A Semi-supervised Interactive Algorithm for Change Point Detection

Zhenxiang Cao[1*], Nick Seeuws[1], Maarten De Vos[1,2] and Alexander Bertrand[1,3]

[1*]STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Department of Electrical Engineering, KU Leuven, Belgium.
[2]Department of Development and Regeneration, KU Leuven, Belgium.
[3]Leuven.AI-KU Leuven institute for AI.

*Corresponding author(s). E-mail(s): zhenxiang.cao@kuleuven.be; Contributing authors: nick.seeuws@kuleuven.be; maarten.devos@kuleuven.be; alexander.bertrand@kuleuven.be;

Abstract

The goal of change point detection (CPD) is to identify abrupt changes in the statistics of signals or time series that reflect transitions in the underlying system's properties or states. While many statistical and learning-based approaches have been proposed to address this task, most state-of-the-art methods still treat this problem in an unsupervised setting. As a result, there is often a large gap between the algorithm-detected results and the expected outcomes of the user. To bridge this gap, we propose an active-learning strategy for the CPD problem that combines with the one-class support vector machine (OCSVM) model, resulting in an interactive CPD algorithm (ICPD) that improves itself by querying the end-user. This approach enables us to focus on detecting the desired change points and ignore false-positives or irrelevant change points. We demonstrate that the interactive OCSVM model can be combined with various unsupervised CPD models to function in a semi-supervised setting, resulting in improved detection accuracy. Our experimental results on various simulated and real-life datasets demonstrate a significant improvement in detection performance on both single- and multi-channel time series, even with a limited number of queries.

Keywords: Active learning, Change point detection, One-class support vector machine

1 Introduction

Time series data are sequences in which the values of one or many specific statistical variables are arranged according to their occurrence in time. The values of these variables are determined by the states of some underlying systems that generate the data. Usually, abrupt changes can be observed in the time series data, which reflect the transition of the underlying system states. Change point detection (CPD) refers to the task of localizing the time points where these abrupt changes occur. The CPD problem has been an actively investigated research topic in wide-ranging application areas, including biomedical signal processing (Bosc et al., 2003; Malladi et al., 2013; Staudacher et al., 2005; Yang et al., 2006), video segmentation (Li et al., 2019; Shou et al., 2021), transportation optimization (Liu et al., 2021; Reddy et al., 2010), climatology(Ducré-Robitaille et al., 2003; Itoh and Kurths, 2010; Reeves et al., 2007), and human activity recognition (Chandola and Vatsavai, 2010; Chib, 1998; Cho and Fryzlewicz, 2015; Cleland et al., 2014), amongst others.

Based on whether the ground truth information is accessible during the fitting process, most existing CPD approaches aim to detect the change points (CPs) in either a supervised or unsupervised context. Methods under the supervised setting treat the CPD task as a multi-class (Reddy et al., 2010) or binary (Desobry et al., 2005) classification problem. Although these approaches present satisfying detection performance on their evaluation datasets, two disadvantages limit the deployment of such supervised CPD methods in real use cases: First, acquiring complete fully-labeled datasets is usually time-consuming or, in some cases, even unaccomplishable. Second, the features selected for training are usually relevant to specific applications, which makes these approaches hard to generalize to other datasets.

In contrast, unsupervised CPD approaches (Appel and Brandt, 1983; Basseville et al., 1993; Brandt, 1983; De Ryck et al., 2021; Ebrahimzadeh et al., 2019; Lee et al., 2018; Liu et al., 2013; Munir et al., 2018; Perslev et al., 2019; Shi and Chehade, 2021; Zhang et al., 2020) localize the CPs by monitoring the (dis)similarity between the consecutive time windows. A CP candidate is reported automatically when the dissimilarity between two adjacent time windows exceeds a threshold. Compared to supervised CPD methods, these unsupervised approaches avoid the reliance on fully-labeled datasets and improve the generalization ability to different types of time series data. However, because of the absence of ground truth labels, these models tend to underperform compared to supervised approaches. Furthermore, unsupervised models cannot access the information about which type of CPs the user is interested in. As a result, many undesired CP candidates will be reported, and some not-so-obvious but essential changes will be ignored.

With the aim to bridge the gap between supervised and unsupervised CPD methods, this paper proposes a semi-supervised method for CPD based on an active-learning strategy. However, when treating CPD as a classification problem, the two classes (change points versus non-change points) are by definition highly imbalanced. Instead, we investigate the CPD problem from the viewpoint of outlier detection because the one-class classification models designed for outlier detection inherently have their advantages in dealing with imbalanced data (Bellinger et al., 2012). In the CPD context, one-class classification models can capture some common properties shared by time windows that do not contain any type of changes. Then, all outlier samples with change-relevant information can be detected automatically. In our implementation, we adopt the one-class support vector machine (OCSVM) (Schölkopf et al., 1999) as the core model in our interactive CPD (ICPD) algorithm to locate the CP candidates. Furthermore, we introduce an active-learning strategy into the learning process of the OCSVM model to leverage the information of users' queries.

The contributions of our work can be summarized as follows:

- We adopt a one-class classification model, i.e., the OCSVM model, as the core classifier and investigate the CPD task in an outlier detection framework to overcome the problem caused by the inherently imbalanced training data in CPD tasks.
- The proposed ICPD model can take the user into the learning loop and report only the location of CPs belonging to the types desired by the user.
- The proposed ICPD model can be easily combined with various unsupervised CPD models, enabling them to function in a semi-supervised setting and achieve significantly improved detection performance on diverse simulated and real-life datasets.

The remainder of the paper is organized as follows: We introduce related topics in Section 2. In Section 3, we detail the proposed ICPD algorithm. We describe the experiments on simulated and real-life datasets in Section 4, and present the corresponding experiment results in Section 5. In Section 6, we present an ablation study to gain a deeper insight into the inner workings of the ICPD algorithm. Then, in Section 7, we discuss potential limitations of the proposed ICPD algorithm and draw conclusions in Section 8.

2 Related Work

2.1 Change point detection

Over the past several decades, various approaches to change-point detection (CPD) have been proposed (Aminikhanghahi and Cook, 2017; Truong et al., 2020). Unsupervised CPD methods in particular have gained traction due to their generalization ability and applicability in various fields. For example, the generalized likelihood ratio (GLR) algorithm (Appel and Brandt, 1983; Brandt, 1983) reports the locations of CPs by comparing distribution models of two adjacent intervals. The relative unconstrained least-squares importance fitting (RuLSIF) model (Liu et al., 2013) estimates the relative density ratio instead of the real distributions to avoid defining an improper distribution model. The kernel change-point detection (KLCPD) framework (Chang et al., 2019) identifies CPs via the kernel two-sample test and introduces an auxiliary generative model to optimize a lower bound of test power. The autoencoder-based breakpoints detection (ABD) approach (Lee et al., 2018) trains an autoencoder to automatically extract features and detects CPs by tracking dissimilarity between these features. The time-invariant representation (TIRE) model improves upon the ABD approach by introducing regularization in the latent space to promote time-invariant features, which are more informative for CPD tasks.

Recently, a semi-supervised CPD method named ALCPD was proposed (De Brabandere et al., 2022a). In this method, the CPD problem is treated as a 2-class classification problem for which a supervised (random forest) classifier is trained based on pseudo-labels from the TIRE CPD algorithm, in combination with an active learning strategy. This strategy allows to iteratively retrain both the TIRE and the random forest classifier. The former then allows to find new undetected candidate change points, whereas the latter allows to reduce the amount of false positives. However, these two models work independently, so the benefit of reducing false-positive detections (the objective of the random forest classifier) is often counteracted by the requirement to find new candidates (the goal of the detection model). Additionally, the sparsity of CPs leads to imbalanced training data for the random forest, resulting in only modest improvements in detection performance. Compared to ALCPD, our ICPD approach combines an active learning strategy with a single one-class model (i.e., OCSVM), which accelerates the training process, avoids the conflict between the goals of two independent models, and solves the problem caused by imbalanced training data.

Finally, we note that CPD approaches are sometimes categorized as online versus offline algorithms. Online algorithms, e.g., Bayesian online CPD (Adams and MacKay, 2007), RuLSIF (Liu et al., 2013), concept drift detection (Oikarinen et al., 2021), and real-time network (Gupta et al., 2022), aim to localize CPs as soon as they appear in real-time settings based only on passed data. In contrast, offline algorithms, e.g., KLCPD (Chang et al., 2019), ABD (Lee et al., 2018), TIRE (De Ryck et al., 2021), and ALCPD (De Brabandere et al., 2022a), detect CPs retrospectively. In this work, we only evaluate our proposed algorithm in an offline setting, although our ICPD approach could in principle also be deployed in an online setting.

2.2 Outlier detection

Outlier detection (OD) or anomaly detection (AD) algorithms aim to automatically discover patterns that deviate significantly from other observations (Chandola et al., 2009). The CPD task could therefore in principle be framed as a special case of OD in which the change points are treated as anomalies or outliers compared to all other time points where no such changes occur. This viewpoint of treating CPD as an OD task is a core of our ICPD method.

