SORBONNE UNIVERSITÉ

Unsupervised Anomaly Detection in Time-Series

Julien Audibert

A thesis submitted in fulfillment of the requirements for the Degree of
Doctor of Philosophy
in the Doctoral School N. 130 :
Computer Science, Telecommunications and Electronics of Paris
of the Sorbonne University

Committee in Charge:

Romain Tavenard	University of Rennes 2	*Reviewer*
Marco Lorenzi	University Cote d'Azur	*Reviewer*
Maria A. Zuluaga	EURECOM	*Advisor*
Xavier Alameda-Pineda	INRIA Grenoble	*Examiner*
Pietro Michiardi	EURECOM	*Examiner*
Maurizio Filippone	EURECOM	*Examiner*
Frédéric Guyard	Orange Labs	*Guest*
Sébastien Marti	Orange	*Guest*

Thursday 9th December, 2021

"What would life be if we had no courage to attempt anything ?"
by Vincent van Gogh

À mes parents, mon frère, et à celle qui partage ma vie.
Pour leur amour sans fin, leur soutien et leurs encouragements.

Abstract

Anomaly detection in multivariate time series is a major issue in many fields. The growing complexity of systems and the explosion of the quantity of data have made its automation essential. Methods based on Deep Learning have shown good results in terms of detection but do not meet the industrial requirements due to their long training and limited robustness. To meet the industrial needs, this thesis proposes a new unsupervised method for anomaly detection in multivariate time series called USAD based on an auto-encoder architecture and adversarial training. This method meets the requirements of robustness and speed of training of the industrial world while achieving state-of-the-art performance in terms of detection. However, deep neural network methods suffer from a limitation in their ability to extract features from the data since they only rely on local information. Thus, in order to improve the performance of these methods, this thesis presents a feature engineering strategy that introduces non-local information. This strategy increases the performance of neural network based approaches without increasing the training time. Given the good performance of deep learning methods for anomaly detection in multivariate time series in recent years, researchers have neglected all other methods in their benchmark, causing the complexity of the proposed methods to explode in the current publication. This lack of comparison with more conventional methods in the literature does not allow to assert that the progress reported in the benchmarks is not illusory and that this increasing complexity is necessary. To address this issue, this thesis proposes a comparison of sixteen methods for anomaly detection in multivariate time series grouped into three categories: Conventional methods, machine-learning methods and approaches based on deep neural networks. This study shows that there is no evidence that deep neural networks are a necessity to address this problem.

Keywords: Anomaly Detection, Time Series, Multivariate, Deep Learning, Unsupervised

Résumé

La détection d'anomalie dans les séries temporelles multivariées est un enjeu majeur dans de nombreux domaines. La complexité croissante des systèmes et l'explosion de la quantité de données ont rendu son automatisation indispensable. Les méthodes basées sur le Deep Learning ont montré de bons résultats en termes de détection mais ne répondent pas aux besoins industriels en raison de leur long apprentissage et de leur robustesse limitée. Pour répondre aux besoins industriels, cette thèse propose une nouvelle méthode non supervisée de détection d'anomalies dans les séries temporelles multivariées appelée USAD basée sur une architecture d'auto-encodeur et un entraînement adversaire. Cette méthode répond aux exigences de robustesse et de rapidité d'apprentissage du monde industriel tout en atteignant des performances de pointe en termes de détection. Cependant, les méthodes de réseaux de neurones profonds souffrent d'une limitation dans leur capacité à extraire des caractéristiques des données puisqu'elles ne s'appuient que sur des informations locales. Ainsi, afin d'améliorer les performances de ces méthodes, cette thèse présente une stratégie d'ingénierie des caractéristiques qui introduit des informations non-locales. Cette stratégie permet d'augmenter les performances des approches basées sur les réseaux de neurones sans augmenter le temps d'apprentissage. Compte tenu de la bonne performance des méthodes d'apprentissage profond pour la détection d'anomalies dans les séries temporelles multivariées ces dernières années, les chercheurs ont négligé toutes les autres méthodes dans leur benchmark, ce qui a provoqué l'explosion de la complexité des méthodes proposées dans les publications actuelles. Ce manque de comparaison avec des méthodes plus conventionnelles dans la littérature ne permet pas d'affirmer que les progrès rapportés dans les benchmarks ne sont pas illusoires et que cette complexité croissante est nécessaire. Pour répondre à cette problématique, cette thèse propose une comparaison de seize méthodes de détection d'anomalies dans les séries temporelles multivariées regroupées en trois catégories : Les méthodes conventionnelles, les méthodes d'apprentissage automatique et les approches basées sur les réseaux neuronaux profonds. Cette étude montre que rien ne prouve que les réseaux de neurones profonds soient une nécessité pour résoudre ce problème.

Mots-clés: Détection d'anomalies, séries temporelles, multivariées, univariées, apprentissage profond, non supervisé.

Acknowledgements

This intense three-year journey that is the PhD has allowed me to meet exciting people who have greatly helped and supported me on a daily basis. These people have allowed me to grow, learn, and progress both scientifically and personally.

I would like to sincerely thank my supervisor Maria A. Zuluaga. She allowed me to push my limits and encouraged me to push my thesis further. I thank her for her kindness and the many hours she devoted to me.

I thank Pietro Michiardi for the numerous advices and his precious help during the change of thesis director.

I thank Benoit Huet for having accepted to be my supervisor and for having allowed me to start this journey.

I would like to thank Frédéric Guyard for having accompanied me during these three years and for having helped me to build my advances and reasonings during enriching discussions.

I greatly thank Sébastien Marti, who made an incredible effort to launch this thesis and to convince the decision makers of Orange to fund it. I also thank him for his support throughout this journey and for the time he devoted to me.

I thank Raphaël Bernhard for his precious help and his knowledge in scientific writing. I also thank him for providing me with the necessary equipment for the good progress of this thesis.

Finally, I thank the management team of Flexper for having allowed me to finalize my thesis serenely.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivations

Time series and their analysis are becoming increasingly important due to the massive production of these data. Time series are used in a large number of fields such as industrial control systems [2], finance [3], and healthcare [4].

Time series analysis consists in extracting information from points arranged in a chronological order, i.e. the time series, and it can serve multiple purposes. The most common is to observe the history of a variable in order to make a forecast. This involves predicting future values of such variable based on previously observed values of it. Another common objective is to discover correlations between time series. This allows to understand the interactions between different variables in a system. Many other objectives explain the popularity of time series analysis such as the search for trends, cycles, seasonal variations or the detection of unusual behavior.

Detecting unexpected behaviors or patterns that do not conform to the expected behavior is an active research discipline called anomaly detection in time series [5]. Anomaly detection is an important field. It consists in detecting rare events or, more generally, observations that are aberrant and different from the majority of data. These rare events can be of various types and they are present in multiple and different domains (fraudulent financial transactions, medical problems or network intrusions). Detecting these rare events is a major issue for many fields. For example, detecting bank transaction fraud could save 32 billion dollars worldwide by 2020 [6]. It is therefore essential for the industry to be able to detect anomalies in their system.

This thesis focuses the critical task of anomaly detection. Specifically, it focuses on a subset of time series anomaly detection methods, which is unsupervised detection. Unlike supervised detection, unsupervised detection methods do not require a label associated to the data samples. The objective is then to detect behavior that would be different from previously observed data [7]. Finally, this thesis focuses on multivariate time series, since it is the most generic context as the univariate time series are only a special case of the multivariate context with $m = 1$ as presented in the section 2.1.1.

Over the last decade, there has been an increased enthusiasm around deep neural networks (DNNs) [8] thanks to their demonstrated ability to infer high-order correlations in complex data with potentially large volume and dimensionality [9, 10]. Anomaly detection in time series has not escaped this trend. DNN-based methods aim to learn deep latent representations of the multivariate time series to infer a model of variability, that is then used for anomaly grading in unseen data. The rationale behind the increased use of DNN architectures lies in the need of learning potentially complex data patterns underlying the temporal evolution of multivariate data. Thus, many approaches have emerged, mainly based on recurrent neural networks to capture temporal information [11, 12, 13]. However, these methods obtain good results at the expense of their training speed. Indeed, none of these methods takes into account the training time in their performance criteria. This is why it is necessary to develop methods with performances equivalent to the state of the art in terms of anomaly detection, while favoring architectures that allow fast and energy-efficient training.

As with any machine learning method, the performance of deep learning methods is correlated to the quality of the extracted features [14]. Feature engineering for augmenting time series data is usually done by bringing external but correlated information as an extra variate to the time series. This, however, requires domain knowledge about the measured process. Another strategy for machine learning methods is to create local features on the time series, such as moving averages or local maximum and minimum. Both strategies, as they are manual, are not very efficient, time consuming and require high domain knowledge expertise [15]. In theory, DNNs have emerged as a promising alternative given their demonstrated capacity to automatically learn local features, thus addressing the limitations of more conventional statistical and machine learning methods. Despite their demonstrated power to learn such local features, it has been shown that feature engineering can accelerate and improve the learning performance of DNNs [16] and an intrinsic limitation of DNN-learned features is that they rely solely on local information. However, there is currently no well established method in the literature that addresses this issue for time series.

As a result of the good performance demonstrated by DNNs in multiple fields [9, 10, 17, 18], in recent years there has been a boom of DNN-based methods for multivariate time series anomaly detection (Table 1.1). These works, however, have moved away from comparisons with more traditional methods, i.e. machine learning [19] and conventional/statistical methods (e.g. [1, 2, 13]) while suggesting methodological advances and improved performance of DNN-based methods. This trend has encouraged the community to develop even more complex models to improve the performance of DNN-based methods, without any theoretical or empirical evidence that these are superior to the more established body of methods in the literature.

DNN-based models are complex to train, involving the estimation of a large amount of parameters and requiring large training sample sizes and computational resources. Moreover, their complexity continues to grow as larger models continue to be developed. Instead, conventional models are simpler, lighter, easier to interpret, and often better adapted to the constraints of real-world applications. It is therefore crucial to determine if the complexity brought in by DNN-based methods is a necessary price to pay for a gain in performance or if the progress reported in recent years is illusory [20] and the use of conventional methods should be preferred. The lack of a general comparison covering all families of methods does not allow to answer this question and hinders the translation and use of DNN-based methods in real-world applications. A complete benchmark of such characteristics is currently missing in the literature.

Table 1.1: Peer-reviewed deep learning-based methods for anomaly detection in multivariate time series from 2018 to 2021

Methods	Description	Datasets
AE [21]	Autoencoder	MSL, SMAP, SMD, SWaT, WADI
Donut [22]	Variational Autoencoder	Private
Bagel [23]	Conditional Variational Autoencoder	Private
RGAN [24]	Recurrent Generative Adversarial Networks	MIT-BIH
OmniAnomaly [13]	Gated Recurrent Unit and Variational Autoencoder	MSL, SMAP, SMD, SWaT, WADI
BeatGAN [25]	Autoencoder and Generative Adversarial Network	CMU Motion Capture
MAD-GAN [12]	Generative Adversarial Networks	SWaT , WADI, KDDCUP99
LSTM-VAE [11]	LSTM-Variationnal Autoencoder	MSL, SMAP, SMD, SWaT, WADI
DeepAnT [26]	Convolutional neural network	Yahoo Webscope
MTS-DCGAN [27]	Deep Convolutional Generative Adversarial Network	Genesis D., Satellite, Shuttle, Gamma P.
USAD [2]	Adversely trained Autoencoders	MSL, SMAP, SMD, SWaT, WADI
FuseAD [28]	ARIMA and Convolutional neural network	Yahoo Webscope, NAB
Telemanom [29]	Vanilla LSTMs	SMAP , MSL
RADM [30]	Hierarchical Temporal Memory and Bayesian Network	NAB
DAGMM [1]	Deep Autoencoding Gaussian Mixture Model	MSL, SMAP, SMD, SWaT, WADI
MTAD-TF [31]	Convolutional and Graph Attention Network	MSL, SMAP, SMD

1.2 Contributions

This thesis is a CIFRE (Convention Industrielle de Formation par la Recherche) thesis which is a collaboration between Orange and EURECOM. Orange is a French telecommunications company. It has nearly 270 million customers worldwide. Thus, the contributions and methods developed during this thesis are intended to be integrated into the industrial environment of Orange. All the contributions of the thesis are the following:

- A fast and stable method called UnSupervised Anomaly Detection for multivariate time series (USAD) based on adversarialy trained auto-encoders. Its auto-encoder architecture makes it capable of unsupervised learning. The use of adversarial training and its architecture allow it to isolate anomalies while providing fast training.

- A novel feature engineering strategy to augment time series data in the context of anomaly detection using DNNs. The goal is two-fold. First, to transform univariate time series into multivariate time series to improve DNNs performance. Second, to use a feature engineering strategy that introduces non-local information into the time series, which DNNs are not able to learn. This is done by using a data structure called Matrix-Profile as a generic non-trivial feature. Matrix-Profile allows to extract non-local features corresponding to the similarity among the sub-sequences of a time series. The performance compared to each individual method shows that the method achieves better performance without increasing the computation time.

- A study of the performance of anomaly detection for sixteen conventional, machine learning-based and deep neural network-based approaches including USAD on five open real-world datasets. The analysis and comparison of the performance of each of the sixteen methods, shows that no family of methods outperforms the others. While deep neural networks appear to perform better when the dataset contains contextual anomalies or when the datasets are large, conventional techniques perform better when the dataset is small. Thus, it is impossible to claim that deep neural networks are superior to previous methods and the community should reincorporate the three categories of methods in the anomaly detection in multivariate time series benchmarks.

1.3 Structure of thesis

The content of the following chapters is summarized in this section.

- Chapter 2 is divided into two main parts. The first one presents the time series and their characteristics. The second part is dedicated to the anomaly detection in time series and presents a state of the art of the methods classified in three main categories: Conventional, Machine-Learning and Deep-Learning methods.
- Chapter 3 introduces a unsupervised anomaly detection method on multivariate time series composed of an adversely trained auto-encoders architecture and shows the performance of the method on five real-world open datasets as well as on Orange's proprietary data.
- Chapter 4 presents a feature engineering strategy that transforms univariate time series into multivariate ones by introducing non-local information and shows that this strategy addresses a limitation of Deep Neural Networks, and demonstrates that the combination outperforms each method individually without increasing computational time.

- Chapter 5 questions the need for ever more complex methods based mainly on Deep Neural Networks for anomaly detection in multivariate time series and proposes a study on sixteen methods belonging to the three categories presented in chapter 2.The performance analysis shows that none of the three categories outperform the others. And is discussed the possible performance illusion of Deep Neural Networks based methods in multivariate time series anomaly detection benchmarks.
- Finally, Chapter 6 summarizes the main contributions presented in this work and present some thoughts on the possible continuation of this research.

1.4 Publications

This thesis is based on research that has led to the publication of articles. Portions of the contents that appear in this dissertation have been published before in:

- Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. *USAD: UnSupervised Anomaly Detection on Multivariate Time Series.* In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20).
- Julien Audibert, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. *From Univariate to Multivariate Time Series Anomaly Detection with Non-Local Information.* In Proceedings of the 6th Workshop on Advanced Analytics and Learning on Temporal Data at ECML PKDD 2021.
- Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. *On the Benefits of Deep Neural Networks for Multivariate Time Series Anomaly Detection* under review at Pattern Recognition 2021.

1.5 Challenge participation

Part of the methods developed in this thesis participated in the KDDCUP2021, on the challenge "Multi-dataset Time Series Anomaly Detection". The participating method obtained the 16th place over 565 participants.

Chapter 2

Anomaly detection in time-series

Contents

This chapter presents in a first part what are time series and their characteristics. Then in a second part, the problem of anomaly detection in time series is formalized, followed by a description of the different methods to solve this problem.

2.1 Time Series

A time series is defined as a sequence of ordered continuous values, which represents the evolution of a numerical variable over time. It is the measurement of a system evolving in time with numerical attributes: for example, the temperature of a computer server, the value of a company's stock or the electrical activity of the heart (ECG). A time series is, therefore, any sequence of observations indexed by time. A time series carries a lot of information about the measured system. This information can be used to ensure the proper functioning of the system.

2.1.1 Univariate vs Multivariate

A univariate time series is a set of measured values that model and represent the behavior of a process over time. A multivariate time series is a set of measurements correlated with each other over time, which models and represents the behavior of a multivariate process in time.

More formally, a univariate time series is a sequence of data points

$$\mathcal{T} = \{x_1, \ldots, x_T\},$$

each one being an observation of a process measured at a specific time t. Univariate time series contain a single variable at each time instant, while multivariate time series record more than one variable at a time; Multivariate time series are denoted by $\mathcal{T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, $\mathbf{x} \in \mathbb{R}^m$.

2.1.2 Decomposition of a time series

A time series ($\mathcal{T}$) is considered to be the result of different fundamental components: Trend, seasonality, level and noise.

2.1.2.1 Trend

The trend (Z_t) represents the long-term evolution of the series studied. It reflects the "average" behavior of the series. For example, the series shown in Figure 2.1 shows an increasing trend from zero to 2000 and a decreasing trend from 2000 to 4000.

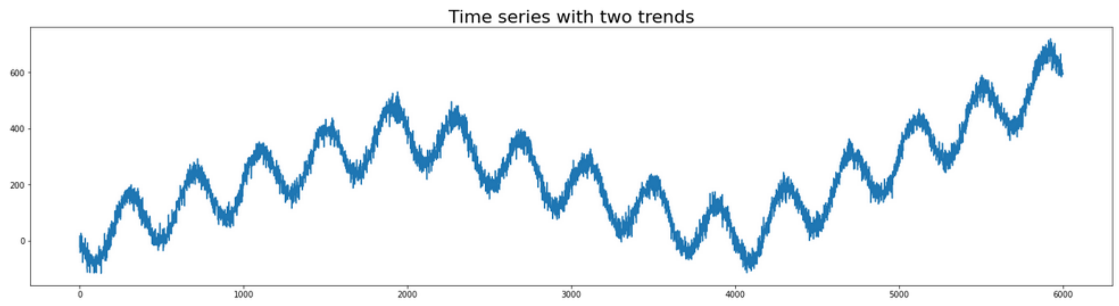

Figure 2.1: Time series with an increasing trend until the 2000 point, a decreasing trend from the 2000 point to the 4000 point and an increasing trend again after the 4000 point.

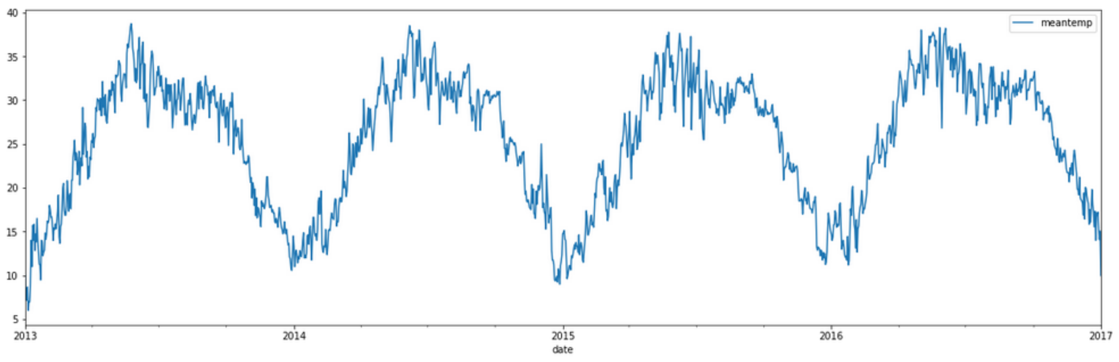

Figure 2.2: Time series representing the average temperature (°C) each day between 2013 and 2017.

2.1.2.2 Seasonality

Seasonal compounding or seasonality corresponds to a phenomenon that repeats itself at regular time intervals (periodicity). For example, the Figure 2.2 shows an annual seasonality. We can see four periods between 2013 and 2017.

2.1.2.3 Level

The level of a time series corresponds to the mean of the time series. If a time series has a trend, then its level changes.

2.1.2.4 Noise

The residual component or noise corresponds to irregular fluctuations, generally of low intensity but of a random nature. For example, the Figure 2.3 shows a time

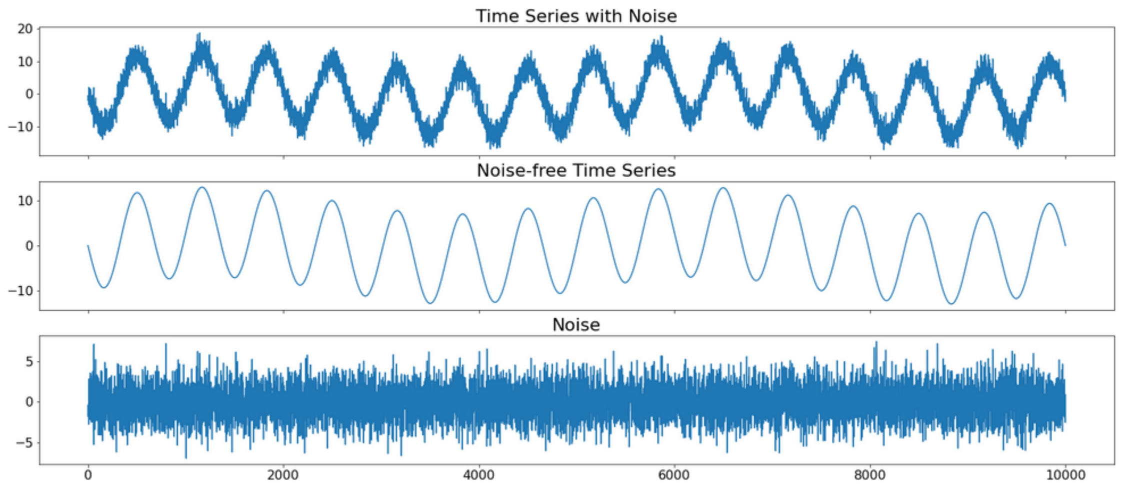

Figure 2.3: Top: Time series with noise. Middle: Time series without noise. Bottom: Noise.

series (top), then its decomposition, i.e. the time series without noise (middle) and the noise (bottom)

2.1.3 Stationarity

The notion of stationarity characterizes the capacity of a process to decorrelate itself completely from the temporal index. The time series $\mathcal{T}$ is said to be stationary at low order, or stationary at second order if the first (mean) and second (variance and autocovariance) moments of the process exist and are independent of t.

Stationarity is a property of stability, the distribution of $\{x_1, \ldots, x_t\}$ is identical to the distribution of $\{x_2, \ldots, x_{t+1}\}$. The series oscillates around its mean with a constant variance. The link between x_t and x_{t-k} then depends only on the interval k and not on the date t.

2.2 Anomaly Detection

An anomaly can be defined as an observation that is unexpected with respect to a set of other pre-established observations considered as normal. More formally, in a set D containing n observations noted x_i, then $x_p \in A$ will be considered as abnormal if it differs, by its characteristics, from the other observations, i.e. from those contained in the set $A \setminus \{x_p\}$. The definition of the term anomaly is specific to the use case. The most common one in the field of detection is an observation which is different from the others by its singularity: it could result from a set of rules which are different from the other observations [32]

Anomalies in time series, also called outliers, are points or sequences of points that do not correspond to normal behavior [33]. The concept of normal behaviour is difficult to formalize. Therefore, another possible definition for anomalies could be a pattern in data that is not expected in comparison to what has been seen before [33]. In fact, an implicit assumption is that anomalies are rare events. Anomalies should not be confused with the noise present in the time series. Noise is a phenomenon which, unlike anomalies, has less interest in being analyzed.

However, anomalies may indicate a significant problem in several applications. For example, an anomaly in industrial control systems may indicate a malfunction, financial anomalies may be the result of a fraud, or they may indicate diseases in healthcare. Being a critical task, there is a wide range of methods that have been developed to address it [34, 35].

Anomaly detection refers to the task of identifying an unseen observation $\hat{x}_t$, $t > T$, based on the fact that it differs significantly from $\mathcal{T}$, thus assuming that $\mathcal{T}$ contains only normal points. The amount by which the unseen sample $\hat{x}_t$ and the normal set $\mathcal{T}$ differ is measured by an anomaly score, which is then compared to a threshold to obtain an anomaly label.

2.2.1 Types of Anomalies in Time Series

Chandola et al. [33] propose to classify time series anomalies into three types, point, contextual and collective anomalies.

