



ML Models for Detecting QoE Degradation in Low-Latency Applications: A Cloud-Gaming Case Study

Joël Roman Ky, Bertrand Mathieu, Abdelkader Lahmadi, Raouf Boutaba

► To cite this version:

Joël Roman Ky, Bertrand Mathieu, Abdelkader Lahmadi, Raouf Boutaba. ML Models for Detecting QoE Degradation in Low-Latency Applications: A Cloud-Gaming Case Study. IEEE Transactions on Network and Service Management, 2023, pp.1-1. 10.1109/TNSM.2023.3293806 . hal-04160235

HAL Id: hal-04160235

<https://hal.science/hal-04160235>

Submitted on 12 Jul 2023





HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

ML Models for Detecting QoE Degradation in Low-Latency Applications: A Cloud-Gaming Case Study

Joël Roman Ky , Bertrand Mathieu , *Senior Member, IEEE*, Abdelkader Lahmadi 
and Raouf Boutaba, , *Fellow, IEEE*

Abstract—Detecting abnormal network events is an important activity of Internet Service Providers particularly when running critical applications (e.g., ultra low-latency applications in mobile wireless networks). Abnormal events can stress the infrastructure and lead to severe degradation of user experience. Machine Learning (ML) models have demonstrated their relevance in many tasks including Anomaly Detection (AD). While promising remarkable performance compared to manual or threshold-based detection, applying ML-based AD methods is challenging for operators due to the proliferation of ML models and the lack of well-established methodology and metrics to evaluate them and select the most appropriate one.

This paper presents a comprehensive evaluation of eight unsupervised ML models selected from different classes of ML algorithms and applied to AD in the context of cloud gaming applications. We collect cloud gaming Key Performance Indicators (KPIs) time-series datasets in real-world network conditions, and we evaluate and compare the selected ML models using the same methodology, and assess their robustness to data contamination, their efficiency and computational complexity. In addition to the traditional F1-score performance metric used in anomaly detection, we use Matthews Coefficient Correlation (MCC) to better differentiate between models' efficiencies. Our proposed methodology relies on window-based anomaly detection techniques as they are more useful for network operators compared to single point detection approaches. However, we found most existing window-based approaches to lack in accuracy and may under or over-estimate a model's performance. Therefore, in this paper, we propose a novel Window Anomaly Decision (WAD) approach that overcomes these drawbacks. We leverage our experimental results to provide insights about the most relevant models for detecting QoE degradation and offer recommendations on their suitability for different application requirements.

Index Terms—low-latency, anomaly detection, unsupervised learning, QoE, 4G networks, metrics

I. INTRODUCTION

RECENT years have witnessed the deployment of high performance networks, e.g., FTTH and 5G mobile networks, to support the stringent requirements in terms of latency, bandwidth, reliability, and jitter of emerging applications. For instance, remote surgery requires reliability and low-latency for video streaming and control feedback; AR/VR applications and the Metaverse require high-bandwidth and

low-latency to allow human interaction with the environment. Cloud Gaming (CG) applications also face challenges in terms of delay and bandwidth consumption which are hardly met by current cellular network architectures, exposing CG users to network impairments that deteriorate their Quality of Experience (QoE). Internet Service Providers (ISPs) need efficient monitoring techniques to detect QoE deterioration that their customers may experience.

A primary approach widely adopted in networking to detect users' QoE degradation is to rely on anomaly detection (AD) methods which are either applied manually or rely on rule-based techniques [1]. However, the increasing complexity of network infrastructures tend to make these techniques impractical and inefficient. The recent advances in Machine Learning (ML) have been leveraged in various domains, including networking where large amounts of data are available and could be used to build classification and prediction models. The popularity of ML is mainly due to the success of supervised learning which requires a plethora of labeled data during the training phase. However, data labeling has to be done by domain experts and has proven to be a long and tedious process. To circumvent the need for labeled data, unsupervised learning techniques are increasingly adopted, in particular for anomaly detection.

Many anomaly detection models based on ML techniques have been proposed in the literature, including distance-based algorithms [2], [3], predictive algorithms, reconstruction-based algorithms [4], [5], one-class algorithms [6], [7], generative algorithms [8], etc. The performance of these models is usually evaluated using the F1-score as the key performance metric. The majority of these studies also used a *point-wise* approach to compute the F1-score, i.e., they consider only anomalous observations. The issue with the *point-wise* approach is that it disregards the fact that degradations can occur in the form of consecutive anomalous observations. To address this limitation, new approaches [9], [10], [11] are proposed to better evaluate the performance of the AD models. These approaches consider degradations as *windows* of anomalies, aggregate the predictions following different criteria and compute the F1-score accordingly. Another issue presented by many of existing ML models is the difficulty in comparing them for a given application since each model is evaluated using a different methodology, benchmark datasets and evaluation metrics. Each method reports higher F1-score than its competitors making it tricky for domain experts to

Joël Roman Ky and Bertrand Mathieu are with Orange Innovation, Lannion, France (email: joelroman.ky@orange.com, bertrand2.mathieu@orange.com).

Abdelkader Lahmadi is with Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France (email: abdelkader.lahmadi@loria.fr).

Raouf Boutaba is with David R. Cheriton School of Computer Science, University of Waterloo (email: rboutaba@uwaterloo.ca).

choose the most appropriate approach that fits their needs.

In this paper, we present a comprehensive comparison and analysis on the performance of unsupervised ML models through experiments while relying on a consistent evaluation methodology. In particular, we focus on the detection of QoE degradation in low-latency applications using a real-world multivariate time-series dataset of Key Performance Indicators in CG sessions collected under different 4G network conditions. The game sessions are recorded using the public Google Stadia CG platform. We objectively evaluate the ability of each model to detect anomalies that can lead to QoE degradation for CG users. The experiments conducted under 4G conditions and the conclusions drawn from them remain applicable to 5G networks, as the Quality of Experience (QoE) degradation we aim to detect are related to features collected at the CG application-level. In addition, we provide insights and offer recommendations to network operators on the most appropriate AD model that meets their application requirements for anomaly detection, including real-time or offline inference, detection accuracy, robustness to data contamination rate due to different environments, etc.

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

- We demonstrate, based on synthetic models, the limitations of existing *window* approaches for evaluating the performance of AD models and then propose our Window Anomaly Decision (WAD) approach.
- We perform an exhaustive evaluation of ML models to assess (i) their robustness by injecting anomalies in their training sets (i.e., data contamination), and (ii) their capability to detect short and longer users' QoE degradation by varying the windows size.
- Based on our experiments, we offer recommendations to network operators and network management experts on the best models for different application requirements.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes our methodology for anomaly detection and exposes the proposed WAD approach. Section IV compares the performance of the ML models and Section V discusses which model is best depending on the use case scenario.

II. BACKGROUND AND RELATED WORK

In this section, we first discuss CG applications and the impact of latency on their performance. Then, we review existing unsupervised learning models and window-based approaches used for anomaly detection using time series data.

A. Cloud Gaming applications

Cloud games are processed and executed on cloud servers and the rendered scenes are streamed over the Internet to client devices removing the need for having powerful gaming computers or consoles [12]. Several works [13], [14], [15] focused on studying the behavior of CG platforms and analysed their respective network protocols. For instance, Google Stadia uses WebRTC for its service and relies on RTP to stream its audio and video contents. Marchal et al. [13] showed that link

capacity and latency are driving Stadia platform to adapt its bitrate and resolution.

On the other hand, several works [16], [17], [18], [19] addressed the impact of network latency on the performance of gamers in cloud and non-cloud games. They showed that the performance of gamers drops with the increase of latency. Raaen et al. [17] and Vlahovic et al. [19] also include in their subjective studies that most of the gamers could not tolerate a delay above 100ms and the most demanding gamers are able to perceive delays below 40ms. These network perturbations can lead to QoE degradation, which this work aims to detect using unsupervised ML algorithms.

B. Unsupervised Learning models for anomaly detection

Anomaly detection is a popular and a well-covered research topic with numerous surveys [20], [21], [22], [23] proposing different taxonomies and categories of techniques, including existing machine learning models for anomaly detection. Some studies [24], [25], [26], [27] have specifically focused on the use of deep-learning for anomaly detection due to its efficiency in handling multivariate and high-dimensional data. Schmidl et al. [21] categorized anomaly detection models for time-series data depending on the way they determine anomalies. Reconstruction-based algorithms detect anomalies by learning a model from *normal* training data. They encode the training features into a low-dimensional (i.e., latent) space and reconstruct the input features from the latent features. An anomaly score is computed by comparing the reconstructed data to the input data in order to detect anomalies. Since the model is built on *normal* data only, anomalous time-series cannot be well-reconstructed and will have a high anomaly score (i.e., above a pre-determined threshold). This class of algorithms, include Principal Component Analysis (PCA) and neural network algorithms such as AutoEncoder, LSTM-VAE [28], DAGMM [4], OmniAnomaly [29], Donut [9], and USAD [5].

One-class classification models detect anomalies by learning a hypersphere that encloses the representation of normal data. Any points that remain outside the learned hypersphere are classified as anomalies. The most popular algorithm from this category is the OC-SVM [6] and its neural network variant Deep-SVDD [7].

Furthermore, it is worth mentioning that there are other families of unsupervised learning models for time-series data, including Isolation methods that build an ensemble of isolation trees to isolate anomalies. Isolation methods such as Isolation Forest [2] assume that anomalies are easy to *isolate* since they are fewer in the data and are starkly different from normal instances. Distance-based methods (e.g., KNN [3], LOF[30]) detect anomalous instances based on their larger distances from normal instances. Predictive methods (e.g., ARIMA) predict time-series sequences and compare them to original time-series to differentiate the anomalies. Generative methods (e.g., GAN [8]) train a model to generate new (i.e., normal or anomalous) data based on real data distribution and a discriminator that learns how to discriminate real data from generated data.

