User-driven Error Detection for Time Series with Events

Kim-Hung Le Paolo Papotti
EURECOM, France – {Kim-Hung.Le,Paolo.Papotti}@eurecom.fr

***Abstract*—Anomalies are pervasive in time series data, such as sensor readings. Existing methods for anomaly detection cannot distinguish between *anomalies* that represent data errors, such as incorrect sensor readings, and notable *events*, such as the watering action in soil monitoring. In addition, the quality performance of such detection methods highly depends on the configuration parameters, which are dataset specific. In this work, we exploit active learning to detect both errors and events in a single solution that aims at minimizing user interaction. For this joint detection, we introduce an algorithm that accurately detects and labels anomalies with a non-parametric concept of neighborhood and probabilistic classification. Given a desired quality, the confidence of the classification is then used as termination condition for the active learning algorithm. Experiments on real and synthetic datasets demonstrate that our approach achieves F-score above 80% in detecting errors by labeling 2 to 5 points in one data series. We also show the superiority of our solution compared to the state-of-the-art approaches for anomaly detection. Finally, we demonstrate the positive impact of our error detection methods in downstream data repairing algorithms.**

I. Introduction

Anomaly detection is an important task in several domains, such as intrusion detection systems, financial fraud detection, and Internet of Thing (IoT). It has been estimated that such data has from 2.3% to 26.9% error rate [14]. Applications built upon imprecise time series can potentially result in losses in the millions of dollars to businesses. As an example, in forest fire detection sensors are deployed to monitor the concentration of carbon-monoxide and organic compounds. Potential hazards are detected by combining sensor data with weather information. Imprecise sensor data could significantly decrease the system reliability and impact remedial actions. The efficacy of such response systems highly depends on the performance of the anomaly detection algorithms [29].

Anomaly detection over time series is applied to filter out noise in the data. Unfortunately, such data may contain notable events, also known as *change points*. These changes occur by accident (e.g., a fire in a forest) or because of human intervention (e.g., watering a field). Preserving these events is essential for any scenario. For example, a company may deploy the sensors to monitor the impact of watering on soil humidity. However, a significant increase of soil humidity due to watering can be detected as anomaly and removed from the data. This highlights the need to explicitly distinguish between anomalies and events.

The first plot from the top in Figure 1 visually presents this problem in real ultrasonic sensor data obtained from an IoT company. The sensor is plugged on the top of a tank to monitor its liquid level (y axis) over time (x axis). As shown in the figure, sudden changes appear in isolation (e.g., 08-November)

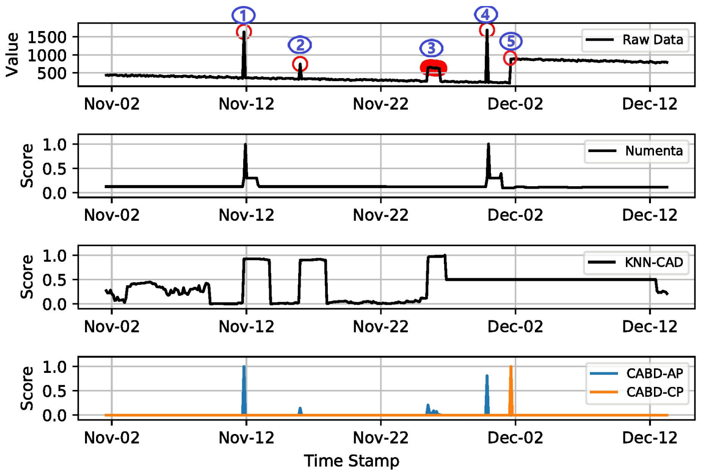

Fig. 1: An example of IoT data (top plot) and anomaly detection results for three algorithms. Points labeled as 1-4 are anomalies, while point 5 represents a water filling event.

and as small groups (24 to 26-November). These abnormal values are sensor errors, and should be fixed or removed from the data set. On the other hand, the data change reflecting the filling of the tank (30-November) should be preserved.

Challenges. The first challenge is that *distinguishing anomalies and events is difficult*. Anomaly detection methods on time series, either neighbor-based [6], [35], [20], ensembles [31], or statistical models [7], [37], [32], [33], [40], classify a data point as abnormal if it significantly differs from historical observations. Unfortunately, change points also show this behaviour and existing detection methods can recognize such points as anomalies. For state-of-the-art anomaly detection methods, the presence of change points in time series significantly decreases the quality of the detection. Since the change points are not flagged in the training data, their presence directly affects the prediction result. As shown in Figure 1, for this dataset Numenta does not detect half of the anomalies, while KNN-CAD fails in all its detection (F-score below 20%). This is a common problem and change points radically affect the performance of both supervised and unsupervised state-of-the-art anomaly detection methods.

The second challenge is that the performance of several methods depends on *parameters that are dataset specific*. For example, KNN-CAD requires a "window length" parameter defining the size of the sliding window, SPOT, DSPOT, and DONUT require a parameter defining the percentage of abnormal data in the dataset. These parameters vary with different datasets.

The third challenge is that *labeled data is not available* in general. Unsupervised detection methods can perform very poorly in these settings. One approach is then to use supervised learning methods to model the distinction between anomalies and change points. However, labeling data requires expensive manual labour.

Our approach. The three challenges above motivate a new approach to tackle the problems.

For the first challenge, we propose an algorithm that accurately *detects both anomalies and change points* based on the combination of a novel concept of neighborhood, namely *Inverse Nearest Neighbor* (INN), and unsupervised probabilistic classification. Given a data series containing an object A, an object B is in the INN of A, if, for any k, object A is a top-k neighbor of B and B is a top-k neighbor of A. Once INNs have been computed for all candidate points, it is possible to classify (as anomaly or change) such points based on INN properties, modeled as features. The bottom plot in Figure 1 shows the output for our algorithm exploiting INNs to predict anomalies (namely CABD-AP) and change points (namely CABD-CP) separately and precisely.

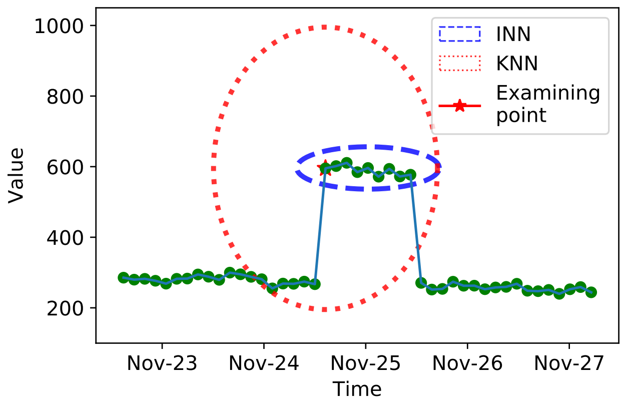

Fig. 2: Inverse Nearest Neighbor vs K-Nearest Neighbor.

The INN concept exploits the asymmetry in the distance relation based on top-k ranking. While simple, this neighborhood definition enables better predictions than existing neighborhood concepts such as k-nearest neighbor (KNN). Most importantly, the number of inverse nearest neighbors for every point is not defined with a parameter. This address the second challenge above, as *we do not need to find the configuration* (such as the correct k-distance value) for every input. Figure 2 illustrates the difference between INN and KNN. When evaluating if a data point (red star) belongs to a group of anomalies, its INN identifies precisely the group while its KNN with an inappropriate k parameter contains both anomalous and normal points. These differences make our solution more robust and directly affect the prediction.

For the third challenge, we introduce an *interactive* approach that minimizes the user involvement in annotating data points while guaranteeing quality over the results of the process. We let the user express a desired *minimum confidence* for the data and the application at hand. The confidence of a classification model is then used as a termination condition for an active learning process, according to the user input. Higher accuracy requirements demand more points labeled to enrich the model until the desired confidence is achieved. Experiments demonstrate that labeling few data points dramatically increases the detection quality.

Contributions. We summarize our contributions as:

1) The concept of *inverse nearest neighbor* (INN) to characterize anomalous data points. INNs are non-parametric and enable the computation of a set of scores that distinguish errors and change points. We introduce algorithms to compute INNs efficiently with methods inspired by binary search and with an aggressive pruning condition.
2) The *Comprehensive Anomaly and Change point Detection* algorithm (CABD), a robust method for detecting anomalies for time series containing also events (i.e., change points). INN scores effectively models the difference between anomalies and events and are used both in an unsupervised version of CABD (without user input) and in an active learning version based on an uncertainty sampling scheme to guide users in annotating points.
3) We have implemented CABD and extensively tested it in a production environment. The prototype produces high-quality error detection in practical IoT use-cases with very few labeled examples from the users, e.g., four annotations for a time series of 2k points. We also demonstrate that CABD can be plugged with existing data repairing algorithms for time series improving their quality 4 times with only 2% of data annotated in the active learning step.

