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, 38000 Grenoble, France.
[2] CentraleSupélec, IETR, 35576 Cesson-Sévigné, France.
[3] Inria, Univ. Grenoble Alpes, CNRS, LJK, 38000 Grenoble, France.
[4] Univ. Grenoble Alpes, CNRS, LPNC, 38000 Grenoble, France.

Abstract

The Variational Autoencoder (VAE) is a powerful deep generative model that is now extensively used to represent high-dimensional complex data via a low-dimensional latent space learned in an unsupervised manner. In the original VAE model, input data vectors are processed independently. In recent years, a series of papers have presented different extensions of the VAE to process sequential data, that not only model the latent space, but also model 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 an extensive literature review of these models. Importantly, we introduce and discuss a general class of models called Dynamical Variational Autoencoders (DVAEs) that encompasses a large subset of these temporal VAE extensions. Then we present in detail seven different instances of DVAE that were recently proposed in the literature, with an effort to homogenize the notations and presentation lines, as well as to relate these models with existing classical temporal models. We reimplemented those seven DVAE models and we present the results of an experimental benchmark conducted on the speech analysis-resynthesis task (the PyTorch code is made publicly available). The paper is concluded with an extensive discussion on important issues concerning the DVAE class of models and future research guidelines.

Contents

Chapter 1

Introduction

Deep Generative Models (DGMs) are a large family of probabilistic models that are currently of high interest in the machine learning and the signal processing scientific communities. Basically, DGMs result from the combination of conventional generative probabilistic models[1] and Deep Neural Networks (DNNs). For both conventional and deep generative models, different non-conflicting taxonomies can be established, due to the richness of the domain and percolation across the different approaches. Nevertheless, those models can be grossly classified in the following two categories. The first category groups models with an explicit formulation of a model of the data probability density function (pdf). The second category groups models that can generate data "directly," without using an explicit formulation and manipulation of a pdf model. Generative Adversarial Networks (GANs) are a now very popular example of this second category (Goodfellow et al., 2014, 2016; Goodfellow, 2016).

In the present review, we focus on the first category. Moreover, we consider the very important case where a parametric pdf model is used. One of the nice features of generative models based on explicit formulation of the model 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 many different applications, e.g., data denoising or data transformation. In any case, the pdf model has to be as close as possible to the true pdf of the modeled data, that is generally unknown. To this aim, the model has to be trained from data, and model parameter estimation is generally done by following the Maximum Likelihood methodology (Goodfellow et al., 2016; Bishop, 2006; Koller and Friedman, 2009). Those principles are of course valid for both conventional generative models and DGMs, but in the case of DGMs the parameters of the pdf model are generally the output of DNNs, which makes model training potentially difficult (we will come back on that important point in the following).

[1] In the present context, "conventional" refers to non-deep models, e.g., classical Hidden Markov Models and, more generally, Bayesian Networks. We discuss Bayesian Networks in the following.

1.1 Deep Dynamical Bayesian Networks

In the present review, we focus on an important sub-family of DGMs: Deep Dynamical Bayesian Networks (DDBNs). As the name indicates, DDBNs are built on the following models:

- Bayesian Networks (BNs) are a popular class of probabilistic models, for which i) the dependencies between all involved random variables are explicitly represented by pdf models,[2] and ii) those dependencies can be schematically represented using a directed acyclic graph (Bishop, 2006; Koller and Friedman, 2009). The structure of those dependencies often reflects (or originates from) some underlying hierarchical generative process.

- Dynamical Bayesian Networks are BNs that include temporal dependencies, and which are widely used to model dynamical systems and/or data sequences. Briefly stated, Dynamical BNs are BNs "repeated over time," that is to say, 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 seen as special cases of Dynamical BNs: Indeed, 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, as in usual SSMs.

- Deep Bayesian Networks (Deep BNs) combine BNs with DNNs: DNNs are used to generate the parameters of the modeled distributions. This gives them the ability to be high-dimensional and highly multi-modal while having a reasonable number of parameters. In short, Deep BNs have the potential to nicely combine the "explainability" of Bayesian models with the modeling power of DNNs.

DDBNs are thus a combination of all those aspects. 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 to model dynamical systems and/or data sequences. However, as already mentioned above, the combination of probabilistic modeling with DNNs in Deep BNs generally makes the training more complex and costly (compared to conventional BNs with the same overall structure but non-deep models for the generation of distribution parameters). This is an even more serious issue for DDBN where the repeating structure due to temporal modeling adds an additional level of complexity.

[2] Hence BNs belongs to the first category of the afore-mentioned simplistic taxonomy.

1.2 Variational inference and Variational Autoencoders

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[3] 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 almost at the same time in (Rezende et al., 2014).[4] The VAE is directly connected to the concepts of *latent* variable and *unsupervised representation learning*: The observed, possibly high-dimensional, random variable representing the data of interest is assumed to be generated from an unobserved low-dimensional latent variable through a probabilistic process. This latent variable is somehow at the heart of the overall model: It is assumed to "encode" the observed data in 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 of the data. When successful, this provides a nice 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, that is defined in the present context as the estimation of latent variable values from observed data, also plays a major role. As we will see in more detail later, in a Deep BN the exact posterior distribution (that is 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. Since the observed data likelihood function is also not tractable, the estimation of the model parameters is done by chaining the inference model (aka 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.[5] From now on, 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 (Kingma and Welling, 2014; Rezende et al., 2014) have had and continue to have a very strong impact on the machine learning community. VAEs have been applied to many signal processing problems such as the generation and the transformation of images and speech signals (we provide a few references in Section 2).

[3] Basically, a low-dimension to high-dimension generative feed-forward deep neural network.

[4] (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).

[5] Note that the idea of using an artificial neural network to approximate an inference model, and chaining the encoder and decoder, dates back to the early work presented in (Hinton et al., 1995). However, the "Wake-Sleep" algorithm presented in this paper for model training is significantly different from the one used to optimize the VAE.

1.3 Dynamical VAEs

As a Deep BN, the original VAE of (Kingma and Welling, 2014) does not include temporal modeling. This means that every data vector from a dataset is processed independently of the other data vectors (and the corresponding latent vector is also processed independently of the other latent vectors). In the years following the publication of (Kingma and Welling, 2014; Rezende et al., 2014), the VAE methodology was extended and successfully applied to several more complex Deep BNs. In particular it has been applied to Deep BNs with a temporal model, i.e., DDBNs, dedicated to the modeling of sequential data exhibiting temporal correlation. This was reported in a series of papers including (Bayer and Osendorfer, 2014; Fabius and van Amersfoort, 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 spirit of the VAE is preserved and cherished in those studies: Those DDBNs mix observed and latent variables, and aim not only at modeling data dynamics but also at discovering the latent factors governing them.

In practice, those different models vary in how they define the dependencies between the observed and latent variables, how they define and parameterize the corresponding generative pdfs, and, importantly, how they define and parameterize the inference model. They also differ on how they combine the variables with RNNs to model temporal dependencies, at both generation and inference. In contrast, and remarkably, the training phase is quite similar between models since it is consistently based on chaining the encoder and decoder and maximizing the VLB over a training dataset, that is applying the VAE methodology, possibly with a few adaptations and refinements.

In the end, it is difficult to say if we have to see those models as variational DDBNs, that is DDBNs immersed in the VAE framework, or as Dynamical VAEs (DVAEs), that is VAEs including a temporal model for modeling sequential data. They are probably both! In the following of the paper, as well as in its title, we chose to use the second term, that is *Dynamical VAEs*, since we assume that the term "VAE" is currently more popular than the term "DBN," and "Dynamical VAEs" gives a more speaking first evocation of those models, compared to "variational DDBNs."

1.4 Aim of the paper

The aim of the paper is manifold. First, to provide the background necessary to understand and motivate DVAEs. Second, to introduce the DVAE as a general class of models that encompasses many different combinations of VAE and temporal models, and provide its formal definition. Third, to review the recent literature that inspired this definition, and to present in detail seven models that are representative of the DVAE class, under the general proposed

definition. Finally, to compare the re-implementation of these seven models and discuss their advantages and drawbacks as well as future research guidelines.

More precisely, the contributions of this paper are the following:

- We provide a formal definition of what a Dynamical Variational Autoencoder is, what are its main properties and characteristics and how is it related to previous classical models such as VAE, RNN or SSM. We discuss the importance of correctly defining the dependencies between random variables as well as how these dependencies are implemented. Finally, we discuss the general methodology used to identify the stochastic dependencies of the latent variables at inference time and to compute the variational lower bound used for training DVAEs.

- We give a proper, detailed and complete technical description of the seven selected DVAE models (which is often overviewed in the original papers, independently of the authors' good will, because of lack of space). This hopefully will enable the reader to access the technical substance of those models much more rapidly and "comfortably" than by analyzing and comparing the original papers by him/herself. We have spent effort on consistency of presentation: For most 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 latent variable given the observed data, and we 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 missing (not always the same parts!) because of lack of space.

- Importantly, we also have spent some effort to make the notations homogeneous across models. In particular, for some models, we have changed the time indexation, and sometimes the name, of some variables. We took great care to do that consistently in the generative model, the inference part, and the VLB, so that this change of notation does not affect the essence and the functioning of the model. Together with consistency of presentation, this enables to better put in evidence the commonalities and the differences across models, and make their comparison easier. Notation remarks are specified in independent dedicated paragraphs when necessary all through the paper to facilitate the connection with the original papers.

- We relate those recent developments with history: Although there are already many papers on the VAE, including tutorials, we present the VAE in the first technical section because all subsequent DVAE models rely on the VAE methodology. Then we show how DVAEs are connected to RNNs and SSMs. The unified notation that we use may help readers from different communities (machine learning, signal processing, control theory, etc.) that are not familiar with those connections to discover them comfortably.

- We discuss the links, similarities and differences of the different DVAE models. We comment on the choices of the authors of the reviewed papers regarding the inference model, its relation with the exact posterior distribution, and implementation issues. Note that in the present review, we discuss only high-level implementation issues, that is the general structure of the neural network that implements a given DVAE at generation or at 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 reimplemented the seven DVAE models detailed in this review, and we have evaluated these models on a basic task (analysis-resynthesis of speech signals). The comparison of performance of the different models from the analysis of the literature is a very difficult task, for many reasons: All models are not evaluated on the same data; Chronological analysis of the publications naturally reveals that a new model improves over some previously proposed model(s), at least on some aspect(s), but we all know that this can depend on model tuning and experimental setup, and comparison done with a subset of previous models is incomplete in essence, etc. In short, an extended benchmark of DVAE models is not yet available. On the other hand, conducting an extended benchmark is a huge amount of work, since there are many possible configurations for the models, and many tasks for evaluating them. In particular, as we will see in this review, it is not yet clear how to evaluate the degree of "disentanglement" of the extracted latent space. For all those reasons (and also because this review paper is already quite long!), we limit the benchmark to a simple task in the present review. We plan to compare more extensively the models in future studies.

- Finally, the code re-implementing the seven DVAE models and used on the benchmark task is made available to the community. The open-source code and the best trained models can be downloaded at the following repository: `https://github.com/XiaoyuBIE1994/DVAE-speech`. 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.

In summary, we believe that comparison of models across papers is a difficult task in essence, whatever 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 help the readers in more easily accessing the technical substance of DVAEs, and see their cross-connections and their connections with pre-existing classical models.

Before we give the outline for the rest of the paper, it is useful to mention what we do *not* present in this review paper:

- All the DVAE models we review in a detailed manner consider *continuous latent random variables*. In other words, we do not consider the case of

discrete latent random variables. The latter can be incorporated in DVAE models, in the line of what is done for conditional-VAEs for exemple (Sohn et al., 2015; Zhao et al., 2017). In contrast, in the presented class of DVAE models, the sequence of observed random variables can be continuous or discrete, as in the original VAE formulation.[6]

- All the DVAE models we review in detail consider a *discrete-time sequence of (continuous or discrete) observed random variables associated with a corresponding discrete-time sequence of (continuous) latent random variable*. In other words, these models function in *sequence-to-sequence mode for both encoding and decoding*. We thus do not detail the VAE-based models specifically designed for text and dialogue generation (Bowman et al., 2016; Miao et al., 2016; Serban et al., 2016, 2017; Yang et al., 2017; Semeniuta et al., 2017; Hu et al., 2017; Zhao et al., 2018; Jang et al., 2019). These models generally have a many-to-one (discrete-to-continuous) encoder and a one-to-many (continuous-to-discrete) decoder, that is a whole sentence (sequence of words) is encoded into a single latent vector, which is in turn decoded into a whole sentence.[7] Even if those models can include some hierarchical structure at encoding and/or at decoding, they do not consider a temporal sequence of latent vectors.

- For the same reason, we do not detail the VAE-based models dedicated to (2D) image processing in this review, even if image rows or columns can be considered as sequences of consecutive (correlated) pixels. VAEs for image modeling generally apply many-to-one encoding and one-to-many decoding: They encode a whole image (that is a large number of pixels) in a single low-dimensional (continuous) latent vector. Note that the early works on VAE (Kingma and Welling, 2014), and many subsequent works, have considered image generation and transformation as a major application. The correlation between neighboring pixels was poorly exploited in early works since the conditional generative model (conditioned on the latent variable) was pixelwise independent. Subsequent works (Gulrajani et al., 2016; Chen et al., 2017; Lucas and Verbeek, 2018; Shang et al., 2018) considered mixing the VAE latent representation with a decoder exploiting local pixel correlations with either convolutive or auto-regressive decoding (van den Oord et al., 2016a,b), possibly combined with a multi-level or hierarchical latent encoding. These models often rely on knowledge accumulated on the use of deep neural networks for image processing, e.g., convolutional neural networks (CNNs) that decompose an image into

[6] Temporal models with binary observed and latent random variables can be found in (Boulanger-Lewandowski et al., 2012; Gan et al., 2015). Those models are based on Restricted Boltzman Machines (RBMs) or Sigmoid Belief Networks (SBNs) combined with RNNs. A detailed analysis of this kind of models is out of the scope of the present review.

[7] See (Roberts et al., 2018) for the same principle applied to music scores modeling. For examples of many-to-one encoding and one-to-many decoding with continuous latent and observed variables, see the model used in (Pereira and Silveira, 2018) for anomaly detection in energy time series (coupled with an attention model) and see the Factorized Hierarchical Variational Autoencoder (FHVAE) presented in (Hsu et al., 2017b). We discuss this latter model in Section 12.

successive feature maps and corresponding recomposition process. This makes those models a bit aside the temporal models we focus on.

- Finally, in this review we focus on DVAEs trained with the variational inference methodology, we do not review models based on GANs and more generally on an adversarial training methodology. Examples of extensions of "static" GANs to sequence modeling and generation, possibly including hierarchical modeling and disentangled representation issues, can be found in (Mathieu et al., 2016; Villegas et al., 2017; Denton and Birodkar, 2017; Tulyakov et al., 2018; Lee et al., 2018). We can note that this approach is particularly popular for separating content and motion in videos.

All those 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 can reveal beneficial to the other.

1.5 Outline of the paper

The following of the paper is organized as follows. We start with the background. We first present the VAE in Section 2, because this model is at the foundation of subsequent DVAE models. Since the introduction of temporal models in the VAE framework is closely linked to RNNs and SSMs, we rapidly present these two classes of models in Section 3.

In Section 4, we present the general class of DVAE models (that encompasses the seven detailed models): We give their definition, we discuss the variables dependencies in the DVAE pdfs, and discuss the extension of the VAE methodology to the DVAE models. To our knowledge, this is the first time this class of models is presented in such a general and unifying manner.

The next seven sections are dedicated to the detailed DVAE models. In Section 5, we present a basic example of combination of SSM with DNNs, trained with the VAE methodology, hence a first exemple of DVAE: The Deep Kalman Filter model (DKF) (Krishnan et al., 2015, 2017). Then we examine in details the Kalman Variational Autoencoder (KVAE) (Fraccaro et al., 2017) in Section 6, the Stochastic Recurrent Neural Network (STORN) (Bayer and Osendorfer, 2014) in Section 7, the Variational Recurrent Neural Network (VRNN) (Chung et al., 2015; Goyal et al., 2017) in Section 8, another type of Stochastic Recurrent Neural Network (SRNN) (Fraccaro et al., 2016) in Section 9, the Recurrent Variational Autoencoder (RVAE) (Leglaive et al., 2020) in Section 10, and finally the Disentangled Sequential Autoencoder (DSAE) (Li and Mandt, 2018) in Section 11.

In Section 12, we propose a more rapid overview of other DVAE models to complement our DVAE literature review. The benchmark of the seven detailed

models on a basic speech modeling task is presented in Section 13. Finally, we conclude this review with a discussion in Section 14.

We wish the reader to enjoy this DVAE tour.

Chapter 2

Variational Autoencoders

In this section, we rapidly present the Variational Autoencoder and the associated variational methodology for model training and inference. An extended tutorial paper on VAEs by the authors of the seminal paper (Kingma and Welling, 2014) can be found in (Kingma and Welling, 2019) (an ArXiv pre-print of this tutorial paper is also available).

2.1 Principle

For clarity of presentation, let us start with an autoencoder (AE). As illustrated in Fig. 2.1, an autoencoder 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 autoencoder 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 $\mathbf{z}$. Note that so far, everything is deterministic: At test time, each time we feed the AE with a specific input vector $\mathbf{x}_0$, the AE will provide the same corresponding output $\hat{\mathbf{x}}_0$.

The Variational Autoencoder (VAE) was initially proposed in (Kingma and Welling, 2014; Rezende et al., 2014). It can been 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 we will see below, the same probabilistic formulation applies to the encoding of $\mathbf{z}$. The resulting probabilistic model can be used to:

- generate new data (from unseen values of $\mathbf{z}$);
- transform existing data within an encoding-modification-decoding scheme; Note that the seminal papers on VAE and many following ones considered image generation and transformation, but examples of speech/music

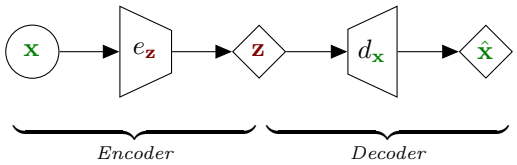

Figure 2.1: Schematic representation of an autoencoder (probabilistic graphical model enriched with DNN representation). The left trapezoid represents a high-to-low dimensional encoder DNN (denoted $e_{\mathbf{z}}$), and the right trapezoid represents a low-to-high dimensional decoder DNN (denoted $d_{\mathbf{x}}$). Calculation of the latent variable $\mathbf{z}$ and output $\hat{\mathbf{x}}$ from input $\mathbf{x}$ is totally deterministic, as represented by diamonds.

signals VAE-based transformation can be found in, e.g., (Blaauw and Bonada, 2016; Hsu et al., 2017a; Esling et al., 2018; Roche et al., 2019; Bitton et al., 2020);

- as a prior distribution of $\mathbf{x}$ in more complex Bayesian models for, e.g., 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 encoder and the decoder. Let us start with the decoder, since it forms the generative part of the model.

2.2 VAE decoder

In the following, $\mathcal{N}\big(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma}\big)$ denotes the 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 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 non-linear 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

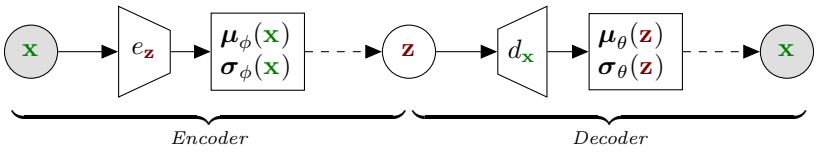

Figure 2.2: Schematic representation of the VAE: Encoder (left) and decoder (right). Dashed lines represent a sampling process. When the encoder and decoder are cascaded, using the same variable name $\mathbf{x}$ at both input and output is a bit abusive, but this is to be more consistent with the separate encoder and decoder equations.

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 Fig. 2.2 (right).

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

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

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

where 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 non-linear functions of $\mathbf{z}$ modeled by the decoder DNN. For the purpose of 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, i.e., we have:

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

A VAE decoder can be seen as a generalization of the probabilistic principal component analysis with a non-linear (instead of linear) relationship between $\mathbf{z}$ and the parameters $\theta_{\mathbf{x}}$. It can also be seen as 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:

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

Since any conditional distribution $p_{\theta_\mathbf{x}}(\mathbf{x}|\mathbf{z})$ can provide a mode, $p_\theta(\mathbf{x})$ can be highly multi-modal (in addition to being potentially highly dimensional). With this in mind it makes sense to set a diagonal covariance matrix in (2.3) since marginal distributions of arbitrary complexity can be obtained by designing and tuning the decoder network. Setting diagonal covariance matrices often makes the mathematical derivations easier.

2.3 VAE encoder

Training the generative model (2.1) consists in learning the parameters θ from a training dataset $\mathbf{X} = \{\mathbf{x}_n \in \mathbb{R}^F\}_{n=1}^{N_{tr}}$, where the training vectors $\mathbf{x}_n$ are assumed i.i.d. This is usually done by finding the values of parameters θ that maximize the training data marginal log-likelihood $\log p_\theta(\mathbf{X})$. Because the relation between $\mathbf{z}$ and $\{\mu_{\theta_\mathbf{x},f}(\mathbf{z}), \sigma^2_{\theta_\mathbf{x},f}(\mathbf{z})\}$ is modeled by a DNN, $p_\theta(\mathbf{X})$ is intractable.[1] Therefore, the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable too, and thus using the classical Expectation-Maximization (EM) algorithm is not possible.

