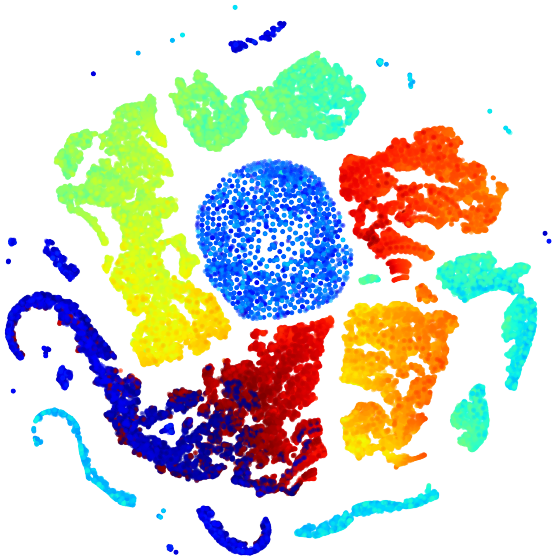
TÉCNICO LISBOA



Unsupervised Anomaly Detection in Time Series Data using Deep Learning

João Pedro Cardoso Pereira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Maria Margarida Campos da Silveira
Eng. Francisco Miguel Pereira Gonçalves

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Maria Margarida Campos da Silveira
Member of the Committee: Prof. Jorge dos Santos Salvador Marques

November 2018

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgements

I would like to thank to my supervisor, Prof. Margarida Silveira, for having proposed this Thesis. In particular, I would like to thank her for giving me the freedom and the encouragement of pursuing my own ideas and for supporting them in a constructive way, adding value to them. Moreover, I thank the opportunity to get involved in the astonishing field of Machine Learning and to work in cutting-edge research topics such as Bayesian deep learning, deep generative modelling, anomaly detection, sequence modelling with recurrent neural networks and attention.

I would like to thank C-Side, in the person of its CEO, Eng. Francisco Gonçalves, for having provided the dataset that made the quest for this Thesis. It was a privilege to develop applied research: to improve the current state of anomaly detection and, at the same time, to solve a practical problem.

I thank to my family for always having provided me with the conditions for my studies at IST and for giving a persistent support, all the time, throughout this journey of my life.

I thank to all those who have accompanied me during this challenging period of life, in particular my colleagues from IST and UCLouvain and my friends outside the university.

I thank to Instituto Superior Técnico for having educated me with the fundamental knowledge at the core of this Thesis. I also thank to Université catholique de Louvain, where I spent a mobility period as an exchange student within the Erasmus+ Programme, for the interesting projects and topics covered in the courses I took there, some of which turned out to be useful for the development of the proposed approach in this Thesis, such as unsupervised dimensionality reduction and the principles of sparse representations.

Finally, I would like to jointly thank to my supervisor, C-Side and IST for the opportunity of publishing the results of my work, so that they can be shared with other researchers and practitioners of the field and, perhaps, be applied to other domains and problems that I have never thought about.

Resumo

Detetar anomalias em séries temporais é um problema importante em áreas como energia, saúde e segurança. O progresso feito em deteção de anomalias tem sido baseado em abordagens que usam algoritmos supervisionados de aprendizagem automática que requerem grandes conjuntos de dados anotados para ser treinados. No entanto, no contexto das aplicações, colecionar e anotar conjuntos de dados em grande escala é um processo difícil, demorado ou até demasiado caro, ao mesmo tempo que requer conhecimento do assunto por especialistas da área de aplicação. Por isso, a deteção de anomalias tem sido um grande desafio para investigadores e profissionais.

Esta Tese propõe uma abordagem genérica, não supervisionada e escalável para deteção de anomalias em séries temporais. A abordagem proposta é baseada num auto-codificador variacional, um modelo generativo profundo que combina inferência variacional com aprendizagem profunda. Por outro lado, a arquitetura integra redes neuronais recorrentes para capturar a natureza sequencial das séries temporais e as suas dependências temporais. Além disso, é introduzido um mecanismo de atenção para melhorar o desempenho do processo de codificação-descodificação.

Os resultados em dados de geração solar fotovoltaica e de electrocardiogramas mostram a capacidade do modelo proposto para detetar padrões anómalos em séries temporais de diferentes áreas de aplicação, ao mesmo tempo fornecendo representações estruturadas e expressivas dos dados.

Palavras-chave: Deteção de Anomalias, Séries Temporais, Auto-Codificadores Variacionais, Redes Neuronais Recorrentes, Mecanismos de Atenção.

Abstract

Detecting anomalies in time series data is an important task in areas such as energy, healthcare and security. The progress made in anomaly detection has been mostly based on approaches using supervised machine learning algorithms that require big labelled datasets to be trained. However, in the context of applications, collecting and annotating such large-scale datasets is difficult, time-consuming or even too expensive, while it requires domain knowledge from experts in the field. Therefore, anomaly detection has been such a great challenge for researchers and practitioners.

This Thesis proposes a generic, unsupervised and scalable framework for anomaly detection in time series data. The proposed approach is based on a variational autoencoder, a deep generative model that combines variational inference with deep learning. Moreover, the architecture integrates recurrent neural networks to capture the sequential nature of time series data and its temporal dependencies. Furthermore, an attention mechanism is introduced to improve the performance of the encoding-decoding process.

The results on solar energy generation and electrocardiogram time series data show the ability of the proposed model to detect anomalous patterns in time series from different fields of application, while providing structured and expressive data representations.

Keywords: Anomaly Detection, Time Series, Variational Autoencoder, Recurrent Neural Networks, Attention Mechanism

Contents

List of Tables

List of Figures

List of Acronyms

AD	Anomaly Detection.
AE	Autoencoder.
AEVB	Auto-Encoding Variational Bayes.
AM	Attention Mechanism.
AUC	Area Under the Curve.
Bi-LSTM	Bidirectional Long Short-Term Memory.
BPTT	Backpropagation Through Time.
DBN	Dynamic Bayesian Network.
DL	Deep Learning.
DR	Dimensionality Reduction.
ECG	Electrocardiogram.
ELBO	Evidence Lower Bound.
GAN	Generative Adversarial Network.
GMM	Gaussian Mixture Model.
HMM	Hidden Markov Model.
IoT	Internet of Things.
KL	Kullback-Leibler.
KPI	Key Performance Indicator.
LSTM	Long Short-Term Memory.
MCMC	Markov Chain Monte Carlo.
MI	Mutual Information.
ML	Machine Learning.
NLP	Natural Language Processing.
OC-SVM	One-Class Support Vector Machine.
PCA	Principal Component Analysis.

PV Photovoltaic.

RE Reconstruction Error.

RNN Recurrent Neural Network.

ROC Receiver Operating Characteristic.

Seq2Seq Sequence-to-Sequence.

SGD Stochastic Gradient Descent.

SVM Support Vector Machine.

t-SNE t-Distributed Stochastic Neighbor Embedding.

UL Unsupervised Learning.

VAE Variational Autoencoder.

VI Variational Inference.

VRAE Variational Recurrent Autoencoder.

VSAM Variational Self-Attention Mechanism.

Chapter 1

Introduction

This chapter introduces the motivation behind this Thesis and its main subject: Anomaly Detection (AD). It is presented an overview of related work on AD as well as an explanation of the importance of unsupervised learning in the context of AD and beyond. Finally, are outlined the objectives and the requirements to be fulfilled by the proposed approach.

1.1 Motivation

In the age of Big Data, time series are being generated in massive amounts. Nowadays, sensors and Internet of Things (IoT) devices are ubiquitous and produce data continuously. While the data gathered by these devices is valuable and can provide meaningful insights, there is a growing need for developing algorithms that can process these data efficiently. Moreover, in the context of applications such as energy, healthcare, security, finance and robotics it is important to analyse and monitor the collected data in order to detect anomalous behaviour that can allow further decisions and actions.

This Thesis is motivated by an application of anomaly detection that arises from the energy field. One of the key assets of the smart grid is the data it collects. Smart grids are enabling an unprecedented data acquisition with the installation of sensors and smart devices. With the integration of renewable energy sources such as solar photovoltaic (PV), it is important to ensure reliability, security and correct operation in order to promote good performances and a long lifetime of the equipments. Such an amount of data gathered from smart meters in the grid makes the quest for developing smart monitoring systems that can detect anomalous behaviour in these systems, trigger alerts and enable maintenance operations.

However, the need for anomaly detection systems goes beyond the energy field. For instance, in healthcare, the availability of patient monitoring data and the increasing amount of wearable sensors that collect vital signs (e.g., heart rate, electrocardiogram) makes the quest for developing AD algorithms that can detect anomalous patterns and provide alerts with minimum delay, thus allowing an early detection of abnormal vital signs and a clinical intervention when required.

The progress made in Machine Learning (ML) and, in particular, in Deep Learning (DL), allows to build models that learn directly from data without extensive pre-processing and significant domain

knowledge from experts in the field of application. However, a unified framework for AD that can leverage the power of recent DL models and, at the same time, that could be applied to any kind of time series data is still to be done.

1.2 Anomaly Detection Overview

Anomaly Detection[1] refers to the problem of finding patterns in data that do not conform to expected or *normal* behaviour [Chandola *et al.*, 2009]. AD has been approached as a particular instance of a classification task that aims to distinguish between *normal* and *anomalous* observations. These classes are typically highly imbalanced, being the normal class often predominant relatively to the anomalous one.

Until a few years ago, the work on anomaly detection was underdeveloped. The quest for developing novel AD models and to move the state-of-the-art further has emerged from both the industry and the academia. Nowadays, anomaly detection is an active area of research that is being applied to a wide range of problems dealing with data of various nature, ranging from time series and text to images and videos. The applications of AD cover very different areas, such as energy, healthcare, security, finance and robotics. In particular, to give some examples, AD has been used to detect faults in power grids [Martinelli *et al.*, 2004], to detect arrhythmia in electrocardiogram time series [Ng *et al.*, 2017], to detect anomalous behaviour in surveillance videos [Mahadevan *et al.*, 2010], to detect fraud with credit cards [Aleskerov *et al.*, 1997], to detect abnormal opinions and sentiment patterns [Wang *et al.*, 2014], to detect anomalous executions in robot assisted feeding [Park *et al.*, 2017], to detect intrusions in networks [García-Teodoro *et al.*, 2009] and to detect unusual segments of text in documents [Guthrie *et al.*, 2007]. All these problems are particular instances of an anomaly detection task.

Recently, AD has been leveraging on the progress made in Machine Learning and, in particular, in Deep Learning (DL). In this framework, the work on anomaly detection has been mostly focused on supervised learning algorithms, which heavily rely on big labelled datasets to be trained. However, in the context of some applications, such as energy or healthcare, labels are difficult to obtain or even too expensive, while the annotation process is time-consuming and requires domain knowledge from experts. Therefore, the application of AD approaches based on supervised models is constrained by the availability of labelled datasets. Furthermore, in the aforementioned applications of AD, data is sequential, including time series, text or videos. Some previous AD methods based on ML algorithms assume that data is independent in time and, hence, they do not consider the temporal dependencies intrinsic to sequential data.

[1] Anomaly detection is also referred to as novelty detection or outlier detection. Even though these designations are sometimes used for slightly different problems, they are often framed as an anomaly detection task.

1.3 Related Work

Anomaly Detection is an old problem that has been tackled using different approaches over time. Chandola *et al.* [2009] and, more recently, Pimentel *et al.* [2014] provided an in-depth overview about the anomaly detection approaches that have been proposed. In particular, Pimentel *et al.* outlines five main classes of AD approaches that are briefly described and summarized in this section. For each method are provided some representative references of its application to AD tasks.

The first class integrates the **probabilistic approaches** and the principle behind them consists on estimating the probability density function (pdf) of the data and then the anomalies correspond to the regions with low probability mass. These methods usually make an independence assumption of the data in time or make specific assumptions about their generation process, such as assuming it is generated from a weighted mixture of Gaussian distributions (GMM) [Song *et al.*, 2007]. Independent Component Analysis (ICA) was also proposed for AD [Hansen *et al.*, 2002]. These techniques require a considerably large amount of data, specially when dealing with high-dimensions where the *curse of dimensionality* comes into play. For time series data, state-space models are usually employed. These models assume that the observations are generated by an underlying hidden state, which may evolve over time. The two most common state-space models are the Hidden Markov Model (HMM) [Yeung and Ding, 2003] and the Kalman filter [Lee and Roberts, 2008], which are both particular instances of Dynamic Bayesian Networks (DBN). However, as pointed out by Bengio *et al.* [2015a], these models often assume simple state transition structures (e.g., linear models in the case of the Kalman filter) or simple internal state structures (e.g., in HMMs the state space consists of a single set of mutually exclusive states).

The second class regards **distance-based approaches** that include Clustering [He *et al.*, 2003; Barbará *et al.*, 2002] or Nearest Neighbour [Boriah *et al.*, 2008; Angiulli and Pizzuti, 2002] methods. These approaches require well-defined distance measures to compute the similarity between data points. They assume that *normal* data points have close neighbours within the training set of normal examples, while *anomalous* data points are located far from those normal points. The more away from the normal data a given test sample is, the more likely it is to be an anomaly. Nevertheless, distance-based approaches sometimes require a significant amount of comparisons between data points, what compromises their scalability to large and high-dimensional datasets. This class of methods is prone to suffer from the curse of dimensionality, due to the concentration of the distances phenomenon in high-dimensional spaces.

The third class concerns **reconstruction-based approaches** and these are connected with neural networks and dimensionality reduction [Hawkins *et al.*, 2002; Williams *et al.*, 2002; Shyu *et al.*, 2003]. The prominent model of this class is the Autoencoder (described in detail in section 2.1), that learns a low-dimensional representation of normal data and then reconstructs it. The reconstruction scores are used as a measure of normality. Considering conventional (feed-forward and deterministic) autoencoders, this reconstruction-based approach is not particularly suited for sequential data such as time series, since it does not take into account the dependencies between the inputs.

The fourth class consists of **domain-based approaches** whose principles differ from the previous methods. Rather than modelling the structure of the data itself, they try to find a boundary around

normal data (in other words, the *domain* of the data). Unseen examples are classified as normal or anomalous depending on whether they are inside or outside the boundary. This class includes, for instance, One-Class Support Vector Machines (OC-SVMs) [Schölkopf *et al.*, 2001; Heller *et al.*, 2003]. The effectiveness of domain-based approaches often depend on the choice of good parameters, for instance the kernel used in the SVM.

The fifth and last main group of AD methods is based on **information-theoretic approaches**. These techniques are built upon the principles of *information theory* and make use of concepts such as the entropy [He *et al.*, 2005], the Mutual Information (MI) or the Kullback-Leibler (KL) divergence [Gamon, 2006]. The core idea behind them is that anomalous data has a different information content of normal data. In this context, entropies are computed (globally or locally) and their changes are used to find anomalous data. However, these approaches require choosing particular metrics that may not be suited for some kinds of anomalies, specially short-term ones.

1.4 Why Unsupervised Learning?

The astonishing success of Deep Learning has been achieved mainly with supervised machine learning algorithms using deep neural networks [Hinton *et al.*, 2012; Krizhevsky *et al.*, 2012]. For training these algorithms, big labelled datasets are required in order to attain good performances and state-of-the-art results. However, as previously mentioned, such large-scale labelled datasets are difficult to obtain and the annotation process requires domain knowledge from experts.

Despite the success of supervised learning in recent years, previously, unsupervised learning attained remarkable results. During the early days of the DL era, for instance, Hinton and Salakhutdinov [2006] showed impressive results in dimensionality reduction using autoencoders. Afterwards, the success of supervised learning in key problems, such as speech recognition and image classification, has concentrated the interest of the research community that developed and improved it significantly, while unsupervised learning was partially disregarded.

Recently, there has been a renewed interest in unsupervised learning that is foreseen to play an important role in the future of machine learning [LeCun, Bengio, and Hinton, 2015]. The explosion of work in unsupervised machine learning was seamlessly connected with the introduction of two novel deep generative models in particular, namely Variational Autoencoders (VAEs) [Kingma and Welling, 2013] and Generative Adversarial Networks (GANs) [Goodfellow *et al.*, 2014].

Still, unsupervised learning is a very challenging field that often under-performs supervised learning in a variety of tasks. In what regards AD, the approaches based on ML algorithms are often very focused on supervised models, even though there is a recent trend of adopting unsupervised approaches. The lack of labelled data is, more and more, making the quest for improving unsupervised learning models, so that they can be applied to other problems and tasks that where sometimes disregarded in the past.

Finally, it is interesting to note that the human way of reacting to unexpected observations of the world, that is to say, the human way of performing AD, is largely unsupervised. A naïve example, inspired by another one that Yann LeCun uses to give about unsupervised learning [LeCun, 2018],

goes as follows. A newborn is often confused about the environment it perceives in its early days of life. However, after experiencing the novel world for a while and collecting some observations of the environment, and still without any sort of supervision, a baby is able to build, that is to say, to learn, its own idea of normality and soon starts recognizing what is familiar, likely or expected. When it looks at an unseen face it often becomes upset, since it is not able to recognize it as familiar. The baby has learned its own representation of *normal* and its now using it as a reference to evaluate unexpected and unseen observations of the world. Just by looking at it, without being told of whether or not it knows someone, it is able to take a reaction. The intuition behind this naïve example frames the principles of unsupervised anomaly detection, which will be the main focus of this Thesis.

Since the dataset that motivates this Thesis is fully unlabelled, this Thesis aims to contribute to improve unsupervised anomaly detection.

1.5 Objectives & Requirements

This thesis aims to design and develop a general framework for anomaly detection in time series data. The proposed methodology should be generic, so that it can be applied to any kind of time series (univariate and multivariate, predictable and unpredictable, periodic and aperiodic) and, desirably, to other kinds of sequential data, such as text and videos. Furthermore, it should be unsupervised and not require anomaly labels. Since time series are sequences, the proposed approach should consider the temporal dependencies intrinsic to time series data. It should also be scalable to large-scale datasets and computationally efficient to allow fast and real-time detection. Finally, it should introduce a novel framework for anomaly detection in time series data and, thus, add a new contribution relatively to previous works.

Developing an approach such that all the aforementioned requirements are fulfilled is both the objective and the major challenge of this Master Thesis.

1.6 Thesis Outline

This Thesis is organized in five chapters. In Chapter 2 is reviewed the relevant background on deep learning (including autoencoders, recurrent neural networks, sequence to sequence models and attention mechanisms), are explained the principles behind autoencoder-based AD and is presented a brief overview of related recent work using this class of models. Chapter 3 introduces the proposed approach for anomaly detection, which relies on two fundamental stages: representation learning and detection. The representation learning model, based on an autoencoder, is described in detail and afterwards are proposed different detection strategies. Chapter 4 presents the results obtained using two datasets coming from different fields: energy and healthcare. These results are, then, analysed in detail and discussed. Finally, in Chapter 5, are presented the conclusions and insights provided by the results, are discussed several lines of future work and are summarized the main contributions of this Thesis. This Thesis also includes an appendix with the publications made during its execution.

Chapter 2

Background & Theory

The revolution will not be supervised.

Yann LeCun

2.1 Autoencoders

Autoencoders (AE) [Rumelhart, Hinton, and Williams, 1986; Bourlard and Kamp, 1988] are neural networks trained in an unsupervised fashion that aim to reconstruct their input. They consist of two parts: an *encoder* and a *decoder*. The encoder is a function f that maps input data $\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}}$ to a latent code/representation $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$. It has the form:

$$\mathbf{z} = f(\mathbf{x}) = s_f\left(\mathbf{W}\mathbf{x} + \mathbf{b}_\mathbf{z}\right) \tag{2.1}$$

where s_f denotes an activation function (often non-linear), $\mathbf{W} \in \mathbb{R}^{d_\mathbf{z} \times d_\mathbf{x}}$ is a weight matrix and $\mathbf{b}_\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$ is a bias vector.
The decoder is function g that maps back from latent code to input space (reconstruction).

$$\hat{\mathbf{x}} = g(\mathbf{z}) = g(f(\mathbf{x})) = s_g\left(\mathbf{W}'\mathbf{z} + \mathbf{b}_\mathbf{x}\right) \tag{2.2}$$

where s_g is the activation function of the decoder, $\mathbf{W}' \in \mathbb{R}^{d_\mathbf{x} \times d_\mathbf{z}}$ is a weight matrix and $b_\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}}$ is a bias vector. Sometimes, the weight matrix of the decoder is the transpose of the encoder weight matrix, i.e. $\mathbf{W}' = \mathbf{W}^\top$. In that case the autoencoder has *tied weights*.

The autoencoders training procedure consists of finding the set of parameters $\theta = (\mathbf{W}, \mathbf{b}_\mathbf{z}, \mathbf{b}_\mathbf{x})$ that minimize a loss function, $L(\mathbf{x}, g(f(\mathbf{x})))$, which measures the quality of the reconstructions and, thus, by making the output reconstruction $\hat{\mathbf{x}}$ as close as possible to the original input $\mathbf{x}$. A typical choice for the loss is the mean squared error.

$$L(\mathbf{x}, g(f(\mathbf{x}))) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{2.3}$$

Autoencoders can be under-complete, i.e. their latent code $\mathbf{z}$ has a lower dimensionality than the input space $\mathbf{x}$ and, in that case, they are forced to learn a compressed representation of the data. In this framework, an autoencoder can be used for dimensionality reduction (DR) tasks. In fact, if the autoencoder has just one hidden layer and if the functions are linear and the loss is the mean squared error, an autoencoder is provably equivalent to Principal Component Analysis (PCA), while the weights to the K hidden units will span the same subspace as the first K principal components of the data [Murphy, 2012; Goodfellow *et al.*, 2016]. Furthermore, if the activation functions are non-linear, autoencoders can find non-linear representations of the data and, therefore, they are a powerful generalization of PCA that has experimentally demonstrated impressive results in the past [Hinton and Salakhutdinov, 2006].

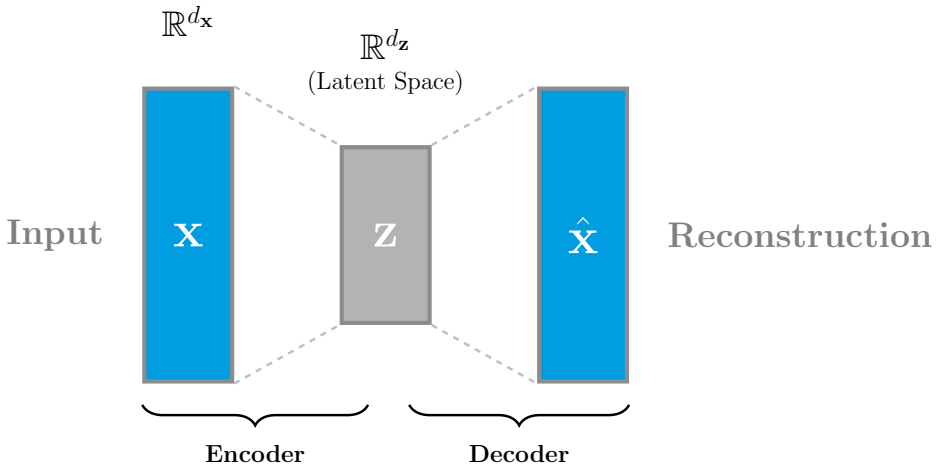


Figure 2.1: Illustration of an autoencoder network.

2.1.1 Denoising Autoencoders

Denoising autoencoders (DAEs) [Vincent *et al.*, 2008; Bengio *et al.*, 2013] learn to reconstruct an input data point, $\mathbf{x}$, from a corrupted version of it, $\tilde{\mathbf{x}}$. The process starts by corrupting the initial input $\mathbf{x}$ into $\tilde{\mathbf{x}}$ by means of a stochastic mapping $p(\tilde{\mathbf{x}}|\mathbf{x})$. The corrupted input in then mapped to a hidden representation/code similarly to a conventional autoencoder, $\mathbf{z} = f(\tilde{\mathbf{x}})$, from which the reconstruction is derived, $\hat{\mathbf{x}} = g(\mathbf{z})$. The training objective usually consists on minimizing a loss function $L(\mathbf{x}, g(f(\tilde{\mathbf{x}})))$.

The corruption process at the input level typically consists on adding Gaussian noise to the inputs, as in equation 2.4, or on randomly setting a fraction of the inputs to zero (zero-masking noise).

$$\tilde{\mathbf{x}} \sim p(\tilde{\mathbf{x}}|\mathbf{x}), \qquad p(\tilde{\mathbf{x}}|\mathbf{x}) = \text{Normal}(\mathbf{x}|\mathbf{0}, \boldsymbol{\sigma}_\mathbf{n}^2\mathbf{I}) \tag{2.4}$$

The denoising autoencoding procedure allows the autoencoder to be robust to data with white noise and to learn only the meaningful patterns of the data, while learning useful and expressive feature representations.

2.1.2 Sparse Autoencoders

The sparse autoencoder is an autoencoder whose loss function includes a sparsity penalty (Ω) on the code layer. By doing so, the autoencoder training objective aims to minimize a reconstruction loss, while promoting sparsity in the code layer.

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{z}) \tag{2.5}$$

Sparse autoencoders are usually employed to learn features for another task, such as supervised classification, that depends on these (sparse) features. An autoencoder that has been regularized to be sparse is likely to respond to unique statistical features of the dataset it has been trained on, rather than simply acting as an identity function [Goodfellow *et al.*, 2016, Ch. 14]. By doing so, a training objective that includes both a reconstruction term and a sparsity penalty leads to a model that has learned useful features as a by-product.

There are various forms of sparsity that can be adopted. For instance, the sparsity penalty can be defined as the ℓ_1-norm of the code $\mathbf{z}$, weighted by a parameter λ. In this case,

$$\Omega(\mathbf{z}) = \lambda \sum_{i=1}^{d_{\mathbf{z}}} |z_i| \tag{2.6}$$

2.1.3 Variational Autoencoders

Preliminaries

Consider the latent variable model of parameter θ with the following factorization:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) \tag{2.7}$$

where $\mathbf{x}$ is the observed variable and $\mathbf{z}$ is the latent variable. In the context of Bayesian learning, $p_\theta(\mathbf{z})$ is often referred to as the *prior distribution* over $\mathbf{z}$, since it is not conditioned on any observations, and $p_\theta(\mathbf{x}|\mathbf{z})$ is the likelihood of the observation $\mathbf{x}$, given the latent code $\mathbf{z}$. In this framework, the inference problem refers to the computation of the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$, that is to say, the conditional density of the latent variables given observations. The posterior distribution can be written as:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{x})} \tag{2.8}$$

where $p_\theta(\mathbf{x})$ is the marginal distribution over the observed variables (also referred to as *marginal likelihood* or *model evidence*). The model evidence is obtained by marginalizing out the latent variables from the joint probability distribution over both the observed variables and the latent variables, $p_\theta(\mathbf{x}, \mathbf{z})$, as in equation 2.9.

$$p_\theta(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{2.9}$$

One of the main difficulties in this kind of probabilistic graphical models is that the evidence integral is

intractable by analytical methods and does not have a closed form solution. For solving this problem can be adopted several approximate inference techniques, such as Variational Inference (VI) and Markov Chain Monte Carlo (MCMC). In the context of Variational Autoencoders, variational inference is often employed to solve the inference problem and it does so by turning it into an optimization one.

Variational inference aims to find a variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ of the intractable true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ by minimizing the Kullback-Leibler (KL) divergence between both. The KL-divergence ($\mathcal{D}_{\mathrm{KL}}$) is a measure of dissimilarity between distributions and, thus, the lower the divergence the higher the similarity between the distributions. If q and p are two continuous probability distribution, the KL-divergence is given by:

$$\mathcal{D}_{\mathrm{KL}}(q\|p) = \int_{\mathbf{z}} q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z})} d\mathbf{z} \tag{2.10}$$

The optimization problem aims to find the approximate posterior parameters ϕ such that the KL-divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$ is minimized:

$$q^*(\mathbf{z}) = \underset{\phi}{\operatorname{argmin}}\ \mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\big) \tag{2.11}$$

Overview

The Variational Autoencoder (VAE) [Kingma and Welling, 2013; Rezende *et al.*, 2014] is a deep generative model rooted in Bayesian inference that constrains the code $\mathbf{z}$ of the autoencoder to be a random variable distributed according to a prior distribution $p_\theta(\mathbf{z})$. The VAE can be analysed from a graphical model perspective, with the representation illustrated in Figure 2.2.

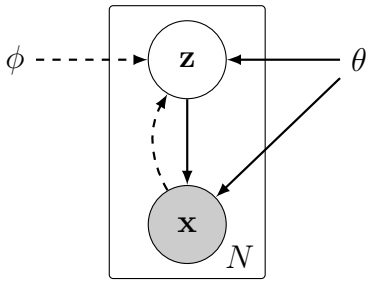


Figure 2.2: The VAE as a (directed) graphical model.

Since the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable for a continuous latent space $\mathbf{z}$, the previously described Variational Inference technique is often used to solve this intractable posterior inference problem in a tractable way. Therefore, it is introduced a variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$ of the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$, which if often called the *encoder* or *recognition/inference* network. The parameters of the encoder, ϕ, are called the *variational parameters* and can be derived using deep neural networks. The prior distribution over the latent variables, $p_\theta(\mathbf{z})$, is often a standard isotropic Normal (diagonal co-variance matrix, with $\mathbf{\Sigma}_p = \mathbf{I}$). In that case, the encoder is designed to output the mean, $\boldsymbol{\mu}_\mathbf{z}$, and the standard deviation, $\boldsymbol{\sigma}_\mathbf{z}$,

parameters of the approximate posterior distribution:

$$(\mu_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}}) = \text{Encoder}_{\phi}(\mathbf{x}) \tag{2.12}$$

$$q_{\phi}(\mathbf{z}|\mathbf{x}) = \text{Normal}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}}\mathbf{I}) \tag{2.13}$$

The decoder network (generative model), $p_{\theta}(\mathbf{x}|\mathbf{z})$, parametrizes the likelihood of data $\mathbf{x}$, given the latent code $\mathbf{z}$. The decoding distribution is usually a multivariate Normal or Bernoulli, depending on the type of data being continuous or binary, respectively.

