USAD : UnSupervised Anomaly Detection on Multivariate Time Series

Julien Audibert
julien.audibert@orange.com
EURECOM
Biot, France
Orange
Sophia Antipolis, France

Pietro Michiardi
pietro.michiardi@eurecom.fr
EURECOM
Biot, France

Frédéric Guyard
frederic.guyard@orange.com
Orange Labs
Sophia Antipolis, France

Sébastien Marti
sebastien.marti@orange.com
Orange
Sophia Antipolis, France

Maria A. Zuluaga
zuluaga@eurecom.fr
EURECOM
Biot, France

ABSTRACT

The automatic supervision of IT systems is a current challenge at Orange. Given the size and complexity reached by its IT operations, the number of sensors needed to obtain measurements over time, used to infer normal and abnormal behaviors, has increased dramatically making traditional expert-based supervision methods slow or prone to errors. In this paper, we propose a fast and stable method called UnSupervised Anomaly Detection for multivariate time series (USAD) based on adversely trained autoencoders. Its autoencoder architecture makes it capable of learning in an unsupervised way. The use of adversarial training and its architecture allows it to isolate anomalies while providing fast training. We study the properties of our methods through experiments on five public datasets, thus demonstrating its robustness, training speed and high anomaly detection performance. Through a feasibility study using Orange's proprietary data we have been able to validate Orange's requirements on scalability, stability, robustness, training speed and high performance.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Anomaly detection**; **Unsupervised learning**; • **Applied computing**;

KEYWORDS

Anomaly detection; Multivariate Time Series; Neural networks; Autoencoders; Adversarial Network; Unsupervised learning; Supervision

ACM Reference Format:
Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. 2020. USAD : UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403392

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08.
https://doi.org/10.1145/3394486.3403392

1 INTRODUCTION

IT system monitoring is a supervision process on measurable events and outputs of a system, which is used as a reference specifying the system's proper functioning. Deviations from the reference are analyzed to determine if there exists a fault. Historically, this analysis has been done by system monitoring experts who establish *normal behavior* thresholds for every measured event/output. If a measurement exceeds its associated expert-defined threshold, it is considered that the system is not behaving as expected. Because of the size and complexity of today's IT operations at Orange, the number of sensors needed to obtain measurements over time has increased dramatically making traditional expert-defined threshold-based methods no longer usable as they are not scalable. Under this scenario, the automation of our IT system monitoring has become a necessity. Automated IT system monitoring required the development of methods that observe the different measurements acquired by the sensors and, from these, infer normal and abnormal behaviors.

Detecting unexpected behavior on a set of measurements correlated with each other over time is an active research discipline called anomaly detection in multivariate time series [2]. In the past years, many approaches have been developed to address this issue. The most commonly used techniques include distance-based techniques such as k-nearest neighbors [3],clustering such as K-means[9], classification with One-Class SVM [11]. However, today's IT systems have reached a complexity that no longer allows the use of these methods. Indeed, as the number of dimensions increases, these techniques generally suffer from sub-optimal performance due to the curse of dimensionality. Most recently, the ability of unsupervised anomaly detection methods based on deep learning to infer correlations between time series which allow identifying anomalous behaviors has received a lot of attention [12][20][17][18].

Among deep learning methods for detecting anomalies on temporal data, methods based on recurrent neural networks [7] (RNNs) are very popular. However, RNNs results are well-known for being computationally hungry and requiring a significant amount of time to be train. Thus, RNNs incur in high costs associated to time,

energy consumption and CO_2 emissions. For Orange, the use of highly scalable, but also low energy consuming methods is a key issue. Indeed, Orange is constantly pursuing its efforts to improve energy efficiency through its"Green IT&Networks" program. These constraints of high scalability and movement towards GreenAI [16], oblige us to rethink the important characteristics of the deep learning methods to be put in place. Therefore, it is desirable to develop implement methods with high algorithmic efficiency.

Other deep learning-based methods that have been of great interest recently are those based on generating adversary networks [5]. However, GAN training is not always easy, due to problems such as mode collapse and non-convergence [1]. The lack of stability is a major obstacle when considering to implement and deploy these methods into production at Orange. A production environment requires the development of robust methods that can be re-trained routinely.

In this paper, we propose a new method called UnSupervised Anomaly Detection for multivariate time series (USAD) based on an autoencoder architecture [15] whose learning is inspired by GANs. The intuition behind USAD is that the adversarial training of its encoder-decoder architecture allows it to learn how to amplify the reconstruction error of inputs containing anomalies, while gaining stability compared to methods based on GANs architectures. Its architecture makes it fast to trained meeting Orange's expectations in terms of scalability and algorithm efficiency. The main contributions of this paper are:

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

The rest of this document is organized as follows. Section 2 discusses methods for detecting unsupervised anomalies in multivariate time series. Section 3 discusses the details of our method. Sections 4 and 5 describe the experiments and demonstrate the state-of-the-art performance of our method.

2 RELATED WORK

Anomaly detection for time series is a complex task that has been largely studied [6]. Among the different taxonomies which have been proposed, methods can be identified as clustering [9], density-based [11], distance-based [3] and isolation-based methods [10].

