IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

An interactive threshold-setting procedure for improved multivariate anomaly detection in time series

ADAM LUNDSTRÖM[1,2], MATTIAS O'NILS[1] AND FAISAL Z. QURESHI[1,3], (Senior Member, IEEE)

[1]Department of Electronics Design, Mid Sweden University, 85170 Sundsvall, Sweden
[2]SCA, 85188, Sundsvall, Sweden (e-mail: adam.lundstrom2@sca.com)
[3]Faculty of Science, University of Ontario Institute of Technology, Oshawa, ON L1G 0C5, Canada (e-mail: faisal.qureshi@ontariotechu.net)

Corresponding author: Adam Lundström (e-mail: adam.lundstrom@miun.se).

This work was supported in part by The Knowledge Foundation (kks.se) within the Industrial graduate school Smart Industry Sweden.

ABSTRACT Anomaly detection in multivariate time series is valuable for many applications. In this context, unsupervised and semi-supervised deep learning methods that estimate how normal a new observation is have shown promising results on benchmark datasets. These methods are dependent on a threshold that determines which points should be regarded as anomalous and not be anomalous. However, finding the optimal threshold is not easy since no information about the ground truth is known in advance, which implies that there are limitations to automatic threshold-setting methods available today. An alternative is to utilize the expertise of users that can interact in a threshold-setting procedure, but for this to be practically feasible, the method needs to be both accurate and efficient in relation to the state-of-the-art automatic methods. Therefore, this study develops an interactive threshold-setting schema and examines to what extent it can outperform the current state-of-the-art automatic threshold-setting methods. The result of the study strongly indicates that the suggested method with little effort can provide higher accuracy than the automatic threshold-setting methods on a general basis.

INDEX TERMS Anomaly detection, Anomaly scoring, Deep learning, Multivariate time series (MVTS).

I. INTRODUCTION

ANOMALY detection in time series is of great importance for numerous applications such as bearing fault detection [1], [2], fault detection in satellites [3], fault detection in ships [4] but also for other applications in smart manufacturing, energy management and computer systems [5]. Significant advances in machine learning (ML) and most notably deep ML have shown great performance on benchmark datasets [5], [6], [7]. The lack of labels is an important reason why unsupervised and semi-supervised ML methods have become prominent for anomaly detection [5], [8]. In this context, a distinction can be made between methods that work under weekly supervised end-to-end scenarios and those that are purely unsupervised or only learns from the normal data [8]. The goal of the latter is to overcome the challenge of obtaining historical examples of faults and to create robust solutions that output an anomaly score that represent an approximation of how normal an observation is [5]. However, while these methods have great potential, their accuracy is dependent on thresholds that determine whether or not a new observation is anomalous. In addition, a reasonable assumption is that this issue can become especially apparent in multivariate data where the complexity is high. Currently, a common method to measure the potential for a new anomaly detection method of this type is to use the available labels from the test or validation dataset [5], [6], [9], [10]. Yet, this is not possible in many real scenarios where labels are not available and the substantial time to label all the data manually makes it an unrealistic task. Instead, several different attempts have been made to automatically estimate the thresholds. For example, two automatic threshold-setting methods: Nonparametric Dynamic Thresholding (NDT) [11] and the Peaks-Over-Threshold (POT) [12] have shown great results. A second option is to let the users actively select the most appropriate thresholds which would lead to more effort yet potentially could reduce the errors made by more auto-

matic methods. Therefore, to evaluate to what extent an interactive threshold-setting schema can outperform the state-of-the-art automatic threshold-setting methods this study develops an effective schema, called User Validated Thresholds (UVT), and compares it to the recently applied threshold-setting methods in multivariate anomaly detection studies top-k [13], NDT [11] and POT [12] using a realistic setup where parameters are kept consistent through all datasets. We will evaluate the following research question:

- To what extent can an interactive threshold schema for machine learning-based anomaly detection in multivariate time series outperforms state-of-the-art automated threshold-setting methods?

The main contribution of this study is that it provides a human-in-the-loop based threshold-setting method that targets machine learning methods for multivariate anomaly detection which outperforms state-of-the-art automated threshold-setting methods and only requires few interactions from the user. It can be used by all anomaly detection methods that generate an anomaly score to describe how normal a new observation is. The purpose of our method, which can be seen in Figure 1, is to generate thresholds from historic data for anomaly detection methods so that they can be applied online. The offline data used for UVT is unlabeled and is assumed to contain both normal and abnormal data. The method is initialized when a model has been built from normal training data. Then, historic data is used, together with the user, to adjust the thresholds for the anomaly prediction without needing to retrain the underlying model.

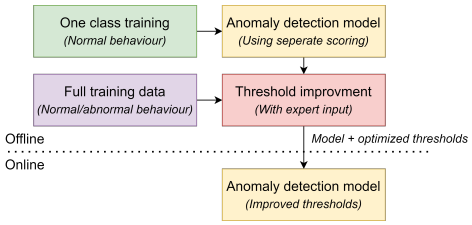

FIGURE 1. Suggestion for how the proposed method should be applied. A model is created based on normal training data and is then improved using expert input and separate scoring from the anomaly detection model.

II. EXTENDED BACKGROUND

The threshold-setting procedure is a challenge for all anomaly detection methods that use an anomaly score to evaluate if an entry is anomalous or not and where the ground truth is unknown. There are several different methods used in previous research but we will focus on the methods that do not need predefined labeled anomalies.

Firstly, one approach, used in [14], is to apply the maximum, or a certain percentile, of the anomaly score from the normal training data. This method is simple yet does not perform well when outliers are present in the training data. Another method, top-k, for example, applied in [13] and [15], is to select a threshold which leads to the expected number of anomalies in the test data. The main issue with this method, however, is that it makes assumptions about the number of anomalies in the test data, which is likely unknown in advance. In addition, it is probable that some methods give higher accuracy when this limit is set to below or above the actual number of anomalies making it arguably sub-optimal as a general-purpose method. Furthermore, tail-p built for streaming scenarios is another threshold-setting method based on the Gaussian probability which was used in [16] and [13]. It uses the Gaussian tail of a streaming window to determine whether or not a certain anomalous point a_t is anomalous. Similar to this method, NDT was suggested in 2018 by K. Hundman *et al.* [11] and has for example been used by [17]. The advantage of NDT and tail-p is that they do not make any assumptions about the distribution of the test data or the number of anomalies in it. This makes it suitable for real applications. Another solution that has been suggested is the POT method by [12] which has been applied by for example [7]. It is based on Extreme Value Theory (EVT) and is constructed with a General Pareto Distribution. Similarly to NDT, it makes relatively few assumptions about the underlying data and can be applied as a general threshold-setting method. Yet the issue, from a practical standpoint, is that both [7] and [12] optimize the parameters for the method with the help of labels, meaning that the performance of the method is not shown for scenarios where the ground truth is unknown.

