\mathop{\mathbb{E}}_{\hat{p}_\lambda} f = \int_x f(x)\frac{p(x)e^{\lambda f(x)}}{Z(\lambda)}dx = \frac{\int_x p(x)f(x)e^{\lambda f(x)}dx}{\int_x p(x)e^{\lambda f(x)}dx} = \frac{\mathbb{E}_p\, fe^{\lambda f}}{\mathbb{E}_p\, e^{\lambda f}} \tag{10.48}$$

Similarly, re-injecting the definition of $\hat{p}_\lambda$ into the KL yields:

$$D_{KL}(\hat{p}_\lambda\|p) = \mathop{\mathbb{E}}_{x\sim\hat{p}_\lambda}\log\frac{\hat{p}_\lambda(x)}{p(x)} \tag{10.49}$$

$$= \mathop{\mathbb{E}}_{x\sim\hat{p}_\lambda}\log\frac{e^{\lambda f(x)}}{Z(\lambda)} \tag{10.50}$$

$$= \lambda\mathop{\mathbb{E}}_{\hat{p}_\lambda} f - \log Z(\lambda) \tag{10.51}$$

And finally, using the previous results:

$$D_{KL}(\hat{p}_\lambda\|p) = \frac{\mathbb{E}_p\, fe^{\lambda f}}{\mathbb{E}_p\, e^{\lambda f}} - \log\left(\mathop{\mathbb{E}}_{p} e^{\lambda f}\right) \tag{10.52}$$

Conclusion and Perspectives

Deep latent variable models (LVM) are at the forefront of the state of the art in generative models, offering both a flexible model space and a principled training methodology, relying on the *Evidence Lower Bound* (ELBO) criterion to maximize the data likelihood.

The analysis presented in the manuscript is conducted in the perspective of the structure design, including the associated inference models, and their impacts. This perspective is contrasted with a number of works at the state of the art, e.g. [BGS16; Hig+17; Ale+18; RV18; Raz+19], that proceed with augmenting the ELBO with ad hoc regularization terms, or with altering the training process, in order to shape the deep LVM and enforce the desired properties.

Quite the contrary, our claim is that the structure design can convey a number of targeted properties in a principled yet efficient way, operating at both static and dynamic levels.

On the static level, mainly two topics have been investigated: the relationship to the data, and the latent structure and symmetries. Regarding the relationship with the data, Chapter 5 emphasizes that the *observation model*, the probabilistic structure defined on the observed variables of the LVM, disentangles the notions of *signal* and *noise*. The signal is encoded into the latent variables; the noise is modeled through the observed variables. At one extreme, (too) powerful autoregressive observation models yield the *posterior collapse* phenomenon, where the LVM essentially models the whole data information based on noise only. A particular case is when observed variables are noisy deterministic functions of the latent variables. In this case, referred to as *quasi-deterministic* observation models and discussed in Chapter 6, a rich representation of the data structure can still be obtained, e.g. using a hierarchy of variables to capture multi-scale representations [Dor+17].
In this context, our first contribution shows theoretically and empirically that such observation models are governed by the scale of their noise. This hyper-parameter actually governs the trade-off between the latent compression and the reconstruction terms of the ELBO. In particular, we show on a proof of concept that an over-estimated noise hyper-parameter can prevent from identifying the data manifold, even in the large sample limit (Theorem 6.1).

Our second contribution is to show how the latent structure organization can be leveraged to enforce sophisticated generative properties, with a motivating application in the domain of smart energy policies. Specifically, the CompVAE model (Chapter 9) uses a hierarchy of latent variables to deliver a programmable generative model, conditioned on a *multiset* of elements. By using one latent variable per conditioning element and aggregating them through a permutation-invariant function, the

permutation symmetries of the input multi-set are seamlessly enforced in the model structure. On top of the aggregation, the structure involves a global latent variable that captures the factors of variation shared by all entitites (in the application domain where the entities are households, and the programmable generative model delivers the aggregated consumption curves of the households, the latter latent variable reflects e.g. the weather). The empirical validation of the approach in the context of the NEXT contract (funded by the French Energy Agency, ADEME) successfully showed the merits and efficiency of the approach.

On the dynamic level, mainly three topics have been of interest: the relationship between the structure and the optimization trajectory; the *a posteriori* refinement of the probabilistic model; and thirdly (Chapter 3) the regularization role of the *inference model*, guiding convergence toward an easy to approximate posterior distribution, as analyzed through the lens of *posterior regularization* [Gan+10; Shu+18].

Regarding the relationship with the optimization trajectory, the learning of the observation noise variance has been studied in Chapter 7. The main result is to show that the impact goes much deeper than only finding the appropriate value of the variance parameter. Specifically, starting with a high noise enables the model to first learn a very smooth approximation of the data manifold, discarding much of the data information. This approximation allows an efficient latent representation to be found; this latent representation is iteratively refined as the noise variance is decreased, enabling the model to gradually capture finer-grained details from the data. Overall, the optimization trajectory yields a much better (quasi) optimum through this interplay between both terms of latent compression and reconstruction loss, akin an annealing procedure. The improvement is equally manifest in terms of the eventual ELBO value and the quality of the generated samples. The impact of this optimization trajectory can be understood in the perspective of Bayesian deep learning [MAV17], stating that a final (sparse) solution can best be found by first considering an unconstrained space (with no sparsity constraint) and only thereafter gradually increasing the sparsity pressure. In our case, the complexity of the learning task gradually and automatically increases as the observation variance decreases, the reduction of variance being itself driven by the progress of the reconstruction loss; the whole model can thus be seen as self-regularized.

Regarding the *a posteriori* refinement of a probabilistic model, our last contribution is the *Boltzmann Tuning of Generative Models (BTGM)* (Chapter 10). Formally, the BTGM framework establishes that a trained deep LVM can be "surgically" altered using Variational Inference principles according to an external criterion (operating on observed variables) through modifying only some specific parts of the learned latent representation. This approach can be applied to *a posteriori* condition a trained model or explore the extreme regions of a distribution, while controlling precisely which factors – or causes – are allowed to be modified, and to which extent. As a result, while not directly linked to model design, BTGM can exploit a pre-existing latent structure for directed tuning. The experimental validation on the same motivating application of CompVAE indeed shows that the BTGM approach constitutes an efficient alternative to rejection sampling in order to explore the extreme regions of the distribution.

In summary, out motto is that encoding model properties through the LVM

structure design is when possible an efficient way to achieve *robust* models which are amenable to being *explainable* and *transparent*. The guiding thread of the presented research is that an LVM can be well characterized from both static and dynamic perspectives from its only structure design, while preserving the probabilistic interpretation of the ELBO criterion. This approach provides a complementary perspective on model design to the introduction of additional objectives or regularization terms within the training loss [BGS16; Hig+17; Ale+18; RV18; Raz+19].

This motto thus raises the question of how to encode domain- or problem-dependent objectives through the LVM design. This question is at the core of the main research perspectives opened by the presented work.

