Dynamical Variational Autoencoders: A Comprehensive Review

Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, Xavier Alameda-Pineda

The version of record is available at:
http://dx.doi.org/10.1561/2200000089

Dynamical Variational Autoencoders: A Comprehensive Review

Laurent Girin[1], Simon Leglaive[2], Xiaoyu Bie[3], Julien Diard[4], Thomas Hueber[1] and Xavier Alameda-Pineda[3]

[1] *Univ. Grenoble Alpes, CNRS, Grenoble-INP, GIPSA-lab; laurent.girin@grenoble-inp.fr, thomas.hueber@grenoble-inp.fr*
[2] *CentraleSupélec, IETR; simon.leglaive@centralesupelec.fr*
[3] *Inria, Univ. Grenoble Alpes, CNRS, LJK; xiaoyu.bie@inria.fr, xavier.alameda-pineda@inria.fr*
[4] *Univ. Grenoble Alpes, CNRS, LPNC; julien.diard@univ-grenoble-alpes.fr*

ABSTRACT

Variational autoencoders (VAEs) are powerful deep generative models widely used to represent high-dimensional complex data through a low-dimensional latent space learned in an unsupervised manner. In the original VAE model, the input data vectors are processed independently. Recently, a series of papers have presented different extensions of the VAE to process sequential data, which model not only the latent space but also the temporal dependencies within a sequence of data vectors and corresponding latent vectors, relying on recurrent neural networks or state-space models. In this paper, we perform a literature review of these models. We introduce and discuss a general class of models, called dynamical variational autoencoders (DVAEs), which

Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber and Xavier Alameda-Pineda (2021), "Dynamical Variational Autoencoders: A Comprehensive Review", Foundations and Trends® in Machine Learning: Vol. 15, No. 1-2, pp 1–175. DOI: 10.1561/2200000089.

encompasses a large subset of these temporal VAE extensions. Then, we present in detail seven recently proposed DVAE models, with an aim to homogenize the notations and presentation lines, as well as to relate these models with existing classical temporal models. We have reimplemented those seven DVAE models and present the results of an experimental benchmark conducted on the speech analysis-resynthesis task (the PyTorch code is made publicly available). The paper concludes with a discussion on important issues concerning the DVAE class of models and future research guidelines.

Contents

1

Introduction

Deep Generative Models (DGMs) constitute a large family of probabilistic models that are currently of high interest in the machine learning and signal processing. They result from the combination of conventional (i.e., non-deep) generative probabilistic models and Deep Neural Networks (DNNs). For both conventional models and DGMs, different nonconflicting taxonomies can be established due to the domain richness and percolation across the different approaches. Nevertheless, these models can be grossly classified into the following two categories. Using the terminology of Diggle and Gratton (1984), the first category corresponds to *prescribed* models for which the probability density function (pdf) of the generative model is defined explicitly, generally through a parametric form. The second category corresponds to *implicit* models that can generate data "directly," without using an explicit formulation and manipulation of a pdf model. Generative adversarial networks (GANs) are a popular example of this second category (Goodfellow *et al.*, 2014; Goodfellow *et al.*, 2016; Goodfellow, 2016).

In the present review, we focus on the first category, in which a parametric pdf model is used. A suitable feature of generative models based on an explicit formulation of the pdf is that they can be easily

plugged into a more general Bayesian framework, not only for generating data but also for modeling the data structure (without actually generating them) in various applications (e.g., data denoising or data transformation). In any case, the pdf model must be as close as possible to the true pdf of the data, which is generally unknown. To achieve this aim, the model must be trained from data, and model parameters are generally estimated by following the maximum likelihood methodology (Goodfellow *et al.*, 2016; Bishop, 2006; Koller and Friedman, 2009). These principles are valid for both conventional generative models and DGMs; however, in the case of DGMs, the pdf parameters are generally the output of DNNs, which makes model training potentially difficult.

1.1 Deep Dynamical Bayesian Networks

In the present review, we focus on an important subfamily of DGMs, namely the deep dynamical Bayesian networks (DDBNs), which are built on the following models:

- Bayesian networks (BNs) are a popular class of probabilistic models for which i) the dependencies among all involved random variables are explicitly represented by conditional pdfs (i.e. BNs are prescribed models), and ii) these dependencies can be schematically represented using a directed acyclic graph (Bishop, 2006; Koller and Friedman, 2009). The structure of these dependencies often reflects (or originates from) an underlying hierarchical generative process.

- Dynamical Bayesian networks are BNs that include temporal dependencies and are widely used to model dynamical systems and/or data sequences. Dynamical BNs are BNs "repeated over time"; that is, they exhibit a repeating dependency structure (a time-slice at discrete time t) and some dependencies across these time-slices (the dynamical models). Recurrent neural networks (RNNs) and state-space models (SSMs) can be considered special cases of dynamical BNs. In fact, a temporal dependency in a dynamical BN is often implemented either as a deterministic recursive process, as in RNNs, or as a first-order Markovian process,

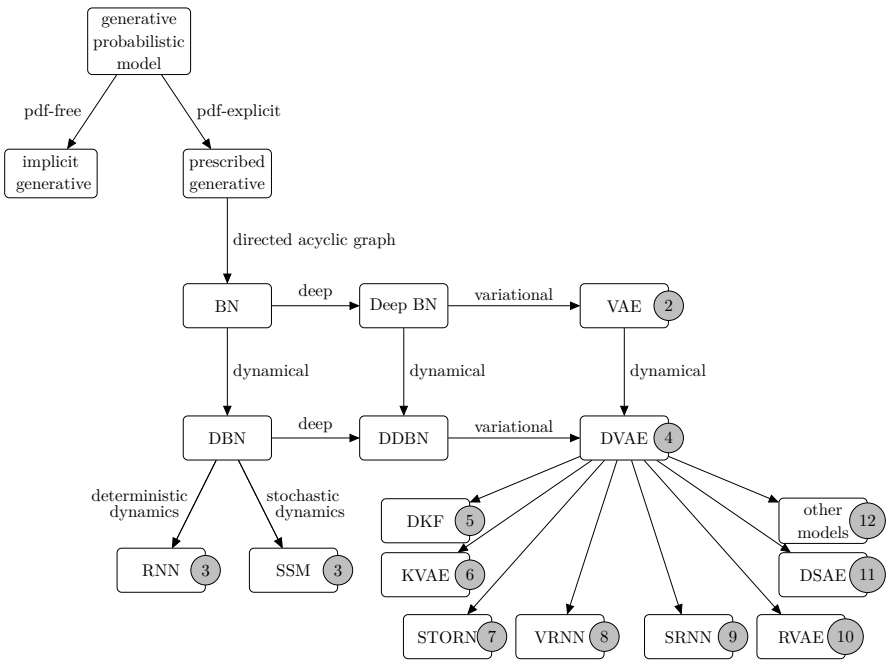


Figure 1.1: A graphical taxonomy of generative probabilistic models. Only the branch corresponding to the models covered in this review is detailed. Nodes represent classes of models, and arrow labels specify some of the relationships between the classes of models. Please refer to the text for details and acronym definitions. The numbered gray circles indicate the section number in which the corresponding class of models is detailed.

as in usual SSMs.

- Deep Bayesian networks combine BNs with DNNs. DNNs are used to generate the parameters of the modeled distributions. This enables them to be high-dimensional and highly multi-modal while having a reasonable number of parameters. In short, deep BNs have can appropriately combine the "explainability" of Bayesian models with the modeling power of DNNs.

DDBNs are thus a combination of all these aspects, as illustrated in Figure 1.1. They can be equally seen as dynamical versions of deep BNs (i.e., deep BNs including temporal dependencies) or deep versions of dynamical BNs (i.e., dynamical BNs mixed with DNNs). As an extension of dynamical BNs, DDBNs are expected to be powerful tools for modeling dynamical systems and/or data sequences. However, as mentioned above, the combination of probabilistic modeling with DNNs in deep BNs can result in a complex and costly model training. This is an even more serious issue for DDBNs, in which the repeating structure due to temporal modeling adds a level of complexity.

1.2 Variational inference and VAEs

Recently, the application of the variational inference methodology (Jordan *et al.*, 1999; Bishop, 2006, Chapter 10; Šmídl and Quinn, 2006; Murphy, 2012, Chapter 21) to a fundamental deep BN architecture –a low-dimensional to a high-dimensional generative feed-forward DNN– has led to efficient inference and training of the resulting model, called a variational autoencoder (VAE) (Kingma and Welling, 2014). A similar approach was proposed the same year (Rezende *et al.*, 2014).[1] The VAE is directly connected to the concepts of a *latent* variable and *unsupervised representation learning*: the observed random variable representing the data of interest is assumed to be generated from an unobserved latent variable through a probabilistic process. Often, this latent variable is of lower dimension than the observed data (which can be high-dimensional) and is assumed to "encode" the observed data in

[1] Kingma and Welling (2014) and Rezende *et al.* (2014) were both pre-published in 2013 as ArXiv papers. Connections also exist with Mnih and Gregor (2014).

a compact manner so that new data can be generated from new values of the latent variable. Moreover, one wishes to extract a latent representation that is *disentangled* (i.e., different latent coefficients encode different properties or different factors of variation in the data). When successful, this provides good interpretability and control of the data generation/transformation process.

The automatic discovery of a latent space structure is part of the model training process. The inference process, which is defined in the present context as the estimation of latent variables from the observed data, also plays a major role. As presented in detail later, in a deep BN, the exact posterior distribution (i.e., the posterior distribution of the latent variable given the observed variable corresponding to the generative model) is generally not tractable. It is thus replaced with a parametric approximate posterior distribution (i.e., an inference model) that is implemented with a DNN. As the observed data likelihood function is also not tractable, the model parameters are estimated by chaining the inference model (also known as the *encoder* in the VAE framework) and the generative model (the *decoder*) and maximizing a lower bound of the log-likelihood function, called the variational lower bound (VLB), over a training dataset.[2] Hereinafter, we refer to this general variational inference and training methodology as *the VAE methodology*.

In summary, the VAE methodology enables deep unsupervised representation learning while providing efficient inference and parameter estimation in a Bayesian framework. As a result, the seminal papers by Kingma and Welling (2014) and Rezende *et al.* (2014) have had and continue to have a strong impact on the machine learning community. VAEs have been applied to many signal processing problems, such as the generation and transformation of images and speech signals (we provide a few references in Chapter 2).

[2]The idea of using an artificial neural network to approximate an inference model and chaining the encoder and decoder dates back to the early studies of Hinton *et al.* (1995) and Dayan *et al.* (1995). However, the algorithms presented in these papers for model training are different from the one used to optimize the VAE.

1.3 Dynamical VAEs

As a deep BN, the original VAE proposed by Kingma and Welling (2014) did not include temporal modeling. This means that each data vector was processed independently of the other data vectors (and the corresponding latent vector was also processed independently of the other latent vectors). This is clearly suboptimal for the modeling of correlated (temporal) vector sequences.

In the years following the publication of Kingma and Welling (2014) and Rezende *et al.* (2014), the VAE methodology was extended and successfully applied to several more complex deep BNs. In particular, it was applied to deep BNs with a temporal model (i.e., DDBNs) dedicated to the modeling of sequential data exhibiting temporal correlation. In the present review, we are particularly interested in the models presented in the following papers: (Bayer and Osendorfer, 2014; Krishnan *et al.*, 2015; Chung *et al.*, 2015; Gu *et al.*, 2015; Fraccaro *et al.*, 2016; Krishnan *et al.*, 2017; Fraccaro *et al.*, 2017; Goyal *et al.*, 2017; Hsu *et al.*, 2017b; Li and Mandt, 2018; Leglaive *et al.*, 2020). In addition to including temporal dependencies, the unsupervised representation learning essence of the VAE is preserved and cherished in these studies. These DDBNs combine the observed and latent variables and aim at modeling not only data dynamics but also discovering the latent factors governing them.

To achieve this aim, these models are trained using the VAE methodology (i.e., design of an inference model and maximization of the corresponding VLB). We can thus encompass these models under the common class and terminology of *variational DDBNs* (i.e., DDBNs immersed in the VAE framework). In the following of the paper, as well as the title, we prefer to refer to them as *dynamical VAEs* (DVAEs) (i.e., VAEs including a temporal model for modeling sequential data). This is simply because we assume that the term "VAE" is currently more popular than the term "DBN," and "dynamical VAEs" gives a more speaking-first evocation of these models, compared to "variational DDBNs." This convergence of DDBNs and VAEs into DVAEs is illustrated in Figure 1.1.

In practice, these different DVAE models vary in how they define the dependencies between the observed and latent variables, how they

define and parameterize the corresponding generative pdfs, and how they define and parameterize the inference model. They also differ in how they combine the variables with RNNs to model temporal dependencies, at both generation and inference. In contrast, they are all characterized by the following common set of features.

First, as stated above, they are all trained using the VAE methodology, possibly with a few adaptations and refinements. In this paper, we do not review models based on GANs and, more generally, on adversarial training. Examples of extensions of "static" GANs to sequence modeling and generation can be found in the literature (Mathieu *et al.*, 2016; Villegas *et al.*, 2017; Denton and Birodkar, 2017; Tulyakov *et al.*, 2018; Lee *et al.*, 2018). This approach is particularly popular for separating content and motion in videos.

Second, even if the observed random vectors can be continuous or discrete, as in the original VAE formulation, they all feature *continuous latent random variables*. In the present review, we do not consider the case of discrete latent random variables. The latter can be incorporated in DVAE models, in the line with the case of, for example, conditional VAEs (Sohn *et al.*, 2015; Zhao *et al.*, 2017). Temporal models with binary observed and latent random variables have been proposed (Boulanger-Lewandowski *et al.*, 2012; Gan *et al.*, 2015). These models are based on restricted Boltzman machines (RBMs) or sigmoid belief networks (SBNs) combined with RNNs. A detailed analysis of such models is beyond the scope of the present review.

Third, all DVAE models we consider feature *a discrete-time sequence of (continuous or discrete) observed random vectors associated with a corresponding discrete-time sequence of (continuous) latent random vectors*. In other words, these models function in a *sequence-to-sequence mode for both encoding and decoding*. Thus, we do not focus on VAE-based models specifically designed for text and dialogue generation (Bowman *et al.*, 2016; Miao *et al.*, 2016; Serban *et al.*, 2016; Serban *et al.*, 2017; Yang *et al.*, 2017; Semeniuta *et al.*, 2017; Hu *et al.*, 2017; Zhao *et al.*, 2018; Jang *et al.*, 2019) or (2D) image modeling (Gulrajani *et al.*, 2016; Chen *et al.*, 2017; Lucas and Verbeek, 2018; Shang *et al.*, 2018). These models generally have a many-to-one encoder and a one-to-many decoder; that is, a long sequence of data (e.g., words or pixels) is

encoded into a single latent vector, which is in turn decoded into a whole data sequence (see also Roberts *et al.* (2018) and Pereira and Silveira (2018) for examples on music score modeling and anomaly detection in energy time series, respectively). Even if those models can include a hierarchical structure at encoding and/or at decoding, they do not consider a temporal sequence of latent vectors.

All these latent-variable deep temporal models, the ones we detail and unify in the DVAE class, and the ones we do not detail, remain strongly connected, with a similar overall encoding-decoding architecture and possibly a similar inference and training VAE methodology. Therefore, we must keep in mind that some of the propositions made in the literature for one type of model can be adapted and be beneficial to the other.

1.4 Aim, contributions, and outline of the paper

This paper aims to provide a comprehensive overview of DVAE models. The contributions of this paper are detailed as follows.

We provide a formal definition of the general class of DVAEs. We describe its main properties and characteristics and how this class is related to previous classical models, such as VAEs, RNNs, and SSMs. We discuss the structure of dependencies between the observed and latent random variables in DVAE pdfs, as well as how these dependencies are implemented with neural networks. We discuss the design of inference models considering the general methodology used to identify the actual dependencies of the latent variables at inference time. We also discuss the VLB computation for training DVAEs. All these points are presented in Chapter 4. To the best of our knowledge, this is the first time this class of models has been presented in such a general and unified manner.

We provide a detailed and complete technical description of seven DVAE models selected from the literature. In Chapter 5, we start with the deep Kalman filter (DKF) (Krishnan *et al.*, 2015; Krishnan *et al.*, 2017), which is a basic combination of an SSM with DNNs. Then, we examine the Kalman variational autoencoder (KVAE) (Fraccaro *et al.*, 2017) in Chapter 6, the stochastic recurrent neural

network (STORN) (Bayer and Osendorfer, 2014) in Chapter 7, the variational recurrent neural network (VRNN) (Chung *et al.*, 2015; Goyal *et al.*, 2017) in Chapter 8, another type of stochastic recurrent neural network (SRNN) (Fraccaro *et al.*, 2016) in Chapter 9, the recurrent variational autoencoder (RVAE) (Leglaive *et al.*, 2020) in Chapter 10, and finally the disentangled sequential autoencoder (DSAE) (Li and Mandt, 2018) in Chapter 11.

We have spent effort on consistency of presentation. For all seven models that we detail, we first present the generative equations in time-step form and then for an entire data sequence. Then, we present the structure of the exact posterior distribution of the latent variables given the observed data and present the inference model as proposed in the original papers. Finally, we present the corresponding VLB. In the original papers, some parts of this complete picture are often overviewed or even missing (not always the same parts), independently of the authors' goodwill, because of lack of space.

We discuss the links, similarities, and differences of the selected DVAE models. We comment on the choices of the authors of the reviewed papers regarding the inference model, its relation to the exact posterior distribution, and implementation issues. In the present review, we discuss only high-level implementation issues related to the general structure of the neural network that implements a given DVAE at generation or inference (e.g., the type of RNN). We do not discuss practical implementation issues (e.g., the number of layers), which are too low-level in the present technical review context.

We have also spent some effort making the notations homogeneous across all models. This is valid for both the review of the seven detailed models and the other sections of the paper, including the general presentation of the DVAE class of models in Chapter 4. For some models, we have changed the time indexation notation, and in some instances, the names of some variables compared to the original papers. We have taken great care to do that consistently in the generative model, the inference part, and the VLB so that these notation changes do not affect the essence and functioning of the model. Together with consistency of presentation, this enables us to better put in evidence the commonalities and differences across models and make their comparison easier.

Notation remarks are specified in independent dedicated paragraphs throughout the paper to facilitate connections with the original papers.

In complement to the detailed review of the seven selected models, **we provide a more rapid overview of other DVAE models presented in the recent literature** in Chapter 12.

We relate the recent developments in DVAEs to the history and technical background of the classical models DVAEs are built on, namely VAEs, RNNs, and SSMs. Although there are already many papers on VAEs, including tutorials, we present them in Chapter 2 because all subsequent DVAE models rely on the VAE methodology. Then, as the introduction of temporal models in the VAE framework is closely linked to RNNs and SSMs, we briefly present these two classes of models in Chapter 3. The unified notation that we use will help readers from different communities (e.g., machine learning, signal processing, and control theory) who are not familiar with the relations among VAEs, RNNs, and SSMs to discover them comfortably.

We provide a quantitative benchmark of the selected DVAE models in an analysis-resynthesis task, as well as qualitative examples of data generation. We have reimplemented the seven DVAE models detailed in this review and evaluated them on two different datasets (speech signals and 3D human motion data). This benchmark is presented in Chapter 13.

The performance comparison of the different models from the literature review is a difficult task for many reasons. First, all models are not evaluated on the same data. Then, a newly proposed model generally performs better than some previously proposed model(s), at least on some aspect(s), but this can depend on model tuning, task, data, and experimental setup. Moreover, the comparison performed with a subset of previous models is incomplete in essence. In short, an extended benchmark of DVAE models is not yet available in the literature. Conducting an extended benchmark is a huge endeavor, as there are many possible configurations for the models and many tasks for evaluating them. In particular, it is not yet clear how to evaluate the degree of "disentanglement" of the extracted latent space. The presented

experiments are a first step in that direction. We plan to exploit and compare the models more extensively and on more complex tasks in future studies. For example, we compared three models in the recently proposed DVAE-based unsupervised speech enhancement method (Bie *et al.*, 2021).

The code reimplementing the seven DVAE models and used on the benchmark task is made available to the community. A link to the open-source code and the best-trained models can be found at https://team.inria.fr/robotlearn/dvae/. We have also taken care, in the code, to follow the unified presentation and notation used in the paper, making it, hopefully, a useful and pedagogical resource.

We provide a discussion to put the DVAE class of models into perspective. We summarize the outcome of this review and discuss the future challenges and possible improvements of VAEs and DVAEs. This is presented in Chapter 14.

In summary, we believe that comparison of models across papers is a difficult task in essence, regardless of the efforts spent by the authors of the original papers, because of the use of different notations, presentation lines, missing information, etc. We hope that the present review paper and accompanying code will enable the readers to access the technical substance of the different DVAE models, their connections with classical models, their cross-connections, and their unification in the DVAE class more rapidly and "comfortably" than by analyzing and comparing the original papers by themselves.

We wish the reader to enjoy this DVAE tour.

2

Variational Autoencoders

In this section, we present the VAE and the associated methodology for model training and approximate posterior distribution estimation (i.e., inference) with variational methods (Kingma and Welling, 2014; Rezende *et al.*, 2014). An extended tutorial on VAEs can be found in Kingma and Welling's (2019) paper.

2.1 Principle

For clarity of presentation, let us start with an autoencoder (AE). As illustrated in Figure 2.1, an AE is a DNN that is trained to replicate an input vector $\mathbf{x} \in \mathbb{R}^F$ at the output (Hinton and Salakhutdinov, 2006; Vincent *et al.*, 2010). At training time, the target output is thus set equal to $\mathbf{x}$, and at test time, the output $\hat{\mathbf{x}}$ is an estimated value of $\mathbf{x}$ (i.e., we have $\hat{\mathbf{x}} \approx \mathbf{x}$). An AE usually has a diabolo shape. The left part of the AE, the *encoder*, provides a low-dimensional latent representation $\mathbf{z} \in \mathbb{R}^L$ of the data vector $\mathbf{x}$, with $L \ll F$, at the so-called bottleneck layer. The right part of the AE, the *decoder*, tries to reconstruct $\mathbf{x}$ from $\mathbf{z}$. So far, everything is deterministic: At test time, each time the AE is fed with a specific input vector $\mathbf{x}_0$, it will provide the same corresponding output $\hat{\mathbf{x}}_0$.

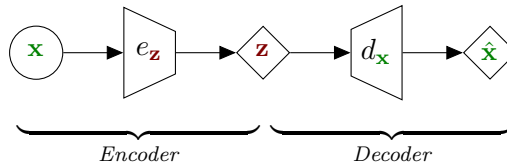


Figure 2.1: Schematic representation of an AE. The left trapezoid represents a high-to-low-dimensional encoder DNN (denoted $e_{\mathbf{z}}$), and the right trapezoid represents a low-to-high-dimensional decoder DNN (denoted $d_{\mathbf{x}}$). Calculation of the latent variable $\mathbf{z}$ and output $\hat{\mathbf{x}}$ from input $\mathbf{x}$ is deterministic. In line with probabilistic graphical models, deterministic variables are represented within diamonds.

The VAE was initially proposed by Kingma and Welling (2014) and Rezende *et al.* (2014). It can be seen as a probabilistic version of an AE, where the output of the decoder is not directly a value of $\mathbf{x}$ but the parameters of a probability distribution of $\mathbf{x}$. As shown below, the same probabilistic formulation applies to the encoding of $\mathbf{z}$. The resulting probabilistic model can be used to generate new data from new values of $\mathbf{z}$. It can also be used to transform existing data within an encoding-modification-decoding scheme. For instance, the seminal papers on VAEs and many subsequent ones have considered image generation and transformation. Examples of speech/music signals transformation based on a VAE can be found in the literature (Blaauw and Bonada, 2016; Hsu *et al.*, 2017a; Esling *et al.*, 2018; Roche *et al.*, 2019; Bitton *et al.*, 2020). Finally, it can be employed as a prior distribution of $\mathbf{x}$ in more complex Bayesian models for, for example, speech enhancement (Bando *et al.*, 2018; Leglaive *et al.*, 2018; Pariente *et al.*, 2019; Leglaive *et al.*, 2019) or source separation (Kameoka *et al.*, 2018).

For clarity of presentation, at this point, it is convenient to separate the presentation of the VAE decoder (i.e., the generative model) and that of the VAE encoder (i.e., the inference model).

2.2 VAE generative model

In the following, $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, $\mathrm{diag}\{\cdot\}$ is the operator that forms a diagonal matrix from a vector by putting the vector entries

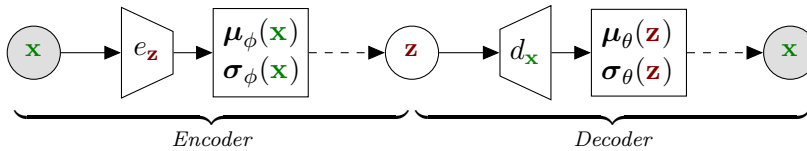


Figure 2.2: Schematic representation of the VAE: Encoder (left) and decoder (right). Dashed lines represent a sampling process. In line with probabilistic graphical models, latent variables are represented within empty circles and observed variables are represented within shaded circles. When the encoder and decoder are cascaded, using the same variable name $\mathbf{x}$ at both the input and output is an abuse of notation, but this is done to be more consistent with the separate encoder and decoder equations.

on the diagonal, $\mathbf{0}_L$ is the zero-vector of size L, and $\mathbf{I}_L$ is the identity matrix of size L. $p_{\theta_\mathbf{x}}(\mathbf{x})$ is a generic notation for a parametric pdf of the random variable $\mathbf{x}$, where $\theta_\mathbf{x}$ is the set of parameters. It is equivalent to $p(\mathbf{x};\theta_\mathbf{x})$.

Formally, the VAE decoder is defined by

$$p_\theta(\mathbf{x},\mathbf{z}) = p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z})p_{\theta_\mathbf{z}}(\mathbf{z}), \tag{2.1}$$

with

$$p_{\theta_\mathbf{z}}(\mathbf{z}) = \mathcal{N}(\mathbf{z};\mathbf{0}_L,\mathbf{I}_L), \tag{2.2}$$

and $p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z})$ is a parametric conditional distribution, the parameters of which are a nonlinear function of $\mathbf{z}$ modeled by a DNN. This DNN is called the *decoder network*, or the *generation network*, and is parametrized by a set of weights and biases denoted $\theta_\mathbf{x}$. In the standard VAE, the set of parameters $\theta_\mathbf{z}$ is empty, but we write it explicitly to be coherent with the rest of the paper, and we have here $\theta = \theta_\mathbf{x} \cup \theta_\mathbf{z} = \theta_\mathbf{x}$. The decoder network is illustrated in Figure 2.2 (right).

The VAE model and associated variational methodology was introduced by Kingma and Welling (2014) in the general framework of parametric distributions, independently of the practical choice of the pdf $p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z})$ (and to a lesser extend of $p_{\theta_\mathbf{z}}(\mathbf{z})$). The observed variable $\mathbf{x}$ can be a continuous or discrete random variable with any arbitrary conditional distribution. The Gaussian case was then presented by Kingma and Welling (2014) as a major example. Of course, other pdfs (rather

than Gaussian) can be used depending on the nature of the data vector $\mathbf{x}$. For example, Gamma distributions better fit the natural statistics of speech/audio power spectra (Girin *et al.*, 2019). For simplicity of presentation and consistency across models, in the present review, $p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z})$ is assumed to be a Gaussian distribution with diagonal covariance matrix for all models; that is,

$$p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{z}), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{x}}(\mathbf{z})\}) \tag{2.3}$$

$$= \prod_{f=1}^{F} p_{\theta_\mathbf{x}}(x_f|\mathbf{z}) = \prod_{f=1}^{F} \mathcal{N}(x_f; \mu_{\theta_\mathbf{x},f}(\mathbf{z}), \sigma^2_{\theta_\mathbf{x},f}(\mathbf{z})), \tag{2.4}$$

where the subscript f denotes the f-th entry of a vector, and $\boldsymbol{\mu}_{\theta_\mathbf{x}} : \mathbb{R}^L \mapsto \mathbb{R}^F$ and $\boldsymbol{\sigma}_{\theta_\mathbf{x}} : \mathbb{R}^L \mapsto \mathbb{R}^F_+$ are nonlinear functions of $\mathbf{z}$ modeled by the decoder DNN. Although from a mathematical perspective, we could choose to work with full covariance matrices, assuming diagonal covariance matrices is preferable for computational reasons, since the number of free parameters of a covariance matrix grows quadratically with the variable dimension. This is problematic not only because we need to learn the neural network that computes all these parameters but also because covariance matrices often need to be inverted. The use of full covariance matrices also requires choosing an appropriate representation, for instance, based on the Cholesky decomposition. Please refer to Section 2.5.1 of Kingma and Welling (2019) for an extended discussion on this topic.

For maintaining consistency with the presentation of the other models in the next sections, we gather into $d_\mathbf{x}$ the functions implemented by the decoder DNN; that is,

$$[\boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{z}), \boldsymbol{\sigma}_{\theta_\mathbf{x}}(\mathbf{z})] = d_\mathbf{x}(\mathbf{z}). \tag{2.5}$$

A VAE decoder can be considered a generalization of the probabilistic principal component analysis (PPCA) (Tipping and Bishop, 1999) with a nonlinear (instead of linear) relationship between $\mathbf{z}$ and the parameters $\theta_\mathbf{x}$. It can also be considered the generalization of a generative mixture models, with a continuous conditional latent variable instead of a discrete one (Kingma and Welling, 2019). Indeed, the marginal distribution of

$\mathbf{x}$, $p_\theta(\mathbf{x})$, is given by

$$p_\theta(\mathbf{x}) = \int p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z}) p_{\theta_\mathbf{z}}(\mathbf{z}) d\mathbf{z}. \tag{2.6}$$

As any conditional distribution $p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z})$ can provide a mode, $p_\theta(\mathbf{x})$ can be highly multimodal (in addition to being potentially high-dimensional). Unlike PPCA, in VAEs, the posterior distribution cannot be written analytically and has to be approximated, as discussed in the next section.

2.3 Learning with variational inference

Training the generative model defined in (2.1)–(2.3) amounts to estimating the parameters θ so as to minimize the Kullback-Leibler (KL) divergence between the *true data distribution* $p^\star(\mathbf{x})$ and the *model distribution* (i.e., the marginal likelihood) $p_\theta(\mathbf{x})$:

$$\begin{aligned} &\min_\theta \Big\{ D_{\mathrm{KL}}(p^\star(\mathbf{x}) \parallel p_\theta(\mathbf{x})) = \mathbb{E}_{p^\star(\mathbf{x})}\big[\log p^\star(\mathbf{x}) - \log p_\theta(\mathbf{x})\big] \Big\} \\ \Leftrightarrow &\max_\theta \mathbb{E}_{p^\star(\mathbf{x})}\big[\log p_\theta(\mathbf{x})\big]. \end{aligned} \tag{2.7}$$

This equivalence relation shows that this definition of model training actually corresponds to the maximum (marginal) likelihood parameter estimation. In practice, the true data distribution $p^\star(\mathbf{x})$ is unknown, but we assume the availability of a training dataset $\mathbf{X} = \{\mathbf{x}_n \in \mathbb{R}^F\}_{n=1}^N$, where the training examples $\mathbf{x}_n$ are independent and identically distributed (i.i.d.) according to $p^\star(\mathbf{x})$. Following the principle of empirical risk minimization, where the risk is here defined as the negative log-marginal likelihood, the intractable expectation in (2.7) is replaced by a Monte Carlo estimate:

$$\max_\theta \frac{1}{N} \sum_{n=1}^N \log p_\theta(\mathbf{x}_n), \qquad \mathbf{x}_n \overset{i.i.d.}{\sim} p^\star(\mathbf{x}). \tag{2.8}$$

The estimated model parameters can then be used, for example, to generate new data from (2.1).

For many generative models with latent variables, directly solving this optimization problem is difficult, if not impossible when the marginal likelihood is analytically intractable, because of the integral

in (2.6), which cannot be computed in closed form. For the VAE, this intractability arises from the nonlinear relationship between the latent and observed variables, the latter being generated from the former through a DNN, which makes $p_{\theta_{\mathbf{x}}}(\mathbf{x}|\mathbf{z})$ in (2.6) a nonlinear function of $\mathbf{z}$. One standard approach then involves leveraging the latent variable nature of the model in to maximize a lower bound of the intractable log-marginal likelihood (Neal and Hinton, 1998), which precisely depends on the *posterior distribution* of the latent variables or its approximation. This strategy leads to the expectation-maximization (EM) algorithm (Dempster *et al.*, 1977) and its variants when the posterior distribution is intractable, such as Monte Carlo EM (Wei and Tanner, 1990) and variational EM (Jordan *et al.*, 1999) algorithms.

As the name suggests, a VAE builds upon variational inference techniques, the general principles of which will now be briefly reviewed. Let $\mathcal{F}$ denote a *variational family* defined as a set of pdfs over the latent variables $\mathbf{z}$. For any variational distribution of pdf $q(\mathbf{z}) \in \mathcal{F}$, the following decomposition of the log-marginal likelihood holds (Neal and Hinton, 1998):

$$\log p_\theta(\mathbf{x}) = \mathcal{L}(\theta, q(\mathbf{z}); \mathbf{x}) + D_{\mathrm{KL}}\big(q(\mathbf{z}) \parallel p_\theta(\mathbf{z}|\mathbf{x})\big), \tag{2.9}$$

where $\mathcal{L}(\theta, q(\mathbf{z}); \mathbf{x})$ is referred to in the literature as the evidence lower bound (ELBO), the negative variational free energy, or the VLB, and is defined as

$$\mathcal{L}(\theta, q(\mathbf{z}); \mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\big[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z})\big] \leq \log p_\theta(\mathbf{x}). \tag{2.10}$$

The inequality in (2.10) is obtained from (2.9) based on the fact that $D_{\mathrm{KL}}(\cdot \parallel \cdot) \geq 0$. Equality holds (i.e., the VLB is tight to the log-marginal likelihood) if and only if the variational distribution $q(\mathbf{z})$ is equal to the exact posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$.

The EM algorithm (Dempster *et al.*, 1977) is an iterative algorithm that consists in alternatively maximizing the VLB with respect to $q(\mathbf{z}) \in \mathcal{F}$ in the E-step and with respect to θ in the M-step (Neal and Hinton, 1998). From (2.9), we see that the E-step involves finding the variational distribution $q(\mathbf{z})$ in the variational family $\mathcal{F}$ that best approximates the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ according to the KL divergence

measure of fit:

$$q^\star(\mathbf{z}) = \arg\max_{q \in \mathcal{F}} \mathcal{L}(\theta, q(\mathbf{z}); \mathbf{x}) = \arg\min_{q \in \mathcal{F}} D_{\mathrm{KL}}\big(q(\mathbf{z}) \parallel p_\theta(\mathbf{z}|\mathbf{x})\big). \quad (2.11)$$

In the exact EM algorithm, the variational family $\mathcal{F}$ is unconstrained, so the solution to the E-step is given by the exact posterior distribution: $q^\star(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$. As this optimal variational distribution over $\mathbf{z}$ is actually conditioned on $\mathbf{x}$, we will now use the notation $q(\mathbf{z}|\mathbf{x})$ instead of $q(\mathbf{z})$. The difficulty arises when the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, which prevents us from solving the E-step analytically. Variational inference then consists in *constraining the variational family* $\mathcal{F}$ and resorting to optimization methods for solving the E-step (Jordan *et al.*, 1999).

Seminal works on variational inference relied on the so-called mean-field approximation, which constrains the variational family $\mathcal{F}$ to be a set of completely factorized distributions (i.e., multivariate distributions over $\mathbf{z}$ that are written as a product of univariate marginal distributions over the entries of $\mathbf{z}$). All marginal posterior dependencies between different entries of $\mathbf{z}$ are ignored here, while the "structured" mean-field approximation (Saul and Jordan, 1996) partially restores some of them. Solving the E-step under the mean-field approximation leads to a set of closed-form coupled solutions for each univariate distribution in the factorization. This approach is also referred to as coordinate-ascent variational inference in the literature (Bishop, 2006; Blei *et al.*, 2017). However, closed-form updates are usually only available for conjugate-exponential models (Winn and Bishop, 2005) when the distribution of each scalar latent variable, conditionally on its parents, belongs to the exponential family and is conjugate with respect to the distribution of these parent variables. Moreover, this coordinate-ascent approach does not scale well for high-dimensional and large-scale inference problems (Hoffman *et al.*, 2013).

An alternative to the mean-field approximation is then to define the variational family as a set of distributions with a certain parametric form $q_\lambda(\mathbf{z}|\mathbf{x})$, where the parameters λ govern the shape of the distribution. For example, we can define the Gaussian variational family where the

parameters λ correspond to the mean vector and covariance matrix:

$$\mathcal{F} = \{q_\lambda(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}),\ \lambda = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}\}. \tag{2.12}$$

As shown below, the optimal parameters λ that maximize the VLB depend on $\mathbf{x}$ and θ. This approach is called fixed-form or structured variational inference (Honkela *et al.*, 2010; Salimans and Knowles, 2013). The VLB in (2.10) then becomes a function of both the generative model parameters θ and *variational parameters* λ:

$$\mathcal{L}(\theta, \lambda; \mathbf{x}) = \mathbb{E}_{q_\lambda(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\lambda(\mathbf{z}|\mathbf{x})\big]. \tag{2.13}$$

The E-step in (2.11) consequently reduces to a parametric optimization problem:

$$\lambda^\star = \arg\max_\lambda \mathcal{L}(\theta, \lambda; \mathbf{x}) = \arg\min_\lambda D_{\mathrm{KL}}(q_\lambda(\mathbf{z}|\mathbf{x}) \parallel p_\theta(\mathbf{z}|\mathbf{x})). \tag{2.14}$$

As the objective function depends on the observed data vector $\mathbf{x}$ and the generative model parameters θ, so does the solution $\lambda^\star$. In fact, the optimal variational distribution depends on $\mathbf{x}$ through the parameters $\lambda^\star$. The M-step remains unchanged in fixed-form variational inference; that is, it consists in updating the generative model parameters by maximizing $\mathcal{L}(\theta, \lambda; \mathbf{x})$ w.r.t. θ, using the current estimate of the variational parameters. If the expectation in (2.13) and its gradient w.r.t. λ can be computed analytically, the optimization problem of the E-step can be solved using gradient-based optimization methods.

In general, given a dataset of i.i.d. data vectors $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$, one needs to find the parameters $\Lambda = \{\lambda_1, ..., \lambda_N\}$ of the variational distributions $q_{\lambda_n}(\mathbf{z}_n|\mathbf{x}_n)$, $n = 1, ..., N$. Taking the same example as before, with $q_{\lambda_n}(\mathbf{z}_n|\mathbf{x}_n) = \mathcal{N}(\mathbf{z}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, we have here $\lambda_n = \{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}$. This problem is solved by maximizing the following total VLB, which is the sum (or equivalently, the mean) of the local VLB defined in (2.13) over each vector in the training dataset:

$$\mathcal{L}(\theta, \Lambda; \mathbf{X}) = \sum_{n=1}^{N} \mathcal{L}(\theta, \lambda_n; \mathbf{x}_n). \tag{2.15}$$

To scale to large amounts of data, *stochastic variational inference* (Hoffman *et al.*, 2013) relies on gradient-based stochastic optimization

(Robbins and Monro, 1951; Bottou, 2004) for maximizing the total VLB in (2.15) w.r.t. the generative model parameters θ. The gradient of the total VLB, $\mathcal{L}(\theta, \Lambda; \mathbf{X})$, is the sum of the gradients of the local VLBs, $\mathcal{L}(\theta, \lambda_n; \mathbf{x}_n)$, defined for each sample $\mathbf{x}_n$ in the dataset. For large datasets, computing this sum to perform a single update of θ with a step of gradient ascent can be inefficient. Therefore, stochastic variational inference exploits a noisy stochastic estimate of the gradient, computed from a single example $\mathbf{x}_n$ or from a mini-batch of examples in the dataset. This is the same principle as that used in stochastic and mini-batch gradient descent (Bottou, 2004), such that stochastic variational inference inherits from the same convergence properties (Robbins and Monro, 1951).

However, the estimation of the complete set of variational parameters can remain expensive for large datasets. Thus, *amortized variational inference* makes a stronger assumption for defining the variational family, by introducing an *inference model* f_ϕ such that

$$\lambda_n = f_\phi(\mathbf{x}_n), \tag{2.16}$$

where ϕ is a set of parameters that is shared among all variational distributions $q_{\lambda_n}(\mathbf{z}_n|\mathbf{x}_n)$. This inference model is used to map the observation $\mathbf{x}_n$ to the local variational parameter λ_n. The variational family $\mathcal{F}$ then corresponds to the set of variational distributions parametrized by ϕ, which are denoted by $q_\phi(\mathbf{z}_n|\mathbf{x}_n)$. For instance, for $q_\phi(\mathbf{z}_n|\mathbf{x}_n) = \mathcal{N}(\mathbf{z}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, we have $\lambda_n = [\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n] = f_\phi(\mathbf{x}_n)$. This amortization principle corresponds to a stronger assumption for the variational family compared to nonamortized fixed-form and mean-field approximations. Therefore, the KL divergence between the exact posterior and its approximation is likely to be larger in the amortized case than in the previous cases. The total VLB for the complete training dataset then becomes a function of ϕ:

$$\mathcal{L}(\theta, \phi; \mathbf{X}) = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}\big[\log p_\theta(\mathbf{x}_n, \mathbf{z}_n) - \log q_\phi(\mathbf{z}_n|\mathbf{x}_n)\big]. \tag{2.17}$$

This means that the optimization of the set of local variational parameters $\Lambda = \{\lambda_1, ..., \lambda_N\}$ is replaced by the optimization of the shared

set of inference model parameters ϕ. Hereinafter, we will use the term inference model to directly denote the variational distribution $q_\phi(\mathbf{z}_n|\mathbf{x}_n)$.

2.4 VAE inference model

VAEs belong to the family of amortized variational inference techniques, where the VLB in (2.17) is optimized using stochastic gradient-based optimization techniques. The VAE generative model $p_\theta(\mathbf{x}, \mathbf{z})$ has already been defined in (2.1)–(2.3). It involves a decoder neural network through $p_\theta(\mathbf{x}|\mathbf{z})$. To fully specify the VLB, which is required to learn the generative model parameters θ, it is also necessary to define the inference model $q_\phi(\mathbf{z}|\mathbf{x})$, which approximates the intractable exact posterior $p_\theta(\mathbf{z}|\mathbf{x})$.

Similar to the generative model, the inference model for $q_\phi(\mathbf{z}|\mathbf{x})$ is defined by an *encoder neural network*. A common choice for the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is to use a Gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{x})\}) \tag{2.18}$$

$$= \prod_{l=1}^{L} q_\phi(z_l|\mathbf{x}) = \prod_{l=1}^{L} \mathcal{N}(z_l; \mu_{\phi,l}(\mathbf{x}), \sigma^2_{\phi,l}(\mathbf{x})), \tag{2.19}$$

where index $l \in \{1, ..., L\}$ is used to denote the l-th entry of the corresponding vectors, and $\boldsymbol{\mu}_\phi : \mathbb{R}^F \mapsto \mathbb{R}^L$ and $\boldsymbol{\sigma}_\phi : \mathbb{R}^F \mapsto \mathbb{R}^L_+$ are nonlinear functions of $\mathbf{x}$, modeled by a DNN called the encoder or recognition network, which is parametrized by a set of weights and biases denoted by ϕ. The encoder network is illustrated in Figure 2.2 (left). As for the VAE generative model, for the sake of consistency with the presentation of the other models, we denote

$$[\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi(\mathbf{x})] = e_\mathbf{z}(\mathbf{x}), \tag{2.20}$$

where $e_\mathbf{z}$ is the nonlinear function implemented by the encoder DNN.

2.5 VAE training

In the VAE methodology (Kingma and Welling, 2014; Rezende *et al.*, 2014), the VLB in (2.17) is optimized using stochastic gradient-based

optimization techniques to learn the generative and inference model parameters. For training the VAE, the encoder and decoder networks are cascaded, as illustrated in Figure 2.2, and the sets of parameter θ and ϕ are jointly estimated from the training data $\mathbf{X}$. This is different from an EM algorithm strategy, which would alternatively optimize the VLB w.r.t. ϕ and θ in the E- and M-steps, respectively. He *et al.* (2018) showed that this joint encoder-decoder training of the VAE can, however, be suboptimal.