Clustering-based OD methods assume that anomalies occur far from normal samples in the data or feature space. Local outlier factor (Breunig et al., 2000) detects anomalies by comparing the local density of target data samples and their neighbors. Isolation forest (Liu et al., 2008) builds a random forest consisting of isolation trees, in which each leaf node contains only one data sample. Anomalies are identified by assuming that outliers are easier to isolate compared to normal samples. Support vector data description (Tax and Duin, 2004) and OCSVM (Schölkopf et al., 1999) adopt the ideas from support vector machines (Cortes and Vapnik, 1995) to produce a decision boundary between normal samples and anomalies.

Autoencoder-based OD methods assume that anomalies are harder to be compressed and reconstructed compared to normal observations. Robust deep autoencoder (Zhou and Paffenroth, 2017) proposes a loss function consisting of a reconstruction term and a regularization term to encourage the autoencoder to be robust enough for noisy training data. Variational autoencoder for OD (An and Cho, 2015) builds a variational autoencoder to reconstruct input data and identifies anomalies based on the reconstruction probability. To some degree, autoencoder-based CPD algorithms like ABD (Lee et al., 2018), TIRE (De Ryck et al., 2021), and real-time network (Gupta et al., 2022) can be regarded as extensions of ideas for autoencoder-based OD.

Similar to CPD, the OD task also faces problems caused by imbalanced data because the occurrence frequency of anomalies is low compared to normal observations. To solve this problem, many OD algorithms are inherently good at dealing with imbalanced data, such as isolation forest and OCSVM. Furthermore, the OCSVM model can separate the normal observations from all kinds of anomalies (CPs in our CPD context) without the need to explicitly model and distinguish the different types of anomalies. Therefore, we will adopt the OCSVM model (Schölkopf et al., 1999) as our core component in the ICPD algorithm.

3 Proposed ICPD Algorithm

In this section, we present the ICPD algorithm, which is summarized in Algorithm 1, and explain the role of its different sub-components. The ICPD algorithm starts with an initial set of candidate change points (CCPs) identified by a generic unsupervised CPD algorithm, which are then used as pseudo-labels to train an OCSVM. The latter uses features that describe changes in the statistics of the samples within two consecutive time windows. Then, the active-learning strategy is applied to refine the list of CPs in order to improve the detection accuracy.

3.1 Problem definition

The goal of CPD is to identify the time points at which changes occur in the observed values of a time series or its underlying model states. Therefore, we can define the problem as follows: given a time series $\mathbf{X} \in \mathbb{R}^{C \times N}$, where C and

Algorithm 1: The full ICPD algorithm

```
Input:
$\mathbf{X} \in \mathbb{R}^{C \times N}$: C-channel time series;
$n$: Window size;
$b$: Budget for active-learning rounds;
$r$: The number of queries between re-training steps of the OCSVM model;
$M_{init}$: Initialization model.
Output:
$T$: Detected CCP set.
1  Obtain initial CCP set: $T_{init} = M_{init}(\mathbf{X})$;
2  Extract feature matrix $\mathbf{F}$ as defined in (2);
3  Generate $\mathbf{F}_0$ by removing samples within the neighbor field (as defined in
     (3)) of $T_{init}$ from $\mathbf{F}$;
4  Train OCSVM model with $\mathbf{F}_0$ and apply it to $\mathbf{F}$ to obtain $\mathbf{s}_0$ as in (1);
5  Smoothen $\mathbf{s}_0$ to obtain $\widetilde{\mathbf{s}}_0$ as in (4);
6  Collect all peaks of $\widetilde{\mathbf{s}}_0$ in $T_0$;
7  Set $Q = \emptyset$ and $P = \emptyset$;
8  while $m = 1, \ldots, b$ do
9      Select query $q_m$ from $T_{m-1}$ (refer to Subsection 3.4.5 for more details);
10     $Q = Q \bigcup q_m$;
11     if $q_m$ is a TP then
12         Ask actual location $\bar{q}_m$ of $q_m$;
13         Remove samples within neighbor field (as in (3)) of $\bar{q}_m$ from $\mathbf{F}_{m-1}$;
14         $P = P \bigcup \bar{q}_m$;
15     else
16         Add all samples within neighbor field (as in (3)) of $q_m$ to $\mathbf{F}_{m-1}$
             (unless they are already in $\mathbf{F}_{m-1}$);
17     if $m \mod r = 0$ then
18         Retrain the OCSVM model with $\mathbf{F}_m$ and apply it to $\mathbf{F}$ to obtain $\mathbf{s}_m$
             as in (1);
19         Smoothen $\mathbf{s}_m$ to obtain $\widetilde{\mathbf{s}}_m$ as in (4);
20         Collect all peaks of $\widetilde{\mathbf{s}}_m$ in $\hat{T}_m$;
21         Remove elements within neighbor field (as in (3)) of elements in $Q$
             from $\hat{T}_m$ and store the remaining samples in $T_m$;
22         Add samples in $P$ to $T_m$ (unless they are already in $T_m$);
23     else
24         Keep $T_m = T_{m-1}$ and $\widetilde{\mathbf{s}}_m = \widetilde{\mathbf{s}}_{m-1}$;
25 Denote $T_m$ as $T$;
26 return $T$
```

N denote the number of channels and the total time steps in $\mathbf{X}$, respectively, our objective is to identify a set of time points $T = \{t_1, t_2, \ldots, t_k\}$ where the statistics or trend of the time series changes. For ease of expression, in this paper, we will use t_i to represent both the i-th detected CCP and its corresponding temporal location.

3.2 OCSVM model

As a variant of the support vector machine (SVM) (Cortes and Vapnik, 1995), the OCSVM (Schölkopf et al., 1999) model is widely used in the outlier detection problem. As opposed to the classical SVM method, the OCSVM only models a single class, and aims to detect outliers that do not belong to this target class. The underlying idea behind the OCSVM model is to map the input features of data samples onto a high-dimensional space, where all non-polluted normal samples are supposed to locate near the origin point, while the outliers are expected to be far from them. The goal is to create a decision boundary that can separate the normal and abnormal samples in this high-dimensional space. An unseen sample is classified as a typical sample or an outlier based on which side of the boundary it falls on. In addition, the OCSVM model outputs a vector $\mathbf{c}$, where each element in $\mathbf{c}$ represents the probability that a sample belongs to the normal class (an observation without change in our CPD setting) according to the OCSVM model.

In the context of CPD, our goal is to train an OCSVM model that maps non-change samples close to the origin, while keeping all types of change samples far away. To ensure that the features can capture change-related information, we use the subtraction of signal features extracted from time intervals before and after each time step as input features for the OCSVM model (see Subsection 3.4.1). In the absence of CPs, these subtractions should result in vectors containing mostly small values. By training the OCSVM model using only the subtracted features of non-change samples, it can learn the distribution of all non-change samples and map observations containing any type of change-related information far from non-change samples.

In the active-learning step described in Section 3.4, we will use a certainty score vector $\mathbf{s}$, which is computed based on the $\mathbf{c}$ vector obtained from the OCSVM model to account for temporal position information and to localize the CPs. This certainty score vector is defined as:

$$\mathbf{s} = max(\mathbf{c}) - \mathbf{c}, \tag{1}$$

which is a measure of the certainty that the corresponding sample does *not* belong to the target class.

3.3 Detecting the initial set of CCPs

During the initialization phase, the algorithm lacks access to any ground truth labels. Therefore, the goal is to extract an initial set of CCPs that can be used to train the OCSVM model without ground truth-relevant information while still achieving acceptable detection performance. This will establish a starting point for subsequent active-learning loops. To extract these initial CCPs, we can use any unsupervised CPD method. In our experiments, we employ four state-of-the-art unsupervised CPD algorithms as the initialization model M_{init} in the ICPD algorithm, namely GLR (Appel and Brandt, 1983; Brandt, 1983), RuLSIF (Liu et al., 2013), KLCPD (Chang et al., 2019), and TIRE (De Ryck et al., 2021). As a result, the selected unsupervised CPD algorithm provides an initial set of CCPs T_{init}.

The OCSVM model has a strict requirement for the cleanliness of the training datasets. The ideal training set should not contain any outlier samples. Since the OCSVM model constructs a model to describe the commonalities of normal data samples, an outlier in the training set can mislead the training of the model. To keep the training set as clean as possible, we remove all samples near the CCPs T_{init} obtained from M_{init} to avoid contamination. Only samples that are not located in the neighborhood of elements in T_{init} are used to train the OCSVM model. The definition of this neighborhood will be further explained in Section 3.4.

3.4 Active-learning

After initialization, an initial set of CCPs T_{init} detected by the initialization model is obtained. In this subsection, we aim to train the OCSVM model with an active-learning strategy to involve users in the learning loop and improve detection performance. To accomplish this task, we continuously ask users for queries to obtain more helpful information and retrain the OCSVM model after collecting a certain amount r of queries until a predefined query budget b ($b > r$) is used.

The details of the active-learning process of the ICPD algorithm can be explained as follows:

3.4.1 Feature extraction

Instead of using raw sensor data, the OCSVM model requires well-designed features as inputs. Therefore, an efficient feature extraction system capable of single- and multi-channel time series is indispensable.

