Differentiable Change-point Detection With Temporal Point Processes

Paramita Koley
IIT Kharagpur

Harshavardhan Alimi
IIT Kharagpur

Shrey Singla
IIT Bombay

Sourangshu Bhattacharya
IIT Kharagpur

Niloy Ganguly
IIT Kharagpur & L3S, Hannover

Abir De
IIT Bombay

Abstract

In this paper, we consider the problem of global change-point detection in event sequence data, where both the event distributions and change-points are assumed to be unknown. For this problem, we propose a Log-likelihood Ratio based Global Change-point Detector, which observes the entire sequence and detects a pre-specified number of change-points. Based on the Transformer Hawkes Process (THP), a well-known neural TPP framework, we develop DCPD, a differentiable change-point detector, along with maintaining distinct intensity and mark predictor for each partition. Further, we propose a sliding-window-based extension of DCPD to improve its scalability in terms of the number of events or change-points with minor sacrifices in performance. Experiments on synthetic datasets explore the effects of run-time, relative complexity, and other aspects of distributions on various properties of our change-point detectors, namely robustness, detection accuracy, scalability, etc., under controlled environments. Finally, we perform experiments on six real-world temporal event sequences collected from diverse domains like health, geographical regions, etc., and show that our methods either outperform or perform comparably with the baselines.

1 INTRODUCTION

Change-point detection in time-series is widely studied across several real-world applications, *e.g.*, finance (Schmitt et al., 2013), genetics (Grzegorczyk and Husmeier, 2009), cyber-security (Polunchenko et al., 2012), cryptocurrency (Thies and Molnár, 2018), crime (Albertetti et al., 2016), climate modelling (Nandhini and Devasena, 2019), earthquake modelling (Touati et al., 2016; Piana Agostinetti and Sgattoni, 2021), robotics (Konidaris et al., 2010), speech recognition (Panda and Nayak, 2016) etc. They proffer several recipes for accurate change point detection, which include Bayesian estimation through run-length modelling (Adams and MacKay, 2007; Fearnhead and Liu, 2007), density ratio estimation (Liu et al., 2013), kernel-based methods using maximum mean discrepancy (Chang et al., 2019; Li et al., 2015), sequential clustering methods (Khaleghi and Ryabko, 2014), *etc.*. However, these methods are suited specifically for discrete time-series data and cannot be immediately leveraged in a setup with continuous-time discrete events.

Responding to the above limitations, there is a flurry of recent works which focus on developing change point detection methods for continuous time event systems modeled using marked temporal point processes (MTPP). In this context, Li et al. (2017) propose a generalized likelihood ratio-based change-point detection framework for multi-dimensional Hawkes process, where they combine sequential hypothesis test and a sliding-window-based distributed, parameter-free EM-like algorithm. Very recently, Wang et al. (2021) presented a decentralized, memory-efficient recursive procedure for detecting change-point in a multi-dimensional Hawkes process. However, they suffer from the following limitations:

1. Such methods often assume prior knowledge about the model parameters, which constrains their practical applicability.
2. These methods are inherently designed for detecting single change-point in sequence and therefore decouple the change-point detection and model selection process. However, feedback from change-point detection process may be critical if multiple change-points are present in sequence. Moreover,

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

naive extensions of such methods for detecting multiple change-points may fail if the change-points are located too close in sequence. Also, these methods are built upon classic Hawkes process and do not consider state-of-the-art neural Hawkes models. Finally, these methods are built upon the general framework where the strength of change of each event in sequence is measured by comparing events in a window before and after the point. In such approaches, only a subset of events can be checked for longer sequences, resulting in compromise in performance at the cost of achieving reasonable run-time.

1.1 Our contributions

In this work, we propose DCPD, a likelihood ratio based change-point detection technique for temporal point process, built upon Transformer Hawkes process (Zuo et al., 2020). Specifically, we make the following contributions.

Novel setup of change point detection in MTPP. At the outset, the change-point in a sequence of events is an event that triggers a significant change in the dynamics of the subsequent events. This induces two different MTPP models before and after a change point event takes place. Thus, one requires to learn $K + 1$ models if an event sequence consists of K change point events. To this end, we first cast our change point detection problem as an instance of a discrete-continuous optimization task, which seeks to select a set of change points from a sequence of events as well as estimate the parameters of the model.

Bilevel optimization framework to detect the change points. Having obtained the optimization setup described above, we convert it into a continuous bi-level optimization problem. Here, we train the models using maximum likelihood estimation, and the change-points are estimated using a simple linear optimizer. In particular, this linear optimizer probes the likelihood ratio at each point and chooses the set of instances having the top-K highest likelihood ratios. Specifically, we use a differentiable convex programming method to train our method end-to-end, where we model the linear optimizer as an additional neural layer. In contrast to the existing sliding-window-based approaches, DCPD processes the entire event sequence. Moreover, it also provides a parameterized model for each partition. Note that our bilevel optimization solver effectively searches across all events for one single change point detection. This results in poor scalability. To improve our method, we also propose a sliding window-based extension of DCPD, which attempts to find change points across different segments of a sequence independently of others. Since the long-term dependencies across events are weaker than short-term dependencies in most practical temporal point process sequences, we observe a significant improvement in scalability with minimal drop in accuracy. We empirically validate our methods in several real-world datasets and observe that our methods perform better than state-of-the-art methods.

2 RELATED WORK

Our work is related to change-point detection in discrete time series, temporal point processes, and change-point detection in temporal point processes. Here, we briefly review some of these works.

Change point detection in discrete time series. Change-point detection in discrete time series are broadly categorized into offline and online methods. Offline methods require the knowledge of the full data sequence to identify the change-points. They include linear time dynamic programming algorithms (Killick et al., 2012; Maidstone et al., 2017), histogram-based methods (Boracchi et al., 2018), nonparametric least square model selection (Mazhar et al., 2018), among the list. In contrast, online change-point detection methods work towards detecting the change in data as the data arrive online. Online change-point detection methods are well-explored in literature (Page, 1957; Hawkins et al., 2003). In recent times, there is a large body of works in the line of Bayesian online change-point detection methods that involve simultaneous parameterized model selection (Adams and MacKay, 2007; Fearnhead and Liu, 2007; Garnett et al., 2009; Saatçi et al., 2010; Knoblauch and Damoulas, 2018; Alami et al., 2020; Titsias et al., 2022). More recently, researchers explore change-point detection in differentially private setting (Cummings et al., 2018; Lim et al., 2020; Cummings et al., 2020).

Among other notable works exist density ratio estimation techniques (Liu et al., 2013), kernel-based methods using maximum mean discriminations (Chang et al., 2019; Li et al., 2015), framing change-point detection as sequential clustering of temporal events (Khaleghi and Ryabko, 2014), robust bias aware regression in multistream data(Dong et al., 2017), only to name a few.

Recent works have also explored retrospective change-point detection methods, which allow a flexible time window to detect the change-point. Chandola et al. (2009) show robust detection performance of retrospective methods. Takeuchi and Yamanishi (2006) presents a unifying framework, where change-point is detected through gradually forgetting out-of-date statistics. Li et al. (2015) propose a set of M-statistics for kernel-based change detection methods. Among other works with parametric models and strong distributional assumptions, Yamanishi et al. (2000) consider change-point detection in auto-regressive models, whereas Kawahara et al. (2007) consider the setting of state-space models for tracking changes in the mean and variances of the distributions. However, most of these techniques are developed on the framework of time-series data or the setting where samples are iid from both back-

ground and post-change distributions.

Temporal point process. Marked temporal point processes have emerged as a powerful tool in modeling sequences of discrete events at different timestamps. Here, the meaning of mark varies across applications, *e.g.*, type or location or other information. Examples include tweets and posts on social media like Twitter and Facebook (Yang et al., 2011; Rodriguez et al., 2011), financial transaction (Bacry et al., 2015), personal healthcare (Wang et al., 2018), neural spike trains (Gerhard et al., 2017), electronic medical records, where tests and diagnoses of each patient can be treated as a sequence of events, etc.

Hawkes process (Hawkes, 1971; Zhou et al., 2013; Salehi et al., 2019) is a powerful tool for such data. Recently, several variants of recurrent neural network have been successfully employed for modeling the complex temporal dependencies of such event sequence (Du et al., 2016; Mei and Eisner, 2017; Xiao et al., 2017). Transformer Hawkes process (Zuo et al., 2020) is one such method that models event sequence data with both long-term and short-term dependencies without sacrificing computational efficiency via intelligent employment of transformers.