The VLB in (2.17) can be reshaped as (Kingma and Welling, 2014)

$$\mathcal{L}(\theta, \phi; \mathbf{X}) = \underbrace{\sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}\big[\log p_{\theta_\mathbf{x}}(\mathbf{x}_n|\mathbf{z}_n)\big]}_{\text{Reconstruction accuracy}} \underbrace{- \sum_{n=1}^{N} D_{KL}\big(q_\phi(\mathbf{z}_n|\mathbf{x}_n) \parallel p_{\theta_\mathbf{z}}(\mathbf{z}_n)\big)}_{\text{Regularization}} . \tag{2.21}$$

The first term on the right-hand side of (2.21) is a reconstruction term that represents the average accuracy of the chained encoding-decoding process. For instance, if the generative model $p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z})$ is chosen to be Gaussian with an identity covariance matrix, the reconstruction term is equal to the opposite of the mean-squared error (MSE) between the original data and decoder output, up to additive constants. The second term is a regularization one, which enforces the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to be close to the prior distribution $p_{\theta_\mathbf{z}}(\mathbf{z})$. Provided that an independent Gaussian prior is used, this term forces $\mathbf{z}$ to be a disentangled data representation; that is, the $\mathbf{z}$ entries tend to be independent and encode a different characteristic (or factor of variation) of the data.

For usual distributions, the regularization term has an analytical expression as a function of θ and ϕ. However, the expectation taken with respect to $q_\phi(\mathbf{z}_n|\mathbf{x}_n)$ in the reconstruction accuracy term is analytically intractable. Therefore, in practice, it is approximated using a Monte Carlo estimate with R samples $\mathbf{z}_n^{(r)}$ independently and identically drawn

from $q_\phi(\mathbf{z}_n|\mathbf{x}_n)$ (for each index n):

$$\mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] \approx \frac{1}{R}\sum_{r=1}^{R} \log p_\theta(\mathbf{x}_n|\mathbf{z}_n^{(r)}). \tag{2.22}$$

The resulting Monte Carlo estimate of the VLB is given by

$$\hat{\mathcal{L}}(\theta, \phi; \mathbf{X}) = \sum_{n=1}^{N} \frac{1}{R}\sum_{r=1}^{R} \log p_\theta(\mathbf{x}_n|\mathbf{z}_n^{(r)}) \; - \sum_{n=1}^{N} D_{KL}(q_\phi(\mathbf{z}_n|\mathbf{x}_n) \parallel p(\mathbf{z}_n)). \tag{2.23}$$

To optimize this objective function, we can typically resort to the (variants of) stochastic or mini-batch gradient descent (on the negative VLB) (Bottou, 2004). While the gradient of $\hat{\mathcal{L}}(\theta, \phi; \mathbf{X})$ w.r.t. θ can be easily computed using the standard backpropagation algorithm, that w.r.t. ϕ is problematic because the sampling operation from $q_\phi(\mathbf{z}_n|\mathbf{x}_n)$ is not differentiable w.r.t. ϕ. The solution to this problem, proposed by Kingma and Welling (2014) and referred to as the *reparameterization trick*, consists in reparametrizing the sample $\mathbf{z}_n^{(r)}$ using a differentiable transformation of a sample $\boldsymbol{\epsilon}^{(r)}$ drawn from a standard Gaussian distribution, which does not depend on ϕ:

$$\mathbf{z}_n^{(r)} = \boldsymbol{\mu}_\phi(\mathbf{x}_n) + \text{diag}\{\boldsymbol{\sigma}_\phi^2(\mathbf{x}_n)\}^{\frac{1}{2}}\boldsymbol{\epsilon}^{(r)}, \qquad \boldsymbol{\epsilon}^{(r)} \sim \mathcal{N}(\mathbf{0}_L, \mathbf{I}_L). \tag{2.24}$$

Using this reparameterization trick, $\hat{\mathcal{L}}(\theta, \phi; \mathbf{X})$ is now differentiable w.r.t. ϕ. This differentiable Monte Carlo approximation of the VLB is referred to as the stochastic gradient variational Bayes (SGVB) estimator (Kingma and Welling, 2014). The gradient of $\hat{\mathcal{L}}(\theta, \phi; \mathbf{X})$ w.r.t. ϕ is an unbiased estimate of the gradient of the exact VLB $\mathcal{L}(\theta, \phi; \mathbf{X})$ (Kingma and Welling, 2019). This property allows using very few samples to compute the SGVB estimator, which however impacts the variance of the estimator. Kingma and Welling (2014) suggested setting $R = 1$ provided that sufficiently large mini-batches are used for the gradient descent. This training procedure of a VAE model is now considered routine within deep learning toolkits, such as TensorFlow (Abadi *et al.*, 2016) and PyTorch (Paszke *et al.*, 2019).

3

Recurrent Neural Networks and State Space Models

As mentioned earlier, DVAEs are formed of combinations of a VAE and temporal models. Most of these temporal models rely on RNNs and/or SSMs. We thus briefly present the basics of RNNs and SSMs in this chapter before moving on to DVAEs in the next chapters. An extended technical overview of RNNs and SSMs, as well as their applications, is beyond of the scope of the present paper.

3.1 Recurrent Neural Networks

3.1.1 Principle and definition

RNNs have been and are still widely used for data sequence modeling and generation and sequence-to-sequence mapping. An RNN is a neural network that processes ordered vector sequences and uses a memory of past input/output data to condition the current output (Sutskever, 2013; Graves *et al.*, 2013). This is achieved using an additional vector that recursively encodes the internal state of the network.

We denote by $\mathbf{x}_{t_1:t_2} = \{\mathbf{x}_t\}_{t=t_1}^{t_2}$ a sequence of vectors $\mathbf{x}_t$ indexed from t_1 to t_2, where $t_1 \leq t_2$. When $t_1 > t_2$, we assume $\mathbf{x}_{t_1:t_2} = \emptyset$. We present RNNs in the general framework of nonlinear systems, which

transform an input vector sequence $\mathbf{u}_{1:T}$ into an output vector sequence $\mathbf{x}_{1:T}$, possibly through an internal state vector sequence $\mathbf{h}_{1:T}$. The input, output, and internal state vectors can have arbitrary (different) dimensions. If $\mathbf{u}_{1:T}$ is an "external" input sequence, the network can be considered as a "system," as is usual in control theory ($\mathbf{u}_{1:T}$ being considered as a command to the system). If $\mathbf{u}_t = \emptyset$, the RNN is in the *undriven* mode. In contrast, if $\mathbf{u}_t = \mathbf{x}_{t-1}$, the RNN is in the *predictive* mode, or *sequence generation* mode, which is a usual mode when we are interested in modeling the evolution of a data sequence $\mathbf{x}_{1:T}$ "alone" (i.e., independently of any external input; in this case, $\mathbf{x}_{1:T}$ can be seen both as an input and an output sequence).

A basic single-layer RNN model is defined by

$$\mathbf{h}_t = d_{\text{hid}}(\mathbf{W}_{\text{in}}\mathbf{u}_t + \mathbf{W}_{\text{rec}}\mathbf{h}_{t-1} + \mathbf{b}_{\text{hid}}), \tag{3.1}$$

$$\mathbf{x}_t = d_{\text{out}}(\mathbf{W}_{\text{out}}\mathbf{h}_t + \mathbf{b}_{\text{out}}), \tag{3.2}$$

where $\mathbf{W}_{\text{in}}$, $\mathbf{W}_{\text{rec}}$ and $\mathbf{W}_{\text{out}}$ are weight matrices of appropriate dimensions; $\mathbf{b}_{\text{hid}}$ and $\mathbf{b}_{\text{out}}$ are bias vectors; and d_{hid} and d_{out} are nonlinear activation functions. We also define the initial internal state vector $\mathbf{h}_0$. This model is extendable to more complex recurrent architectures:

$$\mathbf{h}_t = d_{\mathbf{h}}(\mathbf{u}_t, \mathbf{h}_{t-1}), \tag{3.3}$$

$$\mathbf{x}_t = d_{\mathbf{x}}(\mathbf{h}_t), \tag{3.4}$$

where $d_{\mathbf{h}}$ and $d_{\mathbf{x}}$ denote any arbitrary complex nonlinear functions implemented with a DNN. We assume that this representation includes long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) networks (Cho *et al.*, 2014), which comprise additional internal variables called gates. For simplicity of presentation, these additional internal gates are not formalized in (3.3) and (3.4). The same is true for multi-layer RNNs, where several recursive layers are stacked on top of each other (Graves *et al.*, 2013) (in this case, for the same reason, we do not report layer indexes in (3.3) and (3.4)). This is also true for combinations of multi-layer RNNs and LSTMs (i.e., multi-layer LSTM networks). In summary, we assume that (3.3) and (3.4) are a "generic" or "high-level" representation of an RNN of arbitrary complexity.

Notation remark: To clarify the presentation and links between the different models, we use the same generic notation $d_{\mathbf{x}}$ for the generating function in (2.5) and (3.4), and will do that throughout the paper (and the same for $d_{\mathbf{h}}$ and for $d_{\mathbf{z}}$ later in the paper).

So far, the above RNNs are deterministic: given $\mathbf{u}_{1:T}$ and $\mathbf{h}_0$, $\mathbf{x}_{1:T}$ is completely determined. Such networks are trained by optimizing a deterministic criterion, e.g. the MSE between the target output sequences from a training dataset and the corresponding actual output sequences obtained by the network. The training set of i.i.d. vectors used for VAE training is replaced with that of vector sequences, and consecutive vectors within a training sequence are generally correlated, which is the point of using a dynamical model.

3.1.2 Generative recurrent neural networks

Deterministic RNNs can easily be transformed into *generative* RNNs (GRNNs) by adding stochasticity at the output level. We just have to define a probabilistic observation model and replace the output data sequence with an output sequence of distribution parameters, similar to the VAE decoder:

$$\mathbf{h}_t = d_{\mathbf{h}}(\mathbf{u}_t, \mathbf{h}_{t-1}), \tag{3.5}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{h}_t)] = d_{\mathbf{x}}(\mathbf{h}_t), \tag{3.6}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{h}_t)\}). \tag{3.7}$$

Eq. (3.5) is the same recursive internal state model as (3.3). Eqs. (3.6) and (3.7) constitute the observation model. In (3.7), we use the Gaussian distribution for its generality and for the convenience of illustration, although any distribution can be used, just as for the VAE decoder. Again, one may choose a distribution that is more appropriate for the nature of the data. For example, Graves (2013) proposed using mixture distributions. The complete set of model parameters θ here includes $\theta_{\mathbf{h}}$ and $\theta_{\mathbf{x}}$, the parameters of the networks implementing $d_{\mathbf{h}}$ and $d_{\mathbf{x}}$, respectively. Because the output of $d_{\mathbf{x}}$ in (3.6) is now two vectors of pdf parameters instead of a data vector in (3.4), its size is twice that of the deterministic RNN. When the internal state vector $\mathbf{h}_t$ is of (much)

lower dimension than the output vector $\mathbf{x}_t$, the GRNN observation model becomes similar to the VAE decoder, except that, again, $\mathbf{h}_t$ has a deterministic evolution through time, whereas the latent state $\mathbf{z}$ of the VAE is stochastic and i.i.d., which is a fundamental difference.

Even if the generation of $\mathbf{x}_t$ is now stochastic, the evolution of the internal state is still deterministic. Let us denote $\mathbf{h}_t$ as a function $\mathbf{h}_t = \mathbf{h}_t(\mathbf{u}_{1:t})$ to make the deterministic relation between $\mathbf{u}_{1:t}$ and $\mathbf{h}_t$ explicit (for each time index t).[1] We thus have $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t) = p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t(\mathbf{u}_{1:t}))$. In the predictive mode, we have $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t) = p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{0:t-1}))$.[2] Such stochastic version of the RNN can be trained with a statistical criterion (e.g., maximum likelihood). As for the VAE training, we search for the maximization of the observed data log-likelihood w.r.t. θ over a set of training sequences. For one sequence, with the conditional independence of successive data vectors, the data log-likelihood is given by

$$\log p_{\theta_\mathbf{x}}(\mathbf{x}_{1:T}|\mathbf{u}_{1:T}) = \sum_{t=1}^{T} \log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t(\mathbf{u}_{1:t})). \tag{3.8}$$

3.2 State Space Models

3.2.1 Principle and definition

SSMs are a rich family of models that are widely used to model dynamical systems (e.g., in statistical signal processing, time-series analysis, and control theory) (Durbin and Koopman, 2012). Here, we focus on discrete-time, continuous-valued SSMs of the form

$$[\boldsymbol{\mu}_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t), \boldsymbol{\sigma}_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t)] = d_\mathbf{z}(\mathbf{z}_{t-1}, \mathbf{u}_t), \tag{3.9}$$

$$p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t)\}), \tag{3.10}$$

$$[\boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{z}_t), \boldsymbol{\sigma}_{\theta_\mathbf{x}}(\mathbf{z}_t)] = d_\mathbf{x}(\mathbf{z}_t), \tag{3.11}$$

$$p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{z}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{x}}(\mathbf{z}_t)\}), \tag{3.12}$$

[1] $\mathbf{h}_t(\mathbf{u}_{1:t})$ also depends on the initial internal state vector $\mathbf{h}_0$, but we omit this term as an argument of the function for conciseness.

[2] Here, the first "input" $\mathbf{x}_0$ has to be set arbitrarily, just like $\mathbf{h}_0$. Alternately, one can directly start the generation process from an arbitrary internal state vector $\mathbf{h}_1$.

where $d_{\mathbf{z}}$ and $d_{\mathbf{x}}$ are functions of arbitrary complexity, each being parameterized by a set of parameters denoted $\theta_{\mathbf{z}}$ and $\theta_{\mathbf{x}}$, respectively. As for the complete generative model, we have $\theta = \theta_{\mathbf{x}} \cup \theta_{\mathbf{z}}$, and we retain this notation hereinafter. At this point, $d_{\mathbf{z}}$ and $d_{\mathbf{x}}$ can be linear or nonlinear functions, and we will differentiate the two cases later. The observation model (3.11)–(3.12) is very similar to the GRNN observation model (3.6)–(3.7). However, $\mathbf{z}_t$ is here a stochastic internal state vector in contrast to the deterministic internal state $\mathbf{h}_t$ of the (G)RNN. The distribution of $\mathbf{z}_t$, known as the *state model* or the *dynamical model*, is given by (3.9)–(3.10). It follows a first-order Markov model; that is, a temporal dependency is introduced where $\mathbf{z}_t$ depends on the previous state $\mathbf{z}_{t-1}$ and the corresponding input $\mathbf{u}_t$ through the function $d_{\mathbf{z}}$. In short, the above SSM can be considered a GRNN in which the deterministic internal state $\mathbf{h}_t$ is replaced with a stochastic internal state $\mathbf{z}_t$, as illustrated in Figure 3.1.

Notation remark: In the control theory literature, the input corresponding to the generation of $\mathbf{z}_t$ is often denoted as $\mathbf{u}_{t-1}$, or equivalently, $\mathbf{u}_t$ is used to generate the next state $\mathbf{z}_{t+1}$. This notation is arbitrary. In the present paper, we prefer to realign the temporal indices, so that the input $\mathbf{u}_t$ is used to generate $\mathbf{z}_t$, which in turn is used to generate $\mathbf{x}_t$, to maintain better consistency through all presented models.

As for the complete sequence, given the dependencies represented in Figure 3.1, the joint distribution of all variables can be expressed as

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_t) p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t) p(\mathbf{u}_t), \tag{3.13}$$

from which we can deduce

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_t), \tag{3.14}$$

and

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t). \tag{3.15}$$

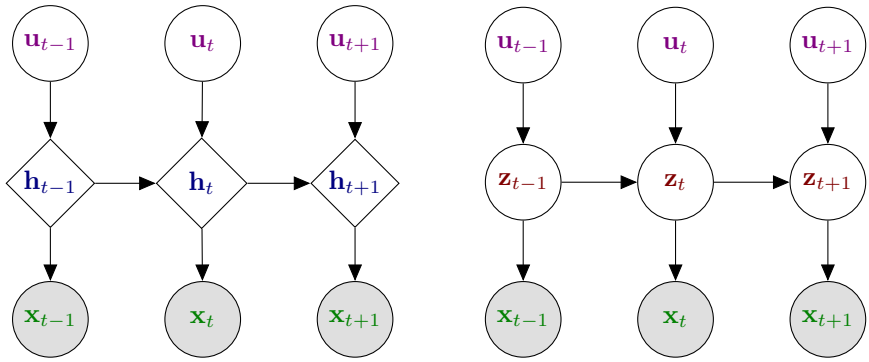


Figure 3.1: GRNN (left) against SSM (right): The two models have an identical structure, though the internal state of the GRNN is deterministic (represented with a diamond), whereas that of the SSM is stochastic (represented with a circle).

Given the state sequence $\mathbf{z}_{1:T}$, the observation vectors at different time frames are mutually independent. The prior distribution of $\mathbf{u}_t$ also factorizes across time frames, but this is of limited interest here. To be complete, we should specify the model "initialization": At $t = 1$, we need to define $\mathbf{z}_0$, which can be set to an arbitrary deterministic value, or defined through a prior distribution $p_{\theta_\mathbf{z}}(\mathbf{z}_0)$ (which then must be added to the right-hand side of (3.13) and (3.15)), or we can set $\mathbf{z}_0 = \emptyset$, in which case the first term of the state model in these equations is $p_{\theta_\mathbf{z}}(\mathbf{z}_1|\mathbf{u}_1)$.

Solving the above SSM means that we run the inference process; that is, we estimate the state vector sequence $\mathbf{z}_{1:T}$ from an observed data vector sequence $\mathbf{x}_{1:T}$. The use of Gaussian distribution in (3.10) and (3.12) is a convenient choice that generally facilitates inference. More generally, these distributions are within the exponential family, so either exact or approximate inference algorithms can be applied, depending on the nature of $d_\mathbf{z}$ and $d_\mathbf{x}$. In the next subsection, we provide an example of a closed-form inference solution when $d_\mathbf{z}$ and $d_\mathbf{x}$ are linear functions.

3.2.2 Kalman filters

Some classical SSMs have been successfully used for decades for a wide set of applications. For example, when $d_\mathbf{x}$ and $d_\mathbf{z}$ are linear functions

of the form

$$\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}, \mathbf{u}_t) = \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{m}_t, \quad \boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}, \mathbf{u}_t) = \boldsymbol{\Lambda}_t, \tag{3.16}$$

$$\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{z}_t) = \mathbf{C}_t \mathbf{z}_t + \mathbf{n}_t, \quad \boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{z}_t) = \boldsymbol{\Sigma}_t, \tag{3.17}$$

where $\mathbf{A}_t$, $\mathbf{B}_t$, $\mathbf{m}_t$, $\boldsymbol{\Lambda}_t$, $\mathbf{C}_t$, $\mathbf{n}_t$, and $\boldsymbol{\Sigma}_t$ are matrices and vectors of appropriate size, the SMM transforms into a linear-Gaussian linear dynamical system (LG-LDS). In this case, the inference has a very popular closed-form solution, known as a *Kalman filter* (Moreno and Pigazo, 2009). More precisely, a Kalman filter is the solution obtained when the past and present observations (outputs and inputs) are used at each time t (i.e., causal inference). When a complete sequence of observations is used at each time t (i.e., noncausal inference), the solution is referred to as a *Kalman smoother*, also obtainable in closed form. In practical problems, $\mathbf{x}_{1:T}$ is generally noisy, and the terms "filter" and "smoother" refer to the estimation of a "clean" state vector trajectory $\mathbf{z}_{1:T}$ from noisy observed data.

The Kalman filter is an iterative solution that alternates between a prediction step and an update step. The prediction step involves computing the *predictive distribution*, which is the posterior distribution of $\mathbf{z}_t$ given the observations up to time $t-1$. Starting from the joint distribution of all variables and exploiting the dependencies in the generative model, the predictive distribution can be expressed as (we omit the input $\mathbf{u}$ for simplicity of presentation)

$$p(\mathbf{z}_t|\mathbf{x}_{1:t-1}) = \int p(\mathbf{z}_t|\mathbf{z}_{t-1}) p(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1}) d\mathbf{z}_{t-1}. \tag{3.18}$$

The update step involves integrating the new (current) observation $\mathbf{x}_t$ using Bayes' rule to obtain the so-called *filtering distribution* (up to some normalizing factor that does not depend on $\mathbf{z}_t$):

$$p(\mathbf{z}_t|\mathbf{x}_{1:t}) \propto p(\mathbf{x}_t|\mathbf{z}_t) \int p(\mathbf{z}_t|\mathbf{z}_{t-1}) p(\mathbf{z}_{t-1}|\mathbf{x}_{1:t-1}) d\mathbf{z}_{t-1}. \tag{3.19}$$

The filtering distribution at time t can be computed recursively from the filtering distribution at time $t-1$ (inside the integral). In the case of linear-Gaussian generative distributions, the filtering distribution is Gaussian, with parameters that can be computed recursively from the

parameters at time $t-1$ and the generative model parameters with basic matrix/vector operations. In practice, these parameters are computed in the following two steps: prediction step and update step. Finally, the mean vector of the filtering distribution, which is often used as the state estimate, is a linear form of the observation vector.

In the noncausal case, a similar two-step predictive/update recursive process can be computed, except that the recursion is processed in both forward (causal) and backward (anticausal) directions, leading to the *smoothing distribution*. A more detailed presentation of the Kalman filter and Kalman smoother is beyond the scope of the present paper.

3.2.3 Nonlinear Kalman filters

Nonlinear dynamical systems (NDS), sometimes abusively referred to as nonlinear Kalman filters, have also been extensively studied, well before the deep learning era. Principled extensions to the Kalman Filter have been proposed to deal with the nonlinearities (e.g., the extended Kalman filter and the unscented Kalman filter) (Wan and Van Der Merwe, 2000; Daum, 2005). The review of nonlinear Kalman filters is beyond the scope of the present paper, to retain the focus on DVAEs.

4

Definition of Dynamical VAEs

In this section, we describe a general methodology for defining and training dynamical VAEs. Our goal is to encompass different models proposed in the literature, which we will describe in detail later. These models can be considered particular instances of this general definition, given simplifying assumptions. This section will prepare the readers to understand well the commonalities and differences among all models that we will review and may motivate future developments. We first define a DVAE in terms of a generative model and then present the general lines of inference and training in the DVAE framework.

4.1 Generative model

As already mentioned, DVAEs consider a sequence of observed random vectors $\mathbf{x}_{1:T} = \{\mathbf{x}_t \in \mathbb{R}^F\}_{t=1}^T$ and that of latent random vectors $\mathbf{z}_{1:T} = \{\mathbf{z}_t \in \mathbb{R}^L\}_{t=1}^T$. As opposed to the a "static" VAE and similarly to SSMs, these two data sequences are assumed to be temporally correlated and can have somewhat complex (cross-)dependencies across time. Defining a DVAE generative model involves specifying the joint distribution of the observed and latent sequential data, $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$, the parameters of which are provided by DNNs, which themselves depend on a set of

parameters θ.

When the model works in the so-called *driven mode*, one additionally considers an input sequence of observed random vectors $\mathbf{u}_{1:T} = \{\mathbf{u}_t \in \mathbb{R}^U\}_{t=1}^T$, and in that case, $\mathbf{x}_{1:T}$ is considered the output sequence. In this case, to define the full generative model, we need to specify the joint distribution $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{u}_{1:T})$. However, in practice, we are usually only interested in modeling the generative process of $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$ given the input sequence $\mathbf{u}_{1:T}$. Loosely speaking, the input sequence is assumed deterministic, while $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$ are stochastic. Therefore, as is commonly observed in the DVAE literature (Krishnan *et al.*, 2015; Fraccaro *et al.*, 2016; Fraccaro *et al.*, 2017), we will only focus on modeling the distribution $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T})$.

In the following section, we will first omit θ when defining the general structure of dependencies in the generative model. We will specify the parameter notation later when introducing how RNNs are used to parametrize the model. In addition, we will consider the model in the driven mode (i.e., with $\mathbf{u}_{1:T}$ as input) as it is more general than that in the undriven mode (i.e., with no "external" input). The undriven mode equations can be obtained from the driven mode equations by simply removing $\mathbf{u}_{1:T}$.

4.1.1 Structure of dependencies in the generative model

As we will discuss in detail in Section 14.3, a DVAE can be considered a structured or hierarchical VAE in which both observed and latent variables are a set of ordered vectors, and the ordering is imposed by time. However, the natural order present in the data does not imply a unique possible structure of variable dependencies for a DVAE generative (or inference) model. In fact, in DVAEs, the joint distribution of the observed and latent vector sequences is usually defined using the chain rule; that is, it is written as a product of conditional distributions over the vectors at different time indices. When writing the chain rule, different orderings of the random vectors can be arbitrarily chosen. This is an important point because the choice of ordering when applying the chain rule yields different practical implementations, which result in different sampling processes.

A natural choice for ordering dependencies at generation is to use a *causal* model. In the present context, a generation (or inference) model is said to be causal if the distribution of the generated (or inferred) variable at time t depends only on its values at previous time indices and/or on the values of the other variables at time t and at previous time indices. If the dependency is only over future time indices, the model is said to be *anticausal*, and if the dependency combines the past, present, and future of the conditioning variables, the model is said to be *noncausal*.

Let us consider the following simple example:

$$p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{z}_1, \mathbf{z}_2) = p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2)p(\mathbf{z}_2|\mathbf{x}_1, \mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1)p(\mathbf{z}_1) \tag{4.1}$$

$$= p(\mathbf{x}_2|\mathbf{x}_1, \mathbf{z}_1, \mathbf{z}_2)p(\mathbf{x}_1|\mathbf{z}_1, \mathbf{z}_2)p(\mathbf{z}_2|\mathbf{z}_1)p(\mathbf{z}_1). \tag{4.2}$$

In (4.1), the sampling is causal because we alternate between sampling $\mathbf{z}_t$ and $\mathbf{x}_t$ from their past value or their past and present values, from $t = 1$ to 2. In contrast, in (4.2), the sampling is not causal because we first have to sample the complete sequence of latent vectors $\mathbf{z}_{1:2}$ before sampling $\mathbf{x}_1$, and then $\mathbf{x}_2$. This principle generalizes to much longer sequences.

In the DVAE literature, causal modeling is the most popular approach. In what follows, we will therefore focus on causal modeling, but the general methodology is similar for noncausal modeling. To the best of our knowledge, only one noncausal model has been proposed in the literature: the RVAE model (Leglaive *et al.*, 2020). In fact, both causal and noncausal versions of RVAE were proposed in this paper, and both versions will be presented in Section 10.

In (causal) DVAEs, the joint distribution of the latent and observed sequences is first factorized according to the time indices using the chain rule:

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}). \tag{4.3}$$

The only assumption made in (4.3) is the causal dependence of $\mathbf{x}_t$ and $\mathbf{z}_t$ on the input sequence $\mathbf{u}_{1:T}$. Then, at each time index $p(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$

is again factorized using the chain rule, so that

$$p(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}). \tag{4.4}$$

This equation is a generalization of (4.1), and again, it exhibits the alternate sampling of $\mathbf{z}_t$ and $\mathbf{x}_t$. Similarly to our remark in Section 3.2.1, for $t = 1$, the first terms of the products in (4.3) and (4.4) are $p(\mathbf{x}_1, \mathbf{z}_1|\mathbf{u}_1)$ and $p(\mathbf{x}_1|\mathbf{z}_1, \mathbf{u}_1)p(\mathbf{z}_1|\mathbf{u}_1)$, respectively. This is consistent with our notation choice of $\mathbf{x}_{1:0} = \mathbf{z}_{1:0} = \emptyset$. Alternatively, we can define $\mathbf{z}_0$ as the initial state vector and consider $p(\mathbf{z}_0)$, $p(\mathbf{z}_1|\mathbf{z}_0, \mathbf{u}_1)$, and so on, in these equations. Hereinafter, for each detailed model, we will present the joint distribution in the general form of a product over frames from $t = 1$ to T, and for conciseness, will not detail the model "initialization."

As will be detailed later, the different models proposed in the literature make different conditional independence assumptions to simplify the dependencies in the conditional distributions of (4.4). For instance, the SSM family presented in Section 3.2 is based on the following conditional independence assumptions:

$$p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t|\mathbf{z}_t), \tag{4.5}$$

$$p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t). \tag{4.6}$$

We have already introduced the concept of the driven mode. In the causal context, we say that a DVAE is in the driven mode if $\mathbf{u}_{1:t}$ is used to generate either $\mathbf{x}_{1:t}$, $\mathbf{z}_{1:t}$, or both. A DVAE is in *predictive* mode if $\mathbf{x}_{1:t-1}$, or part of this sequence, typically $\mathbf{x}_{t-1}$, is used to generate either $\mathbf{x}_t$ or $\mathbf{z}_t$, or both. This corresponds to feedback or closed-loop control in control theory. This is also strongly related to the concept of *autoregressive process*, jointly found in the control theory, machine learning, signal processing, or time-series analysis literature (Papoulis, 1977; Frey, 1998; Durbin and Koopman, 2012; Hamilton, 2020). Therefore, in what follows, we indifferently use the terms *predictive DVAE* or *autoregressive DVAE* to qualify a DVAE in the predictive mode.

In its most general form (4.4), a DVAE is both in the driven and predictive modes; however, it can also be in only one of the two modes

(e.g., the above SSM is in the driven mode but not in the predictive mode), or even in none of them. In the literature, we did not encounter any DVAE in both modes at the same time. Moreover, there are models in the driven and nonpredictive modes that are converted to the undriven and predictive modes by replacing the control input $\mathbf{u}_t$ with the previously generated output $\mathbf{x}_{t-1}$, see (Fraccaro *et al.*, 2016). Note that a model's behavior can be quite different under the various modes. This is consistent with the concept of using a model in an open loop or in a closed loop in control theory. The principle of these different modes has been poorly discussed in the DVAE literature, and it is interesting to clarify it at an early stage of the DVAE presentation.

4.1.2 Parameterization with (R)NNs

The factorization in (4.4) is a general umbrella for all (causal) DVAEs. As discussed above, each DVAE model will make different conditional independence assumptions, which will simplify the general factorization in various ways. Once the conditional assumptions are made, one can easily determine if there is a need to accumulate the past information (e.g., $\mathbf{z}_t$ or $\mathbf{x}_t$ depends on past observations $\mathbf{x}_{1:t}$) or if a first-order Markovian relationship holds (e.g., $\mathbf{z}_t$ and $\mathbf{x}_t$ depend at most on $\mathbf{z}_{t-1}$ and $\mathbf{x}_{t-1}$). Usually, the former is implemented using RNNs, whereas feed-forward DNNs can be used to implement first-order Markovian dependency. Moreover, once the conditional assumptions are made, the remaining dependencies can be implemented in different ways. Therefore, the final family of distributions depends not only on the conditional independence assumptions but also on the networks that are used to implement the remaining dependencies.

Let us showcase this with a concrete example in which we have the following conditional independence assumptions:

$$p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_t), \tag{4.7}$$

$$p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_t). \tag{4.8}$$

Here, we assume that the generation of both $\mathbf{x}_t$ and $\mathbf{z}_t$ depends on $\mathbf{x}_{1:t-1}$. In addition, the generation of $\mathbf{x}_t$ also depends on $\mathbf{z}_t$ and that of $\mathbf{z}_t$ also depends on $\mathbf{u}_t$. To accumulate the information of all past outputs

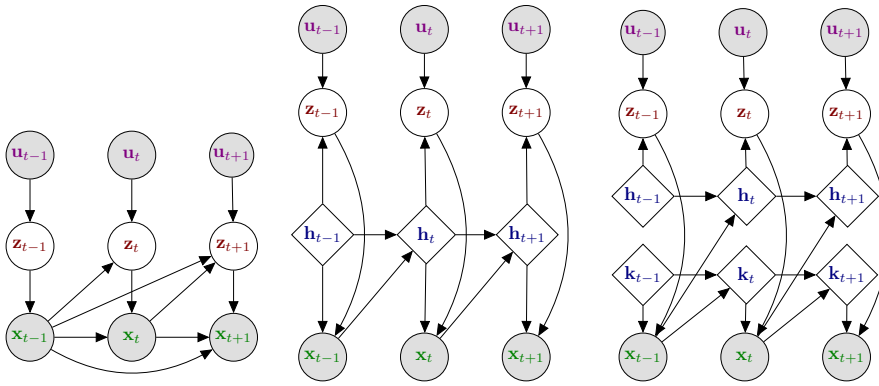


Figure 4.1: Two different implementations of a given factorization. The probabilistic graphical model (left) shows the dependencies between random variables and corresponds to the factorization in (4.7) and (4.8). Two possible implementations based on RNNs are shown: sharing the internal state variables (middle) or with two different internal state variables (right). We refer to the *compact* representation (left) and to the *developed* representations (middle and right). This terminology holds true for both the graphical representations and model formulations.

$\mathbf{x}_{1:t-1}$, one can use an RNN. In practice, the past information is accumulated in the internal state variable of the RNN, namely $\mathbf{h}_t$, computed recurrently at each frame t. Among the many possible implementations, we consider two in this example: in the first implementation, illustrated in Figure 4.1 (middle), a single RNN internal state variable $\mathbf{h}_t$ is used to generate both $\mathbf{x}_t$ and $\mathbf{z}_t$, while in the second implementation, illustrated in Figure 4.1 (right), two different internal state variables, $\mathbf{h}_t$ and $\mathbf{k}_t$, are used to generate $\mathbf{x}_t$ and $\mathbf{z}_t$ separately.

Assuming that all probability distributions are Gaussian, the first implementation can be expressed as

$$\mathbf{h}_t = d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}; \theta_{\mathbf{h}}), \tag{4.9}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t)] = d_{\mathbf{z}}(\mathbf{h}_t, \mathbf{u}_t; \theta_{\mathbf{hz}}), \tag{4.10}$$

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_t) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t)\}\big), \tag{4.11}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t)] = d_{\mathbf{x}}(\mathbf{h}_t, \mathbf{z}_t; \theta_{\mathbf{hx}}), \tag{4.12}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_t) = \mathcal{N}\big(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t)\}\big), \tag{4.13}$$

where $d_{\mathbf{h}}$, $d_{\mathbf{z}}$, and $d_{\mathbf{x}}$ are nonlinear functions implemented with DNNs. It is now clear that the parameters of the conditional distribution of

$\mathbf{z}_t$ are $\theta_{\mathbf{z}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hz}}$, whereas those of the conditional distribution of $\mathbf{x}_t$ are $\theta_{\mathbf{x}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hx}}$. Thus, the two conditional distributions share the recurrent parameters $\theta_{\mathbf{h}}$. Regarding the second implementation, the generative process can be expressed as

$$\mathbf{h}_t = d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}; \theta_{\mathbf{h}}), \tag{4.14}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t)] = d_{\mathbf{z}}(\mathbf{h}_t, \mathbf{u}_t; \theta_{\mathbf{hz}}), \tag{4.15}$$

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{x}_{1:t-1}, \mathbf{u}_t)\}), \tag{4.16}$$

$$\mathbf{k}_t = d_{\mathbf{k}}(\mathbf{x}_{t-1}, \mathbf{k}_{t-1}; \theta_{\mathbf{k}}), \tag{4.17}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t)] = d_{\mathbf{x}}(\mathbf{k}_t, \mathbf{z}_t; \theta_{\mathbf{kx}}), \tag{4.18}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{x}_{1:t-1}, \mathbf{z}_t)\}). \tag{4.19}$$

We have an additional DNN-based nonlinear function $d_{\mathbf{k}}$, and analogously, it is clear that the parameters of the conditional distribution of $\mathbf{z}_t$ are $\theta_{\mathbf{z}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hz}}$, whereas those of the conditional distribution of $\mathbf{x}_t$ are $\theta_{\mathbf{x}} = \theta_{\mathbf{k}} \cup \theta_{\mathbf{kx}}$. In this case, the two conditional distributions do not share any parameter. To ease the notation, hereinafter, we will denote the parameters of $p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ and $p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ as $\theta_{\mathbf{z}}$ and $\theta_{\mathbf{x}}$, respectively, in (4.4), irrespective of whether or not they share some parameters. We will also use θ to denote $\theta_{\mathbf{z}} \cup \theta_{\mathbf{x}}$.

In the equations above, the operators $d_{\mathbf{h}}$, $d_{\mathbf{k}}$, $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$ are nonlinear mappings parametrized by DNNs of arbitrary architecture. How to choose and design these architectures is beyond the scope of this paper, as it largely depends on the target application. To fix ideas, in the present example, $d_{\mathbf{h}}$ and $d_{\mathbf{k}}$ are RNNs, and $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$ are feed-forward DNNs. In this paper, we will not discuss how to select the hyper-parameters of these networks, such as the number of layers, or the number of units per layer.

Note that (4.11) and (4.16) are exactly the same, meaning that the conditional distributions of $\mathbf{z}_t$ are the same for both models. The same remark holds for (4.13) and (4.19), defining the conditional distribution of $\mathbf{x}_t$. However, the computations performed to obtain the parameters $\theta_{\mathbf{z}}$ and $\theta_{\mathbf{x}}$ differ depending on the model. Clearly, we need to make a distinction between the two forms. We propose to call the form of a DVAE model or its graphical representation *compact* when only random variables appear (e.g., (4.11) and (4.13), and Figure 4.1 (left)).

In addition, we propose to call the form of a DVAE or its graphical representation *developed* when both random and deterministic variables appear (e.g., (4.9)–(4.13), (4.14)–(4.19) and Figure 4.1 (middle) and (right)). Each compact form can have different developed forms corresponding to different implementations. The distinction between the compact and developed forms is important as the optimization occurs on the parameters of the developed form, which is only a subgroup of all possible models satisfying the compact form. It is thus important to present the developed form of a model. However, the temporal dependencies of order higher than one are not directly visible in the developed graphical form, as they might be implicitly encoded in the internal state variables. Therefore, when reviewing DVAE models in the following chapters, we will always present both the compact and developed graphical representations.

4.2 Inference model

In the present DVAE context, the posterior distribution of the state sequence $\mathbf{z}_{1:T}$ is $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ in the driven mode or $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ in the undriven mode. As for the standard VAE, this posterior distribution is intractable because of the nonlinearities in the generative model. In fact, having temporal dependencies only makes things even more complicated. Therefore, we also need to define an inference model $q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, which is an approximation of the intractable posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$. As for the standard VAE, this model is required not only for performing inference of the latent sequence $\mathbf{z}_{1:T}$ from the observed sequences $\mathbf{x}_{1:T}$ and $\mathbf{u}_{1:T}$ but also for estimating the parameters of the generative model, as will be seen below. As for the standard VAE again, the inference model also uses DNNs to generate its parameters.

4.2.1 Exploiting D-separation

In a Bayesian network, and in a DVAE in particular, even though the computation of the posterior distribution is often intractable, there exists a general methodology to express its general form (i.e., to specify the

dependencies between the variables of a generative model *at inference time*). This methodology is based on the so-called *D-separation* property of Bayesian networks (Geiger *et al.*, 1990; Bishop, 2006, Chapter 8). The general principle is that some of the conditioning variables in the expression of the posterior distribution of a given variable can vanish depending on whether the nodes between these conditioning variables and the given variable represent variables that are observed or unobserved and depending on the direction of the dependencies (i.e., the direction of the arrows of the graphical representation).

In detail, D-separation is based on the three principles derived for a Bayesian network with three random variables a, b, and c:

- A *tail-to-tail* (or common parent) node c corresponding to the structure $a \leftarrow c \rightarrow b$ makes the two other nodes a and b *conditionally independent* when it is observed. In short, we have $p(a, b|c) = p(a|c)p(b|c)$.
- A *head-to-tail* (or cascade) node c corresponding to the structure $a \rightarrow c \rightarrow b$ or $a \leftarrow c \leftarrow b$ makes the two other nodes a and b *conditionally independent* when it is observed. In short, we have $p(a, b|c) = p(a|c)p(b|c)$.
- A *head-to-head* (or V-structure) node c corresponding to the structure $a \rightarrow c \leftarrow b$ makes a and b *conditionally dependent* when it is observed, hence $p(a, b|c) \neq p(a|c)p(b|c)$.

D-separation consists in applying these three principles recursively to analyze larger Bayesian networks with any arbitrary structure. Let us consider a Bayesian network in which A, B, and C are arbitrary nonintersecting node sets. *A and B are D-separated given C* if all possible paths that connect any node in A to any node in B are blocked given C. A path is said to be *blocked* given a set of observed nodes O if it includes a node c such that either

- c is a tail-to-tail node and $c \in O$ (i.e., it is observed) *or*
- c is a head-to-tail node and $c \in O$ (i.e., it is observed) *or*
- c is a head-to-head node and $c \notin O$ (i.e., it is *not* observed).

Equivalently, A and B are D-separated given C if they are not connected by any active path (i.e., a path that is not blocked). Finally, if A and B are D-separated given C, we have $p(A, B|C) = p(A|C)p(B|C)$.

D-separation is helpful even for more conventional (i.e., nondeep) models because the algebraic derivation of a posterior distribution from a joint distribution is not always easy. In the present variational framework, we can exploit the above methodology to design the approximate posterior distribution q_ϕ. It is reasonable to assume that a good candidate for q_ϕ will have the same structure as the exact posterior distribution in terms of variable dependency. In other words, if we cannot derive the exact posterior distribution, let us at least use an approximation that exhibits the same dependencies between variables so that it is fed with the same information. Yet, it is quite surprising to see that a significant proportion of the DVAE papers we have reviewed, especially the early papers, neither refer to this methodology nor consider looking at the form of the exact posterior distribution when designing an approximate distribution. In the early studies in particular, the formulation of q_ϕ is chosen quite arbitrarily and with no reference to the structure of the exact posterior distribution. In more recent papers however, the structure of q_ϕ generally follows that of the exact posterior distribution. We will come back on this point on a case-by-case basis when presenting the DVAE models of the literature in the next chapters.

4.2.2 Noncausal and causal inference

Being aware of this problem, we can now go back to the general form of the exact posterior distribution and factorize it as follows, applying again the chain rule the same way as we did for the generative model:

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T}). \tag{4.20}$$

For the most general generative model defined in (4.4), the dependencies in each conditional distribution $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ cannot be simplified. In other words, $\mathbf{z}_t$ depends on the past latent vectors $\mathbf{z}_{1:t-1}$ and on the complete sequences of observed vectors $\mathbf{x}_{1:T}$ and $\mathbf{u}_{1:T}$ (past,

current, and future time steps). The exact inference is thus a noncausal process, even if the generation is causal. This is reminiscent of the Kalman smoother (i.e., the noncausal solution to inference in LG-LDS, see Section 3.2.2). As discussed in the previous subsection, the inference model q_ϕ should here have the same most general structure as the exact posterior distribution of (4.20):

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T}). \tag{4.21}$$

Similar to the generative model, each conditional posterior distribution $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ should accumulate information from past latent variables and past observations, but in contrast to the generative model, it should also accumulate information from present and future observations. Typically, this process is implemented with a bidirectional recurrent network.

Depending on the conditional independence assumptions made when defining the generative model, the posterior dependencies in $p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ can be simplified using the D-separation property of Bayesian networks described in the previous subsection. Thus, the posterior dependencies in $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ can be simplified similarly. Of course, it is always possible to use an approximate posterior q_ϕ that does not follow the structure of the exact posterior distribution. In fact, it makes sense to use a simplified version if one wants to decrease the computational cost or satisfy other constraints. In particular, for online or incremental data processing, the inference can be forced to be a causal process by removing the dependencies of q_ϕ on the future observations (and future inputs). This is similar to the Kalman filter for an LG-LDS, see again Section 3.2.2. This will generally be at risk of degrading the inference performance. Again, we will return to these points when reviewing the DVAE models proposed in the literature.

4.2.3 Sharing variables and parameters at generation and inference

We can note a similarity between the (most general causal) generative distribution $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ and the corresponding inference model $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ in terms of random variable dependencies.

For instance, the general form of the dependency of $\mathbf{z}_t$ on past latent vectors is the same at inference and generation: in both cases, $\mathbf{z}_t$ depends on the complete past sequence $\mathbf{z}_{1:t-1}$. Implementing this recurrence at inference and at generation can be made either with a single unique RNN or with two different RNNs, in line with what we discussed in Section 4.1.2. The same principle applies to $\mathbf{u}_{1:t}$ and $\mathbf{x}_{1:t}$, which are both used at generation and inference. Depending on which variables we consider, it can make sense to use the same RNN at generation and inference, meaning that the deterministic link between the realizations of random variables is the same at generation and at inference. If this is the case, the decoder and encoder share some network modules and thus θ and ϕ share some parameters. Note that this is not the case in standard VAEs.