The remainder of the paper is organized as follows. In Section II, we formalize the problem of anomaly detection with user interaction and related definitions. The INN concept and CABD algorithms are presented in Section III and Section IV, respectively. Section V reports the evaluation of our method through real and synthetic datasets. Section VI discusses related work, and conclusions are reported in Section VII.

II. Preliminaries

We first give the basic definitions for our setting and we then introduce our problem formulation.

Definitions. We define a time series as follows:

Definition 1: A *time series* is a set of data points indexed in the time order and collected at successive equally spaced points in time. We denote with $X = \{x_1, x_2, \dots, x_n\}$ a time series with n data points collected at time $\mathcal{T} = \{t_1, t_2, \dots, t_n\}$ where $t_i - t_{i-1} = t_{i-1} - t_{i-2}$ with $i = 1, 2, 3, \dots, n$.

We use the *Euclidean distance* to calculate the distance of two data points. It is the straight-line distance between two points in an Euclidean space and it is calculated by the squared root of the differences between the coordinates of two points:

Definition 2: The Euclidean distance between data point p = (p_1, p_2) and q = (q_1, q_2) is given by

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{1}$$

Standardization is a necessary step to deal with differing scales in input values. Standardizing a dataset consists in rescaling the data distributions so that the overall mean and standard deviation are 0 and 1, respectively. A value x_i in time series data X is standardized as follows:

$$x_i = \frac{x_i - mean(X)}{std(X)} \tag{2}$$

This formula enables the applicability of the Euclidean distance in the INN computation, where we mix distances across different dimensions.

Anomaly and Change Point. Anomaly detection is used to identify unusual patterns that do not conform to expected behaviors, also called outliers. Anomalies can be broadly categorized as [8]:

- **Single anomalies**: A single data point is anomalous if it is significantly different from the remaining data. For example, as shown in Figure 1, the changes at 08-November, 14-November and 28-November are single anomalies.
- **Collective anomalies:** This anomaly type contains a set of consecutive point anomalies represented as an abnormal data *pattern*. This pattern does not comply with the dataset distribution. For example, the small data pattern (24 to 26-November) presented in Figure 1 is a collective anomaly.

A change point, also called a break point, is the point at which the statistical properties of a sequence of observations change [8]. Assume we have a time series $X = \{x_1, x_2, \dots, x_n\}$ with m change points at positions $\mathcal{C} = \{c_1, c_2, \dots, c_m\}$ with $c_m < n$. The change points separate the dataset into $m + 1$ segments ($\{x_1, \dots, x_{c_1}\}$, $\{x_{c_1+1}, \dots, x_2\}, \dots$) that may have different sizes and statistical properties (mean, average, standard deviation).

Problem Statement. We consider a time series $X = \{x_1, x_2, \dots, x_n\}$ of n observations, where x_i is the i^{th} data point, that may contain both errors and events. Its errors could be either single or collective anomalies. Let $ac_i = \{x_i, \dots, x_{i+s} \mid x \in X, s \in \mathbb{N}\}$, as_i and c_i denote a collective anomaly sized s, a single anomaly, and a change point at data point $x_i \in X$, respectively. Our problem statement is formalized as follow:

Problem: *Given a detection confidence q and time series $X = \{x_1, x_2, \dots, x_n\}$, detect any collective anomaly ac_i, single anomaly as_i, and change point c_i with confidence above q while minimizing the number of label requests to the user.*

Example 1: Consider the time series in Figure 1 with a collective anomaly occurring around Nov-25, three single anomalies at Nov-08, Nov-14 and Nov-28, and a change point at Nov-30. For this series, Numenta cannot detect the sequence of errors and confuses change points with abnormal points. Single anomalies are incorrectly detected as collective anomalies by KNN-CAD. Our goal is to identify and distinguish anomalies and change points with minimal user effort in labeling data points.

III. Inverse Nearest Neighbor

We call *r neighbors* of a point p the group of r closest points to p, in symbol $NN_r(p)$. We also call *r reverse neighbors* of a point p the group of points that have p as one of their r nearest neighbors, in symbol $RNN_r(p)$.

A point x_m is in the Inverse Nearest Neighbor (INN) of another point x_i, if, for a given r, x_m belongs to both the r neighbors and the r reverse neighbors of x_i. Given the time series $X = \{x_1, x_2, \dots, x_n\}$ and two points $x_m, x_i \in X$, we define:

$$x_m \in INN_r(x_i) \text{ iff } \begin{cases} x_m \in NN_r(x_i) \\ x_m \in RNN_r(x_i) \end{cases} \tag{3}$$

One way to compute the INN for a point x_i is to not constrain it and take the union of all INN$_r(x_i)$ for any r from 1 to the size of the data series. Another way is to compute only the minimal INN for x_i (INN(x_i)) by starting from the top 1 nearest neighbor for x_i and incrementally grow until there are no more new neighbors in INN. With these notions, the major difference between INN and other neighborhood concepts, such as k-nearest neighbor, is that the number of neighbors for a data point is not pre-defined. We adopt the minimal INN computation algorithm, which is therefore non-parametric. Notice that the number of neighbours in the INNs of points in the same dataset is not necessarily the same.

Algorithm 1 Minimal INN computation for data point x_i

```
Input: Time series X and data point $x_i \in X$
Output: INN($x_i$)
1. Initializing: flag = 0, r = 1, $INN(x_i) = \varnothing$
2. Find the $r$ nearest neighbors $Y$ for $x_i$
        Find the $r$ nearest neighbors for each $y_i \in Y$
        If $x_i \in NN(y_i)$ and $y_i \notin INN(x_i)$ then
            $INN(x_i) = INN(x_i) \cup (y_i, r)$
3. Compute the size $INN(x_i)$
        If this size does not changed
            Return $INN(x_i)$
        Else
            r++
            go to step 2
```

The details of the INN computation for a given point are described in Algorithm 1. The algorithm steps are illustrated in the following example.

Example 2: Consider time series X, similar in shape to the one in Figure 2, with X = {26.9, 26.8, 27.4, 26.7, 64.5, 65.1, 62.1, 64.4, 62.2, 62.7, 27.1, 25.2, 25.4}[1]. X has a collective anomaly on six points from x_4 to x_9. Consider the INN of x_4, a point belonging to a group of anomalies. For r = 1, we have $NN_1(x_4) = \{x_5\}$ and $RNN_1(x_4) = \{x_5\}$. Referring to Equation 3, x_4 and x_5 are INN at distance 1. Similarly, with r values from 2 to 5, we also identify $\{x_6, \dots, x_9\}$ as belonging to the INN of x_4. This is because with $r = 5$, $NN_5(x_9) = \{x_8, x_7, x_6, x_5, x_4\}$ still contains x_4.

To describe $r = 6$, we report the Euclidean distances between data points. With $d(x_4, X)$ =[37.9, 37.8, 37.1, 37.0, 0.0, 1.2, 3.1, 3.0, 4.6, 5.3, 37.4, 39.8, 39.9], we have $NN_6(x_4) = \{x_3, x_5, \dots, x_9\}$. Because of $\{x_5, \dots, x_9\} \in INN(x_4)$ (they are processed at r values from 1 to 5), we examine x_3. We get the distances $d(x_3, X)$ = [3.0, 2.0, 1.2, 0.0, 37.0, 38.5, 35.6, 37.9, 35.8, 36.5, 7.0, 8.1, 9.1]. Based on these values, $NN_6(x_3) = \{x_0, x_1, x_2, x_{10}, x_{11}, x_{12}\}$. Since $x_3 \in NN_6(x_4)$

[1]This example only aims at illustrating the INN generation process. In the experimental evaluation, all datasets are standardized by using Equation 2.

but $x_3 \notin RNN_6(x_4)$, x_3 does not belongs to the INN of x_4. The INN search for x_4 is stopped at $r = 5$ and $INN(x_4) = \{x_5, x_6, x_7, x_8, x_9\}$. With KNN, we would need to provide a constant (5) to detect such group, a data-specific parameter. This shows how INNs identify abnormal groups without pre-defining any parameters.

In the worst-case scenario, the value distribution in the dataset is a flat line and the INN of a point is the whole dataset. We discuss this issue and an optimized INN computation in Section IV. In the following sections, with the "INN of a data point" we refer to the set of points identified by the minimal INN generation Algorithm (Algorithm 1) for that point.