Variational inference is an elegant methodology that overcomes these problems and enables us to efficiently train the VAE. It is grounded in two principles: i) Because the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, an *approximate posterior distribution* of $\mathbf{z}$ is introduced, denoted $q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z}|\mathbf{x})$, that has a similar general form to the generative conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ and that plays the role of the encoder, i.e. it enables the inference of the unobserved latent vector $\mathbf{z}$ from the corresponding observed vector $\mathbf{x}$, and ii) the encoder and the decoder are jointly trained, as we will see in the next subsection.

Let us first specify the encoder. A common choice for the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is to use a Gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\big(\mathbf{z}; \boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{x})\}\big) \tag{2.7}$$

$$= \prod_{l=1}^{L} q_\phi(z_l|\mathbf{x}) = \prod_{l=1}^{L} \mathcal{N}\big(z_l; \mu_{\phi,l}(\mathbf{x}), \sigma^2_{\phi,l}(\mathbf{x})\big), \tag{2.8}$$

where $\boldsymbol{\mu}_\phi : \mathbb{R}^F \mapsto \mathbb{R}^L$ and $\boldsymbol{\sigma}_\phi : \mathbb{R}^F \mapsto \mathbb{R}^L_+$ are non-linear functions of $\mathbf{x}$ modeled by a DNN called the *encoder network*[2] and parametrized by a set of weights and biases denoted ϕ. The encoder network is illustrated in Fig. 2.2 (left). Again, for the sake of consistency with the presentation of the other models, we can redenote:

$$[\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi(\mathbf{x})] = e_\mathbf{z}(\mathbf{x}), \tag{2.9}$$

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

[1] This is because the corresponding complete-data log likelihood $p_\theta(\mathbf{X}, \mathbf{Z})$ cannot be integrated out over the set of latent vectors $\mathbf{Z}$.

[2] The encoder network is also referred to as the recognition network in the literature.

2.4 VAE training

For VAE training, the encoder and the decoder are cascaded, as illustrated in Fig. 2.2, and the sets of parameter θ and ϕ are jointly estimated from the training data $\mathbf{X}$. Indeed, it is shown in (Kingma and Welling, 2014) that even if $\log p_\theta(\mathbf{X})$ is intractable, it is possible to calculate a lower bound of $\log p_\theta(\mathbf{X})$ that depends on both θ and ϕ, and maximize this lower bound w.r.t. both θ and ϕ. This lower bound is called the variational lower bound (VLB) or the evidence lower bound (ELBO) or the negative variational free energy, and is denoted $\mathcal{L}(\theta, \phi; \mathbf{X})$.

It is shown in (Kingma and Welling, 2014) that the VLB is given by:

$$\mathcal{L}(\theta, \phi; \mathbf{X}) = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})}\big[\log p_\theta(\mathbf{X}, \mathbf{Z})\big] \; - \; \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})}\big[\log q_\phi(\mathbf{Z}|\mathbf{X})\big], \tag{2.10}$$

which can be reshaped as:

$$\mathcal{L}(\theta, \phi; \mathbf{X}) = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})}\big[\log p_{\theta_\mathbf{x}}(\mathbf{X}|\mathbf{Z})\big] - D_{KL}\big(q_\phi(\mathbf{Z}|\mathbf{X}) \,\|\, p_{\theta_\mathbf{z}}(\mathbf{Z})\big) \tag{2.11}$$

$$= \underbrace{\sum_{n=1}^{N_{tr}} \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}\big[\log p_{\theta_\mathbf{x}}(\mathbf{x}_n|\mathbf{z}_n)\big]}_{\text{Reconstruction accuracy}} \; \underbrace{- \sum_{n=1}^{N_{tr}} D_{KL}\big(q_\phi(\mathbf{z}_n|\mathbf{x}_n) \,\|\, p_{\theta_\mathbf{z}}(\mathbf{z}_n)\big)}_{\text{Regularization}}, \tag{2.12}$$

where $D_{KL}(p_1 \,\|\, p_2) \geq 0$ denotes the Kullback-Leibler (KL) divergence between probability distributions p_1 and p_2. The equivalence between (2.11) and (2.12) is due to the assumed i.i.d. property of the data vectors. The total VLB is thus the sum of individual VLBs over each training vector. The left term in the right side of (2.11) and (2.12) is a reconstruction term that represents the average accuracy of the chained encoding-decoding process. The right term is a regularization term, that enforces the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to be close to the prior distribution $p_{\theta_\mathbf{z}}(\mathbf{z})$. This term forces $\mathbf{z}$ to be a disentangled data representation, i.e., 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):

$$\mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}[\log p_\theta(\mathbf{x}_n|\mathbf{z}_n)] \approx \frac{1}{R} \sum_{r=1}^{R} \log p_\theta(\mathbf{x}_n|\mathbf{z}_n^{(r)}). \tag{2.13}$$

In the present paper, we keep the same notation $\mathcal{L}(\theta, \phi; \mathbf{X})$ for the resulting approximate VLB, i.e. in practice we have:

$$\mathcal{L}(\theta, \phi; \mathbf{X}) = \sum_{n=1}^{N_{tr}} \frac{1}{R} \sum_{r=1}^{R} \log p_\theta(\mathbf{x}_n|\mathbf{z}_n^{(r)}) \; - \sum_{n=1}^{N_{tr}} D_{KL}\big(q_\phi(\mathbf{z}_n|\mathbf{x}_n) \,\|\, p(\mathbf{z}_n)\big). \tag{2.14}$$

Maximization of $\mathcal{L}(\theta, \phi; \mathbf{X})$ w.r.t. θ and ϕ is obtained with a gradient-ascent algorithm that includes the classical error backpropagation through the network layers. Usually some version of the Stochastic Gradient Descent (SGD) is used, i.e., the gradient descent (on the negative VLB) is applied on subsets of training data called minibatches.[3] For $\mathcal{L}(\theta, \phi; \mathbf{X})$ being differentiable w.r.t. θ and ϕ, one has to use i) "differentiable" encoder and decoder neural networks, which can be assumed for most usual DNN architectures, and ii) an explicit formulation of samples $\mathbf{z}_n^{(r)}$ as a function of the set of parameters ϕ. This is obtained with the so-called *reparameterization trick*, which consists in reformulating this sampling as:

$$z_{n,l}^{(r)} = \mu_{\phi,l}(\mathbf{x}_n) + \varepsilon\, \sigma_{\phi,l}(\mathbf{x}_n) \qquad \varepsilon \sim \mathcal{N}(0,1). \tag{2.15}$$

SGD and update of model parameters θ and ϕ are thus alternated with the above sampling using the lastly updated parameters ϕ. In practice, if we use sufficiently large minibatches, we can set $R = 1$ (Kingma and Welling, 2014). Note that all this optimization process is now considered as routine within deep learning toolkits such as Keras and PyTorch.

Because it is at the foundation of the variational approach, we will largely reuse the generic expression of $\mathcal{L}(\theta, \phi; \mathbf{X})$ given in (2.11) in the following of this review paper.

2.5 VAE improvements and extensions

Following the seminal VAE papers (Kingma and Welling, 2014; Rezende et al., 2014), many papers have been proposed for VAE improvements and extensions. Before we move toward dynamical models based on VAE, we briefly mention here some of those improvements and extensions. This is thus a shallow and non-exhaustive review, the purpose of the present paper is not to deepen this part of VAE literature, it is rather to mention some aspects that open the review toward DVAEs. The interested reader can refer to (Kingma and Welling, 2019) for a more detailed review of the "static" VAE improvements and extensions.

β-VAEs: The authors of (Higgins et al., 2017) proposed to introduce a weighting factor, denoted β, in (2.11)–(2.12) to balance the regularization and reconstruction terms:

$$\mathcal{L}(\theta, \phi, \beta; \mathbf{X}) = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})}\big[\log p_{\theta_\mathbf{x}}(\mathbf{X}|\mathbf{Z})\big] - \beta D_{KL}\big(q_\phi(\mathbf{Z}|\mathbf{X}) \,\|\, p_{\theta_\mathbf{z}}(\mathbf{Z})\big). \tag{2.16}$$

This enables the user to better control the trade-off between output quality and orthogonality/disentanglement of the latent coefficients $\mathbf{z}$. Basically, this principle is totally independent of the issue of "static" vs. temporal modeling, but it can be re-used in the framework of DVAEs.

VAEs with normalizing flows and/or hierarchical latent space: It has been proposed in (Rezende and Mohamed, 2015; Kingma et al., 2016; Louizos

[3] The resulting approximate VLB is sometimes referred to as the Stochastic Gradient Variational Bayes (SGVB) estimator in the literature (Rezende et al., 2014).

and Welling, 2017; Chen et al., 2017) to augment the flexibility (and thus the modeling power) of the approximate posterior distribution (i.e. the encoder) using a so-called normalizing flow, that is a mapping of the latent variable $\mathbf{z}$ from another variable. Several levels/layers of latent variable transformation can be chained, in particular via autoregressive modeling (Kingma et al., 2016). In the context of text/dialogue generation, a piecewise constant distribution for the prior distribution of $\mathbf{z}$ was used in (Serban et al., 2016).

In a more general manner, the latent space can be structured by setting a hierarchical prior distribution on a set of latent variables $\mathbf{z} = \{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_K\}$,[4] and extend/exploit this structure at the encoder level (Kingma and Welling, 2019, Chapter 4) (Salimans, 2016; Sønderby et al., 2016a,b). For example, in (Salimans, 2016), the author proposes to use a (deep) first-order auto-regressive prior model:

$$p_{\theta_\mathbf{z}}(\mathbf{z}) = p_{\theta_\mathbf{z}}(\mathbf{z}_0) \prod_{k=1}^{K} p_{\theta_\mathbf{z}}(\mathbf{z}_k|\mathbf{z}_{k-1}), \tag{2.17}$$

and a corresponding "mirror" approximate posterior model:

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_0|\mathbf{x}) \prod_{k=1}^{K} q_\phi(\mathbf{z}_k|\mathbf{z}_{k-1}, \mathbf{x}). \tag{2.18}$$

Note that in such approach, the latent space is structured but not the data space (we still have a unique observed vector $\mathbf{x}$). Note also that because the above-mentioned normalizing flow technique can consist of chaining multiple transformations of latent variables, it can be seen as a particular case of hierarchical encoder (Kingma and Welling, 2019; Kingma et al., 2016).

Importantly, so far in those hierarchical models, there is no notion of time, but if we think of timely ordered variables, then we go towards Markov models (in the above example a first-order one) that, as we will see later in this paper, will be at the heart of the DVAE framework. Also, the DVAE framework, as we define it, will consider both timely ordered latent variables and timely ordered observed variables.

In a slightly different but related line, the authors of (Bouchacourt et al., 2018) propose a hierarchical "multi-level" VAE with two latent vectors that are defined at different data scales: One latent vector encodes a common content for a group of data, and one other latent vector encodes the style of subgroups of data within a group. Data grouping involves a certain amount of supervision during training. Although there is no temporal aspect in (Bouchacourt et al., 2018), we will see later that the use of different latent variables to encode different levels of information in the data can be applied to sequential data characterized by features with different time resolutions.

Improved VAE decoders: As for using more sophisticated decoders, the authors of (Chen et al., 2017) considered an autoregressive conditional density of

[4] Here the index denotes different latent variables, not a sample in a training set as before.

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 (van den Oord et al., 2016b). Ideally, the local statistics of an image (e.g., local texture) should be modeled by the autoregressive part, whereas the global structural information of the image (e.g., objects) should be encoded in $\mathbf{z}$.

The authors of (Chen et al., 2017) discuss the tendency of the autoregressive part of the model to capture all the information on the data structure and let the latent variable unused. This problem had also been observed and discussed in the context of language/text modeling (Bowman et al., 2016). A general strategy to counter this effect, that is controlling the data features encoded by the RNN and the data features encoded in $\mathbf{z}$, is proposed at the early level of model design (basically, the local autoregressive window is constrained to be small). This can also be done with a hierarchical structure of the latent space (see previous paragraph), possibly combined with the different levels of image feature maps (Gulrajani et al., 2016; Gregor et al., 2016) or by introducing in the training procedure an auxiliary loss function that controls which information is captured by $\mathbf{z}$ and what is left to the autoregressive decoder (Lucas and Verbeek, 2018). Since the DVAE class of models also combines latent variables with RNNs, this problem is notable in our review context and we will come back to it in the discussion of Section 14.

Semi-supervised VAEs: The authors of (Siddharth et al., 2017) propose to force the disentanglement of $\mathbf{z}$ and thus improve 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.

Chapter 3

Recurrent Neural Networks and State Space Models

We have mentioned earlier that, basically, DVAEs are made of combinations of a VAE and temporal models. Most of these temporal models rely on recurrent neural networks (RNNs) and/or state space models (SSMs). We thus rapidly present the RNNs and SSMs in this section, before we go to DVAEs. An extended technical overview of RNNs and SSMs, as well as their applications, is out of the scope of the present paper.

3.1 Recurrent Neural Networks

3.1.1 Principle and definition

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

As for notations, 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, with $t_1 \leq t_2$. When $t_1 > t_2$, we assume $\mathbf{x}_{t_1:t_2} = \emptyset$. We present the RNNs in the general framework of non-linear systems, which transform an input vector sequence $\mathbf{u}_{1:T}$ into an output vector sequence $\mathbf{x}_{1:T}$, possibly through some internal state vector sequence $\mathbf{h}_{1:T}$. Input, output and internal state vectors can have arbitrary (different) dimensions. If $\mathbf{u}_{1:T}$ is an "external" input sequence, the network is considered as a "system" as is usual from the control theory point of view ($\mathbf{u}_{1:T}$ being often considered as a command to the system). If $\mathbf{u}_t = \emptyset$ the RNN is in *undriven* mode. And importantly, if $\mathbf{u}_t = \mathbf{x}_{t-1}$ the RNN is in *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 that 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 non-linear activation functions. We also need to 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 non-linear functions implemented with a deep neural network. 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) that comprise additional internal variables called gates. For simplicity of presentation, those additional internal gates are not formalized in (3.3) (3.4). The same for multi-layer RNNs where several recursive layers are stacked on top of each other (Graves et al., 2013) (in that case, for the same reason, we do not report layer indexes in (3.3) (3.4)). And the same for combinations of multi-layer RNNs and LSTMs, i.e., multi-layer LSTM networks. In summary, in all the following, we assume that equations (3.3) (3.4) are a "generic" or "high-level" representation of an RNN of arbitrary complexity.

Notation remark: Although it is a bit abusive, to clarify the presentation and the 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 we will do that all throughout the paper (and the same for $d_{\mathbf{h}}$ and for $d_{\mathbf{z}}$ to come).

So far, the above RNNs are totally deterministic: Given $\mathbf{u}_{1:T}$ and $\mathbf{h}_0$, $\mathbf{x}_{1:T}$ is totally determined. Such networks are trained by optimizing a deterministic criterion, e.g., the mean squared error (MSE) between target output sequences from a training dataset and corresponding actual output sequences from the network. Note that the training set of i.i.d. vectors used for the VAE training is here replaced with a training set of sequences of vectors, 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 turned into *generative* RNNs (GRNNs) by adding stochasticity at the output level: We just have to define a probabilistic

observation model and replace the output sequence of data with an output sequence of corresponding distribution parameters, just like for the VAE decoder:

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

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

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

Eq. (3.5) is the same recursive internal state model as (3.3). Eq. (3.6) and (3.7) compose the observation model. For the latter, we here use the Gaussian distribution for its generality and for convenience of illustration, though any distribution can be used, just as for the VAE decoder. Again, one may choose a distribution that is more appropriate to the nature of the data. For exemple, using mixture distributions was proposed in (Graves, 2013).

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. We can remark that 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 the size 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 very 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 of course a fundamental difference.

It can also be noted that even if the generation of $\mathbf{x}_t$ is now stochastic, the internal state evolution is still totally deterministic. Let us redenote $\mathbf{h}_t$ 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 every 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 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:

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

3.2 State Space Models

3.2.1 Principle and definition

State Space Models (SSMs) are a rich family of models that are widely used to model dynamical systems (in statistical signal processing, time series analysis,

[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 concision.

[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 $\mathbf{h}_1$.

control theory, etc.) (Durbin and Koopman, 2012). Here, we focus on discrete-time continuous-valued SSMs of the form:

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

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

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

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

where $d_{\mathbf{z}}$ and $d_{\mathbf{x}}$ are functions of arbitrary complexity,[3] each one being parameterized by a set of parameters denoted $\theta_{\mathbf{z}}$ and $\theta_{\mathbf{x}}$, respectively. As for the complete generative model, we thus have $\theta = \theta_{\mathbf{x}} \cup \theta_{\mathbf{z}}$, and we keep this notation in all the following of the paper. The observation model (3.11)–(3.12) is very similar to the GRNN observation model (3.6)–(3.7). Here, $\mathbf{z}_t$ is a stochastic internal state vector. Its distribution, known as the *state model* or the *dynamical model*, is given by (3.9)–(3.10). We can see it follows a first-order Markov model, i.e., a temporal dependency is introduced where $\mathbf{z}_t$ depends on the previous state $\mathbf{z}_{t-1}$ and corresponding input $\mathbf{u}_t$, through the function $d_{\mathbf{z}}$. The use of the Gaussian distribution in (3.10) and (3.12) is a usual convenient choice.[4] In short, the above SSM can be seen as a GRNN in which the deterministic internal state $\mathbf{h}_t$ has been replaced with a stochastic internal state $\mathbf{z}_t$, as illustrated in Fig. 3.1.

Notation remark: In the control theory literature, the input corresponding to the generation of $\mathbf{z}_t$ is often denoted $\mathbf{u}_{t-1}$, 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 temporal indices, so that input $\mathbf{u}_t$ is used to generate $\mathbf{z}_t$ which in turn is used to generate $\mathbf{x}_t$, to be better consistent through all the presented models.

As for the complete sequence, given the dependencies represented in Fig. 3.1, the joint distribution of all variables writes:

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

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:

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

[3] Here, $d_{\mathbf{z}}$ and $d_{\mathbf{x}}$ can be linear or non-linear. We will comment on those different cases in the following.

[4] More generally, those distributions are within the exponential family so that either exact or approximate inference algorithms can be applied, depending on the nature of $d_{\mathbf{z}}$ and $d_{\mathbf{x}}$.

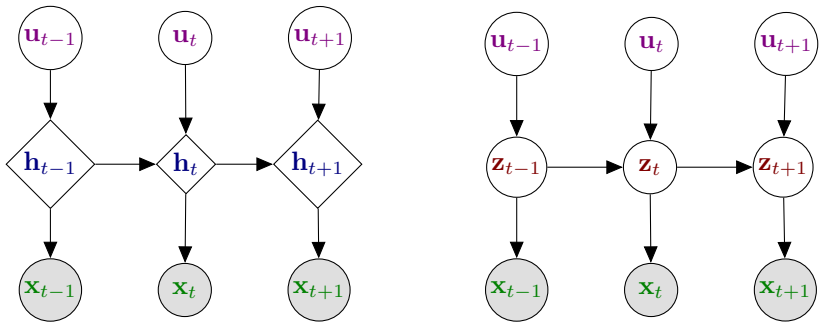

Figure 3.1: GRNN (left) vs. SSM (right): The two models have an identical structure, though the internal state of the GRNN is deterministic (represented with a diamond) whereas the internal state of the SSM is stochastic (represented with a circle).

Note that 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. Note also that, to be complete, we should specify more carefully the model "initialisation": At $t = 1$ we need to define $\mathbf{z}_0$ which can be set to an arbitrary deterministic value, or defined via a prior distribution $p_{\theta_\mathbf{z}}(\mathbf{z}_0)$ (that then must be added in 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 those equations is $p_{\theta_\mathbf{z}}(\mathbf{z}_1|\mathbf{u}_1)$.

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 turns into a Linear-Gaussian Linear Dynamical System (LG-LDS). In that case, the inference, that is the equations for optimal state vector sequence estimation from observed data, has a very popular closed-form solution known as the Kalman Filter (Moreno and Pigazo, 2009).[5] We do not present the equations of the Kalman filter here for concision, let us just mention that the inference is a linear form of the observations, and the parameters of this linear form are obtained from the model parameters with basic matrix/vector operations.

[5]The Kalman filter is the solution to inference using past and present observations (outputs and inputs). When the complete sequence of observations are used, the solution is the Kalman smoother, also obtainable in closed-form.

Non-linear Dynamical Systems (NDS), sometimes abusively referred to as Non-linear Kalman Filters, have also been largely studied, well before the deep learning era. Principled extensions to the Kalman Filter designed to deal with the non-linearities have been proposed, e.g., the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (EKF) (Wan and Van Der Merwe, 2000; Daum, 2005). A review of linear and non-linear Kalman filters is out of the scope of the present paper, to keep it focused on Dynamical VAEs.

Chapter 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, that we will describe in more detail later on. These models can be seen as particular instances of this general definition, given simplifying assumptions. Hopefully, this section will prepare the reader to understand well the commonalities and differences between all the models that we will review, and may motivate future developments. We first define a DVAE in terms of generative model, then we 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 a sequence a latent random vectors $\mathbf{z}_{1:T} = \{\mathbf{z}_t \in \mathbb{R}^L\}_{t=1}^T$. As opposed to the "static" VAE, and similarly to SSMs, those two data sequences are assumed to be temporally correlated and can have more or less complex (cross-)dependencies across time. Defining a DVAE generative model consists in specifying the joint distribution of the observed and latent sequential data, through the pdf $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$, which parameters are provided by DNNs, which themselves depend on a set of parameters θ.