- **Point anomalies.** This is the simplest type of anomaly. It corresponds to a point that differs from the rest of the data (Figure 2.4).
- **Contextual anomalies.** A contextual anomaly can be defined as follows : A data point is anomalous in a specific context, but is considered normal in another context, meaning that observing the same point through different contexts does not always give a sign of abnormal behavior. For example, a temperature of 30°C during summer is normal, while the same temperature in winter is abnormal. Figure 2.5 illustrates a contextual anomaly, where the

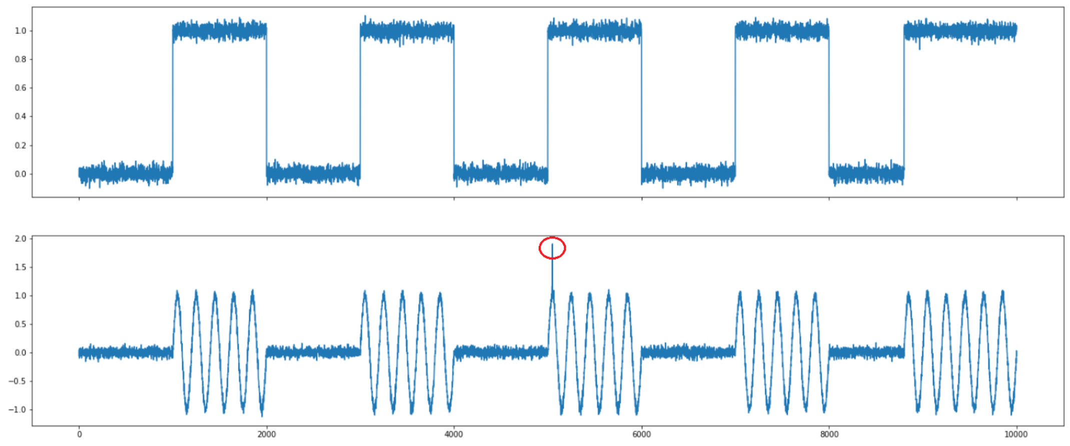

Figure 2.4: An example of a point anomaly (in red)

values of the two time-series are not abnormal taken individually, but, seen in context, the values of the bottom time-series should be close to 0.

- **Collective anomalies** A collective anomaly corresponds to a group of anomaly points. Each individual point might not be an anomaly, but their appearance together is an anomaly (Figure 2.6).

2.2.2 Supervised vs Unsupervised method

Supervised learning consists of input variables x and an output variable Y. You use an algorithm to learn the mapping function f from the input to the output.

$$Y = f(x)$$

The goal is to learn the mapping function such that when you have new input data x, you can predict the output variables Y for that data.

Unsupervised learning involves having only input data x and no corresponding output variables. The goal of unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

In supervised anomaly detection, if we want a model to be able to detect anomalies, it must characterize the system very precisely both in normal behavior and in the presence of anomalies. However, normal behaviors can be multiple, as well as behaviors in the presence of anomalies. The large number of system behaviors leads to the need to provide a large amount of labeled data to capture a maximum of

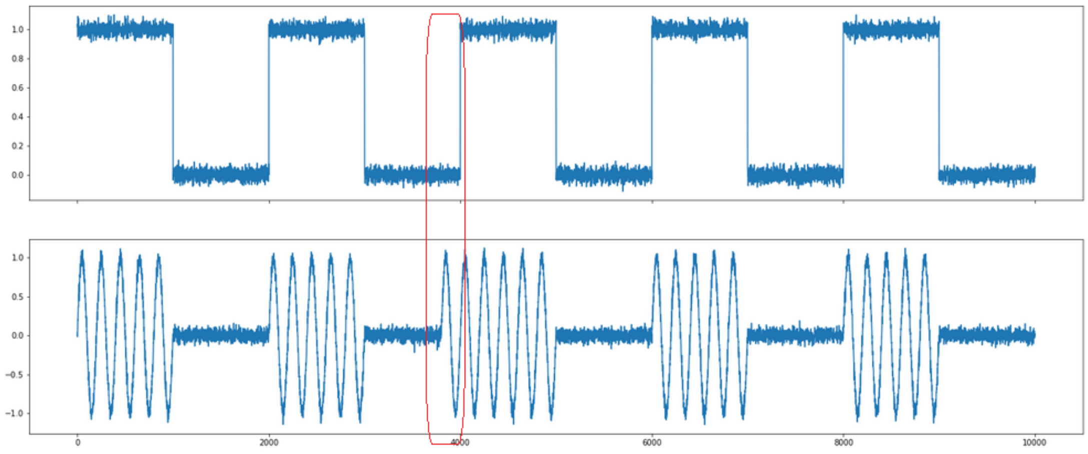

Figure 2.5: An example of a contextual anomaly (in red)

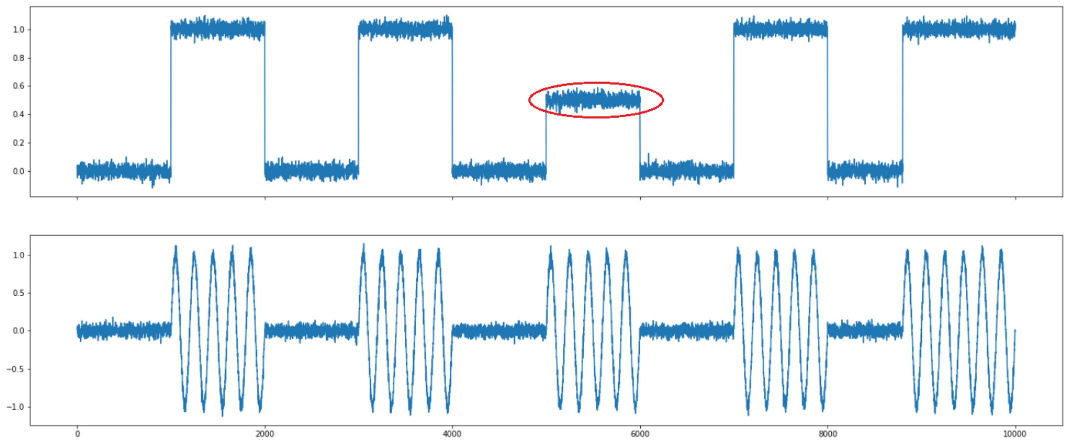

Figure 2.6: An example of a collective anomaly (in red)

different behaviors both normal and abnormal. This is even more complicated as anomalies are rare events. Thus, the shorter the training dataset, the less anomalies it contains, which are precisely the elements we want to be able to discriminate efficiently.

Thus, unsupervised learning is perfectly adapted to the problem of anomaly detection since it is not necessary to label large data sets. Moreover, a part of the anomalies come from new behaviors of the system. By definition, these behaviors could not be correctly classified with supervised anomaly detection methods.

2.2.3 Taxonomy

The taxonomy consists of three classes of anomaly detection methods for multivariate time series. These are: conventional approaches, machine learning-based and DNN-based methods.

Conventional approaches, which are also referred to statistical methods by some authors [36], rely on the assumption that the observed data is generated by a stochastic model and their aim is to estimate a model's parameters from the data and then use the model for prediction [37]. It is often the case that the model hypothesis is considered linear [36].

The boundary between conventional and machine learning-based approaches is not fully clear. **Machine learning-based** models produce predictions about the results of complex mechanisms by mining databases of inputs for a given problem, without necessarily having an explicit assumption about a model's hypothesis. In this setup, a method aims to learn a function that operates an input data to predict output responses [37].

Finally, **DNN-based methods** are a subclass of non-linear machine learning-based methods which use neural networks with multiple layers [8].

2.2.4 Conventional methods

The methods presented in this section offer many different approaches. They are classified into five categories. Control chart, where the objective is to monitor variations in the statistical characteristics of a process. Prediction methods, where the objective is to predict the next point. Decomposition techniques, based on the search for unusual patterns in time series using decomposition and finally similarity-search model based on the search for similar sub-sequences in the data.

2.2.4.1 Control Charts methods

Control charts are a statistical process control tool. Their goal is to track the mean and variance of a time series over time and to detect abrupt changes in its statistical behaviour. These methods are based on the use of Sequential Probability Ratio Tests (SPRT) [38]. Two methods are considered: Multivariate CUmulated SUMs control chart (MCUSUM) and Multivariate Exponential Weighted Moving Average (MEWMA).

In [39], Woodall and Ncube proposed to monitor the performance of multiple variables in a manufacturing system using a control chart called **Multivariate CUmulated SUMs control chart (MCUSUM)**. The MCUSUM uses the fact that the cumulative sum St of the recursive residuals of a statistic s compared to a normal value (like the mean) : $St = s(\mathbf{x}_{t-k+1}, \ldots, \mathbf{x}_t)$ is stable for normal data and increasing after the change. Using this property, the MCUSUM anomaly score g_t is based on the comparison of the increase of S_t with a threshold h:

$$g_t = max(0, St - \mu + g_{t-1}) \tag{2.1}$$

with $g_0 = 0$ and μ the value of St for normal data. MCUSUM iterates over g_t as long as $g_t < h$. If $g_t \geq h$, an alarm is raised and the iteration over g_t is restarted with $g_0 = 0$ in order to detect another change point. Due to this sequential restarting of the test, MCUSUM detects small but persistent structural changes.

The **Multivariate Exponential Weighted Moving Average (MEWMA)** [40] is based, as its name indicates, on the exponential smoothing of data. For a given constant $\lambda \in [0, 1]$, the successive data are smoothed using:

$$g_t = \lambda \mathbf{x}_t + (1 - \lambda) g_{t-1} \tag{2.2}$$

Unlike MCUSUM, MEWMA gives more importance to recent history values.

2.2.4.2 Forecast methods

Control Charts methods are usually based on the hypothesis that the individual data in the time-series are independent and identically distributed. This assumption is rarely satisfied in practice. A generic way to handle this issue is to built a mathematical model, incorporating the known or assumed sequential correlations of the time-series. Using this model, the value $\mathbf{x}_t$ of the data indicator at t is expressed as $\mathbf{x}_t = z_t + e_t$, where z_t is accounting for normal sequential correlations within the data and e_t (the residual) is the noise. Once a mathematical model representing time-series evolution is chosen, a usual approach is to predict at time t the expected value $\hat{\mathbf{x}}_{t+1}$. The anomaly score can then be expressed as the difference between $\hat{\mathbf{x}}_{t+1}$ and the actual value $\mathbf{x}_{t+1}$.

The most commonly used forecast method in multivariate time series is the **Vector Autoregressive (VAR)** method. VAR is a statistical model that captures the inter-dependencies between several time series. In a VAR model, variables are treated symmetrically so that each variable is explained by its own past values and by the past values of the other variables. For example, in [41], VAR is used to monitor multivariate processes or detect anomalies in aviation systems [42].

2.2.4.3 Decomposition methods

Methods in this category use basis functions for decomposing the time series. Given $\mathbf{x}_t$, a point of a multivariate time series, it can be expanded in terms of the eigenfunctions ϕ_j [43] as:

$$\mathbf{x}_t = \sum_{j=1}^{\infty} \alpha_j \phi_{jt}, \tag{2.3}$$

where the coefficients α_j are given by the projection of $\mathbf{x}_t$ on the respective eigenfunctions.

Anomaly detection is then performed by computing the projection of each new point onto the eigenvectors, and a normalized reconstruction error. The normalized error is used as the anomaly score. Three methods are discussed under this subcategory: Principal Component Analysis (PCA), Singular Spectrum Analysis (SSA) and Independent Component Analysis (ICA).

Principal Component Analysis (PCA).

The idea of PCA[44] is to group datapoints into clusters defined as an ellipsoid in $\mathbb{R}^m$ containing the normal data. The idea is that the length of the principal axes of the ellipsoid represents the direction of the variability of the data instances.

The (always positive) eigenvalues λ_i with $i = 1, \ldots, k$ corresponding to the principal component v_i characterize the variability in the dataset captured by the eigenvector v_i. A subspace $S \subset \mathbb{R}^m$ is selected using the r eigenvectors v_i corresponding the r largest eigenvalues.

In [45], the Q-statistic [46] that characterizes the PCA projection residual statistics is used to define the threshold h_α, such that the anomaly score is:

$$|\widetilde{\mathbf{x}}_t| > h_\alpha$$

then $\mathbf{x}_t$ is an anomaly with $1 - \alpha$ confidence where $\alpha \in [0, 1]$ is a manually defined parameter.

Singular Spectrum Analysis (SSA) considers a time series as a projection of the trajectory of a dynamical system in a vector space $V = \mathbb{R}^m$ where $\mathbf{x}_t$ is the position at time t of a state of the system. From dynamical systems theory, it is known that physically observable states of a dynamical system in $\mathbb{R}^m$ are lying on attractors of the dynamics (i.e. subsets of the $\mathbb{R}^m$ capturing all the long term evolution of the system). Future observable states should normally continue to be located on these attractors. As such, $\hat{\mathbf{x}}_t$ are assumed to be located on (or close to) an attractor of the dynamics. So, the distance between $\hat{\mathbf{x}}_t$ and the attractor computed at time t is evaluated and used as anomaly score [47, 48].

Independent Component Analysis (ICA) [49] assumes that the different physical processes that generate multivariate time series are statistically independent of each other [50].

ICA decomposes a multivariate time series into "independent" components by orthogonal rotation and by maximizing the statistical independence between the components assuming their non-Gaussianity. In [51], the authors use Kurtosis, a classical measure of non-Gaussianity, as an anomaly score. A negative Kurtosis indicates a uniformly distributed factor, indicating a clustered structure and a high positive Kurtosis identifies a multivariate anomaly.

2.2.4.4 Similarity-search approach

Similarity search methods [52] are designed to identify patterns in a multivariate time series. These methods compute the distance between all the sub-sequences of the time series. The minimum distance between a sub-sequence and all others is used as an anomaly score.

The **Matrix-Profile (MP)** [53, 54] is a data structure for time series analysis. It consists of three elements. The first element is the distance profile, which is a vector of Euclidean distances between a given sub-sequence W and each sub-sequence in the set A of all sub-sequences of the multivariate time series. The distance is measured using the z-normalized Euclidean distance between sub-sequences. The distance profiles are arranged into a data structure denoted the distance matrix, which corresponds to the second element in MP. The distance matrix stacks together all the distance profiles that have been computed for each reference sub-sequence.

Finally, the matrix profile is the simplification of the distance matrix by looking only at the closest neighbor for each sub-sequence. The vector obtained corresponds to the smallest values of each row in the matrix. It is defined as :

$$MP(i) = min(d(W^i, W^j)) \tag{2.4}$$

with $W^j \in A \setminus \{W^i\}$.

A low value in the matrix profile indicates that the sub-sequence has at least one relatively similar sub-sequence located somewhere in the original series. In [55], it is shown that a high value indicates that the original series must have an abnormal sub-sequence. Therefore the matrix profile can be used as an anomaly score, with a high value indicating an anomaly.

2.2.5 Machine learning-based methods

The methods presented in this section fall into three categories : Isolation, Neighbourhood-based and Domain-based methods, which have been proposed in the survey by Domingues *et al.* [35]. Isolation algorithms consider a point as an anomaly when it is easy to isolate from others. Neighbourhood-based models look at the neighbourhood of each data point to identify outliers. Domain-based methods rely on the construction of a boundary separating the nominal data from the rest of the input space.

A common characteristic of machine learning-based techniques is that they typically model the dependency between a current time point and previous ones by transforming the multivariate time series $\mathcal{T}$ into a sequence of windows $\mathcal{W} = \{W_1, \dots, W_T\}$, where W_t is a time window of length K at a given time t :

$$W_t = \{\mathbf{x}_{t-K+1}, \dots, \mathbf{x}_{t-1}, \mathbf{x}_t\}. \tag{2.5}$$

Figure 2.7 shows this process. In this setup, learning-based anomaly detection methods assign to a window $\widehat{W}_t$, $t > T$, a label y_t to indicate a detected anomaly at time

$$\text{Window t :}\quad \mathbf{x}_1, \dots, \underbrace{\mathbf{x}_{t-K+1}, \mathbf{x}_{t-K+2}, \mathbf{x}_{t-K+3}, \dots, \mathbf{x}_t}, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \dots, \mathbf{x}_N$$

$$\text{Window t+1 :}\quad \mathbf{x}_1, \dots, \mathbf{x}_{t-K+1}, \underbrace{\mathbf{x}_{t-K+2}, \mathbf{x}_{t-K+3}, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}}, \mathbf{x}_{t+2}, \dots, \mathbf{x}_N$$

$$\text{Window t+2 :}\quad \mathbf{x}_1, \dots, \mathbf{x}_{t-K+1}, \mathbf{x}_{t-K+2}, \underbrace{\mathbf{x}_{t-K+3}, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}}, \dots, \mathbf{x}_N$$

Figure 2.7: Splitting a time series into a sequence of time windows

t, *i.e.* $y_t = 1$, or not $(y_t = 0)$ based on the window's anomaly score. This notation will be used in the following.

2.2.5.1 Isolation methods

Isolation methods focus on separating outliers from the rest of the data points. These methods attempt to isolate the anomalies rather than mapping the normal points.

The **Isolation Forest (IF) algorithm** [56, 57] is based on decision trees, similarly to the Random Forest [58]. IF calculates, for each time window an anomaly score. To calculate this score, the algorithm isolates the sample recursively: it chooses a feature and a "cut-off point" at random, then evaluates whether this isolates the sample; if so, the algorithm stops, otherwise, it chooses another feature and another cut-off point at random, and so on until the data is isolated from the rest.

Recursive data partitioning can be represented as a decision tree, and the number of cuts needed to isolate a sample simply corresponds to the path taken in the tree from the root to the leaf, representing the isolated sample. The path length defines the anomaly score: a sample with a very short path, i.e. sample that is easy to isolate, is also likely to be an anomaly since it is very far from the other samples in the dataset as can be seen in Figure 2.8. As with random forests, it is possible to do this independently by using several trees, to combine their results to improve performance. In this case, the anomaly score is the average of the path lengths on the different trees.

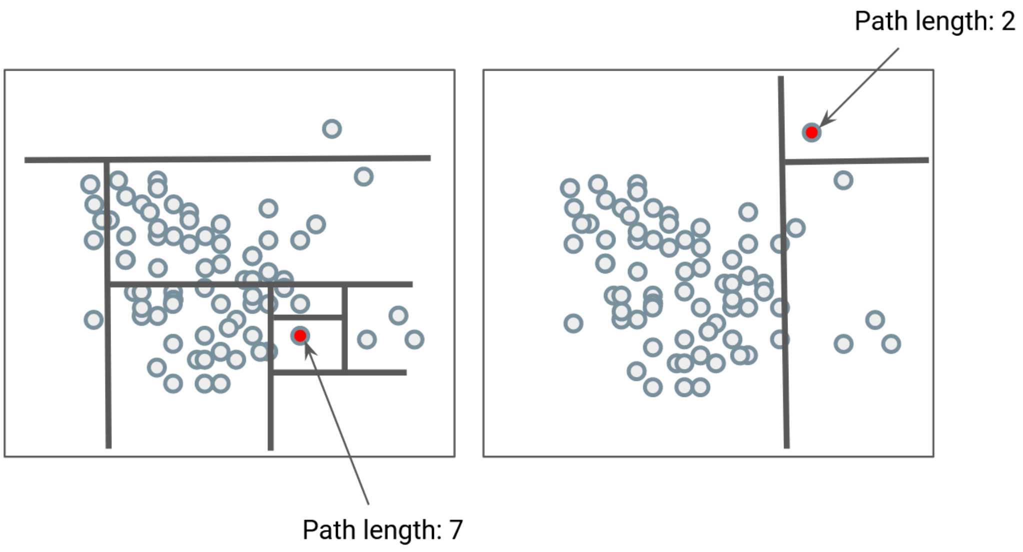

Figure 2.8: Path length to isolate a normal point (left) and an anomalous point (right)

2.2.5.2 Neighbourhood-based methods

Among neighborhood-based methods, which study the neighborhoods of every point to identify anomalies, the **Local Outlier Factor (LOF)** [59] measures the local deviation of a given data point with respect to its neighbours. Based on the K-nearest neighbors [60], the local density of an observation is evaluated by considering the closest K observations in its neighbourhood. The anomaly score is then calculated by contrasting its local density with those of its k-nearest neighbors. A high score indicates a lower density than its neighbors and therefore potentially an anomaly. It has been applied to multivariate time series [61], demonstrating its ability to detect anomalies in long-term data.

Density-based spatial clustering of applications with noise (DBSCAN) [62] is a clustering method that groups data points in high density areas (many nearby neighbors) and marks points in low-density regions (few neighbors) as anomalous. DBSCAN thus classifies the points into three categories. Core points are points containing at least $minPts$ in their ϵ distance neighborhood. The density-reachable points are the points containing at least one core point in their neighborhood. The other points are considered as anomaly. To handle multivariate time series, DBSCAN considers each time window as a point with the anomaly score being the distance from the point to the nearest cluster [63].

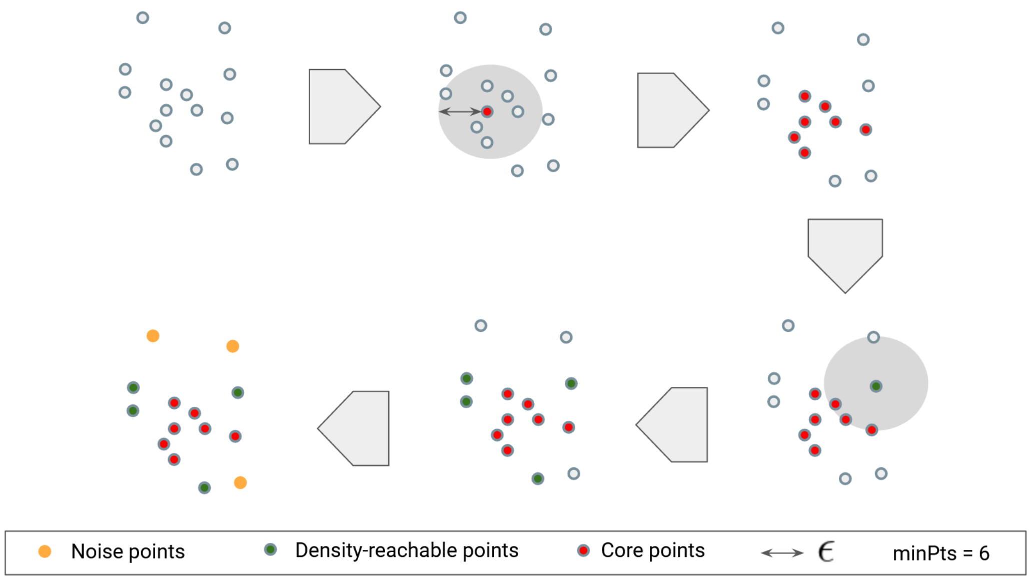

Figure 2.9: Overview of DBSCAN

2.2.5.3 Domain-based methods

Domain-based methods aim to construct a boundary between normal samples and the rest of the input space. The distance of a points to this boundary is used as the anomaly score. Among these, the **One-Class Support Vector Machine (OC-SVM)** [64] method learns the smallest hypersphere containing of all the training data points (Figure 2.10). The learned model classifies points inside the hypersphere as normal and labels those in the rest of the space as anomalous. An anomaly score can also be obtained by taking the signed distance from the hyper-sphere. The signed distance is positive for a normal value and negative for an abnormal. As other machine learning methods One-class SVM has been used for time series anomaly detection by using time windows rather than the raw series. J.Ma and S.Perkins [65] use this approach, while proposing to combine the one-class SVM output for different time windows to produce more robust detection results.

2.2.6 Deep learning-based methods

DNN-based methods are a are sub-category of machine learning-based approaches, which rely on deep neural networks. Given the explosion of DNN-based methods over the last years, they are presented as a separate category.

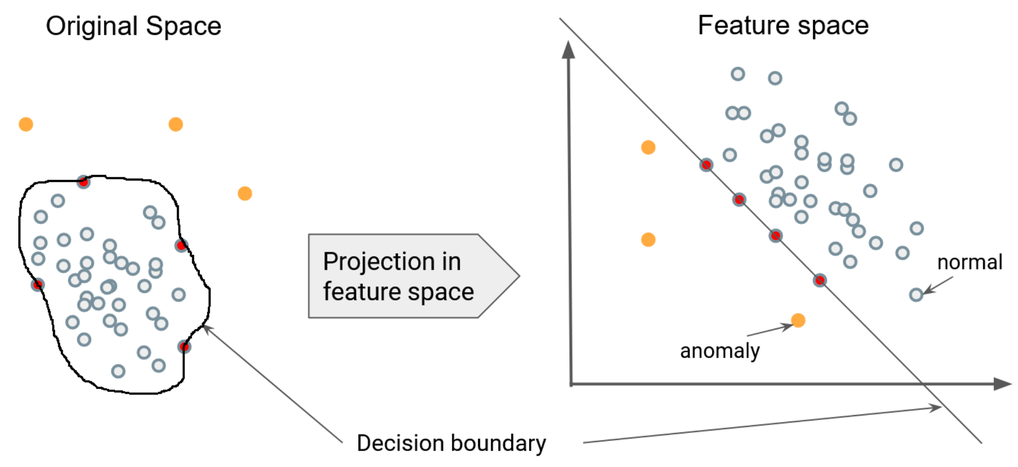

Figure 2.10: Overview of the OC-SVM

An **Auto-Encoder (AE)** [66] is an artificial neural network combining an encoder E and a decoder D. The encoder part takes the input window W and maps it into a set of latent variables Z, whereas the decoder maps the latent variables Z back into the input space as a reconstruction $\widehat{W}$. The difference between the original input vector W and the reconstruction $\widehat{W}$ is called the reconstruction error. Thus, the training objective aims to minimize this error. Auto-encoder-based anomaly detection uses the reconstruction error as the anomaly score. Time windows with a high score are considered to be anomalies [21].

