International Journal of Data Science and Analytics (2020) 9:285–314
https://doi.org/10.1007/s41060-019-00191-3

REGULAR PAPER

Unsupervised online detection and prediction of outliers in streams of sensor data

Niko Reunanen[1] · Tomi Räty[2] · Juho J. Jokinen[2] · Tyler Hoyt[3] · David Culler[3]

Received: 22 August 2018 / Accepted: 23 May 2019 / Published online: 3 June 2019

Abstract
Outliers are unexpected observations, which deviate from the majority of observations. Outlier detection and prediction are challenging tasks, because outliers are rare by definition. A stream is an unbounded source of data, which has to be processed promptly. This article proposes novel methods for outlier detection and outlier prediction in streams of sensor data. The outlier detection is an independent, unsupervised process, which is implemented using an autoencoder. The outlier detection continuously evaluates if the latest data point $\mathbf{x}_i$ from a stream is an inlier or an outlier. This distinction is based on the reconstruction cost accompanied with Chebyshev's inequality and the EWMA (exponentially weighted moving average) model. The outlier prediction uses the results of the outlier detection to form the required training data. The outlier prediction utilizes LR (logistic regression), SGD (stochastic gradient descent) and the hidden representation provided by the autoencoder to predict outliers in streams. The results of the experiments show that the proposed methods (1) provide accurate results, (2) are calculated in reduced computation time and (3) use a low amount of memory. Our proposed methods are suitable for analyzing streams of sensor data and providing results with low latency. The experiments also indicated that the outlier prediction is able to anticipate the occurrence of outliers in streams of sensor data.

Keywords Outlier detection · Outlier prediction · Data streams · Machine learning · Unsupervised learning

1 Introduction

Outliers are unexpected observations, which deviate significantly from the expected observations and typically correspond to critical events [22,40,82]. The expected observations are called inliers. Detecting outliers is beneficial, because outliers contain important information in many application domains [18,32,34,38,62,65,86,91].

The prediction of the occurrence of the outliers is also useful [27,57,80]. In the context of sensor networks, the prediction of the occurrence of outliers could provide a prior indication of mechanical faults [22] and sensor faults [93]. The prediction of the occurrence of outliers is challenging, because outliers are observed infrequently [35] and the training data for detecting outliers is typically not available. This article calls the prediction of the occurrence of outliers, or rare events, as outlier prediction.

Many of the methods in outlier detection [17,43,49–51,59,63,68,92] are designed to detect outliers in static datasets, which have an infinite amount of data. Outlier detection is not limited to the static datasets, because data arrives continuously in many applications [3,21]. The outliers can be detected in a continuous flow of data, which is called stream. A stream is explained in detail in Chapter 2. In the context of analyzing streams of sensor data, this paper proposes novel methods for (1) detecting outliers and (2) predicting the occurrence of outliers in t time steps in the future. The methods learn to detect and predict the outliers in streams

✉ Tomi Räty
tomi.raty@vtt.fi

Niko Reunanen
niko.reunanen@hellon.com

Juho J. Jokinen
juho.jokinen@vtt.fi

Tyler Hoyt
thoyt@berkeley.edu

David Culler
culler@cs.berkeley.edu

[1] Hellon Oy, Pursimiehenkatu 26 C, 00150 Helsinki, Finland

[2] VTT Technical Research Centre of Finland, Kaitoväylä 1, 90571 Oulu, Finland

[3] Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94709, USA

without labeled training data. The proposed approach uses an unsupervised method for detecting outliers in streams. The detected outliers are then used as labels for the prediction algorithm. Chapter 3 introduces the proposed methods in detail.

Evaluation experiments in Chapter 5, which use sensor data, show that our proposed approach is an accurate method compared to the relevant algorithms in the literature. Using real-world sensor data, our proposed method provides comparable or better results with respect to the compared algorithms in outlier detection. However, to acquire superior or comparable results in the evaluation experiments in relation to our proposal, the compared algorithms allocate memory up to 10,000 floating point values. Our proposed method allocates memory only from 17 to 83 floating point values. Therefore, our proposed method allocates a significantly smaller amount of memory. Additionally, the processing of 17 to 83 floating point values is much faster than constantly querying the 10,000 floating point values.

To elaborate further, an autoencoder has been utilized for outlier detection in Ma et al. [55], Sakurada and Yairi [67], Chen et al. [23], Lu et al. [53], Dong and Japkowicz [29]. See Chapter 2.3 for a definition of an autoencoder. However, most of the existing work detects outliers in static datasets and is not suitable for analyzing streams of sensor data. In Dong and Japkowicz [29], an ensemble of autoencoders has been utilized for detecting outliers in streaming data. The classification of a single data point into inliers and outliers happens by the majority vote of the autoencoders. Autoencoders are trained using outlier-free data stream with mini-batch gradient descent for multiple epochs and the detection threshold is empirically set based on this training data. After the training phase, stochastic gradient descent is used to update the parameters in the autoencoders. Chen et al. [23] also use an ensemble of autoencoders to detect outliers, but only in static data sets. They add random edge sampling between the nodes and adaptive data sampling to increase the robustness and to decrease overfitting of the autoencoders. A combination of a denoising autoencoder and recurrent neural network is used in Lu et al. [53] for sequential outlier detection, and the model is trained using static outlier-free training data. Our work also extends the use of an autoencoder for outlier detection in streams of sensor data, which are missing in the literature. Some of the benefits and novelties of our work include:

- A fully automated framework for lightweight and efficient unsupervised outlier detection and prediction in streams of sensor data is introduced by combining an autoencoder and logistic regression (see Chapters 2.2 and 2.3). One of main the benefits of our proposed method of automated outlier prediction is the sharing of information between the outlier detection and outlier prediction.
- A mechanism for classifying data points into outliers and inliers, which can adapt to the changes in the inlier distribution (see Chapter 3.1).
- An approach for training an initial version (calibration) of the autoencoder in an unsupervised fashion (see Chapter 3.2).
- We propose novel methods for the automated calibration of outlier detection and outlier prediction and for the automated and unsupervised outlier detection using the reconstruction cost of the autoencoder. To our knowledge, in outlier prediction, this publication is the first to utilize the hidden representation of an autoencoder (see Chapter 3.3).
- An analysis of the properties (autoencoder parameters, distribution of the reconstruction cost, and integration of outlier detection and prediction) of the outlier detection using an autoencoder with streams (see Chapter 3.4).
- Our proposed methods use a small amount of memory are fast to compute and are accurate for streams of sensor data (see Chapters 4 and 5).

The proposed method for outlier detection is fast to compute, because it does not store the observations from a stream, which results in requiring a low amount of memory. This is beneficial for analyzing sensor data when the readings arrive at a high speed, and the results are expected to have a low latency. The proposed method defines a parametric model, which updates its parameters using one observation at a time. Therefore, the model does not store a reference set of the observations for neighbor queries (contrary to Angiulli and Fassetti [3], Kontaki et al. [48], Yang et al. [90], Cao et al. [20], in which the models stored a reference set). For example, the majority of the recent outlier detection methods, which utilize autoencoders (Chen et al. [23], Lu et al. [53]), are relying on the fact that there exists a static dataset containing either outlier-free training data or labeled training examples for the training of the autoencoder. These trained autoencoders lack adaptation capabilities if the inlier data distribution is evolving. Our method provides a more robust approach by including an adaptation scheme for the inlier distribution, and our initial training of the autoencoder is based on the unsupervised framework, which removes the need for outlier-free training data or labeled training examples. It is also important to detect the critical events accurately without many false positive detections. The accuracy of our proposed method for outlier detection is evaluated in Chapter 5.

For outlier prediction, our proposed method is capable of predicting the occurrence of outliers. Unlike the previous work in outlier prediction, our proposed method uses an autoencoder, which learns a hidden representation for outlier detection, to provide additional information for the outlier prediction. The hidden representation encodes the important information in the data [73]. The hidden presentation can

provide information to the outlier prediction, which is not available in the original data. Our proposed method for outlier prediction is an unsupervised black box model, which analyzes sensor data without external information. The literature does not contain research within the same scope as our research. See Chapter 5.5 for further discussion and the evaluation experiments of the outlier prediction. A complete listing of the benefits and the novelties of our approach are presented in detail in the end of Chapter 4.2.

Souiden et al. [74] list three key requirements associated with the analysis of data streams. The algorithms, which analyze data streams, should be able to scan the arriving data in real time and the possible changes in the data distribution should be accounted for. Also, memory and time constrains can exist and algorithms should be able to provide good performance despite these constraints. Our proposed framework is designed to address all of these three challenges and by doing so we provide a clear contribution when compared to existing methods.

The article is organized as follows. Chapter 2 introduces the required theoretical foundation for understanding the proposed method for outlier detection and prediction in this article. Chapter 3 introduces the proposed methods for outlier detection and prediction. Chapter 4 surveys the recent and related work in outlier detection in streams and outlier prediction. Chapter 5 explains the experiments that validate the proposed methods and reports the results of the experiments. Chapter 6 is the final section, which summarizes and concludes the article.

2 Preliminary topics

This section introduces the required theoretical foundation for our proposed methods. The outliers are detected and predicted in a stream, which is a continuous transmission of values. A stream is defined in detail in Chapter 2.1. The outliers are detected using an autoencoder, which learns a hidden representation of the data. The hidden representation measures how unexpected the data are. An autoencoder is defined in detail in Chapter 2.3. The outliers are predicted using a classifier, which uses the received data and the hidden representation of the autoencoder. The autoencoder has two functions: (1) outlier detection and (2) provision of a hidden representation (see Chapters 2.3 and 3.3) of data as a source of additional information for the outlier prediction. Our article uses logistic regression (LR) as the classifier, and it is defined in detail in Chapter 2.2. A more detailed description is given, because LR is optimized using stochastic gradient descent, instead of the more traditional methods such as iterative re-weighted least squares or conjugate gradient. LR also provides probabilities, which may be used in the future research for measuring the confidence of the predictions. Other widely used classifiers (e.g., support vector machines and perceptrons) do not inherently output classification probabilities, see Bishop [12]. The conversion of a classifier to produce a probabilistic output typically provides poorly calibrated probabilities [58]. LR directly offers a well-calibrated estimate of the classification probability. However, for completeness, Appendix A provides the outlier prediction results using support vector machine and Appendix B using Perceptron. The experimental results in Chapter 5, Appendix A and Appendix B show that LR outperforms support vector machine and perceptron in outlier prediction. Our proposed methods, which utilize the autoencoder and LR, are defined in detail in Chapter 3. The following subsection defines a stream in detail, which is a source of the analyzed data.

2.1 Stream

A stream is a continuous transmission of values, which arrive sequentially. Let a real valued vector ($\mathbf{x}_i \in \mathbb{R}^d$), which is called a data point, denote the ith observed vector of d-dimensional data from a stream. The elements ($x_j \in \mathbf{x}_i = \{x_1, x_2, \ldots, x_d\}$) of $\mathbf{x}_i$ are called features. The vector space ($\mathbb{R}^d$), which is spanned by the d features, is called the feature space.

The high speeds of data flows in a stream requires an efficient processing of the stream [20,45,48,78]. If the observed values in a stream are analyzed, then the analysis has to be fast to provide the results in real time [1]. The analysis of a stream must adjust to the changes in the characteristics of the data. The observed values are not assumed to be created by a stationary process, which makes the most recently observed values important, because they contain the newest information [3]. The newest information may reveal changes in the process, which creates the observations. Cao el al. state that the recent observations have more impact on the outlier detection [20]. Angiulli and Fassetti state that the recent observations are typically more significant in streams [3].

A typical definition of a stream is the following: it (1) contains a high volume of data [1,20,48], (2) does not have a distinctive termination, (3) is continuously updated [4] and (4) is unbounded [3,45,48]. The data cannot be stored entirely in the computer memory [48]. The previously listed facts mean that recording a stream results in a static dataset ($\mathbf{X} \in \mathbb{R}^{\infty \times d}$) with an infinite ($n = \infty$) or undefined number of rows. It is impractical to store and handle a dataset, which exhibits unlimited growth in size. The consumption of the memory [48] and the number of rows have to be restricted. Therefore the typical methods for outlier detection in static datasets are not applicable for streams.

A stream can be utilized by using subsets of the stream. Outlier detection using the subsets is a local method [45] for outlier detection, because the data are used partially. The subsets of the data are assumed to contain the characteristics of

the normal data. The characteristics define the normal region for inliers in the feature space. The outliers are outliers with respect to the data within the subsets [4]. The subsets are called windows, which are stored in the computer memory.

Our work utilizes an approach called count-based sliding window [48] for the outlier prediction. The count-based sliding window contains a continuously updated subset of the stream data. The approach of using a sliding window for outlier detection is used in many publications [1,3,6,20,36,48,75,76,88,90]. The subset of the data in the sliding window defines two endpoints (a point of commencement and a point of termination) in the stream [3]. The members of the subset are called active objects [37], because the objects are activated for use. Let PS (pattern size) denote the number of data points or active objects in a sliding window. A count-based sliding window contains PS active objects [37,48,90] from the stream. The cardinality of the subset (PS) is constant. An active object that is removed from the subset is said to expire [20,48,90]. The count-based window implements a queue of PS objects where the first object to arrive is the last object to expire. The expiration time for an active object is PS, because an active object is removed after PS subsequent observations.

2.2 Logistic regression

Logistic regression (LR) is a statistical model, which can be used to classify or predict the outcome of a binary variable. In binary classification, a classifier assigns a class (y) to a given data sample ($\mathbf{x}$) from the two available classes ($c = \{c_1, c_2\}, y \in c$). The target classes belong in the available classes ($\forall i \; y_i \in c$). The goal of a classifier is to assign a data point ($\mathbf{x}_i$) into a correct target class (y_i) [12]. Let $\mathbf{w}$ denote a d-dimensional vector of weight values ($\mathbf{w} = \{w_1, w_2, \ldots, w_d\}$). LR models the classified phenomenon using a linear combination ($\mathbf{w}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{d} w_i x_i$) of an input vector ($\mathbf{x}$) and the weight vector ($\mathbf{w}$). The linear combination is transformed using a logistic sigmoid function [79], which forces the transformed value within a finite interval (between 0 and 1) [12]. The logistic sigmoid function is defined as

$$s(a) = \frac{e^a}{1 + e^a} = \frac{1}{1 + e^{-a}}. \tag{1}$$

Let $c = \{0, 1\}$ denote the set of the two available classes and let $P(c_1|\mathbf{x})$ denote the probability (P) of a data point ($\mathbf{x}$) belonging to the first class (c_1) and let $P(c_2|\mathbf{x})$ denote the probability (P) of a data point ($\mathbf{x}$) belonging to the second class (c_2). The model of LR for a data point is

$$P(c_2|\mathbf{x}) = s(\mathbf{w}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^{\mathrm{T}}\mathbf{x}}}, \tag{2}$$

which models the posterior probability of the data point belonging to the second class.

The ratio between $P(c_2|\mathbf{x})$ and $P(c_1|\mathbf{x})$ is called odds [12], which measures the relative likeliness of $P(c_2|\mathbf{x})$. LR approximates the logarithm of the odds (log-odds) directly using the linear combination $\mathbf{w}^{\mathrm{T}}\mathbf{x}$ as

$$\ln \frac{P(c_2|\mathbf{x})}{P(c_1|\mathbf{x})} = \ln \frac{P(c_2|\mathbf{x})}{1 - P(c_2|\mathbf{x})} = \mathbf{w}^{\mathrm{T}}\mathbf{x}. \tag{3}$$

Solving the log-odds (Eq. 3) for $P(c_2|\mathbf{x})$ results in Eq. 2. The result can be derived from $\frac{P(c_2|\mathbf{x})}{1-P(c_2|\mathbf{x})} = e^{\mathbf{w}^{\mathrm{T}}\mathbf{x}}$. For minimizing the misclassification rate, a data point ($\mathbf{x}$) is classified in the second class if $P(c_2|\mathbf{x}) > P(c_1|\mathbf{x})$. The decision rule for the second class (c_2) is equivalent to $P(c_2|\mathbf{x}) > 0.5$, because $P(c_2|\mathbf{x}) > 1 - P(c_2|\mathbf{x}) = P(c_1|\mathbf{x})$ given $P(c_2|\mathbf{x}) > 0.5$.

Let $\mathbb{X}$ denote the available data ($\{\mathbf{x}_i, y_i\}_{i=1}^{n}$) of n training examples for determining the weight values ($\mathbf{w}$). LR assumes a binomial likelihood function for the given data and it is defined as

$$\mathcal{L}(\mathbf{w}|\mathbb{X}) = \prod_{i=1}^{n} P(c_2|\mathbf{x}_i)^{y_i} P(c_1|\mathbf{x}_i)^{1-y_i}, \tag{4}$$

which measures the likelihood of observing the data given the specific model. A log-likelihood function (ℓ) is defined as the logarithm of the likelihood function ($\ell = \ln \mathcal{L}$). The log-likelihood function of LR model is

$$\ell(\mathbf{w}|\mathbb{X}) = \sum_{i=1}^{n} y_i \ln P(c_2|\mathbf{x}_i) + (1 - y_i) \ln P(c_1|\mathbf{x}_i), \tag{5}$$

where n is the number of training examples ($\{\mathbf{x}, y\}$) in the training data ($\mathbb{X}$), $P(c_i|\mathbf{x}_i)$ is the posterior probability of a class (c_i) given a data point, y_i is the target class of a training example and the vector $\mathbf{w}$ contains the weight values. The optimal parameters ($\mathbf{w}$) maximize the value of the log-likelihood function. LR can be trained, given training data ($\mathbb{X}$), by estimating the parameters ($\mathbf{w}$) that maximize the log-likelihood (ℓ).

