Assessing Unsupervised Machine Learning solutions for Anomaly Detection in Cloud Gaming Sessions

Joël Roman Ky*†, Bertrand Mathieu*, Abdelkader Lahmadi† and Raouf Boutaba‡
*Orange Innovation Lannion, {joelroman.ky, bertrand2.mathieu}@orange.com
†Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France, abdelkader.lahmadi@loria.fr
‡David R. Cheriton School of Computer Science, University of Waterloo, rboutaba@uwaterloo.ca

***Abstract*—Cloud gaming applications have gained great adoption on the Internet particularly benefiting from the wide availability of broadband access networks. However, they still fail to meet users' quality requirements when accessed using cellular networks due to common wireless channel degradations. Machine Learning (ML) techniques can be leveraged to detect such anomalies during users' cloud gaming sessions. In this respect, unsupervised ML approaches are particularly interesting since they do not require labeled datasets. In this work, we investigate these approaches to understand their performance and their robustness. Our dataset consists of game sessions played on the public Google Stadia Cloud Gaming servers. The game sessions are played using a 4G network emulation replicating the capacity variations sampled on a commercial 4G network. We compare different models ranging from traditional approaches to deep learning and we evaluate their default performance while varying the level of contamination in their training datasets. Our experiments show that Auto-Encoders models achieve the best performance without contamination while the OC-SVM and the Isolation Forest are the most robust to data contamination.**

***Index Terms*—cloud gaming, anomaly detection, unsupervised learning, AI, QoE, mobile networks**

I. Introduction

Many emerging low-latency applications (cloud-gaming, tactile internet, metaverse...) are being widely deployed on Internet. In the area of Cloud Gaming (CG) for instance, tech giants like Google, Microsoft and Amazon have lunched their platforms, respectively Stadia[1], xCloud[2] and Luna[3].

The stringent bandwidth, latency and jitter requirements of these applications and the expected increase of traffic in the forthcoming years [1], exacerbate the high strain on current network infrastructures, even more on cellular networks. Many gaming blogs and discussions[4] point on network issues encountered when using cloud gaming. Therefore, Internet Service Providers (ISP), to provide cloud gaming users their expected quality of experience, need to deploy proper mechanisms for detecting and mitigating the possible anomalies that can hinder the quality of cloud gaming sessions such as drops in resolution, frame rate changes or screen freeze occurrences, which are clearly noticeable to cloud gaming users.

Detecting abnormal network behaviors traditionally requires experts knowledge. With the increasing complexity of contemporary networks and the increase of network data, traditional approaches have become impractical. Recently, Machine Learning (ML) based-approaches have proven to be effective in anomaly detection. Among the ML-based approaches, supervised learning, require large labeled samples for training to be efficient. However, data labeling by network experts is a cumbersome process and faulty samples are scarce in production networks [2]. To bypass the need for labeled observations, Unsupervised Learning approaches are progressively adopted for anomaly detection since they learn to identify anomalies from unlabelled data.

In the literature, only few works have focused on the detection of CG session quality degradation leaving out many open questions. This paper aims at addressing some of these questions. Specifically, it presents an evaluation of unsupervised machine learning approaches for the detection of cloud gaming performance degradation. A particular emphasis of our study is on the detection of anomalies based on multivariate time-series KPIs datasets of cloud gaming sessions under different 4G network conditions. The game sessions are played on the public Google Stadia platform and the game traffic is routed through the Internet to be processed by Stadia's servers. The 4G network conditions are emulated with the Mahimahi tool [3], based on real cellular network conditions from a commercial french ISP. We assess the ability of the models to efficiently detect anomalous behaviours as well as their robustness against data contamination. Unsupervised ML approaches for anomaly detection assume that the data used for training is free from anomalies and contain only *normal* instances. However, the data collected in real-life scenarios is rarely clean and it can be very cumbersome to remove the anomalous samples from the training data. In this paper, we also show how different data splitting strategies and evaluation procedures can impact the achievable results.

The remainder of this paper is organised as follows. Section II discusses previous works on cloud gaming performance analysis studies and unsupervised ML methods for anomaly detection. Section III presents the experimental protocol used to collect the datasets while Section IV describes our methodology for the evaluation of the ML algorithms. We present the evaluation results in Section V and conclude the paper by outlining future work in Section VI.