We use an automatic feature generator, TSfuse[1], which automatically generates features from time series based on a pre-defined set of transformer operations and their combinations (De Brabandere et al., 2022b). To limit the feature dimensions, we employ TSfuse with only a minimal set of simple statistical transformers to extract the feature matrix $\mathbf{G} \in \mathbb{R}^{(N-n)\times l}$, where l denotes the number of statistical features extracted by the TSfuse system, and n represents the pre-defined window size over which the features are computed. To ensure that the extracted TSfuse features reflect changes in the statistics and are relevant for CPD, we implement the following operations: for a time point t, we first extract a feature $\mathbf{g}_t^{before}$ from the temporal interval $[t-n, t]$ as well as a feature $\mathbf{g}_t^{after}$ from the temporal interval $[t, t+n]$. Then, the feature $\mathbf{f}_t$ corresponding to time step t is computed as:

$$\mathbf{f}_t = \mathbf{g}_t^{after} - \mathbf{g}_t^{before}. \tag{2}$$

[1] https://github.com/arnedb/tsfuse

Then, a new feature matrix $\mathbf{F} = [\mathbf{f}_n, \mathbf{f}_{n+1}, \ldots, \mathbf{f}_{N-n-1}]^T \in \mathbb{R}^{(N-2n)\times l}$ is obtained.

3.4.2 Training set construction

As discussed above, the OCSVM model expects a training set with as few outliers as possible to achieve satisfactory detection performance. In this section, we will discuss the procedure to select appropriate samples from $\mathbf{F}$ and construct training sets for the OCSVM model in each round.

During the first training process of the OCSVM model, an initial set of CCPs T_{init} was collected from the initialization phase. All elements within this set can be considered as initial "obvious" CPs, although some of them may be false-positives. From the perspective of the OCSVM model, these elements, together with samples in their neighbor field, should be treated as outlier samples. Specifically, we define the neighbor field of a selected element at time point t_q as the samples located in the temporal interval:

$$[t_q - n, t_q + n], \tag{3}$$

where n denotes the pre-defined window size. Note that we select the same window size as we did for feature computation to minimize the number of hyperparameters. In order to avoid contamination, we remove all features corresponding to the elements in T_{init} and their neighborhoods (within n samples on both sides) from the full feature matrix $\mathbf{F}$. The new, smaller feature matrix is denoted as $\mathbf{F}_0$, which implies that this matrix is used as the training set of the OCSVM model before collecting the first query.

In the following active-learning round m, we update the training feature set $\mathbf{F}_m$ for the OCSVM model based on the annotations of the queried samples by the user (the querying strategy will be defined in Subsection 3.4.5). We assume that each retraining of the OCSVM model involves asking r queries to the user ($r < b$). When asking queries, the user is required first to answer whether the given query q_m is a real CP. If the query q_m is annotated as a true-positive sample (TP), then the user needs to point out the exact location of the corresponding ground truth CP $\bar{q}_m$. In this case, the samples in the neighbor field (defined in (3)) of the ground truth $\bar{q}_m$ are removed from $\mathbf{F}_{m-1}$ to get the updated training set $\mathbf{F}_m$ for the OCSVM model. Otherwise —q_m is a false-positive sample (FP)— we add the samples in the neighbor field of q_m, which are missed in $\mathbf{F}_{m-1}$, to $\mathbf{F}_m$. We define a set P to keep track of all queried CPs, which contains all $\bar{q}$ across different rounds. After collecting all r queries, the OCSVM model is retrained with the updated feature set again to take advantage of the obtained information from the user queries. A smaller value of r implies a more frequent updating of the model, at the cost of an increased computational burden.

3.4.3 Candidate change points localization

After training the OCSVM model on the feature set $\mathbf{F}_m$, we apply the well-trained model to the full feature set $\mathbf{F}$ and obtain the score vector $\mathbf{s}_m$. This vector reflects the predictive certainty for each sample. However, we do not use the classification results directly because the OCSVM model classifies all feature samples in $\mathbf{F}$ without considering the temporal information. This often results in poor consistency or smoothness within short temporal intervals, and many false-positive detection alarms are generated. To ensure the temporal smoothness of the certainty scores, we apply a zero-delay weighted moving average filter to the score vector $\mathbf{s}_m$:

$$\widetilde{\mathbf{s}}_m[i] = \sum_{j=-n+1}^{n-1} \mathbf{v}[n-j] \cdot \mathbf{s}_m[i+j-1], \tag{4}$$

where $\mathbf{v}$ denotes a triangular-shaped weighting window defined as $\mathbf{v}[j] = \mathbf{v}[2n-j] \overset{\text{def}}{=} j/n^2$ for $1 \leq j \leq n$ and $\widetilde{\mathbf{s}}_m$ denotes the averaged certainty score vector. After applying the zero-delay weighted moving average filter, each element in $\widetilde{\mathbf{s}}$ is determined not only by the element at the same time step in $\mathbf{s}_m$ but also by its adjacent neighbors. This ensures local consistency along the temporal axis within $\widetilde{\mathbf{s}}$. Subsequently, we collect the location of local maxima in $\widetilde{\mathbf{s}}_m$ corresponding to points of high confidence and denote them as an intermediate set $\hat{T}_m$.

Notably, the CCPs in $\hat{T}_m$ are determined by identifying the peaks in the certainty score vector $\widetilde{\mathbf{s}}_m$. This indicates that the collection $\hat{T}_m$ comprises the smallest peaks, which are those for which the OCSVM has the least confidence that they are true CPs and, hence, have the greatest probability of being false-positives. One may argue that these little peaks represent data for which the OCSVM is very certain that they are *not* CPs, and that it would be preferable to exclude peaks smaller than a particular threshold ω. However, we shall demonstrate empirically in Subsection 6.3 that establishing such a threshold does not enhance outcomes, which is why it was omitted from the final method.

3.4.4 Query retainment

So far, we have introduced the main operations involved in the active-learning rounds to collect CCPs. However, each training process of the OCSVM model is still somewhat independent of its previous training procedures. For instance, even if a query q_t has been marked as a correctly detected CP (or a false-positive alarm) in the m-th round, it is still possible for the OCSVM model to predict a time point t as a non-CP (or a CCP) after the next training process.

To address this issue, we introduce the query retainment procedure after each training of the OCSVM model in the active-learning process. In this procedure, we adjust the elements in $\hat{T}_m$ based on queried samples in the Q and P sets. Specifically, we compare the detected samples in $\hat{T}_m$ with the queried samples in Q. If some peaks in $\hat{T}_m$ are located in the neighbor field (as defined in Equation (3)) elements in Q, we delete these peaks (including the peaks located at the current element in Q, as the corresponding ground truth samples are already stored in P) from $\hat{T}_m$. This operation helps to avoid too many false-positive alarms by removing CCPs that are too close to already detected CCPs. Additionally, if some queried samples marked as real CPs (elements in P) are not present in $\hat{T}_m$, we manually add these samples to the set of peaks. Finally, we take the adjusted peak set T_m as our ultimate detection results in the m-th round.

3.4.5 Query selection

In each round m, we select the sample in the detected CP set T_{m-1} with the smallest smoothed certainty score $\widetilde{\mathbf{s}}_{m-1}$ as our query q_m. To ensure that we do not query the same sample twice, we maintain a query set Q to keep track of all previous queries. For each iteration, we choose a query that is not in the neighborhood of any sample in Q as defined in (3).

4 Experiment

In this section, we describe the experiment details for evaluating our proposed ICPD algorithm.

4.1 Benchmark Datasets

To evaluate the detection performance of our proposed ICPD algorithm and other baseline methods, we utilize four simulated datasets and three real-life datasets.

We use the same simulation set up as proposed in (De Ryck et al., 2021). The first three simulated data sets are all generated with the following auto-regressive model in which specific manipulations will be added:

$$s(t) = a_1 s(t-1) + a_2 s(t-2) + \epsilon_t, \tag{5}$$

where the error term follows a Gaussian distribution $\epsilon_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$. We fix $s(1) = s(2) = 0$, $a_1 = 0.6$, $a_2 = -0.5$, $\mu_t = 0$, and $\sigma_t = 1.5$, unless explained otherwise.

Jumping Mean (JM): This simulated dataset is produced by manually adjusting the mean value μ_t in the error term to a different one (as described in (6)) at each CP:

$$\mu_t = \begin{cases} 0, & 1 \le t \le t_1 \\ \mu_{t_{n-1}} + n/16, & t_{n-1}+1 \le t \le t_n, \end{cases} \tag{6}$$

where t_i denotes the temporal location of i-th CP.

Scaling Variance (SV): This dataset is generated via resetting the value of

σ_t in the error term (as described in (7)) at each CP:

$$\sigma_t = \begin{cases} 1, & t_{n-1}+1 \leq t \leq t_n \text{ and } n \text{ odd} \\ ln(e+n/4), & t_{n-1}+1 \leq t \leq t_n \text{ and } n \text{ even.} \end{cases} \tag{7}$$

Changing Coefficients (CC): In this dataset, we set $a_2 = 0$, while a_1 are sampled alternatively from two uniform distributions, i.e., $\mathcal{U}([0, 0.5])$ and $\mathcal{U}([0.8, 0.95])$, at each CP.
Gaussian Mixture (GM): Different from the previous three simulated datasets, the Gaussian mixture dataset is generated by alternatively sampling from two different Gaussian mixtures, i.e., $0.5\mathcal{N}(-1, 0.5^2) + 0.5\mathcal{N}(1, 0.5^2)$ and $0.8\mathcal{N}(-1, 1.0^2) + 0.2\mathcal{N}(1, 0.1^2)$.