IV. Detection using INNs and Active Learning

Unlike detection algorithms that only detect either anomalies or change points, our goal is to effectively detect both categories in a single algorithm with minimal input from the user. In this section, we first present the overall algorithm with active learning. We then discuss each step with more details.

Algorithm 2 Anomaly and Change Point Detection

```
Input: Time series X, User defined confidence $\gamma$
Output: Anomaly list $Y$, Change point list $Z$
1. $\theta \leftarrow$ Candidate(X)
2. Y, Z = [ ]
3. For $x_i$ in $\theta$ do
       $\beta^{(x_i)} \leftarrow$ Score($x_i, X$)
4. $\mathcal{CW}, Y, Z \leftarrow$ Evaluate Detection($\beta$)
5. If min($\mathcal{CW}$) $\leq \gamma$ then
       Labeling and Go to step 4
   Return $Y, Z$
```

Algorithm Overview. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a time series, where Y and Z are set of anomaly points and change points of X, respectively. Algorithm 2 presents the major steps of our proposal, which takes X as an input and produces Y, Z together with their *Confidence Weights* (CW). The algorithm allows the user to configure the minimum desired confidence as input in order to ensure detection quality. The major steps of the algorithm are described as below:

1) **Candidate Estimation**, in Line 1, it generates potential candidate points θ from extreme values in the series based on their absolute second derivative (Definition 3).
2) **Score Computation**, in Line 3, it computes a score metric from INN for each candidate x_i in θ. This metric includes magnitude score, correlation score, and variance score, denoted as $\beta^{(x_i)}$.
3) **Score Evaluation**, in Line 4, it uses a probabilistic classification to classify the candidates into three classes (change, anomaly, and normal points) based on their score values. Active learning using the uncertainty model is applied to minimize user interaction. The most uncertain points are queried and labeled by the users. Each point x_i also get assigned a confidence weight $\mathcal{C}^{(x_i)}$.
4) **Classification Evaluation**, in Line 5, it triggers the active learning process if the confidence weight of a candidate point is lower than the user defined minimum quality.

Candidate Estimation. Our goal is to recognize both errors and events. Therefore, we first introduce an unsupervised method to find the points that are candidate to be label as errors or events.

We rely on standard measures such as mean, variance, and correlation to identify the candidates, as done in the literature [4], [29], [8]. More specifically, we identify candidate points based on its absolute second derivative, which represents the rate of changes of a data point; this is widely used to identify the critical points (e.g., the local minimum or local maximum) [5].

Definition 3: The absolute value of second difference of $x_i(p)$, denoted as $\triangle'' x_i$, which is defined that:

$$\triangle'' x_i(p) = |\triangle x_i - \triangle x_{i-1}|,\ i = 1, 2, 3, ..., n \tag{4}$$

While $\triangle x_i$ is the absolute value of first difference of $x_i \in X$, is defined that:

$$\triangle x_i(p) = |x_i(p) - x_{i-1}(p)|,\ i = 1, 2, 3, ..., n \tag{5}$$

Formally, given a time series $X = x_1, x_2, \ldots, x_n$. The anomaly score of X is denoted by ∂:

$$\partial(X) = \{\triangle" x_1, \triangle" x_2, \ldots, \triangle" x_i\} \mid i \in \{1, 2, \ldots, n-1\} \tag{6}$$

To identify the candidates, we use the *Median Absolute Deviation* (MAD), which is a robust measure of the variability in data [29]. In addition, MAD is more resilient to outliers than the standard deviation. If MAD of the anomaly score of a data point is higher than MAD of the anomaly score of the whole data set, it is considered to be a candidate. We validate these candidates in the latter detection steps.

Definition 4: Given time series X and the anomaly score ∂, MAD is defined as the median from sample median.

$$MAD(X) = median(|\partial(X_i) - median(\partial(X)))\mid X_i \in X \tag{7}$$

Notice that the candidate estimation is independent of the INNs and it is a global analysis of the data series.

Score Computation. In this step, we compute the scores based on INNs for each candidate point. These scores are used in next step as the features of a probabilistic classifier, which outputs the probability of a candidate to be an anomaly, change, or normal points. The metric has three scores: *Magnitude*, *Correlation*, and *Variance*. Each score represents a characteristic of the candidate's INN. The union of such characteristics enable the distinction between anomalies and change points.

The *magnitude score* of a point describes the ratio of its INN size over the global dataset size. Based on standard anomaly definition [8], the size of an anomaly pattern (a collective anomaly) should be less than five percent of dataset. Therefore, if the magnitude score is higher than five percent, the candidate may be a normal point. The *correlation score* represents the regularity of its INN pattern over the entire dataset. If this pattern rarely occurs, e.g., it is not periodic, the candidate may be an abnormal point. The *variance score* represents the local impact in terms of standard deviation of the removal of candidate INN. If this score is low (close to 0), the candidate is more likely to be normal. Let SS(x_i) be a function that returns the INN size of data point x_i and SPa(x_i) be the INN of x_i including SS(x_i) number of adjacent points in both sides. The definitions of the three scores are formalized next, starting from the magnitude score.

Definition 5: Magnitude score (MS) of data point is the ratio of its INN size over the size of dataset, denoted by MS. Given time series $X = \{x_1, x_2, ..., x_n\}$ length n and $x_i \in X$, the MS of x_i is defined as:

$$MS(x_i) = \frac{SS(x_i)}{n} \tag{8}$$

For the correlation score, we need to introduce some definitions first.

The Symbolic Aggregate Approximation (SAX) algorithm transforms a time series into a string [26]. This algorithm is based on the piecewise aggregate approximation (PAA) and it is used to detect unusual patterns in time series, duplicated shapes in large databases, and time series motifs. We define PAA and SAX next:

Definition 6: Piecewise Aggregate Approximation (PAA) transforms a time-series X of length n into vector $\bar{X} = (\bar{x}_1, ..., \bar{x}_M)$ with $M \leq n$ where:

$$\bar{x}_i = \frac{M}{n} \sum_{j=n/M(i-1)+1}^{(n/M)i} x_j \tag{9}$$

Definition 7: Symbolic Aggregate Approximation (SAX) transforms a time-series X of length n into an arbitrary string by using PAA. A time series X length n can be represented by a word $\hat{X} = \{\hat{x_1}, \hat{x_2}, \ldots, \hat{x_n}\}$ with $\hat{x_i}$ is a character of alphabet. Let denoted α_j is the j^{th} element of the alphabet. θ_{j-1}, θ_j are given thresholds.

$$\hat{x_i} = \alpha_j \quad s.t \quad \theta_{j-1} \leq PAA(x_i) \leq \theta_j \tag{10}$$

We are now ready to define the correlation score.

Definition 8: Correlation Score (CS) of a data point is the frequency of occurrence of its INN pattern, represented as a string by using SAX, in the dataset (definition 7).

Given time series $X = \{x_1, x_2, ..., x_n\}$ and $x_i \in X$

$$CS(x_i) = frequency\left(\frac{SAX(INN(x_i))}{SAX(X)}\right) \tag{11}$$

Finally, we define the variance score.

Definition 9: Variance Score (VS) of a data point measures how removing the INN for a point changes the standard deviation of its SPa.

$$VS(x_i) = \frac{std(SPa(x_i) - INN(x_i))}{std(SPa(x_i))} \tag{12}$$

Algorithm 3 illustrates the score metric calculation process. At the first step (Line 1), we obtain the INN for each candidate point x_i using Algorithm 1. Then, the three scores are calculated in parallel to optimize performance (Line 3, 4, 5). At the final step (Line 6), all scores are collected to form a feature (three dimensional) matrix for the classification model.

Score Evaluation. This step relies on the metric scores together with a probabilistic classification algorithm to estimate the probability of data point x_i to be an anomaly, change, or normal point. The classification receives the metric scores of x_i as input and identifies the probability of x_i falling into three classes (anomaly, change, or normal point) as output. CABD is designed for high modularity and flexibility. It allows to plug-and-play different classification algorithms. By default, CABD uses the *random forest classification* [25]. Initially, without user intervention, the classification works on a set of hypotheses. The probability produced by the classification algorithm is considered as the confidence weight of the examined data point. With the presence of a user, active learning with an uncertainty sampling scheme [9], namely *CAL*, is applied to increase classification performance by labeling the most uncertain instances.