Figure 2.3 shows a graphical representation of a variational autoencoder.

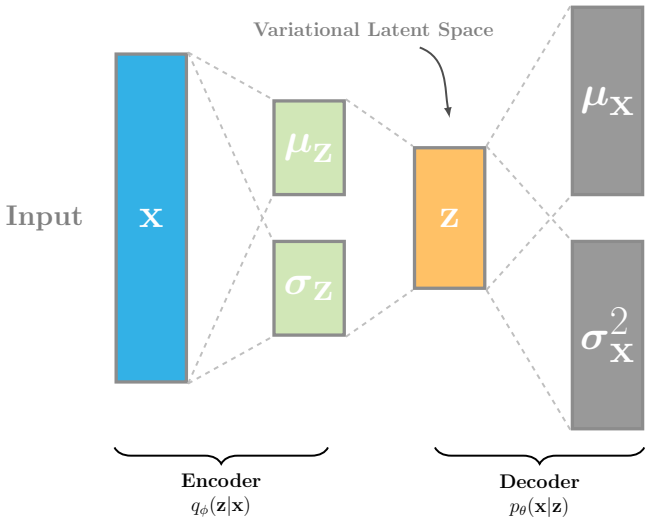


Figure 2.3: Graphical representation of a variational autoencoder.

Figure 2.4 illustrates the encoder and decoder mappings in a variational autoencoder.

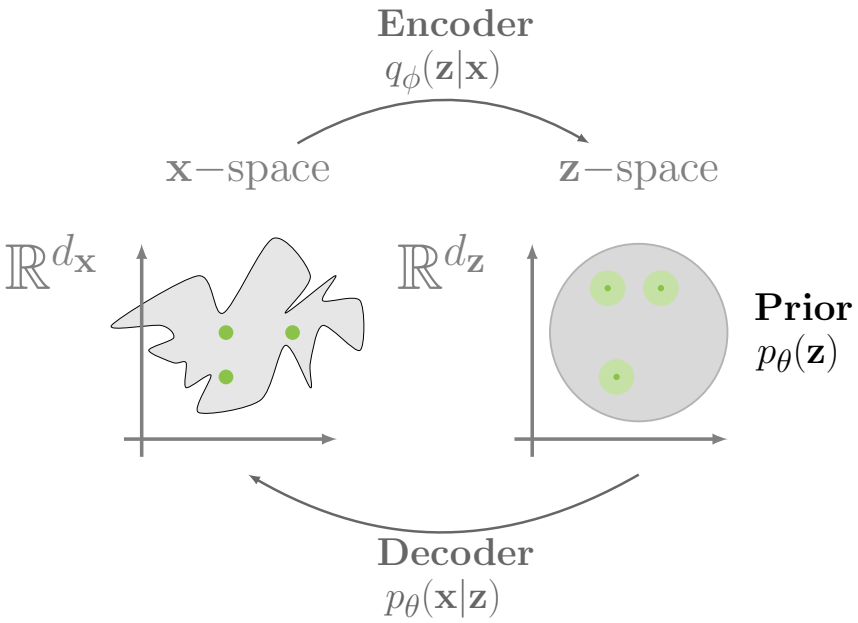


Figure 2.4: Encoder and decoder mappings between the input $\mathbf{x}$-space and the $\mathbf{z}$-space of representations.

Training Objective

The optimization objective of the VAE, similarly to other variational techniques, is the *evidence lower bound* (ELBO), also referred to as *variational lower bound*. Given an inference model $q_\phi(\mathbf{z}|\mathbf{x})$, the ELBO can be derived as follows:

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x})\big] \tag{2.14}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \text{ (Bayes Rule)} \tag{2.15}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \tag{2.16}$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})}\right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \tag{2.17}$$

$$= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\big)}_{=\mathcal{L}_{\mathrm{ELBO}}(\theta,\phi;\mathbf{x})} + \underbrace{\mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\big)}_{\geq 0} \tag{2.18}$$

In equation 2.18, the first term is the *Evidence Lower Bound* and the second term is the Kullback-Leibler divergence ($\mathcal{D}_{\mathrm{KL}}$) between the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$. Since the KL-divergence is always non-negative, the ELBO is a lower bound on the log-likelihood of the data.

$$\begin{aligned}\mathcal{L}_{\mathrm{ELBO}}(\theta,\phi;\mathbf{x}) &= \log p_\theta(\mathbf{x}) - \mathcal{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})\right) \\ &\leq \log p_\theta(\mathbf{x})\end{aligned} \tag{2.19}$$

The VAE training objective is to maximize the ELBO and, therefore, to minimize the loss function given by equation 2.20.

$$\mathcal{L}_{\mathrm{VAE}}(\theta,\phi;\mathbf{x}) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta\left(\mathbf{x}|\mathbf{z}\right)\big] + \mathcal{D}_{\mathrm{KL}}(q_\phi\left(\mathbf{z}|\mathbf{x}\right)\|p_\theta(\mathbf{z})) \tag{2.20}$$

The Reparameterization Trick

The VAE computational graph includes a probabilistic node that refers to the sampling process of the latent variable from the approximate posterior. However, when training the model with Stochastic Gradient Descent this leads to an issue, since it is not possible to differentiate the objective with respect to the variational parameters ϕ because the gradients can not be backpropagated through the latent variable $\mathbf{z}$. Therefore, Kingma and Welling [2013] and Rezende *et al.* [2014] proposed the reparametrization trick for overcoming this problem. This trick consists on sampling an auxiliary (and external) random variable $\boldsymbol{\epsilon}$ from a fixed distribution, $\mathrm{Normal}(\mathbf{0},\mathbf{I})$, and then apply a variable transformation as in equation 2.21.

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathrm{Normal}(\mathbf{0},\mathbf{I}) \tag{2.21}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the variational parameters derived from the encoder.

Figure 2.5 illustrates the computational graph behind the reparametrization trick.

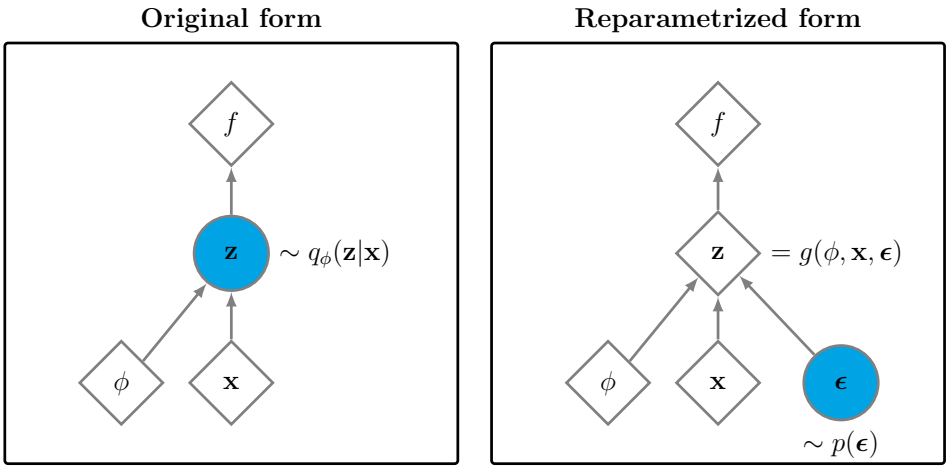


Figure 2.5: Representation of the reparameterization trick. The parameters ϕ of the approximate posterior affect the objective f by means of the latent variable $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. Since it is not possible to differentiate f with respect to ϕ because the gradients can not be backpropagated through the random variable $\mathbf{z}$, it is introduced an external random variable $\boldsymbol{\epsilon}$ sampled from a fixed distribution $p(\boldsymbol{\epsilon}) = \text{Normal}(\mathbf{0}, \mathbf{I})$ and then it is executed a simple variable transformation.

Optimization of the ELBO

The optimization of the evidence lower bound is performed using a stochastic optimization procedure proposed in the VAE original paper by Kingma and Welling [2013]. They called this algorithm Auto-Encoding Variational Bayes (AEVB).

Algorithm 1 Auto-Encoding Variational Bayes Algorithm

Input:
$\mathcal{X}$: Dataset
$q_\phi(\mathbf{z}|\mathbf{x})$: Inference/recognition Model
$p_\theta(\mathbf{x}, \mathbf{z})$: Generative Model
Output:
θ, ϕ: Learned parameters
$(\theta, \phi) \leftarrow$ Initialise parameters
while SGD not converged **do**
$\mathcal{M} \sim \mathcal{X}$ (Draw a random mini-batch of data)
$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ (Random noise for every point within the mini-batch $\mathcal{M}$)
Compute $\mathcal{L}(\theta, \phi; \mathcal{M}, \boldsymbol{\epsilon})$ and its gradients $\nabla\mathcal{L}(\theta, \phi; \mathcal{M}, \boldsymbol{\epsilon})$
Update θ and ϕ using a SGD optimizer
end while

The stochastic optimization procedure in the AEVB algorithm is two-fold, since the noise introduced by either the random choice of a mini-batch $\mathcal{M}$ and by the sampling step $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$.
Given a dataset $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ composed by N independent and identically distributed examples, the global evidence lower bound objective is the sum over the evidence lower bounds of all individual data points $\mathbf{x}^{(n)}$ within $\mathcal{X}$.

$$\mathcal{L}_{\text{ELBO}}(\mathcal{X}) = \sum_{\mathbf{x}^{(n)} \in \mathcal{X}} \mathcal{L}_{\text{ELBO}}(\mathbf{x}^{(n)}) \tag{2.22}$$

2.2 Recurrent Neural Networks

2.2.1 Overview

Feed-forward neural networks and several other machine learning models assume data is independent and identically distributed (i.i.d.) in time or in space. In many applications involving time series, text and videos, data is sequential and, therefore, this assumption does not hold.

Recurrent Neural Networks (RNNs) are powerful sequence learners that overcome this limitation by introducing memory into the network. They receive as input a sequence of $d_{\mathbf{x}}$-dimensional vectors $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ and process them one at a time. For each input vector $\mathbf{x}_t$, the recurrent neural network updates its memory and produces a hidden state $\mathbf{h}_t \in \mathbb{R}^{d_{\mathbf{h}}}$. This hidden state is a summary of the sequence of vectors seen by the network up to timestep t, i.e. $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t)$. The main feature of recurrent networks is the feedback connection between hidden units across time that stablish a recurrence mechanism. This recurrence mechanism defines how the hidden states $\mathbf{h}_t$ are updated. At each time step t, the hidden state $\mathbf{h}_t$ of a RNN in its simplest form ("vanilla" RNN) is updated based on the current input ($\mathbf{x}_t$) and the hidden state at the previous timestep ($\mathbf{h}_{t-1}$), as in equation 2.23.

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{b}) \tag{2.23}$$

In equation 2.23, f is typically a non-linear function such as the $\mathrm{sigmoid}$ or the $\tanh$. $\mathbf{U} \in \mathbb{R}^{d_{\mathbf{h}} \times d_{\mathbf{x}}}$ and $\mathbf{W} \in \mathbb{R}^{d_{\mathbf{h}} \times d_{\mathbf{h}}}$ are weight matrices to be learned that describe the input-to-hidden and the hidden-to-hidden connections and $\mathbf{b} \in \mathbb{R}^{d_{\mathbf{h}}}$ is a bias vector. Moreover, the initial hidden state $\mathbf{h}_0$ is often set to $\mathbf{0}$.

The RNN can produce an output vector at each timestep ($\hat{\mathbf{y}}_t \in \mathbb{R}^{d_{\mathbf{y}}}$).

$$\hat{\mathbf{y}}_t = g(\mathbf{V}\mathbf{h}_t + \mathbf{c}) = g(\mathbf{o}_t) \tag{2.24}$$

In equation 2.24, $\mathbf{V} \in \mathbb{R}^{d_{\mathbf{y}} \times d_{\mathbf{h}}}$ is a weight matrix that describes the hidden-to-output connections and $\mathbf{c} \in \mathbb{R}^{d_{\mathbf{y}}}$ is a bias vector. The function g used to produce the output $\hat{\mathbf{y}}_t$ depends on the application. For instance, in a classification problem, g is often the $\mathrm{softmax}$ function, which produces a probability distribution over classes. In a regression task, g can be a fully connected neural network.

All the parameters (weights and biases) are shared between all time steps.

2.2.2 Training

RNNs training is often executed using mini-batch Stochastic Gradient Descent (SGD) algorithms. These algorithms use a random subset of training examples (a *mini-batch*, $\mathcal{M}$) to compute the gradients and update the weights, one at a time. By considering a mini-batch of examples, training is more stable and consistent relatively to the "online" setting, which updates the gradients using only a single example, and more efficient than the full-batch setting, which needs to go through all examples before performing an update. Finally, the matrix-vector multiplications behind RNNs can be performed as matrix-matrix multiplications using mini-batches of samples and these can be efficiently deployed on GPUs. In a

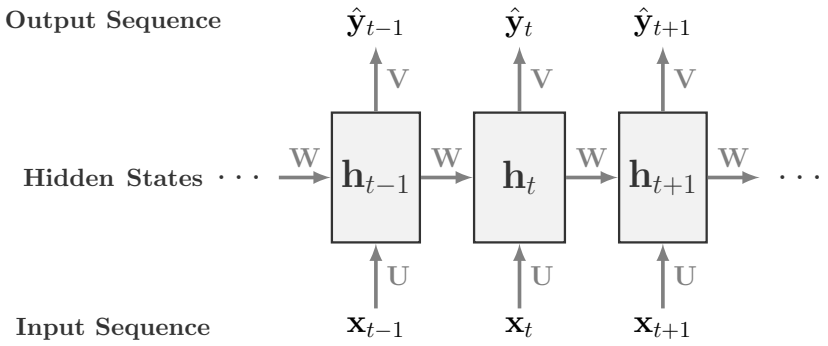


Figure 2.6: A "Vanilla" Recurrent Neural Network.

simplified fashion, the update rule for the weights is given by equation 2.25.

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla_{\mathbf{W}} \mathcal{L} \tag{2.25}$$

In equation 2.25, $\mathbf{W}$ denotes the model weights, η the learning rate (or step size) and $\nabla_{\mathbf{W}} \mathcal{L}$ the gradient of the loss function with respect to the model weights. The objective is to update the model weights $\mathbf{W}$ along the opposite direction of the gradient, such that the loss is minimized.
The update of the weights is performed during a pre-defined number of updates or until the loss $\mathcal{L}$ becomes lower than a threshold. Stochastic Gradient Descent might not find a global optimum, but if the learning rate is reduced during training it will find a local optimum. Several variants of SGD algorithms were proposed over time, such as *AMS-Grad* [Reddi *et al.*, 2018], *Adam* [Kingma and Ba, 2014], *RMS-Prop* [Tieleman and Hinton, 2012] and *AdaGrad* [Duchi *et al.*, 2011].

2.2.3 Backpropagation Through Time

RNNs are trained with special backpropagation algorithm called *Backpropagation Through Time* (BPTT). For computing the gradients and similarly to the backpropagation algorithm in feed-forward neural networks, BPTT makes use of the chain rule for differentiation.
The computation of the gradient with respect to the parameters $\mathbf{V}$ of the output layer, $\mathbf{o}_t$, is given by:

$$\nabla_{\mathbf{V}} \mathcal{L}_t = (\nabla_{\hat{\mathbf{y}}_t} \mathcal{L}_t)\,(\nabla_{\mathbf{V}} \hat{\mathbf{y}}_t) \tag{2.26}$$

$$= (\nabla_{\hat{\mathbf{y}}_t} \mathcal{L}_t)\,(\nabla_{\mathbf{o}_t} \hat{\mathbf{y}}_t)\,(\nabla_{\mathbf{V}} \mathbf{o}_t) \tag{2.27}$$

The computation of the gradient with respect to the weights $\mathbf{W}$ of the recurrent layer is more tricky:

$$\nabla_{\mathbf{W}}\mathcal{L}_t = (\nabla_{\hat{\mathbf{y}}_t}\mathcal{L}_t)\,(\nabla_{\mathbf{W}}\hat{\mathbf{y}}_t) \tag{2.28}$$

$$= (\nabla_{\hat{\mathbf{y}}_t}\mathcal{L}_t)\,(\nabla_{\mathbf{o}_t}\hat{\mathbf{y}}_t)\,(\nabla_{\mathbf{W}}\mathbf{o}_t) \tag{2.29}$$

$$= (\nabla_{\hat{\mathbf{y}}_t}\mathcal{L}_t)\,(\nabla_{\mathbf{o}_t}\hat{\mathbf{y}}_t)\,(\nabla_{\mathbf{h}_t}\mathbf{o}_t)\,(\nabla_{\mathbf{W}}\mathbf{h}_t) \tag{2.30}$$

The problem is that the last term in equation 2.30, which is the gradient of the $\mathbf{h}_t$ with respect to $\mathbf{W}$ depends on $\mathbf{h}_{t-1}$ and so on. This recursion requires multiple applications of the chain rule in order to compute the gradient.

Finally, in BPTT, the total gradient is obtained by summing the contribution of the gradients for all timesteps t.

$$\nabla_{\mathbf{W}}\mathcal{L} = \sum_{t=1}^{T} \nabla_{\mathbf{W}}\mathcal{L}_t \tag{2.31}$$

By doing so, the gradient computation and the weight updates are performed based on the contributions of individual timesteps. This procedure clarifies that a recurrent neural network unrolled is analogous to a feed-forward neural network whose weights are shared over layers.

Furthermore, in practice, it is often used a variant of the BPTT algorithm called *Truncated Backpropagation Through Time* (TBPTT). TBPTT processes a given input sequence one timestep at a time, and every k_1 timesteps, it executes BPTT for k_2 timesteps. By doing so, the update of a parameter can be very efficient if k_2 is small and the hidden states have been exposed to many timesteps and so may contain useful information about the far past [Sutskever, 2013].

2.2.4 Why RNNs?

Recurrent Neural Networks have been applied to sequence modelling tasks and attained state of the art performance in applications such as speech recognition [Graves *et al.*, 2013]. However, the problem of modelling sequences has been tackled for decades using other models, whose most prominent one is the Hidden Markov Model (HMM). Therefore, a justification for using RNNs rather than other models like HMMs is required.

Recurrent Neural Networks and Hidden Markov Models are both powerful sequence learners that share some similarities in their architecture. For instance, either HMMs and RNNs have hidden states. However, in HMMs the state at a current timestep depends only on the previous state and, thus, they make a Markovian assumption on the temporal structure of the data that constrains the model effectiveness. On the other side, the hidden state of a RNN is shared (or distributed) over time and, therefore, it can capture temporal dependencies from a larger history of previous states, making RNNs have significantly richer memory and computational capacity. While it is possible to extend a HMM to consider a larger context window this procedure would grow the state space exponentially with the size of the window, making the model computationally inefficient for modelling long-range dependencies [Graves *et al.*, 2014].

Finally, RNNs are also proven to be Turing complete [Siegelmann and Sontag, 1991], that is to say, in general terms, any computation a computer can execute can be expressed by a RNN.

2.2.5 Long Short-Term Memory Networks

Despite the effectiveness of RNNs for modelling sequential data, they suffer from the vanishing gradient problem, that arises when the output at timestep t depends on inputs much earlier in time. Therefore, Long Short-Term Memory networks (LSTMs) [Hochreiter and Schmidhuber, 1997; Graves, 2013] were proposed to overcome this problem. They do so by means of a memory cell and three gates. The memory cell stores information about the input sequence $\mathbf{x}$ across timesteps. The information flow from and to the memory cell is controlled by *gates*. The gates are functions that control the proportion of the current input to include in the memory cell ($\mathbf{i}_t$), the proportion of the previous memory cell to forget ($\mathbf{f}_t$) and the information to output from the current memory cell ($\mathbf{o}_t$). The memory updates, at each timestep t, are computed as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{x}_t + \mathbf{b}_i) \tag{2.32}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{x}_t + \mathbf{b}_f) \tag{2.33}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{x}_t + \mathbf{b}_o) \tag{2.34}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{x}_t + \mathbf{b}_c) \tag{2.35}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.36}$$

In the previous equations, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ and $\mathbf{h}_t$ denote the input gate, the forget gate, the output gate, the memory cell and the hidden state, respectively. σ is often a smooth function such as the sigmoid. The other parameters are weight matrices to be learned. The initial hidden state and memory cell are also parameters to be learned or, instead, to be initialized with zeros.
The aforementioned vanishing gradient problem is prevented by the gates, in particular, by the forget gate. Figure 2.7 shows a schematic representation of a LSTM cell.

LSTMs can still not integrate information from future instants of time and, therefore, Bidirectional Long Short-Term Memory networks (Bi-LSTM) [Graves *et al.*, 2005] were proposed. Bi-LSTMs exploit the input sequence $\mathbf{x}$ in both directions by means of two LSTMs: one executes a forward pass and the other a backward pass. Hence, two hidden states ($\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$) are produced at each timestep t, one in each direction. These states act like a summary of the past and the future. The hidden states at similar timesteps are often aggregated into a unique vector $\mathbf{h}_t = \left[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t\right]$ that represents the whole context around timestep t, typically through concatenation.

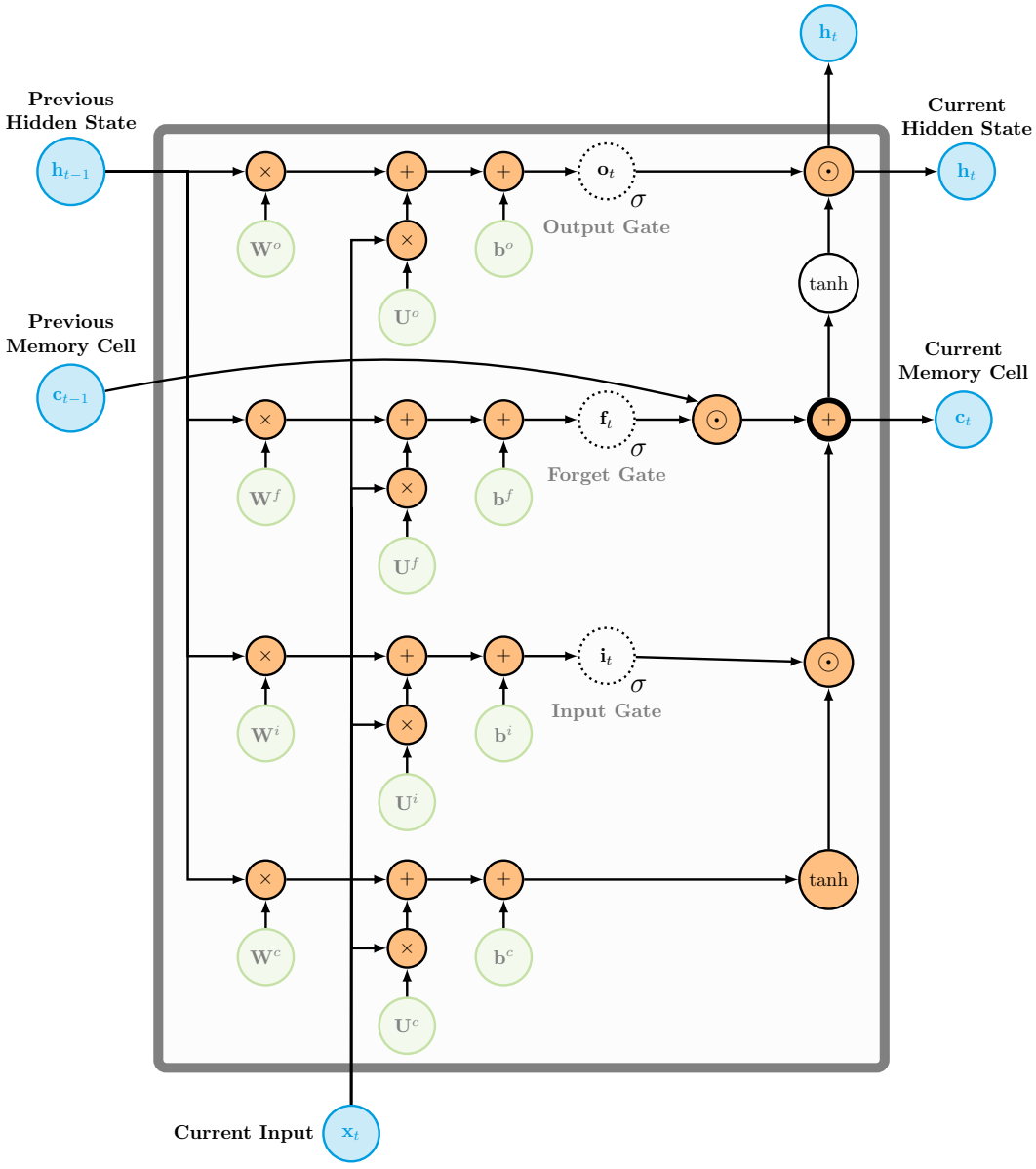


Figure 2.7: Internal representation of a Long Short-Term Memory Network.

2.3 Sequence to Sequence Models

Recurrent Neural Networks, described in section 2.2, are able to map sequences into fixed-length vectors. However, in many applications dealing with sequential data, such as machine translation, speech recognition and time series forecasting, it is useful to convert sequences into sequences. The sequence to sequence (Seq2Seq) learning framework is often linked with a class of encoder-decoder models that was developed precisely for solving this problem of mapping variable-length sequences into variable-length sequences using RNNs. Introduced for the first time by Cho *et al.* [2014] and shortly after by Sutskever *et al.* [2014], in the context of machine translation, these architectures attained state of the art performance in translation tasks and revolutionized the way machine translation is performed. These models operate as follows. The encoder is a recurrent neural network (e.g., LSTM) that reads a variable-length input sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{T_\mathbf{x}}) \in \mathbb{R}^{T_\mathbf{x} \times d_\mathbf{x}}$ and converts it into a fixed-length vector representation (or context vector), $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$, and the decoder is another recurrent neural network that takes this vector representation and converts it back into a variable-length sequence $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{T_\mathbf{y}}) \in \mathbb{R}^{T_\mathbf{y} \times d_\mathbf{y}}$. In the simplest encoder-decoder architecture, it is assumed that the final encoder state has captured all the relevant information of the input sequence and, thus, acts like a summary of it. The learned vector representation corresponds, in this scenario, to this final encoder hidden state, i.e. $\mathbf{z} = \mathbf{h}^\mathrm{e}_T$. Figure 2.8 shows a representation of a Seq2Seq model.

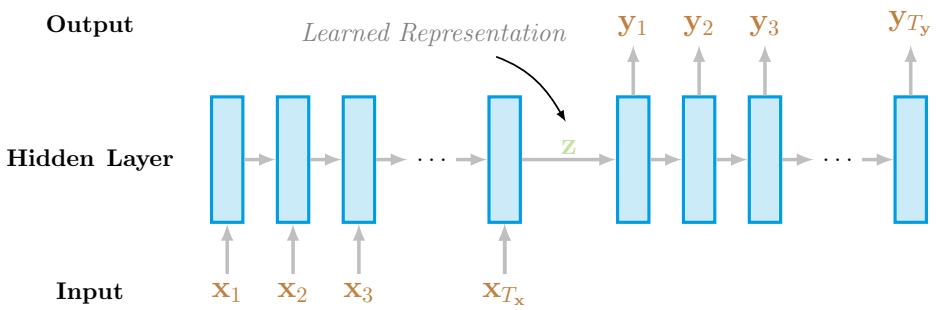


Figure 2.8: Example of an encoder-decoder sequence to sequence model. The encoder reads an input sequence, $\mathbf{x}$, and converts it into a fixed-size vector representation, $\mathbf{z}$, and the decoder takes this vector representation and transforms it back into another sequence, $\mathbf{y}$.

Seq2Seq models can operate under difference settings, depending on the problem being considered. A particular instance of a Seq2Seq model is the Seq2Seq Autoencoder [Srivastava *et al.*, 2015], in which the input and output sequences are aligned in time ($\mathbf{x} = \mathbf{y}$) and, thus, have equal lengths ($T_\mathbf{x} = T_\mathbf{y}$). The model learns to reconstruct the input sequence from an intermediate code (the vector representation $\mathbf{z}$) and, therefore, it resembles the conventional auto-encoding principle at the core of (feed-forward) autoencoders, now extended to tackle sequential data and its temporal dependencies using recurrent neural networks. At the same time, the Seq2Seq Autoencoder is a straightforward way of reducing datasets of sequences, $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^N$, $\mathbf{x}^{(n)} \in \mathbb{R}^{d_\mathbf{x}}$, into data points living in a latent space $\mathcal{Z} = \{\mathbf{z}^{(n)}\}_{n=1}^N$, $\mathbf{z}^{(n)} \in \mathbb{R}^{d_\mathbf{z}}$, often with a lower dimensionality ($d_\mathbf{z} < d_\mathbf{x}$).

2.4 Attention Mechanisms

The idea of integrating *Attention* in neural network models is partially inspired by the human attention system that has the ability of selecting stimulus during the early stages of processing based on elementary stimulus features [Hübner *et al.*, 2010]. An interesting example is the human visual system that can selectively focus its attention on parts of the visual space in order to acquire information when and where it is required and to build its own representation of the scene.

Sequence to Sequence models have their weakness in tackling long sequences (e.g., long time series), mainly because the intermediate fixed-length vector representation does not have enough capacity to capture information from the entire input sequence, $\mathbf{x}$. In other words, longer sequences need to be encoded into the same fixed-length vector representation or context vector.

Rooted in the mechanism behind the human attention system, Attention Mechanisms (AM) were proposed to overcome this limitation by allowing the decoder to selectively *attend* to relevant encoded hidden states.

