POLITECNICO DI TORINO

Master of Science course in
Data Science and Engineering

Master's Degree thesis

Deep Anomaly Detection: an experimental comparison of deep learning algorithms for anomaly detection in time series data

Supervisors
Prof. D. APILETTI
Dott. S. MONACO

Candidate
ANTONIO ALBANESE

April 2023

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the last decades, thanks to the development of information technology, data have proliferated doubling every year since the 1980s [1]. One of the biggest sectors that produce huge amounts of data is Industry, in which the advent of IoT (Internet of Things) enabled a new paradigm based on the automation of processes like data collection or monitoring, indeed researchers need to analyze an ever-increasing amount of data and doing so without the help of automated approaches has become infeasible [2]. Cyber-physical systems employed in industry 4.0 or monitoring systems used in medicine, aircraft, or servers usually record such data in the form of Multivariate Time Series. The huge amount of time-related information that is continuously recorded allows us to analyze such data not only to make predictions but also to monitor systems and eventually identify deviations from the normal behavior [3], in other terms having a big quantity of time-related information enable us to look for anomalies in data that may indicate abnormal and dangerous events in the monitored systems. The problem of finding such deviations in data is called **Anomaly Detection** which is the main topic of this work. Below in this chapter, we will describe the anomaly detection problem along with the time series data and the opportunities that deep learning offers in anomaly detection applied to Multivariate Time Series data.

1.1 Anomaly Detection

Anomaly detection problem refers to finding data instances that do not conform to the expected and general behavior of data, the non-conforming patterns or points found are usually called anomalies or outliers [4]. Sometimes more specific and application-driven terms, which can better describe the meaning and implications of an outlier in its field, may be used instead of the term anomaly. In fact, as we will see later, the definition of anomaly is strictly dependent on the application domain, so that in literature we can find references to terms like anomalies, outliers, exceptions, discordant observations, aberrations, surprises, peculiarities, contaminants, and so on [4].

Therefore in literature we can read of common and important challenges of defining anomalies, as reported by Chandola et al. [4] the simple and abstract definition of anomalies as instances that heavily deviate from the general data trend, leads us to several challenges when searching for a more precise and technical definition of the anomaly:

- It is quite difficult to define a zone of data that includes every conceivable normal activity. Additionally, the boundary between regular activity and anomalous behavior it's not always clear. As a result, an observation near the boundary may appear abnormal when it is normal and vice versa.
- Normal behavior may be dynamic and evolving over time, so it is impossible to represent all future normal trends

- The anomaly definition is different for different domains. For example, a small deviation from the general trend may be seen as normal in stock markets data, while the same small deviation in medical data may be a sign of a disease

- High-quality labeling for data requires time and domain experts making it difficult and costly to obtain labels that are reliable enough to train a classical classification model.

Due to these problems, it is not easy to formulate and solve a general problem for anomaly detection. Most existing anomaly detection techniques are able to (and tuned to) solve only field-specific formulations of the problem. That specific formulation is every time induced by various factors such as the nature of the data, the availability or not of the labels, the type of anomalies to be detected, and so on.

One of the major issues in anomaly detection is to define which type of anomaly we are in search of. In fact, anomalies can appear in different forms which can be categorized into 3 types [2, 4, 5, 6, 7]:

- **Point anomalies:** it is the simplest type of anomaly. These anomalies occur when a single point or instance of data significantly deviates from the rest of the data. For example in Figure (1.1a) point p is a point anomaly compared to normal points in the outlined region.

- **Contextual anomalies:** If a data instance is anomalous in a specific context, but not otherwise, then it is termed a contextual anomaly [8]. Figure (1.1b) illustrates a contextual anomaly in temperature data, where t_1 and t_2 have the same allowed value, but t_1 is a normal point since it is registered in the winter period, whereas t_2 is not normal because it is registered before June.

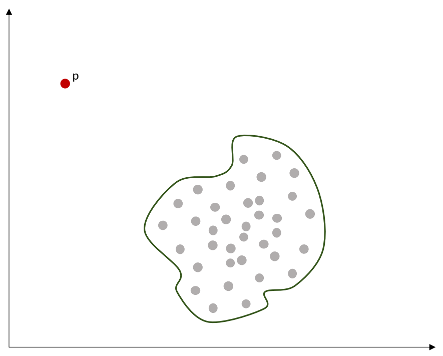
- **Collective anomalies:** occur when a contiguous series of data points is anomalous with respect to the general behavior of data. For example in Figure (1.1c), which reports the graph of an ECG measurement, the highlighted region corresponds to an anomaly which indicates an Atrial Premature Contraction in the patient, in fact, the signal persists for an abnormally long time at the same value, while the value itself is not an anomalous value.

Another major challenge in anomaly detection is the rarity of anomalies in data. Considering information collected from real-world applications, anomalies may correspond to dangerous or catastrophic events, so it is obvious that anomalies are very rare and roughly the entire data contains normal samples. In such a scenario it is difficult to have enough records to train a proper classical and supervised classification model, or even to perform validation and testing in a proper and rigorous way. This fact points us also to the problem of choosing a good metric to fairly evaluate models. We will better address these aspects in the next chapters.

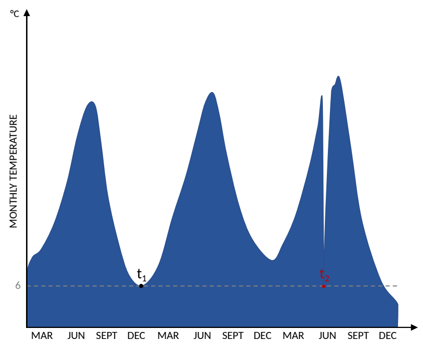
1.2 Time Series Data

Time-related data are widely present in databases of various fields. Time-series data are ordered collections of information obtained through repeated measurements over time, indeed the ordering is assumed to be through the time associated with the measurement.

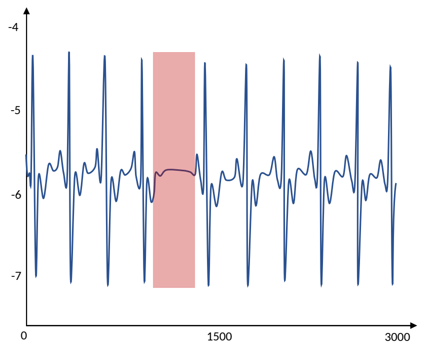
Time series can occur in various fields such as production rates and prices from industry or daily temperature and wind speed in meteorology, etc. The intrinsic nature of this type of data is that observations are strictly dependent and correlated with previous ones.



(a) Point anomaly: in a 2-dimensional space, the point p is an anomaly with respect to other points that are all-encompassed with the green line



(b) Contextual anomaly: point t_2 is a contextual anomaly because even though its value is allowed and already present in the series (t_1), it heavily deviates from the normal behavior of near measurements [4, 5]



(c) Collective anomalies: points in the highlighted region do not follow the normal behavior of the series, and having all the points at that signal level indicates the Atrial Premature Contraction disease [4]

Figure 1.1: Different types of anomalies

Time-based collections of data that report values of a single variable are generally referred to as Univariate Time Series, on the other hand when several related variables are observed simultaneously we talk about Multivariate processes. Multivariate time series data is defined as data taken from multiple sources at the same time and where the output of each source depends on what is happening in the other sources. A very common example of multivariate time series data is the EEG-brain data, in which measurements are collected from different sensors which usually are called channels, an example is reported in Figure(1.2)

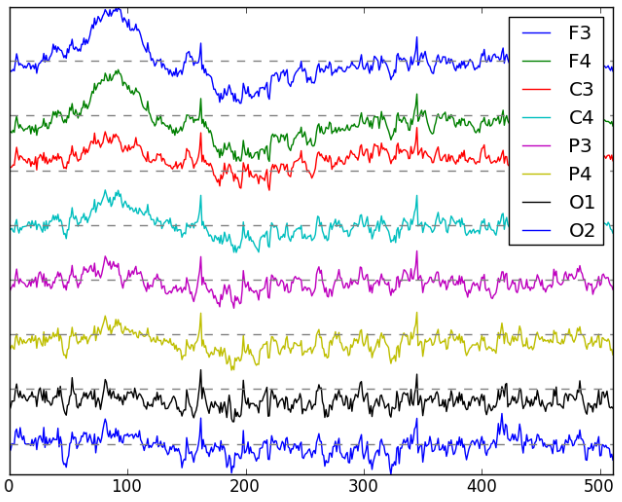


Figure 1.2: EEG signal registered over time in different channels [9]

We can use a simple example to understand the complexity and the power of time series data. Considering information about the daily inventory level of ice cream and waffle cones in a local grocery store, an idea for having always healthy inventory levels would be to use a time series analysis model fitted on data from past years' inventory to predict the demand level. Of course, doing so, we would immediately notice that, besides the general trend of the two variables, also a seasonality would emerge with fewer sales in the cold periods and a strong increment during hot periods. Moreover, we would see how the demands of the two products are highly cross-correlated. Now we could also add another variable to the series, which may be beverage demand. In that case, trends, seasonality, and in particular cross-correlations with other variables, may be very underlying and difficult to spot without a proper analysis framework.

Major time series-related tasks are forecasting, classification, and anomaly detection. Time series analysis during the last decades has gained more and more attention, and although a solid theoretical foundation underlies many time-series analysis methods proposed by researchers, their great number and interdisciplinary diversity make it very difficult to determine how methods developed in different disciplines do relate to one another [10].

For this reason, when approaching time series tasks, it is important to identify the best framework in which to operate, so that we can capture all the possible information that is contained in the series and obtain results that are as consistent as possible with reality.

1.3 Types of classification models

Since the Anomaly Detection task can be brought back to Classification we can introduce a general categorization of models, based on the labeling of data we use for training. Usually, Deep Learning algorithms are classified as: **supervised**, **semi-supervised**, and **unsupervised** methods.

- **Supervised** methods need data to be accurately labeled before training the model. In Anomaly Detection, as already explained this is a problem, since labeling data requires strong domain knowledge and can be really expensive and time-consuming. Moreover, the labeling is usually performed manually by domain experts, so we can never be certain that it is error-free.

- **Unsupervised** do not need data to be labeled and can learn distributions and associations used to classify input without using any labels. For this reason, supervised models are considered more immediate and of simple usage in tasks like Anomaly Detection. In fact, even though some effort is needed to setup the best configuration of the model, it can be more convenient because we can avoid the labeling phase.

- **Semi-supervised** methods are usually in the middle between supervised and unsupervised. A combination of labeled and unlabeled data is usually used to train these methods, with the labeled sets generally smaller than the unlabeled ones. In two-class classification tasks also models which assume that the training set contains data concerning only one of the two classes are considered semi-supervised models. For example, Generative Adversarial Networks for Anomaly Detection are most of the time based on the assumption that the training set is devoid of anomalies, containing only normal records.

1.4 Deep Anomaly Detection

As already said, we know that the problem of Anomaly detection, and more in general time-series analysis, is strictly dependent on the data domain. Research in anomaly detection has been active for many years, but as already explained anomaly detection in Multivariate Time series data points out some challenges that are difficult to overcome with classical statistical analysis frameworks and methods [6]. In the last years, the advent of deep learning enabled new approaches to Anomaly detection. In particular, the use of Deep Neural Networks allowed researchers to work with a large amount of data in a more automatic way. In fact, Deep Neural Networks are designed to identify hidden connections between variables without the need for prior knowledge, unlike classical methods that rely on domain experts to manually examine data for significant patterns or points. It is obvious that the classical approach has a high probability to be costly, difficult, and yielding sub-optimal results when applied to large datasets.

All the best state-of-the-art Deep Anomaly Detection methods which are based on different types of Neural Networks use a specific, and most of the time, unedited way to calculate a score, which is often referred to as **Anomaly Score** on which a threshold must be defined to find anomaly points and normal ones. The advantage given by Deep Learning can give some difficulties in comparing results obtained from different models. In fact, typically, each deep learning model generates its own unique Anomaly Score calculation methodology, so we cannot simply compare scores returned by different methods, but some deeper analysis of results is required in order to understand which model performs better than others.

Chapter 2

Related Works

The topic of Deep Anomaly detection in time series data has started its major development in the last ten years. During this time we can find a lot of methods that rapidly outperform previous ones. In this chapter, we will first describe some basic concepts and the Deep Neural Network architectures which are at the basis of state-of-the-art Deep Anomaly Detection models and lastly, we will provide a description of some of the most cited surveys about this topic clarifying the motivations and needs behind this work.

2.1 State-of-the-art methods

In our exploration of the Deep Anomaly Detection for Time-Series topic, we decided to proceed with dividing explored methods based on the architecture used to implement the deep neural network. In fact, one of the advantages of deep learning is that different networks are able to learn and model the time dependence and cross-correlations between variables in different ways, which can be more suitable for different use cases. In this work, we will experiment with architecture based on Convolutional Neural Networks (CNN), Generative Adversarial Networks (GAN), Transformers, and Auto-Encoders (AE), which, to the best of our knowledge, are four of the most used, newest, and most promising architectures in this research area.

2.1.1 CNNs in Deep Anomaly Detection

Convolutional Neural Network (CNN) is a well-known deep learning architecture that is widely used in computer vision, and in general, is really powerful in image-related tasks. CNNs are generally built on 3 types of layers: convolutional layers, pooling layers, and fully-connected layers. In particular, the convolutional layer is in charge of determining the output of neurons which correspond to local regions of the input through the calculation of the scalar product between their weights and the input region [11]. After big development in computer vision, CNNs have been explored also with time series data. In this case convolutional and pooling layers are used to extract deep features of the raw data which are consequently used to perform the original task which in our case is Anomaly Detection [12, 13]. In the last years CNNs have been used alone and in conjunction with other architectures in many different approaches for time series anomaly detection [5, 6, 14, 12, 15, 16]. Ren et al in [17] at Microsoft propose a model based on Spectral Residual (SR) and CNNs. Their work is the first that tries to borrow the SR model from the visual saliency detection domain to time-series anomaly detection, moreover, they combine SR with CNN improving the SR's performances. Munir et al. in [12] propose a model made of two CNN-based modules, which they call *predictor* and *anomaly detector*, the first one is in charge of learning the data distribution and predicting new data, the second one is in charge of learning the data distribution and find anomalies comparing the predicted time series windows with real ones.

2.1.2 GANs in Deep Anomaly Detection

Generative Adversarial Networks are deep learning models that gained great interest in the last years. GANs are generative models, generally composed of two modules named discriminator and generator, which goal is to learn the probability distribution that generates the training samples and then use this knowledge to generate more samples from the estimated probability distribution [18]. GANs have been proposed in different publications, which preserve the basic process of adversarial training while proposing novelties in the anomaly score evaluation and base architectures. Li et al. in MAD-GAN [19] use a Generative adversarial network in which the generator and the discriminator are implemented as Long-Short Term Memory - Recurrent Neural Networks (LSTM-RNN) with the aim of capturing the latent interactions amongst all the variables in the Multivariate Time Series, in this way they could exploit both Discriminator and Generator to compute an unedited anomaly score named DR-score. Geiger et al. in TadGAN [16] introduce cycle-consistent GAN architectures for time series data, such that Generators can be directly used for time series reconstructions and then use the reconstruction error to compute the anomaly score. Bashar et al. in TAnoGAN [20] implement both discriminator and Generator as LSTM-RNN with the goal of using the learned feature representation to compute the anomaly score, the proposed architecture is shown to be performing really well in particular on small dimensions datasets, but also on the conventional benchmark datasets used in other studies.

2.1.3 AE in Deep Anomaly Detection

Auto-Encoders are neural networks that are designed to encode the input data into a valuable and meaningful compressed representation and then decode it back such that the reconstructed data is as similar as possible to the original input data [21]. In Deep Anomaly Detection AEs are used to learn only the most significant features of a training set that will serve as the reference of normality [22]. For example, Chen et al. in [23] proposed an AE-based model that is capable of realizing real-time anomaly detection on multivariate series data. Que et al. in [24] propose a two-stage approach based on a fine-tuned AE and a stacked LSTM which uses the AE-learned features to predict aircraft time series and to detect anomalies. Audibert et al in [25] proposed a method formulated as an AE architecture that is trained in an adversarial fashion with the aim of making the model capable of identifying normal input data while also performing a good reconstruction.