While active learning for anomaly detection is an area studied before, such as [18] and [19], user active schemes for threshold-setting is something, that to the best of our knowledge, has not been researched to any great extent before. The difference is that the active learning procedure changes the underlying model to better separate anomalous from non-anomalous points while an interactive threshold-setting schema only changes aspects of an already trained model. In contrast to active learning, the second type of method can be directly applied to any ML-based anomaly detection model without having to retrain the underlying model.

It is clear that the threshold-setting method is of great importance for multivariate time series anomaly detection methods, but it is still unclear which of the presented solutions works well in a variety of settings. Therefore, this study compares a validation-based threshold schema with state-of-the-art automatic threshold-setting methods to evaluate which one is better.

III. METHOD

A. PROBLEM DEFINITION

We are provided a multivariate time series data $\boldsymbol{x} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$, where $\boldsymbol{x}_t \in \mathbb{R}^C$, C is the number of channels and $t \in [1, N]$. We are also given the corresponding anomaly scores $\boldsymbol{a} = \{\boldsymbol{a}_1, ..., \boldsymbol{a}_N\}$ that are computed by a pre-trained anomaly detection method $\mathcal{M} : \boldsymbol{x}_t = \mathcal{M}(\boldsymbol{x}_{tl+1}, , \boldsymbol{x}_t)$ where the model uses the last l readings to assign the anomaly score. We cover two cases: (1) separated anomaly scores that are computed on a per-channel basis where $\boldsymbol{a}_t \in \mathbb{R}^C$ and

(2) aggregated anomaly scores where $\boldsymbol{a}_t \in \mathbb{R}$. Our goal is to apply a function $f(\mathcal{M}, \boldsymbol{a})$ that with the help of a user, improves the overall accuracy with respect to the ground truth $\boldsymbol{y} = \{y_1, ..., y_N\}$ by adjusting the channel-wise thresholds that determine whether or not a point is anomalous.

B. OVERVIEW

The suggested method, which is displayed in Figure 2, is initiated with the trained model creating a separate anomaly score for each of the channels using an input window of size l. Using these, the initial thresholds are set using NDT, which specifies the initial anomalous points. These points are clustered into connected segments so that the user can classify points in a context and so that a single interaction leads to a larger number of points being labelled. A prioritization procedure determines which segments should be validated by the user and an inclusion mechanism skips segments based on resemblance to previously validated segments. More segments are added incrementally by adjusting the thresholds and the process continues until certain conditions have been met. The process ends with the thresholds being set based on the labelled data.

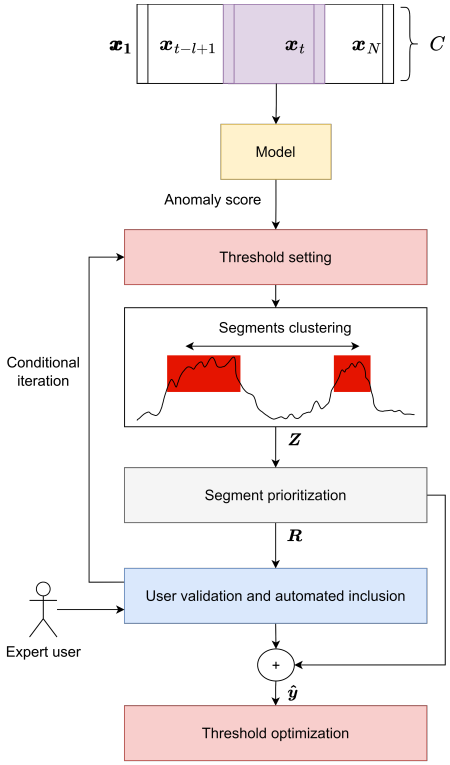

FIGURE 2. A description of the suggested method UVT. A model is created using normal training data and the process for UVT is initiated by using this model to create initial thresholds for the separate anomaly score on the full training set. Then segments are clustered, prioritized and looped through an interactive validation procedure until a certain condition has been met.

C. THRESHOLD INITIATION

At the start of the process, we aim to, based on the anomaly score $\boldsymbol{a}$, create a baseline with points that with a high probability are within the anomalous segments. In this regard, NDT is suitable since it is fairly conservative in its selection of anomalous points. NDT utilize the change in mean and standard deviation when the anomaly score below a threshold is changing. In this method, the threshold, τ, is determined through $\boldsymbol{z}$ which is the changeable parameter containing values between a minimum and a maximum value with a certain step size. As NDT was designed for online scenarios and we will use it on offline data, we modify the method to increase the performance and define $\boldsymbol{\epsilon}$ as:

$$\boldsymbol{\epsilon} = \mu_{\boldsymbol{a}} + \boldsymbol{z}\sigma_{\boldsymbol{a}} \tag{1}$$

Where $\boldsymbol{a}$ is the anomaly score. The threshold is determined by:

$$\tau = \text{argmax}(\boldsymbol{\epsilon}) = \frac{\frac{\Delta\mu_{\boldsymbol{a}}}{\mu_{\boldsymbol{a}} + \frac{\Delta\sigma_{\boldsymbol{a}}}{\sigma_{\boldsymbol{a}}}}}{n_{\boldsymbol{a}}} \tag{2}$$

Where:

$$\begin{aligned} \Delta\mu_{\boldsymbol{a}} &= \mu_{\boldsymbol{a}} - \mu_{\boldsymbol{a}_b} \text{ where } \boldsymbol{a}_b = \{\boldsymbol{X} | X_i \in \boldsymbol{a}, X_i < \epsilon\} \\ \Delta\sigma_{\boldsymbol{a}} &= \sigma_{\boldsymbol{a}} - \sigma_{\boldsymbol{a}_b} \text{ where } \boldsymbol{a}_b = \{\boldsymbol{X} | X_i \in \boldsymbol{a}, X_i < \epsilon\} \\ \boldsymbol{a}_a &= \{\boldsymbol{X} | X_i \in \boldsymbol{a}, X_i > \epsilon\} \\ n_{\boldsymbol{a}_a} &= \text{ Number of } \boldsymbol{a}_a \end{aligned} \tag{3}$$