Stochastic gradient descent (SGD) finds a local minimum of a function with multiple variables [12]. If the function is convex, then the local minimum is the global minimum. If the function is not convex, then SGD does not guarantee that the local minimum is the global minimum. Notice that a maximization problem can be transformed into a minimization problem by changing the sign of the objective function. The negative log-likelihood function in LR is convex, which means that given enough iterations and some general assumptions (see Bottou [13] for details), SGD converges to a global minimum. SGD moves the values of the parameters in the direction of the gradient, which is the direction of the greatest

rate of change in the function value. SGD estimates the gradient using one training example ($\{\mathbf{x}_i, y_i\}$) at a time. Therefore SGD is applicable when the training data does not fit entirely in the computer memory [12]. Only a subset of the data set is retained in the computer memory at once. We use SGD to maximize the log-likelihood (ℓ) with respect to the parameters ($\mathbf{w}$) given training data $\mathbb{X}$.

For utilizing SGD, we use the partial derivatives ($\frac{\partial \ell(\mathbf{w}|\mathbb{X})}{\partial w_j}$) of the log-likelihood function with respect to the individual weights ($w_j \in \mathbf{w}$). The partial derivatives let us to adjust the weights individually for maximizing the log-likelihood function. Let β_i be defined as $\beta_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}_i$. Notice that $P(c_1|\mathbf{x}_i) = 1 - P(c_2|\mathbf{x}_i)$. The expression evaluates to $\frac{1+e^{\beta_i}}{1+e^{\beta_i}} - \frac{e^{\beta_i}}{1+e^{\beta_i}} = \frac{1}{1+e^{\beta_i}}$ using Eq. 1. After substituting $P(c_2|\mathbf{x}_i) = \frac{e^{\beta_i}}{1+e^{\beta_i}}$, $P(c_1|\mathbf{x}_i) = \frac{1}{1+e^{\beta_i}}$ in Eq. 5, and doing algebraic simplification, the following form of the log-likelihood function is acquired:

$$\begin{aligned}\ell(\mathbf{w}|\mathbb{X}) &= \sum_{i=1}^{n} y_i\beta_i - \ln(1+e^{\beta_i}) \\ &= \sum_{i=1}^{n} y_i\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - \ln(1+e^{\mathbf{w}^{\mathrm{T}}\mathbf{x}_i}).\end{aligned} \tag{6}$$

A partial derivative of the previous form (Eq. 6) of the log-likelihood function (Eq. 5) with respect to a single weight value is defined as

$$\begin{aligned}\frac{\partial \ell(\mathbf{w}|\mathbb{X})}{\partial w_j} &= \sum_{i=1}^{n} y_i\mathbf{x}_{ij} - \frac{\mathbf{x}_{ij}e^{\mathbf{w}^{\mathrm{T}}\mathbf{x}_i}}{1+e^{\mathbf{w}^{\mathrm{T}}\mathbf{x}_i}} \\ &= \sum_{i=1}^{n}(y_i - P(c_2|\mathbf{x}_i))\mathbf{x}_{ij},\end{aligned} \tag{7}$$

where $\mathbf{x}_{ij}$ is the jth feature of the ith training example. Instead of using all of the training examples (n), SGD uses a training example at a time to re-evaluate the parameters as

$$\mathbf{w} = \mathbf{w} + \alpha\nabla\ell(\mathbf{w}|\mathbf{x}_i), \tag{8}$$

where the parameter α is called learning rate and $\nabla\ell$ is the gradient formed by the partial derivatives. The learning rate adjusts the magnitude of the change in the parameter values. A high value of the learning rate adapts a model quickly to newest information and makes it less dependent on the old information. A low value of the learning rate adjusts the parameters of the model carefully and relies more on the previously learned information.

2.3 Autoencoder

An autoencoder is an artificial neural network, which learns a hidden representation of the input data [9,84]. Let $\mathbf{x}$ ($\mathbf{y}$) denote the input data (hidden representation) as a d (h)-dimensional vector ($\mathbf{x} \in [0,1]^d$, $\mathbf{y} \in [0,1]^h$) of values between 0 and 1. Let $\mathbf{W}$ denote a weight matrix with h rows and d columns ($\mathbf{W} \in \mathbb{R}^{h\times d}$). Let $\mathbf{b}$ and $\mathbf{b}_z$ denote h-dimensional vectors of real values ($\mathbf{b} \in \mathbb{R}^h$, $\mathbf{b}_z \in \mathbb{R}^h$), which are called bias vectors. The input data ($\mathbf{x}$) are transformed into a hidden representation ($\mathbf{y}$) as

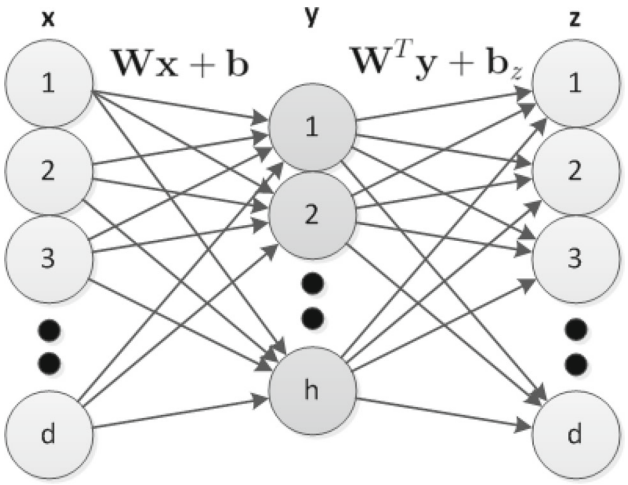

Fig. 1 The structure of an autoencoder where $\mathbf{x} \in \mathbb{R}^d$ contains the input values, $\mathbf{y} \in \mathbb{R}^h$ is the hidden representation of the input values, $\mathbf{z} \in \mathbb{R}^d$ is the reconstruction of the input, $\mathbf{W}$ is a matrix containing the weight values and $\mathbf{b}$, $\mathbf{b}_z$ are bias vectors

$$\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{9}$$

where $s(a)$ is the logistic sigmoid function (see Eq. 1). The bias vectors ($\mathbf{b}$, $\mathbf{b}_z$) translate the location of the sigmoid function with respect to its input. The rows of the weight matrix, the corresponding bias values and the nonlinear transformation create a model called a neuron [84]. The neurons forming the hidden representation are grouped as a layer called hidden layer. The hidden representation is transformed again into the input space ($[0,1]^d$) of the input data as

$$\mathbf{z} = s(\mathbf{W}^{\mathrm{T}}\mathbf{y} + \mathbf{b}_z), \tag{10}$$

where $\mathbf{W}^{\mathrm{T}}$ is the conjugate transpose of the weight matrix. The vector z contains the reconstructed data of the input data ($\mathbf{x}$). The structure of an autoencoder is illustrated in Fig. 1.

We use a loss function (L) which measures the difference of the reconstructed input to the original input data. The difference is measured as the sum of the absolute differences between the features of the original input ($\mathbf{x}_i$) and the reconstructed input ($\mathbf{z}_i$). The loss function for the reconstruction cost is defined as:

$$r_i = \sum_{j=1}^{d} |x_j - z_j|, \tag{11}$$

where all of the d variables are iterated. If the reconstruction cost is zero ($L = 0$), then the hidden representation models the input data perfectly. Note that if $L = 0$, then noise in the input data is also learned [84]. Therefore an autoencoder should discard the noise while retaining the important information. In this article, the hidden representation has a smaller dimensionality than the feature space of the modeled data ($h < l$). This forces the hidden representation to learn high-level features of the data by combining the original features [84]. The smaller dimensionality alleviates the analysis of the data, because (1) the resulting feature space is denser and (2) the high-level features encode nonlinear relationships between the original features. A dense feature space, which is spanned by the learned features for modeling a complex event, alleviates the curse of dimensionality (see Bishop [12]). This is a suitable point to commence the analysis of the data.

Our work uses SGD to continuously re-estimate the weight matrix ($\mathbf{W}$) and the bias vectors ($\mathbf{b}, \mathbf{b}_z$). SGD minimizes the value of reconstruction cost (L) using its partial derivatives with respect to the autoencoder parameters ($\frac{\partial L}{\partial \mathbf{W}}, \frac{\partial L}{\partial \mathbf{b}}, \frac{\partial L}{\partial \mathbf{b}_z}$). The minimization of L, which is called training, forces the autoencoder to learn a hidden representation of the input data. Referring to the generic training mechanism of SGD in Eq. 8, the training rules of an autoencoder using SGD are defined as

$$\mathbf{W}^{\text{new}} = \mathbf{W}^{\text{old}} - \alpha \frac{\partial L}{\partial \mathbf{W}^{\text{old}}}, \tag{12}$$

$$\mathbf{b}^{\text{new}} = \mathbf{b}^{\text{old}} - \alpha \frac{\partial L}{\partial \mathbf{b}^{\text{old}}}, \tag{13}$$

$$\mathbf{b}_z^{\text{new}} = \mathbf{b}_z^{\text{old}} - \alpha \frac{\partial L}{\partial \mathbf{b}_z^{\text{old}}}. \tag{14}$$

The following chapter introduces the methods for outlier detection and prediction, which are proposed in this article. The novelty of the methods compared to the existing work is explained in detail in Chapter 3.

3 The proposed methods

The proposed system (see Fig. 2) for outlier prediction in this article consists of two core functionalities: (1) outlier detection in a stream and (2) outlier prediction in a stream. The outlier detection is an independent process, which is implemented using an autoencoder. The outlier detection continuously evaluates if the latest data point ($\mathbf{x}_i$) from a stream is an inlier or an outlier. The outlier prediction uses the results of the outlier detection to form the required training data. The outlier prediction is not an independent process, because it relies on the results of the outlier detection. Our study assumes that the data contains the required information for the outlier prediction without having an external source of data. The following subsections explain the proposed outlier detection and prediction in detail.

3.1 Outlier detection in a stream

This section describes the framework for the outlier detection. We introduce how the discrimination between inliers and outliers is built using Chebyshev's inequality. Details about the learning rate for the autoencoder and how to handle subsequent identical data points are also given.

Outliers are detected using an autoencoder with a hidden layer of neurons, which use the sigmoid function as the nonlinear transformation. The proposed system defines an outlier as a data point ($\mathbf{x}_i$), which has an unexpectedly high value of the reconstruction cost (Eq. 11), given a distribution of values of the reconstruction cost. An outlier contains unexpected information, which cannot be modeled or reasoned with the available data. The unexpected information is observed as extreme values and deviations in the correlations of the data features. The feature values are scaled between 0 and 1 ($\mathbf{x}_{ij} \in [0, 1]$) as

$$\mathbf{x}_i = \frac{\mathbf{x}_i - \mathbf{x}_{\min}}{\mathbf{x}_{\max} - \mathbf{x}_{\min}}, \tag{15}$$

where $\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$ are vectors that contain the maximum and minimum values of the features, up to the evaluation of the current data point. The scaling is computed element-wise per feature. The limits of the features ($\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$) are not known in advance, because a stream is unbounded.

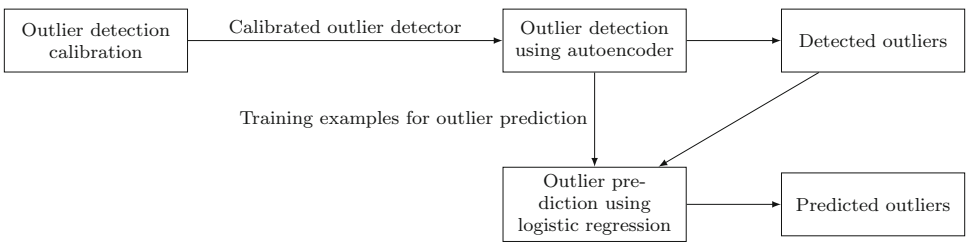

Fig. 2 An illustration of the general process for the proposed method

The limits are continuously updated when data arrives from the stream. However, the limits are not updated if the data point is classified as an outlier. This prevents outliers from affecting the outlier detection, because they are not allowed to modify the scaling limits ($\mathbf{x}_{\max}, \mathbf{x}_{\min}$).

Let $\mathbf{x}_i$ denote a data point that has been acquired from a stream. After scaling the feature values between 0 and 1, the autoencoder model is updated using the data point. The weights of the autoencoder are adjusted using SGD (Eqs. 12–14), after which the reconstruction cost of the data point is recalculated using the autoencoder. The reconstruction costs are positive real numbers, which are not limited. The distribution of the reconstruction costs is not known. It is not possible to define a precise value for a low reconstruction cost (inlier) and a high reconstruction cost (outlier). If we had the distribution available, then we could use it to define a threshold for the reconstruction cost. This threshold would then discriminate the outliers from the inliers with a given probability.

Chebyshev's inequality proves that the majority of distribution values are clustered around the distribution mean [71]. The inequality is defined as

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}, \tag{16}$$

where X is a random variable, σ is the standard deviation and μ is the mean [71]. The inequality calculates an upper bound for the probability of random variates to exceed k standard deviations from the mean. Using the inequality, there is an upper bound of $\frac{1}{9}$ for the probability of observing random variates, which exceed the mean by three standard deviations. Therefore, we suggest to define an outlier as a data point with a reconstruction cost (L) that exceeds the expected reconstruction cost with three standard deviations. The three standard deviations are also utilized in Zhao et al. [94], Ferdowsi et al. [31]. However, in the evaluation of our proposed method in Chapter 5, we test a range of multipliers for the standard deviation.

The data in a stream does not fit entirely in the memory, so only the descriptive statistics (μ, σ) of the reconstruction cost are maintained. The descriptive statistics of the inlier distribution may change with time, so the proposed system calculates the mean and the standard deviation of the reconstruction cost sequentially, using *exponentially weighted moving average* model or *EWMA* model. The algorithm for the sequential calculation comes from Finch [33], which is also utilized in Schuhknecht et al. [70]; Schubert et al. [69]. The statistics are continuously updated as

$$\mu_{\text{new}} = (1 - \gamma)\mu_{\text{old}} + \gamma L, \tag{17}$$

$$S_{\text{new}} = (1 - \gamma)(S_{\text{old}} + \gamma(L - \mu_{\text{old}})^2), \tag{18}$$

$$\sigma_{\text{new}} = \sqrt{S_{\text{new}}}, \tag{19}$$

where μ_{old} and S_{old} are the current estimates of the mean and the variance, L is the reconstruction cost (Eq. 11) of the newest data point ($\mathbf{x}_i$) from the stream and $\gamma \in [0, 1]$ controls the rate of the updates of the descriptive statistics (e.g., if $\gamma = 0$, then the statistics do not change at all). As the EWMA method has been shown to detect shifts in the underlying process (see Lucas and Saccucci [54], Braimah et al. [16], Raza et al. [66]), we utilize it in our method so that the possible changes in the inlier distribution can adapted to. However, data streams from different sources may need different weighting the parameter γ, in order to achieve good results. As a default value, we utilize the value $\gamma = 0.1$ as in Schuhknecht et al. [70].

Identical subsequent data points are ignored for increasing the efficiency and decreasing the computational complexity of the outlier detection. The identical subsequent data points deviate less when more evidence is gathered for their existence. In our work, evidence is defined as the amount of observations of a given phenomenon. If one observes multiple instances of a specific data point, then there is more evidence for the data point.

Ignoring the data points, which are identical and subsequent, protects the autoencoder from becoming too specific (overfit) for the data and the noise in the data. The autoencoder overfits itself by modifying the weights ($\mathbf{W}$) to memorize (not learn) the data and the reconstruction cost results in zero ($L = 0$), which is undesirable. By ignoring identical data, the model is less likely to forget previously learned information. Overfitting the identical data erases the previously learned characteristics of the data. Notice that the recently learned, and then erased, characteristics could still be valid for the data. The erased characteristics are the features, and their respective weight values in $\mathbf{W}$, which are not available anymore at the current time. The reconstruction of data, which have the erased characteristics, results in high values of the reconstruction cost (L) using Eq. 10. These high values of the reconstruction cost result in incorrect outlier detections.

A data point with low evidence might become an inlier in the future by gaining more evidence. This can ultimately lead to an incorrect outlier detection. From the point of view of our proposed method, a data point with low evidence is an outlier. This is desired and normal behavior, because the goal of our proposed method is to detect outliers without external knowledge. Given a data point, the proposed method commences the outlier detection and prediction without waiting for more data. If more evidence becomes available, then the autoencoder will eventually learn the characteristics of the data. In addition to outliers, our proposed method detects novelties or change points, which are previously unseen and different inliers. The novelties are significantly deviating inliers without being outliers.

The learning rate (α) of SGD for the autoencoder is chosen to be a constant value instead of a gradually decreasing value. The amount of the data in a stream is considered infinite and thus the learning rate cannot be decreased as a function of time. The data in a stream is not assumed to originate from a stationary distribution and therefore the autoencoder has to adapt to any changes in the concept of the normal and deviating data. A constant small value of the learning rate allows the model to forget the oldest information and to obtain a small variance in the convergence of the updated parameters. We select a constant learning rate of $\alpha = 0.1$ for SGD to train the autoencoder (as in Cho et al. [24]). The following subsection discusses the calibration of the proposed method, which is required before commencing the classification of data points in outliers and inliers.

3.2 Calibration of the outlier detection

Our proposed method for outlier detection needs to be calibrated before commencing the classification of the observations in outliers and inliers. The calibration is required, because the method does not impose assumptions on the characteristics of the stream data. The calibration consists of two consecutive phases:

1. *Phase 1* The determination of initial scaling limits ($\mathbf{x}_{\max}, \mathbf{x}_{\min}$ in Eq. 15).
2. *Phase 2* Initial training of the autoencoder (Eqs. 12–14 and Eqs. 17–19).