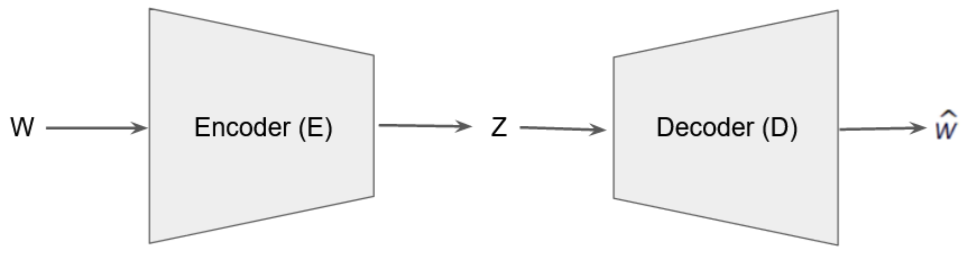

Figure 2.11: Auto-Encoder architecture

The **Generative Adversarial Networks (GANs)** [67] have the ability to know whether an input sample is normal or not. A GAN is an unsupervised artificial neural network based on a two-player minimax adversarial game between two networks, which are trained simultaneously. One network, the generator (G), aims to generate realistic data, whereas the second one acts as a discriminator (D) trying to discriminate real data from that one generated by G. The training objective of G is to maximize the probability of D making a mistake, whereas the training objective D is to minimize its classification error. Similarly to AE-based, GAN-based anomaly detection uses normal data for training. After training the discriminator is used as an anomaly detector. If the input data is different from the learned data distribution, the discriminator considers it as coming from the generator and classifies it as fake, *i.e.* as an anomaly.

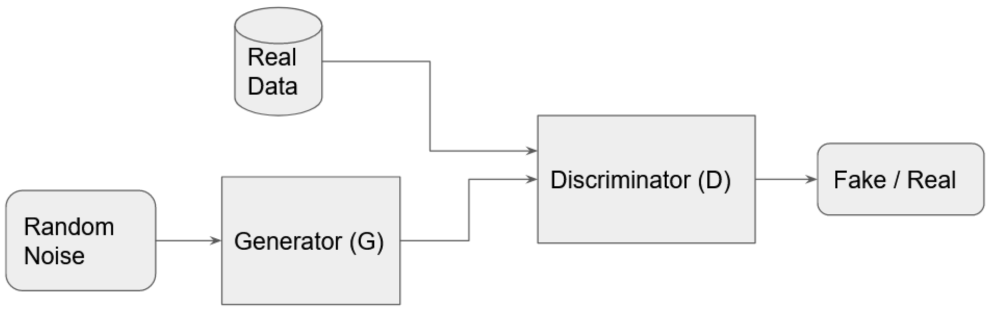

Figure 2.12: Generative Adversarial Network architecture

The **Long Short-Term Memory Variational Auto-Encoders (LSTM-VAE)** [11] combines the LSTM [68] which is a recurrent neural network architecture with a variational auto-encoder (VAE) by replacing the feed-forward network in a VAE with a long short-term memory (LSTM). The LSTM-VAE models the time dependence of time series through LSTM networks. During encoding, the LSTM-VAE projects the input data and its time dependencies into a latent space. During decoding, it uses the latent space representation to estimate the output distribution. Finally, the LSTM-VAE detects an anomaly when the log-likelihood of the current data is below a threshold. In [69], S.Lin et al. show that the LSTM-VAE is capable of identifying anomalies that span over multiple time scales.

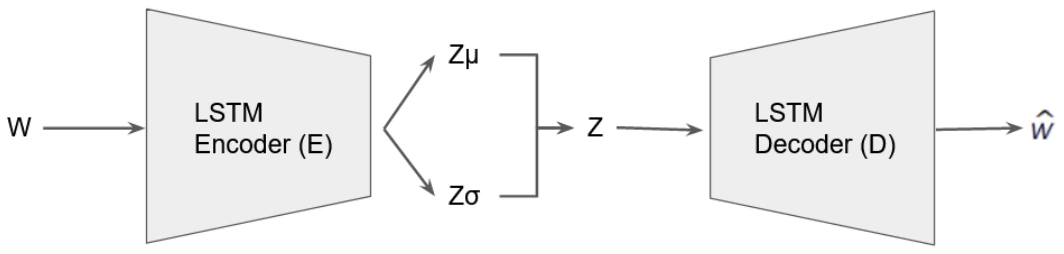

Figure 2.13: LSTM-VAE architecture

Donut [22] is based on a variational autoencoder (VAE), whose means are obtained from linear layers and the standard deviations are determined from soft-plus layers. Donut is trained using the M-ElBO [70] as objective training. Finally, the anomaly score used is the negative reconstruction probability [70]. **Bagel** [23] is an extension of Donut that can handle anomalies related to temporal information, using the conditional variational autoencoder to incorporate temporal information and the dropout layer to avoid overfitting. The calculation of the anomaly score is identical to Donut.

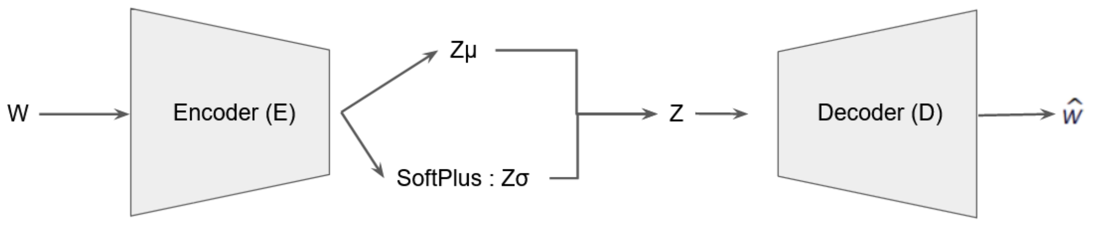

Figure 2.14: Donut architecture

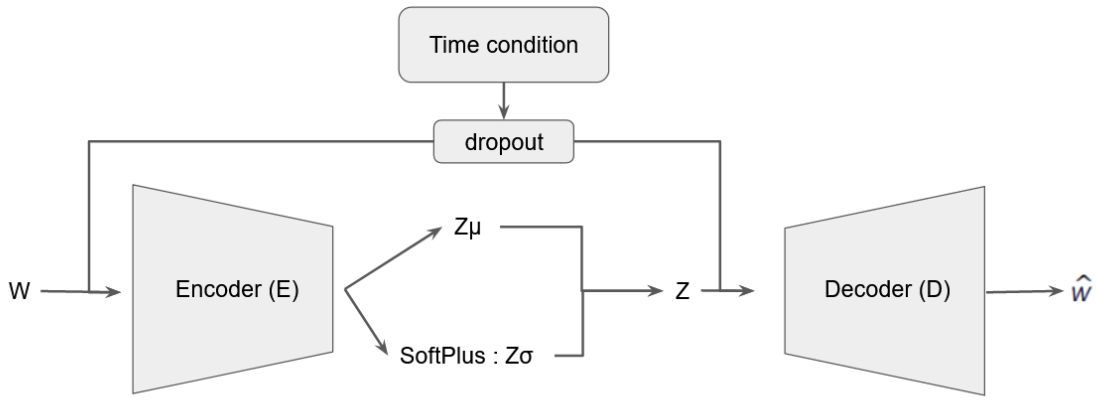

Figure 2.15: Bagel architecture

DeepAnt [26] consists of two modules. The first module is a time series predictor that creates a time series prediction using a deep convolutional neural network (CNN) [71]. The second module uses the predicted value to detect anomalies. The detection module calculates the Euclidean distance between the actual and predicted value. Thus, the Euclidean distance is used as anomaly score.

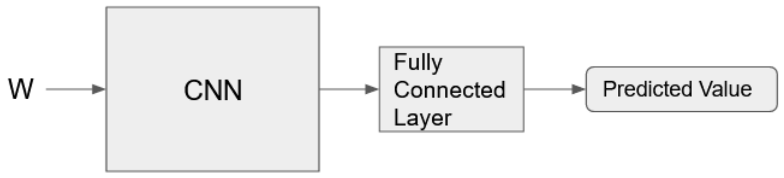

Figure 2.16: DeepAnt architecture

The **Deep Autoencoding Gaussian Mixture Model (DAGMM)** [1] jointly considers a Deep Auto-encoder and a Gaussian Mixture Model (GMM) to model the density distribution of multidimensional data. The purpose of the Deep Auto-encoder is to generate a low-dimensional representation and a reconstruction error for each input data time window. This representation is used as input of a Gaussian Mixture Model (GMM). The parameters of the Deep Auto-encoder and the mixture model are optimized simultaneously from end to end, taking advantage of a separate estimation network to facilitate the learning-based of the parameters of the mixture model. The DAGMM then uses the likelihood to observe the input samples as an anomaly score.

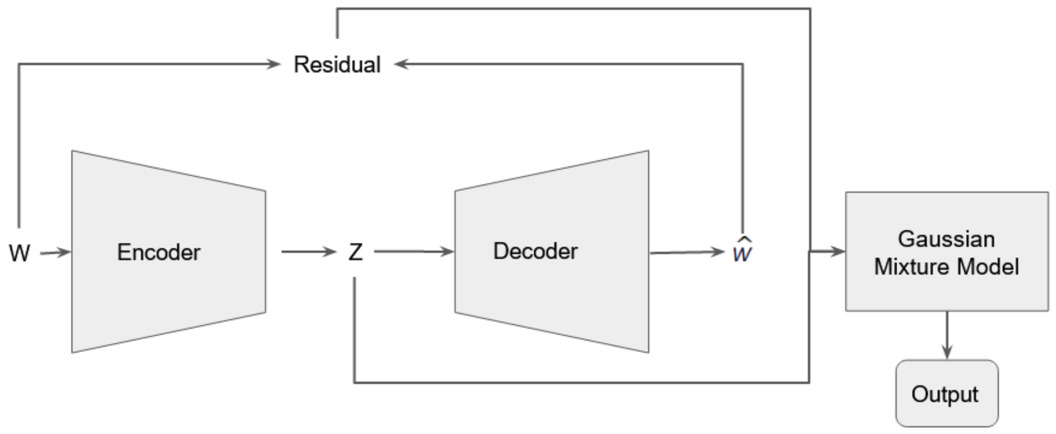

Figure 2.17: DAGMM architecture. Source [1]

The **Multivariate Anomaly Detection with Generative Adversarial Networks (MAD-GAN)** [12] is based on a Generative adversarial network (GAN) [67] architecture composed of LSTMs. MAD-GAN uses an anomaly score called DR-score to detect anomalies. This score is composed of the discrimination between real data and fake data of the discriminator and the reconstruction error of the generator. Indeed, because of the smooth transitions of the latent space, the generator produces similar samples if the entries in the latent space are identical. Thus, we can use the residuals between the test data and their transformation by the generator to identify anomalies in the test data.

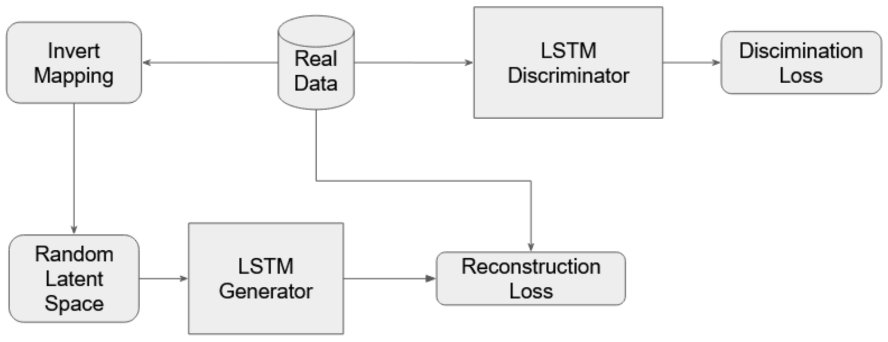

Figure 2.18: MAD-GAN architecture

The **Recurrent Conditional Generative Adversarial Networks (RCGAN)** [24] is an adaptation of AnoGAN [72] using recurrent neural networks. The method is thus based on recurrent GANs. The reconstruction error of the generator is used as an anomaly score.

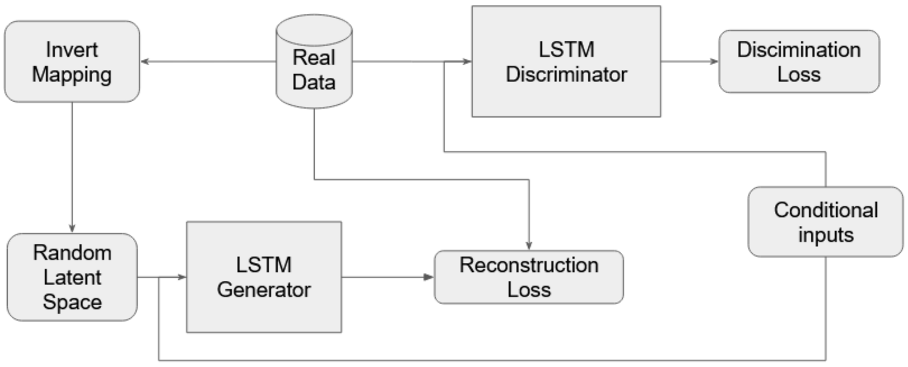

Figure 2.19: RCGAN architecture

The **Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED)** [73] consists of four steps. First, it constructs signatures matrices based on the pair-wise inner product of the time series (i.e., the correlation matrix at each time point) which is encoded via convolutional neural networks [71]. The temporal patterns are learned using attention-based convolutional LSTM networks. The signature matrices are decoded via deconvolutional neural networks. Finally, the residuals between the input and reconstructed matrix signatures are used as anomaly scores.

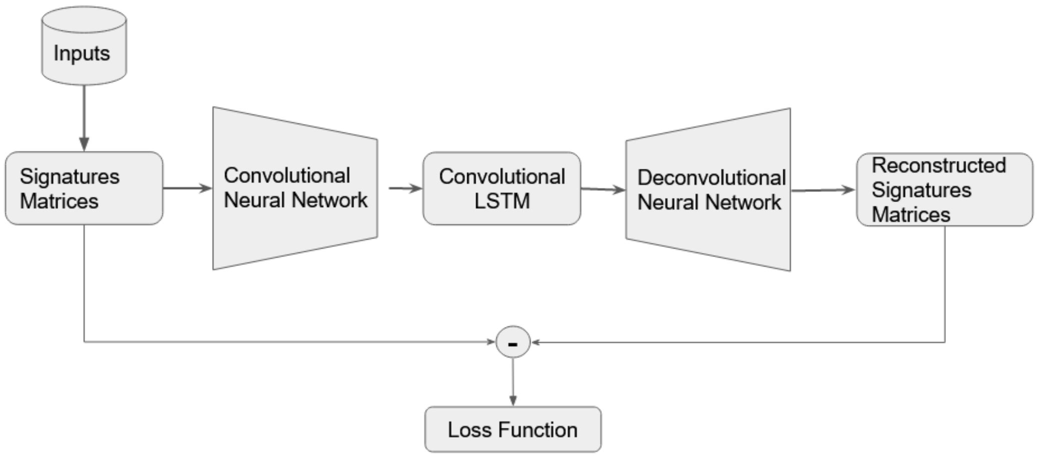
Inputs
Signatures Matrices
Convolutional Neural Network
Convolutional LSTM
Deconvolutional Neural Network
Reconstructed Signatures Matrices
-
Loss Function

Figure 2.20: MSCRED architecture

The **MTS-DCGAN** [27] is based on a Deep Convolutional Generative Adversarial Network. As with MSCRED the first step is to create signature matrices of different time window sizes. The matrices are then sent to a DCGAN [27]. The generator is composed of two fully connected layers and a CNN and the discriminator is composed of a CNN, a flattened layer and two fully connected layers that allow to capture the correlations. A forgetting mechanism in the sliding window allows the recent points to be weighted more heavily. The output probabilities of the discriminator are used as anomaly score.

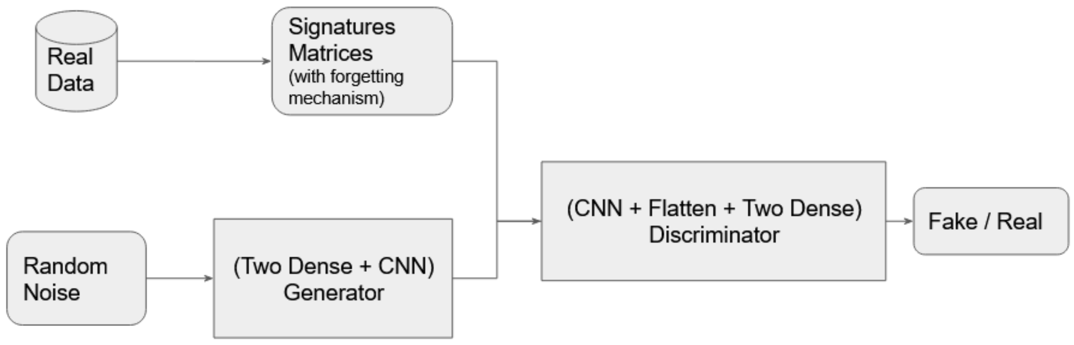
Real Data
Signatures Matrices (with forgetting mechanism)
Random Noise
(Two Dense + CNN) Generator
(CNN + Flatten + Two Dense) Discriminator
Fake / Real

Figure 2.21: MTS-DCGAN architecture

The **BeatGAN** [25] is based on a CNN Auto-Encoder and a Discriminator. The objective function during the training is composed of the usual objective of the auto-encoders which is to minimize the difference between its input and its output as well as an adversarial training of the GANs where the generator is the auto-encoder and must fool the discriminator by making the reconstructed data look real. During the detection only the auto-encoder is used. The anomaly score is the residual between the input data and the reconstructed data.

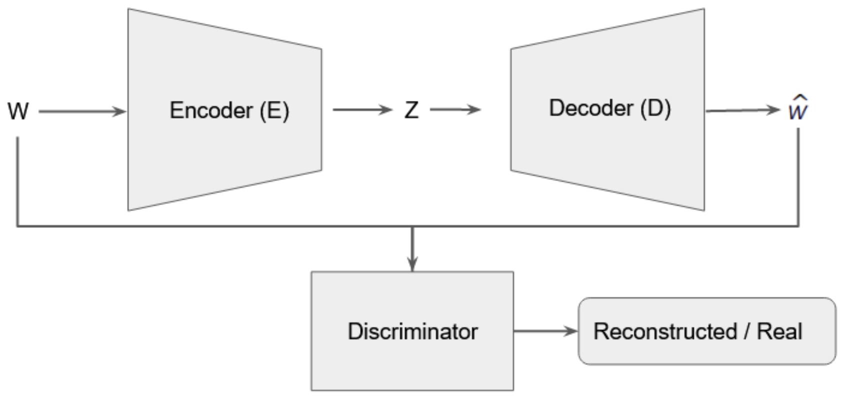

Figure 2.22: BeatGan architecture

FuseAD [28] combines ARIMA [74] and Convolutional Neural Networks. FuseAD is composed of two modules. The first module is called the forecasting pipeline. It consists in combining the value predicted by an ARIMA and a CNN. The anomaly score is produced by a second module called Anomaly detector. It corresponds to the difference between the predicted value and the real value.

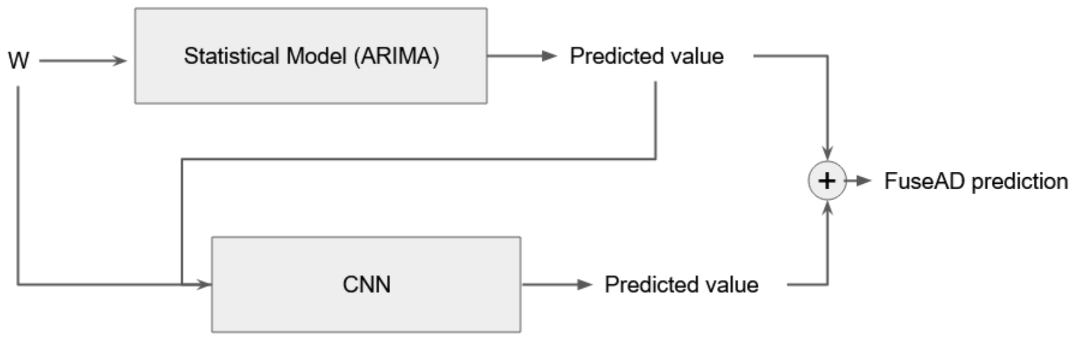

Figure 2.23: FuseAD architecture

The **Real-Time Anomaly Detection in Multivariate Time Series (RADM)** [30] is based on Hierarchical Temporal Memory (HTM) [75] and Bayesian Network (BN) [76]. It is divided into two phases. The first one is the detection of anomalies in the univariate time series composing the multivariate time series using the HTM. This first anomaly score is calculated by the HTM using the distribution of anomaly scores to calculate the anomaly likelihood. Thus, the anomaly score depends on the prediction history of the HTM of the anomaly which defines the degree of anomaly of the state the current state is anomalous based on the prediction history of the

HTM model. In the second phase, a Naive Bayesian Network takes as input the discretized values of the anomaly likelihoods of each univariate series to assign the corresponding weights to each time series according to the structure of the multivariate series. Then the junction tree method [77] is used to infer the classification of anomalies, in order to obtain the list of anomaly regions.

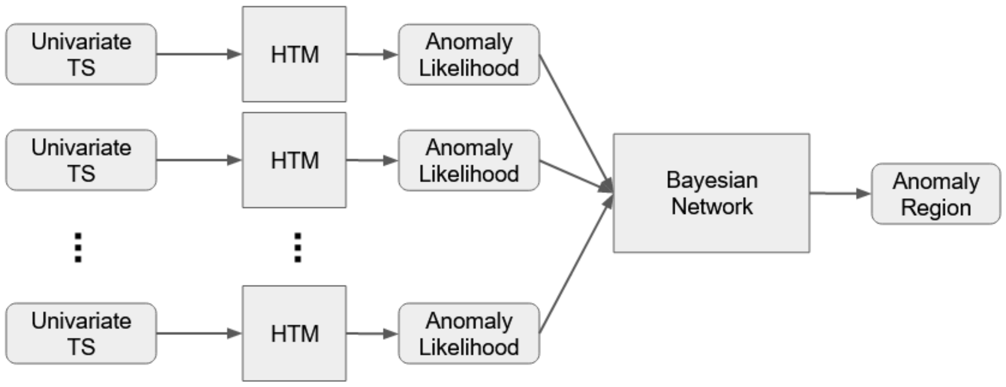

Figure 2.24: RADM architecture

The **Multivariate Time Series Anomaly Detection Using the combination of Temporal pattern and Feature pattern (MTAD-TF)** [31] can be split into two main parts. The first part is called Temporal convolution component. It is based on a multiscale 1D convolution that allows to detect temporal patterns. The second part is called Graph attention component. It allows to learn the correlation between features and is based on a graph attention network [78]. The combination of these two parts provides a prediction. The anomaly score is the squared error between the predicted value and the actual value.

