Comparative Evaluation of Semi-Supervised Anomaly Detection Algorithms on High-Integrity Digital Systems

Gianluca Martino[1, 2], Arne Gruenhagen[1, 2], Julien Branlard[2], Annika Eichler[2], Goerschwin Fey[1], and Holger Schlarb[2]

[1]Hamburg University of Technology (TUHH), Hamburg, Germany
[2]German Electron Synchrotron (DESY), Hamburg, Germany

Abstract—**Anomaly detection algorithms solve the problem of identifying unexpected values in data sets. Such algorithms have been classically used for cleaning unlabelled data sets from potentially unwanted values. However, the ability to detect outlying values in data sets can also be used to detect anomalies in systems. Semi-supervised anomaly detection algorithms learn from data for known correct behavior. Such algorithms have been used in various fields, e.g., system security, fault detection, medical applications.**

In this paper, we use the Area Under the Receiver Operating Characteristic (AUROC) score to evaluate algorithms for semi-supervised anomaly detection when applied to high-integrity distributed digital systems. We identify the relevant parameter for each algorithm and observe how the parameter influences the score and the runtime.

Index Terms—**anomaly detection, novelty detection, outlier detection, semi-supervised, comparative analysis**

I. Introduction

Anomaly detection deals with the problem of identifying unexpected values in data sets. Anomaly detection is commonly used in very different fields such as intrusion detection [1]–[3], fraud detection [4], [5], medical applications [6], fault detection [7]–[9], and more [10]–[13]. Since very different fields use anomaly detection, a variety of different techniques have been developed over the course of the years. However, not all the techniques can be used for all possible data sets and application fields. This is because the nature and characteristics of the data vary significantly depending on the application field. As highlighted in [14], the algorithm's ability to detect anomalies depends on the characteristics of the data analysed, which are depending on the problem to be studied.

In high-integrity systems, traditional classification methods are not suited due to the low number of abnormalities occurring in such systems. The low number of anomalies, in addition to the difficulty of retrieving data describing all possible fault modes of such systems, makes semi-supervised anomaly detection, also referred to as novelty detection, the technique of choice for building self-adaptive high-integrity systems. This is because semi-supervised anomaly detection algorithms can classify data given a single class during the training phase.

We acknowledge the support by DASHH (Data Science in Hamburg - HELMHOLTZ Graduate School for the Structure of Matter) with the Grant-No. HIDSS-0002.

In this paper, we focus on analysing the performance of several anomaly detection techniques using real sensor data coming from a field-programmable gate array (FPGA) based subsystem of the European X-Ray Free-Electron Laser (XFEL). The subsystem considered is the digital portion of the Low-Level Radio Frequency (LLRF) system. In addition to that, the experiments are conducted using publicly available data describing other high-integrity systems. The contributions of this paper are twofold: on the one hand, we give a coherent description of the algorithms selected; since, to our knowledge, this has not been done for semi-supervised anomaly detection, such an organized description could help in identifying the strong and weak points of each algorithm for other applications. On the other hand, the evaluation of state-of-the-art algorithms applied to high-integrity digital systems can help in the design phase in choosing the correct algorithm for a given set of requirements.

The paper is organized as follows: in Section II, we present a selection of publications and describe the differences; in Section III, we introduce some concepts used later; in Section IV, we give a coherent description of the algorithms selected for the evaluation; in Section V, we describe and motivate the performance metric chosen; in Section VI, we describe the experiments and present the results; in Section VII, we conclude the work giving some general indications.

II. Related Work

Despite extensive literature about semi-supervised anomaly detection, existing literature gives no clear indication of the performance of existing algorithms when the application is monitoring high-integrity distributed digital systems.

In [15], the authors thoroughly compare and evaluate a wide range of algorithms using a set of publicly available data sets. The evaluation method used makes it easy to make new comparisons when new algorithms or new implementations are proposed. However, they consider only unsupervised anomaly detection methods. In [16], the authors perform an experimental evaluation of semi-supervised anomaly detection methods to deduct general guidelines on semi-supervised anomaly detection. In the experiments, they use a wide variety of data sets consisting of publicly available data. However, they perform the evaluation using only a few models representing whole classes of algorithms: a Gaussian Mixture model, for density estimation method; two k-nearest neighbor models; and an SVDD model, for support vector machine method. Also, the data sets chosen are not related to hardware systems. In [17], the authors conduct an experimental evaluation of novelty detection methods for discrete sequences. They consider a different set of algorithms with respect to the previous works. However, the main focus of their analysis is on stable discrete sequences. The use cases considered are protein identification for genomics, fraud and intrusion detection, and user behavior analysis. In [18]–[21], the authors surveyed state-of-the-art methods for outlier detection for temporal data and novelty detection. However, they do not evaluate the algorithms. Other comparisons can be found in other papers presenting specific algorithms, but the results give only partial indications.

