UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

SEMI-SUPERVISED LEARNING WITH TEMPORAL VARIATIONAL AUTO-ENCODERS FOR THE DIAGNOSIS OF FAILURE SEVERITIES AND THE PROGNOSIS OF REMAINING USEFUL LIFE

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA

GABRIEL ANTONIO SAN MARTÍN SILVA

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
VIVIANA MERUANE NARANJO
JUAN TAPIA FARÍAS

Este trabajo ha sido parcialmente financiado por Beca Magíster Nacional Conicyt

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: GABRIEL ANTONIO SAN MARTÍN SILVA
FECHA: 2020
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

SEMI-SUPERVISED LEARNING WITH TEMPORAL VARIATIONAL AUTO-ENCODERS FOR THE DIAGNOSIS OF FAILURE SEVERITIES AND THE PROGNOSIS OF REMAINING USEFUL LIFE

Within the field of prognosis and health management, one of the areas that have grown the most in recent years is the research and application of AI techniques and deep learning models to prongosis and diagnosis tasks. There are many possible reasons for this, but maybe the most important one is that together with the capability of building more complex machines and systems, the amount of operation data being generated has increased exponentially. Nevertheless, while larges quantities of data are being generated now than ever, the cost of labeling that data has not changed. In that regard, it becomes very necessary to focus the research efforts towards models and approaches that can learn to predict and detect failures from lower amounts of labeled information, possibly extracting it from the unlabeled portion as well.

In this regard, the principal motivation for this thesis is the need for reliable models for diagnosis and prognosis tasks that can learn in a semi-supervised fashion, taking advantage of both labeled and unlabeled information.

The main objective of this thesis is to assess the improvements achieved in both prognosis and diagnosis tasks with the proposed model that uses a coupled training strategy to optimize simultaneously a temporal VAE (an unsupervised model) and a normal neural network predictor (a fully supervised model) for situations where the amount of labeled information is scarce.

The methodology used in this work consists mainly of four steps. First, a literature review of recent literature regarding semi-supervised approaches and deep generative models is done. Second, a modified version of Variational Auto-Encoder that uses normalizing flows and allows the processing and generation of sequential input and latent variables is proposed and explain. Third, the coupled training process is developed. Finally, the proposed model with the new training strategy is apply on two case studies to obtain results and discussed its advantages and disadvantages with respect to traditional approaches.

The main conclusions extracted from the results of the two case studies presented is that while an improvement of the proposed model is observed with respect to a decoupled training strategy, further experiments demonstrate that it only produces similar prediction performances when compared against a traditional recurrent neural network trained with only labeled data. Future research could focus on trying more advanced Deep Generative Models with the same training approach or testing more complex datasets where generalization is harder than the ones presented in this thesis.

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: GABRIEL ANTONIO SAN MARTÍN SILVA
FECHA: 2020
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

SEMI-SUPERVISED LEARNING WITH TEMPORAL VARIATIONAL AUTO-ENCODERS FOR THE DIAGNOSIS OF FAILURE SEVERITIES AND THE PROGNOSIS OF REMAINING USEFUL LIFE

Dentro del manejo de activos físicos, una de las áreas que más ha crecido en los últimos años es la investigación y aplicación de técnicas de relacionadas a la inteligencia artificial y modelos de aprendizaje profundo para las tareas de diagnóstico y pronóstico en maquinaria y sistemas. Existen muchas razones posibles para esto, pero quizás la más importante es que, junto con la capacidad de diseñar máquinas y sistemas más complejos, la cantidad de datos relacionados a la operación y mantenimiento que se genera actualmente ha aumentado exponencialmente. Sin embargo, aunque ahora se están generando más datos que que nunca, el costo de etiquetar esos datos no ha cambiado. En ese sentido, se hace muy necesario enfocar los esfuerzos de investigación hacia modelos y técnicas que puedan aprender a predecir y detectar fallas usando poca información etiquetada, suplementandola de la parte no etiquetada.

En este sentido, la principal motivación para esta tesis es la necesidad de modelos confiables para las tareas de diagnóstico y pronóstico que puedan aprender de forma semi-supervisada, aprovechando tanto la información etiquetada como la no etiquetada.

El objetivo principal de esta tesis es evaluar las mejoras logradas en las tareas de pronóstico y diagnóstico con el modelo propuesto que utiliza una estrategia de entrenamiento acoplada para optimizar simultáneamente un VAE temporal (un modelo no supervisado) y un predictor de red neuronal normal (un modelo totalmente supervisado) para situaciones en las que la cantidad de información etiquetada es escasa.

La metodología utilizada en este trabajo consta principalmente de cuatro pasos. Primero, se realiza una revisión de literatura sobre enfoques semi-supervisados y modelos generativos profundos. En segundo lugar, se propone y explica una versión modificada del Auto-Encoder Variacional que utiliza flujos de normalización y permite el procesamiento y la generación de datos secuenciales. Tercero, se desarrolla el proceso de entrenamiento acoplado. Finalmente, el modelo propuesto con la nueva estrategia de entrenamiento se aplica en dos estudios de caso para obtener resultados y se discuten sus ventajas y desventajas con respecto a los enfoques tradicionales.

Las principales conclusiones extraídas desde los resultados expuestos es que, si bien se observa una mejora del modelo propuesto con respecto a una estrategia de entrenamiento desacoplada, otros experimentos demuestran que solo produce predicciones similares en comparación con una red neuronal recurrente tradicional entrenada con solo datos etiquetados. La investigación futura podría centrarse en probar modelos generativos profundos más avanzados con el mismo enfoque de entrenamiento o probar conjuntos de datos más complejos donde la generalización sea más difícil que los presentadas en esta tesis.

I want to turn the whole thing upside down
I'll find the things they say just can't be found
I'll share this love I find with everyone
We'll sing and dance to Mother Nature's songs
I don't want this feeling to go away.

Acknowledgments

Primero que todo, me gustaría darle las gracias a mi comisión de magíster, profesores Enrique López Droguett, Viviana Meruane Naranjo y Juan Tapia Farías. Su ayuda y disposición fue invaluable para el desarrollo de este trabajo.

Segundo, me gustaría agradecerle muchísimo a mi familia. A mi madre y padre que hicieron todo lo posible para que yo pudiera estudiar comodamente y sin complicaciones, a mis hermanos que me acompañaron en el proceso y a todos los demas que de una forma u otra hicieron posible esto. También, quiero darle las gracias a mi pareja, Camila, por sus infinitos consejos y la motivación para seguir adelante. Los amo mucho a todos.

Tercero, muchas gracias a todos mis amigos y compañeros cuyo acompañamiento, buenos consejos y el agradable tiempo juntos no solo ayudaron a completar esta tesis, si no que también hicieron que el tiempo que demoró fuera muy disfrutable.

Finalmente, quiero extender mis agradecimientos a Conicyt por financiar esta investigación y mi educación de post-grado. Espero firmemente poder ser un aporte para el país y de esa manera devolver esta oportunidad que se me otorgo.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data, in one way or another, has always been used in the prognostics and health management (PHM) field to assess the operation and maintainability of physical assets. Between the end of the 1990s and beginning of the 2000s, advanced statistical methods and machine learning algorithms started to be used for this extent due to their properties to automatically extracted important features from operational records. Tools such as support vector machines (SVM) [3], decision trees [4] and particle filters [5] are still used today with many success cases in the PHM research community and industry. Nevertheless, due to the increasing availability of sensing technology in mainstream industry, one of the main challenges that reliability engineers face nowadays is how to use the immense amounts of highly dimensional data that is being generated every day from machinery and systems. Traditional machine learning methods such as the described above fell short to produce acceptable results when the input data has a high dimensionality representation, phenomenon that is also known as the *curse of dimensionality* [6]. One possible approach to tackle this problem is the use of deep learning models, which have resurfaced in popularity during the last decade mostly due to the decrease in price of computational hardware that allows for parallel computing (such as GPUs and TPUs). In fact, many recent successful examples of the application of deep learning techniques to the PHM field exists in the literature, such as recurrent neural networks for predicting the remaining useful life of lithium-ion batteries [7] or convolutional neural networks for fault diagnosis in a wind turbine gearbox [8], to name a few.

Nevertheless, the use of traditional deep learning techniques (such as the ones mentioned above) requires higher quantities of labeled data to train the immense number of independent parameters that those model contains. In addition to that, companies and industries are very skeptical to share their data, fearing the reveal of secrets or giving an advantage to the competence. Therefore, nowadays in the PHM field novel algorithms have to be validated using mostly datasets that are released to the public by universities or conferences. This is the case of the CMAPSS TurboFan dataset for remaining useful life prognostics [9], the Case Western Reserve ball bearing dataset for failure diagnostics [10] or the Bearing vibration data collected under time-varying rotational speed conditions dataset from the University of Ottawa [11], to name a few. One of the main problems with these datasets is that they are produced in a controlled environment or are directly produced by running computational simulations. This generate datasets with an abnormal and unrealistic number of labels, which

do not reflect the reality of industry at all. In industry labels are very hard and expensive to produce, and thus real world datasets contain low levels of them. Because of this, this work put an emphasis in researching models that can learn effectively in situations where the amount of labeled data is low, but there is an abundance of unlabeled data. Learning models under these conditions is also known as Semi-Supervised learning.

One possible approach to Semi-Supervised learning is to first transform the input data into an alternative or latent representation in which the most relevant features are identified and enhanced to allow the models to learn from fewer, but better, examples. These alternative representations can be generated using many techniques, but in recent research, Deep Generative models (DGM) have arisen in popularity for this extend due to them being a combination between traditional deep learning models and probabilistic modeling, juxtaposition that gives DGMs flexibility and a probabilistic interpretation of the results. Within deep generative models, Variational Auto-Encoders (VAEs) [12] are probably the most notorious and popular one. Variational Auto-Encoders combine variational inference with the use of neural networks as functions approximators to approximate the posterior probability distribution of the latent representation, effectively transforming the data into a space that can benefit the training of other models.

The traditional approach for semi-supervised learning with VAEs consist on mainly three steps. Firstly, the VAE is trained in an unsupervised manner using the entire dataset until a termination condition (such as the convergence of the cost function) is reached. Then, once the VAE is trained, it is used as an unsupervised feature extractor to transform the labeled portion of the dataset to the latent representation. Finally, this latent representation is used in conjunction with the labels as a new dataset to train a second model, which is often called the *predictor*, to solve a classification or regression problem, for example. This approach have already been used within the PHM research community to perform a variety of tasks. For example, San Martín *et. al.* uses vanilla VAEs and artificial neural networks to perform the diagnosis of health states in ball bearings [13]. Another example is given by Yoon *et. al.*, who uses a modified VAE with recurrent neural networks to perform the prognosis of remaining useful life for turbofans [14].

Nevertheless, the aforementioned applications present three main shortcomings. First, although multiple modified versions of the vanilla VAE model has been proposed to deal with input data in the form of time-series for general contexts [15, 16, 17], very little attention has been given to those approaches within the PHM research community, where the temporality has been either ignored, oversimplified or eliminated from the analysis using strategies to convert sequential data to non-sequential data (such as extracting temporal features) to use traditional VAE models. A second shortcoming is that although a wide variety of improvements has been made to the VAE model since it was first proposed in 2013, the PHM community keeps using what is almost a vanilla VAE model, without experimenting with any of the variations to see the effects that they could have on the diagnosis and prognosis results. Finally, and most importantly, the traditional approach of using VAEs for semi-supervised learning tasks rely on a decoupled training strategy, where, as explained before, the VAE model is trained separately from the predictor model. This produces a series of disadvantages, but maybe the most critical one is that the information contained in the few available labels never reach the VAE model. This represent a serious waste of possible improvement

opportunities for the traditional approach, since it is the VAE who generate the transformed dataset that the predictor model uses as input, and therefore it could benefit greatly from guiding the optimization with more assertive information.

To tackle the three aforementioned shortcomings, this thesis proposes a model that trains the VAE and the predictor model in a coupled manner, fusing them together into what is a mixed model with two different objective functions. In this manner, the labeled information, contained in the gradients of the predictor's loss function is back-propagated to the VAE model. Also, the VAE itself is subject of two modifications in the proposed model. First, its latent representation is enhanced using a technique known as *normalizing flows* [18, 19], which has reported severe improvements in VAEs when they are use for other purposes. Secondly, the VAE model is redesign to allow the direct use of sequential data as its input, generating what will be called during this thesis as a *temporal Variational Auto-Encoder*.

The proposed model is studied under two different datasets. One is the CMAPSS dataset [9], a prognosis example in which the objective is to predict the remaining useful life of simulated turbofans. The second is a diagnosis example in which the objective is to classify the health states of a real pumping system used in a offshore oil extraction platform. To assess the advantages and disadvantages of the proposed model, it is compared against two baselines designed to test particular aspects of the proposed approach.

In the following, the motivation, general objective and specific objectives of the thesis are declared, as well as the resources used for this work.

1.1 Motivation

The motivation for this thesis can be divided in two aspects. First, because the labeling data process is very expensive for most PHM applications, real world datasets often contain a severe under-representation of labeled exampled. For this reason, it is necessary to focus the research efforts towards the development of models and algorithms that can learn meaningful features and characteristics of the data in an unsupervised manner as well as from those few labeled examples that can be afforded. Secondly, since most of the operational data generated in nowadays industries contain some degree of temporality within it, the models developed to handle PHM data should have a mathematical explanation of the assumptions that are being made when the engineer chooses different kind of sequential models. This thesis proposed a model that attempts to contribute towards these two problematics.

1.2 General Objective

Propose and assess a semi-supervised learning approach based on the use of temporal Variational Auto-Encoders trained in conjunction with a neural network predictor to perform tasks such as the diagnosis of failure severities and the prognosis of remaining useful life in cases where the amount of labels is very scarce.

1.3 Specific Objectives

1. Propose a modification for the traditional Variational Auto-Encoder (VAE) models that can process sequential input data and generate a sequential latent space.
2. Propose a framework to incorporate non-sequential normalizing flows in sequential latent spaces.
3. Propose a coupled training procedure to, at the same time, train the modified VAE model and the neural network predictor in a way that labeled information can be used to optimize the encoder's network.
4. Design and execute experiments on the proposed model and suitable baselines in order to assess the advantages and disadvantages of the proposed approach to semi-supervised learning.

1.4 Resources Available for this Thesis

To run the proposed experiments in this thesis, the following hardware resources were kindly facilitated by the SRMI Lab of the University of Chile.

- Desktop computer with Ubuntu 16.04, i7-7700k processor, 32 GB of DDR4 RAM and a GPU NVIDIA Titan X.

As of software resources, everything was developed in open software libraries, which include:

- Python 3.6 programming language.
- TensorFlow 2.0.0 [20] as the primary Deep Learning library.
- Numpy [21], SkLearn[21], Matplotlib [22], Pandas [23] libraries were used for cleaning, visualizing and managing the data.

In addition, financial support during the full duration of this thesis was kindly provided by CONICYT [1] though a scholarship.

[1] Comisión Nacional de Investigación Científica y Tecnológica de Chile

Chapter 2

Methodology

In order to successfully explore and test the capabilities of the tVAEs in a semi-supervised scenario for PHM data, and therefore fulfill the objectives of this thesis, the following methodological steps were developed.

1. **Data Acquisition:** The models that will be designed and developed in this work are trained using automatic differentiation, also known as back-propagation [24], learning the neural networks parameters from data that represent some phenomenon of interest. In this case, this phenomenon is the degradation of an asset or system that causes failures in it's operation. Therefore, the first step in this thesis is the acquisition of operational datasets of mechanical element or systems, to train and validate the proposed models. The acquired datasets are briefly described below:
    - FD004 Turbofan Dataset: The objective of this dataset is to perform prognostic of remaining useful life of turbofans using operational data generated by a simulation algorithm. This dataset is part of the CMAPSS dataset [9]. In particular, the data correspondent to the FD004 subset has the peculiarity of containing the higher number of failure modes and operational conditions of all the subsets, thus resulting in a much more difficult prognostic problem.
    - Oil Pumps: This dataset was kindly provided by a South-American oil company and contain data regarding the health states of pumps used in the offshore extraction of oil. This dataset was chosen because it is a real world dataset, meaning that it was not produced in a laboratory but in a real industrial environment.
2. **Literature Review:** This step includes reading the state of the art literature available at the moment of developing this thesis as well as reading any relevant previous work that relates directly or indirectly to the research. In the following list there is a summary of such topics.
    - The use of Variational Auto-Encoders in cases where the input data and the latent space has a sequential structure.
    - Deep generative models, specially VAEs, used in semi-supervised learning of data.
    - Normalizing flows and it's application to deep generative models such as VAEs.
3. **Development of the temporal VAEs (tVAEs):** This step consist in both the theoretical adaptation of the VAE model to work with sequential data, which in the

PHM context are time series, and the development of such adaptation in code.

4. **Development of the semi-supervised learning framework:** This step includes the development of the proposed training process that uses tVAEs in conjunction with traditional neural network predictors to do semi-supervised learning.
5. **Integration of normalizing flows to the tVAEs models:** Since in current literature normalizing flows are proposed as a possible improvement of traditional VAEs, this thesis proposes a very simple framework that enables the integration of normalizing flows to VAEs working with sequential data, i.e. tVAEs. This include the integration from a model point of view (where should the normalizing flows be included) as well from a mathematical point of view (how does the loss function of the model changes when normalizing flows are included).
6. **Design and execution of the experiments:** To fulfill the objectives of this thesis and extract conclusions, carefully designed experiments are needed. This step include the definition of the parameters that will be varied across experiments and the models that will be tested.
7. **Analysis of the results and concluding remarks:** Once the experiments are executed, results will be extracted to reach conclusions about the advantages, disadvantages and usefulness of the proposed approach. Also, is possible that interesting conclusions will be drawn from the results that in the beginning of this thesis were not contemplated. Such conclusions will also be analyzed in this step.

Chapter 3

Theoretical Background

The following chapter details the necessary background needed to understand and achieve the objectives of this work. First, a quick review on Machine Learning is given. After that, Deep Learning and the principal neural networks architecture are explained, along with the loss functions used in this thesis to optimize the algorithms. Then, variational inference and variational auto-encoders are presented in the non-temporal and temporal setting. Lastly, a technique called normalizing flows commonly used to improve variational auto-encoders latent space is introduced to the reader.

3.1 Machine Learning

Machine Learning can be defined as a set of methods that can automatically detect patterns and structures within data, and then use those learned characteristics to perform predictions or decision making tasks [1]. In general, machine learning is a subset of the artificial intelligence (AI) field, that draw heavily from probability theory, statistics techniques and computer sciences to build general models that can be trained with data, so that explicit programing is not needed for solving complex problems. The types of problems that can be solved with machine learning can be divided in, roughly, four categories: classification, regression, clustering and anomaly detection [25]. In what follows, a brief review of the two types of problems used in this thesis, classification and regression, is presented.

3.1.1 Classification and Regression Problems

In classification problems, the objective is to train a model to map input data x into one or more classes or categories $y_{c=1}^{C}$ where $c \in \{1, 2, ..., C\}$ and C is the number of available classes. This thesis focuses on solving problems in which each data point x belongs to just one class of three or more possible classes. This type of problems are often known as *multi-class classification*.

On the other hand, in regression problems, the objective is to train a model to map input data x into one or more continuous values y. In other words, the trained model is a function f capable of doing the transformation $y = f(x)$. Since the prediction space containing y

is not constrain to a list of integer values in regression problems, these tend to be much more complex and hard to solve than classification problems. This thesis focuses on solving problems in which the prediction y only consist on one continuous value.

In the PHM context, classification problems are presented naturally when engineers are diagnosing the health state of a machine or trying to identify the location of a fault, for example. Regression problems also present themselves in PHM, more frequently in the prediction of the RUL, or *remaining useful life* of a certain machine or system.

Machine Learning problems can also be divided based on the availability of data in two distinct categories: supervised learning and unsupervised learning [1]. These two classes of machine learning problems are presented below.

3.1.2 Supervised and Unsupervised Learning

In supervised learning, input data x as well as labels y are widely available for the training process. This means that the supervised model is designed to focus on the mapping between x and y, focusing in identifying the characteristics present in x that can relate that particular example with the corresponding label y. Typically, regression and classification problems are also supervised problems.

Unsupervised learning problems, on the other hand, are characterized by situations in which only data x is available, but labels y are not. Therefore, unsupervised algorithms focus on finding relationships within the input data, to identify structures of similarity or to transform the data into a better space. Clustering algorithms, such as K-Means [26] or DBScan [27], are examples of unsupervised models in which the objective is to separate the dataset into disjunct classes based on a particular similarity metric. Another example are dimensionality reduction algorithms, such as PCA [1], where the objective is to transform the data x into a better, more concise representation z.

Below, a third category that represent a middle ground between supervised and unsupervised problems is presented.

3.1.3 Semi-Supervised Learning

While fully supervised learning problems deals with data that is completely labeled and unsupervised learning deals with the exact opposite problem, semi-supervised learning present when labels are available, but scarcely. This means that the input data x can be divided into a labeled portion $(x_{\mathrm{i}}, y_{\mathrm{i}})_{\mathrm{i}=1}^{L}$ and an unlabeled portion $(x_{\mathrm{i}})_{\mathrm{i}=1}^{U}$, where U and L represent the number of unlabeled and labeled data points, respectivaly. Generally, U is considerably greater than L. In the PHM context, this type of problems have special importance due to the higher costs associated with labeled data and the lower amount of information that unsupervised algorithms contribute. Therefore, an ideal situation is to have a good performing algorithm that can learn to make diagnosis and prognosis using a very small portion of labeled examples. This thesis focuses on developing models that can solve this type of problems, using a combination of supervised and unsupervised techniques.

In what follows, deep learning is presented as a sub-field of machine learning that deals with

high-dimensional data to build very efficient function approximation models called neural networks.

3.2 Deep Learning

Deep learning is a subfield of machine learning that focus on the use of automatic differentiation to train structures named neural networks. Neural networks are function approximation models used to map a certain input with a desired output. These types of models are composed by layers or cleverly ordered parameters trained using a technique based on gradient descent called automatic differentiation [24]. Neural networks can be used directly as function approximators to perform simpler tasks such as classification and regression, or be included as modules in more complex models to accomplished a wide variety of tasks, such as data reconstruction, speech generation or sentiment analysis.

This section review two types of neural network architecture that are used in this thesis: traditional neural networks and recurrent neural networks.

3.2.1 Traditional Neural Networks

Traditional neural networks (NNs) [28] are designed to map a vector of input data, $\vec{x}$ to a vector of predicted values $\vec{y}$. To accomplish this, NNs apply a linear operation followed by a nonlinear function $h(\cdot)$, presented in equation 3.1.

$$\vec{y} = h(W\vec{x} + \vec{b}) \tag{3.1}$$

Where matrix W and vector $\vec{b}$ are parameters known as the *weights* and *biases* respectively, and are learned from the data. Traditional neural networks can be constructed using a deep architecture, in which equation 3.1 is applied two times or more, where each application has its own weights and biases and is commonly known as layer. For illustration purposes, in equation 3.2 a three layers NN is presented.

$$\vec{y} = h_3(W_3(h_2(W_2h_1(W_1\vec{x} + \vec{b_2}) + \vec{b_2}) + \vec{b_2})) \tag{3.2}$$

3.2.2 Recurrent Neural Networks

One disadvantage of traditional neural networks is their inability to process sequential data of the form $\mathbf{x} = \vec{x}_1, ..., \vec{x}_T$. This type of setting surfaces often when modelling problems in which main phenomenon depends on time, such as a degradation process or the evolution of some operational magnitude. Recurrent neural networks (RNNs) [1] were developed as a model capable of dealing with this type of input data. In RNNs, the input sequence is unfolded and each resulting vector $x\vec{x}_t$ is processed applying the following equation:

$$\vec{y}_t = h(W_x\vec{x}_t + W_y\vec{y}_{t-1} + \vec{b}) \tag{3.3}$$

Where now the recurrent layer has two weight matrices, W_x and W_y, one for the the input vector of that time $\vec{x}_t$ and another for the output of the previous time, $\vec{y}_t$. For the first time step, only the operations related to the input $\vec{x}_0$ are applied.

As it can be seen from equation 3.3, RNNs uses the immediate past output of the sequence to compute the future outcome. Because of this, they are capable of finding patterns not only within the features of $\mathbf{x}$, but across the time dimension as well. A diagram of a RNN is presented in figure 3.1. The sequence of outputs vectors $\vec{y}_{1:T}$ is also known as the states of the RNNs. Nevertheless, in most cases only the last state, $\vec{y}_T$, is used as the output of the RNN.

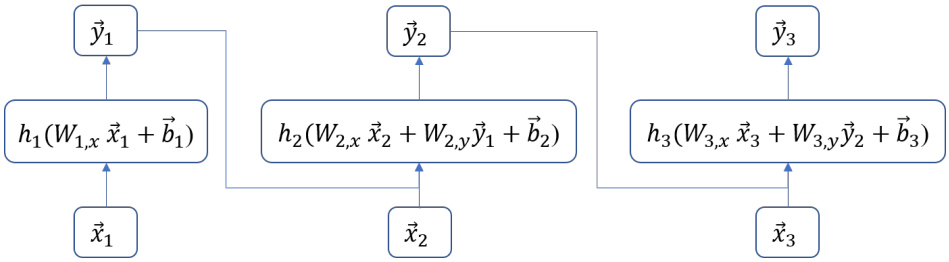


Figure 3.1: Diagram of an RNN applied to sequential input data $\mathbf{x}$ with $T = 3$. Note how for the first time step, only $\vec{x}_1$ is used since $\vec{y}_0$ is not defined yet.

Nevertheless, one drawback of RNNs is that they only have one step backward memory. This can be seen from equation 3.3, where for the computation of $\vec{y}_t$, only $\vec{y}_{t-1}$ is directly used. This result in a neural network model that has a very short memory and therefore can only detects patterns across time of limited duration. To overcome this difficulty, a modification of recurrent neural networks called Long Short Term Memory Networks (LSTM) [29] were proposed. In LSTMs, the block of computation for each time step is much more complex, consisting in two computational flows, one for long term memory and another for short term memory, denoted as the *cell* and *state* respectively. In this nomenclature, they are represented by the variables $\vec{c}_{1:T}$ and $\vec{h}_{1:T}$.

As it can be seen in figure 3.2, LSTMs typically consist of three gates: forget, input and output gates. Each gate is responsible for allowing or prohibiting the flow of information, resulting in a neural network that can effectively *learn to remember*. The forget gate ($\vec{f}_t$) controls which parts of the long term state ($\vec{c}_t$) should be erased. The input gate ($\vec{\mathrm{i}}_t$) controls which parts of the input data $\vec{x}_t$ and the previous hidden state $\mathbf{h}_t$ should be added to the long term state. Finally, the output gate ($\vec{o}_t$) controls which parts of the long term state should be read and output at this time step, both to the next hidden state $\vec{h}_t$ and to $\vec{y}_t$. The specific operations that these gates apply are presented in equation 3.4.

$$
\begin{aligned}
\vec{\mathrm{i}}_t &= \sigma(W_{x,\mathrm{i}}\vec{x}_t + W_{h,\mathrm{i}}\vec{h}_{t-1} + \vec{b}_{\mathrm{i}}) \\
\vec{f}_t &= \sigma(W_{x,f}\vec{x}_t + W_{h,f}\vec{h}_{t-1} + \vec{b}_f) \\
\vec{o}_t &= \sigma(W_{x,o}\vec{x}_t + W_{h,o}\vec{h}_{t-1} + \vec{b}_o) \\
\vec{g}_t &= tanh(W_{x,g}\vec{x}_t + W_{h,g}\vec{h}_{t-1} + \vec{b}_g) \\
\vec{c}_t &= \vec{f}_t \otimes \vec{c}_{t-1} + \vec{\mathrm{i}}_t \otimes \vec{g}_t \\
\vec{y}_t &= \vec{h}_t = \vec{o}_t \otimes tanh(\vec{c}_t)
\end{aligned}
\tag{3.4}
$$

The computational block of the LSTM is presented in figure 3.2.