The determination of the initial scaling limits is the first phase of the calibration. Our proposed method is unsupervised and does not impose assumptions on the data of the analyzed sensor stream. As a result of having no prior information, the scaling limits ($\mathbf{x}_{\max}, \mathbf{x}_{\min}$) in Eq. 15 are not defined in advance. The initial scaling limits are calibrated when the d elements of the lower limits ($\mathbf{x}_{\min}$) are lower than the elements of the upper limits ($\mathbf{x}_{\max}$). Data points are received from the stream until the initial scaling limits are calibrated. The equation $\mathbf{x}_{\max} - \mathbf{x}_{\min} > 0$ holds element-wise for the calibrated scaling limits. As a result, the scaling in Eq. 15 does not divide by zero for any element-wise feature. This is important, because the autoencoder is trained using data points with scaled values of the features.

The initial training of the autoencoder is the second phase of the calibration, which follows after the determination of the initial scaling limits. The autoencoder does not have prior information available of the distribution of the analyzed data. Therefore, for informed outlier detections, we need initial training for (1) the autoencoder and (2) the running estimates of the descriptive statistics (μ, σ) of the reconstruction cost. A sufficient number of data points are processed from the sensor stream before classifying the data points into outliers and inliers. The initial data points train an initial version of the model for outlier detection. The next section introduces a heuristic, which determines when the calibration of the autoencoder is finished (phase 2). The utilization of the heuristic is beneficial, because we do not need to explicitly define a value for the sufficient number of initially consumed data points. The autoencoder calculates the reconstruction cost for the data points. The running estimates are considered as the final estimates at any give time ($\forall i$), because a stream is an unbounded source of data.

3.2.1 Patience heuristic

To automate the calibration of the autoencoder and to determine what is a sufficient number of data points, we apply a heuristic called patience [8]. Let r_i denote the reconstruction cost of a data point ($\mathbf{x}_i$). The reconstruction cost is calculated using Eq. 11. The maximum reconstruction cost per feature is one using Eq. 11. The reconstruction cost has a defined minimum value (zero) and maximum value (the number of features d). This results from scaling the input (Eq. 15) and the output (Eq. 10) features between zero and one. Therefore, the maximum reconstruction cost of a data point is $1 * d = d$, because there are d measured features.

Let $r_{\min}$ denote the smallest reconstruction cost, which is observed since the initiation of the calibration. Let P_{c} denote a counter variable, which identifies the determination of the calibration ($P_{\mathrm{c}} = 0$). The variable P_{c} is a counter, because every iteration in the calibration decrements the value of the counter by one ($P_{\mathrm{c}} = P_{\mathrm{c}} - 1$). Let P_{r} denote a reset value, which resets the counter ($P_{\mathrm{c}} = P_{\mathrm{r}}$). If the calibration needs to reset the counter variable (P_{c}), then the new value of the counter variable is P_{r}. The patience heuristic continues to train the autoencoder until the counter P_{c} equals zero. Every training iteration of the autoencoder decreases the value of the counter by one ($P_{\mathrm{c}} = P_{\mathrm{c}} - 1$). If a training iteration of the autoencoder is successful ($r_i < r_{\min}$), then the counter is reset ($P_{\mathrm{c}} = P_{\mathrm{r}}$) and the minimum reconstruction cost is updated ($r_{\min} = r_i$). Therefore, the patience heuristic attempts to train the autoencoder in a limited number of iterations, which is suitable for the calibration of our proposed method. However, the patience heuristic does not guarantee a termination in a limited number of iterations because, in theory, the improvements in $r_{\min}$ could be arbitrarily small [64]. To have an absolute guarantee of termination, Prechelt [64] suggests to define a maximum number of training iterations. The next section introduces an approach for determining a reset value (P_{r}), which conforms to a limit for the maximum number of training iterations.

3.2.2 Determination of the reset value

To determine when the calibration of the outlier detection is ready, we use the previously described patience heuristic and utilize a limit of a maximum number of initially consumed data points, which is motivated by Prechelt [64]. Let M denote the maximum number of data points for calibration and decmin denote the smallest decrease in the reconstruction cost: $r_{\min} = r_i$ if $r_{\min} - r_i > decmin$. The smallest decrease (decmin) is required, because the decreases of the value of $r_{\min}$ could be arbitrary small. The number of updates is theoretically infinite, because even the smallest decreases in $r_{\min}$ would reset the patience counter. The smallest decrease (decmin) limits the number of possible updates of the value of $r_{\min}$, because the maximum number of updates is d/decmin. The utilization of decmin and the patience heuristic, belong to a category of techniques called early stopping. Early stopping attempts to train a model, without overfitting the model, in a finite amount of time. See Bishop [12], Prechelt [64], Barber [7] for a detailed definition of the early stopping. We define and use the following formula to define a reset value (P_{r}) for a given M and decmin:

$$P_{\text{r}} = \frac{M * \text{decmin}}{d}. \tag{20}$$

The equation defines a value for P_{r} given an absolute upper bound of the number of data points (M). In the theoretical worst case, the autoencoder reduces the reconstruction cost by decmin and resets the patience counter ($P_{\text{c}} = P_{\text{r}}$) at the last possible iteration ($P_{\text{c}} = 1$). This results in a total consumption of $\text{decmin}^{-1} * d * P_{\text{r}}$ data points, because a maximum value of the reconstruction cost is d. If the model would reset the patience counter every time at the last iteration ($P_{\text{c}} = 1$), and the reconstruction cost would decrease by decmin at a time, then M is the resulting number of consumed data points. The autoencoder uses a sigmoid activation function, which ensures that the reconstructed values per feature are between zero and one. As established, the input values are scaled between zero and one. Therefore, the maximum possible reconstruction cost is d.

Fortunately, M is a very conservative maximum bound of the required data points. Instead of constantly reducing the reconstruction cost by decmin, the autoencoder is expected to learn the hidden representation faster. Indeed, on average, the number of consumed data points for the calibration was 72 in the experiments in Chapter 5. In our work, we define $M = 10{,}000$ as the upper bound, which is suggested for the patience value in Bengio [8]. For the minimum decrease of the reconstruction cost (decmin), we utilize a value of $\text{decmin} = 0.01$, because it is a minor (1%) improvement of the maximum reconstruction cost for data with one feature ($d = 1$).

The first phase, which is the calibration of the initial scaling limits, consumes data points from the stream. These consumed data points have to be accounted for in the determination of the maximum bound (M). Otherwise, the maximum bound is defined only for the calibration of the autoencoder (phase 2) and not for the calibration of the scaling limits (phase 1). The solution is to decrement the value of the maximum bound by the number of consumed data points in the first phase. This ensures that the maximum bound (M) is a true maximum bound for the entire calibration procedure. The only requirement for the calibration of the scaling limits is that the variance of the features is greater than zero. If the variance is zero, then a signal of constant values is observed for the feature and the maximum bound is not defined.

3.2.3 Algorithm for calibration of outlier detection

Our proposed method for outlier detection is calibrated and ready to use when (1) the scaling limits are calibrated ($\mathbf{x}_{\min} < \mathbf{x}_{\max}$ element-wise) and (2) the autoencoder is calibrated ($P_{\text{c}} = 0$). Algorithm 1 describes the calibration of the outlier detection. Algorithm 2 describes the proposed approach for outlier detection using an autoencoder, which utilizes the calibration approach in Algorithm 1. The following subsection introduces our proposed method for outlier prediction.

3.3 Outlier prediction in a stream

Outlier detection is responsible for providing the outliers, which are used to continuously train a classifier for predicting the outliers. Our work uses LR as the classifier, and SGD is applied to continuously re-evaluate the weight vector ($\mathbf{w}$) as data arrive from a stream. We utilize LR, because it typically provides a well-calibrated probability of the classification results. The probability may be used in the future as an indicator of how confident the model is in the classification results. LR is previously used to analyze data with outliers by King and Zeng [46]. Classifying the occurrence of an outlier is modeled as a binary outcome of LR. The second class (c_2) denotes the outliers and the first class (c_1) denotes the inliers. A positive classification is performed for the occurrence of an outlier if the probability of an outlier occurring is higher than the probability of an inlier occurring.

Let $\mathbf{x}_i$ denote a data point that is read from a stream. LR predicts an outlier by classifying a successor of $\mathbf{x}_i$ in t time steps ($\mathbf{x}_{i+t}$) as an outlier or as an inlier. The successor of $\mathbf{x}_i$ in t time steps is one data point at time $i + t$. Therefore the time step (t) defines the duration of how far in the future the outliers are predicted.

Logistic regression is trained using PS previous readings ($\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS}$) from the stream and the autoencoder labels the data point $\mathbf{x}_i$ to provide the target class. The hidden

Algorithm 1 Calibration of the outlier detection.

Input: A predefined structure for the autoencoder
Output: Calibrated reconstruction distribution statistics and weights for the autoencoder

Set $\mu_{\text{new}} = \mu_{\text{old}} = S_{\text{new}} = S_{\text{old}} = 0$, $r_{\min} = \infty$
Set $M = 10{,}000$ and decmin $= 0.01$ ▷ See Chapter 3.2.3 for justification
Initialize $\mathbf{b}$, $\mathbf{b}_z$ with zeroes, and $\mathbf{W}$ with random values between zero and one
while $\mathbf{x}_{\min} = \mathbf{x}_{\max}$ for any feature **do** ▷ Phase 1, see Chapter 3.2
    Acquire a data point $\mathbf{x}_i$ from the stream
    Update the limits of the data features ($\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$)
    $M = M - 1$ ▷ Calibration of scaling limits consumes data points
end while
$P_{\text{r}} = M * \text{decmin} * d^{-1}$ ▷ The reset value for patience heuristic (Eq. 20)
Set $P_{\text{c}} = P_{\text{r}}$
while $P_{\text{c}} > 0$ **do** ▷ Phase 2, see Chapter 3.2
    Acquire a data point $\mathbf{x}_i$ from the stream
    Update the limits of the data features ($\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$)
    Scale the feature values ($\mathbf{x}_{ij} \forall j$) using Eq. 15
    Calculate the reconstruction ($\mathbf{z}_i$) of the data point ($\mathbf{x}_i$) using Eq. 10
    Calculate the reconstruction cost (L) of the data point ($\mathbf{x}_i$) using the autoencoder (Eq. 11) and the reconstruction ($\mathbf{z}_i$) of the data point ($\mathbf{x}_i$)
    Update the autoencoder weights using SGD (Eq. 12–14)
    Update the statistics of the distribution of reconstruction cost (Eq. 17–19)
    Calculate $r = \sum_{i=1}^{d} |x_i - z_i|$ ▷ Absolute loss for patience (Eq. 11)
    if $r_{\min} - r >$ decmin **then** ▷ Is the minimum decrease exceeded?
        Set $r_{\min} = r$ and $P_{\text{c}} = P_{\text{r}}$ ▷ Update the patience variables
    end if
    $P_{\text{c}} = P_{\text{c}} - 1$ ▷ Decrement the patience counter
end while

Algorithm 2 The proposed algorithm for the outlier detection in streams.

Input: A predefined structure for the autoencoder
Output: Outlier classification for non-identical data points $\mathbf{x}_i$ in the stream

Calibrate the outlier detection based on the autoencoder structure given (Algorithm 1)
Initialize $\mathbf{x}_{\text{old}}$ with zeroes
Acquire a data point $\mathbf{x}_i$ from the stream
Go to (3) if $\mathbf{x}_i$ is identical with $\mathbf{x}_{\text{old}}$
Set $\mathbf{x}_{\text{old}} = \mathbf{x}_i$
if IS_OUTLIER($\mathbf{x}_i$) == *false* **then** ▷ Outliers do not affect the scaling
    Update the limits of the data features ($\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$)
end if
if IS_OUTLIER($\mathbf{x}_i$) == *true* **then**
    The data point ($\mathbf{x}_i$) is an outlier
end if
Update the autoencoder weights using SGD (Eq. 12–14)
Update the statistics of the distribution of reconstruction cost (Eq. 17–19)
Set $i = i + 1$ and go to (3) ▷ Ready to analyze the next data point

function IS_OUTLIER($\mathbf{x}_i$) ▷ $\mathbf{x}_i$: analyzed data point
    Scale the feature values ($\mathbf{x}_{ij} \forall j$) using Eq. 15
    Calculate the reconstruction ($\mathbf{z}_i$) of the data point ($\mathbf{x}_i$) using Eq. 10
    Calculate the reconstruction cost (L) of the data point ($\mathbf{x}_i$) using the autoencoder (Eq. 11) and the reconstruction ($\mathbf{z}_i$) of the data point ($\mathbf{x}_i$)
    return Outlier detection if $L > \mu_{\text{new}} + k\sigma$ ▷ $k = 3$ suggested by Eq. 16
end function

representation of data ($\{\mathbf{y}_{i-PS+j-t}\}_{j=0}^{PS}$, see Eq. 9) is also utilized if Algorithm 2 is used for the outlier detection. The hidden representation, in addition to the original data, is a source of information for predicting the occurrence of the outliers. To our knowledge, a hidden representation has not been utilized in outlier prediction in the literature. However, our proposed method for outlier prediction does not assume the availability of the hidden representation. If a hidden representation of the data is not available, then only the original data ($\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS}$) are utilized for the outlier prediction. In this article, we use an autoencoder for outlier detection, because the autoencoder provides additional data for the outlier prediction.

The PS readings are formed into sliding windows. Therefore, as defined earlier, the PS is the size of the sliding window. The first element of the sliding window at time i is

$\mathbf{x}_{i-PS+0-t}$, the second is $\mathbf{x}_{i-PS+1-t}$, the third is $\mathbf{x}_{i-PS+2-t}$ and so on. The hidden representation of data is appended in the end of the sliding window. The minimum value of PS is 1. There is no maximum value of PS. The number of the parameters of LR ($|\mathbf{w}|$) increases linearly with the size of the sliding window (PS). The feature space ($\mathbb{R}^d$) has d dimensions of real values. Let h be the number of hidden neurons in the autoencoder. There is a total of $(PS * (d + h)) + 1$ parameters ($|\mathbf{w}| = (PS * (d + h)) + 1$) in LR. The weight vector ($\mathbf{w}$) is grown with $d + h$ elements. LR becomes more complex when the value of PS increases. As with autoencoder, if there are many parameters ($\mathbf{w}$), then the LR only memorizes the training examples. This results in overfitting in which LR does not learn the training examples. The value of PS and the number of parameters ($|\mathbf{w}|$) are constant during the outlier prediction.

The time step (t) is subtracted from the time indices (i) of the elements in the sliding window. The sliding windows, which contain the data of t time steps from the past ($\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS}$ or $\{\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS} \cup \{\mathbf{y}_{i-PS+j-t}\}_{j=0}^{PS}\}$), are the features used for learning to predict. LR requires multiple windows of data for learning to predict outliers. The learning is a continuous process in which the sliding windows are used to train the LR. The target classes (y_i) for the sliding windows are obtained by the outlier detection. In our work, the training examples ($\{\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS} \cup \{\mathbf{y}_{i-PS+j-t}\}_{j=0}^{PS},\ y_i\}$) with the corresponding autoencoder features teach the classifier to recognize the patterns of data points, which indicate the occurrence of the outliers in t steps in the future. LR learns to predict outliers, because the training examples define a mapping of previously observed data points and the occurrence of outliers at a later point in time. The training examples are pairs of sliding windows and classification targets (y_i) where the classification targets originate from the future from the point of view of the sliding windows.

Algorithm 3 describes our proposed method for the outlier prediction, which requires an outlier detection algorithm (line 8) to provide the outlier labels (y) (line 9) for the training examples (line 10). Notice that the outlier prediction is not limited to only using our proposed algorithm for the outlier detection (Algorithm 2) to provide the outlier labels. Our work uses our proposed method for outlier detection (Algorithm 2) to provide the outlier labels (y) and the hidden representation as additional features ($\{\mathbf{y}_{i-\mathrm{PS}+j-t}\}_{j=0}^{PS}$). However, we want to emphasize that the utilization of the hidden representation is not required, if an autoencoder is not available (training examples are $\{\{\mathbf{x}_{i-\mathrm{PS}+j-t}\}_{j=0}^{PS}$).

Our work integrates Algorithms 2 and 3 to implement an unsupervised method for outlier detection and prediction. Instead of coupling the proposed Algorithms 2 and 3, and losing their generality, we introduce Algorithm 4. The algorithm integrates our proposed methods for outlier detection and outlier prediction and defines a single algorithm. Lines 1–15 in Algorithm 4 implement the outlier detection, which is described in Algorithm 2 (lines 1–14). The detected outliers and the observed data are used to create training examples for LR to predict the occurrence of outliers (line 16 and 17). Lines 16–19 implement the outlier prediction, which is described in Algorithm 3 (lines 10–12). The next subsection discusses the properties of our proposed methods.

3.4 Properties of the proposed methods

Our proposed methods for outlier detection (Algorithm 2) and outlier prediction (Algorithm 3) consume a stream by processing a data point ($\mathbf{x}_i$) at a time. The outliers in the context of our work are individual data points ($\mathbf{x}_i$), which do not resemble previously observed data points ($\mathbf{x}_j,\ j < i$). Therefore, the outliers are not unexpected patterns of multiple data points, but defined outlier points in the outlier detection. Outlier detection in the patterns of data points requires a more complex model, because the number of inputs increases, which increases the number of model parameters and the amount of cached data points. The scope of our work is to define an accurate and fast model for outlier detection using individual observations.

Our proposed method for outlier detection needs to be calibrated initially (Algorithm 1), because the method does not impose assumptions on the distribution of the analyzed data. The model needs to configure itself into an initial state before classifying the data points as outliers and inliers. In an optimal situation, the initial data points for calibration do not contain outliers. Using calibration data without outliers, the autoencoder learns a hidden representation of the inlier data. Outliers obtain a high value of the reconstruction cost, because the hidden representation is trained only for the inliers. This helps to distinguish the outliers from the inliers, because the outliers have distinctively larger values of the reconstruction cost than the inliers. The proposed method does not impose an assumption where inlier data is available for the calibration. However, the feature values of the initial data points are required to have a variance greater than zero. If the variance is zero for any feature, then the scaling limits of the outlier detection cannot be defined. We assume that the sensor data contains at least noise from the available sensors, which guarantees that the variance is not zero. This completes the first phase (see Chapter 3.2 and line 4 of Algorithm 1) of the calibration.