Algorithm 3 Score Computation

```
Input: Data point $x_i$, time series X
Output: Score Metric $\beta^{(x_i)}$
1. $\gamma \leftarrow$ INN_searching($x_i$, $X$)
2. $\eta \leftarrow$ SS($x_i, \gamma$)
  3. $\kappa \leftarrow$ MS($x_i$, $\eta$)
  4. $\xi \leftarrow$ CS($x_i$, $\gamma$)
  5. $\varphi \leftarrow$ VS($x_i, \eta, \gamma$)
6. $\beta^{(x_i)} \leftarrow$ [[$\kappa$], [$\xi$], [$\varphi$]]
Return $\beta^{(x_i)}$
```

Without user interaction[2], the initial training set of the classifier is built on a set of hypotheses $\mathcal{H}$, which includes three main decision rules based on the score metric:

1) The magnitude score of an abnormal point must be lower than $k\%$. This means, the spreading pattern size of this point is lower than $k\%$ of data size. Particularly, the spreading pattern size of single anomaly equals 0.
2) The correlation score of an abnormal point must be lower than $c\%$. This means, the spreading pattern of this point must occur lower that $c\%$ frequently in dataset
3) The variance score of an abnormal point must be higher than $v\%$. This means, the standard deviation of spreading pattern with k-neighbors must reduce at least $v\%$ after removing the spreading pattern.

We derive the set of mentioned thresholds by using an unsupervised clustering algorithm that classifies candidates into four groups. Then, the label Y = {single anomaly, collective anomaly, change point, normal point} is assigned to these groups based on observed characteristics. We observe that the "single anomaly" group can be identified by low magnitude and correlation scores, while its variance score is high. The Figure 3 presents magnitude (y axis) and variance score (x axis) of a clustering result from a synthetic dataset. The blue curves are the boundaries of the groups which are used to define the thresholds. As the figure shown, the "single anomaly" group has low magnitude (very small INN size) and high variance score (their removal significantly change the SPa). In our experiment, we use the unsupervised Gaussian Mixture clustering algorithm because it works nicely with clusters that are not round shaped, but any other clustering method can be plugged to our solution.

[2]Experiments in Section V distinguish with/without active learning cases.

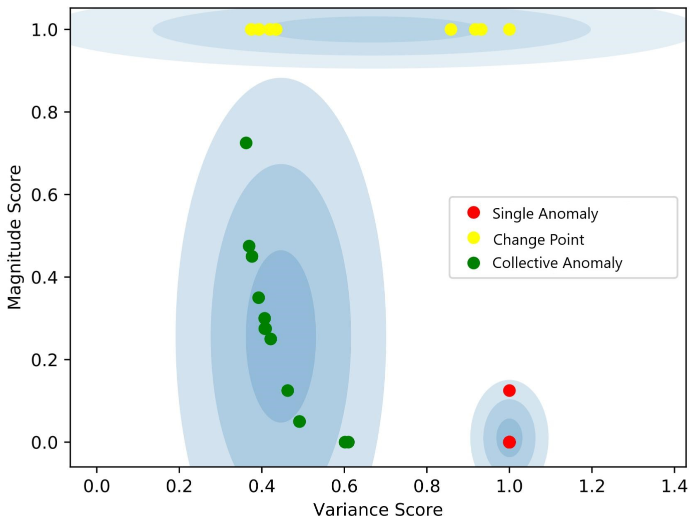

Fig. 3: Unsupervised clustering output for a synthetic dataset.

When we can involve the user, we use *CAL* to get annotations. We examine the most likely class of data points $x_i \in X$ based upon the probability produced by the classification algorithm. We decide whether to require its label $y_i \in Y = \{\text{abnormal point, normal point, change point}\}$ from the user based on the uncertainty of the classification result defined by:

$$\mathcal{U}(x) = 1 - P(\hat{x}|x) \tag{13}$$

where x is the data point and $\hat{x}$ is the most likely classification. The querying process of *CAL* is stopped if all confidence weights are higher than the user defined confidence.

Example 3: If a data point x_i can be classified across the three labels (abnormal point, normal point, change point) with confidences $[0.1, 0.3, 0.6]$, it is considered a change point with 0.6 confidence weight and 0.4 uncertainty.

Algorithm 4 Score Evaluation

```
Input: Unlabeled data set $X$, probabilistic model $Z$,
initial training set $\mathcal{V}$, threshold $\gamma$.
Output: $[CW, \varphi]$
1. $[CW, \varphi]$ = Z(X,$\mathcal{V}$)
2. While $min(CW) \leq \gamma$ do
       $x_t \leftarrow Query(x_t, \varphi, X)$
       Label $y_t$ for $x_t$
       Set $\mathcal{V} = \mathcal{V} \cup \{(x_t, y_t)\}$
       Update $[CW, \varphi]$ = Z(X,$\mathcal{V}$)
Return $[CW, \varphi]$
```

Algorithm 4 reports the *CAL* scheme for active learning in the Score evaluation step. We denote confidence weight and uncertainty with CW and φ, respectively.

Optimizations. We notice that the time spent in the score calculation can be dramatically reduced by analyzing less points in the INN computation for all candidate points. In fact, if the dataset is large, the number of candidates points can be large and for each point we may end up analyzing a large number of neighbors. For this problem, we propose two solutions. First, we show how the number of points involved in the INN computation can be reduced with a binary search method. Second, we add a new stopping condition in the INN computation based on the maximum possible size of the INN.

First, recall that when computing the INN of data point x denoted INN(x) in Algorithm 1, n is the INN size obtained by starting at $= 1$ and increasing by one until n is not changed. The complexity of such approach is O(n) with n being the size of INN(x) in the worst case. This can be optimized by using a binary search to find the INN set for both sides (left and right side) of the data point x. The complete INN is then the union of two sets. Thereby, the complexity is reduced from O(n) to 2*O(Log $\frac{n}{2}$). Second, we use a parameter in the binary search to set up the space of points of interest, i.e., the maximum searching position. In practice, if the size of an abnormal pattern is higher than 5% of the size of the dataset, it should not be considered an anomaly. Thus, this bound is used as the maximum search range. To ensure that the binary search output is minimal INN, we assume that the INN(x) is not segmented.

Algorithm 5 Binary INN computation (right side)

```
Input: Data point $x_i$, search range threshold t
Output: $INN_R(x_i)$
1. $L = i; R = t - 1; INN_R(x_i) = []$;
2. While $L \leq R$ do
       $m = floor((L + R)/2)$
       If $x_m \in NN_m(x_i)$ and $x_m \in RNN_m(x_i)$
           $L = m + 1$
       Else
           $R = m - 1$
3. $INN_R(x_i) = [x_i, \ldots, x_m]$
4. Return $INN_R(x_i)$
```

Given data point x_i and searching range threshold t, Algorithm 5 illustrates the Binary INN computation for the right side of x_i. The procedure for the left side is the same.

V. Evaluation

In this section, we experimentally evaluate the quality and efficiency of our proposal on both real and synthetic datasets. The datasets have been standardized with the methods defined in Section II. Our goal is to demonstrate:

- The superiority of CABD w.r.t. existing algorithms in both detection quality (anomaly, change point detection) and runtime over real and synthetic datasets;
- The effectiveness of INNs and active learning;
- The improvements in data repair for real sensor data when CABD is used to identify quality issues.

A. *Metric of measurement*

To evaluate the effectiveness, we use three metrics. Let S denote anomaly and change points detected by the algorithm and G denote the corresponding ground truth for the same time series. The precision and recall are defined as: $P = (|S| \cap G)/|S|$ and $R = |S| \cap G/|G|$, respectively. F-score or F-measure, combining P and R, is defined as: $F = 2 * \frac{P*R}{P+G}$.

To assess the advantages of using interactive learning in comparison to manually labeling all cases, we use a benefit function calculated from the ratio of the number of human actions (labeling) over the total number of anomaly and change points [17]. Given the number of annotations T_A and the total number of anomaly and change points M, we define the *benefit function* as:

$$BNF = 1 - T_A/M \tag{14}$$

B. Datasets

Real datasets. We created a dataset in collaboration with a company: *IoT data*[3]. Data has been collected every hour from 2 ultrasonic sensors deployed on the top of tanks to monitor liquid levels for a total of 3.1K measures. Errors naturally occur over time. Since the company manages the tank operations, such as filling or consuming, errors and change points have been manually labeled as ground truth. The second dataset is *Yahoo! lab data*[4]. It provides 50 time series with annotated anomalies taken from real products traffic, relations have from 1.5K to 20K records. Finally, we use the *KPI* datasets[5], which contain time-series about key performance indicator (KPI) information from internet companies. Each KPI dataset has around 100K records at 1-minute interval and labeled anomalies. As the Yahoo and KPI datasets have no change points, these datasets are used to measure the quality of anomaly detection only.