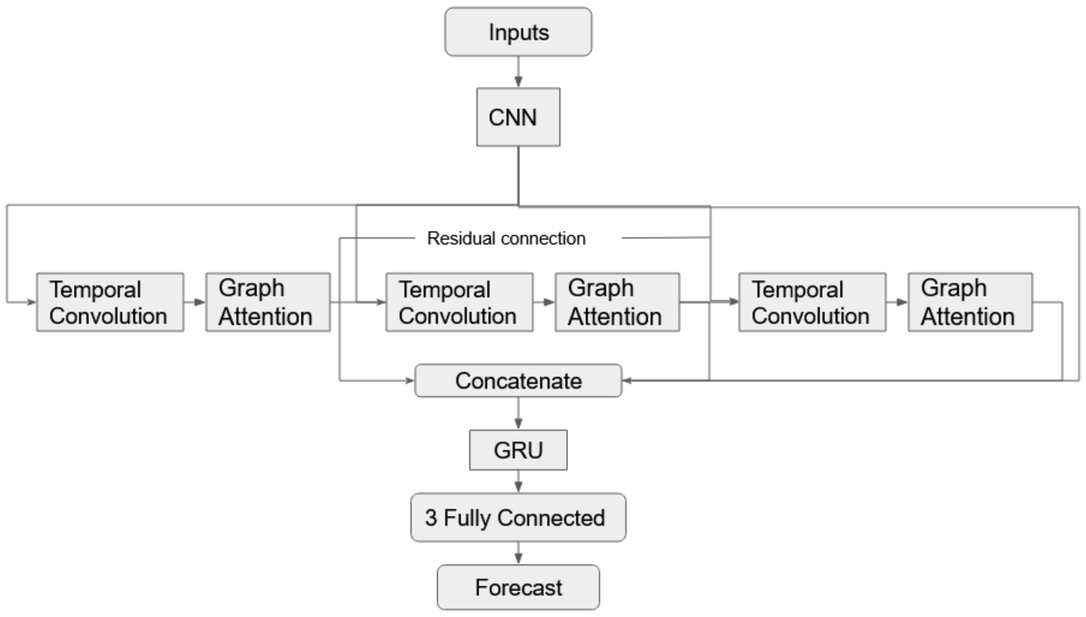

Figure 2.25: MTAD-TF architecture

The **Adversarially Learned Anomaly Detection (ALAD)** [79] ALAD is based on bi-directional GANs that derive adversarially learned features for the anomaly detection task. Unlike GANs, bi-directional GANs are architectures composed of three networks. A Generator, a Discriminator and an Encoder whose role is to project the input data to the latent space. Thus, the encoder is the inverse of the generator. ALAD then uses the reconstruction errors as an anomaly score.

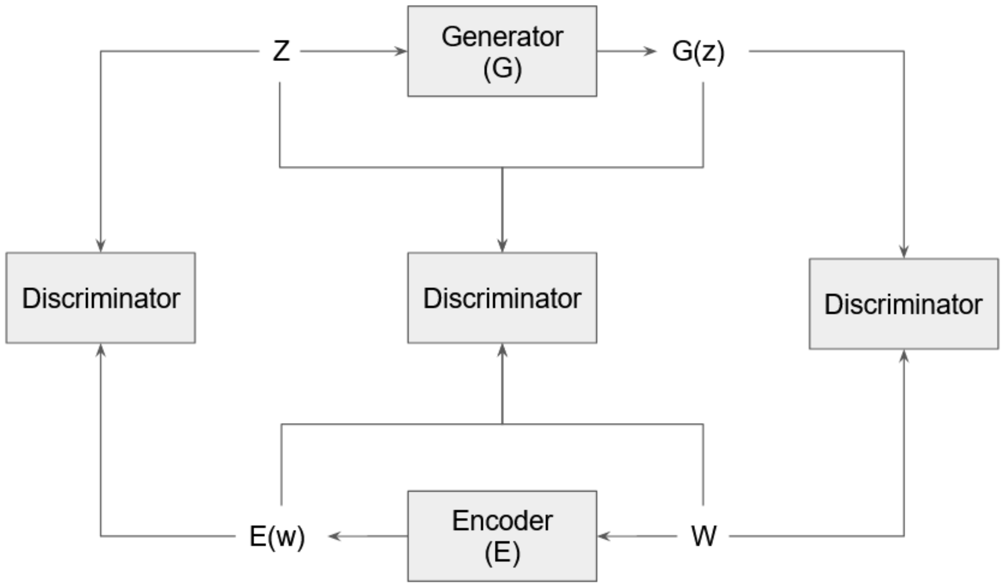

Figure 2.26: ALAD architecture

Finally, **OmniAnomaly (OA)** [13] is a stochastic recurrent neural network for multivariate time series anomaly detection that learns robust multivariate time series' representations with a stochastic variable connection and a planar normalizing flow, and uses the reconstruction probabilities to determine anomalies. OmniAnomaly uses the Gated Recurrent Unit (GRU) to model the time dependencies of multivariate time series. The method also uses a VAE to learn a mapping of the input data W to a representation in a latent space. To model time dependencies in the latent space, OmniAnomaly uses a stochastic variable connection technique. As suggested by [80], the reconstruction can be evaluated by a conditional probability. The anomaly score used is then the probability of reconstruction. A high score means that the input can be well reconstructed. If an observation follows normal time series patterns, it can be reconstructed with high confidence. On the other hand, the lower the score, the less well the observation can be reconstructed and the more likely it is to be anomalous.

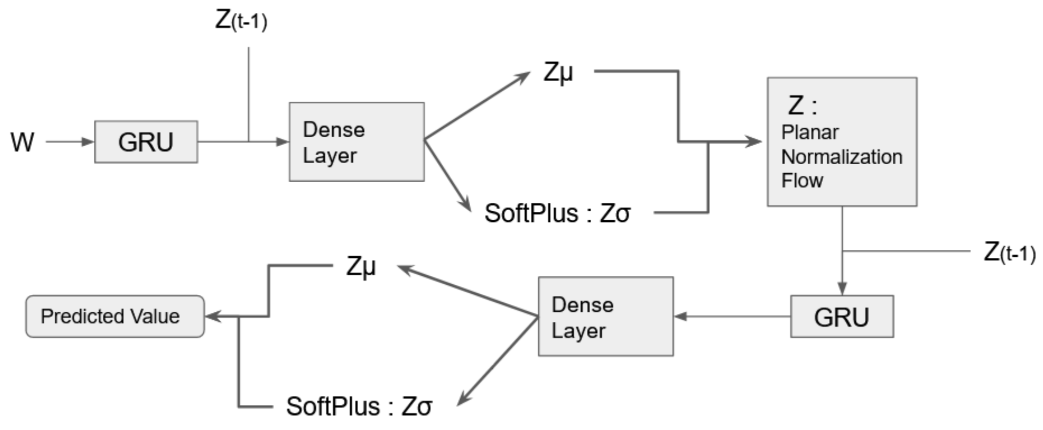

Figure 2.27: OmniAnomaly architecture

Chapter 3

Unsupervised Anomaly Detection on Multivariate Time Series

Contents

This chapter proposes a fast and stable method called UnSupervised Anomaly Detection for multivariate time series (USAD) based on adversely trained autoencoders. Its autoencoder architecture makes it capable of learning in an unsupervised way. The use of adversarial training and its architecture allows it to isolate anomalies while providing fast training. The study of the properties of the method through experiments on five public datasets demonstrates its robustness, learning speed and high performance in anomaly detection. Through a feasibility study using Orange's proprietary data, the method validates Orange's requirements.

3.1 Introduction

This chapter presents a new method called UnSupervised Anomaly Detection for multivariate time series (USAD) based on an autoencoder architecture [66] whose learning is inspired by GANs. The intuition behind USAD is that the adversarial training of its encoder-decoder architecture allows it to learn how to amplify the reconstruction error of inputs containing anomalies, while gaining stability compared to methods based on GANs architectures. Its architecture makes it fast to trained meeting Orange's expectations in terms of scalability and algorithm efficiency. The main contributions of this chapter are:

- It proposes an encoder-decoder architecture within an adversarial training framework that allows to combine the advantages of autoencoders and adversarial training, while compensating for the limitations of each technique.
- It performs an empirical study on publicly available datasets to analyze robustness, training speed and performance of the proposed method.
- It performs a feasibility study with Orange's proprietary data to analyze if the proposed method meets the company's requirements on scalability, stability, robustness, training speed and high performance.

The rest of this chapter is organized as follows. Section 3.2 presents the limitations of Auto-Encoders (AE) and Generative Adversarial Networks (GANs). Section 3.3 discusses the details of the method, describe the experiments and demonstrate the state-of-the-art performance of the method.

3.2 Auto-Encoders and Generative Adversarial Networks limitations

Autoencoder-based anomaly detection uses the reconstruction error as the anomaly score. Points with a high score are considered to be anomalies. Only samples from normal data are used at training. At inference, the AE will reconstruct normal data very well, while failing to do so with anomaly data which the AE has not encountered. If the anomaly is too small, *i.e.* it is relatively close to normal data, the reconstruction error will be small and thus the anomaly will not be detected.

This occurs because the AE aims to reconstruct input data as well (as close to normality) as possible. To overcome this problem, the AE should be able to identify if the input data contains no anomaly before doing a good reconstruction.

The GAN is trained in an adversarial way between the Generator and the Discriminator. GAN-based anomaly detection uses the output of the Discriminator as the anomaly score during inference. GAN training is not always easy, due to problems such as mode collapse and non-convergence [81], often attributed to the imbalance between the generator and the discriminator. Indeed, if an imbalance is created between the generator and the discriminator, the network is no longer able to learn correctly since one of the two is overwhelmed by the performance of the other. Techniques exist to stabilize the training and ensure a good convergence, have been proposed as the WGAN algorithm (Wasserstein GAN [82]), but it remains to be improved, especially because of the added complexity that these methods involve.

3.3 UnSupervised Anomaly Detection (USAD)

The UnSupervised Anomaly Detection (USAD) method is formulated as an AE architecture within a two-phase adversarial training framework. On one hand, this allows to overcome the intrinsic limitations of AEs by training a model capable of identifying when the input data does not contain an anomaly and thus perform a good reconstruction. On the other hand, the AE architecture allows to gain stability during adversarial training, therefore addressing the problem of collapse and non-convergence mode encountered in GANs.

3.3.1 Method

USAD is composed of three elements: an encoder network E and two decoder networks D_1 and D_2. As depicted in Figure 3.1 and 3.2, the three elements are connected into an architecture composed of two autoencoders AE_1 and AE_2 sharing the same encoder network:

$$AE_1(W) = D_1(E(W)), \quad AE_2(W) = D_2(E(W)) \tag{3.1}$$

The architecture from Eq. 3.1 is trained in two phases. First, the two AEs are trained to learn to reconstruct the normal input windows W. Secondly, the two AEs are trained in an adversarial way, where AE_1 will seek to fool AE_2 and AE_2 aims to learn when the data is real (coming directly from W) or reconstructed (coming from AE_1) . Further details are provided in the following.

Phase 1: Autoencoder training. At a first stage, the objective is to train each AE to reproduce the input. Input data W is compressed by encoder E to the latent space Z and then reconstructed by each decoder. The training objectives are :

$$\begin{aligned} \mathscr{L}_{AE_1} &= \|W - AE_1(W)\|_2 \\ \mathscr{L}_{AE_2} &= \|W - AE_2(W)\|_2 \end{aligned} \tag{3.2}$$

Phase 2: Adversarial training. In the second phase, the objective is to train AE_2 to distinguish the real data from the data coming from AE_1, and to train AE_1 to fool AE_2. Data coming from AE_1 is compressed again by E to Z and then reconstructed by AE_2. Using an adversarial training configuration, the objective of AE_1 is to minimize the difference between W and the output of AE_2. The objective of AE_2 is to maximize this difference. AE_1 trains on whether or not it succeeds in fooling AE_2, and AE_2 distinguishes the candidates reconstructed by AE_1 from the real data. The training objective is :

$$\min_{AE_1} \max_{AE_2} \|W - AE_2(AE_1(W))\|_2 \tag{3.3}$$

which account to the following losses

$$\begin{aligned} \mathscr{L}_{AE_1} &= +\, \|W - AE_2(AE_1(W))\|_2 \\ \mathscr{L}_{AE_2} &= -\, \|W - AE_2(AE_1(W))\|_2 \end{aligned} \tag{3.4}$$

Two-phase training.

In this architecture, autoencoders have a dual purpose. AE_1 minimizes the reconstruction error of W (phase 1) and minimizes the difference between W an the reconstructed output of AE_2 (phase 2). As AE_1, AE_2 minimizes the reconstruction error of W (phase 1) but, it then maximizes the reconstruction error of the input data reconstructed by AE_1 (phase 2). The dual purpose training objective of each AE is expressed as the combination of Equations 3.2, 3.4 in an evolutionary scheme, where the proportion of each part evolves with time:

$$\mathscr{L}_{AE_1} = \frac{1}{n} \|W - AE_1(W)\|_2 + \left(1 - \frac{1}{n}\right) \|W - AE_2(AE_1(W))\|_2 \tag{3.5}$$

$$\mathscr{L}_{AE_2} = \frac{1}{n} \|W - AE_2(W)\|_2 - \left(1 - \frac{1}{n}\right) \|W - AE_2(AE_1(W))\|_2 \tag{3.6}$$

and n denotes a training epoch. The two-phase training process is summarized in Algorithm 1.

It is important to remark that AE_2 does not act as a discriminator in the strict sense of GANs, because if its input is the original data, it is the loss from Eq 3.2 that intervenes. When its input is a reconstruction, the objective from Eq. 3.3-3.4 intervenes instead.

Algorithm 1 USAD training algorithm

```
Input: Normal windows Dataset $\mathcal{W} = \{W_1, ..., W_T\}$ , Number epochs $N$
Output: Trained $AE_1$, $AE_2$
  $E, D_1, D_2 \leftarrow$ initialize weights
  $n \leftarrow 1$
  repeat
    for $t = 1$ to $T$ do
      $Z_t \leftarrow E(W_t)$
      $W_t^{1'} \leftarrow D_1(Z_t)$
      $W_t^{2'} \leftarrow D_2(Z_t)$
      $W_t^{2''} \leftarrow D_2(E(W_t^{1'}))$
      $\mathscr{L}_{AE_1} \leftarrow \frac{1}{n} \|W_t - W_t^{1'}\|_2 + \left(1 - \frac{1}{n}\right) \|W_t - W_t^{2''}\|_2$
      $\mathscr{L}_{AE_2} \leftarrow \frac{1}{n} \|W_t - W_t^{2'}\|_2 - \left(1 - \frac{1}{n}\right) \|W_t - W_t^{2''}\|_2$
      $E, D_1, D_2 \leftarrow$ update weights using $\mathscr{L}_{AE_1}$ and $\mathscr{L}_{AE_2}$
    end for
    $n \leftarrow n + 1$
  until $n = N$
```

Inference. During the detection phase (Algorithm 2), the anomaly score is defined as:

$$\mathscr{A}(\widehat{W}) = \alpha\|\widehat{W} - AE_1(\widehat{W})\|_2 + \beta\|\widehat{W} - AE_2(AE_1(\widehat{W}))\|_2 \tag{3.7}$$

where $\alpha+\beta = 1$ and are used to parameterize the trade-off between false positives and true positives. If α is greater than β, the number of true positives and the number of false positives is reduced. Conversely, if a α less than β is taken, the number of true positives and also the number of false positives is increased. Thus, $\alpha < \beta$ denotes a scenario with high detection sensitivity and $\alpha > \beta$ a low detection sensitivity one. This parametrization scheme is of great industrial interest. It allows, using a single trained model, to obtain during the inference a set of different sensitivity anomaly scores. This is further illustrated in Section 3.3.4.2.

Algorithm 2 USAD Detection algorithm

```
**Input:** Test windows Dataset $\widehat{\mathbb{W}} : (\widehat{W}_1, ..., \widehat{W}_{T^*})$ , threshold $\lambda$ , parameters $\alpha$ and $\beta$
**Output:** Labels $\mathbf{y} : \{y_1, ..., y_{T^*}\}$
  for $t = 1$ to $T^*$ do
    $\widehat{W}_t^{1'} \leftarrow D_1(E(\widehat{W}_t))$
    $\widehat{W}_t^{2''} \leftarrow D_2(E(\widehat{W}_t^{1'}))$
    $\mathscr{A} \leftarrow \alpha \left\| \widehat{W}_t - \widehat{W}_t^{1'} \right\|_2 + \beta \left\| \widehat{W}_t - \widehat{W}_t^{2''} \right\|_2$
    if $\mathscr{A} \geq \lambda$ then
      $y_t \leftarrow 1$
    else
      $y_t \leftarrow 0$
    end if
  end for
```

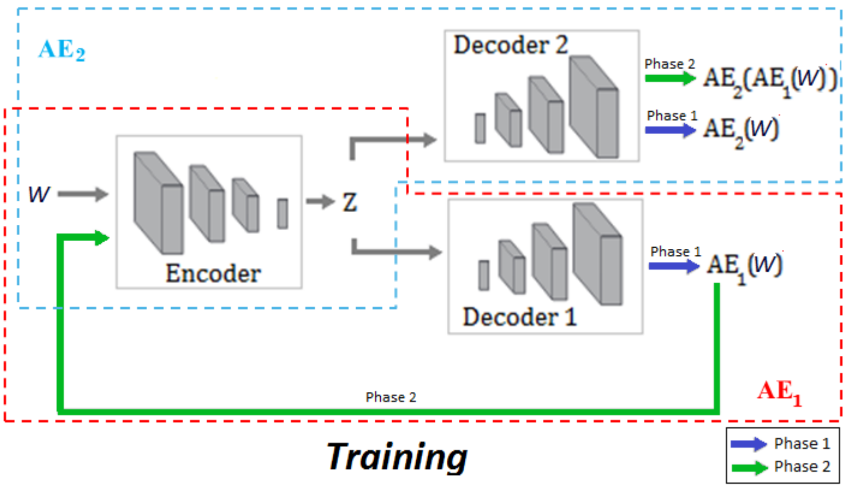

Figure 3.1: Proposed architecture illustrating the information flow at training.

3.3.2 Implementation

This method of anomaly detection is divided into three stages. There is a first data pre-processing stage common to training and detection where data is normalized

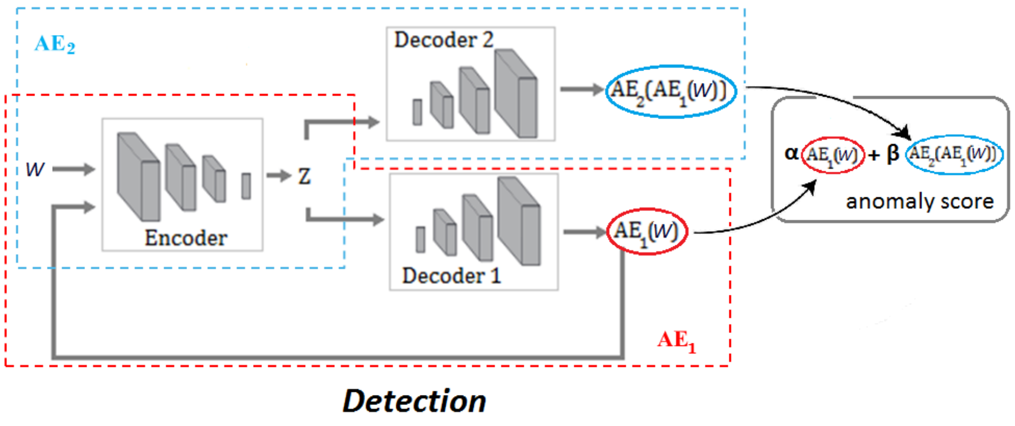

Figure 3.2: Proposed architecture illustrating the information flow at detection stage.

and split into time windows of length K. The second stage is used for training the method. The training is offline and aims to capture the normal behaviors of predefined portions (a few weeks/months) of multivariate time series and to produce an anomaly score for each time window. This offline training procedure can be performed automatically at regular time intervals, taking care to select a training period that does not include too many periods considered abnormal. The last stage is anomaly detection. It is performed online using the model trained at the second stage. As a new time window arrives, the model is used to obtain an anomaly score. If the anomaly score of a window is higher than a defined anomaly threshold, the new time window is declared as abnormal.

3.3.3 Experimental setup

This section describes the datasets and the performance metrics used in the experiments and the feasibility study.

3.3.3.1 Datasets

Five publicly available datasets were used in the experiments. Table 3.1 summarizes the datasets characteristics and they are briefly described in the following.

Secure Water Treatment (SWaT) Dataset The SWaT dataset [1] is a scaled down version of a real-world industrial water treatment plant producing filtered

[1]`https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/#swat`

Table 3.1: Benchmarked Datasets. (%) is the percentage of anomalous data points in the dataset.

Dataset	Train	Test	Dimensions	Anomalies (%)
SWaT	496800	449919	51	11.98
WADI	1048571	172801	123	5.99
SMD	708405	708420	28*38	4.16
SMAP	135183	427617	55*25	13.13
MSL	58317	73729	27*55	10.72
Orange	2781000	5081000	33	33.72

water [83]. The collected dataset [84] consists of 11 days of continuous operation: 7 days collected under normal operations and 4 days collected with attack scenarios.

Water Distribution (WADI) Dataset This dataset [1] is collected from the WADI testbed, an extension of the SWaT tesbed [84]. It consists of 16 days of continuous operation, of which 14 days were collected under normal operation and 2 days with attack scenarios.

Server Machine Dataset SMD is a new 5-week-long dataset from a large Internet company collected and made publicly available [2] [13]. It contains data from 28 server machines each one monitored by $m = 33$ metrics. SMD is divided into two subsets of equal size: the first half is the training set and the second half is the testing set.

Soil Moisture Active Passive (SMAP) satellite and Mars Science Laboratory (MSL) rover Datasets SMAP and MSL are two real-world public datasets, expert-labeled datasets from NASA [85]. They contain respectively the data of 55/27 entities each monitored by $m = 25/55$ metrics.

[1] `https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/#wadi`
[2] `https://github.com/smallcowbaby/OmniAnomaly`

3.3.3.2 Feasibility study: Orange's dataset

The collected data come from technical and business indicators from Orange's advertising network in its website. The data represent a total of $m = 33$ continuous variables including 27 technical and 6 business measurements. The dataset is divided into two subsets: a train set corresponding to about 32 days and a test set corresponding to about 60 days of activity. 60 days of testing were selected corresponding to a critical period for Orange. To obtain this training set, the previous consecutive days without any major incident for the company were selected allowing to obtain a training set of 32 mainly normal days. Anomalies in the test set were labeled by domain experts based on incident reports. Its main characteristics are reported in Table 3.1.

3.3.3.3 Evaluation Metrics

Precision (P), Recall (R), and F1 score (F1) were used to evaluate anomaly detection performance:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

with TP the True Positives, FP the False Positives, and FN the False negatives.

A window is labeled as an anomaly as soon as one of the points it contains is detected as anomalous.

In [13], the authors compute the F1 score using the average precision and average recall. For the sake of completeness, this measure is also reported when comparing the method to their benchmark. This measure is denoted the F1* score:

$$F1^{*} = 2 \cdot \frac{\bar{P} \cdot \bar{R}}{\bar{P} + \bar{R}}$$

where $\bar{P}$, $\bar{R}$ denote the average precision and recall, respectively.

Performance is assessed by comparing the results of each evaluated method with the annotated ground truth. To allow a direct comparison with the benchmark proposed by [13] their approach is used. Anomalous observations usually occur in the form of contiguous anomaly segments. In this approach, if at least one observation of an anomalous segment is correctly detected, all the other observation of the segment are also considered as correctly detected, even if they were not. The observations outside the ground truth anomaly segment are treated as usual. This approach is denoted *point-adjust*. The performance without *point-adjust* is also evaluated on the two datasets (SWaT and WADI) not belonging to the benchmark [13].

Table 3.2: Performance comparison.Precision (P), recall (R) and F1 score with and without point-adjust (Without) in SWaT datasets.

Methods	SWaT					
	Without			With		
	P	R	F1	P	R	F1
AE	0.9903	0.6295	0.7697	0.9913	0.7040	0.8233
IF	0.9512	0.5884	0.7271	0.9620	0.7315	0.8311
LSTM-VAE	0.9897	0.6377	0.7756	0.7123	0.9258	0.8051
DAGMM	0.4695	0.6659	0.5507	0.8292	0.7674	0.7971
OmniAnomaly	0.9825	0.6497	0.7822	0.7223	0.9832	0.8328
USAD	0.9851	0.6618	**0.7917**	0.9870	0.7402	**0.8460**

3.3.4 Experiments and Results

The key properties of USAD are studied by assessing its performance and comparing it to other state of the art methods (3.3.4.1), analyzing how different parameters affect the performance of the method (3.3.4.2), estimating its computational performance (3.3.4.3) and through an ablation study where, at each time, one is suppress of the training phases (3.3.4.4). Finally, in Section 3.3.4.5 a feasibility study using Orange's internal data is reported to demonstrate that USAD meets the requirements needed to be deployed in production.

3.3.4.1 Overall performance

To demonstrate the overall performance of USAD is compared with five unsupervised methods for the detection of multivariate time series anomalies. These are: Isolation Forests (IF) [56], autoencoders (AE), LSTM-VAE [11], DAGMM [1], OmniAnomaly [13]. As not all of the anomaly detection methods used for comparison provide a mechanism to select anomaly thresholds,the possible anomaly thresholds for each model are tested and the results linked to the highest F1 score is reported. Table 3.2, 3.3 and 3.4 detail the obtained performance results for all methods on the public datasets. On Table 3.2 and Table 3.3, the results obtained with SWaT and WADI datasets are presented, whereas the Table 3.4 reports obtained results from the benchmark proposed by [13], using three remaining datasets. USAD outperforms all methods on SWaT, MSL, SMAP and WADI without *point-adjust* datasets, and its

Table 3.3: Performance comparison. Precision (P), recall (R) and F1 score with and without point-adjust (Without) in WADI datasets.