3.4.1 Updates of the autoencoder parameters

The magnitude of the gradient updates of the parameters of the autoencoder and LR are limited by a learning rate (Eq. 8). Therefore, the parameter updates, which are based on the possible outliers during the initial calibration phase, have a

Algorithm 3 The proposed algorithm for the outlier prediction in streams.

Input: Pattern size (PS), time step variable t
Output: Outlier prediction in t time steps in the future
1: Set $i = \mu_{\text{new}} = \mu_{\text{old}} = S_{\text{new}} = S_{\text{old}} = 0$
2: Initialize $\mathbf{x}_{\text{old}}$ with zeroes and $\mathbf{w}$ with random values between zero and one
3: Acquire a data point $\mathbf{x}_i$ from the stream
4: Go to (3) if $\mathbf{x}_i$ is identical with $\mathbf{x}_{\text{old}}$
5: Set $\mathbf{x}_{\text{old}} = \mathbf{x}_i$
6: Update the limits of the data features ($\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$)
7: Scale the feature values ($\mathbf{x}_{ij} \forall j$) using Eq. 15
8: Classify $\mathbf{x}_i$ as an outlier or an inlier using an algorithm for outlier detection (for instance, using Algorithm 2)
9: Store the outlier detection result in variable $y_i \in \{0, 1\}$
10: Create a training example $\{\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS} \cup \{\mathbf{y}_{i-PS+j-t}\}_{j=0}^{PS},\ y_i\}$ for LR, which is used as $\mathbf{x}_i$ in Eq. 8
11: Use SGD to update the parameters ($\mathbf{w}$) of the LR model given the training example (Eq. 8)
12: Predict if an outlier occurs in t time steps in the future using Eq. 2
13: Update $i = i + 1$ and go to (3)

Algorithm 4 An integration of the proposed algorithms for outlier detection and outlier prediction in streams.

Input: A predefined structure for the autoencoder, Pattern size (PS) and time step variable t
Output: Outlier classification for non-identical data points $\mathbf{x}_i$ in the stream and outlier prediction in t time steps in the future
1: Calibrate the outlier detection (execute Algorithm 1)
2: Initialize $\mathbf{x}_{\text{old}}$ with zeroes and $\mathbf{w}$ with random values between zero and one
3: Acquire a data point $\mathbf{x}_i$ from the stream
4: Go to (3) if $\mathbf{x}_i$ is identical with $\mathbf{x}_{\text{old}}$
5: Set $\mathbf{x}_{\text{old}} = \mathbf{x}_i$
6: **if** IS_OUTLIER($\mathbf{x}_i$) == *false* **then** ▷ IS_OUTLIER from Algorithm 2
7: Update the limits of the data features ($\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$)
8: **end if**
9: Set $y_i = 0$
10: **if** IS_OUTLIER($\mathbf{x}_i$) == *true* **then** ▷ IS_OUTLIER from Algorithm 2
11: The data point ($\mathbf{x}_i$) is an outlier
12: Set $y_i = 1$
13: **end if**
14: Update the autoencoder weights using SGD (Eq. 12–14)
15: Update the descriptive statistics of the distribution of the reconstruction cost (Eq. 17–19)
16: Compute the hidden representation ($\{\mathbf{y}_{i-PS+j-t}\}_{j=0}^{PS}$) for the data points in the sliding window ($\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS}$) using Eq. 9
17: Create a training example $\{\{\mathbf{x}_{i-PS+j-t}\}_{j=0}^{PS} \cup \{\mathbf{y}_{i-PS+j-t}\}_{j=0}^{PS},\ y_i\}$ for LR, which is used as $\mathbf{x}_i$ in Eq. 8. The hidden representation ($\{\mathbf{y}_i\}_{j=0}^{PS}$) is utilized for an additional source of features for the prediction.
18: Use SGD to update the parameters ($\mathbf{w}$) of the LR model given the training example (Eq. 8)
19: Predict if an outlier occurs in t time steps in the future
20: Update $i = i + 1$ and go to (3)

negligible influence, because the learning rate limits the magnitude of the gradient. Outliers are also rare by they definition [22], and therefore, their occurrence during the initial calibration phase can be assumed to have a low probability. The possible individual outlier-based updates are also addressed by the consecutive inlier-based updates, which adjust the autoencoder to represent the inlier data by rotating the gradient (Eq. 12–14) toward a low reconstruction cost for the inlier data.

Training the autoencoder is a non-convex optimization task where a global optimum cannot be guaranteed in the minimization of the reconstruction cost [58]. However, as discussed in Chapter 2.3, a reconstruction cost of zero is not desired, because the autoencoder should not model the background noise in the observed data. A local minimum of the reconstruction cost provides an estimate of the important information (which are encoded in the hidden representation [73]) of the data. The background noise is not important information in the data. Therefore the optimization for a local minimum is a compromise between a low reconstruction cost and not overfitting the data. The overfitting would result in an increased number of false positive detections of outliers, because the differences in the observed noise would be incorporated in the definition of an outlier. This is not a desired behavior, because two inliers with different noise are still inliers. The following subsection discusses properties of the distribution of the reconstruction cost, which is used to discriminate outliers from inliers (Algorithm 2, line 19).

3.4.2 Properties of the distribution of the reconstruction cost

Our method assumes that the mean and the variance for the unknown inlier reconstruction cost distribution are defined

and the estimates for these two parameters are continuously updated. Using Chebyshev's inequality and these estimated parameters (see Eq. 16) allows to discriminate reconstruction cost values, that have a low probability of originating from the inlier reconstruction distribution. The work in Bouguessa [15] models inlier and outlier scores using Beta distributions (see Bishop [12] for a definition). An outlier score is a value, which measures the magnitude of the deviation of a data point. Note that the definition of outlier score is not equivalent to the statistical definition of a score (see Hyvärinen [41]). Outlier score only measures the magnitude of how unexpected a data point is and, in our work, the reconstruction cost is an outlier score. The experiments in Bouguessa [15] show that a mixture of Beta distributions can model the scores: the beta component that corresponds to the highest score values for the outliers and the other components for the inliers. The data are more likely to originate from the inlier distribution, which is in line with our analysis. However, we impose fewer assumptions than in Bouguessa [15] by modeling the reconstruction costs using only the first two moments of the inlier reconstruction distribution.

Outliers may occur during the initial calibration of the outlier detection. If outliers occur during the initial calibration, then the values of the estimated statistics (μ, σ) of the reconstruction cost increase (compared to observing inliers). This increases the value of the threshold (Algorithm 2, line 19), which discriminates data points into outliers and inliers. However, the consecutive inliers decrease the values of the estimated statistics (μ, σ). Therefore, the values of the estimated statistics are expected to be increased temporarily, when an outlier is encountered. As an effect of the increased discrimination threshold, the detected outliers are more deviating with respect to the inliers. The outlier detection continues to function, but the detected outliers are temporarily the more obvious outliers. The following subsection discusses properties that apply for the outlier detection and prediction.

3.4.3 Outlier detection and prediction

If outliers are detected and predicted frequently, then the outliers are not rare anymore and the outliers become inliers. There is no external knowledge available for denoting changes in the characteristics of the inliers. The detection of the transformation between outliers and inliers is not in the scope of our work. The abundance of outliers may indicate, for example, of a change in the phenomenon measured by the sensors or a new configuration of the sensors. Our proposed method adapts to these potential changes, which modify the process and the distribution that generates the observed data points. The adaptation is provided by the continuous adjustment of parameters of the autoencoder and LR.

The outlier prediction requires correct results from the outlier detection, because the outlier prediction is trained to predict outliers, which are defined by an algorithm for outlier detection. Therefore, a limitation and a requirement for the outlier prediction is the availability of a reliable source of outlier detection.

3.5 Computational complexities

Both the outlier detection module (autoencoder) and outlier prediction module (logistic regression) use SGD to update the parameters within the module. SGD has shown to be applicable in the online setting and its time complexity, when updating parameters based on a single sample, is $\mathcal{O}(1)$, if we omit constant factors such as the data dimension d [14]. Generally, the time complexity for neural networks (including autoencoders) is constant w.r.t the number neurons. As only one hidden layer is used in the autoencoder, the number of neurons remains at a reasonable level. For the baseline algorithms, time complexities are connected to the window size ($\mathcal{O}(W)$ for STORM2 and $\mathcal{O}((1-c)Wlog((1-c)W) + kWlogk)$ for MCOD, see Tran et al. [81] for more details), which is typically much higher than the dimension of the data, making our approach clearly faster to compute.

The values that are stored in our framework are the autoencoder parameters, the weights for logistic regression, the two inlier distribution statistics and scaling limits for the features. In the experiments, in which the data dimension varied from 2 to 5, these resulted in 17–83 floating points being stored. The main factors affecting the space requirements of our method are the structure of the autoencoder and the data dimension. Compared to the more efficient baselines, the space complexities are $\mathcal{O}(W)$ for the STORM2 and $\mathcal{O}(cW + (1-c)kW)$ for MCOD [81], which are again connected to the window size. So given the structure of our autoencoder and the standard window sizes for the baselines (1000–10,000), our method also has significantly lower space requirements.

The next section surveys the related work in (1) outlier detection in streams and (2) outlier prediction. Our proposed methods are compared to the work in the literature. The benefits and the novelty of our methods are recapitulated in the end of the next section.

4 Related work

4.1 Outlier detection in streams

In the context of analyzing sensor data, our proposed method for outlier detection has three major benefits:

1. Our proposed method detects outliers with a good accuracy. The accuracy is evaluated in the experiments in Chapter 5.
2. Our proposed method uses a low amount of memory. The low memory usage is discussed with respect to existing algorithms of the literature in this section.
3. Our proposed method is fast to compute. The fast computation is discussed with respect to the literature in this section. The computation speed is discussed also in Chapter 5 with respect to baseline algorithms in the evaluation experiments.

To define a computationally lightweight model for outlier detection, our proposed method does not store the observations from an analyzed stream. Our proposed outlier detection method does not utilize sampling (as in Mai et al. [56], Aggarwal et al. [2]) or a sliding window (as in Angiulli and Fassetti [3], Kontaki et al. [48], Aggarwal [1], Cao et al. [20], Yang et al. [90], Assent et al. [6], Subramaniam et al. [75], Wu and Ma [88], Franke and Gertz [36], Tao and Pi [76]). The utilization of sampling and a sliding window (1) requires the storing of observations from a stream for further analysis and (2) requires the determination of the parameter values for the storing of the values (e.g., the number of stored samples). Signal processing methods that have been used for outlier detection include autoregressive models and signal filtering [39]. These methods are typically computed from a set of observations [7,39]. Therefore, sampling, sliding windows and signal processing are not utilized in our work for outlier detection. By not storing the observations, our proposed method saves computation time and memory for other tasks in resource-limited environments.

The approaches of detecting outliers in the streams include distance-based outlier detection [3,20,48,90], distribution-based outlier detection [75], clustering-based outlier detection [6], sampling-based outlier detection [2,56] and supervised learning-based outlier detection [1]. Our proposed method for outlier detection is distance-based, because the reconstruction cost (L) is a distance of the input data ($\mathbf{x}$) to its reconstruction ($\mathbf{z}$). Therefore we focus to survey and compare distance-based work in outlier detection in streams. Many of the distance-based work [3,20,48,90] for outlier detection in streams use the definition of Knorr et al. [47] of an outlier. An outlier is a data point ($\mathbf{x}$) with less than k data points within a distance T in the local neighborhood. A distance-based outlier is an isolated data point in the feature space with a high distance to the neighboring data points.

Angiulli and Fassetti [3] proposed three variants (STORM1, STORM2 and STORM3) of an outlier algorithm called Stream Outlier Miner (STORM) for distance-based outlier detection in streams. STORM1 provides exact results for the outlier detection if all of the active objects (data points in a sliding window) can be stored. STORM2 and STORM3 provide approximated results when the memory for storing the active objects is limited. The approximation is required when the size of the window (PS) is large, considering the capacity of available memory. The algorithms use a concept called safe inlier, which is an active object that is guaranteed to be an inlier until its expiration. A safe inlier $\mathbf{x}_i$ at time i has at least k data points ($\mathbf{x}_j$) in the local neighborhood. The data points ($\mathbf{x}_j$) arrive after $\mathbf{x}_i$, which satisfies the requirements of $j > i$ and $D(\mathbf{x}_i, \mathbf{x}_j) < T$ where D is a distance function. To determine a safe inlier, at least two data points ($\mathbf{x}_i, \mathbf{x}_j$, given $j > i$) are required. A data point ($\mathbf{x}_i$) is a safe inlier if $D(\mathbf{x}_i, \mathbf{x}_j) < T$ and $k = 1$. The use of safe inliers reduces the needed calculations, because the safe inliers will stay as inliers even when more data are acquired from the stream. STORM algorithms utilize the definition of a distance outlier by Knorr et al. [47]. The definition requires the determination of the nearest neighbors, which is a calculationally complex process. Our approach processes a data point at a time. Compared to the work of Angiulli and Fassetti [3], our method (1) does not determine local neighborhoods in outlier detection and (2) does not use a sliding window in outlier detection. These benefits result in a low calculational complexity, and there is no need to approximate the results or determine safe inliers.

Kontaki et al. [48] proposed three distance-based algorithms for detecting outliers in streams: Continuous Outlier Detection (COD), Advanced Continuous Outlier Detection (ACOD) and Micro-cluster-based Continuous Outlier Detection (MCOD). ACOD and MCOD are extensions of COD. COD detects outliers within a sliding window (as done by STORM1, STORM2 and STORM3). The motivation for COD in Kontaki et al. [48] is based on the observations that (1) an expiration can transform inliers into outliers and (2) new active objects can transform outliers into inliers. Similar to the work of Angiulli and Fassetti [3], COD uses local neighborhoods and the concept of safe inliers. The neighborhood of an expired inlier is checked for outliers, because inliers can turn into outliers. COD schedules these checks in advance in a priority queue, because the expiration time of an inlier is known in advance. If an inlier becomes a safe inlier, then it is removed from the queue. The actual outlier detection of COD is similar to the work of Angiulli and Fassetti [3] where the number of the neighbors of a data point determines the outlier status. A threshold for the number of neighbors discriminates the inliers from the outliers. The neighbors ($\mathbf{x}_j$) of a data point ($\mathbf{x}_i$) have a distance $D(\mathbf{x}_i, \mathbf{x}_j)$ less than a threshold T. The main difference between COD and STORM algorithms is that COD analyzes a smaller number of data points in the sliding windows [37]. Our proposed method for outlier detection processes a data point at a time. Therefore it (1) does not use a sliding window and (2) does not need a mechanism to alleviate the computational com-

plexity of determining the local neighborhood. The work of Angiulli and Fassetti [3], and Kontaki et al. [48], analyze the data in a local scope (nearest neighbors). Our proposed method estimates the global distribution of the data.

As previously stated, the neighborhood queries of COD utilize a threshold (T) and the number of neighbors (k). ACOD is an extension of COD where multiple neighborhood queries are combined concurrently [37]. The outliers are detected using a range of parameter values (T, k). ACOD combines multiple queries to improve the results of the outlier detection. However, ACOD is computationally more complex than COD, because ACOD performs multiple neighborhood queries. MCOD is an extension of COD that alleviates the computational cost of COD and ACOD. MCOD maintains clusters of the inlier data, which are used for the neighborhood queries [37]. The number of clusters is smaller than the number of data points within the sliding windows. The utilization of the clusters reduces the number of required computations. MCOD is a computationally less complex version of COD.

Cao et al. [20] state that the work in Kontaki et al. [48] and Angiulli and Fassetti [3] suffers from a considerable overhead in the execution of the outlier detection. The following observations by Cao et al. [20] help to alleviate the overhead:

- Outliers are rare by definition [22]. The computation should focus on inspecting the outlier candidates instead of determining the neighborhoods for the whole population. Our method proposed in this publication removes the need to determine the neighboring data points at all.
- Time should be used as a contextual attribute in a stream to process the data in an order. The time-aware processing speeds up the outlier detection in the stream since safe inliers can be ignored. The benefits of using the recent data are also addressed in [90].

The proposed solution in Cao et al. [20] counts the number of the neighbors only when necessary. The recent observations are defined to be more important than the previous observations, because the recent observations will be retained in the local neighborhoods longer than the previous ones. Inliers are identified as observations with long neighborhood memberships. The approach in Cao et al. [20] gathers a required amount of evidence for inliers and avoids the complete neighborhood searches (as done in Kontaki et al. [48] and Angiulli and Fassetti [3]). An active object is an inlier if it has the smallest possible evidence for being an inlier. The smallest evidence is a set of distances, which is called Minimal Evidence Set for Inlier (MESI). MESI contains the minimum number of distances in the local neighborhood for classifying an observation as an inlier. MESI typically contains a smaller number of the active objects than there are available in the complete neighborhood. The outliers are the active objects that do not satisfy MESI. Our work does not maintain the local neighborhoods of the data points, which (1) does not require the calculations of the intensive neighborhood searches and (2) does not suffer from the overhead of using a sliding window in outlier detection.

Siddiqui et al. [72] introduce an anomaly detection framework, where the feedback received from a human analyst is used to reduce the number of false positives among the detected anomalies or outliers. They show that the human feedback can guide the anomaly detector to give higher anomaly scores to the most important anomalies. The importance can depend on the application and the external information from the human analyst can help the anomaly detector to recognize application-specific anomalies. While their work incorporates external human feedback efficiently, our work focuses more on the case in which adopting external information or feedback from a human analyst is difficult or not even possible. Pang et al. [61] focuses on combining learning expressive low-dimensional representations of ultrahigh-dimensional data and outlier detection. Their main idea is to learn optimal representation for a specific outlier detector. Our work does not focus on ultrahigh-dimensional data, but the ideas presented in Pang et al. [61] could be utilized to our work if the input data stream has very high dimensionality.