Several attention models were proposed in the past few years [Bahdanau *et al.*, 2014; Luong *et al.*, 2015] and, in general, they operate as follows. At each timestep t, during decoding, the attention model computes a context vector $\mathbf{c}_t$ obtained by a weighted sum of the encoder hidden states. The weights of the sum, a_{ij}, are computed by a score function that measures the similarity between the currently decoded hidden state, $\mathbf{h}_t^{\mathrm{d}}$, and the encoded hidden states $\mathbf{h}^{\mathrm{e}} = (\mathbf{h}_1^{\mathrm{e}}, \mathbf{h}_2^{\mathrm{e}}, ..., \mathbf{h}_T^{\mathrm{e}})$. Afterwards, these scores are normalized using the $\mathrm{softmax}$ function, so that they sum to 1 along the second dimension. The computation of the weights and context vectors can be described as follows:

$$a_{ti} = \frac{\exp\left(\mathrm{score}(\mathbf{h}_t^{\mathrm{d}}, \mathbf{h}_i^{\mathrm{e}})\right)}{\sum_{j=1}^{T} \exp\left(\mathrm{score}(\mathbf{h}_t^{\mathrm{d}}, \mathbf{h}_j^{\mathrm{e}})\right)} \tag{2.37}$$

$$\mathbf{c}_t = \sum_{j=1}^{T} a_{tj} \mathbf{h}_j^{\mathrm{e}} \tag{2.38}$$

The score can be computed using, for instance, the following similarity functions [Luong *et al.*, 2015]:

$$\mathrm{score}(\mathbf{h}_t^{\mathrm{d}}, \mathbf{h}_i^{\mathrm{e}}) = \begin{cases} \left(\mathbf{h}_t^{\mathrm{d}}\right)^\top \mathbf{h}_i^{\mathrm{e}} & \textit{dot-product} \\ \left(\mathbf{h}_t^{\mathrm{d}}\right)^\top \mathbf{W}_a \mathbf{h}_i^{\mathrm{e}} & \textit{general} \end{cases} \tag{2.39}$$

Figure 2.9 illustrates an attention mechanism.

Even though Attention was developed mainly in the framework of Natural Language Processing (NLP) tasks involving text data, it can be applied to other problems dealing with other types of data such as time series and videos. Attention mechanisms have shown an impressive success in a variety of applications such as machine translation [Bahdanau *et al.*, 2014], image classification [Wang *et al.*, 2017], speech recognition [Chorowski *et al.*, 2015], pose estimation [Parisotto *et al.*, 2018], sentence summarization [Rush *et al.*, 2015] and image captioning [Xu *et al.*, 2015]. In fact, *attention* is a natural extension of approaches based on Seq2Seq models for any kind of sequential data.

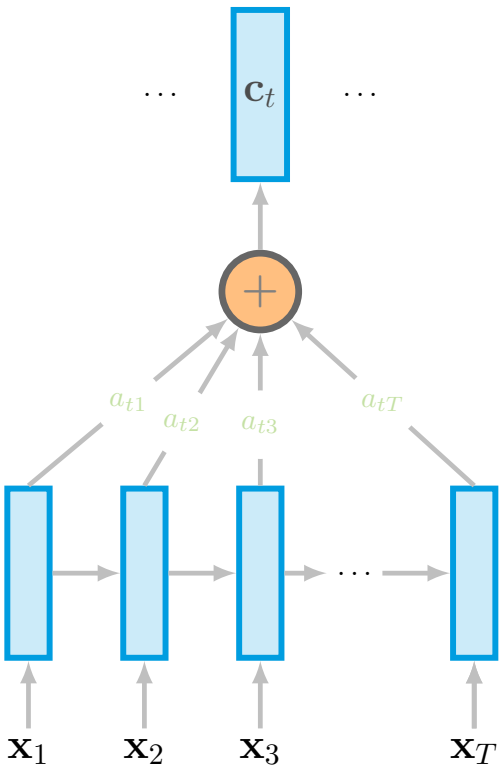


Figure 2.9: Example of an Attention Mechanism. The encoder processes an input sequence $\mathbf{x}$. At each decoding timestep t, the attention model computes a context vector, $\mathbf{c}_t$, as a weighted sum of all the encoded hidden states.

2.5 Autoencoder-based Anomaly Detection

The application of autoencoders in anomaly detection tasks has increased significantly over the past few years. Such interest has came together with the increasing trend of adopting unsupervised learning approaches that are foreseen to play an important role in the future of machine learning [LeCun, Bengio, and Hinton, 2015].

The main idea behind autoencoder-based anomaly detection is to focus on what is normal, rather than modelling what is anomalous. The autoencoder is trained to reconstruct data with normal pattern (e.g., normal time series) by minimizing a loss function that measures the quality of the reconstructions. After training, the model is able to reconstruct well data with normal pattern, while it fails to reconstruct anomalous data, since it never saw them during training. The detection is performed using the reconstruction metrics as anomaly score (e.g., reconstruction error) or using the latent space representations, by considering the low-dimensional manifold of normal data as a reference to evaluate unseen observations.

This approach has several advantages over other methodologies based on supervised learning models that try to classify an anomaly within a set of pre-identified ones. First, in several applications of interest such as fault detection, fraud detection or cyber-security, new anomalies might appear (for instance, new attacks or new types of fraud can emerge). By learning what is normal, the model is ready to even detect data with (anomalous) patterns never seen during training.

2.5.1 Related Work

The related work on anomaly detection using five classes of approaches was previously described in section 1.3. However, since the proposed approach will be based on an autoencoder architecture it is important to describe in detail the relevant recent work on AD using this methodology.

The work on anomaly detection in time series data has increased significantly over the past few years and has benefited from the progress made in Deep Learning. In particular, Seq2Seq and Autoencoder models have been more and more applied to anomaly detection tasks in sequential data, mainly due to the lack of labelled datasets in the context of real applications. Using this framework, Malhotra *et al.* [2015] proposed a prediction-based approach based on LSTMs and used the distribution of the prediction errors for computing an anomaly score. However, this prediction-based approach is not able to predict time series affected by external changes. Later on, reconstruction-based approaches were proposed to overcome this limitation, such as Malhotra *et al.* [2017], that instead of predicting future observations try to reconstruct the input sequence and, then, use the reconstruction errors as anomaly scores.

After the introduction of the Variational Autoencoder by Kingma and Welling [2013], An and Cho [2015] proposed an anomaly detection approach based on a (feed-forward) VAE and introduced a novel probabilistic anomaly score that takes into account the variability of the data (the *reconstruction probability*). Bayer and Osendorfer [2014] used variational inference and recurrent neural networks to model time series data and introduced Stochastic Recurrent Networks (STORNs), that were subsequently applied to anomaly detection in robot time series data [Sölch, 2015; Sölch *et al.*, 2016]. Recently, Park *et al.* [2017] applied a LSTM-based VAE for anomaly detection in robot assisted feeding data and introduced a progress-based prior for the latent variables, $\mathbf{z}$. Finally, Xu *et al.* [2018] applied a VAE to find anomalies in seasonal Key Performance Indicators (KPIs) time series and provided a theoretical explanation for VAE-based anomaly detection. These works summarize the recent approaches proposed for anomaly detection in sequential data using autoencoders and sequence to sequence models.

Chapter 3

Proposed Approach

> Our machines are dumb and we are just trying to make them less dumb.
>
> *Yoshua Bengio*

This chapter presents the proposed approach, which consists of two fundamental stages: representation learning (section 3.1) and detection (section 3.2).

Given a dataset of N independent and identically distributed (i.i.d.) sequences $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$, $\mathbf{x}^{(n)} = \left(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ..., \mathbf{x}_T^{(n)}\right) \in \mathbb{R}^{T \times d_{\mathbf{x}}}$ (e.g., a dataset of N $d_{\mathbf{x}}$-dimensional time series with T timesteps), the representation learning task aims to learn a set of expressive vector representations $\mathcal{Z} = \{\mathbf{z}^{(n)}\}_{n=1}^{N}$, while providing reconstructions of the input data obtained from $\mathbf{z}$.

The second component is the detection task. Detection refers to the problem of finding whether a given example $\mathbf{x}^{(n)}$ is normal or anomalous. Since the representation learning model provides either representations of data and the parameters of their reconstructions, detection can be executed over two spaces: the $\mathbf{z}$-space of latent representations and the input $\mathbf{x}$-space.

It is important to stress, beforehand, that anomaly detection in time series data, without loss of generality for other types of sequential data, can be executed at two different levels of detail: either by providing an anomaly score for each observation $\mathbf{x}_t^{(n)} \in \mathbb{R}^{d_{\mathbf{x}}}$ within a sequence $\mathbf{x}^{(n)}$ or, at a higher level, by assigning a single anomaly score to the whole sequence $\mathbf{x}^{(n)}$. The main difference between both settings is that the latter does not provide information regarding the temporal location of the anomaly in the sequence.

3.1 Representation Learning

3.1.1 Overview

At the heart of this Thesis lies a representation learning model. Learning good representations of data allows to understand the data in meaningful ways and makes it possible to execute further tasks using those representations. In this Thesis, the representations are learned for anomaly detection. First of all, in such framework, it is important to define what is, indeed, a good representation of the data. Good representations are those that capture posterior beliefs about explanatory causes of data, that disentangle their underlying factors of variation [Bengio *et al.*, 2012].

3.1.2 Model

The representation learning model in this Thesis is based on a Variational Recurrent Autoencoder (VRAE): a variational autoencoder whose encoder and decoder are recurrent neural networks. All the components/layers of the proposed model are described in detail in this part.

Input Layer

The model receives as input a sequence of observations $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$. Then, it is applied a denoising autoencoding criterion, similarly to the idea of the denoising autoencoder (previously described in subsection 2.1.1), but now extended to the variational auto-encoding framework [Bengio *et al.*, 2015b]. This is performed through a corruption process, $p(\tilde{\mathbf{x}}|\mathbf{x})$, with additive Gaussian noise (zero-mean).

$$\tilde{\mathbf{x}} \sim p(\tilde{\mathbf{x}}|\mathbf{x}), \qquad p(\tilde{\mathbf{x}}|\mathbf{x}) = \mathrm{Normal}(\mathbf{x}|\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{n}}^2\mathbf{I}) \tag{3.1}$$

By doing so, the autoencoder is forced to learn how to reconstruct the clean version of the inputs, $\mathbf{x}$, from the corrupted one, $\tilde{\mathbf{x}}$. Since it is a regularization technique, this phase is only active at training time.

Encoder

The encoder is parametrized using a Bidirectional Long-Short Term Memory network with $\tanh$ activation that produces a sequence of hidden states in both directions, forward $\rightarrow$ and backward $\leftarrow$. The final encoder hidden states of both passes are concatenated with each other in order to produce a unique vector $\mathbf{h}_T^{\mathrm{e}} = \left[\overrightarrow{\mathbf{h}}_T^{\mathrm{e}}; \overleftarrow{\mathbf{h}}_T^{\mathrm{e}}\right]$.

Variational Layer

The prior distribution over the latent variables, $p_\theta(\mathbf{z})$, is defined as an isotropic Normal distribution, i.e. $p_\theta(\mathbf{z}) = \mathrm{Normal}(\mathbf{0}, \mathbf{I})$. The *variational parameters* of the approximate posterior $\tilde{q}_\phi(\mathbf{z}|\mathbf{x})$, the mean $\boldsymbol{\mu}_{\mathbf{z}}$ and the standard deviation $\boldsymbol{\sigma}_{\mathbf{z}}$, are derived from the final encoder hidden state, $\mathbf{h}_T^{\mathrm{e}}$, using two fully connected layers with Linear and $\mathrm{SoftPlus}$ activations, respectively. The $\mathrm{SoftPlus}$ function is adopted to ensure that the standard deviation is parametrized as non-negative and using a smooth function. Since $p(\tilde{\mathbf{x}}|\mathbf{x})$ (input corruption process) and $q_\phi(\mathbf{z}|\mathbf{x})$ both have Normal distributions, the approximate posterior given

a corruption distribution around $\mathbf{x}$, denoted $\tilde{q}_\phi(\mathbf{z}|\mathbf{x})$, can be represented as a mixture of Gaussians as noted by Bengio *et al.* [2015b]. However, for computational convenience and following the approach of Park *et al.* [2017] a single Gaussian is employed, i.e. $\tilde{q}_\phi(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$. The latent variables are then obtained by sampling from the approximate posterior, $\mathbf{z} \sim \mathrm{Normal}(\boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\sigma}^2_\mathbf{z}\mathbf{I})$, using the reparametrization trick,

$$\mathbf{z} = \boldsymbol{\mu}_\mathbf{z} + \boldsymbol{\sigma}_\mathbf{z} \odot \boldsymbol{\epsilon} \tag{3.2}$$

where $\boldsymbol{\epsilon} \sim \mathrm{Normal}(\mathbf{0}, \mathbf{I})$ is an auxiliary noise variable and $\odot$ represents an element-wise product.

Attention

The model is integrated with a novel attention mechanism, specially designed in the context of this Thesis, called Variational Self-Attention Mechanism (VSAM). The motivation behind such model is described as follows. Previous attention models are able to deal with variable-length output sequences and compute the context vectors $\mathbf{c}_t$ dynamically, at each timestep, during the decoding process. VSAM makes *attention* more suited and efficient for the particular kind of model employed in this work - a Variational Seq2Seq Autoencoder - whose input and output sequences are similar and, in particular, have the same length. The proposed mechanism combines two different ideas recently introduced: the variational approach to attention [Bahuleyan *et al.*, 2017] and the self-attention (or intra-attention) model employed in *Transformer* [Vaswani *et al.*, 2017] (a very successful model developed for Natural Language Processing (NLP) tasks based on self-attention). In general, a simple self-attention model receives as input a sequence of vectors and outputs a sequence of context vectors $\mathbf{c}_t$ with the same length (T), each one of them computed as a weighted sum of all the input vectors. In detail, the proposed mechanism works as follows. First, the relevance of every pair of encoded hidden states $\mathbf{h}^\mathrm{e}_i$ and $\mathbf{h}^\mathrm{e}_j$ is scored (3.3) using the *scaled dot-product* similarity. The use of the dot-product as relevance measure makes the self-attention model more efficient than previous attention mechanisms that need to learn a similarity matrix.

$$s_{ij} = \mathrm{score}(\mathbf{h}^\mathrm{e}_i, \mathbf{h}^\mathrm{e}_j) = \frac{(\mathbf{h}^\mathrm{e}_i)^\top \mathbf{h}^\mathrm{e}_j}{\sqrt{d_{\mathbf{h}^\mathrm{e}}}} \tag{3.3}$$

In equation 3.3, $d_{\mathbf{h}^\mathrm{e}}$ is the size of the encoder Bi-LSTM hidden state. Afterwards, the attention weights a_{ij} are computed by normalizing the scores over the second dimension, as in equation 3.4, where $\mathbf{a}_t = (a_{t1}, a_{t2}, ..., a_{tT})$. This normalisation ensures that, for each timestep t, $\sum_{j=1}^{T} a_{tj} = 1$.

$$\mathbf{a}_t = \mathrm{softmax}(\mathbf{s}_t) \tag{3.4}$$

Finally, for deriving the new context-aware vector representations, $\mathbf{c}_t$, a variational approach is adopted. This choice is motivated by the *bypassing phenomenon* pointed out by Bahuleyan *et al.* [2017]. In fact, if the decoder has a direct and deterministic access to the encoder hidden states through attention, the latent code $\mathbf{z}$ may not be forced to learn expressive representations, since the self-attention mechanism could bypass most of the information to the decoder. This problem can be solved by applying to the context vectors $\mathbf{c}_t$ the same constraint applied to the latent variables of the VAE, that is to say, to model $\mathbf{c}_t$, $\forall_{t=(1,2,...,T)}$, as random variables. To do so, firstly, deterministic context vectors are computed in

a similar fashion to a conventional self-attention model, $\mathbf{c}_t^{\text{det}} = \sum_{j=1}^{T} a_{tj}\mathbf{h}_j$ and, secondly, they are transformed using another layer, similarly to Bahuleyan *et al.* [2017]. The prior distribution over the context vectors is defined as a standard Normal, $p(\mathbf{c}_t) = \text{Normal}(\mathbf{0}, \mathbf{I})$, and the variational parameters of the approximate posterior of the context vectors, $\tilde{q}_\phi^a(\mathbf{c}_t|\mathbf{x})$, mean $\boldsymbol{\mu}_{\mathbf{c}_t}$ and standard deviation $\boldsymbol{\sigma}_{\mathbf{c}_t}$, are derived in similar fashion to the latent variables $\mathbf{z}$ using two fully connected layers, including the dimensionality ($d_{\mathbf{c}_t} = d_{\mathbf{z}}$). The final context vectors are sampled from the approximate posterior, $\mathbf{c}_t \sim \text{Normal}(\boldsymbol{\mu}_{\mathbf{c}_t}, \boldsymbol{\sigma}_{\mathbf{c}_t}^2\mathbf{I})$. Figure 3.1 illustrates the proposed VSAM.

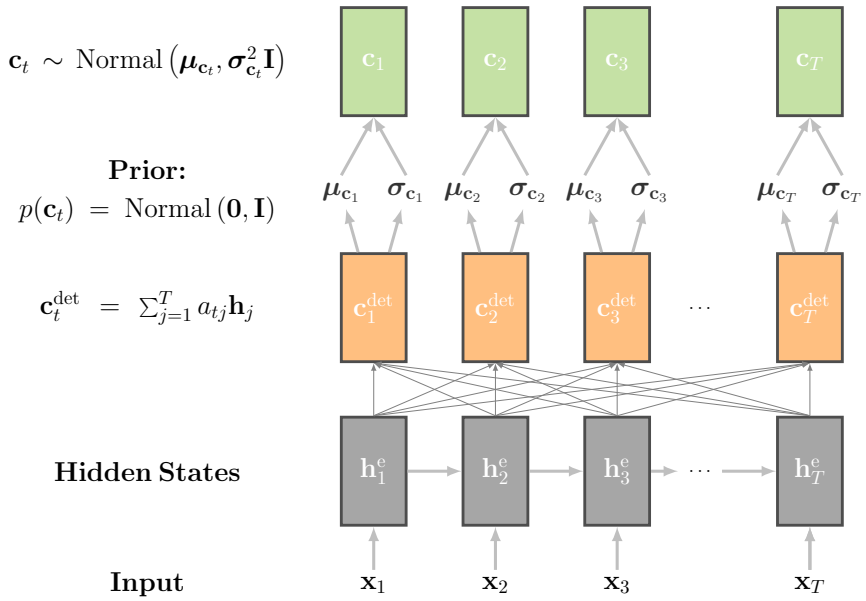


Figure 3.1: Illustration of the proposed Variational Self-Attention Mechanism (VSAM).

Decoder

The decoder is also a Bi-LSTM with $\tanh$ activation that receives, at each timestep t, a latent representation $\mathbf{z}$, shared across timesteps, and a context vector $\mathbf{c}_t$. Unlike other works that use a Normal distribution for $p_\theta(\mathbf{x}_t|\mathbf{z})$, in this Thesis is used a Laplace distribution with parameters $\boldsymbol{\mu}_{\mathbf{x}_t}$ and $\boldsymbol{b}_{\mathbf{x}_t}$. The practical implication of this choice is that the training objective aims to minimize an ℓ_1 reconstruction loss $\propto \|\mathbf{x}_t - \boldsymbol{\mu}_{\mathbf{x}_t}\|_1$ rather than an ℓ_2 reconstruction loss $\propto \|\mathbf{x}_t - \boldsymbol{\mu}_{\mathbf{x}_t}\|_2^2$. The minimization of ℓ_1-norm promotes sparse reconstruction errors.

The outputs of the decoder are the parameters of the reconstructed distribution of the input sequence of observations, mean $\boldsymbol{\mu}_{\mathbf{x}_t}$ and diversity $\boldsymbol{b}_{\mathbf{x}_t}$. These parameters are derived from the decoder hidden states using two fully connected layers with Linear and SoftPlus activations, respectively.

Activations

The activation functions employed in the proposed model are represented in Figure 3.2.

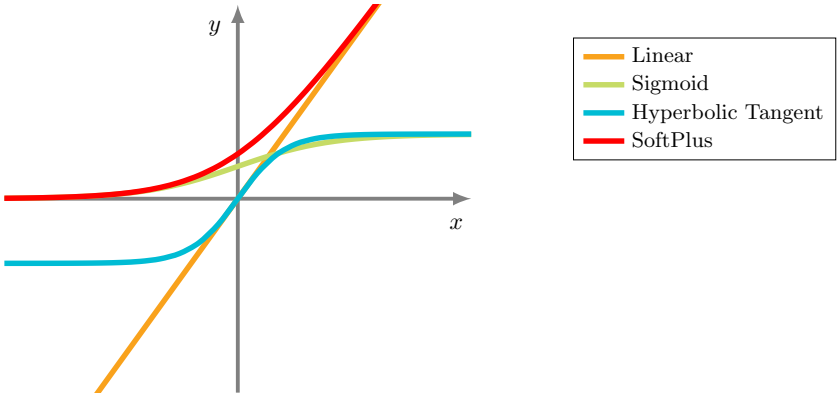


Figure 3.2: Activation Functions. Linear is used for deriving the expectation of the latent variables and the outputs; $\mathrm{Sigmoid}$ is used internally in the LSTMs; $\mathrm{Hyperbolic\ Tangent}$ is applied to the encoder and decoder Bi-LSTM layer; $\mathrm{SoftPlus}$ is used in the variance/diversity layers.

Loss Function

The loss for a particular sequence $\mathbf{x}^{(n)}$ is given by:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}^{(n)}) = &- \mathbb{E}_{\mathbf{z}\sim\tilde{q}_\phi(\mathbf{z}|\mathbf{x}^{(n)}),\ \mathbf{c}_t\sim\tilde{q}^a_\phi(\mathbf{c}_t|\mathbf{x}^{(n)})}\Big[\log p_\theta\big(\mathbf{x}^{(n)}|\mathbf{z},\mathbf{c}\big)\Big] \\ &+ \lambda_{\mathrm{KL}}\Bigg[\mathcal{D}_{\mathrm{KL}}\Big(\tilde{q}_\phi(\mathbf{z}|\mathbf{x}^{(n)})\|p_\theta(\mathbf{z})\Big) + \eta\sum_{t=1}^{T}\mathcal{D}_{\mathrm{KL}}\Big(\tilde{q}^a_\phi(\mathbf{c}_t|\mathbf{x}^{(n)})\|p_\theta(\mathbf{c}_t)\Big)\Bigg]\end{aligned} \tag{3.5}$$

The expectation above can be approximated by Monte Carlo sampling by taking L samples from the approximate posterior of the latent variables $\mathbf{z}$ and from the approximate posterior of the context vectors $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_T)$. In equation 3.5, the parameter λ_{KL} weights the reconstruction and KL losses and the parameter η balances the attention KL loss and the latent space KL loss.

Given a training dataset $\mathcal{X}_{\mathrm{train}} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ with N i.i.d. examples, the total loss is the sum of the losses for all data points $\mathbf{x}^{(n)}$.

$$\mathcal{L}(\theta, \phi; \mathcal{X}_{\mathrm{train}}) = \sum_{n=1}^{N}\mathcal{L}(\theta, \phi; \mathbf{x}^{(n)}) \tag{3.6}$$

The log-likelihood of a sequence $\mathbf{x}^{(n)} = \left(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ..., \mathbf{x}_T^{(n)}\right)$ given a latent code $\mathbf{z}$ decomposes over timesteps and can be written as in equation 3.7.

$$\log p(\mathbf{x}^{(n)}|\mathbf{z}) = \sum_{t=1}^{T}\log p(\mathbf{x}_t^{(n)}|\mathbf{z}) \tag{3.7}$$

Figure 3.3 illustrates the proposed model.

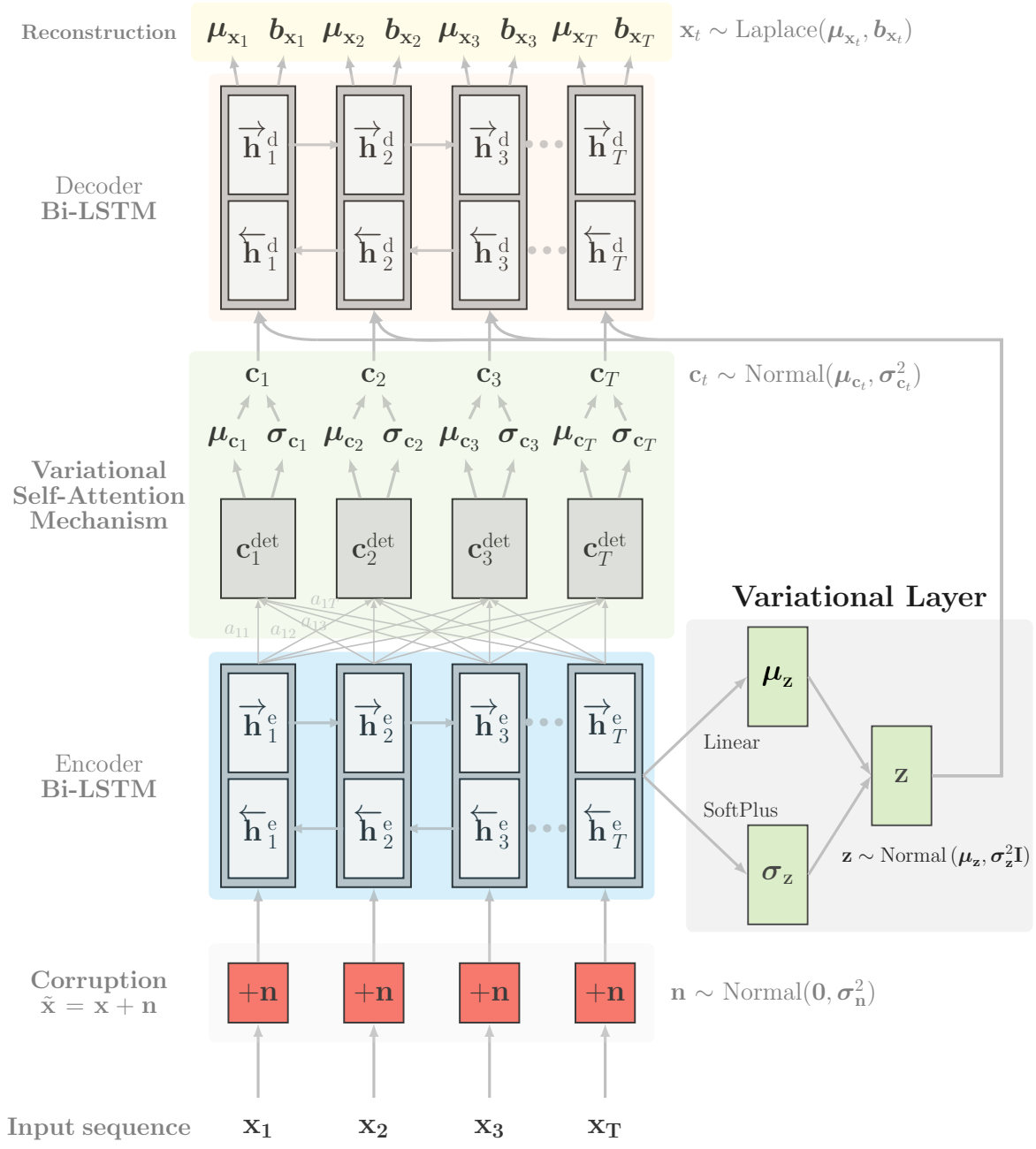


Figure 3.3: Proposed Variational Bi-LSTM Autoencoder with Variational Self-Attention Mechanism.

3.2 Anomaly Detection

The anomaly detection task refers to the problem of finding whether a given observed sequence $\mathbf{x}$ is normal or anomalous. The proposed model makes it possible to perform detection in two different spaces or domains: in the space of the input variable $\mathbf{x}$, using the reconstruction parameters, and in the latent variables $\mathbf{z}$-space, using the representations. In this section both detection methodologies are described in detail.

3.2.1 Reconstruction-based Detection

The reconstruction-based detection strategy is based on the following principle. The Variational Bi-LSTM Autoencoder with Attention is trained on normal data sequences, so that it learns the normal pattern of data. At test time, normal sequences are expected to be well reconstructed whereas anomalous ones are not, since the model has not seen anomalous data during training.

Unlike deterministic autoencoders, the proposed model based on VAE reconstructs the distribution parameters (mean $\boldsymbol{\mu}_{\mathbf{x}}$ and diversity $\boldsymbol{b}_{\mathbf{x}}$, in the case of a $\mathrm{Laplace}$ distribution) of the input variable rather than the input variable itself. Therefore, it is possible to use probability measures as anomaly scores. One approach is to compute the *reconstruction probability*, introduced by An and Cho [2015]. The reconstruction probability is an estimation of the reconstruction term of the VAE loss function by Monte Carlo integration.

$$\mathbb{E}_{\mathbf{z}_l \sim q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] \approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{z}_l)$$

The process can be described as follows. First, an input test sequence $\mathbf{x}$ is propagated through the encoder and the posterior parameters $\boldsymbol{\mu}_{\mathbf{z}}$ and $\boldsymbol{\sigma}_{\mathbf{z}}$ are obtained in a fully deterministic fashion. Then, L samples are drawn from an isotropic Gaussian distribution with these parameters. Each sample $\mathbf{z}_l$ is propagated through the decoder network that outputs the distribution parameters of the reconstruction. Afterwards, it is computed the log-likelihood of the input sample $\mathbf{x}$, given a latent code $\mathbf{z}_l$ drawn from the approximate posterior distribution. Finally, the reconstruction probability is averaged over all samples drawn. The process is summarized in Algorithm 2.

Algorithm 2 Reconstruction Probability Score

Input: $\mathbf{x} \in \mathbb{R}^{T \times d_{\mathbf{x}}}$
Output: $ReconstructionProbability \in \mathbb{R}^T$
$(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}}) \leftarrow \mathrm{Encoder}(\mathbf{x})$
for $l = 1$ to L **do**
  $\mathbf{z}_l \sim \mathrm{Normal}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})$
  $(\boldsymbol{\mu}_{\mathbf{x}}^l, \mathbf{b}_{\mathbf{x}}^l) \leftarrow \mathrm{Decoder}(\mathbf{z}_l)$
  $\mathrm{score}^l \leftarrow \log p(\mathbf{x}|\boldsymbol{\mu}_{\mathbf{x}}^l, \mathbf{b}_{\mathbf{x}}^l)$
end for
$ReconstructionProbability \leftarrow \frac{1}{L}\sum_{l=1}^{L} \mathrm{score}^l$
return $ReconstructionProbability$

The anomaly score itself corresponds to the negative reconstruction probability, so that the lower the reconstruction probability, the higher the anomaly score.