2.1.4 Transformers

Transformers have shown great power in sequential data, even though these architectures have a very short history in the time-series domain. Recent research has demonstrated that Transformers can be highly advantageous in time-series analysis. This is due to the self-attention mechanism incorporated in their architecture, which enables them to identify significant long-term temporal relationships, thereby increasing the reliability of their findings. Chen et al. in [26] propose a novel framework for Multivariate Time Series Anomaly Detection which is capable of learning a graph structure while modeling temporal dependency using a transformer-based architecture. Xu et al. in [27] overcome the limitation of Transformers in anomaly detection by renovating the self-attention mechanism, based on a mechanism that they named association discrepancy.

2.2 Anomaly detection surveys and benchmarks

The variety of proposed models about this topic leads to a prolific literature of surveys that report and categorize every time the state-of-the-art methods for anomaly detection and also give some research directions for improving the research capabilities.

During the last decade, a lot of surveys have been published, each one focusing on different aspects of this complicated field of research. Unfortunately, a small number of these surveys present an experimental approach, most of them can be categorized as literature reviews, which of course give an important contribution to understanding the Deep Anomaly Detection subject from different perspectives of analysis, moreover, an even smaller number of them are completely focused on Deep Learning techniques, and compare Deep Neural Networks models with old-approach statistical models.

In 2016, Wu et al. [1] proposed a complete overview of Anomaly detection for time series, in particular categorizing methods based on the technique used to calculate the anomaly score, finding 5 categories: Anomaly Detection Based on Statistical, Anomaly Detection Based on Clustering, Anomaly Detection Based on Deviation, Anomaly detection based on distance [6, 28] and Anomaly Detection based on Density [6, 29]. With their work authors focused on understanding how the Anomaly Score was implemented in each method and how different strategies of calculation can lead to different findings but also different issues.

In 2019 Chalapathy et al. present a comprehensive review of state-of-the-art research in Deep Anomaly Detection techniques. In their work authors tried to understand and describe correlations between the application domain and the performances of the methods reviewed. They also expressed concerns regarding the data pipeline and the computational resources that are necessary for each method. [5] .

One of the most complete surveys about Deep Anomaly Detection is the one by Choi et al., [6] published only one year ago in 2021, demonstrating how this field of research is in active and fast development. In this survey authors comparatively, analyze state-of-the-art deep-anomaly-detection models for time series with several benchmark datasets, but they use an unclear hybrid approach to build their comparison. Their setup starts from the selection of benchmark datasets and after they proceed with the comparison of different methods, directly reporting original results if available for the selected time series collections in conjunction with the dataset, or re-implementing the methods if the dataset was not considered by the model authors.

In 2022 Han et al. made the most extensive and structured experimental review on Anomaly Detection [30]. Their comprehensive experimentation uncovers important insights into the function of supervision and anomaly types and opens new avenues for algorithm design and selection study. The authors concentrate on three crucial comparison angles: the presence of supervision, the types of anomalies, and the robustness of the algorithm to noise and data manipulation. Anyway, this article is not focused in particular on time series data. In collecting methods to test authors did not concentrate their attention on Deep Neural Networks only, but considered also classical statistical models. Moreover, the majority of the unsupervised methods explored are old considering the big development of this field in the last 5-10 years, and even the two recent unsupervised models reported have not a Deep Learning architecture.

2.2.1 Present work contribution

This study was issued to meet specific needs, that would like to overcome the limitations of previously published reviews. Consider for example, that we want to build a vehicle health monitoring system as part of a predictive maintenance project. The starting point for such an implementation could be to build a system for anomaly detection, in fact, anomalies in the data collected on a vehicle could be indicative of impending failures on that vehicle. In such a situation, the main requirement is to be able to figure out which of the state-of-the-art available methods performs best with the data available.

Indeed, our contribution to this research topic can be resumed as follows:

- we provide a first step for a new approach in cross-field experimental comparison of the state-of-the-art models, which focuses on how the time modeling expressed by the dataset is captured by models.
- we propose a new comparison strategy that starts by categorizing tested methods based on the architecture on which they are built.
- we built and released a simple and effective framework that can be used to implement and compare different methods and chose the best for the application domain and data available.
- finally we propose a novel score, named **Discrimination Score**, that can be used to compare different methods based on the Anomaly Score of normal and anomalous points.

The research project described in this dissertation began by choosing the methods to be evaluated, which were initially grouped based on the type of neural network architecture used. This was done because previous research has demonstrated how the diverse abilities of each neural network in capturing temporal dependencies can impact the model's effectiveness [6]. Once the architectures to be analyzed were chosen, the state-of-the-art methods were selected based on the criteria shown in Chapter (3), then as described in the same chapter, six datasets were chosen to be used as benchmarks to evaluate proposed models.

Chapter (4) shows how the obtained implementation has been built to be extended and used with different datasets and also different Deep Anomaly Detection models so that we can see it as an interface utility to manage the performance comparison of different Deep Anomaly Detection models. Then, we report and comment on the results obtained in our experimentations and explain the reasoning behind the proposed novel metric **Discrimination Score**. Finally, Chapter (5 draws conclusions and proposes future direction to extend this work.

Chapter 3

Methods

In this experimental survey, we decided to compare the state-of-the-art methods for each architecture that was previously reported. In this chapter, we will see details of each selected method and details of each data set used in the experimental evaluation.

3.1 Research criteria

As already mentioned the evolution and advance of research in Deep Anomaly Detection led to a huge quantity of proposed models. We tried to select architectures that seem to be best performing and mainly most promising. To make a good selection in this panorama we first need to define rigorously the problem we want to address. Our research is about Deep Anomaly Detection, which has been developed in various fields and working areas. We want to explore different methods and evaluate them in a cross-field fashion, such that we could assess two derivable at the end of this work:

- Answer the question: Is it possible to identify a general best-performing model for Deep Anomaly detection in today's landscape?

- obtain public available systematic pipeline which can be used to evaluate different Deep Anomaly Detection Models

With this objective in mind, we fixed some criteria on which we based the model selection:

- **Multivariate Time Series capability or extensibility**: the major contribution of deep learning to the problem of anomaly detection is the capability to learn cross-correlations between different features of the series. For this reason, we selected methods that were built to work on Multivariate Time Series or methods for which it was possible to add Multivariate Time Series support without changing the basic structure built by the authors.

- **recently published**: Deep learning in general is in continuous evolution. We considered of limited utility to work on methods that were published before 2019, and for the same reason, as said in the previous chapter (2) we identified 4 main architecture trends, in Deep Anomaly Detection and Deep Learning in general, during the last three years which are CNNs, GANs, Transformers, and AEs.

- **state-of-the-art in its class of models**: In the 4 mentioned classes of architectures we chose methods that to the best of our knowledge are the best performing but also methods that looked most promising in terms of intuitions and novelties.

- **unsupervised method**: as described in Chapter (1) we can categorize

Anomaly detection models in the three major types of *supervised*, *semi-supervised* and *unsupervised* methods. Since we want to work on Deep Learning Algorithms and one of the greatest advantages of Deep Neural Networks is that they can be used even with limited knowledge about data, and since the difficulty of labeling data is one of the major issues in Anomaly Detection, we decide to test only **unsupervised** methods which allow us to overcome these limitations. We will also consider **semi-supervised** methods like GANs, but using them in an unsupervised fashion based on findings of previous works [16, 19, 20] in which emerged that the rarity of anomalies in data makes semi-supervised methods being used in an unsupervised way with promising results.

We decided to explore Deep Learning models built for Anomaly Detection tasks on Time Series data, but it should be noted also that works such [30] count some statistical unsupervised methods in the best performing models for Anomaly Detection. We also need to keep in mind that, even though this is an experimental survey, our main objective was not the study of the implementation of each method, moreover we have to notice that research and implementation of a novel Deep Learning model requires a huge effort in term of work, for this reason, we preferred among others methods that were published with PyTorch implementation or methods with a well-explained implementation such that obtaining a PyTorch implementation capable of replicating published results was not too much difficult.

Based on the presented criteria we decided to experimentally evaluate the following published methods: *DeepAnt* by Munir et al. [12] based on CNN, *TanoGAN* by [20] Bashar et al. which make use of GANs, *USAD* the AE based model proposed by Audibert et al. [25] and *Anomaly Transformer* by Xu et al. built with Transformer Networks [27]. Since this work is an experimental evaluation of the methods reported above, we focused on replicating the published results while modifying them as little as possible. All the methods presented in this work were built, or have been re-implemented, using the PyTorch framework, also the data pre-processing strategy, network architectures, and hyper-parameters values have been kept as much as possible equal to the published ones.

The next section contains a detailed description of selected methods, while a quick summary is reported in Table (3.1).

Model Name	**MTS**	**Pytorch**	**Un/Semi-supervised**
DeepAnt	yes	re-implemented	semi-supervised
TanoGAN	yes	original	unsupervised
USAD	yes	re-implemented	unsupervised
Anomaly Transformer	yes	original	semi-supervised

Table 3.1: Summary of experimented methods, for each, is reported if the method has or not Multivariate Time Series capability, is originally implemented in PyTorch or re-implemented by the author and if it is semi-supervised or unsupervised in its publication

3.2 Selected Methods

3.2.1 DeepAnT

DeepAnT is a Deep Anomaly Detection method based on Convolutional Neural Networks. The architecture uses two CNN modules, the first is named **time series predictor** and predicts the next time window based on the input time window; the second is named **anomaly detector** which is responsible to tag the time series window as **normal** or **anomaly**. DeepAnT is an unsupervised method, it does not need labels at training time, therefore this method is very simple to extend to a real-life scenario.

The time-series predictor architecture: DeepAnT uses a traditional form of CNNs. The time-series predictor's architecture is composed of two convolutional layers each one followed by a max-pooling layer. The input data, which is a time-series window (w), can be represented as a vector to the network. Both the convolutional layers are made of 32 kernels followed by a ReLU activation function. After two pairs of convolutional and max-pooling layers, there is a fully connected layer that outputs the predicted time-series window.

During the training procedure, CNN's parameters (weights and biases) are optimized with the Stochastic Gradient Descent (SGD) paradigm. The loss function to be minimized in this architecture is the Mean Absolute Error (MAE) reported in Equation (3.1). This function is employed as a measure of the discrepancy between the actual time-series window to be predicted (y_i) and the predicted one ($\hat{y}_i$).

$$MAE = \frac{1}{n}\sum_{j=1}^{n} y_i - \hat{y}_i \tag{3.1}$$

The functioning of DeepAnT architecture is reported in Figure (3.1).

Since in this phase we are building a time-series predictor, in order to leverage CNN for this task and considering that this method works in an *Un-Supervised* way, the input data must be set in a compatible form. Therefore each element x_{t+1} at the time stamp $t+1$ is used as a label for the element x_t at the time stamp t. The number of time stamps required to be predicted is referred to as the prediction window (p_w).

Our preliminary studies on this method pointed out that the best p_w is equal to the size of the time-series window (s_w) so that for each time-series window (w) the above-described predictor will learn to predict a time-series of s_w length. As shown in Figure (3.1) considering the window w as input data, the predictor will learn how to make the forecasted window w_{+1} as similar as possible to w'_{+1}. So,

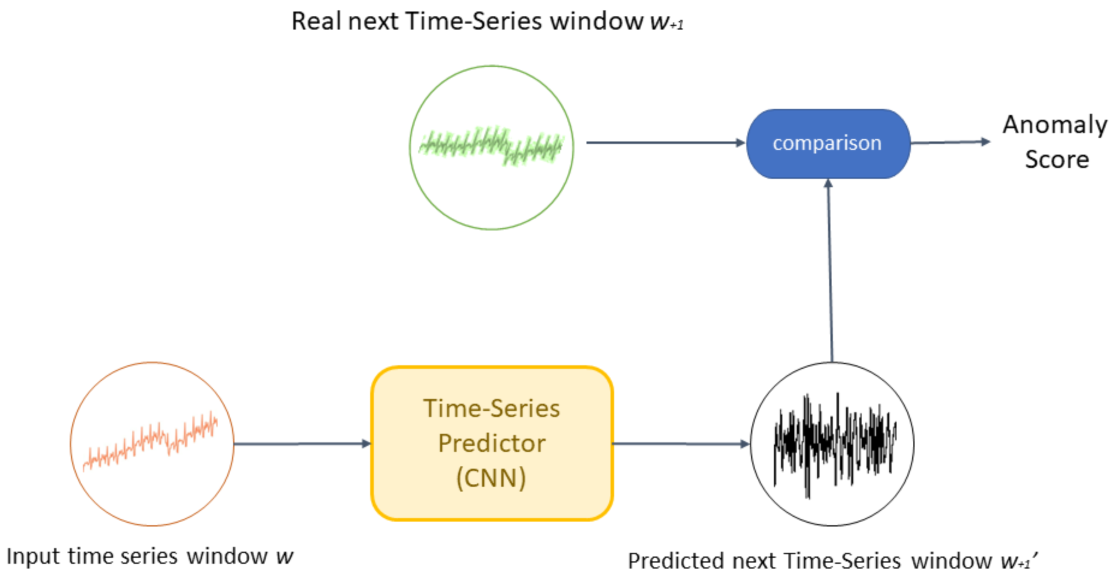

Real next Time-Series window w_{+1}
comparison
Anomaly Score
Time-Series Predictor (CNN)
Input time series window w
Predicted next Time-Series window w_{+1}'

Figure 3.1: DeepAnT architecture: data is passed to the CNN that outputs a prediction for the next time-series window, which then is compared with the real next time-series window.

w'_{+1} will be used as the label for the window w.

The anomaly detector: Once the network outputs the predicted window w_{+1}, that output is passed to the anomaly detector module. This module is in charge of computing the Euclidean Distance in Equation (3.2) between the predicted window w_{+1} and the actual window w'_{+1}

$$(a, b) = \sqrt{(a - b)^2} \tag{3.2}$$

Anomaly evaluation: The computed distance is used as *Anomaly Score*. Therefore a large anomaly score indicates a significant anomaly in the input data. Here a threshold needs to be set and according to the architecture's authors, this threshold needs to be chosen depending on different aspects of the time series in analysis, such as the domain of application, the length of the window, the number of variables, and so on.

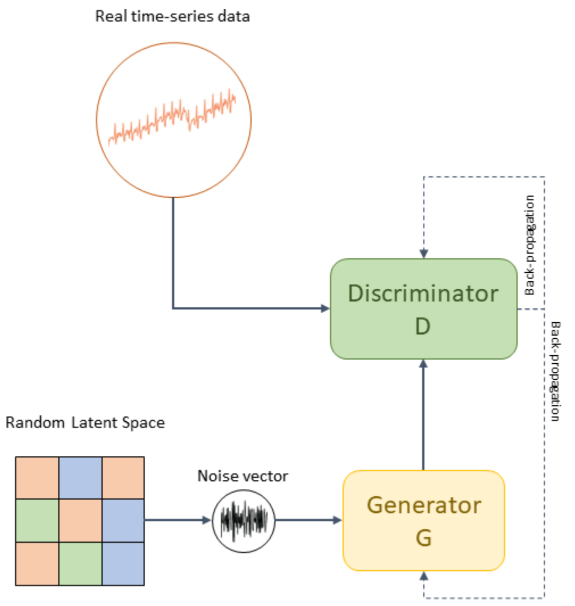
The problem of setting a specific time-series-related threshold is present in all the considered architecture in this work and mostly in all the best-performing state-of-the-art models.

3.2.2 TanoGAN

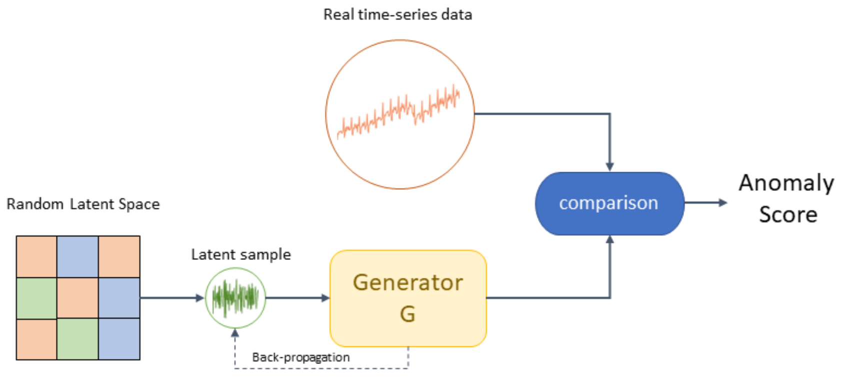
TanoGAN is a GAN-based model proposed for Deep Anomaly Detection. The model is composed of a Discriminator D and a Generator G, To handle the time dependency of time-series data, the authors decided to build G and D as LSTM networks. In particular, they propose a shallow Discriminator made of a single-layer LSTM with 100 hidden units and a medium-depth Generator of 3 stacked LSTM layers with 32, 64, and 128 hidden units. The pipeline of this method can be divided into two sub-processes reported in Figure (3.2) and described below.