Change-point detection in Hawkes process. Unlike change-point detection in time-series data, the literature on change-point detection in temporal point process is quite sparse. Recently, Li et al. (2017) employ generalized likelihood ratio statistics based change-point detection framework for multi-dimensional Hawkes process, where they cast the change-point detection problem into sequential hypothesis test and derive likelihood ratios for the point process. On the contrary, Wang et al. (2021) adopts a recursive CUSUM-based procedure for sequentially detecting change-point in multi-dimensional Hawkes process. However, apriori knowledge of pre and post-change distribution parameters is required for this approach, which is not realistic. Wang et al. (2021) also considers score statistics (Xie and Siegmund, 2012) for change-detection in multi-dimensional Hawkes process. Our work differs from (Li et al., 2017; Wang et al., 2021) by combining model training with change-point detection instead of performing them in a disjoint manner, resulting in improved performance by its ability to exchange feedback among these two learning phases.

3 PROBLEM FORMULATION

Change-point detection problem in event sequence data aims to identify the change in underlying distributions of a sequence of events. Here we are interested in change-point detection in a setting where events are generated from temporal point process (TPP).

Given a sequence of positive random variables $\boldsymbol{t} = \{t_i\}_{i=1}^{N}$ representing the times of random occurrences of a set of N events, let $N(t)$ denote the number of events occurring before time $t \in \mathbb{R}^+$. The conditional probability of a new event occurring at time $(t, t+dt)$, given history, is specified by its conditional intensity function $\lambda(t)$.

$$\Pr(dN(t) = 1|H_t) = \lambda(t)dt \tag{1}$$

where H_t is the history of events till time t. Now we assume events are generated from a TPP, where the model parameters of the generating TPP may change at one or more time points in the sequence. More specifically, let us assume we are given a sequence of events $H = \{e_1, \ldots, e_N\}$. Each event e_i consists of $e_i = (t_i, m_i)$ where t_i is the timestamp of i-th event, m_i is the discrete type information of the event, takes value between 1 and L. Let us denote the set of change-points in the sequence as $\mathcal{C}$. Here, $\mathcal{C}$ splits the sequence into disjoint partitions, with events in each partition sampled from a TPP with its own set of parameters. We are interested in finding the set $\mathcal{C}$ from data. Here we consider the offline setting where the entire sequence is available apriori.

3.1 Detecting single change-point

Before going into the general case, let us begin with the special case of detecting a single change-point in data. Assume k^* be the index of change-point in the sequence and the change-point e_{k^*} splits H into two partitions H_1 and H_2, sampled from two different distribution P_{θ_1} and P_{θ_2} respectively.

Then, $P_{\theta_1}(H_1) \geq P_{\theta_2}(H_1)$ as well as $P_{\theta_2}(H_2) \geq P_{\theta_1}(H_2)$. Hence, change-point can be easily detected as the time-point t_{k^*} that achieves maximum log-likelihood ratio on the rest of the sequence $\{e_k, \ldots, e_N\}$. Formally, we have,

$$k^* = \underset{1 \leq k \leq N}{\operatorname{argmax}} \left[\sum_{i=k}^{N} \log \frac{P_{\theta_2}(e_i)}{P_{\theta_1}(e_i)} \right] \tag{2}$$

This is known as the generalized log-likelihood ratio test for change-point detection (Csörgö et al., 1997; Jandhyala et al., 2002). However, the key bottleneck is that the actual parameters are often not known apriori; therefore, change-point detection through log-likelihood ratio test should be combined with model selection on partitions.

Here, we view our change point detection problem as an instance of simultaneous instance selection and parameter estimation using a bi-level optimization framework where we learn the temporal model of the partitions through piece-wise maximum likelihood estimation. In contrast, we choose the optimal partitioning by maximizing the log-likelihood ratio of the learned models on the partitions. Here we see that when data is fixed, optimal k^* only de-

pends on (θ_1, θ_2).

$$LL_1 = \max_{\theta_1,\theta_2} \left[\sum_{i=1}^{k^*-1} \log P_{\theta_1}(e_i) + \sum_{j=k^*}^{N} \log P_{\theta_2}(e_j) \right] \quad \text{s.t. } k^* = \operatorname*{argmax}_{1\le k\le N} \left[\sum_{i=k}^{N} \log \frac{P_{\theta_2}(e_i)}{P_{\theta_1}(e_i)} \right] \tag{3}$$

We can address the same optimization problem as the following framework, where we jointly optimize k, θ_1, θ_2.

$$LL_2 = \max_{k,\theta_1,\theta_2} \left[\sum_{i=1}^{k-1} \log P_{\theta_1}(e_i) + \sum_{i=k}^{N} \log P_{\theta_2}(e_i) \right] \tag{4}$$

It can be shown that at optimality, $LL_1 = LL_2$ (See Appendix for details).

3.2 Generalization to detection of multiple change-points

Now consider the more general case where the sequence contains K change-points, $\mathcal{C} = \{t_{j_k^*}\}_{k=1}^{K}$ where $\boldsymbol{j}^* = \{j_1^*, \ldots, j_K^*\}$ denotes the set of change-point indices in the sequence with $j_{k_1^*} < j_{k_2^*}$ if $k_1 < k_2$. For convenience, let us assume $j_0^* = 1$ and $j_{K+1}^* = N + 1$. Therefore, we have $K + 1$ partitions $H = \{H_0, \ldots, H_K\}$ generated by $\boldsymbol{\theta} = \{\theta_0, \ldots, \theta_K\}$. Thus we have:

$$\max_{\boldsymbol{\theta}} \sum_{k=0}^{K} \sum_{i=j_k^*}^{j_{k+1}^*-1} \log P_{\theta_k}(e_i) \quad \text{s.t. } \boldsymbol{j}^* = \operatorname*{argmax}_{\boldsymbol{j}} \sum_{k=1}^{K} \sum_{i=j_k}^{N} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)} \tag{5}$$

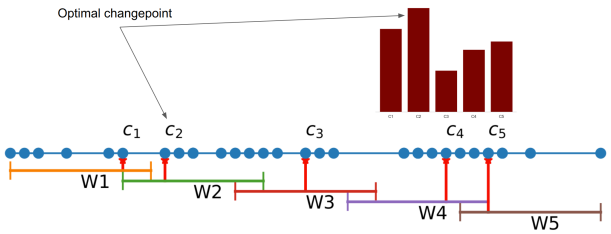
The equivalence of the optimal objective of equations(3) and (4), shown for single change-point, holds here too (see Appendix).

4 DCPD: A DIFFERENTIABLE CHANGE-POINT DETECTION ALGORITHM

Here we present DCPD, a differentiable change-point detection algorithm to the generic change-point detection problem in Eq. (5). We replace the optimal subset $\boldsymbol{j}^*$ selection problem through log-likelihood ratio maximization in Eq. (5) with its convex relaxation as follows. For the set of change-point indices $\boldsymbol{j}^* \in Z^K$, we introduce new matrix variable $\boldsymbol{A} \in [0, 1]^{K\times N}$. The k-th change-point j_k can be retrieved from k-th row of $\boldsymbol{A}$ as follows.

$$\mathcal{C} = \{t_j | j \in \boldsymbol{j}\}, \tag{6}$$

$$\boldsymbol{j} = \{j_k | j_k = \min\{i | a_{k,i} > 0.5\}\}_{1\le k\le K} \tag{7}$$



(a)