There are several advantages in using the reconstruction probability instead of a deterministic reconstruction error, which is commonly used in autoencoder-based anomaly detection approaches. The first one is related with the detection threshold. The reconstruction probability does not requires data-specific thresholds for detecting anomalies, since it is a probabilistic measure. Using such a metric provides a more intuitive and objective way of analysing the results. The second one is that the reconstruction probability takes into account the variability of the data. Intuitively, anomalous data has higher variance than normal data and, hence, the reconstruction probability is likely to be lower for anomalous examples. The integration of the variability of data concept in anomaly detection enriches the expressive power of the proposed model relatively to conventional autoencoders. Even though the representations of normal and anomalous data in the latent space might share the same expectation, $\boldsymbol{\mu}_{\mathbf{z}}$, the variability of anomalous samples relatively to normal ones is likely to be higher, as pointed out by An and Cho [2015] and, thus, provide an extra tool to distinguish anomalous examples from the normal ones.

Is is also interesting to compute a (stochastic) reconstruction error (RE) by Monte Carlo sampling (equation 3.8), which can be used as an alternative anomaly metric.

$$RE_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}(\mathbf{x}) = \frac{1}{L}\sum_{l=1}^{L}\left\|\mathbf{x} - \underbrace{\mathbb{E}\big[p_\theta\left(\mathbf{x}_l|\mathbf{z}_l\right)\big]}_{\boldsymbol{\mu}_{\mathbf{x}_l}}\right\|_1 \tag{3.8}$$

3.2.2 Latent Space-based Detection

The representation learning model learns to map input data sequences $\mathbf{x}$ with different patterns into different regions of the space and, therefore, it is straightforward to use those representations to distinguish between normal and anomalous samples. Hence, this detection strategy operates in the space of representations, rather than in the space of the input data as the reconstruction-based detection does.

Given a set of latent representations, the goal of anomaly detection is to find out whether a given representation is *normal* or *anomalous*. For this purpose, different methodologies can be employed and they can be either supervised or unsupervised. Since this Thesis aims to develop an unsupervised framework for anomaly detection the focus is on unsupervised detection methods. However, when considering latent space-based detection, a supervised model will also be tested, so that it can be used as a reference for evaluating both frameworks, supervised and unsupervised.

The first latent-space detection method consist on applying unsupervised clustering in the $\boldsymbol{\mu}_{\mathbf{z}}$ space $\big(\boldsymbol{\mu}_{\mathbf{z}} = \mathbb{E}\big[q_\phi(\mathbf{z}|\mathbf{x})\big]\big)$, using hierarchical (agglomerative) clustering, spectral clustering and *k*-means++.

The second latent-space detection approach takes into account the variability of the latent representations, rather than just their expectation. For obtaining an anomaly score, it is computed the median *Wasserstein* distance [Villani, 2009] between a test sample $\mathbf{z}^{\text{test}}$ and N_W other samples within the test set of latent representations, so that the similarity between the posterior distribution of a given sample and a subset of other samples is used as anomaly score. This methodology works under the assumption often made in anomaly detection problems that most data are normal. The computations are described

by equations 3.9 and 3.10.

$$W(\mathbf{z}^{\text{test}}, \mathbf{z}^i)^2 = \|\boldsymbol{\mu}_{\mathbf{z}^{\text{test}}} - \boldsymbol{\mu}_{\mathbf{z}^i}\|_2^2 + \|\boldsymbol{\Sigma}_{\mathbf{z}^{\text{test}}}^{1/2} - \boldsymbol{\Sigma}_{\mathbf{z}^i}^{1/2}\|_F^2 \tag{3.9}$$

$$\text{score}(\mathbf{z}^{\text{test}}) = \text{median}\{W(\mathbf{z}^{\text{test}}, \mathbf{z}^i)^2\}_{i=1}^{N_W} \tag{3.10}$$

In equations 3.9 and 3.10, W denotes the *Wasserstein* distance and the subscript 2 and F denote the ℓ_2-norm and the *Frobenius* norm, respectively.

3.2.3 Dimensionality of the Latent Space

The dimensionality of the latent space $\mathbf{z}$, $d_{\mathbf{z}}$, has a major impact on the learning process. Therefore, it is important to discuss the effect of this hyper-parameter from an anomaly detection point of view.

On one hand, choosing a very small $d_{\mathbf{z}}$ would lead to under-fitting to training data and, in that case, the model might not be able to reconstruct well enough the normal pattern of the sequences. On the other hand, choosing a too large $d_{\mathbf{z}}$ could cause over-fitting to the training data and lead to poor generalisation. Moreover, with a code of larger size, the model could start learning to reconstruct even anomalous sequences and, thus, the performance of anomaly detection would be reduced, specially in terms of false negative examples that would not be detected in this scenario.

Therefore, the choice of the dimensionality of the latent space, $d_{\mathbf{z}}$, is just another instance of the bias-variance trade-off. In a fully unsupervised scenario, it is difficult to choose this parameter. Very often in the literature this choice is performed empirically.

Chapter 4

Experiments & Results

> To deal with a 14-dimensional space, visualize a 3D space and say ”fourteen” to yourself very loudly. Everyone does it.
>
> *Geoffrey Hinton*

This Thesis was motivated by a particular application of anomaly detection that aims to find anomalous behaviour in solar energy generation time series data. Nevertheless, one of the goals of this Thesis was to develop a generic framework for anomaly detection, that is to say, one methodology that could work for energy data but, at the same time, that could be applied to other time series (univariate or multivariate) or, in a broader sense, to other kinds of sequential data, such as text and videos. In this context, to test the effectiveness of the model in data from other fields, another dataset is considered. It comes from an important application domain of anomaly detection - healthcare - and it consists of electrocardiogram (ECG) time series, that are a yet challenging problem for machine learning based anomaly detection approaches, specially in the absence of anomaly labels.

This chapter is organized as follows. In section 4.1, are described the training and detection settings. Afterwards, in sections 4.2 and 4.3, are explained the results. In particular, it is presented a description of the dataset, the optimization and regularization settings including the hyper-parameters used in the experiments, the anomaly detection results and an analysis of the representations learned by the model.

4.1 Training and Detection Modes

The proposed model can operate under two modes: off-line and on-line.

The *off-line* mode is mostly employed for finding whether an entire sequence is normal or anomalous. Although this framework is mainly applied to sequence classification problems, in which a single anomaly score or label is produced for an entire sequence, the proposed model outputs reconstructions parameters and anomaly scores for observations at every timestep t, meaning that it is possible to localize the anomaly within a given input sequence $\mathbf{x}$. However, in this mode, the scores at particular timesteps t can depend on future observations within the same window and this is the reason why this

framework corresponds to an off-line setup. Training is performed with non-overlapping sequences of length T and the observations within a sequence share a unique representation in the latent space $\mathbf{z}$.

In some applications it might be important to minimize the detection delay and perform anomaly detection in real-time. This framework requires an on-line strategy. In such scenarios, no information about future observations can be considered. In the *on-line mode*, training is executed using overlapping sequences obtained with a sliding window with a width $T < L$ and a step size of 1. At test time, the detection is performed without considering observations of future time instants, by feeding up the model with a window of observations in which the last point corresponds to the current timestep t. Hence, the anomaly score at time instant t corresponds to the score of the last observation within the sequence whose last timestep is t. In this mode, for a long sequence with length L, are generated $L - T + 1$ windows of length T and each one of them has its own representation in the latent space. Since these windows overlap and, thus, share observations over time, the latent space will exhibit trajectories over time.

4.2 Solar Energy Generation Dataset: Results & Analysis

The solar energy generation dataset was provided by C-Side[1], a Portuguese company that develops intelligent solutions in areas such as energy, automation, surveillance and security systems.
The dataset ($\mathcal{X}$) is composed of univariate time series ($d_{\mathbf{x}} = 1$) of solar photovoltaic (PV) energy generation coming from about 6000 residential installations distributed across Portugal. The measurements are acquired by a smart plug installed near the PV panel with a sampling period of 15 min, communicated to a gateway and stored in [kWh] units. Each installation is expected to produce 96 observations per day. The dataset includes a total of ≈ 100 million entries, corresponding to roughly 1 million PV production curves. Solar photovoltaic generation time series are characterised by a strong seasonality, with predominant seasonal period of a day (24h, 96 samples). An example of a daily production curve in a day without clouds is shown in Figure 4.1.

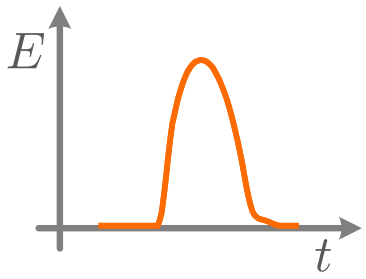


Figure 4.1: Representation of a daily production curve (24h).

The dataset is fully unlabelled, meaning that no information regarding anomalies is available.
The training data was obtained by selecting a subset $\mathcal{X}^{\text{normal}} \subset \mathcal{X}$ of 1430 daily sequences with normal pattern (days without clouds and any kind of anomaly, where the energy generated is as expected). The dataset of normal sequences was divided into two subsets - a training set $\mathcal{X}^{\text{normal}}_{\text{train}}$ and a validation set $\mathcal{X}^{\text{normal}}_{\text{val}}$ - with a splitting ratio of 80/20, respectively. The data was also normalised to the installed capacity, so that the range of observed values lies in the interval $[0, 1]$.

4.2.1 Optimization and Regularization

Optimization was executed using *AMS-Grad* [Reddi *et al.*, 2018] optimizer, a variant of *Adam* [Kingma and Ba, 2014], in mini-batches of size 200 (off-line mode) and 10000 (on-line mode), during 1500 epochs. The learning rate was 0.001 and the network weights were initialized using Xavier initialization [Glorot and Bengio, 2010]. The full model has 274.958 parameters to optimize. The latent space dimensionality and the context vectors dimensionality was set to 3. The encoder and decoder Bi-LSTM both have 256 units, 128 in each direction. The noise added to the inputs for the denoising autoencoding criterion has variance $\sigma^2_{\mathbf{n}} = 0.1\sigma^2_{\mathbf{x}}$. The gradients were clipped by value with a clip value of 1.0. It was also applied a KL-annealing scheme [Bowman *et al.*, 2015] that consists on varying the weight λ_{KL} during training. By doing so, λ_{KL} is initially close to zero in order to allow accurate reconstructions in the early stages of training and is gradually increased to promote smooth encodings and diversity. The parameter η that

[1]Website: `www.cside.pt`

balances the two KL-divergence terms - latent space and attention - was 0.01. A sparsity regularizer was also applied to the activations of the encoder Bi-LSTM [Arpit *et al.*, 2016] that penalizes the ℓ_1-norm of the activations with a weight of 10^{-8}.

4.2.2 Anomaly Detection Results

To illustrate the effectiveness of the proposed approach, some examples of solar energy generation curves representative of different patterns and behaviours were annotated ($\mathcal{X}_{\text{test}}$), such as a sequence with normal pattern used as ground truth, a brief shading, a fault, a spike anomaly, an example of a daily curve where snow covered the surface of the PV panel and a sequence corresponding to a day with clouds. Figure 4.2 shows the annotated examples of solar PV generation daily curves and the corresponding anomaly scores: the *reconstruction probability* (top bar) and the reconstruction error (bottom bar), both obtained using $L = 512$ Monte Carlo samples.

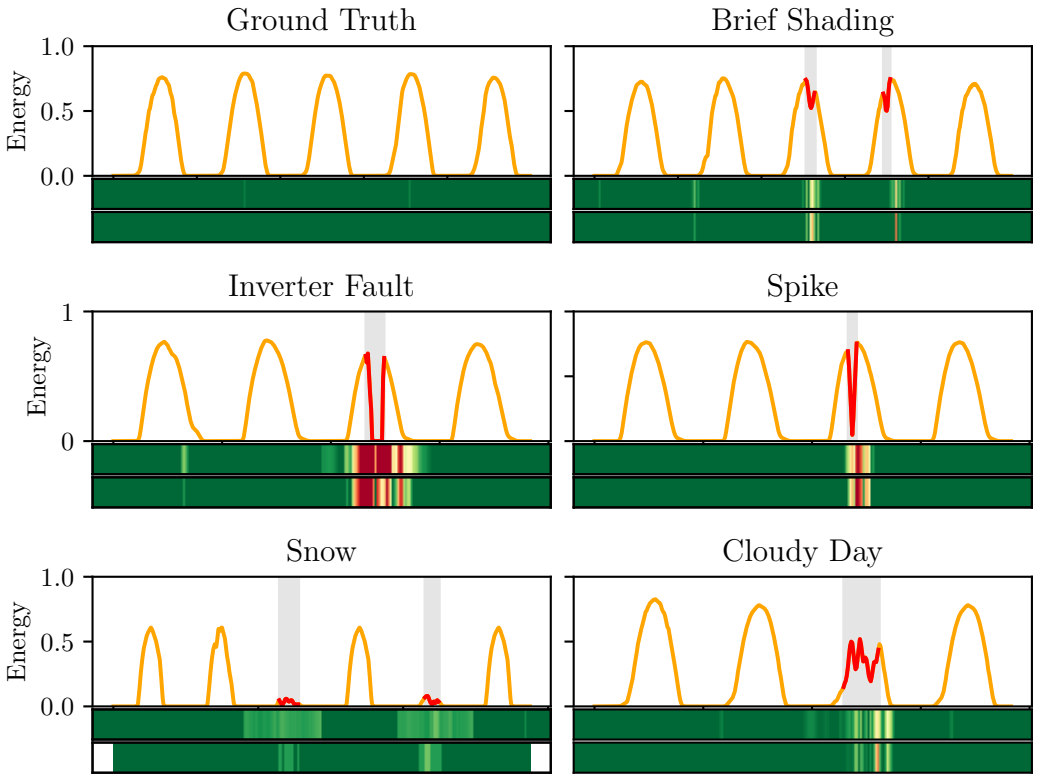


Figure 4.2: Anomaly scores for some representative sequences: reconstruction probability (top bar) and reconstruction error (bottom bar). (off-line mode, non-overlapping sequences with $T = 96$ timesteps).

The training and validation losses are shown in Table 4.1.

Set	**Training** ($\mathcal{X}_{\text{train}}^{\text{normal}}$)	**Validation** ($\mathcal{X}_{\text{val}}^{\text{normal}}$)
Loss	-3.1457	-3.1169

Table 4.1: Training and validation losses.

The training and validation losses are similar, meaning that the model is not over-fitting to training data and is able to generalize to unseen (normal) sequences, reconstructing them well.

The proposed model provides the parameters of the output distribution (mean $\boldsymbol{\mu}_{\mathbf{x}}$ and diversity $\mathbf{b}_{\mathbf{x}}$). As previously mentioned in section 3.2.1, the diversity of the output distribution can improve the anomaly detection task, since anomalous data has, in principle, higher variability. Figure 4.3 shows an example of a time series of a particular installation and the corresponding output parameters of the model.

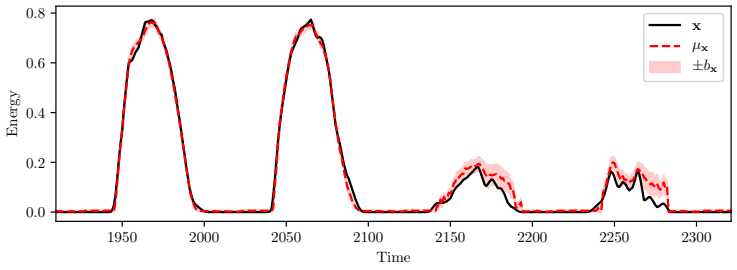


Figure 4.3: Visualization of a 4-day sequence with 2 days of anomalous energy generation and the corresponding parameters of the output distribution, mean $\boldsymbol{\mu}_{\mathbf{x}}$ and diversity $\boldsymbol{b}_{\mathbf{x}}$.

Figure 4.3 shows that, for the last two days (with a lower energy production), the model reconstructs badly the sequence, while the diversity is higher when compared with the first two days, with normal production. This result validates the importance of taking into account the variability of the reconstruction, rather than just its expectation.

4.2.3 Latent Space Analysis

At the heart of an autoencoder lies its latent space: a low-dimensional representation of the data that encodes its underlying factors of variation. Therefore, it is interesting to visualize these representations. For visualization purposes, the dimensionality of the latent space was reduced from 3D ($d_{\mathbf{z}} = 3$) to 2D using Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbour Embedding (t-SNE) [van der Maaten and Hinton, 2008].

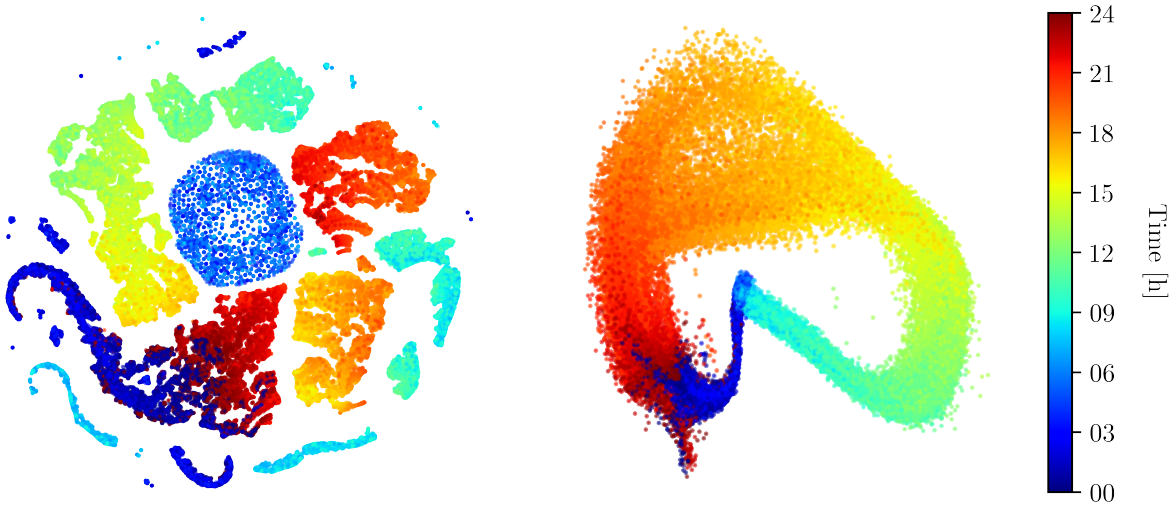


Figure 4.4: Latent Space visualization of $\mathcal{X}_{\text{train}}^{\text{normal}}$ (with only normal samples) in 2D via t-SNE (left) and PCA (right). The label corresponds to the time instant of the last observation within each sequence. (Setup: $d_{\mathbf{z}} = 3$, on-line mode, training executed using overlapping sequences with $T = 12$ timesteps).

The visualization of the latent space shows evidence that the model is mapping sequences aligned in time onto the same region of the $\mathbf{z}$-space and, more interestingly, it reveals a cyclic trajectory whose period matches exactly the seasonal period of the solar PV curves: one day. In other words, the model has learned the seasonal property of the data without being told of it. It is important to recall that no prior information regarding the seasonality property is provided, all the windows were shuffled during training and in this experiment each window of observations has 12 timesteps, which is less than the seasonal period of the data (96). Previous works have shown latent spaces with similar behaviour, even though without analysing and interpreting it, until the recent work of Xu *et al.* [2018] that provided for the first time an explanation for this effect that they called *Time Gradient*.

In the context of time series anomaly detection, it is interesting to exploit how the representations of anomalous data compare with the representations of normal examples. Figure 4.5 shows the latent representations for the sequences that were annotated and validated by the company that provided the dataset. Since the variational latent space is obtained by sampling from the posterior distribution, in this plot is represented the mean $\boldsymbol{\mu}_{\mathbf{z}} = \mathbb{E}\big[q_\phi(\mathbf{z}|\mathbf{x})\big]$ space, which is deterministically obtained from the encoder Bi-LSTM output.

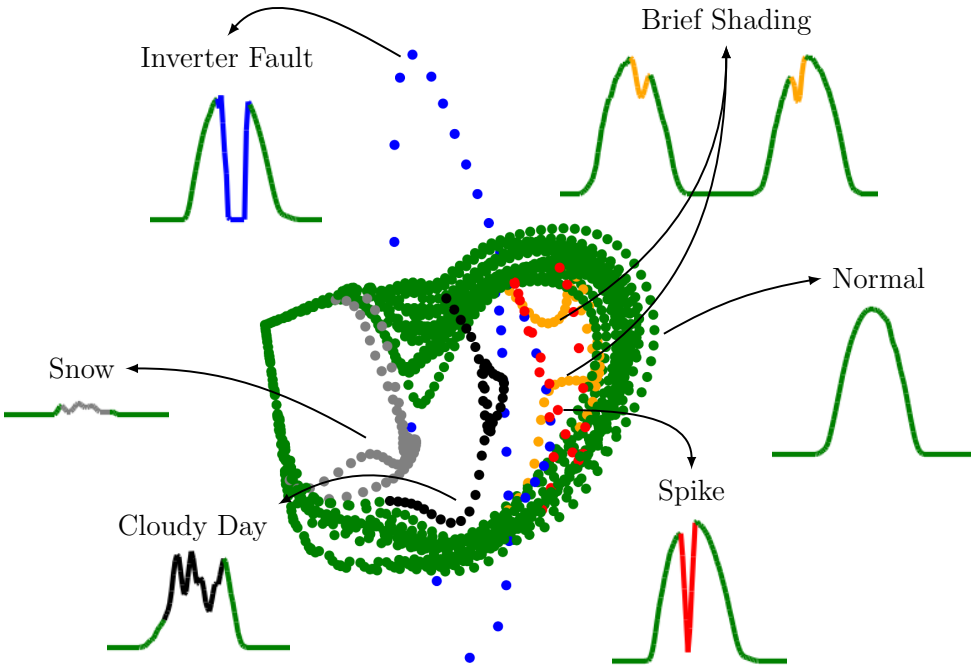


Figure 4.5: Visualization of the latent space in 2D via PCA for the test set ($\mathcal{X}_{\text{test}}$) containing some annotated sequences (on-line mode, $T = 12$ timesteps).

Figure 4.5 shows trajectories over the latent space that were generated by the sliding window approach employed in the on-line anomaly detection mode. The *normal* samples (green) and the *anomalous* ones are represented differently in the space and there is a clear deviation of the anomalous examples from the normal trajectory. This conclusion supports the fact that the model has, indeed, learned a manifold of normal data and its now projecting sequences with different behaviour onto different regions of the space. Moreover, the *normal* data exhibits slightly different trajectories in the space mainly because even though the daily curves have the same qualitative (normal) pattern, they are shifted in time

due to different locations of the installations, where the sun starts shining on the PV panel at different moments and also due to different inclinations of the panel.

The latent variables components can also be visualized in time, i.e. each dimension over a sequence of consecutive overlapping windows. Figure 4.6 shows, on the left side, the mean μ_{z_i} of each component i over 400 consecutive training windows. On the right side, the autocorrelation function is represented for each component of the latent variables. This experiment was performed using a 3-dimensional latent space ($d_{\mathbf{z}} = 3$).

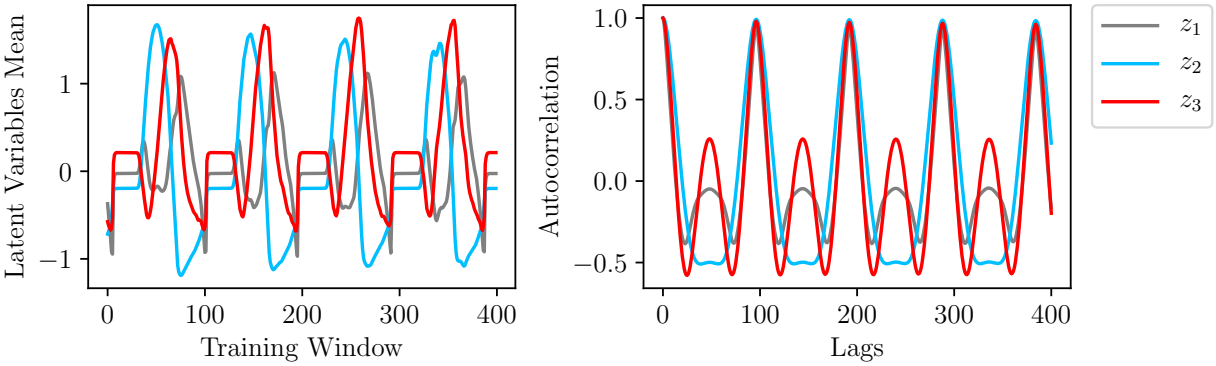


Figure 4.6: Representation of the latent variables and the corresponding autocorrelation over consecutive training windows (on-line mode, $T = 12$ timesteps, $d_{\mathbf{z}} = 3$).

Figure 4.6 clearly shows that the latent variables have a seasonal behaviour. More interestingly, they have a predominant seasonal period equal to 96, which corresponds to one day and matches exactly the seasonal period of the data. This behaviour explains the previously analysed latent space. It seems that the model has learned the seasonal pattern of the solar energy time series and is encoding this property in the latent variables. Such an evidence supports the ability of the proposed model to deal with seasonal data, which is very often a concern in applications dealing with time series data.

4.2.4 Attention Visualization

The Variational Self-Attention Mechanism aids the decoding process by allowing the model to pay more attention to particular hidden states and it does so by computing a set of attention weights $\{a_{ij}\}_{i,j=1}^{T}$. Therefore, the attention model produces a 2D *attention map* for each sequence that shows where the network is focusing its attention. Figure 4.7 shows the attention maps for different test sequences with and without anomalies. The attention weights are represented in a logarithmic scale.
The attention maps show evidence that the self-attention model is producing context-aware representations, which can be seen by the distribution of the attention weights in a small window around the first diagonal of the maps. This result supports the intuition that most of the temporal context of an observation in a time series lies in a narrow window around it. Furthermore, for different anomalies, the maps show different distributions of the attention weights. In some cases, the self-attention model is capturing dependencies between hidden states far in time. This conclusion validates the proposed reconstruction-based anomaly detection approach, since it tells that the network struggles to reconstruct well anomalous sequences and tries to capture long-term context in those.

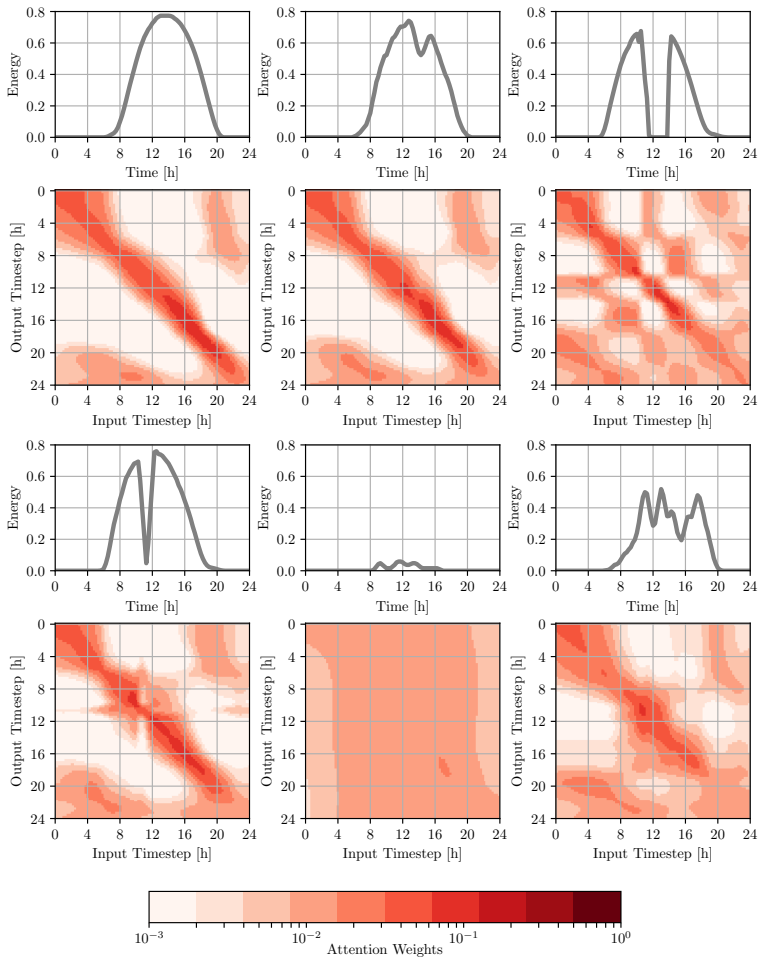


Figure 4.7: Visualization of the attention maps for the annotated set ($\mathcal{X}_{\text{test}}$).

It is also possible to visualize the context vectors, $\mathbf{c}_t$, in the mean space ($\boldsymbol{\mu}_{\mathbf{c}_t}$). The visualization, shown in Figure 4.8, was performed by reducing the dimensionality of $\boldsymbol{\mu}_{\mathbf{c}_t}$ to 2D using PCA. The labels represent the corresponding time instant t. Each context vector is computed as a weighted sum of all the encoded hidden states, so each one of them combines information from different time instants.

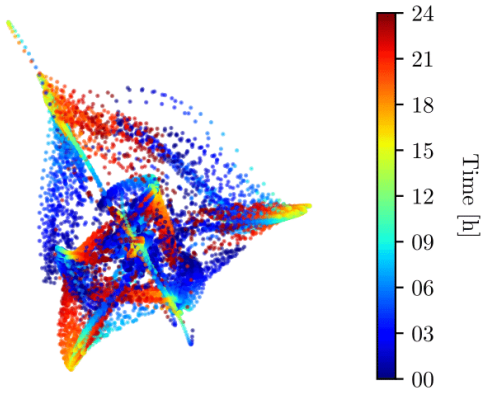


Figure 4.8: Visualization of the context vectors, $\mathbf{c}_t$, of the validation set $\mathcal{X}_{\text{val}}^{\text{normal}}$.