First sub-process: learning data distribution: In the first sub-process Generator G is trained to produce fake data as realistically as possible, and following the adversarial paradigm Discriminator D is trained to distinguish between real and fake data. During this process, the input to the Generator is a noise vector $\mathbf{z}$ randomly selected from the latent space $\mathbf{Z}$. Original training data is then divided into sequences that we call windows w of length s_w. The generator generates fake windows in the same form as real ones. Both real sequences and fake sequences are fed to the Discriminator which learns to distinguish between them, while the Generator learns how to produce more realistic data. $G(z)$ can be defined as the function learned by the Generator to map input noise vectors $\mathbf{z} \in \mathbf{Z}$ to the real time-series space $\mathbf{x} \in \mathbf{X}$, while $D(x)$ is the function that models the Discriminator and outputs the probability of the input $\mathbf{x}$ to be real. As stated by the authors, after enough training iterations G and D will not improve anymore; this means that G is capable of generating realistic data and D is able to effectively distinguish between fake and real data. In this phase G and D attempt to optimize a competitive loss, hence we can see them as two agents that are playing a *min-max* game as described in Equations (3.3), where E_x is the expected value of x and E_z is the expected value of z.

$$\min_G \max_D \{E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(\mathbf{z})))]\} \tag{3.3}$$



(a) Adversarial training



(b) Mapping real-data to the latent space

Figure 3.2: TAnoGAN Architecture: Time Series Anomaly Detection with Generative Adversarial Networks [20]

Second sub-process: mapping real data to latent space: At this stage, the generator knows the mapping $G : \mathbf{Z} \mapsto \mathbf{X}$, i.e. from the latent space representation to the real data space. The aim of the second sub-process is to map real time-series sequences $\mathbf{x} \in \mathbf{X}$ to the learned latent space $\mathbf{z} \in \mathbf{Z}$, but GANs are not designed to have the inverse mapping $G^{-1} : \mathbf{X} \mapsto \mathbf{Z}$ such that $G^{-1}(\mathbf{x}) \in (Z)$. So the model proceeds to find $\mathbf{z} \in \mathbf{Z}$ which is used as input for the generator G returns as output a sequence $\hat{x} = G(\mathbf{z})$ that is as similar as possible to the real time-series sequence $\mathbf{x}$. In particular, this process starts with a randomly sampled $\mathbf{z_1} \in \mathbf{Z}$ and feeding it to the generator G to get a fake sequence $\hat{x} = G(\mathbf{z})$, here a loss function $\mathcal{L}$ is defined to provide gradient for optimizing parameters of $\mathbf{z_1}$ to get e new position $\mathbf{z_2} \in \mathbf{Z}$ in this way the position of $\mathbf{z} \in \mathbf{Z}$ is optimized in an iterative way through back-propagation. The loss function $\mathcal{L}$ is defined as the combination of two parts a residual loss $\mathbf{L}_R$ and a discrimination loss $\mathbf{L}_D$.

Residual loss $\mathbf{L}_R$ is a measure of the dissimilarity between the real sequence $\mathbf{x}$ and the generated fake sequence $\hat{x} = G(\mathbf{z})$

$$\mathcal{L}_R(\mathbf{z}) = \sum |\mathbf{x} - G(\mathbf{z})|. \tag{3.4}$$

Discrimination loss $\mathcal{L}_D$ is defined through an intermediate feature representation of the Discriminator. Being $f(\cdot)$ that intermediate feature representation, $\mathcal{L}_D$ is defined as

$$\mathcal{L}_D(\mathbf{z}) = \sum |f(\mathbf{x}) - f(G(\mathbf{z}))|. \tag{3.5}$$

Defined its two components, the loss function can be obtained as in Equation (3.6), where the $\mathcal{L}_R$ is in charge of enforcing the similarity between the real window and the generated ones, and the $\mathcal{L}_D$ makes the generated sequence $G(\mathbf{z})$ lie in the manifold of $\mathbf{X}$

$$L(z, \gamma) = (1 - \gamma) \cdot \mathcal{L}_R(\mathbf{z}) + \gamma \cdot \mathcal{L}_D(\mathbf{z}). \tag{3.6}$$

It is important to notice that even though G and D are used to optimize parameters of $\mathbf{z}$, their parameters are not updated during this process, i.e. only parameters of $\mathbf{z}$ are updated during back-propagation.

Anomaly evaluation: As during the training the Generator learns how to generate realistic-looking sequences based on the general data distribution, we can directly derive an anomaly score $A(\mathbf{x})$ for an input sequence $\mathbf{x}$ from previously defined loss $\mathcal{L}$

$$A(\mathbf{x}) = (1 - \lambda) \cdot \mathcal{R}(\mathbf{x}) + \lambda \cdot \mathcal{D}(\mathbf{x}), \tag{3.7}$$

where the residual score $\mathcal{R}(\mathbf{x})$ and the discrimination score $\mathcal{D}(\mathbf{x})$ are defined by the residual loss $\mathcal{L}_R(\mathbf{z})$ and the discrimination loss $\mathcal{L}_D(\mathbf{z})$, respectively, at the final updating iteration of the mapping procedure to the latent space.

3.2.3 USAD

USAD is a Deep Anomaly Detection method based on an Encoder-Decoder architecture within an adversarial training framework. It is proposed with the aim of taking all the advantages of Auto-Encoders (AE) and adversarial training while mitigating the limitations of both. This twofold nature of the model allows training a better AE in the adversarial training process while also allowing to overcome problems of mode collapse and non-convergence that can occur in GANs.

USAD is made of 3 components: an Encoder network E and two decoder networks D_1 and D_2. Figure (3.3) shows the relationship between the three components, which form an architecture with two AEs, AE_1, and AE_2, sharing the same encoder network

$$AE_1(\mathbf{w}) = D_1(E(\mathbf{w})), \qquad AE_2(\mathbf{w}) = D_2(E(\mathbf{w})). \tag{3.8}$$

The training procedure can be divided into two phases, firstly both the AEs are

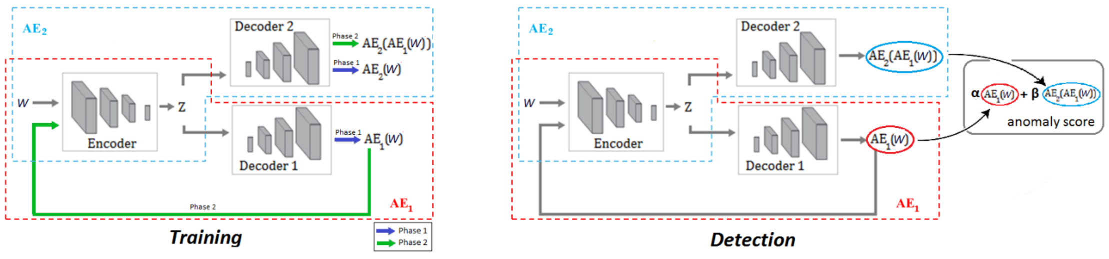


Figure 3.3: USAD architecture illustrating the information flow at training (left) and detection stage (right). [25]

trained to reconstruct the input window $\mathbf{w}$, then are also trained in an adversarial way such that AE_1 acts as a generator and AE_2 acts as a discriminator.

Phase 1: Autoencoder training: The goal in this stage is to teach each AE to replicate the input. The encoder compresses input data $\mathbf{w}$ to the latent space Z and then each decoder recovers it. The difference between the reconstructed samples and the original data, the so-called *Reconstruction Error*, is the minimization objective of this first part of training. The loss of each AE is formalized as

$$\begin{aligned} \mathcal{L}_{AE_1} &= ||\mathbf{w} - AE_1(\mathbf{w})|| \\ \mathcal{L}_{AE_2} &= ||\mathbf{w} - AE_2(\mathbf{w})||. \end{aligned} \tag{3.9}$$

Phase 2: Adversarial training: This is the novelty and most interesting phase of the proposed approach. In this part, exploiting the adversarial framework, AE_1 is considered as *generator*, and AE_2 is considered as *discriminator*. The training process works as follows: input data $\mathbf{w}$ is fed to AE_1, i.e. the Encoder E compresses data into the latent space Z, the compressed data is then reconstructed by the decoder D_1; the obtained output $\hat{\mathbf{w}}_{AE_1}$ is again compressed by the encoder E, and then reconstructed by the decoder D_2. This time the objectives of the two AEs are different. AE_1 learns how to produce an output that when processed by AE_2 gives a result that is as similar as possible to the original data $\mathbf{w}$, in other terms the objective of AE_1 is to minimize the difference between $\mathbf{w}$ and the reconstructed output of AE_2, while the objective of AE_2 is to maximize the error when the input data is from AE_1 and is not original data. In this context as a training objective, we have a GAN-like dual-player min-max game

$$\min_{AE_1} \max_{AE_2} ||\mathbf{w} - AE_2(AE_1(\mathbf{w}))||. \tag{3.10}$$

Formerly the two losses to be minimized during the entire training procedure can be expressed as the combination of Equations 3.8 and 3.10 in an evolutionary scheme where the contribution of each part evolves over time

$$\begin{aligned} \mathcal{L}_{AE_1} &= \frac{1}{n}||\mathbf{w} - AE_1(\mathbf{w})||_2 + \left(1 - \frac{1}{n}\right)||\mathbf{w} - AE_2(AE_1(\mathbf{w}))||_2 \\ \mathcal{L}_{AE_2} &= \frac{1}{n}||\mathbf{w} - AE_2(\mathbf{w})||_2 - \left(1 - \frac{1}{n}\right)||\mathbf{w} - AE_2(AE_1(\mathbf{w}))||_2, \end{aligned} \tag{3.11}$$

where n denotes a training epoch.

Anomaly evaluation: As other methods also USAD provides an Anomaly score (A) computed based on the previously defined loss function of which the two components are parameterized by α and β that are used to control the trade-off between false positives and true positives

$$A(\mathbf{w}) = \alpha||\mathbf{w}||_2 + \beta||\mathbf{w} - AE_2(AE_1(\mathbf{w}))||_2. \tag{3.12}$$

3.2.4 Transformer Anomaly

Transformer networks are among the most innovative architecture of the last years which achieved great success in various areas such as Natural Language Processing (NLP), object detection, and also time-series. This type of network has been able to reach surprising results thanks to its power in the unified modeling of global representation and long-range relations.

The usage of Transformers in anomaly detection comes from two basic intuitions about its functioning. Authors of Anomaly Transformers noticed that the temporal association distribution of each time point, which can be obtained from the self-attention map, can offer a more detailed representation of the temporal context by highlighting dynamic patterns like the period or trend of a time series. This association distribution is referred to as the *series-association.* Moreover, the authors noticed that it is harder for anomalies to build strong associations with the whole series. So, the association distribution of anomaly time points shall concentrate on adjacent time regions. In other words, due to continuity and rarity, anomalies should be concentrated in limited and small adjacent time intervals; authors refer to such an adjacent-concentration inductive bias as the *prior-association*; while normal time points can provide informative associations with the entire time series. These two observations led to a completely new anomaly criterion for each time point defined as the distance between its *prior-association* and its *series-association*, named *Association discrepancy.* From a technical point of view, to embody the *prior-association* and the *series-association* authors introduce the so-called *Anomaly-Attention* block.

Transformer Anomaly architecture: In anomaly transformer the vanilla architecture of Transformers has been renovated with the objective of overcoming original limitations found when using these networks in anomaly detection. The proposed architecture is characterized by alternately stacking the *Anomaly-Attention* blocks and feed-forward layers. Such a stacking strategy enables one to learn underlying associations from deep multi-level features. Since the single-branch mechanism of the vanilla Transformer architecture is not capable of modeling the *prior-association* and the *series-association* simultaneously, authors propose a two-branch structure for Anomaly-Attention block as reported in Figure (3.4). For the **prior-association** they use a learnable Gaussian kernel to find the prior with respect to the relative temporal distance, together with a learnable parameter σ for that Gaussian kernel which makes the prior-association adapt to the various time-series patterns. Meanwhile, the **series-association** is thought to learn the association from the raw series finding the most effective associations.

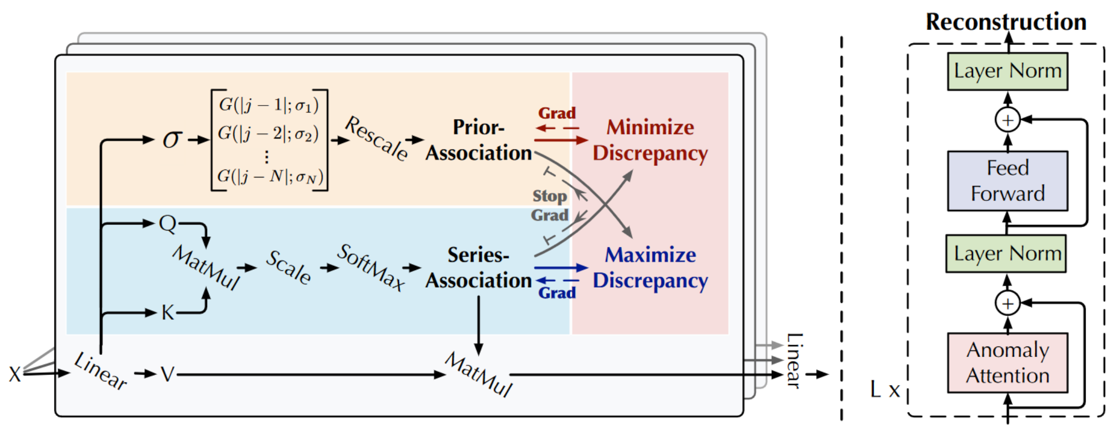


Figure 3.4: "Anomaly Transformer architecture. Anomaly-Attention (left) models the prior-association and series-association simultaneously. In addition to the reconstruction loss, our model is also optimized by the min-max strategy with a specially-designed stop-gradient mechanism (gray arrows) to constrain the prior- and series- associations for more distinguishable association discrepancy." [27]

This two-branch structure maintains the temporal dependencies of each time point which is really informative, and also reflects the adjacent-concentration prior and the learned real association respectively such that their discrepancy (**Association discrepancy**) can be used to distinguish between normal and anomaly points.

- **Association Discrepancy:** The association discrepancy is formalized as the symmetric KL divergence between prior and series associations found by the model. The association discrepancy is averaged from multiple layers to obtain a more informative measure as

$$AssDis(\mathcal{P}, \mathcal{S}; \mathcal{X}) = \left[\frac{1}{L} \sum_{l=1}^{L} \left(KL(\mathcal{P}^l_{i,:} || \mathcal{S}^l_{i,:}) + KL(\mathcal{S}^l_{i,:} || \mathcal{P}^l_{i,:}) \right) \right]_{i=1,\dots,N}, \quad (3.13)$$

where $KL(\cdot||\cdot)$ is the KL-divergence computed between two discrete distributions corresponding to every row of $\mathcal{P}^l$ and $\mathcal{S}^l$. $AssDis(\mathcal{P}, \mathcal{S}; \mathcal{X}) \in \mathbb{R}^{N \times 1}$ is the point-wise association discrepancy of $\mathcal{X}$ with respect to prior-association $\mathcal{P}$ and series-association $\mathcal{S}$ from multiple layers. The i-th element of results corresponds to the i-th time point of $\mathcal{X}$. From previous observations, anomalies will present smaller $AssDis(\mathcal{P}, \mathcal{S}; \mathcal{X})$ than normal time points, which makes **Association Discrepancy** inherently distinguishable.