This means that we are after the threshold that maximises the differences in mean and standard deviation when removing anomalous points using $\boldsymbol{\epsilon}$. The assumption is that this is effective in separating anomalous from non-anomalous points. We use NDT to create the thresholds $\boldsymbol{\tau}$ with size C. In contrast to most work in multivariate anomaly detection we, as suggested in [20], utilize separate scoring instead of aggregated scoring to not overlook information in the aggregation process, which would risk missing anomalous segments. We then construct an array $\boldsymbol{p}$, which describes the anomaly prediction where non-anomalous points are denoted as 0 and anomalous points are denoted as 1. A point is anomalous if any score from a single channel is above its respective threshold.

D. CLUSTERING ANOMALIES INTO SEGMENTS

The next step of the suggested method is to cluster the anomalous data predictions $\boldsymbol{p}$ into continuous sequences of anomalous points named segments. The purpose of this is to reduce the number of validation steps a user has to take. In addition, from a user perspective, it is reasonable to assume that a user would prefer to evaluate points together in a time domain context. We achieve the clustering procedure by linking segments that share some common attributes in accordance with given conditions. We start by creating initial segments defined as $\boldsymbol{S} = \{\boldsymbol{S}_1, ..., \boldsymbol{S}_M\}$ that are constructed by moving from left to right in the time domain and grouping anomalous points that are directly connected. For each segment, we add a subset to $\boldsymbol{S}$ that contains the anomalous indices. This means that a certain segment $\boldsymbol{S}_i$, contains a set of indices, each of which is equal to some time step t that represents an anomalous point, meaning that $\boldsymbol{x}_t \in \boldsymbol{x}$ and that there exists a channel c so that $a_t^c > \tau^c$.

Furthermore, we formulate a function that returns the first and last indice from any given subset in $\boldsymbol{S}$ so that for a segment i, $f_{ind}(\boldsymbol{S}_i, e)$ gives the last indice and $f_{ind}(\boldsymbol{S}_i, s)$ gives the first indice. Then, we start the segment clustering procedure where we iterate the segments from left to right in the time domain and merge temporally adjacent segments i and j where $i \neq j$ and $j > i$ by considering four different attributes $\boldsymbol{A}$: the time proximity between the current and next anomalous segment $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ denoted as A_1, the difference between the maximum amplitude of the data for each channel j and with regards to segments $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ called $\boldsymbol{A}_2$, the difference between the maximum score $\boldsymbol{a}^c$ for each channel c considering the segments $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ named $\boldsymbol{A}_3$ and which channels have a score $\boldsymbol{a}^c$ above each respective threshold τ^c in both $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ denoted as $\boldsymbol{A}_4$:

$$\begin{aligned} A_1 &= f_{ind}(S_j, s) - f_{ind}(S_i, e) \\ A_2^c &= |\max(\boldsymbol{x}^c_{\boldsymbol{S}_i}) - \max(\boldsymbol{x}^c_{\boldsymbol{S}_j})| \\ A_3^c &= |\max(\boldsymbol{a}^c_{\boldsymbol{S}_i}) - \max(\boldsymbol{a}^c_{\boldsymbol{S}_j})| \\ A_4^c &= \begin{cases} 1 & \text{if } \max(\boldsymbol{a}^c_{\boldsymbol{S}_i}) > \tau^c \text{ and } \max(\boldsymbol{a}^c_{\boldsymbol{S}_j}) > \tau^c \\ 0 & \text{else} \end{cases} \end{aligned} \tag{4}$$

Then we define $\boldsymbol{B}$ which holds five different conditions based on the attributes $\boldsymbol{A}$ and the logical output $\boldsymbol{O}$ for each of the outcomes when comparing the conditions in relation to the attributes:

$$\begin{aligned} O_1 &= A_1 \leq B_1 \\ O_2 &= A_1 \leq B_2 \\ O_3 &= \forall_c A_2^c \leq B_3^c, \text{ where } \max(\boldsymbol{a}^c_{\boldsymbol{S}_i}) > \tau^c \\ O_4 &= \forall_c A_3^c \leq B_4^c, \text{ where } \max(\boldsymbol{a}^c_{\boldsymbol{S}_i}) > \tau^c \\ O_5 &= \sum_1^C A_4^c \geq B_5 \end{aligned} \tag{5}$$

Where B_1 is the maximum distance between the two segments for them to be connected. A larger distance than B_1 assumes that the segments are not related regardless of the other attributes. B_2 is the short-time condition that connects two segments assuming that the segments are close enough that they are related. B_3 and B_4 are simply thresholds for the differences between the maximum amplitude of the data and anomaly score, respectively. Lastly, B_5 regulates the number of anomalous channels that need to correspond in the two segments. We formulate the possible connection of $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ as:

$$\boldsymbol{S}_i = \begin{cases} \boldsymbol{S}_i \cup \boldsymbol{S}_j & \text{if } (O_1 \wedge \cup_{i=3}^5 \boldsymbol{O}_i) \vee O_5 \\ \boldsymbol{S}_i & \text{else} \end{cases} \tag{6}$$

We then continue this described process until no temporally adjacent segments i and j are left to evaluate.

E. COLLECTION OF CHANNEL-WISE SEGMENTS

When all segments have been created, we collect the segments for each channel. For each channel c, we start by checking if a point within each segment is anomalous considering the anomaly score $\boldsymbol{a}^c$ and the respective threshold τ^c. For all segments including a point for a particular channel above its threshold, we construct a dedicated set of segments for each channel donated as $\boldsymbol{Z} = \{\boldsymbol{Z}^1, ..., \boldsymbol{Z}^C\}$, where $\boldsymbol{Z}^c = \{\boldsymbol{Z}^c_1, ..., \boldsymbol{Z}^c_{K^c}\}$ and $\boldsymbol{Z}^c_i \in \boldsymbol{S}$. This means that several channels may have the same segment in their respective segment collection $\boldsymbol{Z}^c$ and that for a certain channel c and a segment i $\max(\boldsymbol{a}_{\boldsymbol{Z}^c_i}) > \tau^c$.

