Detecting Change Intervals with Isolation Distributional Kernel

arXiv:submit/4667504 [cs.LG] 30 Dec 2022

ABSTRACT

Detecting abrupt changes in data distribution is one of the most significant tasks in streaming data analysis. Although many unsupervised Change-Point Detection (CPD) methods have been proposed recently to identify those changes, they still suffer from missing subtle changes, poor scalability, or/and sensitive to noise points. To meet these challenges, we are the first to generalise the CPD problem as a special case of the Change-Interval Detection (CID) problem. Then we propose a CID method, named `iCID`, based on a recent Isolation Distributional Kernel (IDK). `iCID` identifies the change interval if there is a high dissimilarity score between two non-homogeneous temporal adjacent intervals. The data-dependent property and finite feature map of IDK enabled `iCID` to efficiently identify various types of change points in data streams with the tolerance of noise points. Moreover, the proposed online and offline versions of `iCID` have the ability to optimise key parameter settings. The effectiveness and efficiency of `iCID` have been systematically verified on both synthetic and real-world datasets.

Artifact Availability:
The source code, data, and/or other artifacts are available at https://github.com/IsolationKernel/Codes/tree/main/IDK/iCID.

1 INTRODUCTION

A common task in streaming data is the identification and analysis of change points that describe events of behaviour change. In statistics, a change point is a time point when an observed variable changes its behaviour in a series or process [**?**]. *Change-Point Detection* (CPD) is an analytical method for identifying behavioural mutations in a temporal sequence of observations [30]. It has attracted significant attention from data analysts because many time-critical applications demand these change points to be examined as soon as possible, e.g., system failures and abnormal network status. CPD methods have been widely used to analyse events or time-dependent anomalies in areas such as signal processing [29], network traffic analysis [18] and human activity recognition [5].

Although many unsupervised CPD methods have been proposed recently, three unresolved challenges remain: (a) poor scalability; (b) have difficulty detecting subtle-change points; and (c) intolerance to noise points. Challenge (a) must be tackled since a data stream can potentially has an infinite number of data points. A method, which is unable to deal with challenge (b), misses the often important subtle-change points in a data stream exactly because they are difficult to detect. A method, which is intolerant to noise points, is unsuitable in the real world as it will produce too many false alarms since noise points are common in data streams.

Traditional point-regression based methods, such as *Autoregressive Moving Average* (ARMA) [3], *Autoregressive Gaussian Process* (ARGP) [4] and *ARGP-BOCPD* [24], usually could not meet the challenge (c). They detect the change points by computing the gap between the true values and the predicted values of time series data. Since noise points are usually far away from the predicted normal value, these methods would inevitably misidentify them as change points. Other regression-based methods are based on interval pattern matches, such as *FLOSS* [12] and *ESPRESSO* [9]. However, they usually have a high time complex and do not meet the challenge (a), thus, they are unable to deal with large data streams.

Recent deep learning-based methods [7, 10, 15] also do not meet the challenge (a) due to training efficiency, although they are powerful to handle unstructured and multivariate data streams. They generally require a large amount of data with heavy training cost.

Morever, distribution-based methods [4, 11, 16, 17, 24, 34] usually have good scalability and tolerate noise based on kernel methods, which are effective and efficiency for solving difficult machine learning tasks. However, existing kernel-based CPD methods usually could not meet the challenge (b), since they use Gaussian kernel as a means to estimate distributions and examine the distributional difference between intervals.

Using Gaussian kernel has two key limitations:

(1) The feature map of Gaussian kernel has intractable dimensionality. As a result, the *MMD* calculation has high time cost or it can be sped up via an approximation such as the Nystrom method [32] to produce a finite-dimensional feature map;
(2) Gaussian kernel is not sensitive to the subtle change of distributions between two temporal adjacent intervals.

Figure 1 shows the results of CPD on a synthetic dataset using two kernels.[1] The streaming data contains five different distributions in five blocks (indicated by five numbers on the top) and a 5 noise points in the first two blocks. When using Gaussian kernel to measure the dissimilarity between adjacent intervals, the obvious change of distributions between blocks 3 & 4 shows high score. However, the change between blocks 4 & 5 is very subtle; the changes between block 1 and 3 could not be detected.

To meet the existing challenges of CPD methods, we propose an **I**solation Distributional Kernel-based **C**hange **I**nterval **D**etection (`iCID`) algorithm. Compared with existing CPD methods, `iCID` has the following key advantages:

(1) Able to identify three types of change points in data streams, which is defined in Section 3.1. Figure 1c shows that the proposed method can detect type I subtle-changes that are missed by using Gaussian kernel. Figure 2 visualises the dissimilarity scores between adjacent intervals based IDK and Gaussian Distributional Kernel (GDK) using MDS.[2] It shows that intervals from different blocks are more uniformly distributed with IDK-based dissimilarity, and the first two intervals are easier to be separated using IDK than using GDK.

[1] Section 5.1.2 provides the properties of S1 dataset. The details of CPD using kernel methods are provided in Section 7.1.

[2] Multidimensional scaling (MDS) [1] performs a transformation from a high-dimensional space to a 2-dimensional space for visualisation by preserving as well as possible the pairwise global distances in both spaces. We first split S1 data into intervals and then calculate the distributional dissimilarity between different intervals based on IDK and GDK. The split intervals are shown as points in Figure 2.

(2) Tolerate to noise points. The change score is calculated based on distribution/interval rather than single point. Figure 2 shows that all five manually added noise points have low changing scores regardless of using GDK or IDK.
(3) Linear time complexity of both offline and online version. The offline version of `iCID` with a parameter optimisation process can identify change intervals on a streaming dataset with 100,000 points in a few minutes. Furthermore, the online version takes only a few seconds on the same dataset.

The contributions of this paper are:

(1) Defining three main types of change points in data streams.
(2) Generalising the CPD problem as a special case of Change-Interval Detection (CID) problem. Although all existing works focus on CPD problem, we find it more efficient and effective to identify change intervals instead.
(3) Proposing IDK-based algorithm `iCID` for change interval detection in streaming datasets. We are the first to adopt the recent Isolation Distributional Kernel to measure the similarity between adjacent intervals.
(4) Verifying the effectiveness and efficiency of `iCID` on synthetic and real-world datasets. Our empirical results show that `iCID` performs better than 7 state-of-the-art methods, including deep learning methods in terms of F1-score and/or runtime.

The rest of the paper is organised as follows. We first describe the related work of CPD in Section 2 and then define the problem for change-interval detection in Section 3. Section 4 introduces IDK for measuring interval similarity and the proposed `iCID` algorithm. The experimental settings and results are presented in Section 5 and Section 6, respectively. Finally, we provide a discussion in Section 7, followed by a conclusion in Section 8.

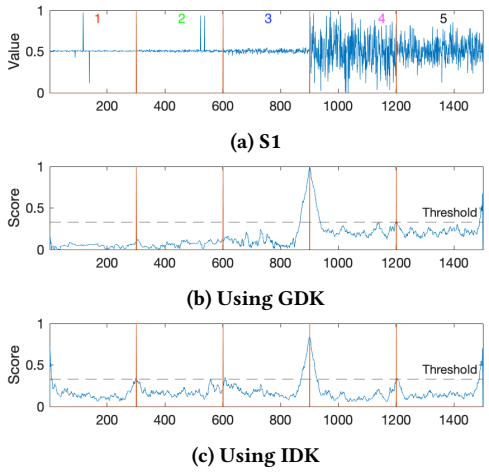
2 RELATED WORK

We summarise the methods for change-point detection in Table 1 and classified them into the following three subsections.

2.1 Point-Regression Based CPD

Point-regression based methods detect the change points by computing the gap between the true values and the predicted values of time series data. Parametric and non-parametric model can be used for time series data regression. *Autoregressive Moving Average* (ARMA) [3] is a parametric model that combines *Autoregressive* (AR) and *Moving Average* (MA) for forecasting time series, where the AR describes the relationship between current and historical values. The MA generates a linear model of the error accumulation of the autoregressive part. Parametric models can greatly simplify the process of model learning because they specify the form of the objective function. However, they usually do not exactly match the underlying objective function, which may lead to underfitting and underperformance.