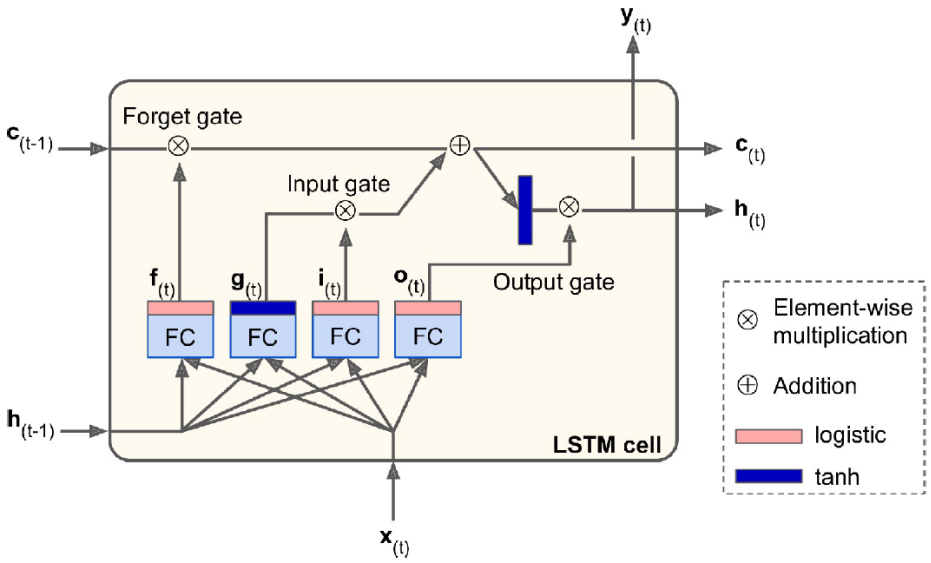


Figure 3.2: Diagram of an LSTM computation block. In this context, **c** and **h** represent the long term and short term memory flows respectively. The acronym FC refers *fully connected*, which is another name for traditional neural networks. For further information regarding the computation block, the interested reader can review [1]. Source: Aurelien Geron, Hands on Machine Learning, 2019 [1].

LSTMs and RNNs are designed to detects patterns in a temporally forward manner. This means that information is analyzed following the natural flow of time within the sequence (from past to future). Nevertheless, there are some cases in which one may desire to search patterns in both directions (past to future and also future to past). For this, a modification of the LSTM called Bidirectional LSTM (BiLSTM) was proposed. BiLSTMs consist on two normal LSTM applied each in one direction. The outputs $\vec{y}_t$ for each time step are concatenated to form the combined output. Figure 3.3 show a general diagram of two LSTM being applied over the same sequence to form a BiLSTM.

In the next section, a brief review on the training procedure of neural networks is presented.

3.2.3 Training of Neural Networks

As it can be seen from the previous sections, each type of neural network can be reduced to a function f with parameters θ (in which all the weights and biases of the varying structures

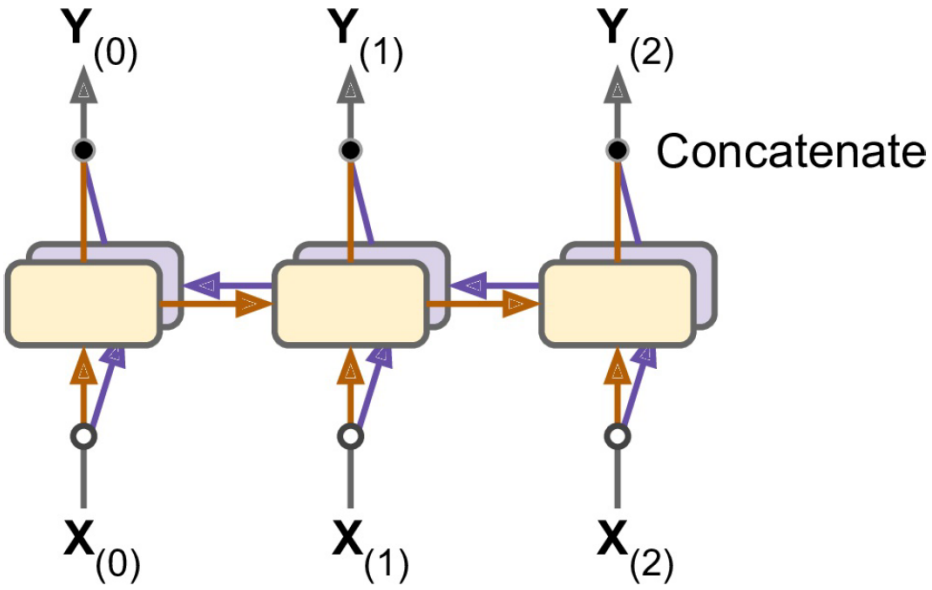


Figure 3.3: Diagram of a BiLSTM applied over sequential data. Note how each LSTM have a different computation direction. Source: Aurelien Geron, Hands on Machine Learning, 2019 [1].

are contained) that do the following mapping $\hat{y} = f(x)$, where x is a vector or a sequence of vectors depending on the type of neural network and $\hat{y}$ are the predicted values. Since neural networks are a data driven type of algorithm, its parameters are learned using a training dataset. This dataset typically consist on pairs of data points x and labels y. The parameters θ are optimized minimizing a loss function which commonly compares the predictions of the neural net, $\hat{y}$, with the true values, or labels y, for each data point.

This minimization process is done first computing the gradients of each parameter contained in θ with respect to the loss function L, $\frac{\mathrm{d}L}{\mathrm{d}\theta_\mathrm{i}}$ with back propagation and then using gradient descent to update each weight θ_i. This process can be done one example at the time, but convergence is faster and more stable if the gradients for each data point within an aleatorily sampled subset of the dataset $(x_\mathrm{i}, y_\mathrm{i})_{\mathrm{i}=1}^{N}$ are averaged, and then applied the corresponding gradient descent to the weights and biases of the neural network. The number of examples within that aleatorily sampled subset is commonly known as the batch size. When every pair of the original dataset has been used to compute gradients at least once, it is said that one epoch of training has passed. The full training procedure usually last for a number of epochs that ranged between 10 and 10000.

For classification problems, the final layer of the neural network is structured to output a vector $\hat{\vec{y}}$ of C components, where C is the number of possible classes. The final activation function in this case is the softmax function, presented in equation 3.5. This function applied to the last layer generate an output vector in which the components sum to i. Because of this, each component of $\hat{\vec{y}}$ can be interpret as the probability that the data point belongs to the corresponding class.

$$\sigma(\hat{\vec{y}})_j = \frac{\mathrm{e}^{\hat{y}_j}}{\sum_{c=1}^{C} \mathrm{e}^{\hat{y}_c}}, \quad for\ j = 1, ..., C \tag{3.5}$$

Usually, cross-entropy is used as the loss function for classification problems. This loss function compare $\hat{\vec{y}}$ and $\vec{y}$, which is the true class formatted as a one hot-encode vector, maximizing the probabilities of the correct classes. Cross-entropy is show in equation 3.6 applied to a particular example. The total loss over N examples is just the average of the particular losses.

$$CE = -\sum_{c=1}^{C} y_c \log(\sigma(\hat{y}_c)) \tag{3.6}$$

Where σ is the previously defined softmax function.

For regression problems with just one value to be predicted, such as the ones presented in this thesis, the output of the last layer of the neural network is a vector of only one component, $\hat{y}$. Frequently, the activation function of the last layer is the identity function. The loss function used in this case is some metric of error between real valued vectors, such as the mean squared error. The form of the mean squared error loss function is presented in equation 3.7 applied to a dataset with N examples.

$$MSE = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}(y_{\mathrm{i}} - \hat{y}_{\mathrm{i}})^2 \tag{3.7}$$

The following section explains a technique for approximating probability distributions called *variational inference*, uses as the theoretical basis for the main model of this thesis: the *Variational Auto-Encoder.*

3.3 Variational Inference

One of the main challenges in probabilistic inference in general is the approximation of difficult to compute probabilities densities.In Bayesian statistics, this problem arises as a fundamental one, since the posterior plays an important role in the inference of the quantities of interest. Currently, a powerful and flexible approach to approximate these probabilities densities is variational inference (VI) [30].

Variational inference treats the problem of finding an approximate distribution as an optimization problem. To see this in a clearer way, it is necessary to first set the problem that variational inference try to solve. Let's consider a set of observations, $x = \{x1, \ldots, x_N\}$, which specific form (whether they are vectors or matrices) is not important at this point. This set of observations is assumed to be controlled by a set of latent variables $z = \{z_1, \ldots, z_N\}$. The joint distribution that make explicit the relationship between x and z is shown in equation 3.8.

$$p(x, z) = p(z)p(x|z) \tag{3.8}$$

In a Bayesian framework, latent variables are sampled from a prior distribution $p(z)$, which is common to be defined as a known probability distribution, and then related to the observations through the likelihood $p(x|z)$, which is the model that is assumed to explain the data. The main objective is to make inference over the posterior distribution $p(z|x)$. This serves two main purposes. Firstly, latent variables inferred from x can be used as a reinterpretation of the original dataset for downstream tasks. Secondly, knowledge of the posterior distribution is imperative for generative models where the main objective is to generate new, unseen data that is alike to the existing data.

For complex models, computing $p(z|x)$ can be very difficult, in some cases even impossible. In situations like these, one possibility is to approximate the posterior $p(z|x)$ by some distribution $q_\phi(z|x)$ that belongs to a family Q of probability distributions that can be parametrized by some set of parameters ϕ. If Q is a complex enough family that can contain a good approximation to the posterior, but not so complex that the search for $q_phi(z|x)$ in that family is infeasible, then variational inference may solve the problem of finding a good approximation for the posterior. Based on this approach, the objective is to solve the optimization problem of finding the closest approximate distribution $q_\phi(z|x)$ to the true posterior $p(z|x)$ when $q_\phi(z|x)$ is restricted to belong to Q. The objective function for this optimization is a notion of *distance* between probability distributions called Kullback-Leibler divergence, presented in equation 3.9.

$$KL(q_\phi(z|x)||p(z|x)) = \int_{\Omega_z} q_\phi(z|x) \log \frac{q_\phi(z|x)}{p(z|x)} \mathrm{d}x \tag{3.9}$$

If this definition is expanded using the definition for the expectation and the properties of the natural logarithm, the following expression is obtain:

$$KL(q_\phi(z|x)||p(z|x)) = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log q_\phi(z|x) - \log p(z|x)] \tag{3.10}$$

The best $q_\phi(z|x)$ within Q is found minimizing this notion of distance, i.e., minimizing the Kullback-Leibler divergence presented in equation 3.10. Nevertheless, the main problem that variational inference faces is that the objective function described in equation 3.10 is not estimable because it needs the computation of the logarithm of $p(z|x)$ which in turn is the function to be approximated, generating a circular problem. To find a computable form for the Kullback-Leibler divergence in order to solve the minimization problem, the Bayes theorem for $p(z|x)$ is used on equation 3.10 to obtain:

$$KL(q_\phi(z|x)||p(z|x)) = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log q_\phi(z|x) - \log \frac{p(x|z)p(z)}{p(x)}] \tag{3.11}$$

Now, using the logarithm property, equation 3.11 can be rewritten as:

$$KL(q_\phi(z|x)||p(z|x)) = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log q_\phi(z|x) - \log p(x|z) - \log p(z) + \log p(x)] \tag{3.12}$$

The last term on the right hand side can be extracted from the expectation, since it does not depend in the latent variable z, resulting in:

$$KL(q_\phi(z|x)||p(z|x)) = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log q_\phi(z|x) - \log p(x|z) - \log p(z)] + \log p(x) \tag{3.13}$$

Rearranging terms, the following is obtained:

$$\log p(x) - KL(q_\phi(z|x)||p(z|x)) = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log p(x|z) - \log q_\phi(z|x) + \log p(z)] \tag{3.14}$$

And using again the linearity of the expectation, the right hand side of equation 3.14 can be separated to obtain:

$$\log p(x) - KL(q_\phi(z|x)||p(z|x)) = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log p(x|z)] - \mathbb{E}_{q_\phi(z|x)\sim Q}[\log q_\phi(z|x) - \log p(z)] \tag{3.15}$$

Finally, from equation 3.10 a new Kullback-Leibler divergence is recognized between the approximate posterior distribution $q_\phi(z|x)$ and the prior distribution $p(z)$, to obtain:

$$\log p(x) - KL(q_\phi(z|x)||p(z|x)) = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log p(x|z)] - KL(q_\phi(z|x)||p(z)) \tag{3.16}$$

Now, our main objective was to minimize the *distance* between the approximate posterior $q_\phi(z|x)$ and the real posterior $p(z|x)$, represented by the Kullback-Leibler divergence $KL(q_\phi(z|x)||p(z|x))$. That expression is contained in the left hand side of equation 3.16. In order to minimize the Kullback-Leibler divergence, only the approximate posterior $q_\phi(z|x)$ can vary, which will not affect the log-probability of the model, $\log p(x)$. Therefore, if the right hand side of equation 3.16 is maximized over $q_\phi(z|x)$, that inevitable will result in minimization of $KL(q(z|x)||p(z|x))$ since $\log p(x)$ will not change. Because of the importance of the right hand side, in the variational inference field it is often given a special name: the Evidence Lower Bound, *ELBO* for short. The name derives from it being always a lower bound for $\log p(x)$ (the evidence) since the Kullback-Leibler divergence is always a non-negative quantity [30]. The form for the ELBO is presented in equation 3.17 for formality:

$$ELBO = \mathbb{E}_{q_\phi(z|x)\sim Q}[\log p(x|z)] - KL(q_\phi(z|x)||p(z)) \tag{3.17}$$

In summary, maximizing the ELBO expression over $q_\phi(z|x)$ is equivalent to minimizing $KL(q_\phi(z|x)||p(z|x))$, thus converting the original variational inference problem, which was intractable, into a feasible one [30].

Therefore, the final optimization problem to find the optimum posterior approximation $q\phi(z|x)$ is presented in equation 3.18.

$$q^*_\phi(z|x) = \underset{q_\phi(z|x)\sim Q}{\arg\max}\, \mathbb{E}_{q_\phi(z|x)\sim Q}[\log p(x|z)] - KL(q_\phi(z|x)||p(z)) \tag{3.18}$$

In what follows, it will be shown how VI can be applied to a machine learning model called Variational Auto-Encoder in order to either generate new data or generate a latent space representation of a certain database.

3.4 Variational Auto-Encoders

Variational Auto-Encoders (VAEs), originally proposed by [12] are generative models that combine neural networks, variational inference and unsupervised learning to address the problem of finding an approximation to a posterior probability distribution $p(z|x)$. They are called "auto-encoders" because they are composed of two main structures: the first one encodes the input data into a stochastic latent representation, whereas the second one decodes this latent representation onto an approximation or reconstruction of the original data. Thus, both parts are called encoder and decoder, respectively, and they resemble the structure of the traditional auto-encoder model [1].

The main differences between VAEs and any other model that uses variational inference can be roughly summarize into two. Firstly, VAEs assumes that $p(z)$ and $p(x|z)$ also belong to a known family of parametric distributions, controlled by a set of parameters denoted by θ, therefore represented by $p_\theta(z)$ and $p_\theta(x|z)$. As shown in the previous section, this is also the case for the posterior approximation $q_\phi(z|x)$, controlled by its own set of parameters denoted by ϕ. Secondly, the set of parameters $\{\theta, \phi\}$ are treated as outputs of neural networks within the model. These two features, i.e., the assumption of known forms for the distributions of the model and the use of neural network to compute the parameters that controls such distributions, represent one of the main advantages of VAEs in comparison to other variational inference based approaches due to the added simplicity that parametric distributions give and the flexibility that neural networks contribute.

Next, a discussion about the parametric distributions forms for $p_\theta(z)$, $q_\phi(z|x)$ and $p_\theta(x|z)$ in the context of VAEs is presented. It will be assumed from here onwards that both x and z are vectors of features, i.e., $\vec{x} \in \mathbb{R}^D$ and $\vec{z} \in \mathbb{R}^K$. Therefore, this section presents the traditional VAE, which is not designed to process sequential input data. In the following sections and chapters a VAE model adapted to process such data will be presented as the temporal VAE (tVAE).

3.4.1 Prior over the latent variables $p(\vec{z})$

One of the main challenges in Bayesian models is defining the latent variables. Questions like the nature of the latent variables (should them be positive numbers, integer numbers) or the relationship between them are difficult to answer. VAEs take a simple approach to this problem assuming and incorporating this lack of knowledge in the form of a very basic and

uninformative choice for the prior distribution of the latent variables. An extremely common choice is to use an isotropic normal distribution for $p(\vec{z})$, dropping the dependence on θ. This election is presented in equation 3.19

$$p(\vec{z}) \sim N(\vec{z}|\vec{0}, I) \tag{3.19}$$

This effectively assumes no prior knowledge over the relationship between the variables nor the sign of them. The only apparent restriction is that they need to be real numbers centered around 0.

3.4.2 The decoder distribution $p_\theta(\vec{x}|\vec{z})$

The distribution $p_\theta(\vec{x}|\vec{z})$ represent the probability that certain data point $\vec{x}$ will be generated under the latent variable $\vec{z}$. The family of this distribution should be decided by the nature of the data itself. As shown in [12], if the original data is restricted to the interval $[0, 1]$, a good choice for $p_\theta(\vec{x}|\vec{z})$ is to use a Bernoulli distribution and consider the vector $\vec{\rho}$ that controls that distribution as the direct reconstruction of the original data. This choice of distribution is presented in equation 3.20.

$$p_\theta(\vec{x}|\vec{z}) \sim B\mathrm{e}(\vec{x}|f_\theta(\vec{z})) \tag{3.20}$$

Where f is a traditional neural network (see section 3.2.1) with weights and biases denoted by θ that takes as input a vector of latent features $\vec{z}$ and then return as output $\vec{\rho}$. For this thesis, a Bernoulli distribution is used to represent $p(\vec{x}|\vec{z})$ since after preprocessing the input data lie within the interval $[0, 1]$.

This probability distribution, $p(\vec{x}|\vec{z})$, and the traditional neural network that outputs its parameters are called the decoder of the VAE since they take the latent representation $\vec{z}$ and outputs a reconstruction, $\vec{\rho}$, of the original data $\vec{x}$. Thus, "reconstructing" or "decoding" the data from its latent representation.

3.4.3 The encoder distribution $q_\phi(\vec{z}|\vec{x})$

The main objective of a VAE is to find a good approximation for the true posterior. The most common decision regarding the family Q (from where the approximate distribution is search) corresponds to the family of diagonal normal distributions. This serves a double purpose. First, the election of this family allows each latent variable to have its own mean and variance, producing a relatively flexible model. Second, a diagonal normal distribution, in conjunction with the selection of the prior for $p(\vec{z})$, makes the Kullback-Leibler divergence in the objective function of equation 3.18 to have a close and easy to compute form [31]. Thus, the following is true for the VAE model:

$$q_\phi(\vec{z}|\vec{x}) \sim N(\vec{z}|\mu_\phi(\vec{x}), \sigma_\phi(\vec{x})I) \tag{3.21}$$

Here, μ_ϕ and $sigma_\phi$ are traditional neural networks that take as input the original data $\vec{x}$ as well as weights and biases denoted by ϕ and then outputs a vector of means and variances for the latent variables. Since the distribution $q_\phi(\vec{z}|\vec{x})$ is forced to be diagonal, that vector of variances is multiplied by the identity matrix to form the covariance matrix. The distribution $q_\phi(\vec{z}|\vec{x})$ and the neural networks that parametrize it are called the encoder of the VAE since its task is to make a codification of the original data into a latent representation $\vec{z}$.

Next, it is discussed how this model is optimized to solve equation 3.18 and find a good approximation to the true posterior. Since VAEs use the variational inference approach, the function ELBO can be used as the objective function (See equation 3.17). Nevertheless, it is needed to be more precise about the explicit form that the ELBO function takes given the decisions made about the distributions $p(\vec{z})$, $q_\phi(\vec{z}|\vec{x})$ and $p_\theta(\vec{x}|\vec{z})$.

3.4.4 Forward pass of the VAE model

The previous sections present the different elements of the VAE. This section describe how they are used to compute the forward pass of the model. This operation flow can be enumerated in the following steps, starting with a particular element of the data set $\vec{x}_\mathrm{i}$:

1. Using $\vec{x}_\mathrm{i}$ as input for the encoder neural network, compute the mean ($\vec{\mu}_\mathrm{i}$) and standard deviation ($\vec{\sigma}_\mathrm{i}$) of the posterior distribution for that point.
2. Sample $\vec{z}_\mathrm{i}$ from the decoder distribution using the parameters $\vec{\mu}_\mathrm{i}$ and $\vec{\sigma}_\mathrm{i}$.
3. Using $\vec{z}_\mathrm{i}$ as input for the decoder neural network, compute the parameter vector $\vec{\rho}_\mathrm{i}$ for the Bernoulli distribution of the decoder. In this thesis, as mentioned before, this vector will be directly used as a reconstruction of $\vec{x}_\mathrm{i}$ without further sampling.

The forward pass compute all the required variables for the computation of the ELBO function of the VAE, as show in the following section.

3.4.5 ELBO function for the VAE model

The ELBO function is made of two parts: the KL divergence between the approximate posterior $q_\phi(\vec{z}|\vec{x})$ and the prior $p(\vec{z})$; and the expectancy over the logarithm of $p_\theta(\vec{x}|\vec{z})$. In what follows, these two parts are tackled separately to show how they are computed based on a single data point $\vec{x}_\mathrm{i}$ and the variables resulting for the forward pass.

Kullback-Leibler Divergence $KL(q(\vec{z}_\mathrm{i}|\vec{x}_\mathrm{i})||p(\vec{z}_\mathrm{i}))$

Normally, the Kullback-Leibler divergence is estimated using Monte Carlo integration. Nevertheless, since $q_\phi(\vec{z}_\mathrm{i}|\vec{x}_\mathrm{i})$ and $p(\vec{z})$ are chosen to be normal distributions, the Kullback-Leibler divergence between them has closed form. As shown in [31], the exact close form between a diagonal normal and an isotropic normal distribution is:

$$KL(N(\vec{\mu}_\mathrm{i}, \vec{\sigma}_\mathrm{i}\mathbb{I})||N(\vec{0}, \mathbb{I})) = \frac{1}{2}(tr(\vec{\sigma}_\mathrm{i}\mathbb{I}) + \vec{\mu}_\mathrm{i}^T\vec{\mu}_\mathrm{i} - K - \log\det(\vec{\sigma}_\mathrm{i}\mathbb{I})) \tag{3.22}$$

Where K is the dimensionality of the diagonal distribution. Since in the VAE model

both distributions outputs vectors or parameters related to the latent variables, K is the dimensionality of the latent variables, i.e., the number of latent variables that controls the model under study.

Expectancy over the logarithm of $p_\theta(\vec{x}_\mathrm{i}|\vec{z}_\mathrm{i})$

As stated in [12], this expectancy can be computed with Monte-Carlo integration using the following formulae:

$$\mathbb{E}_{q_\phi(\vec{z}|\vec{x}_\mathrm{i})\sim Q}[\log p_\theta(\vec{x}_\mathrm{i}|\vec{z})] \approx \sum_{l=1}^{L}[\log p_\theta(\vec{x}_\mathrm{i}|\vec{z}_{\mathrm{i},l})] \tag{3.23}$$

Where $\vec{z}_{\mathrm{i},l}$ represent the sample l from the distribution given by the parameters computed using $\vec{x}_\mathrm{i}$. The equation 3.23 requires sampling L times the latent variables to obtain the approximate cost of one data point. In this thesis L is set to 10 samples, which represent a fine equilibrium between a light model and good approximates.

The exact form of $\log p_\theta(\vec{x}_\mathrm{i}|\vec{z}_{\mathrm{i},l})$ will depend upon the choice of distribution for the decoder. For the specific case of Bernoulli distributions, used in this thesis, $\log p_\theta(\vec{x}_\mathrm{i}|\vec{z}_{\mathrm{i},l})$ will take the following form:

$$\log p_\theta(\vec{x}_\mathrm{i}|\vec{z}_{\mathrm{i},l}) = \vec{x}_\mathrm{i} \log \vec{\rho}_\mathrm{i} + (1 - \vec{x}_\mathrm{i}) \log(1 - \vec{\rho}_\mathrm{i}) \tag{3.24}$$

Where $\vec{\rho}_\mathrm{i}$ is the reconstruction obtained from the encoder, and therefore, a direct function of $\vec{z}_\mathrm{i}$.

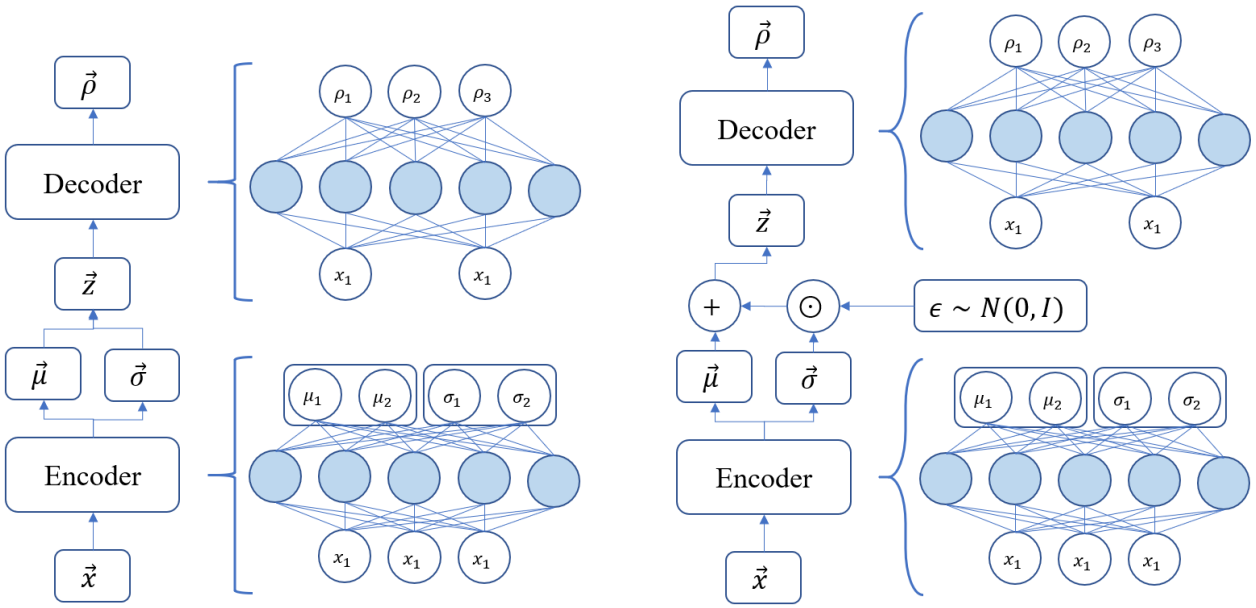
3.4.6 Putting the pieces together

Figure 3.4a show the VAE model comprised of the various elements discussed in the previous sections for the case when the nature of the data requires that the probability distribution of the decoder, $p(\vec{x}|\vec{z})$, belongs to the Bernoulli family. Note how the encoder and the decoder work together to first transform the original data into a stochastic latent representation and then to decode that latent representation into a reconstruction of the original data.

However, there is still one problem with this model. Since it uses neural networks to estimate the parameters of the distributions in the VAE model, the ideal situation would be to use back-propagation to optimize this model along with SGD. But back-propagation cannot work with stochastic units inside the neural network, since they are non-continuous operations and, therefore, the gradient is not defined at these points. Nevertheless, what back-propagation can do is handle stochastic inputs to the network, if they are sampled outside of it and, for the neural network, they work as any other deterministic input. To solve this problem, one can refer to a trick called "the reparameterization trick" as proposed by [12], which works using the following property:

$$N(\vec{\mu}, \Sigma) = \vec{\mu} + \Sigma \cdot \vec{\varepsilon} \tag{3.25}$$

Which states that a normal distribution with mean $\vec{\mu}$ and covariance matrix Σ is equivalent (i.e., has the same distribution) as $\vec{\mu} + \Sigma \cdot \vec{\varepsilon}$ when $\vec{\varepsilon}$ is a random variable following a standard normal distribution. Since in this VAE model $q_\phi(\vec{z}|\vec{x})$ is parametrized by an diagonal normal distribution, the matrix of covariances can be fully represented by its diagonal vector, in this case, $\vec{\sigma}$. Based on equation 3.25, the model can be optimized with back-propagation as illustrated in Figure 3.4b.