Methods	WADI					
	Without			With		
	P	R	F1	P	R	F1
AE	0.9947	0.1310	0.2315	0.3970	0.3220	0.3556
IF	0.2992	0.1583	0.2071	0.6241	0.6155	**0.6198**
LSTM-VAE	0.9947	0.1282	0.2271	0.4632	0.3220	0.3799
DAGMM	0.0651	0.9131	0.1216	0.2228	0.1976	0.2094
OmniAnomaly	0.9947	0.1298	0.2296	0.2652	0.9799	0.4174
USAD	0.9947	0.1318	**0.2328**	0.6451	0.3220	0.4296

F1 is the second best on the SMD dataset. On average over all datasets (Table 3.5) is the best performing method exceeding by 0.096 the current state-of-the-art [13].

Overall, IF and DAGMM present the lowest performance. These are two unsupervised anomaly detection methods that do not exploit temporal information between observations. For time series, temporal information is important and necessary because observations are dependent and historical data are useful for reconstructing current observations. In USAD, for both training and detection, the input is a sequence of observations that contains the temporal relationship to retain this information.

Despite the relative poor results in most datasets, IF achieves the highest F1 score with *point-adjust* on WADI. This is explained by the natures of the *point-adjust* method and the WADI dataset. IF considers each observation/time-point independently and assigns a label to a single time-point and not to a window. WADI's anomalies lasting in time, the point-adjust validates the entirety of an anomaly as being well detected. Thus IF is little impacted by its bad predictions (FPs) affecting only one observation at a time, compared to the advantage obtained with the *point-adjust* which validates whole segments of good prediction despite having potentially missed several abnormalities.

Differently, AE, LSTM-VAE, use sequential observations as input allowing the two methods to retain temporal information. These methods perform the best possible reconstruction regardless of the existence of an anomaly in the input window. This

Table 3.4: Performance comparison in SMAP, MSL and SMD datasets with point-adjust. P, R F1, and F1* are reported.

Methods	SMAP			
	P	R	F1	F1*
AE	0.7216	0.9795	0.7776	0.8310
IF	0.4423	0.5105	0.4671	0.4739
LSTM-VAE	0.7164	0.9875	0.7555	0.8304
DAGMM	0.6334	0.9984	0.7124	0.7751
OmniAnomaly	0.7585	0.9756	0.8054	0.8535
USAD	0.7697	0.9831	**0.8186**	**0.8634**

Methods	MSL			
	P	R	F1	F1*
AE	0.8535	0.9748	0.8792	0.9101
IF	0.5681	0.6740	0.5984	0.6166
LSTM-VAE	0.8599	0.9756	0.8537	0.9141
DAGMM	0.7562	0.9803	0.8112	0.8537
OmniAnomaly	0.9140	0.8891	0.8952	0.9014
USAD	0.8810	0.9786	**0.9109**	**0.9272**

Methods	SMD			
	P	R	F1	F1*
AE	0.8825	0.8037	0.8280	0.8413
IF	0.5938	0.8532	0.5866	0.7003
LSTM-VAE	0.8698	0.7879	0.8083	0.8268
DAGMM	0.6730	0.8450	0.7231	0.7493
OmniAnomaly	0.9809	0.9438	**0.9441**	**0.9620**
USAD	0.9314	0.9617	0.9382	0.9463

Table 3.5: Average performance (± standard deviation) over all datasets using point-adjust.

	P	R	F1	F1*
AE	0.77(0.21)	0.76(0.24)	0.73(0.19)	0.86 (0.04)
IF	0.64(0.17)	0.68(0.11)	0.62(0.12)	0.60 (0.09)
LSTM-VAE	0.72(0.15)	0.80(0.25)	0.75 (0.18)	0.86 (0.04)
DAGMM	0.62(0.21)	0.76(0.29)	0.65(0.22)	0.79 (0.04)
OA	0.73(0.25)	**0.95(0.04)**	0.78(0.19)	0.91(0.04)
USAD	**0.84(0.12)**	0.80(0.25)	**0.79(0.18)**	**0.91(0.04)**

does not allow them to detect anomalies close to the normal data. USAD compensates for this drawback of AE-based methods through its adversarial training. A similar situation occurs with OmniAnomaly, as it does not have a mechanism that allows to amplify "mild" anomalies.

3.3.4.2 Effect of parameters

In this section, the effects that different parameters and factors that can have impact on the performance of USAD is studied. All experiments were done using the SWaT dataset.

The first factor studied is how USAD responds to different down-sampling rates of the training data. Down-sampling speeds up learning by reducing the size of the data and also has a denoising effect. However, it can have a negative effect if too much information is lost. Figure 3.3(A) summarizes the obtained results using 5 different rates $[1, 5, 10, 20, 50]$. Results show that USAD's performance is relatively insensitive to down-sampling, with a relatively constant performance across sampling rates. This indicates that the choice of the down-sampling rate is not critical to the method. For this experiments, a rate of 5 is selected. This is the best trade-off between denoising the training data and limiting the loss of information. Moreover, it allows to reduce by 5 the training time needed for USAD.

The second factor investigated is how USAD responds to different window sizes in the data. The window size has an impact on the type of abnormal behaviors that can be detected a direct impact on the speed of anomaly detection since the speed of detection is defined by the duration of a window. Figure 3.3(B) presents the obtained results for five different window sizes $K \in [5, 10, 20, 50, 100]$. The best result was achieved for window size $K = 10$. USAD can detect behavior changes faster when the window is smaller since each observation has a greater impact on the anomaly score. A window that is too large will have to wait for more observations to detect an anomaly. However, a larger window will detect longer anomalies. If an anomaly is however too short, it may be hidden in the number of points that a too-large window has. For Orange, a small window is better since it allows both faster training and faster detection.

The latent variables Z sit in a $m-$dimensional space, which is assumed to be smaller than one of the original data. The role of m in the performance of USAD is studied. Figure 3.3(C) presents the results for $m \in [5, 10, 20, 40, 100]$. Results show that a very small dimension for Z causes a large loss of information at the encoding stage that the decoder is not then able to recover, thus leading to a poor performance. On the other extreme, using a large value for m results in memorization of the training data causing and a drop in performance. Instead, mid-range values of m do not seem to have a strong effect in the performance, showing both relatively high and stable F1 scores.

USAD is trained under the assumption that the training set is formed using only normal samples. But in practice the training set do not only consist of normal data. Therefore, I investigate to which level the performance of the method is affected when this assumption is broken by injecting noise in the training dataset. A Gaussian noise ($\mu = 0$, $\sigma = 0.3$)is injected in a random selection of time-points representing a percentage of the training dataset size.This percentage varies from 1% to 30%. The noise is injected after down-sampling (rate= 5) to avoid noise attenuation by the down-sampling.

Figure 3.3(D) shows the performance of the method, in terms of P, R and F1, as the level of noise increases. USAD demonstrates its robustness with a relatively constant, high performance for noise levels of up to 5%. When the training set noise is of 10% a slight drop in the performance starts to be observed. However, the overall performance, measured by the F1 score, remains good. Interestingly, this performance drop is caused by a lower precision. As the recall remains relatively constant, this implies that with higher noise in the training set the method begins to be more prone to detect false positives. This behavior suggests that as the noise starts to increase, USAD is no longer able to properly learn the most complex behaviors existing within the training set. As a result, the number of false positives

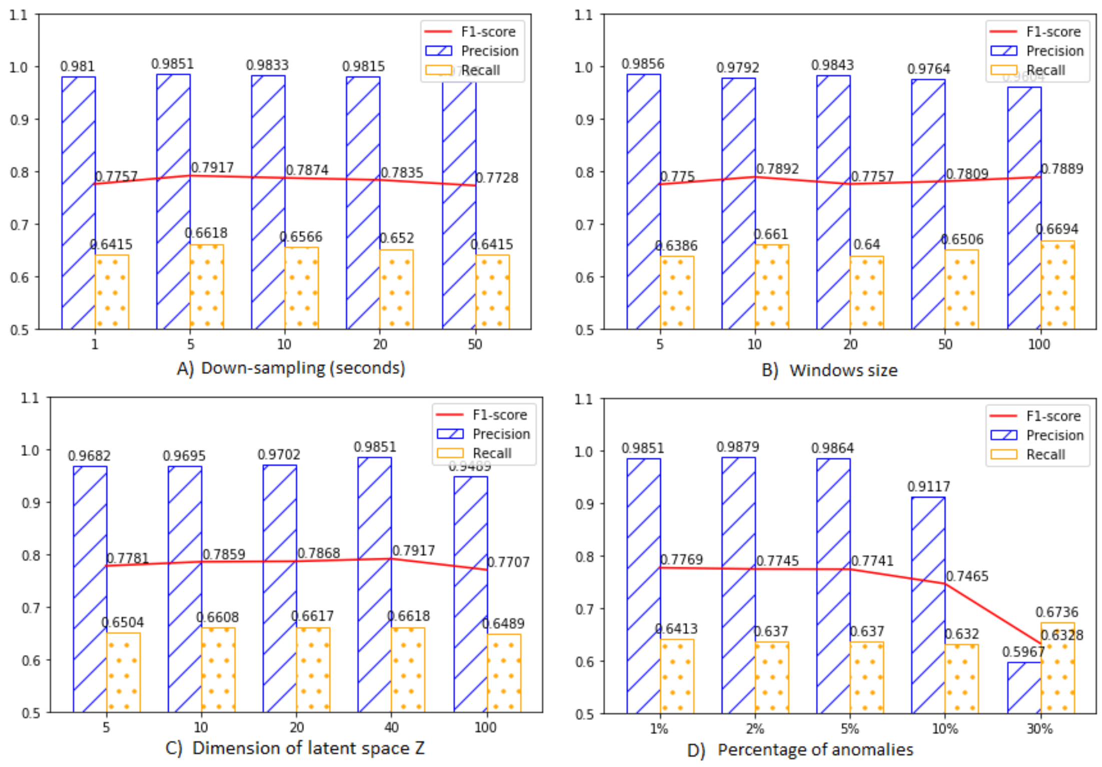

Figure 3.3: Effect of parameters. Precision, Recall and F1-score as a function of A) the training set's down-sampling rate, B) the window size K, C) the dimension of the latent space Z and D) the percentage of anomalies in the training set

increases in the test set, since USAD detects complex normal behaviors as anomalies. Finally, a significant drop in performance can be observed for high noise levels (30%). However, such a high anomaly rate during training in a production environment is not realistic. This means that for a given period of time, 30% of the samples are unnoticed anomalies. As there are so many anomalies in production, it is not realistic that such a large number of incidents are missed by Orange's incident supervision. Thus, it is unlikely that USAD will be confronted with such a high rate of anomalies during its training in a production environment at Orange.

Finally, the role of the sensitivity threshold (equation 3.7) is studied. A large α corresponds to giving more importance to the reconstruction of the AE_1 autoencoder in the anomaly score, while a large β corresponds to giving more importance to the reconstruction of the AE_2 autoencoder (see Figure 3.1). The possibility to tune the detection sensitivity without having to re-train the model is of great importance for Orange.

Table 3.6: Anomaly detection results with various sensitivity thresholds for SWaT dataset

α	β	FP	TP	F1
0.0	1.0	604	35,616	0.7875
0.1	0.9	580	35,529	0.7853
0.2	0.8	571	35,285	0.7833
0.5	0.5	548	34,590	0.7741
0.7	0.3	506	34,548	0.7738
0.9	0.1	299	34,028	0.7684

Table 3.6 reports the effect of varying α, β in the number of detected FPs, TPs and the F1 score.

We can observe that by increasing α and reducing β it is possible to reduce the number of FPs (by a maximum of 50% when passing from 0.0 to 0.9) while limiting the drop in the number of TPs (3% from 0.0 to 0.9). Thus, the regulation of α and β allows parameterizing the sensitivity of USAD to meet the requirements of a production environment. With a model, it is possible to achieve different levels of sensitivity so that detection meets the needs of the different levels of hierarchy within Orange's supervision teams. Managers prefer a lower sensitivity levels, limiting the number of false positives but warning them in case of important incidents, while technicians will prefer a high level of sensitivity, allowing them to miss a minimum of incidents.

3.3.4.3 Training time

In this section the computational performance of USAD is studied and compared to OmniAnomaly, the method offering the closest performance in anomaly detection (see Table 3.5). To do this, the average time taken per epoch on the 5 public data sets is measured. The reference time for SMD, SMAP and MSL is the average time for one epoch over all entities (*i.e.* 28 machines of the SMD, 55 of the SMAP and 27 of the MSL). Both methods were trained using a NVIDIA GeForce GTX 1080 Ti.

Table 3.7 presents the obtained results. USAD provides good performance in unsupervised anomaly detection over multivariate time series while reducing training time by an average of 547 times.

Table 3.7: Training Time (min) per epoch on each dataset

Methods	SWAT	WADI	SMD	SMAP	MSL
OmniAnomaly	13	31	87	48	11
USAD	0.06	0.12	0.06	0.08	0.03
Acceleration factor	216	258	1331	581	349

3.3.4.4 Ablation Study

Using SMD, SMAP and MSL datasets, the effects of the two-phase training of USAD is investigated. Figure 3.4 presents a performance comparison in terms of the F1-score using USAD (Combined), USAD with only phase one training (Autoencoders) and with only phase 2 training (Adversarial). Training USAD without adversarial learning accounts to using the objective presented in equation 3.2, whereas suppressing the autoencoder accounts to use the objective from Equations 3.3-3.4.

GAN-inspired adversarial training represents an increase in performance of 5.88% (F1 score) with respect to the second best option which is USAD without adversarial training and 24.09% with respect to using only adversarial training. This can be explained by the amplified reconstruction error effect introduced by USAD regardless of the presence or not of an anomaly in the input window. Thus, USAD without its adversarial training cannot detect the anomalies closest to the normal data. USAD's poor performance with only adversarial training is explained by the fact that the method does not have the autoencoder training to orientate the weights in a favorable place before starting phase 2 of adversarial training. In conclusion, ablation of any of the training phases leads to poorer performance. For instance, both ablated versions of USAD have a lower F1 score than that of several of the bench-marked methods (Table 3.4, bottom).

3.3.4.5 Feasibility study

The automation of the supervision of complex IT systems is a challenge for Orange. After studying the properties of USAD and assessing its performance in using public datasets, the company must ensure that the method is as effective on its data.

Table 3.8 reports the results obtained in the internal dataset. USAD was able to detect all significant incidents in less than 30 minutes over the two months length of test data. For example, USAD was able to detect in less than 30 minutes an incident

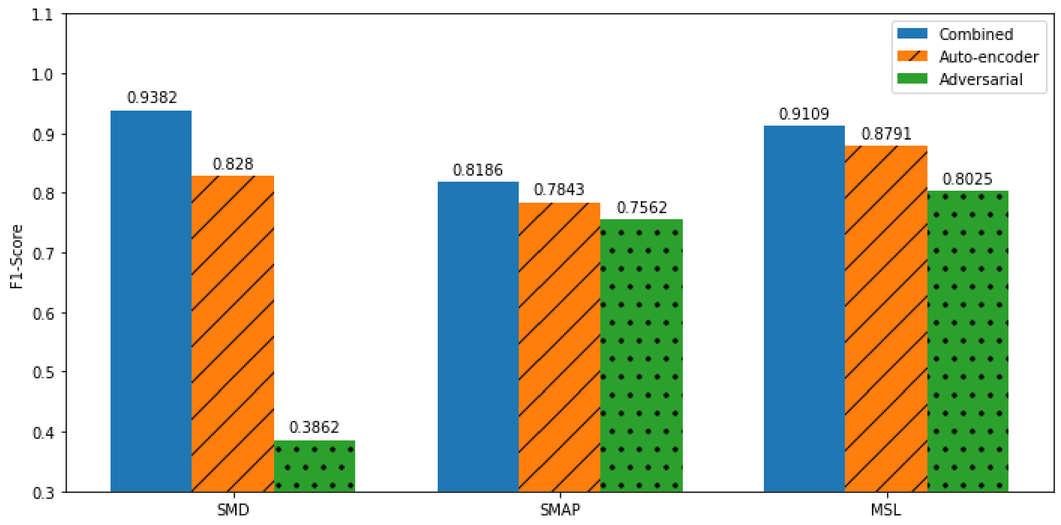

Figure 3.4: Impact with and without adversarial training on USAD

Table 3.8: Anomaly detection results on Orange internal Dataset (without point-adjust)

Method	Precision	Recall	F1-score
USAD	0.7448	0.6428	0.6901

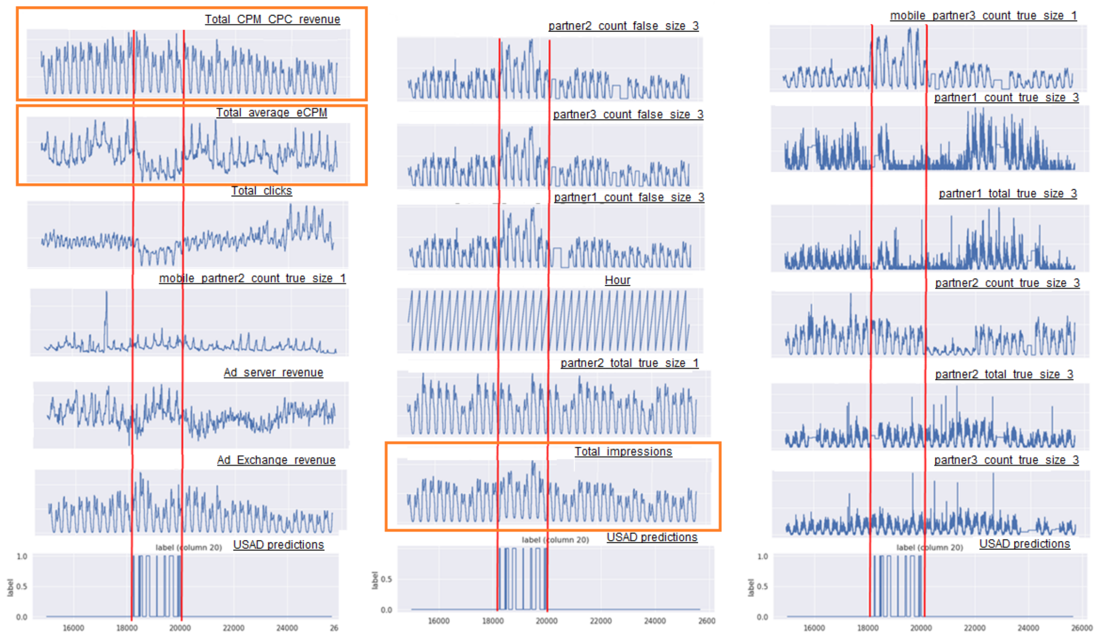

Figure 3.5: Example of a time series from the feasibility study where a configuration incident was detected by USAD. Twenty-four out of the 33 time variables are shown. The orange boxes highlight the variables referred to. In orange, the series referred to in section 3.3.4.5.

that took 24 hours to be detected by the operators in charge of supervision at Orange (Figure 3.5). This incident was caused by an error introduced in the configuration files allowing to assign advertising displays to unexpected partners. This caused the number of advertising displays (TOTAL IMPRESSIONS) to increase, while reducing the average display prices (TOTAL AVERAGE ECPM). As a result, important business indicators such as the revenue (TOTAL CPM CPC REVENUE) remained stable and so, the operators were unable to detect the incident quickly. Faced with the large amount of indicators to survey, people in charge of supervision concentrated their efforts on supervising indicators with high business impact, therefore, explaining, the 24 hours needed to detect this configuration incident.

3.4 Conclusion

This chapter proposes USAD, an UnSupervised Anomaly Detection for multivariate time series method based on autoencoders and trained within an adversarial training inspired by the Generative Adversarial Networks. Its autoencoder architecture makes it an unsupervised method and allows it to show great stability during the adversarial training. A set of five public reference datasets were used to to study the desired properties of USAD. The method demonstrated superior performance over state-of-the-art techniques on public reference datasets in terms of standard F1-score. In addition, its demonstrated fast training, robustness to the choice of parameters and stability allows for high scalability of the model within an industrial setting. USAD also provides the possibility to parameterize its sensitivity and to produce, from a single model, a set of detection levels. This possibility offers Orange's supervision teams essential functionalities enabling the use of the method in production on large-scale infrastructure. Since the teams need to be able to lower the sensitivity of the detection to prevent only major incidents when their workload becomes too high, the ability to multiply detection sensitivities during inference makes the model extremely scalable within the company and brings major advantages. First of all, it allows to limit the time needed to train the supervision models by limiting their number to just one. Secondly, a deep learning model put into production must be monitored and supervised by teams. Limiting the number of models allows to reduce the time spent supervising models in production and therefore free up time from supervisors to be devoted to different tasks.The feasibility study performed using Orange's internal data provided conclusive results which confirm that USAD suggests a promising direction for the automation of IT systems supervision at Orange. It also signaled some of the difficulties that might be encountered on the way to deployment and execution. For example, the data collection process (Section 3.3.3.2) faced the unexpected difficulty of gathering a continuous training period not containing too many anomalies. This is an interesting aspect that makes think on the infrastructure that will need to be put in place to have USAD succesfully deployed.

Chapter 4

From Univariate to Multivariate Time Series Anomaly Detection with Non-Local Information

Contents

This chapter presents a feature engineering strategy to transform univariate time series into a multivariate one by introducing non-local information in the augmented data. In this way, this strategy aims to address an intrinsic limitation of the features learned by DNNs, which is they rely on local information only. The performance of this combination is compared to each individual method and shows that the method achieves better performance without increasing computational time on a set of 250 univariate time series proposed by the University of California, Riverside at the 2021 KDDCup competition.

4.1 Introduction

This chapter proposes a novel feature engineering strategy to augment time series data in the context of anomaly detection using DNNs. The goal is two-fold. First, it aims to transform univariate time series into multi-variate time series to improve DNNs performance. Second, it aims to use a feature engineering strategy that introduces non-local information into the time series, which DNNs are not able to learn. To achieve this, a data structure called Matrix-Profile is used as a generic non-trivial feature. Matrix-Profile allows to extract non-local features corresponding to the similarity among the sub-sequences of a time series. The main contributions of this chapter are:

- It proposes an approach that transforms univariate time series into multivariate by using a feature engineering strategy that introduces non-local information to improve the performance of DNNs.
- It studies and analyzes the performance of this approach and of each method separately using the KDDCup 2021 dataset consisting of 250 univariate time series.

The rest of this chapter is organized as follows. Section 4.2 briefly reviews other works on feature engineering for anomaly detection in time series. The section 4.3 presents the transformation of univariate time series into multivariate one and the methods which constitute the framework. Section 4.4 describe the experiments and demonstrate the performance of this approach. This chapter concludes with some discussion and perspectives in section 4.5.

4.2 Related works

Different studies have raised the importance of feature engineering for the detection of anomalies and the superiority of multivariate models in time series. A first study conducted by Carta *et al* [16] shows that in network anomaly detection, the introduction of new features is essential to improve the performance of state-of-the-art solutions. Fesht *et al* [15] compare the performance of manual and automatic feature engineering methods on drinking-water quality anomaly detection. The study concludes that automatic feature engineering methods obtain better performances in terms of F1-score. Ouyand *et al* [86] shows that feature extraction is one of the essential keys for machine learning and proposes a method called hierarchical time series feature extraction used for supervised binary classification. Panda *et al* [87] demonstrates that appropriate feature engineering in addition to deep learning methods provides better detection of IoT-Botnet cyber attacks than methods alone. Fan *et al* [88] proposes three feature engineering methods based on Auto-Encoders and GANs. Their performance on building energy prediction shows that these methods outperform conventional feature engineering methods. Santos *et al* [89] compares univariate and multivariate models for predicting portfolio value at risk (VaR). Their comparison on both simulated and real data concludes that the multivariate models outperform their univariate equivalents. Finally, in [90], the authors conclude that multivariate models provided a more precise and accurate forecast with smaller confidence intervals and better measures of accuracy. Thus, studies have demonstrated the importance of feature engineering to improve anomaly detection models as well as the performance of multivariate methods compared to univariate ones on time series. Motivated by these ideas, this work aims to investigate how feature engineering using non-local information to achieve variate augmentation in time series can improve the performance of anomaly detection DNN models in univariate time series.

4.3 From univariate to multivariate time series

To take advantage of the performance of multivariate methods of anomaly detection on univariate time series it is necessary to transform the univariate time series into multivariate one. This can be achieved by adding external information to the time series, which requires specific domain knowledge. This strategy, instead, transforms the univariate time series into a multivariate one, without any further information than the original time series, and is generic in that no specific knowledge on what the time series represents is required.

