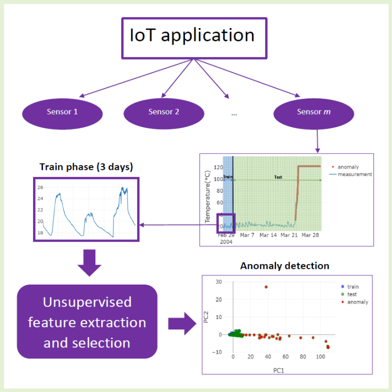
Expect the Unexpected: Unsupervised feature selection for automated sensor anomaly detection

Hui Yie Teh, Kevin I-Kai Wang and Andreas W. Kempa-Liehr

***Abstract*— With the growth of IoT applications, sensor data quality has become increasingly important to ensure the success of these data-driven applications. Sensor data riddled with errors are redundant and affects the accuracy of decision-making results. Moreover, several constraints are inherent in anomaly detection for IoT applications such as limited manpower, time, bandwidth, computational resources and the lack of labelled datasets. Most machine learning algorithms also require a large training set to fit a satisfactory model. In this paper, we propose a fully automated anomaly detection framework, which combines systematic time series feature engineering with unsupervised feature selection. The unsupervised feature selection approach automatically selects time series features based on their predictive power with respect to the statistics of near future measurements. The proposed framework also only needs a short calibration phase with respect to the deployment phase to detect anomalies despite not having seen any in training, which is critical for real-world applications. Its feature selection reduces the amount of data required in the deployment phase on an IoT application. The selected features are suitable for building a reliable anomaly detection model while achieving a similar or better anomaly detection performance than established methods, which are operating on the raw data. Results of the evaluation on two publicly available environmental monitoring datasets show that our proposed unsupervised feature selection approach is a crucial step for having a more accurate anomaly detection while providing complex application-specific time series features, which are safeguarding the sensor system against unseen sensor anomalies.**



***Index Terms*— Anomaly detection, Feature selection, Principal component analysis, Sensor data quality, Time series analysis**

I. Introduction

WITH the advent of Internet of Things (IoT) technologies, vast amounts of data are being generated in an array of fields such as healthcare, industry, agriculture, meteorology and transport. Cisco [1] estimated that there would be approximately 14.7 billion IoT devices by 2023, each generating its own data. The quality of sensor data has become increasingly vital to ensure the success of these data-driven applications. All of the data is then transmitted through the sensor network and into a central system, e.g. an edge or a cloud server. Without proper calibration and laborious manual validation, the data may be riddled with errors, making it redundant. Therefore, sensor data quality plays a critical role, where poor sensor data quality affects the decision-making results and the accuracy of data-driven applications.

For example, in healthcare, low-quality data may affect medical diagnosis and threaten the life of the person whose condition is being monitored [2]. Moreover, weather forecasting models are also dependent on the quality of sensor data and predictions from those models become unreliable as many of them, such as Principal Component Analysis (PCA), are dependent on fault-free training data [3]. Therefore, automated anomaly detection is needed to discern between normal and anomalous data such that an action e.g. an alert or imputation can be carried out to ensure sensor data quality which in turn improves decision-making results.

Sensor data errors or anomalies stem from various sources. IoT sensors themselves operate on constrained resources such as limited battery and computational power and are deployed in hostile environments [4], [5]. The damage or exhaustion of battery in sensor devices often causes data quality to degrade, as towards the end of its battery life, sensors tend to produce unreliable readings. In-situ sensors are also subjected to the volatile environment in which they are deployed, where the extreme conditions might affect the operation of the sensor e.g. a sensor might also be dislodged due to the strong winds and a light sensor may be covered in snow, which causes

H. Y. Teh is with the Department of Electrical, Computer and Software Engineering, The University of Auckland, New Zealand (e-mail: hteh703@aucklanduni.ac.nz).
K. I. Wang is with the Department of Electrical, Computer and Software Engineering, The University of Auckland, New Zealand (e-mail: kevin.wang@auckland.ac.nz).
A. W. Kempa-Liehr is with the Department of Engineering Science, The University of Auckland, New Zealand (e-mail: a.kempa-liehr@auckland.ac.nz) and Freiburg Materials Research Center, University of Freiburg, Germany.

it to produce false readings. From a networking perspective, data loss and corruption arise from congested or unstable communication links in wireless sensor networks [2].

All of these errors are reflected as anomalies in the time series of an IoT application. In this paper, the term *anomalies* relates to the faults that occur in sensor data such as outliers, constant values, missing data and drifts, which should be detected in order to improve sensor data quality. Teh et al. [6] provides a comprehensive description of the types of errors, which is synonymous with the term *anomalies* used in this paper. In general, an anomaly is an observation that largely deviates from the normal behaviour or is inconsistent with the rest of the dataset [7], [8].

Wireless sensor network (WSN) applications such as environmental monitoring have tens and thousands of sensing devices which requires manual calibration and constant monitoring of each individual sensor. This manual approach to validate incoming data is not feasible due to limited manpower and time. The constrained network bandwidth also means that the sending of faulty data will incur unnecessary costs. The following are the constraints of WSNs for detecting anomalies in sensor data:

- Limited manpower and time: Manually inspecting the sensing devices individually is labour and time-intensive.
- Limited data: The amount of calibration and training data is very limited, due to a short calibration period. Other than that, ground-truth data is not readily available and is hard to obtain.
- Limited computational resources and bandwidth: The remote sensing devices are able to conduct less local processing due to limited resources such as power and battery. Therefore, it is more efficient in terms of communication to not send too much data, especially erroneous ones, as it can waste the already limited resources.

Most state-of-the-art anomaly detection solutions are able to overcome the first constraint to some extent, however some works are not fully automated as manual input and domain knowledge is required in selecting parameters such as features for the machine learning model. Other than that, the solutions mostly require a lot of training data, which is not feasible in real-world applications where the calibration phase is short as explained in the second constraint. Supervised methods depend on ground-truth labels, which are hard to get. Lastly, current approaches such as clustering are unable to tackle the third constraint as it requires all data to be sent to the cloud/server-side where the anomalies are detected retrospectively and not in real-time on the sensing devices themselves.

Our proposed approach is a fully automated anomaly detection framework that uses unsupervised feature selection on systematically engineered time series features along with PCA to detect sensor data anomalies. The novel unsupervised feature selection approach can be applied to newly deployed sensors for which no anomalies have been observed, yet. The algorithm automatically selects time series features based on hypothesis testing with respect to their abilities to predict statistics of near-future values. Therefore, the feature selection is unsupervised with respect to the non-existing labels of sensor anomalies. Combining the selected features with PCA allows anomaly detection to be more reliable and fully automated without the need for labelled ground truth data. Once the model is trained, anomaly detection can be done in real-time on the edge devices themselves. The handful of selected features reduces the complexity of the error detection problem which makes it less computationally intensive to train an anomaly detection model and saves bandwidth by detecting anomalies early and not unnecessarily transmitting error-ridden data.

The framework also consists of two phases, the calibration (train) phase and the deployment (test) phase. In this paper, we aim to propose an anomaly detection framework that uses a very short calibration phase with respect to the overall long-term deployment phase for typical IoT applications. In our case, we selected three days of calibration, which corresponds to a period that is 10% or less of the overall deployment period of the evaluated datasets. Our evaluation of two real-world datasets that were both one to three months long, demonstrates that the three-day calibration phase works reliably for anomaly detection problems, despite not seeing any anomalies in training. This is vital for real-world applications where oftentimes it is difficult to obtain labelled datasets and the calibration phase is very short with respect to the deployment phase.

The contributions of this paper are three-fold:

1) A novel unsupervised feature selection approach which combines systematic time series feature engineering with feature selection for sensor anomaly detection problems, which is agnostic to the different types of sensor anomalies and sensor data types,
2) An automated anomaly detection framework for time series data from physical sensors using unsupervised feature selection coupled with Principal Component Analysis,
3) Demonstration of the applicability of the developed framework on two public datasets on environmental monitoring applications.

The rest of the paper is organized as follows. Section II discusses the state-of-art for time series anomaly detection whereas Section III details our proposed unsupervised feature selection approach and the automated anomaly detection framework, which incorporates the unsupervised feature selection technique. Section IV describes the experimental setup and the publicly available datasets used for evaluation and the results of the experiments are shown in Section V. Section VI concludes the paper.

II. Related work

The following subsections present the current research done in the sensor anomaly detection field. It is separated into two subsections. Section II-A introduces the state-of-the-art methods for anomaly detection, which includes mainly machine learning approaches such as Principal Component Analysis, Artificial Neural Networks, Ensemble classifiers and Clustering. The next subsection, Section II-B, subsequently discusses anomaly detection solutions that improve the machine learning models by focusing on feature extraction and selection techniques.

A. Anomaly detection techniques

There has been extensive interest and work done on anomaly detection in sensor data quality-related problems [6], [9]–[12]. PCA is commonly used for anomaly detection in sensor data. Dunia et al. [13] proposed a Sensor Validation Index (SVI) as a means for identifying sensor faults through an iterative reconstruction process that assumes each sensor fails, reconstructs the faulty sensor and compares the Q-statistics before and after reconstruction. Rather than just using the SVI, Harkat et al. [14] also applied PCA to detect anomalies such as outliers, drifts, bias, stuck-at-zero faults using a test on the sum of squares of the residual matrix, i.e. the last non-selected principal components to detect faults. Zhang et al. [15] on the other hand, introduced a combined contributions index using the Q-statistics, which measures the variance of random noise in the residual subspace and Hotelling's T^2-statistics, which represents the variance in the PCA model's subspace.

In order to deal with non-linear systems, Sharifi and Langari [16] suggested a Mixture Probabilistic PCA model for fault diagnosis, which separates the input space into several local linear regions and subsequently has linear sensor fault diagnosis applied to each linear region. Recently, Mansouri et al. [17] came up with another variation of PCA called the Midpoint-radii PCA for fault detection in an air quality monitoring network. The Midpoint-radii PCA allows interval-valued data, which considers the uncertainty in the data to be modelled. Moreover, in the healthcare domain, Zhao and Fu [18] have also proposed a sensor fault detection for outliers and bias using PCA by modelling the normal behaviour for continuous glucose monitoring applications.