[1]https://stadia.google.com/
[2]https://www.xbox.com/fr-FR/xbox-game-pass/cloud-gaming
[3]https://www.amazon.com/luna/landing-page
[4]https://www.snapt.net/content-hub/the-challenge-of-cloud-gaming

978-3-903176-51-5 © 2022 IFIP

II. Related works and Background

A. Cloud gaming performance analysis studies

There are significant works on cloud gaming in the literature. Carrascosa et al. [4] analyzed Google Stadia, to show how it works and the characteristics of its traffic under normal and limited networks conditions. Di Domenico et al [5] and Graff et al. [6] went farther by performing a comparative analysis of the behaviour of different cloud gaming platforms (Stadia, GeForce Now and PS Now for the former and by including XCloud for the latter). They introduced artificial network impairments and showed that each platform can exhibit a different behaviour regarding its downlink bitrate adaptation. These previous studies focused mainly on the behavior of the CG platforms and how they react to network changes. Ta-Chen et al. [7] proposed a latency measurement approach that can be used on closed and proprietary CG platforms. They measured the game delay, which is the time elapsed between pressing the *ESC* key and the display of the menu on the screen. Their method was applied to Cloud-Union, a Chinese CG platform (now outdated) to characterize the video latency and unveil internal insights on this platform. However, Iqbal et al. [8] showed that the method of Ta-Chen et al. end up underestimating the game delay. They therefore proposed an automatic tool, DECAF, to measure efficiently the game delay, with a bot inserting timestamps at each input commands and recording the screen. This way, they were able to characterize and dissect the game delay and the video performance of Stadia, GeForce Now and Luna platforms.

The aforementioned works, were mainly centered on wired networks. Works on cloud gaming in cellular networks are rare. Kämäräinen et al. [9] performed measurement studies on the latency experienced by the users on a mobile cloud gaming platform. They showed by breaking the game latency into its components that the mobile device and the access network are the impactful components in a 4G network. Di Domenico et al. [5] conducted many experiments on 3G/4G mobile networks to measure the network capacity, under different signal quality, by using speed test measurements to observe if it could sustain a game session on Stadia. They also measured the performance of Stadia (frame rate, resolution) under emulated mobile networks. One of the most recent works is Bhuyan et al. [10], where the authors conducted an end-to-end study of mobile cloud-gaming applications under 5G network and WiFi. By using an open-source application, they measured the latency, frame-rate, PSNR (Peak Signal to Noise Ratio) and energy consumption under different network scenarios and assessed the effect of 5G and WiFi on the QoS and the video streaming performance of CG applications. Also worth noting are the works on cloud VR (Virtual Reality) gaming, which is a low-latency application with stringent requirements on bandwidth and latency, on mobile networks. Tan et al. [11] showed in their measurement study that the main bottlenecks that affect latency and prevent from playing cloud VR games in 4G networks are LTE's signaling operations like inter-protocol incoordination and single-protocol overhead. Zhang et al. [12] setup a real testbed of a simplified cloud VR streaming application over a 4G networks. They collected network traces and LTE network signaling traces and showed the impact of congestion, radio configurations and scheduling algorithm on the network latency.

Previous works mostly focused on performance evaluations of cloud-gaming applications under different networks conditions through measurements campaigns. They analyzed how network conditions can harm the performance of those applications. There is however, to the best of our knowledge, no works on the detection of performance degradation of CG applications induced by networks conditions.

B. Unsupervised Learning models for anomaly detection

Anomaly detection is a well-covered subject and many surveys [13], [14] proposed different taxonomies encompassing many of the existing approaches. Blazquez-Garcia et al. [15] proposed a taxonomy of anomaly detection approached for time-series data in an unsupervised learning context. They categorized different approaches based on the nature of the input data, the type of anomaly (referred to as outlier in the survey) and the type of method. Focusing on unsupervised learning approaches for multivariate time-series data, Audibert et al. [16], proposed a different taxonomy by classifying the methods into conventional methods, machine-learning methods and deep-learning methods. The conventional methods, such as VAR (Vector Auto Regressive) model, PCA (Principal Components Analysis) or SSA (Singular Spectrum Analysis), rely on statistical rules or assume that the time-series data come from a (linear) model whose parameters are estimated. For instance, the PCA performs a lossy reconstruction of the data by using the first p principal components to decompose and reconstruct the data. The reconstruction error is defined as the squared error between the original data and the reconstructed data.