In addition to traditional methods, the ability of unsupervised anomaly detection methods based on deep learning to infer correlations between time series has recently received much attention [12, 18, 20]. The Deep Autoencoding Gaussian Mixture Model (DAGMM) [21] jointly considers a Deep Autoencoder and a Gaussian Mixture Model to model the density distribution of multidimensional data. The Multi-Scale Convolutional Recursive Encoder-Decoder (MSCRED) [20] jointly considers time dependence, noise robustness and interpretation of anomaly severity. The LSTM-VAE [14] combines the LSTM with a variational autoencoder (VAE) by replacing the feed-forward network in a VAE with a LSTM. The Adversarially Learned Anomaly Detection (ALAD) [19] is based on bi-directional GANs, that derives adversarially learned features for the anomaly detection task. The LSTM-VAE models the time dependence of time series through LSTM networks and obtains a better generalization capability than traditional methods. Most recently, Su *et al* proposed a stochastic recurrent neural network for multivariate time series anomaly detection, the OmniAnomaly, that learns robust multivariate time series' representations with a stochastic variable connection and a planar normalizing flow, and use the reconstruction probabilities to determine anomalies [17]. However, these methods obtain good results at the expense of their training speed. Indeed, none of these methods take into account the training time (i.e. energy consumption) in their performance criteria. This is why it is necessary today for Orange to develop methods with performances equivalent to the state of the art in terms of anomaly detection, while favoring architectures that allow fast and energy-efficient training.

3 METHOD

We first formalize the problem we are addressing in section 3.1. In 3.2 we present the formulation of our method. Finally, in section 3.3 we describe the method's implementation.

3.1 Problem formulation

A univariate time series is a sequence of data points

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

each one being an observation of a process measured at a specific time t. Univariate time series contain a single variable at each time instant, while multivariate time series record more than one variable at a time; we denote multivariate time series as $\mathcal{T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, $\mathbf{x} \in \mathbb{R}^m$. In this work we focus on the more general setting of multivariate time series, as the univariate setting is a particular case of the multivariate one with $m = 1$.

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

To model the dependence between a current time point and previous ones, let us now define W_t, a time window of length K at given time t:

$$W_t = \{\mathbf{x}_{t-K+1}, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_t\}. \quad (1)$$

It is possible to transform the original time series $\mathcal{T}$ into a sequence of windows $\mathcal{W} = \{W_1, \ldots, W_T\}$ to be used as training input. Given a binary variable $y \in \{0, 1\}$, the goal of our anomaly detection problem is to assign to an unseen window $\widehat{W}_t$, $t > T$, a label y_t to indicate a detected anomaly at time t, *i.e.* $y_t = 1$, or not ($y_t = 0$) based on the window's anomaly score. For the sake of simplicity and without loss of generality we will use W to denote a training input window and $\widehat{W}$ to denote an unseen input one.

3.2 Unsupervised Anomaly Detection

An autoencoder (AE) [15] is an unsupervised artificial neural network combining an encoder E and a decoder D. The encoder part takes the input X and maps it into a set of latent variables Z, whereas the decoder maps the latent variables Z back into the input space as a reconstruction R. The difference between the original input vector X and the reconstruction R is called the reconstruction error. Thus, the training objective aims to minimize this error. It is defined as:

$$\mathscr{L}_{AE} = \|X - AE(X)\|_2, \tag{2}$$

where

$$AE(X) = D(Z), \quad Z = E(X)$$

and $\|\cdot\|_2$ denotes the L2-norm.

Autoencoder-based anomaly detection uses the reconstruction error as the anomaly score. Points with a high score are considered to be anomalies. Only samples from normal data are used at training. At inference, the AE will reconstruct normal data very well, while failing to do so with anomaly data which the AE has not encountered. However, if the anomaly is too small, *i.e.* it is relatively close to normal data, the reconstruction error will be small and thus the anomaly will not be detected. This occurs because the AE aims to reconstruct input data as well (as close to normality) as possible. To overcome this problem, the AE should be able to identify if the input data contains no anomaly before doing a good reconstruction.

The possibility for a method to know whether an input sample is normal or not is what characterizes Generative Adversarial Networks (GANs) [5]. A GAN is an unsupervised artificial neural network based on a two-player minimax adversarial game between two networks, which are trained simultaneously. One network, the generator (G), aims to generate realistic data, whereas the second one acts as a discriminator (D) trying to discriminate real data from that one generated by G. The training objective of G is to maximize the probability of D making a mistake, whereas the training objective D is to minimize its classification error.

Similarly to AE-based, GAN-based anomaly detection uses normal data for training. After training the discriminator is used as an anomaly detector. If the input data is different from the learned data distribution, the discriminator considers it as coming from the generator and classifies it as fake, *i.e.* as an anomaly. However, GAN training is not always easy, due to problems such as mode collapse and non-convergence [1], often attributed to the imbalance between the generator and the discriminator.