In addition, we also include three real-life datasets that contain segments corresponding to various states of the underlying system. The boundaries between consecutive segments are targets of CPD algorithms.
Honeybee Dance: The Honeybee dance dataset is widely used as a benchmark for CPD methods (Chang et al., 2019; Cheng et al., 2020; Turner, 2012; van den Burg and Williams, 2020; Xuan and Murphy, 2007). This dataset is recorded by tracking the movements of a bee during a "dance" (to communicate messages to each other). In total, six sequences with varying lengths from 602 to 1124 time samples are collected. Each sequence consists of three channels representing the coordinates in the 2D plane and the angle differences, respectively. The ground truth CPs are annotated based on the different waggle dance stages.
UCI-HAR test: The UCI-HAR dataset, used for human activity recognition, serves as an evaluation dataset for CPD algorithms (Lee et al., 2018). During data recording, participants perform various activities, including walking, walking upstairs, walking downstairs, sitting, standing, and lying, while a smartphone captures three-axial linear acceleration and three-axial velocity using an embedded accelerometer and gyroscope. Each sensor signal is processed to compute statistical variables in the time and frequency domains, resulting in a time series dataset with 561 channels. The ground truth change points in the UCI-HAR dataset indicate the time instances of transitions between different human activities. In our experiments, we choose to utilize only the first 50 features of the test dataset. This decision is motivated by the high redundancy of features and the difficulty that most baseline models face in handling high-dimensional data. Despite using a reduced feature set, we have observed that these 50 features effectively capture the transitions in human activities without compromising the performance of the baseline models.
BabyECG: The BabyECG dataset records the 1-dimensional heart rate (in beats per minute) data of a 66-days old infant for one night. Besides, the sleep state information of the same infant is measured in the form of EEG (brain wave) and EOG (eye movement) simultaneously and annotated by a trained expert. The ground truth CPs correspond to the transitions between different sleep stages, including quiet sleep, between quiet and active sleep, active sleep,

Table 1: Overview of benchmark datasets

	Nr. series	Series length	Nr. channels	Nr. CPs
JM	10	4836 ∼ 4925	1	48
SV	10	4834 ∼ 4918	1	48
CC	10	4847 ∼ 4932	1	48
MG	10	4864 ∼ 4907	1	48
Honeybee Dance	6	602 ∼ 1124	3	15 ∼ 28
UCI-HAR test	1	2947	50	119
BabyECG	1	2048	1	29

and awake.
We detail the information about all benchmark datasets adopted in our evaluation in Table 1.

4.2 Metric and Baselines

In our experiment, we use the evaluation criterion from (De Ryck et al., 2021) to determine whether a CCP is a TP. A CCP *cp* is regarded as a correct detection corresponding to a ground truth *gt* if and only if it fulfills the following conditions: i) The number of time points between *cp* and *gt* should be less than a pre-defined tolerance τ. ii) No other ground truth gt' is closer to *cp* than *gt*. iii) Each ground truth can only correspond to one CCP.

We used the f1-score metric to evaluate the detection performance of the ICPD algorithm and other baseline CPD approaches, as it is widely used in CPD literature (Deldari et al., 2021; van den Burg and Williams, 2020). While the area under receiver operating characteristic (AUROC) curve is also commonly used to evaluate CPD task (De Ryck et al., 2021; Lee et al., 2018), it does not behave properly in some situations, and is therefore not used in our analysis. We discuss these limitations and artefacts of the AUROC metric in the Appendix, where we also provide visualizations to support our observations.

We evaluate the detection performance of the ICPD algorithm by combining it with four commonly used unsupervised CPD approaches for its initialization phase, namely:
GLR (Appel and Brandt, 1983; Brandt, 1983): The GLR algorithm fits an auto-regressive (AR) model for each pair of consecutive windows as well as for their union, and calculates the generalized likelihood ratio as the dissimilarity. In our implementation, we used an AR model of order 2..
RuLSIF (Liu et al., 2013): The RuLSIF measures the relative density ratio of two adjacent windows to identify the CCPs.
KLCPD (Chang et al., 2019): The KLCPD method uses a kernel two-sample test to localize the CCPs, while training an auxiliary generative model to ensure test power.
TIRE (De Ryck et al., 2021): The TIRE method trains a simple autoencoder with a single fully-connected layer in both the encoder and decoder, using the reconstruction loss and time-invariant loss terms as supervision. Then the

dissimilarity between the latent features of consecutive windows is tracked to identify CCPs.

To ensure consistency, we followed the hyperparameter settings from the original papers for each unsupervised CPD model and used the zero-delay weighted moving average filter as a post-processing step for the dissimilarity measure. For ease of expression, we denote all ICPD models in our comparison as ICPD$_{M_{init}}$, where M_{init} denotes the corresponding initialization model.

Furthermore, we compare the ICPD algorithms to the ALCPD method, which also operates in the semi-supervised setting.

4.3 Experiment details

The radial basis function (RBF) kernel is used in the OCSVM method. Additionally, two crucial hyperparameters in the OCSVM model must be carefully set: the kernel coefficient λ and the error coefficient ν. The value of ν controls the upper bound on the allowed fraction of training errors during the training of the OCSVM model. Theoretically, the values of λ and ν depend on the characteristics of the training dataset. In our implementation, we set $\lambda = 2.5$ and $\nu = 0.001$ for all benchmark datasets. In Section 5, we will demonstrate that these default hyperparameter settings can yield satisfactory detection performance on all benchmark datasets. We retrain the OCSVM model after collecting 10 queries, i.e., $r = 10$ in Algorithm 1.

In our implementation, the window size n is kept the same for all baseline algorithms but selected separately for each dataset to ensure that it is smaller than the expected temporal interval between CPs while being long enough to describe the characteristics in each segment. We selected the values for n to maximize the median f1-score across all algorithms in the comparison. The resulting values for n are $n = 40$ for all simulated datasets and $n = 15$, $n = 8$, $n = 15$ for the Honeybee Dance, the UCI-HAR test, and the BabyECG datasets, respectively. To reduce the number of adjustable hyperparameters, we set the value of the tolerance τ to be equal to the window size n defined in each dataset.

4.4 Measure of labeling effort

As the performance of the semi-supervised CPD algorithms (ALCPD and ICPD) depends on the number of queries collected from the user, we define a ratio:

$$\alpha = \frac{N_{queries}}{N_{gt}} \tag{8}$$

This ratio attempts to reflect the relative number of collected queries, where the denominator represents the number of ground truth CPs in the dataset. It is important to note that a value of $\alpha > 100\%$ does not imply that all ground truth CPs are known, as a large fraction of the queries typically contain false-positive detections. Given that the number of ground truth CPs is usually small (typically $< 2\%$ of the total samples), it is not unusual in practice to have a value of $\alpha > 100\%$. While the inclusion of false-positives in the denominator of (8) might result in a more intuitive metric for quantifying the relative number of queries, it would make α dependent on the results of the initialization, resulting in a stochastic value that is different in each run of the algorithms.

5 Results

We provide a comparison among ICPD algorithms initialized with different unsupervised CPD approaches and the different baseline models, i.e., the corresponding unsupervised CPD algorithms and ALCPD algorithm. Fig. 1 provides the f1-scores as a function of α as defined in (8). Additionally, we present the exact values of the f1-score achieved by all methods when α is set to three specific values (50%, 100%, and 150%) in Table 2 for a detailed comparison.

In Fig. 1, we can see that the f1-score achieved by both the ALCPD and ICPD algorithms increases steadily in all benchmark datasets as the number of queries increases. This observation aligns with our goal of achieving better detection performance by incorporating feedback from end-users into the training process. Notably, the ICPD algorithms consistently outperform the ALCPD baseline in all benchmark datasets. The ALCPD method exhibits an unstable learning curve, particularly in the real-life datasets, and often converges to a relatively low f1-score compared to the ICPD algorithms. These phenomena arise from the adversarial trade-off between the random forest classifier and the modified TIRE model in the ALCPD framework. Moreover, the ALCPD algorithm always starts with an empty set of CCPs when the number of queries is small since the random forest classifier requires a minimum number of queries for training.

Fig. 1 also reveals an interesting observation that in some cases, the ICPD algorithms need a few queries to surpass the corresponding unsupervised models. This can be explained as follows: From the definition of the tolerance τ in Subsection 4.2, a CCP is accepted as a TP sample if it is located in the tolerated area of a ground truth point. However, the CCPs that locate far from the ground truth point but still within the tolerated area may not provide sufficient information to initialize the training of the OCSVM in the ICPD algorithm. Therefore, we sometimes observe a drop in the initial training phase of the ICPD algorithm compared to the unsupervised model that was used for initializing the ICPD algorithm. We find that this problem can be alleviated by fine-tuning the hyperparameters in our OCSVM model for individual datasets. However, we decided to avoid such hyperparameter tuning in our experiments for the sake of fairness and transparency.

Moreover, Fig. 1 also demonstrates that the choice of the initialization algorithm for ICPD becomes less important when the number of queries is high. This phenomenon confirms our argument that the ICPD framework can be combined with different types of unsupervised CPD approaches to provide

Table 2: Mean values and standard deviations of f1-scores achieved by the ICPD and the different baselines. For each dataset, we highlight the best-performing approach in bold. The ratio α is defined in (8). Note that the standard deviation is only caused by the random initialization of the parameters in the unsupervised models. Since GLR and RuLSIF do not require a (random) initialization, the standard deviation is always equal to 0.0 for these methods.