A first perspective is to build a bridge between deep LVMs and causal models, and more specifically to uncover and exploit the causal structure of the real data-generating process to guide the design of the LVM structure. On the one hand the idea is simple and clear: it is generally believed that the true underlying causal structure of the data is among the simplest ones accounting for the data everything else being equal [PJS17; PM18]; the true causal structure thus should yield an easier to train model and deliver a better generalization overall. This claim needs however to be taken with a grain of salt. Firstly, finding the true causal structure is not a simpler problem than building a good generative model. Secondly, the true causal structure is not necessarily aligned with the requirements of the expected usage of the model; it might also make the training procedure harder than it needs to be.

A counter-example backing this remark is again provided by the motivating application of CompVAE. In the context of the generative model of aggregated electrical curves, there exist root causes governing each and every household consumption (e.g. the weather, the holidays, a match on the TV). These root causes operate besides the individual causes attached to each household (e.g. the electrical appliances, the type of contract of the household). In a relevant causal model the outcome thus is defined as the aggregation of all household curves, where each household curve is governed together by the root and the individual causes.

The CompVAE model however first models the aggregation of the households (in an abstract latent space); and the external factors are introduced on top of the aggregation to produce the global aggregated curve. In other words, the root causes appear after the individual causes in the hierarchy. This structure is motivated as external factors affect individual households in a coherent way, and because the targeted outcome is the global behavior only. For this reason, CompVAE does not need to represent the root causes *per se*, through some associated latent variable; instead, it directly encodes the *effect* of those root causes onto the aggregated curve. This architecture comes with two main advantages. Firstly, it results in a significantly simpler computational graph, facilitating its training and contributing to the stability of the solution. Secondly, this architecture allows the latent representation to efficiently disentangle the impact of external factors from the individual factors of each household, supporting an efficient ensemblist manipulation of the household descriptions.

In summary, the LVM architecture can usefully take inspiration from the causal underlying structure of the phenomenon under study, with two caveats. On the one hand, uncovering the causal structure is a problem *per se*; on the other hand, the true causal structure needs be revisited in the perspective of the *usage* and *learning*

of the LVM.

Another, longer-term, perspective of research lies in extending BTGM to tackle inverse problems. Formally, given the objective function and a trained LVM architecture, one might want to determine the *minimal amount of change* required to meet one's objectives in terms of system response. BTGM presently relies on the assumption of an accurate differentiable generative model. A first line of research concerns the extension of the approach to handle binary or categorical variables; a second line of research consists of investigating the impact of model misspecifications.

Formally, BTGM presently proceeds by perturbing the distribution of observed or latent variables. But the sought perturbations can also be thought of in terms of *interventions* on observed or latent variables, and/or on the mechanisms linking the observed to latent variables. Along this line, the extended BTGM could go much beyond counter-factual reasoning, and actually tackle planning (what changes are most effective to reach a given effect). Overall, this extension would help to identify minimal interventions on a system to reach desired behavior. In an applicative perspective, this approach paves the way toward optimal design and control (how to best control the system in order to get a given response).

When applied on a causal model (as opposed to a generative model), this BTGM extension might help identifying the (observed) variables that should best be controlled, and guide the design efforts. When applied to control latent variables, this could allow to identify which parts of the model need to be refined the most, and/or pinpoint the deficiencies of the model, and how to repair it by introducing more epistemic knowledge.

A shorter-term research perspective opened by the presented work is related to the *quasi-deterministic* observation models, as the presented analysis of the relationship between the model and the training data calls for further refinement. Theorem 6.1 provides an intuitive understanding of the smoothing effect of the observation noise on highly curved manifolds. A most interesting extension of this analytical result aims to predict *how* those manifolds are smoothed depending on the precision. Similarly, the precise analysis of non-isotropic noise (e.g. like in Laplacian Pyramid observations [Dor+17] on images) might yield new lessons about very structured and high-dimensional data. Lastly, while the theoretical and empirical results have been established under the large-sample limit assumption, it remains to see how the data shortage might interact with the data precision.

Bibliography

[AB17] Martin Arjovsky and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *arXiv:1701.04862 [cs, stat]* (Jan. 17, 2017). arXiv: 1701.04862 (cit. on pp. 44, 74, 146, 147).

[Aka74] H. Akaike. "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6 (Dec. 1974). Conference Name: IEEE Transactions on Automatic Control, pp. 716–723. ISSN: 1558-2523 (cit. on p. 24).

[Ale+18] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A. Saurous, and Kevin Murphy. "Fixing a Broken ELBO". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. July 3, 2018, pp. 159–168 (cit. on pp. 71, 72, 155, 157).

[Ath+18] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing Robust Adversarial Examples". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 3, 2018, pp. 284–293 (cit. on p. 104).

[BA98] Kenneth P. Burnham and David R. Anderson. "Practical Use of the Information-Theoretic Approach". In: *Model Selection and Inference: A Practical Information-Theoretic Approach*. Ed. by Kenneth P. Burnham and David R. Anderson. New York, NY: Springer, 1998, pp. 75–117. ISBN: 978-1-4757-2917-7 (cit. on p. 24).

[Bed+16] Marcos Vinicius Naves Bedo, Davi Pereira dos Santos, Marcelo Ponciano-Silva, Paulo Mazzoncini de Azevedo-Marques, André Ponce de León Ferreira de Carvalho, and Caetano Traina. "Endowing a Content-Based Medical Image Retrieval System with Perceptual Similarity Using Ensemble Strategy". In: *Journal of Digital Imaging* 29.1 (Feb. 1, 2016), pp. 22–37. ISSN: 1618-727X (cit. on p. 64).

[Ber+19] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. "Sylvester Normalizing Flows for Variational Inference". In: *UAI 2018* (Feb. 20, 2019). arXiv: 1803.05649 (cit. on p. 36).

[Bes94] JE Besag. "Comments on "Representations of knowledge in complex systems" by U. Grenander and MI Miller". In: *J. Roy. Statist. Soc. Ser. B* 56 (1994), pp. 591–592 (cit. on p. 20).

[BG02] Matthew Beal and Zoubin Ghahramani. "The Variational Bayesian EM Algorithm for Incomplete Data : with Application to Scoring Graphical Model Structures". In: *Statistics* (July 25, 2002) (cit. on p. 34).

[BG06] Matthew J. Beal and Zoubin Ghahramani. "Variational Bayesian learning of directed graphical models with hidden variables". In: *Bayesian Analysis* 1.4 (Dec. 2006). Publisher: International Society for Bayesian Analysis, pp. 793–831. ISSN: 1936-0975, 1931-6690 (cit. on p. 34).

[BGS16] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance Weighted Autoencoders". In: *arXiv:1509.00519 [cs, stat]* (Nov. 7, 2016). arXiv: 1509.00519 (cit. on pp. 45, 155, 157).