The non-parametric methods can fit different functional forms without assumption about the probability distribution. *Autoregressive Gaussian Process* (ARGP) [4] as a non-parametric model can be considered as a nonlinear version of AR based on Gaussian process for time series prediction. *ARGP-BOCPD* [24] uses ARGP

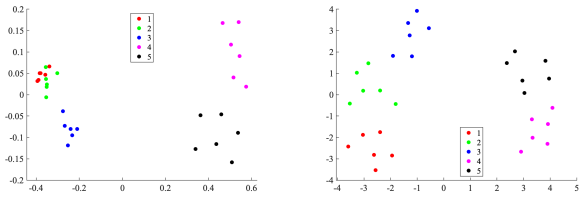


(a) S1

(b) Using GDK

(c) Using IDK

Figure 1: S1 dataset: Comparison of the same proposed distributional kernel-based CID algorithm using GDK vs IDK. (a) shows the data distribution. (b) and (c) plot the change point score with different kernels. Red bars indicate ground-truth change points. The variances of five blocks are 1.0, 2.2, 4.3, 48.3 **&** 28.3**, respectively. There are 5 manually add noise points in the first two blocks.**



(a) Using GDK (b) Using IDK

Figure 2: MDS result based on the dissimilarity matrix of intervals of S1 dataset. The split intervals are shown as points.

in underlying predictive models of *Bayesian online change point detection* (BOCPD) framework to extend BOCPD. There are other methods extracting the shape patterns of time series. *FLOSS* [12] identifies the position of change points by comparing changes in interval shape pattern. *ESPRESSO* [9] combines shape pattern (based on FLOSS) and statistic approach (information-gain based [25]) to identify change points to detect potential segment boundaries.

However, since noise points are usually far away from the predicted value, point-regression based methods would inevitably misidentify them as change points when they set a threshold on the gap between the test value and predicted value. Moreover, the shape-based [9, 12] approaches usually have a high time complexity for subsequence matching.

2.2 Distribution-Based CPD

Distribution-based CPD methods detect the change point via computing the gap between two distributions. They have been shown promising performance in CPD tasks [4, 11, 16, 17, 24, 34]. These methods are usually non-parametric based kernel methods, so that

Table 1: CPD Methods Comparison

Method	Parametric	Time Complexity	Feature Learning	Type	Core Method	Distribution focus
ARMA [3]	YES	$O(n)$	No	Point-regression	Moving Average	No
ARGP [4]	No	$O(n^3)$	No	Point-regression	Kernel-based	Yes
ARGP-BOCPD [24]	No	Polynomial	No	Point-regression	Kernel-based	Yes
FLOSS [12]	No	$O(nlog(n))$	No	Point-regression	Shape-based	No
ESPRESSO [9]	No	$O(n^2)$	No	Point-regression	Shape-based	No
aHSIC [34]	No	$O(n^2)$	No	Distribution	Kernel-based	Yes
KCD [11]	No	$O(n)$	No	Distribution	Kernel-based	Yes
RuLSIF [17]	No	Unstated	No	Distribution	Kernel-based	Yes
Mstat [16]	No	$O(n)$	No	Distribution	Kernel-based	Yes
RNN En/Decoder [7]	No	Polynomial	YES	Supervised	Deep Learning	No
LSTNet [15]	No	Polynomial	YES	Supervised	Deep Learning	No
KL-CPD [6]	No	Polynomial	YES	Distribution	Deep Kernel-based	Yes
TS-CP^2 [10]	No	Polynomial	YES	Self-supervised	Deep Learning	No
iCID	**No**	**O(n)**	**No**	**Distribution**	**Kernel-based**	**Yes**

thay are more robust to provide consistent results on different types of data distributions. *aHSIC* [34] uses Gaussian kernel to calculate the dissimilarity of past and future interval data separability to detect change points. The method *M-stat* proposed in [16] uses *M-statistics* to detect change points based on Gaussian kernel *MMD* for two-sample test. *RuLSIF* [17] uses a density-ratio estimation for CPD and uses Gaussian kernel to model the density-ratio distributions between interval of time series. Whereas, *KCD* [11] uses *One Class Support Vector Machine* (OCSVM) to measure the dissimilarity between past and future sets.

Existing kernel-based CPD methods usually use Gaussian kernel as a means to estimate distributions and examine the distributional difference between different intervals. However, using Gaussian kernel have difficulty capturing subtle change points. We found that *M-stat* is failed to detect the first two subtle change points on S1 dataset neither.

2.3 Deep Learning-Based CPD

Deep learning based CPD methods have been developed in recent years. *RNN Encoder-Decoder* [7] uses two *Recurrent Neural Networks* (RNN) as encoder and decoder respectively to learn the nonlinear features of the sequence and forecast time series. *LSTNet* [15] is a deep learning framework designed for multivariate time series forecasting tasks, which combines *RNN* and *CNN* to forecast future time series by extracting short-term local dependency patterns between variables. *KL-CPD* [6] uses kernel methods and two-sample tests to measure differences between continuous intervals to detect change points. Its definition assumes i.i.d., yet the time dependency is incorporated in the generative modelling using *RNN* (to model the changed distribution which is close to, but different from, the distribution before change. The modelled distribution is a surrogate distribution in order to generate points to simulate points from changed distribution where points are few (or unavailable) in practice. *TS-CP*^{2} [10] uses a single positive pair of contiguous time windows and a set of negative separated across time windows to train the encoder with contrastive learning. *TS-CP*^{2} also used *Wavenet* to learn representation between time intervals and hypothesised that a change point will be presented if there is significant representation difference between time adjacent intervals.

However, Deep-learning-based methods are inefficient to handle massive data streams because they usually require a large amount of data for expensive training.

The closest related works to ours are [6, 16], which estimate the distributions (with i.i.d. assumption) of intervals to conduct a two-sample test with *MMD*. They employ blocks/segments as sets of i.i.d. points, measured in terms of some similarity measure; though the actual measure is different (and with linear time complexity). The key differences between iCID and [6, 16] are the kernel and the distributional measure employed. The use of Gaussian Distributional Kernel has quadratic time complexity, which requires an approximation in order to reduce to linear time complexity. However, since IDK has a finite-dimensional feature map, iCID measures the similarity of two distributions between two intervals in linear time, without any approximation. Furthermore, the data-dependent property of IDK makes iCID easier to detect subtle change points.

3 PROBLEM DEFINITION

3.1 Change-Point Detection

Existing works [6] usually define a change point as a time step when the data distribution before the change point is different from the data distribution after the change point, i.e., the data distribution has a abrupt change after the change point.

Let $\mathcal{P}_L^t$ and $\mathcal{P}_R^t$ be the distributions before and after a point x_t, respectively, in a sequence $x_i \in \mathbb{R}^d,\ i \geq 1$, where the distributions are estimated from w observations $x_i, i \in [t-w, t)$ and $i \in (t, t+w]$ which are assumed to be their i.i.d. samples.

Definition 1. *Given a sequence of observations* $\{x_1, x_2, x_3, ...\}$, $\forall_{t>w}$, x_t *is a*

- *change point if* $\mathcal{P}_L^t \neq \mathcal{P}_R^t$ *and either* $x_t \sim \mathcal{P}_L^t$ *or* $x_t \sim \mathcal{P}_R^t$;
- *same-distribution point if* $\mathcal{P}_L^t = \mathcal{P}_R^t$ *and* $x_t \sim \mathcal{P}_L^t$;

- *noise point if* $\mathcal{P}_L^t = \mathcal{P}_R^t$ *and* $x_t \nsim \mathcal{P}_L^t$.

The scale of the difference in distributions can further create two subcategories of change points: x_t is a subtle-change point if the difference between $\mathcal{P}_L^t$ and $\mathcal{P}_R^t$ is small; otherwise, x_t is an obvious-change point.

There are mainly three types of change points defined as follows:

(I) Sudden change: the distribution is changed suddenly after the change point, i.e., there is only one change point in the sequence. Many existing CPD methods [6, 11, 16, 34] focus on this type of change by identifying the change point defined in Definition 1.

(II) Gradual change: a distribution is gradually changed to another distribution over a short period in which points generated from two different distributions co-exist, i.e., $x \sim (\mathcal{P}_L \cup \mathcal{P}_R)$ within the period, and the distributions before and after the period are $\mathcal{P}_L$ and $\mathcal{P}_R$, respectively.

(III) Continuous change: a distribution keeps changing, potentially, over a period, i.e., for any point x_t within the period, $\mathcal{P}_L^t \neq \mathcal{P}_R^t$.

In real-world applications, identifying every single change point in a sequence is impractical and unnecessary due to three main reasons:

(1) It is time-consuming for any method to check every point in a sequence, especially the sequence can be infinite;

(2) There is no single 'ground-truth' change point in a short sequence, e.g., a gradual change of distribution and an continuous change of distribution.