(a) Without reparameterization trick. **(b)** With reparameterization trick.

Figure 3.4: VAE model with a decoder containing a Bernoulli probability distribution, suited for binary data or real valued data restrain to the $[0, 1]$ interval.

Thus, once the Variational Auto-Encoder model has been trained and optimized, it can be used to generate new data by feeding values of $\vec{z}$ sampled from the prior distribution $p(\vec{z}) \sim N(\vec{0}, \mathbb{I})$ into the decoder, obtain the vector of parameters that controls $p_\theta(\vec{x}|\vec{z})$ and posteriorly, sample from it to obtain a synthetic, newly generated data point. More important in the context of using the VAE in downstream tasks, one can interpret and make use of the obtained latent space from this trained VAE model (the output of its encoder) as a set of features, i.e., a feature map, providing a high-level representation (abstraction) of the input data. Note also that is the user who defines the dimension of such latent space, thus being able to specify the desired level of compression or dimensionality reduction, if required.

Up until this point, only VAEs that use non-sequential input data in the form of vectors of features have been considerer. Because of this, the internal structure of the VAE uses traditional neural networks to approximate the controlling parameters for both the encoder and decoder distributions, as shown in figure 3.4. As discussed in the introduction, PHM data is often sequential, with a strong time-dependent behavior. In the following section is presented a brief overview of variational auto-encoders models adapted to work with sequential input data within general and PHM literature. The specific derivation of the models used in this thesis are presented in section 4.

3.4.7 Sequential Variational Auto-Encoders in Literature

In general deep learning literature, variational auto encoders modified to add compatibility with sequential input data have been developed mostly for generative purposes using non structured data such as videos or to forecast time-series. For example, Bayer and Osendorfer in 2014 [32] proposed a model called STORN (Stochastic Recurrent Networks), which is a combination of traditional RNNs and Variational Inference used for music generation and forecasting of motion walking data in humans. Chung *et al.* in 2015 [17] proposed a stack of T variational auto-encoders (one for each time-step) in which they share information via conditioning the latent variables for each t on past observations and latent variables to generate speech and handwriting sentences. Krishnan *et al.* in 2017 [15] proposed a mixture between Kalman filter and variational methods to include control capability to the VAE model in the form of an input vector $\vec{u}$ to forecast the reaction of patients to certain drugs considering their medical history. More recently, Lim *et al.* in 2019 proposed a more complex approach to variational modelling of time series with their *Recurrent Neural Filters* which for each time step construct a LSTM-like cell of computation that can accept input dynamics and error correction features in order to forecast household electric power consumption and market-share time-series. The proposed model in this thesis is heavily inspired by all these models.

Within the PHM context, VAEs have been adapted to time-series using more simpler techniques. A common approach is to eliminate the temporal dependency from the input data, in the preprocessing stage.This can be done by extracting general features from the time-series such as the RMS, peak to peak, crest factor, among others. San Martín *et al.* [13] applied this approach to process ball-bearing time series using traditional variational auto-encoders. Yoon *et al.* [14] use a different methodology in which VAEs encoders are adapted with RNNs in order to process input time-series, but keeping a non-sequential structure for the latent space. This results in a model that encodes the whole input sequence into a single vector of latent features, allowing for a normal ELBO to be used, but losing any posterior temporal information in the process. According to the revision of the State of the art, more complex approaches within PHM context does not exists to the best of the author's knowledge.

Therefore, a main contribution with this thesis is to present a concise and detailed explanation of the general framework used to adapt a traditional VAE to a temporal setting, taking into consideration the necessary ELBO modifications to allow for a temporal latent space to be generated. This is presented in section 4.

3.5 Normalizing Flows

This section presents a technique used to add flexibility to probability distributions called *normalizing flows* [33] and how it can be implemented using neural networks, in order to seemingly incorporated it to a VAE model. The final implementation of normalizing flows to tVAEs will be presented in section 4.4.

In variational inference, the choice of the approximate posterior is a core problem. Most applications choose to employ simple families for $q_\phi(\vec{z}|\vec{x})$ (such as a diagonal normal distri-

bution) in order to allow for efficient inference and analytics forms for the Kullback-Leibler divergence. This is exactly the case for VAEs, as show in section 3.4. This constraint over the approximate posterior compromise the quality of the final solution for the variational problem and is particularly severe in cases where the encoding produced by $q_phi(\vec{z}|\vec{x})$ is used for downstream tasks.

One possible solution is to add flexibility to the approximate posterior using normalizing flows, which are a sequence of invertible transformations applied in such a way that the final outcome is also a valid probability distribution. Before explaining normalizing flows, it is required to describe the rule of change of variables in probabilities [18]. Let $\vec{z}_0 \in \mathbb{R}^{\mathrm{d}}$ be a random variable produced by $q_0(\vec{z}_0)$ and $f : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}^{\mathrm{d}}$ be an invertible function. The rule of change of variables state that $\vec{z}_1 = f(\vec{z}_0)$ have the following probability distribution:

$$q_1(\vec{z}_1) = q_0(\vec{z}_0)\left|\det \frac{\partial f}{\partial \vec{z}_0}\right|^{-1} \tag{3.26}$$

If a series of these invertible mappings $f_k, k \in 1, ..., K$ is applied, it results in a normalizing flow that take as input the initial distribution $q_0(\vec{z}_0)$ and outputs the transformed distribution $q_K(\vec{z}_K)$, presented in equations 3.27 and 3.28.

$$\vec{z}_K = f_K \circ f_{k-1} \circ ...f_1(\vec{z}_0), \quad \vec{z}_0 \sim q_0(\vec{z}_0) \tag{3.27}$$

$$\vec{z}_K \sim q_K(\vec{z}_K) = q_0(\vec{z}_0)\prod_{k=1}^{K}\left|\det \frac{\partial f_k}{\partial \vec{z}_{k-1}}\right|^{-1} \tag{3.28}$$

Equation 3.28 describe how the initial probability distribution is transformed by the normalizing flow. Since in each step the rule of change of variables was used, the final distribution is valid, and hence can be used as the new approximate posterior. This allow for building very flexible and arbitrarily complex posterior approximations, depending on the election of the functions f_k.

Therefore, the problem reside in finding functions f_k that fulfill two important conditions. Firstly, they need to be very flexible in terms of learning possible outcomes. Because of this, an ideal candidate would be some type of neural network. Secondly, f_k needs to be easily invertible, since in each forward and backward pass of the normalizing flow, the entire sequence needs to be inverted in order to compute the final distribution. Current research is centered around finding alternatives for these functions. This thesis uses *Inverse Auto-Regressive Flows* (IAFs) [19] , which constitute a very modern approach to normalizing flows that uses a type of neural networks called *Masked Auto-Encoders for Density Estimation* or MADEs [2]. In the following, first a review on MADEs is presented and then IAFs are explained, to finish the chapter showing how IAFs can be applied to traditional VAEs. An approach to implement IAFs to sequential VAEs will be presented in section 4.4.

3.5.1 Masked Auto-Encoders for Density Estimation (MADEs)

MADEs [2] are auto-encoders that satisfy the auto-regressive property. The auto-regressive property state that in a sequence $\vec{x} \in \mathbb{R}^D$ with a certain ordering $[x_1, x_2, ..., x_D]$, the value x_i is only computed using the past or previous values within the sequence. This property can be applied to auto-encoders[1], considering that the output $\hat{x}_\text{i}$ (which in an auto-encoder is the reconstruction of the input x_i) should only be computed using $x_{<\text{i}}$. This property can be obtained shutting down any computational path between $\hat{x}_\text{i}$ and inputs $x_{\geq\text{i}}$, which are supposed to be ignored. In practice, between inputs $[x_\text{i}, x_{\text{i}+1}, x_{\text{i}+2}, ..., x_D]$ and output $\hat{x}_\text{i}$, at least one connection has to be zero to nullify the entire flow.

The easiest way of zeroing connections in a neural network is to apply, in an element-wise manner, a binary matrix with zeros corresponding to the connections that should be eliminated. MADEs are the result of applying this matrices to a traditional Auto-Encoder, creating a new model that enforces the auto-regressive property. In figure 3.5 a MADE structure is presented. Note how in the arbitrarily assigned order, the reconstruction of input number **3** (x_1) is only computed using inputs number **1** (x_2) and **2** (x_3).

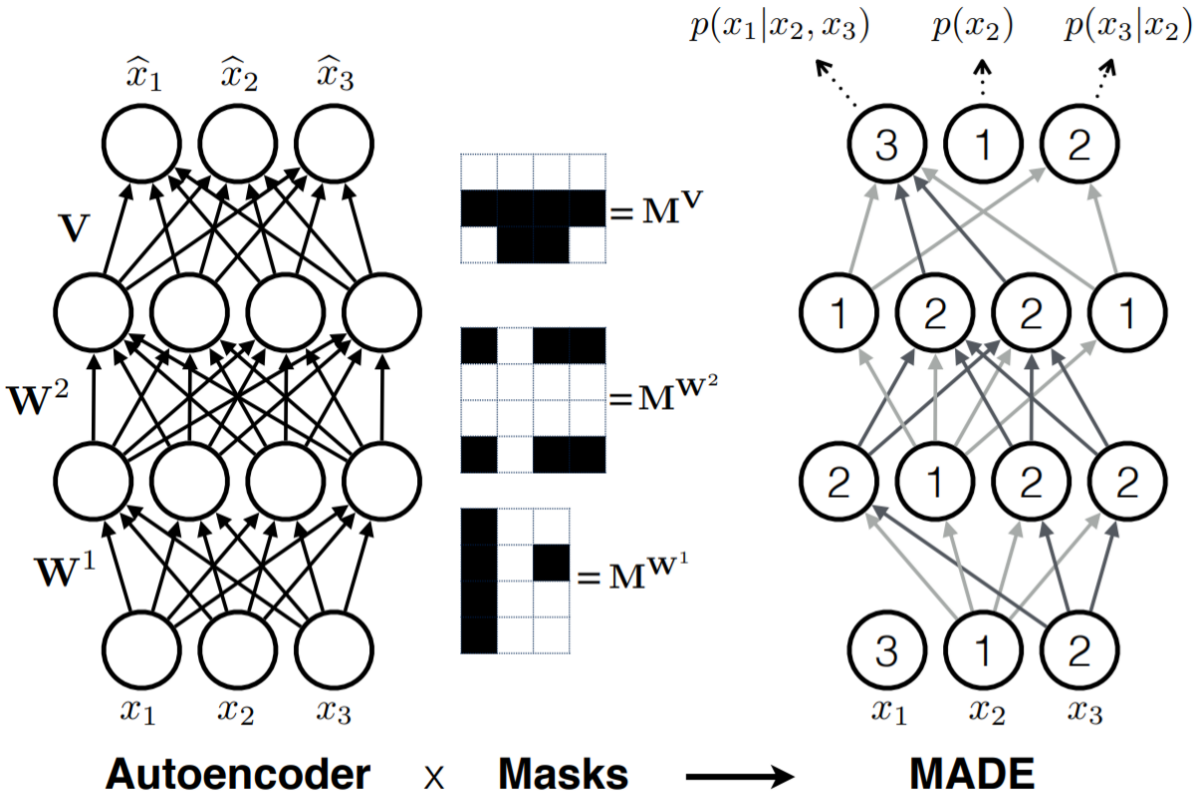


Figure 3.5: Traditional auto-encoder is in the left. Binary masks for each layers are in the center. The final model, MADE, is in the right. Note how connections are turned off by the masks, denoted in a softer gray. This enforces the auto-regressive property in the auto-encoder, producing an output in which each part of the reconstructed sequence in computed using previous inputs. The details of how to compute such binary masks are explained in [2]. Source: *MADE: Masked Autoencoder for Distribution Estimation* [2].

3.5.2 Inverse Auto-Regressive Flows (IAFs)

Inverse Auto-Regressive flows, as any other normalizing flow, consist on the application of an invertible function $f : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}^{\mathrm{d}}$ using the rule of change of variables. For IAFs, the function f that transform $\vec{z}_{k-1}$ into $\vec{z}_k$ is presented in equation 3.29:

$$\vec{z}_k = f(\vec{z}_{k-1}) = \vec{\mu}_k + \vec{\sigma}_k \odot \vec{z}_{k-1} \tag{3.29}$$

Where $\odot$ represent the element-wise multiplication. Both $\vec{\mu}_k$ and $\vec{\sigma}_k$ are the outputs of a MADE network, which is structured to be auto-regressive with respect to the input $\vec{z}_{k-1}$. To show that f is a useful transformation in the context of normalizing flows, the inverse of the determinant of the Jacobian of f is required as equation 3.26 states. In this case, that expression is:

$$\left| \det \frac{\mathrm{d}\vec{z}_k}{\mathrm{d}\vec{z}_{k-1}} \right|^{-1} = \left| \det \left(\frac{\mathrm{d}\vec{\mu}_k}{\mathrm{d}\vec{z}_{k-1}} + \frac{\mathrm{d}\vec{\sigma}_k}{\mathrm{d}\vec{z}_{k-1}} \cdot \vec{z}_{k-1} + \vec{\sigma}_k \cdot I \right) \right|^{-1} \tag{3.30}$$

Using the fact that both $\vec{\mu}_k$ and $\vec{\sigma}_k$ have an auto-regressive structure, it can be prove that their Jacobians, $\frac{\mathrm{d}\vec{\mu}_k}{\mathrm{d}\vec{z}_{k-1}}$ and $\frac{\mathrm{d}\vec{\sigma}_k}{\mathrm{d}\vec{z}_{k-1}}$, are lower-triangular matrices with zeros in the diagonal [19]. To see this, first notice that due to the auto-regressive property which state that since $\mu_{k,\mathrm{i}}$ only depends on the $1 : \mathrm{i} - 1$ components of $\mathrm{d}\vec{z}_{k-1}$, $\frac{\partial \mu_{k,\mathrm{i}}}{\partial z_{k-1,j}} = 0$ for $j \geq \mathrm{i}$. The exact same procedure can be applied to $\vec{\sigma}_k$. Therefore, from equation 3.30 it can be seen that $\frac{\mathrm{d}\vec{\mu}_k}{\mathrm{d}\vec{z}_{k-1}} + \frac{\mathrm{d}\vec{\sigma}_k}{\mathrm{d}\vec{z}_{k-1}} \cdot \vec{z}_{k-1} + \vec{\sigma}_k \cdot I$ will have a lower-triangular structure in which the diagonal is equal to $\vec{\sigma}_k$. Hence, since the determinant of a triangular matrix is the multiplication of the diagonal elements, the following is true:

$$\left| \det \frac{\mathrm{d}\vec{z}_k}{\mathrm{d}\vec{z}_{k-1}} \right|^{-1} = \prod_{\mathrm{i}=1}^{\mathrm{d}} {\sigma_{k,\mathrm{i}}}^{-1} \tag{3.31}$$

With this result, the application of an IAF of K steps to probability distribution $q_0(\vec{z}_0)$ results in the following distribution:

$$\vec{z}_K \sim q_K(\vec{z}_K) = q_0(\vec{z}_0) \prod_{k=1}^{K} \prod_{\mathrm{i}=1}^{\mathrm{d}} {\sigma_{k,\mathrm{i}}}^{-1} \tag{3.32}$$

And therefore, the log-probability of the posterior approximation is:

$$\log q_K(\vec{z}_K) = \log q_0(\vec{z}_0) + \log \prod_{k=1}^{K} \prod_{\mathrm{i}=1}^{\mathrm{d}} {\sigma_{k,\mathrm{i}}}^{-1} \tag{3.33}$$

$$\log q_K(\vec{z}_K) = \log q_0(\vec{z}_0) - \sum_{k=1}^{K} \sum_{\mathrm{i}=1}^{\mathrm{d}} \log \sigma_{k,\mathrm{i}} \tag{3.34}$$

Since both $\vec{\mu}_k$ and $\vec{\sigma}_k$ are the outputs of a MADE, which is a neural network of an arbitrarily number of hidden layers and neuron within those layers, in theory an IAF should be able (with proper training and sufficient amounts of data) to transform the original probability distribution without limits in term of complexity or representation.

3.5.3 IAFs in Traditional Variational Auto-Encoders

In this section, it will be explain how an inverse auto-regressive flow can be used within a traditional variational auto-encoder to add flexibility to the approximate posterior.

In the traditional VAE model, the approximate posterior is defined as a diagonal normal distribution, using the outputs of the encoder network as parameters. As state in [19], the IAF block should take this distribution as the start of the flow, i.e, $q_0(\vec{z}_0|\vec{x}) = q(\vec{z}|\vec{x})$ if the nomenclature of normalizing flows is used. This allow us to obtain, after K steps of the IAF, a final distribution $q_K(\vec{z}_K|\vec{x})$ which in theory should be more flexible in its representation of the latent space. The modified VAE model and the IAF block is presented in the form of a diagram in figure 3.6.

The modifications made by the IAF to the VAE model also affects the ELBO function. Because of this, the ELBO function needs to be derived again using $q_K(\vec{z}|\vec{x})$ as the new approximation of the true posterior $p(\vec{z}|\vec{x})$. As in section 3.3, the process starts with the Kullback-Leibler divergence between these distributions, where the dependency of q and p on the parameters ϕ and θ have been made implicit for readability purposes:

$$KL(q_K(\vec{z}|\vec{x})||p(\vec{z}|\vec{x})) = \mathbb{E}_{\vec{z}\sim q_K(\vec{z}|\vec{x})}[\log q_K(\vec{z}|\vec{x}) - \log p(\vec{z}|\vec{x})] \tag{3.35}$$

The Law of the Unconscious Statistician (LOTUS) [18] state that any expectation $\mathbb{E}_{\vec{z}\sim q_K}[h(\vec{z})]$ can be written as an expectation under q_0 as:

$$\mathbb{E}_{\vec{z}\sim q_K}[h(\vec{z})] = \mathbb{E}_{\vec{z}\sim q_0}[h(\vec{z}_k)] = \mathbb{E}_{\vec{z}\sim q_0}[h(f_k \circ f_{k-1} \circ ... \circ f_1(\vec{z}_0))] \tag{3.36}$$

In this case, function $h(\vec{z})$ is recognized as $\log q_K(\vec{z}|\vec{x}) - \log p(\vec{z}|\vec{x})$ and therefore equation 3.37 can be rewritten as:

$$KL(q_K(\vec{z}|\vec{x})||p(\vec{z}|\vec{x})) = \mathbb{E}_{\vec{z}\sim q_0(\vec{z}|\vec{x})}[\log q_K(\vec{z}_K|\vec{x}) - \log p(\vec{x}|\vec{z}_K) - \log p(\vec{z}_K)] + \log p(x) \tag{3.37}$$

Where the Bayes theorem and the logarithm properties have been used to transform $p(\vec{z}_K|\vec{x})$ and $p(x)$ has been removed from the expectation since it does not depend on the

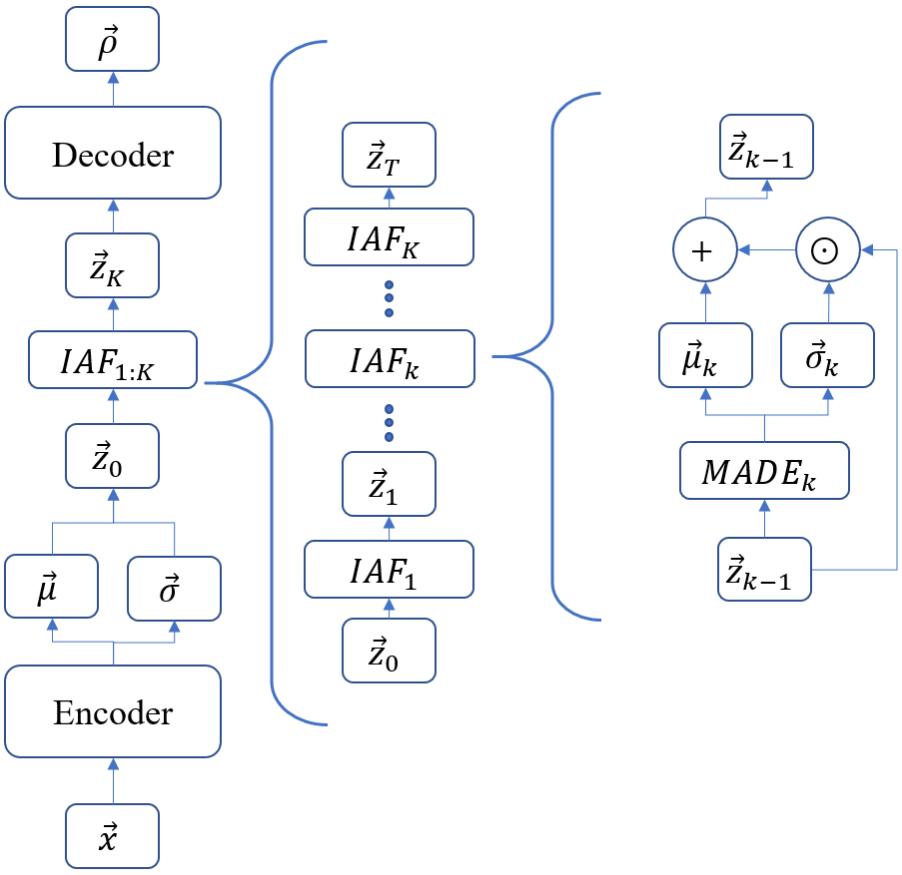


Figure 3.6: Variational Auto-Encoder diagram with the insertion of a IAF computation block of T steps.

latent variables. The new ELBO can be formed in the right hand side rearranging the terms and using the linearity of the expectation:

$$\log p(x) - KL(q_K(\vec{z}|\vec{x})||p(\vec{z}|\vec{x})) = \mathbb{E}_{\vec{z}\sim q_0(\vec{z}|\vec{x})}[\log p(\vec{x}|\vec{z}_K)] - \mathbb{E}_{\vec{z}\sim q_0(\vec{z}|\vec{x})}[\log q_K(\vec{z}_K|\vec{x}) - \log p(\vec{z}_K)] \tag{3.38}$$

Without normalizing flows, the second term of the right hand side resulted in the Kullback-Leibler divergence between the approximate posterior and the prior distribution. When using normalizing flows, specifically inverse auto-regressive flows, this term results in:

$$\mathbb{E}_{\vec{z}\sim q_0(\vec{z}|\vec{x})}[\log q_K(\vec{z}_K|\vec{x}) - \log p(\vec{z}_K)] = \mathbb{E}_{\vec{z}\sim q_0(\vec{z}|\vec{x})}[\log q_0(\vec{z}_0) - \sum_{k=1}^{K}\sum_{\mathrm{i}=1}^{\mathrm{d}} \log \sigma_{k,\mathrm{i}} - \log p(\vec{z}_K)] \tag{3.39}$$

Where equation 3.34 has been used to replace the term $\log q_K(\vec{z}_K|\vec{x})$. As it can be seen, this term does not result in the Kullback-Leibler divergence previously obtained due to the prior distribution, which now has as an argument the transformed latent variable $\vec{z}_K$.

Finally, the ELBO function for the case where IAF are used to add flexibility to the VAE is presented in equation 3.40.

$$ELBO_{IAF} = \mathbb{E}_{\vec{z}\sim q_0(\vec{z}|\vec{x})}[\log p(\vec{x}|\vec{z}_K)] - \mathbb{E}_{\vec{z}\sim q_0(\vec{z}|\vec{x})}[\log q_0(\vec{z}_0) - \sum_{k=1}^{K}\sum_{\mathrm{i}=1}^{\mathrm{d}} \log \sigma_{k,\mathrm{i}} - \log p(\vec{z}_K)] \quad (3.40)$$

Regrettably, this new form for the ELBO function loss the analytic computation of the Kullback-Leibler divergence. Instead, the second term in the right hand side is computed using Monte Carlo approximation taking samples of $\vec{z}_0 \sim q_0$.

Chapter 4

Proposed Temporal Variational Auto-Encoders

This chapter[1] described the proposed modifications to the traditional Variational Auto-Encoder model in order to make it possible to process temporal data and also generate a sequential latent space. For this, three inner structures for the encoder and decoder of the VAE are proposed, each with a different level of complexity and temporal awareness. The modifications that such variants impose over the ELBO function are also studied and a modified objective function is finally derived. The chapter ends with an explanation of the proposed approach to incorporate non-sequential normalizing flows to sequential latent spaces.

In temporal Variational Auto-Encoders (tVAEs), the input data is configured as $\mathbf{x} = \{\vec{x}_1, ..., \vec{x}_T\}$ where T is the number of time-steps of the sequence and $\vec{x}_t$ is a snapshot of operational variables sampled at regular intervals that describe a certain process. The latent variables are also configured in a sequential manner ($\mathbf{z} = \{\vec{z}_1, ..., \vec{z}_T\}$) to allow the latent space to encode temporal information.

This arrangement of the observable and latent features imposes the following modifications to the traditional VAE model:

- The encoder's neural network of the tVAE should be able to admit time series data as the input. It also needs to return sequences for the variational parameters μ and σ in order to generate a sequential latent space. Therefore, the traditional neural network used in the vanilla VAE model is replaced by some other architecture capable of processing sequences, such as a time-distributed neural network or a recurrent neural network.
- The same is also true for the decoder's neural network of the tVAE, since it needs to process the sequential latent space and output a reconstruction of the input time series.

Different architectures for the encoder and decoder will produce variants of the original

[1]From here onwards, this thesis will use bold variables to indicate sequences of vectors, in the format $\mathbf{a} = \vec{a}_1, ..., \vec{a}_T$.

ELBO function. To see this, lets note first that the derivation of the original ELBO is still valid whether the input data and latent space is sequential or non-sequential. Therefore, equation 4.1 presents the same ELBO function as the one presented in section 3.3, where x and z has been changed for $\mathbf{x}$ and $\mathbf{z}$ to explicitly denote the temporality dependence in both the input data and latent space.

$$ELBO = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})\sim Q}[\log p_\theta(\mathbf{x}|\mathbf{z})] - KL(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{4.1}$$

The definition of the Kullback-Leibler divergence is used to expand the previous expression and obtain:

$$ELBO = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})\sim Q}[\log p_\theta(\mathbf{x}|\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{z})] \tag{4.2}$$

At this point, two strong assumptions regarding the independence between time-steps within $\mathbf{x}$ and $\mathbf{z}$ are needed to reach a computable form. Those assumptions are listed bellow:

1. It is assumed that $\vec{x}_t$ for $t = \{1, 2, ..., T\}$ are conditionally independent between them given the full sequence of latent variables $\mathbf{z}$. This supposition is formally presented in equation 4.3.

$$\vec{x}_t \perp \vec{x}_{\neg t} \mid \mathbf{z} \tag{4.3}$$

   By only allowing $\vec{x}_t$ to depend on the latent variables, the model is forced to encode all the necessary information for reconstruction into $\mathbf{z}$, thus generating a very rich alternative representation of the observable variables that can be used later for downstream tasks. Therefore, the decoder distribution of the VAE can rewritten in a temporal or sequential setting as:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{t=1}^{T} p_\theta(\vec{x}_t|\mathbf{z}) \tag{4.4}$$

2. It is also assumed that $\vec{z}_t$ for $t = \{1, 2, ..., T\}$ are independent between them. Although this is not a necessary supposition for the development of the ELBO function in the temporal setting, and approaches the VAEs without this assumption do exists (for example, [15]), models with dependence between latent variables are outside of the scope of this thesis. With this supposition, the encoder distribution of the VAE can be rewritten as:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^{T} q_\phi(\vec{z}_t|\mathbf{x}) \tag{4.5}$$

   The prior distribution is also separable due to the independence between the latent variables, as show in the following equation:

$$p(\mathbf{z}) = \prod_{t=1}^{T} p(\vec{z}_t) \tag{4.6}$$

The modified ELBO function that incorpore these two suppositions is presented in equation 4.7.

$$ELBO = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})\sim Q}[\log \prod_{t=1}^{T} p_\theta(\vec{x}_t|\mathbf{z}) - \log \prod_{t=1}^{T} q_\phi(\vec{z}_t|\mathbf{x}) + \log \prod_{t=1}^{T} p(\vec{z}_t)] \tag{4.7}$$

Using the properties of the logarithm, the product operations can be transformed into sums:

$$ELBO = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})\sim Q}[\sum_{t=1}^{T} \log p_\theta(\vec{x}_t|\mathbf{z}) - \sum_{t=1}^{T} \log q_\phi(\vec{z}_t|\mathbf{x}) + \sum_{t=1}^{T} \log p(\vec{z}_t)] \tag{4.8}$$

Since the expectation is a linear operation, the order of the sum and the expectation can be interchanged to obtain:

$$ELBO = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})\sim Q}[\log p_\theta(\vec{x}_t|\mathbf{z}) - \log q_\phi(\vec{z}_t|\mathbf{x}) + \log p(\vec{z}_t)] \tag{4.9}$$

The final step is to use the definition of the expectation since it is easier to further manipule the expression using integrals.

$$ELBO = \sum_{t=1}^{T} \int_{\Omega_\mathbf{z}} q_\phi(\vec{z}_1|\mathbf{x}) q_\phi(\vec{z}_2|\mathbf{x}) ... q_\phi(\vec{z}_T|\mathbf{x}) (\log p_\theta(\vec{x}_t|\mathbf{z}) - \log q_\phi(\vec{z}_t|\mathbf{x}) + \log p(\vec{z}_t)) \mathrm{d}\mathbf{z} \tag{4.10}$$

Where the independence of latent variables from equation 4.5 has been used.