When the model is working in the so-called *driven mode*, one additionally considers an input sequence of observed random vectors $\mathbf{u}_{1:T} = \{\mathbf{u}_t \in \mathbb{R}^U\}_{t=1}^T$, in which case $\mathbf{x}_{1:T}$ is seen as the output sequence. In that 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 commonly done in the DVAE literature (Krishnan et al., 2015;

Fraccaro et al., 2016, 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, 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. Also, we will consider the model in driven mode, i.e., with $\mathbf{u}_{1:T}$ as input, since it is more general than in undriven mode, i.e., with no "external" input. The undriven mode equations can 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 already started to discuss in Section 2.5, a DVAE can be seen as a structured or hierarchical VAE where both observed and latent variables are a set of ordered vectors, and the ordering is naturally imposed by time. However, the natural order present in the data does not imply a unique possible structure of variables dependencies for a DVAE generative (or inference) model. Indeed, in DVAEs, the joint distribution of the sequences of observed and latent vectors is usually defined by using the chain rule, i.e., 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 the ordering when applying the chain rule yields different practical implementations, which result in different sampling processes.

A natural choice for ordering the 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 only depends 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 *anti-causal*, and if the dependency combines the past, present and future of the conditioning variables, the model is said to be *non-causal*.

Let us consider the following very 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 value, from $t = 1$ to 2. On the contrary, 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 $\mathbf{x}_1$, then $\mathbf{x}_2$. This principle generalizes to much longer sequences of course.

In the DVAE literature, causal modeling is the most popular approach. In the following, we will therefore focus on causal modeling, but the general methodology is very similar for non-causal modeling. Actually, to the best of our knowledge, only one non-causal model was proposed in the literature: The

RVAE model of (Leglaive et al., 2020). In fact, both causal and non-causal versions of the 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 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 a 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 $\mathbf{x}_t$. Note that, similarly to our remark in Section 3.2.1, for $t = 1$ the first terms of the product 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 that $\mathbf{x}_{1:0} = \mathbf{z}_{1:0} = \emptyset$. Alternately, we could 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 those equations. In the following of this paper, for each detailed model, we will give the joint distribution in a general form of a product over frames from $t = 1$ to T and we do not detail the model "initialisation" for concision.

As will be seen later in detail, the different models proposed in the literature make different conditional independence assumptions in order 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 driven mode. In the causal context, we say that a DVAE is in driven mode if $\mathbf{u}_{1:t}$ is used to generate either $\mathbf{x}_{1:t}$, $\mathbf{z}_{1:t}$, or both. We say that 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 connected to the concept of *autoregressive process*, jointly found in the control theory, signal processing or time series analysis literature (Papoulis, 1977; Durbin and Koopman, 2012; Hamilton, 2020).

In its most general form (4.4), a DVAE is both in driven and predictive modes, however it can also be in only one of the two modes (e.g., the above SSM is in driven mode but not in predictive mode), or even in none of them. Note that in the literature, we did not encounter any DVAE in both modes at

the same time. Moreover, there are models in driven and non-predictive mode that are converted to non-driven and predictive mode by replacing the control input $\mathbf{u}_t$ with the previous generated output $\mathbf{x}_{t-1}$, see (Fraccaro et al., 2016). It is important to note that the behavior of a model may be quite different under the various modes. This is consistent with the concept of using a model in open loop or in closed loop in control theory. In a general manner, we remark that the principle of those different modes is poorly discussed (if discussed at all) in the DVAE literature, and we think it is interesting to clarify it at an early stage of the DVAE presentation.

4.1.2 Parametrization with (R)NNs

The factorization in (4.4) is a general umbrella for all (causal) DVAEs. As discussed above, each particular DVAE model will make different conditional independence assumptions that will simplify in various ways the general factorization. Once the conditional assumptions are made, one can easily determine if there is a need to accumulate 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 implementation of the former is done by means of recurrent neural networks, while feed-forward deep neural networks can be used to implement first-order Markovian dependency. Moreover, once the conditional assumptions are made, one may implement the remaining dependencies 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 one 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 $\mathbf{z}_t$ depend on $\mathbf{x}_{1:t-1}$. In addition, the generation of $\mathbf{x}_t$ also depends on $\mathbf{z}_t$ and the generation of $\mathbf{z}_t$ also depends on $\mathbf{u}_t$. Naturally, in order to accumulate the information of all past outputs $\mathbf{x}_{1:t-1}$, one can use a recurrent neural network. In practice, the past information is accumulated in the internal state variable of the RNN, namely $\mathbf{h}_t$, computed recurrently at every frame t. Among the many possible implementations, we consider two in this example: In the first implementation, illustrated in Figure 4.1 (middle), one 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.

Formally, assuming that all probability distributions are Gaussian, the first

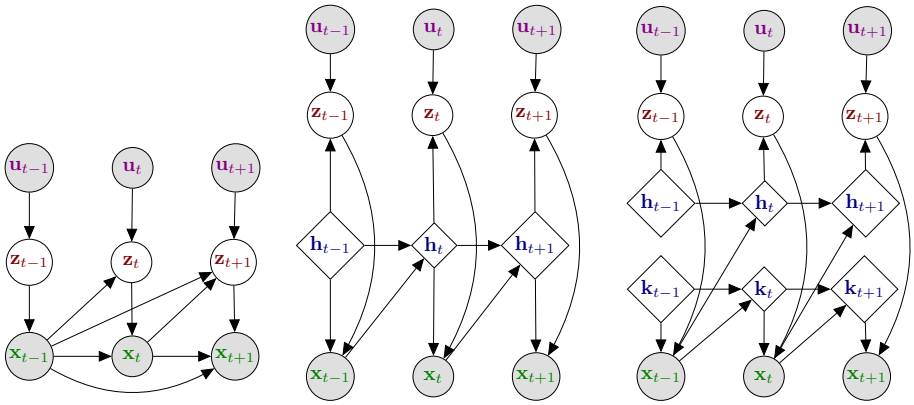

Figure 4.1: Two different implementations of a given factorization. The probabilistic graphical model (left) shows the dependencies between random variables and correspond to the factorization in (4.7)-(4.8). Two possible implementations based on RNNs are shown: Sharing the internal state variables (middle)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

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 non-linear functions implemented with DNNs. It is now clear that the parameters of the conditional distribution of $\mathbf{z}_t$ are $\theta_{\mathbf{z}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hz}}$, while those of the conditional distribution of $\mathbf{x}_t$ are $\theta_{\mathbf{x}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hx}}$. Obviously the two conditional distributions share the recurrent parameters $\theta_{\mathbf{h}}$. Regarding the second implementation, the generative process writes:

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

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

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

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

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

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

We have an additional DNN-based non-linear function $d_{\mathbf{k}}$, and analogously it is clear that the parameters of the conditional distribution of $\mathbf{z}_t$ are $\theta_{\mathbf{z}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hz}}$, while those of the conditional distribution of $\mathbf{x}_t$ are $\theta_{\mathbf{x}} = \theta_{\mathbf{k}} \cup \theta_{\mathbf{kx}}$. In that case, the two conditional distributions do not share any parameters.[1]

In the equations above, the operators $d_{\mathbf{h}}$, $d_{\mathbf{k}}$, $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$ are non-linear mappings parametrized by neural networks of arbitrary architecture. How to choose and design these architectures is out of the scope of this manuscript since 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. The current manuscript will not discuss how to select the hyper-parameters of those networks such as the number of layers, or number of units per layer.

A crucial remark is that equations (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 done to obtain the parameters $\theta_{\mathbf{z}}$ and $\theta_{\mathbf{x}}$ are different depending on the model. It becomes clear now why we make a distinction between the *compact* and the *developed* forms. We call the compact form of a DVAE model or its graphical representation when only random variables appear, e.g., (4.11) and (4.13), and Figure 4.1 (left). We call the developed form of a DVAE or its graphical representation when both random and deterministic variables appear, e.g., (4.9)-(4.13), (4.14)-(4.19) and Fig. 4.1 (middle) and (right). Each compact form may have different developed forms corresponding to different implementations. The distinction between the compact and developed forms is important, since the optimization happens on the parameters of the developed form, which is only a subgroup of all the 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 since they are possibly implicitly encoded in the internal state variables. Therefore, when reviewing in detail DVAE models in the following of the paper, we will systematically present both the compact and the developed graphical representations.

4.2 Inference model

In the present DVAE context, the posterior distribution of state sequence $\mathbf{z}_{1:T}$ is $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ (in driven mode, or $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ in undriven mode). As for the standard VAE, this posterior distribution is intractable due to non-linearities in the generative model. In fact, we could say that 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 $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$. And as for the standard VAE, this model is required not only for performing inference, that is here

[1] To ease the notation, from now on, we will denote by $\theta_{\mathbf{z}}$ and $\theta_{\mathbf{x}}$ the parameters of $p(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ and $p(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$ in (4.4), respectively, independently of whether or not they share some parameters. We will also use θ to denote $\theta_{\mathbf{z}} \cup \theta_{\mathbf{x}}$.

estimating 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 in the following. And as for the standard VAE again, the inference model is also using DNNs for generating 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 exist a general methodology to write 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): Briefly stated, some of the conditioning variables in the expression of the posterior distribution of a given variable can vanish depending on whether the nodes in between those 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), see (Geiger et al., 1990; Bishop, 2006) for details. Note that this methodology is very helpful even for more conventional (i.e., non-deep) 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 would have the same structure as the exact posterior distribution regarding variable dependencies. 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 review, especially the early papers, do not refer to this methodology and do not consider looking at the form of the exact posterior distribution when designing the approximate distribution.[2] 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 the one of the exact posterior distribution. We will come back on this point on a case by case basis when presenting the DVAE models in detail.

[2]We recall that a DVAE can be seen as a particular case of structured/hierarchical VAE, and interestingly and quite surprisingly, we remark that this methodology is not mentioned either in papers on VAEs with a structured/hierarchical latent space (Salimans, 2016; Sønderby et al., 2016a,b; Kingma and Welling, 2019). Yet it is a major principled way to guide the design of inference models in this more general case. In those papers, the design of the inference model is guided by other principles, in particular the consistency of the variables dependencies between the inference model and the prior latent model, and the resulting possibility to share parameters between them, as we will discuss in Section 4.2.3.

4.2.2 Non-causal 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 individual conditional distribution $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ can not be simplified. In other words, $\mathbf{z}_t$ does depend 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 non-causal process, even if generation is causal. As discussed above, the inference model q_ϕ should here have the same most general structure:

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

Similarly to the generative model, each individual conditional 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 some sort of bidirectional recurrent network.

Depending on the conditional independence assumptions made when defining the generative model, the posterior dependencies in $p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ may be simplified, using the above-mentioned D-separation property of Bayesian networks, and thus the posterior dependencies in $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}, \mathbf{u}_{1:T})$ may be simplified in the same manner. Of course, it is always possible to use an approximate posterior q_ϕ which does not follow the structure of the exact posterior distribution. In particular it makes sense to use an even more simplified version if one wants to significantly decrease the computational cost or force the inference to be a causal process (e.g., for online or incremental data processing). But of course this will generally be at the risk of decreasing the performance of inference. Again, we will come back to those points when reviewing the different DVAE models proposed in the literature.

4.2.3 Sharing variables and parameters at generation and inference

Interestingly, 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 may be made with a single unique RNN or with two different RNNs, in the line of what we discussed in Section 4.1.2. The same principle applies for $\mathbf{u}_{1:t}$ 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 realisations of random variables is the same at generation and at inference. If that is the case, the decoder and encoder share some network modules and thus θ and ϕ share some parameters. We remark that this is not the case in standard VAE. In the following of this paper, we will systematically use $\mathbf{h}_t$ to denote the internal state of the decoder, and we will use $\mathbf{g}_t$ to denote the internal state 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.

It can be noted that this principle to share some module and thus parameters between the encoder and the decoder has been pointed before in the VAE literature. In particular, as briefly stated in Section 4.2.1, this was mentioned in the papers dealing with VAEs with hierarchical latent space (Salimans, 2016; Sønderby et al., 2016a,b; Kingma and Welling, 2019). In fact, in those papers, this principle was used as a guiding strategy to design the inference model or choose one among different more or less intuitive structures (e.g., top-down vs. bottom-up inference in (Kingma and Welling, 2019)), in place of the D-separation methodology. This "module sharing" strategy was reported in those papers to lead to faster training and better model-data fitting. Note that, respecting the exact posterior distribution structure (applying D-separation) and module sharing are not necessarily incompatible. For example, the hierarchical model proposed in (Salimans, 2016) and reported in the present paper in (2.17) and (2.18) has it all: The inference model (2.18) both "mirrors" the generative model and follows the structure of the exact posterior distribution, even if this fundamental latter point is not mentioned in (Salimans, 2016).

4.3 Variational lower bound and training of DVAEs

As for the standard VAE, training a DVAE is based on maximization of the variational lower bound (VLB). In the case of DVAEs, the VLB initially defined in (2.10) naturally extends to data sequences as:

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

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:

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

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:

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

To our knowledge, this is the first time the VLB is presented in this most general form that is valid for the large 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 the regularization term require to compute 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 $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1},\mathbf{x}_{1:T},\mathbf{u}_{1:T})$, starting from $t=1$ up 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 reparametrization trick as for standard VAEs, so that the sampling-based VLB estimator remains differentiable with respect to ϕ. The VLB can then be optimized using gradient-ascent-based algorithms.

Note that the VLB is maximized with respect to both ϕ and $\theta = \theta_\mathbf{z} \cup \theta_\mathbf{x}$ (and remind that for DVAEs, ϕ and θ can share parameters, which is different from the "static" VAE, but perfectly alright for the optimization, technically speaking). Finally, note that the VLB is here defined for one single sequence from a training dataset, but a common practice consists in averaging the VLB over a mini-batch of multiple training sequences, before updating the model parameters with gradient ascent.

4.4 DVAE summary

Dynamical VAEs are constructed with various stochastic relationships between the control variables $\mathbf{u}_{1:T}$, the latent variables $\mathbf{z}_{1:T}$ and the 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 realisation of $\mathbf{a}$ is used to compute the parameters of the distribution of $\mathbf{b}$. The computation of these can be a linear or a non-linear mapping of the realisation of $\mathbf{a}$, and can include or not the realisation of other random variables. A DVAE model must contain two kinds 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 temporal link, that is $\mathbf{z}_{t-1}$ 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, i.e., the DKF/DMM 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 seen as additional decoding links as well, in the sense 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) or infinity (implemented with an RNN). However one could in principle use N-order temporal dependencies with $1 < N < \infty$, relying for instance on CNNs with finite-length receptive fields. In particular, temporal convolutional networks (TCNs) (Lea et al., 2016) which are based on dilated convolutions have shown to be competitive with RNNs on several sequence modeling tasks, including generative modeling (Aksan and Hilliges, 2019).

Finally, as discussed before, a DVAE can be in *driven mode*, in *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 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 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.

Chapter 5

Deep Kalman Filters (DKF)

Continuing from the previous section, we start our DVAE tour with the combination of SSMs with neural networks. Such combination is not recent, see e.g., (Haykin, 2004; Raiko and Tornio, 2009), but it has been recently investigated under the VAE angle in two papers by the same authors (Krishnan et al., 2015, 2017). The resulting deep SSM is referred to as a Deep Kalman Filter (DKF) in (Krishnan et al., 2015) or a Deep Markov Model (DMM) in (Krishnan et al., 2017).[1] Therefore, those 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 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 the unsupervised discovery of the latent space that encodes the temporal dynamics of the data. The VAE methodology enables to circumvent the difficulties encountered in previous approaches (Haykin, 2004; Raiko and Tornio, 2009) concerning the computational complexity and practicability of model parameter estimation, allowing in particular to go directly from single-layer neural networks to DNNs.

From now on, and for all detailed DVAE models in the following of this review paper, we first present the generative equations, then we present the inference model (and we discuss its choice by referring to the structure of the exact posterior distribution as computed from the generative model), and then we present the detailed form of the VLB used for model training (together with clues about the optimization algorithm).

[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. On their way the authors go from DKF to DMM, maybe because the second denomination appears more general. In the present review we keep the denomination DKF.

5.1 Generative model

We have already seen the generative equations of the DKF, since they are the same as (3.9)–(3.12) with the specificity that $d_{\mathbf{z}}$ and $d_{\mathbf{x}}$ are here DNNs.[2] In (Krishnan et al., 2015), those DNNs are not described in detail, we can assume that basic feed-forward neural networks, i.e., Multi-Layer Perceptrons (MLPs), are used. In (Krishnan et al., 2017), $d_{\mathbf{x}}$ is implemented with a two-layer MLP, and a slightly more refined model is used 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, ReLU activation and Softmax activation layers for the variance parameter (see Section 13.1.1). According to the authors, this is to let "the model have the flexibility to choose a linear transition for some dimensions while having a non-linear transition for the others."

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

- The stochastic state $\mathbf{z}_t$ of the SSM is very similar in spirit to the latent state of the VAE. In the present DKF case where $d_{\mathbf{x}}$ is implemented 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 on 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* the internal state level. As mentioned before, in a 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 Fig. 3.1 (which does not make the DNNs apparent).

5.2 Inference model

Following the general line of Section 4.2, we first look for 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:

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

[2] In fact, a Bernouilli distribution was considered for $\mathbf{x}_t$ in (Krishnan et al., 2015, 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.

Then one can use the D-separation principle to simplify each term of the product. From the structure represented in Fig. 3.1 (right), we can see that the $\mathbf{z}_{t-1}$ node "blocks" all information coming from the past and flowing to $\mathbf{z}_t$ (that is $\mathbf{z}_{1:t-2}$, $\mathbf{x}_{1:t-1}$ and $\mathbf{u}_{1:t-1}$). In other words, we could say that $\mathbf{z}_{t-1}$ has accumulated this past information, or is a summary of it. 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 thus (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 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 anti-causal process on $\mathbf{x}_t$ and $\mathbf{u}_t$).

In (Krishnan et al., 2015), the authors point out this structure and the fact that we can/should inspire from it to design the approximate posterior q_ϕ. However, somewhat surprisingly, they propose the four following 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 bidirectionnal RNN.

We do not detail those implementations of q_ϕ here, since we will give other detailed examples in the following. We may wonder why is $\mathbf{z}_{t-1}$ is not present in the conditional variables of the approximate posterior, but this may just be an oversight from the authors.[3] The point is that, despite the fact that the authors pointed out the dependency of the exact posterior on present and future observations and inputs, they do not propose a corresponding approximate model.

Notation remark: In (Krishnan et al., 2015), the authors denote by $\mathbf{u}_{t-1}$ the input at time t in the generative model, like in many control theory papers on SSMs, and not $\mathbf{u}_t$ as we do, as we pointed out in a previous footnote. However, when defining the four approximate posterior models, they do 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 in (Krishnan et al., 2015) that we somehow

[3] It is difficult to know from the paper since 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.

implictly corrected by using $\mathbf{u}_t$ instead of $\mathbf{u}_{t-1}$ as the input at time t in the generative model.

In (Krishnan et al., 2017), the authors present basically the same generative model, renamed DMM, and presented in undriven mode, i.e., $\mathbf{u}_{1:T}$ is simply removed. However, they largely clarify and improve on their previous paper regarding the inference model: They propose a new series of inference models that clearly do or do not depend on $\mathbf{z}_{t-1}$ and that also vary regarding the dependency on observed data. They consider again 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, that is $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T})$. In that 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), since it combines information from the past (through $\mathbf{z}_{t-1}$) and information from the present and future observations.

For concision of presentation, we report the detailed inference equations only for the DKS, and we 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 the combination of the RNN output with $\mathbf{z}_{t-1}$:

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

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

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

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

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 here $\phi = \phi_{\overleftarrow{\mathbf{g}}} \cup \phi_{\mathbf{z}}$, 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 $\mathbf{x}_{t:T}$, that we can rewrite $\mathbf{g}_t = \mathbf{g}_t(\mathbf{z}_{t-1}, \mathbf{x}_{t:T})$,[4] 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:

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

which is just a rewriting of (5.3).

As mentioned above, in (Krishnan et al., 2017), this inference model is extended to a non-causal (bidirectional) model regarding the observations, $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T})$.

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

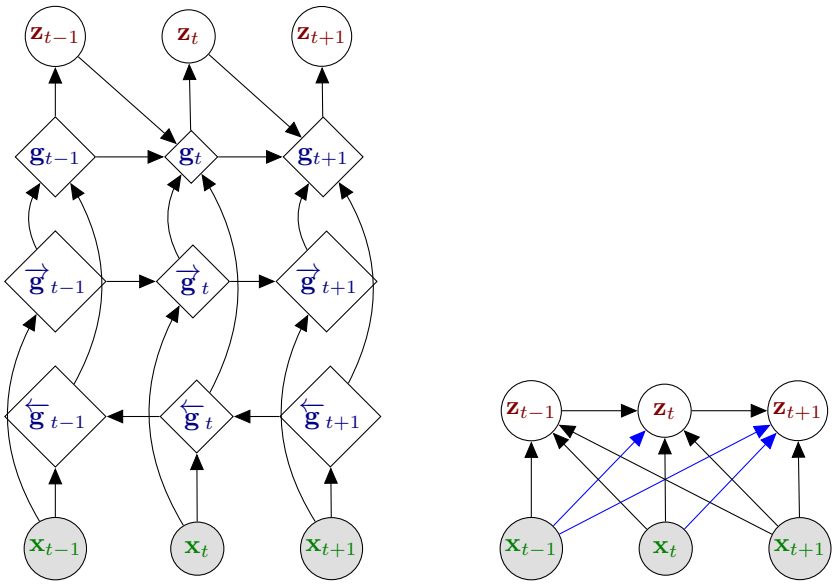

Figure 5.1: Graphical model of DKF at inference time corresponding to the inference model $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{1:T})$, in developed form (left) and compact form (right). The specific DKS model, which functional form $q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}_{t:T})$ 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).