For local and unsupervised time series anomaly detection, a variation of PCA called the One-Class Principal Component Classifier was proposed by Rassam et al. [8]. The approach is divided into two phases, with the first being the offline training phase which trains a PCA model using normal data collected from each sensor and the second phase being the online detection phase where current observations would be projected into the feature subspace and compared with the normal behaviour model based on the dissimilarity matrix. The normal PCA model is also updated and retrained with the new mean and standard deviation of the new data. However, all the PCA methods mentioned previously, including this, do not include any systematic time series feature extraction and selection. Instead, these algorithms operate on the sensor measurements themselves. For datasets with a large amount of time series data, this becomes a problem as it becomes computationally expensive and causes overfitting of the PCA model.

Clustering is also another unsupervised technique for anomaly detection which has the advantage of not requiring prior knowledge of the system model or underlying data distribution. It also does not require the training of a model. However, there are many parameters of the clustering algorithm such as the optimal number of clusters or cluster widths that has to be predetermined by the user. One of the clustering-based outlier detection technique is proposed by Fawzy et al. [19] for WSNs. The algorithm uses an in-network fixed-width clustering algorithm to determine and label each cluster formed as “normal” or “outlier” by calculating the Euclidean distance between clusters. For electric power applications, Liu et al. [20] presented another example of the clustering method used for outlier detection using Time-Relevant k-Means clustering for electric power sensor data. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is also a common clustering technique used in anomaly detection as seen in [21]–[23]. DBSCAN is able to find any irregular-shaped clusters and the number of clusters do not need to be set in advance. Although these clustering algorithms are unsupervised and do not require training, the anomaly detection needs to be done offline on historical data.

Other machine learning models that can be used to detect anomalies include Artificial Neural Networks [24] and its variations Time-Delay Neural Network [25], Extreme Learning Machines [26], [27], Auto-associative Neural Network [28] and Hierarchical Temporal Memory [29]. A Neural Network approach based on Multi-Layer Perceptron has also been proposed by Darvishi et al. [30] for sensor fault detection, isolation and accommodation in Digital Twin applications. However, virtual sensors do not have the same constraints as physical sensors such as the lack of computational power. With that being said, our proposed unsupervised feature selection approach is able to complement the pre-processing of machine learning models, including neural networks.

Furthermore, Ensemble Classifiers [31]–[34] and Support Vector Machines (SVM) with its two different variations, One-Class Centered Quarter-Sphere SVM [35] and One-Class Centered Hyper-Ellipsoidal SVM [36] are machine learning techniques that are also used for anomaly detection. Zidi et al. [37] also proposed a real-time fault detection using SVM classifiers. However, a labelled dataset is required in the learning phase for the decision function used to detect faults, which is different to the unsupervised approach we are aiming for. From a data fusion perspective, Yazidi and Herrera-Viedma [38] proposed a method for identifying unreliable sensors via Learning Automata, however only for binary sensor readings. Russo et al. [39] use active learning which is a special case of machine learning which balances the use of supervised machine learning models and the burdens associated with manual labelling by human experts. There are some existing solutions that include manual feature extraction and selection, such as [25], [33], [34] whereby the features are chosen by domain experts. However, this too is impractical as it is time-consuming and requires a significant amount of expert knowledge and cannot be generalized across all datasets.

B. Feature extraction and selection techniques

As mentioned previously, many of the existing solutions for sensor data anomalies are done using trivial feature extraction and selection. The solutions presented either used raw data e.g. [8], [19], or common features [25], [33], [34] such as:

- rate of change,
- spatial distance,
- mean,
- standard deviation,

- variance,
- correlation coefficient,
- temporal correlation (e.g.moving average or incremental variance),
- spatial correlation (e.g. distance between nodes),
- seasonal distribution error,
- linear prediction error,
- time since last calibration.

This small set of features does not fully describe the characteristics of a dataset or time series and is unable to generalize to other datasets. For example, the seasonal distribution error feature for environment monitoring applications may not be meaningful or applicable to healthcare monitoring applications. Thus, other methods for feature extraction and selection have been studied. Zang et al. [40] introduced a feature extraction approach based on the transfer probability of Markov chains. These Markov features combined with the mean and variance become the feature set to be used in machine learning models.

Recently, `tsfresh`, a machine learning library for systematic time series feature extraction [41], has been a popular choice for feature extraction and has been used in several domain-specific approaches. For power consumption applications, Ouyang et al. [42] proposed a systematic time series feature extraction using `tsfresh` to detect anomalous power consumption users. The 794 extracted features are used as input for a supervised binary classification model. However, Ouyang et al. did not facilitate any feature selection, which renders the approach computationally expensive, such that their proposed solution may not be feasible for real-time deployments and is subjected to the *curse of dimensionality*.

In structural health monitoring systems, Liu et al. [43] suggested a solution for sensor fault classification using `tsfresh` for automated feature extraction and selection. The feature selection algorithm is a combination of automatically configured univariate hypothesis tests with controlling of the False Discovery Rate [44] and it is used to obtain a subset of meaningful features to be used as input to a Long Short-Term Memory Deep Neural Network model. However, this feature selection process still returned more than a hundred features.

In order to detect anomalies in electricity consumption behaviours, Zhang et al. [23] proposed an unsupervised anomaly detection method using clustering (DBSCAN) and feature engineering. The feature engineering process also used `tsfresh` as a feature extraction technique to extract a comprehensive set of features and the feature selection is based on variance and the Maximum-Relevance Minimum-Redundancy technique with Maximal Information Coefficient. However, this approach works on historical data and cannot be applied to real-time/streaming data.

C. Motivation and Contributions

Previous research, which is closely related to this proposed approach is compared in Table I. The characteristics of each anomaly detection method proposed from related works are compared, such as the supervised or unsupervised nature of the method, whether it involves feature engineering/extraction and feature selection, the size of the training set, whether it is able to perform real-time detection and if it was evaluated on real-world environmental data. It is evident that there is a lack of unsupervised feature selection approaches with small training sets which are able to detect anomaly in real-time in the current state-of-the-art. Almost all approaches do not apply feature engineering, save for [34], which only has trivial/manual feature extractions and [23] who has used a more comprehensive feature extraction and selection method, but the features are extracted from a large training set. Furthermore, machine learning classifiers often have large training sets, and although clustering approaches, [21], [23], [36] have no need for training, they are unable to perform real-time anomaly detection.

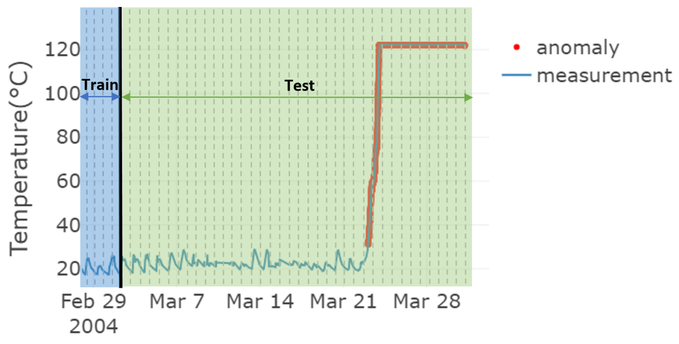


Fig. 1**:** Example sensor time series with initial calibration phase (train) for feature selection and PCA. The plot shows the temperature time series of sensor device 1 from the IBRL dataset [45]. The training period (blue) is only 10% (three days) of the raw time series, which is used in the calibration phase. The test period (green) comprises the remaining 90% of the data Vertical dashed lines indicate boundaries of rolling windows.

Therefore, using our novel, fully automated and unsupervised feature selection approach, the proposed anomaly detection framework is able to tackle the aforementioned issues. The unsupervised feature selection is done by automatically selecting time series features based on their abilities to predict statistics of near-future values. The features are then used as input to train an anomaly detection model based on PCA, which only requires a small calibration phase (training set), which we have selected to be three days in this paper, and without requiring any ground-truth labels. In addition to that, our proposed framework can potentially be deployed on the sensor devices themselves once the PCA model is trained, allowing us to perform anomaly detection in real-time.

III. Methodology

A brief description of the problem is represented in the following subsection, Section III-A. Next, the proposed unsupervised feature selection technique, which is the core contribution of this paper, is detailed in Section III-B. Following that, Section III-C presents an overview of another core contribution of the paper, which is the automated anomaly detection framework with the proposed unsupervised feature selection technique.

Attributes \ Papers	Zhang et al. [36]	Rassam et al. [8]	Rahman et al. [34]	Zhou et al. [21]	Russo et al. [39]	Zhang et al. [23]	Proposed method
Anomaly detection method	OC-SVM	PCA	Ensemble classifiers	DBSCAN	Active learning + ML classifiers	Feature engineering + DBSCAN	Feature engineering + PCA
Unsupervised	●	●	○	◑	○	●	●
Feature engineering	○	○	◑	○	○	●	●
Feature selection	○	○	○	○	○	●	●
Small training set	●	○	○	●	○	○	●
Real-time detection	●	●	○	○	○	○	◑
Environmental data	●	●	●	●	●	○	●

TABLE I**:** Comparison of related works and our proposed approach to identify the gap in the knowledge. The keys, ○, ◑, and ● represent "no/not considered or mentioned", "potentially applicable/partially considered", and "yes/fully considered" respectively. Only one approach included feature selection as part of their overall framework. Moreover, it can be seen that a large training set is required for machine learning classifiers. Clustering approaches, though have no need for training, do not work in real-time scenarios.

A. Problem description

A sensor time series dataset, $\mathcal{D} = \{\vec{s}_1, \dots, \vec{s}_m, \dots \vec{s}_M\}$ contains a collection of time series $\vec{s}_m$ from multiple sensor devices, where m is the index of each sensor device time series and M is the number of sensor devices in the dataset. An example of a raw time series from a sensor device can be seen in Fig. 1, which shows raw time series data collected from a physical sensor device (Sensor device 1 of the Intel Berkeley Research Lab (IBRL) dataset [45]), which is used in an indoor environment monitoring application. The temperature sensor starts off behaving normally, until towards the end, starting about March 23 where the sensor becomes faulty and produces anomalous readings. This is denoted by the red line in the figure. The feature selection process and anomaly detection framework involve a time series from a single sensor device, $\vec{s}_m = [x_1, \dots, x_n \dots, x_N]$, where n is the index of each univariate (e.g. temperature) sample/reading in the time series and N is the number of samples in the time series. Both, the unsupervised feature selection technique and automated anomaly detection framework are described in the following subsections.