- **Min-Max strategy:** Choosing a maximizing strategy during training optimization would end up in an extremely reduced scale of the parameter σ making the prior-association useless. For this reason, authors propose a min-max strategy. In practice, the **minimize phase** consists of making the prior-association $\mathcal{P}^l$ approximate the learned series association $\mathcal{S}^l$, so that the prior-association will adapt to various patterns of the time series. In the **maximization phase** the series association is optimized to enlarge the association discrepancy, forcing the series-association to pay more attention to non-adjacent regions. So, considering also the reconstruction loss, the loss functions of these two phases are:

$$\begin{aligned} \mathcal{L}_{min\_phase} &= \mathcal{L}(\hat{\mathcal{X}}, \mathcal{P}, \mathcal{S}_{detach}, -\lambda, \mathcal{X}) \\ \mathcal{L}_{max\_phase} &= \mathcal{L}(\hat{\mathcal{X}}, \mathcal{P}, \mathcal{S}_{detach}, \lambda, \mathcal{X}) \end{aligned} \quad (3.14)$$

where $\lambda > 0$ and $*_{detach}$ are used to stop the gradient backpropagation of the association. In anomaly points will be much harder for $\mathcal{P}$ to approximate

$\mathcal{S}_{detach}$ in the minimize phases, and gain stronger constraint to the series-association in the maximize phase than in normal points, and this makes the Association Discrepancy well normal-abnormal distinguishable.

Anomaly evaluation: To evaluate anomalies authors propose an anomaly score based on both Association discrepancy and reconstruction error. In fact, to have a better reconstruction, anomalies will usually decrease the association discrepancy which will anyway derive a high anomaly score. So, an high value of the Anomaly score (A) defined in Equation (3.15) means a high probability of the point being an anomaly

$$A = Softmax\Big(- AssDis(\mathcal{P}, \mathcal{S}; \mathcal{X})\Big) \odot \Big[||\mathcal{X}_{i,:} - \hat{\mathcal{X}}_{i,:}||_2^2\Big]_{i=1,\dots,N}. \tag{3.15}$$

3.3 Datasets

The methods deeply described above have been trained and tested on multiple publicly available data sets. By doing so we can explore the behavior of each method with data coming from different domains. In the next paragraphs, we issue an introduction to each of the employed datasets, reporting a brief analysis of the anomalies distribution and some general information about the collected data. Moreover, Table (3.2) reports a summary of the dimensions of the datasets and the number of anomalies in each one.

3.3.1 Numenta Anomaly Detection Benchmark Dataset

The Numenta Anomaly-Detection Benchmark [31] data set (in the following NAB data set) is a collection of more than 50 different real-world and artificial univariate time series data sets that contain different types of anomalies. This data

Dataset	**#features**	**#records**	**anomaly ratio**
NAB TEMP	1	22.6K	11.10%
NAB AD	1	1.6K	11.08%
NAB TRAFFIC	1	2.3K	11.16%
MSL	55	130K	6.24%
SMD	38	1.4M	4.16%
SWAT	51	1M	6.13%

Table 3.2: Summary of the composition of the datasets used in this work and the number of anomalies in each one of them.

set has been built to provide extensive support for Anomaly Detection Algorithm evaluation. As claimed by the authors the data set contains types of anomalies that are representative of all the most encountered types of anomalies in real-world applications. NAB dataset provides a significant contribution to the anomaly detection landscape, in fact, all datasets in the collection are labeled by hand following a meticulous and documented procedure, as we already discussed in the introduction about the difficulties and costs of realizing such a work. For these reasons the NAB data set has become one of the benchmark data sets for the evaluation of Anomaly Detection algorithms.

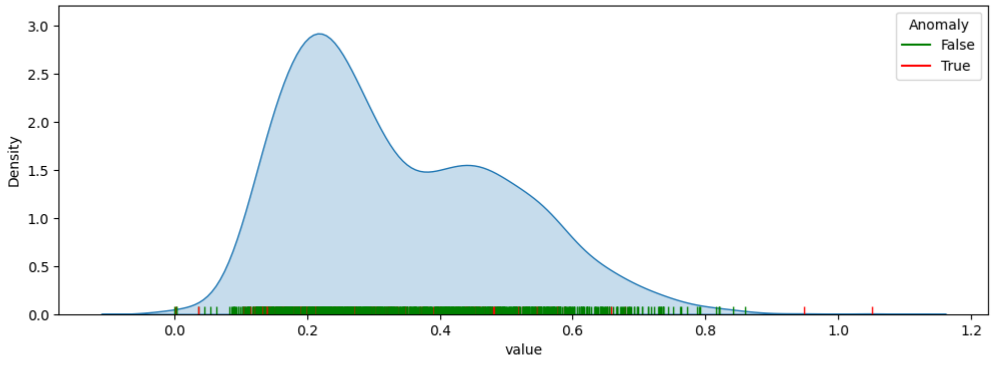
Our work is aimed at anomaly detection on multivariate time series. But, all the methods reported in this research have also been tested by the authors on this Dataset. From the different publications it is not easy to trace the specific time series used by the respective authors, in fact, in most cases the publications, both of the individual methods and of other surveys, report an average of the results obtained on the NAB collection. Since we thought it appropriate to use this collection of datasets as a starting point for the evaluation of the different algorithms, we decided to use 3 of the datasets proposed by the NAB, selecting those on which it was possible to obtain results compatible with the averages published by the authors of each algorithm. We then narrowed our choice to only those datasets reporting anomalies with known causes, and from these we selected:

- Machine temperature measurements: Temperature sensor data of an internal component of a large, industrial machine.

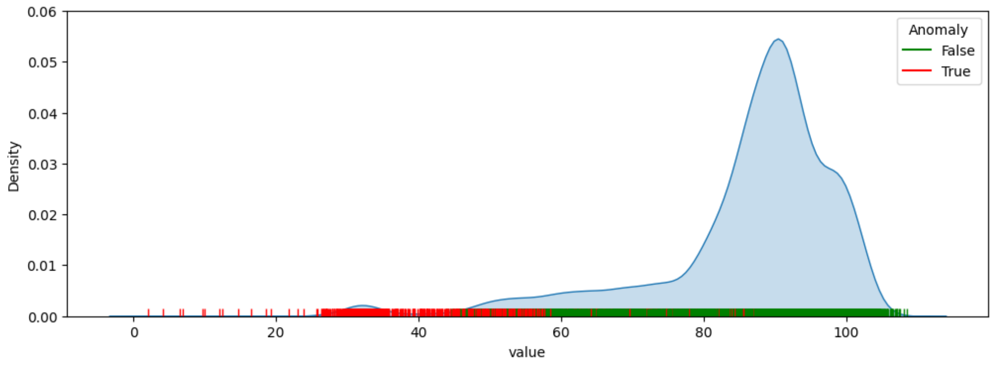
- Real-time traffic occupancy: Real-time traffic data from the Twin Cities Metro area in Minnesota, collected by the Minnesota Department of Transportation.

- Advertisement exchange (CPC): Online advertisement clicking rates, where the metrics are cost-per-click (CPC)

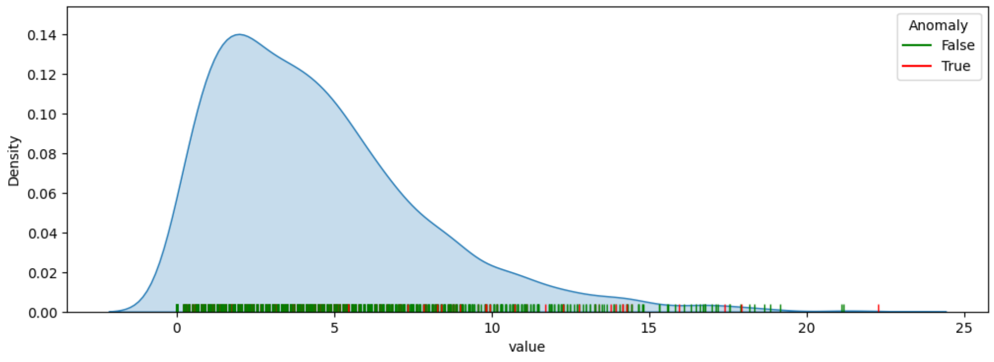
Figure (3.5) reports the KDE plot and the Rug plot for the three NAB Datasets employed. KDE plot allows us to study the probability distribution of data estimating the probability density function in a non-parametric way. The Rug plot shows where anomaly and normal points are located with respect to the density function. We can see that for the NAB_TEMP dataset, the great majority of the anomalies are located on the tail of the density probability function, in this case, the Anomaly Detection task can be assessed only by looking at this function, having a high probability of anomaly when a record has value in the tail. For the other two datasets, the separation between anomalous and normal points is not so evident. However, we can conclude that also in univariate time series the Anomaly Detection task may be non-trivial and Deep Learning can help us to obtain better results.



(a) NAB AD



(b) NAB TEMP



(c) NAB TRAFFIC

Figure 3.5: Density plots for the NAB datasets employed in this work. On the x-axis is reported the rug plot which shows the distribution of anomaly and normal points

The NAB Dataset is publicly available and free to download on the web. [32]

3.3.2 Server Machine Dataset

In 2019, the creators of the anomaly detection technique "OmniAnomaly" released a dataset known as Server Machine Dataset (SMD). Even though this work does not focus on the OmniAnomaly approach, we chose to use SMD in our experimentation due to its popularity among many other surveys that explore the same subject matter. Furthermore, SMD is one of the largest publicly available datasets in this field. As stated by the original authors, SMD contains data gathered from 27 servers of a large Internet Service Provider, with 38 variables measured every minute for 5 weeks. Data has been then manually labeled by domain experts. The dataset's anomalous records make up around 4.16% of its total observations. [33]

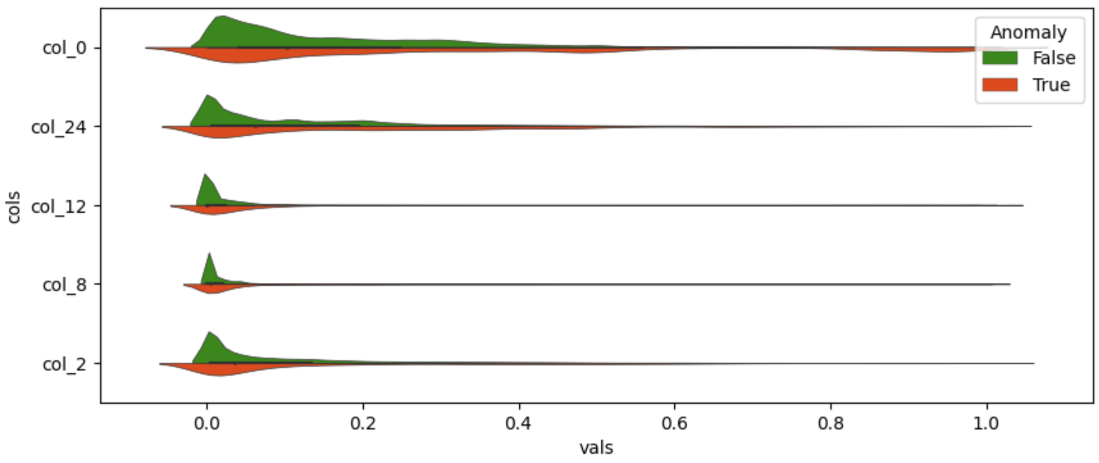


Figure 3.6: Violin plot of top-5 features correlated with the class "Anomaly/Normal" of the SMD dataset.

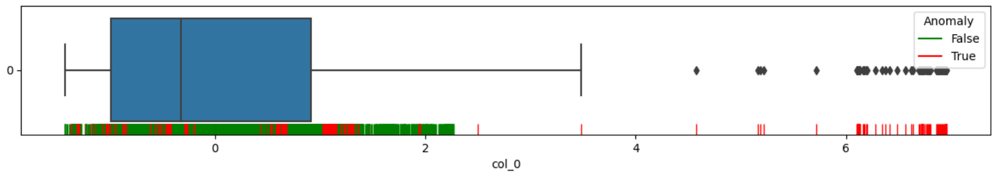
To study the anomalies distribution of the SMD dataset we can use the Violin plot in Figure (3.6) about the top-5 correlated features to the class "Anomaly/Normal" in the dataset. Thanks to this plot we can see that there are no big differences between the distributions of normal points and anomalous points so, the Anomaly Detection task is for sure non-trivial, and Deep Learning methods can help us find underlying correlations and patterns which can describe the abnormality or normality of records.

3.3.3 Mars Science Laboratory Dataset

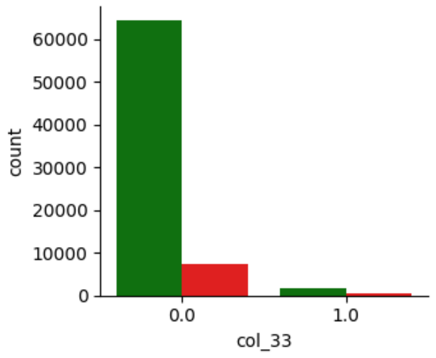
MSL is a public dataset released by NASA and contains data gathered by the Curiosity rover in the Mars Science Laboratory mission, a robotic space mission conducted by NASA to explore the surface and climate of Mars. MSL includes information gathered by the mission's various instruments, including the Mars Hand Lens Imager (MAHLI), the Chemistry and Camera (ChemCam) instrument, the Sample Analysis at Mars (SAM) suite, the Radiation Assessment Detector (RAD), and the Rover Environmental Monitoring Station (REMS).

Data contains measurements recorded during the mission from 55 different devices. MSL has been employed by different authors to train and test new Anomaly detection techniques, due to the fact that it is publicly available and labeled by NASA experts. [34]

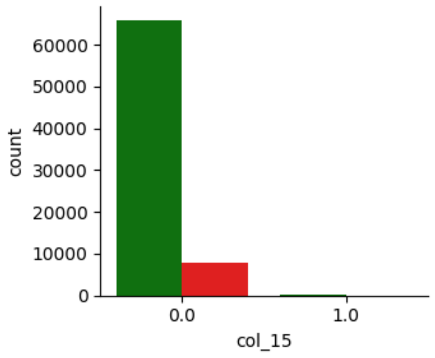
Also in this case we can inspect the top-5 correlated features with the target class ("Anomaly/normal") trying to understand how anomalies are distributed in the dataset. Figure (3.7a)reports the Box plot of the top-1 correlated feature which is a continuous feature, and Figures (3.7b, 3.7c, 3.7d, 3.7e) report bar plots of the other considered features. In this case, the box plot shows that all outliers of this feature correspond to anomalies, but we can see that there is a non-negligible number of anomalies placed in the interquartile range. With bar plots, we can see that the distribution of anomalies in those features follows the general dataset's anomalies distribution. Again, the problem of Anomaly Detection is non-trivial and we can benefit from the application of Deep Learning techniques.



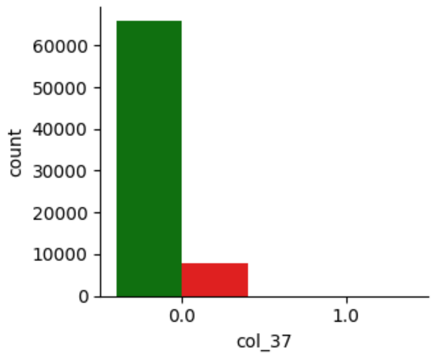
(a)



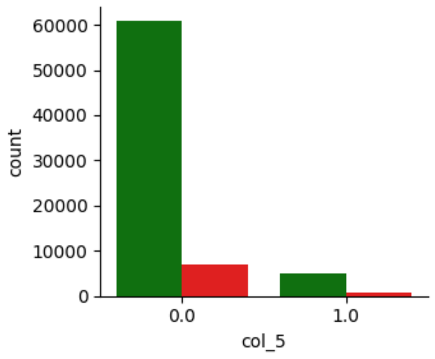
(b)



(c)



(d)



(e)

Figure 3.7: Box plot and bar plots for the MSL dataset of the top-5 correlated features with the target feature.

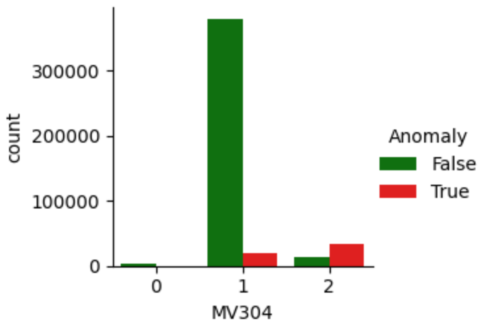
3.3.4 Secure Water Treatment Dataset

The Secure Water Treatment dataset (SWaT) is a publicly available dataset, it was collected from a fully operational scaled-down water treatment plan. The main purpose of the dataset is to support experimentally validated research in the design of secure Cyber-Physical Systems.