Ma et al. [55] used an autoencoder to detect outliers. The method trains an autoencoder with a large amount of training data by using parallelized SGD. The trained autoencoder is used to detect outliers in test data. The proposed system is used for static datasets. The training data are assumed to define the probability distribution of the normal data. Sakurada and Yairi [67] proposed a similar method of using an autoencoder for outlier detection. Unlike the approach by Ma et al. [55], Sakurada and Yairi [67] use SGD instead of parallelized SGD. These previous works [55,67] with autoencoders for outlier detection do not define or classify an arbitrary data point ($\mathbf{x}_i$) as an inlier or an outlier. Our work extends the use of the autoencoder for outlier detection by proposing a mechanism, which classifies an arbitrary data point ($\mathbf{x}_i$) as an inlier or an outlier. This is important, because the outlier detection in streams is often integrated with time-critical mechanisms [1]. The work of Sakurada and Yairi [67], and Ma et al. [55], require the possession of data for defining the inliers. The data are not allowed to contain outliers. The requirement of defining the normal data in advance is not suitable for streams, because the distribution of the normal data may change as time passes. Obtaining the training data might be expensive or impossible [1]. Our work proposes an approach for using autoencoders to detect outliers in streams without using separate training data or test data.

Dong and Japkowicz [29] proposed an anomaly detection method for streaming data using an ensemble of autoencoders. However, the threshold for classifying outliers is

decided during training time and the method also requires the possession of outlier-free training data. The reconstruction cost distribution of the normal data may change as time passes and the threshold value set during training time does not take this into consideration. Chen et al. [23] also use an ensemble of autoencoders to perform anomaly detection for static datasets. They utilize randomized connection dropping between neurons to introduce diversity among autoencoders and also use adaptive data sampling during training to reduce the computational cost. Lu et al. [53] propose sequential outlier detection for time series data by combining denoising autoencoder and recurrent neural networks, but the model is trained using static outlier-free training data.

4.2 Outlier prediction

The previous work in the outlier prediction includes the use of regression [46,80], inferred rules [83], genetic algorithm [87] and artificial neural network [35]. Most of the previous work has been based on regression [35] in which the outliers are modeled as extreme values of the dependent variable (response variable, explained variable). Our work does not limit the definition of an outlier to be an extreme value. Multivariate data may have outliers, which are not defined as observations with extreme values. These types of outliers are detected when the variables contain unlikely combinations of the values. The proposed approach in this article detects and predicts outliers that are (1) extreme values and (2) unlikely combinations of the observed values. Our method also uses the hidden representation of an autoencoder to provide more information for the outlier prediction. To our knowledge, the publication is the first to utilize a hidden representation for outlier prediction.

The work in Torgo and Ribeiro [80] studied the prediction of outliers using standard regression trees with custom splitting criteria. The outliers are predicted in a static dataset. Our work extends the outlier prediction for streams. A regression tree is a supervised method, which poses challenges in outlier detection. The regression in Torgo and Ribeiro [80] maps a set of inputs into an output. Many tasks in outlier detection do not have a dependent variable which is regressed, for example, in intrusion detection. Forcing a problem to have a dependent variable for a set of inputs is very restricting. Our solution does not have a dependent variable, because the analysis of outlier prediction is based on the data itself. The work in Torgo and Ribeiro [80] depends on a set of inputs and outputs while our work depends only on the set of the inputs. The outliers are defined as rare and extreme values. The approach of using a regression tree does not detect outliers which are deviating combinations of the variable values. Our solution uses an autoencoder for outlier detection in which both types of the outliers are detected. Additionally, our solution does not force the task of the outlier detection to be fitted for regression.

Vilalta and Ma [83] use sets of events to predict outliers in static datasets using a sliding window. The sets are constructed using an a priori algorithm, which creates frequent sets of observed events. The events in the sets are observed simultaneously. The frequent events are assumed to be correlated and to precede the outliers. The sets of the frequent events are extracted from the data. The feature space of the events is discrete, because there is a finite amount of defined events. Our work also uses the preceding data to predict the values of the future, but our work operates in a real-valued feature space. The predicted outliers are called target events. The target event is defined in advance to be an event from the collection of possible events. The target events are assumed low in numbers in comparison with the other events. The requirement of defining the outlier is very restricting, because an outlier is rare by definition. The supervised information for defining an outlier and selecting the target event is typically not available. Our approach is unsupervised and does not require such knowledge in advance.

Weiss and Hirsh [87] proposed a similar approach to the work of Vilalta and Ma [83] for predicting outliers in a discrete feature space. Instead of using the frequent item sets of Vilalta and Ma [83], Weiss and Hirsh [87] use a genetic algorithm to identify the patterns for predicting the target events. The target events, which are monitored by the system in Weiss and Hirsh [87], have to be selected in advance. This is a strict requirement, because outliers are rare. Therefore the required knowledge to define the important events might not exist. The benefits of our work compared to Weiss and Hirsh [87] are the same as in Vilalta and Ma [83]: (1) our method uses a real-valued feature space and can analyze complex phenomenon, (2) our method is unsupervised and does not require knowledge of the data in advance and (3) our method does not use an external source of information to define an outlier.

The following list recapitulates the novelty and the benefits of our work compared to the state of the art:

- To our knowledge, the publication is the first to use the combination of an autoencoder, SGD and LR to detect and predict outliers in streams.
- To our knowledge, the publication is the first to utilize a hidden representation provided by the autoencoder for outlier prediction. This hidden representation is learned by packing and reconstructing the original data. See line 16 of Algorithm 4.
- The detection of outliers is unsupervised, and it provides labeled data for the prediction algorithm, which means that no training data is required for defining the characteristics of an outlier. The training data for outliers are

rarely available [49,92], because outliers are rare by the definition [22].
- The outliers are detected and predicted within unbounded streams of data. The algorithms for outlier detection in static datasets are not designed to be used with data streams.
- Our work defines outliers as (1) extreme values and (2) values, which do not follow the observed correlation structure of the features. An autoencoder detects the both types of outliers as data points with a high reconstruction cost.
- The proposed method includes a mechanism for the autoencoder to classify an arbitrary data point ($\mathbf{x}_i$) as an inlier or an outlier. For example, the work in Ma et al. [55] with autoencoders for outlier detection in static datasets does not classify an arbitrary data point ($\mathbf{x}_i$) as an outlier or an inlier.
- Our proposed method does not maintain relationships of neighboring data points, which is a computationally complex process. Instead, the proposed method learns a hidden representation of the data and applies one data point at a time. Additionally, there is no need to search for a suitable distance metric (D) for defining the distance between two data points. Therefore, our proposed method has a very low computational complexity. Our proposed method can be executed within resource-limited environments, which include embedded systems.
- The outliers can be predicted further than one time step ($t \geq 1$) in the future, unlike in Fong et al. [35], Torgo and Ribeiro [80]. For outlier prediction, the experiments in the next section use a range of time step (t) values from one to ten ($t \in \{1, \ldots, 10\}$).

5 Experiments

This section defines and executes experiments to validate our proposed methods for outlier detection and the outlier prediction. The evaluation consists of two phases. The outlier detection is experimented and evaluated separately in the first phase. In the second phase, the integration of the outlier detection and outlier prediction is experimented and evaluated. Before delving into the details of the evaluation phases, the overall setup and the quality metrics of the evaluation process are introduced.

5.1 Baseline algorithms

Our proposed method for outlier detection is compared against STORM2, COD and MCOD, which are established algorithms for outlier detection in streams. The selected algorithms were introduced in Chapter 4.1. STORM2 is selected over STORM, because STORM2 is computationally more efficient than STORM by providing approximated results. This is important, because the conducted experiments, which are defined later in this section, execute the algorithms thousands of times. The computation of STORM is infeasible in comparison with STORM2. Like STORM2, STORM3 is an approximate version of STORM. However, the results of STORM3 are more approximated than the results of STORM2. Because of the heavier approximation, STORM3 is less accurate than STORM2. Our proposed method for outlier detection is compared to STORM2, because STORM2 provides a compromise between the approximation of the results and the computation time. Unlike COD and MCOD, ACOD is not experimented, because the computation time of ACOD is infeasible. The computation of ACOD and STORM both individually exceed months. This is not suitable for analyzing streams of sensor data. Therefore, ACOD and STORM are scoped out of the evaluation experiments.

The baseline algorithms (STORM2, COD and MCOD) are utilized by accessing their implementations in Massive Online Analysis (MOA, Bifet et al. [10]). MOA is a software platform for implementing and executing algorithms against data streams. The implementations of STORM2, COD and MCOD in MOA can be accessed programmatically using Java. We developed an evaluation environment for the algorithms. Therefore, our proposed algorithm and the baseline algorithms are evaluated using same data. This results in a comparable set of results.

5.2 Metrics

The metrics for evaluating the outlier detection are recall (REC), false positive rate (FPR), Receiver Operating Characteristics curve (ROC) and the area under ROC (AUROC). See Fawcett [30] for a detailed definition of the evaluation metrics. The recall is also known as true positive rate in the context of ROC. Let TP be the number of detected true positives, FP the number of detected false positives, TN the number of detected true negatives and FN the number of detected false negatives. Recall and false positive rate are defined as follows:

$$\mathrm{REC} = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN}), \tag{21}$$

$$\mathrm{FPR} = \mathrm{FP}/(\mathrm{FP} + \mathrm{TN}). \tag{22}$$

Recall (REC) is the fraction of relevant data in all of the acquired data, which is the percentage of the detected outliers. False positive rate (FPR) is the fraction of the detected false positives to the number of all negative (inlier) data points. Therefore, a good algorithm for outlier detection results in high REC and low FPR [30].

Algorithms for outlier detection typically utilize a threshold [3,20,47,48,90], which discriminates outliers from inliers. For our proposed method, the multiplier value of the standard

deviation in Eq. 16 defines a threshold for the discrimination of outliers and inliers. The compared baseline algorithms (STORM2, COD and MCOD) utilize a threshold (T), which determines the distance of deciding if two data points are neighbors with each other. A high value of the threshold makes the outlier detection very conservative: only the most deviating outliers are detected. This results in a low value of recall and false positive rate. An efficient algorithm for outlier detection attempts to minimize the false positive rate while maximizing the recall [30]. To study how efficient the algorithms are with different values of the threshold, we utilize ROC to summarize the resulting pairs of recall and false positive rate. ROC is a graph of the resulting REC (y-axis) and FPR (x-axis) values when the threshold is varied. For our proposed method, the number of standard deviations is experimented from 0.1 to 4.0 with increments of 0.1. For STORM2, COD and MCOD, the threshold (T) is experimented from 0.1 to 10.0 with increments of 0.1. The intervals $0.1-4.0$ and $0.1-10.0$ are chosen, because they covered the range of REC and FPR values (REC $\in$ [0, 1], FPR $\in$ [0, 1]) in the experiments. Therefore, we alter the classification mechanism of outliers and inliers by varying the threshold. As a result, the algorithms generate different value pairs of REC and FPR. Computing ROC performs a complete evaluation of the efficiency of the algorithms. See Fawcett [30] for a detailed tutorial on ROC. ROC is used to evaluate outlier detection in Tax and Duin [77], Kriegel et al. [50], Lee et al. [52]. The values for the remaining parameters (number of neighbors k, sliding window size W) of STORM2, COD and MCOD are selected from their original publications.

The resulting ROC graphs are not plotted, because the experiments generate hundreds of ROC graphs. Instead, the ROC graphs are quantified and reported as values between zero and one by computing the area under the ROC curve (AUROC). The value of AUROC encodes the efficiency of an outlier algorithm over a range of parameter values. A perfect algorithm acquires AUROC $= 1.0$ by detecting all of the outliers without false positives [30]. The worst possible algorithm acquires AUROC $= 0.0$ by intentionally misclassifying the data points [30]. Notice that AUROC $= 0.5$ is acquired by randomly guessing if a data point is an outlier or an inlier [30]. The general efficiency of the algorithms is determined by comparing the resulting values of AUROC. The algorithm with the highest value of AUROC is declared the best performing algorithm.

The metrics for the evaluation of the outlier prediction are precision (PRE) and recall (REC), which are used in Kriegel et al. [49], Wang et al. [85], Dokas et al. [28]. Precision is defined as follows:

$$\text{PRE} = \text{TP}/(\text{TP} + \text{FP}). \tag{23}$$

Precision measures the confidence of correctly classifying the outliers. A high precision means that the acquired results are reliable. A perfect method predicts all of the outliers without false positives (REC = PRE = 1.0). To alleviate the process of defining a good pair of precision and recall, a metric called F_1 score is used to integrate the precision and the recall into a single value. The F_1 score is defined as

$$F_1 = 2 * \frac{\text{PRE} * \text{REC}}{\text{PRE} + \text{REC}}, \tag{24}$$

which is the harmonic mean of the precision and recall. Harmonic mean, which is defined as $\frac{n}{\sum_{i=1}^{n} \mathbf{x}_i^{-1}}$, is more sensitive to extreme values than the arithmetic mean [58]. This is desired behavior, because we want simultaneously high values of the precision and the recall for the outlier prediction. The outlier prediction results are ranked using the F_1 score, which allows a meaningful comparison based on a single value. The following subsections describe and report the results of the experiments for evaluating the outlier detection and prediction.

5.3 Outlier detection

The outlier detection is evaluated using synthetic sensor data and real-world sensor data. The real-world sensor data is acquired using Simple Measurement and Actuation Profile (sMAP, Dawson-Haggerty et al. [26]). sMAP is a protocol specification for storing, organizing, and publishing large amounts of heterogeneous stream data. sMAP attempts to alleviate the problems that are encountered in handling a large amount of time series data.

5.3.1 Datasets

The real-world sensor data consists of readings of electric sensors from east passenger elevator in Cory Hall at University of California, Berkeley. The sensors measure the apparent power, the apparent power factor, the current and the reactive power. The stream contains four dimensions of positive real values. The evaluation dataset consists of 100,000 data points between the dates of February 16, 2011 and March 19, 2011. The dataset is extracted using a public front-end [25] for sMAP data. However, for the outlier detection, the 100,000 data points are divided into five sets of 20,000 data points, because the computation time was infeasible for STORM2, COD and MCOD to evaluate multiple experiments. The evaluation of the outlier prediction in Chapter 5.5 is experimented using all of the available (100,000) data points as one dataset.

The five sMAP datasets, which are denoted by **SMAP1**-**SMAP5** in the experiments, are time series data and the data points are dependent of the previous observations. Any pat-

tern, which consists of dependent data points, is lost if the data points are selected randomly from the sMAP data. To preserve the dependency between the observations, the data points are emitted in their original order in the sMAP data. The original sMAP stream has a sampling rate of 20 seconds but the experiment environment emits the data points as fast as the proposed system can process.

The sMAP data do not have labels, which denote outliers within the data. However, the labels are required for determining if an outlier algorithm detected an outlier or an inlier. The calculation of the quality metrics (REC, TPR, AUROC) requires the existence of the outlier labels. Therefore, to create outliers on demand, we swap the values of random feature pairs of a data point. The approach of swapping feature values (1) preserves the order of the data points in the real-world sensor data and (2) provides the evaluation an access to outliers in a deterministic manner. In the literature, feature swapping has been previously utilized to create data for model compression [19].

The synthetic data is sampled from a mixture of two multivariate Gaussian distributions. See Bishop [12] for a definition of a mixture model of multivariate Gaussian distributions. The multivariate Gaussian distribution is selected, because it is a distribution of maximum entropy given a mean and variance [12]. Therefore a minimum number of assumptions are imposed on the synthetic data. The first (second) multivariate Gaussian distribution represents the class of inlier (outlier) observations. The combination of these two observation distributions constitute to a mixture model (of two distributions). The approach of using multivariate Gaussian distributions for evaluation of outlier detection is utilized in Angiulli and Fassetti [3] and Papadimitriou et al. [63]. The synthetic datasets simulate sensor readings, which have specified mean values and a covariance structure. The covariance structure encodes the correlations between sensors where sensor readings depend on each other. The outlier algorithms have to detect two types of outliers in the sensor readings: (1) unexpected individual values and (2) unexpected simultaneous values between the sensors. Therefore, a method for outlier detection has to model the underlying marginal distributions and joint distribution of the sensor values. The following configurations of the mixture of two Gaussian distributions are experimented:

- **GAUSS1** The first and the second multivariate Gaussian distributions have the same mean values, which are sampled from a uniform distribution. The distributions have different covariance structures, which are randomly generated. The challenge for an outlier algorithm is to understand how the features vary together.
- **GAUSS2** The dataset uses the configuration of **GAUSS1** with different mean values, which are sampled from a uniform distribution. The task is less challenging for outlier detection, because the mean values are different.
- **GAUSS3** The dataset uses the configuration of **GAUSS1** with randomly scaled covariance structure. The covariance structure of every feature of the second (outlier) distribution is scaled with a random real value between -2.0 and 2.0. As in **GAUSS1**, the challenge for an outlier algorithm is to understand how the features vary together. However, the variation of the features is larger in **GAUSS3** than in **GAUSS1**, because the covariances in **GAUSS3** are doubled in extreme cases. The task for outlier detection is less challenging than in **GAUSS1** and **GAUSS2**, because the features can have a strong relationship with each other.
- **GAUSS4** The dataset uses the configuration of **GAUSS1** with randomly scaled covariance structure. The covariance structure of every feature of the second (outlier) distribution is scaled with a random real value between -4.0 and 4.0. The task for outlier detection using **GAUSS4** is similar as for **GAUSS3**. However, the covariances in **GAUSS4** can vary even more than in **GAUSS3**. Therefore the task for outlier detection in **GAUSS4** is easier than in **GAUSS1–GAUSS3**.