F. SEGMENT PRIORITIZATION

As the importance of anomalous segments might vary, we need a way to prioritize the segments shown to the user. We therefore define $\boldsymbol{u} = \{\boldsymbol{u}^1, ..., \boldsymbol{u}^C\}$, where $\boldsymbol{u}^c = \{\boldsymbol{u}^c_1, ..., \boldsymbol{u}^c_{K^c}\}$ and u^c_i is the prioritization score for $\boldsymbol{Z}^c_i$:

$$u^c_i = \frac{\max\left(\boldsymbol{a}^c_{\boldsymbol{Z}^c_i}\right)}{\sigma_{\boldsymbol{b}^c}}, \tag{7}$$

Here, σ_{b^c} is the standard deviation of the interquartile range of the anomaly score of the normal training data for a channel c. We then define $\boldsymbol{R} = \{R_1, ..., R_F\}$ which holds the segments that are shown to the user and contains one segment for each channel that has the highest channel-wise prioritization score and has not yet been evaluated where F is the number of channels that has one or more segments that have not been evaluated. To know which segments are yet to be evaluated we define $\boldsymbol{d} = \{\boldsymbol{d}^1, ..., \boldsymbol{d}^C\}$, where $\boldsymbol{d}^c = \{d^c_1, ..., d^c_{K^c}\}$ so that a single segment $\boldsymbol{Z}^c_i$ corresponds to a decision $d^c_i \in \{-1, 0, 1\}$, where 0 means that the segment has not been evaluated, -1 that it has been regarded as normal or excluded and 1 that it has been determined to be anomalous. Then, prior to adding the selected segments to $\boldsymbol{R}$, we sort them so that the segment with the highest prioritization score has the lowest index $\boldsymbol{R}_1$ and the segment with the lowest prioritization score has the highest index $\boldsymbol{R}_C$. Thus, R is defined as:

$$\boldsymbol{R} = \{\boldsymbol{X} | \boldsymbol{X}_c \in \boldsymbol{S}, d^c_{k^c} = 0, \max(\boldsymbol{u}^c_{k^c}) \geq \max(\boldsymbol{u}^{c+1}_{k^c})\} \tag{8}$$

Where k^c is the index of the selected segment for channel c. After all segments in $\boldsymbol{R}$ have been evaluated it is emptied and set with new segments before starting a new validation round. The reason for iterating all channels each round is that different types of anomalies are likely to be present in different channels. This means that only selecting the channel with the highest $\boldsymbol{u}^c$ for each round implies that we are likely to get a reduced recall, which also was shown in initial experiments.

G. VALIDATION, AUTOMATED INCLUSION AND CONDITION-BASED ITERATION

For each new segment shown to the user, they can either define it as anomalous or not anomalous, which is interpreted as 1 and -1 respectively, meaning that for a certain channel c and segment i, $d^c_i \in \{-1, 1\}$. In addition, to speed up the process, we disregard new segments that resemble or are

equal to what the user has already seen. To achieve this, for a new segment i and channel c that has a 10% or larger number of points that are the same as a segment that the user has already validated, we set $d_i^c = -1$. This means that it will not be added to $\boldsymbol{R}$ and thereby not be shown to the user. An exception is made when a new channel c with index i has 50% or larger number of points which are the same as a segment that already has been validated as anomalous by the user. In this case, we automatically set it as anomalous so that $d_i^c = 1$.

Furthermore, as some true positive anomalous points might be below the limit of the initial thresholds, we conditionally append new segments to $\boldsymbol{Z}$ when no new segment for a channel c is left to be evaluated. This is achieved by lowering the threshold τ^c iteratively until at least one new segment has been added or a minimum limit is reached which is set to the same limit as for NDT in Equation 1 using $z \in \boldsymbol{z}$ equal to 2. In this way, it is possible to find new segments that were not identified by the initial thresholds and thereby increase the recall.

When the validation procedure has terminated we simply use all evaluated segments in $\boldsymbol{d}$ to define the ground truth predictions $\hat{\boldsymbol{y}} = \{\hat{y}_1, ..., \hat{y}_N\}$. We define $\hat{\boldsymbol{y}}$ for a certain time step t as:

$$\hat{y}_t = \begin{cases} 1 & \text{if } \exists\{i, c\} : d_i^c = 1 \wedge t \in \boldsymbol{Z}_i^c \wedge a_i^c > \tau^c \\ 0 & \text{else} \end{cases} \tag{9}$$

Then, we use $\hat{\boldsymbol{y}}$ to optimize the thresholds $\boldsymbol{\tau}$:

$$\max_{\boldsymbol{\tau}} g(\boldsymbol{\tau}) = f_1(\hat{\boldsymbol{y}}, \text{prediction}(\boldsymbol{a}, \boldsymbol{\tau})) \tag{10}$$

Where f_1 is the f_1-score and prediction$(\boldsymbol{a}, \boldsymbol{\tau})$, is the anomaly prediction based on the anomaly score and thresholds.

H. STOPPING CRITERIA

We employ stopping criteria for each individual channel and for the validation process as a whole. The channel criteria defined as v_1 determines that if a user has evaluated more than one segment as not anomalous, that channel will be ignored, meaning that for all segments i for a channel c where $d_i^c = 0$, we set $d_i^c = -1$. This is performed because it is assumed that segments with a maximum anomaly score below the current non-anomalous segment, will not be anomalous. The reason why one segment is allowed to be rejected is that the separation between anomalous segments based on the anomaly score is not always linear, meaning that several true positive segments might have a lower anomaly score than a true negative segment. The overall stopping criteria v_2 is for the number of user interactions meaning that the validation procedure can be stopped before v_1 has been reached.

IV. EXPERIMENT

A. DATASETS

In the evaluation, five different types of benchmark datasets for multivariate anomaly detection are used that have been widely applied in previous studies [5]-[7], [10], [13], [21], [22]. These are Water Distribution (WADI) [23], Secure Water Treatment (SWaT) [23], Server Machine dataset (SMD) [12], Mars Science Laboratory (MSL) and Soil Moisture Active Passive (SMAP) [11].