From here, the specific form for the encoder and decoder neural networks of the tVAE are needed to generate additional suppositions regarding the relationship between $\vec{z}_t$ and $\vec{x}_t$ in order to keep developing equation 4.10, since those architectures will impose different temporal dependencies between $\mathbf{x}$ and mz. For this thesis, three architectures with different degrees of complexity in terms of temporal information flux between observable and latent variables are proposed. In what follows, these three alternatives are presented in conjunction with the final ELBO form that they produce taking as a start point equation 4.10.

4.1 Temporal independent tVAE (tVAE1)

The temporal independent tVAE, denoted as tVAE1, supposes that cross-time dependencies between $\mathbf{x}$ and $\mathbf{z}$ are not significant nor needed to generate the latent space or to reconstruct the input data. In other words, for the encoder, the latent vector $\vec{z}_t$ will only depend on the input data for that time-step, $\vec{x}_t$. The same concept is true for the decoder: the reconstruction for time-step t will only depend on the latent space for that time-step, $\vec{z}_t$. These

conditional suppositions for the encoder and decoder are presented in equations 4.11 and 4.12, respectively.

$$\vec{z}_t \sim q_\phi(\vec{z}_t|\vec{x}_t) \tag{4.11}$$

$$\vec{x}_t \sim p_\theta(\vec{x}_t|\vec{z}_t) \tag{4.12}$$

The application of an encoder and decoder with these conditional forms results in an operation that is equivalent to apply a traditional VAE in a time-distributed manner, i.e., applying the same traditional VAE for each time step of the input sequence, in a forward manner from $t = 1$ to $t = T$. This model could be beneficial for those cases where the input data does not present a relevant temporal behavior, for example, in operational data that has an interval between measures considerably longer that the natural scale of the phenomena that is being captured. An important feature of this configuration is that it assumes that the encoder model (equation 4.11) will be capable of compressing all the necessary information to reconstruct $\vec{x}_t$ into just one time-step of the latent variables, $\vec{z}_t$. This is of course a very strong assumption, which will probably not be fulfilled due to the natural limitations in approximation power of the deep learning models that are used to implement this model.

To correctly derive the ELBO function considering the assumptions made for this type of model, the previously mentioned assumptions are integrated into equation 4.13 to yield:

$$ELBO = \sum_{t=1}^{T} \int_{\Omega_{\mathbf{z}}} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_2) ... q_\phi(\vec{z}_T|\vec{x}_T) (\log p_\theta(\vec{x}_t|\vec{z}_t) - \log q_\phi(\vec{z}_t|\vec{x}_t) + \log p(\vec{z}_t)) \mathrm{d}\mathbf{z} \tag{4.13}$$

The previous equation can be expanded to obtain:

$$\begin{aligned}
ELBO = & \\
& \int_{\vec{z}_T} ... \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_2) ... q_\phi(\vec{z}_T|\vec{x}_T) (\log p_\theta(\vec{x}_1|\vec{z}_1) - \log q_\phi(\vec{z}_1|\vec{x}_1) + \log p(\vec{z}_1)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T \\
& + \int_{\vec{z}_T} ... \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_2) ... q_\phi(\vec{z}_T|\vec{x}_T) (\log p_\theta(\vec{x}_2|\vec{z}_2) - \log q_\phi(\vec{z}_2|\vec{x}_2) + \log p(\vec{z}_2)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T \\
& + \ ... \\
& + \int_{\vec{z}_T} ... \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_2) ... q_\phi(\vec{z}_T|\vec{x}_T) (\log p_\theta(\vec{x}_T|\vec{z}_T) - \log q_\phi(\vec{z}_T|\vec{x}_T) + \log p(\vec{z}_T)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T
\end{aligned} \tag{4.14}$$

It is easy to see that for the $\tilde{t}$ term, the only integral that contributes to the final ELBO is the one over the latent variable $\vec{z}_{\tilde{t}}$, since the rest of them integrate only over a probability

distribution, resulting in 1. With this in consideration, equation 4.14 can be reduced to:

$$\begin{aligned} ELBO = & \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1)(\log p_\theta(\vec{x}_1|\vec{z}_1) - \log q_\phi(\vec{z}_1|\vec{x}_1) + \log p(\vec{z}_1))\mathrm{d}\vec{z}_1 \\ & + \int_{\vec{z}_2} q_\phi(\vec{z}_2|\vec{x}_2)(\log p_\theta(\vec{x}_2|\vec{z}_2) - \log q_\phi(\vec{z}_2|\vec{x}_2) + \log p(\vec{z}_2))\mathrm{d}\vec{z}_2 \\ & + \ ... \\ & + \int_{\vec{z}_T} q_\phi(\vec{z}_T|\vec{x}_T)(\log p_\theta(\vec{x}_T|\vec{z}_T) - \log q_\phi(\vec{z}_T|\vec{x}_T) + \log p(\vec{z}_T))\mathrm{d}\vec{z}_T \end{aligned} \tag{4.15}$$

Recognizing a simpler summation of expectations, The final form of the ELBO for the temporal-independent tVAE is reach:

$$ELBO = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\vec{z}_t|\vec{x}_t)\sim Q}[\log p_\theta(\vec{x}_t|\vec{z}_t) - \log q_\phi(\vec{z}_t|\vec{x}_t) + \log p(\vec{z}_t)] \tag{4.16}$$

Equation 4.16 reveals that in this case, the total ELBO is just the sum of the traditional ELBO for each time step of the sequence, in which the Kullback-Liebler divergence can be formed to reach:

$$ELBO = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\vec{z}_t|\vec{x}_t)\sim Q}[\log p_\theta(\vec{x}_t|\vec{z}_t) - KL(q_\phi(\vec{z}_t|\vec{x}_t)||p(\vec{z}_t))] \tag{4.17}$$

In what follows, the implementation of the encoder and decoder neural network of tVAE1 is presented.

4.1.1 Implementation details for the tVAE1 model

From an implementation point of view, the types of conditional relationships that are presented in equation 4.17 between observable and latent variables can be modeled using traditional neural networks (see section 3.2.1) in both the encoder and decoder networks, applied in a time-distributed manner. That is, the same neural networks is applied to each time-step consecutively. The application of any neural network with a more complex architecture, such as a recurrent neural network would cause the violation of the suppositions established in equations 4.11 and 4.12, as it will be shown in the following sections. This is equivalent to apply a traditional VAE model to every vector of the sequence independently.

In terms of parametric distributions, this model uses the same families as those chosen for the traditional VAE model (see section 3.4). That is, the prior $p(\vec{z}_t)$ will be an isotropic normal distribution for each t, the decoder $p_\theta(\vec{x}_t|\vec{z}_t)$ will be a Bernoulli distribution and finally, the decoder $q_\phi(\vec{z}_t|\vec{x}_t)$ will continue to be a diagonal normal distribution. Figure 4.1 presents a diagram of this model, in which the time distributed application of the traditional neural networks is explicitly shown.

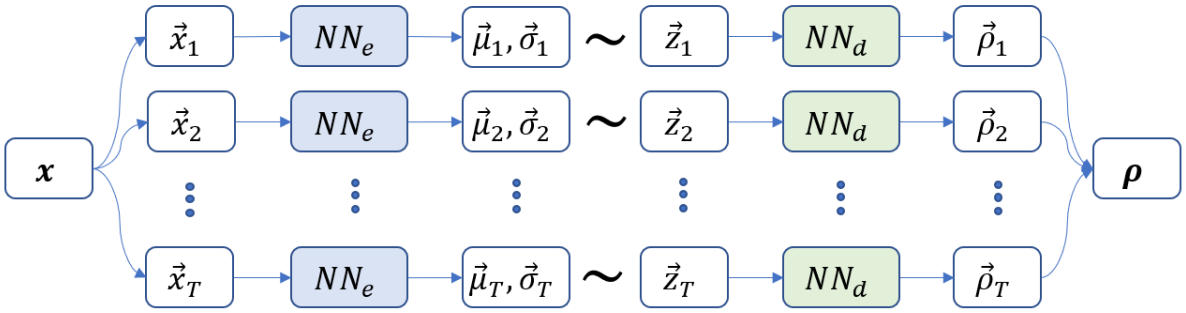


Figure 4.1: Block's diagram corresponding to the time-independent tVAE model. Note how in this model, the same traditional neural network (light blue block) is applied to each vector of the input sequence to obtain the posterior parameters. Then, the latent space is sampled from the posterior distribution using the corresponding set of parameters and the reparameterization trick (omitted here for visualization purposes). Finally, each vector within the generated latent space sequence is fed to the decoder neural network (light green block), which is repeatedly applied for each time step to produce the reconstruction of the input data.

4.1.2 Forward pass for the tVAE1

The forward pass of this model, taking as input the sequence $\mathbf{x} = \{\vec{x}_1, ..., \vec{x}_T\}$, consist on the following steps:

1. Using $\vec{x}_1$ as input, apply the traditional neural network of the encoder to compute the variational parameters $\vec{\mu}_1$ and $\vec{\sigma}_1$.
2. Using the reparameterization trick and the variational parameters from the previous step, sample the latent vector $\vec{z}_1$.
3. Using $\vec{z}_1$ as input, apply the traditional neural network of the decoder to compute the reconstruction vector $\vec{\rho}_1$.
4. Repeat the steps 1-3 for $t = 2, ...T$

Once the forward pass is completed for a particular data point $\mathbf{x}_\text{i}$, the ELBO function for the tVAE1 model presented in equation 4.17 can be computed with the same considerations as the original ELBO (see section 3.4.5).

4.2 Past and present dependent tVAE (tVAE2)

The past and present dependent tVAE, denoted as tVAE2, supposes for the encoder distribution that the latent variables for time t, $\vec{z}_t$, can only depend on past and present input data, $\vec{x}_{1:t}$. A similar supposition is made for the decoder, in which the reconstruction of the observable vector $\vec{x}_t$ can only depend upon the past and present latent variable $\vec{z}_{1_t}$. Therefore, the latent variables act as a summary of past information that contains all the necessary information to reconstruct the present observable state. This conditional suppositions are presented in equations 4.18 and 4.19.

$$\vec{z}_t \sim q_\phi(\vec{z}_t|\vec{x}_{1:t}) \tag{4.18}$$

$$\vec{x}_t \sim p_\theta(\vec{x}_t|\vec{z}_{1:t}) \tag{4.19}$$

For cases where the sequence input data show an important temporal behavior, a tVAE with these suppositions should be able to generate a better latent space than those generated by the tVAE1 model presented in section 4.1 due to the utilization of past information to generate the present state. In addition, because the encoder no longer have to compress all the necessary information to reconstruct $\vec{x}_t$ into $\vec{z}_t$, this configuration lessen the pressure upon that network, possibly allowing for more meaningful representations to be generated.

To correctly derive the ELBO function for this tVAE model considering the assumptions made in equations 4.18 and 4.19, the derivation presented in section 4.1 can be modified from equation 4.13 onward, adding past information to the encoder and decoder distribution. In that case, the following is obtained:

$$ELBO = \sum_{t=1}^{T} \int_{\Omega_{\mathbf{z}}} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_{1:2}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_t|\vec{z}_{1:t}) - \log q_\phi(\vec{z}_t|\vec{x}_{1:t}) + \log p(\vec{z}_t)) \mathrm{d}\mathbf{z} \tag{4.20}$$

In a derivation procedure similar to the one explained in the previous section, this expression can be expanded to obtain:

$$\begin{aligned}
&ELBO = \\
&\int_{\vec{z}_T} ... \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_{1:2}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_1|\vec{z}_1) - \log q_\phi(\vec{z}_1|\vec{x}_1) + \log p(\vec{z}_1)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T \\
&+ \int_{\vec{z}_T} ... \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_{1:2}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_2|\vec{z}_{1:2}) - \log q_\phi(\vec{z}_2|\vec{x}_{1:2}) + \log p(\vec{z}_2)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T \\
&+ \ ... \\
&+ \int_{\vec{z}_T} ... \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1) q_\phi(\vec{z}_2|\vec{x}_{1:2}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_T|\vec{z}_{1:T}) - \log q_\phi(\vec{z}_T|\vec{x}_{1:T}) + \log p(\vec{z}_T)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T
\end{aligned} \tag{4.21}$$

Note in the previous equation how the dependencies have changed to include past information as well. Now, for the $\tilde{t}$ term, the only integrals that contribute to the final ELBO is the one over the latent variables $\vec{z}_{\leq \tilde{t}}$, since the rest of them integrate only over a probability distribution, resulting in the unity. With this in consideration, equation 4.21 can be reduced

to:

$$
\begin{aligned}
&ELBO = \\
&\int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1)(\log p_\theta(\vec{x}_1|\vec{z}_1) - \log q_\phi(\vec{z}_1|\vec{x}_1) + \log p(\vec{z}_1))\mathrm{d}\vec{z}_1 \\
&+ \int_{\vec{z}_2}\int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1)q_\phi(\vec{z}_2|\vec{x}_{1:2})(\log p_\theta(\vec{x}_2|\vec{z}_{1:2}) - \log q_\phi(\vec{z}_2|\vec{x}_{1:2}) + \log p(\vec{z}_2))\mathrm{d}\vec{z}_1\mathrm{d}\vec{z}_2 \\
&+ \ ... \\
&+ \int_{\vec{z}_T} ... \int_{\vec{z}_2}\int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_1)q_\phi(\vec{z}_2|\vec{x}_{1:2})...q_\phi(\vec{z}_T|\vec{x}_{1:T})(\log p_\theta(\vec{x}_T|\vec{z}_{1:T}) - \log q_\phi(\vec{z}_T|\vec{x}_{1:T}) + \log p(\vec{z}_T))\mathrm{d}\vec{z}_1\mathrm{d}\vec{z}_2...\mathrm{d}\vec{z}_T
\end{aligned}
\tag{4.22}
$$

Using the definition of the expectation and reorganizing the terms with a summation the final form of the ELBO for the past and present dependent tVAE is obtained:

$$
ELBO = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\vec{z}_1|\vec{x}_1)...q_\phi(\vec{z}_t|\vec{x}_{1:t})}[\log p_\theta(\vec{x}_t|\vec{z}_{1:t}) - \log q_\phi(\vec{z}_t|\vec{x}_{1:t}) + \log p(\vec{z}_t)] \tag{4.23}
$$

In this case, equation 4.23 shows that the final ELBO is just the sum of the ELBO function for each time step conditioning the observable and latent variables on past and present latent and observable variables, respectively. Nevertheless, the distributions are still defined over single vectors for each time step. Only the conditionals are partial sequences of vectors. This allows the recognition of the Kullback-Leibler divergence term in equation 4.23, to obtain:

$$
ELBO = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\vec{z}_1|\vec{x}_1)...q_\phi(\vec{z}_t|\vec{x}_{1:t})}[\log p_\theta(\vec{x}_t|\vec{z}_{1:t}) - KL(q_\phi(\vec{z}_t|\vec{x}_{1:t})||p(\vec{z}_t))] \tag{4.24}
$$

It is important to notice that this Kullback-Leibler divergence will still have an analytic form since the posterior and prior distributions are still a diagonal Gaussian and an isotropic normal distribution respectively. In what follows, the implementation of the encoder and decoder neural network of the tVAE2 model is presented.

4.2.1 Implementation details for tVAE2 model

From an implementation point of view, equation 4.24 indicate that both the encoder and decoder neural networks need to take information in a sequential manner, considering the past as well as the present. This type of behavior is typical of long-term memory neural networks, such as LSTMs (see section 3.2.2). A traditional recurrent neural networks would not qualify since it can only represent conditional structures conditioning on one time-step in the past (i.e. $\vec{z}_t \sim q_\phi(\vec{z}_t|\vec{x}_{t-1,t})$) due to its short memory span. In terms of parametric distributions for the prior, decoder and encoder, the same families as those chosen for the tVAE1 model are used. In figure 4.2 a diagram of this model is presented.

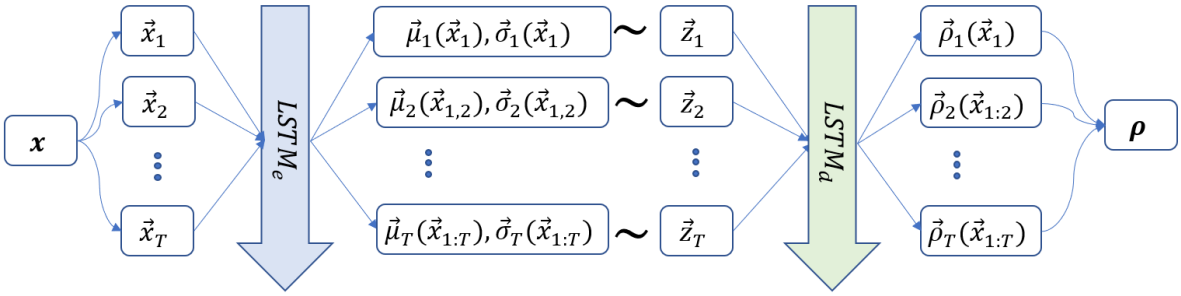


Figure 4.2: Block's diagram corresponding to the Past and Present dependent tVAE model. In this model, a LSTM neural network is applied to the input sequence, generating the desired temporal dependence in the posterior parameters, which is indicated as arguments for $\vec{\mu}_t$ and $\vec{\sigma}_t$. Using this parameters, each vector of the latent space is sampled from the posterior distribution. A second LSTM neural network uses this sequential latent space as input to generate the reconstruction of the input data, also respecting the desired conditional dependence of this model. In both the encoder and decoder's LSTM networks, the direction of the arrow indicate the direction in which information flows through time. For this model, information flows from past to present.

4.2.2 Forward pass for the tVAE2

The forward pass of this model, taking as input the sequence $\mathbf{x} = \{\vec{x}_1, ..., \vec{x}_T\}$, consist on the following steps:

1. Using the encoder's LSTM neural network and the full sequence $\mathbf{x}$ as input, interpret the full sequence of output states as the variational parameters μ and σ. The LSTM inner structure will allow to respect the temporal dependency of this model, obtaining $\vec{\mu}_t$ and $\vec{\sigma}_t$ only as a function of $\vec{x}_{\leq t}$
2. Using the reparameterization trick, sample the latent vector $\vec{z}_t$ using only the variational parameters that correspond with that time step, $\vec{\mu}_t$ and $\vec{\sigma}_t$. This is done for $t = 1, ...T$.
3. Using the decoder's LSTM neural network with the full sequence $\mathbf{z}$ as input, obtain the reconstruction sequence ρ as the full sequence of output states. Again, the LSTM inner structure will allow to respect the temporal dependency of the tVAE2 model, obtaining $\vec{\rho}_t$ only as a function of $\vec{z}_{\leq t}$.

Once the forward pass is completed for a particular data point $\mathbf{x}_\text{i}$, the ELBO function for the tVAE2 model presented in equation 4.24 can be computed with the same considerations as the original ELBO (see section 3.4.5).

4.3 Full temporal dependent tVAE (tVAE3)

For the full temporal dependent tVAE, denoted as tVAE3, the encoder and decoder distributions can condition on the entire sequences, allowing both $\vec{x}_t$ and $\vec{z}_t$ to depend on the full range of the latent or observable variables, respectively. This conditional suppositions is presented in equations 4.25 and 4.26.

$$\vec{z}_t \sim q_\phi(\vec{z}_t|\vec{x}_{1:T}) \tag{4.25}$$

$$\vec{x}_t \sim p_\theta(\vec{z}_t|\vec{x}_{1:T}) \tag{4.26}$$

Here, the use of future information to compute past values of the observable and latent features does not constitute an invalid model given one condition, which is that the tVAE model cannot be used for online learning. This is because in online learning, the whole window of T time-steps should be available for the backwards use of temporal information. If this condition is not violated, there is nothing inheritable wrong to use all the available information to either encode the input data with the encoder (equation 4.25) or to reconstruct the information with the decoder (equation 4.26). The application of this configuration can also be interpreted as using future information to fine tune the current latent or reconstructed observable variables in order to smooth the change between time-steps.

The derivation of the ELBO function for this case follows a very similar derivation as the previous tVAE models. Starting from equation 4.20, the suppositions made in equations 4.18 and 4.19 can be incorporated to obtain:

$$ELBO = \sum_{t=1}^{T} \int_{\Omega_{\mathbf{z}}} q_\phi(\vec{z}_1|\vec{x}_{1:T}) q_\phi(\vec{z}_2|\vec{x}_{1:T}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_t|\vec{z}_{1:T}) - \log q_\phi(\vec{z}_t|\vec{x}_{1:T}) + \log p(\vec{z}_t)) \mathrm{d}\mathbf{z} \tag{4.27}$$

In a derivation procedure similar to the one explained in the previous section, the summation and the integrals can be expanded to obtain:

$$\begin{aligned}
ELBO = & \\
& \int_{\vec{z}_T} \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_{1:T}) q_\phi(\vec{z}_2|\vec{x}_{1:T}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_1|\vec{z}_{1:T}) - \log q_\phi(\vec{z}_1|\vec{x}_{1:T}) + \log p(\vec{z}_1)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T \\
& + \int_{\vec{z}_T} \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_{1:T}) q_\phi(\vec{z}_2|\vec{x}_{1:T}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_2|\vec{z}_{1:T}) - \log q_\phi(\vec{z}_2|\vec{x}_{1:T}) + \log p(\vec{z}_2)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T \\
& + \ ... \\
& + \int_{\vec{z}_T} \int_{\vec{z}_2} \int_{\vec{z}_1} q_\phi(\vec{z}_1|\vec{x}_{1:T}) q_\phi(\vec{z}_2|\vec{x}_{1:T}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T}) (\log p_\theta(\vec{x}_T|\vec{z}_{1:T}) - \log q_\phi(\vec{z}_T|\vec{x}_{1:T}) + \log p(\vec{z}_T)) \mathrm{d}\vec{z}_1 \mathrm{d}\vec{z}_2 ... \mathrm{d}\vec{z}_T
\end{aligned} \tag{4.28}$$

Note that in the previous equation the dependencies have changed to include the full range of temporal information. Now, for the $\tilde{t}$ term, every integral contributes to the final ELBO since in there is full temporal dependence over the latent variables. with this consideration in mind plus using the definition of the expectation and reorganizing the terms with a summation the final form of the ELBO for the full temporal dependent tVAE is obtained:

$$ELBO = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\vec{z}_1|\vec{x}_{1:T}) ... q_\phi(\vec{z}_T|\vec{x}_{1:T})} [\log p_\theta(\vec{x}_t|\vec{z}_{1:T}) - \log q_\phi(\vec{z}_t|\vec{x}_{1:T}) + \log p(\vec{z}_t)] \tag{4.29}$$

Equation 4.29 reveals that in this case, the total ELBO is just the sum of the traditional ELBO for each time step of the sequence with the observable and latent variables conditioned

on the full sequence of latent and observable variables, respectively. Nevertheless, the distributions are still defined over single vectors (only the conditionals are defined as sequences), therefore the Kullback-Liebler divergence can be reformed to reach:

$$ELBO = \sum_{t=1}^{T} \mathbb{E}_{q_\phi(\vec{z}_1|\vec{x}_{1:T})\dots q_\phi(\vec{z}_T|\vec{x}_{1:T})}[\log p_\theta(\vec{x}_t|\vec{z}_{1:T}) - KL(q_\phi(\vec{z}_t|\vec{x}_{1:T})||\log p(\vec{z}_t))] \tag{4.30}$$

The Kullback-Leibler divergence of equation 4.30 will still have an analytic form since the posterior and prior distributions are still a diagonal Gaussian and an isotropic normal distribution respectively. In what follows, the implementation of the encoder and decoder neural network of the tVAE3 model is presented.

4.3.1 Implementation details for the full temporal dependent tVAE

In terms of implementation, the type of behavior in which each output state depends on the full input sequence can be obtained with the use of bi-directional long-term recurrent neural networks such as Bi-LSTMs (section 3.2.2), which are known to traverse the temporal sequence by a forward network and a backward network, effectively using all available information to generate the output. These types of networks are used in both the encoder and decoder networks. In terms of parametric distributions for the prior, decoder and encoder, the same families as those chosen for the tVAE1 and tVAE2 models are used. In figure 4.3 a diagram of this model is presented.

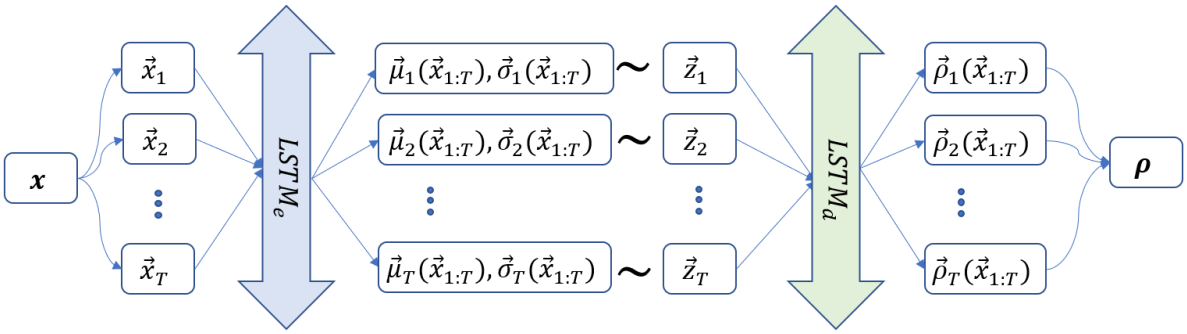


Figure 4.3: Block's diagram corresponding to the full temporal dependent tVAE model. In this model, a Bi-LSTM neural network is applied to the input sequence, giving the posterior parameters for each time-step full awareness of past, present and future states. As always, the latent space is sampled from the variational distribution using this parameters and the reparameterization trick. In the decoder, a second Bi-LSTM neural network uses the sequential latent space as input to generate the reconstruction of the input data, in which for this model each reconstructed vector $\vec{x}_t$ is fully aware of past, present and future latent variables.

4.3.2 Forward pass for the tVAE3

The forward pass of this model, taking as input the sequence $\mathbf{x} = \{\vec{x}_1, ..., \vec{x}_T\}$, consist on the following steps:

1. Using the encoder's Bi-LSTM neural network and the full sequence $\mathbf{x}$ as input, interpret the full sequence of output states as the variational parameters μ and σ. The Bi-LSTM inner structure will allow to respect the temporal dependency of this model, obtaining $\vec{\mu}_t$ and $\vec{\sigma}_t$ as a function of the full sequence $\vec{x}_{1:T}$
2. Using the reparameterization trick, sample the latent vector $\vec{z}_t$ using only the variational parameters that correspond with that time step, $\vec{\mu}_t$ and $\vec{\sigma}_t$. This is done for $t = 1, ...T$.
3. Using the decoder's Bi-LSTM neural network with the full sequence $\mathbf{z}$ as input, obtain the reconstruction sequence ρ as the full sequence of output states. Again, the Bi-LSTM inner structure will allow to respect the temporal dependency of the tVAE2 model, obtaining $\vec{\rho}_t$ as a function of the full sequence $\vec{z}_{1:T}$.

Once the forward pass is completed for a particular data point $\mathbf{x}_\mathrm{i}$, the ELBO function for the tVAE3 model presented in equation 4.30 can be computed with the same considerations as the original ELBO (see section 3.4.5).