Figure 1: Illustration of sliding-window-based approach

After rounding $a_{k,\cdot}$, the k-th row of A, to the closest integer, the location of the first non-zero value in rounded vector indicates j_k, the k-th change-point in the sequence. We can rewrite the problem as

$$\max_{\boldsymbol{\theta}} \overbrace{\sum_{i=1}^{N} \sum_{k=0}^{K} (a_{k,i}^* - a_{k+1,i}^*) \log P_{\theta_k}(e_i)}^{\mathcal{L}(\boldsymbol{\theta},\boldsymbol{A})} \quad \text{s.t. } \boldsymbol{A}^* = \operatorname*{argmax}_{\boldsymbol{A}} \underbrace{\sum_{i=1}^{N} \sum_{k=1}^{K} a_{k,i} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)}}_{\mathcal{L}_R(\boldsymbol{\theta},\boldsymbol{A})} \tag{8}$$

$$a_{0,i} = 1, a_{K+1,i} = 0, 0 \le a_{k,i} \le 1 \; \forall i, k$$

$$a_{k,i} \ge a_{k+1,i}, a_{k,i} \le a_{k,i+1}$$

The feasible set of $\boldsymbol{A}$ is denoted as $\mathbb{A} = \{\boldsymbol{A} \in [0,1]^{K\times N} | a_{0,\cdot} = 1, a_{K+1,\cdot} = 0, a_{k,i} \ge a_{k+1,i}, a_{k,i} \le a_{k,i+1} \forall i, k\}$. The log-likelihood loss is represented by $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{A}) = \sum_{i=1}^{N} \sum_{k=0}^{K} (a_{k,i}^* - a_{k+1,i}^*) \log P_{\theta_k}(e_i)$ and the log-likelihood ratio loss within change-point detection objective is denoted as $\mathcal{L}_R(\boldsymbol{\theta}, \boldsymbol{A}) = \sum_{i=1}^{N} \sum_{k=1}^{K} a_{k,i} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)}$. The optimization is solved iteratively as follows. $\boldsymbol{\theta}^0$ is initialized randomly. In t-th epoch, $\boldsymbol{A}^t$ is computed from $\boldsymbol{\theta}^{t-1}$ as follows.

$$\boldsymbol{A}^t = \operatorname*{argmax}_{\boldsymbol{A}\in\mathbb{A}} \mathcal{L}_R(\boldsymbol{A}, \boldsymbol{\theta}^{t-1}) \tag{9}$$

Following that, model parameters are updated with the gradient of log-likelihood as

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} + \eta^t [\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{A}^t)]_{\boldsymbol{\theta}^{t-1}} \tag{10}$$

where η^t is the learning rate at t-th epoch. The final change-points $\mathcal{C}^*$ can be obtained from $\boldsymbol{A}^*$ using eq.(7).

Bottleneck of end-to-end training: While employing our end-to-end training method in practice, DCPD sometimes tends to converge to a trivial solution, where it repeatedly selects the end-points of the sequence as the set of optimal change-points and keeps on training one partition with the entire sequence whereas hardly allocating

Algorithm 1 DCPD

Input: Event sequence $H = \{e_1, .., e_N\}$, number of change-point K, learning rate η, constants p, r
Output: Change-points $\mathcal{C}^t$

Initialize $\boldsymbol{\theta}^0$ randomly, t=0, State=Train, reset counter n_p and n_r
while convergence is not reached **do**
  $\boldsymbol{A}^t$ = ComputeA($H, K, \boldsymbol{\theta}^{t-1}, \boldsymbol{A}^{t-1}$, State, n_P)
  Compute $\boldsymbol{\theta}^t$ using eq.(10)
  if State is Train **then**
    if $\mathcal{C}^t$ is repeated **then**
      Increment counter n_r
      If n_r reaches r, set State to Perturb, resetting n_p
    else
      Reset counter n_r
    end if
  end if
  if State is Perturb **then**
    Increment counter n_p
    If n_p reaches p, set State to Train, resetting n_r
  end if
  Compute $\mathcal{C}^t$ from $\boldsymbol{A}^t$ using eq.(7)
end while
return $\mathcal{C}^t, \mathcal{L}_R(\boldsymbol{A}^t, \boldsymbol{\theta}^t)$

Algorithm 2 ComputeA

Input: H, K, $\boldsymbol{\theta}_0$, $\boldsymbol{A}_0$,State, n_P
Output: $\boldsymbol{A}$

if State is Perturb **then**
  if n_P is 0 **then**
    Set $\boldsymbol{A}$ arbitrarily
  else
    Set $\boldsymbol{A}$ to its previous value $\boldsymbol{A}_0$
  end if
else
  Compute $\boldsymbol{A}$ using $\boldsymbol{A} = \text{argmax}_{\boldsymbol{A}\in\mathbb{A}}\, \mathcal{L}_{\mathcal{R}}(\boldsymbol{A}, \boldsymbol{\theta}_0)$
end if
return $\boldsymbol{A}$

any event to the other partitions. To avoid this issue, we resort to the following heuristics.

Perturbation: During training, DCPD keeps track of whether the algorithm is converging towards such sub-optimal solution. Upon detecting such an indication, it randomly resets the partitions and allows model parameters to be trained with the newly set partitions for a while before resuming the iterative joint optimization. More precisely, DCPD alternates between two states, namely 'Train' and 'Perturb'. In 'Train' state, $\boldsymbol{A}^t$ is computed optimizing $\mathcal{L}_R$ using eq. (9) and the set of change-points $\mathcal{C}^t$ is computed from $\boldsymbol{A}^t$. n_r keeps track of the number of consecutive repetitions of $\mathcal{C}^t$. If n_r exceeds a threshold r, the algorithm goes to 'Perturb' state, where the partitions as well as $\boldsymbol{A}^t$ are reset randomly. For the next p epochs, $\boldsymbol{A}^t$ is kept fixed and the model parameters are trained under the fixed partitions. After p epochs, the algorithm returns to 'Train' state to resume the usual training. We maintain a list of frequently repeated sets of change-points, along with the number of times it causes a transition to 'Perturb' state. If any solution causes such transition more than a threshold s, any further transition to 'Perturb' mode is disabled for rest of the training. We summarize the algorithm in Algorithm 1.

Implementation. In this section, we explain the implementation details of DCPD in Transformer Hawkes Process (THP) framework. Let H be the event sequence $\{e_i = (t_i, m_i)\}_{i=1}^N$ of N events, where each pair (t_i, m_i) corresponds to an event of mark m_i which has occured at time t_i. We pass H to transformer T, which returns a sequence of embeddings $\mathcal{Z} \in \mathbb{R}^{N\times M} = \{z_1, z_2, .., z_N\}$. The embedding $\mathcal{Z}$, learned via THP, is passed through two parallel feed-forward neural (FNN) networks to model the intensity and mark probability, respectively for each of the $K+1$ partitions for K change-points. For k-th partition, we compute predicted intensity $\tilde{\boldsymbol{\lambda}}^k \in \mathbb{R}^{N\times 1}$ and predicted mark probability $\tilde{\boldsymbol{M}}^k \in \mathbb{R}^{N\times L}$ as follows.

$$\boldsymbol{O}_{\lambda}^k = \text{ReLU}(\mathcal{Z}\boldsymbol{W}_1^k) \qquad \boldsymbol{O}_{\boldsymbol{m}}^k = \text{ReLU}(\mathcal{Z}\boldsymbol{W}_3^k) \tag{11}$$

$$\tilde{\boldsymbol{\lambda}}^k = \text{Softplus}(\boldsymbol{O}_{\boldsymbol{\lambda}}^k\boldsymbol{W}_2^k) \quad \tilde{\boldsymbol{M}}^k = \text{LogSoftmax}(\boldsymbol{O}_{\boldsymbol{M}}^k\boldsymbol{W}_4^k) \tag{12}$$

Softplus operator, defined as $f(x) = \beta\log(1+\exp(\frac{x}{\beta}))$ ensures positive intensity and stable, smooth learning. LogSoftmax applies a Softmax followed by an element-wise logarithm. Finally, l_i^k, log-likelihood of event e_i in k-th partition is generated as $l_i^k = \log\tilde{\lambda}_i^k - \int_{t_{i-1}}^{t_i}\tilde{\lambda}^k(s)\partial s - \mathbb{H}(\mathbb{1}_{m_i}, \tilde{\boldsymbol{m}}_i^k)$. $\tilde{\lambda}_i^k$ and $\tilde{\boldsymbol{m}}_i{}^k$ are computed intensity and mark probability vector of i-th event by k-th model. $\mathbb{1}_{m_i}$ is one-hot vector representation of m_i, $\mathbb{H}(\mathbb{1}_{m_i}, \tilde{\boldsymbol{m}}_i^k)$ is standard cross-entropy loss. We employ CVXPY[1](Agrawal et al., 2018; Diamond and Boyd, 2016) to solve the linear optimization to find $\boldsymbol{A}^t$ in eq.(9) and ADAM optimizer optimizes the loss in eq. (8). We terminate training if loss stops decreasing for more than 50 epochs. Note that, DCPD and DCPD-W are merely templates for change-point detection in MTPP. They are not limited to the THP framework and can be implemented in other existing MTPP frameworks.