The strategy consists in building another time series (i.e. another variate) by extracting non-local information from the raw time series, which DNN approaches fail to obtain as they typically operate in local neighborhood. To this end, the Matrix-Profile (MP) [53, 54], a data structure for time series analysis is used. The proposed strategy is illustrated in Figure 4.1.

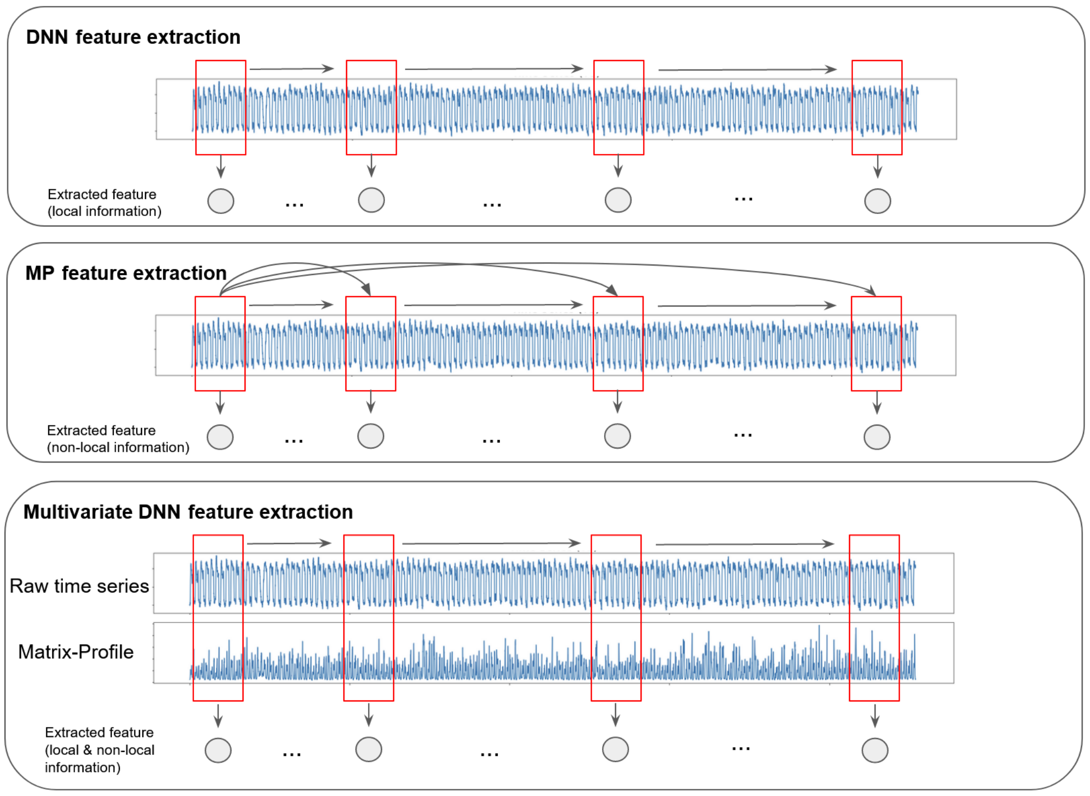
DNN feature extraction
Extracted feature (local information)
MP feature extraction
Extracted feature (non-local information)
Multivariate DNN feature extraction
Raw time series
Matrix-Profile
Extracted feature (local & non-local information)

Figure 4.1: Top: DNN automatic feature learning and extraction is limited to a local neighborhood, which is typically represented by the input window information. Middle: the matrix profile algorithm relies on non-local features, which are obtained by comparing every window of the time series. Bottom: the proposed strategy brings non-local feature information to a DNN by transforming the original univariate time series into a multivariate one by combining the raw time series and the non-local information obtained with matrix profile.

The Matrix profile estimates the minimal distance between all sub-sequences of a time series. Thus, the Matrix-Profile value for a given sub-sequence is the minimum pairwise Euclidean distance to all other sub-sequences of the time series. A low value in the matrix profile indicates that this sub-sequence has at least one relatively similar sub-sequence located somewhere in the original series. In [55], it is shown that a high value indicates that the original series must have an abnormal sub-sequence. Therefore the matrix profile can be used as an anomaly score, with a high value indicating an anomaly.

In this approach, the anomaly score obtained by Matrix-Profile is used over a given time series and merge it point-by-point with the original data. This can be thus seen as a data augmentation procedure using non-local information from the same signal.

As the new time series is just a multivariate time series, any given anomaly detection method can be used to identify anomalous points in it. In this work, three different estimation model-based techniques [34] are investigated as base anomaly detection methods. Among these category of methods, the auto-encoder [66] is among the most commonly used. An auto-encoder (AE) is an artificial neural network combining an encoder E and a decoder D. The encoder part takes the input window W and maps it into a set of latent variables Z, whereas the decoder maps the latent variables Z back into the input space as a reconstruction $\widehat{W}$. The difference between the original input vector W and the reconstruction $\widehat{W}$ is called the reconstruction error. Thus, the training objective aims to minimize this error. Auto-encoder-based anomaly detection uses the reconstruction error as the anomaly score. Time windows with a high score are considered to be anomalies [21].

Alongside the AE, a more complex approach based on a Variational AutoEncoder (VAE) coupled with a recurrent neural network is considered, the Long Short-Term Memory Variational Auto-Encoders (LSTM-VAE) [11]. In the LSTM-VAE, the feed forward network iof the VAE is replaced by a Long Short-Term Memory (LSTM), which allows to model the temporal dependencies. As in the AE, the input data is projected in a latent space. However, differently from the AE, this representation is then used to estimate an output distribution and not to simply reconstruct a sample. An anomaly is detected when the log-likelihood is below a threshold.

The third estimation model-based method considered is denoted UnSupervised Anomaly Detection (USAD) [2]. USAD is composed of three elements: an encoder network and two decoder networks. The three elements are connected into an architecture composed of two auto-encoders sharing the same encoder network within a two-phase adversarial training framework. The adversarial training allows to overcome the intrinsic limitations of AEs by training a model capable of identifying when the input data does not contain an anomaly and thus perform a good reconstruction. At the same time, the AE architecture allows to gain stability during adversarial training of the two decoders.

The architecture is trained in two phases. First, the two AEs are trained to learn to reconstruct the normal input windows. Secondly, the two AEs are trained in an adversarial way, where the first one seeks to fool the second one, while this latter one aims to learn when the data is real (coming directly from the input) or reconstructed (coming from the other autoencoder). As with the base AE, the anomaly score is obtained as the difference between the input data and the data reconstructed by the concatenated autoencoders.

4.4 Experiments and Results

This section first describes the datasets used and the experimental setup used in this work. Then, The performance of the proposed approach is studied and compared against other techniques.

4.4.1 Datasets

In this experiments 250 univariate time series proposed by the University of California, Riverside at the 2021 KDDCup competition, consisting of univariate time series from many different fields are used. The 250 time series are composed of a training part containing data considered as normal and a test part containing one anomaly. The time series range from 6680 points for the smallest to 900000 points for the largest. The length of the training set represents on average 31% of the total length of the time series (i.e. a training on the first 31% points of the time series and a test on the next 69% points) with a minimum length of 2.5% and a maximum of 76.9%. All the time series are min-max normalized.

4.4.2 Experimental setup

The percentage of correctly labeled series is used to evaluate the performance of the method. A time series is considered to be correctly predicted when the index of the point labeled as anomalous is included in a window of 100 points around the true anomaly.

This method is compared against the matrix-profile (MP), the auto-encoder (AE), the LSTM-VAE and USAD without the transformation of the time series. The performance of the three anomaly detection methods AE, LSTM-VAE and USAD on a transformed univariate time series obtained using only non-local information, i.e. with Matrix-profile (MP-AE, MP-LSTM-VAE and MP-USAD) is compared. The AE, LSTM-VAE and USAD's performance using the proposed multivariate transformation, consisting of the original raw time series and the series obtained

Table 4.1: Hyper-parameter settings of the different methods

Method	Paramaters
MP	$window\_size = 100, discords = True$
AE	$window\_size = 100, latent\_dimension = 10, Epochs = 100$
LSTM-VAE	$window\_size = 100, Epochs = 100$
USAD	$window\_size = 100, latent\_dimension = 10, Epochs = 100$

with MP, respectively (TS+MP)-AE, (TS+MP)-LSTM-VAE and (TS+MP)-USAD are assessed. To validate the relevance of the use of non-local information in the transformation of the time series, an identical combination with a local feature engineering strategy is considered. In particular, in the experiments the moving average (MA), respectively (TS+MA)-AE, (TS+MA)-LSTM-VAE and (TS+MA)-USAD) is used.

4.4.2.1 Implementation.

The AE is implemented using Pytorch and publicly available implementations is used for MP[1][1], LSTM-VAE[2] and USAD[3]. Table 4.1 details the hyper-parameter setup used for each method. Where a parameter is not specified, it indicated that those set by default in the original implementation were used.

All experiments are performed on a machine equipped with an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz and 270 GB RAM, in a docker container running CentOS 7 version 3.10.0 with access to an NVIDIA GeForce GTX 1080 Ti 11GB GPU.

4.4.3 Results

Table 4.2 presents the results obtained by the different methods in terms of performance accuracy and computational times. Interestingly, the performance of DNN-based methods on univariate time series is very low and largely surpassed by the more conventional approach, the matrix profile. However, once the same techniques use the proposed data transformation strategy, an important boost in their performance can be observed. The Auto-Encoder and the LSTM-VAE score almost 2.3

[1]`https://stumpy.readthedocs.io`

[2]`https://github.com/TimyadNyda/Variational-Lstm-Autoencoder`

[3]`https://github.com/robustml-eurecom/usad`

Table 4.2: Methods performance and computational time.

Method	Performance	Train and Test time ($s\times10^3$)
Matrix-Profile	0.416	1.47
AE	0.236	22.00
LSTM-VAE	0.198	85.31
USAD	0.276	29.00
MP-AE	0.292	22.16
MP-LSTM-VAE	0.344	84.30
MP-USAD	0.404	29.10
(TS+MA)-AE	0.148	22.38
(TS+MA)-LSTM-VAE	0.134	85.43
(TS+MA)-USAD	0.176	29.12
(TS+MP)-AE	**0.536**	22.50
(TS+MP)-LSTM-VAE	0.446	85.83
(TS+MP)-USAD	0.488	29.28

times higher when the combination of the matrix profile and real data is used as input instead of the original data. Similarly, USAD's performance increases by 1.8 times when the matrix profile and raw time series combination is used compared to its performance using only the raw time series.

Nevertheless, the non-local transformation alone is not enough to boost the performance of DNN methods. For instance, if the input consists only of the univariate time series transformed using the matrix profile, while there is some increased performance, this one is milder than when using a multivariate time series. This confirms that DNN methods perform better in a multivariate setup for anomaly detection.

Regarding the use of local features, i.e. the moving average, adding it does not allow USAD, LSTM-VAE and AE to increase their performance. Indeed, the combination of raw time series and moving average degrades the performance of AE and USAD by about 0.1 and the performance of LSTM-VAE by about 0.06. This suggests that any local features that might be discriminative can be extracted by the DNNs and introducing new manually crafted ones may be detrimental.

Finally, as it is expected, the computational time of DNN-based methods is much longer than the MP. However, what is interesting in the findings is that the computa-

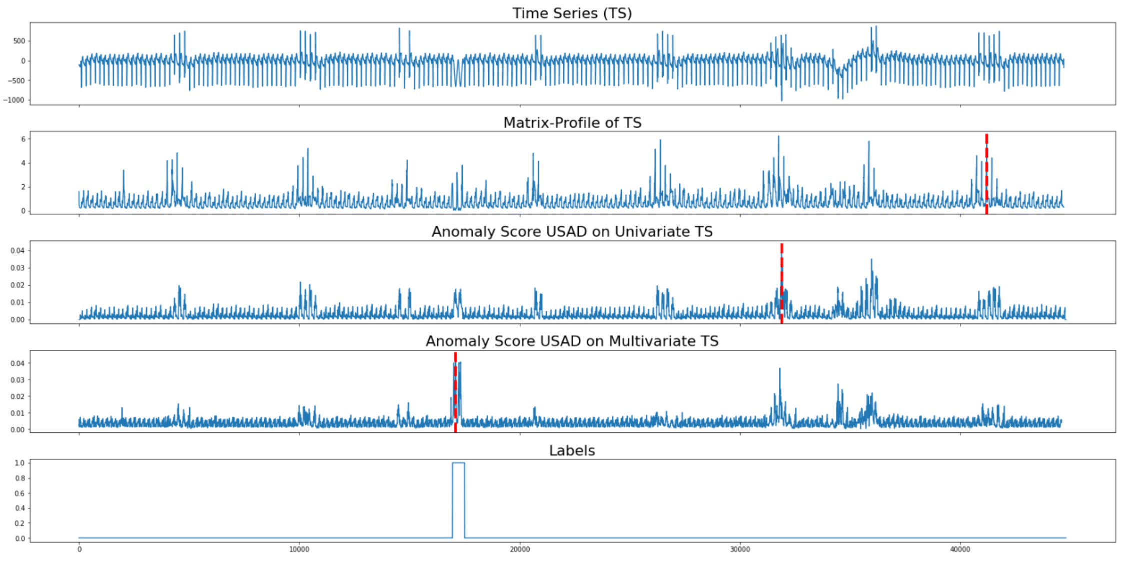

Figure 4.2: Anomaly correctly detected by USAD only with a multivariate input combining Matrix-Profile and original univariate time series

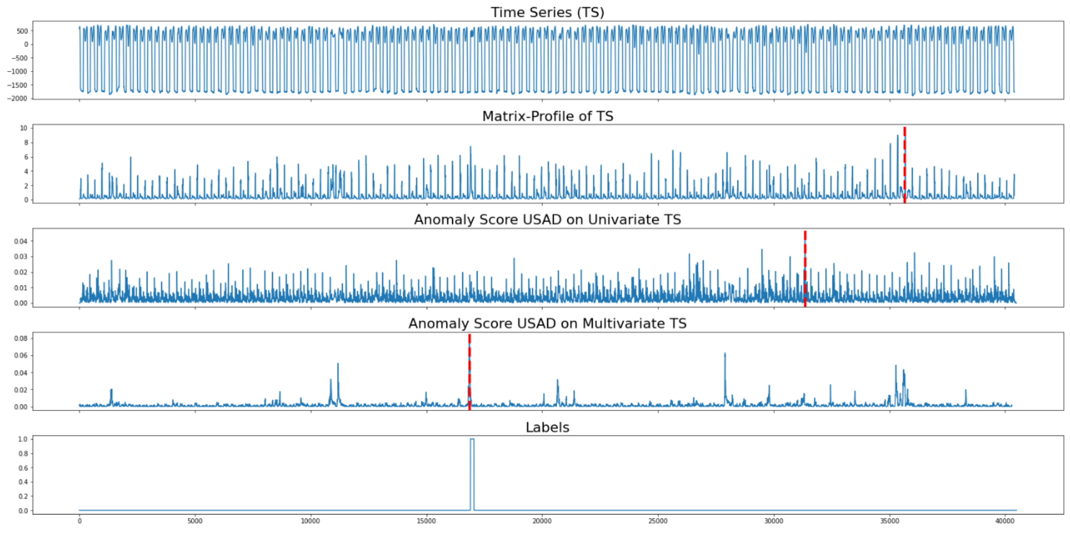

Figure 4.3: Anomaly correctly detected by USAD only with a multivariate input combining Matrix-Profile and original univariate time series

tional time of DNN methods is very little impacted when the dimension of the time series increases. In fact, the AE's computational time goes from 21993 seconds in the fastest univariate configuration to 22491 seconds in the multivariate case. This means an increase of only 2.2% on computational time for a gain in performance of 230%.

4.5 Discussion and Conclusions

This chapter proposes an approach to augment univariate time series using a feature engineering strategy that introduces non-local information in the generation of an additional variate to the series. In this way, this strategy expects to address a limitation of DNNs, as they are not conceived to learn automatically non-local features. Automatic non-local feature extraction is achieved by relying on the Matrix-Profile, a method that computes the minimum pairwise Euclidean distance of all subsequences of the time series, and by combining its output with the original time series.

The KDDcup 2021 competition containing 250 univariate time series is used to study the performance of this method. The performance analysis highlighted the relevance of transforming the univariate time series using the proposed feature engineering and data augmentation strategy. The results show that introducing non-local information to augment the dimension of the series improves the performance of DNN methods. For instance, by using a very simple method, such as an autoencoder, a gain in performance of 230% was obtained, without significantly increasing the computational time. As such, the preliminary results suggest that non-local information represents an important source of additional information that can increase performance of DNN methods.

While this approach focuses on the particular case of transforming uni- to multivariate time series, this idea could be used to augment time series, which are multivariate at origin, as a way to introduce non-local information. In this work, three methods of anomaly detection based on Deep Neural Networks were used in combination with Matrix profile. The good performance on a simple auto-encoder, a recurrent network such as LTSM-VAE and USAD, a state-of-the-art neural network, suggest that this combination could generalize to other DNN methods. Therefore, future works should explore other feature engineering techniques that can provide non-local information, as well as other multivariate DNN anomaly detection methods.

Finally, the findings are consistent with one of the results of the time series prediction competition, the M4 challenge [91], which highlighted the predictive power of ensemble approaches combining learning-based with more conventional statistical methods. Due to the great success of DNN methods in the recent years, it is now often the case that more traditional methods are overseen. The results suggest that the use of hybrid approaches should be further explored.

Chapter 5

Are Deep Neural Networks Methods Needed for Anomaly Detection on Multivariate Time Series?

Contents

This chapter studies the anomaly detection performance of sixteen conventional, machine learning-based and, deep neural network approaches on five real-world open datasets. By analyzing and comparing the performance of each of the sixteen methods, it shows that no family of methods outperforms the others. Thus, it is impossible to affirm that deep neural networks are superior to previous methods.

5.1 Introduction

This chapter presents a comparison between conventional methods, machine learning-based and more recent DNN-based approaches. This work is motivated by different recent works, which have reported on the limitations and drawbacks of DNN-based methods in different application fields [36, 92, 93, 94]. While some of this works have focused on pointing out to the weaknesses of DNN-based methods [92, 93], other works have been able to demonstrate the superiority of more conventional approaches [36, 94].

The main contributions of this chapter are the following:

- It studies and analyze the performance of sixteen of the most commonly used methods for anomaly detection in multivariate time series grouped into three categories: Conventional, machine learning-based and DNN-based over five open real-world data sets.
- It discusses the need for DNN-based approaches and the importance of conventional methods in future benchmarks for multivariate time series anomaly detection.

The rest of this chapter is organized as follows. Section 5.2 briefly reviews other works comparing modern DNN-based methods to previous non-DNN-based works. Sections 5.3 and 5.4 describe the experiments and analyze the performance on the data sets.

5.2 Related work

Different studies have raised the question about the real gain of DNN-based methods in several application fields. A first study by Fernandez-Delgado *et al.* [95] presented a comprehensive evaluation of 179 classifiers from a collection of 17 families of methods, (including neural networks, support vector machines, random forests, generalized linear models, nearest-neighbors, partial least squares and principal component regression among others) on a large number of classification tasks from the UCI Machine Learning Repository[1]. The empirical studied suggested that a relatively simple algorithm, the random forest, was overall the best classifier in terms

[1] `https://archive.ics.uci.edu/ml/index.php`

of accuracy. In [94], Jiao *et al.* showed how conventional linear regression methods outperform DNN-based techniques in two showcased optical imaging problems, i.e. an optical cryptosystem attack and blind reconstruction in single pixel imaging. Autun *et al.* [92] proposed a stability test to demonstrates how DNN-based methods for image reconstruction are very sensitive to tiny perturbations in the input images during training, which leads to unstable results. Furthermore, Heaven [93] showed small changes in a DNN's input, usually imperceptible to humans, can destabilize the best neural networks, thus pointing to the lack of robustness of DNN-based methods and their dependence on large amounts of data. Most recently, in the context of medical image segmentation, Fu *et al.* [96] showed that simpler DNN configurations have better generalization properties than state-of-the-art models but more complex DNN models, thus challenging the current trend towards continuously increasing the model complexity.

In the specific context of time series analysis, the results of the M3 challenge on time series forecasting [36] showed that the accuracy of learning-based models, in general, was lower than that one of conventional approaches, while their computational requirements were considerably greater than those of conventional statistical methods. Similarly, one of the main outcomes of the follow-up M4 competition [91] was that none of the pure ML methods participating was able to outperform the combination of learning-based and statistical (i.e. conventional) methods. For instance, only one DNN approach was more accurate than a naïve random walk model that assumed future values will be the same as those of the last known observation [97].

5.3 Experimental setup

This section describes the datasets and the performance metrics used in the experiments.

5.3.1 Public Datasets

The same five datasets as the previous chapter were used in this experiments. Table 5.1 summarizes the datasets characteristics and the type of anomalies present in each dataset.

Table 5.1: Benchmarked Datasets. (%) is the percentage of anomalous data points in the dataset.

Dataset	Train	Test	Dimensions	Anomalies (%)	Type of anomaly
SWaT	496800	449919	51	11.98	P, C, Col
WADI	1209601	172801	123	5.99	P, C, Col
SMD	708405	708420	28*38	4.16	P, Col
SMAP	135183	427617	55*25	13.13	P, Col
MSL	58317	73729	27*55	10.72	P, Col

P: Point Anomaly, C: Contextual anomaly, Col: Collective anomaly

5.3.2 Evaluation Metrics

As in the previous chapter, performance is assessed by comparing the results of each evaluated method with the annotated ground truth. Precision (P), Recall (R), and F1 score (F1) were used to evaluate anomaly detection performance.

The average precision (AP) is computed from anomaly scores. AP summarizes a precision-recall curve as the weighted mean of precision achieved at every given anomaly score threshold, with the increase in recall from the previous threshold used as the weight:

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

where P_n and R_n are the precision and recall at the n-th threshold. Finally, the area under the Receiver operating characteristic (AUC) curve is computed.

In the experiments, a window is considered labeled as an anomaly as soon as one of the points it contains is detected as anomalous. For methods that do not use a time window, an anomaly is considered detected if the detection occurs within 12 time points on either side of the anomaly.

the *point-adjust* approach is used as in the previous chapter since anomalous observations usually occur in the form of contiguous anomaly segments.

5.4 Experiments and Results

The performance of the sixteen methods discussed in Section 2.2 is studied on the five datasets discussed in Section 5.3.1 in terms of precision, recall, F1-score, AP and AUC (5.4.1). An analysis of the results is performed by comparing the anomalies detected by the conventional methods and the learning-based approaches (5.4.2). The impact of the training set size (5.4.3) is studied. Finally, a synthesis of the results is proposed and discussed (5.4.4).

5.4.1 Benchmark Performance

The performance of the sixteen anomaly detection methods are assessed on the five public real-world datasets. Train and testing splits are reported in Table 5.1.

As not all of the anomaly detection approaches provide a mechanism for selecting anomaly thresholds,a thousand possible anomaly thresholds are tested for each model. The anomaly score is normalized between 0 and 1 and then test on one thousand thresholds in steps of 0.001. For every method, the results associated to the highest achieved F1 score, the AUC and the AP are reported.

Table 5.2 reports mean, minimum and maximum performance of each group of methods (conventional, ML or DNN) on every dataset. Where the minimum value is reported as "-", it denotes that a method in that category did not provide any results after more than ten days of execution in my experiments. Concretely, this occurrs for VAR and SSA on SWaT and WADI dataset, and for MP on WADI dataset. At a first instance, the results suggest that learning-based methods, and in particular DNN-based, outperform conventional ones in terms of average performance, while often reporting the highest performing models (maximum value). Instead, conventional methods typically report the lowest performing method (minimum values). This behaviour is most evident in the SWaT and MSL datasets.

The estimation of precision, recall and the F1 score requires the computation of a threshold, which can be extremely complicated to define. The AUC-ROC and AP metrics allow to obtain a score that takes into account the ability of the methods to rank the anomalies without having to define a threshold and provide a complementary information w.r.t P, R and F1. In particular, AP is more sensitive to the positive class, i.e. the anomalies, than the AUC-ROC, so it is better suited to highly unbalanced data [98], as it is the case in anomaly detection problem. A detailed comparison of the performance of the different methods in terms of AP shows that conventional methods outperform all families of methods in the MSL dataset and they are the second best ranked after DNN methods for the remaining datasets.

A Welch's t-test over the obtained results indicates that only AP in WADI shows a significant difference among the different families of methods. Thus, only WADI can differentiate performance between the three categories of approaches. This result indicates that while DNN methods would perform better in some cases, more globally, there is no significant difference in performance between the method families in four out of five datasets.