The aforementioned anomaly detection models were evaluated in different studies [31], [32], [23] either with broad or domain-specific benchmark datasets. Our work provides a comprehensive and consistent experimental evaluation of anomaly detection models for QoE degradation detection, while taking into account the impact of data contamination and window size.

C. Window-based approaches for anomaly detection

Chandola et al. [20] categorize anomalies into three classes: *point anomalies* which are individual observations that deviate from normal ranges; *collective anomalies* that represent a group of anomalous observations and *contextual anomalies* that represent a group of observations that can be considered as anomalous in a specific context. Point anomalies are the most widely addressed anomalies in the literature.

In most time-series, anomalies occur as contiguous anomalous observations (i.e., *collective/contextual anomalies*) rather than individual *point anomalies*. Assessing the performance of AD models for time-series data using *point-wise* (PW) approaches is ineffective as it disregards the contiguous nature of anomalies. To overcome this limitation, *Point-adjust* (PA) approach is proposed by Xu et al. [9] and is often employed [29], [5] to deal with windows of anomalies. PA approach considers that all the anomalies of a window are correctly detected by a AD model if any of the anomalous observations in the window is correctly detected. However, it was shown in [33], [34], [35] that PA approach presents limitations as well and may overestimate the performance of an AD model. The authors showed that a well-trained and efficient algorithm provides the same F1-score as a random model with PA approach. Hence, with the PA, it is difficult to conclude that a model outperforms others.

To address the limitations of PA approach, *revised point-adjust* approach (RPA) [11] and *PA%K* [33] were proposed. Unlike PA, RPA is less tolerant to high false-positive rates by severely penalizing them. Other approaches such as Numenta Anomaly Benchmark (NAB) [10] and range-based Precision/Recall [36] have also been proposed, but they are too complex to be widely adopted. In this paper, we propose the WAD approach, which addresses the aforementioned limitations and aims at fairly assessing the performance of anomaly detection models.

III. METHODOLOGY

In this section, we describe our general methodology for evaluating ML models to detect anomalies in cloud gaming sessions. We first formulate the anomaly detection task for anomalous windows in CG sessions and introduce the collected datasets. Our new approach, called Window Anomaly Decision (WAD), proposed to address the limitations of existing window approaches, is then presented. This is followed by an introduction on the unsupervised ML models used in our comparative evaluation.

A. Problem statement

Our goal in this paper is to detect the end-users' QoE degradation in CG sessions. For this, we decided to base our detection analysis using windows of observations, instead of individual points of observation. Indeed, QoE degradation lasting 5ms is not perceptible by human beings for whom the perception of latency is around 150ms [37]. Therefore, we evaluate the ML models with 3 window sizes $p \in \{10, 20, 30\}$ to perform a detection within 50, 100, 150 ms respectively, and have a representative time of perception (i.e., very reactive people to less reactive ones, via mean value).

As such, we can formalize our problem as follows: we denote the time-series of our datasets as $x = \{x_1, x_2, \dots, x_T\}$, where T is the length of x and $x_t \in \mathbb{R}^m$ denotes a m -dimensional vector corresponding to the values of our m features at time t . For the representation of observations in windows, we split x into sequences of windows $W = \{w_1, w_2, \dots, w_{T-p+1}\}$ with stride 1 where $w_t = \{x_t, x_{t+1}, \dots, x_{t+p-1}\}$, p being the window size.

Given an unsupervised anomaly detection model \mathcal{M} , a set of parameters \mathcal{W} is learnt to output an anomaly score $s(\tilde{x}_t)$ for each unseen observation \tilde{x}_t . From this anomaly score and a carefully chosen threshold δ , a binary variable $\tilde{y}_t \in \{0, 1\}$ is assigned for each observation as follows:

$$\tilde{y}_t = \begin{cases} 1, & \text{if } s(\tilde{x}_t) > \delta \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The goal of the window-based detection of anomalies is to correctly map the binary value of each observation within a window to a binary value for the whole window. Since we focus on perceptible end-users' QoE degradation, a large part of the window observations should be anomalous so that the windows is classified as anomalous. In the remainder of the paper, we consider a window as anomalous if at least 80% of observations in the window are anomalous.

B. Data collection

We rely on the datasets used in our previous work [38]¹ which contain time-series features of QoE and QoS collected while playing racing games on public cloud gaming platforms:

- *Dirt 4* for Google Stadia (STD);
- *TrackMania* for Nvidia GeForce Now (GFN);
- *F1 2021* for Microsoft Xbox Cloud (XC).

A set of 14 features (cf. Table I) were collected through the Chromium WebRTC API adapted by the DECAF tool [14]. The measurements were performed under 6 different 4G network conditions, with 5 differing in the average downlink throughput and 1 in a mobility scenario on a highway. More details about the testbed, the network conditions and the datasets are available in [38].

The collected datasets are unlabelled (i.e., with no ground-truths). Consequently, evaluation of the models performance with well-known ML evaluation metrics, such as Precision,

¹Datasets are available as OpenData : https://cloud-gaming-traces.lhs.loria.fr/ANR-19-CE25-0012_std_gfn_xc_cg_webrtc_metrics.7z

TABLE I
DESCRIPTION OF COLLECTED FEATURES

Features	Description
Network RTT	Network RTT computed during game session.
Decoding delay	Delay to decode each video frame.
Jitter buffer delay	Delay between the time, the first packet of a video frame enters the jitter buffer and the time the whole frame exits the jitter buffer.
Video rendering jitter	Time between two consecutive video frames.
Uplink Bitrate	Number of bits-per-second (bps) sent by the client.
Downlink Bitrate	Number of bits-per-second (bps) received by the client.
Frame rate	Frames received per-second (FPS).
Height resolution	Number of pixels in the frame height.
Width resolution	Number of pixels in the frame width.
Freeze	A freeze is count if an inter-frame delay is greater than a value defined as $\max(3 * \text{avgInterFrameDelay}, \text{avgInterFrameDelay} + 150\text{ms})$.
Frames dropped	Number of video frames dropped before decoding step.
Frames decoded	Number of video frames decoded.
Packets received	Number of packets received.
Downlink throughput	Max downlink capacity allowed by the 4G network conditions.

Recall or F1-score is not possible. We hence create ground-truth $(y_t)_{t \in [1, T]}$ (i.e., labels) for evaluation purposes (and intrinsically to split the data), but the labels are not used by ML algorithms during training. According to the CG platform's recommendations for high quality streaming,^{2,3} we define the ground-truths based on the following criteria ($\gamma = 1080$ for STD and XC and $\gamma = 768$ for GFN):

$$y_t = \begin{cases} 1, & \text{if } \begin{cases} \text{resolution}(x_t) < \gamma \text{ or} \\ \text{frameRate}(x_t) < 60 \text{ or} \\ \text{freeze}(x_t) = 1 \end{cases} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

We would like to stress here that the primary objective of this paper is to compare the performance of unsupervised ML models in detecting anomalies on a CG case study. The objective is not to detect degradation using simple rules, but rather to utilize such rules to establish ground-truths required for computing the ML metrics. Although being simple, there rules present a significant challenge for our tasks, particularly when they are unknown a-priori as shown later in the paper our results in Section IV.

Using these ground-truth labels as a reference, we define a ground-truth window as anomalous if at least 80% of the labels of the window are anomalous.

TABLE II
DATASETS SUMMARY

Datasets	Train normal windows	Contamination set windows	Test windows	Test anomalies ratio (%)
STD	80486	59480	169706	52.57
GFN	27415	22667	61417	55.36
XC	83611	17918	110487	24.32

C. Data processing and splitting

The features of the datasets are resampled to have a fixed time-step of 5ms and they are normalized before training. We build the training and testing sets following the splitting strategy proposed by Zong et al. [4] which ensures consistency over different experiments and allow to perform a fair evaluation. The entire dataset is split as follows: 50% of the normal samples are associated to the training set while the remaining 50% are considered as a test set. The test set also contains 60% of the anomalous samples. The remaining 40% of the anomalous samples compose a set called the contamination set. The anomalous samples in the latter set, are actual instances of anomalies that occurred during gameplay, and were subsequently collected in our datasets. They serve to *contaminate* the training set with anomalous samples (with a ratio $c \in \{0\%, 4\%, 8\%, 12\%, 20\%\}$) and study the robustness of ML algorithms to data contamination. The rationale behind this splitting strategy is to maintain consistency across various experiments and ensure fairness in evaluating model performance under conditions of data contamination. Table II shows the summary of our datasets after the splitting step.

D. Existing window evaluation approaches

As mentioned in Section II-C, PW approaches are unsuitable for AD in CG sessions and window approaches are preferred. PA, PA%K and RPA approaches were proposed in [9] [33] and [11] respectively.

PA approach assumes that if any observation in a ground-truth anomaly segment is correctly detected, all the observations in this segment are considered as anomalous and correctly detected (i.e., p true positives are recorded). If none of the observations in the ground-truth anomaly segment is correctly detected, p false negatives are recorded. Observations that are not in a ground-truth anomaly segment are treated as point-wise.

RPA approach behaves in the same way as PA but for ground-truth anomaly segment: instead of recording p true positives, it records only 1 true positive for the whole window if any observation in a ground-truth anomaly segment is correctly detected. It records 1 false negative if none of the observations in the ground-truth anomaly segment is correctly detected. The observations that are not in a ground-truth anomaly segment are treated as point-wise.