(3) Many change points occur in a contiguous sequence, as in the two last types of change points.

3.2 Change-Interval Detection

Here we argue that the CPD problem can be better treated when it is defined as a *change-interval detection* (CID) problem, where the detection focus is the interval that contains at least one change points. In addition, the CPD problem is a special case of CID problem. Although all existing works focus on CPD problem, we find that it is more efficient and effective to identify change intervals instead.

Definition 2. *A* ***change interval*** *contains one or more change points.*

Thus, the distribution in the change interval is different from that in the previous interval. Given a distributional measure $f(.)$ to calculate the similarity between two distributions, a f_τ-change interval is formally defined as follows.

Definition 3. *Given two adjacent subsequences L and R, each having length w, R is a* f_τ*-change interval if the similarity between the two distributions* $\mathcal{P}_L$ *and* $\mathcal{P}_R$ *is less than a threshold* τ*:*

$$f(\mathcal{P}_L, \mathcal{P}_R) \leq \tau. \tag{1}$$

Different distributional measures have been developed to compute the similarity between two intervals, e.g., *KL (Kullback–Leibler) divergence* [8], *JS (Jensen-Shannon) divergence* [19], *Wasserstein metric* [31], and *MMD* [2].

The CID method we proposed in this paper is a generic method for CPD. This CID method can be converted to a CPD method using a sliding window approach. The details are provided in Section 7.1. Nevertheless, both methods overcome the three unresolved challenges.

Regarding interval splitting for CID methods in practice, a smaller length of the interval is preferable to detect the change interval sooner than using a larger length of interval.

4 CHANGE-INTERVAL DETECTION BASED ON DISTRIBUTIONAL KERNEL

We provide the pertinent details of distributional kernels, used for measuring similarities between distributions, in the first subsection. Then, we propose a new algorithm for change-interval detection called `iCID`, based on the distributional kernel, in the second subsection.

4.1 Distributional Kernels

Given two probability distributions $\mathcal{P}_X$ and $\mathcal{P}_Y$, the similarity between them can be measured using a distributional kernel.

Definition 4. *A* ***distributional kernel****, based on Kernel Mean Embedding (KME) [20], is given as:*

$$\begin{aligned} \mathcal{K}(\mathcal{P}_X, \mathcal{P}_Y) &= \frac{1}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} \kappa(x, y) \\ &= \left\langle \widehat{\Phi}(\mathcal{P}_X), \widehat{\Phi}(\mathcal{P}_Y) \right\rangle, \end{aligned} \tag{2}$$

where $\widehat{\Phi}(\mathcal{P}_X)$ is the kernel mean map of $\mathcal{K}$:

$$\widehat{\Phi}(\mathcal{P}_X) = \frac{1}{|X|} \sum_{x \in X} \phi(x), \tag{3}$$

and ϕ is the feature map of point kernel $\kappa(x, y) = \langle \Phi(x), \phi(y) \rangle$.

KME often uses Gaussian kernel as κ; and we call it Gaussian Distributional kernel (GDK) in this paper.

Recently, a new *Isolation Distributional Kernel* (IDK) has been proposed to perform point & group anomaly detection [**?**] and time series anomaly detection [26].

IDK has two distinctive characteristics in comparison with GDK. First, IDK has a finite-dimensional feature map that enables efficient computation of distributional similarity with linear time complexity. In contrast, GDK has a feature map with infinite-dimensionality.

Second, IDK is a data-dependent measure, i,e., two distributions, as measured by IDK derived in a sparse region, are more similar than the same two distributions, as measured by IDK derived in a dense region. GDK is a data independent kernel.

An example of the data-dependent property is shown in Figure 1: the dissimilarity between blocks 1 and 2 is enlarged; but the dissimilarity between blocks 4 and 5 is reduced. Therefore, the dissimilarity scores between different adjacent blocks reach to similar level such that subtle change points in the original data stream become globally obvious change points that are easier to be detected using a single threshold, as shown in Figure 1c.

In this paper, we focus on using IDK as the distributional measure f to compute the similarity between two adjacent intervals. To be consistent with other existing measures of difference or discrepancy, we use $\mathfrak{S}(X, Y) = 1 - f(\mathcal{P}_X, \mathcal{P}_Y)$ as the dissimilarity score between

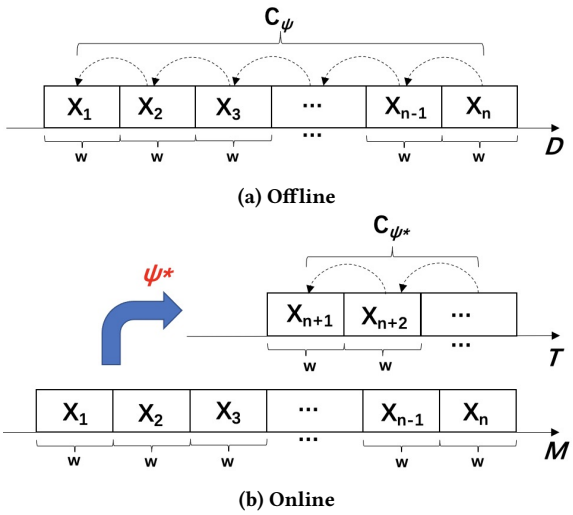


Figure 3: Illustration of (a) offline: the best ψ^* value is determined by the whole data stream; (b) online: the best ψ^* value is determined by the reference data M. w is the window size.

two intervals X and Y hereafter. The details of IDK are provided in Appendix A.

4.2 The Proposed iCID

In this subsection, we propose the Isolation Distributional Kernel-based Change Interval Detection (iCID).

The key procedures of iCID are as follows. It first splits the dataset D into N non-overlapping time intervals, i.e., $X_1, X_2, ...X_N$, where each X_i has the same window size w. Then it uses IDK to map each time interval X_i into a point in the Isolation kernel space using KME, and computes the similarity between paired adjacent intervals. The iCID score of interval X_i is:

$$\texttt{iCID}(X_i) = \mathfrak{S}(X_i, X_{i-1}) = 1 - \widehat{\mathcal{K}}_I(\mathcal{P}_{X_i}, \mathcal{P}_{X_{i-1}}). \tag{4}$$

Algorithm 1 illustrates the detailed steps of iCID.

Algorithm 1 iCID

Input: Dataset D; window size w; subsample size ψ for IDK
Output Scores iCID, $i = 1, \ldots, N$

1: Split D into N non-overlapping time intervals, each having w points, i.e.,
$D \rightarrow \{X_i, i = 1, \ldots, N\}$, where $N = \lfloor length(D)/w \rfloor$
2: $\widehat{\Phi} \leftarrow$ Create *IDKFeatureMap* (D, ψ)
3: **for** $i = 1, ..., N$ **do**
4: $\quad \texttt{iCID}(X_i) = \mathfrak{S}(X_i, X_{i-1})$
5: **end for**

Based on the availability of the unlabelled data, we propose offline and online versions of iCID, as illustrated in Figure 3. In both versions, we propose an optimisation method for ψ parameter settings for deploying iCID in real practice.

4.2.1 Offline version. We search the best value of ψ based on approximate entropy, which is a measure to quantify the amount of regularity and unpredictability of fluctuations over a time series [22]. A low value of the approximate entropy indicates that the time series has higher regularity while a high value indicates high randomness.

Change points are, by Definition 1, rare in time series, therefore, let $C_\psi = \{\texttt{iCID}(X_1), \texttt{iCID}(X_2), ..., \texttt{iCID}(X_N) \mid \psi, D\}$, the best ψ leads to a minimum approximated entropy, i.e.,

$$\psi^* = \underset{\psi}{\operatorname{argmin}}\, \bar{E}(C_\psi), \tag{5}$$

where $\bar{E}(.)$ is the approximated entropy estimation [22].

Moreover, a threshold τ can be identified as

$$\tau(C_{\psi^*}) = \mu + \alpha \times \sigma, \tag{6}$$

where μ and σ are the mean and standard deviation of C_{ψ^*}, respectively; and α is a power factor parameter.

4.2.2 Online version. Supposing that we have a reference dataset M, we can estimate the best ψ^* from M, given a window size w. For the incoming new dataset T, we can still split the data into intervals of the same length w and calculate the CID score of each interval based on Equation 5 with ψ^* and τ.

In both online and offline fashions, an interval with a CID score higher than τ is identified as a change interval.

5 EXPERIMENTAL SETTINGS

In this section, we introduce the evaluation datasets and parameter settings in our experiments.