5.4.2 Analysis of WADI

A more detailed analysis is performed on the performance results obtained for each type of method. The WADI dataset is used because it is the only dataset that shows a significant difference in performance between the method families.

Figure 5.1 (top) represents the false negatives of all the conventional methods. In other words, a value of 1.0 represents an anomaly labeled as normal by all the conventional techniques. The second plot represents the same information for the machine learning models and the third one for DNN approaches. The fourth series in green represents the false negatives of the conventional approaches which are true positives for the DNNs methods. In other words, a value of 1.0 corresponds to an anomaly detected by at least one DNN algorithm but by no conventional methods. Finally, the purple series shows the ground truth labels. Overall, WADI contains 14 anomalies distributed over the test (Figure 5.1 bottom).

An inspection of the plots shows that four anomalies are not detected by any of the DNNs approaches, while seven anomalies are not detected by the conventional and machine learning models. Thus, these three anomalies explain the performance gap between the DNNs and conventional methods, which are the second best performing family in terms of AP on this dataset (see fourth series).

Figure 5.2 presents a detailed view of the first anomaly from the green series. It is caused by the variate 1_MV_001 increasing its value to 2.0 before the variate 1_LT_001 has reached its mininimal expected value (40). It is a contextual anomaly, as the separate behavior of each variate does not constitute an anomalous behavior on its own. The remaining two anomalies also shows that they are contextual anomalies. This suggests that the performance gap between DNNs and conventional methods comes from a better detection of contextual anomalies by DNNs approaches.

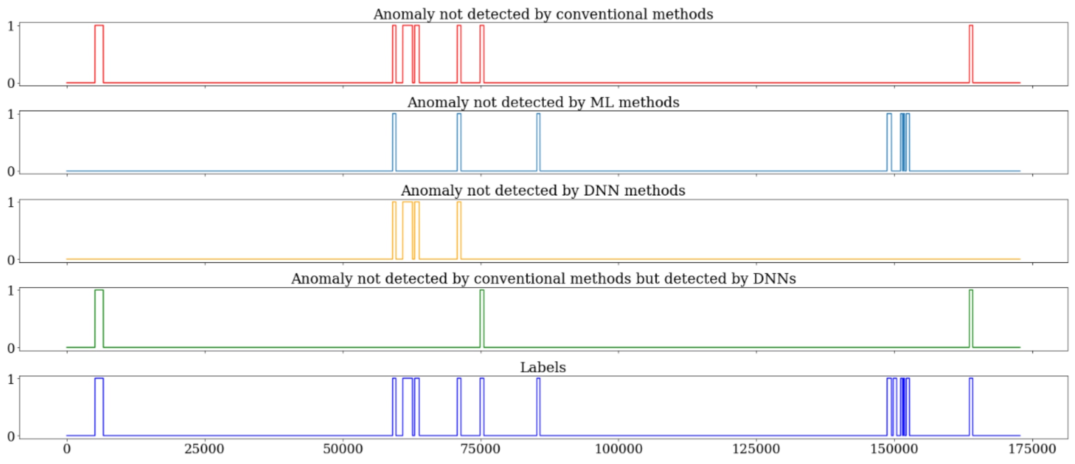

Figure 5.1: Top row: False negatives of conventional methods; Second row: False negatives of Machine learning algorithms; Third row: False negatives of DNNs approaches; Fourth row: False negatives of conventional methods that were well predicted by DNNs methods; Fifth row: Ground truth Labels

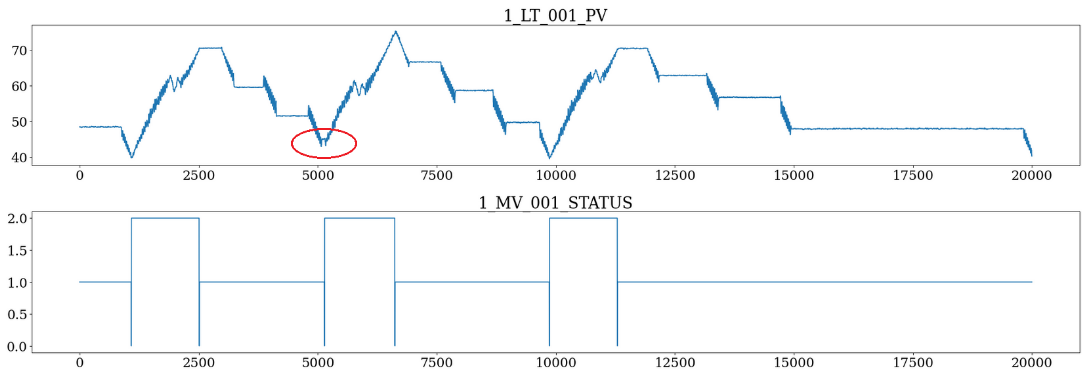

Figure 5.2: Contextual anomaly in red due to the activation of a motorized valve (1_MV_001) which causes the filling of the tank (1_LT_001) before it reaches the switching threshold (located at 40)

5.4.3 Impact of training set size

Using WADI and SWaT datasets, The impact of the training set size on the performance of the methods is studied. the size of the training set is reduced by keeping the most recent points, i.e. the closest to the test set. 10% , 25% , 50% and 75% of the original training points are kept. For each set, one model per method is trained and its performance assessed. Figures 5.3 and 5.4 present respectively the AUC and the AP obtained on the SWaT dataset and Figures 5.5 and 5.6 on the WADI dataset. Conventional methods globally obtain better results when the dataset size is less than 50% of the original one. For example, on the SwaT dataset, MP, PCA and ICA outperform in terms of AP all ML and DNN methods when the training set is 50% or less. Their performance is matched by OC-SVM at 50%, and it is outperformed only when three quarters of the training dataset is retained. The performance of conventional approaches remains relatively constant, despite changes in the training set size, meaning this has little impact in their performance. Instead, ML and DNN methods perform better when increasing the size of the training dataset, except for IF on the WADI dataset which performs slightly better when only 50% of the dataset is kept. This can be explained by the fact that the isolation of anomalous points can be more complex when the training set is larger. In general, it is only above 50% that DNNs approaches seem to perform better than conventional methods.

5.4.4 Discussion

The experimental results for multivariate anomaly detection using conventional, machine learning and DNN methods did not prove the superiority of one category over the others over the datasets. However, there are some indications that in the particular case of contextual anomalies, DNN-based methods perform better and may be necessary. However, this analysis is only true when the dataset is large enough for them to learn properly, otherwise conventional methods seem to outperform them. While a detailed analysis of the performance of different methods in contextual anomalies is, therefore, necessary, it is complicated to obtain real datasets containing contextual anomalies to validate this hypothesis. Indeed, contextual anomalies are by definition difficult to detect visually by experts since taken alone the series seem normal. It is therefore challenging for the community to obtain datasets containing contextual anomalies without these having been manually induced. In view

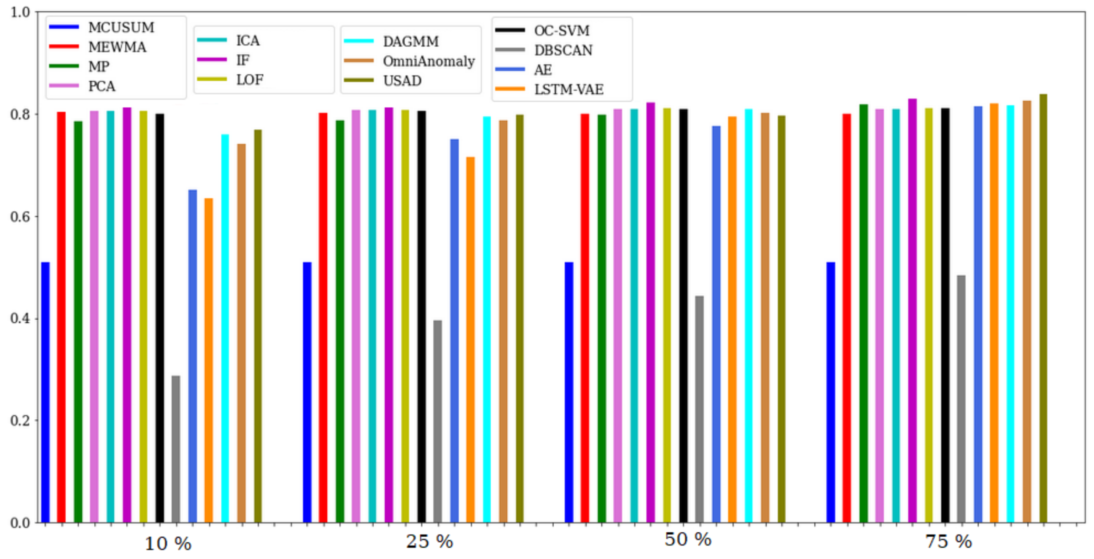

Figure 5.3: Area under the Receiver operating characteristic curve (AUC) on SWaT dataset.

of these results, it is impossible to affirm that the most complex methods based on DNNs have allowed a real advance in the problem of anomaly detection in multivariate time series. It is therefore essential for the community to reintegrate machine learning methods, but also conventional methods in the benchmarks to ensure that the new methods proposed improve the performance in the detection of anomalies in time series. Furthermore, the lack of good results from the most complex methods in some cases, invites to encourage the community to continue to propose methods from all three categories and not to focus only on DNN-based methods.

This chapter focuses on some aspects of the performance of different types of methods, while leaving aside a detailed analysis of the computational times required by each technique. It is now relatively well established that ML and DNN-based approaches are more computationally demanding than conventional methods. This is mainly due to the fact that training time can be very expensive, especially for DNN-based methods. Conventional methods have the advantage of not requiring a training phase, which is where ML and DNN-based methods consume most of the computational resources. However, at inference time, DNN-based methods can be much faster. As an example, Matrix-Profile is a fast conventional approach [99], which, however, requires to compute the distance of every new sub-sequence w.r.t. old available data.

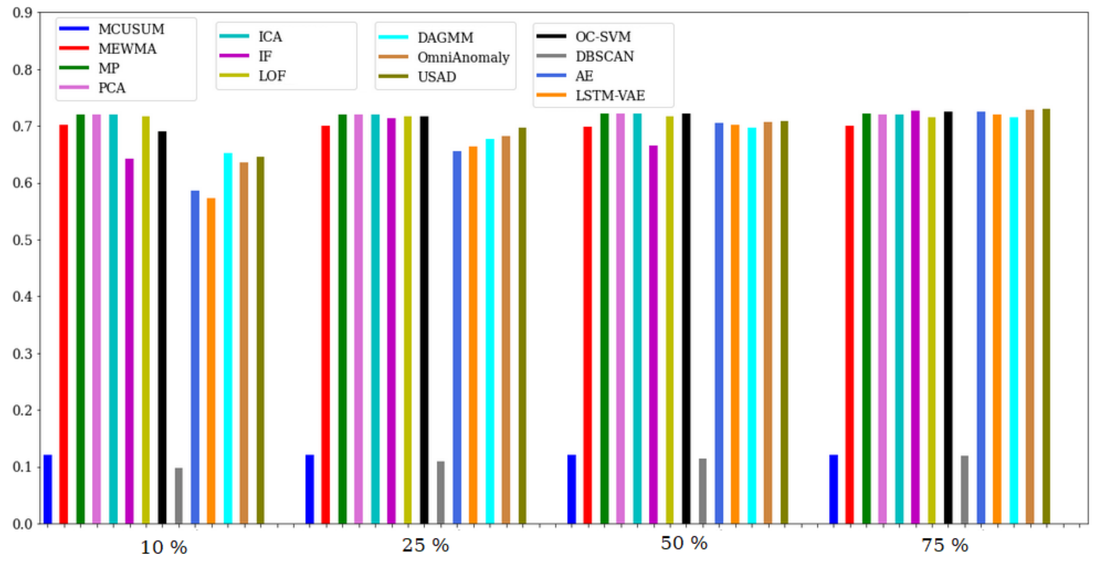

Figure 5.4: Average precision (AP) on SWaT dataset.

Another important observation from this study regards scalability. Some conventional methods failed to converge when the data set was too large (i.e. VAR and SSA on SWaT and WADI and MP only on WADI). Therefore, the size of the datasets is an important criterion in the choice of a methods to use. While DNN-based methods are a more suitable choice for larger sets of data, despite their overhead in computational time during training, conventional approaches seem to be a better choice in a small data regime.

Finally, an important point to consider is also the difficulty to reproduce the results of DNN-based methods compared to the other two categories of methods. Indeed, there is a plethora of open implementations of conventional [1] and machine learning based methods [2], while some DNN-based approaches can be difficult to implement, sometimes no implementation is available or it is difficult to set up.

5.5 Conclusion

This chapter provides a comparative analysis of conventional, machine learning-based and DNN-based methods. The performance of sixteen algorithms is evaluated on five open real-world datasets to understand whether the complexity provided by

[1]PYOD : https://github.com/yzhao062/pyod
[2]Scikit Learn : https://scikit-learn.org/

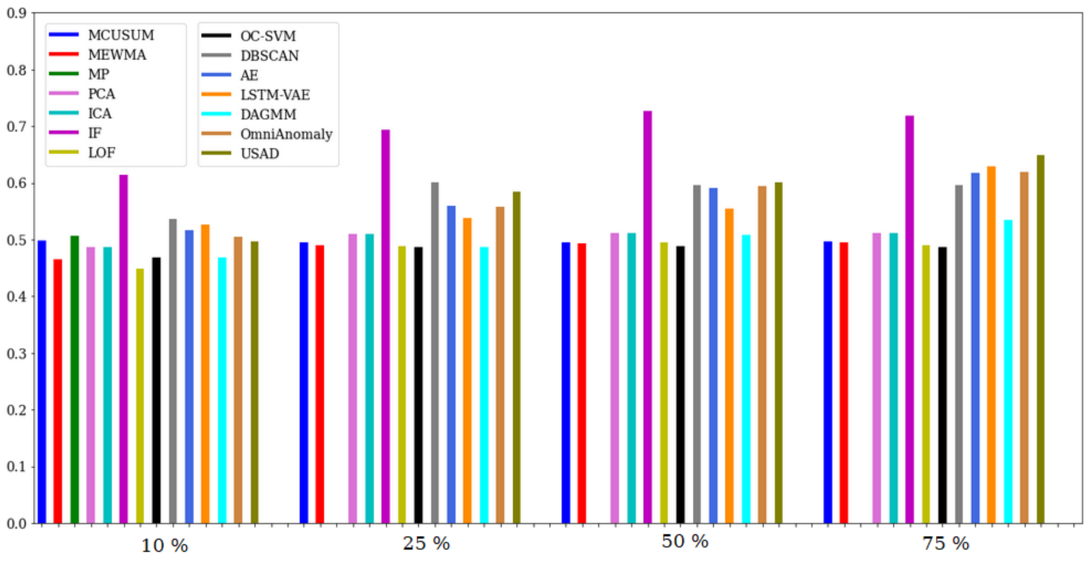

Figure 5.5: Area under the Receiver operating characteristic curve (AUC) on WADI dataset.

DNN-based approaches is necessary for anomaly detection in multivariate time series. The impact of the training set size on the performance of these methods is studied. The performance analysis did not allow to observe a superiority of one category of methods over the others in terms of performance. The results showed that all three categories can outperform the other two according to the criteria of the dataset. The DNN based methods seem to perform better when the dataset contains contextual anomalies. However, this finding could only be made on one of the five datasets, so more experimentation is needed to confirm that DNN methods outperform the other categories in terms of contextual anomaly detection. Furthermore, if the training set is not large enough, the conventional methods outperform the other two categories. Thus, the size of the training set is an important criterion in the choice of the category of methods to detect anomalies in multivariate time series. Some conventional methods have failed to scale on large data sets.

In view of all these results, it is not possible to affirm that the performance claimed in recent years in benchmark papers that omitted one of the three categories are not illusory [20]. I therefore encourage the community to reincorporate the three categories of methods in the benchmarks of anomaly detection in multivariate time series. Moreover, it seems essential to multiply the number of datasets compared in the benchmarks in order to ensure that all eventualities are covered. For this, the community will have to obtain new real world datasets containing contextual

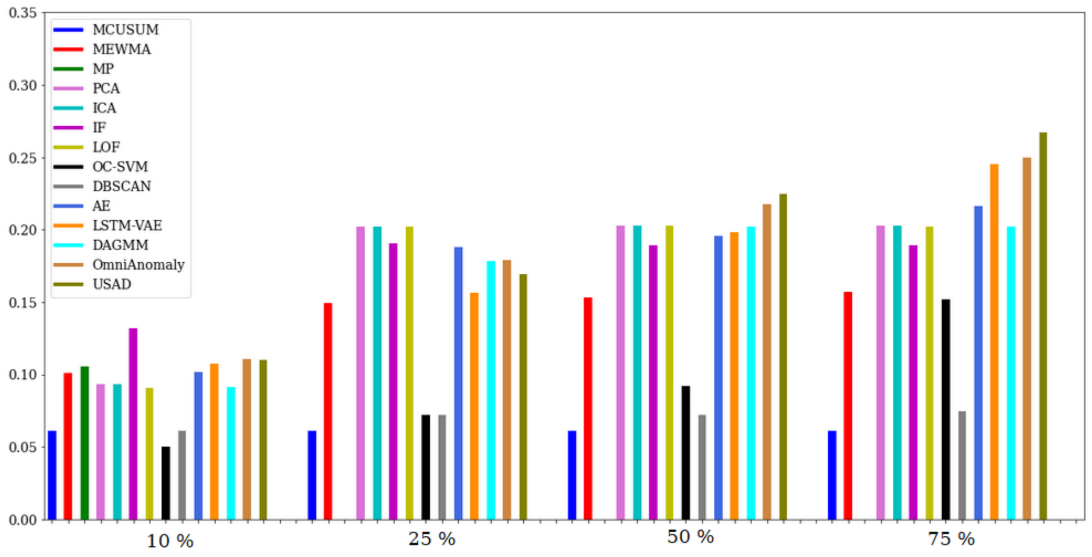

Figure 5.6: Average precision (AP) on WADI dataset.

anomalies. Indeed, the difficulty for experts to visually label contextual anomalies in multivariate time series makes it difficult to obtain test sets covering this criterion. It is then complicated to assert that DNN methods are necessary although it seems that they are able to outperform conventional approaches in this domain when the data set is large enough.

Table 5.2: Summarized performance measures per each category of methods. For each measure the mean ± standard deviation, and the [minimum, maximum] values per category are reported. Values are reported as %

Dataset	Metric	Conventional	Machine learning	DNN
SWaT	P	75.1±31.4 [-,92.5]	**84.8**±16.8 [60.2,98.2]	**84.8**±13.6 [71.2,99.1]
	R	74.5±2.6 [-,78.5]	80.8±11.4 [73.8,97.7]	**82.4**±12.3 [70.4,98.3]
	F1	70.6±22.2 [-,82.7]	80.9±4.4 [74.5,84.3]	**82.1**±2.0 [79.7,84.6]
	AUC	74.9±13.4 [-,81.5]	76.1±11.4 [59.1,83.3]	**83.0**±0.8 [82.2,84.0]
	AP	59.4±26.4 [-,71.6]	57.0±30.0 [12.0,72.7]	**72.8**±0.7 [71.5,73.4]
WADI	P	29.8±15.5 [-,50.8]	**49.4**±42.2 [9.1,98.5]	39.9±16.8 [22.3,64.5]
	R	30.1±4.1 [-,32.5]	**64.9**±37.5 [15.9,95.1]	42.9±31.3 [19.8,98.0]
	F1	27.7±6.0 [-,32.5]	34.6±19.3 [16.6,61.7]	**35.8**±8.9 [20.9,43.0]
	AUC	50.8±0.8 [-,51.3]	57.6±10.0 [48.6,70.8]	**63.9**±2.8 [59.3,66.8]
	AP	16.1±6.7 [-,20.3]	15.4±5.7 [7.5,20.4]	**25.5**±3.0 [21.5,29.3]
SMD	P	72.9±23.4 [21.7,89.1]	72.2±18.8 [47.0,92.6]	**86.7**±11.7 [67.3,98.1]
	R	70.1±7.1 [56.3,78.0]	**95.6**±6.2 [86.3,99.0]	86.9±8.0 [78.8,96.2]
	F1	62.9±20.8 [20.4,80.1]	75.6±13.1 [56.8,87.2]	**84.8**±9.3 [72.3,94.4]
	AUC	66.2±3.9 [59.9,70.1]	72.4±4.8 [66.3,77.1]	**74.4**±3.2 [69.5,77.5]
	AP	24.9±7.1 [17.2,34.6]	23.3±9.8 [8.7,29.5]	**29.6**±3.7 [26.1,35.2]
SMAP	P	60.0±17.4 [38.7,78.4]	**74.9**±9.7 [60.5,80.5]	72.7±4.2 [66.3,77.0]
	R	96.4±3.5 [89.0,98.8]	92.1±11.5 [74.9,98.7]	**98.3**±0.9 [97.6,99.8]
	F1	64.5±16.4 [42.8,82.0]	**79.6**±8.5 [66.9,84.3]	77.4±4.3 [71.2,81.9]
	AUC	59.8±4.2 [52.7,65.7]	58.4±4.6 [52.8,63.9]	**63.5**±1.7 [60.7,65.2]
	AP	24.4±4.0 [16.7,28.8]	21.5±5.7 [14.7,26.4]	**27.3**±1.6 [24.9,29.1]
MSL	P	71.6±15.0 [49.3,87.3]	81.0±8.9 [70.1,91.3]	**85.3**±5.9 [75.6,91.4]
	R	94.2±4.3 [84.7,97.1]	83.6±6.9 [73.7,89.8]	**96.0**±4.0 [88.9,98.0]
	F1	74.4±15.1 [51.0,90.1]	78.1±7.5 [69.3,86.0]	**87.0**±3.9 [81.1,91.1]
	AUC	60.9±3.0 [55.3,64.7]	60.1±3.6 [56.3,63.4]	**62.5**±0.5 [61.8,63.2]
	AP	**25.0**±4.0 [20.2,29.8]	21.6±5.9 [15.1,27.0]	24.4±1.5 [22.5,26.1]

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

This thesis focuses on the unsupervised detection of anomalies in multivariate time series. Being able to detect anomalous behavior is an increasingly complex task due to the growing size and complexity of the systems being measured. However, anomaly detection is one of the key issues for a large number of domains such as finance, health or supervision. To this end, this thesis proposed a method called USAD that can partially address the problem of anomaly detection in multivariate time series such as the automation of the supervision of large computer systems in Orange since it meets the constraints of scalability and fast learning while offering possibilities of managing the level of detection and very good performance compared to other deep neural network methods of the state of the art. This thesis also presents a feature engineering strategy to improve the performance of DNN methods such as USAD by introducing non-local information and to transform a univariate time series into a multivariate one. However, while these methods appear to be effective for time series anomaly detection and their performance on contextual anomalies could be important for automating the supervision of large computer systems, it is currently impossible to argue that their larger complexity compared to conventional methods is really necessary to solve this problem as shown in the study proposed in this thesis. It is important to encourage the community to reintegrate all three families of methods into future benchmarks to ensure that the announced progress is achieved. Finally, this thesis highlighted the power of ensemble approaches combining learning-based and more conventional statistical methods and suggest that the use of hybrid approaches should be further explored.

Chapter 3 proposes a new method called UnSupervised Anomaly Detection for multivariate time series (USAD) [2] based on an autoencoder architecture whose learning is inspired by GANs. The intuition behind USAD is that the adversarial learning of its encoder-decoder architecture allows it to learn to amplify the construction error of inputs containing anomaly, while gaining stability over methods based on GANs architectures. The method has demonstrated superior performance to state-of-the-art techniques on public benchmark datasets. In addition, its learning speed, robustness to parameter selection and stability allow a high scalability of the model in an industrial context. USAD also offers the possibility to parameterize its sensitivity and to produce, from a single model, a set of detection levels.

Chapter 4 investigates a new feature engineering strategy for augmenting time series data in the context of anomaly detection using deep neural networks [100]. The objective is to transform univariate time series into multivariate time series to improve performance. But also, the use of a feature engineering strategy that introduces non-local information in the time series. This chapter therefore proposes a strategy that consists in engineering another time series (i.e. another variable) by extracting non-local information from the raw time series, which deep neural network approaches fail to obtain because they usually operate in a local neighborhood. To this purpose, Matrix Profile (MP), a method that computes the minimum pairwise Euclidean distance of all subsequences of the time series is used, and its result is combined with the original time series. The performance on the KDDcup 2021 competition data containing 250 univariate times shows that introducing non-local information to increase the dimension of the time series improves the performance of the deep neural network method. Although this approach focuses on the special case of transforming univariate time series into multivariate time series, this idea could be used to augment time series, which are multivariate in origin, as a way to introduce non-local information.