PA%K approach aims to minimize overestimation errors in the PA approach by using a hyper-parameter K to adjust the anomaly prediction threshold. Specifically, PA%K considers an anomaly segment to be correctly detected if at least $K\%$

²<https://support.google.com/stadia/answer/9607891?hl=fr/>

³<https://www.nvidia.com/en-us/geforce/products/geforce-now/system-reqs/>

Ground truth	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Anomaly score predicted	0.9	0.8	0.4	0.9	0.8	0.4	0.2	0.7	0.9	0.3	0.4	0.3	0.1	0.2	0.3	0.8	0.7	0.6	0.7	0.2
PW approaches	1	1	0	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	1	0
PA approaches	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0
PA%K approaches	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
RPA approaches	1					1					1					1	1	1	1	0
WAD approaches	1					0					1					1				

(a) Prediction on anomaly windows

Ground truth	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Anomaly score predicted	0.4	0.3	0.1	0.2	0.3	0.8	0.7	0.6	0.7	0.2	0.4	0.3	0.1	0.2	0.3	0.8	0.7	0.6	0.7	0.2
PW approaches	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
PA approaches	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
PA%K approaches	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
RPA approaches	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
WAD approaches	1					1					1					1				

(b) Prediction on normal windows

Fig. 1. Illustration of PW, PA, RPA and WAD approaches. 0 is normal and 1 is anomalous. The anomaly score threshold to decide if an observation is anomalous or not is $\delta = 0.5$. Window size $p = 5$, $\alpha = 0.8$.

of its observations are accurately detected (i.e. p true positives are recorded if so and p false negatives otherwise.). The observations outside of a ground-truth anomaly segment are treated as point-wise.

Analysing this behavior (and later demonstrated by our experiments in Section IV-A), we can first observe that the PA approach is more tolerant to high false positive rates in a large anomaly windows and hence leads to high scores, thereby overestimating the performance of a model. In contrast, the RPA approach gives lower scores due to its differential and unfair evaluation between anomaly windows and normal windows. Classifying a window as anomalous, based on only one observation correctly detected (as done by PA/RPA approach) can result in many false alarms, which can have extra unnecessary costs and makes the detection model unpractical. Moreover, the three aforementioned approaches necessitate having access to accurate ground-truth segments in order to evaluate the model.

Given the limitations and shortcomings of these approaches when comparing various ML models, we therefore propose a Window Anomaly Decision (WAD) approach to accurately evaluate ML models for window-based anomaly detection.

E. Window Anomaly Decision (WAD) approach

WAD approach is designed to fairly evaluate anomalous and normal windows. Furthermore, compared to the previous approaches, which require a ground-truth to compute their scores, the WAD approach only uses the model output to classify a window as anomalous or normal. Hence, WAD is not only an evaluation approach, but also a decision approach.

The WAD computes the score for the whole window based on the model output for each observation. If more than a rate α of observations are classified as anomalous, the window itself will be classified as anomalous. Otherwise, the WAD approach will classify the window as normal. Specifically, WAD works as follows: given an anomalous window, a true positive is recorded if the model correctly detects a rate of α anomalous observations, otherwise a false negative is recorded. For a normal window, a true negative is recorded if less than a rate of α anomalous observations is detected, otherwise a false positive is recorded.

The approach is formulated as follows, where \tilde{w}_t denotes an unseen window of observations, $\mathbb{1}_{\tilde{y}_i=1}$ is the characteristic function that equals 1 if $\tilde{y}_i = 1$ and 0 otherwise, and p is the size of the window.

$$WAD(\tilde{w}_t) = \begin{cases} 1, & \text{if } (\sum_{i \in \{1..p\}} \mathbb{1}_{\tilde{y}_i=1}) \geq \text{int}(\alpha \cdot p) \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

with

$$\mathbb{1}_{\tilde{y}_i=1} = \begin{cases} 1, & \text{if } \tilde{y}_i = 1 \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

In this manner, WAD gives a higher score to a model that can detect at least a rate α of all the anomalous observations in the window and lower scores to a model that cannot.

We can use the parameter α to adjust the desired accuracy of the model by making WAD more or less tolerant to fault (i.e., error of the ML model or error with the measured metric). If the goal is to evaluate only perfect models, we can configure $\alpha = 1$. A smaller value can be chosen if a larger tolerance is desired. We leave to the domain experts the possibility to adjust the WAD α value according to their domain-specific considerations. In our evaluation study, we consider that a rate $\alpha = 0.8$ is reasonable since under this rate, a window does not contain enough anomalous observations to be considered as anomalous with respect to CG end-users' QoE degradation detection.

In order to prevent the possibility of missing anomalies spread across two consecutive windows, our approach should be used with overlapping consecutive windows as done in this study wherein a stride of 1 was employed.

Following the previous explanations, Fig 1 presents an example of how the PA, PA%K, RPA and WAD approaches work given anomalous and normal windows. There are four scenarios (two on anomalous windows and two on normal windows), each with the anomaly score predicted by an unsupervised model for some input windows of $p = 5$ observations, whose ground-truths are depicted. Given the anomaly score and a threshold $\delta = 0.5$, each approach assigns a binary variable either at the observation level (for PW, PA and RPA) or at the window level (WAD and RPA). WAD approach is used with $\alpha = 0.8$ and PA%K with $K = 80$ for fair comparison.

F. Performance evaluation metrics

The performance of the unsupervised ML algorithms is assessed using Precision (P), Recall (R) and F1-score (F1) which are defined as follows:

$$P = \frac{TP}{TP + FP} \quad (5)$$

$$R = \frac{TP}{TP + FN} \quad (6)$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \quad (7)$$

where TP, FP, FN denote the number of True Positives, False Positives and False Negatives, respectively. Precision denotes the fraction of relevant anomalies among all the instances identified as anomalous by the model, while Recall denotes the fraction of relevant anomalies among all the actual anomalies in the dataset. F1-score is the harmonic mean of Precision and Recall. The aforementioned performance metrics have a score of 1 (or 100%) if the model is perfect, and 0 (0%) otherwise.

However, we note that F1-score presents some limitations: F1-score does not consider the number of True Negatives (TN) and is not invariant to class swapping (i.e., if the positive class becomes the negative class and vice versa). These limitations are highlighted and discussed in [39] and the Matthews Coefficient Correlation (MCC) score is recommended for a better evaluation of AD models.

MCC score [40] is a binary classification metric that is similar to the Pearson coefficient correlation. It gives a score of +1 (100%) for a model that correctly predicts the anomalous and normal instances (i.e., positive and negative class, respectively), 0 (0%) for a model that is not better than a random guessing classifier, and -1 (-100%) for the worst model. MCC is defined as follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

In AD tasks, taking the negative class (here the number of normal observations detected) into consideration can be seen as meaningless since the goal is to detect anomalies (the positive class). However, we experimentally show that ignoring the negative class while using the F1-score can lead to unreliable conclusions about the model performance.

In our experiments, we both considered F1-score and MCC to compare the ML models. We do not include the AUC score which can lead to misleading results when the datasets are imbalanced [41], [38].

G. Unsupervised anomaly detection models

We present below the descriptions of eight unsupervised AD models that are used in our comparative evaluation. We only focus on reconstruction-based, one-class classification and isolation-based algorithms because they are the most widely studied in the literature, while generative and predictive

algorithms suffer from limitations such as higher computational complexity and expensive training due to instability and reproduction issues [22].

The eight unsupervised models evaluated in this paper have been selected since they are used in many evaluations studies or to benchmark new approaches for various anomaly detection tasks [42], [43], [44].

- **PCA:** Principal Component Analysis is often used as a baseline for AD tasks. It performs a lossy reconstruction using the principal components computed with the Singular Value Decomposition (SVD). We choose a number of principal components that preserve 90% of the variance in the data in our Scikit-Learn implementation.
- **iForest:** Isolation Forest uses *isolation* trees to recursively isolate anomalies. Its performance relies on the number of trees t , the sub-sampling size ϕ , and the assumed amount of contamination in the dataset. We use the default values of these hyper-parameters in the Scikit-Learn library.
- **OC-SVM:** One-Class SVM [6] is a popular and efficient shallow ML model. OC-SVM uses a hyper-parameter, ν , which is an upper bound on the fraction of outliers in the dataset. We use Scikit-Learn implementation of OC-SVM with *rbf* function and $\nu = 0.1$, which is the default parameter used in [6]. Due to the computational complexity of OC-SVM, we process the training inputs with PCA by retaining 70% explained variance.
- **Deep-SVDD:** Deep Support Vector Data Description [7] can be seen as a deep learning implementation of OC-SVM. Deep-SVDD benefits from the efficiency of deep learning on large, high-dimensional data. We use the *soft-boundary objective* function which assumes that the training data may contain a ratio ν of anomalies. We use the PyTorch implementation of Deep-SVDD on Github.⁴
- **AE:** AutoEncoder (AE) is a neural network architecture composed of an encoder, that encodes input data into a low-dimensional space and a decoder that reconstructs input data from the low-dimensional features. We choose an AE with feed-forward network and *Tanh* activation function for our custom implementation in PyTorch.
- **LSTM-VAE:** It combines a neural network designed for time-series, the LSTM, to an autoencoder with bayesian inference for reconstruction of input data [28]. The reconstruction error is used as anomaly score and we provide a custom implementation of LSTM-VAE in PyTorch based on TensorFlow implementation on Github.⁵
- **DAGMM:** Deep Autoencoder Gaussian Mixture Model combines an autoencoder and a gaussian mixture model, where the representation given by the autoencoder is feed to the gaussian model to produce an energy used as anomaly score. We process the training inputs with PCA by retaining 90% explained variance to decorrelate the features for DAGMM and avoid runtime issues. Our implementation is based on the PyTorch implementation on Github.⁶

⁴<https://github.com/lukasruff/Deep-SVDD-PyTorch>

⁵https://github.com/paya54/Anomaly_Detect_LSTM_VAE

⁶<https://github.com/danieltan07/dagmm>

TABLE III
COMPARISON OF WAD, PA, PA%K AND RPA APPROACHES WITH MCC AND F1 SCORE.

