Change Point Detection in Time Series Data using Autoencoders with a Time-Invariant Representation

Tim De Ryck*†, Maarten De Vos*‡, Alexander Bertrand*
* *STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics, Department of Electrical Engineering (ESAT), KU Leuven, Belgium*
† *Seminar for Applied Mathematics, Department of Mathematics, ETH Zürich, Switzerland*
‡ *Department of Development and Regeneration, KU Leuven, Belgium*

***Abstract*—Change point detection (CPD) aims to locate abrupt property changes in time series data. Recent CPD methods demonstrated the potential of using deep learning techniques, but often lack the ability to identify more subtle changes in the autocorrelation statistics of the signal and suffer from a high false alarm rate. To address these issues, we employ an autoencoder-based methodology with a novel loss function, through which the used autoencoders learn a partially time-invariant representation that is tailored for CPD. The result is a flexible method that allows the user to indicate whether change points should be sought in the time domain, frequency domain or both. Detectable change points include abrupt changes in the slope, mean, variance, autocorrelation function and frequency spectrum. We demonstrate that our proposed method is consistently highly competitive or superior to baseline methods on diverse simulated and real-life benchmark data sets. Finally, we mitigate the issue of false detection alarms through the use of a postprocessing procedure that combines a matched filter and a newly proposed change point score. We show that this combination drastically improves the performance of our method as well as all baseline methods.**

***Index Terms*—change point detection, time series segmentation, autoencoder, deep learning**

I. Introduction

In the era of big data, where Internet of Things (IoT) devices and other sensors provide endless data streams, the importance of time series analysis techniques can hardly be overestimated. One particular task, that has drawn attention from statistics and data mining communities for decades [1]–[4], is *change point detection* (CPD): the detection of abrupt changes in the temporal evolution of time series data. Change point detection can be a goal in itself or it can be used as a preprocessing tool to segment a time series in homogeneous segments (which can then be further analysed, clustered or classified). Real-life applications of CPD include, but are not limited to, the analysis of climate data [5], financial market data [6], [7], genetic data [8] sensor network data [9], [10] and medical data [11], [12].

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

CPD methods can be categorized according to many different criteria. It is common to make the distinction between online CPD, which provides real-time detections, and retrospective (offline) CPD, which provides more robust detections at the cost of needing more future data. In this paper, we focus on the second category. Many CPD algorithms compare past and future time series intervals by means of a dissimilarity measure. An alarm is issued when the two intervals are sufficiently dissimilar. A first group of methods defines this dissimilarity measure based on the difference in distribution of the two intervals. CUSUM and related methods [4], [13] track changes in the parameter of a chosen distribution, the generalized likelihood ratio (GLR) procedure [14], [15] monitors the likelihood that both intervals are generated from the same distribution, and subspace methods [16], [17] measure the distance between subspaces spanned by the columns of an observability matrix. All these methods however strongly rely on the assumption that the time series data is generated using some parametric probability distribution (CUSUM), autoregressive model (GLR) or state-space model (subspace method). Bayesian online CPD [18] is another notable algorithm that depends on distributional assumptions. Unsurprisingly, the performance of these parametric methods heavily depends on how well the actual data follows the assumed model. Parameter-free alternatives are kernel density estimation [2], [19], [20] and the related density ratio estimation [21], [22]. A more complete overview of CPD methods can be found in e.g. [23]–[26].

Following the successful application of deep learning techniques in anomaly detection, a promising approach for CPD was to base the dissimilarity measure on the distance between features automatically learned by an autoencoder [27]. Main advantages of this approach are the absence of distributional assumptions and the ability of autoencoders to extract complex features from data in a cost-efficient way. There are however also some severe drawbacks. First, there are no guarantees that the distance between consecutive features reflects the actual dissimilarity of the intervals, i.e. features may vary significantly even in the absence of a change point. Second, the correlated nature of time series samples is not adequately leveraged by vanilla autoencoders, which makes it challenging to detect abrupt changes in the frequency domain. This is not uncommon in CPD literature [4], [18], [20], [28]. Some methods explicitly focus on abrupt changes in the spectrum [29], [30], thereby often ignoring changes in the time domain.

Finally, the absence of a postprocessing procedure preceding the detection of peaks in the dissimilarity measure often leads to a high number of false positive detection alarms [31].

Building on [27], we propose a new autoencoder-based CPD method using a partially time-invariant representation (TIRE) that aims to overcome the aforementioned concerns. Our main contributions can be summarized as follows.

- We propose a new CPD framework based on a novel adaptation of the autoencoder with a loss function that promotes time-invariant features. Through our choice of loss function, we aim for the autoencoder to learn a representation that is better suited for CPD. Based on this encoding, we define a dissimilarity measure to detect change points. We evaluate the performance of our algorithm on diverse simulated and real-life benchmark data sets and compare with other dissimilarity-measure-based CPD algorithms.
- Whereas many change point algorithms assume the time series data to consist of independent identically distributed (iid) samples, we explicitly focus on non-iid data. We use the discrete Fourier transform to obtain temporally localized spectral information and propose an approach that combines time-domain and frequency-domain information. When domain knowledge is available, our approach allows the user to only focus on the time or frequency domain.
- Finally, we propose a way of identifying change points from the dissimilarity measure data by applying the notion of topographic prominence [32] to the CPD setting. We emphasize the general importance of postprocessing steps in CPD through numerous experiments.

II. Problem formulation

Let $\mathbf{X}$ be a d-channel time series of length T for which there exist time stamps $0 = T_0 < T_1 < \ldots < T_p = T$ such that every subsequence of the form $(\mathbf{X}[T_k + 1], \ldots, \mathbf{X}[T_{k+1}])$ is a realisation of a discrete time weak-sense stationary stochastic (WSS) process, whereas the union of two such consecutive subsequences is not. The time stamps $T_1, T_2, \ldots$ are referred to as *change points*. The goal of *change point detection* (CPD) is to estimate these change points without any prior knowledge on the number and the locations of the change points [31]. The piecewise WSS assumption is not a strict prerequisite for the algorithm to work, but it does accurately summarize the kind of change points our proposed algorithm will be able to detect. Examples of violations of the WSS conditions, and therefore examples of change points we wish to detect, are changes in mean, variance and autocorrelation. Note that changes in the latter are also reflected in the frequency spectrum through the Wiener-Khinchin theorem [33], [34].

We focus on CPD algorithms that are based on a *dissimilarity measure*. Such methods calculate for every time stamp t the dissimilarity between the windows $(\mathbf{X}[t - N + 1], \ldots, \mathbf{X}[t])$ and $(\mathbf{X}[t + 1], \ldots, \mathbf{X}[t + N])$, where N is a user-defined window size. Our first main goal is to develop a CPD-tailored feature embedding and a corresponding dissimilarity measure $\mathcal{D}_t$, which peaks when the WSS restriction is violated. The nominal approach for identifying change points would then be to determine all local maxima and label each local maximum of which the height exceeds a user-defined detection threshold τ as a change point [27], [35]. However, given a window size N, the width of this peak will theoretically be $2N - 1$ time stamps, making it likely that noise will cause multiple detection alarms for each ground-truth change point [31]. Our second objective is to mitigate the impact of this issue.

III. Autoencoder-based change point detection

A. Preprocessing

Let $\mathbf{X}$ be a d-channel time series of length T, where we denote the i-th channel by $\mathbf{X}^i$. We first divide each channel into windows of size N,

$$\mathbf{x}_t^i = \left[\mathbf{X}^i[t - N + 1], \ldots, \mathbf{X}^i[t]\right]^T \in \mathbb{R}^N. \tag{1}$$

We then combine for every t the corresponding windows of each channel into a single vector,

$$\mathbf{y}_t = \left[(\mathbf{x}_t^1)^T, \ldots, (\mathbf{x}_t^d)^T\right]^T \in \mathbb{R}^{Nd}. \tag{2}$$

Furthermore, we use the discrete Fourier transform (DFT) on each window $\mathbf{x}_t^i$ to obtain temporally localized spectral information. The length of the transformed window is then cropped to a predefined length M. Finally, the modulus of the transformed window is computed. Bundling all these transformations as a single mapping $\mathcal{F} : \mathbb{R}^N \to \mathbb{R}^M$, we obtain the frequency-domain counterpart of $\mathbf{y}_t$:

$$\mathbf{z}_t = \left[\mathcal{F}(\mathbf{x}_t^1)^T, \ldots, \mathcal{F}(\mathbf{x}_t^d)^T\right]^T \in \mathbb{R}^{Md}. \tag{3}$$

B. Feature encoding

Building on [27], we use autoencoders (AEs) to extract features for change point detection from the time-domain (TD) windows $\{\mathbf{y}_t\}_t$. We expand the approach in [27] by also extracting features from the frequency-domain (FD) windows $\{\mathbf{z}_t\}_t$ and through the proposal of a new loss function that explicitly promotes time-invariance of the features in consecutive windows. The latter is a relevant property in order to perform CPD based on a dissimilarity metric.