Notice that the Gaussian experiments are sorted by the expected challenge of the task of detecting an outlier. **GAUSS1** creates data where outliers resemble inliers and **GAUSS4** creates data where the outliers do not resemble the inliers as much as in **GAUSS1–GAUSS3**. The multivariate Gaussian experiments (**GAUSS1–GAUSS4**) are repeated with a number of features (d) from two to five ($d \in \{2, 3, 4, 5\}$). Therefore, the outlier algorithms are evaluated using 16 synthetic datasets and five real-world datasets (**SMAP1**-**SMAP5**). This results in a total of 21 types of evaluation datasets, which offer a thorough evaluation of the outlier detection algorithms.

5.3.2 Experimental setting

The experimental environment simulates a stream by emitting rows in their original order from a dataset to an evaluated outlier algorithm. The labels of the utilized datasets allow a controlled selection of inliers and outliers. The experiment environment emits an inlier or outlier at will and observes the result of the outlier detection. The values of TP, FP, TN and FN are updated after observing the result of the outlier detection. The values of the metrics (Eq. 22 and 21) are recalculated. Determining the values of FPR and REC is possible, because it is known when outliers and inliers are emitted.

The outliers are emitted with 3% probability and the inliers are emitted with 97% probability. Therefore the outliers form a group of unexpected data points, which is the definition of an outlier [22]. A total of twenty thousand records emitted

to simulate a stream. The synthetic sensor data and the real-world sensor data contain from two to five features.

5.3.3 Parameter setting

The autoencoder employs from two to five weight values per a hidden neuron according to features. We do not want the autoencoder to overfit the data. Therefore the autoencoder is experimented with a low number of hidden neurons: between two (4–10 hidden weight values) and ten (20–50 hidden weight values). However, to reduce the computation time of the experiments, the autoencoder is experimented with an even number of hidden neurons ($h \in 2, 4, 6, 8, 10$). The resulting AUROC of each autoencoder configurations ($h \in 2, 4, 6, 8, 10$) is reported before reporting and comparing AUROC of STORM2, COD and MCOD.

STORM2 uses a sliding window size of $W = 10{,}000$ and the number of neighbors $k = 30$, which were used for the experiments in Angiulli and Fassetti [3]. MCOD uses a sliding window size of $W = 10{,}000$ and the number of neighbors $k = 10$, which were used for the experiments in Kontaki et al. [48]. COD could not be experimented with $W = 10{,}000$, because the computation would have lasted for months, which is not a suitable for analysis of stream data. Therefore COD uses a sliding window size of $W = 1000$ in the experiments (instead of $W = 10{,}000$). However, for COD, the number of neighbors k was 10 as in the experiments in Kontaki et al. [48]. For completeness and to provide additional test results, we evaluate STORM2 and MCOD using a sliding window size of $W = 1000$. Notice that STORM2 and MCOD utilize more information ($W = 10{,}000$) compared to our proposed method ($W = 1$). Our proposed method allocates from 17 to 83 floating point values. Therefore our proposed method is faster to compute than the compared algorithms as the amount of processed information is considerably smaller. Our proposed method allocates memory from 0.17% ($\frac{17}{10{,}000}$) to 8.30% ($\frac{83}{1000}$) with respect to the memory utilized by the compared methods.

The evaluation experiments are repeated ten times to form a sampling distribution of AUROC values. The utilization of ten repetitions per experiment is a typical approach to evaluate learning algorithms for streams [11]. The generation of the evaluation data includes randomness (emission of outliers and inliers). Therefore, the mean value of the AUROC sampling distribution is reported for the comparison of the algorithms. Algorithm 5 describes the method for evaluating the outlier detection using our proposed method and the baseline algorithms. The data points in **SMAP1**–**SMAP5** datasets are emitted in their original order (line 6 in Algorithm 5), which retains the time dependency in the time series data. The data points for **GAUSS1**–**GAUSS4** are repeatedly sampled (line 6 in Algorithm 5) from the Gaussian distributions.

5.4 Results

Table 1 reports the outlier detection results of our proposed method using synthetic data (**GAUSS1**–**GAUSS4**) and real-world data (**SMAP1**–**SMAP5**). The number of hidden neurons was varied to provide an extensive evaluation of our proposed method. The cells denote the average AUROC value of ten repetitions. The results show that the dataset types (**GAUSS1**–**GAUSS4**) are ordered by the challenge of the outlier detection task. This is evident as the average AUROC increases consistently between **GAUSS1** and **GAUSS4** dataset types. Based on the evaluation results, the outlier detection is more effective when the data contains multiple features. The autoencoder learns a hidden representation of the data, which encodes the relationships between the original features in a compact form. The utilization of the hidden representation is an effective approach for outlier detection, because the hidden representation models how the original features vary together. All of the average AUROC values were above 0.5. Therefore, our proposed method for outlier detection succeeded to deterministically classify outliers from inliers. For the dataset type **GAUSS4** with multiple features, the results are near of a perfect method for outlier detection.

Using the real-world data, the number of hidden neurons did not change the values of AUROC significantly. The results imply that the autoencoders did not overfit the real-world data by maintaining a hidden representation of the sensor data. Therefore, the possibly complicated interactions between the sensor values can be modeled using a various number of hidden neurons. The average AUROC of all results was above 0.8. Our proposed method is capable of detecting outliers in real-world data.

The previous results show that our proposed method for outlier detection is effective using a various number of hidden neurons. This is especially evident using the real-world data in which the results do not significantly change with the number of hidden neurons. The autoencoder implemented a successful dimensionality reduction, which learned h features (hidden representation) that represent the data. The features spanned a vector space, which contained the most of the variance of the data. The features are nonlinear transformations (Eq. 1) of linear combinations of the original features. As a comparison, consider principal component analysis (PCA), which is a widely used method for dimensionality reduction [12]. PCA models the data using orthogonal linear combinations of the features. Unlike PCA, an autoencoder with nonlinear transformations can model nonlinear relationships between the features. The nonlinearity of an autoencoder becomes evident when the autoencoder uses multiple layers of hidden, nonlinear neurons.

Figure 3 is a histogram of the number of calibration rounds (Algorithm 1) for our proposed method in the experiments.

Algorithm 5 The evaluation algorithm for an outlier detection algorithm in streams.

1: Read a dataset
2: Let k_1 be the number of repetitions to form a sampling distribution of AUROC and k_2 be the number of data points in the simulated stream
3: **for** $k_1 = 1; k_1 \leq 10$ **do**
4: **for** $T \in \{0.1, 0.2, \ldots, 4.0\}$ for our proposed method and $T \in \{0.1, 0.2, \ldots, 10.0\}$ for COD/MCOD/STORM2 **do**
5: Install the threshold: T for COD/MCOD/STORM2 and T for the number of standard deviations in Eq. 16 for our proposed method
6: **for** $k_2 = 1; k_2 \leq 20000$ **do**
7: Sample a random value (v), which is a random value, from the uniform distribution ($v \sim \mathbb{U}(0, 1)$)
8: Sample an inlier (**x**) if $v > 0.03$
9: Sample an outlier (**x**) if $v <= 0.03$
10: Process **x** using the experimented outlier algorithm: our proposed method (Algorithm 2), COD, MCOD or STORM2
11: Classify **x** as an inlier or as an outlier
12: **end for**
13: **end for**
14: Calculate the resulting AUROC
15: **end for**
16: Form the sampling distribution of AUROC

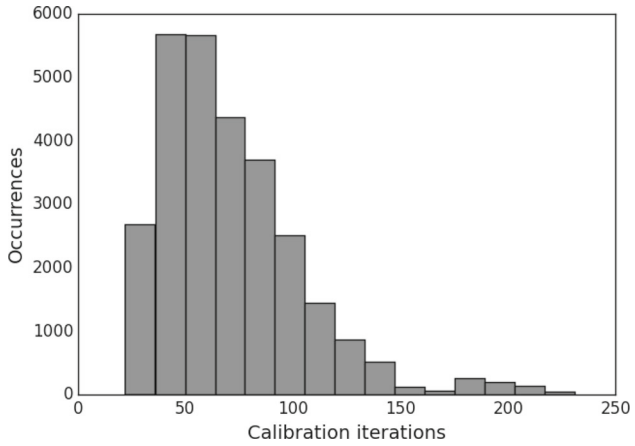

Fig. 3 Histogram of the calibration iterations (Algorithm 1) of our proposed method in the experiments. On average, 72 calibration iterations were carried in all of the experiments. The maximum bound (M) for the number of calibration rounds was $M = 10{,}000$ in the experiments. Therefore, the maximum bound is very conservative as discussed in Chapter 3.2.3

In the experiments, 72 calibration iterations were carried on average. The maximum number of calibration rounds was 231 and the minimum number of calibration rounds was 22. Therefore, as we discussed in Chapter 3.2.3, the bound (M) of maximum calibration rounds in Eq. 20 is very conservative. The observed results support our reasoning of the conservative nature of the maximum bound. The maximum bound in our experiments was $M = 10{,}000$, which is a much larger value than the observed maximum of 231 calibration rounds in our experiments.

The evaluation results in Table 1 show that our proposed method is efficient with variety of configurations. For a complete evaluation of our proposed method, we compare the experiment results of STORM2, COD and MCOD to the experiment results of our proposed method. For our proposed method, the results using two hidden neurons ($h = 2$) and ten hidden neurons ($h = 10$) are reported in the comparison. These two configurations represent a simple configuration ($h = 2$) and a complex configuration ($h = 10$) of our proposed method in the experiments.

Table 2 reports the evaluation results for STORM2, COD and MCOD using a sliding window with 1000 elements. Our proposed method detected outliers with better accuracy or comparable accuracy to those of STORM2, COD and MCOD. However, our proposed method outperformed STORM2, COD and MCOD using **GAUSS1** and **GAUSS2**, which were the most challenging datasets of the synthetic datasets. Our proposed method utilizes less information ($W = 1$) than STORM2, COD and MCOD ($W = 1000$). Therefore, our proposed method implements a deterministic method for detecting outliers that (1) is more effective than random guessing ($AUROC > 0.5$) [30] and (2) allocates a minimal amount of memory (17 to 83 floating point values).

Outlier detection in synthetic data with multiple features was an easier task for STORM2, COD and MCOD. These methods provided slightly better results than our proposed method using datasets **GAUSS3**-**GAUSS4** (see Table 2). STORM2, COD and MCOD have 1000 data points available at any given time in the sliding window. With multiple features, the outliers in **GAUSS3** and **GAUSS4** are effectively detected by comparing the data points with each other using the nearest neighbor queries. Our proposed method has to determine the characteristics of outliers using one data point at a time. Therefore, our proposed method (1) is computationally much more efficient, because there is no need to maintain a sliding window and (2) outperforms or acquires comparable results to those of STORM2, COD and MCOD.

For the challenging datasets (**GAUSS1**,**GAUSS2**), our proposed method provided consistently better results than STORM2, COD and MCOD. The outliers in the datasets **GAUSS1** and **GAUSS2** are detected by modeling how the feature values vary together. COD, MCOD and STORM2 do not explicitly model the relationships between the fea-

Table 1 The evaluation results of our proposed method (AE) for outlier detection using synthetic datasets (**GAUSS1–GAUSS4**) and real-world sensor data (**SMAP1–SMAP5**)

	GAUSS1				**GAUSS2**			
	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
Panel A: synthetic datasets								
AE h = 2	0.61	0.61	0.65	0.62	**0.74**	0.69	0.76	0.72
AE h = 4	**0.62**	0.66	0.71	0.68	**0.74**	**0.72**	0.80	0.79
AE h = 6	**0.62**	**0.67**	**0.73**	0.70	0.73	**0.72**	**0.81**	0.80
AE h = 8	**0.62**	**0.67**	**0.73**	0.71	0.72	0.71	**0.81**	**0.81**
AE h = 10	**0.62**	**0.67**	**0.73**	**0.72**	0.72	0.71	**0.81**	**0.81**
	GAUSS3				**GAUSS4**			
	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
AE h = 2	**0.77**	0.78	0.85	0.88	**0.82**	**0.91**	**0.94**	0.95
AE h = 4	0.75	**0.81**	**0.86**	0.91	0.80	0.90	**0.94**	**0.96**
AE h = 6	0.73	0.80	**0.86**	**0.92**	0.80	0.89	0.93	**0.96**
AE h = 8	0.73	0.78	0.85	**0.92**	0.80	0.88	0.93	**0.96**
AE h = 10	0.72	0.78	0.85	0.91	0.79	0.87	0.92	0.95
	SMAP1		**SMAP2**		**SMAP3**	**SMAP4**		**SMAP5**
Panel B: real-world sensor data								
AE h = 2	0.88		**0.89**		**0.88**	0.88		**0.88**
AE h = 4	0.88		**0.89**		**0.88**	0.88		0.87
AE h = 6	**0.89**		**0.89**		**0.88**	**0.89**		0.87
AE h = 8	**0.89**		**0.89**		**0.88**	**0.89**		0.87
AE h = 10	**0.89**		**0.89**		**0.88**	0.88		0.88

The number of hidden neurons was varied ($h \in 2, 4, 6, 8, 10$) to provide an extensive evaluation of our proposed method. For the synthetic data, the number of data features (d) was varied ($d \in 2, 3, 4, 5$). The cells denote the average AUROC value of ten repetitions. The best individual results per dataset type are bolded

tures. COD, MCOD and STORM2 rely on nearest neighbor queries, which are based on distance measurements between the data points. However, the autoencoder in our proposed method for outlier detection does model the dependencies between the features and is able to detect the outliers.

For the real-world data (**SMAP1–SMAP5**), COD and MCOD very slightly outperformed our proposed method (see Table 2 for exact AUROC results). However, the differences are minimal and the benefit of our proposed method is in the computational efficiency, which makes our proposed method very fast to compute. Our proposed method outperformed STORM2. As previously discusses, our proposed method learns a hidden representation of the input features, unlike COD, MCOD and STORM2. By utilizing the hidden representation, our proposed model is capable of detecting outliers in various types of sensor data. The benefits of our unsupervised approach for the streams include:

- The approach uses a small amount of memory, because the data are processed for a data point at a time. The method is suitable for larger datasets.
- The method does not need separate training data, which are labeled in advance. The labeling of the training data is especially challenging, because the outliers are rare by definition. The proposed method relaxes this requirement.
- Our proposed model is computationally more efficient than the compared methods (COD, MCOD and STORM2), because our proposed method does not utilize a sliding window in outlier detection.

Table 3 reports the evaluation results for STORM2 and MCOD using a sliding window of 10,000 elements. The results of COD are not reported, because the computation would have lasted for months, which is not a suitable performance for analyzing streams of data. As in the experiment results in Table 2, the results of our proposed method ($h \in \{2, 10\}$) are included for comparison.

Out of the 16 evaluation experiments with synthetic data, our proposed method outperformed or acquired similar results to MCOD and STORM2 in seven evaluation experiments. As in the evaluation results in Table 2, the outliers are effectively detected by comparing the data points with each other. Our proposed method has only access to one data point at a time, which is not as informative as having 10,000 data points available. However, the computational complexity of

Table 2 Evaluation results of outlier detection for COD, MCOD and STORM2 using synthetic sensor datasets (**GAUSS1**–**GAUSS4**) and real-world sensor data (**SMAP1**–**SMAP5**)

	GAUSS1				**GAUSS2**			
	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
Panel A: synthetic datasets								
AE h = 2	0.61	0.61	0.65	0.62	0.74	0.69	0.76	0.72
AE h = 10	**0.62**	**0.67**	**0.73**	**0.72**	0.72	**0.71**	**0.81**	**0.81**
COD	0.61	0.63	0.68	0.67	**0.75**	0.70	0.79	0.80
MCOD	0.61	0.63	0.68	0.67	**0.75**	0.70	0.79	0.80
STORM2	0.58	0.58	0.62	0.62	0.74	0.66	0.74	0.74
	GAUSS3				**GAUSS4**			
	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
AE h = 2	0.77	0.78	0.85	0.88	0.82	0.91	0.94	0.95
AE h = 10	0.72	0.78	0.85	0.91	0.79	0.87	0.92	0.95
COD	**0.78**	**0.80**	**0.88**	**0.93**	**0.84**	**0.93**	**0.96**	**0.98**
MCOD	**0.78**	**0.80**	**0.88**	**0.93**	**0.84**	**0.93**	**0.96**	**0.98**
STORM2	0.76	0.75	0.85	0.90	0.83	0.92	0.95	0.97

	SMAP1	**SMAP2**	**SMAP3**	**SMAP4**	**SMAP5**
Panel B: real-world sensor data					
AE h = 2	0.88	0.89	0.88	0.88	0.88
AE h = 10	0.89	0.89	0.88	0.88	0.88
COD	**0.90**	**0.91**	**0.90**	**0.90**	**0.90**
MCOD	0.89	0.87	0.89	0.88	0.89
STORM2	0.87	0.88	0.86	0.86	0.86

The size of the sliding window for COD, MCOD and STORM2 was $W = 1000$. For comparison, the results of our proposed method (AE, hidden neurons $h \in \{2, 10\}$) are reported from Table 1. The cells denote the resulting average values of AUROC after ten repetitions. The best individual results per dataset type are bolded

MCOD and STORM2 is very high, because the sliding window contains 10,000 elements. With the challenging datasets (**GAUSS1**, **GAUSS2**), our proposed method slightly outperformed MCOD and STORM2, while they have 10,000 times more of information available. Our proposed method detects outliers effectively in different types of data. Similar results are visible in Table 2.

For real-world data, our proposed method outperformed the compared methods. Additionally, the complex computation MCOD and STORM2 impose restrictions on the latency of the results in outlier detection. The results using the real-world sensor data, and the interpretation of the results, are similar with the results in Table 2.