		Simulated datasets				Real-life datasets		
Approach	α	JM	SV	CC	MG	Honeybee Dance	UCI-HAR test	BabyECG
ALCPD(De Brabandere et al., 2022a)	50%	0.938±0.012	0.836±0.032	0.893±0.008	0.983±0.002	0.496±0.122	0.305±0.126	0.491±0.052
	100%	0.946±0.002	0.928±0.004	0.903±0.008	0.984±0.003	0.573±0.053	0.700±0.098	0.505±0.054
	150%	0.945±0.002	0.932±0.006	0.902±0.007	0.980±0.002	0.790±0.025	0.790±0.019	0.556±0.060
GLR (Appel and Brandt, 1983; Brandt, 1983)		0.808±0.0	0.860±0.0	0.885±0.0	0.934±0.0	0.712±0.0	0.660±0.0	0.510±0.0
ICPD$_{\text{GLR}}$	50%	0.978±0.0	0.954±0.0	0.941±0.0	0.989±0.0	0.847±0.0	0.765±0.0	0.466±0.0
	100%	0.993±0.0	0.966±0.0	0.967±0.0	**1.0±0.0**	0.923±0.0	0.867±0.0	0.597±0.0
	150%	0.994±0.0	0.966±0.0	**0.978±0.0**	**1.0±0.0**	0.919±0.0	0.867±0.0	0.689±0.0
RuLSIF (Liu et al., 2013)		0.867±0.0	0.874±0.0	0.832±0.0	0.917±0.0	0.684±0.0	0.689±0.0	0.395±0.0
ICPD$_{\text{RuLSIF}}$	50%	0.984±0.0	0.967±0.0	0.858±0.0	**1.0±0.0**	0.785±0.0	0.714±0.0	0.535±0.0
	100%	0.993±0.0	0.976±0.0	0.937±0.0	**1.0±0.0**	0.887±0.0	0.841±0.0	0.615±0.0
	150%	0.993±0.0	0.976±0.0	0.949±0.0	**1.0±0.0**	0.912±0.0	**0.883±0.0**	**0.714±0.0**
KLCPD (Chang et al., 2019)		0.792±0.008	0.692±0.022	0.699±0.016	0.824±0.013	0.655±0.078	0.691±0.036	0.389±0.062
ICPD$_{\text{KLCPD}}$	50%	0.966±0.006	0.891±0.013	0.769±0.010	0.994±0.005	0.765±0.065	0.766±0.046	0.560±0.043
	100%	0.982±0.002	0.962±0.006	0.907±0.006	0.998±0.001	0.865±0.068	0.840±0.018	0.633±0.060
	150%	0.985±0.002	0.975±0.004	0.932±0.005	0.999±0.001	0.875±0.062	0.844±0.013	0.677±0.033
TIRE (De Ryck et al., 2021)		0.946±0.007	0.885±0.014	0.913±0.007	0.985±0.005	0.697±0.032	0.779±0.021	0.504±0.042
ICPD$_{\text{TIRE}}$	50%	0.988±0.004	0.948±0.008	0.870±0.012	0.997±0.001	0.781±0.034	0.811±0.014	0.578±0.018
	100%	0.993±0.002	0.976±0.003	0.936±0.010	0.998±0.001	0.898±0.017	0.870±0.006	0.648±0.026
	150%	**0.995±0.002**	**0.981±0.002**	0.946±0.008	0.999±0.001	**0.933±0.006**	0.881±0.003	0.690±0.015

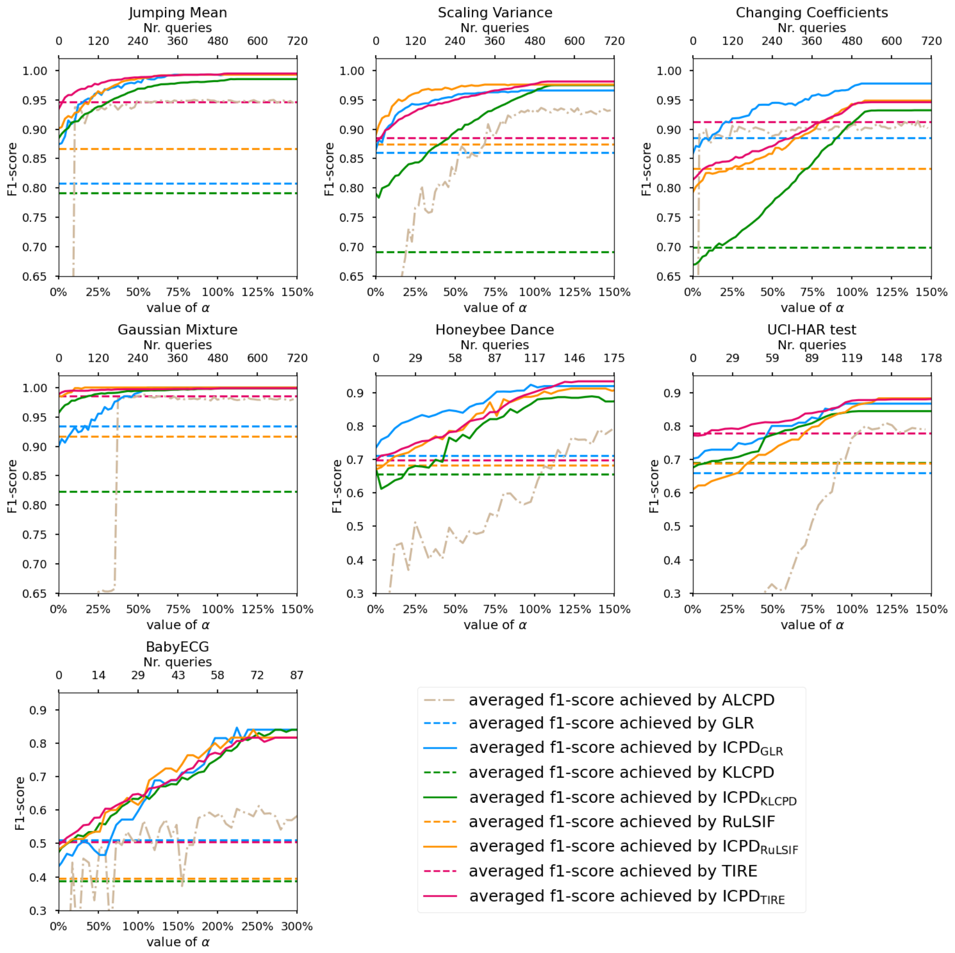

Fig. 1: Comparison between averaged f1-scores achieved by ALCPD, ICPD and corresponding initialization models on benchmark datasets across 10 repetitions. The results of ICPD are marked with solid curves while the f1-scores achieved by corresponding unsupervised initialization models are plotted with dashed flat lines in the same color.

improved detection accuracy. However, the choice of the initialization algorithm has an impact on how fast the optimal performance is reached in terms of number of queries.

After examining the results presented in Table 2, we can observe that the ICPD algorithms outperform their initialization baselines on most benchmark datasets with the $\alpha = 50\%$ setting. Furthermore, by setting $\alpha = 250\%$, the f1-scores achieved by the ICPD algorithms converge for each dataset and show significant improvements compared to the baselines.

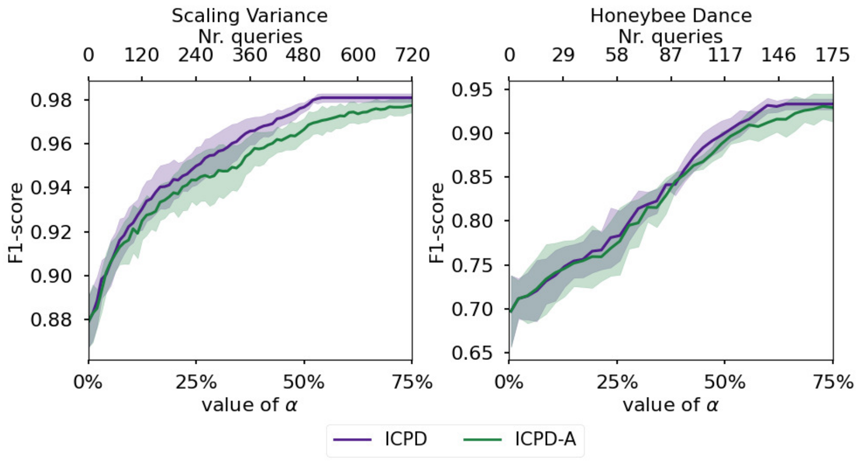

Fig. 2: Comparison between the ICPD algorithm and its variant ICPD-A. The solid lines represent the mean values over ten repetitions, while the transparent bands denote the standard deviations across these repetitions.

6 Ablation Study

In this section, we conduct a series of experiments to gain insights into the contribution of specific components in the ICPD algorithm, which lead to its improved CPD performance. For the sake of conciseness and intelligibility, we refer to the ICPD approach initialized with the TIRE model as the "ICPD" algorithm throughout the following experiments. This is because both the TIRE model and the ICPD$_{\text{TIRE}}$ algorithm typically achieve acceptable detection performance on all benchmark datasets. We keep the initialization model of the ICPD algorithm fixed at the TIRE model.