	(1 - β) perfect detector	PA	RPA	WAD $_{\alpha=0.8}$	PA%K $_{K=80}$	WAD $_{\alpha=0.9}$	PA%K $_{K=90}$	WAD $_{\alpha=1}$	PA%K $_{K=100}$
MCC-Score	0.05	95.31 (± 0.02)	87.01 (± 0.06)	97.93 (± 0.02)	93.38 (± 0.03)	90.62 (± 0.07)	85.61 (± 0.05)	64.30 (± 0.08)	58.97 (± 0.05)
	0.1	90.70 (± 0.03)	76.34 (± 0.07)	91.36 (± 0.03)	81.94 (± 0.02)	74.56 (± 0.08)	63.81 (± 0.07)	44.79 (± 0.11)	33.37 (± 0.09)
	0.15	86.13 (± 0.01)	67.20 (± 0.04)	80.80 (± 0.07)	65.74 (± 0.07)	59.19 (± 0.08)	42.11 (± 0.08)	32.07 (± 0.09)	13.37 (± 0.09)
	0.2	81.60 (± 0.01)	59.17 (± 0.08)	68.60 (± 0.11)	47.36 (± 0.10)	46.18 (± 0.05)	22.09 (± 0.03)	23.03 (± 0.05)	-03.02 (± 0.06)
	0.25	77.11 (± 0.02)	51.97 (± 0.09)	56.48 (± 0.20)	28.66 (± 0.16)	35.51 (± 0.06)	3.96 (± 0.06)	16.40 (± 0.09)	-16.08 (± 0.10)
	0.3	72.62 (± 0.02)	43.35 (± 0.09)	45.23 (± 0.15)	10.50 (± 0.11)	26.78 (± 0.09)	-12.09 (± 0.09)	11.39 (± 0.08)	-26.28 (± 0.11)
F1-Score	0.5	54.11 (± 0.03)	21.96 (± 0.07)	0.13 (± 0.13)	-46.37 (± 0.07)	00.15 (± 0.06)	-51.71 (± 0.06)	00.04 (± 0.19)	-52.06 (± 0.09)
	0.05	98.06 (± 0.01)	89.29 (± 0.06)	99.04 (± 0.01)	97.31 (± 0.01)	95.20 (± 0.04)	93.72 (± 0.03)	74.89 (± 0.08)	75.93 (± 0.06)
	0.1	96.19 (± 0.01)	80.31 (± 0.07)	95.72 (± 0.02)	92.44 (± 0.01)	84.42 (± 0.06)	82.14 (± 0.05)	51.60 (± 0.16)	54.66 (± 0.13)
	0.15	94.36 (± 0.01)	72.63 (± 0.06)	89.34 (± 0.05)	84.81 (± 0.04)	70.10 (± 0.08)	67.79 (± 0.06)	32.85 (± 0.13)	38.24 (± 0.12)
	0.2	92.60 (± 0.01)	66.02 (± 0.09)	79.95 (± 0.09)	74.63 (± 0.06)	54.23 (± 0.06)	52.78 (± 0.03)	19.40 (± 0.06)	26.74 (± 0.06)
	0.25	90.89 (± 0.02)	60.28 (± 0.10)	68.05 (± 0.19)	62.60 (± 0.13)	38.91 (± 0.08)	39.04 (± 0.07)	10.69 (± 0.10)	19.25 (± 0.08)
	0.3	89.23 (± 0.02)	55.23 (± 0.10)	54.56 (± 0.17)	49.79 (± 0.11)	25.75 (± 0.12)	27.74 (± 0.08)	5.49 (± 0.06)	14.53 (± 0.09)
	0.5	82.99 (± 0.02)	39.88 (± 0.08)	10.00 (± 0.07)	12.15 (± 0.07)	02.11 (± 0.04)	07.48 (± 0.05)	00.19 (± 0.02)	07.18 (± 0.07)

- **USAD:** UnSupervised Anomaly Detection [5] adversely trains two autoencoders sharing the same encoder under two objectives: (i) reconstruct input data, and (ii) discriminate real data from reconstructed data. Our implementation of USAD is based on the PyTorch implementation on Github.⁷

The aforementioned reconstruction-based algorithms require a threshold that needs to be carefully chosen. We found in our previous work [38] that using the 3σ rule-of-thumb as a thresholding rule may lead to poor scores, due to a threshold too low to detect all the anomalies in the test set (i.e., low recall scores). In this paper, we use a different strategy: we randomly select 20% of the test set and select the threshold δ that gives the best F1-score on this sample of the test set. This threshold is then kept and used for evaluation on the remaining 80% of the test set. This thresholding strategy often used in AD [4], [5] reports the best performance that the model can achieve.

We do not perform any hyper-parameters tuning in this study. Instead, we adopt the parameter settings documented in the respective papers of each model, as these configurations have been validated across multiple datasets and determined to be the optimal choices. We then train the neural network models using Adam optimizer with a learning rate of 10^{-3} and a batch size of 128 during 100 epochs. Early stopping is applied to avoid overfitting and longer training time, i.e., training is stopped when the validation loss do not decrease during 10 consecutive epochs. For each experiment, the models are evaluated five times to draw reliable conclusions except for OC-SVM which is run once due to its computational complexity. The details on our implementations are available on Github.⁸

IV. EXPERIMENTAL EVALUATION

In this section we first validate the proposed WAD approach by showing that it produces more accurate results compared

to the existing approaches. Furthermore, we experimentally demonstrate with synthetic datasets that F1-score presents some limitations. We then perform comparative evaluations of the eight aforementioned models while studying the impact of data contamination c and window size p .

A. Comparison of WAD with existing approaches

In this section, we show that the WAD approach can yield more accurate performance results compared to existing approaches. We then define a synthetic model that produce a rate β of detection errors on the test set. Specifically, this model has an incorrect prediction for $\beta * 100$ observations over a set of 100 observations. We refer to this model as $(1 - \beta)$ -perfect detector, which is perfect when $\beta = 0$ and is always wrong when $\beta = 1$. We select different values for $\beta \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.5\}$ and compute F1-score and MCC score accordingly. Table III depicts the F1 and MCC score of the $(1 - \beta)$ -perfect detector for the WAD approach.

We notice that both MCC and F1-scores decrease for each window approach as the error rate of the synthetic model increases. On the one hand, when β varies from 0.05 to 0.25, PA still gives high scores to the model (from a F1-score of 98% to 90%), while RPA results in low scores (from a F1-score of 89% to 60%). On the other hand, WAD $_{\alpha=0.8}$ and PA%K $_{K=80}$ strike a balance to provide a more accurate reflection of model quality. For instance, MCC score decreases from 98% to 68% for WAD and from 93% to 47% for PA%K. Both approaches increase their penalization as β increases. However, when β reaches 0.5, which corresponds to random predictions, WAD approach reports an MCC score of 0%, while PA%K continues to penalize the model and produces negative MCC scores (worse than a random model). The reason behind this behavior of PA%K is related to its way of adjusting differently for anomalous and normal segments prediction compared to WAD approach, which handles both segment types similarly.

As for the PA%K approach, the score reported with WAD also depends on a tolerance parameter, here α . Increasing α from 0.8 to 1, results in low scores, even lower than those reported with RPA on a near-perfect model. For instance,

⁷<https://github.com/manigalati/usad>

⁸<https://github.com/mosaico-anr/unsupervised-ml-ad-qoe-deg>

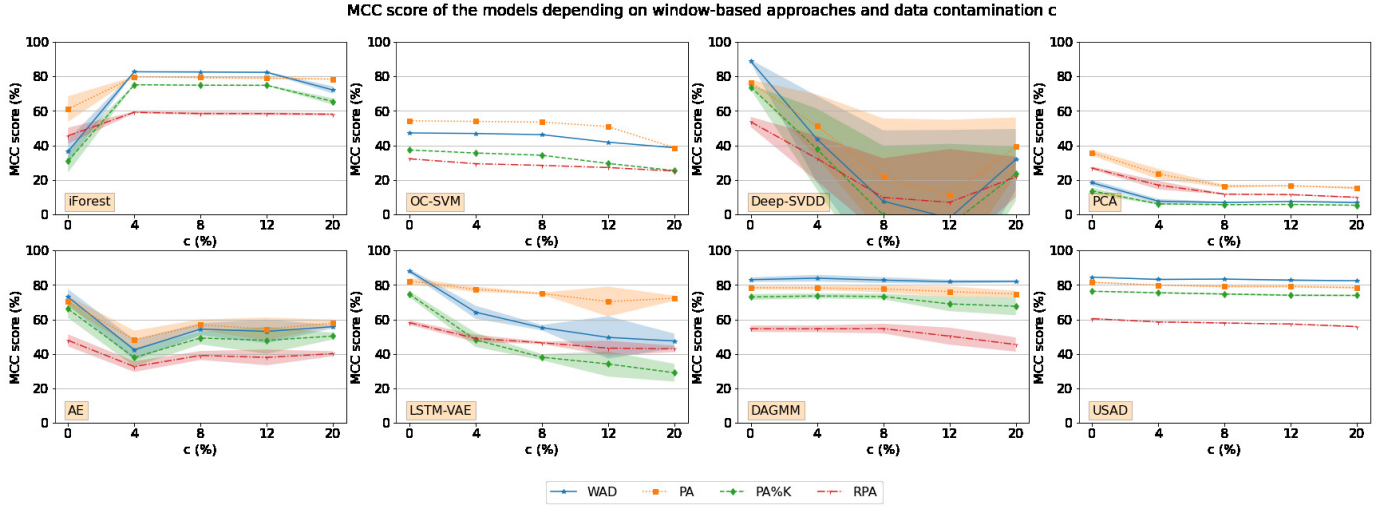


Fig. 2. MCC score of unsupervised ML models with STD dataset according to the evaluation window-based approaches and the data contamination ratio c .