1) WADI

WADI is a dataset generated from a testbed with 123 variables describing a water distribution system that has run for 16 days; 14 days under normal conditions and two days with attack scenarios.

2) SWaT

SWaT is a dataset of a secure water treatment procedure generated from a testbed and consists of 51 different variables. It includes 11 days of operation with seven days where conditions are normal and four days under attack.

3) SMD

SMD is a server machine dataset with 38 variables collected from an internet company. The data is collected from 28 different machines meaning that the training and testing are conducted on each machine separately.

4) MSL

MSL is a real-world dataset by NASA of a spacecraft mission collected from a rover. It consists of 27 unique entities each containing 19 different variables. The first variable in each entity is the telemetry value and the rest is one-hot encoded data about communication

5) SMAP

SMAP is another real-world NASA spacecraft mission with 55 unique entities and 28 variables collected from a satellite. As with MSL, the first value in each dataset is the telemetry value and the rest is on-hot encoded communication.

B. EVALUATION METRIC

When evaluating the performance of different methods, a metric is needed that both shows how effective each method is at capturing segments, and how precise each method is but also how much effort is needed from the user to validate the segments for UVT. Considering the user validation effort, we simply measure the number of segments that the user needs to validate. For the accuracy of the method we apply the f_{c1} scoring method by [13] which uses segment recall, R_e and point-wise precision, P_t:

$$f_{c_1} = 2 \cdot \frac{R_e \cdot P_t}{R_e \cdot P_t} \tag{11}$$

We use this method instead of the widely used point adjust method suggested by [12] which does not consider the number of segments captured by the method and gives unreasonably high accuracy when long true positive segments are present, as shown in [13].

C. SETUP

In the experiment, we used two well-known different underlying machine learning algorithms with their respective anomaly scoring function, which are GDN [21] and OmniAnomaly [12]. As these are only used to evaluate the threshold-setting methods, we will not describe the characteristics of these. Instead, detailed descriptions can be found in the original works [21] and [12]. The threshold-setting methods used in the experiment were UVT, top-k, NDT and POT. In addition, to show the effort and theoretical performance of letting the user label all anomalous points based on the initial thresholds (based on III-B), we included the method denoted as Raw Validation (RV). We kept a consistent setup throughout the experiments for each of the methods and datasets and choose parameters with high general performance instead of the optimal parameters which we assume to be unknown. This means that we used the same data preprocessing for both GDN and OmniAnomaly and the same parameters for the threshold-setting methods on all datasets in the experiment. This was done to simulate real applications where the distribution of the anomaly data is assumed to be unknown in advance. Lastly, as all datasets used contains a training set of normal data and a test set, we treated the training data as the normal data so that for example $\boldsymbol{b}$ from Equation 7 is based on the training set. All threshold-setting methods were applied and evaluated on the test set.

1) Preprocessing

We normalized SWaT and WADI to between 0 and 1 and kept the rest of the dataset as is since they are already in a span of [-1,1] or [0,1]. Because of the relatively large size of SWaT and WADI, we down-sampled these with a factor of 10 to speed up training as was done in for example [21].

2) Training parameters

Considering training parameters, we applied the default settings from the respective official repository for GDN[1] and OmniAnomaly[2].

3) Threshold-setting methods

For the application of NDT in UVT and RV we set $\boldsymbol{z}$ from Equation 1 to a minimum of 2 and maximum of 10 with a step size of 0.5. This was done for each individual channel. In addition, the maximum number of validation iterations, v_1, was set to the length of the test set divided by 1000. If this number was lower than 30, we set it to 30. These limits were chosen as they, in initial experiments, seemed to provide a reasonable trade-off between the number of validation iterations and accuracy. For the clustering conditions $\boldsymbol{B}$ described in Equation 5, B_1 was set to 500-time points, B_2 to 10, B_3^c to the standard deviation of $\boldsymbol{x}^c$ for the normal data, B_4^c to the threshold for each channel τ^c and B_5 to 1.

[1]https://github.com/d-ailin/GDN
[2]https://github.com/NetManAIOps/OmniAnomaly

For NDT, as with the original work, we set $\boldsymbol{z}$ to a maximum of 10 and a minimum of 2 with a stepsize of 0.5 for each of the datasets. In addition, we removed $n_{\boldsymbol{a}_a}$ from Equation 3 because this was shown to generally improve the accuracy in initial experiments.

For POT the adjustable parameters which affect the sensitivity of the method are q and a variable called lower quantile in the original paper [12]. We set q to 10^{-4} as in the original work. The other adjustable parameter, lower quantile, can not be set in accordance with the original paper as it was empirically changed for each individual dataset. Therefore, we set this variable to the mean of the different values used in the original paper (different values were set for tested datasets), which is 0.02.

The only parameter in top-k is the estimated amount of faults in the test set. We chose this parameter by using the mean amount of anomalies for the datasets which is 9.2%.

4) Validation

To achieve a fair and consistent interaction, we simulated the actions of the user, meaning that no experiments with a real user were conducted. This was achieved by first assuming that the user had knowledge of the ground truth, meaning that when the user was shown a certain segment they would be able to correctly classify it. Secondly, it was assumed that a reasonable user would only accept anomalous segments with a high ratio of true positive points. We set this limit to 10% for segments with less than 300 points, 30% for more than 300 points, 50% for more than 500 points, 70% for more than 700 points and 90% for more than 900. This means for example that if a segment with 100 points has more than 10 true positive points, the segment will be accepted and rejected otherwise.

V. RESULT

A. COMPARISON TO AUTOMATIC THRESHOLD-SETTING METHODS

The accuracy result for the experiment is presented in Table 1. As can be seen, the performance for the general settings of NDT, top-k and POT varies significantly between different datasets. More importantly, UVT achieved on average a 65% increase in f_{c1} score from the mean accuracy of the best performing automated threshold-setting method (see Table 2 and Figure 3). Not surprisingly, top-k showed the worst performance of all methods. Considering the user validation of all anomalous points based on the initial thresholds (RV), the accuracy goes up but so does the overall number of interactions. In addition, UVT was able to perform only slightly worse than RV which highlights the effectiveness of the suggested method. Figure 4 shows the average interactions of the simulated user. As can be seen, relatively little effort was needed considering both OmniAnomaly and GDN to achieve the result.