Hereinafter, we will use $\mathbf{h}_t$ to denote the internal state of the decoder and $\mathbf{g}_t$ to denote that of the encoder if it is different from the internal state of the decoder. Otherwise, we will use $\mathbf{h}_t$ for the encoder as well.

4.3 VLB and training of DVAEs

As for the standard VAE, training a DVAE is based on the maximization of the VLB. In the case of DVAEs, the VLB initially defined in (2.13) is extended to data sequences as follows:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\big[\log p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}|\mathbf{u}_{1:T})\big] \\ - \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\big[\log q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})\big]. \quad (4.22)\end{aligned}$$

With the factorization in (4.21), the expectation in (4.22) can be expressed as a cascade of expectations taken with respect to conditional distributions over individual latent vectors at different time indices:

$$\begin{aligned}\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}[\psi(\mathbf{z}_{1:T})] = \mathbb{E}_{q_\phi(\mathbf{z}_1|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\mathbb{E}_{q_\phi(\mathbf{z}_2|\mathbf{z}_1,\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\big[\dots \\ \mathbb{E}_{q_\phi(\mathbf{z}_T|\mathbf{z}_{1:T-1},\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\big[\psi(\mathbf{z}_{1:T})\big]\dots\big]\Big], \quad (4.23)\end{aligned}$$

where $\psi(\mathbf{z}_{1:T})$ denotes any function of $\mathbf{z}_{1:T}$. Then, by injecting (4.4) and (4.21) into (4.22), and using the above cascade, we can develop the

VLB as follows:

$$\begin{aligned}\mathcal{L}(\theta,\phi;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) &= \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\big[\log p_\theta(\mathbf{x}_{1:T},\mathbf{z}_{1:T}|\mathbf{u}_{1:T}) \\ &\qquad - \log q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})\big] \\ &= \sum_{t=1}^{T}\mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\big[\log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1},\mathbf{z}_{1:t},\mathbf{u}_{1:t})\big] \\ &\quad - \sum_{t=1}^{T}\mathbb{E}_{q_\phi(\mathbf{z}_{1:t-1}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\big[D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1},\mathbf{x}_{1:T},\mathbf{u}_{1:T})\,\|\right. \\ &\qquad \left. p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1},\mathbf{z}_{1:t-1},\mathbf{u}_{1:t})\right)\big]\,. \end{aligned} \tag{4.24}$$

To the best of our knowledge, this is the first time that the VLB is presented in this most general form, which is valid for the entire class of (causal) DVAE models.

As for the standard VAE, the VLB contains a reconstruction accuracy term and a regularization term. However, in contrast to the standard VAE, where the regularization term has an analytical form for usual distributions, here, both the reconstruction accuracy and regularization term require the computation of Monte Carlo estimates (i.e., empirical averages) using samples drawn from $q_\phi(\mathbf{z}_{1:\tau}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})$, where $\tau \in \{1,...,T\}$ is an arbitrary index. Using the chain rule in (4.21), we sample from the joint distribution $q_\phi(\mathbf{z}_{1:\tau}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})$ by sampling recursively from $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1},\mathbf{x}_{1:T},\mathbf{u}_{1:T})$, going from $t=1$ to $t=\tau$. Sampling each random vector $\mathbf{z}_t$ at a given time instant is straightforward, as $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1},\mathbf{x}_{1:T},\mathbf{u}_{1:T})$ is analytically specified by the chosen inference model (e.g., Gaussian with mean and variance provided by an RNN). We have to use a similar reparameterization trick as for standard VAEs, so the sampling-based VLB estimator remains differentiable with respect to ϕ. The VLB can then be maximized with respect to both ϕ and $\theta = \theta_\mathbf{z} \cup \theta_\mathbf{x}$ using gradient-ascent-based algorithms. We recall that for DVAEs, ϕ and θ can share parameters, which is different from the "static" VAE, but perfectly alright for the optimization. Finally, the VLB is here defined here for a single data sequence, but a common practice is to average the VLB over a mini-batch of training data sequences before updating the model parameters with gradient ascent.

4.4 Additional dichotomy for autoregressive DVAE models

A DVAE can be used to generate new data, for analysis-synthesis (by chaining the encoder and decoder), or for data transformation, by modifying the latent vector sequence in between analysis and synthesis. In the case of DVAE models functioning in the predictive mode (i.e., autoregressive DVAEs, see Section 4.1.1), these tasks can be processed in different manners, leading to an additional dichotomy of functioning modes. We describe these functioning modes in the next subsection before we see the implications for model training in the following subsection. Because these additional different modes concern the recursive part of the models, nonpredictive DVAEs are not concerned here.

4.4.1 Teacher forcing against generation mode

In practice, for autoregressive DVAE models, we have two generation modes, for the generation of both $\mathbf{x}_t$ and $\mathbf{z}_t$. A mode in which we assume that the ground-truth past observed vectors $\mathbf{x}_{1:t-1}$ are used for generating the current vector ($\mathbf{x}_t$ or $\mathbf{z}_t$), and a mode in which the generated past observed vectors are used for generating the current vector. At this point, it is important to distinguish between the notation for the ground-truth value of the observed data vector $\mathbf{x}_t$ and that for its modeled version produced by a DVAE, which we denote by $\hat{\mathbf{x}}_t$. In practice, $\mathbf{x}_{1:T}$ is a given data sequence that we want to model with a DVAE (or that we use for model training, as shown below), and $\hat{\mathbf{x}}_{1:T}$ is the actual output of the DVAE.

This issue of either using the ground-truth past observed data vectors $\mathbf{x}_{1:t-1}$ or reinjecting the previously generated vectors $\hat{\mathbf{x}}_{1:t-1}$ at the input of a generative model is a classical problem of recursive models and, in particular, of RNNs. Yet it is poorly discussed in the DVAE literature. In the RNN literature, the first configuration is sometimes referred to as *teacher-forcing* (Williams and Zipser, 1989), as it is assumed that a teacher (or oracle) can provide the model with ground-truth values. Hereinafter, we will use this terminology. This is a classical configuration at training time, when the whole sequence $\mathbf{x}_{1:T}$ is available and the model is tuned so that $\hat{\mathbf{x}}_{1:T}$ fits $\mathbf{x}_{1:T}$. However, this is unrealistic at generation

time, when the model produces a new sequence $\hat{\mathbf{x}}_{1:T}$. Here, the second configuration must be used. We refer to this second configuration as the *generation mode*. Regarding the generation of $\mathbf{z}_{1:T}$, the teacher-forcing concept is irrelevant as the concept of ground-truth values for latent vectors is questionable in essence. In practice, $\mathbf{z}_{1:T}$ is either the output of the inference model (this is the case during DVAE training or in analysis-synthesis) or any arbitrary latent vector sequence (generated with $p_{\theta_{\mathbf{z}}}$ or predefined). Note also that because of the recursivity of the generative process, data generation with a DVAE (strongly) depends on the initialization of the generative process. We do not detail this aspect in the present review.

If we now focus on the analysis-synthesis task, we have to chain the encoder and decoder. The encoder takes $\mathbf{x}_{1:T}$ as the input and produces a sequence of latent vectors $\mathbf{z}_{1:T}$. Then, the decoder uses $\mathbf{z}_{1:T}$ to generate $\hat{\mathbf{x}}_{1:T}$. Decoding can be performed with either teacher-forcing or generation mode. The former case is expected to produce a sequence $\hat{\mathbf{x}}_{1:T}$ that is closer to $\mathbf{x}_{1:T}$ than in the latter case, as it uses ground-truth values, whereas the generation mode uses approximate values. However, it suffers from the same "unrealistic" aspect as that used for data generation. This configuration can be used to evaluate the prediction power of DVAE models in an ideal (oracle) setting. In contrast, analysis-synthesis with the generation mode is expected to yield lower performance but is the natural configuration from an information-theoretic viewpoint. Here, we test the capability of the model to encode the information of a (generally high-dimensional) data sequence $\mathbf{x}_{1:T}$ into a (generally low-dimensional) latent sequence $\mathbf{z}_{1:T}$. From an application point of view, this corresponds to telecommunication or storage applications, where $\mathbf{x}_{1:T}$ would be encoded into $\mathbf{z}_{1:T}$, $\mathbf{z}_{1:T}$ would be transmitted or stored, and then $\hat{\mathbf{x}}_{1:T}$ would be decoded from $\mathbf{z}_{1:T}$. In short, we apply DVAEs to source coding, and the DVAE turns into a codec, apart from quantization issues. Such coding/decoding scheme can be applied offline by using a noncausal inference model (with an optimal structure following that of the exact posterior distribution) or online with a (suboptimal) causal inference model. If some amount of latency is tolerated, one can also use a noncausal inference model with a suitable lookahead. The interest of DVAE models for source coding is further discussed in

Section 14.4.

4.4.2 Train/test matching

As mentioned in the previous subsection, teacher-forcing is a conventional strategy used for training recursive models. However, in practice, when using autoregressive DVAEs for data generation or compression, the generation mode must be used. This leads to a mismatch between the training and testing conditions, a general problem in machine learning that leads to performance degradation compared to the case in which the same configuration is used for training and testing. Therefore, if the generation mode is used in a practical DVAE use-case, it can be beneficial to use the generation mode during model training as well, so that the training configuration matches the practical use-case configuration. In practice, this implies replacing $\mathbf{x}_{1:t-1}$ with $\hat{\mathbf{x}}_{1:t-1}$ in the conditioning variables in the VLB equations of Section 4.3. In Chapter 13, we illustrate this strategy in our experimental benchmark. We observe in our experiments that using the generation mode at both model training and testing leads to a significant gain in performance compared to the mismatched configuration, though the performance remains slightly lower than that in the case where teacher-forcing is used at both training and testing. More details are given in Section 13.3.3.

4.5 DVAE summary

Dynamical VAEs are constructed with various stochastic relationships amonng the control variables $\mathbf{u}_{1:T}$, latent variables $\mathbf{z}_{1:T}$, and observed variables $\mathbf{x}_{1:T}$. We recall that a random variable $\mathbf{a}$ is called a parent of another random variable $\mathbf{b}$ when the realization of $\mathbf{a}$ is used to compute the parameters of the distribution of $\mathbf{b}$. These parameters can be obtained with a linear or a nonlinear mapping of the realization of $\mathbf{a}$ (and possibly of other random variables). A DVAE model must contain two types of relationships:

- **Decoding link:** $\mathbf{z}_t$ is always a parent of $\mathbf{x}_t$. Graphically, there is always an arrow from $\mathbf{z}_t$ to $\mathbf{x}_t$ in the compact graphical representation. This is a fundamental characteristic inherited from the

standard VAE.

- **Temporal link:** At least one element in $\mathbf{z}_{1:t-1}$ or in $\mathbf{x}_{1:t-1}$ is parent to either $\mathbf{z}_t$ or $\mathbf{x}_t$. One of the simplest forms of a temporal link, namely $\mathbf{z}_{t-1}$ is a parent of $\mathbf{z}_t$, is a fundamental characteristic of first-order SSMs.

In a way, the "minimal DVAE" is the straightforward combination of a first-order SSM and a VAE, which is the DKF model that we will detail in Section 5. Other DVAEs include additional temporal links. Moreover, temporal links such as "$\mathbf{z}_{t-1}$ is a parent of $\mathbf{x}_t$" can be considered additional decoding links, in that $\mathbf{x}_t$ is generated from $\mathbf{z}_t$ *and* $\mathbf{z}_{t-1}$. As for temporal links, in the papers that we detail in this overview, $\mathbf{z}_t$ and/or $\mathbf{x}_t$ depend either on $\mathbf{z}_{t-1}$ and/or $\mathbf{x}_{t-1}$, or on $\mathbf{z}_{1:t-1}$ and/or $\mathbf{x}_{1:t-1}$. In other words, the order of temporal dependencies is either 1 (implemented with a basic feed-forward neural network, such as a Multi-Layer Perceptron (MLP)) or infinity (implemented with an RNN). However, one can, in principle, use N-order temporal dependencies with $1 < N < \infty$, relying, for instance, on convolutional neural networks (CNNs) with finite-length receptive fields. In particular, temporal convolutional networks (TCNs) (Lea *et al.*, 2016), which are based on dilated convolutions, are competitive with RNNs on several sequence modeling tasks, including generative modeling (Aksan and Hilliges, 2019).

Finally, as discussed before, a DVAE can be in the *driven mode*, in the *predictive mode*, in both, or in none of these modes. This is also modeled by parenthood relationships that may or may not exist.

- **Driving link:** A DVAE is said to be in the driven mode if $\mathbf{u}_t$ is a parent of either $\mathbf{z}_t$ or $\mathbf{x}_t$, or both.

- **Predictive link:** A DVAE is said to be in the predictive mode if $\mathbf{x}_{1:t-1}$, or part of this sequence, is a parent of either $\mathbf{z}_t$ or $\mathbf{x}_t$, or both. In practice, a predictive link is implemented either in the teacher-forcing mode (using the ground-truth past vector sequence) or in the generation mode (using the previously generated vector sequence). Generally, better performance is achieved if the same mode is used at training time and test time.

5

Deep Kalman Filters

Continuing from the previous section, we start our DVAE tour by combining SSMs with neural networks. Such a combination is not recent, see, e.g., (Haykin, 2004; Raiko and Tornio, 2009), but has been recently investigated under the VAE angle in two papers by the same authors (Krishnan *et al.*, 2015; Krishnan *et al.*, 2017). The resulting deep SSM is referred to as a DKF (Krishnan *et al.*, 2015) or a deep Markov model (DMM) (Krishnan *et al.*, 2017).[1] Therefore, these papers do not provide a new concept in terms of models, but they provide a solution to the joint problem of inference and model parameter estimation in the VAE methodological framework applied to the SSM model architecture. In other words, this is, to the best of our knowledge, the first example of unsupervised training of a deep SSM by chaining an approximate inference model with a generative model and using the VLB maximization methodology. This training leads to an unsupervised discovery of the latent space that encodes the temporal

[1]The same generative model is considered in both papers, but as we will detail later, the second paper proposes notable improvements regarding the inference model. The authors change the model name from DKF to DMM, maybe because the second denomination appears more general. In the present review, we retain the denomination DKF.

dynamics of the data. The VAE methodology circumvents the difficulties encountered in previous approaches (Haykin, 2004; Raiko and Tornio, 2009) concerning the computational complexity and practicability of model parameter estimation, particularly allowing to move directly from single-layer neural networks to DNNs.

Hereinafter, for all detailed DVAE models, we first present the generative equations, then the inference model (and we discuss its choice by referring to the exact posterior distribution structure as deduced from the generative model), and finally the detailed form of the VLB used for model training together with clues about the optimization algorithm.

5.1 Generative model

We have already seen the generative equations of the DKF, as they are the same as (3.9)–(3.12), with the specificity that $d_{\mathbf{z}}$ and $d_{\mathbf{x}}$ are DNNs here.[2] Krishnan *et al.* (2015) did not specify these DNNs; we can assume that basic feed-forward neural networks (i.e., MLPs) were used. In their second paper, Krishnan *et al.* (2017) implemented $d_{\mathbf{x}}$ with a two-layer MLP and used a slightly more refined model for $d_{\mathbf{z}}$: a gated linear combination of a linear model and an MLP for the mean parameter, where the gate is itself provided by an MLP, and the chaining of MLP, rectified linear unit (ReLU) activation, and Softmax activation layers for the variance parameter (see Section B.1). According to the authors, this allows "the model have the flexibility to choose a linear transition for some dimensions while having a nonlinear transition for the others."

Even if we are still at an early point in our presentation of the different DVAE models, we can make a first series of remarks to clarify the links between the models we have discussed so far.

- The stochastic state $\mathbf{z}_t$ of an SSM is similar in essence to the latent state of the VAE. In the present DKF case, where $d_{\mathbf{x}}$ is implemented

[2] In fact, a Bernouilli distribution was considered for $\mathbf{x}_t$ by Krishnan *et al.* (2015) and Krishnan *et al.* (2017), but we have already mentioned that different pdfs can be considered for $p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_t)$ depending on the data, without affecting the fundamental issues of this review. Hence, we consider here a Gaussian distribution for a better comparison with the other models.

with a DNN, if $\mathbf{z}_t$ is of reduced dimension compared to $\mathbf{x}_t$, *the DKF observation model is identical to the VAE decoder.*

- Consequently, a DKF can be viewed as a VAE decoder with a temporal (Markovian) model of the latent variable $\mathbf{z}$.

- A (deep) SSM can also be viewed as a "fully stochastic" version of a (deep) RNN, where stochasticity is introduced at both the observation model level (like a GRNN) *and* internal state level. As mentioned before, in an SSM, the deterministic internal state $\mathbf{h}_t$ of the (G)RNN is simply replaced with a stochastic state $\mathbf{z}_t$.

- In summary, DKF = deep SSM = "Markovian" VAE decoder = "fully stochastic" RNN. The graphical model of the DKF is given by the right-hand schema in Figure 3.1 (which does not make the DNNs apparent).

5.2 Inference model

Following the general line of Section 4.2, we first identify the structure of the SSM/DKF posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$. Let us first recall that applying the chain rule enables us to rewrite this distribution as follows:

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T}). \tag{5.1}$$

Then, D-separation can be used to simplify each term of the product. The structure presented in Figure 3.1 (right) shows that the $\mathbf{z}_{t-1}$ node "blocks" all information coming from the past and flowing to $\mathbf{z}_t$ (i.e., $\mathbf{z}_{1:t-2}$, $\mathbf{x}_{1:t-1}$, and $\mathbf{u}_{1:t-1}$). In other words, $\mathbf{z}_{t-1}$ has accumulated this past information or is a summary of this information. We thus have $p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T}, \mathbf{u}_{t:T})$, and therefore (with $\mathbf{z}_0$ being arbitrarily set)

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T}, \mathbf{u}_{t:T}). \tag{5.2}$$

At each time t, the posterior distribution of $\mathbf{z}_t$ depends on the previous latent state $\mathbf{z}_{t-1}$ and on the present and *future* observations $\mathbf{x}_{t:T}$ and inputs $\mathbf{u}_{t:T}$ (it is thus a first-order Markovian causal process on $\mathbf{z}_t$ combined with an anticausal process on $\mathbf{x}_t$ and $\mathbf{u}_t$).

Krishnan *et al.* (2015) indicated this structure and the fact that we should inspire from it to design the approximate posterior q_ϕ. However, somewhat surprisingly, they proposed the following four different models:

- an instantaneous model: $q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{u}_t)$ parameterized by an MLP;
- a model with local past and future context: $q_\phi(\mathbf{z}_t|\mathbf{x}_{t-1:t+1}, \mathbf{u}_{t-1:t+1})$ parameterized by an MLP;
- a model with the complete past context: $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{u}_{1:t})$ parameterized by an RNN;
- a model with the whole sequence: $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ parameterized by a bidirectional RNN.

We do not detail these implementations of q_ϕ here, as we will provide other detailed examples in the next chapters. One might wonder why $\mathbf{z}_{t-1}$ is not present in the conditional variables of the approximate posterior, but this might just be an oversight from the authors. This is difficult to be determined from the paper, as the implementation is not detailed. The authors mention an RNN to model the dependencies on $\mathbf{x}$ and $\mathbf{u}$, but the implementation of the dependency on $\mathbf{z}_{t-1}$ is not specified. The point is that, although the authors pointed out the dependency of the exact posterior on the present and future observations and inputs, they did not propose a corresponding approximate model.

Notation remark: Krishnan *et al.* (2015) denoted by $\mathbf{u}_{t-1}$ the input at time t in the generative model, as in many control theory papers published on SSMs, and not $\mathbf{u}_t$, as we do (as pointed out in a previous footnote). However, when defining the four approximate posterior models, they did it exactly as we report here (i.e., with $\mathbf{u}_t$ being synchronous to $\mathbf{z}_t$ and $\mathbf{x}_t$). We conjecture that this problem is just a notation mistake made by Krishnan *et al.* (2015), which we have implicitly corrected by using $\mathbf{u}_t$ instead of $\mathbf{u}_{t-1}$ as the input at time t in the generative model.

Krishnan *et al.* (2017) proposed the same generative model, renamed it DMM and presented it in the undriven mode, in which $\mathbf{u}_{1:T}$ was simply removed. However, they largely clarified and improved on their previous paper regarding the inference model. They proposed a new series of inference models that clearly do or do not depend on $\mathbf{z}_{t-1}$ and varied the dependency on the observed data. They again considered the case of dependency on the past and present data sequence $\mathbf{x}_{1:t}$ and on the complete data sequence $\mathbf{x}_{1:T}$. More importantly, they now also consider the case of an inference model with a functional form that corresponds exactly to the form of the exact posterior distribution, namely $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T})$. In this case, for a complete data sequence, we have

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T}). \tag{5.3}$$

This model is referred to as the deep Kalman smoother (DKS), as it combines information from the past, through $\mathbf{z}_{t-1}$, and information from the present and future observations.

For conciseness, we report the detailed inference equations only for the DKS and will comment on their extension to the other proposed inference models. The DKS is implemented with a backward RNN on $\mathbf{x}_t$, followed by an additional layer for combining the RNN output with $\mathbf{z}_{t-1}$:

$$\overleftarrow{\mathbf{g}}_t = e_{\overleftarrow{\mathbf{g}}}(\overleftarrow{\mathbf{g}}_{t+1}, \mathbf{x}_t), \tag{5.4}$$

$$\mathbf{g}_t = \frac{1}{2}(\tanh(\mathbf{W}\mathbf{z}_{t-1} + \mathbf{b}) + \overleftarrow{\mathbf{g}}_t), \tag{5.5}$$

$$[\boldsymbol{\mu}_\phi(\mathbf{g}_t), \boldsymbol{\sigma}_\phi(\mathbf{g}_t)] = e_{\mathbf{z}}(\mathbf{g}_t), \tag{5.6}$$

$$q_\phi(\mathbf{z}_t|\mathbf{g}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_\phi(\mathbf{g}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{g}_t)\}). \tag{5.7}$$

In the above equations, $e_{\mathbf{z}}$ is a basic combining network, parameterized by $\phi_{\mathbf{z}}$, $\boldsymbol{\mu}_\phi(\mathbf{g}_t)$ is an affine function of $\mathbf{g}_t$, and $\boldsymbol{\sigma}^2_\phi(\mathbf{g}_t)$ is a Softplus of an affine function of $\mathbf{g}_t$. We thus have $\phi = \phi_{\overleftarrow{\mathbf{g}}} \cup \phi_{\mathbf{z}}$, assuming $\{\mathbf{W}, \mathbf{b}\} \in \phi_{\mathbf{z}}$ for simplicity.

Because of the recursivity in (5.4), we can see $\mathbf{g}_t$ as an unfolded deterministic function of $\mathbf{z}_{t-1}$ and $\mathbf{x}_{t:T}$, which we can rewrite as $\mathbf{g}_t =$

$\mathbf{g}_t(\mathbf{z}_{t-1}, \mathbf{x}_{t:T})$,[3] and we have $q_\phi(\mathbf{z}_t|\mathbf{g}_t) = q_\phi(\mathbf{z}_t|\mathbf{g}_t(\mathbf{z}_{t-1}, \mathbf{x}_{t:T}))$. For a complete data sequence, we have

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{g}_t) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{g}_t(\mathbf{z}_{t-1}, \mathbf{x}_{t:T})), \tag{5.8}$$

which is just a rewriting of (5.3). In short, the inference of $\mathbf{z}_t$ with a DKS requires a first backward pass from $\mathbf{x}_T$ up to $\mathbf{x}_t$ to compute $\overleftarrow{\mathbf{g}}_t$, which is then combined with $\mathbf{z}_{t-1}$.

As mentioned above, this inference model is extended to a noncausal (bidirectional) model regarding the observations, $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T})$. This is done by adding a forward RNN on $\mathbf{x}_t$ and sending its output $\overrightarrow{\mathbf{g}}_t$ to the combining network, in addition to $\mathbf{z}_{t-1}$ and $\overleftarrow{\mathbf{g}}_t$. For conciseness, we do not report the corresponding detailed equations, but this more general model is represented in Figure 5.1. The other models proposed by Krishnan *et al.* (2017) can be deduced from this general model by removing some elements. In particular, DKS is obtained by simply removing the forward RNN. Moreover, models that do not depend on $\mathbf{z}_{t-1}$ are obtained by removing the arrows between $\mathbf{z}_{t-1}$ and $\mathbf{g}_t$ for all t.

5.3 Training

A comparison of the compact form of the DKF model in (3.13) with the general compact form of a DVAE in (4.4) shows that the DKF model makes the following conditional independence assumptions:

$$\begin{aligned} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) &= p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t); \\ p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) &= p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t). \end{aligned} \tag{5.9}$$

Using these two simplifications along with the inference model (5.3) (extended to be in the driven mode for the sake of generality), the VLB

[3] This function is also a function of the initial state $\overleftarrow{\mathbf{g}}_T$ of the backward RNN.

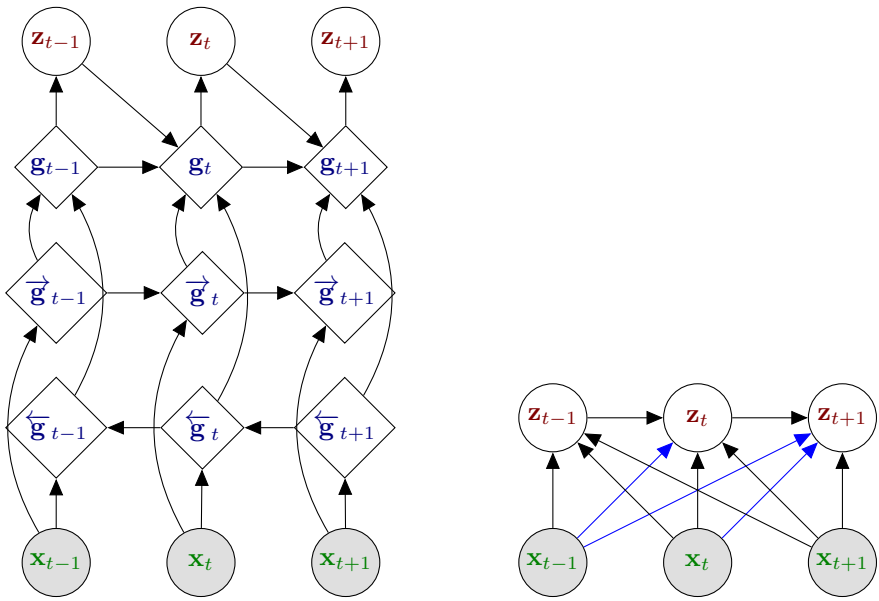


Figure 5.1: Graphical model of DKF at inference time corresponding to the inference model $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T})$, in developed form (left) and compact form (right). The specific DKS model, which functional form $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T})$ perfectly corresponds to the form of the exact posterior distribution, is obtained by removing the forward RNN (and removing the blue arrows on the right-hand schema).

in its most general form (4.24) can be simplified as follows:

$$\mathcal{L}(\theta, \phi; \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\left[\log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t)\right] - \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{t-1}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\left[D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T}, \mathbf{u}_{t:T}) \parallel p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t)\right)\right]. \tag{5.10}$$

The KL divergence in (5.10) can be computed analytically, while the two expectations are intractable. (Krishnan *et al.*, 2015; Krishnan *et al.*, 2017) provided no detailed information regarding how to approximate these expectations; they only mentioned that "stochastic backpropagation" is used, referring the reader to the papers of Kingma and Welling (2014) and Rezende *et al.* (2014), who introduced the reparameterization trick for standard "static" VAEs. However, due to the dynamical nature of the model, the sampling procedure required for stochastic backpropagation in DKF is more complicated than in standard VAEs. In particular, we do not have an analytical form for $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, and only have one for $q_\phi(\mathbf{z}_\tau|\mathbf{z}_{\tau-1}, \mathbf{x}_{\tau:T}, \mathbf{u}_{\tau:T})$. Therefore, we need to exploit the chain rule and the "cascade trick" to develop and then approximate the intractable expectations in (5.10). The first expectation in this VLB expression can be developed as follows:

$$\begin{aligned}\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}[f(\mathbf{z}_t)] &= \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}[f(\mathbf{z}_t)] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}_1|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}[\\ &\quad \mathbb{E}_{q_\phi(\mathbf{z}_2|\mathbf{z}_1,\mathbf{x}_{2:T},\mathbf{u}_{2:T})}[\dots \\ &\quad \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T},\mathbf{u}_{t:T})}[f(\mathbf{z}_t)]\dots]], \end{aligned} \tag{5.11}$$

where $f(\mathbf{z}_t)$ denotes an arbitrary function of $\mathbf{z}_t$. A similar procedure can be used to develop the second expectation in (5.10). Each intractable expectation in this cascade of expectations can then be approximated with a Monte Carlo estimate. It requires to sample $q_\phi(\mathbf{z}_\tau|\mathbf{z}_{\tau-1}, \mathbf{x}_{\tau:T}, \mathbf{u}_{\tau:T})$ iteratively from $\tau = 1$ to t, using the same reparametrization trick as in standard VAEs. Doing so, the VLB becomes differentiable w.r.t. $\theta = \theta_\mathbf{x} \cup \theta_\mathbf{z}$ and $\phi = \phi_{\overleftarrow{\mathbf{g}}} \cup \phi_\mathbf{z}$, and it can be optimized with gradient-ascent-based techniques.

6

Kalman Variational Autoencoders

The KVAE model was presented by Fraccaro *et al.* (2017). This model can be considered a variant of the DKF model, and hence as another deep SSM, where an additional random variable, denoted $\mathbf{a}_t$, is inserted between the latent vector $\mathbf{z}_t$ and the observed vector $\mathbf{x}_t$, as illustrated in Figure 6.1. This enables us to separate the model into two parts: A deep feature extractor linking $\mathbf{a}_t$ and $\mathbf{x}_t$ and the dynamical model on $\mathbf{z}_t$ with "new observations" $\mathbf{a}_t$. As we will see below, this provides the model with interesting properties for inference and training.

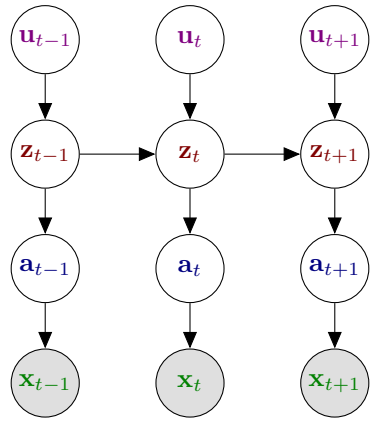


Figure 6.1: KVAE's graphical model.

6.1 Generative model

The general formulation of the KVAE model is given by

$$[\boldsymbol{\mu}_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t), \boldsymbol{\sigma}_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t)] = d_\mathbf{z}(\mathbf{z}_{t-1}, \mathbf{u}_t), \tag{6.1}$$

$$p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{u}_t)\}), \tag{6.2}$$

$$[\boldsymbol{\mu}_{\theta_\mathbf{a}}(\mathbf{z}_t), \boldsymbol{\sigma}_{\theta_\mathbf{a}}(\mathbf{z}_t)] = d_\mathbf{a}(\mathbf{z}_t), \tag{6.3}$$

$$p_{\theta_\mathbf{a}}(\mathbf{a}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{a}_t; \boldsymbol{\mu}_{\theta_\mathbf{a}}(\mathbf{z}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{a}}(\mathbf{z}_t)\}), \tag{6.4}$$

$$[\boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{a}_t), \boldsymbol{\sigma}_{\theta_\mathbf{x}}(\mathbf{a}_t)] = d_\mathbf{x}(\mathbf{a}_t), \tag{6.5}$$

$$p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{a}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{a}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{x}}(\mathbf{a}_t)\}). \tag{6.6}$$

In Fraccaro *et al.*'s (2017) paper, (6.1) and (6.3) are linear equations; that is, they are given as (3.16) and (3.17), respectively (with $\mathbf{a}$ in place of $\mathbf{x}$, null bias vectors $\mathbf{n}_t$ and $\mathbf{m}_t$, and time-invariant covariance matrices $\boldsymbol{\Lambda}$ and $\boldsymbol{\Sigma}$). Therefore, we have $\theta_\mathbf{z} = \{\mathbf{A}_t, \mathbf{B}_t\}_{t=1}^T \cup \{\boldsymbol{\Lambda}\}$ and $\theta_\mathbf{a} = \{\mathbf{C}_t\}_{t=1}^T \cup \{\boldsymbol{\Sigma}\}$, and the submodel on $\{\mathbf{u}_t, \mathbf{z}_t, \mathbf{a}_t\}$ is a classical (non-deep) LG-LDS. In contrast, $d_\mathbf{x}$ in (6.5) is implemented with a DNN, with parameter set $\theta_\mathbf{x}$ (e.g., a basic MLP or a CNN for video sequence modeling). This network plays the role of a deep feature extractor, with the dimension of $\mathbf{a}_t$ being possibly much smaller than that of $\mathbf{x}_t$. The feature vector $\mathbf{a}_t$ is expected to encode the properties of the "object(s)"

present in the observation $\mathbf{x}_t$, whereas $\mathbf{z}_t$ is expected to encode the dynamics of these objects, which is an important application of the previously described disentanglement concept. As we will see, in the KVAE case, this can be a great advantage for solving the dynamical part of the model.

The joint distribution of all random variables can be factorized as follows:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{a}_{1:T}, \mathbf{z}_{1:T}, \mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{a}_t) p_{\theta_\mathbf{a}}(\mathbf{a}_t|\mathbf{z}_t) p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t) p(\mathbf{u}_t), \tag{6.7}$$

and we also have

$$p_{\theta_\mathbf{x}}(\mathbf{x}_{1:T}|\mathbf{a}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{a}_t), \tag{6.8}$$

$$p_{\theta_\mathbf{a}}(\mathbf{a}_{1:T}|\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{a}}(\mathbf{a}_t|\mathbf{z}_t), \tag{6.9}$$

$$p_{\theta_\mathbf{z}}(\mathbf{z}_{1:T}|\mathbf{u}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{u}_t). \tag{6.10}$$

Given the state sequence $\mathbf{z}_{1:T}$, the features $\mathbf{a}_{1:T}$ are independent, and given the sequence of features, the observations $\mathbf{x}_{1:T}$ are independent.

Fraccaro *et al.* (2017) mentioned the classical limitation of LDS for modeling abrupt changes in trajectories. A classical solution to this problem is to include in the model a "switching strategy" between different models or different parameterizations of the model, see, e.g., the switching Kalman filter (Murphy, 1998; Fox *et al.*, 2011). Fraccaro *et al.* (2017) proposed to define each LDS parameter at each time t (e.g., matrix $\mathbf{A}_t$) as a linear combination of predefined matrices/vectors from a parameter bank, and the coefficients of the linear combination were estimated at each time t from the past features $\mathbf{a}_{1:t-1}$ using an LSTM network. Although it is an interesting contribution on its own, we do not further consider this part of the KVAE model here, as it is loosely relevant to our model review. A similar transition model was proposed independently by Karl *et al.* (2017) as an instance of a deep variational Bayesian filter (DVBF), an extended class of SSM-based DVAE models

enriched with stochastic transition parameters (see also Watter *et al.* (2015)).

6.2 Inference model

For the KVAE model, the posterior distribution of all latent variables given the observed variables, $p_\theta(\mathbf{a}_{1:T}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, can be factorized as follows:

$$\begin{aligned} p_\theta(\mathbf{a}_{1:T}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) &= p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T}) p_\theta(\mathbf{a}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}), \\ &= p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T}, \mathbf{u}_{1:T}) p_\theta(\mathbf{a}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}), \end{aligned} \tag{6.11}$$

where the simplification of the first term on the right-hand side results from D-separation. A keypoint that appears here is that, if the sequence of features $\mathbf{a}_{1:T}$ is known, then $p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T}, \mathbf{u}_{1:T})$ has a closed-form solution, which is a Kalman filter or a Kalman smoother (see Section 3.2.2). This Kalman solution is classic, and it is not detailed here for conciseness. We can simply mention that it depends only on $\theta_\mathbf{a} \cup \theta_\mathbf{z}$ but not on $\theta_\mathbf{x}$. The other factor, $p_\theta(\mathbf{a}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, is more problematic.

Fraccaro *et al.* (2017) did not discuss the form of the exact posterior distribution, yet they proposed the following factorized inference model, which exploits the Kalman solution:

$$q_\phi(\mathbf{a}_{1:T}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T}) = p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T}, \mathbf{u}_{1:T}) q_\phi(\mathbf{a}_{1:T}|\mathbf{x}_{1:T}) \tag{6.12}$$

$$= p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T}, \mathbf{u}_{1:T}) \prod_{t=1}^{T} q_\phi(\mathbf{a}_t|\mathbf{x}_t), \tag{6.13}$$

where

$$q_\phi(\mathbf{a}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{a}_t; \boldsymbol{\mu}_\phi(\mathbf{x}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{x}_t)\}) \tag{6.14}$$

is implemented with a fully-connected DNN:

$$[\boldsymbol{\mu}_\phi(\mathbf{x}_t), \boldsymbol{\sigma}_\phi(\mathbf{x})] = e_\mathbf{a}(\mathbf{x}_t). \tag{6.15}$$

Eqs. (6.5), (6.6), (6.14), and (6.15) are identical to (2.5), (2.3), (2.18), and (2.20), respectively, with $\mathbf{a}_t$ substituting $\mathbf{x}$ and $\mathbf{z}_t$ substituting $\mathbf{z}$. This means that, with the proposed inference model, a KVAE is

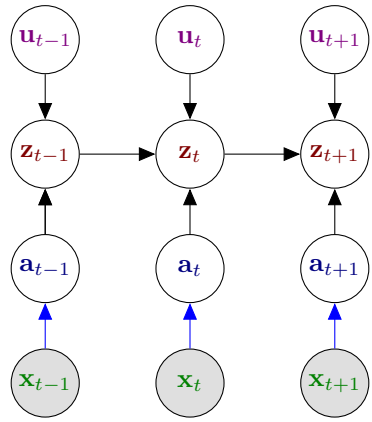


Figure 6.2: KVAE's graphical model at inference time. The black arrows represent the Kalman filter solution (causal solution) of LG-LDS on $\{\mathbf{u}_{1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{1:T}\}$. The blue arrows represent the VAE encoder from $\mathbf{x}_t$ to $\mathbf{a}_t$. The inference solution for the complete KVAE model is a combination of these two.

composed of a VAE modeling the relationship between $\mathbf{a}_t$ and $\mathbf{x}_t$, placed on top of an LG-LDS on $\mathbf{u}_t$, $\mathbf{z}_t$ and $\mathbf{a}_t$. Therefore, the inference solution of the VAE and that of LG-LDS can be combined for the solution of this combined model. This is illustrated in Figure 6.2.

This inference model is thus designed to benefit from both the VAE methodology and the well-known efficiency of the "simple" LG-LDS model for tracking data dynamics. The Kalman solution, which requires inverse matrix calculation, greatly benefits from the notable $\mathbf{x}_t$-to-$\mathbf{a}_t$ dimension reduction. In addition, although this was not exactly discussed in these terms by Fraccaro *et al.* (2017), one possible motivation for designing the KVAE model is that the joint learning of all parameters (see the next subsection) can encourage the feature extractor to provide $\mathbf{a}_t$ features that are well-suited to a linear dynamical model; that is, the nonlinear relations between observations $\mathbf{x}_t$ and dynamics $\mathbf{z}_t$ are (at least largely) captured by the VAE.

6.3 Training

The VLB for the KVAE model writes as follows:

$$\mathcal{L}(\phi,\theta;\mathbf{x}_{1:T},\mathbf{u}_{1:T}) = \mathbb{E}_{q(\mathbf{z}_{1:T},\mathbf{a}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T})}\Big[\log p_\theta(\mathbf{x}_{1:T}|\mathbf{z}_{1:T},\mathbf{a}_{1:T},\mathbf{u}_{1:T})\Big]$$
$$- D_{KL}(q_\phi(\mathbf{z}_{1:T},\mathbf{a}_{1:T}|\mathbf{x}_{1:T},\mathbf{u}_{1:T}) \parallel p_\theta(\mathbf{z}_{1:T},\mathbf{a}_{1:T}|\mathbf{u}_{1:T})) \tag{6.16}$$
$$= \mathbb{E}_{q_\phi(\mathbf{a}_{1:T}|\mathbf{x}_{1:T})p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T},\mathbf{u}_{1:T})}\Big[\log p_\theta(\mathbf{x}_{1:T}|\mathbf{a}_{1:T})\Big]$$
$$- D_{KL}(q_\phi(\mathbf{a}_{1:T}|\mathbf{x}_{1:T})p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T},\mathbf{u}_{1:T}) \parallel p_\theta(\mathbf{z}_{1:T},\mathbf{a}_{1:T}|\mathbf{u}_{1:T})) \tag{6.17}$$
$$= \mathbb{E}_{q_\phi(\mathbf{a}_{1:T}|\mathbf{x}_{1:T})}\Big[\log p_\theta(\mathbf{x}_{1:T}|\mathbf{a}_{1:T})/q_\phi(\mathbf{a}_{1:T}|\mathbf{x}_{1:T})$$
$$- D_{KL}(p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T},\mathbf{u}_{1:T}) \parallel p_\theta(\mathbf{a}_{1:T}|\mathbf{z}_{1:T})p_\theta(\mathbf{z}_{1:T}|\mathbf{u}_{1:T}))\Big], \tag{6.18}$$

where we use the proposed decomposition of both the generative and inference models.

In practice, one first samples from $q_\phi(\mathbf{a}_t|\mathbf{x}_t)$ for each t. These samples are fed to a standard Kalman smoother that computes $p_\theta(\mathbf{z}_{1:T}|\mathbf{a}_{1:T},\mathbf{u}_{1:T})$. We can then easily sample from this distribution. This allows for jointly learning the parameters of the VAE (both the encoder and decoder) and those of the LG-LDS.

7

STOchastic Recurrent Networks

To the best of our knowledge, STORN (Bayer and Osendorfer, 2014) is the first DVAE model to combine an internal deterministic state $\mathbf{h}_t$ and an internal stochastic state $\mathbf{z}_t$. Bayer and Osendorfer (2014) presented STORN in the undriven and predictive modes (i.e., with $\mathbf{u}_t = \mathbf{x}_{t-1}$). Hereinafter, we retain this mode for STORN, VRNN, and SRNN, for an easier comparison, but these models can also be easily set up in the driven mode with an external input $\mathbf{u}_t$.

7.1 Generative model

The STORN observation model is given by

$$\mathbf{h}_t = d_{\text{hid}}(\mathbf{W}_{\text{in}}\mathbf{x}_{t-1} + \mathbf{W}_{\text{lat}}\mathbf{z}_t + \mathbf{W}_{\text{rec}}\mathbf{h}_{t-1} + \mathbf{b}_{\text{hid}}), \tag{7.1}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{h}_t)] = d_{\text{out}}(\mathbf{W}_{\text{out}}\mathbf{h}_t + \mathbf{b}_{\text{out}}), \tag{7.2}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t), \text{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{h}_t)\}). \tag{7.3}$$

Therefore, (7.2) and (7.3) are the same as those for a basic single-layer GRNN, but in STORN, $\mathbf{z}_t$ forms an input additional to the internal state $\mathbf{h}_t$. Moreover, STORN was originally presented in the above framework

of a single-layer RNN, but it can be easily generalized to a deep RNN, defined by (3.5)–(3.7), by inserting $\mathbf{z}_t$ as an additional input to the network (and setting $\mathbf{u}_t = \mathbf{x}_{t-1}$):

$$\mathbf{h}_t = d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1}), \tag{7.4}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{h}_t)] = d_{\mathbf{x}}(\mathbf{h}_t), \tag{7.5}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{h}_t)\}). \tag{7.6}$$

In the following, we retain this latter more general formulation for easier comparison with the other models. We denote by $\theta_{\mathbf{h}}$ and $\theta_{\mathbf{hx}}$ the set of parameters of the networks implementing $d_{\mathbf{h}}$ and $d_{\mathbf{x}}$, respectively, and have $\theta_{\mathbf{x}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hx}}$.