The traditionally used machine learning methods rely on different categories ranging from distance-based ones like k-nearest neighbors, isolation-based ones like the Isolation Forest [17] to classification-based ones like OC-SVM (One-Class Support Vector Machine) [18]. Isolation Forest (iForest) detects anomalies by *isolating* them. It builds an ensemble of Isolation Trees by recursively performing a random partition of the observations until the anomalous observations are isolated. OC-SVM learns the smallest hypersphere that encompasses all normal data. The anomalous samples are those that lie beyond that hypersphere.

Due to their performance on different learning tasks, the methods that have more interest in the literature are those based on neural networks and deep-learning methods. Recent methods that obtain state-of-the-art results on benchmark time-series datasets are based on Auto-Encoders (AE). Auto-Encoders learn how to reconstruct input data from a low-dimensional representation (referred as latent variable) by minimizing the reconstruction error and use an anomaly score (that is the reconstruction error) to identify anomalies. Many variants of Auto-Encoders were developed for anomaly detec-

tion. For instance, the LSTM-VAE [19] uses a LSTM (Long Short-Term Memory) neural network architecture, to model temporal dependencies in the data, and a VAE (Variational Auto-Encoder) to estimate the latent variable and the distribution of the data. The anomalous samples are those with a low log-likelihood. DAGMM (Deep Auto-encoder Gaussian Mixture Model) [20] model combines an auto-encoder with a Gaussian mixture model and then can model the probability distribution of the output samples. OmniAnommaly [21] uses a VAE, a planar NF (Normalizing Flows), a Gaussian state model and a GRU (Gated Recurrent Units which is a LSTM variant) to learn temporal dependencies and stochasticity of the multivariate time-series for anomaly detection.

The methods discussed above were applied to detect anomalies based on time-series datasets for different application domains (industry, server machines, engines, web applications...). To the best of our knowledge, none of these works evaluated anomaly detection in cloud gaming sessions which is the focus of this paper.

III. Experimental Setup

The following section describes our experimental setup to generate the emulated 4G network conditions and our testbed to collect the CG sessions datasets.

A. 4G network conditions measurements

The LinkShell tool[5] from Mahimahi platform [3] can be used to emulate fixed or time-varying network conditions through log files called transmission opportunities (txops) files. However, the txops files available on Mahimahi Github repository[6] are not representative of current 4G networks capacities (dated from 2016 with a maximum capacity of about 5-10 Mbps). We thus generate new txops files, by using the Saturator tool developped by Winstein et al. [22].

The Saturator tool[7] allows to capture the behavior of a commercial base station for one user, by saturating the radio link between a client and a server. We use a Ubuntu 20.04 laptop (client) and a cloud Ubuntu 18.04 (server) that exchange packets using a 4G connection provided by 2 Xiaomi User Equipments (UE). By recording the timestamps of the packets sent and the acknowledgements received, we can generate the uplink and downlink txops files where each line matches a time in milliseconds when the real base station can deliver a MTU-sized packet.

The new txops files are generated on 6 different conditions (5 static ones and one in a mobility scenario on the highway) on Orange 4G network in Lannion (France). The static conditions differ in the mean downlink throughput that can be achieved (from 40 Mbps to 220 Mbps). The txops files data and further details of this experiment are available on this website[8].

[5]http://mahimahi.mit.edu/
[6]https://github.com/ravinet/mahimahi
[7]https://github.com/keithw/multisend/
[8]https://cloud-gaming-traces.lhs.loria.fr/cellular.html

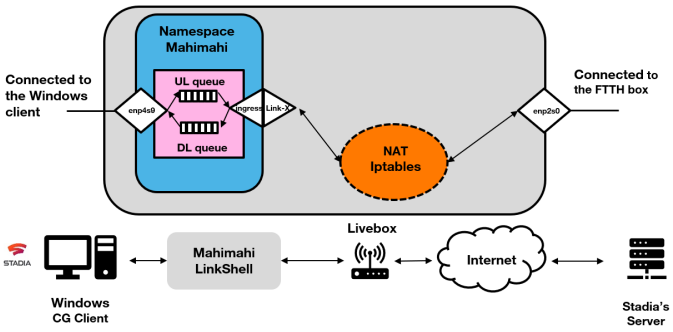

Fig. 1. CG measurements testbed.

B. Testbed and data collection

With the txops files collected, we perform the measurements of the CG time-series KPIs. We set up a testbed, depicted in Fig. 1, consisting of a Windows 10 laptop client (where the games are played through the Chromium desktop) and a Ubuntu 20.04 laptop as a network emulator (where the Mahimahi platform is installed). The latter can access the cloud gaming servers through a FTTH (Fiber-To-The-Home) Internet connection to avoid shifting the bottleneck from the network emulator to the WAN interface. Thereby, during the game sessions, the incoming uplink/downlink packets in the network emulator are queued and released by the LinkShell tool, in accordance with the transmission opportunities files representing the behavior of the 4G collected network conditions.