5.1 Benchmark Datasets

In order to demonstrate the effectiveness of the proposed iCID method in a variety of applications, we include 6 real-world and 2 synthetic datasets. Table 2 presents the properties of each dataset. All datasets are normalised such that each dimension in the same range of [0, 1], before experiments.

5.1.1 Real-world Datasets.

- **Yahoo-16**[2] records hardware resource usage during the operation of the PNUTS/Sherpa database (e.g. CPU utilization, memory utilization, disk utilization, network traffic, etc). We select the 16^{th} of total 68 representative time series data after removing some sequences with duplicate patterns in anomalies [6]. Yahoo-16 dataset has 5 type I change points.
- **USC-HAD**[1] [35] includes 14 subjects and 12 daily activities. Each subject was fitted with a 3-axial accelerometer and 3-axial gyroscope, which were fixed in front of the right hip joint and sampled at 100 Hz. We randomly chose 30 activities from the first six participants and stitched the selected recordings together in a random manner [10]. USC-HAD dataset has 4 type I obvious-change points mentioned in Section 3.1.

[2] https://webscope.sandbox.yahoo.com/catalog.php?datatype=s
[1] http://sipi.usc.edu/had

Table 2: The properties of the datasets used in our experiments.

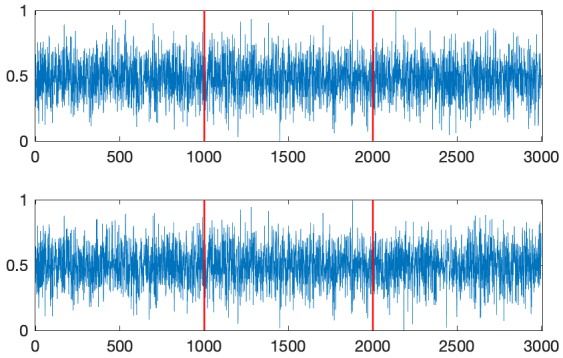
Dataset	Size	#dimensionality	#Changing interval
S1	15000	1	4
S2	3000	2	2
USC-HAD	93635	1	6
Yahoo-16	424	1	5
Google-trend	219	1	2
well_log	4050	1	10
Weather	3288	1	2
HASC	39397	3	65

- **Google-Trend**[3] is a monthly time series data of the Google Search popularity of the word 'beach' over the US. This time series is regularly-spaced and the frequency period is 1 year. Google-trend dataset has 2 type I change points.
- **Well_Log**[4] [21] contains measurements of the nuclear magnetic response and conveying information about the structure of the rock. The series has been sampled every 6 iterations to reduce the length [30]. Well_log dataset has 10 type I change points. In addition, it contains some noise points.
- **Weather**[5] is global monthly mean temperature time series data in degrees Celsius from 1880 to 2022. Note that this data does not have labels for change points, but we can easily observe the change trends from the value distribution along the timeline, i.e., the period of 1,400-2,000 and 2,400-3,288 has type II and III change points, respectively.
- **HASC**[6] [13, 14] is human activity data with 3 dimentions provided by HASC challenge 2011. We used a same subset of the HASC dataset from *KL-CPD* [6]. HASC dataset has both type I sudden-change and subtle-change points.

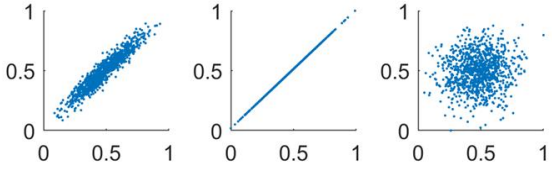
5.1.2 Synthetic Datasets. We generated two synthetic datasets to simulate different types of change points in streaming data. Note that each data is piece-wise i.i.d. and has no time dependency within each segment or between segments. Both datasets has type I obvious-change and sudden-change points.

- **S1:** There are 5 one-dimensional Gaussians of 300 points with the same $\mu = 0$ but different $\sigma = 1, 2.2, 4.3, 48.3, 28.3$, respectively. In addition, there are 5 noise points located at 89, 117, 139, 523 and 537, respectively. The data distribution is shown in Figure 1a.
- **S2:** There are three two-dimensional Gaussians of 1000 points with the same $\mu = 0$ but different covariances Cov=[0.9 0.4;0.4 0.2], [0.5 0.5;0.5 0.5], [0.9 0.1;0.1 0.9], respectively. The data distribution is shown in Figure 4. Note that the data distribution on each attribute is the same over the whole period, while there are two type I change points in the data when considering the two-dimensional covariance.

[3]https://github.com/zhaokg/Rbeast/blob/master/Matlab/testdata/googletrend.mat
[4]https://github.com/alan-turing-institute/TCPD/tree/master/datasets/well_log
[5]https://datahub.io/core/global-temp#readme
[6]http://hasc.jp/hc2011/



(a) 1-dimensional plot



(b) 2-dimensional plot

Figure 4: Data distribution of S2. (a) shows the data distribution on each attribute. (b) shows the data distribution using two attributes for each of the three intervals, split by the two change points indicated as red bars in (a).

Table 3: Parameters search ranges for iCID

Parameter	Search ranges
Subsampe size	$\psi \in \{2, 4, 8, 16, 32, 64\}$
Window size	$w \in \lceil 10, 15, ..., 400 \rceil$
Power factor	$\alpha \in \lceil 0, 0.1, 0.2, ..., 3 \rceil$

5.2 Parameter Setting

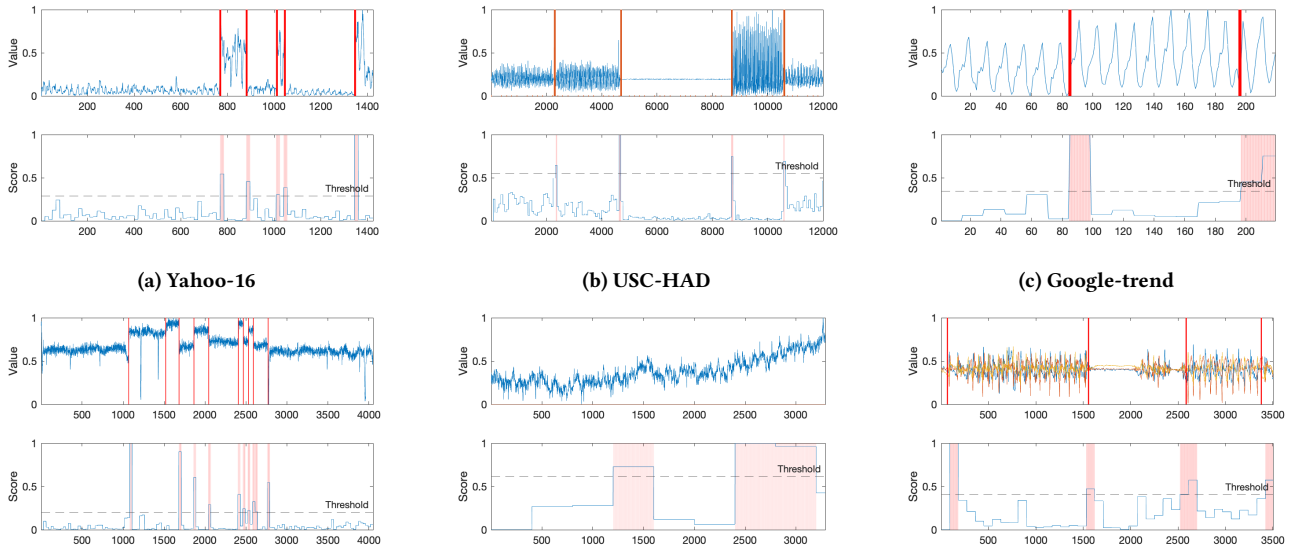
Both online and offline versions of `iCID` have two manual parameters, i.e., window size w and α. The window size is the length of intervals [10].

The parameter search ranges are provided in Table 3. For the online `iCID`, each dataset is split into two equal subsets, the reference set contains the first 50% of data samples for parameter ψ optimisation. Then it evaluates the remaining test 50% of data samples based on the best ψ^*.

6 EMPIRICAL EVALUATION

In this section, we aim to answer the following questions:

- Is `iCID` an effective method to identify three types of change points in univariate and multivariate data streams?
- Is `iCID` resistant to noise points and scalable to large data size?
- Does `iCID` have any advantage over other CPD algorithms, in terms of detection performance and time complexity?