6.1 Influence of ground truth marking

As discussed in Subsection 3.4.2, the user provides the ground truth label for the queried sample as well as the actual time stamp at which the ground truth change occurs (in case of a true-positive). This subsection discusses the benefit of providing information about the actual time location of the CPs. We define another variant of the ICPD algorithm: ICPD-A, in which we cannot get access to the information about the ground truth locations and only remove the samples located in the neighbor field of the queried samples in the training set of the OCSVM model. The metrics achieved by ICPD and ICPD-A are visualized in Fig. 2.

Theoretically, the quality of the training dataset of the OCSVM model can be improved by introducing ground truth-related information. As demonstrated in Fig. 2, the ICPD algorithm performs better than the ICPD-A on both simulated and real-life datasets. Although the final f1-scores achieved by

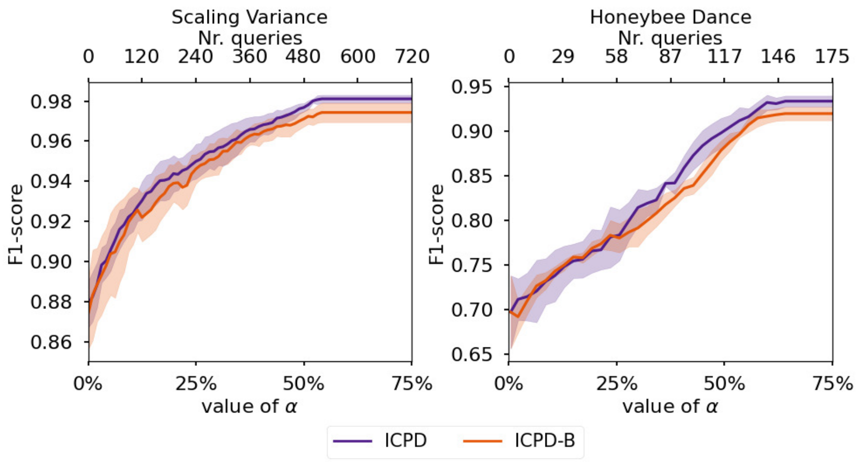

Fig. 3: Comparison between the ICPD algorithm and its variant ICPD-B. The solid lines represent the mean values over ten repetitions, while the transparent bands denote the standard deviations across these repetitions.

ICPD and ICPD-A are similar after their convergence, the ICPD converges faster than its variant ICPD-A, which implies that the ground truth marking operation contributes to providing the same detection performance on these datasets with fewer queries. As the introduction of the ground truth information never leads to a drop in detection performance on all benchmark datasets, we keep it in the proposed full algorithm.

6.2 Influence of query strategy

As discussed in Subsection 3.4.5, we always select the sample with the lowest smoothed certainty score $\widetilde{\mathbf{s}}_m$ in the detected CCP set T. In order to prove that the least certainty query strategy is meaningful, we compare it with the random selection strategy. For ease of expression, we denote the ICPD algorithm with the random selection strategy as ICPD-B. The comparison between ICPD and ICPD-B is visualized in Fig. 3.

Compared to the random selection strategy adopted in the variant ICPD-B, the least certainty query strategy in our ICPD algorithm provides a better detection performance. In other words, the ICPD algorithm can achieve the same detection performance on these datasets with fewer queries and offer a higher f1-score after convergence of the algorithm. In addition, the fluctuations in the learning curve of ICPD-B imply that the random selection strategy is not conducive to the algorithm's stability.

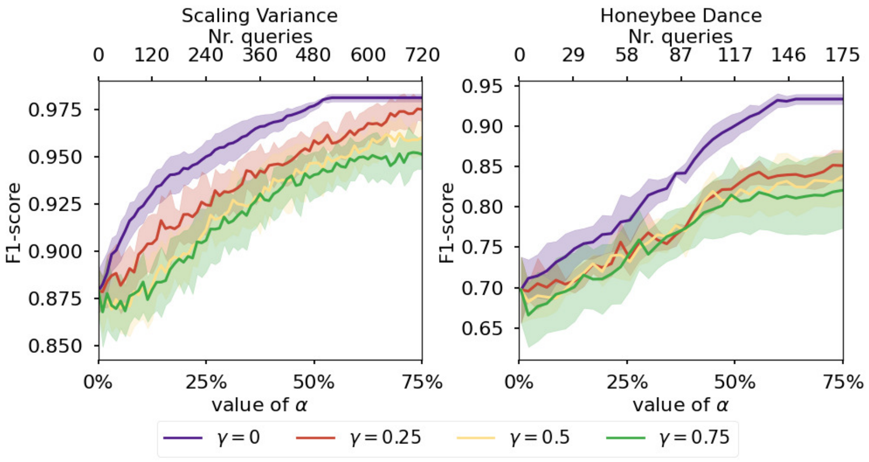

Fig. 4: Visualization of the influence of different γ values (as defined in (9)). The solid lines represent the mean values over ten repetitions, while the transparent bands denote the standard deviations across these repetitions.

6.3 Influence of setting threshold for selecting CCPs

We noted in Subsection 3.4.3 that CCPs are selected based on a peak finding algorithm across the entries in $\widetilde{\mathbf{s}}$. Therefore, there will be several spurious peaks that have a score in $\widetilde{\mathbf{s}}$ that is close to 0, i.e., for which the current OCSVM model is quite certain that they are *not* CPs. One potential improvement is to set a threshold ω on the certainty score and only select peaks whose certainty scores exceed ω in order to avoid introducing too many false-positives. To test this, we use the quantile function to define the value of ω:

$$\omega = quantile(\widetilde{\mathbf{s}}, \gamma), \tag{9}$$

where the value of γ is set as 0, 0.25, 0.5, and 0.75. Note that $\gamma = 0$ corresponds to the situation in the original ICPD algorithm.

As demonstrated in Fig. 4, introducing the threshold ω leads to a drop in the detection performance on both simulated and real-life datasets. With increasing the value of ω (γ), the f1-score achieved by our algorithm becomes worse. One possibility for this is the following: Some real CPs are not detected in the initialization phase of the ICPD algorithm, and their features are adopted to train the OCSVM model as non-change samples. Therefore, the next OCSVM model will assign relatively small certainty scores to these samples. This leads to some kind of a "self-fulfilling prophecy" mechanism such that eventually these CPs are never detected and also never queried.

Table 3: F1-score achieved by the ICPD algorithm and its variant ICPD-C after collecting a certain number of queries ($\alpha = 150\%$ as defined in (8)).

	Simulated datasets				Real-life datasets		
Approach	JM	SV	CC	GM	Honeybee Dance	UCI-HAR test	BabyECG
ICPD-C	0.977±0.004	0.948±0.004	0.905±0.008	0.997±0.003	0.856±0.030	0.829±0.005	0.599±0.064
ICPD	**0.995±0.002**	**0.981±0.002**	**0.946±0.008**	**0.999±0.001**	**0.933±0.006**	**0.881±0.003**	**0.690±0.015**

6.4 Influence of re-training OCSVM model

In the ICPD algorithm, the OCSVM model is retrained after collecting r queries from the user, which is relatively time-consuming and determines the efficiency of the entire algorithm. In this subsection, we aim to discuss whether the re-training of the OCSVM model is necessary. To this end, we define another variant of the ICPD algorithm: ICPD-C, in which the OCSVM model is only retrained after the full query budget b has been spent. In our implementation, we run the ICPD-C on all benchmark datasets with constrained query budgets corresponding to the case $\alpha = 150\%$. Then, we compare the results to ones acquired from the actual ICPD algorithm after the collecting same number of queries.

As shown in Table 3, we can observe that the introduction of the re-training mechanism in the ICPD algorithm can improve the f1-score on all benchmark datasets. This is because the re-training of the OCSVM model allows to take the labels from the previous queries into account, in order to discover new CP candidates and/or select more informative queries. Note that this allows to identify new queries that were not detected in the first training of the OCSVM model.

6.5 Comparison against Naïve correction

In theory, semi-supervised ICPD approaches can always outperform unsupervised methods because they can access valuable information provided by ground truth or end-users. In this subsection, our goal is to determine whether the ICPD algorithm is able to actually learn from the queries in order to detect other change points instead of solely relying on queried information to correct its detected results. To this end, we compare the ICPD approach to a naïve correction mechanism, in which we ask the same number of queries as in ICPD, but where we only use these to remove the false positives found by the initialization algorithm (i.e., TIRE) and align the true positives based on these queries.

As shown in Figure 5, the ICPD algorithm can outperform the naive correction mechanism, indicating that it can leverage the queried information from the end-user to improve CPD accuracy. This suggests that the method can learn from the provided information to achieve more accurate CPD performance.

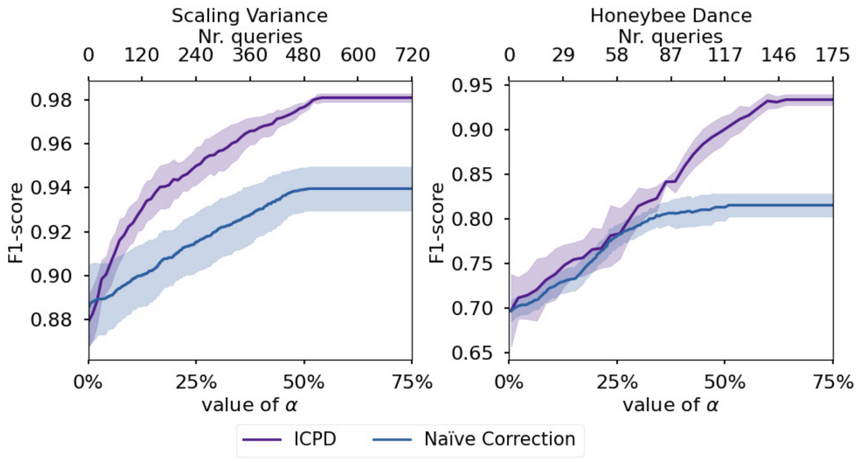

Fig. 5: Comparison between the ICPD algorithm and a naïve correction mechanism. The solid lines represent the mean values over ten repetitions, while the transparent bands denote the standard deviations across these repetitions.