We perform the measurements on Google Stadia, playing 15 minutes the racing game Dirt 4 in each of the 6 network conditions. We use Google Stadia on the Chrome browser which uses WebRTC API for real-time communication. The CG KPIs are collected through the Chromium WebRTC API provided by the DECAF tool [8]. Redundant information are removed from the KPIs and only meaningful ones for CG quality are kept. For our evaluation, we use 14 metrics (from DECAF and ours) including frame rate, resolution, bitrate, network RTT, freeze occurrences, frames dropped, video rendering jitters, client-processing delays and txops downlink throughput.

IV. Methodology

In this section, we formulate the problem addressed in this paper and we describe our data splitting strategies, our training and evaluation procedure with the selected metrics.

A. Problem formulation

According to Chandola et al. [13], anomalies (or outliers) can be classified into three classes: point anomalies (individual anomalous observations), contextual anomalies (observations considered as anomalous in a specific context) and collective anomalies (group of anomalous observations). In this work, we focus on point anomalies in cloud gaming sessions and leave the other classes of anomalies for future work.

To detect anomalies, we use unlabelled KPIs datasets composed of multivariate time-series which are sequences of data

observations $x = \{x_1, x_2, ..., x_T\}$ where T is the length of x, and an observation $x_t \in \mathbb{R}^m$ corresponds to a m-dimensional vector at the time t. Unsupervised Learning anomaly detection task on multivariate data consists of learning a model given the set of *normal* observations x and assigning for each unseen observation $\tilde{x_t}$ a binary variable $y_t \in \{0; 1\}$ based on a anomaly score that measures how much the unseen observation differs from the set of normal observations. Hence, $y_t = 1$ if the observation is an anomaly and $y_t = 0$ otherwise.

Since our goal is to compare different unsupervised learning approaches, ground truths are required to assess the performance of each model using well-known ML performance metrics (F1-score, Precision, Recall). Hence, we choose to define as an anomaly, the observation that satisfies the following criteria, recommended by CG platforms for high quality streaming[9] [10]: either the frame rate is below 60 FPS, the resolution is below 1080p or a freeze occurs.

B. Data processing and splitting strategies

The features of our datasets are the DECAF metrics and ours related to the 4G emulated network. The DECAF metrics are logged and stored on the instrumented Chromium, each time a video frame finishes the frame processing pipeline stage in WebRTC. Since we include our own metrics which do not have the same temporal alignment, we need to resample the data to have a fixed time-step of 5ms between each observation. The features of the datasets are normalized by removing the mean and scaling to unit variance with Standard Scaler[11] before the training step.

To obtain the training set and test set, we use the same methodology as Zhong et al. [20]: all the data collected are merged and split into train/test as follows. The training set contains 50% of the normal samples of the whole dataset and the remaining 50% are in the test set. The test set additionally includes $p = 60\%$ of the anomalous samples and the other $(1 - p)\%$ are in a set called contamination set. In the experiments, $\delta\%$ (with $\delta \in \{0, 5, 10, 20, 50, 100\}$) of the data coming from this contamination set are fed back into the training set to identify how the presence of anomalous samples can impact the training and the performance of the models. We use this training strategy by assuming that *normal* samples in cloud gaming sessions can also be collected in low throughput network conditions. The models then, learn to detect anomalous sessions even on 4G network conditions with lower throughput. In the rest of the paper, this data splitting strategy is referred to as the *mixed-dataset* strategy.

We also choose a different data splitting strategy to evaluate its impact on model performance. We assume that normal observations in cloud-gaming session occur only at higher downlink bitrates (> 100 Mbps). The data collected in the networks conditions with an average downlink throughput higher than 100 Mbps are used as the training set and all the data coming from the other conditions as the test set. Anomalous samples from the training set are removed and used as the contamination set from which some samples are fed back into the training set in the same way as mentioned previously. The issue with this strategy is that the datasets are severely imbalanced: the data used for the test set contains many anomalous samples and the evaluation of the models on this test set does not allow to assess the performance on observations coming from the "best" networks conditions. This splitting is referred to as the *high-bitrate* one.

