Journal of Artificial Intelligence Research 79 (2024) 273-306 Submitted 03/2023; published 01/2024

Detecting Change Intervals with Isolation Distributional Kernel

Yang Cao CHARLES.CAO@IEEE.ORG
Ye Zhu YE.ZHU@IEEE.ORG
Deakin University, Geelong, Australia

Kai Ming Ting TINGKM@NJU.EDU.CN
Nanjing University, Nanjing, China

Flora D. Salim FLORA.SALIM@UNSW.EDU.AU
University of New South Wales, Kensington, Australia

Hong Xian Li HONG.LI@DEAKIN.EDU.AU
Luxing Yang Y.LUXING@DEAKIN.EDU.AU
Gang Li GANG.LI@DEAKIN.EDU.AU
Deakin University, Geelong, Australia

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 sensitivity to outliers. 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 outliers. 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.

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 (Basseville et al., 1993). *Change-Point Detection* (CPD) is an analytical method for identifying behavioural mutations in a temporal interval of observations (van den Burg & Williams, 2020). 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 (Truong et al., 2020), network traffic analysis (Lung-Yut-Fong et al., 2012) and human activity recognition (Chamroukhi et al., 2013).

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

Traditional point-regression-based methods, such as *Autoregressive Moving Average* (ARMA) (Box, 2013), *Autoregressive Gaussian Process* (ARGP) (Candela et al., 2003) and *ARGP-BOCPD* (Saatcci et al., 2010), usually cannot 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 outliers 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* (Gharghabi et al., 2019) and *ESPRESSO* (Deldari et al., 2020). However, they usually have a high time complexity and do not meet the challenge (a), thus, they are unable to deal with large data streams.

Recent deep learning-based methods (Cho et al., 2014; Lai et al., 2018; Deldari et al., 2021) also do not meet the challenge (a) due to training efficiency, although they are powerful enough to handle unstructured and multivariate data streams. They generally require a large amount of data with heavy training costs.

Moreover, distribution-based methods (Candela et al., 2003; Saatcci et al., 2010; Yamada et al., 2013; Liu et al., 2013; Li et al., 2015) usually have good scalability and tolerate outliers based on kernel methods, which are effective and efficient for solving difficult machine learning tasks. However, existing kernel-based CPD methods usually cannot meet the challenge (b), since they use Gaussian distributional kernel as a means to compute the similarity between distributions of intervals.

Using Gaussian distributional kernel has two key limitations:

1. The feature map of Gaussian kernel has intractable dimensionality. As a result, measuring the similarity between distributions with Gaussian distributional kernel has a high time cost or it can be sped up via an approximation such as the Nystrom method (Williams & Seeger, 2001) to produce a finite-dimensional feature map;

2. Gaussian distributional 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 Gaussian Distributional Kernel (GDK) and Isolation Distributional Kernel (IDK)[1]. The streaming data contains five different distributions in five blocks (indicated by five numbers on the top) and five outliers in the first two blocks.

When using Gaussian distributional kernel, shown in Figure 1(b), to measure the dissimilarity between adjacent intervals, the obvious change of distributions between blocks 3 & 4 shows a high score. However, the change between blocks 4 & 5 is very subtle; the changes between blocks 1 & 2 as well as blocks 2 & 3 cannot be detected.

1. Section 5.1.2 provides the properties of S1 dataset. The details of the proposed CPD methods are provided in Section 4. The evaluation results using other kernel methods are given in Appendix A.

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 distributional kernel. Figure 2 visualises the dissimilarity scores between adjacent intervals based GDK and IDK 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 separate using IDK than using GDK.

2. Robust to outliers. The change score is calculated based on distribution/interval rather than a single point. Figure 2 shows that all five manually added outliers have low changing scores regardless of using GDK or IDK.

3. Linear time complexity of both offline and online versions. 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.

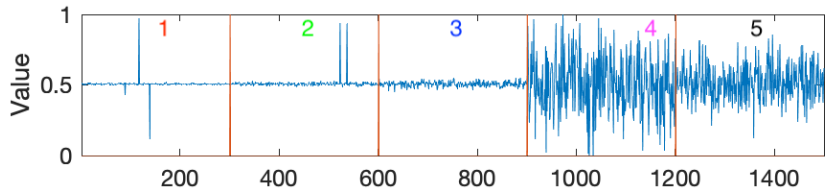
2. Generalising the CPD problem as a special case of Change-Interval Detection (CID) problem. Although all existing works focus on the CPD problem, we find it more efficient and effective to identify change intervals instead.

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. The source code of `iCID` can be obtained from `https://github.com/IsolationKernel/iCID`.

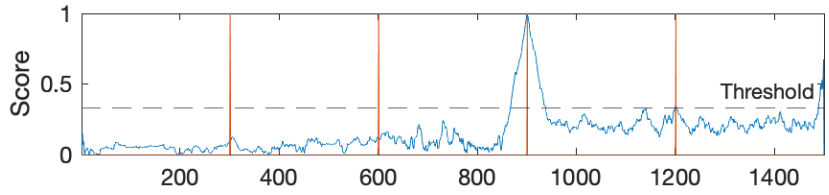
4. Verifying the effectiveness and efficiency of `iCID` on synthetic and real-world datasets. Our empirical results show that `iCID` performs better than 6 state-of-the-art methods, including deep learning methods in terms of F1-score and/or runtime.

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.

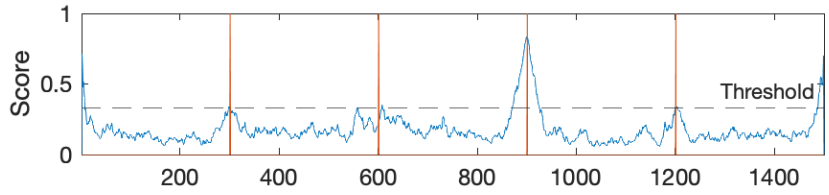
2. Multidimensional scaling (MDS) (Borg et al., 2012) performs a transformation from a high-dimensional space to a 2-dimensional space for visualisation by preserving as well as possible the pairwise global distances in both spaces. We first split S1 data into intervals and then calculate the distributional dissimilarity between different intervals based on IDK and GDK. The split intervals are shown as points in Figure 2.



(a) S1

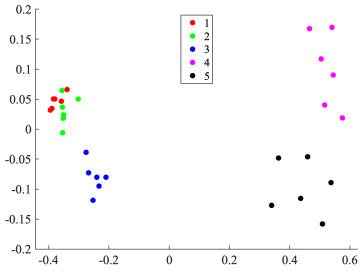


(b) Using GDK

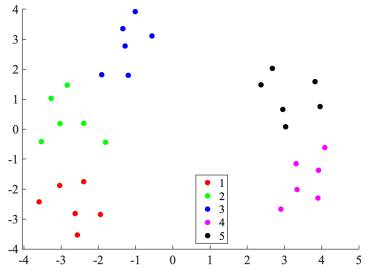


(c) Using IDK

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 added outliers 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.

2. Related Work

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

Table 1: A Comparison of CPD Methods. n in time complexity is the size of the window or the number of observations.

Method	Type	Time Complexity	Feature Learning	Core Method	Distribution focus
ARMA (Box, 2013)	Point-regression	$O(n)$	No	Moving Average	No
ARGP (Candela et al., 2003)	Point-regression	$O(n^3)$	No	Kernel-based	Yes
ARGP-BOCPD (Saatcci et al., 2010)	Point-regression	Polynomial	No	Kernel-based	Yes
FLOSS (Gharghabi et al., 2019)	Point-regression	$O(nlog(n))$	No	Shape-based	No
ESPRESSO (Deldari et al., 2020)	Point-regression	$O(n^2)$	No	Shape-based	No
aHSIC (Yamada et al., 2013)	Distribution	$O(n^2)$	No	Kernel-based	Yes
RuLSIF (Liu et al., 2013)	Distribution	Unstated	No	Kernel-based	Yes
Mstat (Li et al., 2015)	Distribution	$O(n)$	No	Kernel-based	Yes
e-divisive (Matteson & James, 2014)	Distribution	$O(n^2)$	No	Statistical-based	Yes
KLIEP (Kawahara & Sugiyama, 2009)	Distribution	$O(n^2)$	No	Statistical-based	Yes
RNN En/Decoder (Cho et al., 2014)	Deep Learning	Polynomial	YES	Supervised	No
LSTNet (Lai et al., 2018)	Deep Learning	Polynomial	YES	Supervised	No
KL-CPD (Chang et al., 2019)	Deep Learning	Polynomial	YES	Kernel-based	Yes
TS-CP^2 (Deldari et al., 2021)	Deep Learning	Polynomial	YES	self-supervised	No
iCID	**Distribution**	**O(n)**	**No**	**Kernel-based**	**Yes**

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) (Box, 2013) 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 assumptions about the probability distribution. *Autoregressive Gaussian Process* (ARGP) (Candela et al., 2003) as a non-parametric model can be considered as a nonlinear version of AR based on Gaussian process for time series prediction. *ARGP-BOCPD* (Saatcci et al., 2010) uses ARGP 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* (Gharghabi et al., 2019) identifies the position of change-points by comparing changes in interval shape pattern. *ESPRESSO* (Deldari et al., 2020) combines shape patterns (based on FLOSS) and the statistic approach (information-gain based (Sadri et al., 2017)) to identify change-points to detect potential segment boundaries.

However, since outliers 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 the predicted value. Moreover, the shape-based (Gharghabi et al., 2019; Deldari et al., 2020) approaches usually have a high time complexity for intervals matching.

2.2 Distribution-Based CPD

Distribution-based CPD methods detect change-points by identifying the difference between two distributions via some statistic, where points in each consecutive interval in a data

stream are assumed to be a set of i.i.d points generated from some unknown distribution. They have shown promising performance in CPD tasks (Yamada et al., 2013; Liu et al., 2013; Li et al., 2015). These methods are usually non-parametric based kernel methods, thus, they are more robust to outliers in providing consistent results on different types of data distributions than point-regression based CPD algorithms.

aHSIC (Yamada et al., 2013) uses Gaussian distributional kernel to calculate the dissimilarity of past and future interval data separability to detect change-points. The method *Mstat* proposed in (Li et al., 2015) uses *M-statistics* to detect change-points based on Gaussian distributional kernel *Maximum Mean Discrepancy* (MMD) for a two-sample test between the current interval distribution with a reference interval distribution having no change-points. *RuLSIF* (Liu et al., 2013) uses a density-ratio estimation for CPD and uses Gaussian distributional kernel to model the density-ratio distributions between intervals of time series. However, existing GDK-based CPD methods have limited sensitivity to subtle changes, as they rely on a data-independent kernel that cannot adapt to the varied density of the data. Compared to GDK-based CPD methods, IDK has a finite-dimensional feature map, and the data-dependent property of IDK makes `iCID` easier to detect subtle change-points.