Drawback of DCPD: DCPD scales quadratically with the number of events and change-points for it maintains $O(kn)$ parameters for $\boldsymbol{A}^t$ for a sequence of n events and k change-points. Also, the log-likelihood ratio optimiza-

[1]CVXPY is a domain-specific language for convex optimization embedded in Python, which is convenient to combine convex optimization with Pytorch.

Algorithm 3 DCPD-W

Input: $H = \{(t_i, m_i\}_{i=1}^N$ the event sequence, γ the step-size, L the window length, K the number of change-point
Output: Changepoints $\mathcal{C}^*$

Set $w = 0$
Set i such that $t_i \leq \frac{L}{2}$ & $t_{i+1} > \frac{L}{2}$
while $t_i + \frac{L}{2}$ does not exceed end of sequence **do**
  $S_w = [t_i - \frac{L}{2}, t_i + \frac{L}{2}]$
  $c_w, l_w^r = \text{DCPD}(S_w, 1)$ `Find change-point within` S_w
  $i = i + \gamma, w = w + 1$ `Shift the window`
end while
$\mathcal{C}^* = \{c_w | w \in S^*, S^* = \text{argmax}_{|S|=K} \sum_{w \in S} l_w^r\}$ `Select events with` K `highest scores`
return $\mathcal{C}^*$

tion with differentiable CVXPY layer is an inherently serial problem and, thereby, difficult to be parallelized.

4.1 DCPD-W: A window-based extension

To improve its practical utility, we propose DCPD-W, an immediate and intuitive sliding-window-based extension of DCPD. In DCPD-W, we split the given long sequence into multiple overlapping windows. DCPD is repeatedly applied to consecutive windows to select the most probable point of change from the events. The detected change-points from consecutive windows are further filtered by their scores to find the final change-points. We illustrate our algorithm in Figure 1. Let us formally state the method here. We consider S_w, a fixed-length window of consecutive events on the given sequence, and find c_w, the most probable change-point from that window. We measure the strength of c_w as a change-point by l_w^r, its log-likelihood ratio within the window as returned by DCPD. We repeatedly apply DCPD on consecutive windows by shifting window by γ events, where γ is a hyper-parameter of DCPD-W. We choose the final K change-points from the set of local change-points $\{c_w\}_{w=1}^W$, corresponding to top-K log-likelihood ratio scores, where W is the total number of windows. This approach substantially reduces both time and space complexity with a tolerable compromise with performance, as we empirically verify in the experiment section. We summarize the algorithm in Algorithm 3.

To sumarize, DCPD-W is proposed to improve scalability of DCPD with minor sacrifice in performance. Therefore, in a test setting, one should prefer DCPD over DCPD-W if the number of events is low. In case the number of change-points is known apriori, one can prefer DCPD if $n \times k$ is of reasonable size.

5 EXPERIMENTS

This section presents a comparative evaluation of proposed DCPD, DCPD-W and the three representative baselines on the synthetic datasets and six real-world datasets collected from diverse domains, namely spatio-temporal geography, disaster, health, and crime. In addition, the open-source code is publicly available on GitHub[2]. Here we briefly describe the metrics and baselines.

5.1 Metric.

NAE: Given true set of change-points $\boldsymbol{c}$ and estimated change-points $\tilde{\boldsymbol{c}}$, we define the metric as Normalized Average Error (NAE)$(\boldsymbol{c}, \tilde{\boldsymbol{c}}) = \min_{\tilde{c}^\pi \in \Pi(\tilde{\boldsymbol{c}})} \frac{\sum_i |c_i - \tilde{c_i}^\pi|}{|\boldsymbol{c}|}$, where $\Pi(\tilde{\boldsymbol{c}})$ is the set of all permutations of $\tilde{\boldsymbol{c}}$. For each dataset, we report the average NAE across all seeds across all sequences. NAE is a minor modification of detection delay, commonly used in CPD literature (Alami et al., 2020). Detection delay measures the delay between the real change-point and the detected change-point after the real change-point, whereas NAE measures absolute error between true change-point and detected change-point.

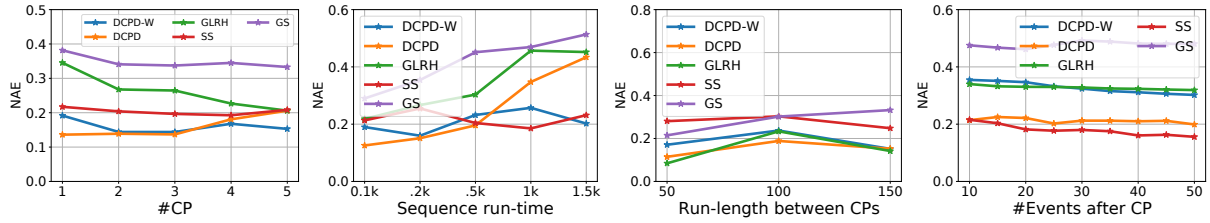
AUC: For quantitative evaluation of the scores returned by the score-based methods, we compute receiver operating characteristics (ROC) curves of change-point detection results and report the area-under-the-curve (AUC). For multiple change-points, AUC is calculated for each change-point separately, and the average AUC across all change-points is reported. This metric is presented for DCPD-Wand the baselines. AUC is commonly used in CPD literature (Chang et al., 2019). This metric summarizes overall quality of the scores returned by the method instead of only considering top-k scorer events.

5.2 Baselines

We compare DCPD and DCPD-W with three MTPP-based change-point detectors, GLRH, ScoreStat and Greedy-Sel. To ensure a fair comparison, THP is employed as the underlying MTPP framework for all baselines.

- Score Statistics (Wang et al., 2021): This method adopts a sliding window approach where for each event e, it trains events till e to obtain θ_{pre} and computes event score as $\text{Score}_e = \frac{1}{w} \frac{\partial(l_t - l_{t-w})}{\partial \theta_{pre}^T} I_0^{-1} \frac{\partial(l_t - l_{t-w})}{\partial \theta_{pre}}$, where w is the length of the window and l_t is the log-likelihood of events till time t. Essentially, it considers the derivative of log-likelihood to detect a change from the null hypothesis. It selects k change-points by marking events with k top scores.

[2] https://github.com/paramita1024/Changepoint-TPP



(a) (b) (c) (d)

Figure 2: Performance comparison on synthetic datasets across all methods.

- Generalized Log-likelihood Ratio Hawkes (GLRH) (Li et al., 2017): This method adopts a sliding window approach where for each event e, it trains two consecutive non-over-lapping window of events H_{pre}, H_{post} before and after e by $\theta_{pre}, \theta_{post}$ and set event score as $score_e = \sum_{e_i \in H_{post}} \log \frac{P_{\theta_{post}}(e_i)}{P_{\theta_{pre}(e_i)}}$. It selects the change-points by marking events with k top scores.

  For both methods above, we compute respective scores at events with small fixed intervals instead of all.

- Greedy Selection: This method adopts a score-based approach where for each event e, it partition the sequence and trains the two partitions of events H_{pre}, H_{post} before and after e by $\theta_{pre}, \theta_{post}$ and set event score as $score_e = \sum_{e_i \in H_{post}} \log \frac{P_{\theta_{post}}(e_i)}{P_{\theta_{pre}(e_i)}}$. It selects change-points by marking events with k top scores. The difference with GLRH is that instead of a fixed size window, it considers the entire sequence, splitting it into two parts. Because of the scalability issue, we need to randomly sample a subset of events instead of computing the score for all events, which affects its detection performance.

5.3 Experiments on synthetic datasets

To further explore various characteristics of DCPD, we perform the following experiments, where we vary the complexity of the change-point detection task in multiple ways and observe how our algorithm behaves in comparison with the baselines against various artifacts of data.

[Fig 2a] Varying the number of change-points: To investigate the effect of the number of change-points on performance, we perform the following test. We vary the number of change-points from 1 to 5 and generate 100 sequences for each setting, each sequence with run-time of 200. We observe that all methods show equivalent performance with change-points ≥ 4; however, DCPD performs significantly well when the number of changepoints is low. An explanation is that if the number of change-points is small, sliding-window-based approaches often mistake the true change-point with the other change-point candidates in the sequence because of a lack of complete information, which DCPD avoids because of its knowledge of the entire sequence. If we increase the number of change-points, the chance for such mistakes decreases because some of the potential change-points in the sequence are now part of true change-points. However, window-based DCPD-W retains its superior performance with increasing change-points, confirming our claim.