In STORN, $\mathbf{z}_t$ is assumed i.i.d. with a standard Gaussian distribution:

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_{\mathbf{z}}}(\mathbf{z}_t) \qquad \text{with} \qquad p_{\theta_{\mathbf{z}}}(\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_t; \mathbf{0}, \mathbf{I}_L). \tag{7.7}$$

In short, there is no temporal model on the prior distribution of $\mathbf{z}_t$ and $\theta_{\mathbf{z}} = \emptyset$ (and therefore $\theta = \theta_{\mathbf{x}}$). Here, it is the temporal recursion on $\mathbf{h}_t$ and the use of $\mathbf{h}_t$ to generate $\mathbf{x}_t$ that makes STORN a member of the DVAE family. The graphical model of STORN is shown in Figure 7.1 (left).

Notation remark: To ensure homogeneous notations across models, we slightly change the notation used by Bayer and Osendorfer (2014) by "synchronizing" $\mathbf{x}_t$ and $\mathbf{h}_t$; that is, in our presentation of STORN, $\mathbf{x}_t$ is generated from $\mathbf{h}_t$ (which is generated from $\mathbf{x}_{t-1}$, $\mathbf{h}_{t-1}$, and $\mathbf{z}_t$). In contrast, in Eq. (4) in Bayer and Osendorfer's (2014) paper, $\mathbf{x}_{t+1}$ is generated from $\mathbf{h}_t$ (which is generated from $\mathbf{x}_t$, $\mathbf{h}_{t-1}$, and $\mathbf{z}_t$). In other words, we replace $\mathbf{x}_t$ with $\mathbf{x}_{t-1}$. This change of notation does not change the model in essence but makes the comparison with other models easier.

Eq. (7.4) shows that the two states, $\mathbf{h}_t$ and $\mathbf{z}_t$, are intricate, and the interpretation of $\mathbf{h}_t$ as a deterministic state is now an issue. In fact, $\mathbf{h}_t$ is now a random variable, but it is not a "free" one, as it is a deterministic function of the latent random variables $\mathbf{z}_t$ and $\mathbf{x}_{t-1}$ and of its previous

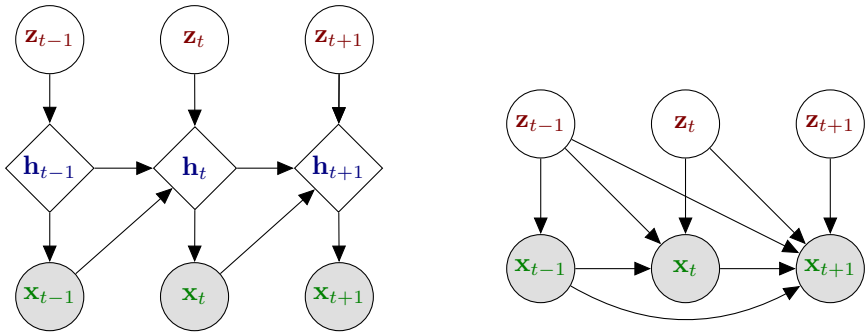


Figure 7.1: STORN's graphical model in developed form (left) and compact form (right).

value $\mathbf{h}_{t-1}$. We present a proper treatment of this issue in Appendix A. The recurrence on $\mathbf{h}_t$ is unfolded to consider $\mathbf{h}_t$ as a deterministic function of $\mathbf{z}_{1:t}$ and $\mathbf{x}_{1:t-1}$, which we denote $\mathbf{h}_t = \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$,[1] and we consider a Dirac probability distribution over $\mathbf{h}_t$, positioned at that function's output value. These points were briefly discussed by Bayer and Osendorfer (2014). In Appendix A, it is shown that marginalizing the joint density $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{h}_{1:T})$ w.r.t. $\mathbf{h}_{1:T}$ leads to

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}))p(\mathbf{z}_t). \tag{7.8}$$

From the above equation and (7.7), we deduce the conditional distribution

$$p_{\theta_\mathbf{x}}(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})). \tag{7.9}$$

Note that these data sequence densities factorize across time frames, but the whole history of the present and past latent variables and past outputs is necessary to generate the present output. This history is summarized in $\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$.

In line with the discussion in Section 4.1.2, an alternate description of STORN can be written, where we remove the internal deterministic

[1] This function also depends on the initial vectors $\mathbf{x}_0$ and $\mathbf{h}_0$ (and we can set $\mathbf{x}_0 = \emptyset$), but we omit them for clarity of presentation.

state $\mathbf{h}$ and express the model only in terms of the free random variables $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})p(\mathbf{z}_t), \tag{7.10}$$

and

$$p_{\theta_\mathbf{x}}(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}). \tag{7.11}$$

The above two equations are more general than (7.8) and (7.9), but they lose some information on the deterministic link between $\mathbf{x}_{1:t-1}$ and $\mathbf{z}_{1:t}$ in the process of generating $\mathbf{x}_t$. The compact graphical representation corresponding to this alternate formulation is given in Figure 7.1 (right).

7.2 Inference model

Following Section 4.2, it is easy to show that the exact posterior distribution of STORN takes the following form:

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}). \tag{7.12}$$

In fact, this expression is obtained by the chain rule, and it cannot be simplified by applying D-separation. This is because any vector in $\mathbf{z}_{1:t-1}$ and $\mathbf{x}_{1:T}$ is either a child, a parent, or a co-parent of $\mathbf{z}_t$ in the graphical representation of STORN. In other words, each product term at time t in (7.12) depends on the past observed and latent state vectors that propagate through the internal state, and it depends on the present and future observed vectors, as $\mathbf{z}_t$ propagates to them through the internal hidden states.

To complement the discussion on the form of the exact posterior distribution, we note the following:

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) \propto p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = p_{\theta_\mathbf{x}}(\mathbf{x}_{1:T}|\mathbf{z}_{1:T})p(\mathbf{z}_{1:T}). \tag{7.13}$$

Thus, combining the above equation with (7.8), we get

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) \propto \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}))p(\mathbf{z}_t). \tag{7.14}$$

As $\mathbf{z}_t$ is present in all terms of $\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$ from t to T, it is confirmed that (7.14), considered as a function of $\mathbf{z}_t$, depends not only on $\mathbf{x}_{1:t-1}$ but also on $\mathbf{x}_{t:T}$.

As for the practical inference in STORN, the approximate posterior distribution q_ϕ was chosen by Bayer and Osendorfer (2014) as generated by an additional *forward* RNN. Little information is available on the implementation. The parameters of q_ϕ are said to be generated from $\mathbf{x}_{1:t}$, which is slightly odd as $\mathbf{x}_{t+1}$ was generated from $\mathbf{z}_t$ (with their notations; see our remark in the previous subsection). There is thus a one-step lag between generation and inference, which is difficult to justify (in practice, we have found that this leads to significantly inferior inference performance). With our change in notation at generation, we somehow automatically compensate for this gap and assume that the "correct" detailed inference equations are given as

$$\mathbf{g}_t = e_{\mathbf{g}}(\mathbf{W}_{\text{in}}^{\text{enc}}\mathbf{x}_t + \mathbf{W}_{\text{rec}}^{\text{enc}}\mathbf{g}_{t-1} + \mathbf{b}_{\text{hid}}^{\text{enc}}), \tag{7.15}$$

$$[\boldsymbol{\mu}_\phi(\mathbf{g}_t), \boldsymbol{\sigma}_\phi(\mathbf{g}_t)] = e_{\mathbf{z}}(\mathbf{W}_{\text{out}}^{\text{enc}}\mathbf{g}_t + \mathbf{b}_{\text{out}}^{\text{enc}}), \tag{7.16}$$

$$q_\phi(\mathbf{z}_t|\mathbf{g}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_\phi(\mathbf{g}_t), \text{diag}\{\boldsymbol{\sigma}_\phi^2(\mathbf{g}_t)\}), \tag{7.17}$$

where $\mathbf{g}_t$ denotes the inference RNN internal state,[2] and $e_{\mathbf{g}}$ and $e_{\mathbf{z}}$ are nonlinear activation functions. Similarly to the generative model, because of the recursivity in (7.15), $\mathbf{g}_t$ can be considered as an unfolded deterministic function of $\mathbf{x}_{1:t}$, which we note $\mathbf{g}_t = \mathbf{g}_t(\mathbf{x}_{1:t})$.[3] For a complete data sequence, we have

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{g}_t) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{g}_t(\mathbf{x}_{1:t})). \tag{7.18}$$

This inference model can be rewritten in compact form as

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t}). \tag{7.19}$$

The corresponding graphical model is shown in Figure 7.2. Note that this inference model is inconsistent with the exact posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ at several points: At each time t, it neither considers

[2] In Bayer and Osendorfer's (2014) paper, it is denoted as $\mathbf{h}_t^r$.

[3] This function is also a function of $\mathbf{g}_0$.

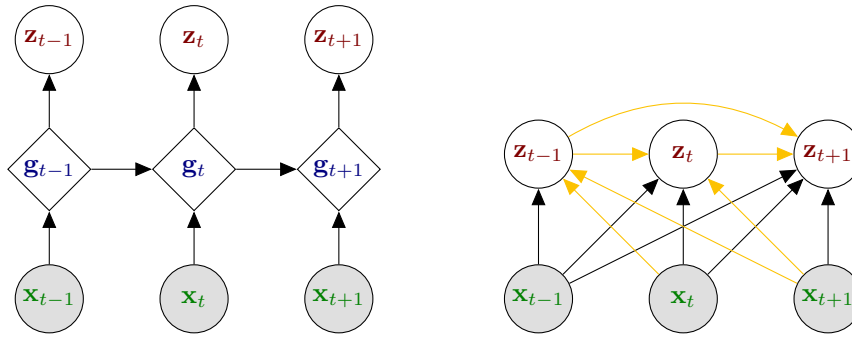


Figure 7.2: STORN's graphical model at inference time in developed form (left) and compact form (right). Golden arrows correspond to missing links on the proposed probabilistic dependencies (compared to the exact inference dependencies).

future observations $\mathbf{x}_{t+1:T}$ nor past latent states $\mathbf{z}_{1:t-1}$. As discussed in Section 4.2.3, the internal states of the encoder and of decoder can be identical, or they can be different. In STORN, given the choice of the inference model, these internal states depend on different variables and, therefore, are different.

7.3 Training

A comparison of the compact form of STORN in (7.10) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$) shows that STORN makes the following conditional independence assumption:

$$p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = p(\mathbf{z}_t). \tag{7.20}$$

Using this single simplification, the VLB given in its most general form in (4.24) becomes

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}_{1:T}) = &\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T})}\left[\log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\right] \\ &- \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t-1}|\mathbf{x}_{1:T})}\left[D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) \parallel p(\mathbf{z}_t)\right)\right].\end{aligned} \tag{7.21}$$

This expression of the VLB relies on an inference model that is consistent with the exact posterior distribution (7.12). However, as discussed above,

STORN assumes an inference model of the form: $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) = q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})$. Consequently, the VLB in (7.21) can be simplified as follows:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}_{1:T}) = &\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T})}\left[\log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\right] \\ &- \sum_{t=1}^{T} D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t}) \parallel p(\mathbf{z}_t)\right). \end{aligned} \tag{7.22}$$

The KL divergence in this expression can be computed analytically, while the expectation is intractable and should be approximated by a Monte Carlo estimate, using samples drawn recursively from $q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T})$ based on the inference model (7.18). As for DKF, using the reparameterization trick for this recursive sampling leads to an objective function, which is differentiable w.r.t. θ and ϕ.

8

Variational Recurrent Neural Networks

The VRNN model was proposed by Chung *et al.* (2015) as a combination of a VAE and an RNN.

8.1 Generative model

The VRNN observation model is given by

$$\mathbf{h}_t = d_{\mathbf{h}}(\varphi_{\mathbf{x}}(\mathbf{x}_{t-1}), \varphi_{\mathbf{z}}(\mathbf{z}_{t-1}), \mathbf{h}_{t-1}), \tag{8.1}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t)] = d_{\mathbf{x}}(\varphi_{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_t), \tag{8.2}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t)\}), \tag{8.3}$$

where $\varphi_{\mathbf{z}}$ and $\varphi_{\mathbf{x}}$ are feature extractors, which were mentioned by Chung *et al.* (2015) to be important in practice. These feature extractors are DNNs parameterized by a set of weights and biases, denoted as τ.

The generative distribution of $\mathbf{z}_t$ is given by

$$[\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{h}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{z}}}(\mathbf{h}_t)] = d_{\mathbf{z}}(\mathbf{h}_t), \tag{8.4}$$

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{h}_t)\}). \tag{8.5}$$

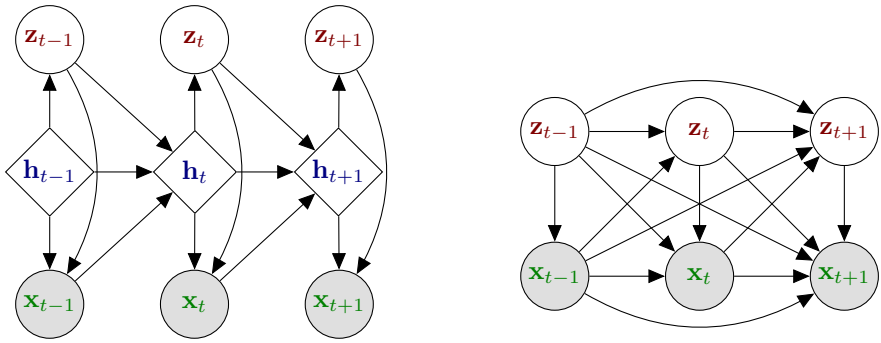


Figure 8.1: VRNN's graphical model in developed (left) and compact (right) forms.

Notation remark: In Chung *et al.*'s (2015) paper, τ denotes the set of parameters for both feature extractors, which are denoted as $\varphi_\tau^{\mathbf{x}}$ and $\varphi_\tau^{\mathbf{z}}$, respectively, as well as for $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$, which are denoted $\varphi_\tau^{\text{dec}}$ and $\varphi_\tau^{\text{prior}}$, respectively. Moreover, $d_{\mathbf{h}}$ is denoted d_θ. We find this a bit confusing and prefer to distinguish among τ, $\theta_{\mathbf{x}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hx}}$, $\theta_{\mathbf{z}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hz}}$, and $\theta = \tau \cup \theta_{\mathbf{x}} \cup \theta_{\mathbf{z}}$. Moreover, one may also want to distinguish between τ_x and τ_z to clarify that the two feature extractors are different. We retain τ for simplicity. Besides, we replace $\mathbf{h}_{t-1}$ in the paper by Chung *et al.* (2015) with $\mathbf{h}_t$ to synchronize $\mathbf{h}_t$, $\mathbf{z}_t$, and $\mathbf{x}_t$; that is, $\mathbf{x}_t$ is generated from $\mathbf{h}_t$ and $\mathbf{z}_t$. This arbitrary reindexing of $\mathbf{h}_t$ does not change the model conceptually.

In VRNN, we thus have multiple intrications of $\mathbf{h}_t$ and $\mathbf{z}_t$ in both the observation model and the distribution of $\mathbf{z}_t$. The graphical model of VRNN is given in Figure 8.1 (left). The generative process starts with an initial internal state $\mathbf{h}_1$, from which we generate $\mathbf{z}_1$. From $\mathbf{h}_1$ and $\mathbf{z}_1$, we generate $\mathbf{x}_1$. Then, $\mathbf{h}_2$ is deterministically calculated from $\mathbf{h}_1$, $\mathbf{z}_1$, and $\mathbf{x}_1$, and so on, except that hereinafter, $\mathbf{z}_t$ is generated from $\mathbf{z}_{t-1}$, $\mathbf{h}_{t-1}$, *and* $\mathbf{x}_{t-1}$. Using the "unfolding the recurrence" trick mentioned in the previous sections, we can here denote $\mathbf{h}_t = \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$[1] and have $p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{h}_t) = p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}))$ and $p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t) = p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}))$. This provides both $\mathbf{z}_t$ and $\mathbf{x}_t$ with an implicit temporal model.

[1] This function also depends on $\mathbf{h}_1$, which is omitted for clarity of presentation.

As for a data sequence, when marginalizing the joint distribution of all variables w.r.t. $\mathbf{h}_{1:T}$ following the line of Appendix A, we get

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}))p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})) \tag{8.6}$$

$$= \prod_{t=1}^{T} p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})). \tag{8.7}$$

Again, we have a factorization of the conditional densities over time frames. However, we do *not* have conditional independence of $\mathbf{x}_t$ and $\mathbf{z}_t$ conditionally to the state $\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$ due to the direct link from $\mathbf{z}_t$ to $\mathbf{x}_t$.

As for STORN, we can provide a more general alternate expression for $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$ that does not make the internal state explicit but only represents the general dependencies among the "free" random variables:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) \tag{8.8}$$

$$= \prod_{t=1}^{T} p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}). \tag{8.9}$$

The corresponding compact graphical model is shown in Figure 8.1 (right).

8.2 Inference model

The general form of the exact posterior distribution of VRNN is identical to the one of STORN; that is, it factorizes into

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}), \tag{8.10}$$

and, here also, no further simplification can be obtained from D-separation.

The approximate posterior distribution q_ϕ was chosen by Chung

et al. (2015) as

$$[\boldsymbol{\mu}_\phi(\mathbf{x}_t, \mathbf{h}_t), \boldsymbol{\sigma}_\phi(\mathbf{x}_t, \mathbf{h}_t)] = e_{\mathbf{z}}(\varphi_{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_t), \tag{8.11}$$

$$q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{h}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_\phi(\mathbf{x}_t, \mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{x}_t, \mathbf{h}_t)\}), \tag{8.12}$$

where $e_{\mathbf{z}}$ is the encoder DNN, parameterized by $\phi_{\mathbf{z}}$. As for data sequence inference, we have

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{x}_t, \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})). \tag{8.13}$$

In contrast to STORN, the same internal state $\mathbf{h}_t$ is here shared by the VRNN encoder and decoder, which, in our opinion, makes the approximate model more consistent with the exact posterior distribution. This makes the set of parameters $\theta_{\mathbf{h}}$ common to the encoder and decoder. Because the feature extractor $\varphi_{\mathbf{x}}(\mathbf{x}_t)$ is also used at the encoder, the same remark applies to its parameter set $\tau_{\mathbf{x}}$. In addition, the inference at time t depends on past outputs *and* past latent states, which also makes the inference model closer to the exact posterior distribution. However, compared to the exact posterior, the future observations (from $t+1$ to T) are missing again. In short, here also, the approximate inference is causal, whereas the exact posterior distribution is noncausal. The graphical model corresponding to the VRNN approximate inference process is shown in Figure 8.2. The inference model can be rewritten in the following general form:

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t}). \tag{8.14}$$

8.3 Training

A comparison of the compact form of VRNN in (8.8) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$) shows that VRNN does not make any conditional independence assumption in the generative model. In this sense, VRNN is the most general DVAE model we have seen so far. The expression of the VLB for VRNN should therefore be the one given in (4.24). However, as discussed above,

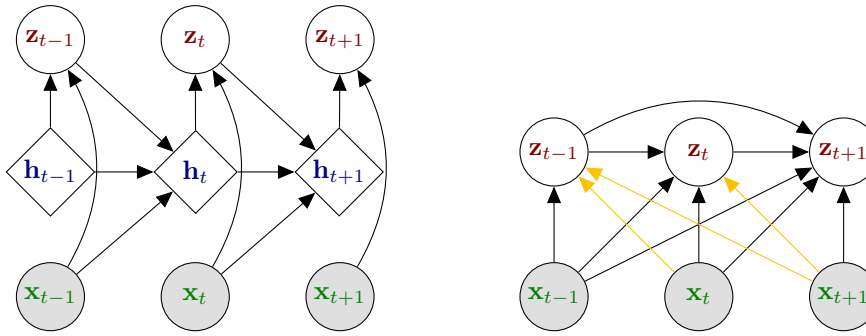


Figure 8.2: VRNN's graphical model at inference time in developed form (left) and compact form (right). Golden arrows correspond to missing links on the proposed probabilistic dependencies (compared to the exact inference dependencies).

the inference model in VRNN is inconsistent with the exact posterior distribution, as the following conditional independence assumption is made: $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) = q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})$. Consequently, the VLB in (4.24) can be simplified as follows:

$$\mathcal{L}(\theta, \phi; \mathbf{x}_{1:T}) = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T})} \big[\log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) \big] - \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t-1}|\mathbf{x}_{1:T})} \left[D_{\mathrm{KL}} \left(q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t}) \parallel p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) \right) \right]. \tag{8.15}$$

As for the previously presented DVAE models, the KL divergence can be computed analytically, and intractable expectations are approximated by Monte Carlo estimates.

8.4 Improved VRNN and VRNN applications

To complement this VRNN section, we report that an improved version of VRNN was presented by Goyal *et al.* (2017). The authors pointed out the difficulty in learning meaningful latent variables when coupled with a strong autoregressive decoder. We further discuss this point and we provide a series of references in Chapter 14. Goyal *et al.* (2017) proposed to improve the inference and training of VRNN with the following three features.

First, possibly inspired by Krishnan *et al.* (2017), as well as (Fraccaro *et al.*, 2016) (see Section 9.2), they introduced a backward RNN on $\mathbf{x}_t$ to feed the approximate posterior distribution $q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, in line with DKS (see Section 5.2). Therefore, they accounted for the future observations in the inference process, as opposed to the original VRNN, moving toward a better compliance with the structure of the exact posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$.

Second, they forced the latent variable $\mathbf{z}_t$ to contain relevant information about the future of the sequence by connecting $\mathbf{z}_t$ with the internal state of the inference backward network (denoted $\mathbf{b}_t$ by Goyal *et al.* (2017)). This was achieved by introducing an additional conditional model $p_\xi(\mathbf{b}_t|\mathbf{z}_t)$ and adding it in the VLB. Similarly, they also considered an additional conditional model $p_\xi(\mathbf{x}_t|\mathbf{b}_t)$.

Finally, they slightly modified the VRNN model itself by removing the direct link between $\mathbf{z}_t$ and $\mathbf{x}_t$; that is, they replaced (8.2)–(8.3) with (7.5)–(7.6), while all other equations remained identical to the VRNN equations. The authors reported that "[they] observed better performance by avoiding the latent variables from directly producing the next output."

An adaptation of VRNN to automatic language translation was proposed by Su *et al.* (2018). This is doubly interesting because this paper considers a sequence of discrete inputs and outputs, which contrasts with the "all continuous" models we focus on. This paper also contrasts with the previously proposed VAE-based models for text/language processing, which, as mentioned in the Introduction, usually consider a single latent vector to encode the whole input sequence (a full sentence). In Su *et al.*'s (2018) paper, it is the sequence of latent vectors $\mathbf{z}_{1:T}$ that encodes the semantic content of the sequence to translate "over time."

Finally, we can also mention the study by Lee *et al.* (2018), which uses VRNN for speech synthesis and adopts adversarial training. Besides VRNN, all these papers illustrate the flexibility of the DVAE class of models.

9

Stochastic Recurrent Neural Networks

The SRNN model was proposed by Fraccaro *et al.* (2016), with an objective to "glue (or stack) a deterministic recurrent neural network and a state space model together to form a stochastic and sequential neural generative model."

Notation remark: In Fraccaro *et al.*'s (2016) paper, $\mathbf{h}_t$ is denoted as $\mathbf{d}_t$, and the model is presented in the driven mode. We replace the external input $\mathbf{u}_t$ with $\mathbf{x}_{t-1}$ (i.e., predictive mode) for a better comparison with VRNN and STORN.

9.1 Generative model

The SRNN observation model is given by

$$\mathbf{h}_t = d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}), \tag{9.1}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t)] = d_{\mathbf{x}}(\mathbf{z}_t, \mathbf{h}_t), \tag{9.2}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{z}_t, \mathbf{h}_t)\}). \tag{9.3}$$

Eq. (9.1) is identical to (3.3) (with $\mathbf{x}_{t-1}$ instead of $\mathbf{u}_t$) and thus refers to the usual deterministic RNN. Eqs. (9.2) and (9.3) are quite similar

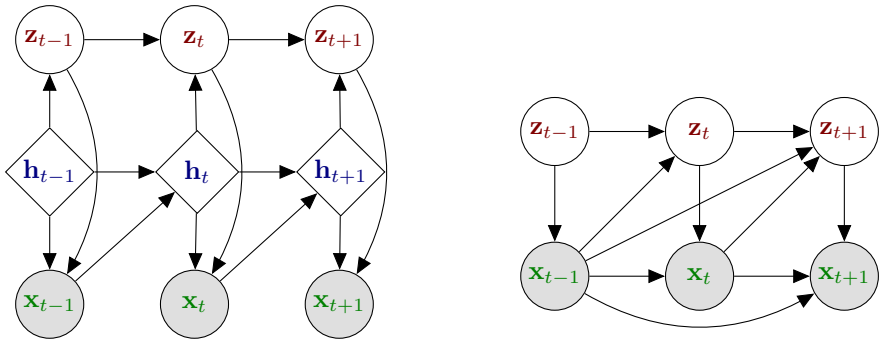


Figure 9.1: SRNN's graphical model in developed (left) and compact (right) forms.

to (8.2) and (8.3), respectively. Thus, the internal state $\mathbf{h}_t$ remains deterministic here, and the latent state $\mathbf{z}_t$ is integrated at the $d_{\mathbf{x}}$ level. This justifies the "clear(er) separation of deterministic and stochastic layers" claimed by Fraccaro *et al.* (2016), compared to VRNN.

Fraccaro *et al.* (2016) also introduced an explicit temporal model on the distribution of $\mathbf{z}_t$ (as opposed to implicit temporal dependency through $\mathbf{h}_t$ in VRNN), in addition to the dependency on the internal state $\mathbf{h}_t$:

$$[\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}, \mathbf{h}_t), \boldsymbol{\sigma}_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}, \mathbf{h}_t)] = d_{\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{h}_t), \tag{9.4}$$

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}, \mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}, \mathbf{h}_t)\}). \tag{9.5}$$

Compared to VRNN, the arrow from $\mathbf{z}_{t-1}$ to $\mathbf{h}_t$ is replaced with an arrow from $\mathbf{z}_{t-1}$ to $\mathbf{z}_t$, leading to an explicit first-order Markovian dependency for $\mathbf{z}_t$ (which is combined with the dependency on $\mathbf{h}_t$). In addition, compared to VRNN, no feature extractor is mentioned in SRNN, so we have here $\theta = \theta_{\mathbf{x}} \cup \theta_{\mathbf{z}}$ (and again $\theta_{\mathbf{x}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hx}}$ and $\theta_{\mathbf{z}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hz}}$). The graphical model of SRNN is shown in Figure 9.1 (left). In Fraccaro *et al.*'s (2016) paper, both $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$ are two-layer feed-forward networks. The function $d_{\mathbf{h}}$ is a GRU RNN, so that, according to the authors, "the SSM can therefore utilize long-term information captured by the RNN."

Using the same "unfolding the recurrence" trick as in the previous sections, we here denote $\mathbf{h}_t$ as $\mathbf{h}_t(\mathbf{x}_{1:t-1})$[1] and have $p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t) =$

[1] Again, we omit the initial term $\mathbf{h}_1$ for clarity of presentation.

$p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t(\mathbf{x}_{1:t-1}))$ and $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t) = p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t(\mathbf{x}_{1:t-1}))$. Again, if we follow the line of Appendix A, marginalizing the joint distribution of all variables w.r.t. $\mathbf{h}_{1:T}$ leads to

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t(\mathbf{x}_{1:t-1}))p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t(\mathbf{x}_{1:t-1})). \quad (9.6)$$

As for VRNN, we have a factorization over time frames, but no independence of $\mathbf{x}_t$ and $\mathbf{z}_t$ conditionally to the state $\mathbf{h}_t(\mathbf{x}_{1:t-1})$. As for STORN and VRNN, (9.6) can be reshaped into the following more general expression:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_t)p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:t-1}). \quad (9.7)$$

The corresponding compact graphical model is shown in Figure 9.1 (right).

9.2 Inference model

For SRNN, because of the dependencies in the generative model, the general form of the exact posterior distribution is given by

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T}). \quad (9.8)$$

At each time t, the posterior distribution of $\mathbf{z}_t$ depends on the previous latent state and the whole observation sequence.

Fraccaro *et al.* (2016) indicated this structure and proposed an approximate posterior distribution q_ϕ with the same structure. This is the second time this proper methodology is considered in the present review after the DKS in Section 5.2, but in the publication chronology, to the best of our knowledge, this was the first time. The dependency of q_ϕ on future observations, as well as on past observations through the future internal states, is implemented with a gated backward RNN. This network is denoted by $e_{\overleftarrow{\mathbf{g}}}$ in the equations below, it is parameterized by $\phi_{\overleftarrow{\mathbf{g}}}$, and $\overleftarrow{\mathbf{g}}_t$ denotes its internal state, where the right-to-left arrow highlights the backward nature of the process. This backward RNN is

followed by a basic feed-forward network $e_{\mathbf{z}}$, which is parameterized by $\phi_{\mathbf{z}}$. Formally, q_ϕ can be written as

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t), \tag{9.9}$$

with

$$\overleftarrow{\mathbf{g}}_t = e_{\overleftarrow{\mathbf{g}}}([\mathbf{h}_t, \mathbf{x}_t], \overleftarrow{\mathbf{g}}_{t+1}), \tag{9.10}$$

$$[\boldsymbol{\mu}_\phi(\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t), \boldsymbol{\sigma}_\phi(\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t)] = e_{\mathbf{z}}(\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t), \tag{9.11}$$

$$q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_\phi(\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t))\big). \tag{9.12}$$

We thus have here $\phi = \phi_{\overleftarrow{\mathbf{g}}} \cup \phi_{\mathbf{z}}$.

Notation remark: In Fraccaro *et al.*'s (2016) paper, $\mathbf{h}_t$ is denoted by $\mathbf{d}_t$, $\overleftarrow{\mathbf{g}}_t$ is denoted by $\mathbf{a}_t$, and $q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ is denoted by $q_\phi(\mathbf{z}_{1:T}|\mathbf{d}_{1:T}, \mathbf{x}_{1:T})$. Because we have $\mathbf{h}_t = \mathbf{h}_t(\mathbf{x}_{1:t-1})$, we can stick to $q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$.

The above equations show that inference requires a forward pass on the internal state $\mathbf{h}_t$ (which is shared by the encoder and decoder), its combination with $\mathbf{x}_t$, and a backward pass on the inference RNN, which makes $\overleftarrow{\mathbf{g}}_t$ a deterministic function of the whole data sequence $\mathbf{x}_{1:T}$. Similarly to $\mathbf{h}_t$, we can denote this function by $\overleftarrow{\mathbf{g}}_t(\mathbf{x}_{1:T})$ to make this latter point explicit; however, this would be poorly informative about the way $\overleftarrow{\mathbf{g}}_t$ depends on $\mathbf{x}_{1:T}$. The graphical model corresponding to the inference process in SRNN is shown in Figure 9.2. The inference model can be rewritten in the following general form:

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T}). \tag{9.13}$$

Fraccaro *et al.* (2016) stated that this *smoothing* process (combination of forward and backward RNNs on $\mathbf{x}_t$) can be replaced with a *filtering* process, by replacing (9.10)–(9.11) with an "instantaneous" DNN $e_{\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{h}_t, \mathbf{x}_t)$.

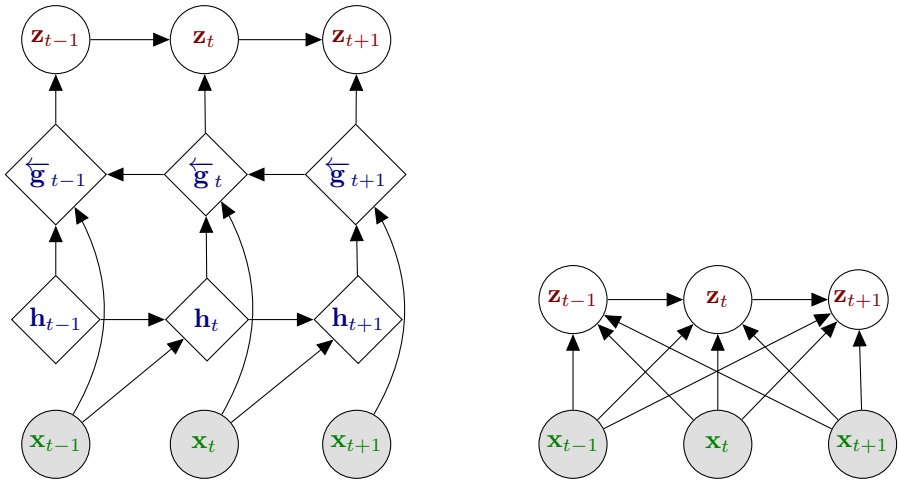


Figure 9.2: SRNN's graphical model at inference time in developed form (left) and compact form (right). In this case, there are no missing links on the proposed probabilistic dependencies (compared to the exact inference dependencies). In Fraccaro *et al.*'s (2016) paper, the dependencies of $\mathbf{h}_t$ were omitted in the inference graphical model for clarity. We make them explicit here to recall that $\mathbf{h}_t$ follows the deterministic update (9.1).

9.3 Training

A comparison of the compact form of SRNN in (9.7) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$) shows that the SRNN model makes the following conditional independence assumptions:

$$\begin{aligned} p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) &= p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_t); \\ p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) &= p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{t-1}). \end{aligned} \tag{9.14}$$

Using these two simplifications, along with the inference model (9.13) (which we recall is consistent with the exact posterior distribution), the

VLB in its most general form (4.24) can be simplified as follows:

$$\mathcal{L}(\theta, \phi; \mathbf{x}_{1:T}) = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T})} \left[\log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_t) \right] - \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{t-1}|\mathbf{x}_{1:T})} \left[D_{\mathrm{KL}} \left(q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T}) \parallel p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{t-1}) \right) \right]. \tag{9.15}$$

Again, the KL divergence can be computed analytically, and intractable expectations are approximated by Monte Carlo estimates. The procedure to sample from $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T})$ and $q_\phi(\mathbf{z}_{t-1}|\mathbf{x}_{1:T})$ relies on the "cascade trick," as for DKF (see Section 5.3).

10

Recurrent Variational Autoencoders

The RVAE model was introduced by Leglaive *et al.* (2020) to represent clean speech signals in a speech enhancement application. It was combined with a Gaussian noise model with nonnegative Matrix factorization of the variance within a Bayesian framework. The RVAE parameters were estimated offline on a large dataset of clean speech signals using the VAE methodology (maximization of the VLB). A variational expectation-maximization (VEM) algorithm was used for estimating the remaining parameters from a noisy speech signal, and probabilistic Wiener filters were then derived for speech enhancement. Here, we present only the RVAE model.

10.1 Generative model

The RVAE model was designed to model speech signals in the short-term Fourier transform (STFT) domain. This implies that the model applies to a sequence of *complex-valued* vectors. Therefore, the observation model uses a multivariate zero-mean circular complex Gaussian distribution (Neeser and Massey, 1993), denoted $\mathcal{N}_c$, instead of the usual multivariate real-valued Gaussian distribution. This observation model

has the following generic form:

$$\boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{z}_{\mathcal{T}}) = d_{\mathbf{x}}(\mathbf{z}_{\mathcal{T}}), \tag{10.1}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_{\mathcal{T}}) = \mathcal{N}_c(\mathbf{x}_t; \mathbf{0}, \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{z}_{\mathcal{T}})\}), \tag{10.2}$$

where $\mathcal{T}$ denotes a set of time frames, and the following three cases are considered: i) an instantaneous model: $\mathcal{T} = \{t\}$, which only considers the current latent state vector to model the observation at time t; ii) a causal model: $\mathcal{T} = \{1 : t\}$, which considers the sequence of past and present latent state vectors; and iii) a noncausal model: $\mathcal{T} = \{1 : T\}$, which considers the complete sequence of latent state vectors.

This model can be adapted to real-valued observations with a usual Gaussian distribution:[1] we just have to replace $\mathcal{N}_c$ with $\mathcal{N}$, replace $\mathbf{0}$ with a mean parameter $\boldsymbol{\mu}_\theta(\mathbf{z}_{\mathcal{T}})$ in (10.2), and add this mean parameter to the left-hand side of (10.1), as usual. This is what we have done hereinafter for easier comparison with the other models.

As in STORN, $\mathbf{z}_t$ is assumed i.i.d. with a standard Gaussian distribution:

$$p(\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_t) \qquad \text{with} \qquad p(\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_t; \mathbf{0}, \mathbf{I}_L). \tag{10.3}$$

Therefore, there is no explicit temporal model on $\mathbf{z}_t$, and $\mathbf{x}_t$ possibly depends on the past and future values of the latent state through (10.2). We have here $\theta_{\mathbf{z}} = \emptyset$ and $\theta = \theta_{\mathbf{x}}$. As case i) is strictly equivalent to the original VAE of Section 2, with no temporal model at all, we will now focus on cases ii) and iii).

Leglaive *et al.* (2020) only mentioned that cases ii) and iii) are implemented using a forward RNN and a bidirectional RNN, respectively, which take as input the sequence $\mathbf{z}_{1:t}$ or $\mathbf{z}_{1:T}$, respectively. The authors did not provide detailed implementation equations (though they provided a link to some supplementary material, including informative schemas). Let us write them now for easier comparison with the other models (for the same reason, we consider real-valued observations).

[1] Or any other distribution for real-valued vectors, as already mentioned. In fact, under some conditions, the complex proper Gaussian distribution applied on STFT coefficients corresponds to a Gamma distribution on the squared magnitude of those coefficients (Girin *et al.*, 2019).

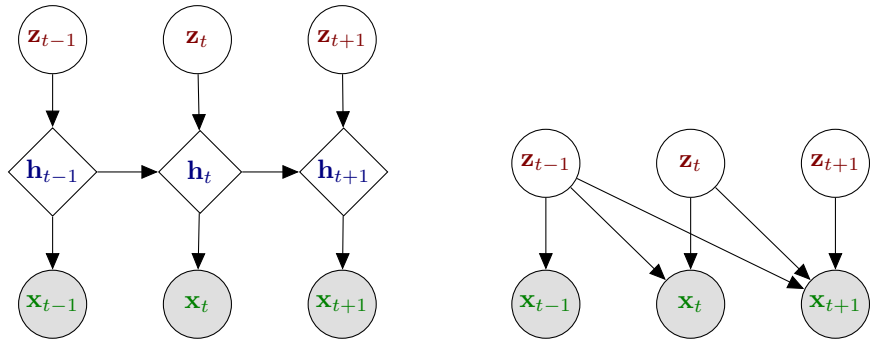


Figure 10.1: Causal RVAE's graphical model in developed form (left) and compact form (right).

Causal case: Let us start with the causal case, for which we have

$$\mathbf{h}_t = d_{\mathbf{h}}(\mathbf{z}_t, \mathbf{h}_{t-1}), \tag{10.4}$$

$$[\boldsymbol{\mu}_\theta(\mathbf{h}_t), \boldsymbol{\sigma}_\theta(\mathbf{h}_t)] = d_{\mathbf{x}}(\mathbf{h}_t), \tag{10.5}$$

$$p_\theta(\mathbf{x}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_\theta(\mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_\theta(\mathbf{h}_t)\}). \tag{10.6}$$

Eq. (10.4) is similar to the RNN internal state update (3.3) or (9.1), with the major difference being that the latent state $\mathbf{z}_t$ is used as an input instead of an external input $\mathbf{u}_t$ or previous observation vector $\mathbf{x}_{t-1}$. Alternatively, (10.4) can be viewed as a simplified version of the STORN or VRNN internal state updates (7.1) or (8.1), where only $\mathbf{z}_t$ and $\mathbf{h}_{t-1}$ (and not $\mathbf{x}_{t-1}$) are used as inputs. Considering both the observation model and the prior latent-state model, the causal RVAE model is quite close to STORN. The two differences with STORN are that here $\mathbf{x}_{t-1}$ is not reinjected as input to the internal state $\mathbf{h}_t$ and an LSTM network is used instead of a single-layer RNN in the original STORN formulation.

The graphical model of RVAE (causal case) is shown in Figure 10.1. As is now usual in our developments, we rewrite $\mathbf{h}_t = \mathbf{h}_t(\mathbf{z}_{1:t})$[2] and have $p_\theta(\mathbf{x}_t|\mathbf{h}_t) = p_\theta(\mathbf{x}_t|\mathbf{h}_t(\mathbf{z}_{1:t}))$. For a complete data sequence, we have

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta\big(\mathbf{x}_t|\mathbf{h}_t(\mathbf{z}_{1:t})\big)p(\mathbf{z}_t), \tag{10.7}$$

[2]This is also a function of $\mathbf{h}_0$, which we omit for clarity.

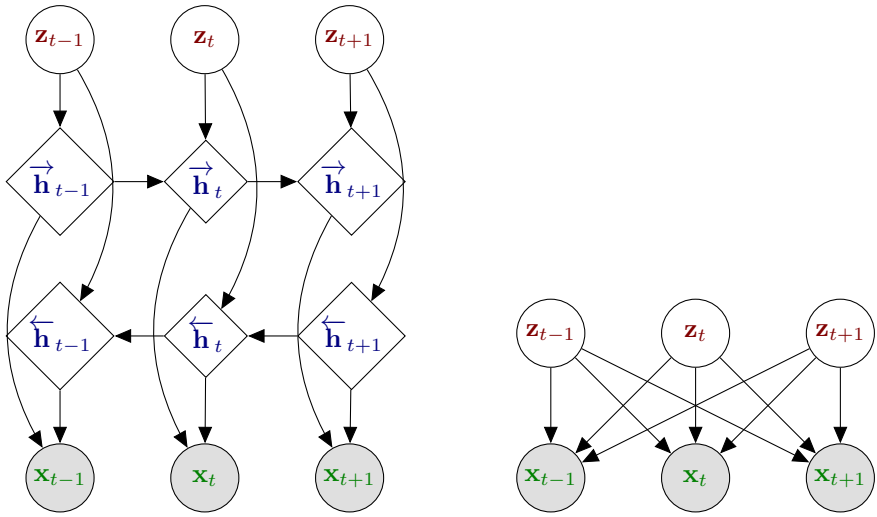


Figure 10.2: Noncausal RVAE's graphical model in developed form (left) and compact form (right).

which, as for the other models, can be reshaped into the following more general expression:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_{1:t})p(\mathbf{z}_t). \tag{10.8}$$

Noncausal case: All DVAE models we have seen so far are causal (at generation). The noncausal case presented by Leglaive *et al.* (2020) is the first noncausal DVAE model found in the literature. It is implemented with a combination of a forward RNN and a backward RNN on $\mathbf{z}_t$:

$$\overrightarrow{\mathbf{h}}_t = d_{\overrightarrow{\mathbf{h}}}(\mathbf{z}_t, \overrightarrow{\mathbf{h}}_{t-1}), \tag{10.9}$$

$$\overleftarrow{\mathbf{h}}_t = d_{\overleftarrow{\mathbf{h}}}(\mathbf{z}_t, \overleftarrow{\mathbf{h}}_{t+1}), \tag{10.10}$$

$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t], \tag{10.11}$$

$$[\boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{h}_t), \boldsymbol{\sigma}_{\theta_\mathbf{x}}(\mathbf{h}_t)] = d_\mathbf{x}(\mathbf{h}_t), \tag{10.12}$$

$$p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{x}}(\mathbf{h}_t)\}). \tag{10.13}$$

We thus have $\mathbf{h}_t = \mathbf{h}_t(\mathbf{z}_{1:T})$.[3] The graphical representation of the

[3]This function is also a function of the initial internal states $\overrightarrow{\mathbf{h}}_0$ and $\overleftarrow{\mathbf{h}}_{T+1}$,

noncausal RVAE model is shown in Figure 10.2. For a complete data sequence, we have

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t(\mathbf{z}_{1:T}))p(\mathbf{z}_t), \tag{10.14}$$

which can be reshaped into

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_{1:T})p(\mathbf{z}_t). \tag{10.15}$$

10.2 Inference model

As for the inference model, Leglaive *et al.* (2020) first remarked that, using the chain rule and D-separation, the posterior distribution of the latent vectors can be expressed as follows:

$$p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{\mathcal{T}'}), \tag{10.16}$$

where in the causal case, $\mathcal{T}' = \{t : T\}$, and in the noncausal case, $\mathcal{T}' = \{1 : T\}$. For the causal generative model, the latent vector at a given time step depends (a posteriori) on past latent vectors and on present and future observations, whereas for the noncausal generative model, it also depends on the past observations. Therefore, the authors chose to define the variational distribution q_ϕ with the same form:

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{\mathcal{T}'}). \tag{10.17}$$

As for the generative model, we now detail the implementation of the inference model.

Causal case: The inference corresponding to the causal generative model is implemented by combining a forward RNN on the latent

which we omit for clarity.