[Bis+98] Christopher M. Bishop, Neil D. Lawrence, Tommi Jaakkola, and Michael I. Jordan. "Approximating Posterior Distributions in Belief Networks Using Mixtures". In: *Advances in Neural Information Processing Systems 10*. Ed. by M. I. Jordan, M. J. Kearns, and S. A. Solla. MIT Press, 1998, pp. 416–422 (cit. on pp. 20, 35).

[BK18] D. T. Braithwaite and W. B. Kleijn. "Bounded Information Rate Variational Autoencoders". In: *KDD 2018* (July 25, 2018). arXiv: 1807.07306 (cit. on p. 73).

[Bow+15] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. "Generating Sentences from a Continuous Space". In: (Nov. 19, 2015) (cit. on pp. 66, 116).

[BPC19] Gwendoline De Bie, Gabriel Peyré, and Marco Cuturi. "Stochastic Deep Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, May 24, 2019, pp. 1556–1565 (cit. on pp. 116, 132).

[BS20a] Victor Berger and Michèle Sebag. "From Abstract Items to Latent Spaces to Observed Data and Back: Compositional Variational Auto-Encoder". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Ulf Brefeld, Elisa Fromont, Andreas Hotho, Arno Knobbe, Marloes Maathuis, and Céline Robardet. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 274–289. ISBN: 978-3-030-46150-8 (cit. on pp. 8, 115, 122).

[BS20b] Victor Berger and Michèle Sebag. "Variational Auto-Encoder: not all failures are equal". In: *arXiv:2003.01972 [cs, eess, stat]* (Mar. 4, 2020). arXiv: 2003.01972 (cit. on pp. 8, 73, 87).

[BS21] Victor Berger and Michele Sebag. "Boltzmann Tuning of Generative Models". In: *arXiv:2104.05252 [cs]* (Apr. 12, 2021). arXiv: 2104.05252 (cit. on pp. 8, 135).

[BWS15] Pratik Brahma, Dapeng Wu, and Yiyuan She. "Why Deep Learning Works: A Manifold Disentanglement Perspective". In: *IEEE Transactions on Neural Networks and Learning Systems* 27 (Dec. 16, 2015), pp. 1–12 (cit. on p. 73).

[Cat07] Ariel Caticha. "Information and Entropy". In: *AIP Conference Proceedings* 954 (2007), pp. 11–22. ISSN: 0094243X. arXiv: 0710.1068 (cit. on p. 136).

[Cay05] Lawrence Cayton. "Algorithms for manifold learning". In: *Univ. of California at San Diego Tech. Rep* 12.1 (2005), p. 1 (cit. on p. 74).

[CCR21] Marissa Connor, Gregory Canal, and Christopher Rozell. "Variational Autoencoder with Learned Latent Structure". In: *International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics. ISSN: 2640-3498. PMLR, Mar. 18, 2021, pp. 2359–2367 (cit. on p. 73).

[CDS18] Anthony L Caterini, Arnaud Doucet, and Dino Sejdinovic. "Hamiltonian Variational Auto-Encoder". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 8167–8177 (cit. on p. 45).

[CH92] Gregory F Cooper and Edward Herskovits. "A Bayesian method for the induction of probabilistic networks from data". In: *Machine Learning* 9.4 (1992). Publisher: Springer, pp. 309–347 (cit. on p. 24).

[Che+17] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Variational Lossy Autoencoder". In: *ICLR 2017* (Mar. 4, 2017). arXiv: 1611.02731 (cit. on p. 70).

[Che+18a] Ricky T. Q. Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. "Isolating Sources of Disentanglement in Variational Autoencoders". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 2610–2620 (cit. on pp. 50, 135).

[Che+18b] X. I. Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. "PixelSNAIL: An Improved Autoregressive Generative Model". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 3, 2018, pp. 864–872 (cit. on p. 68).

[CHM04] David Maxwell Chickering, David Heckerman, and Christopher Meek. "Large-Sample Learning of Bayesian Networks is NP-Hard". In: *Journal of Machine Learning Research* 5 (Oct 2004), pp. 1287–1330. ISSN: ISSN 1533-7928 (cit. on p. 24).

[Chu+15] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. "A Recurrent Latent Variable Model for Sequential Data". In: *Advances in Neural Information Processing Systems* 28 (2015), pp. 2980–2988 (cit. on pp. 66, 67, 116).

[Cia+13] Lucio Ciabattoni, Massimo Grisostomi, Gianluca Ippoliti, and Sauro Longhi. "A Fuzzy Logic tool for household electrical consumption modeling". In: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society. ISSN: 1553-572X. Nov. 2013, pp. 8022–8027 (cit. on p. 115).

[CJ11] Cassio P. de Campos and Qiang Ji. "Efficient Structure Learning of Bayesian Networks using Constraints". In: *Journal of Machine Learning Research* 12.20 (2011), pp. 663–689. ISSN: 1533-7928 (cit. on p. 24).

[CJA19] Hyunsun Choi, Eric Jang, and Alexander A. Alemi. "WAIC, but Why? Generative Ensembles for Robust Anomaly Detection". In: *arXiv:1810.01392 [cs, stat]* (May 23, 2019). version: 4. arXiv: 1810.01392 (cit. on pp. 105, 111).

[CMD17] Chris Cremer, Quaid Morris, and David Duvenaud. "Reinterpreting Importance-Weighted Autoencoders". In: *arXiv:1704.02916 [stat]* (Aug. 14, 2017). arXiv: 1704.02916 (cit. on p. 45).

[CW17] Nicholas Carlini and David Wagner. "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. AISec '17. New York, NY, USA: Association for Computing Machinery, Nov. 3, 2017, pp. 3–14. ISBN: 978-1-4503-5202-4 (cit. on p. 104).

[Dai+18] Bin Dai, Yu Wang, John Aston, Gang Hua, and David Wipf. "Connections with Robust PCA and the Role of Emergent Sparsity in Variational Autoencoder Models". In: *Journal of Machine Learning Research* 19.41 (2018), pp. 1–42. ISSN: 1533-7928 (cit. on p. 73).

[DB16] Alexey Dosovitskiy and Thomas Brox. "Generating Images with Perceptual Similarity Metrics based on Deep Networks". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 658–666 (cit. on p. 65).

[Den+09] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition. ISSN: 1063-6919. June 2009, pp. 248–255 (cit. on p. 65).

[Dil+17] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. "Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders". In: *arXiv:1611.02648 [cs, stat]* (Jan. 13, 2017). arXiv: 1611.02648 (cit. on p. 46).

[DKB15] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear Independent Components Estimation". In: *arXiv:1410.8516 [cs]* (Apr. 10, 2015). arXiv: 1410.8516 (cit. on p. 36).

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977). Publisher: [Royal Statistical Society, Wiley], pp. 1–38. ISSN: 0035-9246 (cit. on p. 36).