Another way to measure the dissimilarity between distributions is to use a statistical-based method. *e-divisive* (Matteson & James, 2014) uses a U-statistic test to measure the dissimilarity between the distributions of two adjacent intervals to identify change-points, but it has a time complexity $O(n^2)$ that cannot handle the large scale of data. *KLIEP* (Kawahara & Sugiyama, 2009) detects change-points by using the Kullback Leibler (KL) divergence to estimate the probability of density between two intervals. However, *KL-divergence* is sensitive to outliers or noise.

In summary, a general approach for distribution-based CPD methods is to treat each interval as a distribution and use a measure to compute the similarity between them. The main difference among these methods lies in the choice of the similarity measure.

2.3 Deep Learning-Based CPD

Deep learning based CPD methods have been developed in recent years. *RNN Encoder-Decoder* (Cho et al., 2014) uses two *Recurrent Neural Networks* (RNN) as encoder and decoder respectively to learn the nonlinear features of the interval and forecast time series. *LSTNet* (Lai et al., 2018) 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* (Chang et al., 2019) 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 the change. The modelled distribution is a surrogate distribution in order to generate points to simulate points from a changed distribution where points are few (or unavailable) in practice. *TS-CP*^{2} (Deldari et al., 2021) 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 a significant representation difference between time-adjacent intervals.

However, deep-learning-based methods are inefficient in handling massive data streams because they usually require a large amount of data for expensive training. Thus, they cannot be used in the online setting.

3. Problem Definition

3.1 Change-Point Detection

Existing works (Chang et al., 2019) 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 an 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 series of observations $\{x_1, x_2, x_3, ...\}$, where each distribution is estimated from w observations before and after x_t which are assumed to be their i.i.d. samples.

Definition 1. *Given a series of observations $\{x_1, x_2, x_3, ...\}$, 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 $x_t \sim \mathcal{P}_L^t$ or $x_t \sim \mathcal{P}_R^t$;*
- ***outlier*** *if $\mathcal{P}_L^t = \mathcal{P}_R^t$, and neither $x_t \sim \mathcal{P}_L^t$ nor $x_t \sim \mathcal{P}_R^t$.*

The small difference in distributions can be further defined as **subtle-change**: x_t is a subtle-change-point if the difference between $\mathcal{P}_L^t$ and $\mathcal{P}_R^t$ is small.

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 interval. Many existing CPD methods (Yamada et al., 2013; Li et al., 2015; Chang et al., 2019) 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 is generated from a mixture of $\mathcal{P}_L$ and $\mathcal{P}_R$ within the period. 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., many points 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 data stream is impractical and unnecessary due to two main reasons:

1. It is time-consuming for any method to check every point in a data stream, especially since the data stream can be infinite;
2. In the period of gradual change or continuous change, there are many 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-point. In addition, the CPD problem is a special case of CID problem. Most of the existing works on CPD focus on identifying the exact locations of changes in a dataset. However, we argue that it is more efficient and effective to detect change intervals instead, as many datasets have inaccurate labels that can affect the performance of point-based methods significantly. In contrast, interval-based methods can handle such label issues more robustly because it detects the intervals that cover the change-points instead of the exact location of the change-points.

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 $f(\cdot)$ to calculate the similarity between two distributions, a f_τ-change interval is formally defined as follows.

Definition 3. *Given two temporal adjacent intervals 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* (Csiszár, 1975), *JS (Jensen-Shannon) divergence* (Manning & Schutze, 1999), *Wasserstein metric* (Vaserstein, 1969), and *MMD* (Borgwardt et al., 2006).

We propose a generic method for CID which can be converted to detect CPD by using sliding windows. The details of the proposed CID and CPD method are provided in Section 4 and 7.1, respectively. Nevertheless, both methods overcome the three unresolved challenges (a), (b) & (c) stated in Section 1.

4. Change-interval Detection Based on Distributional Kernel

We first 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. The third subsection introduces the automatic parameter selection for offline and online versions of `iCID` for real practice.

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) (Muandet et al., 2017), 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 $\widehat{\Phi}(\mathcal{P}_X) = \frac{1}{|X|}\sum_{x\in X}\Phi(x)$ is the feature map of the kernel mean embedding.

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 (Ting et al., 2022) and time series anomaly detection (Ting et al., 2022a).

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 $\mathbb{H}_\psi(D)$ denotes 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 $\mathcal{D}$ (Qin et al., 2019).

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:*

$$\kappa_I(x, y \mid D) = \mathbb{E}_{\mathbb{H}_\psi(D)}[\mathbb{1}(x, y \in \theta \mid \theta \in H)]. \tag{3}$$

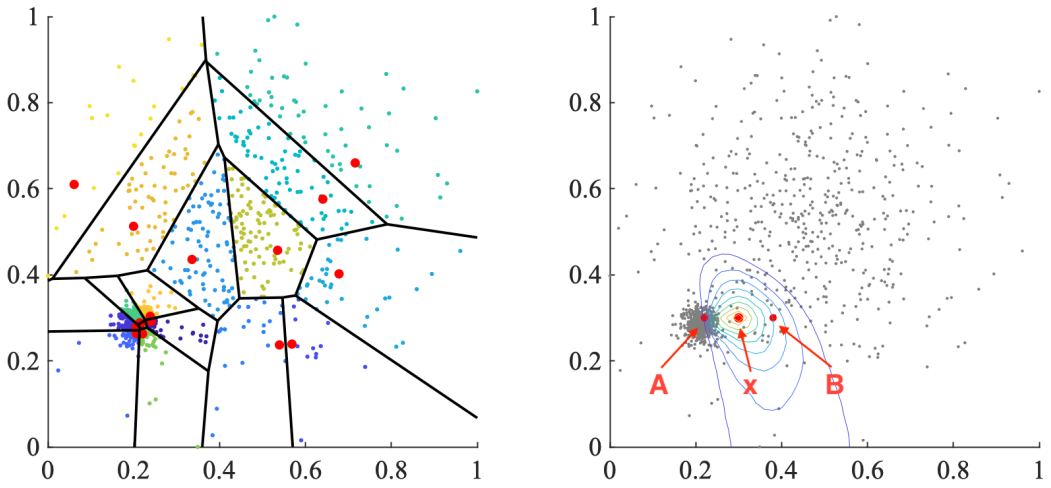
In practise, κ_I is constructed using a finite number of partitionings $H_i, i = 1, \ldots, t$, (t=200 as default) where each H_i is generated by using randomly subsampled $\mathcal{D}_i \subset D$; θ is a shorthand for $\theta[z]$; ψ is the sharpness parameter:

$$\kappa_I(x, y \mid D) = \frac{1}{t}\sum_{i=1}^{t}\mathbb{1}(x, y \in \theta \mid \theta \in H_i). \tag{4}$$

In this paper, we use the Voronoi diagram (Aurenhammer, 1991) to partition the space, i.e., each $H \in \mathbb{H}_\psi(D)$ is a Voronoi diagram and each sample point $z \in \mathcal{D}$ is the centre of each cell in Voronoi diagram. We illustrate a partitioning of H on the S2 dataset with $\psi = 16$ in Figure 3a, i.e, the data space is partitioned into 16 cells based on the 16 red points.

Figure 3a shows that the dense region is partitioned into smaller cells than the sparse region. Consequently, data points in the dense region (being in smaller cells) have lower

similarity scores than data points with the same inter-point distance in the sparse region (being in the larger cells). Figure 3b illustrates this phenomenon, such that data point $x = (0.3, 0.3)$ is more similar to $B = (0.38, 0.3)$ from a sparse region than $A = (0.22, 0.3)$ from a dense region, although x has the same distance to both A and B.



(a) Sample Partitioning (b) Contour

Figure 3: Illustration of the similarity calculation using Isolation kernel ($\psi = 16$): (a) An example partitioning H. (b) Contours with reference to point $x = (0.3, 0.3)$.

Isolation kernel has a finite feature map, which is defined as follows (Ting et al., 2022b, 2020):

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 (4) using Φ is shown as follows:

$$\kappa_I(x, y \mid D) = \frac{1}{t} \langle \Phi(x|D), \Phi(y|D) \rangle . \tag{5}$$

Given the feature map Φ (defined in Definition 6) and Equation (5), 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 (Ting et al., 2020):

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

$$\mathcal{K}_I(\mathcal{P}_X, \mathcal{P}_Y \mid D) \quad = \quad \frac{1}{t} \left\langle \widehat{\Phi}(\mathcal{P}_X|D), \widehat{\Phi}(\mathcal{P}_Y|D) \right\rangle , \tag{6}$$

where $\widehat{\Phi}(\mathcal{P}_X|D) = \frac{1}{|X|} \sum_{x\in X} \Phi(x|D)$ is the empirical feature map of kernel mean embedding.

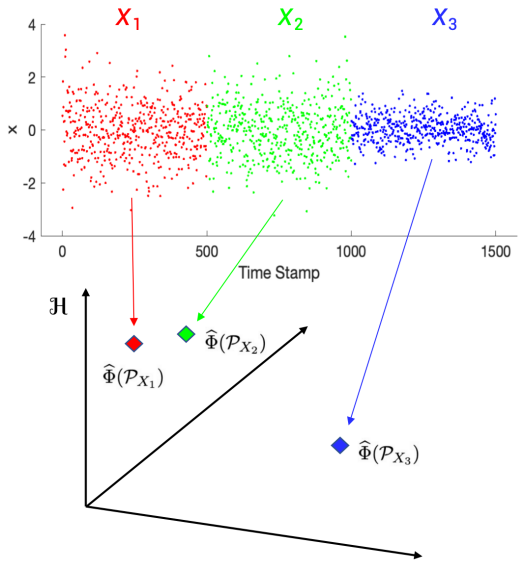


Figure 4: Feature mapping of each time interval X_i as a distribution to a point $\widehat{\Phi}(\mathcal{P}_{X_i})$ in the feature space of IDK or KME. Similar distributions are mapped into the same region of the feature space; and different distributions are mapped into different regions.