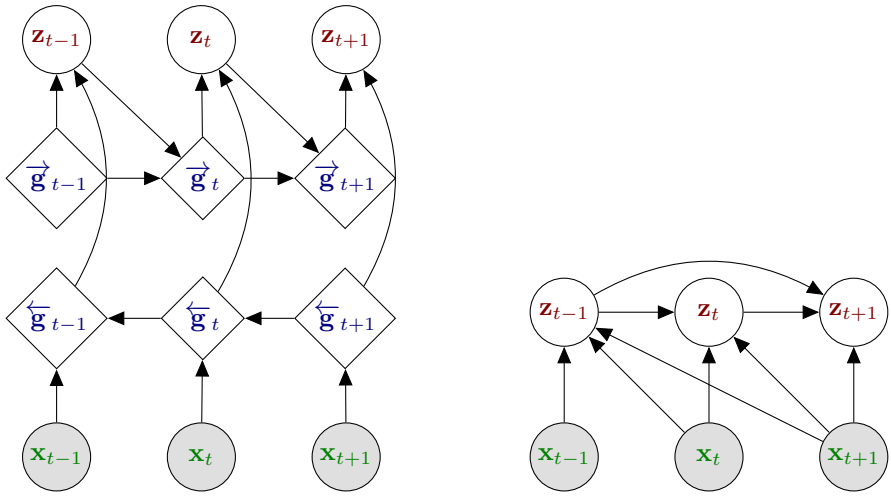


Figure 10.3: Graphical model of causal RVAE at inference time in developed form (left) and compact form (right).

vectors and a backward RNN on the observations:

$$\overrightarrow{\mathbf{g}}_t = e_{\overrightarrow{\mathbf{g}}}(\mathbf{z}_{t-1}, \overrightarrow{\mathbf{g}}_{t-1}), \tag{10.18}$$

$$\overleftarrow{\mathbf{g}}_t = e_{\overleftarrow{\mathbf{g}}}(\mathbf{x}_t, \overleftarrow{\mathbf{g}}_{t+1}), \tag{10.19}$$

$$\mathbf{g}_t = [\overrightarrow{\mathbf{g}}_t, \overleftarrow{\mathbf{g}}_t], \tag{10.20}$$

$$[\boldsymbol{\mu}_\phi(\mathbf{g}_t), \boldsymbol{\sigma}_\phi(\mathbf{g}_t)] = e_{\mathbf{z}}(\mathbf{g}_t), \tag{10.21}$$

$$q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{t:T}) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_\phi(\mathbf{g}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{g}_t)\}), \tag{10.22}$$

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{t:T}). \tag{10.23}$$

This inference model is shown in Figure 10.3.

Noncausal case: The inference corresponding to the noncausal generative model is similar to the causal case, except that the RNN on the

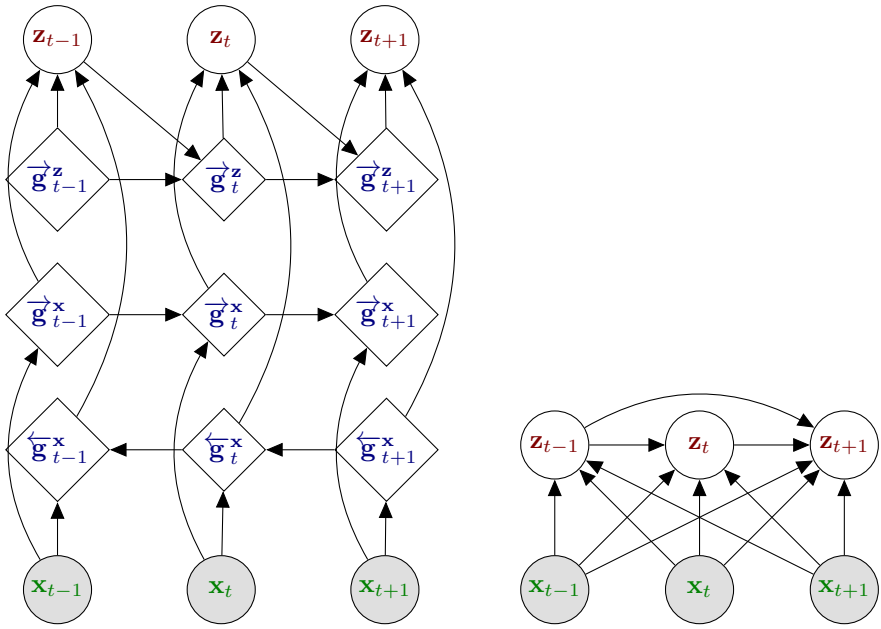


Figure 10.4: Graphical model of noncausal RVAE at inference time in developed form (left) and compact form (right).

observations is bidirectional:

$$\overrightarrow{\mathbf{g}}^{\mathbf{z}}_t = e_{\overrightarrow{\mathbf{g}}^{\mathbf{z}}}(\mathbf{z}_{t-1}, \overrightarrow{\mathbf{g}}^{\mathbf{z}}_{t-1}), \tag{10.24}$$

$$\overrightarrow{\mathbf{g}}^{\mathbf{x}}_t = e_{\overrightarrow{\mathbf{g}}^{\mathbf{x}}}(\mathbf{x}_t, \overrightarrow{\mathbf{g}}^{\mathbf{x}}_{t-1}), \tag{10.25}$$

$$\overleftarrow{\mathbf{g}}^{\mathbf{x}}_t = e_{\overleftarrow{\mathbf{g}}^{\mathbf{x}}}(\mathbf{x}_t, \overleftarrow{\mathbf{g}}^{\mathbf{x}}_{t+1}), \tag{10.26}$$

$$\mathbf{g}_t = [\overrightarrow{\mathbf{g}}^{\mathbf{z}}_t, \overrightarrow{\mathbf{g}}^{\mathbf{x}}_t, \overleftarrow{\mathbf{g}}^{\mathbf{x}}_t], \tag{10.27}$$

$$[\boldsymbol{\mu}_\phi(\mathbf{g}_t), \boldsymbol{\sigma}_\phi(\mathbf{g}_t)] = e_{\mathbf{z}}(\mathbf{g}_t), \tag{10.28}$$

$$q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_\phi(\mathbf{g}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{g}_t)\}), \tag{10.29}$$

$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}). \tag{10.30}$$

This inference model is shown in Figure 10.4.

10.3 Training

For conciseness, and because we focus on reviewing causal DVAEs, we only describe in this section the VLB for the causal RVAE model. The methodology to derive the VLB in the noncausal case is similar.

A comparison of the compact form of causal RVAE in (10.8) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$) shows that the causal RVAE model makes the following conditional independence assumptions:

$$\begin{aligned} p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) &= p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_{1:t}); \\ p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) &= p(\mathbf{z}_t). \end{aligned} \tag{10.31}$$

Using these two simplifications, along with the inference model (10.23) (which we recall is consistent with the exact posterior distribution), the VLB can be simplified as follows:

$$\begin{aligned} \mathcal{L}(\theta, \phi; \mathbf{x}_{1:T}) = &\sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T})} \left[\log p_{\theta_{\mathbf{x}}}(\mathbf{x}_t|\mathbf{z}_{1:t}) \right] \\ &- \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}_{1:t-1}|\mathbf{x}_{1:T})} \left[D_{\mathrm{KL}} \left(q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{t:T}) \parallel p(\mathbf{z}_t) \right) \right]. \end{aligned} \tag{10.32}$$

As for the previous models, the KL divergence can be computed analytically, while the two intractable expectations are approximated by Monte Carlo estimates using samples drawn from $q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{1:T})$ and $q_\phi(\mathbf{z}_{1:t-1}|\mathbf{x}_{1:T})$ in a recursive manner.

11

Disentangled Sequential Autoencoders

Li and Mandt (2018) proposed a hierarchical model called DSAE. This model introduces the idea of adding to the usual sequence of latent variables $\mathbf{z}_{1:T}$ a sequence-level latent vector $\mathbf{v}$ (denoted $\mathbf{f}$ by Li and Mandt (2018)), which is assumed to encode the sequence-level characteristics of the data. Therefore, $\mathbf{z}_t$ is assumed to encode time-dependent data features (e.g., the dynamics of an object in a video clip), and $\mathbf{v}$ is assumed to encode "everything else" (e.g., object characteristics in a video clip).

11.1 Generative model

Li and Mandt (2018) only provided the following general form of the generative DSAE model for a complete data sequence:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{v}) = p_{\theta_\mathbf{v}}(\mathbf{v}) \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{v}) p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}). \tag{11.1}$$

More detailed information about the pdfs and implementation issues are found in annexes from the ArXiv version of the paper. The authors used different variants for different datasets according to the nature of the data (e.g., video clips or speech signals). Here, we only report the model

implemented for speech signals. The dynamical model $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1})$ is a Gaussian distribution whose parameters are provided by an LSTM network. The observation model $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{v})$ is a Gaussian distribution whose parameters are provided by a feed-forward DNN. With the simplified generic RNN formalism used for LSTM (see Section 3.1.1), we can thus write

$$\mathbf{h}_t = d_\mathbf{h}(\mathbf{z}_{t-1}, \mathbf{h}_{t-1}), \tag{11.2}$$

$$[\boldsymbol{\mu}_{\theta_\mathbf{z}}(\mathbf{h}_t), \boldsymbol{\sigma}_{\theta_\mathbf{z}}(\mathbf{h}_t)] = d_\mathbf{z}(\mathbf{h}_t), \tag{11.3}$$

$$p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{h}_t) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_\mathbf{z}}(\mathbf{h}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{z}}(\mathbf{h}_t)\}), \tag{11.4}$$

$$[\boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{z}_t, \mathbf{v}), \boldsymbol{\sigma}_{\theta_\mathbf{x}}(\mathbf{z}_t, \mathbf{v})] = d_\mathbf{x}(\mathbf{z}_t, \mathbf{v}), \tag{11.5}$$

$$p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{v}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{z}_t, \mathbf{v}), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_\mathbf{x}}(\mathbf{z}_t, \mathbf{v})\}). \tag{11.6}$$

The graphical representation of DSAE is shown in Figure 11.1. This model is similar to a DKF in the undriven mode conditioned on variable $\mathbf{v}$, except that, in addition to this conditioning, the first-order Markov temporal model of DKF is replaced with a virtually infinite-order model owing to the LSTM. Although this is poorly discussed in the DVAE papers in general, it is an example of interesting model extensions that are easy to implement in the deep learning and VAE frameworks. As stated by Krishnan *et al.* (2015), "using deep neural networks, we can enhance Kalman filters with arbitrarily complex transition dynamics and emission distributions. [...] we can tractably learn such models by optimizing a bound on the likelihood of the data."

11.2 Inference model

For DSAE, the posterior distribution of latent variables is given by

$$p_\theta(\mathbf{z}_{1:T}, \mathbf{v}|\mathbf{x}_{1:T}) = p_\theta(\mathbf{v}|\mathbf{x}_{1:T}) p_\theta(\mathbf{z}_{1:T}|\mathbf{v}, \mathbf{x}_{1:T}) \tag{11.7}$$

$$= p_\theta(\mathbf{v}|\mathbf{x}_{1:T}) \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{v}, \mathbf{x}_{1:T}) \tag{11.8}$$

$$= p_\theta(\mathbf{v}|\mathbf{x}_{1:T}) \prod_{t=1}^{T} p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{v}, \mathbf{x}_{t:T}). \tag{11.9}$$

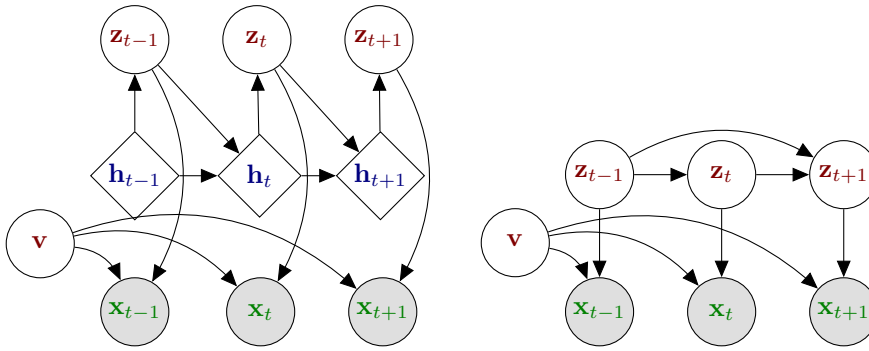


Figure 11.1: DSAE's graphical model in developed form (left) and compact form (right).

The simplification in the last line results from D-separation. This decomposition can be interpreted as follows: The whole sequence of observations $\mathbf{x}_{1:T}$ is used to estimate the "object" representation $\mathbf{v}$, and then $\mathbf{v}$, the present and future observations $\mathbf{x}_{t:T}$, and previous latent state vectors $\mathbf{z}_{1:t-1}$ are used to update the object dynamics.

As for the approximate posterior distribution q_ϕ, Li and Mandt (2018) proposed two models. The first one, referred to as "factorized," is expressed as

$$q_\phi(\mathbf{z}_{1:T}, \mathbf{v}|\mathbf{x}_{1:T}) = q_{\phi_\mathbf{v}}(\mathbf{v}|\mathbf{x}_{1:T}) \prod_{t=1}^{T} q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_t). \tag{11.10}$$

This model thus relies on an instantaneous frame-wise inference model $q_\phi(\mathbf{z}_t|\mathbf{x}_t)$ for encoding the latent vector dynamics. This approach is oversimplistic compared to the exact posterior distribution and yields inferior performance to that of the second inference model. We thus focus on the latter, which is referred to as "full," and is given by

$$q_\phi(\mathbf{z}_{1:T}, \mathbf{v}|\mathbf{x}_{1:T}) = q_{\phi_\mathbf{v}}(\mathbf{v}|\mathbf{x}_{1:T}) q_{\phi_\mathbf{z}}(\mathbf{z}_{1:T}|\mathbf{v}, \mathbf{x}_{1:T}). \tag{11.11}$$

As the authors mentioned, "The idea behind [this] structured approximation is that content may affect dynamics." So far, this model has been compliant with the exact posterior distribution, as expressed by (11.7). From the information given in the annexes of the ArXiv version of the paper, we can write the detailed equations of the full inference

model as follows:

$$\overrightarrow{\mathbf{g}}^{\mathbf{v}}_t = e_{\overrightarrow{\mathbf{g}}^{\mathbf{v}}}(\mathbf{x}_t, \overrightarrow{\mathbf{g}}^{\mathbf{v}}_{t-1}), \tag{11.12}$$

$$\overleftarrow{\mathbf{g}}^{\mathbf{v}}_t = e_{\overleftarrow{\mathbf{g}}^{\mathbf{v}}}(\mathbf{x}_t, \overleftarrow{\mathbf{g}}^{\mathbf{v}}_{t+1}), \tag{11.13}$$

$$\mathbf{g}^{\mathbf{v}} = [\overrightarrow{\mathbf{g}}^{\mathbf{v}}_T, \overleftarrow{\mathbf{g}}^{\mathbf{v}}_1], \tag{11.14}$$

$$[\boldsymbol{\mu}_{\phi_{\mathbf{v}}}(\mathbf{g}^{\mathbf{v}}), \boldsymbol{\sigma}_{\phi_{\mathbf{v}}}(\mathbf{g}^{\mathbf{v}})] = e_{\mathbf{v}}(\mathbf{g}^{\mathbf{v}}), \tag{11.15}$$

$$q_{\phi_{\mathbf{v}}}(\mathbf{v}|\mathbf{x}_{1:T}) = \mathcal{N}(\mathbf{v}; \boldsymbol{\mu}_{\phi_{\mathbf{v}}}(\mathbf{g}^{\mathbf{v}}), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\phi_{\mathbf{v}}}(\mathbf{g}^{\mathbf{v}})\}), \tag{11.16}$$

$$\overrightarrow{\mathbf{g}}^{\mathbf{z}}_t = e_{\overrightarrow{\mathbf{g}}^{\mathbf{z}}}([\mathbf{x}_t, \mathbf{v}], \overrightarrow{\mathbf{g}}^{\mathbf{z}}_{t-1}), \tag{11.17}$$

$$\overleftarrow{\mathbf{g}}^{\mathbf{z}}_t = e_{\overleftarrow{\mathbf{g}}^{\mathbf{z}}}([\mathbf{x}_t, \mathbf{v}], \overleftarrow{\mathbf{g}}^{\mathbf{z}}_{t+1}), \tag{11.18}$$

$$\mathbf{g}^{\mathbf{z}}_t = [\overrightarrow{\mathbf{g}}^{\mathbf{z}}_t, \overleftarrow{\mathbf{g}}^{\mathbf{z}}_t], \tag{11.19}$$

$$[\boldsymbol{\mu}_{\phi_{\mathbf{z}}}(\mathbf{g}^{\mathbf{z}}_t), \boldsymbol{\sigma}_{\phi_{\mathbf{z}}}(\mathbf{g}^{\mathbf{z}}_t)] = e_{\mathbf{z}}(\mathbf{g}^{\mathbf{z}}_t), \tag{11.20}$$

$$q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{v}, \mathbf{x}_{1:T}) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\phi_{\mathbf{z}}}(\mathbf{g}^{\mathbf{z}}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\phi_{\mathbf{z}}}(\mathbf{g}^{\mathbf{z}}_t)\}), \tag{11.21}$$

and for the full sequence $\mathbf{z}_{1:T}$ we have

$$q_{\phi_{\mathbf{z}}}(\mathbf{z}_{1:T}|\mathbf{v}, \mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{v}, \mathbf{x}_{1:T}). \tag{11.22}$$

Note that none of the two approximations proposed by the authors (factorized and full) actually follow the dependencies of the exact posterior distribution shown in (11.9). The graphical representation of the full inference model is shown in Figure 11.2.[1]

11.3 Training

To derive the VLB for the DSAE model, we apply the same strategy as that applied for the other models (i.e., inject the generative model and the approximate posterior in the VLB general formulation). We do so

[1] In Li and Mandt's (2018) paper, Appendix A, the schematic representation of the inference model given in Figure 9(b) is not consistent with the following sentence (reported with our notations): "Finally the parameters of $q_\phi(\mathbf{z}_{1:T}|\mathbf{v}, \mathbf{x}_{1:T})$ are computed by a simple RNN with input $[\overrightarrow{\mathbf{g}}^{\mathbf{z}}_t, \overleftarrow{\mathbf{g}}^{\mathbf{z}}_t]$ at time t." It is indeed inconsistent and a bit odd that $\mathbf{z}_{t-1}$ is not mentioned as an input of the $\mathbf{z}_t$ inference process, as is well apparent in Figure 9(b). We base the writing of (11.19)–(11.21) on their text and not on their figure.

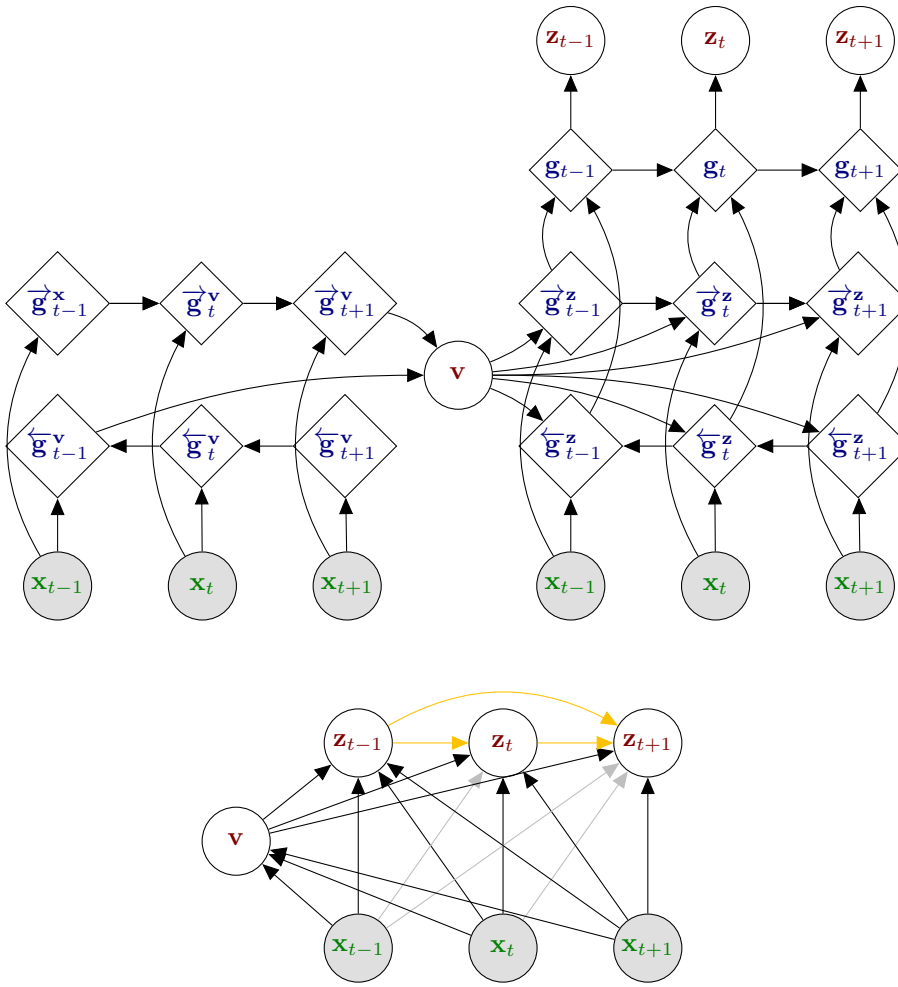


Figure 11.2: Graphical model of DSAE at inference time in developed form (top) and compact form (bottom). In addition to the missing arrows shown in gold, we display in silver the arrows that should not be used, as compared to the structure of the exact posterior distribution. To the best of our knowledge, DSAE is the only model that uses probabilistic dependencies that do not appear in the exact posterior distribution.

for the full inference model (11.11) and obtain

$$\begin{aligned}\mathcal{L}(\theta, \phi, \mathbf{x}_{1:T}) &= \mathbb{E}_{q_\phi(\mathbf{v}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\Big[\log p_{\theta_\mathbf{x}}(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}, \mathbf{v})\Big] \\ &\quad - D_{KL}\Big(q_\phi(\mathbf{v}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T}) \parallel p_\theta(\mathbf{v}, \mathbf{z}_{1:T})\Big) \qquad (11.23) \\ &= \mathbb{E}_{q_{\phi_\mathbf{v}}(\mathbf{v}|\mathbf{x}_{1:T})}\Bigg[\sum_{t=1}^{T} \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{v}, \mathbf{x}_{1:T})}\Big[\log p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{v})\Big] \\ &\quad - \sum_{t=1}^{T} \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}_{1:t-1}|\mathbf{v}, \mathbf{x}_{1:T})}\Big[D_{KL}(q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{v}, \mathbf{x}_{1:T}) \parallel p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}))\Big]\Bigg] \\ &\quad - D_{KL}(q_{\phi_\mathbf{v}}(\mathbf{v}|\mathbf{x}_{1:T}) \parallel p_{\theta_\mathbf{v}}(\mathbf{v})). \qquad (11.24)\end{aligned}$$

Therefore, one must first compute $q_{\phi_\mathbf{v}}(\mathbf{v}|\mathbf{x}_{1:T})$ to then sample from it. Once this is achieved, the parameters of $q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{v}, \mathbf{x}_{1:T})$ for all time-steps t can be computed without sampling from any random variable. Once this is achieved, the samples of $\mathbf{z}_{1:t-1}$ are used to compute the t-th KL divergence term in (11.24).

12

Brief tour of other models

In this section, we briefly present a few other models that have been recently proposed in the literature and can be considered members of the DVAE family. We choose not to present them in a detailed manner, as in the previous sections, because they are either too far from the scope of the review, which focuses on models associating a sequence of observations with a sequence of latent variables, or too close to the already presented models.

12.1 Models related to DKF

Latent LDS and structured VAE: Johnson *et al.* (2016) considered several models. One of them is a simplified DKF in which the latent variable model (i.e., the dynamical model) is linear-Gaussian; that is, it follows (3.16) (with $\mathbf{u}_t$ following a standard Gaussian distribution), while the observation model is a DNN-based nonlinear model similar to the DKF observation model. They called it a latent LDS and extended it to a latent switching LDS model based on a bank of dynamical models (actually the same latent LDS but with different parameters) and an additional *discrete* latent variable that controls the switch between the dynamical models over time to adjust to the observed data dynamics.

This model can be considered an extension of the switching Kalman filter (Murphy, 1998; Fox *et al.*, 2011) to a DNN-based observation model (see also Linderman *et al.* (2016) for a similar combination). Johnson *et al.* (2016) did not provide detailed equations for these models. Rather, they showed how the use of structured mean-field approximation in the inference model, combined with the use of an observation model that is conjugate to the latent variable model, can make the inference and training processes particularly efficient. They called the resulting model a structured VAE (SVAE). As they presented these developments in the general framework of probabilistic graphical models (Koller and Friedman, 2009), which is more general than the DVAE framework, they did not provide "temporal equations." This makes this paper somewhat poorly connected to our review, although they mentioned that the DKF model (Krishnan *et al.*, 2015) is strongly related to their work (they also implied that using RNN models for implementing time dependencies is restrictive in the general framework that they present).

Black-box deep SSM: Similarly, Archer *et al.* (2015) also focused on the structure of the approximate posterior distribution to improve the computational efficiency of the inference. They proposed using a multivariate Gaussian approximate posterior with a block tridiagonal inverse covariance matrix. They also proposed a corresponding fast and scalable inference algorithm. This general approach can be applied with different (deep and nondeep) parameterizations of the inference model and is applicable to a large family of SSMs, hence the "black box" denomination in the paper title. The authors mainly focused and experimented on an LG-LDS (to show that their algorithm can efficiently recover the solution of the Kalman filter), an LDS with a linear-Poisson observation model (which has no closed-form inference solution), and a basic one-dimensional nonlinear LDS. This study is well-connected with the DKF model and with deep SSMs in general. Interestingly, in this study, which, again, focuses more on the inference model than on the generative model, only the inference model is deep, whereas the generative models used in the experiments are nondeep. This study was later extended, notably addressing online learning and real-time issues (Zhao and Park, 2017; Zhao *et al.*, 2019).

Deep variational Bayesian filters: We have already mentioned this class of models in Section 6.1. DVBFs, which were proposed by (Karl *et al.*, 2017), are an extension of the class of SSM-based DVAE models with dynamical models that depend on stochastic parameters. For example, the transition model at time t (i.e., between $\mathbf{z}_t$ and $\mathbf{z}_{t+1}$) can be a linear-Gaussian model with matrices and vectors that are a weighted sum of matrices/vectors randomly selected in a predefined set (possibly learned from data) with weights that are provided by a DNN. A similar transition model was applied within the KVAE model in Fraccaro *et al.*'s (2017) paper (see also Watter *et al.* (2015)).

Disentangled SSM: Miladinović *et al.* (2019) recently proposed a model called the disentangled state-space model (DSSM), in line with the DSAE model and, more generally, with models that attempt to separate the encoding of the content/object at the sequence level from that of its dynamics at the time-frame level. However, in contrast to DSAE, in which the sequence-level latent variable conditions the observation model, in DSSM, this sequence-level variable conditions the dynamical model. It is assumed to model the fact that the dynamics of an object are dependent on the considered applicative domain (e.g., enzyme kinetics or bouncing ball kinematics). In other words, (11.1) for DSAE can be reshaped in DSSM as[1]

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}, \mathbf{v}) = p_{\theta_\mathbf{v}}(\mathbf{v}) p_{\theta_\mathbf{z}}(\mathbf{z}_0) \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t) p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{v}). \quad (12.1)$$

Here, the dynamical model is a (conditioned) first-order model. Miladinović *et al.* (2019) also proposed a filtering inference model that is implemented in the more general DVBF framework mentioned above.

12.2 Models related to STORN, VRNN, and SRNN

Variational recurrent autoencoder (VRAE): The VRAE model, proposed by Fabius and Amersfoort (2014), can been considered a simplified version of STORN, from which, according to the authors

[1] In Miladinović *et al.*'s (2019) paper, the sequence-level variable is denoted as D for "domain." For consistency, we retain the notation $\mathbf{v}$ used in DSAE.

themselves, it took inspiration. Here, the data sequence $\mathbf{x}_{1:T}$ is encoded by a single latent random vector $\mathbf{z}$, instead of the sequence $\mathbf{z}_{1:T}$. The compact form of the generative model is given by

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}) = p_{\theta_\mathbf{z}}(\mathbf{z}) \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}), \tag{12.2}$$

Fabius and Amersfoort (2014) provided no information about $p_{\theta_\mathbf{z}}(\mathbf{z})$. The generative distribution $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z})$ is implemented with a forward RNN that uses $\mathbf{z}$ to calculate the first hidden state $\mathbf{h}_1$ and then iteratively takes $\mathbf{x}_{t-1}$ as input to calculate $\mathbf{h}_t$, which provides the parameters of the distribution of $\mathbf{x}_t$. Conversely, the inference model $q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{x}_{1:T})$ is based on a forward RNN that takes $\mathbf{x}_{1:T}$ as input and delivers a final internal state $\mathbf{g}_T$, from which we obtain the distribution parameters for $\mathbf{z}$. In short, the VRAE generative model can be described by Figure 7.1, where the sequence $\mathbf{z}_{1:T}$ is replaced with a single input $\mathbf{z}$ for $\mathbf{h}_1$, and the VRAE inference model can be described by Figure 7.2, where the sequence $\mathbf{z}_{1:T}$ is replaced with a single output $\mathbf{z}$ for $\mathbf{g}_T$. We thus have a sequence-to-one encoding and a one-to-sequence decoding, which evoke the models designed for text/language processing mentioned in the Introduction. As Fabius and Amersfoort's (2014) paper was published in 2014 and was part of the early papers on DVAEs, it was probably inspiring for the natural language processing (NLP) community.

A similar model was proposed by Babaeizadeh *et al.* (2018), with a difference being that several vectors $\mathbf{x}_{t:T}$ are predicted from the past context $\mathbf{x}_{1:t-1}$ and from the unique latent vector $\mathbf{z}$. The inference model is also of the form $q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{x}_{1:T})$, as in Fabius and Amersfoort's (2014) study. Babaeizadeh *et al.* (2018) compared this model with a "baseline" model in which the latent vector is defined on a frame-by-frame basis (i.e., $\mathbf{z}_t$). The observation model of this baseline model is similar to that of SRNN, and the prior over $\mathbf{z}_t$ is an i.i.d. standard Gaussian distribution.

Factorized hierarchical variational autoencoder (FHVAE): The FHVAE model was proposed by Hsu *et al.* (2017b), to learn disentangled and interpretable latent representations from sequential data without supervision. To achieve this aim, FHVAE explicitly models the multi-

scaled aspect of the temporal information contained in the data. This is done by splitting each sequence of data vectors into a set of fixed-size consecutive sub-sequences, called segments, and defining two latent variables $\mathbf{z}$ and $\mathbf{v}$ at the segment level.[2] The former is dedicated to capturing data information at the segment level, whereas the latter is dedicated to capturing data information across segments (i.e., at the sequence level). This model is particularly appropriate for speech signals: In this case, 200-ms segments represent the approximate duration of a syllable, and thus, $\mathbf{z}$ would typically encode phonetic information, whereas $\mathbf{v}$ would typically encode speaker information at the level of a complete utterance. In essence, FHVAE is strongly related to DSAE, which also contains a sequence-level latent variable $\mathbf{v}$ but preserves a time-frame resolution for the dynamical latent variable $\mathbf{z}_t$ (see Section 11). In fact, DSAE was published after FHVAE, from which it was probably inspired.

Even if we do not detail this model, we report a few equations to help better understand how segmental modeling works. Let here $t \in [1, T]$ denote the index of a vector within a segment (each segment has T vectors), and let $n \in [1, N]$ denote the index of a segment within a sequence. The FHVAE observation model *for each individual segment of data* is given by

$$\mathbf{h}_t^{(n)} = d_{\mathbf{h}}(\mathbf{z}^{(n)}, \mathbf{v}^{(n)}, \mathbf{h}_{t-1}^{(n)}), \tag{12.3}$$

$$[\boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t^{(n)}), \boldsymbol{\sigma}_{\theta_{\mathbf{x}}}(\mathbf{h}_t^{(n)})] = d_{\mathbf{x}}(\mathbf{h}_t^{(n)}), \tag{12.4}$$

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_t^{(n)}|\mathbf{h}_t^{(n)}) = \mathcal{N}(\mathbf{x}_t^{(n)}; \boldsymbol{\mu}_{\theta_{\mathbf{x}}}(\mathbf{h}_t^{(n)}), \mathrm{diag}\{\boldsymbol{\sigma}^2_{\theta_{\mathbf{x}}}(\mathbf{h}_t^{(n)})\}). \tag{12.5}$$

In (12.3), $\mathbf{z}^{(n)}$ and $\mathbf{v}^{(n)}$ respectively denote the latent vectors $\mathbf{z}$ and $\mathbf{v}$ for the considered n-th segment. There is a single pair of such vectors for each segment, and hence, a many-to-one encoding and one-to-many decoding at the segment level. In practice, these equations are implemented with an LSTM network. The prior distribution of $\mathbf{z}^{(n)}$, $p_{\theta_{\mathbf{z}}}(\mathbf{z}^{(n)})$, is a centered isotropic Gaussian that is independent of both the segment and the sequence. In contrast, the prior distribution of $\mathbf{v}^{(n)}$ depends on a latent variable $\mathbf{w}$, which is defined at the sequence level and whose

[2] In Hsu *et al.*'s (2017) paper, $\mathbf{z}$ and $\mathbf{v}$ are denoted $\mathbf{z}_1$ and $\mathbf{z}_2$, respectively. We changed the notation to avoid confusion between the variable index and time index.

prior distribution $p_{\theta_{\mathbf{w}}}(\mathbf{w})$ is also a centered isotropic Gaussian. The distribution of $\mathbf{v}^{(n)}$ is then given by $p_{\theta_{\mathbf{v}}}(\mathbf{v}^{(n)}|\mathbf{w}) = \mathcal{N}(\mathbf{v}^{(n)}; \mathbf{w}, \sigma^2_{\theta_{\mathbf{v}}}\mathbf{I}_{L_v})$. For a given sequence, $p_{\theta_{\mathbf{v}}}(\mathbf{v}^{(n)}|\mathbf{w})$ depends on the value of $\mathbf{w}$ drawn for that particular sequence. In practice, all generated $\mathbf{v}^{(n)}$ vectors within a sequence are close to $\mathbf{w}$. This makes $\mathbf{v}^{(n)}$ a sequence-dependent latent factor, whereas $\mathbf{z}^{(n)}$ behaves as a segment-dependent and sequence-independent latent factor. The joint density of a sequence is given by

$$p_\theta(\mathbf{x}_{1:T}^{(1:N)}, \mathbf{z}^{(1:N)}, \mathbf{v}^{(1:N)}, \mathbf{w}) = p_{\theta_{\mathbf{w}}}(\mathbf{w}) \prod_{n=1}^{N} \prod_{t=1}^{T} p_{\theta_{\mathbf{x}}}(\mathbf{x}_t^{(n)}|\mathbf{h}_t(\mathbf{z}^{(n)}, \mathbf{v}^{(n)})) \\ p_{\theta_{\mathbf{z}}}(\mathbf{z}^{(n)}) p_{\theta_{\mathbf{v}}}(\mathbf{v}^{(n)}|\mathbf{w}), \quad (12.6)$$

where, as in the previous sections, $\mathbf{h}_t(\mathbf{z}^{(n)}, \mathbf{v}^{(n)})$ is a shortcut for the function that results from unfolding the recurrence in (12.3).

The inference model is a many-to-one encoder that works at the segment level: each segment $\mathbf{x}_{1:T}^{(n)}$ is encoded into a pair $\{\mathbf{z}^{(n)}, \mathbf{v}^{(n)}\}$ (plus an estimate of $\mathbf{w}$ for each whole sequence). As for the variational approximate posterior q_ϕ, Hsu *et al.* (2017b) proposed the following model:

$$q_\phi(\mathbf{z}^{(1:N)}, \mathbf{v}^{(1:N)}, \mathbf{w}|\mathbf{x}_{1:T}^{(1:N)}) = q_{\phi_{\mathbf{w}}}(\mathbf{w}) \prod_{n=1}^{N} q_{\phi_{\mathbf{z}}}(\mathbf{z}^{(n)}|\mathbf{x}_{1:T}^{(n)}, \mathbf{v}^{(n)}) q_{\phi_{\mathbf{v}}}(\mathbf{v}^{(n)}|\mathbf{x}_{1:T}^{(n)}), \quad (12.7)$$

where $q_{\phi_{\mathbf{z}}}$ and $q_{\phi_{\mathbf{v}}}$ are both implemented with a forward LSTM network, whose last state vector is passed to a DNN to provide the distribution parameters. Two encoders are chained here: The first one is used to generate $\mathbf{v}^{(n)}$ (by sampling $q_{\phi_{\mathbf{v}}}(\mathbf{v}^{(n)}|\mathbf{x}_{1:T}^{(n)})$), and then, $\mathbf{v}^{(n)}$ is injected in the second encoder to generate $\mathbf{z}^{(n)}$. As for $q_{\phi_{\mathbf{w}}}(\mathbf{w})$, it is a Gaussian distribution whose mean vector is obtained from a look-up table that is jointly learned with the model parameters (see Hsu *et al.* (2017b) for details). Cascading the sequence-to-one encoder with the one-to-sequence decoder results in a sequence-to-sequence neural network architecture that is trained by maximizing the VLB (not detailed here). The model can be optimized at the segment level instead of the sequence level; that is, each data segment can be used as a batch dataset.

According to Hsu *et al.* (2017b), this can solve scalability issues when the training sequences become too long.

Deep recurrent attentive writer (DRAW): A somewhat dual model of VRAE was proposed by Gregor *et al.* (2015) and called DRAW. This model considers a sequence of latent vectors $\mathbf{z}_{1:T}$ to encode a single static but highly structured data $\mathbf{x}$ (typically a low-resolution image). The generative model is of the general form $p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z}_{1:T})$. It involves the iterative construction of a sequence $\hat{\mathbf{x}}_{1:T}$ that can be considered the sequence of images resulting from the "natural" drawing of $\mathbf{x}$ over time. The dependency of $\mathbf{x}$ on $\mathbf{z}_{1:T}$ is implemented by combining the output of a decoder RNN (which takes $\mathbf{z}_t$ as the input) and a so-called canvas matrix $\mathbf{c}_{t-1}$, which encodes the difference between the final target image $\mathbf{x}$ and the current draw $\hat{\mathbf{x}}_{t-1}$. Hence, the model combines deep learning and some form of predictive coding (Gersho and Gray, 2012). The inference model is of the form $q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x})$ and is implemented with an encoder RNN. This network takes as inputs a combination of $\mathbf{x}$, $\hat{\mathbf{x}}_{t-1}$, and the decoder output at the previous time step; hence, the predictive coding is implemented in closed-loop mode (Gersho and Gray, 2012). As the name indicates, DRAW includes a selective attention model that enables it to focus on the most relevant parts of the observation. The description of such an attention model is beyond the scope of the present review (see Gregor *et al.* (2015) and references therein for details).

Neural adaptive sequential Monte Carlo (NASMC): Gu *et al.* (2015) proposed the NAMSC generative model, which combines an infinite-order Markovian model on $\mathbf{z}_t$ (as that used in DSAE) with the most general observation model, which is that used in STORN and VRNN. In one variant of this model, they replaced the Markovian model on $\mathbf{z}_t$ with an i.i.d. model, and thus, this variant is similar to STORN. The inference model has the same general form as that of VRNN; that is, it follows (8.14) and is parametrized by an RNN. The originality of NASMC lies in connecting the DVAE inference framework with sequential Monte Carlo (SMC) sampling. The inference model was used as a proposal distribution for SMC sampling. In fact, a complete framework to learn the parameters of the generative model, the proposal model, and for sampling from the posterior distribution with SMC

was proposed. In other words, Gu *et al.* (2015) showed that their sampling-based approach can be used to optimize the observed data marginal likelihood for estimating the generative model parameters in the variational framework. For other applications of SMC methods in the general context of variational approximations, see (Maddison *et al.*, 2017; Le *et al.*, 2018; Naesseth *et al.*, 2018).

Recurrent state-space model (RSSM): Hafner *et al.* (2018) used the RSSM model, which is very similar, if not identical, to the VRNN model used in the driven mode. In fact, RSSM corresponds to VRNN with an external input $\mathbf{u}_t$ (denoted as $\mathbf{a}_{t-1}$ by Hafner *et al.* (2018)), which is used in place of $\mathbf{x}_{t-1}$ to compute $\mathbf{h}_t$. This is how VRNN was presented in the "Related Works" section of the SRNN paper (Fraccaro *et al.*, 2016) (see in particular Figure 4(b) of this latter paper). Hafner *et al.* (2018) used this RSSM model for learning the dynamics and planning the actions of a synthetic agent from image sequences in a reinforcement learning framework. They also presented a way to perform multi-step prediction (i.e., prediction several steps ahead).

Stochastic video generation (SVG): Denton and Fergus (2018) presented a model very similar to STORN and applied it to stochastic video generation and multiple-frame video prediction, similarly to (Babaeizadeh *et al.*, 2018). The inference model is of the form $q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t})$. It is thus also similar to the inference model of STORN (see (7.19)). A variant of the generative distribution of $\mathbf{z}_t$, called "learned prior" in the paper, is also proposed. It is of the form $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1})$ and can be considered a simplification of the generative distribution of $\mathbf{z}_t$ in VRNN or SRNN.

12.3 Other models

Factorized variational autoencoder (FVAE): Deng *et al.* (2017) proposed the FVAE model, which combines a VAE with tensor factorization (Kuleshov *et al.*, 2015; Huang *et al.*, 2016). This latter is applied on the latent vector $\mathbf{z}$. As one of the tensor factorization dimension is discrete time, this model implicitly involves data dynamics modeling. However, the temporal patterns are sampled from a standard log-normal

distribution, hence independently over time, and data decoding is also processed independently at each time frame. It is thus unclear how temporal dynamics are actually encoded.

Gaussian process variational autoencoder (GP-VAE): Fortuin *et al.* (2020) recently combined a VAE for the observed data dimension reduction and a multivariate Gaussian process (GP) (Williams and Rasmussen, 2006) for modeling the dynamics of the resulting latent vector $\mathbf{z}_t$. For the GP model, they used a Cauchy kernel, which is appropriate to model data with multiscale time dynamics. They proposed using an approximate posterior distribution that is also a multivariate GP (here, a first-order one). The resulting overall GP-VAE model was trained with the VAE methodology and then used to efficiently recover missing data in test sequences (in videos and medical data). One interesting property of this model is that it provides interpretable uncertainty estimates.