(a) Yahoo-16 (b) USC-HAD (c) Google-trend

(d) well_log (e) Weather (f) HASC

Figure 5: Offline iCID results on real-world datasets. the first row is the data distribution with change point location presented by red bars. The second row shows the change score and the detected change intervals as red areas. For USC-HAD and HASC datasets, we show the results on a subsequence of 12,000 and 3,500 samples, respectively.

To answer those questions, we present the empirical evaluation results of our proposed iCID algorithm. It first visualises the performance of online and offline versions of iCID on real-world and synthetic datasets. Then it reports the quantitative evaluation results on the two largest real-world datasets, compared with other 7 state-of-the-art CPD algorithms.

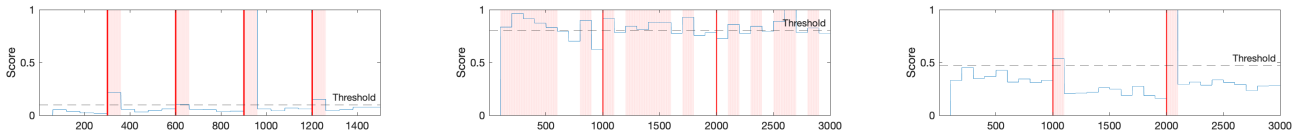
6.1 Offline iCID

The performance of offline iCID on 6 real-world datasets and 2 synthetic datasets are visualised in Figures 5 and 6, respectively. Since there is no ground truth about change intervals, if a CID method detects a change interval containing ground truth change points, then it correctly identifies a change interval.

Observations on real-world datasets are given below:

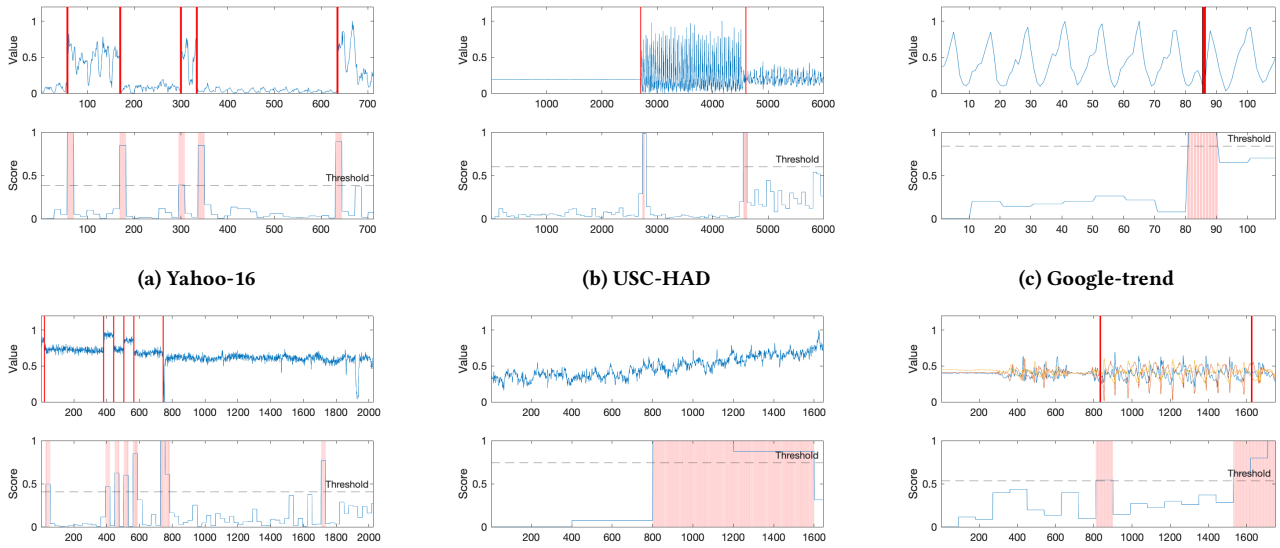
- Yahoo-16 shown in Figure 5a contains several change points. Our proposed algorithm iCID detected all intervals containing change points with a single threshold automatically.
- Figure 5b shows that iCID can detect four type I change points on USC-HAD datasets easily, although the first change point seems subtle.
- iCID detected all subtle-change and sudden-change points on S1 dataset. There are some noise points in S1 but they do not affect the detection of changing distribution for iCID, as shown in Figure 6a.
- Google-trend dataset is a periodic dataset. iCID is able to detect the change intervals which contain the change points and the window size is 14 which matches the length of a single period.
- As shown in Figure 5d, the well_log dataset contains a few noise points which could easily be mistaken as change points. iCID successfully found 9 out of 10 change points without misclassifying any noise point. It is worth mentioning that all change points can be correctly identified if manually set a large parameter ψ value.
- The Weather dataset differs from other datasets because it has both gradual (type II) and continuous (type III) change points, as shown in Figure 5e. iCID discovered that there were two types of change points in historical average temperatures, i.e., a short fluctuation around 1,500 and a continuously increasing trend after 2,500.
- HASC is the human activity data which has three dimensions and iCID detected all the change points in this subsequence, as shown in Figure 5f. It is worth mentioning that although the HASC dataset has 3 dimensions, they are highly correlated, i.e., the change points can be detected based on any one of the three dimensions using iCID.
- When examining the S2 dataset, there is no distribution change on any individual attribute. The change points can only be detected by considering both attributes, as shown in Figure 4b. The results presented in Figure 6c show that iCID has the capability to detect the two change-intervals containing the change points on this multivariate dataset.

Overall, the offline iCID can detect all three types of change points and tolerate noise points with effective parameter optimisation. It detects all change intervals which contain the change points on all datasets, except one change-point on the well_log dataset, with automatic ψ setting. It also has the capability to detect change points in both univariate and multivariate streaming datasets.



(a) S1 dataset (b) S2 dataset (only using the 1st feature) (c) S2 dataset

Figure 6: Offline iCID results on Synthetic datasets. The score distribution using the 2nd feature of S2 dataset is similar to (b).



(a) Yahoo-16 (b) USC-HAD (c) Google-trend

(d) Well_log (e) Weather (f) HASC

Figure 7: Online CPD results on real-world datasets. In each subfigure, the first row is the data distribution with change point location presented by red bars. The second row shows the change score and the detected change interval as red areas.

6.2 Online iCID

The performance of online iCID evaluated on both real-world datasets and synthetic datasets are shown in Figure 7 and 8, respectively.

We observed similar results to the offline version as follows:

- The online iCID finds all change points on all datasets, except one false change point on Well_log dataset. This error can be easily rectified if using a better ψ setting.
- Although some test datasets contain different types of change points from the reference datasets, the online iCID still can detect those change points well based on the ψ optimised from the reference datasets. In other words, ψ value is not sensitive for iCID, and this property makes online iCID robust to different types of change points. We provide the sensitivity test in Section 7.3 that conforms to this claim.

6.3 Comparison With State-of-the-Art Methods

In this subsection, we compare the performance of iCID against 7 state-of-the-art methods , including *ESPRESSO* [9], *FLOSS* [12], *aHSIC* [34], *RuLSIF* [17], *KL-CPD* [6] and *TS-SP*^{2} [10], on the two largest real-world datasets, i.e., USC-HAD and HASC.

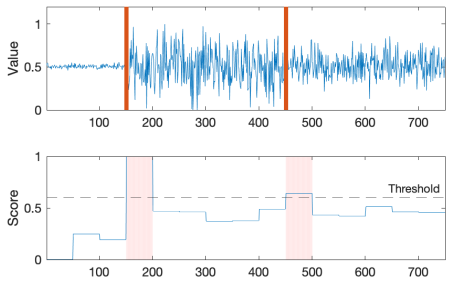
For a fair comparison, we followed the same evaluation strategy used in the paper [10] and calculated the highest F1-score of iCID with manually tuned parameters and detection margins. Only if a change point/interval detected is located within the specified margin of the ground truth change point, then it is a true positive. Therefore, the evaluation is still based on intervals rather than points.

Table 4 presents the performance comparison of 8 algorithms. We have the following observations:

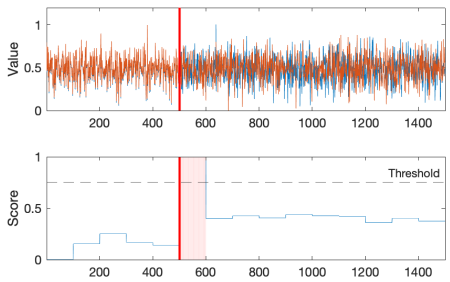
- Compared with the other 7 algorithms, iCID has the highest F1 score on HASC dataset when the detection margin is 100 and 200. The best window sizes are only around half of the corresponding detection margins.
- Both *FLOSS* and *ESPRESSO* are shape-based CPD methods, their performance is usually far below than that of iCID on both two datasets under different margin values. The reason could be the fact that shape-based CPD methods are very intolerance to noise points.
- Comparing the three Gaussian-based kernel methods, including *Mstat*, *aHSIC* and *RuLSIF*, iCID outperforms them on both datasets with the detection margin greater than or equal to 100.