4.3 Electrocardiogram Dataset: Results & Analysis

The electrocardiogram (ECG) dataset is the `ECG5000`, which was donated by Eamonn Keogh and Yanping Chen and is publicly available in the UCR Time Series Classification archive [Chen *et al.*, 2015]. This dataset is composed of 5000 univariate time series ($d_{\mathbf{x}} = 1$) with 140 observations ($T = 140$). Each sequence corresponds to one heartbeat. Five classes are annotated, corresponding to the following labels: *Normal* (N), *R-on-T Premature Ventricular Contraction* (R-on-T PVC), *Premature Ventricular Contraction* (PVC), *Supra-ventricular Premature or Ectopic Beat* (SP or EB) and *Unclassified Beat* (UB). In the original data source, the dataset is provided with a splitting into two subsets: a training set with 500 sequences and a test set with 4500 sequences. Both the training and the test set contain all classes, meaning that the training set contain both normal and anomalous data. Moreover, the classes are highly imbalanced, the normal class is the predominant one followed by the class with label R-on-T PVC. For validation, the original training set was divided into two subsets - one for training the model ($\mathcal{X}_{\text{train}}$) and one for validation ($\mathcal{X}_{\text{val}}$) - with a splitting ratio of $80/20$, respectively. No further pre-processing was executed. Figure 4.9 shows the density of each class per set.

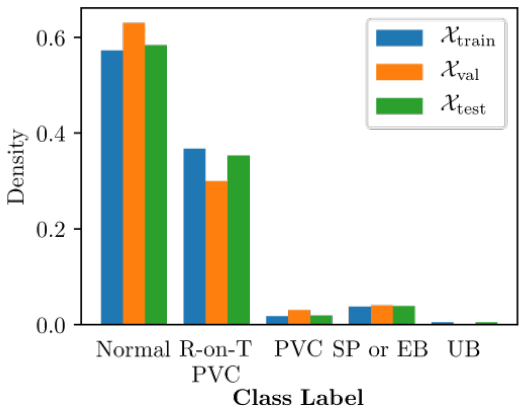


Figure 4.9: Class densities per set.

The `ECG5000` dataset is labelled and, therefore, it is possible to evaluate the performance of the unsupervised anomaly detection methods using conventional classification scores. Every sequence has a single label, meaning that no information regarding the location of the anomalous region of the ECG is provided. Hence, the detection strategy will be based on the representations space (previously described in section 3.2.2). Given a set of representations of heartbeats in the latent space, the objective is to find which sequences are normal and anomalous. Rather than computing an anomaly score for every observation within a sequence, the objective is to conclude about the whole sequence. Since the detection is based on the latent representations, the attention model, which aids the decoding phase, is not considered in this framework.

4.3.1 Optimization and Regularisation

Training was performed using *AMS-Grad* [Reddi *et al.*, 2018] optimizer, a variant of *Adam* [Kingma and Ba, 2014], with a learning rate of 0.001. Gradient computation and weight updates are performed in

mini-batches of size 500, during 1500 epochs. The latent space space dimensionality, $d_{\mathbf{z}}$, was set to 5, corresponding to an encoding compression ratio of 28. The encoder and decoder Bi-LSTM both have 256 units in total, 128 in each direction. The noise added at the input level has a standard deviation $\sigma_{\mathbf{n}} = 0.8\sigma_{\mathbf{x}}$. The number of Monte Carlo samples, L, is set to 1 during training, following the work of Kingma and Welling [2013]. To compute the anomaly scores based on the Wasserstein distance, $N_W = 4000$ examples are used. To promote stability during training, the gradients were clipped by value with a limit on their magnitude of 5.0. To prevent the KL-divergence term vanishing problem [Bowman *et al.*, 2015], a KL-annealing strategy is applied in order to vary the weight λ_{KL} of the KL-divergence term in the loss function. By doing so, the weight λ_{KL} is initially close to zero - to promote accurate reconstructions of $\mathbf{x}$ in the early stages of training - and gradually increased to encourage smooth encodings and diversity. Furthermore, a sparse regularization criterion is employed to promote sparsity in the hidden layer of the encoder Bi-LSTM [Arpit *et al.*, 2016], by adding a penalty on the ℓ_1-norm of the activations with a weight parameter of 10^{-7}. The total number of parameters to optimize is 273.420.

4.3.2 Anomaly Detection Results

The anomaly detection results are evaluated using Area Under the Curve (AUC), Accuracy, Precision, Recall and F_1-score. These scores are weighted per-class. Since the output of a clustering algorithm might provide permuted labels, i.e. the cluster assignments may be permuted between the normal and anomalous classes, a search is performed over all possible matches between cluster assignments and ground-truth labels and the combination corresponding to the best scores is chosen, similarly to Farhadi *et al.* [2015]. Table 4.2 presents the detection results computed on the test set, $\mathcal{X}_{\mathrm{test}}$, using different clustering algorithms and a linear SVM. All results reported were averaged over 10 runs of both the representation learning and detection models.

Metric	Hierarchical	Spectral	k-Means	Wasserstein	SVM
AUC	0.9569	0.9591	0.9591	**0.9819**	0.9836
Accuracy	0.9554	0.9581	**0.9596**	0.9510	0.9843
Precision	**0.9585**	0.9470	0.9544	0.9469	0.9847
Recall	0.9463	0.9516	**0.9538**	0.9465	0.9843
F_1-score	0.9465	0.9474	**0.9522**	0.9461	0.9844

Table 4.2: Anomaly detection scores for the electrocardiogram $\mathrm{ECG5000}$ dataset. The best *unsupervised* anomaly detection scores are emphasized in bold.

Figure 4.10 shows the Receiver Operating Characteristic curve for the Wasserstein distance metric, that yields the best unsupervised anomaly detection AUC score.

The results obtained for the three clustering algorithms are roughly identical. This fact supports the idea that the key challenge in unsupervised anomaly detection is to learn good (expressive) representations of the data. This is the reason why this Thesis is strongly focused on representation learning. Furthermore, the *Wasserstein* distance-based score outperforms clustering-based detection in terms

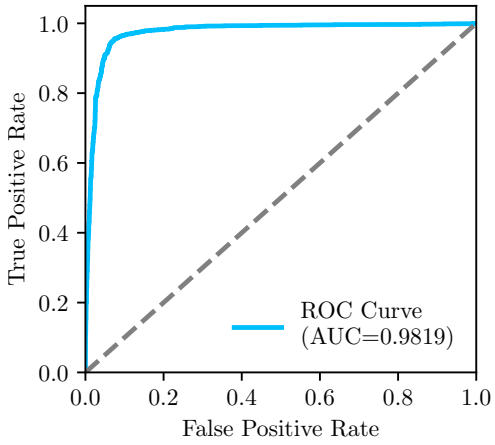


Figure 4.10: Receiver operating characteristic curve for the Wasserstein distance-based detection.

of AUC and is similar in terms of the other metrics. This result is expected since this score is taking into account the variability of the representations in the latent space, rather than just their mean. The supervised Support Vector Machine performs very well, while the unsupervised detection methods stay roughly competitive. Anyway, all detection strategies attained relatively high detection scores.

Other works have used the same dataset mainly in a supervised multi-class classification framework, instead of anomaly detection that is a two-class problem. Even though both schemes can not be compared in general, since the dataset is highly imbalanced, with a large predominance of the normal and one of the anomalous classes (Figure 4.9), the multi-class classification problem is almost degenerated in a two-class one. Therefore, it is interesting to compare the results obtained with the proposed approach with the results reported in other works that considered different techniques. Table 4.3 summarizes the best scores obtained using both supervised and unsupervised learning models in several recent works and the best results for each metric are emphasized in bold.

Source	**S/U**[a]	**Model**	**AUC**	**Acc**	$\mathbf{F_1}$
Proposed	S	VRAE+SVM	**0.9836**	**0.9843**	**0.9844**
	U	VRAE+Clust/W	**0.9819**	**0.9596**	**0.9522**
Lei *et al.* [2017]	S	SPIRAL-XGB	0.9100	-	-
Karim *et al.* [2017]	S	F-t ALSTM-FCN	-	0.9496	-
Malhotra *et al.* [2016]	S	SAE-C	-	0.9340	-
Liu *et al.* [2018]	U	oFCMdd	-	-	0.8084

[a]Supervised/Unsupervised

Table 4.3: Comparison of the results with other works using the $ECG5000$ dataset.

Under the two-class approximation made above, the proposed unsupervised approach outperforms previous supervised and unsupervised learning models in every score reported.

4.3.3 Latent Space Analysis

Figure 4.11 shows the latent space of the entire test set $\mathcal{X}_{\text{test}}$ with 4500 sequences. Each datapoint is labelled with one of the five possible classes annotated. For visualization purposes, the dimensionality of the latent space is reduced from 5 to 2 dimensions using Principal Component Analysis and t-Distributed Stochastic Neighbour Embedding. For the t-SNE embedding, the perplexity parameter was set to 50.0, while the number of iterations was 2000.

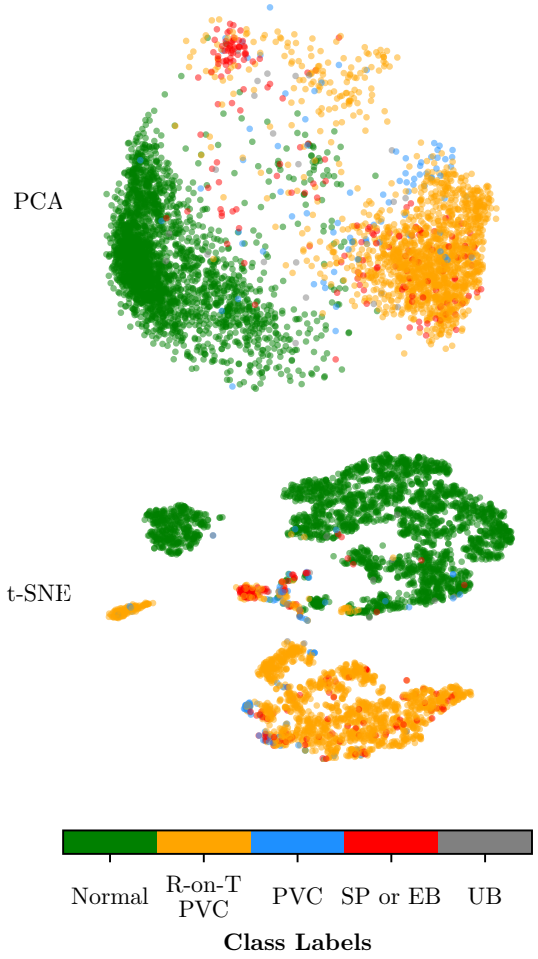


Figure 4.11: Latent Space Visualization of $\mathcal{X}_{\text{test}}$ in 2D via PCA (top) and t-SNE (bottom).

Figure 4.11 reveals a structured and expressive latent space. The sequences (heartbeats) of the *normal* class, represented in green, lie in a region of the latent space different from the *anomalous* ones, while similar heartbeats are mapped onto the same region of the space. Moreover, it is also clear that different anomalies are represented in distinct regions of the space. The anomalous heartbeats in blue and orange, which refer to premature ventricular contractions, are represented close to each other. Interestingly, the anomaly with label "R-on-T PVC", represented in orange, has a smaller cluster apart from the larger one (top of the figure). This might be an interesting result to be analysed by experts.

4.4 Implementation, Hardware & Computational Efficiency

All the models were implemented using the Keras deep learning library for Python [Chollet, 2015], with TensorFlow backend [Abadi *et al.*, 2015]. Training was performed using a single NVIDIA GTX 1080 TI graphics processing unit with 11GB of memory, in a machine with an 8th generation i7 processor and 16GB of DDR4 RAM.

One important aspect of anomaly detection is efficiency. In many applications involving real world data, it is important to ensure that the anomaly scores are computed in a short amount of time and with a minimum delay.

Dataset	# parameters	# timesteps	# sequences	**Training Time** [ms/seq]	**Inference Time** [ms/seq]	**Anomaly Scores** [ms/seq]
Energy	274.958	32	109728	0.09970	0.02244	21.15
ECG	273.420	140	400	2.002	2.4214	31.34

Table 4.4: Computational efficiency of training, inference and anomaly scores computation. The anomaly scores are the reconstruction probability and the reconstruction error (for the solar energy dataset), computed using $L = 512$ Monte Carlo samples, and the Wasserstein score based on $N_W = 4000$ representations (for the ECG dataset).

The model can infer and produce an anomaly score within a few dozens of milliseconds, which is a very short period of time. In some real-world applications the decisions can be usually made in dozens of milliseconds and, thus, the proposed model would be suitable for those. In the energy application, the detection period is not very important, since the objective is to find if the system is behaving well, for instance in a daily or weekly basis. However, when dealing with electrocardiogram data, the detection delay is much more relevant, since an early alert of anomaly may allow someone to react soon and to take the necessary measures.

4.5 Discussion

Clearly, the metrics and scores based on the latent space representations show the ability of the method to perform anomaly detection, as well as the reconstruction-based metrics. Such results validate the choice of a representation learning based model for this Thesis, which allows different detection strategies. More importantly, as desired since the beginning, both the Variational Bi-LSTM Autoencoder and the detection models are unsupervised and have proven to outperform previous supervised and unsupervised approaches. From the point of view of the author, the unsupervised nature of the approach is one of the key achievements of this Thesis.

In reconstruction-based detection, the reconstruction probability reveals an improved capacity of detecting anomalous patterns relatively to the reconstruction error, specially in the case where the production curves show a reduced energy production (e.g., snow anomaly) and the predicted mean is close

to the original value.
In what concerns latent-space based detection, the different clustering algorithms tested revealed similar scores, while the proposed Wasserstein distance score yields a better result in terms of AUC. Both detection strategies were, thus, validated experimentally.

The latent space for both datasets is structured and expressive. For the solar energy generation, the latent space is cyclic, which is an interesting effect considering the seasonality property of these time series. Moreover, the representations of the ECG sequences are also very interesting, since different anomalies are represented in different regions of the space and this was obtained without any kind of supervision. This result makes the quest for a possible line of future work (section 5.3) that can tackle the problem of distinguishing between different anomalies.

Chapter 5

Conclusions

> People worry that computers will get too smart and take over the world, but the real problem is that they are too stupid and they have already taken over the world.
>
> *Pedro Domingos*

5.1 Lessons Learned & Final Remarks

One of the major challenges of this work was the full absence of labels for the energy dataset, which made the quest for this Thesis. Actually, this a common scenario in the context of real-world applications in areas such as energy that makes anomaly detection, indeed, a great challenge. Such a scenario motivated the proposed unsupervised framework for anomaly detection. The main advantage of following such an approach is that it can be applied to a large amount of time series data of very different natures. On the other hand, the main difficulty found due to the lack of labels was evaluation, since it is not possible to compute conventional classification metrics under this scenario. In fact, evaluation metrics and criteria for unsupervised anomaly detection algorithms, in the absence of labels and ground truth, remains a challenging practical problem where the literature is still scarce, even though some recent work has been done on the subject [Goix, 2016].

Furthermore, in unsupervised anomaly detection, the concept of *normality*, often very intuitive for experts, turns out to be hard to define in formal terms. Nevertheless, from an anomaly detection perspective, defining what is normal seems to be more reasonable than defining what is anomalous, outlier or novel. In some real applications the adoption of supervised learning approaches narrows the effectiveness of anomaly detection and delays the detection. For instance, in fraud detection, when a new fraud happens, supervised models are not trained for detecting that new type of fraud, and the system needs to be retrained to detect it. Instead, by modelling what is normal, abnormal behaviour may be detected in advance and new frauds can be detected still in real-time. In applications like network intrusion detection and cyber-fraud, where the attacks profile is always changing, this may also play an important role. Moreover, when dealing with time series, it is hard to stablish the boundaries of what an anomaly is. In large-scale applications of AD, the definition of what is anomalous is often conditioned

by the response capacity of the company to those anomalies. Under such constraints, the goal of an AD algorithm is to identify the *most anomalous* observations that can be checked and confirmed by the company or service provider.

5.2 Summary of Contributions

This Thesis presents a generic, unsupervised and scalable framework for anomaly detection in time series data. The proposed approach fulfils all the requirements defined in the early stages of this work and is able to detect anomalous behaviour in data of very different fields and, therefore, it is a contribution for several possible domains of application of AD, such as energy and healthcare. In particular, the growing need for monitoring the data gathered by the smart grid (e.g., consumption, production time series) and wearable devices (e.g., beat rate, electrocardiogram time series), to give two examples from very different fields, makes the contribution of this work even more relevant.

From the point of view of the author, one of the main contributions of this Thesis lies on the unsupervised nature of the approach. Even though unsupervised learning is still much more challenging than supervised learning, the proposed framework reveals a promising ability to learn expressive representations of data and, afterwards, to detect anomalous observations. The two detection strategies are proven to perform well in different datasets and provide a straightforward way of finding data that do not conform with normal behaviour.

This Thesis is built on top of several powerful ideas and models developed in the framework of different applications, such as Natural Language Processing, Speech Recognition and Image Processing, which were not previously transferred and seamlessly integrated for tasks dealing with time series data. One of the purposes of this Thesis was to unify concepts from these areas into a single framework. In particular, to leverage the power of Bayesian deep learning models such as the variational autoencoder for AD, recurrent neural networks, with their ability to capture the sequential structure of data, and even sparsity that introduces prior knowledge of the anomalies rareness into the model training objective. On top of the seq2seq model, *attention* was also introduced in the context of anomaly detection by means of a novel variational self-attention mechanism, with its ability to improve the encoding-decoding process and to provide a visualization scheme for the sequences.

Finally, another contribution of this Thesis are two scientific papers that comprise the main results obtained with the datasets exploited, using two different detection strategies. The purpose of these papers is to share the approach of this Thesis with the community of researchers and practitioners of anomaly detection. These papers are introduced and attached in the Appendix A of this Thesis.

5.3 Future Work

The proposed framework for anomaly detection makes the quest for possible lines of future work.

First, even though the proposed model was applied univariate time series data, it is suitable to multivariate data as well, in which $\mathbf{x}_t$ can be a $d_{\mathbf{x}}$-dimensional vector of observations. All the derivations made for the proposed model allow this scenario. It is even suitable to be applied to other types of sequential data beyond time series, such as videos and text, for tasks like detecting abnormal behaviour in surveillance videos or detecting abnormal opinions or sentiment patterns in social networks.

Second, the philosophy behind the proposed model is to build an understanding of *normality* by learning a normal data manifold that can be used as a reference for evaluating novel and unseen observations. However, the concept of *normal* might be prone to change/drift over time. Dealing with concept drift is also a subject that can be addressed in future work.

Third, the attention maps provided by the model show that sequences with different patterns exhibit different attention maps. In particular, normal and anomalous sequences have different attention maps. This result can reveal the usefulness of the attention maps for detection. A possible line of future work would be to use the attention maps as a feature map for classification, in the same fashion as other approaches that compute the time series spectrogram and use it to solve anomaly detection as an image classification problem, using convolutional neural networks for instance.

Forth, in this Thesis, anomaly detection was tackled from the point of view of classifying normal and anomalous data. This is a common scenario in other works because AD is, by definition, a two-class problem. However, the proposed approach can be extended to the multi-class case, to allow distinguishing between anomalies. The representations learned are likely to be structured and expressive enough to allow for such a scenario that is still to be done in unsupervised anomaly detection. So far, the representations obtained for the several datasets already allow to take meaningful insights in their fields and provide a visualization strategy to analyse the results.

Finally, in some applications of AD, labelled examples are available and, in that case, it makes sense to use them to improve the model effectiveness. Therefore, an extension of the current work is to design a semi-supervised setting that would allow to take possibly available labels into consideration. Moreover, in this scenario, it would be interesting to exploit a transfer learning approach that would allow to transfer knowledge between a source task where enough labels are available to a target task with fewer labels. This extension would allow to leverage the features learned across different datasets in other similar tasks.

Bibliography

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

Aleskerov, E.; Freisleben, B., and Rao, B. CARDWATCH: A Neural Network based Database Mining System for Credit Card Fraud Detection. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, pages 220–226, March 1997. doi: 10.1109/CIFER.1997.618940.

An, Jinwon and Cho, Sungzoon. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. *SNU Data Mining Center*, 2015.

Angiulli, Fabrizio and Pizzuti, Clara. Fast Outlier Detection in High Dimensional Spaces. In Elomaa, Tapio; Mannila, Heikki, and Toivonen, Hannu, editors, *Principles of Data Mining and Knowledge Discovery*, pages 15–27, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45681-0.

Arpit, Devansh; Zhou, Yingbo; Ngo, Hung, and Govindaraju, Venu. Why Regularized Auto-Encoders learn Sparse Representation? In Balcan, Maria Florina and Weinberger, Kilian Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 136–144, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/arpita16.html`.

Bahdanau, Dzmitry; Cho, Kyunghyun, and Bengio, Yoshua. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

Bahuleyan, Hareesh; Mou, Lili; Vechtomova, Olga, and Poupart, Pascal. Variational Attention for Sequence-to-Sequence Models. *CoRR*, abs/1712.08207, 2017.

Barbará, Daniel; Li, Yi, and Couto, Julia. COOLCAT: An Entropy-based Algorithm for Categorical Clustering. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 582–589, New York, NY, USA, 2002. ACM. ISBN 1-58113-492-4. doi: 10.1145/584792.584888. URL http://doi.acm.org/10.1145/584792.584888.

Bayer, J. and Osendorfer, C. Learning Stochastic Recurrent Networks. *ArXiv e-prints*, November 2014.

Bengio, Yoshua; Courville, Aaron C., and Vincent, Pascal. Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives. *CoRR*, abs/1206.5538, 2012. URL http://arxiv.org/abs/1206.5538.

Bengio, Yoshua; Yao, Li; Alain, Guillaume, and Vincent, Pascal. Generalized Denoising Auto-encoders As Generative Models. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 899–907, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999611.2999712.

Bengio, Yoshua; Chung, Junyoung; Kastner, Kyle; Dinh, Laurent; Goel, Kratarth, and Courville, Aaron C. A Recurrent Latent Variable Model for Sequential Data. *CoRR*, abs/1506.02216, 2015a.

Bengio, Yoshua; Im, Daniel Jiwoong; Ahn, Sungjin, and Memisevic, Roland. Denoising Criterion for Variational Auto-Encoding Framework. *CoRR*, abs/1511.06406, 2015b.

Boriah, Shyam; Chandola, Varun, and Kumar, Vipin. Similarity Measures for Categorical Data: A Comparative Evaluation. In *In Proceedings of the eighth SIAM International Conference on Data Mining*, pages 243–254, 2008.

Bourlard, H. and Kamp, Y. Auto-association by Multilayer Perceptrons and Singular Value Decomposition. *Biol. Cybern.*, 59[4-5]:291–294, September 1988. ISSN 0340-1200. doi: 10.1007/BF00332918. URL http://dx.doi.org/10.1007/BF00332918.

Bowman, Samuel R.; Vilnis, Luke; Vinyals, Oriol; Dai, Andrew M.; Józefowicz, Rafal, and Bengio, Samy. Generating Sentences from a Continuous Space. *CoRR*, abs/1511.06349, 2015.

Chandola, Varun; Banerjee, Arindam, and Kumar, Vipin. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41[3]:15:1–15:58, July 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL http://doi.acm.org/10.1145/1541880.1541882.

Chen, Yanping; Keogh, Eamonn; Hu, Bing; Begum, Nurjahan; Bagnall, Anthony; Mueen, Abdullah, and Batista, Gustavo. The UCR Time Series Classification Archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/.

Cho, Kyunghyun; van Merrienboer, Bart; Gülçehre, Çaglar; Bougares, Fethi; Schwenk, Holger, and Bengio, Yoshua. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Chollet, François. Keras. https://keras.io, 2015.

Chorowski, Jan; Bahdanau, Dzmitry; Serdyuk, Dmitriy; Cho, KyungHyun, and Bengio, Yoshua. Attention-Based Models for Speech Recognition. *CoRR*, abs/1506.07503, 2015.

Duchi, John; Hazan, Elad, and Singer, Yoram. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

Farhadi, Ali; Xie, Junyuan, and Girshick, Ross B. Unsupervised Deep Embedding for Clustering Analysis. *CoRR*, abs/1511.06335, 2015. URL http://arxiv.org/abs/1511.06335.

Gamon, Michael. Graph-based Text Representation for Novelty Detection. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 17–24, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1654758.1654762.

García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G., and Vázquez, E. Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges. *Computers and Security*, 28[1): 18 – 28, 2009. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2008.08.003. URL http://www.sciencedirect.com/science/article/pii/S0167404808000692.

Glorot, Xavier and Bengio, Yoshua. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In Teh, Yee Whye and Titterington, Mike, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

Goix, N. How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms? *ArXiv e-prints*, July 2016.

Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron, and Bengio, Yoshua. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Goodfellow, Ian; Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Graves, Alex. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850, 2013.

Graves, Alex; Fernández, Santiago, and Schmidhuber, Jürgen. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ICANN'05, pages 799–804, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-28755-8, 978-3-540-28755-1. URL http://dl.acm.org/citation.cfm?id=1986079.1986220.

Graves, Alex; Mohamed, Abdel-rahman, and Hinton, Geoffrey E. Speech Recognition with Deep Recurrent Neural Networks. *CoRR*, abs/1303.5778, 2013.

Graves, Alex; Wayne, Greg, and Danihelka, Ivo. Neural Turing Machines. *CoRR*, abs/1410.5401, 2014.

Guthrie, David; Guthrie, Louise; Allison, Ben, and Wilks, Yorick. Unsupervised Anomaly Detection. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1624–1628, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1625275.1625538.

Hansen, Lars Kai; Højen-Sørensen, Pedro A. D. F. R., and Winther, Ole. Mean-field Approaches to Independent Component Analysis. *Neural Comput.*, 14[4):889–918, April 2002. ISSN 0899-7667. doi: 10.1162/089976602317319009. URL http://dx.doi.org/10.1162/089976602317319009.

Hawkins, Simon; He, Hongxing; Williams, Graham, and Baxter, Rohan. Outlier Detection Using Replicator Neural Networks. In Kambayashi, Yahiko; Winiwarter, Werner, and Arikawa, Masatoshi, editors, *Data Warehousing and Knowledge Discovery*, pages 170–180, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-46145-6.

Hübner, Ronald; Steinhauser, Marco, and Lehle, Carola. A dual-stage two-phase model of selective attention. *Psychological Review*, 117[3):759 – 784, 2010. ISSN 0033-295X. URL http://search.ebscohost.com/login.aspx?direct=true&db=pdh&AN=2010-14834-002&lang=pt-br&site=eds-live&scope=site.

He, Zengyou; Xu, Xiaofei, and Deng, Shengchun. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24[9):1641 – 1650, 2003. ISSN 0167-8655. doi: https://doi.org/10.1016/S0167-8655(03)00003-5. URL http://www.sciencedirect.com/science/article/pii/S0167865503000035.

He, Zengyou; Deng, Shengchun, and Xu, Xiaofei. An Optimization Model for Outlier Detection in Categorical Data. In Huang, De-Shuang; Zhang, Xiao-Ping, and Huang, Guang-Bin, editors, *Advances in Intelligent Computing*, pages 400–409, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31902-3.

Heller, Katherine A.; Svore, Krysta M.; Keromytis, Angelos D., and Stolfo, Salvatore J. One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses. In *In Proc. of the workshop on Data Mining for Computer Security*, 2003.

Hinton, G.; Deng, L.; Yu, D.; Dahl, G. E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T. N., and Kingsbury, B. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29[6):82–97, Nov 2012. ISSN 1053-5888. doi: 10.1109/MSP.2012.2205597.

Hinton, Geoffrey and Salakhutdinov, Ruslan. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313[5786):504 – 507, 2006.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long Short-Term Memory. *Neural Comput.*, 9[8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Karim, Fazle; Majumdar, Somshubra; Darabi, Houshang, and Chen, Shun. LSTM Fully Convolutional Networks for Time Series Classification. *CoRR*, abs/1709.05206, 2017.

Kingma, Diederik. *Variational inference and deep learning: A new synthesis*. PhD thesis, University of Amsterdam, 2017.

Kingma, Diederik P. and Ba, Jimmy. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Kingma, Diederik P. and Welling, Max. Auto-Encoding Variational Bayes. *CoRR*, abs/1312.6114, 2013. URL http://arxiv.org/abs/1312.6114.

Krizhevsky, Alex; Sutskever, Ilya, and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257.

LeCun, Yann. Obstacle to Progress in Deep Learning & AI. https://engineering.nyu.edu/news/revolution-will-not-be-supervised-promises-facebooks-yann-lecun-kickoff-ai-seminar, 2018.

LeCun, Yann; Bengio, Yoshua, and Hinton, Geoffrey E. Deep learning. *Nature*, 521[7553):436–444, 2015.

Lee, H. and Roberts, S. J. On-line novelty detection using the Kalman filter and extreme value theory. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, Dec 2008. doi: 10.1109/ICPR.2008.4761918.

Lei, Qi; Yi, Jinfeng; Vaculín, Roman; Wu, Lingfei, and Dhillon, Inderjit S. Similarity Preserving Representation Learning for Time Series Analysis. *CoRR*, abs/1702.03584, 2017.

Liu, Yongli; Chen, Jingli; Wu, Shuai; Liu, Zhizhong, and Chao, Hao. Incremental fuzzy C medoids clustering of time series data using dynamic time warping distance. *PLOS ONE*, 13[5):1–25, 05 2018. doi: 10.1371/journal.pone.0197499. URL https://doi.org/10.1371/journal.pone.0197499.