[Fig 2b] Varying run-time: Further we vary run-length from 100 to 2000 and we see that DCPD outperforms baselines till the run-length 500. However, sliding-window baselines are more accurate if we increase the run length beyond that. We can see that DCPD is more suitable for settings where we are more interested in detecting the change-points with higher accuracy in a smaller region. For finding such a region in a very long sequence, sliding-window-based methods can be more suitable. In contrast to DCPD, we find DCPD-W to degrade gracefully against increasing run-length, confirming our intuition that extending DCPD to window-based methods with appropriate window-length can improve performance for longer sequences.

[Fig 2c] Varying distance between change-points: Further, we complicate the change-point detection task by varying the distance between two change-point locations from 50 to 150. DCPD performs well here in comparison with baselines. DCPD-W behaves comparably with DCPD with a performance gap, showing the superiority of DCPD if run-time is limited.

[Fig 2d] Varying number of events after the change-point: To investigate the minimum amount of events required for the algorithms to detect the change-point, we perform the following experiment, where we fix a single change-point in data and vary the number of events after the change-point from 10 to 50. We see that ScoreStat is most accurate here, with maximum improvement in accuracy with increasing post-change-point events. However, DCPD remains a close competitor, outperforming GLRH here.

Datasets	DCPD-W	DCPD	ScoreStat	GLRH	Greedy-Sel
EQ	0.0837 ± 0.0709	**0.0225 ± 0.0003**	0.1950 ± 0.1004	0.3407 ± 0.2110	0.4770 ± 0.0008
Crime	0.1484 ± 0.2680	**0.0381 ± 0.0606**	0.1689 ± 0.1449	0.0725 ± 0.1100	0.5693 ± 0.1409
Rat	0.1506 ± 0.0873	**0.1460 ± 0.0564**	0.1880 ± 0.0375	0.1652 ± 0.0720	0.2335 ± 0.1203
Particle	0.2370 ± 0.1227	**0.2167 ± 0.1431**	0.2389 ± 0.0996	0.2555 ± 0.0446	0.4227 ± 0.0567
Anesthesia	**0.1013 ± 0.0442**	0.1591 ± 0.0181	0.3522 ± 0.1179	0.1312 ± 0.0396	0.2616 ± 0.0164
Arithmetic	0.0973 ± 0.0310	0.1683 ± 0.1621	0.1689 ± 0.0547	**0.0015 ± 0.0008**	-

Table 1: Comparative evaluation of DCPD and DCPD-W against baselines in six real-world datasets. For each dataset, best performance is indicated with bold font, and second best performance is underlined.

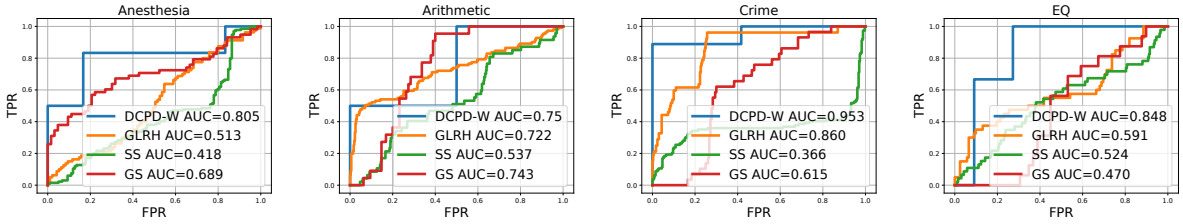


Figure 3: ROC curve of real datasets for DCPD-W against different baselines.

Dataset	# Events	# CP
EQ	4000	1
Crime	4687	1
Rat	3128	2
Particle	343	3
Anesthesia	1701	2
Arithmetic	6901	1

Table 2: Dataset description

5.4 Experiments on real-world datasets

Dataset: The data statistics are summarized in Table 2. All datasets are pre-processed by scaling the run-time to 100 and features to $[0, 1]$.

Particle (Altieri et al., 2015) records a single sequence of 343 events, presenting the occurrence of radio-active particles on Sandside beaches in Scotland, with three change-points, where the first two change-points took place due to instrument installation and the third change-point is due to change in data distributions, which is assumed to be the effect of the environmentalist activities and constant pressure from local inhabitants.

EQ (Altieri et al., 2016) records the sequence of seismic events in Italy from July 2001 to May 2014, recording a single sequence of 4000 events with magnitude > 3. After 2008, two major seismic events in L'Aquila and Emilia-Romagna region shocked Italy. Until 2008, earthquakes were evenly distributed all along the Appennini; afterward, a clusterization took place around the central-east part of Italy and the volcanic islands close to Sicily. Following this fact, the year 2009 is considered the change-point.

Rat (Watson et al., 2016a,b) records neural spike trains recorded in the rats. In the experiment, we consider four sequences, each with 3128 events on average. Each sequence includes a "Wake-Sleep" episode, where at least 7 minutes of wake is followed by 20 minutes of sleep, consisting of REM, non-REM, and micro-arousal phases. In our sequence, the cluster of microarousal phases indicates the end of sleep phase. Change of phase from wake to sleep or vice versa causes a change in the interarrival-time distributions of putative neurons (Watson et al., 2016b). Accordingly, the start and end of sleep mark the change-points.

Arithmetic (Zyma et al., 2019; Goldberger et al., 2000) contains EEG recordings of subjects before and during the performance of mental arithmetic tasks, namely serial subtraction of two numbers. For each subject, EEG of 60 seconds before and during arithmetic operation is recorded. The start of arithmetic operation indicates the changepoint. The dataset consists of EEG values at fixed intervals, which is converted to discrete marked event sequence by discretizing the range of EEG values into 16 categories.

Crime contains crime events in Chicago. During covid outbreak, the government proposed a number of orders, including 'Stay-at-Home' from March 2020. Reportedly since this period, the temporal distribution of crime changed drastically. The dataset can be downloaded from open source repository[3]. We consider events from

[3] https://rb.gy/8xvjn5

01/01/2017 to present, with 22/03/2020 as the change-point. Moreover, we consider events inward #42, district #1, and community #32, and we consider the five most frequent crimes - retail theft, theft from a building, credit card fraud, theft of at most $ 500, and pick-pocketing.

Anesthesia (Goldberger et al., 2000; Subramanian et al., 2021) records electro-dermal activity (EEG) of healthy volunteers when propofol infusion rate is varied in phases. Propofol produces unconsciousness, along with autonomic effects as a vasodilator and myocardial depressant. In our experiment, we use a truncated sequence with 3 such phases, and each phase change is considered as change-point, causing 2 change-points in the sequence.

In both spatio-temporal datasets (Particle and EQ), event locations are clustered, and cluster labels are used as marks of the events. In health datasets (Anesthesia, Arithmetic, and Rat), event-specific continuous health measurements are converted to discrete marks by splitting the range of values into disjoint intervals. The five types of crimes considered in the crime dataset are used as the marks in Crime.

Results on real-world datasets: In Table 1, we present a comparative evaluation of DCPD, DCPD-W and baselines in six real-world datasets. DCPD shows significant improvements on all datasets, except being in third place on the Arithmetic and Anesthesia datasets. We find that EQ, where DCPD shows the biggest improvement, is a complex spatio-temporal dataset with a single change-point, causing a single spatio-temporal shift. However, many minor short-lived variations exist throughout, and sliding-window-based baselines sometimes choose them as the change-point, lowering the average accuracy. Performance of DCPD-W indicates the utility of joint learning, even in small windows. Crime contains single change-point towards end of the sequence and both DCPD and GLRH detect that with negligible detection error with DCPD marginally outperforming GLRH. Rat is a neural spike train dataset, which inherently exhibits self-excitation effects, making it suitable for Hawkes. As we see here, joint training of model learning and change detection gives DCPD additional advantage over GLRH or ScoreStat, resulting in significant improvement over the baselines. Particle presents a challenging setting, with a small sample size of 353 events and 3 visually not-so-prominent change-points. Here better knowledge of the process through joint training of model learning and change detection allows DCPD to enjoy an additional advantage over local observation-based methods like GLRH or ScoreStat. Finally, Arithmetic is a dataset with a single sharp change and almost no other significant variation. GLRH is well-suited for detecting this kind of change, for it works by comparing two consecutive short windows. However, DCPD-W, the window-based variant of DCPD manages to secure second best position, showing the efficiency of window-based methods for detecting such changes. We observe DCPD-W to perform second best to DCPD for almost all datasets except optimal performance in Anesthesia and third position in Crime. The minor performance gap between DCPD-W and DCPD is intuitive because of its partial observation to facilitate scalability. DCPD-W often outperforms other sliding-window-based methods such as GLRH or ScoreStat because of its ability to detect change-point more accurately within the window. These results re-confirm that DCPD is more effective for a more challenging setting with not-so-long sequences over the competitors. For longer sequences, DCPD-W is more appropriate for its scaling property.