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

$$\widehat{\mathcal{K}}_I(\mathcal{P}_X, \mathcal{P}_Y \mid D) = \frac{\langle \widehat{\Phi}(\mathcal{P}_X|D), \widehat{\Phi}(\mathcal{P}_Y|D) \rangle}{\sqrt{\langle \widehat{\Phi}(\mathcal{P}_X|D), \widehat{\Phi}(\mathcal{P}_X|D) \rangle} \sqrt{\langle \widehat{\Phi}(\mathcal{P}_Y|D), \widehat{\Phi}(\mathcal{P}_Y|D) \rangle}}. \tag{7}$$

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 temporal 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 two intervals X and Y hereafter.

4.2 The Proposed `iCID`

The main idea of **I**solation Distributional Kernel-based **C**hange **I**nterval **D**etection (`iCID`) is to treat each interval in a data stream as a distribution and use IDK as the measure

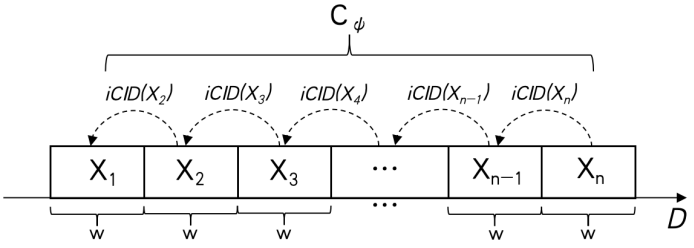


Figure 5: Illustration of `iCID` calculation.

to calculate the dissimilarity between distributions of an interval and its earlier adjacent interval. Each interval is assigned an iCID score based on this dissimilarity, if the score exceeds a threshold, the interval is detected as a change interval.

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 $\widehat{\Phi}(\mathcal{P}_{X_i})$ in the feature space of IDK which is shown in Figure 4, and computes the dissimilarity between intervals X_i and its adjacent interval X_{i-1}, as illustrated in Figure 5. Let $C_\psi = \{\texttt{iCID}(X_1), \texttt{iCID}(X_2), ..., \texttt{iCID}(X_N) \mid \psi, D\}$ be the set of all scores of N intervals in D, obtained using ψ, a parameter for `iCID`. The `iCID` score of interval X_i is:

$$\texttt{iCID}(X_i) = \mathfrak{S}(X_i, X_{i-1}) = 1 - \widehat{\mathcal{K}}_I(\mathcal{P}_{X_i}, \mathcal{P}_{X_{i-1}} \mid D). \tag{8}$$

where $\widehat{\mathcal{K}}_I(\mathcal{P}_{X_i}, \mathcal{P}_{X_{i-1}} \mid D)$ is the normalised IDK similarity between the distributions of two adjacent intervals.

Here we propose two versions of `iCID`, i.e., the offline version and the online version. Recall that IDK uses subsamples $\mathcal{D}$ (each having ψ points, as described in Section 4.1) to build the partitionings $H \in \mathbb{H}_\psi(D)$. In the offline version, the subsamples $\mathcal{D}$ are uniformly sampled from the whole dataset D to calculate the `iCID` scores for all intervals. However, for the online version with limited memory, $\mathcal{D}$ is sampled from the latest k data points in order to compute the `iCID` score for the current interval. This is because the latest data points are the best representation of the current interval.

4.3 Automatic Parameter Selection

Since IDK has a key parameter ψ, here we propose a best ψ selection method as follows. Assuming that change-points are rare, the `iCID` scores in a data stream would have a few large scores only while others have small scores, i.e., the `iCID` scores are stable overall. Therefore, we aim to search for the best parameter ψ in IDK in order to produce the most stable `iCID` scores.

Let $C_\psi = \{\texttt{iCID}(X_1), \texttt{iCID}(X_2), ..., \texttt{iCID}(X_N) \mid \psi, D\}$, the best ψ leads to a maximum of stability or a minimum of instability, i.e.,

$$\psi^* = \underset{\psi}{\operatorname{argmin}}\, \bar{E}(C_\psi), \tag{9}$$

where $\bar{E}(.)$ is a measure of instability, e.g., approximated entropy estimation (Pincus, 1991), variance or Gini coefficient (Gini, 1912).[3] Note that a lower $\bar{E}(.)$ value means a higher stability.

Once obtaining the iCID score for each interval, we can set a threshold τ to identify the changing interval X_j with $\texttt{iCID}(X_j) > \tau$. A threshold τ can be identified as

$$\tau(C_{\psi^*}) = \mu + \alpha \times \sigma, \tag{10}$$

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

For the offline version, C_ψ is calculated based on all intervals, and we can search the ψ^* in a reasonable pre-defined range. Algorithm 1 illustrates the steps of offline `iCID`.

Algorithm 1 `Offline_iCID(D,w,Ψ)`

Input: Dataset D; Window Size w; Subsample Size List Ψ
Output: C_{ψ^*} - a set of N Interval Scores
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: Search the best ψ^* from the Ψ, i.e., $\psi^* = \underset{\psi}{\operatorname{argmin}}\, \bar{E}(C_\psi)$
3: **Return** C_{ψ^*}

For the online version, the ψ^* is obtained based on a reference dataset D, e.g., first k points of the streaming data. For calculating the iCID of a current interval X_i, the ψ^* subsamples used to build IDK is obtained from the latest streaming dataset D' with k points ending at X_i, such that the feature map keeps updating to adapt the distribution of the current interval. Algorithm 2 illustrates the steps of online iCID.

Algorithm 2 `Online_iCID(D,D',w,Ψ)`

Input: Reference dataset D; Current streaming data D'; Window Size w; Subsample Size List Ψ
Output: S - the iCID score of the last interval in D'
1: $\psi^* = \text{Offline\_iCID}(D, w, \Psi)$
2: Let X_N and X_{N-1} be the last two adjacent intervals in D' where each interval has w points
3: $S = 1 - \widehat{\mathcal{K}}_I(\mathcal{P}_{X_N}, \mathcal{P}_{X_{N-1}} \mid \psi^*,\ D')$
4: **Return** S

Note that the automatic parameter selection does not involve optimisation or learning. In other words, the entire process of `iCID` requires no optimisation or learning for both the online and offline versions. This is the advantage of most existing kernel and statistical-based change point detection methods.

3. The following experiment results are obtained by using the approximate entropy (Pincus, 1991). We find that the results of using the other two measurements are similar. The details of the comparison are given in Appendix B.

5. Experimental Settings

In this section, we introduce the datasets and parameter settings used 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 every dimension is in the same range of [0, 1] before experiments are conducted.

5.1.1 Real-world Datasets

- **Yahoo-16**[4] records hardware resource usage during the operation of the PNUTS/Sherpa database (e.g. CPU utilization, memory utilization, disk utilization, network traffic, etc). We directly used the dataset from previous work (Chang et al., 2019) which chose the 16^{th} of a total of 68 representative time series after removing some intervals with duplicate patterns in anomalies. Yahoo-16 dataset had 5 type I change-points.

- **USC-HAD**[5] (Zhang & Sawchuk, 2012) includes 14 subjects and 12 daily activities. Each subject was fitted with a 3-axial accelerometer and 3-axial gyroscope, which are fixed in front of the right hip joint and sampled at 100 Hz. We reused the same dataset from (Deldari et al., 2021) which randomly chose 30 activities from the first six participants and stitched the selected recordings together in a random manner. USC-HAD dataset has type I sudden-change-points mentioned in Section 3.1.

- **Google-Trend**[6] 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 sudden-change-points.

- **Well_Log**[7] (Ó Ruanaidh & Fitzgerald, 1996) contains measurements of the nuclear magnetic response and conveys information about the structure of the rock. The series has been sampled every 6 iterations to reduce the length (van den Burg & Williams, 2020). Well_log dataset has 10 type I change-points. In addition, it contains some outliers which are defined in Section 3.1.

- **Weather**[8] 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.

4. https://webscope.sandbox.yahoo.com/catalog.php?datatype=s
5. http://sipi.usc.edu/had
6. https://github.com/zhaokg/Rbeast/blob/master/Matlab/testdata/googletrend.mat
7. https://github.com/alan-turing-institute/TCPD/tree/master/datasets/well_log
8. https://datahub.io/core/global-temp#readme

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

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	$\alpha \in \lceil 0, 0.1, 0.2, ..., 3 \rceil$

- **HASC**[9] (Kawaguchi et al., 2011a, 2011b) is human activity data with 3 dimensions provided by HASC challenge 2011. We used the same subset of the HASC dataset from *KL-CPD* (Chang et al., 2019). HASC dataset has type I sudden-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 have type I obvious-change and sudden-change-points.

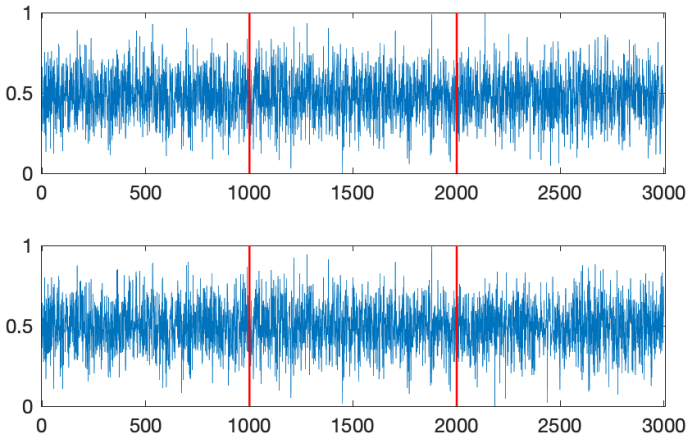
- **S1:** There are 5 one-dimensional Gaussians of 300 points with the same $\mu = 0$ but different $\sigma = 1, 2.2, 4.3, 48.3, 28.3$, respectively. In addition, there are 5 outliers located at 89, 117, 139, 523 and 537, respectively. The data distribution is shown in Figure 1a.

- **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 6. 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.