This is done by adding a forward RNN on $\mathbf{x}_t$, and sending the output $\overrightarrow{\mathbf{g}}_t$ of this forward RNN into the combining network, in addition to $\mathbf{z}_{t-1}$ and $\overleftarrow{\mathbf{g}}_t$. For concision, we do not report the corresponding detailed equations, but this more general model is represented in Fig. 5.1. The other models proposed in (Krishnan et al., 2017) can be deduced from this general model by removing 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.

It is clear from the above equations and from Fig. 5.1 that the inference of $\mathbf{z}_t$ with a DKS requires a first backward pass from $\mathbf{x}_T$ up to $\mathbf{x}_1$. Then for each $t \in [1, T]$, $\overleftarrow{\mathbf{g}}_t$ is combined with a previous sample of $\mathbf{z}_{t-1}$, then $\mathbf{z}_t$ is sampled, and then we can process to the inference of $\mathbf{z}_{t+1}$. In short, a complete backward pass on $\mathbf{x}_t$ is followed by a complete forward pass on $\mathbf{z}_t$ which include iterative sampling of $\mathbf{z}_t$. As for the bidirectional version, we just need an additional forward pass on $\mathbf{x}_t$.

5.3 Training

Comparing the specific compact form of the DKF model in (3.13) with the general compact form of a DVAE in (4.4), we see 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 driven mode for the sake of generality), the VLB in its most general form (4.24) can be simplified as follows:

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

The KL divergence in (5.10) can be computed analytically while the two expectations are intractable. In (Krishnan et al., 2015) and (Krishnan et al., 2017), no detailed information is provided regarding how to approximate these expectations, it is only mentioned that "stochastic backpropagation" is used, referring the reader to (Kingma and Welling, 2014) and (Rezende et al., 2014) where the so-called reparametrization trick was introduced 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, and it deserves some more detail that we give now. It is important to note that we do not have an analytical form for $q_\phi(\mathbf{z}_t|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, we only have one for $q_\phi(\mathbf{z}_\tau|\mathbf{z}_{\tau-1}, \mathbf{x}_{\tau:T}, \mathbf{u}_{\tau:T})$. Therefore, we have 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 expression of the VLB can be trivially developed as follows:

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

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 individual intractable expectation in this cascade of expectations can then be approximated with a Monte Carlo estimate. It requires to sample iteratively $q_\phi(\mathbf{z}_\tau|\mathbf{z}_{\tau-1}, \mathbf{x}_{\tau:T}, \mathbf{u}_{\tau:T})$ (which we know analytically) from $\tau = 1$ to t, using the same reparametrization

trick as in standard VAEs. Doing so, the VLB becomes differentiable w.r.t θ and ϕ and can be optimized with gradient-ascent-based techniques.[5]

[5]Note that optimization of the VLB w.r.t. θ and ϕ actually means w.r.t. $\theta_{\mathbf{x}}$, $\theta_{\mathbf{z}}$, $\phi_{\overleftarrow{\mathbf{g}}}$ and $\phi_{\mathbf{z}}$ (we recall that $\theta = \theta_{\mathbf{x}} \cup \theta_{\mathbf{z}}$ and $\phi = \phi_{\overleftarrow{\mathbf{g}}} \cup \phi_{\mathbf{z}}$).

Chapter 6

Kalman Variational Autoencoders (KVAE)

The Kalman Variational Autoencoder (KVAE) model was presented in (Fraccaro et al., 2017). Basically, the KVAE model can be seen as a variant of the DKF model, hence another deep SSM, where an additional random variable denoted $\mathbf{a}_t$ is inserted in between the latent vector $\mathbf{z}_t$ and the observed vector $\mathbf{x}_t$, as illustrated in Fig. 6.1. This enables to separate the model in 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 very interesting properties for inference and training.

6.1 Generative model

The general formulation of the KVAE model is given by:

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

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

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

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

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

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

In (Fraccaro et al., 2017), Eqs. (6.1) and (6.3) are linear equations, i.e., they are given by (3.16) and (3.17), respectively (with $\mathbf{a}$ in place of $\mathbf{x}$, null bias vectors $\mathbf{n}_t, \mathbf{m}_t$ and time-invariant covariance matrices $\boldsymbol{\Lambda}, \boldsymbol{\Sigma}$). 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 sub-model on $\{\mathbf{u}_t, \mathbf{z}_t, \mathbf{a}_t\}$ is a very classical (non-deep) Linear-Gaussian LDS. In contrast,

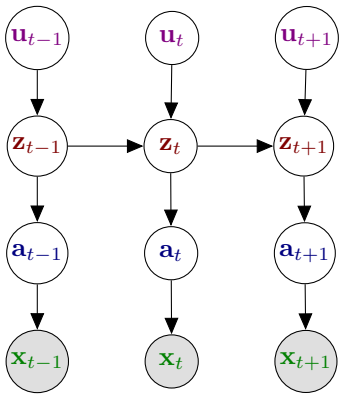

Figure 6.1: KVAE's graphical model.

$d_{\mathbf{x}}$ in (6.5) is implemented with a DNN, of 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 the dimension of $\mathbf{x}_t$. The feature vector $\mathbf{a}_t$ is expected to encode the properties of the "object(s)" present in the observation $\mathbf{x}_t$, while $\mathbf{z}_t$ is expected to encode the dynamics of those objects, which is an important application of the disentanglement concept that we have already mentioned. 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 factorizes as:

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

and we also have:

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

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

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

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.

In (Fraccaro et al., 2017), the authors mention the classical limitation of LDS for modeling abrupt changes in the trajectory of the LDS output (here $\mathbf{a}_t$). 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). In (Fraccaro et al., 2017) the authors propose to define each parameter of the LDS at each time t (e.g., matrix $\mathbf{A}_t$) as a linear combination of pre-defined matrices/vectors from a parameter bank, and the coefficients of the linear combination are estimated at each time t from the past features $\mathbf{a}_{1:t-1}$ using an LSTM network. Although very interesting, and a contribution on its own, we do not consider further this part of the KVAE model here, since it is relatively poorly relevant to our model review. We however provide a few more information on its implementation in Section 13.1.2. Note that a quite similar transition model was proposed independently in (Karl et al., 2017) as an instance of 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 observed variables is $p_\theta(\mathbf{a}_{1:T}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$. This latter factorizes as:

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

where the simplification of the first term on the right hand side comes 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 the Kalman filter or Kalman smoother (see Section 3.2.2). This Kalman solution is very classic and it is not detailed here for concision.[1] The other factor, $p_\theta(\mathbf{a}_{1:T}|\mathbf{x}_{1:T}, \mathbf{u}_{1:T})$, is more problematic.

The authors of (Fraccaro et al., 2017) do not discuss the form of the exact posterior distribution, yet they propose the following factorized inference model, that 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

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

is implemented with a fully-connected DNN:

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

Importantly, if we look at (6.5), (6.6), (6.14) and (6.15), we can see that they are identical to (2.5), (2.3), (2.7) and (2.9), with $\mathbf{a}_t$ in place of $\mathbf{x}$, and $\mathbf{z}_t$ in

[1] We can simply note that here, the Kalman solution depends only on $\theta_{\mathbf{a}} \cup \theta_{\mathbf{z}}$, but not on $\theta_{\mathbf{x}}$.

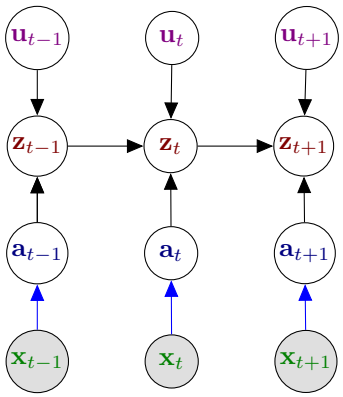

Figure 6.2: KVAE's graphical model at inference time. The black arrows represent the Kalman filter solution (causal solution) of the LG-LDS on $\{\mathbf{u}_{1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{1:T}\}$. The blue arrows represent the VAE encoder from $\mathbf{x}_t$ to $\mathbf{a}_t$. The inference solution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

place of $\mathbf{z}$. This means that, with the proposed inference model, a KVAE is composed of a VAE modeling the relationship between $\mathbf{a}_t$ and $\mathbf{x}_t$, placed on top of a Linear-Gaussian LDS on $\mathbf{u}_t$, $\mathbf{z}_t$ and $\mathbf{a}_t$, and that the inference solutions of the VAE and of the LG-LDS can be combined for the solution of this combined model. This is illustrated in Fig. 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. Note that the LG-LDS solution, which requires inverse matrix calculation, highly benefits from notable $\mathbf{x}_t$-to-$\mathbf{a}_t$ dimension reduction. Also, although this is not exactly discussed in those terms in (Fraccaro et al., 2017), we may envision that one possible motivation for designing the KVAE model is that the joint learning of all parameters (see next subsection) may encourage the feature extractor to provide $\mathbf{a}_t$ features that are well-suited to a linear dynamical model, i.e., the non-linear relations between observations $\mathbf{x}_t$ and dynamics $\mathbf{z}_t$ are (at least largely) captured via the VAE.

6.3 Training

The variational lower-bound for the KVAE model starts as usual:

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

where we have used the proposed decomposition of the posterior as well as the generative model.

In practice, one first samples from $q_\phi(\mathbf{a}_t|\mathbf{x}_t)$ for every 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. Importantly, this allows for jointly learning the parameters of the VAE (both encoder and decoder) and of the LG-LDS. Alternatively, one could get rid of $\mathbf{z}_{1:T}$ by integrating this out, but this would then not allow for learning the parameters of the LG-LDS using stochastic gradient ascent.

Chapter 7

STOchastic Recurrent Networks (STORN)

To our knowledge, the STOchastic Recurrent Networks model (STORN) (Bayer and Osendorfer, 2014) is the first DVAE model to combine an internal deterministic state $\mathbf{h}_t$ and an internal stochastic state $\mathbf{z}_t$. STORN was presented in (Bayer and Osendorfer, 2014) in predictive mode (i.e., with $\mathbf{u}_t = \mathbf{x}_{t-1}$), and from now on we keep this mode for all models for easier comparison, but STORN can also be easily set-up in driven mode with an external input $\mathbf{u}_t$.

7.1 Generative model

The STORN observation model is given by:

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

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

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

Therefore, Equations (7.2) and (7.3) are the same as for a basic single-layer GRNN, but in STORN $\mathbf{z}_t$ forms an additional input to the internal state $\mathbf{h}_t$. We discuss some consequences below. Note that 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 of course setting $\mathbf{u}_t = \mathbf{x}_{t-1}$):

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

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

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

In the following, we keep this latter more general formulation, for easier comparison with the other models. We recall that we denote by $\theta_\mathbf{h}$ and $\theta_{\mathbf{hx}}$ the set

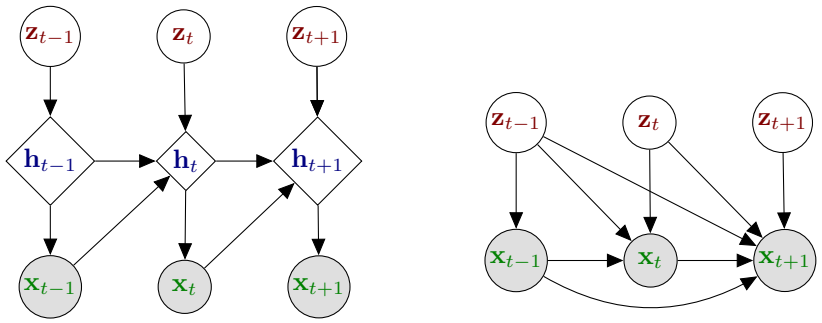

Figure 7.1: STORN's graphical model in developed form (left) and compact form (right).

of parameters of the networks implementing $d_{\mathbf{h}}$ and $d_{\mathbf{x}}$, respectively, and we have $\theta_{\mathbf{x}} = \theta_{\mathbf{h}} \cup \theta_{\mathbf{hx}}$.

In STORN, $\mathbf{z}_t$ is assumed i.i.d. with 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 $\mathbf{z}_t$. 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. Note that here we have $\theta_{\mathbf{z}} = \emptyset$, and therefore $\theta = \theta_{\mathbf{x}}$. The graphical model of STORN is given in Fig. 7.1 (left).

Notation remark: To ensure homogeneous notations across models, we slightly changed the notation of (Bayer and Osendorfer, 2014) by "synchronizing" $\mathbf{x}_t$ and $\mathbf{h}_t$, i.e., in our presentation of STORN, $\mathbf{x}_t$ is generated from $\mathbf{h}_t$ (and $\mathbf{h}_t$ is generated from $\mathbf{x}_{t-1}$, $\mathbf{h}_{t-1}$ and $\mathbf{z}_t$). In contrast, in (Bayer and Osendorfer, 2014) (Eq. (4)), $\mathbf{x}_{t+1}$ is generated from $\mathbf{h}_t$ (and $\mathbf{h}_t$ is generated from $\mathbf{x}_t$, $\mathbf{h}_{t-1}$ and $\mathbf{z}_t$). In other words, we have replaced $\mathbf{x}_t$ with $\mathbf{x}_{t-1}$. This change of notation does not change the model in essence while it 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, since it is a deterministic function of the latent random variables $\mathbf{z}_t$ and $\mathbf{x}_{t-1}$ and of its previous value $\mathbf{h}_{t-1}$. We present a proper treatment of this issue in Section 15: The recurrence on $\mathbf{h}_t$ is unfolded to consider $\mathbf{h}_t$ as a deterministic function of $\mathbf{z}_{1:t}$ and $\mathbf{x}_{1:t-1}$, that we can 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 (note that these points are rapidly discussed in (Bayer and Osendorfer, 2014), but not much detailed). It is shown

[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.

in Section 15 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:

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

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

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

We can remark that those data sequence densities factorize across time frames, but the whole history of 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 the line of the discussion in Section 4.1.2, an alternate description of STORN can be written, where we remove the internal deterministic 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 two above 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 Fig. 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 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 does not simplify 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 past observed and latent state vectors that propagate through the internal state, and it depends

on present and future observed vectors since $\mathbf{z}_t$ propagates to them through the internal hidden states.

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

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

and thus, combining with (7.8), we get:

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

This last equation is not practical since its complete calculation would require knowledge of $p_\theta(\mathbf{x}_{1:T})$ which is not tractable. However, we can note that $\mathbf{z}_t$ is present in all the terms $\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$ from t to T, and thus this confirms again that (7.14) seen 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 practical inference in STORN, the approximate posterior distribution q_ϕ is chosen in (Bayer and Osendorfer, 2014) as generated by an additional *forward* RNN. Few information is given on the implementation. It is said that the parameters of the approximate distribution of $\mathbf{z}_t$ are generated from $\mathbf{x}_{1:t}$, which is a bit odd since $\mathbf{x}_{t+1}$ was generated from $\mathbf{z}_t$ (with their notations; see our remark in the previous subsection). There is thus a 1-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 of notation at generation, we somehow automatically compensate for this gap, and we assume that the "correct" detailed inference equations are given by:

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

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

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

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

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

Note that this inference model can be rewritten in the general compact form:

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

[2] In (Bayer and Osendorfer, 2014), it is denoted $\mathbf{h}^r_t$.

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

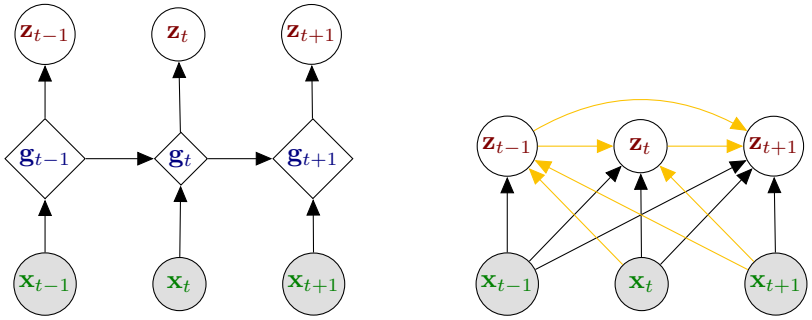

Figure 7.2: STORN's graphical model at inference time, in developed form (left) and compact form (right). Golden arrows correspond to missing links on the proposed probabilistic dependencies (compared to the exact inference dependencies).

The corresponding graphical model is given in Fig. 7.2. We remark that this inference model is inconsistent with the exact posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ in several points: At each time t, it does not consider 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 the 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 they are different.

7.3 Training

Comparing the compact form of STORN in (7.10) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$), we see 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:

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

This expression of the VLB relies on an inference model which is consistent with the exact posterior distribution (7.12). However, as discussed above, STORN assumes an inference model of the form: $q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) = q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t})$. Con-

sequently, the VLB in (7.21) simplifies as:

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

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 reparametrization trick for this recursive sampling leads to an objective function which is differentiable w.r.t. θ and ϕ.

Chapter 8

Variational Recurrent Neural Networks (VRNN)

The Variational Recurrent Neural Network model (VRNN) was proposed in (Chung et al., 2015), as a combination of a VAE and a RNN.

8.1 Generative model

The VRNN observation model is given by:

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

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

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

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

The generative distribution of $\mathbf{z}_t$ is given by:[1]

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

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

Notation remark: In (Chung et al., 2015), τ is used to denote the set of parameters for both feature extractors (respectively denoted $\varphi^{\mathbf{x}}_\tau$ and $\varphi^{\mathbf{z}}_\tau$ in (Chung et al., 2015)), as well as for $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$ (respectively denoted $\varphi^{\mathrm{dec}}_\tau$ and $\varphi^{\mathrm{prior}}_\tau$ in (Chung et al., 2015)). Moreover, in (Chung et al., 2015), $d_{\mathbf{h}}$ is denoted d_θ. We find this a bit confusing and we prefer to distinguish 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

[1] The authors of (Chung et al., 2015) use the terms "conditional prior distribution" for this distribution, which is a terminology we prefer to avoid.

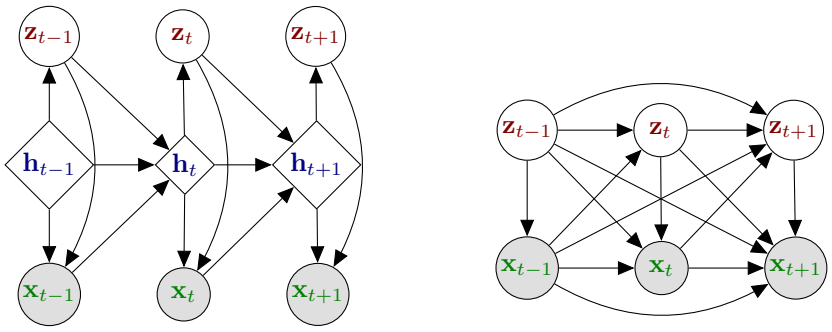

Figure 8.1: VRNN's graphical model in developed (left) and compact (right) forms.

τ_x and τ_z to make clear that the two feature extractors are different. We keep τ for simplicity. Besides, we replaced $\mathbf{h}_{t-1}$ in (Chung et al., 2015) with $\mathbf{h}_t$ to have $\mathbf{h}_t$, $\mathbf{z}_t$ and $\mathbf{x}_t$ synchronized, i.e., $\mathbf{x}_t$ 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 Fig. 8.1 (left). The generative process starts here 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 from now on, $\mathbf{z}_t$ is generated from $\mathbf{z}_{t-1}$, $\mathbf{h}_{t-1}$, *and* $\mathbf{x}_{t-1}$. Using the same "unfolding the recurrence" trick as in the previous sections, we can here redenote $\mathbf{h}_t = \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$,[2] and we 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 $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t) = p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{z}_t, \mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}))$. Note that this provides both $\mathbf{z}_t$ and $\mathbf{x}_t$ with an implicit temporal model.

As for a data sequence, when marginalizing the joint distribution of all variables w.r.t. $\mathbf{h}_{1:T}$ following the line of Section 15, we get:[3]

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

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

Again, we have a factorization of the conditional densities over time frames. Note that 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 $\mathbf{x}_t$.

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

[3] We obtain the same result with $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = p_\theta(\mathbf{x}_T, \mathbf{z}_T|\mathbf{x}_{1:T-1}, \mathbf{z}_{1:T-1}) p_\theta(\mathbf{x}_{1:T-1}, \mathbf{z}_{1:T-1}) = p_\theta(\mathbf{x}_T, \mathbf{z}_T|\mathbf{h}_t(\mathbf{x}_{1:T-1}, \mathbf{z}_{1:T-1})) p_\theta(\mathbf{x}_{1:T-1}, \mathbf{z}_{1:T-1})$, and applying the recurrence.

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 between 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 given in Fig. 8.1 (right).

8.2 Inference model

The general form of the exact posterior distribution of VRNN is identical to the one of STORN, i.e., it trivially 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 come from D-separation.