An autoencoder is a type of artificial neural network that aims to learn a low-dimensional encoding (i.e. features) from a higher-dimensional input by reconstructing the input from the encoding as accurately as possible. It is often used as a dimension reduction technique and can be seen as a non-linear generalization of PCA [36]. In its simplest form, an autoencoder consists of one hidden layer. The encoder maps the input $\mathbf{y}_t \in \mathbb{R}^{Nd}$ (resp. $\mathbf{z}_t$) to its encoded form $\mathbf{h}_t \in \mathbb{R}^h$ as

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{y}_t + \mathbf{b}), \tag{4}$$

where $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector and σ is a non-linear activation function that is applied element-wise. The decoder then maps the encoded representation back to the original input space,

$$\tilde{\mathbf{y}}_t = \sigma'(\mathbf{W}'\mathbf{h}_t + \mathbf{b}'). \tag{5}$$

We choose $\sigma = \sigma'$ to be the hyperbolic tangent function, with as a consequence that each channel of the time series

should be rescaled to the interval $[-1, 1]$. We use individual instead of joint rescaling to ensure that all channels have a comparable magnitude. The goal of the AE is then to minimize the difference between the input and the output, i.e. minimize $\|\mathbf{y}_t - \tilde{\mathbf{y}}_t\|$, by optimizing the choice of $\mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}'$. In [27], the learned features $\mathbf{h}_t$ are then used for CPD by measuring the dissimilarity between consecutive feature vectors ($\mathbf{h}_t$ vs. $\mathbf{h}_{t-1}$). However, the learned features $\mathbf{h}_t$ will then unavoidably also contain information that is not relevant for CPD (e.g. phase shift or noise information), which may generate large dissimilarities even when there is no actual change point.

We try to remedy this by introducing the notions of *time-invariant* and *instantaneous features*. The idea is that features learned from consecutive windows are only useful for CPD when they are approximately equal to each other in the absence of a change point (e.g. mean, amplitude and frequency should not change much within a WSS segment). We will refer to them as *time-invariant* features as they are aimed to be invariant over time within a WSS segment. All other information that is needed for a good reconstruction, but that may differ for consecutive windows, is aimed to be encoded in *instantaneous* features. This then gives

$$\mathbf{h}_t = \left[(\mathbf{s}_t)^T, (\mathbf{u}_t)^T\right]^T, \tag{6}$$

where $\mathbf{s}_t \in \mathbb{R}^s$ are the time-invariant features and $\mathbf{u}_t \in \mathbb{R}^{h-s}$ are the instantaneous features. To obtain both a good reconstruction and time-invariant features, we propose to minimize the loss function

$$\sum_t \left(\|\mathbf{y}_t - \tilde{\mathbf{y}}_t\|_2 + \lambda \|\mathbf{s}_t - \mathbf{s}_{t-1}\|_2\right), \tag{7}$$

where $\lambda > 0$ control the amount of regularization of the time-invariant features. Here we make the implicit assumption that the number of terms in (7) that correspond to a window containing a change point is very small compared to T.

It is very uncommon in machine learning to directly minimize the loss function (7), i.e. take all t into account for every step of gradient descent. To improve convergence, it is advisable to first randomly partition all time stamps t over J smaller *mini-batches* $\mathcal{T}_j$ [37]. The mini-batch stochastic gradient descent (SGD) version of minimizing (7) would then consist of updating the network parameters by calculating the gradient of

$$\sum_{t \in \mathcal{T}_j} \left(\|\mathbf{y}_t - \tilde{\mathbf{y}}_t\|_2 + \lambda \|\mathbf{s}_t - \mathbf{s}_{t-1}\|_2\right) \tag{8}$$

for some j, followed by performing one gradient descent step and repeating this for all other mini-batches. Note that formulation (8) would require to use time stamps from other batches, i.e. $t \in \mathcal{T}_j$ does not generally imply that $t - 1 \in \mathcal{T}_j$. However, we choose to generalize (8), and minimize the following loss function for each mini-batch,

$$\sum_{t \in \mathcal{T}_j} \left(\|\mathbf{y}_t - \tilde{\mathbf{y}}_t\|_2 + \frac{\lambda}{K} \sum_{k=0}^{K-1} \|\mathbf{s}_{t-k} - \mathbf{s}_{t-k-1}\|_2\right), \tag{9}$$

where $K \in \mathbb{N}$. For $K = 1$ this equation reduces to (8). For $K > 1$, this approach has the advantage that now

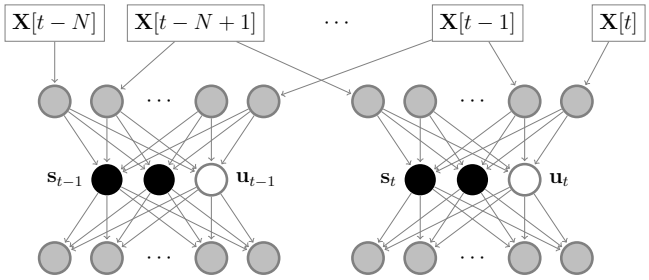

Fig. 1. Visualization of time-invariant feature encoding for $K = 1$. The TD autoencoder is shown two times, once with input $\mathbf{y}_{t-1}$ and once with input $\mathbf{y}_t$. The corresponding time-invariant features $\mathbf{s}_{t-1}$ and $\mathbf{s}_t$ are forced to be approximately equal because of the chosen loss function (9). Frequency-domain time-invariant features are obtained analogously.

$K + 1$ consecutive features are jointly and simultaneously considered during the computation of the gradient, resulting in an additional smoothing effect of the stochastic gradient in the direction of the minimization of the penalty term in (7). Thereby further promoting the aimed time invariance of the features $\mathbf{s}_t$. It may help to think of (9) as $K + 1$ parallel autoencoders with identical weights and biases, where the k-th autoencoder receives $\mathbf{y}_{t+k-K-1}$ as input and where a subset of the latent variables (i.e. the time-invariant features) of the parallel autoencoders are forced to be close together to obtain a partially time-invariant representation (Figure 1). Note that even though the difference in formulation between (7) and (9) impacts the training of the autoencoder, the resulting loss functions are essentially the same when summing over all t.

To avoid that the autoencoder encodes all information in the unregularized instantaneous features, the number of instantaneous features should be taken as small as possible. Depending on the data, one might add additional regularization terms to the loss function or use a more advanced type of autoencoder (e.g. weight regularized, deep/stacked, tied-weights, variational, recurrent autoencoder). In an entirely similar fashion, we train a second autoencoder on $\{\mathbf{z}_t\}_t$ with a similar loss function to obtain frequency-domain time-invariant features. We will use the superscripts TD and FD to distinguish between parameters and features corresponding to the time and frequency domain, respectively.

C. Postprocessing and peak detection

In this section we first describe how to construct a dissimilarity measure that complies with the needs formulated in Section II based on the time-invariant features from the previous section. Next, we discuss multiple methods to suppress the number of false positives when determining the detection alarms.

1) Postprocessing: We first combine the TD and FD time-invariant features into a single time-invariant features vector,

$$\mathbf{s}_t = \left[\alpha \cdot (\mathbf{s}_t^{\text{TD}})^T, \beta \cdot (\mathbf{s}_t^{\text{FD}})^T\right]^T, \tag{10}$$

where $\alpha, \beta > 0$ are parameters that control the relative contribution of the TD and FD time-invariant features. Next, we use a zero-delay weighted moving average filter to smoothen the time-invariant features, as small fluctuations in the features

will affect the performance of the method. The moving average filtering operation can be described as follows,

$$\tilde{\mathbf{s}}_t[i] = \sum_{k=-N+1}^{N-1} \mathbf{v}[N-k] \cdot \mathbf{s}_{t+k}[i], \tag{11}$$

with $\mathbf{v}[k] = \mathbf{v}[2N-k] \triangleq k/N^2$ for $1 \leq k \leq N$, where N is the window size as defined in (1), resulting in a triangular shaped weighting window. We use edge value padding in order for the equation to be defined for all t. We then propose the following definition for the dissimilarity measure $\mathcal{D}$:

$$\mathcal{D}_t = \|\tilde{\mathbf{s}}_t - \tilde{\mathbf{s}}_{t+N}\|_2, \tag{12}$$