III. Background

Anomaly detection techniques use data sets containing vastly varying information, e.g., temperature (real number), distance (non-negative number), activation status (binary class). Additionally, the same type of information could be represented in different ways, e.g., measuring temperatures using different measurement units. An important step for the anomaly detection algorithms is to represent all the different dimensions of the data set in a common reference frame. This pre-processing step is called normalization. In the experiments, we use Z-normalization, calculating the mean value and standard deviation on the training set.

Anomaly detection is usually categorized [15] as follows:

- supervised anomaly detection, when the algorithm uses a set of labeled training data for the detector's initial training. In this category, we can find algorithms traditionally used for pattern recognition;
- semi-supervised anomaly detection or novelty detection, when the algorithm requires a training phase using a data set containing only positive examples, i.e., correct behavior, or negative examples, i.e., anomalous behavior;
- unsupervised anomaly detection, when the data available are not labelled. In this case, the algorithm will use the same data for both the training phase and the inference phase.

In the following, we refer to a data set as an $m \times n$-matrix where m is the number of dimensions of the data, i.e., the values coming from all the different signals at each time step, and n is the number of time steps. A sample p is the vector of size m containing the values of all the dimensions in a single time step.

IV. Selected Algorithms

In the next sections, we will give a description of the algorithms and their implementation for semi-supervised anomaly detection.

A. K-Nearest Neighbors (KNN/aKNN)

K-nearest neighbors (KNN) [22], not to be confused with k-nearest neighbor classification, uses either the distance to the kth-nearest neighbor (KNN) or the average of the distances with the k-nearest neighbors (aKNN) to calculate a score. The aKNN score for a sample p is calculated as follows:

$$\mathit{aKNN}(p) = \frac{\sum_{\forall o \in N_k} \mathit{Dist}(p, o)}{k} \tag{1}$$

where k is the number of neighbors, N_k is the set of k-nearest neighbors, and $\mathit{Dist}(p, o)$ is the distance between the data point p and the data point o.

B. Local Outlier Factor (LOF)

The local outlier factor (LOF) [23] is an anomaly detection algorithm that gives a score based on the local densities of both the sample and the k-nearest neighbors. The LOF score for each sample p is defined as:

$$\mathit{LOF}(p) = \frac{\sum_{\forall o \in N_k(p)} \frac{LRD_k(o)}{LRD_k(p)}}{|N_k(p)|} \tag{2}$$

with:

$$\mathit{LRD}_k(p) = 1 / \frac{\sum_{\forall o \in N_k(p)} \mathit{ReachDist}_k(p, o)}{|N_k(p)|} \tag{3}$$

where $|N_k(p)|$ is the number of k-nearest-neighbors to p, $ReachDist_k(p, o)$ is the reachability distance between p and o, and *LRD* is the local reachability density. The implementation used in this paper is available as part of the framework Scikit-learn [24]. The usage as a semi-supervised anomaly detection algorithm involves a training phase that produces a threshold and a prediction phase where the threshold is used for the classification. The threshold is selected as the value of the cth percentile of the LOF score for the training set, where c is the estimated proportion of outliers in the data set.

C. Cluster-based Local Outlier Factor (CBLOF)

The cluster-based local outlier factor (CBLOF) [25] was proposed to solve the problem of not properly considering both clustering and outlier discovery in the data set with previous algorithms. The algorithm gives a score based on the distances between the clusters and the samples, and on the clusters' size. The CBLOF distinguishes between small and large clusters using the parameters α and β. In particular, the authors use the numeric parameters α and β to construct a formula that defines the boundary between small and large clusters.

The CBLOF is defined as:

$$\mathit{CBLOF}(p) = \begin{cases} \min_{C_j \in \boldsymbol{C}} \mathit{Dist}(p, C_j) & \text{if } p \in SC \\ \mathit{Dist}(p, C_i) & \text{if } p \in LC \end{cases} \tag{4}$$

where $p \in C_i$. The set $\boldsymbol{C}$ is the set of all clusters, SC and LC are the set of small clusters and large clusters, respectively, $\mathit{Dist}(p, C)$ is the distance between the data point p and the center of the cluster C.

For clustering, the authors propose the *Squeezer* algorithm [26], but any clustering algorithm can be used in practice.

D. Histogram-based Outlier Detection (HBOS)

The histogram-based outlier detection (HBOS) [27] is a statistical anomaly detection algorithm. This algorithm's main characteristic is that each dimension of the data is considered independent of the others. The HBOS is defined as:

$$\mathit{HBOS}(p) = \sum_{i=0}^{d} \log\left(\frac{1}{\mathit{Hist}_i(p)}\right) \tag{5}$$

where d is the number of dimensions of the data and $\mathit{Hist}(p)$ is the height of the bin of p after the normalization of the histograms, a density estimation of the data point p. The creation of the bins of a histogram should use the dynamic bins method, which means that the number of elements in a single bin depends on the element's values, such that larger values form smaller bins.

E. Angle-based Outlier Detector (ABOD/FastABOD)

Angle-based outlier detector (ABOD) [28] was designed for solving the issues that other algorithms have with high-dimensional data. The main idea is to consider the directions of the distance vectors between the points of the data set. If the spectrum of the observed directions by a point is wide, the point is inside (or close to) a cluster. Otherwise, it is an outlier.