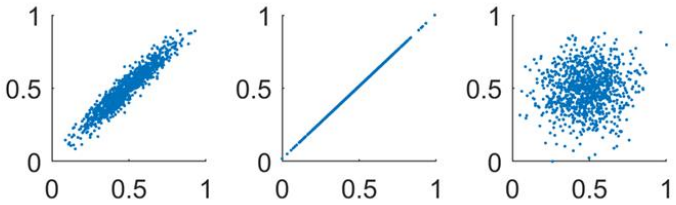
5.2 Parameter Setting

Offline `iCID` has three inputs: Dataset D, window Size w and Subsample Size List Ψ. For the online `iCID`, we used the first half of the whole datasets as the reference D and applied

9. http://hasc.jp/hc2011/



(a) 1-dimensional plot



(b) 2-dimensional plot

Figure 6: 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).

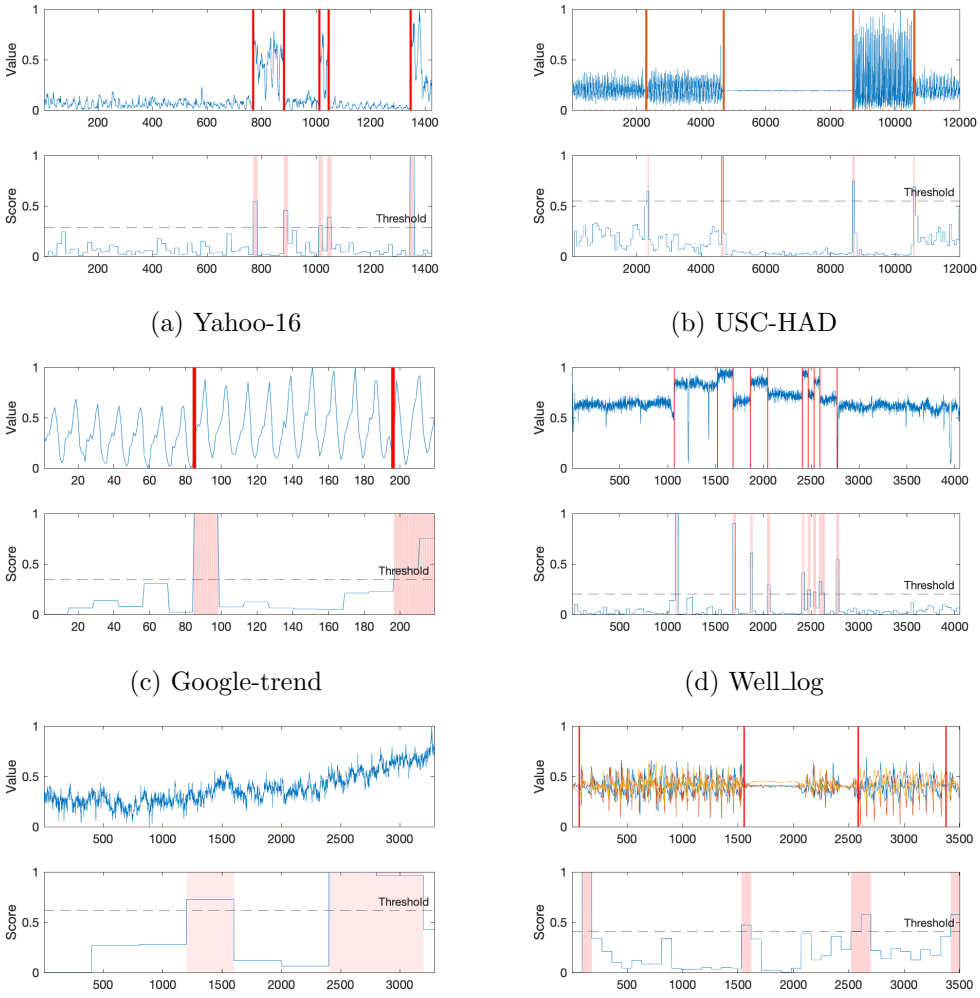
a sliding window to generate the current streaming data D'. The size of sliding window was set to the half of the dataset length. The parameter search ranges are provided in Table 3.

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 outliers and scalable to large data sizes?
- Does `iCID` have any advantage over other CPD algorithms, in terms of detection performance and time complexity?

To answer the first two questions, we present the empirical evaluation results of the online and offline versions of `iCID` on the real-world and synthetic datasets in the next two subsections. The last two subsections provide the answer to the third question by reporting



(a) Yahoo-16

(b) USC-HAD

(c) Google-trend

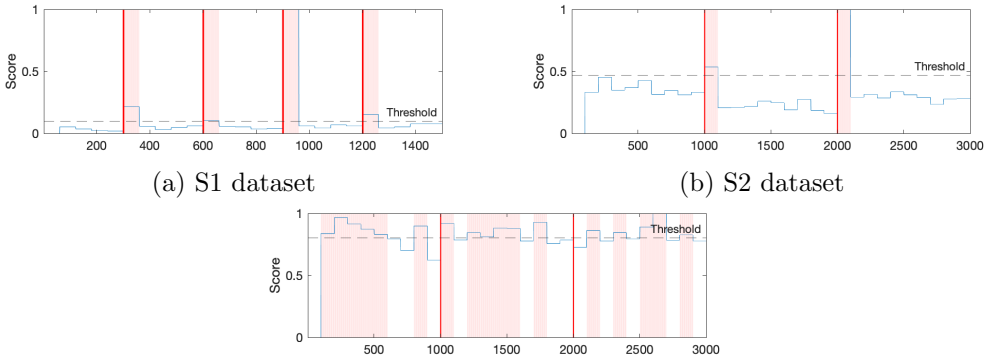
(d) Well_log

(e) Weather

(f) HASC

Figure 7: Offline iCID results on real-world datasets. In each subfigure (on one dataset), the first row shows the data distribution with change-point locations marked by red bars. The second row shows the change scores and the detected change intervals, marked as red areas. Since the USC-HAD and HASC datasets are very large, here we only show the results on a segment that includes obvious changes for demonstration.

the quantitative evaluation results on the two largest real-world datasets, in comparison with 6 state-of-the-art CPD algorithms.



(a) S1 dataset

(b) S2 dataset

(c) S2 dataset (only using the 1st feature)

Figure 8: Offline `iCID` results on synthetic datasets. (c) is the score of just using 2nd feature in S2 dataset.

6.1 Offline `iCID`

The performance of offline `iCID` on 6 real-world datasets and 2 synthetic datasets are visualised in Figures 7 and 8, respectively [10]. Note that a CID method detects a change interval that contains at least one ground truth change-points is regarded to have correctly identified a change interval, as stated in Definition 2.

Observations on the real-world datasets are given below:

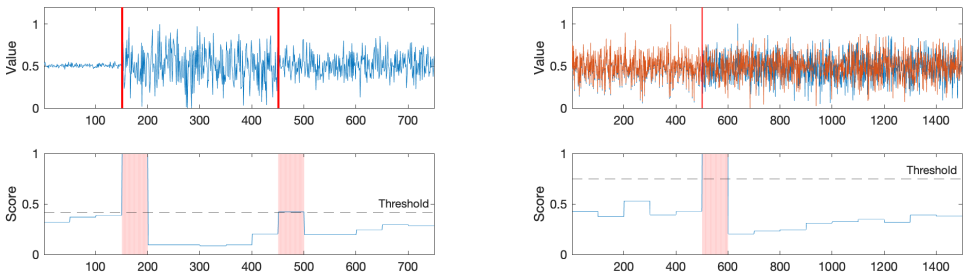
- In S2 dataset, there is no distribution change on any individual attribute. The change-points can only be detected by considering both attributes, as shown in Figure 6b. The results presented in Figure 8b show that `iCID` has the capability to detect the two change-intervals containing the change-points on this multivariate dataset.

- The Google-trend dataset is a periodic dataset. `iCID` is able to detect the change intervals which contain the all change-points. It is imperative to note that for optimal results in a periodic dataset, the window size should ideally approximate the length of a single period.

- As shown in Figure 7d, the Well_log dataset contains a few outliers which can easily be mistaken as change-points. `iCID` successfully found 9 out of 10 change-points without misclassifying any outliers. 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 7e. `iCID` discovers that there are 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.

10. We also evaluated e-divisive and found it has shown good performance on S1 dataset, i.e., the identified top four change points are very close to the ground truth. We didn't run e-divisive on other datasets due to its high time complexity of $O(n^2)$.



(a) S1 (b) S2

Figure 9: Online iCID results on the Synthetic datasets.

In summary, the offline iCID detects all three types of change-points and can tolerate outliers with the effective and automatic parameter ψ setting. It detected all change intervals that contain the change-points on all datasets. The only exception is one change-point on the Well_log dataset.

6.2 Online iCID

The performance of online iCID evaluated on both synthetic datasets and real-world datasets are shown in Figures 9 and 10, respectively.

We observe similar results to the offline version. The online iCID can detect all change-points on all datasets except Well_log dataset. Figure 10d shows that iCID does not detect the 6th change-point on the Well_log dataset, but this change-point should be considered as an anomaly instead of a change-point according to Definition 1. Moreover, online iCID produces a false-positive change-point on the Google-trend dataset as shown in Figure 10c. This error can be easily rectified by manually setting a better ψ value.