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

USAD is composed of three elements: an encoder network E and two decoder networks D_1 and D_2. As depicted in Figure 1, the three elements are connected into an architecture composed of two autoencoders AE_1 and AE_2 sharing the same encoder network:

$$AE_1(W) = D_1(E(W)), \quad AE_2(W) = D_2(E(W)) \tag{3}$$

The architecture from Eq. 3 is trained in two phases. First, the two AEs are trained to learn to reconstruct the normal input windows W. Secondly, the two AEs are trained in an adversarial way, where AE_1 will seek to fool AE_2 and AE_2 aims to learn when the data is real (coming directly from W) or reconstructed (coming from AE_1) . Further details are provided in the following.

Phase 1: Autoencoder training. At a first stage, the objective is to train each AE to reproduce the input. Input data W is compressed by encoder E to the latent space Z and then reconstructed by each decoder. According to Eq. 2, the training objectives are :

$$\begin{aligned} \mathscr{L}_{AE_1} &= \|W - AE_1(W)\|_2 \\ \mathscr{L}_{AE_2} &= \|W - AE_2(W)\|_2 \end{aligned} \tag{4}$$

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

$$\min_{AE_1} \max_{AE_2} \|W - AE_2(AE_1(W))\|_2 \tag{5}$$

which account to the following losses

$$\begin{aligned} \mathscr{L}_{AE_1} &= +\|W - AE_2(AE_1(W))\|_2 \\ \mathscr{L}_{AE_2} &= -\|W - AE_2(AE_1(W))\|_2 \end{aligned} \tag{6}$$

Two-phase training. In our architecture, autoencoders have a dual purpose. AE_1 minimizes the reconstruction error of W (phase 1) and minimizes the difference between W an the reconstructed output of AE_2 (phase 2). As AE_1, AE_2 minimizes the reconstruction error of W (phase 1) but, it then maximizes the reconstruction error of the input data reconstructed by AE_1 (phase 2). The dual purpose training objective of each AE is expressed as the combination of Equations 4, 6 in an evolutionary scheme, where the proportion of each part evolves with time:

$$\mathscr{L}_{AE_1} = \frac{1}{n}\|W - AE_1(W)\|_2 + \left(1 - \frac{1}{n}\right)\|W - AE_2(AE_1(W))\|_2 \tag{7}$$

$$\mathscr{L}_{AE_2} = \frac{1}{n}\|W - AE_2(W)\|_2 - \left(1 - \frac{1}{n}\right)\|W - AE_2(AE_1(W))\|_2 \tag{8}$$

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

It is important to remark that AE_2 does not act as a discriminator in the strict sense of GANs, because if its input is the original data, it is the loss from Eq 4 that intervenes. When its input is a reconstruction, the objective from Eq. 5-6 intervenes instead.

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

$$\mathscr{A}(\widehat{W}) = \alpha\|\widehat{W} - AE_1(\widehat{W})\|_2 + \beta\|\widehat{W} - AE_2(AE_1(\widehat{W}))\|_2 \tag{9}$$

where $\alpha + \beta = 1$ and are used to parameterize the trade-off between false positives and true positives. If we α is greater than β, we reduce the number of true positives and reduce the number

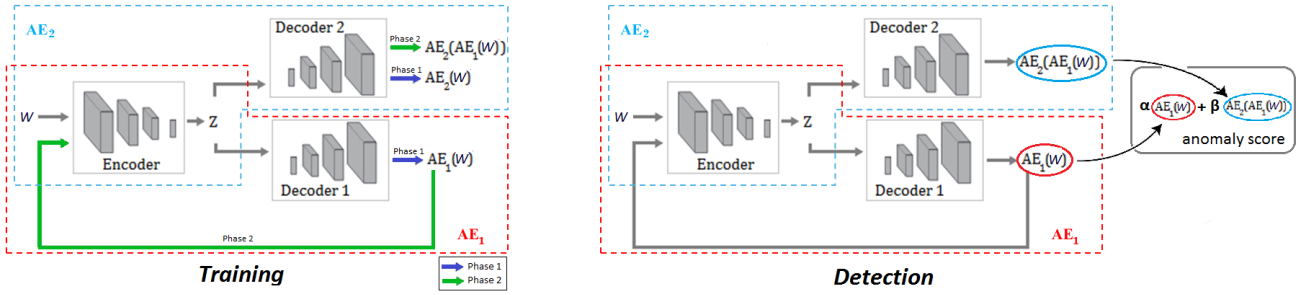

Figure 1: Proposed architecture illustrating the information flow at training (left) and detection stage (right).