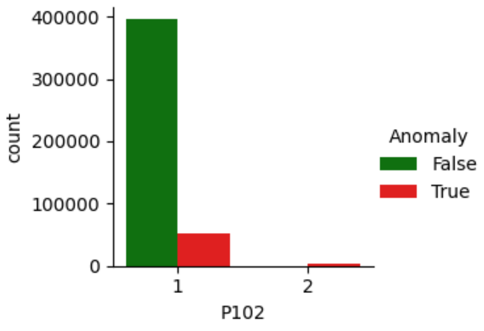
The dataset includes data collected for 11 days from several sensors and actuators in the plant. During the 11 days, the plant was functioning non-stop 24 hours/day. During the first 7 days the system was run without any kind of attack, in the last 4 days several attacks with duration varying from seconds to one hour have been done on the system in a structured way. Since attacks were done in a controlled way and with the specific purpose of collecting logs and data, the labeling of normal and anomaly records was straightforward and 100% reliable.

The dataset is made of about 1M measurements of 51 attributes. It is published and publicly available on the Singapore University of Technology and Design website [35].

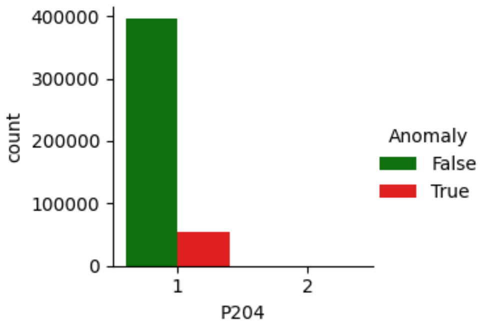
Also for this Multivariate dataset, we can inspect the distribution of anomalies in the top-5 correlated features with the target feature, which for $SWAT$ dataset, are all categorical features. Figure (3.8 shows bar plots of the selected features. In this case, we have an interesting behavior in feature *MV304* where the value 2 has a distribution of anomalies that do not match with the general distribution of the dataset, in fact, there is a majority of anomalies for records in which the value of this variable is 2. Meanwhile, for other features, we can not spot any blatant deviation from the general dataset trend.



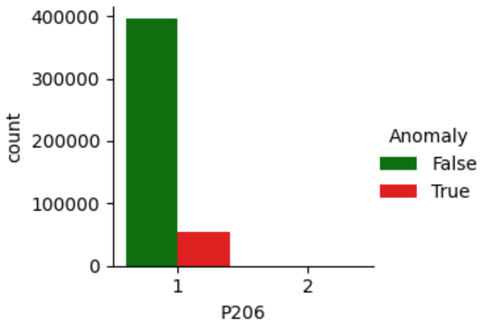
(a)



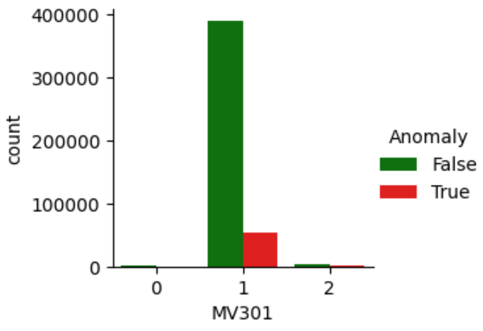
(b)



(c)



(d)



(e)

Figure 3.8: Bar plots for the SWAT dataset of the top-5 correlated features with the target feature.

Chapter 4

Experimental results

In this chapter, we will explain the implementation of the experiments along with the structure of the code used to run them. Then we will introduce the design choices made to be able to replicate tested methods and the choices made in setting the different hyper-parameters required. A specific section will discuss the metrics used to evaluate the various algorithms. Finally, the last section will show the results obtained from the different methods studied when tested on the datasets selected as benchmarks.

4.1 Experiments implementation

The goal of this research was to be able to evaluate in the most general way possible the different approaches to Time Series Deep Anomaly Detection published so far. Thus, the main output of this work is an evaluation tool, which can also be adapted to other methods, and which is responsible for acting as an interface between the different proposed benchmark datasets and the algorithms to be tested. The work done to achieve the research objective can thus be divided into 2 main parts.

1. **A first implementation of methods and exploration of hyperparameters**: at this stage, the code of all proposed methods was collected or re-implemented. Then, starting from the values given by the authors of each method, the optimal hyperparameters were explored to replicate the results published in the respective articles. At this stage it became clear, confirming the information reported in the various publications, that the length of the windows used for time series analysis (hereafter SEQ_LEN) is a hyperparameter that heavily influences the final performance of the algorithms. At the end of this phase, optimal values of other hyperparameters such as learning rate, size of internal layers, the maximum number of epochs for training, etc. were then explored and identified, leaving SEQ_LEN as a parameter to investigate in the second phase.

2. **Running the various experiments**: at this stage, the code for the massive execution of the various experiments was prepared. Each of the 4 methods was then initialized with the previously obtained hyperparameters, trained, and tested for each of the 6 proposed datasets with 3 different SEQ_LEN values, namely $[30, 50, 100]$. The choice of these 3 values is due to the need of having a well-defined and limited set on which was possible to test all the datasets and methods. The process of choosing the values started by collecting values proposed by the method's authors for datasets on which they also worked and after that reducing values to an acceptable number trying to replicate published performances per each method. As we will see later, the SEQ_LEN parameter is a fundamental one for the performance of models on different datasets, since it indicates the length of the windows to be fed to algorithms and can have a big impact on the way each dataset expresses

its intrinsic time dependency to the model. Obtained the general setting for the experimentation, the program was run several times, by setting a different seed value each time in order to ensure the reproducibility of the results, this is the actual comparison part of the process. In this last phase also emerged the problem of *finding an appropriate threshold*, described in the next paragraph.

4.1.1 The threshold problem

A common problem for all the methods studied is the identification of an appropriate threshold for the classification of anomalies based on the anomaly scores produced by the algorithms. In fact, each algorithm proposes a novel method for calculating the anomaly score, and thus it was not possible to directly compare the obtained scores, let alone establish an unambiguous threshold for all of them. The classification phase of all algorithms stipulates that, when an appropriate threshold is defined, for values less than that threshold the data are classified as non-anomaly, while for values higher than the threshold the corresponding data are classified as anomalies. Therefore, it was decided to scale the obtained anomaly scores in the range [0,1] and then explore threshold values between [0.3 and 0.9] in order to subsequently study the evolution of the evaluation metrics based on the variation of this threshold. Studying values of threshold lower or higher than the interval chosen would be pointless. After this scaling, we expect to have the Anomaly Score equal to 1 for records that are seen as certain anomalies by models, and equal to 0 for points that are seen as certain normal by models, so that, we can interpret the scaled Anomaly Score as a probability of anomaly that the model assigns to each record. It should therefore be specified that in a practical application of one of these methods, it would be necessary to carry out such a study concerning the threshold at the validation stage. In this work, we decided to assess the threshold identification at test time to be able to study how the scoring of each algorithm changes based on this parameter.

4.2 Comparison Metrics

Traditional classification models are evaluated with metrics that compare the predicted class from the classifier with the actual class of the samples. Since in this study, we are talking about Anomaly Detection, our main task is to classify data points as anomalies or normal, so we can safely use traditional metrics to evaluate the performance of tested models. To build a proper evaluation strategy most of the time it is useful to construct a **confusion matrix** where the (j, k)-th element counts the number of times that the actual class is j whereas the predicted class is k.

In the binary case (such as our case, since we are classifying Anomaly/Non-Anomaly) the confusion matrix becomes really simple to build and interpret. If the two classes to be predicted are *True* (Anomaly) and *False* (Non-anomaly) the confusion matrix is the one depicted in Table 4.1.

		Actual class	
		Positive	**Negative**
Predicted class	**Positive**	TP	FP
	Negative	FN	TN

Table 4.1: Binary classification confusion matrix

The following terminology is often used when referring to the counts tabulated in a confusion matrix:

- **True positive (TP)**: the number of positive examples correctly predicted by the classification model.

- **False negative (FN)**: the number of positive examples wrongly predicted as negative by the classification model.
- **False positive (FP)**: the number of negative examples wrongly predicted as positive by the classification model.
- **True negative (TN)**: the number of negative examples correctly predicted by the classification model.

Accuracy, Precision, and Recall

Starting from the confusion matrix we can define different metrics for the evaluation of our model.

- **Accuracy**: is the performance measure generally associated with classification algorithms. It is the ratio of correct predictions over the total number of data points classified
$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}, \tag{4.1}$$
- **Precision** (also called *positive predictive value*): Indicates how many of a j-object (in binary classification, commonly *True* class is considered) predictions are correct. It is defined as the ratio of correct positive predictions and overall positive predictions
$$\text{Precision, } p = \frac{TP}{TP + FP}, \tag{4.2}$$
- **Recall** (also called *sensitivity*): Indicates how many of the j-object (in binary classification, commonly *True* class is considered) samples are correctly classified. It is defined as the fraction of j-object predictions over the total number of j-object samples
$$\text{Recall, } r = \frac{TP}{TP + FN}, \tag{4.3}$$

At first look *Accuracy* could seem to be a good metric for comparing the performance of tested methods. Actually, it is a measure that treats every class as equally important, and we are in a highly imbalanced situation, in fact as shown in Table (3.2) we have a very small number of anomalies compared to the total size of the datasets. In this kind of situation using accuracy may lead to an overestimation of the models' performance since having a high number of normal points makes the models to be really good classifiers for the negative class, but with poor scoring on the positive class. The problem is that in situations like this, the *rare class* (Anomaly samples) is considered more interesting than the majority class, so we need metrics that are class-specific and can allow us to understand how good models are in finding anomalies. *Precision* and *Recall* are such a kind of metric and are widely employed in applications similar to ours, in which the successful detection of the rare class is more significant than the detection of the other.

So, now the challenge is to find the model that is the best in maximizing both *Precision* and *Recall.* Hence, the two metrics are usually summarized in a new metric that is **F1-score**. In practice *F1-score* represents the harmonic mean between precision and recall, so, an high value ensures that both are reasonably high:

$$\text{F1-score} = \frac{2}{\frac{1}{r} + \frac{1}{p}} = \frac{2rp}{r+p}. \tag{4.4}$$

In choosing the metric to use for the comparison, based on the reasoning made above, to the best of our knowledge, we are perfectly in line with other surveys in the literature. Actually, in the field of anomaly detection, a further step is needed. We need to keep in mind two facts on which our experimentation and the totality of literature are based:

- **windowed data**: we are arranging data in windows of dimension equal to

SEQ_LEN, which as we will see later is an important parameter. Researchers that studied this topic used to consider windows that contain anomalies as anomalies themselves.

- **consecutive anomalies**: when working on real-world data, researchers start with the assumption that in a real-world working environment, if the anomaly detection system finds an anomaly, that anomaly should be assessed and removed, to make the system restart to work in a proper way. This is translated in considering consecutive anomalies as one anomaly, so classifying a point as anomalous in a consecutive window of anomalies, makes all the consecutive points be classified also as anomalies.

The metrics obtained with measures based on those two concepts in literature are defined as Point-Adjusted (PA) metrics. Since our study did not focus on which is the best metric for comparing Deep Anomaly Detection methods, but our objective was to find a good general strategy of comparison of the state-of-the-art models we thought it right to use the Point-Adjusted scores as was done in all publications on which this work is based, that means that we will study PA-Precision, PA-Recall, and PA-F1-Score. For the sake of simplicity from now on, we will omit the specification "Point-Adjusted" and we will refer to these scores respectively as Precision, Recall, and F1-Score.

4.3 Results

4.3.1 The influence of SEQ_LEN parameter

During the preparation of our experiments, we left the parameter SEQ_LEN for a deeper a-posteriori analysis, and it is the point on which we will start the exploration of obtained results. The appendix A contains figures that show how F1-Score that is obtained by each method varies with respect to SEQ_LEN and *threshold* parameters. Since we run a large number of experiments obtaining complex graphical representations of results, we can use Figure (4.1) and Figure (4.2) to resume the findings described in detail in the figures in the appendix. Figure (4.1) reports the maximum obtained F1 score of each method with the three values of SEQ_LEN considered and aggregated by datasets. Figure (4.2) shows the same aggregating results by methods.

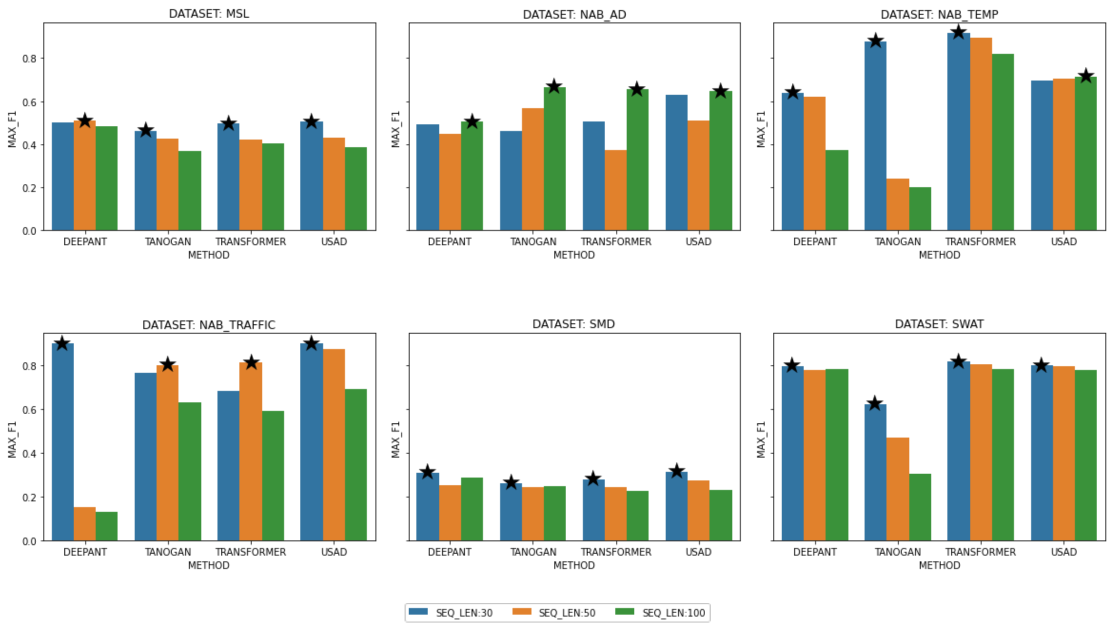


Figure 4.1: F1-Score obtained by each method when tested on different datasets with different SEQ_LEN values. The star indicates the SEQ_LEN value which gave the best performance for each method with the same dataset.

In Figure (4.1) most of the datasets behave better always with the same SEQ_LEN value, disregarding of the method tested. While the same is not valied for methods, which perform better with different SEQ_LEN values when applied to different datasets.

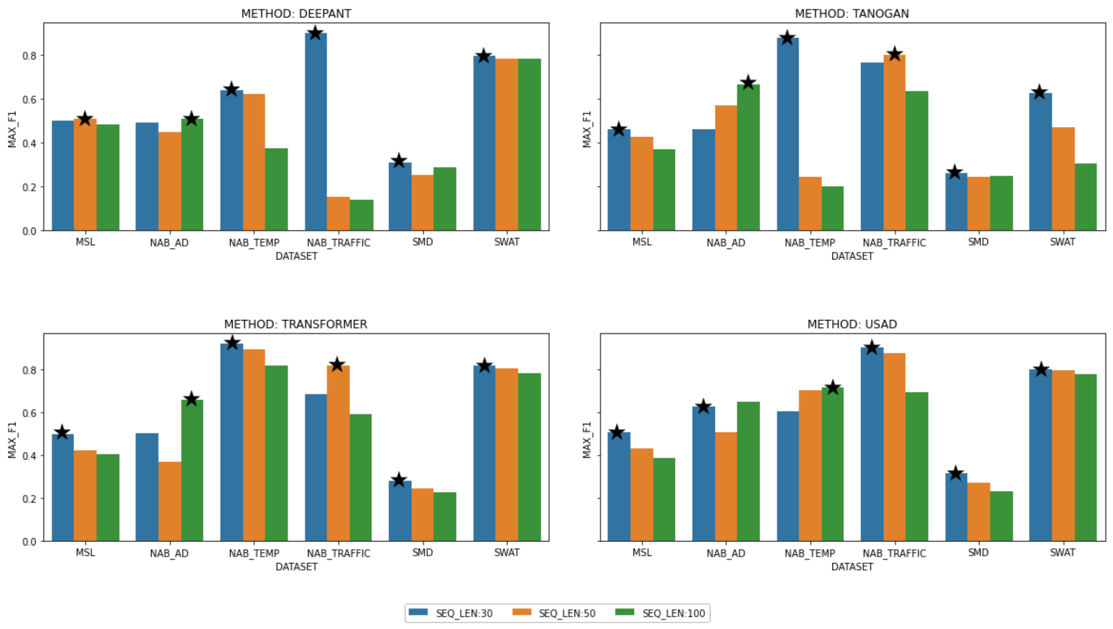


Figure 4.2: F1-Score obtained by each method when tested on different datasets with different SEQ_LEN values. The star indicates the SEQ_LEN value which gave the best performance for each dataset when used in different methods.