where N is the window size as defined in (1). In some applications, domain-specific knowledge might suggest that only TD (resp. FD) information is relevant for CPD. This expert knowledge can be incorporated in the dissimilarity measure by setting $\alpha = 1$ and $\beta = 0$ (resp. $\alpha = 0$ and $\beta = 1$) in (10). We denote the obtained dissimilarity measure by $\mathcal{D}_t^{\text{TD}}$ (resp. $\mathcal{D}_t^{\text{FD}}$). Using $\mathcal{D}_t^{\text{TD}}$ and $\mathcal{D}_t^{\text{FD}}$, we can also set α and β automatically in such a way that the TD and FD time-invariant features contribute in a comparable fashion to $\mathcal{D}_t$. We let

$$\alpha = Q(\{\mathcal{D}_t^{\text{FD}}\}_t, 0.95) \quad \text{and} \quad \beta = Q(\{\mathcal{D}_t^{\text{TD}}\}_t, 0.95), \tag{13}$$

where Q is the quantile function, i.e. for a set of real numbers A and $0 < p \leq 1$ it holds that $Q(A, p)$ is the smallest number such that $p \cdot 100\%$ of the elements of the set A are smaller than $Q(A, p)$. We use the 95-percentile as a measure of the heights of the peaks in the dissimilarity scores, where outliers are ignored. By setting α and β in (10) according to (13), the peaks in $\{\mathcal{D}_t^{\text{FD}}\}_t$ and $\{\mathcal{D}_t^{\text{TD}}\}_t$ contribute approximately equally to $\{\mathcal{D}_t\}_t$. As all learned features lie in the interval $[-1, 1]$, the robustness of using a quantile-based fusion approach is guaranteed.

2) Peak detection: If the time-invariant features are indeed similar across successive windows within a WSS segment, the dissimilarity measure $\mathcal{D}_t$, as defined in (12), will peak at or near a change point. Determining reasonable detection alarms from these peaks is an often neglected task in current literature. In some cases, the problem is avoided by focusing on time series containing only one change point [35]. In other cases all local maxima of the dissimilarity measure are considered to be detection alarms [27], leading to unreasonably many false positives. Liu et al. [22] propose to reduce the number of false positives by deleting detections that are too close to the previous detection. As their method might also delete correct detections, it is clearly not optimal. Recently, the use of a matched filter was investigated as a way to improve detection and localization of change points [28], [31]. It is however difficult to automatically select a representative peak to base the matched filter on [28], nor is it possible to unambiguously derive an asymptotically matched filter [31] for our dissimilarity measure. We therefore propose to reuse the impulse response $\mathbf{v}$ from the moving average filtering (11) as it is has a comparable effect to that of a matched filter, as a consequence of its width and shape. This then leads to

$$\tilde{\mathcal{D}}_t = \sum_{k=-N+1}^{N-1} \mathbf{v}[N-k] \cdot \mathcal{D}_{t+k}. \tag{14}$$

The detection alarms then correspond to all local maxima of the series $(\tilde{\mathcal{D}}_N, \tilde{\mathcal{D}}_{N+1}, \ldots, \tilde{\mathcal{D}}_{T-N})$ [28], [31].

Aiming to further improve detection accuracy, we propose to use a different, parameter-free approach for peak detection. In topography, the *prominence* of a peak is the minimum height that one needs to descend in order to be able to ascend to a higher peak [32]. The idea is that even though every peak in the dissimilarity measure might consist of multiple local maxima that all have a large height, only one of these maxima will have a large prominence. This measure has previously been successfully applied in the analysis of population data [38], super-resolution microscopy data [39] and neural signals [40]. Given that $\mathcal{D}_t$ is a local maximum, we first define the two closest time stamps left and right of t for which the dissimilarity measure is larger than $\mathcal{D}_t$, and denote them by t_L and t_R respectively, i.e.,

$$t_L = \max\left\{\sup\{t^* \mid \mathcal{D}_{t^*} > \mathcal{D}_t \text{ and } t^* < t\}, N\right\}, \tag{15}$$

$$t_R = \min\{\inf\{t^* \mid \mathcal{D}_{t^*} > \mathcal{D}_t \text{ and } t^* > t\}, T-N\}, \tag{16}$$

where the $\max$ and $\min$ operators ensure that t_L and t_R stay at a distance N from the boundaries of the time series. We then define the prominence $\mathcal{P}(\mathcal{D}_t)$ of local maximum $\mathcal{D}_t$ by

$$\mathcal{P}(\mathcal{D}_t) = \mathcal{D}_t - \max\left\{\min_{t_L < t^* < t} \mathcal{D}_{t^*}, \min_{t < t^* < t_R} \mathcal{D}_{t^*}\right\}. \tag{17}$$

If $\mathcal{D}_t$ is not a local maximum we set $\mathcal{P}(\mathcal{D}_t) = 0$ by definition. We propose to combine the matched filter (14) and the prominence measure (17), i.e. by calculating the prominences for $\{\tilde{\mathcal{D}}\}_t$ instead of $\{\mathcal{D}\}_t$. A change point is then detected if the prominence $\mathcal{P}(\tilde{\mathcal{D}}_t)$ is above a predefined threshold τ.

D. Summary: the TIRE method

Finally, we summarize all the steps of the proposed Time-Invariant REpresentation (TIRE) change point detection method. If only time-domain or frequency-domain information is used, we will refer to the method using the acronym TIRE-TD or TIRE-FD, respectively.

1) Construct time-domain windows $\{\mathbf{y}_t\}_t$ (2) and frequency-domain windows $\{\mathbf{z}_t\}_t$ (3) from a time series $\mathbf{X}$.
2) Using these windows as training data sets, train two autoencoders by minimizing loss function (9).
3) Use (13) to determine α and β or set one of them to zero based on domain knowledge. Construct the combined time-invariant features according to (10).
4) Smoothen the time-invariant features according to (11).
5) Calculate the dissimilarity measures for all t using (12).
6) Apply a matched filter on the dissimilarity measures following (14) and compute the prominence of all local maxima using (17).
7) If the prominence (17) of a local maximum is higher than some user-defined detection threshold τ, a change point has been detected.

An implementation of our TIRE methods has been made available at https://github.com/deryckt/TIRE.

IV. Experiments

A. Evaluation measure

In our setting, the goal of a CPD algorithm is to identify the location of change points as accurately as possible. Given a toleration distance δ we say that a ground-truth change point a is correctly detected by a detection alarm b if the following three conditions are satisfied [27]:

1) No other ground-truth change point is closer to b than a.
2) The time distance between a and b is smaller than the toleration distance, i.e. $|a - b| \leq \delta$.
3) Every detection alarm can only contribute to the correct detection of at most one ground-truth change point.

To evaluate the performance of our method, we will construct receiver operating characteristic (ROC) curves and use the area under this curve (AUC) as a performance metric, as is common practice. Following [20]–[22], [27], we define the true positive rate (TPR) and false positive rate (FPR) of our detection algorithm as

$$\text{TPR} = \frac{N_{\text{CR}}}{N_{\text{GT}}} \quad \text{and} \quad \text{FPR} = \frac{N_{\text{AL}} - N_{\text{CR}}}{N_{\text{AL}}}, \tag{18}$$

where N_{GT} denotes the number of ground-truth change points, N_{AL} denotes the number of all detection alarms by the algorithm and N_{CR} is the number of times a ground-truth change point is correctly detected. We obtain the ROC curve by varying the detection threshold τ. Unlike in the binary classification setting, the ROC curve is not necessarily monotonously increasing, as the FPR does not need to be a monotonous function of τ. Nevertheless, it still holds that $0 \leq \text{AUC} \leq 1$. Moreover, note that a TPR of 1.0 can be obtained by setting the detection threshold to zero $\tau = 0$ (i.e. all time stamps are detection alarms), though the FPR will always be strictly smaller than 1.0 for $\tau = 0$ when at least one change point is present. We therefore extend the ROC curve by manually adding the point $(\text{FPR}, \text{TPR}) = (1.0, 1.0)$. This ensures that a perfect performance corresponds to an AUC of 1.

B. Data sets

We demonstrate the performance of our method on four simulated and three real-life benchmark data sets, of which six are typical benchmark data sets in CPD literature. A summary of their properties can be found in Table I.

1) Simulated data: We consider the one-dimensional autoregressive (AR) model $y(t) = a_1 y(t-1) + a_2 y(t-2) + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$ and $y(1) = y(2) = 0$. We generate 50 random change points t_n with $t_0 = 0$, $t_n = t_{n-1} + \lfloor \tau_n \rfloor$ and $\tau_n \sim \mathcal{N}(100, 10)$. Following the parameter choices of [20]–[22], [41], we create the following data sets, each consisting of ten randomly generated time series.