Table 4: Performance Comparison with state-of-the-art methods on the two largest real-world datasets. The highest F1-score on each dataset with the same detection margin is boldfaced. The results of existing methods are from the *TS-SP*^{2} paper [10]. Their source codes are either unavailable publicly or unexecutable.

Dataset	Methods	Best Window	F1-score	Best Window	F1-score	Best Window	F1-score
	Detection Margin	60		100		200	
HASC	*FLOSS*	60	0.3088	60	0.3913	100	0.5430
	aHSIC	40	0.2308	40	0.3134	40	0.4167
	RuLSIF	200	0.3433	200	0.4999	200	0.4999
	ESPRESSO	100	0.2879	60	0.4233	100	0.6933
	Mstat	35	0.2958	35	0.4246	15	0.6372
	KL-CPD	60	**0.4785**	100	0.4726	200	0.4669
	TS-SP^2	60	0.40	100	0.4375	200	0.6316
	`gCID(MMD)`	65	0.3522	55	0.5493	90	0.7813
	`iCID(MMD)`	40	0.3117	50	0.4734	100	0.7612
	`iCID`	65	0.3333	85	**0.5630**	120	**0.7943**
	Detection Margin	100		200		400	
USC-HAD	*FLOSS*	100	0.2666	100	0.3666	400	0.4333
	aHSIC	50	0.3333	50	0.3333	50	0.3999
	RuLSIF	400	0.4666	400	0.4666	400	0.5333
	ESPRESSO	100	0.6333	100	0.8333	100	0.8333
	Mstat	35	0.5185	30	0.5352	30	0.6774
	KL-CPD	100	0.7426	200	0.7180	400	0.6321
	TS-SP^2	100	**0.8235**	200	**0.8571**	400	0.8333
	`gCID(MMD)`	60	0.6786	65	0.7941	255	**0.8451**
	`iCID(MMD)`	70	0.7273	65	0.7632	255	0.8421
	`iCID`	70	0.6857	70	0.7536	255	0.8378



(a) S1



(b) S2

Figure 8: Online `iCID` results on the Synthetic datasets.

- With a large detection margin, `iCID` performs significantly better than *TS-SP*^{2} on HASC, but is comparable to *TS-SP*^{2} on USC-HAD dataset. However, *TS-SP*^{2} is a self-supervised deep learning-based method that needs heavy training.
- The F1 scores of the models suppose to increase as the detection margin increases. However, the F1 scores of deep learning-based methods, i.e., *KL-CPD* and *TS-SP*^{2}, decrease as the detection margin increases. The reason could be the instability of deep learning methods.

We also investigate two variants of `iCID`, i.e., `iCID(MMD)` and `gCID(MMD)`, both use *MMD* [2]. The latter uses Gaussian kernel in MMD, as used in the two-sample test algorithms *Mstat* and *KL-CPD*; and the former uses Isolation Kernel as used in `iCID`.

It is interesting to mention that `iCID` and its two variants show similar performance on our evaluation datasets, as shown in Table 4. Moreover, `gCID(MMD)` shows much better performance than *Mstat* and *KL-CPD* on the two large real-world datasets in Table 4. The reason could be using a fix number reference blocks before the potential change point used in *Mstat*. Some of those blocks may be outdated. Since the parameter of *KL-CPD* is searched based on deep learning, it may not be optimised to detect different types of change points.

6.4 Scalability Evaluation

The runtime ratios of `iCID` and *TS-SP*^{2} against a different number of points are shown in Figure 9. The results show that both versions of `iCID` have linear time complexity. Note that the offline `iCID`

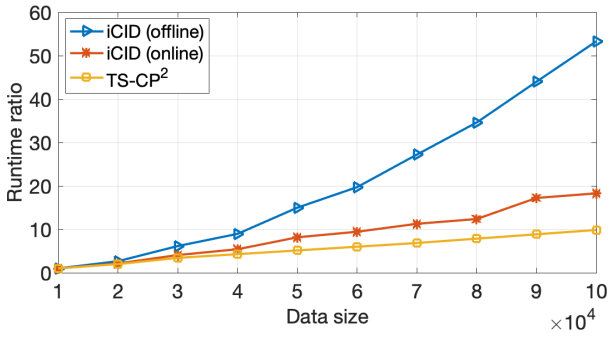


Figure 9: Scaleup test. iCID is tested using Matlab R2022b with Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz and *TS-SP*^{2} is tested using Python with NVIDIA Quadro RTX 5000 GPU.

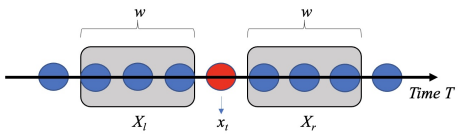


Figure 10: Overview of change-point detection based on IDK.

needs to run iCID multiple times in order to determine the best ψ value, but online iCID does not need to search the parameters. Thus, the runtime ratio of offline iCID is higher than that of online iCID.

It is interesting to point out that the exact CPU runtimes for the offline and online versions of iCID on a data stream with 100,000 points are 203.09s and 3.71s, respectively. However, we found that a deep learning-based method *TS-SP*^{2} takes about 170 GPU hours to train with the default parameter setting using a high-end GPU. Although *TS-SP*^{2} shows a linear runtime due to using a fixed number of epochs, it will cost much more time to determine the best parameters.

In summary, iCID is comparable to the latest deep-learning based method in terms of detection accuracy, but it has a much faster run time because of no learning or training.

7 DISCUSSION

In this section, we first present the change point detection task based on iCID, and then discuss the label issues found in the real-world dataset. Moreover, we conduct a sensitivity analysis of iCID with respect to different parameter settings.

7.1 Change-Point Detection with IDK

Using IDK for change-point detection is a special case of change-interval detection. We still can use IDK to measure the dissimilarity between the past interval $X_l = \{x_{t-w}, x_{t-w+1}, ..., x_{t-1}\}$ and the current interval $X_r = \{x_{t+1}, x_{t+2}, ..., x_{t+w}\}$ at each point x_t, as shown in Figure 10. The change point is the one with a large dissimilarity score, i.e., $\mathfrak{S}(X_l, X_r) \geq \tau$.

In order to detect all change points, we have to use a sliding window method to check every point from the streaming data.

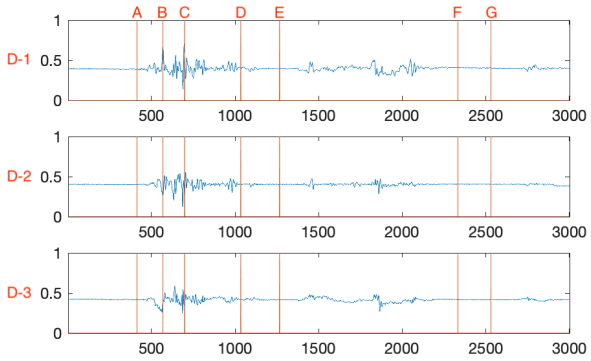


Figure 11: Demonstration of label issues on HASC dataset. D-1, D-2 and D-3 refer to different features.

Thus, it will result in more computational time than interval-based iCID. Since the scores of iCID is based on non-overlapping intervals only, they are a subset of scores using the sliding window method. Figure 1c shows the result of iCID using the sliding window method on S1 dataset.

7.2 About Ground-Truth Label Issues

Manually labelled real-world time series datasets usually contain mislabelled ground truth [33]. We visualised our evaluation real-world datasets and found that many of them have different labelling problems. To illustrate, Figure 11 presents a subsequence of the 3 dimension HASC dataset. Here we observe three types of issues as follows:

- **Wrong label position:** change points are labelled at incorrect or inaccurate location. For label D, the location of the label is far away from the exact change point.
- **No-labelling:** change points are not labelled. Around point 2000, the distribution of the data has some significant change but none of points are labelled as change.
- **Mislabelling:** normal points or noise points are mistaken for change points. For labels A, E, F and G, we cannot find any distribution change before and after these points in all three dimensions.