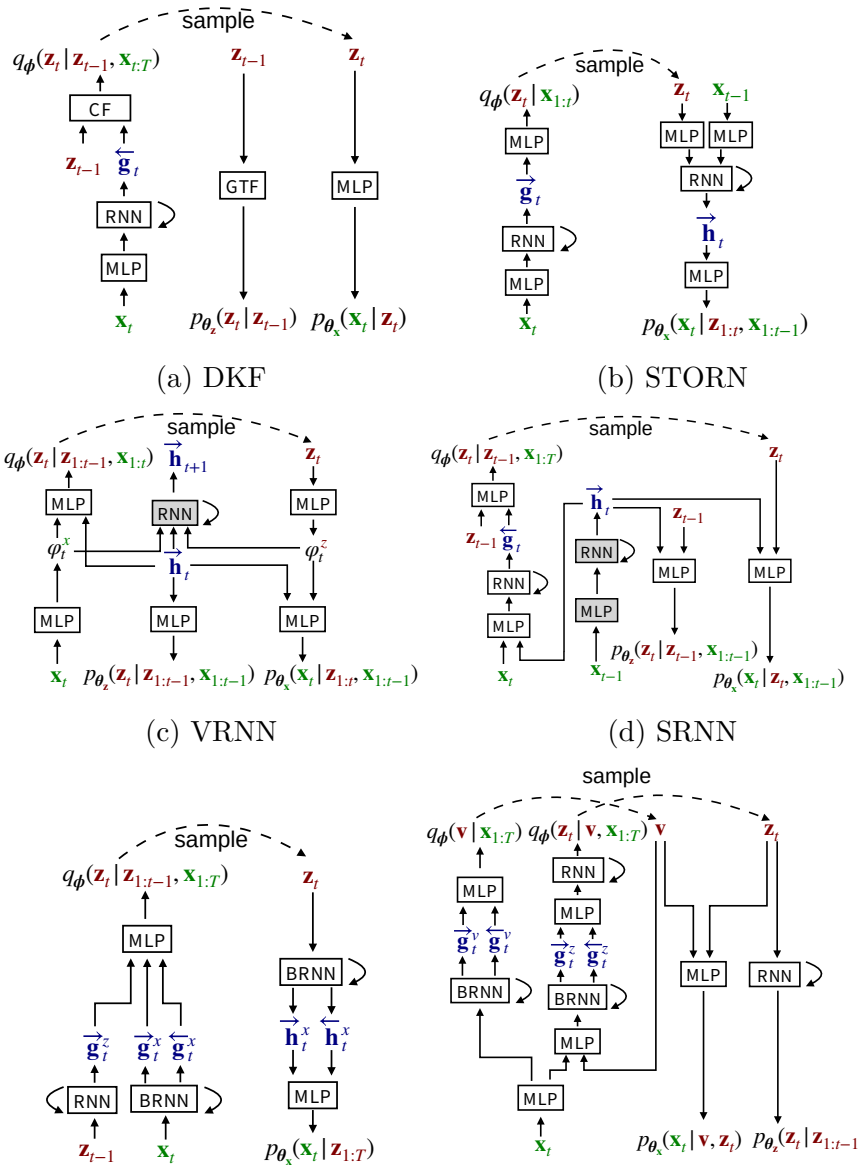
13

Experiments

In this chapter, we present an experimental benchmark of six of the DVAE models detailed in the previous chapters (DKF, STORN, VRNN, SRNN, RVAE, DSAE). This benchmark is conducted on a dataset of speech signals and a dataset of 3D human motion data. We provide a series of quantitative results on the task of analysis-resynthesis; that is encoding with the DVAE encoder followed by decoding with the DVAE decoder. We also provide qualitative results, in the form of examples of data generated by the models. We first present the models implementation in Section 13.1. Then, we describe the experimental protocol, datasets, model training and testing settings, and evaluation metrics in Section 13.2. Finally, we present and discuss the results in Sections 13.3 and 13.4. We recall that a link to the open-source code and the best-trained models can be found at https://team.inria.fr/robotlearn/dvae/.

13.1 DVAE architectures

The architectures of the six DVAE models that we benchmark are summarized in Figure 13.1. For each model, we represent the high-level computational graph corresponding to the encoding, sampling



(a) DKF (b) STORN

(c) VRNN (d) SRNN

(e) RVAE (f) DSAE

Figure 13.1: Model architecture for the six tested DVAE models. For DKF, CF and GTF are a combiner function and a gated transition function, respectively. These functions are described in Appendix B (Section B.1) and in Krishnan *et al.*'s (2015) paper. For VRNN and SRNN, the gray-shaded boxes are modules shared by the encoder and decoder.

and decoding processes. In particular, we show the types of layers that compose the encoder and decoder networks. Note that none of the DVAEs is used in the driven mode; that is, none of them feature an external input $\mathbf{u}_{1:T}$. The MLPs are generally used to extract high-level features and/or as a combiner function, whereas the recurrent networks (RNNs and BRNNs) are used to accumulate the information over time. All RNNs are instantiated as LSTM networks (and BRNNs are instantiated as bidirectional LSTM networks). DKF includes a specific combiner function (CF) at the end of the encoder and a gated transition function (GTF) to implement the dynamical model. These functions are described in the original DKF paper by Krishnan *et al.* (2015) and we report them in Appendix B (Sections B.1). We recall that the internal state vector $\mathbf{h}_t$ of VRNN and SRNN is shared between the encoder and decoder (see Sections 8.2 and 9.2). In Figure 13.1, we represent the corresponding shared networks with grey-shaded boxes. We also recall that RVAE was presented in two versions: causal and noncausal (see Section 10.1). Figure 13.1 only shows the architecture of the noncausal RVAE. The schema of the causal RVAE architecture is obtained by replacing the BRNN in the inference and generative models with a backward and a forward RNN, respectively. Finally, for all output variance parameters, we use log-parameterization (i.e., the output of the network corresponding to a variance parameter σ^2 is actually $\log \sigma^2$).

The general architectures shown in Figure 13.1 are common to the two sets of experiments that we conducted on speech data and on human motion data, although with different layer dimensions. For our experiments with speech data, the observed, latent, and RNN internal state vectors are of dimension 513, 16, and 128, respectively, for all DVAE models. For our experiments with human motion data, they are of dimension 96, 10, and 64, respectively, for all DVAE models (see Sections 13.2.1 and 13.2.2 for the definition of the observed data vectors). Low-level implementation details such as the number of layers and the number of units per layer may also differ between the two sets of experiments on speech and human motion data. These details are provided in Appendices B and C).

13.2 Experimental protocol

13.2.1 Speech data

For our experiments with speech data, we used the Wall Street Journal dataset (WSJ0; Garofolo *et al.*, 1993), which comprises speech read from WSJ news. We used the speaker-independent, medium vocabulary (5k words) subset of the corpus. More precisely, the *si_tr_s* subset (∼25 h) was used for training, the *si_dt_05* subset (∼2 h) was used for validation, and the *si_et_05* subset (∼1.5 h) was used for testing.

The raw speech waveform was sampled at 16 kHz. Analysis-resynthesis was performed with the DVAEs in the time-frequency domain on power spectrograms. Time-domain speech signals were thus preprocessed with the short-time Fourier transform (STFT), using a 64-ms sine window (1,024 samples) with 25%-overlap to obtain sequences of 513-dimensional discrete Fourier spectra (for positive frequencies). Then, we computed the squared magnitude of these STFT spectrograms. For the training dataset, we set $T = 50$, meaning that speech utterances of 0.8 s were extracted from the raw dataset and pre-processed with the STFT. In summary, each training speech sequence is a 513×50 STFT power spectrogram. This data preprocessing resulted in a set of $N_{\text{tr}} = 46,578$ training sequences (representing about 10.3 hours of speech signal) and $N_{\text{val}} = 7,775$ validation sequences ($\sim$ 1.7 h). For testing, we used the STFT spectrogram of each complete test sequence (with the beginning and ending silence portions removed), which can be of variable length, most often larger than 2.4 s.

As discussed by Leglaive *et al.* (2020) and mentioned in Section 10.1, the complex-valued STFT coefficients are modeled with a zero-mean circular complex Gaussian distribution (see (10.2)), whereas $\mathbf{z}_t$ is modeled as usual with a real-valued Gaussian distribution. The data sequence $\mathbf{x}_{1:T}$ processed by the DVAE models is the squared magnitude of the STFT spectrogram (i.e., a real-valued nonnegative power spectrogram). The corresponding phase spectrogram is directly combined with the DVAE output magnitude spectrogram to reconstruct the output speech signal using inverse STFT with overlap-add. Modeling the STFT coefficients with a zero-mean circular complex Gaussian distribution with

variance $\sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot)$ amounts to modeling each entry $x_{f,t}$ of the power spectrogram $\mathbf{x}_{1:T}$ with a Gamma distribution with shape parameter 1 and scale parameter $\sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot)$ (i.e., $x_{f,t} \sim \mathcal{G}(1, 1/\sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot))$).[1] This also amounts to using the Itakura-Saito divergence between $x_{f,t}$ and $\sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot)$ in the reconstruction term of the VLB (Girin *et al.*, 2019). We recall that all presented DVAE models are versatile regarding the conditional pdf of $\mathbf{x}_t$, and using a Gamma distribution (more appropriate for speech/audio power spectrograms) in place of the Gaussian distribution that was used in the generic presentation of the models does not present any problem. The linear layer estimating the parameters of this distribution has 513 output units corresponding to the log-variance parameters $\{\log \sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot)\}_{f=1}^F$.

13.2.2 3D human motion data

For our experiments with 3D human motion data, we used the H3.6M dataset (Ionescu *et al.*, 2014), which is one of the largest dataset of the kind and has been widely used in video prediction (Finn *et al.*, 2016), human pose and shape estimation (Bogo *et al.*, 2016), and human motion prediction (Martinez *et al.*, 2017). This dataset was obtained from multi-view video recordings of 11 professional actors performing 17 various scenarios (e.g., discussing, smoking, taking a picture, or talking on the phone), using four calibrated cameras with 50 Hz resolution. The 3D $\{x, y, z\}$ coordinates of 32 human skeleton joints were extracted from these multi-view recordings. Each set of coordinates was centered w.r.t. the coordinates of the pelvis joint.

For our experiments with DVAEs, each data frame is organized as a 96-dimensional vector $\mathbf{x}_t$ by concatenating the 3D coordinates of the 32 joints. We used sequences of $T = 50$ consecutive vectors, which represent a duration of 2 s (the data were previously downsampled by a factor 2). These sequences were obtained by applying a 50-frame sliding window on the original H3.6M sequences, with a shift of two frames. In summary, each example in our dataset is thus a matrix of 3D coordinates of skeleton joints of size 96×50, which corresponds to

[1] Here, we do not specify the variables generating the variance, as they depend on the DVAE model. Instead, the subscripts indicate frequency bin f and time frame t.

2 s of human motion.

In H3.6M, 15 scenarios from 7 actors are provided with the ground-truth annotations. Similarly to Mao *et al.* (2020), we used the data of all 15 scenarios from 5 actors (Actors 1, 6, 7, 8, and 9) for training and from 1 actor (Actor 11) for testing. Applying the sequence extraction procedure described above led to $N_{\text{tr}} = 88,952$ training sequences (~ 50 h) and $N_{\text{test}} = 13,838$ test sequences (~ 8 h). For validation, to reduce the computation time, we selected 128 sequences for each of the 15 scenarios by Actor 5 ($N_{\text{val}} = 1,920$ sequences, ~ 1 h).

Following Bayer and Osendorfer (2014) and Petrovich *et al.* (2021), the 3D human motion data vectors $\mathbf{x}_t$ are modeled by a Gaussian generative conditional distribution with a covariance matrix equal to the identity matrix.

13.2.3 Training and testing

All tested models were implemented in PyTorch (Paszke *et al.*, 2019). To train the models, we used the Adam optimizer (Kingma and Ba, 2014) with mini-batches of size 128. For the speech data, we set the learning rate to 0.002, whereas we used 0.0001 for the human motion data. We also used early stopping on the validation set with a patience of 50 epochs for the speech data and 30 epochs for the human motion data. During the training of the models with the human motion data, we applied warm-up to the KL regularization term in the VLB by multiplying it with a factor β and linearly increasing this factor from 0 to 1 after each epoch, during the first 50 epochs (Sønderby *et al.*, 2016b; Vahdat and Kautz, 2020).

Once a model was trained on the training set, with early stopping on the validation set, its weights were fixed and the model was run on the test set. We report the average performance obtained on the test set using the metrics presented in the next subsection.

13.2.4 Evaluation metrics

For the experiments on speech data, we used three metrics to evaluate the resynthesized speech quality and compare the performance of the different DVAE models: the scale-invariant signal-to-distortion ratio

(SI-SDR) in dB (Le Roux *et al.*, 2019), the perceptual evaluation of speech quality (PESQ) score in $[-0.5, 4.5]$ (Rix *et al.*, 2001), and the extended short-time objective intelligibility (ESTOI) score in $[0, 1]$ (Taal *et al.*, 2011). For all metrics, the higher the better. Note that these metrics are applied on the time-domain signals (i.e., speech waveforms). We combined the reconstructed magnitude spectrogram with the phase spectrogram of the original signal to obtain the analyzed-resynthesized speech waveform (using the inverse STFT).

For 3D human motion data, we can directly compare each original sequence with the corresponding analyzed-resynthesized sequence. We used the mean per joint position error (MPJPE) proposed by Ionescu *et al.* (2014), which is an averaged Euclidean distance per joint. We report the results in millimeters (mm). Note that this error corresponds to the log-likelihood term (or reconstruction error term) of the VLB, up to a constant factor that is controlled through the setting of the variance of the data conditional distribution.

13.3 Results on speech data

13.3.1 Analysis-resynthesis

We first present the results of analysis-resynthesis performed on the speech data. The values of the three metrics described in the previous subsection and averaged over the test dataset are reported in Table 13.1. In this experiment, all three autoregressive models (STORN, VRNN, and SRNN) were trained and tested in the teacher-forcing mode (i.e., using the ground-truth values of past observed vectors $\mathbf{x}_{1:t-1}$ when generating $\mathbf{x}_t$, see Section 4.4.1). Results of analysis-resynthesis in the generation mode will be discussed in Section 13.3.3.

From Table 13.1, we can draw the following conclusions. First, all tested DVAE models lead to fair signal reconstruction, with an SI-SDR ranging in 6.9–11 dB. This range is in accordance with the fact that we compress each 513-dimensional data vector into a 16-latent vector (using also the 128-dimensional RNN internal state that encodes the past data vectors in the case of autoregressive models). The quality of the reconstructed spectrograms is illustrated in Figures 13.2, 13.3 and 13.4.

DVAE	SI-SDR (dB)	PESQ	ESTOI
VAE	5.3	2.97	0.83
DKF	9.3	3.53	0.91
STORN	6.9	3.42	0.90
VRNN	10.0	3.61	0.92
SRNN	**11.0**	**3.68**	**0.93**
RVAE-causal	9.0	3.49	0.90
RVAE-noncausal	8.9	3.58	0.91
DSAE	9.2	3.55	0.91
SRNN-TF-GM	−1.0	1.93	0.64
SRNN-GM	7.8	3.37	0.88

Table 13.1: Performance of the DVAE models tested in our speech analysis-resynthesis experiment. The SI-SDR, PESQ, and ESTOI scores are averaged over the test subset of the WSJ0 dataset. STORN, SRNN and VRNN were trained and tested in the teacher-forcing mode. SRNN-TF-GM stands for the SRNN model trained in the teacher-forcing mode and tested in the generation mode. SRNN-GM stands for the SRNN model trained and tested in generation mode.

Figure 13.2(top) shows the power spectrogram of a speech signal uttered by a female speaker. Figure 13.2(bottom), and Figures 13.3 and 13.4 show the corresponding spectrogram obtained after analysis-resynthesis by the six tested models. Actually, the first 2 s on the left of the red line are obtained with analysis-resynthesis; the following 2 s are obtained by switching the models into generation mode, as presented in the next subsection. As for the analysis-resynthesis part (the first 2 s), we can see in these plots that the reconstructed spectrograms are all quite close to the original spectrogram. They look like a slightly smoothed or blurred version of the original spectrogram, which is typical of a data compression effect, although retaining most of the speech content characteristics. Regarding the perceptual quality of the reconstructed speech signals, Table 13.1 shows the PESQ scores that range from fair to good. The STOI scores, generally higher than 0.90, show good intelligibility.

Second, all DVAE models outperform the standard VAE model by a large margin (except maybe for STORN, which is "only" 1.6 dB SI-SDR

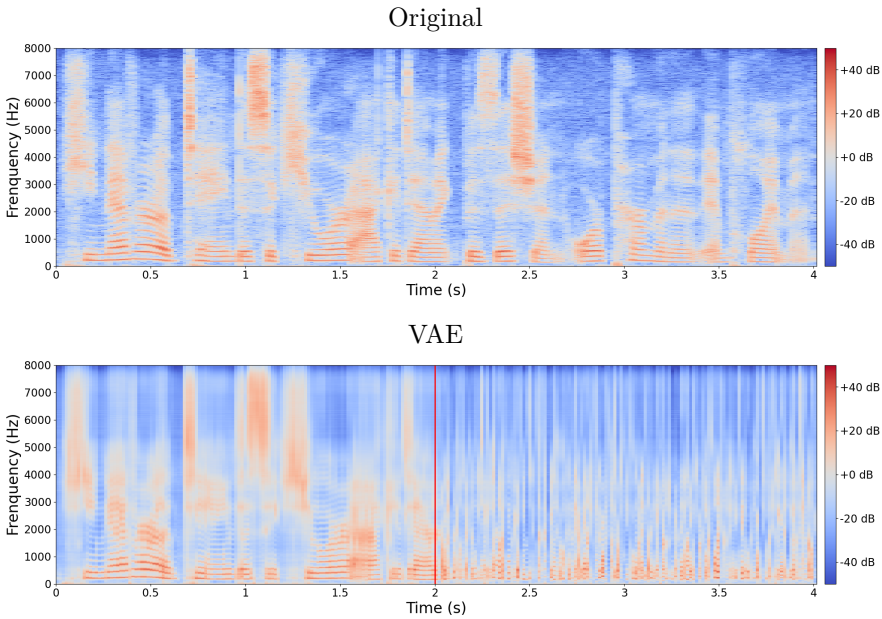


Figure 13.2: Example of power spectrogram for a speech signal uttered by a female speaker. Top: spectrogram of the original signal. Bottom: spectrogram reconstructed (0-2 s) and generated (2-4 s) with a vanilla VAE (the red line indicates the transition between reconstruction and generation).

higher). The harmonics in the spectrogram reconstructed with the VAE in Figure 13.2(bottom) are slightly noisy and blurrier compared to the harmonics reconstructed with the DVAE models. This demonstrates the merit of including temporal modeling in the VAE framework for modeling sequential data, such as speech signals. SRNN exhibits the best performance and VRNN comes second. By looking at the associated probabilistic models, we can observe that SRNN and VRNN are the most complex models in terms of dependencies between observed and latent variables. We believe that these dependencies allow SRNN and VRNN to better capture the temporal structure of speech signal than the other models. SRNN performs slightly better than VRNN (e.g., it is 1 dB SI-SDR above VRNN), although VRNN has richer variable dependencies. This may be because the inference model of SRNN respects the structure of the exact posterior distribution, whereas that of VRNN (as proposed

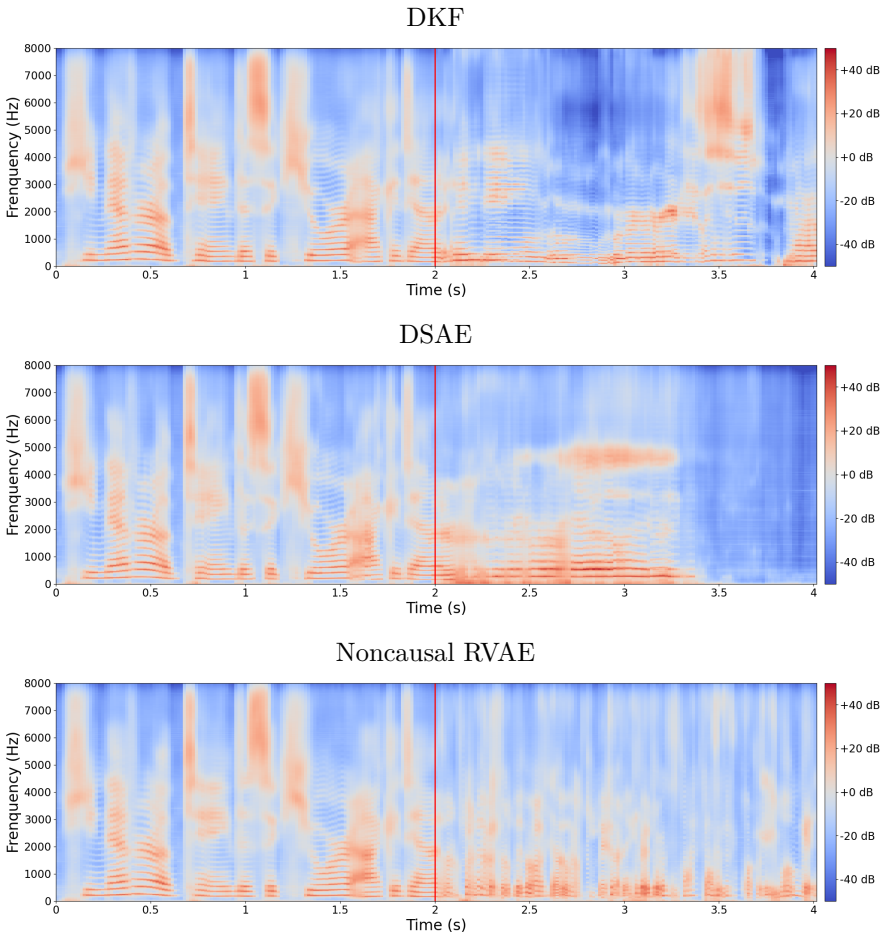


Figure 13.3: Example of speech power spectrogram reconstructed (0-2 s) and generated (2-4 s) by a DVAE model (the original spectrogram is in Figure 13.2 (top)). Top: DKF; middle: DSAE; bottom: noncausal RVAE. The red line indicates the transition between reconstruction and generation.

in the original paper and implemented here) does not. In both cases, the exact posterior of $\mathbf{z}_t$ at each time t depends on all observations $\mathbf{x}_{1:T}$. However, the inference model of VRNN only takes the causal observations $\mathbf{x}_{1:t}$ into account.

The performance scores of DKF and DSAE are very close to each

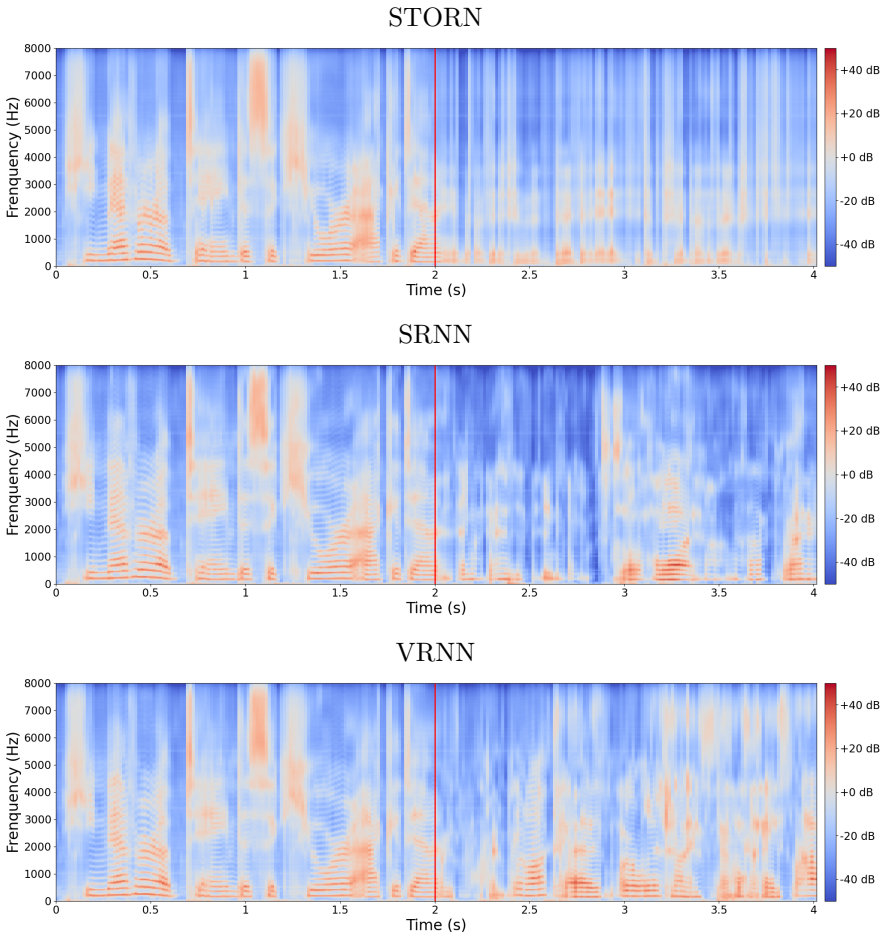


Figure 13.4: Example of speech power spectrogram reconstructed (0-2 s) and generated (2-4 s) by a DVAE model (the original spectrogram is in Figure 13.2 (top)). Top: STORN; middle: SRNN; bottom: VRNN. The red line indicates the transition between reconstruction and generation. Note that this figure was obtained with the models being trained and evaluated in a slightly different configuration regarding STFT parameters. Here, we have a window length of 512 points and an overlap of 50%. This is to illustrate the robustness of the results w.r.t. the "audio parameterization."

other, and slightly below those of VRNN. This is an interesting result, as we recall that DKF and DSAE are SSM-like models, with no explicit

temporal dependency between $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, but only between $\mathbf{z}_{t-1}$ (or $\mathbf{z}_{1:t-1}$ for DSAE) and $\mathbf{z}_t$. Thus, even if they are expected to have less "predictive" power compared to SRNN or VRNN, their SSM structure appears quite efficient at encoding the speech dynamics. DSAE can be considered as an improved version of DKF, with an additional sequence-level variable $\mathbf{v}$ and infinite-order temporal dependency of $\mathbf{z}_t$ (as opposed to first-order dependency for DKF). The fact that this more sophisticated structure does not lead to improved performance over DKF might be explained by the structure of the inference model. For DSAE, the inference of $\mathbf{z}_t$ depends on $\mathbf{x}_{1:T}$, whereas the exact posterior distribution depends on $\mathbf{z}_{1:t-1}$ and $\mathbf{x}_{t:T}$. Thus, the inference model of DSAE is not only missing some dependencies it should have (previous latent variables) but it is adding dependencies that it should not have (previous observed variables), see Section 11.2. In contrast, the DKF inference model respects the structure of the exact posterior distribution. In the end, all these differences between DKF and DSAE may compensate each other, leading to similar results. One way to improve DSAE may be to design an inference model with the structure of the exact posterior distribution.

The SI-SDR scores obtained by the RVAE model are just below those of DKF and DSAE, whereas the PESQ score of noncausal RVAE is slightly superior to those of DKF and DSAE. This is interesting considering that as DKF and DSAE, RVAE has no predictive link (e.g., no direct dependency between $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$), and contrary to DKF and DSAE, RVAE does not have a dynamical model on the latent vector $\mathbf{z}_t$ (i.e., it is modeled as an i.i.d. variable). However, in the analysis-resynthesis framework, the sequence $\mathbf{z}_{1:t-1}$ or $\mathbf{z}_{1:T}$ efficiently encodes $\mathbf{x}_{1:T}$ owing to an efficient inference model, and then, the generative model is able to exploit this whole sequence to regenerate $\mathbf{x}_t$. As expected, the noncausal version of RVAE is slightly better than the causal version.

As for STORN, its performance revealed a bit disappointing in our speech analysis-resynthesis experiment. Here also, this can be explained by the fact that the inference model of STORN does not respect the structure of the exact posterior distribution. In particular it does not use $\mathbf{z}_{1:t-1}$ nor $\mathbf{x}_{t+1:T}$, which is doubly penalizing compared to if the structure of the exact posterior distribution was considered. Another reason for

the relatively low scores obtained by STORN in these experiments is given in Section 13.3.4.

13.3.2 Generation of speech spectrograms

In this subsection, we briefly illustrate the ability of the DVAE models to generate "speech-like" spectrograms with a qualitative example. This example is the "continuation" of the example that we have seen in the previous subsection, provided in Figures 13.2, 13.3 and 13.4. In these figures, the first 2 s of each spectrogram (on the left of the red line) was obtained with analysis-resynthesis; that is, the latent vectors were provided by the inference model, using the ground-truth observed vectors as input, and the output spectrogram was then provided by the generative model using the inferred latent vectors (and the ground-truth past observed vectors for the autoregressive models). After 2 s (on the right of the red line), we turn the models to pure generation mode; that is, the latent vectors and the output spectrogram are now both provided by the generative model, without relying on the inference model and ground-truth past observed vectors anymore. This strategy allows the generation mode to benefit from a good initialization, induced by the analysis-resynthesis part. Indeed, at the time instant corresponding to 2 s, the generation starts with the past latent vectors and current RNN internal state provided by the analysis-resynthesis part, thus encoding the past observed speech data. Therefore, we can expect a smooth transition from the analysized-resynthesized spectrogram to the generated one.

As expected, we observe in Figure 13.2 that a vanilla VAE is not able to generate a spectrogram with a realistic speech-like structure. In particular, the successive spectrogram "chunks" are too short and with too abrupt transitions to be speech sounds. This is due to the fact that there is no temporal modeling.

Figure 13.3 shows the results obtained with the nonautoregressive DVAE models. We can see that the spectrograms generated by DKF and DSAE, although different, both exhibit a harmonic structure and a variety of different speech-like sounds, which smoothly evolve with time. The smoothness probably comes from the use of a Markov model for

the latent vector, which precisely enforces smoothness. In the original DSAE paper, Li *et al.* (2017) did not provide examples of generated speech spectrograms but they presented good results in voice conversion obtained by exchanging the value of the variable $\mathbf{v}$ across two sentences spoken by a different speaker. Regarding RVAE, even though the latent vectors are independently and identically sampled from a standard Gaussian distribution, we observe a generated spectrogram with a temporal structure. This means that the RVAE model is able to "recreate" correlation in the generated data by combining the vectors of an uncorrelated sequence. However, the energy is mostly concentrated in low frequencies, and the harmonic structure, although present, is not as clearly visible as for DKF and DSAE. Moreover, the segments corresponding to the successive speech sounds are shorter than for DKF and DSE and seem shorter than what is expected in natural speech.

Figure 13.4 shows the generation results with autoregressive DVAE models. We see that, as the other models, STORN manages to ensure a smooth transition between the analysis-resynthesis and generation parts, but then the quality of the generated spectrogram becomes much lower than with the other autoregressive DVAE models. SRNN generates a spectrogram with a speech-like structure and a lot of variability. The best result for this example sentence is obtained with VRNN, for which we can observe segments that resemble different phonemes, with smooth transitions between them. The harmonic structure is also clearly visible and the generated data cover the full bandwidth.

13.3.3 Training with scheduled sampling

To complement the previous results, we performed an additional analysis-resynthesis experiment with the autoregressive models being trained and tested in the generation mode instead of the teacher-forcing mode; that is, using the previously generated data vectors $\hat{\mathbf{x}}_{1:t-1}$ in place of the ground-truth past vectors $\mathbf{x}_{1:t-1}$ when generating $\mathbf{x}_t$ (see Section 4.4.1). Note that here, the sequence $\mathbf{z}_{1:T}$ is still provided by the encoder. For conciseness, we present only the results for SRNN, which was the model performing best in the first experiment above.

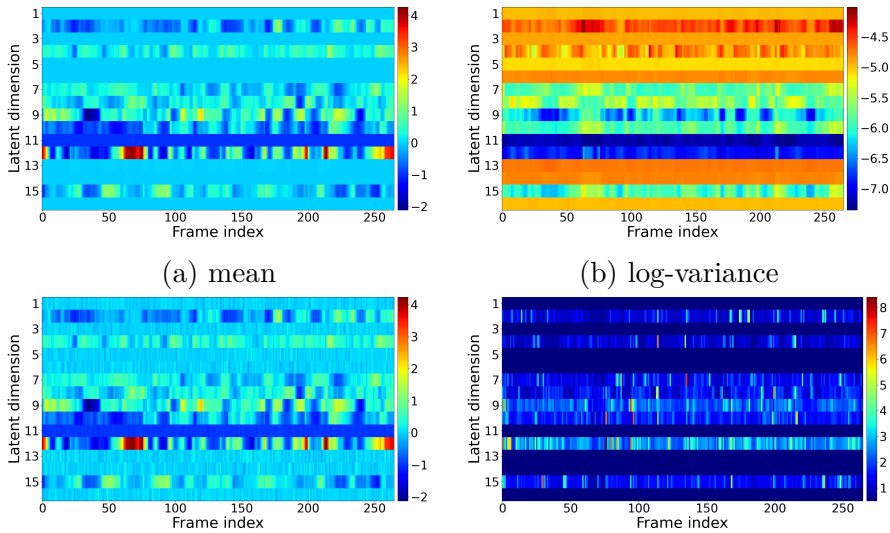
Directly training a model in the generation mode was observed to

be difficult, so we adopted a *scheduled sampling* approach (Bengio *et al.*, 2015). We started with the SRNN model trained in the teacher-forcing mode used in the previous experiment. Then, we fine-tuned this model by randomly replacing $\mathbf{x}_{1:t-1}$ with $\hat{\mathbf{x}}_{1:t-1}$ at the input of the encoder-decoder shared module, when estimating $\mathbf{x}_t$. We replaced 20% of the $\mathbf{x}_{1:t-1}$ vectors for the first 50 epochs, and increased to 40% for the next 50 epochs, and so on, until we completely replaced the ground-truth clean speech signal with generated speech signals. Then we fine-tuned with totally generated speech signals for another 300 epochs. Overall, we fine-tuned the model for 500 epochs. The resulting model is referred to as SRNN-GM hereinafter. Moreover, we also evaluated the initial SRNN model (trained with teacher-forcing) in the generation mode (i.e., we use here $\mathbf{x}_{1:t-1}$ during training and $\hat{\mathbf{x}}_{1:t-1}$ during testing when generating $\mathbf{x}_t$). We refer to this "hybrid" configuration as SRNN-TF-GM.

We can see from Table 13.1 that SRNN trained in the teacher-forcing mode and tested in the generation mode (SRNN-TF-GM) obtains very poor results. This illustrates the problem of train/test mismatch discussed in Section 4.4.2. The strategy that consists in training SRNN in the generation mode using scheduled sampling (SRNN-GM) is shown to be effective, as the gap between SRNN-TF-GM and SRNN is largely reduced. Nevertheless, SRNN-GM remains a bit below DKF and RVAE in this experiment, showing that it is more difficult to exploit the predictive links in a practical application where the ground-truth values of the observed vectors are not available, compared to the "oracle" configuration of teacher-forcing.

13.3.4 Visualization of the latent vector sequence

In this subsection, we illustrate the behavior of the learned latent vector. Figure 13.5 displays the trajectories of the mean vector of $q_{\phi_{\mathbf{z}}}$, the log-variance vector of $q_{\phi_{\mathbf{z}}}$, a vector $\mathbf{z}_t$ sampled from $q_{\phi_{\mathbf{z}}}$, and the KL divergence term of the VLB, for an example sentence of the speech test dataset and for the SRNN model trained and applied in the teacher-forcing mode. We observe that some of the dimensions of the latent vector, for example the first one, show a steady profile of the mean (Figure 13.5 (a)) and log-variance (Figure 13.5 (b)) for these dimensions,



(a) mean (b) log-variance

(c) sampled latent vector (d) KL divergence

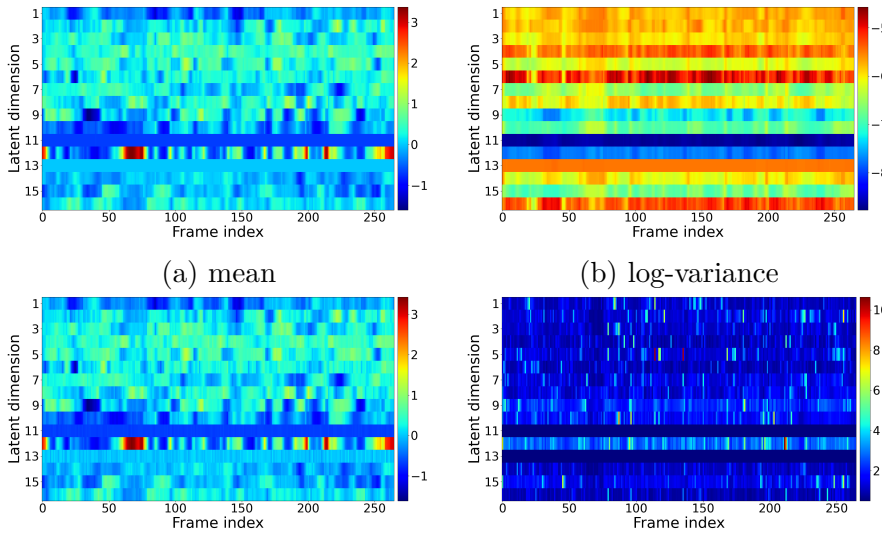
Figure 13.5: Example of the behavior of the latent vector of a speech spectrogram for SRNN (trained with teacher-forcing). (a) mean value of the posterior distribution (i.e., $\boldsymbol{\mu}_{\phi_{\mathbf{z}}}(\cdot)$); (b) log-variance of the posterior distribution (i.e., $\boldsymbol{\sigma}^2_{\phi_{\mathbf{z}}}(\cdot)$); (c) sampled latent vector $\mathbf{z}_t$; (d) KL divergence term of the VLB.

with the mean being close to zero and the variance being much lower than 1. As a result, the corresponding entries of the $\mathbf{z}_t$ samples shown in Figure 13.5 (c) look like noise with small fluctuations around zero. In short, those dimensions are noninformative, whereas the other "active" dimensions show much larger, and thus informative, fluctuations (note that for these "active" dimensions, because the variance is also small, yet not steady, the sampled latent values are close to the mean). This inactivity of certain dimensions is the sign of the *posterior collapse* problem. The latent vector $\mathbf{z}_t$, or at least some dimensions of it, becomes noninformative, as its posterior distribution becomes too close to its prior (generative) distribution, as illustrated by the small (and steady) values of the KL divergence term of the VLB, compared to the other "active" dimensions (Figure 13.5 (d)). This problem has been well identified and largely discussed in the VAE literature (Bowman *et al.*, 2016; Serban *et al.*, 2016; Chen *et al.*, 2017; Lucas *et al.*, 2019; Razavi *et al.*, 2019;

Dai *et al.*, 2020). It remains a largely open topic in the framework of DVAE, as further discussed in Chapter 14.

Overall, the input of the decoder consists of some dimensions containing informative patterns and some other dimensions containing low-variance stationary noise. In the present example of the SRNN model, we identified 8 dimensions out of 16 that are active and 8 dimensions that seem to have collapsed, suggesting that we would obtain similar analysis-resynthesis performance by setting the size of the latent vector to 8. In these experiments with speech signals, we observed that VRNN has only 2 collapsed dimensions out of 16, whereas STORN had 12 collapsed dimensions out of 16; that is, only 4 active dimensions for STORN. This latter point is to be related to the fact that STORN is the less efficient of the three autoregressive models in these experiments, with performance significantly below that of SRNN and VRNN. In contrast, we did not observe posterior collapse of any dimension for the nonautoregressive models (DKF, DSAE, and RVAE) in our experiments; that is, for the nonautoregressive models, all 16 dimensions are useful to encode $\mathbf{x}_t$. This is consistent with the fact that, for these models, there is no predictive link and all the information in $\mathbf{x}_{1:T}$ must be encoded in $\mathbf{z}_{1:T}$.

We have checked that, for a given model, the active/inactive dimensions remain the same across different examples. In addition, as shown in the example of Figure 13.5, the dimensions suffering from posterior collapse are the same over time. In principle, a small KL divergence at a certain time frame means that the posterior and the generative distributions of $\mathbf{z}_t$ are very close to each other for that time frame. However, this does not guarantee that the generative distribution on two consecutive frames remains the same, and the same for the posterior distribution. In practice, we observe that both the generative and posterior distributions of collapsed dimensions are time-invariant and noninformative. We believe this is due to the way these models are implemented in practice, since the architectures used to implement the DVAEs are time-invariant; that is, the same architecture with the same weights is used at every time step. Combined with the fact that noninformative generative distributions are zero-mean and low-variance Gaussians, this leads us to conjecture that posterior collapse in a given



(a) mean (b) log-variance

(c) sampled latent vector (d) KL divergence

Figure 13.6: Example of the behavior of the latent vector of a speech spectrogram for SRNN (trained in the generation mode with scheduled sampling). (a) mean value of the posterior distribution (i.e., $\boldsymbol{\mu}_{\phi_{\mathbf{z}}}(\cdot)$); (b) log-variance of the posterior distribution (i.e., $\boldsymbol{\sigma}^2_{\phi_{\mathbf{z}}}(\cdot)$); (c) sampled latent vector $\mathbf{z}_t$; (d) KL divergence term of the VLB.

dimension of $\mathbf{z}$ is associated to very small weights to compute the mean and log-variance of that dimension. We have verified this statement by visualising the weights of the output linear layers (not shown here for conciseness).

Figure 13.6 displays the same trajectories as Figure 13.5, for the same example sentence, but for the SRNN model trained in the generation mode (with scheduled sampling) instead of the teacher-forcing mode. By comparing the two figures, we observe that the SRNN model trained in the generation mode exhibits much less posterior collapse, with a larger number of "active" dimensions. By definition, the generated data vectors are approximate values of the ground-truth vectors, hence the model trained in the generation mode uses less accurate and thus less reliable past values of $\mathbf{x}_t$ to generate the current value, compared to the same model trained in the teacher-forcing mode. Therefore, the

DVAE	MPJPE (mm)
VAE	48.69
DKF	42.21
STORN	9.47
VRNN	9.22
SRNN	7.86
RVAE-Causal	31.09
RVAE-NonCausal	28.59
DSAE	28.61
SRNN-TF-GM	221.87
SRNN-GM	43.98

Table 13.2: Performance of the DVAE models tested on 3D human motion data analysis-resynthesis. The MPJPE scores are averaged over the test subset of the H3.6M dataset.

model trained in the generation mode needs more informative latent dimensions to resynthesize $\mathbf{x}_t$.

13.4 Results on 3D human motion data

13.4.1 Analysis-resynthesis

Table 13.2 shows the results of the analysis-resynthesis experiment with the 3D human motion data. The MPJPE values are approximately within 9–49mm, which is relatively small compared to the average amplitude of the joint coordinates in a human body, and therefore show a fair to good reconstruction for all models. As for the speech analysis-resynthesis experiment, all DVAE models outperform the vanilla VAE model. This confirms the interest of using DVAE models for modeling sequential data.

In this experiment with human motion data, the autoregressive DVAEs largely outperform the nonautoregressive DVAEs. STORN, VRNN, and SRNN have an MPJPE of about 9.5, 9.2 and 7.9 mm, respectively, whereas DKF, RVAE and DSAE (noncausal) obtain about 42.2, 28.6 and 28.6 mm, respectively. Therefore, the performance gap

between the autoregressive models (trained and tested in the teacher-forcing mode) and the nonautoregressive models is larger than in our experiment with speech signals. We conjecture that this is because the 3D human motion data has less variability (or, say, smoother trajectories) compared to speech data. Therefore, for such data, knowing the ground-truth values of the previous observation(s) ($\mathbf{x}_{t-1}$ or $\mathbf{x}_{1:t-1}$) is a very strong information for predicting the current observation.

Again, SRNN exhibits the best performance, which is consistent with the analysis-resynthesis results obtained with the speech data. In this new experiment with human motion data, STORN is more efficient than in our experiment with speech data. In contrast, DKF underperforms compared to the other nonautoregressive models and exhibits a quite limited improvement over the vanilla VAE.

13.4.2 Generation of 3D human motion sequences

Example videos of a human "skeleton" animated from 3D motion data sequences generated by the different DVAE models are available at https://team.inria.fr/robotlearn/dvae/.

13.4.3 Training with scheduled sampling

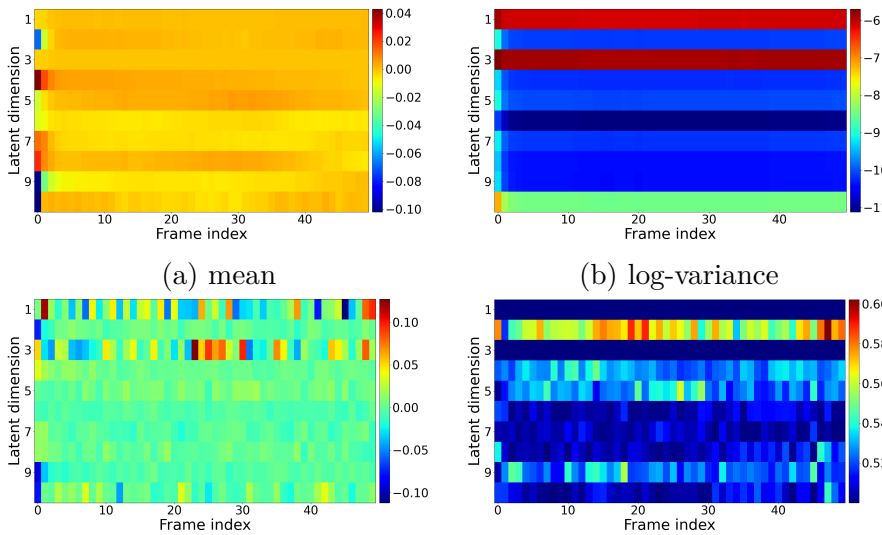
As for the experiment with speech signals, we have tested the influence of training and testing the models in the generation mode (using scheduled sampling for training). Here, we briefly report and comment the results obtained on the 3D human motion data with SRNN. We can see from Table 13.2 that, similarly to what we observed in our experiment with speech signals, SRNN-TF-GM has very poor performance, whereas training SRNN with scheduled sampling partially addresses this problem, placing SRNN-GM between the vanilla VAE and the nonautoregressive DVAEs. However, the gain in performance of SRNN-GM over the vanilla VAE is here quite limited, and we believe there is room for improvement when designing the model adaptation method. In other words, in these experiments, we adopted a simple scheduled sampling strategy and did not further investigated this issue, but other strategies to bridge the gap between ground-truth and generated data could be investigated.