Table (4.2) reports the best values of SEQ_LEN found in our experimentation for each of the datasets tested. We could clearly find a unique value for most of the dataset. $SWAT$, SMD, and NAB_AD scored best with the same $SEQ\_LEN =$ 30. For dataset MSL 3 out of 4 methods perform best with $SEQ\_LEN = 30$ and 1 method ($DeepAnT$) performs best with $SEQ\_LEN = 50$, but we can see that the difference between the best score and other value is really small so that, also taking a kind of majority-vote approach, we can safely use $SEQ\_LEN = 30$ for all methods in future comparisons of results about this dataset. The same reasoning is valid for the NAB_TEMP dataset. The $NAB\_TRAFFIC$ dataset shows to have a more balanced situation, even though we can notice that there is a high discrepancy of results for method $DeepAnT$ and this makes us think that in the case we would define a sub-optimal value for this dataset, $SEQ\_LEN = 30$

would be the one since having other values would penalize too much the *DeepAnT* method.

DATASET	SEQ_LEN	# optimal	# sub-optimal
MSL	30	3	1
NAB_AD	100	4	-
NAB_TEMP	30	3	1
NAB_TRAFFIC	30	2	2
SMD	30	4	-
SWAT	30	4	-

Table 4.2: Optimal values of *SEQ_LEN* parameter found for each dataset.

We interpret these observations as saying that the *SEQ_LEN* parameter, seems to be strictly related to the dataset on which Deep Anomaly Detection is performed, thus it needs to be explored and studied in a proper way. This is the first contribution of this work as we have been able to understand that this parameter is *dependent* on the dataset. Although it is not possible to generalize this dependence, based on results obtained and graphs shown we support the concept that the choice of this parameter has to be done based on dataset characteristics which can be, for example, the anomaly ratio in data or the distance between anomalies. So, instead of tuning the *SEQ_LEN* parameter in a sort-of random-blind way, we can a-priori define a restricted set of values looking at the nature of the data we are working on.

4.3.2 Performance vs threshold

After finding the optimal value of the *SEQ_LEN* parameter, we can proceed with a deeper performance analysis made on results obtained with the settings described above. Figures below show F1-score, Precision, and Recall for all the methods tested with each dataset considered, and the specific *SEQ_LEN* found previously. As it would seem obvious at this point of the dissertation, performance metrics evolve with respect to the value of the threshold. In fact, for an ideal and perfectly working model, all anomaly data points would get a high anomaly score, while all the normal data points would get a low anomaly score following a mechanism reported by the example in Figure (4.3). Of course, in real models, the separation between normal data scores and anomaly data scores can be smaller and the model may also be wrong in classifying some records. The importance of the threshold resides in this situation. In enough-good working models, we suppose that higher anomaly scores are most likely associated with an anomaly and vice versa. Playing around with the threshold value can allow us to reduce the number of *false negatives*, while also mitigating the presence of *false positives* in order to obtain a good classification score. In such a situation the threshold value

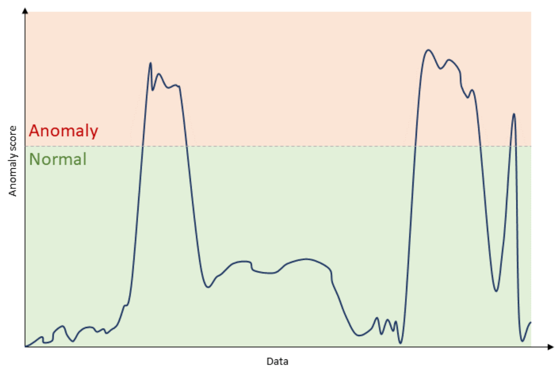


Figure 4.3: Anomaly score example: data which score is below the threshold is marked as *normal*, data which score is above the threshold is marked as *anomaly*

can be seen as a sort of severity of the model, such that with a high value of the threshold we are saying to the model to consider as anomaly only points for which it is extremely secure that they are anomalies resulting in having a high *Precision* score, but our model is not perfect and this kind of setting will produce also a high number of false negatives resulting in a drop of the *Recall* score.

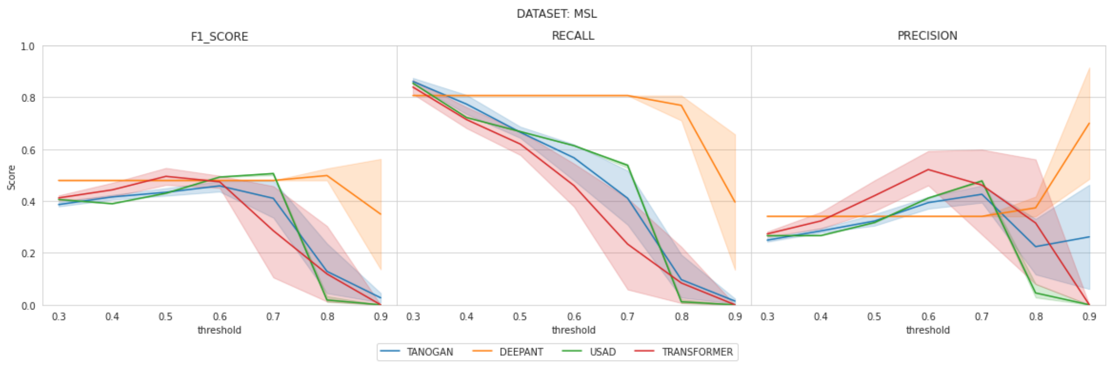


Figure 4.4: F1-Score, Recall, and Precision for dataset MSL. The color of the lines indicates the different models tested.

Figure (4.4) reports scores for dataset MSL. We can not spot singular trends with all methods performing roughly as we would expect with an exact decreasing trend for Recall, a less-clear increasing trend for Precision but an overall expected F1-Score trend that decreases as the threshold increases demanding the model for a higher severity.

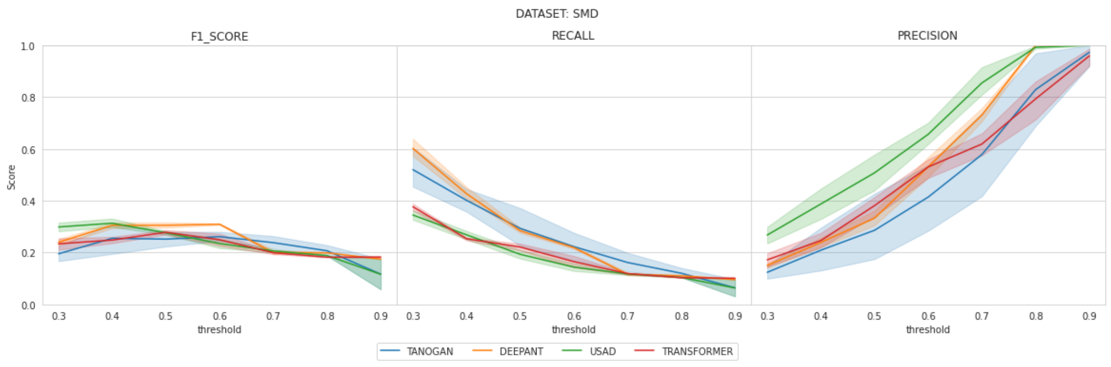


Figure 4.5: F1-Score, Recall, and Precision for dataset SMD. The color of the lines indicates the different models tested.

Figure (4.5) shows scores for the dataset SMD. Trends for this dataset are perfectly fitting our expectations, with decreasing Recall, increasing Precision, and really stable yet decreasing trend for F1-Score over the increase of the threshold value. This let us think that for this dataset the influence of the threshold parameter is not as strong as others.

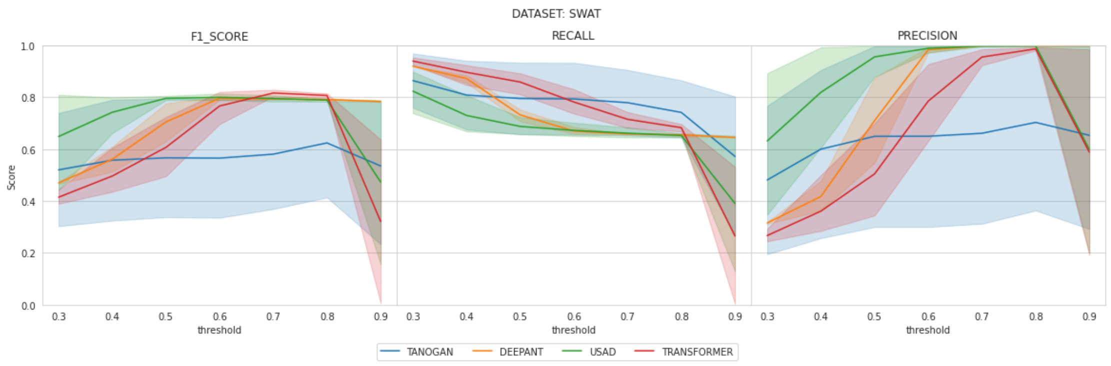
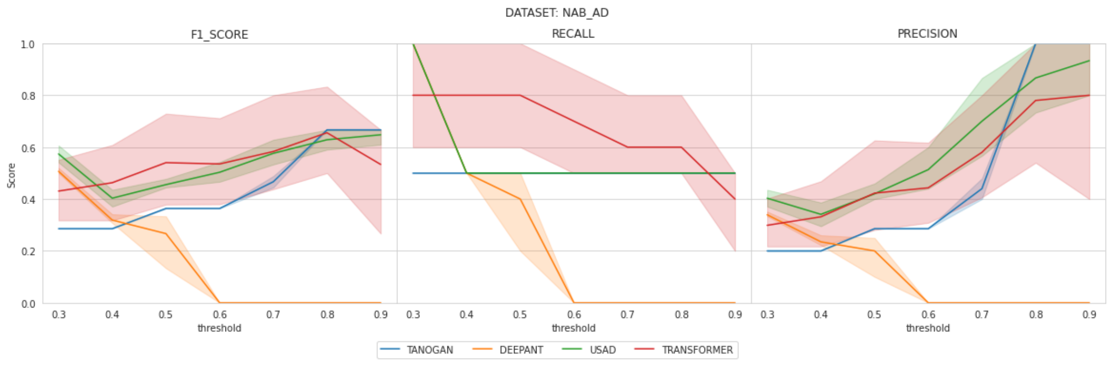
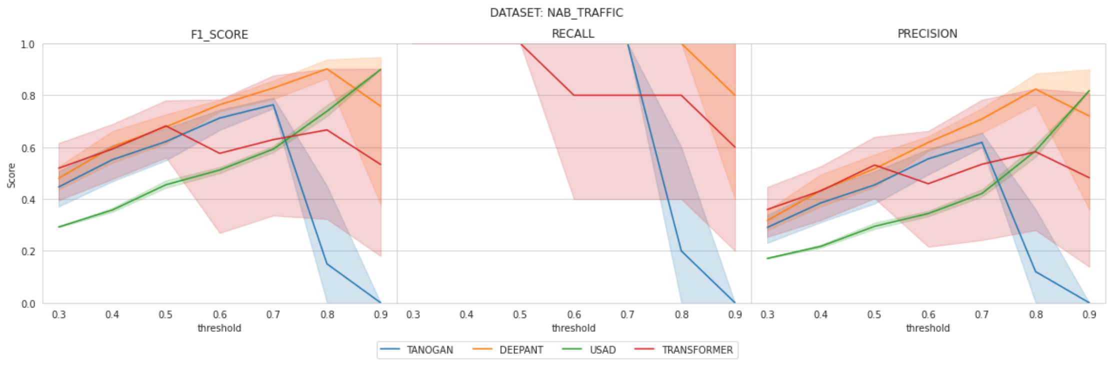


Figure 4.6: F1-Score, Recall, and Precision for dataset *SWAT*. The color of the lines indicates the different models tested.

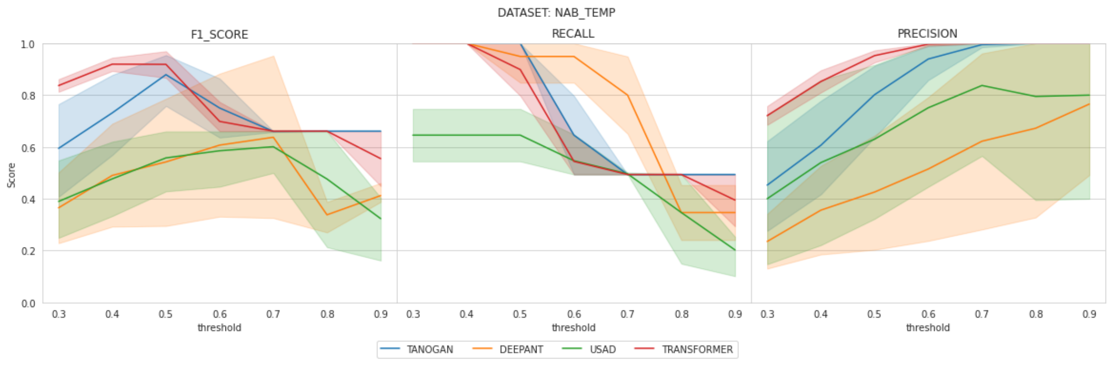
Figure (4.6) shows scores for the dataset *SWAT*. Also in this case, trends are perfectly in line with our expectations, with decreasing Recall, increasing Precision, and really stable yet decreasing trend for F1-Score over the increase of the threshold value. The two noticeable facts are the big drop in performances with a very high value of the threshold, and the much lower (even if more stable) performance of *TAnoGAN* compared with other models. For the drop with a high value of the threshold, we expect to have some isolated high peaks of the Anomaly-Scores which actually are false positive instances that lower performance.



(a) NAB-AD



(b) NAB-TRAFFIC



(c) NAB-TEMP

Figure 4.7: F1-Score, Recall, and Precision for NAB datasets. The color of the lines indicates the different models tested.

Figure (4.7) shows scores for the three datasets of the *NAB* collection. Here we have a more confusing situation, that might be difficult to study. The unusual curves are due to the limited dimension of these datasets. In fact, having fewer data points with also fewer anomalies and smaller windows of consecutive anomalies make the model reach good Recall scores with small values of the threshold with huge drops over the increasing of the parameter.

We have analyzed the results obtained in our experimentation following the previously described classification metrics. We could not identify a best-overall method, but we could understand how the threshold can be important in determining the performance of Deep Anomaly Detection models. This is the second contribution of our work, we comparatively analyzed the importance of the Anomaly Score threshold of Deep Anomaly Detection models, overcoming previous surveys which focused only on the best-obtained score of the methods without a proper discussion on the importance of this parameter.

4.3.3 A new comparison metric: Discrimination Score

The threshold mechanism explained above can be better verified in Figure (4.8) which reports the anomaly score produced for all methods tested on the *SWAT* dataset. In the figure, the plotted line shows the **anomaly score** obtained, while green dots indicate anomaly points **marked as an anomaly** from the model (*True Positives*), and red dots indicate anomaly points **marked as normal** from the model (*False Negatives*), as we can see each model presents a different curve and its score can vary based on the threshold value chosen. This variability of performance makes it difficult to build a fair comparison of methods even when considering a single dataset, but looking at the graphs in Figure (4.8) we can make some reasonings that may help in finding a good path for the comparison. We can now analyze the anomaly score plots trying to find evidence of what we said before and also trying to establish a fair comparison method for the models.