The angle-based outlier factor (ABOF) is calculated considering each triplet of points and calculating:

$$\mathit{ABOF}(p) = \mathit{Var}_{\forall o', o'' \in \mathcal{D}}\left(\frac{\langle \overline{po'}, \overline{po''}\rangle}{\left\|\overline{po'}\right\|^2 \cdot \left\|\overline{po''}\right\|^2}\right) \tag{6}$$

where $\mathcal{D}$ is the data set, $\overline{po'}$ and $\overline{po''}$ represent the distance vector from p to o' and from p to o'', respectively, $\|\cdot\|$ is the norm of a vector, and $\langle \cdot, \cdot \rangle$ is the scalar product of two vectors.

To use this algorithm as a semi-supervised anomaly detection algorithm, the data points to which each sample is compared are only the ones of the training set.

Since the complexity is $O(n^3)$ due to the need to consider each triplet in the data set, the FastABOD solves this problem by restricting the set of points to be considered for each point to the k-nearest neighbors.

F. Minimum Covariance Determinant (MCD)

The minimum covariance determinant (MCD) estimator [29] is a robust estimator of a data set's covariance. MCD is used to estimate the covariance matrix of a portion of the training data such that the determinant results minimal. The idea is that the covariance matrix having the minimal determinant represents the set of data closer to each other.

The score is then the Mahalanobis distance between the sample considered and the distribution represented by the covariance matrix:

$$D(p) = \sqrt{(p - m)^T \cdot C^{-1} \cdot (p - m)} \tag{7}$$

where m is the average of the subset of the training data used to construct the covariance matrix, and C^{-1} is the inverse of the covariance matrix.

G. One-class Support Vector Machine (OCSVM)

A classifier that is based on a one-class support vector machine (OCSVM) [30] constructs a hyperplane and finds a linear boundary on the hyperplane. The classifier separates between anomalous data and non-anomalous data using the boundary defined. The hyperplane constructed, also known as kernel space, is commonly obtained from the training data using a transformation based on the radial basis function (RBF) kernel [31].

After having a defined transformation function, the decision function gives a classification score based on the sample's position on the hyperplane with respect to the linear boundary defined.

H. Principal Component Analysis (PCA)

Principal component analysis (PCA) [32] is commonly used for the reduction of the data dimensionality by identifying the principal components and then using the main ones discarding the others. The same technique can be used for anomaly detection [33]. In this case, the entity of deviation from the components identified can be used as an anomaly score.

For calculating the set of principal components, which is the set of new variables obtained by linearly combining the random variables of the data, an eigenanalysis of the covariance matrix is sufficient. The new set of variables will be uncorrelated and ordered from the component with the highest variance to the component with the least variance.

The usage for semi-supervised anomaly detection requires a training phase where the set of principal components are identified. Using the new data, the score is then calculated as follows:

$$\mathit{PCA}(p) = \sum_{\forall_{c \in C}} \frac{\mathit{Dist}(p, c)}{c_w} \tag{8}$$

where $c \in C$ is a component, part of the set of components C, identified during the training phase and c_w is the component weighted by the eigenvalue.

V. Performance metric

In order to be able to communicate the result of the evaluation in a synthetic way, we use a score that gives an idea of the performances of each algorithm. For this reason, the Area Under the Receiver Operating Characteristic (AUROC) score was adopted. The AUROC score takes into account both the capability of the algorithms to detect anomalies, i.e., distinguishing between classes, and the scores assigned to each datapoint [34]. It is defined as the area underneath the ROC curve and ranges between 0 and 1. An AUROC score of 1 represents a predictor whose predictions are $100\,\%$ correct. An AUROC score of 0 represents a predictor whose predictions are $100\,\%$ wrong, i.e., opposite predictions. Finally an AUROC score of 0.5 represents a predictor whose predictions are random guesses.

The AUROC score is the most used performance metric for evaluating classifiers when the data to be analysed present an imbalance between the classes [35]. This is because the metric can cope with the imbalances of the data without showing a bias. However, the main factor to be considered for the choice of the performance metric is the focus on classification success shown by this metric [36], [37]. Like the one presented in this paper, some applications can tolerate classification errors, and the score does not mask poor performances. Different applications, i.e., medical diagnostic, should instead use different metrics.

VI. Evaluation

The experiments were performed using the Python libraries PyOD [38] and Scikit-learn [24]. The runtimes were measured on a system running Python 3.7 on Windows 10 with a processor Intel(R) Core(TM) i7-8750H @ 2.20GHz and 32 GB of RAM.