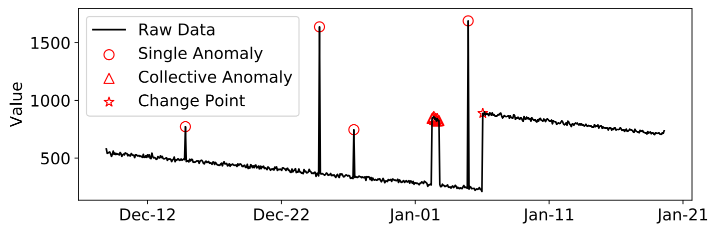

Fig. 4: Example of synthetic dataset.

Synthetic dataset. This dataset aims to assess the effectiveness of our algorithm in the presence of various anomaly types (single and collective ones) and change points in different proportions. We first generate the data points following real data distributions. Then, we fit this data to a time series obtained from the production environment to preserve the trend and seasonality. Lastly, we randomly inject a mix of anomaly types with varying lengths and magnitudes in 25 different relations with 20K records each. These anomalies are stored to evaluate the output of the algorithms. Figure 4 illustrates a part of a synthetic dataset (namely *ds-1*) that includes both single and collective anomalies, and a change point.

C. Results

Real datasets. We first evaluate the detection quality of our proposal over the real datasets. Since Yahoo and KPI did not record the change points, we only perform anomaly detection on such datasets. In this and the following supervised experiments, the default user confidence value is set at 0.8.

[3]Available at https://github.com/kimhungGCZ/anomaly_dataset
[4]Available at http://labs.yahoo.com/Academic_Relations
[5]Available at http://iops.ai/competition_detail/?competition_id=5&flag=1

Dataset	%An	%CP	W/O AL		W/ AL		# of queries
			AP F-score	CP F-score	AP F-score	CP F-score	
Synthetic	12.5	9.5	32.1	34.2	67.9	83.6	38.7 (avg)
Yahoo	1.0	-	43.5	-	78.8	-	5.0 (avg)
KPI	1.8	-	55.6	-	70.3	-	5.3 (avg)
IoT	0.8	1.0	53.7	33.3	100.0	100.0	4.0

TABLE I: Qualitative evaluation of CABD for Anomaly Prediction (AP) and Change Point Prediction (CP) on all datasets.

We report results for all datasets in Table I. For IoT, we note that, without active learning, the detection recall on average is 100%. This means all abnormal points are recognized and is not surprising as in the unsupervised setting there is no enforcement of a desired confidence. However, the F-scores of anomaly and change point detection are only 53.7% and 33.3%, respectively. With active learning, the F-score reaches 100% after labeling only four candidate points. These results prove that our proposal effectively detects both anomalies and change points with limited labeling efforts.

For Yahoo, we present the results averaged over the 50 datasets in Table I (the online tech. report [24] contains detailed results). Without active learning, the average detection precision and recall are 47.4% and 60.4%, respectively. With active learning, precision and recall increase to 87.1% and 77.4%, respectively. Moreover, the average numbers of queries is 5, with an average benefit score (BNF) of 0.3, i.e., labeling two candidate points leads on average to the classification of a third. By manually analyzing the incorrect classifications, we notice that false negatives usually occur at the boundaries of abnormal data, especially at the end of a collective anomaly. Experimental results with KPI datasets are similar to the Yahoo and IoT scenarios. Without labels, the F-score of our algorithm is 55.6%, but labeling 5 data points (on average) increases this value to 70.3%.

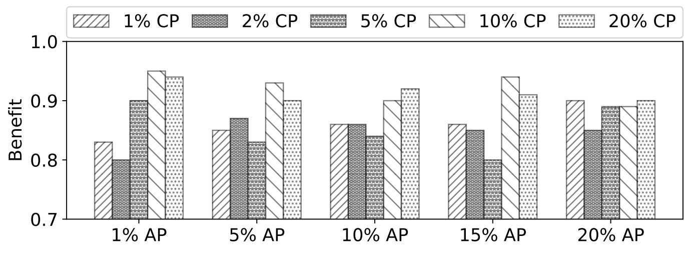

Fig. 5: BNF with increasing anomaly and change points.

Synthetic datasets. Next, we evaluate CABD on synthetic datasets with increasing percentages of anomaly and change points (from 1% to 20% of the data size). We note in Table I that CABD with active learning significantly improves the anomaly and change point detection accuracy. The average F-score increases from 32.1% to 67.9% and from 34.2% to 83.6% for anomaly and change point detection, respectively. Remarkably, the active learning takes more benefits at low anomaly percentage. For example: in dataset with 1% anomaly such as *ds-1*, the F-score increases by about 80% (absolute value) from 17.3% to 97.3% after active learning.

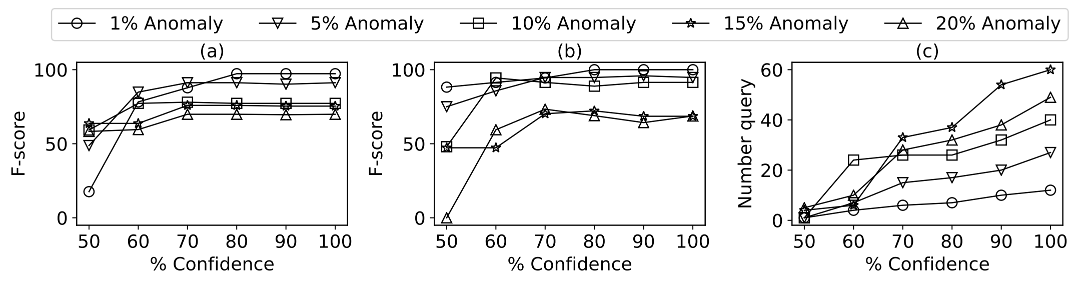

Fig. 6: Increasing the desired confidence: (a) Anomaly detection quality; (b) Change point detection quality; (c) # of queries.

The benefit score (BNF) is about 0.88 on average, which means that labeling 12 candidate points leads to recognize 100 anomalies. Regardless of the changes in the percentage of anomaly and change points, the query benefit score is consistent from 0.8 to 0.96, as shown in Figure 5. This result demonstrates that the INN score metrics are highly related to detecting anomaly and change points.

To assess the impact of the user defined minimum confidence, we report results when varying it for anomaly detection and change point detection in Figures 6(a) and 6(b), respectively. We use different synthetic datasets with different values of anomaly percentages, from 1 to 20%. As expected, higher requirements lead to better F-score results. We also observe in Figure 6(c) that more anomalies in data series lead to a bigger increase in the number of user queries when we vary the required confidence. Plots in Figure 6 also show that high percentage of anomalies and change points impact the effectiveness of CABD. Increasing the anomaly percentage from 1% to 20% decreases the F-score down to 27.4% and 31% for anomaly and change point detection, respectively. We observe that if a single anomaly point is very close to a change point, its INN is larger than when in isolation. Thus, the metric score of such point is likely to lead to a change point classification.

D. Comparison with baselines

We first compare our algorithm to both unsupervised and supervised anomaly detection methods. We then compare our solution to existing methods for change point detection and against a combination of baseline methods for the two tasks.

Anomaly Unsupervised. We take several state-of-the-art anomaly detection methods, including Numenta, Twitter-AD [37], Luminol[6], KNN-CAD, ContextOSE[7], Multinomial Relative Entropy [39], Bayesian Online detection [3][8].

A comparative analysis of detection quality on all datasets in Figure 7 shows that the detection quality (F-score) for most algorithms is fairly low when dealing with various anomaly types. The average of F-measure values is under 20% even with the the best performing baseline (Luminol) for KPI and Synthetic datasets. In contrast, CABD always shows significantly better results on all cases. For example, the F-scores of Luminol over Yahoo and IoT datasets are 41% and

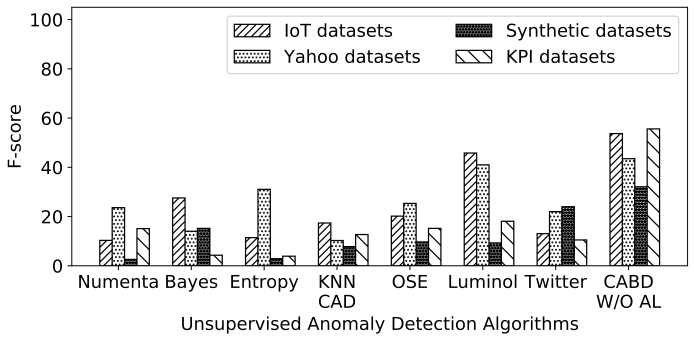

Fig. 7: Comparing detection quality with **unsupervised** anomaly detection algorithms over all datasets.