Luong, Minh-Thang; Pham, Hieu, and Manning, Christopher D. Effective Approaches to Attention-based Neural Machine Translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025.

Mahadevan, V.; Li, W.; Bhalodia, V., and Vasconcelos, N. Anomaly detection in crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1975–1981, June 2010. doi: 10.1109/CVPR.2010.5539872.

Malhotra, Pankaj; Vig, Lovekesh; Shroff, Gautam, and Agarwal, Puneet. Long Short Term Memory Networks for Anomaly Detection in Time Series. In ., 2015.

Malhotra, Pankaj; Ramakrishnan, Anusha; Anand, Gaurangi; Vig, Lovekesh; Agarwal, Puneet, and Shroff, Gautam. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *CoRR*, abs/1607.00148, 2016.

Malhotra, Pankaj; TV, Vishnu; Vig, Lovekesh; Agarwal, Puneet, and Shroff, Gautam. TimeNet: Pretrained deep recurrent neural network for time series classification. *CoRR*, abs/1706.08838, 2017.

Marques, Jorge S. *Reconhecimento de Padrões - Métodos Estatísticos e Neuronais*. IST Press, 2005.

Martinelli, Marco; Tronci, Enrico; Dipoppa, Giovanni, and Balducelli, Claudio. Electric Power System Anomaly Detection Using Neural Networks. In Negoita, Mircea Gh.; Howlett, Robert J., and Jain, Lakhmi C., editors, *Knowledge-Based Intelligent Information and Engineering Systems*, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30132-5.

Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

Ng, Andrew Y.; Rajpurkar, Pranav; Hannun, Awni Y.; Haghpanahi, Masoumeh, and Bourn, Codie. Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. *CoRR*, abs/1707.01836, 2017.

Parisotto, Emilio; Chaplot, Devendra Singh; Zhang, Jian, and Salakhutdinov, Ruslan. Global Pose Estimation with an Attention-based Recurrent Network. *CoRR*, abs/1802.06857, 2018.

Park, Daehyung; Hoshi, Yuuna, and Kemp, Charles C. A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder. *CoRR*, abs/1711.00614, 2017.

Pimentel, Marco A.F.; Clifton, David A.; Clifton, Lei, and Tarassenko, Lionel. A Review of Novelty Detection. *Signal Processing*, 99:215 – 249, 2014. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2013.12.026. URL `http://www.sciencedirect.com/science/article/pii/S016516841300515X`.

Reddi, Sashank J.; Kale, Satyen, and Kumar, Sanjiv. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ryQu7f-RZ`.

Rezende, Danilo Jimenez; Mohamed, Shakir, and Wierstra, Daan. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Xing, Eric P. and Jebara, Tony, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL `http://proceedings.mlr.press/v32/rezende14.html`.

Rifai, Salah; Vincent, Pascal; Muller, Xavier; Glorot, Xavier, and Bengio, Yoshua. Contractive Auto-encoders: Explicit Invariance During Feature Extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 833–840, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104587.

Rumelhart, David E.; Hinton, Geoffrey E., and Williams, Ronald J. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

Rush, Alexander M.; Chopra, Sumit, and Weston, Jason. A Neural Attention Model for Abstractive Sentence Summarization. *CoRR*, abs/1509.00685, 2015.

Schölkopf, Bernhard; Platt, John C.; Shawe-Taylor, John C.; Smola, Alex J., and Williamson, Robert C. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.*, 13[7):1443–1471, July 2001. ISSN 0899-7667. doi: 10.1162/089976601750264965. URL https://doi.org/10.1162/089976601750264965.

Shyu, M-L; Chen, S-C; Sarinnapakorn, K., and Chang, L. A novel anomaly detection scheme based on principal component classifier. In *IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with ICDM'03*, pages 171–179, 2003.

Siegelmann, Hava T. and Sontag, Eduardo D. Turing Computability with Neural Nets. *Applied Mathematics Letters*, 4[6):77 – 80, 1991. ISSN 0893-9659. doi: https://doi.org/10.1016/0893-9659(91)90080-F. URL http://www.sciencedirect.com/science/article/pii/089396599190080F.

Sölch, Maximilian. Detecting anomalies in robot time series data using stochastic recurrent networks. Master's thesis, University of Toronto, 2015.

Sölch, Maximilian; Bayer, Justin; Ludersdorfer, Marvin, and van der Smagt, Patrick. Variational Inference for On-line Anomaly Detection in High-Dimensional Time Series. *CoRR*, abs/1602.07109, 2016.

Song, X.; Wu, M.; Jermaine, C., and Ranka, S. Conditional Anomaly Detection. *IEEE Transactions on Knowledge and Data Engineering*, 19[5):631–645, May 2007. ISSN 1041-4347. doi: 10.1109/TKDE.2007.1009.

Srivastava, Nitish; Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised Learning of Video Representations using LSTMs. *CoRR*, abs/1502.04681, 2015.

Sutskever, Ilya. *Training Recurrent Neural Networks*. PhD thesis, University of Toronto, Toronto, Ont., Canada, Canada, 2013. AAINS22066.

Sutskever, Ilya; Vinyals, Oriol, and Le, Quoc V. Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. Technical report, University of Toronto, 2012.

van der Maaten, Laurens and Hinton, Geoffrey. Visualizing Data using t-SNE . *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz, and Polosukhin, Illia. Attention Is All You Need. *CoRR*, abs/1706.03762, 2017.

Villani, Cédric. *The Wasserstein distances*, pages 93–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9_6. URL https://doi.org/10.1007/978-3-540-71050-9_6.

Vincent, Pascal; Larochelle, Hugo; Bengio, Yoshua, and Manzagol, Pierre-Antoine. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390294. URL http://doi.acm.org/10.1145/1390156.1390294.

Wang, Fei; Jiang, Mengqing; Qian, Chen; Yang, Shuo; Li, Cheng; Zhang, Honggang; Wang, Xiaogang, and Tang, Xiaoou. Residual Attention Network for Image Classification. *CoRR*, abs/1704.06904, 2017.

Wang, Zhaoxia; Tong, Victor Joo Chuan; Xin, Xin, and Chin, Hoong Chor. Anomaly Detection through Enhanced Sentiment Analysis on Social Media Data. *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 917–922, 2014.

Williams, G.; Baxter, R.; He, Hongxing; Hawkins, S., and Gu, Lifang. A comparative study of RNN for outlier detection in data mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 709–712, Dec 2002. doi: 10.1109/ICDM.2002.1184035.

Xu, Haowen; Chen, Wenxiao; Zhao, Nengwen; Li, Zeyan; Bu, Jiahao; Li, Zhihan; Liu, Ying; Zhao, Youjian; Pei, Dan; Feng, Yang; Chen, Jie; Wang, Zhaogang, and Qiao, Honglin. Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. *CoRR*, abs/1802.03903, 2018. URL http://arxiv.org/abs/1802.03903.

Xu, Kelvin; Ba, Jimmy; Kiros, Ryan; Cho, Kyunghyun; Courville, Aaron C.; Salakhutdinov, Ruslan; Zemel, Richard S., and Bengio, Yoshua. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *CoRR*, abs/1502.03044, 2015.

Yeung, Dit-Yan and Ding, Yuxin. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36[1):229 – 243, 2003. ISSN 0031-3203. doi: https://doi.org/10.1016/S0031-3203(02)00026-2. URL http://www.sciencedirect.com/science/article/pii/S0031320302000262.

Appendix A

Publications

This Thesis motivated two scientific papers, which were developed during its execution. They are attached in this Appendix and are the following:

- João Pereira and Margarida Silveira. "Unsupervised Anomaly Detection in Energy Time Series Data using Variational Recurrent Autoencoders with Attention". *Accepted for Oral Presentation in the 17th IEEE International Conference on Machine Learning and Applications (ICMLA-18)*.
- João Pereira and Margarida Silveira. "Learning Representations from Healthcare Time Series Data for Unsupervised Anomaly Detection". *Accepted for Oral Presentation in the IEEE International Conference on Big Data and Smart Computing (BigComp-19)*.

Unsupervised Anomaly Detection in Energy Time Series Data using Variational Recurrent Autoencoders with Attention

João Pereira
Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal
joao.p.cardoso.pereira@tecnico.ulisboa.pt

Margarida Silveira
Institute for Systems and Robotics, Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal
msilveira@isr.tecnico.ulisboa.pt

***Abstract*—In the age of big data, time series are being generated in massive amounts. In the energy field, smart grids are enabling a unprecedented data acquisition with the integration of sensors and smart devices. In the context of renewable energies, there has been an increasing interest in solar photovoltaic energy generation. These installations are often integrated with smart sensors that measure the energy production. Such amount of data collected makes the quest for developing smart monitoring systems that can detect anomalous behaviour in these systems, trigger alerts and enable maintenance operations.**

In this paper, we propose a generic, unsupervised and scalable framework for anomaly detection in time series data, based on a variational recurrent autoencoder. Furthermore, we introduce attention in the model, by means of a variational self-attention mechanism (VSAM), to improve the performance of the encoding-decoding process. Afterwards, we perform anomaly detection based on the probabilistic reconstruction scores provided by our model.

Our results on solar energy generation time series show the ability of the proposed approach to detect anomalous behaviour in time series data, while providing structured and expressive representations. Since it does not need labels to be trained, our methodology enables new applications for anomaly detection in energy time series data and beyond.

***Index Terms*—Anomaly Detection, Variational Recurrent Autoencoder, Attention, Solar Photovoltaic Energy**

I. Introduction

One of the key assets of the smart grid is the data it collects. The data gathered from smart meters in the grid makes it possible to develop machine learning algorithms that can analyse and monitor the data collected and detect anomalous behaviour. With the integration of renewable energy sources such as solar photovoltaic, it is important to ensure reliability, security and correct operation of these systems in order to promote good performances and a long lifetime of the equipments.

The problem of finding patterns in data that do not conform to expected or normal behaviour is often referred to as Anomaly Detection (AD) [1]. Over time many approaches to anomaly detection have been proposed. In particular, with the progress made in deep learning, new frameworks to tackle the challenges of anomaly detection were developed. However, a significant amount of these approaches are based on supervised machine learning models that require (big) labelled datasets to be trained. In the context of applications such as energy, annotating large datasets is difficult, time-consuming or even too expensive, while it requires domain knowledge from experts in the field. The lack of labels is, indeed, one of the reasons why anomaly detection has been such a great challenge for researchers and practitioners.

Furthermore, some of the proposed methods do not consider the sequential nature of the data by assuming it is independent in time. Smart grid data is often sequential by nature and mostly time series and, hence, it is crucial to take into account the order and structure of the data.

The main contributions of this work can be summarized as follows:

- Unsupervised reconstruction-based model using a variational autoencoder with recurrent encoder and decoder;
- Variational self-attention mechanism to improve the encoding-decoding process;
- Generic framework for anomaly detection in time series data;
- Application to solar photovoltaic generation time series.

II. Background

In this section, we revise autoencoders, recurrent neural networks, attention mechanisms and autoencoder-based anomaly detection.

A. Autoencoder (AE)

Autoencoders [2, 3] are neural networks that aim to reconstruct their input. They consist of two parts: an *encoder* and a *decoder*. The encoder maps input data $\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}}$ to a latent space (or code) $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$ and the decoder maps back from latent space to input space.

The autoencoders training procedure is unsupervised and it consists of finding the parameters that make the reconstruction $\hat{\mathbf{x}}$ as close as possible to the original input $\mathbf{x}$, by minimizing a loss function that measures the quality of the reconstructions (e.g., mean squared error).

Typically the latent space $\mathbf{z}$ has a lower dimensionality than the input space $\mathbf{x}$ and, hence, AEs are forced to learn compressed representations of the input data. This characteristic makes them suitable for dimensionality reduction (DR) tasks, where they were proven to perform much better than other DR techniques, such as Principal Component Analysis [4].

B. Variational Autoencoder (VAE)

The variational autoencoder [5, 6] is a deep generative model that constrains the latent code $\mathbf{z}$ of the conventional AE to be a random variable distributed according to a prior distribution $p_\theta(\mathbf{z})$, usually a standard Normal distribution, $\mathrm{Normal}(\mathbf{0}, \mathbf{I})$. Since the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable for a continuous latent space $\mathbf{z}$, the variational inference technique is often used to find a deterministic approximation, $q_\phi(\mathbf{z}|\mathbf{x})$, of the intractable true posterior. The parameters of the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$, often called the *variational parameters*, are derived using neural networks (e.g., mean $\boldsymbol{\mu}_\mathbf{z}$ and variance $\boldsymbol{\sigma}^2_\mathbf{z}$, in the case of a Normal distribution).
Hence, the training objective of the VAE is to maximize the evidence lower bound (ELBO) on the training data log-likelihood. For a data point $\mathbf{x}$, the evidence lower bound is given by the following equation, where θ and ϕ are the encoder and decoder parameters, respectively.

$$\mathcal{L}_{\mathrm{ELBO}}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})\big)$$

The expectation in the equation above can be approximated by Monte Carlo integration. The second term represents the Kullback-Leibler divergence ($\mathcal{D}_{\mathrm{KL}}$) between the approximate posterior and the prior. The distribution for the likelihood is usually a multivariate Normal or $\mathrm{Bernoulli}$, depending on the type of data being continuous or binary, respectively.

C. RNNs, LSTMs and Bi-LSTMs

Conventional (feed-forward) neural networks make the assumption that data is independent in time. However, this assumption does not hold for sequential data, such as time series. Therefore, with time series data, recurrent neural networks (RNNs) are often used.
RNNs are powerful sequence learners designed to capture the temporal dependencies of the data by introducing *memory*. They read a sequence of input vectors $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ and, at each timestep t, they produce a hidden state $\mathbf{h}_t$. The main feature of RNNs is a feedback connection that establishes a recurrence mechanism that decides how the hidden states $\mathbf{h}_t$ are updated. In simple "vanilla" RNNs, the hidden states $\mathbf{h}_t$ are updated based on the current input, $\mathbf{x}_t$, and the hidden state at the previous timestep, $\mathbf{h}_{t-1}$, as $\mathbf{h}_t = f(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1})$. f is usually a $\tanh$ or $\mathrm{sigmoid}$ function and $\mathbf{U}$ and $\mathbf{W}$ are weight matrices to learn, shared across timesteps. The hidden state $\mathbf{h}_t$ can, thus, be interpreted as a summary of the sequence of input vectors up to timestep t. Given a sequence of hidden states $\mathbf{h}_t$, a RNN can generate an output, $\mathbf{o}_t$, at every timestep or produce a single output, $\mathbf{o}_T$, in the last timestep.

Despite the effectiveness of RNNs for modeling sequential data, they suffer from the vanishing gradient problem, that arises when the output at timestep t depends on inputs much earlier in time. Therefore, long short-term memory networks (LSTMs) [7, 8] were proposed to overcome this problem and they do so by means of a memory cell and three gates. The memory cell ($\mathbf{c}_t$) stores information about the input sequence across timesteps. The *gates* are functions that control the proportion of the current input to include in the memory cell ($\mathbf{i}_t$), the proportion of the previous memory cell to forget ($\mathbf{f}_t$) and the information to output from the current memory cell ($\mathbf{o}_t$). The memory updates, at each timestep t, are computed as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{x}_t) \tag{1}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{x}_t) \tag{2}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{x}_t) \tag{3}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{x}_t) \tag{4}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{5}$$

In the previous equations, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ and $\mathbf{h}_t$ denote the input gate, the forget gate, the output gate, the memory cell and the hidden state, respectively. $\odot$ denotes an element-wise product. The other parameters are weight matrices to be learned, shared between all timesteps.

LSTMs can still not integrate information from future instants of time and, therefore, bidirectional long short-term memory networks (Bi-LSTMs) [9] were proposed. Bi-LSTMs exploit the input sequence, $\mathbf{x}$, in both directions by means of two LSTMs: one executes a forward pass and the other a backward pass. Hence, two hidden states ($\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$) are produced for timestep t, one in each direction. These states act like a summary of the past and the future. The hidden states at similar timesteps are often aggregated into a unique vector $\mathbf{h}_t = \big[\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t\big]$ that represents the whole context around timestep t, typically through concatenation.

D. Sequence to Sequence Models and Attention Mechanisms

The sequence to sequence (Seq2Seq) learning framework [10, 11] is often linked with a class of encoder-decoder models, in which the encoder and decoder are RNNs. The encoder reads a variable-length input sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{T_\mathbf{x}}) \in \mathbb{R}^{T_\mathbf{x} \times d_\mathbf{x}}$ and converts it into a fixed-length vector representation (or context vector), $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$, and the decoder takes this vector representation and converts it back into a variable-length sequence $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{T_\mathbf{y}}) \in \mathbb{R}^{T_\mathbf{y} \times d_\mathbf{y}}$. In general, the learned vector representation corresponds to the final hidden state of the encoder network, which acts like a summary of the whole sequence. A particular instance of a Seq2Seq model is the Seq2Seq autoencoder, in which the input and output sequences are aligned in time ($\mathbf{x} = \mathbf{y}$) and, thus, have equal lengths ($T_\mathbf{x} = T_\mathbf{y}$).

Seq2Seq models have their weakness in tackling long sequences (e.g., long time series), mainly because the intermediate vector representation $\mathbf{z}$ does not have enough capacity to capture information of the entire input sequence $\mathbf{x}$.
Therefore, attention mechanisms were proposed to allow the decoder to selectively *attend* to relevant encoded hidden states. Several attention models were proposed in the past few years [12, 13] and, in general, they operate as follows. At each timestep t, during decoding, the attention model computes a context vector $\mathbf{c}_t$ obtained by a weighted sum of all the encoder hidden states. The weights of the sum, a_{ij}, are computed by a score function that measures the similarity between the currently decoded hidden state, $\mathbf{h}^{\mathrm{d}}_t$, and all the

encoded hidden states $\mathbf{h}^{\mathrm{e}} = (\mathbf{h}_1^{\mathrm{e}}, \mathbf{h}_2^{\mathrm{e}}, ..., \mathbf{h}_{T_{\mathbf{x}}}^{\mathrm{e}})$. Afterwards, these scores are normalized using the $\mathrm{softmax}$ function, so that they sum to 1 along the second dimension. The computation of the weights and the context vectors can be described as follows:

$$a_{ti} = \frac{\exp\left(\mathrm{score}(\mathbf{h}_t^{\mathrm{d}}, \mathbf{h}_i^{\mathrm{e}})\right)}{\sum_{j=1}^{T_{\mathbf{x}}} \exp\left(\mathrm{score}(\mathbf{h}_t^{\mathrm{d}}, \mathbf{h}_j^{\mathrm{e}})\right)} \tag{6}$$

$$\mathbf{c}_t = \sum_{j=1}^{T_{\mathbf{x}}} a_{tj} \mathbf{h}_j \tag{7}$$

Attention mechanisms were developed mainly for natural language processing (NLP) tasks and improved significantly the performance of Seq2Seq models in applications such as machine translation [12]. Even though *attention* has been mostly applied to NLP problems involving text data, it is suitable for other tasks dealing with other types of data such as time series and videos. In fact, *attention* is a natural extension of Seq2Seq models for any kind of sequential data.

E. Autoencoder-based Anomaly Detection

The main idea behind autoencoder-based anomaly detection is to focus on what is normal, rather than modelling what is anomalous. The autoencoder is trained to reconstruct data with normal pattern (e.g., normal time series) by minimizing a loss function that measures the quality of the reconstructions. After training, the model is able to reconstruct well data with normal pattern, while it fails reconstruct anomalous data, since it never saw them during training. The detection is performed using the reconstruction metrics (e.g., reconstruction error) as anomaly score. In other words, the model learns a normal data manifold and the distance between a given observation and the normal data manifold is used to compute anomaly scores, either in the latent space of representations, $\mathbf{z}$, or in the reconstructions space, $\hat{\mathbf{x}}$.

III. Related Work

The work on anomaly detection in time series data has increased significantly over the past few years and has benefited from the progress made in deep learning. In particular, Seq2Seq and autoencoder models have been applied with success to time series AD tasks. Using this framework, Malhotra *et al.* [14] proposed a prediction-based model based on LSTMs and used the distribution of the prediction errors to compute anomaly scores. However, this approach is not suitable for time series affected by external factors not captured by sensors, making them unpredictable. Later on, reconstruction-based approaches were proposed to overcome this limitation, such as [15], which try to reconstruct the input time series and use the reconstruction errors as anomaly scores. After the introduction of the variational autoencoder, Bayer and Osendorfer [16] used variational inference and RNNs to model time series data and introduced stochastic recurrent networks (STORNs), which were subsequently applied to anomaly detection in robot time series data [17]. An and Cho [18] proposed a method based on a VAE and introduced a novel probabilistic anomaly score that takes into account the variability of the data (the *reconstruction probability*). Recently, Park *et al.* [19] applied a LSTM-based variational autoencoder to AD in robot assisted feeding data and introduced a progress-based prior over the latent variables. Finally, Xu *et al.* [20] applied a VAE to AD in seasonal key performance indicators (KPIs) time series and provided a theoretical explanation for VAE-based anomaly detection.

IV. Proposed Model

In this section we describe our proposed approach, that relies on two fundamental stages: the reconstruction model (autoencoder) and the detection strategy. Let $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ denote a dataset composed of N independent sequences of observations. Each sequence $\mathbf{x}^{(n)}$ has T timesteps, i.e. $\mathbf{x}^{(n)} = (\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ..., \mathbf{x}_T^{(n)})$, and each observation at timestep t, $\mathbf{x}_t^{(n)}$, is a $d_{\mathbf{x}}$-dimensional vector. Therefore, the dataset $\mathcal{X}$ has dimensions $(N, T, d_{\mathbf{x}})$.

A. Variational Bi-LSTM Autoencoder

The model takes as input a sequence of observations $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$. We then apply a denoising autoencoding criterion [21] that consists on adding noise $\mathbf{n} \sim \mathrm{Normal}(\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{n}}^2 \mathbf{I})$ to the inputs and force the autoencoder to reconstruct the clean version of the input, $\mathbf{x}$, from the corrupted one, $\tilde{\mathbf{x}}$. Since it is a regularization technique, this phase is only active at training time. The encoder is parametrized using a Bi-LSTM with $\tanh$ activation that generates a sequence of hidden states in both directions, forward $\rightarrow$ and backward $\leftarrow$. The final encoder hidden states of both passes are concatenated with each other to produce the vector $\mathbf{h}_T^{\mathrm{e}} = \left[\overrightarrow{\mathbf{h}}_T^{\mathrm{e}}; \overleftarrow{\mathbf{h}}_T^{\mathrm{e}}\right]$.
The prior distribution $p_\theta(\mathbf{z})$ over the latent variables $\mathbf{z}$ is defined as an isotropic multivariate Normal, i.e. $p_\theta(\mathbf{z}) = \mathrm{Normal}(\mathbf{0}, \mathbf{I})$. The parameters of the approximate posterior - the mean $\boldsymbol{\mu}_{\mathbf{z}}$ and the co-variance $\boldsymbol{\Sigma}_{\mathbf{z}} = \boldsymbol{\sigma}_{\mathbf{z}}^2 \mathbf{I}$ - are derived from the final encoder hidden state $\mathbf{h}_T^{\mathrm{e}}$ using two fully connected layers with Linear and $\mathrm{SoftPlus}$ activations, respectively.
To simplify the implementation of the denoising criterion, we adopted the same approach as Park *et al.* [19] and define the approximate posterior given a corruption distribution around $\mathbf{x}$ with a single Gaussian, i.e. $\tilde{q}_\phi(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$, instead of a mixture of Gaussians as in [21]. The latent variables are obtained by sampling from the approximate posterior, $\mathbf{z} \sim \mathrm{Normal}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}} \mathbf{I})$, using the re-parametrization trick $\mathbf{z} = \boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\sigma}_{\mathbf{z}} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathrm{Normal}(\mathbf{0}, \mathbf{I})$ is an auxiliary noise variable and $\odot$ represents an element-wise product.

Furthermore, we integrate a special attention mechanism in the reconstruction model, that we call Variational Self-Attention Mechanism (VSAM). The self-attention model receives as input a sequence of encoded hidden states and outputs a sequence of context vectors $\mathbf{c}_t$, with the same length (T), each one of them computed as a weighted sum of all the encoded hidden states. In detail, the mechanism works as follows. First, the relevance of every pair of encoded hidden states $\mathbf{h}_i^{\mathrm{e}}$ and $\mathbf{h}_j^{\mathrm{e}}$ is scored (eq. 8) using the *scaled dot-product* similarity, employed in *Transformer* [22] (a neural

network model for NLP, based on a self-attention mechanism). The use of the dot-product as relevance measure makes the self-attention model more efficient than previous attention mechanisms that need to learn a similarity matrix.

$$s_{ij} = \text{score}(\mathbf{h}_i^{\text{e}}, \mathbf{h}_j^{\text{e}}) = \frac{(\mathbf{h}_i^{\text{e}})^T \mathbf{h}_j^{\text{e}}}{\sqrt{d_{\mathbf{h}^{\text{e}}}}} \tag{8}$$

In equation 8, $d_{\mathbf{h}^{\text{e}}}$ is the size of the encoder Bi-LSTM state. Second, the attention weights a_{ij} are computed by normalizing the scores over the second dimension, as in equation 9, where $\mathbf{a}_t = (a_{t1}, a_{t2}, ..., a_{tT})$. This normalisation ensures that, for each timestep t, $\sum_{j=1}^{T} a_{tj} = 1$.

$$\mathbf{a}_t = \text{softmax}(\mathbf{s}_t) \tag{9}$$

Finally, for deriving the new context-aware vector representations $\mathbf{c}_t$ we adopted a variational approach. This choice is motivated by the *bypassing phenomenon* pointed out by Bahuleyan *et al.* [23]. In fact, if the decoder has a direct and deterministic access to the encoder hidden states through attention, the latent code $\mathbf{z}$ may not be forced to learn expressive representations, since the self-attention mechanism could bypass most of the information to the decoder. This problem can be solved by applying to the context vectors $\mathbf{c}_t$ the same constraint applied to the latent variables of the VAE, by modelling them as random variables. To do so, we first compute deterministic context vectors, $\mathbf{c}_t^{\text{det}} = \sum_{j=1}^{T} a_{tj}\mathbf{h}_j$ and then transform them using another layer, similarly to [23]. The prior distribution over the context vectors is defined as a standard Normal, $p(\mathbf{c}_t) = \text{Normal}(\mathbf{0}, \mathbf{I})$, and the parameters of the approximate posterior $\tilde{q}_\phi^a(\mathbf{c}_t|\mathbf{x})$, $\boldsymbol{\mu}_{\mathbf{c}_t}$ and $\boldsymbol{\Sigma}_{\mathbf{c}_t}$, are derived in similar fashion to the latent variables $\mathbf{z}$, including the dimensionality ($d_{\mathbf{c}_t} = d_{\mathbf{z}}$). The final context vectors are sampled from the approximate posterior, $\mathbf{c}_t \sim \text{Normal}(\boldsymbol{\mu}_{\mathbf{c}_t}, \boldsymbol{\Sigma}_{\mathbf{c}_t})$.

The decoder is also a Bi-LSTM with $\tanh$ activation that receives, at each timestep t, a latent representation $\mathbf{z}$, shared across timesteps, and a context vector $\mathbf{c}_t$. Unlike other works that use a Normal distribution for $p_\theta(\mathbf{x}_t|\mathbf{z})$, we use a Laplace distribution with parameters $\boldsymbol{\mu}_{\mathbf{x}_t}$ and $\boldsymbol{b}_{\mathbf{x}_t}$. The practical implication of this choice is that the training objective aims to minimize an ℓ_1 reconstruction loss $\propto \|\mathbf{x}_t - \boldsymbol{\mu}_{\mathbf{x}_t}\|_1$ rather than an ℓ_2 reconstruction loss $\propto \|\mathbf{x}_t - \boldsymbol{\mu}_{\mathbf{x}_t}\|_2^2$. The ℓ_1-norm promotes sparse reconstruction errors. Such a choice is motivated by the assumption that anomalous observations are rare and sparse, which is, indeed, the case in several applications of interest. The outputs of the decoder are the parameters of the reconstructed distribution of the input sequence of observations, $\boldsymbol{\mu}_{\mathbf{x}_t}$ and $\boldsymbol{b}_{\mathbf{x}_t}$. These parameters are derived from the decoder Bi-LSTM hidden states using two fully connected layers with Linear and SoftPlus activations, respectively.

The loss function for a particular sequence $\mathbf{x}^{(n)}$ is given by:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}^{(n)}) = -\mathbb{E}_{\mathbf{z}\sim\tilde{q}_\phi(\mathbf{z}|\mathbf{x}^{(n)}),\mathbf{c}_t\sim\tilde{q}_\phi^a(\mathbf{c}_t|\mathbf{x}^{(n)})}\big[\log p_\theta(\mathbf{x}^{(n)}|\mathbf{z},\mathbf{c})\big] \\ + \lambda_{\text{KL}}\Big[\mathcal{D}_{\text{KL}}\Big(\tilde{q}_\phi(\mathbf{z}|\mathbf{x}^{(n)})\|p_\theta(\mathbf{z})\Big) \\ + \eta\sum_{t=1}^{T}\mathcal{D}_{\text{KL}}\Big(\tilde{q}_\phi^a(\mathbf{c}_t|\mathbf{x}^{(n)})\|p(\mathbf{c}_t)\Big)\Big]\end{aligned}$$

where λ_{KL} weights the reconstruction and KL losses and η balances the attention KL loss and the latent space KL loss. Figure 1 illustrates the proposed model.