7 Discussion

In this section, we discuss some potential limitations and trade-offs when using the ICPD algorithm.

First, there is a trade-off between the query budget b and the resolution with which ICPD can detect CPs, i.e., the minimum number of samples between two consecutive CPs. This is because we set a neighbor field as in (3) and exclude any CCPs that are detected within this neighbor field in future iterations. This is necessary in order to avoid that the same CP is queried multiple times, albeit at slightly different time points, thereby eating up the query budget b. Figure 6 shows an illustrative example from the BabyECG dataset. It is observed that ICPD corrects many false positives from the TIRE initialization, yet even after convergence some false negatives persist, even though they were originally found by the TIRE initialization. These false negatives always correspond to CPs that are close to another CP, and which are therefore excluded due to the use of the neighbor field. In our implementation, we allow the user to adjust the size of the neighbor field to find the optimal balance for real-world applications.

In our experiments, we always kickstart the ICPD algorithm with the results of unsupervised CPD approaches to expedite its convergence. In real-world scenarios, it is possible that the user has already collected some ground truth CPs, in which case the initialization phase of the ICPD approach can be simply replaced with the collected ground truth CPs. The performance of this initialization then depends on the size and quality of the provided ground truth set. For instance, if the collected ground truth set only contains a few types of change points existing in the entire time series, one can not expect the

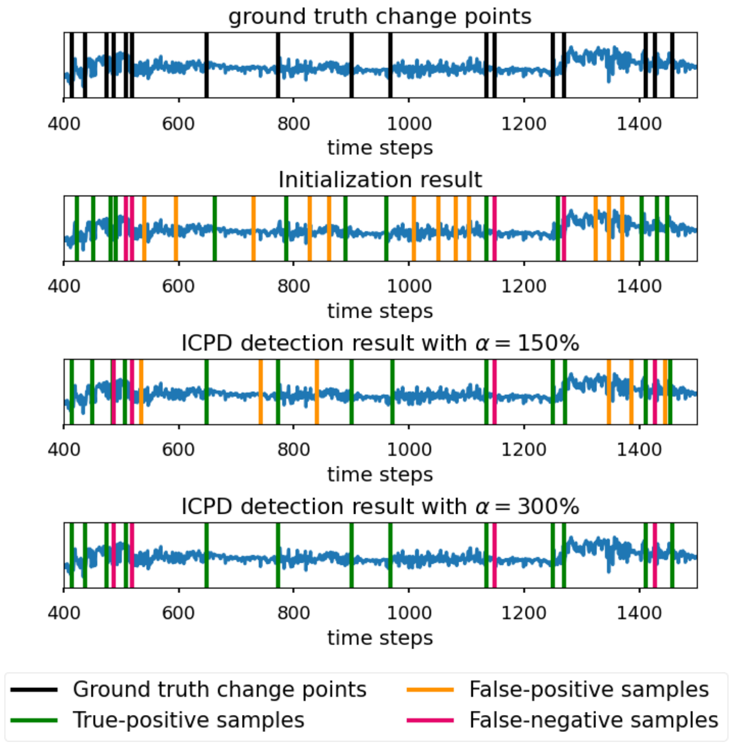

Fig. 6: The detection results of ICPD methods and its unsupervised baseline on BabyECG dataset. From top to bottom, we illustrate the ground truth location of change points, the detected CCPs by the TIRE initialization, the detected CCPs from ICPD with $\alpha = 150\%$, and the detected CCPs from ICPD after full convergence.

ICPD method to learn to detect other types of changes. The bottleneck concerning the type of CPs that the ICPD algorithm can detect also exists in our previous setting, where we employ an unsupervised method to generate the initial set of CCPs. This method establishes the initial classification boundary for the OCSVM model. However, if certain types of changes are not detected by this method, the overall ICPD algorithm is likely to overlook them as well, unless a smaller neighbor field is permitted at the expense of the query budget.

In our implementation, we assume that the end-user always provides correct annotations, which is a common assumption in the active learning setting. However, in real applications the end-user might make mistakes and offer incorrect annotations during the training process. Investigating the impact of such incorrect annotations could be an interesting topic for future research.

As mentioned in Section 2, the ICPD algorithm implemented in this study focuses on detecting change points in the offline setting. Therefore, any location in the input time series can be queried based on the certainty score assigned by the OCSVM model, using information from the entire time series (both before and after the CP). In an online setting, the OCSVM (as well as the initialization algorithm) can only uses past data for training and inference, except for a short buffer, which then introduces an algorithmic delay.

8 Conclusion

We have presented a semi-supervised interactive CPD algorithm by introducing an active-learning strategy based on an OCSVM approach. Compared to state-of-the-art unsupervised approaches, the ICPD algorithm takes the queried information from the users into account during re-training the OCSVM model. As a result, the ICPD algorithm can report more inconspicuous CPs usually neglected by other approaches and avoid introducing too many false-positive alarms by exploiting the new information provided by the users. Our exhaustive experiment results demonstrate the effectiveness of the proposed ICPD algorithm and the corresponding ablation study has provided insights into the specific features that lead to a better CPD performance on both simulated and real-life benchmark datasets.

Declarations

- Funding
  This research received funding from the Flemish Government (AI Research Program) and from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.802895). All authors are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium.

- Conflict of interest
  We declare that we have no conflict of interest.
- Code availability
  The code described in the article is available in the following repository: https://github.com/caozhenxiang/ICPD.

References

Ryan P. Adams and David John Cameron MacKay. Bayesian online changepoint detection. *arXiv: Machine Learning*, 2007.

Samaneh Aminikhanghahi and Diane J. Cook. A survey of methods for time series change point detection. *Knowl. Inf. Syst.*, 51(2):339–367, may 2017. ISSN 0219-1377. doi: 10.1007/s10115-016-0987-z. URL https://doi.org/10.1007/s10115-016-0987-z.

Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18, 2015.

Ulrich Appel and Achim V Brandt. Adaptive sequential segmentation of piecewise stationary time series. *Information sciences*, 29(1):27–56, 1983.

Michele Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs, 1993.

Colin Bellinger, Shiven Sharma, and Nathalie Japkowicz. One-class versus binary classification: Which and when? In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 102–106, 2012. doi: 10.1109/ICMLA.2012.212.

Marcel Bosc, Fabrice Heitz, Jean-Paul Armspach, Izzie Namer, Daniel Gounot, and Lucien Rumbach. Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. *NeuroImage*, 20(2): 643–656, 2003.

AV Brandt. Detecting and estimating parameter jumps using ladder algorithms and likelihood ratio tests. In *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, pages 1017–1020. IEEE, 1983.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

Varun Chandola and Ranga Raju Vatsavai. Scalable time series change detection for biomass monitoring using gaussian process. In *CIDU*, pages 69–82, 2010.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models. *arXiv preprint arXiv:1901.06077*, 2019.

Kevin C Cheng, Eric L Miller, Michael C Hughes, and Shuchin Aeron. On matched filtering for statistical change point detection. *IEEE Open Journal of Signal Processing*, 1:159–176, 2020.

Siddhartha Chib. Estimation and comparison of multiple change-point models. *Journal of econometrics*, 86(2):221–241, 1998.

Haeran Cho and Piotr Fryzlewicz. Multiple-change-point detection for high dimensional time series via sparsified binary segmentation. *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pages 475–507, 2015.

Ian Cleland, Manhyung Han, Chris Nugent, Hosung Lee, Sally McClean, Shuai Zhang, and Sungyoung Lee. Evaluation of prompted annotation of activity data recorded from a smart phone. *Sensors*, 14(9):15861–15879, 2014.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

Arne De Brabandere, Zhenxiang Cao, Maarten De Vos, Alexander Bertrand, and Jesse Davis. Semi-supervised change point detection using active learning. In *International Conference on Discovery Science*, pages 74–88. Springer, 2022a.

Arne De Brabandere, Tim Op De Beéck, Kilian Hendrickx, Wannes Meert, and Jesse Davis. Tsfuse: automated feature construction for multiple time series data. *Machine Learning*, 2022b. doi: 10.1007/s10994-021-06096-2. URL https://doi.org/10.1007/s10994-021-06096-2.

Tim De Ryck, Maarten De Vos, and Alexander Bertrand. Change point detection in time series data using autoencoders with a time-invariant representation. *IEEE Transactions on Signal Processing*, 2021.

Shohreh Deldari, Daniel V Smith, Hao Xue, and Flora D Salim. Time series change point detection with self-supervised contrastive predictive coding. In *Proceedings of the Web Conference 2021*, pages 3124–3135, 2021.

Frédéric Desobry, Manuel Davy, and Christian Doncarli. An online kernel change detection algorithm. *IEEE Transactions on Signal Processing*, 53 (8):2961–2974, 2005.