The approximate posterior distribution q_ϕ is chosen in (Chung et al., 2015) as:

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

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

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

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

We can see that, as opposed to STORN, the same internal state $\mathbf{h}_t$ is here shared by the VRNN encoder and the VRNN decoder, which, to our opinion, makes the approximate model more consistent with the exact posterior distribution.[4] Also, the inference at time t depends here on past outputs *and* past latent states, which also make it 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 is non-causal. The graphical model corresponding to the

[4] Note that this makes the set of parameters $\theta_\mathbf{h}$ common to the encoder and the decoder. Because the feature extractor $\varphi_\mathbf{x}(\mathbf{x}_t)$ is also used at the encoder, we have the same for its parameter set $\tau_\mathbf{x}$.

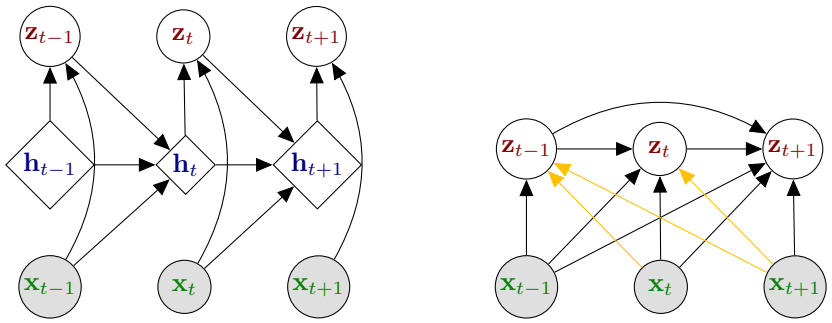

Figure 8.2: VRNN's graphical model at inference time, in developed form (left) and compact form (right). Golden arrows correspond to missing links on the proposed probabilistic dependencies (compared to the exact inference dependencies).

VRNN approximate inference process is given in Fig. 8.2. The inference model can be rewritten in the 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

Comparing the compact form of VRNN in (8.8) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$), we see that VRNN does not make any conditional independence assumption in the generative model. In this sense, VRNN is the most general DVAE model that 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, 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) simplifies as:

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

As for previous 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 can report that an improved version of VRNN was presented in (Goyal et al., 2017). The authors point out the difficulty in learning meaningful latent variables when coupled with a strong autoregressive decoder. We rediscuss this point and we provide a series of references in the next subsection and in Section 14. In (Goyal et al., 2017), the authors propose to improve the inference and training of VRNN with the three following features.

First, possibly inspired by (Krishnan et al., 2017) (and also by (Fraccaro et al., 2016), see Section 9.2), they introduce a backward RNN on $\mathbf{x}_t$ to feed the approximate posterior distribution $q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$, in the spirit of the one used in DKS (see Section 5.2). Therefore, they take into account future observations in the inference process, as opposed to the original VRNN, going toward a better compliance with the structure of the exact posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$.

Second, they force the latent variable $\mathbf{z}_t$ to contain relevant information about the future of the sequence by tying $\mathbf{z}_t$ with the internal state of the inference backward network (denoted $\mathbf{b}_t$ in (Goyal et al., 2017)). This is done by introducing an additional conditional model $p_\xi(\mathbf{b}_t|\mathbf{z}_t)$ and adding it in the VLB. In the same spirit they also consider an additional conditional model $p_\xi(\mathbf{x}_t|\mathbf{b}_t)$.

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

An adaptation of VRNN to automatic language translation is proposed in (Su et al., 2018). This is doubly interesting because this paper considers a sequence of discrete inputs and discrete outputs, which contrasts to the "all continuous" models we focus on. This paper also contrasts with previous VAE-based models for text/language processing, which, as mentioned in the introduction, usually consider a single latent vector to encode a whole input sequence (a full sentence). In (Su et al., 2018) 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 briefly mention here the study in (Lee et al., 2018) which uses a VRNN for speech synthesis and adopt adversarial training. Beyond VRNN, all those papers illustrate the flexibility of the DVAE class of models.

Chapter 9

Stochastic Recurrent Neural Networks (SRNN)

The Stochastic Recurrent Neural Network model (SRNN) has been proposed in (Fraccaro et al., 2016). According to the authors, the objective is 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., 2016), $\mathbf{h}_t$ is denoted $\mathbf{d}_t$, and the model is presented in driven mode, i.e., with an external input $\mathbf{u}_t$, which is here replaced 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:

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

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

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

Eq. (9.1) is identical to (3.3)[1] and thus refers to the usual deterministic RNN. Eq. (9.2) and (9.3) are quite similar to (8.2) and (8.3). So, the internal state $\mathbf{h}_t$ remains here totally deterministic and the integration of the latent state $\mathbf{z}_t$ is done at the $d_{\mathbf{x}}$ level. This justifies the "clear(er) separation of deterministic and stochastic layers" claimed by the authors of (Fraccaro et al., 2016), compared to VRNN.

The authors of (Fraccaro et al., 2016) also introduce an explicit temporal model on the distribution of $\mathbf{z}_t$ (as opposed to implicit temporal dependency

[1] with $\mathbf{x}_{t-1}$ instead of $\mathbf{u}_t$.

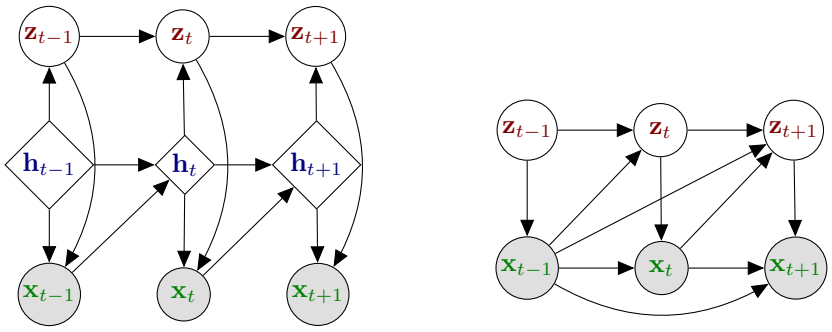

Figure 9.1: SRNN's graphical model in developed (left) and compact (right) forms.

through $\mathbf{h}_t$ in VRNN), in addition to the dependency on the internal state $\mathbf{h}_t$:

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

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

Basically, 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 $\mathbf{h}_t$). Also, compared to VRNN, no feature extractor is mentioned in SRNN, so that 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 given in Fig. 9.1 (left). Note that both $d_\mathbf{x}$ and $d_\mathbf{z}$ are two-layer feed-forward networks in (Fraccaro et al., 2016). $d_\mathbf{h}$ 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 redenote $\mathbf{h}_t$ as $\mathbf{h}_t(\mathbf{x}_{1:t-1})$,[2] and we have: $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t) = p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{h}_t(\mathbf{x}_{1:t-1}))$ 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 Section 15, marginalizing the joint distribution of all variables w.r.t. $\mathbf{h}_{1:T}$ leads here to:

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

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 more general alternate expression:

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

The corresponding compact graphical model is given in Fig. 9.1 (right).

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

9.2 Inference model

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

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

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

Importantly, the authors of (Fraccaro et al., 2016) point out this structure and propose 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 our knowledge, this was the first time. The dependency of q_ϕ on future observations, and also on past observations through the future internal states, is implemented with a gated backward RNN. This network is denoted $e_{\overleftarrow{\mathbf{g}}}$ in the equations below. It is parameterized by $\phi_{\overleftarrow{\mathbf{g}}}$,[3] and it is followed by basic feed-forward network $e_{\mathbf{z}}$, parameterized by $\phi_{\mathbf{z}}$. Formally, q_ϕ writes:

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

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., 2016), $q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ is denoted $q_\phi(\mathbf{z}_{1:T}|\mathbf{d}_{1:T}, \mathbf{x}_{1:T})$, with $\mathbf{d}_t = \mathbf{h}_t$. 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 same principle applies to the exact posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$.

As can be seen from the above equations, inference requires a forward pass on the internal state $\mathbf{h}_t$ (that 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 $\mathbf{x}_{1:T}$.[4] The graphical model corresponding to the inference process in SRNN is given in Fig. 9.2. The

[3] In (Fraccaro et al., 2016), the internal state of the decoder is denoted $\mathbf{a}_t$. We consistently use $\mathbf{g}_t$ and we add here a right-to-left arrow to highlight the backward nature of the process.

[4] Similarly to $\mathbf{h}_t$, we can redenote $\overleftarrow{\mathbf{g}}_t$ by $\overleftarrow{\mathbf{g}}_t(\mathbf{x}_{1:T})$ to make this latter point explicit, but this is poorly informative about the way $\overleftarrow{\mathbf{g}}_t$ depends on $\mathbf{x}_{1:T}$.

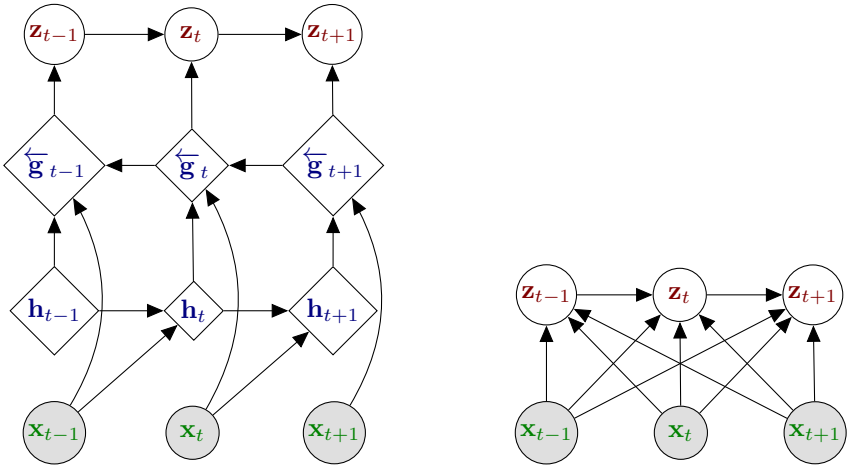

Figure 9.2: SRNN's graphical model at inference time, in developed form (left) and compact form (right). In this case, there are no missing links on the proposed probabilistic dependencies (compared to the exact inference dependencies). Note that in the inference graphical model in (Fraccaro et al.,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

inference model can be rewritten in the general form:

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

Note that the authors also state 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 $e_{\mathbf{z}}(\mathbf{z}_{t-1}, \mathbf{h}_t, \mathbf{x}_t)$.

9.3 Training

Comparing the compact form of SRNN in (9.7) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$), we see 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:

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

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).

Chapter 10

Recurrent Variational Autoencoders (RVAE)

A recurrent VAE (RVAE) was introduced in (Leglaive et al., 2020). This model was used to represent the clean speech signal in a speech enhancement application. It was combined with a Gaussian noise model with Non-negative Matrix Factorization of the variance, within a Bayesian framework. The parameters of the RVAE were estimated offline on a larget dataset of clean speech signals, using the VAE methodology (maximization of the VLB). A variational Expectation-Maximization (VEM) algorithm was then used for the estimation of 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 proposed in (Leglaive et al., 2020) was designed to model the signal (here a clean speech signal) 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 generic form:

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

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

where $\mathcal{T}$ denotes a set of time frames, and we have the three following cases: 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

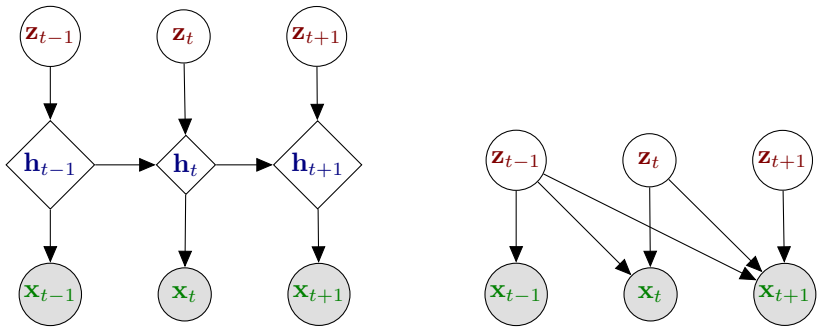

Figure 10.1: Causal RVAE's graphical model in developed form (left) and compact form (right).

non-causal model: $\mathcal{T} = \{1 : T\}$, which considers the complete sequence of latent state vectors.

Of course, 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 do from now on for easier comparison with the other models.

As in STORN, $\mathbf{z}_t$ is assumed i.i.d. with 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 $\theta = \theta_\mathbf{x}$. Note that case i) is strictly equivalent to the original VAE of Section 2, with no temporal model at all, and we will thus focus now on cases ii) and iii).

In (Leglaive et al., 2020), it is 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 paper does not provide the detailed implementation equations (though it provides a link to some supplementary material, including informative schemas). Let us write them now, for easier comparison with the other models (remind that 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).

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

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

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

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

Eq. (10.4) is very similar in form to the RNN internal state update (3.3) or (9.1), with the major difference that the latent state $\mathbf{z}_t$ is used as an input instead of an external input $\mathbf{u}_t$ or previous observation vector $\mathbf{x}_{t-1}$. Alternately, (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 observation model and 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 given in Fig. 10.1. As is now usual in our developments, we rewrite $\mathbf{h}_t = \mathbf{h}_t(\mathbf{z}_{1:t})$,[2] and we 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:

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

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

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

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

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

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

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

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

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

We thus have here $\mathbf{h}_t = \mathbf{h}_t(\mathbf{z}_{1:T})$.[3] The graphical representation of the non-causal RVAE model is shown in Fig. 10.2. For a complete data sequence, we

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

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

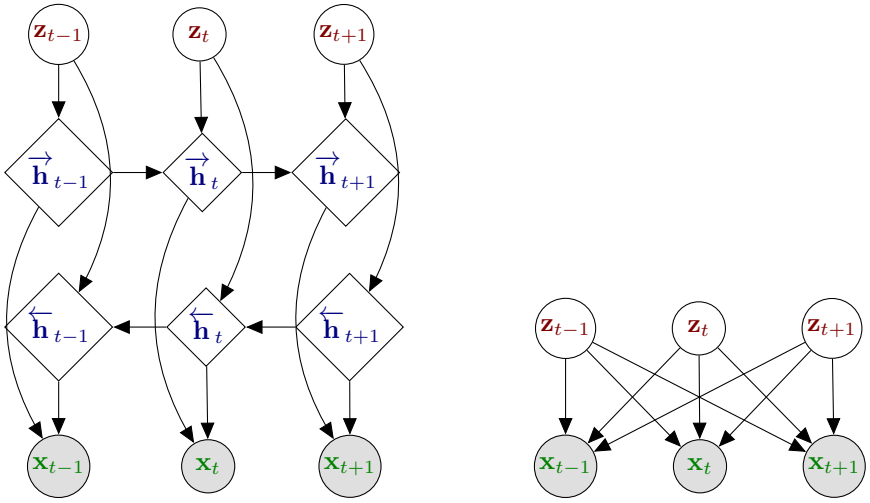

Figure 10.2: Non-causal RVAE's graphical model in developed form (left) and compact form (right).

have:

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

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, the authors in (Leglaive et al., 2020) first remark 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 non-causal case $\mathcal{T}' = \{1 : T\}$. We can see that for the causal generative model, the latent vector at a given time step depends (a posteriori) on past latent vectors but also on present and future observations, whereas for the non-causal generative model, it also depends on the past observations. Therefore, they chose to define the variational distribution

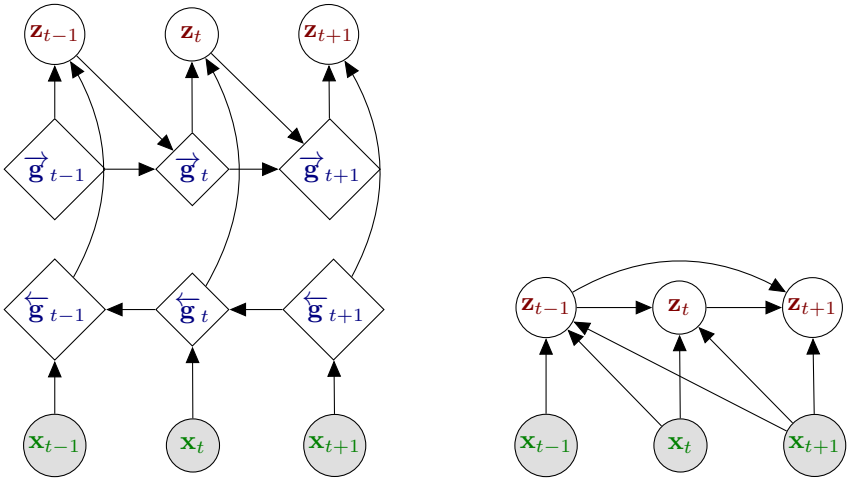

Figure 10.3: Graphical model of causal RVAE at inference time, in developed form (left) and compact form (right).

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 vectors and a backward RNN on the observations:

$$\overrightarrow{\mathbf{g}}_t = e_{\overrightarrow{\mathbf{g}}}(\mathbf{z}_{t-1}, \overrightarrow{\mathbf{g}}_{t-1}), \tag{10.18}$$
$$\overleftarrow{\mathbf{g}}_t = e_{\overleftarrow{\mathbf{g}}}(\mathbf{x}_t, \overleftarrow{\mathbf{g}}_{t+1}), \tag{10.19}$$
$$\mathbf{g}_t = [\overrightarrow{\mathbf{g}}_t, \overleftarrow{\mathbf{g}}_t], \tag{10.20}$$
$$[\boldsymbol{\mu}_\phi(\mathbf{g}_t), \boldsymbol{\sigma}_\phi(\mathbf{g}_t)] = e_{\mathbf{z}}(\mathbf{g}_t), \tag{10.21}$$
$$q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{t:T}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_\phi(\mathbf{g}_t), \mathrm{diag}\{\boldsymbol{\sigma}^2_\phi(\mathbf{g}_t)\}\big), \tag{10.22}$$
$$q_\phi(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_\phi(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{t:T}). \tag{10.23}$$

This inference model is represented in Fig. 10.3.

Non-causal case: The inference corresponding to the non-causal generative model is similar to the causal case except that the RNN on the observations is

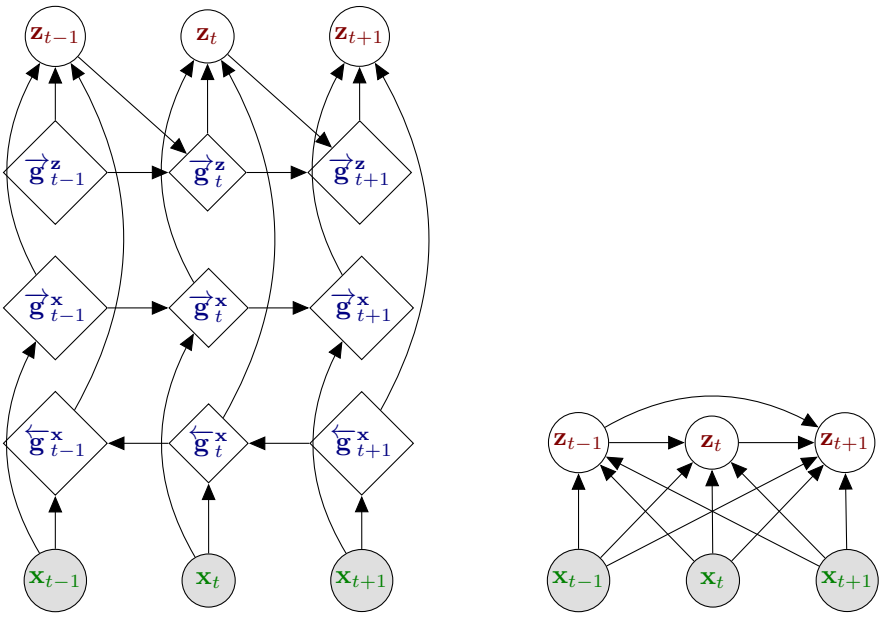

Figure 10.4: Graphical model of non-causal RVAE at inference time, in developed form (left) and compact form (right).

bidirectionnal:

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

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

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

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

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

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

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

This inference model is represented in Fig. 10.4.

10.3 Training

For the sake of concision and because we focus on reviewing causal DVAEs, we only describe in this section the VLB for the causal RVAE model, but the methodology to derive the VLB in the non-causal case is very similar.

Comparing the compact form of causal RVAE in (10.8) with the general compact form of a DVAE in (4.4) (simplified without $\mathbf{u}_{1:T}$), we see 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:

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

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

Chapter 11

Disentangled Sequential Autoencoders (DSAE)

The authors of (Li and Mandt, 2018) proposed a hierarchical model called Disentagled Sequential Autoencoder (DSAE). The DSAE 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 $\mathbf{f}$ in (Li and Mandt, 2018)), that 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

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

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

More detailed information about the pdfs and implementation issues are found in annexes from the ArXiv version of the paper. The authors use different variants for different datasets according to the nature of the data (basically, video clips or speech signals). We only report here 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

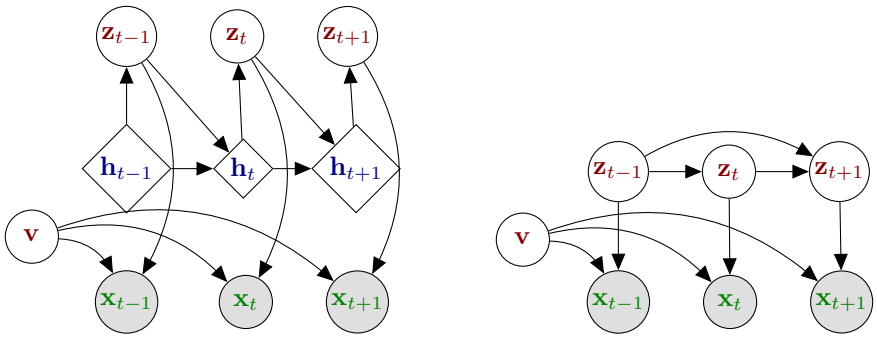

Figure 11.1: DSAE's graphical model in developed (left) and compact (right) forms.