A. Datasets Summary

The evaluation of semi-supervised anomaly detection algorithms requires two sets of unlabelled data:

- a training set used for training the detector, containing only positive samples, i.e., non-anomalous data;
- a testing set used for the inference phase, including both normal data and anomalies;

In the following paragraphs, we give a brief description of each data set used for the evaluation.

xfel-dcm This data set was extracted from the European X-Ray Free-Electron Laser (XFEL). When the data was recorded, a component of the LLRF subsytem was turned off for a short period of time, and then turned on again . The signals include temperature, current, voltage and load of the power suppliers.

xfel-door Also this data set was extracted from XFEL. The data describes another anomaly artificially created. In this case, a door of the rack containing one of the LLRF stations was opened. The signals used include temperature, current, load and voltage of the system contained in the opened rack.

xfel-pcie In this case, the data describes a failure in the PCI Express interface that stops all communication between a specific board and the host system. The signals used include temperature, current, and voltage of the failing board.

shuttle This data set containing data from the shuttle statlog was obtained from the UCI Machine Learning Repository [39]. The 9 dimensions of this data set describe the radiator position during flight. For the experiments, all anomalous data points were removed from the training set.

Table I shows a summary of the properties of the data sets used. For all the experiments, the data sets were normalized using Z-normalization [40].

TABLE I: Data sets summary

Data set	Dimensions	Training	Testing	Outliers	%Outliers
xfel-dcm	70	40000	100000	5188	$5.19\,\%$
xfel-door	83	40000	166837	40617	$24.35\,\%$
xfel-pcie	12	43308	57743	15731	$27.24\,\%$
shuttle	9	34108	14500	3022	$20.84\,\%$

B. Algorithms Parameters

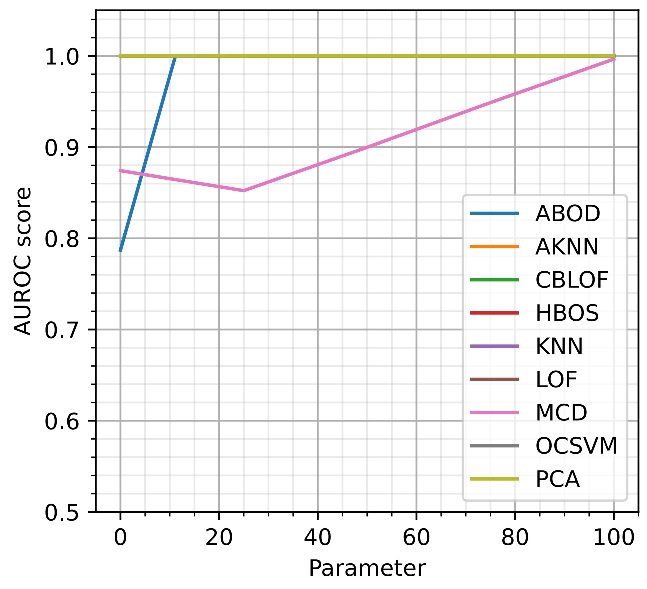
All the algorithms selected contain a controllable parameter. The value of the parameter influences different aspects of the calculations, both during testing and inference. A summary of all the parameters used is presented in Table II.

TABLE II: Summary of the algorithms' parameters

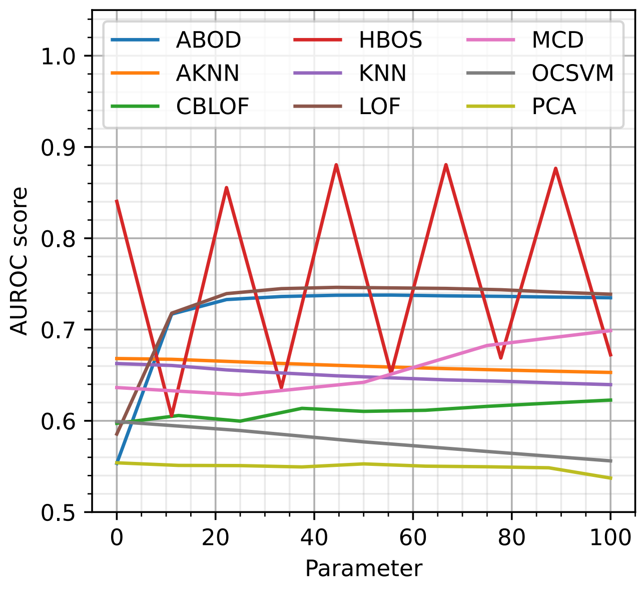
Algorithm	Parameter	Interval	Step Size
ABOD	#neighbors	5 - 50	5
AKNN	#neighbors	5 - 50	5
CBLOF	#clusters	5 - 50	5
HBOS	#bins	5 - 50	5
KNN	#neighbors	5 - 50	5
LOF	#neighbors	5 - 50	5
MCD	support fraction	0.1 - 0.9	0.2
OCSVM	ν	0.1 - 0.9	0.2
PCA	#components	1 - 11[1]	1