Algorithm 1 USAD training algorithm

Input: Normal windows Dataset $\mathcal{W} = \{W_1, ..., W_T\}$, Number epochs N
Output: Trained AE_1, AE_2

$E, D_1, D_2 \leftarrow$ initialize weights
$n \leftarrow 1$
repeat
  for $t = 1$ to T **do**
    $Z_t \leftarrow E(W_t)$
    $W_t^{1'} \leftarrow D_1(Z_t)$
    $W_t^{2'} \leftarrow D_2(Z_t)$
    $W_t^{2''} \leftarrow D_2(E(W_t^{1'}))$
    $\mathscr{L}_{AE_1} \leftarrow \frac{1}{n}\left\|W_t - W_t^{1'}\right\|_2 + \left(1 - \frac{1}{n}\right)\left\|W_t - W_t^{2''}\right\|_2$
    $\mathscr{L}_{AE_2} \leftarrow \frac{1}{n}\left\|W_t - W_t^{2'}\right\|_2 - \left(1 - \frac{1}{n}\right)\left\|W_t - W_t^{2''}\right\|_2$
    $E, D_1, D_2 \leftarrow$ update weights using $\mathscr{L}_{AE_1}$ and $\mathscr{L}_{AE_2}$
  end for
  $n \leftarrow n + 1$
until $n = N$

Algorithm 2 USAD Detection algorithm

Input: Test windows Dataset $\widehat{\mathcal{W}} : (\widehat{W}_1, ..., \widehat{W}_{T^*})$, threshold λ , parameters α and β
Output: Labels $\mathbf{y} : \{y_1, ..., y_{T^*}\}$

for $t = 1$ to T^* **do**
  $\widehat{W}_t^{1'} \leftarrow D_1(E(\widehat{W}_t))$
  $\widehat{W}_t^{2''} \leftarrow D_2(E(\widehat{W}_t^{1'}))$
  $\mathscr{A} \leftarrow \alpha\left\|\widehat{W}_t - \widehat{W}_t^{1'}\right\|_2 + \beta\left\|\widehat{W}_t - \widehat{W}_t^{2''}\right\|_2$
  if $\mathscr{A} \geq \lambda$ **then**
    $y_t \leftarrow 1$
  else
    $y_t \leftarrow 0$
  end if
end for

of false positives. Conversely, if we take an α less than β, we increase the number of true positives at the cost of also increasing the number of false positives. We denote $\alpha < \beta$ a high detection sensitivity scenario and $\alpha > \beta$ a low detection sensitivity one. This parametrization scheme is of great industrial interest. It allows, using a single trained model, to obtain during the inference a set of different sensitivity anomaly scores. This is further illustrated in Section 5.2.

3.3 Implementation

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

Table 1: Benchmarked Datasets. (%) is the percentage of anomalous data points in the dataset.

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

4 EXPERIMENTAL SETUP

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

4.1 Public Datasets

Five publicly available datasets were used in our experiments. Table 1 summarizes the datasets characteristics and they are briefly described in the following.

Secure Water Treatment (SWaT) Dataset. The SWaT dataset [1] is a scaled down version of a real-world industrial water treatment plant producing filtered water [4]. The collected dataset [13] consists of 11 days of continuous operation: 7 days collected under normal operations and 4 days collected with attack scenarios.

Water Distribution (WADI) Dataset. This dataset [2] is collected from the WADI testbed, an extension of the SWaT tesbed [13]. It consists of 16 days of continuous operation, of which 14 days were collected under normal operation and 2 days with attack scenarios.

Server Machine Dataset. SMD is a new 5-week-long dataset from a large Internet company collected and made publicly available [3] [17]. It contains data from 28 server machines each one monitored by $m = 33$ metrics. SMD is divided into two subsets of equal size: the first half is the training set and the second half is the testing set.

Soil Moisture Active Passive (SMAP) satellite and Mars Science Laboratory (MSL) rover Datasets. SMAP and MSL are two real-world public datasets, expert-labeled datasets from NASA [8]. They contain respectively the data of 55/27 entities each monitored by $m = 25/55$ metrics.

4.2 Feasibility study: Orange's dataset

Our feasibility study was performed on an internal dataset, collected specifically for this purpose. The collected data come from technical and business indicators from Orange's advertising network in its website. The data represent a total of $m = 33$ continuous variables including 27 technical and 6 business measurements. The dataset is divided into two subsets: a train set corresponding to about 32 days and a test set corresponding to about 60 days of activity. We have selected 60 days of testing corresponding to a critical period for Orange. To obtain our training set, we selected the previous consecutive days without any major incidents for the company. We were able to obtain a training set of 32 mainly normal days. Anomalies in the test set were labeled by domain experts based on incident reports. Its main characteristics are reported in Table 1.

4.3 Evaluation Metrics

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

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

with TP the True Positives, FP the False Positives, and FN the False negatives. We consider a window is labeled as an anomaly as soon as one of the points it contains is detected as anomalous.

In [17], the authors compute the F1 score using the average precision and average recall. For the sake of completeness, we report this measure when comparing our method to their benchmark. We denote this measure the F1* score:

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

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

Performance is assessed by comparing the results of each evaluated method with the annotated ground truth. To allow a direct comparison with the benchmark proposed by [17] we use their approach. Anomalous observations usually occur in the form of contiguous anomaly segments. In this approach, if at least one observation of an anomalous segment is correctly detected, all the other observation of the segment are also considered as correctly detected, even if they were not. The observations outside the ground truth anomaly segment are treated as usual. We denote this approach *point-adjust*. We also assess performance without *point-adjust* on the two datasets (SWaT and WADI) not belonging to the benchmark [17].