Therefore, due to the presence of error labels, Table 4 cannot accurately represent the best performance of different CPD algorithms. However, even with the existence of label errors in the datasets, our proposed algorithm iCID still achieves the state-of-the-art performance in F1 score.

7.3 Parameter Sensitivity Analysis

There are three parameters in iCID, i.e., the subsample size ψ used to build Isolation kernel, the interval/window size w and the power factor α used to set the threshold.

Generally, a larger ψ will make a sharper dissimilarity distribution, similar to a smaller bandwidth set in Gaussian kernel. For unsupervised learning, we have shown that ψ determined by Equations 5 can produce promising performance in last section. Figure 12 illustrates the effects of different ψ values on three datasets. It

can be seen from the results that a large ψ value can perform well on these datasets.

Figure 13 shows the sensitivity analysis of w on three datasets. The results indicate that the best w is varied over different datasets. Although the window size w should contain a sufficient amount of points to represent the data distribution, a smaller length is preferable to detect the change of distribution sooner.

The other parameter α in Equation 6 is a power factor parameter that denotes the level of sensitive for change interval identification. When we assume that the distribution of change scores conforms or is close to a Gaussian distribution, α can control the expected proportions of the sample as change intervals, e.g. any point which is larger than $\mu + 3\sigma$ will be considered as a change point when $\alpha = 3$. In practice, we can set $\alpha \in [1, 3]$ and a higher α will filter more subtle change points.

8 CONCLUSION

In this paper, we systematically analyse three key challenges of existing change-point detection methods and propose to design a change-interval detection method to address those challenges.

We verify that Isolation Distributional Kernel is capable of effectively measuring the dissimilarity between adjacent intervals in large data streams. Its data-dependent property enables the proposed `iCID` to detect both subtle change and obvious change points. `iCID` also presents promising performance with automated kernel parameter setting.

Our extensive evaluation confirms that `iCID` is effective in capturing thee types of change points in both univariate and multivariate streaming data. In addition, `iCID` has linear runtime with regard to the size of datasets, making them suitable for detecting change intervals in large data streams.

Therefore, `iCID` is a powerful streaming data mining tool for analysing massive streaming data, standing out for its data-dependent property and fast running speed. In the future, we will investigate the ability of `iCID` on detecting drifts in the non-i.i.d. streaming data.

ACKNOWLEDGEMENTS

This project is supported by National Natural Science Foundation of China (Grant No. 62076120).

APPENDIX

A ISOLATION DISTRIBUTIONAL KERNEL

The key idea of the Isolation kernel is using a space partitioning strategy to split the whole data space into ψ non-overlapping partitions based on a random sample of ψ points from a given dataset. The similarity between any two points is how likely these two points can be split into the same partition.

Let $D \subset \mathcal{X} \subseteq \mathbb{R}^d$ be a dataset sampled from an unknown probability distribution $\mathcal{P}_D$; Additionally, let $\mathbb{H}_\psi(D)$ denote the set of all partitionings H admissible from the given dataset $\mathcal{D} \subset D$, where each point $z \in \mathcal{D}$ has the equal probability of being selected from D. Each of the ψ isolating partitions $\theta[\mathbf{z}] \in H$ isolates one data point $\mathbf{z}$ from the rest of the points in a random subset $\mathcal{D}$, and $|\mathcal{D}| = \psi$. Each H denotes a partitioning from each $\mathcal{D}$ [23].

Definition 5. *For any two points $x, y \in \mathbb{R}$, Isolation Kernel of x and y is defined to be the expectation taken over the probability distribution on all partitionings $H \in \mathbb{H}_\psi(D)$ that both x and y fall into the same isolating partition $\theta[z] \in H$, $z \in \mathcal{D} \subset D$, where $\mathbb{1}(\cdot)$ be an indicator function:*

$$\kappa_I(x, y \mid D) \quad = \quad \mathbb{E}_{\mathbb{H}_\psi(D)}[\mathbb{1}(x, y \in \theta \mid \theta \in H)]. \tag{7}$$

In practise, κ_I is constructed using a finite number of partitionings H_i, $i = 1, \ldots, t$, (t=200 as default) where each H_i is generated by using randomly subsampled $\mathcal{D}_i \subset D$; θ is a shorthand for $\theta[z]$; ψ is the sharpness parameter:

$$\kappa_I(x, y \mid D) \quad = \quad \frac{1}{t}\sum_{i=1}^{t} \mathbb{1}(x, y \in \theta \mid \theta \in H_i). \tag{8}$$

Isolation kernel has a finite feature map, which is defined as follows [27, 28]:

Definition 6. ***Feature map of Isolation Kernel.*** *For point $x \in \mathbb{R}$, the feature mapping $\Phi : x \to \{0, 1\}^{t\times\psi}$ of κ_I is a vector that represents the partitions in all the partitioning $H_i \in \mathbb{H}_\psi(D)$, $i = 1, \ldots, t$; where x falls into either only one of the ψ hyperspheres in each partitioning H_i.*

Re-express Equation 8 using Φ is shown as follows:

$$\kappa_I(x, y \mid D) = \frac{1}{t}\left\langle \Phi(x|D), \Phi(y|D) \right\rangle. \tag{9}$$

Given the feature map Φ (defined in Definition 6) and Equation 9, the estimation of KME can be expressed based on the feature map of Isolation Kernel $\kappa_I(x, y)$.

Then IDK is defined as follows [28]:

Definition 7. *Isolation Distributional Kernel of two distributions $\mathcal{P}_X$ and $\mathcal{P}_Y$ is shown as follows:*

$$\mathcal{K}_I(\mathcal{P}_X, \mathcal{P}_Y \mid D) \quad = \quad \frac{1}{t}\left\langle \widehat{\Phi}(\mathcal{P}_X|D), \widehat{\Phi}(\mathcal{P}_Y|D) \right\rangle, \tag{10}$$

where $\widehat{\Phi}(\mathcal{P}_X|D) = \frac{1}{|X|}\sum_{x\in X} \Phi(x|D)$.

Note that we can normalise the IDK similarity to $[0, 1]$ as follows:

$$\widehat{\mathcal{K}}_I(\mathcal{P}_X, \mathcal{P}_Y \mid D) = \frac{\langle \widehat{\Phi}(\mathcal{P}_X|D), \widehat{\Phi}(\mathcal{P}_Y|D)\rangle}{\sqrt{\langle \widehat{\Phi}(\mathcal{P}_X|D), \widehat{\Phi}(\mathcal{P}_X|D)\rangle}\sqrt{\langle \widehat{\Phi}(\mathcal{P}_Y|D), \widehat{\Phi}(\mathcal{P}_Y|D)\rangle}}. \tag{11}$$

REFERENCES

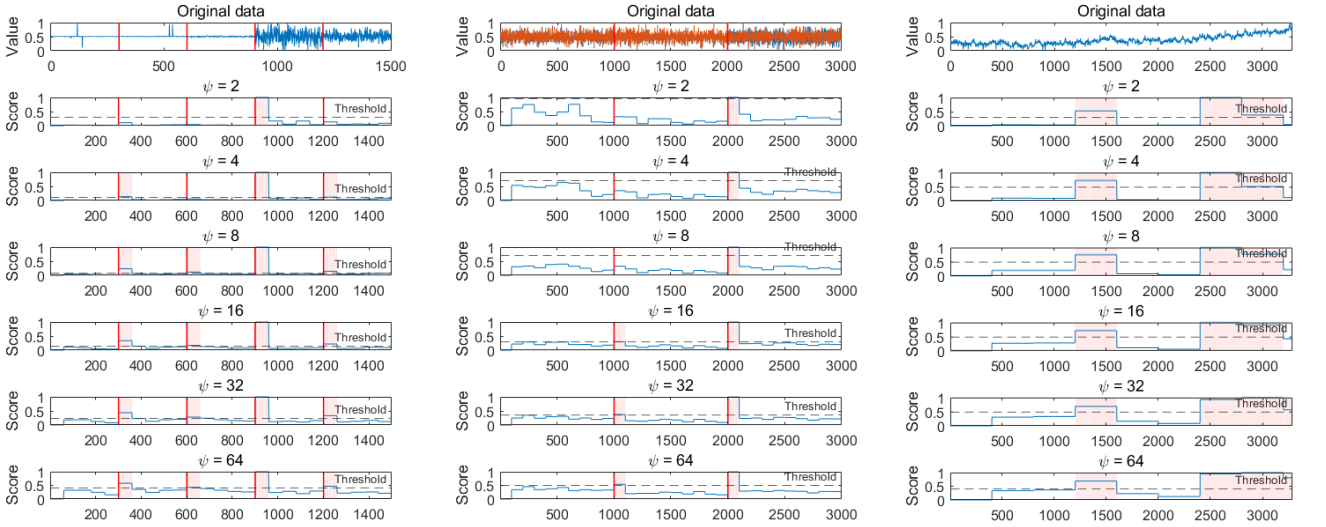
[1] Ingwer Borg, Patrick JF Groenen, and Patrick Mair. 2012. *Applied multidimensional scaling.* Springer Science & Business Media.