For AKNN and KNN, the parameter chosen is the number of neighbors used during the clustering phase. The interval is between 5 and 50 since a value in that range is the typical choice. The step size selected is 5, i.e., the interval between each selected experimental value. For evaluating ABOD, we use the FastABOD implementation with the same parameters and intervals used for the previous algorithms. We use the same parameters and intervals also for LOF. In this case, we fix the parameter c to 0.1. The parameter chosen for CBLOF is the number of clusters formed during the execution of the clustering step. Additionally, we use k-means for clustering, and we fix the parameters α and β to 0.9 and 5, respectively, which means that the large clusters will contain 90% of the data and that the large clusters are a least five times bigger than the small clusters. For HBOS, the parameter chosen is the number of bins composing the histograms. Also in these cases, the interval chosen is between 5 and 50, and the step size selected is 5. In the case of MCD, the parameter chosen is the support fraction, i.e., the proportion of points to include in the MCD estimation support. The parameter values are chosen in the interval between 0.1 and 0.9. The step size is 0.2. For OCSVM, the parameter ν is both an upper bound on the

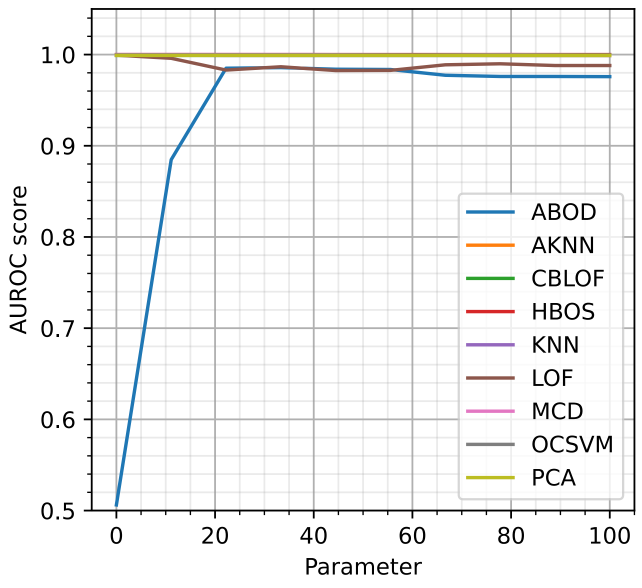
[1]In the data set *shuttle*, the interval is 1 - 8 due to the lower number of dimensions available.



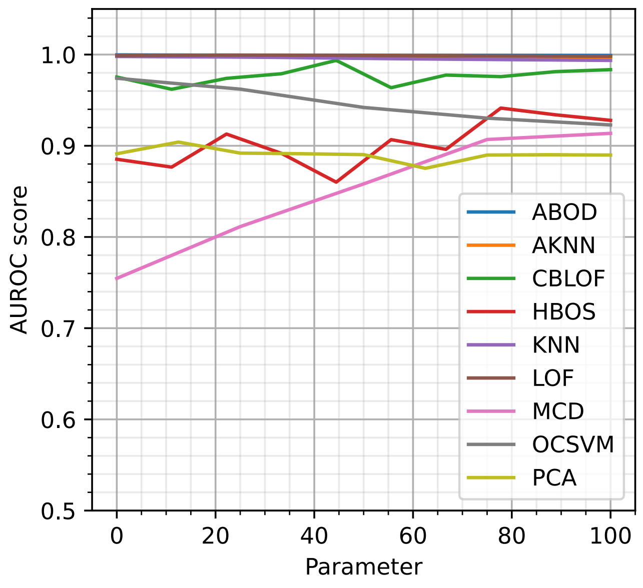
(a) xfel-dcm



(b) xfel-door



(c) xfel-pcie



(d) shuttle

Fig. 1: AUROC scores for the data sets at different values of the selected parameter

fraction of outliers in the training set and a lower bound on the fraction of examples used as support vectors. The values for the parameter are chosen in the interval between 0.1 and 0.9, and the step size is 0.2 also in this case. Finally, the parameter used for PCA is the number of components selected from the training phase. In this case, all cases between 1 and the number of dimensions of the data sets are considered.

We use the Minkowski distance for all algorithms using a calculation of the distance between points.

C. Results

The experimental results were obtained by running each algorithm once for each value of the parameter. We show the results for all data sets in Fig. 1. We plot the value of the AUROC score for each value of the parameter. Each plot is then rescaled in the X-axis to fit in the interval 0–100. For example, for KNN, the value 5 of the parameter is represented as 0, and the value 50 is represented as 100; for MCD, the value 0.1 of the parameter is represented as 0, and the value 0.9 is represented as 100. This is done to show comparably