(see Section 3.1.1), we thus can write:

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

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

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

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

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

The graphical representation of DSAE is given in Fig. 11.1. This model looks very much like a DKF in undriven mode conditioned on the variable $\mathbf{v}$, except that, in addition to this conditioning, the first-order Markov temporal model of the DKF is replaced with a virtually infinite-order model thanks to the LSTM. Although this is poorly discussed in the DVAE papers in general, this issue is an example of very interesting model extensions that are easy to implement in the deep learning and VAE framework.[1]

11.2 Inference model

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

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

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

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

[1] In (Krishnan et al., 2015), the authors mention that "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."

where the simplification in the last line comes 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, the authors of (Li and Mandt, 2018) propose two models. The first one, referred to as "factorized", is given by:

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

This model thus rely on an instantaneous frame-wise inference model $q_\phi(\mathbf{z}_t|\mathbf{x}_t)$ for the latent vector encoding dynamics. This approach is oversimplistic compared to the exact posterior distribution, and leads to reported performance that is inferior to the one 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 say, "the idea behind [this] structured approximation is that content may affect dynamics." So far, this model is compliant with the exact posterior distribution as expressed by (11.7). If we go now into more detail, 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:

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

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

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

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

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

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

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

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

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

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

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

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

We remark that none of the two approximations proposed by the authors (factorized and full) actually follow the dependencies of the exact posterior distri-

bution shown in (11.9). The graphical representation of the full inference model is represented in Fig. 11.2.[2]

11.3 Training

To derive the VLB for the DSAE model, we apply the same strategy as for other models, i.e., inject the generative model and the approximate posterior in the general formulation. We do so for the full inference model (11.11) and get:

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

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 t can be computed, without sampling from any random variable. Once this is achieved, then samples from $\mathbf{z}_{t-1}$ as well as from $\mathbf{z}_t$ are used to compute the t-th KL divergence term in (11.24).

[2]Note that in (Li and Mandt, 2018), Appendix A, the schematic representation of the inference model given in Fig. 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 on Fig. 9(b). We base the writing of (11.19)–(11.21) on their text and not on their figure.

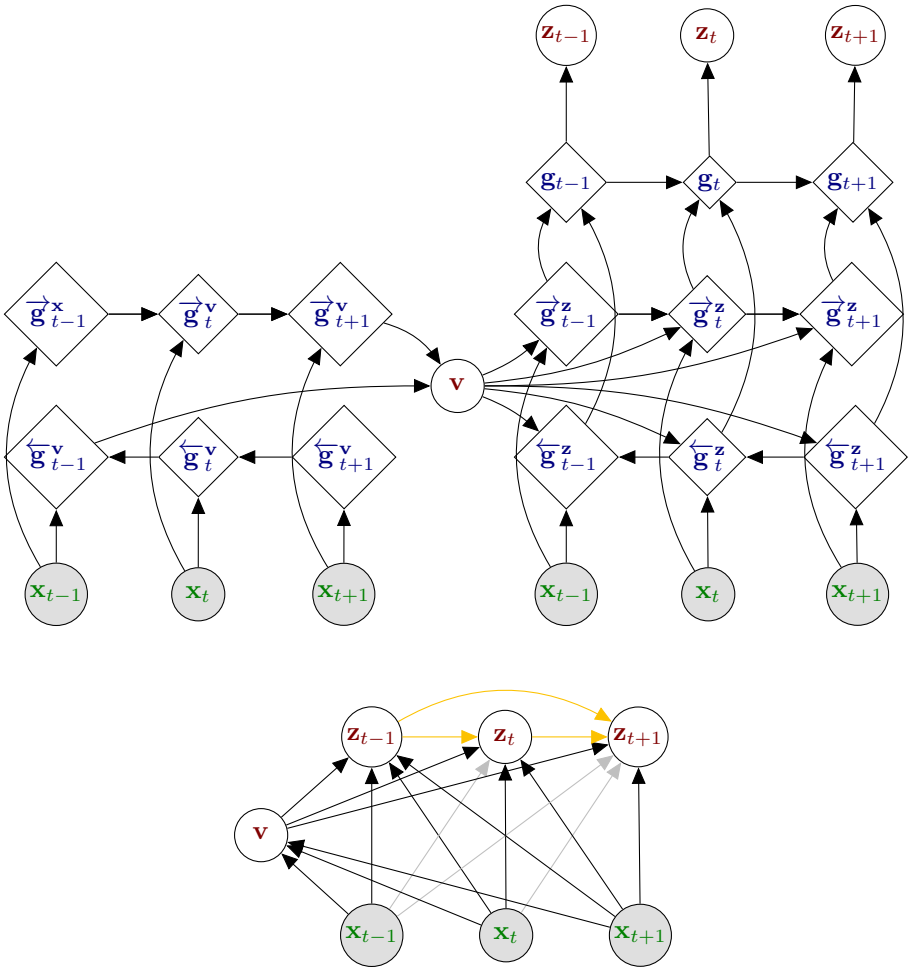

Figure 11.2: Graphical model of DSAE at inference time, in developed form (top) and compact form (bottom). In addition to the missing arrows shown in gold, we display in silver the arrows that should not be used, as compared to the structure of the exact posterior distribution. To our knowledge, DSAE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

Chapter 12

A rapid tour of other models

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

12.1 Models connected to the DKF model

Latent LDS and Structured VAE: The authors of (Johnson et al., 2016) consider several models. One of them is a simplified DKF where 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 non-linear model similar to the DKF observation model. They call it a Latent LDS. They extend this model to a Latent Switching LDS model that is based on a bank of dynamical models (actually the same linear dynamical model as above 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 seen as 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. The authors of (Johnson et al., 2016) do not provide detailed equations for these models. Rather, they show 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 call the resulting model a structured VAE (SVAE). Since they provide those developments in the general framework of probabilistic graphical models (Koller

and Friedman, 2009), which is more general that the DVAE framework, they do not provide "temporal equations." This makes this paper somehow poorly connected to our review, in spite of its strong interest. For example, although they clearly mention that the DKF model (Krishnan et al., 2015) is strongly related to their work, they also imply that using RNN models for implementing time dependencies is restrictive in the general framework that they present.

Black-box deep SSM: In a similar spirit, the authors of (Archer et al., 2015) also focus on the structure of the approximate posterior distribution to improve the computational efficiency of the inference. They propose to use a multivariate Gaussian approximate posterior with block tri-diagonal inverse covariance matrix. They also propose a corresponding inference algorithm that is fast and scalable. This general approach can be applied with different (deep and non-deep) 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 focus and experiment on a Linear-Gaussian LDS,[1] an LDS with a linear-Poisson observation model (which has no closed-form inference solution), and a basic one-dimensional non-linear LDS. This work is well connected with the DKF model and with deep SSMs in general. Interestingly, in this work 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 non-deep. This work was later extended, notably addressing online learning and real-time issues, in (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 in (Karl et al., 2017), extend 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., 2017), see also (Watter et al., 2015).

Disentangled SSM: The authors of (Miladinović et al., 2019) recently proposed a model called Disentangled State Space Model (DSSM) that is in the line of the DSAE model and, more generally, of models that attempt to separate the encoding of the content/object at the sequence level from the encoding of its dynamics at the time-frame level. However, in contrast to DSAE where 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 others

[1] This was to show that their algorithm can efficiently recover the solution of the Kalman Filter. In fact, the block tri-diagonal structure of their inference model is inspired by the Kalman solution.

words, (11.1) for DSAE is basically reshaped in DSSM as:[2]

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

Note that here, the dynamical model is a (conditioned) first-order model. The authors of (Miladinović et al., 2019) also propose a filtering inference model that is implemented in the more general DVBF framework mentioned just above.

12.2 Models connected to STORN, VRNN and SRNN

VRAE: The Variational Recurrent Autoencoder (VRAE) model presented in (Fabius and van Amersfoort, 2014) can been seen as a simplified version of STORN, from which, according to the authors themselves, it took inspiration: Here we have a sequence of data $\mathbf{x}_{1:T}$ that is encoded by a single latent random vector $\mathbf{z}$, instead of a sequence $\mathbf{z}_{1:T}$. The compact form of the generative model is given by:

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

No information is given in (Fabius and van Amersfoort, 2014) about $p_{\theta_\mathbf{z}}(\mathbf{z})$. $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z})$ 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 also 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 represented by Fig. 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 represented by Fig. 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, that strongly evokes the models designed for text/language processing mentioned in the introduction. Since (Fabius and van Amersfoort, 2014) was published in 2014 and is part of the early papers on DVAEs, this paper was probably very inspiring for the NLP community.

A very similar model was proposed in (Babaeizadeh et al., 2018), four years after (Fabius and van Amersfoort, 2014),[3] the difference being that in (Babaeizadeh et al., 2018) several vectors $\mathbf{x}_{t:T}$ are predicted from 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 van Amersfoort, 2014). The authors of (Babaeizadeh et al., 2018) compare this model with a more conventional model

[2] In (Miladinović et al., 2019), the sequence-level variable is denoted D for "domain." We keep the notation $\mathbf{v}$ as for our description of DSAE for consistency.

[3] (Fabius and van Amersfoort, 2014) is not cited in (Babaeizadeh et al., 2018).

where the latent vector is defined on a frame-by-frame basis, i.e., $\mathbf{z}_t$ (basically the observation model of this baseline model is similar to the one of SRNN and the prior over $\mathbf{z}_t$ is an i.i.d. standard Gaussian distribution).

FHVAE: A Factorized Hierarchical Variational Autoencoder (FHVAE) is proposed in (Hsu et al., 2017b), which learns disentangled and interpretable latent representations from sequential data without supervision by explicitly modeling 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 consecutive subsequences of fixed size (called segments), and defining two latent variables $\mathbf{z}$ and $\mathbf{v}$ at the segment level.[4] The former is dedicated to capture data information at the segment level, whereas the latter is dedicated to capture data information across segments, that is at the sequence level. This model is particularly appropriate for speech signals: In that 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 spirit, FHVAE is strongly connected 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 the 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 segment within a sequence. The FHVAE observation model *for each individual segment of data* is given by:

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

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

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

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. Note that there is one single pair of such vectors for each segment, hence a many-to-one encoding and one-to-many decoding at the segment level. In practice, those 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 which prior distribution $p_{\theta_{\mathbf{w}}}(\mathbf{w})$ is also a centered isotropic Gaussian. The distribution of $\mathbf{v}^{(n)}$ is then given by $p_{\theta_{\mathbf{v}}}(\mathbf{v}^{(n)}|\mathbf{w}) = \mathcal{N}(\mathbf{v}^{(n)}; \mathbf{w}, \sigma_{\theta_{\mathbf{v}}}^2 \mathbf{I}_{L_v})$. 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 $\mathbf{z}^{(n)}$ will behave as a segment-dependent and

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

sequence-independent latent factor. The joint density of a sequence is given by:

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

where, as in the previous sections, $\mathbf{h}_t\left(\mathbf{z}^{(n)}, \mathbf{v}^{(n)}\right)$ 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, the authors of (Hsu et al., 2017b) propose the following model:

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

where $q_{\phi_\mathbf{z}}$ and $q_{\phi_\mathbf{v}}$ are both implemented with a forward LSTM network, of which the last state vector is passed to a DNN network to provide the distribution parameters. Note that two encoders are chained here: The first one is used to generate $\mathbf{v}^{(n)}$ (by sampling $q_{\phi_\mathbf{v}}\left(\mathbf{v}^{(n)} | \mathbf{x}_{1:T}^{(n)}\right)$). 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 which mean vector is taken 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). Note that the model can be optimized at the segment level instead of at the sequence level, that is each data segment can be used as a batch dataset. According to the authors of (Hsu et al., 2017b), this can solve scalability issues when training sequences become too long.

DRAW: A somehow dual model of (Fabius and van Amersfoort, 2014) has been proposed in (Gregor et al., 2015) and called DRAW for Deep Recurrent Attentive Writer: DRAW considers a sequence of latent vectors $\mathbf{z}_{1:T}$ to encode a single static but highly structured data $\mathbf{x}$ (a digit image or a low-resolution image from the CIFAR database). 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 seen as 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 input) and a so-called canvas matrix $\mathbf{c}_{t-1}$ that 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 output of the decoder at

previous time step, hence the predictive coding is implemented in close-loop mode (Gersho and Gray, 2012). As the name indicates, DRAW includes a selective attention model that enables the model to focus on the most relevant parts of the observation. The description of such an attention model is out of the scope of the present review, see (Gregor et al., 2015) and references therein for details.

NASMC: The authors of (Gu et al., 2015) propose a generative model called Neural Adaptive Sequential Monte Carlo (NASMC) that is very close to STORN.[5] It can also be seen as a simplified version of VRNN or SRNN. The inference model has the same general form as VRNN, i.e., it follows (8.14), and it is parameterized by an RNN. The originality of NASMC is to connect the DVAE inference framework with sequential Monte Carlo (SMC) sampling. The inference model is 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 is proposed. In other words, the authors of (Gu et al., 2015) show 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.

RSSM: The authors of (Hafner et al., 2018) use a model that they refer to as Recurrent SSM (RSSM), and that is very similar, if not identical, to VRNN used in driven mode, i.e. VRNN with an external input $\mathbf{u}_t$, that is denoted $\mathbf{a}_{t-1}$ in (Hafner et al., 2018), to feed $\mathbf{h}_t$, instead of reinjecting $\mathbf{x}_{t-1}$ into $\mathbf{h}_t$.[6] They use this RSSM model for learning the dynamics and planning the actions of a synthetic agent from image sequence observations, in a reinforcement learning framework. Interestingly, they also present a way to do multi-step prediction, that is prediction several steps ahead.

12.3 Other models

FVAE: The authors of (Deng et al., 2017) propose a model called Factorized VAE (FVAE) that combines the VAE with tensor factorization (Kuleshov et al., 2015; Huang et al., 2016), which is applied on latent vector $\mathbf{z}$. Since one of the dimensions of the tensor factorization 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 every time-frame. It is thus unclear how the temporal dynamics are actually encoded.

GP-VAE: In the recent paper (Fortuin et al., 2020), the authors combine a VAE for observed data dimension reduction and a multivariate Gaussian Process (GP) (Williams and Rasmussen, 2006) for modeling the dynamics of the

[5] In the supplementary material to the paper, they break the dependency between $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$ that is set in the main text of the paper, and thus the model becomes identical to STORN.

[6] Note that the VRNN model was presented in such a driven mode in the SRNN paper (Fraccaro et al., 2016) in their "Related Works" section, see their Figure 4(b).

resulting latent vector $\mathbf{z}_t$. More specifically, for the Gaussian Process model they use a Cauchy kernel, which is appropriate to model data with multi-scale time dynamics. They propose to use an approximate posterior distribution that is also a multivariate GP (here a first-order one). The resulting overall GP-VAE model is trained with the VAE methodology. It is then used to efficiently recover missing data in test sequences (in videos and medical data). One nice property of this model is that it provides interpretable uncertainty estimates.

Chapter 13

Experiments

In the following, we present an experimental benchmark of the seven DVAE models that we detailed in the previous chapters. We showcase this benchmark with the task of speech analysis-resynthesis, that is encoding and reconstructing speech signals, that here will play the role of the data $\mathbf{x}_{1:T}$, encoded into, and resynthesized from, a latent vector sequence $\mathbf{z}_{1:T}$. We first specify in Section 13.1 the implementation of each model. We then describe the experimental protocol, dataset, model training and evaluation metrics in Section 13.2. Finally, we provide and discuss the obtained results in Section 13.3. We recall that the open-source code and the best trained models can be downloaded at the following repository: `https://github.com/XiaoyuBIE1994/DVAE-speech`.

13.1 Implementation of the DVAE models

In our implementation of each DVAE, we have tried to find a good trade-off between respecting the architecture of the model as described in the original paper and ensuring a fair comparison between the different models. For this aim, we tried our best to have the same design choice for the different models, whenever possible. Therefore, rather than using the same dimensions and hyper-parameters as presented in the original papers, we set some parameters to be equal among all models, and chosen specifically for the speech analysis-resynthesis task. We first describe the parameters common to all models, and then we focus on the specifics of each model.

We use the following specifications common to all DVAEs:

- None of the DVAEs is used in driven mode, that is, for any model, there is no external input $\mathbf{u}_{1:T}$. The DVAEs proposed in predictive mode in the literature are used in predictive mode in our experiments as well;
- The dimension of the observation vector $\mathbf{x}_t$ is set to 257 (see the speech data pre-processing in Section 13.2);
- The dimension of the latent vector $\mathbf{z}_t$ is set to 16;

- Unless specified, the dimension of hidden internal state vectors ($\mathbf{h}_t$ or $\mathbf{g}_t$) is set to 128;
- Unless specified in the original paper, all RNNs are instantiated as LSTM networks;
- For all variance parameters at the output of a network, we use log-parameterization (i.e., the output of network corresponding to a variance parameter σ^2 is actually $\log \sigma^2$).

In the following, we provide the full specifications for each model. As done previously when presenting each DVAE, we will first present the generation network (decoder) and then the inference network (encoder). Since many of the networks are multi-layer perceptrons (MLP), we define a notation to refer to these architectures concisely: MLP($\mathbf{y}$, n_1, f_1, ..., n_L, f_L) refers to a 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. Possible activation functions are: rectified linear unit (ReLU), sigmoid (Sigmoid), hyperbolic tangent (Tanh) and linear $\mathbb{I}$. Importantly, 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 a RNN, is always linearly activated and will not be made explicit for the sake of concision. 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, both with Sigmoid activation function, and an output layer with dimension 2×16 (for mean and log-variance vectors, which are both the same size as $\mathbf{z}_t$) with linear activation.

13.1.1 DKF

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

$$\boldsymbol{\nu}_t = \text{MLP}(\mathbf{z}_{t-1}, 16, \text{ReLU}, 16, \text{Sigmoid}) \tag{13.1}$$

$$\boldsymbol{\mu}_t^{\text{nonlin}} = \text{MLP}(\mathbf{z}_{t-1}, 16, \text{ReLU}, 16, \mathbb{I}) \tag{13.2}$$

$$\boldsymbol{\mu}_t^{\text{lin}} = \text{MLP}(\mathbf{z}_{t-1}, 16, \mathbb{I}) \tag{13.3}$$

$$\boldsymbol{\mu}_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}) = (1 - \boldsymbol{\nu}_t) \odot \boldsymbol{\mu}_t^{\text{lin}} + \boldsymbol{\nu}_t \odot \boldsymbol{\mu}_t^{\text{nonlin}} \tag{13.4}$$

$$\boldsymbol{\sigma}^2_{\theta_{\mathbf{z}}}(\mathbf{z}_{t-1}) = \text{MLP}(\text{ReLU}(\boldsymbol{\mu}_{\text{t}}^{\text{nonlin}}), 16, \text{Softplus}), \tag{13.5}$$

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

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

Inference: We implemented the deep Kalman smoother (DKS) inference model of (Krishnan et al., 2017), which follows equations (5.4)–(5.7). $e_{\overleftarrow{\mathbf{g}}}$ is implemented using a backward LSTM fed with an MLP($\mathbf{x}_t$, 256, Tanh). In order to keep a consistent structure for the links between different variables, the affine function of $\mathbf{z}$ in (5.4) is replaced with a two-layer MLP($\mathbf{z}_{t-1}$, 32, Tanh, 64, Tanh). $e_{\mathbf{z}}(\mathbf{g}_t)$ is implemented with an MLP($\mathbf{g}_t$, 64, Tanh, 32, Tanh).

13.1.2 KVAE

We recall that the KVAE model consists of a combination of a VAE and an LG-LDS (Linear-Gaussian Linear Dynamical System). Therefore, the implementation of KVAE is a bit specific. In practice, for training, the internal state sequence $\mathbf{a}_{1:T}$ is inferred from data input $\mathbf{x}_{1:T}$ frame by frame with a consistent encoder network. Then $\mathbf{a}_{1:T}$ is considered as the observation of the LG-LDS, which is solved with the Kalman filter (that computes $p(\mathbf{z}_t|\mathbf{a}_{1:t}, \mathbf{u}_{1:t})$) or with the Kalman smoother (that computes $p(\mathbf{z}_t|\mathbf{a}_{1:T}, \mathbf{u}_{1:T})$). We use the former in our experiments, which are set in predictive mode, i.e., we set $\mathbf{u}_t = \mathbf{a}_{t-1}$. These posterior distributions on $\mathbf{z}_t$ are then used to compute the most likely value of $\mathbf{a}_t$, denoted by $\tilde{\mathbf{a}}_t$, see (3.17). Finally, $\mathbf{x}_{1:T}$ can be generated from $\tilde{\mathbf{a}}_{1:T}$ frame by frame with a consistent decoder network.