5 EXPERIMENTS AND RESULTS

We study the key properties of USAD by assessing its performance and comparing it to other state of the art methods (5.1), analyzing how different parameters affect the performance of the method (5.2), estimating its computational performance (5.3 and through an ablation study where, at each time, we suppress one of the training phases (5.4). Finally, in Section 5.5 we report a feasibility study using Orange's internal data to demonstrate that USAD meets the requirements needed to be deployed in production.

5.1 Overall performance

To demonstrate the overall performance of USAD we compare it with five unsupervised methods for the detection of multivariate time series anomalies. These are: Isolation Forests (IF) [10], autoencoders (AE), LSTM-VAE [14], DAGMM [21], OmniAnomaly [17]. As not all of the anomaly detection methods used for comparison provide a mechanism to select anomaly thresholds, we tested possible anomaly thresholds for every model and report the results linked to the highest F1 score. Table 2 details the obtained performance results for all methods on the public datasets. On top, the results obtained with SWaT and WADI datasets are presented, whereas the bottom part of the table reports obtained results from the benchmark proposed by [17], using three remaining datasets. USAD outperforms all methods on SWaT, MSL, SMAP and WADI without *point-adjust* datasets, and its F1 is the second best on the SMD dataset. On average over all datasets (Table 3) is the best performing method exceeding by 0.096 the current state-of-the-art [17].

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

Despite the relative poor results in most datasets, IF achieves the highest F1 score with *point-adjust* on WADI. This is explained by the natures of the *point-adjust* method and the WADI dataset. IF considers each observation/time-point independently and assigns a label to a single time-point and not to a window. WADI's anomalies lasting in time, the point-adjust validates the entirety

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

Table 2: Performance comparison. Top: precision (P), recall (R) and F1 score with and without point-adjust (Without) in SWaT and WADI datasets. Bottom: Using the benchmark proposed by [17] with point ajust. P, R F1, and F1* are reported.

Methods	SWaT						WADI					
	Without			With			Without			With		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
AE	0.9903	0.6295	0.7697	0.9913	0.7040	0.8233	0.9947	0.1310	0.2315	0.3970	0.3220	0.3556
IF	0.9512	0.5884	0.7271	0.9620	0.7315	0.8311	0.2992	0.1583	0.2071	0.6241	0.6155	**0.6198**
LSTM-VAE	0.9897	0.6377	0.7756	0.7123	0.9258	0.8051	0.9947	0.1282	0.2271	0.4632	0.3220	0.3799
DAGMM	0.4695	0.6659	0.5507	0.8292	0.7674	0.7971	0.0651	0.9131	0.1216	0.2228	0.1976	0.2094
OmniAnomaly	0.9825	0.6497	0.7822	0.7223	0.9832	0.8328	0.9947	0.1298	0.2296	0.2652	0.9799	0.4174
USAD	0.9851	0.6618	**0.7917**	0.9870	0.7402	**0.8460**	0.9947	0.1318	**0.2328**	0.6451	0.3220	0.4296

Methods	SMD				SMAP				MSL			
	P	R	F1	F1*	P	R	F1	F1*	P	R	F1	F1*
AE	0.8825	0.8037	0.8280	0.8413	0.7216	0.9795	0.7776	0.8310	0.8535	0.9748	0.8792	0.9101
IF	0.5938	0.8532	0.5866	0.7003	0.4423	0.5105	0.4671	0.4739	0.5681	0.6740	0.5984	0.6166
LSTM-VAE	0.8698	0.7879	0.8083	0.8268	0.7164	0.9875	0.7555	0.8304	0.8599	0.9756	0.8537	0.9141
DAGMM	0.6730	0.8450	0.7231	0.7493	0.6334	0.9984	0.7124	0.7751	0.7562	0.9803	0.8112	0.8537
OmniAnomaly	0.9809	0.9438	**0.9441**	**0.9620**	0.7585	0.9756	0.8054	0.8535	0.9140	0.8891	0.8952	0.9014
USAD	0.9314	0.9617	0.9382	0.9463	0.7697	0.9831	**0.8186**	**0.8634**	0.8810	0.9786	**0.9109**	**0.9272**

Table 3: Average performance (± standard deviation) over all datasets using point-adjust.

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

of an anomaly as being well detected. Thus IF is little impacted by its bad predictions (FPs) affecting only one observation at a time, compared to the advantage obtained with the *point-adjust* which validates whole segments of good prediction despite having potentially missed several abnormalities.

Differently, AE, LSTM-VAE, use sequential observations as input allowing the two methods to retain temporal information. These methods perform the best possible reconstruction regardless of the existence of an anomaly in the input window. This does not allow them to detect anomalies close to the normal data. USAD compensates for this drawback of AE-based methods through its adversarial training. A similar situation occurs with OmniAnomaly, as it does not have a mechanism that allows to amplify "mild" anomalies.