TABLE III: Runtime results for each value of the parameter

Algorithm	Phase	Runtimes [s]										
ABOD	Training	61.778	106.874	124.179	133.455	181.943	239.390	293.399	364.965	438.775	516.536	
	Inference	298.511	349.968	425.240	536.542	574.979	743.492	895.438	1010.831	1210.836	1448.307	
AKNN	Training	83.041	123.916	126.315	131.549	152.409	138.919	146.942	143.617	145.615	133.765	
	Inference	424.036	410.104	424.954	418.653	423.985	442.249	438.314	451.077	456.117	457.288	
CBLOF	Training	1.848	2.523	2.782	3.368	4.331	4.758	5.485	6.334	6.885	7.467	
	Inference	0.012	0.178	0.191	0.197	0.214	0.212	0.215	0.215	0.219	0.246	
HBOS	Training	1.975	0.110	0.098	0.102	0.096	0.094	0.091	0.093	0.100	0.115	
	Inference	0.223	0.215	0.213	0.216	0.216	0.220	0.226	0.226	0.226	0.229	
KNN	Training	88.032	125.650	127.860	130.078	131.072	132.761	134.993	134.728	135.493	137.127	
	Inference	418.207	448.857	457.032	464.615	463.075	465.926	472.526	468.538	485.157	498.525	
LOF	Training	92.161	134.313	136.824	145.971	145.057	139.354	140.069	136.827	142.165	143.569	
	Inference	424.946	436.484	432.711	457.123	449.688	452.324	453.971	453.710	493.484	450.258	
MCD	Training	23.265	24.797	27.282	27.762	26.714						
	Inference	0.395	0.379	0.379	0.391	0.381						
OCSVM	Training	45.013	120.807	174.803	211.216	225.742						
	Inference	31.136	102.501	167.109	233.242	296.449						
PCA	Training	0.166	0.175	0.182	0.188	0.196	0.197	0.189	0.188	0.230	0.207	0.217
	Inference	0.051	0.058	0.062	0.067	0.072	0.077	0.089	0.087	0.120	0.128	0.132

the trend created by the variation of the value, regardless of its meaning.

In Fig. 1a, we can see that for the *xfel-dcm* data set, all algorithms but ABOD and MCD have a score of 0.99 or above for all values of the parameter. ABOD has a lower score only when using 5 as the parameter for the number of neighbors to consider. In MCD's case, the AUROC score stays lower but reaches a maximum value of 0.99 for a value of the support fraction of 0.9. In Fig. 1b, we see that the data set *xfel-door* is more challenging for all the algorithms. In this case, all algorithms perform much worse than for the other data sets. We also notice that the AUROC score of HBOS has large swings depending on the value of the parameter. In Fig. 1c, we can see that for the *xfel-pcie* data set, all algorithms but ABOD perform very well for all values of the parameters. In ABOD's case, selecting only 5 neighbors, the AUROC score is 0.50, i.e., a random guess. In Fig. 1d, we can see that most of the algorithms tend to perform the same or better, using a higher value of the parameter. The only exception to this is OCSVM, for which the AUROC score decreases linearly from 0.97 to 0.92. Also in this case, we notice the swinging behavior of HBOS, although with lower variability than the *xfel-door* case.

From Fig. 1, we deduce that the effect of the parameter is not always noticeable in the AUROC score. Another general takeaway point is that choosing a higher value of the parameter returns better results for the interval that we considered. Especially when using ABOD, a value of the parameter that is too low results in a decreased AUROC score. However, the parameter sometimes also influences the runtime of some algorithms. Table III shows the average runtimes for both the training and the inference phases at each value of the parameter over all data sets. Here, we can see that the runtimes of ABOD, CBLOF, OCSVM depend on the parameter. We also notice that using only 5 neighbors with HBOS, the training phase runtime increases by almost 20 times. This behavior is consistent across all considered data sets.

Table IV reports the mean value and the standard deviation over all the runs for each data set. The last two columns give a summative result for all data sets: in the first column, we report the average of the scores reported in the previous columns for all the data sets; in the second column, we report the average variance's square root. The last row gives a summative result for all algorithms instead. Also in this case, the first column contains the average of the scores reported in the previous rows for all the data sets, and the second column contains the average variance's square root.

The table shows that almost all algorithms have a very good AUROC score for the data set *xfel-dcm*. Only MCD and, to a lower extent, ABOD have a worse score. In the data set *xfel-door*, all algorithms have a lower score. The algorithms PCA and OCSVM perform only slightly better than a random guess. In this case, only HBOS, and only for some values of the parameter, achieves a maximum AUROC score of 0.88. The table shows very well also the minor differences in the *xfel-pcie* data set. ABOD is penalized in this table by the low score obtained with a lower number of neighbors. For this algorithm, the mean is 0.9804, and the standard deviation is 0.0042, removing the first two values obtained (0.50 and 0.88, respectively). For the *shuttle* data set, Table IV shows that ABOD has the best overall score. In this case, the AUROC score remains above 0.99 for all values of the parameter. From the last two columns, we can see that LOF has the best overall score, and MCD has the worst overall score.

Table V and Table VI report the runtime values for the training and testing phases, respectively. From the two tables, we can immediately understand that the number of samples considered for each phase is the main factor for the differences.