VAE: The VAE encoder of KVAE, that is (6.5), is implemented as an MLP($\mathbf{x}_t$, 256, Tanh, 128, Tanh). The associated decoder, that is (6.15), is implemented as an MLP($\mathbf{a}_t$, 128, Tanh, 256, Tanh).

LG-LDS: The solution to the LG-LDS part of KVAE is obtained with the Kalman smoother, see (Murphy, 2012, Section 18.3) or (Bishop, 2006, Section 13.3). Moreover, in KVAE, the LDS parameters are actually a linear combination of LDS parameters taken in a predefined parameter bank of K LDSs $\{\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}\}_{k=1}^{K}$. This enables the KVAE to smoothly switch from one dynamical mode to another over time. Formally, we have:

$$\mathbf{A}_t = \sum_{k=1}^{K} \alpha^{(k)}(\mathbf{a}_{0:t-1})\mathbf{A}^{(k)} \tag{13.6}$$

$$\mathbf{B}_t = \sum_{k=1}^{K} \alpha^{(k)}(\mathbf{a}_{0:t-1})\mathbf{B}^{(k)} \tag{13.7}$$

$$\mathbf{C}_t = \sum_{k=1}^{K} \alpha^{(k)}(\mathbf{a}_{0:t-1})\mathbf{C}^{(k)}, \tag{13.8}$$

where $\alpha^{(k)}(\mathbf{a}_{0:t-1})$ is a normalized weight obtained with an LSTM network with input $\mathbf{a}_{t-1}$ and internal state of dimension 50, followed by a Softmax function. The parameters of this LSTM are learned during the KVAE training, as the other parameters. For training, all matrices $\mathbf{A}^{(k)}$ are initialized as the

identity matrix, all matrices $\mathbf{B}^{(k)}$ and $\mathbf{C}^{(k)}$ are initialized randomly with a scaled Gaussian sampling, $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ are initialized as isotropic matrices with pre-defined scale values, and $\mathbf{a}_0$ is zero-initialized.

Training procedure: In the original paper, once all weights and parameters are initialized, there is a two-step training procedure. First, the parameters of the VAE are trained, keeping the random initialisation of the LG-LDS parameters constant. Second, a joint VAE LG-LDS training step is run. Note that, at the time of first release of this review paper, we did not manage to achieve satisfactory training convergence of the KVAE model with this strategy in our experiments, and therefore we will not report the results for this model for the time being.

13.1.3 STORN

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

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

13.1.4 VRNN

We recall that, unlike STORN, VRNN employs a shared RNN for inference and generation, with 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: $d_{\mathbf{h}}(\varphi_{\mathbf{x}}(\mathbf{x}_{t-1}), \varphi_{\mathbf{z}}(\mathbf{z}_{t-1}), \mathbf{h}_{t-1})$ in (8.1) is implemented with an LSTM network with input $[\varphi_{\mathbf{x}}(\mathbf{x}_{t-1}), \varphi_{\mathbf{z}}(\mathbf{z}_{t-1})]$. $d_{\mathbf{z}}(\mathbf{h}_t)$ in (8.4) is implemented with an MLP($\mathbf{h}_t$, 64, Tanh, 32, Tanh). $d_{\mathbf{x}}(\varphi_{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_t)$ in (8.2) is implemented with an MLP($[\varphi_{\mathbf{z}}(\mathbf{z}_t), \mathbf{h}_t]$, 256, Tanh).

Inference: $e_{\mathbf{z}}(\varphi_{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_t)$ in (8.11) is implemented with an MLP($[\varphi_{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_t]$, 128, Tanh, 64, Tanh).

13.1.5 SRNN

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

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

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

13.1.6 RVAE

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

Generation: Regarding the causal case, $d_{\mathbf{h}}(\mathbf{z}_t, \mathbf{h}_{t-1})$ in (10.4) is a forward LSTM network with input MLP($\mathbf{z}_t$, 32, Tanh, 64, Tanh), and $d_{\mathbf{x}}(\mathbf{h}_t)$ in (10.5) is an MLP($\mathbf{h}_t$, 256, Tanh). For the non-causal case, the generation of $\mathbf{h}_t$ in (10.9)–(10.11) is implemented with a bi-directional 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, $e_{\overrightarrow{\mathbf{g}}}(\mathbf{z}_{t-1}, \overrightarrow{\mathbf{g}}_{t-1})$ in (10.18) is a forward LSTM network with input MLP($\mathbf{z}_{t-1}$, 32, Tanh, 64, Tanh), $e_{\overleftarrow{\mathbf{g}}}(\mathbf{x}_t, \overleftarrow{\mathbf{g}}_{t+1})$ in (10.19) is a backward LSTM network with input MLP($\mathbf{x}_t$, 256, Tanh), and $e_{\mathbf{z}}(\mathbf{g}_t)$ in (10.21) is an MLP($[\overrightarrow{\mathbf{g}}_t, \overleftarrow{\mathbf{g}}_t]$, 64, Tanh, 32, Tanh). In the non-causal case, $e_{\overrightarrow{\mathbf{g}}_{\mathbf{z}}}$ in (10.24) and $e_{\overleftarrow{\mathbf{g}}_{\mathbf{x}}}$ in (10.26) follow the same architecture as $e_{\overrightarrow{\mathbf{g}}}$ and $e_{\overleftarrow{\mathbf{g}}}$ in the causal inference model, respectively, and $e_{\overrightarrow{\mathbf{g}}_{\mathbf{x}}}$ in (10.25) is a forward LSTM with input MLP($\mathbf{x}_t$, 256, Tanh).

13.1.7 DSAE

We recall that, compared to the other models, DSAE has a extra sequence-level latent variable $\mathbf{v}$. We assume that $\mathbf{v}$ has the same dimension as $\mathbf{z}_t$. Since the total dimension of $\mathbf{z}_{1:T}$ is T times the dimension 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: $d_{\mathbf{h}}(\mathbf{z}_{t-1}, \mathbf{h}_{t-1})$ in (11.2) is a forward LSTM network with a hidden layer of dimension 32,[1] and $d_{\mathbf{z}}(\mathbf{h}_t)$ in (11.3) is a one-layer linear network to project the dimension of $\mathbf{h}_t$ onto the dimension of $\mathbf{z}_t$. $d_{\mathbf{x}}(\mathbf{z}_t, \mathbf{v})$ 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 bi-directional many-to-one LSTM network with input MLP($\mathbf{x}_t$, 256, Tanh) and output $\mathbf{g}^{\mathbf{v}}$, followed by an MLP($\mathbf{g}^{\mathbf{v}}$, 64, Tanh, 32, Tanh). As for the inference of $\mathbf{z}_t$, Eqs. (11.17)–(11.19) are implemented with a bi-directional LSTM with input MLP([$\mathbf{v}$, MLP($\mathbf{x}_t$, 256, Tanh)], 256, Tanh). Finally, $e_{\mathbf{z}}(\mathbf{g}_t^{\mathbf{z}})$ in (11.20) is an RNN with hidden layer of dimension 32.

13.2 Experimental Protocol

13.2.1 Dataset and preprocessing

For the present speech analysis-resynthesis task, we used the Wall Street Journal (WSJ0) dataset (Garofolo et al., 1993), which consists of speech read from Wall Street Journal news. We used the speaker-independent, medium (5k words) vocabulary subset of the corpus. More precisely, the *si_tr_s* subset (around 25 hours) was used for training, the *si_dt_05* subset (around 2 hours) was used for validation, and the *si_et_05* subset (around 1.5 hours) was used for testing.

The raw speech waveform was sampled at 16 kHz. Analysis-resynthesis was processed with the DVAEs in the time-frequency domain. To this aim, the speech signals were preprocessed with the short-time Fourier transform (STFT) using a 32-ms sine window (512 samples) with 50%-overlap to obtain sequences of 257-dimensional discrete Fourier spectra (for positive frequencies). We set $T = 150$, meaning that speech utterances of 2.4 s were extracted from the raw dataset and pre-processed with the STFT. In summary, each speech sequence is a 150×257 STFT spectrogram. This data preprocessing resulted in a set of $N_{\text{tr}} = 13,272$ training sequences (about 9 hours of speech signal) and $N_{\text{val}} = 2,143$ validation sequences (about 1.5 hour). For testing, we used the STFT spectrogram of each complete test sequence (with beginning and ending silence portions removed), which can be of variable length, most often larger than 2.4 s.

As discussed in (Leglaive et al., 2020) and briefly 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. In practice, 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 non-negative) 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

[1] Here we set 32 and not 128 because $\mathbf{h}_t$ is an internal state vector of an RNN with input $\mathbf{z}_t$.

distribution with variance $\sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot)^2$ 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., $x_{f,t} \sim \mathcal{G}(1, 1/\sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot))$. This also amounts to using the Itakura-Saito distance between $x_{f,t}$ and $\sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot)$ in the reconstruction term of the VLB, see (Girin et al., 2019). We remind that all presented DVAE models are very versatile regarding the output data conditional pdf model, 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. We can note that the linear layer estimating the parameters of $\mathbf{x}_t$ has 257 output units corresponding to the log-variance parameters $\{\ln \sigma^2_{\theta_{\mathbf{x}},f,t}(\cdot)\}_{f=1}^F$.

13.2.2 Training and testing

All the tested models were implemented in PyTorch (Paszke et al., 2019).[3] In order to train the models we used mini-batch stochastic gradient descent, in particular the Adam optimizer (Kingma and Ba, 2014), with a learning rate of 0.0001 and a batch size of 32. We also use early stopping on the validation set with a patience of 20 epochs. Once a model is trained on the training set, with early stopping on the validation set, its weights are fixed, and the model is run on the test set. We then report the average performance on the test set using the metrics presented in the next subsection.

13.2.3 Evaluation metrics

We used three metrics to evaluate the resynthesized speech quality and compare the performance of the different DVAE models: The root mean squared error (RMSE) between the original and reconstructed waveforms, Perceptual Evaluation of Speech Quality (PESQ) scores (Rix et al., 2001) and Short-Time Objective Intelligibility (STOI) scores (Taal et al., 2010). The amplitude of each original speech waveform is normalized in $[-1, 1]$, so the RMSE is (positive and) generally much lower than 1, and it directly represents a percentage of the maximum amplitude value. PESQ scores are in $[-0.5, 4.5]$ and STOI scores are in $[0, 1]$. For both, the higher the better.

13.3 Results

We first present the loss curves (i.e., VLB up to a constant term) obtained on the training data and the validation data in Fig. 13.1. Those curves suggest that the training procedure has successfully converged for all the implemented DVAE models. We observe certain differences in the convergence behavior of

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

[3] The code will be made freely available for non-commercial purposes.

DVAE	RMSE	PESQ	STOI
VAE	0.0510	2.05	0.86
DKF	0.0344	3.30	0.94
STORN	0.0338	3.05	0.93
VRNN	0.0267	3.60	0.96
SRNN	**0.0248**	**3.64**	**0.97**
RVAE-Causal	0.0499	2.27	0.89
RVAE-NonCausal	0.0479	2.37	0.89
DSAE	0.0469	2.32	0.90

Table 13.1: Performance of the tested DVAE models in our speech analysis-resynthesis experiment. The RMSE, PESQ and STOI scores are averaged over the test subset of the WSJ0 dataset.

the different models, but a thorough examination of these aspects is beyond the scope of this paper.

The values of the three metrics described in the previous section averaged over the above-described test dataset at the end of model training are reported in Table 13.1. From this table, we can draw the following comments:

- First, all tested DVAE models lead to "correct" signal reconstruction, with an RMSE that represents only a few percents (generally lower than 5%) of the maximum waveform amplitude. The quality of the reconstructed speech signals, as measured by PESQ, goes from fair to good. STOI scores, generally higher than 0.90 show their good intelligibility.

- All DVAE models outperform the standard VAE model. This demonstrates the interest of including temporal modeling in the VAE framework for modeling sequential data such as speech signals.

- VRNN and SRNN are the two methods with highest performance, with a notable gain in performance over all other models, and SRNN is slightly better than VRNN. By looking at the associated probabilistic models, we can observe that both SRNN and VRNN contain many dependencies with the past observed and latent variables, in contrast to the other implemented DVAEs, which only depend mildly on the past observed and latent variables. We believe that these dependencies allow VRNN and SRNN to better capture the temporal patterns associated to the speech signal.

- When it comes to SRNN performing slightly better than VRNN, this could be due to the fact that the inference model of SRNN respects the structure of the exact posterior distribution while the inference model of VRNN (as proposed in the original paper) does not. Indeed, in both cases the exact posterior of any latent variable $\mathbf{z}_t$ depends on all observations $\mathbf{x}_{1:T}$. However, the inference model of VRNN (as presented in the original paper) takes into account only the causal observations $\mathbf{x}_{1:t}$.

- The performance scores of DKF and STORN are quite equivalent, but below those of SRNN and VRNN. This is likely due to the fact that the temporal dependencies in DKF and STORN are less rich than in VRNN and SRNN. DKF is an SSM-like model, where there is no explicit temporal dependency between $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, but only between $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$. Somehow it is the other way round for STORN. In fact, in STORN $\mathbf{x}_t$ depends on $\mathbf{x}_{1,t-1}$ and $\mathbf{z}_{1:t}$, so one could think that this model is richer that DKF and should have better performance. However, we found out that DKF slightly outperforms STORN in terms of PESQ and STOI (but not in RMSE). We can hypothesize that, as for SRNN vs. VRNN, the difference in performance between DKF and STORN is (at least partly) due to the fact that the inference model of DKF does respect the structure of the exact posterior distribution whereas the inference model of STORN does not (for STORN, the inference of $\mathbf{z}_t$ depends only on $\mathbf{x}_{1:t}$ and not on $\mathbf{z}_{t-1}$ nor $\mathbf{x}_{t+1:T}$). Another possible explanation for the differences in performance between DKF and STORN is that the prior distribution of the latent vectors $\{\mathbf{z}_t\}_{t=1}^T$ is i.i.d. in STORN, while it has temporal dependencies in DKF. In a general manner, we hypothesize that models with i.i.d. prior over time risk to underperform w.r.t. models that have a prior temporal model on $\mathbf{z}_t$ or that are defined via a temporal generative model of $\mathbf{z}_t$ (like VRNN and SRNN). This is because the $\mathbf{z}_{1:T}$ sequence is assumed to encode high-level characteristics of the data $\mathbf{x}_{1:T}$ that generally evolve smoothly over time (at least for some of these characteristics). This is not ensured by the i.i.d. standard Gaussian prior distribution of $\mathbf{z}_t$ used in STORN.

- The performance of DSAE is quite disappointing, especially compared to the performance of DKF. Indeed, as DKF, DSAE is also an SSM-like model. Actually, DSAE can be seen 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 for DKF). Again, this poor performance could come from the structure of the inference model. In particular, for DSAE, the inference model of $\mathbf{z}_t$ depends on $\mathbf{x}_{1:T}$, whereas the exact posterior distribution depends on $\mathbf{z}_{1:t-1}$ and $\mathbf{x}_{t:T}$. So 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).

- As for RVAE, this model exhibits the worst performance of all the implemented DVAE models. Here also, i.i.d. modeling of $\mathbf{z}_t$ may be suboptimal. In addition to that, there is no explicit modeling of the temporal dependencies on $\mathbf{x}_t$ (e.g., $\mathbf{x}_t$ does not depend on $\mathbf{x}_{t-1}$), hence leading to a model with weak "predictive power." The non-causal version of RVAE is slightly better than the causal version, which was expected. We can note that (causal) RVAE is only slightly better than VAE in terms of RMSE. However, even if still small, the difference with VAE is a little bit larger in

terms of PESQ and STOI, so RVAE does capture some part of the speech dynamics, but in a quite limited extent.

In summary, in a practical application requiring the modeling of speech spectrograms, one should choose VRNN or SRNN. Also, at the light of the above results and associated discussion, we suspect that having an inference model that respects the exact variable dependencies at inference time is very important for obtaining optimal performance. Of course, this is not always possible, since some applications require a causal inference model for online processing.

We insist on the fact that the above "model ranking" is valid only for the presented experiments, which consist of pure analysis-resynthesis of speech spectrograms. For other tasks such as speech signal generation (from new values of the latent vectors) or speech signal transformation (with modification of the latent vectors), we have no claim on how the implemented models would behave. In particular, it is very difficult to know how much of the information about $\mathbf{x}_{1:T}$, and what kind of information, is encoded into $\mathbf{z}_{1:T}$, and in particular, we do not know about the "disentanglement power" of each model. And importantly, there is no clear methodology to address those issues, i.e., to evaluate the generated data and the representation power of the latent variable. Those points will be rediscussed more extensively in Section 14. Experiments dedicated to illustrate them are out of the scope of the present paper.

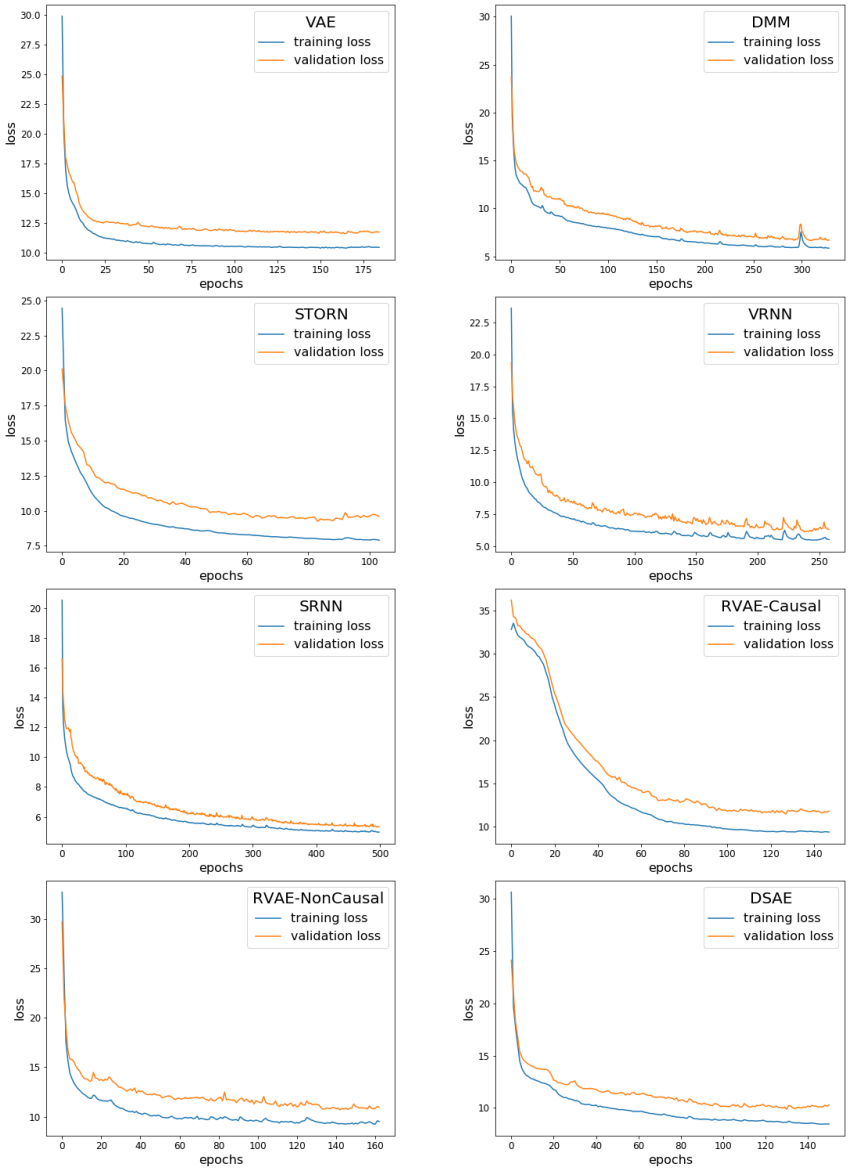

Figure 13.1: Training curves (for training and validation datasets) for the different tested DVAE models (indicated in the legend).

Chapter 14

Discussion

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

14.1 Fundamental motivation for DVAEs

The fundamental motivation for designing and using DVAEs is to mix the world of dynamical models, aimed at modeling the dynamics of sequential data, and the world of 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 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 a lot of flexibility and modeling power to the conditional output density. Let us here quote the authors of (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."

In the same vein, we can point out the superiority of DVAE over classical (non-deep) DBNs and SSMs thanks to the deep non-linear layers of information processing. Again, let us quote the authors of (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, this kind of statement applies to the whole DVAE family of models.

14.2 DVAE outcome: A story of flexibility

14.2.1 Flexibility of the generative model(s)

As we have seen in this review, a lot of different possible generative models can be derived from the general form (4.4) by simplifying variable dependencies. The models we have reviewed in detail, STORN, VRNN, SRNN, etc, are some instances of these possible generative models, but there are other possibilities. Moreover, each model has several variants: Driven/undriven mode, predictive/non-predictive mode, and with one or several feature extractors.

When designing the generative model, complexity issues may be considered. For example, we can cite the authors of (Bayer and Osendorfer, 2014):

> "[...] we can restrict ourselves to prior distributions over the latent variables that factorise 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 $\mathbf{z}_t$."