Figure 3 presents ROC curve of DCPD-W against baselines for four selected datasets for selected seeds. We omit DCPD here as DCPD does not provide any location-wise scores. ROC curve facilitates qualitative evaluation of the entire score function instead of only assessing quality of top-k events returned by the algorithm. Here we find DCPD-W to perform promising in comparison with baselines, supporting its detection error-based superiority over baselines.

6 CONCLUSION

We propose DCPD, a new principled change-point detector for event sequence data by formulating a novel bi-level optimization problem to combine change-point detection with model selection in an offline way. We have extended the framework of Transformer Hawkes process, giving the first neural change-point detector for the Hawkes process to the best of our knowledge. We also propose DCPD-W, that is particularly suitable for longer sequences and highly scalable with minor sacrifice in performance. Extensive evaluation of DCPD on synthetic and multiple real-world datasets shows the utility and efficacy of DCPD in difficult (to identify) real-world change-point detection scenarios.

Acknowledgements

We thank all the reviewers and program committee members for their insightful suggestions and useful comments and all the colleagues who contributed to the ideas. This research was (partially) funded by the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor with grant No. 01DD20003 and an Intel Inc, India project. Abir De acknowledges a Google Faculty Grant and IBM AI Horizon grant.

References

Adams, R. P. and MacKay, D. J. (2007). Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*.

Agrawal, A., Verschueren, R., Diamond, S., and Boyd, S. (2018). A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60.

Alami, R., Maillard, O., and Féraud, R. (2020). Restarted bayesian online change-point detector achieves optimal detection delay. In *International conference on machine learning*, pages 211–221. PMLR.

Albertetti, F., Grossrieder, L., Ribaux, O., and Stoffel, K. (2016). Change points detection in crime-related time series: an on-line fuzzy approach based on a shape space representation. *Applied Soft Computing*, 40:441–454.

Altieri, L., Cocchi, D., Greco, F., Illian, J. B., and Scott, E. (2016). Bayesian p-splines and advanced computing in r for a changepoint analysis on spatio-temporal point processes. *Journal of Statistical Computation and Simulation*, 86(13):2531–2545.

Altieri, L., Scott, E. M., Cocchi, D., and Illian, J. B. (2015). A changepoint analysis of spatio-temporal point processes. *Spatial Statistics*, 14:197–207.

Bacry, E., Bompaire, M., Gaïffas, S., and Poulsen, S. (2017). Tick: a python library for statistical learning, with a particular emphasis on time-dependent modelling. *arXiv preprint arXiv:1707.03003*.

Bacry, E., Mastromatteo, I., and Muzy, J.-F. (2015). Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01):1550005.

Boracchi, G., Carrera, D., Cervellera, C., and Maccio, D. (2018). Quanttree: Histograms for change detection in multivariate data streams. In *International Conference on Machine Learning*, pages 639–648. PMLR.

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

Chang, W.-C., Li, C.-L., Yang, Y., and Póczos, B. (2019). Kernel change-point detection with auxiliary deep generative models. *arXiv preprint arXiv:1901.06077*.

Csörgö, M., Csörgö, M., and Horváth, L. (1997). Limit theorems in change-point analysis.

Cummings, R., Krehbiel, S., Lut, Y., and Zhang, W. (2020). Privately detecting changes in unknown distributions. In *International Conference on Machine Learning*, pages 2227–2237. PMLR.

Cummings, R., Krehbiel, S., Mei, Y., Tuo, R., and Zhang, W. (2018). Differentially private change-point detection. *Advances in neural information processing systems*, 31.

Diamond, S. and Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5.

Dong, B., Li, Y., Gao, Y., Haque, A., Khan, L., and Masud, M. M. (2017). Multistream regression with asynchronous concept drift detection. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 596–605.

Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., and Song, L. (2016). Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1555–1564.

Fearnhead, P. and Liu, Z. (2007). On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):589–605.

Garnett, R., Osborne, M. A., and Roberts, S. J. (2009). Sequential bayesian prediction in the presence of changepoints. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 345–352.

Gerhard, F., Deger, M., and Truccolo, W. (2017). On the stability and dynamics of stochastic spiking neuron models: Nonlinear hawkes process and point process glms. *PLoS computational biology*, 13(2):e1005390.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220.

Grzegorczyk, M. and Husmeier, D. (2009). Non-stationary continuous dynamic bayesian networks. *Advances in neural information processing systems*, 22.

Hawkes, A. G. (1971). Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90.

Hawkins, D. M., Qiu, P., and Kang, C. W. (2003). The changepoint model for statistical process control. *Journal of quality technology*, 35(4):355–366.

Jandhyala, V. K., Fotopoulos, S. B., and Hawkins, D. M. (2002). Detection and estimation of abrupt changes in the variability of a process. *Computational statistics & data analysis*, 40(1):1–19.

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

Khaleghi, A. and Ryabko, D. (2014). Asymptotically consistent estimation of the number of change points in highly dependent time series. In *International Conference on Machine Learning*, pages 539–547. PMLR.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012). Optimal detection of changepoints with a linear computa-

tional cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Knoblauch, J. and Damoulas, T. (2018). Spatio-temporal bayesian on-line changepoint detection with model selection. In *International Conference on Machine Learning*, pages 2718–2727. PMLR.

Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. (2010). Constructing skill trees for reinforcement learning agents from demonstration trajectories. *Advances in neural information processing systems*, 23.

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

Li, S., Xie, Y., Farajtabar, M., Verma, A., and Song, L. (2017). Detecting changes in dynamic events over networks. *IEEE Transactions on Signal and Information Processing over Networks*, 3(2):346–359.

Lim, H., Che, G., Lee, W., and Yang, H. (2020). Differentiable algorithm for marginalising changepoints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4828–4835.

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

Maidstone, R., Hocking, T., Rigaill, G., and Fearnhead, P. (2017). On optimal multiple changepoint algorithms for large data. *Statistics and computing*, 27(2):519–533.

Mazhar, O., Rojas, C., Fischione, C., and Hesamzadeh, M. R. (2018). Bayesian model selection for change point detection and clustering. In *International Conference on Machine Learning*, pages 3433–3442. PMLR.

Mei, H. and Eisner, J. M. (2017). The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30.

Nandhini, V. and Devasena, M. G. (2019). Predictive analytics for climate change detection and disease diagnosis. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, pages 1152–1155. IEEE.

Page, E. (1957). On problems in which a change in a parameter occurs at an unknown point. *Biometrika*, 44(1/2):248–252.

Panda, S. P. and Nayak, A. K. (2016). Automatic speech segmentation in syllable centric speech recognition system. *International Journal of Speech Technology*, 19(1):9–18.

Piana Agostinetti, N. and Sgattoni, G. (2021). Changepoint detection in seismic double-difference data: application of a trans-dimensional algorithm to data-space exploration. *Solid Earth*, 12(12):2717–2733.

Polunchenko, A. S., Tartakovsky, A. G., and Mukhopadhyay, N. (2012). Nearly optimal change-point detection with an application to cybersecurity. *Sequential Analysis*, 31(3):409–435.

Rodriguez, M. G., Balduzzi, D., and Schölkopf, B. (2011). Uncovering the temporal dynamics of diffusion networks. *arXiv preprint arXiv:1105.0697*.

Saatçi, Y., Turner, R. D., and Rasmussen, C. E. (2010). Gaussian process change point models. In *ICML*.

Salehi, F., Trouleau, W., Grossglauser, M., and Thiran, P. (2019). Learning hawkes processes from a handful of events. *Advances in Neural Information Processing Systems*, 32.

Schmitt, T. A., Chetalova, D., Schäfer, R., and Guhr, T. (2013). Non-stationarity in financial time series: Generic features and tail behavior. *EPL (Europhysics Letters)*, 103(5):58003.