[2] Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. 2006. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics* 22, 14 (2006), 49–57.

[3] George Box. 2013. Box and Jenkins: time series analysis, forecasting and control. In *A Very British Affair.* Springer, 161–215.

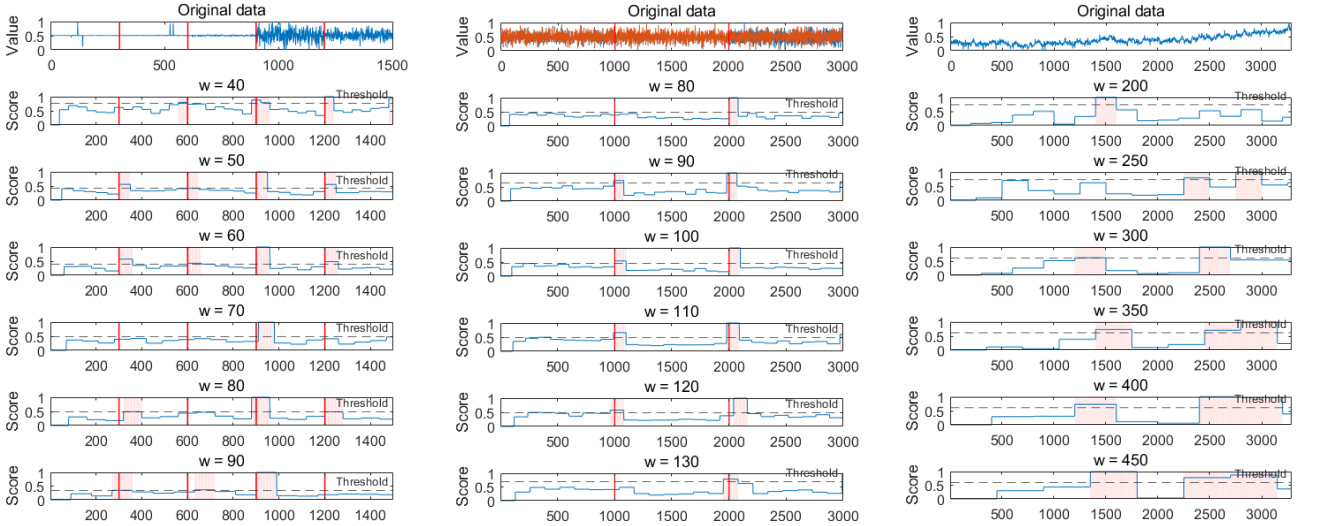
[4] Joaquin Quinonero Candela, Agathe Girard, Jan Larsen, and Carl Edward Rasmussen. 2003. Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, Vol. 2. IEEE, II–701.

[5] Faicel Chamroukhi, Samer Mohammed, Dorra Trabelsi, Latifa Oukhellou, and Yacine Amirat. 2013. Joint segmentation of multivariate time series with hidden process regression for human activity recognition. *Neurocomputing* 120 (2013), 633–644.



(a) S1 dataset with $w = 60$ **(b) S2 dataset with $w = 100$** **(c) Weather dataset with $w = 400$**

Figure 12: Sensitivity analysis of ψ on three datasets.



(a) S1 dataset with $\psi = 64$ **(b) S2 dataset with $\psi = 64$** **(c) Weather dataset with $\psi = 16$**

Figure 13: Sensitivity analysis of w on three datasets.

[6] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. 2019. Kernel Change-point Detection with Auxiliary Deep Generative Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1GbfhRqF7

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Imre Csiszár. 1975. I-divergence geometry of probability distributions and minimization problems. *The annals of probability* (1975), 146–158.

[9] Shohreh Deldari, Daniel V Smith, Amin Sadri, and Flora Salim. 2020. ESPRESSO: Entropy and ShaPe awaRe timE-Series SegmentatiOn for processing heterogeneous sensor data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 3 (2020), 1–24.

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

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

[12] Shaghayegh Gharghabi, Chin-Chia Michael Yeh, Yifei Ding, Wei Ding, Paul Hibbing, Samuel LaMunion, Andrew Kaplan, Scott E Crouter, and Eamonn Keogh. 2019. Domain agnostic online semantic segmentation for multi-dimensional time series. *Data mining and knowledge discovery* 33, 1 (2019), 96–130.

[13] Nobuo Kawaguchi, Nobuhiro Ogawa, Yohei Iwasaki, Katsuhiko Kaji, Tsutomu Terada, Kazuya Murao, Sozo Inoue, Yoshihiro Kawahara, Yasuyuki Sumi, and Nobuhiko Nishio. 2011. HASC Challenge: gathering large scale human activity corpus for the real-world activity understandings. In *Proceedings of the 2nd augmented human international conference*. 1–5.

[14] Nobuo Kawaguchi, Ying Yang, Tianhui Yang, Nobuhiro Ogawa, Yohei Iwasaki, Katsuhiko Kaji, Tsutomu Terada, Kazuya Murao, Sozo Inoue, Yoshihiro Kawahara, et al. 2011. HASC2011corpus: towards the common ground of human activity

recognition. In *Proceedings of the 13th international conference on Ubiquitous computing*. 571–572.

[15] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 95–104.

[16] Shuang Li, Yao Xie, Hanjun Dai, and Le Song. 2015. M-statistic for kernel change-point detection. *Advances in Neural Information Processing Systems* 28 (2015).

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

[18] Alexandre Lung-Yut-Fong, Céline Lévy-Leduc, and Olivier Cappé. 2012. Distributed detection/localization of change-points in high-dimensional network traffic data. *Statistics and Computing* 22, 2 (2012), 485–496.

[19] Christopher Manning and Hinrich Schutze. 1999. *Foundations of statistical natural language processing*. MIT press.

[20] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. 2017. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning* 10, 1-2 (2017), 1–141.

[21] J. J. K. Ó Ruanaidh and W. J. Fitzgerald. 1996. *Numerical Bayesian Methods Applied to Signal Processing*. Springer.

[22] Steven M Pincus. 1991. Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences* 88, 6 (1991), 2297–2301.

[23] Xiaoyu Qin, Kai Ming Ting, Ye Zhu, and Vincent Lee. 2019. Nearest-neighbour-induced isolation similarity and its impact on density-based clustering. In *Proceedings of the 33rd AAAI Conference on AI (AAAI 2019), AAAI Press*.

[24] Yunus Saatçi, Ryan D Turner, and Carl Edward Rasmussen. 2010. Gaussian process change point models. In *ICML*.

[25] Amin Sadri, Yongli Ren, and Flora D Salim. 2017. Information gain-based metric for recognizing transitions in human activities. *Pervasive and Mobile Computing* 38 (2017), 92–109.

[26] Kai Ming Ting, Zongyou Liu, Hang Zhang, and Ye Zhu. 2022. A new distributional treatment for time series and an anomaly detection investigation. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2321–2333.

[27] Kai Ming Ting, Jonathan R Wells, and Ye Zhu. 2022. Point-Set Kernel Clustering. *IEEE Transactions on Knowledge and Data Engineering* (2022). https://doi.org/10.1109/TKDE.2022.3144914

[28] Kai Ming Ting, Bi-Cun Xu, Takashi Washio, and Zhi-Hua Zhou. 2020. Isolation Distributional Kernel: A New Tool for Kernel based Anomaly Detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 198–206.

[29] Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing* 167 (2020), 107299.

[30] Gerrit JJ van den Burg and Christopher KI Williams. 2020. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222* (2020).

[31] Leonid Nisonovich Vaserstein. 1969. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii* 5, 3 (1969), 64–72.

[32] Christopher K. I. Williams and Matthias Seeger. 2001. Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp (Eds.). 682–688.

[33] Renjie Wu and Eamonn Keogh. 2021. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[34] Makoto Yamada, Akisato Kimura, Futoshi Naya, and Hiroshi Sawada. 2013. Change-point detection with feature selection in high-dimensional time-series data. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[35] Mi Zhang and Alexander A Sawchuk. 2012. USC-HAD: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*. 1036–1043.