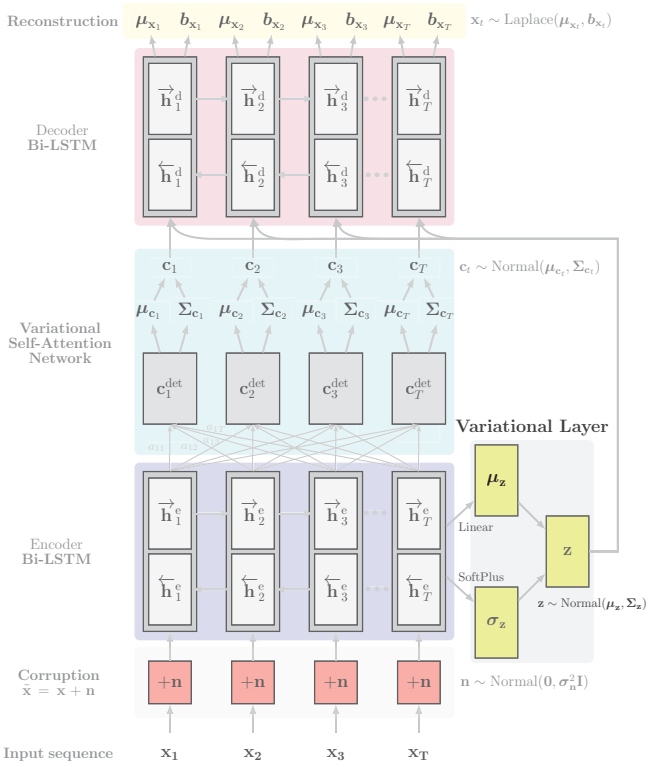


Fig. 1. Variational Bi-LSTM Autoencoder with Variational Self-Attention.

B. Anomaly Detection

The anomaly detection strategy is based on the following principle. The Variational Bi-LSTM Autoencoder with Attention is trained on normal sequences, so that it learns the normal pattern of data. At test time, normal sequences are expected to be well reconstructed whereas anomalous ones are not, since the model has not seen anomalous data during training.

Unlike deterministic autoencoders, the proposed model based on a VAE reconstructs the distribution parameters (mean $\boldsymbol{\mu}_{\mathbf{x}}$ and diversity $\boldsymbol{b}_{\mathbf{x}}$) of the input variable rather than the input variable itself. Therefore, it is possible to use probability measures as anomaly scores. One approach is to compute the *reconstruction probability*, introduced by An and Cho [18], that is an estimation of the reconstruction term of the VAE loss function by Monte Carlo integration.

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] \approx \frac{1}{L}\sum_{l=1}^{L}\log p(\mathbf{x}|\mathbf{z}_l)$$

The process can be described as follows. First, an input sequence $\mathbf{x}$ is propagated through the encoder and the posterior parameters $\boldsymbol{\mu}_{\mathbf{z}}$ and $\boldsymbol{\Sigma}_{\mathbf{z}}$ are obtained in a fully deterministic fashion. Then, L samples are drawn from an isotropic Gaussian distribution with these parameters. Each sample $\mathbf{z}_l$ is

propagated though the decoder that outputs the distribution parameters of the reconstruction. Afterwards, the log-likelihood of the input sample $\mathbf{x}$, given a latent code $\mathbf{z}_l$ drawn from the approximate posterior distribution is computed. Finally, the reconstruction probability is averaged over all $\mathbf{z}$ samples. Algorithm 1 summarizes the computation process.

Algorithm 1 Reconstruction Probability Score

Input: $\mathbf{x} \in \mathbb{R}^{T \times d_{\mathbf{x}}}$
Output: $ReconstructionProbability \in \mathbb{R}^T$
$(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{z}}) \leftarrow \text{Encoder}(\mathbf{x})$
for $l = 1$ to L **do**
  $\mathbf{z}_l \sim \text{Normal}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\Sigma}_{\mathbf{z}})$
  $(\boldsymbol{\mu}_{\mathbf{x}}^l, \mathbf{b}_{\mathbf{x}}^l) \leftarrow \text{Decoder}(\mathbf{z}_l)$
  $\text{score}^l \leftarrow \log p(\mathbf{x}|\boldsymbol{\mu}_{\mathbf{x}}^l, \mathbf{b}_{\mathbf{x}}^l)$
end for
$ReconstructionProbability \leftarrow \frac{1}{L}\sum_{l=1}^{L} \text{score}^l$
return $ReconstructionProbability$

The anomaly score itself is the negative reconstruction probability, so that the lower the reconstruction probability, the higher the anomaly score. There are several advantages of using the reconstruction probability instead of a deterministic reconstruction error which is commonly used in autoencoder-based anomaly detection approaches. The first one is that the reconstruction probability does not requires data-specific detection thresholds, since it is a probabilistic measure. Using such a metric provides a more intuitive way of analysing the results. The second one is that the reconstruction probability takes into account the variability of the data. Intuitively, anomalous data has higher variance than normal data and, hence, the reconstruction probability is likely to be lower for anomalous examples. The idea of using the variability of data for anomaly detection enriches the expressive power of the proposed model relatively to conventional autoencoders. Even in the case where normal and anomalous data can share the same expected value, the variability is different and, thus, provide an extra tool to distinguish anomalous examples from normal ones. For comparison purposes we also compute a (stochastic) reconstruction error (RE), given by equation 10.

$$RE_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}(\mathbf{x}) = \frac{1}{L}\sum_{l=1}^{L} \Big\| \mathbf{x} - \underbrace{\mathbb{E}\big[p_\theta\left(\mathbf{x}_l|\mathbf{z}_l\right)\big]}_{\boldsymbol{\mu}_{\mathbf{x}_l}} \Big\|_1 \qquad (10)$$

V. Training Framework

A. Data

The energy data in this work is a dataset $\mathcal{X}$ of univariate time series ($d_{\mathbf{x}} = 1$) of solar photovoltaic (PV) energy generation from several residential installations. The training data was obtained by selecting a subset $\mathcal{X}^{\text{normal}}$ of 1430 daily sequences with normal pattern (days without clouds and any kind of anomaly, where the energy generated is as expected). Samples were recorded each 15 min and, therefore, each (daily) sequence as 96 observations. The solar PV curves have a strong seasonality, with a predominant seasonal period of a day. We divided our dataset of normal sequences into two subsets - a training set $\mathcal{X}_{\text{train}}^{\text{normal}}$ and a validation set $\mathcal{X}_{\text{val}}^{\text{normal}}$ - with a splitting ratio of 80/20, respectively. The data was also normalised to the installed capacity, so that the range of observed values lies in the interval $[0, 1]$.

B. Modes

The proposed approach for anomaly detection can work under the following two modes:

- *Off-line Mode*: Training is performed with non-overlapping sequences of length T and the observations within a sequence share a unique representation in the latent space $\mathbf{z}$. All the scores for an input window are considered for detection and the score at a particular timestep t in a window can depend on future observations within the same window.
- *On-line Mode*: Training is executed using overlapping sequences obtained with a sliding window with a width T and a step size of 1. At test time, detection is performed without considering observations of future time instants, by feeding to the model a window of observations in which the last point corresponds to the current timestep t. The anomaly score at timestep t corresponds to the score of the last observation within each sequence. In this mode, for a long sequence with length L, $L - T + 1$ windows are produced, each one of them having its own representation in the $\mathbf{z}$-space. Since these windows overlap, the latent space will exhibit trajectories over time.

C. Optimization and Regularization

The models were implemented using the Keras deep learning library for Python [24], running on top of TensorFlow [25]. Optimization was performed using *AMS-Grad* optimizer [26], a variant of *Adam* [27], in mini-batches of size 200 (off-line mode) and 10000 (on-line mode), during 1500 epochs. The learning rate was 0.001. The full model has 274.958 parameters to optimize. We set the latent space dimensionality ($d_{\mathbf{z}}$) and the context vectors dimensionality ($d_{\mathbf{c}_t}$) to 3. The encoder and decoder Bi-LSTM both have 256 units, 128 in each direction. The noise added at the input level for the denoising autoencoding criterion has variance $\sigma_{\mathbf{n}}^2 = 0.1\sigma_{\mathbf{x}}^2$. We set the number L of Monte Carlo samples to 1 during training, following the work of Kingma and Welling [5]. The gradients were clipped by value with a clip value of 1.0. To prevent the KL-divergence vanishing problem, we applied a KL-annealing scheme [28] that consists on varying the weight λ_{KL} during training. By doing so, λ_{KL} is initially close to zero in order to allow accurate reconstructions in the early stages of training and is gradually increased to promote smooth encodings and diversity. The parameter η is 0.01. We also apply a sparsity regularizer in the hidden layer of the encoder Bi-LSTM [29], that penalizes the ℓ_1-norm of the activations with a weight of 10^{-8}.

Training was done on a single NVIDIA GTX 1080 TI GPU

with 11GB of memory, in a machine with an 8th generation i7 processor and 16GB of DDR4 RAM.

VI. Experiments and Results

In this section, we present the results of the experiments obtained with our proposed model. To illustrate the effectiveness of our approach, a few examples of solar energy generation curves representative of different patterns and behaviours ($\mathcal{X}_{\text{test}}$) were annotated, such as a normal sequence used as ground truth, a brief shading, a fault, a spike anomaly, an example of a daily curve where snow covered the surface of the PV panel and a sequence corresponding to a cloudy day.

We evaluate the training results using the training and validation losses, presented in Table I.

TABLE I
Training and validation losses.

Set	**Training** $(\mathcal{X}_{\text{train}}^{\text{normal}})$	**Validation** $(\mathcal{X}_{\text{val}}^{\text{normal}})$
Loss	-3.1457	-3.1169

The training and validation losses are similar, meaning that the model is not over-fitting to the training data and is being able to generalize to unseen (normal) sequences.

A. Anomaly Scores

Figure 2 shows some examples of solar PV generation daily curves with different kinds of patterns and the corresponding anomaly scores: the reconstruction probability (top bar) and the reconstruction error (bottom bar), both obtained by Monte Carlo integration using $L = 512$ samples.

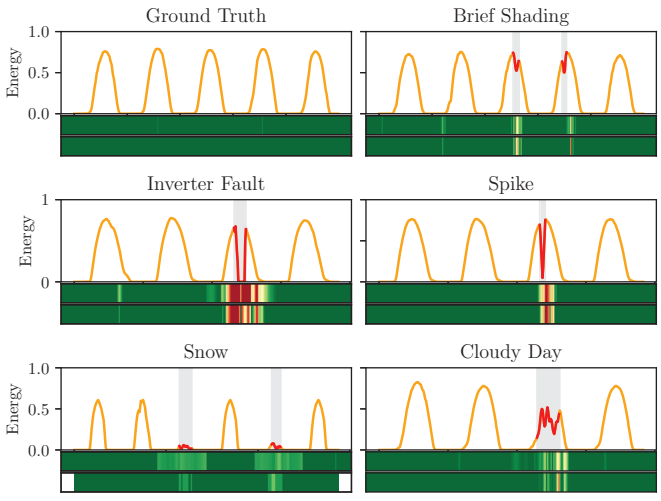


Fig. 2. Anomaly scores for some representative sequences (off-line mode, non-overlapping sequences with $T = 96$ timesteps).

B. Latent Space Analysis

The experiments were performed using a 3-dimensional latent space ($d_{\mathbf{z}} = 3$). For visualization purposes, we reduced the dimensionality of the latent space to 2D using Principal Component Analysis (PCA) and t-distributed Stochastic Neighbour Embedding (t-SNE) [30]. Figure 3 represents the latent space $\mathbf{z}$ of the training set containing only normal sequences ($\mathcal{X}_{\text{train}}^{\text{normal}}$). The label corresponds to the time instant of the last observation within each sequence.

Fig. 3. Latent space visualization of $\mathcal{X}_{\text{train}}^{\text{normal}}$ in 2D via t-SNE (left) and PCA (right). (on-line mode, training executed using overlapping sequences with $T = 32$ timesteps).

The latent space shows evidence that the model is mapping sequences aligned in time onto the same region of the $\mathbf{z}$-space and, more interestingly, it reveals a cyclic trajectory whose period matches exactly the seasonal period of the solar PV curves: one day. In other words, the model has learned the seasonal property of the data without being told of it and using training sequences with a length $32 < 96$, shuffled during training. Previous works have shown latent spaces with this behaviour, even though without analysing it, until the recent work of Xu *et al.* [20] that provided for the first time an explanation for this effect that they called *Time Gradient*.

In the context of time series anomaly detection, it is interesting to exploit the latent representations to find out how the representations of anomalous data compare with the ones of normal examples. Figure 4 shows the representations of the sequences that we annotated. Since the variational latent space is obtained by sampling from the approximate posterior, in this plot we represent the mean $\boldsymbol{\mu}_{\mathbf{z}} = \mathbb{E}\big[q_\phi(\mathbf{z}|\mathbf{x})\big]$ space, which is deterministically obtained from the encoder Bi-LSTM output.

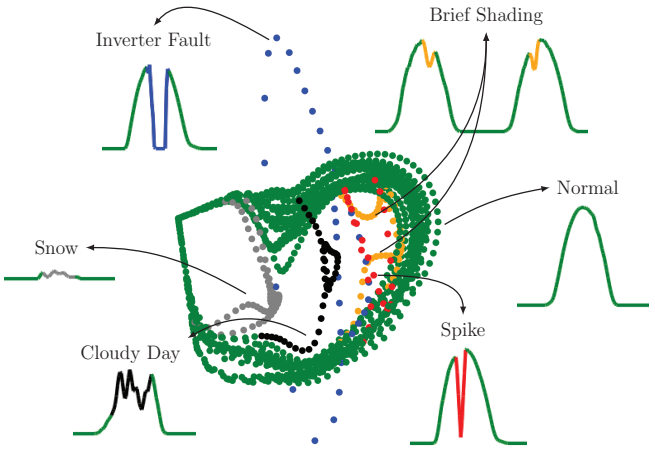


Fig. 4. Latent space visualization of $\mathcal{X}_{\text{test}}$ in 2D via PCA (on-line mode, training executed using overlapping sequences with $T = 32$ timesteps).

Figure 4 shows structured and expressive representations of sequences with various patterns. The *normal* examples (green) and the *anomalous* ones are represented differently in the space and there is clear a deviation of anomalous windows from the normal trajectory. The normal data have also slightly different trajectories in the space mainly because even though the curves have the same qualitative (normal) pattern, they are shifted in time due to different locations of the installations where the sun starts shining on the PV panel at different moments and also due to different inclinations.

C. Attention Visualization

The Variational Self-Attention Mechanism learns to pay more attention to particular encoded hidden states. Therefore, the attention model produces a 2D map for each sequence, with length T, that shows where the network is *putting* its attention. Figure 5 shows the attention maps for different test sequences with and without anomalies.

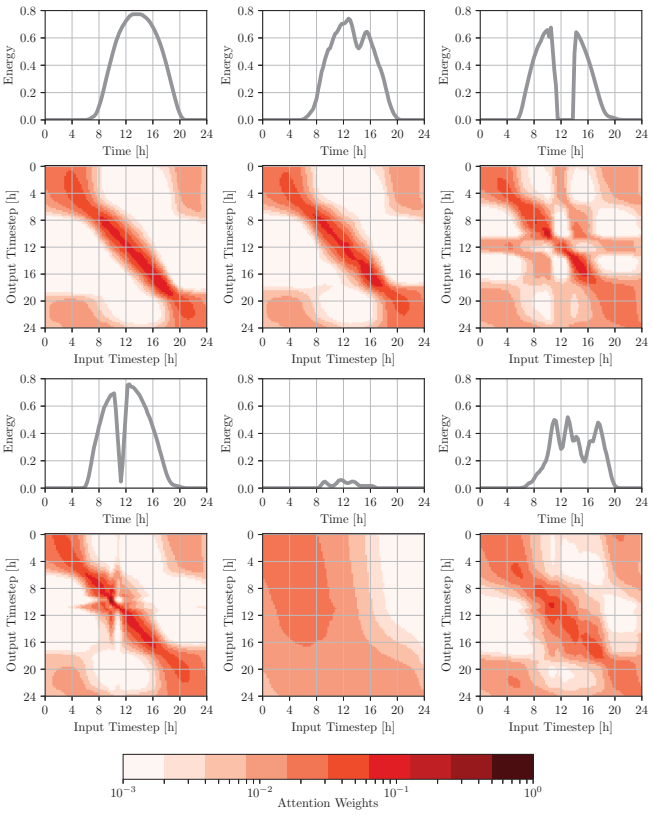


Fig. 5. Attention maps for sequences with different patterns. The attention weights are represented in a logarithmic scale.

The attention maps show evidence that the self-attention model is producing context-aware representations, which can be seen by the distribution of the attention weights in a small window around the first diagonal of the maps. This result supports the intuition that most of the temporal context of an observation in a time series lies in a narrow window around it. Furthermore, for different anomalies, the maps show different distributions of the attention weights. In some cases, the self-attention model is capturing dependencies between hidden states far in time. This conclusion validates the proposed reconstruction-based anomaly detection approach, since it tells that the network struggles to reconstruct well anomalous sequences while it tries to capture long-term dependencies in those.

It is also possible to visualize the context vectors $\mathbf{c}_t^{\text{det}}$ in the mean space $\boldsymbol{\mu}_{\mathbf{c}_t}$. The visualization, shown in Figure 6, was performed by reducing the dimensionality of $\boldsymbol{\mu}_{\mathbf{c}_t}$ to 2D using PCA. The labels represent the corresponding time instant t. Each context vector is computed as a weighted sum of all the encoder hidden states, so each one of them combines information from different time instants.

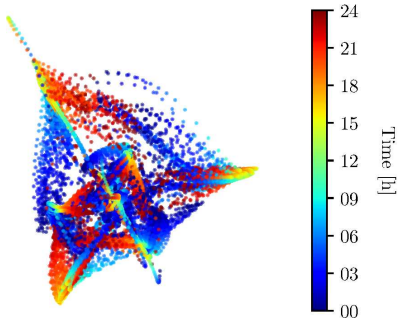


Fig. 6. Context vectors of the validation set $\mathcal{X}_{\text{val}}^{\text{normal}}$.

Figure 6 shows that context vectors aligned in time tend to be roughly represented in the same region of the space, while the mixed structure suggests that different sequences lead to context vectors that combine the encoder hidden states differently, using different attention weights.

VII. Conclusions and Future Work

In this paper, we presented a generic, unsupervised and scalable framework for anomaly detection in time series data that can operate both off-line and on-line. Our approach consists of a reconstruction model based on a variational autoencoder. We parametrized the encoder and decoder with recurrent neural networks to take into account the temporal dependencies of time series data. We also proposed a variational self-attention mechanism that aids the decoding process by allowing the model to pay more attention to particular encoded hidden states and, at the same time, provides a straightforward visualization scheme for the sequences.

Our results show that the model is able to detect anomalous patterns by using the probabilistic reconstruction metrics as anomaly scores. Moreover, the attention maps show evidence that the model changes its attention according to the kind of pattern of the input sequence. In particular, it *attends* differently depending on whether the sequence is normal or anomalous. A future line of work can exploit the usefulness of the attention maps for detection.

Furthermore, even though we applied the proposed model to solar PV generation univariate time series, it is suitable to

multivariate data as well, in which $\mathbf{x}$ can be a $d_{\mathbf{x}}$-dimensional vector of observations. Moreover, it can even be applied to other types of sequential data beyond time series, such as text and videos.

One of the major challenges of this work was the full absence of labels that is, actually, a common scenario in the context of real-world applications, such as energy. This motivated the unsupervised framework for anomaly detection that we proposed. The main advantage of following such an approach is that it can be applied to a wide range of time series data available. On the other hand, the main difficulty that we found due to the lack of labels was evaluation, since it is not possible to compute conventional classification metrics under this scenario. In fact, evaluation metrics for unsupervised anomaly detection algorithms, in the absence of labels and ground truth, remains a challenging problem where the literature is still scarce, even though some recent work has been done on the subject [31].

In this work, we focused on assigning an anomaly score to every observation in a sequence and not discriminating between different anomalies. However, the proposed approach can be extended to a multi-class framework, to allow distinguishing between anomalies. For this purpose, the detection phase might take into account the representations learned in the $\mathbf{z}$-space, which reveal to be expressive enough to allow for such a scenario.

Finally, in unsupervised anomaly detection, the concept of *normality* turns out to be hard to define in formal terms and might be prone to change/drift over time. Dealing with concept drift is a subject that we intend to address in future work.

ACKNOWLEDGEMENT

This work was funded by FCT project UID/EEA/50009/2013.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, pp. 533–536, 1986.

[3] H. Bourlard and Y. Kamp, "Auto-association by Multilayer Perceptrons and Singular Value Decomposition," *Biological Cybernetics*, vol. 59, no. 4, pp. 291–294, Sep 1988. [Online]. Available: https://doi.org/10.1007/BF00332918

[4] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[5] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *CoRR*, vol. abs/1312.6114, 2013. [Online]. Available: http://arxiv.org/abs/1312.6114

[6] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1278–1286. [Online]. Available: http://proceedings.mlr.press/v32/rezende14.html

[7] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[8] A. Graves, "Generating Sequences With Recurrent Neural Networks," *CoRR*, vol. abs/1308.0850, 2013.

[9] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition," in *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ser. ICANN'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 799–804. [Online]. Available: http://dl.acm.org/citation.cfm?id=1986079.1986220

[10] I. Sutskever, Q. V. Le, and O. Vinyals, "Sequence to Sequence Learning with Neural Networks," *CoRR*, vol. abs/1409.3215, 2014.

[11] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *CoRR*, vol. abs/1406.1078, 2014.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

[13] M. Luong, H. Pham, and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation," *CoRR*, vol. abs/1508.04025, 2015.

[14] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series," in *Proceedings of the 23rd European Symposium on Artificial Neural Networks*, 2015.

[15] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection," *CoRR*, vol. abs/1607.00148, 2016.

[16] J. Bayer and C. Osendorfer, "Learning Stochastic Recurrent Networks," *ArXiv e-prints*, Nov. 2014.

[17] M. Sölch, J. Bayer, M. Ludersdorfer, and P. van der Smagt, "Variational Inference for On-line Anomaly Detection in High-Dimensional Time Series," *CoRR*, vol. abs/1602.07109, 2016.

[18] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," *CoRR*, vol. 2015-2, 2015.

[19] D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder," *CoRR*, vol. abs/1711.00614, 2017.

[20] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications," *CoRR*, 2018. [Online]. Available: http://arxiv.org/abs/1802.03903

[21] Y. Bengio, D. J. Im, S. Ahn, and R. Memisevic, "Denoising Criterion for Variational Auto-Encoding Framework," *CoRR*, vol. abs/1511.06406, 2015.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *CoRR*, vol. abs/1706.03762, 2017.

[23] H. Bahuleyan, L. Mou, O. Vechtomova, and P. Poupart, "Variational Attention for Sequence-to-Sequence Models," *CoRR*, vol. abs/1712.08207, 2017.

[24] F. Chollet, "Keras," https://keras.io, 2015.

[25] M. Abadi *et al.*, " TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[26] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[28] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," *CoRR*, vol. abs/1511.06349, 2015.

[29] D. Arpit, Y. Zhou, H. Ngo, and V. Govindaraju, "Why Regularized Auto-Encoders learn Sparse Representation?" in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 136–144. [Online]. Available: http://proceedings.mlr.press/v48/arpita16.html

[30] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[31] N. Goix, "How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?" *ArXiv e-prints*, 2016.

Learning Representations from Healthcare Time Series Data for Unsupervised Anomaly Detection

João Pereira
Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal
joao.p.cardoso.pereira@tecnico.ulisboa.pt

Margarida Silveira
Institute for Systems and Robotics, Instituto Superior Técnico
University of Lisbon
Lisbon, Portugal
msilveira@isr.tecnico.ulisboa.pt

***Abstract*—The amount of time series data generated in Healthcare is growing very fast and so is the need for methods that can analyse these data, detect anomalies and provide meaningful insights. However, most of the data available is unlabelled and, therefore, anomaly detection in this scenario has been a great challenge for researchers and practitioners.**
Recently, unsupervised representation learning with deep generative models has been applied to find representations of data, without the need for big labelled datasets. Motivated by their success, we propose an unsupervised framework for anomaly detection in time series data. In our method, both representation learning and anomaly detection are fully unsupervised. In addition, the training data may contain anomalous data. We first learn representations of time series using a Variational Recurrent Autoencoder. Afterwards, based on those representations, we detect anomalous time series using Clustering and the *Wasserstein* distance.
Our results on the publicly available `ECG5000` electrocardiogram dataset show the ability of the proposed approach to detect anomalous heartbeats in a fully unsupervised fashion, while providing structured and expressive data representations. Furthermore, our approach outperforms previous supervised and unsupervised methods on this dataset.

***Index Terms*—Variational Recurrent Autoencoder, Representation Learning, Clustering, Electrocardiogram.**

I. Introduction

Detecting anomalies in time series data is an important problem of interest in applications such as healthcare, energy and cyber-security. Many Anomaly Detection (AD) approaches have been proposed over time [1, 2]. However, most of these approaches are based on supervised machine learning models that require (big) labelled datasets to be trained. In applications like healthcare, labels are often difficult to obtain, while the annotation process is time-consuming and requires domain-knowledge from experts in the field. Hence, the application of supervised models is limited by this constraint.
Furthermore, some previous anomaly detection approaches do not take into account the sequential nature of data by assuming it is independent and identically distributed in time. When dealing with time series data it is crucial to consider the temporal dependencies of the data.
Recently, there is a renewed interest in unsupervised learning, which is more and more foreseen to play an important role in the future of machine learning [3].
In this work, we propose an unsupervised framework for anomaly detection in sequential data, based on representation learning using a Variational Recurrent Autoencoder and anomaly detection in the representation's space via Clustering and the *Wasserstein* distance [4].

This paper is organized as follows. We start by revising Autoencoders, Variational Autoencoders and Recurrent Neural Networks. Then, we present a summary of recent approaches to anomaly detection in time series data. Afterwards, we introduce our proposed representation learning model and detection methodology. Finally, we present and analyse the results obtained with our model in electrocardiogram (ECG) time series.
Our contributions in this work can be summarized as:

- Unsupervised representation learning of time series data through a Variational Recurrent Autoencoder;
- Latent space-based detection using Clustering and the Wasserstein distance.

II. Background

In this section, we revise Autoencoders, Variational Autoencoders and Recurrent Neural Networks, including Long Short-Term Memory Networks.

A. Autoencoder (AE)

Autoencoders [5, 6] are neural networks trained in an unsupervised fashion that aim to reconstruct their input. They consist of two parts: an *encoder* and a *decoder*. The encoder maps input data $\mathbf{x} \in \mathbb{R}^{d_\mathbf{x}}$ to a latent code/representation $\mathbf{z} \in \mathbb{R}^{d_\mathbf{z}}$ and the decoder maps back from latent code to input space.
Training is executed by minimizing a reconstruction loss and, thus, by making the output of the decoder $\hat{\mathbf{x}}$ as close as possible to the original input $\mathbf{x}$.
Very often autoencoders are undercomplete, i.e. their latent code $\mathbf{z}$ has a lower dimensionality than the input space $\mathbf{x}$ and, hence, they are forced to learn compressed representations of the input data. This characteristic makes them suitable for dimensionality reduction (DR) tasks, where they were proven to work much better than other DR techniques, such as Principal Component Analysis [7].

B. Variational Autoencoder

The interest in autoencoders, and unsupervised learning in general, was strongly revived by the introduction of the variational autoencoder (VAE) [8, 9].

The variational autoencoder is a deep generative model that adds a new constraint on the code $\mathbf{z}$ of the autoencoder. The VAE assumes that the latent code $\mathbf{z}$ is a random variable distributed according to a prior distribution $p_\theta(\mathbf{z})$, which is often defined as a standard Normal distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$. However, the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable for continuous latent variables $\mathbf{z}$. Therefore, variational inference is applied to find a deterministic approximation $q_\phi(\mathbf{z}|\mathbf{x})$ of the intractable true posterior. Hence, the inference problem is tackled by solving an optimization one. The approximate posterior is usually a multivariate Normal distribution, $\mathcal{N}(\boldsymbol{\mu}_\mathbf{z}, \boldsymbol{\Sigma}_\mathbf{z})$, whose parameters are derived using neural networks.

The VAE training objective aims to maximize an evidence lower bound (ELBO) on the training data log-likelihood given by the following equation, where ϕ and θ are the encoder and decoder parameters, respectively.

$$\mathcal{L}_{\mathrm{ELBO}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})\big) \quad (1)$$

The distribution for the likelihood term is often a multivariate Normal or $\mathrm{Bernoulli}$, depending on the type of data being continuous or discrete, respectively.

The expectation may be approximated using Monte Carlo integration by drawing L samples from the approximate posterior.

$$\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \approx \frac{1}{L}\sum_{l=1}^{L} \log p_\theta\left(\mathbf{x}|\mathbf{z}_l\right) \quad (2)$$

C. Recurrent Neural Networks

Feed-forward Neural Networks assume data is independent in time. However, when dealing with sequential data such as time series this assumption does not hold and, thus, recurrent neural networks (RNNs) are often applied. Recurrent neural networks are powerful sequence learners designed to model the temporal dependencies of the data by introducing memory into the network. They receive a sequence of input vectors $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ and, at each timestep t, they compute a hidden state $\mathbf{h}_t$. The key aspect about RNNs is a feedback connection that builds a recurrence mechanism. This mechanism decides how the hidden states $\mathbf{h}_t$ are updated. In simple "vanilla" RNNs, the hidden states are updated based on the current input and the hidden state at the previous timestep, $\mathbf{h}_t = f(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1})$. The function f is usually a $\tanh$ or $\mathrm{sigmoid}$ and $\mathbf{U}$ and $\mathbf{W}$ are weight matrices shared across timesteps, to be learned. The hidden state $\mathbf{h}_t$ acts like a summary of the sequence of inputs already seen up to timestep t. RNNs can (optionally) produce an output based on the hidden state $\mathbf{h}_t$ at every timestep or just a single output in the last timestep T.

However, when dealing with sequences with long term dependencies, RNNs suffer from the vanishing gradient problem. This happens when the output at timestep t depends on inputs much earlier in time. Long Short-Term Memory (LSTM) networks [10, 11] are a variant of RNN proposed to overcome this limitation.