13.4.4 Visualization of the latent vector sequence

To conclude this set of experiments, we also provide an example of visualization of the latent space of the 3D human data, as we did for speech signals.

Figure 13.7 displays the trajectories of the mean vector of $q_{\phi_{\mathbf{z}}}$, the log-variance vector of $q_{\phi_{\mathbf{z}}}$, a vector $\mathbf{z}_t$ sampled from $q_{\phi_{\mathbf{z}}}$, and the KL divergence term of the VLB, for an example sequence of the 3D human motion test dataset and for the SRNN model trained and applied in the teacher-forcing mode. Figure 13.8 displays the corresponding trajectories for the SRNN model trained and applied in the generation mode (trained with scheduled sampling). We can see in Figure 13.7 that the trajectories of the parameters (and of the sampled $\mathbf{z}_t$ vector) are (much) smoother than in the case of speech signals, as, again, the 3D human motion data themselves have smoother trajectories compared to speech data. Some dimensions seem more "active" than others, even if, without a deeper investigation, it is quite difficult to interpret the range of values covered by the entries of the latent vector. Such a thorough investigation is beyond the scope of the present paper. We simply note here that there are two dimensions, dimensions 1 and 3, that seem to collapse. For these two dimensions, the mean is steady around zero and the variance has a large (and steady) value, hence the sampled trajectory of the corresponding $\mathbf{z}_t$ entries looks like noise. In contrast, some other dimensions seem to have an interesting informative profile. For example, dimensions 4 and 5 have an opposite fluctuation, probably encoding an opposite evolution of the corresponding factors of data variation.

In Figure 13.8, the latent vector generally seems more "active" or informative than in Figure 13.7. For most dimensions, the ratio between the range of the mean variation and the range of variance variation is larger. The range of the KL divergence term values is also larger than in Figure 13.7, indicating that the approximate posterior and the "prior" (generative distribution of $\mathbf{z}_t$) are less close to each other than in the teacher-forcing case. These observations are consistent with those made on the speech signals in Section 13.3.4. Our conjecture that the latent variable was less important in the teacher-forcing mode than in the



(a) mean (b) log-variance

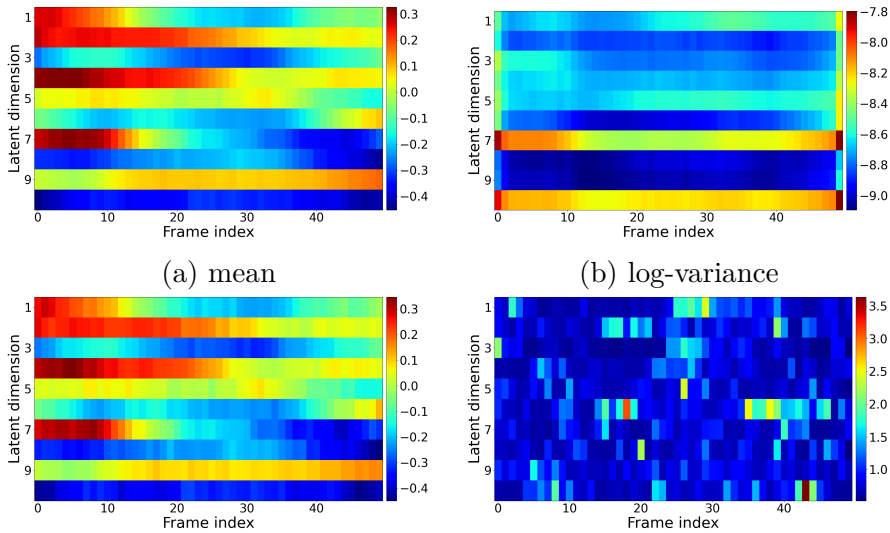
(c) sampled latent vector (d) KL divergence

Figure 13.7: Example of the behavior of the latent vector for a test sequence from the H3.6M dataset and for SRNN (trained with teacher-forcing). (a) mean value of the posterior distribution (i.e., $\boldsymbol{\mu}_{\phi_{\mathbf{z}}}(\cdot)$); (b) log-variance of the posterior distribution (i.e., $\boldsymbol{\sigma}^2_{\phi_{\mathbf{z}}}(\cdot)$); (c) sampled latent vector $\mathbf{z}_t$; (d) KL divergence term of the VLB.

generation mode is thus confirmed with the motion data.

13.5 Conclusion

In a practical application requiring the modeling of temporal data such as speech spectrograms or 3D human motion data, using either VRNN or SRNN seems a relevant choice, especially if autoregressive models can be used in the teacher-forcing mode. In addition, considering the above results and associated discussion, we suspect that having an inference model that respects the exact variable dependencies at inference time is important for obtaining the optimal performance. However, this is not always possible, as some applications require a causal inference model for online processing. Finally, in a practical application where only $\mathbf{z}_{1:T}$ needs to be transmitted, data resynthesis from $\mathbf{z}_{1:T}$ with an autoregressive model used in the generation mode was shown to



(a) mean (b) log-variance

(c) sampled latent vector (d) KL divergence

Figure 13.8: Example of the behavior of the latent vector for a test sequence from the H3.6M dataset and for SRNN (trained in generation mode with scheduled sampling). (a) mean value of the posterior distribution (i.e., $\boldsymbol{\mu}_{\phi_{\mathbf{z}}}(\cdot)$); (b) log-variance of the posterior distribution (i.e., $\boldsymbol{\sigma}^2_{\phi_{\mathbf{z}}}(\cdot)$); (c) sampled latent vector $\mathbf{z}_t$; (d) KL divergence term of the VLB.

be reasonably robust in our experiment with SRNN and speech data, provided that the model is fine-tuned in the generation mode (using here scheduled sampling). For 3D motion data, SRNN-GM was shown in our experiments to perform more poorly (with a limited gain over the vanilla VAE).

The performance of some other models, in particular DKF, also seem to depend on the data type. We thus insist that the above "model ranking" is valid only for the presented experiments, which involve pure analysis-resynthesis of speech spectrograms or 3D human motion data. For data generation, we presented only a limited set of qualitative examples to illustrate the behavior of the models (evaluating the quality of generated data is still a very difficult problem and a hot topic of machine learning with generative model). For other tasks, such as signal/data transformation (with modification of the latent vectors

between analysis and resynthesis), we do not know if our experimental results would generalize. In particular, it is difficult to know how much of the information contained in $\mathbf{x}_{1:T}$ is encoded into $\mathbf{z}_{1:T}$, or what "features" of $\mathbf{x}_{1:T}$ are encoded. In particular, in the presented experiments we did not investigate the disentanglement power of each model and how the disentangled latent dimensions can be interpreted as relevant factors of variation of the data (from a physical point of view for example). Also, we illustrated the posterior collapse problem but we did not proceed to a thorough quantitative investigation of this issue. These points will be further discussed in the next chapter. However, the experiments dedicated to illustrating them are beyond the scope of the present paper.

14

Discussion

In this chapter, we conclude our review of DVAEs with a discussion. First, we recall the fundamental motivation for designing and using DVAEs and then comment on their remarkable flexibility at multiple levels (design of the generation and inference models, high-level and low-level implementation). Then, we return to the crucial point of the disentanglement of latent factors in the present context of sequential data processing. Finally, we present some perspectives on data source coding.

14.1 Fundamental motivation for DVAEs

The fundamental motivation for designing and using DVAEs is to combine various dynamical models, aimed at modeling the dynamics of sequential data, and various VAEs, aimed at modeling the latent factors of data variations. In doing so, we expect to separate the data dynamics from the other factors of variations (see Section 14.3.3 below dedicated to this specific point) and use the latter to augment the expressivity of the models. Another way to express this idea is to point out the superiority of DVAEs over RNNs: Adding a latent variable $\mathbf{z}_t$ within an RNN adds considerable flexibility and modeling power to the

conditional output density. Let us here quote Chung *et al.* (2015):

> "We show that the introduction of latent random variables can provide significant improvements in modelling highly structured sequences such as natural speech sequences. We empirically show that the inclusion of randomness into high-level latent space can enable the VRNN to model natural speech sequences with a simple Gaussian distribution as the output function. However, the standard RNN model using the same output function fails to generate reasonable samples. An RNN-based model using more powerful output function such as a GMM can generate much better samples, but they contain a large amount of high-frequency noise compared to the samples generated by the VRNN-based models."

Similarly, we can point out the superiority of DVAE over classical (nondeep) DBNs and SSMs, owing to the deep nonlinear layers of information processing. Again, let us quote Chung *et al.* (2015):

> "Drawing inspiration from simpler dynamic Bayesian networks (DBNs) such as HMMs and Kalman filters, the proposed variational recurrent neural network (VRNN) explicitly models the dependencies between latent random variables across subsequent timesteps. However, unlike these simpler DBN models, the VRNN retains the flexibility to model highly non-linear dynamics."

Of course, such a statement applies to the entire DVAE family of models.

14.2 DVAE outcome: A story of flexibility

14.2.1 Flexibility of the generative model(s)

As seen in this review, various generative models can be derived from the general form (4.4) by simplifying variable dependencies. The models we have reviewed (such as STORN, VRNN, and SRNN) are instances of these possible generative models, but there are other possibilities.

Moreover, each model has several variants: Driven/undriven mode, predictive/nonpredictive mode, and with one or several feature extractors.

When designing a generative model, complexity issues can be considered. For example, we can quote Bayer and Osendorfer (2014):

> "[...] we can restrict ourselves to prior distributions over the latent variables that factorize over time steps, i.e., $p(\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{z}_t)$. This is much easier to handle in practice, as calculating necessary quantities such as the KL-divergence can be done independently over all time steps and components of $\mathbf{z}_t$."

However, at the same time, the systematic aspect of the VAE methodology (and the versatility of the current deep learning toolkits) enables, in principle, to train a model of arbitrary complexity. Hence, if one is not limited by computational cost, this offers new possibilities. For example, we can recall our remark in Section 11.1 about the choice of the dynamical model in deep SSMs. In the DVAE framework, not only is it easy to move from a linear dynamical model to a nonlinear one, but also to move from a first-order temporal model to a (much) higher order.

14.2.2 Flexibility of the inference model(s)

In DVAEs, as in standard VAEs, the exact posterior distribution is usually intractable due to nonlinearities, which is why we have to define an inference model in addition to the generative model (we cannot apply the Bayes rule analytically). However, a key feature of DVAEs with respect to standard VAEs is that we must define an inference model over a *sequence* of latent vectors. Even though the exact posterior distribution over this latent sequence is analytically intractable, we can leverage the chain rule and the D-separation principle to analyze the structure of the exact posterior distribution induced by the chosen generative model. It seems quite natural to exploit this knowledge to design the structure of the inference model so that it is consistent with the structure of dependencies in the exact intractable posterior. Yet, several seminal papers on DVAEs have not followed this "consistency principle," and,

more importantly, not justified the chosen structure of the inference model. Nevertheless, it is not mandatory to follow the structure of the exact posterior distribution to design the inference model. For instance, if the structure of the exact posterior distribution implies a noncausal processing of the observations, the anticausal dependencies can be dropped for the purpose of online applications. Simplifying posterior dependencies can also be motivated by a need to reduce the computational complexity of inference.

Another key difference between DVAEs and VAEs relates to how the VLB (or, actually, its estimate) is computed. The VLB involves intractable expectations, which are usually replaced with empirical averages, using samples drawn from the inference model. The sampling procedure in DVAEs has to be recursive due to the dynamical nature of the model, a constraint that standard VAEs do not have. This recursive sampling is related to the use of RNNs and can be costly. As will be discussed below, other neural network architectures can be more computationally efficient than RNNs.

14.2.3 Flexibility of the implementation

As already discussed in Section 4.1.2, various possibilities exist for the high-level implementation of DVAEs. We recall that various developed model representations can correspond to the same compact representation. In fact, the compact form describes all parent-child relationships among random variables, regardless of how these relationships are implemented in practice. Therefore, the compact representation is important to understand the probabilistic dependencies between variables. However, one must be aware that the optimization does not search for all possible models satisfying the relationships of the compact representation but only for a specific model corresponding to the developed representation. This representation allows us to understand how the dependencies are implemented in practice, and therefore, over which parameter space the model is optimized. This representation typically involves a recurrent architecture. While such architectures allow the encoding of high-order temporal dependencies, their developed graphical representations generally do not exhibit dependencies higher than

first-order. Therefore, the developed representation can be "visually misleading." This duality is important in DVAEs, and we encourage to provide both representations when presenting and discussing DVAEs, as done in this review.

Once the high-level DVAE architecture is chosen, various possibilities exist for the low-level implementation: network type (e.g., LSTM against GRU) and low-level (hyper)parameterization (number of layers in a network, number of units per layer, type of activation function, and classical deep learning modules, such as batch normalization). We choose not to detail these low-level implementation aspects in this review, and instead, considered them a deep learning routine. All these choices (or at least part of them) depend on data nature and datasets and can significantly impact the modeling performance.

14.2.4 Other network architectures for sequential data modeling

In this review, we focus on deep generative latent-variable models of sequential data using RNNs (or simple feed-forward fully-connected DNNs for first-order temporal dependencies). However, other neural network architectures can deal with sequential data of arbitrary length, the most popular ones probably being convolutional architectures. While RNNs are (virtually) based on infinite-order temporal modeling, CNNs generally have a fixed-length receptive field, which implies a finite-order temporal modeling. In particular, temporal convolutional networks (TCNs) are becoming increasingly popular due to their competitive performance with RNNs (e.g., in speech separation (Luo and Mesgarani, 2019)) while being more flexible and computationally efficient (Bai *et al.*, 2018). A TCN is based on dilated 1D convolutions, sharing similarities with the Wavenet architecture (Oord *et al.*, 2016a), and just as an RNN, it outputs a sequence of the same length as that of the input sequence. The combination of TCNs with VAEs was explored by Aksan and Hilliges (2019).

Another popular neural network architecture that can deal with sequential data is the transformer (Vaswani *et al.*, 2017), which is based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. However, only a few studies have considered

leveraging transformers for generative modeling in the VAE framework. We found only transformer-based VAEs recently proposed for sentence generation (Liu and Liu, 2019), story completion (Wang and Wan, 2019), and music representation learning (Jiang *et al.*, 2020).

14.3 VAE improvements and extensions applicable to DVAEs

Following the seminal VAE papers by Kingma and Welling (2014) and Rezende *et al.* (2014), many papers have been proposed for VAE improvements and extensions. In this subsection, we mention some of these improvements and extensions and discuss their relation and possible adaptation to DVAEs. This is a nonexhaustive review; the purpose of the present paper is not to deepen this rich part of VAE literature but rather to show that the DVAE development is still largely open, and one way to improve DVAEs is to get inspired from the recent studies on VAEs. The interested readers can refer to Kingma and Welling's (2019) paper for a more detailed review of the "static" VAE improvements and extensions.

14.3.1 Improved VAE decoders and the posterior collapse problem

The mathematical formulation of the VAE in the seminal paper by Kingma and Welling (2014) considered a 1D data vector framework; that is, $\mathbf{x}$ is a fixed-size F-dimensional vector. What happens if we want to apply the VAE to 2D data, such as images, or more generally, to N-dimensional data? What happens if one of the dimensions is variable, like variable-length time sequences? By considering variable-size sequences and, in particular, variable-length time sequences, we take a step towards DVAEs. However, we consider models with many-to-one encoding and one-to-many decoding more as VAEs than DVAEs, following the line announced in the Introduction.

Kingma and Welling (2014) considered the application of the VAE to 2D image modeling. However, the correlation between neighboring pixels was poorly exploited, as the conditional generative model (conditioned on the latent variable) was pixelwise independent.[1] In such a setting, $\mathbf{z}$

[1] Technically, an image is arbitrarily reshaped into a vector $\mathbf{x}$, with each pixel

encodes both the local statistics of an image (e.g., local texture) and the global structural information of the image (e.g., objects), whereas it is desirable to separate this information, following the essence of latent factors disentanglement.

Subsequent studies (Gulrajani *et al.*, 2016; Gregor *et al.*, 2016; Chen *et al.*, 2017; Lucas and Verbeek, 2018; Shang *et al.*, 2018) considered mixing the VAE latent representation with a more sophisticated decoder exploiting local pixel correlations with either convolutive or autoregressive decoding (Oord *et al.*, 2016b; Oord *et al.*, 2016c), possibly combined with a multilevel or hierarchical latent encoding (see Section 14.3.4). For example, Chen *et al.* (2017) considered an autoregressive conditional density of the form $p_{\theta_{\mathbf{x}}}(\mathbf{x}|\mathbf{z}) = \prod_i p_{\theta_{\mathbf{x}}}(\mathbf{x}_i|\mathbf{z}, \mathbf{x}_{\mathrm{nb}[i]})$ with application to 2D image modeling, where $\mathbf{x}_i$ is the i-th pixel of the image and $\mathbf{x}_{\mathrm{nb}[i]}$ are the neighboring pixels. The autoregressive part is typically implemented with an RNN (Oord *et al.*, 2016c). Ideally, the local statistics of an image should be modeled by the autoregressive part, whereas the global structural information of the image should be encoded in $\mathbf{z}$. Another example of a structured VAE for modeling images is VAEs based on convolutional neural networks (CNNs) (Gulrajani *et al.*, 2016; Gregor *et al.*, 2016), which decompose/recompose an image into/from successive feature maps.

Chen *et al.* (2017) discussed the tendency of the autoregressive part of the model to capture all information on the data structure and let the latent variable remain unused. This problem is referred to as *latent variable vanishing* or as *posterior collapse* in the literature, a term that we have already encountered in the preceding chapter. A general strategy to counter this effect (i.e., controlling the data features encoded by the RNN and the data features encoded in $\mathbf{z}$) is proposed by Chen *et al.* (2017) at an early level of the model design: The local autoregressive window is constrained to be small, weakening the modeling power of the decoder. This can also be done with a hierarchical structure of the latent space (see the next subsection), possibly combined with the different levels of image feature maps in CNNs (Gulrajani *et al.*, 2016;

being an entry of this vector, and the conventional "vector" VAE model, as described in Section 2, is applied. Therefore, each pixel is modeled independently conditioned on $\mathbf{z}$, even though all pixels are not assumed marginally independent.

Gregor *et al.*, 2016), or by introducing in the training procedure an auxiliary loss function that controls which information is captured by **z** and what is left to the autoregressive decoder (Lucas and Verbeek, 2018).

The posterior collapse problem has also been observed and discussed in the context of natural language processing (Bowman *et al.*, 2016; Serban *et al.*, 2016). Here, a sequence of words, individually pre-encoded into word embedding vectors, is encoded into and/or decoded from a single latent vector **z**. In Bowman *et al.*'s (2016) paper, both the encoder and decoder are single-layer LSTM RNNs. In this case, the problem is that it is difficult for the latent vector **z** to encode the content of a long input sentence, and again, the RNN internal state vectors tend to encode the whole information, leaving **z** unused. Bowman *et al.* (2016) proposed two strategies to address this problem. The first one is applying annealing to the KL term of the VAE: A weighting factor growing from 0 to 1 is applied progressively to this term during training, first forcing **z** to encode the data information and only then forcing **z** to get disentangled. The second strategy is, as above, a deliberate weakening of the decoder, here by masking a part of the word embedding sequence during training. A more complex strategy was proposed by Yeung *et al.* (2017), where an extra latent variable was added to activate/deactivate certain subvectors of **z**. As only a small part of the latent representation is used at each learning step, the VAE does not need to deactivate some of the dimensions of the latent variable. A heuristic approach was proposed by He *et al.* (2018), where the encoder is aggressively trained (i.e., trained for many iterations) before each training iteration of the decoder. The main intuition behind this approach is that the encoder has difficulties catching up with the changes in the exact posterior distribution and is lagging behind. Aggressively training the encoder allows it to catch up with the evolution of the posterior distribution at each encoder update. Other more recent solutions to the posterior collapse problem in VAEs have been proposed and discussed by Lucas *et al.* (2019), Razavi *et al.* (2019), and Dai *et al.* (2020).

Generally, the solutions to the posterior collapse problem proposed in the literature have yielded a more influential, as well as a more disentangled latent representation. Yet, there is still room for improvement. The

DVAEs focused on in this review do not consider a single latent vector $\mathbf{z}$ for a data sequence; rather, they consider a latent vector sequence $\mathbf{z}_{1:T}$, which is generally synchronized with the data sequence $\mathbf{x}_{1:T}$ and with the sequence(s) of internal state vectors of the temporal models. This raises new issues and challenges, compared to the studies conducted on, for example, 2D image or language/text modeling. However, an important remark that is worth mentioning, although quite trivial, is that this DVAE configuration first solves the encoding capacity problem for large data sequences. As mentioned by Li and Mandt (2018),

> "[the model] keeps track of the time-varying aspects of $\mathbf{x}_t$ in $\mathbf{z}_t$ for every t, making the reconstruction to be time-local and therefore much easier. Therefore, the stochastic model is better suited if the sequences are long and complex."

In short, with DVAEs, it is quite unlikely that a posterior collapse finds its origin in the limited capacity of the latent vector. In fact, we conjecture that it may be the opposite (i.e., a too large capacity of the latent vector sequence $\mathbf{z}_{1:T}$, depending on the size of $\mathbf{z}_t$) that leads to posterior collapse. This is suggested by what we observed in our experiments in Sections 13.3.4 and 13.4.4 with the autoregressive models, where inactive entries of $\mathbf{z}_{1:T}$ might be considered as "superfluous" components. Therefore, adjusting the dimension of $\mathbf{z}_t$ so that it can optimally fit to the content of the observed data sequence (i.e., adjusting the "coding cost" of the latent representation) while limiting the computational complexity is a major issue in DVAEs. All that being said, in autoregressive DVAE models such as SRNN or VRNN, even if we have a "high-capacity" sequence of latent variables, there is no guarantee that the autoregressive part of the model will not capture most of the information, thus ignoring the sequence of latent vectors. This remains an open problem in the DVAE literature.

14.3.2 Improved inference models and algorithms

As shown in Chapter 2, learning in the VAE framework relies on amortized variational inference techniques. In fixed-form variational inference (Honkela *et al.*, 2010; Salimans and Knowles, 2013) and, in particular,

stochastic variational inference (Hoffman *et al.*, 2013), the approximate posterior distribution is fixed to a certain parametric form, say Gaussian for instance, and its mean and variance parameters are "freely" optimized through direct maximization of the VLB. In amortized variational inference, the approximate posterior distribution is still Gaussian, but there is an additional constraint imposed by the fact that its parameters are provided by an inference model corresponding to the encoder network in the VAE case. This is an additional constraint in the definition of the variational family, and the resulting amortized approximate posterior distribution is generally less expressive than its counterpart with free parameters. Consequently, the KL divergence between the approximate and exact posterior distributions is generally increased in the amortized variational inference setting, which is referred to as the *amortization gap* (Cremer *et al.*, 2018; Krishnan *et al.*, 2018). This issue can also limit the performance of the learned generative model, as the amortization gap is directly related to the gap between the VLB and the intractable log-marginal likelihood of the data, which is the criterion that we would ideally like to optimize to learn the generative model parameters (i.e., the parameters of the VAE decoder).

To reduce this gap, several studies have proposed resorting to more sophisticated inference models. Normalizing flow (Rezende and Mohamed, 2015) builds arbitrarily complex approximate posterior distributions with tractable densities by applying a series of invertible transformations to a simple initial distribution. Various normalizing flows have been proposed in the literature, for instance, based on autoregressive models (Kingma *et al.*, 2016; Chen *et al.*, 2017). Because a normalizing flow consists in chaining multiple transformations of an initial latent variable, it can be considered a particular case of a hierarchical model (Kingma and Welling, 2019; Kingma *et al.*, 2016), a type of model that we will discuss in Section 14.3.4. An expressive approximate posterior distribution can also be defined as a mixture of simpler distributions by introducing auxiliary latent variables in the approximate posterior itself and then marginalizing (Maaløe *et al.*, 2016; Ranganath *et al.*, 2016; Salimans *et al.*, 2015).

An alternative to the design of more sophisticated inference models consists in directly modifying the inference algorithm. Marino *et al.*

(2018a), proposed an iterative amortized inference technique, which consists in iteratively estimating the approximate posterior parameters using a parametric *iterative inference model* that takes as input the current estimate of the parameters, the approximate gradient of the VLB (w.r.t. the approximate posterior parameters), and potentially the observed data. This iterative inference model can, for instance, be defined using a neural network. Similarly to the "learning to learn" principle (Andrychowicz *et al.*, 2016), iterative inference models learn to perform optimization of the VLB for approximate posterior estimation. Moreover, through the encoding of the VLB gradient, iterative inference models naturally account for the top-down information obtained from the data and bottom-up information obtained from the prior to estimate the approximate posterior distribution. This feature complies with the fundamental principle of the Bayes rule, in contrast to the standard inference models, which are purely bottom-up, by simply mapping the observed data to the approximate posterior. With the same objective of overcoming the limitations of standard amortized variational inference, the semi-amortized VAEs proposed by Kim *et al.* (2018) use a standard inference model (i.e., an encoder network) to provide an initial estimate of the approximate posterior parameters and then run stochastic variational inference (Hoffman *et al.*, 2013) to refine them.

Of particular relevance to DVAEs, the amortized variational filtering algorithm proposed by Marino *et al.* (2018b) generalizes iterative inference models (Marino *et al.*, 2018a) to a general class of dynamical latent variable models for sequential data processing. This algorithm is a general method for performing causal variational inference, using only past and present observed data. When combined with DNNs, the considered general class of dynamical latent variable models corresponds to the DVAE class. The proposed inference method is thus applicable to DVAEs, and the authors conducted experiments using VRNN and SRNN (among other models). An interesting feature of this method is its versatility. In the context of standard amortized variational inference, the DVAE inference model should be designed in accordance with the form of the DVAE generative model, following, for instance, the structure of the exact posterior distribution which can be identified using D-separation (see Chapter 4). In contrast, the

amortized variational filtering algorithm is agnostic of the form of the generative model. Another strength of this algorithm is inherited from iterative inference models (Marino *et al.*, 2018a), which combine information from both the data and the prior to compute the approximate posterior distribution parameters. This principle is also applicable in the context of the amortized variational filtering algorithm, where the "prior" (e.g., $p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$) and the approximate posterior (e.g., $q(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1})$) vary in time. The resulting amortized variational filtering algorithm thus resembles classical Bayesian inference filtering methods, such as a Kalman filter, where at a given time instant, the posterior distribution is computed by updating the predictive distribution (involving the prior distribution at the current time instant and the posterior distribution at the previous time instant) using the current observation, as discussed in Section 3.2.2.

14.3.3 Disentanglement of latent factors

A common crucial issue for VAEs and DVAEs is how to ensure the disentanglement of latent factors. As stated by Chen *et al.* (2017),

> "A key goal of representation learning is to identify and disentangle the underlying causal factors of the data, so that it becomes easier to understand the data, to classify it, or to perform other tasks."

Such disentanglement is not necessarily natural or efficient in the standard VAE; it somehow has to be "encouraged," either in the model design or in the training procedure (or both).

Semi-supervised VAEs

Siddharth *et al.* (2017) proposed forcing the disentanglement of $\mathbf{z}$ and thus improving its interpretability by using a small amount of supervision during training. This study does not particularly deal with static or dynamical VAEs, and this weak supervision principle can be applied to both.

Modification of the loss function

A second strategy to improve the disentanglement of latent factors is to modify the loss function (i.e., the VLB). In this line, Higgins *et al.* (2017) introduced a weighting factor, denoted β, to weight the regularization term in (2.21), so that the VLB becomes

$$\mathcal{L}(\theta, \phi, \beta; \mathbf{X}) = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})}\big[\log p_{\theta_\mathbf{x}}(\mathbf{X}|\mathbf{Z})\big] - \beta D_{KL}(q_\phi(\mathbf{Z}|\mathbf{X}) \,\|\, p_{\theta_\mathbf{z}}(\mathbf{Z})). \tag{14.1}$$

A value of β larger than 1 favors the KL term; hence, it encourages independence/disentanglement of the latent vector entries, although at the price of lower reconstruction/generation quality. For example, the images reconstructed with a β-VAE can be slightly blurred compared to those reconstructed with a standard VAE. However, the control of the properties of the objects represented in the image from the latent vectors is improved (Higgins *et al.*, 2017).

Chen *et al.* (2018) went a step ahead. Starting from the VLB (2.21), they introduced the *aggregated posterior* $q_\phi(\mathbf{z}) = \frac{1}{N_{tr}} \sum_{n=1}^{N_{tr}} q_\phi(\mathbf{z}|\mathbf{x}_n)$ and then decomposed the KL term of the VLB (summed over the training data) into a sum of three terms. The first one, referred to as the index-code mutual information, quantifies the mutual information between data and latent variables. The second one, referred to as the total correlation, is the KL divergence between the aggregated posterior and the product of its marginals (i.e., $D_{KL}(q_\phi(\mathbf{z}) \,\|\, \prod_{l=1}^{L} q_\phi(z_l))$). It quantifies the independence of the latent vector entries independently of data inputs (i.e., marginal independence as opposed to conditional independence). The third term is the sum over entries of the entry-wise KL divergence between the marginal aggregated posterior $q_\phi(z_l)$ and prior $p(z_l)$. The authors noted that minimizing the KL term of the VAE encourages the independence of the latent vector entries through minimization of the total correlation. However, it also penalizes the mutual information between the data and latent vectors, hence decreasing the power of latent components to explain the data. Therefore, they proposed applying a weighting factor (larger than 1) to the total correlation only, leaving the mutual information term unchanged. They experimentally demonstrated the advantage of this strategy over the

β-VAE. A similar idea was proposed independently by Kim and Mnih (2018), with a slightly different decomposition of the VLB KL term and a different implementation (based on an adversarial training of the model). For an extensive discussion and benchmark on disentangled representation learning with VAEs, see Locatello *et al.* (2020).

Such a general principle of enforcing latent factor disentanglement by modifying the loss function is independent of the issue of "static" against temporal modeling. It can thus, in principle, be applied to the DVAE framework. Finding a relevant decomposition of the loss function in the DVAE framework is still an open topic. Due to the more complex (temporal) intrications of the observed and latent variables, it is difficult to say if terms equivalent to total correlation or mutual information can be evidenced easily. Future studies should consider this aspect to make DVAE models more controllable and interpretable.

14.3.4 Hierarchical VAEs and DVAEs

A structured VAE, or hierarchical VAE, is a general subclass of VAEs where the latent space is structured by setting a hierarchical prior distribution on a set of latent variables $\mathbf{z} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_K\}$ (Kingma and Welling, 2019, Chapter 4) (Salimans, 2016; Sønderby *et al.*, 2016b; Sønderby *et al.*, 2016a). Here, the index denotes different latent variables, not a sample in a training set or a time index in a sequence as before. For example, a hierarchical "multilevel" VAE was proposed by Bouchacourt *et al.* (2018), with two latent vectors defined at different data scales: One latent vector encodes a common content for a group of data and the other latent vector encodes the style of subgroups of data within a group. In Bouchacourt *et al.*'s (2018) paper, data grouping involves a certain amount of supervision during training.

Another notable example of a hierarchical VAE was presented by (Salimans, 2016), who proposed to use a (deep) first-order autoregressive prior model:

$$p_{\theta_\mathbf{z}}(\mathbf{z}) = p_{\theta_\mathbf{z}}(\mathbf{z}_0) \prod_{k=1}^{K} p_{\theta_\mathbf{z}}(\mathbf{z}_k|\mathbf{z}_{k-1}). \tag{14.2}$$

In such an approach, the latent space is structured but not the data

space (we still have a unique observed vector $\mathbf{x}$). In this hierarchical latent variable model, there is no notion of time, but if we think of timely ordered variables, then we move toward Markov models (in the above example, a first-order one). In addition, if we consider both timely ordered latent variables and timely ordered observed variables, we step into the world of DVAEs. In other words, a DVAE can be considered a particular case of structured/hierarchical VAEs, with timely ordered latent and observed variables.

At this point, we can make an interesting parallel among the DVAE papers that we have reviewed and the hierarchical VAE papers (Salimans, 2016; Sønderby *et al.*, 2016b; Sønderby *et al.*, 2016a; Kingma and Welling, 2019) concerning the design of the inference model. We have seen in Section 4.2.1 that the D-separation methodology was not systematically used in the design of DVAE inference models. Interestingly and quite surprisingly, this methodology is not mentioned either in the above hierarchical VAE papers. Yet, we recall that D-separation is a major principled way to guide the design of inference models, including in this more general case. In these papers, the authors rather chose one model among different somewhat intuitive structures (e.g., top-down against bottom-up inference (Kingma and Welling, 2019)). One strategy to make the choice is to favor an inference model with variable dependencies that "mirror" those of the generative model, so that some "module" and parameters can be shared between them. This is a suitable feature that we have discussed for DVAEs in Section 4.2.3 and was applied in the VRNN model for example. In the hierarchical VAE papers, this "module sharing" strategy reportedly led to faster training and better fitting of model and data.

That being said, module sharing is not incompatible with respecting the exact posterior distribution structure. For example, in Salimans's (2016) paper, the inference model is defined by

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_0|\mathbf{x}) \prod_{k=1}^{K} q_\phi(\mathbf{z}_k|\mathbf{z}_{k-1}, \mathbf{x}). \tag{14.3}$$

Again, it was chosen by the author because it "mirrors" the generative model. This inference model follows the structure of the exact posterior distribution, even though this fundamental latter point was not men-

tioned by Salimans (2016). In the design of a DVAE, we can apply this principle: We can look for an inference model that both respects the structure of the exact posterior and shares some module(s) with the generative model.

In DVAEs, the problem of disentangling the factors of data variation takes a new flavor, as different factors of variation can have different dynamics. In this context, one way to address the disentanglement challenge is therefore to apply different levels of hierarchical modeling of the latent factors on the time dimension; that is, we can design models with a different time resolution for different latent variables, which is of course not incompatible with other types of hierarchical models. In particular, one general challenge is to separate the data dynamics (i.e., their temporal trajectories) and other factors of variations that are more constant over time (e.g., speaker identity for speech data, or objects present in the scene for videos).[2] For example, we have seen in this review the DSAE model (Li and Mandt, 2018) and the FHVAE model (Hsu *et al.*, 2017b), which include latent variables defined at the sequence level, segment level (subsequence of consecutive frames), or frame level. For speech signal modeling, this appears as a promising way to separate the modeling and control of phonetic information, which is defined at the segment or frame level, and speaker/session information, which is defined at the sequence level. A generalization of this approach would be to impose a prior distribution of $\mathbf{z}_{1:T}$ that fits the dynamics of the latent factors to extract, which can be significantly different from the data dynamics.

In general, the issue of separating data dynamics and other factors of variation is still largely open in the literature on DVAE models with a sequence of latent vectors. For example, we were surprised to notice that there are very few experiments and information available on the explainability of the extracted sequence of latent factors. Experiments involving swapping of the extracted latent factors across two data sequences before resynthesizing them were reported by, for example, Hsu *et al.* (2017b). These experiments show that for speech signals, speaker

[2]Several papers dealing with disentanglement and separate control of content and dynamics in videos have reported impressive results in an adversarial training framework (Denton and Birodkar, 2017; Villegas *et al.*, 2017; Tulyakov *et al.*, 2018).

identity can be exchanged between two sentences while preserving the same phonetic content, which is a very nice result. Yet, the issues of disentangling and controlling speech production factors separately remain largely open. Moreover, basic questions such as the impact of the size of $\mathbf{z}_t$ and $\mathbf{h}_t$ on modeling quality and the relevance of extracted latent factors have been poorly considered so far. For example, for speech processing, what happens if the size of $\mathbf{z}_t$ is reduced to a few entries, while the size of $\mathbf{h}_t$ is kept comparable to that of data $\mathbf{x}_t$?

Hsu *et al.* (2017b) indicated that

> "to the best of our knowledge, there has not been any attempt to learn disentangled and interpretable representations without supervision from sequential data."

Regarding SRNN (Fraccaro *et al.*, 2016), VRNN (Chung *et al.*, 2015), and SVAE (Johnson *et al.*, 2016), Hsu *et al.* (2017b) said

> "[...] it remains unclear whether independent attributes are disentangled in the latent space. Moreover, the learned latent variables in these models are not interpretable without manually inspecting or using labeled data."

Hence, the models such as STORN, VRNN, and SRNN provide an elegant and powerful mathematical and methodological framework for sequential data representation learning; however, there is still a lot of work to be done on the disentanglement challenge. Solutions for the disentanglement of $\mathbf{z}_t$ in DVAEs, inspired by or combined with existing structured or hierarchical models such as the ones presented by Salimans (2016) and Sønderby *et al.* (2016b) and Sønderby *et al.* (2016a), still have to be developed.

14.4 Perspectives on source coding

Although VAE and DVAE are excellent frameworks for extracting efficient and compact data representations, there are relatively few studies on their practical application to source coding (i.e., data compression including quantization and bitrate issues for data transmission or storage). We have mentioned above the problem of $\mathbf{z}$ "vanishing" or "being

ignored" when a powerful deterministic temporal encoder-decoder is used, and a few papers have related this problem to the need to better encode $\mathbf{z}$, in the source coding sense, with an information-theoretic interpretation of a VAE as a lossy coder (Kingma *et al.*, 2016; Chen *et al.*, 2017). Among the few papers on the practical application of (D)VAE to data coding, we can mention the ConvDRAW model proposed by Gregor *et al.* (2016), which learns and encodes a hierarchy of latent variables, resulting in an image lossy compression that performs similarly to JPEG. Other examples include VQ-VAE, which is a mix of VAE and vector quantization of $\mathbf{z}$, applied to speech coding (Oord *et al.*, 2017; Gârbacea *et al.*, 2019) and video compression with rate-distortion autoencoders (Habibian *et al.*, 2019).

As for a general approach to source coding based on DVAEs with a sequence of latent variables, we can mention two recent papers: Lombardo *et al.* (2019) and Yang *et al.* (2020). Lombardo *et al.* (2019) presented a video codec based on the DSAE model (Li and Mandt, 2018), which we reviewed in Chapter 11. The sequence of latent vectors $\mathbf{z}_{1:T}$ extracted by the DSAE encoder is quantized and transformed into a binary minimum-length sequence by an arithmetic coder, which exploits the DSAE dynamical model $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1})$ for entropy coding. The chaining of inverse operations (i.e., arithmetic decoding, inverse quantization and DSAE decoder) enables to obtain the decoded data sequence $\hat{\mathbf{x}}_{1:T}$. The global variable $\mathbf{v}$ (see Section 11.1) is encoded separately with a similar scheme. The resulting video codec is shown to exhibit rate-distortion performance that is comparable to the state-of-the art video codecs (such as VP9) on generic video sequences while drastically improving the performance on video sequences with specialized content (similar to the content of videos used to train the model). Lombardo *et al.* (2019) provide no information on the control of the coded data sequence quality or that of the bitrate. They only mention that the arithmetic encoding of $\mathbf{z}_{1:T}$ requires a number of iterations. Moreover, the DSAE model is an SSM-like model; that is, $\mathbf{x}_t$ is generated from $\mathbf{z}_t$ "alone," not considering the potential of using $\mathbf{x}_{t-1}$ (or its quantized version) for predicting $\mathbf{x}_t$ and encoding it more efficiently.

In contrast, Yang *et al.* (2020) proposed different schemes for encoding a data sequence $\mathbf{x}_{1:T}$ through the inference and quantization of

the corresponding sequence of latent vectors $\mathbf{z}_{1:T}$, with different options for recurrent connections. One of them, called feedback recurrent autoencoder (FRAE), has recurrent connections at both the encoder and decoder, and a feedback connection from the decoder to encoder. The recurrent connections are reminiscent of the *predictive coding* principle that is classical in source coding theory (Gersho and Gray, 2012). In fact, FRAE can be considered a *nonlinear predictive coding* scheme, in which the encoder forms a latent code that encodes only the residual information that is missing when reconstructing a data vector from the deterministic internal state, which depends on past data vectors. This concept of predictive coding is strongly related to that of the predictive mode for the DVAE models that we discussed in general terms in Section 4.1.1 and that we have seen implemented in different (autoregressive) DVAE models. Therefore, from this viewpoint, FRAE is strongly related to STORN, VRNN, and SRNN. The feedback connection from the decoder to encoder is reminiscent of another classical principle of source coding –*closed-loop coding* (Gersho and Gray, 2012)– even though Yang *et al.* (2020) did not refer to it explicitly. In short, closed-loop coding enables the decoder to use the quantized previous data vectors in place of the unquantized ones (not available at the decoder) for predicting the current data vector.

This line of research on nonlinear predictive coders based on DVAEs is quite promising and is only at its infancy. As Yang *et al.* (2020) wrote

> "There is no standard autoencoder architecture for temporally correlated data that has variable-length and long range dependencies such as video, speech, and text. The main challenge lies in the difficulty in capturing correlation information at different time-scales in an online/sequential fashion."

This agrees with the concluding remark of Chen *et al.* (2017):

> "We believe it's exciting to extend this principle of learning lossy codes [of the latent variable $\mathbf{z}$] to other forms of data, in particular those that have a temporal aspect like audio and video."

Acknowledgements

This work was partially funded by the Multidisciplinary Institute in Artificial Intelligence MIAI@Grenoble-Alpes (ANR-19-P3IA-0003), the ANR ML3RI project (ANR-19-CE33-0008-01), and the H2020 SPRING project (under GA #871245).

Appendices

A

Marginalization of $\mathbf{h}_{1:T}$ in STORN

In this appendix, we present how the internal state vector sequence $\mathbf{h}_{1:T}$ can be "marginalized" in a DVAE model formulation; that is, how we can express $\mathbf{h}_t$ as a deterministic function of the other random variables (and thus move from the developed form of the model to the compact form). This is presented for the STORN model, but a similar derivation can be obtained for the other models as well.

For conciseness, we replace here (7.1) with the generic notation $\mathbf{h}_t = f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1})$. From the dependencies between the different variables, represented by the graphical model in Figure 7.1, the joint distribution between all variables is given by

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{h}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t) p_\theta(\mathbf{h}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1}) p(\mathbf{z}_t). \quad \text{(A.1)}$$

As $\mathbf{h}_t$ is a deterministic function of $\mathbf{x}_{t-1}$, $\mathbf{z}_t$, and $\mathbf{h}_{t-1}$, its conditional density is a Dirac distribution with a mode given by $f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1})$:

$$p_\theta(\mathbf{h}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1}) = \delta(\mathbf{h}_t; f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1})). \quad \text{(A.2)}$$

Let us denote with $\mathbf{d}_{1:T}$ the sequence $\mathbf{h}_{1:T}$ considered a (deterministic) function of $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$ only; that is, at each time step, we have $\mathbf{d}_t = \mathbf{h}_t = \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, f_h(\mathbf{x}_{t-2}, \mathbf{z}_{t-1}, ...))$, with a recursive

injection of the recurrent terms into this latter expression up to the first term $f_h(\mathbf{x}_0, \mathbf{z}_1, \mathbf{h}_0)$. Marginalizing (A.1) with respect to $\mathbf{h}_{1:T}$ leads to

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \int_{\mathcal{R}^{H\times T}} \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t)\delta(\mathbf{h}_t; f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1}))p(\mathbf{z}_t)d\mathbf{h}_{1:T} \tag{A.3}$$

$$= \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{d}_t)p(\mathbf{z}_t). \tag{A.4}$$

To move from (A.3) to (A.4), one can start by marginalizing over $\mathbf{h}_T$, so that $\mathbf{h}_T$ is replaced with $f_h(\mathbf{x}_{T-1}, \mathbf{z}_T, \mathbf{h}_{T-1})$, and then marginalizing over $\mathbf{h}_{T-1}$, and so on. Hereinafter, for simplification of notations, we identify $\mathbf{d}_t$ with $\mathbf{h}_t$, but we must keep in mind that when doing so, we see $\mathbf{h}_t$ as a deterministic function of $\mathbf{x}_{1:t-1}$ and $\mathbf{z}_{1:t}$ with the recurrence being "unfolded," and not as a free random variable. Thus, we have

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t)p(\mathbf{z}_t) = \prod_{t=1}^{T} p_\theta\big(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\big)p(\mathbf{z}_t). \tag{A.5}$$

From the above equation and (7.7), we deduce the following conditional distribution:

$$p_\theta(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t) = \prod_{t=1}^{T} p_\theta\big(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\big). \tag{A.6}$$

We insist that, in the above equations, $\mathbf{h}_{1:T}$ is to be considered as the set of vectors $\{\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\}_{t=1}^{T}$ and not as a free random variable.