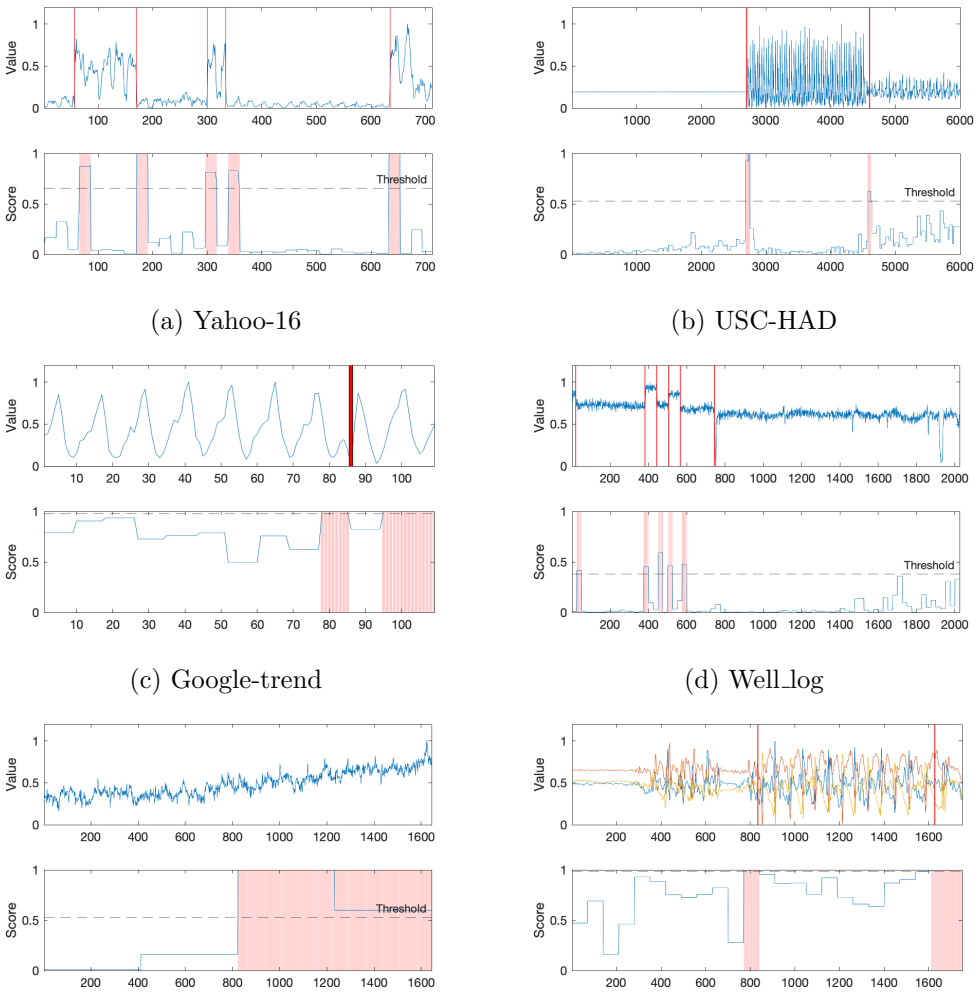
It is worth mentioning that the performance online version is a bit lower than that of the offline version. Recall that online iCID requires $\mathcal{D}$ used in IDK to be continuously updated from a limited number of previous points. As a result, $\mathcal{D}$ in online iCID may be less representative than it sampled from the whole dataset used in the offline iCID.

6.3 Comparison With State-of-the-Art Methods

In this subsection, we compare the performance of iCID against 6 state-of-the-art (SOTA) methods, including *ESPRESSO* (Deldari et al., 2020), *aHSIC* (Yamada et al., 2013), *RuLSIF* (Liu et al., 2013), *KL-CPD* (Chang et al., 2019) and *TS-CP*^{2} (Deldari et al., 2021), on the two largest real-world datasets, i.e., USC-HAD and HASC.

For a fair comparison, we followed the same evaluation strategy used previously (Deldari et al., 2021) and calculated the highest F1-score of iCID with manually tuned parameters and detection margins. Only if a detected change-point/interval is located within the specified margin of the ground truth change-point, then it is a true positive [11].

11. Researchers can also expand the change points detected by existing change point methods to change intervals of the same width as used in proposed method, and then evaluate as done for the proposed method.



(a) Yahoo-16

(b) USC-HAD

(c) Google-trend

(d) Well_log

(e) Weather

(f) HASC

Figure 10: 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.

Table 4 presents the performance comparison. We have the following observations regarding the comparison between `iCID` and the 6 SOTA algorithms:

- `iCID` has the highest F1 score on the HASC dataset when the detection margin is 100 and 200. The best window sizes are only around half of the corresponding detection margins.

- *ESPRESSO* is a shape-based CPD method, its performance is usually far below than that of `iCID` on both datasets under different margin values. The reason can be the fact that shape-based CPD methods are intolerance to outliers.

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-CP*^{2} paper (Deldari et al., 2021).

Dataset	Methods	Best Window	F1-score	Best Window	F1-score	Best Window	F1-score
	Detection Margin	60		100		200	
HASC	*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-CP^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	*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-CP^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

- Comparing the three Gaussian-based kernel methods, i.e., *Mstat*, *aHSIC* and *RuLSIF*, `iCID` outperforms them on both datasets with the detection margin greater than or equal to 100.

- With a large detection margin, `iCID` performs significantly better than *TS-CP*^{2} on HASC, but is comparable to *TS-CP*^{2} on USC-HAD. *TS-CP*^{2} is a self-supervised deep learning-based method that needs heavy training, whereas `iCID` requires no learning.

- The F1 score of a model shall increase if the detection margin increases, i.e., a larger margin reduces potential misalignments between the labels and the change score due to lag. However, the F1 scores of two deep learning-based methods, i.e., *KL-CPD* and *TS-CP*^{2}, decrease as the detection margin increases. The reason can be the instability of these deep learning methods.

- Based on the results on the largest detection margin, `iCID`'s closest contenders are *TS-CP*^{2} and *ESPRESSO* on both datasets. `iCID` win on a large F1 difference on HASC and a small one on USC-HAD.

We also investigate two variants of `iCID`, i.e., `iCID(MMD)` and `gCID(MMD)`, both use *MMD* (Borgwardt et al., 2006). The latter uses Gaussian distributional kernel in MMD, as used in the two-sample test algorithms *Mstat* and *KL-CPD*; and the former uses Isolation distributional kernel as used in `iCID`.

It is interesting to mention that iCID and its two variants show similar performance on the datasets, as shown in Table 4. Moreover, gCID(MMD) shows much better performance than *Mstat* and *KL-CPD*. The reason can be using a fixed number of 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-CP*^{2} against a different number of points are shown in Figure 11. The results show that both versions of iCID have linear time complexity. Note that the offline iCID needs to run iCID multiple times and measure the stability 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 seconds and 4 seconds, respectively. Yet, a deep learning-based method *TS-CP*^{2} took about 170 GPU hours to train with the default parameter setting using a high-end GPU. Although *TS-CP*^{2} shows a linear runtime due to using a fixed number of epochs, it will cost much more time when the best parameters are to be tuned. Note that although the online *TS-CP*^{2} also runs a few seconds once the model has been trained, it still needs a long time to retrain and update the model to adapt to a new data distribution.

Section Summary

We find that iCID has three advantages over existing methods.

1. iCID can handle data with subtle-change-points and perform better than data-independent kernel-based methods such as *Mstat* which is shown in Figure 1.
2. iCID has a linear time complexity and does not require training. As a result, it runs much faster than deep learning-based methods in real-time.
3. In the online setting, iCID can select the best parameter to adapt to a new data distribution in real time; but deep learning-based methods have difficulty retraining or updating a model online in a short time.

7. Discussion

In this section, we first present the change-point detection task based on iCID, and then discuss the label issues in the real-world datasets as mentioned in Section 3.2. 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. Let the past interval $X_l^t = \{x_{t-w}, x_{t-w+1}, ..., x_{t-1}\}$ and the current interval $X_r^t = \{x_{t+1}, x_{t+2}, ..., x_{t+w}\}$

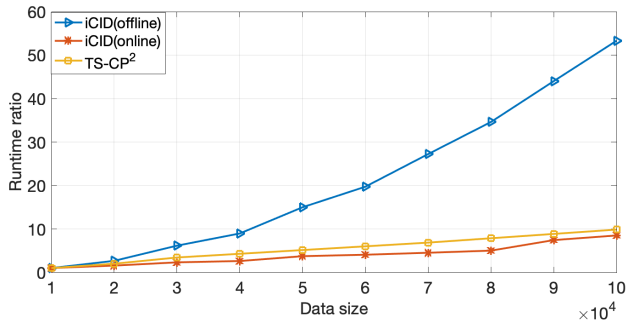


Figure 11: Scaleup test on a synthetic dataset. `iCID` is tested using Matlab R2022b with Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz and *TS-CP*^{2} is tested using Python with NVIDIA Quadro RTX 5000 GPU. For *TS-CP*^{2}, We include the total time of both training and testing processes. For `iCID`, we include the parameter search and model update time for offline and online versions, respectively.

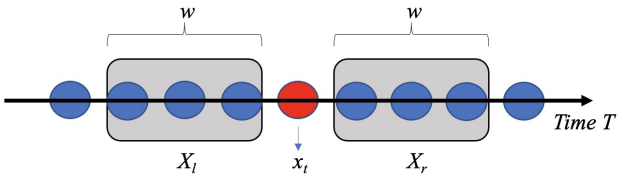


Figure 12: Overview of change-point detection based on IDK.

at each point x_t, as shown in Figure 12. The change-point is the x_t which has a large score more than a threshold τ, i.e., $\mathfrak{S}(X_l^t, X_r^t) \geq \tau$.

In order to detect all change-points in a data stream, we only need to slide the w-size intervals by one point to score every point in the stream. As a result, this `iCPD` method needs significantly more computational time than the interval-based `iCID`. This is because `iCID` computes a score for each non-overlapping interval only. The time complexity is the same to `iCID` with w times slower than `iCID`. Figure 1c shows the result of iCID using the sliding window method on the S1 dataset.

7.2 About Ground-Truth Label Issues

We visualised our evaluation of real-world datasets and found that many of them have different labelling problems. To illustrate, Figure 13 presents an interval of the 3-dimensional HASC dataset. Here we observe three types of issues as follows:

- **Wrong label position:** change-points are labelled at incorrect or inaccurate locations. 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 the points is labelled as change.

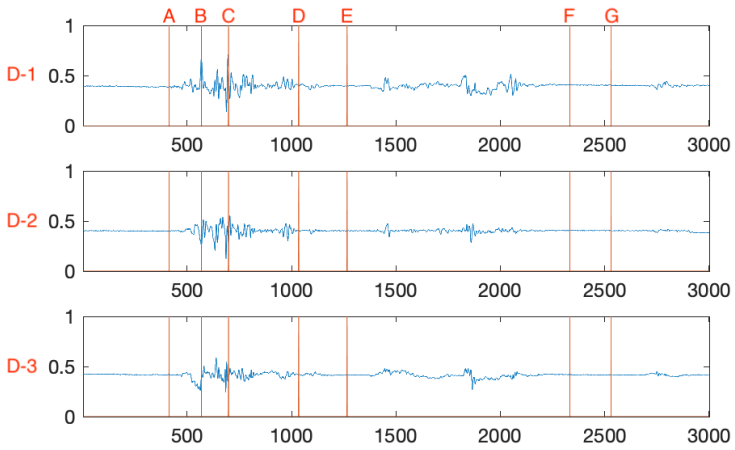


Figure 13: Demonstration of label issues on HASC dataset. D-1, D-2 and D-3 refer to different features.