B. Feature selection for unsupervised learning on sensor data

A time series typically contains a lot of noise and redundancies given the sheer volume of data [41]. Though feature extraction and selection is not compulsory for most machine learning algorithms as they are able to process the raw time series data directly, having appropriate features has been known to increase the accuracy of the machine learning models as meaningful features get rid of the uncertainty in the raw time series [43]. Moreover, feature selection reduces the dimensionality of the problem, which improves computational efficiency and reduces the risk of overfitting. Therefore, identifying time series features that are capturing the fault-free characteristics of the sensor is vital in detecting anomalies in sensor time series data. The following sections present two different versions of the proposed unsupervised feature selection, namely sensor-specific and deployment-specific feature selection, to be used in the anomaly detection framework

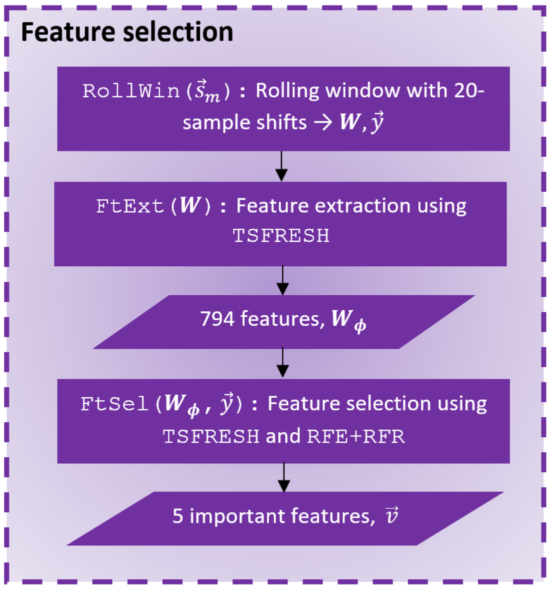

Feature selection
RollWin ($\vec{s}_m$) : Rolling window with 20-sample shifts → $\boldsymbol{W}, \vec{y}$
FtExt ($\boldsymbol{W}$) : Feature extraction using TSFRESH
794 features, $\boldsymbol{W}_\phi$
FtSel ($\boldsymbol{W}_\phi$, $\vec{y}$) : Feature selection using TSFRESH and RFE+RFR
5 important features, $\vec{v}$

Fig. 2**:** Flowchart of the proposed unsupervised feature selection for sensor data with generation of rolling windows (`RollWin`), feature extraction and standardization (`FtExt`), and feature selection (`FtSel`) combining univariate hypothesis testing (`TSFRESH`) with a random forest regressor (`RFE`) and recursive feature elimination (`RFE`). The algorithm is agnostic with respect to labelled anomalies but uses statistics $\vec{y}$ of future measurements for the feature selection.

(Section III-C). For the sensor-specific feature selection, each sensor device has its own set of selected features, whereas for the deployment-specific case, all sensor devices will share the same generic set of selected features.

1) Sensor-specific feature selection: Fig. 2 shows a flowchart of the proposed feature selection algorithm, which does not require any prior knowledge of the anomalies or ground-truth labels, thus is unsupervised. A rolling window is applied to the time series, which returns a matrix of windows, $\boldsymbol{W} = [\vec{w}_1, \dots, \vec{w}_p, \dots, \vec{w}_P]^\mathrm{T}$ where p is the index of each window, and P is the number of windows after the rolling/sliding window process. Every window, $\vec{w}_p$ contains $K = 120$ samples where each iteration of slide/roll will shift the window by $\Delta = 20$ samples. This rolling window process

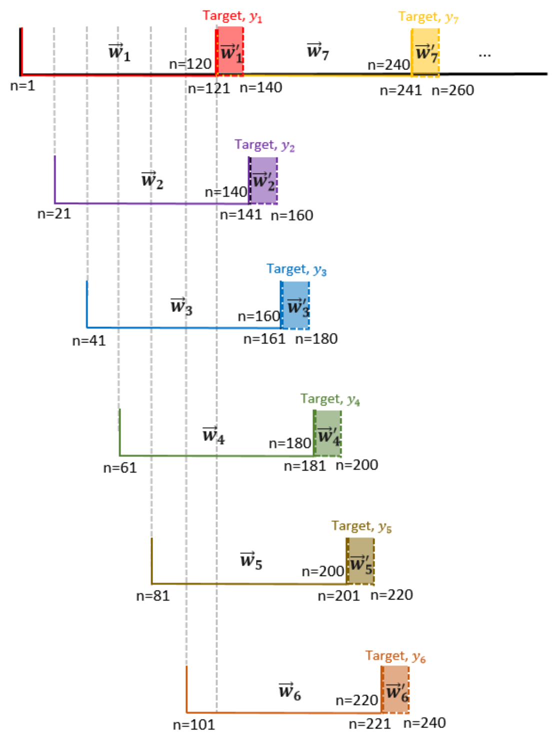


Fig. 3**:** Rolling window applied of calibration period. Every window is shifted by 20 samples, e.g. the first window, $\vec{w}_1$ starts at sample $n = 1$, and shifts by 20 samples in the next rolling window iteration, starting at $n = 21$, and so on.

is visualized in Fig. 3. The first window $\vec{w}_1$ starts from sample x_1 and ends at sample x_{120}. The next window $\vec{w}_2$ is obtained by shifting the starting point by 20 samples, such that the resulting window comprises samples x_{21} to x_{140}. For window $\vec{w}_p$ the samples are

$$\vec{w}_p = [x_{\Delta(p-1)+1}, \dots, x_{\Delta(p-1)+k}, \dots, x_{\Delta(p-1)+K}]. \quad (1)$$

The number of samples in each window remains fixed at $K = 120$ samples, i.e. $\vec{w}_p \in \mathbb{R}^{120}$. Furthermore, another matrix of windows, $\boldsymbol{W}' = [\vec{w}'_1, \dots, \vec{w}'_p, \dots, \vec{w}'_P]^T$ is generated as the target window matrix to be used for feature selection. Every window $\vec{w}'_p$ contains the next $I = 20$ samples for every current window $\vec{w}_p$. Thus,

$$\vec{w}'_p = [x_{\Delta(p-1)+K+1}, \dots, x_{\Delta(p-1)+K+i}, \dots, x_{\Delta(p-1)+K+I}]. \quad (2)$$

From the target window matrix $\boldsymbol{W}'$, we compute a target vector $\vec{y} = [y_1, \dots, y_p, \dots, y_P]$. Each target value y_p is obtained by extracting a specific feature from the values of the corresponding target window. E.g.,

$$y_p = \psi_{\text{mean}}(\vec{w}'_p) = \frac{1}{I}\sum_{i=1}^{I} x_{\Delta(p-1)+K+i}$$

computes the empirical mean of the target window $\vec{w}'_p$ (2). In general, different features should be considered for target generation, e.g. the first value of the target window, or a statistic like mean, standard deviation, or range of the time series values within the target window.

Next, feature extraction is performed on the rolled time series window matrix $\boldsymbol{W}$ using `tsfresh` [41], which extracts $Q = 794$ time series features automatically. Given $\boldsymbol{W}$, a feature mapping $\phi(\vec{w})$ is used to characterize each time series window. For example, the mapping $\phi_{\max}(\vec{w}_p)$ finds the maximal value in the rolled time series window $\vec{w}_p$ (1):

$$\phi_{\max}(\vec{w}_p) = \max\ \{x_{\Delta(p-1)+k}\}_{k=1}^{K}.$$

This feature extraction process results in a feature matrix $\boldsymbol{W}_\phi \in \mathbb{R}^{P\times Q}$ with

$$\boldsymbol{W}_\phi = \begin{bmatrix} \phi_1(\vec{w}_1) & \dots & \phi_q(\vec{w}_1) & \dots & \phi_Q(\vec{w}_1) \\ \vdots & \vdots & \vdots & & \\ \phi_1(\vec{w}_p) & \dots & \phi_q(\vec{w}_p) & \dots & \phi_Q(\vec{w}_p) \\ \vdots & \vdots & \vdots & & \\ \phi_1(\vec{w}_P) & \dots & \phi_q(\vec{w}_P) & \dots & \phi_Q(\vec{w}_P) \end{bmatrix}.$$

Here, q is the index of the feature mapping. This process is known as feature extraction or systematic time series feature engineering. Not all extracted features are useful for the detection of anomalies. Redundant features may cause overfitting of the machine learning model and reduce the performance of detecting anomalies. Therefore, feature selection is needed to choose a small number of the most relevant features for the purpose of anomaly detection.

Because the proposed feature selection needs to be unsupervised with respect to sensor data anomalies, but we still require to select meaningful features (columns) of feature matrix $\boldsymbol{W}_\phi$, we are going to use the target values y_p of adjacent windows $\vec{w}'_p$ for feature selection. The target values allow the feature selection process to be unsupervised with respect to anomalies (ground truth), because time series features, which are identified as regressors for the statistics of future sensor data, are expected to capture the characteristics of the intact sensor. Therefore, any anomalies will cause these features to have values that greatly differ from the values obtained from a normal sensor reading.

Our proposed feature selection starts with univariate hypothesis testing as provided by `tsfresh`, which selects a several statistically relevant features by calculating their p-values and controlling the false discovery rate by applying the Benjamini-Yekutieli procedure [44]. The hypothesis testing rules out features that are not statistically significant for predicting the selected statistics of future windows.

Because the set of statistically significant features still comprises about 100 characteristics, from which some features are expected to be collinear, a second feature selection step is needed. A robust approach [46] for this kind of problem is the combination of Recursive Feature Elimination (`RFE`) with a Random Forest Regressor (`RFR`), which narrows down the number of selected features from hundreds of features to a small user-defined number of features.