[DMM09] David L. Donoho, Arian Maleki, and Andrea Montanari. "Message-passing algorithms for compressed sensing". In: *Proceedings of the National Academy of Sciences* 106.45 (Nov. 10, 2009). Publisher: National Academy of Sciences Section: Physical Sciences, pp. 18914–18919. ISSN: 0027-8424, 1091-6490 (cit. on p. 17).

[Don+19] Wei Dong, Qinliang Su, Dinghan Shen, and Changyou Chen. "Document Hashing with Mixture-Prior Generative Models". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. EMNLP-IJCNLP 2019. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5226–5235 (cit. on p. 46).

[Dor+17] Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill D.F. Campbell, Simon Prince, and Ivor Simpson. "Laplacian Pyramid of Conditional Variational Autoencoders". In: *Proceedings of the 14th European Conference on Visual Media Production (CVMP 2017)*. CVMP 2017. London, United Kingdom: Association for Computing Machinery, Dec. 11, 2017, pp. 1–9. ISBN: 978-1-4503-5329-8 (cit. on pp. 75, 77, 155, 158).

[DP97] Pedro Domingos and Michael Pazzani. "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss". In: *Machine Learning* 29.2 (Nov. 1, 1997), pp. 103–130. ISSN: 1573-0565 (cit. on p. 104).

[DSB17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *ICLR 2017* (Feb. 27, 2017). arXiv: 1605.08803 (cit. on pp. 69, 70).

[DSL20] Jacob Deasy, Nikola Simidjievski, and Pietro Lió. "Constraining Variational Inference with Geometric Jensen-Shannon Divergence". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 10647–10658 (cit. on p. 73).

[Dur+19] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. "Neural Spline Flows". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d{\textbackslash}textquotesingle Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 7511–7522 (cit. on p. 36).

[DW19] Bin Dai and David Wipf. "Diagnosing and Enhancing VAE Models". In: *ICLR 2019* (Oct. 30, 2019). arXiv: 1903.05789 (cit. on pp. 55, 56).

[ES17] Harrison Edwards and Amos Storkey. "Towards a Neural Statistician". In: *ICLR 2017* (Mar. 20, 2017). arXiv: 1606.02185 (cit. on p. 117).

[Esl+18] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. "Neural Scene Representation and Rendering". In: *Science* 360.6394 (June 15, 2018), pp. 1204–1210 (cit. on p. 117).

[FA15] Otto Fabius and Joost R. van Amersfoort. "Variational Recurrent Auto-Encoders". In: *arXiv:1412.6581 [cs, stat]* (June 15, 2015). arXiv: 1412.6581 (cit. on pp. 66, 116).

[Fal+19] Luca Falorsi, Pim de Haan, Tim R. Davidson, and Patrick Forré. "Reparameterizing Distributions on Lie Groups". In: *AISTATS 2019* (Mar. 7, 2019). arXiv: 1903.02958 (cit. on p. 46).

[FMN16] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. "Testing the manifold hypothesis". In: *Journal of the American Mathematical Society* 29.4 (2016), pp. 983–1049. ISSN: 0894-0347, 1088-6834 (cit. on p. 73).

[Fra+16] Marco Fraccaro, Sø ren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. "Sequential Neural Models with Stochastic Layers". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016, pp. 2199–2207 (cit. on pp. 66, 116).

[Gan+10] Kuzman Ganchev, Jo\&\#227, O Graça, Jennifer Gillenwater, and Ben Taskar. "Posterior Regularization for Structured Latent Variable Models". In: *Journal of Machine Learning Research* 11.67 (2010), pp. 2001–2049. ISSN: 1533-7928 (cit. on pp. 49, 156).

[Gar+18] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. "Conditional Neural Processes". In: *International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 3, 2018, pp. 1704–1713 (cit. on p. 117).

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. ISSN: 1938-7228. JMLR Workshop and Conference Proceedings, June 14, 2011, pp. 315–323 (cit. on pp. 53, 118).

[GG14] Samuel Gershman and Noah Goodman. "Amortized Inference in Probabilistic Reasoning". In: *Proceedings of the Annual Meeting of the Cognitive Science Society* 36.36 (2014). ISSN: 1069-7977 (cit. on p. 42).

[GG84] Stuart Geman and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (Nov. 1984). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 721–741. ISSN: 1939-3539 (cit. on p. 19).

[GHB12] Samuel J. Gershman, Matthew D. Hoffman, and David M. Blei. "Nonparametric variational inference". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, June 26, 2012, pp. 235–242. ISBN: 978-1-4503-1285-1 (cit. on pp. 34, 35).

[Gho+20] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. "From Variational to Deterministic Autoencoders". In: *ICLR 2020* (May 29, 2020). arXiv: 1903.12436 (cit. on p. 75).

[Gil+17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. "Neural Message Passing for Quantum Chemistry". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 17, 2017, pp. 1263–1272 (cit. on p. 121).

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680 (cit. on pp. 7, 65, 147).

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *arXiv:1412.6572 [cs, stat]* (Mar. 20, 2015). arXiv: 1412.6572 (cit. on p. 104).

[GSS20] Kamal Gupta, Saurabh Singh, and Abhinav Shrivastava. "PatchVAE: Learning Local Latent Codes for Recognition". In: *arXiv:2004.03623 [cs]* (Apr. 7, 2020). arXiv: 2004.03623 (cit. on pp. 75, 82).

[Gul+16] Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. "PixelVAE: A Latent Variable Model for Natural Images". In: *arXiv:1611.05013 [cs]* (Nov. 15, 2016). arXiv: 1611.05013 (cit. on pp. 67, 70).

[Guo+17] Fangjian Guo, Xiangyu Wang, Kai Fan, Tamara Broderick, and David B. Dunson. "Boosting Variational Inference". In: *arXiv:1611.05559 [cs, stat]* (Mar. 1, 2017). arXiv: 1611.05559 (cit. on p. 35).

[Guo+20] C. Guo, J. Zhou, H. Chen, N. Ying, J. Zhang, and D. Zhou. "Variational Autoencoder With Optimizing Gaussian Mixture Model Priors". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 43992–44005. ISSN: 2169-3536 (cit. on p. 46).

[Han+17] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. "Alternating Back-Propagation for Generator Network". In: *Thirty-First AAAI Conference on Artificial Intelligence*. Thirty-First AAAI Conference on Artificial Intelligence. Feb. 13, 2017 (cit. on p. 45).

[Has70] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (Apr. 1, 1970). Publisher: Oxford Academic, pp. 97–109. ISSN: 0006-3444 (cit. on p. 19).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, pp. 770–778 (cit. on p. 53).

[He+18] Jiawei He, Yu Gong, Joseph Marino, Greg Mori, and Andreas Lehrmann. "Variational Autoencoders with Jointly Optimized Latent Dependency Structure". In: International Conference on Learning Representations 2019. Sept. 27, 2018 (cit. on pp. 58, 60).

[He+19] Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. "Lagging Inference Networks and Posterior Collapse in Variational Autoencoders". In: *ICLR 2019* (Jan. 28, 2019). arXiv: 1901.05534 (cit. on p. 70).