45.8%, respectively, whereas the ones of (unsupervised) CABD are 43.5% and 55.6%. The average F-measure score for CABD in the unsupervised setting is 45.3%.

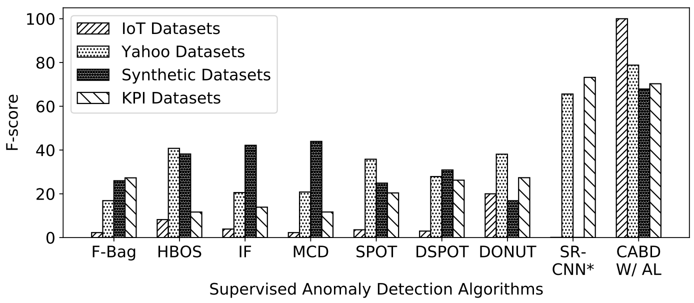

Fig. 8: Comparing detection quality with **supervised** anomaly detection algorithms over all datasets.

Anomaly Supervised. We compare detection quality between CABD and supervised outlier detection algorithms. We trained with annotated datasets Feature Bagging (F-Bag) [23], Histogram-based Outlier Score (HBOS) [15], Isolation Forest (IF) [27], Minimum Covariance Determinant (MCD) [16][9], SPOT and DSPOT [33], and DONUT [40]; results for SR-CNN are reported from the original paper as code is not available [32].

Figure 8 presents the results on all datasets. CABD always shows significantly better performance comparing with others

[6]https://github.com/linkedin/luminol
[7]https://github.com/smirmik/CAD
[8]Source code and parameter settings from the *Numenta Benchmark* [22].
[9]Source code obtained from https://github.com/yzhao062/pyod

with one exception. For example, on Yahoo datasets, the average F-score of CABD after active learning is 78.8% while the best baseline achieves 40% (HBOS). Similarly, F-scores on synthetic datasets are 67.9% and 40.7% for CABD and the best of the supervised algorithms (IF), respectively. In one dataset (KPI), the detection performance of our method is comparable to the one reported in the SR-CNN paper [32] (70.3% vs 73.2%).

Change Point Detection. In the next study, we compare detection quality between CABD and three change point detection algorithms: Pruned Exact Linear Time (PELT) [19], Bottom-up [12], and Binary segmentation [13]. These algorithms are implemented by using the *ruptures* library [36]. The performance of these algorithms depends on a parameter, namely "penalty value", considered as a detection threshold. In our comparison, the best penalty value is found by a bruce-force search from 0 to 100.

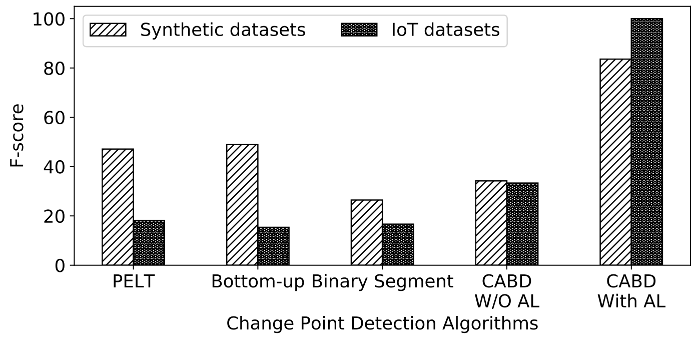

Fig. 9: Comparing change point detection quality over IoT and Synthetic datasets.

Figure 9 shows the results on the IoT and synthetic datasets (change points in Yahoo datasets are not marked). Without active learning, the detection performance of CABD roughly equals its competitors, while CABD with active learning does better. For example, given the best "penalty value" acquired by brute-force search, the F-scores of PELT, Bottom-up, and binary segmentation algorithms on synthetic datasets are 47.1%, 48.9%, and 26.4%, respectively. This means that our competitors could correctly detect half the number of change points with perfect input parameters. The score of CABD with active learning is nearly double (about 83.6%) by exposing to the user only 2% of the dataset on average (about 39 data instances). Note that CABD is designed for both anomaly and change point detection purposes. Thus, the user-annotated data instances might be abnormal or change points. A similar result is also found in IoT datasets.

After separately evaluating anomaly and change point detection methods, we combine one anomaly (HBOS) and one change point detection method (PELT) to compare their combination with our proposal. As shown in Figure 10, CABD's detection quality (F-score) is superior w.r.t. the combination of HBOS and PELT on IoT datasets. However, the score of CABD without active learning is slightly lower on synthetic datasets. In more detail, the average F-score of combining HBOS and PELT on synthetic datasets is 42.5%, while CABD without active learning is 33.1%. Labeling 2% datasets (about

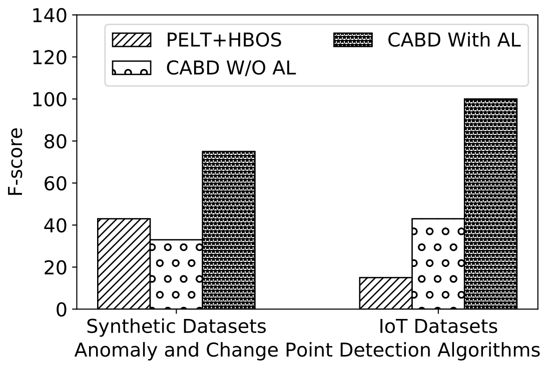

Fig. 10: Comparing detection quality between CABD and a combined baseline (PELT + HBOS) over Synthetic and IoT datasets.

39 data points) on average using active learning, the CABD's F-score increases to 75.8%.

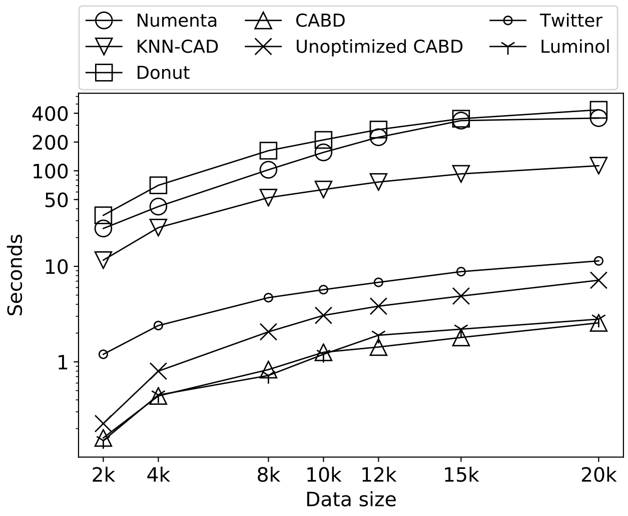

Fig. 11: Runtime of anomaly detection algorithms over different data sizes for Yahoo datasets.

Runtime. We compare the runtime of the algorithms on data series with up to 20K points. Evaluations are performed on a machine with Intel i5-6200U CPU@2.3GHz, 2 Cores, 8GB of RAM and Windows 10. Labeling time from the en-user in active learning is not included. We use a *KD-tree* [30] to enhance search performance.

Figure 11 reports that the runtime of CABD with optimization strategies is roughly equal to the fastest method (Luminol). Also, the optimized CABD is up to 3.5 times faster than its unoptimized version. With 2K points, Numenta, KNN-CAD and Twitter-AD take 24.9, 11.6, and 1.2 seconds, respectively, while CABD without optimization takes 0.21 seconds and its optimized version 0.16 seconds. For 20K data points, Numenta, KNN-CAD, Twitter-AD and unoptimized CABD take 356.0, 113.0, 11.4 and 7.2 seconds, respectively, while optimized CABD takes 2.5 seconds. Deep learning models for anomaly detection are also slow, with Donut with 5 layers taking 34 seconds for 2K points and 435 seconds for

20k. Similar results are also found in the other datasets. The optimization strategies are more effective with larger datasets because of the larger number of candidates that our algorithm analyzes to compute INNs.

In summary, CABD performs better both in detection quality and running time than state-of-the-art algorithms.

E. Effectiveness of Active learning

Summarizing the evaluation results from Table I, the detection quality of CABD with active learning always outperforms the one without it. This stems from the fact that active learning leads the probabilistic classification model to be more accurate.