5.2 Effect of parameters

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

The first factor we study is how USAD responds to different down-sampling rates of the training data. Down-sampling speeds up learning by reducing the size of the data and also has a denoising effect. However, it can have a negative effect if too much information is lost. Figure 2(A) summarizes the obtained results using 5 different rates $[1, 5, 10, 20, 50]$. Results show that USAD's performance is relatively insensitive to down-sampling, with a relatively constant performance across sampling rates. This indicates that the choice of the down-sampling rate is not critical to the method. For our experiments, we selected a rate of 5. This is the best trade-off between denoising the training data and limiting the loss of information. Moreover, it allows to reduce by 5 the training time needed for USAD.

The second factor we investigate is how USAD responds to different window sizes in the data. The window size has an impact on the type of abnormal behaviors that can be detected a direct impact on the speed of anomaly detection since the speed of detection is defined by the duration of a window. Figure 2(B) presents the obtained results for five different window sizes $K \in [5, 10, 20, 50, 100]$. The best result was achieved for window size $K = 10$. USAD can detect behavior changes faster when the window is smaller since each observation has a greater impact on the anomaly score. A window that is too large will have to wait for more observations to detect an anomaly. However, a larger window will detect longer anomalies. If an anomaly is however too short, it may be hidden in the number of points that a too-large window has. For Orange, a small window is better since it allows both faster training and faster detection.

The latent variables Z sit in a $m-$dimensional space, which is assumed to be smaller than one of the original data. We study the role of m in the performance of USAD. Figure 2(C) presents the results for $m \in [5, 10, 20, 40, 100]$. Results show that a very small dimension for Z causes a large loss of information at the encoding

stage that the decoder is not then able to recover, thus leading to a poor performance. On the other extreme, using a large value for m results in memorization of the training data causing and a drop in performance. Instead, mid-range values of m do not seem to have a strong effect in the performance, showing both relatively high and stable F1 scores.

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

Figure 2(D) shows the performance of our method, in terms of P, R and F1, as the level of noise increases. USAD demonstrates its robustness with a relatively constant, high performance for noise levels of up to 5%. When the training set noise is of 10% a slight drop in the performance starts to be observed. However, the overall performance, measured by the F1 score, remains good. Interestingly, this performance drop is caused by a lower precision. As the recall remains relatively constant, this implies that with higher noise in the training set the method begins to be more prone to detect false positives. This behavior suggests that as the noise starts to increase, USAD is no longer able to properly learn the most complex behaviors existing within the training set. As a result, the number of false positives increases in the test set, since USAD detects complex normal behaviors as anomalies. Finally, a significant drop in performance can be observed for high noise levels (30%). However, such a high anomaly rate during training in a production environment is not realistic. This means that for a given period of time, 30% of the samples are unnoticed anomalies. As there are so many anomalies in production, it is not realistic that such a large number of incidents are missed by Orange's incident supervision. Thus, it is unlikely that USAD will be confronted with such a high rate of anomalies during its training in a production environment at Orange.

Finally, we study the role of the sensitivity threshold (equation 9). A large α corresponds to giving more importance to the reconstruction of the AE_1 autoencoder in the anomaly score, while a large β corresponds to giving more importance to the reconstruction of the AE_2 autoencoder (see Figure 1). The possibility to tune the detection sensitivity without having to re-train the model is of great importance for Orange. Table 4 reports the effect of varying α, β in the number of detected FPs, TPs and the F1 score.

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

Table 4: Anomaly detection results with various sensitivity thresholds for SWaT dataset

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

Table 5: Training Time (min) per epoch on each dataset

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

5.3 Training time

In this section we study the computational performance of USAD and we compare it to OmniAnomaly, the method offering the closest performance in anomaly detection (see Table 3). To do this, we measured the average time taken per epoch on the 5 public data sets. The reference time for SMD, SMAP and MSL is the average time for one epoch over all entities (*i.e.* 28 machines of the SMD, 55 of the SMAP and 27 of the MSL). Both methods were trained using a NVIDIA GeForce GTX 1080 Ti.

Table 5 presents the obtained results. USAD provides good performance in unsupervised anomaly detection over multivariate time series while reducing training time by an average of 547 times.

5.4 Ablation Study

Using SMD, SMAP and MSL datasets, we investigate the effects of the two-phase training of USAD. Figure 3 presents a performance comparison in terms of the F1-score using USAD (Combined), USAD with only phase one training (Autoencoders) and with only phase 2 training (Adversarial). Training USAD without adversarial learning accounts to using the objective presented in equation 4, whereas suppressing the autoencoder accounts to use the objective from Equations 5-6.