TABLE I
COLLECTED DATASETS

Data split Strategy	# Train instances	Train Anomalies ratio (%)	# Test instances	Test Anomalies ratio (%)
Mixed-dataset	138021	48.8	171704	58.85
High-bitrate dataset	168408	34.56	141318	77.98

Table I provides a summary of the number of instances of the train/test set with their respective anomaly ratio for each splitting strategy.

C. Models training and implementations

This section describes the models implemented and the details of their implementations.

- **PCA:** It belongs to the conventional algorithms and we use it as a baseline model in our evaluation. The training time of this model is relatively low due to the SVD (Singular Value Decomposition). We choose a number of principal components that preserve 90% of the variance of the data to perform the lossy reconstruction.
- **OC-SVM:** In the category of ML approaches, OC-SVM is a shallow model with high performance in the anomaly detection tasks. We kept the default hyper-parameters like the *radial basis function* kernel and $\nu = 0.5$ value as the upper bound of the fraction of outliers in the training set.
- **Isolation Forest:** iForest, belongs also in the category of ML approaches. It achieves good performance with a nearly linear computational complexity. We kept the numbers of iTrees at 100 and the ratio of outliers in the training set at 0.1.
- **Auto-Encoder:** As mentioned in Section II, the AE is a popular deep learning approach for anomaly detection. We implement a feed-forward AE with *Tanh* activations functions.
- **LSTM-VAE:** Among the state-of-the-art anomaly detection models, we select the LSTM-VAE, that combines a VAE, an improvement of AE with bayesian inference and a LSTM, to model temporal dependencies. We write our custom implementation in PyTorch from different Github available implementations[12][13].

For the two DL models (Auto-Encoder, LSTM-VAE), the Adam optimizer with a learning rate of 10^{-3} and a batch

[9]https://support.google.com/stadia/answer/9607891?hl=fr/

[10]https://www.nvidia.com/en-us/geforce/products/geforce-now/system-reqs/

[11]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html/

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

[13]https://github.com/paya54/Anomaly_Detect_LSTM_VAE

size of 128 are used. To avoid overfitting of the models, an early stopping strategy with a patience of 10 is implemented. The base implementations of PCA, IForest and OC-SVM are from Scikit-Learn while the Auto-Encoder and LSTM-VAE are from our custom implementation with PyTorch. The details, the code and the hyper-parameters values of our implementation are available on Github[14].

For each model, we perform 5 training runs, each with one of the two aforementioned data splitting strategies. This allows us to report reliable results since the induced variability of the random sampling in the splitting and the parameters initialization is taken into consideration.

D. Performance metrics

To evaluate the performance of our models, we select the Precision (P), the Recall (R) and the F1-Score (F1) computed as follows:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad F1 = 2.\frac{P.R}{P + R}$$

were TP denote the True Positives, FP the False Positives and FN the False Negatives. Our evaluation also includes the AUC (Area Under the Curve) metric which represents the probability that the model ranks higher a random anomalous sample than a random normal sample. To be computed from anomaly score, those metrics (except the AUC) require a threshold. A simple threshold like the 3σ rule of thumb (which is an empirical choice, commonly used in anomaly detection [23]) is used: during the training phase, the mean μ_{train} and the standard deviation σ_{train} values of the reconstruction error $error_{train}$ are computed and stored. At the testing phase, an observation is classified as an anomaly if :

$$\|error_{test} - \mu_{train}\| > 3.\sigma_{train}$$

V. Evaluation and Results

A. Performance with the mixed-dataset

Table II reports the precision, recall and F1-score of the five models included in our study, with the different anomaly contamination ratio δ on the mixed-datasets. The F1-score is a harmonic mean of the Precision and the Recall. The Precision assess the ability of the model to precisely detect the anomalies and the Recall its ability to detect all the anomalies. The best model will therefore have higher Precision and Recall scores. All the models have their best F1-score with no contamination ($\delta = 0$) except for the iForest model. The AE and LSTM-VAE models have the best score (F1-score $= 88\%$) and this results from high Precision ($= 98\%$) and Recall ($= 80\%$) values. We can have high confidence on the detected anomalies but due to the lower Recall value, some of the anomalies of the test set may be missed by these models.

The iForest model has the worst F1 score when there is no contamination (F1$= 1\%$) due to a lower Recall. In fact, the model does not have the ability to isolate the anomalous samples since it assumes that there are few anomalies, and they are different from normal samples in the training dataset. It uses this information to isolate the anomalies. The OC-SVM, similar to iForest, assumes a certain proportion of anomalies in the training set and performs better with no contamination (F1$= 74\%$). It efficiently detects most of the anomalies in the test set (R$= 98\%$) but identifies some normal observations as anomalous (P$= 59\%$). It presents the best Recall score among all the evaluated models. The PCA model also presents bad performance due to a low ability to detect all the anomalies in the test set (R$= 8\%$).