But at the same time, the systematic aspect of the VAE methodology (and the versatility of 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, see for example our remark in Section 11.1: In this framework, not only it is easy to go from a linear dynamical model to a non-linear one, but it is also easy to go from a first-order temporal model to (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 non-linearities, which is why we have to define an inference

model in addition to the generative model (we cannot apply Bayes rule analytically). However, a key feature of DVAEs with respect to standard VAEs is that we have to 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 (induced by the chosen generative model), that is the dependencies between a latent random vector at a given time step and the remaining latent and observed sequential data. 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, we have seen that several seminal papers on DVAEs did not follow this "consistency principle," and more importantly did not justify 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 non-causal processing of the observations, one may drop the non-causal dependencies for the purpose of online applications. Simplifying the posterior dependencies may also be motivated by the computational complexity of inference.

Another key difference between DVAEs and VAEs relates to how the VLB (or actually an estimate of the VLB) 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 due to the use of RNNs, and it 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, a lot of different possibilities exist for the high-level implementation of DVAEs: We recall that (many) different developed probabilistic model representations can correspond to the same compact representation. Indeed, the compact form describes all the parent-child relationships between 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 one specific model corresponding to the developed representation. Indeed, this representation allows to understand how the dependencies are implemented in practice, and therefore over which parameter space the model is optimised. We have seen that this representation typically involves some kind of recurrent architecture. While such architectures allow to encode 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 very important in DVAE, and we encourage

to systematically present both representations when presenting and discussing DVAEs, as we did in this review.

Once the high-level DVAE architecture is chosen, a lot of different possibilities also exist for the low-level implementation: Network type (e.g., LSTM vs GRU) and low-level (hyper-)parameterization (number of layers in a network, number of units per layers, type of activation function, and classical deep learning modules such as batch normalisation for example). We chose not to deal extensively with these low-level implementation aspects in this review, and considered them instead as deep learning routine. Of course, all those choices (or at least part of them) depend on data nature and datasets, and can significantly influence modeling performance.

14.2.4 Other network architectures for sequential data modeling

In this review, we focused 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 an infinite-order temporal modeling, CNNs generally have a fixed-length receptive field implying a finite-order temporal modeling. In particular, temporal convolutional networks (TCNs) are getting more and more popular due to 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 1-dimensional convolutions, sharing similarities with the Wavenet architecture (van den Oord et al., 2016), and just as an RNN it outputs a sequence of the same length as the input sequence. Combining TCNs with VAEs has been explored in (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 very few works have considered leveraging transformers for generative modeling in the VAE framework. We only found 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 DVAEs and latent factors disentanglement

As stated in (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."

In a general manner, a DVAE mixes such (stochastic) unsupervised representation learning with a (deterministic) temporal encoding-decoding model. One general challenge here is to separate the data dynamics (i.e., their temporal structure) and other factors of variations (e.g., speaker identity for speech data, image content in terms of objects for 2D-images and videos).

Latent factors disentanglement is not necessarily natural nor efficient in the standard VAE, it somehow has to be "encouraged" (Higgins et al., 2017). The same issue holds with DVAEs. Moreover, as we already started to report in Section 2.5, one general problem when mixing latent factors learning with a deterministic temporal model, is that, if the latter is powerful (e.g., an autoregressive model, implemented with an RNN), it can have a tendency to capture all of data information without the latent variable being actually used. This problem has been reported in several papers, in particular for many-to-one encoding / one-to-many decoding architectures dedicated either to image modeling, e.g., (Chen et al., 2017; Lucas and Verbeek, 2018) or text/dialog modeling, e.g., (Bowman et al., 2016; Serban et al., 2016). Beyond the general disentanglement challenge, this problem is linked with the difficulty for a single latent vector $\mathbf{z}$ to efficiently encode the content of a long input sequence. As we reported before, different solutions have been proposed in the literature to tackle this problem, e.g., "weakening" the decoder or structuring the latent representation with hierarchical models (see Section 2.5). This generally yields a more influential and more disentangled latent representation. Yet it seems that there is still room for improvement. Note that 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).

The DVAEs we focused on in this review do not consider a single latent vector $\mathbf{z}$ for a data sequence, but they rather consider a latent vector sequence $\mathbf{z}_{1:T}$, that 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 works done in 2D-image or language/text modeling for example. A first remark that is worth mentioning, although quite trivial, is that this DVAE configuration first solves the encoding capacity problem for large data sequences: Here, as mentioned in (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." Of course, this generally comes at the price of additional computational complexity and "coding cost" for the latent representation.

Then, in such DVAEs, the problem of separating data dynamics and other causal factors of variations takes a new flavor since the latter are here allowed to have their own dynamics. In this context, one way to address the disentanglement challenge is still to apply hierarchical modeling to the latent factors, but here on the time dimension, that is to design models with a different time resolu-

tion for different latent variables.[1] 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 signals modeling, this appears as a promising way to separate the modeling and control of phonetic information (defined at the segment or frame level) and speaker/session information (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, that can be significantly different from the data dynamics.

In a general manner, the issue of separating dynamics and other factors of variation is still largely open in the literature of DVAE models with a sequence of latent vectors. For example, we were surprised to notice that there is in general very few experiments and information on the explainability of the extracted sequence of latent factors. Experiments consisting of swapping the extracted latent factors across two data sequences before resynthesizing them were reported in, e.g., (Hsu et al., 2017b). Those experiments show for example that for speech signals, speaker 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 separately speech production factors remain quite open. Moreover, basic questions such as the impact of the size of $\mathbf{z}_t$ and $\mathbf{h}_t$ on modeling quality and 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, whereas the size of $\mathbf{h}_t$ is kept comparable to the size of data $\mathbf{x}_t$?

Note that we can read in (Hsu et al., 2017b) (in 2017) that

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

About SRNN (Fraccaro et al., 2016), VRNN (Chung et al., 2015), and SVAE (Johnson et al., 2016), the authors of (Hsu et al., 2017b) say:

> "[...] 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, STORN, VRNN, SRNN, etc, provide an elegant and powerful mathematical and methodological framework for sequential data representation learning, but it seems that there is still a lot of work to do on the disentanglement challenge. Solutions for the disentanglement of $\mathbf{z}_t$, inspired by existing structured or hierarchical static models, still have to be developed.

[1] This approach is not incompatible with other types of hierarchical models of course.

14.4 Perspectives in source coding

Although the VAE and DVAE are in essence an excellent framework for extracting efficient and compact data representations, there are relatively few studies on their practical application to data source coding, i.e., 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 the VAE as a lossy coder (Kingma et al., 2016; Chen et al., 2017). Among the very few papers on practical application of (D)VAE to data coding, we can mention the ConvDRAW model of (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 (that is a mix of VAE and vector quantization of $\mathbf{z}$) applied to speech coding (van den 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 the two recent papers (Lombardo et al., 2019) and (Yang et al., 2020). In (Lombardo et al., 2019), the authors present a video codec based on the DSAE model (Li and Mandt, 2018) that we reviewed in detail in Section 11. The sequence of latent vectors $\mathbf{z}_{1:T}$ extracted by the DSAE encoder (see Section 11.1) 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 are comparable to the state-of-the art video codecs (such as VP9) on generic video sequences while drastically improving these performance on video sequences with specialized content (similar to the content of videos used to train the model). In this paper, no information is given on the control of the coded data sequence quality or of the bitrate. The paper only mentions that the arithmetic encoding of $\mathbf{z}_{1:T}$ requires a number of iterations. Moreover, the DSAE model is an SSM-like model, i.e. $\mathbf{x}_t$ is generated from $\mathbf{z}_t$ "alone," not considering the potential of using $\mathbf{x}_{t-1}$ (or its quantized version) for predicting and encoding more efficiently $\mathbf{x}_t$.

In contrast, in (Yang et al., 2020), the authors propose 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 the decoder, and a feedback connection from decoder to encoder. The recurrent connections are reminiscent of the *predictive coding* principle that is classical in the source coding theory (Gersho and Gray, 2012): FRAE can be interpreted as a *non-linear predictive*

coding scheme, where the encoder forms a latent code which encodes only the residual information that is missing when reconstructing a data vector from the deterministic internal state, which depends on past data vectors. Of course, this concept of predictive coding is strongly connected to the concept of 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 DVAE models. Therefore, from this point of view, FRAE is strongly connected to STORN, VRNN, and SRNN. The feedback connection from decoder to encoder is reminiscent of another classical principle of source coding that is *closed-loop coding* (Gersho and Gray, 2012) even if the authors of (Yang et al., 2020) do not refer to it explicitly. In short, it enables the decoder to use the quantized previous vectors in place of the unquantized ones (not available at the decoder) for the prediction of the current vector.

This line of research on non-linear predictive coders based on DVAEs is quite promising and is only at its infancy. As the authors of (Yang et al., 2020) write:

> "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 meets a concluding remark by the authors 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."

Chapter 15

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

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

For concision, 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 Fig. 7.1, the joint distribution between all variables is given by:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{h}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t) p_\theta(\mathbf{h}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1}) p(\mathbf{z}_t). \tag{15.1}$$

Since $\mathbf{h}_t$ 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 $f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1})$, i.e.:

$$p_\theta(\mathbf{h}_t|\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1}) = \delta(\mathbf{h}_t; f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1})). \tag{15.2}$$

Let us redenote with $\mathbf{d}_{1:T}$ the sequence $\mathbf{h}_{1:T}$ seen as a (deterministic) function of $\mathbf{x}_{1:T}$ and $\mathbf{z}_{1:T}$ only, i.e., 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 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

(15.1) with respect to $\mathbf{h}_{1:T}$ leads to:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \int_{\mathcal{R}^{H\times T}} \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t)\delta(\mathbf{h}_t; f_h(\mathbf{x}_{t-1}, \mathbf{z}_t, \mathbf{h}_{t-1}))p(\mathbf{z}_t)d\mathbf{h}_{1:T} \quad (15.3)$$

$$= \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{d}_t)p(\mathbf{z}_t). \quad (15.4)$$

To go from (15.3) to (15.4), 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. From now on, 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. We thus have:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t)p(\mathbf{z}_t) = \prod_{t=1}^{T} p_\theta\big(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\big)p(\mathbf{z}_t). \quad (15.5)$$

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

$$p_\theta(\mathbf{x}_{1:T}|\mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{h}_t) = \prod_{t=1}^{T} p_\theta\big(\mathbf{x}_t|\mathbf{h}_t(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})\big). \quad (15.6)$$

We insist on the fact 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.

Bibliography

E. Aksan and O. Hilliges. STCN: Stochastic temporal convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2019.

E. Archer, I. M. Park, L. Buesing, J. Cunningham, and L. Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. In *International Conference on Learning Representations (ICLR)*, 2018.

S. Bai, Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Y. Bando, M. Mimura, K. Itoyama, K. Yoshii, and T. Kawahara. 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, 2018.

J. Bayer and C. Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

A. Bitton, P. Esling, and T. Harada. Neural granular sound synthesis. *arXiv preprint arXiv:2008.01393*, 2020.

M. Blaauw and J. Bonada. Modeling and transforming speech using variational autoencoders. In *Conference of the International Speech Communication Association (Interspeech)*, San Francisco, CA, 2016.

D. Bouchacourt, R. Tomioka, and S. Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *AAAI Conference on Artificial Intelligence*, 2018.

N. Boulanger-Lewandowski, Y. Bengio, P. Vincent, P. Gray, and C. Naguri. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *International Conference on Machine Learning (ICML)*, 2012.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *International Conference on Computational Natural Language Learning (CoNLL)*, 2016.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. In *International Conference on Learning Representations (ICLR)*, 2017.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pages 2980–2988, 2015.

F. Daum. Nonlinear filters: Beyond the Kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, 20(8):57–69, 2005.

Z. Deng, R. Navarathna, P. Carr, S. Mandt, Y. Yue, I. Matthews, and G. Mori. Factorized variational autoencoders for modeling audience reactions to movies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

E. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems*, pages 4414–4423, 2017.

J. Durbin and S. J. Koopman. *Time series analysis by state space methods*. Oxford university press, 2012.

P. Esling, A. Chemla-Romeu-Santos, and A. Bitton. Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre space. In *International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

O. Fabius and J. R. van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

V. Fortuin, D. Baranchuk, G. Rätsch, and S. Mandt. GP-VAE: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1651–1661, 2020.

E. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Bayesian nonparametric inference of switching dynamic linear models. *IEEE Transactions on Signal Processing*, 59(4):1569–1585, 2011.

M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, 2016.

M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3601–3610, 2017.

Z. Gan, C. Li, R. Henao, D. E. Carlson, and L. Carin. Deep temporal sigmoid belief networks for sequence modeling. In *Advances in Neural Information Processing Systems*, pages 2467–2475, 2015.

C. Gârbacea, A. van den Oord, Y. Li, F. S. Lim, A. Luebs, O. Vinyals, and T. C. Walters. Low bit-rate speech coding with vq-vae and a wavenet decoder. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 735–739, 2019.

J. Garofolo, D. Graff, D. Paul, and D. Pallett. Csr-i (wsj0) sennheiser ldc93s6b. https://catalog.ldc.upenn.edu/ldc93s6b. *Philadelphia: Linguistic Data Consortium*, 1993.

D. Geiger, T. Verma, and J. Pearl. Identifying independence in bayesian networks. *Networks*, 20(5):507–534, 1990.

A. Gersho and R. M. Gray. *Vector quantization and signal compression*. Springer Science & Business Media, 2012.

L. Girin, F. Roche, T. Hueber, and S. Leglaive. Notes on the use of variational autoencoders for speech and audio spectrogram modeling. In *Digital Audio Effects Conference (DAFx)*, 2019.

I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, 2014.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, pages 6713–6723, 2017.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. *International Conference on Machine Learning (ICML)*, 2015.

K. Gregor, F. Besse, D. J. Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression. In *Advances In Neural Information Processing Systems*, pages 3549–3557, 2016.

S. Gu, Z. Ghahramani, and R. E. Turner. Neural adaptive sequential Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 2629–2637, 2015.

I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville. PixelVAE: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

A. Habibian, T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen. Video compression with rate-distortion autoencoders. In *IEEE International Conference on Computer Vision*, pages 7033–7042, 2019.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

J. D. Hamilton. *Time series analysis*. Princeton University Press, 2020.

S. Haykin. *Kalman filtering and neural networks*, volume 47. John Wiley & Sons, 2004.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-vae: learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

G. Hinton, P. Dayan, B. Frey, and R. Neal. The "Wake-Sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

W.-N. Hsu, Y. Zhang, and J. Glass. Learning latent representations for speech generation and transformation. *arXiv preprint arXiv:1704.04222*, 2017a.

W.-N. Hsu, Y. Zhang, and J. Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems*, pages 1878–1889, 2017b.

Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Toward controlled generation of text. In *International Conference on Machine Learning (ICML)*, pages 1587–1596, 2017.

K. Huang, N. D. Sidiropoulos, and A. P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19):5052–5065, 2016.

M. Jang, S. Seo, and P. Kang. Recurrent neural network-based semantic variational autoencoder for sequence-to-sequence learning. *Information Sciences*, 490:59–73, 2019.

J. Jiang, G. Xia, D. Carlton, C. Anderson, and R. Miyakawa. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 516–520, 2020.

M. J. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, pages 2946–2954, 2016.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

H. Kameoka, L. Li, S. Inoue, and S. Makino. Semi-blind source separation with multichannel variational autoencoder. *arXiv preprint arXiv:1808.00892*, 2018.

M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational Bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations (ICLR)*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*, 12:307–392, 2019.

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

D. Koller and N. Friedman. *Probabilistic graphical models: Principles and techniques*. MIT press, 2009.

R. Krishnan, U. Shalit, and D. Sontag. Deep Kalman filters. In *arXiv preprint arXiv:1511.05121*, 2015.

R. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *AAAI Conference on Artificial Intelligence*, 2017.

V. Kuleshov, A. Chaganty, and P. Liang. Tensor factorization via matrix factorization. In *Artificial Intelligence and Statistics*, pages 507–516, 2015.

C. Lea, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision (ECCV)*, pages 47–54, 2016.

J. Y. Lee, S. J. Cheon, B. J. Choi, N. S. Kim, and E. Song. Acoustic modeling using adversarially trained variational recurrent neural network for speech synthesis. In *Interspeech*, pages 917–921, 2018.

S. Leglaive, L. Girin, and R. Horaud. A variance modeling framework based on variational autoencoders for speech enhancement. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2018.

S. Leglaive, L. Girin, and R. Horaud. Semi-supervised multichannel speech enhancement with variational autoencoders and non-negative matrix factorization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

S. Leglaive, X. Alameda-Pineda, L. Girin, and R. Horaud. A recurrent variational autoencoder for speech enhancement. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020.

Y. Li and S. Mandt. Disentangled sequential autoencoder. In *International Conference on Machine Learning (ICML)*, pages 5670–5679, 2018.

S. W. Linderman, A. C. Miller, R. P. Adams, D. M. Blei, L. Paninski, and M. J. Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

D. Liu and G. Liu. A Transformer-based variational autoencoder for sentence generation. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2019.

S. Lombardo, J. Han, C. Schroers, and S. Mandt. Deep generative video compression. In *Advances in Neural Information Processing Systems*, pages 9287–9298, 2019.

C. Louizos and M. Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning (ICML)*, pages 2218–2227, 2017.

T. Lucas and J. Verbeek. Auxiliary guided autoregressive variational autoencoders. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 443–458, 2018.

Y. Luo and N. Mesgarani. Conv-Tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 27(8):1256–1266, 2019.

M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048, 2016.

Y. Miao, L. Yu, and P. Blunsom. Neural variational inference for text processing. In *International Conference on Machine Learning (ICML)*, pages 1727–1736, 2016.

D. Miladinović, M. W. Gondal, B. Schölkopf, J. M. Buhmann, and S. Bauer. Disentangled state space representations. *arXiv preprint arXiv:1906.03255*, 2019.

A. Mnih and K. Gregor. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

V. M. Moreno and A. Pigazo. *Kalman filter: Recent advances and applications*. BoD–Books on Demand, 2009.

K. P. Murphy. Switching Kalman filters. *Unpublished technical report – Available online*, 1998.

K. P. Murphy. *Machine learning: A probabilistic perspective*. MIT press, 2012.

F. D. Neeser and J. L. Massey. Proper complex random processes with applications to information theory. *IEEE Transactions on Information Theory*, 39 (4):1293–1302, 1993.

A. Papoulis. *Signal analysis*. McGraw-Hill New York, 1977.

M. Pariente, A. Deleforge, and E. Vincent. A statistically principled and computationally efficient approach to speech enhancement using variational autoencoders. *arXiv preprint arXiv:1905.01209*, 2019.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.

J. Pereira and M. Silveira. Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1275–1282, 2018.

T. Raiko and M. Tornio. Variational Bayesian learning of nonlinear hidden state-space models for model predictive control. *Neurocomputing*, 72(16-18): 3704–3712, 2009.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning (ICML)*, 2014.

A. Rix, J. Beerends, M. Hollier, and A. Hekstra. 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)*, 2001.

A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A hierarchical latent vector model for learning long-term structure in music. *arXiv preprint arXiv:1803.05428*, 2018.

F. Roche, T. Hueber, S. Limier, and L. Girin. Autoencoders for music sound synthesis: A comparison of linear, shallow, deep and variational models. In *Sound and Music Conference*, 2019.

T. Salimans. A structured variational auto-encoder for learning deep hierarchies of sparse features. *arXiv preprint arXiv:1602.08734*, 2016.

S. Semeniuta, A. Severyn, and E. Barth. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*, 2017.

I. V. Serban, A. G. Ororbia, J. Pineau, and A. Courville. Piecewise latent variables for neural variational text processing. *arXiv preprint arXiv:1612.00377*, 2016.

I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI Conference on Artificial Intelligence*, 2017.

W. Shang, K. Sohn, and Y. Tian. Channel-recurrent autoencoding for image modeling. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1195–1204, 2018.

N. Siddharth, B. Paige, J.-W. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.

V. Šmídl and A. Quinn. *The variational Bayes method in signal processing*. Springer Science & Business Media, 2006.

K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3483–3491, 2015.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3738–3746, 2016a.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. How to train deep variational autoencoders and probabilistic ladder networks. In *International Conference on Machine Learning (ICML)*, 2016b.

J. Su, S. Wu, D. Xiong, Y. Lu, X. Han, and B. Zhang. Variational recurrent neural machine translation. In *AAAI International Conference on Artificial Intelligence*, 2018.

I. Sutskever. *Training recurrent neural networks*. University of Toronto, 2013.

C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4214–4217, 2010.

S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. MoCoGan: Decomposing motion and content for video generation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1526–1535, 2018.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016a.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learning (ICML)*, 2016b.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. In *International Conference on Learning Representations (ICLR)*, 2017.

P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11: 3371–3408, 2010.

E. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

T. Wang and X. Wan. T-cvae: Transformer-based conditioned variational autoencoder for story completion. In *AAAI International Joint Conference on Artificial Intelligence*, pages 5233–5239, 2019.

M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, pages 2746–2754, 2015.

C. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT press, Cambridge, MA, 2006.

Y. Yang, G. Sautière, J. J. Ryu, and T. S. Cohen. Feedback recurrent autoencoder. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3347–3351, 2020.

Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learning (ICML)*, pages 3881–3890, 2017.

T. Zhao, R. Zhao, and M. Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*, 2017.

T. Zhao, K. Lee, and M. Eskenazi. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*, 2018.

Y. Zhao and I. M. Park. Variational online learning of neural dynamics. *arXiv preprint arXiv:1707.09049*, 2017.

Y. Zhao, J. Nassar, I. Jordan, M. Bugallo, and I. M. Park. Streaming variational Monte Carlo. *arXiv preprint arXiv:1906.01549*, 2019.