- **Mislabelling:** normal points or outliers 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.

Similar labelling issues have been identified in time series datasets (Wu & Keogh, 2021; Freeman et al., 2021). 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 Multi-Dimensional Evaluation

In Section 6, we evaluated the performance of `iCID` on two multi-dimensional datasets: S2 (2-dimensional) and HASC (3-dimensional). To test the scalability of iCID on high-dimensional data, we conducted an additional experiment on the MNIST dataset, which has 784 dimensions and 10 classes. We simulated a data stream by ordering the appearance of each class one after another.

Figure 14 shows that the offline `iCID` can successfully detect all the change intervals. This implies that `iCID` can perform well if the distributional change between adjacent intervals can be measured using the Isolation distributional kernel in multi-dimensional datasets.

Having said that, when a high-dimensional dataset is plagued with the curse of high dimensionality which has multi-faceted issues (see e.g., (Keogh & Mueen, 2017)), the true distributional change can be masked. In this case, no methods can do well.

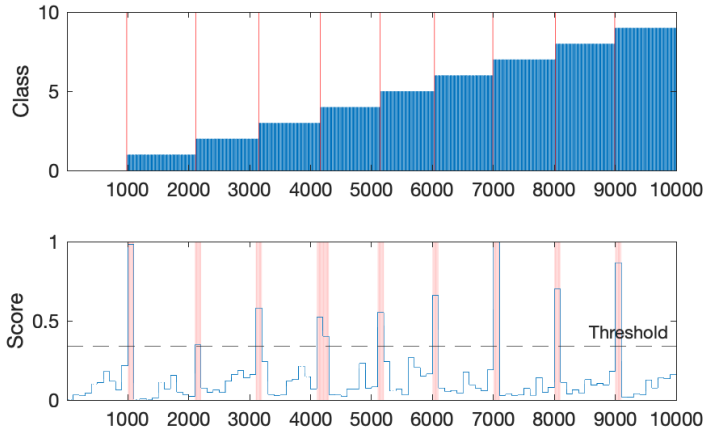


Figure 14: Offline `iCID` result on MNIST datasets. The first figure shows the 10 ordered classes in the MNIST dataset with change-point locations indicated by red bars. The second row shows the change score and the detected change intervals are marked as red areas.

7.4 Parameter Sensitivity Analysis

There are three parameters in the offline `iCID`, i.e., the subsample size ψ used to build Isolation distributional 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 Equation (9) can produce promising performance in the previous section. Figure 15 illustrates the effects of different ψ values on four datasets. It can be seen from the results that a large ψ value can perform well on these datasets. It is worth mentioning that ψ^* searched by our proposed method is close to the best ψ value on most datasets.

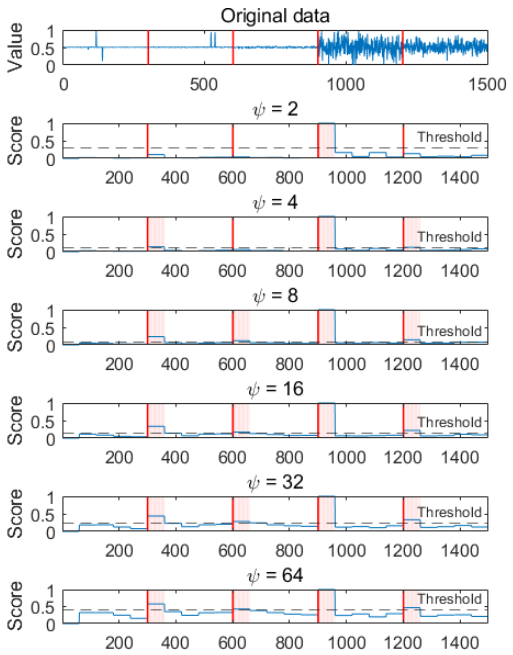
Regarding interval length w setting for CID methods in practice, a smaller length of the interval is preferable to detect the change interval sooner than using a larger length of the interval.

Figure 16 shows the sensitivity analysis of w on four 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.

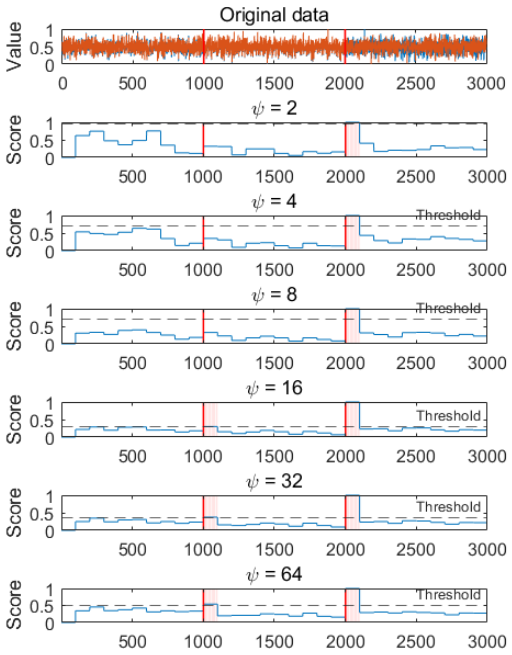
The other parameter α in Equation (10) is a power factor parameter that denotes the level of sensitivity for change interval identification. When we assume that the distribution of change scores conforms or is close to a Gaussian distribution, α can control the expected proportions of the sample as change 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.

For the online `iCID`, we report the results using the ψ^* searched from the first half of the data in previous experiments. It is possible to update the ψ^* for each new coming interval based on the latest k points, but it will cost much more running time. We report the results

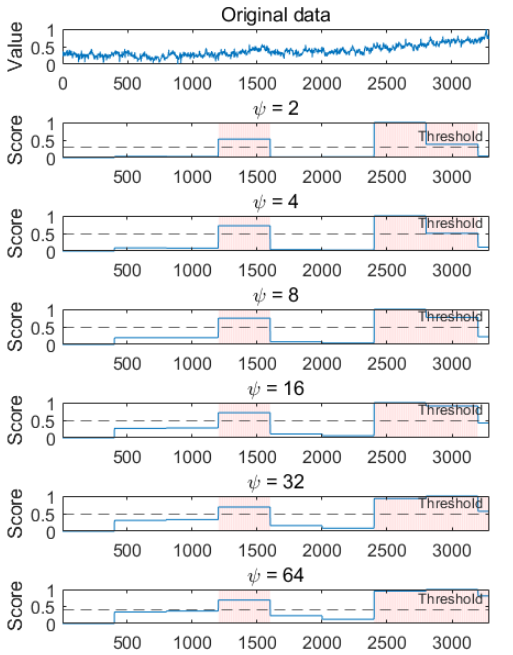
of online `iCID` using different ψ on four datasets in Figure 17. The results show that our parameter search strategy produces a similar `iCID` score to that using the best ψ value.



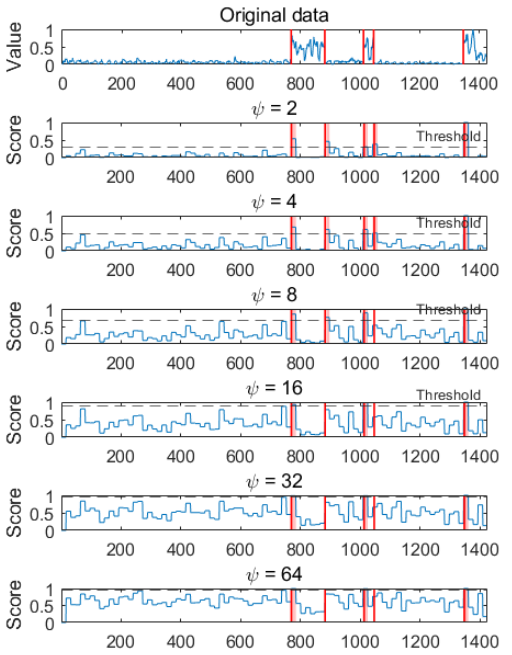
(a) S1 dataset with $w = 60$



(b) S2 dataset with $w = 100$

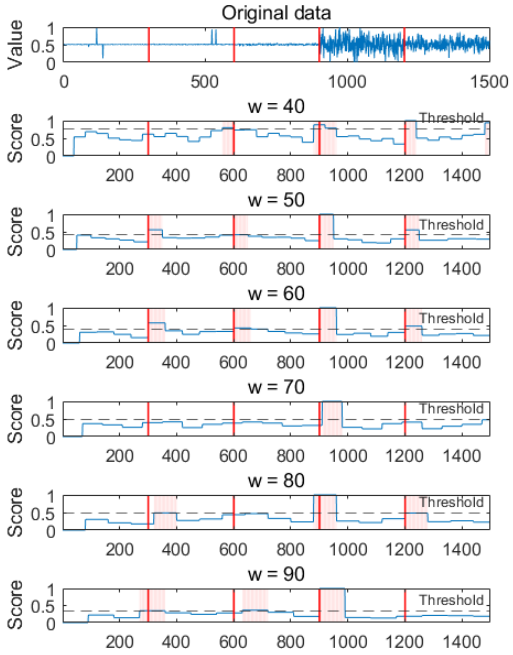


(c) Weather dataset with $w = 400$

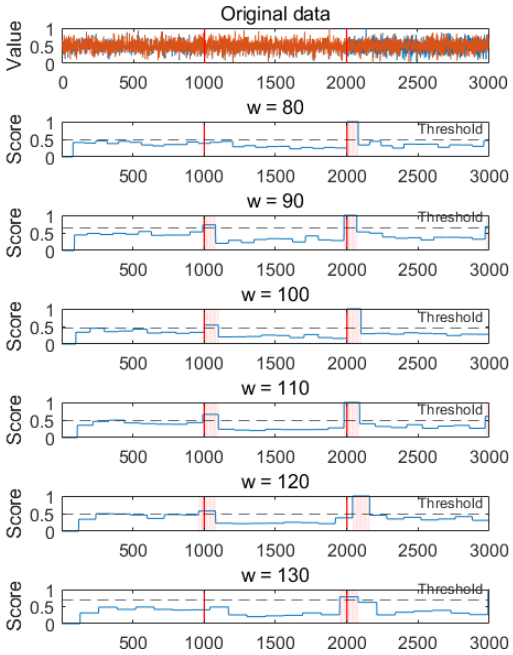


(d) Yahoo-16 dataset with $w = 16$

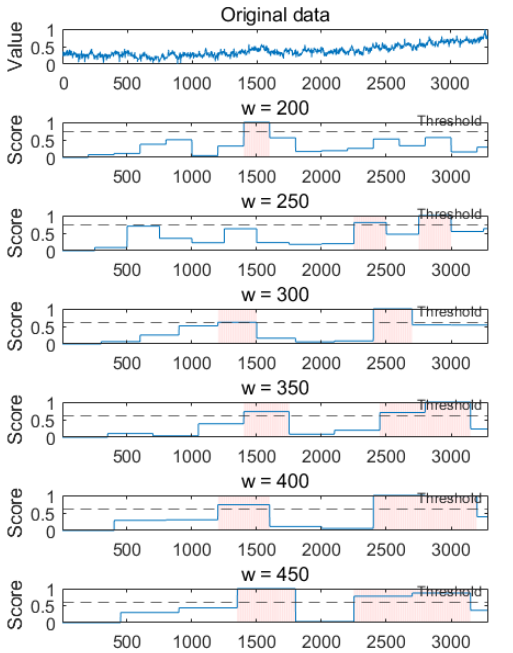
Figure 15: Sensitivity analysis of ψ on four datasets.