Nevertheless, the results of all the models, except the iForest, drop with the increase of the contamination ratio. The performance of the AE and LSTM-VAE plummets as soon as anomalies are injected (from 88% to 1%) while the results of the PCA remain low and decrease as the contamination ratio increases. The bad results of the PCA, AE and LSTM-VAE could be explained by the fact that those models need only normal samples to be efficiently trained. With the increase of anomalous samples, the models can not detect all of them in the test set. This leads to low Recall values. For our evaluation, we use the 3σ rule for the anomaly detection threshold, but this rule assumes that the reconstruction errors can be modelled by a normal distribution, which is not always the case. With

TABLE II
Overall performance (mean and standard deviations over the 5 runs) on the mixed-datasets strategy. δ is the anomaly contamination ratio in the training dataset.

		Mixed-datasets		
	δ (%)	Precision	Recall	F1-score
PCA	0	82.01±0.14	8.83±0.1	15.94±0.16
	5	88.76±4.7	5.77±1.56	10.8±2.79
	10	83.92±2.75	5.74±0.9	10.73±1.6
	20	73.44±1.07	4.00±0.23	7.59±0.42
	50	65.36±0.89	2.97±0.12	5.69±0.22
	100	53.72±0.45	1.82±0.02	3.53±0.03
iForest	0	68.18±2.8	1±0.41	1.97±0.8
	5	62.18±0.58	89.77±1.96	73.47±0.96
	10	63.19±0.35	81.5±3.34	71.15±1.34
	20	68.44±1.05	62.53±5.47	65.23±3.32
	50	77.21±2.03	20.85±4.29	32.58±5.35
	100	74.61±3.25	1.76±0.74	3.43±1.42
OC-SVM	0	59.29±0.01	98.59±0.07	74.01±0.02
	5	59.5±0.02	97.82±0.14	74±0.03
	10	59.86±0.02	95.52±0.23	73.6±0.07
	20	60.51±0.04	88.65±0.25	71.93±0.07
	50	60.98±0.05	68.65±0.54	64.59±0.24
	100	60.28±0.05	54.47±0.31	57.23±0.19
AE	0	99.02±0.05	79.65±0.06	88.28±0.03
	5	95.55±0.43	7.86±0.07	14.52±0.12
	10	94.09±0.79	5.02±0.08	9.53±0.14
	20	91.45±1.28	3.00±0.07	5.80±0.14
	50	80.61±1.86	1.44±0.04	2.83±0.07
	100	77.68±1.12	1.28±0.03	2.52±0.06
LSTM-VAE	0	98.44±0.76	80.27±1.3	88.42±0.83
	5	98.58±0.7	2.25±1.02	4.38±1.95
	10	98.43±4.01	1.15±0.8	2.25±1.56
	20	83.95±25.31	0.83±0.43	1.65±0.83
	50	88.83±14.22	0.72±0.37	1.43±0.74
	100	95.37±2.21	0.75±0.26	1.48±0.51

[14]https://github.com/joelromanky/cg-ano-detect-eval

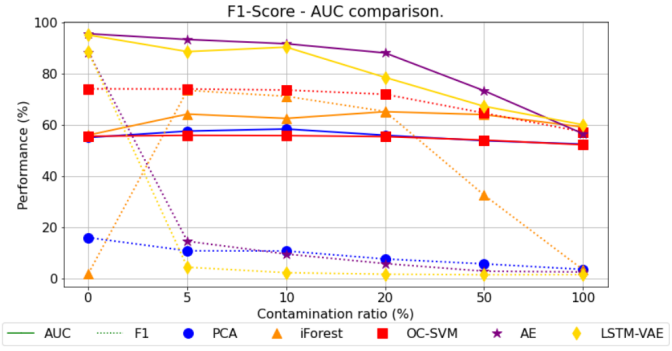

Fig. 2. Comparison of F1-Score and AUC.