Chapter 5 presents an in-depth comparison between conventional methods, machine learning-based approaches and more recent approaches based on deep neural networks. Indeed, the lack of a general comparison covering all families of methods in the literature and the explosion of more and more complex methods based on deep neural networks, has called into question the methodological advances and performance improvements reported in the benchmarks for anomaly detection in multivariate time series. The analysis of the performance of sixteen methods on five datasets did not allow to observe a superiority of one category of methods over the others. The results showed that the three categories can perform better than the other two depending on the dataset criteria. The deep neural network based methods seem to perform better when the dataset contains contextual anomalies or when the datasets are large, while the conventional techniques perform better when the dataset is small.

6.2 Perspectives

This thesis opens several research perspectives.

First, Chapter 5 shows that it would appear that deep neural networks would perform better on contextual anomalies, however, this finding could only be made on one of the five datasets, so further experimentation is needed to confirm that deep neural network methods perform better than the other categories in terms of contextual anomaly detection. It seems then essential to multiply the number of datasets compared in the benchmarks in order to ensure that all eventualities are covered. To do so, the community will have to obtain new real-world datasets containing contextual anomalies but the difficulty for experts to visually label contextual anomalies in multivariate time series makes it difficult to obtain test sets covering this criterion. Obtaining these new sets will potentially allow the community to answer the question: Are deep neural network methods really necessary in the detection of anomalies in multivariate time series?

Second, the results in Chapter 4 suggest that the use of hybrid approaches could be an alternative to the choice between different families of methods. This would require exploring other feature engineering techniques that can provide non-local information, as well as other deep neural network methods for anomaly detection. Thus, the creation of hybrid approaches could potentially address the limitations and constraints of each family of methods in order to get closer to the automatic supervision of large computer systems.

Third, in order to limit the complexity of the models, low complexity model ensembling could be considered. Indeed, the very recent results of the KDDCUP2021 on anomaly detection in univariate time series show that the podium of solutions are based on a low complexity model ensembling with selection mechanisms based on anomaly scores. Thus, the approach of model ensembling seems promising and is complementary with a hybrid approach as proposed previously.

Finally, the ability to detect abnormal behavior in a complex system is not the only important task. To be able to use these models in a real industrial environment, it seems important to be able to identify the variables causing the anomaly. Indeed, detecting an abnormal period in a time series is only the first step. After the detection it is essential to be able to identify the causes in order to apply the adequate countermeasures. Thus, this work will have to be extended in this direction before taking its full place in the industrial world.

References

[1] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. **Deep autoencoding Gaussian mixture model for unsupervised anomaly detection**. In *6th International Conference on Learning Representations, ICLR 2018*, pages 1–19, Toulon, France, 2018. xi, 3, 4, 25, 26, 44

[2] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. **USAD: UnSupervised Anomaly Detection on Multivariate Time Series**. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 3395–3404, New York, NY, USA, 2020. Association for Computing Machinery. 1, 3, 4, 59, 82

[3] Panayiotis T. Theodossiou. **Predicting Shifts in the Mean of a Multivariate Time Series Process: An Application in Predicting Business Failures**. *Journal of the American Statistical Association*, **88**(422):441–449, 1993. 1

[4] D. C. Kale, D. Gong, Z. Che, Y. Liu, G. Medioni, R. Wetzel, and P. Ross. **An Examination of Multivariate Time Series Hashing with Applications to Health Care**. In *2014 IEEE International Conference on Data Mining*, pages 260–269, 2014. 1

[5] Raghavendra Chalapathy and Sanjay Chawla. **Deep learning for anomaly detection: A survey**, 2019. 1

[6] Nilsonreport.com. **Issue 1164**. *Nilson Report*, 2019. 1

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. **Anomaly Detection: A Survey**. *ACM Comput. Surv.*, **41**, 07 2009. 2

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. **Deep learning**. *nature*, **521**(7553):436–444, 2015. 2, 14

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. **Imagenet classification with deep convolutional neural networks**. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012. 2, 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Delving deep into rectifiers: Surpassing human-level performance on imagenet classification**. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 2, 3

[11] Daehyung Park, Yuuna Hoshi, and Charles C. Kemp. **A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder**. *IEEE Robotics and Automation Letters*, **3**(3):1544–1551, 2018. 2, 4, 23, 44, 59

[12] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. *MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks*, pages 703–716. 09 2019. 2, 4, 26

[13] Ya Su, Rong Liu, Youjian Zhao, Wei Sun, Chenhao Niu, and Dan Pei. **Robust anomaly detection for multivariate time series through stochastic recurrent neural network**. **1485**, pages 2828–2837, 2019. 2, 3, 4, 32, 42, 43, 44, 45

[14] Ankit Narendrakumar Soni. **Feature Extraction Methods for Time Series Functions using Machine Learning**. *International Journal of Innovative Research in Science, Engineering and Technology*, **7**(8):8661–8665, 2018. 2

[15] Valerie Fehst, Huu Chuong La, Tri-Duc Nghiem, Ben E. Mayer, Paul Englert, and Karl-Heinz Fiebig. **Automatic vs. Manual Feature Engineering for Anomaly Detection of Drinking-Water Quality**. In *Proceedings of the Genetic and Evolutionary Computation Conference*

Companion, GECCO '18, page 5–6, New York, NY, USA, 2018. Association for Computing Machinery. 2, 56

[16] Salvatore Carta, Alessandro Sebastian Podda, Diego Reforgiato Reforgiato Recupero, and Roberto Saia. **A Local Feature Engineering Strategy to Improve Network Anomaly Detection**. *Future Internet*, **12**(10):177, 2020. 3, 56

[17] Ronan Collobert and Jason Weston. **A unified architecture for natural language processing: Deep neural networks with multitask learning**. In *Proceedings of the International Conference on Machine learning (ICML)*, pages 160–167, 2008. 3

[18] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. **Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups**. *IEEE Signal processing magazine*, **29**(6):82–97, 2012. 3

[19] Victoria Hodge and Jim Austin. **A survey of outlier detection methodologies**. *Artificial intelligence review*, **22**(2):85–126, 2004. 3

[20] Renjie Wu and Eamonn J. Keogh. **Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress**, 2020. 3, 77

[21] Cheng Fan, Fu Xiao, Yang Zhao, and Jiayuan Wang. **Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data**. *Applied Energy*, **211**:1123 – 1135, 2018. 4, 22, 59

[22] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, and Honglin Qiao. **Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications**. In *Proceedings of the 2018 World Wide Web Conference*, WWW

'18, page 187–196, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. 4, 24

[23] Zeyan Li, Wenxiao Chen, and Dan Pei. **Robust and Unsupervised KPI Anomaly Detection Based on Conditional Variational Autoencoder**. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pages 1–9. IEEE, 2018. 4, 24

[24] F. Lüer, D. Mautz, and C. Böhm. **Anomaly Detection in Time Series using Generative Adversarial Networks**. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 1047–1048, 2019. 4, 26

[25] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. **BeatGAN: Anomalous Rhythm Detection using Adversarially Generated Time Series**. pages 4433–4439, 08 2019. 4, 28

[26] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed. **DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series**. *IEEE Access*, **7**:1991–2005, 2019. 4, 25

[27] Haoran Liang, Lei Song, Jianxing Wang, Lili Guo, Xuzhi Li, and Ji Liang. **Robust unsupervised anomaly detection via multi-time scale DCGANs with forgetting mechanism for industrial multivariate time series**. *Neurocomputing*, 2020. 4, 27, 28

[28] Mohsin Munir, Shoaib Siddiqui, Muhammad Chattha, Andreas Dengel, and Sheraz Ahmed. **FuseAD: Unsupervised Anomaly Detection in Streaming Sensors Data by Fusing Statistical and Deep Learning Models**. *Sensors*, **19**, 05 2019. 4, 29

[29] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. **Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding**. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2018. 4

[30] Nan Ding, Huanbo Gao, Hongyu Bu, Haoxuan Ma, and Huaiwei Si. **Multivariate-Time-Series-Driven Real-time Anomaly Detection Based on Bayesian Network**. *Sensors*, **18**(10):3367, Oct 2018. 4, 29

[31] He Q, Zheng Y, Zhang C, Wang H, and Zhang T. **MTAD-TF: Multivariate Time Series Anomaly Detection Using the Combination of Temporal Pattern and Feature Pattern**. 2020. 4, 30

[32] Douglas M Hawkins. *Identification of outliers*, **11**. Springer, 1980. 11

[33] Varun Chandola, Arindam Banerjee, and Vipin Kumar. **Anomaly detection: A survey**. *ACM computing surveys (CSUR)*, **41**(3):1–58, 2009. 11

[34] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. **A Review on outlier/Anomaly Detection in Time Series Data**. *ACM Computing Surveys (CSUR)*, **54**(3):1–33, 2021. 11, 59

[35] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. **A comparative evaluation of outlier detection algorithms: Experiments and analyses**. *Pattern Recognition*, **74**:406–421, 2018. 11, 18

[36] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. **Statistical and Machine Learning forecasting methods: Concerns and ways forward**. *PloS one*, **13**(3):e0194889, 2018. 14, 68, 69

[37] Leo Breiman et al. **Statistical modeling: The two cultures**. *Statistical science*, **16**(3):199–231, 2001. 14

[38] CYNTHIA A. Lowry and DOUGLAS C. Montgomery. **A review of multivariate control charts**. *IIE Transactions*, **27**(6):800–810, 1995. 15

[39] William H. Woodall and Matoteng M. Ncube. **Multivariate CUSUM Quality-Control Procedures**. *Technometrics*, **27**(3):285–292, 1985. 15

[40] Cynthia A. Lowry, William H. Woodall, Charles W. Champ, and Steven E. Rigdon. **A Multivariate Exponentially Weighted Moving Average Control Chart**. *Technometrics*, **34**(1):46–53, 1992. 15

[41] Xia Pan and Jeffrey Jarrett. **Using vector autoregressive residuals to monitor multivariate processes in the presence of serial correlation**. *International Journal of Production Economics*, **106**(1):204–216, 2007. 16

[42] Igor Melnyk, Bryan Matthews, Hamed Valizadegan, Arindam Banerjee, and Nikunj Oza. **Vector Autoregressive Model-Based Anomaly Detection in Aviation Systems**. *Journal of Aerospace Information Systems*, **13**:1–13, 03 2016. 16

[43] Victor M. Panaretos and Shahin Tavakoli. **Cramér–Karhunen–Loève representation and harmonic principal component analysis of functional time series**. *Stochastic Processes and their Applications*, **123**(7):2779–2807, 2013. A Special Issue on the Occasion of the 2013 International Year of Statistics. 16

[44] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. **A Novel Anomaly Detection Scheme Based on Principal Component Classifier**. 2003. 16

[45] J. Edward Jackson and Govind S. Mudholkar. **Control Procedures for Residuals Associated with Principal Component Analysis**. *Technometrics*, **21**(3):341–349, 1979. 17

[46] J. Jackson and Govind Mudholkar. **Control Procedures for Residuals Associated With Principal Component Analysis**. *Technometrics*, **21**:341–349, 08 1979. 17

[47] Qi Dong, Zekun Yang, Yu Chen, Xiaohua Li, and Kai Zeng. **Exploration of Singular Spectrum Analysis for Online Anomaly Detection in CRNs**. *ICST Transactions on Security and Safety*, **4**:153516, 12 2017. 17

[48] Nina Golyandina and Anatoly Zhigljavsky. *Singular Spectrum Analysis for Time Series*. 01 2013. 17

[49] L. Zonglin, H. Guangmin, and Y. Xingmiao. **Multi-dimensional traffic anomaly detection based on ICA**. In *2009 IEEE Symposium on Computers and Communications*, pages 333–336, 2009. 17

[50] Roberto Baragona and Francesco Battaglia. **Outliers Detection in Multivariate Time Series by Independent Component Analysis**. *Neural Comput.*, **19**(7):1962–1984, July 2007. 17

[51] Md Reza. **Multivariate Outlier Detection Using Independent Component Analysis**. *Science Journal of Applied Mathematics and Statistics*, **3**:171, 01 2015. 17

[52] Betty Xia. **Similarity search in time series data sets**. 1997. 17

[53] C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. **Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets**. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1317–1322, 2016. 17, 57

[54] Chin-Chia Michael Yeh, Nickolas Kavantzas, and Eamonn Keogh. **Matrix profile vi: meaningful multidimensional motif discovery**. In *2017 IEEE international conference on data mining (ICDM)*, pages 565–574. IEEE, 2017. 17, 57

[55] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn J. Keogh. **Matrix profile goes MAD: variable-length motif and discord discovery in data series**. *Data Mining and Knowledge Discovery*, **34**:1022–1071, 2020. 18, 58

[56] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. **Isolation Forest**. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008*, pages 413–422, Pisa, Italy, 2008. 19, 44

[57] Y. Qin and Y. Lou. **Hydrological Time Series Anomaly Pattern Detection based on Isolation Forest**. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1706–1710, 2019. 19

[58] Leo Breiman. **Random forests**. *Machine learning*, **45**(1):5–32, 2001. 19

[59] Markus Breunig, Hans-Peter Kriegel, Raymond Ng, and Joerg Sander. **LOF: Identifying Density-Based Local Outliers. 29**, pages 93–104, 06 2000. 20

[60] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. **Efficient Algorithms for Mining Outliers from Large Data Sets. 29**, pages 427–438, 06 2000. 20

[61] Stefan Oehmcke, Oliver Zielinski, and Oliver Kramer. **Event Detection in Marine Time Series Data**. In Steffen Hölldobler, Rafael Peñaloza, and Sebastian Rudolph, editors, *KI 2015: Advances in Artificial Intelligence*, pages 279–286, Cham, 2015. Springer International Publishing. 20

[62] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. **A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise**. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996. 20

[63] Haiwen Chen, Guang Yu, Fang Liu, Zhiping Cai, Anfeng Liu, Shuhui Chen, Hongbin Huang, and Chak Cheang. **Unsupervised Anomaly Detection via DBSCAN for KPIs Jitters in Network Managements**. *Computers, Materials & Continua*, **61**:917–927, 01 2019. 20

[64] Bernhard Schölkopf, John Platt, John Shawe-Taylor, Alexander Smola, and Robert Williamson. **Estimating Support of a High-Dimensional Distribution**. *Neural Computation*, **13**:1443–1471, 07 2001. 21

[65] Junshui Ma and S. Perkins. **Time-series novelty detection using one-class support vector machines**. **3**, pages 1741 – 1745 vol.3, 08 2003. 21

[66] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. 22, 36, 59

[67] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. **Generative Adversarial Nets**. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 2672–2680, Montreal, Quebec, Canada, 2014. 22, 26

[68] Sepp Hochreiter and Jürgen Schmidhuber. **Long Short-Term Memory**. *Neural Computation*, **9**(8):1735–1780, 1997. 23

[69] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts. **Anomaly Detection for Time Series Using VAE-LSTM Hybrid Model**. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4322–4326, 2020. 23

[70] Jinwon An and Sungzoon Cho. **Variational autoencoder based anomaly detection using reconstruction probability**. *Special Lecture on IE*, **2**(1):1–18, 2015. 24

[71] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. **Object recognition with gradient-based learning**. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999. 25, 27

[72] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. **Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery**, 2017. 26

[73] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. **A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data**. *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**:1409–1416, 2019. 27

[74] H. Zare Moayedi and M. A. Masnadi-Shirazi. **Arima model for network traffic prediction and anomaly detection**. In *2008 International Symposium on Information Technology*, **4**, pages 1–6, 2008. 29

[75] Jeff Hawkins and Sandra Blakeslee. *On intelligence*. Macmillan, 2004. 29

[76] Nir Friedman, Dan Geiger, and Moises Goldszmidt. **Bayesian network classifiers**. *Machine learning*, **29**(2):131–163, 1997. 29

[77] Finn V Jensen and Frank Jensen. **Optimal junction trees**. In *Uncertainty Proceedings 1994*, pages 360–366. Elsevier, 1994. 30

[78] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. **Graph Attention Networks**, 2018. 30

[79] Houssam Zenati, Manon Romain, Chuan-Sheng Foo, Bruno Lecouat, and Vijay Chandrasekhar. **Adversarially Learned Anomaly Detection**. In *IEEE International Conference on Data Mining, ICDM 2018*, pages 727–736, Singapore, 2018. 31

[80] Haowen Xu, Yang Feng, Jie Chen, Zhaogang Wang, Honglin Qiao, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, and et al. **Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications**. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, 2018. 32

[81] Martín Arjovsky and Léon Bottou. **Towards Principled Methods for Training Generative Adversarial Networks**. In *5th International Conference on Learning Representations, ICLR 2017*, Toulon, France, 2017. OpenReview.net. 37

[82] Martin Arjovsky, Soumith Chintala, and Léon Bottou. **Wasserstein GAN**, 2017. 37

[83] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. **A Dataset to Support Research in the Design of Secure Water Treatment Systems**. In *Critical Information Infrastructures Security*, pages 88–99, 2017. 42

[84] Aditya P. Mathur and Nils Ole Tippenhauer. **SWaT: a water treatment testbed for research and training on ICS security**. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, pages 31–36, 2016. 42

[85] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Söderström. **Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding**. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 387–395, 2018. 42

[86] Zhiyou Ouyang, Xiaokui Sun, and Dong Yue. **Hierarchical time series feature extraction for power consumption anomaly detection**. In *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration*, pages 267–275. Springer, 2017. 57

[87] Mrutyunjaya Panda, Abd Allah A. Mousa, and Aboul Ella Hassanien. **Developing an Efficient Feature Engineering and Machine Learning Model for Detecting IoT-Botnet Cyber Attacks**. *IEEE Access*, **9**:91038–91052, 2021. 57

[88] Cheng Fan, Yongjun Sun, Yang Zhao, Mengjie Song, and Jiayuan Wang. **Deep learning-based feature engineering methods for improved building energy prediction**. *Applied Energy*, **240**:35–45, 2019. 57

[89] André A. P. Santos, Francisco J. Nogales, and Esther Ruiz. **Comparing Univariate and Multivariate Models to Forecast Portfolio Value-at-Risk**. *Journal of Financial Econometrics*, **11**(2):400–441, 10 2012. 57

[90] Patrick Aboagye-Sarfo, Qun Mai, Frank M. Sanfilippo, David B. Preen, Louise M. Stewart, and Daniel M. Fatovich. **A comparison of multivariate and univariate time series approaches to modelling and forecasting emergency department demand in Western Australia**. *Journal of Biomedical Informatics*, **57**:62–73, 2015. 57

[91] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. **The M4 Competition: 100,000 time series and 61 forecasting methods**. *International Journal of Forecasting*, **36**(1):54–74, 2020. M4 Competition. 65, 69

[92] Vegard Antun, Francesco Renna, Clarice Poon, Ben Adcock, and Anders C Hansen. **On instabilities of deep learning in image reconstruction and the potential costs of AI**. *Proceedings of the National Academy of Sciences*, **117**(48):30088–30095, 2020. 68, 69

[93] Douglas Heaven. **Why deep-learning AIs are so easy to fool**. *Nature*, **574**(7777):163–166, 2019. 68, 69

[94] Shuming Jiao, Yang Gao, Jun Feng, Ting Lei, and Xiaocong Yuan. **Does deep learning always outperform simple linear regression in optical imaging?** *Optics express*, **28**(3):3717–3731, 2020. 68, 69

[95] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. **Do we need hundreds of classifiers to solve real world classification problems?** *The journal of machine learning research*, **15**(1):3133–3181, 2014. 68

[96] Weilin Fu, Katharina Breininger, Roman Schaffert, Zhaoya Pan, and Andreas Maier. **"Keep it simple, scholar": an experimental analysis of few-parameter segmentation networks for retinal vessels in fundus imaging**. *International Journal of Computer Assisted Radiology and Surgery*, pages 1–12, 2021. 69

[97] Spyros Makridakis, Evangelos Spiliotis, and Vassilis Assimakopoulos. **The M4 Competition: Results, findings, conclusion and way forward**. *International Journal of Forecasting*, **34**, 06 2018. 69

[98] Takaya Saito and Marc Rehmsmeier. **The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets**. *PloS one*, **10**:e0118432, 03 2015. 71

[99] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. **Matrix profile XI: SCRIMP++: time**

series motif discovery at interactive speeds. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 837–846. IEEE, 2018. 75

[100] Julien Audibert, Frédéric Michiardi, Sébastien Marti, and Maria A. Zuluaga. **From Univariate to Multivariate Time Series Anomaly Detection with Non-Local Information**. In *Proceedings of the 6th Workshop on Advanced Analytics and Learning on Temporal Data*, ECML-PKDD '21, 2021. 82

Appendices

A.1 Material for USAD Reproducibility

Experimental Setting

All experiments are performed on a machine equipped with an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz and 270 GB RAM, in a docker container running CentOS 7 version 3.10.0 with access to an NVIDIA GeForce GTX 1080 Ti 11GBGDDR5X GPU. The Isolation Forest (IF) comes from the scikit-learn [1] implementation. The DAGMM comes from a Tensorflow implementation on Github [2]. The LSTM-VAE comes from a Github implementation [3]. The OmniAnomaly comes from the authors' Tensorflow implementation of Github [4]. Finally, the USAD and AE were developed by me in Pytorch.

Packages Used in the Implementation

The relevant packages and their versions used in the algorithm implementation are listed as follows:

- python==3.6.8
- pytorch==1.3.1
- cuda==10.0
- scikit-learn==0.20.2

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html`

[2] `https://github.com/tnakae/DAGMM`

[3] `https://github.com/Danyleb/Variational-Lstm-Autoencoder`

[4] `https://github.com/NetManAIOps/OmniAnomaly`

- numpy==1.15.4

USAD Hyper-parameters for each dataset

For each dataset we have 4 parameters. The size of the windows, corresponding to the size of the sequence of time series we have in input. The number of epochs, the dimension of Z which is the USAD latent space and finally the down-sampling rate during pre-processing. The down-sampling is done by taking the median value of each feature.

Table 1: USAD Hyper-parameters for each dataset. K denotes the window size and m the dimension of the latent space.

Datasets	K	Epochs	m	Down-sampling
SWat	12	70	40	5
WADI	10	70	100	5
SMD	5	250	38	5
SMAP	5	250	55	5
MSL	5	250	33	5

USAD Implementation

The input size corresponds to the size of the window multiplied by the number of dimensions of the multivariate time series.

Encoder

- Linear : input size -> input size / 2
- Relu
- Linear : input size /2 -> input size / 4
- Relu
- Linear : input size /4 -> latent space size
- Relu

Decoder

Both decoders have the same architecture.

- Linear : latent space size -> input size / 4
- Relu
- Linear : input size /4 -> input size / 2
- Relu
- Linear : input size /4 -> input size
- Sigmoid

As optimizer we use Adam's pytorch implementation with his default learning rate.

A.2 Material for Benchmark Reproducibility

Experimental Setting

All experiments are performed on a machine equipped with an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz and 270 GB RAM, in a docker container running CentOS 7 version 3.10.0 with access to an NVIDIA GeForce GTX 1080 Ti 11GB GPU.

Implementation

Methods	Implementation
VAR	https://www.statsmodels.org/stable/index.html
MCUSUM	https://github.com/Marco-Christiani/MITTEN
MEWMA	https://github.com/Marco-Christiani/MITTEN
MP	https://stumpy.readthedocs.io
PCA	https://github.com/yzhao062/pyod
SSA	https://github.com/kieferk/pymssa
ICA	Implementation based on scikit-learn
IF	https://github.com/yzhao062/pyod
LOF	https://scikit-learn.org
OC-SVM	https://scikit-learn.org
DBSCAN	https://scikit-learn.org
AE	Implementation based on Pytorch
LSTM-VAE	https://github.com/TimyadNyda/Variational-Lstm-Autoencoder
DAGMM	https://github.com/tnakae/DAGMM
OmniAnomaly	https://github.com/NetManAIOps/OmniAnomaly
USAD	https://github.com/robustml-eurecom/usad

Parameters

All unspecified parameters were used by default in the implementation.

Methods	Parameters
VAR	$maxlags = 100$
MCUSUM	$k = 0.5$
MEWMA	-
MP	$window\_size = 100, discords = True$
PCA	$n\_components = 20$
SSA	$n\_components = 20, novelty = True$
ICA	$n\_components = 20$
IF	-
LOF	$novelty = True$
OC-SVM	-
DBSCAN	$eps = 0.15, min\_samples = 25$
AE	$window\_size = 12, latent\_dimension = 40, Epochs = 70$
LSTM-VAE	$window\_size = 12$
DAGMM	-
OmniAnomaly	$window\_size = 12$
USAD	$window\_size = 12, Epochs = 70$