Round	Y!_1		Y!_23		Y!_42		iot_1		iot_2	
	acc	conf	acc	conf	acc	conf	acc	conf	acc	conf
1	0.1	0.5	0.0	0.5	0.0	0.7	0.8	0.7	0.9	0.7
2	0.1	0.4	0.6	0.5	0.1	0.5	1.0	0.6	1.0	0.7
3	0.6	0.4	0.5	0.5	0.2	0.5	1.0	0.7	1.0	0.7
4	0.6	0.4	0.7	0.7	0.6	0.5	1.0	0.8	1.0	1.0
5	1.0	0.6	0.8	0.6	0.4	0.4				
6	1.0	0.6	0.8	0.5	1.0	0.6				
7	1.0	0.8	1.0	0.7	1.0	0.4				
8			1.0	0.8	1.0	0.9				

TABLE II: Accuracy and confidence for active learning with increasing number of user annotations (user required confidence = 0.8).

Table II reports the accuracy (acc) and the *confidence* of the model (conf) with an increasing number of interactions (rounds) with the user for five datasets. Accuracy here measures the number of correct predictions (true positive and true negative) divided by the union of the model predictions (S) and of the ground truth (G).

We note that the model can identify correctly all points (reach 100% accuracy score) after a small number of queries. For example, after 4 queries, the model reaches 100% accuracy for $Y!\_1$ Yahoo datasets. In some rounds, the model effective accuracy decreases after labeling a candidate point. This happens in cases where the abnormal point is very close to a change point. In this case, the metric scores for this point are similar to change point behaviour, i.e., high magnitude score and correlation score together with low variance score. In these corner cases, CABD is mislead and incorrectly detects the anomaly as a change point until in a following step the model is updated and fixes the issue.

F. Effectiveness of INN and associated scores

To demonstrate the efficacy of INN in comparison with KNN, we study INN and KNN impact in our approach. The best k parameter is determined by brute-force searching in the range from 0 to data size.

Figure 12 demonstrates that CABD using INN (CABD-INN) achieves better performance in comparison with its variant using KNN (CABD-KNN) both with and without active learning (AL). In the Yahoo dataset, the F-scores of CABD-KNN are 31% and 48.32% for before and after AL, respectively. Replacing KNN by INN, the F-score significantly increases to 43.5% for the unsupervised cases and to 78.8% with user involvement. Differences are even bigger with synthetic datasets. In change point detection, the F-score of CABD-INN is 34.2% before AL and 83.6% after AL, while for CABD-KNN these values are only about 4.4% and 6.5%, respectively. These results confirm the superiority of INN over KNN, even when k is manually determined.

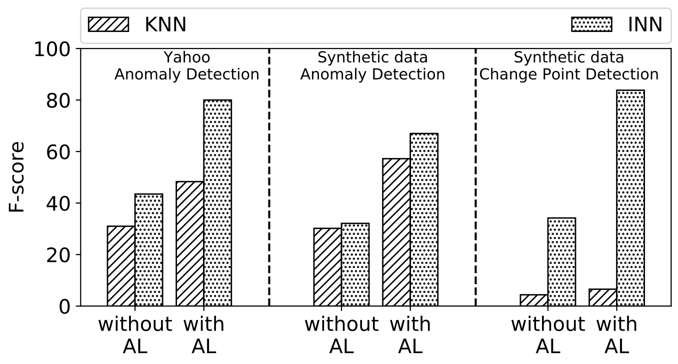

Fig. 12: INN and KNN with and w/out active learning.

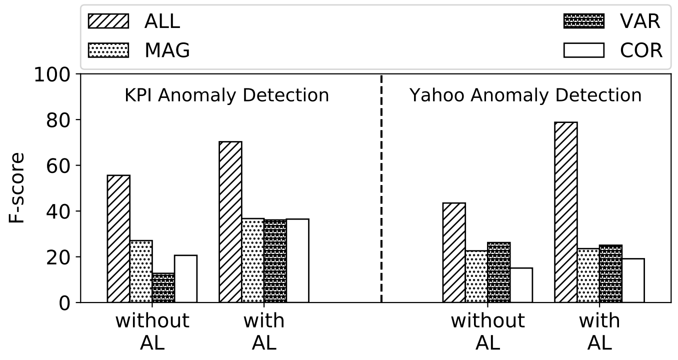

Fig. 13: The influence of INN-based Magnitude (MAG), correlation (COR) and variance (VAR) scores in anomaly detection performance (F-score) over KPI and Yahoo datasets.

To better understand the impact of the different scores that exploits INNs, Figure 13 reports anomaly detection quality results with INN scores used in isolation. The experiments are performed over the Yahoo and KPI datasets. The results show that the Magnitude score is the most influent factor for the algorithm performance over KPI. The F-scores of the classifier using only magnitude score as an input feature with and without active learning are 27.1% and 36.4%, respectively, in comparison to 55.6% and 70.3% of the execution with all scores are combined. For the Yahoo datasets, the Variance is the most effective score. The F-scores of CABD with variance score only with and without active learning are 26.2% and 25.1%, respectively. These results are significantly lower than the ones of CABD with all scores in the metric, with 43.5% and 78.8%, respectively. These results demonstrate the effectiveness of the proposed score metric, which is built from

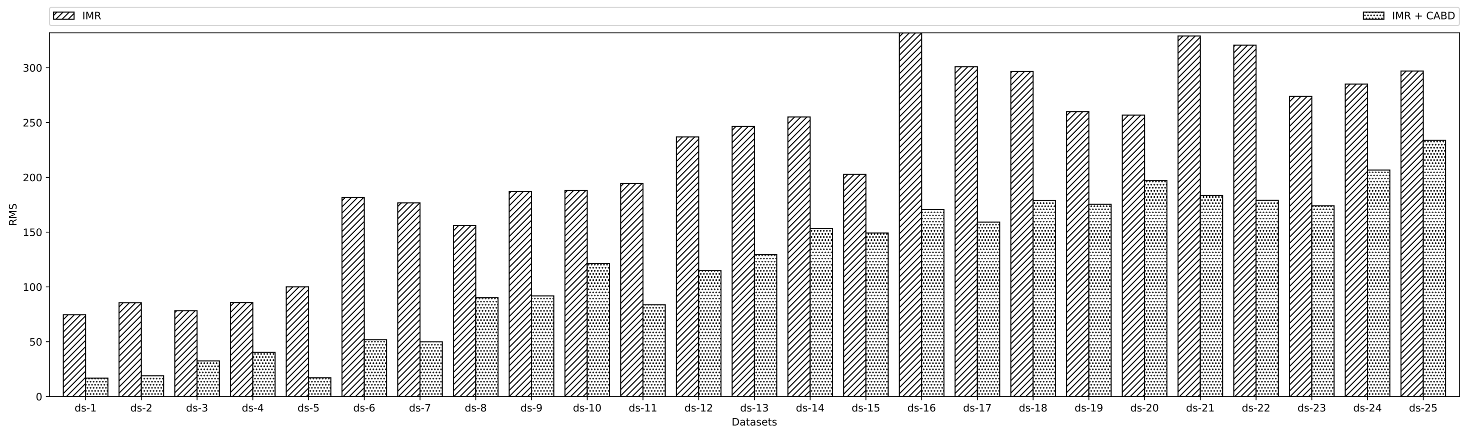

Fig. 14: RMS error in the cleaning process with and w/out our CABD in the IMR data repair algorithm.

our INN concept.

G. *Enhancing data repair*

To minimize the impact of anomalies on data reliability, a repairing process is usually triggered after detecting anomalies and errors. We evaluate the integration of our proposal with a recently proposed data-repairing algorithm, namely *Iterative Minimum Repairing* (IMR) [34], [42]. We show that the quality of the automatic data repairs of IMR is improved by using the Active Learning mechanism of CABD in labeling anomalies. The repairing quality is measured here by a Root Mean Square (*RMS*) error, which evaluates the distance between ground truth and repaired values. Lower RMS error values indicate better results.

Figure 14 illustrates the experiment over our 25 synthetic datasets with varying anomaly and change point percentages. IMR with CABD for the labeling of the data shows significantly better repairs than the original IMR (based on random value selections) in all datasets. For example, in dataset *ds-1* and *ds-2*, CABD reduces RMS error 4 times from about 74.5 and 85.4 to 16.7 and 18.9, respectively. Moreover, for 2000 data points, the average number of labeled data points for synthetic datasets is 38.72. This means that CABD labels about 2% data points to achieve such results comparing with 20% in the original IMR proposal. These results prove the utility of our algorithm in both improving the repairing quality and in reducing the data labeling effort.