Jumping mean (JM). For this data set, let $a_1 = 0.6$, $a_2 = -0.5$ and $\sigma_t = 1.5$. We set the noise mean as

$$\mu_t = \begin{cases} 0 & 1 \leq t \leq t_1 \\ \mu_{t_{n-1}} + n/16 & t_{n-1} + 1 \leq t \leq t_n. \end{cases} \tag{19}$$

Scaling variance (SV). For this data set, let $a_1 = 0.6$, $a_2 = -0.5$ and $\mu_t = 0$. We set the noise standard deviation as

$$\sigma_t = \begin{cases} 1 & t_{n-1} + 1 \leq t \leq t_n \text{ and } n \text{ odd} \\ \ln(e + n/4) & t_{n-1} + 1 \leq t \leq t_n \text{ and } n \text{ even}. \end{cases} \tag{20}$$

Changing coefficients (CC). We set $a_2 = 0$, $\mu_t = 0$ and $\sigma_t = 1.5$. To take the burn-in time into account, we set $\tau_n \sim \mathcal{N}(1000, 100)$. For every segment, the coefficient a_1 is alternatively sampled from $\mathcal{U}([0, 0.5])$ and $\mathcal{U}([0.8, 0.95])$, leading to clear differences in autocorrelation and frequency content between consecutive segments.

Gaussian mixtures (GM). Here we abandon the AR model and instead simulate a piecewise iid sequence alternatively sampled between the Gaussian mixtures $0.5\mathcal{N}(-1, 0.5^2) + 0.5\mathcal{N}(1, 0.5^2)$ and $0.8\mathcal{N}(-1, 1.0^2) + 0.2\mathcal{N}(1, 0.1^2)$. Change points are generated using the same mechanism as for JM and SV.

2) Real-life data sets: **Bee dance** [42] is an often used data set to evaluate CPD algorithms [20], [25], [31], [43], [44]. It consists of six three-dimensional time series of the bees position (location in 2D plane and angle differences) while it performs a three-stage waggle dance, which is of interest to ethnologists.

HASC-2011 is a subset of the HASC Challenge 2011 dataset [45], which provides human activity data from portable three-axis accelerometers. The six activities carried out are staying still, walking, jogging, skipping, taking the stairs up or down. Following respectively [28] and [22], we use the data from person 671 and convert the data to a 1D time series by taking the l^2-norm of the three-dimensional samples. Human activity recognition data is commonly used in CPD literature [20]–[22], [28], [31], [35], [45].

Well log [46] consists of nuclear magnetic resonance measurements taken from a drill while drilling a well. Changes in the mean of the time series correspond to changes in rock stratification, outliers should be ignored [47]. Other results on this data set in the context of CPD evaluation include [18], [25], [44], [46]–[48].

TABLE I
Overview of data sets. For data sets consisting of multiple time series, mean and standard deviation are reported. Q10, Q50 and Q90 denote the 10%, 50% and 90% quantile, resp.

Data set	Length	#series	#CPs	CP distances		
				Q10	Q50	Q90
JM, SV, GM	4900 ± 22	10	48	96	100	104
CC	49000 ± 70	10	48	987	1000	1013
Bee dance	827 ± 202	6	20 ± 4	28	39	56
HASC-2011	39397	1	39	69	427	2509
Well log	4050	1	9	55	170	390

C. Parameter settings and baseline methods

For TIRE, we report the results for two different parameter settings. Parameter setting *a* corresponds to the case without instantaneous features: in both time and frequency domain the autoencoder learns only 1 (time-invariant) feature (i.e.

$h^{\text{TD}} = s^{\text{TD}} = h^{\text{FD}} = s^{\text{FD}} = 1$). Furthermore we set $K = 2$, $\lambda^{\text{TD}} = 1$ and $\lambda^{\text{FD}} = 1$. Parameter setting b corresponds to the case with 1 instantaneous and 2 time-invariant features in the time domain (i.e. $h^{\text{TD}} = 3$, $s^{\text{TD}} = 2$) and furthermore we set $h^{\text{FD}} = s^{\text{FD}} = 1$, $K = 2$, $\lambda^{\text{TD}} = 1$ and $\lambda^{\text{FD}} = 1$. For TIRE-TD we set $\alpha = 1$ and $\beta = 0$ in (10), and vice versa for TIRE-FD. For the combined approach, we set α and β following (13). We train all networks for 200 epochs using the Adam optimizer [49] with default settings. For both parameter settings, we choose window sizes and toleration distances based on domain knowledge and sampling frequency. We set $N = 20$ and $\delta = 15$ for JM, SC and GM; $N = 200$ and $\delta = 150$ for CC; $N = 10$ and $\delta = 15$ for bee dance; $N = 100$ and $\delta = 300$ for HASC-2011 and $N = 75$ and $\delta = 50$ for well log. The influence of these parameter settings will be discussed in Section V-E. In terms of postprocessing, we use a matched filter and calculate our proposed prominence score (cf. Section III). The advantageous effect of this postprocessing stage is analyzed in Section V-C. In order to obtain a fair comparison, we also apply these postprocessing steps to all undermentioned baseline methods which do not explicitly define such a procedure.

The first baseline method we use is the **generalized likelihood ratio** (GLR) procedure [14], [15], which has been shown to have a good performance for detecting changes in the autocorrelation function or the frequency spectrum [50]. A conceptually similar method is described in [51]. We use a sliding window approach, where an AR(2)-model is fit on every two neighbouring windows as well as their union. A generalized log-likelihood ratio is used as dissimilarity measure. For a fair comparison, we use the same window sizes and postprocessing steps as for TIRE.

Second, we consider a density-ratio estimation method called **relative unconstrained least-squares importance fitting** (RuLSIF) that has been applied to CPD [22]. Like with the closely related uLSIF [21], the idea is to estimate and compare the density ratio of two neighbouring windows instead of the individual densities. Because the validation data sets in [22] largely overlap with ours, we adopt the same parameter choices and postprocessing steps as described in the original paper.

Next, **kernel learning CPD** (KL-CPD) [20] is a recently proposed kernel learning framework for time series CPD that optimizes a lower bound of test power via an auxiliary generative model. Features are learned using a recurrent neural network and the dissimilarity measure is based on the maximum mean discrepancy. Given the large overlap in used benchmark data sets, we use the original default parameter settings in [20] without adaptation (e.g. window size of 25). We train the networks for 200 epochs, as longer training did not result in improved results. For a fair comparison, we use the same postprocessing steps as for TIRE as none were proposed in [20].

Finally, we compare with the **autoencoder-based breakpoint detection procedure** (ABD) [27]. ABD only uses time-domain information and does not include any regularization to promote time-invariant features. We set parameters using the parameter guidelines in the original paper. This leads to a window size of 96 for JM, SV and GM; 995 for CC; 26 for bee dance; 158 for HASC-2011 and 155 for well log.

V. Results

A. Performance results

In Table II, the performances of all versions of TIRE and the baseline methods are listed. For all data sets, we report the mean AUC and its standard error. All data sets, methods and abbreviations are described in Section IV-C. The highest mean AUC for each data set can be found in bold. In the following, we discuss some important observations.

The GLR procedure gives very good results on the simulated data sets, but its performance degrades on the real-life data sets. This confirms the common observation that the performance of model-based CPD procedures heavily relies on how well the actual data can be described using the chosen model. In this case, both the simulated data and the GLR procedure are based on a second-order autoregressive model, which is why GLR performs well on this data. RuLSIF and KL-CPD do not perform well on data sets in which the change points manifest themselves in the frequency domain, since they do not leverage the sequential nature of the time series data, i.e. they assume the data samples to be iid. Note that AUC values for KL-CPD differ from those in [20] as CPD is there interpreted as a binary classification problem. Next, ABD generally does not give good results, which can by explained by ABD's inability to detect changes in the frequency domain and the often noisy features (cf. Figure 2). In addition, ABD's *normalized* dissimilarity measure (eq. (10) in [27]), given by,

$$\mathcal{D}_t^{\text{ABD}} = \|\mathbf{h}_t - \mathbf{h}_{t+N}\|_2 / \sqrt{\|\mathbf{h}_t\|_2 \cdot \|\mathbf{h}_{t+N}\|_2}, \qquad (21)$$

where $\mathbf{h}_t$ is the vector of learned time-domain features, is not invariant to a shift of the features (i.e. adding a constant to all features); it even diverges when the norm of one of the features vanishes, which is not reasonable.