TABLE 1. Accuracy using the different threshold-setting methods on the different datasets.

Dataset	Scoring method	Threshold-setting method									
		POT		NDT		Top-k		UVT		RV	
		Interactions	f_{c1}	Interactions	f_{c1}	Interactions	f_{c1}	Interactions	f_{c1}	Interactions	f_{c1}
SWaT	OmniAnomaly	-	0.2845	-	0.0273	-	0.0265	44	0.5635	1640	0.6226
	GDN	-	0.4204	-	0.205	-	0.354	44	0.6225	4272	0.7007
WADI	OmniAnomaly	-	0.1081	-	0.3501	-	0.0935	30	0.6781	2228	0.7662
	GDN	-	0.6015	-	0.4390	-	0.4158	30	0.6963	1294	0.8369
SMD	OmniAnomaly	-	0.2031	-	0.5177	-	0.3328	30	0.7614	472	0.7664
	GDN	-	0.5113	-	0.6414	-	0.4026	30	0.8396	512	0.8910
MSL	OmniAnomaly	-	0.1615	-	0.3885	-	0.2574	25	0.4892	414	0.5476
	GDN	-	0.2358	-	0.5165	-	0.3365	29	0.5506	325	0.5478
SMAP	OmniAnomaly	-	0.5874	-	0.5934	-	0.2959	17	0.6233	193	0.4558
	GDN	-	0.2140	-	0.2093	-	0.2247	22	0.6602	302	0.4473

TABLE 2. Average accuracy for each of the threshold-setting methods.

Threshold-setting method	OmniAnomaly				GDN			
	P_t	R_e	f_{c1}	Interactions	P_t	R_e	f_{c1}	Interactions
POT	0.1835	**0.8200**	0.2690	-	0.2998	0.7034	0.3966	-
NDT	0.5933	0.3104	0.3754	-	0.5906	0.3925	0.4022	-
Top-k	0.1200	0.6422	0.2012	-	0.3313	0.7714	0.3467	-
UVT	**0.6381**	0.6852	0.6231	**30**	**0.7172**	0.7034	0.6738	**31**
RV	0.5831	0.7721	**0.6318**	989	0.65867	**0.8032**	**0.6847**	1341

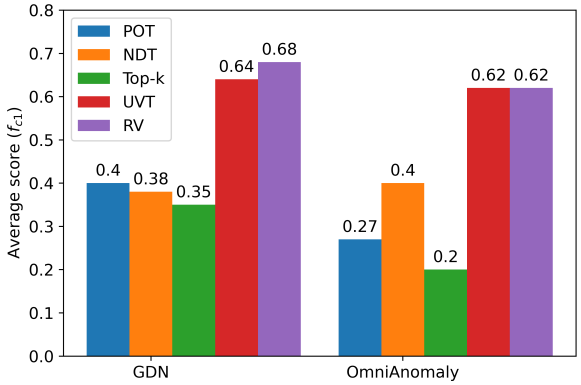

FIGURE 3. Average f_{c1} score for the datasets based on OmniAnomaly and GDN using the different threshold-setting methods.

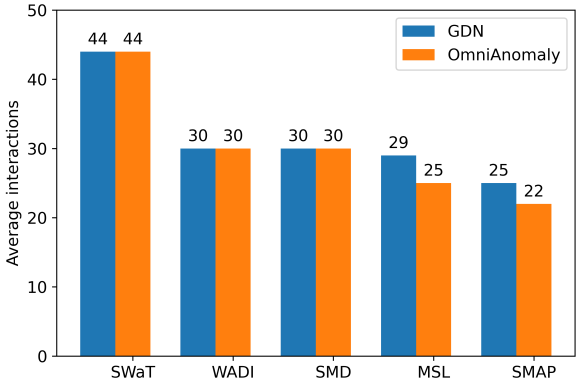

FIGURE 4. Average simulated interaction for all datasets and underlying anomaly scoring method.

B. ABLATION STUDY

We also explored the implication of removing individual parts of UVT with the purpose of evaluating the importance of these. We considered two different scenarios, namely without the separated scoring and clustering which can be seen in Table 3 and 4. Without the separation of the different channels, the number of interactions goes down, however, so does the accuracy. In addition, the performance is higher than simply using the automated methods which, considering the relatively few interactions required, seems to be an alternative when aggregated scoring is utilized. The smallest difference was with regard to the clustering procedure. A similar accuracy could be observed for the different underlying anomaly detection methods and datasets. However, for both GDN and OmniAnomaly, the number of interactions was lower when using the clustering procedure. In addition, it is clear that the interaction time when not using the method becomes extensive.

VI. DISCUSSION

The results from the presented UVT method generate excellent thresholds with limited expert interactions, which is supported by the 65% increase in f_{c1} score from the tested automated threshold-setting method with the best performance only using on average around 30 user interactions. Considering this substantial increase in performance, the number of interactions is easily justifiable. Furthermore, the

TABLE 3. Accuracy for the different interactive threshold-setting scenarios on the different datasets.

Dataset	Scoring method	Interative threshold-setting scenario					
		UVT		UVT w/ aggregated scoring		UVT w/o cluster	
		Interactions	f_{c1}	Interactions	f_{c1}	Interactions	f_{c1}
SWaT	OmniAnomaly	44	0.5635	22	0	44	0.5587
	GDN	44	0.6225	4	0.2050	44	0.6242
WADI	OmniAnomaly	30	0.6781	4	0.3501	30	0.6831
	GDN	30	0.6963	4	0.4390	30	0.6891
SMD	OmniAnomaly	30	0.7614	18	0.6651	30	0.7185
	GDN	30	0.8396	25	0.7179	30	0.8085
MSL	OmniAnomaly	25	0.4892	9	0.5882	28	0.5858
	GDN	29	0.5506	15	0.3789	28	0.4974
SMAP	OmniAnomaly	17	0.6233	6	0.6747	21	0.6082
	GDN	22	0.6602	11	0.6366	25	0.6896

TABLE 4. Average accuracy for each of the interactive threshold-setting methods.