LSTMs integrate a memory cell and three gates that control the proportion of the current input to include in the memory cell $\mathbf{i}_t$, the proportion of the previous memory cell to forget $\mathbf{f}_t$ and the information to output from the current memory cell, $\mathbf{o}_t$. The updates of the memory at each timestep t are computed as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{x}_t) \quad (3)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{x}_t) \quad (4)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{x}_t) \quad (5)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{x}_t) \quad (6)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (7)$$

In the previous equations, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ and $\mathbf{h}_t$ denote the input gate, the forget gate, the output gate, the memory cell, and the hidden state. $\odot$ denotes an element-wise product.The other parameters are weight matrices to be learned, shared between all timesteps.

Despite the success of LSTMs for sequence modeling, they still can not integrate information from future timesteps. To solve this problem, Bidirectional Long Short-Term Memory networks (Bi-LSTMs) [12] were proposed. Bi-LSTMs exploit the input sequence in both directions by means of two LSTMs: one executes a forward pass and the other a backward pass. As a result, two hidden states are produced at each timestep t, one in each direction, $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$. Each one of these states acts like a summary of the past and the future. By concatenating both of them, a global hidden state $\mathbf{h}_t$ that represents the whole context around timestep t is obtained.

III. Related Work

The problem of finding sequences (e.g., ECG heartbeats) that do not conform with the normal pattern is often framed as a time series anomaly detection (AD) task, which is a particular instance of a classification problem (two-class). The work on AD has increased significantly over the past few years and has benefited from the progress made in the framework of deep learning (DL). In healthcare applications dealing with time series data in particular, the work on anomaly detection has been mostly based on (supervised) deep neural network models using either recurrent neural networks or convolutional neural networks (CNNs). In this context, Ng *et al.* [13] applied a 34-layer convolutional neural network for classification of ECG time series. Vig *et al.* [14] used long short-term memory networks for anomaly detection in ECG data. Malhotra *et al.* [15] introduced TimeNet, a sequence to sequence autoencoder model for time series feature extraction, and performed classification using a supervised classifier trained on the extracted features. Other works try to mix different neural network models, such as Karim *et al.* [16] that proposed an architecture that integrates both RNNs and CNNs for time series classification. On the unsupervised learning side, the amount of work developed in the framework of anomaly detection in time series

data is less than the one exploiting supervised models and the proposed approaches still do not yield impressive results. However, recently, there has been an increasing interest in adopting unsupervised learning models for anomaly detection, mainly in the framework of representation learning. In this line, Lei *et al.* [17] proposed a representation learning approach that converts time series of possibly unequal lengths to a matrix form while preserving pair-wise similarities between them and apply it to time series clustering and classification tasks. Aytekin *et al.* [18] used a feed-forward autoencoder for extracting representations of images and performed anomaly detection using clustering.

All in all, even though some of the aforementioned works attained state-of-the-art performances, the literature is still very focused on supervised learning models that heavily rely on good labels to be trained.

IV. Proposed Model

In this section, we present our proposed approach that is based on two fundamental steps: representation learning and detection. The main difference between our work and previous approaches is that both representation learning and anomaly detection are performed in an unsupervised fashion.

A. Representation Learning

Consider a dataset $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ composed of N observed sequences (e.g., time series), where each sequence n has length T, $\mathbf{x}^{(n)} = \left(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ..., \mathbf{x}_T^{(n)}\right)$, and each datapoint $\mathbf{x}_t^{(n)}$ is a $d_{\mathbf{x}}$-dimensional vector.

The proposed representation learning model is a Variational Recurrent Autoencoder that works as follows.

The model reads an input time series $\mathbf{x}^{(n)}$ with T timesteps. Afterwards, a local denoising criterion [19] is applied by adding noise to the inputs:

$$\tilde{\mathbf{x}} \sim p(\tilde{\mathbf{x}}|\mathbf{x}), \qquad p(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \boldsymbol{\sigma}_{\mathbf{n}}^2\mathbf{I}) \tag{8}$$

This corruption process, at the input level, forces the model to reconstruct the original input, $\mathbf{x}$, from a corrupted version of it, $\tilde{\mathbf{x}}$.

The encoder is parametrized by a bidirectional long short-term memory network of parameter ϕ that processes the input time series and produces a sequence of hidden states in both directions. The final hidden states of the forward ($\rightarrow$) and the backward ($\leftarrow$) passes generated by the encoder Bi-LSTM are, then, concatenated in a single vector $\mathbf{h}_T^e = \left[\overrightarrow{\mathbf{h}}_T^e; \overleftarrow{\mathbf{h}}_T^e\right]$. This global hidden state $\mathbf{h}_T^e$ is a fixed-length vector representation/summary of the entire sequence $\mathbf{x}$.

Similarly to Park *et al.* [20] we simplified the denoising criterion by modelling the posterior distribution given a corruption distribution around $\mathbf{x}$ with a single Gaussian, $\tilde{q}_\phi(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\tilde{\mathbf{x}})$.

The prior distribution over the latent variables, $p_\theta(\mathbf{z})$, is defined as an isotropic multivariate Normal distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The parameters $\boldsymbol{\mu}_{\mathbf{z}}$ and $\boldsymbol{\Sigma}_{\mathbf{z}}$ of the approximate posterior distribution $\tilde{q}_\phi(\mathbf{z}|\mathbf{x})$ are derived from the final encoder hidden state, $\mathbf{h}_T^e$, using two fully connected layers with Linear and $\mathrm{SoftPlus}$ activations, respectively. The $\mathrm{SoftPlus}$ function is used to ensure that the variance is parametrized as non-negative and activated by a smooth function.

The latent variables $\mathbf{z}$ are sampled from the approximate posterior and computed using the re-parametrization trick as follows,

$$\mathbf{z} = \boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\sigma}_{\mathbf{z}} \odot \boldsymbol{\epsilon} \tag{9}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is an auxiliary (external) noise variable and $\odot$ denotes an element-wise product.

The decoder (generative model) is another Bi-LSTM that receives as input a sample $\mathbf{z}$ drawn from the approximate posterior and outputs, at each timestep t, the parameters of the reconstruction of the input variable $\mathbf{x}$. The decoding distribution $p_\theta(\mathbf{x}_t|\mathbf{z})$ is defined as a multivariate Normal with diagonal co-variance matrix, $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}_t}, \boldsymbol{\Sigma}_{\mathbf{x}_t})$.

Both the encoder and the decoder Bi-LSTMs are activated by a $\tanh$ function.

The training objective is to minimize:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{x}^{(n)}) = &-\mathbb{E}_{\tilde{q}_\phi(\mathbf{z}^{(n)}|\mathbf{x}^{(n)})}\left[\log p_\theta(\mathbf{x}^{(n)}|\mathbf{z}^{(n)})\right] \\ &+ \lambda_{\mathrm{KL}}\mathcal{D}_{\mathrm{KL}}\big(\tilde{q}_\phi(\mathbf{z}^{(n)}|\mathbf{x}^{(n)})\|p_\theta(\mathbf{z}^{(n)})\big)\end{aligned} \tag{10}$$

We included a weight parameter λ_{KL} in order to adjust the trade-off between the reconstruction term and the KL-divergence term.

The expectation in the training objective is approximated by Monte Carlo integration. The log-likelihood of a particular sequence $\mathbf{x}^{(n)}$ decomposes across timesteps:

$$\log p_\theta\big(\mathbf{x}^{(n)}|\mathbf{z}^{(n)}\big) = \sum_{t=1}^{T} \log p_\theta\big(\mathbf{x}_t^{(n)}|\mathbf{z}^{(n)}\big) \tag{11}$$

Since the prior on the latent variables is defined as an isotropic multivariate Normal distribution, the KL-divergence term in the training objective has a closed form solution, given by equation 12, and does not require estimation.

$$\mathcal{D}_{\mathrm{KL}}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})\big) \approx 1/2\left[\mathrm{tr}(\boldsymbol{\Sigma}_{\mathbf{z}}) - \boldsymbol{\mu}_{\mathbf{z}}^T\boldsymbol{\mu}_{\mathbf{z}} - d_{\mathbf{x}} - \log(|\boldsymbol{\Sigma}_{\mathbf{z}}|)\right] \tag{12}$$

Figure 1 illustrates the proposed representation learning model.

B. Anomaly Detection

In this work, anomaly detection is performed on the representations provided by the Variational Bi-LSTM Autoencoder model. The representation learning model learns to map input data sequences $\mathbf{x}$ with different patterns into different regions of the space and, therefore, it is straightforward to use those representations to distinguish between normal and anomalous samples.

Given a set of latent representations, the goal of anomaly detection is to find out whether a given representation is *normal* or *anomalous*. For this purpose, we consider three different methodologies: detection via Clustering in the $\boldsymbol{\mu}_{\mathbf{z}}$ space $\big(\boldsymbol{\mu}_{\mathbf{z}} = \mathbb{E}\big[q_\phi(\mathbf{z}|\mathbf{x})\big]\big)$, detection using a metric based on the Wasserstein distance and detection using a supervised Support

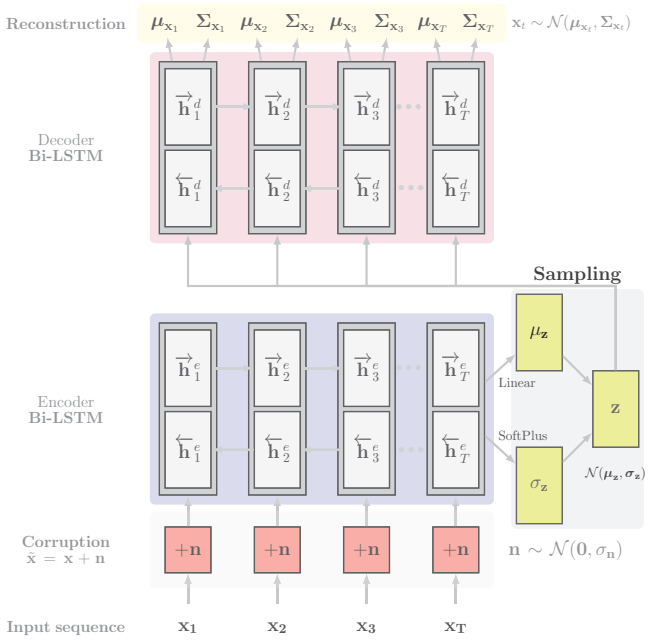


Fig. 1. Illustration of the proposed representation learning model: Variational Bi-LSTM Autoencoder.

Vector Machine (SVM) with linear kernel. The latter is used as a reference to compare the performance of unsupervised *vs* supervised anomaly detection. Note that, in this work, anomaly detection is approached as a two-class classification problem that is not focused on distinguishing between anomalies.

1) Clustering: The detection approach based on clustering consists on applying unsupervised clustering to the latent representations in the approximate posterior mean space ($\boldsymbol{\mu}_{\mathbf{z}}$) and aims to find the clusters that best describe the *normal* and *anomalous* classes of the data. The principle behind this technique in rooted on the assumption that most data used for training the representation learning model are *normal* and, therefore, the representations of anomalous samples will lie in a different region of the latent space. In other words, there will be a cluster containing the predominant (normal) examples and all the others will be represented far from those and assigned to the cluster of anomalous examples.

For this technique we applied three different clustering algorithms in the representations space: hierarchical clustering [21], spectral clustering [22] and k-means++ [23]. The clustering algorithms were set to find 2 clusters, one for each class (normal and anomalous). The output of these algorithms is, then, matched with the normal/anomalous classes by setting the cluster with higher number of data points assigned to be the *normal* one.

2) Wasserstein Distance: Since the model parametrizes either the mean $\boldsymbol{\mu}_{\mathbf{z}}$ and variance $\boldsymbol{\sigma}^2_{\mathbf{z}}$ of the latent variables (approximate posterior parameters), in the framework of anomaly detection, it makes sense to take into account the variability of the latent representations, $\boldsymbol{\sigma}^2_{\mathbf{z}}$, instead of just their expectation, $\boldsymbol{\mu}_{\mathbf{z}}$. This idea is motivated by the fact that even though the representations of normal and anomalous samples in the latent space might share the same mean, $\boldsymbol{\mu}_{\mathbf{z}}$, the variability of anomalous samples relatively to normal ones is likely to be higher, as pointed out by Cho *et al.* [24]. For obtaining an anomaly score, we compute the median *Wasserstein* distance between a test sample $\mathbf{z}^{\text{test}}$ and N_W other samples within the test set of latent representations, so that the similarity between the posterior distribution of a given sample and subset of other samples is used as anomaly score. This methodology works under the assumption often made in anomaly detection problems that most data are normal. The process can be described by equations 13 and 14.

$$W(\mathbf{z}^{\text{test}}, \mathbf{z}^i)^2 = \|\boldsymbol{\mu}_{\mathbf{z}^{\text{test}}} - \boldsymbol{\mu}_{\mathbf{z}^i}\|_2^2 + \|\boldsymbol{\Sigma}^{1/2}_{\mathbf{z}^{\text{test}}} - \boldsymbol{\Sigma}^{1/2}_{\mathbf{z}^i}\|_F^2 \quad (13)$$

$$\text{score}(\mathbf{z}^{\text{test}}) = \text{median}\{W(\mathbf{z}^{\text{test}}, \mathbf{z}^i)^2\}_{i=1}^{N_W} \quad (14)$$

In equations 13 and 14, W denotes the *Wasserstein* distance and the subscript 2 and F denote the ℓ_2-norm and the *Frobenius* norm, respectively.

V. Experiments

A. Data

We applied the proposed model to electrocardiogram (ECG) time series data. The dataset is the `ECG5000`, which was donated by Eamonn Keogh and Yanping Chen and is publicly available in the UCR Time Series Classification archive [25]. This dataset contains a set of $N = 5000$ univariate time series ($d_{\mathbf{x}} = 1$) with 140 timesteps ($T = 140$). Each sequence corresponds to one heartbeat. Five classes are annotated, corresponding to the following labels: *Normal* (N), *R-on-T Premature Ventricular Contraction* (R-on-T PVC), *Premature Ventricular Contraction* (PVC), *Supra-ventricular Premature or Ectopic Beat* (SP or EB) and *Unclassified Beat* (UB). In the original data source, the dataset is provided with a splitting into two subsets: a training set with $N_{\text{train}} = 500$ sequences and a test set with $N_{\text{test}} = 4500$ sequences. Both the training and test sets contain all classes of data, meaning that the training set contains both normal and anomalous data. Moreover, the classes are highly imbalanced: the normal class is the predominant one followed by the class with label R-on-T PVC. For validation purposes, we divided the original training dataset into two subsets - one for training the model ($\mathcal{X}_{\text{train}}$) and one for validation ($\mathcal{X}_{\text{val}}$) - with a splitting ratio of $80/20$, respectively. No further pre-processing was executed. Figure 2 shows the density of each class per set.

B. Training Setup

All the models were implemented using the Keras deep learning library [26], with TensorFlow backend.

Training was performed using *AMS-Grad* [27] optimiser, a variant of *Adam* [28], with a learning rate of 0.001. Gradient computation and weight updates are performed in mini-batches of size 500 during 1500 epochs. We set the latent space dimensionality, $d_{\mathbf{z}}$, to 5, corresponding to an encoding compression ratio of 28. The encoder and the decoder Bi-LSTM both have 256 units in total, 128 in each direction. The noise added at

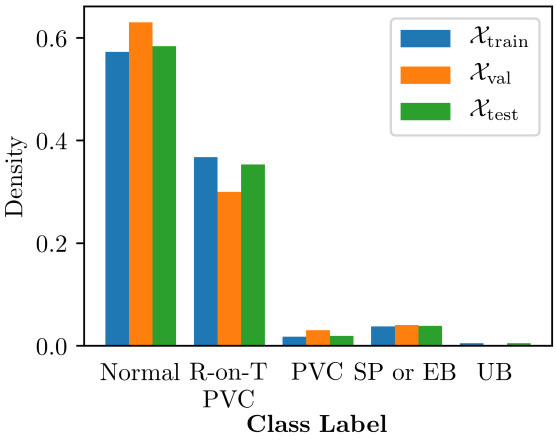


Fig. 2. Class densities per set.

the input level has a standard deviation $\sigma_{\mathbf{n}} = 0.8\sigma_{\mathbf{x}}$. We set the number of Monte Carlo samples L to 1 during training, following the work of Kingma and Welling [8]. To compute the Wasserstein anomaly score we use $N_W = 4000$. To promote stability during training, the gradients were clipped by value with a limit on their magnitude of 5.0. To prevent the KL-divergence term vanishing problem [29], we adopted a KL-annealing strategy in order to vary the weight λ_{KL} of the KL-divergence term in the loss function (equation 10). By doing so, the weight λ_{KL} is initially close to zero - to promote accurate reconstructions of $\mathbf{x}$ in the early stages of training - and gradually increased to encourage smooth encodings and diversity.

Furthermore, we adopted a sparse regularisation criterion to promote sparsity in the hidden layer of the Bi-LSTM encoder [30], by applying a penalty on the ℓ_1-norm of the activations, with a weight parameter of 10^{-7}. The total number of parameters to optimize is 273.420.

Training was executed on a NVIDIA GTX 1080TI graphics processing unit with 11GB of memory, in a machine with an 8^{th} generation i7 processor and 16GB of DDR4 RAM.

VI. Results

In this section, we present the results obtained with the proposed approach. We analyse the representations learned by the model and evaluate the anomaly detection results. All the results reported are evaluated on the test set $\mathcal{X}_{\text{test}}$ composed of 4500 sequences.

A. Latent Space Analysis

Figure 3 shows the latent space of the entire test set ($\mathcal{X}_{\text{test}}$) with 4500 sequences. Each datapoint is labelled with one of the five possible classes annotated. For visualization purposes, we reduced the dimensionality of the latent space from 5 to 2 dimensions using Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbour Embedding (t-SNE) [31]. For the t-SNE embedding, we set the perplexity parameter to 50.0 and the number of iterations to 2000.

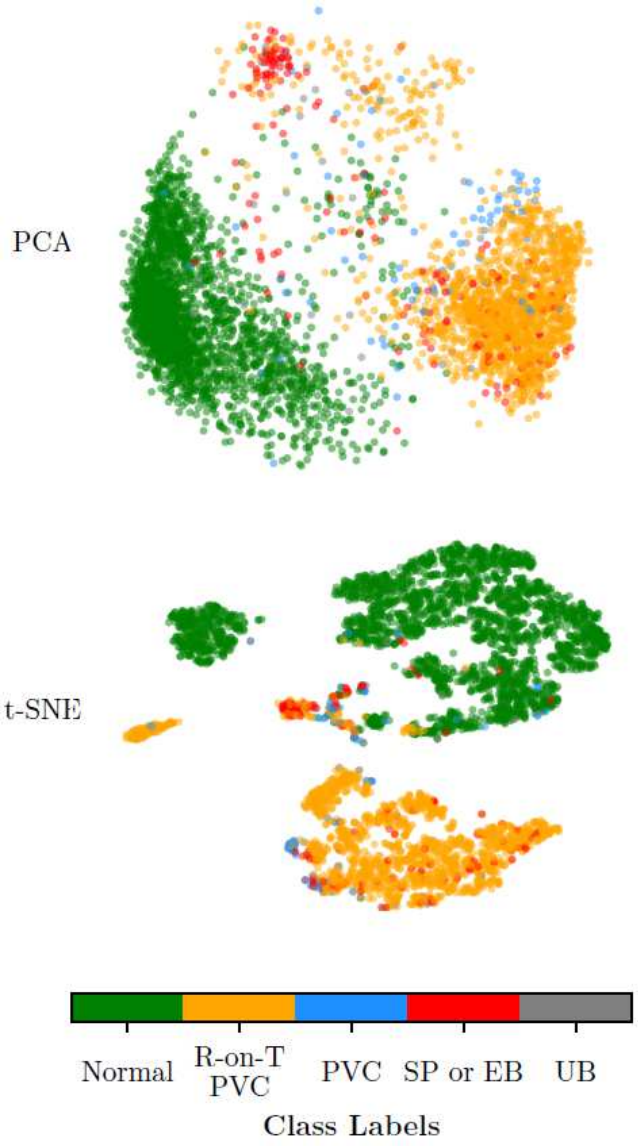


Fig. 3. Latent space visualization of $\mathcal{X}_{\text{test}}$ in 2D via PCA and t-SNE.

Figure 3 reveals a structured and expressive latent space. The sequences (heartbeats) of the *normal* class, represented in green, lie in a region of the latent space different from the *anomalous* ones, while similar heartbeats are mapped onto the same region of the space. Moreover, it is also clear that different anomalies are represented in distinct regions of the space. The anomalous heartbeats in blue and orange, which refer to Premature Ventricular Contractions, are represented close to each other. Interestingly, the anomaly with label "R-on-T PVC", represented in orange, has a smaller cluster apart from the larger one (top of the figure). This might be an interesting result to be analysed by experts.

B. Anomaly Detection

The anomaly detection results are evaluated using Area Under the Curve (AUC), Accuracy, Precision, Recall and F_1-score. These scores are weighted per-class. The process of computing the scores for the different detection methods proposed makes use of the anomaly labels available, but those are employed only for evaluation purposes. Since the output of a clustering algorithm might provide permuted labels, i.e. the cluster assignments may be permuted between the normal and anomalous classes, the assignment can be executed under the assumption that most data are normal, by matching the cluster with higher number of data points with the normal class.

In the methodology based on the *Wasserstein* distance, the AUC is computed by building the receiver operating characteristic (ROC) curve based on the false positive (FP) and true positive (TP) rates obtained for all possible detection thresholds, whereas the other metrics are computed for the detection threshold that leads to the higher scores. For the clustering approach, since it provides a hard classification result rather than an anomaly score, the AUC is computed for a ROC curve with the corresponding TP and FP rate.

In Table I we present the detection results evaluated on the test set, $\mathcal{X}_{\text{test}}$, using different clustering algorithms and a linear SVM. All results reported were averaged over 10 runs of both the representation learning and detection models.

TABLE I
SUMMARY OF THE RESULTS OBTAINED WITH THE PROPOSED MODEL.

Metric	Hierarchical	Spectral	*k*-Means	Wasserstein	SVM
AUC	0.9569	0.9591	0.9591	**0.9819**	0.9836
Accuracy	0.9554	0.9581	**0.9596**	0.9510	0.9843
Precision	**0.9585**	0.9470	0.9544	0.9469	0.9847
Recall	0.9463	0.9516	**0.9538**	0.9465	0.9843
F1-score	0.9465	0.9474	**0.9522**	0.9461	0.9844

The best *unsupervised* anomaly detection scores are emphasized in bold.

The *Wasserstein* distance-based anomaly metric yields the best unsupervised anomaly detection score in terms of AUC. The results obtained for the three clustering algorithms are roughly identical. This fact supports the idea that the key challenge in unsupervised anomaly detection is to learn good (expressive) representations of data. This is the reason why this work is strongly focused on representation learning.

Furthermore, the *Wasserstein* distance-based score outperforms clustering-based detection in terms of AUC and is similar in terms of the other metrics. This result is expected since this score is taking into account the variability of the representations in the latent space, rather than just their mean. The supervised Support Vector Machine performs very well, while the unsupervised detection methods stay roughly competitive. Anyway, all detection strategies attained relatively high detection scores.

Other works have used the same dataset mainly in a supervised multi-class classification framework, instead of anomaly detection that is a two-class problem. Even though both schemes can not be compared in general, since the dataset is highly imbalanced, with a large predominance of the normal and one of the anomalous classes (Figure 2), the multi-class classification problem is almost degenerated in a two-class one. Therefore, it is interesting to compare our method with the results reported in other works that considered different techniques. Table II summarizes the best scores obtained using both supervised and unsupervised learning models in several recent works and the best results for each metric are emphasized in bold.

TABLE II
RESULTS OBTAINED ON THE $ECG5000$ DATASET.

Source	S/U[a]	Model	AUC	Acc	F_1
Ours	S	VRAE+SVM	**0.9836**	**0.9843**	**0.9844**
	U	VRAE+Clust/W	**0.9819**	**0.9596**	**0.9522**
Lei *et al.* [17]	S	SPIRAL-XGB	0.9100	–	–
Karim *et al.* [16]	S	F-t ALSTM-FCN	–	0.9496	–
Malhotra *et al.* [33]	S	SAE-C	–	0.9340	–
Liu *et al.* [34]	U	oFCMdd	–	–	0.8084

[a]Supervised/Unsupervised; $- \equiv$ score not reported in the cited paper.

Most of the previous works that considered the same dataset use supervised machine learning models, while just one follows an unsupervised approach, up to the authors best knowledge. Under the two-class approximation made above, our unsupervised approach outperforms previous supervised learning models in every score reported.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an unsupervised approach to anomaly detection based on representation learning and latent space-based detection. Not only does the proposed representation learning model does not require labels to be trained but also the training data might contain anomalous data. The ratio of anomalous *vs* normal data used for training can be diverse, provided that most data are normal. The method can even deal with ratios of about 40% anomalous data, as was the case in this work. Since it does not depend on the existence of anomaly labels, the proposed approach is suitable for a wide range of applications where time series data are unlabelled, such as healthcare.

Even though the proposed model is generic to be applied to other types of sequential data, both univariate and multivariate, in this work, we focused on healthcare time series data, since it is an important field of application where the methodologies are still very focused on supervised machine learning models.

The results obtained are very encouraging, showing that it is possible to perform anomaly detection when no labels are available. In fact, our fully unsupervised approach attained results that compete with a conventional supervised learning model (the SVM) and outperforms supervised and unsupervised models recently proposed in other works. Nevertheless,

we think much work is still to be done to make unsupervised learning better in anomaly detection.

Finally, in this work, we tackled anomaly detection from the point of view of classifying normal and anomalous data. We plan to extend this framework to the multi-class case, to allow distinguishing between anomalies. We think the representations learned are structured and expressive enough to allow for such a scenario.

ACKNOWLEDGEMENT

This work was funded by FCT project UID/EEA/50009/2013.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[2] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A Review of Novelty Detection," *Signal Processing*, vol. 99, pp. 215 – 249, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016516841300515X

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[4] C. Villani, *The Wasserstein distances*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 93–111. [Online]. Available: https://doi.org/10.1007/978-3-540-71050-9_6

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, pp. 533–536, 1986.

[6] H. Bourlard and Y. Kamp, "Auto-Association by Multilayer Perceptrons and Singular Value Decomposition," *Biological Cybernetics*, vol. 59, no. 4, pp. 291–294, Sep 1988. [Online]. Available: https://doi.org/10.1007/BF00332918

[7] G. Hinton and R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[8] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *CoRR*, vol. abs/1312.6114, 2013. [Online]. Available: http://arxiv.org/abs/1312.6114

[9] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32. Bejing, China: PMLR, 22–24 Jun 2014, pp. 1278–1286. [Online]. Available: http://proceedings.mlr.press/v32/rezende14.html

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[11] A. Graves, "Generating Sequences With Recurrent Neural Networks," *CoRR*, vol. abs/1308.0850, 2013.

[12] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition," in *Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications - Volume Part II*, ser. ICANN'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 799–804. [Online]. Available: http://dl.acm.org/citation.cfm?id=1986079.1986220

[13] A. Y. Ng, P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, and C. Bourn, "Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks," *CoRR*, vol. abs/1707.01836, 2017.

[14] S. Chauhan and L. Vig, "Anomaly Detection in ECG Time Signals via Deep Long Short-term Memory Networks," *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–7, 2015.

[15] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "TimeNet: Pretrained Deep Recurrent Neural Network for Time Series Classification," *CoRR*, vol. abs/1706.08838, 2017.

[16] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," *CoRR*, vol. abs/1709.05206, 2017.

[17] Q. Lei, J. Yi, R. Vaculín, L. Wu, and I. S. Dhillon, "Similarity Preserving Representation Learning for Time Series Analysis," *CoRR*, vol. abs/1702.03584, 2017.

[18] Ç. Aytekin, X. Ni, F. Cricri, and E. Aksu, "Clustering and Unsupervised Anomaly Detection with L2 Normalized Deep Auto-Encoder Representations," *CoRR*, vol. abs/1802.00187, 2018.

[19] Y. Bengio, D. J. Im, S. Ahn, and R. Memisevic, "Denoising Criterion for Variational Auto-Encoding Framework," *CoRR*, vol. abs/1511.06406, 2015.

[20] D. Park, Y. Hoshi, and C. C. Kemp, "A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-based Variational Autoencoder," *CoRR*, vol. abs/1711.00614, 2017.

[21] Y. Zhao, G. Karypis, and U. Fayyad, "Hierarchical Clustering Algorithms for Document Datasets," *Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 141–168, Mar 2005. [Online]. Available: https://doi.org/10.1007/s10618-005-0361-3

[22] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01. Cambridge, MA, USA: MIT Press, 2001, pp. 849–856. [Online]. Available: http://dl.acm.org/citation.cfm?id=2980539.2980649

[23] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[24] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," *CoRR*, vol. 2015-2, 2015.

[25] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The UCR Time Series Classification Archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[26] F. Chollet, "Keras," https://keras.io, 2015.

[27] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ

[28] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[29] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," *CoRR*, vol. abs/1511.06349, 2015.

[30] D. Arpit, Y. Zhou, H. Ngo, and V. Govindaraju, "Why Regularized Auto-Encoders learn Sparse Representation?" in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 136–144. [Online]. Available: http://proceedings.mlr.press/v48/arpita16.html

[31] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE ," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[32] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised Deep Embedding for Clustering Analysis," *CoRR*, vol. abs/1511.06335, 2015. [Online]. Available: http://arxiv.org/abs/1511.06335

[33] P. Malhotra, A. Ramakrishnan, G. Anand, and L. Vig, "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection," *CoRR*, vol. abs/1607.00148, 2016. [Online]. Available: http://arxiv.org/abs/1607.00148

[34] Y. Liu, J. Chen, S. Wu, Z. Liu, and H. Chao, "Incremental Fuzzy C Medoids Clustering of Time Series Data using Dynamic Time Warping Distance," *PLOS ONE*, vol. 13, no. 5, pp. 1–25, 05 2018. [Online]. Available: https://doi.org/10.1371/journal.pone.0197499