In the experiments in Table 2, the sliding window size was 1000 for COD, MCOD and STORM2. In the experiments in Table 3, the sliding window size was 10,000 for MCOD and STORM2. The baseline algorithms stored at least 10,000 data points, which were updated and analyzed every time when a new data point arrived. The analysis using the baseline algorithms (COD, MCOD and STORM2) is a computationally demanding process. The long processing time should be justified by a high accuracy for outlier detection. However, the experiment results show that the baseline algorithms provide comparable or slightly better results than of our proposed method. Our proposed method for outlier detection provides accurate results consistently using a range of parameter values. Therefore, for analyzing streams of sensor data, our proposed method is capable of providing accurate results with a computationally lightweight model. Our proposed method, unlike MCOD, COD and STORM2, (1) does not store data points and (2) is fast to evaluate, because nearest neighbor queries are not used. Nearest neighbor queries are computationally demanding operations, and approximated solutions are sometimes preferred (see [5,42]).

The previous experiments (Tables 1, 2, 3) show that our proposed method (Algorithm 2) detects outliers efficiently. The experiments also showed that the optimal window size for the baseline algorithms (STORM2, COD and MCOD), when performance metrics are considered, is dataset dependent. The two different window sizes used (1000 and 10,000) give a general overview on how the window size may affect performance in the baseline algorithms compared to our work. When both window sizes are considered, our proposed method provided significantly better results for real-world sensor data and synthetic sensor data with a low number of dimensions. For outlier prediction, an accurate detection

Table 3 Evaluation results of outlier detection for MCOD and STORM2 using synthetic sensor datasets (**GAUSS1–GAUSS4**) and real-world sensor data (**SMAP1–SMAP5**)

	GAUSS1				**GAUSS2**			
	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
Panel A: synthetic datasets								
AE h = 2	0.61	0.61	0.65	0.62	**0.74**	0.69	0.76	0.72
AE h = 10	**0.62**	**0.67**	**0.73**	**0.72**	0.72	0.71	**0.81**	0.81
MCOD	0.61	0.66	**0.73**	**0.72**	0.69	**0.72**	**0.81**	**0.84**
STORM2	0.60	0.64	0.69	0.68	0.73	0.71	0.79	0.81
	GAUSS3				**GAUSS4**			
	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
AE h = 2	**0.77**	0.78	0.85	0.88	0.82	0.91	0.94	0.95
AE h = 10	0.72	0.78	0.85	0.91	0.79	0.87	0.92	0.95
MCOD	0.75	**0.83**	**0.90**	**0.95**	0.83	**0.94**	**0.97**	**0.99**
STORM2	**0.77**	0.81	0.89	0.94	**0.84**	0.93	0.96	0.98
	SMAP1		**SMAP2**		**SMAP3**		**SMAP4**	**SMAP5**
Panel B: real-world sensor data								
AE h = 2	0.88		**0.89**		**0.88**		**0.88**	**0.88**
AE h = 10	**0.89**		**0.89**		**0.88**		**0.88**	**0.88**
MCOD	0.76		0.72		0.77		0.70	0.76
STORM2	0.88		0.83		**0.88**		0.81	0.86

The size of the sliding window for MCOD and STORM2 was $W = 10{,}000$. The results of COD are not reported for $W = 10{,}000$, because of a very long computation time. For comparison, the results of our proposed method (AE, hidden neurons $h \in \{2, 10\}$) are reported from Table 1. The cells denote the resulting average values of AUROC after ten repetitions. The best individual results per dataset type are bolded

of outliers is important, because the prediction is trained using the outlier detection. It is essential to train the outlier prediction to predict outliers and not inliers. The following subsection evaluates the integration of outlier detection and prediction.

5.5 Outlier prediction

This section evaluates the outlier prediction using the real-world sensor data, which was acquired using sMAP and used in the evaluation of the outlier detection.

The literature does not have an unsupervised method for predicting the occurrence of outliers in streams. The proposed methods in Torgo and Ribeiro [80], King and Zeng [46], Vilalta and Ma [83], Weiss and Hirsh [87], Fong et al. [35] use separate training data or are used to analyze offline datasets, and are not applicable for analyzing streams of data in an online setting. Our proposed method for outlier prediction does not use separately annotated training data and provides analysis results with a low latency. A distinctive difference to the existing work is that our proposed method is a black box model, which analyzes a stream of sensor data without having knowledge of the analyzed data in advance. The literature does not have existing work available as a comparable baseline in the scope of our work, which is the unsupervised outlier prediction in streams of sensor data.

The experiment environment, as in the outlier detection, simulates a stream by continuously emitting rows from the extracted dataset. The outlier detection and prediction are continuously applied to the emitted data points. As in the evaluation of the outlier detection, the emitted data points are selected in the original order within the dataset. This preserves the dependence between two successive data points and therefore keeps time as an underlying element in the outlier prediction.

The experiments test the combinations of the following parameter values:

- The pattern size (PS) of the training examples for training LR: 1–10
- The number of time steps in the future (t) of the prediction: 1–10 time steps
- The number of the hidden neurons in the autoencoder for the outlier detection: 1–10

A specific combination of the parameter values is called a configuration. There is a total of 1000 different configurations, which evaluate the quality of the outlier prediction. Notice that the data have four features ($d = 4$): the apparent

power, the apparent power factor, the current and the reactive power. It is not reasonable to evaluate the autoencoder with the number of hidden neurons being significantly greater than the number of data features. The autoencoder would overfit the training examples. The number of tested configurations grows quickly, which requires a compromise between computation time and the test coverage. Adding a new parameter value to the current test coverage results in 100 new configurations. Therefore a subset of the configurations is evaluated exhaustively.

As with the evaluation of the outlier detection, we use ten repetitions to form sampling distributions of the precision, recall and F_1 score. The ten repetitions compute a total of 10,000 values of recall, precision and F_1 score. Algorithm 6 describes the method for evaluating the outlier prediction of a single configuration.

Algorithm 6 The evaluation algorithm for the outlier detection and prediction in streams.

1: Read a dataset and set a temporal variable $k_1 = 10$, which is the number of repetitions for forming the sampling distribution of the PRE, REC and F_1
2: **while** $k_1 > 0$ **do**
3: Use Algorithm 2 to detect the outliers in the dataset
4: **while** The dataset has unread rows **do**
5: Read a row from the dataset
6: Emit the row for the outlier detection and prediction (Algorithm 4)
7: **end while**
8: Calculate the resulting PRE, REC and F_1 for the outlier prediction
9: $k_1 = k_1 - 1$
10: **end while**
11: Form the sampling distribution of the PRE, REC and F_1 for the outlier prediction

Table 4 lists the ten configurations with the highest F_1 scores. The best configuration obtained a precision of 0.8190 and a recall of 0.5062 on average with the standard deviation being close to zero. The configuration was able to predict 50.62% of all outliers and the predictions were correct 81.90% of the time. The results are good, because over half of the present outliers are predicted correctly with a high confidence.

The best result in Weiss and Hirsh [87] for predicting outliers ($t = 1$) has a precision of 0.9000, a recall of 0.0044 and F_1 score of 0.0088. The proposed method in Weiss and Hirsh [87] is not designed for streams and uses a separate dataset for training the model. Therefore, the results of the model in Weiss and Hirsh [87] are not directly comparable, because the model uses separate training data. However, no previous results in outlier prediction in our scope are available. The results in Weiss and Hirsh [87] are compared to provide a rough indication of the significance of the results of our proposed method for outlier prediction. In our evaluation, a total of 100 configurations out of the 1000 configurations

Table 4 Mean and standard deviation of the sampling distributions of the precision and the recall for the outlier prediction

F_1 score	PRE	REC	PS	t	h
0.625 ± 0.025	0.819 ± 0.017	0.506 ± 0.031	5	1	4
0.624 ± 0.015	0.792 ± 0.009	0.515 ± 0.017	7	1	4
0.622 ± 0.033	0.791 ± 0.011	0.514 ± 0.046	4	1	9
0.620 ± 0.040	0.754 ± 0.018	0.528 ± 0.051	7	1	9
0.619 ± 0.024	0.849 ± 0.019	0.488 ± 0.029	3	2	4
0.619 ± 0.021	0.780 ± 0.019	0.513 ± 0.022	7	1	5
0.619 ± 0.022	0.815 ± 0.016	0.499 ± 0.027	4	1	6
0.618 ± 0.033	0.809 ± 0.015	0.502 ± 0.042	4	1	7
0.618 ± 0.021	0.844 ± 0.026	0.488 ± 0.025	3	2	5
0.618 ± 0.029	0.821 ± 0.019	0.495 ± 0.032	4	1	5

The top ten configurations of the 1000 configurations are listed. Notice that the top configurations predict one time step in the future ($t = 1$)

predicted the outliers in the immediate future ($t = 1$, as in Weiss and Hirsh [87]). The maximum F_1 score of the 100 configurations for our proposed method was 0.6251 and the minimum was 0.2490. Each of the 100 configurations had a considerably higher F_1 score than the best result of 0.0088 in Weiss and Hirsh [87]. The remaining 900 configurations predicted further in the future ($t > 1$) and are not compared with the work in Weiss and Hirsh [87].

Appendix A provides the outlier prediction results using support vector machine. The evaluation experiment using support vector machine was identical with the evaluation experiment using LR. The results for outlier prediction using LR (Table 4) are better than the results of outlier prediction using support vector machine (Appendix A). LR also provides classification probabilities, which can be used as a confidence of the predictions. Support vector machine does not provide the classification probabilities without an explicit conversion [58].

Appendix B provides the outlier prediction results using perceptron. The evaluation experiment using perceptron was identical with the evaluation experiments using LR and support vector machine. The results for outlier prediction using LR (Table 4) and support vector machine (Appendix A) are better than the results of outlier prediction using perceptron (Appendix B). As with support vector machine, perceptron does not provide classification probabilities.

Figure 4 plots all the values of the precision and the recall, which were obtained in all the 1000 configurations. Many false positives were predicted if the time step was between five and ten. The explanation is that the data itself do not contain the information for predicting the outliers from five to ten time steps in the future. The data may also contain periodic patterns, which are not detected in the local neighborhood of the current location in the stream. The periodic patterns include sinusoidal curves with varying frequencies, which

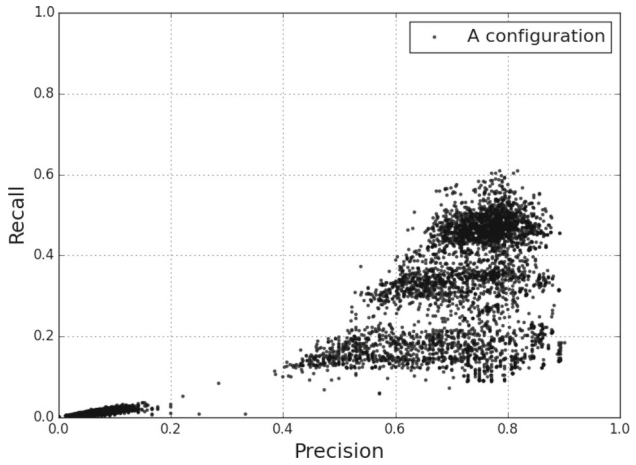

Fig. 4 All the resulted values of the precision and the recall, which were acquired in all the 1000 experiments

could be detected using Fourier transform. These patterns could offer additional information for the outlier prediction. Feature engineering should be applied in the future work for determining a set of variables, which would offer more information for the outlier detection and prediction. As an example, the concept of time (day, hour, etc.). could be incorporated in the time series measurements.

The configurations with the best results prefer to classify the outliers correctly by minimizing the number of false positives, which increases the precision and lowers the recall. The trade-off between the recall and precision can be adjusted by changing the number of standard deviations in the definition of an outlier in the line 19 of Algorithm 2. A small (high) number of standard deviations results in high (low) recall and low (high) precision.

5.6 Scalability test

We examine the scalability of our proposed method w.r.t. the data size and the dimensionality similarly to Pang et al. [60], Xu et al. [89], Jian et al. [44]. The tests are executed on the **GAUSS1** dataset, which is the most challenging dataset presented in this paper. First, we draw six datasets from the mixture of two multivariate Gaussian distribution with five dimensions to examine the scalability w.r.t. the data size. The smallest dataset consists of 6250 data points. Each successive dataset increases the data size by the factor of two until the largest dataset that consists of 200,000 data points. Then, we draw five more datasets from the mixture of multivariate Gaussian distribution to examine the scalability w.r.t. the data dimensionality. Each of these datasets consists of 6250 data points, while the number of dimensions is varied. The dataset with the lowest dimensionality consists of 250 dimensions. The dimensionality is increased in each successive dataset by the factor of two until the dataset with the largest dimensionality (with 4000 dimensions) is acquired.

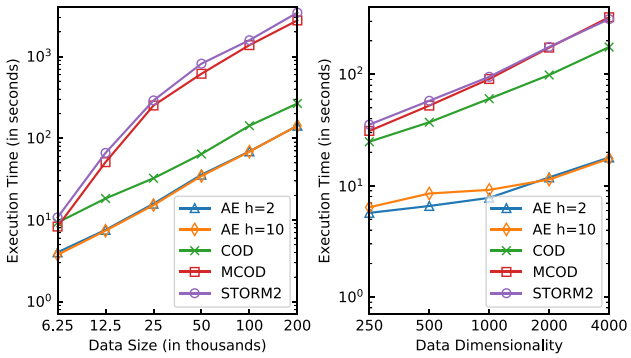

Fig. 5 Scalability test results

The results of the scalability test are illustrated in Fig. 5. The execution time of our proposed method is linear w.r.t. the data size. The execution time is linear for COD, too, and for MCOD and STORM2 with the larger datasets with more than 25000 data points (which is more than twice the window size $W = 10{,}000$ for both MCOD and STORM2). The execution times for all the methods (AE, COD, MCOD and STORM2) are quadratic w.r.t. the data dimensionality. Our proposed method has the fastest execution time out of all the tested methods w.r.t. both the data size and the dimensionality. The following section concludes this article by recapitulating our study and discussing about the possibilities of future work.

6 Conclusion

The prediction of the occurrence of outliers is a challenging task, because the outliers are rare by definition. The task is even more challenging with streams in which the data has to be processed immediately without having the option to store the data. We have proposed a novel approach for (1) detecting outliers in streams of sensor data and (2) predicting the occurrence of the outliers in streams of sensor data. In the context of analyzing streams of sensor data, the proposed methods (1) provide accurate results, (2) use a low amount of memory and (3) are fast to compute. The outlier detection is implemented by an autoencoder and the outlier prediction is implemented by LR. SGD is used to continuously and immediately train the models whenever data are available from the stream. The autoencoder learns a hidden representation of the data, which is utilized by LR for outlier prediction.

The novel methods for outlier detection and outlier prediction in streams of sensor data were evaluated using synthetic sensor data and real-world sensor data (electric readings). The experiments show that the outlier detection and prediction acquire good values of precision, recall and AUROC. For outlier detection, our proposed method outperforms the following algorithms found in the literature: STORM2, COD and MCOD. Our proposed method provides higher or com-

parable accuracy and has a significantly lower computation time.

Most of the previous work, which uses autoencoders to detect outliers [23,53,55,67], analyze static datasets and are not usable for streams of sensor data. Sequential outlier detection is proposed in Lu et al. [53], but static outlier-free training data is required, which makes their method unsuitable for streams of sensor data. Dong and Japkowicz [29] detect outliers using an ensemble of autoencoders from streaming data. Their detection threshold is empirically set during training time. However, the reconstruction cost distribution of normal data may change as time passes. We provide an adaptive mechanism for classifying outliers into inliers, by allowing the reconstruction cost distribution of inliers to change as time passes. We also provide a calibration algorithm for our proposed method for outlier detection in streams and an analysis of the properties of our proposed method for outlier detection.

The experiments also show that our proposed method for outlier prediction is capable of predicting the occurrence of outliers. However, an exact baseline is not available for our proposed method of outlier prediction in the literature, because our scope is in the unsupervised prediction of outliers in a black box approach. This scope has not been studied in the literature. However, for outlier prediction, LR provides better results (Table 4) than support vector machine (Appendix A) and perceptron (Appendix B). The topics of the future research and implementation work include:

- The experiments of using of stacked autoencoders [84] in the outlier detection, which are autoencoders with multiple layers of hidden neurons.
- The experiments of using denoising autoencoders [84] in the outlier detection, which are autoencoders that corrupt the input data for learning abstract and robust features.
- The evaluation of various (including nonlinear) classifiers for outlier prediction in an online setting.
- Apply the continuous outlier detection and prediction in a variety of application domains.
- Study the effects of using an adaptive learning rate (α) for SGD and an adaptive value of γ in Eqs. 17–18.

Acknowledgements Open access funding provided by Technical Research Centre of Finland (VTT). The authors wish to express their gratitude to Timo Lintonen from VTT, Technical Research Centre of Finland, for his assistance in Chapter 5.6 (Scalability test).

Compliance with ethical standards

Conflict of interest The author declares that there is no conflict of interest.

Appendix

A Outlier prediction results using support vector machine

See Table 5.

Table 5 Mean and standard deviation of the sampling distributions of the precision and the recall for the outlier prediction using support vector machine

F_1 score	PRE	REC	PS	t	h
0.618 ± 0.024	0.755 ± 0.023	0.523 ± 0.029	4	1	6
0.612 ± 0.018	0.770 ± 0.023	0.508 ± 0.017	4	1	4
0.611 ± 0.030	0.750 ± 0.015	0.516 ± 0.036	4	1	5
0.610 ± 0.045	0.733 ± 0.038	0.524 ± 0.052	4	1	10
0.609 ± 0.032	0.732 ± 0.024	0.522 ± 0.038	5	1	5
0.608 ± 0.017	0.783 ± 0.023	0.496 ± 0.014	4	1	3
0.605 ± 0.021	0.725 ± 0.021	0.519 ± 0.025	5	1	7
0.604 ± 0.010	0.780 ± 0.013	0.492 ± 0.008	4	1	2
0.603 ± 0.018	0.779 ± 0.027	0.492 ± 0.019	3	2	4
0.601 ± 0.023	0.735 ± 0.026	0.509 ± 0.032	5	1	6

The top ten configurations of the 1000 configurations are listed

B Outlier prediction results using Perceptron

See Table 6.