In the following section the proposed approach to use inverse auto-regressive flows in tVAEs is presented.

4.4 Normalizing Flows in tVAEs

Normalizing flows, as explained in section 3.5, are designed to be applied on non-sequential latent spaces. Since for this thesis the proposed models generate a sequential latent space, normalizing flows need to be adapted for that case. This section presents a very simple way of applying non-sequential normalizing flows to the tVAEs models presented in section 4.

For the three tVAEs models proposed, it was assumed that the latent variables were independent across time-steps (see equation 4.5). This assumption should be respected when applying normalizing flows to the approximate posterior distribution. Otherwise, the derivation of the ELBO function derived in section 4 would not be valid. If an inverse auto-regressive flow is directly applied to the sequential latent space, the neural networks within the IAF would mix the vectors $\vec{z}_{1:T}$, making dependencies across time-steps, violating the aforementioned assumption. Therefore, this thesis proposed the use of inverse auto-regressive flows in a time-distributed manner, applying a non-sequential IAF computation block to each sequence within the latent space, one step at a time. This configuration can be seen in figure 4.4.

Using this approach, since the ELBO function of the three tVAE models was the result of the sum of independent traditional ELBO functions with only changes in the conditionals of the distributions, the modifications imposed by the use of inverse auto-regressive flows seen in section 3.5 can be applied separatedly for each time step.

In what follows, the proposed model for semi-supervised learning using tVAEs is presented.

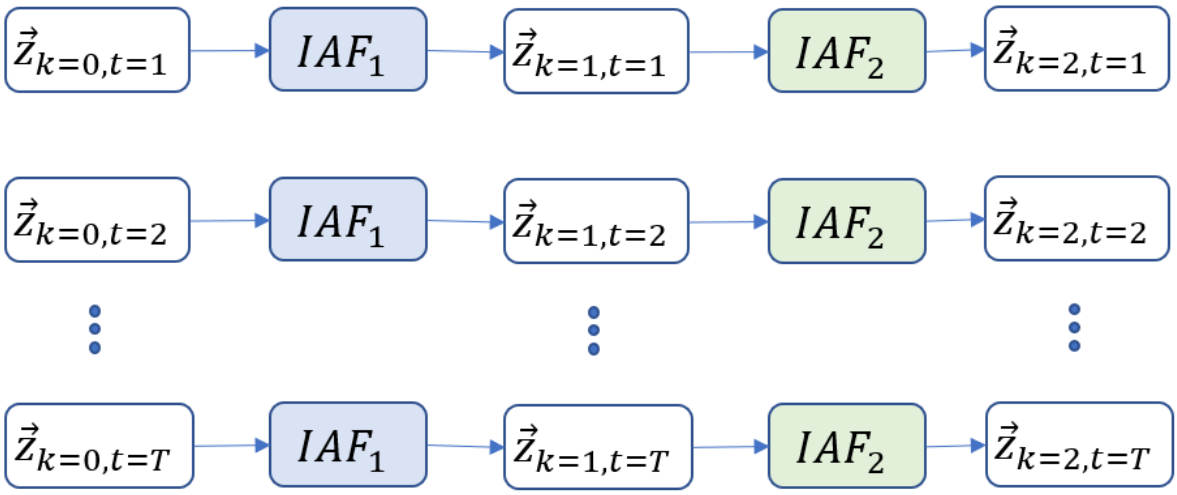


Figure 4.4: An inverse autoregressive flow of two steps ($K = 2$) applied to the sequential latent space of a tVAE. Note how for each step of the flow (denoted by k), the same block. with the same parameters, is applied in each time-step (denoted by t). This notion is reinforced by the use of two distinct colors to indicate two different IAF layers that belongs to the same flow.

Chapter 5

Proposed Approach for Semi-Supervised Learning with tVAEs

In this chapter, the proposed approach that uses tVAEs to perform semi-supervised training is presented. The process starts with a collection of N operational data-points $\mathbb{D} = \{\mathbf{x}_i\}_{i=1}^{N}$, each with dimension $\mathbb{R}^{T \times D}$. A percentage p of the dataset $\mathbb{D}$ correspond to labeled data, denoted as $\mathbb{D}_L = \{\mathbf{x}_i^*, y_i\}_{i=1}^{N_L}$, where $N_L = p \cdot N$ and the superscript $\mathbf{x}^*$ indicate that the data point is labeled. The proposed approach can be summarized on the following steps:

1. Acquire the collection of data-points and identified the data-points that have labels.
2. Choose a tVAE model from the three different variants presented in chapter 4.
3. Construct a recurrent neural network predictor to perform classification or regression, depending on the problem to solve.. From here onwards this recurrent neural network will be refer as the *predictor*.
4. Build a combined model using the tVAE and the predictor following figure 5.1. Note that the predictor take as inputs the location parameter μ of the latent variables as input instead of the observable data $\mathbf{x}$.
5. Train the combined model. For this:
   (a) Sample a batch of b elements from $\mathbb{D}$. This batch also have a labeled subset with an expected value of $p \cdot b$ labeled examples.
   (b) Compute the cost function of the tVAE, denoted as $ELBO(\mathbf{x})$, using all the data-points of the batch.
   (c) Compute the cost function of the predictor, denoted as $L(\mathbf{x}^*, y)$, using only the labeled portion of the batch.
   (d) Obtain the total cost of the model as $C(\mathbf{x}, \mathbf{x}^*, y) = \alpha ELBO(\mathbf{x}) + L(\mathbf{x}^*, y)$, where α is an hyperparameter to control the weight of the ELBO function in the total cost.
   (e) Compute the gradients of the batch with respect to each model parameters.
   (f) Apply the gradients performing stochastic gradient descent to minimize the combined cost of the model C.
   (g) Repeat the training process sampling different batches of $\mathbb{D}$ until convergence.

6. Once the combined model is trained, the sub-model produce by the concatenation of the encoder of the tVAE and the predictor can be used for the supervised tasks (regression or classification)

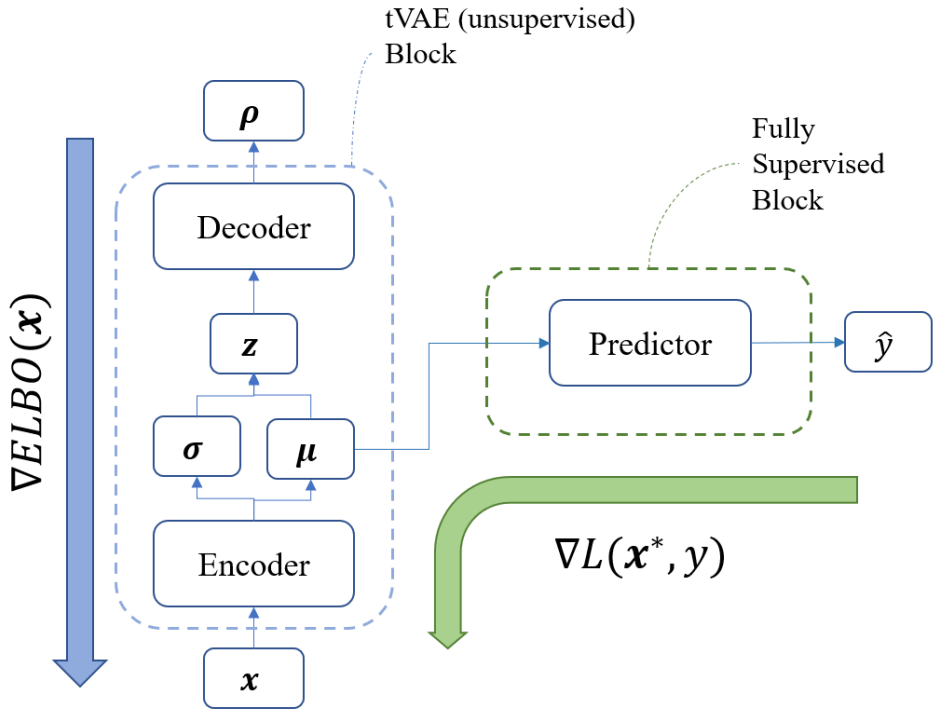


Figure 5.1: The proposed tVAE-SSL model. Note how the gradients of the predictor's loss function, ∇L, also affects the encoder of the VAE.

The justification for a conjunct training between the tVAE and the predictor is explain now. Traditionally, VAEs are used for semi-supervised learning using a decoupled training scheme, as in [14] and [13], where the optimization of the full model is divided in two steps: first, the VAE is trained using all the data in an unsupervised manner to then do a separate supervised training cycle for the predictor neural network in which only the transformed data that originally had labels is used. Nevertheless, this decoupled strategy have two main disadvantages. The first one is that the models need to be tuned separately, which is usually slower than tuning just one model. The second disadvantage is that the most important source of information for the downstream task (the labels) is not being used to train the encoder of the VAE, which is the part of the model that generate the latent space. This is clearly a waste of information that can be corrected with the proposed approach. Using a coupled training process, the VAE as well as the predictor can be tuned in just one training cycle, and as it show in figure 5.1, the gradients of L with respect to the parameters of the encoder will not be zero since the prediction $\hat{y}$ is a direct function of them, therefore using labeled information to tune the encoder.

One of the objectives of this thesis is to verify if this training procedure would improve results in semi-supervised learning tasks due to the latent space being formed by two sources of information. In one hand, the unsupervised learned characteristics from the unlabeled data, and in the other hand, the information coming from the labels. This would would guide the transformation $\mathbf{x} \rightarrow \mathbf{z}$, produced by the encoder, towards better predictions for the

downstream tasks.

From here onwards, this proposed model will be refer as the tVAE-SSL model. Additionally, using the notation presented in section 4, the tVAE-SSL model have three variants depending on the temporal structures within the encoder and decoder of the tVAE: tVAE1-SSL, tVAE2-SSL, tVAE3-SSL, for the temporal independent dependence, past and present dependence and full temporal range dependence respectively. The predictor neural network, however, remains the same for these three variants for simplicity.

To research the benefit of using inverse normalizing flows to improve the latent space, a modification of the proposed approach with inverse-auto-regressive flows in the tVAE model is designed. As shown in figure 5.2, this modification only changes the inner functioning of the forward pass of the tVAE, not affecting the aforementioned steps of the proposed approach. This modified model will be denoted from here onwards as the IAFtVAE-SSL model. Following the same explanation as for the tVAE-SSL model, three variants are possible depending on the temporal structure of the encoder and decoder, denoted as IAFtVAE1-SSL, IAFtVAE2-SSL and IAFtVAE3-SSL.

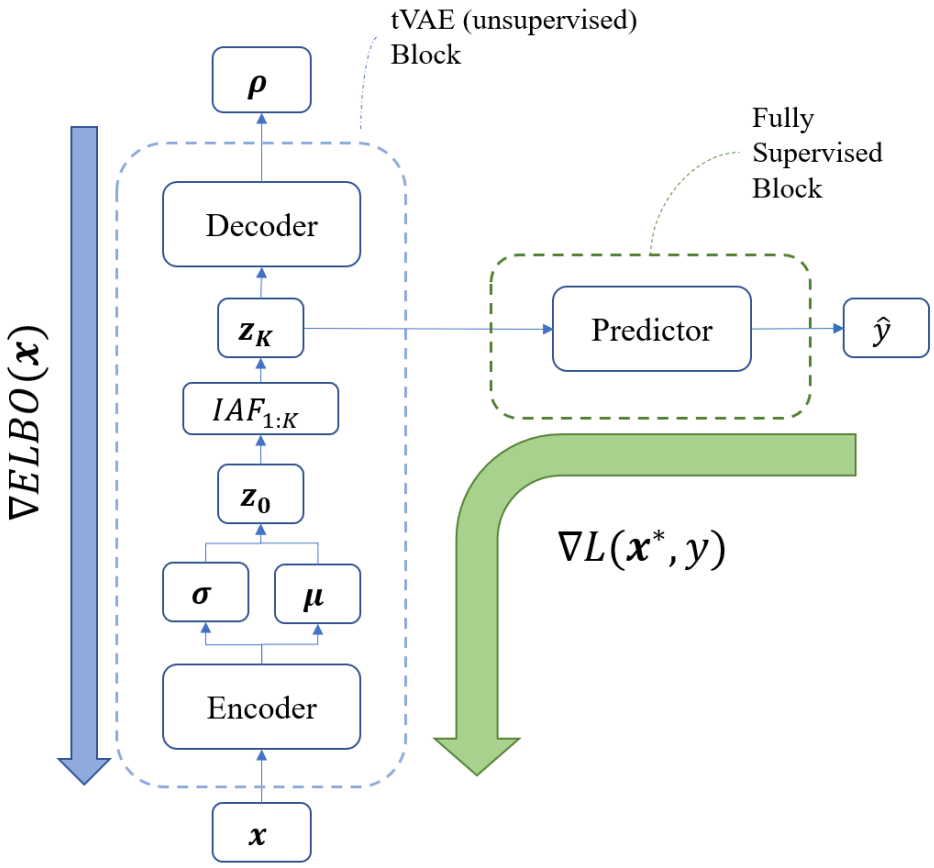


Figure 5.2: The modified IAFtVAE-SSL model.

The coupled training strategy presented in this chapter is inspired in the work of Socher *et al.* [34], who uses a similar approach for traditional Auto-Encoders in the context of predicting sentiment attributes in text. This thesis uses a similar concept, but applied to temporal Variational Auto-Encoders (tVAEs) in the context of PHM for diagnosis and prognosis.

Next chapter defines the experiments performed fulfill the objectives of this thesis.

Chapter 6

Experimental Setting

This chapter describe the general experimental setting used to fulfill the objectives of this thesis. First, with the purpose of comparing the proposed model with current techniques for semi-supervised learning, two baseline models are presented. Then, this chapter focuses on describing the experiment that will be executed for each proposed and baseline model in order to draw comparisons. Finally, the specific architectures for all the models are specified, along with the parameters used for the training of the models.

6.1 Baseline Models

In this section two baseline models are presented. In order to assure a fair comparison in which the novel aspects of the proposed model are evaluated, the baseline models will maintain the exact same architectures and inner structures for the encoder, decoder and predictor, changing other characteristics such as the total loss function or the training strategy. The baseline models are described below.

6.1.1 Baseline 1: Ablation Model

This model represent an ablation study of the proposed model in which the ELBO is removed from the total cost function to verify if learning a latent space using an approximate posterior report any benefits over using a simpler optimization. The elimination of the ELBO function is done setting the hyperparameter α, described in section 5, to zero. This transform the total cost function of the proposed model to:

$$C(\mathbf{x}, \mathbf{x}^*, y) = 0 \cdot ELBO(\mathbf{x}) + L(\mathbf{x}^*, y) \rightarrow C(\mathbf{x}^*, y) = L(\mathbf{x}^*, y) \tag{6.1}$$

This baseline model has the following characteristics:

- The latent space and decoder are not used, since the gradients of L with respecto to them is zero.
- The encoder's parameters are only trained based on the gradients of L.

- Since the decoder is inactive, the model is not able to learn in an unsupervised manner. This can be seen in equation 6.1, where the dependency of C has been updated to show that only labeled data is used to compute the total cost of the model.

As mentioned before, the objective of this baseline model is to verify if learning the latent space as the product of the approximate posterior distribution reports any benefit versus a much more simpler training process. In fact, as it can be seen from figure 6.1 (which shows only the active parts of the original proposal), this ablation study is for all practical purposes, a simple recurrent neural network where the encoder and the predictor are separate blocks of neural networks working in series.

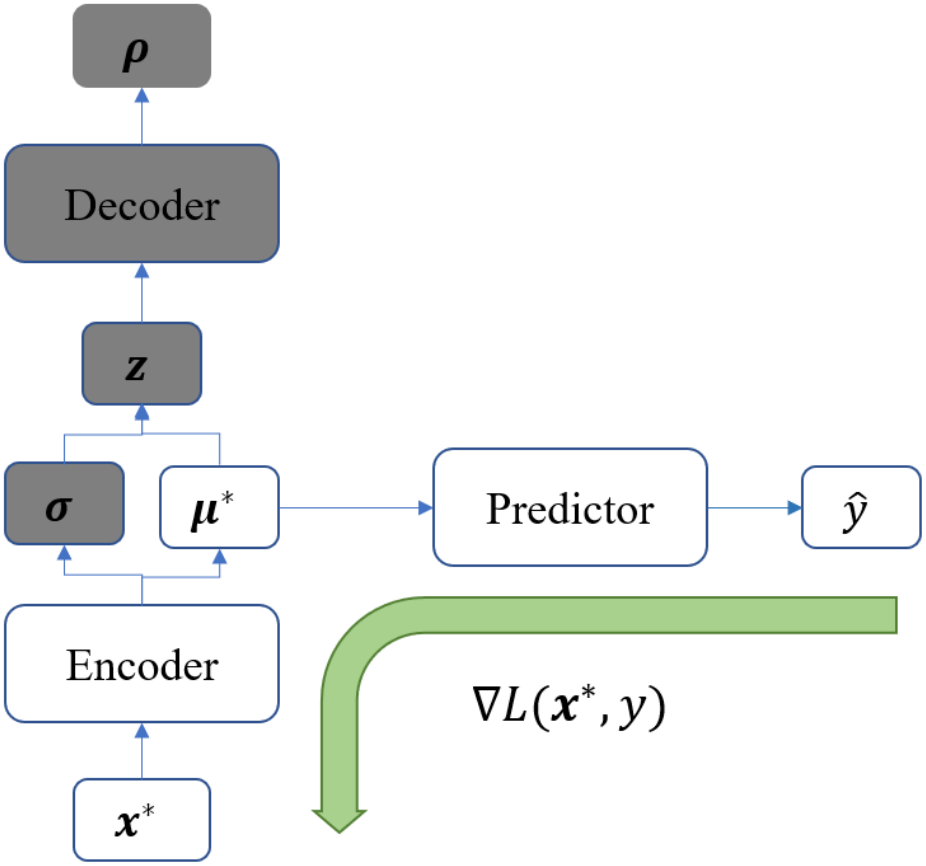


Figure 6.1: Ablation model's diagram. Grey blocks indicate the inactive parts of the original model due to the elimination of the ELBO function from the total loss function C.

From here onwards, this model will be refer to with the identification tag "Ablation".

6.1.2 Baseline 2: Decoupled Model

In this baseline model, the tVAE is used only as an unsupervised feature extractor, capable of processing and transforming the input data without labels. To train this model, a decoupled learning procedure is used, summarized in the following steps:

1. The tVAE is trained first using the full dataset $\mathbf{x}$, without labels, in an unsupervised manner.

2. Once the VAE is trained, its encoder is used to transform the labeled data to the latent space: $\mathbf{x}^* \rightarrow \mathbf{z}^*$.
3. Finally, this labeled portion of the original data, $\mathbf{z}^*$ is used as input for a predictor recurrent neural network.

Figure 6.2 shows a diagram of this baseline model. Note how the two parts of the proposed approach, the tVAE and the predictor, are now trained separately based only on their own loss functions.

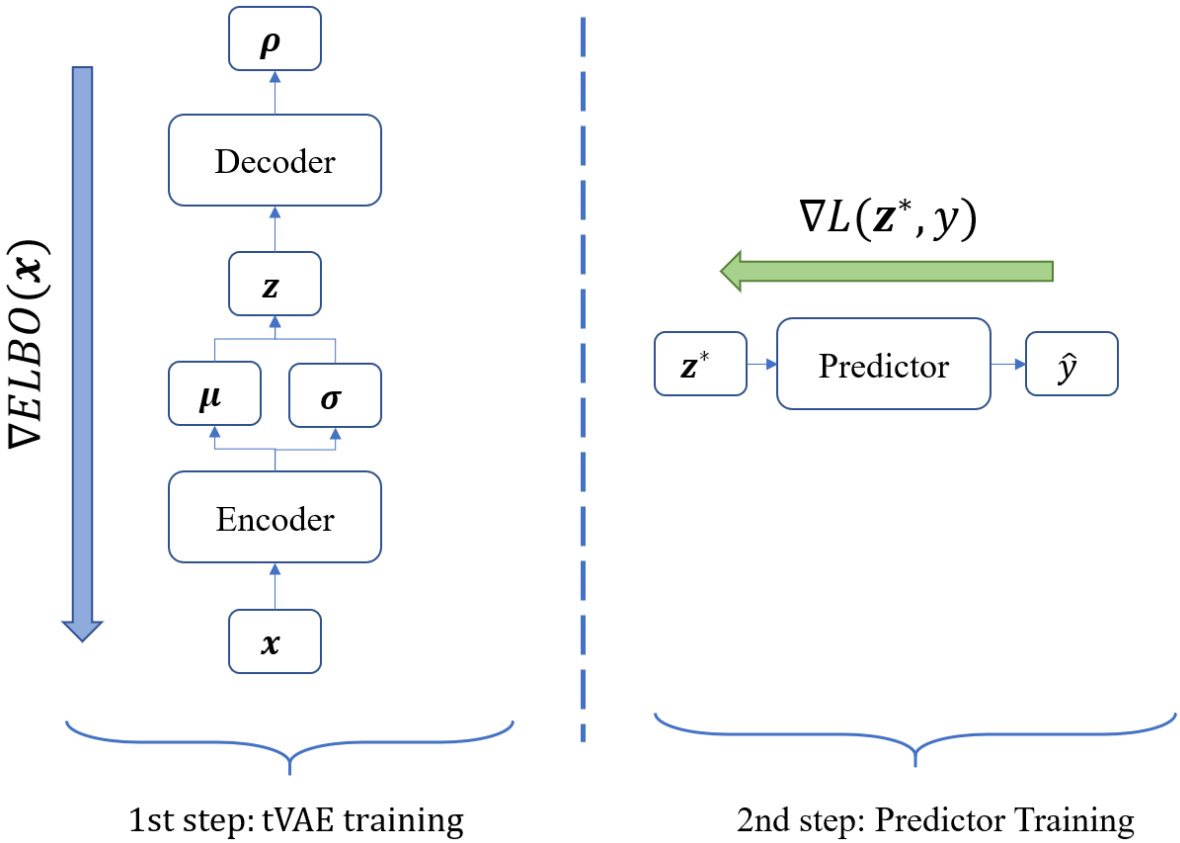


Figure 6.2: Decoupled model's diagram. The tVAE and predictor structures are now trained in a decoupled manner.

The objective of this baseline model is to check if the conjunction training, i.e., the mixed flow of gradients in the encoder, is effectively helping to make a better latent space for downstream purposes.

From here onwards, this model will be refer to with the identification tag "Decoupled"

Temporal dependencies of the Baseline Models

This section presented two baseline alternatives that modified the proposed approach in either their training procedure or cost function. Nevertheless, the predictor, encoder and decoder neural networks were kept the same. Therefore, the three temporal structures presented in section 4 for the encoder and decoder of the tVAE can also be used in all three baselines.

This is done to assure a comparison as fair as possible between the proposed approach and the baseline models.

Table 6.1 summarize all the models used in this thesis. Remember that the following codes are used to identify the temporal structures:

- **Variant 1:** Temporal independent dependence encoder and decoder (see section 4.1)
- **Variant 2:** Past and present dependent encoder and decoder (see section 4.2)
- **Variant 3:** Full temporal dependent encoder and decoder (see section 4.3)

Table 6.1: Summary of the models (proposed and baseline) used in this thesis

Model ID	**Description**
tVAE{1,2,3}-SSL	Proposed model that uses tVAEs and a neural network predictor trained in a conjunct manner to perform semi-supervised learning.
IAFtVAE{1,2,3}-SSL	Modification of the proposed model that adds flexibility to the latent space using inverse auto-regressive flows, prior to the use of the latent variables for the downstream task.
Ablation{1,2,3}	Ablation study of the proposed model that remove the ELBO function for the total cost of the model.
Decoupled{1,2,3}	Change in the training strategy of the proposed model to use a decoupled strategy.

6.2 Description of the Experiment

To avoid confussion, the differences between experiments, case studies and models will be explained prior to continuing with the rest of the thesis. In this context, a model is a particular deep learning structure that produces a classification or regression result, wheter an experiment is the application of that structure to a specific set of parameters or conditions. Lastly, a case study is a set of experiments grouped by the data on which they are performed.

As mentioned before, for this thesis there are five possible models (a proposed model, a modification of such proposal and two baselines) with three temporal variants each that need to be tested in order to draw conclusions. These models were presented in table 6.1 along with their IDs.

This thesis will present two case studies, which are based on a dataset with an specific objective. Each case study uses a different dataset. Regrettably, the datasets are design to assure that every data point $\mathbf{x}$ have a corresponding label. To simulate the semi-supervised scenario, only a percentage of the labels will be considered in the experiments. The portion of labels considerer will be selected at random and controlled by a parameter p that can take one of the following values: [1%, 2%, 5%, 10%, 20%, 35%, 50%, 75%, 100%]. The situation where $p = 100\%$ represent a fully supervised training case, where all the available data points $\mathbf{x}$ have labels.

The experiment starts with one of these datasets, which are already preprocessed and divided into a training set and testing set, and ready to be used. The preprocessing procedure

for each case study will be explained in the corresponding chapter. The following steps describe an experiment run for this thesis.

1. Divide the training set into a training and validation subsets in a $80\% - 20\%$ ratio.
2. Only for the remaining training set, chose a percentage of survival labels l_p from the aforementioned list and randomly delete $(100 - p)\%$ of the training labels. This will produce a subset of the training dataset that is labeled.
3. Choose a model from table 6.1 with a temporal dependency from section 4 and use the training dataset to train the model. Every time the validation cost is improved, the parameters of the model are saved. Once the number of epochs reach 500 or the validation cost does not improve for 25 consecutive epochs, finish the training cycle and restore the model that achieved the best validation cost.
4. Using the restored encoder and predictor at the end of the training cycle and the test set, compute the required metrics of the downstream task.

For each model of table 6.1 and value p in the aforementioned list, the experiment are run 10 times to obtain the mean and standard deviation of the desired metrics.

The following section describe the architectures used for the common encoder and decoder of the models presented in table 6.1.

6.3 Model's Neural network Architectures

As mentioned in section 6.1, the baseline and proposed models share the same inner structures alternatives for the encoder and decoder neural networks. Figures 6.3 and 6.4 present the encoder and decoder architectures, respectively.

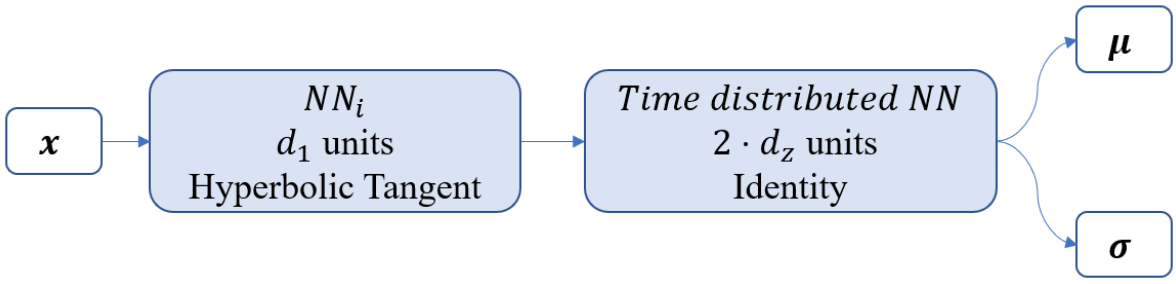


Figure 6.3: Encoder architecture for all the models except the Auto-Encoder baseline. NN_{i} can refer to a time distributed neural network, an LSTM neural network or a Bi-LSTM network depending on the desired temporal structure.

Depending on the desired temporal structure of the model (see section 4), the first computation block of the encoder and decoder, denoted as NN_{i}, is replaced for an specific neural network structure. For the temporal independent version of the models, NN_{i} is a time distributed traditional neural network of d_1 units or neurons. For the past and present dependent version NN_{i} is a LSTM network with d_1 units that return the full sequence of outputs states y_{i}. Finally, for the full temporal range dependant version, NN_{i} is a Bi-LSTM network with d_1 units that also return all the output states. The second block in the encoder and decoder

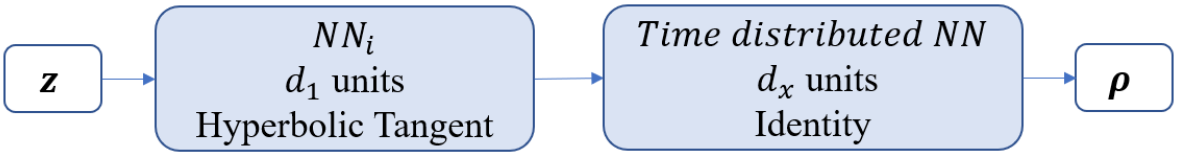


Figure 6.4: Decoder architecture for all the models. NN_{i} can refer to a time distributed neural network, an LSTM neural network or a Bi-LSTM network depending on the desired temporal structure of the decoder.

models correspond to a time distributed traditional neural network and is used to add an extra degree of flexibility to the models without changing the temporal dependence imposed by NN_{i}.