B

DVAE implementation with speech data

In this section, we provide the complete specifications of the DVAE models used in our experiments with the speech data. For each DVAE model, we will first present the generation network (decoder) and then the inference network (encoder). As many of the networks are MLPs, we define a notation to refer to these architectures concisely: MLP($\mathbf{y}$, n_1, f_1, $\ldots$, n_L, f_L) refers to an L-layer MLP with input $\mathbf{y}$, and n_ℓ and f_ℓ denote the output dimension and the (element-wise) activation function of the ℓ-th layer, respectively. The possible activation functions are ReLU, Sigmoid, hyperbolic tangent (Tanh), and linear $\mathbb{I}$. This latter activation function is always used for the last layer of the networks computing the parameters of the random variables $\mathbf{x}_t$ or $\mathbf{z}_t$, whether they are the output of an MLP or of an RNN. Therefore, it will not be made explicit for conciseness. For example, if we define the generative network for $\mathbf{z}_t$ as MLP($\mathbf{h}_t$, 64, Sigmoid, 32, Sigmoid), this means that we use an MLP with two hidden layers of dimension 64 and 32, respectively, both with Sigmoid activation function, and an output layer of dimension 2×16 (for mean and log-variance vectors, which are both the same size as $\mathbf{z}_t$) with linear activation.

B.1 DKF

Generation: Following Krishnan *et al.* (2017), we use a gated transition function to implement $d_{\mathbf{z}}$ in (3.9):

$$\boldsymbol{\nu}_t = \text{MLP}(\mathbf{z}_{t-1}, 16, \text{ReLU}, 16, \text{Sigmoid}) \tag{B.1}$$
$$\boldsymbol{\mu}_t^{\text{nonlin}} = \text{MLP}(\mathbf{z}_{t-1}, 16, \text{ReLU}, 16, \mathbb{I}) \tag{B.2}$$
$$\boldsymbol{\mu}_t^{\text{lin}} = \text{MLP}(\mathbf{z}_{t-1}, 16, \mathbb{I}) \tag{B.3}$$
$$\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}) = (1 - \boldsymbol{\nu}_t) \odot \boldsymbol{\mu}_t^{\text{lin}} + \boldsymbol{\nu}_t \odot \boldsymbol{\mu}_t^{\text{nonlin}} \tag{B.4}$$
$$\boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}) = \text{MLP}(\text{ReLU}(\boldsymbol{\mu}_{\text{t}}^{\text{nonlin}}), 16, \text{Softplus}), \tag{B.5}$$

where $\odot$ denotes element-wise multiplication. $\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1})$ is a gated combination of a linear and a nonlinear estimate of the mean vector. The nonlinear estimate is also used to compute the variance $\boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1})$.

As for the generation of $\mathbf{x}_t$, the function $d_{\mathbf{x}}(\mathbf{z}_t)$ in (3.11) is implemented with an MLP($\mathbf{z}_t$, 32, Tanh, 64, Tanh, 128, Tanh, 256, Tanh).

Inference: We implement the DKS inference model proposed by Krishnan *et al.* (2017), which follows equations (5.4)–(5.7), where (5.5) is the combiner function shown in Figure 13.1 (a). The function $e_{\overleftarrow{\mathbf{g}}}$ is implemented using a backward LSTM fed with an MLP($\mathbf{x}_t$, 256, Tanh). The affine function of $\mathbf{z}$ in (5.5) is implemented with a one-layer MLP($\mathbf{z}_{t-1}$, 32, Tanh), and the function $e_{\mathbf{z}}(\mathbf{g}_t)$ is implemented with an MLP($\mathbf{g}_t$, 32, Tanh).

B.2 STORN

Generation: We first recall that in STORN, the prior distribution of $\mathbf{z}_{1:T}$ is an i.i.d. standard Gaussian distribution. Therefore, $\mathbf{z}_{1:T}$ can be sampled without requiring a dedicated network. As for $\mathbf{h}_t$, the function $d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1})$ in (7.4) is implemented with the concatenation of an MLP($\mathbf{x}_{t-1}$, 256, Tanh) and an MLP($\mathbf{z}_t$, 32, Tanh, 64, Tanh) (which can both be considered feature extractors), followed by a forward LSTM network. As for $\mathbf{x}_t$, the function $d_{\mathbf{x}}(\mathbf{h}_t)$ in (7.5) is implemented with an MLP($\mathbf{h}_t$, 256, Tanh).

Inference: The function $e_{\mathbf{g}}(\mathbf{x}_t, \mathbf{g}_{t-1})$ in (7.15) is implemented with an MLP($\mathbf{x}_t$, 256, Tanh), followed by a forward LSTM network. The function $e_{\mathbf{z}}(\mathbf{g}_t)$ in (7.16) is implemented with an MLP($\mathbf{g}_t$, 64, Tanh, 32, Tanh).

B.3 VRNN

We recall that, unlike STORN, VRNN employs a shared RNN for inference and generation with an internal state vector $\mathbf{h}_t$. Furthermore, VRNN explicitly introduces feature extractors for $\mathbf{x}_t$ and $\mathbf{z}_t$ and uses the extracted features to feed the different encoder and decoder modules.

Feature extraction: $\varphi_{\mathbf{x}}(\mathbf{x}_t)$ is an MLP($\mathbf{x}_t$, 256, Tanh) and $\varphi_{\mathbf{z}}(\mathbf{z}_t)$ is an MLP($\mathbf{z}_t$, 32, Tanh, 64, Tanh).

Generation: The function $d_{\mathbf{h}}(\varphi_{\mathbf{x}}(\mathbf{x}_{t-1}), \varphi_{\mathbf{z}}(\mathbf{z}_{t-1}), \mathbf{h}_{t-1})$ in (8.1) is implemented with an LSTM network with input $[\varphi_{\mathbf{x}}(\mathbf{x}_{t-1}), \varphi_{\mathbf{z}}(\mathbf{z}_{t-1})]$. The function $d_{\mathbf{z}}(\mathbf{h}_t)$ in (8.4) is implemented by directly mapping $\mathbf{h}_t$ to the dimension of latent space (i.e. the output layer mentioned in the beginning of this subsection). The function $d_{\mathbf{x}}(\varphi_{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_t)$ in (8.2) is also implemented with an simple output layer.

Inference: The function $e_{\mathbf{z}}(\varphi_{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_t)$ in (8.11) is still implemented with an output layer where the input is the concatenation of $\varphi_{\mathbf{x}}(\mathbf{x}_t)$ and $\mathbf{h}_t$.

B.4 SRNN

We recall that, as with VRNN, SRNN shares an internal recurrent state vector $\mathbf{h}_t$ between the generation and inference models.

Generation: The function $d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{h}_{t-1})$ in (9.1) is implemented with an LSTM network with input MLP($\mathbf{x}_{t-1}$, 256, Tanh). The function $d_{\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{h}_t)$ in (9.4) is implemented with an MLP($[\mathbf{z}_{t-1}, \mathbf{h}_t]$, 64, Tanh, 32, Tanh). The function $d_{\mathbf{x}}(\mathbf{z}_t, \mathbf{h}_t)$ in (9.2) is implemented with an MLP($[\mathbf{z}_t, \mathbf{h}_t]$, 256, Tanh).

Inference: The function $e_{\overleftarrow{\mathbf{g}}}([\mathbf{h}_t, \mathbf{x}_t], \overleftarrow{\mathbf{g}}_{t+1})$ in (9.10) is a backward LSTM network with input MLP($[\mathbf{h}_t, \mathbf{x}_t]$, 256, Tanh). The function $e_{\mathbf{z}}(\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t)$ in (9.11) is an MLP($[\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t]$, 64, Tanh, 32, Tanh).

B.5 RVAE

As STORN, RVAE assumes an i.i.d. standard Gaussian prior for $\mathbf{z}_{1:T}$, so no network is required to generate $\mathbf{z}_t$. We recall that RVAE has causal and noncausal versions depending on whether the generation of $\mathbf{x}_t$ uses $\mathbf{z}_{t+1:T}$ or not, respectively.

Generation: Regarding the causal case, the function $d_{\mathbf{h}}(\mathbf{z}_t, \mathbf{h}_{t-1})$ in (10.4) is a forward LSTM network with input $\mathbf{z}_t$. The function $d_{\mathbf{x}}(\mathbf{h}_t)$ in (10.5) is an single output layer. For the noncausal case, the generation of $\mathbf{h}_t$ in (10.9)–(10.11) is implemented with a bidirectional LSTM with the same input as in the causal case, and $d_{\mathbf{x}}(\mathbf{h}_t)$ is also the same.

Inference: In the causal case, the function $e_{\overrightarrow{\mathbf{g}}}(\mathbf{z}_{t-1}, \overrightarrow{\mathbf{g}}_{t-1})$ in (10.18) is a forward LSTM network with input $\mathbf{z}_{t-1}$. The function $e_{\overleftarrow{\mathbf{g}}}(\mathbf{x}_t, \overleftarrow{\mathbf{g}}_{t+1})$ in (10.19) is a backward LSTM network with input $\mathbf{x}_t$. The function $e_{\mathbf{z}}(\mathbf{g}_t)$ in (10.21) is single output layer. In the noncausal case, the function $e_{\overrightarrow{\mathbf{g}}_{\mathbf{z}}}$ in (10.24) follows the same architecture as $e_{\overrightarrow{\mathbf{g}}}$ in the causal inference model, whereas the backward LSTM $e_{\overleftarrow{\mathbf{g}}_{\mathbf{x}}}$ is replaced with a bidirectional LSTM. The outputs are indicated as $e_{\overrightarrow{\mathbf{g}}_{\mathbf{x}}}$ and $e_{\overleftarrow{\mathbf{g}}_{\mathbf{x}}}$ in function (10.25) and (10.26).

B.6 DSAE

We recall that compared to the other models, DSAE has an extra sequence-level latent variable $\mathbf{v}$. We assume that $\mathbf{v}$ has the same dimension as $\mathbf{z}_t$. As the total dimension of $\mathbf{z}_{1:T}$ is T times that of $\mathbf{z}_t$, introducing this extra latent variable $\mathbf{v}$ will not change the total number of latent variables much. Therefore, we can still consider that it is fair to compare DSAE to the other models in such a configuration. The generation of $\mathbf{v}$ is not detailed in the original paper, and we assume it follows a standard Gaussian distribution.

Generation: The function $d_{\mathbf{h}}(\mathbf{z}_{t-1}, \mathbf{h}_{t-1})$ in (11.2) is a forward LSTM network with a hidden layer of dimension 128. The function $d_{\mathbf{z}}(\mathbf{h}_t)$ in (11.3) is a one-layer linear network to project the dimension of $\mathbf{h}_t$ onto the dimension of $\mathbf{z}_t$. The function $d_{\mathbf{x}}(\mathbf{z}_t, \mathbf{v})$ in (11.5) is an MLP($[\mathbf{z}_t, \mathbf{v}]$, 32, Tanh, 64, Tanh, 128, Tanh, 256, Tanh).

Inference: As for the inference of $\mathbf{v}$, Eqs. (11.12)–(11.15) are implemented with a bidirectional many-to-one LSTM network with input $\mathbf{x}_t$ and output $\mathbf{g}^{\mathbf{v}}$, followed by an single output layer. As for the inference of $\mathbf{z}_t$, Eqs. (11.17)–(11.19) are implemented with a bidirectional LSTM with input $[\mathbf{v}, \mathbf{x}_t]$. Finally, the function $e_{\mathbf{z}}(\mathbf{g}_t^{\mathbf{z}})$ in (11.20) is an RNN with a hidden layer of dimension 128.[1]

[1] In Figure 13.1 (f), we plot an MLP block after $\mathbf{x}_t$ and another one before the BRNN to output $\overrightarrow{\mathbf{g}}_t^{\mathbf{z}}$ and $\overleftarrow{\mathbf{g}}_t^{\mathbf{z}}$, whereas in the implementation we do not use them, or they can be considered as an identity layer. We made this choice because it provides better performance than applying several dense layers in our experiments, but we keep the possibility to use dense layers in our open-source code.

C

DVAE implementation with 3D human motion data

In this section, we provide the complete specifications of the DVAE models used in our experiments with the 3D human motion data. The notations have been defined in the previous section. We retain the general same architectures as for the speech data, because the inputs are still sequences of 1D vectors. Considering that the dimension of human pose vectors is smaller than that of speech vectors (96 vs 513), we reduce the dimension of the latent variable $\mathbf{z}_t$ (and also $\mathbf{v}$ in DSAE) to 10 and use more lightweight LSTM networks with hidden state of dimension 64.

C.1 DKF

Generation: The gated transition function for motion data is the same as for speech data, except that the hidden dimension for all MLP in (B.1) - (B.5) is reduced to 10. As for the generation of $\mathbf{x}_t$, the function $d_{\mathbf{x}}(\mathbf{z}_t)$ in (3.11) is implemented with an MLP($\mathbf{z}_t$, 32, Tanh, 64, Tanh).

Inference: Similar to the implementation for speech data, the function $e^{\leftarrow}_{\mathbf{g}}$ is implemented using a backward LSTM fed with an MLP($\mathbf{x}_t$, 64, Tanh), the affine function of $\mathbf{z}$ in (5.5) is replaced with a one-layer

MLP($\mathbf{z}_{t-1}$, 16, Tanh) and the function $e_{\mathbf{z}}(\mathbf{g}_t)$ is implemented with an MLP($\mathbf{g}_t$, 32, Tanh).

C.2 STORN

Generation: For $\mathbf{h}_t$, the function $d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1})$ in (7.4) is implemented with the concatenation of an MLP($\mathbf{x}_{t-1}$, 64, Tanh) and an MLP($\mathbf{z}_t$, 32, Tanh), followed by a forward LSTM network. As for $\mathbf{x}_t$, the function $d_{\mathbf{x}}(\mathbf{h}_t)$ in (7.5) is the output layer.

Inference: The function $e_{\mathbf{g}}(\mathbf{x}_t, \mathbf{g}_{t-1})$ in (7.15) is implemented with an MLP($\mathbf{x}_t$, 64, Tanh), followed by a forward LSTM network. The function $e_{\mathbf{z}}(\mathbf{g}_t)$ in (7.16) is implemented with an MLP($\mathbf{g}_t$, 32, Tanh).

C.3 VRNN

Feature extraction: $\varphi_{\mathbf{x}}(\mathbf{x}_t)$ is an MLP($\mathbf{x}_t$, 64, Tanh) and $\varphi_{\mathbf{z}}(\mathbf{z}_t)$ is an MLP($\mathbf{z}_t$, 16, Tanh, 32, Tanh).

Generation: Same to speech data, the function $d_{\mathbf{h}}(\varphi_{\mathbf{x}}(\mathbf{x}_{t-1}), \varphi_{\mathbf{z}}(\mathbf{z}_{t-1}), \mathbf{h}_{t-1})$ in (8.1) is implemented with an LSTM network with input $[\varphi_{\mathbf{x}}(\mathbf{x}_{t-1}), \varphi_{\mathbf{z}}(\mathbf{z}_{t-1})]$. The function $d_{\mathbf{z}}(\mathbf{h}_t)$ in (8.4) is implemented by directly mapping $\mathbf{h}_t$ to the dimension of latent space (i.e. the output layer mentioned in the beginning of this subsection). The function $d_{\mathbf{x}}(\varphi_{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_t)$ in (8.2) is also implemented with an simple output layer.

Inference: Same to speech data, the function $e_{\mathbf{z}}(\varphi_{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_t)$ in (8.11) is still implemented with an output layer where the input is the concatenation of $\varphi_{\mathbf{x}}(\mathbf{x}_t)$ and $\mathbf{h}_t$.

C.4 SRNN

Generation: The function $d_{\mathbf{h}}(\mathbf{x}_{t-1}, \mathbf{h}_{t-1})$ in (9.1) is implemented with an LSTM network with input MLP($\mathbf{x}_{t-1}$, 64, Tanh). The function $d_{\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{h}_t)$ in (9.4) is implemented with an MLP($[\mathbf{z}_{t-1}, \mathbf{h}_t]$, 32, Tanh). The function $d_{\mathbf{x}}(\mathbf{z}_t, \mathbf{h}_t)$ in (9.2) is implemented with an MLP($[\mathbf{z}_t, \mathbf{h}_t]$, 64, Tanh).

Inference: The function $e_{\overleftarrow{\mathbf{g}}}([\mathbf{h}_t, \mathbf{x}_t], \overleftarrow{\mathbf{g}}_{t+1})$ in (9.10) is a backward LSTM network with input MLP($[\mathbf{h}_t, \mathbf{x}_t]$, 64, Tanh). The function $e_{\mathbf{z}}(\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t)$ in (9.11) is an MLP($[\mathbf{z}_{t-1}, \overleftarrow{\mathbf{g}}_t]$, 32, Tanh).

C.5 RVAE

Generation: The implementation is the same as for speech data for both the causal case and the noncausal case, except the different dimension of the LSTM hidden state vector.

Inference: The implementation is the same as for speech data, except the different dimension of the LSTM hidden state vector.

C.6 DSAE

Generation: The function $d_{\mathbf{x}}(\mathbf{z}_t, \mathbf{v})$ in (11.5) is an MLP($[\mathbf{z}_t, \mathbf{v}]$, 32, Tanh, 64, Tanh). The others are the same as for speech data, except different hidden dimension of LSTM.

Inference: Same as for speech data, except different hidden dimension of LSTM.

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.* (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". *arXiv preprint arXiv:1603.04467*.

Aksan, E. and O. Hilliges. (2019). "STCN: Stochastic temporal convolutional networks". In: *International Conference on Learning Representations (ICLR)*. New Orleans, LA.

Andrychowicz, M., M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. (2016). "Learning to learn by gradient descent by gradient descent". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Archer, E., I. M. Park, L. Buesing, J. Cunningham, and L. Paninski. (2015). "Black box variational inference for state space models". *arXiv preprint arXiv:1511.07367*.

Babaeizadeh, M., C. Finn, D. Erhan, R. H. Campbell, and S. Levine. (2018). "Stochastic variational video prediction". In: *International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Bai, S., Z. Kolter, and V. Koltun. (2018). "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". *arXiv preprint arXiv:1803.01271*.

Bando, Y., M. Mimura, K. Itoyama, K. Yoshii, and T. Kawahara. (2018). "Statistical speech enhancement based on probabilistic integration of variational autoencoder and non-negative matrix factorization". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, Canada.

Bayer, J. and C. Osendorfer. (2014). "Learning stochastic recurrent networks". *arXiv preprint arXiv:1411.7610*.

Bengio, S., O. Vinyals, N. Jaitly, and N. Shazeer. (2015). "Scheduled sampling for sequence prediction with recurrent neural networks". *arXiv preprint arXiv:1506.03099*.

Bie, X., S. Leglaive, X. Alameda-Pineda, and L. Girin. (2021). "Unsupervised speech enhancement using dynamical variational autoencoders". *arXiv preprint arXiv:2106.12271*.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bitton, A., P. Esling, and T. Harada. (2020). "Neural granular sound synthesis". *arXiv preprint arXiv:2008.01393*.

Blaauw, M. and J. Bonada. (2016). "Modeling and transforming speech using variational autoencoders". In: *Conference of the International Speech Communication Association (Interspeech)*. San Francisco, CA.

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe. (2017). "Variational inference: A review for statisticians". *Journal of the American Statistical Association*. 112(518): 859–877.

Bogo, F., A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. (2016). "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image". In: *European Conference on Computer Vision (ECCV)*. Amsterdam, The Netherlands.

Bottou, L. (2004). "Stochastic Learning". In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia*. 146–168.

Bouchacourt, D., R. Tomioka, and S. Nowozin. (2018). "Multi-level variational autoencoder: Learning disentangled representations from grouped observations". In: *AAAI Conference on Artificial Intelligence*. New Orleans, LA.

Boulanger-Lewandowski, N., Y. Bengio, P. Vincent, P. Gray, and C. Naguri. (2012). "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription". In: *International Conference on Machine Learning (ICML)*. Edinburgh, Scotland.

Bowman, S. R., L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. (2016). "Generating sentences from a continuous space". *International Conference on Computational Natural Language Learning (CoNLL)*.

Chen, R. T., X. Li, R. Grosse, and D. Duvenaud. (2018). "Isolating sources of disentanglement in variational autoencoders". *arXiv preprint arXiv:1802.04942*.

Chen, X., D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. (2017). "Variational lossy autoencoder". In: *International Conference on Learning Representations (ICLR)*. Toulon, France.

Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". *arXiv preprint arXiv:1406.1078*.

Chung, J., K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio. (2015). "A recurrent latent variable model for sequential data". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Canada.

Cremer, C., X. Li, and D. Duvenaud. (2018). "Inference suboptimality in variational autoencoders". In: *International Conference on Machine Learning (ICML)*. Stockholm, Sweden.

Dai, B., Z. Wang, and D. Wipf. (2020). "The usual suspects? Reassessing blame for VAE posterior collapse". In: *International Conference on Machine Learning (ICML)*. Virtual online.

Daum, F. (2005). "Nonlinear filters: Beyond the Kalman filter". *IEEE Aerospace and Electronic Systems Magazine*. 20(8): 57–69.

Dayan, P., G. E. Hinton, R. M. Neal, and R. S. Zemel. (1995). "The Helmholtz machine". *Neural Computation*. 7(5): 889–904.

Dempster, A. P., N. M. Laird, and D. B. Rubin. (1977). "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the Royal Statistical Society. Series B (Methodological)*. 39(1): 1–38.

Deng, Z., R. Navarathna, P. Carr, S. Mandt, Y. Yue, I. Matthews, and G. Mori. (2017). "Factorized variational autoencoders for modeling audience reactions to movies". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HW.

Denton, E. and V. Birodkar. (2017). "Unsupervised learning of disentangled representations from video". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Denton, E. and R. Fergus. (2018). "Stochastic video generation with a learned prior". In: *International Conference on Machine Learning (ICML)*. Stockholm, Sweden.

Diggle, P. J. and R. J. Gratton. (1984). "Monte Carlo methods of inference for implicit statistical models". *Journal of the Royal Statistical Society: Series B (Methodological)*. 46(2): 193–212.

Durbin, J. and S. J. Koopman. (2012). *Time series analysis by state space methods*. Oxford University Press.

Esling, P., A. Chemla-Romeu-Santos, and A. Bitton. (2018). "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre space". In: *International Society for Music Information Retrieval Conference (ISMIR)*. Paris, France.

Fabius, O. and J. R. van Amersfoort. (2014). "Variational recurrent auto-encoders". *arXiv preprint arXiv:1412.6581*.

Finn, C., I. Goodfellow, and S. Levine. (2016). "Unsupervised learning for physical interaction through video prediction". *Advances in Neural Information Processing Systems (NeurIPS)*.

Fortuin, V., D. Baranchuk, G. Rätsch, and S. Mandt. (2020). "GP-VAE: Deep probabilistic time series imputation". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. Palermo, Italy.

Fox, E., E. B. Sudderth, M. I. Jordan, and A. S. Willsky. (2011). "Bayesian nonparametric inference of switching dynamic linear models". *IEEE Transactions on Signal Processing*. 59(4): 1569–1585.

Fraccaro, M., S. K. Sønderby, U. Paquet, and O. Winther. (2016). "Sequential neural models with stochastic layers". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Fraccaro, M., S. Kamronn, U. Paquet, and O. Winther. (2017). "A disentangled recognition and nonlinear dynamics model for unsupervised learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Frey, B. J. (1998). *Graphical models for machine learning and digital communication*. Ed. by T. G. Dietterich. Cambridge, MA: MIT Press.

Gan, Z., C. Li, R. Henao, D. E. Carlson, and L. Carin. (2015). "Deep temporal sigmoid belief networks for sequence modeling". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Canada.

Gârbacea, C., A. van den Oord, Y. Li, F. S. Lim, A. Luebs, O. Vinyals, and T. C. Walters. (2019). "Low bit-rate speech coding with VQ-VAE and a WaveNet decoder". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK.

Garofolo, J., D. Graff, D. Paul, and D. Pallett. (1993). "CSR-I (WSJ0) Sennheiser LDC93S6B. https://catalog.ldc.upenn.edu/LDC93S6B". *Philadelphia: Linguistic Data Consortium*.

Geiger, D., T. Verma, and J. Pearl. (1990). "Identifying independence in Bayesian networks". *Networks*. 20(5): 507–534.

Gersho, A. and R. M. Gray. (2012). *Vector quantization and signal compression*. Springer Science & Business Media.

Girin, L., F. Roche, T. Hueber, and S. Leglaive. (2019). "Notes on the use of variational autoencoders for speech and audio spectrogram modeling". In: *Digital Audio Effects Conference (DAFx)*. Birmingham, UK.

Goodfellow, I., Y. Bengio, and A. Courville. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. (2014). "Generative adversarial nets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Canada.

Goodfellow, I. (2016). "NIPS 2016 tutorial: Generative adversarial networks". *arXiv preprint arXiv:1701.00160*.

Goyal, A., A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. (2017). "Z-forcing: Training stochastic recurrent networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Graves, A. (2013). "Generating sequences with recurrent neural networks". *arXiv preprint arXiv:1308.0850*.

Graves, A., A.-r. Mohamed, and G. Hinton. (2013). "Speech recognition with deep recurrent neural networks". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada.

Gregor, K., F. Besse, D. J. Rezende, I. Danihelka, and D. Wierstra. (2016). "Towards conceptual compression". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Gregor, K., I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. (2015). "DRAW: A recurrent neural network for image generation". In: *International Conference on Machine Learning (ICML)*. Lille, France.

Gu, S., Z. Ghahramani, and R. E. Turner. (2015). "Neural adaptive sequential Monte Carlo". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Canada.

Gulrajani, I., K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville. (2016). "PixelVAE: A latent variable model for natural images". *arXiv preprint arXiv:1611.05013*.

Habibian, A., T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen. (2019). "Video compression with rate-distortion autoencoders". In: *IEEE International Conference on Computer Vision (ICCV)*. Seoul, Korea.

Hafner, D., T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. (2018). "Learning latent dynamics for planning from pixels". *arXiv preprint arXiv:1811.04551*.

Hamilton, J. D. (2020). *Time series analysis*. Princeton University Press.

Haykin, S. (2004). *Kalman filtering and neural networks*. John Wiley & Sons.

He, J., D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick. (2018). "Lagging inference networks and posterior collapse in variational autoencoders". In: *International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Higgins, I., L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. (2017). "β-VAE: learning basic visual concepts with a constrained variational framework". In: *International Conference on Learning Representations (ICLR)*. Toulon, France.

Hinton, G. and R. Salakhutdinov. (2006). "Reducing the dimensionality of data with neural networks". *Science*. 313(5786): 504–507.

Hinton, G., P. Dayan, B. Frey, and R. Neal. (1995). "The "Wake-Sleep" algorithm for unsupervised neural networks". *Science*. 268(5214): 1158–1161.

Hochreiter, S. and J. Schmidhuber. (1997). "Long short-term memory". *Neural Computation*. 9(8): 1735–1780.

Hoffman, M. D., D. M. Blei, C. Wang, and J. Paisley. (2013). "Stochastic variational inference". *Journal of Machine Learning Research*. 14(1): 1303–1347.

Honkela, A., T. Raiko, M. Kuusela, M. Tornio, and J. Karhunen. (2010). "Approximate Riemannian conjugate gradient learning for fixed-form variational Bayes". *Journal of Machine Learning Research*. 11(Nov): 3235–3268.

Hsu, W.-N., Y. Zhang, and J. Glass. (2017a). "Learning latent representations for speech generation and transformation". *arXiv preprint arXiv:1704.04222*.

Hsu, W.-N., Y. Zhang, and J. Glass. (2017b). "Unsupervised learning of disentangled and interpretable representations from sequential data". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Hu, Z., Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. (2017). "Toward controlled generation of text". In: *International Conference on Machine Learning (ICML)*. Sydney, Australia.

Huang, K., N. D. Sidiropoulos, and A. P. Liavas. (2016). "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization". *IEEE Transactions on Signal Processing*. 64(19): 5052–5065.

Ionescu, C., D. Papava, V. Olaru, and C. Sminchisescu. (2014). "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 36(7): 1325–1339.

Jang, M., S. Seo, and P. Kang. (2019). "Recurrent neural network-based semantic variational autoencoder for sequence-to-sequence learning". *Information Sciences*. 490: 59–73.

Jiang, J., G. Xia, D. Carlton, C. Anderson, and R. Miyakawa. (2020). "Transformer VAE: A hierarchical model for structure-aware and interpretable music representation learning". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Virtual Barcelona.

Johnson, M. J., D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. (2016). "Composing graphical models with neural networks for structured representations and fast inference". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. (1999). "An introduction to variational methods for graphical models". *Machine Learning*. 37: 183–233.

Kameoka, H., L. Li, S. Inoue, and S. Makino. (2018). "Semi-blind source separation with multichannel variational autoencoder". *arXiv preprint arXiv:1808.00892*.

Karl, M., M. Soelch, J. Bayer, and P. van der Smagt. (2017). "Deep variational Bayes filters: Unsupervised learning of state space models from raw data". In: *International Conference on Learning Representations (ICLR)*. Toulon, France.

Kim, H. and A. Mnih. (2018). "Disentangling by factorising". In: *International Conference on Machine Learning (ICML)*. Stockholm, Sweden.

Kim, Y., S. Wiseman, A. Miller, D. Sontag, and A. Rush. (2018). "Semi-amortized variational autoencoders". In: *International Conference on Machine Learning (ICML)*. Stockholm, Sweden.

Kingma, D. P. and J. Ba. (2014). "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and M. Welling. (2014). "Auto-encoding variational Bayes". In: *International Conference on Learning Representations (ICLR)*. Banff, Canada.

Kingma, D. P. and M. Welling. (2019). "An introduction to variational autoencoders". *Foundations and Trends in Machine Learning*. 12(4): 307–392.

Kingma, D. P., T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. (2016). "Improved variational inference with inverse autoregressive flow". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Koller, D. and N. Friedman. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

Krishnan, R., U. Shalit, and D. Sontag. (2015). "Deep Kalman filters". In: *arXiv preprint arXiv:1511.05121*.

Krishnan, R., U. Shalit, and D. Sontag. (2017). "Structured inference networks for nonlinear state space models". In: *AAAI Conference on Artificial Intelligence*. San Francisco, CA.

Krishnan, R., D. Liang, and M. Hoffman. (2018). "On the challenges of learning with inference networks on sparse, high-dimensional data". In: *International Conference on Artificial Intelligence and Statistics*. Lanzarote, Spain.

Kuleshov, V., A. Chaganty, and P. Liang. (2015). "Tensor factorization via matrix factorization". In: *Artificial Intelligence and Statistics*. 507–516.

Le Roux, J., S. Wisdom, H. Erdogan, and J. R. Hershey. (2019). "SDR: Half-baked or well done?" In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK.

Le, T. A., M. Igl, T. Rainforth, T. Jin, and F. Wood. (2018). "Auto-encoding sequential Monte Carlo". In: *International Conference on Learning Representations (ICLR)*. Vancouver, Canada.

Lea, C., R. Vidal, A. Reiter, and G. D. Hager. (2016). "Temporal convolutional networks: A unified approach to action segmentation". In: *European Conference on Computer Vision (ECCV)*. Amsterdam, The Netherlands.

Lee, J. Y., S. J. Cheon, B. J. Choi, N. S. Kim, and E. Song. (2018). "Acoustic modeling using adversarially trained variational recurrent neural network for speech synthesis". In: *Conference of the International Speech Communication Association (Interspeech)*. Hyderabad, India.

Leglaive, S., L. Girin, and R. Horaud. (2018). "A variance modeling framework based on variational autoencoders for speech enhancement". In: *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. Aalborg, Denmark.

Leglaive, S., L. Girin, and R. Horaud. (2019). "Semi-supervised multichannel speech enhancement with variational autoencoders and non-negative matrix factorization". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, UK.

Leglaive, S., X. Alameda-Pineda, L. Girin, and R. Horaud. (2020). "A recurrent variational autoencoder for speech enhancement". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Virtual Barcelona.

Li, X., L. Girin, and R. Horaud. (2017). "An EM algorithm for audio source separation based on the convolutive transfer function". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. New Paltz, NJ.

Li, Y. and S. Mandt. (2018). "Disentangled sequential autoencoder". In: *International Conference on Machine Learning (ICML)*. Stockholm, Sweden.

Linderman, S. W., A. C. Miller, R. P. Adams, D. M. Blei, L. Paninski, and M. J. Johnson. (2016). "Recurrent switching linear dynamical systems". *arXiv preprint arXiv:1610.08466*.

Liu, D. and G. Liu. (2019). "A Transformer-based variational autoencoder for sentence generation". In: *IEEE International Joint Conference on Neural Networks (IJCNN)*. Budapest, Hungary.

Locatello, F., S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem. (2020). "A sober look at the unsupervised learning of disentangled representations and their evaluation". *Journal of Machine Learning Research*. 21: 1–62.

Lombardo, S., J. Han, C. Schroers, and S. Mandt. (2019). "Deep generative video compression". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vancouver, Canada.

Lucas, J., G. Tucker, R. B. Grosse, and M. Norouzi. (2019). "Don't blame the ELBO! a linear VAE perspective on posterior collapse". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vancouver, Canada.

Lucas, T. and J. Verbeek. (2018). "Auxiliary guided autoregressive variational autoencoders". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Dublin, Ireland.

Luo, Y. and N. Mesgarani. (2019). "Conv-Tasnet: Surpassing ideal time-frequency magnitude masking for speech separation". *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 27(8): 1256–1266.

Maaløe, L., C. K. Sønderby, S. K. Sønderby, and O. Winther. (2016). "Auxiliary deep generative models". In: *International Conference on Machine Learning (ICML)*. New York, NY.

Maddison, C. J., J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. (2017). "Filtering variational objectives". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Mao, W., M. Liu, and M. Salzmann. (2020). "History repeats itself: Human motion prediction via motion attention". In: *European Conference on Computer Vision (ECCV)*. Virtual online.

Marino, J., Y. Yue, and S. Mandt. (2018a). "Iterative amortized inference". In: *International Conference on Machine Learning (ICML)*. Stockholm, Sweden.

Marino, J., M. Cvitkovic, and Y. Yue. (2018b). "A general method for amortizing variational filtering". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Canada.

Martinez, J., M. J. Black, and J. Romero. (2017). "On human motion prediction using recurrent neural networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HW.

Mathieu, M. F., J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. (2016). "Disentangling factors of variation in deep representation using adversarial training". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Miao, Y., L. Yu, and P. Blunsom. (2016). "Neural variational inference for text processing". In: *International Conference on Machine Learning (ICML)*. New York, NY.

Miladinović, D., M. W. Gondal, B. Schölkopf, J. M. Buhmann, and S. Bauer. (2019). "Disentangled state space representations". *arXiv preprint arXiv:1906.03255*.

Mnih, A. and K. Gregor. (2014). "Neural variational inference and learning in belief networks". *arXiv preprint arXiv:1402.0030*.

Moreno, V. M. and A. Pigazo. (2009). *Kalman filter: Recent advances and applications*. BoD–Books on Demand.

Murphy, K. P. (1998). "Switching Kalman filters". *Unpublished technical report – Available online*.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

Naesseth, C., S. Linderman, R. Ranganath, and D. Blei. (2018). "Variational sequential Monte Carlo". In: *International Conference on Artificial Intelligence and Statistics*. Lanzarote, Spain.

Neal, R. M. and G. E. Hinton. (1998). "A view of the EM algorithm that justifies incremental, sparse, and other variants". In: *Learning in graphical models*. Springer. 355–368.

Neeser, F. D. and J. L. Massey. (1993). "Proper complex random processes with applications to information theory". *IEEE Transactions on Information Theory*. 39(4): 1293–1302.

Oord, A. van den, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. (2016a). "Wavenet: A generative model for raw audio". *arXiv preprint arXiv:1609.03499*.

Oord, A. van den, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.* (2016b). "Conditional image generation with PixelCNN decoders". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Oord, A. van den, N. Kalchbrenner, and K. Kavukcuoglu. (2016c). "Pixel recurrent neural networks". *International Conference on Machine Learning (ICML)*.

Oord, A. van den, O. Vinyals, and K. Kavukcuoglu. (2017). "Neural discrete representation learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Papoulis, A. (1977). *Signal analysis*. McGraw-Hill New York.

Pariente, M., A. Deleforge, and E. Vincent. (2019). "A statistically principled and computationally efficient approach to speech enhancement using variational autoencoders". *arXiv preprint arXiv:1905.01209*.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. (2019). "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vancouver, Canada.

Pereira, J. and M. Silveira. (2018). "Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention". In: *IEEE International Conference on Machine Learning and Applications (ICMLA)*. Orlando, FL.

Petrovich, M., M. J. Black, and G. Varol. (2021). "Action-conditioned 3D human motion synthesis with Transformer VAE". In: *International Conference on Computer Vision (ICCV)*. Virtual online.

Raiko, T. and M. Tornio. (2009). "Variational Bayesian learning of nonlinear hidden state-space models for model predictive control". *Neurocomputing*. 72(16-18): 3704–3712.

Ranganath, R., D. Tran, and D. Blei. (2016). "Hierarchical variational models". In: *International Conference on Machine Learning (ICML)*. New York, NY.

Razavi, A., A. van den Oord, B. Poole, and O. Vinyals. (2019). "Preventing posterior collapse with delta-VAEs". *arXiv preprint arXiv:1901.03416*.

Rezende, D. J., S. Mohamed, and D. Wierstra. (2014). "Stochastic backpropagation and approximate inference in deep generative models". In: *International Conference on Machine Learning (ICML)*. Beijing, China.

Rezende, D. J. and S. Mohamed. (2015). "Variational inference with normalizing flows". *arXiv preprint arXiv:1505.05770*.

Rix, A., J. Beerends, M. Hollier, and A. Hekstra. (2001). "Perceptual evaluation of speech quality (PESQ): A new method for speech quality assessment of telephone networks and codecs". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Salt Lake City, Utah.

Robbins, H. and S. Monro. (1951). "A stochastic approximation method". *The Annals of Mathematical Statistics*: 400–407.

Roberts, A., J. Engel, C. Raffel, C. Hawthorne, and D. Eck. (2018). "A hierarchical latent vector model for learning long-term structure in music". *arXiv preprint arXiv:1803.05428*.

Roche, F., T. Hueber, S. Limier, and L. Girin. (2019). "Autoencoders for music sound synthesis: A comparison of linear, shallow, deep and variational models". In: *Sound and Music Conference (SMC)*. Malaga, Spain.

Salimans, T. (2016). "A structured variational auto-encoder for learning deep hierarchies of sparse features". *arXiv preprint arXiv:1602.08734*.

Salimans, T., D. Kingma, and M. Welling. (2015). "Markov chain Monte Carlo and variational inference: Bridging the gap". In: *International Conference on Machine Learning (ICML)*. Lille, France.

Salimans, T. and D. A. Knowles. (2013). "Fixed-form variational posterior approximation through stochastic linear regression". *Bayesian Analysis*. 8(4): 837–882.

Saul, L. K. and M. I. Jordan. (1996). "Exploiting tractable substructures in intractable networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Denver, CO.

Semeniuta, S., A. Severyn, and E. Barth. (2017). "A hybrid convolutional variational autoencoder for text generation". *arXiv preprint arXiv:1702.02390*.

Serban, I. V., A. G. Ororbia, J. Pineau, and A. Courville. (2016). "Piecewise latent variables for neural variational text processing". *arXiv preprint arXiv:1612.00377*.

Serban, I. V., A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. (2017). "A hierarchical latent variable encoder-decoder model for generating dialogues". In: *AAAI Conference on Artificial Intelligence*. San Francisco, CA.

Shang, W., K. Sohn, and Y. Tian. (2018). "Channel-recurrent autoencoding for image modeling". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. Lake Tahoe, NV.

Siddharth, N., B. Paige, J.-W. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. (2017). "Learning disentangled representations with semi-supervised deep generative models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Šmídl, V. and A. Quinn. (2006). *The variational Bayes method in signal processing*. Springer Science & Business Media.

Sohn, K., H. Lee, and X. Yan. (2015). "Learning structured output representation using deep conditional generative models". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Canada.

Sønderby, C. K., T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. (2016a). "How to train deep variational autoencoders and probabilistic ladder networks". In: *International Conference on Machine Learning (ICML)*. New York, NY.

Sønderby, C. K., T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. (2016b). "Ladder variational autoencoders". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Barcelona, Spain.

Su, J., S. Wu, D. Xiong, Y. Lu, X. Han, and B. Zhang. (2018). "Variational recurrent neural machine translation". In: *AAAI International Conference on Artificial Intelligence*. New Orleans, NV.

Sutskever, I. (2013). *Training recurrent neural networks*. University of Toronto.

Taal, C. H., R. C. Hendriks, R. Heusdens, and J. Jensen. (2011). "An algorithm for intelligibility prediction of time-frequency weighted noisy speech". *IEEE Transactions on Audio, Speech, and Language Processing*. 19(7): 2125–2136.

Tipping, M. E. and C. M. Bishop. (1999). "Probabilistic principal component analysis". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 61(3): 611–622.

Tulyakov, S., M.-Y. Liu, X. Yang, and J. Kautz. (2018). "MoCoGan: Decomposing motion and content for video generation". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, Utah.

Vahdat, A. and J. Kautz. (2020). "NVAE: A deep hierarchical variational autoencoder". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Virtual online.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Long Beach, CA.

Villegas, R., J. Yang, S. Hong, X. Lin, and H. Lee. (2017). "Decomposing motion and content for natural video sequence prediction". In: *International Conference on Learning Representations (ICLR)*. Toulon, France.

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". *Journal of Machine Learning Research*. 11: 3371–3408.

Wan, E. and R. Van Der Merwe. (2000). "The unscented Kalman filter for nonlinear estimation". In: *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*. Lake Louise, Canada.

Wang, T. and X. Wan. (2019). "T-CVAE: Transformer-based conditioned variational autoencoder for story completion". In: *AAAI International Joint Conference on Artificial Intelligence (IJCAI)*. Macao, China.

Watter, M., J. Springenberg, J. Boedecker, and M. Riedmiller. (2015). "Embed to control: A locally linear latent dynamics model for control from raw images". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Montréal, Canada.

Wei, G. C. and M. A. Tanner. (1990). "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms". *Journal of the American Statistical Association*. 85(411): 699–704.

Williams, C. and C. E. Rasmussen. (2006). *Gaussian processes for machine learning*. MIT Press.

Williams, R. J. and D. Zipser. (1989). "A learning algorithm for continually running fully recurrent neural networks". *Neural Computation*. 1(2): 270–280.

Winn, J. and C. M. Bishop. (2005). "Variational message passing". *Journal of Machine Learning Research*. 6(Apr): 661–694.

Yang, Y., G. Sautière, J. J. Ryu, and T. S. Cohen. (2020). "Feedback recurrent autoencoder". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Virtual Barcelona.

Yang, Z., Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. (2017). "Improved variational autoencoders for text modeling using dilated convolutions". In: *International Conference on Machine Learning (ICML)*. Sydney, Australia.

Yeung, S., A. Kannan, Y. Dauphin, and L. Fei-Fei. (2017). "Tackling over-pruning in variational autoencoders". In: *International Conference on Machine Learning – Workshop on Principled Approaches to Deep Learning*. Sydney, Australia.

Zhao, T., K. Lee, and M. Eskenazi. (2018). "Unsupervised discrete sentence representation learning for interpretable neural dialog generation". *arXiv preprint arXiv:1804.08069*.

Zhao, T., R. Zhao, and M. Eskenazi. (2017). "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders". *arXiv preprint arXiv:1703.10960*.

Zhao, Y., J. Nassar, I. Jordan, M. Bugallo, and I. M. Park. (2019). "Streaming variational Monte Carlo". *arXiv preprint arXiv:1906.01549*.

Zhao, Y. and I. M. Park. (2017). "Variational online learning of neural dynamics". *arXiv preprint arXiv:1707.09049*.