The number of features was chosen with a need for a small subset in mind. This allows us to reduce the complexity of the anomaly detection model, which is beneficial for the resource-constrained edge devices where we plan to implement our framework as part of the sensor network (Section III-D). It is seen in [46] that a small number of features is capable of modelling complex systems. As such, we chose to reduce the set of features to five for the sensor-specific case of our proposed unsupervised feature selection technique.

This additional feature selection step reduces the data volume needed to build the anomaly detection model. The resulting vector $\vec{v} \in \mathbb{R}^5$ of selected features from `RFE+RFR` is a subset of the original $Q = 794$ extracted features, i.e. $\vec{v}_p = [\hat{\phi}_1(\vec{w}_p), \dots, \hat{\phi}_5(\vec{w}_p)]$ and

$$\{\hat{\phi}_\kappa(\vec{w}_p)\}_{\kappa=1}^{5} \subset \{\phi_q(\vec{w}_p)\}_{q=1}^{794}.$$

Both feature selection processes (`tsfresh` and `RFR+RFE`) use the same target value vector $\vec{y}$.

2) Deployment-specific feature selection: In the sensor-specific feature selection, each sensor device has its own set of five selected features. However, with limited data, each individual sensor may not be able to come up with the most useful set of features. Therefore, a variant of the feature selection process that generates a uniform set of features for all sensors, is also developed. This means all sensor devices share a generic set of features, making the features generalizable across the sensor deployment. This may help cover sensor devices that lack data by using the collective effort of other sensor devices. This also allows us to use a bigger pool of information from all sensor devices with the intention of extracting better features that can be used for all sensor devices.

In a deployment-specific selection process, the count of the number of times a feature $\hat{\phi}$ has been selected for all M sensor devices is obtained, giving a set of selected features and their count $\#_{\hat{\phi}}$. The table is then sorted in descending order. Starting from the feature with the highest count, its covariance to the remaining features is calculated and any feature that has a correlation of more than 0.85 from the higher-ranked feature is removed. After removing any correlated features, up to ten non-collinear features with the highest $\#_{\hat{\phi}}$ are selected. The number of features is increased to up to ten to capture the variability across different sensor devices, as it takes into account all features selected from various sensor devices which may not be in close spatial proximity. This extended feature vector $\vec{v}'_p = [\hat{\phi}'_1(\vec{w}_p), \dots, \hat{\phi}'_{10}(\vec{w}_p)]$ covers a broader range of time series characteristics across the entire deployment.

C. Anomaly detection framework

The framework of the proposed automated anomaly detection via unsupervised feature selection is illustrated as a flowchart in Fig. 4, which starts with a raw time series $\vec{s}_m$ from sensor device m.

The raw time series $\vec{s}_m$ has N samples and is segmented into non-overlapping windows of length $K = 120$, which results in a matrix $\boldsymbol{Z} = [\vec{z}_1, \dots, \vec{z}_c, \dots, \vec{z}_C]^{\mathrm{T}} \in \mathbb{R}^{C \times K}$, where c is the index of the window in the segmented time series matrix and $C = \left\lceil \frac{N}{k} \right\rceil$ is the number of rows/windows after chunking. The chunking process is visualized in Fig. 5. The non-overlapping windows $\boldsymbol{Z}$ are split into train set $\boldsymbol{Z}_{train}$ and test set $\boldsymbol{Z}_{test}$. This is also visualized in Fig. 1, where the train set (blue) comprises only a short period (roughly three days) and the test set (green) consists of the remaining windows, which is data obtained during the deployment phase. Our proposed unsupervised feature selection approach only learns from the small training set of the calibration phase.

During the calibration phase, the train set, $\boldsymbol{Z}_{train}$ undergoes the proposed unsupervised feature selection process, either using the sensor-specific case (Section III-B.1) or deployment-specific case (Section III-B.2), which extracts and selects a small set of five or ten statistically significant features. Once the feature selection process has been completed, the selected feature vectors $\vec{v}_c$ are computed from $\boldsymbol{Z}_{train}$, which gives us the feature matrix $\boldsymbol{X}_{train}$. For example, using the sensor-specific version of feature selection, where each sensor device has its own set of five selected features, the feature matrix becomes

$$\boldsymbol{X}_{train} = \begin{bmatrix} \hat{\phi}_1(\vec{z}_1) & \dots & \hat{\phi}_\kappa(\vec{z}_1) & \dots & \hat{\phi}_5(\vec{z}_1) \\ \vdots & \vdots & \vdots & & \\ \hat{\phi}_1(\vec{z}_c) & \dots & \hat{\phi}_\kappa(\vec{z}_c) & \dots & \hat{\phi}_5(\vec{z}_c) \\ \vdots & \vdots & \vdots & & \\ \hat{\phi}_1(\vec{z}_C) & \dots & \hat{\phi}_\kappa(\vec{z}_C) & \dots & \hat{\phi}_5(\vec{z}_C) \end{bmatrix}.$$

The feature matrix $\boldsymbol{X}_{train}$ is standardized by subtracting every column $[\hat{\phi}_\kappa(\vec{z}_1), \dots, \hat{\phi}_\kappa(\vec{z}_C)]$ by its empirical mean μ_κ and dividing the result by the empirical standard deviation σ_κ of the column respectively time series feature. Both, the mean μ_κ and the standard deviation σ_κ are stored as parameter set $h(X)$, which is subsequently used to transform the time series features in the deployment phase.

The standardized feature matrix $\bar{\boldsymbol{X}}_{train}$ is the input to the PCA model. PCA is a technique that can be used to find patterns in the data or the correlation between variables by computing the eigenvectors of the covariance matrix [47, Chap 11]. These eigenvectors or principal components are orthogonal to each other. Therefore, other than being used as a feature reduction technique, PCA can also be used for fault detection [6], [13]. A PCA model $g(\boldsymbol{X}|\boldsymbol{X}_{train})$ with 2 principal components is fitted and used to transform the matrix of standardized time series features, which returns the matrix $\hat{\boldsymbol{X}}_{train}$. The fitted PCA model $g(\boldsymbol{X}|\boldsymbol{X}_{train})$ is used in the deployment phase to project the incoming data into the principal component subspace.

In the test respectively deployment phase, only the five or ten selected time series features $\vec{v}$ are extracted from the test set $\boldsymbol{Z}_{test}$. This gives the matrix $\boldsymbol{X}_{test}$, which is standardized by applying $h(\boldsymbol{X}_{test}|\boldsymbol{X}_{train})$. The standardized feature matrix $\bar{\boldsymbol{X}}_{test}$ is projected into the principal component subspace using the fitted PCA model $g(\bar{\boldsymbol{X}}_{test}|\boldsymbol{X}_{train})$ giving $\hat{\boldsymbol{X}}_{test} \in \mathbb{R}^2$.

A visualization of the feature values in PCA subspace can be seen in Fig. 6. Fig. 6(a) shows the train and test data projected on the PCA subspace where most of the normal, train data (blue) can be seen to be clustered around the origin, whereas the test data (green) is more spread out from the origin. In Fig. 6(b), the red crosses show the anomalous data, which are seen to be further away from the origin compared to the normal, error-free data. The automated anomaly detection is done by computing the Euclidean distance of each data point to the origin of the PCA subspace. Distances beyond a certain distance threshold α are identified as an anomaly. This process is summarised by a vector of labels $\vec{r} = [r_1, \dots, r_c \dots r_C]$, where $r_c \in \{0, 1\}$ encodes the two distinct classes of normal (0) and abnormal (1) sensor data for the respective windows. Thus, each window is automatically labelled as having normal data if the Euclidean distance to the origin of the PCA