Subramanian, S., Purdon, P., Barbieri, R., and Brown, E. (2021). Behavioral and autonomic dynamics during propofol-induced unconsciousness.

Takeuchi, J.-i. and Yamanishi, K. (2006). A unifying framework for detecting outliers and change points from time series. *IEEE transactions on Knowledge and Data Engineering*, 18(4):482–492.

Thies, S. and Molnár, P. (2018). Bayesian change point analysis of bitcoin returns. *Finance Research Letters*, 27:223–227.

Titsias, M. K., Sygnowski, J., and Chen, Y. (2022). Sequential changepoint detection in neural networks with checkpoints. *Statistics and Computing*, 32(2):1–19.

Touati, S., Naylor, M., and Main, I. (2016). Detection of change points in underlying earthquake rates, with application to global mega-earthquakes. *Geophysical Journal International*, 204(2):753–767.

Wang, H., Xie, L., Xie, Y., Cuozzo, A., and Mak, S. (2021). Sequential change-point detection for mutually exciting point processes over networks. *arXiv preprint arXiv:2102.05724*.

Wang, L., Zhang, W., He, X., and Zha, H. (2018). Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2447–2456.

Watson, B., Levenstein, D., Greene, J. P., Gelinas, J. N., and Buzsáki, G. (2016a). Multi-unit spiking activity recorded from rat frontal cortex (brain regions mpfc, ofc, acc, and m2) during wake-sleep episode wherein at least 7 minutes of wake are followed by 20 minutes of sleep. *CRCNS. org*, 10:K02N506Q.

Watson, B. O., Levenstein, D., Greene, J. P., Gelinas, J. N., and Buzsáki, G. (2016b). Network homeostasis and state dynamics of neocortical sleep. *Neuron*, 90(4):839–852.

Xiao, S., Yan, J., Yang, X., Zha, H., and Chu, S. (2017). Modeling the intensity function of point process via recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Xie, Y. and Siegmund, D. (2012). Spectrum opportunity detection with weak and correlated signals. In *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 128–132. IEEE.

Yamanishi, K., Takeuchi, J.-I., Williams, G., and Milne, P. (2000). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 320–324.

Yang, S.-H., Long, B., Smola, A., Sadagopan, N., Zheng, Z., and Zha, H. (2011). Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*, pages 537–546.

Zhou, K., Zha, H., and Song, L. (2013). Learning triggering kernels for multi-dimensional hawkes processes. In *International conference on machine learning*, pages 1301–1309. PMLR.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. (2020). Transformer hawkes process. In *International Conference on Machine Learning*, pages 11692–11702. PMLR.

Zyma, I., Tukaev, S., Seleznov, I., Kiyono, K., Popov, A., Chernykh, M., and Shpenkov, O. (2019). Electroencephalograms during mental arithmetic task performance. *Data*, 4(1):14.

A PROOF OF EQUIVALENCE OF BI-LEVEL AND JOINT OPTIMIZATION

Equivalence: It can be shown that at optimality, the objective in equation (3) can be reduced to the objective in equation (4). Let

$$k^*(\theta_1, \theta_2) = \underset{1 \le k \le N}{\operatorname{argmax}} \left[\sum_{i=k}^{N} \log \frac{P_{\theta_2}(e_i)}{P_{\theta_1}(e_i)} \right]$$

We can write LL_1 as

$$\begin{aligned} LL_1 &= \max_{\theta_1, \theta_2} \left[\sum_{i=1}^{N} \log P_{\theta_1}(e_i) + \sum_{i=k^*(\theta_1,\theta_2)}^{N} \log \frac{P_{\theta_2}(e_i)}{P_{\theta_1}(e_i)} \right] \\ &= \max_{\theta_1, \theta_2} \left[\sum_{i=1}^{N} \log P_{\theta_1}(e_i) + \max_{k} \sum_{i=k}^{N} \log \frac{P_{\theta_2}(e_i)}{P_{\theta_1}(e_i)} \right] \\ &= \max_{k, \theta_1, \theta_2} \left[\sum_{i=1}^{N} \log P_{\theta_1}(e_i) + \sum_{i=k}^{N} \log \frac{P_{\theta_2}(e_i)}{P_{\theta_1}(e_i)} \right] \\ &= \max_{k, \theta_1, \theta_2} \left[\sum_{i=1}^{k-1} \log P_{\theta_1}(e_i) + \sum_{i=k}^{N} \log P_{\theta_2}(e_i) \right] = LL_2 \end{aligned}$$

Theorem 1 *The optimization problems*

$$\begin{aligned} LL_1 &= \max_{\theta_1, \theta_2} \left[\sum_{i=1}^{k^*-1} P_{\theta_1}(e_i) + \sum_{i=k^*}^{N} P_{\theta_2}(e_i) \right] \\ \textit{s.t. } k^* &= \underset{1 \le k \le N}{\operatorname{argmax}} \sum_{i=k}^{N} \log \frac{P_{\theta_2}(e_i)}{P_{\theta_1}(e_i)} \end{aligned} \tag{13}$$

and

$$LL_2 = \max_{\theta_1, \theta_2, k} \left[\sum_{i=1}^{k-1} \log P_{\theta_1}(e_i) + \sum_{i=k}^{N} \log P_{\theta_2}(e_i) \right] \tag{14}$$

are same.

Theorem 2 *This same idea can be extended to the multiple change-point case as well. Suppose there are K change-points in the sequence. The optimization problems*

$$\begin{aligned} LL_1 &= \max_{\boldsymbol{\theta}} \sum_{k=0}^{K} \sum_{i=j_k^*}^{j_{k+1}^*-1} \log P_{\theta_k}(e_i) \\ \textit{s.t. } \boldsymbol{j}^* &= \underset{\boldsymbol{j}}{\operatorname{argmax}} \sum_{k=1}^{K} \sum_{i=j_k}^{N} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)} \end{aligned} \tag{15}$$

and

$$LL_2 = \max_{\boldsymbol{\theta}, \boldsymbol{j}} \sum_{k=0}^{K} \sum_{i=j_k}^{j_{k+1}-1} \log P_{\theta_k}(e_i) \tag{16}$$

are same.

Proof: Let

$$\boldsymbol{j}^*(\boldsymbol{\theta}) = \underset{\boldsymbol{j}}{\operatorname{argmax}} \sum_{k=1}^{K} \sum_{i=j_k}^{N} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)}$$

LL_1 can also be written as

$$\begin{aligned}
LL_1 &= \max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{N} \log P_{\theta_0}(e_i) + \sum_{k=1}^{K} \sum_{i=j_k^*(\boldsymbol{\theta})}^{N} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)} \right] \\
&= \max_{\boldsymbol{\theta}} \left[\sum_{i=1}^{N} \log P_{\theta_0}(e_i) + \max_{\boldsymbol{j}} \sum_{k=1}^{K} \sum_{i=j_k}^{N} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)} \right] \\
&= \max_{\boldsymbol{\theta}, \boldsymbol{j}} \left[\sum_{i=1}^{N} \log P_{\theta_0}(e_i) + \sum_{k=1}^{K} \sum_{i=j_k}^{N} \log \frac{P_{\theta_k}(e_i)}{P_{\theta_{k-1}}(e_i)} \right] \\
&= \max_{\boldsymbol{\theta}, \boldsymbol{j}} \left[\sum_{k=0}^{K} \sum_{i=j_k}^{j_{k+1}-1} \log P_{\theta_k}(e_i) \right] \\
&= LL_2
\end{aligned}$$

Hence proved.