Table 6 Mean and standard deviation of the sampling distributions of the precision and the recall for the outlier prediction using Perceptron

F_1 score	PRE	REC	PS	t	h
0.595 ± 0.017	0.649 ± 0.016	0.549 ± 0.024	4	1	4
0.594 ± 0.038	0.642 ± 0.033	0.553 ± 0.043	4	1	6
0.587 ± 0.046	0.612 ± 0.047	0.565 ± 0.051	4	1	9
0.582 ± 0.030	0.636 ± 0.034	0.537 ± 0.031	4	1	5
0.578 ± 0.025	0.623 ± 0.034	0.538 ± 0.023	4	1	7
0.576 ± 0.040	0.619 ± 0.030	0.540 ± 0.054	5	1	8
0.571 ± 0.024	0.619 ± 0.018	0.531 ± 0.032	5	1	5
0.569 ± 0.033	0.605 ± 0.030	0.539 ± 0.045	5	1	9
0.569 ± 0.028	0.614 ± 0.037	0.531 ± 0.024	4	1	8
0.566 ± 0.017	0.607 ± 0.013	0.531 ± 0.022	7	1	4

The top ten configurations of the 1000 configurations are listed. Notice that the top configurations predict one time step in the future ($t = 1$)

References

1. Aggarwal, C.: On abnormality detection in spuriously populated data streams. In: Proceedings of the ACM SIAM Conference on Data Mining, pp. 80–91 (2005)
2. Aggarwal, C., Zhao, Y., Yu, P.: Outlier detection in graph streams. In: Proceedings of the 27th International Conference on Data Engineering, pp. 399–409 (2011)
3. Angiulli, F., Fassetti, F.: Distance-based outlier queries in data streams: the novel task and algorithms. Data Min. Knowl. Dis. **20**(2), 290–324 (2010)
4. Angiulli, F., Pizzuti, C.: Fast outlier detection in high dimensional spaces. In: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery, Springer-Verlag, London, UK, PKDD '02, pp. 15–26 (2002)
5. Arya, S., Mount, D., Netanyahu, N., Silverman, R., Wu, A.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM **45**(6), 891–923 (1998)
6. Assent, I., Kranen, P., Baldauf, C., Seidl, T.: Anyout: Anytime outlier detection on streaming data. In: Proceedings of the 17th International Conference on Database Systems for Advanced Applications, Springer-Verlag, Berlin, Heidelberg, DASFAA'12, pp. 228–242 (2012)
7. Barber, D.: Bayesian Reasoning and Machine Learning. Cambridge University Press, New York (2012)
8. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol. 7700, Springer, Berlin, pp. 437–478 (2012)
9. Bengio, Y., Yao, L., Alain, G., Vincent P.: Generalized Denoising Auto-Encoders as Generative Models. Advances in Neural Information Processing Systems (2013)
10. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
11. Bifet, A., de Francisci Morales, G., Read, J., Holmes, G., Pfahringer, B.: Efficient online evaluation of big data stream classifiers. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, pp. 59–68 (2015)
12. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Secaucus (2006)
13. Bottou, L.: Online learning and stochastic approximations. On-Line Learn. Neural Netw. **17**(9), 142 (1998)
14. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer, pp. 177–186 (2010)
15. Bouguessa, M.: Modeling outlier score distributions. In: International Conference on Advanced Data Mining and Applications. Springer, pp. 713–725 (2012)
16. Braimah, O., Osanaiye, P., Omaku, P., Saheed, Y., Eshimokhai, S.: On the use of exponentially weighted moving average (ewma) control chart in monitoring road traffic crashes. Int. J. Math. Stat. Invent. (IJMSI) **2**(5), 01–09 (2014)
17. Breunig, M., Kriegel, H.P., Ng, R., Sander, J.: LOF: identifying density-based local outliers. In: Proceedings of the International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '00, pp. 93–104 (2000)
18. Bruno, G., Garza, P.: TOD: temporal outlier detection by using quasi-functional temporal dependencies. Data Knowl. Eng. **69**(6), 619–639 (2010)
19. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA. KDD '06, pp. 535–541 (2006)
20. Cao, L., Yang, D., Wang, Q., Yu, Y., Wang, J., Rundensteiner, E.: Scalable distance-based outlier detection over high-volume data streams. In: Proceedings of the 30th International Conference on Data Engineering, ICDE, pp. 76–87 (2014)
21. Carcillo, F., Le Borgne, Y.A., Caelen, O., Bontempi, G.: Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization. Int. J. Data Sci. Anal. **5**(4), 285–300 (2018)
22. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009)
23. Chen, J., Sathe, S., Aggarwal, C., Turaga, D.: Outlier detection with autoencoder ensembles. In: Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, pp. 90–98 (2017)
24. Cho, K., Raiko, T., Ilin, A.: Enhanced gradient and adaptive learning rate for training restricted boltzmann machines. In: Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28–July 2, 2011, pp. 105–112 (2011)
25. Dawson-Haggerty, S.: sMAP 2.0 Plotting Engine. http://www.openbms.org/plot/ (2015). Accessed 2 Feb 2015
26. Dawson-Haggerty, S., Jiang, X., Tolle, G., Ortiz, J., Culler, D.: sMAP: a simple measurement and actuation profile for physical information. In: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, ACM, New York, NY, USA, SenSys '10, pp. 197–210 (2010)
27. De Stefani, J., Le Borgne, Y.A., Caelen, O., Hattab, D., Bontempi, G.: Batch and incremental dynamic factor machine learning for multivariate and multi-step-ahead forecasting. Int. J. Data Sci. Anal. (2018). https://doi.org/10.1007/s41060-018-0150-x
28. Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., Tan, P.: Data mining for network intrusion detection. In: Proceedings of the NSF workshop on next generation data mining, pp. 21–30 (2002)
29. Dong, Y., Japkowicz, N.: Threaded ensembles of autoencoders forstream learning. Comput. Intell. **34**(1), 261–281 (2018). https://doi.org/10.1111/coin.12146
30. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
31. Ferdowsi, H., Jagannathan, S., Zawodniok, M.: An online outlier identification and removal scheme for improving fault detection performance. IEEE Trans. Neural Netw. Learn. Syst. **25**(5), 908–919 (2014)
32. Fileto, R., May, C., Renso, C., Pelekis, N., Klein, D., Theodoridis, Y.: The Baquara2 knowledge-based framework for semantic enrichment and analysis of movement data. Data Knowl. Eng. **98**, 104–122 (2015)
33. Finch, T.: Incremental calculation of weighted mean and variance. Technical report, University of Cambridge (2009)
34. Folino, F., Greco, G., Guzzo, A., Pontieri, L.: Mining usage scenarios in business processes: outlier-aware discovery and run-time prediction. Data Knowl. Eng. **70**(12), 1005–1029 (2011)
35. Fong, S., Nannan, Z., Wong, R., Yang, X.: Rare events forecasting using a residual-feedback GMDH neural network. In: Proceedings of the 2012 IEEE 12th International Conference on Data Mining Workshops, ICDMW, pp. 464–473 (2012)
36. Franke, C., Gertz, M.: Detection and exploration of outlier regions in sensor data streams. In: Proceedings of the IEEE International Conference on Data Mining Workshops, ICDMW, pp. 375–384
37. Georgiadis, D., Kontaki, M., Gounaris, A., Papadopoulos, A., Tsichlas, K., Manolopoulos, Y.: Continuous outlier detection in data streams: an extensible framework and state-of-the-art algorithms. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '13, pp. 1061–1064 (2013)
38. Giacometti, A., Soulet, A.: Anytime algorithm for frequent pattern outlier detection. Int. J. Data Sci. Anal. **2**(3–4), 119–130 (2016)

39. Gupta, M., Gao, J., Aggarwal, C., Han, J.: Outlier detection for temporal data: a survey. IEEE Trans. Knowl. Data Eng. **26**(9), 2250–2267 (2014)
40. Hawkings, D.: Identification of Outliers. Chapman and Hall, London (1980)
41. Hyvärinen, A.: Estimation of non-normalized statistical models by score matching. J. Mach. Learn. Res. **6**, 695–709 (2005)
42. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, ACM, New York, NY, USA, STOC '98, pp. 604–613 (1998)
43. Janssen, J., Huszar, F., Postma, E., van den Herik, E.: Stochastic outlier selection (2012)
44. Jian, S., Pang, G., Cao, L., Lu, K., Gao, H.: Cure: Flexible categorical data representation by hierarchical coupling learning. IEEE Trans. Knowl. Data Eng. **31**(5), 853–866 (2019). https://doi.org/10.1109/TKDE.2018.2848902
45. Kao, L., Huang, Y.: Association rules based algorithm for identifying outlier transactions in data stream. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 3209–3214 (2012)
46. King, G., Zeng, L.: Logistic regression in rare events data. Political Anal. **9**(2), 137–163 (2001)
47. Knorr, E., Ng, R., Tucakov, V.: Distance-based outliers: algorithms and applications. VLDB J. **8**(3–4), 237–253 (2000)
48. Kontaki, M., Gounaris, A., Papadopoulos, A., Tsichlas, K., Manolopoulos, Y.: Continuous monitoring of distance-based outliers over data streams. In: Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE), pp. 135–146 (2011)
49. Kriegel, H.P., Hubert, M., Zimek, A.: Angle-based outlier detection in high-dimensional data. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '08, pp. 444–452 (2008)
50. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Loop: local outlier probabilities. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management, ACM, New York, NY, USA, CIKM '09, pp. 1649–1652 (2009)
51. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Interpreting and unifying outlier scores. In: Proceedings of the SIAM International Conference on Data Mining, SIAM/Omnipress, SDM, pp. 13–24 (2011)
52. Lee, Y., Yeh, Y., Wang, Y.: Anomaly detection via online oversampling principal component analysis. IEEE Trans. Knowl. Data Eng. **25**(7), 1460–1470 (2013)
53. Lu, W., Cheng, Y., Xiao, C., Chang, S., Huang, S., Liang, B., Huang, T.: Unsupervised sequential outlier detection with deep architectures. IEEE Trans. Image Process. **26**(9), 4321–4330 (2017)
54. Lucas, J.M., Saccucci, M.S.: Exponentially weighted moving average control schemes: properties and enhancements. Technometrics **32**(1), 1–12 (1990)
55. Ma, Y., Zhang, P., Cao, Y., Guo, L.: Parallel auto-encoder for efficient outlier detection. In: Proceedings of the 2013 IEEE International Conference on Big Data, pp. 15–17 (2013)
56. Mai, J., Chuah, C., Sridharan, A., Ye, T., Zang, H.: Is sampled data sufficient for anomaly detection? In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, ACM, IMC '06, pp. 165–176 (2006)
57. Moniz, N., Branco, P., Torgo, L.: Resampling strategies for imbalanced time series forecasting. Int. J. Data Sci. Anal. **3**(3), 161–181 (2017). https://doi.org/10.1007/s41060-017-0044-3
58. Murphy, K.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
59. Nguyen, H., Ang, H., Gopalkrishnan, V.: Mining outliers with ensemble of heterogeneous detectors on random subspaces. In: Proceedings of the 15th International Conference on Database Systems for Advanced Applications, Springer-Verlag, Berlin, Heidelberg, DASFAA'10, pp. 368–383 (2010)
60. Pang, G., Xu, H., Cao, L., Zhao, W.: Selective value coupling learning for detecting outliers in high-dimensional categorical data. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, New York, NY, USA, CIKM '17, pp. 807–816. (2017). https://doi.org/10.1145/3132847.3132994
61. Pang, G., Cao, L., Chen, L., Liu, H.: xLearning representations of ultrahigh-dimensional data for random distance-based outlier detection. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, New York, NY, USA, KDD '18, pp. 2041–2050. (2017). https://doi.org/10.1145/3219819.3220042
62. Pang, L., Chawla, S., Liu, W., Zheng, Y.: On detection of emerging anomalous traffic patterns using GPS data. Data Knowl. Eng. **87**, 357–373 (2013)
63. Papadimitriou, S., Kitagawa, H., Gibbons, P., Faloutsos, C.: Loci: Fast outlier detection using the local correlation integral. In: Proceedings of the International Conference on Data Engineering, IEEE Computer Society, ICDE, pp. 315–326 (2003)
64. Prechelt, L.: Early stopping–but when? In: Montavon, G., Orr, G., Müller, K.R. (eds.) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 7700. Springer, New York (2012)
65. Rasheed, F., Alhajj, R.: A framework for periodic outlier pattern detection in time-series sequences. IEEE Trans. Cybern. **44**(5), 569–582 (2014)
66. Raza, H., Prasad, G., Li, Y.: EWMA based two-stage dataset shift-detection in non-stationary environments. In: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, pp. 625–635 (2013)
67. Sakurada, M., Yairi, T.: Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis, ACM, New York, NY, USA, MLSDA'14, pp. 4:4–4:11 (2014)
68. Sarasamma, S., Zhu, Q., Huff, J.: Hierarchical Kohonenen net for anomaly detection in network security. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **35**(2), 302–312 (2005)
69. Schubert, E., Weiler, M., Kriegel, H.: Signitrend: Scalable detection of emerging topics in textual streams by hashed significance thresholds. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '14, pp. 871–880 (2014)
70. Schuhknecht, F., Jindal, A., Dittrich, J.: The uncracked pieces in database cracking. Proc. VLDB Endow. **7**(2), 97–108 (2013)
71. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2003)
72. Siddiqui, M.A., Fern, A., Dietterich, T.G., Wright, R., Theriault, A., Archer, D.W.: Feedback-guided anomaly discovery via online optimization. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '18, pp. 2200–2209. (2018). https://doi.org/10.1145/3219819.3220083
73. Snoek, J., Adams, R., Larochelle, H.: Nonparametric guidance of autoencoder representations using label information. J. Mach. Learn. Res. **13**(1), 2567–2588 (2012)
74. Souiden, I., Brahmi, Z., Toumi, H.: A survey on outlier detection in the context of stream mining: review of existing approaches and recommadations. In: International Conference on Intelligent Systems Design and Applications, Springer, pp. 372–383 (2016)
75. Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Online outlier detection in sensor data using non-parametric models. In: Proceedings of the 32nd international conference on Very large data bases, pp. 187–198 (2006)

76. Tao, Y., Pi, D.: Unifying density-based clustering and outlier detection. In: Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining, WKDD, pp. 644–647 (2009)
77. Tax, D., Duin, R.: Support vector data description. Mach. Learn. **54**(1), 45–66 (2004)
78. Teffer, D., Srinivasan, R., Ghosh, J.: Adahash: hashing-based scalable, adaptive hierarchical clustering of streaming data on mapreduce frameworks. Int. J. Data Sci. Anal. (2018). https://doi.org/10.1007/s41060-018-0145-7
79. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 4th edn. Academic Press, London (2008)
80. Torgo, L., Ribeiro, R.: Predicting outliers. In: Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 447–458 (2003)
81. Tran, L., Fan, L., Shahabi, C.: Distance-based outlier detection in data streams. Proc. VLDB Endow. **9**(12), 1089–1100 (2016)
82. Trittenbach, H., Böhm, K.: Dimension-based subspace search for outlier detection. Int. J. Data Sci. Anal. **7**(2), 87–101 (2019). https://doi.org/10.1007/s41060-018-0137-7
83. Vilalta, R., Ma, S.: Predicting rare events in temporal domains. In: Proceedings of the IEEE International Conference On Data Mining, pp. 474–481 (2002)
84. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
85. Wang, G., Hao, J., Ma, J., Huang, L.: A new approach to intrusion detection using artificial neural networks and fuzzy clustering. Expert Syst. Appl. **37**(9), 6225–6232 (2010)
86. Wang, H., Liu, R.: Hiding outliers into crowd: privacy-preserving data publishing with outliers. Data Knowl. Eng. **100**(Part A), 94–115 (2015)
87. Weiss, G., Hirsh, H.: Learning to predict rare events in event sequences. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, pp. 359–363 (1998)
88. Wu, Q., Ma, S.: Detecting outliers in sliding window over categorical data streams. In: Proceedings of the 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD, vol. 3, pp. 1663–1667 (2011)
89. Xu, H., Wang, Y., Cheng, L., Wang, Y., Ma, X.: Exploring a high-quality outlying feature value set for noise-resilient outlier detection in categorical data. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, New York, NY, USA, CIKM '18, pp. 17–26. (2018). https://doi.org/10.1145/3269206.3271721
90. Yang, D., Rundensteiner, E., Ward, M.: Neighbor-based pattern detection for windows over streaming data. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM, New York, NY, USA, EDBT '09, pp. 529–540 (2009)
91. Zhang, H., Nian, K., Coleman, T.F., Li, Y.: Spectral ranking and unsupervised feature selection for point, collective, and contextual anomaly detection. Int. J. Data Sci. Anal. (2018). https://doi.org/10.1007/s41060-018-0161-7
92. Zhang, K., Hutter, M., Jin, H.: A new local distance-based outlier detection approach for scattered real-world data. Data Min. Knowl. Discov. (2009). https://doi.org/10.1007/978-3-642-01307-2_84
93. Zhang, Y., Meratnia, N., Havinga, P.: Outlier detection techniques for wireless sensor networks: a survey. IEEE Commun. Surv. Tutor. **12**(2), 159–170 (2010)
94. Zhao, Y., Lehman, B., Ball, R., Mosesian, J., de Palma, J.: Outlier detection rules for fault detection in solar photovoltaic arrays. In: Proceedings of the 2013 Twenty-Eighth Annual IEEE Applied Power Electronics Conference and Exposition (APEC), pp 2913–2920 (2013)