GAN-inspired adversarial training represents an increase in performance of 5.88% (F1 score) with respect to the second best option which is USAD without adversarial training and 24.09% with respect to using only adversarial training. This can be explained by the amplified reconstruction error effect introduced by USAD regardless of the presence or not of an anomaly in the input window. Thus, USAD without its adversarial training cannot detect the anomalies closest to the normal data. USAD's poor performance with only adversarial training is explained by the fact that the method does not have the autoencoder training to orientate the weights in a favorable place before starting phase 2 of adversarial training. In

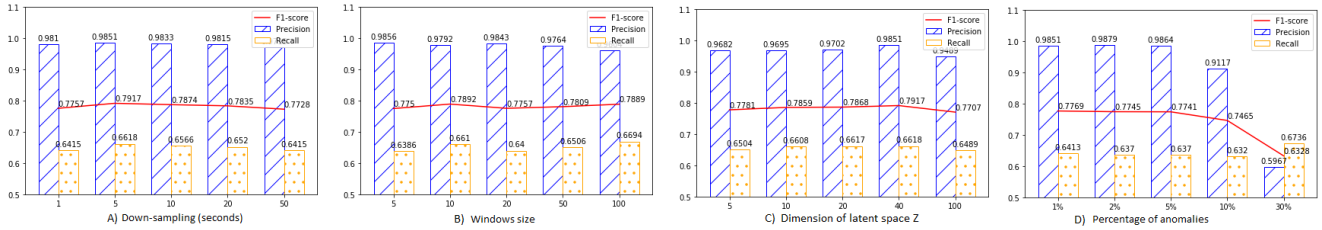

Figure 2: Effect of parameters. Precision, Recall and F1-score as a function of A) the training set's down-sampling rate, B) the window size K, C) the dimension of the latent space Z and D) the percentage of anomalies in the training set

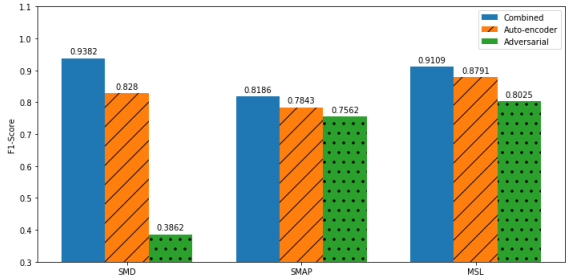

Figure 3: Impact with and without adversarial training on USAD

Table 6: Anomaly detection results on Orange internal Dataset (without point-adjust)

Method	Precision	Recall	F1-score
USAD	0.7448	0.6428	0.6901

conclusion, ablation of any of the training phases leads to poorer performance. For instance, both ablated versions of USAD have a lower F1 score than that of several of the bench-marked methods (Table 2, bottom).

5.5 Feasibility study

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

Table 6 reports the results obtained in the internal dataset. USAD was able to detect all significant incidents in less than 30 minutes over the two months length of test data. For example, USAD was able to detect in less than 30 minutes an incident that took 24 hours to be detected by the operators in charge of supervision at Orange (Figure 4). This incident was caused by an error introduced in the configuration files allowing to assign advertising displays to unexpected partners. This caused the number of advertising displays (TOTAL IMPRESSIONS) to increase, while reducing the average display prices (TOTAL AVERAGE ECPM). As a result, important business indicators such as the revenue (TOTAL CPM CPC REVENUE) remained stable and so, the operators were unable to detect the incident quickly. Faced with the large amount of indicators to survey, people in charge of supervision concentrated their efforts on supervising indicators with high business impact, therefore, explaining, the 24 hours needed to detect this configuration incident.

6 CONCLUSIONS

In this paper, we propose USAD, an UnSupervised Anomaly Detection for multivariate time series method based on autoencoders and trained within an adversarial training inspired by the Generative Adversarial Networks. Its autoencoder architecture makes it an unsupervised method and allows it to show great stability during the adversarial training. We used a set of five public reference datasets to study the desired properties of USAD. The method demonstrated superior performance over state-of-the-art techniques on public reference datasets in terms of standard F1-score. In addition, its demonstrated fast training, robustness to the choice of parameters and stability allows for high scalability of the model within an industrial setting. USAD also provides the possibility to parameterize its sensitivity and to produce, from a single model, a set of detection levels. This possibility offers Orange's supervision teams essential functionalities enabling the use of the method in production on large-scale infrastructure. Since the teams need to be able to lower the sensitivity of the detection to prevent only major incidents when their workload becomes too high, the ability to multiply detection sensitivities during inference makes the model extremely scalable within the company and brings major advantages. First of all, it allows us to limit the time needed to train the supervision models by limiting their number to just one. Secondly, a deep learning model put into production must be monitored and supervised by teams. Limiting the number of models allows us to reduce the time spent supervising models in production and therefore free up time from supervisors to be devoted to different tasks.

The feasibility study performed using Orange's internal data provided conclusive results which confirm that USAD suggests a promising direction for the automation of IT systems supervision at Orange. It also signaled some of the difficulties that might be encountered on the way to deployment and execution. For example, during the data collection process (Section 4.2) we were faced with the unexpected difficulty of gathering a continuous training period not containing too many anomalies. This is an interesting aspect that makes us think on the infrastructure that will need to be put in place to have USAD succesfully deployed.