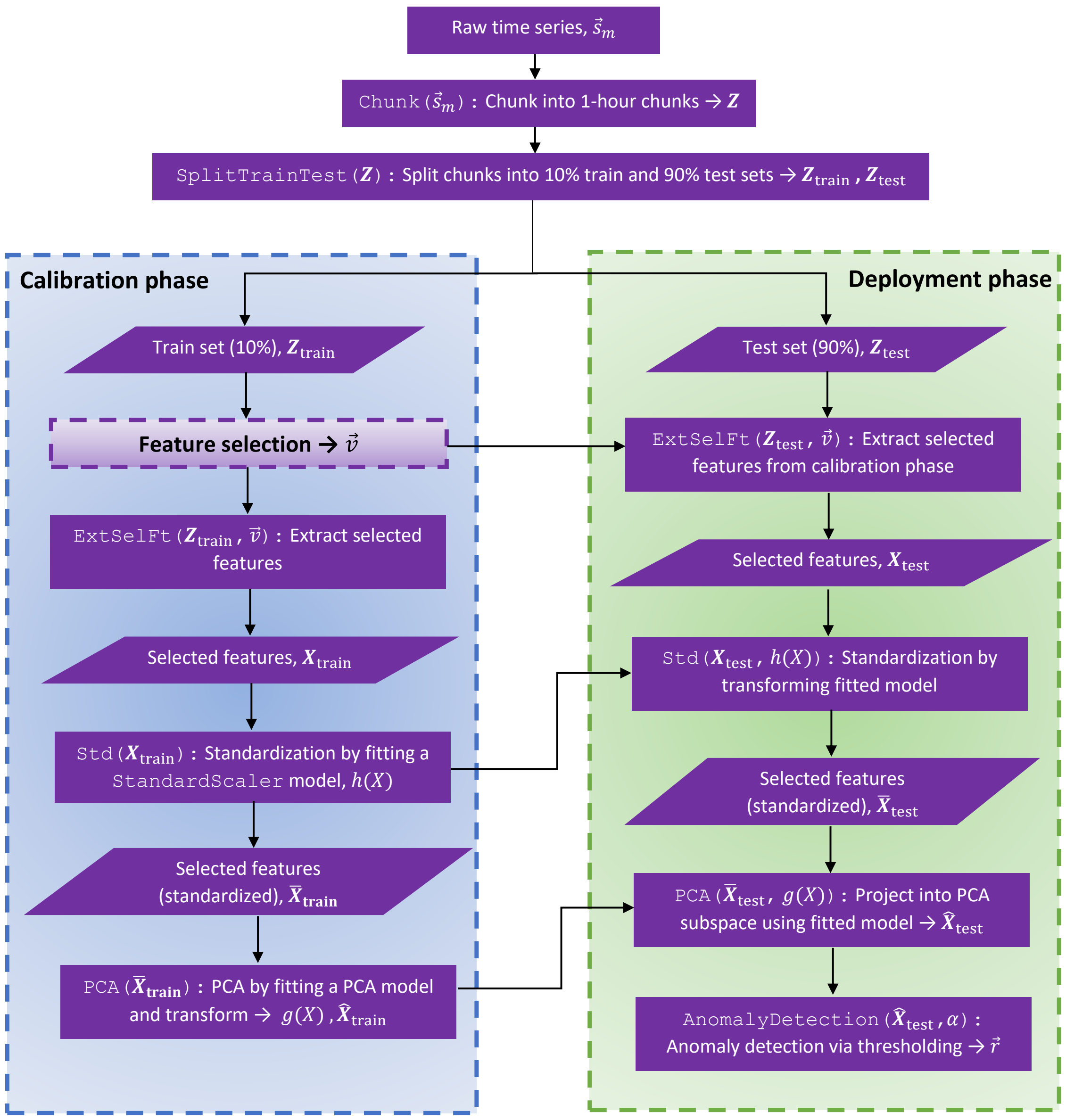

Raw time series, $\vec{s}_m$
Chunk($\vec{s}_m$): Chunk into 1-hour chunks → $\boldsymbol{Z}$
SplitTrainTest($\boldsymbol{Z}$): Split chunks into 10% train and 90% test sets → $\boldsymbol{Z}_{\text{train}}$, $\boldsymbol{Z}_{\text{test}}$
Calibration phase
Train set (10%), $\boldsymbol{Z}_{\text{train}}$
Feature selection → $\vec{v}$
ExtSelFt($\boldsymbol{Z}_{\text{train}}$, $\vec{v}$): Extract selected features
Selected features, $\boldsymbol{X}_{\text{train}}$
Std($\boldsymbol{X}_{\text{train}}$): Standardization by fitting a StandardScaler model, $h(X)$
Selected features (standardized), $\bar{\boldsymbol{X}}_{\text{train}}$
PCA($\bar{\boldsymbol{X}}_{\text{train}}$): PCA by fitting a PCA model and transform → $g(X)$, $\hat{\boldsymbol{X}}_{\text{train}}$
Deployment phase
Test set (90%), $\boldsymbol{Z}_{\text{test}}$
ExtSelFt($\boldsymbol{Z}_{\text{test}}$, $\vec{v}$): Extract selected features from calibration phase
Selected features, $\boldsymbol{X}_{\text{test}}$
Std($\boldsymbol{X}_{\text{test}}$, $h(X)$): Standardization by transforming fitted model
Selected features (standardized), $\bar{\boldsymbol{X}}_{\text{test}}$
PCA($\bar{\boldsymbol{X}}_{\text{test}}$, $g(X)$): Project into PCA subspace using fitted model → $\hat{\boldsymbol{X}}_{\text{test}}$
AnomalyDetection($\hat{\boldsymbol{X}}_{\text{test}}$, α): Anomaly detection via thresholding → $\vec{r}$

Fig. 4: Workflow of the proposed automated anomaly detection using unsupervised feature selection with two phases: calibration phase (blue) and deployment phase (green). The raw time series, $\vec{s}_m$ firstly undergo a segmentation process and the non-overlapping windows, $\boldsymbol{Z}$ are then split into train and test sets. Using the small train set, $\boldsymbol{Z}_{train}$ the feature selection process (Section III-B) is performed which returns a subset of five or ten meaningful features, $\vec{v}$. The selected features extracted returns $\boldsymbol{X}_{train}$ and is subsequently standardized, giving $\bar{\boldsymbol{X}}_{train}$. It is used as input for a PCA model, $g(X)$, returning the two principal components of each non-overlapping window, $\hat{\boldsymbol{X}}_{train}$. In the deployment phase, the same five or ten selected features are extracted and standardized from the test set, $\boldsymbol{Z}_{test}$ which the resulting standardized selected features, $\bar{\boldsymbol{X}}_{test}$ is then projected into the principal component subspace using $g(X)$. Anomaly detection is then done via thresholding where the Euclidean distance of each point to the origin of the PCA subspace is calculated and compared against a threshold value, α. This returns two distinct labels where each non-overlapping window is labelled as either normal or anomaly, $\vec{r}$.

subspace is smaller than the threshold and conversely, as an anomaly if the Euclidean distance to the origin of the PCA subspace is larger than the threshold.

D. Sensor network architecture

Fig. 7 shows the high-level structure of a typical IoT sensor network, which consists of three layers, the central or *Cloud Layer*, the network or *Edge Layer* and the perception or *Sensor Layer*. Firstly, a physical sensor device in the *Sensor Layer* has sensors that measure environmental variables such as temperature, humidity, light and wind which produce readings at a fixed sampling rate, e.g. 30 seconds.

These readings are then transmitted to their local edge device in the *Edge Layer*. The edge device allows for real-time processing of data, where it also has a connection to the Internet. Here, it is critical to reduce the load to be sent to the central server as we do not want to burden the network with useless data, especially if they are erroneous as it is wasteful in terms of bandwidth and storage space.

Therefore, our focus is on the *Edge Layer*, where anomaly detection should be done in real-time to notify or send alerts as soon as an anomaly is found as it might suggest that the sensor device is faulty and requires a maintenance check. Our proposed framework, denoted by the dotted purple lines in

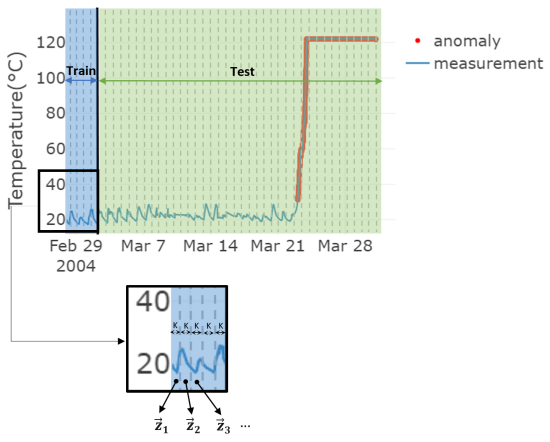
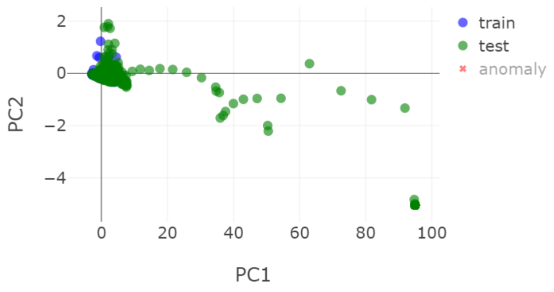
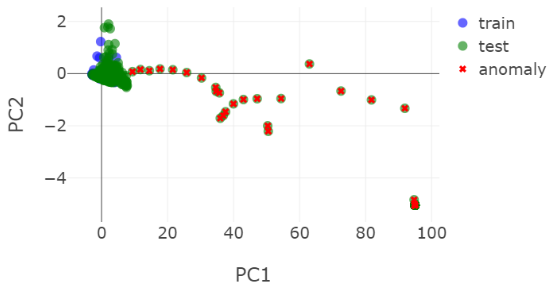


Fig. 5**:** Chunking process of the raw time series, where it is segmented into one-hour non-overlapping windows where every window has length, $K = 120$ samples, i.e. $\texttt{len}(\vec{z}_1) = \texttt{len}(\vec{z}_2) = \texttt{len}(\vec{z}_3) = 120$.



(a) PCA without labelled anomalies.



(b) PCA with labelled anomalies.

Fig. 6**:** (a) Example of PCA projection of train (blue) and test (green) data. Most of the train data (normal behaviour) is centered around the origin as opposed to the test data which spreads further away from the origin. (b) PCA projection with superimposed anomalies. With the anomalies superimposed, it can be seen that the anomalous data are further away from the origin than the normal data.

the figure, will be deployed in each edge device to perform anomaly detection. Furthermore, each edge device has spatial proximity and a relationship with the sensor devices that are connected to it, which allows for more precise error detection as different neighbourhoods of sensors might have different normal behaviours. Lastly, detecting anomalies in this layer would reduce unnecessary data communication to the cloud.

Finally, the *Cloud Layer* is where the cloud server resides. The sensor data is stored here where any data warehousing and big data processing including deep learning is carried out.

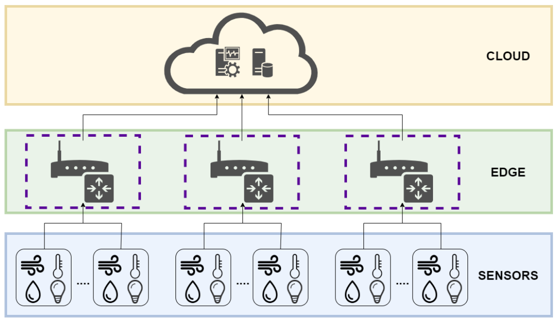


Fig. 7**:** Architecture diagram of a wireless IoT sensor network where the proposed framework is applied in the edge computing layer.

IV. Experiments and Datasets

Experiments were done in order to evaluate the proposed automated anomaly detection framework via unsupervised feature selection. The following subsections detail the setup of the experiments and the description of the publicly available datasets along with the preparation of the data (e.g. data labelling) to be used in the experiments to validate the proposed framework.

A. Experimental setup

All experiments are implemented and executed on a single computer with a CPU of 2.2GHz and a 16GB memory and the project was planned and conducted following closely to the steps introduced in [48]. The experiments ran on Python version 3.7 and its respective libraries, which are briefly described in the following: Python Data Analysis Library, better known as `pandas` is an open-sourced library that provides data analysis tools and converts data (e.g. CSV files) into easy-to-use Python objects called DataFrames, which consists of rows and columns. It is also used in conjunction with `NumPy`, which is another common Python library supporting mathematical and logical operations on arrays and matrices. The plots obtained from the data analysis are created using `plotly` [49], an open-sourced browser-based Python graphing library, which allows interactive graph visualizations.