For all the experiments and case studies presented in this thesis, d_1 is set to 32. In preliminar experiments it was found that this number of units worked well for the diagnosis and prognosis study cases while also producing a relatively light network in terms of computation time. In the decoder model, d_x represent the number of features of each vector within the sequence input data, and therefore it changes for each case study. The dimensionality reduction of input data is given by the number of features of the latent space, d_z. Various degrees of dimensionality reduction were tested in preliminar experiments, but to avoid adding an extra degree of freedom to the modelling it was fixed as $\text{d}_z = \text{d}_x/2$, as this value was found to give good results while preventing the encoder to learn a simple, uninformative encoding function (which can happen when d_x is very similar to d_z). The final block of the encoder outputs $2 \cdot \text{d}_z$ features since the first half is interpreted as the mean of the latent variables and the second half as the standard deviation.

Figure 6.5 present the predictor architecture, which will be the same for every model and variant. The activation function of the last layer, f_{act}, will depend on the particular downstream task. For diagnosis and prognosis problems, as mentioned in section 3.2.3, the last activation function is a softmax function and an identity function, respectively.

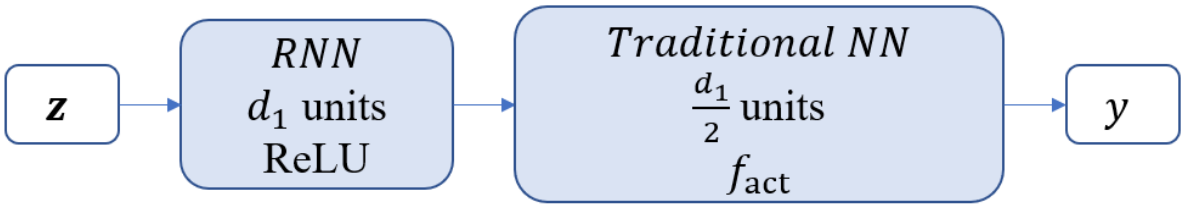


Figure 6.5: Predictor architecture for all models. The activation function of the last layer, f_{act}, depends on the downstream task.

Finally, the inverse auto-regressive flows, for those models that use it, are composed by three independent flow blocks in which each block is composed by one MADE network of 16 neurons. This configuration was chosen based on preliminar experiments, which shown that heavier structures resulted in models that were too slow for the purposes of this thesis.

Chapter 7

Case Study 1: CMAPSS dataset for Turbofan Engines Remaining Useful Life Estimation

The first case study presented in this thesis is a prognosis example, corresponding to the estimation of remaining useful life in turbofan engines. The data for this case study comes from the well-known CMAPSS dataset, originated for the PHM 08' data competition [9] and used since then as a benchmark dataset for PHM purposes. This chapter starts with the pre-processing procedure for this dataset.

7.1 Data Preparation

The CMAPSS dataset is composed of multivariate operational time-series obtained by simulating a turbofan under various fault modes and operational conditions using the *Commercial Aero Propulsion System Simulation* [9] (hence the initials CMAPSS). The dataset is organized into four categories depending on the number of fault modes and operational conditions that are present in the corresponding simulations. Table 7.1 present the characteristics of these categories.

As it can be seen from table 7.1, FD004 is the most complex category since it present multiple operational conditions and multiple fault modes. In this thesis, only results regarding this subset will be presented.

Table 7.1: CMAPSS four categories features.

Subset	Operational Conditions	Fault Mode
FD001	One	One (HPC Degradation)
FD002	Six	One (HPC Degradation)
FD003	One	Two (HPC and Fan Degradation)
FD004	Six	Two (HPC and Fan Degradation)

Table 7.2: FD004 sensors. LPC: Low Pressure Compressor. HPC: High Pressure Compressor. LPT: Low Pressure Turbine. HPT: High Pressure Turbine.

Sensor	Description	Unit
1	Total temperature at fan inlet	°R
2	Total temperature at LPC outlet	°R
3	Total temperature at HPC outlet	°R
4	Total temperature at LPT outlet	°R
5	Pressure at fan inlet	psia
6	Total pressure in bypass-duct	psia
7	Total pressure in HPC outlet	psia
8	Physical fan speed	rpm
9	Physical core speed	rpm
10	Engine pressure ratio (P50/P2)	-
11	Static pressure at HPC outlet	psia
12	Ratio of fuel flow to Ps30	peps/psi
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass ratio	-
16	Burner fuel-air ratio	-
17	Bleed Enthalpy	-
18	Demanded fan speed	rpm
19	Demanded corrected fan speed	rpm
20	HPT coolant bleed	lbm/s
21	LPT coolant bleed	lbm/s

Each one of the train and test trajectories are flight simulations in the form of a table, in which each row represents one time-step measured in cycles and each column is a different sensor measurements. FD004 simulate the output of 21 sensors, listed in table 7.2. Both training and testing trajectories are simulated starting from a normal operational state with an unknown level of degradation. While the train trajectories are simulated until the engine present one of the corresponding fault modes, the testing simulations are stopped at a random moment before that. The objective of original competition was to train models capable of predicting the number of cycles until failure (RUL) of these test trajectories. Nowadays the RUL for the test examples have been made available to the public with the rest of the data.

One important aspect is that each simulated engine from the FD004 subset is considered to come from a fleet of engines of identical characteristics. Therefore, the full dataset can be considered as a series of run-to-failure tests executed on the same engine. This allow the use of all the training trajectories to learn the same model.

As mentioned before, each training and testing trajectory can be defined as a two dimensional table $M \in \mathbb{R}^{L \times F}$, where L is the length in cycles of the trajectory and F is the number of available sensors (21 in this case). The first step towards generating the final dataset is to verify if every sensor contain information that changes with time. For this, a visual inspection is performed, resulting in the identification of sensors 10, 15 and 16 as static in-

formation, eliminating them from the analysis. A second step is to normalize the data. This normalization is done for each sensor separately, using a min-max approach, restricting each column to the $[0, 1]$ range as required for the tVAE models. The final step is to increase the number of training data-points and standardize the length of them, since the proposed approach is not design to support inputs of variable length. One common strategy for this is to generate moving windows of length T time-steps, which slides over each trajectory with an overlap v. In this tesis, after some preliminary experiments, the length of the time-steps and the overlap were set to $T = 19$ and $v = 18$, respectively. For the testing set, a similar process is done with the exception that only the last window of the incomplete series is kept.

In the FD004, only the label (RUL) of the testing trajectories is dispatched with the dataset. For the training trajectories, the label for each resulting window is obtained as the remaining operating cycles until the failure counting from the last time step within the window. The sliding mechanism and the label generation process for the training trajectories are presented in figure 7.1.

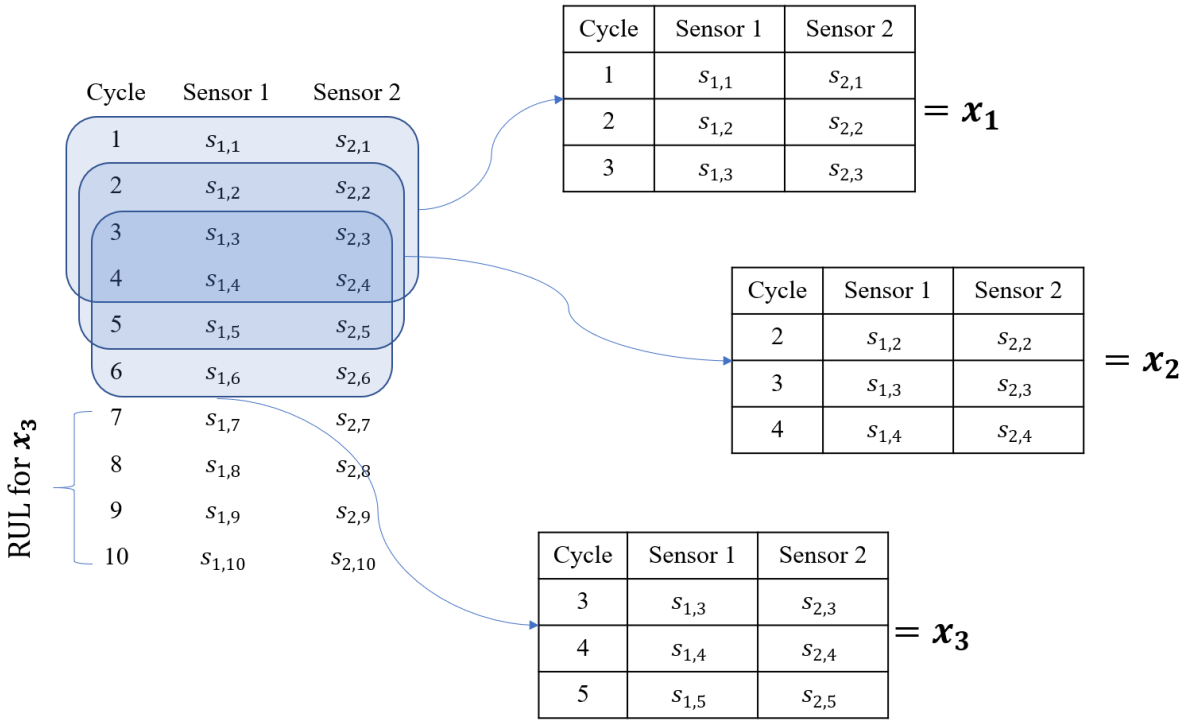


Figure 7.1: Window sliding mechanism to generate the data-points. For this toy example, the length of the window is $T = 3$ and the overlap is $v = 2$. The RUL computation is only show for the $\mathbf{x}_3$ window, which correspond to four cycles.

The resulting training and testing datasets consist on matrices $\mathbf{x}_\text{i} \in \mathbb{R}^{T \times D}$ with $T = 19$ and $D = 18$, which are the remaining valid sensors. Each data point $\mathbf{x}_\text{i}$ has a corresponding label y_i that represent the remaining useful life for that operational window. With this process, the final number of training samples and testing samples is $45,015$ and 248 respectively.

Finally, an additional step is performed on the labels y_i. For this thesis it will be assumed that the degradation of a component cannot be easily detected until a certain threshold in the remaining useful life has been reached. This threshold will be set to $R_{early} = 130$ cycles,

as proposed in [14]. In practice, this means that every RUL value that meet the condition $y_{\text{i}} > R_{early}$ will be manually changed to $y_{\text{i}} = R_{early}$.

The following section presents the results obtained for this case study.

7.2 Results and Discussion for the FD004 Dataset

This section review the results obtained from executing the experiments described in section 6.2 using the proposed and baseline models for the RUL prediction in the FD004 dataset. It will be divided in sub-sections that aim to extract conclusions about particular components of the proposed approach, by either comparing it the to baseline models or between it's different time variants. All the results shown in this section are obtained using the testing dataset, unless state otherwise.

7.2.1 Assessing the different time-structures of the encoder and decoder

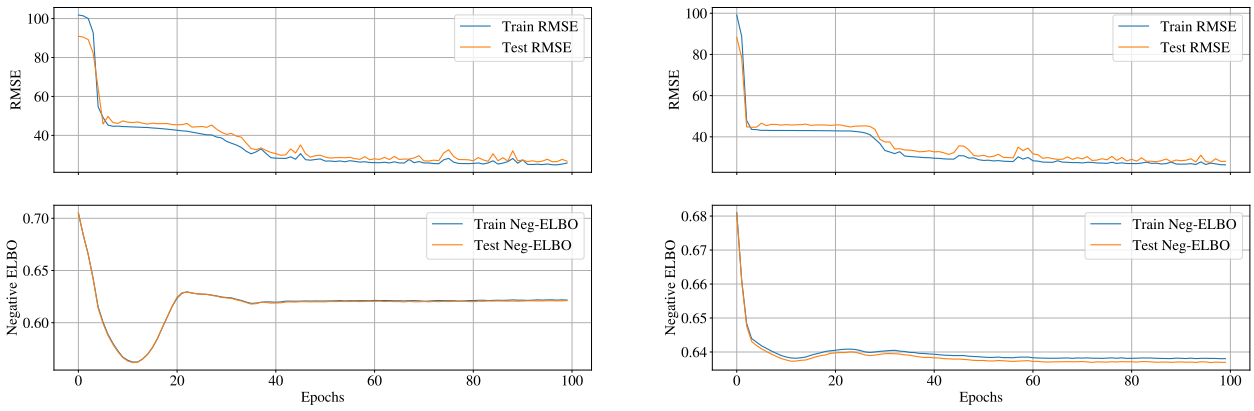
As presented in section 4, the proposed model is designed with three variants for the encoder and decoder. They are summarized in table 7.3.

Table 7.3: Temporal variants for the proposed model. These are also presented in the baseline models and the IAF modification.

Name	TAG	Encoder	Decoder
Variant 1	tVAE1-SSL	$\vec{z}_t \sim q(\vec{z}_t \vert \vec{x}_t)$	$\vec{x}_t \sim p(\vec{x}_t \vert \vec{z}_t)$
Variant 2	tVAE2-SSL	$\vec{z}_t \sim q(\vec{z}_t \vert \vec{x}_{1:t})$	$\vec{x}_t \sim p(\vec{x}_t \vert \vec{z}_{1:t})$
Variant 3	tVAE3-SSL	$\vec{z}_t \sim q(\vec{z}_t \vert \vec{x}_{1:T})$	$\vec{x}_t \sim p(\vec{x}_t \vert \vec{z}_{1:T})$

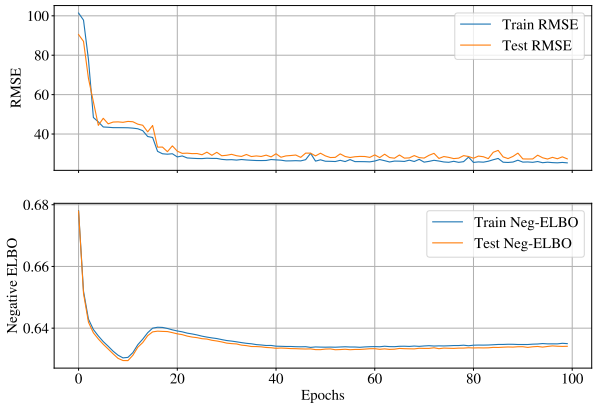
First, the evolution of the cost functions for the predictor (RMSE) and tVAE (ELBO) during training is presented in figures 7.2. All the figures contained the evolution of the curves for both the training and testing dataset, to check if the training process developed some type of overfitting. While the number of epochs was higher than 100 for all the models, the curves are only shown until that point since after that the improvements over the costs are very minimal. Also, the curves are obtained using $p = 50\%$ to show a representative semi-supervised case. Other cases were tested but the curves do not changed considerably in a way that affects the analysis presented below.

As figure 7.2 shows, there is no overfitting in the training process of the proposed model for any of the temporal variants. The RMSE cost show in all cases slightly better results for the training curve, which is expected and normal. The ELBO (or negative ELBO, since the optimization is perform though a minimization) cost show very similar curves for both the training and testing dataset, which is also expected for two main reasons: first, both subsets came from a simulated scenario, in which all the variables are controlled and therefore the engines behaves in a very similar way, and second, the ELBO do not depend on the labels, consequently is not affected by the lack of generalization over the predictions of the RUL.



(a) Temporal independent structure

(b) Past and present dependent structure



(c) Full temporal range dependent structure

Figure 7.2: Evolution of the RMSE and ELBO costs of the proposed model for the train and test dataset during training.

Another interesting aspect of figure 7.2 is that for the three variants, both the ELBO and the RMSE cost functions improve over the full training process, showing that the proposed model is learning how to do two functions simultaneously: generate or reconstruct the input sequences through the tVAE and predict the remaining useful life of the engines through the predictor network. Nevertheless, both error curves (RMSE and negative ELBO) are not close to zero, so there is still room for improvement in the prediction and reconstruction abilities of the model.

A final commentary with respect to the training curves is that in the the beginning of the process, the ELBO improves very rapidly, but towards the middle section of the training, its value gets worse to allow for the improvement of the second part of the total cost function, the RMSE, which coincidentally at the same time starts improving again. One possible explanation for this is that, due to the coupled training process that the proposed model uses, the latent space is being *pulled* towards a reconstructive objective and then towards a predictive objective to finally settle in an equilibrium that allow the realization of both functions.

The effect of the different temporal variants over the RMSE of the remaining useful life

for the proposed model is presented in figure 7.3.

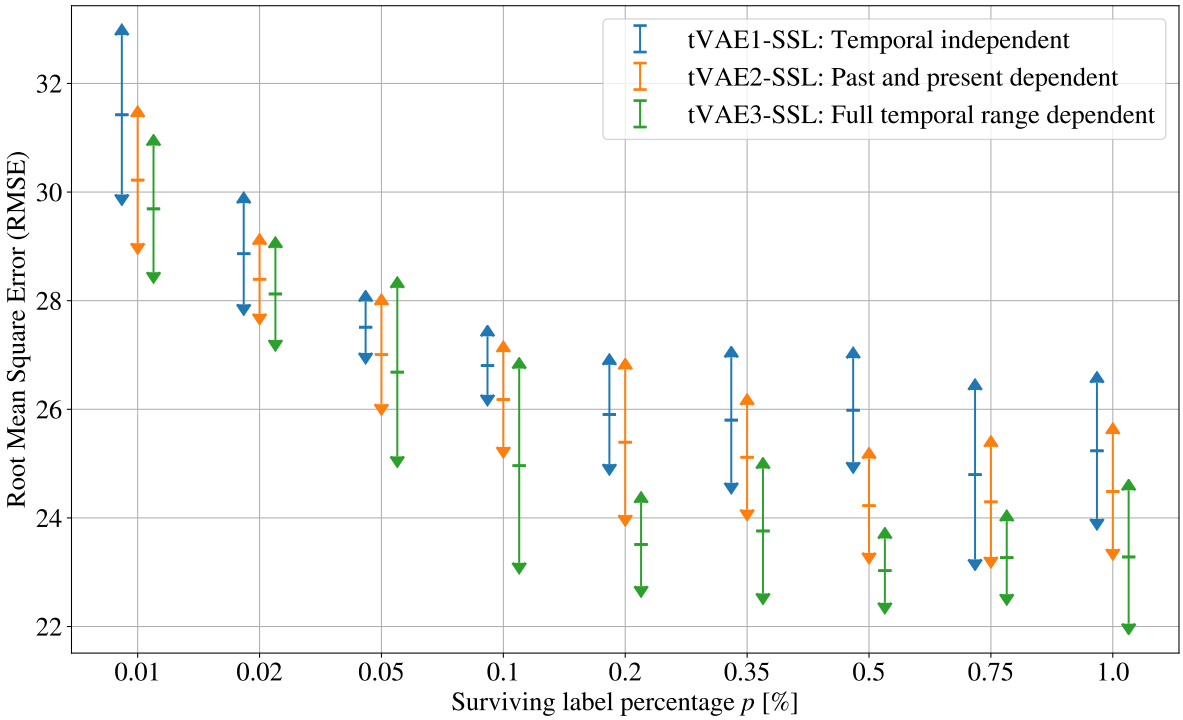


Figure 7.3: RMSE score for the different temporal structures of the proposed model.

The first important result that can be extracted from figure 7.3 is that for every level of labeled information p, better prediction scores are achieved when the encoder complexity increases. This can be seen clearly by the better RMSE achieved by the green case versus the blue and orange cases. The only option for this is that a better latent representation is being generated, since the predictor network is the same for all the variants. This clearly indicates that the dataset used in this case study contains features that are highly dependant on time, and therefore the regression task will benefit from models capable of processing temporal information better.

Figure 7.3 also shows that an increase in prediction performance happens for all variants of the proposed model when the amount of labeled information is increased. This is a very expected behavior since generalization depends in great measure on having access to a wide variety of examples. However, an interesting aspect is that variant 3 reach an stabilization point for the RMSE score sooner than the other variants, at around $p = 0.2$, in which more labels do not contribute new information to improve the results further. The same effect, in a lesser degree, is observed when comparing variant 2 to variant 1. This evidence that allowing the encoder to access different time states to infer the latent sequence enables the model to achieve better degrees of generalization using lower levels of labeled information.

Finally, table 7.4 present the overall improvement of prediction performance as the amount of labeled information increases ($p = 0.01 \rightarrow 1$) for the three temporal variants of the proposed model. As it can be seen, the difference between the three variants in terms of this overall improvements are negligible if the standard deviation of the results is considered. Therefore, while a more complex inner structure within the tVAE effectively results in better

RMSE scores, the rate at which models improve with more available labeled information is very similar.

Table 7.4: Performance improvement as the amount of labeled information increases from $p = 0.01$ to $p = 1$

Temporal Variant	RMSE improvement
tVAE1-SSL	6.18
tVAE2-SSL	5.73
tVAE3-SSL	6.40

From these results is clear that better downstream tasks performances in both semi-supervised ($p < 1$) and fully-supervised ($p = 1$) tasks can be achieved using more temporally-aware inner structures for the tVAE in the proposed model. What is yet to be discovered is if these results can be improved further by adding flexibility to the latent space, which in the tVAE-SSL model is parameterized by a diagonal Gaussian distribution. The following section analize the results obtained from adding a normalize flow to the proposed model in order to answer this question.

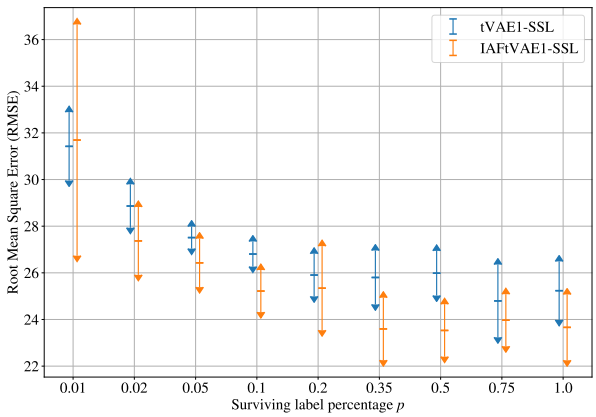
7.2.2 Assessing the improvements achieved by using Inverse Auto-Regressive Flows

The following results, presented in figure 7.4, compare the proposed approach (tVAE-SSL) with the modified model that uses an inverse auto-regressive flow to transform the sequential latent space (IAFtVAE-SSL).

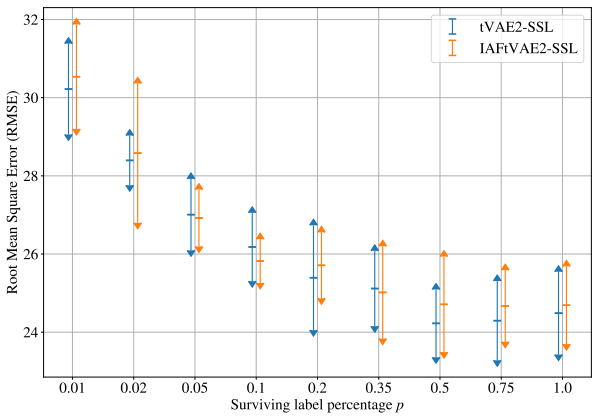
These results show that only for the temporal independent case (figure 7.4a) the RMSE score improves with the use of inverse auto-regressive flows to transform the latent space. For temporal dependent structures, portrayed in figures 7.4b and 7.4c, these improvements banish, removing the utility of using normalizing flows in variants 2 and 3. One possible explanation for this behavior is that the inverse auto-regressive flow effectively improve the quality of the latent variables, but only to a certain extent. Therefore, this improvement is noticeable only in simpler structures, such as variant 1, but overshadowed by more capable encoders that can process temporal data and build a latent space using past and/or future states, represented by variants 2 and 3. Nevertheless, note that the Y-axis does not start in zero, so there is still room for improvement over this models in terms of their predicting capabilities.

This hypothesis is enforced by the reconstruction metrics shown in table 7.5, where both the log-probability and RMSE of the reconstructed input data $\hat{\mathbf{x}}$ generated by the tVAE (not to be confused with the RMSE of the RUL) are considerably lower only for the IAFtVAE-SSL model that uses the variant 1, which is the simpler one. In variants 2 and 3, the metrics are lower, but by a very slight margin. This evidence that inverse auto-regressive flows improvements are only noticeable in very simple models, where flexibility is not already added by the existent structures.

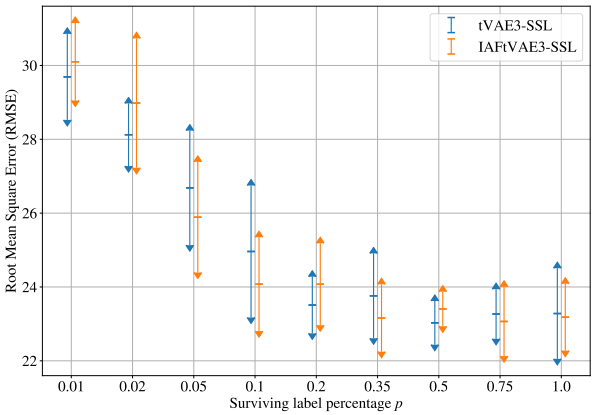
Note that since this reconstruction measures come from the unsupervised part of the



(a) Temporal independent structure



(b) Past and present dependent structure



(c) Full temporal range dependent structure

Figure 7.4: RMSE results between the proposed model and the modification that incorporates the inverse-auto-regressive block.

model, they are the same for each value of p, and therefore the average value is presented in table 7.5.

Therefore, from this experiment is clear that for this case study, normalizing flows improve the latent space only in cases where the generation is conditioned on insufficient information, such as in variant 1. For that case, the semi-supervised learning capabilities of the proposed model are enhanced in almost the full range of labeled information ($p \in [0.02, 1]$). For models where the encoder and decoder are more complex, such as in variants 2 and 3, their contributions are negligible and consequently, their use is unjustified.

Next section is targeted towards verifying if the coupled training process holds any improvement over the traditional approach of optimizing the tVAE and the predictor separately.

7.2.3 Assessing the improvements achieved by the coupled training process

The following results, presented in figure 7.5, compare the proposed approach (tVAE-SSL) with the decoupled baseline model (Decoupled-SSL) that uses a disjunct training process to

Table 7.5: Reconstruction metrics for the tVAE model with and without a IAF block for the three temporal variants.

	tVAE-SSL		**IAFtVAE-SSL**	
	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$
Variant 1	-0.605	0.290	-0.390	0.048
Variant 2	-0.634	0.311	-0.615	0.298
Variant 3	-0.629	0.308	-0.613	0.296

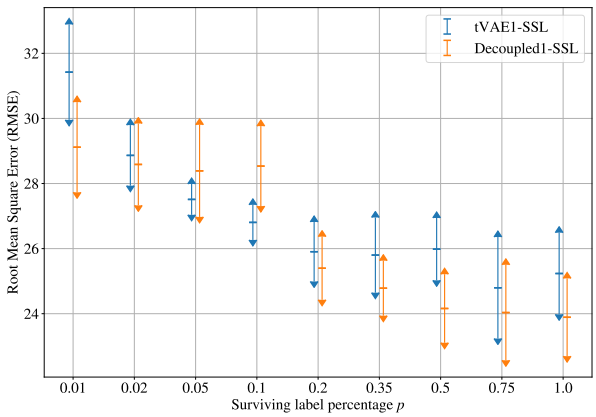
optimize the tVAE and the predictor network.

For the temporal independent variant, presented in subfigure 7.5a, the decoupled model presents slightly better results by a margin of 1% when a high number of labels is available ($p \geq 50$). Nevertheless, these improvements are very minor and often fall within the error region given by the standard deviations. In contrast, for the temporal variant 2, portrayed in figure 7.5b, significant improvements in the RMSE score of the predictions are achieved when using a coupled training strategy. The same situation repeats, with even bigger differences in performance, for the temporal variant 3, shown in figure 7.5c. This means that the use of labeled information to tune the encoder of the tVAE only has a noticeable effect when the encoder itself is able to process temporal information from the input sequence.