TABLE IV
COMPARISON BETWEEN F1-SCORE AND MCC COMPUTED ON SYNTHETIC MODELS.

Model	Window approach	Precision	Recall	F1-score	MCC
Zero-rule	PW	58.38	100	73.72	0.0
	PA	58.38	100	73.72	0.0
	RPA	19.06	100	32.02	0.0
	$WAD_{\alpha=0.8}$	54.34	100	70.41	0.0
	$PA\%K_{K=80}$	58.38	100	73.72	0.0
	$WAD_{\alpha=1}$	52.57	100	68.92	0.0
	$PA\%K_{K=100}$	58.38	100	73.72	0.0
Random	PW	58.37	49.97	53.89	0.0
	PA	73.97	96.20	82.99	54.10
	RPA	28.61	77.77	39.88	21.95
	$WAD_{\alpha=0.8}$	54.32	5.47	9.94	-0.01
	$PA\%K_{K=80}$	19.70	8.74	12.11	-46.40
	$WAD_{\alpha=1}$	52.19	0.10	0.20	-0.03
	$PA\%K_{K=80}$	12.40	5.05	7.18	-52.06

a model with a low error rate ($\beta = 0.05$) and WAD with tolerance $\alpha = 1$, achieves an F1-score of 74%, while the PA and RPA approaches report a F1-score of 98% and 89%, respectively. We obtain such scores because with a tolerance $\alpha = 1$, only models that can detect all anomalies in the window get high scores.

Unlike PA, PA%K and RPA, the WAD approach offers a more accurate evaluation of model performance by equally adjusting the prediction for anomalous and normal segments and the score obtained by a model is proportional to its ability to accurately detect a sequence of anomalous observations long enough to be severe.

B. Comparison between F1-score and MCC

We compare in this section the F1-score and MCC metrics on synthetic anomaly detection models. We first consider a *zero-rule* model that always outputs $\hat{y}_t = 1 \forall \tilde{x}_t$ (i.e., this model always classifies any input \tilde{x}_t as an anomaly). This model, if deployed, triggers false alarms and may introduce high anomaly mitigation costs. We report the F1-score and

the MCC of the *zero-rule* model on our datasets with each window approach strategy in Table IV.

The results of Table IV show that F1-score reports higher scores ($\sim 70\%$) for each window approach (except for RPA), while the MCC reports a score of 0% (i.e., worse than a tossing coin classifier) regardless of the approach. High F1-scores are mainly due to perfect recall scores, i.e., the model detects all the anomalies in the datasets. But these F1-scores are misleading as the model identifies many normal instances as anomalous and then outputs a high number of false positives but no true negatives, which are not taken into account the F1-score. Consequently, when evaluating models having a low rate of true negatives, the F1-score metric should be carefully used.

Furthermore, we make the same observations in Table IV with a random guessing model. Indeed, we notice that F1-score is higher than MCC score, particularly for each approach. For instance, F1-score reported with PA (resp. $WAD_{\alpha=0.8}$) is 83% (resp. 9.9%), while the MCC score is 54% (resp. 0%). Since PA is known to overestimate the model performance, we also compute the F1-score and the MCC with the PW approach which are 53% and 0% respectively. The high F1-score obtained with this model regardless of the approach used asserts the observations made with zero-rule model.

If considering all conditions (TP, TN, FP, FN) of the models is required, the MCC score should be preferred over the F1-score. The results of this experiment confirm the conclusions drawn by Chicco et al. [39]. In general, we recommend to use the MCC score and F1-score metrics together to avoid unreliable conclusions.

C. Data contamination impact on unsupervised models

This section presents the evaluation results for the eight unsupervised ML models. We first present the impact of data contamination on the performance of these models while using window-based approaches (WAD, PA, PA%K and RPA). Fig. 2 depicts the variation of their MCC scores using data contamination ratio $c \in \{0\%, 4\%, 8\%, 12\%, 20\%\}$ on the STD dataset. We choose these values for data contamination as there

TABLE V

OVERALL PERFORMANCE (MEAN AND STANDARD DEVIATIONS OVER 5 RUNS) ACCORDING TO THE TRAINING SET'S DATA CONTAMINATION RATIO c AND USING THE WAD APPROACHES ON THE 3 DATASETS. BOLD VALUES INDICATE THE BEST SCORE FOR EACH MODEL.

Categories	Models	c (%)	STD		GFN		XC	
			F1	MCC	F1	MCC	F1	MCC
Isolation-based	iForest	0	48.68(± 9.33)	36.45(± 6.97)	25.65(± 2.19)	19.20(± 1.74)	54.36 (± 3.26)	48.60 (± 2.82)
		4	90.99 (± 0.12)	82.70 (± 0.19)	49.89(± 0.89)	37.26(± 0.72)	49.69(± 3.40)	44.34(± 2.96)
		8	90.93(± 0.15)	82.59(± 0.40)	50.15 (± 0.85)	37.49 (± 0.36)	48.66(± 3.20)	44.34(± 2.43)
		12	90.82(± 0.19)	82.39(± 0.36)	48.58(± 0.25)	36.42(± 0.38)	46.01(± 2.26)	42.37(± 2.28)
		20	83.77(± 1.48)	72.13(± 2.11)	48.00(± 0.46)	36.16(± 0.36)	43.39(± 1.46)	40.98(± 1.34)
One-class classification	OC-SVM	0	77.67	47.24	69.88	29.41	66.27	51.79
		4	77.35	46.93	70.37	30.36	65.15	48.51
		8	76.79	46.23	68.68	28.24	64.18	46.60
		12	74.20	41.87	67.02	26.19	63.32	44.98
		20	71.77	38.66	63.47	21.78	62.03	42.65
	Deep-SVDD	0	90.91 (± 0.59)	88.85 (± 0.83)	66.07(± 7.34)	8.69(± 20.27)	49.61(± 14.19)	31.72(± 19.7)
		4	74.05(± 10.99)	43.64(± 25.11)	67.49(± 4.78)	12.07(± 13.27)	46.57(± 17.96)	23.9(± 26.11)
		8	56.48(± 19.1)	7.63(± 41.11)	65.03(± 6.16)	5.82(± 16.13)	62.87 (± 13.46)	46.74 (± 19.20)
		12	53.08(± 23.06)	-2.13(± 51.12)	67.44(± 6.68)	11.69(± 19.17)	37.07(± 7.59)	13.64(± 9.18)
		20	68.82(± 7.65)	31.9(± 17.65)	68.3 (± 7.69)	13.34 (± 21.11)	34.8(± 10.89)	7.70(± 15.14)
	PCA	0	61.79 (± 0.90)	18.33 (± 1.73)	63.59 (± 0.52)	1.89 (± 0.81)	36.38 (± 0.29)	10.96 (± 0.25)
		4	58.26(± 0.21)	7.64(± 1.29)	63.13(± 0.70)	-2.40(± 0.97)	33.32(± 0.24)	6.63(± 0.35)
		8	58.80(± 0.24)	6.87(± 0.49)	63.56(± 0.26)	-2.19(± 0.52)	33.85(± 0.66)	7.58(± 0.79)
		12	58.92(± 0.13)	7.50(± 0.25)	63.62(± 0.28)	-1.96(± 0.58)	34.53(± 0.43)	8.45(± 0.41)
		20	58.66(± 0.14)	6.87(± 0.58)	63.13(± 0.16)	-3.25(± 0.24)	35.43(± 0.39)	9.42(± 0.58)
	AE	0	86.73 (± 2.04)	73.05 (± 4.61)	79.44 (± 4.08)	47.17 (± 10.72)	76.84 (± 6.48)	68.12 (± 8.43)
		4	74.09(± 2.56)	42.38(± 6.03)	65.38(± 1.65)	10.98(± 4.32)	49.23(± 10.70)	30.97(± 15.82)
		8	79.01(± 1.85)	54.44(± 3.47)	65.11(± 1.29)	10.02(± 2.25)	40.77(± 2.66)	18.98(± 2.63)
		12	78.29(± 3.32)	53.1(± 6.84)	64.34(± 2.42)	8.36(± 4.97)	45.74(± 8.53)	25.25(± 11.12)
		20	79.67(± 0.86)	55.85(± 0.99)	64.48(± 2.11)	6.59(± 6.83)	42.98(± 2.75)	21.15(± 3.36)
	LSTM-VAE	0	94.44 (± 0.57)	88.13 (± 1.30)	79.30 (± 0.63)	50.68 (± 1.65)	64.15 (± 12.44)	54.95 (± 11.11)
		4	81.9(± 2.02)	64.23(± 3.78)	76.87(± 0.45)	43.28(± 0.84)	63.50(± 2.31)	51.90(± 3.41)
		8	76.51(± 0.93)	55.26(± 1.56)	74.94(± 1.23)	38.09(± 3.25)	59.05(± 4.17)	45.65(± 4.99)
		12	74.14(± 5.12)	49.61(± 12.3)	73.00(± 1.58)	34.56(± 4.25)	57.75(± 0.53)	44.44(± 0.65)
		20	71.91(± 2.62)	47.55(± 4.23)	70.28(± 0.86)	31.24(± 1.88)	53.29(± 1.82)	39.16(± 2.85)
Reconstruction-based	DAGMM	0	91.23(± 0.71)	83.11(± 1.37)	74.31 (± 1.86)	32.08 (± 3.64)	68.10 (± 2.82)	55.76 (± 3.82)
		4	91.84 (± 0.77)	83.95 (± 1.96)	76.72(± 1.81)	38.08(± 3.87)	66.16(± 1.34)	53.18(± 2.13)
		8	91.50(± 0.79)	82.83(± 1.74)	75.09(± 3.04)	34.47(± 7.15)	63.02(± 2.98)	49.17(± 3.80)
		12	91.33(± 0.48)	81.92(± 1.12)	73.86(± 2.9)	31.32(± 7.31)	63.04(± 2.48)	49.14(± 3.20)
		20	91.43(± 0.3)	82.12(± 0.65)	71.66(± 3.16)	24.80(± 6.55)	60.10(± 3.14)	45.54(± 4.05)
	USAD	0	92.79 (± 0.11)	84.57 (± 0.20)	76.73 (± 0.41)	38.77 (± 1.03)	73.27 (± 1.52)	62.30 (± 1.87)
		4	92.09(± 0.20)	83.23(± 0.41)	76.64(± 0.66)	38.04(± 1.56)	65.74(± 1.56)	52.60(± 2.03)
		8	91.78(± 0.14)	83.44(± 0.28)	75.25(± 0.77)	34.31(± 2.80)	65.16(± 3.45)	51.77(± 4.31)
		12	91.63(± 0.08)	82.82(± 0.55)	75.06(± 0.38)	33.69(± 1.03)	61.24(± 4.49)	47.97(± 4.65)
		20	91.36(± 0.17)	82.48(± 0.13)	74.64(± 1.30)	32.23(± 2.99)	61.06(± 5.74)	46.38(± 7.02)