The machine learning models in particular such as `RFE` and `RFR`, as well as `PCA` and clustering, were implemented using `scikit-learn` [50], a free, open-source machine learning Python library, which provides tools for various unsupervised as well as supervised (classification and regression) data mining algorithms. Moreover, as mentioned in Section III-B, a Python package named Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests, or in short `tsfresh` [41] is used to carry out the systematic feature extraction of a time series. This package automates and speeds up the feature extraction process by computing a comprehensive total of 794 univariate time series features by default. It is also used for the unsupervised feature selection process along with `RFE` and `RFR`.

B. Dataset preparation

There are two publicly available datasets used in this paper, namely the Intel Berkeley Research Lab (IBRL) dataset [45] and Lausanne Urban Canopy Experiment (LUCE) dataset, a WSN deployment under the Sensorscope project [51], [52]. They are introduced in more detail in [6], but the following are a brief description of the two datasets.

- IBRL – Obtained from an indoor WSN deployed in the Intel Berkeley Research Lab, for about a month between February 28th to April 5th 2004. It has 54 sensors deployed indoors in the lab itself, collecting data about the environment such as the temperature, humidity, and light around the lab. It also consists of the date, time, epoch, sensor/mote ID, and voltage values. The dataset consists of 2.3 million readings, obtained every 30 seconds from the 54 sensors.
- LUCE – Obtained from an outdoor WSN deployment, which is used to study heat flux in urban environments [53]. The WSN was deployed all around the École Polytechnique Fédérale de Lausanne (EFPL) campus via the Sensorscope project. About 44.4 million readings were collected every 30 seconds for around a year, from July 2006 and May 2007, using about 97 unique sensor nodes. The sensor devices measure environmental variables such as the ambient temperature, surface temperature, relative humidity, solar radiation, soil moisture, watermark, rain meter, wind speed, and wind direction. Along with that are also metadata such as the datetime (in year, month, day, hour, minute and second), epoch and sequence number.

Since the two datasets do not contain any ground-truth for anomaly labels, the data points are labelled per window, using a minority labelling approach. This means that if a window contains an anomalous data point, the whole window is considered an anomaly. These anomalies are labelled according to heuristics and are not artificially imputed. Moreover, these heuristics are not used as part of our framework but only for labelling the used datasets for evaluation purposes. In order to determine if a reading is anomalous, we used an automated labelling approach [31]. A data point x_t is labelled as an anomaly if it satisfies either one of three heuristics, which defines an outlier, a spike and a constant value anomaly respectively [6]:

1) The data point x_t is more than 3 standard deviations away from the mean,
$$x_t \leq 3\sigma - \mu, \; x_t \geq 3\sigma + \mu. \tag{3}$$
2) The difference between x_t and its neighbouring readings is larger than 3 units,
$$|x_t - x_{t-1}| > 3, \; |x_{t+1} - x_t| > 3. \tag{4}$$
3) The readings are constant for the next 1 hour,
$$x_t = x_{t+1} = \cdots = x_{t+120}. \tag{5}$$

For evaluation purposes, a total of 52 sensor devices from IBRL and 15 sensor devices from LUCE, which contain at least one or more anomalies were considered. The data were cleaned and sanitized before being split into train and test sets because some sensors had missing data. For example, since the sampling rate is every 30 seconds, each window should have 120 samples. However, due to missing data, not all windows were of equal length and contained 120 measurements. To ensure that each window has the same number of samples, the data in each window were upsampled and interpolated if the window had less than 120 samples.

V. Results and discussion

Our proposed automated anomaly detection framework with unsupervised feature selection is evaluated against the raw PCA version, Raw, where anomaly detection is done on the PCA model without any feature extraction or selection. The evaluation also includes a version combining PCA and feature extraction, but without any feature selection, i.e. PCA with all 794 features, denoted by FE. We have evaluated our proposed unsupervised feature selection approach with Raw and FE to validate the importance of feature selection. Our proposed automated anomaly detection model with the sensor-specific unsupervised feature selection is represented as FS_{mean} and FS_{std} for the two different target statistics (the mean and standard deviation of the next 20 samples of the current window respectively). Moreover, the deployment-specific version of our proposed unsupervised feature selection method is denoted by $FS_{mean}{}^*$ and $FS_{std}{}^*$.

The performance metric used in this paper is adapted from Matthew's Correlation Coefficient (MCC),

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

which is a robust metric for imbalanced classification problems. MCC takes into account all four quadrants of the confusion matrix and the ratio of their sizes. It has a score that ranges from -1 to 1, where 0 indicates a by chance result, which could be by simply guessing that there are no anomalies, whereas 1 indicates a perfect agreement between the predictions and actual labels and -1 indicates perfect disagreement between the predictions and actual labels [54]. The average MCC value, $AvgMCC$, is calculated by adding the MCC value per sensor, MCC_m divided by M, where $m = 1, .., M$ and M is the number of sensor devices. The experiment is then repeated for each threshold, α, starting at $\alpha = 0$ to $\alpha = 2000$.

Fig. 8 shows the comparison of the results of all six anomaly detection versions obtained from the experiment using the IBRL datasets. The graph shows the average MCC value over all 52 sensor devices for each threshold for the six anomaly detection versions. Table II details the results of all versions, along with their characteristics, i.e. whether it has feature selection, the feature selection target value (mean or standard deviation) and the feature selection type (sensor- or deployment-specific). Their best average MCC is also tabulated at their respective thresholds α and total amount Σ of training data required for PCA. It can be seen that almost all the proposed automated anomaly detection approaches with unsupervised feature selection perform relatively well for thresholds α ranging from 5 to 25. The two versions, which performs the best is the Raw version without any feature extraction and selection and our proposed anomaly detection

Version	Feature selection (FS)	FS target value	FS type	α	Σ	Best $AvgMCC$
Raw	None (raw data)	–	–	60	429,240	0.978
FE	None (all 794 features)	–	–	120	2,840,138	0.936
FS_{mean}	Yes	Mean	Sensor	15	17,885	0.967
FS_{std}	Yes	Standard deviation	Sensor	10	17,885	0.651
$FS_{mean}{}^{*}$	Yes	Mean	Deployment	5	35,770	0.882
$FS_{std}{}^{*}$	Yes	Standard deviation	Deployment	25	35,770	0.941

TABLE II**:** The best average MCC result across all 52 sensors in the IBRL dataset for the different versions, along with their characteristics (feature selection, feature selection target value, feature selection type) at their respective thresholds, α and total amount of training data required for PCA, Σ. The two best-performing versions with the highest $AvgMCC$ is the Raw version without any feature engineering and our proposed sensor-specific feature selection approach with mean as the target statistic FS_{mean}. They are both comparable with their best $AvgMCC$ values of 0.978 and 0.967 respectively. However, our proposed approach has the added benefit of requiring lesser training data for PCA (17,885 vs. 429,240).

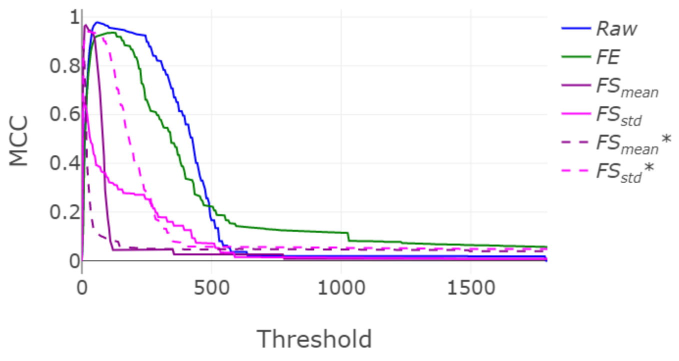


Fig. 8**:** Results of the average MCC value for a range of threshold, α from 0 to 2000 across all 52 sensors of the IBRL dataset for each anomaly detection version with or without feature selection.

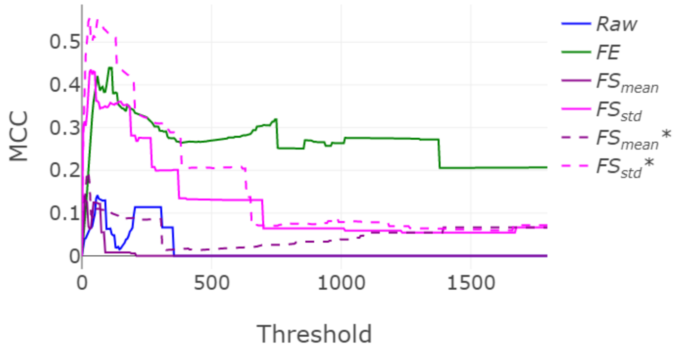


Fig. 9**:** Results of the average MCC value for a range of threshold from 0 to 2000 across all 15 sensors of the LUCE dataset for each anomaly detection version with or without feature selection.

framework with sensor-specific unsupervised feature selection with mean as the target statistic, FS_{mean}, where their best $AvgMCC$ values are both comparable to each other, at 0.978 and 0.967 respectively. On top of that, the proposed approach has an added benefit of performing anomaly detection using only a small amount of data, Σ i.e. 24 times less compared to the raw PCA case where the entire data is required for training.

The proposed framework is further evaluated on the LUCE outdoor environment monitoring dataset to validate the results obtained for the IBRL indoor environment monitoring dataset. Fig. 9 shows the comparison of the average MCC values for each anomaly detection version for different thresholds on the LUCE dataset and Table III details the different anomaly detection versions (and their characteristics) along with the best average MCC values at their respective thresholds α and total number Σ of training data needed for PCA for the LUCE dataset. Here, the performance of the proposed feature selection approach is evidently better than the other versions, especially the Raw case which performs rather poorly, where the maximum average MCC achieved is barely above 0.1. The deployment-specific unsupervised feature selection $FS_{std}{}^{*}$, which selects a set of ten non-collinear features by using the standard deviation of the near-future values as the target statistic, has the highest average MCC compared to the other versions, at a maximum $AvgMCC$ of 0.557 for $\alpha = 30$.