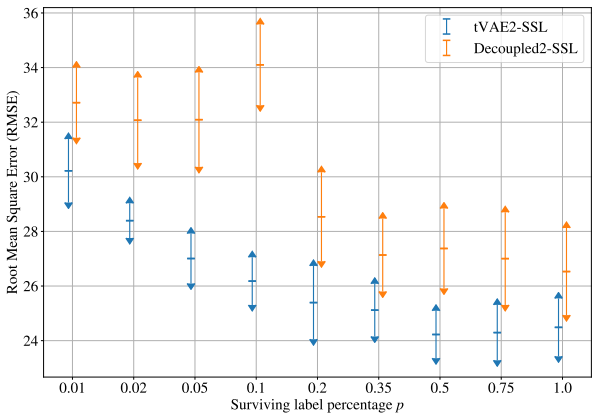
One possible explanation for these observations, in which the coupled strategy is justified only for tVAE models that uses a temporal-aware inner structure, is related to the nature of the data. As mentioned in section 7.2.1, the prediction quality for this dataset is highly dependent on the ability of the model to extract and analyze temporal data. Therefore, it is expected that when the encoder of the tVAE is a LSTM or Bi-LSTM neural network, which is the case for variants 2 and 3 respectively, labeled information contained in the gradients of the predictor's loss function will produce a bigger impact on the transformation capabilities of the encoder.

Another important result that can be derived from this experiment is that for all the tVAE temporal variants, only the coupled training strategy improves its results when p ranges from 0.01 to 0.1, while the decoupled strategy remains almost static. This puts evidence that the gradient's flow from the labels to the encoder network produces better RMSE scores when the labels are scarce, resulting in improved semi-supervised capabilities versus a decoupled strategy.

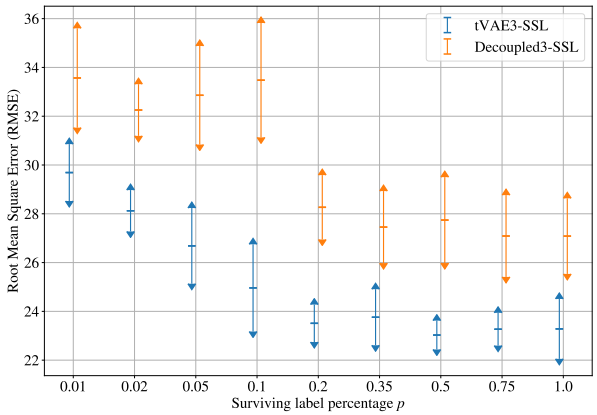
Finally, table 7.6 compare the decoder's reconstruction ability of both the proposed model and the decoupled baseline model for every temporal variant. As a reminder, variant 1 correspond to the temporal independent encoder/decoder, variant 2 correspond to the past and present dependant encoder and decoder and finally, variant 3 correspond to the full time range dependant encoder and decoder. These results evidence that there are not significant differences in the reconstruction capabilities of the tVAE between a coupled and decoupled training strategy. This is an expected behavior since the decoder network is not directly affected by the flow of labeled information and therefore the results should be very similar. As a secondary observation, table 7.6 indicates that the inclusion of labeled information for generation purposes via the proposed coupled strategy do not produce any benefit over the



(a) Temporal independent structure



(b) Past and present dependent structure



(c) Full temporal range dependent structure

Figure 7.5: RMSE results between the proposed model and decoupled baseline.

traditional approach of only using unsupervised information.

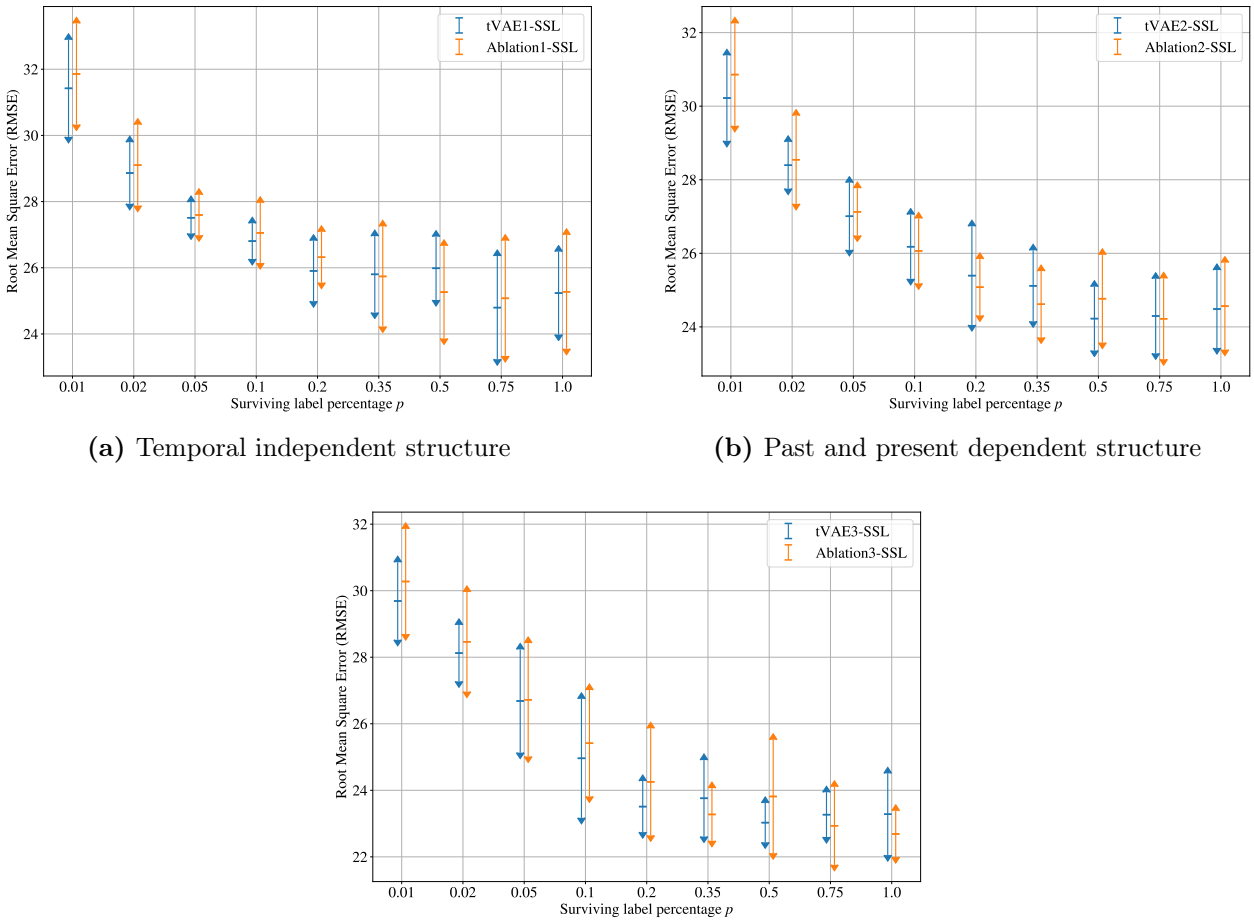
Table 7.6: Reconstruction metrics for the proposed model and the decoupled baseline.

	tVAE-SSL		**Decoupled-SSL**	
	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$
Variant 1	-0.605	0.290	-0.639	0.315
Variant 2	-0.634	0.311	-0.637	0.316
Variant 3	-0.629	0.308	-0.638	0.314

Therefore, this experiment show that the improvements obtained by switching the training strategy from a decoupled process to a coupled one are exclusively observed in the downstream task performance, while the generation (i.e. reconstruction) capabilities of the tVAE are not affected. The final section of this chapter is devoted to discovering why the coupled training purposes improves the regression results. In particular, which one of the sources of information (unsupervised via the ELBO or supervised via the predictor's loss) the encoder is using the most to improve the latent space of the tVAE.

7.2.4 Results from the Ablation Study

The final comparation for this chapter is presented in figure 7.6, where the proposed model is contrasted against the ablation study baseline. The ablation study model, as described in section 6.1.1, is the result of removing from the proposed model any capability of generating the latent space using unsupervised information via the elimination of the ELBO function from the total cost C. This transforms the proposed model into what is, for all practical purposes, a recurrent neural network that only learn from the supervised portion of the data using the predictor's loss function L, ignoring the unlabeled data-points.



(a) Temporal independent structure

(b) Past and present dependent structure

(c) Full temporal range dependent structure

Figure 7.6: RMSE results between the proposed model and Ablation study baseline.

Figure 7.6 shows that, for all levels of labeled information and the three temporal variants, there are no statistically significant improvements by using variational inference to learn unsupervised characteristics in the latent space. The fact that the proposed model do not achieve better results over the ablation study for lower values of p indicates that the coupled training strategy do not improves the regression results by capturing additional information from the labels to the latent space, but instead that **all** the important information to generate predictions is captured from this few labeled examples.

However, as discussed in section 7.2.1, the training curves for the proposed model show evidence that the tVAE is effectively learning how to reconstruct the input sequences with

the decoder, at the same time that learning how to encode the input data in a way that can achieve good results in the regression tasks. This can be seen from both costs being lower at the end of the training process with respect to the start of it. Therefore, one possible conclusion is that while the proposed model is not able to improve upon the results obtained by the ablation study using only labeled data, a dual function latent space is, in fact, being generated. In one hand, this latent space is capable of producing good quality predictions for downstream tasks such as regression, while on the other hand it could also be used for generation purposes. These two functions are performed without any one of them totally disrupting the other. This is in fact a very interesting side result from these experiments, and represent a possible way in which to continuate this research.

Finally, it is important to note that since the Y-axis on the plots show in 7.6 does not start in zero, there is still an important amount of room for improvements in terms of prediction performance in this dataset.

A complete set of the RMSE results for all the models, temporal variants and levels of labeled information is included in the appendix, table 10.1.

The following chapter will present the results for the second case study of this thesis, which is a diagnosis example.

Chapter 8

Case Study 2: Pumping system dataset for Diagnosis of Health State

The second case study presented in this thesis is a diagnosis example, in which the objective is to classify the level of damage present in a pumping system used for offshore oil extraction. The dataset for this case study was kindly provided by a Latin-American oil company, who gave permission to use the data for a variety of experiments, including this thesis, under the condition of not disclosing its name.

This chapter starts with the pre-processing procedure for this dataset.

8.1 Data Preparation

This dataset is delivered in one large csv file, in which the rows and columns represent time-stamps and sensors measures, respectively. The csv file capture the operation of the system in the period comprised from January, 2010 to December, 2012.

The sensors used in the system are listed in table 8.1 with the following keys:

- **VF:** Vibration
- **ZT:** Deviation of the shaft
- **IT:** Electric current
- **PIT:** Pressure
- **Ti, TT, TIT:** Temperature
- **PDIT:** Pressure difference
- **FIT:** Flow rate

The location of the sensors within the system are shown in figure 8.1.

Upon a preliminar inspection of the dataset, three main difficulties are found. First, the sampling rate of the sensors is irregular, averaging one measure every two minutes, approximately. Secondly, for an unknown reason, the sensors are not coordinated between

Table 8.1: Available sensors for the pump system.

#	TAG	Unit	#	TAG	Unit
1	PDIT-301D	kPa	13	TI-310D	°C
2	TI-307D	°C	14	TI-306D	°C
3	TI-308D	°C	15	TI-305D	°C
4	VT-301D	μ mm	16	VT-321D	μ mm
5	VT-302D	μ mm	17	VT-322D	μ mm
6	ZT-301D	mm	18	VT-323D	μ mm
7	ZT-302D	mm	19	PIT-302D	kPa
8	TI-309D	°C	20	PIT-303D	kPa
9	PIT-306D	kPa	21	PIT-304D	kPa
10	PIT-305D	kPa	22	HORIMETRO	h
11	VT-303D	μ mm	23	FIT-323D	m3/h
12	VT-304D	μ mm	24	IT-514001031	A

them. This means that for a given minute, not every sensor captures a measure. Finally, the system present a large number of failures in the data capturing mechanism. This three difficulties result in a dataset with a large number of NaN events. Therefore, one of the first steps of the pre-processing procedure will be the cleaning and consolidation of the data.

Starting from the csv file, the pre-processing steps are as follow:

1. **Cleaning the data:** The csv file is filled with "I/O Error", "I/O TimeOut", "Shut-down" among others string messages for when the sensors were out of service or out of synch between them. Therefore, the first step is to replace all this values with the "NaN" (Not-a-Number) value to indicate that those sensors do not have a valid measures for those time stamps.
2. From the prior step, the dataset is filled with "NaN" values. Therefore, the second step is to find the minimal time interval in which all the sensors had at least one valid measure. This time interval was found equal to one hour. Then, the measures within each hour were averaged to find a representative value for each sensor. This process discards the vibrations measures since their averages over such a long period of time hold no physical meaning.
3. Once the dataset is consolidated, it is column-wise normalized using a min-max scaler between 0 and 1.
4. To generate sequential data-points, a similar window-sliding process described in section 7.1 is applied using a window length of 24 time-steps. This produces data-points $\mathbf{x} \in \mathbb{R}^{24\times17}$ that represent a 1 day period operation measured by the 17 remaining sensors.

For the labels, every time-stamp within the original csv file was kindly labeled by the engineers of the company, using expert knowledge and field inspections. This label consist in the current health state of the pumping system, categorized in three possible classes:

- Normal operation, where no failure or damage is detected. This state is assigned to

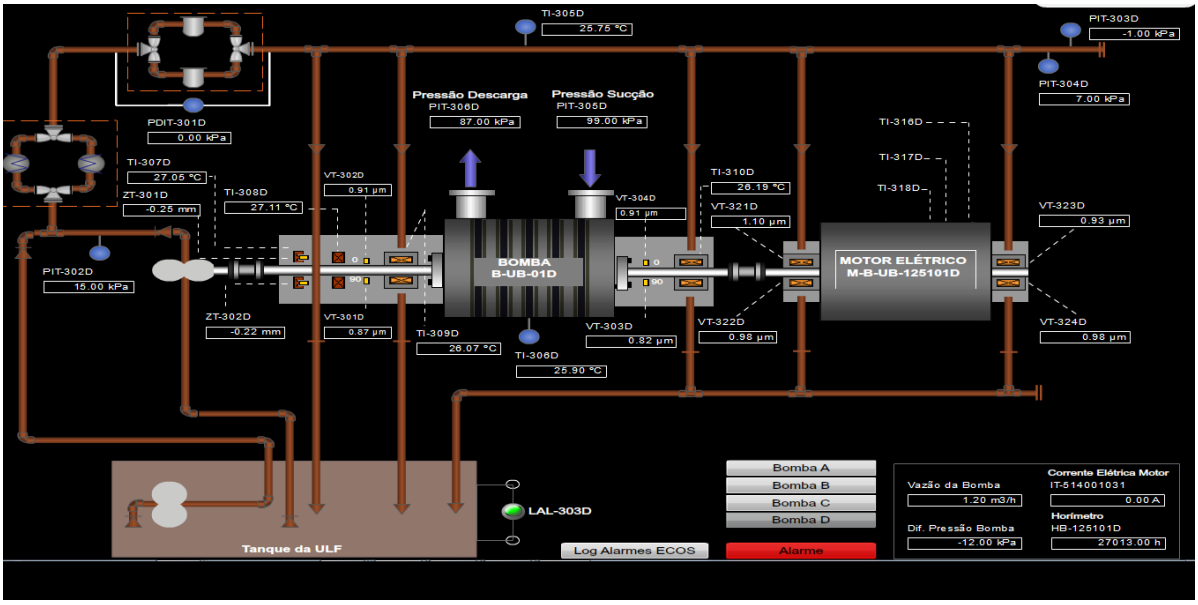


Figure 8.1: PID of the pump system, indicating the location of all the sensors listed in table 8.1.

class 0.

- Incipient damage, where a low, but increasing damage state is detected. This state is assigned to class 1.
- Severe damage, where a high damage state that can cause the sudden stop of the operation is detected. This state is assigned to class 2.

This dataset is heavily unbalanced, which means that the number of examples for each class varies widely. Table 8.2 shows the number of examples per class for the entire dataset. As it can be seen, the number of examples for the healthy dataset are larger than for the other two classes.

Table 8.2: Number of examples per class for the pumping dataset.

Normal Operation	Incipient Damage	Severe Damage
10926	1644	419

Due to this imbalance, the accuracy metric for this dataset is replaced by the *balanced accuracy* [35], in which data points that belong to a less frequent class have a heavier weight in the final value of the metric, thus producing a more fair and honest way of measure the results.

For each resulting window $\mathbf{x}$, the state of the machine two hours into the future is used as the final label. For the experiments proposed in this thesis, the pre-processes dataset is further divided into a training and testing set in a $80\% - 20\%$ ratio.

The following section presents the results obtained for this case study.

8.2 Results and Discussion for the Pumping System Dataset

This section review the results obtained from executing the experiments described in section 6.2 using the proposed and baseline models for the pumping system dataset. The structure will be very similar to the one used in section 7.2 and while it may read repetitive, it is done this way to assure that each case study is as self-contained as possible. Therefore, this case study will also be divided in sub-sections that aim to extract conclusions about particular components of the proposed approach. All the results shown in this section are obtained using the testing dataset, unless state otherwise.

8.2.1 Assessing the different time-structures of the encoder and decoder

First, the evolution of the cost functions for the predictor (cross-entropy) and tVAE (ELBO) during the training process are presented in figures 8.2. All the figures contained the evolution of the curves for both the training and testing dataset, to check if the training process developed some type of overfitting. These curves are obtained using $p = 50\%$ to show a representative semi-supervised case. Other cases were tested but the curves do not changed considerably in a way that affects the analysis presented below.

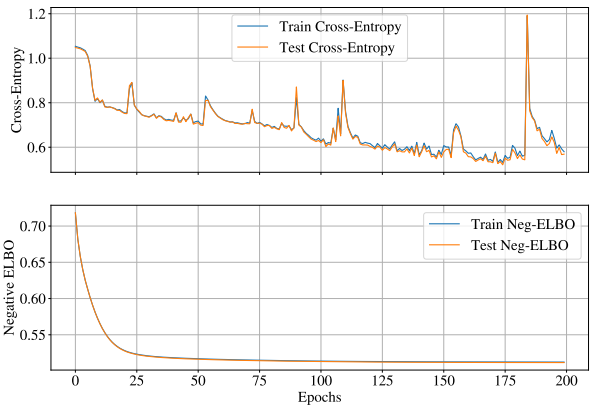
As figure 8.2 shows, there is no overfitting in the training process of the proposed model for any of the temporal variants. Both curves show similar profiles, which indicate that the model is able to generalize the information from the training to the testing dataset.

Another interesting aspect of figure 7.2 is that for the three variants, both the ELBO and the cross-entropy cost functions improve over the full training process, therefore, the proposed model is learning how to do two functions simultaneously: generate or reconstruct the input sequences through the tVAE and diagnose the health state of the pumping system through the predictor network. A final commentary with respect to the training curves is that while the ELBO improves rapidly and smoothly, the cross-entropy loss is very chaotic and takes a large amount of epochs to reach the termination condition for the training, described in section 6.2. A possible explanation for this is that the tVAE is optimizing first its ability to generate the input data, and based on that latent representation, then it is adapting it to also use it for the classification task. Nevertheless, both error curves (Cross-Entropy and negative ELBO) are not close to zero, so there is still room for improvement in the prediction and reconstruction abilities of the model.

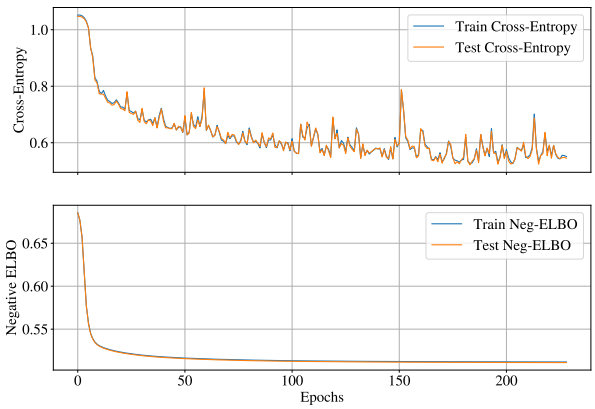
The effect of the temporal variants presented in table 7.3 over the balanced accuracy metric for the proposed model is presented in figure 8.3.

The first notable aspect of figure 8.3 is that for all temporal variants of the proposed model, the balanced accuracy metric improves with more labeled examples. This is typical in semi-supervised learning, where more labeled information directly benefits the generalization ability of the model to recognize the classes in the testing set from the examples seen in the training set.

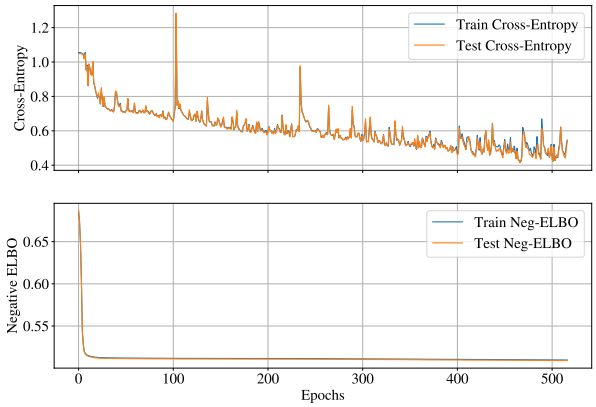
A second aspect that can be extracted from these results is that the stabilization of the balanced accuracy with respect to the increment of labeled information only starts to



(a) Temporal independent structure



(b) Past and present dependent structure



(c) Full temporal range dependent structure

Figure 8.2: Evolution of the cross-entropy and ELBO costs of the proposed model for the train and test dataset during training.

happen at a higher value of p than for the previous case study. Here, minor improvements are still achieved when augmenting p from 75% to 100%, at least for the models that use temporal variant 2 and 3. This puts in evidence that the generalization process for this dataset is relatively harder than for the FD004 dataset. Finally, comparing the different temporal variants between them, it is found that the differences in performance are much more prominent for the cases where the amount of labeled information is scarce ($p \leq 0.2$). In fact, for those cases, variant 3 is clearly the best model in average performance. This indicate that the dataset effectively contain temporal dependencies that are fundamental to predict the future health states, and therefore models that can extract information from sequences are better suited for the diagnosis tasks in this case study, especially in the semi-supervised scenario.

Table 8.3 show the overall improvements achieved in terms of balanced accuracy with the increase in labeled information, from $p = 0.01$ to $p = 1$. As it can be seen, for this case study the differences between them are also negligible if the standard deviation is considered. Therefore, while better accuracy results are achieved when using a more complex variant of the tVAE, the rate at which these variants improve with more available labeled information is very similar.

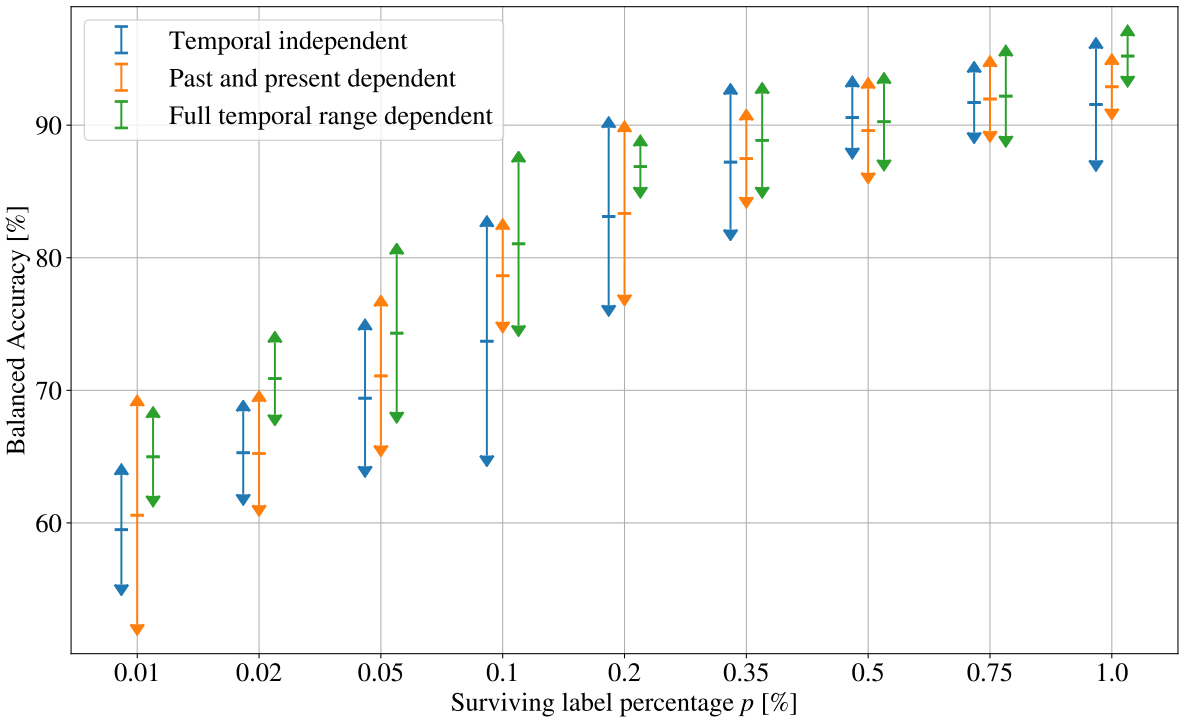


Figure 8.3: Balanced accuracy score for the different temporal structures of the proposed model.

Table 8.3: Performance improvement as the amount of labeled information increases from $p = 0.01$ to $p = 1$

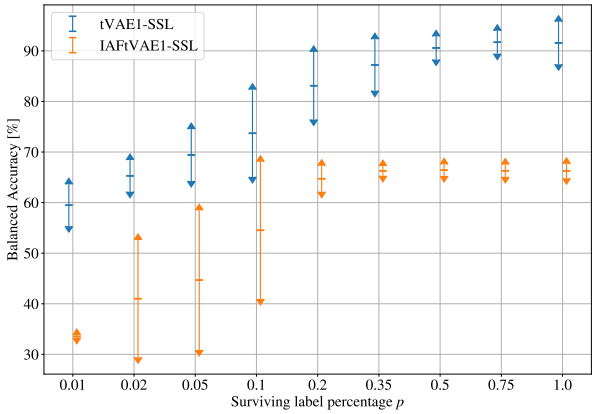
Temporal Variant	Balanced accuracy improvement [%]
tVAE1-SSL	32.07
tVAE2-SSL	32.31
tVAE3-SSL	30.22

From these results is clear that better downstream tasks performances in cases where the amount of labeled information is very scarce ($p \leq 0.2$) can be achieved using more temporally-aware inner structures for the tVAE in the proposed model. However, for cases where labels are abundant, the differences between different temporal variants starts to banish and the extra complexity and computational cost is not justifiable. What is yet to be discovered is if these results can be improved further by adding flexibility to the latent space, which in the tVAE-SSL model is parameterized by a diagonal normal distribution for each temporal state. The following section analize the results obtained from adding a normalize flow to the proposed model in order to answer this question.

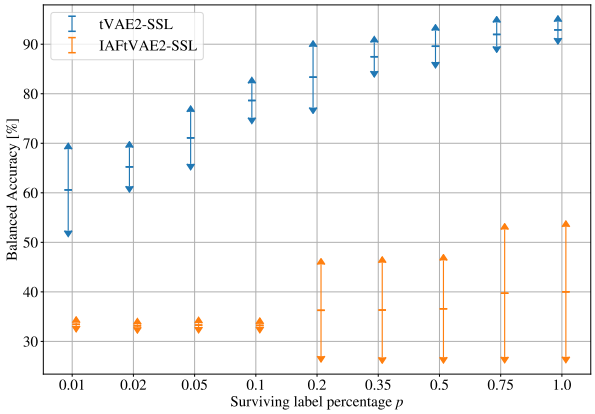
8.2.2 Assessing the improvements achieved by using Inverse Auto-Regressive Flows

The following results, presented in figure 8.4, compare the proposed approach (tVAE-SSL) with the inverse auto-regressive flow modified model (IAFtVAE-SSL) to evaluate if the flexibilization of the latent space can help to achieve better accuracy results when the amount

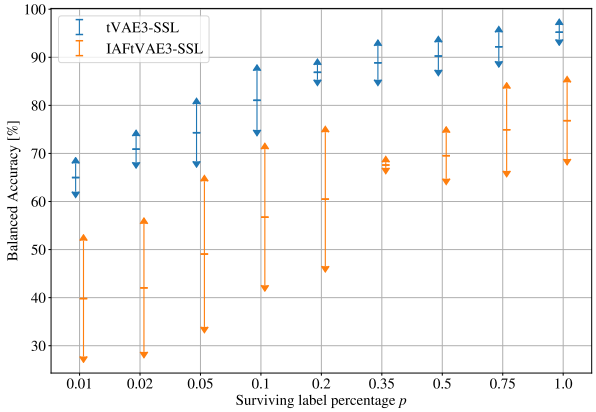
of labeled information varies.