B IMPLEMENTATION

B.1 Transformer Hawkes process

Here we briefly describe the Transformer Hawkes process (THP). Given the event sequence $H = \{e_i\}_{1 \le i \le N}$, THP employs a trigonometric function-based operation on the event time t_i to generate temporal encoding $z_i \in \mathbb{R}^M$ as follows.

$$[z_i]_j = \begin{cases} \cos(t_i / 10000^{\frac{j-1}{M}}), & \text{if } j \text{ is odd,} \\ \sin(t_i / 10000^{\frac{j-1}{M}}), & \text{otherwise} \end{cases}$$

Similarly, THP trains an embedding matrix $U \in \mathbb{R}^{M \times L}$ for encoding event types. Let $Y \in \mathbb{R}^{L \times N}$ be the one-hot vector representation of event types $\{m_i\}_{1 \le i \le N}$. Then $UY \in \mathbb{R}^{M \times N}$ denotes the type embedding of events. The encoded event sequence denoted as $X = (UY + Z)^T \in \mathbb{R}^{N \times M}$ is further passed through the self-attention module to compute attention output S.

$$S = \text{Softmax}(\frac{QK^T}{\sqrt{M_K}})V \tag{17}$$

$$Q = XW^Q, K = XW^K, V = XW^V \tag{18}$$

Here Q, K, and V are query, key, and value matrices obtained by different transformations of X and $W^Q, W^K \in \mathbb{R}^{M \times M_K}$ and $W^V \in \mathbb{R}^{M \times M_V}$ are weights for linear transformations. Now, H_T number of attention outputs are aggregated in final attention output S_f as follows.

$$S_f = [S_1, S_2, \ldots, S_{H_T}]W^O \tag{19}$$

where $W^O \in \mathbb{R}^{H_T M_V \times M}$ is the aggregation matrix. Finally $S_f \in \mathbb{R}^{N \times M}$ passes through a feed-forward neural network to generate hidden representation $\mathcal{Z}$:

$$\mathcal{Z} = \text{ReLU}(S_f W_1^{FC} + b_1)W_2^{FC} + b_2 \tag{20}$$

Here $W_1^{FC} \in \mathbb{R}^{M \times M_H}, W_2^{FC} \in \mathbb{R}^{M_H \times M}$ are linear transformations. $b_1 \in \mathbb{R}^{M_H}, b_2 \in \mathbb{R}^M$. Each row of $\mathcal{Z} \in \mathbb{R}^{N \times M}$ represents the embedding of a particular event. In practice, the transformer passes the input through attention modules sequentially to capture high-level dependency in data.

We describe the complete implementation of DCPD in Figure 4.

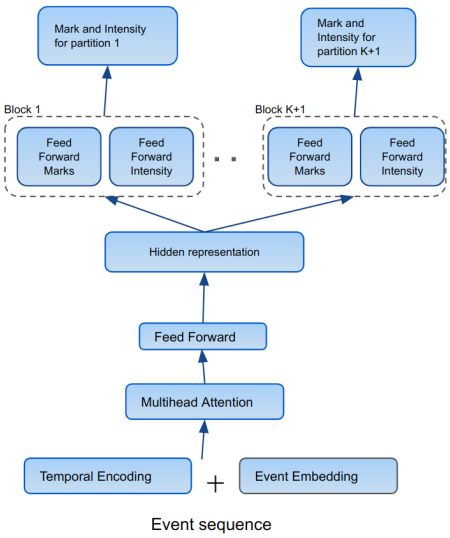


Figure 4: Architecture of DCPD. Event sequence H is encoded and passed through the attention module to generate hidden representation $\mathcal{Z}$. $\mathcal{Z}$ is passed through $K+1$ FFN modules to generate mark and intensity according to the corresponding module.

B.2 Implementational details of DCPD-W

Hyper-parameters: The accuracy of DCPD-W closely depends on γ, the step size, and L, the window length. Smaller γ increases the run-time, requiring a higher number of windows. Increasing L results in more accurate change-point detection within the windows but increases the run-time per window. In practice, we need to tradeoff between performance and run-time while choosing optimal L. The nature of data also influences the optimal value of L; for example, noisy data requires a bigger L.

Threshold While employing DCPD-W in practice, we can take following approach. Let $\{l_w^r\}_{w=1}^W$ be the log-likelihood ratio scores of W candidate changepoints $\{c_w\}_{w=1}^W$, obtained via running DCPD on W partially over-lapping windows of the sequence. Let τ be a pre-decided threshold. We defer the discussion on the choice of τ to the end of this section. Then, final changepoints are $\{c_w | l_w^r > \tau\}$. The key step in such a detection algorithm is finding an appropriate threshold. The choice of threshold controls a tradeoff between two performance measures, namely false alarm rate and detection delay. Higher τ results in a smaller number of false positives with higher detection delay and vice versa. Following usual practice, we can estimate the threshold via direct Monte Carlo by running our DCPD-W over null distribution while setting the average run length to a fixed value.

C EXPERIMENTAL SETUP AND ADDITIONAL RESULTS

C.1 Synthetic data generation

To vary the number of change-points, we generate 100 sequences, each for run-time 200 for number of change-points between 1 to 5. For the experiment of varying the run-time of the sequence, we vary run-time from 100 to 1500. For each of the run-time, we generate 10 sequences, each with 2 change-points. To vary run-length between change-points, we vary run-length between change-points in the range $\{50, 100, 150\}$. For each of these run-lengths, we generate 10 sequences, each with total run-time 500 and 2 change-points. In the experiment, where we vary the number of events after change-point from 10 to 50, we generate 50 sequences for each of the configurations, each sequence with single change-point. The run-time of each sequence is set to 100 here. To vary difference between baseline intensity of pre and post-change-point distribution, we set the difference in baseline intensity from 0.1 to 0.5. For each of the configurations, we generate 20 sequences, each with run-time 500 and single change-point. In a similar experiment, where instead of baseline intensity difference, we vary difference of intensity influence coefficients from 0.1 to 0.3. For each configuration, we generate 20 sequences, each with 500 run-time and single change-point. Finally, where we vary the dimension of the generating point process from 1 to 5, we generate 20 sequences per configuration, each sequence with run-time 500

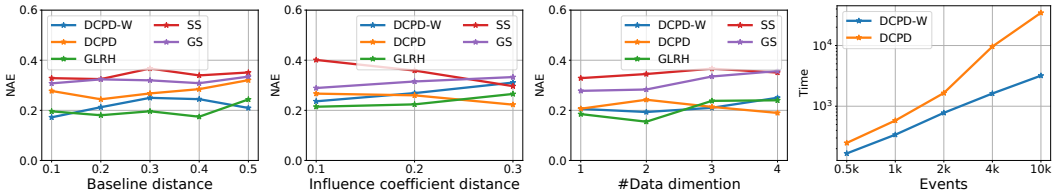


Figure 5: Performance comparison on synthetic datasets across all methods.

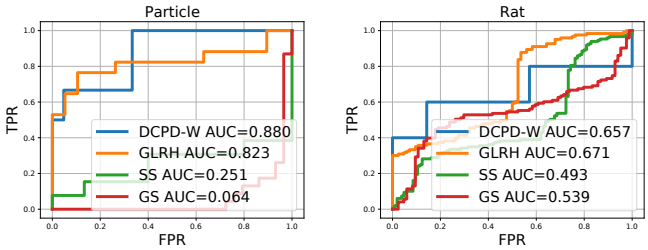


Figure 6: ROC curve of rest datasets for DCPD-W against different baselines.

and single change-point. Each sequence is generated using *SimuHawkesExpKernels* function of tick (Bacry et al., 2017) package, using decay as 1. If not mentioned otherwise, baselines are usually set between $\{1, 2, 3\}$ and coefficients are chosen from the range $[0, 1]$.

C.2 Additional synthetic experiments

Here we briefly describe additional synthetic experiments.

[Fig 5a, Fig 5b] Varying difference between distribution parameters around the change-point: Here we test our algorithms by varying the difference between model parameters before and after the change-point. First, we gradually increase the difference between the **baseline intensity** of the two distributions. Similarly, we gradually increase the difference between **influence strength coefficients** of the two distributions before and after the change-point. We see that in both cases DCPD-W and DCPD perform comparably with best performing GLRH with DCPD outperforming GLRH in some cases. Also, DCPD is more accurate in the case of increasingly different influence coefficients. It shows DCPD as a winner for a more challenging setting as it is more difficult to detect change based on the difference of influence strength coefficients than baseline intensities.

[Fig 5c] Varying dimention of distribution: Finally, we set the number of change-points to 2 and vary the dimension of generating the Hawkes process from 1 to 5. The result validates the comparatively better accuracy of DCPD in higher dimensions and comparable performance of DCPD-W throughout, confirming DCPD-W as an acceptable alternative of DCPD.

[Fig 5d] Checking scalability: Here we vary the number of events in sequence from 500 to 10000 and report the average time DCPD and DCPD-W take across 5 sequences on a logarithmic scale. We find DCPD-W to scale linearly with the number of events in contrast to poor scalability of DCPD.

C.3 Additional results on real-world datasets

Figure 6 presents ROC curve of DCPD-W against baselines for rest of the datasets (not mentioned in the main draft) for selected seeds. In general, DCPD-W continues to perform promising in comparison with baselines.