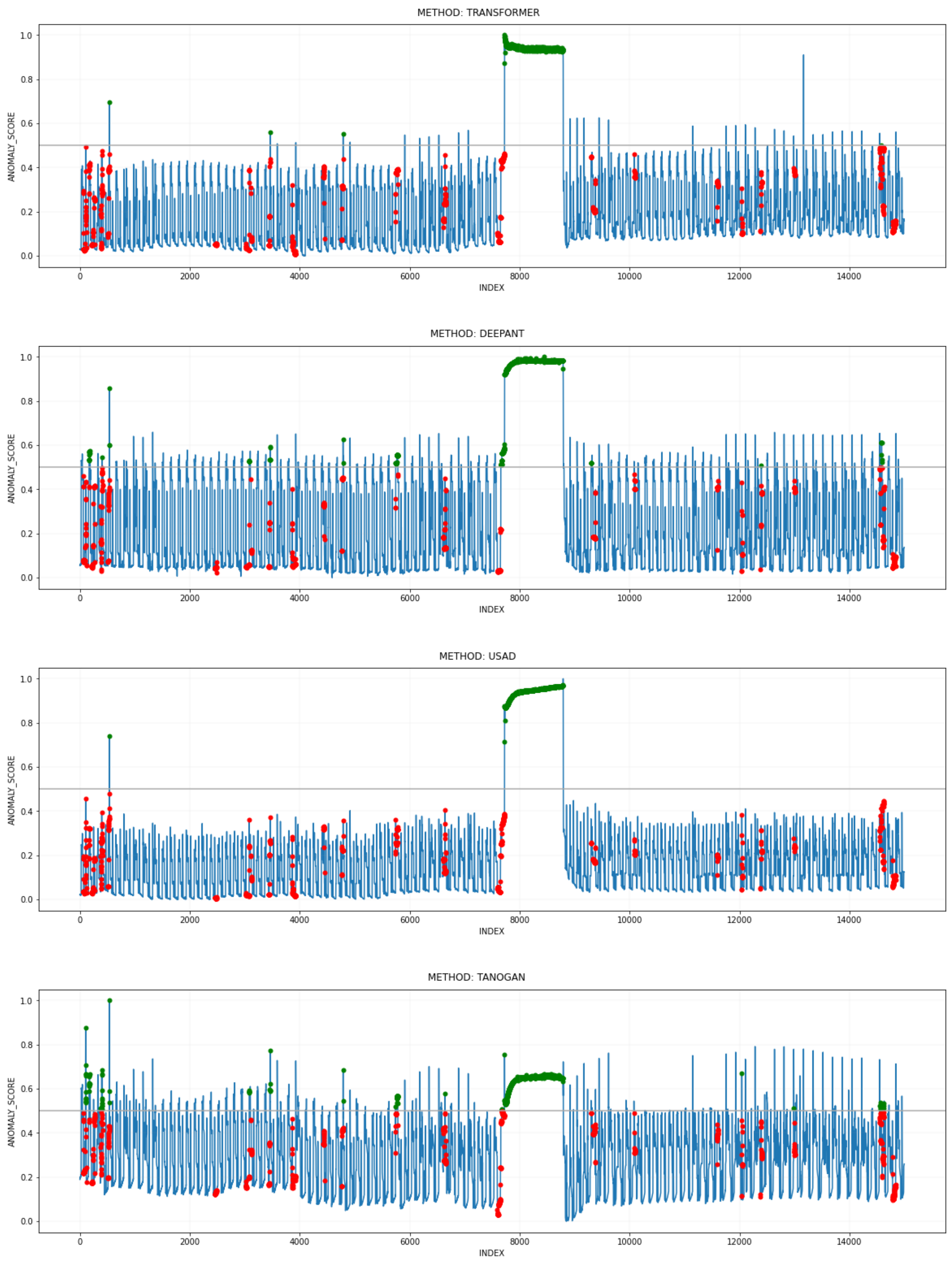


Figure 4.8: SWAT anomaly score

In the figure, we can see that the dataset has a wide window of consecutive anomalies. All methods have been able to detect that window, but the *TAnoGAN* model did not assign to that window Anomaly Scores as high as other models, and this produce the drop in performance with a high value of threshold that we described in the previous paragraph. Also, we can see that anomaly score oscillations in *DeepAnT* and *TAnoGAN* are much wider than in *Transformer* and *USAD*. So, we can say, also supported by scores in Figure (4.6), that *TAnoGAN* and *USAD* are more stable with respect to the threshold than the other two methods. The amplitude of oscillations and the threshold affect the detection of anomalies and the scoring of each method. For example with the same threshold $th = 0.5$ *TAnoGAN* gets more *true positives* than other methods, but also more *false positives*, while models with fewer oscillations miss some anomalies, but also are able to avoid a lot of *false positives*.

With the analysis of the Anomaly Score obtained with this experimentation, we could understand that it is not possible to detach the performance of the model from the threshold problem, but we can take a new path by trying to study how much each model is able to distinguish between anomalies and normal points without using the well-know classification metrics. So, we define a more general metric of comparison starting from the definition of the Anomaly Score. The Anomaly Score in this work ranges between 0 and 1. It is assigned to records in such a way that higher and lower scores correspond to anomalies and normal data, respectively. Considering an ideal perfectly working model, it would assign an Anomaly Score of 1 to all anomalies and an Anomaly Score of 0 to all normal points.

We can now consider the mean score assigned from the model to points known to be anomalies and the mean score assigned to points known to be normal, that for the ideal perfect model would be $\mu_{Anomaly} = 1$ and $\mu_{Normal} = 0$, at this point we can define a new score which we name **Discrimination Score** (DS) as in Equation (4.5) so that, our ideal model would get $DS = 1$

$$DS = \mu_{Anomaly} - \mu_{Normal}. \tag{4.5}$$

For a real model, we can formalize the behavior of this score following observations below:

- The score is can take values between -1 and 1: $-1 \leq DS \leq 1$
- A DS score less than 0 indicates that the model is not working at all since the mean score of normal points would be greater than the mean score of anomaly points.
- An high DS score indicates that the model is very good at identifying both anomalies and normal points.
- A low DS score indicates that the model has not enough discrimination power, and assigns either a too-low score to anomalies or a too-high score to normal points.

With this score, the object of comparison of different models with the same dataset becomes to find the model which provides the higher Discrimination Score. We can now compare the findings obtained with our Discrimination Score with the obtained results of each method. Figure (4.10) reports per each dataset the best F1-Score obtained by all methods, while Figure (4.9) shows per each dataset the Discrimination Scores obtained by all methods, in this figure the star indicates the method that obtained the best F1-Score with the corresponding dataset. When the star is empty it means that the best discrimination score did not match with the best F1-Score, the filled star indicates that the best Discrimination Score matched the best F1-Score.

We can see in this figure how the Discrimination Score reflects the behavior of the Anomaly Score for the *SWAT* dataset. We have that *Transformer*, *DeepAnT*, and *USAD* which have a more net separation between anomalous points and normal points in Figure (4.8), have also higher DS score compared with TAnoGAN method which has more oscillations and not such a net separation.

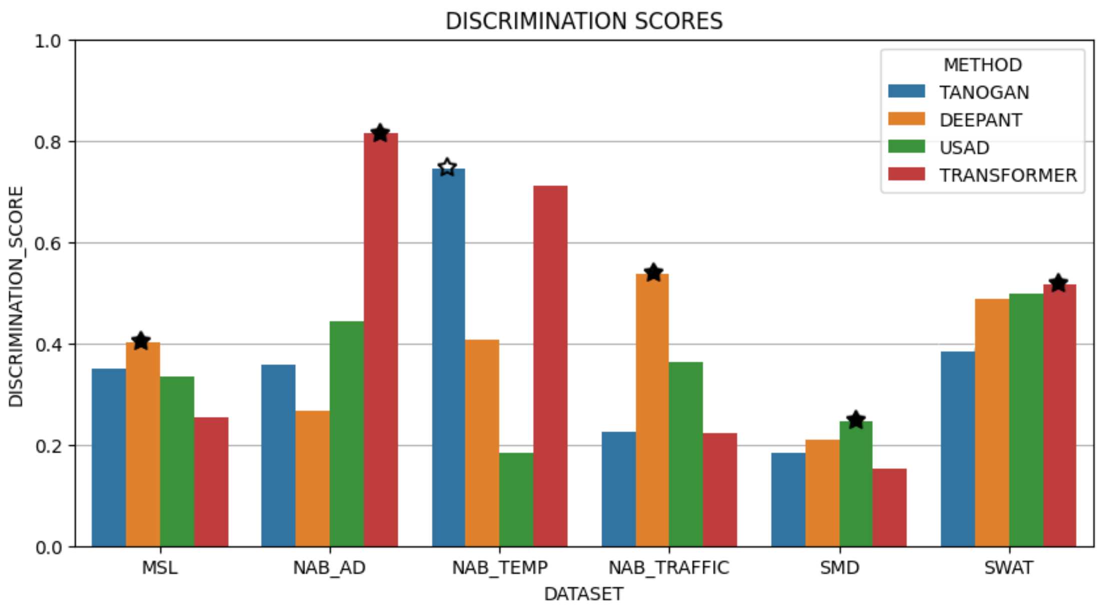


Figure 4.9: Discrimination Scores per each dataset of methods tested in this work. A star indicates that the DS score is the best obtained for a specific method, when there is a correspondence between the best DS score and the best F1-Score the star is filled, and the star is empty otherwise.

We can see that in 5 out of 6 datasets, there is a correspondence between the best DS score and the best F1-Score. We have one dataset for which there is no such matching, actually, this is normal and can happen. In fact, a method that has the best DS score may anyway be overcome by other methods with a proper adjustment of the threshold. We can conclude with the last important contribution of this work, saying that the DS score can be an effective way to compare the discrimination power of different methods on the same dataset. Even though the

DS score helps us in comparing the discrimination power of different methods untying the reasoning from the threshold problem, it alone is not sufficient to define which method is the best, and also observations made before need to be taken into account.

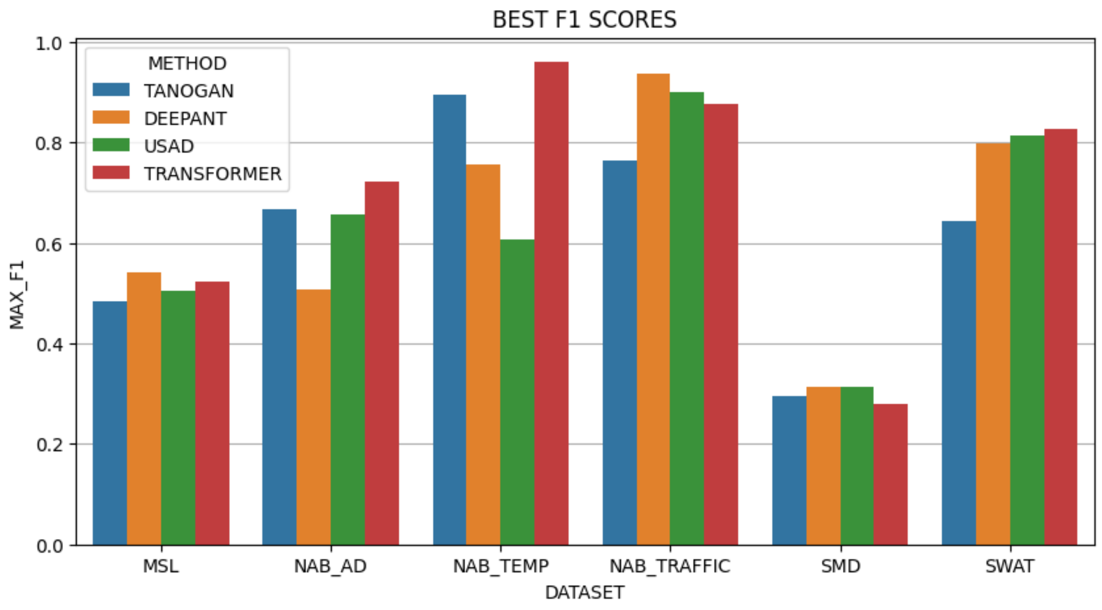


Figure 4.10: Best F1-Score per each dataset obtained by tested methods.

Chapter 5

Conclusions and future works

We started this work with an extensive explanation of what is Anomaly Detection. We reported the most challenging issues in this field of research and also reported the most extensive and regarded surveys. Based on the misses of those surveys and also based on directions and hints found in the literature we built an experimental survey with a different structure than the previous ones. We focused on understanding how different Neural Networks architectures are able to catch underlying correlations and time dependency expressed by different datasets. To limit our work to the newest and most innovative models we defined specific research criteria that allowed us to select 4 Deep Anomaly Detection methods to test. We then selected 6 datasets between the most used in literature for time series Anomaly Detection tasks. During the analysis of the Deep Anomaly Detection methods we also highlighted the **problem of the threshold** which we show to have a big influence on models' performance. In Chapter (4) at first, we analyzed the results obtained in the extensive experimentation finding that the dimension of the time window passed to the models has a big impact on the performance of each model. In that section, we could find out that the time window's dimension parameter is strictly related to the dataset, having most of the dataset reach the best performance always with the same SEQ_LEN value. Then, we explored how the results varied when choosing different values of the threshold for the Anomaly Score classification. We conclude that the threshold is one of the most important parameters in this kind of model and a proper study needs to be assessed when implementing a Deep Anomaly Detection method in order to avoid overfitting and overestimation of models' performance. Then, the extensive study proposed for the threshold parameter enabled us to do some reasoning which led to the definition of a new comparison metric that we named **Discrimination Score**. The aim of this metric is to provide an idea of how much discrimination power is held by the method when employed with a specific dataset. Reported results showed that the Discrimination Score results agree with the classical F1-Score results and we concluded that thanks to the fact that Discrimination Scores is untied from the threshold parameter, we can use it to have a fair idea of what is the capability of each model to distinguish between Normal and Anomalous points.

To sum up the conclusions and findings of this work we can say that the Anomaly Detection problem remains an open problem and an active field of development in the Deep Learning panorama. It is not possible to establish if one model is better than others, but deeper analysis and considerations are always needed. Indeed, we proposed a first step towards a new way of setting up the comparison of different methods. Moreover, we expressed concerns about topics that were not addressed by other surveys and finally we proposed a new score that can be used in conjunction with other classical comparison metrics to understand which algorithm is the best for a given dataset.

Of course, this was only a first step with a new implant of comparison and can be extended and improved under different aspects. Possible forthcoming research may focus for example on enlarging the class of architectures considered, or could focus on studying the newest non-Deep models following the directions pointed out by this research. We think that different new paths may be opened from our work which may deserve to be explored.

Appendix A

Details of results obtained

The Figure (A.1) reports for each **dataset** (rows), the F1-score obtained by each method (colored plots) with each value of *SEQ_LEN* tested (columns), and Figure (A.2) reports for each **method** (rows), the F1-score obtained with each dataset (colored plots) and each value of *SEQ_LEN* tested (columns). Comparing these two figures it is possible to notice an interesting behavior. In fact, while plots in Figure (A.2) are very confused and do not allow us to spot any significant pattern, in Figure (A.1) we can see that for some datasets, best performances of each method are always achieved with the same value of the parameter *SEQ_LEN*.