are only few anomalies encountered in production networks, and these values are those encountered in previous works [45], [31]. We go up to 20% to stress the models.

As depicted in Fig 2, data contamination impacts the ML models differently depending on the model family. Isolation-based models appear to benefit from data contamination. iForest shows low MCC score when there is no data contamination. Its performance largely increases with data contamination $c = 4\%$ and slightly decreases as data contamination ratio increases. This is the expected behavior of iForest which assumes that anomalies are present in the training set but in few quantity and starkly differ from normal instances. The

presence of anomalies helps iForest during training but with the increasing anomalies in training set, its performance drops when c reaches 20%. Therefore, the iForest can not efficiently isolate anomalies when there are a lot of anomalies in the training set.

One-class classification models, OC-SVM and Deep-SVDD, have their best performance with no data contamination (MCC_{WAD} of 47.24% and 88.65%, respectively for instance). However, their performance decreases when anomalies are included in the training set. MCC score of OC-SVM slightly decreases and remains at a similar level. Surprisingly, Deep-SVDD performance (which has the best MCC score

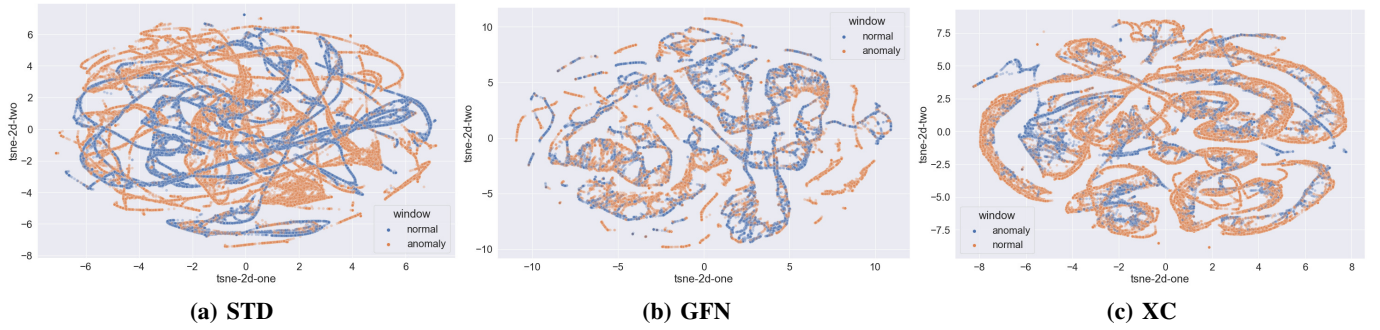


Fig. 3. Visualization of normal and anomalous windows for the three datasets in a two-dimensional space by t-SNE. Normal windows in blue and anomalous windows in orange.

among all the models without data contamination), drops significantly and becomes even worse than a random classifier when $c = 12\%$ (i.e., $MCC_{WAD} = -2\%$) and presents high standard deviations errors (i.e., 51%). Deep-SVDD, although being a neural implementation of OC-SVM with the same objective function, presents high variability in results from one run to another. Moreover, it is difficult to contrast these results on Deep-SVDD with previous works as there is, to the best of our knowledge, either no work on data contamination impact with time-series data using Deep-SVDD or the existing works focus on image datasets.

Reconstruction-based models present two different behavior when facing data contamination. On one hand, we note that PCA, AE and LSTM-VAE are less robust to the presence of anomalies in the training set. PCA, which is the less efficient model for anomaly detection, achieves a MCC_{WAD} score of 18% when $c = 0\%$ which drops to 7% when $c = 4\%$. From this contamination level, increasing the rate of anomalies seems to have no significant impact on the PCA model, as its performance remains at the same level. AE and LSTM-VAE report a MCC_{WAD} score of 73% and 88%, respectively which decreases to 55% and 47%, respectively, when data contamination ratio reaches $c = 20\%$. AE and LSTM-VAE are the most impacted models to data contamination since a contamination ratio of $c = 4\%$ is enough to deteriorate their performance by up to 30%. On the other hand, the impact of data contamination on the performance of DAGMM and USAD models is negligible. USAD presents its highest score at $c = 0\%$ with the lowest standard deviation. The performance of DAGMM model peaks at 83% MCC_{WAD} with $c = 4\%$. Unlike our previous work [38], the performance of reconstruction-based models does not collapse with data contamination. This is the consequence of the thresholding rule used in this work, which is better than 3σ thresholding rule. F1-score reported for AE and USAD in this work are in the same magnitude as those presented in previous works [31], [5]. The only surprising results are those with DAGMM which contradict those in previous works. The DAGMM model in this study is less impacted by data contamination, while in [4], [31], [45] the performance drop is between 10-50% with a data contamination rate of 12%. A reason behind the better performance of DAGMM in this work is the datasets used that

present correlated features. DAGMM model use Cholesky matrix factorization to compute the anomaly score. However, this factorization fails with our datasets because matrices computed from our features present negative eigenvalues when Cholesky factorization requires strictly positive eigenvalues. Hence, to run DAGMM model on our datasets, a PCA processing step is required to decorrelate the features.

Table V presents the F1-score and MCC score obtained using the WAD window approach on the three datasets (STD, GFN and XC). Our study show that the impact of data contamination observed with the STD dataset is consistent across all three datasets. In particular, our analysis reveals the following key findings:

- iForest model benefits from moderate data contamination when applied to the GFN dataset but its performance declines when applied to the XC dataset;
- OC-SVM model's performance decreases with increasing levels of data contamination while Deep-SVDD model's performance exhibits more fluctuation with varying degrees of data contamination when applied to GFN and XC datasets, as well as STD dataset;
- Reconstruction-based models' performance significantly decline as data contamination increases.

However, the models performance with GFN and XC datasets are not as impressive as those achieved with the STD dataset. For instance, LSTM-VAE without data contamination achieves a MCC of 50.68% for GFN and 54.95% for XC. Similarly to the results on STD dataset, iForest, DAGMM and USAD models remain robust to data contamination in GFN and XC datasets, but their performance levels are lower. To further understand this difference, we use t-SNE projection [46] to visualize in a two-dimensional representation the normal and anomalous windows of each dataset. As depicted in Fig. 3, the GFN and XC datasets have numerous overlaps between normal and anomalous windows, while the STD dataset's normal windows cluster in the center of the figure. Consequently, detecting anomalies in the former datasets may be more challenging since anomalous and normal windows are indistinguishable.

Based on our evaluation, we can conclude that while isolation-based models may benefit from data contamination up to a certain moderate level, reconstruction and one-class

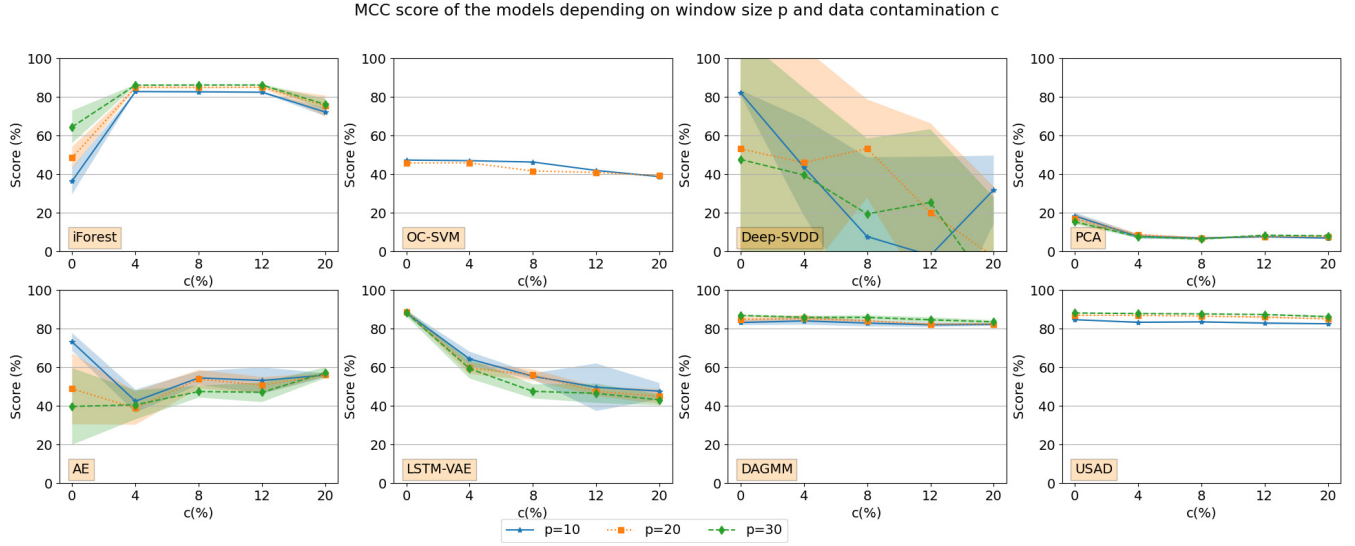


Fig. 4. MCC score of unsupervised ML models with STD dataset according to data contamination ratio c and window size p .