(a) Temporal independent structure



(b) Past and present dependent structure



(c) Full temporal range dependent structure

Figure 8.4: Balanced accuracy results between the proposed model and the modification that incorporates the inverse-auto-regressive block.

In contraposition with the results obtained for the prognosis case study (see section 7.2), figure 8.4 shows that for all temporal variants of the tVAE and for all levels of labeled information, it is clear that the use of inverse auto-regressive flows heavily decrease the performance of the models. In fact, there exist cases where the model does not learn anything, evidence by the obtained accuracy 33%, which in a classification problem of three categories is the same as a random guess. In other cases, the dataset collapses one of the three available clases into the other two, resulting in a 66% of accuracy.

One possible cause for the differences in performance of the IAFtVAE-SSL model in this case study versus the prognosis case (see figure 7.4) is related to the proposed form in which the inverse auto-regressive flow was adapted to the sequential latent space. As explained in section 4.4, the normalizing flow is applied in a time-distributed manner, to each state t separately. Whereas the latent states are already independent between them, this approach further differentiate them without taking into consideration their origins, which are the observable variables. This additional transformation of the latent variables could cause the distributions for each $\vec{z}_t$ to take very different forms between them, to a point in which a diagnosis task, that is based in similarities between data points of the same class, is heavily

affected. Regression, on the other hand, would not be affected that much by this loss of similarity since it is not directly based on it.

Additional insights can be extracted table 8.4, where both the log-probability and RMSE of the reconstructed input data $\mathbf{x}$ are shown. As it can be seen, the reconstruction measures are very similar with or without the inclusion of an IAF block. Therefore, while the use of normalizing flows to improve the downstream tasks are clearly worsen, the latent space with added flexibility is still useful for generation purposes (although no improvements are observed in this regard either). This evidence that for this case study, a simple diagonal distribution is enough to encode a latent representation that can produce the necessary features to both generate data via the decoder and classify the health states with the predictor.

Table 8.4: Reconstruction metrics for the tVAE model with and without a IAF block for the three temporal variants.

	tVAE-SSL		**IAFtVAE-SSL**	
	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$
Variant 1	-0.513	0.085	-0.511	0.082
Variant 2	-0.512	0.084	-0.510	0.081
Variant 3	-0.507	0.074	-0.508	0.076

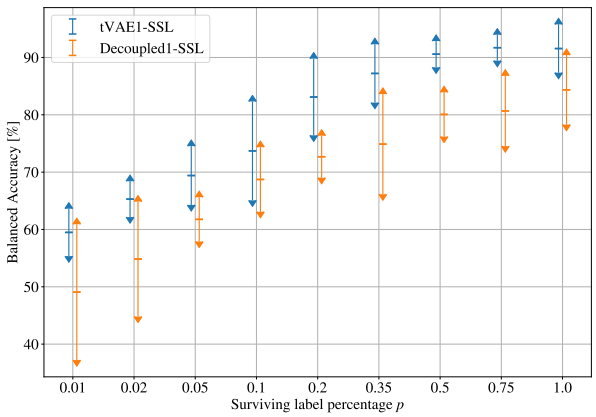
The next section is targeted towards verifying if the coupled training process holds any improvement over the traditional approach of optimizing the tVAE and the predictor separately.

8.2.3 Assessing the improvements achieved by the coupled training process

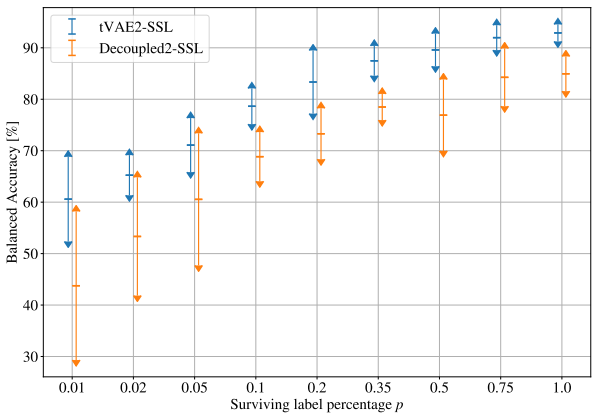
Figure 8.5 presents the comparison between the proposed model and the baseline that uses a decoupled training strategy. The objective of this experiment is to evaluate the value in performing a coupled training process for a diagnosis case.

As it can be seen from this experiment, improvements in the classification results are obtained by using a coupled training process versus a traditional decoupled approach. This improvements are found consistently in every temporal variant and for every level of labeled information, with some very slight variations. In variations 1 and 2 it is also found that for p values under 0.05, i.e., very low levels of labels, the proposed approach obtain more stable results, evident in its lower standard deviations. For variant 3 the deviations from the average accuracy are similar for both models, probably due to the increase in the encoder inner computation power. These results show that, for this case study, using the gradients from the predictor's loss function to tune the encoder's parameters propitiate the generation of better latent variables for downstream tasks purposes. In this regard, the same effect is observed in the previous prognosis case study.

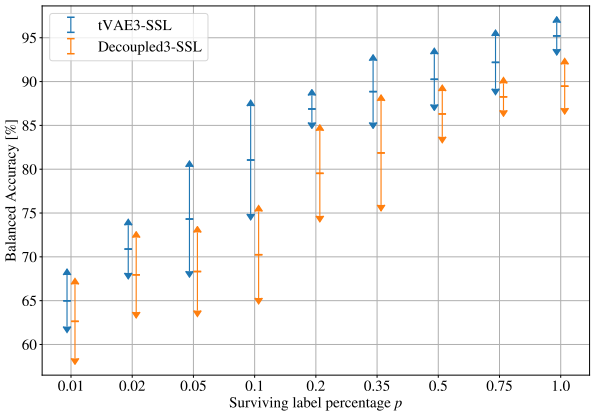
From table 8.5, which shows the reconstruction errors of the tVAE for the proposed model and the decoupled baseline, it can be observed that the direct use of labeled information for the generation of the latent space produces no noticeable effect on the generation capabilities



(a) Temporal independent structure



(b) Past and present dependent structure



(c) Full temporal range dependent structure

Figure 8.5: Balanced accuracy results between the proposed model and decoupled baseline.

of the tVAE, except for the third temporal variant, where the proposed model improves its metrics by a very slight margin while the decoupled model does not. This show that the gradient information that comes from the predictor's end only helps significantly for downstream tasks and do not transfer to the generation abilities of the model. In simpler words, for a particular data-point $\mathbf{x_i}$, the information contained in the correspondent label is not being used by the decoder to generate the reconstruction $\hat{\mathbf{x}}_\mathbf{i}$, and therefore a coupled strategy with the main objective of training a generative model is not justifiable for this case study.

Table 8.5: Reconstruction metrics for the proposed model and the decoupled baseline.

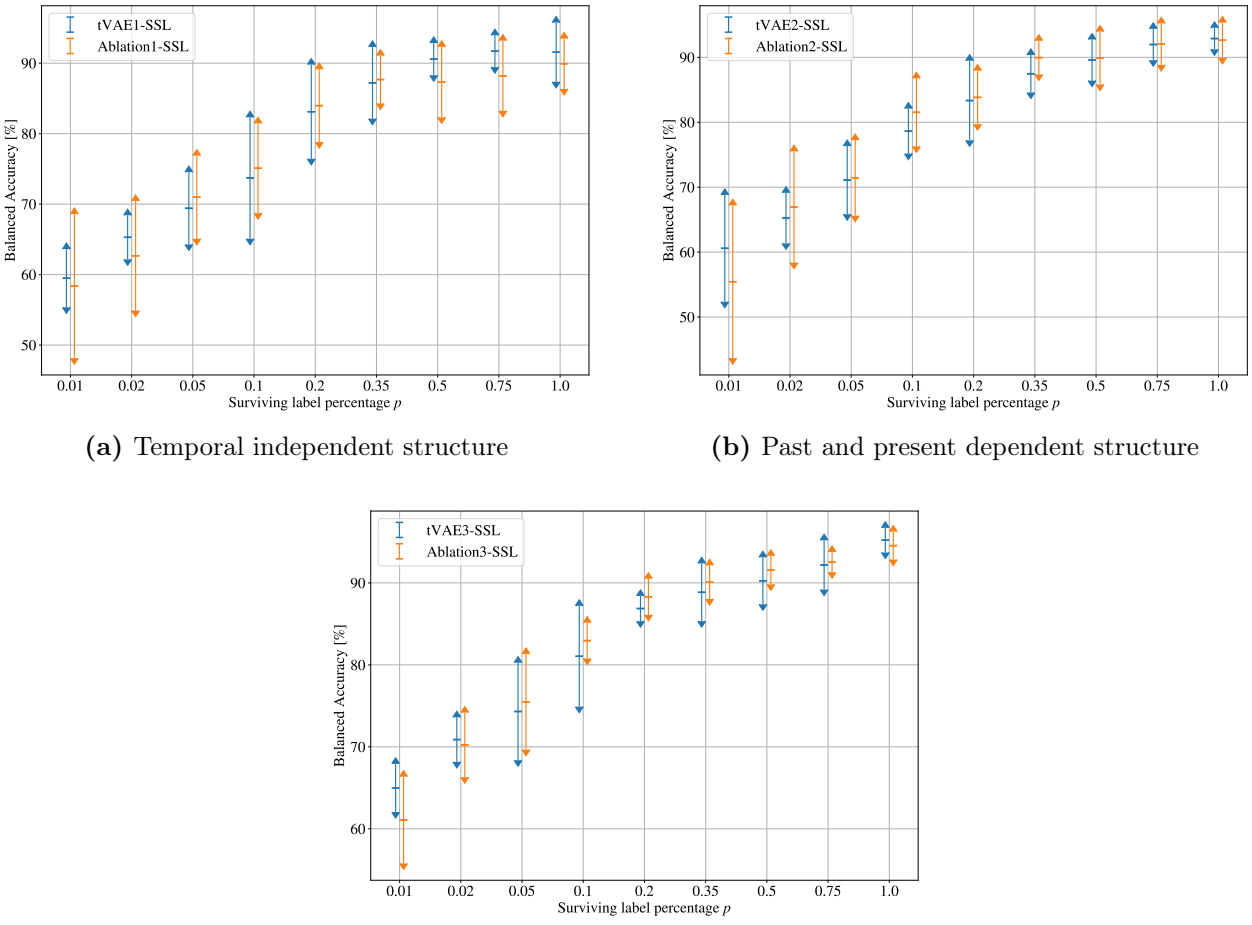
	tVAE-SSL		**Decoupled-SSL**	
	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$	$\log p(\mathbf{x})$	$RMSE(\mathbf{x}, \hat{\mathbf{x}})$
Variant 1	-0.513	0.085	-0.512	0.084
Variant 2	-0.512	0.084	-0.511	0.085
Variant 3	-0.507	0.074	-0.513	0.084

As a summary, this experiment show that the improvements obtained by switching the training strategy from a decoupled process to a coupled one are only observed in the down-

stream task performance in a consistent way, for every value of p and every temporal variant, while the generation (i.e. reconstruction) capabilities of the tVAE are almost not affected. The final section of this chapter is devoted to discovering why the coupled training purposes improves the classification results. In particular, which one of the sources of information (unsupervised via the ELBO or supervised via the predictor's loss) the encoder is using the most to improve the latent space of the tVAE.

8.2.4 Results from the Ablation Study

The final comparation for this chapter is presented in figure 8.6, where the proposed model is contrasted against the ablation study baseline. The ablation study model, as described in section 6.1.1, is the result of removing from the proposed model any capability of generating the latent space using unsupervised information via the elimination of the ELBO function from the total cost C. This transforms the proposed model into what is, for all practical purposes, a recurrent neural network that only learn from the supervised portion of the data using the predictor's loss function L, ignoring the unlabeled data-points.



(a) Temporal independent structure

(b) Past and present dependent structure

(c) Full temporal range dependent structure

Figure 8.6: Balanced accuracy results between the proposed model and Ablation study baseline.

Figure 8.6 shows that, for all levels of labeled information and the three temporal variants, there are no statistically significant improvements in the balanced accuracy results by using

variational inference to learn unsupervised characteristics in the latent space. The fact that the proposed model do not achieve better results over the ablation study for lower values of p indicates that the coupled training strategy do not improves the semi-supervised regression results by capturing additional information from the labels to the latent space, but instead that **all** the important information to generate predictions is captured from this few labeled examples. This effectively means that the tVAE model is not able to capture useful features for the classification task from the unsupervised portion of the data, or at least is not able to capture features that could generate an improvement over those captured from the supervised portion of the data by the predictor's network. This result is very similar to that obtained in the prognosis case study.

However, as discussed in section 8.2.1, the training curves for the proposed model show evidence that the tVAE is effectively learning two functions at the same time: how to generate and reconstruct the input sequences in the outputs and how to encode the input data in a way that can achieve good results in the classification tasks. This can be seen from the fact that both costs are lower at the end of the training process with respect to the start of it. Therefore, one possible conclusion is that while the proposed model is not able to use the tVAE to extract unsupervised information for the classification task, it is able to generate a dual function latent space which is trained to perform two different tasks at the same time.

In a similar fashion to what happens in the prognosis case study, it results very interesting to note that while the semi-supervised learning results are not improved by the proposed model, as the ablation study shows, the coupled training strategy can produce a single latent representation capable of both functions performing two different functions at the same time, without any of them being totally disrupted by the other. This represent an unexpected result from this experimentation and could be used as a start point for future research work.

A complete set of the accuracy results for all the models, temporal variants and levels of labeled information is included in the appendix, table 10.2.

Chapter 9

Concluding Remarks

Prognosis and diagnosis tasks have become an essential part of system health and operation management, allowing engineers and other personnel to anticipate risk situations, reducing operational costs, maintaining higher levels of productivity and guarding the safety of its workers. During recent years, the PHM field have starting to use deep learning techniques for these purposes, with great success in a variety of research examples. Nevertheless, one problem that keep all these model from being directly applicable to real world situations is the lack of labeled data, from which deep learning models traditionally learn its patter recognition abilities. For this purpose, this thesis proposes the research of a semi-supervised learning approach that combine deep learning techniques with deep generative models to learn from both labeled and unlabeled information.

In particular, this thesis proposes a model that trains a temporal VAE and a predictor neural network in a coupled manner, fusing them together into what is a mixed model with two different objective functions to perform prognosis and diagnosis tasks using operational time-series as input data. This coupled training procedure allows for the flow of information originally contained in the labels to the encoder of the tVAE, possibly improving the approximation of the posterior distribution $q(\mathbf{z}|\mathbf{x})$ and therefore resulting in better results for downstream tasks. The proposed model takes advantage of the unsupervised nature of the tVAE and the fully supervised nature of the predictor to fully utilize the information, whether it is labeled or not. Also, normalizing flows are tested to assess their ability to further improve the latent space of the proposed model by transforming the encoder's distribution to arbitrarily complex distributions.

A complete set of experiments are performed over the proposed model, where the inner structure of the tVAE as well as the level of labeled information used is varied. In particular, three different temporal variants for the encoder and decoder of the tVAE are proposed, each one representing a different level of complexity. These experiments are executed over two case studies. The first one is a prognosis example that uses the well known benchmark dataset FD004 for the prediction of RUL in turbofans. The second example of application is a diagnosis example that uses data from a pumping system for offshore oil extraction. Additionally, two baseline models are designed and tested to validate the advantages of using the proposed approach over more simple methodologies. The first baseline model uses the

same architecture as the proposed model but is trained in a decoupled manner, following the traditional approach of using VAEs to perform semi-supervised learning. The second baseline model is the result of an ablation study in which the VAE loss function is removed from the proposed model, effectively transforming it into a fully supervised recurrent neural network.

The performed experiments show that for the proposed model, general improvements can be achieved by using more temporally-aware inner structures in both the encoder and decoder of the tVAE. This was an expected result due to the temporal nature of the operational data from both case studies, which explain why better semi-supervised results are obtained with the use of neural networks designed to process time-series.

With respect to the use of normalizing flows, specifically Inverse Auto-Regressive Flows (IAF), to increase the flexibility of the latent variables, different results are found for each case study. Whereas in the prognosis example IAF produced very slight improvements, for the diagnosis example they totally disrupted the results, making their use not recommended. It is discussed that one possible explanation for this has relation with the added flexibility that IAF induced in the latent variables, which could possible also tend to differentiate them. This loss of similarity would generate a loss of performance for classification tasks, where similarity between examples is important, but leave regression problems more or less untouched where the optimization is based on other aspects.

From the comparison of the proposed approach with the first baseline model, that uses a decoupled training strategy, it can be concluded that the proposed conjunct training generate better results for both the prognosis and diagnosis example in almost all levels of labeled information and all temporal variants. Therefore, it is true that the use of labeled information to tune the encoder of the VAE effectively is a better training strategy than the traditional approach, where the VAE and the predictor are trained separately.

However, the comparison with the second baseline, which is the ablation study, show that these improvements come exclusively from a more powerful supervised neural network, product of the concatenation of the encoder and the predictor neural network, instead of the unsupervised features extracted by the tVAE. In this regard, the use of a mixed model is unjustified for both case studies, since comparable results can be obtained from using a deeper neural network with only the labeled portion of the data, for all levels of labeled information. This might be explained, in part, by the relatively easiness of the problems explored in this work or the inability of the VAE model to extract useful unsupervised features. Therefore, future research could focus on trying more advanced Deep Generative Models with the same training approach or testing more complex datasets where generalization is harder than the ones presented in this thesis.

Nevertheless, as the training curves for the proposed model show, in both case studies the prediction cost and tVAE cost decrease as the training process is executed. This exhibit that the proposed model is learning how to both reconstruct the input data with the tVAE and generate predictions with the predictor neural network using the same latent space as a starting point. While the use of the proposed model is not justified if the only objective is to perform semi-supervised learning towards obtaining better predictions, this dual function latent space could result in an interesting model for tasks in which the generation of new data and the assessment of the health state of the system is required at the same time. Since

the objectives of this thesis were focused on the predictive abilities of the proposed model, this new aspect is also left as a possible direction for future research works.

Chapter 10

Appendix

Table 10.1: Complete RMSE results for all models in the FD004 dataset.

p/Model	**tVAE-SSL**	**IAFtVAE-SSL**	**Decoupled**	**Ablation**
Temporal Variant 1				
0.01	31.42 ± 1.46	31.69 ± 4.95	29.11 ± 1.38	31.85 ± 1.52
0.02	28.86 ± 0.93	27.36 ± 1.45	28.58 ± 1.25	29.10 ± 1.22
0.05	27.51 ± 0.47	26.42 ± 1.04	28.38 ± 1.41	27.59 ± 0.61
0.1	26.80 ± 0.53	25.21 ± 0.90	28.53 ± 1.22	27.05 ± 0.91
0.2	25.90 ± 0.91	25.34 ± 1.80	25.39 ± 0.96	26.31 ± 0.76
0.35	25.80 ± 1.15	23.59 ± 1.33	24.78 ± 0.84	25.73 ± 1.51
0.5	25.98 ± 0.95	23.53 ± 1.12	24.16 ± 1.05	25.26 ± 1.40
0.75	24.79 ± 1.55	23.96 ± 1.11	24.03 ± 1.46	25.07 ± 1.74
1.0	25.23 ± 1.25	23.66 ± 1.41	23.89 ± 1.19	25.27 ± 1.72
Temporal Variant 2				
0.01	30.22 ± 1.16	30.53 ± 1.33	32.71 ± 1.28	30.85 ± 1.39
0.02	28.39 ± 0.63	28.58 ± 1.78	32.07 ± 1.56	28.54 ± 1.19
0.05	27.00 ± 0.91	26.92 ± 0.72	32.09 ± 1.72	27.12 ± 0.64
0.1	26.17 ± 0.87	25.82 ± 0.55	34.10 ± 1.48	26.06 ± 0.88
0.2	25.39 ± 1.34	25.71 ± 0.84	28.53 ± 1.62	25.07 ± 0.76
0.35	25.11 ± 0.96	25.01 ± 1.18	27.13 ± 1.33	24.61 ± 0.90
0.5	24.22 ± 0.86	24.71 ± 1.22	27.37 ± 1.46	24.76 ± 1.19
0.75	24.29 ± 1.01	24.67 ± 0.91	27.00 ± 1.69	24.22 ± 1.09
1.0	24.48 ± 1.05	24.68 ± 0.99	26.53 ± 1.59	24.56 ± 1.18
Temporal Variant 2				
0.01	29.69 ± 1.16	30.10 ± 1.05	33.56 ± 2.03	30.27 ± 1.58
0.02	28.12 ± 0.84	28.98 ± 1.75	32.25 ± 1.05	28.46 ± 1.49
0.05	26.68 ± 1.55	25.89 ± 1.49	32.85 ± 2.02	26.72 ± 1.70
0.1	24.96 ± 1.78	24.07 ± 1.27	33.47 ± 2.34	25.41 ± 1.59
0.2	23.51 ± 0.76	24.07 ± 1.11	28.26 ± 1.31	24.25 ± 1.60
0.35	23.75 ± 1.14	23.15 ± 0.91	27.45 ± 1.47	23.27 ± 0.79
0.5	23.02 ± 0.59	23.40 ± 0.46	27.74 ± 1.76	23.81 ± 1.70
0.75	23.26 ± 0.67	23.06 ± 0.93	27.08 ± 1.67	22.93 ± 1.17
1.0	23.28 ± 1.22	23.18 ± 0.90	27.08 ± 1.54	22.68 ± 0.69

Table 10.2: Complete balanced accuracy (in percentage) results for all models in the pumping system dataset.

p/Model	**tVAE-SSL**	**IAFtVAE-SSL**	**Decoupled**	**Ablation**
		Temporal Variant 1		
0.01	59.48 ± 4.13	33.54 ± 0.27	49.06 ± 11.8	58.37 ± 10.2
0.02	65.28 ± 3.13	41.01 ± 11.6	54.83 ± 10.0	62.66 ± 7.78
0.05	69.40 ± 5.13	44.68 ± 13.8	61.76 ± 3.86	70.97 ± 5.90
0.1	73.70 ± 8.62	54.52 ± 13.5	68.72 ± 5.63	75.09 ± 6.37
0.2	83.11 ± 6.69	64.72 ± 2.57	72.68 ± 3.66	83.97 ± 5.17
0.35	87.21 ± 5.09	66.26 ± 0.96	74.88 ± 8.74	87.68 ± 3.40
0.5	90.57 ± 2.29	66.41 ± 1.16	80.06 ± 3.86	87.31 ± 5.00
0.75	91.70 ± 2.26	66.30 ± 1.22	80.66 ± 6.13	88.21 ± 4.98
1.0	91.55 ± 4.21	66.24 ± 1.42	84.35 ± 6.06	89.92 ± 3.57
		Temporal Variant 2		
0.01	60.58 ± 8.22	33.44 ± 0.33	43.75 ± 14.4	55.43 ± 11.7
0.02	65.23 ± 3.90	33.14 ± 0.29	53.33 ± 11.4	66.95 ± 8.57
0.05	71.08 ± 5.24	33.29 ± 0.38	60.54 ± 12.8	71.43 ± 5.82
0.1	78.63 ± 3.47	33.23 ± 0.30	68.83 ± 4.77	81.52 ± 5.24
0.2	83.34 ± 6.13	36.27 ± 9.24	73.29 ± 4.94	83.82 ± 4.12
0.35	87.47 ± 2.89	36.33 ± 9.54	78.49 ± 2.54	89.95 ± 2.60
0.5	89.58 ± 3.17	36.58 ± 9.76	76.90 ± 6.93	89.89 ± 4.08
0.75	91.97 ± 2.42	39.72 ± 12.8	84.26 ± 5.61	92.03 ± 3.21
1.0	92.89 ± 1.65	40.01 ± 13.1	84.94 ± 3.37	92.64 ± 2.71
		Temporal Variant 2		
0.01	64.98 ± 2.94	39.80 ± 12.0	62.63 ± 4.22	61.06 ± 5.29
0.02	70.88 ± 2.72	42.03 ± 13.3	67.94 ± 4.24	70.23 ± 3.94
0.05	74.31 ± 5.94	49.06 ± 15.1	68.32 ± 4.45	75.48 ± 5.84
0.1	81.04 ± 6.15	56.73 ± 14.1	70.24 ± 4.93	82.95 ± 2.19
0.2	86.87 ± 1.54	60.48 ± 13.9	79.52 ± 4.85	88.30 ± 2.19
0.35	88.85 ± 3.52	67.58 ± 0.58	81.85 ± 5.93	90.09 ± 2.05
0.5	90.25 ± 2.87	69.53 ± 4.78	86.30 ± 2.61	91.56 ± 1.74
0.75	92.19 ± 3.01	74.92 ± 8.56	88.25 ± 1.53	92.53 ± 1.24
1.0	95.21 ± 1.50	76.81 ± 7.97	89.47 ± 2.49	94.55 ± 1.70

Bibliography

[1] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[2] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889, 2015.

[3] Leo H Chiang, Mark E Kotanchek, and Arthur K Kordon. Fault diagnosis based on fisher discriminant analysis and support vector machines. *Computers & chemical engineering*, 28(8):1389–1401, 2004.

[4] Bo Suk Yang, Chul Hyun Park, and Ho Jong Kim. An efficient method of vibration diagnostics for rotating machinery using a decision tree. *International Journal of Rotating Machinery*, 6(1):19–27, 2000.

[5] Richard Dearden and Dan Clancy. Particle filters for real-time fault detection in planetary rovers. 2001.

[6] Behrad Bagheri, Hojat Ahmadi, and Reza Labbafi. Application of data mining and feature extraction on intelligent fault diagnosis by artificial neural network and k-nearest neighbor. In *The XIX International Conference on Electrical Machines-ICEM 2010*, pages 1–7. IEEE, 2010.

[7] Yongzhi Zhang, Rui Xiong, Hongwen He, and Michael G Pecht. Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, 67(7):5695–5705, 2018.

[8] Guoqian Jiang, Haibo He, Jun Yan, and Ping Xie. Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox. *IEEE Transactions on Industrial Electronics*, 66(4):3196–3207, 2018.

[9] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE, 2008.

[10] Bearing data center. `https://csegroups.case.edu/bearingdatacenter/home`. (Accessed on 02/04/2020).

[11] Huan Huang and Natalie Baddour. Bearing vibration data collected under time-varying rotational speed conditions. *Data in brief*, 21:1745–1749, 2018.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Gabriel San Martin, Enrique López Droguett, Viviane Meruane, and Márcio das Chagas Moura. Deep variational auto-encoders: A promising tool for dimensionality reduction and ball bearing elements fault diagnosis. *Structural Health Monitoring*, 18(4):1092–1128, 2019.

[14] Andre S Yoon, Taehoon Lee, Yongsub Lim, Deokwoo Jung, Philgyun Kang, Dongwon Kim, Keuntae Park, and Yongjin Choi. Semi-supervised learning with deep generative models for asset failure prediction. *arXiv preprint arXiv:1709.00845*, 2017.

[15] Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Thirty-first aaai conference on artificial intelligence*, 2017.

[16] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3601–3610. Curran Associates, Inc., 2017.

[17] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[18] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[19] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[20] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[21] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay

Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[22] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[23] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[24] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

[25] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[26] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[27] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[28] Mohamad H Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[30] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[31] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[32] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

[33] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[34] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics, 2011.

[35] John D Kelleher, Brian Mac Namee, and Aoife D'arcy. *Fundamentals of machine learning*

for predictive data analytics: algorithms, worked examples, and case studies. MIT press, 2015.