For all data sets and both parameter settings *a* and *b*, either TIRE-TD or TIRE-FD outperforms (almost) all other baseline methods or has an AUC higher than 0.90. In many real-life cases, it is a priori clear whether TD (e.g. well log) or FD (e.g. HAR data, audio, ...) information should be used. Moreover, our framework for combining the time-invariant features from the time and frequency domain still gives consistently good results even when in one of the two domains no change point information is present. This means that the combined TD-FD approach can always be selected as a safe choice when it is unclear in which domain the change points mainly manifest themselves. Finally, the different parameter settings seem have no significant influence on the performance of TIRE. The sensitivity of the proposed method to parameter choices will be further discussed in Section V-E.

B. Insight in encoded features and reconstruction

To gain insight into the working of the TIRE method, we investigate how the (partially) time-invariant representation and the corresponding reconstructions look like. We do this by conducting a case study on the jumping mean and bee dance data set.

TABLE II
COMPARISON OF THE AUC OF THE PROPOSED TIME-INVARIANT REPRESENTATION CPD METHODS (TIRE) WITH BASELINE METHODS.

	Mean	**Variance**	**Coefficient**	**Gaussian**	**Bee dance**	**HASC-2011**	**Well log**	**Average**
GLR [14], [15]	0.73 ± 0.02	0.81 ± 0.02	$\mathbf{1.00} \pm 0.01$	$\mathbf{0.989} \pm 0.004$	0.55 ± 0.06	0.6431	0.2109	0.71 ± 0.01
RuLSIF [22]	0.708 ± 0.008	0.65 ± 0.02	0.36 ± 0.02	0.874 ± 0.007	0.47 ± 0.06	0.3162	0.798	0.597 ± 0.009
KL-CPD [20]	0.872 ± 0.007	0.23 ± 0.02	0.11 ± 0.01	0.84 ± 0.07	0.56 ± 0.07	0.343	0.4247	0.48 ± 0.01
ABD [27]	0.22 ± 0.02	0.17 ± 0.02	0.08 ± 0.02	0.18 ± 0.02	0.20 ± 0.04	0.2487	0.477	0.224 ± 0.008
TIRE-TD-*a*	0.86 ± 0.01	0.25 ± 0.01	0.26 ± 0.01	0.958 ± 0.009	0.36 ± 0.05	0.4166	0.8002	0.558 ± 0.007
TIRE-FD-*a*	0.86 ± 0.01	$\mathbf{0.85} \pm 0.01$	0.96 ± 0.01	0.83 ± 0.04	$\mathbf{0.70} \pm 0.10$	**0.6504**	0.6278	$\mathbf{0.78} \pm 0.02$
TIRE-*a*	0.86 ± 0.01	$\mathbf{0.85} \pm 0.01$	0.74 ± 0.05	0.92 ± 0.02	0.65 ± 0.09	0.6172	0.7656	0.77 ± 0.02
TIRE-TD-*b*	$\mathbf{0.882} \pm 0.009$	0.26 ± 0.02	0.26 ± 0.02	0.965 ± 0.006	0.42 ± 0.06	0.4284	**0.8151**	0.58 ± 0.01
TIRE-FD-*b*	0.86 ± 0.01	0.84 ± 0.02	0.95 ± 0.02	0.74 ± 0.03	0.69 ± 0.10	0.6261	0.200	0.70 ± 0.02
TIRE-*b*	0.877 ± 0.009	0.83 ± 0.02	0.76 ± 0.05	0.89 ± 0.02	0.60 ± 0.09	0.6258	0.8134	0.77 ± 0.01

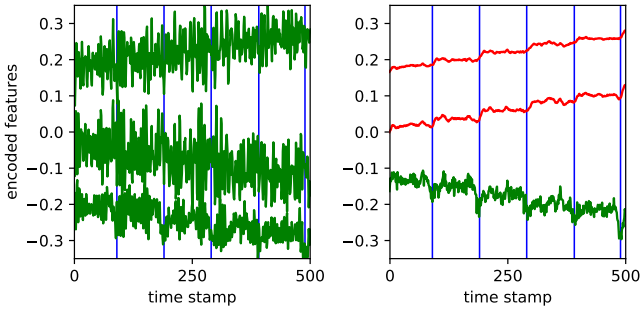

Fig. 2. Example of the three-dimensional learned representation on a part of the jumping mean data set for ABD (left) and TIRE-TD (right) with two time-invariant features (in red) and one instantaneous feature (in green). The features were vertically shifted (but not rescaled) for clarity. Blue vertical lines indicate the locations of ground-truth change points.

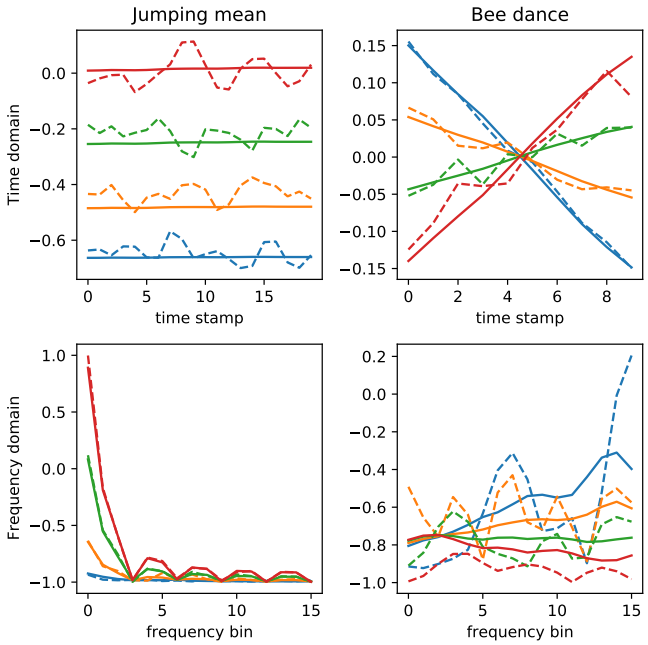

Fig. 3. Examples of time-domain and frequency-domain windows (dashed lines) and their reconstructions by the autoencoder used in our proposed method (full lines). In the jumping mean data set, the change points consist of abrupt changes in mean. For bee dance, the goal is to detect abrupt changes in slope (upper right) and amplitude (lower right).

First, we demonstrate the effect of our proposed penalty in the autoencoder loss function (9). In Figure 2 we show the non-smoothed encoded features (i.e. without applying (11)) for a part of the jumping mean data set for both ABD and TIRE-TD. For both methods, we use three features, of which two are time-invariant in the case of TIRE. Other parameter settings are as in parameter setting *b* of Section IV-C. Whereas the features learned by ABD are very variable and noisy, the time-invariant features of TIRE-TD are approximately constant within each segment. For TIRE, the only significant variations in the features are near the ground-truth change points. These observations match exactly with the intention of our proposed loss function.

Second, we conduct a case study on the reconstruction of both TD and FD windows. Since the number of features we propose to use is very small, these reconstructions might be lossy and deviate from the original windows. We train TD and FD autoencoders with only one (time-invariant) feature following parameter setting *a* (cf. Section IV-C) for jumping mean and bee dance data. We select four distinct windows and their reconstruction for each data set. The results are shown in Figure 3 in different colours. In case of the jumping mean data set, the autoencoder unsurprisingly reconstructs the mean of each interval, ignoring all noise. In the frequency domain, the mean manifests itself in the DC component (first frequency bin). The values of most other frequency bins seem to be encoded in the weights and biases. Next, we consider the bee dance data set. In the time domain, we use one location coordinate of the bee. As the bee moves back and forth in its waggle dance, the location coordinate resembles a triangular wave. The autoencoder can track the bees location through the variation in the slope of the location coordinate windows. The reconstruction in Figure 3 indeed shows approximately straight lines with varying slope. In the frequency domain, we only consider the angle of the head of the bee in this case study. As the bee shakes its head in some parts of the waggle dance, the goal is to pick up the presence of high-frequency oscillations. Indeed, the reconstruction only varies notably in the bins corresponding to higher frequencies. As we use only one latent variable, the decoded reconstruction does not fully capture all variations in the frequency spectrum, yet it captures the slope of the upward trend towards higher frequencies. We conclude that autoencoders can automatically identify and construct CPD-relevant features, in contrast to

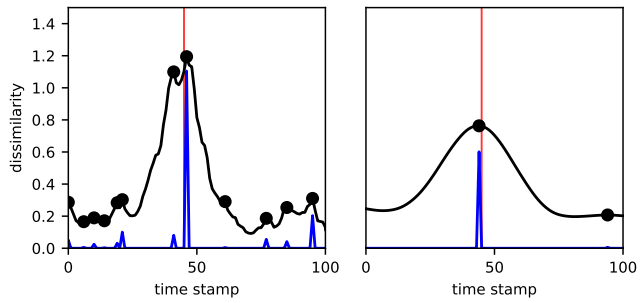

Fig. 4. Example of the peak in our proposed dissimilarity measure (black line) near a ground-truth change point (red vertical line) both without (left) and with (right) the use of a matched filter. Black dots correspond to local maxima (i.e. detection alarms), our proposed prominence measure is shown in blue. The matched filter drastically reduces the number of false positive detection alarms, whereas the prominence measure makes sure that there is only one detection alarm with a large change point score. The example was generated using KL-CPD on the Gaussian mixture data set.