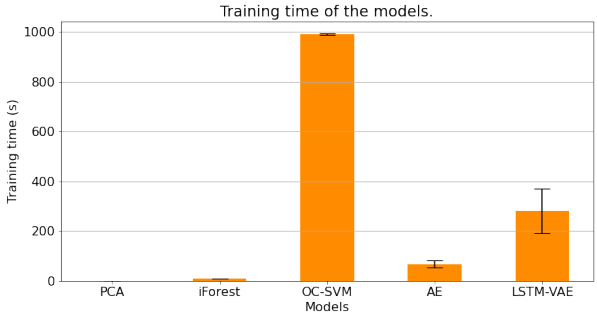

Fig. 3. Models mean training time with the standard deviations over 5 runs

TABLE III
OVERALL PERFORMANCE (MEAN AND STANDARD DEVIATIONS OVER THE 5 RUNS) ON THE HIGH-BITRATE TRAINING SET STRATEGY. δ IS THE ANOMALY CONTAMINATION RATIO IN THE TRAINING DATASET.

		High-bitrate training datasets		
	δ (%)	Precision	Recall	F1-score
PCA	0	98.01±0	10.89±0	19.6±0
	5	95.36±3.64	7.38±0.24	13.7±0.42
	10	97.84±1.25	8.1±0.65	14.96±1.11
	20	90.90±2.87	7.89±0.21	14.51±0.36
	50	89.88±0.34	5.85±0.2	10.99±0.36
	100	92.21±0	5.93±0	11.14±0
iForest	0	88.92±1.33	38.21±4.47	53.33±4.56
	5	98.18±0.42	88.94±0.56	93.33±0.26
	10	98.63±0.48	88.21±0.4	93.13±0.16
	20	98.56±0.15	89.16±0.51	93.62±0.26
	50	99.37±0.13	87.5±0.29	93.06±0.12
	100	99.16±0.48	78.3±4.26	87.43±2.57
AE	0	99.68±0.11	86.77±0.22	92.77±0.08
	5	99.43±0.12	8.48±1.87	15.57±3.26
	10	99.28±0.29	3.95±0.52	7.58±0.97
	20	99.23±0.08	1.82±0.27	3.58±0.53
	50	99.03±0.28	0.77±0.02	1.53±0.04
	100	99.17±0.01	0.76±0.01	1.51±0.01
LSTM-VAE	0	99.79±0.1	86.59±0.72	92.72±0.41
	5	97.5±3.47	7.59±11.53	12.03±16.95
	10	89.7±19.85	1±0.54	1.98±1.06
	20	99.49±0.16	1.12±0.54	2.21±1.06
	50	94.32±9.62	0.93±0.3	1.83±0.59
	100	99.49±0.27	1.55±1.43	3.02±2.72

the presence of anomalies in the training set, this threshold turns out to be too high, and thus prevents the models from detecting all the anomalies. Previous works such as [16], [24], tested different thresholds and reported those that led to the best F1-score which explains why they got better results. On the other hand, the performance of iForest model reaches the same performance level as OC-SVM but decreases afterwards with the increase of the contamination ratio.

We compare in Fig 2 the F1-score and the AUC score of the models. Our results show high AUC score while the F1-score drops when the contamination ratio increases. For instance the AUC scores for the AE are from 95% to 56% while the F1-scores are from 88% to 2%. This shows how misleading the AUC score can be even if our test set is not too much imbalanced (58% of anomaly ratio) and confirms the conclusions drawn in [25] about the use of AUC with imbalanced datasets.

Apart from the performance of the models on the anomaly detection tasks, we also evaluate the training time of each model in a given configuration. These results are depicted in Fig 3. The OC-SVM has the largest training time and its inference time is also in the same magnitude. The PCA and the iForest have the smallest fitting time since the PCA performs only one SVD and the iForest has a log-linear complexity. The training time of AE and LSTM-VAE can be highly variable (high standard deviations values in Fig 3) and this is the result of the early stopping strategy that prevents the model to overfit the data during the training phase. The training is indeed stopped if the validation loss does not improve during 10 consecutive epochs.