The Raw version performed badly for the LUCE dataset, which suggests that using only the raw data without any feature extraction or selection is not sufficient to reliably detect anomalies. This seems to be the case for this outdoor monitoring dataset as it contains a wide range of time series patterns due to its heterogeneous nature. The sensor-specific versions of our proposed approach also do not perform as good, suggesting that more features are needed to cover a broader range of sensor characteristics across the entire deployment. The collective effort of data obtained from multiple sensor devices helped extract a better set of features that can be used for anomaly detection across all sensor devices.

From the experiments, it can be seen that the proposed anomaly detection framework with unsupervised feature selection performs at least as good as or significantly better than the version without any feature selection. It is also more generalizable and consistent across both indoor and outdoor IoT application datasets, where it performs consistently well for both datasets whereas using only Raw without any feature selection fails in the outdoor LUCE dataset. Furthermore, the thresholds found to be the best for these two applications are somewhere around the range of 15 to 30, though as with any hyperparameter, it needs to be tuned to each application as it is problem-specific.

Fig. 10 shows the line plot of the average MCC values for both IBRL and LUCE datasets with respect to the number of features, κ used in the $FS_{std}{}^{*}$ version. The shaded area represents the 95% confidence interval, which indicates the probability of observing a value outside of the shaded area to be less than 5%. The reason for the small number, κ

Version	Feature selection (FS)	FS target value	FS type	α	Σ	Best $AvgMCC$
Raw	None (raw data)	–	–	60	129,600	0.142
FE	None (all 794 features)	–	–	115	857,520	0.441
FS_{mean}	Yes	Mean	Sensor	15	5,400	0.130
FS_{std}	Yes	Standard deviation	Sensor	35	5,400	0.435
$FS_{mean}{}^*$	Yes	Mean	Deployment	20	10,800	0.198
$FS_{std}{}^*$	Yes	Standard deviation	Deployment	30	10,800	0.557

TABLE III**:** The best average MCC result across all 15 sensors in the LUCE dataset for the different versions, along with their characteristics (feature selection, feature selection target value, feature selection type) at their respective thresholds, α and the total number of training data required for PCA, Σ. The deployment-specific version of the proposed feature selection method with standard deviation as the target statistic, $FS_{std}{}^*$ performs the best and significantly better than the other versions, with the best average MCC of 0.557. It also uses less data than the raw version at a factor of $\frac{1}{12}$.

Feature	`tsfresh` algorithm	Parameters
Maximum	`maximum`	None
Median	`median`	None
Minimum	`minimum`	None
Quantile	`quantile`	`q=0.9`
Quantile	`quantile`	`q=0.8`

TABLE IV**:** The five time series features selected using the proposed sensor-specific unsupervised feature selection approach for detecting anomalies in Sensor Device 1 of the IBRL dataset.

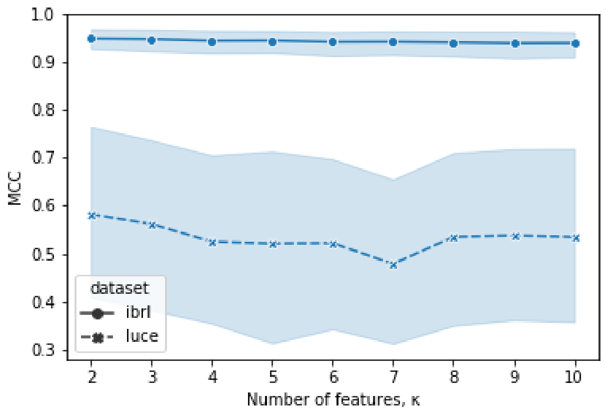


Fig. 10**:** Results of the average MCC value for different numbers of features, κ for IBRL and LUCE datasets using $FS_{std}{}^*$, where the shaded area represents the 95% confidence interval.

is explained in Section III-B.1 and Section III-B.2. It can be seen that the MCC does not vary significantly given the number of features. Presumably, this is caused by the additional dimensionality reduction of the superimposed PCA model.

Features selected for the two datasets are presented in Table IV for Sensor 1 of the IBRL dataset using the proposed sensor-specific unsupervised feature selection approach and Table V for the LUCE dataset using the proposed deployment-specific unsupervised feature selection approach. Some features selected automatically by the proposed framework, such as `maximum`, `minimum`, `median` and `agg_linear_trend` (linear least-squares regression) are similar to common features selected manually by domain experts as seen in related works. This shows that our approach is able to pick up these features automatically without the need for manual input or expert knowledge. It also selected some more sophisticated features which are useful in detecting anomalies in sensor time series which are not found in related works. These more complex features are `change_quantiles`, which calculates the mean absolute value of consecutive changes of the time series in a specified window, `spkt_welch_density`, which estimates the cross power spectral density of the time series at different frequencies, `time_reversal_asymmetry_statistic`, a feature proposed in [55] for time series classification, and `fft_coefficient`, which calculates the Fourier coefficients of the one-dimensional discrete Fourier Transform. Refer to [46] for a more detailed description of `agg_linear_trend` and `change_quantiles` and to [56] for a comprehensive overview of all `tsfresh` features.

VI. Conclusion

Overall, our proposed unsupervised feature selection method of selecting time series features, which are meaningful in predicting statistics (e.g. mean and standard deviation) of near future values is a crucial step for reliable anomaly detection in time series data. The proposed automated time series anomaly detection framework with unsupervised feature selection is also shown to be comparable or more accurate than directly applying anomaly detection techniques on the raw time series with the added benefit of having lesser data to communicate and compute. By determining a small set of common features that reliably detects anomalies in the calibration phase, our proposed methods help reduce the communication costs in IoT applications, where anomaly detection can be done at the edge/sensor devices themselves, without needing to send the massive amount of data to the central processing centres. On top of that, our proposed framework only requires a short calibration phase, up to three days to reliably detect anomalies in time series data.

By evaluating on the two publicly available datasets, IBRL and LUCE, our proposed unsupervised feature selection framework, which uses the mean and standard deviation of the near future values as the target statistics for feature selection, is seen to perform effectively in detecting anomalies for sensor time series. Future work includes implementing an online feature learning in the deployment phase for real-time scenarios so that the features can be updated when more data is obtained. This would lead to a more robust and accurate anomaly detection model that takes into account, for example, seasonal changes, automatically. Moreover, dynamic thresholding is required as different sensor devices have a different PCA subspace, thus requiring different thresholds for different sensor devices. Other factors such as finding the optimal number of features and comparing with other machine learning models are also prospects of future research.

Feature	`tsfresh` algorithm	Parameters
Aggregated Linear Trend	`agg_linear_trend`	`f_agg="var", chunk_len=10, attr="stderr"`
Aggregated Linear Trend	`agg_linear_trend`	`f_agg="var", chunk_len=5, attr="slope"`
Aggregated Linear Trend	`agg_linear_trend`	`f_agg="var", chunk_len=10, attr="rvalue"`
Conditional Dynamics	`change_quantiles`	`f_agg="mean", isabs=True, qh=1.0, ql=0.6`
Conditional Dynamics	`change_quantiles`	`f_agg="mean", isabs=True, qh=0.8, ql=0.0`
Conditional Dynamics	`change_quantiles`	`f_agg="mean", isabs=True, qh=0.8, ql=0.4`
Conditional Dynamics	`change_quantiles`	`f_agg="mean", isabs=False, qh=0.8, ql=0.2`
Conditional Dynamics	`change_quantiles`	`f_agg="var", isabs=True, qh=1.0, ql=0.8`
Time Reversal Asymmetry	`time_reversal_asymmetry`	`lag=3`
Power Spectral Density	`spkt_welch_density`	`coeff=8`

TABLE V**:** The ten time series features selected using the proposed deployment-specific unsupervised feature selection approach for detecting anomalies in the LUCE dataset.

References

[1] Cisco, "Cisco annual internet report (2018–2023)," Whitepaper c11-741490, Cisco Systems Inc., San Jose, CA, 2020.

[2] H. Zhang, J. Liu, and A.-C. Pang, "A Bayesian network model for data losses and faults in medical body sensor networks," *Computer Networks*, vol. 143, pp. 166 – 175, 2018.

[3] Y. Hu, H. Chen, G. Li, H. Li, R. Xu, and J. Li, "A statistical training data cleaning strategy for the PCA-based chiller sensor fault detection, diagnosis and data reconstruction method," *Energy and Buildings*, vol. 112, pp. 270 – 278, 2016.

[4] J. Ye, G. Stevenson, and S. Dobson, "Detecting abnormal events on binary sensors in smart home environments," *Pervasive and Mobile Computing*, vol. 33, pp. 32 – 49, 2016.

[5] T. Chakraborty, A. U. Nambi, R. Chandra, R. Sharma, M. Swaminathan, Z. Kapetanovic, and J. Appavoo, "Fall-curve: A novel primitive for IoT Fault Detection and Isolation," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems - SenSys '18*, (Shenzhen, China), pp. 95–107, ACM Press, 2018.

[6] H. Y. Teh, A. W. Kempa-Liehr, and K. I.-K. Wang, "Sensor data quality: a systematic review," *Journal of Big Data*, vol. 7, p. 11, Feb. 2020.

[7] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier Analysis* (C. C. Aggarwal, ed.), pp. 1–34, Springer International Publishing, 2017.

[8] M. A. Rassam, M. A. Maarof, and A. Zainal, "Adaptive and online data anomaly detection for wireless sensor systems," *Knowledge-Based Systems*, vol. 60, pp. 44 – 57, 2014.

[9] M. Gupta, J. Gao, C. Aggarwal, and J. Han, "Outlier Detection for Temporal Data," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 5, pp. 1–129, Mar. 2014.

[10] A. Barde and S. Jain, "A Survey of Multi-Sensor Data Fusion in Wireless Sensor Networks," SSRN Scholarly Paper ID 3167286, Social Science Research Network, Rochester, NY, Apr. 2018.

[11] Y. Li, J. Chen, and L. Feng, "Dealing with Uncertainty: A Survey of Theories and Practices," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 2463–2482, Nov. 2013.

[12] B. Prathiba, K. J. Sankar, and V. Sumalatha, "Enhancing the data quality in wireless sensor networks - a review," in *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, pp. 448–454, Sep. 2016.