VI. Related Work

Nearest neighbor techniques detect the abnormality by examining the distance or similarity between two data instances. The distance can be calculated in different ways. For example, Euclidean distance is used for continuous attributes [6], [35], [20]. Nearest neighbor anomaly detection defines the anomaly score of a data instance as its distance to its k^{th} nearest neighbor in a data set [29]. The basic technique has been improved in various aspects. In Resolution Outlier Factor [11], points are outliers depending on the resolution of the distance thresholds. Conformalized density- and distance-based anomaly detection (KNN-CAD) [7] measures the dissimilarity between observations by combining feature extraction method and conformal paradigm. The density-based anomaly detection is based on the idea that the abnormal points lie in a neighborhood with low density, whereas the normal points lie in a dense area [8]. In Local Outlier Factor (LOF) [6], for a given data instance, the anomaly score is the ratio of the average local densities of its k-nearest neighbors over the local density itself. However, LOF is not effective in detecting regions that are not clearly separated. Several works have extended LOF, such as Cluster-Based Local Outlier Factor [18], which calculates the anomaly score from local distances to nearby clusters. Authors in [33] proposed two algorithms, namely SPOT and DSPOT, to detect outliers in streaming time series based on extreme value theory.

Leveraging the benefit of expert intervention, The Active Anomaly Discovery method aims to reduce the number of faults [10]. It calculates anomaly scores by an ensemble of anomaly detection algorithms [31]. It then associates each score with a weight, which is optimized by labeling the top anomaly score. A similar approach has also been presented for the detection of rare events in network security [38]. The major limitation of these two works is that the inference of the anomaly score threshold and the weight is estimated from a fixed value. Also, the method to minimize expert intervention is not discussed. To mitigate the human effort in labeling data, Microsoft has proposed an anomaly detection algorithm, SR-CNN, that uses Spectral Residual (SR) and Convolutional Neural Network (CNN) to self-generate training data [32]. In a similar attempt, authors in [40] use Variational Auto-Encoder to generate “clean” datasets to support training.

We also position our work w.r.t. data repairing approaches. The SCREEN algorithm [34] proposes a solution for stream data to identify and repair the anomalous “jump” values in a given sequence (windows data) w.r.t the speed constraint while minimizing the repair distance. However, the SCREEN can show high performance in repairing single anomaly but hardly handle a collective anomaly. In addition, the repairing results strongly reply on the correctness of some assumptions. Being aware such limitations, a latter algorithm named Iterative Minimum Repairing (IMR) [42] is proposed. The general idea is that combining between labeling some dirty observations and iterative repairing from high to low confidence repairs could enhance the performance. The final goal of CABD, SCREEN, and IMR are to preserve the notable information in time series data. We have shown how to combine our anomaly detection procedure with IMR to improve its repair quality.

Other methods have studied the problem of interactive error detection [17], [21], [41], [2]. While active learning has been explored for outlier detection [1], our method is the first specifically designed for time series with a focus on anomalies and events. Also, a configuration-free error detection system has recently been proposed to use active learning to steer an ensemble of error detection algorithms [28]. It would be interesting to plug the time series specific algorithms in our proposal in their ensemble framework.

VII. Conclusion

In this paper, we proposed a new unsupervised algorithm for anomaly and change point detection. The proposed method combines the novel concept of invert nearest neighbor and active learning. Unlike most of previous outlier detection approaches, our algorithm not only effectively detects both single and collective anomalies but also preserves and highlights change points. Moreover, we solve the challenge regarding the sensitiveness of parameters in existing methods by applying active learning to discover their right values. Experimental results clearly indicate that the effectiveness of the proposed method for real IoT use-cases.

We plan to study how our techniques apply on multi-dimensional times series. Also, to further prove the effectiveness of invert nearest neighbor and active learning, we plan to apply them not only for anomaly detection but also for data repairing and clustering.

References

[1] N. Abe, B. Zadrozny, and J. Langford. Outlier detection by active learning. In *SIGKDD*, pages 504–509, 2006.

[2] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, 2016.

[3] R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. *arXiv:0710.3742*, 2007.

[4] S. Aminikhanghahi and D. Cook. A survey of methods for time series change point detection. *Knowledge and inf. sys.*, 51(2):339–367, 2017.

[5] B. Andrews. Classification of limiting shapes for isotropic curve flows. *Journal of the American mathematical society*, 16(2):443–459, 2003.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[7] E. Burnaev and V. Ishimtsev. Conformalized density-and distance-based anomaly detection in time-series data. *arXiv:1608.04585*, 2016.

[8] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *CSUR*, 41(3):15, 2009.

[9] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[10] S. Das, W. Wong, T. Dietterich, A. Fern, and A. Emmott. Incorporating expert feedback into active anomaly discovery. In *ICDM*, 2016.

[11] H. Fan, O. Zaïane, A. Foss, and J. Wu. A nonparametric outlier detection for effectively discovering top-n outliers from engineering data. *Advances in Knowledge Discovery and Data Mining*, 2006.

[12] P. Fryzlewicz. Unbalanced haar technique for nonparametric function estimation. *Journal of the American Statistical Association*, 102(480):1318–1327, 2007.

[13] P. Fryzlewicz et al. Wild binary segmentation for multiple change-point detection. *The Annals of Statistics*, 42(6):2243–2281, 2014.

[14] S. I. Goldberg, A. Niemierko, and A. Turchin. Analysis of data errors in clinical research databases. In *AMIA*, volume 2008, page 242, 2008.

[15] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI*, pages 59–63, 2012.

[16] J. Hardin and D. M. Rocke. Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. *Computational Statistics & Data Analysis*, 44(4):625–638, 2004.

[17] J. He, E. Veltri, D. Santoro, G. Li, G. Mecca, P. Papotti, and N. Tang. Interactive and deterministic data cleaning. In *SIGMOD*, pages 893–907, 2016.

[18] Z. He, S. Deng, and X. Xu. Outlier detection integrating semantic knowledge. In *WAIM*, pages 126–131, 2002.

[19] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[20] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Loop: local outlier probabilities. In *CIKM*, pages 1649–1652, 2009.

[21] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *PVLDB*, 9(12):948–959, 2016.

[22] A. Lavin and S. Ahmad. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In *ICMLA*, 2015.

[23] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *KDD*, pages 157–166, 2005.

[24] K.-H. Le and P. Papotti. An active learning method for robust error detection in time series. Technical report, EURECOM, Data Science Department, 2019. http://www.eurecom.fr/~papotti/iotDetectTr.pdf.

[25] A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[26] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: A novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, 2007.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *ICDM*, pages 413–422. IEEE, 2008.

[28] M. Mahdavi, Z. Abedjan, R. C. Fernandez, et al. Raha: A configuration-free error detection system. In *SIGMOD*, 2019.

[29] K. G. Mehrotra, C. K. Mohan, and H. Huang. *Anomaly Detection Principles and Algorithms*. Springer, 2017.

[30] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[31] T. Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

[32] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang. Time-series anomaly detection service at microsoft. In *KDD*, pages 3009–3017, 2019.

[33] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet. Anomaly detection in streams with extreme value theory. In *SIGKDD*, pages 1067–1075, 2017.

[34] S. Song, A. Zhang, J. Wang, and P. S. Yu. Screen: Stream data cleaning under speed constraints. In *SIGMOD*, pages 827–841, 2015.

[35] J. Tang, Z. Chen, A. W.-c. Fu, and D. Cheung. A robust outlier detection scheme for large data sets. In *PAKDD*, 2001.

[36] C. Truong, L. Oudre, and N. Vayatis. ruptures: change point detection in python. *arXiv preprint arXiv:1801.00826*, 2018.

[37] O. Vallis, J. Hochenbaum, and A. Kejariwal. A novel technique for long-term anomaly detection in the cloud. In *HotCloud*, 2014.

[38] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li. Ai^2: training a big data machine to defend. In *BigDataSecurity*, 2016.

[39] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan. Statistical techniques for online anomaly detection in data centers. In *Integrated Network Management*, pages 385–392, 2011.

[40] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In *TheWebConf*, pages 187–196, 2018.

[41] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *PVLDB*, 4(5):279–289, 2011.

[42] A. Zhang, S. Song, J. Wang, and P. Yu. Time series data cleaning: from anomaly detection to anomaly repairing. *PVLDB*, 10(10):1046–1057, 2017.