Take-away: *The best models without data contamination are the AE and LSTM-VAE but they show poorer performance with data contamination. Since in real-life situations, the collected data always contain anomalies, the OC-SVM or the iForest should be preferred although the OC-SVM has a longer training time.*

B. Impact of the splitting dataset strategy

The performance of the models on the high-bitrate datasets is reported in Table III. We do not test the OC-SVM on this data splitting strategy due to the large training time of this model. The same findings, as those in the previous section, can be made. However, we note the better performance of iForest regarding the contamination ratio and the better performance of AE and LSTM-VAE when $\delta = 0$. Those results can be explained by the significant difference between the training set and the test set. The downlink bitrates of the training set are about 120-200 Mbps while those of the test sets are lower (40-80Mbps). This difference seems to favor iForest allowing it to easily isolate the anomalous samples. For the AE and LSTM-VAE, the performance reported here can be misleading about the efficiency of those models in dealing with the datasets, since as mentioned previously, the used threshold is sub-optimal and does not allow to catch all the anomalies. Consequently, the models can not be fairly compared and we

need to perform more evaluations and compare using the best configuration of each model. The variation of the performance from one splitting strategy to another raises the problem of how to define anomalous situations for cloud gaming session quality. A low resolution at lower bitrates can be interpreted as a normal situation because it is the best that the network architecture can provide at this given situation.

Take-away: *The high-bitrate splitting strategy improves the Recall of the AE, LSTM-VAE and the iForest models. iForest is the most robust to data contamination while reconstruction-based approaches show poorer performance.*

VI. Conclusion

This paper presented a comparative evaluation of unsupervised machine learning models for anomaly detection in cloud gaming sessions. It presents our analysis of the performance of different ML algorithms and our assessment of the robustness of these models in the face of contaminated datasets. Our results show that the OC-SVM is the most robust to the contamination but requires longer time for training and inference, while the iForest model requires the presence of contamination to be efficient. Our comparative study also shows that reconstruction-based approaches do not perform well with an empirical rule of thresholding as they fail to catch all the anomalies in the data. They require a good threshold value to be assessed fairly in a comparative study. For the detection of cloud gaming quality, we found that a thorough attention must be given to the data splitting and also to the definition of an anomalous situation.

As future work, we plan to perform additional evaluations of ML approaches, especially deep-learning approaches, with the goal of finding the optimal configuration leading to the best performance and robustness in anomaly detection in CG applications.

Acknowledgements

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

References

[1] M. Marsden, M. Hazas, and M. Broadbent, "From one edge to the other: Exploring gaming's rising presence on the network," in *Proceedings of the 7th International Conference on ICT for Sustainability*, ser. ICT4S2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 247–254. [Online]. Available: https://doi.org/10.1145/3401335.3401366

[2] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. E. Solano, and O. M. C. Rendon, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, pp. 1–99, 2018.

[3] R. Netravali, A. Sivaraman, S. R. Das, A. Goyal, K. Winstein, J. W. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for http," in *USENIX Annual Technical Conference*, 2015.

[4] M. Carrascosa and B. Bellalta, "Cloud-gaming: Analysis of google stadia traffic," *ArXiv*, vol. abs/2009.09786, 2020.

[5] 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.

[6] P. Graff, X. Marchal, T. Cholez, S. Tuffin, B. Mathieu, and O. Festor, "An analysis of cloud gaming platforms behavior under different network constraints," *2021 17th International Conference on Network and Service Management (CNSM)*, pp. 551–557, 2021.

[7] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C. L. Lei, "Measuring the latency of cloud gaming systems," *Proceedings of the 19th ACM international conference on Multimedia*, 2011.

[8] 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.

[9] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, and P. Hui, "A measurement study on achieving imperceptible latency in mobile cloud gaming," *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017.

[10] S. Bhuyan, S. Zhao, Z. Ying, M. T. Kandemir, and C. R. Das, "End-to-end characterization of game streaming applications on mobile platforms," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 6, pp. 1 – 25, 2022.

[11] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu, "Supporting mobile vr in lte networks: How close are we?" *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, 2018.

[12] Z. Zhang, S. Shi, V. Gupta, and R. Jana, "Analysis of cellular network latency for edge-based remote rendering streaming applications," *Proceedings of the ACM SIGCOMM 2019 Workshop on Networking for Emerging Applications and Technologies*, 2019.

[13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, 2009.

[14] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, pp. 949–961, 2017.

[15] A. Bl'azquez-Garc'ia, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys (CSUR)*, vol. 54, pp. 1 – 33, 2021.

[16] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Do deep neural networks contribute to multivariate time series anomaly detection?" *ArXiv*, vol. abs/2204.01637, 2022.

[17] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008.

[18] J. Z. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3, pp. 1741–1745 vol.3, 2003.

[19] 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.

[20] 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.

[21] 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.

[22] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *NSDI*, 2013.

[23] A. Lahmadi, A. Duque, N. Heraief, and J. Francq, "Mitm attack detection in ble networks using reconstruction and classification machine learning techniques," in *PKDD/ECML Workshops*, 2020.

[24] 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.

[25] 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.