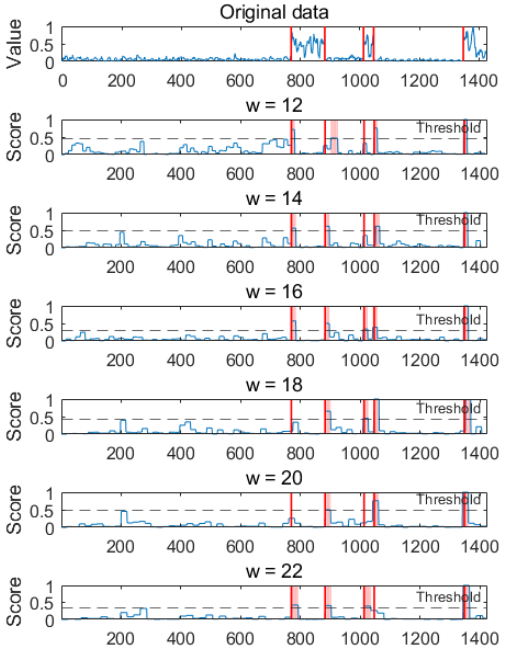
(a) S1 dataset with $\psi = 64$



(b) S2 dataset with $\psi = 64$

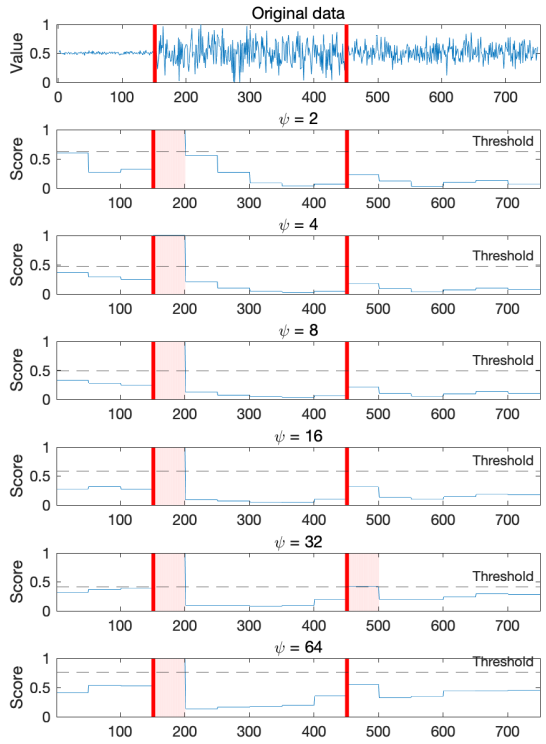


(c) Weather dataset with $\psi = 16$

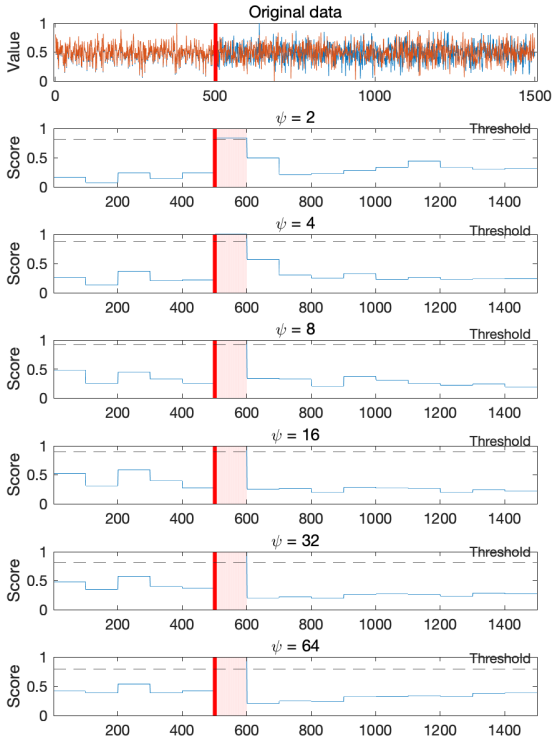


(d) Yahoo-16 dataset with $\psi = 2$

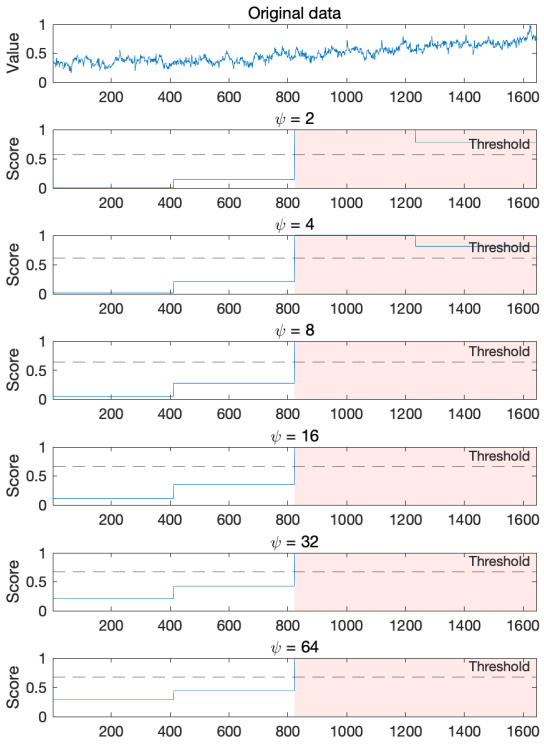
Figure 16: Sensitivity analysis of w on four datasets.



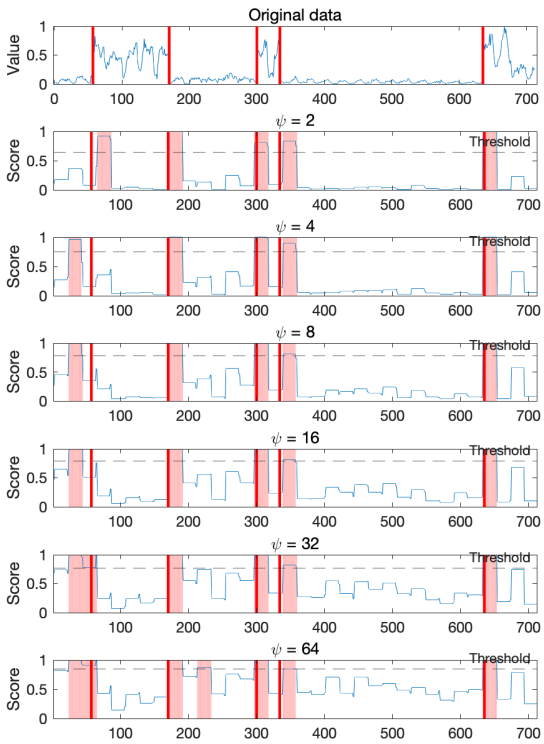
(a) S1 dataset with $w = 50$



(b) S2 dataset with $w = 100$



(c) Weather dataset with $w = 411$



(d) Yahoo-16 dataset with $w = 21$

Figure 17: Online `iCID` Sensitivity analysis of ψ on four datasets.

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 three 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` to detect drifts in the non-i.i.d. streaming data.

ACKNOWLEDGEMENTS

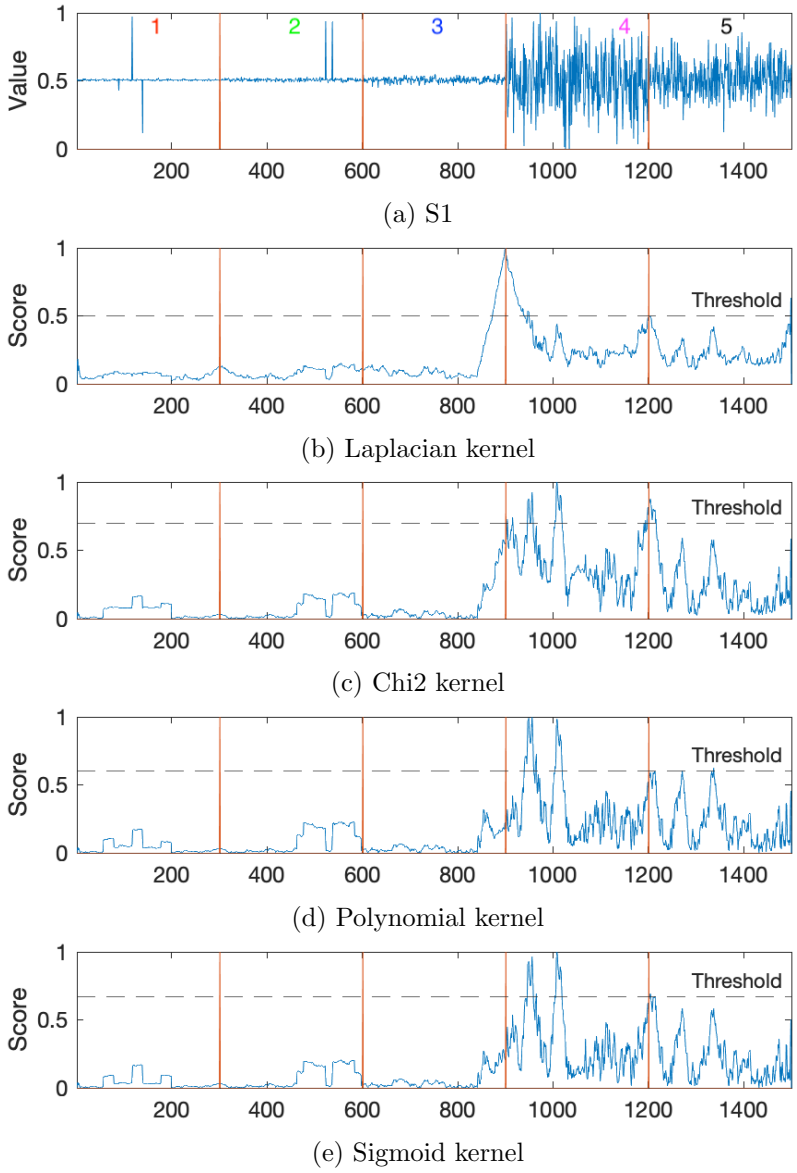
This project is supported by National Natural Science Foundation of China (Grant No. 62076120).