TABLE IV: AUROC results

Algorithm	*xfel-dcm*		*xfel-door*		*xfel-pcie*		*shuttle*		*Summary*	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ABOD	0.9786	0.0639	0.7158	0.0545	0.9234	0.1420	0.9993	0.0002	0.9043	0.0825
AKNN	1.0000	0.0000	0.6604	0.0052	0.9992	0.0001	0.9976	0.0004	0.9143	0.0026
CBLOF	1.0000	0.0000	0.6107	0.0080	0.9991	0.0000	0.9766	0.0087	0.8966	0.0059
HBOS	0.9999	0.0001	0.7568	0.1119	1.0000	0.0000	0.9033	0.0249	0.9150	0.0573
KNN	1.0000	0.0000	0.6497	0.0076	0.9991	0.0001	0.9958	0.0015	0.9112	0.0039
LOF	1.0000	0.0001	0.7248	0.0471	0.9885	0.0053	0.9988	0.0004	0.9280	0.0237
MCD	0.9143	0.0522	0.6577	0.0277	0.9997	0.0004	0.8489	0.0599	0.8552	0.0421
OCSVM	0.9999	0.0000	0.5776	0.0154	0.9996	0.0000	0.9463	0.0192	0.8809	0.0123
PCA	1.0000	0.0000	0.5490	0.0040	0.9991	0.0000	0.8904	0.0068	0.8596	0.0040
Summary	0.9881	0.0275	0.6558	0.0458	0.9898	0.0474	0.9508	0.0229	0.8961	0.0375

In particular, in Table VI, we can see that, since the number of samples of the testing set of the *xfel-pcie* data set is 4 times larger than the *shuttle* training set, the runtime numbers are 4 times larger as well. Another difference that affects the runtimes is the number of dimensions. Considering this time the training time for the data sets *xfel-door* and *xfel-pcie*, we have very similar ratios in the number of dimensions of the two training sets and average training times.

The algorithm summaries show that the most noticeable result is the poor runtime of ABOD with respect to the other algorithms. From the same columns, we can also see that the best performing algorithms in the training phase are HBOS and PCA. Since, as seen in Table III, HBOS requires a much larger time for training using 5 bins. We calculated that the mean value is 0.1 s removing those values. The best runtimes for the testing phase are obtained using CBLOF, HBOS, MCD, and PCA.

VII. Conclusion

In this work, we performed a comprehensive evaluation of novelty detection methods and studied their performance and scalability when applied to the data produced by distributed high-integrity systems. We tested a wide variety of algorithms using 3 data sets that we produced and characterized, coming from the European X-Ray Free-Electron Laser, and a standard data set acquired from the public UCI Machine Learning Repository [39]. The data sets selected show the distinct characteristics of each algorithm. Besides that, we also gave a unified description of the algorithms. This has a twofold use: it gives a precise representation of the algorithms to the researchers of the novelty detection community. It also gives the users a better understanding of the algorithms and the performances.

From the experimental results, we conclude that for semi-supervised anomaly detection, simple algorithms can perform satisfactorily. This makes it feasible to use such algorithms for real-time or near real-time monitoring of high-integrity digital systems. Also, a careful selection of the signals is necessary since the prediction can be more or less accurate depending on the anomaly observed.

References

[1] L. Portnoy, “Intrusion detection with unlabeled data using clustering,” Bachelor’s Thesis, Columbia University, 2000.

[2] D.-Y. Yeung and Y. Ding, “Host-based intrusion detection using dynamic and static behavioral models,” *Pattern Recognition*, vol. 36, no. 1, pp. 229–243, 2003.

[3] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers & Security*, vol. 28, no. 1, pp. 18–28, 2009.

[4] R. J. Bolton and D. J. Hand, “Statistical fraud detection: A review,” *Statistical science*, pp. 235–249, 2002.

[5] C. Phua, V. C. S. Lee, K. S. Miles, and R. W. Gayler, “A comprehensive survey of data mining-based fraud detection research,” *CoRR*, vol. abs/1009.6119, 2010.

[6] J. Lin, E. J. Keogh, A. W.-C. Fu, and H. V. Herle, “Approximations to magic: finding unusual medical time series,” in *Symposium on Computer-Based Medical Systems*, 2005, pp. 329–334.

[7] D. Dasgupta and S. Forrest, “Novelty detection in time series data using ideas from immunology,” in *International conference on intelligent systems*, 1996, pp. 82–87.

[8] D. Dasgupta and F. Nino, “A comparison of negative and positive selection algorithms in novel pattern detection,” in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, 2000, pp. 125–130.

[9] S. P. King, D. M. King, K. Astley, L. Tarassenko, P. Hayton, and S. Utete, “The use of novelty detection techniques for monitoring high-integrity plant,” in *International Conference on Control Applications*, vol. 1, 2002, pp. 221–226.

[10] A. Basharat, A. Gritai, and M. Shah, “Learning object motion patterns for anomaly detection and improved object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[11] J. Gebhardt, M. Goldstein, F. Shafait, and A. Dengel, “Document authentication using printing technique features and unsupervised anomaly detection,” in *International Conference on Document Analysis and Recognition*, 2013, pp. 479–483.

[12] V. Hodge and J. Austin, “A survey of outlier detection methodologies,” *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85 – 126, 2004.

[13] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, 2009.

[14] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong, “A meta-analysis of the anomaly detection problem,” *arXiv e-prints*, pp. 1–35, 2016.