[Hig+17] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *ICLR*. 2017 (cit. on pp. 50, 76, 82, 88, 97, 99, 155, 157).

[HN92] David E. Heckerman and Bharat N. Nathwani. "An evaluation of the diagnostic accuracy of Pathfinder". In: *Computers and Biomedical Research* 25.1 (Feb. 1, 1992), pp. 56–74. ISSN: 0010-4809 (cit. on p. 16).

[Ho+19] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, May 24, 2019, pp. 2722–2730 (cit. on p. 70).

[Hoc+01] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies". In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by S. C. Kremer and J. F. Kolen. IEEE Press, 2001 (cit. on p. 53).

[Hof+13] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. "Stochastic variational inference". In: *The Journal of Machine Learning Research* 14.1 (May 1, 2013), pp. 1303–1347. ISSN: 1532-4435 (cit. on pp. 35, 45, 54).

[Hou+17] Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. "Deep Feature Consistent Variational Autoencoder". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). Mar. 2017, pp. 1133–1141 (cit. on pp. 65, 75, 82).

[HS06] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (July 28, 2006). Publisher: American Association for the Advancement of Science Section: Report, pp. 504–507. ISSN: 0036-8075, 1095-9203 (cit. on p. 16).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1, 1997). Publisher: MIT Press, pp. 1735–1780. ISSN: 0899-7667 (cit. on pp. 63, 66, 68).

[Hua+18] Huaibo Huang, zhihang li zhihang, Ran He, Zhenan Sun, and Tieniu Tan. "IntroVAE: Introspective Variational Autoencoders for Photographic Image Synthesis". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 52–63 (cit. on pp. 75, 82).

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". In: *arXiv:1503.02531 [cs, stat]* (Mar. 9, 2015). arXiv: 1503.02531 (cit. on p. 69).

[HY01] David J. Hand and Keming Yu. "Idiot's Bayes—Not So Stupid After All?" In: *International Statistical Review* 69.3 (2001), pp. 385–398. ISSN: 1751-5823 (cit. on p. 104).

[Ily+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. "Adversarial Examples Are Not Bugs, They Are Features". In: *Advances in Neural Information Processing Systems* 32 (2019), pp. 125–136 (cit. on p. 104).

[Jaa+10] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. "Learning Bayesian Network Structure using LP Relaxations". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. ISSN: 1938-7228. JMLR Workshop and Conference Proceedings, Mar. 31, 2010, pp. 358–365 (cit. on p. 24).

[Jay57] E. T. Jaynes. "Information Theory and Statistical Mechanics". In: *Physical Review* 106.4 (May 15, 1957). Publisher: American Physical Society, pp. 620–630 (cit. on p. 136).

[JGP17] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax". In: *arXiv:1611.01144 [cs, stat]* (Aug. 5, 2017). arXiv: 1611.01144 (cit. on p. 48).

[JOA90] Finn Verner Jensen, Kristian G. Olesen, and Stig Kjaer Andersen. "An algebra of bayesian belief universes for knowledge-based systems". In: *Networks* 20.5 (1990), pp. 637–659. ISSN: 1097-0037 (cit. on p. 17).

[Kar+17] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data". In: *ICLR 2017* (Mar. 3, 2017). arXiv: 1605.06432 (cit. on pp. 106–109, 113).

[KB17] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR 2015* (Jan. 29, 2017). arXiv: 1412.6980 (cit. on p. 54).

[KD18] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 10215–10224 (cit. on p. 70).

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN: 978-0-262-01319-2 (cit. on pp. 12, 14, 15).

[KGB17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *arXiv:1607.02533 [cs, stat]* (Feb. 10, 2017). arXiv: 1607.02533 (cit. on p. 104).

[Kim+18] Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. "Semi-Amortized Variational Autoencoders". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. July 3, 2018, pp. 2678–2687 (cit. on p. 45).

[Kin+14] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. "Semi-supervised Learning with Deep Generative Models". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 3581–3589 (cit. on pp. 105, 106, 113).

[Kin+17] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. "Improving Variational Inference with Inverse Autoregressive Flow". In: *arXiv:1606.04934 [cs, stat]* (Jan. 30, 2017). arXiv: 1606.04934 (cit. on pp. 36, 45).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* 25 (2012), pp. 1097–1105 (cit. on p. 65).

[KW14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *arXiv:1312.6114 [cs, stat]* (May 1, 2014). arXiv: 1312.6114 (cit. on pp. 7, 8, 34, 41, 43, 44, 76, 87, 88).

[Lar+16] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. "Autoencoding beyond pixels using a learned similarity metric". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. June 11, 2016, pp. 1558–1566 (cit. on pp. 65, 82, 135).

[LB08] Hugo Larochelle and Yoshua Bengio. "Classification using discriminative restricted Boltzmann machines". In: *Proceedings of the 25th international conference on Machine learning*. ICML '08. New York, NY, USA: Association for Computing Machinery, July 5, 2008, pp. 536–543. ISBN: 978-1-60558-205-4 (cit. on p. 16).

[LB98] Yann LeCun and Yoshua Bengio. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks*. Cambridge, MA, USA: MIT Press, Oct. 1, 1998, pp. 255–258. ISBN: 978-0-262-51102-5 (cit. on p. 63).

[LBS19] Yingzhen Li, John Bradshaw, and Yash Sharma. "Are Generative Classifiers More Robust to Adversarial Attacks?" In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. May 24, 2019, pp. 3804–3814 (cit. on p. 104).

[LC19] Gabriel Loaiza-Ganem and John P Cunningham. "The continuous Bernoulli: fixing a pervasive error in variational autoencoders". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d{\textbackslash}textquotesingle Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 13287–13297 (cit. on p. 97).

[Lec+98a] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998). Conference Name: Proceedings of the IEEE, pp. 2278–2324. ISSN: 1558-2256 (cit. on p. 112).

[Lec+98b] Y. Lecun, L. Bottou, G. B. Orr, and K.-R. Müller. "Efficient backprop". In: *Lecture notes in computer science*. Neural networks : tricks of the trade (1996). ISSN: 0302-9743, 1998, pp. 9–50. ISBN: 978-3-540-65311-0 (cit. on pp. 54, 57).

[Lee+18] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 7167–7177 (cit. on p. 105).

[Lee+19] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. "Set Transformer: A Framework for Attention-Based Permutation-Invariant Neural Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, May 24, 2019, pp. 3744–3753 (cit. on p. 117).

[Li+19] Xiaopeng Li, Zhourong Chen, Leonard K. M. Poon, and Nevin L. Zhang. "Learning Latent Superstructures in Variational Autoencoders for Deep Multidimensional Clustering". In: *ICLR 2019* (Feb. 22, 2019). arXiv: 1803.05206 (cit. on pp. 58, 60).

[Liu+15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild". In: Proceedings of the IEEE International Conference on Computer Vision. 2015, pp. 3730–3738 (cit. on p. 65).

[Loc+19] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. May 24, 2019, pp. 4114–4124 (cit. on p. 50).