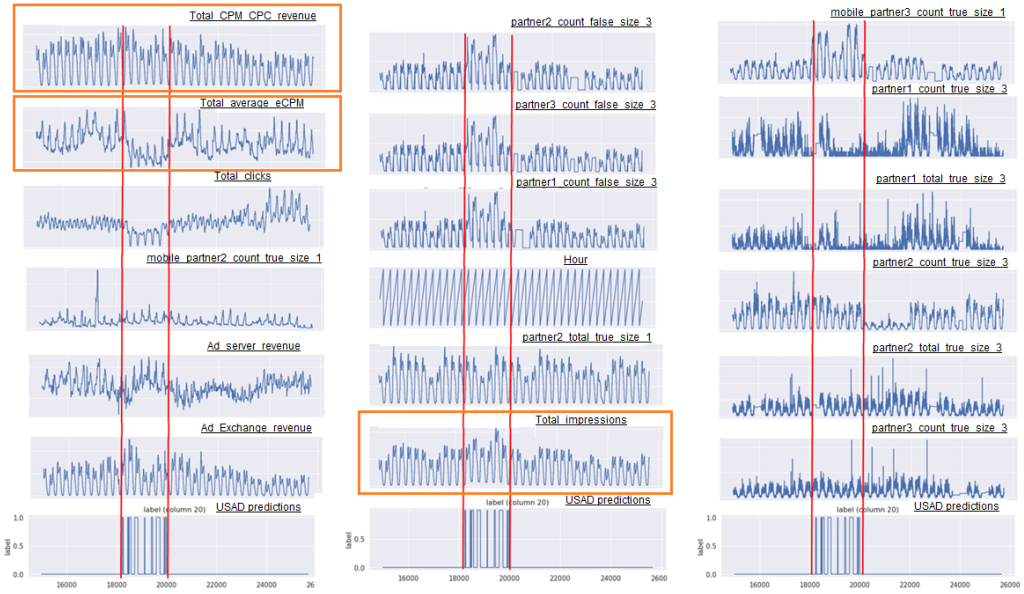

Figure 4: Example of a time series from the feasibility study where a configuration incident was detected by USAD. Twenty-four out of the 33 time variables are shown. The orange boxes highlight the variables referred to. In orange, the series referred to in section 5.5.

REFERENCES

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

[2] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep Learning for Anomaly Detection: A Survey. *CoRR* abs/1901.03407 (2019).

[3] Wanpracha Art Chaovalitwongse, Ya-Ju Fan, and Rajesh C. Sachdeo. 2007. On the Time Series K-Nearest Neighbor Classification of Abnormal Brain Activity. *IEEE Trans. Systems, Man, and Cybernetics, Part A* 37, 6 (2007), 1005–1016.

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

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

[6] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. 2013. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (2013), 2250–2267.

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

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

[9] Istvan Kiss, Bela Genge, Piroska Haller, and Gheorghe Sebestyen. 2014. Data clustering-based anomaly detection in industrial control systems. In *Proceedings - 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing, ICCP 2014*. Cluj-Napoca, Romania, 275–281.

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

[11] Junshui Ma and Simon Perkins. 2003. Time-series Novelty Detection Using One-class Support Vector Machines. *Proceedings of the International Joint Conference on Neural Networks* 3 (2003), 1741–1745.

[12] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. (2016). arXiv.1607.00148

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

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

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

[16] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. (Jul 2019). arXiv.1907.10597

[17] Ya Su, Rong Liu, Youjian Zhao, Wei Sun, Chenhao Niu, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 1485 (2019), 2828–2837.

[18] Ye Yuan, Guangxu Xun, Fenglong Ma, Yaqing Wang, Nan Du, Kebin Jia, Lu Su, and Aidong Zhang. 2018. MuVAN: A Multi-view Attention Network for Multivariate Temporal Data. *Proceedings - IEEE International Conference on Data Mining, ICDM* 2018-November (2018), 717–726.

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

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

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

A SUPPLEMENTARY MATERIAL FOR REPRODUCIBILITY

A.1 Experimental Setting

All experiments are performed on a machine equipped with an Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz and 270 GB RAM, in a docker container running CentOS 7 version 3.10.0 with access to an NVIDIA GeForce GTX 1080 Ti 11GBGDDR5X GPU. The Isolation Forest (IF) comes from the scikit-learn [4] implementation. The DAGMM comes from a Tensorflow implementation on Github [5]. The LSTM-VAE comes from a Github implementation [6]. The OmniAnomaly comes from the authors' Tensorflow implementation of Github [7]. Finally, the USAD and AE were developed by us in Pytorch.

A.2 Packages Used in Our Implementation

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

- python==3.6.8
- pytorch==1.3.1
- cuda==10.0
- scikit-learn==0.20.2
- numpy==1.15.4

A.3 USAD Hyper-parameters for each dataset

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

Table 7: USAD Hyper-parameters for each dataset. K denotes the window size and m the dimension of the latent space.

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

A.4 USAD Implementation

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

A.4.1 Encoder.

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

A.4.2 Decoder. Both decoders have the same architecture.

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

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

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html
[5] https://github.com/tnakae/DAGMM
[6] https://github.com/Danyleb/Variational-Lstm-Autoencoder
[7] https://github.com/NetManAIOps/OmniAnomaly