TABLE III
COMPARISON OF THE AUC OF DIFFERENT POSTPROCESSING TECHNIQUES ON DISSIMILARITY-MEASURE-BASED CPD METHODS.

	Height	**Height+MF**	**Prominence**	**Prom.+MF**
GLR	0.42 ± 0.05	0.67 ± 0.04	0.58 ± 0.04	$\mathbf{0.71} \pm 0.04$
RuLSIF	0.37 ± 0.07	0.63 ± 0.05	0.60 ± 0.07	$\mathbf{0.64} \pm 0.05$
KL-CPD	0.28 ± 0.10	0.46 ± 0.08	0.44 ± 0.10	$\mathbf{0.48} \pm 0.07$
TIRE	0.40 ± 0.08	0.67 ± 0.10	0.56 ± 0.08	$\mathbf{0.79} \pm 0.07$

CPD methods based on parametric models where the relevant parameters need to be chosen in advance.

C. Importance of postprocessing

In Section III, we conjectured the importance of suitable postprocessing steps to mitigate the effect of false positive detection alarms. An example of the effect of our postprocessing steps can be found in Figure 4. The use of the prominence as a change point score allows us to automatically retain only one significant detection alarm per peak, whereas a height-based dissimilarity score would lead to a false positive detection alarm even if the detection threshold is set high. Furthermore, the matched filter automatically removes most false positive detections. The use of our proposed prominence score then ensures that the remaining false positive detections have a negligible change point score.

Next, we quantitatively compare peak height and peak prominence (17) as change point score and investigate the effect of applying a matched filter (14). We report the average and standard deviation of the AUC on all seven data sets for the GLR procedure, RuLSIF, KL-CPD and TIRE in Table III. Both the matched filter and the use of the peak prominence result in an increase in the average AUC, with best results for when both postprocessing techniques are combined. Most notably, our proposed postprocessing approach almost leads to a doubling of the average AUC compared to naive peak detection for all methods.

D. Run time

We compare the run times of the different methods on the jumping mean data set by reporting the mean and standard deviation of run times under 10 random seeds. The GLR procedure takes (6.6 ± 0.4)s, RuLSIF needs (69.6 ± 1.5)s, KL-CPD needs (390 ± 5)s for 200 epochs and TIRE takes (32.5 ± 0.2)s for 200 epochs. The run times of all methods scale linearly with the length of the time series. Unsurprisingly, the very simple GLR procedure is by far the fastest method. KL-CPD, which involves the training of a generative adversarial network and a recurrent neural network, is the slowest. Comparing the run times for 200 epochs, we see that TIRE is faster than KL-CPD. Note that the comparison between TIRE and KL-CPD is difficult, as both are iterative methods and convergence rates may differ. In the code accompanying [20], a stop criterion for KL-CPD is provided, but this criterion was never satisfied sooner than 200 epochs on the used data sets. We conclude that TIRE has a very reasonable run time compared to other methods, albeit not the best.

E. Sensitivity analysis

We investigate to which extent the performance of the proposed method depends on the parameters chosen in Section IV-C. Ideally, each parameter can either be set following some general guidelines, or the method should not be sensitive to the exact parameter value.

First, we examine how the performance depends on the chosen window size. It is clear that a constant window size would in general be an unreasonable demand: when a time series is down- or upsampled, the window size should change accordingly. Some attempts to provide guidelines on how to choose a window size have been made [27], but these often give rise to unreasonable choices and poor performance (see ABD in Table II). Moreover, one can even argue that a good window size is inherently dependent on the interpretation and goals of the practitioner, and can not be deduced from the data alone. For example, this would be the case for a superposition of two CC time series (cf. Section IV-B) with frequencies at two distinct scales, of which only one is of interest to the practitioner. Following amongst others [31], we therefore advise to set the window size based on domain knowledge (cf. Section IV-C). To inspect the sensitivity of TIRE to these choices, we show in Figure 5 the mean AUC and its standard error for all seven data sets for window sizes that are 0.25, $1/2\sqrt{2}$, 0.5, $1/\sqrt{2}$, 1, $\sqrt{2}$, 2, $2\sqrt{2}$ and 4 times the domain-knowledge-based window size as defined in Section IV-C. Furthermore we let again $K = 2$, $\lambda^{\text{TD}} = 1$ and $\lambda^{\text{FD}} = 1$. The larger standard error for the bee dance data set in Figure 5 is primarily caused by the large variation in difficulty between the different time series, and not by the method. For most data sets only limited variations in AUC are present in the interval $[0.5, 2]$, such that a small to moderate change in window size would not affect the positioning of the performance of the proposed TIRE method compared to the results of the considered baseline methods. For the changing coefficients (CC) data set and the well log data set, the variations in AUC are more substantial. The AUC for CC increases steadily as the

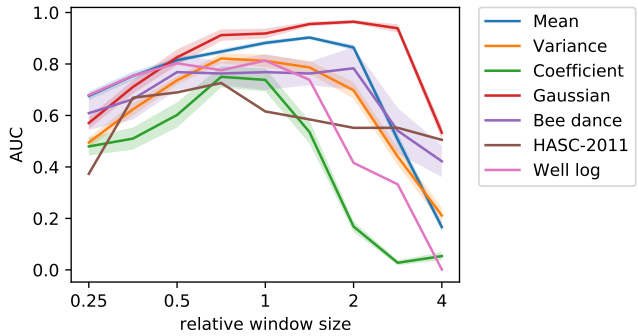

Fig. 5. Influence of the window size to the performance of TIRE. We report the mean AUC and its standard error for window sizes ranging between one quarter to four times the window size chosen in Section IV-C.

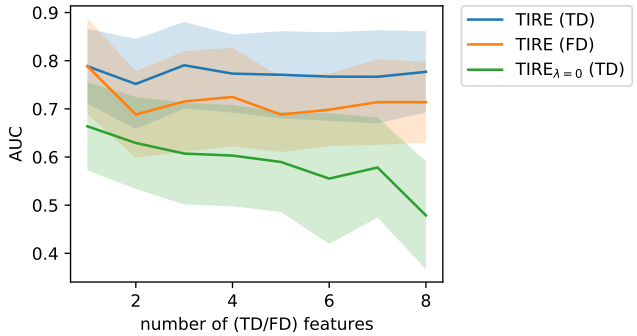

Fig. 6. Sensitivity of performance of TIRE to the total number of TD features h^{TD} and FD features h^{FD} of the used autoencoder. The average AUC over all data sets and its standard deviation is shown. We also compare to the dependence of TIRE$_{\lambda=0}$ on the latent dimension. Whereas TIRE (with $\lambda = 1$) seems on average robust to the number of TD features, the AUC for TIRE$_{\lambda=0}$ decreases.

window size grows since the DFT can better capture the long-range dependences in the data set, but also decreases sharply when the window size is large compared to the distance between the change points. In the well log case, the difficulty is that some change points are very close to each other. When the window size grows large, two nearby peaks in the dissimilarity measure will not be resolved anymore. In this case, the use of a matched filter is thus even disadvantageous. This also explains why the AUC decreases sharply for all data sets when an unreasonably large window size is chosen.

Second, we investigate the influence of the latent dimension of the used autoencoder. We let the total number of time-domain features h^{TD} vary from 1 to 10 and set the number of time-invariant features to $s^{\mathrm{TD}} = \max\{h^{\mathrm{TD}}-1, 1\}$. Furthermore we let $s^{\mathrm{FD}} = h^{\mathrm{FD}} = 1$, $K = 2$, $\lambda^{\mathrm{TD}} = 1$ and $\lambda^{\mathrm{FD}} = 1$ (cf. parameter settings *a* and *b*). We use at most one instantaneous feature to avoid that the autoencoder would leak valuable CPD-relevant information into the instantaneous features (cf. Section III). We also let the number of frequency-domain features vary analogously and investigate the advantage of using time-invariant features. We do the latter by comparing to TIRE$_{\lambda=0}$, a version of TIRE with $\lambda = 0$ in the loss function (9) (i.e. no time-invariant features) and without the smoothing as in (11), as this is not necessarily a meaningful operation in this case. The average AUCs over all data sets are shown in Figure 6. The large standard deviation stems from the diversity of the different data sets. For TIRE, the average AUC remains very stable when the number of TD features is varied. Furthermore, the performance of TIRE seems optimal for 1 time-invariant FD feature, the average AUC when two or more FD feature are used is lower but does not further decrease with the number of FD features. Furthermore, we can observe that the performance of TIRE with $\lambda = 1$ is clearly superior over TIRE$_{\lambda=0}$. The increase in AUC is more distinct for higher numbers of TD features. This is unsurprising, as a larger latent dimension allows an autoencoder without time-invariant features to encode the feature more freely, making the positive effect of adding the time-invariant feature term to the loss function (9) all the more pronounced.