Jean-François Ducré-Robitaille, Lucie A Vincent, and Gilles Boulet. Comparison of techniques for detection of discontinuities in temperature series. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, 23(9):1087–1101, 2003.

Zahra Ebrahimzadeh, Min Zheng, Selcuk Karakas, and Samantha Kleinberg. Deep learning for multi-scale changepoint detection in multivariate time series, 2019.

Muktesh Gupta, Rajesh Wadhvani, and Akhtar Rasool. Real-time change-point detection: A deep neural network-based adaptive approach for detecting changes in multivariate time series data. *Expert Systems with Applications*, 209:118260, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.118260. URL https://www.sciencedirect.com/science/article/pii/S0957417422014026.

Naoki Itoh and Jürgen Kurths. Change-point detection of climate time series by nonparametric method. In *Proceedings of the world congress on engineering and computer science*, volume 1, pages 445–448. Citeseer, 2010.

Wei-Han Lee, Jorge Ortiz, Bongjun Ko, and Ruby Lee. Time series segmentation through automatic feature learning. *arXiv preprint arXiv:1801.05394*, 2018.

Jun Li, Peng Lei, and Sinisa Todorovic. Weakly supervised energy-based learning for action segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6243–6251, 2019.

Lishuai Li, R John Hansman, Rafael Palacios, and Roy Welsch. Anomaly detection via a gaussian mixture model for flight operation and safety monitoring. *Transportation Research Part C: Emerging Technologies*, 64:45–57, 2016.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

Wenyao Liu, Joshua Qiang Li, Wenying Yu, and Guangwei Yang. Change-point detection approaches for pavement dynamic segmentation. *Journal of Transportation Engineering, Part B: Pavements*, 147(2):06021001, 2021.

Rakesh Malladi, Giridhar P Kalamangalam, and Behnaam Aazhang. Online bayesian change point detection algorithms for segmentation of epileptic activity. In *2013 Asilomar Conference on Signals, Systems and Computers*, pages 1833–1837. IEEE, 2013.

Nour Moustafa, Gideon Creech, and Jill Slay. Anomaly detection system using beta mixture models and outlier detection. In *Progress in Computing, Analytics and Networking: Proceedings of ICCAN 2017*, pages 125–135. Springer, 2018.

Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access*, 7:1991–2005, 2018.

Emilia Oikarinen, Henri Elias Tiittanen, Andreas Henelius, and Kai Puolamäki. Detecting virtual concept drift of regressors without ground truth values. *Data Mining and Knowledge Discovery*, 35:726–747, 2021. ISSN 1384-5810. doi: 10.1007/s10618-021-00739-7.

Mathias Perslev, Michael Hejselbak Jensen, Sune Darkner, Poul Jørgen Jennum, and Christian Igel. U-time: A fully convolutional network for time series segmentation applied to sleep staging. *arXiv preprint arXiv:1910.11162*, 2019.

Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):1–27, 2010.

Jaxk Reeves, Jien Chen, Xiaolan L Wang, Robert Lund, and Qi Qi Lu. A review and comparison of changepoint detection techniques for climate data. *Journal of applied meteorology and climatology*, 46(6):900–915, 2007.

Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999.

Zunya Shi and Abdallah Chehade. A dual-lstm framework combining change point detection and remaining useful life prediction. *Reliability Engineering & System Safety*, 205:107257, 2021.

Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. Generic event boundary detection: A benchmark for event segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8075–8084, 2021.

M Staudacher, S Telser, A Amann, H Hinterhuber, and M Ritsch-Marte. A new method for change-point detection developed for on-line analysis of the heart beat variability during sleep. *Physica A: Statistical Mechanics and its Applications*, 349(3-4):582–596, 2005.

David MJ Tax and Robert PW Duin. Support vector data description. *Machine learning*, 54:45–66, 2004.

Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020. ISSN 0165-1684. doi: https://doi.org/10.1016/j.sigpro.2019.107299. URL https://www.sciencedirect.com/science/article/pii/S0165168419303494.

Ryan Darby Turner. *Gaussian processes for state space models and change point detection*. PhD thesis, University of Cambridge, 2012.

Gerrit J. J. van den Burg and Christopher K. I. Williams. An evaluation of change point detection algorithms, 2020.

Xiang Xuan and Kevin Murphy. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 1055–1062, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273629. URL https://doi.org/10.1145/1273496.1273629.

P. Yang, G. Dumont, and J.M. Ansermino. Adaptive change detection in heart rate trend monitoring in anesthetized children. *IEEE Transactions on Biomedical Engineering*, 53(11):2211–2219, 2006. doi: 10.1109/TBME.2006.877107.

Ruohong Zhang, Yu Hao, Donghan Yu, Wei-Cheng Chang, Guokun Lai, and Yiming Yang. Correlation-aware unsupervised change-point detection via graph neural networks, 2020.

Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

Appendix: Limitation of AUROC in Evaluating CPD Algorithms

We discuss the limitations of using AUROC to evaluate CPD algorithms via a toy example. Before presenting the example, we will first introduce the definition of the AUROC metric in the context of CPD.

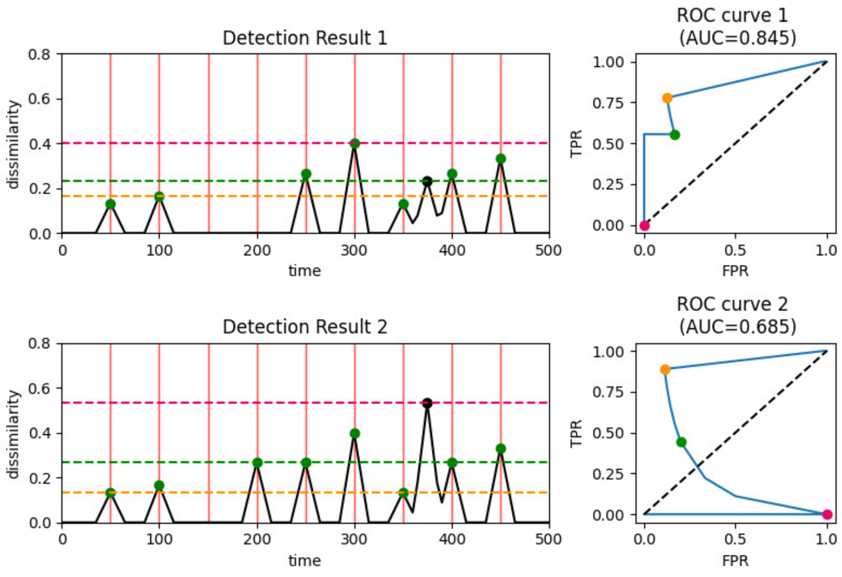

Fig. 7: Toy example for showing a limitation of the AUROC metric. In the left column, the two sub-figures simulate two detection results obtained from two hypothetical CPD algorithms on the same time series. We mark the locations of ground truth change points with the vertical lines in red. The black curve denotes the dissimilarity measurement produced by the CPD algorithm (a high value corresponds to a potential CP). Green points and black points represent true positive and false positive samples, respectively. Three colored flat dashed lines represent three values of the detection threshold υ during generating of the ROC curve. In the right column, the corresponding ROC curves are plotted. Three colored points on the ROC curves correspond to the υ values in the left sub-figures in the same colors.

Since imbalanced data is common in CPD tasks, many papers in the CPD literature (e.g., KLCPD (Chang et al., 2019), RuLSIF (Liu et al., 2013), ABD (Lee et al., 2018), and TIRE (De Ryck et al., 2021)) define the true positive rate (TPR) and false positive rate (FPR) as:

$$TPR = \frac{N_{TP}}{N_{GT}} \tag{10}$$

and

$$FPR = \frac{N_{FP}}{N_{TP} + N_{FP}}. \tag{11}$$

The ROC curve is then obtained by varying a detection threshold (v) from high to low. Point $(FPR, TPR) = (1.0, 1.0)$ is manually added to the ROC curve to ensure that a perfect performance corresponds to an AUROC of 1 (De Ryck et al., 2021).

Based on these definitions, we show an illustrative toy example in Fig. 7. Here, we show two possible detection results, e.g., by two hypothetical CPD algorithms. In Table 4, we summarize the number of samples in these two detection results and compute the f1-scores.

Table 4: Summary of detection results in toy example.

	N_{TP}	N_{FP}	N_{FN}	f1-score	AUROC
Result 1	7	**1**	2	0.824	**0.845**
Result 2	**8**	**1**	**1**	**0.889**	0.685

As shown in Table 4, Result 2 detects one more change point and one less flase negative thereby producing a better f1-score than Result 1. However, the AUROC achieved by Result 2 is much worse than that of Result 1. This is because CPD algorithm 2 assigns a higher dissimilarity value to the false positive sample located at time step 375, causing the corresponding ROC curve to start from point (1.0, 0.0), resulting in an awkward shape. In real-world applications, the dissimilarity produced by a CPD algorithm is determined locally and is usually very sensitive to outlier data samples and initial model parameters. This makes the start of the ROC curve very sensitive to stochastic effects, often leading to awkward shapes as in the second example of Fig. 7. The same CPD algorithm can even result in very different AUROC values on the same dataset due to different initial weights. This is why we chose to evaluate the CPD approaches in this study based on the f1-score.