[Lor+19] Guy Lorberbom, Andreea Gane, Tommi Jaakkola, and Tamir Hazan. "Direct Optimization through \textbackslash arg \textbackslash max for Discrete Variational Auto-Encoder". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d{\textbackslash}textquotesingle Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 6203–6214 (cit. on p. 48).

[LS88] S. L. Lauritzen and D. J. Spiegelhalter. "Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 50.2 (1988). Publisher: [Royal Statistical Society, Wiley], pp. 157–224. ISSN: 0035-9246 (cit. on p. 17).

[Luc+19] James Lucas, George Tucker, Roger Baker Grosse, and Mohammad Norouzi. "Understanding Posterior Collapse in Generative Latent Variable Models". In: *DGS@ICLR*. 2019 (cit. on pp. 70, 72–74, 92).

[LW17] Christos Louizos and Max Welling. "Multiplicative Normalizing Flows for Variational Bayesian Neural Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. July 17, 2017, pp. 2218–2227 (cit. on p. 36).

[Mat+16] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. "Disentangling factors of variation in deep representation using adversarial training". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 5040–5048 (cit. on p. 135).

[Mat+19a] Emile Mathieu, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh. "Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d{\textbackslash}textquotesingle Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 12565–12576 (cit. on p. 45).

[Mat+19b] Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. "Disentangling Disentanglement in Variational Autoencoders". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. May 24, 2019, pp. 4402–4412 (cit. on pp. 50, 135).

[MAV17] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. "Variational Dropout Sparsifies Deep Neural Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 17, 2017, pp. 2498–2507 (cit. on p. 156).

[Met+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (June 1, 1953). Publisher: American Institute of Physics, pp. 1087–1092. ISSN: 0021-9606 (cit. on p. 19).

[MF18] Pierre-Alexandre Mattei and Jes Frellsen. "Leveraging the Exact Likelihood of Deep Latent Variable Models". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 3855–3866 (cit. on pp. 76, 87, 91).

[MFA17] Andrew C. Miller, Nicholas J. Foti, and Ryan P. Adams. "Variational boosting: iteratively refining posterior approximations". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 6, 2017, pp. 2420–2429 (cit. on p. 35).

[Moy+18] Daniel Moyer, Shuyang Gao, Rob Brekelmans, Aram Galstyan, and Greg Ver Steeg. "Invariant Representations without Adversarial Training". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 9084–9093 (cit. on p. 135).

[MS96] Vladimir Maz'ya and Gunther Schmidt. "On approximate approximations using Gaussian kernels". In: *IMA Journal of Numerical Analysis* 16.1 (Jan. 1, 1996). Publisher: Oxford Academic, pp. 13–29. ISSN: 0272-4979 (cit. on p. 31).

[Nal+19a] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. "Detecting Out-of-Distribution Inputs to Deep Generative Models Using Typicality". In: *arXiv:1906.02994 [cs, stat]* (Oct. 16, 2019). arXiv: 1906.02994 (cit. on pp. 105, 111).

[Nal+19b] Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. "Do Deep Generative Models Know What They Don't Know?" In: *ICLR* (2019) (cit. on pp. 103, 105, 111).

[NG06] Dirk Neumann and Karl R. Gegenfurtner. "Image retrieval and perceptual similarity". In: *ACM Transactions on Applied Perception* 3.1 (Jan. 1, 2006), pp. 31–47. ISSN: 1544-3558 (cit. on p. 64).

[OKK16] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. June 11, 2016, pp. 1747–1756 (cit. on pp. 67, 116).

[ONS18] Victor M.-H. Ong, David J. Nott, and Michael S. Smith. "Gaussian Variational Approximation With a Factor Covariance Structure". In: *Journal of Computational and Graphical Statistics* 27.3 (July 3, 2018). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10618600.2017.1390472, pp. 465–478. ISSN: 1061-8600 (cit. on p. 35).

[Oor+16a] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu koray, Oriol Vinyals, and Alex Graves. "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 4790–4798 (cit. on p. 67).

[Oor+16b] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "WaveNet: A Generative Model for Raw Audio". In: *arXiv:1609.03499 [cs]* (Sept. 19, 2016). arXiv: 1609.03499 (cit. on pp. 68, 116).

[Oor+18] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. "Parallel WaveNet: Fast High-Fidelity Speech Synthesis". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 3, 2018, pp. 3918–3926 (cit. on p. 68).

[Ort+14] Joana Ortiz, Francesco Guarino, Jaume Salom, Cristina Corchero, and Maurizio Cellura. "Stochastic model for electrical loads in Mediterranean residential buildings: Validation and applications". In: *Energy and Buildings* 80 (Sept. 1, 2014), pp. 23–36. ISSN: 0378-7788 (cit. on p. 115).

[OVK17] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. "Neural Discrete Representation Learning". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 6306–6315 (cit. on p. 71).

[Pan+20] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. "Learning Latent Space Energy-Based Prior Model". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 22015–22029 (cit. on p. 47).

[PDZ18] Tianyu Pang, Chao Du, and Jun Zhu. "Max-Mahalanobis Linear Discriminant Analysis Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 3, 2018, pp. 4016–4025 (cit. on p. 105).

[Pea82] Judea Pearl. "Reverend bayes on inference engines: a distributed hierarchical approach". In: *Proceedings of the Second AAAI Conference on Artificial Intelligence*. AAAI'82. Pittsburgh, Pennsylvania: AAAI Press, Aug. 18, 1982, pp. 133–136 (cit. on p. 17).

[PJS17] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference : Foundations and Learning Algorithms*. Accepted: 2019-01-20 23:42:51 Journal Abbreviation: Foundations and Learning Algorithms. The MIT Press, 2017. ISBN: 978-0-262-03731-0 (cit. on pp. 16, 157).

[PM18] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018 (cit. on pp. 16, 157).

[Pol+19] Adrian Pol, Victor Berger, Gianluca Cerminara, Cécile Germain, and Maurizio Pierini. "Anomaly Detection With Conditional Variational Autoencoders". In: ICMLA 2019 - 18th IEEE International Conference on Machine Learning and Applications. Dec. 16, 2019 (cit. on pp. 8, 103, 109–113).

[Pol20] Adrian Alan Pol. "Machine Learning Anomaly Detection Applications to Compact Muon Solenoid Data Quality Monitoring". PhD thesis. Université Paris-Saclay, June 8, 2020 (cit. on p. 109).

[Raz+19] Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. "Preventing Posterior Collapse with delta-VAEs". In: *ICLR 2019* (Jan. 10, 2019). arXiv: 1901.03416 (cit. on pp. 71–73, 155, 157).

[RDL21] Oleh Rybkin, Kostas Daniilidis, and Sergey Levine. "Simple and Effective VAE Training with Calibrated Decoders". In: *International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 1, 2021, pp. 9179–9189 (cit. on p. 88).