Next, we determine how sensitive TIRE is to the parameter K in the training loss (9). We let K vary from 1 to 10, with

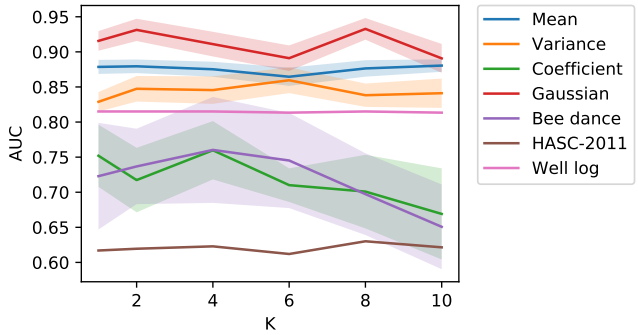

Fig. 7. Sensitivity of performance of TIRE to the parameter K in the TIRE training loss (9). We report for each data set the mean AUC and its standard error for K between 1 and 10.

other parameters as in previous experiments, and present the result in Figure 7. For most data sets the performance is stable with respect to changes in K, only for CC and bee dance a decrease in AUC is observed for large K. As also the runtime increases with K, we advise to set K rather small, e.g. $K \in [1, 5]$.

Finally, we investigate the sensitivity of TIRE with respect to the change magnitude at the change points (relative to the noise level). We do this by varying the standard deviation of the noise in the jumping mean data set (cf. Section IV-B), leaving the change magnitudes unchanged. The jumps in the mean are of magnitude $1/16, 2/16, \ldots, 3$ and we let the standard deviation of the noise vary from 0.5 to 3. Figure 8 shows a decrease of the AUC that is roughly proportionate to the fraction of change points for which the change magnitude is larger than the noise standard deviation. This is in line with expectations.

In general, we can conclude that the performance of TIRE does not depend critically on the exact value of the window size N, the number of features h and the parameter K.

VI. Discussion

In this section, we discuss some algorithmic design choices and mention potential limitations of the proposed method.

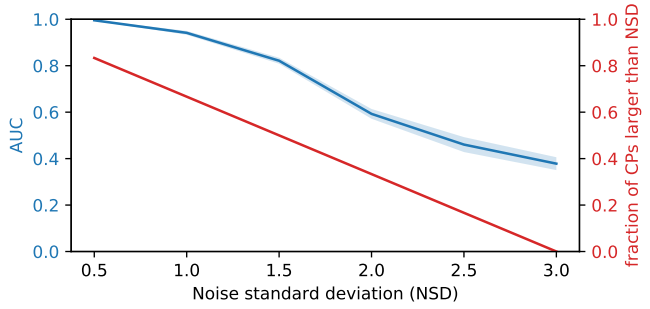

Fig. 8. Sensitivity of performance of TIRE to the standard deviation of the noise in the jumping mean data set. The average AUC over ten realizations and its standard deviation is shown together with the fraction of change points for which the change magnitude is larger than the noise standard deviation.

First, the combination of time-domain and frequency-domain information is extensively studied in the field of multi-view learning [52] and its applications. One approach is to simply concatenate separately learned TD and FD features, e.g. [53]. Another approach is to find a joint representation, which needs to take both views into account in an effective way. This can for instance be achieved using adaptive gradient blending [54]. In the context of CPD, it is however a priori unclear how to optimize this joint representation during training. We therefore choose to train the TD and FD autoencoders separately and use a CPD-tailored data-driven weighted concatenation to fuse both views into one representation. From Table II, it is clear that the AUC of TIRE (i.e. with TD and FD combined) is in general only slightly lower than the maximum of the AUCs of TIRE-TD and TIRE-FD, illustrating the good performance of our fusion approach.

Second, in this paper we focused on time series with only few channels. In this setting, we showed that the latent dimension of the autoencoder has little influence on the performance. Our method deliberately targets a lossy reconstruction due to a compressed representation in order to only learn the most important time-invariant properties of the time series segment. For high-dimensional time series data, e.g. supervisory control and data acquisition (SCADA) or electroencephalography (EEG) data, the choice of latent dimension might need further investigation. Alternatively, relevant channels can be selected using an application-specific method, e.g. [55].

We demonstrated the performance of TIRE using the AUC, but practitioners need to choose a suitable value of τ (cf. Section III-D) in order to use the method. As τ critically depends on domain knowledge and the needs of the practioner (e.g. their willingness to make a type I, resp. type II, error), we do not provide explicit guidelines. The tuning of τ can be facilitated if some prior knowledge is available, e.g. when part of the data is labelled or when an estimate of the number of change points is available. In case such information is not available and in case of doubt, we advise to underestimate τ as our proposed post-processing procedure effectively reduces the number of false positives.

It is also worth noting that TIRE can be interpreted as a nonlinear parametric CPD method that learns the relevant parameters from the data. Whereas classical parametric methods are often able to provide an (asymptotically correct) significance level for change point probabilities [13], [23], [31], [51], the interpretation of our change point score is rather limited. These theoretical guarantees for classical parametric methods however only hold under very specific assumptions on the data distribution, which are often not satisfied when real life data is used.

Finally, we showed that the use of filters to both smoothen the features itself (11) and the dissimilarity measure (14) generally leads to a significant improvement in AUC (see Table III). Care should however be taken when the peaks in the unfiltered dissimarity measure are either skewed or very close to each other. In the first case, the peak location might shift, leading to a false negative when the toleration distance is set too small. In the second case, the two peaks might either be joined to one peak, or one of the two peaks will have a very low prominence-based change point score. Given the good performance of TIRE (Table II), it is however clear that these are only minor concerns.

VII. Conclusion

We have proposed a novel distribution-free change point detection method based on autoencoders that learn a partially time-invariant representation that complies with the needs of CPD. Change points are calculated using a dissimilarity measure based on the Euclidean distance between the features learned from consecutive windows. We have mitigated the effect of false positive detections by proposing a postprocessing procedure using a matched filter and a prominence-based change point score. Furthermore, we have explicitly focused on non-iid time series by including temporally localized spectral information in the input of the autoencoder. The resulting method is very flexible, as it allows the user to indicate whether change points should be sought in the frequency domain, time domain or both. Examples of change points that can be detected are abrupt changes in the slope, mean, variance, autocorrelation function and frequency spectrum. Finally, we have showed that the performance of TIRE is consistently superior or highly competitive compared to baseline methods on benchmark data sets. A sensitivity analysis reveals that this good performance does not critically depend on the window size, nor on the latent dimension of the autoencoder. This robustness, together with the lack of distributional assumptions, make TIRE an easy-to-use change point detection method, whilst still offering a great deal of flexibility.

References

[1] A. Wald, *Sequential analysis*. Courier Corporation, 2004.
[2] E. Brodsky and B. S. Darkhovsky, *Nonparametric methods in change point problems*. Springer Science & Business Media, 2013, vol. 243.
[3] F. Gustafsson and F. Gustafsson, *Adaptive filtering and change detection*. Citeseer, 2000, vol. 1.
[4] M. Basseville, I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*. prentice Hall Englewood Cliffs, 1993, vol. 104.
[5] J. Reeves, J. Chen, X. L. Wang, R. Lund, and Q. Q. Lu, "A review and comparison of changepoint detection techniques for climate data," *Journal of Applied Meteorology and Climatology*, vol. 46, no. 6, p. 900–915, 2007.

[6] A. Pepelyshev and A. S. Polunchenko, "Real-time financial surveillance via quickest change-point detection methods," *Statistics and Its Interface*, vol. 10, no. 1, p. 93–106, 2017.

[7] D. A. Hsu, "A bayesian robust detection of shift in the risk structure of stock market returns," *Journal of the American Statistical Association*, vol. 77, no. 377, p. 29–39, 1982.