Threshold-setting method	OmniAnomaly		GDN	
	Interactions	f_{c1}	Interactions	f_{c1}
UVT	30	0.6231	31	**0.6738**
UVT w/ aggregated scoring	**12**	0.4556	**12**	0.4755
UVT w/o cluster	31	**0.6309**	33	0.6618

results indicate that the method can be applied to different underlying models regardless of scoring functions without needing to retrain or in other ways change the model. As several different datasets were used, the results strongly suggest that using a labelling strategy such as the one presented in this paper can outperform an automated threshold-setting method on a general basis. Still, there were cases where the automated threshold-setting methods with a consistent setup performed close to the suggested method.

In terms of the complexity of the suggested method, it includes several different parameters that likely have an impact on the validation process. No extensive parameter search has been conducted, however, the ablation experiment showed that the clustering procedure potentially had the least effect on the result. In addition, it is reasonable to assume that the number of interactions has a significant impact on the overall performance. As an example, Figure 5 shows the performance for GDN on the SWaT dataset where the performance for clustering and not clustering is compared. In the clustering case, the performance increases faster than when not using clustering but does not have as high maximum accuracy as when not clustering. This makes sense, since when clustering fewer segments are shown to the user, but it risks making slight errors such as providing false positive points connected to a ground truth segment. The graph also shows the importance of the number of iterations for performance and while the optimal number of interactions is unknown in advance, it is reasonable to set the max number based on the number of data points, such as was done in the experiments, and to use a minimum limit such as 30. Furthermore, the consistency of the user interaction will have a large impact on the performance of the suggested method because it affects the possibility to optimize the threshold after the validation procedure.

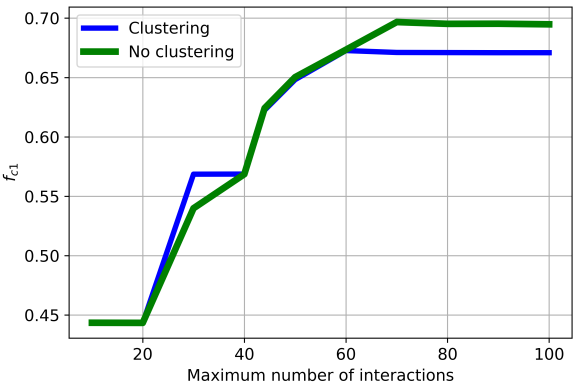

FIGURE 5. Number of interactions for UVT with and without clustering.

The time cost of the suggested method is of importance to consider but has not been explored, similarly to other user interaction studies [18], [19]. The reason for this was the use of a simulated user where conversion to human interaction time cannot be properly translated. Furthermore, the interaction time is also likely dependent on how the system that incorporates UVT is implemented. Lastly, it is reasonable to assume that a user who gets more practice will perform the validation procedure faster over time.

Overall, the right method for constructing thresholds for an anomaly detection method for multivariate anomaly detection arguably differs depending on several factors. It is reasonable to assume that the best threshold-setting method is a method that generates a high accuracy with no interactions. Unfortunately, the result of this study suggests that this is difficult to achieve in practice and the performance of the automated methods may vary depending on the dataset, the underlying model and the scoring function. The automated threshold-setting methods explored in this study can be useful in scenarios when an interactive validation procedure is

not possible to conduct or when for example precision is not that important for the application. However, when input from the user is possible, the result from this study strongly suggests that the UVT is most suitable on a general basis.

Lastly, this study has restricted the evaluation of UVT on machine learning-based methods. However, since UVT works independently of the underlying method provided that it generates an anomaly score, it is likely that it will show similar performance on methods with other types of underlying models, such as statistical. Independently of model type, it is recommended for best performance, as shown in the experiments, to use UVT based on a scoring function utilizing separate scoring. In addition, while the method has specifically been designed for multivariate data, it will also work on univariate data.

VII. CONCLUSION

This study has examined to what extent a user validation schema to set thresholds for anomaly detection methods in multivariate time series can outperform state-of-the-art automated threshold-setting methods. To achieve this, a new method called UVT has been developed that utilizes separate scoring, clustering and prioritization to minimize user interaction, but with high accuracy. The result has shown that UVT can, with little effort from the user, outperform the automated threshold-setting methods POT, top-k and NDT. Furthermore, it has shown the effectiveness of two different models and scoring functions without the need to change or retrain the underlying model. Therefore, the result strongly indicates that UVT can successfully be used as a general method to optimize the performance of anomaly detection models on multivariate time series data.

References

[1] W. Mao, H. Shi, G. Wang, and X. Liang, "Unsupervised deep multitask anomaly detection with robust alarm strategy for online evaluation of bearing early fault occurrence," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–13, 2022, ISSN: 1557-9662. DOI: 10.1109/TIM.2022.3200092.

[2] Z. Li, Y. Sun, L. Yang, Z. Zhao, and X. Chen, "Unsupervised machine anomaly detection using autoencoder and temporal convolutional network," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–13, 2022, ISSN: 1557-9662. DOI: 10.1109/TIM.2022.3212547.

[3] Z. Zeng, G. Jin, C. Xu, S. Chen, Z. Zeng, and L. Zhang, "Satellite telemetry data anomaly detection using causal network and feature-attention-based LSTM," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–21, 2022, ISSN: 1557-9662. DOI: 10.1109/TIM.2022.3151930.

[4] A. Listou Ellefsen, P. Han, X. Cheng, F. T. Holmeset, V. Æsøy, and H. Zhang, "Online fault detection in autonomous ferries: Using fault-type independent spectral anomaly detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 8216–8225, Oct. 2020, ISSN: 1557-9662. DOI: 10.1109/TIM.2020.2994012.

[5] 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, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3107975. [Online]. Available: https://ieeexplore.ieee.org/document/9523565/ (visited on 03/14/2022).

[6] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: UnSupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20, New York, NY, USA: Association for Computing Machinery, Aug. 23, 2020, pp. 3395–3404, ISBN: 9781450379984. DOI: 10.1145/3394486.3403392. [Online]. Available: https://doi.org/10.1145/3394486.3403392 (visited on 03/14/2022).

[7] S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep transformer networks for anomaly detection in multivariate time series data," *Proceedings of the VLDB Endowment*, vol. 15, no. 6, pp. 1201–1214, Feb. 2022, ISSN: 2150-8097. DOI: 10.14778/3514061.3514067. [Online]. Available: https://dl.acm.org/doi/10.14778/3514061.3514067 (visited on 05/23/2023).