Appendix A. CPD Algorithms Based on Different Kernels

We compare the performance of the same CPD algorithms using 4 different kernels on the S1 dataset, including Laplacian, Chi2, Polynomial and Sigmoid kernels. Figure 18 shows the detection results indicating that none of them can reliably locate all the change-points in the dataset. This is because all these kernel methods are data-independent and have the similar issue as Gaussian kernel to identify the type I subtle-changes.

Appendix B. Alternative Criterion For Best ψ Selection

We compare the experiment result of `iCID` using three measurements for best ψ selection following Equation (9) on S1 dataset, including approximate entropy, variance and Gini coefficient. All these measurements can lead to the ψ which can detect all change intervals on S1 dataset, which is shown in Figure 19.



(a) S1

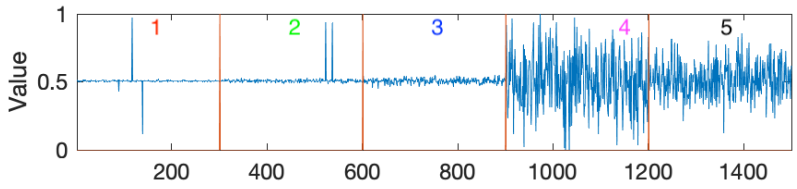
(b) Laplacian kernel

(c) Chi2 kernel

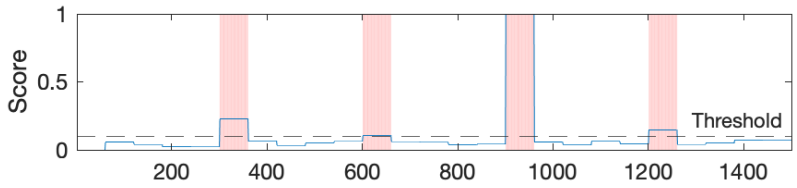
(d) Polynomial kernel

(e) Sigmoid kernel

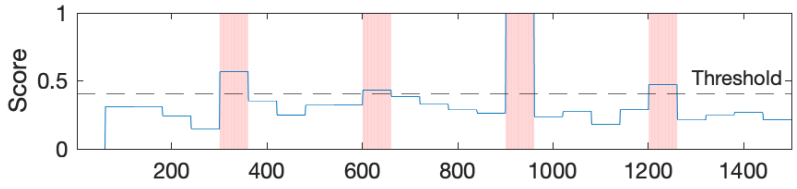
Figure 18: S1 dataset: Comparison of the same proposed distributional kernel-based CID algorithm using different kernels on S1.



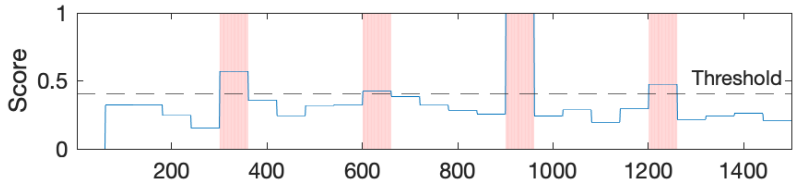
(a) S1



(b) Entropy



(c) Variance



(d) Gini Coefficient

Figure 19: S1 dataset: Comparison of the same proposed distributional kernel-based CID algorithm using different measurements on S1.

References

Aurenhammer, F. (1991). Voronoi diagrams—A survey of a fundamental geometric data structure. *ACM Computing Surveys*, *23*(3), 345–405.

Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*, Vol. 104. prentice Hall Englewood Cliffs.

Borg, I., Groenen, P. J., & Mair, P. (2012). *Applied multidimensional scaling*. Springer Science & Business Media.

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., & Smola, A. J. (2006). Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics*, *22*(14), 49–57.

Box, G. (2013). Box and jenkins: time series analysis, forecasting and control. In *A Very British Affair*, pp. 161–215. Springer.

Candela, J. Q., Girard, A., Larsen, J., & Rasmussen, C. E. (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.*, Vol. 2, pp. II–701. IEEE.

Chamroukhi, F., Mohammed, S., Trabelsi, D., Oukhellou, L., & Amirat, Y. (2013). Joint segmentation of multivariate time series with hidden process regression for human activity recognition. *Neurocomputing*, *120*, 633–644.

Chang, W.-C., Li, C.-L., Yang, Y., & Póczos, B. (2019). Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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

Deldari, S., Smith, D. V., Sadri, A., & Salim, F. (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), 1–24.

Deldari, S., Smith, D. V., Xue, H., & Salim, F. D. (2021). Time series change point detection with self-supervised contrastive predictive coding. In *Proceedings of the Web Conference 2021*, pp. 3124–3135.

Freeman, C., Merriman, J., Beaver, I., & Mueen, A. (2021). Experimental comparison and survey of twelve time series anomaly detection algorithms. *Journal of Artificial Intelligence Research*, *72*, 849–899.

Gharghabi, S., Yeh, C.-C. M., Ding, Y., Ding, W., Hibbing, P., LaMunion, S., Kaplan, A., Crouter, S. E., & Keogh, E. (2019). Domain agnostic online semantic segmentation for multi-dimensional time series. *Data mining and knowledge discovery*, *33*(1), 96–130.

Gini, C. (1912). *Variabilità e mutabilità: contributo allo studio delle distribuzioni e delle relazioni statistiche.[Fasc. I.]*. Tipogr. di P. Cuppini.

Kawaguchi, N., Ogawa, N., Iwasaki, Y., Kaji, K., Terada, T., Murao, K., Inoue, S., Kawahara, Y., Sumi, Y., & Nishio, N. (2011a). Hasc challenge: gathering large scale human activity corpus for the real-world activity understandings. In *Proceedings of the 2nd augmented human international conference*, pp. 1–5.

Kawaguchi, N., Yang, Y., Yang, T., Ogawa, N., Iwasaki, Y., Kaji, K., Terada, T., Murao, K., Inoue, S., Kawahara, Y., et al. (2011b). Hasc2011corpus: towards the common ground of human activity recognition. In *Proceedings of the 13th International conference on Ubiquitous computing*, pp. 571–572.

Kawahara, Y., & Sugiyama, M. (2009). Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM international conference on data mining*, pp. 389–400. SIAM.

Keogh, E., & Mueen, A. (2017). *Curse of Dimensionality*, pp. 314–315. Springer US, Boston, MA.

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (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*, pp. 95–104.

Li, S., Xie, Y., Dai, H., & Song, L. (2015). M-statistic for kernel change-point detection. *Advances in Neural Information Processing Systems*, *28*.

Liu, S., Yamada, M., Collier, N., & Sugiyama, M. (2013). Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, *43*, 72–83.

Lung-Yut-Fong, A., Lévy-Leduc, C., & Cappé, O. (2012). Distributed detection/localization of change-points in high-dimensional network traffic data. *Statistics and Computing*, *22*(2), 485–496.

Manning, C., & Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Matteson, D. S., & James, N. A. (2014). A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, *109*(505), 334–345.

Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B., et al. (2017). Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, *10*(1-2), 1–141.

Ó Ruanaidh, J. J. K., & Fitzgerald, W. J. (1996). *Numerical Bayesian Methods Applied to Signal Processing*. Springer.

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

Qin, X., Ting, K. M., Zhu, Y., & Lee, V. C. (2019). Nearest-neighbour-induced isolation similarity and its impact on density-based clustering.. Vol. 33, pp. 4755–4762.

Saatcci, Y., Turner, R., & Rasmussen, C. E. (2010). Gaussian process change point models. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 927–934.

Sadri, A., Ren, Y., & Salim, F. D. (2017). Information gain-based metric for recognizing transitions in human activities. *Pervasive and Mobile Computing*, *38*, 92–109.

Ting, K. M., Liu, Z., Zhang, H., & Zhu, Y. (2022a). A new distributional treatment for time series and an anomaly detection investigation. *Proceedings of the VLDB Endowment*, *15*(11), 2321–2333.

Ting, K. M., Wells, J. R., & Zhu, Y. (2022b). Point-set kernel clustering. *IEEE Transactions on Knowledge and Data Engineering*.

Ting, K. M., Xu, B.-C., Washio, T., & Zhou, Z.-H. (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*, pp. 198–206.

Ting, K. M., Xu, B.-C., Washio, T., & Zhou, Z.-H. (2022). Isolation distributional kernel a new tool for point & group anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*.

Truong, C., Oudre, L., & Vayatis, N. (2020). Selective review of offline change point detection methods. *Signal Processing*, *167*, 107299.

van den Burg, G. J., & Williams, C. K. (2020). An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*.

Vaserstein, L. N. (1969). Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, *5*(3), 64–72.

Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., & Tresp, V. (Eds.), *Advances in Neural Information Processing Systems 13*, pp. 682–688.

Wu, R., & Keogh, E. (2021). Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*.

Yamada, M., Kimura, A., Naya, F., & Sawada, H. (2013). Change-point detection with feature selection in high-dimensional time-series data. In *Twenty-Third International Joint Conference on Artificial Intelligence*, pp. 1827–1833.

Zhang, M., & Sawchuk, A. A. (2012). Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 1036–1043.