[8] Y. Wang, C. Wu, Z. Ji, B. Wang, and Y. Liang, "Non-parametric change-point method for differential gene expression detection," *PLoS ONE*, vol. 6, no. 5, 2011.

[9] A. G. Tartakovsky and V. V. Veeravalli, "Quickest change detection in distributed sensor systems," in *Proceedings of the 6th International Conference on Information Fusion*, 2003, pp. 756–763.

[10] ——, "Asymptotically optimal quickest change detection in distributed sensor systems," *Sequential Analysis*, vol. 27, no. 4, pp. 441–475, 2008.

[11] D. Michael and J. Houchin, "Automatic eeg analysis: a segmentation procedure based on the autocorrelation function," *Electroencephalography and clinical neurophysiology*, vol. 46, no. 2, pp. 232–235, 1979.

[12] R. Malladi, G. P. Kalamangalam, and B. Aazhang, "Online bayesian change point detection algorithms for segmentation of epileptic activity," in *2013 Asilomar Conference on Signals, Systems and Computers*. IEEE, 2013, pp. 1833–1837.

[13] X. Shao and X. Zhang, "Testing for change points in time series," *Journal of the American Statistical Association*, vol. 105, no. 491, pp. 1228–1240, 2010.

[14] A. Brandt, "Detecting and estimating parameter jumps using ladder algorithms and likelihood ratio tests," *ICASSP 83. IEEE International Conference on Acoustics, Speech, and Signal Processing*.

[15] U. Appel and A. V. Brandt, "Adaptive sequential segmentation of piecewise stationary time series," *Information Sciences*, vol. 29, no. 1, p. 27–56, 1983.

[16] T. Idé and K. Tsuda, "Change-point detection using krylov subspace learning," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 515–520.

[17] Y. Kawahara, T. Yairi, and K. Machida, "Change-point detection in time-series data based on subspace identification," in *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE, 2007, pp. 559–564.

[18] R. P. Adams and D. J. C. MacKay, "Bayesian Online Changepoint Detection," 2007. [Online]. Available: http://arxiv.org/abs/0710.3742

[19] M. Csörgő and L. Horvath, "20 nonparametric methods for changepoint problems," *Handbook of statistics*, vol. 7, pp. 403–425, 1988.

[20] W. C. Chang, C. L. Li, Y. Yang, and B. Póczos, "Kernel change-point detection with auxiliary deep generative models," *7th International Conference on Learning Representations, ICLR 2019*, pp. 1–14, 2019.

[21] Y. Kawahara and M. Sugiyama, "Sequential change-point detection based on direct density-ratio estimation," *Statistical Analysis and Data Mining*, vol. 5, no. 2, pp. 114–127, 2012.

[22] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, 2013.

[23] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020. [Online]. Available: https://doi.org/10.1016/j.sigpro.2019.107299

[24] B. Namoano, A. Starr, C. Emmanouilidis, and R. C. Cristobal, "Online change detection techniques in time series: An overview," in *IEEE International Conference on Prognostics and Health Management*, 2019.

[25] G. J. J. v. d. Burg and C. K. I. Williams, "An Evaluation of Change Point Detection Algorithms," pp. 1–33, 2020. [Online]. Available: http://arxiv.org/abs/2003.06222

[26] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 339–367, 2017.

[27] W.-H. Lee, J. Ortiz, B. Ko, and R. Lee, "Time Series Segmentation through Automatic Feature Learning," 2018. [Online]. Available: http://arxiv.org/abs/1801.05394

[28] K. C. Cheng, S. Aeron, M. C. Hughes, E. Hussey, and E. L. Miller, "Optimal transport based change point detection and time series segment clustering," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2020, pp. 6034–6038.

[29] V. Moskvina, "Change-point detection algorithm based on the singular-spectrum analysis," *Communications in Statistics: Simulation and Computation*, vol. 32, pp. 319–352, 2003.

[30] G. Chen, G. Lu, W. Shang, and Z. Xie, "Automated Change-Point Detection of EEG Signals Based on Structural Time-Series Analysis," *IEEE Access*, vol. 7, pp. 180 168–180 180, 2019.

[31] K. C. Cheng, S. Aeron, M. C. Hughes, and E. L. Miller, "On Matched Filtering for Statistical Change Point Detection," pp. 1–13, 2020. [Online]. Available: http://arxiv.org/abs/2006.05539

[32] M. Llobera, "Building past landscape perception with gis: Understanding topographic prominence," *Journal of Archaeological Science*, vol. 28, no. 9, p. 1005–1014, 2001.

[33] N. Wiener, "Generalized harmonic analysis," *Acta Mathematica*, vol. 55, p. 117–258, 1930.

[34] A. Khintchine, "Korrelationstheorie der stationären stochastischen prozesse," *Mathematische Annalen*, vol. 109, no. 1, p. 604–615, 1934.

[35] S. Li, Y. Xie, H. Dai, and L. Song, "M-statistic for kernel change-point detection," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 3366–3374. [Online]. Available: http://papers.nips.cc/paper/5684-m-statistic-for-kernel-change-point-detection.pdf

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[37] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," 2018.

[38] G. D. Nelson and R. Mckeon, "Peaks of people: Using topographic prominence as a method for determining the ranked significance of population centers," *The Professional Geographer*, vol. 71, no. 2, p. 342–354, 2019.

[39] J. Griffié, L. Boelen, G. Burn, A. P. Cope, and D. M. Owen, "Topographic prominence as a method for cluster identification in single-molecule localisation data," *Journal of Biophotonics*, vol. 8, no. 11-12, p. 925–934, 2015.

[40] M.-H. Choi, J. Ahn, D. J. Park, S. M. Lee, K. Kim, D.-I. D. Cho, S. S. Senok, K.-I. Koo, and Y. S. Goo, "Topographic prominence discriminator for the detection of short-latency spikes of retinal ganglion cells," *Journal of Neural Engineering*, vol. 14, no. 1, p. 016017, 2017.

[41] J. I. Takeuchi and K. Yamanishi, "A unifying framework for detecting outliers and change points from time series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 4, pp. 482–492, 2006.

[42] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert, "Learning and inferring motion patterns using parametric segmental switching linear dynamic systems," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 103–124, 2008.

[43] X. Xuan and K. Murphy, "Modeling changing dependency structure in multivariate time series," *ACM International Conference Proceeding Series*, vol. 227, pp. 1055–1062, 2007.

[44] R. D. Turner, "Gaussian Processes for State Space Models and Change Point Detection," *Learning*, 2011.

[45] N. Kawaguchi, Y. Yang, T. Yang, N. Ogawa, Y. Iwasaki, K. Kaji, T. Terada, K. Murao, S. Inoue, Y. Kawahara, Y. Sumi, and N. Nishio, "HASC2011corpus: Towards the common ground of human activity recognition," *UbiComp'11 - Proceedings of the 2011 ACM Conference on Ubiquitous Computing*, no. October 2014, pp. 571–572, 2011.

[46] T. O. Sort, J. J. O Ruanaidh, W. J. Fitzgerald, and K. J. Pope, "Recursive Bayesian location of a discontinuity in time series," in *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. iv, 4 1994, pp. IV/513–IV/516 vol.4.

[47] J. Knoblauch, J. Jewson, and T. Damoulas, "Doubly robust Bayesian inference for non-stationary streaming data with β-divergences," *Advances in Neural Information Processing Systems*, vol. 2018-Decem, no. NeurIPS 2018, pp. 64–75, 2018.

[48] P. Fearnhead and P. Clifford, "On-line inference for hidden Markov models via particle filters," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 65, no. 4, pp. 887–899, 2003. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00421

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[50] U. Appel and A. v. Brandt, "A comparative study of three sequential time series segmentation algorithms," *Signal Processing*, vol. 6, no. 1, pp. 45–60, 1984.

[51] R. A. Davis, D. Huang, and Y.-C. Yao, "Testing for a change in the parameter values and order of an autoregressive model," *The Annals of Statistics*, pp. 282–304, 1995.

[52] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Information Fusion*, vol. 38, pp. 43–54, 2017.

[53] Y. Yuan, G. Xun, K. Jia, and A. Zhang, "A multi-view deep learning method for epileptic seizure detection using short-time fourier transform," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2017, pp. 213–222.

[54] H. Phan, O. Y. Chén, P. Koch, A. Mertins, and M. De Vos, "Xsleepnet: Multi-view sequential model for automatic sleep staging," *arXiv preprint arXiv:2007.05492*, 2020.

[55] A. M. Narayanan and A. Bertrand, "Analysis of miniaturization effects and channel selection strategies for eeg sensor networks with application to auditory attention detection," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 1, pp. 234–244, 2020.