[RGB14] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black Box Variational Inference". In: *Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics. ISSN: 1938-7228 Section: Machine Learning. Apr. 2, 2014, pp. 814–822 (cit. on p. 34).

[Rif+11] Salah Rifai, Yann N. Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. "The Manifold Tangent Classifier". In: *Advances in Neural Information Processing Systems* 24 (2011) (cit. on p. 73).

[RM15] Danilo Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. June 1, 2015, pp. 1530–1538 (cit. on pp. 35, 36, 45).

[RMC16] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *arXiv:1511.06434 [cs]* (Jan. 7, 2016). arXiv: 1511.06434 (cit. on pp. 65, 135).

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. Jan. 27, 2014, pp. 1278–1286 (cit. on pp. 7, 8, 34, 41).

[Rol17] Jason Tyler Rolfe. "Discrete Variational Autoencoders". In: *arXiv:1609.02200 [cs, stat]* (Apr. 21, 2017). arXiv: 1609.02200 (cit. on pp. 48, 71).

[ROV19] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. "Generating Diverse High-Fidelity Images with VQ-VAE-2". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d{\textbackslash}textquotesingle Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 14866–14876 (cit. on p. 71).

[RR98] Gareth O. Roberts and Jeffrey S. Rosenthal. "Optimal scaling of discrete approximations to Langevin diffusions". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60.1 (1998), pp. 255–268. ISSN: 1467-9868 (cit. on p. 20).

[RSP17a] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. "Deep Learning with Sets and Point Clouds". In: *arXiv:1611.04500 [cs, stat]* (Feb. 23, 2017). arXiv: 1611.04500 (cit. on p. 132).

[RSP17b] Siamak Ravanbakhsh, Jeff Schneider, and Barnabás Póczos. "Equivariance Through Parameter-Sharing". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 17, 2017, pp. 2892–2901 (cit. on p. 116).

[RT96] Gareth O. Roberts and Richard L. Tweedie. "Exponential convergence of Langevin distributions and their discrete approximations". In: *Bernoulli* 2.4 (Dec. 1996). Publisher: Bernoulli Society for Mathematical Statistics and Probability, pp. 341–363. ISSN: 1350-7265 (cit. on p. 20).

[RV18] Danilo Jimenez Rezende and Fabio Viola. "Taming VAEs". In: *arXiv:1810.00597 [cs, stat]* (Oct. 1, 2018). arXiv: 1810.00597 (cit. on pp. 73, 76, 81, 87, 88, 91, 155, 157).

[RVW06] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006 (cit. on p. 17).

[RWD17] Geoffrey Roeder, Yuhuai Wu, and David K. Duvenaud. "Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 6925–6934 (cit. on p. 57).

[Sad+19] Hossein Sadeghi, Evgeny Andriyash, Walter Vinci, Lorenzo Buffoni, and Mohammad H. Amin. "PixelVAE++: Improved PixelVAE with Discrete Prior". In: *arXiv:1908.09948 [cs, stat]* (Aug. 26, 2019). arXiv: 1908.09948 (cit. on p. 71).

[Saj+18] Mehdi S. M. Sajjadi, Giambattista Parascandolo, Arash Mehrjou, and Bernhard Schölkopf. "Tempered Adversarial Networks". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. July 3, 2018, pp. 4451–4459 (cit. on p. 149).

[Sal+17] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications". In: *ICLR* (2017) (cit. on pp. 68, 125).

[Sca+09] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (Jan. 2009). Conference Name: IEEE Transactions on Neural Networks, pp. 61–80. ISSN: 1941-0093 (cit. on p. 121).

[Sch+18] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. "Towards the first adversarially robust neural network model on MNIST". In: *ICLR2019* (Sept. 20, 2018). arXiv: 1805.09190 (cit. on p. 104).

[Sch78] Gideon Schwarz. "Estimating the Dimension of a Model". In: *Annals of Statistics* 6.2 (Mar. 1978). Publisher: Institute of Mathematical Statistics, pp. 461–464. ISSN: 0090-5364, 2168-8966 (cit. on p. 24).

[She97] Prakash P. Shenoy. "Binary join trees for computing marginals in the Shenoy-Shafer architecture". In: *International Journal of Approximate Reasoning*. Uncertainty in AI (UAI'96) Conference 17.2 (Aug. 1, 1997), pp. 239–263. ISSN: 0888-613X (cit. on p. 17).

[Shu+18] Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. "Amortized Inference Regularization". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 4393–4402 (cit. on pp. 49, 50, 156).

[SJJ96] L. K. Saul, T. Jaakkola, and M. I. Jordan. "Mean Field Theory for Sigmoid Belief Networks". In: *Journal of Artificial Intelligence Research* 4 (Mar. 1, 1996), pp. 61–76. ISSN: 1076-9757 (cit. on pp. 20, 35).

[SM06] Tomi Silander and Petri Myllymäki. "A simple approach for finding the globally optimal Bayesian network structure". In: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*. UAI'06. Arlington, Virginia, USA: AUAI Press, July 13, 2006, pp. 445–452. ISBN: 978-0-9749039-2-7 (cit. on p. 24).

[SMH07] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering". In: *Proceedings of the 24th international conference on Machine learning*. ICML '07. New York, NY, USA: Association for Computing Machinery, June 20, 2007, pp. 791–798. ISBN: 978-1-59593-793-3 (cit. on p. 16).

[Sno+19] Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D. Sculley, Joshua Dillon, Jie Ren, and Zachary Nado. "Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d{\textbackslash}textquotesingle Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 13991–14002 (cit. on p. 103).

[Sø +16] Casper Kaae Sø nderby, Tapani Raiko, Lars Maaløe, Sø ren Kaae Sø nderby, and Ole Winther. "Ladder Variational Autoencoders". In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 3738–3746 (cit. on pp. 53, 55).

[STG13] Andreas Stuhlmüller, Jacob Taylor, and Noah Goodman. "Learning Stochastic Inverses". In: Advances in Neural Information Processing Systems. 2013, pp. 3048–3056 (cit. on p. 42).

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *arXiv:1409.1556 [cs]* (Apr. 10, 2015). arXiv: 1409.1556 (cit. on p. 65).

[TW18] Jakub Tomczak and Max Welling. "VAE with a VampPrior". In: *International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics. ISSN: 1938-7228 Section: Machine Learning. Mar. 31, 2018, pp. 1214–1223 (cit. on p. 46).

[Vah+18] Arash Vahdat, William Macready, Zhengbing Bian, Amir Khoshaman, and Evgeny Andriyash. "DVAE++: Discrete Variational Autoencoders with Overlapping Transformations". In: *International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228 Section: Machine Learning. July 3, 2018, pp. 5035–5044 (cit. on p. 48).

[VAM18] Arash Vahdat, Evgeny Andriyash, and William Macready. "DVAE#: Discrete Variational Autoencoders with Relaxed Boltzmann Priors". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 1864–1874 (cit. on p. 48).