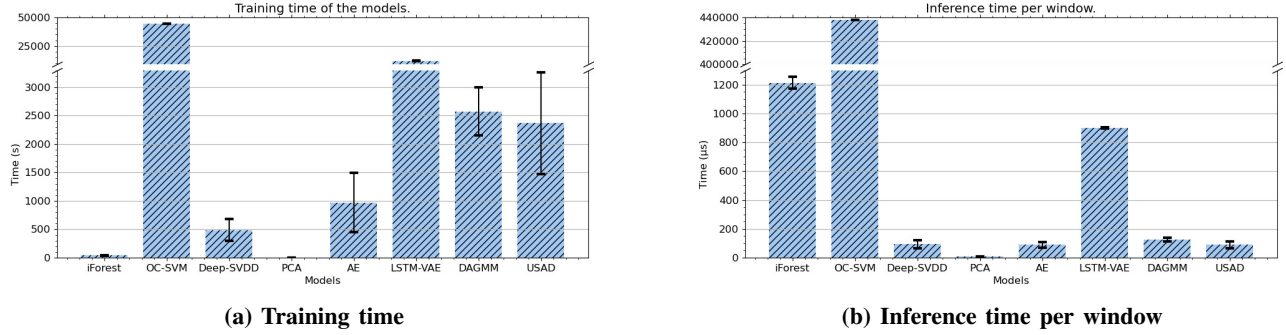


Fig. 5. Train and inference time.

models show a decrease in their performance when faced with anomalies in the training sets.

D. Impact of window size

We also analyze the impact of window size on model performance. The window size represents the duration of the anomalous events that can occur during CG sessions. We choose 3 different values for window size ($p \in \{10, 20, 30\}$) to have window length representative of user perceptions (50ms, 100ms and 150ms) and to analyze if the models can efficiently detect short or long anomalous events in CG sessions. Fig. 4 depicts the MCC score of each model along with the standard deviations with STD dataset. For each model, we represent the impact of the window size p and the contamination ratio c on its performance.

It is worth noting that increasing the window size seems to slightly improve the performance of iForest, DAGMM and USAD models. For instance, the iForest model shows a +30% increase with $c = 0\%$ when the window size varies from 10 to 30. DAGMM and USAD show a moderate increase in performance (i.e., +4%). Their performance remain better with higher window size regardless of the data contamination ratio.

Conversely, OC-SVM, PCA and LSTM-VAE models achieve high performance value with a window size of 10, which decreases as the window size increases. The high variability of Deep-SVDD and the performance variation of AE do not allow to observe a significant pattern. Our findings remain consistent upon studying the impact of window size on both the GFN and XC datasets: increasing the window size may improve very slightly the performance of iForest and USAD but not the other models.

Apart from the USAD model, to the best of our knowledge, there are no studies on the impact of window size on the AD learning models. Moreover, Audibert et al. [5] showed that increasing the window size has insignificant impact on the performance of USAD, while, in our case, it leads to an increase in model performance. We attribute the performance improvement of iForest and DAGMM (and in the same way the performance degradation of OC-SVM, PCA and AE) to the fact that increasing window sizes leads to the presence of anomalous observations in the training set that may improve (or deteriorate for OC-SVM, PCA and AE) the performance of the models.

The takeaway from these experiments is that detecting

longer anomalous events is less efficient for most of the unsupervised models except for iForest, DAGMM and USAD which are able to better learn anomalies over larger windows.

E. Computational time performance

This section analyzes the computational performance of unsupervised ML models. We measure the training time for each model on the training set and compute the average time taken by each model for inference given a window size of 10. The models are trained and tested on a Google Cloud Platform (GCP) VM with 8 CPUs and 30GB RAM. To accelerate the training of LSTM-VAE, a NVIDIA T4 GPU is used. Fig. 5 presents the training and inference times.

PCA takes the lowest training time (i.e., 2s) while iForest training time is 20x longer. Neural networks based models present longer training time ranging from 970s for AE to 2500s for DAGMM. Despite the use of GPU, LSTM-VAE takes 4.6 times longer than DAGMM to complete its training. The high variability in their training times is due to early stopping strategy that prevents the models from overfitting. The most time consuming model is OC-SVM that takes 44700s for its training. Compared to other shallow models such as iForest, OC-SVM require 1000x longer to train. We make similar observations when comparing the inference time for a window of observations of size $p = 10$. Deep learning based models have inference time between 89 and 126 μ s (except for LSTM-VAE that goes up to 900 μ s with GPU). Unlike neural network models, shallow models like iForest and OC-SVM take much more inference time, i.e., iForest takes 1213 μ s and OC-SVM takes 360x longer.

Algorithms with the lowest training/inference time like PCA or with highest training/inference time like OC-SVM or LSTM-VAE do not provide as good qualitative results as iForest, DAGMM and USAD models. The latter present good performance and robustness with reasonable training and test times.

V. DISCUSSION

A. ML Models recommendation

Based on our comparative study, we highlight the advantages and benefits of the models and provide recommendations to network operators according to the following requirements (summarized in Tab VI):

Performance: Deep-SVDD and LSTM-VAE are the models with the best performance without data contamination. They achieve near-optimal detection of anomalies in the dataset with low false alarms. However, they present performance degradation when anomalies are present in the training set. The performance instability of Deep-SVDD in the face of anomalies makes it unreliable. The LSTM-VAE model based on LSTM neural networks has high computing cost during training and testing (e.g., requiring more GPUs) incurring high costs in energy consumption in deployment.

Robustness: There is no guarantee that data from production network environments are free of anomalies and removing them for training the models is a difficult task. Therefore, we recommend the use of algorithms such as iForest, DAGMM and USAD in such situations since they are robust to data

TABLE VI
ML MODELS RECOMMENDATION

Model	Performance	Robustness	Deployment	Explainability
iForest	+	++	-	-
OC-SVM	-	+	- -	- -
Deep-SVDD	++	- -	++	- -
PCA	- -	++	++	++
AE	+	-	++	- -
LSTM-VAE	++	- -	-	- -
DAGMM	++	++	+	- -
USAD	++	++	+	- -

++: good; +: somewhat good; -: somewhat bad; - -: bad.

contamination and will ensure reasonable performance when data contamination is not too high.

Deployment: For deployment recommendations, we consider the training and inference time. All the models used in this work are trained *offline*. Neural networks are trained during several epochs and take a long time for their training phase while PCA and iForest are fast to train. In production, deep learning models like DAGMM and USAD can be used for *real-time* prediction since they have low inference latency. Real-time inference with iForest or LSTM-VAE can be difficult due to their high inference time which also exacerbates scalability issues for the former and resource requirements for the latter.

Explainability: Machine-learning techniques are known to suffer from the lack of transparency and explainability. Network experts in practice need to better understand the decisions taken by ML models to efficiently monitor and manage network systems. None of the algorithms used in this study achieve the trade-off between efficiency and explainability. iForest and neural networks-based algorithms (LSTM-VAE, DAGMM, USAD) are seen as *black-box* and do not allow for easy interpretation of their anomaly detection.

B. Limitations

This work presents some limitations worth discussing. First, the thresholding strategy used in this evaluation, which selects the threshold that lead to the best score, is not applicable in practice since it requires the use ground-truth test sets. Further investigations should be carried out to automatically determine and select the best threshold values. Some works [47], [48] advocate for the use of evaluation metrics insensitive to the selected threshold. Second, the chosen criteria to introduce anomalies in the collected datasets may result in introducing too many of them (cf. II) which can raise questions worthy of further investigation. Additionally, as discussed in Section IV-C, a PCA processing step was employed to decorrelate the features before utilizing the DAGMM model. It is worth noting that the performance of this model may differ when applied to datasets with correlated or uncorrelated features. Moreover, as mentioned in Section III-G, we could not include in our evaluation study, generative and predictive models since they may suffer from computational constraints.

Furthermore, in our methodology, we used the DECAF tool [14] to collect the QoE/QoS time-series features on the client-side. However, this tool is only compatible with CG platforms based on WebRTC and cannot be used with other CG platforms to collect the time-series features employed in our study. Finally, network operators may not always have the possibility to gather such information at the client-side. Consequently, they may need to develop QoE degradation detection algorithms based on features that can be readily collected at the network edge, such as network packets.