[8] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys*, vol. 54, no. 2, 38:1–38:38, Mar. 5, 2021, ISSN: 0360-0300. DOI: 10.1145/3439950. [Online]. Available: https://doi.org/10.1145/3439950 (visited on 03/28/2022).

[9] M. Zhang, D. Wang, W. Lu, J. Yang, Z. Li, and B. Liang, "A deep transfer model with wasserstein distance guided multi-adversarial networks for bearing fault diagnosis under different working conditions," *IEEE Access*, vol. 7, pp. 65 303–65 318, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2916935.

[10] D. Li, D. Chen, L. Shi, B. Jin, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," *arXiv:1901.04997 [cs, stat]*, Jan. 15, 2019. arXiv: 1901.04997. [Online]. Available: http://arxiv.org/abs/1901.04997 (visited on 03/15/2022).

[11] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, New York, NY, USA: Association for Computing Machinery, Jul. 19, 2018, pp. 387–395, ISBN: 9781450355520. DOI: 10.1145/3219819.3219845. [Online]. Available: https://doi.org/10.1145/3219819.3219845 (visited on 06/20/2022).

[12] 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," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19, New York, NY, USA: Association for Computing Machinery, Jul. 25, 2019, pp. 2828–2837, ISBN: 9781450362016. DOI: 10.1145/3292500.3330672. [Online]. Available: https://doi.org/10.1145/3292500.3330672 (visited on 03/15/2022).

[13] 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*, pp. 1–10, 2021, ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2021.3105827. [Online]. Available: https://ieeexplore.ieee.org/document/9525836/ (visited on 03/14/2022).

[14] A. L. Alfeo, M. G. C. A. Cimino, G. Manco, E. Ritacco, and G. Vaglini, "Using an autoencoder in the design of an anomaly detector for smart manufacturing," *Pattern Recognition Letters*, vol. 136, pp. 272–278, Aug. 1, 2020, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2020.06.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520302269 (visited on 11/09/2022).

[15] B. Zong, Q. Song, M. R. Min, *et al.*, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," presented at the International Conference on Learning Representations, Feb. 15, 2018. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0- (visited on 03/15/2022).

[16] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, Online Real-Time Learning Strategies for Data Streams, vol. 262, pp. 134–147, Nov. 1, 2017, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.04.070. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217309864 (visited on 03/23/2023).

[17] S. Guan, B. Zhao, Z. Dong, M. Gao, and Z. He, "GTAD: Graph and temporal neural network for multivariate time series anomaly detection," *Entropy*, vol. 24, no. 6, p. 759, Jun. 2022, ISSN: 1099-4300. DOI: 10.3390/e24060759. [Online]. Available: https://www.mdpi.com/1099-4300/24/6/759 (visited on 09/27/2022).

[18] Z. Li, Y. Zhao, Y. Geng, *et al.*, "Situation-aware multivariate time series anomaly detection through active learning and contrast VAE-based models in large distributed systems," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2746–2765, Sep. 2022, ISSN: 1558-0008. DOI: 10.1109/JSAC.2022.3191341.

[19] W. Wang, P. Chen, Y. Xu, and Z. He, "Active-MTSAD: Multivariate time series anomaly detection with active learning," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, ISSN: 2158-3927, Jun. 2022, pp. 263–274. DOI: 10.1109/DSN53405.2022.00036.

[20] A. Lundström, M. O'nils, F. Z. Qureshi, and A. Jantsch, "Improving deep learning based anomaly detection on multivariate time series through separated anomaly scoring," *IEEE Access*, vol. 10, pp. 108 194–108 204, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3213038.

[21] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. [Online]. Available: https://www.aaai.org/AAAI21Papers/AAAI-5076.DengA.pdf (visited on 03/15/2022).

[22] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning graph structures with transformer for multivariate time series anomaly detection in IoT," *IEEE Internet of Things Journal*, pp. 1–1, 2021, ISSN: 2327-4662, 2372-2541. DOI: 10.1109/JIOT.2021.3100509. [Online]. Available: https://ieeexplore.ieee.org/document/9497343/ (visited on 03/15/2022).

[23] "iTrust, centre for research in cyber security, singapore university of technology and design," iTrust. (), [Online]. Available: https://itrust.sutd.edu.sg/itrust-labs_datasets/ (visited on 03/15/2022).

ADAM LUNDSTRÖM received the M.S. degree in Industrial Engineering and Management from Mid Sweden University and the M.S. degree in Computer and Systems Sciences from Stockholm University in 2020. He is currently a Ph.D. student at the Industrial graduate school Smart Industry Sweden and studying the potential for machine learning solutions in predictive maintenance. He is employed by SCA and a part of the Department of Electronics Design at Mid Sweden University.

MATTIAS ONILS received the B.S. degree in electrical engineering from Mid Sweden University, Sundsvall, Sweden, in 1993, and the Licentiate and Ph.D. degrees in electronic systems design from the Royal Institute of Technology, Stockholm, Sweden, in 1996 and 1999, respectively. He is currently a professor with the Department of Electronics Design and leads a research group in Embedded IoT Systems at Mid Sweden University. His current research interests include design methods and implementation of embedded DNN-based systems, especially in the implementation of real-time video processing and time series processing systems.

FAISAL Z. QURESHI (Senior Member, IEEE; Member ACM; Secretary and Member CIPPRS) received the B.Sc. degree in Mathematics and Physics from Punjab University, Lahore, Pakistan, in 1993, the M.Sc. degree in Electronics from Quaid-e-Azam University, Islamabad, Pakistan, in 1995, and the M.Sc. and Ph.D. degrees in Computer Science from the University of Toronto, Toronto, Canada, in 2000 and 2007, respectively. He is a Professor of Computer Science in the Faculty of Science, Ontario Tech University, where he leads the Visual Computing Lab. His research focuses on computer vision, and his scientific and engineering interests center on the study of computational models of visual perception to support autonomous, purposeful behavior in the context of ad hoc networks of smart cameras. Dr. Qureshi is also active in journal special issues and conference organizations. He served as the general co-chair for the Workshop on Camera Networks and Wide-Area Scene Analysis (co-located with CVPR) in 2011-13. He also served as the co-chair of Computer and Robot Vision (CRV) conference 2015/16 meetings.