[13] R. Dunia, S. Joe Qin, T. F. Edgar, and T. J. McAvoy, "Use of principal component analysis for sensor fault identification," *Computers & Chemical Engineering*, vol. 20, pp. S713–S718, Jan. 1996.

[14] M. F. Harkat, G. Mourot, and J. Ragot, "Sensor Failure Detection of Air Quality Monitoring Network," *IFAC Proceedings Volumes*, vol. 33, no. 11, pp. 529 – 534, 2000.

[15] A. Alawi, S. W. Choi, E. Martin, and J. Morris, "Sensor Fault Identification Using Weighted Combined Contribution Plots," in *Fault Detection, Supervision and Safety of Technical Processes 2006* (H.-Y. Zhang, ed.), pp. 908 – 913, 2007.

[16] R. Sharifi and R. Langari, "Nonlinear sensor fault diagnosis using mixture of probabilistic PCA models," *Mechanical Systems and Signal Processing*, vol. 85, pp. 638 – 650, 2017.

[17] M. Mansouri, M.-F. Harkat, M. Nounou, and H. Nounou, "Midpoint-radii principal component analysis -based EWMA and application to air quality monitoring network," *Chemometrics and Intelligent Laboratory Systems*, vol. 175, pp. 55 – 64, 2018.

[18] C. Zhao and Y. Fu, "Statistical analysis based online sensor failure detection for continuous glucose monitoring in type I diabetes," *Chemometrics and Intelligent Laboratory Systems*, vol. 144, pp. 128 – 137, 2015.

[19] A. Fawzy, H. M. O. Mokhtar, and O. Hegazy, "Outliers detection and classification in wireless sensor networks," *Egyptian Informatics Journal*, vol. 14, no. 2, pp. 157 – 164, 2013.

[20] H. Liu, J. Chen, F. Huang, and H. Li, "An Electric Power Sensor Data Oriented Data Cleaning Solution," in *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks 2017 11th International Conference on Frontier of Computer Science and Technology 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*, pp. 430–435, June 2017.

[21] Y. Zhou, W. Hu, Y. Min, L. Zheng, B. Liu, R. Yu, and Y. Dong, "A semi-supervised anomaly detection method for wind farm power data preprocessing," in *2017 IEEE Power Energy Society General Meeting*, pp. 1–5, July 2017. ISSN: 1944-9933.

[22] H. Saeedi Emadi and S. M. Mazinani, "A Novel Anomaly Detection Algorithm Using DBSCAN and SVM in Wireless Sensor Networks," *Wireless Personal Communications*, vol. 98, pp. 2025–2035, Jan. 2018.

[23] W. Zhang, X. Dong, H. Li, J. Xu, and D. Wang, "Unsupervised Detection of Abnormal Electricity Consumption Behavior Based on Feature Engineering," *IEEE Access*, vol. 8, pp. 55483–55500, 2020.

[24] K. Smarsly and K. H. Law, "Decentralized fault detection and isolation in wireless structural health monitoring systems using analytical redundancy," *Advances in Engineering Software*, vol. 73, pp. 1 – 10, 2014.

[25] G. Jäger, S. Zug, T. Brade, A. Dietrich, C. Steup, C. Moewes, and A. M. Cretu, "Assessing neural networks for sensor fault detection," in *2014 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 70–75, May 2014.

[26] H. H. Bosman, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta, "Spatial anomaly detection in sensor networks using neighborhood information," *Information Fusion*, vol. 33, pp. 41 – 56, 2017.

[27] P. Wang, R. X. Gao, X. Tang, and Z. Fan, "Sensing Uncertainty Evaluation for Product Quality," *Procedia CIRP*, vol. 41, pp. 706 – 711, 2016.

[28] H. Xiao, D. Huang, Y. Pan, Y. Liu, and K. Song, "Fault diagnosis and prognosis of wastewater processes with incomplete data by the auto-associative neural networks and ARMA model," *Chemometrics and Intelligent Laboratory Systems*, vol. 161, pp. 96 – 107, 2017.

[29] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134 – 147, 2017.

[30] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. S. Rossi, "Sensor-Fault Detection, Isolation and Accommodation for Digital Twins via Modular Data-Driven Architecture," *IEEE Sensors Journal*, vol. 21, pp. 4827–4838, Feb. 2021. Conference Name: IEEE Sensors Journal.

[31] H. H. W. J. Bosman, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta, "Ensembles of incremental learners to detect anomalies in ad hoc sensor networks," *Ad Hoc Networks*, vol. 35, pp. 14 – 36, 2015.

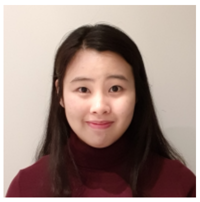
[32] D.-I. Curiac and C. Volosencu, "Ensemble based sensing anomaly detection in wireless sensor networks," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9087 – 9096, 2012.

[33] G. R. Abuaitah and B. Wang, "Data-centric anomalies in sensor network deployments: analysis and detection," in *2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012)*, vol. Supplement, pp. 1–6, Oct. 2012.

[34] A. Rahman, D. V. Smith, and G. Timms, "A Novel Machine Learning Approach Toward Quality Assessment of Sensor Data," *IEEE Sensors Journal*, vol. 14, pp. 1035–1047, Apr. 2014.

[35] Y. Zhang, N. Meratnia, and P. Havinga, "Adaptive and Online One-Class Support Vector Machine-Based Outlier Detection Techniques for Wireless Sensor Networks," in *2009 International Conference on Advanced

Information Networking and Applications Workshops, pp. 990–995, May 2009.
[36] Y. Zhang, N. Meratnia, and P. J. M. Havinga, "Distributed online outlier detection in wireless sensor networks using ellipsoidal support vector machine," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1062 – 1074, 2013.
[37] S. Zidi, T. Moulahi, and B. Alaya, "Fault Detection in Wireless Sensor Networks Through SVM Classifier," *IEEE Sensors Journal*, vol. 18, pp. 340–347, Jan. 2018. Conference Name: IEEE Sensors Journal.
[38] A. Yazidi and E. Herrera-Viedma, "A new methodology for identifying unreliable sensors in data fusion," *Knowledge-Based Systems*, vol. 136, pp. 85–96, Nov. 2017.
[39] S. Russo, M. Lürig, W. Hao, B. Matthews, and K. Villez, "Active learning for anomaly detection in environmental data," *Environmental Modelling & Software*, vol. 134, p. 104869, Dec. 2020.
[40] D. Zang, J. Liu, and H. Wang, "Markov chain-based feature extraction for anomaly detection in time series and its industrial application," in *2018 Chinese Control And Decision Conference (CCDC)*, pp. 1059–1063, June 2018. ISSN: 1948-9447.
[41] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series FeatuRe extraction on basis of scalable hypothesis tests (tsfresh – a Python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.
[42] Z. Ouyang, X. Sun, and D. Yue, "Hierarchical Time Series Feature Extraction for Power Consumption Anomaly Detection," in *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration* (K. Li, Y. Xue, S. Cui, Q. Niu, Z. Yang, and P. Luk, eds.), Communications in Computer and Information Science, (Singapore), pp. 267–275, Springer, 2017.
[43] G. Liu, L. Li, L. Zhang, Q. Li, and S. S. Law, "Sensor faults classification for SHM systems using deep learning-based method with Tsfresh features," *Smart Materials and Structures*, vol. 29, p. 075005, July 2020.
[44] Y. Benjamini and D. Yekutieli, "The Control of the False Discovery Rate in Multiple Testing under Dependency," *The Annals of Statistics*, vol. 29, no. 4, pp. 1165–1188, 2001. Publisher: Institute of Mathematical Statistics.
[45] S. Madden, "Intel lab data." `http://db.csail.mit.edu/labdata/labdata.html`, 2004.
[46] A. Kennedy, G. Nash, N. Rattenbury, and A. W. Kempa-Liehr, "Modelling the projected separation of microlensing events using systematic time-series feature engineering," *Astronomy and Computing*, vol. 35, no. 100460, pp. 1–14, 2021.
[47] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes. The Art of Scientific Computing. Second Edition*. Cambridge: Cambridge University Press, 3rd ed., 2007.
[48] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. Sebastopol, CA, United States: O'Reilly Media, 4th release ed., 2017.
[49] P. T. Inc., "Collaborative data science," 2015.
[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[51] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Trans. Sens. Network*, vol. 6, 01 2010.
[52] G. Barrenetxea, "Sensorscope: Sensor networks for environmental monitoring," 2018.
[53] D. F. Nadeau, W. Brutsaert, M. B. Parlange, E. Bou-Zeid, G. Barrenetxea, O. Couach, M.-O. Boldi, J. S. Selker, and M. Vetterli, "Estimation of urban sensible heat flux using a dense wireless network of observations," *Environmental Fluid Mechanics*, vol. 9, pp. 635–653, Dec. 2009.
[54] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, dec 2017.
[55] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," May 2014.
[56] M. Christ, N. Braun, and J. Neuffer, "Overview on extracted features." `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html`, 2016-2021.



Hui Yie Teh is a PhD student in the Department of Electrical, Computer and Software Engineering at The University of Auckland. She graduated with a BTech (Hons) degree back in 2017 in the same university. Her main research interest centres around time series analysis, anomaly detection, and data science approaches to solving the sensor data quality-related issues.

Kevin I-Kai Wang is a Senior Lecturer in the Department of Electrical, Computer and Software Engineering, the University of Auckland. He worked in industries as a research engineer designing commercial home automation systems and traffic sensing systems from 2009 to 2011. His current research interests include wireless sensor network based ambient intelligence, pervasive healthcare systems, human activity recognition, behaviour data analytics and bio-cybernetic systems.

Andreas W. Kempa-Liehr is a Senior Lecturer at the Department of Engineering Science of the University of Auckland, New Zealand, and an Associate Member of the Freiburg Materials Research Center (FMF) at the University of Freiburg, Germany. He received his doctorate from the University of Münster in 2004 and continued his research as head of the service group Scientific Information Processing at FMF. From 2009 to 2016, he was working in different data science roles at EnBW Energie Baden-Württemberg AG and Blue Yonder GmbH in Karlsruhe, Germany.