VI. CONCLUSION AND FUTURE WORK

In this paper, we performed a comparative evaluation of eight unsupervised ML models applied to the detection of users' QoE degradation in CG sessions. Our evaluation showed that the F1-score metric, which is widely used for model evaluation, has limitations and should be combined with the MCC score for more accurate model evaluation. Moreover, existing window-based approaches, used to cover sequences of events including anomalies, lead to erroneous conclusions regarding model performance. To address these limitations, we proposed the WAD approach to allow for a fair and better assessment. The WAD approach offers the possibility of parameter calibration to detect more or less severe anomalies. As practical considerations in the use of ML models in networking, we also showed that data contamination has a considerable impact on unsupervised ML models, and revealed their disparity with respect to their computational performance. Our study has shown that the use of models such as those included in our evaluation in an industrial context requires further investigation of their applicability and calibration. Highly performing state-of-the-art ML models have not been necessarily designed with industrial considerations in mind such as robustness, energy consumption, explainability and likely others. Many of these considerations can be conflicting with commonly used performance evaluation metrics. In summary, our evaluation study emphasizes the importance of employing a consistent methodology and appropriate metrics when evaluating ML models. In particular, we found no one-size-fits-all solution as some solutions may be preferred to others depending on the requirements of the operational environment under consideration.

As future work, we plan to investigate and develop anomaly detection models capable to achieve the right trade-offs between these seemingly conflicting goals when approaching the detection of abnormal network events and their impact on user QoE particularly in the case of demanding applications such as low-latency applications. To address these objectives, we will specifically focus on QoE degradation detection based on network packets collected at the network edge. We will collect these packets in various experimental and in-production scenarios, which will enable us to build more robust and generalizable ML models. Furthermore, we will conduct root-cause analysis for the identified abnormal network events, which is critical to automatically initiate mitigation operations. Our ultimate aim is to advance the state-of-the-art in anomaly detection techniques for efficient troubleshooting of

low-latency applications and for minimizing the impact of abnormal events on user QoE.

ACKNOWLEDGEMENTS

This work is partially funded by the French National Research Agency (ANR) MOSAICO project, under grant No ANR-19-CE25-0012.

REFERENCES

- [1] J. Cannady, "Artificial neural networks for misuse detection," in *National information systems security conference*, vol. 26. Baltimore, 1998, pp. 443–456.
- [2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.
- [3] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *SIGMOD '00*, 2000.
- [4] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Ki Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.
- [5] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.
- [6] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *NIPS*, 1999.
- [7] L. Ruff, N. Gornitz, L. Deecke, S. A. Siddiqui, R. A. Vandermeulen, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *ICML*, 2018.
- [8] GoodfellowIan, Pouget-AbadieJean, MirzaMehdi, Xubing, Warde-FarleyDavid, OzairSherjil, CourvilleAaron, and BengioYoshua, "Generative adversarial networks," *Communications of The ACM*, 2020.
- [9] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," *Proceedings of the 2018 World Wide Web Conference*, 2018.
- [10] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 38–44, 2015.
- [11] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.
- [12] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. M. Leung, and C.-H. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.
- [13] X. Marchal, P. Graff, J. R. Ky, T. Cholez, S. Tuffin, B. Mathieu, and O. Festor, "An analysis of cloud gaming platforms behaviour under synthetic network constraints and real cellular networks conditions," *Journal of Network and Systems Management*, vol. 31, 2023.
- [14] H. Iqbal, A. Khalid, and M. Shahzad, "Dissecting cloud gaming performance with decaf," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 5, pp. 1–27, 2021.
- [15] A. D. Domenico, G. Perna, M. Trevisan, L. Vassio, and D. Giordano, "A network analysis on cloud gaming: Stadia, geforce now and psnow," *ArXiv*, vol. abs/2012.06774, 2021.
- [16] M. Claypool and D. Finkel, "The effects of latency on player performance in cloud-based games," *2014 13th Annual Workshop on Network and Systems Support for Games*, pp. 1–6, 2014.
- [17] K. Raaen, R. Eg, and C. Griwodz, "Can gamers detect cloud delay?" *2014 13th Annual Workshop on Network and Systems Support for Games*, pp. 1–3, 2014.
- [18] R. Eg, K. Raaen, and M. Claypool, "Playing with delay: With poor timing comes poor performance, and experience follows suit," *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, 2018.
- [19] S. Vlahovic, M. Suznjec, and L. Skorin-Kapov, "The impact of network latency on gaming qoe for an fps vr game," *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–3, 2019.

- [20] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, 2009.
- [21] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: A comprehensive evaluation," *Proc. VLDB Endow.*, vol. 15, pp. 1779–1797, 2022.
- [22] J. Ren, F. Xia, Y. Liu, and I. Lee, "Deep video anomaly detection: Opportunities and challenges," *2021 International Conference on Data Mining Workshops (ICDMW)*, pp. 959–966, 2021.
- [23] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, pp. 6481–6494, 2020.
- [24] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection," *ACM Computing Surveys (CSUR)*, vol. 54, pp. 1–38, 2020.
- [25] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Muller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, pp. 756–795, 2020.
- [26] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021.
- [27] Y. Liu, Y. Zhou, K. Yang, and X. Wang, "Unsupervised deep learning for iot time series," *IEEE Internet of Things Journal*, pp. 1–1, 2023.
- [28] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1544–1551, 2018.
- [29] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.
- [30] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *SIGMOD '00*, 2000.
- [31] D. K. Nkashama, A. Soltani, J.-C. Verdier, M. Frappier, P.-M. Tardif, and F. Kabanza, "Robustness evaluation of deep unsupervised learning algorithms for intrusion detection systems," *ArXiv*, vol. abs/2207.03576, 2022.
- [32] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo, "An evaluation of anomaly detection and diagnosis in multivariate time series," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 2508–2517, 2022.
- [33] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," in *AAAI*, 2022.
- [34] R. Wang, C. Liu, X. Mou, K. Gao, X. Guo, P. Liu, T. Wo, and X. Liu, "Deep contrastive one-class time series anomaly detection," in *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 2023, pp. 694–702.
- [35] W.-S. Hwang, J.-H. Yun, J. Kim, and B. gil Min, "do you know existing accuracy metrics overrate time-series anomaly detections?," *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022.
- [36] N. Tatbul, T. J. Lee, S. B. Zdonik, M. Alam, and J. E. Gottschlich, "Precision and recall for time series," in *NeurIPS*, 2018.
- [37] K. Amano, N. Goda, S. Nishida, Y. Ejima, T. Takeda, and Y. Ohtani, "Estimation of the timing of human visual perception from magnetoencephalography," *The Journal of Neuroscience*, vol. 26, pp. 3981–3991, 2006.
- [38] J. R. Ky, B. Mathieu, A. Lahmadi, and R. Boutaba, "Assessing unsupervised machine learning solutions for anomaly detection in cloud gaming sessions," *2022 18th International Conference on Network and Service Management (CNSM)*, pp. 367–373, 2022.
- [39] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, 2020.
- [40] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et biophysica acta*, vol. 405 2, pp. 442–51, 1975.
- [41] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, 2015.
- [42] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 142–32 159, 2022.
- [43] N. Zhao, B. Han, R. Li, J. Su, and C. Zhou, "A multivariate kpis anomaly detection framework with dynamic balancing loss training," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.
- [44] J. Paparrizos, Y. Kang, P. Boniol, R. Tsay, T. Palpanas, and M. J. Franklin, "Tsb-uad: An end-to-end benchmark suite for univariate time-

series anomaly detection," *Proc. VLDB Endow.*, vol. 15, pp. 1697–1711, 2022.

- [45] S. S. Johari, N. Shahriar, M. Tornatore, R. Boutaba, and A. Saleh, "Anomaly detection and localization in nfv systems: an unsupervised learning approach," *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2022.
- [46] L. van der Maaten and G. E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
- [47] C. Cao, D. Chicco, and M. M. Hoffman, "The mcc-f1 curve: a performance evaluation technique for binary classification," *ArXiv*, vol. abs/2006.11278, 2020.
- [48] D. Fourure, M. U. Javid, N. Posocco, and S. Tihon, "Anomaly detection: How to artificially increase your f1-score with a biased evaluation protocol," in *ECML/PKDD*, 2021.



Joël Roman Ky is a Ph.D student with Orange Innovation and University of Lorraine (Nancy, France). He obtained his engineering degree in computer science and telecommunications at Toulouse INP-ENSEEIH, France and M.Sc degree in computer science from University of Toulouse INP, France (2021). His research interests are low-latency applications, cellular network and machine/deep learning techniques for network anomaly detection and root-cause analysis.



ferences TPC.

Bertrand Mathieu (Senior Member, IEEE) received the M.Sc. degree from the University of Marseille, the Ph.D. degree from the Sorbonne University, Paris, and the Habilitation à Diriger des Recherches degree from the University of Rennes. He is a Senior Research Engineer with Orange Innovation. He contributed to 14 national/European projects and published more than 80 papers in international conferences, journals or books. He is working on programmable networks, QoS and QoE and new network solutions. He is a member of several con-



machine learning techniques.

Abdelkader Lahmadi is Associate Professor of Computer Science at University of Lorraine, France. He is a permanent member of the RESIST research team at LORIA - INRIA Nancy Grand Est, working on security monitoring and management. He received a PhD degree in Computer Science (2007). He published in all major conferences in Network and Service Management. His research interests are mainly focusing on the security monitoring of networked systems including Software Defined Networks, IoT and SCADA systems by applying



Canada, the Canadian Academy of Engineering, and the Royal Society of Canada.

Raouf Boutaba received the M.Sc. and Ph.D. degrees in computer science from Sorbonne University in 1990 and 1994, respectively. He is currently a University Chair Professor and the Director of the David R. Cheriton School of Computer science at the University of Waterloo (Canada). He is the founding Editor-in-Chief of the IEEE Transactions on Network and Service Management (2007-2010) and served as the Editor-in-Chief of the IEEE Journal on Selected Areas in Communications (2018-2021). He is a fellow of the IEEE, the Engineering Institute of