[15] M. Goldstein and S. Uchida, “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” *PLoS One*, vol. 11, no. 4, 2016.

[16] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, “An experimental evaluation of novelty detection methods,” *Neurocomputing*, vol. 135, pp. 313–327, 2014.

[17] R. Domingues, P. Michiardi, J. Barlet, and M. Filippone, “A comparative evaluation of novelty detection algorithms for discrete sequences,” *Artificial Intelligence Review*, vol. 53, no. 5, pp. 3787–3812, 2020.

TABLE V: Training time results [s]

Algorithm	*xfel-dcm*		*xfel-door*		*xfel-pcie*		*shuttle*		*Summary*	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ABOD	181.470	139.418	468.874	182.835	132.797	97.622	201.377	168.923	246.129	150.774
AKNN	238.707	41.041	257.993	36.334	26.554	1.173	7.181	1.914	132.609	27.430
CBLOF	8.062	2.689	8.317	2.767	1.009	0.415	2.192	0.935	4.895	1.996
HBOS	0.362	0.594	0.380	0.551	0.203	0.540	0.204	0.565	0.287	0.563
KNN	202.689	28.534	274.871	24.593	26.605	1.597	6.952	1.588	127.780	18.869
LOF	235.795	34.984	273.423	25.760	26.265	1.184	7.041	1.566	135.631	21.745
MCD	38.841	2.460	52.768	2.499	2.834	0.113	9.412	2.949	25.964	2.292
OCSVM	218.806	93.180	309.065	129.763	53.946	23.788	40.248	17.817	155.516	81.247
PCA	0.288	0.068	0.336	0.090	0.092	0.026	0.056	0.015	0.193	0.058
Summary	125.002	38.108	182.892	45.021	30.034	14.051	30.518	21.808	92.112	58.618

TABLE VI: Inference time results [s]

Algorithm	*xfel-dcm*		*xfel-door*		*xfel-pcie*		*shuttle*		*Summary*	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ABOD	844.195	397.339	1868.200	855.200	203.637	144.922	81.626	66.400	749.414	19.130
AKNN	563.947	35.635	1141.651	88.584	28.124	2.503	4.990	0.703	434.678	5.644
CBLOF	0.268	0.027	0.543	0.045	0.022	0.003	0.006	0.001	0.210	0.137
HBOS	0.282	0.026	0.578	0.026	0.022	0.002	0.003	0.001	0.221	0.116
KNN	603.362	41.275	1221.880	39.791	26.970	2.187	4.772	0.685	464.246	4.581
LOF	587.019	33.565	1190.931	50.398	20.865	2.299	3.064	0.673	450.470	4.662
MCD	0.379	0.015	1.149	0.019	0.010	0.000	0.002	0.000	0.385	0.094
OCSVM	186.148	99.540	447.118	257.089	26.195	14.963	4.889	2.790	166.087	9.674
PCA	0.130	0.112	0.272	0.257	0.015	0.004	0.003	0.001	0.105	0.306
Summary	309.525	67.504	652.480	143.490	33.984	18.543	11.039	7.917	251.757	7.705

[18] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014.

[19] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[20] M. Markou and S. Singh, "Novelty detection: a review—part 1: statistical approaches," *Signal Processing*, vol. 83, no. 12, pp. 2481–2497, 2003.

[21] M. Markou and S. Singh, "Novelty detection: a review—part 2:: neural network based approaches," *Signal Processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[22] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *Principles of Data Mining and Knowledge Discovery*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 15–27.

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *ACM SIGMOD International Conference on Management of Data*, 2000, p. 93–104.

[24] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[26] Z. He, X. Xu, and S. Deng, "Squeezer: An efficient algorithm for clustering categorical data," *Journal of Computer Science and Technology*, vol. 17, no. 5, pp. 611–624, Sep. 2002.

[27] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm," in *KI-2012: Poster and Demo Track*, 2012, pp. 59–63.

[28] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, p. 444–452.

[29] P. J. Rousseeuw and K. van Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[30] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *International Conference on Neural Information Processing Systems*, 1999, p. 582–588.

[31] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear svm," *Journal of Machine Learning Research*, vol. 11, no. 48, pp. 1471–1490, 2010.

[32] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[33] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *IEEE Foundations and New Directions of Data Mining Workshop*, 2003, pp. 172–179.

[34] D. J. Hand and R. J. Till, "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems," *Machine Learning*, vol. 45, no. 2, pp. 171–186, Nov. 2001.

[35] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[36] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data-recommendations for the use of performance metrics," in *Humaine Association Conference on Affective Computing and Intelligent Interaction*, 2013, pp. 245–251.

[37] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognition*, vol. 91, pp. 216–231, 2019.

[38] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, no. 96, pp. 1–7, 2019.

[39] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[40] D. Q. Goldin and P. C. Kanellakis, "On similarity queries for time-series data: Constraint specification and implementation," in *International Conference on Principles and Practice of Constraint Programming*, 1995, pp. 137–153.