[Vri12] Scott I. Vrieze. "Model selection and psychological theory: A discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC)". In: *Psychological Methods* 17.2 (2012). Place: US Publisher: American Psychological Association, pp. 228–243. ISSN: 1939-1463(Electronic),1082-989X(Print) (cit. on p. 24).

[Web+18] Stefan Webb, Adam Golinski, Rob Zinkov, Siddharth N, Tom Rainforth, Yee Whye Teh, and Frank Wood. "Faithful Inversion of Generative Models for Effective Amortized Inference". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 3070–3080 (cit. on p. 52).

[Wei00] Yair Weiss. "Correctness of Local Probability Propagation in Graphical Models with Loops". In: *Neural Computation* 12.1 (Jan. 1, 2000). Publisher: MIT Press, pp. 1–41. ISSN: 0899-7667 (cit. on p. 17).

[Wu+21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (Jan. 2021). Conference Name: IEEE Transactions on Neural Networks and Learning Systems, pp. 4–24. ISSN: 2162-2388 (cit. on p. 121).

[Xu+19] Ming Xu, Matias Quiroz, Robert Kohn, and Scott A. Sisson. "Variance reduction properties of the reparameterization trick". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. The 22nd International Conference on Artificial Intelligence and Statistics. ISSN: 2640-3498. PMLR, Apr. 11, 2019, pp. 2711–2720 (cit. on p. 43).

[XYA20] Zhisheng Xiao, Qing Yan, and Yali Amit. "Exponential Tilting of Generative Models: Improving Sample Quality by Training and Sampling from Latent Energy". In: *arXiv:2006.08100 [cs, stat]* (June 14, 2020). arXiv: 2006.08100 (cit. on p. 47).

[Yan05] Yuhong Yang. "Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation". In: *Biometrika* 92.4 (Dec. 1, 2005). Publisher: Oxford Academic, pp. 937–950. ISSN: 0006-3444 (cit. on p. 24).

[ZDM18] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. "Fixup Initialization: Residual Learning Without Normalization". In: International Conference on Learning Representations. Sept. 27, 2018 (cit. on p. 57).

[Zha+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning requires rethinking generalization". In: *ICLR 2017* (Feb. 26, 2017). arXiv: 1611.03530 (cit. on p. 91).

[Zha+18] Xiaoou Monica Zhang, Katarina Grolinger, Miriam A. M. Capretz, and Luke Seewald. "Forecasting Residential Energy Consumption: Single Household Perspective". In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). Dec. 2018, pp. 110–117 (cit. on p. 115).

[Zhe+19] Huangjie Zheng, Jiangchao Yao, Ya Zhang, Ivor W. Tsang, and Jia Wang. "Understanding VAEs in Fisher-Shannon Plane". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (01 July 17, 2019), pp. 5917–5924. ISSN: 2374-3468 (cit. on p. 73).

[ZP94] Nevin Lianwen Zhang and David Poole. "A simple approach to Bayesian network computations". In: *Proc. of the Tenth Canadian Conference on Artificial Intelligence*. 1994 (cit. on p. 17).

[ZSE18] Shengjia Zhao, Jiaming Song, and Stefano Ermon. *The Information Autoencoding Family: A Lagrangian Perspective on Latent Variable Generative Models*. July 7, 2018. arXiv: 1806.06514 [cs, stat]. URL: http://arxiv.org/abs/1806.06514 (visited on 09/15/2021) (cit. on p. 73).

université PARIS-SACLAY
DÉPARTEMENT
Sciences et Technologies de l'Information et de la Communication

Titre : Modèles à Variables Latentes Profonds : des propriétés aux structures

Mots clés : Modèles Génératifs, Apprentissage Profond, Modèles à Variables Latentes, Réseaux Bayésiens

Résumé : Les Modèles à Variables Latentes Profonds sont des modèles génératifs combinant les Réseaux Bayésiens avec l'apprentissage profond, illustrés par le célèbre Auto-encodeur Variationnel. Cette thèse se focalise sur leur structure, entendue comme la combinaison de 3 aspects : le graphe du Réseau Bayésien, le choix des familles probabilistes des variables, et l'architecture des réseaux de neurones. Nous démontrons que de nombreux aspects et propriétés de ces modèles peuvent être compris et contrôlés par cette structure, sans altérer l'objectif d'entraînement construit sur l'Evidence Lower Bound.

La première contribution concerne l'impact du modèle d'observation – la modélisation probabiliste des variables observées – sur le processus d'entraînement : comment il détermine la séparation entre signal et bruit, ainsi que son impact sur la dynamique de l'entraînement lorsque son paramètre d'échelle est appris plustôt que fixé, où il agit alors comme un processus de recuit simulé.

La seconde contribution, CompVAE, est centrée sur la structure hiérarchique des variables latentes : un modèle génératif conditionné par un multi-ensemble d'élements à combiner dans la génération finale. CompVAE démontre comment des propriétés globales – des manipulations ensemblistes dans ce cas – peuvent être atteintes par la seule conception structurale. Ce modèle est de plus validé empiriquement sur des données réelles, pour la génération de courbes de consommation électrique.

La troisième contribution, Boltzmann Tuning of Generative Models (BTGM), est un cadre permettant d'ajuster un modèle génératif pré-entraîné selon un critère extérieur, en trouvant les ajustements minimaux nécessaire. Ceci est fait tout en contrôlant finement quelles variables latentes sont ajustées, et comment elles le sont. Nous démontrons empiriquement comment BTGM peut être utilisé pour spécialiser un modèle déjà entraîné, ou pour explorer les parties extrêmes d'une distribution générée.

Title: Deep Latent Variable Models: from properties to structures

Keywords: Generative Models, Deep Learning, Latent Variable Models, Bayesian Networks

Abstract: Deep Latent Variable Models are generative models combining Bayesian Networks and deep learning, illustrated by the renowned Variational Autoencoder. This thesis focuses on their structure, understood as the combination of 3 aspects: the Bayesian Network graph, the choice of probability distribution families for the variables, and the neural architecture. We show that and how several aspects and properties of those models can be understood and controlled through this structure, without altering the training objective constructed from the Evidence Lower Bound.

The first contribution concerns the impact of the observation model – the probabilistic modeling of the observed variables – on the training process: how it determines the demarcation between signal and noise and its impact on training dynamic when its scale parameter is learned rather than fixed. It then behaves similarly to a simulated annealing process.

The second contribution, CompVAE, is centered on the hierarchical structure of latent variables: a generative model conditioned by a multi-set of elements to be combined in the final generation. CompVAE demonstrates how global properties – ensemblist manipulations in this case – can be achieved by solely structural design. The model is furthermore empirically validated on real data to generate electrical consumption curves.

The third contribution, Boltzmann Tuning of Generative Models (BTGM), is a framework for adjusting trained generative models according to an externally provided criterion while finding the minimal required adjustments. This is done while finely controlling which latent variables are adjusted and how the are. We empirically demonstrate how BTGM can be used to specialize a trained model or to explore the extreme parts of a generative distribution.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France