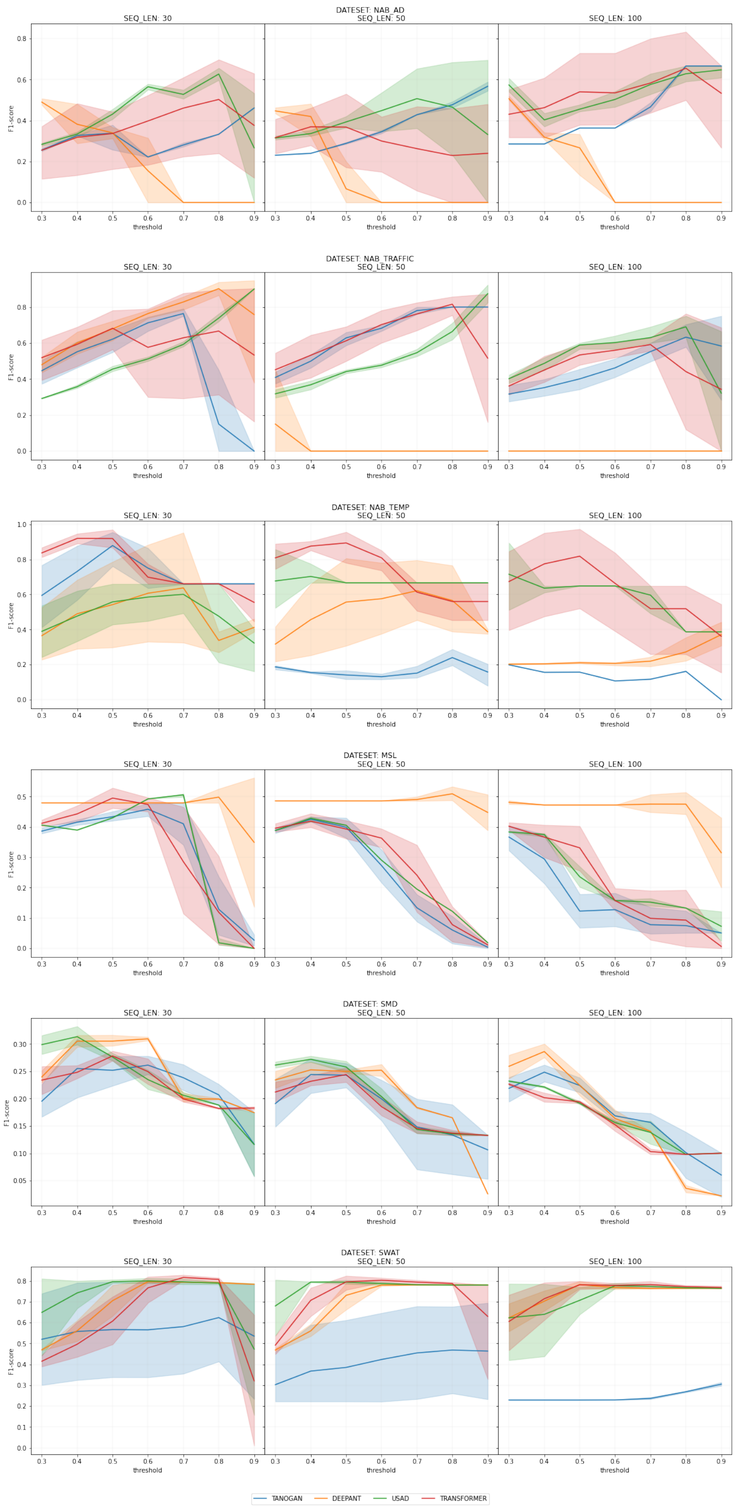


Figure A.1: *SEQ_LEN_DS*

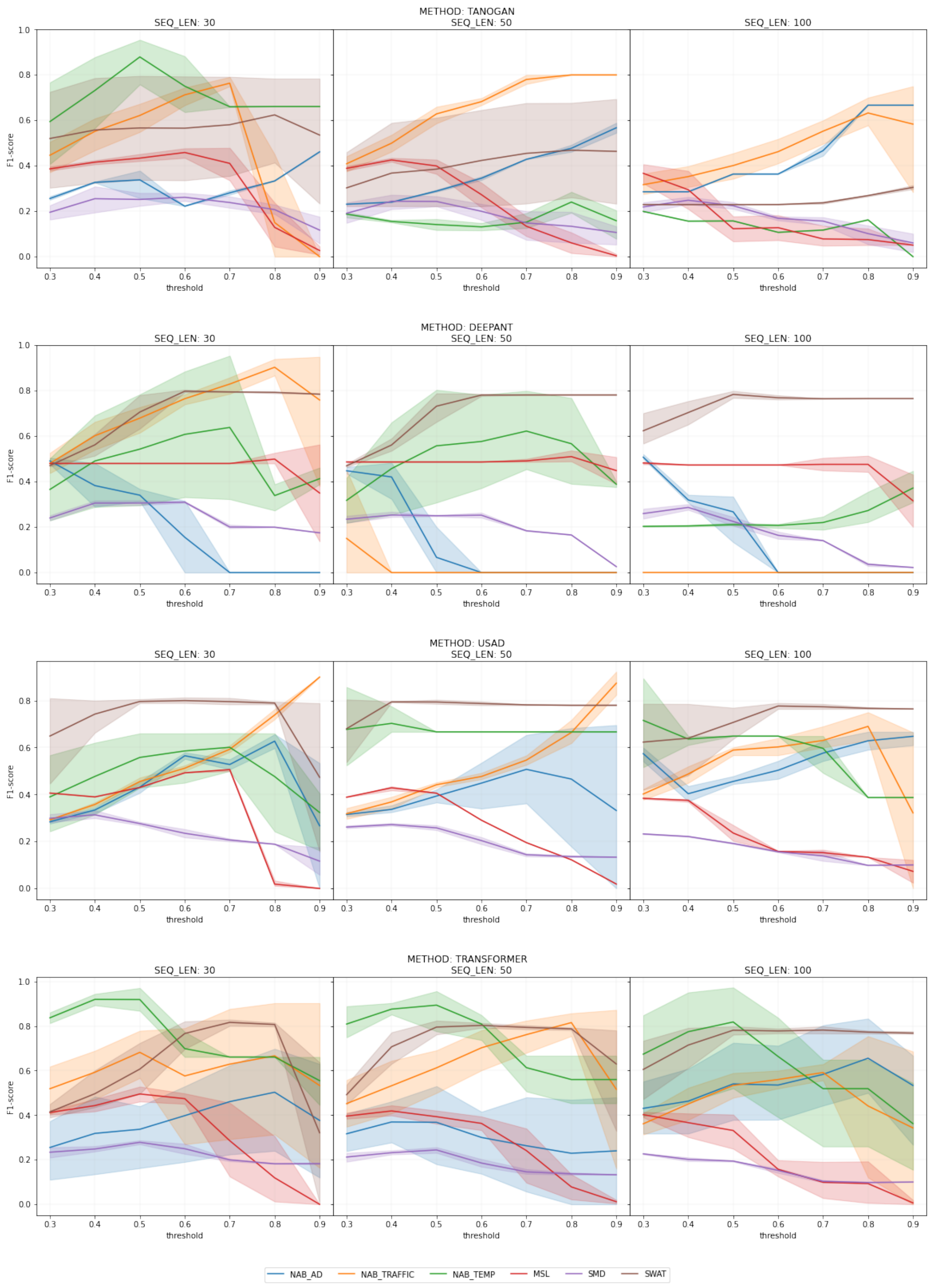


Figure A.2: *SEQ_LEN_METHOD*

Bibliography

[1] Hu-Sheng Wu. «A survey of research on anomaly detection for time series». In: *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). Dec. 2016, pp. 426–431. DOI: 10.1109/ICCWAMTIP.2016.8079887 (cit. on pp. 1, 14).

[2] Arnaldo Sgueglia, Andrea Di Sorbo, Corrado Aaron Visaggio, and Gerardo Canfora. «A systematic literature review of IoT time series anomaly detection solutions». In: *Future Generation Computer Systems* 134 (Sept. 2022), pp. 170–186. ISSN: 0167739X. DOI: 10.1016/j.future.2022.04.005. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167739X22001285 (visited on 10/18/2022) (cit. on pp. 1, 3).

[3] Astha Garg, Wenyu Zhang, Jules Samaran, Savitha Ramasamy, and Chuan-Sheng Foo. «An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series». In: *IEEE Trans. Neural Netw. Learning Syst.* 33.6 (June 2022), pp. 2508–2517. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2021.3105827. arXiv: 2109.11428[cs,stat]. URL: http://arxiv.org/abs/2109.11428 (visited on 10/20/2022) (cit. on p. 1).

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. «Anomaly detection: A survey». In: *ACM Comput. Surv.* 41.3 (July 2009), pp. 1–58. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/1541880.1541882. URL: https://dl.acm.org/doi/10.1145/1541880.1541882 (visited on 11/28/2022) (cit. on pp. 2, 3, 5).

[5] Raghavendra Chalapathy and Sanjay Chawla. *Deep Learning for Anomaly Detection: A Survey.* Jan. 23, 2019. arXiv: 1901.03407[cs,stat]. URL: http://arxiv.org/abs/1901.03407 (visited on 10/19/2022) (cit. on pp. 3, 5, 11, 15).

[6] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. «Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines». In: *IEEE Access* 9 (2021). Conference Name: IEEE Access, pp. 120043–120065. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3107975 (cit. on pp. 3, 8, 11, 14, 15, 17).

[7] Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. «Towards a Rigorous Evaluation of Time-Series Anomaly Detection». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.7 (June 28, 2022). Number: 7, pp. 7194–7201. ISSN: 2374-3468. DOI: 10.1609/aaai.v36i7.20680. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20680 (visited on 10/18/2022) (cit. on p. 3).

[8] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. «Conditional Anomaly Detection». In: *IEEE Transactions on Knowledge and Data Engineering* 19.5 (May 2007). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 631–645. ISSN: 1558-2191. DOI: 10.1109/TKDE.2007.1009 (cit. on p. 3).

[9] *Dealing with artifacts¶*. URL: `https://www.cs.colostate.edu/eeg/data/json/doc/tutorial/_build/html/artifacts.html` (cit. on p. 6).

[10] Ben D. Fulcher, Max A. Little, and Nick S. Jones. «Highly comparative time-series analysis: the empirical structure of time series and their methods». In: *Journal of The Royal Society Interface* 10.83 (June 6, 2013). Publisher: Royal Society, p. 20130048. DOI: `10.1098/rsif.2013.0048`. URL: `https://royalsocietypublishing.org/doi/10.1098/rsif.2013.0048` (visited on 11/28/2022) (cit. on p. 7).

[11] Keiron O'Shea and Ryan Nash. «An Introduction to Convolutional Neural Networks». In: *CoRR* abs/1511.08458 (2015). arXiv: `1511.08458`. URL: `http://arxiv.org/abs/1511.08458` (cit. on p. 11).

[12] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. «DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series». In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 1991–2005. ISSN: 2169-3536. DOI: `10.1109/ACCESS.2018.2886457` (cit. on pp. 11, 20).

[13] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. «Convolutional neural networks for time series classification». In: *Journal of Systems Engineering and Electronics* 28.1 (Feb. 2017). Conference Name: Journal of Systems Engineering and Electronics, pp. 162–169. ISSN: 1004-4132. DOI: `10.21629/JSEE.2017.01.18` (cit. on p. 11).

[14] Tailai Wen and Roy Keyes. *Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning*. May 31, 2019. arXiv: `1905.13628[cs,stat]`. URL: `http://arxiv.org/abs/1905.13628` (visited on 12/14/2022) (cit. on p. 11).

[15] Mohammad Braei and Sebastian Wagner. *Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art.* Apr. 1, 2020. arXiv: 2004.00433[cs,stat]. URL: http://arxiv.org/abs/2004.00433 (visited on 10/19/2022) (cit. on p. 11).

[16] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. *TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks.* version: 3. Nov. 14, 2020. arXiv: 2009.07769[cs,stat]. URL: http://arxiv.org/abs/2009.07769 (visited on 10/19/2022) (cit. on pp. 11, 12, 20).

[17] Hansheng Ren et al. «Time-Series Anomaly Detection Service at Microsoft». In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* KDD '19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Anchorage AK USA: ACM, July 25, 2019, pp. 3009–3017. ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330680. URL: https://dl.acm.org/doi/10.1145/3292500.3330680 (visited on 10/19/2022) (cit. on p. 11).

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. «Generative adversarial networks». In: *Commun. ACM* 63.11 (Oct. 22, 2020), pp. 139–144. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3422622. URL: https://dl.acm.org/doi/10.1145/3422622 (visited on 12/15/2022) (cit. on p. 12).

[19] Dan Li, Dacheng Chen, Lei Shi, Baihong Jin, Jonathan Goh, and See-Kiong Ng. *MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks.* Jan. 15, 2019. arXiv: 1901.04997[cs,

stat]. URL: http://arxiv.org/abs/1901.04997 (visited on 12/15/2022) (cit. on pp. 12, 20).

[20] Md Abul Bashar and Richi Nayak. «TAnoGAN: Time Series Anomaly Detection with Generative Adversarial Networks». In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI).* Dec. 1, 2020, pp. 1778–1785. DOI: 10.1109/SSCI47803.2020.9308512. arXiv: 2008.09567[cs,stat]. URL: http://arxiv.org/abs/2008.09567 (visited on 10/21/2022) (cit. on pp. 12, 20, 25).

[21] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders.* Apr. 3, 2021. arXiv: 2003.05991[cs,stat]. URL: http://arxiv.org/abs/2003.05991 (visited on 12/16/2022) (cit. on p. 13).

[22] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. «A Review on Outlier/Anomaly Detection in Time Series Data». In: *ACM Comput. Surv.* 54.3 (Apr. 30, 2022), pp. 1–33. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3444690. URL: https://dl.acm.org/doi/10.1145/3444690 (visited on 10/19/2022) (cit. on p. 13).

[23] Tingting Chen, Xueping Liu, Bizhong Xia, Wei Wang, and Yongzhi Lai. «Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder». In: *IEEE Access* 8 (2020), pp. 47072–47081. DOI: 10.1109/ACCESS.2020.2977892 (cit. on p. 13).

[24] Zhiqiang Que, Yanyang Liu, Ce Guo, Xinyu Niu, Yongxin Zhu, and Wayne Luk. «Real-Time Anomaly Detection for Flight Testing Using AutoEncoder and LSTM». In: *2019 International Conference on Field-Programmable Technology (ICFPT).* 2019, pp. 379–382. DOI: 10.1109/ICFPT47387.2019.00072 (cit. on p. 13).

[25] 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: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Virtual Event CA USA: ACM, Aug. 23, 2020, pp. 3395–3404. ISBN: 978-1-4503-7998-4. DOI: 10.1145/3394486.3403392. URL: https://dl.acm.org/doi/10.1145/3394486.3403392 (visited on 10/30/2022) (cit. on pp. 13, 20, 28).

[26] Zekai Chen, Dingshuo Chen, Xiao Zhang, Zixuan Yuan, and Xiuzhen Cheng. «Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT». In: *IEEE Internet of Things Journal* 9.12 (June 2022). Conference Name: IEEE Internet of Things Journal, pp. 9179–9189. ISSN: 2327-4662. DOI: 10.1109/JIOT.2021.3100509 (cit. on p. 13).

[27] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. *Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy*. version: 5. June 29, 2022. arXiv: 2110.02642[cs]. URL: http://arxiv.org/abs/2110.02642 (visited on 10/19/2022) (cit. on pp. 13, 20, 31).

[28] Fabrizio Angiulli and Clara Pizzuti. «Fast Outlier Detection in High Dimensional Spaces». In: *Principles of Data Mining and Knowledge Discovery*. Ed. by Tapio Elomaa, Heikki Mannila, and Hannu Toivonen. Red. by Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Jaime G. Carbonell, and Jörg Siekmann. Vol. 2431. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 15–27. DOI: 10.1007/3-540-45681-3_2. URL: http://link.springer.com/10.1007/3-540-45681-3_2 (visited on 12/07/2022) (cit. on p. 14).

[29] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. «LOF: Identifying Density-Based Local Outliers». In: (2000), p. 12 (cit. on p. 14).

[30] Songqiao Han, Xiyang Hu, Hailiang Huang, Mingqi Jiang, and Yue Zhao. «ADBench: Anomaly Detection Benchmark». In: (2022) (cit. on pp. 15, 20).

[31] Alexander Lavin and Subutai Ahmad. «Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark». en. In: *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. Miami, FL, USA: IEEE, Dec. 2015, pp. 38–44. ISBN: 978-1-5090-0287-0. DOI: `10.1109/ICMLA.2015.141`. URL: `http://ieeexplore.ieee.org/document/7424283/` (visited on 02/17/2023) (cit. on p. 33).

[32] *Dealing with artifacts¶*. URL: `https://github.com/numenta/NAB` (cit. on p. 36).

[33] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. «Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network». en. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, July 2019, pp. 2828–2837. ISBN: 978-1-4503-6201-6. DOI: `10.1145/3292500.3330672`. URL: `https://dl.acm.org/doi/10.1145/3292500.3330672` (visited on 10/18/2022) (cit. on p. 37).

[34] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. «Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding». In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. New York, NY, USA: Association for Computing Machinery, July

2018, pp. 387–395. ISBN: 978-1-4503-5552-0. DOI: `10.1145/3219819.3219845`. URL: `https://doi.org/10.1145/3219819.3219845` (visited on 02/17/2023) (cit. on p. 38).

[35] *Singapore University of Technology and Design.* URL: `https://itrust.sutd.edu.sg/` (cit. on p. 40).